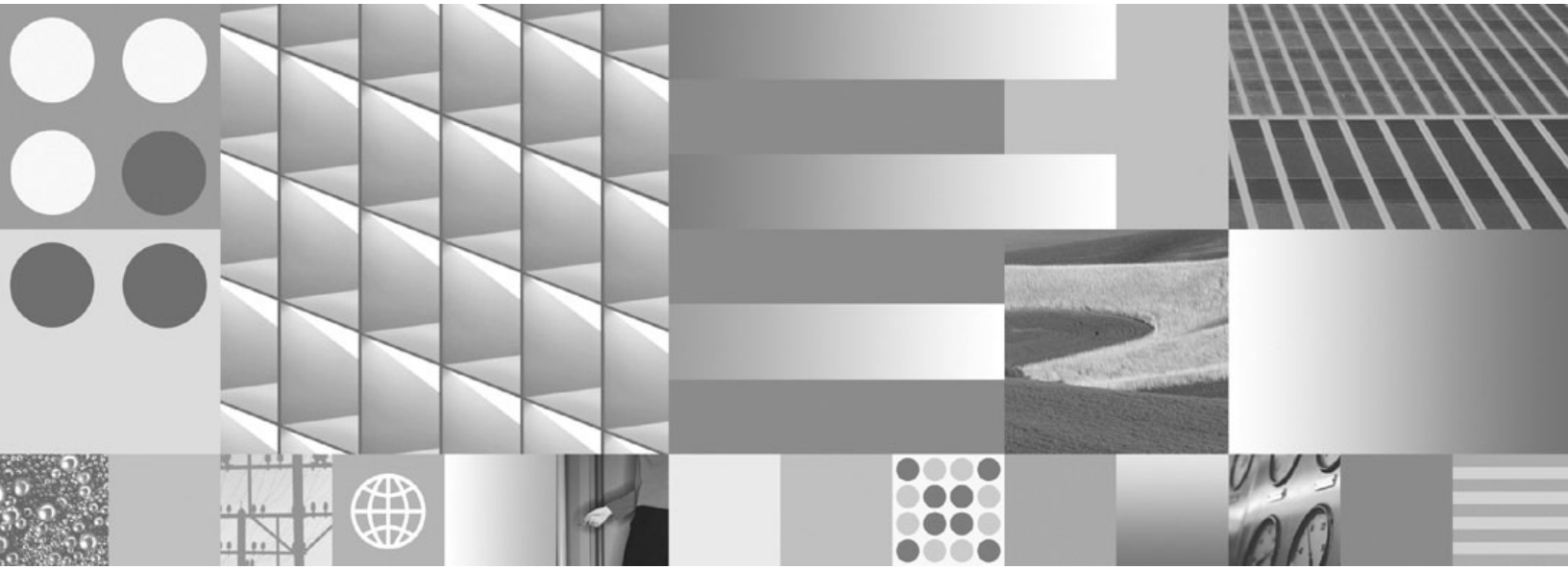
DB2 Version 9.5
for Linux, UNIX, and Windows

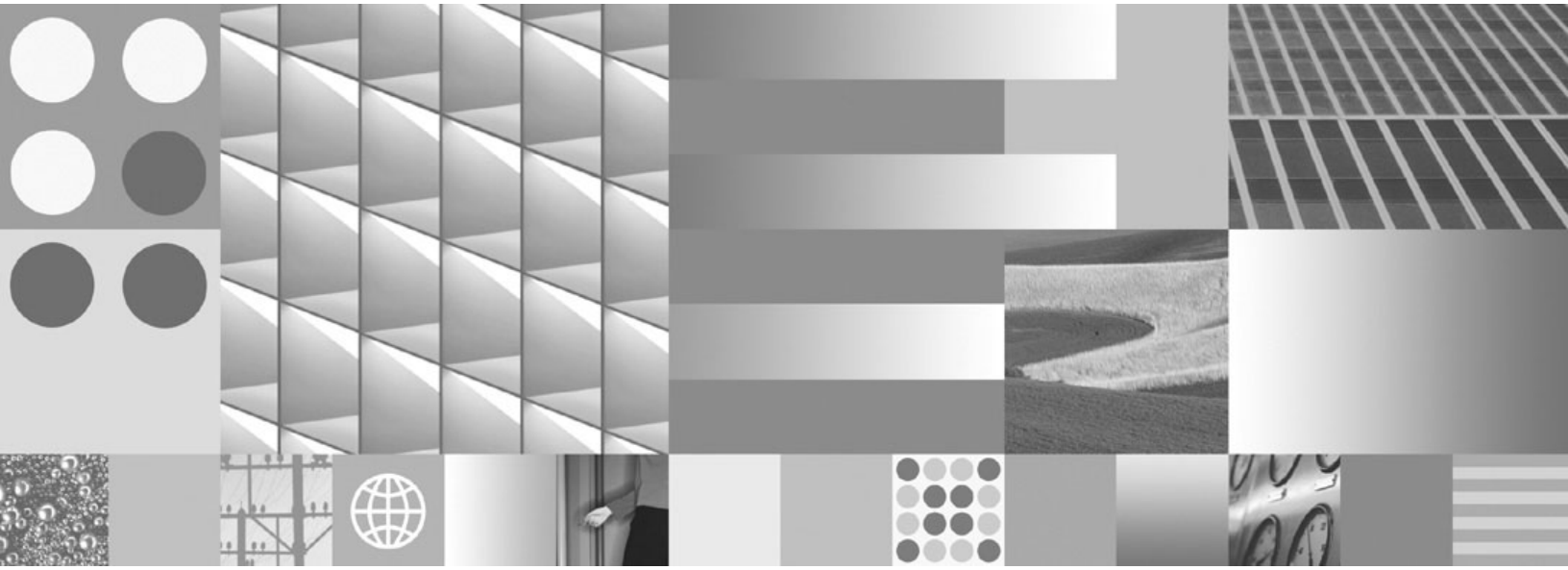


パーティションおよびクラスタリングのガイド

最終更新: 2009 年 4 月

SC88-4419-02
(英文原典：SC23-5860-02)

DB2 Version 9.5
for Linux, UNIX, and Windows



パーティションおよびクラスタリングのガイド

最終更新: 2009 年 4 月

SC88-4419-02
(英文原典：SC23-5860-02)

ご注意

本書および本書で紹介する製品をご使用になる前に、 485 ページの『付録 E. 特記事項』に記載されている情報をお読みください。

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

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

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

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

原典： SC23-5860-02
DB2 Version 9.5 for Linux, UNIX, and Windows
Partitioning and Clustering Guide
Updated April, 2009

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2009.3

目次

本書について

「パーティションおよびクラスタリングのガイド」にようこそ。

DB2® リレーショナル・データベース管理システムの機能は、パーティションおよびクラスタリング・フィーチャーによって大きな影響を受けます。これらによって管理者やシステム・オペレーターは、データベースのパフォーマンスを効果的に向上させ、複数のハードウェア・リソースの間で多数のデータベース・オブジェクトを分散させることができるようになります。高速のデータ・リトリーブ、および増大していく複数のハードウェア・リソースにわたってオブジェクトを分散させる機能によって、並列処理とストレージ容量を活用できるので、生産性が大きく向上します。本書には DB2 ライブラリーから編成した一連のトピックが含まれています。本書は、データベース・パーティション、表パーティション、表クラスター、表範囲クラスター、マルチディメンション・クラスタリング表、および並列処理の計画、設計、インプリメンテーション、使用、および保守に焦点を当てた包括的な情報を単一の情報源にまとめたものです。

本書の対象読者

本書は、ローカル・クライアントおよびリモート・クライアントがアクセスするパーティション・データベースやクラスター・データベースの設計、インプリメント、保守などを行う必要のあるデータベース管理者、システム管理者、セキュリティー管理者、およびシステム・オペレーターを主要な対象としています。また、本書は 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 (パーティション化、クラスタリング、および並列処理フィーチャーが関係する) に関する包括的な情報源および理解を必要とするアプリケーション開発者および他のユーザーにもご利用いただけます。本書で説明している主要なフィーチャーの一部またはすべてをインプリメンテーションすることを考慮中のユーザーにとっては、最適の情報源となります。

本書の構成

DB2 ライブラリーからトピックを収集することにより、DB2 パーティション、クラスタリング、および並列処理にのみ焦点を当てた包括的な情報を単一の情報源にまとめています。本書は便宜上また効率を考えて 6 つの主な部に分けられています。最初の 5 つの部では、管理者、システム・オペレーター、およびアプリケーション開発者にとって関心の対象となる主な管理テーマが示されています。本書の主な部に含まれるトピックは DB2 ライブラリーの他の文書の内容を表すテーマにマップすることができます。これにより、他の DB2 フィーチャーおよびオブジェクトのホストに関連するより一般的な情報への容易な相互参照を行うことができます。例えば、第 4 部、第 20 章にある、マルチディメンション・クラスタリング表の最適化作業がパフォーマンスの改善を示す方法に関するトピックを読んだ後、その特定の例のトピックがマップする先の「データベース・パフォーマンスのチューニング」というマニュアルを参照して、構成できる REGULAR 表の他の一般的パフォーマンスの向上を検討したいと思うかもしれません。以下の表 1 では、本書の主な部と、他の DB2 オブジェクトおよびフィーチャーに関する、同様のテーマを扱った

追加情報について参照できる他の資料との間のマッピングを示しています。

表 1. 本書の部と DB2 ライブラリーの他の資料とのマッピング

「パーティションとクラスタリングのガイド」の部	**DB2 ライブラリーの資料へのマッピング**
第 1 部 - 計画および設計上の考慮事項	*データ・サーバー、データベース、およびデータベース・オブジェクトのガイド* *データベース・セキュリティー・ガイド*
第 2 部 - インストールの注意点	*データ・サーバー、データベース、およびデータベース・オブジェクトのガイド* *DB2 サーバー機能 概説およびインストール*
第 3 部 - インプリメンテーションおよび保守	*データ移動ユーティリティー ガイドおよびリファレンス* *データ・リカバリーと高可用性 ガイドおよびリファレンス* *データ・サーバー、データベース、およびデータベース・オブジェクトのガイド* *マイグレーション・ガイド* *システム・モニター ガイドおよびリファレンス* *問題判別ガイド* *Visual Explain チュートリアル* *XQuery リファレンス*
第 4 部 - パフォーマンスの問題	*データ・サーバー、データベース、およびデータベース・オブジェクトのガイド* *データベース・パフォーマンスのチューニング* *Visual Explain チュートリアル*

表 1. 本書の部と DB2 ライブラリーの他の資料とのマッピング (続き)

「パーティションとクラスタリングのガイド」の部	DB2 ライブラリーの資料へのマッピング
第 5 部 - 管理 API、コマンド、SQL ステートメント	*管理 API リファレンス* *管理ルーチンおよびビュー* *コマンド・リファレンス* *ADO.NET および OLE DB アプリケーションの開発* *組み込み SQL アプリケーションの開発* *Java アプリケーションの開発* *Perl および PHP アプリケーションの開発* *SQL および外部ルーチンの開発* *データベース・アプリケーション開発の基礎* *SQL リファレンス 第 1 巻* *SQL リファレンス 第 2 巻*
第 6 部 - 付録	*データ・リカバリーと高可用性 ガイドおよびリファレンス* *DB2 サーバー機能 概説およびインストール* *SQL リファレンス 第 1 巻*

本書の章で説明されている主なサブジェクト・エリアは、以下のとおりです。

第 1 部 - 計画および設計上の考慮事項

以下のすべての章には、パーティション化、クラスタリング、または並列データベース・システムで使用されるデータベース/表の計画や設計に関係する概念的な情報が含まれています。

- 第 1 章『パーティション化されたデータベースおよび表』では、データベースおよび表のパーティション化のフィーチャーおよび利点に関する、関連した概念を紹介しています。
- 第 2 章『範囲クラスター表』では、範囲クラスター表のフィーチャーおよびそれを使用することの利点に関する一般的な概念的情報を提供しています。
- 第 3 章『マルチディメンション・クラスタリング (MDC) 表』では、マルチディメンション・クラスタリングを、表のデータのクラスタリングを実行する優れた方式として使用することについて説明しています。
- 第 4 章『並列データベース・システム』では、並列処理を活用してパフォーマンスをどれだけ著しく向上させることができるかを説明しています。

第 2 部 - インストールの注意点

以下の章では、データベース・パーティションのための準備に必要なプリインストール・タスクおよびインストール・タスクに関する情報が提供されています。

- 第 5 章『インストールの前提条件』では、パーティション・データベース環境に組み込まれる DB2 サーバーの準備に関する前提条件および制限について説明しています。
- 第 6 章『インストールする前に』では、UNIX® および Linux® システムの場合における、追加のプリインストール・タスクおよび考慮事項について説明しています。
- 第 7 章『DB2 サーバー製品のインストール』では、データベース・パーティション・サーバーのインストール方法およびパーティション・データベース環境のセットアップ方法について説明しています。
- 第 8 章『インストールした後に』では、Windows®、UNIX および Linux システムへのインストールの検証方法について説明しています。

第 3 部 - インプリメンテーションおよび保守

計画、設計、およびインストール手順が完了した後に参照する情報が含まれています。以下の章では、前の手順で準備が行われたフィーチャーやオブジェクトのインプリメントおよび保守を行う方法を説明しています。

- 第 9 章『データベース作成の前に』では、データベースの作成の前に考慮する必要がある事項について説明しています。これには、並列処理を使用可能にすること、パーティション・データベース環境の作成、ノード/パーティションの作成と構成、およびデータベース・パーティション間の通信の確立などが含まれます。
- 第 10 章『パーティション・データベース環境の作成と管理』では、データベース・パーティションおよびパーティション・グループの作成および管理方法について説明しています。
- 第 11 章『表およびその他の関連する表オブジェクトの作成』では、パーティション表、範囲クラスター表、および MDC 表の作成およびセットアップ方法に関する情報が提供されています。
- 第 12 章『データベースを変更する』では、インスタンスやデータベースを変更する方法について説明しています。
- 第 13 章『表およびその他の関連する表オブジェクトを変更する』では、パーティション表を変更する方法に関する情報が提供されています。
- 第 14 章『ロード』では、並列処理、マルチディメンション・クラスタリング、およびパーティション表の場合におけるロードの考慮事項について説明しています。
- 第 15 章『パーティション・データベース環境でのデータのロード』では、パーティション・データベース環境でデータのロード操作を開始、再開、再始動または終了する方法について説明しています。
- 第 16 章『パーティション・データベース環境のマイグレーション』では、パーティション・データベースのマイグレーションに関する概要を簡潔に説明しています。また、詳細情報に関する参照を提供しています。
- 第 17 章『スナップショット・モニターおよびイベント・モニターの使用』では、CREATE EVENT MONITOR ステートメントの使用方法の説

明に加えて、スナップショット・モニターの結果を使用して表の再編成をモニターしたり、パーティション・データベース・システムの全体的な状況を評価したりすることに関する情報が提供されています。

- 第 18 章『望ましいバックアップおよびリカバリー計画の作成』では、パーティション・データベース環境のクラッシュ・リカバリーに関する概念を説明しています。これは、障害が発生する前にバックアップおよびリカバリー計画を作成するために役立ちます。
- 第 19 章『トラブルシューティング』では、トラブルシューティングの簡潔な概要を説明しています。また、トラブルシューティングに役立つコマンド (db2trc など) を、インスタンス内のすべてのコンピューター間で、またはすべてのデータベース・パーティション・サーバーに対して発行する方法について役立つ情報も提供しています。

第 4 部 - パフォーマンスの問題

以下の章には、パーティション環境やクラスター環境のパフォーマンスを向上させるうえで役立つ関連情報が含まれています。

- 第 20 章『データベース設計でのパフォーマンスの問題』では、表パーティションおよびマルチディメンション・クラスタリングのパフォーマンス向上フィーチャーに関して説明しています。それぞれに関する最適化ストラテジーも含まれています。
- 第 21 章『索引』では、パーティション表の索引に関して理解するのに役立つ概念的な情報が提供されています。
- 第 22 章『設計アドバイザー』では、設計アドバイザーを使用して単一パーティション・データベースから複数パーティション・データベースへのマイグレーションに関する情報を取得する方法について説明しています。また、データの分散や、新規の索引、マテリアライズ照会表、およびマルチディメンション・クラスター表の作成に関する推奨事項も説明しています。
- 第 23 章『並行性の管理』では、ロック・モードに関する情報が提供されています。
- 第 24 章『エージェント管理』では、アプリケーション要求をサービスするために使用されるデータベース・エージェントを最適化する方法について説明しています。
- 第 25 章『アクセス・プランの最適化』では、アクセス・プランを改善する方法、オプティマイザーがさまざまなスキャンからの情報を使用してデータ・アクセス計画を最適化する方法について説明しています。結合ストラテジーに関する情報も含まれます。これらの情報はすべて、パーティション・データベース環境、クラスター表、および/または並列処理を使用するシステムでのパフォーマンスを向上させることを目的としています。
- 第 26 章『データの再分散』では、データ再配布を行う必要があるかどうかを決定するのに助けとなる情報を提供し、必要がある場合にデータをデータベース・パーティション間で再配布する方法について説明しています。
- 第 27 章『セルフチューニング・メモリーの構成』では、パーティション・データベース環境でのセルフチューニング・メモリー・フィーチャーの使用について説明しています。構成に関する推奨事項も記載しています。

- 第 28 章『DB2 構成パラメーターおよび変数』では、複数のパーティションにわたってデータベース構成パラメーターおよび環境変数を設定する方法についての情報を提供しています。また、パーティション・データベース環境および並列処理フィーチャーに関連したパラメーターおよび変数がリストされています。

第 5 部 - 管理 API、コマンド、SQL ステートメント

以下の章では、パーティション・データベース環境に関する管理 API、コマンド、および SQL エレメントについての情報を包括的にまとめて提供しています。

- 第 29 章『管理 API』では、パーティション・データベース環境にのみ関係する API に関する情報が提供されています。
- 第 30 章『コマンド』では、パーティション・データベース環境にのみ関係するコマンドに関する情報が提供されています。
- 第 31 章『SQL 言語エレメント』では、データベース・パーティションと互換性のあるデータ・タイプおよび特殊レジスターを示しています。
- 第 32 章『SQL 関数』では、パーティション・データベース環境にのみ関係する SQL 関数について説明しています。
- 第 33 章『SQL ステートメント』では、パーティション・データベース環境にのみ関係する SQL ステートメントについて説明しています。
- 第 34 章『サポートされる管理 SQL ルーチンおよび管理ビュー』では、パーティション・データベース環境にのみ関係する SQL ルーチンおよびビューについて説明しています。

第 6 部 - 付録

- 付録 A『非ルート・ユーザーとしてのインストール』では、DB2 製品を非ルート・ユーザーとして UNIX および Linux システムにインストールする方法について説明しています。
- 付録 B『バックアップの使用』では、BACKUP DATABASE コマンドの使用方法について説明しています。
- 付録 C『パーティション・データベース環境カタログ・ビュー』には、パーティション・データベース環境に固有のカタログ・ビューがリストされています。

強調表示規則

本書では、以下の強調表示規則を使用します。

太字	コマンド、キーワード、および名前がシステムによって事前定義されている他の項目を表します。
イタリック	以下のいずれかを示します。 • ユーザーが指定する必要がある名前または値 (変数) • 一般的な強調 • 初出の新規用語 • 別の情報ソースへの参照

モノスペース	以下のいずれかを示します。 • ファイルおよびディレクトリー • コマンド・プロンプトまたはウィンドウに入力するよう指示されている情報 • 特定のデータ値の例 • システムによって表示される内容に該当するテキストの例 • システム・メッセージの例 • プログラミング・コードのサンプル

第 1 部 計画および設計上の考慮事項

第 1 章 パーティション・データベースおよび表

パーティション・データベース環境のセットアップ

複数パーティション・データベースを作成するという決定は、データベースを作成する前に行わなければなりません。データベース設計に関して行う決定の一部として、データベース・パーティションが提供できるパフォーマンス改善を利用するかどうかを決定しておかなければなりません。

パーティション・データベース環境では、CREATE DATABASE コマンドまたは sqlecrea() 関数を使ってデータベースを作成します。どちらの方法を使う場合も、db2nodes.cfg ファイルにリストされたパーティションを通して要求を行うことができます。

複数パーティション・データベース作成前に、どのデータベース・パーティションをそのデータベースのカタログ・パーティションとするかを選択しなければなりません。その後、そのデータベース・パーティションからデータベースを直接作成するか、またはそのデータベース・パーティションにアタッチされたリモート・クライアントからデータベースを作成できます。アタッチして CREATE DATABASE コマンドを実行するデータベース・パーティションは、その特定のデータベースに対する*カタログ・パーティション* になります。

カタログ・パーティションは、すべてのシステム・カタログ表が保管されるデータベース・パーティションです。システム表に対するすべてのアクセスは、このデータベース・パーティションを通して行わなければなりません。すべてのフェデレーテッド・データベース・オブジェクト (ラッパー、サーバー、ニックネームなど) は、このデータベース・パーティションのシステム・カタログ表に保管されます。

可能であれば、各データベースを別個のインスタンスの中に作成してください。これが可能でない場合 (つまり、1 インスタンス当たり複数のデータベースを作成しなければならない場合)、カタログ・パーティションを使用可能なデータベース・パーティションに配分させる必要があります。これを行うと、単一データベース・パーティションにおけるカタログ情報の競合が削減されます。

注: 他のデータでバックアップに必要な時間が増えてしまうため、定期的にカタログ・パーティションのバックアップをとり、 (可能ならば) そこにユーザー・データを書き込むのを避けるべきです。

データベースを作成すると、 db2nodes.cfg ファイルに定義されたすべてのデータベース・パーティションにわたって自動的に作成されます。

システムに最初のデータベースが作成されると、システム・データベース・ディレクトリーが作成されます。これは、作成した他のデータベースについての情報と一緒に追加されます。 UNIX の場合、システム・データベース・ディレクトリーは sqldbdir であり、ホーム・ディレクトリーの下、または DB2 データベースのインストール・ディレクトリーの下の sqllib ディレクトリーに配置されます。 UNIX では、このディレクトリーは、パーティション・データベース環境を形成するすべて

のデータベース・パーティションに対する唯一のシステム・データベース・ディレクトリーであるため、共有ファイル・システム (例えば、UNIX プラットフォーム上の NFS) に常駐しなければなりません。 Windows の場合、システム・データベース・ディレクトリーはインスタンス・ディレクトリー内に置かれます。

さらに、sqldbdir ディレクトリーにはシステム・インテンション・ファイルも置かれます。これは sqldbins と呼ばれ、データベース・パーティションが同期を維持できるようにするものです。このファイルも、すべてのデータベース・パーティションにわたって 1 つのディレクトリーしかないため、共有ファイル・システムに常駐しなければなりません。このファイルは、データベースを形成するすべてのデータベース・パーティションで共用されます。

データベース・パーティションを利用するためには、構成パラメーターを修正しなければなりません。 GET DATABASE CONFIGURATION コマンドおよび GET DATABASE MANAGER CONFIGURATION コマンドを使用して、特定のデータベースまたはデータベース・マネージャー構成ファイルの中の個々の項目の値を調べることができます。特定のデータベースまたはデータベース・マネージャー構成ファイルの個々の項目を修正するためには、それぞれ UPDATE DATABASE CONFIGURATION コマンドと UPDATE DATABASE MANAGER CONFIGURATION コマンドを使用します。

パーティション・データベース環境に影響を与えるデータベース・マネージャー構成パラメーターには、**conn_elapse**、 **fcm_num_buffers**、 **fcm_num_channels**、**max_connretries**、**max_coordagents**、**max_time_diff**、**num_poolagents**、および **stop_start_time**があります。

複数のデータベース・パーティションにまたがるデータベース・パーティション化

データベース・マネージャーは、パーティション・データベースの複数のデータベース・パーティション (ノード) にまたがってデータを柔軟に拡散させることができます。ユーザーは、分散キーを宣言することによってデータを分散する方法を選択することができ、また、データを保管するデータベース・パーティション・グループおよび表スペースを選択することによって、いくつの、そしてどのデータベース・パーティションに表データを分散できるかを決定することができます。

さらに、分散マップ (更新可能) は、分散キー値のデータベース・パーティションへのマッピングを指定します。これにより、大きな表では 1 つのパーティション・データベース全体にまたがってワークロードを柔軟に均等化することができる一方、小さな表の場合はアプリケーション設計者の選択しだいで、1 つまたは少数のデータベース・パーティションに保管することもできます。ローカルの各データベース・パーティションに、保管するデータの索引も作成されるため、ローカル・データへのアクセス性能が向上します。

パーティション・データベースで、分散キーは一連のデータベース・パーティションに表データを分散するために使用されます。索引データも、それに対応する表とともにパーティション化され、各データベース・パーティションにローカル保管されます。

データベース・パーティションを使用してデータを保管するには、事前にパーティションをデータベース・マネージャーに対して定義しておく必要があります。データベース・パーティションは、db2nodes.cfg というファイルに定義されます。

パーティション・データベース・パーティション・グループの表スペースの表の分散キーは、CREATE TABLE ステートメント、または ALTER TABLE ステートメントに指定されます。指定されていない場合、デフォルト解釈によって、表の分散キーは、主キーの最初の列から作成されます。主キーが定義されていない場合、デフォルトの分散キーは、その表で定義されている、データ・タイプが long または LOB 以外の最初の列になります。パーティション・データベース内の表には、データ・タイプが long でも LOB でもない列が少なくとも 1 つは必要になります。単一パーティション・データベース・パーティション・グループの表スペースの表は、明示的に指定されている場合に限り、分散キーを持ちます。

行は、以下のようにデータベース・パーティション内に配置されます。

1. ハッシュ・アルゴリズム (データベース・パーティション機能) が分散キーのすべての列に適用され、その結果として分散マップの索引の値が生成されます。
2. 分散マップで、その索引の値にあるデータベース・パーティション番号は、行が保管されるデータベース・パーティションを識別します。

データベース・マネージャーは、*部分デクラスタリング* をサポートします。これは、システム内のデータベース・パーティションのサブセット (つまりデータベース・パーティション・グループ) に表を配分できることを意味しています。システム内のすべてのデータベース・パーティションにわたって表を配分する必要はありません。

データベース・マネージャーは、結合や副照会でアクセスされているデータが同じデータベース・パーティション・グループ内の同じデータベース・パーティションにある場合に、それを認識する能力を備えています。これを、*表コロケーション* といいます。同一の分散キー値を使用して連結されている表の行は、同一のデータベース・パーティションに置かれます。データベース・マネージャーは、データが保管されているデータベース・パーティションでの結合処理や副照会処理の実行を選択できます。これによって、大幅なパフォーマンスの改善が得られる場合もあります。

連結する表は、以下の条件を満たしている必要があります。

- 同一のデータベース・パーティション・グループにあり、再配分されていない。(再分散されると、データベース・パーティション・グループ内の表は別の分散マップを使用する可能性があります。このような表は連結されません。)
- 分散キーの列の数が同数である。
- 分散キーの対応する列に、データベース・パーティションの面で互換性がある。
- 同一のデータベース・パーティションに定義されている単一のパーティション・データベース・パーティション・グループにある。

パーティション・データベースの認証に関する考慮事項

パーティション・データベースでは、データベースの各区画に、同じ組のユーザーとグループが定義されていなければなりません。定義が同じでないと、ユーザー

は、異なる区画で異なることを実行できるように許可されてしまうことがあります。すべての区画にわたって一貫していることが推奨されます。

データベース・パーティション・グループ

データベース・パーティション・グループは、1 つのデータベースに属するものとして定義されている 1 つ以上のデータベース・パーティションのセットです。データベースに対して表を作成する場合、まず、表スペースが保管されるデータベース・パーティション・グループを作成した後、表が保管される表スペースを作成します。

1 つのデータベースの中に、 1 つ以上のデータベース・パーティションのサブセットを名前付きで定義することができます。定義する各サブセットをデータベース・パーティション・グループ と呼びます。複数のデータベース・パーティションが含まれる各サブセットを複数パーティションのデータベース・パーティション・グループ と呼びます。複数パーティションのデータベース・パーティション・グループは、同じインスタンスに属するデータベース・パーティションでのみ定義することができます。データベース・パーティション・グループには、1 つのデータベース・パーティションだけを含めることもできれば、データベース内のすべてのデータベース・パーティションにまで範囲を広げることもできます。

7 ページの図 1 は、5 つのデータベース・パーティションを持つデータベースの例を示したものであり、その中は以下のようになっています。

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

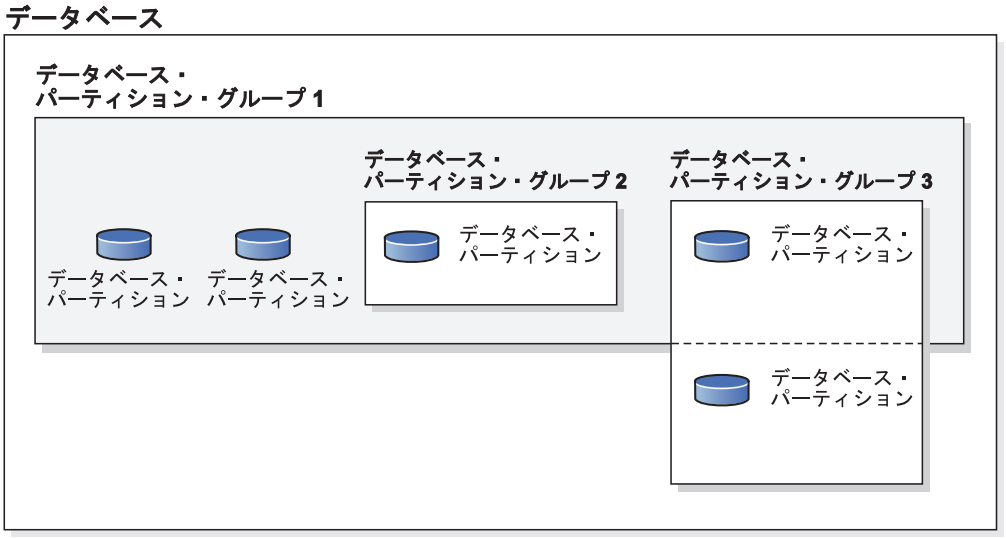


図 1. データベース内のデータベース・パーティション・グループ

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

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

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

非パーティション・データベースでは、分散キーおよび分散マップは必要ありません。データベース・パーティションとはデータベースの一部分であり、ユーザー・データ、索引、構成ファイル、およびトランザクション・ログで構成されます。データベースが作成されたときに作成されたデフォルトのデータベース・パーティション・グループは、データベース・マネージャーによって使用されます。 IBMCATGROUP は、システム・カタログが入っている表スペースのデフォルトのデータベース・パーティション・グループです。 IBMTEMPGROUP は、 SYSTEM TEMPORARY 表スペース用のデフォルトのデータベース・パーティション・グループです。 IBMDEFAULTGROUP は、そこに書き込むことのできる、ユーザー定義

の表が入る表スペース用のデフォルトのデータベース・パーティション・グループです。宣言済み一時表のための USER TEMPORARY 表スペースは、IBMDEFAULTGROUP または任意のユーザー作成のデータベース・パーティション・グループの中に作成できますが、 IBMTEMPGROUP の中には作成できません。

データベース・パーティション・グループを処理するときには、以下のことを実行できます。

- データベース・パーティション・グループを作成する。
- データベース・パーティション・グループに関連したコメントを変更する。
- データベース・パーティション・グループにデータベース・パーティションを追加する。
- データベース・パーティション・グループからデータベース・パーティションをドロップする。
- データベース・パーティション・グループ内の表データを再配分する。

データベース・パーティション・グループの設計

単一パーティション・データベースを使用している場合には、データベース・パーティション・グループについての設計上の考慮事項はありません。 DB2 設計アドバイザーは、データベース・パーティション・グループを推奨するために使用できるツールです。 DB2 設計アドバイザーは、コントロール・センターから、あるいはコマンド行プロセッサーから db2advis を使用して利用できます。

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

- 複数パーティションのデータベース・パーティション・グループでは、分散キーのスーパーセットである場合にのみ、ユニーク索引を作成することができます。
- データベース内のデータベース・パーティションの数によっては、 1 つ以上の単一パーティションのデータベース・パーティション・グループ、および 1 つ以上の複数パーティションのデータベース・パーティション・グループを作成できます。
- 各データベース・パーティションには固有の番号を割り当てる必要があります。同じデータベース・パーティションが 1 つ以上のデータベース・パーティション・グループに存在することもできます。
- システム・カタログ表を含んでいるデータベース・パーティションを速やかにリカバリーさせるためには、同じデータベース・パーティションにユーザー表を入れないようにしてください。こうするには、 IBMCATGROUP データベース・パーティション・グループのデータベース・パーティションを含まないデータベース・パーティション・グループの中に、ユーザー表を入れます。

小さな表は、大きな表とのコロケーション を利用する場合を除き、単一パーティションのデータベース・パーティション・グループの中に置いてください。コロケーションとは、同じデータベース・パーティションにある、関連データが入った複数の異なる表からの行を配置することです。連結された表を使用して、 DB2 Database for Linux, UNIX, and Windows 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。連結された表は、1 つの単一パーティションのデータベース・パーティション・グループの中に存在することができます。複数の表が 1 つの複数パーティ

ションのデータベース・パーティション・グループの中に存在し、分散キーの中に同数の列を持ち、対応する列のデータ・タイプに互換性がある場合、それらの表は連結されていると見なされます。同じ分散キー値を持つ連結された表の中の行は、同じデータベース・パーティションに置かれます。それぞれの表が同じデータベース・パーティション・グループの中の別個の表スペースの中に入っていても、連結されていると見なされます。

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

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

分散マップ

パーティション・データベース環境で、データベース・マネージャーは、必要とするデータの場所を認識していなければなりません。データベース・マネージャーは、データを見付けるために*分散マップ* というマップを使用します。

分散マップは内部で生成された配列であり、複数パーティションのデータベース・パーティション・グループの場合は 4 096 項目が、単一パーティションのデータベース・パーティション・グループの場合は単一の項目が入っています。単一パーティションのデータベース・パーティション・グループの場合、分散マップの項目は 1 つのみで、そこには、データベース表のすべての行が保管されているデータベース・パーティションの番号が入っています。複数パーティションのデータベース・パーティション・グループの場合、データベース・パーティション・グループの番号は、各データベース・パーティションが次々と使用されていき、それがマップ全体にわたって均等に配分されるような方法で指定されます。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、*分散キー* を使用して、データが保管されているロケーション (データベース・パーティション) を判別します。

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

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

データベース・パーティションの 1 および 2 を使用してデータベース・パーティション・グループがデータベース内に作成されている場合、そのデータベース・パーティション・グループの分散マップは、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分散キーが 1 と 500 000 の間の有効な値を持つ整数である場合、分散キーは、0 と 4 095 の間の番号になるようにハッシュが行われます。この番号は、その行のデータベース・パーティションを選択するための分散マップの索引として使用されます。

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

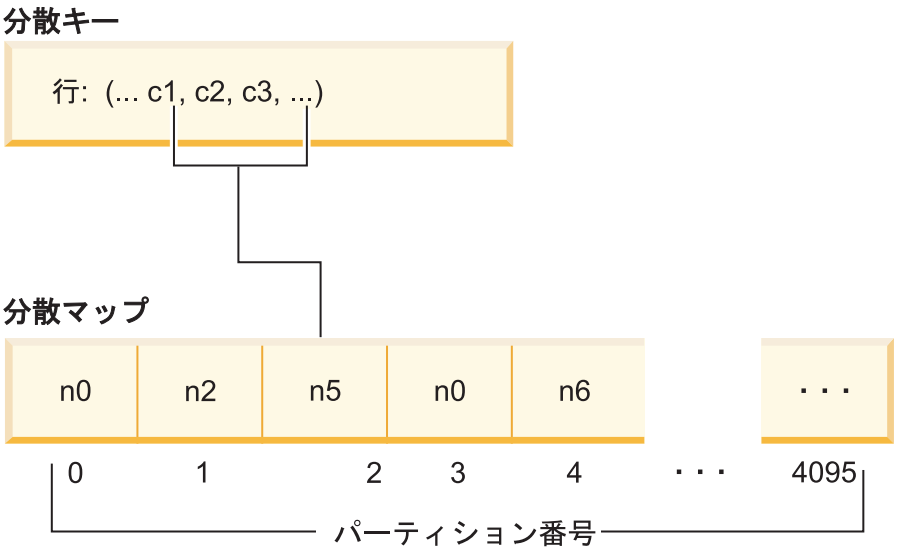


図 2. 分散マップを使用したデータ分散

分散マップは、複数パーティション・データベースのどこにデータが保管されるかを制御するための柔軟性のある手段です。データベース内のデータベース・パーティションにわたるデータ配分を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。

sqlugtpi API を使用して、見ることができる分散マップのコピーを入手することができます。

分散キー

分散キーとは、特定のデータ行を保管するデータベース・パーティションの判別に使用する列 (または列のグループ) のことです。分散キーは、CREATE TABLE ステートメントを使用して表の上に定義されます。データベース・パーティション・グループ内の複数のデータベース・パーティションにわたって分割された表スペース内の表に対して分散キーが定義されていない場合、デフォルトによって、分散キーが主キーの最初の列から作成されます。

主キーが指定されていない場合は、デフォルトの分散キーは、その表に定義された最初のロング・フィールド列以外の列になります。 (*長形式* には、すべてのロング・データ・タイプとすべてのラージ・オブジェクト (LOB) データ・タイプが含まれます。) 単一パーティション・データベース・パーティション・グループに関連した表スペースの中に表を作成している場合、分散キーが必要であれば、分散キーを明示して定義しなければなりません。デフォルトでは、分散キーは作成されません。

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

適切な分散キーを選択することが重要です。以下の点を考慮してください。

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

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

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

これらの特性により、同じ分散キー値を持つ連結された表の各行は、確実に同じデータベース・パーティションに配置されるようになります。

分散キーが不適切であると、データの分散が不均一になる可能性があります。不均一に分配されたデータを持つ列、および異なる値の数が少ない列は、分散キーとして選択するべきではありません。異なる値の数は、データベース・パーティション・グループ内のすべてのデータベース・パーティションにわたって行を均等に配分するのに十分な大きさでなければなりません。分散アルゴリズムを適用するためのコストは、分散キーのサイズに比例します。分散キーは 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、分散キーの中に含めるべきではありません。

分散キーを定義する場合には、以下の点を考慮する必要があります。

- ロング・データ・タイプ (LONG VARCHAR、 LONG VARGRAPHIC、 BLOB、CLOB、または DBCLOB) のみを含む複数パーティションの表を作成することはできません。
- 分散キーの定義は変更できません。
- 分散キーには、最も頻繁に結合される列を含める必要があります。
- 分散キーは、GROUP BY 節に頻繁に関与している列で構成する必要があります。
- どのユニーク・キーまたは主キーにも、すべての分散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、分散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。例えば、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 の適切な単一列分散キーとして使用できるでしょう。

データベース・パーティション とは、表内の各行の配置を決定する方式です。この方式は、以下のような仕組みです。

1. ハッシュ・アルゴリズムが、分散キーの値に適用され、ゼロ (0) と 4095 の間の番号を生成します。
2. データベース・パーティション・グループが作成されるときに、分散マップが作成されます。番号のそれぞれは、分散マップを充てんするために、ラウンドロビン方式で順番に繰り返されます。
3. 番号は、分散マップへの索引として使用されます。分散マップの中のそのロケーションにある番号は、その行が保管されているデータベース・パーティションの番号になります。

表のコロケーション

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

表は、同じデータベース・パーティション・グループ内に保管されており、それらの分散キーが互換性がある場合に連結 (collocate) されます。両方の表を同じデータベース・パーティション・グループに置くことによって、共通の分散マップにすることができます。これらの表は異なる表スペースに入れることは可能ですが、その表スペースは同じデータベース・パーティション・グループに関連付けられていなければなりません。各分散キーの中の対応する列のデータ・タイプは、*パーティション互換* でなければなりません。

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

パーティションの互換性

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

パーティションの互換性は、以下の特性を持ちます。

- ある基本データ・タイプは、同じ基本データ・タイプの別のものと互換性があります。
- 内部形式は、DATE、TIME、および TIMESTAMP データ・タイプに使用されます。これらは相互に互換性はなく、どれも文字データ・タイプまたはグラフィック・データ・タイプとの互換性はありません。
- パーティションの互換性は、列の NULL 可能性の影響を受けません。
- パーティションの互換性は、照合の影響を受けます。ロケールに依存する UCA ベースの照合は、照合の強さ属性が無視される以外、照合のときに完全な一致を必要とします。他のすべての照合は、パーティションの互換性を判別する目的で同等とみなされます。
- ロケールに依存する UCA ベースの照合以外の照合が使用されるとき、FOR BIT DATA で定義される文字列は、FOR BIT DATA のない文字列とのみ互換性があります。
- 互換データ・タイプの NULL 値は、同一のものとして扱われます (非互換データ・タイプの NULL 値はそのように扱われません)。
- 「ユーザー定義タイプ」という基本データ・タイプは、パーティションの互換性を分析するために使用されます。
- 分散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。
- ロケールに依存する UCA ベースの照合が使用されるとき、CHAR、VARCHAR、GRAPHIC、および VARGRAPHIC は互換データ・タイプです。その他の照合が使用されるときは、異なる長さの CHAR と VARCHAR が互換タイプ、GRAPHIC と VARGRAPHIC が互換タイプですが、CHAR と VARCHAR は GRAPHIC と VARGRAPHIC とは互換タイプではありません。
- LONG VARCHAR、LONG VARGRAPHIC、CLOB、DBCLOB、および BLOB データ・タイプは分散キーとしてサポートされないため、パーティションの互換性はこれらには適用されません。

パーティション表

表パーティション機能は、DB2 バージョン 9.1 Enterprise Server Edition for Linux, UNIX, and Windows 以降で使用できます。パーティション化された表は、表の 1 つ以上の表パーティション・キー列の値に従って、表データが、データ・パーティションまたは*範囲*と呼ばれる複数のストレージ・オブジェクトに分割されるデータ編成スキームを使用します。

データ・パーティションまたは範囲は表の行のサブセットを含む表の部分で、他の行のセットとは別に保管されます。指定された表のデータは、CREATE TABLE ステートメントの PARTITION BY 節で提供された仕様に基づいて、複数のデータ・オブジェクトにパーティション化されます。このデータ・パーティションまたは範

囲は異なる表スペース、同じ表スペース内、またはその両方に配置することができます。表が PARTITION BY 節を使用して作成されると、表はパーティション化されます。

指定されたすべての表スペースは、同一のページ・サイズ、エクステント・サイズ、ストレージ・メカニズム (DMS、SMS)、およびタイプ (REGULAR または LARGE) でなければならず、すべての表スペースは同一のデータベース・パーティション・グループになければなりません。

パーティション化された表は、表データのロールインおよびロールアウトを単純化し、通常の表よりも非常に多くのデータを含めることができます。最大で 32767 のデータ・パーティションを持つパーティション化された表を作成することができます。データ・パーティションは、パーティション化された表に対して追加、アタッチ、デタッチすることができ、表からの複数のデータ・パーティション範囲を 1 つの表スペースに保管することができます。

制限: パーティション化された階層または一時表、範囲クラスター表、およびパーティション化されたビューは、パーティション化された表での使用にはサポートされません。パーティション化された表では、XML 列タイプの使用もサポートされません。

表パーティション化

表パーティション化は、1 つ以上の表列の値に従って、表データがデータ・パーティションまたは*範囲*と呼ばれる複数のストレージ・オブジェクトに分割されるデータ編成スキームです。各データ・パーティションは別々に保管されます。このストレージ・オブジェクトは異なる表スペース、同じ表スペース内、またはその両方に配置することができます。

記憶オブジェクトは個々の表と同様の動作をします。このため、ALTER TABLE ...ATTACH ステートメントを使用して既存の表をパーティション表に取り込むことによって、高速なロールインを容易に行うことができます。同様に、ALTER TABLE ...DETACH ステートメントを使用して容易にロールアウトを行うことができます。また、照会処理はデータ分離を利用して無関係のデータのスキャンを避けることができます。これにより、多くのデータウェアハウス・スタイルの照会の照会パフォーマンスを向上させることができます。

表データは、CREATE TABLE ステートメントの PARTITION BY 節の指定にしたがってパーティション化されます。この定義で使用される列は、表パーティション・キー列と呼ばれます。

編成スキームは単独で使用することも、他の編成スキームとともに使用することもできます。CREATE TABLE ステートメントの DISTRIBUTE BY 節と PARTITION BY 節を結合することによって、データを複数の表スペースにまたがるデータベース・パーティションに広げることができます。編成スキームには、以下のものが含まれます。

- DISTRIBUTE BY HASH
- PARTITION BY RANGE
- ORGANIZE BY DIMENSIONS

表パーティション機能は、DB2 バージョン 9.1 Enterprise Server Edition for Linux, UNIX, and Windows 以降で使用できます。

表パーティションの利点

以下の状況のいずれかが自分自身や自分の組織に当てはまる場合、表パーティションの数多くの利点について考慮してください。

- 表データのロールインやロールアウトを容易にすることによって便利になるデータウェアハウスがある
- 大規模な表が含まれるデータウェアハウスがある
- 以前のリリースまたは競合データベース製品から バージョン 9.1 データベースへのマイグレーションを考慮している。
- 階層型ストレージ管理 (HSM) ソリューションをより効果的に使用する必要がある

表パーティションによって、容易に表データのロールインやロールアウトができ、管理も容易で、索引を柔軟に配置し、照会処理をより効果的に行うことができます。

効率的なロールインおよびロールアウト

表パーティション化は、表データの効率的なロールインおよびロールアウトを可能にします。ALTER TABLE ステートメントの ATTACH PARTITION 節および DETACH PARTITION 節の使用により、これを実現できます。パーティション表データをロールインすることによって、新しい範囲を追加のデータ・パーティションとして簡単にパーティション表に取り込むことができます。パーティション表データのロールアウトによって、その後のパージまたはアーカイブのためにデータの範囲を簡単にパーティション表から分けることができます。

大規模な表の容易な管理

表レベルの管理は、個々のデータ・パーティションに対して管理用タスクを実行できるので、より柔軟です。これらのタスクには、データ・パーティションのデタッチおよび再アタッチ、個々のデータ・パーティションのバックアップおよびリストア、および個々の索引の再編成が含まれます。一連のより細かい操作に分けることによって、時間のかかる保守操作の時間を短くできます。例えば、複数のデータ・パーティションが異なる表スペースに配置されている場合、バックアップ操作は個々のデータ・パーティションごとに対して実行できます。つまり、パーティション表のデータ・パーティションを一度に 1 つバックアップすることができます。

柔軟な索引の配置

索引を異なる表スペースに配置して、索引配置をより細かく制御できるようになりました。この新しい設計のには、以下のような利点があります。

- 索引のドロップおよびオンラインの索引の作成に関するパフォーマンスが向上します。
- 表の各索引における表スペース特性に異なる値を使用できます (たとえば、スペースの使用効率をより良いものにするには、各索引でページ・サイズを異ならせるのが適切な場合があります)。

- 入出力競合を削減して、表の索引データに効率的な同時アクセスができます。
- 個々の索引をドロップすると、索引再編成を行わなくてもシステムがスペースをすぐに使用できます。
- 索引再編成を実行することを選択した場合、個々の索引を再編成することができます。

DMS 表スペースと SMS 表スペースは両方とも、表とは別の場所にある索引の使用をサポートしています。

ビジネス・インテリジェンス・スタイルの照会のパフォーマンスの改善
照会の処理が拡張され、照会の述部に基づいて自動的にデータ・パーティションを除去できるようになりました。*Data Partition Elimination* として知られるこの機能によって、多くの意思決定支援のための照会が向上します。

以下の例では、表 *customer* が作成されます。ここで、 l_shipdate >= '01/01/2006' および l_shipdate <= '03/31/2006' のある行は表スペース ts1 に格納され、l_shipdate >= '04/01/2006' および l_shipdate <= '06/30/2006' のある行は表スペース ts2 に格納され、以下同様となります。

```
CREATE TABLE customer (l_shipdate DATE, l_name CHAR(30))
IN ts1, ts2, ts3, ts4, ts5
PARTITION BY RANGE(l_shipdate) (STARTING FROM ('01/01/2006')
ENDING AT ('12/31/2006') EVERY (3 MONTHS))
```

データ・パーティションおよび範囲

パーティション化された表は、表の 1 つ以上の表パーティション・キー列の値に従って、表データが、データ・パーティションまたは*範囲*と呼ばれる複数のストレージ・オブジェクトに分割されるデータ編成スキームを使用します。データ・パーティションごとに指定された範囲は、表の作成時に自動または手動で生成することができます。

データ・パーティションは、DB2 ライブラリー全体でさまざまな方法で参照されます。以下のリストは、最も一般的な参照を表しています。

- DATAPARTITIONNAME は、作成時に指定された表のデータ・パーティションに割り当てられる永続名です。この列値は、SYSCAT.DATAPARTITIONS カタログ・ビューに保管されます。この名前は、アタッチまたはデタッチ操作では保存されません。
- DATAPARTITIONID は、作成時に指定された表のデータ・パーティションに割り当てられる永続 ID です。これは、指定された表で特定のデータ・パーティションを一意的に識別するのに使用されます。この ID は、アタッチまたはデタッチ操作では保存されません。この値は、システム生成のものであり、さまざまなユーティリティーからの出力で表示されることがあります。
- SEQNO は、表内の他のデータ・パーティション範囲との関連における特定のデータ・パーティション範囲の順序を示します。デタッチされたデータ・パーティションは、すべての可視のもの、およびアタッチされたデータ・パーティションの後ろにソートされます。

データ編成スキーム

表パーティションの概要にあるとおり、DB2 データベースでは 3 つのレベルのデータ編成スキームが提供されています。データを編成する方法を示すアルゴリズムを含むのは、CREATE TABLE ステートメントの 3 つの節です。

以下の 3 つの節は、相互に組み合わせて使用できるデータ編成のレベルを示しています。

- DISTRIBUTE BY。データ・パーティション間でデータを均等に配分します (照会内並列処理を可能にし、各データベース・パーティション間で負荷のバランスを取ります)(データベース・パーティション化)
- PARTITION BY。同一のデータ・パーティション内の単一ディメンションの類似値を持つ行をグループ化します (表パーティション化)
- ORGANIZE BY。同一の表範囲内で複数ディメンションの類似値を持つ行をグループ化します (マルチディメンション・クラスタリング)

この構文を使用すると節間の整合性を取ることができ、データ編成の将来のアルゴリズムに備えることもできます。こうした各節は単独で使用することも、他の節とともに使用することもできます。CREATE TABLE ステートメントの DISTRIBUTE BY 節と PARTITION BY 節を結合することによって、データを複数の表スペースにまたがるデータベース・パーティションに広げることができます。この方法では、Informix® Dynamic Server および Informix Extended Parallel Server ハイブリッド機能に対しても同様の動作が可能です。

1 つの表において、各データ編成スキームで使用されている複数の節を結合して、より高度なパーティション・スキームを作成することができます。例えば、 DB2 Database Partitioning Feature (DPF) は表パーティションに対して互換性があるだけでなく補完的でもあります。

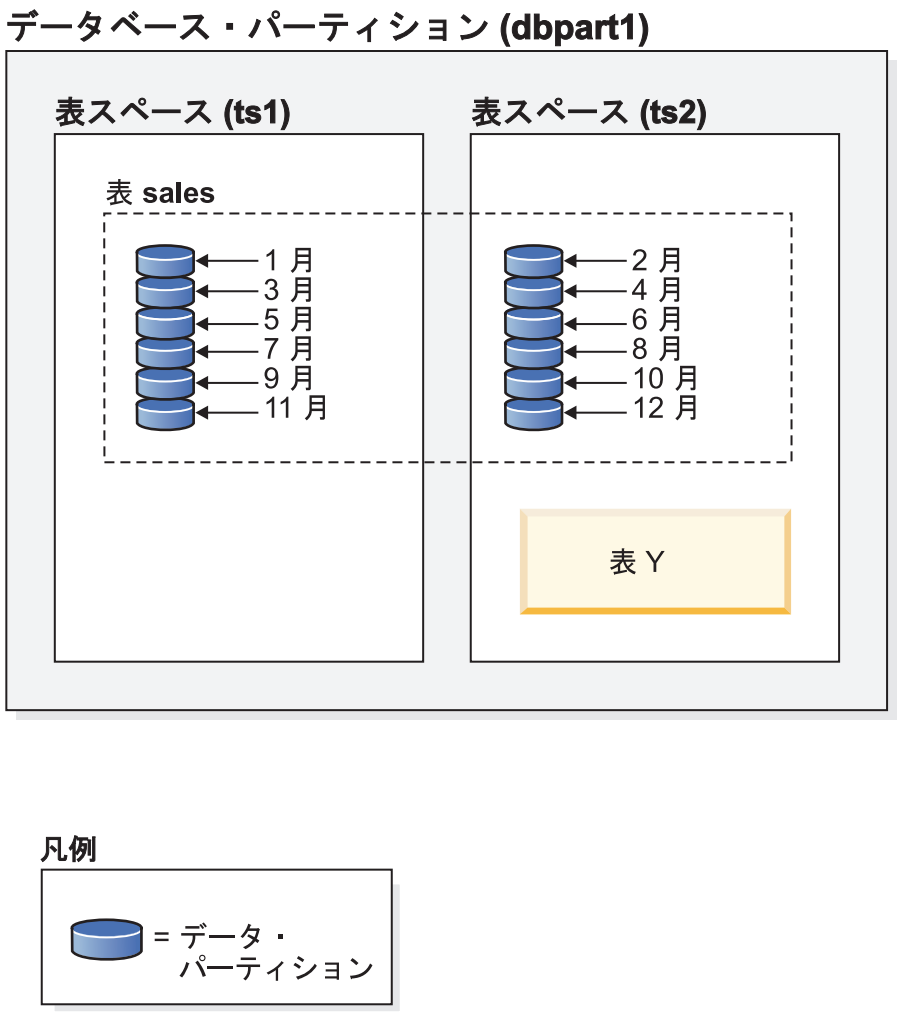


図 3. 月間売上高データを示す 1 つの表が複数のデータ・パーティションにパーティション化される表パーティション編成スキームを示しています。この表は、2 つの表スペースにまたがっています (ts1 と ts2)。

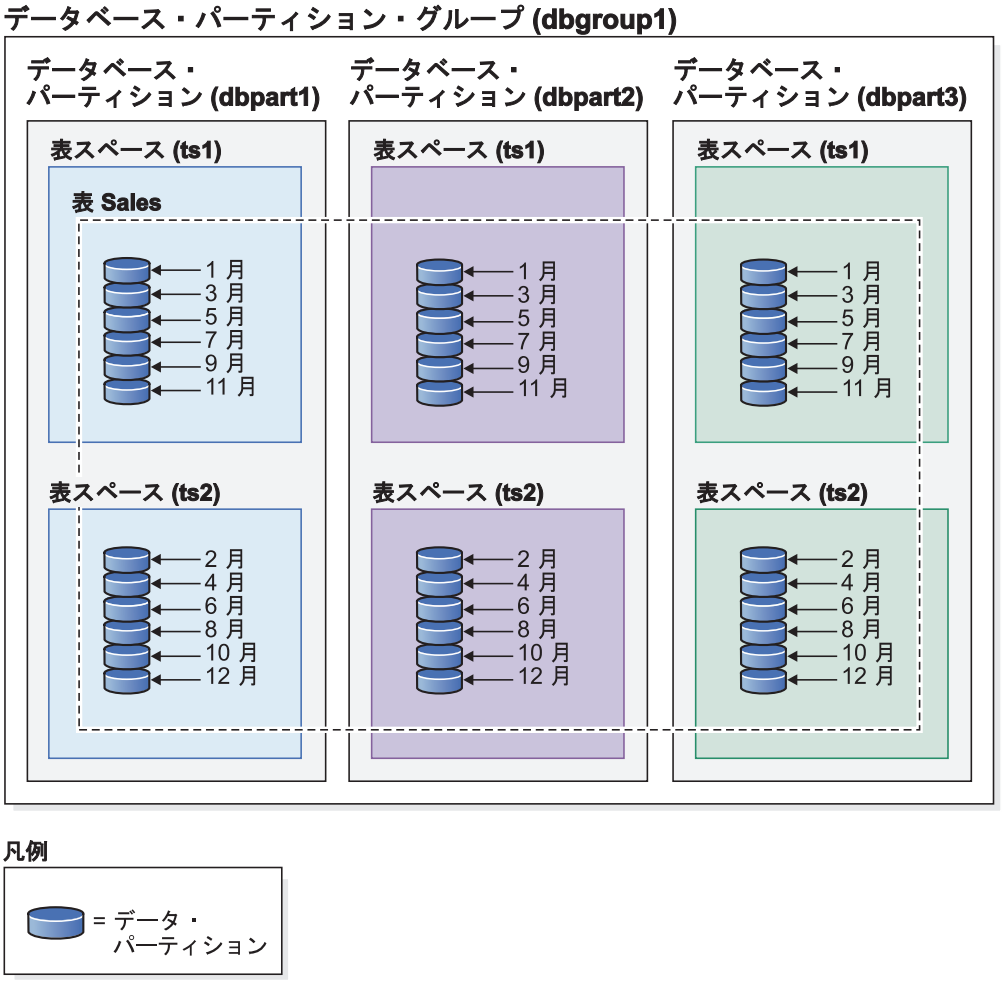


図 4. データベース・パーティションと表パーティションの補完的編成スキームを示しています。月間売上高データを示す 1 つの表は複数のデータ・パーティションにパーティション化されます。データは 2 つの表スペース (ts1 と ts2) にわたり、この 2 つの表スペースは、1 つのデータベース・パーティション・グループ (dbgroup1) の複数のデータベース・パーティション (dbpart1、dbpart2、dbpart3) に配分されます。

マルチディメンション・クラスタリング (MDC) と表パーティションの顕著な違いは、マルチ・ディメンションと単一ディメンションです。 MDC はキューブ (つまり複数のディメンションを有する複数の表) に適していますが、表パーティションは DATE 列などの、データベース設計の中心となる単一ディメンションが存在する場合に十分に機能します。 MDC と表パーティションは、こうした条件の両方が一致する場合に補完的になります。 20 ページの図 5 にそのことが示されています。

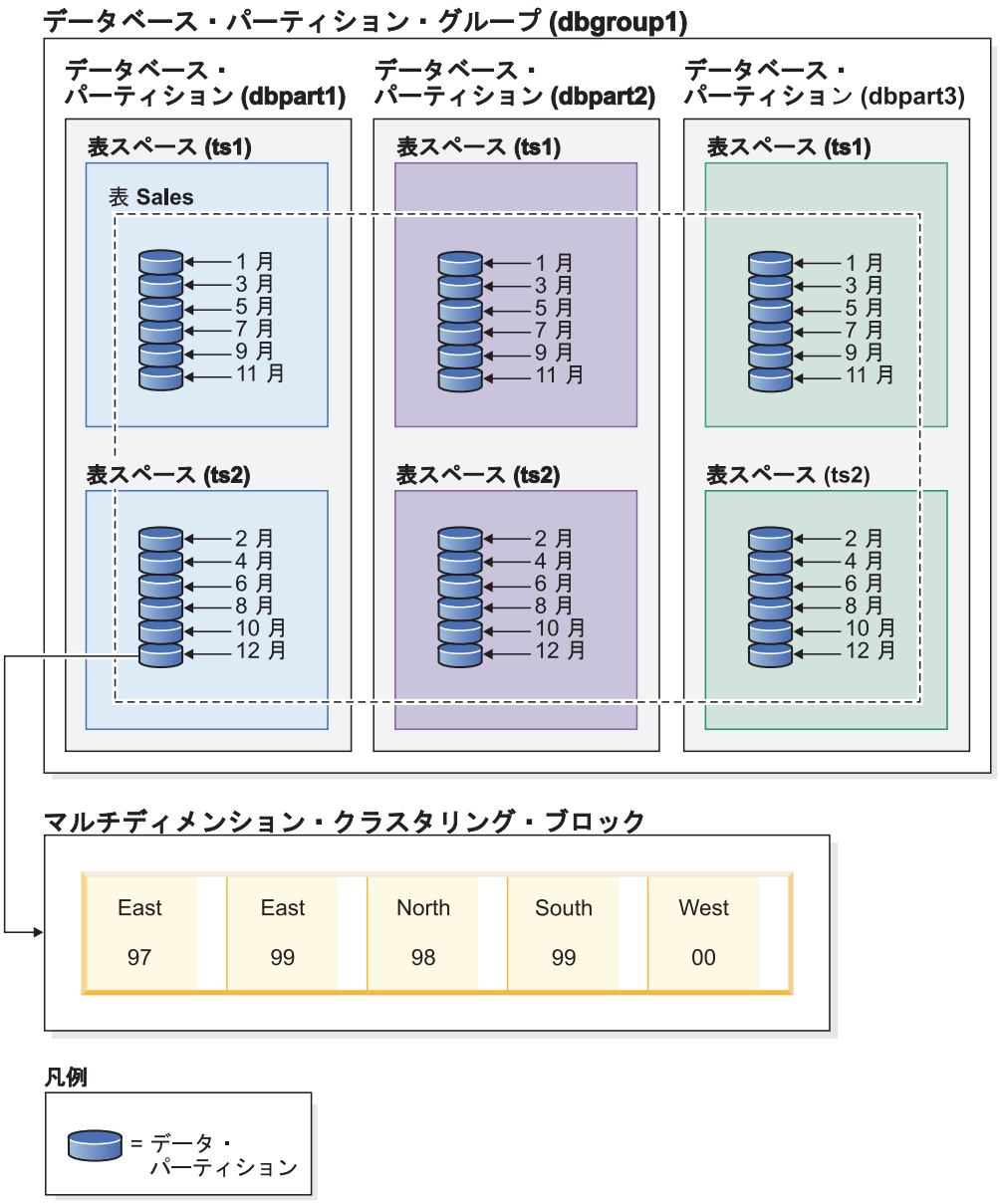


図 5. SALES 表からのデータが複数のデータベース・パーティションにまたがって配分され、表スペース ts1 と ts2 にパーティション化されるだけでなく、類似値を持つ行が Date ディメンションおよび Region ディメンションの両方にグループ化される、データベース・パーティション編成スキーム、表パーティション編成スキーム、およびマルチディメンション編成スキームの表示。

上にリストされているスキームとは同時には使用できないデータ編成スキームが他にもあります。そのスキームは ORGANIZE BY KEY SEQUENCE です。各レコードを、表の作成時 (範囲クラスター表) にそのレコード用に予約された行に挿入するために使用します。

データ編成用語

データベース・パーティション化

表の 1 つ以上の分散キー列にあるハッシュ値に従って、およびデータベース・パーティションの分散マップの使用に基づいて、表データが複数のデータベース・パーティションに分割されるデータ編成スキーマ。特定の表にあ

るデータは、CREATE TABLE ステートメントの DISTRIBUTE BY HASH 節にある指定に基づいて配分されます。

データベース・パーティション
データベース自体のユーザー・データ、索引、構成ファイル、およびトランザクション・ログからなる、データベース・パーティション・サーバー上のデータベースの一部。論理または物理データベース・パーティションのどちらも可能です。

表パーティション化
表の 1 つ以上のパーティション列にある値に従って表データを複数のデータ・パーティションに分割するデータ編成スキーム。指定された表のデータは、CREATE TABLE ステートメントの PARTITION BY 節で提供された仕様に基づいて、複数のストレージ・オブジェクトにパーティション化されます。これらのストレージ・オブジェクトは異なる表スペースに配置することができます。

データ・パーティション
他の行のセットとは別に保管され、CREATE TABLE ステートメントの PARTITION BY RANGE 節で提供された仕様によってグループ化された、表の行のセット。

マルチディメンション・クラスタリング (MDC)
ORGANIZE BY DIMENSIONS 節で指定された、1 つ以上のディメンション、またはクラスタリング・キーとともに、データが複数のブロックに物理的に編成された表。

各データ編成スキームの利点

各データ編成スキームの利点を理解すると、データベース・システム要件の計画、設計、または再評価時に最善の方法を判別するのに役立ちます。 表 2 には一般的なお客様の要件に関する高水準の概要が提供されていて、こうした要件をさまざまなデータ編成スキームがどのように満たすことができるかが示されています。

表 2. データベース・パーティション・フィーチャーを持つ表パーティションの使用

課題	推奨スキーム	説明
データのロールアウト	表パーティション化	中断を最小限にして最大量のデータをロールアウトするためにデタッチを使用します。
並列処理照会の実行 (照会パフォーマンス)	データベース・パーティション・フィーチャー	照会パフォーマンスを改良するため、照会並列処理を提供します。
データ・パーティションの除去 (照会パフォーマンス)	表パーティション化	照会パフォーマンスを改善するため、データ・パーティションの除去を提供します。
照会パフォーマンスの最大化	両方	両方とも使用すると照会パフォーマンスが最大になります。照会並列処理とデータ・パーティションの除去は補完的です。

表 2. データベース・パーティション・フィーチャーを持つ表パーティションの使用 (続き)

課題	推奨スキーム	説明
大量の管理者ワークロード	データベース・パーティション・フィーチャー	各データベース・パーティションで多くのタスクを実行します。

表 3. MDC 表での表パーティション化の使用

課題	推奨スキーム	説明
ロールアウト時のデータ可用性	表パーティション化	DETACH PARTITION 節を使用して、中断を最小限にしつつ、大量のデータをロールアウトします。
照会パフォーマンス	両方	MDC は複数ディメンションの照会に最善です。表パーティション化は、データ・パーティションの除去に有用です。
再編成の最小化	MDC	MDC はクラスタリングを維持し、再編成の必要性を削減します。

注: 表パーティション化は、現在、UNION ALL ビューに代わって推奨されています。

DB2 および Informix データベースにおけるデータ編成スキーム

表パーティション化は、1 つ以上の表列の値に従って、表データがデータ・パーティションまたは範囲と呼ばれる複数のストレージ・オブジェクトに分割されるデータ編成スキームです。各データ・パーティションは別々に保管されます。このストレージ・オブジェクトは異なる表スペース、同じ表スペース内、またはその両方に配置することができます。

表データは、CREATE TABLE ステートメントの PARTITION BY 節の指定にしたがってパーティション化されます。この定義で使用される列は、表パーティション・キー列と呼ばれます。DB2 の表パーティション化は、Informix Dynamic Server および Informix Extended Parallel Server によって提供される、データ編成に対するデータのフラグメント化アプローチにマップされます。

Informix アプローチ

Informix は幾つかのデータ編成スキームをサポートしていて、それらは Informix 製品では*フラグメント化* と呼ばれています。一般的なタイプのフラグメント化の 1 つは、FRAGMENT BY EXPRESSION です。このタイプのフラグメント化の働きは CASE ステートメントとよく似ており、表の各フラグメントに関連付けられた式があります。こうした式を検証して、行を配置する場所を判別します。

Informix データベース・システムと DB2 データベース・システムの比較

DB2 データベースは、Informix データ編成スキームに直接マップする補足的なフィーチャーのセットを豊富に提供しています。これにより、顧客にとって Informix 構文から DB2 構文に変換することが比較的容易になります。 DB2 データベース・マネージャーは、生成列と CREATE TABLE ステートメントの PARTITION BY RANGE 節の組み合わせを使用して、複雑な Informix スキームを処理します。 表 4 は、Informix データベース製品と DB2 データベース製品で使用されるデータ編成スキームを比較しています。

表 4. すべての Informix と DB2 データ編成スキームのマッピング

データ編成スキーム	Informix 構文	DB2 バージョン 9.1 構文
• Informix: 式ベース • DB2: 表パーティション化	FRAGMENT BY EXPRESSION	PARTITION BY RANGE
• Informix: ラウンドロビン • DB2: デフォルト	FRAGMENT BY ROUND ROBIN	構文なし: DB2 データベース・マネージャーがコンテナー間で自動的にデータを配分させる
• Informix: 範囲配分 • DB2: 表パーティション化	FRAGMENT BY RANGE	PARTITION BY RANGE
• Informix: システム定義ハッシュ • DB2: データベース・パーティション化	FRAGMENT BY HASH	DISTRIBUTE BY HASH
• Informix: HYBRID • DB2: 表パーティション化を使用したデータベース・パーティション化	FRAGMENT BY HYBRID	DISTRIBUTE BY HASH, PARTITION BY RANGE
• Informix: なし • DB2: マルチディメンション・クラスタリング	なし	ORGANIZE BY DIMENSION

例

以下の例では、式スキームを使用して DB2 データベースで Informix フラグメントと同等の結果を得る方法について詳細を示します。

例 1: 以下の基本的な CREATE TABLE ステートメントは、Informix フラグメント化と、それに相当する DB2 データベース・システム用の表パーティション化構文を示します。

Informix 構文:

```
CREATE TABLE demo(a INT) FRAGMENT BY EXPRESSION
 a = 1 IN db1,
 a = 2 IN db2,
 a = 3 IN db3;
```

DB2 構文:

```
CREATE TABLE demo(a INT) PARTITION BY RANGE(a)
 (STARTING(1) IN db1,
 STARTING(2) IN db2,
 STARTING(3) ENDING(3) IN db3);
```

Informix XPS は、ハイブリッドとして知られる 2 つのレベルのフラグメント化スキームをサポートします。これは、データが最初の式を使用して共通サーバー間で広げられ、2 番目の式を使用して共通サーバー内で広げられるものです。これによって、すべての共通サーバーが照会でアクティブになり (つまり、すべての共通サーバーにデータが存在する)、その照会はデータ・パーティションを対象外とする動作を活用することができます。

DB2 データベース・システムは、CREATE TABLE ステートメントの DISTRIBUTE BY 節および PARTITION BY 節の組み合わせを使用して、Informix ハイブリッドに相当する編成スキームを実現します。

*例 2:*以下の例は、結合された節の構文を示しています。

Informix 構文

```
CREATE TABLE demo(a INT, b INT) FRAGMENT BY HYBRID HASH(a)
 EXPRESSION b = 1 IN dbsl1,
   b = 2 IN dbsl2;
```

DB2 構文

```
CREATE TABLE demo(a INT, b INT) IN dbsl1, dbsl2
 DISTRIBUTE BY HASH(a),
 PARTITION BY RANGE(b) (STARTING 1 ENDING 2 EVERY 1);
```

さらに、マルチディメンション・クラスタリングを使用して、追加のレベルのデータ編成を実現することができます。

```
CREATE TABLE demo(a INT, b INT, c INT) IN dbsl1, dbsl2
 DISTRIBUTE BY HASH(a),
 PARTITION BY RANGE(b) (STARTING 1 ENDING 2 EVERY 1)
 ORGANIZE BY DIMENSIONS(c);
```

このように、列の同じ値 **a** を持つ行はすべて、同じデータベース・パーティションに配置されます。列の同じ値 **b** を持つ行はすべて、同じ表スペースに配置されます。指定された値 **a** および **b** に対して、同じ値 **c** を持つ行はすべて、ディスク上で一緒にクラスター化されます。このアプローチは OLAP タイプのドリルダウン操作に最適です。このタイプの照会を満足させるためにスキャンする必要があるのは、単一データベース・パーティションでの単一表スペース内の 1 つまたは幾つかのエクステント (ブロック) だけだからです。

共通のアプリケーション問題に適用される表パーティション化

以下のセクションでは、 DB2 表パーティション化のさまざまなフィーチャーを共通アプリケーション問題に適用する方法について説明します。各セクションでは、さまざまな Informix フラグメント化スキームを同等の DB2 表パーティション方式にマッピングするための最良事例に特別の注意が向けられます。

単純なデータ・パーティション範囲を作成する際の考慮事項

表パーティション化の最も一般的な適用の 1 つは、日付キーに基づいて大きなファクト表をパーティション化することです。均等なサイズの日付範囲を作成する必要がある場合は、CREATE TABLE 構文の自動生成フォームの使用を検討してください。

例

例 1: 以下の例は、構文の自動生成フォームを示しています。

```
CREATE TABLE orders
(
 l_orderkey  DECIMAL(10,0) NOT NULL,
 l_partkey  INTEGER,
 l_suppkey  INTEGER,
 l_linenumber  INTEGER,
 l_quantity  DECIMAL(12,2),
 l_extendedprice  DECIMAL(12,2),
 l_discount  DECIMAL(12,2),
 l_tax   DECIMAL(12,2),
 l_returnflag  CHAR(1),
 l_linestatus  CHAR(1),
 l_shipdate  DATE,
 l_commitdate  DATE,
 l_receiptdate  DATE,
 l_shipinstruct  CHAR(25),
 l_shipmode  CHAR(10),
 l_comment  VARCHAR(44))
  PARTITION BY RANGE(l_shipdate)
  (STARTING '1/1/1992' ENDING '12/31/1993' EVERY 1 MONTH);
```

これにより、24 の範囲 (1992 年から 1993 年の各月に 1 つ) が作成されます。l_shipdate がその範囲外である行を挿入しようとすると、エラーが発生します。

例 2: 先の例と以下の Informix 構文を比較してください。

```
create table orders
(
 l_orderkey  decimal(10,0) not null,
 l_partkey  integer,
 l_suppkey  integer,
 l_linenumber integer,
 l_quantity  decimal(12,2),
 l_extendedprice decimal(12,2),
 l_discount  decimal(12,2),
 l_tax   decimal(12,2),
 l_returnflag char(1),
 l_linestatus char(1),
 l_shipdate  date,
 l_commitdate date,
 l_receiptdate date,
 l_shipinstruct char(25),
 l_shipmode  char(10),
 l_comment  varchar(44)
) fragment by expression
l_shipdate < '1992-02-01' in ldbs1,
l_shipdate >= '1992-02-01' and l_shipdate < '1992-03-01' in ldbs2,
l_shipdate >= '1992-03-01' and l_shipdate < '1992-04-01' in ldbs3,
l_shipdate >= '1992-04-01' and l_shipdate < '1992-05-01' in ldbs4,
l_shipdate >= '1992-05-01' and l_shipdate < '1992-06-01' in ldbs5,
l_shipdate >= '1992-06-01' and l_shipdate < '1992-07-01' in ldbs6,
l_shipdate >= '1992-07-01' and l_shipdate < '1992-08-01' in ldbs7,
l_shipdate >= '1992-08-01' and l_shipdate < '1992-09-01' in ldbs8,
```

```
l_shipdate >= '1992-09-01' and l_shipdate < '1992-10-01' in ldbs9,
l_shipdate >= '1992-10-01' and l_shipdate < '1992-11-01' in ldbs10,
l_shipdate >= '1992-11-01' and l_shipdate < '1992-12-01' in ldbs11,
l_shipdate >= '1992-12-01' and l_shipdate < '1993-01-01' in ldbs12,
l_shipdate >= '1993-01-01' and l_shipdate < '1993-02-01' in ldbs13,
l_shipdate >= '1993-02-01' and l_shipdate < '1993-03-01' in ldbs14,
l_shipdate >= '1993-03-01' and l_shipdate < '1993-04-01' in ldbs15,
l_shipdate >= '1993-04-01' and l_shipdate < '1993-05-01' in ldbs16,
l_shipdate >= '1993-05-01' and l_shipdate < '1993-06-01' in ldbs17,
l_shipdate >= '1993-06-01' and l_shipdate < '1993-07-01' in ldbs18,
l_shipdate >= '1993-07-01' and l_shipdate < '1993-08-01' in ldbs19,
l_shipdate >= '1993-08-01' and l_shipdate < '1993-09-01' in ldbs20,
l_shipdate >= '1993-09-01' and l_shipdate < '1993-10-01' in ldbs21,
l_shipdate >= '1993-10-01' and l_shipdate < '1993-11-01' in ldbs22,
l_shipdate >= '1993-11-01' and l_shipdate < '1993-12-01' in ldbs23,
l_shipdate >= '1993-12-01' and l_shipdate < '1994-01-01' in ldbs24,
l_shipdate >= '1994-01-01' in ldbs25;
```

Informix 構文は、予期される範囲にない日付をキャッチするために、上限および下限のない範囲を提供します。 DB2 構文は、MINVALUE および MAXVALUE を使用する範囲を追加することによって、Informix 構文と一致するように変更できます。

例 3: 以下の例では、例 1 を変更して Informix 構文の鏡映を成しています。

```
CREATE TABLE orders
(
 l_orderkey  DECIMAL(10,0) NOT NULL,
 l_partkey  INTEGER,
 l_suppkey  INTEGER,
 l_linenumber INTEGER,
 l_quantity  DECIMAL(12,2),
 l_extendedprice DECIMAL(12,2),
 l_discount  DECIMAL(12,2),
 l_tax   DECIMAL(12,2),
 l_returnflag CHAR(1),
 l_linestatus CHAR(1),
 l_shipdate  DATE,
 l_commitdate DATE,
 l_receiptdate DATE,
 l_shipinstruct CHAR(25),
 l_shipmode  CHAR(10),
 l_comment  VARCHAR(44)
) PARTITION BY RANGE(l_shipdate)
 (STARTING MINVALUE,
  STARTING '1/1/1992' ENDING '12/31/1993' EVERY 1 MONTH,
  ENDING MAXVALUE);
```

この技法を使用して、任意の日付を表に挿入することができます。

生成列を使用した式によるパーティション

DB2 データベースは、式によるパーティション化を直接サポートしていませんが、生成列でのパーティション化はサポートされ、同じ結果を実現することが可能です。

このアプローチを使用するかどうかを決定する前に、以下の使用ガイドラインを検討してください。

- 生成列は、物理ディスク・スペースを占有する実際の列です。生成列を使用する表は、やや大きくなる可能性があります。

- 列に対する生成列式の変更は、パーティション表がその列でパーティション化される場合、サポートされません。変更しようとすると、メッセージ SQL0190 が表示されます。一般的には、次のセクションで説明されている方法で、生成列を使用する表に新規のデータ・パーティションを追加するためには、生成列を定義する式を変更する必要があります。生成列を定義する式を変更することは、現在サポートされていません。
- 表が生成列を使用する際に、データ・パーティションの除去を適用する時期については制限があります。

例

例 1: 以下は Informix 構文を使用します。ここでは、生成列を使用するのが適当です。以下の例では、パーティション化される列はカナダの州および地域を保持します。州のリストが変更される可能性は低いため、生成列式が変更される可能性は低くなります。

```
CREATE TABLE customer (
 cust_id  INT,
 cust_prov CHAR(2))
 FRAGMENT BY EXPRESSION
  cust_prov = "AB" IN dbspace_ab
  cust_prov = "BC" IN dbspace_bc
  cust_prov = "MB" IN dbspace_mb
  ...
  cust_prov = "YT" IN dbspace_yt
  REMAINDER IN dbspace_remainder;
```

例 2: この例では、 DB2 表は、生成列を使用してパーティション化されます。

```
CREATE TABLE customer (
 cust_id     INT,
 cust_prov   CHAR(2),
 cust_prov_gen GENERATED ALWAYS AS (CASE
  WHEN cust_prov = 'AB' THEN 1
  WHEN cust_prov = 'BC' THEN 2
  WHEN cust_prov = 'MB' THEN 3
  ...
  WHEN cust_prov = 'YT' THEN 13
  ELSE 14 END))
 IN tbspace_ab, tbspace_bc, tbspace_mb, .... tbspace_remainder
 PARTITION BY RANGE (cust_prov_gen)
  (STARTING 1 ENDING 14 EVERY 1);
```

ここでは、CASE ステートメント内の式は、FRAGMENT BY EXPRESSION 節の対応する式と一致します。 CASE ステートメントは、それぞれの元の式を番号にマップし、その番号は生成列 (この例では、cust_prov_gen) に保管されます。この列は、ディスク上に保管される実際の列であるため、DB2 が式によってパーティションを直接サポートする場合に必要とするスペースよりも、少し多くのスペースが表によって占有される可能性があります。この例では、短い形式の構文を使用します。このため、データ・パーティションを配置する表スペースは、CREATE TABLE ステートメントの IN 節にリストされている必要があります。長い形式の構文を使用する場合は、各データ・パーティションごとに個別の IN 節が必要になります。

注: この技法は、すべての FRAGMENT BY EXPRESSION 節に適用することができます。

表パーティション・キー

表パーティション・キーとは、表の 1 つ以上の列の順序セットのことです。表パーティション・キー列の値は、各表の行が属するデータ・パーティションを決定するのに使用されます。

表で表パーティション・キーを定義するには、PARTITION BY 節のある CREATE TABLE ステートメントを使用します。

効率的な表パーティション・キー列を選択することは、表パーティションの利点を十分に活用する上で重要です。以下のガイドラインは、パーティション化された表に対して最も効率的な表パーティション・キー列を選択するのに役立ちます。

- データのロールイン・サイズと一致する範囲を定義する。日付列または時刻列でデータをパーティション化するのが最も一般的です。
- データのロールアウトと一致する範囲の細分度を定義する。月または四半期の細分度が最も一般的です。
- パーティションを対象外とする動作において効果が得られる列で、パーティション化する。

サポートされるデータ・タイプ

表 5 は、表パーティション・キー列としての使用をサポートされるデータ・タイプ (シノニムを含む) を示しています。

表 5. サポートされるデータ・タイプ

データ・タイプ列 1	データ・タイプ列 2
SMALLINT	INTEGER
INT	BIGINT
FLOAT	REAL
DOUBLE	DECIMAL
DEC	NUMERIC
NUM	CHARACTER
CHAR	VARCHAR
DATE	TIME
GRAPHIC	VARGRAPHIC
CHARACTER VARYING	TIMESTAMP
CHAR VARYING	CHARACTER FOR BIT DATA
CHAR FOR BIT DATA	VARCHAR FOR BIT DATA
CHARACTER VARYING FOR BIT DATA	CHAR VARYING FOR BIT DATA
ユーザー定義タイプ (特殊)	

サポートされないデータ・タイプ

以下のデータ・タイプはパーティション化された表に現れることはありますが、表パーティション・キー列としての使用はサポートされていません。

- ユーザー定義タイプ (構造化)
- LONG VARCHAR

- LONG VARCHAR FOR BIT DATA
- BLOB
- BINARY LARGE OBJECT
- CLOB
- CHARACTER LARGE OBJECT
- DBCLOB
- LONG VARGRAPHIC
- REF
- C 用の可変長ストリング
- Pascal 用の可変長ストリング

パーティション表では、XML データ・タイプはサポートされません。

CREATE TABLE ステートメントの EVERY 節を使用して、自動的にデータ・パーティションを生成することを選択した場合、表パーティション・キーとして使用できるのは 1 つの列だけです。 CREATE TABLE ステートメントの PARTITION BY 節でそれぞれの範囲を指定することにより、データ・パーティションを手動で生成することを選択した場合、以下の例で示されているように、複数の列を表パーティション・キーとして使用することができます。

```
CREATE TABLE sales (year INT, month INT)
   PARTITION BY RANGE(year, month)
   (STARTING FROM (2001, 1) ENDING (2001,3) IN tbsp1,
   ENDING (2001,6) IN tbsp2, ENDING (2001,9)
   IN tbsp3, ENDING (2001,12) IN tbsp4,
   ENDING (2002,3) IN tbsp5, ENDING (2002,6)
   IN tbsp6, ENDING (2002,9) IN tbsp7,
   ENDING (2002,12) IN tbsp8)
```

これにより、8 つのデータ・パーティションが作成されます (2001 年および 2002 年の四半期ごとに 1 つ)。

注:

1. 複数の列が表パーティション・キーとして使用される場合、後続の列は先行する列に従属しているという意味で、それらは複合キー (索引での複合キーと似ている) として扱われます。それぞれの開始値または終了値 (列のすべての合計) は、512 文字以下で指定される必要があります。この制限は、SYSCAT.DATAPARTITIONS カタログ・ビューの LOWVALUE 列および HIGHVALUE 列のサイズに対応します。指定された開始値または終了値が 512 文字を超えると、エラー SQL0636N、理由コード 9 が発生します。
2. 表パーティションは、マルチディメンションではなくマルチ列です。表パーティションでは、使用されるすべての列が単一ディメンションの一部になります。

生成列

生成列は、表パーティション・キーとして使用することができます。以下の例では、12 のデータ・パーティション (月ごとに 1 つ) を持つ表を作成します。すべての年の 1 月の行はすべて最初のデータ・パーティションに配置され、2 月の行は 2 番目のデータ・パーティションに配置され、以下同様です。

例 1

```
CREATE TABLE monthly_sales (sales_date date,
 sales_month int GENERATED ALWAYS AS (month(sales_date)))
     PARTITION BY RANGE (sales_month)
     (STARTING FROM 1 ENDING AT 12 EVERY 1);
```

注:

1. 表パーティション・キーで使用される生成列の式を変更したりドロップすることはできません。表パーティション・キーで使用される列の生成された列式を追加することは許可されていません。表パーティション・キーで使用される列の生成された列式を追加、ドロップ、または変更しようとすると、エラー (SQL0270N rc=52) が発生します。
2. 生成列が単調ではない場合、または生成列が単調であることをオプティマイザーが検出できない場合は、データ・パーティションを対象外とする動作は範囲述部には使用されません。単調ではない式が存在する場合、データ・パーティションの除去は、等式述部または IN 述部に対してのみ行われます。単調性に関する詳細な説明および例については、 56 ページの『MDC 表を作成する際の考慮事項』を参照してください。

パーティション表におけるロードの考慮事項

以下の一般制約事項を除き、既存のすべてのロード・フィーチャーはターゲット表がパーティション化されている場合にサポートされます。

- パーティション・エージェントが複数存在する場合、整合点はサポートされません。
- データ・パーティションのサブセットにデータをロードしている間、その他のデータ・パーティションを完全にオンラインのままにしておく機能はサポートされません。
- ロード操作で使用される例外表は、パーティション化できません。
- ロード・ユーティリティーが挿入モードまたは再始動モードで実行されていて、デタッチされた従属データがロード・ターゲット表にある場合には、ユニーク索引を再作成することはできません。
- MDC 表のロードと同様、入力データ・レコードの厳密な順序は、パーティション表をロードする際には保持されません。順序はセルまたはデータ・パーティションの中のみで維持されます。
- 各データベース・パーティションで複数のフォーマッターを使用するロード操作では、入力レコードの大まかな順序のみを保持します。各データベース・パーティション上で単一のフォーマッターを実行すると、入力レコードがセルまたは表パーティション・キーごとにグループ化されます。各データベース・パーティション上で単一のフォーマッターを実行するには、明示的に CPU_PARALLELISM に 1 を要求してください。

一般的なロードの動作

ロード・ユーティリティーは、データ・レコードを適切なデータ・パーティションに挿入します。ロードの前に入力データをパーティション化するための外部ユーティリティー (スプリッターなど) を使用する上での要件はありません。

ロード・ユーティリティーは、アタッチまたはデタッチされたデータ・パーティションにアクセスしません。データは可視のデータ・パーティションのみに挿入されます。可視のデータ・パーティションは、アタッチされたりデタッチされたりしま

せん。また、ロード置換操作では、アタッチまたはデタッチされたデータ・パーティションを切り捨てることはありません。ロード・ユーティリティーではカタログ・システム表上のロックを獲得するため、ロード・ユーティリティーはコミットされていない ALTER TABLE トランザクションがあれば待機します。そのようなトランザクションは、カタログ表内の関連する行の排他ロックを獲得します。排他ロックを終了しなければロード操作は進行できません。これは、ロード操作の実行中は、コミットされていない ALTER TABLE ...ATTACH、DETACH、または ADD PARTITION トランザクションはありえないということを意味します。アタッチまたはデタッチされたデータ・パーティションに宛てられたすべての入力ソース・レコードはリジェクトされ、例外表が指定されている場合にはそこから取得できます。ターゲット表データ・パーティションの一部がアタッチまたはデタッチされた状態であったことを示すため、通知メッセージがメッセージ・ファイルに書き込まれます。ターゲット表に対応するカタログ表の関連する行のロックは、ロード・ユーティリティーの実行中に ALTER TABLE ...ATTACH、DETACH、または ADD PARTITION 操作を実行することによりユーザーがターゲット表のパーティションを変更することを防ぎます。

無効な行の処理
ロード・ユーティリティーで可視のデータ・パーティションのいずれにも属さないレコードが検出されると、そのレコードはリジェクトされ、ロード・ユーティリティーは処理を継続します。範囲制約違反のためにリジェクトされたレコードの数は明示的には表示されませんが、リジェクトされたレコードの全体数には含められます。範囲違反のためにレコードをリジェクトしても行の警告数は増加しません。範囲違反が検出されたものの、レコードごとのメッセージはログに記録されないということを示す単一のメッセージ (SQL0327N) がロード・ユーティリティーのメッセージ・ファイルに書き込まれます。例外表には、ターゲット表のすべての列に加えて、特定の行で発生した違反のタイプを記述する列が含まれます。無効データを含む行 (パーティション化できないデータを含む) は、ダンプ・ファイルに書き込まれます。

例外表への挿入は非効率であるため、どの制約違反を例外表に挿入するかを制御できます。例えば、ロード・ユーティリティーのデフォルトの動作は、範囲制約違反またはユニーク制約違反のためにリジェクトされた (その違反がなければ有効だった) 行を例外表に挿入することです。この動作は、FOR EXCEPTION 節を使用し、NORANGEEXC (範囲制約違反の場合) または NOUNIQUEEXC (ユニーク制約違反の場合) を指定することによってオフにすることができます。それらの制約違反を例外表に挿入しないことを指定する場合、または例外表を指定しない場合、範囲制約またはユニーク制約に違反する行に関する情報は失われます。

履歴ファイル
ターゲット表がパーティション化されている場合、対応する履歴ファイルの項目は、ターゲット表により範囲を設定された表スペースのリストを含みません。操作対象のオブジェクト ID (「T」ではなく「R」) は、ロード操作がパーティション表に対して実行されたことを示します。

ロード操作の終了
ロード置換を終了すると、すべてのデータ・パーティションが完全に切り捨てられ、ロード挿入を終了すると、すべてのデータ・パーティションがロード前の長さに切り捨てられます。ロード・コピー・フェーズで失敗した ALLOW READ

ACCESS 操作の終了中に索引は無効になります。索引にタッチした ALLOW NO ACCESS ロード操作を終了する時にも索引は無効になります (それが無効になるのは、索引付けモードが再作成されているか増分保守の間にキーが挿入されたために索引が不整合状態になっているためです)。データを複数のターゲットにロードしても、ロード・フェーズ中に取られた整合点からロード操作を再始動できない点を除き、ロード・リカバリー操作に何の影響もありません。この場合、ターゲット表がパーティション化されている場合には、SAVECOUNT ロード・オプションは無視されます。この動作は、MDC ターゲット表へのデータのロードと一貫しています。

生成列
生成される列がパーティション・キー、ディメンション・キー、または分散キーのいずれかにある場合、 `generatedoverride` ファイル・タイプ修飾子は無視され、ロード・ユーティリティーは `generatedignore` ファイル・タイプ修飾子が指定された場合のように値を生成します。ここで不正な生成列値をロードすると、レコードが不適切な物理ロケーション (例えば不適切なデータ・パーティション、MDC ブロック、またはデータベース・パーティション) に配置されてしまう可能性があります。例えば、あるレコードがいったん間違ったデータ・パーティションに置かれると、整合性の設定ではそのレコードを別の物理ロケーションに移動しなければなりませんが、それはオンラインでの整合性の設定操作中には行えません。

データの可用性
現行の ALLOW READ ACCESS ロード・アルゴリズムは、パーティション表に拡張されています。ALLOW READ ACCESS ロード操作では、複数のリーダーが同時に表全体 (ロードするデータ・パーティションとロードしないものの両方を含む) にアクセスすることができます。

データ・パーティションの状態
ロードに成功した後、特定の条件下では、可視のデータ・パーティションの表の状態が「SET INTEGRITY ペンディング」または「読み取りアクセスのみ」のいずれかまたは両方に変更される場合があります。ロード操作で保守できない制約が表にある場合に、データ・パーティションはこれらの状態になる可能性があります。そのような制約には、チェック制約とデタッチされたマテリアライズ照会表が含まれる場合があります。ロード操作に失敗すると、すべての可視のデータ・パーティションの表の状態が「ロード・ペンディング」になります。

エラー分離
データ・パーティション・レベルでのエラー分離はサポートされていません。エラーを分離するとは、エラーにならなかったデータ・パーティションでロードを継続し、エラーになったデータ・パーティションで停止するということを意味します。エラーは異なるデータベース・パーティションの間で分離できますが、ロード・ユーティリティーは可視のデータ・パーティションのサブセット上でトランザクションをコミットしたり、残りの可視のデータ・パーティションをロールバックしたりすることはできません。

その他の考慮事項

- いずれかの索引が無効とマークされている場合には増分索引付けはサポートされません。索引の再作成が必要な場合、またはデタッチされた従属物が SET INTEGRITY ステートメントでの妥当性検査を必要としている場合には、索引は無効であると見なされます。

- 範囲別パーティション化、ハッシュによる分散、またはディメンションによる編成のいずれかのアルゴリズムの組み合わせを使用してパーティション化された表へのロードもサポートされています。
- ロードの影響を受けるオブジェクトと表スペース ID のリストが含まれるログ・レコードの場合、これらのログ・レコード (LOAD START および COMMIT (PENDING LIST)) のサイズは非常に大きくなる場合があり、そうなると、他のアプリケーションで使用できるアクティブ・ログ・スペースの量が減少してしまいます。
- 表がパーティション化されていて、かつ分散されている場合、パーティション・データベースのロードがすべてのデータベース・パーティションには影響を与えない場合があります。出力データベース・パーティション上のオブジェクトのみが変更されます。
- ロード操作中、パーティション表のメモリー使用量は表の数とともに増加します。増加の合計は直線的にならない点に注意してください。データ・パーティションの数に比例するのはメモリー所要量全体のうちのほんの僅かだからです。

複製されたマテリアライズ照会表

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

パーティション・データベース環境では、マテリアライズ照会表を複製し、それ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。*複製されたマテリアライズ照会表* の基になっているのは、単一パーティションのデータベース・パーティション・グループで作成された可能性があるが、別のデータベース・パーティション・グループ内のすべてのデータベース・パーティションにわたって複製しようとしている表です。マテリアライズ照会表を作成するには、 REPLICATED キーワードを指定して CREATE TABLE ステートメントを呼び出します。

複製されたマテリアライズ照会表を使用することによって、一般的には連結されない表間でのコロケーションを実現できます。複製されたマテリアライズ照会表は、1 つの大きいファクト表と小さいディメンション表のある結合について特に役立ちます。必要とされるストレージの拡張を最小限にし、すべてのレプリカを更新する影響を最小化するためには、複製する表は小さく、更新頻度の低いものでなければなりません。

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

複製表の定義に使われる副選択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。

データベース・パーティション・グループの表スペース

複数パーティションを持つデータベース・パーティション・グループの中に表スペースを置くことによって、その表スペース内のすべての表が、そのデータベース・パーティション・グループ内の各データベース・パーティションにわたって分割、つまりパーティション化されます。

表スペースはデータベース・パーティション・グループ内に作成されます。いったん 1 つのデータベース・パーティション・グループ内に入ると、表スペースは、そこにとどまらなければならず、別のデータベース・パーティション・グループに変更することはできません。 CREATE TABLESPACE ステートメントは、表スペースとデータベース・パーティション・グループを関連付けるために使用されます。

表パーティション化とマルチディメンション・クラスタリング表

表では、マルチディメンション・クラスタリングとパーティション化の両方ができます。マルチディメンション・クラスタリングとパーティション化の両方を行った表では、列は表パーティション化の範囲パーティション仕様と MDC キーの両方で使用されます。これは、どちらかが単独で機能するよりも、データ・パーティションの細分性を良くし、ブロックを除去するのに役立ちます。また、表がパーティション化されるよりも、MDC キーに異なる複数の列を指定することが役立つ多くのアプリケーションがあります。なお、MDC はマルチディメンションですが、表パーティション化は複数列であることに注意してください。

主流の DB2 V9.1 データウェアハウスの特性

以下の推奨事項は、DB2 V9.1 にとって新しい、典型的で主流となるウェアハウスに焦点を合わせています。以下のような特性があると想定されています。

- データベースは、複数のマシンまたは複数の AIX 論理パーティションで実行される。
- データベース・パーティション・フィーチャー (DPF) が使用される (表は DISTRIBUTE BY HASH 節を使用して作成される)。
- 4 から 50 個以内のデータ・パーティションがある。
- MDC および表パーティション化が考慮されている表が、主なファクト表である。
- 表には 1 億から 1000 億以内の行がある。
- 新規データは、毎夜、毎週、毎月といった様々な時間フレームでロードされる。
- 毎日の取得ボリュームは、1 万から 1000 万以内のレコードである。
- データ・ボリュームが変化する。最大月は最小月のサイズの 5 倍になります。同様に、最大ディメンション (製品ライン、地域) は 5 倍のサイズ範囲となります。
- 1 から 5 年分の詳細データが保存される。
- 有効期限切れデータは、毎月または四半期ごとにロールアウトされる。
- 表は、広範囲の照会タイプを使用する。しかし、ワークロードはほとんど、OLTP ワークロードに比べると、以下の特性を持つ分析照会です。
  - 200 万行までの、より大きな結果セット
  - ほとんどまたはすべての照会が、基本表ではなくビューをヒットする

- 範囲 (BETWEEN 節)、リストにある項目などでデータを選択する SQL 節。

主流の DB2 V9.1 データウェアハウスのファクト表の特性

典型的なウェアハウスのファクト表は、以下の設計を使用すると考えられます。

- 月列にデータ・パーティションを作成する。
- ロールアウトする期間 (たとえば 1 カ月、3 カ月) ごとに、データ・パーティションを定義する。
- 日および 1 から 4 つ以内の追加のディメンション上に MDC ディメンションを作成する。典型的なディメンションは、製品ラインおよび地域です。
- すべてのデータ・パーティションおよび MDC クラスターが、すべてのデータベース・パーティションに広がっている。

MDC および表パーティション化は、重複した利点のセットを提供します。以下の表では、お客様の組織で必要になる可能性のあるものをリストし、以前に識別された特性を基にして、推奨される編成スキームを識別します。

表 6. MDC 表での表パーティション化の使用

課題	推奨スキーム	推奨
ロールアウト時のデータ可用性	表パーティション化	DETACH PARTITION 節を使用して、中断を最小限にしつつ、大量のデータをロールアウトします。
照会パフォーマンス	表パーティション化および MDC	MDC は複数ディメンションの照会に最善です。表パーティション化は、データ・パーティションの除去に有用です。
再編成の最小化	MDC	MDC はクラスタリングを維持し、再編成の必要性を削減します。
従来のオフライン期間内での、1 カ月かそれ以上のデータのロールアウト	表パーティション化	データ・パーティション化はこの必要を完全に処理します。 MDC は何も追加することはなく、これ自体にはあまり適していません。
マイクロ・オフライン期間 (1 分より小さい) 内での、1 カ月かそれ以上のデータのロールアウト	表パーティション化	データ・パーティション化はこの必要を完全に処理します。 MDC は何も追加することはなく、これ自体にはあまり適していません。
少しもサービスを損失することなく、照会をサブミットするビジネス・ユーザーが、表を完全に使用できるように保ちながら、1 カ月かそれ以上のデータをロールアウトする	MDC	MDC だけが、この必要をいくらか処理します。表パーティション化は、表が短期間オフラインになるので、適切ではありません。

表 6. MDC 表での表パーティション化の使用 (続き)

課題	推奨スキーム	推奨
毎日のデータのロード (ALLOW READ ACCESS または ALLOW NO ACCESS)	表パーティション化および MDC	この場合、MDC はほとんどの利点を提供します。表パーティション化は増分的な利点を提供します。
「継続的な」データのロード (ALLOW READ ACCESS)	表パーティション化および MDC	この場合、MDC はほとんどの利点を提供します。表パーティション化は増分的な利点を提供します。
「従来の BI」照会の場合の照会実行パフォーマンス	表パーティション化および MDC	MDC は、キューブ/複数ディメンションの照会に最適です。表パーティション化は、パーティションの除去の点で補助します。
再編成の必要を避けたり、タスクの実行に関連した手間を削減することにより、再編成の手間を最小化にする	MDC	MDC はクラスタリングを維持して REORG の必要性を削減します。 MDC が使用される場合、データ・パーティション化は増分的な利点を提供しません。しかし、MDC が使用されない場合には、範囲パーティション化が、パーティション・レベルで何らかの過程の粗いクラスタリングを維持することによって、REORG の必要を削減するのに役立ちます。

例 1:

キー列 YearAndMonth および Province のある表を考慮します。この表のプランとして妥当な方法は、1 つのデータ・パーティションあたり 2 カ月で、日付によってパーティション化することです。加えて、 37 ページの図 6 に示されているように、任意の 2 カ月の日付範囲内にある特定の州のすべての行は一緒にクラスター化されるので、Province によって編成することもできます。

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

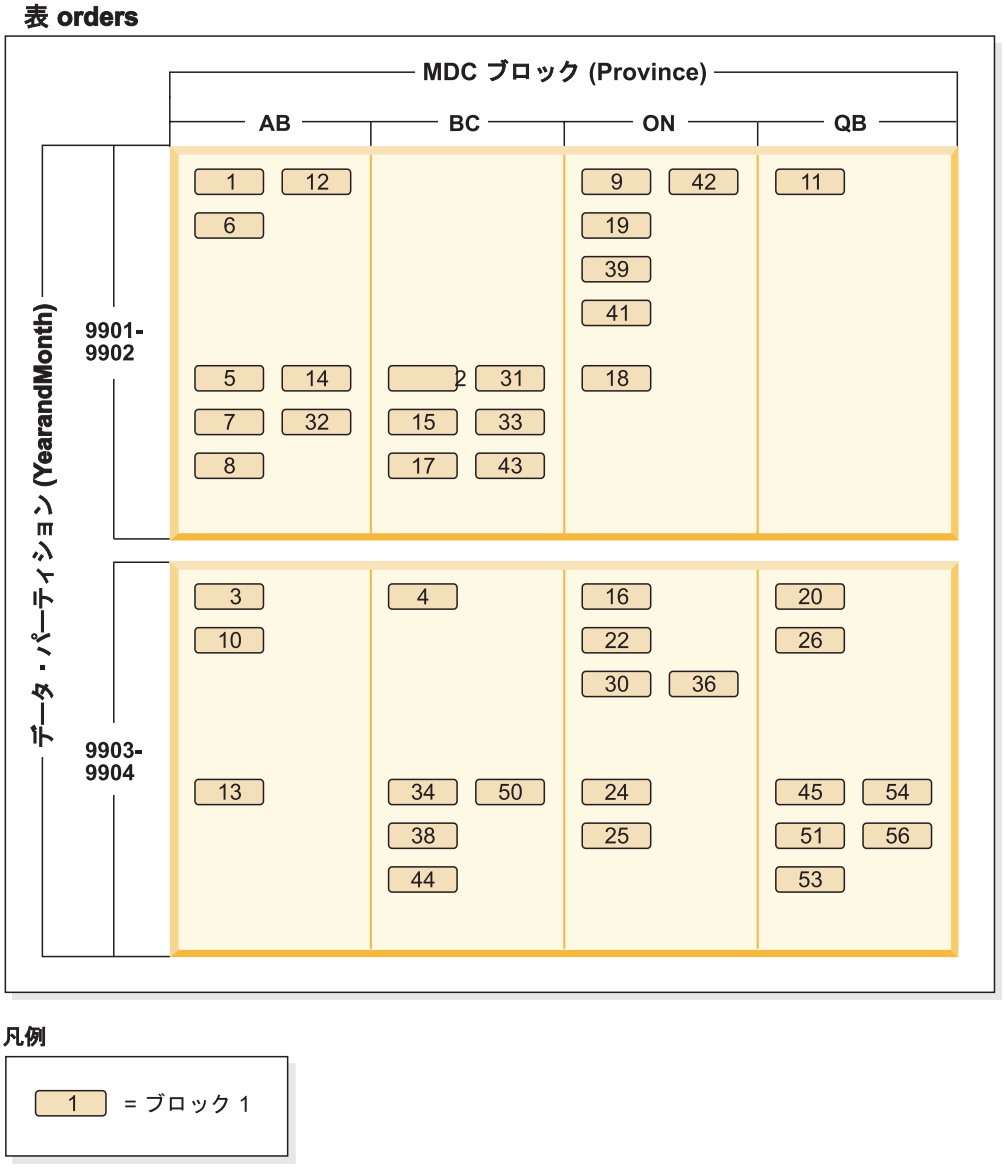


図 6. YearAndMonth によってパーティション化され、Province によって編成される表

例 2:

38 ページの図 7 に示されているように、YearAndMonth を ORGANIZE BY 節に追加することによって、より良い細分化を行うことができます。

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

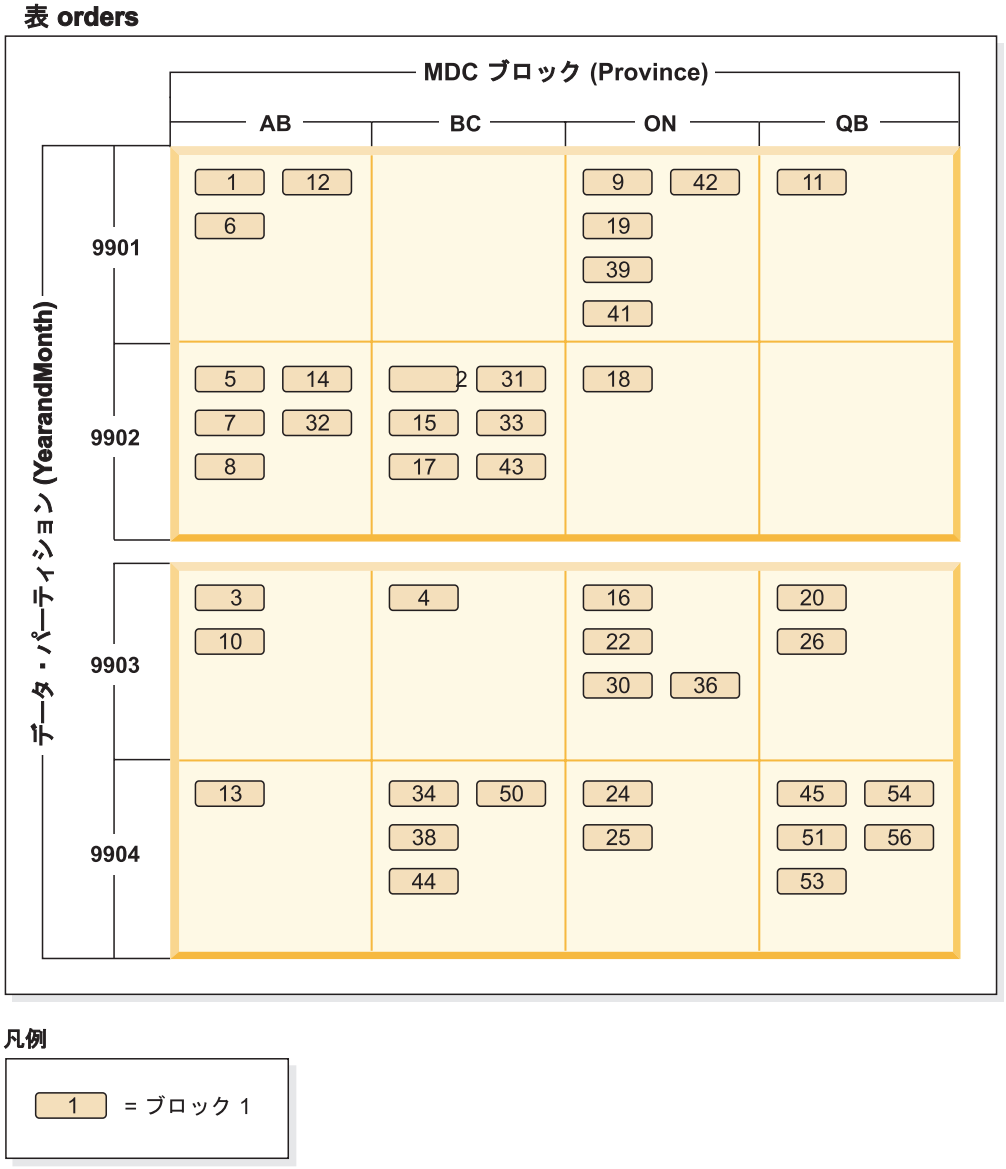


図 7. YearAndMonth によってパーティション化され、Province および YearAndMonth によって編成される表

各範囲に単一値のみがあるようなパーティション化の場合、MDC キーに表パーティション列を含めても、何も得られません。

考慮事項

- 基本表と比較して、MDC 表およびパーティション表は多くのストレージを必要とします。これらのストレージ必要量は付加的なものですが、利点を考えると妥当なものと考えられます。
- パーティション・データベース環境で表パーティション化と MDC 機能を組み合わせないことを選択するなら、確信をもってデータ配分を予測できるような場合 (一般的にここで説明されているシステムのタイプの場合) には、表パーティション化が最善です。そうでない場合には、MDC を考慮する必要があります。

第 2 章 範囲クラスター表

範囲クラスター表 (RCT) は、表の中でレコードを探索するために内部で使用される ID であるレコード ID (RID) が事前定義されている、表のレイアウト体系です。

データを保持する各表について、使用できるどの表タイプが要件に対して最適かを考慮してください。例えば、緩くクラスター化されている (単調増加しない) データ・レコードがある場合は、通常表と索引を使用することを考慮してください。キー内に重複する (固有でない) 値を持つデータ・レコードがある場合は、範囲クラスター表を使用しないでください。使用する範囲クラスター表に一定量のディスク・ストレージを事前割り当てする余裕がない場合は、このタイプの表は使用しないでください。これらの要素は、範囲クラスター表として使用できるデータがあるかどうかを判断するのに役立ちます。

範囲クラスター表の構造に関連した利点

範囲クラスター表を使用すると、いくつかの利点があります。

- 直接アクセス

  アクセスは、範囲クラスター表のキー/RID マッピング機能を通して行われます。
- 保守が少ない

  B+ ツリーなどの 2 次構造は、それぞれの INSERT、UPDATE、または DELETE ごとに更新を行う必要がありません。
- ロギングが少ない

  同じサイズの通常表や関連する B+ ツリー索引と比較して、範囲クラスター表では、ロギングの実行が少なくなっています。
- 必要なバッファー・プール・メモリーが少ない

  2 次の構造の保管に、追加のメモリーが必要ありません。
- B+ ツリー表の配列プロパティー

  追加のレベルや B+ ツリーの次のキーのロッキング体系を使用しなくても、 B+ ツリー表に同じ配列のレコードが見つかります。 RCT では、通常の B+ ツリー索引に比べて、コード・パス長が短くなっています。ただし、この利点を活用するためには、DISALLOW OVERFLOW で範囲クラスター表を作成し、 (まばらではなく) 高密度にデータを入れる必要があります。
- 1 つのより小さな索引

  各キーをディスク上のロケーションにマッピングすることによって、今まで別に必要だった索引が不要になり、より小さな 1 つの索引を使用して表を作成できるようになりました。範囲クラスター表では、表内のデータにアクセスするためのアプリケーション要件によっては、 2 番目の別の索引が不要になる場合があります。アプリケーションが必要とすれば、通常の索引を作成することは可能です。

範囲クラスター表の非互換性

これらの考慮事項に加えて、範囲クラスター表の使用できる場所が限定されるか、さもなければ他のユーティリティーがそれらの表についてうまく動作しなくなる非互換性がいくつかあります。

範囲クラスター表に関する制限には、次のことがあります。

- パーティション表での範囲クラスター表はサポートされていません。

  レンジ・クラスタリングを使用してパーティション表を作成しようとすると、エラー・メッセージ SQL0270 rc=87 が戻されます。
- 宣言済みグローバル一時表 (DGTT) はサポートされません。

  それらの一時表では、範囲クラスター・プロパティーを使用できません。
- 自動サマリー表 (AST) はサポートされません。

  それらの表では、範囲クラスター・プロパティーを使用できません。
- ロード・ユーティリティーはサポートされません。

  行は、インポート操作または並列挿入アプリケーションにより、一度に 1 個ずつ挿入する必要があります。
- REORG TABLE ユーティリティーはサポートされません。

  DISALLOW OVERFLOW として定義されている範囲クラスター表を再編成する必要はありません。それでも、ALLOW OVERFLOW として定義されている範囲クラスター表で、そのオーバーフロー領域内のデータを再編成することは許されません。
- 範囲クラスター表は 1 つの論理マシンにおいてのみ可能です。

  データベース・パーティション・フィーチャー (Database Partitioning Feature (DPF)) 付きの Enterprise Server Edition (ESE) の場合、範囲クラスター表が、複数のデータベース・パーティションを含むデータベース内に存在することは許されません。
- 設計アドバイザーでは、範囲クラスター表は勧められていません。
- 範囲クラスター表は、定義により既にクラスター化されています。

  したがって、次のクラスタリング・スキームは、範囲クラスター表と互換性がありません。
  - マルチディメンション・クラスタリング (MDC) 表
  - クラスタリング索引
- 値とデフォルトの圧縮はサポートされません。
- 範囲クラスター表に対するリバース・スキャンはサポートされません。
- IMPORT コマンドの REPLACE オプションはサポートされません。
- ALTER TABLE ... ACTIVATE NOT LOGGED INITIALLY ステートメントの WITH EMPTY TABLE オプションは、サポートされません。

範囲クラスター表と範囲外のレコード・キー値

レコードのオーバーフローを可能にする範囲クラスター表 (RCT) の振る舞いは、CREATE TABLE ステートメントと ALLOW OVERFLOW オプションを使用して制御します。この方法により、確実に、定義された範囲内で表に必要なすべてのページが即時に割り振られます。

一度作成されると、定義された範囲内に置かれたすべてのキー付きレコードは、表の作成時にオーバーフロー・オプションが有効になっていたかどうかに関係なく、同じ働きをします。違いが生じるのは、定義された範囲の外に置かれたキー付きレコードがある場合です。この場合、表でオーバーフローが有効になっていると、レコードは、動的に割り振られてオーバーフロー域の中に置かれます。定義された範囲の外からさらにレコードが追加されると、これらはオーバーフロー域に置かれ、オーバーフロー域は大きくなっていきます。すると、このオーバーフロー域に関係する、表に対するアクションが行われたとき、アクションにはオーバーフロー域へのアクセスが含まれるため、アクションの処理時間が長くかかるようになります。オーバーフロー域へのアクセスにかかる時間は、オーバーフロー域が大きくなるほど長くなります。ある程度長い期間オーバーフロー域を使用した後、新しい拡張された範囲を使用して定義した新しい範囲クラスター表に既存の表のデータをエクスポートして、既存の表のサイズを小さくすることを考慮してください。

挿入するレコード・キー値が範囲クラスター表で定義された範囲外である場合があるかもしれません。このタイプの RCT を存在させるためには、 CREATE TABLE ステートメントで DISALLOW OVERFLOW オプションを使用する必要があります。このタイプの RCT を作成したときは、レコード・キー値が許可または定義された範囲の外に置かれることを伝えるエラー・メッセージを受け入れる必要があります。

範囲クラスター表のロック

通常の処理において、レコードのロックは常に 1 つのアプリケーションやユーザーしか、レコードまたはレコード・グループへアクセスできないようにする場合に行われます。範囲クラスター表では、キーのロックや Next Key ロックの代わりに、「離散ロック」が使用されます。

この方式では、アプリケーションやユーザーが要求した操作によって影響を受ける (もしくは、影響を受ける可能性のある) すべてのレコードをロックします。行われるロックの数は、分離レベルに依存します。

現行で空になっていても事前割り振りされている、範囲クラスター表の限定行はロックされます。これにより、次のキーのロックは必要なくなります。結果として、高密度の範囲クラスター表に必要なロックは少なくなります。

第 3 章 マルチディメンション・クラスタリング (MDC) 表

マルチディメンション・クラスタリング表

マルチディメンション・クラスタリング (MDC) は、マルチディメンションの表のデータ・クラスタリングを柔軟、連続的、かつ自動的に実行する優れた方式です。MDC によって照会のパフォーマンスをかなり向上させることができ、

さらに、挿入、更新、削除の間の再編成や索引保守操作などデータ保守操作のオーバーヘッドが大きく削減されます。 MDC の主な目的は、データウェアハウジングおよび大規模データベース環境での使用ですが、オンライン・トランザクション処理 (OLTP) 環境でもこれを使用することができます。

通常表と MDC 表の比較

通常表には、レコードに基づく索引があります。それらの索引のクラスタリングは、単一のディメンションに制限されています。バージョン 8 より前のデータベース・マネージャーは、クラスター索引を介して、単一ディメンションのデータ・クラスタリングだけをサポートしていました。表でレコードが挿入および更新されるとき、データベース・マネージャーはクラスター索引を使用して、データの物理的順序を索引のキー順序のページで保守します。

クラスター索引は、述部にクラスター索引キーが (1 つまたは複数) 含まれるような照会範囲のパフォーマンスを大きく改善します。優れたクラスター索引により、表の一部分だけにアクセスするだけで済み、プリフェッチをさらに効率的に実行できるため、パフォーマンスが向上します。

しかし、クラスター索引を使用したデータ・クラスタリングには、いくつかの欠点もあります。第 1 の点として、時間が経過するにつれてスペースがデータ・ページで埋まるため、クラスタリングが保証されません。挿入操作においてレコードが追加されるページは、そのレコードと同一または類似のクラスター化キー値のレコードの近くですが、理想的な場所にスペースがないなら、それは表の中のどこか別の場所に挿入されることになります。したがって、表をクラスター化し直し、将来のクラスター化挿入要求に備えてフリー・スペースを余分に設けるようページをセットアップするために、定期的な表の再編成が必要になります。

第 2 の点として、データのクラスター化は 1 ディメンションに関してだけなので、「クラスター」索引として指定できるのは 1 個の索引だけであり、他のすべての索引は非クラスター索引になります。この制限は、バージョン 8.1 より前の索引がすべてそうであるようにクラスター索引がレコード・ベースであるということに関連しています。

第 3 の点として、レコード・ベースの索引の場合、表のあらゆる単一のレコードについてポインターが含まれているため、サイズが巨大になる場合があります。

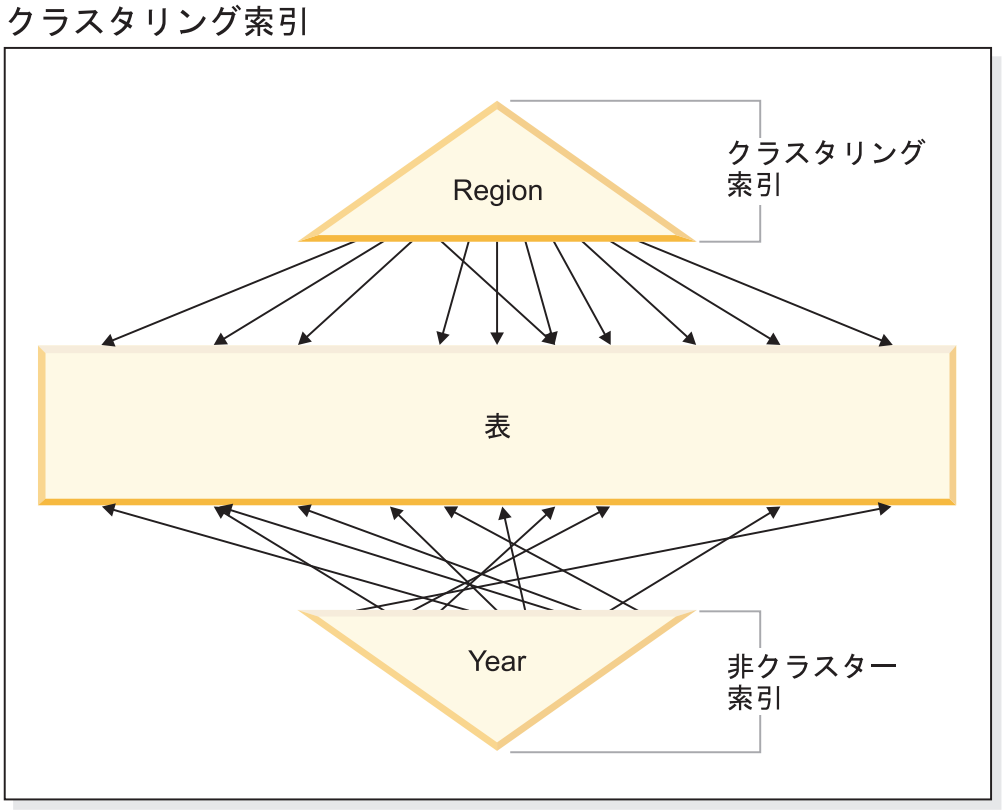


図 8. クラスター索引を伴う通常表

図 8 の表には、2 個のレコード・ベースの索引があります。

- 『Region』 上のクラスター索引
- 『Year』 上の別の索引

『Region』 索引は、クラスター索引です。したがって、その索引の中でキーをスキャンする際、対応するレコードは表の中の同じページか、その近くのページにほとんどが存在しているはずです。一方、『Year』 索引は非クラスター索引なので、その索引の中でキーをスキャンする際、表全体の中で対応するレコードが存在する位置はランダムです。クラスター索引のスキャンのほうが入出力パフォーマンスが高く、その索引に対してデータのクラスター化の程度が高いほど順次プリフェッチのメリットが大きくなります。

MDC には、ブロック・ベースの索引が導入されています。「ブロック索引」は、個々のレコードではなくレコードのブロック (グループ) へのポインターです。クラスター化値に従って MDC 表のデータを物理的に複数のブロックに編成し、ブロック索引を使用してそれらのブロックにアクセスすることによって、 MDC はクラスター索引の欠点を解決するだけでなく、さらにパフォーマンスが大きく改善されます。

第 1 の点として、MDC では、 1 つの表を同時に複数のキー (つまりディメンション) に基づいて物理的にクラスター化できます。 MDC では、単一ディメンション・クラスタリングの利点が複数ディメンション、またはクラスター化キーにまで拡大されます。表のうち指定された 1 つ以上のディメンションのクラスタリングがあると、照会のパフォーマンスが向上します。このような照会は、正しいディメン

ション値を持つレコードが含まれるページにのみアクセスします。しかも、該当するページはブロックまたはエクステントごとにグループ化されます。

第 2 の点として、クラスター索引を持つ表の場合、時間の経過とともに非クラスタリングされる可能性があります。しかし、たいていの場合 MDC 表は、すべてのディメンションにわたって自動的かつ連続的にクラスタリングを保守することができます。したがって、データの物理的順序をリストアするために MDC 表を頻繁に再編成する必要がありません。ブロック内のレコード順序は常に保守されますが、ブロックの物理順序 (つまり、ブロック索引スキャン時のブロック間の距離) は挿入時に (または場合によっては初期ロード時にも) 保守されません。

第 3 の点として MDC の場合、クラスター索引はブロック・ベースです。そのような索引は、通常のレコード・ベースの索引に比べて格段に小さいため、ディスク・スペースがずっと少なくて済み、スキャンも高速です。

MDC 表のディメンションの選択

マルチディメンション・クラスタリング表を使用することにした場合、選択するディメンションは、それらの表を使用したりブロック・レベルのクラスタリングを利用したりする照会のタイプに依存するだけでなく、もっと重要なこととして実際のデータの量と分布に依存します。 MDC 表の設計の側面と、適切なディメンションおよびブロック・サイズの選択に関する指針については、関連する概念のリンクを参照してください。

MDC の利点を生かせる照会

ご使用の表のクラスタリング・ディメンションの選択の際の 1 番目の考慮事項は、ブロック・レベルでクラスタリングにより恩恵を受ける照会の判別です。データに対して実施する作業を構成する照会に基づいてディメンションを選択する場合、複数の候補があるのが一般的です。それらの候補のランキングは重要です。列、特にカーディナリティーの低い列のうち、等式条件または範囲条件の述部を含む照会に関係したものは、クラスタリング・ディメンションのメリットを最大限活用でき、実際それらは候補として考慮するべきです。ディメンション表を持つスター型結合に含まれている MDC ファクト表内の外部キーのディメンションの作成を考慮する場合もあるでしょう。複数のディメンションの自動および継続クラスタリング、およびエクステント・レベルまたはブロック・レベルでのクラスタリングのパフォーマンス上の利点に留意する必要があります。

マルチディメンション・クラスタリングを使用できる照会はたくさんあります。そのような照会の例を以下に示します。以下に示す例のいくつかでは、ディメンション c1、c2、および c3 の MDC 表 t1 を前提としています。その他の例では、ディメンション color および nation の MDC 表 mdctable を前提としています。

例 1:

```
SELECT .... FROM t1 WHERE c3 < 5000
```

この照会には、単一ディメンション上に範囲述部が含まれているので、 c3 上のディメンション・ブロック索引を使用して表にアクセスするように内部に再書き込みすることができます。索引は 5000 より少ない値をキーとするブロック ID (BID)

のためにスキャンされ、小規模なリレーショナル・スキャンは、実際のレコードを検索するためにブロックの結果セットに適用されます。

例 2:

```
SELECT .... FROM t1 WHERE c2 IN (1,2037)
```

この照会には、単一ディメンション上に IN 述部が含まれており、ブロック索引ベースのスキャンをトリガーすることができます。この照会は、 c2 上のディメンション・ブロック索引を使用して表にアクセスするように内部に再書き込みすることができます。索引は、1 および 2037 の値を持つキーの BID のためにスキャンされ、小規模なリレーショナル・スキャンは、実際のレコードを検索するためにブロックの結果セットに適用されます。

例 3:

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' AND NATION='USA'
```

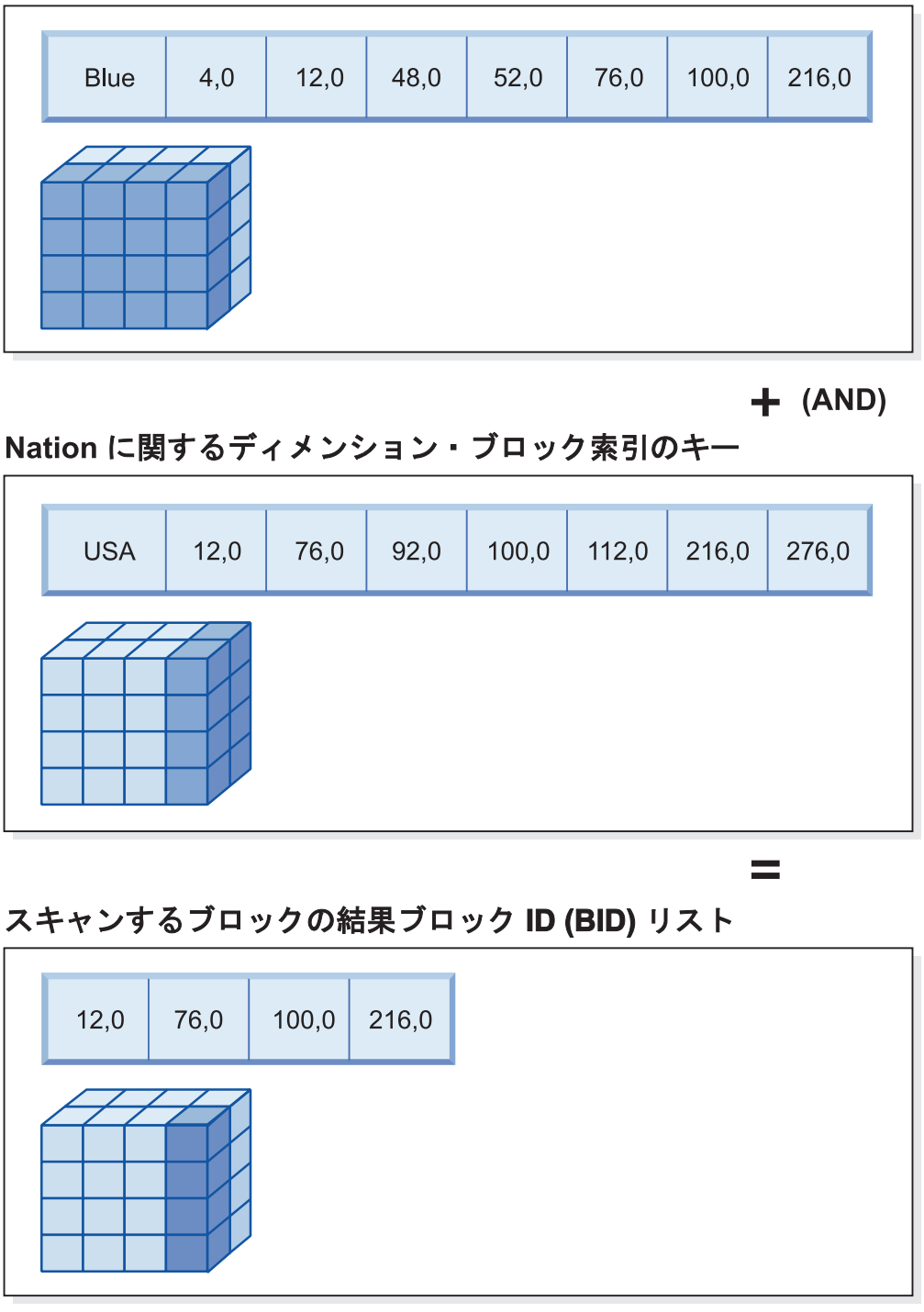


図 9. 2 個のブロック索引との論理 AND 演算を使用する照会要求

この照会要求を実行するため、次のようにします (図 9 を参照)。

- ディメンション・ブロック索引が検査されます。 1 つは Blue スライスのため、もう 1 つは USA スライスのためです。
- ブロック論理 AND 演算を実行して、 2 つのスライスの論理積を求めます。つまり、この論理 AND 演算により、 2 つのスライスの両方にあるブロックだけが判別されます。

- 表内の結果ブロックの小規模なリレーション・スキャンが実行されます。

例 4:

```
SELECT ... FROM t1
   WHERE c2 > 100 AND c1 = '16/03/1999' AND c3 > 1000 AND c3 < 5000
```

この照会には、c2 と c3 に関する範囲述部と、c1 に関する等式述部、そして論理 AND 演算が含まれています。これは、内部的にはディメンション・ブロック索引のそれぞれに対する表アクセスとして書き直すことができます。

- 値が 100 より大きいキーの BID を検索するため、 c2 ブロック索引のスキャンが実行されます。
- 値が 1000 と 5000 の間にあるキーの BID を検索するため、 c3 ブロック索引のスキャンが実行されます。
- 値が '16/03/1999' であるキーの BID を検索するため、 c1 ブロック索引のスキャンが実行されます。

次に、各ブロック・スキャンの結果 BID に対して論理 AND 演算が実行されて、その論理積が求められ、実際のレコードを検索するため、ブロックの結果セットに対して小規模なリレーショナル・スキャンが適用されます。

例 5:

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' OR NATION='USA'
```

この照会要求を実行するため、次のようにします。

- ディメンション・ブロック索引が検査されます。各スライスに対して 1 つずつ実行されます。
- 論理 OR 演算により、2 つのスライスの和集合が求められます。
- 表内の結果ブロックの小規模なリレーション・スキャンが実行されます。

例 6:

```
SELECT .... FROM t1 WHERE c1 < 5000 OR c2 IN (1,2,3)
```

この照会には、 c1 ディメンションに関する範囲述部、c2 ディメンションに対する IN 述部、そして論理 OR 演算が含まれています。この照会は、ディメンション・ブロック索引 c1 および c2 に関する表アクセスを実行するように内部的に書き直すことができます。 5000 未満の値の検索のため c1 ディメンション・ブロック索引のスキャンが実行された後、値 1、2、および 3 の検索のため c2 ディメンション・ブロック索引の別のスキャンが実行されます。各ブロック索引スキャンの結果 BID に対して論理 OR 演算が実行された後、実際のレコードを検索するため、ブロックの結果セットに対して小規模なリレーショナル・スキャンが適用されます。

例 7:

```
SELECT .... FROM t1 WHERE c1 = 15 AND c4 < 12
```

この照会には、c1 ディメンションに関する等式述部、ディメンションでない列に関する範囲述部、および論理 AND 演算が含まれています。これは、c1 に 15 という値を持つ表のスライスからブロックのリストを入手するために、 c1 上のディメンション・ブロック索引にアクセスするように内部的に書き直すことができます。 c4 上に RID 索引がある場合、 12 より少ない c4 を持つレコードの RID を検索する

ために索引スキャンを実行してから、レコードのこのリストとブロックの結果リストの論理 AND 演算を実行できます。この論理積は c1 に 15 という値を持つブロック内にない RID を除去し、修飾するブロック内にあるリスト済み RID のみを表から検索します。

c4 上に RID 索引がない場合、条件を満たすブロックのリストを探してブロック索引をスキャンでき、それぞれのブロックの小規模なリレーショナル・スキャンの際に、見つかったそれぞれのレコードに述部 c4 < 12 を適用できます。

例 8:

color、year、nation、および部品番号 (PARTNO) に対する行 ID (RID) 索引のためのディメンションがあるというシナリオでは、次の照会が可能です。

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' AND PARTNO < 1000
```

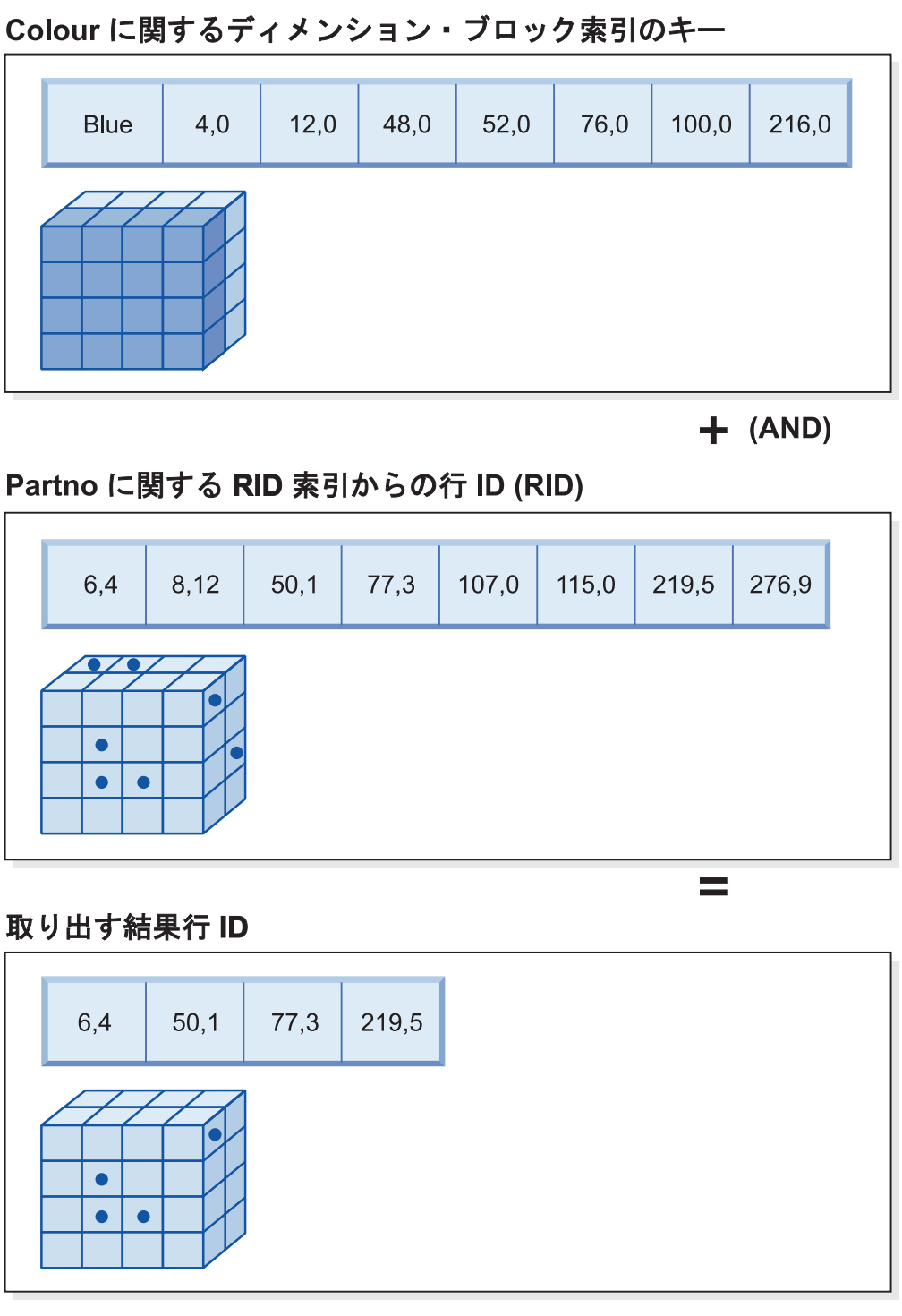


図 10. 1 つのブロック索引および行 ID (RID) 索引に対する論理 AND 演算を使用する照会要求

この照会要求を実行するため、次のようにします (図 10 を参照)。

- ディメンション・ブロック索引と RID 索引が検査されます。
- ブロックと RID の論理 AND 演算を使用することにより、スライスと、述部条件を満たす行の論理積を求めます。
- その結果は、修飾ブロックにも属している RID だけです。

例 9:

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' OR PARTNO < 1000
```

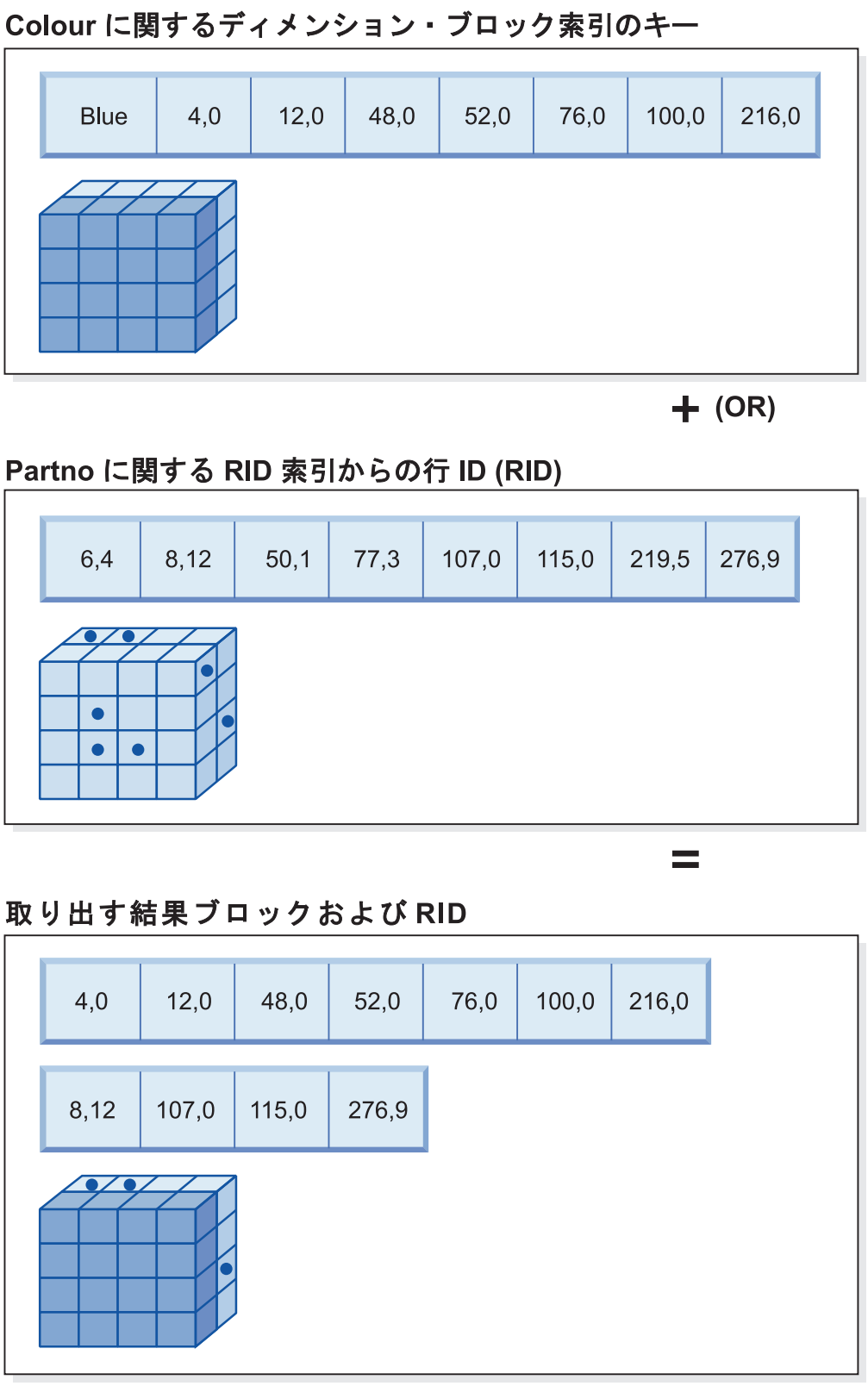


図 11. 論理 OR 演算を使用したブロック索引と行 ID の動作

この照会要求を実行するため、次のようにします (図 11 を参照)。

- ディメンション・ブロック索引と RID 索引が検査されます。

- ブロックと RID の論理 OR 演算を使用することにより、スライスと、述部条件を満たす行の和集合を求めます。
- 結果は、条件を満たすブロックに含まれる行と、該当ブロックに入らない付加的な RID のうち述部条件を満たすもののすべてです。各ブロックにおいて小規模なリレーショナル・スキャンが実行されることにより、そのレコードが取り出され、さらにそれらのブロックには含まれない付加的なレコードが別個に取り出されます。

例 10:

```
SELECT ... FROM t1 WHERE c1 < 5 OR c4 = 100
```

この照会には、 c1 ディメンションに関する範囲述部、非ディメンション列 c4 に対する等式述部、そして論理 OR 演算が含まれています。 c4 列に対する RID 索引があるなら、これは、 c1 に関するディメンション・ブロック索引と c4 に関する RID 索引の論理 OR 演算を実行するように内部的に書き直すことができます。 c4 に対する索引がない場合には、すべてのレコードをチェックする必要があるため、表スキャンを選択できます。 c1 に対する 4 未満の値のブロック索引スキャン、そして c4 に対する値 100 の RID 索引スキャンが、論理 OR 演算で結合されることになります。該当するブロックの中のすべてのレコードが条件を満たし、さらにそれらのブロックには含まれないレコードからも付加的な RID が取り出されるため、該当する各ブロックに対して小規模なリレーショナル・スキャンが実行されます。

例 11:

```
SELECT .... FROM t1,d1,d2,d3
   WHERE t1.c1 = d1.c1 and d1.region = 'NY'
      AND t2.c2 = d2.c3 and d2.year='1994'
      AND t3.c3 = d3.c3 and d3.product='basketball'
```

この照会には、Star Join が含まれています。この例では、t1 はファクト表で、主キー d1、d2、および d3 (ディメンション表) に対応する外部キー c1、c2、および c3 を持っています。ディメンション表は、MDC 表である必要がありません。 region、year、および product は、 (ディメンション表が MDC 表である場合) 通常の (正規) 索引またはブロック索引を使用して索引化できるそれぞれのディメンション表の列です。 c1、c2、および c3 値上のファクト表にアクセスする際には、これらの列に対してディメンション・ブロック索引のブロック索引スキャンを実行し、次に、結果 BID の論理 AND 演算を実行できます。ブロックのリストがある場合は、レコードを入手するためにそれぞれのブロック上で小規模なリレーショナル・スキャンを実行できます。

セルの密度

該当するディメンションとエクステント・サイズの選択は、 MDC の設計において**非常に**重要です。それらの要因により、表に関して予期されるセル密度が決まります。セル内のレコード数には関係なくすべての既存のセルに対してエクステントが割り振られるため、それらの要因は重要です。適切な値を選択するなら、ブロック・ベースの索引付けとマルチディメンション・クラスタリングを活用して、パフォーマンスが向上します。目指すところは、密度の高いブロックでマルチディメンション・クラスタリングを最大限に活用し、最良のスペース使用率を引き出すことです。

したがって、マルチディメンション表の設計の際に非常に重要になる考慮事項は、現在のデータおよび予測データに基づく、表内のセルの予期される密度です。照会パフォーマンスに基づいてディメンションのセットを選択し、それぞれのディメンションごとの可能な値の数に基づいて、表内のセルの潜在的な数が非常に大きくなるようにします。表内の可能なセルの数は、それぞれのディメンションのカーディナリティーのデカルト積と等しくなります。例えば、ディメンション Day、Region、および Product に関する表をクラスター化し、5 年分のデータを包含する場合、表内に 1821 days * 12 regions * 5 products = 109 260 個の異なるセルが必要になる可能性があります。 2、3 のレコードしか含んでいないセルは、そのセル用のレコードを保管するために、そのセルに割り振られたページのブロック全体が必要です。ブロック・サイズが大きい場合、この表は、実際に必要なサイズよりもさらに大きくなる可能性があります。

セルの密度を最適にすることに貢献する可能性のある設計上の要因がいくつかあります。

- ディメンション数。
- 1 つ以上のディメンションの細分度。
- 表スペースのブロック (エクステント) サイズとページ・サイズ。

設計を最高のものにするため、次のステップを実行してください。

1. 候補となるディメンションを識別します。

   どの照会についてブロック・レベル・クラスタリングを利用できるかを判断します。次の特性のうちの一部または全部が当てはまる列があるかどうかに関して、潜在的なワークロードを調べてください。

   - IN リスト述部の範囲と等価性
   - データのロールインまたはロールアウト
   - GROUP BY 節と ORDER BY 節
   - 結合節 (特にスター・スキーマ環境の場合)
2. セルの数を見積もります。

   候補となるディメンション・セットにより編成された表の中に、どれだけのセルが可能かを識別します。データの中に出現するディメンション値の固有の組み合わせの数を判別します。表が存在するなら、その表に関してディメンションとなる各列の中で、異なる値の種類数を単に選択することによって、現在のデータに関して正確な数を判別できます。あるいは、表の統計データしかない場合には、ディメンション候補の列のカーディナリティーを乗算することにより概算できます。

   注: パーティション・データベース環境の表の場合、考慮するどのディメンションにも分散キーが関連しないなら、データの全体をデータベース・パーティションの数で除算することによって、 1 セル当たりの平均データ量を計算することが必要になります。
3. スペースの占有度または密度を見積もります。

   平均して、格納されている行の数が少ないため一部しかデータが入っていないブロックが 1 セルにつき 1 個あることを考慮してください。 1 セル当たりの行数が小さいほど、データが一部しか入っていないブロックが多くなります。ま

た、(データの偏りがほとんど、あるいはまったくないとして) 平均して、 1 セル当たりのレコード数は、表の中のレコード数をセル数で除算して得られることに注意してください。しかし、パーティション・データベース環境の表の場合は、各データベース・パーティションごとに 1 セル当たりのレコード数を考慮する必要があります。ブロックは、データベース・パーティションごとのデータに対して割り振られるからです。パーティション・データベース環境においてスペースの占有率と密度を見積もる際には、表全体ではなく各データベース・パーティションごとに 1 セル当たりの平均レコード数を考慮する必要があります。詳しくは、『マルチディメンション・クラスタリング (MDC) 表の作成、配置、および使用』を参照してください。

密度を改善するには、いくつかの方法があります。

- ブロック・サイズを小さくすることにより、一部しかデータが入っていないブロックの占めるスペースを少なくする。

  エクステント・サイズをある程度小さく設定することにより、各ブロックのサイズを小さくする。一部しかデータが入っていないブロックを含むセル、あるいはいくらかのレコードが含まれるだけのブロックを 1 個だけ含むセルに関して、各セルごとに無駄になるスペースが少なくなります。しかし、レコードの数の多いセルの場合、レコードを含めるために必要なブロック数が増えるというデメリットがあります。これにより、ブロック索引内のセル用のブロック ID (BID) の数が増加します。これらの索引がより大きくなり、これらの索引への挿入や削除がさらに増えるようになると、ブロックはより迅速に空になったり、満杯になったりします。また、追加して取り込まれたセル値用の表内のクラスター化されたデータはより小さいグループに分けられるのに対して、クラスター化されたデータはより少数のより大きいグループに分けられます。
- ディメンション数を少なくすることにより、または生成される列でのセルの細分度を高めることにより、セルの数を少なくする。

  1 つ以上のディメンションをロールアップして細分度を低下させることにより、カーディナリティーを下げることができます。例えば、Region および Product における以前の例でデータのクラスター化を継続することができますが、 Day というディメンションを YearAndMonth のディメンションに置き換えます。これは、3600 という可能なセル数を使用して、YearAndMonth、Region、および Product に 60 (12 カ月× 5 年)、12、および 5 というカーディナリティーを提供します。それにより、各セルに含まれる値の範囲が広くなり、レコード数が少なくなる可能性が低くなります。

  さらに、関係する列に対して頻繁に使用される述部を考慮する必要があります。例えば、月、四半期、あるいは日に対するものが大半であるかどうか、などです。これは、ディメンションの細分度の変更希望に影響を与えます。例えば、ほとんどの述部が特定の Day 上にあり、Month 上に表をクラスター化した場合、 DB2 Database for Linux, UNIX, and Windows は、YearAndMonth 上のブロック索引を使用して、希望する Day が含まれている Month を迅速に限定し、関連したブロックのみにアクセスすることができます。しかし、ブロックのスキャンでは、どの日が条件を満たすかを判別するために Day 述部を適用する必要があります。ただし、Day 上でクラスター化する場合 (そして Day のカーディナリティーが高い場合)、スキャンするブロックを判別するために Day 上のブロック索引を使用でき、 Day 述部は修飾するそれぞれのセ

ルの最初のレコードにのみ再適用されなければなりません。この場合、ユーザー定義関数を使用して、 Region 列 (12 の異なる値を含む) を Regions West、North、South、および East にロールアップするように、セルの密度を増やすためにその他のディメンションの 1 つをロールアップすることを考慮する必要があります。

MDC 表を作成する際の考慮事項

MDC 表を作成する際には、さまざまな要素を考慮する必要があります。以下のセクションでは、 MDC 表を作成、配置、および使用する方法の決定に対して、現在使用しているデータベース環境 (例えばパーティション・データベースかどうかなど)、また MDC 表のためのディメンションの選択がどう影響するかについて説明します。さらに、DB2 設計アドバイザーと、それを使用して前述の問題に関するアドバイスを提供する方法についても説明します。

既存の表から MDC 表へのデータの移動

データウェアハウスまたは大規模データベース環境において、照会のパフォーマンスを向上させ、データ保守操作のオーバーヘッドを少なくするため、通常表のデータを、マルチディメンション・クラスタリング (MDC) 表に移動することができます。既存の表から MDC 表へデータを移動するには、データをエクスポートし、元の表をドロップし (オプション)、 (ORGANIZE BY DIMENSIONS 節を含む CREATE TABLE ステートメントを使用して) マルチディメンション・クラスタリング (MDC) 表を作成し、その MDC 表にデータをロードします。

SYSPROC.ALTOBJ という ALTER TABLE プロシージャーを使用すると、既存の表から MDC 表へのデータ変換を実行できます。このプロシージャーは、DB2 設計アドバイザーから呼び出されます。表と表の間でのデータ変換にはかなりの時間が必要になる場合があり、それは表のサイズや変換の必要なデータの量によって異なります。

ALTOBJ プロシージャーは、表変更において次のことを実行します。

- 表の従属オブジェクトをすべてドロップします。
- 表の名前を変更します。
- 新しい定義を使用して表を作成します。
- 表の従属オブジェクトをすべて再作成します。
- 表に既存のデータを、新しい表で必要なデータに変換します。つまり、変換前の表のデータを選択し、そのデータを新しい表にロードします。その際には、変換前のデータ・タイプから変換後のデータ・タイプに変換するため、列関数が使用される場合があります。

SMS 表スペース内の MDC 表

MDC 表を SMS 表スペースに格納することを予定している場合には、複数ページ・ファイル割り振りを使用する必要があります (複数ページ・ファイル割り振りは、バージョン 8.2 以降で新たに作成されるデータベースではデフォルトです)。これは、 MDC 表は常にエクステントを単位として拡張されるため、それらのエクステント内のすべてのページが物理的に連続していることが重要になるからです。したがって、複数ページ・ファイル割り振りを無効にすることには、スペースに関する

メリットは何もありません。それだけでなく、それを有効にすることにより、各エクステント内のページが物理的に連続する可能性が格段に大きくなります。

DB2 設計アドバイザーと比較した場合の MDC Advisor のフィーチャー

DB2 設計アドバイザー (db2advis) (旧称 Index Advisor) には、 MDC フィーチャーが含まれています。このフィーチャーでは、処理のパフォーマンスを向上させるため、基本列に対する低密度化を含め、MDC 表でクラスタリング・ディメンションを使用することを推奨します。*低密度化* という語は、クラスタリング・ディメンションのカーディナリティー (値の種類数) を少なくすることに関する数学用語です。低密度化の一般的な例としては、日付、曜日、月、四半期による低密度化があります。

DB2 設計アドバイザーの MDC フィーチャーを使用するには、データベース内に少なくとも複数のデータのエクステントがなければなりません。 DB2 設計アドバイザーはデータを使用して、データ密度とカーディナリティーのモデル化を行います。

データベースの表の中にデータがない場合、DB2 設計アドバイザーは MDC を推奨しません。たとえデータベースの表が空であり、しかしデータが取り込まれるデータベースを示すモックアップされた統計のセットを持つ場合でも同じです。

推奨事項には、ディメンションの低密度化を定義する潜在的な生成列を識別することが含まれています。推奨事項には、可能性のあるブロック・サイズは含まれていません。 MDC 表の推奨事項作成では、表スペースのエクステント・サイズが使用されます。推奨される MDC 表が既存の表と同じ表スペース内で作成され、したがってエクステント・サイズが同じであることが前提になっています。 MDC ディメンションに関する推奨事項は、表スペースのエクステント・サイズによって変わることがあります。というのは、エクステント・サイズは、 1 個のブロックまたはセルに入るレコード数に影響するからです。それはセルの密度に直接影響します。

表に対して単一ディメンションでも複数ディメンションでも推奨されることがありますが、複合列ディメンションではなく単一列ディメンションだけが考慮されます。 MDC フィーチャーは、結果となる MDC ソリューションでのセルのカーディナリティーを小さくすることを目標として、サポートされているほとんどのデータ・タイプに関して低密度化を推奨します。データ・タイプの例外には CHAR、VARCHAR、GRAPHIC、および VARGRAPH のデータ・タイプが含まれます。サポートされているデータ・タイプは、すべて INTEGER にキャストされ、生成される式によって低密度化されます。

DB2 設計アドバイザーの MDC フィーチャーの目標は、パフォーマンスが改善される MDC ソリューションを選択することです。第 2 の目標は、データベースのストレージ拡張を控え目なレベルに保つことです。表ごとの最大ストレージ拡張の判別には、統計的手法が使用されます。

Advisor 内の分析操作には、ブロック索引アクセスのメリットだけでなく、表のディメンションに対する挿入、更新、および削除操作における MDC の影響も含まれます。表に対するそれらのアクションには、セル間でレコードを移動させることにな

る可能性があります。また、分析操作は、特定の MDC ディメンションに関するデータの編成処理の結果としての表拡張による潜在的なパフォーマンスの影響をモデル化します。

MDC フィーチャーを有効にするには、 db2advis ユーティリティーで -m <advise type> フラグを使用します。マルチディメンション・クラスタリング表を指定するため、アドバイス・タイプとして「C」を使用します。アドバイス・タイプには、「I」(索引)、「M」(マテリアライズ照会表)、「C」(MDC)、および「P」(パーティション・データベース環境) があります。アドバイス・タイプは、互いに組み合わせて使用できます。

注: DB2 設計アドバイザーは、サイズが 12 エクステントより小さい表を検討しません。

Advisor は、推奨事項提供において MQT と通常の基本表の両方を分析します。

MDC フィーチャーからの出力には、次のものが含まれます。

- MDC ソリューションに出現する低密度化されたディメンションについて、表ごとに生成される列式。
- 各表について推奨される ORGANIZE BY 節。

推奨事項は、 stdout と、EXPLAIN 機能の一部である ADVISE 表の両方に報告されます。

MDC 表とパーティション・データベース環境

マルチディメンション・クラスタリングは、パーティション・データベース環境と組み合わせて使用することができます。実際、MDC はパーティション・データベース環境を補います。パーティション・データベース環境は、複数の物理または論理ノードの間に表データを配分させるために使用されます。その目的は、次のとおりです。

- 複数のマシンを利用して、並列処理される要求の数を増やす。
- 単一データベース・パーティションの限界を超えて、表の物理サイズを大きくする。
- データベースのスケーラビリティーを改善する。

表を配分する理由は、表が MDC 表か通常表かということとは別です。例えば、分散キーを構成する列を選択するための規則は同じです。 MDC 表の分散キーには、列がその表のディメンションの一部であるかどうかによらず、どんな列でも関係することがあります。

分散キーが表のディメンションと同一なら、各データベース・パーティションにはそれぞれ表の異なる部分が入れられます。例えば、この章で使用しているサンプルの MDC 表は、 2 つのデータベース・パーティションにわたって color により配分され、その後、Color 列を使用してデータが分割されます。その結果、Red スライスと Blue スライスが 1 つのデータベース・パーティションに、 Yellow スライスがもう 1 つのパーティションになることがあります。分散キーが表のディメンションと同一でない場合には、各データベース・パーティションには、それぞれ各スライスのデータのサブセットが入れられることになります。ディメンションを選択し、セル占有度を見積もる際には (『セルの密度』のセクションを参照)、平均とし

て 1 セル当たりの合計データ量は、データ全体をデータベース・パーティション数で除算して得られることに注意してください。

マルチディメンションの MDC 表

照会で一部の特定の述部が頻繁に使用されることがあらかじめわかっている場合、ORGANIZE BY DIMENSIONS 節を使って、関連する列に基づいて表をクラスター化することができます。

例 1:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, c3, c4)
```

例 1 の表は、論理キューブを形成する 3 つの固有な列における値に基づいてクラスター化されています (つまり 3 ディメンションを持っている)。こうすれば、照会処理の際、 1 つまたは複数のディメンションにおいて表をスライスすることにより、適切なスライスまたはセルに含まれるブロックだけがリレーショナル演算子によって処理されるようにすることができます。ブロック・サイズ (ページ数) が、表のエクステント・サイズになることに注意してください。

複数列に基づくディメンションの MDC 表

それぞれのディメンションを、1 つまたは複数の列で構成することが可能です。一例として、2 つの列を含むディメンションに基づいてクラスター化される表を作成することができます。

例 2:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, (c3, c4))
```

例 2 では、c1 および (c3, c4) の 2 つのディメンションに基づいて表がクラスター化されます。こうすると、照会処理においては、ディメンション c1 もしくは複合ディメンション (c3, c4) に基づいて表を論理的にスライスすることができます。この表のブロック数は例 1 の表と同じですが、ディメンション・ブロック索引の数は 1 つ少なくなります。例 1 では、列 c1、c3、 c4 それぞれに関する 3 つのディメンション・ブロック索引が作成されます。例 2 の場合は、列 c1 に関するディメンション・ブロック索引と、列 c3 および c4 に関するディメンション・ブロック索引の 2 つが作成されます。この 2 つの方法の主な違いは、例 1 の場合、 c4 のみを扱う照会が c4 に関するディメンション・ブロック索引を使用して、関連するデータ・ブロックに素早く直接的にアクセスできることです。例 2 では c4 がディメンション・ブロック索引の 2 番目のキー部分であるため、 c4 のみを扱う照会はより多くの処理を行います。ただし、例 2 の場合、保守および保管されるブロック索引の数は 1 つ少なくなります。

DB2 設計アドバイザーは、複数列を含むディメンションに関する推奨事項を作成しません。

列式をディメンションとする MDC 表

ディメンションをクラスタリングする際、列式を使用することもできます。列式に基づくクラスター化は、細分性を粗くしてディメンションをロールアップする場合

に便利です。例えば、住所を地理上の場所または地域にロールアップする場合や、日付を週、月、または年にロールアップする場合などです。このようなディメンションのロールアップを実装するには、生成された列を使用することができます。このタイプの列定義では、ディメンションを表す式を使って列を作成することができます。例 3 のステートメントによって作成される表は、基本となる 1 つの列、および 2 つの列式に基づいてクラスター化されます。

例 3:

```
CREATE TABLE T1(c1 DATE, c2 INT, c3 INT, c4 DOUBLE,
   c5 DOUBLE GENERATED ALWAYS AS (c3 + c4),
   c6 INT GENERATED ALWAYS AS (MONTH(C1)))
      ORGANIZE BY DIMENSIONS (c2, c5, c6)
```

例 3 では、列 c5 が c3 および c4 に基づく式であり、列 c6 は列 c1 を時間的に粗い細分性にロールアップします。このステートメントによって、列 c2、c5、および c6 の値に基づいて表がクラスター化されます。

生成列ディメンションに対する範囲照会

生成列ディメンションに対する範囲照会では単調列関数が必要です。生成列に対してディメンションの範囲述部を派生させるには、式が単調でなければなりません。生成列に対してディメンションを作成する場合、基本列に対する照会では、その生成列に対するブロック索引を利用することによりパフォーマンスが改善されます。ただし、1 つの例外があります。基本列 (日付など) に対する範囲照会で、ディメンション・ブロック索引による範囲スキャンを使用するには、 CREATE TABLE ステートメントで列を生成するために使用する式が単調でなければなりません。列式には任意の有効な式 (ユーザー定義関数 (UDF) を含む) を含めることができますが、その式が単調でない場合、基本列に対する述部のうち、照会を満たすためにブロック索引を使用できるのは等式述部または IN 述部だけです。

例えば、`month = INTEGER (date)/100` という生成列に対してディメンションが定義されている MDC があるとしましょう。ディメンション (month) に対する照会では、ブロック索引スキャンが可能です。基本列 (date) に対する照会でも、ブロック索引スキャンを実行することによってスキャン対象のブロックを限定してから、それらのブロックだけに含まれる行に対して、date に関する述部を適用することができます。

コンパイラーは、ブロック索引スキャンで使用する付加的な述部を生成します。例えば、次の照会の場合、

```
SELECT * FROM MDCTABLE WHERE DATE > '1999-03-03' AND DATE < '2000-01-15'
```

コンパイラーは、『month >= 199903』 かつ 『month <= 200001』 という、付加的な述部を生成します。これは、ディメンション・ブロック索引スキャンの述部として使用できます。結果ブロックのスキャンにおいては、オリジナルの述部がブロック中の行に適用されます。

非単調式の場合、そのディメンションに対して適用できるのは等式述部だけです。非単調関数の 1 つの例として、例 3 の列 c6 の定義に出てきた MONTH( ) があります。列 c1 が日付、タイム・スタンプ、または日付やタイム・スタンプを表す有効なストリング表記である場合、この関数は 1 から 12 までの範囲の整数値を戻し

ます。この関数の出力は決まりきったものですが、実際には以下のようなステップ関数と同じような出力 (つまり循環パターン) を生成します。

```
MONTH(date('01/05/1999')) = 1
MONTH(date('02/08/1999')) = 2
MONTH(date('03/24/1999')) = 3
MONTH(date('04/30/1999')) = 4
...
MONTH(date('12/09/1999')) = 12
MONTH(date('01/18/2000')) = 1
MONTH(date('02/24/2000')) = 2
...
```

この例の一連の日付は単調に大きくなっていますが、 MONTH(date) はそうではありません。より厳密に言うと、date1 が date2 よりも大きいとき、常に MONTH(date1) が MONTH(date2) より大きいか等しくなるとは限りません。単調性を考慮する必要があるのは、このような場合です。このような非単調性は許可されていますが、基本となる列の範囲述部がそのディメンションの範囲述部を生成できないという点で、ディメンションを制限してしまいます。しかし、式の範囲述部 (例えば `where month(c1) between 4 and 6`) は使用できます。こうすれば、開始キーを 4、終了キーを 6 として、ディメンションに関する索引を通常の方法で使用できます。

この関数を単調にするには、月の上位部として年を使用する必要があります。バージョン 9.2 の組み込み関数 INTEGER には、日付に関する単調式を定義するのに役立つ拡張機能があります。 INTEGER(date) は日付の整数表記を戻します。この表記をさらに分割して、年および月を表す表記を検出することができます。例えば、`INTEGER(date('2000/05/24'))` は `20000524` を戻し、 `INTEGER(date('2000/05/24'))/100 = 200005` となります。関数 INTEGER(date)/100 は単調です。

同様に、組み込み関数 DECIMAL および BIGINT にもまた、単調関数を導出できる拡張機能があります。 DECIMAL(timestamp) はタイム・スタンプの 10 進表記を戻します。この表記を単調式で使用して、月、日、時間、分などに関して単調増加する値を導出することができます。 BIGINT(date) は INTEGER(date) と同様に、日付に関する BIGINT 表記を戻します。

表において生成された列を作成するとき、あるいはディメンション節の式からディメンションを作成するときに、データベース・マネージャーは可能な限り式の単調性を判別します。 DATENUM( )、 DAYS( )、 YEAR( ) などの一部の関数は、単調性を保持する関数として認識されます。さらに、さまざまな数式 (例えば列や定数の除算、乗算、加算) が単調性を保持します。式の単調性が保持されないと DB2 が判断した場合、あるいは判別が不可能な場合には、ディメンションの基本列では等式述部のみを使用することができます。

MDC 表のためのロード考慮事項

データを定期的にデータウェアハウスにロールインする場合、 MDC 表の利点を活用することができます。 MDC 表では、ロードの際に空のブロックがまず再使用され、その後で、残りのデータ用に表を拡張して新しいブロックが追加されます。

あるデータの集合 (例えば 1 カ月分の全データ) を削除した後、ロード・ユーティリティーを使って翌月のデータをロールインすれば、 (コミット済み) 削除によって空になったブロックを再使用できます。バージョン 9.5 では、MDC ロールアウト

機能が据え置きクリーンアップとともに追加されています。ロールアウト (削除でもある) がコミットされた後、ブロックは解放されず、まだ再利用できません。索引に基づいてレコード ID (RID) を保守するための、バックグラウンド・プロセスが呼び出されます。保守が完了すると、ブロックは解放され、再利用できます。

データを MDC 表にロードした時点で、入力データはソートされた状態またはソートされていない状態のいずれかにすることができます。ソートされていない状態の場合には、次の点に注意してください。

- *util_heap_sz* 構成パラメーターの設定を大きくしてください。

  MDC 表のロード時のロード・ユーティリティーのパフォーマンスを改善するには、 *util_heap_sz* データベース構成パラメーター値を大きくしなければなりません。ユーティリティーで使用できるメモリーを増やすと、 mdc-load アルゴリズムのパフォーマンスが大きく向上します。こうすると、ロード・フェーズで実行されるデータのクラスタリング時に、ディスクの入出力を減らすことができます。 LOAD コマンドを使用して複数の MDC 表を同時にロードする場合には、それに応じて、*util_heap_sz* の値を大きくしなければなりません。
- LOAD コマンドの DATA BUFFER 節に指定する値を大きくしてください。

  この値を大きくすると、単一ロード要求に影響します。ユーティリティー・ヒープ・サイズは、複数の並行ロード要求の可能性に対応できるだけの大きさでなければなりません。 LOAD コマンドの DATA BUFFER オプションが指定される際には、この値も大きくしなければなりません。
- バッファー・プールのために使用されるページ・サイズは、 TEMPORARY 表スペースの最大ページ・サイズと同じでなければなりません。

  ロード・フェーズでは、ブロック・マップの保守のために余分のロギングが実行されます。割り振られるエクステントごとに、おおよそ 2 つの余分のログ・レコードがあります。パフォーマンスを良くするためには、このことを考慮に入れた値に *logbufsz* データベース構成パラメーターを設定する必要があります。

以下の制約事項はマルチディメンション・クラスタリング (MDC) 表へのデータのロード時に適用されます。

- LOAD コマンドの SAVECOUNT オプションはサポートされていません。
- これらの表は独自のフリー・スペースを管理するため、 `totalfreespace` ファイル・タイプ修飾子はサポートされていません。
- MDC 表には `anyorder` ファイル・タイプ修飾子が必要です。 `anyorder` 修飾子を使わないで MDC 表へのロードを実行すると、 `anyorder` 修飾子はユーティリティーによって暗黙的に使用可能になります。

MDC 表に LOAD コマンドを使用する場合には、ユニーク制約の違反は以下のように処理されます。

- ロードするデータと同じユニーク・キーを持つレコードが (ロード操作の開始前に既に) 表に存在する場合、元のレコードは残り、新規レコードが削除フェーズで削除されます。
- ロードするデータと同じユニーク・キーを持つレコードが (ロード操作の開始前には) 表に存在しない場合、ユニーク・キーとそれと重複する (同じユニーク・キ

ーを持つ) レコードの両方が表にロードされる場合には、レコードのうち 1 つだけがロードされ、その他のレコードは削除フェーズで除去されます。

注: どのレコードがロードされて、どのレコードが削除されるかを判別するための明示的な技法はありません。

ロードはブロック境界から始まるため、それは新しいセルに属するデータのため、または表の初期データのために使用するのが最善です。

すべての MDC 表にはブロック索引があるため、MDC ロード操作には常に構築フェーズがあります。

MDC 表のためのロギング考慮事項

以前に RID で索引付けされた列を新たにディメンションにして、ブロック索引を使って索引付けすれば、索引の保守とロギングが大きく削減されます。

ブロック全体の最後のレコードが削除された場合に限って、データベース・マネージャーは BID をブロック索引から除去し、この索引操作をログに記録する必要があります。同様に、新しいブロックにレコードが挿入された場合 (つまり、それが論理セルの最初のレコードである場合、またはブロックいっぱいになった論理セルに挿入する場合) に限って、データベース・マネージャーは BID をブロック索引に挿入し、その操作をログに記録する必要があります。ブロック内のレコード・ページの数は 2 から 256 までであるため、このようなブロック索引の保守およびロギングは比較的小規模です。表と RID 索引の追加と削除は、引き続きログに記録されます。ロールアウト削除の場合、削除レコードはログに記録されません。その代わり、レコードの入ったページはページのパーツを再フォーマットすることによって空にされます。再フォーマットされたパーツに対する変更はログに記録されますが、レコード自体はログに記録されません。

MDC 表のためのブロック索引

MDC 表のディメンションを定義すると、ディメンション・ブロック索引が作成されます。さらに、マルチディメンションが定義されている場合には、複合ブロック索引も作成されることがあります。しかし、MDC 表にディメンションをただ 1 つだけ定義した場合、 DB2 はブロック索引を 1 つだけ作成します。これは、ディメンション・ブロック索引および複合ブロック索引として機能します。

同様に、列 A および (列 A、列 B) をディメンションとする MDC 表を作成した場合、 DB2 は列 A に関するディメンション・ブロック索引と、 (列 A、列 B) に関するディメンション・ブロック索引を作成します。複合ブロック索引は表内のすべてのディメンションに関するブロック索引であるため、 (列 A、列 B) に関するディメンション・ブロック索引もまた複合ブロック索引として機能します。

さらに複合ブロック索引は、照会処理において、表内の特定の複数のディメンション値を持つデータにアクセスするためにも使われます。複合ブロック索引のキー部分の順序は、その使用法、または照会処理の適用度に影響する場合があることに注意してください。キー部分の順序は、 MDC 表の作成時に使われる ORGANIZE BY DIMENSIONS 節全体の中にある列の順序によって決定されます。例えば、次のステートメントを使用して表が作成された場合、

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)
```

この場合、複合ブロック索引は列 (c4,c3,c1,c2) で作成されます。c1 は DIMENSIONS 節で 2 度指定されていますが、複合ブロック索引のキー部分では一度しか使用されず、最初に検出された順序で使用されることに注意してください。挿入処理の場合、複合ブロック索引内のキー部分の順序は無関係ですが、照会処理の場合には順序が影響を与える可能性があります。したがって、複合ブロック索引の列の順序を (c1,c2,c3,c4) とする場合には、以下のステートメントを使って表を作成してください。

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
   ORGANIZE BY DIMENSIONS (c1, c2, (c3,c1), c4)
```

MDC 表のブロック索引

このトピックでは、ブロック索引を使用する MDC 表でレコードがどのように編成されるかを説明します。

65 ページの図 12 の MDC 表は、『Region』 と 『Year』 の値が同じであるレコードをまとめて、いくつかの別々のブロック、つまりエクステントとなるように、物理的に編成されています。1 つのエクステントは、ディスク上のいくつかの連続したページで構成されているため、それらのレコード・グループは、物理的に連続したデータとしてクラスター化されています。表の各ページはそれぞれちょうど 1 個のブロックに属しており、どのブロックもすべて同じサイズ (同じページ数) です。ブロックのサイズは表スペースのエクステント・サイズと同じなので、ブロックの境界はエクステントの境界と揃うようになっています。この例の場合、2 個のブロック索引が作成されており、 1 個は 『Region』 ディメンション、もう 1 個は 『Year』 ディメンションに関するものです。それらのブロック索引には、表の中のブロックのみに対するポインターが含まれています。 『Region』 ブロック索引により 『Region』 が 『East』 である全レコードをスキャンすると、条件を満たすブロックが 2 個見つかります。それらの 2 個のブロックには 『Region』 が 『East』 であるレコードのすべて、そしてそのようなレコードだけが含まれており、連続したページまたはエクステントからなるセット 2 個に対してクラスター化されます。同時に、またそれとは完全に独立して、『Year』 索引によって、 1999 年から 2000 年までのレコードをスキャンすると、条件を満たすブロックが 3 個見つかります。それら 3 個のブロックのそれぞれに対するデータ・スキャンでは、1999 年から 2000 年までのレコードがすべて、そしてそのようなレコードのみ戻され、それらのレコードは、各ブロック内で連続したページ上にクラスター化されています。

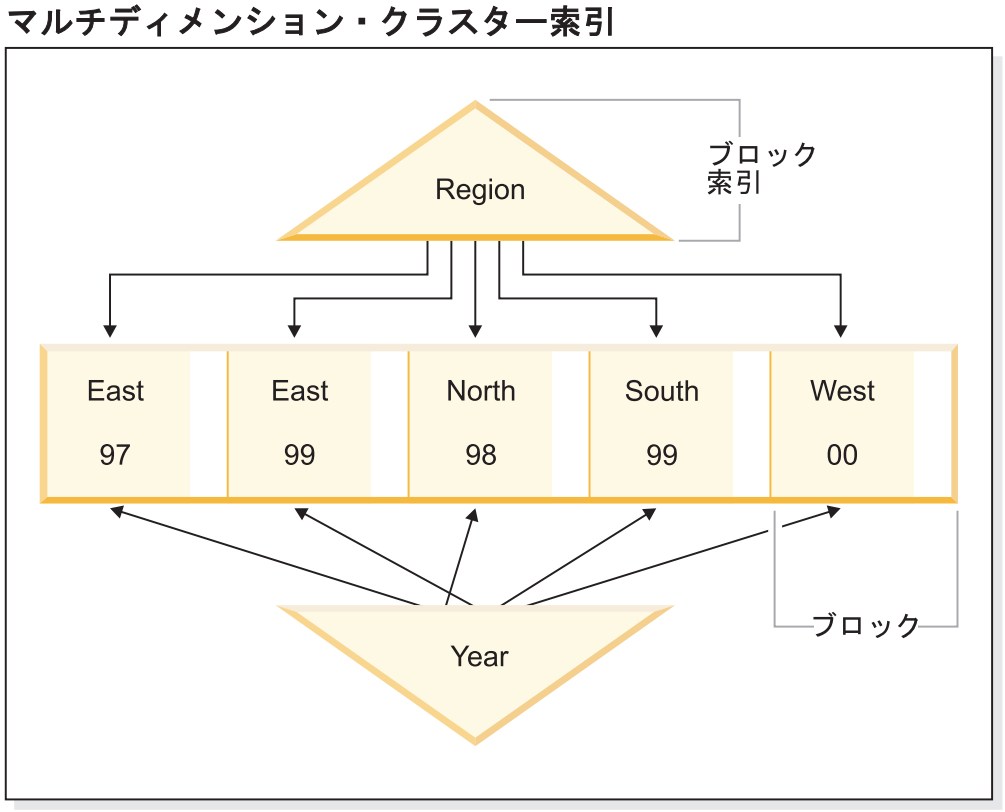


図 12. マルチディメンション・クラスター化表

クラスター化に関するこれらの改善点以外に、 MDC 表には次のようなメリットがあります。

- ブロック索引は、レコード・ベースの索引と比較して大幅にサイズが小さいため、プローブとスキャンはずっと高速になります。
- ブロック索引とそれに対応するデータ編成により、きめ細かい「データベース・パーティション除去」、あるいは選択的表アクセスが可能になります。
- ブロック索引を利用した照会では、索引サイズが小さく、ブロックのプリフェッチが最適化されており、および対応するデータのクラスター化が保証されているというメリットがあります。
- 一部の照会においては、ロッキングおよび述部評価が少なくなります。
- ブロック索引では、ロギングおよび保守のためのオーバーヘッドが非常に少なくて済みます。ブロック索引の更新が必要になるのは、ブロックに最初のレコードを追加した時点か、ブロックから最後のレコードが除去された時点だけだからです。
- ロールインされるデータは、以前にロールアウトされたデータによって残された連続したスペースを再利用できます。

注: 単一ディメンションで定義された MDC 表であっても、 MDC のこれらの属性によるメリットがあります。また、クラスター索引を伴う通常表のための発展的な代替手段となり得ます。それは、ワークロードを構成する照会、表のデータの性質や分布など、数多くの要因に基づいて決定する必要があります。 45 ページの『MDC 表のディメンションの選択』 および「設計アドバイザー」を参照してください。

表を作成するとき、1 つまたは複数のキーを、データ・クラスター化に関連したディメンションとして指定することができます。それぞれの MDC ディメンションは、通常の索引キーと同様に 1 つまたは複数の列から構成されます。指定したそれぞれのディメンションごとにディメンション・ブロック索引 が自動的に作成され、オプティマイザーはこれを使用して、各ディメンションのデータに素早く効率的にアクセスします。さらに、すべてのディメンションにわたってすべての列を含む複合ブロック索引 も自動的に作成されます。これは、挿入および更新アクティビティーにおいてデータのクラスター化を維持するために使われます。複合ブロック索引が作成されるのは、単一のディメンションにすべてのディメンション・キー列が含まれないような場合のみです。また、複合ブロック索引は、列ディメンションの一部または全部の値を満たすデータに効率的にアクセスするために、オプティマイザーによって選択されることがあります。

注: 照会処理におけるこの索引の便利さは、そのキー部分の順序に依存します。キー部品の順序は、 CREATE TABLE ステートメントの ORGANIZE BY 節で指定されるディメンションをパーサーが解析する際に、パーサーが検出する列の順序によって決まります。詳しくは、 63 ページの『MDC 表のためのブロック索引』 を参照してください。

ブロック索引は、レコードではなくブロックを指すという点を除けば、その構造は通常の索引と同じです。ブロック索引は、ブロック・サイズに 1 ページの平均レコード数を乗算した分だけ通常の索引よりも小さくなります。 1 ブロック内のページ数は表スペースのエクステント・サイズと同じであり、 2 ページから 256 ページの範囲になります。ページ・サイズとしては 4 KB、8 KB、16 KB、または 32 KB が可能です。

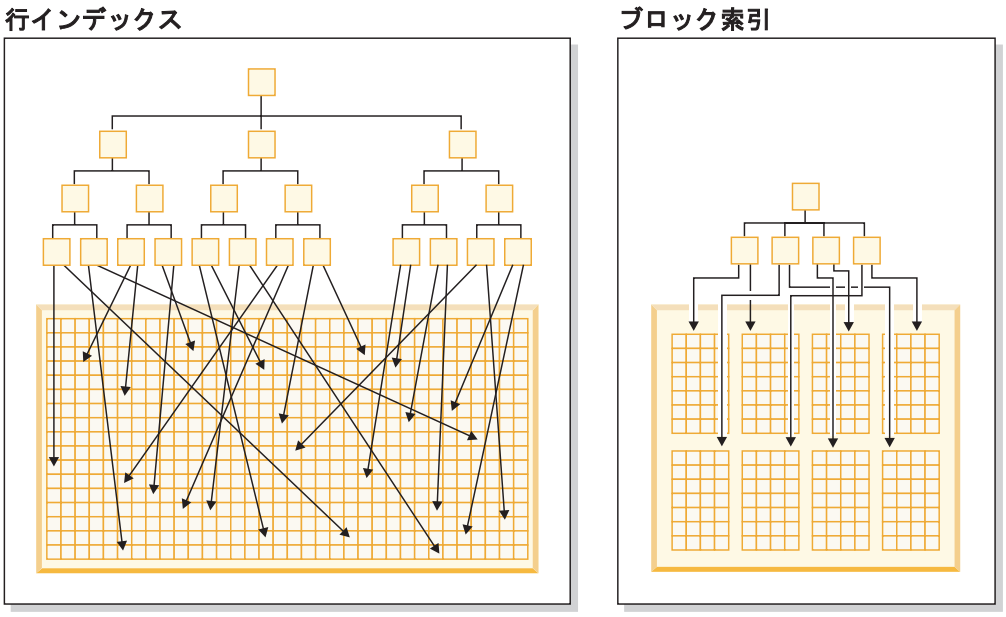


図 13. 行索引とブロック索引の違い

図 13 からわかるように、ブロック索引では、1 行に 1 つの項目ではなく、 1 ブロックに対して 1 個の索引項目が対応します。そのため、ブロック索引では、ディスク使用量が大幅に少なくなり、データ・アクセスははるかに高速です。

MDC 表では、複数のディメンション値のあらゆるユニークな組み合わせが論理セルを形成します。これは、物理的には 1 つ以上のページ・ブロックで構成される場合があります。論理セルには、その論理セルと同じディメンション値のレコードを入れるのに必要な数のブロックだけが含まれています。特定の論理セルと同じディメンション値の表の中にレコードが何も含まれていないなら、その論理セルにはブロックが割り振られません。特定のディメンション・キー値のデータを含むブロックの集合は、*スライス* と呼ばれます。

シナリオ: マルチディメンション・クラスタリング (MDC) 表

MDC 表を扱うシナリオとして、全国的な小売店の販売データを記録するための『Sales』 という MDC 表があるとします。この表は、 『YearAndMonth』 (年月) および 『Region』 (地域) という 2 つのディメンションでクラスター化されます。この表のレコードはブロック内に保管されます。それらのブロックには、エクステントを入れるための十分な数の連続したディスク上のページが含まれます。

68 ページの図 14 では、1 つのブロックが長方形として表され、表内に割り振られたエクステントの論理順序に従ってブロックに番号が付いています。図の中のグリッド (格子) はこれらのブロックの論理データベース・パーティションを示し、それぞれの四角は論理セルを表します。 グリッド内の列または行は、特定のディメンションのスライスを示します。例えば、『Region』 列に値 'South-central' (中南部) が含まれるすべてのレコードは、グリッドの 'South-central' 列として定義されたスライスに含まれるブロックの中にあります。実際、このスライス内の各ブロックには、 『Region』 フィールドが 'South-central' であるレコードのみが含まれます。このように、 『Region』 フィールドが 'South-central' であるレコードを含むブロックのみが、このスライス (つまりグリッドの列) に含まれます。

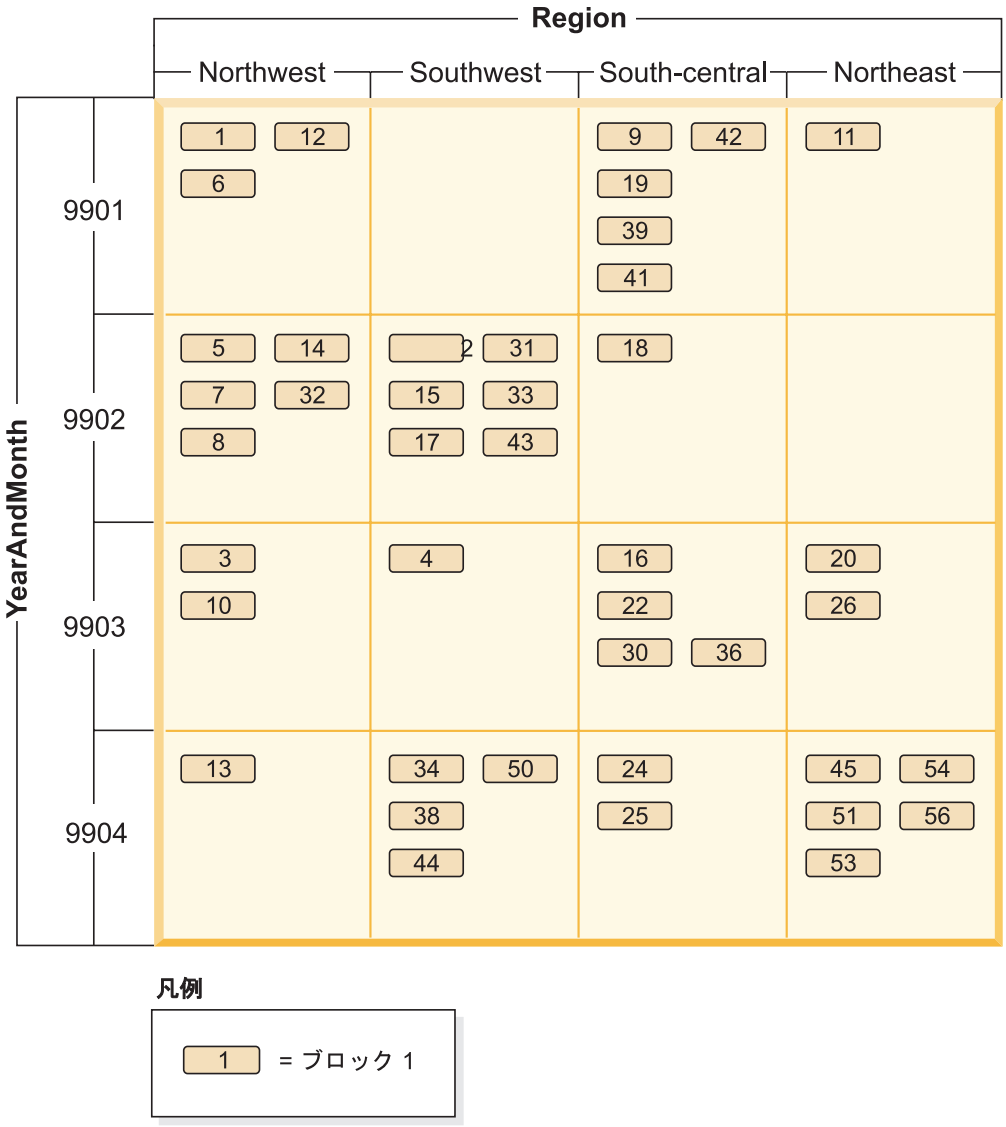


図 14. ディメンション 'Region' および 'YearAndMonth' が含まれるマルチディメンション表 Sales

スライスを構成するブロックを判別するために、あるいは特定のディメンション・キー値を持つすべてのレコードを含むブロックを判別するために、表の作成時に、各ディメンションごとにディメンション・ブロック索引が自動的に作成されます。

69 ページの図 15 では、ディメンション 『YearAndMonth』 (年月) および 『Region』 (地域) に関するディメンション・ブロック索引がそれぞれ作成されています。それぞれのディメンション・ブロック索引の構造は従来の RID 索引と同様です。ただし、リーフ・レベルでは、キーがレコード ID (RID) でなくブロック ID (BID) をポイントします。 RID は、物理ページ番号とスロット番号 (レコードの存在するページ上のスロット) によって、表の中のレコードの位置を特定します。BID は、そのエクステントの最初のページの物理ページ番号、およびダミー・スロット (0) によってブロックを表します。ブロック内のすべてのページはそこから始まって物理的に連続しており、ブロックのサイズがわかっているので、この BID を使用することにより、ブロック内のすべてのレコードを検索できます。

スライス (つまり、ディメンションにおいて特定のキー値を持つすべてのレコード・ページを含むブロックの集まり) は、関連するディメンション・ブロック索引において、そのキー値に関する BID リストによって表されます。

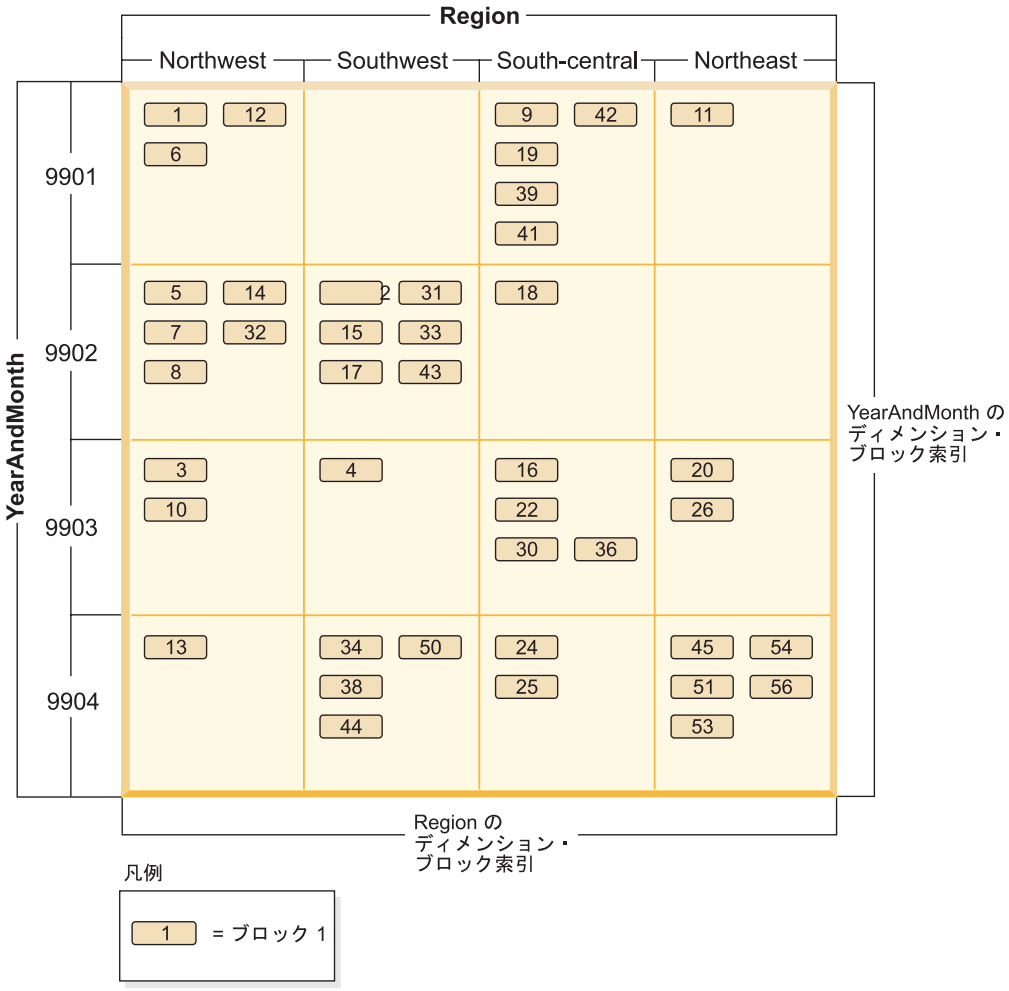


図 15. 'Region' および 'YearAndMonth' のディメンションを含む Sales 表とディメンション・ブロック索引

図 16 は、 『Region』 に関するディメンション・ブロック索引のキーを例示しています。このキーは、キー値 ('South-central') と BID リストで構成されています。各 BID にはブロックのロケーションが含まれています。 図 16 にリストされているブロック番号は、 Sales 表 (68 ページの図 14 を参照) のグリッドにある 'South-central' スライスに含まれる番号と同じです。

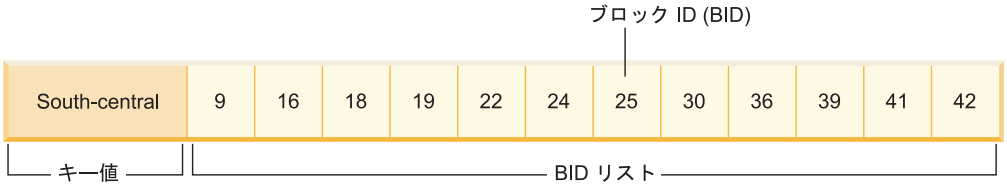


図 16. 'Region' に関するディメンション・ブロック索引のキー

同様に、ディメンション 『YearAndMonth』 の値が '9902' のすべてのレコードを含むブロックをリストするには、この値を 『YearAndMonth』 ディメンション・ブロック索引で検索します (図 17 を参照)。

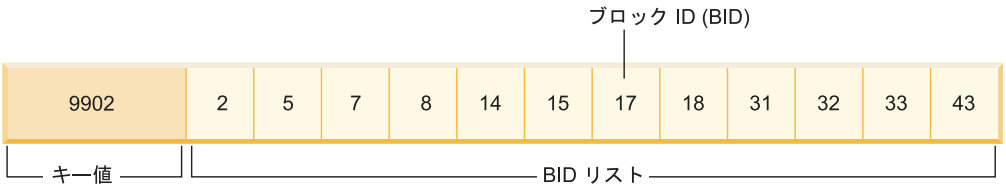


図 17. 'YearAndMonth' に関するディメンション・ブロック索引のキー

MDC 表のブロック索引と照会のパフォーマンス

MDC 表のブロック索引のいずれかに関するスキャンでは、表のうち、指定されたディメンション値のデータを含むことが保証されている連続したページの集合に、各ブロック ID (BID) が対応しているため、クラスター化データ・アクセスが提供されます。さらに、ディメンションまたはスライスには、他のディメンションまたはスライスのクラスター因子に関係なく、そのブロック索引によって互いに独立してアクセスできます。これにより、マルチディメンション・クラスター化のマルチディメンション性が実現されます。

ブロック索引アクセスを利用する照会では、パフォーマンスを向上させるさまざまな要素があります。第 1 の点として、ブロック索引は通常の索引に比べてはるかに小さいため、ブロック索引のスキャンは非常に効率的です。第 2 の点として、データ・ページのプリフェッチは、ブロック索引を使用した順次検出に依存していません。 DB2 は索引を先読みし、大ブロック入出力を使用してブロックのデータ・ページをメモリーにプリフェッチすることにより、スキャンで表の中のデータ・ページにアクセスする際に入出力が発生しないようにします。第 3 の点として、表のデータが連続したページ上でクラスター化されているため、入出力が最適化され、結果セットが表の選択した部分に配置されます。ブロック・ベースのバッファー・プールが使用されていて、そのブロック・サイズがエクステント・サイズの場合、ディスク上の連続したページから MDC ブロックがプリフェッチされて、メモリー内の連続ページに入れられます。そのようにして、クラスター化によるパフォーマンスがさらに向上します。最後の点として、各ブロックのレコードは、そのデータ・ページの小規模なリレーショナル・スキャンを使用して取り出されます。これは、RID ベースの取り出しに比べて、データ・スキャンのためのより高速な方法であることが少なくありません。

照会でブロック索引を使用するなら、表のうち、特定のディメンション値または値の範囲を含む部分に限定することができます。これにより、きめ細かい「データベース・パーティション除去」、つまりブロック除去が提供されます。その場合、他の照会、ロード、挿入、更新、および削除の操作では、この照会のデータ・セットとのやり取りをすることなく他のブロックにアクセスできるため、表の並行性がさらに高くなる可能性があります。

さらに、Sales 表が 3 つのディメンションに基づいてクラスター化されている場合には、表のすべてのディメンションのサブセットに対する照会を満たすようなレコードを含むブロック・セットを見つけるために、個々のディメンション・ブロック索引を使用することもできます。表のディメンションが 『YearAndMonth』、『Region』、および 『Product』 (製品) であれば、図 18 のように、これを論理キューブと考えることができます。

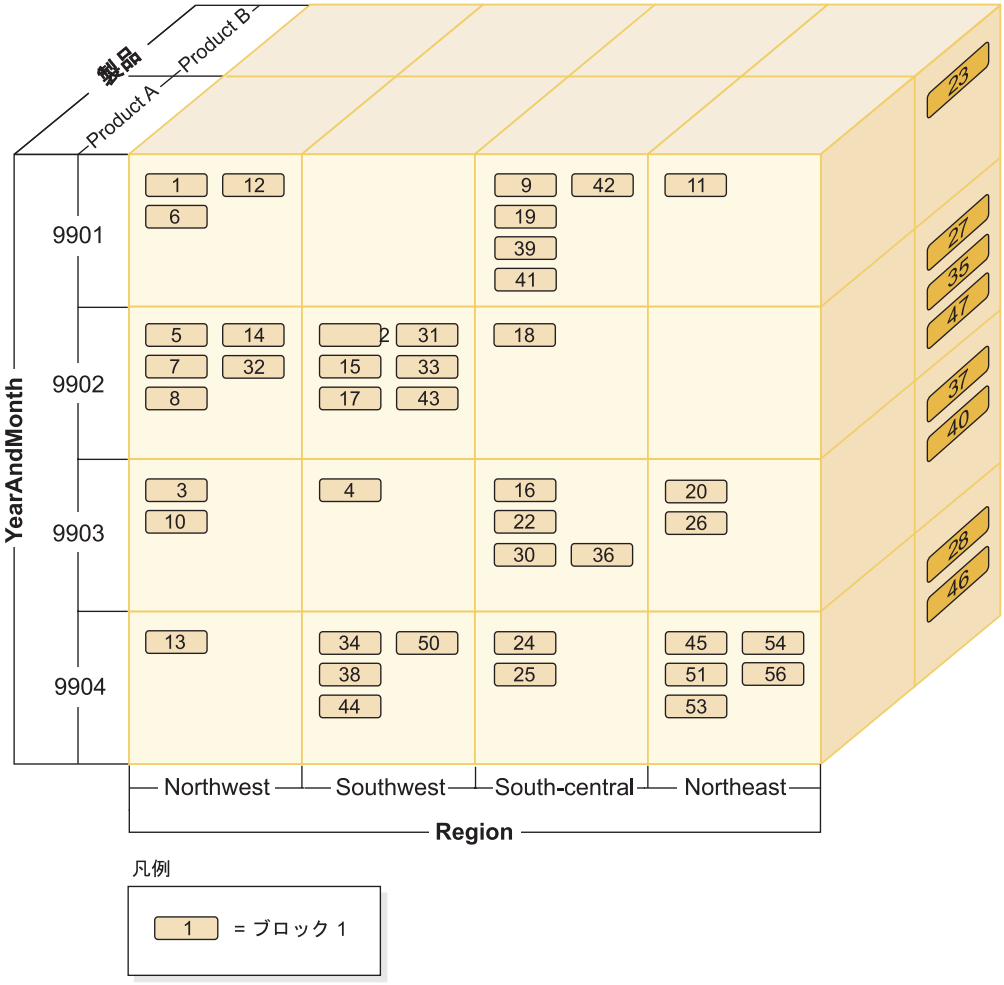


図 18. ディメンション 'Region'、'YearAndMonth'、および 'Product' が含まれるマルチディメンション表

図 18 に示されている MDC 表に対して、4 個のブロック索引が作成されます。それぞれのディメンション 『YearAndMonth』、『Region』、および 『Product』 に対応するブロック索引が 3 個、そしてすべてのディメンション列をキーとするブロック索引が 1 個です。 『Product』 が 『ProductA』、かつ 『Region』 が 『Northeast』 であるすべてのレコードを取り出す場合、データベース・マネージャーはまず 『Product』 ディメンション・ブロック索引から ProductA キーを検索します。 (72 ページの図 19 を参照。)その後、データベース・マネージャーは 『Region』 ディメンション・ブロック索引で 『Northeast』 キーを検索することにより、 『Region』 が 『Northeast』 であるすべてのレコードを含むブロックを判別します。(72 ページの図 20 を参照。)

図 19. 'Product' に関するディメンション・ブロック索引のキー

図 20. 'Region' に関するディメンション・ブロック索引のキー

論理 AND および論理 OR の演算子を使用すれば、複数のブロック索引スキャンを組み合わせることができます。また、スキャンするブロックの結果リストでも、クラスター化データ・アクセスが提供されます。

上の例で、 2 つのディメンション値を持つすべてのレコードを含むブロックのセットを見つけるには、この 2 つのスライスの交点を検出する必要があります。そのためには、 2 つのブロック索引キーの BID リストに対して論理 AND 演算を使用します。共通している BID 値は 11、20、26、45、54、51、53、および 56 です。

次の例は、2 ディメンションが関係する述部を含む照会において、ブロック索引での論理 OR 演算の使用方法を示すものです。 73 ページの図 21 では、『Colour』 および 『Nation』 という 2 つのディメンションを含む MDC 表を使用していることが前提になっています。目標は、この MDC 表の中から、『Colour』 が 『blue』 であるか、または 『Nation』 の名前が 『USA』 であるという条件を満たすレコードをすべて検索することです。

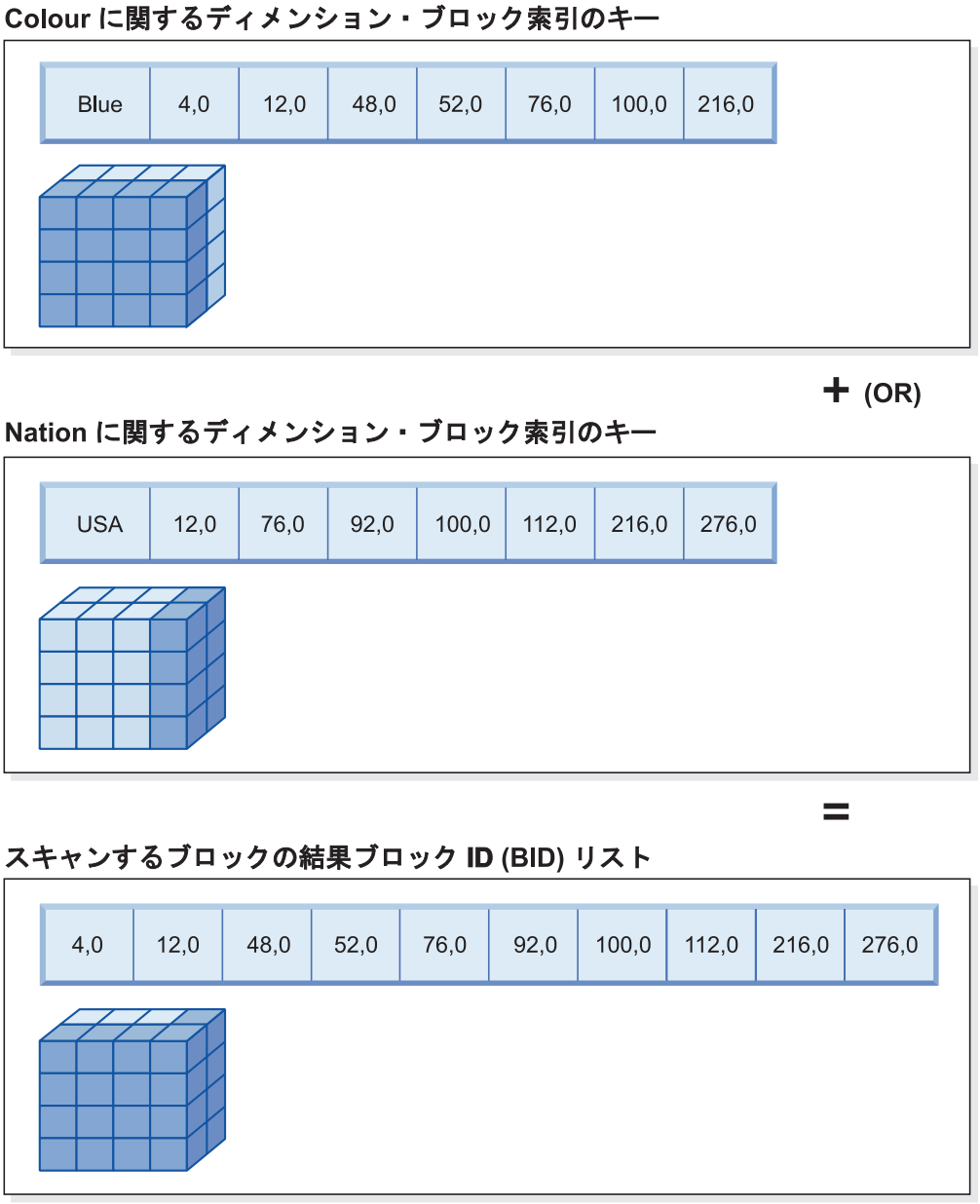


図 21. ブロック索引での論理 OR 演算の使用方法

この図には、2 つの別個のブロック索引スキャンの結果を結合して、述部制約を満たす値の範囲を決定する方法が示されています。 (数値は、レコード ID (RID)、スロット・フィールドを示します。)

SELECT ステートメントの述部に基づいて、 2 個の別個のディメンション・ブロック索引スキャンが実行されます。 1 つは blue スライス、もう 1 つは USA スライスに対してです。 2 つのスライスの論理和を得て、 2 つのスライス内のブロックの和集合 (重複ブロックの除去を含む) を求めるためにメモリー内で論理 OR 演算が実行されます。

データベース・マネージャーにスキャン対象のブロックのリストがある場合、データベース・マネージャーは各ブロックについて小規模なリレーショナル・スキャンを実行できます。ブロックのプリフェッチを実行することができます。各ブロック

はディスク上のエクステントとして保管されており、単体としてバッファー・プールに読み取られるので、実行される入出力処理は各ブロックごとにただ 1 つだけです。データに述部を適用することが必要なら、ブロック内のすべてのレコードのディメンション・キー値は同じであることが確実なため、ディメンション述部で必要なのは単にブロック内の 1 レコードの述部を再適用することだけです。他の述部が存在する場合、データベース・マネージャーはブロック内の残りのレコードに関してそれを検査するだけです。

MDC 表では、通常の RID ベースの索引もサポートされています。 RID とブロック索引は論理 AND 演算または論理 OR 演算で索引と結合できます。ブロック索引によりオプティマイザーは、アクセス・プランの選択肢が増えます。従来のアクセス・プラン (RID スキャン、結合、表スキャンなど) も使用できます。ブロック索引プランは、特定の照会に関して可能性のある他のすべてのアクセス・プランと共にコスト計算され、コストが最低のプランがオプティマイザーによって選択されます。

DB2 Design Advisor では、MDC 表に対して RID ベースの索引を推奨したり、表に対して MDC ディメンションを推奨したりできます。

INSERT 操作中に自動的にクラスター化を維持する

複合ブロック索引を使用すると、 MDC 表のデータ・クラスタリングが自動的に維持されます。これは、INSERT 操作の際、表のディメンションに基づいて物理的なデータ・クラスタリングを動的に管理および保守するために使われます。

この複合ブロック索引には、レコードを含む表の各論理セルに関してのみ、キーが存在します。したがって、INSERT 中にこのブロック索引が使用されることにより、論理セルが表の中に存在するかどうかが短時間で効率的に判別され、存在する場合には、そのセルの特定のディメンション値セットを含むレコードが含まれるブロックが正確にどれであるかが判別されます。

挿入操作が実行されると、

- 挿入するレコードのディメンション値に対応する論理セルに対して、複合ブロック索引がプローブされます。
- 論理セルのキーが索引内にあるなら、そのブロック ID (BID) のリストにより、表のうち、論理セルと同じディメンション値のブロックの完全なリストが提供されます。 (75 ページの図 22 を参照。) これにより、表の中で、レコードを挿入するためのスペースを検索する対象となるエクステントの数が限定されます。
- 論理セルのキーが索引内にない場合、またはそれらの値を含むエクステントに余地がない場合、新しいブロックがその論理セルに割り当てられます。可能なら、新しいページ・エクステント (新しいブロック) で表を拡張する前に、表の中の空ブロックが再利用されます。

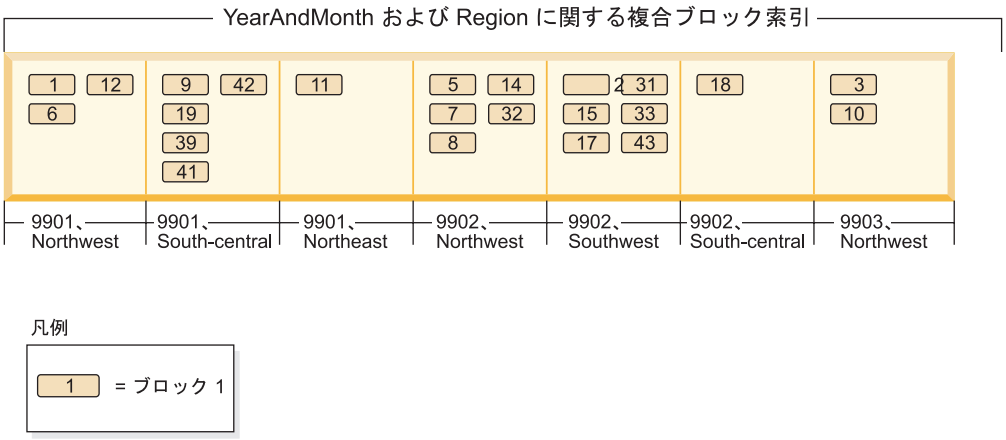


図 22. 'YearAndMonth' および 'Region' に関する複合ブロック索引

特定のディメンション値のデータ・レコードは、その値を含むすべてのレコードだけを含むブロック・セットの中にあることが保証されています。 ブロックは、ディスク上の連続したページで構成されています。そのため、それらのレコードへのアクセスは順次アクセスであり、クラスター化が提供されます。そのレコードと同じディメンション値のセルのブロックにのみレコードが挿入されるようにすることにより、そのクラスター化が自動的に維持されます。論理セル中の既存のブロックに余地がないなら、空ブロックが再利用されるか、または新しいブロックが割り振られてその論理セルのブロック・セットに追加されます。あるブロックにデータ・レコードがなくなって空になると、そのブロック ID (BID) がブロック索引から除去されます。それにより、そのブロックはどの論理セル値とも関連がなくなり、将来別の論理セルによって再利用できるようになります。このように、セルとそれに関連するブロック索引項目は、表内に実際に存在するデータだけを入れるようにするため、必要に応じて表に動的に追加および削除されます。それを管理するために複合ブロック索引が使用されます。複合ブロック索引は、論理セル値を、それらの値のレコードが含まれるブロックにマップするからです。

クラスター化がこのようにして自動的に維持されるため、データの再クラスター化のために MDC 表の再編成が必要になることは決してありません。しかし、スペースの再利用のために再編成を使用することは可能です。例えば、セルに数多くの低密度のブロックがあって、もっと少ないブロックでもデータが入る場合、あるいはオーバーフロー・レコードがある場合には、表を再編成することにより、各論理セルに属するレコードを圧縮すると共に、ブロックが必要最小限の数になるようにし、オーバーフロー・レコードを除去することができます。

以下の例は、照会処理のために複合ブロック索引を使用する方法を示しています。例えば、図 22 の表の中で、『Region』 が 'Northwest' で 『YearAndMonth』 が '9903' のレコードをすべて検出する場合、データベース・マネージャーはキー値 9903, Northwest を複合ブロック索引内で検索します (76 ページの図 23 を参照)。キーはキー値 (つまり '9903,Northwest') と BID リストで構成されます。ここでリストされている BID は 3 と 10 のみですが、実際、この 2 つの値を持つレコードを含むブロックは、 Sales 表の中に 2 つしかありません。

図 23. 'YearAndMonth' および 'Region' に関する複合ブロック索引のキー

挿入の際に複合ブロック索引がどのように使われるかを理解するために、ディメンション値が 9903 および Northwest である追加のレコードを挿入する場合を考えてください。データベース・マネージャーはこのキー値を複合ブロック索引内で検索し、ブロック 3 および 10 の BID を検出します。これらのブロックに、この 2 つのディメンション・キー値を持つレコードがすべて含まれます (他のブロックには含まれません)。利用可能なスペースがあるなら、データベース・マネージャーは新しいレコードをそれらのブロックのいずれかに挿入します。これらのブロックのどのページにもスペースがない場合、データベース・マネージャーは新しいブロックを表に割り振るか、表内のすでに空になっているブロックを使用します。この例の場合、ブロック 48 が現在どの表によっても使用されていないことに注意してください。データベース・マネージャーはレコードをブロックに挿入し、そのブロックの BID を複合ブロック索引と各ディメンション・ブロック索引に追加することによって、そのブロックを現在の論理セルに関連付けます。ブロック 48 を追加した後のディメンション・ブロック索引のキーが、 図 24 に図示されています。

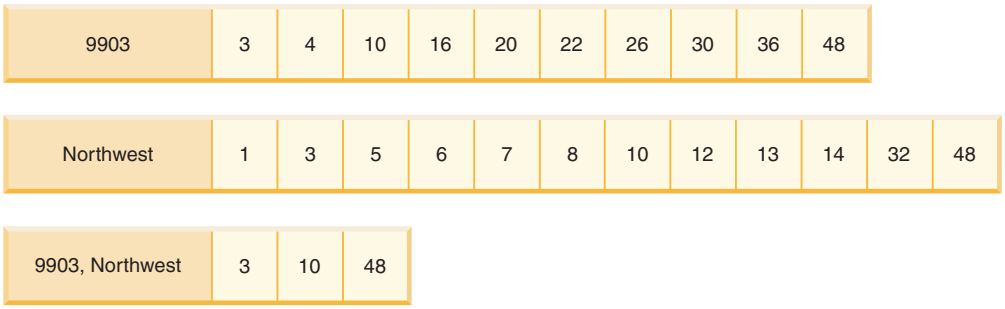


図 24. ブロック 48 追加後のディメンション・ブロック索引のキー

MDC 表のブロック・マップ

ブロックが空になった場合には、その BID がブロック索引から除去されることにより、その現在の論理セル値との関連が解除されます。そのブロックは、別の論理セルによって再利用できるようになります。それにより、新しいブロックを追加して表を拡張する必要が少なくなります。

新しいブロックが必要になった場合には、以前に空になったブロックを、表から検索することなく短時間で見つかるようになっていなければなりません。

ブロック・マップは、MDC 表の中から空のブロックを検索するために使用される構造です。ブロック・マップは、別個のオブジェクトとして格納されます。

- SMS の場合、別個の .BKM ファイルとして
- DMS の場合、オブジェクト表の中の新しいオブジェクト記述子として

ブロック・マップは、表の各ブロックごとに 1 個ずつの項目を含む配列です。各項目は、1 つのブロックの一連の状況ビットで構成されます。

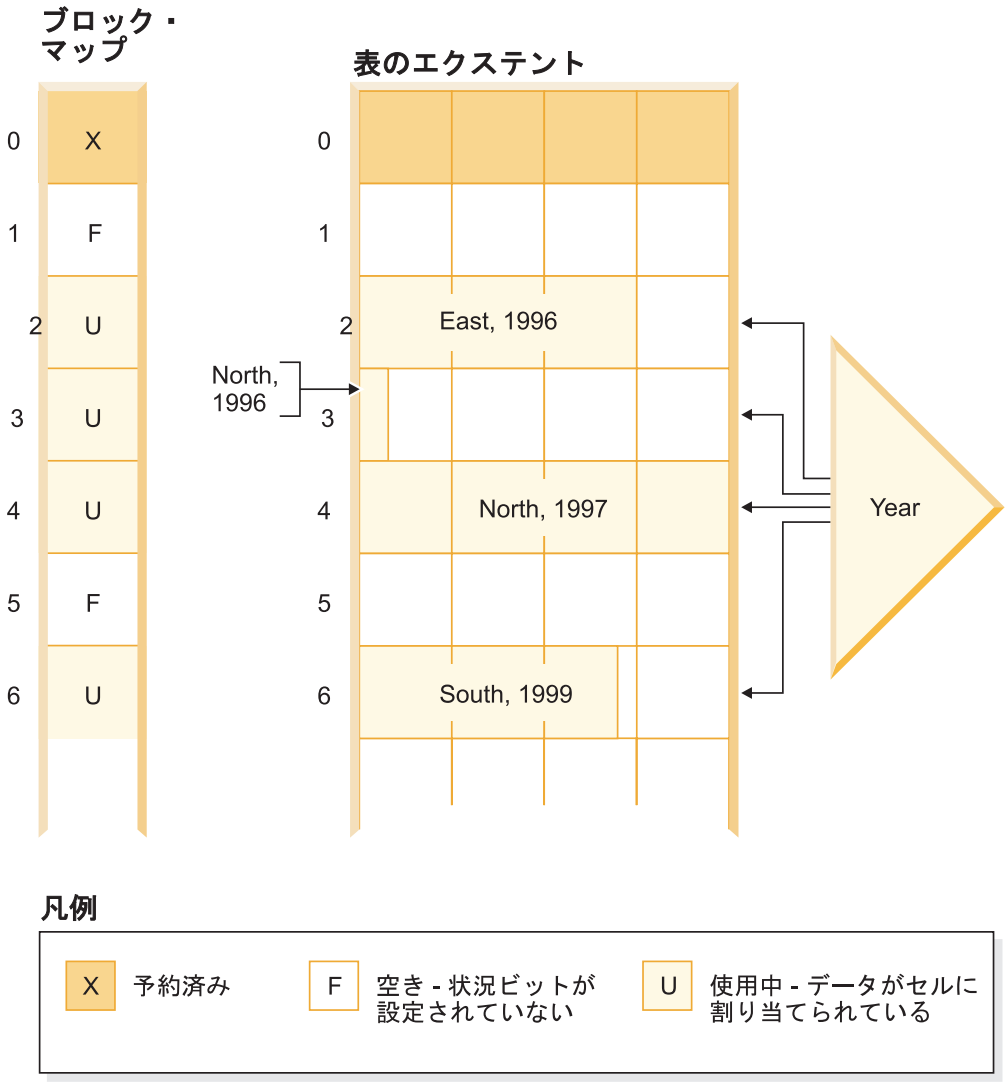


図 25. ブロック・マップの動作

図 25 の左側には、表の各ブロックごとのさまざまな項目を含むブロック・マップ配列が示されています。右側には、表の各エクステントの使用状況が示されています。空きはいくらかありますが、ほとんどは使用中です。レコードは、ブロック・マップの中で使用中のマークが付けられたブロックだけに入っています。単純にするため、この図には 2 つのディメンション・ブロック索引のうち 1 つだけが示されています。

注:

1. ブロック索引に含まれるポインターは、ブロック・マップの中で IN USE とマークされているブロックに対するものだけです。
2. 最初のブロックは予約済みです。このブロックには、表のシステム・レコードが含まれています。

セルの中で使用できる空きブロックを見つけるのは簡単です。ブロック・マップをスキャンして FREE ブロック、つまりどのビットもセットされていないブロックを検索するだけです。

また、表スキャンでは、現在データが含まれているエクステントだけにアクセスするためにもブロック・マップが使用されます。使用中でないエクステントは、表スキャンに含める必要がありません。例えばこの例 (77 ページの図 25) の場合、表スキャンは、最初の予約済みエクステントとその次の空エクステントをスキップして、表の 3 番目のエクステント (エクステント 2) から始まります。表のブロック 2、3、および 4 をスキャンした後、次のエクステントをスキップし (そのエクステントのデータ・ページには何も触らない)、その次からスキャンを続けます。

MDC 表からの削除

MDC 表でレコードを削除する場合、それがブロック内の最後のレコードでないなら、データベース・マネージャーは単にそのレコードを削除し、その表にレコード・ベースの索引が定義されているならそこからそのレコードの RID を削除します。

しかし、ブロック内の最後のレコードを削除する場合、データベース・マネージャーは、その IN_USE 状況ビットを変更して、そのブロックの BID をすべてのブロック索引から削除することによって、ブロックを解放します。また、レコード・ベースの索引も存在するなら、RID をそこから削除します。

注: したがって、ブロック索引の項目の削除は、ブロックが完全に空である場合にのみブロック全体について 1 回しか必要ではありません。レコード・ベースの索引で削除する行ごとに実行する必要はありません。

MDC 表の更新

MDC 表の場合、非ディメンション値の更新は、通常表の場合のように、その同じ場所で実行されます。その更新操作で可変長列が影響を受け、そのレコードがページに入らなくなる場合には、十分なスペースのある別のページが検索されます。

その新しいページの検索は、まず同じブロックから開始されます。そのブロックにスペースがないなら、新しいレコード挿入のアルゴリズムが使用されて、十分なスペースを含む論理セルの中からページが検索されます。セル内にスペースがないために新しいブロックをセルに追加することが必要になるのでない限り、ブロック索引を更新する必要はありません。

ディメンション値の更新操作では、レコードの属する論理セルが変わるため、現在のレコードを削除した後、変更後のレコードを挿入するという操作として処理されます。現在のレコードを削除するとブロックが空になる場合には、ブロック索引を更新する必要があります。同じように、新しいレコードの挿入操作で新しいブロックに挿入することが必要な場合にも、ブロック索引を更新する必要があります。

ブロック索引を更新することが必要なのは、ブロックに最初のレコードを挿入する時点と、ブロックから最後のレコードを削除する時点だけです。したがって、ブロック索引の保守とロギングに関連した索引オーバーヘッドは、通常の索引に関連した索引オーバーヘッドに比べてずっと小さくなります。通常の索引としても可能な索引をブロック索引にするなら、そのようなブロック索引すべてに関して、保守とロギングのオーバーヘッドが大幅に削減されます。

MDC 表は既存の他の表と同様に扱われます。つまり、トリガー、参照整合性、ビュー、およびマテリアライズ照会表を MDC 表に対して定義することができます。

表パーティション化とマルチディメンション・クラスタリング表

表では、マルチディメンション・クラスタリングとパーティション化の両方ができます。マルチディメンション・クラスタリングとパーティション化の両方を行った表では、列は表パーティション化の範囲パーティション仕様と MDC キーの両方で使用されます。これは、どちらかが単独で機能するよりも、データ・パーティションの細分性を良くし、ブロックを除去するのに役立ちます。また、表がパーティション化されるよりも、MDC キーに異なる複数の列を指定することが役立つ多くのアプリケーションがあります。なお、MDC はマルチディメンションですが、表パーティション化は複数列であることに注意してください。

主流の DB2 V9.1 データウェアハウスの特性

以下の推奨事項は、DB2 V9.1 にとって新しい、典型的で主流となるウェアハウスに焦点を合わせています。以下のような特性があると想定されています。

- データベースは、複数のマシンまたは複数の AIX 論理パーティションで実行される。
- データベース・パーティション・フィーチャー (DPF) が使用される (表は DISTRIBUTE BY HASH 節を使用して作成される)。
- 4 から 50 個以内のデータ・パーティションがある。
- MDC および表パーティション化が考慮されている表が、主なファクト表である。
- 表には 1 億から 1000 億以内の行がある。
- 新規データは、毎夜、毎週、毎月といった様々な時間フレームでロードされる。
- 毎日の取得ボリュームは、1 万から 1000 万以内のレコードである。
- データ・ボリュームが変化する。最大月は最小月のサイズの 5 倍になります。同様に、最大ディメンション (製品ライン、地域) は 5 倍のサイズ範囲となります。
- 1 から 5 年分の詳細データが保存される。
- 有効期限切れデータは、毎月または四半期ごとにロールアウトされる。
- 表は、広範囲の照会タイプを使用する。しかし、ワークロードはほとんど、OLTP ワークロードに比べると、以下の特性を持つ分析照会です。
  - 200 万行までの、より大きな結果セット
  - ほとんどまたはすべての照会が、基本表ではなくビューをヒットする
- 範囲 (BETWEEN 節)、リストにある項目などでデータを選択する SQL 節。

主流の DB2 V9.1 データウェアハウスのファクト表の特性

典型的なウェアハウスのファクト表は、以下の設計を使用すると考えられます。

- 月列にデータ・パーティションを作成する。
- ロールアウトする期間 (たとえば 1 カ月、3 カ月) ごとに、データ・パーティションを定義する。
- 日および 1 から 4 つ以内の追加のディメンション上に MDC ディメンションを作成する。典型的なディメンションは、製品ラインおよび地域です。
- すべてのデータ・パーティションおよび MDC クラスターが、すべてのデータベース・パーティションに広がっている。

MDC および表パーティション化は、重複した利点のセットを提供します。以下の表では、お客様の組織で必要になる可能性のあるものをリストし、以前に識別された特性を基にして、推奨される編成スキームを識別します。

表 7. MDC 表での表パーティション化の使用

課題	推奨スキーム	推奨
ロールアウト時のデータ可用性	表パーティション化	DETACH PARTITION 節を使用して、中断を最小限にしつつ、大量のデータをロールアウトします。
照会パフォーマンス	表パーティション化および MDC	MDC は複数ディメンションの照会に最善です。表パーティション化は、データ・パーティションの除去に有用です。
再編成の最小化	MDC	MDC はクラスタリングを維持し、再編成の必要性を削減します。
従来のオフライン期間内での、1 カ月かそれ以上のデータのロールアウト	表パーティション化	データ・パーティション化はこの必要を完全に処理します。 MDC は何も追加することはなく、これ自体にはあまり適していません。
マイクロ・オフライン期間 (1 分より小さい) 内での、1 カ月かそれ以上のデータのロールアウト	表パーティション化	データ・パーティション化はこの必要を完全に処理します。 MDC は何も追加することはなく、これ自体にはあまり適していません。
少しもサービスを損失することなく、照会をサブミットするビジネス・ユーザーが、表を完全に使用できるように保ちながら、1 カ月かそれ以上のデータをロールアウトする	MDC	MDC だけが、この必要をいくらか処理します。表パーティション化は、表が短期間オフラインになるので、適切ではありません。
毎日のデータのロード (ALLOW READ ACCESS または ALLOW NO ACCESS)	表パーティション化および MDC	この場合、MDC はほとんどの利点を提供します。表パーティション化は増分的な利点を提供します。

表 7. MDC 表での表パーティション化の使用 (続き)

課題	推奨スキーム	推奨
「継続的な」データのロード (ALLOW READ ACCESS)	表パーティション化および MDC	この場合、MDC はほとんどの利点を提供します。表パーティション化は増分的な利点を提供します。
「従来の BI」照会の場合の照会実行パフォーマンス	表パーティション化および MDC	MDC は、キューブ/複数ディメンションの照会に最適です。表パーティション化は、パーティションの除去の点で補助します。
再編成の必要を避けたり、タスクの実行に関連した手間を削減することにより、再編成の手間を最小化にする	MDC	MDC はクラスタリングを維持して REORG の必要性を削減します。 MDC が使用される場合、データ・パーティション化は増分的な利点を提供しません。しかし、MDC が使用されない場合には、範囲パーティション化が、パーティション・レベルで何らかの過程の粗いクラスタリングを維持することによって、REORG の必要を削減するのに役立ちます。

例 1:

キー列 YearAndMonth および Province のある表を考慮します。この表のプランとして妥当な方法は、1 つのデータ・パーティションあたり 2 カ月で、日付によってパーティション化することです。加えて、 37 ページの図 6 に示されているように、任意の 2 カ月の日付範囲内にある特定の州のすべての行は一緒にクラスター化されるので、Province によって編成することもできます。

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

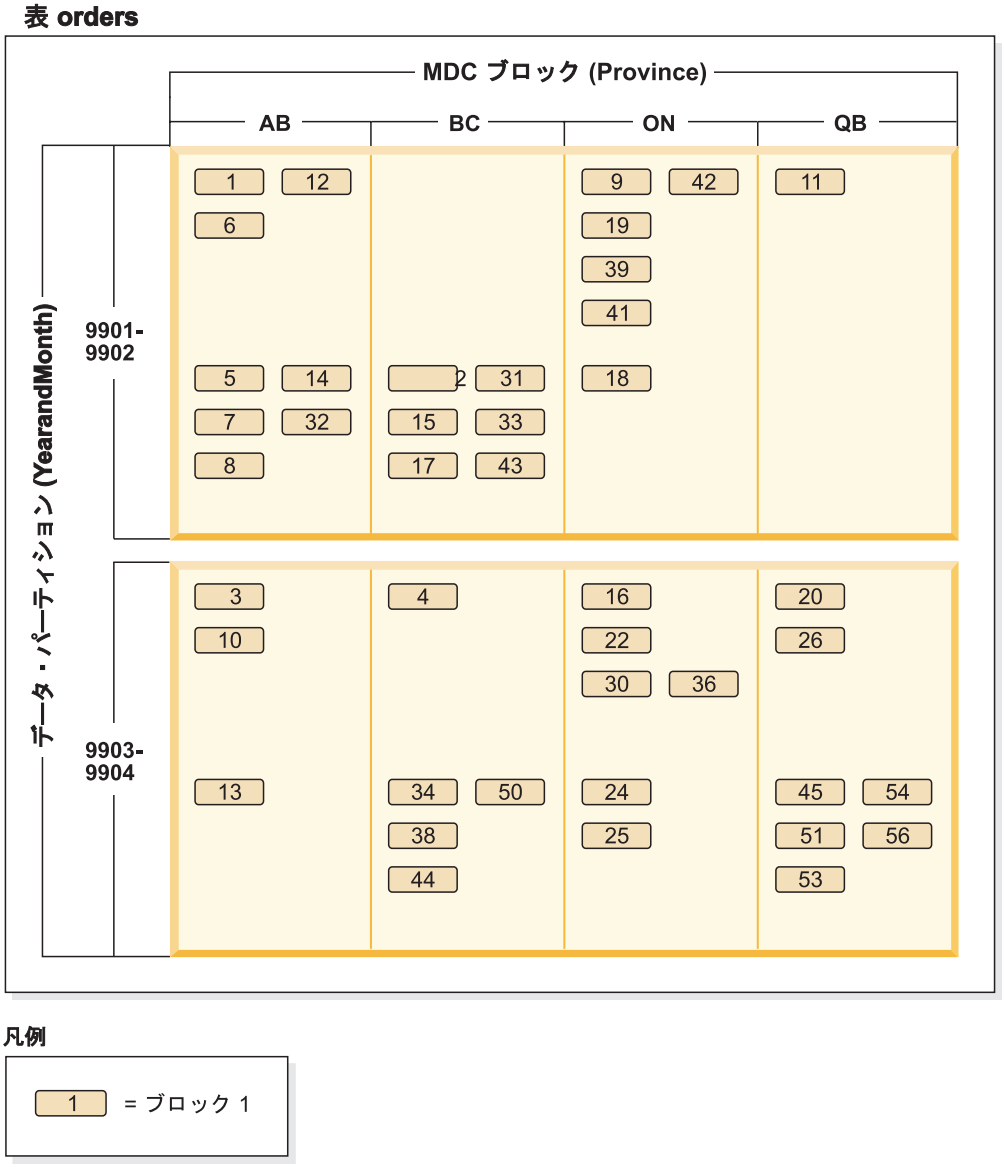


図 26. YearAndMonth によってパーティション化され、Province によって編成される表

例 2:

38 ページの図 7 に示されているように、YearAndMonth を ORGANIZE BY 節に追加することによって、より良い細分化を行うことができます。

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

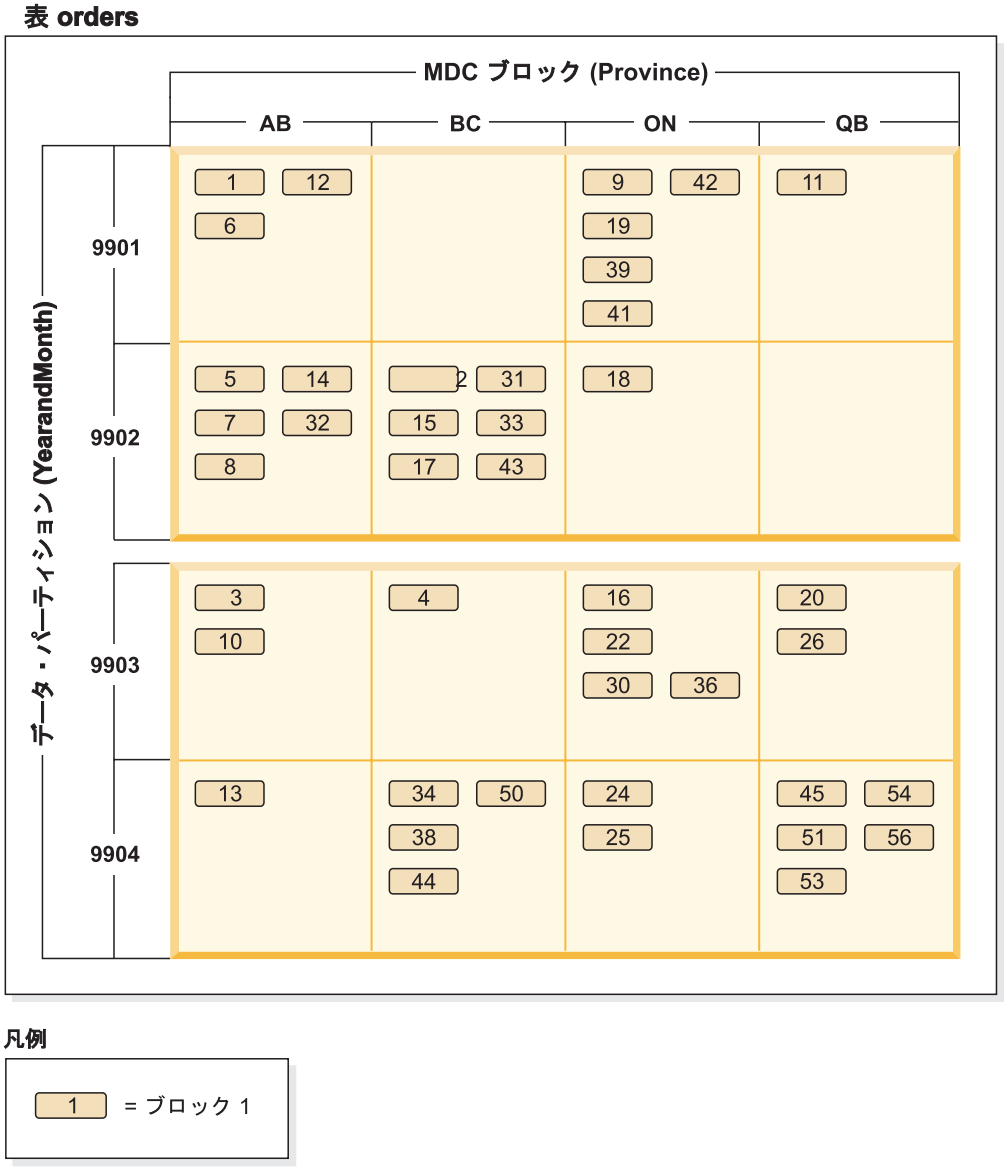


図 27. YearAndMonth によってパーティション化され、Province および YearAndMonth によって編成される表

各範囲に単一値のみがあるようなパーティション化の場合、MDC キーに表パーティション列を含めても、何も得られません。

考慮事項

- 基本表と比較して、MDC 表およびパーティション表は多くのストレージを必要とします。これらのストレージ必要量は付加的なものですが、利点を考えると妥当なものと考えられます。
- パーティション・データベース環境で表パーティション化と MDC 機能を組み合わせないことを選択するなら、確信をもってデータ配分を予測できるような場合 (一般的にここで説明されているシステムのタイプの場合) には、表パーティション化が最善です。そうでない場合には、MDC を考慮する必要があります。

第 4 章 並列データベース・システム

並列処理

データベース照会などの作業のコンポーネントは、並列に実行することにより、パフォーマンスを大幅に強化できます。作業の性質、データベース構成、およびハードウェア環境すべてによって、DB2 データベース製品が作業を並列に実行する方法は異なります。

これらの考慮事項は相互に関連しているため、データベースの物理的および論理的な設計の作業をする際に、これらを一緒に検討する必要があります。 DB2 データベース・システムでは、以下のタイプの並列処理がサポートされています。

- I/O
- 照会
- ユーティリティー

入出力の並列処理

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

照会並列処理

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

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

照会内並列処理 とは、*パーティション内並列処理* またはパーティション間並列処理 (あるいはその両方) を使用して、単一の照会の一部分を同時に処理することです。

パーティション内並列処理

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

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

86 ページの図 28 は、3 つのピース (部分) に分割して並列に実行できるようにする照会を示したものであり、照会が順次に実行された場合に比べてより速く結果が

戻されます。これらのピースは、お互いのコピーです。パーティション内並列処理を利用するためには、データベースを適切に構成する必要があります。並列処理の度合いは自分で選択するか、またはシステムに選択させるようにすることができます。並列処理の度合いは、並列に実行する照会のピースの数を表しています。

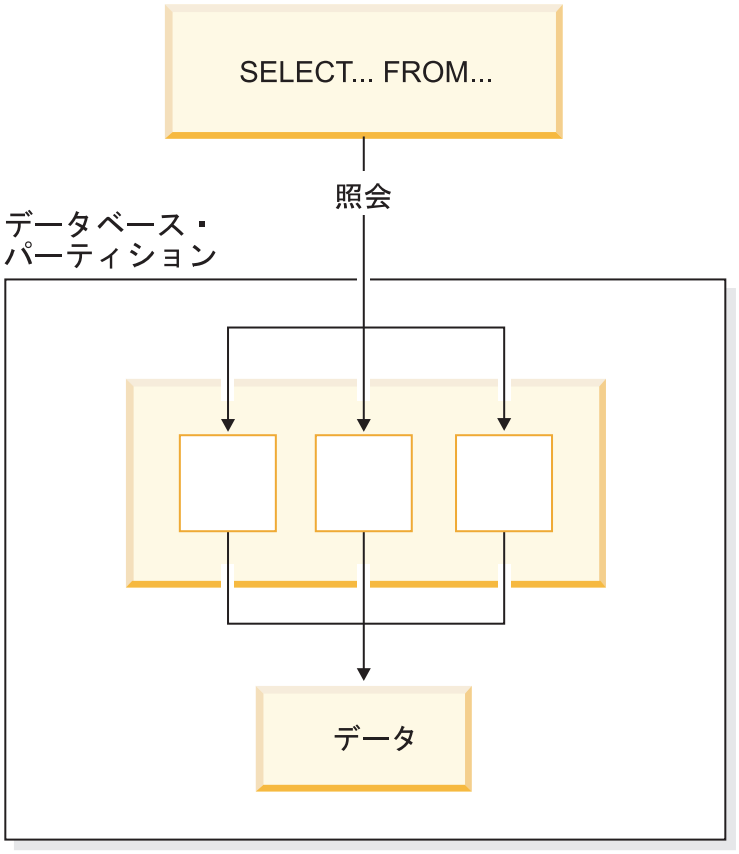


図 28. パーティション内並列処理

パーティション間並列処理

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

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

87 ページの図 29 は、3 つのピースに分割して並列に実行できるようにする照会を示したものであり、照会が単一のデータベース・パーティション内で順次に実行された場合に比べてより速く結果が戻されます。

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

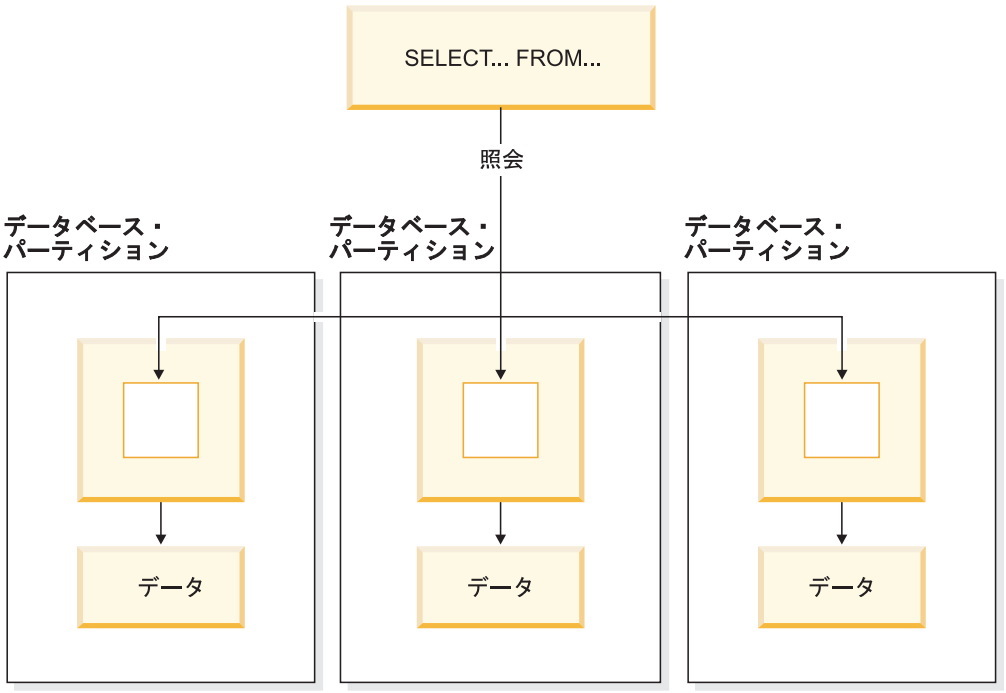


図 29. パーティション間並列処理

パーティション内並列処理とパーティション間並列処理の同時使用

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

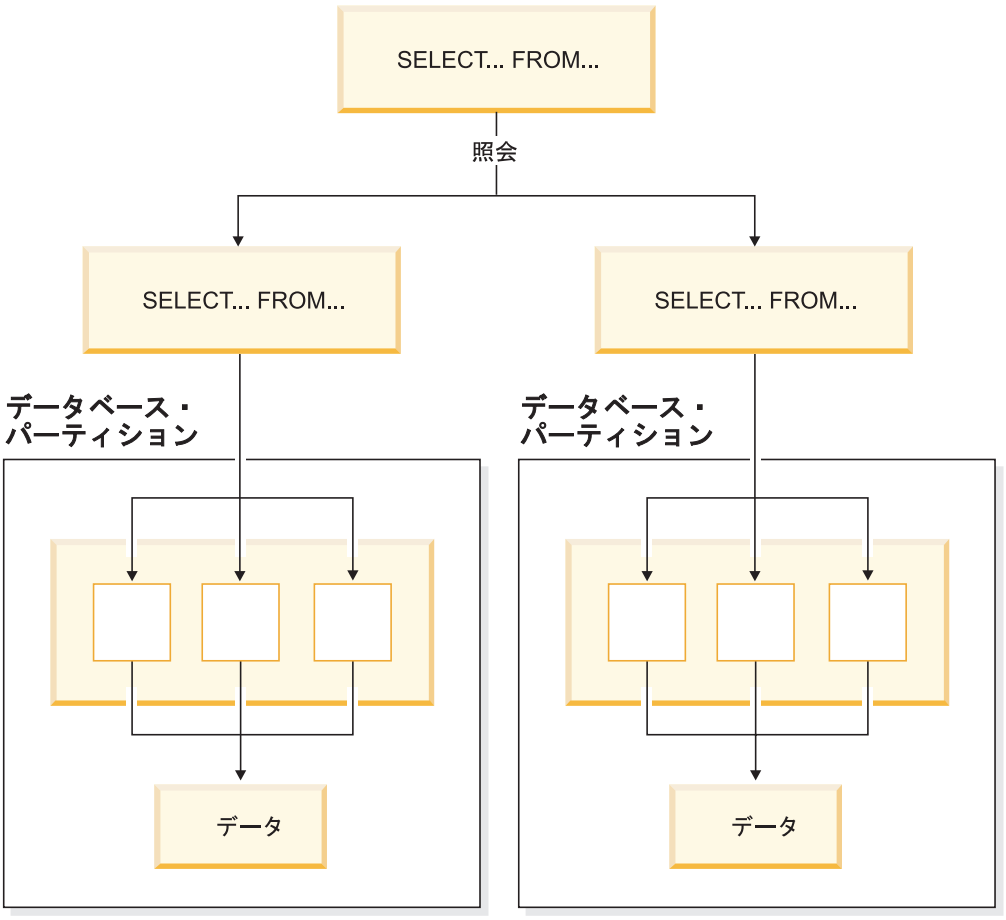


図 30. パーティション間並列処理とパーティション内並列処理の同時使用

ユーティリティー並列処理

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

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

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

索引の作成時には、データのスキャンとその後のソートが並列して実行されます。DB2 システムでは、索引を作成するときに、入出力並列処理とパーティション内並列処理の両方を利用しています。これは、再始動時 (索引が無効としてマーク付けされている場合) およびデータの再編成時に、 CREATE INDEX ステートメントが出されたときの索引作成のスピードアップに役立ちます。

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

パーティション・データベース環境

データベース・パーティション機能 (DPF) は、データベース・マネージャーの機能を、並列、マルチノードの環境に拡張します。

- *データベース・パーティション* は、データベースの一部であり、それ自体のデータ、索引、構成ファイル、およびトランザクション・ログからなります。データベース・パーティションは、ノードまたはデータベース・ノードと呼ばれる場合があります。パーティション・データベース環境とは、データベース・パーティション全体へのデータの配分をサポートするデータベースのインストール済み環境です。
- *単一パーティション・データベース* は、 1 つだけのデータベース・パーティションを持つデータベースです。データベース内のすべてのデータが、その 1 つのデータベース・パーティションに保管されます。この場合、データベース・パーティション・グループがあっても、追加の機能は提供されません。
- *複数パーティション・データベース* は、 2 つ以上のデータベース・パーティションを持つデータベースです。表は、1 つ以上のデータベース・パーティションに配置することができます。表が複数のデータベース・パーティションからなるデータベース・パーティション・グループ内にある場合、その行の一部が 1 つのデータベース・パーティションに保管され、その他の行は他のデータベース・パーティションに保管されます。

通常、物理マシンごとに 1 つのデータベース・パーティションが存在し、各システムのプロセッサーが各データベース・パーティションのデータベース・マネージャーによって使用されて、データベース内の全データのうちの一部を管理します。

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

ユーザーとの対話は、そのユーザー用の*コーディネーター・パーティション* である、1 つのデータベース・パーティションを介して行われます。コーディネーター・パーティションは、アプリケーションと同じデータベース・パーティションで実行されるか、またはリモート・アプリケーションの場合、そのアプリケーションが接続されるデータベース・パーティションで実行されます。任意のデータベース・パーティションをコーディネーター・パーティションとして使用することができます。

データベース・マネージャーは、データベース内の複数のデータベース・パーティションにわたってデータを保管できるようにします。つまり、データが物理的には複数のデータベース・パーティションにわたって保管されていても、 1 つの同じ場

所に置かれているかのようにアクセスできます。複数パーティション・データベースのデータにアクセスするアプリケーションやユーザーは、データが物理的にどこにあるかを認識する必要がありません。

データは、物理的には分割されていますが、1 つの論理的な統一体として使用および管理されます。ユーザーは、分散キーを宣言することによって、自分のデータを分散する方法を選択することができます。ユーザーはまた、データを保管する表スペースと関連するデータベース・パーティション・グループを選択することによって、自分のデータが配分されるデータベース・パーティションを指定したり、データベース・パーティションの数を決定することもできます。 DB2 設計アドバイザーを使用することにより、配分とレプリケーションに関する提案を行うことができます。さらに、更新可能な分散マップをハッシュ・アルゴリズムとともに使用して、データベース・パーティションへの分散キー値のマッピングを指定します (これによってデータの各行の配置と検索が決まります)。その結果、大きな表のためのワークロードを複数パーティション・データベース全体に配分させると同時に、小さい表を 1 つまたは複数のデータベース・パーティションに保管することができます。それぞれのデータベース・パーティションは保管するデータのローカル索引を持っており、その結果、ローカルのデータ・アクセスのパフォーマンスが向上します。

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

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

DB2 データベース製品の非 root インストールは、データベース・パーティションがサポートされません。その結果として、ノード追加操作を実行できません。`db2nodes.cfg` ファイルは手動で更新しないでください。手動更新を実行すると、SQL6031N エラーが表示されます。

データベース・パーティションおよびプロセッサー環境

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

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。

- シングル・プロセッサー (ユニプロセッサー) 上での単一データベース・パーティション
- 複数プロセッサーを備えた単一データベース・パーティション (SMP)
- 複数データベース・パーティション構成
  - 1 つのプロセッサーを備えたデータベース・パーティション (MPP)

– 複数のプロセッサーを備えたデータベース・パーティション (SMP のクラスター)
– 論理データベース・パーティション

容量および拡張容易性は、それぞれの環境ごとに説明します。

シングル・プロセッサー上での単一データベース・パーティション

この環境は、メモリーとディスクからなりますが、単一の CPU しか含まれていません (図 31 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、シングル・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 (シングル・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。

ユニプロセッサー環境

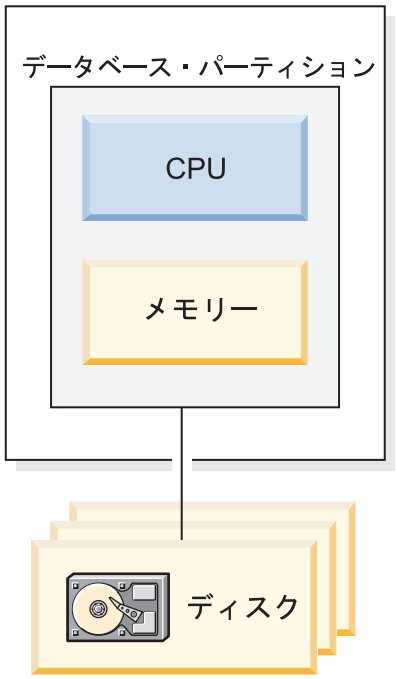


図 31. シングル・プロセッサー上での単一データベース・パーティション

容量および拡張容易性

この環境では、さらにディスクを追加することができます。各ディスクが 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。

シングル・プロセッサー・システムは、プロセッサーが処理できるディスク・スペースの量によって制限されます。ワークロードが増加するにつれ、コンポーネント (例えばメモリーやディスク) を追加するかどうかに関係なく、単一の CPU ではユ

ーザー要求をそれ以上速く処理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データベース・パーティション・システムに移行することを検討します。

複数プロセッサーを備えた単一データベース・パーティション

この環境は通常、同じマシン内の複数の等価の処理能力を持つプロセッサーからなり (図 32 を参照)、*対称型マルチプロセッサー (SMP)* システムと呼ばれます。ディスク・スペースおよびメモリーなどのリソースは、*共有* されます。

複数のプロセッサーが使用可能なので、異なるデータベースの操作をより速く完了させることができます。また DB2 データベース・システムでは、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、例えばデータのロード、表スペースのバックアップおよびリストア、および既存のデータの索引の作成などにも、複数のプロセッサーを利用できます。

対照型マルチプロセッサー (SMP) 環境

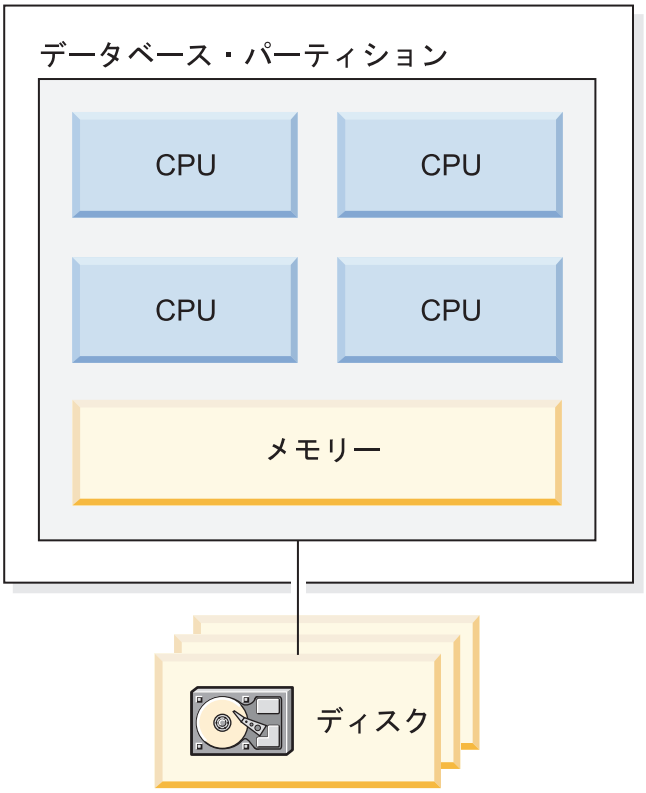


図 32. 単一パーティション・データベース対称型マルチプロセッサー環境

容量および拡張容易性

この環境では、さらにプロセッサーを追加することができます。ただし、異なるプロセッサーが同じデータのアクセスを試みる可能性があるため、業務の操作が拡大

するにつれて、この環境での制約が発生してくる可能性があります。共有メモリーと共有ディスクを使用すれば、すべてのデータベース・データを効率的に共有することができます。

ディスクの数を増やすことにより、プロセッサーに関連するデータベース・パーティションの入出力容量を増やすことができます。特に入出力要求を処理するために、入出力サーバーを設定することができます。各ディスクが 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、複数データベース・パーティションを備えたシステムに移行することを検討します。

複数データベース・パーティション構成

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

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

1 つのプロセッサーを備えたデータベース・パーティション

この環境には多くのデータベース・パーティションがあります。それぞれのデータベース・パーティションは独自のマシン上に常駐しており、独自のプロセッサー、メモリー、およびディスクを持っています (94 ページの図 33)。各マシンはそれぞれ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、ユニプロセッサー・クラスター、超並列処理 (MPP) 環境、およびシェアード・ナッシング (shared-nothing) 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正確に反映したものです。 SMP 環境と異なり、MPP 環境ではメモリーまたはディスクが共有されません。そのため MPP 環境では、メモリーおよびディスクの共有による制約は存在しません。

パーティション・データベース環境では、物理的には 1 つのデータベースを複数のデータベース・パーティションに分割できますが、論理的にはそれを 1 つのデータベースとして扱えます。データの配分は、ほとんどのユーザーに知られずに行われます。作業は、データベース・マネージャー間で分割できます。各データベース・パーティションのそれぞれのデータベース・マネージャーは、自分が担当する部分のデータベースに対して作業を行います。

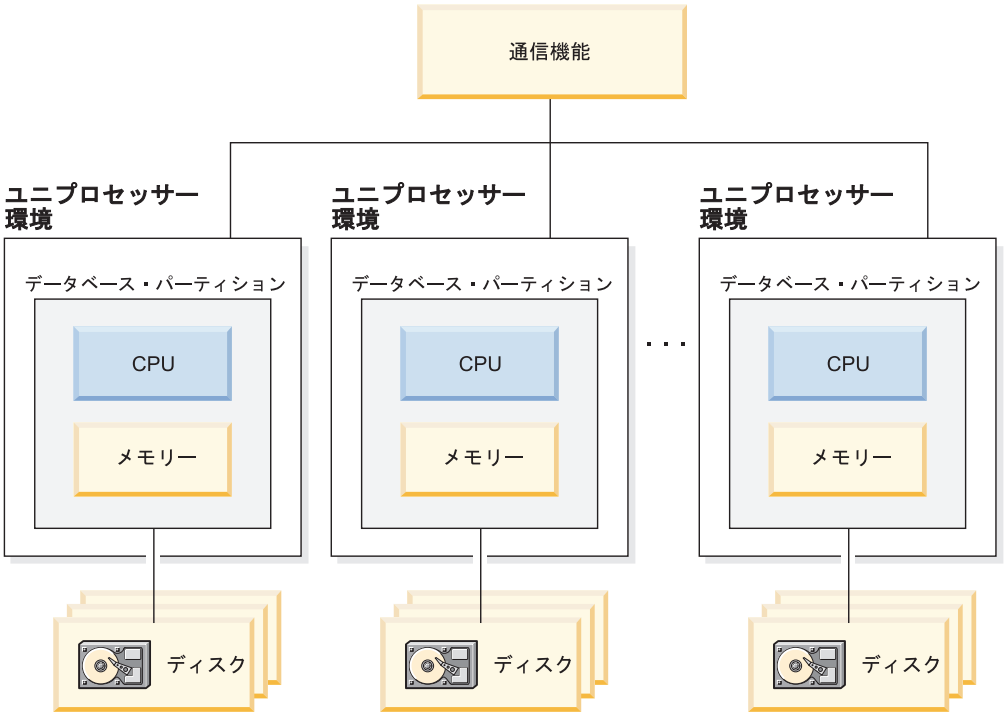


図 33. 超並列処理 (MPP) 環境

容量および拡張容易性

この環境では、データベース・パーティション (ノード) を構成に追加することができます。一部のプラットフォーム (例えば RS/6000® SP) では、最大数は 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存在する場合があります。

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、各データベース・パーティションが複数のプロセッサーを備えたシステムに移行することを検討します。

複数プロセッサーを備えたデータベース・パーティション

各データベース・パーティションがシングル・プロセッサーを持つ構成に代わる構成として、各データベース・パーティションが複数のプロセッサーを持つ構成があります。これは、*SMP クラスター* と呼ばれます (95 ページの図 34)。

この構成は、SMP 並列処理と MPP 並列処理の利点を組み合わせたものです。これは、1 つの照会を複数プロセッサーにわたって単一データベース・パーティションで実行できることを意味します。また、1 つの照会を複数データベース・パーティションにわたって並列に実行できることも意味します。

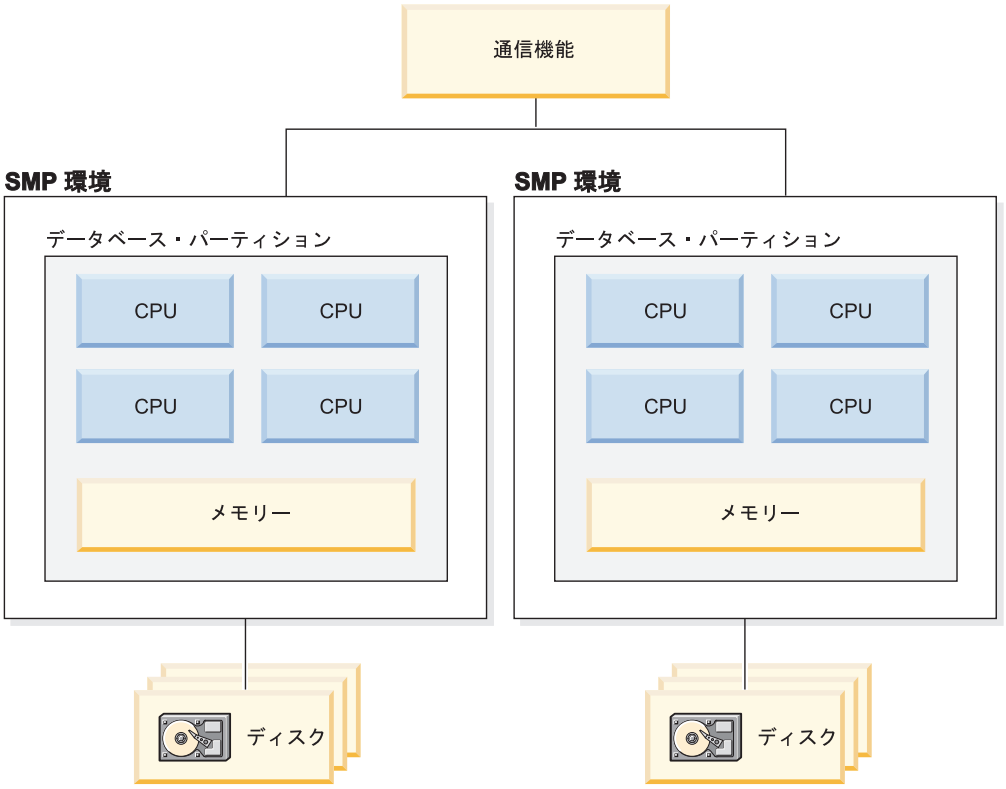


図 34. 複数の対称型マルチプロセッサー (SMP) 環境からなるクラスター

容量および拡張容易性

この環境では、データベース・パーティションを追加したり、既存のデータベース・パーティションにプロセッサーを追加したりすることができます。

論理データベース・パーティション

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

論理データベース・パーティションには拡張容易性が備えられています。複数の論理パーティションで実行され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・マネージャーよりも有効に使用可能リソースを利用できます。 96 ページの図 35 は、SMP マシン上でデータベース・パーティションを追加することによって拡張容易性を向上させることができることを示しています。このことは、プロセッサーを多数持つマシンに特に当てはまります。データベースを配分することによって、それぞれのデータベース・パーティションを個別に管理およびリカバリーすることができます。

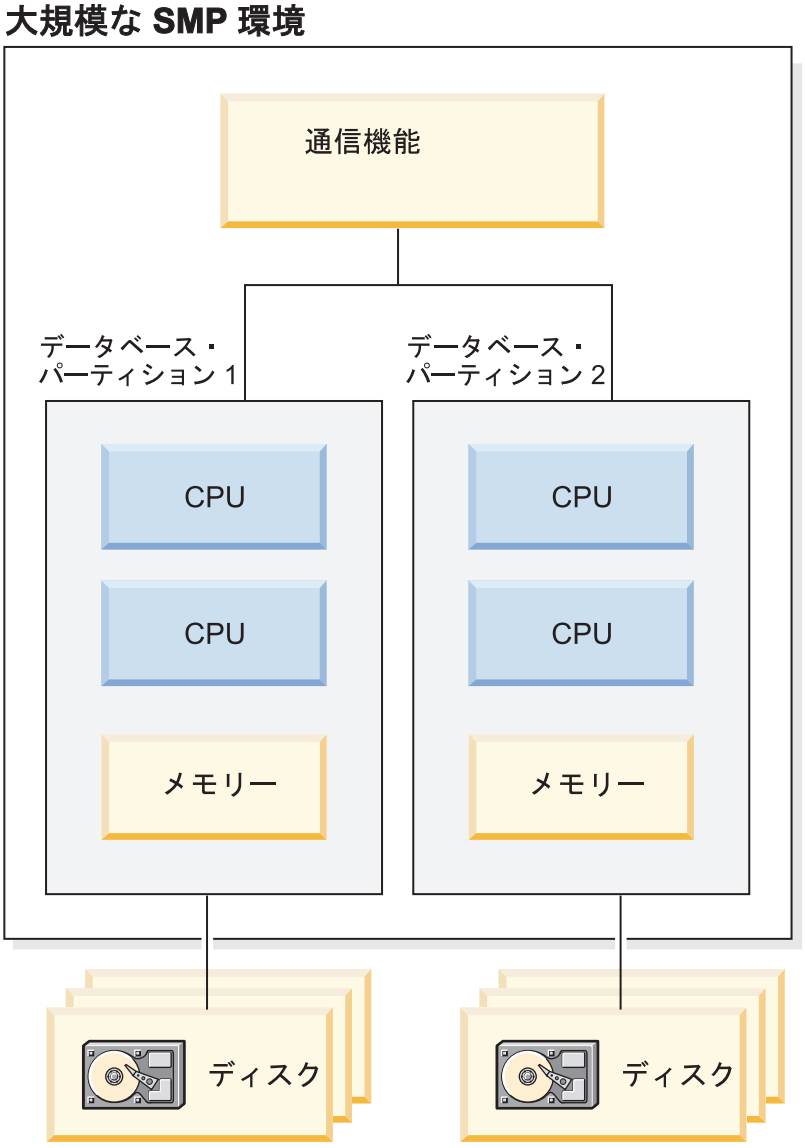


図 35. 対称型マルチプロセッサー環境でのパーティション・データベース

97 ページの図 36 は、処理能力を高めるために、図 35 に示された構成を増やせることを示しています。

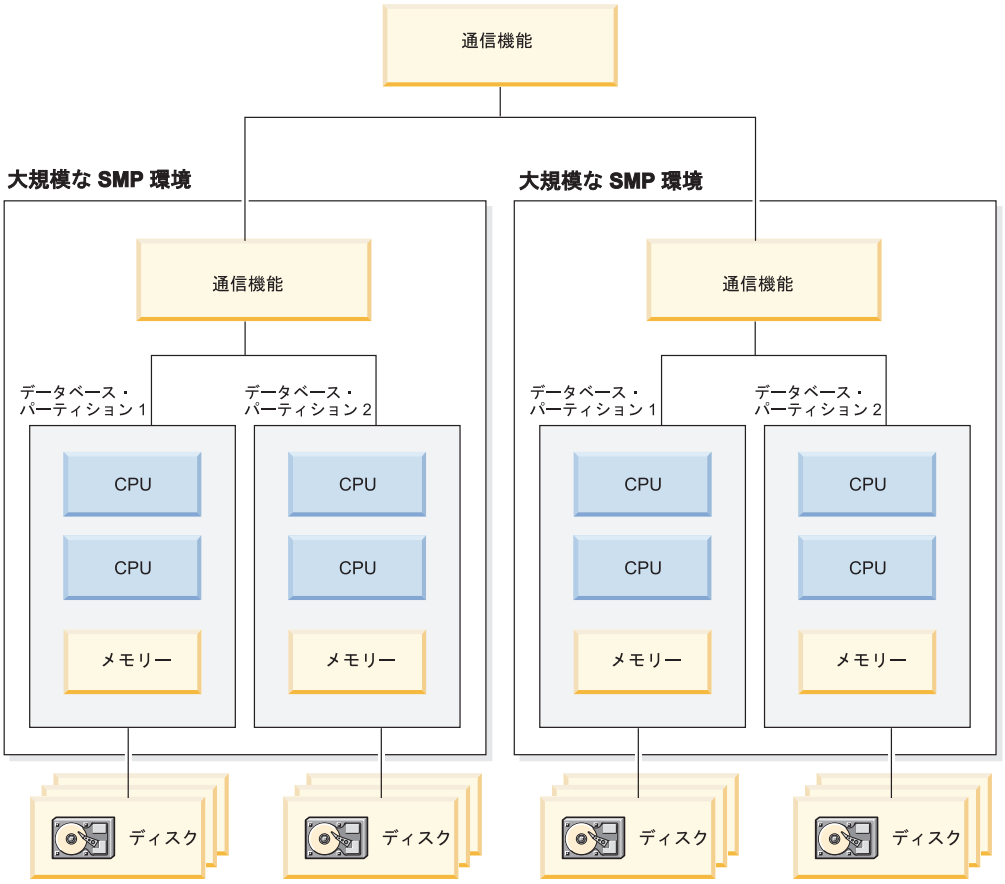


図 36. クラスターを構成する複数の対称型マルチプロセッサー環境を含むパーティション・データベース

注: (プロセッサーの数に関わりなく) 複数のデータベース・パーティションを同じマシン上に共存させることができるため、高可用性構成とフェイルオーバーの計画の設計をより柔軟に行うことができます。マシン故障の際に、同じデータベースの別のデータベース・パーティションがすでに含まれている 2 番目のマシンに自動的にデータベース・パーティションを移動して再始動できます。

各ハードウェア環境に最も適した並列処理のサマリー

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

表 8. それぞれのハードウェア環境で考えられる並列処理のタイプ

ハードウェア環境	入出力並列処理	照会内並列処理	
		パーティション内並列処理	パーティション間並列処理
単一データベース・パーティション、シングル・プロセッサー	あり	不適切 注 (1)	なし
単一データベース・パーティション、複数プロセッサー (SMP)	あり	あり	なし

表 8. それぞれのハードウェア環境で考えられる並列処理のタイプ (続き)

ハードウェア環境	入出力並列処理	照会内並列処理	
		パーティション内並列処理	パーティション間並列処理
複数データベース・パーティション、シングル・プロセッサー (MPP)	あり	不適切 注 (1)	あり
複数データベース・パーティション、複数プロセッサー (SMP のクラスター)	あり	あり	あり
論理データベース・パーティション	あり	あり	あり
注: (1) (構成パラメーターの 1 つを使用する) 並列処理の度合いを 1 より大きい値に設定することにより好ましい結果が得られる場合があります。これはシングル・プロセッサーのシステムでも効果がありますし、実行する照会が (入出力の制約などにより) CPU を十分利用していない場合には特に有効です。			

第 2 部 インストールの注意点

第 5 章 インストールの前提条件

DB2 サーバーのインストール (Windows)

このタスクでは、Windows 上で DB2 セットアップ・ウィザードを開始する方法を説明します。 DB2 セットアップ・ウィザードを使用して、インストールを定義し、DB2 データベース製品をご使用のシステムにインストールします。

始める前に

DB2 セットアップ・ウィザードを開始する前に、以下の事柄を行います。

- パーティション・データベース環境のセットアップを予定している場合は、『パーティション・データベース環境のセットアップ』を参照してください。
- ご使用のシステムがインストール、メモリー、およびディスクの各要件に合うことを確認します。
- Windows 上で LDAP を使用して、DB2 サーバーを Active Directory に登録する予定であれば、インストールの前にディレクトリー・スキーマを拡張する必要があります。そうでない場合は、手動でノードを登録してからデータベースをカタログする必要があります。詳しくは、『LDAP ディレクトリー・サービス用の Active Directory スキーマの拡張 (Windows)』のトピックを参照してください。
- インストールを実行するために推奨されるユーザー権限を持つ、ローカル管理者ユーザー・アカウントを持っている必要があります。 LocalSystem を DAS および DB2 インスタンス・ユーザーとして使用できる、データベース・パーティション・フィーチャーを使用していない DB2 データベース・サーバーでは、システム特権を持つ非管理者ユーザーがインストールを実行できます。

  注: 非管理者ユーザー・アカウントが製品のインストールを実行する場合、DB2 データベース製品のインストールを試行する前に VS2005 ランタイム・ライブラリーがインストールされている必要があります。 DB2 データベース製品をインストールする前にオペレーティング・システムには VS2005 ランタイム・ライブラリーが必要です。 VS2005 ランタイム・ライブラリーは、Microsoft® ランタイム・ライブラリー・ダウンロードの Web サイトから入手できます。次の 2 つの選択が存在します。vcredist_x86.exe (32 ビット・システム用) または vcredist_x64.exe (64 ビット・システム用)
- 必須ではありませんが、リブートなしでインストール・プログラムがコンピューター上の任意のファイルを更新できるようにするために、すべてのプログラムを閉じることをお勧めします。

制約事項

- DB2 コピー名とインスタンス名は、数値で始めることはできません。
- DB2 コピー名とインスタンス名は、すべての DB2 コピーの間で固有でなければなりません。
- XML フィーチャーは、データベース・パーティションが 1 個のみであるデータベースでのみ使用できます。

- 以下のいずれかが既にインストールされている場合は、同じパスに他の DB2 データベース製品をインストールすることはできません。
  - IBM® Data Server Runtime Client
  - IBM Data Server Driver Package
  - *DB2 インフォメーション・センター*
- DB2 セットアップ・ウィザード・フィールドでは英語以外の文字を受け入れません。
- Windows Vista で拡張セキュリティーを有効にする場合、ローカル DB2 コマンドとアプリケーションを実行するために、ユーザーは DB2ADMNS または DB2USERS グループに属している必要があります。これは、ローカル管理者にデフォルトで付与されている特権を制限する特別なセキュリティー・フィーチャー (ユーザー・アクセス制御) のためです。ユーザーがこれらのグループの 1 つに属していない場合、ローカル DB2 構成またはアプリケーション・データに対する読み取りアクセス権限が与えられません。

手順

次のようにして、DB2 セットアップ・ウィザードを開始します。

1. DB2 インストール用に定義したローカル管理者アカウントで、システムにログオンします。
2. DB2 データベース製品 DVD を所有している場合は、これをドライブに挿入します。自動実行フィーチャーを有効にしている場合、DB2 セットアップ・ランチパッドが自動的に開始されます。自動実行機能が作動しない場合は、Windows エクスプローラを使用し、DB2 データベース製品 DVD をブラウズして setup アイコンをダブルクリックし、DB2 セットアップ・ランチパッドを開始します。
3. DB2 データベース製品をパスポート・アドバンテージからダウンロードした場合は、実行可能ファイルを実行して DB2 データベース製品インストール・ファイルを解凍します。 Windows エクスプローラを使用し、DB2 インストール・ファイルをブラウズして setup アイコンをダブルクリックし、DB2 セットアップ・ランチパッドを開始します。
4. DB2 セットアップ・ランチパッドから、インストールの前提条件およびリリース情報を表示することができます。あるいは、インストールに直接進むこともできます。後で追加されたインストール前提条件およびリリース情報を参照することもできます。
5. 「**製品のインストール**」をクリックすると、「製品のインストール」ウィンドウに、インストールに使用できる製品が表示されます。

   既存の DB2 データベース製品がコンピューターにインストールされていない場合は、「**新規インストール**」をクリックして、インストールを起動します。DB2 セットアップ・ウィザードのプロンプトに従ってインストールを進めます。

   既存の DB2 データベース製品が 1 つ以上コンピューターにインストールされている場合は、次のようにできます。
   - 新しい DB2 コピーを作成するには、「**新規インストール**」をクリックします。

- 既存の DB2 コピーのアップグレード、既存の DB2 コピーへの機能追加、既存の DB2 バージョン 8 またはバージョン 9.1 のコピーのマイグレーション、またはアドオン製品のインストールを実行するには、「**既存の処理**」をクリックします。

6. DB2 セットアップ・ウィザードは、システム言語を判別してから、その言語用のセットアップ・プログラムを立ち上げます。残りのステップについて説明しているオンライン・ヘルプを利用できます。オンライン・ヘルプを呼び出すには、「**ヘルプ**」をクリックするか、または **F1** を押します。「**キャンセル**」をクリックすれば、いつでもインストールを終了できます。

結果

DB2 データベース製品がインストールされるデフォルトの場所は *Program_Files*\IBM\sqllib ディレクトリーで、 *Program_Files* は Program Files ディレクトリーの場所を表します。

インストール先のシステムでこのディレクトリーが既に使用中の場合、DB2 データベース製品のインストール・パスに *_xx* が追加されます。 *xx* は 01 で始まる数字で、インストール済みの DB2 コピーの数に応じて増加します。

独自の DB2 データベース製品のインストール・パスを指定することもできます。

次の作業

- します。インストールを検証します。
- します。インストール後の必要な作業を実行します。

インストール時に検出されるエラーの詳細については、My Documents¥DB2LOG¥ ディレクトリーにあるインストール・ログ・ファイルを確認してください。ログ・ファイルは DB2-*ProductAbrrev-DateTime*.log という形式になります (例えば DB2-ESE-Tue Apr 04 17_04_45 2006.log)。

ローカル・コンピューターか、ネットワーク上の別のコンピューターにある DB2 資料に DB2 データベース製品からアクセスできるようにする場合は、*DB2 インフォメーション・センター* をインストールする必要があります。 *DB2 インフォメーション・センター* には、DB2 データベース・システムと DB2 関連製品の資料が収録されています。デフォルトでは、*DB2 インフォメーション・センター* がローカルにインストール済みでなければ、Web を介して DB2 情報にアクセスできます。

DB2 Express™ Edition および DB2 Workgroup Server Edition のメモリー限度

DB2 Express Edition をインストールしている場合、このインスタンスで許可される最大メモリーは 4 GB です。

DB2 Workgroup Server Edition をインストールしている場合、このインスタンスで許可される最大メモリーは 16 GB です。

インスタンスに割り振られるメモリー量は、**INSTANCE_MEMORY** データベース・マネージャー構成パラメーターによって決まります。

バージョン 9.1 からマイグレーションする際の重要な注意事項:

- バージョン 9.1 DB2 データベース製品のメモリー構成が許容限度を超過すると、DB2 データベース製品は現行バージョンへのマイグレーション後に開始しない可能性があります。

- セルフチューニング・メモリー・マネージャーを使用する場合、ライセンス限度を超えてインスタンス全体のメモリー限度が増やされることはありません。

パーティション DB2 サーバーの環境の準備 (Windows)

このトピックでは、DB2 製品のパーティション・インストールのための Windows 環境を準備するために必要なステップを説明します。

それぞれの関与するコンピューターには、同じオペレーティング・システムが必要です。

以下のようにして、インストールのために Windows 環境を準備します。

1. 基本コンピューターおよび関与するコンピューターが同じ Windows ドメインに属していることを確認します。「コントロール パネル」からアクセスできる「システム プロパティ」ダイアログを使用して、コンピューターが属するドメインを調べることができます。
2. 基本コンピューターと関与するコンピューターの時刻と日付の設定が整合していることを確認してください。整合していると見なすためには、すべてのコンピューターの GMT (グリニッジ標準時) 時刻の差が 1 時間以内でなければなりません。

   システム日付と時刻は、「コントロール パネル」からアクセスできる「日付と時刻」ダイアログを使用して変更することができます。 `max_time_diff` 構成パラメーターを使えば、この制限を変更することが可能です。このデフォルトは `max_time_diff = 60` になっており、この場合に許容される差は 60 分未満です。
3. パーティション・データベース環境に加わっている各コンピューター・オブジェクトに、「Trust computer for delegation」(コンピューターを委任に対して信頼する) 特権のフラグが立っていることを確認してください。「Active Directory ユーザーとコンピュータ」コンソールの各コンピューターのアカウントの「プロパティ (Properties)」ダイアログ・ボックスの「全般 (General)」タブにある「コンピューターを委任に対して信頼する (Trust computer for delegation)」チェック・ボックスがチェックされていることを確認します。
4. すべての関与するコンピューターが TCP/IP を使用して相互に通信できることを確認します。
   a. 1 つの関与するコンピューター上で hostname コマンドを入力します。このコマンドはそのコンピューターのホスト名を戻します。
   b. 別の関与するコンピューターで、以下のコマンドを入力します。

      ```
      ping hostname
      ```

      hostname は、基本コンピューターのホスト名を表します。テストが成功した場合は、以下のような出力を受け取ります。

      ```
      Pinging ServerA.ibm.com [9.21.27.230] with 32 bytes of data:

      Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
      Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
      Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
      ```

すべての関与するコンピューターが TCP/IP を介して相互に通信できることを確認できるまで、以上のステップを繰り返します。それぞれのコンピューターに静的 IP アドレスがなければなりません。

複数のネットワーク・アダプターを使用する予定であれば、データベース・パーティション・サーバーの相互通信に使用するアダプターを指定することができます。インストール完了後に、db2nchg コマンドを使用して、db2nodes.cfg ファイルの netname フィールドを指定します。

5. インストール中に、DB2 Administration Server ユーザー・アカウントを入力するよう指示されます。これは、DB2 Administration Server (DAS) で使用されるローカルまたはドメインのユーザー・アカウントです。 DAS は、GUI ツールをサポートするために使用される管理サービスで、管理タスクを援助します。ここでユーザーを定義することもできますし、DB2 セットアップ・ウィザードに作成させることもできます。 DB2 セットアップ・ウィザードに新規ドメイン・ユーザーを作成させたい場合には、インストールを実行するために使用するアカウントが、ドメイン・ユーザーを作成する権限を持っている必要があります。
6. 基本コンピューターで、インスタンス所有のデータベース・パーティション・サーバーをインストールする場合には、ローカル*管理者* グループに属するドメイン・ユーザー・アカウントが必要です。 DB2 のインストール時には、このユーザーとしてログオンします。同じユーザー・アカウントは、それぞれの関与するコンピューター上のローカル*管理者* グループにも追加する必要があります。このユーザーには、「*オペレーティング システムの一部として機能する*」 というユーザー権限も設定する必要があります。
7. インスタンス中のすべてのコンピューターで、データベース・ディレクトリーがあるローカル・ドライブ名が同じであることを確認します。GET DATABASE CONFIGURATION コマンドを実行して、DFTDBPATH DBM 構成パラメーターの値を検査することにより、この状態を確認できます。
8. インストール中に、DB2 インスタンスに関連付けられたドメイン・ユーザー・アカウントを入力するよう指示されます。どの DB2 インスタンスにも、1 つのユーザーが割り振られます。インスタンスの開始時に、DB2 はこのユーザー名でログオンします。ここでユーザーを定義することもできますし、DB2 セットアップ・ウィザードに新規ドメイン・ユーザーを作成させることもできます。

   新しいノードをパーティション環境に追加する場合、DB2 コピー名はすべてのコンピューターの間で同じでなければなりません。

   DB2 セットアップ・ウィザードに新規ドメイン・ユーザーを作成させたい場合には、インストールを実行するために使用するアカウントが、ドメイン・ユーザーを作成する権限を持っている必要があります。インスタンス・ユーザー・ドメイン・アカウントは、すべての関与するコンピューター上でローカル*管理者* グループに属している必要があり、以下のユーザー権限を付与されることになります。

   - オペレーティング・システムの一部として機能
   - トークン・オブジェクトの作成
   - メモリー内のページのロック
   - サービスとしてログオン
   - クォータの増加

- プロセス・レベル・トークンの置き換え

拡張セキュリティーを選択した場合は、アカウントは DB2ADMNS グループのメンバーでもなければなりません。DB2ADMNS グループには既にこれらの特権があるので、アカウントに特権を明示的に追加する必要はありません。

高速コミュニケーション・マネージャー (Windows)

高速コミュニケーション・マネージャー (FCM) は、同じインスタンスに属する DB2 サーバー製品の通信サポートを提供します。それぞれのデータベース・パーティション・サーバーには、データベース・パーティション・サーバー間の通信機能を提供する 1 つの FCM 送信側デーモンと 1 つの FCM 受信側デーモンがあり、これにより、エージェント要求を処理して、メッセージ・バッファーをやり取りします。インスタンスを開始すると、FCM デーモンが開始されます。

データベース・パーティション・サーバーの相互通信で障害が発生した場合や、または通信が再確立された場合、 FCM スレッドは情報 (データベース・システム・モニターで照会できる情報) を更新し、適切な処置 (影響を受けたトランザクションのロールバックなど) をとらせます。データベース・システム・モニターを使用すると、FCM 構成パラメーターを設定するのに役立ちます。

FCM メッセージ・バッファーの数は、データベース・マネージャー構成パラメーターの*fcm_num_buffers*で指定することができます。FCM チャネルの数は、データベース・マネージャー構成パラメーターの *fcm_num_channels* で指定することができます。データベース・マネージャー構成パラメーターの *fcm_num_buffers* および *fcm_num_channels* は、デフォルト値として AUTOMATIC に設定されますこれらのパラメーターのいずれかが AUTOMATIC に設定されていると、FCM はリソースの使用状況をモニターして、リソースを徐々に解放していきます。これらのパラメーターは、AUTOMATIC に設定したままにしておくことをお勧めします。

DB2 サーバー製品のインストールの概要 (Linux および UNIX)

このトピックでは、AIX®、HP-UX、Linux、および Solaris 上へのDB2 サーバー製品のインストール・ステップを概説します。

DB2 サーバー製品をインストールするには、次のようにします。

1. DB2 製品の前提条件を確認します。
2. 該当する場合は、DB2 のマイグレーション情報を確認してください。
3. HP-UX、Linux、および Solaris でカーネル・パラメーターに変更を加えます。x86_32 上の Linux 以外のすべてのプラットフォームで、インストールに進むには、その前にユーザーは 64 ビット・カーネルをインストールしなければなりません。インストールしないと、インストールは失敗します。
4. インストール・メディアを準備します。

   製品 DVD

   DB2 製品 DVD が自動マウントされない場合は、DB2 製品 DVD をマウントします。

インストール・イメージ
インストール・イメージをダウンロードしたら、そのファイルを untar します。

5. 以下の使用可能な方法の 1 つを使用して、DB2 製品をインストールします。
   - DB2 セットアップ・ウィザード
   - db2_install コマンド
   - 応答ファイルによるサイレント・インストール
   - ペイロード・ファイルのデプロイメント

   DB2 サーバーの場合、DB2 セットアップ・ウィザードを使用して、以下のようなインストールと構成の各タスクを実行することができます。
   - DB2 インストール・タイプ (標準、コンパクト、またはカスタム) の選択。
   - DB2 製品のインストール場所の選択。
   - この製品のインターフェースとメッセージのデフォルト言語として後で指定できる言語のインストール。
   - IBM Tivoli® System Automation for Multiplatforms 基本コンポーネントのインストールまたはアップグレード (Linux および AIX)。
   - DB2 インスタンスのセットアップ。
   - DB2 Administration Server のセットアップ (DAS ユーザーのセットアップを含む)。
   - DB2 テキスト検索サーバーのセットアップ。
   - 管理連絡先およびヘルス・モニター通知のセットアップ。
   - インスタンスのセットアップと構成 (インスタンス・ユーザーのセットアップを含む)。
   - Informix データ・ソース・サポートのセットアップ。
   - DB2 ツール・カタログの準備。
   - DB2 インフォメーション・センター・ポートの指定。
   - 応答ファイルの作成。
6. DB2 セットアップ・ウィザード以外の方法を使用して DB2 サーバーをインストールした場合は、インストール後の構成ステップが必要です。

DB2 のインストール方式

このトピックでは、DB2のインストール方式について説明します。以下の表は、オペレーティング・システムごとに使用できるインストール方式を示しています。

表 9. オペレーティング・システムごとのインストール方式

インストール方式	Windows	Linux または UNIX
DB2 セットアップ・ウィザード	あり	あり
応答ファイル・インストール	あり	あり
db2_install コマンド	なし	あり
ペイロード・ファイルのデプロイメント	なし	あり

DB2 のインストール方式を以下のリストにまとめます。

DB2 セットアップ・ウィザード

DB2 セットアップ・ウィザードは、Linux、UNIX、Windowsの各オペレーティング・システムで使用できる GUI インストーラーです。DB2 セットアップ・ウィザードには、DB2 製品をインストールし、初期のセットアップおよび構成タスクを実行するための使いやすいインターフェースが用意されています。

DB2 セットアップ・ウィザードを使用して、このインストールを他のマシンに複写するのに使用できる DB2 インスタンスや応答ファイルを作成することもできます。

注: Linux および UNIX プラットフォーム上の非ルート・インストールの場合、存在できる DB2 インスタンスは 1 つのみです。DB2 セットアップ・ウィザードは、非ルート・インスタンスを自動的に作成します。

Linux および UNIX プラットフォームでは、DB2 セットアップ・ウィザードを表示するには、X サーバーが必要です。

応答ファイル・インストール

応答ファイルは、セットアップ値と構成値を入れたテキスト・ファイルです。 DB2 セットアップ・プログラムは、そのファイルを読み取り、指定されている値に基づいてインストールを実行します。

応答ファイル・インストールは、サイレント・インストールとも呼ばれます。

応答ファイルの別の利点として、DB2 セットアップ・ウィザードを使用して設定できないパラメーターへのアクセスも提供します。

Linux および UNIX オペレーティング・システムでは、DB2 インストール・イメージをご自分のアプリケーションに組み込んだ場合、アプリケーションは、インストーラーからのインストール進行情報およびプロンプトをコンピューターが読み取り可能な形式で受け取ることができます。この動作は、INTERACTIVE 応答ファイル・キーワードで制御します。

応答ファイルを作成する方法がいくつかあります。

応答ファイル生成プログラムの使用 (Windows プラットフォーム)

Windows では、応答ファイル生成プログラムを使用し、応答ファイルを作成して、既存のインストールを複製することができます。例えば、 IBM データ・サーバー・クライアントをインストールし、そのクライアントの構成を十分に行った後、応答ファイルを生成して、そのクライアントのインストールおよび構成を他のコンピューターに複製することができます。

DB2 セットアップ・ウィザードの使用

DB2 セットアップ・ウィザードの場合は、DB2 セットアップ・ウィザードで項目の選択を進めながら、その選択内容に基づいて応答ファイルを作成できます。つまり、選択内容を応答ファイルに記録し、そのファイルをシステム上の特定の場所に保管できる、ということです。パーティション・データベースのインストールを選択し

た場合は、2 つの応答ファイルが生成されます。1 つはインスタンスを所有するコンピューターのため、もう 1 つは参加するコンピューターのためです。

このインストール方式の利点の 1 つは、インストールを実行せずに応答ファイルを作成できることです。このフィーチャーは、DB2 製品のインストールに必要なオプションを把握するのに役立ちます。後でこの応答ファイルを使用すれば、指定したオプションに従って DB2 製品をインストールできます。

クライアントまたはサーバーの構成内容を保管するためにクライアント・プロファイルまたはサーバー・プロファイルをエクスポートするには、db2cfexp コマンドを使用します。それから db2cfimp コマンドを使用すれば、プロファイルを簡単にインポートできます。db2cfexp コマンドを使用してエクスポートされたクライアント・プロファイルまたはサーバー・プロファイルは、CLIENT_IMPORT_PROFILE キーワードを使用して応答ファイルのインストール時にインポートすることもできます。

データ・ソースのインストールとカタログを実行した後に、クライアントまたはサーバー・プロファイルをエクスポートする必要があります。

各 DB2 製品に用意されているサンプル応答ファイルのカスタマイズ
応答ファイル生成プログラムまたは DB2 セットアップ・ウィザードを使用して応答ファイルを作成する代わりに、サンプル応答ファイルを手動で変更することもできます。サンプル応答ファイルは、DB2 製品 DVD に用意されています。サンプル応答ファイルは、各製品ごとに有効なすべてのキーワードについての詳細情報を提供します。

db2_install コマンド (Linux および UNIX プラットフォームのみ)
db2_install コマンドは、指定した DB2 製品のすべての コンポーネントと英語のインターフェース・サポートをインストールします。-L パラメーターを使用すれば、サポートする追加の言語を選択できます。コンポーネントを選択または選択解除することはできません。

db2_install コマンドは、指定した DB2 製品のすべてのコンポーネントをインストールしますが、ユーザーおよびグループの作成、インスタンスの作成、構成は実行しません。このインストール方式は、インストール後に構成を行う場合に有利です。インストール後ではなくインストール中に DB2 製品を構成する場合は、DB2 セットアップ・ウィザードを使用することを考慮してください。

Linux および UNIX オペレーティング・システムでは、DB2 インストール・イメージをご自分のアプリケーションに組み込んだ場合、アプリケーションは、インストーラーからのインストール進行情報およびプロンプトをコンピューターが読み取り可能な形式で受け取ることができます。

このインストール方式では、製品ファイルのデプロイ後に手動構成が必要になります。

ペイロード・ファイルのデプロイメント (Linux および UNIX のみ)
この方式は、上級のインストール方式であり、ほとんどのユーザーにはお勧

めできません。ペイロード・ファイルをユーザーが物理的にインストールする必要があります。ペイロード・ファイルとは、1 つのインストール可能コンポーネントのすべてのファイルとメタデータを含んだ圧縮 tar ファイルです。

このインストール方式では、製品ファイルのデプロイ後に手動構成が必要になります。

注: DB2 製品のインストール・パッケージは、Linux および UNIX プラットフォーム上のオペレーティング・システム・パッケージではなくなりました。したがって、インストールのためにオペレーティング・システム・コマンドを使用することもできなくなりました。 DB2 のインストール・パッケージと対話して照会を実行するための既存のスクリプトは、変更が必要です。

DB2 セットアップ・ウィザードによる DB2 サーバーのインストール (Linux および UNIX)

このタスクでは、Linux およびUNIX オペレーティング・システムで DB2 セットアップ・ウィザードを開始する方法を説明します。 DB2 セットアップ・ウィザードを使用して、インストール設定を定義し、ご使用のシステムに DB2 データベース製品をインストールします。

始める前に

DB2 セットアップ・ウィザードを開始する前に、以下の事柄を行います。

- パーティション・データベース環境のセットアップを予定している場合は、*DB2 サーバー機能 概説およびインストール* の 『パーティション・データベース環境のセットアップ』 を参照してください。
- ご使用のシステムがインストール、メモリー、およびディスクの各要件に合うことを確認します。
- DB2 データベース・サーバーは、root 権限か non-root (非ルート) 権限のどちらを使用してもインストールできます。非ルート・インストールについて詳しくは、「*DB2 サーバー機能 概説およびインストール*」の『非ルート・インストールの概要 (Linux および UNIX)』を参照してください。
- DB2 データベース製品イメージが使用可能でなければなりません。DB2 インストール・イメージは、物理的な DB2 データベース製品の DVD を購入するか、またはパスポート・アドバンテージからインストール・イメージをダウンロードすることによって入手することができます。
- 英語版以外の DB2 データベース製品をインストールする場合は、該当する National Language Packages が必要になります。
- DB2 セットアップ・ウィザードは、グラフィック・インストーラーです。ご使用のマシンで DB2 セットアップ・ウィザードを実行するには、グラフィカル・ユーザー・インターフェースを表示できる X windows ソフトウェアが必要です。X windows サーバーが実行中であることを確認します。ディスプレイを正しくエクスポートしたことを確認してください。例えば、`export DISPLAY=9.26.163.144:0` のようにします。

- セキュリティー・ソフトウェアを使用している環境の場合、DB2 セットアップ・ウィザードを開始する前に、必要な DB2 ユーザーを手動で作成しなければなりません。

制約事項

- XML フィーチャーは、コード・セット UTF-8 で定義され、データベース・パーティションが 1 個のみであるデータベースでのみ使用できます。
- DB2 セットアップ・ウィザード・フィールドでは英語以外の文字を受け入れません。

手順

次のようにして、DB2 セットアップ・ウィザードを開始します。

1. 物理的な DB2 データベース製品 DVD を入手している場合は、次のコマンドを入力することによって、DB2 データベース製品 DVD がマウントされているディレクトリーに移動します。

   ```
   cd /dvdrom
   ```

   ここで、*/dvdrom* は、DB2 データベース製品 DVD のマウント・ポイントを表しています。
2. DB2 データベース製品イメージをダウンロードした場合は、製品ファイルを解凍して untar しなければなりません。
   a. 以下のようにして、製品ファイルを解凍します。

      ```
      gzip -d product.tar.gz
      ```

      ここで、*product* はダウンロードした製品の名前です。
   b. 以下のようにして、製品ファイルを untar します。

      Linux オペレーティング・システムの場合

      ```
      tar -xvf product.tar
      ```

      AIX、HP-UX、および Solaris オペレーティング・システムの場合

      ```
      gnutar -xvf product.tar
      ```

      ここで、*product* はダウンロードした製品の名前です。
   c. 以下のようにディレクトリーを変更します。

      ```
      cd ./product
      ```

      ここで、*product* はダウンロードした製品の名前です。

   注: National Language Package をダウンロードした場合、同じディレクトリーに untar します。それぞれのサブディレクトリー (例えば、./nlpack/disk1) が同じディレクトリーに作成されるので、インストーラーは、プロンプト画面を表示しなくてもインストール・イメージを自動的に検出できます。
3. データベース製品イメージのあるディレクトリーから ./db2setup コマンドを入力して、DB2 セットアップ・ウィザードを開始します。
4. 「IBM DB2 セットアップ・ランチパッド」 がオープンします。このウィンドウから、インストールの前提条件およびリリース・ノートを表示することができま

す。あるいは、インストールに直接進むこともできます。後で追加されたインストール前提条件およびリリース情報を参照することもできます。

5. 「**製品のインストール**」をクリックすると、「**製品のインストール**」ウィンドウに、インストールに使用できる製品が表示されます。

   「**新規インストール**」をクリックすることにより、インストールを起動します。DB2 セットアップ・ウィザードのプロンプトに従ってインストールを進めます。

   インストールを開始したなら、DB2 セットアップ・ウィザードのインストール・パネルに従って、選択を行ってください。残りのステップについて説明しているインストール操作のヘルプを利用できます。インストール操作のヘルプを呼び出すには、「**ヘルプ (Help)**」をクリックするか、または F1 を押します。「**キャンセル**」をクリックすれば、いつでもインストールを終了できます。

結果

非ルート (non-root) インストールの場合、DB2 データベース製品は必ず $*HOME*/sqllib ディレクトリーにインストールされます。ここで、$*HOME* は非ルート (non-root) ユーザーのホーム・ディレクトリーを表します。

ルート (root) インストールの場合には、DB2 データベース製品はデフォルトでは以下のいずれかのディレクトリーにインストールされます。

AIX、HP-UX、および Solaris
/opt/IBM/db2/V9.5

Linux /opt/ibm/db2/V9.5

インストール先のシステムでこのディレクトリーが既に使用中の場合、DB2 データベース製品のインストール・パスに *_xx* が追加されます。 *_xx* は 01 で始まる数字で、インストール済みの DB2 コピーの数に応じて増加します。

独自の DB2 データベース製品のインストール・パスを指定することもできます。

DB2 インストール・パスには、以下の規則があります。

- 英小文字 (a から z)、英大文字 (A から Z)、および下線文字 (_) を使用できます。
- 128 文字を超えることはできません。
- スペースは使用できません。
- 英語以外の文字は使用できません。

インストール・ログ・ファイルは、以下で構成されています。

- DB2 セットアップ・ログ・ファイル。このファイルは、エラーを含むすべての DB2 インストール情報をキャプチャーします。
  - ルート (root) インストールの場合、DB2 セットアップ・ログ・ファイル名は db2setup.log です。
  - 非ルート (non-root) インストールの場合、DB2 セットアップ・ログ・ファイル名は db2setup_*username*.log となり、*username* はインストールを実行した非ルート (non-root) ユーザー ID です。

- DB2 エラー・ログ・ファイル。このファイルは、Java™ によって戻されるエラー出力 (例外やトラップ情報など) をキャプチャーします。
  - ルート (root) インストールの場合、DB2 エラー・ログ・ファイル名は db2setup.err です。
  - 非ルート (non-root) インストールの場合、DB2 エラー・ログ・ファイル名は db2setup_*username*.err となり、*username* はインストールを実行した非ルート (non-root) ユーザー ID です。

デフォルトでは、/tmp ディレクトリーにこうしたログ・ファイルがあります。これらのログ・ファイルの場所を指定できます。

db2setup.his ファイルはなくなりました。代わりに、DB2 インストーラーは DB2 セットアップ・ログ・ファイルのコピーを DB2_DIR/install/logs/ ディレクトリーに保管し、名前を db2install.history に変更します。この名前が既存の場合は、DB2 インストーラーは名前を db2install.history.*xxxx* (*xxxx* はこのマシンにインストールした数に応じて 0000 から 9999 になる) に変更します。

ヒストリー・ファイルのリストはインストール・コピーごとに異なります。インストール・コピーが除去されると、このインストール・パスの下のヒストリー・ファイルもまた除去されます。このコピー・アクションはインストールの終了直前に行われるので、完了前にプログラムが停止したり異常終了したりすると、ヒストリー・ファイルは作成されません。

次の作業

- します。インストールを検証します。
- します。インストール後の必要な作業を実行します。

また National Language Packages は、DB2 データベース製品のインストール後に、National Language Packages があるディレクトリーから ./db2setup コマンドを実行するとインストールできます。

Linux x86 では、ローカル・コンピューターか、ネットワーク上の別のコンピューターにある DB2 資料に DB2 データベース製品からアクセスできるようにする場合は、*DB2 インフォメーション・センター* をインストールする必要があります。*DB2 インフォメーション・センター* には、DB2 データベース・システムと関連製品に関する資料が含まれています。

DB2 Express Edition および DB2 Workgroup Server Edition のメモリー限度

DB2 Express Edition をインストールしている場合、このインスタンスで許可される最大メモリーは 4 GB です。

DB2 Workgroup Server Edition をインストールしている場合、このインスタンスで許可される最大メモリーは 16 GB です。

インスタンスに割り振られるメモリー量は、**INSTANCE_MEMORY** データベース・マネージャー構成パラメーターによって決まります。

バージョン 9.1 からマイグレーションする際の重要な注意事項:

- バージョン 9.1 DB2 データベース製品のメモリー構成が許容限度を超過すると、DB2 データベース製品は現行バージョンへのマイグレーション後に開始しない可能性があります。

- セルフチューニング・メモリー・マネージャーを使用する場合、ライセンス限度を超えてインスタンス全体のメモリー限度が増やされることはありません。

高速コミュニケーション・マネージャー (Linux および UNIX)

高速コミュニケーション・マネージャー (FCM) は、データベース・パーティション・フィーチャー (DPF) を使用する DB2 サーバー製品の通信サポートを提供します。

複数パーティション・インスタンスの場合、それぞれのデータベース・パーティション・サーバーには、データベース・パーティション・サーバー間の通信機能を提供する 1 つの FCM 送信側デーモンと 1 つの FCM 受信側デーモンがあり、これにより、エージェント要求を処理して、メッセージ・バッファーをやり取りします。複数パーティション・インスタンスを開始すると、FCM デーモンが開始されます。

データベース・パーティション・サーバー間の通信で障害が発生したり、通信が再確立されたりすると、FCM デーモンは情報を更新します。データベース・システム・モニターを使用してこの情報を照会できます。FCM デーモンは必要なアクションも起動します。そのようなアクションの例としては、影響を受けたトランザクションのロールバックがあります。データベース・システム・モニターを使用すると、 FCM 構成パラメーターを設定するのに役立ちます。

FCM メッセージ・バッファーの数は、 データベース・マネージャー構成パラメーターの *fcm_num_buffers* で指定することができます。また、FCM チャネルの数は、データベース・マネージャー構成パラメーターの *fcm_num_channels* で指定することができます。データベース・マネージャー構成パラメーターの *fcm_num_buffers* および *fcm_num_channels* は、デフォルト値として AUTOMATIC に設定されます。これらのパラメーターのいずれかがAUTOMATICに設定されていると、FCM はリソースの使用状況をモニターして、リソースを徐々に解放していきます。これらのパラメーターは、AUTOMATIC に設定したままにしておくことをお勧めします。

第 6 章 インストールする前に

追加のパーティション・データベース環境でのプリインストール作業 (Linux および UNIX)

パーティション DB2 インストールのための環境設定の更新 (AIX)

このタスクでは、パーティション・データベース・システムに参加するそれぞれのコンピューター上で更新する必要のある、環境設定を記述しています。

以下のようにして、AIX 環境設定を更新します。

1. root 権限を持つユーザーとしてコンピューターにログオンします。
2. 以下のコマンドを発行して、AIX の maxuproc (各ユーザーごとの最大プロセス数) 装置属性を *4096* に設定します。

   ```
   chdev -l sys0 -a maxuproc='4096'
   ```

   注: 別のイメージを実行する場合は、bosboot/reboot を 64 ビット・カーネルに切り替える必要が生じることがあります。
3. パーティション・データベース・システムに参加するすべてのワークステーションで、 TCP/IP ネットワーク・パラメーターを以下のような値に設定します。これらの値は、これらのパラメーターの最小値です。ネットワーク関連パラメーターが既にもっと高い値に設定されている場合には、それを変更しないでください。

   ```
   thewall       = 65536
   sb_max        = 1310720
   rfc1323       = 1
   tcp_sendspace = 221184
   tcp_recvspace = 221184
   udp_sendspace = 65536
   udp_recvspace = 65536
   ipqmaxlen     = 250
   somaxconn     = 1024
   ```

   ネットワーク関連のパラメーターの現行設定値をすべてリスト表示するには、以下のコマンドを入力します。

   ```
   no -a | more
   ```

   パラメーターを設定するには、以下のようなコマンドを入力します。

   ```
   no -o parameter_name=value
   ```

   説明:

   - *parameter_name* は、設定するパラメーターを表します。
   - *value* は、このパラメーターに設定する値を表します。

   例えば、 tcp_sendspace パラメーターを 221184 に設定するには、以下のようなコマンドを入力します。

   ```
   no -o tcp_sendspace=221184
   ```

4. 高速相互接続を使う場合は、 *css0* の *spoolsize* と *rpoolsize* を以下のような値に設定する必要があります。

   ```
   spoolsize     16777216
   rpoolsize     16777216
   ```

   これらのパラメーターの現行設定値をリスト表示するには、以下のコマンドを入力します。

   ```
   lsattr -l css0 -E
   ```

   これらのパラメーターを設定するには、以下のコマンドを入力します。

   ```
   /usr/lpp/ssp/css/chgcss -l css0 -a spoolsize=16777216
   /usr/lpp/ssp/css/chgcss -l css0 -a rpoolsize=16777216
   ```

   システムを調整するために /tftpboot/tuning.cst ファイルを使用しない場合、インストール後にサンプル・スクリプト・ファイル DB2DIR/misc/rc.local.sample を使って、ネットワーク関連パラメーターを更新することができます (DB2DIR は DB2 製品のインストール先パス)。インストール後にサンプル・スクリプト・ファイルを使ってネットワーク関連パラメーターを更新するには、以下のステップで行います。

   a. 以下のようなコマンドを入力して、このスクリプト・ファイルを /etc ディレクトリーにコピーし、 root によってそれを実行可能にします。

      ```
      cp /usr/opt/db2_09_01/misc/rc.local.sample /etc/rc.local
      chown root:sys /etc/rc.local
      chmod 744 /etc/rc.local
      ```

   b. /etc/rc.local ファイルを調べて、必要であれば、更新します。

   c. マシンがリブートされるときに必ず /etc/rc.local スクリプトが実行されるように、 /etc/inittab ファイルに項目を追加します。 mkitab コマンドを使用して、 /etc/inittab ファイルに項目を追加することができます。この項目を追加するには、以下のようなコマンドを入力します。

      ```
      mkitab "rclocal:2:wait:/etc/rc.local > /dev/console 2>&1"
      ```

   d. 以下のようなコマンドを入力して、 /etc/rc.nfs 項目に必ず /etc/inittab ファイルが入るようにします。

      ```
      lsitab rcnfs
      ```

   e. 以下のようなコマンドを入力して、マシンをリブートしないでネットワーク・パラメーターを更新します。

      ```
      /etc/rc.local
      ```

5. DB2 ESE のパーティション・インストールを実行するのに十分なページ・スペースがあることを確認してください。十分なページ・スペースがない場合、仮想メモリーを最も多く使用するプロセス (DB2 プロセスのうちの 1 つが可能性が高い) が、オペレーティング・システムによって強制終了されます。使用できるページ・スペースをチェックするには、以下のようなコマンドを入力します。

   ```
   lsps -a
   ```

   このコマンドは、以下のような出力を戻します。

   ```
   Page Space  Physical Volume  Volume Group  Size  %Used  Active  Auto  Type
   paging00    hdisk1          rootvg        60MB     19     yes   yes    lv
   hd6         hdisk0          rootvg        60MB     21     yes   yes    lv
   hd6         hdisk2          rootvg        64MB     21     yes   yes    lv
   ```

使用できるページ・スペースを、コンピューターにインストールされている物理メモリーの 2 倍の容量にしてください。

6. 小さいサイズあるいは中間サイズまでのパーティション・データベース・システムを作成するときは、インスタンス所有者のコンピューター上のネットワーク・ファイル・システム・デーモン (NFSD) の数を、ほぼ以下の値にする必要があります。

   ```
   # of biod on a computer × # of computers in the instance
   (1 台のコンピューター上の biod 数 × インスタンス内のコンピューター数)
   ```

   コンピューターごとに 10 個の biod プロセスを実行することが理想的です。 4 つのコンピューター・システムに 10 個の biod プロセスがある場合、この公式に従えば 40 個の NFSD を使用することになります。

   大型システムをインストールする場合は、コンピューターには最高 120 までの NFSD をもつことができます。

   NFS の追加情報については、NFS の資料を参照してください。

ESE ワークステーションにコマンドを配布する一括作業のセットアップ (AIX)

AIX のパーティション・データベース環境では、パーティション・データベース・システムに参加する RS/6000 SP™ ワークステーションのセットにコマンドを配布するための、一括作業をセットアップすることができます。 dsh コマンドによって、ワークステーションにコマンドを配布することができます。

これは、AIX でパーティション・データベース・システムをインストールまたは管理する場合に役立つことがあります。その環境にあるすべてのコンピューター上で同じコマンドを、少ないエラーで素早く実行することができるからです。

一括作業に組み込むそれぞれのコンピューターのホスト名を知っている必要があります。

root 権限を持つユーザーとして制御ワークステーションにログオン状態である必要があります。

パーティション・データベース・システムに参加する、すべての RS/6000 SP ワークステーションのホスト名をリストしたファイルを用意します。以下のようにして、ワークステーションのこのリストにコマンドを配布する一括作業をセットアップします。

1. 一括作業に組み込むすべてのワークステーションの ホスト名 をリストする、eeelist.txt というファイルを作成します。

   例えば、 workstation1 および workstation2 という 2 つの SP ノードを指定して、一括作業を作成しようとしているとします。 eeelist.txt の内容は以下のようになります。

   ```
   workstation1
   workstation2
   ```

2. 一括作業環境変数を更新します。以下のコマンドを入力して、このリストを更新します。

```
export WCOLL=path/eeelist.txt
```

ここで *path* は *eeelist.txt* が作成されたロケーションになります。*eeelist.txt* は、一括作業に組み込まれた RS/6000 SP ワークステーションをリストするために作成したファイルの名前です。

3. 以下のようなコマンドを入力して、一括作業ファイル内の名前が本当に、組み込みたいワークステーションであることを確認します。

```
dsh -q
```

以下のような出力が表示されます。

```
Working collective file /eeelist.txt:
workstation1
workstation2
Fanout: 64
```

NFS 稼働の検査 (Linux および UNIX)

データベース・パーティション環境をセットアップする前に、パーティション・データベース・システムに参加する各コンピューター上で、ネットワーク・ファイル・システム (NFS) が稼働していることを確認する必要があります。

それぞれのコンピューター上で、NFS が稼働している必要があります。

それぞれのコンピューター上で NFS が稼働していることを確認するには、以下のようにします。

AIX オペレーティング・システム

それぞれのコンピューター上で以下のコマンドを入力します。

```
lssrc -g nfs
```

NFS プロセスの「状況 (Status)」フィールドが、 「アクティブ (active)」と表示されていなければなりません。

それぞれのシステムで NFS が稼働していることを確認した後、 DB2 が必要とする特定の NFS プロセスを検査する必要があります。必要なプロセスとは、以下のものです。

```
rpc.lockd
rpc.statd
```

HP-UX および Solaris オペレーティング・システム

それぞれのコンピューター上で以下のコマンドを入力します。

```
showmount -e hostname
```

showmount コマンドを *hostname* パラメーターを指定せずに入力して、ローカル・システムを検査します。

NFS がアクティブでない場合には、以下のようなメッセージを受け取ります。

```
showmount: ServerA: RPC: Program not registered
```

それぞれのシステムで NFS が稼働していることを確認した後、 DB2 が必要とする特定の NFS プロセスを検査する必要があります。

```
rpc.lockd
rpc.statd
```

以下のコマンドを使用して、これらのプロセスを検査することができます。

```
ps -ef | grep rpc.lockd
ps -ef | grep rpc.statd
```

Linux オペレーティング・システム

それぞれのコンピューター上で以下のコマンドを入力します。

```
showmount -e hostname
```

showmount コマンドを *hostname* パラメーターを指定せずに入力して、ローカル・システムを検査します。

NFS がアクティブでない場合には、以下のようなメッセージを受け取ります。

```
showmount: ServerA: RPC: Program not registered
```

それぞれのシステムで NFS が稼働していることを確認した後、 DB2 が必要とする特定の NFS プロセスを検査する必要があります。必要なプロセスは rpc.statd です。

このプロセスを検査するには、 ps -ef | grep rpc.statd コマンドを使用します。

これらのプロセスが実行されていない場合は、オペレーティング・システムの資料を参照してください。

関与するコンピューター上のポート範囲の可用性の検査 (Linux および UNIX)

このタスクでは、関与するコンピューター上のポート範囲の可用性を検査するために必要なステップを記述します。ポート範囲は、高速コミュニケーション・マネージャー (FCM) が使用します。 FCM は、データベース・パーティション・サーバー間の通信を取り扱う DB2 のフィーチャーです。

関与するコンピューター上のポート範囲の可用性の検査は、インスタンス所有データベース・パーティション・サーバーをインストールしてから、なおかつ参加データベース・パーティション・サーバーをインストールする前に実行してください。

基本コンピューター上にインスタンス所有のデータベース・パーティション・サーバーをインストールする際に、DB2 はパーティション・データベース環境に参加している論理データベース・パーティション・サーバーの指定数に応じて、ポート範囲を予約します。デフォルトの範囲は 4 つのポートです。パーティション・データベース環境に加わっているサーバーごとに、/etc/services ファイルを FCM ポートのために手動で構成する必要があります。 FCM ポートの範囲は、関与するコンピューターで使用する論理区画の数によって異なります。最低 2 つの項目が必要です。それは、**DB2_<instance>** と **DB2_<instance>_END** です。関与するコンピューターに指定される FCM ポートについて、以下のような他の要件があります。

- 開始ポート番号は、1 次コンピューターの開始ポート番号に一致している必要があります。

- 後続のポートは順次番号付けする必要があります。
- 指定されるポート番号はフリーでなければなりません。

services ファイルに変更を加えるには、 root 権限が必要です。

以下のようにして、関与するコンピューター上のポート範囲の可用性を検査します。

1. /etc/services ディレクトリーにある services ファイルをオープンします。
2. DB2 高速コミュニケーション・マネージャー (FCM) 用に予約されたポートを探し出します。エントリーは以下のように表示されるはずです。

   ```
   DB2_db2inst1          60000/tcp
   DB2_db2inst1_1        60001/tcp
   DB2_db2inst1_2        60002/tcp
   DB2_db2inst1_END      60003/tcp
   ```

   DB2 は 60000 以降で使用できる最初の 4 つのポートを予約します。
3. それぞれの関与するコンピューター上で、services ファイルをオープンし、基本コンピューターのサービス・ファイルにある、DB2 FCM に予約されたポートが使用中でないかを確認します。
4. 必要なポートが関与するコンピューターで使用中である場合は、すべてのコンピューターで使用できるポート範囲を識別し、基本コンピューターのサービス・ファイルも含めて、それぞれのサービス・ファイルを更新します。

基本コンピューター上にインスタンス所有のデータベース・パーティション・サーバーをインストールした後、参加データベース・パーティション・サーバーに DB2 製品をインストールする必要があります。パーティション・サーバー用に生成された応答ファイル (デフォルト名 db2ese_addpart.rsp) を使用できますが、FCM ポート用に /etc/services ファイルを手動で構成する必要があります。FCM ポートの範囲は、現行マシン上で使用する論理パーティション数によって異なります。最小エントリーは、DB2_ と DB2__END の 2 つのエントリー用で、後は空きポート番号が続きます。各参加マシンで使用される FCM ポート番号の開始ポート番号は同じでなければならず、後続のポートは連続的に番号付けを行う必要があります。

パーティション DB2 サーバー用のファイル・システムの作成 (Linux)

このタスクは、パーティション・データベース・システムのセットアップの一部です。このタスクでは、以下の方法について説明します。

- DB2 ホーム・ファイル・システムの作成
- ホーム・ファイル・システムの NFS エクスポート
- それぞれの関与するコンピューターからのホーム・ファイル・システムの NFS マウント

パーティション・データベース・システムに参加するすべてのマシンで使用できるファイル・システムが必要です。このファイル・システムは、インスタンスのホーム・ディレクトリーとして使用されます。

1 つのデータベース・インスタンスに複数のマシンを使う構成の場合、 NFS (Network File System) を使用して、このファイル・システムを共有します。一般に

は、クラスター内の 1 つのマシンを使用し、 NFS を使用してファイル・システムをエクスポートします。そしてクラスター内の残りのマシンは、このマシンから NFS ファイル・システムをマウントします。ファイル・システムをエクスポートするマシンは、ローカルにマウントされたファイル・システムを所有しています。

コマンドの詳細は、Linux ディストリビューションの資料を参照してください。

このファイル・システムを作成するには、以下のステップを実行します。

1. 1 つのマシンで、ディスク・パーティションを選択するか、 fdisk を使用してそれを作成します。
2. mkfs のようなユーティリティーを使用し、このパーティション上にファイル・システムを作成します。ファイル・システムは、必要な DB2 プログラム・ファイルはもちろん、データベースに必要なスペースも十分含められるだけの大きさでなければなりません。
3. 作成したばかりのこのファイル・システムをローカル・マウントしてから、システムのリブートのたびにこのファイル・システムがマウントされるよう、/etc/fstab ファイルに項目を追加します。例:

   ```
   /dev/hda1    /db2home    ext3   defaults   1 2
   ```

4. ブート時に、自動的に NFS ファイル・システムを Linux へエクスポートするには、 /etc/exports ファイルへ項目を追加します。クラスター内に含まれるすべてのホスト名だけでなく、マシンのそれぞれの名前すべてを含めるようにします。さらに、クラスター内の各マシンに、「root」オプションを使用してエクスポートしたファイル・システムに対する、root 権限があることを確認します。

   /etc/exports ファイルは、以下のタイプの情報を含んだ ASCII ファイルです。

   ```
   /db2home machine1_name(rw) machine2_name(rw)
   ```

   以下を実行して、NFS ディレクトリーをエクスポートします。

   ```
   /usr/sbin/exports -r
   ```

5. クラスター内に残っている各マシンで、 /etc/fstab ファイルへ項目を追加し、ブート時にファイル・システムを自動的に NFS マウントさせるようにします。以下の例で示すように、マウント・ポイント・オプションを指定するときには、ブート時にファイル・システムがマウントされること、読み取り/書き込み可能なこと、ハード・マウントされること、bg (バックグラウンド) オプションが含まれること、そして setuid プログラムを適切に実行できることを確認します。

   ```
   fusion-en:/db2home /db2home nfs rw,timeo=7,
    hard,intr,bg,suid,lock
   ```

   fusion-en はマシン名を表します。

6. 以下のコマンドを入力し、エクスポートしたファイル・システムを、クラスター内の残りのマシンのそれぞれに NFS マウントします。

   ```
   mount /db2home
   ```

   マウント・コマンドに失敗したら、showmount コマンドを使い、 NFS サーバーの状況を調べます。例:

   ```
   showmount -e fusion-en
   ```

   fusion-en はマシン名を表します。

この showmount コマンドは、 fusion-en というマシンからエクスポートされるファイル・システムをリストするものです。このコマンドが失敗する場合、NFS サーバーが始動していない可能性があります。 NFS サーバーのルートで以下のコマンドを実行して、サーバーを手動で始動します。

```
/etc/rc.d/init.d/nfs restart
```

現在の実行レベルが 3 である場合には、ディレクトリー /etc/rc.d/rc3.d で K20nfs を S20nfs にリネームすることによって、このコマンドをブート時に自動的に実行させることができます。

7. 以下のステップが正常に実行されたことを確認します。
   a. クラスター内の 1 つのマシンで、インスタンスおよびホーム・ディレクトリーとして使用するファイル・システムを作成したこと。
   b. 1 つのデータベース・インスタンスに複数のマシンを使う構成の場合、 NFS を使用してこのファイル・システムをエクスポートしたこと。
   c. クラスター内の残りのマシンのそれぞれに、エクスポートしたファイル・システムをマウントしたこと。

パーティション・データベース・システム用の DB2 ホーム・ファイル・システムの作成 (AIX)

このタスクは、パーティション・データベース・システムのセットアップの一部です。このタスクでは、以下の方法について説明します。

- DB2 ホーム・ファイル・システムの作成
- ホーム・ファイル・システムの NFS エクスポート
- それぞれの関与するコンピューターからのホーム・ファイル・システムの NFS マウント

DB2 製品 DVD 上の内容と同じサイズのホーム・ファイル・システムを作成することをお勧めします。以下のコマンドを使用して、サイズを検査することができます (KB 単位で表示されます)。

```
du -sk <DVD mounting point>
```

DB2 インスタンスは、最低 50 MB のスペースを必要とします。十分なフリー・スペースがない場合には、内容をディスクにコピーする代わりに、それぞれの関与するコンピューターから DB2 製品 DVD をマウントすることができます。

以下の条件が必要です。

- ファイル・システムを作成するために root 権限が必要です。
- ファイル・システムが物理的に常駐するボリューム・グループを作成済みであることが必要です。

DB2 ホーム・ファイル・システムを作成、NFS エクスポート、および NFS マウントするには、以下のようなステップを行います。

DB2 ホーム・ファイル・システムの作成

ご使用のパーティション・データベース・システムの基本コンピューター (ServerA) に、 root 権限を持つユーザーとしてログオンし、ご使用のパーティション・データベース・システムのために /db2home というホーム・ファイル・システムを作成します。

1. **smit jfs** コマンドを入力します。
2. 「**ジャーナル・ファイル・システムの追加 (Add a Journaled File System)**」アイコンをクリックします。
3. 「**標準ジャーナル・ファイル・システムの追加 (Add a Standard Journaled File System)**」アイコンをクリックします。
4. そのファイル・システムを物理的に常駐させる既存のボリューム・グループを、「**ボリューム・グループ名 (Volume Group Name)**」リストから選択します。
5. 「**ファイル・システムのサイズ (512 バイト・ブロック単位) (数) (SIZE of file system (in 512-byte blocks) (Num.))**」フィールドで、ファイル・システムのサイズを設定します。このサイズ設定は 512 バイト・ブロック単位で列挙されます。したがって、インスタンス・ホーム・ディレクトリー用のファイル・システムだけを作成する必要がある場合には、180 000 (約 90 MB) を使用できます。インストールを実行するために製品 DVD イメージをコピーする必要がある場合、値 2 000 000 (約 1 GB) を使ってこれを作成できます。
6. このファイル・システムのマウント・ポイントを「**マウント・ポイント (MOUNT POINT)**」フィールドに入力します。この例では、マウント・ポイントは /db2home です。
7. 「**システムの再始動時に自動マウント (Mount AUTOMATICALLY at system restart)**」フィールドを「はい (Yes)」に設定します。

   残りのフィールドは、デフォルト設定のままにしてもかまいません。
8. 「**OK**」をクリックします。

DB2 ホーム・ファイル・システムのエクスポート

1. /db2home ファイル・システムを NFS エクスポートし、パーティション・データベース・システムの一員となるすべてのコンピューターで、このファイルを使えるようにします。
   a. **smit nfs** コマンドを入力します。
   b. 「**ネットワーク・ファイル・システム (NFS) (Network File System (NFS))**」アイコンをクリックします。
   c. 「**エクスポート・リストへのディレクトリーの追加 (Add a Directory to Exports List)**」アイコンをクリックします。
   d. パス名とエクスポートするディレクトリー (例えば /db2home) を、「**エクスポートするディレクトリーのパス名 (PATHNAME of directory to export)**」フィールドに入力します。
   e. パーティション・データベース・システムの一員となる各ワークステーションの名前を、「**root アクセスできるホスト (HOSTS allowed root access)**」フィールドに入力します。各名前の間の区切り文字としてコンマ (,) を使用します。例えば ServerA, ServerB, ServerC のようにします。高速相互接続を使用する場合、各ワークステーション

用の高速相互接続名もこのフィールドに指定することをお勧めします。残りのフィールドは、デフォルト設定のままにしてもかまいません。

f. 「**OK**」をクリックします。

2. ログアウトします。

それぞれの関与するコンピューターからの DB2 ホーム・ファイル・システムのマウント

以下のようなステップを行って、各 関与するコンピューター (ServerB、ServerC、ServerD) にログオンし、エクスポートしたファイル・システムを NFS マウントします。

1. **smit nfs** コマンドを入力します。
2. 「**ネットワーク・ファイル・システム (NFS) (Network File System (NFS))**」アイコンをクリックします。
3. 「**マウント用のファイル・システムの追加 (Add a File System for Mounting)**」アイコンをクリックします。
4. マウント・ポイントのパス名を「**マウント・ポイントのパス名 (パス) (PATHNAME of the mount point (Path))**」フィールドに入力します。

   マウント・ポイントのパス名は、DB2 ホーム・ディレクトリーを作成する場所になります。この例では、/db2home を使用します。
5. リモート・ディレクトリーのパス名を「**リモート・ディレクトリーのパス名 (PATHNAME of the remote directory)**」フィールドに入力します。

   例えば、「**マウント・ポイントのパス名 (パス) (PATHNAME of the mount point (Path))**」フィールドに入力したのと同じ値を入力してください。
6. ファイル・システムをエクスポートしたマシンのホスト名 を、「**リモート・ディレクトリーが置かれるホスト (HOST where the remote directory resides)**」フィールドに入力します。

   この値は、マウントしようとしているファイル・システムが作成されたマシンのホスト名です。

   パフォーマンスを向上させるには、作成したファイル・システムを高速相互接続を介して NFS マウントするとよいかもしれません。高速相互接続を介してそのファイル・システムをマウントする場合、その名前を「**リモート・ディレクトリーが置かれるホスト (HOST where the remote directory resides)**」フィールドに入力します。

   なんらかの理由で高速相互接続が使えなくなった場合、パーティション・データベース・システムに参加しているすべてのワークステーションが、その DB2 ホーム・ディレクトリーにアクセスできなくなることに注意してください。
7. 「**ただちにマウント、項目を /etc/filesystems に追加、またはこの両方 (MOUNT now, add entry to /etc/filesystems or both?)**」フィールドを「両方 (both)」に設定します。

8. 「**/etc/filesystems 項目はシステムの再始動時にディレクトリーをマウント (/etc/filesystems entry will mount the directory on system RESTART)**」フィールドを「はい (yes)」に設定します。
9. 「**この NFS ファイル・システムのモード (MODE for this NFS file system)**」フィールドを「読み取り/書き込み (read-write)」に設定します。
10. 「**ファイル・システムのソフト・マウントまたはハード・マウント (Mount file system soft or hard)**」フィールドを「ソフト (soft)」に設定します。

    ソフト・マウントとは、コンピューターが、際限なくディレクトリーのリモート・マウントを試みない ことを意味します。ハード・マウントとは、マシンが、際限なくディレクトリーのマウントを試みることを意味します。そのため、システム破損という問題を生じることがあります。このフィールドを「ソフト (soft)」に設定することをお勧めします。

    残りのフィールドは、デフォルト設定のままにしてもかまいません。
11. このファイル・システムをマウントするときは、必ず「**このファイル・システムで SUID および sgid プログラムを実行してもよい (Allow execution of SUID and sgid programs in this file system?)**」フィールドを「はい (Yes)」に設定してください。これがデフォルト設定です。
12. 「**OK**」をクリックします。
13. ログアウトします。

パーティション・データベース環境での DB2 サーバーのインストールに必要なユーザーの作成 (Linux)

DB2 データベースの操作には、3 つのユーザーおよびグループが必要です。この後の解説で使用しているユーザーおよびグループの名前を下の表に示してあります。各システムの命名規則と DB2 の命名規則に準拠している限り、独自のユーザー名とグループ名を指定することができます。

DB2 セットアップ・ウィザードを使用して DB2 製品をインストールする予定の場合は、DB2 セットアップ・ウィザードによりこれらのユーザーが作成されます。

表 10. 必要なユーザーおよびグループ

必要なユーザー	ユーザー名	グループ名
インスタンス所有者	db2inst1	db2iadm1
fenced ユーザー	db2fenc1	db2fadm1
DB2 Administration Server のユーザー	dasusr1	dasadm1

DB2 Administration Server ユーザーが既存ユーザーである場合は、インストール前にこのユーザーがすべての関与するコンピューター上になければなりません。 DB2 セットアップ・ウィザードを使用して、インスタンス所有のコンピューター上で DB2 Administration Server に新規ユーザーを作成する場合には、応答ファイルのイ

ンストール中にこの新規ユーザーが、関与するコンピューター上にも作成されます (必要であれば)。ユーザーが既に関与するコンピューター上に存在している場合には、そのユーザーは同じプライマリー・グループを持っている必要があります。

前提条件

- ユーザーおよびグループを作成するためには、root 権限が必要です。
- セキュリティー・ソフトウェアでユーザーとグループを管理する場合、DB2 ユーザーとグループを定義する際に追加の手順が必要になることがあります。

制約事項

作成するユーザー名は、オペレーティング・システムの命名規則と DB2 の命名規則に沿ったものでなければなりません。

これらの 3 種類のユーザーをすべて作成するには、以下のようなステップを実行します。

1. 基本コンピューターにログオンします。
2. 以下のようなコマンドを入力して、インスタンス所有者のグループ (例えば、db2iadm1)、UDF またはストアード・プロシージャーを実行するグループ (例えば、db2fadm1)、および DB2 Administration Server を所有するグループ (例えば、dasadm1) を作成します。

   ```
   groupadd -g 999 db2iadm1
   groupadd -g 998 db2fadm1
   groupadd -g 997 dasadm1
   ```

   使用する特定の各番号が現在どのマシン上にも存在していないことを確認してください。
3. 以下のようなコマンドを使用して、前のステップで作成した各グループに属するユーザーを作成します。それぞれのユーザーのホーム・ディレクトリーは、ユーザーが以前に作成し共用した DB2 ホーム・ディレクトリー (db2home) となります。

   ```
   useradd -u 1004 -g db2iadm1 -m -d /db2home/db2inst1 db2inst1
   useradd -u 1003 -g db2fadm1 -m -d /db2home/db2fenc1 db2fenc1
   useradd -u 1002 -g dasadm1 -m -d /home/dasusr1 dasusr1
   ```
4. 以下のようなコマンドを入力して、作成した各ユーザーの初期パスワードを設定します。

   ```
   passwd db2inst1    passwd db2fenc1    passwd dasusr1
   ```
5. ログアウトします。
6. 作成した各ユーザー (db2inst1、db2fenc1、および dasusr1) として、基本コンピューターにログオンします。それぞれのユーザーのパスワードを変更するようプロンプトで指示されることがあります。そのユーザーがシステムにログオンするのはこれが初めてだからです。
7. ログアウトします。
8. パーティション・データベース環境に参加するそれぞれのコンピューター上に、まったく同じユーザー・アカウントおよびグループ・アカウントを作成します。

パーティション・データベース環境での DB2 サーバーのインストールに必要なユーザーの作成 (AIX)

DB2 データベースの操作には、3 つのユーザーおよびグループが必要です。この後の解説で使用しているユーザーおよびグループの名前を下の表に示してあります。各システムの命名規則と DB2 の命名規則に準拠している限り、独自のユーザー名とグループ名を指定することができます。

DB2 セットアップ・ウィザードを使用して DB2 製品をインストールする予定の場合は、DB2 セットアップ・ウィザードによりこれらのユーザーが作成されます。

表 11. 必要なユーザーおよびグループ

必要なユーザー	ユーザー名	グループ名
インスタンス所有者	db2inst1	db2iadm1
fenced ユーザー	db2fenc1	db2fadm1
DB2 Administration Server のユーザー	dasusr1	dasadm1

DB2 Administration Server ユーザーが既存ユーザーである場合は、インストール前にこのユーザーがすべての関与するコンピューター上になければなりません。 DB2 セットアップ・ウィザードを使用して、インスタンス所有のコンピューター上で DB2 Administration Server に新規ユーザーを作成する場合には、応答ファイルのインストール中にこの新規ユーザーが、関与するコンピューター上にも作成されます (必要であれば)。ユーザーが既に関与するコンピューター上に存在している場合には、そのユーザーは同じプライマリー・グループを持っている必要があります。

前提条件

- ユーザーおよびグループを作成するためには、root 権限が必要です。
- セキュリティー・ソフトウェアでユーザーとグループを管理する場合、DB2 ユーザーとグループを定義する際に追加の手順が必要になることがあります。

制約事項

作成するユーザー名は、オペレーティング・システムの命名規則と DB2 の命名規則に沿ったものでなければなりません。

これらの 3 種類のユーザーをすべて作成するには、以下のようなステップを実行します。

1. 基本コンピューターにログオンします。
2. 以下のようなコマンドを入力して、インスタンス所有者のグループ (例えば、db2iadm1)、UDF またはストアード・プロシージャーを実行するグループ (例えば、db2fadm1)、および DB2 Administration Server を所有するグループ (例えば、dasadm1) を作成します。

```
mkgroup id=999 db2iadm1
mkgroup id=998 db2fadm1
mkgroup id=997 dasadm1
```

3. 以下のようなコマンドを使用して、前のステップで作成した各グループに属するユーザーを作成します。それぞれのユーザーのホーム・ディレクトリーは、ユーザーが以前に作成し共用した DB2 ホーム・ディレクトリー (db2home) となります。

```
mkuser id=1004 pgrp=db2iadm1 groups=db2iadm1 home=/db2home/db2inst1
   core=-1 data=491519 stack=32767 rss=-1 fsize=-1 db2inst1
mkuser id=1003 pgrp=db2fadm1 groups=db2fadm1 home=/db2home/db2fenc1
   db2fenc1
mkuser id=1002 pgrp=dasadm1 groups=dasadm1 home=/home/dasusr1
   dasusr1
```

4. 以下のようなコマンドを入力して、作成した各ユーザーの初期パスワードを設定します。

```
   passwd db2inst1
   passwd db2fenc1
   passwd dasusr1
```

5. ログアウトします。
6. 作成した各ユーザー (db2inst1、db2fenc1、および dasusr1) として、基本コンピューターにログオンします。それぞれのユーザーのパスワードを変更するようプロンプトで指示されることがあります。そのユーザーがシステムにログオンするのはこれが初めてだからです。
7. ログアウトします。
8. パーティション・データベース環境に参加するそれぞれのコンピューター上に、まったく同じユーザー・アカウントおよびグループ・アカウントを作成します。

第 7 章 DB2 サーバー製品のインストール

パーティション・データベース環境のセットアップ

このトピックでは、パーティション・データベース環境をセットアップする方法を説明します。 DB2 セットアップ・ウィザードを使用して、インスタンス所有データベース・サーバーをインストールし、関連するデータベース・サーバーの作成に使用する応答ファイルを作成することになります。

始める前に

注: パーティション・データベース環境は非ルート・インストールではサポートされません。

- 関連するすべてのコンピューターにコピーする必要がある DB2 Warehouse アクティベーション CD のライセンス・キーがあることを確認してください。
- パーティション・データベース環境に加わるそれぞれのコンピューターで、同数の連続ポートがフリーでなければなりません。例えば、パーティション・データベース環境が 4 台のコンピューターによって構成される場合、4 台のコンピューターのそれぞれで、同じ 4 つの連続ポートがフリーでなければなりません。インスタンス作成時に、現行のサーバー上の論理区画の数と同数のポートが、`/etc/services` (Linux と UNIX の場合) および `%SystemRoot%¥system32¥drivers¥etc¥services` (Windows の場合) で予約されます。これらのポートは高速コミュニケーション・マネージャーによって使用されます。予約されたポートは以下の形式になります。

  ```
  DB2_InstanceName
  DB2_InstanceName_1
  DB2_InstanceName_2
  DB2_InstanceName_END
  ```

  必須の項目は、開始 (DB2_InstanceName) および終了 (DB2_InstanceName_END) のポートのみです。他の項目は、他のアプリケーションがそれらのポートを使用しないようにサービス・ファイルに予約されます。
- 複数の関連する DB2 データベース・サーバーをサポートするには、DB2 のインストール先のコンピューターがアクセス可能ドメインに属していなければなりません。しかし、このコンピューターがドメインに属していない場合でも、このコンピューターにローカル・パーティションを追加できます。
- Linux システムと UNIX システムの場合は、パーティション・データベース・システム用にリモート・シェル・ユーティリティーが必要です。 DB2 データベース・システムでは、以下のリモート・シェル・ユーティリティーがサポートされています。
  - rsh
  - ssh

  デフォルトで DB2 データベース・システムは、リモート DB2 データベース・パーティションを起動する場合など、リモート DB2 ノードに対してコマンドを実行する際に rsh を使用します。 DB2 のデフォルトを使用するには、rsh-server

パッケージがインストールされている必要があります。詳細については、データベース・セキュリティー・ガイド の 『DB2 データベース・マネージャーのインストールおよび使用時のセキュリティー問題』 を参照してください。

rsh リモート・シェル・ユーティリティーを使用する場合は、inetd (または xinetd) をインストールして実行することも必要です。 ssh リモート・シェル・ユーティリティーを使用する場合は、DB2 のインストールが完了した直後に、**DB2RSHCMD** レジストリー変数を設定する必要があります。このレジストリー変数が設定されていない場合は、rsh が使用されます。

- Linux および UNIX のオペレーティング・システムでは、IP アドレス 『127.0.0.2』 がマシンの完全修飾ホスト名にマップされている場合に、etc ディレクトリーにある hosts ファイルに、その IP アドレスの項目が存在しないことを確認してください。

注: XML フィーチャーを使用すると、その後でパーティション・データベース環境を使用できなくなります。

このタスクについて

データベース・パーティションはデータベースの一区画であり、独自のデータ、索引、構成ファイル、およびトランザクション・ログで構成されます。パーティション・データベースとは、複数のパーティションを持つデータベースのことです。

手順

パーティション・データベース環境をセットアップするには、以下のようにします。

1. DB2 セットアップ・ウィザードを使用して、インスタンス所有データベース・サーバーをインストールします。詳細な作業手順については、ご使用のプラットフォームに該当する「DB2 サーバーのインストール」トピックを参照してください。
   - 「**インストール、応答ファイルの作成、またはその両方の選択**」ウィンドウで、「**インストール設定を応答ファイルに保管する**」オプションを選択していることを確認します。インストールが完了した後に、PROD_ESE.rsp と PROD_ESE_addpart.rsp の 2 つのファイルが DB2 セットアップ・ウィザードで指定したディレクトリーにコピーされます。ファイル PROD_ESE.rsp は、インスタンス所有データベース・サーバーの応答ファイルです。ファイル PROD_ESE_addpart.rsp は、関連するデータベース・サーバーの応答ファイルです。
   - 「**DB2 インスタンス用のパーティション・オプションのセットアップ**」ウィンドウで、「**複数パーティション・インスタンス**」を選択し、論理パーティションの最大数を入力します。
2. パーティション・データベース環境のすべての関連するコンピューターが DB2 インストール・イメージを利用できるようにします。
3. 関連するデータベース・サーバーの応答ファイル (PROD_ESE_addpart.rsp) を配布します。

4. 関連する各コンピューターに DB2 データベース・サーバーをインストールします。Linux と UNIX では db2setup コマンドを使用し、Windows では setup コマンドを使用します。

   Linux および UNIX

   DB2 データベース製品コードを使用できるディレクトリーに移動して、次のコマンドを実行します。

   ```
   ./db2setup -r /responsefile_directory/response_file_name
   ```

   Windows

   ```
   setup -u x:¥responsefile_directory¥response_file_name
   ```

   例えば、PROD_ESE_addpart.rsp を応答ファイルとして使用する場合には、次のコマンドを実行します。

   Linux および UNIX

   DB2 データベース製品コードを使用できるディレクトリーに移動して、次のコマンドを実行します。

   ```
   ./db2setup -r /db2home/PROD_ESE_addpart.rsp
   ```

   ここで、/db2home は応答ファイルをコピーしたディレクトリーです。

   Windows

   ```
   setup -u c:¥resp_files¥PROD_ESE_addpart.rsp
   ```

   ここで、c:¥resp_files¥ は応答ファイルをコピーしたディレクトリーです。

5. (Linux および UNIX のみ) db2nodes.cfg ファイルを構成します。 DB2 インストールでは、現行のコンピューターに使用することを希望する最大数の論理区画を確保するだけで、db2nodes.cfg ファイルの構成は行いません。 db2nodes.cfg ファイルを構成しない場合、インスタンスは単一パーティション・インスタンスのままです。
6. 参加しているサーバー上の services ファイルを更新して、 DB2 インスタンス用の対応する FCM ポートを定義します。 services ファイルは、次の場所にあります。
   - /etc/services (Linux および UNIX の場合)
   - %SystemRoot%¥system32¥drivers¥etc¥services （Windows の場合）

応答ファイルを使用した、関与するコンピューター上でのデータベース・パーティション・サーバーのインストール (Windows)

このタスクでは、DB2 セットアップ・ウィザードを使用して作成した応答ファイルを使用して、関与するコンピューターにデータベース・パーティション・サーバーをインストールします。

前提条件

- DB2 セットアップ・ウィザードを使用して、基本コンピューター上に DB2 コピーをインストールしていること。
- 関与するコンピューターにインストールするための応答ファイルを作成し、関与するコンピューターにそれをコピーしていること。

- 関与するコンピューターに対して管理権限を持っていること。
- DB2 製品 DVD の内容を関与するコンピューターにコピーしていること。

以下のようにして、応答ファイルを使用して、追加のデータベース・パーティション・サーバーをインストールします。

1. DB2 インストール用に定義したローカル管理者アカウントで、パーティション・データベース環境に関与するコンピューターにログオンします。
2. DB2 製品 DVD の内容をコピーしたディレクトリーに移動します。例:

   ```
   cd c:¥db2dvd
   ```

   ここで、db2dvd は、DB2 製品 DVD の内容をコピーしたディレクトリーの名前です。
3. コマンド・プロンプトから、以下のように setup コマンドを入力します。

   ```
   setup -u responsefile_directory¥response_file_name
   ```

   以下の例では、応答ファイル Addpart.file が c:¥responsefile ディレクトリーで検出されるようになります。この例に従うと、コマンドは以下のようになります。

   ```
   setup -u c:¥reponsefile¥Addpart.file
   ```
4. インストールが完了したならば、ログ・ファイルにあるメッセージをチェックします。ログ・ファイルは My Documents¥DB2LOG¥ ディレクトリーにあります。ログ・ファイルの末尾には、以下に類似した出力があるはずです。

   ```
   === Logging stopped: 5/9/2007  10:41:32 ===
   MSI (c) (C0:A8) [10:41:32:984]: Product: DB2
   Enterprise Server Edition - DB2COPY1 -- Installation
   operation completed successfully.
   ```
5. 基本コンピューター上にインスタンス所有のデータベース・パーティション・サーバーをインストールする際に、DB2 製品は、パーティション・データベース環境に参加している論理データベース・パーティション・サーバーの指定数に応じて、ポート範囲を予約します。デフォルトの範囲は 4 つのポートです。パーティション・データベース環境に加わっているサーバーごとに、/etc/services ファイルを FCM ポートのために手動で構成する必要があります。 FCM ポートの範囲は、関与するコンピューターで使用する論理区画の数によって異なります。最低 2 つの項目が必要です。それは、**DB2_<instance>** と **DB2_<instance>_END** です。関与するコンピューターに指定される FCM ポートについて、以下のような他の要件があります。
   - 開始ポート番号は、1 次コンピューターの開始ポート番号に一致している必要があります。
   - 後続のポートは順次番号付けする必要があります。
   - 指定されるポート番号はフリーでなければなりません。

それぞれの関与するコンピューターにログオンしてこれらのステップを繰り返す必要があります。

ローカル・コンピューターか、ネットワーク上の別のコンピューターにある DB2 資料に DB2 製品からアクセスできるようにする場合は、DB2 インフォメーショ

ン・センターをインストールする必要があります。 DB2 インフォメーション・センターには、DB2 データベース・システムと DB2 関連製品の資料が収録されています。

応答ファイルを使用した、関与するコンピューター上でのデータベース・パーティション・サーバーのインストール (Linux および UNIX)

このタスクでは、DB2 セットアップ・ウィザードを使用して作成した応答ファイルを使用して、関与するコンピューターにデータベース・パーティション・サーバーをインストールします。

前提条件

- DB2 セットアップ・ウィザードを使用して、基本コンピューター上に DB2 をインストールし、関与するコンピューターにインストールするための応答ファイルを作成していること。
- 関与するコンピューターに対して root 権限を持っている必要があります。

以下のようにして、応答ファイルを使用して、追加のデータベース・パーティション・サーバーをインストールします。

1. パーティション・データベース環境に参加するコンピューターに、root としてログオンします。
2. DB2 製品 DVD の内容をコピーしたディレクトリーに移動します。例:

   ```
   cd /db2home/db2dvd
   ```

3. db2setup コマンドを次のように入力します。

   ```
   ./db2setup -r /responsefile_directory/response_file_name
   ```

   この例では、応答ファイル AddPartitionResponse.file を /db2home ディレクトリーに保管しているので、コマンドは以下のようになります。

   ```
   ./db2setup -r /db2home/AddPartitionResponse.file
   ```

4. インストールが完了したならば、ログ・ファイルにあるメッセージをチェックします。

それぞれのコンピューターにログオンして、応答ファイル・インストールを実行する必要があります。

ローカル・コンピューターか、ネットワーク上の別のコンピューターにある DB2 資料に DB2 製品からアクセスできるようにする場合は、DB2 インフォメーション・センターをインストールする必要があります。 DB2 インフォメーション・センターには、DB2 データベース・システムと DB2 関連製品の資料が収録されています。

第 8 章 インストールした後に

インストールの検査

パーティション・データベース環境のインストールの検査 (Windows)

DB2 サーバーのインストールが成功したかを検査するためには、サンプル・データベースを作成し、SQL コマンドを実行してサンプル・データを検索し、データがすべての参加データベース・パーティション・サーバーに分散されているかを確認します。

すべてのインストール・ステップを完了していること。

以下のようにして、SAMPLE データベースを作成します。

1. SYSADM 権限を持つユーザーとして、基本コンピューター (ServerA) にログオンします。
2. db2sampl コマンドを入力して、 SAMPLE データベースを作成します。

   このコマンドの処理には、数分間かかることがあります。コマンド・プロンプトが戻ると、プロセスは完了です。

   SAMPLE データベースが作成されると、自動的にデータベース別名 SAMPLE としてカタログされます。
3. db2start コマンドを入力して、データベース・マネージャーを開始します。
4. 以下の DB2 コマンドを DB2 コマンド・ウィンドウから入力して、 SAMPLE データベースに接続し、部門 20 で作業しているすべての従業員のリストを検索します。

   ```
   db2 connect to sample
   db2 "select * from staff where dept = 20"
   ```
5. すべてのデータベース・パーティション・サーバーにデータが分散されたことを確認するため、DB2 コマンド・ウィンドウから以下のコマンドを入力します。

   ```
   db2 "select distinct dbpartitionnum(empno) from employee"
   ```

   出力では employee 表によって使用されるデータベース・パーティションをリストします。データベース内のデータベース・パーティションの数と、employee 表が作成された表スペースによって使用されるデータベース・パーティション・グループ内のデータベース・パーティションの数によって、それぞれの出力は異なります。

インストールを検査し終わったら、 SAMPLE データベースを除去してディスク・スペースを解放することができます。しかし、サンプル・アプリケーションを使用する予定の場合は、サンプル・データベースを維持しておくと便利です。

SAMPLE データベースをドロップするには、 db2 drop database sample コマンドを入力します。

パーティション・データベース・サーバーのインストールの検査 (Linux および UNIX)

DB2 サーバーのインストールが成功したかを検査するためには、サンプル・データベースを作成し、SQL コマンドを実行してサンプル・データを検索し、データがすべての参加データベース・パーティション・サーバーに分散されているかを確認します。

以下のステップを実行する前に、すべてのインストール・ステップが完了していることを確認してください。

以下のようにして、SAMPLE データベースを作成します。

1. 基本コンピューター (ServerA) に、インスタンス所有者ユーザーとしてログオンします。この例では、db2inst1 がインスタンス所有者ユーザーです。
2. db2sampl コマンドを入力して、 SAMPLE データベースを作成します。デフォルトでは、サンプル・データベースがインスタンス所有者のホーム・ディレクトリーに作成されます。この例では、/db2home/db2inst1/ がインスタンス所有者のホーム・ディレクトリーです。インスタンス所有者のホーム・ディレクトリーは、デフォルトのデータベース・パスです。

   このコマンドの処理には、数分間かかることがあります。完了メッセージはありません。コマンド・プロンプトが戻ると、プロセスは完了です。

   SAMPLE データベースが作成されると、自動的にデータベース別名 SAMPLE としてカタログされます。
3. db2start コマンドを入力して、データベース・マネージャーを開始します。
4. 以下の DB2 コマンドを DB2 コマンド・ウィンドウから入力して、 SAMPLE データベースに接続し、部門 20 で作業しているすべての従業員のリストを検索します。

   ```
   db2 connect to sample
   db2 "select * from staff where dept = 20"
   ```
5. すべてのデータベース・パーティション・サーバーにデータが分散されたことを確認するため、DB2 コマンド・ウィンドウから以下のコマンドを入力します。

   ```
   db2 "select distinct dbpartitionnum(empno) from employee"
   ```

   出力では employee 表によって使用されるデータベース・パーティションをリストします。実際の出力は、以下の要素に依存します。
   - データベース内のデータベース・パーティションの数
   - employee 表が作成された表スペースによって使用されるデータベース・パーティション・グループ内のデータベース・パーティションの数

インストールを検査し終わったら、 SAMPLE データベースを除去してディスク・スペースを解放することができます。 SAMPLE データベースをドロップするには、db2 drop database sample コマンドを入力します。

第 3 部 インプリメンテーションおよび保守

第 9 章 データベースの作成の前に

パーティション・データベース環境のセットアップ

複数パーティション・データベースを作成するという決定は、データベースを作成する前に行わなければなりません。データベース設計に関して行う決定の一部として、データベース・パーティションが提供できるパフォーマンス改善を利用するかどうかを決定しておかなければなりません。

パーティション・データベース環境では、CREATE DATABASE コマンドまたは sqlecrea() 関数を使ってデータベースを作成します。どちらの方法を使う場合も、db2nodes.cfg ファイルにリストされたパーティションを通して要求を行うことができます。

複数パーティション・データベース作成前に、どのデータベース・パーティションをそのデータベースのカタログ・パーティションとするかを選択しなければなりません。その後、そのデータベース・パーティションからデータベースを直接作成するか、またはそのデータベース・パーティションにアタッチされたリモート・クライアントからデータベースを作成できます。アタッチして CREATE DATABASE コマンドを実行するデータベース・パーティションは、その特定のデータベースに対する*カタログ・パーティション* になります。

カタログ・パーティションは、すべてのシステム・カタログ表が保管されるデータベース・パーティションです。システム表に対するすべてのアクセスは、このデータベース・パーティションを通して行わなければなりません。すべてのフェデレーテッド・データベース・オブジェクト (ラッパー、サーバー、ニックネームなど) は、このデータベース・パーティションのシステム・カタログ表に保管されます。

可能であれば、各データベースを別個のインスタンスの中に作成してください。これが可能でない場合 (つまり、1 インスタンス当たり複数のデータベースを作成しなければならない場合)、カタログ・パーティションを使用可能なデータベース・パーティションに配分させる必要があります。これを行うと、単一データベース・パーティションにおけるカタログ情報の競合が削減されます。

注: 他のデータでバックアップに必要な時間が増えてしまうため、定期的にカタログ・パーティションのバックアップをとり、 (可能ならば) そこにユーザー・データを書き込むのを避けるべきです。

データベースを作成すると、 db2nodes.cfg ファイルに定義されたすべてのデータベース・パーティションにわたって自動的に作成されます。

システムに最初のデータベースが作成されると、システム・データベース・ディレクトリーが作成されます。これは、作成した他のデータベースについての情報と一緒に追加されます。 UNIX の場合、システム・データベース・ディレクトリーは sqldbdir であり、ホーム・ディレクトリーの下、または DB2 データベースのインストール・ディレクトリーの下の sqllib ディレクトリーに配置されます。 UNIX では、このディレクトリーは、パーティション・データベース環境を形成するすべて

のデータベース・パーティションに対する唯一のシステム・データベース・ディレクトリーであるため、共有ファイル・システム (例えば、UNIX プラットフォーム上の NFS) に常駐しなければなりません。 Windows の場合、システム・データベース・ディレクトリーはインスタンス・ディレクトリー内に置かれます。

さらに、sqldbdir ディレクトリーにはシステム・インテンション・ファイルも置かれます。これは sqldbins と呼ばれ、データベース・パーティションが同期を維持できるようにするものです。このファイルも、すべてのデータベース・パーティションにわたって 1 つのディレクトリーしかないため、共有ファイル・システムに常駐しなければなりません。このファイルは、データベースを形成するすべてのデータベース・パーティションで共用されます。

データベース・パーティションを利用するためには、構成パラメーターを修正しなければなりません。 GET DATABASE CONFIGURATION コマンドおよび GET DATABASE MANAGER CONFIGURATION コマンドを使用して、特定のデータベースまたはデータベース・マネージャー構成ファイルの中の個々の項目の値を調べることができます。特定のデータベースまたはデータベース・マネージャー構成ファイルの個々の項目を修正するためには、それぞれ UPDATE DATABASE CONFIGURATION コマンドと UPDATE DATABASE MANAGER CONFIGURATION コマンドを使用します。

パーティション・データベース環境に影響を与えるデータベース・マネージャー構成パラメーターには、**conn_elapse**、 **fcm_num_buffers**、 **fcm_num_channels**、**max_connretries**、**max_coordagents**、**max_time_diff**、**num_poolagents**、および **stop_start_time**があります。

ノード構成ファイルの作成

データベースをパーティション・データベース環境で操作する場合、 db2nodes.cfg というノード構成ファイルを作成する必要があります。

このファイルは、並列機能を持ったデータベース・マネージャーを複数データベース・パーティションにわたって開始する前に、そのインスタンスに対するホーム・ディレクトリーの sqllib サブディレクトリーの中に配置されていなければなりません。このファイルには、1 つのインスタンスの中のすべてのデータベース・パーティションの構成情報が含まれ、そのインスタンスのすべてのデータベース・パーティションによって共用されます。

Windows での考慮事項

DB2 Enterprise Server Edition を Windows で使用している場合、インスタンス作成時にノード構成ファイルが作成されます。ノード構成ファイルは手動で作成したり変更したりしないでください。 db2ncrt コマンドを使用して、データベース・パーティション・サーバーをインスタンスに追加することができます。 db2ndrop コマンドを使用して、データベース・パーティション・サーバーをインスタンスからドロップすることができます。db2nchg コマンドを使用すれば、データベース・パーティション・サーバーの構成を変更することができます。たとえば、1 つのコンピューターから別のコンピューターへのデータベース・パーティション・サーバーの移動、 TCP/IP ホスト名の変更、または別の論理ポートやネットワーク名の選択を行うことができます。

注: インスタンスが削除された場合にデータの消失を避けるため、sqllib サブディレクトリーには、データベース・マネージャーによって作成されたもの以外のファイルまたはディレクトリーを作成しないでください。ただし、以下の 2 つの例外があります。システムがストアード・プロシージャーをサポートしている場合は、ストアード・プロシージャー・アプリケーションを sqllib サブディレクトリーの下の function サブディレクトリーに入れます。もう 1 つの例外は、ユーザー定義関数 (UDF) が作成される場合です。 UDF の実行可能コードは、同じディレクトリーに入れることが許されます。

ファイルには、1 つのインスタンスに属する各データベース・パーティションごとに 1 行が含まれます。それぞれの行は、以下の形式になっています。

dbpartitionnum hostname [*logical-port* [*netname*]]

トークンはブランクで区切られます。変数は、以下のとおりです。

dbpartitionnum

データベース・パーティションを固有に定義するデータベース・パーティション番号 (0 から 999 まで)。データベース・パーティション番号は、昇順でなければなりません。間の番号が抜けていてもかまいません。

いったんデータベース・パーティション番号が割り当てられると、それを変更することはできません。 (変更すると、データを分散する方法を指定する分散マップの中の情報が信用できないものになります。)

データベース・パーティションをドロップした場合、そのデータベース・パーティション番号は、追加する任意の新しいデータベース・パーティション用に再使用することができます。

データベース・パーティション番号は、データベース・ディレクトリー内にデータベース・パーティション名を生成するために使用されます。ノード名は、以下の形式になります。

```
NODE nnnn
```

nnnn はデータベース・パーティション番号で、左側はゼロで埋められます。このデータベース・パーティション番号は、 CREATE DATABASE コマンドおよび DROP DATABASE コマンドでも使用されます。

hostname

パーティション間通信のための IP アドレスのホスト名。ホスト名には、完全修飾名を使用します。 /etc/hosts ファイルも、完全修飾名を使用する必要があります。 db2nodes.cfg ファイルおよび /etc/hosts ファイルで完全修飾名を使用しない場合、エラー・メッセージ SQL30082N RC=3 を受け取る場合があります。

(netname が指定された場合は、例外です。この場合、netname がほとんどの通信で使用され、 hostname は db2start、db2stop、および db2_all でのみ使用されます。)

logical-port

このパラメーターの指定は任意であり、データベース・パーティションの論理ポート番号を指定します。この番号は、データベース・マネージャー・インスタンス名と一緒に使用され、 etc/services ファイルの中の TCP/IP サービス名項目を識別します。

IP アドレスと論理ポートの組み合わせは、既知のアドレスとして使用され、データベース・パーティション間の通信接続をサポートするために、すべてのアプリケーションの間で固有のものでなければなりません。

各 hostname について、1 つの *logical-port* は、0 かまたはブランク (0 がデフォルト) でなければなりません。この *logical-port* に関連付けられるデータベース・パーティションは、クライアントが接続するホスト上のデフォルトのノードです。これは、db2profile スクリプト内の **DB2NODE** 環境変数か、または sqlesetc() API を使用してオーバーライドすることができます。

netname

このパラメーターの指定は任意であり、それぞれが独自のホスト名を持つ、複数のアクティブな TCP/IP インターフェースを持ったホストをサポートするために使用されます。

以下の例は、RS/6000 SP システムのための可能なノード構成ファイルを示したものであり、 SP2EN1 には複数の TCP/IP インターフェースと 2 つの論理パーティションがあり、 DB2 データベースのインターフェースとして SP2SW1 を使用しています。この例は、データベース・パーティション番号が (0 ではなく) 1 から始まり、 *dbpartitionnum* の順番は間の番号が抜けていることも示しています。

表 12. データベース・パーティション番号の例示表

dbpartitionnum	*hostname*	*logical-port*	*netname*
1	SP2EN1.mach1.xxx.com	0	SP2SW1
2	SP2EN1.mach1.xxx.com	1	SP2SW1
4	SP2EN2.mach1.xxx.com	0	
5	SP2EN3.mach1.xxx.com		

好みのエディターを使用して、db2nodes.cfg ファイルを更新することができます。(例外: Windows ではエディターは使用しないでください。) ただし、データベース・パーティションでは、ノード構成ファイルが db2start を出したときにロックされ、db2stop がデータベース・マネージャーを停止させた後でアンロックされることが必要なため、ファイル内の情報の整合性を注意して保護する必要があります。ファイルがロックされている場合、db2start コマンドで、必要に応じて、ファイルを更新することができます。例えば、**RESTART** オプションまたは **ADD DBPARTITIONNUM** オプションを指定して db2start を発行することができます

注: db2stop コマンドが失敗し、ノード構成ファイルがアンロックされない場合、アンロックするために db2stop **FORCE** を発行してください。

DB2 ノード構成ファイルの形式

db2nodes.cfg ファイルを使用して、DB2 インスタンスに関与するデータベース・パーティション・サーバーを定義します。また、データベース・パーティション・サーバー通信に高速相互接続を使用する場合には、db2nodes.cfg ファイルを使用して、高速相互接続の IP アドレスまたはホスト名を指定します。

Linux および UNIX オペレーティング・システムでの db2nodes.cfg ファイルの形式は以下のとおりです。

nodenumber hostname logicalport netname resourcesetname

nodenumber、*hostname*、*logicalport*、*netname*、および *resourcesetname* の定義を以下にまとめます。

Windows オペレーティング・システムでの db2nodes.cfg ファイルの形式は以下のとおりです。

nodenumber hostname computername logicalport netname resourcesetname

Windows オペレーティング・システムでは、db2ncrt または db2 add db partition コマンドによって db2nodes.cfg にこれらの項目が追加されます。直接これらの行を追加したり、このファイルを編集したりしないでください。

nodenumber

0 から 999 の固有の番号。パーティション・データベース・システム内のデータベース・パーティション・サーバーを識別します。

パーティション・データベース・システムを拡大/縮小するには、それぞれのデータベース・パーティション・サーバーの項目を db2nodes.cfg ファイルに追加します。追加のデータベース・パーティション・サーバー用に選択する *nodenumber* 値は、昇順になっていなければなりませんが、その順序内にギャップがあってもかまいません。論理パーティション・サーバーを追加する予定があって、ノードをこのファイル内に論理的にグループに分けて保管しておきたい場合、 *nodenumber* の値と値の間にギャップを置いてもかまいません。

この項目は必須です。

hostname

FCM で使用するための、そのデータベース・パーティション・サーバーの TCP/IP ホスト名。この項目は必須です。

db2nodes.cfg ファイルで、IP アドレスの代わりにホスト名が提供されている場合、データベース・マネージャーはホスト名を動的に解決しようとします。解決は、マシン上の OS 設定で決定されているように、ローカル側または登録済みドメイン・ネーム・サーバー (DNS) の参照のいずれかによって行うことができます。

DB2 バージョン 9.1 から、TCP/IPv4 プロトコルと TCP/IPv6 プロトコルの両方がサポートされています。ホスト名を解決する方式が変更されました。

バージョン 9.1 より前のリリースでは、db2nodes.cfg ファイルで定義されたストリングを解決する方式が使用されていたのに対し、バージョン 9.1 以降では、db2nodes.cfg ファイルで短縮名が定義されている場合、完全修飾ドメイン・ネーム (FQDN) の解決を試行する方式が使用されます。完全修飾ホスト名の構成で短縮名を指定すると、ホスト名を解決するプロセスにおいて不要な遅延が発生する可能性があります。

ホスト名の解決を必要とする DB2 コマンドで遅延が発生しないようにするには、以下のいずれかの回避策を使用します。

1. db2nodes.cfg ファイルおよびオペレーティング・システムのホスト・ファイルで短縮名が指定されている場合、オペレーティング・システムのホスト・ファイルのホスト名に、短縮名および完全修飾ドメイン・ネームを指定します。

2. DB2 サーバーが IPv4 ポートで listen していることが分かっている場合に IPv4 アドレスのみを使用するには、以下のコマンドを発行します。

```
db2 catalog tcpip4 node db2tcp2 remote 192.0.32.67 server db2inst1
   with "Look up IPv4 address from 192.0.32.67"
```

3. DB2 サーバーが IPv6 ポートで listen していることが分かっている場合に IPv6 アドレスのみを使用するには、以下のコマンドを発行します。

```
db2 catalog tcpip6 node db2tcp3 1080:0:0:0:8:800:200C:417A server 50000
   with "Look up IPv6 address from 1080:0:0:0:8:800:200C:417A"
```

logicalport

データベース・パーティション・サーバー用の論理ポート番号を指定します。このフィールドは、論理データベース・パーティション・サーバーを実行するワークステーションで、個々のデータベース・パーティション・サーバーを指定するのに使います。

DB2 は、インストール時のパーティション間通信用に、/etc/services ファイル中でポート範囲 (60000 から 60003 など) を予約しています。db2nodes.cfg 中のこの *logicalport* フィールドは、この範囲内のどのポートを特定の論理パーティション・サーバーに割り当てるのかを指定します。

このフィールド用の項目がない場合のデフォルト値は 0 です。ただし、*netname* フィールドの項目を追加した場合、 *logicalport* フィールドに番号を入力しなければなりません。

論理データベース・パーティションを使用する場合、指定する *logicalport* 値は、 0 から開始し、昇順にしなければなりません (例えば、 0,1,2)。

さらに、1 つのデータベース・パーティション・サーバーに *logicalport* 項目を指定する場合、 db2nodes.cfg ファイルにリストされているそれぞれのデータベース・パーティション・サーバーごとに、 *logicalport* を指定する必要があります。

このフィールドがオプションであるのは、論理データベース・パーティションや高速相互接続を使用しない 場合だけです。

netname

FCM 通信での高速相互接続のホスト名または IP アドレスを指定します。

このフィールドの項目を指定すると、データベース・パーティション・サーバー相互の通信 (db2start、 db2stop、および db2_all コマンドで起動した通信を除く) は、高速相互接続を通して処理されます。

このパラメーターが必要なのは、データベース・パーティションの通信に高速相互接続を使用する場合だけです。

resourcesetname

resourcesetname は、ノードを開始するオペレーティング・システム・リソースを定義します。 *resourcesetname* は、プロセス類縁性をサポートし、Multiple Logical Node (MLN) で使用されます。このサポートには、ストリング・タイプのフィールドが備えられ、以前は quadname と呼ばれていました。

このパラメーターは、AIX、HP-UX、Solaris オペレーティング・システム上だけでサポートされています。

この概念は、AIX では「リソース・セット」と呼ばれ、Solaris オペレーティング・システムでは「プロジェクト」と呼ばれています。リソース管理について詳しくは、ご使用のオペレーティング・システムの資料を参照してください。

HP-UX 上では、*resourcesetname* パラメーターは PRM グループの名前です。詳しくは、 HP から「HP-UX Process Resource Manager User Guide (B8733-90007)」を参照してください。

Windows オペレーティング・システムでは、論理ノードのプロセス類縁性は、**DB2PROCESSORS** レジストリー変数で定義できます。

Linux オペレーティング・システムでは、*resourcesetname* 列により、システム上の Non-Uniform Memory Access (NUMA) ノードに対応する番号を定義します。 NUMA ポリシー・サポートを備えた 2.6 Kernel とともに、システム・ユーティリティーの numactl を使用できる状態にする必要があります。

resourcesetname パラメーターを使用する場合には、*netname* パラメーターの指定が必要です。

構成の例

以下の構成例を参考にして、ユーザーの環境に適切な構成を判別してください。

1 台のコンピューター、4 つのデータベース・パーティション・サーバー

クラスター化された環境を使用しておらず、 ServerA という 1 つの物理ワークステーション上に、 4 つのデータベース・パーティション・サーバーを設けようとした場合、 db2nodes.cfg ファイルを以下のように更新します。

```
0          ServerA        0
1          ServerA        1
2          ServerA        2
3          ServerA        3
```

2 台のコンピューター、1 台のコンピューターにつき 1 つのデータベース・パーティション・サーバー

ServerA および ServerB という 2 つの物理ワークステーションを、パーティション・データベース・システムに組み込む場合、以下のように db2nodes.cfg ファイルを更新します。

```
0          ServerA        0
1          ServerB        0
```

2 台のコンピューター、1 台のコンピューター上に 3 つのデータベース・パーティション・サーバー

ServerA および ServerB という 2 つの物理ワークステーションをパーティション・データベース・システムに組み込む場合に、ServerA が 3 つのデータベース・パーティション・サーバーを実行していれば、以下のように db2nodes.cfg ファイルを更新します。

```
4          ServerA        0
6          ServerA        1
8          ServerA        2
9          ServerB        0
```

2 台のコンピューター、高速スイッチを持つ 3 つのデータベース・パーティション・サーバー

ServerA および ServerB という 2 つのコンピューターをパーティション・データベース・システムに組み込む (ServerB は、2 つのデータベース・パーティション・サーバーを実行中) 場合に、 switch1 および switch2 という高速相互接続を使いたければ、以下のように db2nodes.cfg ファイルを更新します。

```
0          ServerA        0              switch1
1          ServerB        0              switch2
2          ServerB        1              switch2
```

resourcesetname の使用例

以下の例では、以下の制約事項が適用されます。

- この例は、構成中に高速相互接続がない場合の *resourcesetname* の使用法を示しています。
- *netname* は 4 つ目の列で、スイッチ名がなく *resourcesetname* を使用する場合は、この列に *hostname* も指定できます。*resourcesetname* を定義する場合は、5 つ目のパラメーターになります。リソース・グループ仕様は、db2nodes.cfg ファイル中の 5 つ目の列以外にすることはできません。したがって、リソース・グループを指定する場合は、 4 つ目の列も入力しなければなりません。 4 つ目の列は高速スイッチが対象になっています。
- 高速スイッチがないか使用しない場合には、 *hostname* を入力しなければなりません (2 つ目の列と同じ)。つまり、DB2 データベース管理システムは、db2nodes.cfg ファイル中の列のギャップ (または相互交換) をサポートしていません。既にこの制約事項は先頭 3 列に適用されていましたが、現在は 5 つの列すべてに適用されています。

AIX の例

AIX オペレーティング・システムの場合にリソース・セットをセットアップする方法の例を示します。

この例では、1 つの物理ノードに、 32 のプロセッサーと 8 つの論理データベース・パーティション (MLN) があります。この例では、個々の MLN にプロセス類縁性を備える方法を示します。

1. /etc/rset 中にリソース・セットを定義します。

```
DB2/MLN1:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000,sys/cpu.00001,sys/cpu.00002,sys/cpu.00003

DB2/MLN2:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004,sys/cpu.00005,sys/cpu.00006,sys/cpu.00007

DB2/MLN3:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
```

```
    resources = sys/cpu.00008,sys/cpu.00009,sys/cpu.00010,sys/cpu.00011

DB2/MLN4:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00012,sys/cpu.00013,sys/cpu.00014,sys/cpu.00015

DB2/MLN5:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00016,sys/cpu.00017,sys/cpu.00018,sys/cpu.00019

DB2/MLN6:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00020,sys/cpu.00021,sys/cpu.00022,sys/cpu.00023

DB2/MLN7:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00024,sys/cpu.00025,sys/cpu.00026,sys/cpu.00027

DB2/MLN8:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00028,sys/cpu.00029,sys/cpu.00030,sys/cpu.00031
```

2. 下記のコマンドを入力することによって、メモリー親和性を使用可能にします。

```
vmo -p -o memory_affinity=1
```

3. リソース・セットを使用するインスタンス許可を付与します。

```
chuser capabilities=
    CAP_BYPASS_RAC_VMM,CAP_PROPAGATE,CAP_NUMA_ATTACH  db2inst1
```

4. db2nodes.cfg 中に 5 つ目の列としてリソース・セット名を追加します。

```
1 regatta 0 regatta DB2/MLN1
2 regatta 1 regatta DB2/MLN2
3 regatta 2 regatta DB2/MLN3
4 regatta 3 regatta DB2/MLN4
5 regatta 4 regatta DB2/MLN5
6 regatta 5 regatta DB2/MLN6
7 regatta 6 regatta DB2/MLN7
8 regatta 7 regatta DB2/MLN8
```

HP-UX の例

この例は、4 つの CPU と 4 つの MLN のあるマシン上で PRM グループを使用して CPU を共用し、MLN 当たり 24% の CPU を共用し、4% を他のアプリケーション用に残しておく方法を示しています。DB2 インスタンス名は db2inst1 です。

1. /etc/prmconf の GROUP セクションを編集します。

```
  OTHERS:1:4::
 db2prm1:50:24::
  db2prm2:51:24::
   db2prm3:52:24::
  db2prm4:53:24::
```

2. /etc/prmconf にインスタンス所有者項目を追加します。

```
 db2inst1::::OTHERS,db2prm1,db2prm2,db2prm3,db2prm4
```

3. 以下のコマンドを入力し、グループを初期設定して CPU マネージャーを有効にします。

```
prmconfig -i
prmconfig -e CPU
```

4. 5 つ目の列として PRM グループ名を db2nodes.cfg に追加します。

```
1 voyager 0 voyager db2prm1
2 voyager 1 voyager db2prm2
3 voyager 2 voyager db2prm3
4 voyager 3 voyager db2prm4
```

対話式 GUI ツール xprm を使用して PRM の構成 (ステップ 1 から 3) を行うこともできます。

Linux の例

Linux オペレーティング・システムでは、*resourcesetname* 列により、システム上の Non-Uniform Memory Access (NUMA) ノードに対応する番号を定義します。 NUMA ポリシー・サポートを備えた 2.6 カーネルに加えて、numactl システム・ユーティリティーを使用できる状態にする必要があります。 Linux オペレーティング・システムの NUMA サポートの詳細については、numact1 のマニュアル・ページを参照してください。

1 台の NUMA コンピューターに 4 つのノードを設定し、それぞれの論理ノードに 1 つの NUMA ノードを関連付ける例を以下に示します。

1. NUMA 機能がシステムに存在することを確認します。
2. 次のコマンドを発行する。

```
$ numactl --hardware
```

以下のような出力が表示されます。

```
available: 4 nodes (0-3)
node 0 size: 1901 MB
node 0 free: 1457 MB
node 1 size: 1910 MB
node 1 free: 1841 MB
node 2 size: 1910 MB
node 2 free: 1851 MB
node 3 size: 1905 MB
node 3 free: 1796 MB
```

3. この例では、システムに 4 つの NUMA ノードがあります。 db2nodes.cfg ファイルを以下のように編集して、それぞれの MLN にシステム上の 1 つの NUMA ノードを関連付けます。

```
0 hostname 0 hostname 0
1 hostname 1 hostname 1
2 hostname 2 hostname 2
3 hostname 3 hostname 3
```

Solaris の例

Solaris バージョン 9 の場合にプロジェクトをセットアップする方法の例を示します。

この例では、1 つの物理ノードに 8 つのプロセッサーがあります。デフォルトのプロジェクト用に 1 つの CPU が使用され、Application Server 用に 3 つの CPU が使用され、DB2 用に 4 つの CPU が使用されます。インスタンス名は db2inst1 です。

1. エディターを使用して、リソース・プール構成ファイルを作成します。この例では、ファイルの名前は pool.db2 です。内容は以下のとおりです。

```
create system hostname
create pset pset_default (uint pset.min = 1)
create pset db0_pset (uint pset.min = 1; uint pset.max = 1)
create pset db1_pset (uint pset.min = 1; uint pset.max = 1)
create pset db2_pset (uint pset.min = 1; uint pset.max = 1)
create pset db3_pset (uint pset.min = 1; uint pset.max = 1)
create pset appsrv_pset (uint pset.min = 3; uint pset.max = 3)
create pool pool_default (string pool.scheduler="TS";
     boolean pool.default = true)
create pool db0_pool (string pool.scheduler="TS")
create pool db1_pool (string pool.scheduler="TS")
create pool db2_pool (string pool.scheduler="TS")
create pool db3_pool (string pool.scheduler="TS")
create pool appsrv_pool (string pool.scheduler="TS")
associate pool pool_default (pset pset_default)
associate pool db0_pool (pset db0_pset)
associate pool db1_pool (pset db1_pset)
associate pool db2_pool (pset db2_pset)
associate pool db3_pool (pset db3_pset)
associate pool appsrv_pool (pset appsrv_pset)
```

2. 以下のように、/etc/project ファイルを編集して DB2 プロジェクトと appsrv プロジェクトを追加します。

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
appsrv:4000:App Serv project:root::project.pool=appsrv_pool
db2proj0:5000:DB2 Node 0 project:db2inst1,root::project.pool=db0_pool
db2proj1:5001:DB2 Node 1 project:db2inst1,root::project.pool=db1_pool
db2proj2:5002:DB2 Node 2 project:db2inst1,root::project.pool=db2_pool
db2proj3:5003:DB2 Node 3 project:db2inst1,root::project.pool=db3_pool
```

3. リソース・プールを作成します: `# poolcfg -f pool.db2`
4. リソース・プールをアクティブにします: `# pooladm -c`
5. db2nodes.cfg ファイル中に 5 つ目の列としてプロジェクト名を追加します。

```
0 hostname 0 hostname db2proj0
1 hostname 1 hostname db2proj1
2 hostname 2 hostname db2proj2
3 hostname 3 hostname db2proj3
```

パーティション・データベース環境でのマシンのリストの指定

デフォルトでは、コンピューターのリストは、ノード構成ファイル db2nodes.cfg から取得されます。

以下のようにして、これをオーバーライドすることができます。

- 環境変数 RAHOSTFILE をエクスポート (Linux および UNIX プラットフォームの場合) または設定 (Windows の場合) することによって、コンピューターのリストが含まれるファイルのパス名を指定する。

- 環境変数 RAHOSTLIST をエクスポート (Linux および UNIX プラットフォームの場合) または設定 (Windows の場合) することによって、リストを明示的に、名前のストリングとしてスペースで区切って指定する。

  注: これらの環境変数が両方とも指定されている場合は、 RAHOSTLIST の方が優先します。

注: Windows の場合、ノード構成ファイル内で不整合が生じないようにするために、このファイルを編集しないでください。インスタンスにあるコンピューターのリストを取得するには、 db2nlist コマンドを使用してください。

パーティション・データベース環境でのマシンのリストからの重複項目の除去

1 つのコンピューターで複数の論理データベース・パーティション・サーバーを使って DB2 Enterprise Server Edition を実行している場合、 db2nodes.cfg にはそのコンピューターに対する項目が複数含まれます。

この状況では、各コンピューターにつき 1 回だけコマンドを実行するのか、それとも db2nodes.cfg ファイルにリストされている各論理データベース・パーティションごとに 1 回ずつコマンドを実行するのかを、 rah コマンドが知っていなければなりません。コンピューターを指定するには rah コマンドを使用します。論理データベース・パーティションを指定するには db2_all コマンドを使用します。

注: Linux および UNIX プラットフォームでは、コンピューターを指定した場合、通常、 rah はコンピューター・リストから重複を除去します。ただし、論理データベース・パーティションを指定した場合、 db2_all は以下の割り当てをコマンドの前に付加します。

```
export DB2NODE=nnn  (Korn シェル構文の場合)
```

ここで、nnn は、目的のデータベース・パーティション・サーバーにコマンドを経路指定するために、 db2nodes.cfg ファイル内の対応する行から取得されるデータベース・パーティション番号です。

論理データベース・パーティションを指定するときには、 <<-nnn< および <<+nnn< 接頭部シーケンスを使って、 1 つを除くすべての論理データベース・パーティションが含まれるようにリストを制限したり、1 つだけを指定したりすることができます。これを行うのは、データベース・パーティションをカタログするコマンドを最初に実行して、完了後に他のすべてのデータベース・パーティション・サーバーで (おそらく並列に) 同じコマンドを実行するような場合です。 db2 restart database コマンドを実行するときには、通常、これが必要です。これを行うには、カタログ・パーティションのデータベース・パーティション番号を知っておく必要があります。

rah コマンドを使用して db2 restart database を実行するとき、重複項目はコンピューターのリストから除去されます。しかし「″」接頭部を指定する場合、「″」接頭部は各コンピューターではなく各データベース・パーティション・サーバーに送信することを意味するため、重複は除去されません。

ノード構成ファイルの更新 (Linux および UNIX)

このタスクは、db2nodes.cfg ファイルを更新して、関与するコンピューターのための項目を組み込むためのステップを提供します。

ノード構成ファイル (db2nodes.cfg) は、インスタンス所有者のホーム・ディレクトリーにありますが、これには、どのサーバーがパーティション・データベース環境下のインスタンスに参加するかを DB2 に知らせる構成情報が入っています。パーティション・データベース環境にあるそれぞれのインスタンスごとに、db2nodes.cfg ファイルがあります。

db2nodes.cfg ファイルには、インスタンスに参加するそれぞれのサーバーごとに 1 つの項目がなければなりません。インスタンスを作成すると、db2nodes.cfg ファイルが自動的に作成され、インスタンス所有のサーバーの項目が追加されます。

例えば、 DB2 セットアップ・ウィザードを使用して DB2 インスタンスを作成した場合は、インスタンス所有サーバー ServerA 上で、 db2nodes.cfg ファイルが以下のように更新されます。

```
0     ServerA     0
```

前提条件

- 関与するコンピューターのすべてに DB2 アプリケーションがインストールされていなければなりません。
- 基本コンピューター上に DB2 インスタンスが存在していなければなりません。
- ユーザーは SYSADM 権限を持つユーザーでなければなりません。
- 以下の条件のいずれかが当てはまる場合、構成例と、DB2 ノード構成ファイル・トピックの形式で提供されるファイル形式情報を検討してください。
  - データベース・パーティション・サーバー間での通信に高速スイッチの使用を予定している。
  - パーティション構成が複数の論理パーティションを持つことになる。

制約事項

『手順』のステップで使用されているホスト名は、完全修飾ホスト名でなければなりません。

以下に示すステップを実行して、db2nodes.cfg ファイルを更新します。

1. インスタンス所有者としてログオンします。(この例では、db2inst1 がインスタンス所有者)
2. 以下のコマンドを入力して、DB2 インスタンスが停止することを確認します。

   ```
   INSTHOME/sqllib/adm/db2stop
   ```

   INSTHOME は、インスタンス所有者のホーム・ディレクトリーです (db2nodes.cfg ファイルは、インスタンスの実行中はロックされ、インスタンスの停止時にしか編集できません)。

   例えば、ご使用のインスタンス・ホーム・ディレクトリーが /db2home/db2inst1 である場合には、以下のコマンドを入力します。

```
/db2home/db2inst1/sqllib/adm/db2stop
```

3. それぞれの DB2 インスタンスの項目を、 .rhosts ファイルに追加します。以下の内容を追加して、ファイルを更新します。

   ```
   <hostname> <db2instance>
   ```

   <hostname> はデータベース・サーバーの TCP/IP ホスト名で、<db2instance> はデータベース・サーバーへのアクセスに使用するインスタンスの名前です。

4. 個々の参加サーバーの項目を、 db2nodes.cfg ファイルに追加します。まず最初に db2nodes.cfg ファイルを表示すると、以下のような項目があるはずです。

   ```
   0     ServerA     0
   ```

   この項目には、データベース・パーティション・サーバー番号 (ノード番号)、データベース・パーティション・サーバーが常駐するサーバーの TCP/IP ホスト名、およびデータベース・パーティション・サーバーの論理ポート番号が含まれます。

   例えば、4 つのコンピューターを備えていて、それぞれのコンピューター上にデータベース・パーティション・サーバーが 1 つずつあるパーティション構成をインストールする場合には、db2nodes.cfg が更新されて、以下のように表示されるはずです。

   ```
   0     ServerA     0
   1     ServerB     0
   2     ServerC     0
   3     ServerD     0
   ```

5. db2nodes.cfg ファイルの更新が完了してから、INSTHOME/sqllib/adm/db2start コマンドを入力します (*INSTHOME* は、インスタンス所有者のホーム・ディレクトリー)。例えば、ご使用のインスタンス・ホーム・ディレクトリーが /db2home/db2inst1 である場合には、以下のコマンドを入力します。

   ```
   /db2home/db2inst1/sqllib/adm/db2start
   ```

6. ログアウトします。

複数の論理ノードのセットアップ

DB2 Enterprise Server Edition では、各コンピューターにデータベース・パーティション・サーバーが 1 つずつ割り当てられるように構成するのが一般的です。ただし、状況によっては、複数のデータベース・パーティション・サーバーを同一のコンピューターで実行したほうがよい場合もあります。

つまり、構成にコンピューターの数よりも多くのデータベース・パーティションが含まれることがあります。その場合、それらのデータベース・パーティションが*同じ* インスタンスに参加しているのであれば、そのコンピューターでは*複数のデータベース・パーティション* または *複数の論理ノード* が実行されているといいます。それらのデータベース・パーティションが異なる複数のインスタンスに参加している場合には、このコンピューターは複数の論理パーティションをホスティングしていません。

複数の論理パーティションがサポートされているため、次のような 3 種類の構成の中から選ぶことができます。

- 各コンピューターにデータベース・パーティション・サーバーが 1 つずつ用意されている、標準的な構成。
- 1 台のコンピューターに複数のデータベース・パーティション・サーバーがある、複数論理パーティション構成。
- 複数のコンピューターのそれぞれで複数の論理パーティションが実行される構成。

複数の論理パーティションを使用する構成は、対称マルチプロセッサー (SMP) アーキテクチャーのコンピューター上でシステムが照会を実行するときに便利です。さらに、1 つのマシンに複数の論理パーティションを構成すると、いずれかのコンピューターで障害が発生した場合にも効果を発揮します。あるコンピューターで障害が発生しても (その結果、そのマシン上の 1 つまたは複数のデータベース・パーティション・サーバーが使用できなくなっても)、 DB2START NODENUM コマンドを使用すれば、他のコンピューターでそのデータベース・パーティション・サーバーを再始動できます。これにより、ユーザー・データを確実に引き続き利用できます。

もう 1 つの利点は、複数の論理パーティションがあれば SMP ハードウェア構成を十分に活用できることです。さらに、データベース・パーティション・サーバーが小さくなるので、データベース・パーティションや表スペースのバックアップとリストア、索引の作成などのタスクを実行するときのパフォーマンスが向上します。

複数の論理パーティションの構成

複数の論理パーティションを構成するには、2 つの方法があります。

- db2nodes.cfg ファイル内で論理パーティション (データベース・パーティション) を構成します。構成後、db2start コマンドとその関連 API を使用して、すべての論理パーティションとリモート・パーティションを開始できます。

  注: Windows 環境では、システム内にデータベースが 1 つもない場合は、db2ncrt を使用してデータベース・パーティションを追加する必要があります。1 つまたは複数のデータベースがある場合は、db2start addnode コマンドを使用します。Windows 内では、 db2nodes.cfg ファイルを手動で編集することは絶対に避けてください。
- 他の論理パーティション (ノード) がすでに実行している、別のプロセッサー上で論理パーティションを再始動します。この場合、 db2nodes.cfg 内で論理パーティションに指定したホスト名とポート番号をオーバーライドできます。

db2nodes.cfg 内で論理パーティション (ノード) を構成するには、このファイル内に項目を作成して、データベース・パーティションの論理ポート番号を割り当てなければなりません。次の構文を使用する必要があります。

```
nodenumber hostname logical-port netname
```

注: Windows 環境では、システム内にデータベースが 1 つもない場合は、db2ncrt を使用してデータベース・パーティションを追加する必要があります。1 つまたは複数のデータベースがある場合は、db2start addnode コマンドを使用します。Windows 内では、 db2nodes.cfg ファイルを手動で編集することは絶対に避けてください。

Windows での db2nodes.cfg ファイルの形式は、 UNIX での同じファイルとは異なります。Windows での列形式は、次のとおりです。

```
nodenumber hostname computername logical_port netname
```

ホスト名には、完全修飾名を使用します。 /etc/hosts ファイルも、完全修飾名を使用する必要があります。 db2nodes.cfg ファイルおよび /etc/hosts ファイルで完全修飾名を使用しない場合、エラー・メッセージ SQL30082N RC=3 を受け取る場合があります。

FCM 通信にとって十分な数のポートを etc ディレクトリーの services ファイルに定義しなければなりません。

照会のパーティション間並列処理を使用可能にする

パーティション間並列処理は、データベース・パーティションの数およびこれらのデータベース・パーティションにわたるデータの配分に基づいて、自動的に行われます。

注: データベース・パーティション内または非パーティション・データベース内で並列処理を利用するためには、構成パラメーターを修正しなければなりません。例えば、パーティション内並列処理を使用して、対称マルチプロセッサー (SMP) マシン上の複数のプロセッサーを利用することができます。

データ・ロードの並列処理を使用可能にする

ロード・ユーティリティーは、自動的に並列処理を使用可能にします。または、LOAD コマンドで以下のパラメーターを使うことができます。

- CPU_PARALLELISM
- DISK_PARALLELISM

パーティション・データベース環境では、ターゲット表が複数のデータベース・パーティション上に定義されるとき、データ・ロード用のパーティション間並列処理が自動的に発生します。データ・ロード用のパーティション間並列処理は、 OUTPUT_DBPARTNUMS を指定することによってオーバーライドできます。さらにロード・ユーティリティーもまた、ターゲット・データベース・パーティションのサイズに応じてデータベース・パーティション並列処理を自動的に使用可能にします。load ユーティリティーによって選択される並列処理の度合いの最大値を制御するには、MAX_NUM_PART_AGENTS を使用することができます。データベース・パーティション並列処理は、 ANYORDER とともに PARTITIONING_DBPARTNUMS を指定することによってオーバーライドできます。

索引の作成の並列処理を使用可能にする

索引の作成中に並列処理を使用可能にするには、以下のとおりにしてください。

- 表は、並列処理の益が得られる十分な大きさでなければなりません。
- 1 つの SMP コンピューターで、複数プロセッサーが使用可能でなければなりません。

データベースまたは表スペースのバックアップで入出力並列処理を使用可能にする

データベースまたは表スペースのバックアップで入出力並列処理を使用可能にするには、以下のようにします。

1. 複数の宛先メディアを使用します。
2. 複数のコンテナーを定義することによって並列入出力用の表スペースを構成します。または、複数ディスクを使用する単一のコンテナーを使い、 DB2_PARALLEL_IO レジストリー変数を適切に設定します。並列入出力を利用する場合は、コンテナーを定義する前に必要な作業の意味を考慮してください。これは、必要が生じるたびに常に行えるとは限りません。データベースや表スペースをバックアップする必要が生じる前に、あらかじめ計画しておく必要があります。
3. BACKUP コマンドで PARALLELISM パラメーターを使用し、並列処理の度合いを指定します。
4. BACKUP コマンドで WITH num-buffers BUFFERS パラメーターを使用し、並列処理の度合いに見合う十分なバッファーを使用できるようにします。バッファーの数は、宛先メディア、選択した並列処理の度合い、そして多少の余分を合計した数に等しいものにします。

   また以下のような、バックアップ・バッファー・サイズを使用します。

   - 可能な限り大きなサイズにする。 4 MB か 8 MB (1024 か 2048 ページ) が良いようです。
   - 少なくとも、バックアップする表スペースの可能な最大数を含められるだけの大きさにする (extentsize * コンテナー数)。

データベースまたは表スペースのリストアで入出力並列処理を使用可能にする

データベースまたは表スペースのリストアで入出力並列処理を使用可能にするには、以下のようにします。

- 複数のソース・メディアを使用します。
- 並列入出力の表スペースを構成します。コンテナーを定義する前に、このオプションの使用について決定しておく必要があります。これは、必要が生じるたびに常に行えるとは限りません。データベースや表スペースをリストアする必要が生じる前に、あらかじめ計画しておく必要があります。
- RESTORE コマンドで PARALLELISM パラメーターを使用し、並列処理の度合いを指定します。
- RESTORE コマンドで WITH num-buffers BUFFERS パラメーターを使用し、並列処理の度合いに見合う十分なバッファーを使用できるようにします。バッファーの数は、宛先メディア、選択した並列処理の度合い、そして多少の余分を合計した数に等しいものにします。

  また以下のような、リストア・バッファー・サイズを使用します。

  – 可能な限り大きなサイズにする。 4 MB か 8 MB (1024 か 2048 ページ) が良いようです。
  – 少なくとも、リストアする表スペースの可能な最大数を含められるだけの大きさにする (extentsize * コンテナー数)。
  – バックアップ・バッファー・サイズと同じか、その倍数である。

照会のパーティション内並列処理を使用可能にする

このほか GET DATABASE CONFIGURATION コマンドおよび GET DATABASE MANAGER CONFIGURATION コマンドを使用して、特定のデータベースまたはデータベース・マネージャー構成ファイルの中の個々の項目の値を調べることもできます。特定のデータベースまたはデータベース・マネージャー構成ファイルの個々の項目を修正するためには、それぞれ UPDATE DATABASE CONFIGURATION コマンドと UPDATE DATABASE MANAGER CONFIGURATION コマンドを使用します。

パーティション内並列処理に影響を与える構成パラメーターには、 *max_querydegree* と *intra_parallel* のデータベース・マネージャー・パラメーター、および *dft_degree* データベース・パラメーターがあります。

照会のパーティション内並列処理を行わせるためには、 1 つまたは複数のデータベース構成パラメーター、データベース・マネージャー構成パラメーター、プリコンパイル・オプションまたは BIND オプション、または特殊レジスターを修正する必要があります。

intra_parallel
: 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・マネージャーでパーティション内並列処理を使用できるかどうかを指定します。デフォルトでは、パーティション内並列処理を使用しません。

max_querydegree
: このデータベース・マネージャー構成パラメーターは、このインスタンスで実行中の SQL ステートメントで使用される、パーティション内並列処理の最大度を指定します。 SQL ステートメントは、データベース・パーティション内で並列操作を行うとき、このパラメーターを超える数を使用することはありません。さらに、*max_querydegree* の値を使用するためには、構成パラメーター *intra_parallel* を ″YES″ に設定する必要もあります。この構成パラメーターのデフォルト値は -1 です。この値は、オプティマイザーによって決められた並列処理の度合いがシステムで使用されることを意味します。それ以外の場合は、ユーザー指定の値が使用されます。

dft_degree
: DEGREE BIND オプションおよび CURRENT DEGREE 特殊レジスターに対するデフォルトを提供するデータベース構成パラメーター。デフォルト値は 1 です。値 ANY は、オプティマイザーによって決められた並列処理の度合いがシステムで使用されることを意味します。

DEGREE
: 静的 SQL に対するプリコンパイルまたはバインドのオプション。

CURRENT DEGREE
: 動的 SQL に対する特殊レジスター。

データ・サーバー容量の管理

データ・サーバーの容量が、現在または将来の必要を満たさない場合、ディスク・スペースを追加して追加のコンテナーを作成するか、メモリーを追加することによってその容量を拡張することができます。これらの単純なストラテジーで、必要な容量を追加できない場合は、プロセッサーまたは物理パーティションを追加することを考慮してください。環境を変更してシステムを拡大または縮小するときは、そのような変更が、データのロード、またはデータベースのバックアップおよびリストアなどのデータベース手順に与える影響をよく知っている必要があります。

プロセッサーの追加

1 台のプロセッサーを単一パーティション・データベース構成で使用しており、しかもそれを最大限まで使用してしまっている場合、プロセッサーを追加するか、データベース・パーティションを追加したほうがよいでしょう。プロセッサーを追加することの利点は、処理力が増すことです。 SMP システムでは、プロセッサーはメモリーおよびストレージ・システム・リソースを共用します。すべてのプロセッサーが 1 つのシステムに収まっているため、システム間の通信回線、およびシステム間のタスクの調整などについての、オーバーヘッドとなる考慮事項が加わることがありません。ロード、バックアップ、およびリストアなどのユーティリティーは、追加のプロセッサーを利用することができます。

注: Solaris オペレーティング・システムのように、オペレーティング・システムによっては、プロセッサーを動的にオンラインまたはオフラインに調整することができるものがあります。

プロセッサーを追加する場合、使用されるプロセッサーの数を左右する、いくつかのデータベース構成パラメーターを検討し、変更してください。次のデータベース構成パラメーターは、使用するプロセッサー数を判別するもので、更新の必要の可能性があります。

- デフォルトのパーティション内並行度 (dft_degree)
- 最大並列処理の多重度 (max_querydegree)
- パーティション内並列処理機能の使用可能化 (intra_parallel)

また、アプリケーションが並列処理を実行する方法を決定するパラメーターを評価することも必要です。

通信に TCP/IP が使用されている環境では、 DB2TCPCONNMGRS レジストリー変数の値を考慮する必要があります。

物理パーティションの追加

データベース・マネージャーが現在パーティション・データベース環境にある場合、個別の単一プロセッサーまたはマルチプロセッサーの物理パーティションを追加することによって、データ・ストレージ・スペースを広げ、処理能力を高めることができます。各データベース・パーティション上のメモリーおよびストレージ・システム・リソースは、他のデータベース・パーティションと共用されません。データベース・パーティションの追加の結果、通信およびタスク調整の問題が発生することがありますが、この選択には、

複数のシステム間でデータおよびユーザー・アクセスのバランスを取ることができる、という利点があります。データベース・マネージャーはこの環境をサポートします。

データベース・パーティションの追加は、データベース・マネージャー・システムの実行中でも停止中でも行えます。ただし、システムの実行中にデータベース・パーティションを追加すると、データベースがその新しいデータベース・パーティションにマイグレーションする前にシステムを一度停止し、再始動することが必要です。

新しいデータベース・パーティションを追加するときは、その処理が完了し、新しいサーバーが正常にシステムに組み込まれるまでは、新規データベース・パーティションを活用するデータベースのドロップまたは作成を行うことはできません。

高速コミュニケーション・マネージャー

高速コミュニケーション・マネージャー (Windows)

高速コミュニケーション・マネージャー (FCM) は、同じインスタンスに属する DB2 サーバー製品の通信サポートを提供します。それぞれのデータベース・パーティション・サーバーには、データベース・パーティション・サーバー間の通信機能を提供する 1 つの FCM 送信側デーモンと 1 つの FCM 受信側デーモンがあり、これにより、エージェント要求を処理して、メッセージ・バッファーをやり取りします。インスタンスを開始すると、FCM デーモンが開始されます。

データベース・パーティション・サーバーの相互通信で障害が発生した場合や、または通信が再確立された場合、 FCM スレッドは情報 (データベース・システム・モニターで照会できる情報) を更新し、適切な処置 (影響を受けたトランザクションのロールバックなど) をとらせます。データベース・システム・モニターを使用すると、FCM 構成パラメーターを設定するのに役立ちます。

FCM メッセージ・バッファーの数は、データベース・マネージャー構成パラメーターの*fcm_num_buffers*で指定することができます。FCM チャネルの数は、データベース・マネージャー構成パラメーターの *fcm_num_channels* で指定することができます。データベース・マネージャー構成パラメーターの *fcm_num_buffers* および *fcm_num_channels* は、デフォルト値として AUTOMATIC に設定されますこれらのパラメーターのいずれかが AUTOMATIC に設定されていると、FCM はリソースの使用状況をモニターして、リソースを徐々に解放していきます。これらのパラメーターは、AUTOMATIC に設定したままにしておくことをお勧めします。

高速コミュニケーション・マネージャー (Linux および UNIX)

高速コミュニケーション・マネージャー (FCM) は、データベース・パーティション・フィーチャー (DPF) を使用する DB2 サーバー製品の通信サポートを提供します。

複数パーティション・インスタンスの場合、それぞれのデータベース・パーティション・サーバーには、データベース・パーティション・サーバー間の通信機能を提供する 1 つの FCM 送信側デーモンと 1 つの FCM 受信側デーモンがあり、これ

により、エージェント要求を処理して、メッセージ・バッファーをやり取りします。複数パーティション・インスタンスを開始すると、FCM デーモンが開始されます。

データベース・パーティション・サーバー間の通信で障害が発生したり、通信が再確立されたりすると、FCM デーモンは情報を更新します。データベース・システム・モニターを使用してこの情報を照会できます。FCM デーモンは必要なアクションも起動します。そのようなアクションの例としては、影響を受けたトランザクションのロールバックがあります。データベース・システム・モニターを使用すると、 FCM 構成パラメーターを設定するのに役立ちます。

FCM メッセージ・バッファーの数は、 データベース・マネージャー構成パラメーターの *fcm_num_buffers* で指定することができます。また、FCM チャネルの数は、データベース・マネージャー構成パラメーターの *fcm_num_channels* で指定することができます。データベース・マネージャー構成パラメーターの *fcm_num_buffers* および *fcm_num_channels* は、デフォルト値として AUTOMATIC に設定されます。これらのパラメーターのいずれかがAUTOMATICに設定されていると、FCM はリソースの使用状況をモニターして、リソースを徐々に解放していきます。これらのパラメーターは、AUTOMATIC に設定したままにしておくことをお勧めします。

FCM 通信でデータベース・パーティション間通信を使用可能にする

パーティション・データベース環境では、データベース・パーティション間のほとんどの通信は、高速コミュニケーション・マネージャー (FCM) によって処理されます。

データベース・パーティションで FCM を使用可能にし、他のデータベース・パーティションとの通信ができるようにするには、下記に示すように、データベース・パーティションの `etc` ディレクトリーの `services` ファイル内にサービス項目を作成する必要があります。FCM は、指定されたポートを使用して通信を行います。同じホスト上に複数のデータベース・パーティションを定義している場合、以下に示すように、ある範囲のポートを定義しなければなりません。

高速コミュニケーション・マネージャー (FCM) のメモリーを手動で構成しようとする前に、まず FCM バッファー数 (*fcm_num_buffers*) および FCM チャネル数 (*fcm_num_channels*) の自動設定 (デフォルト設定) から始めることをお勧めします。この設定が適切であるかを判断するには、FCM アクティビティーのシステム・モニター・データを使用してください。

Windows での考慮事項

DB2 Enterprise Server Edition を Windows 環境で使用している場合、TCP/IP のポート範囲は次のものによって自動的にサービス・ファイルに追加されます。

- インストール・プログラムがインスタンスを作成したり新しいデータベース・パーティションを追加したりするときに、インストール・プログラムによって
- db2icrt ユーティリティーが新しいインスタンスを作成するときに、db2icrt ユーティリティーによって

- db2ncrt ユーティリティーがコンピューターに最初のデータベース・パーティションを追加したときに、 db2ncrt ユーティリティーによって

サービス項目の構文は、以下のとおりです。

```
DB2_instance port/tcp #comment
```

DB2_*instance*
: *instance* の値は、データベース・マネージャー・インスタンスの名前です。名前の中のすべての文字は小文字でなければなりません。 db2puser というインスタンス名であるとすれば、 DB2_db2puser というように指定します。

port/**tcp**
: データベース・パーティションのために予約する TCP/IP ポート。

#*comment*
: この項目と関連付ける任意の注釈。注釈の前には、ポンド記号 (#) を付けなければなりません。

etc ディレクトリーの services ファイルが共用されている場合、ファイル内に割り当てられるポートの数は、そのインスタンス内の複数のデータベース・パーティションの最大数より大きいか等しくなるようにしなければなりません。ポートを割り当てる場合には、バックアップとして使用できるプロセッサーもその数の中に入れるようにしなければなりません。

etc ディレクトリーの services ファイルが共有されていない場合、同じ考慮事項が適用されます。ただし追加の考慮事項として、DB2 データベース・インスタンス用に定義される項目は、 etc ディレクトリーのすべての services ファイルで同じでなければなりません (パーティション・データベース環境に適用されない他の項目は、同じである必要はありません)。

1 つのインスタンス内で同じホスト上に複数のデータベース・パーティションがある場合、使用する FCM のために複数のポートを定義しなければなりません。そのためには、etc ディレクトリーの services ファイルの中に 2 行を組み込んで、割り当てるポートの範囲を示します。最初の行は最初のポートを指定し、2 番目の行は複数のポート・ブロックの終わりを示します。以下の例では、sales というインスタンスに 5 つのポートが割り当てられます。これは、そのインスタンスには、 5 つを超えるデータベース・パーティションを持つプロセッサーはないことを意味します。例:

```
DB2_sales         9000/tcp
DB2_sales_END     9004/tcp
```

注: END は、大文字でのみ指定しなければなりません。また、両方の下線 (_) 文字も含めるようにしなければなりません。

データベース・パーティション・サーバーの相互通信を有効にする (Linux および UNIX)

このタスクは、パーティション・データベース・システムに参加するデータベース・パーティション・サーバーの相互通信を有効にする方法について説明します。データベース・パーティション・サーバーの相互通信は、高速コミュニケーション・マネージャー (FCM) によって処理されます。 FCM を有効にするには、ポー

トまたはポート範囲を、パーティション・データベース・システム内のそれぞれのコンピューター上の /etc/services ファイルに入れて保管する必要があります。

root 権限を付与されたユーザー ID がなければなりません。

このタスクは、インスタンスに参加しているすべてのコンピューター上で実行する必要があります。

FCM に予約するポートの数は、インスタンス内のいずれかのコンピューターによってホストされるか、またはホストされる可能性のあるデータベース・パーティションの最大数と等しくします。

次の例では、db2nodes.cfg ファイルには以下のエントリーが含まれています。

```
0 server1 0
1 server1 1
2 server2 0
3 server2 1
4 server2 2
5 server3 0
6 server3 1
7 server3 2
8 server3 3
```

FCM ポート番号の先頭を 60000 から始めて番号を付けるとします。この場合、以下のようになります。

- server1 では、その 2 つのデータベース・パーティション用に 2 つのポート (60000、60001) が使用されます。
- server2 では、その 3 つのデータベース・パーティション用に 3 つのポート (60000、60001、60002) が使用されます。
- server3 では、その 4 つのデータベース・パーティション用に 4 つのポート (60000、60001、60002、60003) が使用されます。

この場合、すべてのコンピューターで、60000、60001、60002、および 60003 を予約する必要があります。これはインスタンス内のいずれかのコンピューターによって必要とされる最大のポート範囲であるためです。

データベース・パーティションをあるコンピューターから別のコンピューターにフェイルオーバーするために、High Availability Cluster Multi-Processing (HACMP™) や Tivoli System Automation (TSA) などの高可用性ソリューションを使用している場合は、潜在的なポート要件を明らかにする必要があります。例えば、あるコンピューターで通常 4 つのデータベース・パーティションがホストされている場合に、別のコンピューターの 2 つのデータベース・パーティションがこのコンピューターにフェイルオーバーされる可能性がある場合は、このコンピューターに 6 つのポートを計画する必要があります。

インスタンスを作成すると、ポート範囲が基本コンピューターに予約されます。基本コンピューターは、インスタンス所有コンピューターともいいます。ただし、/etc/services ファイルに最初に追加されたポート範囲が、お客様のニーズに不十分な場合は、さらにエントリーを手動で追加して予約されたポートの範囲を拡張する必要があります。

以下のようにして、/etc/services を使用したパーティション・データベース環境でのサーバー間の通信を有効にします。

1. root 権限を持つユーザーとして、基本コンピューター (インスタンス所有のコンピューター) にログオンします。
2. インスタンスを作成します。
3. /etc/services ファイルに保管されているデフォルトのポート範囲を参照します。基本構成に加えて、 FCM ポートは以下のようになっているはずです。

```
db2c_db2inst1         50000/tcp
#Add FCM port information
DB2_db2inst1          60000/tcp
DB2_db2inst1_1        60001/tcp
DB2_db2inst1_2        60002/tcp
DB2_db2inst1_END      60003/tcp
```

   デフォルトでは、最初のポート (50000) は接続要求に予約され、また 60000 以上の使用できる最初の 4 つのポートが FCM 通信に予約されます。これらのポートは、インスタンス所有データベース・パーティション・サーバー用に 1 つ、論理データベース・パーティション・サーバー (インストール完了後にコンピューターに追加するよう選択できる) 用に 3 つです。

   ポート範囲には、開始エントリーと終了 (END) エントリーを含める必要があります。中間のエントリーはオプションです。中間値を明示的に含めることは、他のアプリケーションによるこれらのポートの使用を防止することに役立つ場合がありますが、これらのエントリーはデータベース・マネージャーによっては検査されません。

   DB2 ポート項目は、以下のような形式を使用します。

```
DB2_instance_name_suffix  port_number/tcp  # comment
```

   説明:

   - *instance_name* は、パーティション・インスタンスの名前です。
   - *suffix* は、最初の FCM ポートには使用されません。中間のエントリーは、最低のポート番号と最高のポート番号の間にあるポート番号です。最初と最後の FCM ポートの間に中間のエントリーを含める場合は、*suffix* を追加するポートごとに 1 つずつ増加させた整数で構成します。例えば、2 番目のポートには 1 と番号を付け、3 番目のポートには 2 と番号を付けるなどしてユニークになるようにします。 END という語を最後のエントリーの *suffix* に使用する必要があります。
   - *port_number* は、データベース・パーティション・サーバーの通信用に予約するポート番号です。
   - *comment* は、エントリーについて説明するオプションのコメントです。
4. FCM 通信用に予約されたポートが十分に存在しているようにしてください。予約されたポートの範囲が不十分な場合は、新規エントリーをこのファイルに追加します。
5. インスタンスに参加するすべてのコンピューターごとに root ユーザーとしてログオンし、同一のエントリーを /etc/services ファイルに追加します。

第 10 章 パーティション・データベース環境の作成と管理

初期データベース・パーティション・グループ

データベースを最初に作成するときに、 db2nodes.cfg ファイルに指定されたすべてのデータベース・パーティションについて、データベース・パーティションが作成されます。その他のデータベース・パーティションは、ADD DBPARTITIONNUM および DROP DBPARTITIONNUM VERIFY コマンドを使用して追加またはドロップすることができます。

以下の 3 つのデータベース・パーティション・グループが定義されます。

- SYSCATSPACE 表スペース用の IBMCATGROUP (システム・カタログ表を保持します)
- TEMPSPACE1 表スペース用の IBMTEMPGROUP (データベース処理の間に作成された一時表を保持します)
- USERSPACE1 表スペース用の IBMDEFAULTGROUP (デフォルトで、ユーザー表と索引を保持します)

データベース・パーティション・グループの作成

CREATE DATABASE PARTITION GROUP ステートメントを使用してデータベース・パーティション・グループを作成します。このステートメントは、表スペース・コンテナーおよび表データが存在するデータベース・パーティションのセットを指定します。

コンピューターおよびシステムが使用可能で、かつパーティション・データベース環境を扱えなければなりません。すでに DB2 Enterprise Server Editionを購入してインストールしてあります。データベースが存在していることが必要です。

このステートメントは、以下のことも行います。

- データベース・パーティション・グループの分散マップの作成。
- 分散マップ ID の生成。
- 以下のカタログ表へのレコードの挿入。
  - SYSCAT.DBPARTITIONGROUPS
  - SYSCAT.PARTITIONMAPS
  - SYSCAT.DBPARTITIONGROUPDEF

コントロール・センターを使用してデータベース・パーティションを作成するには、以下のようにします。

1. オブジェクト・ツリーを順に展開し、「**データベース・パーティション・グループ**」フォルダーを表示します。
2. 「**データベース・パーティション・グループ**」 フォルダーを右クリックして、ポップアップ・メニューから「**作成**」を選択します。

3. 「データベース・パーティション・グループの作成」ウィンドウで、情報をすべて入力し、矢印を使用してノードを「使用可能ノード (Available nodes)」ボックスから「選択済みデータベース・パーティション」ボックスに移動して、「**OK**」をクリックします。

コマンド行を使用してデータベース・パーティション・グループを作成するには、以下のように入力します。

```
CREATE DATABASE PARTITION GROUP db-partition-group-name
   ON DBPARTITIONNUM (db-partition-number1,db-partition-number1)
```

または

```
CREATE DATABASE PARTITION GROUP db-partition-group-name
   ON DBPARTITIONNUMS (db-partition-number1
   TO db-partition-number2)
```

例えば、データベース内のデータベース・パーティションのサブセット上にいくつかの表をロードするとします。以下のコマンドを使用して、少なくとも 3 つのデータベース・パーティション (0 ～ 2) からなるデータベース内に、 2 つのデータベース・パーティション (1 と 2) のデータベース・パーティション・グループを作成します。

```
CREATE DATABASE PARTITION GROUP mixng12 ON DBPARTITIONNUM (1,2)
```

または

```
CREATE DATABASE PARTITION GROUP mixng12 ON DBPARTITIONNUMS (1 TO 2)
```

CREATE DATABASE コマンドまたは sqlecrea() API は、デフォルトのシステム・データベース・パーティション・グループである、IBMDEFAULTGROUP、IBMCATGROUP、および IBMTEMPGROUP も作成します。

データベース・パーティション・グループの表スペース

複数パーティションを持つデータベース・パーティション・グループの中に表スペースを置くことによって、その表スペース内のすべての表が、そのデータベース・パーティション・グループ内の各データベース・パーティションにわたって分割、つまりパーティション化されます。

表スペースはデータベース・パーティション・グループ内に作成されます。いったん 1 つのデータベース・パーティション・グループ内に入ると、表スペースは、そこにとどまらなければならず、別のデータベース・パーティション・グループに変更することはできません。 CREATE TABLESPACE ステートメントは、表スペースとデータベース・パーティション・グループを関連付けるために使用されます。

データベース・パーティションを管理する

コントロール・センターの「パーティション」ビューを使用して、パーティションの開始および停止、パーティションのドロップおよびトレース、診断ログの表示を行えます。

データベース・パーティションを処理するには、あるいは DB2 ログを表示するには、インスタンスにアタッチするための権限が必要です。 SYSADM または DBADM 権限がある人から、特定のインスタンスにアクセスするための権限を付与してもらうことができます。

 IBM サポートから要求された場合は、指示されたオプションを使用してトレース・ユーティリティーを実行します。トレース・ユーティリティーは DB2 の動作情報を記録し、この情報を読み取れる形にフォーマットします。詳しくは、『db2trc - トレース: DB2』 トピックを参照してください。

重要: DB2 カスタマー・サービスあるいは技術サポート担当者によって指示された場合に限り、トレース機能を使用してください。

「診断ログ」ウィンドウを使用して、DB2 トレース・ユーティリティーによってログに記録されたテキスト情報を表示します。

「パーティション」ビューには以下の情報が表示されます。

ノード番号

この列には、アイコンおよびノード番号が含まれています。ノード番号はユニークな値で、0 から 999 までの値を使用することができます。この番号は、db2nodes.cfg ファイル内に保管されています。ノード番号は、途中に空きがあることもありますが、昇順で表示されます。

一度割り当てられたノード番号は変更できません。この安全機能によって、分散マップ内の情報 (データのパーティション方法を詳細に記述している) が変更されないようになります。

ホスト名

ホスト名は、内部通信用の高速コミュニケーション・マネージャー (FCM) によって使用される IP アドレスです。 (ただし、スイッチ名が指定されれば、FCM はスイッチ名を使用します。 この場合、ホスト名は DB2START、DB2STOP、および db2_all に対してだけ使用されます。) ホスト名は、db2nodes.cfg ファイル内に保管されています。

ポート番号

ポート番号は、ノード用の論理ポート番号です。この番号は、データベース・マネージャー・インスタンス名と一緒に使用され、/etc/services ファイルの中の TCP/IP サービス名項目を識別します。この番号は、db2nodes.cfg ファイル内に保管されています。

IP アドレス (ホスト名) と論理ポートの組み合わせは、予約済みのアドレスとして使用され、ノード間の通信接続をサポートするためにすべてのアプリケーションの間で固有でなければなりません。

表示されたホスト名ごとに、1 つのポート番号が 0 になります。ポート番号 0 は、クライアントが接続するホスト上のデフォルト・ノードを示しています。(この動作を取り消すには、db2profile スクリプト内の **DB2NODE** 環境変数を使用してください。)

スイッチ名

スイッチ名 (またはネット名) は、複数のアクティブな TCP/IP インターフェース (それぞれに固有なホスト名が指定されている) を持つホストをサポ

ートするために使用されます。これは、高速スイッチを共有するノード間での高速通信 (FCM としても知られる) にも使用できます。スイッチ名は、db2nodes.cfg ファイル内に保管されています。スイッチ名が db2nodes.cfg ファイル内に指定されていない場合には、スイッチ名はホスト名と同じです。

Resourcesetname

resourcesetname は、ノードを開始するオペレーティング・システム・リソースを定義します。 resourcesetname は、プロセス類縁性をサポートし、複数論理ノード (MLN) で使用され、ストリング・タイプのフィールドが備えられ、以前は quadname と呼ばれていました。

1. 次のようにして、「パーティション」ビューをオープンします。コントロール・センターで、パーティションを表示するインスタンスが表示されるまでオブジェクト・ツリーを展開します。目的のインスタンスを右クリックし、ポップアップ・メニューから「**オープン**」 → 「**パーティション**」を選択します。 「パーティション」ビューがオープンします。
2. パーティションを開始するには、1 つ以上のパーティションを強調表示し、「**パーティション**」 → 「**開始**」を選択します。 選択したパーティションが開始します。
3. パーティションを停止するには、1 つ以上のパーティションを強調表示し、「**パーティション**」 → 「**停止**」を選択します。 選択したパーティションが停止します。
4. パーティション上でトレース・ユーティリティーを実行するには、次のようにしてください。
   a. 次のようにして、「DB2 トレース」ウィンドウをオープンします。パーティションを強調表示し、「**パーティション**」 → 「**サービス**」 → 「**トレース**」を選択します。 「DB2 トレース」ウィンドウがオープンします。
   b. トレース・オプションを指定します。
   c. 情報の記録を開始するには「**開始**」をクリックし、情報の記録を停止するには「**停止**」をクリックし、情報をファイルに保管するには「**別名保管**」をクリックします。
   d. オプション: ログを表示します。
   e. 求められた場合は、IBM サポートにトレース出力を送信します。

パーティション・データベース環境でのデータベース・パーティションの追加

システムの稼働中、またはシステムの停止時に、パーティション・データベース・システムにデータベース・パーティションを追加することができます。新しいサーバーを追加するのは時間のかかる作業なので、これはデータベース・マネージャーがすでに実行しているときに行うこともできます。

ADD DBPARTITIONNUM コマンドを使用すると、システムにデータベース・パーティションを追加することができます。このコマンドは以下のようにして呼び出すことができます。

- `db2start` のオプションとして
- コマンド行プロセッサー ADD DBPARTITIONNUM コマンドを使って

- API 関数 sqleaddn を使って
- API 関数 sqlepstart を使って

システムが停止している場合、db2start を使用します。実行中の場合は、その他の選択肢のどれでも使用できます。

新規のデータベース・パーティションを ADD DBPARTITIONNUM コマンドを使用してシステムに追加すると、すでにインスタンスに入っているすべてのデータベースはその新規データベース・パーティションに拡張されます。システム管理者は、データベース用の TEMPORARY 表スペースに使用するためのコンテナーを指定することもできます。このコンテナーは、以下のようにすることができます。

- 各データベースのカタログ・パーティションに定義されるものと同じにする(これはデフォルトです)。
- 他のデータベース・パーティションに定義されているものと同じにする。
- まったく作成しない。 ALTER TABLESPACE ステートメントを使用して、各データベースに TEMPORARY 表スペース・コンテナーを追加してからでないと、データベースを使用できません。

新規データベース・パーティション上のデータベースは、 1 つ以上のデータベース・パーティション・グループを変更して新規のデータベース・パーティションを含めるようにするまでは、データを入れるために使用できません。

単にご使用のシステムにデータベース・パーティションを追加するだけでは、単一パーティション・データベースを複数パーティション・データベースに変更することはできません。データベース・パーティション間でデータを再分散するには、該当する各表に分散キーが必要だからです。分散キーは、複数パーティション・データベースで表が作成されたときに自動的に生成されます。単一パーティション・データベースの場合、分散キーは CREATE TABLE または ALTER TABLE SQL ステートメントによって明示的に作成できます。

注: システムにデータベースが定義されておらず、 UNIX オペレーティング・システムで Enterprise Server Edition を実行している場合、db2nodes.cfg ファイルを編集して、新規データベース・パーティション 定義を追加します。ここで説明されている手順はどれも使用してはなりません。適用されるのは、データベースが存在する場合だけです。

Windows についての考慮事項: Windows 上の Enterprise Server Edition を使用している場合、インスタンスにデータベースがなければ、 DB2NCRT コマンドを使ってデータベース・システムを拡大または縮小してください。しかし、データベースがあれば、DB2START ADDNODE コマンドを使用し、システムのサイズ変更時に既存のデータベースごとにデータベース・パーティションを確実に作成してください。 Windows では、ノード構成ファイル (db2nodes.cfg) を手作業で編集することは避けてください。これを行うと、ファイル内での整合性が失われるおそれがあります。

実行中のデータベース・システムへのデータベース・パーティションの追加

システムが稼働していて、アプリケーションがデータベースに接続されている間に、パーティション・データベース環境に新しいデータベース・パーティションを追加することができます。しかし、新規に追加されたサーバーがすべてのデータベースに使用できるようになるのは、データベース・マネージャーがシャットダウンされて再び始動された後です。

コマンド行を使用してデータベース・パーティションを実行中のデータベース・マネージャーに追加するには、次のようにします。

1. 既存のデータベース・パーティションで、DB2START コマンドを実行します。

   すべてのプラットフォームで、DBPARTITIONNUM、ADD DBPARTITIONNUM、HOSTNAME、PORT、および NETNAME パラメーターに対して新データベース・パーティション値を指定します。 Windows プラットフォームでは、COMPUTER、USER、および PASSWORD パラメーターも指定します。

   また、データベースに作成する必要のある任意の TEMPORARY 表スペース・コンテナー定義のためのソースを指定することもできます。表スペース情報を提供しないと、 TEMPORARY 表スペース・コンテナー定義は各データベースのカタログ・パーティションから検索されます。

   この DB2START コマンドが完了すると、新しいサーバーは停止します。
2. DB2STOP コマンドを実行することによって、すべてのデータベース・パーティション上のデータベース・マネージャーを停止します。

   システム内のすべてのデータベース・パーティションを停止すると、ノード構成ファイルが更新されて新規データベース・パーティションが組み込まれます。DB2STOP が実行されるまでは、ノード構成ファイルがこの新しいサーバー情報によって更新されることはありません。このため、ADD DBPARTITIONNUM コマンド (DB2START コマンドに ADDNODE パラメーターが指定されたときに呼ばれる) は確実に正しいデータベース・パーティションで実行されます。このユーティリティーが終了すると、新しいサーバー・パーティションは停止します。
3. DB2START コマンドを実行してデータベース・マネージャーを開始します。

   これで、新規に追加されたデータベース・パーティションが残りのシステムと共に開始されます。

   システム内のすべてのデータベース・パーティションが実行中になると、データベースの作成またはドロップなどのシステム全般にわたる活動を行うことができます。

   注: 新しい `db2nodes.cfg` ファイルにアクセスするには、すべてのデータベース・パーティション・サーバーに関して DB2START コマンドを 2 回発行しなければならない場合があります。
4. オプション: 新規データベース・パーティションのすべてのデータベースのバックアップをとる。

5. オプション: 新規データベース・パーティションにデータを再配分する。

停止中のデータベース・システムへのデータベース・パーティションの追加 (Windows)

パーティション・データベース・システムの停止時に、新しいデータベース・パーティションを追加することができます。新規に追加されたデータベース・パーティションがすべてのデータベースに利用可能になるのは、データベース・マネージャーを再び始動したときです。

データベース・パーティションを作成する前に、新しいサーバーをインストールすることが必要です。

コマンド行を使用して、停止中のパーティション・データベース・サーバーにデータベース・パーティションを追加する方法は、以下のとおりです。

1. DB2STOP を出して、すべてのデータベース・パーティションを停止します。
2. ADD DBPARTITIONNUM コマンドを新規サーバーで実行します。

   データベース・パーティションは、システムにすでにある各データベースに対してローカルに作成されます。新規データベース・パーティションのためのデータベース・パラメーターは、デフォルト値に設定されます。各データベース・パーティションは、ユーザーがそこにデータを移すまでは空のままです。データベース構成パラメーター値を更新して、他のデータベース・パーティション上の値と一致させてください。
3. DB2START コマンドを実行して、データベース・システムを開始します。ノード構成ファイル (.cfg) は、すでに新規のサーバーのインストール時に更新されており、その新規のサーバーは組み込まれています。
4. 新しいデータベース・パーティションで構成ファイルを更新する方法は次のとおりです。
   a. 既存のデータベース・パーティションで、DB2START コマンドを実行します。

      DBPARTITIONNUM、ADDDBPARTITIONNUM、HOSTNAME、 PORT、および NETNAME パラメーターに加え、COMPUTER、USER、および PASSWORD パラメーターに新データベース・パーティション値を指定します。

      また、データベースに作成する必要のある任意の TEMPORARY 表スペース・コンテナー定義のためのソースを指定することもできます。表スペース情報を提供しないと、 TEMPORARY 表スペース・コンテナー定義は各データベースのカタログ・パーティションから検索されます。

      この DB2START コマンドが完了すると、新しいサーバーは停止します。
   b. DB2STOP コマンドを実行することによってデータベース・マネージャー全体を停止します。

      システム内のすべてのデータベース・パーティションを停止すると、ノード構成ファイルが更新されて新規データベース・パーティションが組み込まれます。 DB2STOP が実行されるまでは、ノード構成ファイルがこの新しいサ

ーバー情報によって更新されることはありません。このため、ADD DBPARTITIONNUM コマンド (DB2START コマンドに ADDDBPARTITIONNUM パラメーターが指定されたときに呼ばれる) は確実に正しいデータベース・パーティションで実行されます。このユーティリティーが終了すると、新しいサーバー・パーティションは停止します。

5. DB2START コマンドを実行してデータベース・マネージャーを開始します。

   これで、新規に追加されたデータベース・パーティションが残りのシステムと共に開始されます。

   システム内のすべてのデータベース・パーティションが実行中になると、データベースの作成またはドロップなどのシステム全般にわたる活動を行うことができます。

   注: 新しい db2nodes.cfg ファイルにアクセスするには、すべてのデータベース・パーティション・サーバーに関して DB2START コマンドを 2 回発行しなければならない場合があります。
6. オプション: 新規データベース・パーティションのすべてのデータベースのバックアップをとる。
7. オプション: 新規データベース・パーティションにデータを再配分する。

停止中のデータベース・システムへのデータベース・パーティションの追加 (UNIX)

パーティション・データベース・システムの停止時に、新しいデータベース・パーティションを追加することができます。新規に追加されたデータベース・パーティションがすべてのデータベースに利用可能になるのは、データベース・マネージャーを再び始動したときです。

サーバーが存在しない場合、以下のタスクも含め、新しいサーバーをインストールする必要があります。

- 実行可能モジュールをアクセス可能にする (共用ファイル・システム・マウントまたはローカル・コピーを使用する)
- オペレーティング・システム・ファイルを既存のプロセッサー上のシステム・ファイルと同期化する
- sqllib ディレクトリーがファイル共用システムとしてアクセス可能であることを確認する
- 関連するオペレーティング・システム・パラメーター (プロセスの最大数など) が適切な値に設定されていることを確認する

また、すべてのデータベース・パーティションにおいて、 etc ディレクトリーの hosts ファイルのネーム・サーバーにホスト名を登録することも必要です。

コマンド行を使用して、停止中のパーティション・データベース・サーバーにデータベース・パーティションを追加する方法は、以下のとおりです。

1. DB2STOP を出して、すべてのデータベース・パーティションを停止します。
2. ADD DBPARTITIONNUM コマンドを新規サーバーで実行します。

データベース・パーティションは、システムにすでにある各データベースに対してローカルに作成されます。新規データベース・パーティションのためのデータベース・パラメーターは、デフォルト値に設定されます。各データベース・パーティションは、ユーザーがそこにデータを移すまでは空のままです。データベース構成パラメーター値を更新して、他のデータベース・パーティション上の値と一致させてください。

3. DB2START コマンドを実行して、データベース・システムを開始します。ノード構成ファイル (.cfg) は、すでに新規のサーバーのインストール時に更新されており、その新規のサーバーは組み込まれています。
4. 新しいデータベース・パーティションで構成ファイルを更新する方法は次のとおりです。
   a. 既存のデータベース・パーティションで、DB2START コマンドを実行します。

      DBPARTITIONNUM、ADDDBPARTITIONNUM、HOSTNAME、 PORT、および NETNAME パラメーターに加え、COMPUTER、USER、および PASSWORD パラメーターに新データベース・パーティション値を指定します。

      また、データベースに作成する必要のある任意の TEMPORARY 表スペース・コンテナー定義のためのソースを指定することもできます。表スペース情報を提供しないと、 TEMPORARY 表スペース・コンテナー定義は各データベースのカタログ・パーティションから検索されます。

      この DB2START コマンドが完了すると、新しいサーバーは停止します。
   b. DB2STOP コマンドを実行することによってデータベース・マネージャー全体を停止します。

      システム内のすべてのデータベース・パーティションを停止すると、ノード構成ファイルが更新されて新規データベース・パーティションが組み込まれます。 DB2STOP が実行されるまでは、ノード構成ファイルがこの新しいサーバー情報によって更新されることはありません。このため、ADD DBPARTITIONNUM コマンド (DB2START コマンドに ADDDBPARTITIONNUM パラメーターが指定されたときに呼ばれる) は確実に正しいデータベース・パーティションで実行されます。このユーティリティーが終了すると、新しいサーバー・パーティションは停止します。
5. DB2START コマンドを実行してデータベース・マネージャーを開始します。

これで、新規に追加されたデータベース・パーティションが残りのシステムと共に開始されます。

システム内のすべてのデータベース・パーティションが実行中になると、データベースの作成またはドロップなどのシステム全般にわたる活動を行うことができます。

注: 新しい `db2nodes.cfg` ファイルにアクセスするには、すべてのデータベース・パーティション・サーバーに関して DB2START コマンドを 2 回発行しなければならない場合があります。

6. オプション: 新規データベース・パーティションのすべてのデータベースのバックアップをとる。
7. オプション: 新規データベース・パーティションにデータを再配分する。

また、次のように構成ファイルを手動で更新することもできます。

1. db2nodes.cfg ファイルを編集し、新規データベース・パーティションをそこに追加します。
2. 次のコマンドを出して、新しいデータベース・パーティションを開始します。
   ```
   DB2START DBPARTITIONNUM partitionnum
   ```
   新しいデータベース・パーティションに割り当てる番号を nodenum の値として指定します。
3. この新規サーバーを論理パーティション (すなわち、データベース・パーティション 0 ではない) にする場合は、 db2set コマンドを使用して、DBPARTITIONNUM レジストリー変数を更新します。追加するデータベース・パーティションの数を指定します。
4. ADD NODE コマンドを新規データベース・パーティションで実行します。

   このコマンドは、システムにすでにある各データベースに対してローカルにデータベース・パーティションを作成します。新規データベース・パーティションのためのデータベース・パラメーターは、デフォルト値に設定されます。各データベース・パーティションは、ユーザーがそこにデータを移すまでは空のままです。データベース構成パラメーター値を更新して、他のデータベース・パーティション上の値と一致させてください。
5. ADD DBPARTITIONNUM コマンドが完了したら、DB2START コマンドを出して、システム内の他のデータベース・パーティションを開始します。

   すべてのデータベース・パーティションが正常に開始するまでは、データベースの作成またはドロップなどのシステム全般にわたる活動を行わないでください。

データベース・パーティションを追加するときのエラー・リカバリー

データベース・マネージャーはシステム・バッファー・プールを作成して、すべてのバッファー・プールのページ・サイズにデフォルト自動サポートを提供するので、バッファー・プールが存在しないためにデータベース・パーティションの追加が失敗することはありません。しかし、これらのシステム・バッファー・プールのうち 1 つが使用されると、システム・バッファー・プールは非常に小さいため、パフォーマンスが大きな影響を受けることがあります。システム・バッファー・プールが使用される場合、管理通知ログにメッセージが書き込まれます。

システム・バッファー・プールは、次の状況で、データベース・パーティション追加のシナリオにおいて使用されます。

- デフォルト (4KB) と異なるページ・サイズで、 1 つ以上の SYSTEM TEMPORARY 表スペースを持つ分割したデータベース環境にデータベース・パーティションを追加する場合。データベース・パーティションの作成時には IBMDEFAULTBP バッファー・プールだけしか存在せず、このバッファー・プールのページ・サイズが 4KB です。

次の例を考慮してください。

1. 現在の複数パーティション・データベースにデータベース・パーティションを追加するため、db2start コマンドを使用します。

   ```
   DB2START DBPARTITIONNUM 2 ADD DBPARTITIONNUM HOSTNAME newhost PORT 2
   ```

2. 新しいデータベース・パーティション記述で、db2nodes.cfg ファイルを手動で更新した後に、ADD DBPARTITIONNUM コマンドを使用します。

これらの問題を予防する 1 つの方法としては、ADD NODE コマンドや db2start コマンドで WITHOUT TABLESPACES 節を指定する方法があります。コマンド実行後、CREATE BUFFERPOOL ステートメントを使用して、バッファー・プールを作成し、ALTER TABLESPACE ステートメントを使用して、バッファー・プールに SYSTEM TEMPORARY 表スペースを関連付ける必要があります。

- デフォルト・ページ・サイズ (4KB) と異なるページ・サイズで、 1 つ以上の表スペースを持つ既存のデータベース・パーティション・グループにデータベース・パーティションを追加する場合。これは、デフォルトでないページ・サイズ・バッファー・プールを、表スペースとしてアクティブになっていない、新規のデータベース・パーティション上に作成した場合に発生します。

  注: 以前のバージョンでは、このコマンドは DATABASE PARTITION GROUP キーワードではなく NODEGROUP キーワードを使用していました。

  次の例を考慮してください。

  – データベース・パーティション・グループにデータベース・パーティションを追加するために ALTER DATABASE PARTITION GROUP ステートメントを次のように使用します。

    ```
    DB2START
    CONNECT TO mpp1
    ALTER DATABASE PARTITION GROUP ng1 ADD NODE (2)
    ```

    この問題を予防する 1 つの方法としては、それぞれのページ・サイズのバッファー・プールを作成し、その後、ALTER DATABASE PARTITION GROUP ステートメントを発行する前にデータベースに再接続する方法があります。

    ```
    DB2START
    CONNECT TO mpp1
    CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
    CONNECT RESET
    CONNECT TO mpp1
    ALTER DATABASE PARTITION GROUP ng1 ADD NODE (2)
    ```

注: デフォルトのページ・サイズの表スペースを持つデータベース・パーティション・グループの場合は、メッセージ SQL1759W が返されます。

データベース・パーティションのドロップ

どのデータベースにも使用されていないデータベース・パーティションをドロップして、他の使用のためにコンピューターを解放することができます。

DROP DBPARTITIONNUM VERIFY コマンドまたは sqledrpn API を使用して、そのデータベース・パーティションが使用中ではないことを確認します。

- メッセージ SQL6034W (データベース・パーティションはほかのデータベースによって使用されていません - Database partition not used in any database) が出た場合は、データベース・パーティションをドロップできます。

- メッセージ SQL6035W (データベース・パーティションはデータベースによって使用中です - Database partition in use by database) が出た場合は、REDISTRIBUTE NODEGROUP コマンドを使用して、ドロップするデータベース・パーティションのデータを、データベース別名から他のデータベース・パーティションへ再配分します。

また、このデータベース・パーティションをコーディネーターとしていたすべてのトランザクションがすべて正常にコミットされている、またはロールバックされていることを確認してください。このためには、他のサーバーでクラッシュ・リカバリーを行う必要があることがあります。例えば、コーディネーター・パーティションをドロップすると、そのコーディネーター・パーティションがドロップされる前にトランザクションに関連する他のデータベース・パーティションが破損した場合、破損したデータベース・パーティションはどの未確定トランザクションの結果についてもコーディネーター・パーティションを照会することができなくなります。

コマンド行を使用してデータベース・パーティションをドロップするには、DROP NODENUM パラメーターを指定した DB2STOP コマンドを出して、データベース・パーティションをドロップします。このコマンドが正常に終了すると、システムは停止します。次に、DB2START コマンドを実行してデータベース・マネージャーを開始します。

インスタンス内のデータベース・パーティション・サーバーのリスト

Windows で db2nlist コマンドを使えば、インスタンスに関与するデータベース・パーティション・サーバーのリストを取得できます。

コマンドは、次のように使用します。

```
db2nlist
```

上記のようにコマンドを使用する場合、デフォルトのインスタンスは (DB2INSTANCE 環境変数によって設定される) 現行インスタンスです。特定のインスタンスを指定するには、次のコマンドを使ってインスタンスを指定できます。

```
db2nlist /i:instName
```

ここで、instName は、必要とする特定のインスタンス名です。

次のコマンドを使えば、各データベース・パーティション・サーバーの状況を要求することもできます。

```
db2nlist /s
```

各データベース・パーティション・サーバーの状況は、開始中、実行中、停止中、または停止済みのいずれかになります。

インスタンスへのデータベース・パーティション・サーバーの追加 (Windows)

Windows で db2ncrt コマンドを使用すると、インスタンスにデータベース・パーティション・サーバーを追加できます。

注: このインスタンスの中にデータベースがすでに含まれている場合は、 db2ncrt コマンドを使用しないでください。代わりに、db2start addnode コマンドを使用します。上記のコマンドにより、新しいデータベース・パーティション・サーバーにデータベースを正しく追加することができます。 db2nodes.cfg ファイルは**編集しないでください**。このファイルを変更すると、パーティション・データベース環境に不整合が生じる可能性があるからです。

このコマンドには、以下の必須パラメーターがあります。

```
db2ncrt /n:node_number
        /u:username,password
        /p:logical_port
```

/n: データベース・パーティション・サーバーを識別するための固有のデータベース・パーティション番号です。番号には、昇順で 1 から 999 までを指定できます。

/u: DB2 サービスのログオン・アカウント名とパスワードです。

/p:logical_port
論理ポートがゼロ (0) でない場合に、データベース・パーティション・サーバーに使用する論理ポート番号。このパラメーターを指定しないと、論理ポート番号には 0 が割り当てられます。

コンピューターに最初のデータベース・パーティションを作成するときには、論理ポート・パラメーターはオプショナルです。論理データベース・パーティションを作成する場合は、このパラメーターを指定し、未使用の論理ポート番号を選択しなければなりません。このパラメーターの使用に関して、いくつかの制約事項があります。

- どのコンピューターにも、論理ポート 0 のデータベース・パーティション・サーバーが 1 つずつ存在しなければなりません。
- %SystemRoot%¥system32¥drivers¥etc ディレクトリーのサービス・ファイル内で、FCM 通信のために予約されているポート範囲よりも大きいポート番号を選択することはできません。例えば、現行インスタンスのために 4 つのポートの範囲を予約する場合、最大ポート番号は 3 になるはずです (ポート 1、2、3。ポート 0 はデフォルトの論理データベース・パーティション)。ポート範囲は、 db2icrt を /r:base_port, end_port パラメーターと一緒に使用するときに定義します。

次のようなオプション・パラメーターもあります。

/g:network_name
データベース・パーティション・サーバーのネットワーク名を指定します。このパラメーターを指定しなかった場合、 DB2 はシステムで最初に検出した IP アドレスを使用します。

コンピューター上に複数の IP アドレスがあり、データベース・パーティション・サーバーに特定の IP アドレスを割り当てたい場合に、このパラメーターを使用します。ネットワーク名や IP アドレスを network_name パラメーターに入力することができます。

/h:host_name
TCP/IP ホスト名。ホスト名がローカル・ホスト名でない場合に FCM が内

部通信用に使用します。このパラメーターが必要になるのは、データベース・パーティション・サーバーをリモート・コンピューターに追加する場合です。

/i:instance_name
インスタンス名。デフォルトは、現行インスタンスです。

/m:computer_name
データベース・パーティションが存在する Windows ワークステーションのコンピューター名。デフォルトの名前は、ローカル・コンピューターのコンピューター名です。

/o:instance_owning_computer
インスタンス所有コンピューターであるコンピューターのコンピューター名。デフォルトはローカル・コンピューターです。このパラメーターは、インスタンス所有コンピューターでないコンピューターで db2ncrt コマンドを呼び出すときに必須です。

例えば、(複数の論理データベース・パーティションを実行するために) 新しいデータベース・パーティション・サーバーを、インスタンス所有コンピューター MYMACHIN 上のインスタンス TESTMPP へ追加して、この新しいデータベース・パーティションを論理ポート 1 を使用するデータベース・パーティション 2 として認識されるようにするには、次のように入力します。

```
db2ncrt /n:2 /p:1 /u:my_id,my_pword /i:TESTMPP
   /M:TEST /o:MYMACHIN
```

「パーティションの追加」ウィザードを使用してインスタンスにデータベース・パーティションを追加する

パーティションの追加ウィザードを使用して、パーティションを作成し、それを 1 つまたは複数のデータベース・パーティション・グループに追加します。 まずインスタンスに新しいパーティションを追加し、そのパーティションを 1 つまたは複数のデータベース・パーティション・グループに割り当ててから、さらに詳細な選択を行います。

データベース・パーティション・グループを処理するには、SYSADM または DBADM 権限が必要です。

1. パーティションの追加ウィザードをオープンします。
   a. コントロール・センターから、操作したいインスタンス・オブジェクトが見つかるまでオブジェクト・ツリーを展開します。 インスタンス・オブジェクトを右クリックし、ポップアップ・メニューで「パーティションの追加」をクリックします。「パーティションの追加」ランチパッドがオープンします。
   b. 「**パーティションの追加**」ボタンをクリックします。 パーティションの追加ウィザードがオープンします。
2. ウィザードの該当するページをそれぞれ完了させます。詳しくは、先頭ページにあるウィザードの概要についてのリンクをクリックしてください。パーティションを追加するための十分な情報をウィザードに指定すると、「**完了**」ボタンが使用できるようになります。

データベース・パーティションの変更 (Windows)

Windows では、データベース・パーティションを変更するには db2nchg コマンドを使用します。

- データベース・パーティションをあるコンピューターから別のコンピューターに移動する。
- コンピューターの TCP/IP ホスト名を変更する。

  複数のネットワーク・アダプターを使用する予定の場合には、このコマンドを使用して、db2nodes.cfg ファイルの ″netname″ フィールドに TCP/IP アドレスを指定しなければなりません。
- 異なる論理ポート番号を使用する。
- データベース・パーティション・サーバーに異なる名前を使用する。

このコマンドには、以下の必須パラメーターがあります。

```
db2nchg /n:node_number
```

パラメーター /n: は、構成を変更するデータベース・パーティション・サーバーの番号です。このパラメーターは必須です。

オプション・パラメーターには、以下のものがあります。

/i:instance_name
このデータベース・パーティション・サーバーが参加しているインスタンスを指定します。このパラメーターを指定しなかった場合、デフォルトである現行インスタンスが使用されます。

/u:username,password
DB2 データベース・サービスのログオン・アカウント名とパスワードを変更します。このパラメーターを指定しなかった場合、ログオン・アカウント名とパスワードは変わりません。

/p:logical_port
データベース・パーティション・サーバーの論理ポートを変更します。データベース・パーティション・サーバーを異なるコンピューターへ移動させる場合、このパラメーターの指定は必須です。このパラメーターを指定しなかった場合、論理ポート番号は変わりません。

/h:host_name
FCM が内部通信のために使用する TCP/IP ホスト名を変更します。このパラメーターを指定しなかった場合、ホスト名は変わりません。

/m:computer_name
データベース・パーティション・サーバーを別のコンピューターへ移動させます。データベース・パーティション・サーバーを移動できるのは、インスタンス内にデータベースが 1 つもない場合だけです。

/g:network_name
データベース・パーティション・サーバーのネットワーク名を変更します。

コンピューター上に複数の IP アドレスがあり、データベース・パーティション・サーバーに特定の IP アドレスを割り当てる場合に、このパラメーターを使用します。ネットワーク名や IP アドレスを network_name に入力することができます。

例えば、データベース・パーティション 2 に割り当てられている論理ポート (インスタンス TESTMPP に参加している) が論理ポート 3 を使用するように変更する場合は、次のコマンドを入力します。

```
db2nchg /n:2 /i:TESTMPP /p:3
```

DB2 データベース・マネージャーには、リモート・コンピューター上にあるインスタンス・レベルの DB2 データベース・システムのレジストリー変数にアクセスする機能があります。現在、DB2 データベース・システムのレジストリー変数は、コンピューターまたはグローバル・レベル、インスタンス・レベル、およびデータベース・パーティション・レベルの 3 つの異なるレベルに保管されています。インスタンス・レベル (データベース・パーティション・レベルも含む) に保管されているレジストリー変数は、DB2REMOTEPREG を使用して別のコンピューターにリダイレクトすることができます。 DB2REMOTEPREG が設定されると、DB2 データベース・マネージャーは、 DB2REMOTEPREG が示すコンピューターから DB2 データベース・システムのレジストリー変数にアクセスします。 db2set コマンドは、次のようになります。

```
db2set DB2REMOTEPREG=<remote workstation>
```

ここで remote workstation> は、リモート・ワークステーション名です。

注:

- すべての DB2 データベースのインスタンス・プロファイルとインスタンス・リストは、指定されたリモート・コンピューター名で位置指定されるため、このオプションの設定には注意が必要です。
- ご使用の環境にドメインからのユーザーが含まれる場合、DB2 インスタンス・サービスに関連したログオン・アカウントがドメイン・アカウントであることを確認してください。これにより、DB2 インスタンスはドメイン・レベルでグループを列挙するための適切な特権を持つことになります。

このフィーチャーは、レジストリーが含まれる同じコンピューター上のリモート LAN 装置を指すために、 DBINSTPROF の設定と組み合わせて使用することができます。

データベース・パーティション内の SMS 表スペースへのコンテナーの追加

コンテナーを追加できるのは、現在コンテナーがないデータベース・パーティション上の SMS 表スペースに対してのみです。

コマンド行を使用して、SMS 表スペースにコンテナーを追加するには、以下のように入力します。

```
ALTER TABLESPACE <name>
   ADD ('<path>')
   ON DBPARTITIONNUM (<database partition_number>)
```

番号で指定されたデータベース・パーティション、およびパーティションの範囲内のすべてのデータベース・パーティションは、表スペースが定義されているデータベース・パーティション・グループに存在していなくてはなりません。データベースの partition_number は、ステートメントのただ 1 つの *db-partitions-clause* で、明示的にのみ、または範囲で表される場合もあります。

以下の例では、UNIX ベースのオペレーティング・システム上で、表スペース「plans」が使用しているデータベース・パーティション・グループの 3 番データベース・パーティションにどのように新規のコンテナーを追加するかを示しています。

```
ALTER TABLESPACE plans
   ADD ('/dev/rhdisk0')
   ON DBPARTITIONNUM (3)
```

インスタンスからのデータベース・パーティションのドロップ (Windows)

Windowsで db2ndrop コマンドを使うと、データベースのないインスタンスからデータベース・パーティション・サーバーをドロップできます。 データベース・パーティション・サーバーをドロップする場合、そのデータベース・パーティション番号は新しいデータベース・パーティション・サーバーに再使用することができます。

インスタンスからデータベース・パーティション・サーバーをドロップする場合は、注意してください。インスタンス所有データベース・パーティション・サーバーのゼロ (0) をインスタンスからドロップすると、インスタンスは使用できなくなります。インスタンスをドロップする場合は、db2idrop コマンドを使用します。

注: このインスタンスの中にデータベースが含まれている場合は、 db2ndrop コマンドを使用しないでください。代わりに、db2stop drop nodenum コマンドを使用します。上記のコマンドにより、新しいデータベース・パーティションからデータベースを正しく除去することができます。 db2nodes.cfg ファイルは**編集しないでください**。このファイルを変更すると、パーティション・データベース環境に不整合が生じる可能性があるからです。

複数の論理データベース・パーティションが実行されているコンピューターから、論理ポート 0 に割り当てられているデータベース・パーティションをドロップする場合は、 0 以外の論理ポートに割り当てられているデータベース・パーティションをすべてドロップしてからでないと、論理ポート 0 に割り当てられているデータベース・パーティションをドロップできません。どのデータベース・パーティション・サーバーにも、論理ポート 0 に割り当てられているデータベース・パーティションが 1 つずつなければなりません。

このコマンドには、以下のパラメーターがあります。

```
db2ndrop /n:node_number /i:instance_name
```

/n: データベース・パーティション・サーバーを識別するための固有のデータベース・パーティション番号です。これは必須パラメーターです。番号には、昇順でゼロ (0) から 999 までを指定できます。データベース・パーティションのゼロ (0) はインスタンス所有コンピューターを表すことに注意してください。

/i:instance_name

インスタンス名です。これはオプション・パラメーターです。このパラメーターを指定しない場合、デフォルトのインスタンスは (DB2INSTANCE レジストリー変数によって設定される) 現行インスタンスです。

「パーティションのドロップ」ランチパッドを使用してインスタンスからデータベース・パーティションをドロップする

「パーティションのドロップ」ランチパッドを使用すると、データベース・パーティションをデータベース・パーティション・グループからドロップする、データベース・パーティション・グループ内でデータを再配分する、そして、インスタンスからパーティションをドロップするのに必要なタスクを順に実施することができます。

このタスクについて

注: データベース・パーティション・グループからデータベース・パーティションをドロップする場合、データベース・パーティションは即時にはドロップされません。その代わりに、ドロップするデータベース・パーティションにフラグが立てられ、データベース・パーティション・グループ内でデータを再配分するときに、そこからデータが移動されるようになります。

データベース・パーティション・グループ内のデータを再配分する前と後に、インスタンス内のすべてのデータベースをバックアップすることをお勧めします。 データベースをバックアップしないと、データベースが破壊され、リカバリーできない場合があります。

手順

「パーティションのドロップ」ランチパッドを使用してパーティションをドロップするには、次のようにしてください。

1. オプション: 「バックアップ」 ウィザードを使用してデータをバックアップします。
2. 「パーティションのドロップ」ランチパッドをオープンします。
   a. 次のようにして、「パーティション」ウィンドウをオープンします。コントロール・センターから、パーティションを表示するインスタンスが表示されるまでオブジェクト・ツリーを展開します。インスタンスを右マウス・ボタンでクリックし、ポップアップ・メニューから「**データベース・パーティション・サーバーのオープン**」を選択します。選択されたインスタンスについての「パーティション」ウィンドウがオープンします。
   b. ドロップするパーティションを選択します。
   c. 選択したパーティションで右クリックし、ポップアップ・メニューで「**ドロップ**」をクリックします。 「パーティションのドロップ」ランチパッドがオープンします。
3. データベース・パーティション・グループからデータベース・パーティションをドロップします。
   a. データベース・パーティション・グループからドロップするデータベース・パーティションを確認します。

注:

- インスタンスからパーティションをドロップする前に、データベース・パーティション・グループからデータベース・パーティションをドロップする必要があります。
- この操作で、データベース・パーティションは即時にはドロップされません。その代わりに、ドロップするデータベース・パーティションにフラグが立てられ、データベース・パーティション・グループ内でデータを再配分するときに、そこからデータを移動できるようにしています。

4. 。データベース・パーティション・グループ内のデータを再配分します。
5. インスタンスからパーティションをドロップします。
   a. 「インスタンスからのパーティションのドロップの確認」ウィンドウをオープンします。
      - 上で説明されたとおりに「パーティション」ウィンドウをオープンします。
      - ドロップするパーティションを選択します。
      - 選択したパーティションで右クリックし、ポップアップ・メニューで「**ドロップ**」をクリックします。 「パーティションのドロップ」ランチパッドがオープンします。
      - 「**インスタンスからのパーティションのドロップ**」ボタンをクリックします。 「インスタンスからのパーティションのドロップの確認」ウィンドウがオープンします。
   b. 「**ドロップ**」列で、選択したインスタンスのパーティションをドロップすることを確認します。
   c. 「**OK**」をクリックすると、パーティションをドロップするスケジュールを設定できるウィンドウがオープンします。
6. オプション: 「バックアップ」 ウィザードを使用してデータをバックアップします。

シナリオ: データベース内のデータのパーティション化

このシナリオでは、データベースに新規のデータベース・パーティションを追加し、データベース・パーティション間にデータを再配分する方法を示します。 REDISTRIBUTE DATABASE PARTITION GROUP コマンドは、データベース・パーティション・グループ内の別々の表セットのデータを再配分する方法の例の一部として示されています。

シナリオ:

データベース DBPG1 にはデータベース・パーティションが 2 つあり、(0, 1) と指定されています。データベース・パーティション・グループ定義は (0, 1) です。

データベース・パーティション・グループ DBPG_1 には、以下の表スペースが定義されています。

- 表スペース TS1 - この表スペースには、T1 と T2 の 2 つの表があります。

- 表スペース TS2 - この表スペースには、T3、T4、T5 の 3 つの表が定義されています。

DBPG1 内のデータベース・パーティション間のデータの配分:

データベースに 3 つの新規データベース・パーティションを追加するには、以下のコマンドを発行します。

```
DB2START DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME <HOSTNAME3>
 PORT <PORT3>;

DB2START DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME <HOSTNAME4>
 PORT <PORT4>;

DB2START DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME <HOSTNAME5>
 PORT <PORT5>;

DB2STOP;

DB2START;
```

次の再配分コマンドは、DBPG_1 定義を (0, 1) から (0, 1, 3, 4, 5) に変更し、データの再配分も行います。

```
DB2 REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 UNIFORM ADD DBPARTITIONNUM (3 TO 5) STOP AT 2006-03-10-07.00.00.000000;
```

コマンドは、表 T1、T2、T3 について正常に実行された後、STOP AT が指定されているので停止したとします。

データベース・パーティション・グループのデータの再配分を打ち切って、表 T1、T2、T3 に加えられた変更を元に戻すには、次のコマンドを発行します。

```
DB2 REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD
 RECOVERABLE ABORT;
```

データ再配分中にエラーや中断が発生した場合で、再配分操作を続行しない場合は、データの再配分を打ち切る必要があります。このシナリオでは、このコマンドが正常に実行され、表 T1 および T2 がそれぞれ元の状態に戻ったとします。

DATA BUFFER として 4K ページを 5000 個使用して T5 と T4 だけを再配分するには、次のようにします。

```
DB2 REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 UNIFORM ADD DBPARTITIONNUM (3 TO 5) TABLE (T5, T4) ONLY  DATA BUFFER 5000;
```

このコマンドの実行が正常に終了すると、表 T4 および T5 内のデータは正常に再配分されています。

T1、T2、T3 のデータの再配分を、指定した順序で完了するには、次のコマンドを発行します。

```
DB2 REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 CONTINUE TABLE (T1) FIRST;
```

TABLE (T1) FIRST を指定すると、データベース・マネージャーは最初に表 T1 を処理するので、表 T1 は他の表よりも先にオンライン (読み取り専用) 状態に戻ることができます。他のすべての表は、データベース・マネージャーが決定する順序で処理されます。

注:

- ADD DBPARTITIONNUM オプションと DROP DBPARTITIONNUM オプションは、指定する必要はありません。その代わりに、REDISTRIBUTE DATABASE PARTITION GROUP コマンドが実行される前に、ALTER DATABASE PARTITION GROUP ステートメントを使用してデータベース・パーティションを追加またはドロップする必要があります。その場合、REDISTRIBUTE DATABASE PARTITION GROUP コマンドはパーティションの追加またはドロップを行わず、指定されたオプションに従ってデータの再配分のみ行います。
- ユーザーは、REDISTRIBUTE DATABASE PARTITION GROUP コマンドを実行する前に、データベースのオフライン・データベース・バックアップを取ることが強く推奨されています。このアクションは上記の例では示されていません。
- REDISTRIBUTE DATABASE PARTITION GROUP コマンドは、ロールフォワード・リカバリー可能ではありません。この問題の全解説については、『REDISTRIBUTE DATABASE PARTITION GROUP コマンド』を参照してください。
- REDISTRIBUTE DATABASE PARTITION GROUP コマンドが完了した後、そのコマンドがアクセスしたすべての表スペースは BACKUP PENDING 状態のままになります。表スペースに含まれる表が書き込みアクティビティー用にアクセス可能になる前に、そのような表スペースのバックアップを取る必要があります。

これらの手順は、データベース・パーティション間でデータを再配分する場合の再配分コマンドのバリエーションの使用方法を例で示したものです。

パーティション・データベース環境でのコマンドの実行

パーティション・データベース環境では、インスタンスにあるコンピューターで、あるいはデータベース・パーティション・サーバー (ノード) で実行するコマンドを発行する場合があります。このような場合には、rah コマンドまたは db2_all コマンドを使用することができます。 rah コマンドを使用すれば、インスタンス内のすべてのコンピューターで実行するコマンドを発行できます。

インスタンス内のデータベース・パーティション・サーバーでコマンドを実行する場合は、 db2_all コマンドを実行します。このセクションでは、これらのコマンドについての概要を説明します。以下の情報は、パーティション・データベース環境だけに適用されます。

注:

1. Linux および UNIX プラットフォームでは、ログイン・シェルを Korn シェルまたは他のシェルにすることができます。ただし、特殊文字を含むコマンドをシェルが処理する方法は、シェルによってさまざまに異なります。
2. また、Linux および UNIX プラットフォームでは、rah は DB2RSHCMD レジストリー変数によって指定されるリモート・シェル・プログラムを使用します。ssh (セキュリティーを追加する場合) または rsh (HP-UX では remsh) の 2 つのリモート・シェル・プログラムの中から選択できます。 ssh リモート・シェル・プログラムは、UNIX オペレーティング・システム環境で平文のパスワードが伝送

されるのを回避するために使用されます。このレジストリー変数が設定されない場合、rsh (HP-UX では remsh) が使用されます。

3. Windows で rah コマンドまたは db2_all コマンドを実行するには、管理者グループのメンバーになっているユーザー・アカウントでログオンしなければなりません。

コマンドの有効範囲を調べるには、「コマンド・リファレンス」を参照してください。コマンドが 1 つのデータベース・パーティション・サーバーで実行されるか、それともすべてのサーバーで実行されるかが示されます。 1 つのデータベース・パーティション・サーバーで実行されるコマンドをすべてのサーバーで実行させるには、 db2_all を使用してください。 db2trc コマンドは例外で、 1 つのコンピューター上のすべての論理ノード (データベース・パーティション・サーバー) で実行します。すべてのコンピューター上のすべての論理ノードで db2trc を実行する場合は、 rah を使用してください。

rah および db2_all コマンドの概要

コマンド群の実行は、1 つのデータベース・パーティション・サーバーから別のデータベース・パーティション・サーバーへと順次行うことも、並列に行うこともできます。 Linux および UNIX プラットフォームでコマンドを並列に実行する場合は、出力をバッファーに送ってまとめて表示する (デフォルトの動作) か、コマンドが発行されるコンピューターに出力を表示することができます。

Windows では、コマンドを並列に実行すると、出力はそのコマンドが発行されるコンピューターに表示されます。

rah コマンドを使用するには、次のように入力してください。

```
rah command
```

db2_all コマンドを使用するには、次のように入力してください。

```
db2_all command
```

rah 構文に関するヘルプを表示するには、次のように入力してください。

```
rah "?"
```

対話式プロンプトで入力できるほとんどのコマンドを使用できます。例えば、複数のコマンドを順番に実行することも可能です。 Linux および UNIX プラットフォームでは、セミコロン (;) を使って複数のコマンドを分離します。 Windows では、アンパーサンド (&) を使って複数のコマンドを分離します。最後のコマンドの後には、区切り文字を使用しないでください。

以下の例は、db2_all コマンドを使用して、ノード構成ファイルで指定したすべてのデータベース・パーティションでデータベース構成を変更する方法を示したものです。 ; 文字が二重引用符の内側にあるので、要求は同時に実行されます。

```
db2_all ";DB2 GET DB CFG FOR sample USING LOGFILSIZ 100"
```

rah および db2_all コマンドの指定

rah コマンドは、コマンド行からパラメーターとして指定できます。パラメーターを何も指定しない場合は、プロンプトへの応答として指定できます。

コマンドに次のような特殊文字が入っている場合は、プロンプト方式を使用する必要があります。

```
| & ; < > ( ) { } [ ] unsubstituted $
```

コマンド行でパラメーターとしてコマンドを指定するときに、上にリストしたような特殊文字のいずれかが含まれている場合は、二重引用符で囲む必要があります。

注: Linux および UNIX プラットフォームでは、プロンプトで入力した場合と同様に、コマンドがコマンド履歴に追加されます。

コマンドの中の特殊文字はすべて、正常に入力することができます (¥ 以外は引用符で囲まずに)。コマンドに ¥ を入れる必要があるときは、円記号を 2 つ (¥¥) 入力しなければなりません。

注: Linux および UNIX プラットフォームでは、Korn シェルを使用していない場合は、コマンドの中の特殊文字はすべて正常に入力することができます (″、¥、置換不能文字 $、および単一引用符 (') 以外は、引用符に入れずに)。コマンドにこれらの文字のいずれかを入れる必要があるときは、円記号を 3 つ (¥¥¥) 前に置かなければなりません。例えば、コマンドに ¥ を入れる必要があるときは、円記号を 4 つ (¥¥¥¥) 入力しなければなりません。

コマンドに二重引用符 (″) を入れる必要があるときは、円記号を 3 つ前に付けて、例えば、¥¥¥″ のように入力しなければなりません。

注:

1. Linux および UNIX プラットフォームでは、単一引用符付きストリングの内側に単一引用符を入れる何らかの方法をコマンド・シェルが提供しないかぎり、単一引用符 (') をコマンドに含めることはできません。
2. Windows では、単一引用符付きストリングの内側に単一引用符を入れる何らかの方法をコマンド・ウィンドウが提供しないかぎり、コマンドに単一引用符 (') を含めることはできません。

stdin からバックグラウンドで読み取りを行うロジックを含む Korn シェルのシェル・スクリプトを実行する場合、 stdin をソースに明示的にリダイレクトする必要があります。そうすれば、プロセスは端末上で停止されることなく読み取りを行うことができます (SIGTTIN メッセージ)。 stdin をリダイレクトするには、指定されている入力がなければ次の形式のスクリプトを実行します。

```
shell_script </dev/null &
```

同様に、db2_all をバックグラウンドで実行する際には、常に </dev/null を指定する必要があります。例:

```
db2_all ";run_this_command" </dev/null &
```

これを行うことにより、端末上で停止させなくても stdin をリダイレクトすることができます。

別の方法として、リモート・コマンドからの出力が必要ない場合には、次のように db2_all 接頭部でデーモン化オプションを使用することもできます。

```
db2_all ";daemonize_this_command" &
```

コマンドの並列実行 (Linux、UNIX)

デフォルトでは、コマンドはそれぞれのコンピューターで順次的に実行されますが、特定の接頭部シーケンスをコマンドの前につけることによって、バックグラウンド rshell を使って、コマンドを並列に実行するよう指定できます。rshell がバックグラウンドで実行されている場合、それぞれのコマンドは、リモート・コンピューターにあるバッファー・ファイルに出力を入れます。

注: この節の情報は Linux および UNIX プラットフォームだけに適用されます。

このプロセスでは、出力は次のように 2 つに分けて取り出されます。

1. リモート・コマンドが完了した後。
2. 何らかのプロセスがまだ実行されている場合は、あとで実行される可能性のある rshell が終了した後。

デフォルトでは、バッファー・ファイルの名前は /tmp/$USER/rahout ですが、環境変数 $RAHBUFDIR/$RAHBUFNAME によって名前を指定することができます。

複数のコマンドを並行して実行するよう指定した場合、デフォルトでこのスクリプトは、すべてのホストにコマンドを送信する前に、 $RAHBUFDIR と $RAHBUFNAME で指定されるバッファー・ファイルが使用できるかどうかチェックします。存在していない場合、$RAHBUFDIR を作成します。この動作を省略するには、環境変数 RAHCHECKBUF=no をエクスポートします。ディレクトリーが存在していて、使用可能であることがわかっている場合は、このようにすると時間を節約できます。

rah を使用して複数のコンピューターでコマンドを同時に実行する前に、以下を行ってください。

- それぞれのコンピューターごとに、使用しているユーザー ID 用のディレクトリー /tmp/$USER が存在することを確認する。このディレクトリーがまだ存在していない場合は、それを作成するために以下を実行してください。

  ```
  rah ")mkdir /tmp/$USER"
  ```

- 以下の行を .kshrc (Korn シェル構文の場合) または .profile に追加して、現行のセッションにそれを入力する。

  ```
  export RAHCHECKBUF=no
  ```

- リモート・コマンドを実行するそれぞれのコンピューター ID の .rhosts ファイル内に、rah を実行する ID に対応する項目があることを確認する。および rah を実行する ID の .rhosts ファイル内に、リモート・コマンドを実行するそれぞれのコンピューター ID に対応する項目があることを確認する。

rah コマンドの拡張によるツリー・ロジックの使用 (AIX および Solaris)

パフォーマンスを向上させるために、 rah は大規模なシステムで tree_logic を使うように拡張されています。つまり、rah はリストに含まれるノード数を検査し、その数がしきい値を超過するなら、リストのサブセットを作成して、それ自体の再帰的呼び出しをそれぞれのノードに送信します。

それぞれのノードでは、再帰的に呼び出された rah は前述の同じ論理に従います。これは、リストが十分に小さくなり、「リスト上のすべてのノードにコマンドを送信する」という標準的な論理 (ここでは ″ツリーのリーフ″ という論理) に従えるようになるまで続きます。このときのしきい値は、環境変数 RAHTREETHRESH で指定できます。これを指定しないと、デフォルトの 15 になります。

物理ノードに対して複数の論理ノードが存在するシステムの場合は、 db2_all は再帰的な呼び出しをそれぞれの物理ノードに送信してから、その同じ物理ノード上の他の論理ノードに rsh するので、物理ノード間のトラフィックも少なくなります。(この点は、db2_all だけに当てはまるもので rah には当てはまりません。 rah は常にそれぞれの物理ノードだけに送信します。)

rah および db2_all コマンド

このトピックでは、rah および db2_all コマンドについて説明します。

コマンド
説明

rah すべてのコンピューターでコマンドを実行します。

db2_all
指定したすべてのデータベース・パーティション・サーバーでコマンドを実行します。

db2_kill
複数のデータベース・パーティション・サーバーで実行されているすべてのプロセスを突然停止し、すべてのデータベース・パーティション・サーバーのすべてのリソースを終結処理します。このコマンドは、データベースを不整合にします。このコマンドは IBM サービスからの指示による以外は発行しないでください。

db2_call_stack
Linux および UNIX プラットフォームでは、すべてのデータベース・パーティション・サーバーで実行されているすべてのプロセスが、呼び出しトレースバックを syslog に書き出すようにします。

Linux および UNIX プラットフォームでは、これらのコマンドは、次のような特定の暗黙的な設定で rah を実行します。

- すべてのコンピューターで並列に実行する。
- コマンド出力を /tmp/$USER/db2_kill、 /tmp/$USER/db2_call_stack にそれぞれバッファーする。

コマンド db2_call_stack は、Windows では使用できません。代わりに db2pd -stack コマンドを使用してください。

rah コマンドの接頭部シーケンス

接頭部シーケンスは、1 桁以上の特殊文字です。

コマンドの文字の直前に、ブランクをあけずに 1 つまたは複数の接頭部シーケンスを入力してください。シーケンスを 2 つ以上指定する場合は、任意の順序でタイプすることができますが、複数文字のシーケンスの中にある文字は、順番に入力する

必要があります。接頭部シーケンスを入力するときは、次の例に示すように、接頭部シーケンスも含めてコマンド全体を二重引用符で囲んでください。

- Linux および UNIX プラットフォームでは、以下のようにします。

```
rah "};ps -F pid,ppid,etime,args -u $USER"
```

- Windows では、以下のようにします。

```
rah "||db2 get db cfg for sample"
```

接頭部シーケンスは次のとおりです。

シーケンス
目的

| バックグラウンドでコマンドを順に実行します。

|& バックグラウンドで順番にコマンドを実行し、さらにすべてのリモート・コマンドが完了したあとで、まだいくつかのプロセスが実行中であっても、コマンドを終了します。例えば、子プロセス (Linux および UNIX プラットフォームの場合) またはバックグラウンド・プロセス (Windows の場合) がまだ実行中であれば、もっと後になる可能性があります。このケースでは、コマンドは、別個のバックグラウンド・プロセスを開始して、コマンド終了後に生成されたリモート出力を検索し、もとのコンピューターに書き戻します。

注: Linux および UNIX プラットフォームで & を指定すると、 rsh コマンドがさらに必要になるので、パフォーマンスが低下します。

|| バックグラウンドでコマンドを並列に実行します。

||& バックグラウンドで並列にコマンドを実行し、さらにすべてのリモート・コマンドが完了した後で、上記の |& のケースで述べたようにしてコマンドを終了します。

注: Linux および UNIX プラットフォームで & を指定すると、 rsh コマンドがさらに必要になるので、パフォーマンスが低下します。

; 上記の ||& と同じ。これは、省略形です。

注: Linux および UNIX プラットフォームで ; を指定すると、 rsh コマンドがさらに必要になるので、 || に比べてパフォーマンスが低下します。

] コマンドを実行する前に、ユーザーのプロファイルのドット実行 (dot-execution(..)) を頭に付けます。

注: Linux および UNIX プラットフォームに限り使用できます。

} コマンドを実行する前に、 $RAHENV (多分 .kshrc) で指名されたファイルのドット実行を頭に付けます。

注: Linux および UNIX プラットフォームに限り使用できます。

]} コマンドを実行する前に、ユーザーのプロファイルのドット実行を頭に付け、その後で、 $RAHENV (多分 .kshrc) で指名されたファイルを実行します。

注: Linux および UNIX プラットフォームに限り使用できます。

) ユーザーのプロファイルおよび $RAHENV で指名されたファイルの実行を抑止します。

注: Linux および UNIX プラットフォームに限り使用できます。

' コマンドの起動をコンピューターにエコーします。

< このコンピューター以外のすべてのコンピューターに送信します。

<<-nnn<

nnn 以外のすべてのデータベース・パーティション・サーバー (ノード番号 *nnn* を除いて db2nodes.cfg にあるすべてのデータベース・パーティション・サーバー。この表の最後の接頭部文字の後の最初の段落を参照してください) に送信します。

<<+nnn<

データベース・パーティション・サーバー *nnn* (データベース・パーティション番号が *nnn* の db2nodes.cfg にあるデータベース・パーティション・サーバー。この表の最後の接頭部文字の後の最初の段落を参照してください) だけに送信します。

(ブランク文字)

リモート・コマンドを、stdin、 stdout および stderr をすべてクローズしてバックグラウンドで実行します。このオプションは、バックグラウンドでコマンドを実行するときだけ、つまり ¥ または ; も含んでいる接頭部シーケンスの中でだけ有効です。このようにすると、コマンドは非常に早く完了することができます (リモート・コマンドが開始されるとすぐに)。この接頭部シーケンスを rah コマンド行で指定する場合は、コマンドを単一引用符で囲むか、またはコマンドを二重引用符で囲んでから接頭部文字の前に ¥ を置きます。例:

```
rah '; mydaemon'
```

または

```
rah ";¥ mydaemon"
```

rah コマンドは、バックグラウンド・プロセスとして実行されるとき、出力が戻されるのを待ちません。

> > のオカレンスをコンピューター名と置換します。

" () のオカレンスをコンピューター索引と置換し、 ## のオカレンスをデータベース・パーティション番号と置換します。

注:

1. コンピューター索引とは、データベース・システムのコンピューターに関連した番号のことです。複数の論理パーティションを実行していない場合は、コンピューターのコンピューター索引は、ノード構成ファイル内のそのコンピューターのデータベース・パーティション番号に対応します。複数の論理パーティション・データベース環境のコンピューターに関するコンピューター索引を取得するには、複数の論理パーティションを実行しているコンピューターの重複項目をカウントしないでください。例えば MACH1 と MACH2 の両方とも 2 つの論理パーティション

を実行している場合は、ノード構成ファイル内の MACH3 のデータベース・パーティション番号は 5 になります。しかし、MACH3 のコンピューター索引は 3 になります。

Windows の場合、ノード構成ファイルを編集しないでください。コンピューター索引を取得するには、 db2nlist コマンドを使用してください。

2. " が指定されている場合は、重複はコンピューターのリストから除去されません。

<<-nnn< と <<+nnn< 接頭部シーケンスを使用しているとき、 *nnn* は 1、2 または 3 桁の任意のデータベース・パーティション番号にすることができますが、これは、 db2nodes.cfg ファイルの *nodenum* 値と一致している必要があります。

注: 接頭部シーケンスは、コマンドの一部と見なされます。接頭部シーケンスをコマンドの一部として指定するときは、接頭部シーケンスも含めてコマンド全体を二重引用符で囲んでください。

rah コマンドの制御

このトピックでは、rah コマンドを制御するための環境変数をリストしています。

表 13. rah コマンドを制御する環境変数

名前	意味	デフォルト
$RAHBUFDIR **注:** Linux および UNIX プラットフォームに限り使用できます。	バッファーのディレクトリー	/tmp/$USER
$RAHBUFNAME **注:** Linux および UNIX プラットフォームに限り使用できます。	バッファーのファイル名	rahout
$RAHOSTFILE (Linux および UNIX プラットフォームの場合)、RAHOSTFILE (Windows の場合)	ホストのリストが入っているファイル	db2nodes.cfg
$RAHOSTLIST (Linux および UNIX プラットフォームの場合)、RAHOSTLIST (Windows の場合)	ストリングとしてのホストのリスト	$RAHOSTFILE から抽出
$RAHCHECKBUF **注:** Linux および UNIX プラットフォームに限り使用できます。	″no″ に設定されていると、チェックはバイパスします。	設定されない

表 13. rah コマンドを制御する環境変数 (続き)

名前	意味	デフォルト
$RAHSLEEPTIME (Linux および UNIX プラットフォームの場合)、RAHSLEEPTIME (Windows の場合)	並列に実行されるコマンドからの最初の出力をこのスクリプトが待つ時間 (秒数)。	db2_kill の場合は 86400 秒、他のすべての場合は 200 秒。
$RAHWAITTIME (Linux および UNIX プラットフォームの場合)、RAHWAITTIME (Windows の場合)	Windows の場合、リモート・ジョブがまだ実行されていることを連続チェックするインターバルの秒数 Linux および UNIX プラットフォームの場合、リモート・ジョブがまだ実行されていることを連続チェックしてから、`rah: waiting for pid> ...` メッセージまでのインターバルの秒数 プラットフォームに関係なく、正の整数を指定します。メッセージを抑止するには先行ゼロを値の前に付けます。例えば、RAHWAITTIME=045 を指定してエクスポートします。 rah は、ジョブの完了を検知するためにこれらのチェックには依存しないので、低い値を指定する必要はありません。	45 秒
$RAHENV **注:** Linux および UNIX プラットフォームに限り使用できます。	$RAHDOTFILES=E または K または PE または B の場合に実行されるファイル名を指定します。	$ENV
$RAHUSER (Linux および UNIX プラットフォームの場合)、RAHUSER (Windows の場合)	Linux および UNIX プラットフォームの場合、リモート・コマンドがその下で実行されるユーザー ID Windows の場合、 DB2 リモート・コマンド・サービスに関連したログオン・アカウント	$USER

注: Linux および UNIX プラットフォームでは、リモート・シェルで設定される値 (存在する場合) ではなく、 rah が実行される $RAHENV の値が使用されます。

rah とともに実行される . ファイルの指定 (Linux および UNIX)

このトピックでは、接頭部シーケンスが指定されない場合に実行される .file をリストしています。

注: この節の情報は Linux および UNIX プラットフォームだけに適用されます。

P .profile

E $RAHENV で指名されたファイル (通常は .kshrc)

K E と同じ

PE $RAHENV で指名されたファイル (通常は .kshrc) が後に続く .profile

B PE と同じ

N なし (またはどちらでもない)

注: ログイン・シェルが Korn シェルでない場合は、実行を指定した任意のドット・ファイルは Korn シェル・プロセスで実行されるので、 Korn シェル構文に従う必要があります。したがって、例えばログイン・シェルが C シェルの場合、rah で実行されるコマンド用に .cshrc 環境をセットアップするには、.cshrc に相応する Korn シェル INSTHOME/.profile を作成して、INSTHOME/.cshrc の中で指定する必要があります。

```
setenv RAHDOTFILES P
```

あるいは、.cshrc に相応する Korn シェル INSTHOME/.kshrc を作成して、INSTHOME/.cshrc の中で指定する必要があります。

```
setenv RAHDOTFILES E
setenv RAHENV INSTHOME/.kshrc
```

また、tty が存在しない場合 (rsh によって起動される場合など) には、 .cshrc を stdout に書き出さないようにすることも重要です。そうするには、stdout に書き出す行を、例えば次のように囲むことができます。

```
if { tty -s } then echo "executed .cshrc";
endif
```

rah に関する問題の判別 (Linux、UNIX)

このトピックでは、rah の実行時に発生する可能性のあるいくつかの問題を取り上げ、その対処方法のヒントを示します。

注: この節の情報は Linux および UNIX プラットフォームだけに適用されます。

1. rah がハングしている (または非常に長時間かかっている)。

   この問題は、下記が原因とみられます。

   - 出力をバッファーに入れる必要があると rah が判断したが、RAHCHECKBUF=no がエクスポートされなかった。このため、rah はコマンドを実行する前にコマンドをすべてのコンピューターに送って、バッファー・ディレクトリーの存在をチェックし、存在しなければバッファー・ディレクトリーを作成します。
   - コマンド送信先のコンピューターのうち、1 つ以上が応答していない。rsh コマンドは最終的にタイムアウトになりますが、タイムアウトのインターバルは極めて長く、通常は約 60 秒です。

2. 次のようなメッセージを受け取った。

   - ログインの誤り
   - 許可が否定された

   いずれかのコンピューターの .hosts ファイル内で、 rah を実行する ID が正しく定義されていません。あるいは、 rah を実行する ID の .rhosts ファイル内で、いずれかのコンピューターが正しく定義されていません。ssh を使用するよ

うに DB2RSHCMD レジストリー変数が構成されている場合、各コンピューター上の ssh クライアントとサーバーが正しく構成されていない可能性があります。

注: データベース・パーティション同士の平文でのパスワードの伝送に関する、より強力なセキュリティーが必要になる場合があります。これは使用するリモート・シェル・プログラムに依存します。 rah は DB2RSHCMD レジストリー変数によって指定されるリモート・シェル・プログラムを使用します。 ssh (セキュリティーを追加する場合) または rsh (HP-UX では remsh) の 2 つのリモート・シェル・プログラムの中から選択できます。このレジストリー変数が設定されない場合、rsh (HP-UX では remsh) が使用されます。

3. バックグラウンドのリモート・シェルを使用して並列にコマンドを実行している場合、コマンドが実行されてコンピューターで予期された経過時間内に完了するが、 rah がそれを検知してシェル・プロンプトを準備するのに時間がかかる。

   rah を実行している ID の .rhosts ファイル内で、いずれかのコンピューターが正しく定義されていないか、または ssh を使用するように DB2RSHCMD レジストリー変数が構成されている場合、各コンピューター上の ssh クライアントとサーバーが正しく構成されていない可能性があります。

4. rah がシェル・コマンド行からは正しく実行されるものの、例えば、次のように rsh を使ってリモートで rah を実行すると、

   ```
   rsh somewher -l $USER db2_kill
   ```

   rah が完了しない。

   これは正常です。 rah は、自らが終了した後にも実行を続けるバックグラウンドのモニター・プロセスを開始します。これらのプロセスは、通常、実行したコマンドに関連するすべてのプロセスが終了するまで続行します。 db2_kill の場合は、これは、すべてのデータベース・マネージャーの終了を意味します。関連するコマンドが rahwaitfor および kill process_id> であるようなプロセスを探すことによって、モニター・プロセスを終了することができます。シグナル番号を指定してはなりません。代わりに、デフォルトの (15) にしておきます。

5. 同じ $RAHUSER の下で複数の rah コマンドが発行されると、 rah からの出力が正しく表示されない。または、rah は $RAHBUFNAME が存在しないと誤って報告する。

   これは、複数の rah が同時に実行して、出力のバッファリング用として同じバッファー・ファイル ($RAHBUFDIR/$RAHBUFNAME など) を使用しようとするためです。このような問題を避けるために、同時に実行されるそれぞれの rah コマンドごとに、別の $RAHBUFNAME を使用してください。例えば、以下の ksh では、

   ```
   export RAHBUFNAME=rahout
   rah ";$command_1" &
   export RAHBUFNAME=rah2out
   rah ";$command_2" &
   ```

   のようにするか、または次のようにして、一意の名前をシェルに自動的に選択させるようにします。

   ```
   RAHBUFNAME=rahout.$$ db2_all "....."
   ```

どの方法を使用する場合も、ディスク・スペースに制約があるならば、いずれかの時点でバッファー・ファイルを確実に終結処理する必要があります。 rah は実行の終わりにバッファー・ファイルを消去しませんが、次に同じバッファー・ファイルを指定した場合、既存のファイルを消去して再使用します。

6. 次のように入力して、

```
rah '"print from ()'
```

以下のメッセージを受け取った。

```
ksh: syntax error at line 1 : (' unexpected
```

() および ## の置換の前提条件は次のとおりです。

- rah ではなく、 db2_all を使用する。
- RAHOSTFILE を設定するか、またはデフォルトである /sqllib/db2nodes.cfg ファイルにすることによって、 RAHOSTFILE が使用されていることを確認する。このような前提条件がない場合は、rah は、 () と ## を現状のままにします。コマンド print from () は有効でないので、エラーになります。

コマンドを並列実行している場合、パフォーマンス改善のヒントとして、 & によって提供される機能が本当に必要でない限り、 |& ではなく | を、 ||& ; ではなく || を使用してください。 & を指定すると、必要なリモート・シェル・コマンドが増えて、パフォーマンスが低下するためです。

rah プロセスのモニター (Linux、UNIX)

リモート・コマンドがまだ実行しているか、またはバッファー出力がまだ累積されている間は、rah によって開始されたプロセスは、アクティビティーをモニターすることによって、実行が完了していないコマンドを示すメッセージを端末に書き出し、バッファー出力を取り出します。

注: この節の情報は Linux および UNIX プラットフォームだけに適用されます。

環境変数 RAHWAITTIME によって制御されるインターバルで、通知メッセージが書き出されます。この指定方法の詳細については、ヘルプ情報を参照してください。環境変数 RAHWAITTIME=0 を設定することによって、すべての通知メッセージを完全に抑止できます。

1 次モニター・プロセスは、 rahwaitfor という名前のコマンド (ps コマンドで示される) です。 最初の通知メッセージで、このプロセスの pid (プロセス ID) が示されます。その他のすべてのモニター・プロセスは、 rah スクリプト (またはシンボリック・リンクの名前) を実行する ksh コマンドとして表示されます。必要であれば、次のコマンドによって、すべてのモニター・プロセスを停止することができます。

```
kill <pid>
```

ここで、<pid> は、1 次モニター・プロセスのプロセス ID です。シグナル番号を指定してはなりません。デフォルトである 15 のままにしてください。これは、リモート・コマンドにはまったく影響しませんが、バッファー出力を自動的に表示しないようにします。 rah の 1 回の実行中に、 2 つ以上の異なるモニター・プロセスのセットが、異なる時点で実行される可能性があることに注意してください。ただし、任意の時点で現行のセットを停止すると、その後はもう開始されません。

通常のログイン・シェルが /bin/ksh のような Korn シェルでない場合には、 rah を使用できますが、以下の特殊文字が含まれるコマンドを入力するときの規則が少し異なります。

```
"  unsubstituted $ '
```

詳細情報を表示するには、rah "?" と入力してください。また、Linux および UNIX 環境では、リモート・コマンドを実行する ID のログイン・シェルが Korn シェルでない場合、 rah を実行する ID のログイン・シェルもまた、 Korn シェルであってはなりません。(rah は、リモート ID のシェルが、ローカル ID に基づく Korn シェルであるかどうか判断します。) シェルは、単一引用符で囲まれたストリングに対して、置換または特別な処理を行ってはなりません。厳密に元のままにする必要があります。

Windows での rah のデフォルト環境プロファイルの設定

rah コマンドのデフォルト環境プロファイルを設定するには、db2rah.env ファイルを使用します。これはインスタンス・ディレクトリー内に作成する必要があります。

注: この節の情報は Windows だけに適用されます。

ファイルは以下の形式にする必要があります。

```
; This is a comment line
DB2INSTANCE=instancename
DB2DBDFT=database
; End of file
```

rah の環境を初期設定するのに必要な環境変数をすべて指定できます。

第 11 章 表およびその他の関連する表オブジェクトの作成

パーティション・データベース環境での表

パーティション・データベース環境で幾つかのデータベース・パーティションにまたがった表を作成することには、パフォーマンス上の利点があります。データの検索に関連した作業は、データベース・パーティションの中で分割することができます。

物理的に分割つまり配分される表を作成する前に、以下のことを考慮する必要があります。

- 表スペースは、複数のデータベース・パーティションにわたって展開することができます。展開するデータベース・パーティションの数は、データベース・パーティション・グループの中のデータベース・パーティションの数によって決まります。
- 表は、同じ表スペースに置かれるか、または、最初の表スペースに加えて、同じデータベース・パーティション・グループに関連する別の表スペースの中に置かれることによって、併置を行うことができます。

表を作成している時、いくつかのデータベース・パーティションの一部になる表を作成するように指定されます。パーティション・データベース環境で表を作成する場合、*分散キー* という、追加のオプションがあります。分散キーは、表の定義の一部であるキーです。このキーは、各データ行が保管されるデータベース・パーティションを判別します。

分散キーを明示して指定しない場合、以下のデフォルトが使用されます。*デフォルトの分散キーが適切であることを確認*してください。

- 主キーが CREATE TABLE ステートメントに指定された場合、主キーの最初の列が分散キーとして使用されます。
- 複数パーティションのデータベース・パーティション・グループでは、主キーがない場合、ロング・フィールドでない最初の列が使用されます。
- デフォルトの分散キーの要件を満たす列がない場合、表は分散キーなしで作成されます (これは、単一パーティションのデータベース・パーティション・グループでのみ許されます)。

*後から変更することはできない*ため、適切な分散キーを注意深く選択する必要があります。さらに、何らかのユニーク索引を (したがって、ユニーク・キーまたは主キーも)、分散キーのスーパーセットとして定義しなければなりません。つまり、分散キーが定義された場合、ユニーク・キーおよび主キーは、分散キーと同じ列をすべて含まなければなりません (ユニーク・キーと主キーには、それ以上の列が含まれる場合があります)。

表のデータベース・パーティションのサイズは、使用されている表スペースのタイプとページ・サイズに関連した特定の制限の量か、使用できるディスク・スペースの量のうち小さい方になります。例えば、ページ・サイズが 4 KB の大きな DMS 表スペースを想定すると、表のサイズは、2 TB にデータベース・パーティションの

数を乗算した量か、使用できるディスク・スペースの量のうち小さい方になります。データベース・マネージャーのページ・サイズ制限の完全なリストについては、関連リンクを参照してください。

コマンド行を使用してパーティション・データベース環境に 1 つの表を作成するには、以下のように入力します。

```
CREATE TABLE name>
  (<column_name> <data_type> <null_attribute>)
  IN <tagle_space_name>
  INDEX IN <index_space_name>
  LONG IN <long_space_name>
  DISTRIBUTE BY HASH (<column_name>)
```

以下はその例です。

```
CREATE TABLE MIXREC (MIX_CNTL INTEGER NOT NULL,
                     MIX_DESC CHAR(20) NOT NULL,
                     MIX_CHR  CHAR(9) NOT NULL,
                     MIX_INT INTEGER NOT NULL,
                     MIX_INTS SMALLINT NOT NULL,
                     MIX_DEC DECIMAL NOT NULL,
                     MIX_FLT FLOAT NOT NULL,
                     MIX_DATE DATE NOT NULL,
                     MIX_TIME TIME NOT NULL,
                     MIX_TMSTMP TIMESTAMP NOT NULL)
                     IN MIXTS12
                     DISTRIBUTE BY HASH (MIX_INT)
```

上記の例で、表スペースは MIXTS12 であり、分散キーは MIX_INT です。分散キーが明示して指定されない場合、分散キーは MIX_CNTL になります。 (主キーが指定されず、分散キーが定義されない場合、分散キーは、リスト内の最初のロング列以外の列になります。)

1 つの表の 1 つの行、およびその行に関するすべての情報は、常に同じデータベース・パーティション上に常駐します。

パーティション表でのラージ・オブジェクトの動作

パーティション表は、データ・パーティションまたは範囲と呼ばれる複数のストレージ・オブジェクトに表データを分割するというデータ編成スキームを使用します。分割は、表の 1 つ以上の表パーティション・キー列の値に従って行われます。指定された表のデータは、CREATE TABLE ステートメントの PARTITION BY 節で提供された仕様に基づいて、複数のストレージ・オブジェクトにパーティション化されます。このストレージ・オブジェクトは異なる表スペース、同じ表スペース内、またはその両方に配置することができます。

パーティション表のラージ・オブジェクトは、デフォルトでは、対応するデータ・オブジェクトと同じ表スペースに保管されます。このことは、表スペースを 1 つだけ使用するパーティション表にも、複数の表スペースを使用するパーティション表にも当てはまります。パーティション表のデータが複数の表スペースに保管される場合は、ラージ・オブジェクト・データも複数の表スペースに保管されます。

このデフォルト動作をオーバーライドするには、CREATE TABLE ステートメントの LONG IN 節を使用します。表の LONG データが保管される表スペースのリストを指定できます。デフォルト動作をオーバーライドするようにする場合、LONG

IN 節で指定する表スペースは LARGE 表スペースでなければなりません。 1 つ以上のデータ・パーティションの LONG データが別個の表スペースに保管されるように指定する場合は、その表のすべてのデータ・パーティションについてそうする必要があります。つまり、一部のデータ・パーティションについてはリモート側で LONG データを保管し、他のデータ・パーティションについてはローカル側で LONG データを保管するということはできません。デフォルト動作を使用しても、LONG IN 節を使用してデフォルト動作をオーバーライドしても、各データ・パーティションに対応した LONG オブジェクトが作成されます。SMS 表スペースの場合、LONG データは、それが属するデータ・オブジェクトと同じ表スペースになければなりません。各データ・パーティションに対応する LONG データ・オブジェクトを保管するために使用されるすべての表スペースで、ページ・サイズ、エクステント・サイズ、ストレージ・メカニズム (DMS または SMS)、タイプ (REGULAR または LARGE) が、同じでなければなりません。リモート LARGE 表スペースは LARGE タイプである必要があります。また、SMS は使用できません。

例えば次の CREATE TABLE ステートメントは、各データ・パーティションの CLOB データのオブジェクトを、データと同じ表スペースに作成します。

```
CREATE TABLE document(id INT, contents CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1,
 STARTING FROM 101 ENDING AT 200 IN tbsp2,
 STARTING FROM 201 ENDING AT 300 IN tbsp3,
 STARTING FROM 301 ENDING AT 400 IN tbsp4);
```

LONG IN を使用することにより、データがある表スペースとは別個の 1 つ以上の LARGE 表スペースに CLOB データを配置できます。

```
CREATE TABLE document(id INT, contents CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1 LONG IN large1,
STARTING FROM 101 ENDING AT 200 IN tbsp2 LONG IN large1,
STARTING FROM 201 ENDING AT 300 IN tbsp3 LONG IN large2,
STARTING FROM 301 ENDING AT 400 IN tbsp4 LONG IN large2);
```

注: LONG IN 節は、表レベルでデータ・パーティションごとに 1 つだけ使用できます。

パーティション表の作成

パーティション化された表は、表の 1 つ以上の表パーティション・キー列の値に従って、表データが、データ・パーティションまたは範囲と呼ばれる複数のストレージ・オブジェクトに分割されるデータ編成スキームを使用します。指定された表のデータは、CREATE TABLE ステートメントの PARTITION BY 節で提供された仕様に基づいて、複数のストレージ・オブジェクトにパーティション化されます。このストレージ・オブジェクトは異なる表スペース、同じ表スペース内、またはその両方に配置することができます。

表を作成するには、ステートメントの許可 ID によって保持される特権に、以下の権限または特権の少なくとも 1 つが含まれていなければなりません。

- データベースに対する CREATETAB 権限、表によって使用されるすべての表スペースに対する USE 特権、および以下のいずれか。

- データベースに対する IMPLICIT_SCHEMA 権限 (表の暗黙または明示スキーマ名がない場合)
- スキーマに対する CREATEIN 特権 (表のスキーマ名が既存のスキーマを指す場合)

- SYSADM または DBADM 権限

パーティション表の作成は、CREATE TABLE ステートメントを使って行えます。

コマンド行からパーティション表を作成するには、次のように CREATE TABLE ステートメントを発行します。

```
CREATE TABLE <NAME> (<column_name>  <data_type>  <null_attribute>) IN
 <table space list> PARTITION BY RANGE (<column expression>)
 STARTING FROM <constant> ENDING <constant> EVERY <constant>
```

例えば、以下のステートメントは、a ≥ 1 および a ≤ 20 である行が PART0 (最初のデータ・パーティション) に、21 ≤ a ≤ 40 の行が PART1 (2 番目のデータ・パーティション) に、と順に上がっていき、最後に 81 ≤ a ≤ 100 が PART4 (最後のデータ・パーティション) にある表を作成します。

```
CREATE TABLE foo(a INT)
 PARTITION BY RANGE (a) (STARTING FROM (1)
 ENDING AT (100) EVERY (20))
```

パーティション表の範囲の定義

各データ・パーティションの範囲は、パーティション表の作成時に指定できます。パーティション表では、表の表パーティション・キー列にある値に従って表データを複数のデータ・パーティションに分割するデータ編成スキームを使用します。

指定された表のデータは、CREATE TABLE ステートメントの PARTITION BY 節で提供された仕様に基づいて、複数のストレージ・オブジェクトにパーティション化されます。範囲は、PARTITION BY 節の STARTING FROM 値および ENDING AT 値によって指定されます。

各データ・パーティションの範囲を完全に定義するには、適切な境界を指定する必要があります。以下は、パーティション表の範囲を定義するときに検討すべきガイドラインのリストです。

- STARTING 節は、データ・パーティション範囲の下の境界を指定します。この節は、最低データ・パーティション範囲で必須です (ただし境界は MINVALUE と定義することができます)。最低データ・パーティション範囲とは、指定された下限境界を持つデータ・パーティションのことです。
- ENDING (または VALUES) 節は、データ・パーティション範囲の上の境界を指定します。この節は、最高データ・パーティション範囲で必須です (ただし境界は MAXVALUE と定義することができます)。最高データ・パーティション範囲とは、指定された上限境界を持つデータ・パーティションのことです。
- データ・パーティションに ENDING 節を指定しない場合、次に大きいデータ・パーティションが STARTING 節を指定しなければなりません。同様に、STARTING 節を指定しない場合、前のデータ・パーティションが ENDING 節を指定しなければなりません。

- MINVALUE は、使用されている列タイプに指定できるどの値よりも小さな値を指定します。 MINVALUE と INCLUSIVE、または MINVALUE と EXCLUSIVE の組み合わせで指定することはできません。
- MAXVALUE は、使用されている列タイプに指定できるどの値よりも大きな値を指定します。 MAXVALUE と INCLUSIVE、または MAXVALUE と EXCLUSIVE の組み合わせで指定することはできません。
- INCLUSIVE は、指定された値に等しい値すべてが、この境界を含むデータ・パーティションに組み込まれることを指示します。
- EXCLUSIVE は、指定された値に等しい値すべてが、この境界を含むデータ・パーティションに組み込まれないことを指示します。
- CREATE TABLE ステートメントの NULL 節は、データ・パーティションの配置を考慮する際に NULL 値を高位にソートするかまたは低位にソートするかを指定します。デフォルトでは、NULL 値は高位にソートされます。表パーティション・キー列の NULL 値は正の無限大として扱われ、MAXVALUE で終わる範囲に配置されます。そのデータ・パーティションが定義されない場合、NULL 値は範囲外とみなされます。 NULL 値を表パーティション・キー列から除外する場合は NOT NULL 制約を使用します。 LAST は、NULL 値が値のソート・リストの最後に現れることを指定します。 FIRST は、NULL 値が値のソート・リストの最初に現れることを指定します。
- 長形式の構文を使用する場合、各データ・パーティションに少なくとも 1 つの境界を指定する必要があります。

ヒント: 表のデータ・パーティションの定義を開始する前に、表をパーティション化することによって得られるメリット、およびパーティション列として選択する列に影響を与える要因を理解しておくことが大切です。

それぞれのデータ・パーティションごとに指定される範囲は自動的に生成することも、手動で生成することもできます。

自動生成

自動生成は、多くのデータ・パーティションを素早く、簡単に作成するための単純な方法です。この方法は、日付または番号に基づいて同じサイズになる範囲に適しています。

例 1 および 2 は、CREATE TABLE ステートメントを使用して、各データ・パーティションに指定される範囲を自動的に定義および生成する方法を示しています。

例 1:

次の範囲を定義して、CREATE TABLE ステートメントを発行します。

```
CREATE TABLE lineitem (
  l_orderkey    DECIMAL(10,0) NOT NULL,
  l_quantity    DECIMAL(12,2),
  l_shipdate    DATE,
  l_year_month  INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate)))
     PARTITION BY RANGE(l_shipdate)
    (STARTING ('1/1/1992') ENDING ('12/31/1992') EVERY 1 MONTH);
```

このステートメントは、それぞれ 1 つのキー値を持つ 12 のデータ・パーティションになります。(l_shipdate) >= ('1/1/1992')、(l_shipdate) < ('3/1/1992')、(l_shipdate)

< ('4/1/1992')、(l_shipdate) < ('5/1/1992')、... (l_shipdate) < ('12/1/1992')、(l_shipdate) < ('12/31/1992') というようになります。

開始境界全体 ('1/1/1992') が包含的であるため (デフォルト)、最初のデータ・パーティションの開始値は包含的になります。同様に、終了境界全体 ('12/31/1992') が包含的であるため (デフォルト)、最後のデータ・パーティションの終了境界は包含的になります。残りの STARTING 値は包含的で、残りの ENDING 値はすべて排他的です。各データ・パーティションは n 個のキー値を保持します (n は EVERY 節によって与えられます)。各データ・パーティションの範囲の終わりを見つけるには、公式 (start + every) を使用します。 EVERY 値が均等に START と END の範囲に分かれない場合、最後のデータ・パーティションのキー値は少なくなる可能性があります。

例 2:

次の範囲を定義して、CREATE TABLE ステートメントを発行します。

```
CREATE TABLE t(a INT, b INT)
 PARTITION BY RANGE(b) (STARTING FROM (1)
  EXCLUSIVE ENDING AT (1000) EVERY (100))
```

このステートメントは、それぞれ 100 のキー値を持つ 10 のデータ・パーティションになります (1 < b <= 101、101 < b <= 201、... 901 < b <= 1000 というようになります)。

開始境界全体 (1) が排他的であるため、最初のデータ・パーティション (b > 1 および b <= 101) の開始値は排他的になります。同様に、終了境界全体 (1000) が包含的であるため、最後のデータ・パーティション (b > 901、b <= 1000) の終了境界は包含的になります。残りの STARTING 値はすべて排他的で、残りの ENDING 値はすべて包含的です。各データ・パーティションは n 個のキー値を保持します (n は EVERY 節によって与えられます)。補足: もし、開始境界だけでなく終了境界も排他的な指定を行った場合、最後のデータ・パーティション (tbsp10) の終了境界は排他的になります。それ以外の ENDING 値はすべて包含的です。よって、最後以外のデータ・パーティションは n 個のキー値を保持し、最後のパーティションは n-1 個のキー値を保持します (n は EVERY 節によって与えられます)。

手動生成

手動生成では、PARTITION BY 節でリストされるそれぞれの範囲ごとに新規のデータ・パーティションが作成されます。この形式の構文では、範囲を定義して、データおよび LOB の配置オプションをより柔軟に増やすことができます。例 3 および 4 は、CREATE TABLE ステートメントを使用して、データ・パーティションに指定される範囲を手動で定義および生成する方法を示しています。

例 3:

このステートメントは、2 つの日付列 (どちらも生成列) でパーティション化します。 CREATE TABLE 構文の自動生成形式の使用、および各範囲の一方の終端のみが指定されることにご注意ください。他方の終端は隣接のデータ・パーティションおよび INCLUSIVE オプションの使用により暗黙指定されます。

```
CREATE TABLE sales(invoice_date date, inv_month int NOT NULL
GENERATED ALWAYS AS (month(invoice_date)), inv_year INT NOT
NULL GENERATED ALWAYS AS ( year(invoice_date)),
item_id int NOT NULL,
cust_id int NOT NULL) PARTITION BY RANGE (inv_year,
inv_month)
(PART Q1_02 STARTING (2002,1) ENDING (2002, 3) INCLUSIVE,
PART Q2_02 ENDING (2002, 6) INCLUSIVE,
PART Q3_02 ENDING (2002, 9) INCLUSIVE,
PART Q4_02 ENDING (2002,12) INCLUSIVE,
PART CURRENT ENDING (MAXVALUE, MAXVALUE));
```

範囲内のギャップは許可されています。 CREATE TABLE 構文は、前のデータ・パーティションの ENDING 値に反しない範囲の STARTING 値を指定できるようにすることにより、ギャップをサポートします。

例 4:

101 から 200 の間の値のギャップを持つ表を作成します。

```
CREATE TABLE foo(a INT)
 PARTITION BY RANGE(a)
   (STARTING FROM (1) ENDING AT (100),
   STARTING FROM (201) ENDING AT (300))
```

ALTER TABLE ステートメントを使用すると、データ・パーティションを追加したり除去できますが、範囲内でギャップが発生する可能性もあります。

パーティション表に行を挿入すると、その行はそのキー値およびそれが含まれる範囲に基づいて適切なデータ・パーティションに自動的に配置されます。その行が表に対する定義の範囲外にある場合、挿入は失敗し、アプリケーションに次のエラーが戻されます。

```
SQL0327N  The row cannot be inserted into table <tablename>
because it is outside the bounds of the defined data partition ranges.
   SQLSTATE=22525
```

制約事項

- 表レベルの制約事項は以下のとおりです。
  - 構文の自動生成形式 (EVERY 節を含む) を使用して作成された表は、表パーティション・キーで数値または日付時刻タイプを使用するという制限があります。
- ステートメント・レベルの制約事項は以下のとおりです。
  - 構文の自動生成形式では MINVALUE および MAXVALUE はサポートされません。
  - 範囲は昇順です。
  - 構文の自動生成形式に指定できる列は 1 つだけです。
  - EVERY 節の増分はゼロより大きくなければなりません。
  - ENDING 値は STARTING 値以上でなければなりません。

既存の表およびビューをパーティション表にマイグレーションする

バージョン 9.1 の表のデータを空のパーティション表にマイグレーションするには、LOAD コマンドを使用します。

既存の表またはビューをパーティション表にマイグレーションするために使用できるアプローチとして、次の 3 つがあります。

- REGULAR 表からのマイグレーション時、空のパーティション表を新規作成し、LOAD from CURSOR を使用して、古い表にあるデータをパーティション表に、中間のステップを介さず直接移動する。
- REGULAR 表からのマイグレーション時、エクスポート・ユーティリティーまたは High Performance Unload を使用してソース表をアンロードし、空のパーティション表を新規作成し、LOAD コマンドを使用して空のパーティション表にデータを取り込む。
- Union All ビューからのマイグレーション時、1 つのダミー・データ・パーティションを持つパーティション表を作成し、その後すべての表をアタッチする。

例 1: REGULAR 表 t1 があるとします。

```
CREATE TABLE t1 (c1 int, c2 int);
```

空のパーティション表を新規作成します。

```
CREATE TABLE sales_dp (c1 int, c2 int)
   PARTITION BY RANGE (c1)
   (STARTING FROM 0 ENDING AT 10 EVERY 2);
```

表 t1 にデータを取り込みます。

```
INSERT INTO t1 VALUES (0,1), (4, 2), (6, 3);
```

データ・コピー用の中間フラット・ファイルを作成しないため、LOAD コマンドを発行して、SQL 照会からデータを直接プルし、新規のパーティション表にそのデータを取り込みます。

```
   SELECT * FROM t1;
   DECLARE c1 CURSOR FOR SELECT * FROM t1;
   LOAD FROM c1 of CURSOR INSERT INTO sales_dp;SELECT * FROM sales_dp;
SELECT * FROM sales_dp;
```

古い表をドロップします。

```
DROP TABLE t1;
```

UNION ALL ビューの変換

UNION ALL ビューにあるバージョン 9.1 データはパーティション表に変換することができます。 UNION ALL は、ブランチの除去に関してパフォーマンス上の利点を提供しつつ、大きな表を管理し、簡単に表データをロールインおよびロールアウトできるようにするために使用されます。表のパーティション化はこのすべてを実現しながら、管理はより容易になります。 ALTER TABLE ...ATTACH 操作を使用すると、基本表のデータを移動せずに変換を行えます。変換の後、索引および従属するビュー、またはマテリアライズ照会表 (MQT) を再作成する必要があります。

推奨されているのは、1 つのダミー・データ・パーティションを持つパーティション表を作成し、その後すべての union all ビューの表をアタッチする方法です。範囲のオーバーラップの問題を回避するために、ダミー・データ・パーティションは必ずプロセスの初期にドロップしておいてください。

例 2:

最初の表の表構文を UNION に作成します。

```
CREATE TABLE sales_0198(
 sales_date DATE NOT NULL,
 prod_id INTEGER,
 city_id INTEGER,
 channel_id INTEGER,
 revenue DECIMAL(20,2),
  CONSTRAINT ck_date
  CHECK
  (sales_date BETWEEN '01-01-1998' AND '01-31-1998'));
```

union all ビューのビュー構文を作成します。

```
CREATE VIEW all_sales AS
 (
 SELECT * FROM sales_0198
 WHERE sales_date BETWEEN '01-01-1998' AND '01-31-1998'
 UNION ALL
 SELECT * FROM sales_0298
 WHERE sales_date BETWEEN '02-01-1998' AND '02-28-1998'
 UNION ALL
 ...
 UNION ALL
 SELECT * FROM sales_1200
 WHERE sales_date BETWEEN '12-01-2000' AND '12-31-2000'
 );
```

1 つのダミー・パーティションを持つパーティション表を作成します。アタッチされる最初のデータ・パーティションとオーバーラップしないように範囲を選択する必要があります。

```
CREATE TABLE sales_dp (
 sales_date DATE NOT NULL,
 prod_id INTEGER,
 city_id INTEGER,
 channel_id INTEGER,
 revenue DECIMAL(20,2))
 PARTITION BY RANGE (sales_date)
 (PART dummy STARTING FROM '01-01-1900' ENDING AT '01-01-1900');
```

最初の表をアタッチします。

```
ALTER TABLE sales_dp ATTACH PARTITION
STARTING FROM '01-01-1998' ENDING AT '01-31-1998'
FROM sales_0198;
```

ダミー・パーティションをドロップします。

```
ALTER TABLE sales_dp DETACH PARTITION dummy
INTO dummy;
 DROP TABLE dummy;
```

残りのパーティションをアタッチします。

```
ALTER TABLE sales_dp ATTACH PARTITION STARTING
FROM '02-01-1998' ENDING AT '02-28-1998' FROM sales_0298;

...

ALTER TABLE sales_dp ATTACH PARTITION STARTING
FROM '12-01-2000' ENDING AT '12-31-2000' FROM sales_1200;
```

SET INTEGRITY ステートメントを発行して、アタッチされたデータ・パーティションをオンラインにします。

```
SET INTEGRITY FOR sales_dp IMMEDIATE CHECKED
FOR EXCEPTION IN sales_dp USE sales_ex;
```

必要に応じて索引を作成します。

変換に関する考慮事項

ソース列とターゲット列両方の特定の列の SYSCAT.COLUMNS IMPLICITVALUE フィールドの値が非ヌル値でありそれらの値が一致しない場合は、データ・パーティションのアタッチを行うことができません。これが当てはまる場合は、ソース表をドロップした後それを再作成する必要があります。

列の SYSCAT.COLUMNS IMPLICITVALUE フィールドに非ヌル値を指定できるのは、次の条件のいずれかが満たされる場合です。

- ALTER TABLE ...ADD COLUMN ステートメントの結果として列が作成される
- アタッチ時にソース表から IMPLICITVALUE フィールドが伝搬される
- デタッチ時にソース表から IMPLICITVALUE フィールドが継承される
- V8 から V9 へのマイグレーション時に IMPLICITVALUE フィールドが設定される (ここで、列が追加列であるかどうか、あるいはその可能性があるかどうかが判別される)。データベースの列が追加列であるかどうかが不確かな場合には追加列として扱われます。追加列とは、ALTER TABLE ...ADD COLUMN ステートメントの結果として作成される列のことです。

これらの不整合を回避するために、 ATTACH 操作が関係するソースおよびターゲット表は、常に同一の列で定義して作成することが推奨されています。特に、ALTER TABLE ステートメントを使用して、ATTACH 操作のターゲット表へ列を追加することはしないでください。

パーティション表の作業時にミスマッチを回避する上でのベスト・プラクティスについては、 227 ページの『パーティション表へのデータ・パーティションのアタッチに関するガイドライン』を参照してください。

パーティション化されたマテリアライズ照会表 (MQT) の動作

すべてのタイプのマテリアライズ照会表 (MQT) がパーティション表でサポートされます。パーティション化された MQT を扱う際、アタッチおよびデタッチされたデータ・パーティションを最も効率的に管理するのに役立つ多くの指針があります。

以下の指針および制限事項は、デタッチされた従属のあるパーティション化された MQT またはパーティション表を扱う際に適用されます。

- DETACH PARTITION 操作を実行し、デタッチされるデータ・パーティションに関して増分的に保守していく必要のある従属表がある場合 (これらの従属表は、デタッチされた従属表と呼ばれる)、新たにデタッチされた表には最初はアクセスすることができません。 SYSCAT.TABLES カタログ・ビューの TYPE 列の表に L とマークされます。これはデタッチされた表と呼ばれます。これにより、デタッチされた従属表を増分的に保守していくために SET INTEGRITY ステートメントを実行するまで、表に対する読み取り、変更、またはドロップを防ぐことがで

きます。デタッチされたすべての従属表に対して SET INTEGRITY ステートメントが実行された後、デタッチされた表は通常の表に移行され、完全にアクセス可能になります。

- デタッチされた表がまだアクセス可能になっていないことを検出するには、SYSCAT.TABDETACHEDDEP カタログ・ビューを照会します。アクセス不能なデタッチされた表が検出される場合、すべてのデタッチされた従属に対して SET INTEGRITY ステートメントを IMMEDIATE CHECKED オプションを指定して実行し、デタッチされた表を通常のアクセス可能な表に遷移します。すべてのデタッチされた従属を保守する前に、デタッチされた表にアクセスすると、エラー・コード SQL20285N が戻されます。
- DATAPARTITIONNUM 関数は、マテリアライズ照会表 (MQT) 定義では使用できません。この関数を使用して MQT を作成しようとすると、エラー (SQLCODE SQL20058N、SQLSTATE 428EC) が戻されます。
- デタッチされたデータ・パーティションを持つ表で索引を作成すると、その索引にはデタッチされたデータ・パーティション内のデータは含まれません。ただし、デタッチされたデータ・パーティションが、そのデータ・パーティションに関連して増分リフレッシュを行う必要がある従属マテリアライズ照会表 (MQT) を持っている場合は別です。この場合、索引には、そのデタッチされたデータ・パーティションのデータが含まれます。
- アタッチされたデータ・パーティションを持つ表を MQT に変更することは許可されません。
- パーティション化されたステージング表はサポートされていません。
- MQT へのアタッチは直接サポートされていません。詳細については、例 1 を参照してください。

例 1: 通常の表へのパーティション化された MQT の変換

パーティション化された MQT で ATTACH 操作は直接サポートされていませんが、パーティション化された MQT を通常の表に変換して、表データを適切にロールインおよびロールアウトし、その後に表を MQT に再び変換すると同じ効果を得ることができます。以下の CREATE TABLE ステートメントおよび ALTER TABLE ステートメントは、この効果を例示しています。

```
CREATE TABLE lineitem (
  l_orderkey    DECIMAL(10,0) NOT NULL,
  l_quantity    DECIMAL(12,2),
  l_shipdate    DATE,
  l_year_month  INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate)))
    PARTITION BY RANGE(l_shipdate)
    (STARTING ('1/1/1992') ENDING ('12/31/1993') EVERY 1 MONTH);
CREATE TABLE lineitem_ex (
  l_orderkey    DECIMAL(10,0) NOT NULL,
  l_quantity    DECIMAL(12,2),
  l_shipdate    DATE,
  l_year_month  INT,
  ts            TIMESTAMP,
  msg           CLOB(32K));

CREATE TABLE quan_by_month (
  q_year_month, q_count) AS
    (SELECT l_year_month AS q_year_month, COUNT(*) AS q_count
     FROM lineitem
     GROUP BY l_year_month)
     DATA INITIALLY DEFERRED REFRESH IMMEDIATE
```

```
     PARTITION BY RANGE(q_year_month)
     (STARTING (199201) ENDING (199212) EVERY (1),
      STARTING (199301) ENDING (199312) EVERY (1));
CREATE TABLE quan_by_month_ex(
  q_year_month  INT,
  q_count       INT NOT NULL,
  ts            TIMESTAMP,
  msg           CLOB(32K));

SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED;
CREATE INDEX qbmx ON quan_by_month(q_year_month);

ALTER TABLE quan_by_month DROP MATERIALIZED QUERY;
ALTER TABLE lineitem DETACH PARTITION part0 INTO li_reuse;
ALTER TABLE quan_by_month DETACH PARTITION part0 INTO qm_reuse;

SET INTEGRITY FOR li_reuse OFF;
ALTER TABLE li_reuse ALTER l_year_month SET GENERATED ALWAYS
AS (YEAR(l_shipdate)*100 + MONTH(l_shipdate));

LOAD FROM part_mqt_rotate.del OF DEL MODIFIED BY GENERATEDIGNORE
MESSAGES load.msg REPLACE INTO li_reuse;

DECLARE load_cursor CURSOR FOR
  SELECT l_year_month, COUNT(*)
    FROM li_reuse
    GROUP BY l_year_month;
LOAD FROM load_cursor OF CURSOR MESSAGES load.msg
  REPLACE INTO qm_reuse;

ALTER TABLE lineitem ATTACH PARTITION STARTING '1/1/1994'
  ENDING '1/31/1994' FROM li_reuse;

SET INTEGRITY FOR lineitem ALLOW WRITE ACCESS IMMEDIATE CHECKED
FOR EXCEPTION IN lineitem USE lineitem_ex;

ALTER TABLE quan_by_month ATTACH PARTITION STARTING 199401
  ENDING 199401 FROM qm_reuse;

SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED
  FOR EXCEPTION IN quan_by_month USE quan_by_month_ex;

ALTER TABLE quan_by_month ADD MATERIALIZED QUERY
  (SELECT l_year_month AS q_year_month, COUNT(*) AS q_count
     FROM lineitem
       GROUP BY l_year_month)
         DATA INITIALLY DEFERRED REFRESH IMMEDIATE;

SET INTEGRITY FOR QUAN_BY_MONTH ALL IMMEDIATE UNCHECKED;
```

SET INTEGRITY ステートメントを IMMEDIATE CHECKED オプションを指定して使用し、アタッチされたデータ・パーティションの整合性違反を検査します。このステップは、表を MQT に再び変更する前に必要です。SET INTEGRITY ステートメントを IMMEDIATE UNCHECKED オプションを指定して使用すると、MQT の必要なフル・リフレッシュを迂回します。最良のパフォーマンスを得るためには、MQT 上の索引が必要です。適切であれば、SET INTEGRITY ステートメントで例外表を使用することが推奨されています。

一般に、パーティション化されている大きなファクト表にパーティション化された MQT を作成します。大きなファクト表の表データにロールインまたはロールアウトする場合、例 2 で示されているように、パーティション化された MQT を手動で調整する必要があります。

例 2: パーティション化された MQT の手動調整

MQT (quan_by_month) を変更して、通常のパーティション表に変換します。

```
ALTER TABLE quan_by_month DROP MATERIALIZED QUERY;
```

ロールアウトするデータをファクト表 (lineitem) と MQT からデタッチし、ロールインする新しいデータをステージング表 li_reuse に再ロードします。

```
ALTER TABLE lineitem DETACH PARTITION part0 INTO li_reuse;

LOAD FROM part_mqt_rotate.del OF DEL MESSAGES load.msg REPLACE INTO li_reuse;

ALTER TABLE quan_by_month DETACH PARTITION part0 INTO qm_reuse;
```

挿入を行う前に qm_reuse を整理します。これにより、副選択データを挿入する前に、デタッチされたデータが削除されます。このことは、ロードのデータ・ファイルが副選択の内容となるような MQT へのロード置換により、行われます。

```
db2 load from datafile.del of del replace into qm_reuse
```

INSERT INTO ... (SELECT ...) を使用すると、表を手動でリフレッシュできます。新しいデータでのみこれは必要で、アタッチの前にこのステートメントを実行する必要があります。

```
INSERT INTO qm_reuse
  (SELECT COUNT(*) AS q_count, l_year_month AS q_year_month
       FROM li_reuse
         GROUP BY l_year_month);
```

これで、ファクト表の新しいデータをロールインできます。

```
ALTER TABLE lineitem ATTACH PARTITION STARTING '1/1/1994'
ENDING '1/31/1994' FROM TABLE li_reuse;
SET INTEGRITY FOR lineitem ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR
EXCEPTION IN li_reuse USE li_reuse_ex;
```

次に、MQT のデータをロールインします。

```
ALTER TABLE quan_by_month ATTACH PARTITION STARTING 199401
ENDING 199401 FROM TABLE qm_reuse;
SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED;
```

データ・パーティションをアタッチした後、新しいデータが範囲内にあることを検査する必要があります。

```
ALTER TABLE quan_by_month ADD MATERIALIZED QUERY
  (SELECT COUNT(*) AS q_count, l_year_month AS q_year_month
     FROM lineitem
       GROUP BY l_year_month)
         DATA INITIALLY DEFERRED REFRESH IMMEDIATE;
SET INTEGRITY FOR QUAN_BY_MONTH ALL IMMEDIATE UNCHECKED;
```

データが SET INTEGRITY ステートメントによって検証されるまでは、データにアクセスすることはできません。 REFRESH TABLE 操作はサポートされますが、このシナリオでは、ATTACH PARTITION 操作および DETACH PARTITION 操作を使用してパーティション化された MQT の手動による保守を例示しています。 SET INTEGRITY ステートメントの IMMEDIATE UNCHECKED 節により、ユーザーによって検証済みとしてデータがマークされます。

範囲クラスター表の作成

範囲クラスター表に使用されるアルゴリズム

レコードのキー値を表内の特定の行のロケーションと等価にするためのアルゴリズムが使用されます。基本のアルゴリズムは、かなりシンプルなものです。最も基本的な形 (複数列を使用せず、単一の列のみでキーが構成されている) では、アルゴリズムは、論理行番号にシーケンス番号をマップします。

また、アルゴリズムは、レコードのキーを使用して、論理ページ番号とスロット番号を判別します。このプロセスは、レコード、つまり表内の特定の行へのアクセスをかなり高速なものにします。

アルゴリズムには、ハッシュは関係していません。ハッシュは、キー値の配列を保持しないからです。キー値の配列の保持は、過去の表データを再編成する必要を省いてくれるので、不可欠です。

表の各レコード・キーには、以下の特性が必要です。

- ユニーク
- 非 NULL
- 整数 (SMALLINT、INTEGER、または BIGINT)
- 単調に増加する
- 事前に決定された、キーの各列に基づく一連の範囲内にある

  ALLOW OVERFLOW オプションは、キー値が定義されている範囲を超える表を作成するのに使用できます。 DISALLOW OVERFLOW オプションは、キー値が定義されている範囲を超えない表を作成するのに使用できます。この場合、範囲によって示されている境界の外にレコードが挿入されると、 SQL エラー・メッセージが戻されます。

シーケンス・キー範囲がきつくクラスタリングされた (高密度な) アプリケーションは、範囲クラスター表に最適な候補といえます。このタイプのキーを使用して範囲クラスター表を作成する場合、表の行の論理ロケーションを生成するためにキーが使用されます。このプロセスにより、別個の索引が必要でなくなります。

範囲クラスター表の索引

範囲クラスター表では、レコードのキーに基づくレコードの探索、スキャンの開始と停止の適用、データの縦方向の配分が、索引によって行われます。 RCT を使用する場合、考慮に入れられない索引の唯一のプロパティーはデータの縦方向の配分です。

通常表との違い

範囲クラスター表の使用を決定する際は、通常表にはない固有の特性を考慮する必要があります。

- 範囲クラスター表には、フリー・スペース制御レコード (FSCR) がありません。
- スペースが事前割り振りされます。

表のレコードがいっぱいになっていない場合であっても、表で使用するための表のスペースが事前割り振りおよび予約されています。表の作成時、表には何もレコードが入っていませんが、ページの範囲全体は事前割り振りされています。事前割り振りは、レコードのサイズと、保管されるレコードの最大数に基づいて決定されます。

  - 各レコードで VARCHAR のような可変長フィールドが使用されるときは、フィールドの最大長が使用され、すべてのレコード・サイズが固定長になります。各レコードの固定長は、すべて、レコードの最大数と合わせて、必要なスペースを決定するために使用されます。
  - そのため、効率的に利用できない余分のスペースが割り振られることになる場合があります。
  - キー値がまばらである場合は、未使用のスペースが存在することになり、範囲スキャンのパフォーマンスが低くなります。
  - 範囲スキャンでは、それらのキー値を含む行がまだデータベースに挿入されていない場合であっても、範囲内のレコードのうち可能性のあるあらゆるレコードをスキャンする必要があります。

- スキーマの変更はできません。

  範囲クラスター表に対してスキーマの変更が必要になった場合は、表を再作成して、新しいスキーマ名と以前の表のすべてのデータをその表に含める必要があります。特に、

  - キー範囲の変更はサポートされていません。

    この点は重要です。というのは、表の範囲を変更することが必要な場合には、目的の範囲の新しい表を作成しなければならず、新しい表に以前の表のデータを入れなければならないからです。

- 重複キー値は許されません。
- 定義されている範囲に含まれないキー値は許されません。

  これは、 DISALLOW OVERFLOW として定義されている範囲クラスター表についてのみ当てはまります。

  - NULL 値は明示的に禁止されています。

- 範囲クラスター索引は物理的には作成されません。

  RCT キー・プロパティーがある索引は、システム・カタログに示され、オプティマイザーによって選択できますが、索引はディスク上に作成されません。通常表では、表に関連付けられた各索引に、スペースも与える必要があります。 RCT では、RCT 索引にスペースは必要ありません。オプティマイザーが、この RCT 索引を参照するシステム・カタログの情報を使用して、表への正しいアクセス方式が確実に選択されるようにします。

- 範囲クラスター表と同じ定義に対して、主キーまたはユニーク・キーを作成することは、冗長であるため許されていません。
- 範囲クラスター表は、オリジナルのキー値の配列を保持します。キー値の配列は、表内の行のクラスタリングを保証する重要な機構です。

範囲クラスター表使用のガイドライン

範囲クラスター表 (RCT) を扱うときは、従う必要があるガイドラインがあります。

- キー値の範囲を定義する際、最小値の指定はオプションです。指定がない場合、最小値はデフォルトで 1 になります。最小値や最大値に負の値を指定することも可能です。負の値を扱う場合は、必ず最小値を明示的に宣言してください。例えば、ORGANIZE BY KEY SEQUENCE (F1 STARTING FROM -100 ENDING AT -10) のようにします。
- 範囲クラスター表の定義に使用されたのと同じキー値で通常の索引を作成することはできません。
- 範囲クラスター表では、一部の ALTER TABLE オプションが使用できません。表の物理構造に影響を与えないオプションは使用できます。
- 範囲クラスター表の作成プロセスでは必要なディスク・スペースの事前割り振りが行われるため、その分のスペースが使用可能でないと、表の作成は失敗します。

SQL コンパイラーによる範囲クラスター表の処理方法

SQL コンパイラーは、 2 次 B+ ツリー索引を持つ通常の表と同様の方法で範囲クラスター表 (RCT) を処理します。 B+ ツリー索引を通してレコードのロケーションやレコード ID (RID) を決定するのではなく、 RCT は、レコード・キー値に関係した機能的な検索と範囲定義のアルゴリズムを使用します。これは、RID を素早く取得するためにキー値を使用するため、索引を持つのと似ています。

要求されたデータへの最良のアクセス・パスを決定しようと機能するとき、 SQL コンパイラーは、表に関して保持されている統計情報を使用します。 RUNSTATS コマンドが出されると、表のスキャンの間に、索引統計が収集されます。 RCT の場合、表は通常の表としてモデル化され、索引は機能ベースの索引としてモデル化されます。

範囲クラスター表の作成でオーバーフローが許可されている場合は、表内でのレコードの順序は保証されません。

シナリオ: 範囲クラスター表

以下のシナリオは、範囲クラスター表の作成方法を示す簡単な例です。これらは、表のキーとして単一列または複数列を使用する方法を示しています。加えて、データをオーバーフローできる表とデータをオーバーフローできない表の作成方法も示します。

シナリオ 1: 範囲クラスター表の作成

このシナリオは、STUDENT_ID を使用して生徒を探し出すときに使用する範囲クラスター表を示しています。各生徒のレコードには、以下の情報が含まれます。

- 学校 ID
- プログラム ID
- 生徒番号
- 生徒 ID
- 生徒のファーストネーム

- 生徒のラストネーム
- 生徒の成績平均点 (GPA)

この場合、生徒のレコードは、STUDENT_ID のみを基にしています。STUDENT_ID は、生徒のレコードの追加、更新、および削除に使用されます。

注: その他の索引は、別の時に別個に追加できます。ただし、この例の目的からして、表の編成と表のデータへのアクセス方法は、表の作成時に定義されます。

以下は、この表に必要な構文です。

```
CREATE TABLE STUDENTS
   (SCHOOL_ID      INT NOT NULL,
    PROGRAM_ID     INT NOT NULL,
    STUDENT_NUM    INT NOT NULL,
    STUDENT_ID     INT NOT NULL,
    FIRST_NAME     CHAR(30),
    LAST_NAME      CHAR(30),
    GPA            FLOAT)
   ORGANIZE BY KEY SEQUENCE
   (STUDENT_ID     STARTING FROM 1 ENDING AT 1000000)
    ALLOW OVERFLOW
;
```

各レコードのサイズは、列の合計です。この場合、10 バイトのヘッダー + 4 + 4 + 4 + 4 + 30 + 30 + 8 + 3 (ヌル可能列の場合) で、 97 バイトになります。ページ・サイズが 4 KB (つまり 4096 バイト) であれば、オーバーヘッドを計算した結果が 4038 バイトありますから、ページごとに 42 レコード分のスペースがあることになります。 100 万人の生徒のレコードを保管できるようにする場合は、 100 万をページ当たりのレコード数 42 で除算して、23809.5 ページが必要になります。つまり、切り上げて 23810 ページが必要、ということです。これに、表のオーバーヘッド分 4 ページと、エクステント・マッピング用の 3 ページが追加されます。結果として、4 KB のページを 23817 ページ事前割り振りすることが必要になります。 (エクステント・マッピングは、この表が 1 つのコンテナーに保持されることを前提としています。マッピング用のページは、コンテナーごとに 3 ページ必要です。)

シナリオ 2: 範囲クラスター表の作成 (オーバーフローが許可されない)

このシナリオは、1 つ目の例を少し変えたもので、教育委員会を念頭に置いています。教育委員会に属する学校は 200 あり、そのそれぞれには、 35 人の生徒を定員とする 20 のクラスルームがあります。これは、この教育委員会が最大 140,000 人の生徒に対応できる計算になります。

この場合は、生徒のレコードは 3 つの要素、つまり SCHOOL_ID、CLASS_ID、および STUDENT_NUM の値に基づいています。 これらの 3 つの列のそれぞれは、固有値を持ち、合わせて生徒のレコードの追加、更新、および削除に使用されます。

注: 1 つ目の例と同様、その他の索引は、別のときに別個に追加できます。

以下は、この表に必要な構文です。

```
CREATE TABLE STUDENTS
   (SCHOOL_ID      INT NOT NULL,
    CLASS_ID       INT NOT NULL,
    STUDENT_NUM    INT NOT NULL,
    STUDENT_ID     INT NOT NULL,
    FIRST_NAME     CHAR(30),
    LAST_NAME      CHAR(30),
    GPA            FLOAT)
   ORGANIZE BY KEY SEQUENCE
   (SCHOOL_ID      STARTING FROM 1 ENDING AT 200,
    CLASS_ID       STARTING FROM 1 ENDING AT 20,
    STUDENT_NUM    STARTING FROM 1 ENDING AT 35)
    DISALLOW OVERFLOW
;
```

この例では、オーバーフローは許可されません。これは、教育委員会で、方針として、各クラスの生徒数に制限を設けることが少なくないゆえに意味のある例です。このシナリオの場合、クラスのサイズは最大でも 35 です。この要素と、クラスルームや学校の数によって発生する物理的な限界を合わせて考えると、教育委員会に属する生徒の数にオーバーフローがありえないことは明らかです。

学校のクラスルームの数が変動することは、可能性としてありえることです。もしそのような場合には、クラスルーム数の範囲を定義する (CLASS_ID を使用) 際に、すべての学校を考慮して、クラスルームの最大数を上限に設定する必要があります。これは、比較的小さい学校 (大きな学校よりもクラスルームの数が少ない) にとっては、使用されることのない生徒レコード用のスペースが発生する可能性があることを意味します (ただし、例えば、仮設のクラスルームが学校に追加される場合を除く)。

1 つ目の例と同じ 4 KB のページ・サイズと同じ生徒レコードのサイズで計算すると、 1 ページあたりのレコード数として 42 が算出できます。 生徒のレコードが 140,000 ですから、3333.3 ページ、切り上げて 3334 ページが必要になります。なお、表情報用のページが 2 ページ、エクステント・マッピング用のページが 3 ページ必要です。結果として、4 KB のページを 3339 ページ事前割り振りすることが必要になります。

MDC 表を作成する際の考慮事項

MDC 表を作成する際には、さまざまな要素を考慮する必要があります。以下のセクションでは、 MDC 表を作成、配置、および使用する方法の決定に対して、現在使用しているデータベース環境 (例えばパーティション・データベースかどうかなど)、また MDC 表のためのディメンションの選択がどう影響するかについて説明します。さらに、DB2 設計アドバイザーと、それを使用して前述の問題に関するアドバイスを提供する方法についても説明します。

既存の表から MDC 表へのデータの移動

データウェアハウスまたは大規模データベース環境において、照会のパフォーマンスを向上させ、データ保守操作のオーバーヘッドを少なくするため、通常表のデータを、マルチディメンション・クラスタリング (MDC) 表に移動することができます。既存の表から MDC 表へデータを移動するには、データをエクスポートし、元の表をドロップし (オプション)、 (ORGANIZE BY DIMENSIONS 節を含む

CREATE TABLE ステートメントを使用して) マルチディメンション・クラスタリング (MDC) 表を作成し、その MDC 表にデータをロードします。

SYSPROC.ALTOBJ という ALTER TABLE プロシージャーを使用すると、既存の表から MDC 表へのデータ変換を実行できます。このプロシージャーは、DB2 設計アドバイザーから呼び出されます。表と表の間でのデータ変換にはかなりの時間が必要になる場合があり、それは表のサイズや変換の必要なデータの量によって異なります。

ALTOBJ プロシージャーは、表変更において次のことを実行します。

- 表の従属オブジェクトをすべてドロップします。
- 表の名前を変更します。
- 新しい定義を使用して表を作成します。
- 表の従属オブジェクトをすべて再作成します。
- 表に既存のデータを、新しい表で必要なデータに変換します。つまり、変換前の表のデータを選択し、そのデータを新しい表にロードします。その際には、変換前のデータ・タイプから変換後のデータ・タイプに変換するため、列関数が使用される場合があります。

SMS 表スペース内の MDC 表

MDC 表を SMS 表スペースに格納することを予定している場合には、複数ページ・ファイル割り振りを使用する必要があります (複数ページ・ファイル割り振りは、バージョン 8.2 以降で新たに作成されるデータベースではデフォルトです)。これは、 MDC 表は常にエクステントを単位として拡張されるため、それらのエクステント内のすべてのページが物理的に連続していることが重要になるからです。したがって、複数ページ・ファイル割り振りを無効にすることには、スペースに関するメリットは何もありません。それだけでなく、それを有効にすることにより、各エクステント内のページが物理的に連続する可能性が格段に大きくなります。

DB2 設計アドバイザーと比較した場合の MDC Advisor のフィーチャー

DB2 設計アドバイザー (db2advis) (旧称 Index Advisor) には、 MDC フィーチャーが含まれています。このフィーチャーでは、処理のパフォーマンスを向上させるため、基本列に対する低密度化を含め、MDC 表でクラスタリング・ディメンションを使用することを推奨します。*低密度化* という語は、クラスタリング・ディメンションのカーディナリティー (値の種類数) を少なくすることに関する数学用語です。低密度化の一般的な例としては、日付、曜日、月、四半期による低密度化があります。

DB2 設計アドバイザーの MDC フィーチャーを使用するには、データベース内に少なくとも複数のデータのエクステントがなければなりません。 DB2 設計アドバイザーはデータを使用して、データ密度とカーディナリティーのモデル化を行います。

データベースの表の中にデータがない場合、DB2 設計アドバイザーは MDC を推奨しません。たとえデータベースの表が空であり、しかしデータが取り込まれるデータベースを示すモックアップされた統計のセットを持つ場合でも同じです。

推奨事項には、ディメンションの低密度化を定義する潜在的な生成列を識別することが含まれています。推奨事項には、可能性のあるブロック・サイズは含まれていません。 MDC 表の推奨事項作成では、表スペースのエクステント・サイズが使用されます。推奨される MDC 表が既存の表と同じ表スペース内で作成され、したがってエクステント・サイズが同じであることが前提になっています。 MDC ディメンションに関する推奨事項は、表スペースのエクステント・サイズによって変わることがあります。というのは、エクステント・サイズは、 1 個のブロックまたはセルに入るレコード数に影響するからです。それはセルの密度に直接影響します。

表に対して単一ディメンションでも複数ディメンションでも推奨されることがありますが、複合列ディメンションではなく単一列ディメンションだけが考慮されます。 MDC フィーチャーは、結果となる MDC ソリューションでのセルのカーディナリティーを小さくすることを目標として、サポートされているほとんどのデータ・タイプに関して低密度化を推奨します。データ・タイプの例外には CHAR、VARCHAR、GRAPHIC、および VARGRAPH のデータ・タイプが含まれます。サポートされているデータ・タイプは、すべて INTEGER にキャストされ、生成される式によって低密度化されます。

DB2 設計アドバイザーの MDC フィーチャーの目標は、パフォーマンスが改善される MDC ソリューションを選択することです。第 2 の目標は、データベースのストレージ拡張を控え目なレベルに保つことです。表ごとの最大ストレージ拡張の判別には、統計的手法が使用されます。

Advisor 内の分析操作には、ブロック索引アクセスのメリットだけでなく、表のディメンションに対する挿入、更新、および削除操作における MDC の影響も含まれます。表に対するそれらのアクションには、セル間でレコードを移動させることになる可能性があります。また、分析操作は、特定の MDC ディメンションに関するデータの編成処理の結果としての表拡張による潜在的なパフォーマンスの影響をモデル化します。

MDC フィーチャーを有効にするには、 db2advis ユーティリティーで `-m <advise type>` フラグを使用します。マルチディメンション・クラスタリング表を指定するため、アドバイス・タイプとして「C」を使用します。アドバイス・タイプには、「I」(索引)、「M」(マテリアライズ照会表)、「C」(MDC)、および「P」(パーティション・データベース環境) があります。アドバイス・タイプは、互いに組み合わせて使用できます。

注: DB2 設計アドバイザーは、サイズが 12 エクステントより小さい表を検討しません。

Advisor は、推奨事項提供において MQT と通常の基本表の両方を分析します。

MDC フィーチャーからの出力には、次のものが含まれます。

- MDC ソリューションに出現する低密度化されたディメンションについて、表ごとに生成される列式。
- 各表について推奨される ORGANIZE BY 節。

推奨事項は、 stdout と、EXPLAIN 機能の一部である ADVISE 表の両方に報告されます。

MDC 表とパーティション・データベース環境

マルチディメンション・クラスタリングは、パーティション・データベース環境と組み合わせて使用することができます。実際、MDC はパーティション・データベース環境を補います。パーティション・データベース環境は、複数の物理または論理ノードの間に表データを配分させるために使用されます。その目的は、次のとおりです。

- 複数のマシンを利用して、並列処理される要求の数を増やす。
- 単一データベース・パーティションの限界を超えて、表の物理サイズを大きくする。
- データベースのスケーラビリティーを改善する。

表を配分する理由は、表が MDC 表か通常表かということとは別です。例えば、分散キーを構成する列を選択するための規則は同じです。 MDC 表の分散キーには、列がその表のディメンションの一部であるかどうかによらず、どんな列でも関係することがあります。

分散キーが表のディメンションと同一なら、各データベース・パーティションにはそれぞれ表の異なる部分が入れられます。例えば、この章で使用しているサンプルの MDC 表は、 2 つのデータベース・パーティションにわたって color により配分され、その後、Color 列を使用してデータが分割されます。その結果、Red スライスと Blue スライスが 1 つのデータベース・パーティションに、 Yellow スライスがもう 1 つのパーティションになることがあります。分散キーが表のディメンションと同一でない場合には、各データベース・パーティションには、それぞれ各スライスのデータのサブセットが入れられることになります。ディメンションを選択し、セル占有度を見積もる際には (『セルの密度』のセクションを参照)、平均として 1 セル当たりの合計データ量は、データ全体をデータベース・パーティション数で除算して得られることに注意してください。

マルチディメンションの MDC 表

照会で一部の特定の述部が頻繁に使用されることがあらかじめわかっている場合、ORGANIZE BY DIMENSIONS 節を使って、関連する列に基づいて表をクラスター化することができます。

例 1:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, c3, c4)
```

例 1 の表は、論理キューブを形成する 3 つの固有な列における値に基づいてクラスター化されています (つまり 3 ディメンションを持っている)。こうすれば、照会処理の際、 1 つまたは複数のディメンションにおいて表をスライスすることにより、適切なスライスまたはセルに含まれるブロックだけがリレーショナル演算子によって処理されるようにすることができます。ブロック・サイズ (ページ数) が、表のエクステント・サイズになることに注意してください。

複数列に基づくディメンションの MDC 表

それぞれのディメンションを、1 つまたは複数の列で構成することが可能です。一例として、2 つの列を含むディメンションに基づいてクラスター化される表を作成することができます。

例 2:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
   ORGANIZE BY DIMENSIONS (c1, (c3, c4))
```

例 2 では、c1 および (c3, c4) の 2 つのディメンションに基づいて表がクラスター化されます。こうすると、照会処理においては、ディメンション c1 もしくは複合ディメンション (c3, c4) に基づいて表を論理的にスライスすることができます。この表のブロック数は例 1 の表と同じですが、ディメンション・ブロック索引の数は 1 つ少なくなります。例 1 では、列 c1、c3、 c4 それぞれに関する 3 つのディメンション・ブロック索引が作成されます。例 2 の場合は、列 c1 に関するディメンション・ブロック索引と、列 c3 および c4 に関するディメンション・ブロック索引の 2 つが作成されます。この 2 つの方法の主な違いは、例 1 の場合、 c4 のみを扱う照会が c4 に関するディメンション・ブロック索引を使用して、関連するデータ・ブロックに素早く直接的にアクセスできることです。例 2 では c4 がディメンション・ブロック索引の 2 番目のキー部分であるため、 c4 のみを扱う照会はより多くの処理を行います。ただし、例 2 の場合、保守および保管されるブロック索引の数は 1 つ少なくなります。

DB2 設計アドバイザーは、複数列を含むディメンションに関する推奨事項を作成しません。

列式をディメンションとする MDC 表

ディメンションをクラスタリングする際、列式を使用することもできます。列式に基づくクラスター化は、細分性を粗くしてディメンションをロールアップする場合に便利です。例えば、住所を地理上の場所または地域にロールアップする場合や、日付を週、月、または年にロールアップする場合などです。このようなディメンションのロールアップを実装するには、生成された列を使用することができます。このタイプの列定義では、ディメンションを表す式を使って列を作成することができます。例 3 のステートメントによって作成される表は、基本となる 1 つの列、および 2 つの列式に基づいてクラスター化されます。

例 3:

```
CREATE TABLE T1(c1 DATE, c2 INT, c3 INT, c4 DOUBLE,
   c5 DOUBLE GENERATED ALWAYS AS (c3 + c4),
   c6 INT GENERATED ALWAYS AS (MONTH(C1)))
      ORGANIZE BY DIMENSIONS (c2, c5, c6)
```

例 3 では、列 c5 が c3 および c4 に基づく式であり、列 c6 は列 c1 を時間的に粗い細分性にロールアップします。このステートメントによって、列 c2、c5、および c6 の値に基づいて表がクラスター化されます。

生成列ディメンションに対する範囲照会

生成列ディメンションに対する範囲照会では単調列関数が必要です。生成列に対してディメンションの範囲述部を派生させるには、式が単調でなければなりません。

生成列に対してディメンションを作成する場合、基本列に対する照会では、その生成列に対するブロック索引を利用することによりパフォーマンスが改善されます。ただし、1 つの例外があります。基本列 (日付など) に対する範囲照会で、ディメンション・ブロック索引による範囲スキャンを使用するには、 CREATE TABLE ステートメントで列を生成するために使用する式が単調でなければなりません。列式には任意の有効な式 (ユーザー定義関数 (UDF) を含む) を含めることができますが、その式が単調でない場合、基本列に対する述部のうち、照会を満たすためにブロック索引を使用できるのは等式述部または IN 述部だけです。

例えば、month = INTEGER (date)/100 という生成列に対してディメンションが定義されている MDC があるとしましょう。ディメンション (month) に対する照会では、ブロック索引スキャンが可能です。基本列 (date) に対する照会でも、ブロック索引スキャンを実行することによってスキャン対象のブロックを限定してから、それらのブロックだけに含まれる行に対して、date に関する述部を適用することができます。

コンパイラーは、ブロック索引スキャンで使用する付加的な述部を生成します。例えば、次の照会の場合、

```
SELECT * FROM MDCTABLE WHERE DATE > '1999-03-03' AND DATE < '2000-01-15'
```

コンパイラーは、『month >= 199903』 かつ 『month <= 200001』 という、付加的な述部を生成します。これは、ディメンション・ブロック索引スキャンの述部として使用できます。結果ブロックのスキャンにおいては、オリジナルの述部がブロック中の行に適用されます。

非単調式の場合、そのディメンションに対して適用できるのは等式述部だけです。非単調関数の 1 つの例として、例 3 の列 c6 の定義に出てきた MONTH( ) があります。列 c1 が日付、タイム・スタンプ、または日付やタイム・スタンプを表す有効なストリング表記である場合、この関数は 1 から 12 までの範囲の整数値を戻します。この関数の出力は決まりきったものですが、実際には以下のようなステップ関数と同じような出力 (つまり循環パターン) を生成します。

```
MONTH(date('01/05/1999')) = 1
MONTH(date('02/08/1999')) = 2
MONTH(date('03/24/1999')) = 3
MONTH(date('04/30/1999')) = 4
...
MONTH(date('12/09/1999')) = 12
MONTH(date('01/18/2000')) = 1
MONTH(date('02/24/2000')) = 2
...
```

この例の一連の日付は単調に大きくなっていますが、 MONTH(date) はそうではありません。より厳密に言うと、date1 が date2 よりも大きいとき、常に MONTH(date1) が MONTH(date2) より大きいか等しくなるとは限りません。単調性を考慮する必要があるのは、このような場合です。このような非単調性は許可されていますが、基本となる列の範囲述部がそのディメンションの範囲述部を生成できないという点で、ディメンションを制限してしまいます。しかし、式の範囲述部 (例えば where month(c1) between 4 and 6) は使用できます。こうすれば、開始キーを 4、終了キーを 6 として、ディメンションに関する索引を通常の方法で使用できます。

この関数を単調にするには、月の上位部として年を使用する必要があります。バージョン 9.2 の組み込み関数 INTEGER には、日付に関する単調式を定義するのに役立つ拡張機能があります。 INTEGER(date) は日付の整数表記を戻します。この表記をさらに分割して、年および月を表す表記を検出することができます。例えば、`INTEGER(date('2000/05/24'))` は `20000524` を戻し、 `INTEGER(date('2000/05/24'))/100 = 200005` となります。関数 INTEGER(date)/100 は単調です。

同様に、組み込み関数 DECIMAL および BIGINT にもまた、単調関数を導出できる拡張機能があります。 DECIMAL(timestamp) はタイム・スタンプの 10 進表記を戻します。この表記を単調式で使用して、月、日、時間、分などに関して単調増加する値を導出することができます。 BIGINT(date) は INTEGER(date) と同様に、日付に関する BIGINT 表記を戻します。

表において生成された列を作成するとき、あるいはディメンション節の式からディメンションを作成するときに、データベース・マネージャーは可能な限り式の単調性を判別します。 DATENUM( )、 DAYS( )、 YEAR( ) などの一部の関数は、単調性を保持する関数として認識されます。さらに、さまざまな数式 (例えば列や定数の除算、乗算、加算) が単調性を保持します。式の単調性が保持されないと DB2 が判断した場合、あるいは判別が不可能な場合には、ディメンションの基本列では等式述部のみを使用することができます。

第 12 章 データベースを変更する

インスタンスを変更する

複数のパーティションでのデータベース・パーティション構成の変更

複数のデータベース・パーティションにわたって配分されたデータベースを持っている場合、データベース構成ファイルは、すべてのデータベース・パーティションで同じものである必要があります。

SQL コンパイラーは、ノードの構成ファイルの情報に基づいて配分 SQL ステートメントをコンパイルし、 SQL ステートメントのニーズを満足させるためのアクセス・プランを作成するので、これらの構成ファイルには整合性が必要です。データベース・パーティションごとに異なる構成ファイルを維持していると、どのデータベース・パーティションでステートメントが準備されたかによって、異なるアクセス・プランが作成される可能性があります。 db2_all を使用して、すべてのデータベース・パーティションで構成ファイルを保守してください。

データベースを変更する

データベース・パーティション・グループの変更

コマンド行プロセッサーを使用してデータベース・パーティション・グループを変更するには、 REDISTRIBUTE DATABASE PARTITION GROUP コマンドを使用します。データベース・パーティションを追加またはドロップしたら、データベース・パーティション・グループ内の新しいデータベース・パーティションのセットにわたって、現行データを再配分しなければなりません。

コントロール・センターからデータベース・パーティションを管理する

コントロール・センターの「データベース・パーティション」ビューを使用して、データベース・パーティションを処理できます。

データベース・パーティションを処理するには、インスタンスにアタッチするための権限が必要です。SYSADM または DBADM 権限がある人から、特定のインスタンスにアクセスするための権限を付与してもらうことができます。

データベース・パーティションを構成する、またはデータベース・パーティションをロールフォワード・ペンディング状態から回復するには、SYSADM、SYSCTRL、または SYSMAINT 権限が必要です。

「データベース・パーティション」ビューを使用して、データベース・パーティションの再始動、データベース・パーティションのロールフォワード・ペンディング状態からの回復、データベース・パーティションのバックアップ、データベース・

パーティションのリストア、または構成アドバイザーを使用したデータベース・パーティションの構成を行うことができます。

コントロール・センターから「データベース・パーティション」ビューをオープンするには、次のようにします。

1. コントロール・センターで、データベース・パーティションを表示するパーティション・データベースが表示されるまでオブジェクト・ツリーを展開します。
2. 目的のパーティション・データベースを右クリックし、メニュー・リストから「データベース・パーティションのオープン」を選択します。
3. 選択されたパーティション・データベースについての「データベース・パーティション」ビューがオープンします。

データベース・パーティションを構成するには、以下のようにします。

1. 「データベース・パーティション」ビューから目的のデータベース・パーティションを選択します。
2. リストから「データベース・パーティション」を選択し、「構成アドバイザー」を右クリックして選択します。
3. 構成アドバイザーがオープンします。構成アドバイザーを使用して、データベース構成パラメーターの値を指定します。

第 13 章 表およびその他の関連する表オブジェクトを変更する

パーティション表の変更

パーティション表では ALTER TABLE ステートメントに関連したすべての節がサポートされます。さらに、ALTER TABLE ステートメントを使用すると、新規データ・パーティションの ADD、新規データ・パーティションのロールイン (ATTACH)、および既存のデータ・パーティションのロールアウト (DETACH) が可能になります。

データ・パーティションをデタッチする ALTER をパーティション表に対して行うには、ユーザーに次の権限または特権が必要です。

- DETACH 操作を実行するユーザーには、ソース表に対する ALTER、SELECT、および DELETE に必要な権限が必要です。
- ユーザーは、ターゲット表を作成するために必要な権限も持っていなければなりません。したがって、データ・パーティションをデタッチする ALTER TABLE を行うには、ステートメントの許可 ID によって保持される特権に、ターゲット表に対する以下の権限または特権が少なくとも 1 つ含まれていなければなりません。
  - SYSADM または DBADM 権限
  - データベースに対する CREATETAB 権限と表により使用される表スペースに対する USE 特権、および次のいずれか:
    - データベースに対する IMPLICIT_SCHEMA 権限 (表の暗黙または明示スキーマ名がない場合)
    - スキーマに対する CREATEIN 特権 (表のスキーマ名が既存のスキーマを指す場合)

データ・パーティションを追加する ALTER をパーティション表に行うには、ステートメントの許可 ID によって保持される特権に、ソース表に対する以下の権限または特権が少なくとも 1 つ含まれていなければなりません。

- ソース表に対する SELECT 特権およびソース表のスキーマに対する DROPIN 特権
- ソース表に対する CONTROL 特権
- SYSADM または DBADM 権限

データ・パーティションを追加する ALTER をパーティション表に行うには、ステートメントの許可 ID によって保持される特権に新規パーティションが追加される表スペースを使用するための特権があり、さらにソース表に対する以下の権限または特権が少なくとも 1 つ含まれていなければなりません。

- ALTER 特権
- CONTROL 特権
- SYSADM
- DBADM

- 表スキーマの ALTERIN 特権

使用のガイドライン

- PARTITION 節を指定して発行する ALTER TABLE ステートメントはそれぞれ別個の SQL ステートメントになければなりません。
- ALTER TABLE...PARTITION 操作を含む SQL ステートメントでは、これ以外の ALTER 操作は許可されていません。例えば、一度の SQL ステートメントでデータ・パーティションのアタッチと表への列の追加を行うことはできません。
- 複数の ALTER ステートメントを実行した後、SET INTEGRITY ステートメントを一度実行することができます。

コマンド行からパーティション表を変更するには、ALTER TABLE ステートメントを発行します。

パーティション表の変更に関するガイドラインと制約事項

このトピックでは、最も一般的な表変更アクション、およびアタッチされたデータ・パーティションやデタッチされたデータ・パーティションが存在する場合の特別な考慮事項を示します。

チェック制約または外部キー制約の追加または変更

アタッチされたデータ・パーティションおよびデタッチされたデータ・パーティションに関するチェック制約または外部キー制約の追加がサポートされています。

列の追加

アタッチされたデータ・パーティションを持つ表に列を追加すると、アタッチされたデータ・パーティションにもその列が追加されます。デタッチされたデータ・パーティションを持つ表に列を追加しても、その列は、デタッチされたデータ・パーティションには追加されません。デタッチされたデータ・パーティションとその表との物理的関連がなくなっているからです。

列の変更

アタッチされたデータ・パーティションを持つ表の列を変更すると、アタッチされたデータ・パーティションのその列も変更されます。デタッチされたデータ・パーティションを持つ表の列を変更しても、デタッチされたデータ・パーティションのその列は変更されません。デタッチされたデータ・パーティションとその表との物理的関連がなくなっているからです。

生成列の追加

アタッチまたはデタッチされたデータ・パーティションを持つパーティション表に生成列を追加するときは、他のタイプの列を追加する場合の規則を守る必要があります。

索引の追加または変更

アタッチされたデータ・パーティションを持つ表に索引の作成、再作成、または再編成を行っても、その索引にはアタッチされたデータ・パーティション内のデータは含まれません。SET INTEGRITY ステートメントが、アタッチされたすべてのデータ・パーティションについて、すべての索引を保守するからです。デタッチされたデータ・パーティションを持つ表に索引の作成、再作成、または再編成を行っても、その索引にはデタッチされたデー

タ・パーティション内のデータは含まれません。ただし、デタッチされたデータ・パーティションが、データ・パーティションに関連して増分リフレッシュを行う必要があるデタッチされた従属物またはステージング表を持っている場合は別です。この場合、索引には、そのデタッチされたデータ・パーティションのデータが含まれます。

WITH EMPTY TABLE
デタッチされたデータ・パーティションを持つ表を空にすることはできません。

ADD MATERIALIZED QUERY AS
アタッチされたデータ・パーティションを持つ表を MQT に変更することは許可されません。

データ・パーティションに保管される追加の表属性の変更
データ・パーティションには、以下の表属性も保管されます。これらの属性に対する変更は、アタッチされたデータ・パーティションには反映されますが、デタッチされたデータ・パーティションには反映されません。

- DATA CAPTURE
- VALUE COMPRESSION
- APPEND
- COMPACT/LOGGED FOR LOB COLUMNS
- ACTIVATE NOT LOGGED INITIALLY (WITH EMPTY TABLE)

データ・パーティションのアタッチ

表パーティション化は、表データの効率的なロールインおよびロールアウトを可能にします。この効果は、ALTER TABLE ステートメントの ATTACH PARTITION 節および DETACH PARTITION 節の使用により実現されます。パーティション表データをロールインすることによって、新しい範囲を追加のデータ・パーティションとして簡単にパーティション表に取り込むことができます。

データ・パーティションをアタッチする ALTER TABLE を行うには、ステートメントの許可 ID によって保持される特権に、ソース表に対する以下の権限および特権が少なくとも 1 つ含まれていなければなりません。

- ソース表に対する SELECT 特権およびソース表のスキーマに対する DROPIN 特権
- ソース表に対する CONTROL 特権
- SYSADM または DBADM 権限

ATTACH PARTITION 節は既存の表 (ソース表) を取り、それを新規のデータ・パーティションとしてターゲット表にアタッチします。新しくアタッチされたデータ・パーティションには最初、照会のためにアクセスすることができません。表の残りはオンラインのままとなります。アタッチされたデータ・パーティションをオンラインにするには、SET INTEGRITY ステートメントの呼び出しが必要です。

制約事項および使用ガイドライン

データ・パーティションをアタッチする前に、以下の条件を満たさなければなりません。

- 新規のデータ・パーティションをアタッチする表 (つまりターゲット表) が既存のパーティション表に存在していなければなりません。
- ソース表は、既存の非パーティション表であるか、またはデータ・パーティションを 1 つのみ持ち、ATTACHED または DETACHED データ・パーティションを持たないパーティション表でなければなりません。複数のデータ・パーティションをアタッチするには、複数の ATTACH ステートメントを発行する必要があります。
- ソース表は階層 (型付き表) になることができません。
- ソース表は範囲クラスター表 (RCT) になることができません。
- ソース表の表定義はターゲット表と一致していなければなりません。
- 列の番号、タイプ、および順序は、ソース表とターゲット表で一致していなければなりません。
- デフォルト値を含むかどうかについて、両方の表の列で一致していなければなりません。 ALTER TABLE ADD COLUMN を使ってソース列が作成される場合、つまり、SYSCOLUMNS.ADD_DEFAULT = 'Y' となる場合、existDefault 値 (SYSCOLUMNS.ADDED_DEFAULT) はターゲット列のものと一致していなければなりません。
- NULL を許可するかどうかについて、両方の表の列で一致していなければなりません。
- VALUE COMPRESSION 値および SYSTEM COMPRESSION DEFAULT 値の両方を含む圧縮節は、ソース表とターゲット表で一致していなければなりません。
- データ・キャプチャー・オプションを指定した APPEND 節と、NOT LOGGED INITIALLY オプションの使用が一致していなければなりません。
- ターゲット列が生成列であり、ソース列がそうでない場合でも、データ・パーティションのアタッチは可能です。このステートメント SET INTEGRITY FOR T ALLOW WRITE ACCESS IMMEDIATE CHECKED FORCE GENERATED は、アタッチ行の生成列の値を生成します。生成列と一致する列は、タイプと NULL 可能性が一致しなければなりません。この列はデフォルト値を必要としません。推奨されるアプローチは、ATTACH のソース表が生成列に適切な生成値を必ず持つようにするというものです。その場合、FORCE GENERATED オプションを使用する必要はありません。次のステートメントを使用できます。

  ```
  SET INTEGRITY FOR T GENERATED COLUMN IMMEDIATE UNCHECKED
  (生成列の検査を迂回するよう、システムに指示する)
  SET INTEGRITY FOR T ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR EXCEPTION
  IN T USE T_EX (アタッチされたパーティションの整合性検査は実行するが、
  生成列の正確さは検査しない)
  ```

- ターゲット列が ID であり、ソース列がそうでない場合でも、データ・パーティションのアタッチは可能です。ステートメント SET INTEGRITY IMMEDIATE CHECKED は、アタッチ行に ID 値を生成しません。ステートメント SET INTEGRITY FOR T GENERATE IDENTITY ALLOW WRITE ACCESS IMMEDIATE CHECKED は、アタッチ行に ID 値を入力します。 ID 列と一致する列は、タイプと NULL 可能性が一致しなければなりません。この列はデフォルト値を必要としません。推奨されるのは、ステージング表に正しい ID 値を入力

するというアプローチです。 ATTACH の後 GENERATE IDENTITY オプションを使用する必要はありません。 ID 値はすでにソース表で保証されているからです。

- データがデータベース・パーティション全体に分散される表の場合、ソース表も、同じ分散キーおよび同じ分散マップを使って同じデータベース・パーティション・グループ内に分散する必要があります。
- ソース表はドロップ可能でなければなりません (つまり RESTRICT DROP セットを持つことができません)。
- DATAPARTITIONNAME が指定される場合、それがターゲット表に存在していてはなりません。
- ターゲット表がマルチディメンション・クラスタリング (MDC) 表の場合、ソース表も MDC 表でなければなりません。
- ソース表のデータ表スペースとターゲット表のデータ表スペースは、タイプ (つまり DMS か SMS か)、pageSize、extentSize、およびデータベース・パーティション・グループが一致していなければなりません。 prefetchSize が一致しないと、ユーザーに警告が戻されます。ソース表の LARGE 表スペースとターゲット表の LARGE 表スペースは、タイプ、データベース・パーティション・グループ、および pageSize が一致している必要があります。

DB2 コマンド行を使用してパーティション表を変更し、その表にデータ・パーティションをアタッチするには、ALTER TABLE ステートメントを発行します。

パーティション表へのデータ・パーティションのアタッチに関するガイドライン

このトピックでは、ALTER TABLE ...ATTACH PARTITION ステートメントを発行してデータ・パーティションをパーティション表にアタッチする際に生じる可能性のある、さまざまなタイプの不一致を訂正する方法に関する指針が提供されています。ターゲット表の特性と一致するようにソース表を変更するか、またはソース表の特性と一致するようにターゲット表を変更することにより、表間の一致を達成できます。

ソース表とは、ターゲット表にアタッチする既存の表のことです。ターゲット表とは、新しいデータ・パーティションのアタッチ先となる表です。

アタッチを成功裏に実行するために提案されている 1 つの方法として、ターゲット表に対して実行したのと同じように CREATE TABLE ステートメント (ただし PARTITION BY 節は使用しない) をソース表に使用します。互換性のために、ソース表またはターゲット表のいずれかの特性を変更するのが困難な場合には、ターゲット表と互換性のある新しいソース表を作成できます。新しいソースの作成に関する詳細は、『*既存の表の類似表の作成*』を参照してください。

不一致が生じるのを避けるため、225 ページの『データ・パーティションのアタッチ』の「制約事項および使用ガイドライン」のセクションを参照してください。そのセクションでは、データ・パーティションを正常にアタッチするために事前に満たしておく必要のある条件について略述されています。リストされている条件を満たせないと、エラー SQL20408N または SQL20307N が戻ります。

以降のセクションでは、生じ得るさまざまなタイプの不一致について説明し、表間での一致を達成するために提案されているステップが提供されています。

(VALUE) COMPRESSION 節 (SYSCAT.TABLES の COMPRESSION 列) が一致しない。 (SQL20307N 理由コード 2)

値圧縮の一致を達成するには、以下のいずれかのステートメントを使用してください。

```
ALTER TABLE... ACTIVATE VALUE COMPRESSION
または
ALTER TABLE... DEACTIVATE VALUE COMPRESSION
```

行圧縮の一致を達成するには、以下のいずれかのステートメントを使用してください。

```
ALTER TABLE... COMPRESS YES
または
ALTER TABLE... COMPRESS NO
```

表の APPEND モードが一致しない。 (SQL20307N 理由コード 3)

付加モードの一致を達成するには、以下のいずれかのステートメントを使用してください。

```
ALTER TABLE ... APPEND ON
または
ALTER TABLE ... APPEND OFF
```

ソースおよびターゲット表のコード・ページが一致しない。 (SQL20307N 理由コード 4)

新しいソースを作成してください。

ソース表が、複数のデータ・パーティション、またはアタッチあるいはデタッチされたデータ・パーティションのあるパーティション表である。 (SQL20307N 理由コード 5)

以下のステートメントを使用して、1 つの表示可能なデータ・パーティションが存在するようになるまで、ソース表からデータ・パーティションをデタッチします。

```
ALTER TABLE ... DETACH PARTITION
```

必要な SET INTEGRITY ステートメントを組み込みます。ソース表に索引がある場合、ソース表をすぐにアタッチできない可能性があります。デタッチされたキーをすべての索引でクリーンアップするまでは、デタッチされたデータ・パーティションはデタッチされたままの状態になります。すぐにアタッチを実行する場合には、ソース表の索引をドロップしてください。そうしない場合には、新しいソースを作成してください。

ソース表が、システム表、ビュー、型付き表、表 ORGANIZED BY KEY SEQUENCE、または宣言済みグローバル一時表である。 (SQL20307N 理由コード 6)

新しいソースを作成してください。

ターゲット表とソース表が同一である。 (SQL20307N 理由コード 7)

表を自分自身にアタッチすることはできません。ソース表またはターゲット表として使用する正しい表を判別してください。

ソース表またはターゲット表のいずれかに対して NOT LOGGED INITIALLY 節が指定されたが、両方には指定されていない。 (SQL20307N 理由コード 8)

NOT LOGGED INITIALLY 属性の表の方を COMMIT ステートメントを発行してログに記録されるようにするか、以下のステートメントを入力してログに記録されている表を最初はログに記録されないようにします。

```
ALTER TABLE .... ACTIVATE NOT LOGGED INITIALLY
```

ソース表またはターゲット表のいずれかに対して DATA CAPTURE CHANGES 節が指定されたが、両方には指定されていない。 (SQL20307N 理由コード 9)

データ・キャプチャー変更をデータ・キャプチャー変更がオンになっていない表で使用可能にするには、次のステートメントを実行してください。

```
ALTER TABLE ... DATA CAPTURE CHANGES
```

データ・キャプチャー変更をデータ・キャプチャー変更がオンにされている表で使用不可にするには、次のステートメントを実行してください。

```
ALTER TABLE ... DATA CAPTURE NONE
```

表の配分節が一致しない。ソース表とターゲット表の分散キーを同じにすることが必要。 (SQL20307N 理由コード 10)

新しいソース表を作成することをお勧めします。複数のデータベース・パーティションにまたがる表の分散キーを変更することはできません。単一パーティションのデータベース内の表にある分散キーを変更するには、以下のステートメントを実行します。

```
ALTER TABLE ... DROP DISTRIBUTION;
    ALTER TABLE ... ADD DISTRIBUTION(key-specification)
```

1 つの表だけに ORGANIZE BY DIMENSIONS 節が指定されているか、編成ディメンションが異なる。 (SQL20307N 理由コード 11)

新しいソースを作成してください。

列のデータ・タイプ (TYPENAME) が一致しない。 (SQL20408N 理由コード 1)

データ・タイプの不一致を訂正するには、以下のステートメントを発行します。

```
ALTER TABLE ... ALTER COLUMN ... SET DATA TYPE...
```

列の NULL 可能性 (NULLS) が一致しない。 (SQL20408N 理由コード 2)

表のいずれかと一致しない列の NULL 可能性を変更するには、以下のいずれかのステートメントを発行してください。

```
ALTER TABLE... ALTER COLUMN... DROP NOT NULL
または
ALTER TABLE... ALTER COLUMN... SET NOT NULL
```

列の暗黙的なデフォルト値 (SYSCAT.COLUMNS IMPLICITVALUE) に互換性がない。 (SQL20408N 理由コード 3)

新しいソース表を作成してください。ターゲット表列とソース表列の両方が暗黙的なデフォルトを持っている場合 (IMPLICITVALUE が NULL でない場合)、その暗黙的なデフォルトは正確に一致していなければなりません。

ターゲット表内のある列に対して IMPLICITVALUE が NULL ではなく、ソース表内の対応する列に対して IMPLICITVALUE が NULL ではない場合、各列は、その表に対する元の CREATE TABLE ステートメントの後に追加されたものです。この場合、この列に対して IMPLICITVALUE に保管されている値は一致していなければなりません。

V9.1 前の表からのマイグレーション、または V9.1 前の表からのデータ・パーティションのアタッチが行われた際に、列が元の CREATE TABLE ステートメント後に追加されたものかどうかをシステムが認知しなかったため、IMPLICITVALUE が NULL でない場合があります。データベースの列が追加列であるかどうかが不確かな場合には追加列として扱われます。追加列とは、ALTER TABLE ...ADD COLUMN ステートメントの結果として作成される列のことです。この場合、アタッチの続行が許可されると列の値が破壊される可能性があるため、このステートメントは許可されません。データはソース表から新規表 (この列に対する IMPLICITVALUE は NULL) にコピーし、アタッチ操作には新規表をソース表として使用しなければなりません。

列のコード・ページ (COMPOSITE_CODEPAGE) が一致しない。 (SQL20408N 理由コード 4)

新しいソース表を作成してください。

システム圧縮のデフォルト節 (COMPRESS) が一致しない。 (SQL20408N 理由コード 5)

列のシステム圧縮を変更するには、以下のステートメントのいずれかを発行して不一致を訂正します。

```
ALTER TABLE ... ALTER COLUMN ...  COMPRESS SYSTEM DEFAULT
または
ALTER TABLE ... ALTER COLUMN ... COMPRESS OFF
```

データ・パーティションのデタッチ

表パーティション化は、表データの効率的なロールインおよびロールアウトを可能にします。この効果は、ALTER TABLE ステートメントの ATTACH PARTITION 節および DETACH PARTITION 節の使用により実現されます。

パーティション表からデータ・パーティションをデタッチするには、次の権限または特権が必要です。

- DETACH 操作を実行するユーザーには、ソース表に対する ALTER、SELECT、および DELETE に必要な権限が必要です。
- ユーザーは、ターゲット表を作成するために必要な権限も持っていなければなりません。したがって、データ・パーティションをデタッチする ALTER TABLE を行うには、ステートメントの許可 ID によって保持される特権に、ターゲット表に対する以下の権限または特権が少なくとも 1 つ含まれていなければなりません。

– SYSADM または DBADM 権限
– データベースに対する CREATETAB 権限と表により使用される表スペースに対する USE 特権、および次のいずれか:
  - データベースに対する IMPLICIT_SCHEMA 権限 (表の暗黙または明示スキーマ名がない場合)
  - スキーマに対する CREATEIN 特権 (表のスキーマ名が既存のスキーマを指す場合)

注: データ・パーティションをデタッチするとき、ステートメントの許可 ID は事実上 CREATE TABLE ステートメントを実行しようとするため、その操作の実行に必要な特権が必要になります。 ALTER TABLE ステートメントの許可 ID は、ユーザーが CREATE TABLE ステートメントを発行したかのように、CONTROL 権限を持つ新規表の定義者になります。変更されている表の権限はいずれも新規表には転送されません。 ALTER TABLE ... DETACH PARTITION ステートメントの直後にデータにアクセスできるのは、ALTER TABLE ステートメントの許可 ID、および DBADM または SYSADM のみです。

パーティション表データのロールアウトによって、データの範囲を簡単にパーティション表から分けることができます。データ・パーティションがデタッチされて個々の表に入れられると、その表をさまざまな方法で扱えるようになります。例えば、個々の表をドロップする (これを行うとデータ・パーティションにあるデータは破棄される)、表をアーカイブするか、そうしない場合はそれを個々の表として使用する、表を他のパーティション表、例えば履歴表にアタッチする、または表を操作、クレンジング、トランスフォームする、あるいは元の表か他の何らかのパーティション表に再びアタッチする、などを行えます。

制約事項

ソース表がマルチディメンション・クラスタリング (MDC) 表である場合、ALTER TABLE ...DETACH 操作と同じ作業単位では、新規にデタッチされた表へのアクセスは許可されていません。ブロック索引は、ALTER TABLE ...DETACH 操作がコミットされた後に、表に最初にアクセスした際に作成されます。ブロック索引が作成されている間はアクセス時間が短くなります。

DETACH 操作を実行する前に、以下の条件を満たさなければなりません。

- (ソース表から) デタッチされる表が存在し、それはパーティション表でなければなりません。
- デタッチされるデータ・パーティションはソース表に存在していなければなりません。
- ソース表には複数のデータ・パーティションがなければなりません。パーティション表には少なくとも 1 つのデータ・パーティションがなければなりません。可視のアタッチされたデータ・パーティションのみがこのコンテキストに関係します。アタッチされたデータ・パーティションとは、アタッチされているものの、まだ SET INTEGRITY ステートメントによって妥当性検査されていないデータ・パーティションのことです。
- DETACH 操作によって作成される表 (ターゲット表) の名前が存在していてはなりません。

- 施行される参照整合性 (RI) のリレーションシップの親である表では DETACH が許可されません。
- デタッチされるデータ・パーティションに関して増分的に保守していく必要のある従属表がある場合 (これらの従属表は、デタッチされた従属表と呼ばれる)、新たにデタッチされた表には最初アクセスすることができません。
  SYSCAT.TABLES カタログ・ビューの TYPE 列の表に L とマークされます。これはデタッチされた表と呼ばれます。これにより、デタッチされた従属表を増分的に保守していくために SET INTEGRITY ステートメントを実行するまで、表に対する読み取り、変更、またはドロップを防ぐことができます。デタッチされたすべての従属表に対して SET INTEGRITY ステートメントが実行された後、デタッチされた表は通常の表に移行され、完全にアクセス可能になります。

パーティション表を変更し、その表からデータ・パーティションをデタッチするには、コマンド行から、DETACH PARTITION 節を指定して ALTER TABLE ステートメントを発行します。

デタッチされたデータ・パーティションの属性

ALTER TABLE ステートメントの DETACH PARTITION 節を使用して、パーティション化された表からデータ・パーティションをデタッチする場合、これはスタンドアロンのターゲット表になります。新規のターゲット表の属性の多くは、ソース表から継承されます。ソース表から継承されない属性はすべて、DETACH 操作を実行中のユーザーがターゲット表を作成するかのように設定されます。

注: デタッチされた従属が存在する場合、デタッチされるデータ・パーティションは、デタッチ時にスタンドアロン表になりません。この場合、デタッチを完了して表にアクセスできるようにするには、SET INTEGRITY ステートメントを発行する必要があります。

ターゲット表によって継承される属性

ターゲット表によって継承される属性には以下のものがあります。

- 以下の列定義が継承されます。
  - 列名
  - データ・タイプ (CHAR および DECIMAL など、長さと精度を持つタイプの場合は、その長さおよび精度を含む)
  - NULL 可能性
  - 列のデフォルト値
  - コード・ページ (SYSCAT.COLUMNS カタログ・ビューの CODEPAGE 列)
  - LOB のロギング (SYSCAT.COLUMNS カタログ・ビューの LOGGED 列)
  - LOB の圧縮 (SYSCAT.COLUMNS カタログ・ビューの COMPACT 列)
  - 圧縮 (SYSCAT.COLUMNS カタログ・ビューの COMPRESS 列)
  - 隠し列のタイプ (SYSCAT.COLUMNS カタログ・ビューの HIDDEN 列)
  - 列の順序

- ソース表がマルチディメンション・クラスタリング表 (MDC) である場合、ターゲット表も同じディメンション列で定義される MDC 表になります。ソース表が MDC である場合、デタッチと同じ作業単位では、新規にデタッチされた表へのアクセスは許可されていません。
- ブロック索引の定義。DETACH 操作がコミットされた後、新規にデタッチされた独立表への最初のアクセスで、索引は再作成されます。
- 表スペース ID および表オブジェクト ID は、ソース表からではなくデータ・パーティションから継承されます。これは、DETACH 操作時には表データが移動されないためです。カタログに関しては、ソース・データ・パーティションの SYSCAT.DATAPARTITIONS カタログ・ビューの TBSPACEID 列が、SYSCAT.TABLES カタログ・ビューの TBSPACEID 列になります。表スペース名に変換されると、これはターゲット表の SYSCAT.TABLES カタログ・ビューの TBSPACE 列になります。ソース・データ・パーティションの SYSCAT.DATAPARTITIONS カタログ・ビューの PARTITIONOBJECTID 列は、ターゲット表の SYSCAT.TABLES カタログ・ビューの TABLEID 列になります。
- ソース・データ・パーティションの SYSCAT.DATAPARTITIONS カタログ・ビューの LONG_TBSPACEID 列は、表スペース名に変換され、ターゲット表の SYSCAT.TABLES の LONG_TBSPACE 列になります。
- 表スペースの場所
- 複数パーティション・データベース用の分散マップの ID (SYSCAT.TABLES カタログ・ビューの PMAP_ID 列)
- 空きパーセント (SYSCAT.TABLES カタログ・ビューの PCTFREE 列)
- 追加モード (SYSCAT.TABLES カタログ・ビューの APPEND_MODE 列)
- 優先されるロック細分性 (SYSCAT.TABLES カタログ・ビューの LOCKSIZE 列)
- データ・キャプチャー (SYSCAT.TABLES カタログ・ビューの DATA_CAPTURE 列)
- VOLATILE (SYSCAT.TABLES カタログ・ビューの VOLATILE 列)
- DROPRULE (SYSCAT.TABLES カタログ・ビューの DROPRULE 列)
- 圧縮 (SYSCAT.TABLES カタログ・ビューの COMPRESSION 列)
- 最大フリー・スペースの検索 (SYSCAT.TABLES カタログ・ビューの MAXFREESPACESEARCH 列)

注: パーティション化された階層または一時表、範囲クラスター表、およびパーティション化されたビューはサポートされません。

ソース表から継承されない属性

ソース表から継承されない属性には以下のものがあります。

- ターゲット表タイプは継承されません。ターゲット表は常に regular 表になります。
- 特権および権限
- スキーマ

- 生成列、ID 列、チェック制約、参照制約。ソース列が生成列または ID 列である場合、対応するターゲット列は明示的なデフォルト値を持ちません。つまり、デフォルト値は NULL になります。
- 索引表スペース (SYSCAT.TABLES カタログ・ビューの INDEX_TBSPACE 列)。値は NULL に設定されます。DETACH の結果の表の索引は、表と同じ表スペースに配置されます。
- トリガー
- 主キー
- 統計
- 属性のリストに記述されていない他の属性はすべて、ソース表から明示的に継承されます。

パーティション表へのデータ・パーティションの追加

表の作成後、ALTER TABLE ステートメントを使用してパーティション表を変更することができます。特に、ADD PARTITION 節を使用して、既存のパーティション表に新規データ・パーティションを追加することができます。データがデータ・パーティションに次第に追加されていくというケースで、データが外部ソースから一度に大量に入ってくるのではなく少しずつ入って来るような場合、またはデータを直接パーティション表に挿入またはロードしている場合には、データ・パーティションをアタッチするよりも、パーティション表にデータ・パーティションを追加する方が適しています。具体的な例としては、1 月のデータのデータ・パーティションにデータを毎日ロードする、または、個々の行が継続して挿入されるような場合などです。

制約事項および使用ガイドライン

- データ・パーティションを非パーティション表に追加することはできません。既存の表のパーティション表へのマイグレーションの詳細については、203 ページの『既存の表およびビューをパーティション表にマイグレーションする』を参照してください。
- それぞれの新規データ・パーティションごとの値の範囲は、STARTING 節と ENDING 節によって決まります。
- STARTING 節および ENDING 節のいずれか、またはその両方を指定する必要があります。
- 新規の範囲は、既存のデータ・パーティションの範囲とオーバーラップしてはなりません。
- 最初の既存のデータ・パーティションの前に新規のデータ・パーティションを追加するときは、STARTING 節を指定する必要があります。この範囲を際限なしにするには MINVALUE を使用します。
- 同様に、最後の既存のデータ・パーティションの後に新規のデータ・パーティションを追加する場合は、ENDING 節を指定する必要があります。この範囲を際限なしにするには MAXVALUE を使用します。
- STARTING 節が省略される場合、データベースは、前のデータベース・パーティションの終了境界の直後に開始境界を作成します。同様に、ENDING 節が省略される場合、データベースは、次のデータ・パーティションの開始境界直前に終了境界を作成します。

- 開始節および終了節の構文は、CREATE TABLE ステートメントで指定するものと同じです。
- ADD PARTITION に IN または LONG IN 節が指定されない場合、データ・パーティションを配置する表スペースは、CREATE TABLE ステートメントによって使用されるのと同じ方式を使用して選択されます。
- パッケージは ALTER TABLE ...ADD PARTITION 操作中は無効にされます。
- 新しく追加されるデータ・パーティションは、ALTER TABLE ステートメントがコミットされた後、使用可能になります。

ADD または ATTACH 操作で STARTING または ENDING 境界を省略して範囲値のギャップを埋めることもできます。以下は、開始境界のみが指定されている ADD 操作を使用してギャップを埋める例です。

```
CREATE TABLE hole (c1 int) PARTITION BY RANGE (c1)
(STARTING FROM 1 ENDING AT 10, STARTING FROM 20 ENDING AT 30);
DB20000I The SQL command completed successfully.

ALTER TABLE hole ADD PARTITION STARTING 15;
DB20000I The SQL command completed successfully.

SELECT SUBSTR(tabname, 1,12) tabname,
SUBSTR(datapartitionname, 1, 12) datapartitionname,
seqno, SUBSTR(lowvalue, 1, 4) lowvalue, SUBSTR(highvalue, 1, 4) highvalue
FROM SYSCAT.DATAPARTITIONS WHERE TABNAME='HOLE' ORDER BY seqno;

TABNAME DATAPARTITIONNAME SEQNO LOWVALUE HIGHVALUE
------------ ----------------- ----------- -------- ---------
HOLE PART0 0 1 10
HOLE PART2 1 15 20
HOLE PART1 2 20 30

3 record(s) selected.
```

パーティション表を変更し、その表に新規のデータ・パーティションを追加するには、コマンド行から、ADD PARTITION 節を指定して ALTER TABLE ステートメントを発行します。

例 1: 901 から 1000 の範囲の値を持つ既存のパーティション表にデータ・パーティションを追加します。表 sales が値 900 までの 9 つの範囲 (0-100、101-200...) を持つとします。この例では、表の最後に追加範囲を加えています (STARTING 節がないことでそれが示されています)。

```
 ALTER TABLE sales ADD PARTITION dp10
(ENDING AT 1000 INCLUSIVE)
```

データ・パーティションのドロップ

DETACH PARTITION 節を指定して ALTER TABLE ステートメントを使用することによりデータ・パーティションをドロップすることができます。ただし、個別の表をドロップするには、DROP TABLE ステートメントを使用できます。

データ・パーティションをパーティション表からデタッチするには、ユーザーは以下の権限または特権を持っていなければなりません。

- DETACH を実行するユーザーは、ソース表に対して ALTER、SELECT、および DELETE 操作を行うために必要な権限を持っていなければなりません。
- ユーザーは、ターゲット表を作成するために必要な権限も持っていなければなりません。そのため、データ・パーティションをデタッチするために表を変更する

には、ステートメントの許可 ID が所有する特権に、少なくとも、ターゲット表に対する以下の特権の 1 つが含まれている必要があります。

- SYSADM または DBADM 権限
- データベースに対する CREATETAB 権限と表により使用される表スペースに対する USE 特権、および次のいずれか:
  - データベースに対する IMPLICIT_SCHEMA 権限 (表の暗黙または明示スキーマ名がない場合)
  - スキーマに対する CREATEIN 特権 (表のスキーマ名が既存のスキーマを指す場合)

表をドロップするには、ユーザーに次の権限または特権が必要です。

- SYSCAT.TABLES の DEFINER 列に記録されている定義者であるか、または次の特権の少なくとも 1 つを持っている必要があります。
  - SYSADM または DBADM 権限
  - 表のスキーマに対する DROPIN 特権
  - 表に対する CONTROL 特権

注: データ・パーティションのデタッチを行うと、ステートメントの許可 ID は事実上 CREATE TABLE ステートメントを発行しようとするため、その操作の実行に必要な特権が必要になります。表スペースは、デタッチされているデータ・パーティションがすでに存在する表スペースです。 ALTER TABLE ステートメントの許可 ID は、ユーザーが CREATE TABLE ステートメントを発行したかのように、CONTROL 権限を持つ新規表の定義者になります。変更されている表の権限はいずれも新規表には転送されません。 ALTER TABLE ... DETACH PARTITION 操作の直後にデータにアクセスできるのは、ALTER TABLE ステートメントの許可 ID、および DBADM または SYSADM のみです。

パーティション表のデータ・パーティションをデタッチするには、コマンド行から、DETACH PARTITION 節を指定して ALTER TABLE ステートメントを発行します。

表のドロップは、DB2 コマンド行プロセッサー (CLP) から行えます。

表をドロップするには、コマンド行から DROP TABLE ステートメントを発行します。次の例では、表 stock からデータ・パーティション dec01 がデタッチされ、表 junk に置かれます。その後、表 junk をドロップし、これによりそれに関連したデータ・パーティションもドロップされます。

```
ALTER TABLE stock DETACH PART dec01 INTO junk;
DROP TABLE junk;
```

注: DETACH 操作の実行速度をできるだけ上げるために、バックグラウンドの非同期索引クリーンアップ・プロセスを使用して、ソース表に対する索引クリーンアップが自動的に行われます。デタッチされた従属が存在する場合、デタッチされるデータ・パーティションは、デタッチ時にスタンドアロン表になりません。この場合、デタッチを完了して表にアクセスできるようにするには、SET INTEGRITY ステートメントを発行する必要があります。

シナリオ: パーティション表でのデータの入れ替え

DB2 データベースでデータを入れ替えると言う場合、これは廃止するデータを表から除去した後、新規データを追加することによってデータ・パーティションのスペースを再利用する方式のことを指します。表パーティション機能により、廃止するデータを持つデータ・パーティションをデタッチした後、最新データを持つ新規のデータ・パーティションをアタッチすることが可能になります。

データ・パーティションをパーティション表からデタッチするには、ユーザーは以下の権限または特権を持っていなければなりません。

- DETACH 操作を実行するユーザーには、ソース表に対する ALTER、SELECT、および DELETE に必要な権限が必要です。
- ユーザーは、ターゲット表を作成するために必要な権限も持っていなければなりません。したがって、データ・パーティションをデタッチする ALTER TABLE を行うには、ステートメントの許可 ID によって保持される特権に、ターゲット表に対する以下の権限または特権が少なくとも 1 つ含まれていなければなりません。
  - SYSADM または DBADM 権限
  - データベースに対する CREATETAB 権限と表により使用される表スペースに対する USE 特権、および次のいずれか:
    - データベースに対する IMPLICIT_SCHEMA 権限 (表の暗黙または明示スキーマ名がない場合)
    - スキーマに対する CREATEIN 特権 (表のスキーマ名が既存のスキーマを指す場合)

データ・パーティションをアタッチする ALTER TABLE を行うには、ユーザーに次の権限または特権が必要です。

- アタッチを実行するユーザーは、ターゲット表に対して ALTER および INSERT 操作を行うために必要な権限を持っていなければなりません。
- さらに、ユーザーはソース表に対して SELECT および DROP 操作を行うことができなければなりません。このため、データ・パーティションをアタッチするために表を変更するには、ステートメントの許可 ID が所有する特権に、少なくとも、ソース表に対する以下の特権の 1 つが含まれている必要があります。
  - ソース表に対する SELECT 特権およびソース表のスキーマに対する DROPIN 特権
  - ソース表に対する CONTROL 特権
  - SYSADM または DBADM 権限

パーティション表でデータを入れ替えるには、コマンド行から ALTER TABLE ステートメントを発行します。次の例は、2001 年 12 月のデータを除去し、それを 2003 年 12 月の最新データで置き換えることにより、在庫表を更新する方法を示しています。

1. 表 *stock* から古いデータを除去する。

   ```
   ALTER TABLE stock DETACH PARTITION dec01 INTO newtable;
   ```

2. 新規データをロードする。 REPLACE オプションを指定して LOAD を使用すると、既存のデータが上書きされます。

```
LOAD FROM data_file OF DEL REPLACE INTO newtable
```

注: デタッチされたパーティションに依存関係のあるオブジェクトがある場合、デタッチされた表をロードする前に、そのデタッチされた従属物に対して SET INTEGRITY ステートメントを実行する必要があります。

3. 必要に応じて、データ・クレンジングを実行する。データ・クレンジング・アクティビティーには次の操作が含まれます。
   - 欠落値の埋め込み
   - 矛盾データおよび不完全データの削除
   - 複数のソースに由来する冗長データの除去
   - データのトランスフォーム
     - 正規化 (同じ値を異なる方法で表現する別のソースからのデータは、データのウェアハウスへのローリング操作の一部として調整する必要があります。)
     - 集約 (詳しすぎてウェアハウスに保管できないロー・データは、ロールイン中に事前集約する必要があります。)
4. 新規データを新規範囲としてアタッチする。

   ```
   ALTER TABLE stock ATTACH PARTITION dec03
   STARTING '12/01/2003' ENDING '12/31/2003'
    FROM newtable;
   ```

   データ・パーティションをアタッチすると、照会が排出され、パッケージが無効になります。
5. SET INTEGRITY ステートメントを使用して、索引およびその他の従属オブジェクトを更新します。 SET INTEGRITY ステートメントの実行中には、読み取り/書き込みアクセスが許可されます。

   ```
   SET INTEGRITY FOR stock ALLOW WRITE ACCESS
   IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;
   ```

シナリオ: パーティション表データのロールインおよびロールアウト

ここでのシナリオは、新しいデータを各月の始めロールインし、古いデータを特定の日付に基づいて場合によってはロールアウトする、データウェアハウス内での共通の管理操作を扱っています。

シナリオ 1 は、不要になったデータを表から除去することによる DETACH 操作 (ロールアウト) を網羅しています。バリエーションとして、データの削除、および別の表へのデータの移動が含まれます。ここでのシナリオは、新規データを表にロードすることにより、ADD 操作と ATTACH 操作 (ロールイン) の両方を網羅しています。これには、次のバリエーションがあります。

1. データを変換し、それを非パーティション表にロードし、データ・パーティションをアタッチする (従来からの抽出、変換、ロード (ETL))。
2. データを非パーティション表にロードして、そのデータを変換する。
3. データ・パーティションをアタッチする。

シナリオ 1: パーティション表を使用すると、ロールアウト操作は単に適切なデータ・パーティションでの DETACH 操作となります。

```
ALTER TABLE stock DETACH PART dec01 INTO stock_drop;
COMMIT WORK
```

DETACH 操作の速度を上げるために、バックグラウンドの非同期索引クリーンアップ・プロセスを介して、ソース表に対する索引クリーンアップが自動的に行われます。従属オブジェクトがソース表に対して定義されていない場合、 DETACH 操作を完了するために SET INTEGRITY ステートメントを発行する必要はありません。

表をドロップする代わりに、表を別の表にアタッチしたり、それを切り捨てて新規データのロード先の表として使用してから再アタッチしたりすることができます。これらの操作は、デタッチされた従属オブジェクトがパーティションを構成する表 (例では STOCK 表) にある場合を除いて、たとえ非同期索引クリーンアップが完了する前であっても、即時に実行できます。

デタッチされた表がまだアクセス可能になっていないことを検出するには、SYSCAT.TABDETACHEDDEP カタログ・ビューを照会します。アクセス不能なデタッチされた表が検出される場合、すべてのデタッチされた従属に対して SET INTEGRITY ステートメントを IMMEDIATE CHECKED オプションを指定して実行し、デタッチされた表を通常のアクセス可能な表に遷移します。すべてのデタッチされた従属を保守する前に、デタッチされた表にアクセスすると、エラー・コード SQL20285N が戻されます。

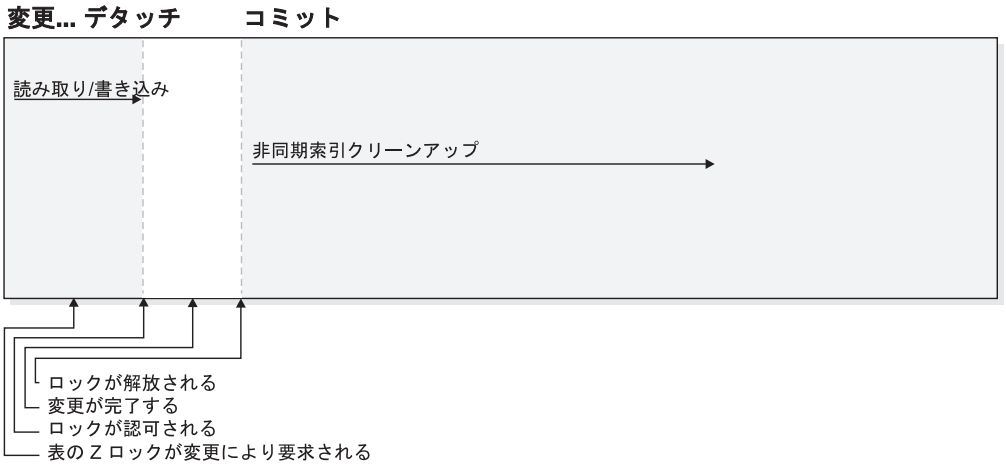


図 37. この図は、DETACH 操作中のデータ可用性のステージを示しています。デタッチされた従属がない場合、非同期索引クリーンアップは、DETACH 操作がコミットされた直後に開始されます。デタッチされた従属がある場合、非同期索引クリーンアップは、デタッチされた従属の保守がコミットされた後に開始されます。

シナリオ 2: 空の範囲の新規作成

次のシナリオは、データを非パーティション表にロードしてから、そのデータ・パーティションを表の他の部分に追加する手順を示しています。

```
ALTER TABLE stock ADD PARTITION dec03
STARTING FROM '12/01/2003' ENDING AT '12/31/2003';
```

この ALTER TABLE ... ADD 操作は、STOCK 表に対して実行されている照会をドレーンして、パッケージを無効にします。つまり、ADD 操作が続行される前に、既存の照会は正常に完了します。 ADD 操作が発行されると、STOCK 表にアクセスするすべての新規照会はロックによりブロックされます。

次のように、データを表にロードします。

```
LOAD FROM data_file OF DEL INSERT
INTO stock ALLOW READ ACCESS;
```

次のように SET INTEGRITY ステートメントを使用して、制約を妥当性検査してから、従属のマテリアライズ照会表 (MQT) をリフレッシュします。

```
SET INTEGRITY FOR stock ALLOW READ
ACCESS IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;

COMMIT WORK;
```

ヒント:LOAD 操作の後に ALTER TABLE ... ATTACH を使用する方法と比較して、ALTER TABLE ... ADD PARTITION の後に LOAD 操作を使用する方法の利点の 1 つは、表に制約または MQT が定義されていない場合に、新規のデータを使用可能にするために SET INTEGRITY ステートメントが必要なくなるということです。新規のデータ・パーティションを追加してデータを表に直接ロードする方法には、いくつかの欠点があります。 ALTER TABLE ... ADD PARTITION ステートメントを使用する方法の主な欠点は、ロード操作の期間中と、その後の SET INTEGRITY ステートメントの期間中の両方で、表の更新が妨げられることです。ALTER TABLE ... ADD PARTITION ステートメントと ALTER TABLE ... ATTACH PARTITION ステートメントはどちらもパッケージを無効にしますが、LOAD コマンドの後に ALTER ... ATTACH PARTITION 操作を使用するとデータの可用性は高まります。ただし、ALTER TABLE ... ADD PARTITION ステートメントの後に IMPORT コマンドを使用する方法、または通常の INSERT ステートメントを使用する方法は、データが大きなブロックでロールインされるのではなく少しずつ取り込まれる状況のときには効果があります。ロールインされるデータがデータ・パーティションの境界と適合しない場合にも、データ・パーティションを追加することは効果があります。

シナリオ 3: パーティション表を使用すると、ロールイン操作は単に新規にロードされたデータ・パーティションの ATTACH 操作となります。

このシナリオでは、ATTACH を使用して新規のデータ範囲を既存のパーティション表にロードすることを容易にします。通常、データは新規の空の表にロードされて、ターゲット表に影響を与えることなく、必要なデータのクリーニングおよび検査が行われます。データが準備された後、新規にロードされたデータ・パーティションがアタッチされます。

```
CREATE TABLE dec03(.....);
LOAD FROM data_file OF DEL REPLACE INTO dec03;
```

表データをロールインする前に、そのデータをアタッチする前にデータ・クレンジングが必要となることがあります。データ・クレンジング・アクティビティーには次の操作が含まれます。

- 欠落値の埋め込み
- 矛盾データおよび不完全データの削除

- 複数のソースに由来する冗長データの除去
- データのトランスフォーム
  - 正規化 (同じ値を異なる方法で表現する別のソースからのデータは、データのウェアハウスへのローリング操作の一部として調整する必要があります。)
  - 集約 (詳しすぎてウェアハウスに保管できないロー・データは、ロールイン中に事前集約する必要があります。)

次に、データをロールインします。

```
ALTER TABLE stock ATTACH PARTITION dec03
STARTING FROM '12/01/2003' ENDING AT '12/31/2003'
FROM dec03;
```

アタッチ処理においては、STARTING 節、ENDING 節のどちらかまたは両方を指定する必要があります。また、STARTING 節に定義する下限の値 (STARTING) は、ENDING 節に定義する上限値 (ENDING) より小さいか同じである必要があります。既存の表に新たにデータ・パーティションをアタッチする場合は、ターゲット表の既存のデータ・パーティションの範囲に重なり合ってはなりません。一番上位のレンジが MAXVALUE で定義されている場合は、新しく上位のレンジを定義しようとしても既存のレンジと重なり合ってしまうため定義できません。この制約は最下限のレンジが MINVALUE で定義されている場合にも当てはまります。範囲内の既存のギャップに収まらない限り、新しいデータ・パーティションを中間に追加またはアタッチすることはできません。ユーザーによって指定されない境界は、表の作成時に決定されます。

ALTER TABLE ... ATTACH 操作は、すべての照会をドレーンして、STOCK 表にある従属のパッケージを無効にします。つまり、ATTACH 操作が続行される前に、既存の照会は正常に完了します。 ATTACH 操作が発行されると、STOCK 表にアクセスするすべての新規照会はロックによりブロックされます。この遷移の間、STOCK 表は z ロックされます (完全にアクセス不能)。アタッチされたデータ・パーティション内のデータは、まだ SET INTEGRITY ステートメントによって妥当性検査されていないので、表示することができません。**ヒント:** ATTACH 操作の直後に COMMIT WORK ステートメントを発行して、その表を使用可能にしてください。

```
COMMIT WORK;
```

SET INTEGRITY ステートメントは、新規にアタッチされたデータが範囲内にあることを検証するために必要です。このステートメントはさらに、索引および MQT などの他の従属オブジェクトの必要な保守を行います。 SET INTEGRITY ステートメントがコミットされるまで、新しいデータを表示することはできません。オンライン SET INTEGRITY を使用した場合、STOCK 表内の既存のデータは、読み取りと書き込みの両方に関して完全にアクセス可能となります。 SET INTEGRITY 実行中のデフォルトは、ALLOW NO ACCESS モードです。

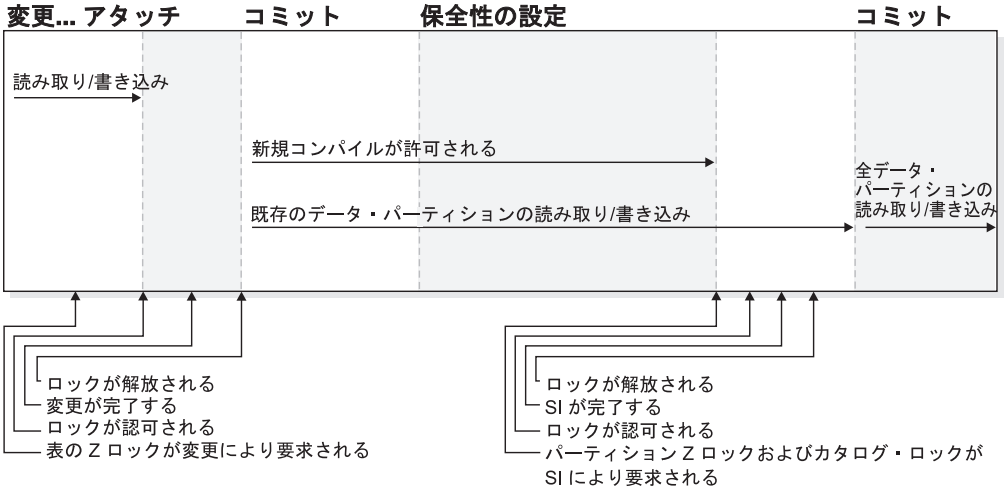


図 38. この図は、ATTACH 操作中のデータ可用性のステージを示しています。

注: SET INTEGRITY の実行中に、DDL またはユーティリティー・タイプの操作を表に対して実行することはできません。それらの操作には、
LOAD、REORG、REDISTRIBUTE、ALTER TABLE (例えば、列の追加、 『NOT LOGGED INITIALLY』に対して ALTER を使用する
ADD、ATTACH、DETACH、TRUNCATE)、および INDEX CREATE があり、その他にも該当する操作がある可能性があります。

```
SET INTEGRITY FOR stock ALLOW WRITE ACCESS
IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;
```

SET INTEGRITY は、新規にアタッチされたデータ・パーティション内のデータを妥当性検査します。

次に、トランザクションをコミットして表を使用可能にします。

```
COMMIT WORK;
```

範囲外であるか、または他の制約に違反する行は、例外表 stock_ex に移動されます。 stock_ex を照会して、違反している行を検査することができ、場合によっては、それらをクリーンアップして表に再挿入することもできます。

第 14 章 ロード

並列処理とロード

ロード・ユーティリティーは、複数のプロセッサーや複数の記憶装置が使用されているハードウェア構成 (対称マルチプロセッサー (SMP) 環境など) を利用します。

ロード・ユーティリティーを使って大容量データの並列処理を実行する方法はいくつかあります。その 1 つは複数の記憶装置を使用する方法であり、ロード操作中に入出力の並列処理が可能になります (図 39 を参照)。別の方法は SMP 環境における複数のプロセッサーの使用が関係しており、パーティション内の並列処理が可能になります (図 40 を参照)。これらの両方の方法を併用すれば、データのロード時間をさらに短くすることができます。

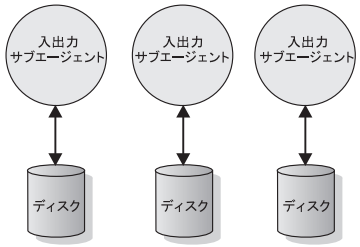


図 39. データ・ロード時に入出力の並列処理を利用する

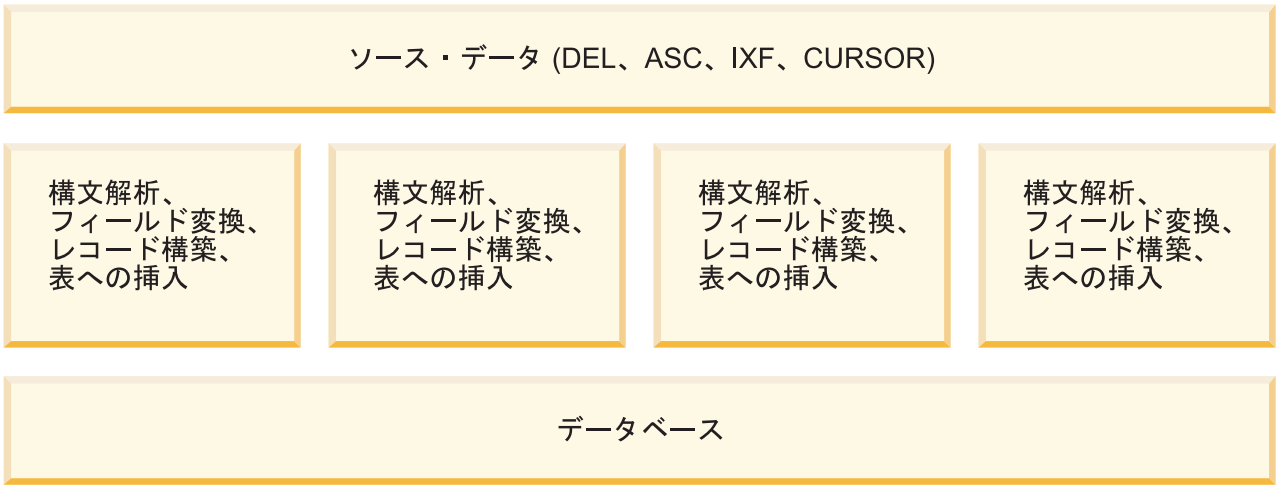


図 40. データ・ロード時のパーティション内の並列処理の利用

のマルチディメンション・クラスタリングの考慮事項

以下の制約事項はマルチディメンション・クラスタリング (MDC) 表へのデータのロード時に適用されます。

- LOAD コマンドの SAVECOUNT オプションはサポートされていません。
- これらの表は独自のフリー・スペースを管理するため、 `totalfreespace` ファイル・タイプ修飾子はサポートされていません。
- MDC 表には `anyorder` ファイル・タイプ修飾子が必要です。 `anyorder` 修飾子を使わないで MDC 表へのロードを実行すると、 `anyorder` 修飾子はユーティリティーによって暗黙的に使用可能になります。

MDC 表に LOAD コマンドを使用する場合には、ユニーク制約の違反は以下のように処理されます。

- ロードするデータと同じユニーク・キーを持つレコードが (ロード操作の開始前に既に) 表に存在する場合、元のレコードは残り、新規レコードが削除フェーズで削除されます。
- ロードするデータと同じユニーク・キーを持つレコードが (ロード操作の開始前には) 表に存在しない場合、ユニーク・キーとそれと重複する (同じユニーク・キーを持つ) レコードの両方が表にロードされる場合には、レコードのうち 1 つだけがロードされ、その他のレコードは削除フェーズで除去されます。

  注: どのレコードがロードされて、どのレコードが削除されるかを判別するための明示的な技法はありません。

パフォーマンスの考慮

MDC 表のロード時のロード・ユーティリティーのパフォーマンスを改善するには、*util_heap_sz* データベース構成パラメーター値を大きくしなければなりません。ユーティリティーで使用できるメモリーを増やすと、 mdc-load アルゴリズムのパフォーマンスが大きく向上します。こうすると、ロード・フェーズで実行されるデータのクラスタリング時に、ディスクの入出力を減らすことができます。 LOAD コマンドの DATA BUFFER オプションが指定される際には、この値も大きくしなければなりません。 LOAD コマンドを使用して複数の MDC 表を同時にロードする場合には、それに応じて、*util_heap_sz* の値を大きくしなければなりません。

すべての MDC 表にはブロック索引があるため、MDC ロード操作には常に構築フェーズがあります。

ロード・フェーズでは、ブロック・マップの保守のために余分のロギングが実行されます。割り振られるエクステントごとに、おおよそ 2 つの余分のログ・レコードがあります。パフォーマンスを良くするためには、このことを考慮に入れた値に *logbufsz* データベース構成パラメーターを設定する必要があります。

MDC 表にデータをロードするために、索引付きのシステム一時表が使用されます。表のサイズはロードされる個々のセルの数に比例します。表にあるそれぞれの行のサイズは MDC 次元キーのサイズに比例します。ロード操作時にこの表の操作によるディスク入出力を最小限に抑えるには、 TEMPORARY 表スペースのバッファー・プールの大きさが十分であることを確認してください。

パーティション表におけるロードの考慮事項

以下の一般制約事項を除き、既存のすべてのロード・フィーチャーはターゲット表がパーティション化されている場合にサポートされます。

- パーティション・エージェントが複数存在する場合、整合点はサポートされません。
- データ・パーティションのサブセットにデータをロードしている間、その他のデータ・パーティションを完全にオンラインのままにしておく機能はサポートされません。
- ロード操作で使用される例外表は、パーティション化できません。
- ロード・ユーティリティーが挿入モードまたは再始動モードで実行されていて、デタッチされた従属データがロード・ターゲット表にある場合には、ユニーク索引を再作成することはできません。
- MDC 表のロードと同様、入力データ・レコードの厳密な順序は、パーティション表をロードする際には保持されません。順序はセルまたはデータ・パーティションの中のみで維持されます。
- 各データベース・パーティションで複数のフォーマッターを使用するロード操作では、入力レコードの大まかな順序のみを保持します。各データベース・パーティション上で単一のフォーマッターを実行すると、入力レコードがセルまたは表パーティション・キーごとにグループ化されます。各データベース・パーティション上で単一のフォーマッターを実行するには、明示的に CPU_PARALLELISM に 1 を要求してください。

一般的なロードの動作

ロード・ユーティリティーは、データ・レコードを適切なデータ・パーティションに挿入します。ロードの前に入力データをパーティション化するための外部ユーティリティー (スプリッターなど) を使用する上での要件はありません。

ロード・ユーティリティーは、アタッチまたはデタッチされたデータ・パーティションにアクセスしません。データは可視のデータ・パーティションのみに挿入されます。可視のデータ・パーティションは、アタッチされたりデタッチされたりしません。また、ロード置換操作では、アタッチまたはデタッチされたデータ・パーティションを切り捨てることはありません。ロード・ユーティリティーではカタログ・システム表上のロックを獲得するため、ロード・ユーティリティーはコミットされていない ALTER TABLE トランザクションがあれば待機します。そのようなトランザクションは、カタログ表内の関連する行の排他ロックを獲得します。排他ロックを終了しなければロード操作は進行できません。これは、ロード操作の実行中は、コミットされていない ALTER TABLE ...ATTACH、DETACH、または ADD PARTITION トランザクションはありえないということを意味します。アタッチまたはデタッチされたデータ・パーティションに宛てられたすべての入力ソース・レコードはリジェクトされ、例外表が指定されている場合にはそこから取得できます。ターゲット表データ・パーティションの一部がアタッチまたはデタッチされた状態であったことを示すため、通知メッセージがメッセージ・ファイルに書き込まれます。ターゲット表に対応するカタログ表の関連する行のロックは、ロード・ユーティリティーの実行中に ALTER TABLE ...ATTACH、DETACH、または ADD PARTITION 操作を実行することによりユーザーがターゲット表のパーティションを変更することを防ぎます。

無効な行の処理
ロード・ユーティリティーで可視のデータ・パーティションのいずれにも属さないレコードが検出されると、そのレコードはリジェクトされ、ロード・ユーティリティーは処理を継続します。範囲制約違反のためにリジェクトされたレコードの数は明示的には表示されませんが、リジェクトされたレコードの全体数には含められます。範囲違反のためにレコードをリジェクトしても行の警告数は増加しません。範囲違反が検出されたものの、レコードごとのメッセージはログに記録されないということを示す単一のメッセージ (SQL0327N) がロード・ユーティリティーのメッセージ・ファイルに書き込まれます。例外表には、ターゲット表のすべての列に加えて、特定の行で発生した違反のタイプを記述する列が含まれます。無効データを含む行 (パーティション化できないデータを含む) は、ダンプ・ファイルに書き込まれます。

例外表への挿入は非効率であるため、どの制約違反を例外表に挿入するかを制御できます。例えば、ロード・ユーティリティーのデフォルトの動作は、範囲制約違反またはユニーク制約違反のためにリジェクトされた (その違反がなければ有効だった) 行を例外表に挿入することです。この動作は、FOR EXCEPTION 節を使用し、NORANGEEXC (範囲制約違反の場合) または NOUNIQUEEXC (ユニーク制約違反の場合) を指定することによってオフにすることができます。それらの制約違反を例外表に挿入しないことを指定する場合、または例外表を指定しない場合、範囲制約またはユニーク制約に違反する行に関する情報は失われます。

履歴ファイル
ターゲット表がパーティション化されている場合、対応する履歴ファイルの項目は、ターゲット表により範囲を設定された表スペースのリストを含みません。操作対象のオブジェクト ID (「T」ではなく「R」) は、ロード操作がパーティション表に対して実行されたことを示します。

ロード操作の終了
ロード置換を終了すると、すべてのデータ・パーティションが完全に切り捨てられ、ロード挿入を終了すると、すべてのデータ・パーティションがロード前の長さに切り捨てられます。ロード・コピー・フェーズで失敗した ALLOW READ ACCESS 操作の終了中に索引は無効になります。索引にタッチした ALLOW NO ACCESS ロード操作を終了する時にも索引は無効になります (それが無効になるのは、索引付けモードが再作成されているか増分保守の間にキーが挿入されたために索引が不整合状態になっているためです)。データを複数のターゲットにロードしても、ロード・フェーズ中に取られた整合点からロード操作を再始動できない点を除き、ロード・リカバリー操作に何の影響もありません。この場合、ターゲット表がパーティション化されている場合には、SAVECOUNT ロード・オプションは無視されます。この動作は、MDC ターゲット表へのデータのロードと一貫しています。

生成列
生成される列がパーティション・キー、ディメンション・キー、または分散キーのいずれかにある場合、 `generatedoverride` ファイル・タイプ修飾子は無視され、ロード・ユーティリティーは `generatedignore` ファイル・タイプ修飾子が指定された場合のように値を生成します。ここで不正な生成列値をロードすると、レコードが不適切な物理ロケーション (例えば不適切なデータ・パーティション、MDC ブロック、またはデータベース・パーティション) に配置されてしまう可能性があります。例えば、あるレコードがいったん間違ったデータ・パーティションに置かれる

と、整合性の設定ではそのレコードを別の物理ロケーションに移動しなければなりませんが、それはオンラインでの整合性の設定操作中には行えません。

データの可用性
現行の ALLOW READ ACCESS ロード・アルゴリズムは、パーティション表に拡張されています。ALLOW READ ACCESS ロード操作では、複数のリーダーが同時に表全体 (ロードするデータ・パーティションとロードしないものの両方を含む) にアクセスすることができます。

データ・パーティションの状態
ロードに成功した後、特定の条件下では、可視のデータ・パーティションの表の状態が「SET INTEGRITY ペンディング」または「読み取りアクセスのみ」のいずれかまたは両方に変更される場合があります。ロード操作で保守できない制約が表にある場合に、データ・パーティションはこれらの状態になる可能性があります。そのような制約には、チェック制約とデタッチされたマテリアライズ照会表が含まれる場合があります。ロード操作に失敗すると、すべての可視のデータ・パーティションの表の状態が「ロード・ペンディング」になります。

エラー分離
データ・パーティション・レベルでのエラー分離はサポートされていません。エラーを分離するとは、エラーにならなかったデータ・パーティションでロードを継続し、エラーになったデータ・パーティションで停止するということを意味します。エラーは異なるデータベース・パーティションの間で分離できますが、ロード・ユーティリティーは可視のデータ・パーティションのサブセット上でトランザクションをコミットしたり、残りの可視のデータ・パーティションをロールバックしたりすることはできません。

その他の考慮事項
- いずれかの索引が無効とマークされている場合には増分索引付けはサポートされません。索引の再作成が必要な場合、またはデタッチされた従属物が SET INTEGRITY ステートメントでの妥当性検査を必要としている場合には、索引は無効であると見なされます。
- 範囲別パーティション化、ハッシュによる分散、またはディメンションによる編成のいずれかのアルゴリズムの組み合わせを使用してパーティション化された表へのロードもサポートされています。
- ロードの影響を受けるオブジェクトと表スペース ID のリストが含まれるログ・レコードの場合、これらのログ・レコード (LOAD START および COMMIT (PENDING LIST)) のサイズは非常に大きくなる場合があり、そうなると、他のアプリケーションで使用できるアクティブ・ログ・スペースの量が減少してしまいます。
- 表がパーティション化されていて、かつ分散されている場合、パーティション・データベースのロードがすべてのデータベース・パーティションには影響を与えない場合があります。出力データベース・パーティション上のオブジェクトのみが変更されます。
- ロード操作中、パーティション表のメモリー使用量は表の数とともに増加します。増加の合計は直線的にならない点に注意してください。データ・パーティションの数に比例するのはメモリー所要量全体のうちのほんの僅かだからです。

第 15 章 パーティション・データベース環境でのデータのロード

ロードの概要 - パーティション・データベース環境

複数パーティション・データベースでは、大量のデータが多数のデータベース・パーティションに散在しています。データの各部分がどのデータベース・パーティションに入るかは、分散キーによって決定されます。また、適切なデータベース・パーティションにデータをロードする前に、データを分散 しておく必要があります。

複数パーティション・データベースに表をロードする際に、ロード・ユーティリティーは以下を実行できます。

- 入力データを並列で分散します。
- データをそれぞれ対応するデータベース・パーティションに同時にロードします。
- あるシステムから別のシステムにデータを転送します。

複数パーティション・データベースへのデータのロードは、セットアップとロードの 2 つのフェーズで実行されます。セットアップ・フェーズ では表ロックなどのデータベース・パーティション・リソースが獲得され、ロード・フェーズ ではデータがデータベース・パーティションにロードされます。 LOAD コマンドの ISOLATE_PART_ERRS オプションを使用すると、これらのフェーズのいずれかで発生するエラーを処理する方法、および 1 つまたは複数のデータベース・パーティションでのエラーが、エラーのないデータベース・パーティションでのロード操作にどのように影響を与えるかを選択できます。

複数パーティション・データベースにデータをロードする際には、以下のいずれかのモードを使用できます。

PARTITION_AND_LOAD
データは (多くの場合は並列で) 分散され、それぞれ対応するデータベース・パーティションに同時にロードされます。

PARTITION_ONLY
データは (多くの場合は並列で) 分散され、それぞれのロード・データベース・パーティションの指定したファイルに出力が書き込まれます。それぞれのファイルにはパーティション・ヘッダーがあり、データがいくつかのデータベース・パーティションに分散された方法、および LOAD_ONLY モードを使用してデータベースにファイルをロードできることを示します。

LOAD_ONLY
データはすでにいくつかのデータベース・パーティションに分散されているものとします。この場合は分散プロセスが省略され、データはそれぞれ対応するデータベース・パーティションに同時にロードされます。

LOAD_ONLY_VERIFY_PART
データはすでにいくつかのデータベース・パーティションに分散されているものとしますが、データ・ファイルにはパーティション・ヘッダーがありません。分散プロセスは省略され、データはそれぞれ対応するデータベース・

パーティションに同時にロードされます。ロード操作時には、それぞれの行が正しいデータベース・パーティションにあることがチェックされます。dumpfile ファイル・タイプ修飾子が指定されている場合には、データベース・パーティション違反のある行がダンプ・ファイルに入れられます。そうでなければ、行は廃棄されます。ロードしている特定のデータベース・パーティションにデータベース・パーティション違反がある場合、そのデータベース・パーティションのロード・メッセージ・ファイルに 1 つの警告が書き込まれます。

ANALYZE
すべてのデータベース・パーティションに均一に分散する最適な分散マップが生成されます。

概念および用語

複数データベース・パーティションを持つパーティション・データベース環境でのロード・ユーティリティーの動作および操作について説明する際には、以下の用語が使用されます。

- *コーディネーター・パーティション* は、ロード操作を実行するためにユーザーが接続するデータベース・パーティションです。PARTITION_AND_LOAD、PARTITION_ONLY、および ANALYZE モードでは、LOAD コマンドの CLIENT オプションが指定されていない限り、データ・ファイルはこのデータベース・パーティションに存在するものとされます。 CLIENT を指定すると、ロードされるデータが、リモートで接続されるクライアントに存在することを示します。
- PARTITION_AND_LOAD、PARTITION_ONLY、および ANALYZE モードでは、*事前パーティション化エージェント* がユーザー・データを読み取り、データを分散するパーティション化エージェント にラウンドロビン方式でこれを分散します。この処理は、コーディネーター・パーティションで常に実行されます。どのロード操作の場合でも、 1 つのデータベース・パーティションにつき最大 1 つのパーティション化エージェントが許可されます。
- PARTITION_AND_LOAD、LOAD_ONLY、および LOAD_ONLY_VERIFY_PART モードでは、それぞれの出力データベース・パーティションで*ロード・エージェント* が実行し、そのデータベース・パーティションへのデータのロードを調整します。
- *ファイル・エージェントへのロード* は、 PARTITION_ONLY ロード操作時にそれぞれの出力データベース・パーティションで実行します。これらはパーティション化エージェントからデータを受信し、これをデータベース・パーティションにあるファイルに書き込みます。
- SOURCEUSEREXIT オプションを使用すると、カスタマイズしたスクリプトまたは実行ファイル (ここでは*ユーザー出口*と呼びます) をロード・ユーティリティーが実行するための機構が提供されます。

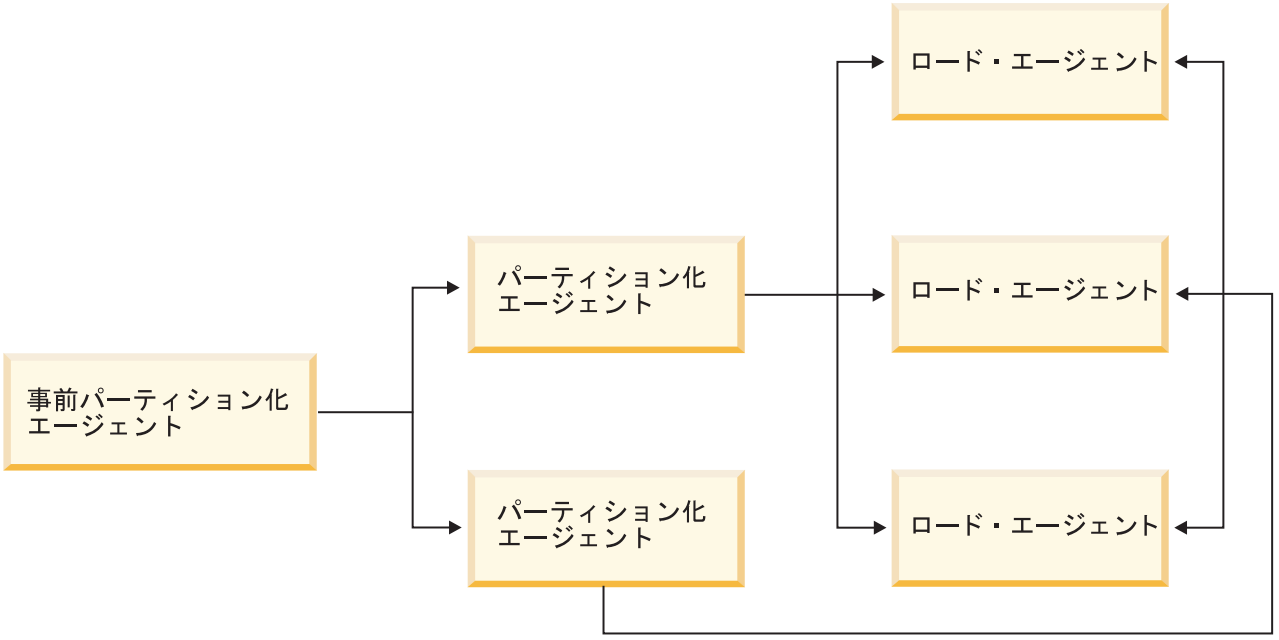


図 41. パーティション・データベース・ロードの概説： 事前パーティション化エージェントによりソース・データを読み、2 つのパーティション化エージェントそれぞれに約半分のデータが送られます。パーティション化エージェントはデータを分散し、各パーティションを 3 つのデータベース・パーティションのいずれかに送ります。各データベース・パーティションのロード・エージェントがデータをロードします。

パーティション・データベース環境でのデータのロード - ヒント

複数パーティション・データベースに表をロードする前に、以下のことを考慮してください。

- ロード構成オプションに慣れるために、まず小さなデータを使ってこのユーティリティーを操作してください。
- 入力データがあらかじめソートされている場合、または特定の順序になっている場合に、ロード・プロセス中にその順序を維持するとき、分散に使用するデータベース・パーティションは 1 つだけにしてください。並列分散では、必ずしもデータを受け取ったのと同じ順序でロードするとは限りません。 LOAD コマンドで `anyorder` 修飾子を指定しないと、ロード・ユーティリティーはデフォルトで単一パーティション化エージェントを選択します。
- 複数に分割されたファイルからラージ・オブジェクト (LOB) をロードする場合 (つまりロード・ユーティリティーの `lobsinfile` 修飾子を使っている場合)、LOB ファイルの入っているすべてのディレクトリーは、ロード先のすべてのデータベース・パーティションから読み取り可能でなければなりません。 LOB を処理する場合、ロードのパラメーター *lob-path* は完全修飾パスでなければなりません。
- ISOLATE_PART_ERRS オプションを SETUP_ERRS_ONLY または SETUP_AND_LOAD_ERRS に設定することにより、(起動時に) ロード操作でロード・データベース・パーティションや関連する表スペースまたは表がオフラインになっていることを検出した場合でも、複数パーティション・データベースで実行されているジョブを続行させることができます。

- STATUS_INTERVAL ロード構成オプションを使用して、複数パーティション・データベースで実行されているジョブの進行をモニターします。ロード操作は、指定された時間間隔でメッセージを生成し、事前パーティション化エージェントによって読み取られたデータの量をメガバイト単位で表示します。これらのメッセージは、事前パーティション化エージェント・メッセージ・ファイルにダンプされます。ロード操作中にこのファイルの内容を表示するには、コーディネーター・パーティションに接続してから、ターゲット表に対して LOAD QUERY コマンドを発行します。
- (PARTITIONING_DBPARTNUMS オプションで定義される) 分散プロセスに関係しているデータベース・パーティションが、 (OUTPUT_DBPARTNUMS オプションで定義される) ロード・データベース・パーティションと異なっている場合、CPU サイクルに対する競合が少なくなるため、パフォーマンスの改善が望めます。複数パーティション・データベースにデータをロードする際には、ロード・ユーティリティーを分散操作にもロード操作にも関係しないデータベース・パーティションで起動してください。
- LOAD コマンドに MESSAGES パラメーターを指定すると、事前パーティション化、パーティション化、およびロード・エージェントからのメッセージ・ファイルを保管して、ロード操作の終了時に参照できるようにします。ロード操作中にこれらのファイルの内容を表示するには、希望するデータベース・パーティションに接続してから、ターゲット表に対して LOAD QUERY コマンドを発行します。
- ロード・ユーティリティーは、統計情報を収集する出力データベース・パーティションを 1 つだけ選択します。そのデータベース・パーティションを指定するには、 RUN_STAT_DBPARTNUM データベース構成オプションを使用できます。
- 複数パーティション・データベースでデータをロードする場合、事前に設計アドバイザーを実行して、各表ごとに最適なパーティションを判別します。詳しくは、データベース・パフォーマンスのチューニングの『設計アドバイザー』を参照してください。

トラブルシューティング

ロード・ユーティリティーが停止した場合には、以下を実行できます。

- STATUS_INTERVAL パラメーターを使用して、複数パーティション・データベース・ロード操作の進行をモニターすることができます。状況インターバル情報は、コーディネーター・パーティションの事前パーティション化エージェント・メッセージ・ファイルにダンプされます。
- パーティション化エージェント・メッセージ・ファイルをチェックして、それぞれのデータベース・パーティションでのパーティション化エージェント・プロセスの状況を調べます。エラーなしでロードが進行しており、TRACE オプションが設定された場合には、これらのメッセージ・ファイル内に多くのレコードに関するトレース・メッセージがあるはずです。
- ロード・メッセージ・ファイルをチェックして、ロード・エラー・メッセージがあるかどうかを調べます。

  注: これらのファイルを出力するためには、LOAD コマンドの MESSAGES オプションを指定しなければなりません。

- ロード・プロセスのいずれかに問題が発生したことを暗示するエラーを発見した場合、現在のロード操作を中断します。

パーティション・データベース環境でのデータのロード

ロード・ユーティリティーを使用して、パーティション・データベース環境にデータをロードします。

複数パーティションを持つデータベースに表をロードする前に確認する事項：

1. データベース・マネージャー構成パラメーター *svcename*、およびプロファイル・レジストリー変数 **DB2COMM** が正しく設定されていることを確認してください。ロード・ユーティリティーは TCP/IP を使用して事前パーティション化エージェントからパーティション化エージェントにデータを転送し、さらにパーティション化エージェントからロード・データベース・パーティションにデータを転送するため、このことは重要です。
2. ロード・ユーティリティーを起動するには、その前にデータのロード先となるデータベースに接続されているか、または暗黙接続が可能な状態になっていなければなりません。ロード・ユーティリティーは COMMIT ステートメントを発行するため、ロード・ユーティリティーの呼び出し前に COMMIT または ROLLBACK ステートメントを発行することにより、すべてのトランザクションを完了し、すべてのロックを解除しておかなければなりません。
   PARTITION_AND_LOAD、 PARTITION_ONLY、または ANALYZE モードが使用されている場合には、ロードされるデータ・ファイルは、以下の状態になっていない限り、このデータベース・パーティションになければなりません。
   a. CLIENT オプションが指定されている場合。この場合には、データがクライアント・マシンになければなりません。
   b. 入力ソース・データが CURSOR である場合。この場合には、入力ファイルはありません。
3. 設計アドバイザーを実行して、各表ごとに最適なデータベース・パーティションを決定します。詳しくは、データベース・パフォーマンスのチューニングの『設計アドバイザー』を参照してください。

ロード・ユーティリティーを使用して複数パーティション・データベースにデータをロードする際には、以下の制約が適用されます。

- ロード操作の入力ファイルのロケーションとして磁気テープ装置を指定することはできません。
- ANALYZE モードが使用されていない限り、ROWCOUNT オプションはサポートされません。
- ターゲット表に分散に必要な ID 列があり、`identityoverride` ファイル・タイプ修飾子が指定されていない場合、または複数のデータベース・パーティションを使ってデータを分散してからロードする場合、LOAD コマンドにおいて 0 より大きい SAVECOUNT 値の使用はサポートされていません。
- ID 列が分散キーの一部を構成している場合は、PARTITION_AND_LOAD モードだけがサポートされます。
- LOAD コマンドの CLIENT オプションを指定する際には、 LOAD_ONLY およびLOAD_ONLY_VERIFY_PART モードは使用できません。

- CURSOR 入力ソース・タイプを指定する際には、LOAD_ONLY_VERIFY_PART モードは使用できません。
- LOAD コマンドの ALLOW READ ACCESS および COPY YES オプションを指定する際には、分散エラー分離モード LOAD_ERRS_ONLY および SETUP_AND_LOAD_ERRS は使用できません。
- OUTPUT_DBPARTNUMS および PARTITIONING_DBPARTNUMS オプションによって指定されるデータベース・パーティションがオーバーラップしない場合には、複数のロード操作が同じ表に同時にデータをロードすることができます。例えば、表がデータベース・パーティション 0 から 3 に定義されている場合、 1 つのロード操作がデータベース・パーティション 0 および 1 にデータをロードする一方で、 2 番目のロード操作でデータベース・パーティション 2 および 3 にデータをロードすることができます。
- 区切りなし ASCII (ASC) および区切り付き ASCII (DEL) ファイルのみ、複数のデータベース・パーティションにまたがる複数の表に分散できます。 PC/IXF ファイルは分散できませんが、LOAD_ONLY_VERIFY_PART モードでロード操作を行って、複数のデータベース・パーティションに分散されている表にロードすることはできます。

以下の例では、LOAD コマンドを使用して各種のロード操作を開始する方法を説明します。以下の例で使用されるデータベースには、5 つのデータベース・パーティション、0、1、2、3、および 4 があります。データベース・パーティションにはそれぞれローカル・ディレクトリー /db2/data/ があります。データベース・パーティション 0、1、3、および 4 では、2 つの表、TABLE1 および TABLE2 が定義されます。クライアントからロードする際には、ユーザーにはデータベース・パーティションのいずれかではないリモート・クライアントへのアクセスがあります。

サーバー・パーティションからのロード

分散およびロードの例

このシナリオでは、 TABLE1 が定義されているか、または定義されていないデータベース・パーティションに接続しているものとします。データ・ファイル load.del は、このデータベース・パーティションの現行作業ディレクトリーにあります。load.del から、TABLE1 が定義されているすべてのデータベース・パーティションにデータをロードするには、次のコマンドを発行します。

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
```

注: この例では、すべてのパーティション・データベース環境の構成パラメーターでデフォルト値を使用します。MODE パラメーターのデフォルトは PARTITION_AND_LOAD で、 OUTPUT_DBPARTNUMS オプションのデフォルトは、TABLE1 が定義されているすべてのデータベース・パーティションです。また、PARTITIONING_DBPARTNUMS のデフォルトは、LOAD コマンド規則に従って選択したデータベース・パーティションのセットです。この規則は、データベース・パーティションが指定されていない場合にそれを選択するためのものです。

データがデータベース・パーティション 3 および 4 に分散されるようにロード操作を実行するには、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
                    PARTITIONED DB CONFIG PARTITIONING_DBPARTNUMS (3,4)
```

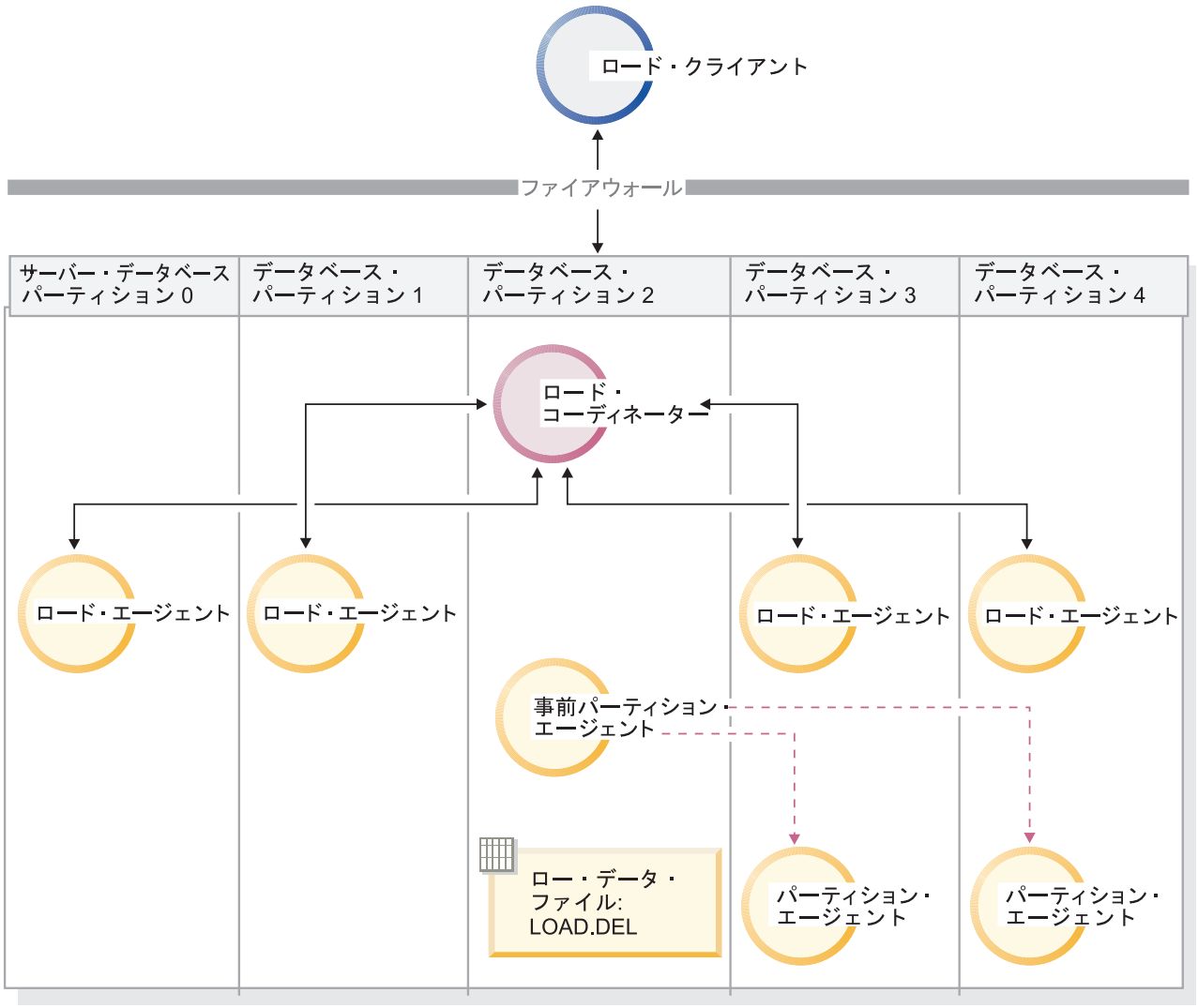


図 42. この図では、前のコマンドが発行された結果の動作を説明します。データは、データベース・パーティション 3 および 4 にロードされます。

分散のみの例

このシナリオでは、 TABLE1 が定義されているか、または定義されていないデータベース・パーティションに接続しているものとします。データ・ファイル load.del は、このデータベース・パーティションの現行作業ディレクトリーにあります。TABLE1 が定義されているすべてのデータベース・パーティションに load.del を分散する (ロードはしない) には、データベース・パーティション 3 および 4 を使用し、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
                      PARTITIONED DB CONFIG MODE PARTITION_ONLY
                                             PART_FILE_LOCATION /db2/data
                                             PARTITIONING_DBPARTNUMS (3,4)
```

これにより、ファイル load.del.xxx は、それぞれのデータベース・パーティションにある /db2/data ディレクトリーに保管されます。ここで、xxx は、3 桁表記のデータベース・パーティション番号です。

データベース・パーティション 0 (PARTITIONING_DBPARTNUMS のデフォルト) で実行しているパーティション化エージェントを 1 つだけ使用して、 load.del ファイルをデータベース・パーティション 1 および 3 に分散するには、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE PARTITION_ONLY
                                              PART_FILE_LOCATION /db2/data
                                              OUTPUT_DBPARTNUMS (1,3)
```

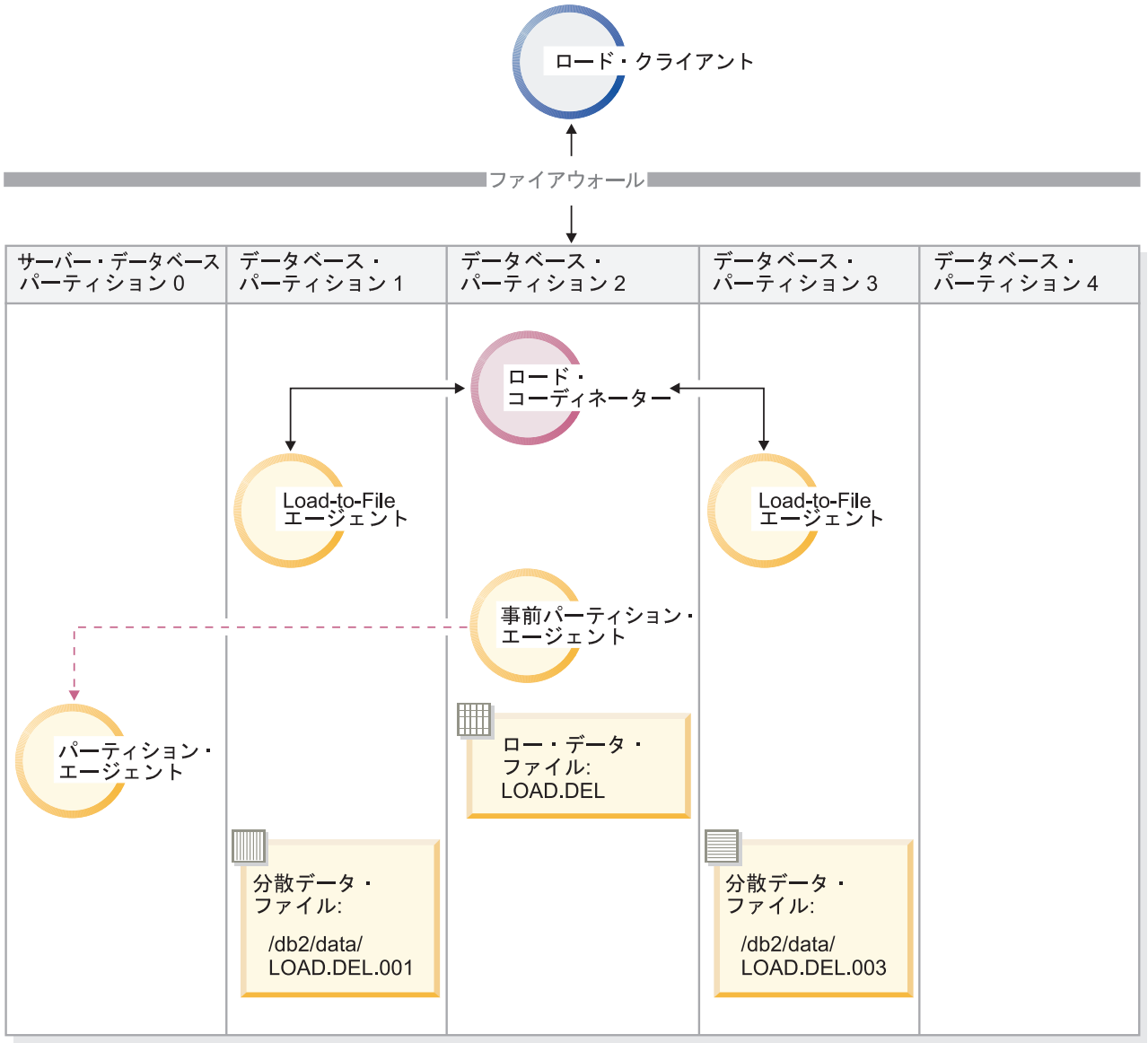


図 43. この図では、前のコマンドが発行された結果の動作を説明します。データは、データベース・パーティション 0 で実行する 1 パーティション化エージェントを使用して、データベース・パーティション 1 および 3 にロードされます。

ロードのみの例

すでに PARTITION_ONLY モードでロード操作を実行しており、 TABLE1 が定義されているすべてのデータベース・パーティションにそれぞれのロード・データベ

ース・パーティションの /db2/data ディレクトリーにあるパーティション・ファイルをロードする場合には、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE LOAD_ONLY
                                               PART_FILE_LOCATION /db2/data
```

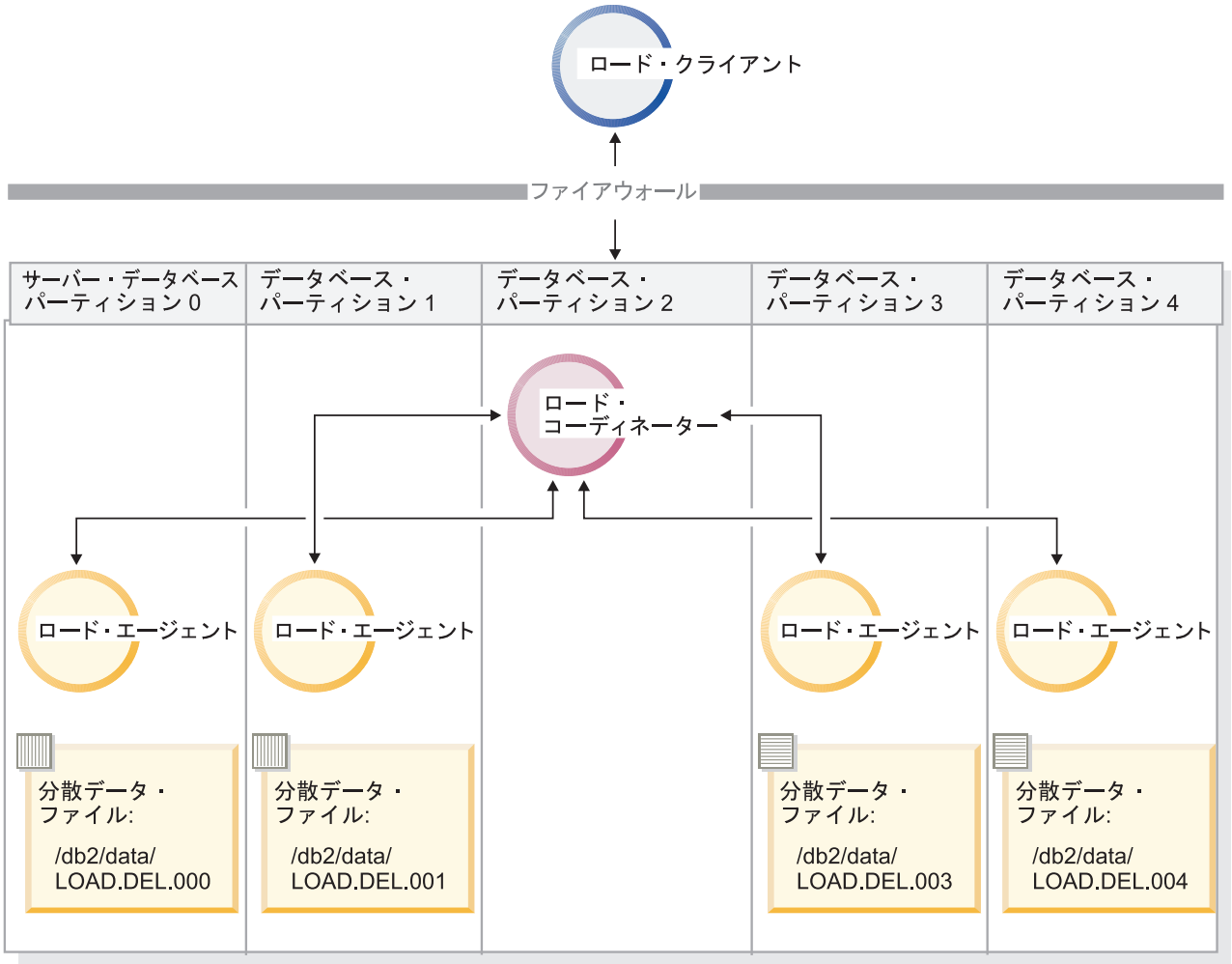


図 44. この図では、前のコマンドが発行された結果の動作を説明します。分散データは、TABLE1 が定義されているすべてのデータベース・パーティションにロードされます。

データベース・パーティション 4 にだけロードするには、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
                        PARTITIONED DB CONFIG MODE LOAD_ONLY
                                               PART_FILE_LOCATION /db2/data
                                               OUTPUT_DBPARTNUMS (4)
```

分散マップ・ヘッダーのない事前分散ファイルのロード

LOAD コマンドを使用して、分散ヘッダーのないデータ・ファイルを、いくつかのデータベース・パーティションに直接ロードすることができます。 TABLE1 が定義されているそれぞれのデータベース・パーティションの /db2/data ディレクトリー

にデータ・ファイルが存在しており、この名前が load.del.xxx である場合 (xxx はデータベース・パーティション番号)、以下のコマンドを発行することによってファイルをロードできます。

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
                       REPLACE INTO TABLE1
                       PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
                                             PART_FILE_LOCATION /db2/data
```

データベース・パーティション 1 にだけデータをロードするには、以下のコマンドを発行します。

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
                       REPLACE INTO TABLE1
                       PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
                                             PART_FILE_LOCATION /db2/data
                                             OUTPUT_DBPARTNUMS (1)
```

注: ロード元のデータベース・パーティションにない行が指定された場合には、これはリジェクトされ、ダンプ・ファイルに入れられます。

リモート・クライアントから複数パーティション・データベースへのロード

リモート・クライアントにあるファイルから複数パーティション・データベースにデータをロードするには、 LOAD コマンドの CLIENT オプションを指定して、サーバー・パーティションにデータ・ファイルが存在しないことを示す必要があります。例:

```
LOAD CLIENT FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

注: LOAD_ONLY または LOAD_ONLY_VERIFY_PART モードを CLIENT オプションと共に使用することはできません。

カーソルからのロード

単一パーティション・データベースの場合と同様に、カーソルから複数パーティション・データベースにロードすることができます。この例では、PARTITION_ONLY および LOAD_ONLY モードの場合、 PART_FILE_LOCATION オプションは完全修飾ファイル名を指定しなければなりません。この名前は、それぞれの出力データベース・パーティションで作成またはロードされる分散ファイルの完全修飾基本ファイル名になります。ターゲット表に LOB 列がある場合には、指定されたベース名で複数のファイルを作成できます。

将来 TABLE2 にロードする目的で、 /db2/data/select.out.xxx (ここで、xxx はデータベース・パーティション番号) という名前のそれぞれのデータベース・パーティションにあるファイルに、ステートメント SELECT * FROM TABLE1 の応答セット内のすべての行を分散するには、以下のコマンドを発行します。

```
DECLARE C1 CURSOR FOR SELECT * FROM TABLE1

LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
                       PARTITIONED DB CONFIG MODE PARTITION_ONLY
                       PART_FILE_LOCATION /db2/data/select.out
```

上記の操作によって生成されるデータ・ファイルは、以下の LOAD コマンドを発行することによってロードできます。

```
LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
                        PARTITIONED CB CONFIG MODE LOAD_ONLY
                        PART_FILE_LOCATION /db2/data/select.out
```

LOAD QUERY コマンドを使用したパーティション・データベース環境でのロード操作のモニター

パーティション・データベース環境でロード操作を行う際、いくつかのロード・プロセスによって、それらが実行されるデータベース・パーティションにメッセージ・ファイルが作成されます。

これらのメッセージ・ファイルには、すべての情報、ロード操作の実行時に生成される警告およびエラー・メッセージが保管されます。ユーザーが表示できるメッセージ・ファイルを生成するロード・プロセスは、ロード・エージェント、事前パーティション化エージェント、およびパーティション化エージェントです。メッセージ・ファイルの内容は、ロード操作が終了してからでなければ使用できません。

ロード操作時に個々のデータベース・パーティションに接続し、ターゲット表に対して LOAD QUERY コマンドを発行できます。このコマンドが CLP から発行されると、LOAD QUERY コマンドで指定された表の、現在このデータベース・パーティションにあるすべてのメッセージ・ファイルの内容が表示されます。

例えば、データベース・パーティション 0 から 3 で構成されるデータベース WSDB の表、TABLE1 が定義されているとします。データベース・パーティション 0 に接続され、以下の LOAD コマンドを発行します。

```
load from load.del of del replace into table1 partitioned db config
    partitioning_dbpartnums (1)
```

このコマンドは、データベース・パーティション 0、1、2、および 3 で実行するロード・エージェント、データベース・パーティション 1 で実行するパーティション化エージェント、データベース・パーティション 0 で実行する事前パーティション化エージェントを組み込むロード操作を開始します。

データベース・パーティション 0 には、事前パーティション化エージェントのメッセージ・ファイルが 1 つ、およびそのデータベース・パーティションでのロード・エージェントのファイルが 1 つ配備されます。これらのファイルの内容を同時に表示するには、新しいセッションを開始し、以下のコマンドを CLP から発行します。

```
set client connect_node 0
connect to wsdb
load query table table1
```

データベース・パーティション 1 には、ロード・エージェントのファイルが 1 つ、およびパーティション化エージェントのファイルが 1 つ配備されます。これらのファイルの内容を表示するには、新しいセッションを開始し、以下のコマンドを CLP から発行します。

```
set client connect_node 1
connect to wsdb
load query table table1
```

注: STATUS_INTERVAL ロード構成オプションによって生成されるメッセージは、事前パーティション化エージェント・メッセージ・ファイルに表示されます。ロード操作時にこれらのメッセージを表示するには、コーディネーター・パーティションに接続してから、LOAD QUERY コマンドを発行しなければなりません。

メッセージ・ファイルの内容の保管

db2Load API を介してロード操作が開始される場合には、メッセージ・オプション (piLocalMsgFileName) を指定しなければならず、メッセージ・ファイルはサーバーからクライアントに移動して、表示できるように保管されます。

CLP から開始される複数パーティション・データベースのロード操作では、メッセージ・ファイルがコンソールに表示されたり、保持されたりすることはありません。複数パーティション・データベースのロードの完了後にこれらのファイルの内容を保管または表示するには、LOAD コマンドの MESSAGES オプションを指定しなければなりません。このオプションが使用される場合、ロード操作が完了すると、それぞれのデータベース・パーティションのメッセージ・ファイルはクライアント・マシンに転送され、MESSAGES オプションによって示されるベース名を使用してファイルに保管されます。複数パーティション・データベースのロード操作の場合の、生成されたロード・プロセスに対応するファイルの名前を以下にリストします。

プロセス・タイプ	ファイル名
ロード・エージェント	<message-file-name>.LOAD.<dbpartition-number>
パーティション化エージェント	<message-file-name>.PART.<dbpartition-number>
事前パーティション化エージェント	<message-file-name>.PREP.<dbpartition-number>

例えば、MESSAGES オプションが `/wsdb/messages/load` を指定すると、データベース・パーティション 2 のロード・エージェント・メッセージ・ファイルは `/wsdb/messages/load.LOAD.002` です。

注: CLP から開始した複数パーティション・データベースのロード操作には、MESSAGES オプションを使用することを強くお勧めします。

パーティション・データベース環境でのロード操作の再開、再始動、または終了

パーティション・データベース環境でロード操作が失敗した場合、その後に実行すべきステップは、失敗がいつ発生したかによって異なります。

複数パーティション・データベースでのロード・プロセスは 2 つのステージで構成されます。

1. 1 つはセットアップ・ステージ です。このステージ中に、出力データベース・パーティションに対する表ロックなどのデータベース・パーティション・レベルのリソースを取得します。

通常、セットアップ・ステージで障害が発生した場合には、操作の再始動および終了は必要ではありません。行うべき作業は、失敗したロード操作で指定されたエラー分離モードによって異なります。

セットアップ・ステージ・エラーを分離しないことをロード操作で指定した場合、ロード操作全体がキャンセルされ、それぞれのデータベース・パーティションの表の状態は、ロード操作以前の状態にロールバックされます。

セットアップ・ステージ・エラーを分離することをロード操作で指定した場合、ロード操作は、セットアップ・ステージが正常に実行されたデータベース・パーティションで継続しますが、失敗したそれぞれのデータベース・パーティションにある表は、ロード操作以前の状態にロールバックされます。これは、セットアップ・ステージ中に失敗するパーティションとロード・ステージ中に失敗するパーティションがある場合、単一のロード操作が複数のステージで失敗する可能性があることを意味します。

2. もう 1 つは*ロード・ステージ* です。このステージ中にデータがフォーマットされ、データベース・パーティション上の表にロードされます。

   複数パーティション・データベースのロード操作のロード・ステージで少なくとも 1 つのデータベース・パーティションにおいてロード操作が失敗した場合には、LOAD RESTART または LOAD TERMINATE コマンドを発行しなければなりません。これが必要になるのは、複数パーティション・データベースでのデータのロードが単一のトランザクションで実行されるためです。

   ロード失敗の原因となった問題を修正できる場合は LOAD RESTART を選択してください。ロードの再始動操作が開始されると、ロードが中断した時点から、すべてのデータベース・パーティションでロード操作が継続されるので、これは時間の節約になります。

   表を初期ロード操作前の状態に戻す場合は LOAD TERMINATE を選択してください。

手順

ロードがいつ失敗したかの判別

パーティション環境でロード操作が失敗したらまず、どのパーティションでロード操作が失敗したか、またそれぞれどのステージで失敗したかを判別する必要があります。これは、パーティション・サマリーを調べることによって行えます。ロード・コマンドが CLP から発行された場合、パーティション・サマリーはロードの最後に表示されます (以下の例を参照)。ロード・コマンドが db2Load API から発行された場合、パーティション・サマリーは db2PartLoadOut 構造の poAgentInfoList フィールドで確認できます。

ある特定のパーティションの ″Agent Type″ の項目が ″LOAD″ となっている場合、そのパーティションがロード・ステージに達していることを示しています。それ以外の場合、失敗はセットアップ・ステージ中に発生したことを示します。負の SQL コードは失敗を示します。以下は、ロード・ステージ中にパーティション 1 で失敗したロードの例です。

```
Agent Type     Node     SQL Code     Result
_____________________________________________________________
LOAD           000      +00000000    Success.
_____________________________________________________________
LOAD           001      -00000289    Error. May require RESTART.
_____________________________________________________________
LOAD           002      +00000000    Success.
_____________________________________________________________
LOAD           003      +00000000    Success.
.
.
.
```

失敗したロードの再開、再始動、または終了

セットアップ・ステージ中に失敗するのは、SETUP_ERRS_ONLY または SETUP_AND_LOAD_ERRS を指定した ISOLATE_PART_ERRS オプションを使用するロードだけです。このステージ中に少なくとも 1 つの出力データベース・パーティションで失敗したロードについては、 LOAD REPLACE または LOAD INSERT コマンドを発行することができます。 OUTPUT_DBPARTNUMS オプションを使用して、失敗が発生したデータベース・パーティションのみを指定してください。

ロード・ステージ中に少なくとも 1 つの出力データベース・パーティションで失敗するロードについては、LOAD RESTART または LOAD TERMINATE コマンドを発行してください。

セットアップ・ステージ中に少なくとも 1 つの出力データベース・パーティションで、またロード・ステージ中に少なくとも 1 つの出力データベース・パーティションで失敗するロードについては、前述したとおり、失敗したロードを再開するために 2 つのロード操作 (1 つはセットアップ・ステージでの失敗に対するもので、もう 1 つはロード・ステージでの失敗に対するもの) を実行する必要があります。このタイプの失敗ロード操作を効率的に取り消すには、LOAD TERMINATE コマンドを発行します。ただし、セットアップ・ステージ中に失敗したパーティション上の表に対しては変更が行われず、ロード・ステージ中に失敗したパーティションに関してはすべての変更が取り消されたため、このコマンドを発行した後はそれに伴う対応をすべてのパーティションに対して行う必要があります。

例えば、データベース・パーティション 0 から 3 で構成されるデータベース WSDB で TABLE1 が定義されているとします。以下のコマンドが発行されます。

```
load from load.del of del insert into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

セットアップ・ステージ中に出力データベース・パーティション 1 で失敗が発生します。セットアップ・ステージ・エラーは分離されるため、ロード操作は継続します。しかし、ロード・ステージ中にパーティション 3 で失敗が発生します。ロードを操作を再開するには、以下のコマンドを発行します。

```
load from load.del of del replace into table1 partitioned db config
output_dbpartnums (1)

load from load.del of del restart into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

注: ロード再始動操作では、LOAD RESTART コマンドで指定されるオプションが使用されるため、元の LOAD コマンドで指定されるものと同一であることが重要です。

パーティション・データベース環境でのロード構成オプション

MODE X

複数パーティション・データベースをロードする際に実行するロード操作のモードを指定します。 PARTITION_AND_LOAD がデフォルトです。有効な値は以下のとおりです。

- PARTITION_AND_LOAD. データは (多くの場合は並列で) 分散され、それぞれ対応するデータベース・パーティションに同時にロードされます。
- PARTITION_ONLY. データは (多くの場合は並列で) 分散され、それぞれのロード・データベース・パーティションの指定したファイルに出力が書き込まれます。 CURSOR 以外のファイル・タイプの場合、各データベース・パーティションの出力ファイル名のフォーマットは `filename.xxx` です。ここで `filename` は、LOAD コマンドで指定された入力ファイル名で、 `xxx` は 3 桁のデータベース・パーティション番号です。 CURSOR ファイル・タイプの場合、各データベース・パーティションの出力ファイルの名前は PART_FILE_LOCATION オプションによって決められます。各データベース・パーティションの分散ファイルの位置を指定する方法の詳細については、 PART_FILE_LOCATION オプションを参照してください。

  注:

  1. このモードは CLI ロード操作には使用できません。
  2. 分散に必要な ID 列が表に収められている場合は、 `identityoverride` ファイル・タイプ修飾子が指定されない限り、このモードはサポートされません。
  3. ファイル・タイプ CURSOR 用に生成される分散ファイルは、 DB2 の異なるリリース間で互換性がありません。これは、前のリリースで生成されたファイル・タイプ CURSOR の分散ファイルは、LOAD_ONLY モードを使用してロードできないということを意味します。同様に、現行リリースで生成されたファイル・タイプ CURSOR の分散ファイルは、将来のリリースでは LOAD_ONLY モードを使用してロードできません。
- LOAD_ONLY。 データはすでに分散されているものとします。この場合は分散プロセスが省略され、データはそれぞれ対応するデータベース・パーティションに同時にロードされます。 CURSOR 以外のファイル・タイプの場合、各データベース・パーティションの入力ファイル名のフォーマットは `filename.xxx` となります。ここで `filename` は、LOAD コマンドで指定されたファイルの名前で、 `xxx` は 3 桁のデータベース・パーティション番号です。 CURSOR ファイル・タイプの場合、各データベース・パーティションの入力ファイルの名前は PART_FILE_LOCATION オプションによって決められます。各データベース・パーティションの分散ファイルの位置を指定する方法の詳細については、PART_FILE_LOCATION オプションを参照してください。

  注:

  1. このモードは CLI ロード操作には使用できず、 LOAD コマンドの CLIENT オプションが指定されている場合にも使用できません。

2. 分散に必要な ID 列が表に収められている場合は、 identityoverride ファイル・タイプ修飾子が指定されない限り、このモードはサポートされません。

- LOAD_ONLY_VERIFY_PART. データはすでに分散されているものとしますが、データ・ファイルにはパーティション・ヘッダーがありません。分散プロセスは省略され、データはそれぞれ対応するデータベース・パーティションに同時にロードされます。ロード操作時には、それぞれの行が正しいデータベース・パーティションにあることがチェックされます。 dumpfile ファイル・タイプ修飾子が指定されている場合には、データベース・パーティション違反のある行がダンプ・ファイルに入れられます。そうでなければ、行は廃棄されます。ロードしている特定のデータベース・パーティションにデータベース・パーティション違反がある場合、そのデータベース・パーティションのロード・メッセージ・ファイルに 1 つの警告が書き込まれます。各データベース・パーティションの入力ファイル名のフォーマットは filename.xxx となり、ここで filename は LOAD コマンドで指定されたファイルの名前で、 xxx は 3 桁のデータベース・パーティション番号です。各データベース・パーティションの分散ファイルの位置を指定する方法の詳細については、 PART_FILE_LOCATION オプションを参照してください。

  注:

  1. このモードは CLI ロード操作には使用できず、 LOAD コマンドの CLIENT オプションが指定されている場合にも使用できません。
  2. 分散に必要な ID 列が表に収められている場合は、 identityoverride ファイル・タイプ修飾子が指定されない限り、このモードはサポートされません。

- ANALYZE。 すべてのデータベース・パーティションに均一に分散する最適な分散マップが生成されます。

PART_FILE_LOCATION X

PARTITION_ONLY、LOAD_ONLY、および LOAD_ONLY_VERIFY_PART モードでは、このパラメーターは、分散ファイルのロケーションを指定するために使用できます。このロケーションは、OUTPUT_DBPARTNUMS オプションによって指定される各データベース・パーティションに存在しなければなりません。指定されたロケーションが相対パス名の場合には、そのパスが現行ディレクトリーに追加されて、分散ファイルのロケーションが作成されます。

CURSOR ファイル・タイプの場合、このオプションを指定しなければならず、ロケーションは完全修飾されたファイル名を参照していなければなりません。この名前は、PARTITION_ONLY モードの場合は、各出力データベース・パーティションで作成された分散ファイルの完全修飾された基本ファイル名、または LOAD_ONLY モードの場合は、各データベース・パーティションから読み取ることのできるファイルのロケーションです。 PARTITION_ONLY モードの使用時に、ターゲット表中に LOB 列があると、指定した基本名のファイルが複数作成されることがあります。

CURSOR 以外のファイル・タイプの場合、このオプションが指定されないと、現行ディレクトリーが分散ファイルに使用されます。

OUTPUT_DBPARTNUMS X

X は、データベース・パーティション番号のリストを示します。データベース・パーティション番号は、ロード操作が実行されるデータベース・パーティションを示します。データベース・パーティション番号は、表が定義されているデータベース・パーティションのサブセットでなければなりません。デフォルトでは、すべてのデータベース・パーティションが選択されます。リストは括弧で囲まなければならず、リスト内のアイテムはコンマで区切らなければなりません。範囲を指定できます (例えば、(0, 2 to 10, 15))。

PARTITIONING_DBPARTNUMS X

X は、分散プロセスで使用されるデータベース・パーティション番号のリストを示します。リストは括弧で囲まなければならず、リスト内のアイテムはコンマで区切らなければなりません。範囲を指定できます (例えば、(0, 2 to 10, 15))。分散プロセスで指定されるデータベース・パーティションは、ロードされるデータベース・パーティションと異なっていてもかまいません。PARTITIONING_DBPARTNUMS が指定されない場合には、最適なパフォーマンスを実現するために、ロード・ユーティリティーにより必要なデータベース・パーティションの数と使用するデータベース・パーティションが判別されます。

LOAD コマンドで anyorder ファイル・タイプ修飾子が指定されていない場合、ロード・セッションではパーティション化エージェントが 1 つだけ使用されます。さらに、OUTPUT_DBPARTNUMS オプションにデータベース・パーティションが 1 つだけ指定されている場合、またはロード操作のコーディネーター・パーティションが OUTPUT_DBPARTNUMS のエレメントではない場合、分散プロセスでロード操作のコーディネーター・パーティションが使用されます。その他の場合には、OUTPUT_DBPARTNUMS で最初のデータベース・パーティション (コーディネーター・パーティションではない) が分散プロセスで使用されます。

anyorder ファイル・タイプ修飾子が指定されている場合には、分散プロセスで使用されるデータベース・パーティションの数は、(OUTPUT_DBPARTNUMS のパーティションの数)/4 + 1 で決定されます。

MAX_NUM_PART_AGENTS X

ロード・セッションで使用されるパーティション化エージェントの最大数を指定します。デフォルトは 25 です。

ISOLATE_PART_ERRS X

個々のデータベース・パーティションで発生するエラーにロード操作がどのように対応するかを指示します。 LOAD コマンドで ALLOW READ ACCESS および COPY YES オプションの両方が指定される場合 (この場合のデフォルトは NO_ISOLATION) を除き、デフォルトは LOAD_ERRS_ONLY です。有効な値は以下のとおりです。

- SETUP_ERRS_ONLY。 セットアップ時にデータベース・パーティションにエラーが発生すると (データベース・パーティションへのアクセス時の障害、またはデータベース・パーティション上の表スペースまたは表へのアクセス時の障害など)、ロード操作は障害のあるデータベース・パーティションでは停止しますが、残りのデータベース・パーティションでは実行を継続します。データのロード中にデータベース・パーティションでエラーが発生すると、全体の操作が失敗します。

- LOAD_ERRS_ONLY。 セットアップ時にデータベース・パーティションにエラーが発生すると、ロード操作全体が失敗します。データのロード中にエラーが発生する場合、ロード操作はエラーが生じたデータベース・パーティションで停止します。残りのデータベース・パーティションでは、障害が発生するか、すべてのデータがロードされるまでロードが継続されます。新しくロードされたデータは、ロード再始動操作が実行されて正常に完了するまで表示されません。

  注: LOAD コマンドで ALLOW READ ACCESS および COPY YES オプションの両方が指定される場合には、このモードは使用できません。
- SETUP_AND_LOAD_ERRS。 このモードでは、セットアップまたはデータのロード時に生じるデータベース・パーティション・レベルのエラーによって、影響を受けたデータベース・パーティション上でのみ、処理が停止します。 LOAD_ERRS_ONLY モードの場合と同様、データのロード中にパーティション・エラーが発生した場合、新しくロードされたデータはロード再始動操作が実行されて正常に完了するまで表示されません。

  注: LOAD コマンドで ALLOW READ ACCESS および COPY YES オプションの両方が指定される場合には、このモードは使用できません。
- NO_ISOLATION。 ロード操作時にエラーがあれば、ロード操作は失敗します。

STATUS_INTERVAL X
X は、読み取られたデータのボリュームを通知する頻度を示します。メジャー単位はメガバイト (MB) です。デフォルトは 100 MB です。有効な値は 1 から 4000 の整数です。

PORT_RANGE X
X は、内部通信のためのソケットの作成に使う TCP ポートの範囲を表します。デフォルトの範囲は 6000 から 6063 です。 **DB2ATLD_PORTS** レジストリー変数の値が起動時に定義される場合には、その値は PORT_RANGE ロード構成オプションの値で置き換えられます。 **DB2ATLD_PORTS** レジストリー変数の場合、範囲は以下のフォーマットで提供されます。

```
<lower-port-number:higher-port-number>
```

CLP からの場合は、以下のフォーマットです。

```
( lower-port-number, higher-port-number )
```

CHECK_TRUNCATION
プログラムが入出力時にデータ・レコードの切り捨てをチェックするように指定します。デフォルトの動作では、入出力時にはデータの切り捨てをチェックしません。

MAP_FILE_INPUT X
X は、分散マップの入力ファイル名を指定します。このパラメーターはカスタマイズされた分散マップの入ったファイルを示すため、分散マップがカスタマイズされている場合にはこのパラメーターを指定しなければなりません。カスタマイズされた分散マップを作成するには、 db2gpmap プログラムを使用してデータベース・システム・カタログ表からマップを抽出するか、または、LOAD コマンドの ANALYZE モードを使用して最適なマップを生成します。ロード操作を継続するには、その前に ANALYZE モードを

使用して生成されるマップをデータベース内のそれぞれのデータベース・パーティションに移動する必要があります。

MAP_FILE_OUTPUT X

X は、分散マップの出力ファイル名を示します。出力ファイルが作成されるのは、LOAD コマンドが発行されたデータベース・パーティションです。ただし、これは、そのデータベース・パーティションが、パーティション化の実行されるデータベース・パーティション・グループに関係している場合です。パーティション化に関係していないデータベース・パーティション (PARTITIONING_DBPARTNUMS によって定義される) 上で LOAD コマンドが呼び出された場合、出力ファイルは、PARTITIONING_DBPARTNUMS パラメーターを使って最初に定義されたデータベース・パーティションに作成されます。次のパーティション・データベース環境のセットアップを考慮してください。

```
1 serv1 0
2 serv1 1
3 serv2 0
4 serv2 1
5 serv3 0
```

serv3 で次の LOAD コマンドを実行すると、serv1 上に分散マップが作成されます。

```
LOAD FROM file OF ASC METHOD L ( ...) INSERT INTO table CONFIG
MODE ANALYZE PARTITIONING_DBPARTNUMS(1,2,3,4)
MAP_FILE_OUTPUT '/home/db2user/distribution.map'
```

このパラメーターは、ANALYZE モードが指定される際に使用しなければなりません。すべてのデータベース・パーティションに均一に分散する最適な分散マップが生成されます。このパラメーターが指定されておらず ANALYZE モードが指定されている場合には、プログラムは終了してエラーを戻します。

TRACE X

データ変換プロセスとハッシュ値の出力のダンプを調べることが必要になった場合にトレースするレコードの数を指定します。デフォルトは 0 です。

NEWLINE

入力データ・ファイルが、それぞれのレコードが改行文字によって区切られた ASC ファイルであり、 LOAD コマンドで reclen ファイル・タイプ修飾子が指定されている場合に使用されます。このオプションが指定されると、それぞれのレコードの改行文字がチェックされます。また、reclen ファイル・タイプ修飾子で指定されたレコード長もチェックされます。

DISTFILE X

このオプションを指定すると、ロード・ユーティリティーは、指定された名前のデータベース・パーティション分散ファイルを生成します。データベース・パーティション分散ファイルには 4096 個の整数が入っており、それぞれはターゲット表の分散マップの各項目に対応しています。ファイル内の各整数は、ロードされる入力ファイルの中で、対応する分散マップ項目にハッシュされる行数を表します。この情報はデータの中のスキューを識別するのに役立ちます。さらに、ユーティリティーの ANALYZE モードを使って表

の新しい分散マップを生成すべきかどうか判断するのに役立ちます。このオプションを指定しない場合、ロード・ユーティリティーのデフォルト動作として、配布ファイルを生成しません。

注: このオプションを指定すると、最大で 1 つのパーティション化エージェントがロード操作のために使用されます。複数のパーティション化エージェントを明示的に要求した場合でも、1 つのみが使用されます。

OMIT_HEADER
分散ファイルに分散マップ・ヘッダーを組み込まないように指定します。指定されていなければ、ヘッダーが生成されます。

RUN_STAT_DBPARTNUM X
LOAD コマンドに STATISTICS YES パラメーターが指定された場合には、 1 つのデータベース・パーティションでのみ統計が収集されます。このパラメーターは、統計を収集するデータベース・パーティションを指定します。値が -1 か、またはまったく指定されない場合には、出力データベース・パーティション・リストの最初のデータベース・パーティションで統計が収集されます。

パーティション・データベース環境でのロード・セッション - CLP の例

以下の例は、複数パーティション・データベースでのデータのロードを示しています。

データベースに 0 から 3 の番号の付いた 4 つのデータベース・パーティションがあります。データベース WSDB がすべてのデータベース・パーティションで定義されており、表 TABLE1 が、すべてのデータベース・パーティションでも定義されているデフォルト・データベース・パーティション・グループにあります。

例 1

データベース・パーティション 0 にあるユーザー・データ・ファイル load.del から TABLE1 にデータをロードするには、データベース・パーティション 0 に接続してから、以下のコマンドを発行します。

```
load from load.del of del replace into table1
```

ロード操作が正常に実行された場合、出力は以下のようになります。

```
Agent Type     Node     SQL Code     Result
_____________________________________________________
LOAD           000      +00000000    Success.
_____________________________________________________
LOAD           001      +00000000    Success.
_____________________________________________________
LOAD           002      +00000000    Success.
_____________________________________________________
LOAD           003      +00000000    Success.
_____________________________________________________
PARTITION      001      +00000000    Success.
_____________________________________________________
PRE_PARTITION  000      +00000000    Success.
_____________________________________________________
RESULTS:       4 of 4 LOADs completed successfully.
_____________________________________________________
```

```
Summary of Partitioning Agents:
Rows Read                    = 100000
Rows Rejected                = 0
Rows Partitioned             = 100000

Summary of LOAD Agents:
Number of rows read          = 100000
Number of rows skipped       = 0
Number of rows loaded        = 100000
Number of rows rejected      = 0
Number of rows deleted       = 0
Number of rows committed     = 100000
```

出力では、それぞれのデータベース・パーティションごとに 1 つのロード・エージェントがあり、それぞれが正常に実行されたことを示しています。また、コーディネーター・パーティションで実行する事前パーティション化エージェントが 1 つ、およびデータベース・パーティション 1 で実行するパーティション化エージェントが 1 つあったことも示しています。これらのプロセスは、通常の SQL 戻りコード 0 を戻して正常に完了しました。統計のサマリーでは、事前パーティション化エージェントは 100,000 行を読み取り、パーティション化エージェントは 100,000 行を分散し、ロード・エージェントによってロードされるすべての行の合計は、100,000 行であることを示します。

例 2

以下の例では、データは PARTITION_ONLY モードで TABLE1 にロードされます。分散出力ファイルは、ディレクトリー /db/data の出力データベース・パーティションのそれぞれに保管されます。

```
load from load.del of del replace into table1 partitioned db config mode
   partition_only part_file_location /db/data
```

LOAD コマンドからの出力は、以下のようになります。

```
Agent Type     Node     SQL Code     Result
_____________________________________________________
LOAD_TO_FILE   000      +00000000    Success.
_____________________________________________________
LOAD_TO_FILE   001      +00000000    Success.
_____________________________________________________
LOAD_TO_FILE   002      +00000000    Success.
_____________________________________________________
LOAD_TO_FILE   003      +00000000    Success.
_____________________________________________________
PARTITION      001      +00000000    Success.
_____________________________________________________
PRE_PARTITION  000      +00000000    Success.
_____________________________________________________

Summary of Partitioning Agents:
Rows Read                    = 100000
Rows Rejected                = 0
Rows Partitioned             = 100000
```

出力では、それぞれの出力データベース・パーティションで実行する load-to-file エージェントがあり、これらのエージェントが正常に実行されたことを示しています。コーディネーター・パーティションには事前パーティション化エージェントがあり、データベース・パーティション 1 で実行するパーティション化エージェントがあります。統計のサマリーでは、事前パーティション化エージェントによって 100,000 行が正常に読み取られ、パーティション化エージェントによって 100,000

行が正常に分散されたことを示しています。表には行がロードされなかったため、ロードされた行の数のサマリーは表示されません。

例 3

上記の PARTITION_ONLY ロード操作時に生成されたファイルをロードするには、以下のコマンドを発行します。

```
load from load.del of del replace into table1 partitioned db config mode
    load_only part_file_location /db/data
```

ロード・コマンドからの出力は、以下のようになります。

```
Agent Type     Node     SQL Code     Result
_____________________________________________________
LOAD           000      +00000000    Success.
_____________________________________________________
LOAD           001      +00000000    Success.
_____________________________________________________
LOAD           002      +00000000    Success.
_____________________________________________________
LOAD           003      +00000000    Success.
_____________________________________________________
RESULTS:       4 of 4 LOADs completed successfully.
_____________________________________________________

Summary of LOAD Agents:
Number of rows read         = 100000
Number of rows skipped      = 0
Number of rows loaded       = 100000
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 100000
```

出力では、それぞれの出力データベース・パーティションのロード・エージェントが正常に実行し、すべてのロード・エージェントによってロードされる行の数の合計が 100,000 であることを示しています。分散は実行されなかったため、分散される行のサマリーはありません。

例 4 - ロード操作の失敗

以下の LOAD コマンドを発行したとします。

```
load from load.del of del replace into table1
```

ロード操作中に、ロード・データベース・パーティションの 1 つが表スペースでスペース不足になっているため、以下の出力が戻されます。

```
SQL0289N  Unable to allocate new pages in table space "DMS4KT".
SQLSTATE=57011

Agent Type     Node     SQL Code     Result
_____________________________________________________________________
LOAD           000      +00000000    Success.
_____________________________________________________________________
LOAD           001      -00000289    Error. May require RESTART.
_____________________________________________________________________
LOAD           002      +00000000    Success.
_____________________________________________________________________
LOAD           003      +00000000    Success.
_____________________________________________________________________
PARTITION      001      +00000000    Success.
_____________________________________________________________________
```

```
PRE_PARTITION  000      +00000000    Success.
_____________________________________________________________________
RESULTS:       3 of 4 LOADs completed successfully.
_____________________________________________________________________

Summary of Partitioning Agents:
Rows Read                   = 0
Rows Rejected               = 0
Rows Partitioned            = 0

Summary of LOAD Agents:
Number of rows read         = 0
Number of rows skipped      = 0
Number of rows loaded       = 0
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 0
```

出力では、ロード操作でエラー SQL0289 が戻されたことを示します。このデータベース・パーティションのサマリーでは、データベース・パーティション 1 でスペースが足りないことを示しています。データベース・パーティション 1 の表スペースのコンテナーにスペースが追加された場合、以下のようにしてロード操作を再始動できます。

```
load from load.del of del restart into table1
```

マイグレーションおよびバージョン互換性

複数パーティション・データベースにおいて DB2® Universal Database™ バージョン 8 以前のロードの動作に戻る場合は、**DB2_PARTITIONEDLOAD_DEFAULT** レジストリー変数を使用できます。

注: バージョン 9.5 以降、DB2_PARTITIONEDLOAD_DEFAULT レジストリー変数は非推奨となり、将来のリリースでは存在しなくなる可能性があります。

複数パーティション・データベースにおいて DB2 UDBバージョン 8 以前の LOAD コマンドの動作に戻ると、パーティション・データベース構成オプションを追加で指定しなくても、有効な分散ヘッダーのあるファイルを単一データベース・パーティションにロードすることができます。これは、**DB2_PARTITIONEDLOAD_DEFAULT** の値を NO に設定することによって行えます。単一データベース・パーティションに LOAD コマンドを発行する既存のスクリプトを変更したくない場合、このオプションを使用することをお勧めします。例えば、4 つのデータベース・パーティションを持つデータベース・パーティション・グループに属する表のデータベース・パーティション 3 に配布ファイルをロードするには、以下のコマンドを発行します。

```
db2set DB2_PARTITIONEDLOAD_DEFAULT=NO
```

それから、DB2 コマンド行プロセッサーから以下のコマンドを発行します。

```
CONNECT RESET

SET CLIENT CONNECT_NODE 3

CONNECT TO DB MYDB

LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

複数パーティション・データベースでは、複数パーティション・データベース・ロード構成オプションが指定されない場合には、表が定義されているすべてのデータベース・パーティションでロード操作が実行されます。入力ファイルには分散ヘッダーは必要ではなく、 MODE オプションはデフォルトの PARTITION_AND_LOAD になります。単一データベース・パーティションをロードするには、OUTPUT_DBPARTNUMS オプションを指定しなければなりません。

第 16 章 パーティション・データベース環境のマイグレーション

パーティション・データベースのマイグレーション

パーティション・データベース環境をマイグレーションするためには、すべてのデータベース・パーティション・サーバーに DB2 バージョン 9.5 をインストールし、インスタンスをマイグレーションし、その後データベースをマイグレーションする必要があります。

カタログ・データベース・パーティション・サーバーやその他のデータベース・パーティション・サーバーからデータベース・パーティション・サーバーのマイグレーションを行うことができます。マイグレーション・プロセスが失敗した場合は、カタログ・データベース・パーティション・サーバーやその他のデータベース・パーティション・サーバーからのマイグレーションを再試行することができます。

この種のマイグレーションは大規模な作業となるため、マイグレーションの手順の説明、前提条件および制約事項は本書の取り扱い範囲を超えています。詳しい説明は、「*マイグレーション・ガイド*」のトピック『パーティション・データベース環境のマイグレーション』で提供されています。この資料から、マイグレーションを実行する前に検討できるトピックを他にも多数参照できます。

第 17 章 スナップショットおよびイベント・モニターの使用

スナップショット・モニター・データを使用したパーティション表の再編成のモニター

以下の情報は、表再編成の全体的な状況をモニターするための便利な方法のいくつかを説明しています。

パーティション表の表の再編成の状況全体を示す、別個のデータ・グループはありません。パーティション表は、データ・パーティションまたは範囲と呼ばれる複数のストレージ・オブジェクトに表データを分割するというデータ編成スキームを使用します。分割は、表の 1 つ以上の表パーティション・キー列の値に従って行われます。しかし、再編成中の個々のデータ・パーティションのデータ・グループ内のエレメントの値から、表再編成の全体的な状況を推察することができます。以下の情報は、表再編成の全体的な状況をモニターするための便利な方法のいくつかを説明しています。

再編成中のデータ・パーティションの数の判別

1 つの表の再編成中のデータ・パーティションの総数は、表名とスキーマ名が同じ表データのモニター・データ・ブロックの数を数えることで判別できます。この値は再編成が開始したデータ・パーティションの数を示します。例 1 と 2 は、3 つのデータ・パーティションが再編成中であることを示します。

再編成中のデータ・パーティションの識別

フェーズの開始時刻 (reorg_phase_start) から、再編成中の現行データ・パーティションを推察することができます。 SORT/BUILD/REPLACE フェーズの間は、再編成中のデータ・パーティションに対応するモニター・データが最新のフェーズ開始時刻を示します。 INDEX_RECREATE フェーズの間は、データ・パーティションのフェーズの開始時刻がすべて同じになります。例 1 と 2 では INDEX_RECREATE フェーズが示されており、すべてのデータ・パーティションの開始時刻が同じになっています。

索引再ビルド要件の識別

再編成中のいずれか 1 つのデータ・パーティションと対応する最大再編成フェーズ・エレメント (reorg_max_phase) の値を入手することにより、索引の再ビルドが必要かどうかを判別することができます。 reorg_max_phase の値が 3 または 4 の場合、索引の再ビルドが必要になります。例 1 および 2 では reorg_max_phase が 3 と報告されており、これは索引の再ビルドが必要であるということを意味しています。

以下の出力例は、3 つのデータ・パーティションを持つ 1 つの表を含んだ、3 ノードで構成されているサーバーからのものです。

```
CREATE TABLE  sales (c1 INT, c2 INT, c3 INT)
    PARTITION BY RANGE (c1)
        (PART P1 STARTING FROM (1)  ENDING AT (10) IN parttbs,
         PART P2 STARTING FROM (11) ENDING AT (20) IN parttbs,
         PART P3 STARTING FROM (21) ENDING AT (30) IN parttbs)
    DISTRIBUTE BY (c2)
```

実行されるステートメント:

```
REORG TABLE sales ALLOW NO ACCESS ON ALL DBPARTITIONNUMS
```

例 1:

```
GET SNAPSHOT FOR TABLES ON DPARTDB GLOBAL
```

出力は関係のある表の情報だけを含むように変更されています。

```
           Table Snapshot

First database connect timestamp     = 06/28/2005 13:46:43.061690
Last reset timestamp                 = 06/28/2005 13:46:47.440046
Snapshot timestamp                   = 06/28/2005 13:46:50.964033
Database name                        = DPARTDB
Database path                        = /work/sales/NODE0000/SQL00001/
Input database alias                 = DPARTDB
Number of accessed tables            = 5

Table List
 Table Schema        = NEWTON
 Table Name          = SALES
 Table Type          = User
 Data Partition Id   = 0
 Data Object Pages   = 3
 Rows Read           = 12
 Rows Written        = 1
 Overflows           = 0
 Page Reorgs         = 0
 Table Reorg Information:
   Node number       = 0
   Reorg Type        =
        Reclaiming
        Table Reorg
        Allow No Access
        Recluster Via Table Scan
        Reorg Data Only
   Reorg Index       = 0
   Reorg Tablespace  = 3
 Long Temp space ID  = 3
   Start Time        = 06/28/2005 13:46:49.816883
   Reorg Phase       = 3 - Index Recreate
   Max Phase         = 3
   Phase Start Time  = 06/28/2005 13:46:50.362918
   Status            = Completed
   Current Counter   = 0
   Max Counter       = 0
   Completion        = 0
   End Time          = 06/28/2005 13:46:50.821244

 Table Reorg Information:
   Node number       = 1
   Reorg Type        =
        Reclaiming
        Table Reorg
        Allow No Access
        Recluster Via Table Scan
        Reorg Data Only
```

```
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:49.822701
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.420741
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.899543


Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:49.814813
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619


Table Schema        = NEWTON
Table Name          = SALES
Table Type          = User
Data Partition Id   = 1
Data Object Pages   = 3
Rows Read           = 8
Rows Written        = 1
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 0
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.014617
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.362918
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.821244

Table Reorg Information:
```

```
  Node number       = 1
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.026278
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.420741
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.899543

Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.006392
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619


Table Schema        = NEWTON
Table Name          = SALES
Table Type          = User
Data Partition Id   = 2
Data Object Pages   = 3
Rows Read           = 4
Rows Written        = 1
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 0
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.199971
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.362918
  Status            = Completed
```

```
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.821244

 Table Reorg Information:
  Node number       = 1
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
 Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.223742
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.420741
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.899543

 Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
 Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.179922
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619
```

例 2:

```
GET SNAPSHOT FOR TABLES ON DPARTDB AT DBPARTITIONNUM 2
```

出力は関係のある表の情報だけを含むように変更されています。

```
            Table Snapshot

First database connect timestamp     = 06/28/2005 13:46:43.617833
Last reset timestamp                 =
Snapshot timestamp                   = 06/28/2005 13:46:51.016787
Database name                        = DPARTDB
Database path                        = /work/sales/NODE0000/SQL00001/
Input database alias                 = DPARTDB
Number of accessed tables            = 3

Table List
 Table Schema        = NEWTON
 Table Name          = SALES
```

```
Table Type          = User
Data Partition Id   = 0
Data Object Pages   = 1
Rows Read           = 0
Rows Written        = 0
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:49.814813
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619


Table Schema        = NEWTON
Table Name          = SALES
Table Type          = User
Data Partition Id   = 1
Data Object Pages   = 1
Rows Read           = 0
Rows Written        = 0
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.006392
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619


Table Schema        = NEWTON
Table Name          = SALES
Table Type          = User
Data Partition Id   = 2
Data Object Pages   = 1
Rows Read           = 4
Rows Written        = 1
```

```
Overflows           = 0
Page Reorgs         = 0
Table Reorg Information:
  Node number       = 2
  Reorg Type        =
       Reclaiming
       Table Reorg
       Allow No Access
       Recluster Via Table Scan
       Reorg Data Only
  Reorg Index       = 0
  Reorg Tablespace  = 3
Long Temp space ID  = 3
  Start Time        = 06/28/2005 13:46:50.179922
  Reorg Phase       = 3 - Index Recreate
  Max Phase         = 3
  Phase Start Time  = 06/28/2005 13:46:50.344277
  Status            = Completed
  Current Counter   = 0
  Max Counter       = 0
  Completion        = 0
  End Time          = 06/28/2005 13:46:50.803619
```

例 3:

```
SELECT * FROM SYSIBMADM.SNAPLOCK WHERE tabname = 'SALES';
```

出力は、関係のある表の情報のサブセットだけを含むように変更されています。

```
...   TBSP_NAME TABNAME LOCK_OBJECT_TYPE   LOCK_MODE  LOCK_STATUS ...
      --------- ------- ------------------ ---------- -----------
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...

 9 record(s) selected.
```

この照会の出力 (続き)。

```
... LOCK_ESCALATION LOCK_ATTRIBUTES DATA_PARTITION_ID DBPARTITIONNUM
    --------------- --------------- ----------------- --------------
...               0 INSERT          2                 2
...               0 NONE            -                 2
...               0 NONE            2                 2
...               0 INSERT          0                 0
...               0 NONE            -                 0
...               0 NONE            0                 0
...               0 INSERT          1                 1
...               0 NONE            -                 1
...               0 NONE            1                 1
```

例 4:

```
SELECT * FROM SYSIBMADM.SNAPTAB WHERE tabname = 'SALES';
```

出力は、関係のある表の情報のサブセットだけを含むように変更されています。

```
... TABSCHEMA TABNAME TAB_FILE_ID TAB_TYPE   DATA_OBJECT_PAGES ROWS_WRITTEN ...
... --------- ------- ----------- ---------- ----------------- ------------ ...
... NEWTON    SALES             2 USER_TABLE                 1            1 ...
```

```
... NEWTON    SALES            4 USER_TABLE                 1            1 ...
... NEWTON    SALES            3 USER_TABLE                 1            1 ...

 3 record(s) selected.
```

この照会の出力 (続き)。

```
... OVERFLOW_ACCESSES PAGE_REORGS DBPARTITIONNUM TBSP_ID DATA_PARTITION_ID
... ----------------- ----------- -------------- ------- -----------------
...                 0           0              0       3                 0
...                 0           0              2       3                 2
...                 0           0              1       3                 1
```

例 5:

```
SELECT * FROM SYSIBMADM.SNAPTAB_REORG WHERE tabname = 'SALES';;
```

出力は、関係のある表の情報のサブセットだけを含むように変更されています。

```
  REORG_PHASE   REORG_MAX_PHASE REORG_TYPE                                    ...
    ------------- --------------- --------------------------------------------
 INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...
 INDEX_RECREATE               3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+DATAONLY  ...

 9 record(s) selected.
```

この照会の出力 (続き)。

```
... REORG_STATUS REORG_TBSPC_ID DBPARTITIONNUM DATA_PARTITION_ID
    ------------------------- -------------- -----------------
... COMPLETED                3              2                 0
... COMPLETED                3              2                 1
... COMPLETED                3              2                 2
... COMPLETED                3              1                 0
... COMPLETED                3              1                 1
... COMPLETED                3              1                 2
... COMPLETED                3              0                 0
... COMPLETED                3              0                 1
... COMPLETED                3              0                 2
```

パーティション・データベース・システムでのグローバル・スナップショット

パーティション・データベース・システムでは、現行パーティション、指定したパーティション、またはすべてのパーティションのスナップショットをとることができます。パーティション・データベースのすべてのパーティションに渡ってグローバル・スナップショットをとる場合は、データが集約されてから、結果が戻されます。

データは、以下のようなさまざまなエレメント・タイプについて集約されます。

- **カウンター、時間、ゲージ**

  インスタンス内のそれぞれのパーティションから収集されたすべての同種値の合計が入っています。例えば、GET SNAPSHOT FOR DATABASE XYZ ON TEST

GLOBAL は、パーティション・データベース・インスタンス内のすべてのパーティションについて、データベースから読み取られた行数 (rows_read) を戻します。

- **水準点**

  パーティション・データベース・システム内のパーティションについて検出される最高値 (高位水準点) または最低値 (低位水準点) を戻します。戻された値に注目する場合は、個々のパーティションのスナップショットを取って、特定のパーティションが使用過剰になっているのか、またはインスタンス全体に問題があるのかを判別することができます。

- **タイム・スタンプ**

  スナップショット・モニター・エージェントがアタッチされているパーティションのタイム・スタンプ値に設定します。すべてのタイム・スタンプ値は、「**タイム・スタンプ**」モニター・スイッチの制御下にあります。

- **情報**

  作業を妨害している可能性のあるパーティションに関する最も重要な情報を戻します。例えば、エレメント `appl_status` について、あるパーティションでの状況が「UOW 実行」であり、別のパーティションでは「ロック待機」の場合には、「ロック待機」が戻されます。なぜなら、これはアプリケーションの実行を保留にしている状況だからです。

さらに、カウンターのリセット、モニター・スイッチの設定、モニター・スイッチ設定値の検索はパーティション・データベース内の個々のパーティション、またはすべてのパーティションに対して行うことができます。

注: グローバル・スナップショットをとる際に、1 つ以上のパーティションでエラーが生じると、データはスナップショットを正常にとることのできたパーティションから収集され、さらに警告 (sqlcode 1629) が戻されます。モニター・スイッチのグローバル取得または更新が失敗したり、 1 つ以上のパーティションでカウンター・リセットが失敗すると、それらのパーティションではスイッチの設定やデータのリセットは行われません。

パーティション・データベース用のイベント・モニターの作成

パーティション・データベース・システムでファイルまたはパイプ・イベント・モニターを作成するときには、収集するモニター・データの有効範囲を決定する必要があります。

パーティション・データベース用のイベント・モニターを作成するには、DBADM 権限が必要です。

イベント・モニターは、オペレーティング・システムのプロセスまたはスレッドを使用して、イベント・レコードを作成します。このプロセスまたはスレッドを実行しているデータベース・パーティションのことを、モニター・パーティションといいます。ファイルおよびパイプ・イベント・モニターは、モニター・パーティション上でローカルに起こったイベントをモニターしたり、DB2 データベース・マネージャーを実行しているすべてのパーティションで起こったイベントをグローバルに

モニターしたりします。グローバル・イベント・モニターは、すべてのパーティションのアクティビティーを含むトレースを、モニター・パーティション上に 1 つ作成します。イベント・モニターがローカルまたはグローバルのどちらであるかは、モニター有効範囲として示します。

モニター・パーティションおよびモニター有効範囲のどちらも、CREATE EVENT MONITOR ステートメントで指定します。

イベント・モニターは、モニター・パーティションがアクティブである場合にのみ活動化できます。イベント・モニターを活動化するために SET EVENT MONITOR ステートメントが使用されたものの、モニター・パーティションがまだアクティブではない場合には、イベント・モニターの活動化はモニター・パーティションが次回開始される時に行われます。また、イベント・モニターが明示的に非活動にされるか、インスタンスが明示的に非活動にされるまで、イベント・モニターの活動化は自動的に行われます。例えば、データベース・パーティション 0 では次のようにします。

```
db2 connect to sample
db2 create event monitor foo ... on dbpartitionnum 2
db2 set event monitor foo state 1
```

上記のコマンドを実行した後、イベント・モニター foo は、データベース sample がデータベース・パーティション 2 でアクティブになる時に自動的にアクティブになります。この自動活動化は、db2 set event monitor foo state 0 が出されるか、パーティション 2 が停止するまで行われます。

表書き込みイベント・モニターについては、ローカルまたはグローバル有効範囲の概念は該当しません。表書き込みイベント・モニターが活動化されているときには、イベント・モニターはすべての パーティションで実行されます。 (より正確に言えば、イベント・モニター処理は、ターゲット表があるデータベース・パーティション・グループに属するパーティションに対して実行されます。) また、イベント・モニター処理が実行するそれぞれのパーティションには、同じターゲット表のセットがあります。これらの表のデータは、それがモニター・データの個々のパーティションのビューを表すという点で異なります。それぞれのパーティションのイベント・モニターのターゲット表の中の目的とする値にアクセスする SQL ステートメントを発行することによって、すべてのパーティションの集約値を入手することができます。

各ターゲット表の最初の列は PARTITION_KEY という名前で、これは表のパーティション・キーとして使用されます。この列の値は、各イベント・モニター・プロセスがそのプロセスが実行されるデータベース・パーティションにデータを挿入できるように選択されます。つまり挿入操作は、イベント・モニター・プロセスが実行されるデータベース・パーティションでローカルに実行されます。どのデータベース・パーティションでも、 PARTITION_KEY フィールドには同じ値が含まれます。このことは、データ・パーティションがドロップされてデータの再分散が行われる場合、ドロップされたデータベース・パーティション上のすべてのデータは均等に分散されるのではなく、他の 1 つのデータベース・パーティションに移動することを意味します。そのため、データベース・パーティションを除去する場合は、そのデータベース・パーティションにある表のすべての行を削除することを事前に検討してください。

その他、表ごとに PARTITION_NUMBER という名前の列を定義することができます。この列には、データが挿入されたパーティションの番号が含まれます。PARTITION_NUMBER 列は PARTITION_KEY 列と違って必須ではありません。

ターゲット表が定義されている表スペースは、イベント・モニター・データが書き込まれるすべてのパーティションに存在しなければなりません。この規則に従わないと、表スペースが存在しない (イベント・モニターがある) ログオン・パーティションにレコードが書き込まれないことになります。ただし、イベントは表スペースが存在するパーティションに書き込まれ、エラーは戻されません。この動作を利用すると、ユーザーは、特定のパーティションにのみ存在する表スペースを作成することにより、モニター用にパーティションのサブセットを選択できることになります。

表書き込みイベント・モニターの活動化の際、 FIRST_CONNECT および EVMON_START に対するコントロール表の各行は、カタログ・データベース・パーティションにのみ挿入されます。そのため、カタログ・データベース・パーティションにコントロール表のための表スペースが存在しなければなりません。そのスペースがカタログ・データベース・パーティションに存在しない場合は、これらの挿入は行われません。

表書き込みイベント・モニターが活動化されるときにパーティションがまだアクティブでない場合は、そのパーティションが次回活動化された時にイベント・モニターが活動化されます。

注: 詳細付きデッドロック接続イベントにおけるロック・リストには、ロックを待機しているパーティション上のアプリケーションによって保留されているロックだけが含まれます。例えば、デッドロックに関係したアプリケーションがノード 20 上でロックを待機している場合、ノード 20 上のそのアプリケーションによって保留されているロックだけがリストに含まれます。

1. モニター対象のパーティションを指定します。

   ```
   CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                       WRITE TO FILE '/tmp/dlevents'
                       ON PARTITION 3
   ```

   dlmon は、イベント・モニターの名前を表します。

   /tmp/dlevents は、イベント・モニターがイベント・ファイルを書き込むディレクトリー・パス (UNIX 上) の名前です。

   3 は、モニター対象のパーティション番号を表します。

2. イベント・モニター・データをローカル有効範囲で収集するか、またはグローバル有効範囲で収集するかを指定します。すべてのパーティションからのイベント・モニター・レポートを収集するには、次のステートメントを発行します。

   ```
   CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                       WRITE TO FILE '/tmp/dlevents'
                       ON PARTITION 3 GLOBAL
   ```

   デッドロックおよび詳細付きデッドロック・イベント・モニターに限り、GLOBAL として定義することができます。すべてのパーティションは、デッドロック関連のイベント・レコードをパーティション 3 に報告します。

3. ローカル・パーティションからのみイベント・モニター・レポートを収集するには、次のステートメントを発行します。

```
CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                     WRITE TO FILE '/tmp/dlevents'
                     ON PARTITION 3 LOCAL
```

   これは、パーティション・データベースでのファイルおよびパイプ・イベント・モニターのデフォルトの動作です。表書き込みイベント・モニターの場合、LOCAL および GLOBAL 節は無視されます。

4. 既存のイベント・モニターのモニター・パーティションおよび有効範囲値を検討することができます。次のステートメントで、SYSCAT.EVENTMONITORS 表を照会して、このことを行います。

```
SELECT EVMONNAME, NODENUM, MONSCOPE FROM SYSCAT.EVENTMONITORS
```

イベント・モニターが作成されて活動化されると、指定されたイベントが発生するたびに、モニター・データを記録します。

第 18 章 望ましいバックアップおよびリカバリー計画の作成

クラッシュ・リカバリー

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) は、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。クラッシュ・リカバリー とは、データベースを整合した使用可能な状態に戻すプロセスのことです。これは、未完了のトランザクションをロールバックし、破損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 45)。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「整合点」にあることになります。

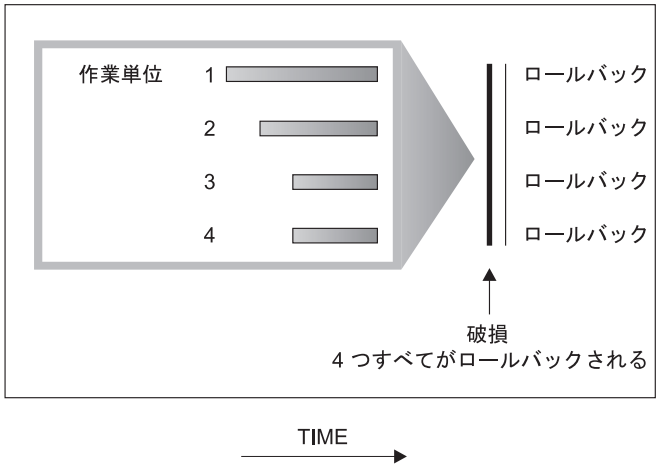


図 45. 作業単位のロールバック (クラッシュ・リカバリー)

トランザクション障害 は、データベースまたはデータベース・マネージャーを異常終了させる重大なエラーまたは状態が起きると発生します。障害が発生した時点で、一部だけ完了している作業単位 (UOW) がディスクにフラッシュされていないと、データベースは矛盾した状態のままになっています。トランザクション障害が発生したら、データベースをリカバリーしなければなりません。以下の条件がトランザクション障害の原因になることがあります。

- マシンの電源障害。そのマシン上のデータベース・マネージャーやデータベース・パーティションがダウンします。
- メモリー破壊、ディスク、CPU、またはネットワーク障害などのハードウェア障害。
- DB2 がダウンするほどの重大なオペレーティング・システム・エラー。
- アプリケーションの異常終了。

障害発生時に未完了になっている作業単位をデータベース・マネージャーによって自動的にロールバックする場合は、自動再始動 (*autorestart*) データベース構成パラメーターを ON に設定し、使用可能にしてください。 (これはデフォルト値です。) 自動再始動を動作したくない場合、 *autorestart* データベース構成パラメーターを

「OFF」に設定してください。結果として、データベース障害の発生時に RESTART DATABASE コマンドを発行する必要があります。破損が発生する前にデータベース入出力が SUSPEND された場合には、クラッシュ・リカバリーが継続するように、RESTART DATABASE コマンドの WRITE RESUME オプションを指定する必要があります。データベースが再始動操作を開始すると、管理通知ログに記録されます。

順方向リカバリーが使用可能になっている (つまり *logarchmeth1* 構成パラメーターが OFF にセットされていない) データベースで、クラッシュ・リカバリーを実行する場合に、個別の表スペースが原因でクラッシュ・リカバリー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り、修復されるまでアクセスできなくなります。クラッシュ・リカバリーは続行します。クラッシュ・リカバリーが完了した時点で、データベースに入っている他の表スペースはアクセス可能であり、データベースへの接続は確立できます。しかしながら、オフラインになった表スペースがシステム・カタログを含む表スペースである場合には、その表スペースは、いずれかの接続が許可される前に修復されなければなりません。

パーティション・データベース環境におけるトランザクション障害のリカバリー

パーティション・データベース環境でトランザクション障害が起きた場合には、通常、障害を引き起こしたデータベース・パーティション・サーバーと、トランザクションに参加していた他のデータベース・パーティション・サーバーとの両方で、データベース・リカバリー処理を実行する必要があります。

- クラッシュ・リカバリーは、障害を引き起こした状態が訂正された後に、障害を引き起こしたデータベース・パーティション・サーバーで実行されます。
- 他の (アクティブのままの) データベース・パーティション・サーバーにおけるデータベース・パーティション・リカバリー処理 は、障害が検出された直後に行われます。

パーティション・データベース環境では、アプリケーションがサブミットされているデータベース・パーティション・サーバーはコーディネーター・パーティションで、最初にアプリケーションの処理を実行するエージェントはコーディネーター・エージェントです。コーディネーター・エージェントは他のデータベース・パーティション・サーバーに対し作業を分配し、どのサーバーがトランザクションに関係するかを追跡します。アプリケーションがトランザクションの COMMIT ステートメントを出すと、コーディネーター・エージェントは 2 フェーズ・コミット・プロトコルを使用してトランザクションをコミットします。最初のフェーズでは、コーディネーター・パーティションはトランザクションに関係している他のすべてのデータベース・パーティション・サーバーに対し PREPARE 要求を配布します。これを受け取ると、これらのサーバーは次のいずれかで応答します。

READ-ONLY
サーバーではデータの変更は行われなかった。

YES サーバーではデータの変更が行われた。

NO エラーが発生したため、サーバーはコミットの準備ができない。

いずれかのサーバーが NO で応答すると、トランザクションはロールバックされます。そうでない場合は、コーディネーター・パーティションは 2 番目のフェーズを開始します。

2 番目のフェーズでは、コーディネーター・パーティションは COMMIT ログ・レコードを書き出した後、YES で応答したすべてのサーバーに対し COMMIT 要求を配布します。他のすべてのデータベース・パーティション・サーバーがコミットを完了すると、それらのサーバーはコーディネーター・パーティションに対し COMMIT の肯定応答を送信します。関係するすべてのサーバーからすべての COMMIT 肯定応答をコーディネーター・エージェントが受け取ると、トランザクションは完了します。この時点で、コーディネーター・エージェントは FORGET ログ・レコードを書き出します。

アクティブ・データベース・パーティション・サーバーにおけるトランザクション障害のリカバリー

データベース・パーティション・サーバーが他のサーバーのダウンを検出すると、障害データベース・パーティション・サーバーと関連するすべての作業は、次のように停止されます。

- アクティブ・データベース・パーティション・サーバーがアプリケーションのコーディネーター・パーティションで、障害データベース・パーティション・サーバー (ただし COMMIT の準備はできていない) でそのアプリケーションが実行されていた場合は、コーディネーター・エージェントは障害リカバリーを実行するための割り込みが行われます。コーディネーター・エージェントが COMMIT 処理の 2 番目のフェーズにある場合は、アプリケーションに SQL0279N が戻されてから、データベース接続が切断されます。そうでない場合は、コーディネーター・エージェントはトランザクションに関係する他のすべてのサーバーに対して ROLLBACK 要求を配布し、 SQL1229N がアプリケーションに戻されます。
- 障害データベース・パーティション・サーバーがアプリケーションのコーディネーター・パーティションであった場合は、アクティブ・サーバーでそのアプリケーションに対し現在でも作業を実行しているエージェントは、障害リカバリーを実行するための割り込みが行われます。トランザクションが準備済みの状態にない各データベース・パーティションでは、トランザクションはローカルにロールバックされます。トランザクションが準備済みの状態にあるデータベース・パーティションでは、トランザクションは未確定になります。コーディネーター・データベース・パーティションが使用可能でないため、コーディネーター・データベース・パーティションは、いくつかのデータベース・パーティションでトランザクションが未確定であることを認識しません。
- 障害データベース・パーティション・サーバーにアプリケーションが接続されていて (障害発生前)、ローカル・データベース・パーティション・サーバーも障害データベース・パーティション・サーバーもコーディネーター・パーティションでない場合は、このアプリケーションの処理を実行しているエージェントは割り込みが行われます。コーディネーター・パーティションは、ROLLBACK または切断メッセージを他のデータベース・パーティション・サーバーに送信します。コーディネーター・パーティションが SQL0279 を戻す場合、トランザクションは依然としてアクティブなデータベース・パーティション・サーバーで単に未確定になるだけです。

障害サーバーに対し要求を送信しようとするプロセス (エージェントまたはデッドロック検出機能) には、要求が送信できない旨のメッセージが送られます。

障害の発生したデータベース・パーティション・サーバーにおけるトランザクション障害のリカバリー

トランザクション障害が発生しデータベース・マネージャーが異常終了した場合、データベース・パーティションが再始動できれば、 RESTART オプションを指定して db2start コマンドを出し、データベース・マネージャーを再始動することができます。データベース・パーティションを再始動できない場合は、別のデータベース・パーティションで db2start を出して、データベース・マネージャーを再始動させることができます。

データベース・マネージャーが異常終了すると、サーバー上のデータベース・パーティションは矛盾状態になることがあります。データベース・パーティションを使用可能にするために、クラッシュ・リカバリーを、データベース・パーティション・サーバー上で次のように起動することができます。

- 明示的に RESTART DATABASE コマンドを使用する。
- *autorestart* データベース構成パラメーターが ON のときは、 CONNECT 要求により暗黙的に開始される。

クラッシュ・リカバリーではアクティブ・ログ・ファイルに含まれるログ・レコードを再適用し、完全に実行されたトランザクションの結果がすべてデータベースに反映されるようにします。変更項目が再適用されると、未確定のトランザクションを除き、コミットされていないすべてのトランザクションがローカルにロールバックされます。パーティション・データベース環境では、2 種類の未確定トランザクションがあります。

- コーディネーター・パーティションではないデータベース・パーティション・サーバーでは、PREPARE 要求に応答していてもまだコミットされていなければ、トランザクションは未確定になります。
- コーディネーター・パーティションでは、コミットされていてもログに完了の印が付けられていなければ (つまり、FORGET レコードがまだ書き出されていない) トランザクションは未確定になります。この状態が発生するのは、コーディネーター・エージェントが、アプリケーションに対して処理実行したすべてのサーバーから、 COMMIT 肯定応答を受け取っていないときです。

クラッシュ・リカバリーでは、以下に述べる処置のいずれかを実行することで、すべての未確定トランザクションの解決を試みます。実行されるアクションは、データベース・パーティション・サーバーがアプリケーションのコーディネーター・パーティションであったかどうかにより異なります。

- 再始動されたサーバーがアプリケーションのコーディネーター・パーティションでない場合は、そのサーバーはコーディネーター・エージェントに照会メッセージを送信し、トランザクションの結果を見つけます。
- 再始動されたサーバーがアプリケーションのコーディネーター・パーティションである 場合、そのサーバーはコーディネーター・エージェントが COMMIT 肯定応答の待ち状態である旨のメッセージを、他のすべてのエージェント (従属エージェント) に送信します。

クラッシュ・リカバリーですべての未確定トランザクションが解決できない場合もあります。たとえば、一部のデータベース・パーティション・サーバーが使用不能の場合などです。コーディネーター・パーティションが、トランザクションにかかわっている他のデータベース・パーティションよりも前にクラッシュ・リカバリーを完了すると、クラッシュ・リカバリーで未確定トランザクションを解決できなくなります。そのような動作になるのは、クラッシュ・リカバリーがデータベース・パーティションごとに独立して実行されるからです。この場合、SQL 警告メッセージ SQL1061W が戻されます。未確定トランザクションはロックおよびアクティブ・ログ・スペースなどのリソースを保留するので、アクティブ・ログ・スペースが未確定トランザクションにより使用されたままになるため、データベースに対して変更を加えられなくなる場合があります。このため、クラッシュ・リカバリー後に未確定トランザクションが残っているかどうかを判別し、未確定トランザクションを解決しなければならないすべてのデータベース・パーティション・サーバーを、できるだけ早期にリカバリーする必要があります。

注: パーティション・データベース・サーバー環境では、RESTART データベース・コマンドはノードごとに実行されます。すべてのノードでデータベースが再始動されるように、以下のコマンドを使用することをお勧めします。

```
db2_all "db2 restart database <database_name>"
```

未確定トランザクションの解決に必要な 1 つまたは複数のサーバーのリカバリーが間に合わない場合に、他のサーバーのデータベース・パーティションにアクセスしなければならないときは、ヒューリスティックな決定を下すことで未確定トランザクションの解決を手作業で行うことができます。 LIST INDOUBT TRANSACTIONS コマンドを使用し、サーバー上の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。

注: 分散トランザクション環境でも、 LIST INDOUBT TRANSACTIONS コマンドは使用されます。 2 種類の未確定トランザクションを区別するために、 LIST INDOUBT TRANSACTIONS コマンドが戻す出力の *originator* フィールドには以下のいずれかが表示されます。

- DB2 Enterprise Server Edition。これは、パーティション・データベース環境で作成されたトランザクションを示しています。
- XA。これは、分散環境で作成されたトランザクションを示しています。

障害のあるデータベース・パーティション・サーバーの識別

データベース・パーティション・サーバーに障害が発生すると、アプリケーションは通常以下のいずれかの SQLCODE を受け取ります。障害が発生したデータベース・マネージャーを検出する方法は、受け取られた SQLCODE により異なります。

SQL0279N
この SQLCODE は、トランザクションに関係するデータベース・パーティション・サーバーが COMMIT 処理中に終了すると受け取られます。

SQL1224N
この SQLCODE は、障害が発生したデータベース・パーティション・サーバーがトランザクションのコーディネーター・パーティションであるときに受け取られます。

SQL1229N

この SQLCODE は、障害が発生したデータベース・パーティション・サーバーがトランザクションのコーディネーター・パーティションでないときに受け取られます。

どのデータベース・パーティション・サーバーに障害が発生したかの判別は、2 つのステップで構成されます。 SQLCODE SQL1229N と関連する SQLCA には、*sqlerrd* フィールドの 6 番目の配列位置にエラーを検出したサーバーのノード番号が入っています。 (サーバーについて書き出されるノード番号は、 db2nodes.cfg ファイルに含まれるノード番号に対応しています。) エラーを検出するデータベース・パーティション・サーバーでは、障害サーバーのノード番号を示すメッセージが管理通知ログに書き込まれます。

注: 複数の論理ノードが 1 つのプロセッサーで使用されている場合は、1 つの論理ノードに障害が発生すると、同じプロセッサー上の他の論理ノードにも障害が発生することがあります。

データベース・パーティション・サーバーの障害からのリカバリー

データベース・パーティション・サーバーの障害からリカバリーするためには、以下のステップを実行します。

1. 障害を引き起こした問題を訂正します。
2. 任意のデータベース・パーティション・サーバーから、 db2start コマンドを出してデータベース・マネージャーを再始動します。
3. 障害のあるデータベース・パーティション・サーバー (複数の場合もある) で、RESTART DATABASE コマンドを出してデータベースを再始動します。

パーティション・データベースの再ビルド

パーティション・データベースを再ビルドするには、各データベース・パーティションを個別に再ビルドします。各データベース・パーティションごとに、カタログ・パーティションを初めとして、必要とするすべての表スペースをまずリストアします。リストアされない表スペースはすべてリストア・ペンディング状態になります。すべてのデータベース・パーティションがリストアされたら、カタログ・パーティションで ROLLFORWARD DATABASE コマンドを発行し、すべてのデータベース・パーティションをロールフォワードします。

注: 再ビルド・フェーズに最初から組み込まれていなかった表スペースを後でリストアする必要がある場合、表スペースを以後ロールフォワードする際に、ロールフォワード・ユーティリティーによってデータベース・パーティション間のすべてのデータが同期化されていることを確認する必要があります。最初のリストアおよびロールフォワード操作時に表スペースが欠落している場合、データにアクセスしようとしてデータ・アクセス・エラーが発生するまで、表スペースが検出されない可能性があります。その場合、欠落している表スペースをリストアしてからロールフォワードし、パーティションの残りの部分と同期するようにしなければなりません。

表スペース・レベルのバックアップ・イメージを使用してパーティション・データベースを再ビルドするには、以下の例を考慮してください。

この例では、以下の 3 つのデータベース・パーティションを含む SAMPLE というリカバリー可能なデータベースが存在します。

- データベース・パーティション 1 には、表スペース SYSCATSPACE、USERSP1、および USERSP2 が含まれます。これはカタログ・パーティションです。
- データベース・パーティション 2 には、表スペース USERSP1 および USERSP3 が含まれます。
- データベース・パーティション 3 には、表スペース USERSP1、USERSP2、および USERSP3 が含まれます。

次のバックアップが取られています。BKxy は、パーティション y のバックアップ番号 x を表します。

- BK11 は、SYSCATSPACE、USERSP1、および USERSP2 のバックアップ
- BK12 は、USERSP2 および USERSP3 のバックアップ
- BK13 は、USERSP1、USERSP2、および USERSP3 のバックアップ
- BK21 は、USERSP1 のバックアップ
- BK22 は、USERSP1 のバックアップ
- BK23 は、USERSP1 のバックアップ
- BK31 は、USERSP2 のバックアップ
- BK33 は、USERSP2 のバックアップ
- BK42 は、USERSP3 のバックアップ
- BK43 は、USERSP3 のバックアップ

以下の手順では、CLP を介して発行される RESTORE DATABASE コマンドおよび ROLLFORWARD DATABASE コマンドを使用し、ログの末尾にデータベース全体を再ビルドすることについて例示しています。

1. データベース・パーティション 1 で、次のように REBUILD オプションを指定して RESTORE DATABASE コマンドを発行します。

   ```
   db2 restore db sample rebuild with all tablespaces in database
      taken at BK31 without prompting
   ```

2. データベース・パーティション 2 で、次のように REBUILD オプションを指定して RESTORE DATABASE コマンドを発行します。

   ```
   db2 restore db sample rebuild with tablespaces in database
      taken at BK42 without prompting
   ```

3. データベース・パーティション 3 で、次のように REBUILD オプションを指定して RESTORE DATABASE コマンドを発行します。

   ```
   db2 restore db sample rebuild with all tablespaces in database
      taken at BK43 without prompting
   ```

4. カタログ・パーティションで、TO END OF LOGS オプションを指定して ROLLFORWARD DATABASE コマンドを発行します。

   ```
   db2 rollforward db sample to end of logs
   ```

5. 次のように、STOP オプションを指定して ROLLFORWARD DATABASE コマンドを発行します。

   ```
   db2 rollforward db sample stop
   ```

この時点で、データベースはすべてのデータベース・パーティションで接続可能であり、すべての表スペースは NORMAL 状態になっています。

db2adutl を使用したデータのリカバリー

以下の例では、db2adutl コマンドと、*logarchopt1* および *vendoropt* データベース構成パラメーターを使用して、ノード間リカバリーを実行する方法を示します。

以下の例で、コンピューター 1 は bar という名前で、AIX が稼働しています。このマシンの所有者は roecken です。 bar 上のデータベースは zample という名前です。コンピューター 2 は dps という名前です。このマシンも AIX が稼働しており、所有者は regress9 です。

PASSWORDACCESS = generate

コンピューター 1

1. データベースのログ・アーカイブを TSM にセットアップします。 zample データベースのデータベース構成パラメーター *logarchmeth1* を更新します。

   ```
   bar:/home/roecken> db2 update db cfg for zample using LOGARCHMETH1 tsm
   ```

   次の情報が戻されます。

   ```
   DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
   ```

   注: データベース構成を更新する前に、データベースのオフライン・バックアップを取っておく必要があるかもしれません。
2. アプリケーションを強制終了します。

   ```
   db2 force applications all
   ```
3. すべてのアプリケーションが強制終了されたことを確認します。

   ```
   db2 list applications
   ```

   データが戻されなかったことを示すメッセージを受け取るはずです。

   注: パーティション・データベース環境では、すべてのデータベース・パーティションでこのステップを実行する必要があります。
4. データベースのバックアップを取ります。

   ```
   db2 backup db zample use tsm
   ```

   次のような情報が戻されます。

   ```
   Backup successful. The timestamp for this backup image is : 20040216151025
   ```

   注: パーティション・データベース環境では、すべてのデータベース・パーティションでこのステップを実行する必要があります。データベース・パーティション上でこのステップを実行する順序は、オンライン・バックアップを実行するか、それともオフライン・バックアップを実行するかによって違います。詳しくは、 465 ページの『バックアップの使用』を参照してください。
5. zample データベースに接続し、そこに表を作成します。
6. 新しい表にデータをロードします。この例では、表を a とし、区切り文字で区切られている ASCII ファイル mr からデータをロードするものとします。COPY

YES オプションを指定することにより、ロードするデータのコピーを作成します。また、USE TSM オプションにより、データのコピーを Tivoli Storage Manager に保管するよう指定します。

注: COPY YES オプションを指定できるのは、データベースがロールフォワード・リカバリーを使用可能な場合だけです。つまり、*logarchmeth1* データベース構成パラメーターが USEREXIT または LOGRETAIN のどちらかに設定されていなければなりません。

```
bar:/home/roecken> db2 load from mr of del modified by noheader replace
   into a copy yes use tsm
```

ユーティリティーは、進行状況を示すために一連のメッセージを戻します。

```
SQL3109N  The utility is beginning to load data from file "/home/roecken/mr".

SQL3500W  The utility is beginning the "LOAD" phase at time "02/16/2004
15:12:13.392633".

SQL3519W  Begin Load Consistency Point. Input record count = "0".

SQL3520W  Load Consistency Point was successful.

SQL3110N  The utility has completed processing.  "1" rows were read from the
input file.

SQL3519W  Begin Load Consistency Point. Input record count = "1".

SQL3520W  Load Consistency Point was successful.

SQL3515W  The utility has finished the "LOAD" phase at time "02/16/2004
15:12:13.445718".


Number of rows read         = 1
Number of rows skipped      = 0
Number of rows loaded       = 1
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 1
```

この時点で、TSM には 1 つのバックアップ・イメージ、1 つのロード・コピー、および 1 つのログ・ファイルがあるはずです。 zample データベースに対する照会は、次のようにして実行できます。

```
bar:/home/roecken/sqllib/adsm> db2adutl query db zample
```

次の情報が戻されます。

```
Retrieving FULL DATABASE BACKUP information.
    1 Time: 20040216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
    Sessions: 1


Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE


Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE


Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE
```

```
Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE


Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE


Retrieving LOAD COPY information.
    1 Time: 20040216151213


Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2004-02-16-15.10.38
```

7. ノード間リカバリーを使用可能にするには、もう一方のノードとアカウントに bar コンピューター上のオブジェクトへのアクセス権を付与する必要があります。この例では、ノード dps とユーザー regress9 にアクセス権を付与します。

```
bar:/home/roecken/sqllib/adsm> db2adutl grant user regress9
   on nodename dps for db zample
```

次の情報が戻されます。

```
Successfully added permissions for regress9 to access ZAMPLE on node dps.
```

db2adutl GRANT 操作の結果を照会するには、次のコマンドを発行します。

```
bar:/home/roecken/sqllib/adsm> db2adutl queryaccess
```

次の情報が戻されます。

```
Node                 Username             Database Name   Type
-------------------------------------------------------------
DPS                  regress9             ZAMPLE          A
-------------------------------------------------------------
 Access Types:  B - backup images   L - logs   A - both
```

PASSWORDACCESS = generate 環境

コンピューター 2

コンピューター 2 (dps) はまだセットアップされていません。 dps の zample データベースに db2adutl 照会を実行すると、次の結果が戻されます。

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
--- Database directory is empty ---
Warning: There are no file spaces created by DB2 on the ADSM server
Warning: No DB2 backup images found in ADSM for any alias.


dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename
    bar owner roecken
--- Database directory is empty ---

Query for database ZAMPLE


Retrieving FULL DATABASE BACKUP information.
    1 Time: 20040216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
    Sessions: 1
```

```
Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE


Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE


Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE


Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE


Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE


Retrieving LOAD COPY information.
    1 Time: 20040216151213


Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2004-02-16-15.10.38
```

dps コンピューターにはまだ zample データベースが存在していません。

1. zample データベースを dps コンピューターにリストアします。

   ```
   dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
   ```

   次の情報が戻されます。

   ```
   DB20000I  The RESTORE DATABASE command completed successfully.
   ```

   注: dps に zample データベースがすでに存在している場合は、 OPTIONS パラメーターを省略し、データベース構成パラメーター *vendoropt* を使用することになります。この構成パラメーターは、バックアップまたはリストア操作の OPTIONS パラメーターをオーバーライドします。

   zample データベースに対してロールフォワード操作を実行すると、失敗することになります。ロールフォワード・ユーティリティーがログ・ファイルを見つけられないためです。ロールフォワード操作の一例は次のとおりです。

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   次のエラーが戻されます。

   ```
   SQL4970N  Roll-forward recovery on database "ZAMPLE" cannot reach the
   specified stop point (end-of-log or point-in-time) because of missing log
   file(s) on node(s) "0".
   ```

2. 強制的にロールフォワード・ユーティリティーに別のマシンでログ・ファイルを探させるため、適切な *logarchopt* 値を構成する必要があります。この状況では、*logarchopt1* データベース構成パラメーターになります。

   ```
   dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
   ```

3. ロールフォワード・ユーティリティーがロード・コピー・イメージを使用できるようにするため、 *vendoropt* データベース構成パラメーターも設定する必要があります。

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
"'-fromnode=bar -fromowner=roecken'"
```

4. ここで zample データベースがロールフォワード可能になります。

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

次の情報が戻されます。

```
                                  Rollforward Status

Input database alias                   = zample
Number of nodes have returned status   = 1

Node number                            = 0
Rollforward status                     = not pending
Next log file to be read               =
Log files processed                    = S0000000.LOG - S0000000.LOG
Last committed transaction             = 2004-02-16-20.10.38.000000 UTC

DB20000I  The ROLLFORWARD command completed successfully.
```

PASSWORDACCESS = prompt 環境

PROMPT 環境では、余分の情報が必要です。特に、オブジェクトを作成したマシンの TSM ノード名とパスワードです。

db2adutl に関して、 dsm.sys ファイル (Windows ベースのプラットフォームでは *dsm.opt* ファイル) を更新し、 NODENAME bar をサーバー節に追加します (bar はソース・コンピューターの名前であるため)。

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename bar
owner roecken password *******
```

次の情報が戻されます。

```
Query for database ZAMPLE


Retrieving FULL DATABASE BACKUP information.
    1 Time: 20040216151025  Oldest log: S0000000.LOG  DB Partition Number: 0
    Sessions: 1


Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for ZAMPLE


Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for ZAMPLE


Retrieving TABLESPACE BACKUP information.
  No TABLESPACE BACKUP images found for ZAMPLE


Retrieving INCREMENTAL TABLESPACE BACKUP information.
  No INCREMENTAL TABLESPACE BACKUP images found for ZAMPLE


Retrieving DELTA TABLESPACE BACKUP information.
  No DELTA TABLESPACE BACKUP images found for ZAMPLE
```

```
Retrieving LOAD COPY information.
    1 Time: 20040216151213


Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, DB Partition Number: 0,
   Taken at: 2004-02-16-15.10.38
```

1. データベースが存在しない場合は、空の zample データベースを作成します。zample データベースがすでに存在している場合は、このステップと、データベース構成を更新するための次の 2 つのステップをスキップできます。

   ```
   dps:/home/regress9> db2 create db zample
   ```

2. zample データベースのデータベース構成パラメーター *tsm_nodename* を更新します。

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
   ```

3. zample データベースのデータベース構成パラメーター *tsm_password* を更新します。

   ```
   dps:/home/regress9> db2 update db cfg for zample using
   tsm_password ********
   ```

4. zample データベースをリストアします。

   ```
   dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
   ```

   リストア操作は正常に完了するものの、警告が出されます。

   ```
   SQL2540W  Restore is successful, however a warning "2523" was
   encountered during Database Restore while processing in No
   Interrupt mode.
   ```

   ここでも、ロールフォワード・ユーティリティーは正しいログ・ファイルを見つけることができません。

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   次のエラー・メッセージが戻されます。

   ```
   SQL1268N  Roll-forward recovery stopped due to error "-2112880618"
   while retrieving log file "S0000000.LOG" for database "ZAMPLE" on node "0".
   ```

5. データベースのリストア操作でデータベース構成ファイルが置き換えられるため、 TSM データベース構成値を正しい値に設定する必要がある。最初に *tsm_nodename* 構成パラメーターを再設定する必要があります。

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
   ```

6. *tsm_password* データベース構成パラメーターを再設定する必要があります。

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_password *******
   ```

7. ロールフォワード・ユーティリティーが正しいログ・ファイルを見つけられるよう、 *logarchopt1* データベース構成パラメーターを再設定する必要があります。

   ```
   dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
   ```

8. ロード・リカバリー・ファイルも使用できるよう、 *vendoropt* データベース構成パラメーターも再設定する必要があります。

   ```
   dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-fromnode=bar -fromowner=roecken'"
   ```

9. データベース構成パラメーターを設定すると、データベースをロールフォワードできるようになります。

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

zample データベースに対して ROLLFORWARD QUERY STATUS コマンドを実行すると、次のように表示されます。

```
                                  Rollforward Status

 Input database alias                   = zample
 Number of nodes have returned status   = 1

 Node number                            = 0
 Rollforward status                     = not pending
 Next log file to be read               =
 Log files processed                    = S0000000.LOG - S0000000.LOG
 Last committed transaction             = 2004-02-16-20.10.38.000000 UTC

DB20000I  The ROLLFORWARD command completed successfully.
```

パーティション・データベース環境におけるクロックの同期化

データベース・パーティション・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、順方向リカバリー性に制限が加わらないようにします。データベース・パーティション・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、パーティション・データベース環境の 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場合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・パーティション・サーバー A のシステム・クロックを、2003 年であるのを間違って 2005 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・パーティション・サーバーのデータベース・パーティション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、2003 年 11 月 7 日から 2005 年 11 月 7 日までの間の任意の時点は、ロールフォワード・リカバリーでは実際には処理不能になります。データベース・パーティション・サーバー A でコミット (COMMIT) が実行されると、データベース・ログのタイム・スタンプは 2005 に設定され、データベース・ログのクロックは 2005 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 (2003 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 時間までです。
- 非カタログ・パーティションに最初の接続要求が出されると、データベース・パーティション・サーバーはその時間をデータベースのカタログ・パーティションに送信します。カタログ・パーティションはその時間を受け取ると、接続を要求するデータベース・パーティション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チェックします。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・パーティション・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・パーティション・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・パーティション・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・パーティション・サーバーへ正しくない時間が伝搬されるのを防ぎます。

第 19 章 トラブルシューティング

DB2 のトラブルシューティング

一般にトラブルシューティングには、問題を切り分け、識別してから、解決方法を探ることが求められます。このセクションでは、DB2 製品の特定のフィーチャーに関係したトラブルシューティング情報を提供します。

共通問題が識別されるにつれて、検出された事項がチェックリストの形でこのセクションに追加されていきます。チェックリストでは解決方法に到達できない場合は、追加の診断データを収集してご自分で分析し、そのデータを IBM ソフトウェア・サポートに分析用に提出することができます。

以下の質問により、適切なトラブルシューティング・タスクに誘導されます。

1. 既知のすべてのフィックスパックを適用しましたか? まだしていないなら、「*DB2 サーバー機能 概説およびインストール*」の『フィックスパックの適用』を検討してください。
2. 問題が発生するのは以下の場合ですか?
   - DB2 データベース・サーバーまたはクライアントのインストール中。この場合は、本書の別の場所にある『インストール問題についてのデータの収集』のトピックを参照してください。
   - インスタンスまたは DB2 管理サーバー (DAS) の作成、ドロップ、更新、またはマイグレーション中。この場合は、本書の別の場所にある『DAS およびインスタンス管理の問題についてのデータの収集』のトピックを参照してください。
   - EXPORT、IMPORT、LOAD、または db2move コマンドを使用してデータを移動中。この場合は、本書の別の場所にある『データ移動問題についてのデータの収集』のトピックを見てください。

問題がこれらのカテゴリーのいずれにも属さなくても、IBM ソフトウェア・サポートに連絡を取る場合には基本的な診断データが必要になることがあります。本書の別の場所にある『DB2 についてのデータの収集』のトピックを参照してください。

パーティション・データベース環境のトラブルシューティング

パーティション・データベース環境でのコマンドの実行

パーティション・データベース環境では、インスタンスにあるコンピューターで、あるいはデータベース・パーティション・サーバー (ノード) で実行するコマンドを発行する場合があります。このような場合には、rah コマンドまたは db2_all コマンドを使用することができます。 rah コマンドを使用すれば、インスタンス内のすべてのコンピューターで実行するコマンドを発行できます。

インスタンス内のデータベース・パーティション・サーバーでコマンドを実行する場合は、 db2_all コマンドを実行します。このセクションでは、これらのコマンドについての概要を説明します。以下の情報は、パーティション・データベース環境だけに適用されます。

注:

1. Linux および UNIX プラットフォームでは、ログイン・シェルを Korn シェルまたは他のシェルにすることができます。ただし、特殊文字を含むコマンドをシェルが処理する方法は、シェルによってさまざまに異なります。
2. また、Linux および UNIX プラットフォームでは、rah は DB2RSHCMD レジストリー変数によって指定されるリモート・シェル・プログラムを使用します。ssh (セキュリティーを追加する場合) または rsh (HP-UX では remsh) の 2 つのリモート・シェル・プログラムの中から選択できます。 ssh リモート・シェル・プログラムは、UNIX オペレーティング・システム環境で平文のパスワードが伝送されるのを回避するために使用されます。このレジストリー変数が設定されない場合、rsh (HP-UX では remsh) が使用されます。
3. Windows で rah コマンドまたは db2_all コマンドを実行するには、管理者グループのメンバーになっているユーザー・アカウントでログオンしなければなりません。

コマンドの有効範囲を調べるには、「*コマンド・リファレンス*」を参照してください。コマンドが 1 つのデータベース・パーティション・サーバーで実行されるか、それともすべてのサーバーで実行されるかが示されます。 1 つのデータベース・パーティション・サーバーで実行されるコマンドをすべてのサーバーで実行させるには、 db2_all を使用してください。 db2trc コマンドは例外で、 1 つのコンピューター上のすべての論理ノード (データベース・パーティション・サーバー) で実行します。すべてのコンピューター上のすべての論理ノードで db2trc を実行する場合は、 rah を使用してください。

第 4 部 パフォーマンスの問題

第 20 章 データベース設計でのパフォーマンスの問題

パフォーマンスを向上させるフィーチャー

表パーティション化とマルチディメンション・クラスタリング表

表では、マルチディメンション・クラスタリングとパーティション化の両方ができます。マルチディメンション・クラスタリングとパーティション化の両方を行った表では、列は表パーティション化の範囲パーティション仕様と MDC キーの両方で使用されます。これは、どちらかが単独で機能するよりも、データ・パーティションの細分性を良くし、ブロックを除去するのに役立ちます。また、表がパーティション化されるよりも、MDC キーに異なる複数の列を指定することが役立つ多くのアプリケーションがあります。なお、MDC はマルチディメンションですが、表パーティション化は複数列であることに注意してください。

主流の DB2 V9.1 データウェアハウスの特性

以下の推奨事項は、DB2 V9.1 にとって新しい、典型的で主流となるウェアハウスに焦点を合わせています。以下のような特性があると想定されています。

- データベースは、複数のマシンまたは複数の AIX 論理パーティションで実行される。
- データベース・パーティション・フィーチャー (DPF) が使用される (表は DISTRIBUTE BY HASH 節を使用して作成される)。
- 4 から 50 個以内のデータ・パーティションがある。
- MDC および表パーティション化が考慮されている表が、主なファクト表である。
- 表には 1 億から 1000 億以内の行がある。
- 新規データは、毎夜、毎週、毎月といった様々な時間フレームでロードされる。
- 毎日の取得ボリュームは、1 万から 1000 万以内のレコードである。
- データ・ボリュームが変化する。最大月は最小月のサイズの 5 倍になります。同様に、最大ディメンション (製品ライン、地域) は 5 倍のサイズ範囲となります。
- 1 から 5 年分の詳細データが保存される。
- 有効期限切れデータは、毎月または四半期ごとにロールアウトされる。
- 表は、広範囲の照会タイプを使用する。しかし、ワークロードはほとんど、OLTP ワークロードに比べると、以下の特性を持つ分析照会です。
  - 200 万行までの、より大きな結果セット
  - ほとんどまたはすべての照会が、基本表ではなくビューをヒットする
- 範囲 (BETWEEN 節)、リストにある項目などでデータを選択する SQL 節。

主流の DB2 V9.1 データウェアハウスのファクト表の特性

典型的なウェアハウスのファクト表は、以下の設計を使用すると考えられます。

- 月列にデータ・パーティションを作成する。
- ロールアウトする期間 (たとえば 1 カ月、3 カ月) ごとに、データ・パーティションを定義する。
- 日および 1 から 4 つ以内の追加のディメンション上に MDC ディメンションを作成する。典型的なディメンションは、製品ラインおよび地域です。
- すべてのデータ・パーティションおよび MDC クラスターが、すべてのデータベース・パーティションに広がっている。

MDC および表パーティション化は、重複した利点のセットを提供します。以下の表では、お客様の組織で必要になる可能性のあるものをリストし、以前に識別された特性を基にして、推奨される編成スキームを識別します。

表 14. MDC 表での表パーティション化の使用

課題	推奨スキーム	推奨
ロールアウト時のデータ可用性	表パーティション化	DETACH PARTITION 節を使用して、中断を最小限にしつつ、大量のデータをロールアウトします。
照会パフォーマンス	表パーティション化および MDC	MDC は複数ディメンションの照会に最善です。表パーティション化は、データ・パーティションの除去に有用です。
再編成の最小化	MDC	MDC はクラスタリングを維持し、再編成の必要性を削減します。
従来のオフライン期間内での、1 カ月かそれ以上のデータのロールアウト	表パーティション化	データ・パーティション化はこの必要を完全に処理します。 MDC は何も追加することはなく、これ自体にはあまり適していません。
マイクロ・オフライン期間 (1 分より小さい) 内での、1 カ月かそれ以上のデータのロールアウト	表パーティション化	データ・パーティション化はこの必要を完全に処理します。 MDC は何も追加することはなく、これ自体にはあまり適していません。
少しもサービスを損失することなく、照会をサブミットするビジネス・ユーザーが、表を完全に使用できるように保ちながら、1 カ月かそれ以上のデータをロールアウトする	MDC	MDC だけが、この必要をいくらか処理します。表パーティション化は、表が短期間オフラインになるので、適切ではありません。
毎日のデータのロード (ALLOW READ ACCESS または ALLOW NO ACCESS)	表パーティション化および MDC	この場合、MDC はほとんどの利点を提供します。表パーティション化は増分的な利点を提供します。

表 14. MDC 表での表パーティション化の使用 (続き)

課題	推奨スキーム	推奨
「継続的な」データのロード (ALLOW READ ACCESS)	表パーティション化および MDC	この場合、MDC はほとんどの利点を提供します。表パーティション化は増分的な利点を提供します。
「従来の BI」照会の場合の照会実行パフォーマンス	表パーティション化および MDC	MDC は、キューブ/複数ディメンションの照会に最適です。表パーティション化は、パーティションの除去の点で補助します。
再編成の必要を避けたり、タスクの実行に関連した手間を削減することにより、再編成の手間を最小化にする	MDC	MDC はクラスタリングを維持して REORG の必要性を削減します。 MDC が使用される場合、データ・パーティション化は増分的な利点を提供しません。しかし、MDC が使用されない場合には、範囲パーティション化が、パーティション・レベルで何らかの過程の粗いクラスタリングを維持することによって、REORG の必要を削減するのに役立ちます。

例 1:

キー列 YearAndMonth および Province のある表を考慮します。この表のプランとして妥当な方法は、1 つのデータ・パーティションあたり 2 カ月で、日付によってパーティション化することです。加えて、 37 ページの図 6 に示されているように、任意の 2 カ月の日付範囲内にある特定の州のすべての行は一緒にクラスター化されるので、Province によって編成することもできます。

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

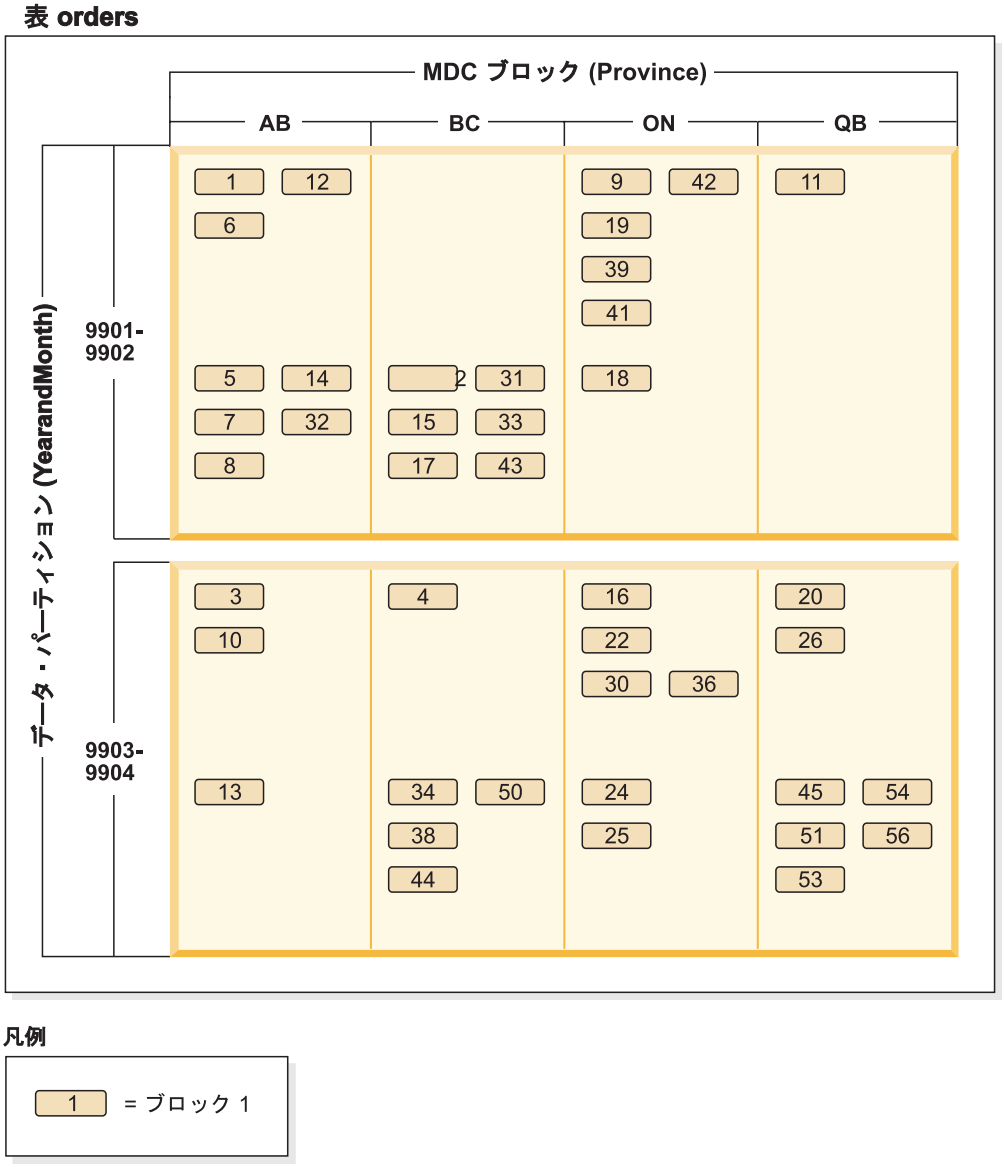


図 46. YearAndMonth によってパーティション化され、Province によって編成される表

例 2:

38 ページの図 7 に示されているように、YearAndMonth を ORGANIZE BY 節に追加することによって、より良い細分化を行うことができます。

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

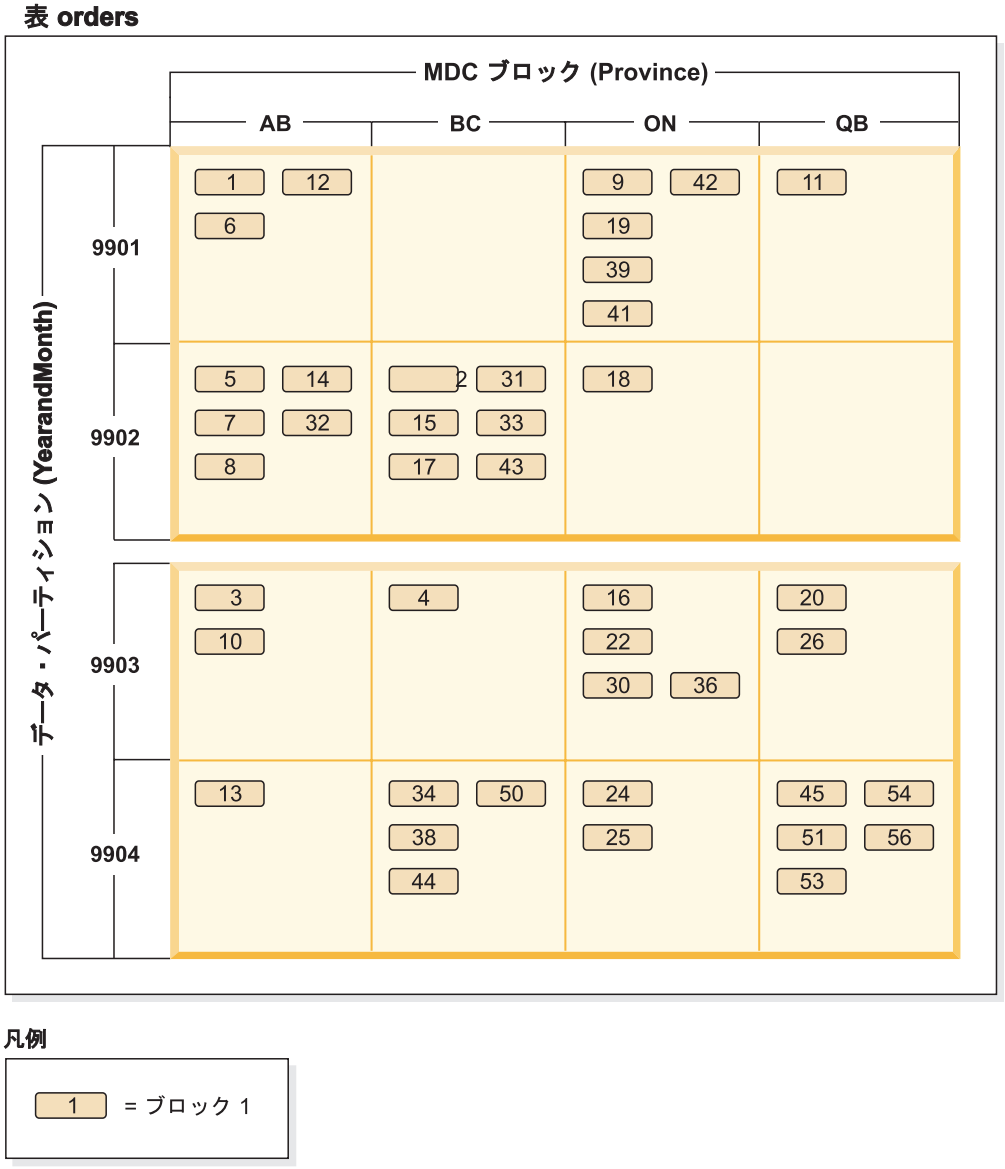


図 47. YearAndMonth によってパーティション化され、Province および YearAndMonth によって編成される表

各範囲に単一値のみがあるようなパーティション化の場合、MDC キーに表パーティション列を含めても、何も得られません。

考慮事項

- 基本表と比較して、MDC 表およびパーティション表は多くのストレージを必要とします。これらのストレージ必要量は付加的なものですが、利点を考えると妥当なものと考えられます。
- パーティション・データベース環境で表パーティション化と MDC 機能を組み合わせないことを選択するなら、確信をもってデータ配分を予測できるような場合 (一般的にここで説明されているシステムのタイプの場合) には、表パーティション化が最善です。そうでない場合には、MDC を考慮する必要があります。

パーティション表の最適化ストラテジー

照会述部に基づき、照会に応答するためにアクセスする必要があるのは表のデータ・パーティションのサブセットのみであると判断するデータベース・サーバーの機能のことです。データ・パーティションの除去は、パーティション表に対して意思決定支援照会を実行する際に特に役立ちます。

パーティション表は、データ・パーティションまたは範囲と呼ばれる複数のストレージ・オブジェクトに表データを分割するというデータ編成スキームを使用します。分割は、表の 1 つ以上の表パーティション・キー列の値に従って行われます。指定された表のデータは、CREATE TABLE ステートメントの PARTITION BY 節で提供された仕様に基づいて、複数のストレージ・オブジェクトにパーティション化されます。このストレージ・オブジェクトは異なる表スペース、同じ表スペース内、またはその両方に配置することができます。

以下の例は、データ・パーティションの除去のパフォーマンス上の利点について示しています。次のステートメントを発行した場合について考慮してみます。

```
CREATE TABLE custlist(subsdate DATE, Province CHAR(2), AccountID INT)
PARTITION BY RANGE(subsdate)
(STARTING FROM '1/1/1990' IN ts1,
STARTING FROM '1/1/1991' IN ts1,
STARTING FROM '1/1/1992' IN ts1,
STARTING FROM '1/1/1993' IN ts2,
STARTING FROM '1/1/1994' IN ts2,
STARTING FROM '1/1/1995' IN ts2,
STARTING FROM '1/1/1996' IN ts3,
STARTING FROM '1/1/1997' IN ts3,
STARTING FROM '1/1/1998' IN ts3,
STARTING FROM '1/1/1999' IN ts4,
STARTING FROM '1/1/2000' IN ts4,
STARTING FROM '1/1/2001' ENDING '12/31/2001' IN ts4);
```

2000 年の顧客情報に関心があるとします。以下の照会を発行する場合について考えます。

```
SELECT * FROM custlist WHERE subsdate BETWEEN '1/1/2000' AND '12/31/2000';
```

313 ページの図 48 に示されているように、データベース・サーバーはこの照会を解決するために、表スペース 4 (ts4) の 1 つのデータ・パーティションのみにアクセスする必要があると判断します。

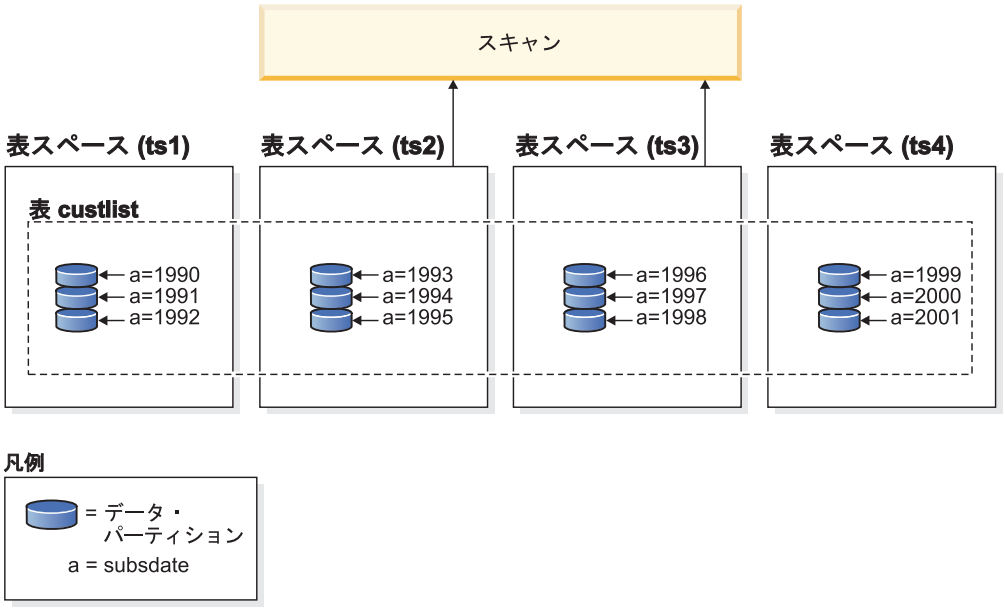


図 48. パーティション表におけるデータ・パーティションの除去のパフォーマンス上の利点

図 49 に示されているデータ・パーティションの除去の別の例は、2 つの索引が関係する索引スキャンで以下のスキームに基づいています。

```
CREATE TABLE multi (sale_date date, region char(2))
PARTITION BY (sale_date)
(STARTING '01/01/2005' ENDING '12/31/2005' EVERY 1 MONTH);
CREATE INDEX sx ON multi(sale_date);
CREATE INDEX rx ON multi(region);
```

以下の照会を発行する場合について考えます。

```
SELECT * FROM multi WHERE
sale_date BETWEEN '6/1/2005' AND '7/31/2005' AND REGION = 'NW';
```

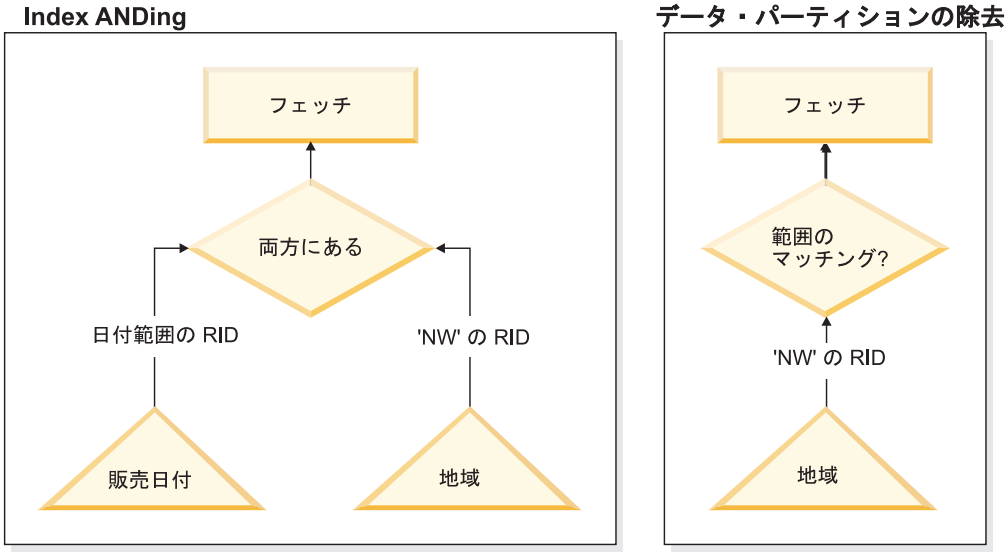


図 49. 表のパーティション化と索引 AND 演算の両方のオプティマイザーの決定パス

表のパーティション化を使用しないで考えられる 1 つのプランは、索引 AND 演算です。索引 AND 演算は、以下のタスクを実行します。

- 各索引から関連するすべての索引項目を読み取る
- 行 ID (RID) の両方のセットを保管する
- RID を突き合わせて、両方の索引のどちらで生じたかを判別する
- RID を使用して行を取り出す

313 ページの図 49 で示されているように、表のパーティション化を使用すると、region と sale_date の両方での一致を検出するために索引が読み取られ、それにより一致する行を素早く取得することができます。

DB2 Explain

DB2 Explain を使用して、 DB2 オプティマイザーが選択したパーティションの除去を判別することもできます。**DP Elim Predicates** 情報には、以下の照会を解決するためにどのデータ・パーティションがスキャンされるかが示されます。

```
SELECT * FROM custlist WHERE subsdate
BETWEEN '12/31/1999' AND '1/1/2001'

Arguments:
---------
DPESTFLG: (Number of data partitions accessed are Estimated)
    FALSE
DPLSTPRT: (List of data partitions accessed)
    9-11
DPNUMPRT: (Number of data partitions accessed)
   3

DP Elim Predicates:
------------------
Range 1)
  Stop  Predicate: (Q1.A <= '01/01/2001')
 Start Predicate: ('12/31/1999' <= Q1.A)



Objects Used in Access Plan:
 ---------------------------

  Schema: MRSRINI
  Name:    CUSTLIST
  Type:    Data Partitioned Table
   Time of creation:      2005-11-30-14.21.33.857039
   Last statistics update:    2005-11-30-14.21.34.339392
    Number of columns:      3
    Number of rows:      100000
   Width of rows:      19
   Number of buffer pool pages:    1200
    Number of data partitions:    12
    Distinct row values:      No
    Tablespace name:      <VARIOUS>
```

複数列のサポート

データ・パーティションの除去は、表パーティション・キーとして複数列が使用されている場合に向いています。

たとえば、以下のステートメントを発行した場合について考えています。

```
CREATE TABLE sales(year INT, month INT)
 PARTITION BY RANGE(year, month)
 (STARTING FROM (2001, 1) ENDING AT(2001,3) IN ts1,
 ENDING AT(2001,6) IN ts2,
 ENDING AT(2001,9) IN ts3,
 ENDING AT(2001,12) IN ts4,
 ENDING AT(2002,3) IN ts5,
 ENDING AT(2002,6) IN ts6,
 ENDING AT(2002,9) IN ts7,
 ENDING AT(2002,12) IN ts8)
```

次に、以下の照会を発行します。

```
SELECT * FROM sales WHERE year = 2001 AND month < 8
```

照会オプティマイザーは、この照会を解決するために ts1、ts2 および ts3 内のデータ・パーティションのみにアクセスする必要があると推論します。

注: 表パーティション・キーが複数列から構成されている場合、複合キーの先頭列に述部がある場合に限ってデータ・パーティションの除去が可能です。表パーティション・キーとして使用される非先頭列は独立していないからです。

複数範囲のサポート

複数範囲 (が OR で結ばれたもの) を持つデータ・パーティションでデータ・パーティションの除去を行うことが可能です。前述の例で作成された表を使用して、以下の照会を実行します。

```
SELECT * FROM sales
WHERE (year = 2001 AND month <= 3) OR (year = 2002 and month >= 10)
```

データベース・サーバーは、2001 年の最初の四半期と 2002 年の最後の四半期のデータのみにアクセスします。

生成列

生成列を表パーティション・キーとして使用できます。

たとえば、以下のステートメントを発行できます。

```
CREATE TABLE sales(a INT, b INT GENERATED ALWAYS AS (a / 5))
 IN ts1,ts2,ts3,ts4,ts5,ts6,ts7,ts8,ts9,ts10
 PARTITION BY RANGE(b)
 (STARTING FROM (0) ENDING AT(1000) EVERY (50))
```

この場合、生成列の述部がデータ・パーティションの除去に使用されます。加えて、列を生成するために使用される式が単調の場合、データベース・サーバーはソース列上の述部を生成列上の述部に変換し、生成列でデータ・パーティションの除去を可能にします。

たとえば、以下の照会があるとします。

```
SELECT * FROM sales WHERE a > 35
```

データベース・サーバーは、a (a > 35) から b (b > 7) に追加の述部を生成し、データ・パーティションの除去が可能になります。

結合述部

結合述部が表アクセス・レベルに下がると、結合述部をデータ・パーティションの除去に使用することもできます。結合述部は、ネストされたループ結合 (NLJN) の内部の表アクセス・レベルにのみ下がります。

次の表を考慮してください。

```
CREATE TABLE T1(A INT, B INT)
 PARTITION BY RANGE(A, B)
 (STARTING FROM (1, 1)
 ENDING (1,10) IN ts1, ENDING (1,20) IN ts2,
 ENDING (2,10) IN ts3, ENDING (2,20) IN ts4,
  ENDING (3,10) IN ts5, ENDING (3,20) IN ts6,
 ENDING (4,10) IN ts7, ENDING (4,20) IN ts8)

CREATE TABLE T2 (A INT, B INT)
```

使用する述部は以下のとおりです。

```
P1: T1.A = T2.A
P2: T1.B > 15
```

この例では、結合の外部値が不明なため、コンパイル時にアクセスされるデータ・パーティションを正確に判別することはできません。この場合、ホスト変数またはパラメーター・マーカーが使用される場合と同様に、必要な値が結合される実行時にデータ・パーティションの除去が生じます。

実行時に T1 が NLJN の内部にある場合、述部に基づいて、T2.A のすべての外部値に対してデータ・パーティションの除去が動的に生じます。実行時に、述部 T1.A = 3 および T1.B > 15 が外部値 T2.A = 3 に対して適用され、表スペース ts6 と ts7 のデータ・パーティションにアクセスする資格を与えます。

表 T1 と T2 の列 A に以下の値がある場合を考えてみてください。

外部表 T2: 列 A	内部表 T1: 列 A	内部表 T1: 列 B	内部表 T1: データ・パーティション・ロケーション
2	3	20	ts6
3	2	10	ts3
3	2	18	ts4
	3	15	ts6
	1	40	ts3

(内部表に対する表スキャンを前提として) ネスト・ループ結合を実行するために、データベース・マネージャーは以下のステップを実行します。

1. T2 から最初の行を読み取ります。A の値は 2 です。
2. T2.A の値 (2) を結合述部 T1.A = T2.A で列 T2.A にバインドします。述部は T1.A = 2 となります。
3. 述部 T1.A = 2 および T1.B > 15 を使用して、データ・パーティションの除去を適用します。これにより、表スペース ts4 と ts5 のデータ・パーティションが資格を得ます。

4. T1.A = 2 および T1.B > 15 の適用後に行が検出されるまで、表 T1 の表スペース ts4 と ts5 のデータ・パーティションをスキャンします。資格にかなう最初の検出行は、T1 の行 3 です。
5. 一致した行を結合します。
6. 次の一致 (T1.A = 2 および T1.B > 15) が検出されるまで、表 T1 の表スペース ts4 と ts5 のデータ・パーティションをスキャンします。それ以上、行は見つかりません。
7. T2 のすべての行がなくなるまで、T2 の次の行 (A の値を 3 に置換する) に対してステップ 1 から 6 までを繰り返します。

MDC 表の最適化ストラテジー

マルチディメンション・クラスタリング (MDC) 表を作成した場合、オプティマイザーは追加の最適化ストラテジーを適用できるので、多くの照会のパフォーマンスが向上する可能性があります。これらのストラテジーは主にブロック索引の効率が改善されたことに基づいていますが、複数ディメンションでのクラスタリングによる利点もデータ検索の高速化を可能にしています。

注: MDC 表の最適化ストラテジーは、パーティション内並列処理およびパーティション間並列処理によるパフォーマンス上の利点もインプリメントすることができます。

MDC 表による以下の特定の利点を検討してください。

- ディメンション・ブロック索引の参照数により、表の必要な部分を識別して必要なブロックだけを高速にスキャンすることができます。
- ブロック索引は RID 索引よりも小さいので、参照数はより高速になります。
- 索引の AND 操作および OR 操作をブロック・レベルで実行して、 RID と結合することができます。
- データはエクステント上でクラスター化されることが保証されているので、検索がより高速になります。
- ロールアウトを使用できるときに行の削除が高速になります。

SALES という名前で、ディメンションが **region** および **month** 列に定義されている MDC 表についての次の簡単な例を検討してください。

```
SELECT * FROM SALES
    WHERE MONTH='March' AND REGION='SE'
```

この照会では、オプティマイザーはディメンション・ブロック索引のルックアップを実行して、 month が March で region が SE のブロックを検索することができます。その後、その結果となる表のブロックだけを高速にスキャンして、結果セットを取り出すことができます。

ロールアウト削除

ロールアウトを使用する削除が許可される条件が満たされれば、MDC 表から行を削除するための、さらに効果的な方法が使用されます。条件は以下のとおりです。

- DELETE ステートメントが検索されたが、位置を特定できない (つまり、『WHERE CURRENT OF』 節を使用しない)。

- WHERE 節がない (すべての行が削除される) または WHERE 節の条件だけがディメンションにある。
- 表が DATA CAPTURE CHANGES 節で定義されていない。
- 表が参照整合性リレーションシップで親でない。
- 表に削除トリガーが定義されていない。
- 表が即時に更新される MQT で使用されない。
- カスケード削除操作がロールアウトできる (その外部キーがその表のディメンション列のサブセットである場合)。
- DELETE ステートメントは、(CREATE TRIGGER ステートメント上の OLD TABLE AS 節により指定される) トリガー SQL 操作の前に、影響を受ける一連の行を識別する一時表に対して実行される SELECT ステートメントには入れることができません。

ロールアウト削除の場合、削除レコードはログに記録されません。その代わり、レコードの入ったページはページのパーツを再フォーマットすることによって空にされます。再フォーマットされたパーツに対する変更はログに記録されますが、レコード自体はログに記録されません。

デフォルトの動作である*即時クリーンアップ・ロールアウト* は、削除時に RID 索引をクリーンアップするためのものです。このモードは、**DB2_MDC_ROLLOUT** レジストリー変数を IMMEDIATE に設定するか、または IMMEDIATE を SET CURRENT MDC ROLLOUT MODE ステートメントで指定することでも指定できます。標準の削除と比較して、索引の更新のロギングに変更がない場合、パフォーマンスの向上は RID 索引の数によって異なります。 RID 索引が少ないと、合計時間およびログ・スペースの割合が改善されます。

ログに保存されるスペース量の見積もりは、この公式で出すことができます。ここで、N は削除されるレコードの数であり、S は削除されるレコードの合計サイズ (NULL 標識および varchar の長さなどのオーバーヘッドを含む) であり、P は削除されるレコードの入ったブロックのページ数です。

```
S + 38*N - 50*P
```

この数値は、実際のログ・データを縮約したものです。アクティブ・ログ保存でのスペース所要量は、ロールバックに予約されているスペースの保存のため 2 倍になります。

代替方法として、*据え置きクリーンアップ・ロールアウト* を使用して、トランザクション・コミット後に RID 索引を更新できます。このモードは、**DB2_MDC_ROLL_OUT** レジストリー変数を DEFER に設定するか、または DEFERRED を SET CURRENT MDC ROLLOUT MODE ステートメントで指定することでも指定できます。据え置きロールアウトで、RID 索引は削除のコミット後に、バックグラウンドで非同期にクリーンアップされます。このロールアウトのメソッドは、非常に大規模な削除の場合、または表に多数の RID 索引が存在する場合は結果としてかなり高速な削除となります。即時索引クリーンアップでは索引内の各行は 1 つずつクリーンアップされるのに対し、据え置き索引クリーンアップ中に索引は並列でクリーンアップされるので、クリーンアップ操作全体の速度は向上します。さらに、非同期索引クリーンアップでは、索引キーではなく索引ページによ

り索引更新をログ記録するので、DELETE ステートメントのトランザクション・ログスペース所要量は大幅に削減されます。

注: 据え置きクリーンアップ・ロールアウトには、データベース・ヒープから取られる追加のメモリー・リソースが必要です。 DB2 が必要とするメモリー構造を割り振れない場合、据え置きクリーンアップ・ロールアウトは失敗し、メッセージが管理者ログに書き込まれます。

据え置きクリーンアップ・ロールアウトを使用する状況

削除のパフォーマンスが最も重要な要因であり、RID 索引が表に定義されている場合は、据え置きクリーンアップ・ロールアウトを使用します。索引クリーンアップの前に、ロールアウトされたブロックの索引ベースのスキャンは、ロールアウトされたデータの量に応じてパフォーマンスがやや低下します。即時索引クリーンアップと据え置き索引クリーンアップとを決定する場合に考慮すべき、以下のような他の問題があります。

- 削除のサイズ: 非常に大規模な削除に対しては、据え置きクリーンアップ・ロールアウトを選択します。ディメンション削除ステートメントが数多くの小さな MDC 表に対して発行される場合は、非同期に索引オブジェクトをクリーンアップするためのオーバーヘッドが、削除中の時間の節約という利点を上回ってしまう可能性があります。
- 索引の数およびタイプ: 表に行レベルの処理を必要とする数多くの RID 索引が含まれている場合は、据え置きクリーンアップ・ロールアウトを使用します。
- ブロック可用性: 削除ステートメントのコミット直後に、その削除ステートメントにより解放されるブロック・スペースを使用できるようにするには、即時クリーンアップ・ロールアウトを使用します。
- ログ・スペース: ログ・スペースが制限されている場合、大規模削除の据え置きクリーンアップ・ロールアウトを使用します。
- メモリー制約: 据え置きクリーンアップ・ロールアウトは、据え置きクリーンアップが保留中のすべての表の追加のデータベース・ヒープを消費します。

削除でロールアウト動作を無効にするには、**DB2_MDC_ROLLOUT** レジストリー変数を OFF に設定するか、または `NONE` を SET CURRENT MDC ROLLOUT MODE ステートメントで指定できます。

第 21 章 索引

パーティション表の索引

パーティション表での索引の動作について

パーティション表の索引は通常の表の索引と似ていて、各索引には表のすべてのデータ・パーティション内の行に対するポインターが含まれています。しかし重要な 1 つの相違点は、パーティション表の各索引は独立したオブジェクトであるということです。パーティション化されたデータベース環境では、表と同じ方法で索引はデータベース・パーティションに分散されます。パーティション表の索引は他の索引に対して独立して動作できるので、パーティション表に索引を作成する際には、どの表スペースを使用するかを特に考慮する必要があります。

パーティション表上の索引は、表のデータ・パーティションが複数の表スペースにまたがっている場合であっても、単一の表スペースに作成されます。DMS 表スペースと SMS 表スペースは両方とも、表とは別の場所にある索引の使用をサポートしています。指定されたすべての表スペースは、同一のデータベース・パーティション・グループになければなりません。各索引は、LARGE 表スペースを含めて独自の表スペースに配置できます。各索引表スペースは、データ・パーティションとして DMS か SMS のどちらかの同一のストレージ・メカニズムを使用しなければなりません。 LARGE 表スペース内の索引は、最大 2^29 ページまで含めることが可能です。

パーティション表上の索引に関する別の利点には、以下のものがあります。

- 索引のドロップおよびオンラインの索引の作成に関するパフォーマンスが向上します。
- 表の各索引における表スペース特性に異なる値を使用できます (たとえば、スペースの使用効率をより良いものにするには、各索引でページ・サイズを異ならせるのが適切な場合があります)。
- 入出力競合を削減して、表の索引データに効率的な同時アクセスができます。
- 個々の索引をドロップすると、索引再編成を行わなくてもシステムがスペースをすぐに使用できます。
- 索引再編成を実行することを選択した場合、個々の索引を再編成することができます。

322 ページの図 50 は、1 つの表スペースに内在する、パーティション表上のパーティション化されていない索引を示しています。

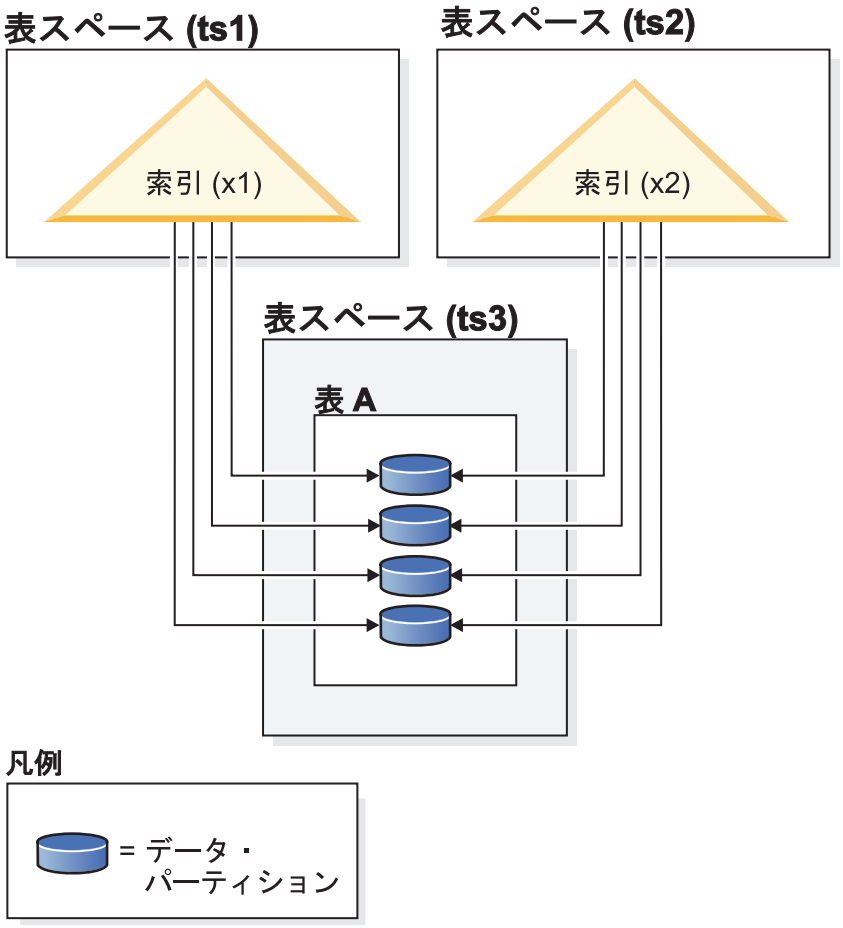


図 50. パーティション表上の索引の動作

323 ページの図 51 は、複数のデータベース・パーティションにも分散されている、パーティション表上の索引の動作を示しています。

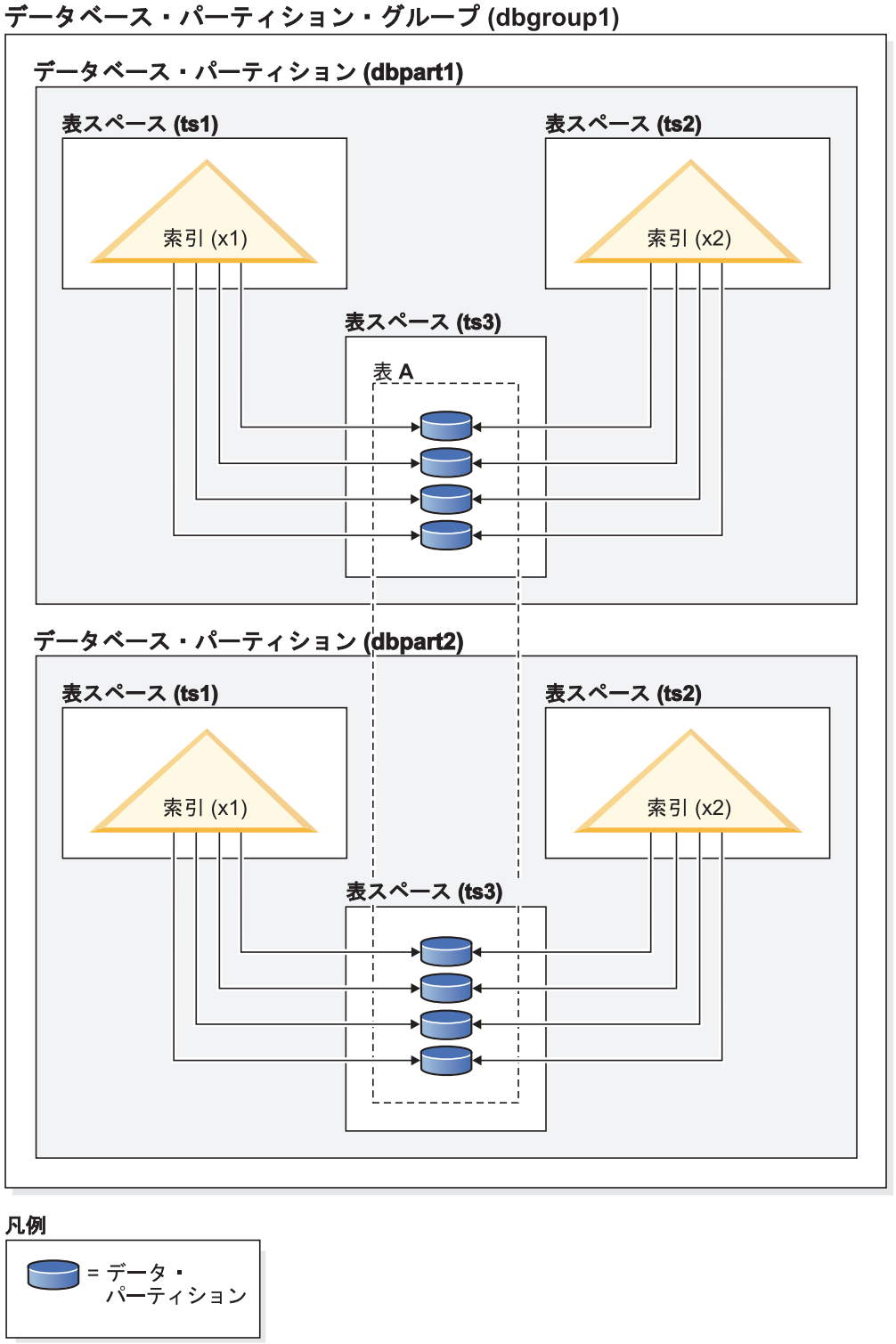


図 51. 分散とパーティション化の両方が行われている表上の索引の動作。

CREATE INDEX ...IN <tbspace1> ステートメントでパーティション表に索引表スペースを指定でき、これは CREATE TABLE .. INDEX IN <tbspace2> ステートメントで指定した索引表スペースとは異なるものにすることも可能です。

パーティション表に限り、CREATE INDEX ステートメントで IN 節を使用して索引位置をオーバーライドでき、それにより索引の表スペース位置を指定することができます。この方法により、必要に応じて、別の表スペース内に、1 つのパーティション表上の複数の異なる索引を配置することができます。パーティション化されていない索引を配置する場所を指定しないでパーティション表を作成し、特定の表スペースを指定しない CREATE INDEX ステートメントを使用して索引を作成する場合、最初のアタッチされたデータ・パーティションまたは表示可能なデータ・パーティションの表スペースに索引が作成されます。次の 3 つの考え得る各ケースをケース 1 から順番に評価して、索引が作成される場所を判別します。この評価は、いずれかのケースに一致すると停止します。

```
ケース 1:
When an index table space is specified in CREATE INDEX ... IN
<tbspace1> statement the table space specified in <tbspace1>
is used for this index.
ケース 2:
When an index table space is specified in the CREATE TABLE .. INDEX IN
<tbspace2> statement the table space specified in <tbspace2>
is used for this index.
ケース 3:
When no table space is specified, use the table space used by the first
attached or visible data partition.
```

索引が作成される場所は、CREATE TABLE ステートメントの実行時に何が行われるかによって異なります。パーティション化されていない表の場合、INDEX IN 節を指定しないと、データベースがユーザーに代わって指定を行い、それはユーザーのデータ表スペースと同じになります。パーティション表の場合には、ブランクのままにするとブランクとして残り、ケース 3 が当てはまります。

例 1: この例では、パーティション表 sales (a int、b int、c int) が存在することを想定し、表スペース 'ts1' にユニーク索引 'a_idx' を作成します。

```
CREATE UNIQUE INDEX a_idx ON sales ( a ) IN ts1
```

例 2: この例では、パーティション表 sales (a int、b int、c int) が存在することを想定し、表スペース 'ts2' に索引 'b_idx' を作成します。

```
CREATE INDEX b_idx ONsales ( b ) IN ts2
```

パーティション表でのクラスタリング索引の動作について

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

任意のクラスタリング・キーを使用して、パーティション表にクラスタリング索引を作成できます。データベース・サーバーは、クラスタリング索引を使用して、各データ・パーティション内でデータをローカルにクラスター化しようとします。クラスター化された挿入の際、索引でルックアップが行われ、適切な行 ID (RID) を検索します。この RID は、表内のスペースを検索してレコードを挿入する際の開始点として使用されます。効率が良く、十分に保守されたクラスタリングを実行するには、索引列と表パーティション・キー列間に相関がなければなりません。そのような相関を確保する 1 つの方法は、以下の例に示されているように、表パーティション・キー列ごとに索引列を先頭に付けることです。

```
PARTITION BY RANGE (Month, Region)
CREATE INDEX ...(Month, Region, Department) CLUSTER
```

データベース・サーバーがこの相関を強制することはありませんが、適切なクラスタリングを行うために索引中のすべてのキーがパーティション ID ごとに互いにグループ化されていることが期待されています。たとえば、表を四半期でパーティション化し、クラスタリング索引を日付に定義すると、その四半期と日付間には関連があるため、効率のよい最適なデータのクラスタリングを行うことが可能です。データ・パーティションのすべてのキーが索引内で相互にグループ化されているからです。

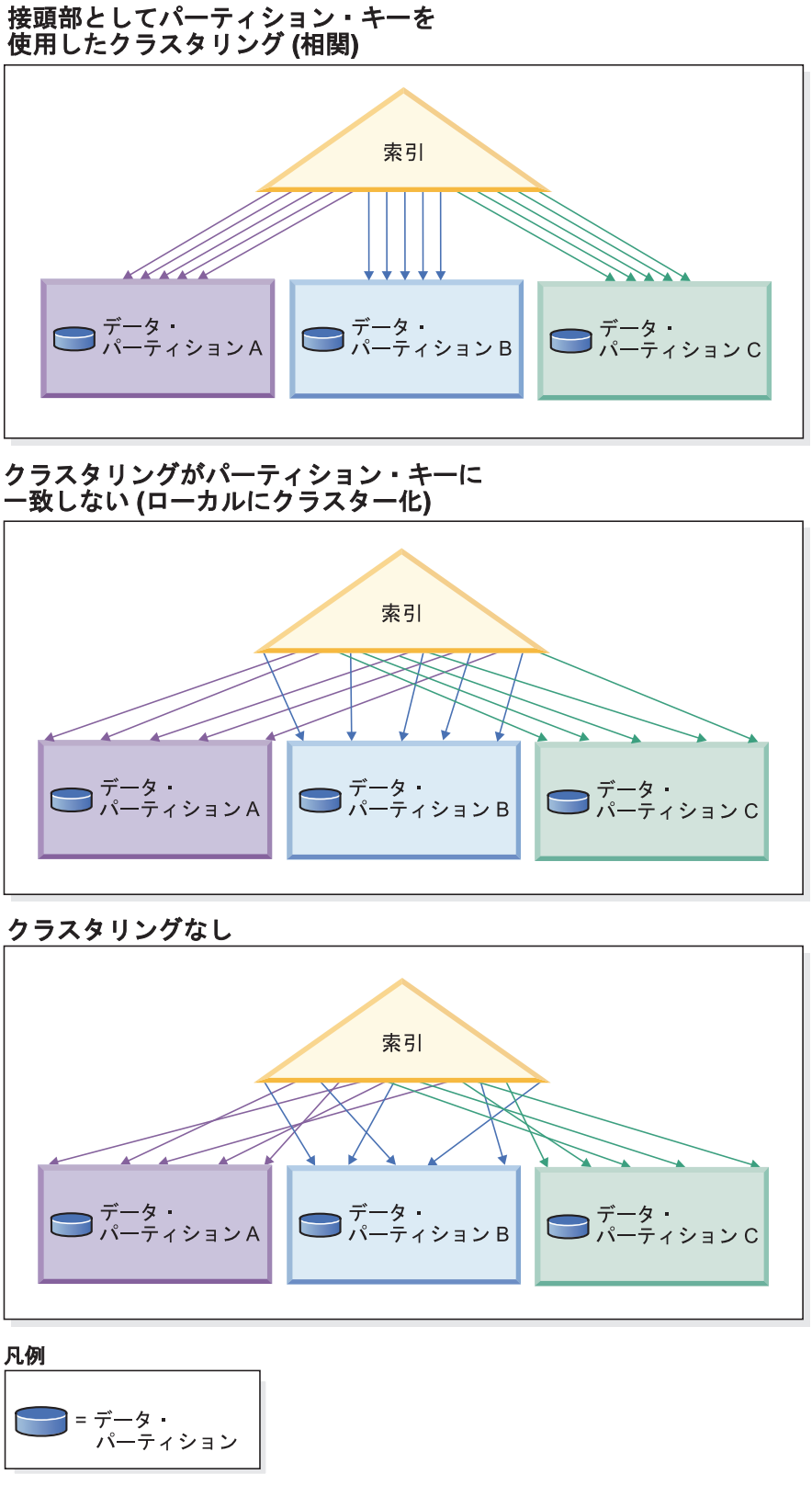


図 52. パーティション表でのクラスター化された索引の見込まれる効果。最初の図では、データはグローバルとローカルの両方でクラスター化されます。

326 ページの図 52 に示されているように、各例の索引およびデータのレイアウトが指定されていると、クラスタリングが表パーティション・キーと相関関係にある場合に最適なスキャン・パフォーマンスが得られます。クラスタリングが表パーティション・キーと相関していないと、索引がローカルにクラスター化されることはおそらくありません。表パーティション列と索引列間に相関があることが期待されているため、ローカルでの最適なクラスター化シナリオが生じることは極めてまれです。

クラスタリングの利点には、以下のものがあります。

- 各データ・パーティション内で、行はキーの順序になります。
- クラスタリング索引は、キーの順序で表内をトラバースするため、スキャンのパフォーマンスが向上します。なぜなら、スキャンは最初のページの最初の行を取り出し、その後そのページのすべての行を取り出すまで同じページの各行を取り出してから、次に移動するからです。つまり、どの時点でも表の 1 ページのみがバッファー・プール内になければならないという意味です。対照的に、表がクラスター化されていない場合、異なるページから各行が取り出される確率が高くなります。バッファー・プールに表全体を保持できる余地がある場合を除いて、各ページが何度も取り出されることになり、スキャンの速度がとても遅くなります。

  パーティション表の場合、スキャンの際に 1 度限り各ページが取り出されるという理想的な状況は、表パーティション・キーがクラスタリング・キーの前に付いている場合に限り生じ得ます (326 ページの図 52 の最初の図を参照してください)。しかし、前述のようにクラスタリング・キーが表パーティション・キーと相関しておらず、データがローカルにクラスター化される場合、バッファー・プールに各データ・パーティションの 1 ページを保持できるスペースが十分にあれば、クラスター化された索引の利点を十分に生かすことが依然として可能です。指定のデータ・パーティションの取り出されるそれぞれの行が、同じデータ・パーティションの以前に取り出された行の近くにあるからです (326 ページの図 52 の 2 番目の図を参照してください)。既に言及したように、クラスタリング・キーが表パーティション・キーと相関していない場合にはクラスタリングを十分に維持することができませんが、ご使用の表で高水準の挿入、更新、および削除活動を期待しない場合にはこの方法が役に立ちます。

  バッファー・プールにすべてのデータ・パーティションのページを保持するための十分なスペースがない場合でも、クラスター化索引を定義すると、幾らかの利点を引き続き活用できます。

第 22 章 設計アドバイザー

設計アドバイザーを使用した、単一パーティション・データベースから複数パーティション・データベースへのマイグレーション

単一パーティション・データベースから複数パーティション・データベースへのマイグレーションの際に、設計アドバイザーを役立てることができます。

このタスクについて

設計アドバイザーは、データの分散についての勧告だけでなく、新規の索引、マテリアライズ照会表 (MQT)、およびマルチディメンション・クラスタリング (MDC) 表についての勧告も提供します。

手順

1. db2licm コマンドの使用による DB2 製品またはフィーチャー・ライセンス・キーの登録。
2. 複数パーティション・データベースのパーティション・グループに少なくとも 1 つの表スペースを作成します。

   注: 設計アドバイザーは既存の表スペースへのデータの再配分のみを推奨する可能性があるため、設計アドバイザーを実行する前に、設計アドバイザーの考慮対象にする表スペースがパーティション化されたデータベース内にあるようにしてください。
3. データベース・パーティション化機能を設計アドバイザー GUI で選択するか、または db2advis コマンドにパーティション化オプションを指定して、設計アドバイザーを実行します。
4. コントロール・センターで設計アドバイザーを使用している場合は、データベース・パーティション化の推奨値を自動的にインプリメントできます。db2advis コマンドを使用している場合は、 db2advis 出力ファイルを若干変更してから、設計アドバイザーによって生成された DDL ステートメントを実行します。

第 23 章 並行性の管理

MDC 表の表および RID 索引スキャンのロック・モード

マルチディメンション・クラスタリング (MDC) 環境では、さらにロック・レベル BLOCK が使用されます。以下の表には、種々のアクセス・プランの各レベルで取得されるロックのタイプをリストしています。各項目は、表ロック、ブロック・ロック、行ロックの 3 つの部分から成ります。ダッシュは、特定のレベルのロッキングは使用されないことを示します。

注: ロック・モードは、SELECT ステートメントのロック要求節を使用して明示的に変更できます。

表 15. 述部なしの表スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	S/-/-	U/-/-	SIX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/U	IX/X/-	IX/I/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

表 16. ディメンション列上の述部のみでの表スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/X/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/U/-	X/X/-

表 17. 索引と他の述部 (sargs、resids) での 表索引スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

以下の 2 つの表には、 MDC 表上の RID 索引のロック・モードが表されています。

表 18. 述部なしの RID 索引スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	S/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

表 19. 単一修飾行での RID 索引スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/IS/S	IX/IX/U	IX/IX/X	X/X/X	X/X/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

表 20. 開始述部と停止述部のみでの RID 索引スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/IS/S	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

表 21. 索引述部のみでの RID 索引スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 22. 他の述部 (sargs、resids) での RID 索引スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 22. 他の述部 (sargs、resids) での RID 索引スキャンのロック・モード (続き)

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

注: 据え置きデータ・ページ・アクセスに使用される RID 索引スキャンのロック・モードを示す以下の表において、 IN ロックを含む UR 分離レベルでは、タイプ 1 索引の場合や索引の組み込み列上に述部が存在している場合、分離レベルは CS にアップグレードされ、ロックは IS 表ロック、IS ブロック・ロック、および NS 行ロックにアップグレードされます。

表 23. 据え置きデータ・ページ・アクセスに使用される RID 索引スキャンのロック・モード: 述部なしでの RID 索引スキャン

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/S/S	IX/IX/S		X/-/-	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

表 24. 据え置きデータ・ページ・アクセスに使用される RID 索引スキャンのロック・モード: 据え置きデータ・ページ・アクセス (述部なしでの RID 索引スキャン後)

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IN/IN/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

表 25. 据え置きデータ・ページ・アクセスに使用される RID 索引スキャンのロック・モード: 述部 (sargs、resids) での RID 索引スキャン

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/S/-	IX/IX/S		IX/IX/S	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

表 26. 据え置きデータ・ページ・アクセスに使用される RID 索引スキャンのロック・モード: 据え置きデータ・ページ・アクセス (述部 (sargs、resids) での RID 索引スキャン後)

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 27. 据え置きデータ・ページ・アクセスに使用される RID 索引スキャンのロック・モード: 開始述部と停止述部のみの RID 索引スキャン

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/IS/S	IX/IX/S		IX/IX/X	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

表 28. 据え置きデータ・ページ・アクセスに使用される RID 索引スキャンのロック・モード: 据え置きデータ・ページ・アクセス (開始述部と停止述部のみの RID 索引スキャン後)

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IS/-/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

MDC 表のブロック索引スキャンのロック

以下の表には、種々のアクセス・プランの各レベルで取得されるロックのタイプをリストしています。各項目は、表ロック、ブロック・ロック、行ロックの 3 つの部分から成ります。 ダッシュは、特定のレベルのロッキングは行われないことを示します。

注: ロック・モードは、SELECT ステートメントのロック要求節を使用して明示的に変更できます。

表 29. 述部なしでの索引スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	S/--/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--

表 29. 述部なしでの索引スキャンのロック・モード (続き)

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/--	X/X/--

表 30. ディメンション述部のみでの索引スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

表 31. 開始述部と停止述部のみでの索引スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/S/-	IX/IX/S	IX/IX/S	IX/IX/S	IX/IX/S
RS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
CS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-

表 32. 述部での索引スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/S/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

以下の表には、据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モードをリストしています。

表 33. 据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モード: 述部なしでのブロック索引スキャン

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/S/--	IX/IX/S		X/--/--	

表 33. 据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モード: 述部なしでのブロック索引スキャン (続き)

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

表 34. 据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モード: 据え置きデータ・ページ・アクセス (述部なしでのブロック索引スキャン後)

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IN/IN/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	X/X/--	X/X/--

表 35. 据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モード: ディメンション述部のみでのブロック索引スキャン

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/S/--	IX/IX/--		IX/S/--	
RS	IS/IS/NS	IX/--/--		IX/--/--	
CS	IS/IS/NS	IX/--/--		IX/--/--	
UR	IN/IN/--	IX/--/--		IX/--/--	

表 36. 据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モード: 据え置きデータ・ページ・アクセス (ディメンション述部のみでのブロック索引スキャン後)

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/S/--	IX/X/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--

表 37. 据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モード: 開始述部と停止述部のみのブロック索引スキャン

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/S/--	IX/IX/--		IX/X/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

表 38. 据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モード: 据え置きデータ・ページ・アクセス (開始述部と停止述部のみのブロック索引スキャン後)

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IN/IN/--	IX/IX/X		IX/X/--	
RS	IS/IS/NS	IN/IN/--		IN/IN/--	
CS	IS/IS/NS	IN/IN/--		IN/IN/--	
UR	IS/--/--	IN/IN/--		IN/IN/--	

表 39. 据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モード: 他の述部 (sargs、resids) でのブロック索引スキャン

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IS/S/--	IX/IX/--		IX/IX/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

表 40. 据え置きデータ・ページ・アクセスに使用されるブロック索引スキャンのロック・モード: 据え置きデータ・ページ・アクセス (他の述部 (sargs、resids) でのブロック索引スキャン後)

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索型更新または削除	
		スキャン	現在の場所	スキャンまたは削除	更新
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

パーティション表での動作をロックする

表全体のロックに加えて、パーティション表の各データ・パーティションごとのロックがあります。これにより、非パーティション表と比べて、より良い細分性および並行性の増大が可能になります。新規のデータ・パーティション・ロックは、db2pd コマンド、イベント・モニター、管理ビュー、および表関数の出力によって識別されます。

表にアクセスするとき、ロック動作は表ロックを最初に取得し、次にアクセスされたデータの命令に従ってデータ・パーティション・ロックを取得します。アクセス方式および分離レベルは、結果セットに含まれていないデータ・パーティションのロックを必要とする場合があります。これらのデータ・パーティション・ロックが取得されると、表ロックと同じだけ長く保持されます。たとえば、カーソル固定 (CS) の索引のスキャンは、以前にアクセスされたデータ・パーティションでロックを保持し、後続のキーでそのデータ・パーティションが参照される場合に、データ・パーティション・ロックを再取得するコストを削減する可能性があります。さらにデータ・パーティション・ロックは、表スペースへのアクセスを確保するコストを担います。非パーティション表の場合、表スペースのアクセスは表ロックによって処理されます。したがって、パーティション表の表レベルに排他ロックまたは共有ロックがある場合でも、データ・パーティション・ロックは発生します。

より良い細分性によって、1 つのトランザクションは、他のトランザクションが他のデータ・パーティションにアクセスしている間に、指定されたデータ・パーティションへ排他的にアクセスでき、行ロックを避けることができます。これは、大量更新用に選択されるプランの結果として、またはデータ・パーティション・レベルへのロックのエスカレーションによって生じることがあります。データ・パーティションが共有または排他的にロックされている場合であっても、多数のアクセス方式の表ロックは、通常意図的ロックです。これにより、並行性が増大します。しかし、非意図的ロックがデータ・パーティション・レベルで必要とされ、すべてのデータ・パーティションがアクセスされる可能性があることをプランが示す場合には、並行トランザクションからデータ・パーティション・ロック間のデッドロックが発生しないようにするために、表レベルで非意図が選択されることがあります。

SQL LOCK TABLE ステートメントのロック

パーティション表の場合、LOCK TABLE ステートメントに対して取得される唯一のロックは表レベルであり、データ・パーティション・ロックは取得されません。これは、行、ブロック、またはデータ・パーティション・レベルでデッドロックを避けるだけでなく、後続の DML ステートメントで表に対して行ロックが起きないようにします。 LOCK TABLE IN EXCLUSIVE MODE の使用は、索引を更新するときに排他的アクセスを保証するのに使用されますが、大きな更新の間にタイプ 2 索引の増大を制限するのに役立ちます。

ALTER TABLE ステートメントの LOCKSIZE TABLE パラメーターの影響

ALTER TABLE ステートメントには、LOCKSIZE TABLE の設定についてのオプションがありますが、これは表が意図的ロックなしで共有または排他的にロックされるようにします。パーティション表の場合、このロック計画は、表ロック、および

アクセスされるあらゆるデータ・パーティションのデータ・パーティション・ロックの両方に適用されます。

行ロックおよびブロック・ロックのエスカレーション

パーティション表の場合、行ロックおよびブロック・ロックのエスカレーションの単位は、データ・パーティション・レベルになります。これは、データ・パーティションが SHARE、EXCLUSIVE、SUPER EXCLUSIVE にエスカレートされる場合にも、エスカレートされない他のデータ・パーティションは影響を受けず、表が他のトランザクションによりさらにアクセスできることを意味します。トランザクションは、指定のデータ・パーティションのエスカレーション後に、他のデータ・パーティションの行ロックを継続できます。エスカレーションの通知ログ・メッセージには、パーティション表の名前に加えて、エスカレートされたデータ・パーティションが含まれます。したがって、ロック・エスカレーションによって索引への排他的アクセスを確保することはできません。ステートメントが排他表レベル・ロックを使用するか、明示的 LOCK TABLE IN EXCLUSIVE MODE ステートメントが発行されるか、または表が LOCKSIZE TABLE 属性を使用するかのいずれかが必要です。アクセス方式の全体の表ロックはオプティマイザーによって選択され、データ・パーティションの除去に依存します。データ・パーティションの除去が発生しない場合、表への大きな更新により、排他表ロックが選択される可能性があります。

ロック情報の解釈

以下の SNAPLOCK 管理ビューの例は、パーティション表から戻されたロック情報を解釈する際の参考にしてください。

例 1:

この SNAPLOCK 管理ビューは、オフラインでの索引再編成中にキャプチャーされたものです。

```
SELECT SUBSTR(TABNAME, 1, 15) TABNAME, TAB_FILE_ID, SUBSTR(TBSP_NAME, 1, 15) TBSP_NAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE,
LOCK_MODE, LOCK_ESCALATION FROM SYSIBMADM.SNAPLOCK where TABNAME like 'TP1' and LOCK_OBJECT_TYPE like 'TABLE_%'
ORDER BY TABNAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE, TAB_FILE_ID, LOCK_MODE

TABNAME         TAB_FILE_ID          TBSP_NAME       DATA_PARTITION_ID LOCK_OBJECT_TYPE   LOCK_MODE  LOCK_ESCALATION
--------------- -------------------- --------------- ----------------- ------------------ ---------- ---------------
TP1                           32768 -                              -1 TABLE_LOCK         Z                       0
TP1                               4 USERSPACE1                     0 TABLE_PART_LOCK    Z                       0
TP1                               5 USERSPACE1                     1 TABLE_PART_LOCK    Z                       0
TP1                               6 USERSPACE1                     2 TABLE_PART_LOCK    Z                       0
TP1                               7 USERSPACE1                     3 TABLE_PART_LOCK    Z                       0
TP1                               8 USERSPACE1                     4 TABLE_PART_LOCK    Z                       0
TP1                               9 USERSPACE1                     5 TABLE_PART_LOCK    Z                       0
TP1                              10 USERSPACE1                     6 TABLE_PART_LOCK    Z                       0
TP1                              11 USERSPACE1                     7 TABLE_PART_LOCK    Z                       0
TP1                              12 USERSPACE1                     8 TABLE_PART_LOCK    Z                       0
TP1                              13 USERSPACE1                     9 TABLE_PART_LOCK    Z                       0
TP1                              14 USERSPACE1                    10 TABLE_PART_LOCK    Z                       0
TP1                              15 USERSPACE1                    11 TABLE_PART_LOCK    Z                       0
TP1                               4 USERSPACE1                     - TABLE_LOCK         Z                       0
TP1                               5 USERSPACE1                     - TABLE_LOCK         Z                       0
TP1                               6 USERSPACE1                     - TABLE_LOCK         Z                       0
TP1                               7 USERSPACE1                     - TABLE_LOCK         Z                       0
TP1                               8 USERSPACE1                     - TABLE_LOCK         Z                       0
TP1                               9 USERSPACE1                     - TABLE_LOCK         Z                       0
TP1                              10 USERSPACE1                     - TABLE_LOCK         Z                       0
TP1                              11 USERSPACE1                     - TABLE_LOCK         Z                       0
TP1                              12 USERSPACE1                     - TABLE_LOCK         Z                       0
TP1                              13 USERSPACE1                     - TABLE_LOCK         Z                       0
```

```
TP1                             14 USERSPACE1                   - TABLE_LOCK        Z                   0
TP1                             15 USERSPACE1                   - TABLE_LOCK        Z                   0
TP1                             16 USERSPACE1                   - TABLE_LOCK        Z                   0

  26 record(s) selected.
```

この例では、パーティション表 TP1 に対するアクセスと並行性の制御に、タイプ TABLE_LOCK のロック・オブジェクトと DATA_PARTITION_ID -1 を使用しています。 タイプ TABLE_PART_LOCK のロック・オブジェクトは、各データ・パーティションのほとんどのアクセスおよび並行性の制御に使用されます。

この出力では、他にもタイプ TABLE_LOCK のロック・オブジェクトがありますが (TAB_FILE_ID 4 から 16)、これらのロック・オブジェクトには DATA_PARTITION_ID の値がありません。この種のロック (オブジェクトがデータ・パーティションまたはパーティション表の索引に対応する TAB_FILE_ID と TBSP_NAME を持つ) は、オンライン・バックアップ・ユーティリティーとの並行性を制御するのに使用される場合があります。

第 24 章 エージェント管理

パーティション・データベースにおけるエージェント

パーティション・データベース環境、およびパーティション内並列処理が使用可能になっている環境の場合には、各データベース・パーティション (つまり、各データベース・サーバーまたはノード) が独自のエージェント・プールを持っていて、そこからサブエージェントを引き出すことができます。このプールがあるので、必要になったり作業を終了したりするたびに、サブエージェントを作成したり破棄したりする必要がありません。サブエージェントはプール内に関連エージェントとして残されるので、それらが関連付けされたアプリケーションから、または新規のアプリケーションから新しい要求が出された場合には、データベース・マネージャーによってそれらのサブエージェントが使用されます。

パーティション・データベース環境およびパーティション内並列処理が使用可能になっている環境の場合、システム内のパフォーマンスとメモリー・コストへの影響は、以下に示すエージェント・プールのチューニング方法に強く関係しています。

- エージェント・プール・サイズに関するデータベース・マネージャー構成パラメーター (*num_poolagents*) は、1 つのデータベース・パーティション (ノードとも呼ばれる) でアプリケーションとの関連付けを保持できるエージェントとサブエージェントの両方の数に影響します。プール・サイズが小さすぎるので、プールが満杯になった場合には、サブエージェントは作業を行っているアプリケーションと自分自身との関連付けを切り離し、終了します。サブエージェントを作成してアプリケーションと再度関連付けをするということを常に行わなければならないため、パフォーマンスが低下します。

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

  さらに、*num_poolagents* の値が小さすぎた場合には、ある 1 つのアプリケーションが関連サブエージェントによってプールを満杯にしてしまう場合があります。そして、他のアプリケーションが新しいサブエージェントを要求したときに、関連エージェント・プール内にサブエージェントがないとすると、そのアプリケーションは、他のアプリケーションのエージェント・プールからアクティブでないサブエージェントをリサイクルします。この動作により、リソースは完全に使用されます。
- エージェントを少ししか持たないことと、任意の時点で多数のエージェントをアクティブにする場合のリソース・コストと比較してください。

  たとえば、*num_poolagents* の値が大きすぎる場合には、関連するサブエージェントは、長い間未使用のままプール内に置かれ、他のタスクでは使用可能になっていないデータベース・マネージャー・リソースが使用される可能性があります。

  注: 接続コンセントレーターが有効な場合、*num_poolagents* によって指定されたエージェント数は、同時にプール内でアイドル状態のままであるエージェントの

正確な数を必ずしも反映するわけではありません。エージェント数は、さらに多くのワークロード・アクティビティーのオカレンスを処理するために、一時的に超過する可能性があります。

その他の非同期プロセスおよび非同期スレッド

データベース・マネージャーが独自のプロセスまたはスレッドとして実行する非同期アクティビティーは、データベース・エージェント以外にもあります。たとえば、次のようなアクティビティーがあります。

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

第 25 章 アクセス・プランの最適化

索引アクセスとクラスター率

アクセス・プランを選択するとき、オプティマイザーはディスクから必要なページをバッファー・プールに取り出すのに必要な入出力の数を見積もります。この見積もりには、バッファー・プール使用率の予測も含まれています。すでにバッファー・プールの中にあるページに含まれている行を読み取るには、追加の入出力が必要ないためです。

索引スキャンの場合、オプティマイザーは、システム・カタログ表 (SYSCAT.INDEXES) の情報を使用して、データ・ページをバッファー・プール内に読み込むための入出力コストを見積もります。 SYSCAT.INDEXES 表の以下の列の情報を使用します。

- CLUSTERRATIO 情報はこの索引に関連して表データのクラスター化の程度を示します。数が大きいほど、行は索引キーの順序に並んでいます。表の行がほとんど索引キーの順序に並んでいれば、いくつもの行を 1 つのデータ・ページがバッファーにある内にそのページから読み取ることができます。この列の値が -1 の場合、オプティマイザーは、使用可能なら PAGE_FETCH_PAIRS および CLUSTERFACTOR 情報を使用します。
- PAGE_FETCH_PAIRS には CLUSTERFACTOR 情報と共に、データ・ページをさまざまなサイズのバッファー・プールに読み込むのに必要な入出力の数をモデル化するための数値の対がいくつか含まれています。これらの列のデータは、索引に対して RUNSTATS を DETAILED 節付きで実行した場合だけ集められます。

索引のクラスタリング統計が使用不可である場合、オプティマイザーはデフォルト値を使います。この値は索引に関するデータのクラスタリング率が低いことを想定しています。

索引のデータがどの程度クラスター化されているかによってパフォーマンスに重大な影響を与える可能性があるため、表の索引の 1 つを 100% クラスター化に近く維持してください。

一般的に、キーがクラスター索引のキーのスーパーセットである場合と 2 つの索引のキー列間に事実上の相関がある場合を除いて、 100% クラスタリングできるのは 1 つの索引だけです。

表を再編成するときに、行をクラスター化し、挿入処理中、この特性を保持するのに使用する索引を指定することができます。更新および挿入を行うと索引に対する表のクラスター化が低下するため、定期的に表を再編成することが必要な場合があります。 INSERT、UPDATE、および DELETE により頻繁に変更が加えられる表に対する再編成の頻度を少なくするには、ALTER TABLE で PCTFREE パラメーターを変更します。こうすると、追加の挿入データがあっても既存のデータとのクラスター化が維持されます。

MDC 表のための表および索引管理

マルチディメンション・クラスタリング (MDC) 表の表および索引の編成は、標準の表編成と同じ論理構造に基づいています。標準の表と同じく、 MDC 表はデータの行を含むページに編成されて、列に分割され、各ページの行は行 ID (RID) によって識別されます。しかしそれに加えて、MDC 表のページはエクステント・サイズのブロックにグループ化されます。たとえば、エクステント・サイズが 4 の表を示す下の例で、 0 ～ 3 の番号が付けられた最初の 4 ページは表の最初のブロックとなります。 4 ～ 7 の番号が付けられた次のページのセットは、表の 2 番目のブロックとなります。

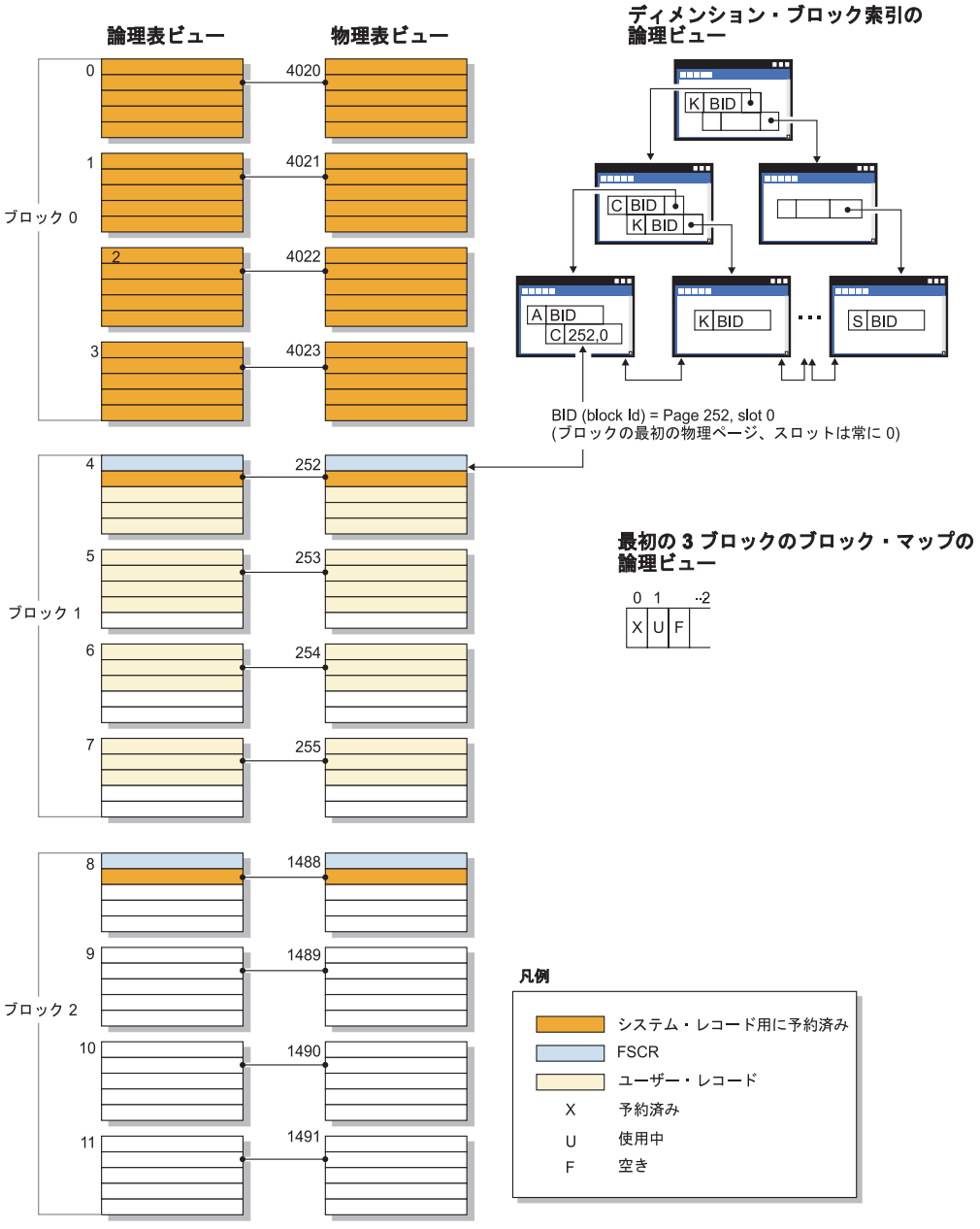


図 53. MDC 表の論理表、レコード、および索引構造

最初のブロックには、DB2 が表を管理するために使用する、フリー・スペース制御レコード (FSCR) を含む特殊な内部レコードが含まれます。続くブロックでは、最初のページに FSCR が含まれます。 FSCR はブロック内の各ページに存在する新規のレコード用にフリー・スペースをマップします。この使用可能なフリー・スペースは、表にレコードを挿入する際に使用されます。

名前が暗黙に示すように、MDC 表は複数のディメンションのデータをクラスター化します。各ディメンションは、CREATE TABLE ステートメントの ORGANIZE BY DIMENSIONS 節で指定した列または列のセットによって決まります。 MDC 表を作成するとき、以下の 2 種類の索引が自動的に作成されます。

- 単一のディメンションの各ブロックに対するポインターを含む、ディメンション・ブロック索引。
- すべてのディメンションのキー列を含む、複合ブロック索引。複合ブロック索引を使用して、挿入および更新の際のクラスタリングを保守することができます。

オプティマイザーは、特定の照会に最適のアクセス・プランを判別する際にディメンション・ブロック索引を利用するアクセス・プランを検討します。照会にディメンション値についての述部があるとき、オプティマイザーはディメンション・ブロック索引を使用して、これらの値を含むエクステントを識別し、そこからフェッチします。エクステントはディスク上の物理的に連続したページなので、これによりパフォーマンスが向上して入出力が最少になります。

さらに、データ・アクセス・プランの分析によって特定の RID 索引が照会のパフォーマンスを改善することが示された場合、その索引を作成することができます。

ディメンション・ブロック索引および複合ブロック索引に加えて、 MDC 表は各ブロックの可用性状況を示すビットマップを含むブロック・マップを保守します。以下の属性は、ビットマップ・リスト内にコード化されています。

- X (予約済み): 最初のブロックには表のシステム情報だけが含まれます。
- U (使用中): このブロックはディメンション・ブロック索引と共に使用され、それに関連付けられています。
- L (ロード済み): このブロックは現行のロード操作によってロードされました。
- C (チェック制約): このブロックはロード中の増分制約チェックを指定するためにロード操作によって設定されました。
- T (表のリフレッシュ): このブロックは AST 保守が必要であることを指定するためにロード操作によって設定されました。
- F (フリー): 他の属性が設定されていない場合、ブロックはフリーと見なされます。

各ブロックはブロック・マップ・ファイル内に項目があるので、表が拡大するとファイルも拡大します。この表は別個のオブジェクトとして保管されます。 SMS 表スペース内で、これは新規のファイル・タイプです DMS 表スペース内で、これはオブジェクト表に新規のオブジェクト記述子を持ちます。

クラスタリング

時間がたつにつれ、更新によってデータ・ページ上の行の位置が変わり、索引とデータ・ページの間に存在するクラスタリング の度合いが低下する場合があります。

選択された索引に関する表を再編成すると、データが再クラスター化されます。クラスター索引は、基本表内のデータのより良い順次アクセスを可能にするので、範囲述部のある列に最も役立ちます。このようにすると、類似値が同一のデータ・ページにあるため、ページ・フェッチは少なくなります。

一般に、高度なクラスタリングが可能なのは、1 つの表の中で 1 つの索引だけです。

索引のクラスタリングの度合いを調べるには、そのノードをダブルクリックし、「索引統計」ウィンドウを表示します。クラスター化率またはクラスター因子の値がこのウィンドウに表示されます。値が低い場合は、表のデータを再編成することをご検討ください。

パーティション内並列処理の最適化ストラテジー

SQL ステートメントのコンパイル時に並列処理の多重度が指定された場合、オプティマイザーは、シングル・データベース・パーティション内で並列して照会を実行するアクセス・プランを選択します。

実行時には、サブエージェントと呼ばれる複数のデータベース・エージェントが作成されて、照会を実行します。サブエージェントの数は、SQL ステートメントのコンパイル時に指定された並列処理の多重度以下になります。

オプティマイザーは、アクセス・プランを並列化するため、プランを各サブエージェントによって実行される部分とコーディネーター・エージェントによって実行される部分とに分割します。サブエージェントは、表キューを介して、データをコーディネーター・エージェントか他のサブエージェントに渡します。パーティション・データベース環境では、サブエージェントは、表キューを介して、他のデータベース・パーティションのサブエージェントとの間でデータの送受信を行うことができます。

パーティション内の並列スキャン方式

リレーショナル・スキャンおよび索引スキャンは、同じ表または索引上で並列して実行することができます。並列リレーショナル・スキャンの場合、表は、ページ範囲または行範囲に分割されます。分割後、ページ範囲または行範囲がサブエージェントに割り当てられます。サブエージェントは割り当てられた範囲をスキャンし、その現行の範囲での作業が完了した時点で別の範囲が割り当てられます。

並列索引スキャンの場合には、索引は、索引キー値およびキー値あたりの索引項目数に基づいて、レコード範囲に分割されます。並列索引スキャンは、並列表スキャンと同様に、レコード範囲を割り当てられたサブエージェントを使用して行われます。サブエージェントには、現行の範囲での作業が完了した時点で新しい範囲が割り当てられます。

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

並列スキャンは、サブエージェント間で均等になるように作業を分散します。並列スキャンの目標は、サブエージェント間の負荷を均衡させて、サブエージェントが同等に使用されるようにすることです。使用中のサブエージェントの数が使用可能

なプロセッサーの数と等しく、ディスクが入出力要求で過度に作動しているということがない場合には、マシン・リソースは効率的に使用されていると言えます。

他のアクセス・プラン方式によっては、照会の実行時にデータの不均衡が生じることがあります。オプティマイザーは、サブエージェント間でデータのバランスを維持できるように並列方式を選択します。

パーティション内の並列ソート方式

オプティマイザーは、以下のいずれかの並列ソート方式を選択します。

- **ラウンドロビン・ソート**

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

- **パーティション・ソート**

  このソートは、ソートが各サブエージェントごとに作成されるという点では、ラウンドロビン・ソートに似た働きをします。このソートでは、サブエージェントはハッシュ関数をソート列に適用して、行をどのソートに挿入するかを判別します。たとえば、マージ結合の内部表と外部表がパーティション・ソートの場合、サブエージェントは、マージ結合を使用することによって、対応する表の部分を結合し並列で実行できます。

- **複製ソート**

  このソートは、各サブエージェントがすべてのソート出力を必要とする場合に使用されます。あるソートが作成されると、サブエージェントは、そのソートに行が挿入されるときに同期化されます。ソートが完了すると、各サブエージェントがソート全体の読み取りを行います。このソートは、行数が少ないとき、データ・ストリームのバランスをとり直すのに使用できます。

- **共有ソート**

  このソートは、複製ソートと同様の働きをしますが、共有ソートの場合は、ソートされた結果に対してサブエージェントが並列スキャンをオープンし、ラウンドロビン・ソートと同様の方法でサブエージェント間にデータが分配されます。

パーティション内並列一時表

サブエージェントが共同して同じ表に行を挿入することによって、一時表を生成できます。この表は、**共有一時表**と呼ばれます。サブエージェントは、データ・ストリームが複製されるか分割されるかに応じて、専用スキャンまたは並列スキャンのいずれかを共有一時表上でオープンします。

パーティション内の並列集約方式

集約操作は、サブエージェントによって並列に実行することができます。集約操作では、データをグループ化列上に配列する必要があります。サブエージェントがグループ化列の値の集合に関する行をすべて確実に受け取ることができれば、集約を最後まで完全に実行できます。これは、以前のパーティション・ソートのためにグループ化列上のストリームがすでに分割されている場合に生じます。

上記以外の場合は、サブエージェントは部分的に集約を実行し、別の方式を使用して集約を完了させます。その方式は以下のとおりです。

- 表キューをマージして、部分的に集約されたデータをコーディネーター・エージェントに送る。コーディネーターにより集約が完全に行われます。
- 部分的に集約データをパーティション・ソートに挿入します。このソートは、グループ化列上で分割されるため、グループ化列の集合に関するすべての行が確実に 1 つのソート・パーティションに入れられます。
- 処理のバランスをとるためにストリームの複製が必要な場合は、部分的に集約データを複製ソートに挿入できます。個々のサブエージェントは複製ソートを使用して集約を完成させ、集約の結果と同じ内容のコピーを受け取ります。

パーティション内の並列結合方式

結合操作は、サブエージェントによって並列に実行することができます。並列結合の方式は、データ・ストリームの特性によって決められます。

結合は、結合の内部表または外部表でデータ・ストリームをパーティションに分割または複製 (あるいは、その両方) することによって、並列化できます。たとえば、ネスト・ループ結合は、外部のストリームが並列スキャンのためにパーティション化され、さらに内部のストリームが各サブエージェントで別々に再評価されると並列化できます。マージ結合は、内部ストリームと外部ストリームがパーティション・ソートのために値でパーティション化されると並列化できます。

db2expln および dynexpln 出力の例

ここに示されている例は、 db2expln および dynexpln からの出力のレイアウトと形式を理解するために役立ちます。これらの例は、別段の説明がない限り、DB2 で提供される SAMPLE データベースに対して実行されたものです。それぞれの例について、簡単な説明が添えられています。 1 つの例と次の例との重要な相違点は、**太字**で示してあります。

例 1: 非並列

この例は、全従業員の名前、職種、部門名とその場所、および現在携わっているプロジェクト名のリストを要求するだけのものです。このアクセス・プランの特徴は、指定したそれぞれの表から関係するデータを結合するのにハッシュ結合を使用するという点です。索引を使用することができないので、アクセス・プランは各表が結合される際にリレーション・スキャンを行います。

```
******************** PACKAGE ****************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:05:00
```

```
 Bind Timestamp = 2002-01-04-14.05.00.415403

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION --------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 120.518692
Estimated Cardinality = 221.535980

(    6) Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  #Columns = 3
        |  Relation Scan
        |  |  Prefetch: Eligible
        |  Lock Intents
        |  |  Table: Intent Share
        |  |  Row  : Next Key Share
(    6) |  Process Build Table for Hash Join
(    2) Hash Join
        |  Estimated Build Size: 7111
        |  Estimated Probe Size: 9457
(    5) |  Access Table Name = DOOLE.PROJECT  ID = 2,7
        |  |  #Columns = 2
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    5) |  |  Process Build Table for Hash Join
(    3) |  Hash Join
        |  |  Estimated Build Size: 5737
        |  |  Estimated Probe Size: 6421
(    4) |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
        |  |  |  #Columns = 3
        |  |  |  Relation Scan
        |  |  |  |  Prefetch: Eligible
        |  |  |  Lock Intents
        |  |  |  |  Table: Intent Share
        |  |  |  |  Row  : Next Key Share
(    4) |  |  |  Process Probe Table for Hash Join
(    1) Return Data to Application
        |  #Columns = 5

End of section


Optimizer Plan:

               RETURN
               (   1)
                 |
```

```
                HSJOIN
                (   2)
               /      \
         HSJOIN        TBSCAN
         (   3)        (   6)
        /      \         |
    TBSCAN    TBSCAN   Table:
    (   4)    (   5)   DOOLE
      |         |      EMPLOYEE
  Table:      Table:
  DOOLE       DOOLE
  DEPARTMENT  PROJECT
```

アクセス・プランの最初の部分では、DEPARTMENT および PROJECT 表にアクセスし、ハッシュ結合を使ってそれらの表を結合します。 この結合の結果は EMPLOYEE 表に結合されます。結果行はアプリケーションに戻されます。

例 2: パーティション内並列処理による単一パーティションのプラン

この例は、最初の例と同じ SQL ステートメントを示していますが、この照会は、4-way の SMP マシン用にコンパイルされたものです。

```
******************** PACKAGE ****************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:12:38

 Bind Timestamp = 2002-01-04-14.12.38.732627

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = Yes (Bind Degree = 4)

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ----------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 133.934692
Estimated Cardinality = 221.535980

(    2) Process Using 4 Subagents
(    7) |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  |  #Columns = 3
        |  |  Parallel Scan
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
```

```
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    7) |  |  Process Build Table for Hash Join
(    3) |  Hash Join
        |  |  Estimated Build Size: 7111
        |  |  Estimated Probe Size: 9457
(    6) |  |  Access Table Name = DOOLE.PROJECT  ID = 2,7
        |  |  |  #Columns = 2
        |  |  |  Parallel Scan
        |  |  |  Relation Scan
        |  |  |  |  Prefetch: Eligible
        |  |  |  Lock Intents
        |  |  |  |  Table: Intent Share
        |  |  |  |  Row  : Next Key Share
(    6) |  |  |  Process Build Table for Hash Join
(    4) |  |  Hash Join
        |  |  |  Estimated Build Size: 5737
        |  |  |  Estimated Probe Size: 6421
(    5) |  |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
        |  |  |  |  #Columns = 3
        |  |  |  |  Parallel Scan
        |  |  |  |  Relation Scan
        |  |  |  |  |  Prefetch: Eligible
        |  |  |  |  Lock Intents
        |  |  |  |  |  Table: Intent Share
        |  |  |  |  |  Row  : Next Key Share
(    5) |  |  |  |  Process Probe Table for Hash Join
(    2) |  Insert Into Asynchronous Local Table Queue  ID = q1
(    2) Access Local Table Queue  ID = q1  #Columns = 5
(    1) Return Data to Application
        |  #Columns = 5

End of section


Optimizer Plan:

              RETURN
              (   1)
                |
               LTQ
              (   2)
                |
              HSJOIN
              (   3)
             /      \
       HSJOIN         TBSCAN
       (   4)         (   7)
      /      \          |
   TBSCAN    TBSCAN   Table:
   (   5)    (   6)   DOOLE
     |         |      EMPLOYEE
 Table:      Table:
 DOOLE       DOOLE
 DEPARTMENT  PROJECT
```

このプランは、最初の例のプランとほとんど同じです。主な相違は、プランが最初に開始されるときに 4 つのサブエージェントを作成すること、および、アプリケーションに戻す前におのおののサブエージェントの作業の結果を収集するために、プランの終了時に表キューを作成することです。

例 3: パーティション間並列処理による複数パーティションのプラン

この例は、最初の例と同じ SQL ステートメントを示していますが、この照会は、3 つのデータベース・パーティションからなるパーティション・データベースでコンパイルされたものです。

```
******************** PACKAGE ************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:54:57

 Bind Timestamp = 2002-01-04-14.54.57.033666

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = Yes
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION -----------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 118.483406
Estimated Cardinality = 474.720032

        Coordinator Subsection:
(-----)    Distribute Subsection #2
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #3
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #1
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(    2)    Access Table Queue  ID = q1  #Columns = 5
(    1)    Return Data to Application
           |  #Columns = 5

        Subsection #1:
(    8)    Access Table Queue  ID = q2  #Columns = 2
(    3)    Hash Join
           |  Estimated Build Size: 5737
           |  Estimated Probe Size: 8015
(    6)    |  Access Table Queue  ID = q3  #Columns = 3
(    4)    |  Hash Join
           |  |  Estimated Build Size: 5333
           |  |  Estimated Probe Size: 6421
(    5)    |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
           |  |  |  #Columns = 3
           |  |  |  Relation Scan
```

```
            |  |  |  |  Prefetch: Eligible
            |  |  |  Lock Intents
            |  |  |  |  Table: Intent Share
            |  |  |  |  Row  : Next Key Share
(    5)     |  |  |  Process Probe Table for Hash Join
(    2)     Insert Into Asynchronous Table Queue  ID = q1
            |  Broadcast to Coordinator Node
            |  Rows Can Overflow to Temporary Table

         Subsection #2:
(    9)     Access Table Name = DOOLE.PROJECT  ID = 2,7
            |  #Columns = 2
            |  Relation Scan
            |  |  Prefetch: Eligible
            |  Lock Intents
            |  |  Table: Intent Share
            |  |  Row  : Next Key Share
(    9)     |  Insert Into Asynchronous Table Queue  ID = q2
            |  |  Hash to Specific Node
            |  |  Rows Can Overflow to Temporary Tables
(    8)     Insert Into Asynchronous Table Queue Completion  ID = q2

         Subsection #3:
(    7)     Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
            |  #Columns = 3
            |  Relation Scan
            |  |  Prefetch: Eligible
            |  Lock Intents
            |  |  Table: Intent Share
            |  |  Row  : Next Key Share
(    7)     |  Insert Into Asynchronous Table Queue  ID = q3
            |  |  Hash to Specific Node
            |  |  Rows Can Overflow to Temporary Tables
(    6)     Insert Into Asynchronous Table Queue Completion  ID = q3

End of section


Optimizer Plan:

                RETURN
                (   1)
                  |
                 BTQ
                (   2)
                  |
                HSJOIN
                (   3)
               /      \
          HSJOIN          DTQ
          (   4)        (   8)
         /      \         |
    TBSCAN      DTQ     TBSCAN
    (   5)     (   6)   (   9)
      |          |        |
 Table:       TBSCAN   Table:
 DOOLE        (   7)   DOOLE
 DEPARTMENT     |      PROJECT
             Table:
             DOOLE
             EMPLOYEE
```

このプランは、最初の例のプランと全く同じ内容ですが、セクションが 4 つのサブセクションに分けられています。サブセクションは、次のようなタスクを行います。

- **コーディネーター・サブセクション**。このサブセクションは、他のサブセクションを調整するものです。このプランでは、他のサブセクションを分散させ、アプリケーションに戻される結果を集めるために表キューを使用します。
- **サブセクション #1**。このサブセクションは表キュー q2 をスキャンし、ハッシュ結合を使って表キュー q3 からのデータと結合します。 2 番目のハッシュ結合は、DEPARTMENT 表のデータへの追加を行います。結合された行は次に、表キュー q1 を使ってコーディネーター・サブセクションに送られます。
- **サブセクション #2**。このサブセクションは、PROJECT 表をスキャンして結果を使用して特定のノードへハッシュします。これらの結果は、サブセクション #1 が読み取ります。
- **サブセクション #3**。このサブセクションは、EMPLOYEE 表をスキャンして結果を使用して特定のノードへハッシュします。これらの結果は、サブセクション #1 が読み取ります。

例 4: パーティション間並列処理とパーティション内並列処理による複数パーティションのプラン

この例は、最初の例と同じ SQL ステートメントを示していますが、この照会は、3 つのデータベース・パーティション (4-way SMP マシン上にある) から成るパーティション・データベースでコンパイルされたものです。

```
******************** PACKAGE ****************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:58:35

 Bind Timestamp = 2002-01-04-14.58.35.169555

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = Yes
 Intra-Partition Parallel = Yes (Bind Degree = 4)

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 145.198898
Estimated Cardinality = 474.720032

        Coordinator Subsection:
(-----)    Distribute Subsection #2
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
```

```
(-----)    Distribute Subsection #3
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #1
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(    2)    Access Table Queue  ID = q1  #Columns = 5
(    1)    Return Data to Application
           |  #Columns = 5

        Subsection #1:
(    3)    Process Using 4 Subagents
(   10)    |  Access Table Queue  ID = q3  #Columns = 2
(    4)    |  Hash Join
           |  |  Estimated Build Size: 5737
           |  |  Estimated Probe Size: 8015
(    7)    |  |  Access Table Queue  ID = q5  #Columns = 3
(    5)    |  |  Hash Join
           |  |  |  Estimated Build Size: 5333
           |  |  |  Estimated Probe Size: 6421
(    6)    |  |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
           |  |  |  |  #Columns = 3
           |  |  |  |  Parallel Scan
           |  |  |  |  Relation Scan
           |  |  |  |  |  Prefetch: Eligible
           |  |  |  |  Lock Intents
           |  |  |  |  |  Table: Intent Share
           |  |  |  |  |  Row  : Next Key Share
(    6)    |  |  |  |  Process Probe Table for Hash Join
(    3)    |  Insert Into Asynchronous Local Table Queue  ID = q2
(    3)    Access Local Table Queue  ID = q2  #Columns = 5
(    2)    Insert Into Asynchronous Table Queue  ID = q1
           |  Broadcast to Coordinator Node
           |  Rows Can Overflow to Temporary Table

        Subsection #2:
(   11)    Process Using 4 Subagents
(   12)    |  Access Table Name = DOOLE.PROJECT  ID = 2,7
           |  |  #Columns = 2
           |  |  Parallel Scan
           |  |  Relation Scan
           |  |  |  Prefetch: Eligible
           |  |  Lock Intents
           |  |  |  Table: Intent Share
           |  |  |  Row  : Next Key Share
(   11)    |  Insert Into Asynchronous Local Table Queue  ID = q4
(   11)    Access Local Table Queue  ID = q4  #Columns = 2
(   10)    Insert Into Asynchronous Table Queue  ID = q3
           |  Hash to Specific Node
           |  Rows Can Overflow to Temporary Tables

        Subsection #3:
(    8)    Process Using 4 Subagents
(    9)    |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
           |  |  #Columns = 3
           |  |  Parallel Scan
           |  |  Relation Scan
           |  |  |  Prefetch: Eligible
           |  |  Lock Intents
           |  |  |  Table: Intent Share
           |  |  |  Row  : Next Key Share
(    8)    |  Insert Into Asynchronous Local Table Queue  ID = q6
(    8)    Access Local Table Queue  ID = q6  #Columns = 3
(    7)    Insert Into Asynchronous Table Queue  ID = q5
           |  Hash to Specific Node
           |  Rows Can Overflow to Temporary Tables
```

```
End of section


Optimizer Plan:

               RETURN
               (   1)
                 |
                BTQ
               (   2)
                 |
                LTQ
               (   3)
                 |
               HSJOIN
               (   4)
              /      \
         HSJOIN       DTQ
         (   5)      (  10)
        /      \       |
    TBSCAN     DTQ     LTQ
    (   6)    (   7)  (  11)
      |         |       |
 Table:        LTQ     TBSCAN
 DOOLE        (   8)  (  12)
 DEPARTMENT     |       |
              TBSCAN   Table:
              (   9)   DOOLE
                 |     PROJECT
            Table:
            DOOLE
            EMPLOYEE
```

このプランは、3 番目の例にあるプランと似ていますが、複数のサブエージェントが各サブセクションを実行する点が異なります。また、各サブセクションの最後に、ローカル表キューが、すべてのサブエージェントの結果を収集してから、修飾行が 2 番目の表キューに挿入され、特定のノードでハッシュされる点も異なります。

結合

*結合*とは、情報の何らかの共通の領域に基づいて複数の表からの情報を組み合わせるプロセスのことです。 1 つの表の行は別の表の行と、対応する行が結合基準に合致する場合に、組にされます。

たとえば、次の 2 つの表を考えてみてください。

Table1

PROJ	PROJ_ID
A	1
B	2
C	3
D	4

Table2

PROJ_ID	NAME
1	Sam
3	Joe
4	Mary
1	Sue
2	Mike

表の ID 列が同じ値である場合に Table1 と Table2 を結合するには、次に示した SQL ステートメントを使用します。

```
SELECT PROJ, x.PROJ_ID, NAME
  FROM TABLE1 x, TABLE2 y
  WHERE x.PROJ_ID = y.PROJ_ID
```

この照会で次の結果行のセットが生成されます。

PROJ	PROJ_ID	NAME
A	1	Sam
A	1	Sue
B	2	Mike
C	3	Joe
D	4	Mary

結合述部が存在するかどうか、また表と索引の統計によって判別される各種コストに応じて、オプティマイザーは以下の結合方式のどれかを選択します。

- ネスト・ループ結合
- マージ結合
- ハッシュ結合

2 つの表を結合する場合、1 つの表は外部表として選択され、もう一方の表は内部表として選択されます。外部表は最初にアクセスされ、1 回だけスキャンされます。内部表が複数回スキャンされるかどうかは、結合の種類および存在する索引によります。照会によって 3 つ以上の表が結合されるとしても、オプティマイザーは 1 回に 2 つの表だけを結合します。必要なら中間結果を保持するために一時表が作成されます。

INNER または LEFT OUTER JOIN のような明示的な結合演算子を指定して、結合で表をどう使用するかを決定することができます。しかしながら、この方法で照会を変更する前に、オプティマイザーにどう表を結合するか決定させてみるべきです。それから、照会のパフォーマンスを分析して結合演算子を追加するかどうか決めてください。

照会の最適化に影響を与えるデータベース・パーティション・グループ

パーティション・データベース環境では、オプティマイザーは表のコロケーションを認識し、照会に対する最適のアクセス・プランを判別する際にそのコロケーションを使用します。表が頻繁に結合照会に関係する場合、それらの表は、結合される各表にある行が同じデータベース・パーティションにあるように、パーティション・データベース環境のデータベース・パーティション間で分割する必要があります。結合操作を実行するときに、結合される両方の表にあるデータのコロケーションによって、データをあるデータベース・パーティションから別のデータベース・パーティションに移動されなくなります。同じデータベース・パーティション・グループに両方の表を置き、表のデータが確実に同じパーティションに入れられるようにしてください。

パーティション・データベース環境では、表のサイズによって、データをより多くのデータベース・パーティションに配分すると、照会の実行にかかる見積時間 (つ

まりコスト) が減少します。表の数、表のサイズ、それらの表のデータがある場所、および結合が必要かどうかといった照会のタイプはすべて照会のコストに影響を与えます。

パーティション・データベースでの結合ストラテジー

いくつかの面でパーティション・データベース環境の結合のための戦略はパーティションのないデータベース環境と違います。パフォーマンスを改善するために、標準の結合方式に加えて別の技法を適用することができます。

パーティション・データベース環境で頻繁な結合が行われる表についての考慮事項の一つに、表コロケーションがあります。表コロケーションは、パーティション・データベース環境において、ある表のデータを、同じ分散キーに基づいて、同じデータベース・パーティションにある別の表のデータを使用して見つけ出すための手段を提供します。コロケーションが行われると、照会の中で結合されるデータは、照会作業の一部として別のデータベース・パーティションに移動せずに処理されます。結合の応答セットのみがコーディネーター・ノードに移動されます。

表キュー

パーティション・データベース環境での結合技法の説明は以下の用語を使用します。

- 表キュー (*TQ* と呼ばれることもある)

  データベース・パーティション間 (または、単一パーティション・データベースの場合はプロセッサー間) で行を転送するための機構。
- 指示表キュー (*DTQ* と呼ばれることもある)

  行が受信データベース・パーティションの 1 つにハッシュされる表キュー。
- ブロードキャスト表キュー (*BTQ* と呼ばれることもある)

  行がすべての受信データベース・パーティションに送信されるが、ハッシュは行われない表キュー。

表キューは、以下の環境で使用されます。

- パーティション間並列処理の使用時に、あるデータベース・パーティションの表データを別のデータベース・パーティションに渡す
- パーティション内並列処理の使用時に、1 つのデータベース・パーティション内で表データを渡す
- 単一パーティション・データベースの使用時に、1 つのデータベース・パーティション内で表データを渡す

各表キューは単一方向にデータを渡します。コンパイラーはどこで表キューが必要とされているかを判断し、それらをプランに組み込みます。プランが実行されると、データベース・パーティション間の接続を行うとその表キューが開始されます。表キューがクローズされるのは、処理が終了したときです。

表キューには、以下に示すようにいくつかの種類があります。

- *非同期表*キュー。 これらの表キューが非同期と呼ばれるのは、アプリケーションによって FETCH が出される前に、行の読み取りを行うためです。 FETCH が出されたときには、行はこの表キューから取り出されます。

  非同期表キューは、SELECT ステートメントに FOR FETCH ONLY 節を指定した場合に使用されます。行の取り出しだけを行う場合には、非同期表キューが他よりも速い方法になります。
- *同期表*キュー。 これらの表キューが同期と呼ばれるのは、アプリケーションによって FETCH が出されるたびに行を 1 行読み取るためです。各データベース・パーティションでは、カーソルが、そのデータベース・パーティションから次に読み取られる行に位置づけられます。

  同期表キューは、 SELECT ステートメントに FOR FETCH ONLY 節が指定されていない場合に使用されます。パーティション・データベース環境では、行の更新を行う場合には、データベース・マネージャーは同期表キューを使用します。
- マージ*表*キュー。

  これらの表キューは、順序を保存します。
- *非*マージ*表*キュー。

  これらの表キューは「正規」表キューとも呼ばれます。この表キューは、順序を保存しません。
- *Listener 表*キュー *(LTQ と呼ばれることもある)*

  これらの表キューは、相関副照会とともに使用されます。相関値が副照会に渡された後、このタイプの表キューを使用して、結果が親照会ブロックに戻されます。

パーティション・データベース環境での結合方式

以下の図はパーティション・データベース環境における結合方式を示します。

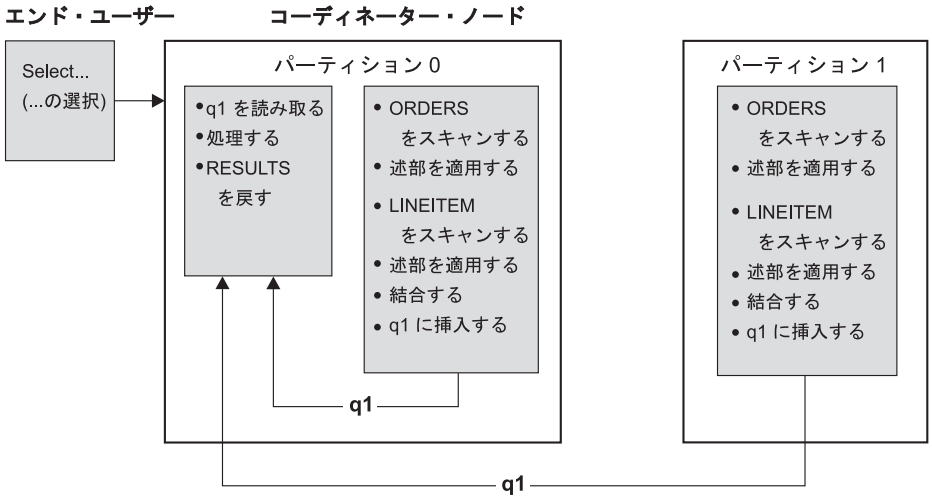
注: 図中の q1、q2、q3 は、例に出てくる表キューと対応しています。図に示されている表は、これらのシナリオの目的に合わせて、 2 つのデータベース・パーティションにまたがって分割されています。矢印は、表キューが送られる方向を示します。なお、コーディネーター・ノードはデータベース・パーティション 0 です。

コロケーテッド結合

コロケーテッド結合はデータがあるデータベース・パーティションでローカルに発生します。結合の完成後、そのデータベース・パーティションはデータを他のデータベース・パーティションに送信します。オプティマイザーがコロケーテッド結合を処理するためには、結合される表は連結され、対応する分散キーのすべての対が等価結合述部に入れられなければなりません。

次の図に例を示します。

注: 複製マテリアライズ照会表はコロケーテッド結合の可能性を高めます。



LINEITEM 表と ORDERS 表は ORDERKEY 列上でパーティション化される。
結合は、各データベース・パーティションでローカルに行われる。
この例では、結合述部は次のように想定されている。
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

図 54. コロケーテッド結合の例

外部表のブロードキャスト結合

外部表のブロードキャスト結合は、結合される表の間に等価結合述部がない場合に使用できる並列結合方式です。また、この結合方式は、これが最も費用対効果が良い結合方式である状況でも使用されます。たとえば、外部表のブロードキャスト結合は、非常に大きな表が 1 つと非常に小さな表が 1 つあり、どちらの表も結合述部列上で分割されていない場合に使用されます。両方の表をパーティションに分割するよりも、小さな表を大きな表にブロードキャストするほうがコストがかからない可能性があります。次の図に例を示します。

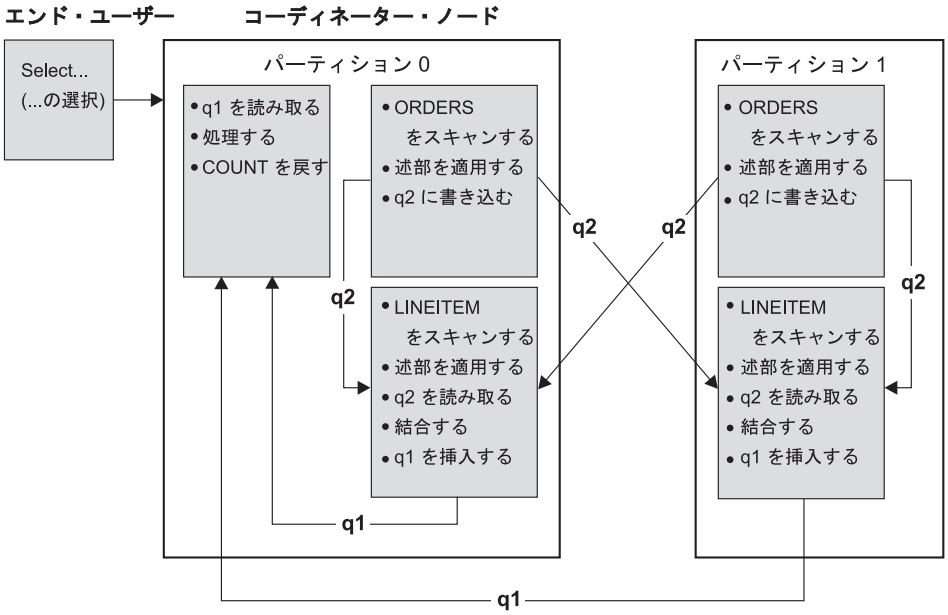
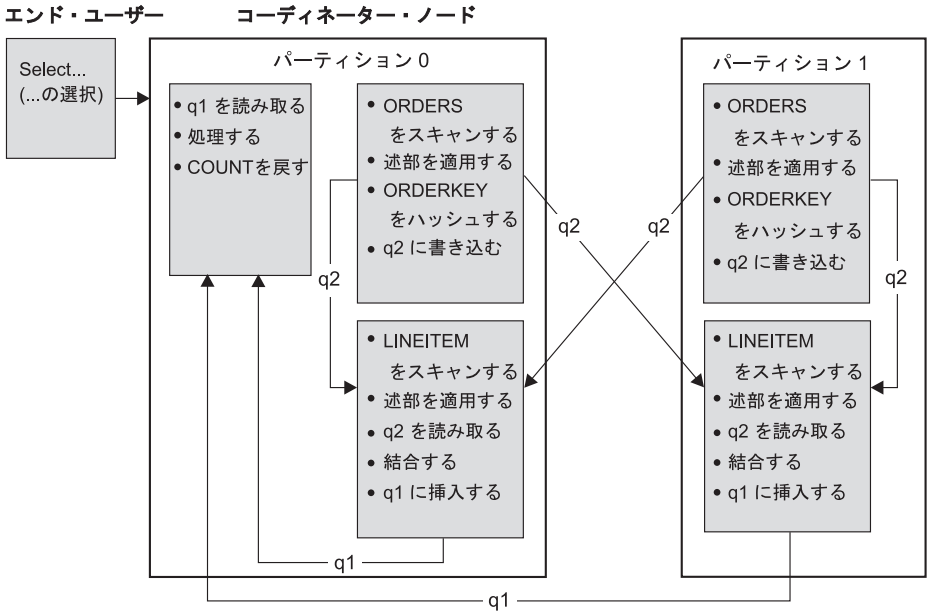


図 55. 外部表のブロードキャスト結合の例

外部表の指示結合

外部表の指示結合方式では、外部表の各行を内部表の分割属性に基づいて内部表の一部分に送ります。結合は、このデータベース・パーティション上で行われます。次の図に例を示します。



LINEITEM 表は ORDERKEY 列上でパーティション化される。
ORDERS 表は別の列でパーティション化される。
ORDERS 表がハッシュされて、適切な LINEITEM 表の
データベース・パーティションに送られる。

この例では、結合述部は次のように想定されている。
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

図 56. 外部表の指示結合の例

内部表および外部表の指示結合

内部表および外部表の指示結合方式では、結合を行う列の値に基づいて、外部表および内部表の行がデータベース・パーティションのセットに送られます。結合は、これらのデータベース・パーティション上で行われます。次の図に例を示します。例は次の図で示します。

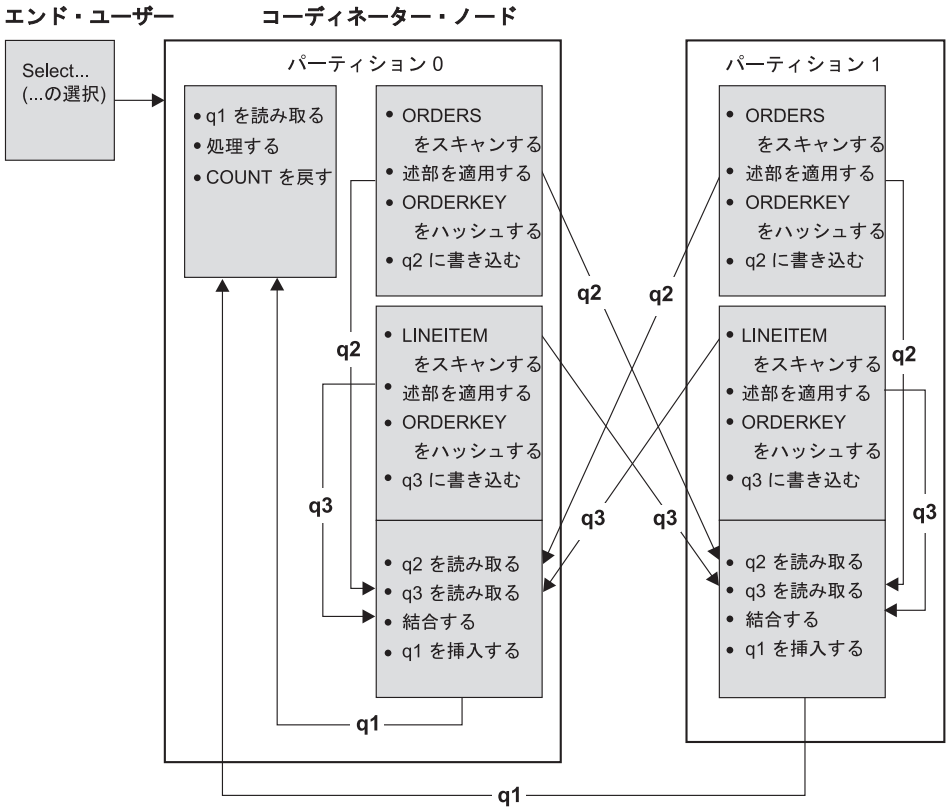


図 57. 内部表および外部表の指示結合の例

内部表のブロードキャスト結合

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

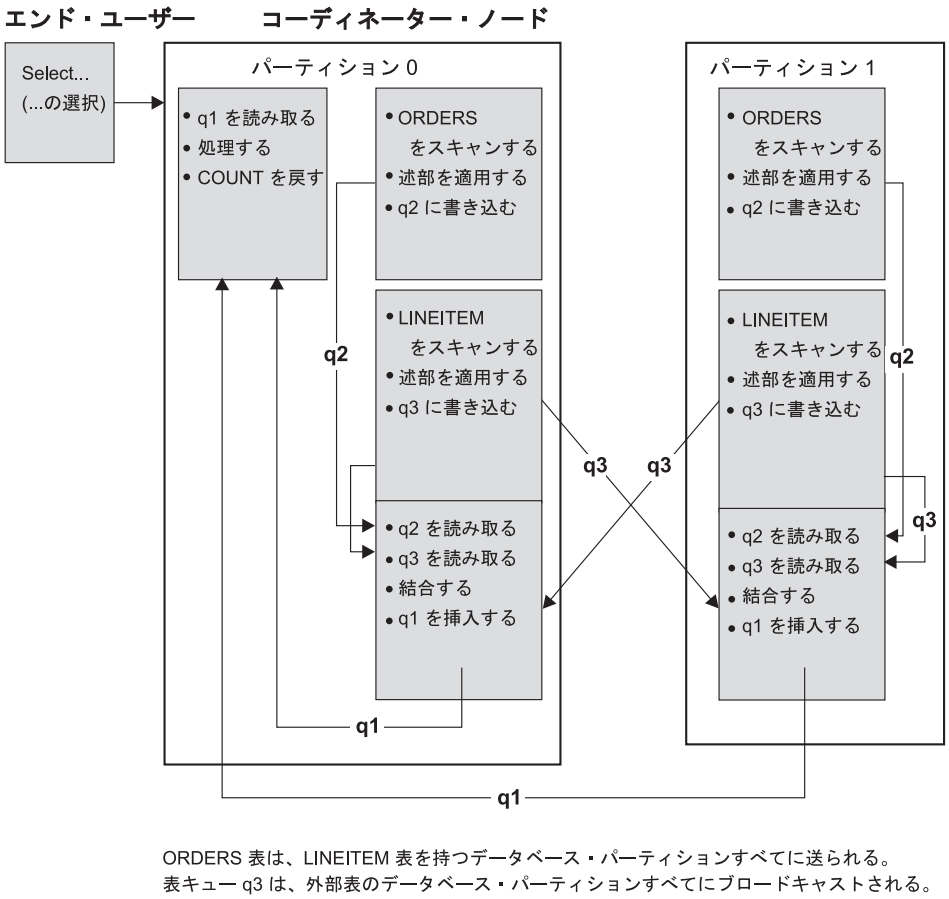


図 58. 内部表のブロードキャスト結合の例

内部表の指示結合

内部表の指示結合方式では、内部表の各行を、外部表の分割属性に基づいて外部結合表のデータベース・パーティションの 1 つに送ります。結合は、このデータベース・パーティション上で行われます。次の図に例を示します。

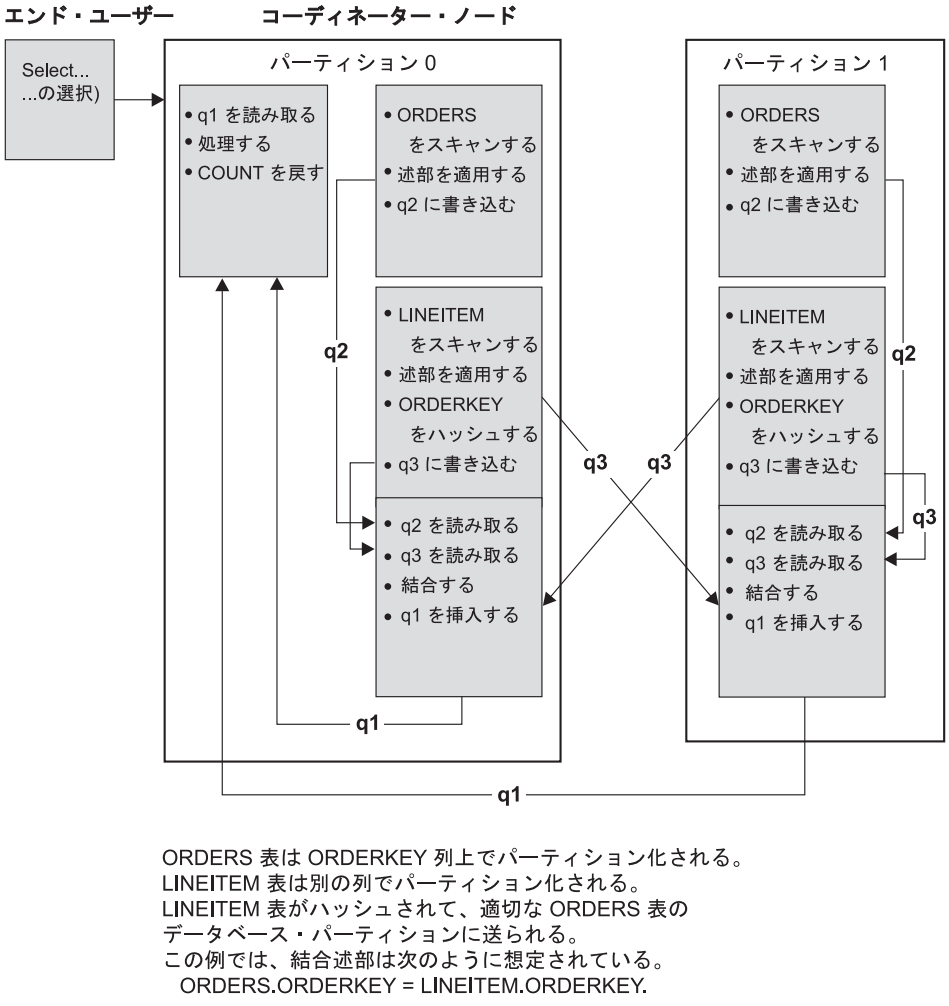


図 59. 内部表の指示結合の例

パーティション・データベース環境の複製されたマテリアライズ照会表

複製マテリアライズ照会表は、データベースが表データの事前計算された値を管理できるようにすることによって、パーティション・データベース環境で頻繁に実行される結合のパフォーマンスを改善します。

照会および複製マテリアライズ表の例を考えてみてください。次のような前提事項があります。

- SALES 表は、複数パーティション表スペース REGIONTABLESPACE にあり、REGION 列で分割されています。
- EMPLOYEE 表および DEPARTMENT 表が単一パーティション・データベース・パーティション・グループにあります。

EMPLOYEE 表の情報に基づき、複製マテリアライズ照会表を作成します。

```
CREATE TABLE R_EMPLOYEE
  AS (
      SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT
```

```
      FROM EMPLOYEE
     )
DATA INITIALLY DEFERRED REFRESH IMMEDIATE
IN REGIONTABLESPACE
REPLICATED;
```

複製マテリアライズ照会表の内容を更新するには、次のステートメントを実行します。

```
REFRESH TABLE R_EMPLOYEE;
```

注: REFRESH ステートメントを使用した後は、他の表と同様に複製表で RUNSTATS を実行する必要があります。

次の例は、従業員ごとの売上、部署の合計、総合計を計算します。

```
SELECT d.mgrno, e.empno, SUM(s.sales)
FROM   department AS d, employee AS e, sales AS s
WHERE  s.sales_person = e.lastname
   AND e.workdept = d.deptno
GROUP BY ROLLUP(d.mgrno, e.empno)
ORDER BY d.mgrno, e.empno;
```

1 つのデータベース・パーティションにしか存在しない EMPLOYEE 表を使用するのではなく、データベース・マネージャーは SALES 表が保管されている各データベース・パーティションで複製される R_EMPLOYEE 表を使用します。結合を計算するために、従業員の情報をネットワークを超えてそれぞれのデータベース・パーティションに移動させる必要はないので、パフォーマンスが向上します。

コロケーテッド結合における複製マテリアライズ照会表

複製マテリアライズ照会表は結合のコロケーションでも助けになります。たとえば、スター・スキーマに 20 のノードにまたがる大規模なファクト表がある場合、ファクト表とディメンション表の結合はこれらの表が連結されていると最も効率的です。同一のデータベース・パーティション・グループにすべての表があれば、多い場合でも 1 つのディメンション表がコロケーテッド結合のために正しくパーティション化されます。他のディメンション表はすべてコロケーテッド結合では使用することができません。それは、ファクト表上の結合列がファクト表の分散キーと対応していないためです。

C1 で分割された FACT (C1、C2、C3、...) という表、C1 で分割された DIM1 (C1、dim1a、dim1b、...) という表、C2 で分割された DIM2 (C2、dim2a、dim2b、...) という表などがある場合を考えてみてください。

この場合、述部 DIM1.C1 = FACT.C1 は連結できるので、 FACT と DIM1 の間の結合は完全であることが分かります。これらの表は両方とも C1 列で分割されています。

しかし述部 WHERE DIM2.C2 = FACT.C2 による DIM2 の結合は連結できません。これは FACT が C1 列で分割されており、C2 列ではないからです。この場合、DIM2 をファクト表のデータベース・パーティション・グループに複製して、データベース・パーティションごとにローカルに結合するようにできます。

注: この複製マテリアライズ照会表についての説明はデータベース内複製と関連しています。データベース間複製はサブスクリプション、コントロール表、異なったデータベース、および異なったオペレーティング・システムに配置されたデータと関連しています。

複製マテリアライズ照会表を作成する場合、ソース表はデータベース・パーティション・グループの単一ノード表でもマルチノード表でも構いません。ほとんどの場合、複製される表は小さく、単一ノード・データベース・パーティション・グループ内に配置することができます。表からの列のサブセットだけを指定することにより、または述部を使用して行数を指定することにより、さらには両方の方式を使用することにより、複製されるデータを制限することができます。複製マテリアライズ照会表を機能させるにはデータ・キャプチャー・オプションは必要ありません。

ソース表のコピーをすべてのデータベース・パーティションに作成するため、マルチノード・データベース・パーティション・グループに複製マテリアライズ照会表を作成することもできます。すべてのデータベース・パーティションに対しソース表をブロードキャストするよりも、大規模なファクト表とディメンション表間の結合は、この環境内でローカルで行う方が良いです。

複製された表の索引は、自動的に作成されません。ソース表のものとは異なる索引を作成することができます。しかし、ソース表になかった制約違反を防ぐため、ユニーク索引の作成や複製された表に制約を加えることはできません。制約はソース表に同じ制約があっても許可されません。

複製された表は照会内で直接参照できますが、特定のパーティション上の表データを見るために複製された表で NODENUMBER() 述部を使用することはできません。

EXPLAIN 機能を使用して、複製マテリアライズ照会表が照会のためのアクセス・プランで使用されたかどうかを調べてください。オプティマイザーが選択するアクセス・プランが複製マテリアライズ照会表を使用するかどうかは、結合される必要のある情報に依存します。オプティマイザーがオリジナル・ソース表をデータベース・パーティション・グループの他のデータベース・パーティションにブロードキャストするほうがコストがかからないと判断した場合、オプティマイザーが複製マテリアライズ照会表を使用しない場合もあります。

レッスン 4. パーティション・データベース環境でのアクセス・プランの改善

さまざまなチューニング・アクティビティーを実行したときに、アクセス・プランと、基本照会の関連ウィンドウがどのように変化するかを学習します。

runstats コマンドを実行したり適切な索引を追加すると、どんな単純な照会でもアクセス・プランの推定合計コストが向上することを、一連の例と図により学習します。

Visual Explain を使いこなせば、様々な方法でも照会をチューニングすることができるようになります。

アクセス・プラン・グラフでの作業

4 つのサンプル EXPLAIN スナップショットを使用して、チューニングがデータベース・パフォーマンスの重要な部分であることを学習します。

EXPLAIN スナップショットに関連付けられた照会は、1 番から 4 番までです。どの照会も同じ SQL または XQuery ステートメント (レッスン 1 を参照) を使用します。

```
SELECT S.ID,SNAME,O.DEPTNAME,SALARY+COMM
FROM ORG O, STAFF S
WHERE
 O.DEPTNUMB = S.DEPT AND
 S.JOB <> 'Mgr' AND
 S.SALARY+S.COMM > ALL ( SELECT ST.SALARY*.9
                         FROM STAFF ST
                         WHERE ST.JOB='Mgr' )
ORDER BY S.NAME
```

ただし、それぞれの照会では、それ以前に実行した照会よりも多くのチューニング技法を使用します。たとえば、Query 1 ではパフォーマンスのチューニングを実行しませんでしたが、 Query 4 ではチューニングを最も多く実行しました。それぞれの照会の違いは、以下のとおりです。

Query 1
索引または統計を使用しない照会の実行
Query 2
照会での表および索引の現在の統計の収集
Query 3
照会で複数の表を結合するために使用される列に対する索引の作成
Query 4
表の列に対する追加の索引の作成

これらの例は、7 個の物理ノードが搭載されている RS/6000 SP マシンで、パーティション間並列処理を使用して生成されました。

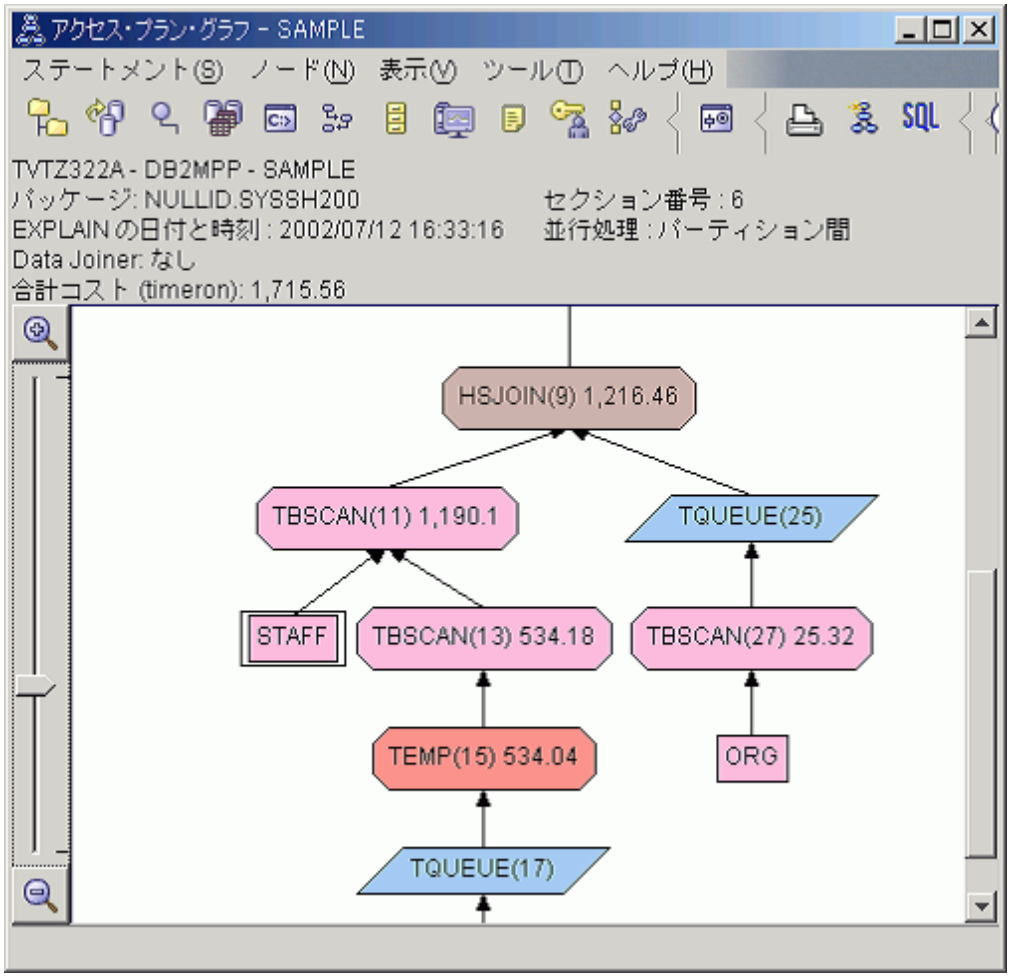
パーティション・データベース環境における索引および統計を使用しない照会の実行

この例では、索引または統計を使用しない SQL 照会のためにアクセス・プランを作成しました。

この照会 (Query 1) のアクセス・プラン・グラフを表示するには、以下の手順に従ってください。

1. コントロール・センターで、SAMPLE データベースが表示されるまでオブジェクト・ツリーを展開します。
2. データベースを右クリックし、ポップアップ・メニューから「**EXPLAIN されたステートメントヒストリーの表示 (Show explained statements history)**」 を選択します。
   「EXPLAIN されたステートメント履歴」ウィンドウがオープンします。

3. Query Number 1 という名前の項目をダブルクリックします (**Query Number** 列が見つからない場合は、画面を右にスクロールしてください)。
ステートメントの「アクセス・プラン・グラフ」ウィンドウがオープンします。



以下の質問に答えることにより、照会を改善する方法を学習します。

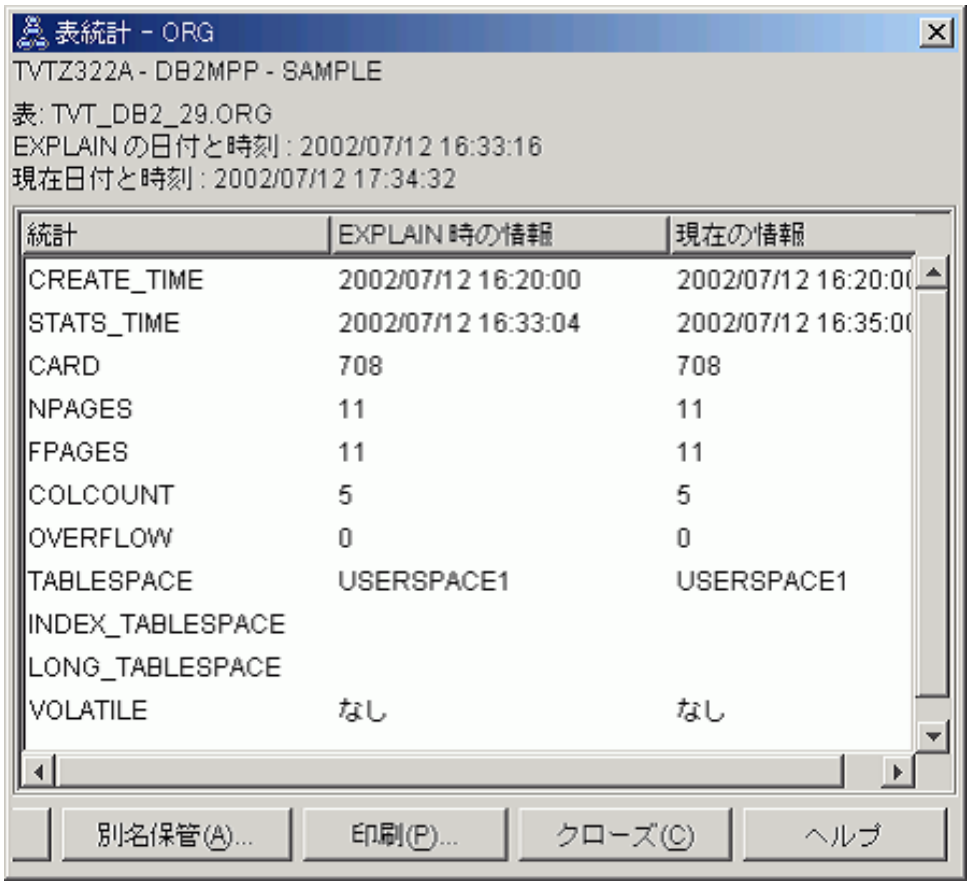
1. 照会で使用する各表の最新統計が存在していますか?

   照会で参照する各表の最新の統計が存在しているかどうかを確認するには、アクセス・プラン・グラフでそれぞれの表ノードをダブルクリックします。対応する「表統計」ウィンドウがオープンし、**EXPLAIN 時の情報**列の **STATS_TIME** 行に、スナップショットの作成時に統計が収集されていないことを意味する「更新されていない統計」という言葉が表示されます。

   最新の統計が存在していない場合、オプティマイザーはデフォルトの統計を使用しますが、この統計は実際の統計とは異なる可能性があります。「表統計」ウィンドウの **EXPLAIN 時の情報**列に「デフォルト」という語が表示されている場合、デフォルトの統計が使用されたことを示します。

   ORG 表に関する「表統計」ウィンドウの情報を見ると、オプティマイザーでデフォルトの統計が使用されたことがわかります (EXPLAIN 時の情報の隣の値がそのことを表している)。スナップショットを作成したときに実際の統計は使用

可能でなかったため (**STATS_TIME** 行の値がそのことを表している)、デフォルトの統計が使用されました。



2. アクセス・プランでは、データへの最適なアクセス方法が使用されていますか?

   このアクセス・プランには、索引スキャンではなく表スキャンが含まれています。表スキャンは八角形で表示されており、TBSCAN 演算子というラベルが付けられています。索引スキャンが使用された場合、索引スキャンはひし形で表示され、IXSCAN というラベルが付けられます。抽出するデータの量が少ない場合は、表スキャンを実行するより、表に対して作成された索引を使用した方がコスト効率が向上します。

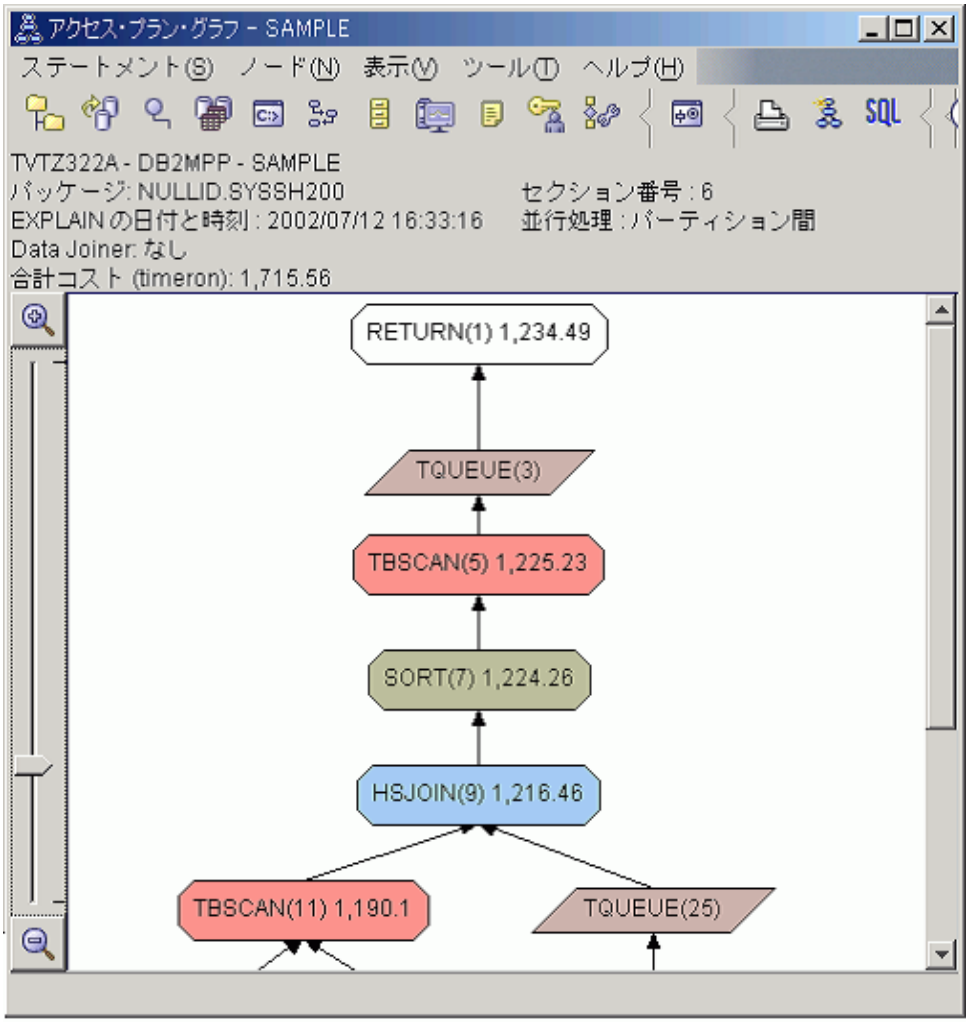
3. このプランの効果性はどれほどですか?

   アクセス・プランの効果を評価するには、アクセス・プランが実際の統計に基づいていなければなりません。このアクセス・プランでは、オプティマイザーがデフォルトの統計を使用しているため、プランの効果を評価することはできません。

   通常は、後で変更済みのアクセス・プランと比較できるように、アクセス・プランの合計見積コストを書き留めておく必要があります。なお、各ノードでリストされているコストは、照会の最初のステップからそのノードまで (ノード自体を含む) の累積値です。

   注: パーティション・データベースの場合、この値は、リソースの使用量が最も多いノードの累積コストです。

「アクセス・プラン・グラフ」ウィンドウでは、グラフの先頭の **RETURN (1)** に示されているとおり、合計コストは約 1,234 timeron です。推定合計コストは、ウィンドウの最上部にも表示されます。



次のステップ

Query 2 に進みます。

Query 2 では、runstats を実行した後の、基本照会用のアクセス・プランを検証します。 runstats コマンドを実行すると、照会で参照するすべての表に関する現在の統計がオプティマイザーに提供されます。

パーティション・データベース環境における runstats を使用した表および索引の現在の統計の収集

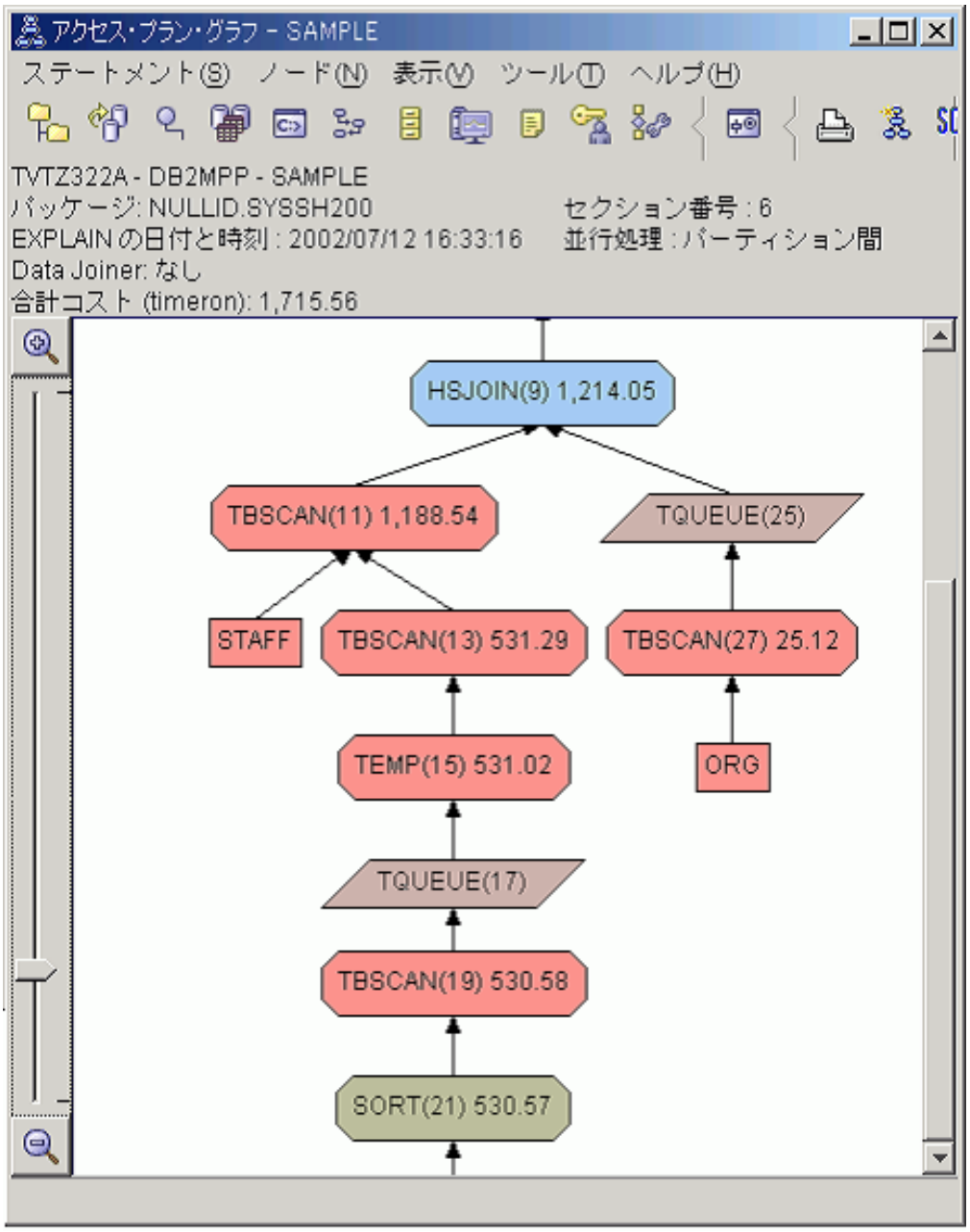
この例では、runstats コマンドを実行して現在の統計を収集し、Query 1 で説明したアクセス・プランを基にしたアクセス・プランを作成します。

runstats コマンドを使用して、表および索引に関する現在の統計を収集することを、強くお勧めします。最後に runstats コマンドを実行してから、大きな更新アクティビティーが発生した場合や新しい索引が作成されている場合は特にその必要があり

ます。これにより、オプティマイザーに、最適なアクセス・プランの決定に使用できる、最も正確な情報が提供されます。現在の統計を利用できないと、オプティマイザーは不正確なデフォルト統計に基づいて効率的でないアクセス・プランを選択してしまう可能性があります。

表の更新を行った後 で、 runstats を使用するようにしてください。そうしないと、オプティマイザーが表を空とみなす可能性があります。この問題は、「オペレーター詳細 (Operator Details)」ウィンドウのカーディナリティーが 0 である場合に明らかとなります。この場合、表更新を完了してから、 runstats コマンドを再実行し、関係する表の EXPLAIN スナップショットを再作成してください。

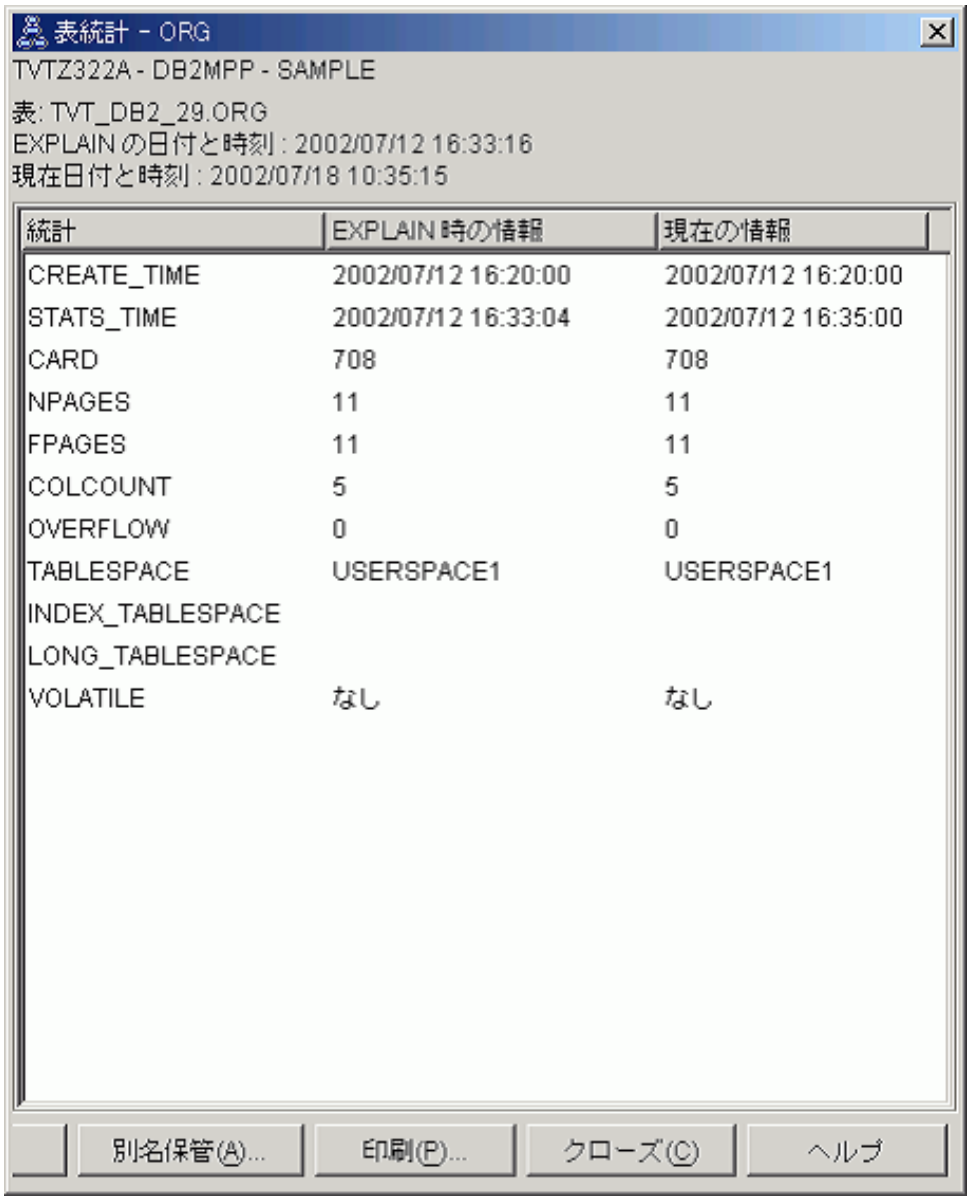
この照会 (Query 2) のアクセス・プラン・グラフを表示するには、「EXPLAIN されたステートメント履歴」ウィンドウで、Query Number 2 という名前の項目をダブルクリックします。このようにステートメントを実行するための「アクセス・プラン・グラフ」ウィンドウがオープンします。



以下の質問に答えることにより、照会を改善する方法を学習します。

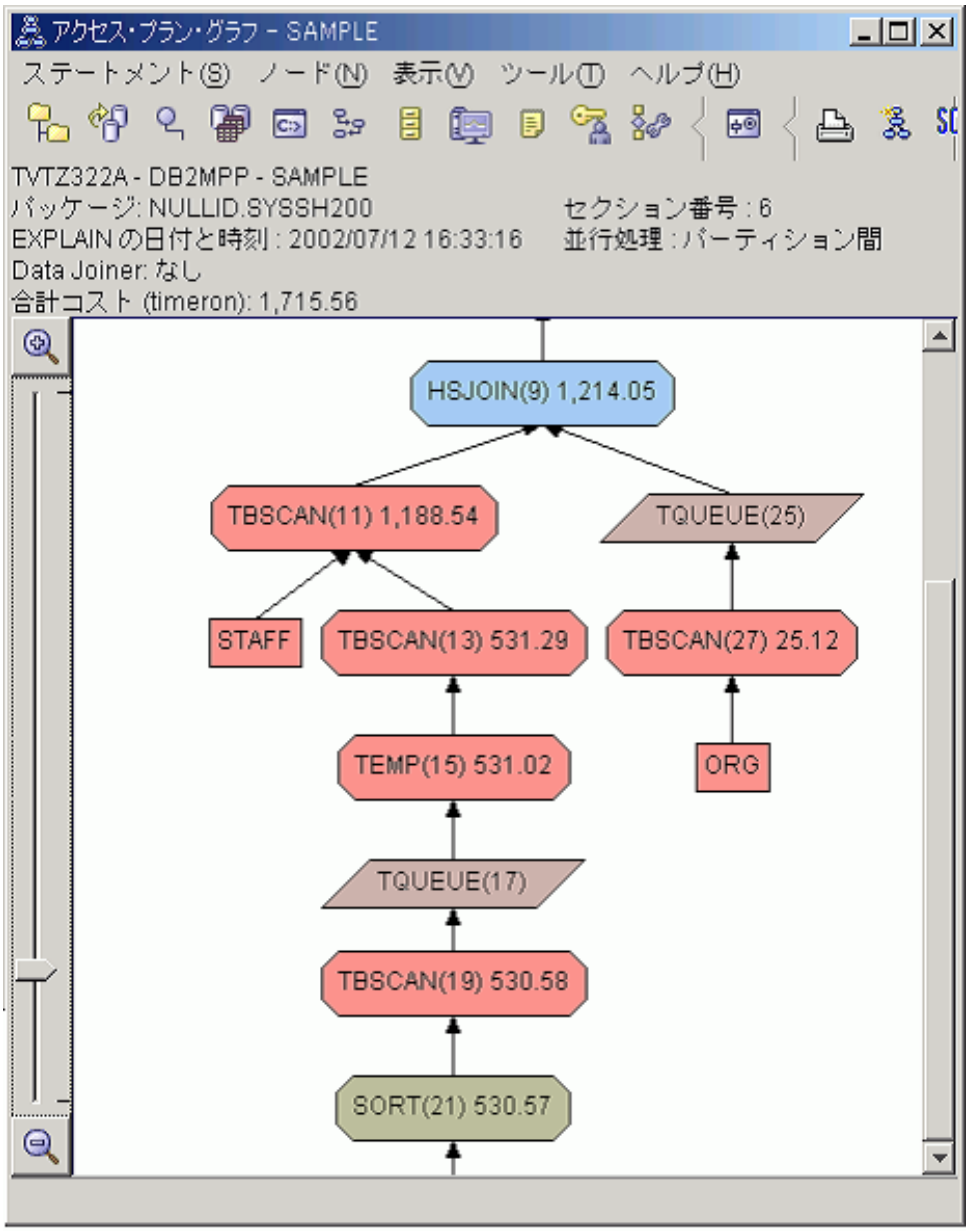
1. 照会で使用する各表の最新統計が存在していますか?

   ORG 表の「表統計」ウィンドウは、オプティマイザーで実際の統計が使用されたことを示しています (STATS_TIME 値は、統計が実際に収集された時刻です)。統計が正確かどうかは、runstats コマンドを実行した後に、表の内容が大幅に変更されたかどうかに依存しています。

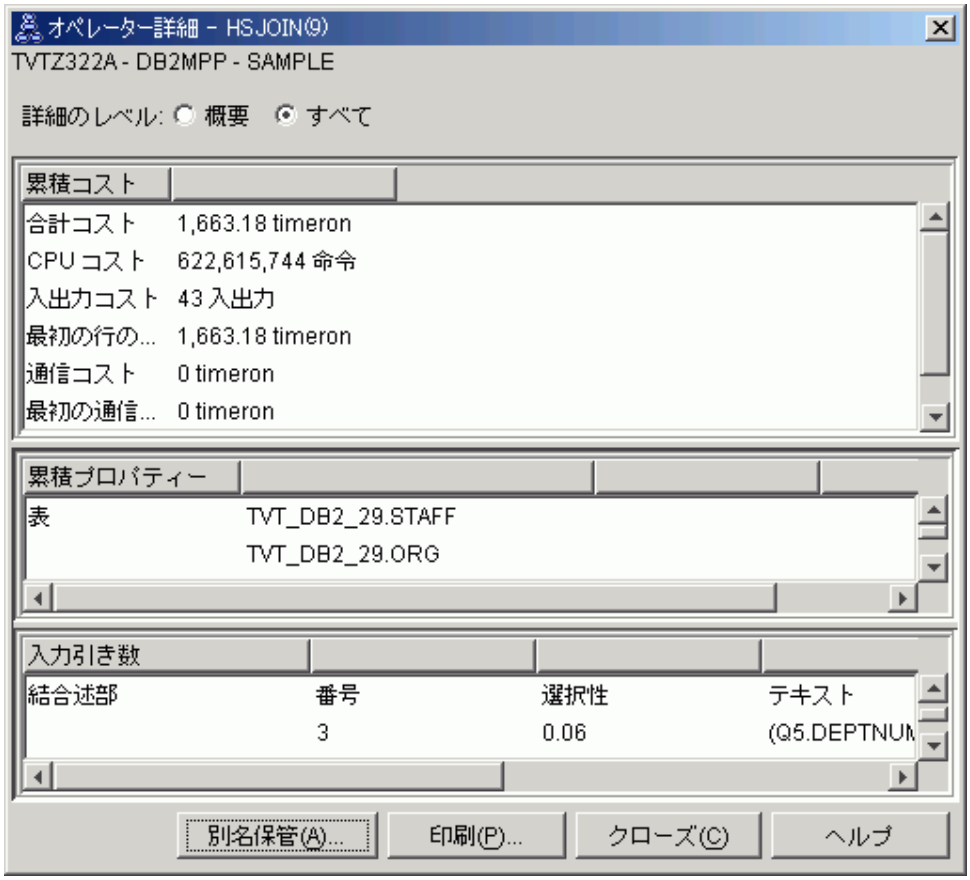


2. アクセス・プランでは、データへの最適なアクセス方法が使用されていますか?

   Query 1 と同様、Query 2 のアクセス・プランでは索引スキャン (IXSCAN) ではなく、表スキャン (TBSCAN 演算子) を使用します。照会で参照する列に対して索引が定義されていないため、現在の統計が存在している場合でも、索引スキャンは実行されません。照会を改善する 1 つの方法は、オプティマイザーのために、表の結合で使用する列 (つまり、結合述部に指定する列) に対して索引を定義することです。この例では、最初のマージ・スキャン結合 HSJOIN (9) を使用します。



HSJOIN (9) 演算子の「演算子詳細」ウィンドウで、「**入力引数**」の下の「**結合述部**」セクションを確認します。この結合操作で使用する列は、**Text** 列にリストされます。この例の場合、列の名前は DEPTNUMB および DEPT です。



3. このアクセス・プランの効果性はどれほどですか?

   アクセス・プランで最新の統計を使用すると、実際の値に近い推定コスト (単位は timeron) を得ることができます。 Query 1 の推定コストはデフォルトの統計に基づいていたため、これら 2 つのアクセス・プラン・グラフの推定コストを比較して、どちらがより効果的かを判断することはできません。この場合、コストが高いか低いかは関係ありません。効果性について有効な値を得るには、実際の統計に基づいたアクセス・プランのコストを比較する必要があります。

次のステップ

Query 3 に進みます。

Query 3 では、DEPTNUMB 列および DEPT 列に索引を追加したときの効果を検証します。結合述部で使用される列に対して索引を追加すると、パフォーマンスを向上させることができます。

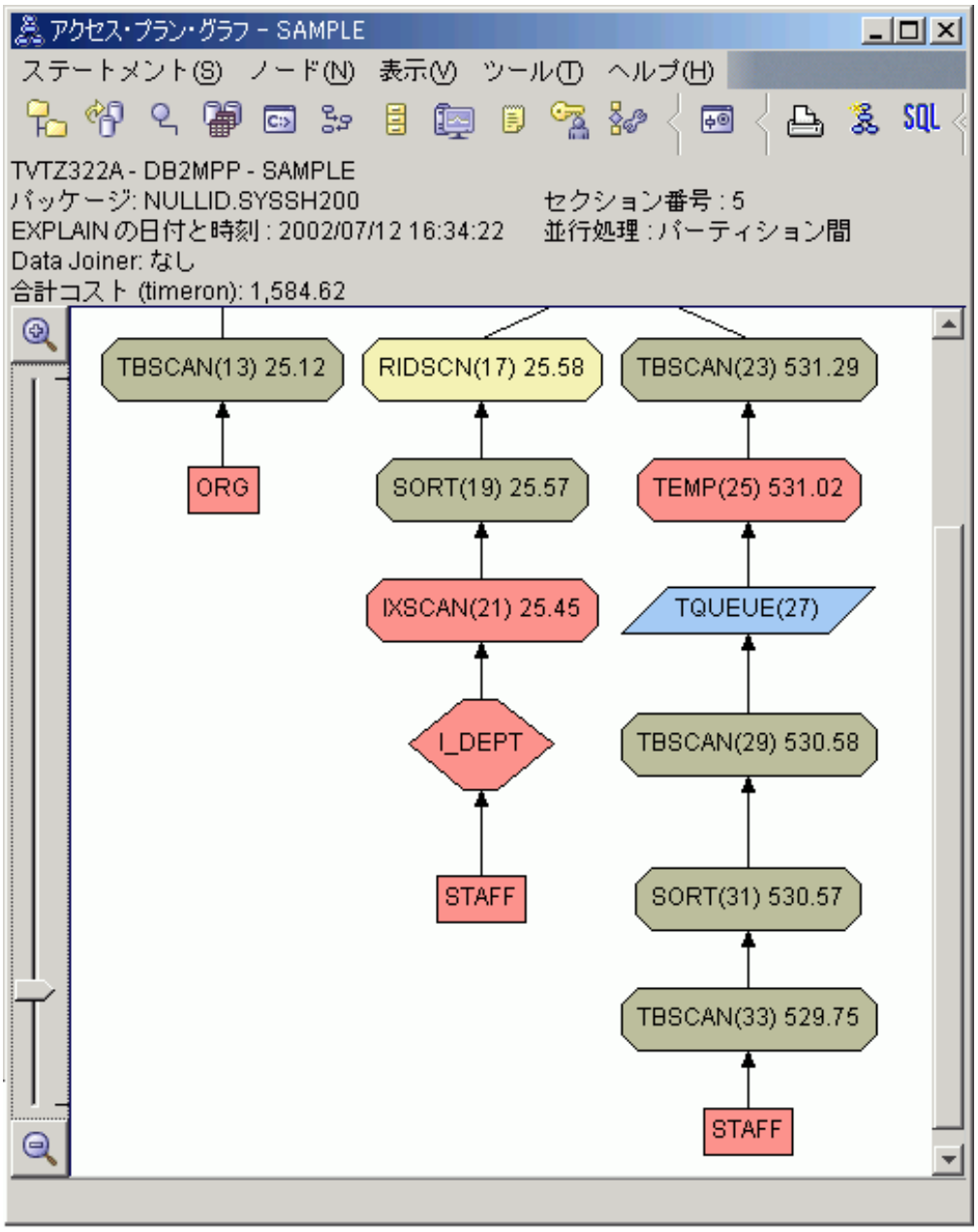
パーティション・データベース環境において、照会で複数の表を結合するために使用される列に対する索引の作成

この例では、Query 2 で説明したアクセス・プランをさらにビルドします。具体的には、STAFF 表の DEPT 列と、ORG 表の DEPTNUMB 列に索引を作成します。

注: 推奨されている索引を Design Advisor で作成することができます。

この照会 (Query 3) のアクセス・プラン・グラフを表示するには、「EXPLAIN されたステートメント履歴」ウィンドウで、Query Number 3 という名前の項目をダブルクリックします。このようにステートメントを実行するための「アクセス・プラン・グラフ」ウィンドウがオープンします。

注: DEPTNUM に対して索引を作成しましたが、オプティマイザーはその索引を使用しませんでした。

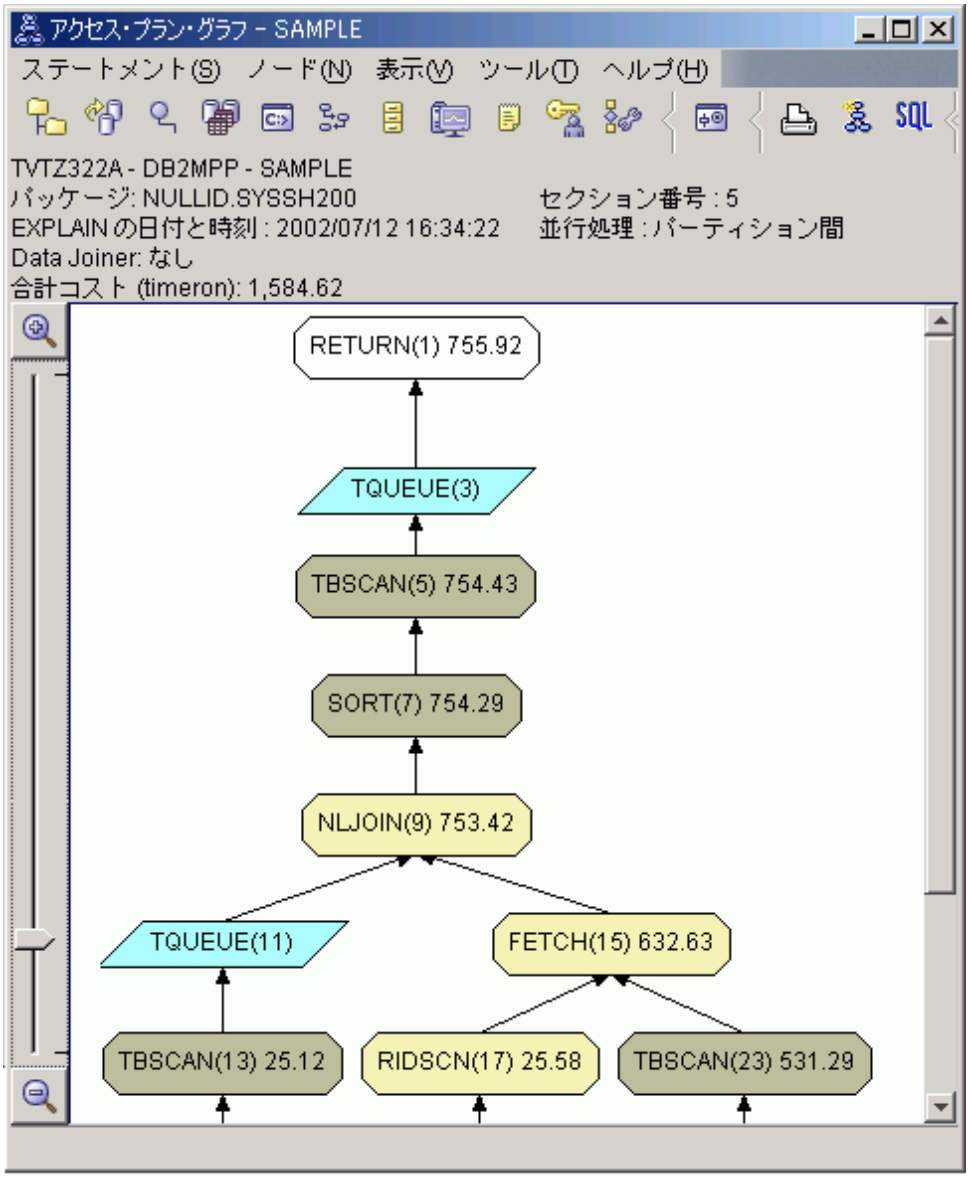


以下の質問に答えることにより、照会を改善する方法を学習します。

1. 索引の追加によりアクセス・プランはどのように変化しましたか?

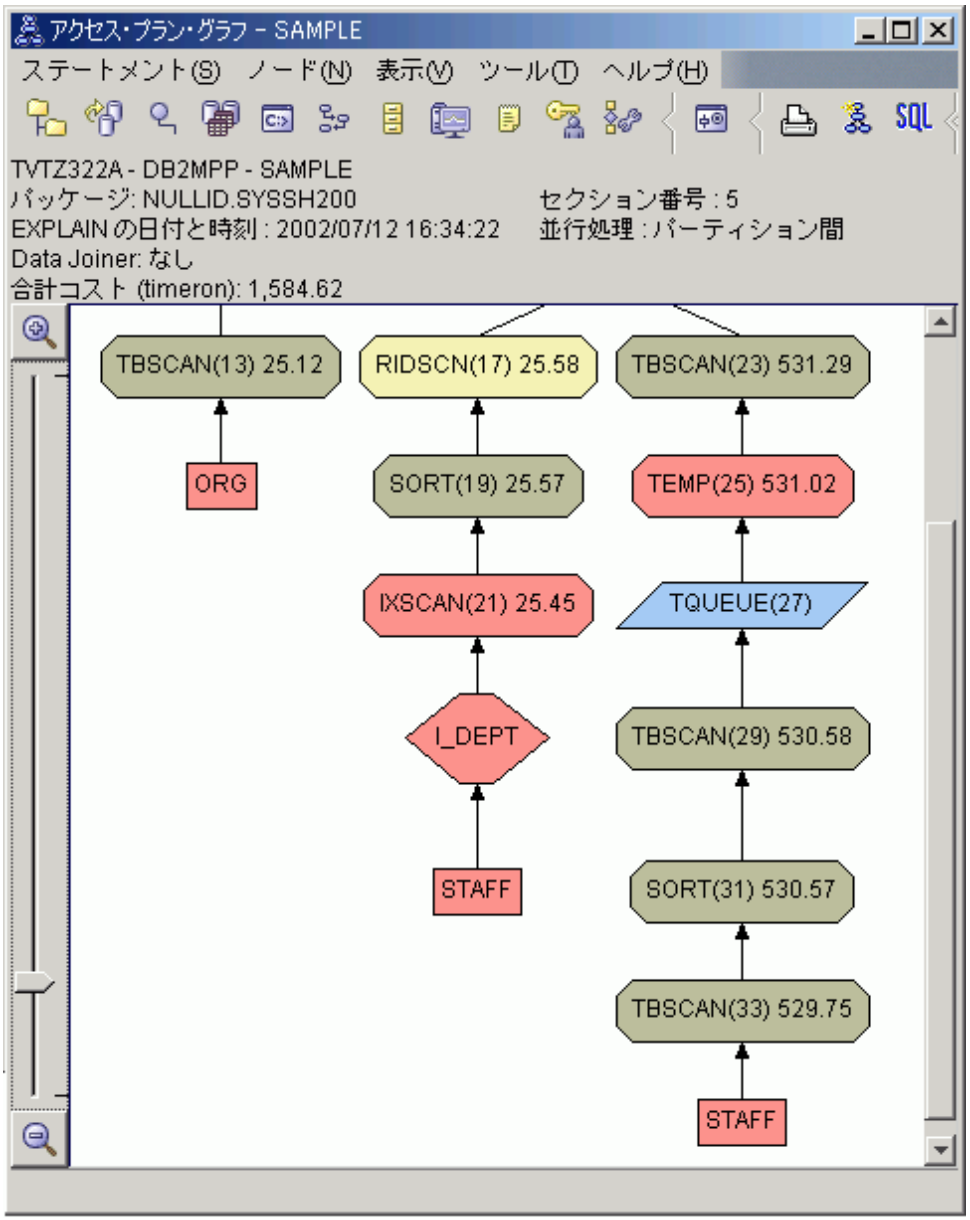
   STAFF 表のすぐ上に、新しいひし形のノード **I_DEPT** が追加されました。このノードは、DEPT に対して作成された索引を表し、取得する行を判別するため

に、オプティマイザーが表スキャンではなく索引スキャンを使用したことを意味します。

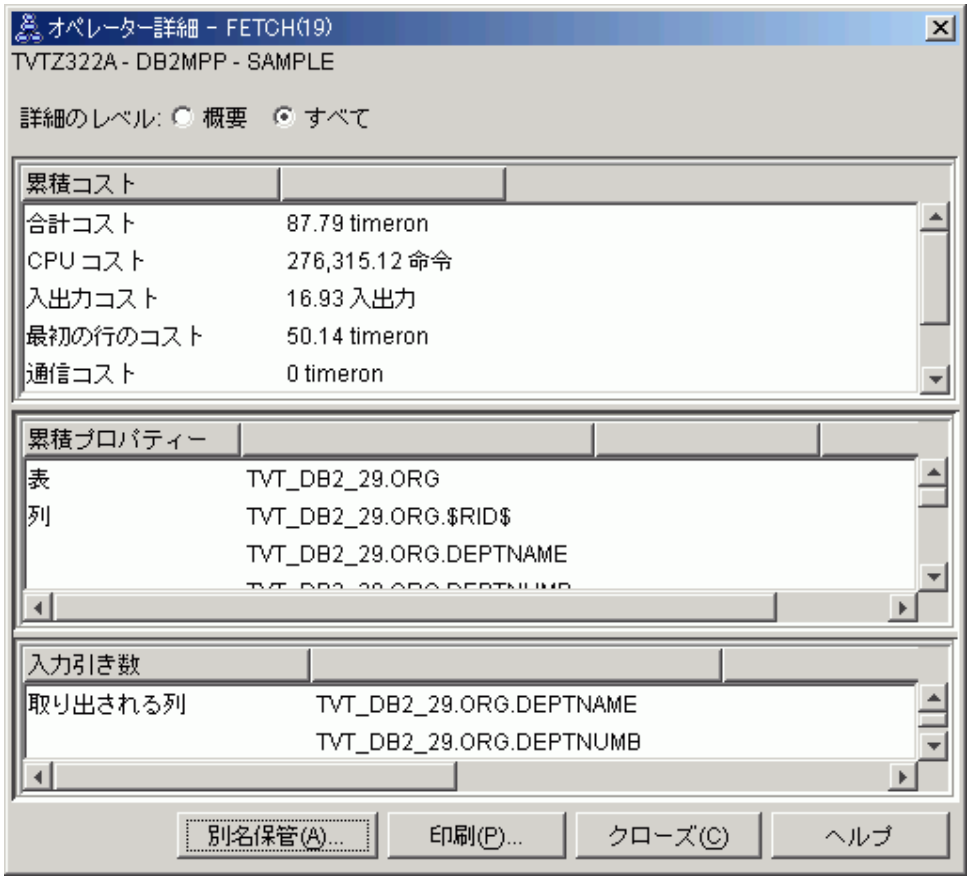


2. アクセス・プランでは、データへの最適なアクセス方法が使用されていますか?

   この照会のアクセス・プランは、ORG 表の DEPTNUMB 列、およびSTAFF 表の DEPT 列に対して索引を作成したことによる効果を示しています。FETCH (15) および IXSCAN (21) によってそれがわかります。 Query 2 では、この索引が定義されていなかったため表スキャンを使用しました。



FETCH (15) 演算子の「演算子詳細」ウィンドウは、この操作で使用されている列を示しています。



索引スキャンとフェッチ操作を組み合わせて使用すると、以前のアクセス・プランで使用した完全表スキャンよりもコストは減少します。

3. このアクセス・プランの効果性はどれほどですか?

   このアクセス・プランの効率は、前の例よりも向上しました。累積コストは、Query 2 では約 1,214 timeron だったのが、 Query 3 では約 755 timeron に減少しました。

次のステップ

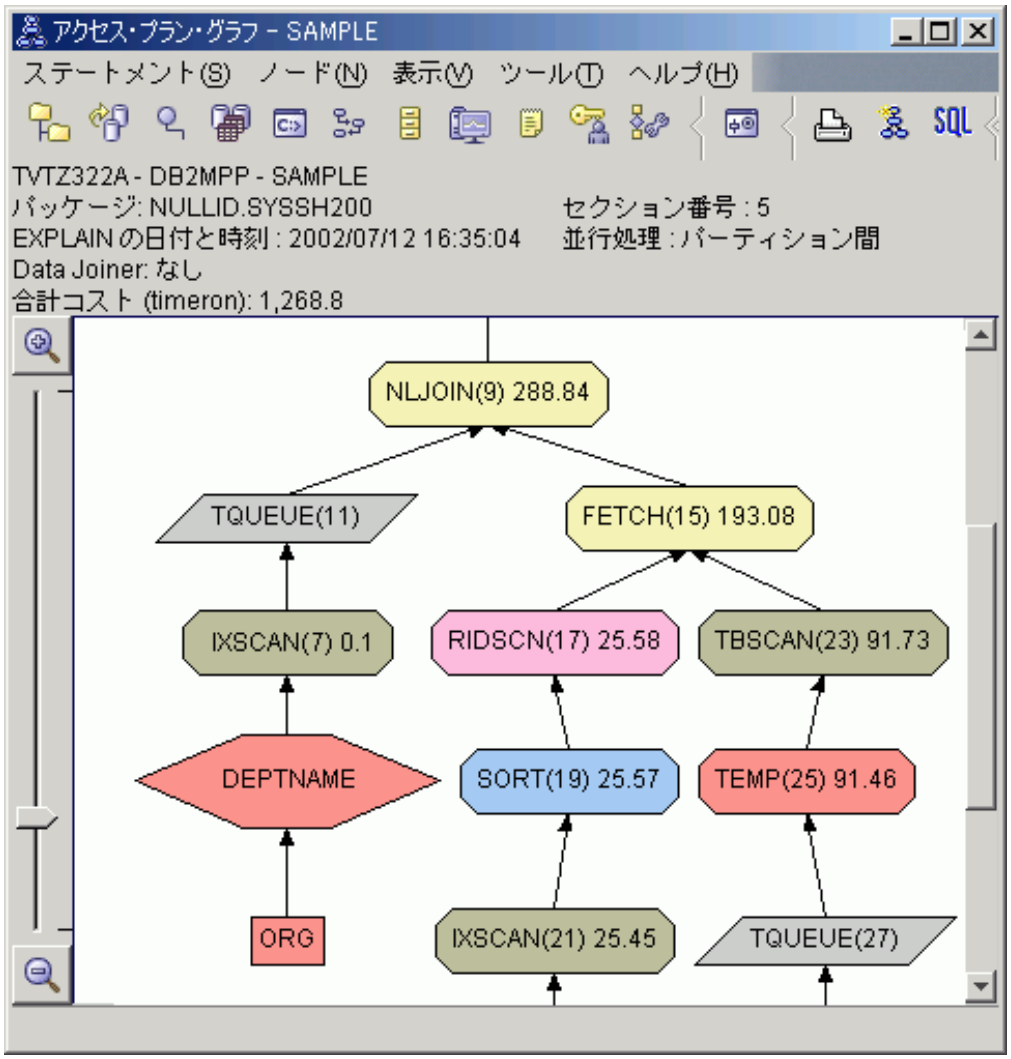
Query 4 に進みます。

Query 4 では、フェッチ操作とインデックス・スキャンを行っていた処理を、フェッチ操作を使用せずに単一の索引スキャンのみを行うように修正します。追加の索引を作成すると、アクセス・プランの推定コストが減少することがあります。

パーティション・データベース環境における表の列に対する追加の索引の作成

この例では、Query 3 で説明したアクセス・プランをさらにビルドします。具体的には、STAFF 表の JOB 列に対して索引を作成し、 ORG 表に定義されている既存の索引に DEPTNAME を追加します。 (別の索引を追加すると、それに伴って追加のアクセスが必要になってしまいます。)

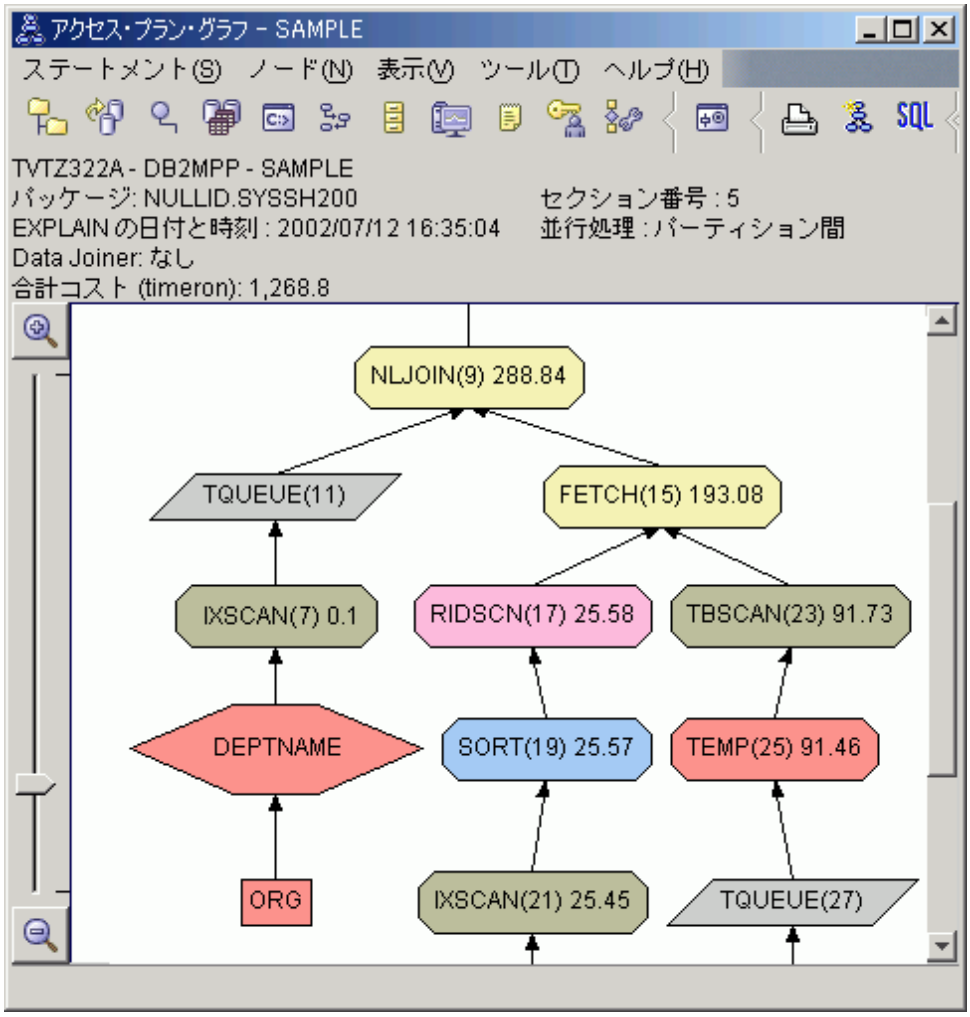
この照会 (Query 4) のアクセス・プラン・グラフを表示するには、「EXPLAIN されたステートメント履歴」ウィンドウで、Query Number 4 という名前の項目をダブルクリックします。このようにステートメントを実行するための「アクセス・プラン・グラフ」ウィンドウがオープンします。



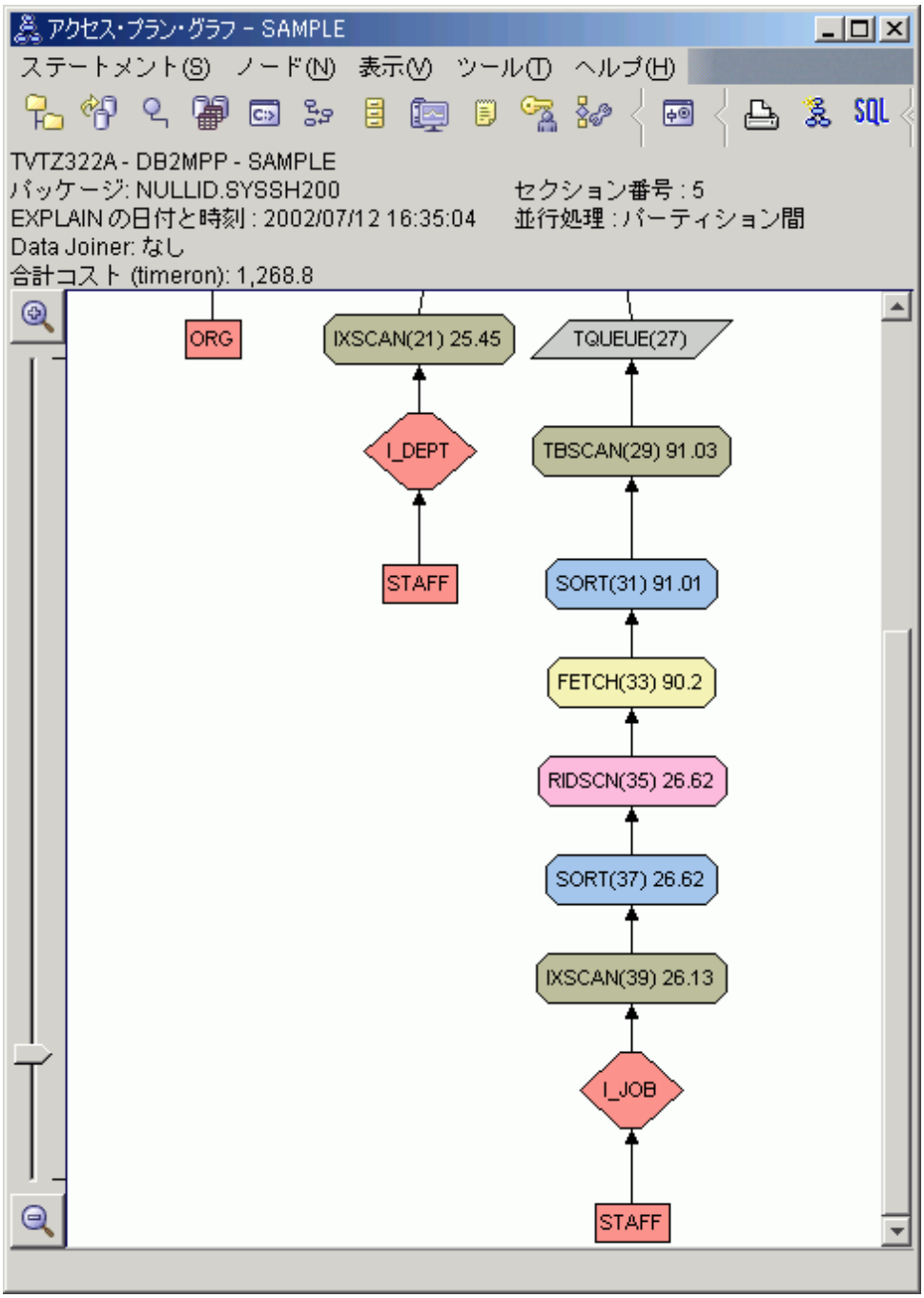
以下の質問に答えることにより、照会を改善する方法を学習します。

1. 追加の索引の作成によりプロセス・プランはどのように変化しますか?

   アクセス・プラン・グラフの中間部に表示されている ORG 表で、以前は表スキャンが使用されていたのが、現在は索引スキャンのみ IXSCAN (7) に変化したことに注目してください。 ORG 表の索引に DEPTNAME 列を追加すると、オプティマイザーは、表スキャンが関係したアクセスを最適化することができます。



アクセス・プラン・グラフの最終部に表示されている STAFF 表で、以前は索引スキャンとフェッチ操作が使用されていたのが、現在は索引スキャン IXSCAN (39) のみに変化したことに注目してください。 STAFF 表に JOB 索引を作成すると、オプティマイザーでは、フェッチ操作のための余分なアクセスを省略することができます。



2. このアクセス・プランの効果性はどれほどですか?

   このアクセス・プランのコスト効率は、前の例よりも向上しました。累積コストは、Query 3 では約 753 timeron だったのが、 Query 4 では約 288 timeron に減少しました。

次のステップ

独自の SQL または XQuery ステートメントのパフォーマンスの改善

パフォーマンスを改善するために行える追加ステップについての詳細情報を調べるには、*DB2 インフォメーション・センター* を参照してください。その後 Visual Explain に戻って、アクションの影響にアクセスします。

第 26 章 データの再配分

データの再配分は、主に、ストレージ・スペースの使用量のバランスを取るため、パーティションを追加または除去するときにパーティション・データベース環境内でデータを移動したり、データベースのシステム・パフォーマンスを向上させたり、他のシステム要件を満たすために実行されるデータベース管理操作です。

データの再配分は、以下のいずれかのインターフェースを使用して実行できます。

- REDISTRIBUTE DATABASE PARTITION GROUP コマンド
- ADMIN_CMD システム定義プロシージャー
- STEPWISE REDISTRIBUTE_DBPG システム定義プロシージャー
- sqludrdt API

パーティション・データベース内のデータの再配分は、以下のいずれかの理由によって行われます。

- 新規のデータベース・パーティションがデータベース環境に追加されるか、または既存のデータベース・パーティションが除去されるたびに、データのバランスを再調整する。
- ユーザー固有のパーティション間のデータ配分を導入する。
- 特定のパーティション内で機密データを分離することによってそのデータを保護する。

データの再配分は、カタログ・データベース・パーティションでデータベースに接続されることによって、またサポートされているインターフェースのいずれかを使用して特定のパーティション・グループに対してデータ再配分操作を開始することによって、実行されます。データを再配分するには、パーティション・グループ内の表に対する分散キーの定義が存在することが必要です。表内のデータの行の分散キー値は、データの行がどのパーティションに保管されるかを判別するのに使用されます。分散キーは、マルチパーティション・データベースのパーティション・グループ内に表が作成されるときに自動的に生成するか、あるいは CREATE TABLE または ALTER TABLE ステートメントを使用して明示的に定義することができます。デフォルトでは、データの再配分によって、特定のノード・グループ内の各表に、表データがデータベース・パーティション内で均等に分割されて再配分されますが、データの配分方法を定義する入力分散マップを指定することによって、スキュー配分などの他の配分を行うことができます。分散マップは、将来の利用のためにデータ再配分操作中に生成することができますが、手動で作成することもできます。

データ再配分に関する制約事項

データの再配分を進める前、またはデータの再配分に関連する問題をトラブルシューティングするときに、データ再配分に関する制約事項に注意するのは重要です。

注: このトピックには、DB2 9.5 フィックスパック 1 をインストールして初めて有効になるキーワード更新について言及している箇所があります。DB2 9.5 フィッ

クスパック 1 がまだ入手できないかインストールされていない場合には、このトピックの DB2 9 バージョンを、次の場所にある DB2 9 インフォメーション・センターから参照してください。http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp

データの再配分には以下の制約事項があります。

- 表がパーティション・キーの定義を持っていないパーティション上でのデータの再配分は制限されます。
- データ再配分の進行中には、以下の制限があります。
  - データベース・パーティション・グループに対する別の再配分操作の開始は制限されます。
  - データベース・パーティション・グループのドロップは制限されます。
  - データベース・パーティション・グループの変更は制限されます。
  - データベース・パーティション・グループ内の任意の表に対する ALTER TABLE ステートメントの実行は制限されます。
  - データの再配分が進行中の表での新規索引の作成は制限されます。
  - データの再配分が進行中の表で定義されていた索引のドロップは制限されます。
  - データの再配分が進行中の表でのデータの照会は制限されます。
  - データの再配分が進行中の表の更新は制限されます。
- NOT ROLLFORWARD RECOVERABLE オプションが指定された REDISTRIBUTE DATABASE PARTITION GROUP コマンドを使用してデータの再配分が開始され、その再配分が進行中のデータベースでの表の更新は制限されます。更新を行うことはできますが、データの再配分が中断されると、データへの変更が失われる場合があるので、この方法は使用しないことを強くお勧めします。
- REDISTRIBUTE DATABASE PARTITION GROUP コマンドが発行されるときに、NOT ROLLFORWARD RECOVERABLE オプションが指定される場合、以下を検討します。
  - データの再配分の一部として行われるデータの変更は、ロールフォワード・リカバリー可能ではありません。
  - 一方、データベースがリカバリー可能である場合、パーティション内の最初の表にアクセスした後、表スペースは BACKUP PENDING 状態にされます。表をこの状態から解除するには、再配分操作が完了するときに、表スペースの変更のバックアップを取る必要があります。
  - データの再配分中、再配分されているデータベース・パーティション・グループにある表のデータは更新できません。その間、データは読み取り専用です。再配分中の表はアクセス不能です。
- 型付き (階層) 表の場合、REDISTRIBUTE DATABASE PARTITION GROUP コマンドが使用されていて、TABLE オプションに値 ONLY が指定されているとき、表名はルート表の名前のみに制限されます。副表の名前は指定できません。
- 範囲パーティション表の場合、データ・パーティション表の範囲間のデータ移動は制限されます。ただし、データの再配分はデータベース・パーティション間のデータ移動についてはサポートされます。

- パーティション表の場合、以下の両方が真でないとき、データの再配分は制限されます。
  - パーティション表のアクセス・モードが、systables.access_mode カタログ表で FULL ACCESS になっている。
  - パーティション表のどのパーティションも、現在アタッチもデタッチもされていない。
- 複製されたマテリアライズ照会表の場合、データベース・パーティション・グループ 内のデータに複製されたマテリアライズ照会表が含まれている場合、データを再配分する前にこれらの表をドロップする必要があります。データの再配分が完了した後に、マテリアライズ照会表を再作成することができます。
- マルチディメンション・クラスタリング (MDC) 表を含むデータベース・パーティションの場合、REDISTRIBUTE DATABASE PARTITION GROUP コマンドの使用が制限され、データベース・パーティション・グループに、クリーンアップが保留になっているロールアウトされたブロックを含むマルチディメンション・クラスタリング表がある場合には、正常に実行されません。これらの MDC 表は、データの再配分が再開または再始動される前にクリーンアップする必要があります。
- 「再配分進行中」状態として DB2 カタログ・ビューに現在マークされている表のドロップは制限されます。この状態の表をドロップするには、表の再配分が完了または異常終了するように、まず ABORT または CONTINUE オプションと適切な表リストを指定して REDISTRIBUTE DATABASE PARTITION GROUP ユーティリティーを実行します。

ノード・グループ内および表での現行のデータ配分の判別

ノード・グループまたは表の現行のデータ配分を判別することは、データの再配分が必要かどうかを判断し、データの配分方法を指定するために使用されるカスタム分散マップを作成するのに役立ちます。

新規データベース・パーティションがデータベース・パーティション・グループに追加される場合、または既存のデータベース・パーティションがデータベース・パーティション・グループからドロップされる場合、すべてのデータベース・パーティション間のデータのバランスを調整するために、データの再配分が実行されます。

データベース・パーティションが追加されていない、またはデータベース・パーティション・グループからドロップされていない場合、データの再配分は通常、データベース・パーティション・グループのデータベース・パーティション間のデータが不均等に配分されているときにのみ指示されます。データの不均等な配分が望ましい場合もあることに注意してください。例えば、いくつかのデータ・パーティションが特に強力なマシン上にある場合、それらのデータベース・パーティションに他のパーティションよりも多くのデータを入れる方が良い場合があります。

データベース・パーティション・グループ内のデータベース・パーティション間の現在のデータ配分について情報を得るには、データベース・パーティション・グループ内の最も大きな表 (あるいは、代表的な表) で、以下の照会を実行します。

```
SELECT DBPARTITIONNUM(column_name), COUNT(*) FROM table_name
      GROUP BY DBPARTITIONNUM(column_name)
      ORDER BY DBPARTITIONNUM(column_name) DESC
```

ここで、column_name は、表 table_name の分散キーの名前です。

照会の結果が、データベース・パーティション間のデータの配分が望ましくないことを示す場合、以下の照会を実行して、ハッシュ・パーティション間のデータの配分を取得します。

```
SELECT PARTITION(column_name), COUNT(*) FROM table_name
      GROUP BY PARTITION(column_name)
      ORDER BY PARTITION(column_name) DESC
```

REDISTRIBUTE DATABASE PARTITION GROUP コマンドの USING DISTFILE オプションが指定されている場合 (配分ファイルのフォーマットの説明については、『REDISTRIBUTE DATABASE PARTITION GROUP コマンド』のコマンド・リファレンスのセクションを参照してください)、この照会の出力を使用して、必要な配分ファイルを簡単に構成できます。

USING DISTFILE オプションが指定されると、REDISTRIBUTE DATABASE PARTITION GROUP コマンドはこのファイル内の情報を使用して、データベース・パーティション間にデータが均等に配分されるように、データベース・パーティション・グループのための新規パーティション・マップを生成します。

均等な配分が望ましくない場合、ユーザーは再配分操作のための独自のターゲット・パーティション・マップを構成することができます。これは、REDISTRIBUTE DATABASE PARTITION GROUP コマンドの USING TARGETMAP オプションを使用して指定できます。

この調査の後、データが均等に配分されているかどうか、またはデータの再配分が必要かどうかが判別できます。データが再配分を必要とする場合、サポートされているインターフェースのいずれかを使用して、システム保守の機会に、再配分を行うよう計画できます。

REDISTRIBUTE DATABASE PARTITION GROUP コマンドを使用したデータベース・パーティションでのデータの再配分

データの再配分は、REDISTRIBUTE DATABASE PARTITION GROUP コマンドを使用して、正常に実行することができます。これは、データの再配分を実行するのに推奨されているインターフェースです。

注: このトピックには、DB2 9.5 フィックスパック 1 をインストールして初めて有効になるキーワード更新について言及している箇所があります。DB2 9.5 フィックスパック 1 がまだ使用できないかインストールされていない場合、DB2 9 インフォメーション・センターにある DB2 9 用の同じトピック (http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/t0005017.htm) を参照してください。

制約事項

- 385 ページの『データ再配分に関する制約事項』を参照。

REDISTRIBUTE DATABASE PARTITION GROUP コマンドを使用して、データベース・パーティション・グループ内のデータベース・パーティション間でデータを再配分するには、次のようにします。

1. データベースのバックアップを実行します。 BACKUP コマンドを参照してください。
2. システム・カタログ表を含むデータベース・パーティションに接続します。 CONNECT コマンドを参照してください。
3. REDISTRIBUTE DATABASE PARTITION GROUP コマンドを出す。

   注: DB2 の以前のバージョンでは、このコマンドは DATABASE PARTITION GROUP キーワードではなく NODEGROUP キーワードを使用していました。

   以下の引数を指定します。

   database partition group name
   データをその中に再配分する、データベース・パーティション・グループを指定する必要があります。

   UNIFORM
   オプション: データを均等に配分し、その後も均等に配分されたままにしておくように指定します。配分タイプが指定されないとき、UNIFORM はデフォルトなので、その他の配分タイプが指定されていない場合には、このオプションの省略も有効です。

   USING DISTFILE *distfile-name*
   オプション: カスタマイズされた配分を使用すること、および必要なデータ・スキューを定義するデータを含む配分ファイルのファイル・パス名を指定します。このファイルの内容は、ターゲット分散マップを生成するのに使用されます。

   USING TARGETMAP *targetmap-name*
   オプション: ターゲット・データ再配分マップが使用されること、およびターゲット再分散マップを含むファイルの名前を指定します。

   詳細については、 REDISTRIBUTE DATABASE PARTITION GROUP コマンド行ユーティリティーの情報を参照してください。
4. コマンドは割り込まれずに実行されます。コマンドが完了するとき、データの再配分が正常に実行された場合は、以下のことを行います。
   - BACKUP PENDING 状態になっている、データベース・パーティション・グループ中のすべての表スペースのバックアップを取ります。あるいは、完全なデータベース・バックアップを実行できます。注: データベースがリカバリー可能で、REDISTRIBUTE DATABASE PARTITION GROUP コマンドの NOT ROLLFORWARD RECOVERABLE オプションが使用される場合、表スペースは BACKUP PENDING 状態になるだけです。
   - 再配分の前にドロップした、複製されたマテリアライズ照会表を再作成します。
   - REDISTRIBUTE DATABASE PARTITION GROUP コマンドの STATISTICS NONE オプションが指定されているか、NOT ROLLFORWARD RECOVERABLE オプションが省略されており (これらは両方とも、データの再配分中に統計が収集されなかったことを意味します)、データベース・パー

ティション・グループ中に統計プロファイルを持つ表がある場合、RUNSTATS コマンドを実行してデータ配分統計を収集し、SQL コンパイラーおよびオプティマイザーが照会のデータ・アクセス・プランを選択するときにそれを使用できるようにします。

- NOT ROLLFORWARD RECOVERABLE オプションを指定した場合、以下のパスにある制御ファイルを削除します。
  - Linux および UNIX オペレーティング・システム: DIAGPATH/redist/*db_name*/*db_partitiongroup_name*/*timestamp*/
  - Windows オペレーティング・システム: DIAGPATH¥redist¥*db_name*¥*db_partitiongroup_name*¥*timestamp*¥

データの再配分が正常に完了し、データの再配分処理についての情報が再配分のログ・ファイルで使用できます。使用された分散マップについての情報は、DB2 Explain 表にあります。

データベース・パーティション・グループ内でのデータの再配分

データの再配分ウィザードを使用して、データベース・パーティション・グループの効果的な再配分プランを立て、データを再配分します。まず、再配分方式とストラテジーを選択してから、より詳細な項目を選択します。

データベース・パーティション・グループを処理するには、*sysadm* または *dbadm* 権限が必要です。

データベース・パーティション・グループ内のデータを再配分するには、次のようにします。

1. 次のようにして、「データの再配分」ウィザードをオープンします。コントロール・センターで、「**データベース・パーティション・グループ**」フォルダーが表示されるまでオブジェクト・ツリーを展開します。既存のデータベース・パーティション・グループは、ウィンドウの右側のコンテンツ・ペインに表示されます。作業したいデータベース・パーティション・グループを右クリックし、ポップアップ・メニューから「**再配分**」を選択すると、データの再配分ウィザードがオープンします。データの再配分ウィザードがオープンします。

   「パーティションの追加」ランチパッドを使用してデータベース・パーティションを追加する または *「パーティションのドロップ」ランチパッドを使用してインスタンスからデータベース・パーティションをドロップする* 操作から、データの再配分ウィザードをオープンすることもできます。
2. ウィザードの該当するページをそれぞれ完了させます。詳しくは、先頭ページにあるウィザードの概要についてのリンクをクリックしてください。データを再配分するための十分な情報をウィザードに指定すると、「**完了**」ボタンが使用できるようになります。

データ再配分のログ・スペース所要量

データの再配分操作を正常に実行するには、データの再配分操作が開始される前に、適切なログ・ファイル・スペースを割り振って、データの再配分が中断されないようにする必要があります。

注: このトピックには、DB2 9.5 フィックスパック 1 をインストールして初めて有効になるキーワード更新について言及している箇所があります。DB2 9.5 フィックスパック 1 がまだ入手できないかインストールされていない場合には、このトピックの DB2 9 バージョンを、次の場所にある DB2 9 インフォメーション・センターから参照してください。http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp

必要なログ・ファイル・スペースの量は、REDISTRIBUTE DATABASE PARTITION GROUP コマンドのどのオプションを使用するかを含めて、複数の要因に依存しています。

REDISTRIBUTE DATABASE PARTITION GROUP コマンドが使用され、かつ、NOT ROLLFORWARD RECOVERABLE オプションが使用**されない場合**、あるいはデータの再配分がロールフォワード・リカバリー可能ではないその他のサポート対象のインターフェースから再配分が実行される場合、以下について検討します。

- ログは、データが再配分されているデータベース・パーティションごとに INSERT 操作 および DELETE 操作に対応できるだけの大きさでなければなりません。ロギング所要量は、最大量のデータを失うデータベース・パーティション、または最大量のデータを獲得するデータベース・パーティションで最も大きくなります。
- より多くのデータベース・パーティションに移動しようとしている場合は、新しいデータベース・パーティション数に対する現行データベース・パーティションの比率を使用して、INSERT 操作および DELETE 操作の数を見積もってください。例えば、再配分の実行前に、データを均等に配分する場合の再配分について検討してください。4 つのデータベース・パーティションから 5 つのデータベース・パーティションに移動しようとする場合、元の 4 つのデータベース・パーティションの約 20% が新しいデータベース・パーティションに移動します。これは、20% の DELETE 操作が元の 4 つの各データベース・パーティションで行われ、INSERT 操作のすべてが新しいデータベース・パーティションで行われることを意味します。
- 分散キーに多くの NULL 値が含まれているといった、データの配分が不均等な場合について考えてみてください。この場合、分散キーに NULL 値を含むすべての行が、以前の配分方式のデータベース・パーティションから、新規の配分方式の別のデータベース・パーティションに移動します。その結果、それら 2 つのデータベース・パーティションで必要とされるログ・スペースの量が増え、おそらく均等な配分を想定した場合の計算量を超えることになります。
- 各表の再配分は単一トランザクションです。このため、ログ・スペースを見積もるときは、20% などの変更のパーセントに最大の表のサイズを乗算します。ただし、最大の表は均等に配分されているものの、例えば 2 番目に大きい表には 1 つ以上の満杯のデータベース・パーティションが存在する場合もあることを考慮してください。そのような場合には、最大の表の代わりに、不均等な配分表を使用することを考慮してください。

注: データベース・パーティションで挿入および削除されるデータの最大量を見積もった後、その見積もりを 2 倍して、アクティブ・ログのピーク・サイズを決定します。この見積もりがアクティブ・ログの限界の 512 GB を超える場合、複数のステップでデータの再配分を行う必要があります。バージョン 9.5 フィックスパック 3 において、アクティブ・ログの限界が 1024 GB に引き上げられました。「makepmap」ユーティリティーを使用して、各ステップに 1 つずつの、一連のタ

ーゲット分散マップを生成してください。また、logsecond データベース構成パラメーターを -1 に設定して、大半のログ・スペースの問題を回避することもできます。

REDISTRIBUTE DATABASE PARTITION GROUP コマンドが使用され、NOT ROLLFORWARD RECOVERABLE オプションが使用されるか、またはデータの再配分がロールフォワード・リカバリー可能ではないその他のサポートされているインターフェースから再配分が実行される場合、以下を検討します。

- データの再配分の一部として行を移動するとき、ログ・レコードは作成されません。これにより、ログ・ファイルのスペース所要量を大幅に減らせますが、このオプションが使用されると、データベースのロールフォワード・リカバリーが実行されるときに、再配分操作のログ・レコードをロールフォワードすることができず、ロールフォワード操作の一部として処理される表は UNAVAILABLE 状態のままになります。NOT ROLLFORWARD RECOVERABLE オプションを使用した結果についての説明は、「コマンド・リファレンス」を参照してください。
- データの再配分を受けるデータベース・パーティション・グループが、ロング・フィールド (LF) またはラージ・オブジェクト (LOB) データのある表を含む場合、ログ・レコードはデータの各行ごとに作成されるので、データの再配分中に生成されるログ・レコードの数はさらに多くなります。この場合、データベース・パーティションごとのログ・スペース所要量は、そのパーティションで移動するデータ量 (送信、受信、またはその両方が実行されるデータ) のおよそ 3 分の 1 と予測します。 LF/LOB データの存在に関係なく、受信側のパーティションには、エクステント割り振りログ・レコードという一種のログ・レコードがあり、これには移動するデータの量に依存するログ・レコードの数について書き込まれます。ただし、これらのログ・レコードの合計スペース所要量は少なく、移動するユーザー・データの合計のごく一部分という程度です。

再配分イベント・ログ・ファイル

データの再配分中に、イベント・ロギングが実行されます。イベント情報は、イベント・ログ・ファイルにログとして記録され、後でエラー・リカバリーを実行するのに使用できます。

注: このトピックには、DB2 9.5 フィックスパック 1 をインストールして初めて有効になるキーワード更新について言及している箇所があります。DB2 9.5 フィックスパック 1 がまだ入手できないかインストールされていない場合には、このトピックの DB2 9 バージョンを、次の場所にある DB2 9 インフォメーション・センターから参照してください。http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp

データの再配分が実行されるとき、処理される各表についての情報が、単一の再配分イベント・ログ・ファイルにログとして記録されます。

イベント・ログ・ファイル名は、database-name.database-partition-group-name.timestamp.log のようにフォーマット設定されます。ログ・ファイルは、以下のように配置されます。

- Linux® および UNIX® ベースのシステム上では homeinst/sqllib/redist ディレクトリー。

- Windows® オペレーティング・システム上では DB2INSTPROF¥instance¥redist ディレクトリー。ここで DB2INSTPROF は DB2INSTPROF レジストリー変数の値です。

イベント・ログ・ファイル名の例を以下に示します。

```
DB819.NG1.2007062419415651.log
```

このイベント・ログ・ファイルは、現地時間で 2007 年 6 月 24 日の午後 7 時 41 分に作成された NG1 という名前のデータベース・パーティション・グループを持つ、DB819 という名前のデータベース上での再配分操作に関するものです。

イベント・ログ・ファイルの 3 つの主な用法は、以下のとおりです。

- 新旧の分散マップなどの、再分散操作に関する一般情報を提供する。
- ユーティリティーによりその時点までで再配分されている表の追跡に役立つ情報をユーザーに提供する。
- 再配分されている各表についての情報を提供する。これには表に使用されている索引モード、表が正常に再配分されているかどうかの表示、および表に対する再配分操作の開始および終了時刻が含まれます。

再配分のログ・ファイル項目についての詳細、およびデータの再配分中にエラーをリカバリーする方法については、以下を参照してください。

STEPWISE_REDISTRIBUTE_DBPG プロシージャーを使用したデータベース・パーティション・グループの再配分

データの再配分は、STEPWISE_REDISTRIBUTE_DBPG システム定義プロシージャーを使用して実行することができます。

データベース・パーティション・グループの再配分は、STEPWISE_REDISTRIBUTE_DBPG システム定義プロシージャーおよびその他のシステム定義プロシージャーを使用して行うことができます。

以下のステップは、何が行われるべきかの概要を説明しています。これらのステップの実行例がその後に続きます。

1. *ANALYZE_LOG_SPACE プロシージャー - ログ・スペース分析情報の検索* を使用してログ・スペースの可用性およびデータ・スキューに関してデータベース・パーティション・グループを分析します。

   analyze_log_space 関数は、ログ・スペース分析の結果セット (オープン・カーソル) を返します。この中には、指定されたデータベース・パーティション・グループの各データベース・パーティションのフィールドが含まれます。

2. *GENERATE_DISTFILE プロシージャー - データ配分ファイルの生成* を使用して指定した表のデータ配分ファイルを作成します。

   generate_distfile 関数は、指定された表のデータ配分ファイルを生成し、提供されたファイル名を使用して、それを保管します。

3. *STEPWISE_REDISTRIBUTE_DBPG プロシージャー - データベース・パーティション・グループの一部の再配分* を使用してデータベース・パーティション・グループのステップ単位再配分プランの内容を作成し、報告します。

4. *GET_SWRD_SETTINGS プロシージャー - 再配分情報の検索* と *SET_SWRD_SETTINGS プロシージャー - 再配分レジストリーの作成または変更* を使用して指定した表のデータ配分ファイルを作成します。

   get_swrd_settings 関数は、指定したデータベース・パーティション・グループの既存の再配分レジストリー・レコードを読み取ります。

   set_swrd_settings 関数は、再配分レジストリーを作成または変更します。レジストリーが存在しない場合は作成され、レコードが追加されます。レジストリーがすでに存在する場合は、overwriteSpec を使用して上書きされる必要のあるフィールド値を識別します。 overwriteSpec フィールドはこの関数を使用可能にして、更新される必要のないフィールドに NULL 入力を指定します。

5. *STEPWISE_REDISTRIBUTE_DBPG プロシージャー - データベース・パーティション・グループの一部の再配分* を使用してプランに従って、データベース・パーティション・グループを再配分します。

   stepwise_redistribute_dbpg 関数は、入力および設定ファイルに従って、データベース・パーティション・グループの一部を再配分します。

使用例

以下に示すのは、AIX での CLP スクリプトの例です。

```
# ------------------------------------------------------------------------------------
# Set the database you wish to connect to
# ------------------------------------------------------------------------------------
dbName="SAMPLE"

# ------------------------------------------------------------------------------------
# Set the target database partition group name
# ------------------------------------------------------------------------------------
dbpgName="IBMDEFAULTGROUP"

# ------------------------------------------------------------------------------------
# Specify the table name and schema
# ------------------------------------------------------------------------------------
tbSchema="$USER"
tbName="STAFF"

# ------------------------------------------------------------------------------------
# Specify the name of the data distribution file
# ------------------------------------------------------------------------------------
distFile="$HOME/sqllib/function/$dbName.IBMDEFAULTGROUP_swrdData.dst"

export DB2INSTANCE=$USER
export DB2COMM=TCPIP

# ------------------------------------------------------------------------------------
# Invoke call statements in clp
# ------------------------------------------------------------------------------------
db2startdb2 -v "connect to $dbName"

# ------------------------------------------------------------------------------------
# Analysing the effect of adding a database partition without applying the changes - a 'what if'
# hypothetical analysis
#
# - In the following case, the hypothesis is adding database partition 40, 50 and 60 to the
#   database partition group, and for database partitions 10,20,30,40,50,60, using a respective
#   target ratio of 1:2:1:2:1:2.
#
# NOTE: in this example only partitions 10, 20 and 30 actually exist in the database
```

```
#       partition group
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'A', '40,50,60', '10,20,30,40,50,60', '1,2,1,2,1,2')"

# ------------------------------------------------------------------------------------
# Analysing the effect of dropping a database partition without applying the changes
#
# - In the following case, the hypothesis is dropping database partition 30 from the database
#   partition group, and redistributing the data in database partitions 10 and 20 using a
#   respective target ratio of 1 : 1
#
# NOTE: In this example all database partitions 10, 20 and 30 should exist in the database
#       partition group
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'D', '30', '10,20','1,1')"

# ------------------------------------------------------------------------------------
# Generate a data distribution file to be used by the redistribute process
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.generate_distfile('$tbSchema', '$tbName', '$distFile')"

# ------------------------------------------------------------------------------------
# Write a step wise redistribution plan into a registry
#
# Setting the 10th parameter to 1, may cause a currently running step wise redistribute
# stored procedure to complete the current step and stop, until this parameter is reset
# to 0, and the redistribute stored procedure is called again.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.set_swrd_settings('$dbpgName', 255, 0, ' ', '$distFile', 1000,
12, 2, 1, 0, '10,20,30', '50,50,50')"

# ------------------------------------------------------------------------------------
# Report the content of the step wise redistribution plan for the given database
# partition group.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.get_swrd_settings('$dbpgName', 255, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"

# ------------------------------------------------------------------------------------
# Redistribute the database partition group "dbpgName" according to the redistribution
# plan stored in the registry by set_swrd_settings. It starting with step 3 and
# redistributes the data until 2 steps in the redistribution plan are completed.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.stepwise_redistribute_dbpg('$dbpgName', 3, 2)"
```

第 27 章 セルフチューニング・メモリーの構成

パーティション・データベース環境での自己チューニング・メモリー

パーティション・データベース環境で自己チューニング・メモリー機能を使用する場合、この機能がシステムを適切に調整するかどうかを決定するいくつかの要因があります。

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

この単一チューニング・パーティション・モデルでは、メモリー要件が類似しているデータベース・パーティションでのみこの機能を使用する必要があります。以下に示すのは、パーティション・データベースで自己チューニング・メモリーを使用可能にするかどうかを判別する際に使用するガイドラインです。

パーティション・データベースで自己チューニングが推奨される場合

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

- すべてのデータベース・パーティションが同一のハードウェア上に存在し、複数の論理ノードが複数の物理ノードに均一に分散されている
- データが完璧かほぼ完璧に分散されている
- データベース・パーティション上で実行されるワークロードがデータベース・パーティション間で均一に分散されている。つまり、1 つ以上のヒープについてメモリー要件が高くなるデータベース・パーティションは存在しません。

そのような環境では、すべてのデータベース・パーティションを同等に構成することが望ましいと言えます。このとき、自己チューニング・メモリーはシステムを適切に構成します。

パーティション・データベースで注意しながらの自己チューニングが推奨される場合

環境内のほとんどのデータベース・パーティションでメモリー要件が類似していて、それらのパーティションが類似のハードウェア上で稼働している場合は、初期構成にいくらかの注意を払うかぎり、自己チューニング・メモリーを使用することができます。これらのシステムには、データ用のデータベース・パーティションのセットと、ずっと少ない数のコーディネーター・パーティションのセット、およびカタログ・パーティションが存在する場合があります。このような環境では、コー

ディネーター・パーティションとカタログ・パーティションを、データを含むデータベース・パーティションとは異なった構成にすると便利です。

この環境では、いくらかの小さいセットアップを行うことにより、引き続き自己チューニング・メモリー機能から益を得ることができます。データを含むデータベース・パーティションによってデータベース・パーティションの大部分が構成されるため、これらのデータベース・パーティションのすべてで自己チューニングを使用可能にし、これらのデータベース・パーティションの 1 つをチューニング・パーティションとして指定する必要があります。加えて、カタログ・パーティションとコーディネーター・パーティションが異なる構成になっている可能性があるため、自己チューニング・メモリーをこれらのパーティションで使用不可にする必要があります。カタログおよびコーディネーター・パーティションで自己チューニングを使用不可にするには、これらのパーティションで *self_tuning_mem* データベース構成パラメーターを OFF に更新してください。

パーティション・データベースで自己チューニングが推奨されない場合

各データベース・パーティションのメモリー要件が異なっている環境や、さまざまなデータベース・パーティションが大幅に異なるハードウェア上で稼働している場合では、自己チューニング・メモリー機能を使用不可にすることをお勧めします。これを行うには、すべてのパーティションで *self_tuning_mem* データベース構成パラメーターを OFF にします。

異なるデータベース・パーティションのメモリー要件の比較

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

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

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

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

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

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

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

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

パーティション・データベース環境でのセルフチューニング・メモリーの使用

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

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

- チューニング・パーティションとして現在指定されているデータベース・パーティションを判別するには、以下の ADMIN_CMD を使用します。

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

- チューニング・パーティションを変更するには、以下の ADMIN_CMD を使用します。

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

  このコマンドを発行すると、チューニング・パーティションは、非同期で更新されるか次回のデータベース始動時に更新されます。

- メモリー・チューナーが自動的にチューニング・パーティションを再選択するようにするには、<db_partition_num> 値に -1 を入力します。

DPF システムでのメモリー・チューナーの開始

メモリー・チューナーが DPF 環境において開始されるのは、データベースが明示的な ACTIVATE DATABASE コマンドによってアクティブ化された場合のみです。セルフチューニングでは、複数パーティション・システムでメモリーを正しく調整する前にすべてのパーティションがアクティブであることが必要だからです。

指定されたデータベース・パーティションでセルフチューニングを無効にする

- データベース・パーティションのサブセットでセルフチューニングを無効にするには、調整しないデータベース・パーティションに対して *self_tuning_mem* 構成パラメーターを OFF に設定します。
-

  特定のデータベース・パーティションの構成パラメーターによって制御された、メモリー・コンシューマーのサブセットに対してセルフチューニングを無効にす

るには、関連する構成パラメーターの値またはバッファー・プール・サイズの値を MANUAL またはそのデータベース・パーティションの特定の値に設定します。ただし、セルフチューニングの構成パラメーターの値は、すべての稼働パーティション間で整合させることをお勧めします。

- データベース・パーティションの特定のバッファー・プールで調整を無効にするには、ALTER BUFFER POOL コマンドを発行し、セルフチューニングを無効にするパーティションについて、サイズ値および PARTITIONNUM パラメーターの値を指定します。

  PARTITIONNUM 節を使用する特定データベース・パーティション上で、サイズを指定するALTER BUFFERPOOL ステートメントは、与えられたバッファー・プールに関する例外項目を SYSCAT.SYSBUFFERPOOLNODES カタログに作成します。あるいは例外項目が既に存在する場合は、これを更新します。このカタログにバッファー・プールの例外項目が存在する場合に、デフォルトのバッファー・プール・サイズが AUTOMATIC に設定されていると、そのバッファー・プールはセルフチューニングに関与しません。例外項目を除去して、バッファー・プールをセルフチューニング用に再び有効にするには、以下のようにします。

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

非均一環境でメモリーのセルフチューニングを有効にする

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

第 28 章 DB2 構成パラメーターおよび変数

複数パーティション間でのデータベースの構成

データベース・マネージャーでは、複数のパーティション間のすべてのデータベース構成エレメントの単一ビューを提供します。これは、各データベース・パーティションに対して db2_all コマンドを呼び出さなくても、すべてのデータベース・パーティション間のデータベース構成を更新またはリセットできることを意味します。

データベースが存在するパーティションから 1 つの SQL ステートメントを発行するか、1 つの管理コマンドを発行するだけで、複数パーティションのデータベース構成を更新できます。データベース構成を更新またはリセットする方法は、デフォルトではすべてのデータベース・パーティションに対して です。

コマンド・スクリプトおよびアプリケーションの後方互換性については、次の 3 つのオプションがあります。

- db2set コマンドを使用して、次のように **DB2_UPDDBCFG_SINGLE_DBPARTITION** レジストリー変数を TRUE に設定します。

  ```
  DB2_UPDDBCFG_SINGLE_DBPARTITION=TRUE
  ```

  注: レジストリー変数の設定は、ADMIN_CMD プロシージャーを使用して行われる UPDATE DATABASE CONFIGURATION または RESET DATABASE CONFIGURATION 要求には適用されません。

- UPDATE DATABASE CONFIGURATION または RESET DATABASE CONFIGURATION コマンド、あるいは ADMIN_CMD プロシージャーのいずれかで **DBPARTITIONNUM** パラメーターを使用します。例えば、すべてのデータベース・パーティション上のデータベース構成を更新するには、次のように ADMIN_CMD プロシージャーを呼び出します。

  ```
  CALL SYSPROC.ADMIN_CMD
    ('UPDATE DB CFG USING sortheap 1000')
  ```

  単一のデータベース・パーティションを更新するには、次のように ADMIN_CMD プロシージャーを呼び出します。

  ```
  CALL SYSPROC.ADMIN_CMD
    ('UPDATE DB CFG DBPARTITIONNUM 10 USING sortheap 1000')
  ```

- db2CfgSet API で **DBPARTITIONNUM** パラメーターを使用します。 **db2Cfg** 構造内の各フラグによって、データベース構成の値を単一のデータベース・パーティションに適用するかどうかが指示されます。フラグを設定する場合、**DBPARTITIONNUM** 値も指定する必要があり、例えば次のようにします。

  ```
  #define db2CfgSingleDbpartition          256
  ```

  データベース・マネージャーまたはデータベース構成パラメーターを設定する db2CfgSet API に対して、db2CfgSingleDbpartition 値を設定していない場合、**DB2_UPDDBCFG_SINGLE_DBPARTITION** レジストリー変数を TRUE に設定

していないか、または *versionNumber* をバージョン 9.5 のバージョン番号より小さいものに設定していなければ、そのデータベース構成の値はすべてのデータベース・パーティションに適用されます。

データベースをバージョン 9.5 にマイグレーションする場合、既存のデータベース構成パラメーターは、一般的な規則として、マイグレーション後もそのままの値を保持します。ただし、新規のパラメーターがデフォルト値を使用して追加され、既存のパラメーターのいくつかにも、バージョン 9.5 の新規のデフォルト値が設定されます。既存のデータベース構成パラメーターに対する変更について詳しくは、「*マイグレーション・ガイド*」のトピック『DB2 サーバー動作の変更点』を参照してください。マイグレーション後のデータベースに対して、データベース構成の更新またはリセットを要求する場合はすべて、構成の更新またはリセットの要求に関するバージョン 9.5 の方式が使用されます。

既存の更新またはリセットのコマンド・スクリプトの場合、上で説明したのと同じ規則が適用されます。つまり、バージョン 9.5 より前の方法を使用するか、UPDATE DATABASE CONFIGURATION または RESET DATABASE CONFIGURATION コマンドの **DBPARTITIONNUM** オプションを組み込むようにスクリプトを変更するか、あるいは **DB2_UPDDBCFG_SINGLE_DBPARTITION** レジストリー変数を設定することができます。

db2CfgSet API を呼び出す既存のアプリケーションの場合、バージョン 9.5 の方法を使用する必要があります。バージョン 9.5 より前の方法を使用する場合、**DB2_UPDDBCFG_SINGLE_DBPARTITION** レジストリー変数を設定できます。あるいは、新規の db2CfgSingleDbpartition フラグ、および特定のデータベース・パーティションのデータベース構成を更新またはリセットするための新規の **dbpartitionnum** フィールドを組み込み、バージョン 9.5 のバージョン番号を指定してこの API を呼び出すようにアプリケーションを変更することもできます。

注: データベース構成値に整合性がないことがわかる場合、それぞれのデータベース・パーティションを個別に更新またはリセットできます。

パーティション・データベース環境変数

DB2CHGPWD_EEE

- オペレーティング・システム: DB2ESE on AIX、Linux、およびWindows
- デフォルト = NULL。値: YES または NO
- この変数は、AIX または Windows ESE システムでのパスワード変更を他のユーザーに許可するかどうかを指定します。すべてのデータベース・パーティションまたはノードのパスワードは、 Windows ドメイン・コントローラ (Windows の場合) または LDAP (AIX の場合) を使って集中保守する必要があります。集中保守しないと、すべてのデータベース・パーティションまたはノード間でパスワードが一致しなくなるおそれがあります。その結果、ユーザーが変更を加えるために接続するデータベース・パーティションでのみパスワードが変更される可能性があります。

DB2_FCM_SETTINGS

- オペレーティング・システム: Linux

- Default=NULL, Values:
  FCM_MAXIMIZE_SET_SIZE:[YES|TRUE|NO|FALSE].
  FCM_MAXIMIZE_SET_SIZE のデフォルト値は NO です。
- バージョン 9.5 フィックスパック 4 から、**DB2_FCM_SETTINGS** レジストリー変数を FCM_MAXIMIZE_SET_SIZE トークンと共に設定して、高速コミュニケーション・マネージャー (FCM) バッファー用にデフォルトの 2 GB スペースを事前割り振りできるようになりました。このフィーチャーを使用可能にするには、このトークンの値が YES または TRUE のいずれかである必要があります。

DB2_NUM_FAILOVER_NODES

- オペレーティング・システム: すべて
- デフォルト = 2。値: 0 からデータベース・パーティションの必要数まで
- 特定のデータベースが存在する各マシン上で、**DB2_NUM_FAILOVER_NODES** を設定して、そのデータベース内のデータベース・パーティションの総数を指定します。

  DB2 データベースの高可用性ソリューションでは、データベース・サーバーで障害が発生した際に、障害が発生したマシンのデータベース・パーティションを別のマシンで再始動できます。高速コミュニケーション・マネージャー (FCM) は、**DB2_NUM_FAILOVER_NODES** を使用して、このフェイルオーバーを機能させるために各マシンでどれほどのメモリーを予約しておくかを計算します。

  例えば、次の構成を考えてみましょう。
  - マシン A には 1 と 2 という 2 つのデータベース・パーティションがあります。
  - マシン B には 3 と 4 という 2 つのデータベース・パーティションがあります。
  - A と B 両方のマシンで **DB2_NUM_FAILOVER_NODES** は 4 に設定されています。

  DB2START の際、FCM は、A と B の両方に最大 4 つのデータベース・パーティションを管理できるだけの十分なメモリーを予約して、一方のマシンで障害が発生しても、もう一方のマシンで障害が発生したマシンの 2 つのデータベース・パーティションを再始動できるようにしておきます。マシン A で障害が発生した場合は、マシン B でデータベース・パーティション 1 と 2 を再始動できます。マシン B で障害が発生した場合は、マシン A でデータベース・パーティション 3 と 4 を再始動できます。

DB2_PARTITIONEDLOAD_DEFAULT

- オペレーティング・システム: サポートされる ESE プラットフォームのすべて
- デフォルト = YES。値: YES または NO
- **DB2_PARTITIONEDLOAD_DEFAULT** レジストリー変数によって、ESE に特定のロード・オプションを指定しない場合、ユーザーは ESE 環境内のロード・ユーティリティーのデフォルト動作を変更できます。デフ

ォルト値は YES であり、これは ESE 環境で ESE に特定のロード・オプションを指定しない場合に、ロードがターゲット表が定義されているすべてのデータベース・パーティション上で試行されることを指定します。値が NO であると、ロードはロード・ユーティリティーが現在接続されているデータベース・パーティション上だけで試行されます。

注: この変数は、推奨されておらず、今後のリリースでは除去される可能性があります。 LOAD コマンドに、同じ動作を実現するために使用できるさまざまなオプションがあります。次のコマンドを LOAD コマンドとともに指定することにより、この変数に NO を設定した場合と同じ結果を得ることができます。`PARTITIONED DB CONFIG MODE LOAD_ONLY OUTPUT_DBPARTNUMS x`。ここで、x はデータをロードするパーティションのパーティション番号です。

DB2PORTRANGE

- オペレーティング・システム: Windows
- 値: nnnn:nnnn
- この値は、FCM によって使用される TCP/IP ポート範囲に設定されるので、別のマシン上に作成される追加のデータベース・パーティションも同じポート範囲になります。

パーティション・データベース環境の構成パラメーター

通信

conn_elapse - 接続経過時間

このパラメーターは、TCP/IP 接続が 2 つのデータベース・パーティション・サーバー間に設定される必要がある時間 (秒数) を指定します。

構成タイプ
データベース・マネージャー

適用 ローカルとリモート・クライアントを持つパーティション・データベース・サーバー

パラメーター・タイプ
オンラインで構成可能

伝搬クラス
即時

デフォルト [範囲]
10 [0–100]

単位 秒

このパラメーターによって指定された時間内に接続の試みが成功すれば、通信が確立されます。接続に失敗した場合は、通信を確立する試みがもう一度行われます。*max_connretries* パラメーターによって指定された回数だけ接続が試みられ、しかもそのすべてがタイムアウトになった場合は、エラーになります。

fcm_num_buffers - FCM バッファー数

このパラメーターは、データベース・サーバー間とデータベース・サーバー内の両方での内部通信 (メッセージ) 用として使用される 4KB バッファーの数を指定します。

構成タイプ
データベース・マネージャー

適用

- ローカルおよびリモート・クライアントを持つデータベース・サーバー
- ローカル・クライアントを持つデータベース・サーバー
- ローカルとリモート・クライアントを持つパーティション・データベース・サーバー

パラメーター・タイプ
オンラインで構成可能

伝搬クラス
即時

デフォルト [範囲]

32 ビット・プラットフォーム
Automatic [128 - 65 300]

64 ビット・プラットフォーム
Automatic [128 - 524 288]

- ローカルとリモート・クライアントを持つデータベース・サーバー。デフォルトは、1024。
- ローカル・クライアントを持つデータベース・サーバー。デフォルトは、512。
- ローカルとリモート・クライアントを持つパーティション・データベース・サーバー。デフォルトは、4096。

単一パーティション・データベース・システムでは、*intra_parallel* パラメーターをデフォルト値の NO から YES に変更してパーティション内並列処理を有効にした場合に限り、このパラメーターが使用されます。

初期値と AUTOMATIC 属性の両方を設定することが可能です。

AUTOMATIC に設定されている場合、FCM はリソースの使用状況をモニターして、リソースを増やしたり、30 分間使用されていないリソースがあればリソースを減らしたりすることができます。増減可能なリソースの量はプラットフォームによって異なります。特に Linux の場合は、開始値の 25 % までしか増加させることができません。インスタンスの開始時に指定されたリソース数をデータベース・マネージャーが割り振れない場合には、インスタンスを開始できるようになるまで、構成値を徐々に減らします。

同じマシン上に複数の論理ノードがある場合は、このパラメーターの値を大きくする必要がある場合もあります。また、システム上のユーザーの数、システム上のデ

ータベース・パーティション・サーバーの数、または複雑なアプリケーションのためにメッセージ・バッファーを使い尽くした場合は、このパラメーターの値を大きくしなければならない場合があります。

複数の論理ノードを使用している場合は、*fcm_num_buffers* バッファーの 1 つのプールが同じマシン上の複数の論理ノードすべてで共有されます。プールのサイズは、*fcm_num_buffers* 値にその物理マシン上の論理ノードの数を乗算して決定します。使用中の値をもう一度調べて、複数の論理ノードがあるマシン上で割り振られる FCM バッファーの合計数を考慮してください。

fcm_num_channels - FCM チャネル数

このパラメーターは、各データベース・パーティションの FCM チャネル数を指定します。

構成タイプ

データベース・マネージャー

適用

- ローカルおよびリモート・クライアントを持つデータベース・サーバー
- ローカル・クライアントを持つデータベース・サーバー
- ローカルとリモート・クライアントを持つパーティション・データベース・サーバー
- ローカル・クライアントを持つサテライト・データベース・サーバー

パラメーター・タイプ

オンラインで構成可能

伝搬クラス

即時

デフォルト [範囲]

UNIX 32 ビット・プラットフォーム

自動。開始値は 256、512、2 048 [128 - 120 000]

UNIX 64 ビット・プラットフォーム

自動。開始値は 256、512、2 048 [128 - 524 288]

Windows 32 ビット

自動。開始値は 10 000 [128 - 120 000]

Windows 64 ビット

自動。開始値は 256、512、2 048 [128 - 524 288]

- ローカルとリモート・クライアントを持つデータベース・サーバーの場合、開始値は 512 です。
- ローカル・クライアントを持つデータベース・サーバーの場合、開始値は 256 です。
- ローカルとリモート・クライアントを持つパーティション・データベース環境サーバーの場合、開始値は 2 048 です。

非パーティションのデータベース環境では、*intra_parallel* パラメーターがアクティブでないと、*fcm_num_channels* を使用できません。

FCM チャネルは、DB2 エンジンで実行される EDU 間の論理通信エンドポイントです。両方の制御フロー (要求と応答) とデータ・フロー (表キュー・データ) はチャネルにより、パーティション間のデータを転送します。

AUTOMATIC に設定されている場合、FCM はチャネルの使用状況をモニターし、要件の変化に応じて徐々にリソースの割り振りや解放を行います。

max_connretries - ノード接続再試行回数

このパラメーターは、2 つのデータベース・パーティション・サーバー間で TCP/IP 接続の確立を試行する最大回数を指定します。

構成タイプ
データベース・マネージャー

適用 ローカルとリモート・クライアントを持つパーティション・データベース・サーバー

パラメーター・タイプ
オンラインで構成可能

伝搬クラス
即時

デフォルト [範囲]
5 [0–100]

2 つのデータベース・パーティション・サーバー間に通信を確立する試みが失敗した (例えば、*conn_elapse* パラメーターによって指定された値に達した) 場合は、*max_connretries* で、データベース・パーティション・サーバーに対して実行できる接続再試行回数を指定します。このパラメーターに指定された値を超えた場合は、エラーが戻されます。

max_time_diff - ノード間最大時差

このパラメーターは、ノード構成ファイルにリストされているデータベース・パーティション・サーバー間に許容されている最大時差を分数で指定します。

構成タイプ
データベース・マネージャー

適用 ローカルとリモート・クライアントを持つパーティション・データベース・サーバー

パラメーター・タイプ
構成可能

デフォルト [範囲]
60 [1 - 1 440]

単位 分

各データベース・パーティション・サーバーには、それぞれ固有のシステム・クロックがあります。複数のデータベース・パーティション・サーバーが 1 つのトランザクションに関連付けられていて、それらのクロックがこのパラメーターで指定されている時間内で同期していないと、そのトランザクションはリジェクトされ、

SQLCODE が戻されます。 (トランザクションがリジェクトされるのは、データ変更がトランザクションに関連している場合だけです)。

DB2 では *協定世界時* (UTC) を使用しているので、このパラメーターを設定すると、異なる時間帯は考慮事項になりません。協定世界時とは、グリニッジ標準時と同じものです。

start_stop_time - タイムアウトの開始および停止

このパラメーターは分単位で指定します。この範囲内にすべてのデータベース・パーティション・サーバーが DB2START または DB2STOP コマンドに応答する必要があります。また、ADD DBPARTITIONNUM 操作時にタイムアウト値としても使用されます。

構成タイプ
データベース・マネージャー

適用 ローカルおよびリモート・クライアントを持つデータベース・サーバー

パラメーター・タイプ
オンラインで構成可能

伝搬クラス
即時

デフォルト [範囲]
10 [1 - 1 440]

単位 分

指定された時間内に DB2START コマンドに応答しないデータベース・パーティション・サーバーは、インスタンスのホーム・ディレクトリーの `sqllib` サブディレクトリーの `log` サブディレクトリーにある `db2start` エラー・ログにメッセージを送信します。それらを再始動する前に、これらのノードで DB2STOP を出す必要があります。

指定された時間内に DB2STOP コマンドに応答しないデータベース・パーティション・サーバーは、インスタンスのホーム・ディレクトリーの `sqllib` サブディレクトリーの `log` サブディレクトリーにある `db2stop` エラー・ログにメッセージを送信します。応答しないそれぞれのデータベース・パーティション・サーバーごと、またはそのすべてに対して、`db2stop` を出すことができます。 (すでに停止しているサーバーは、停止していることを示す記述を返します。)

複数パーティション・データベースにおける `db2start` または `db2stop` 操作が、*start_stop_time* データベース・マネージャー構成パラメーターにより指定された値以内で完了しなかった場合、タイムアウトとなったデータベース・パーティションは内部で強制終了します。 *start_stop_time* の値が低いデータベース・パーティションが多数ある環境では、この動作が発生する可能性があります。この動作による問題を解決するには、*start_stop_time* にもっと大きい値を指定してください。

DB2START、START DATABASE MANAGER、ADD DBPARTITIONNUM のいずれかのコマンドを使用して新規データベース・パーティションを追加する場合、データベース・パーティション追加操作で、インスタンス内の各データベースの自動ストレージが使用可能かどうかを判別する必要があります。これは、各データベース

のカタログ・パーティションと通信することで行います。自動ストレージが使用可能な場合、ストレージ・パスの定義はその通信の中で取得されます。同様に、データベース・パーティションを使用して SYSTEM TEMPORARY 表スペースを作成する操作でも、別のデータベース・パーティション・サーバーと通信して、そのサーバーにあるデータベース・パーティションの表スペース定義を取得する必要がある場合があります。これらの要素は、*start_stop_time* パラメーターの値を決定するときに考慮する必要があります。

並列処理

intra_parallel - パーティション内並列処理機能の使用可能化

このパラメーターは、データベース・マネージャーでパーティション内並列処理を使用できるかどうかを指定します。

構成タイプ

データベース・マネージャー

適用

- ローカルおよびリモート・クライアントを持つデータベース・サーバー
- ローカル・クライアントを持つデータベース・サーバー
- ローカルとリモート・クライアントを持つパーティション・データベース・サーバー

パラメーター・タイプ

構成可能

デフォルト [範囲]

NO (0) [SYSTEM (-1), NO (0), YES (1)]

値が -1 の場合、データベース・マネージャーが動作しているハードウェアに基づき、パラメーター値を「YES」または「NO」に設定します。

このパラメーターが「YES」の場合、並列操作によってパフォーマンスを改善できる操作の中には、データベース照会と索引作成が含まれます。

注:

- 並列索引作成では、この構成パラメーターは使用されません。
- このパラメーター値を変更すると、パッケージがデータベースに再バインドされることがあり、パフォーマンスが低下する場合もあります。

max_querydegree - 照会の最大並列処理多重度

このパラメーターは、データベース・マネージャーのこのインスタンスで実行中の SQL ステートメントで使用される、パーティション内並列処理の最大多重度を指定します。 SQL ステートメントの実行中に、そのステートメントによってデータベース・パーティション内でこの数より多くの並列操作が行われることはありません。

構成タイプ

データベース・マネージャー

適用

- ローカルおよびリモート・クライアントを持つデータベース・サーバー

- ローカル・クライアントを持つデータベース・サーバー
- ローカルとリモート・クライアントを持つパーティション・データベース・サーバー

パラメーター・タイプ
オンラインで構成可能

伝搬クラス
ステートメント境界

デフォルト [範囲]
-1 (ANY) [ANY, 1 - 32 767] (ANY はシステム判別を指す)

データベース・パーティションでSQL ステートメントのパーティション内並列処理を使用できるようにするには、*intra_parallel* 構成パラメーターを「YES」に設定する必要があります。 *intra_parallel* パラメーターは、並列索引作成には必要なくなりました。

この構成パラメーターのデフォルト値は -1 です。この値は、オプティマイザーによって決められた並列処理の度合いがシステムで使用されることを意味します。それ以外の場合は、ユーザー指定の値が使用されます。

注: SQL ステートメントの並列処理の多重度は、CURRENT DEGREE 特殊レジスターまたは DEGREE BIND オプションを使用して、ステートメントのコンパイル時に指定することができます。

アクティブなアプリケーションの照会の最大並列処理多重度は、SET RUNTIME DEGREE コマンドを使用して変更することができます。実際に実行時に使用される多重度は、次の値より低くなります。

- *max_querydegree* 構成パラメーター
- アプリケーション実行時の多重度
- SQL ステートメント・コンパイル時の多重度

この構成パラメーターは、照会にのみ適用されます。

第 5 部 管理 API、コマンド、SQL ステートメント

第 29 章 管理 API

sqleaddn - パーティション・データベース環境へのデータベース・パーティション・サーバーの追加

新しいデータベース・パーティション・サーバーをパーティション・データベース環境に追加します。この API は、インスタンスで現在定義されているデータベースをすべて対象にして、新規データベース・パーティション・サーバー上にデータベース・パーティションを作成します。ユーザーは、任意の SYSTEM TEMPORARY 表スペース用のソース・データベース・パーティション・サーバーをデータベースとともに作成するか、または SYSTEM TEMPORARY 表スペースを作成しないかを指定することができます。この API は追加するデータベース・パーティション・サーバーから発行する必要があり、データベース・パーティション・サーバー上でのみ発行可能です。

有効範囲

この API は、それが実行されるデータベース・パーティション・サーバーにのみ影響を与えます。

許可

以下のいずれか。

- sysadm
- sysctrl

必要な接続

なし

API 組み込みファイル

```
sqlenv.h
```

API とデータ構造構文

```
SQL_API_RC SQL_API_FN
  sqleaddn (
        void * pAddNodeOptions,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgaddn (
        unsigned short addnOptionsLen,
        struct sqlca * pSqlca,
        void * pAddNodeOptions);
```

sqleaddn API パラメーター

pAddNodeOptions
入力。オプション sqle_addn_options 構造を指すポインター。 SYSTEM TEMPORARY 表スペース定義がある場合は、この構造によってそのソー

ス・データベース・パーティション・サーバーを指定します。この定義はノードの追加操作中に作成されたすべてのデータベース・パーティションに関するものです。何も指定されていない場合 (つまり、NULL ポインターが指定されている場合)、 SYSTEM TEMPORARY 表スペース定義はカタログ・パーティションの定義と同じになります。

pSqlca 出力。sqlca 構造を指すポインター。

sqlgaddn API 固有パラメーター

addnOptionsLen
入力。オプション sqle_addn_options 構造の長さを示す 2 バイトの符号なし整数 (バイト単位) です。

使用上の注意

新規のデータベース・パーティション・サーバーを追加する前に、十分なストレージを確保しておいてください。このストレージには、システム上に存在するすべてのデータベース用のコンテナーを作成する必要があります。

ノードの追加操作によって、新規データベース・パーティション・サーバー上に、インスタンス内にあるすべてのデータベース用の空のデータベース・パーティションが作成されます。新規のデータベース・パーティション用の構成パラメーターはデフォルト値に設定されます。

データベース・パーティションをローカルに作成している間にノードの追加操作が失敗すると、クリーンアップ段階に入ります。この処理では、作成されたデータベースがすべてローカルでドロップされます。これは、追加されたデータベース・パーティション・サーバー (つまり、ローカル・データベース・パーティション・サーバー) だけからデータベース・パーティションが除去されるということです。その他のデータベース・パーティション・サーバー上に存在しているデータベース・パーティションは影響を受けません。これが失敗した場合、クリーンアップは終了し、エラーが戻されます。

ALTER DATABASE PARTITION GROUP ステートメントを使用して、データベース・パーティション・サーバーを、データベース・パーティション・グループに追加してからでなければ、新規のデータベース・パーティション・サーバー上のデータベース・パーティションに、ユーザー・データを含めることはできません。

データベース作成操作またはデータベースのドロップ操作が進行中の場合、この API は正常に実行されません。操作が一度完了してしまうと、この API をもう一度呼び出すことができます。

システム中のデータベースで、XML 列を持つユーザー表の作成が行われた場合や、XSR オブジェクトの登録が行われた場合、それが成功してもしなくても、この API は必ず失敗します。

データベースの自動ストレージが有効になっているかどうかを判別するために、sqleaddn API は、インスタンス中の各データベースについて、カタログ・パーティションと通信を行う必要があります。自動ストレージが有効になっていれば、その通信によってストレージ・パス定義が得られます。同様に、データベース・パーティションとともに SYSTEM TEMPORARY 表スペースが作成される場合、表スペー

ス定義を検索するために、sqleaddn API はパーティション・データベース環境で他のデータベース・パーティション・サーバーと通信することが必要になる場合があります。 start_stop_time データベース・マネージャー構成パラメーターを使用して、時間 (分) を指定します。他のデータベース・パーティション・サーバーはこの時間内で自動ストレージおよび表スペース定義に応答する必要があります。時間を超過すると、API は失敗します。 start_stop_time の値を大きくして、API をもう一度呼び出してください。

REXX API 構文

この API は、SQLDB2 インターフェースを使って、REXX から呼び出すことができます。

sqlecran - データベース・パーティション・サーバー上へのデータベース作成

API を呼び出すデータベース・パーティション・サーバー上だけでデータベースを作成します。この API は汎用目的ではありません。例えば、あるデータベース・パーティション・サーバーのデータベース・パーティションが損傷して再作成が必要な場合に、 db2Restore とともに使用する必要があります。この API を不適切な仕方で使用すると、システム内に不整合が生じるので、注意して使用してください。

注: この API を (損傷したという理由で) ドロップされたデータベース・パーティションを再作成するために使用した場合、このデータベース・パーティション・サーバーのデータベースはリストア・ペンディング状態になります。データベース・パーティションを再作成したら、ただちにデータベースをこのデータベース・パーティション・サーバー上にリストアする必要があります。

有効範囲

この API は、それが呼び出されたデータベース・パーティション・サーバーにのみ影響を与えます。

許可

以下のいずれか。

- sysadm
- sysctrl

必要な接続

インスタンス。データベースを別のデータベース・パーティション・サーバーで作成する場合、まずそのデータベース・パーティション・サーバーにアタッチすることが必要になります。データベース接続は、この API によって処理中に一時的に確立されます。

API 組み込みファイル

```
sqlenv.h
```

API とデータ構造構文

```
SQL_API_RC SQL_API_FN
  sqlecran (
        char * pDbName,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgcran (
        unsigned short reservedLen,
        unsigned short dbNameLen,
        struct sqlca * pSqlca,
        void * pReserved,
        char * pDbName);
```

sqlecran API パラメーター

pDbName
入力。作成されるデータベースの名前を含むストリングを指定します。NULL にはしないでください。

pReserved
入力。 NULL に設定されたスペア・ポインター、またはゼロを指すスペア・ポインター。将来の使用のために予約されています。

pSqlca 出力。sqlca 構造を指すポインター。

sqlgcran API 固有パラメーター

reservedLen
入力。 pReserved の長さのために予約されています。

dbNameLen
入力。データベース名の長さを示す 2 バイトの符号なし整数 (バイト単位) です。

使用上の注意

データベースが正常に作成されると、リストア・ペンディング状態になります。このデータベースを使用するには、その前にデータベースをこのデータベース・パーティション・サーバー上にリストアする必要があります。

REXX API 構文

この API は、SQLDB2 インターフェースを使って、REXX から呼び出すことができます。

sqledpan - データベース・パーティション・サーバーでのデータベースのドロップ

指定されたデータベース・パーティション・サーバーでデータベースをドロップします。パーティション・データベース環境でのみ実行可能です。

有効範囲

この API は、それが呼び出されたデータベース・パーティション・サーバーにのみ影響を与えます。

許可

以下のいずれか。

- sysadm
- sysctrl

必要な接続

なし。インスタンス接続は呼び出しの期間中に確立されます。

API 組み込みファイル

```
sqlenv.h
```

API とデータ構造構文

```
SQL_API_RC SQL_API_FN
  sqledpan (
        char * pDbAlias,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdpan (
        unsigned short Reserved1,
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        void * pReserved2,
        char * pDbAlias);
```

sqledpan API パラメーター

pDbAlias
入力。ドロップされるデータベースの別名を含むストリングを指定します。これは、システム・データベース・ディレクトリーにある実際のデータベース名を参照するための名前です。

pReserved
予約済み。 NULL にする必要があります。

pSqlca 出力。sqlca 構造を指すポインター。

sqlgdpan API 固有パラメーター

Reserved1
将来の使用のために予約されています。

DbAliasLen
入力。データベース別名の長さを示す 2 バイトの符号なし整数 (バイト単位) です。

pReserved2
NULL に設定されたスペア・ポインター、またはゼロを指すスペア・ポインター。将来の使用のために予約されています。

使用上の注意

この API を不適切な仕方で使用すると、システム内に不整合が生じるので、注意して使用してください。

REXX API 構文

この API は、SQLDB2 インターフェースを使って、REXX から呼び出すことができます。

sqledrpn - データベース・パーティション・サーバーがドロップ可能かどうかの検査

データベース・パーティション・サーバーがデータベースによって使用されているかどうかを検査します。データベース・パーティション・サーバーをドロップできるかどうかを示すメッセージが戻されます。

有効範囲

この API は、それが発行されたデータベース・パーティション・サーバーにのみ影響を与えます。

許可

以下のいずれか。

- sysadm
- sysctrl

API 組み込みファイル

```
sqlenv.h
```

API とデータ構造構文

```
SQL_API_RC SQL_API_FN
  sqledrpn (
        unsigned short Action,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdrpn (
        unsigned short Reserved1,
        struct sqlca * pSqlca,
        void * pReserved2,
        unsigned short Action);
```

sqledrpn API パラメーター

アクション
要求されたアクション。有効値は以下のとおりです。
SQL_DROPNODE_VERIFY

pReserved
予約済み。 NULL にする必要があります。

pSqlca 出力。sqlca 構造を指すポインター。

sqlgdrpn API 固有パラメーター

Reserved1

pReserved2 の長さのために予約されています。

pReserved2

NULL に設定されたスペア・ポインター、または 0 を指すスペア・ポインター。将来使用するために予約されています。

使用上の注意

データベース・パーティション・サーバーが使用されていないことを示すメッセージが戻された場合は、 db2stop コマンドと DROP NODENUM を使用して、db2nodes.cfg ファイルからそのデータベース・パーティション・サーバーの項目をドロップします。そうすると、パーティション・データベース環境からデータベース・パーティション・サーバーがドロップされます。

データベース・パーティション・サーバーが使用されていることを示すメッセージが戻された場合は、以下の処置を実行してください。

1. ドロップされるデータベース・パーティション・サーバーには、インスタンス内の各データベース用にデータベース・パーティションがあります。これらのデータベース・パーティションのいずれかにデータが含まれている場合は、データベース・パーティションを使用するデータベース・パーティション・グループを再分散します。データベース・パーティション・グループを再分散し、ドロップされないデータベース・パーティション・サーバーにあるデータベース・パーティションにデータを移動させます。
2. データベース・パーティション・グループの再分散後、データベース・パーティションを使用する各データベース・パーティション・グループからデータベース・パーティションをドロップします。データベース・パーティションをデータベース・パーティション・グループからドロップするには、 sqludrdt API のノード・ドロップ・オプション、または ALTER DATABASE PARTITION GROUP ステートメントを使用できます。
3. データベース・パーティション・サーバーで定義されているイベント・モニターをドロップします。
4. sqledrpn を再実行して、データベース・パーティション・サーバーのデータベース・パーティションが使用されていないことを確認します。

REXX API 構文

この API は、SQLDB2 インターフェースを使って、REXX から呼び出すことができます。

sqlugrpn - 特定の行についてのデータベース・パーティション・サーバー番号の取得

分散キー値に基づいてデータベース・パーティション番号およびデータベース・パーティション・サーバー番号を戻します。アプリケーションは、この情報を使用して、表の特定の行が保管されているデータベース・パーティション・サーバーを判別することができます。

パーティション・データ構造 (sqlupi) はこの API への入力となります。この構造は sqlugtpi API によって戻すことができます。別の入力としては、対応する分散キー値の文字表示があります。出力は、分散ストラテジーによって生成されたデータベース・パーティション番号と、分散マップからの対応するデータベース・パーティション・サーバー番号です。分散マップ情報が提供されていない場合には、データベース・パーティション番号のみが戻されます。この API は、データ分散を分析する際に役立ちます。

この API の呼び出し時にデータベース・マネージャーが実行している必要はありません。

有効範囲

この API は、db2nodes.cfg ファイル内のデータベース・パーティション・サーバーから呼び出す必要があります。この API は、クライアントとサーバーの間でコード・ページやバイトの並び順などに違いがあると、誤ったデータベース・パーティション情報が戻される危険があるため、クライアントから呼び出すべきではありません。

許可

なし

API 組み込みファイル

```
sqlutil.h
```

API とデータ構造構文

```
SQL_API_RC SQL_API_FN
  sqlugrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_ctrycode,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);

SQL_API_RC SQL_API_FN
  sqlggrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_code,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);
```

sqlugrpn API パラメーター

num_ptrs
ptr_array 内のポインターの数。この値は、part_info パラメーターに指定されるものと同じでなければなりません。つまり、part_info->sqld です。

ptr_array
part_info に指定された分散キーの各パーツに対応する値の文字表示を指すポインターの配列。 NULL 値が必要とされる場合には、対応するポインターが NULL に設定されます。生成された列に関しては、この機能は行の値を生成しません。ユーザーは行を正しくパーティション化することにつながるような値を提供する責任があります。

ptr_lens
part_info に指定されたパーティション・キーの各部に対応する値の文字表示の長さを表す符号なし整数の配列。

territory_ctrycode
ターゲット・データベースの国地域コード。この値は、GET DATABASE CONFIGURATION コマンドを使用してデータベース構成ファイルから入手することもできます。

codepage
ターゲット・データベースのコード・ページ。この値は、GET DATABASE CONFIGURATION コマンドを使用してデータベース構成ファイルから入手することもできます。

part_info
sqlupi 構造を指すポインター。

part_num
データベース・パーティション番号の保管に使用される 2 バイトの符号付き整数を指すポインター。

node_num
ノード番号の保管に使用される SQL_PDB_NODE_TYPE フィールドを指すポインター。ポインターが NULL の場合、ノード番号は戻されません。

chklvl 入力パラメーターに対して行われる検査のレベルを指定する符号なし整数。指定された値がゼロである場合、検査は行われません。ゼロ以外の値が指定された場合には、すべての入力パラメーターがチェックされます。

sqlca 出力。sqlca 構造を指すポインター。

dataformat
分散キー値の表示を指定します。有効な値は以下のとおりです。

SQL_CHARSTRING_FORMAT
すべての分散キー値は文字ストリングによって表示されます。これはデフォルト値です。

SQL_IMPLIEDDECIMAL_FORMAT
暗黙指定されている小数点の位置が列定義によって決定されます。例えば、列定義が DECIMAL(8,2) である場合、値 12345 は 123.45 として処理されます。

SQL_PACKEDDECIMAL_FORMAT
すべての 10 進数列分散キー値はパック 10 進数形式になります。

SQL_BINARYNUMERICS_FORMAT
すべての数値分散キー値はビッグ・エンディアン・バイナリー数形式になります。

pReserved1
将来の使用のために予約されています。

pReserved2
将来の使用のために予約されています。

使用上の注意

オペレーティング・システムでサポートされるデータ・タイプは、分散キーとして定義できるものと同じです。

注: CHAR、VARCHAR、GRAPHIC、および VARGRAPHIC データ・タイプは、この API を呼び出す前にデータベース・コード・ページに変換しなければなりません。

数値および日時データ・タイプの場合、文字表示は、 API が呼び出されるそれぞれのシステムのコード・ページで表記しなければなりません。

node_num が NULL でない場合には、分散マップを提供しなければなりません。つまり、 part_info パラメーターの pmaplen フィールド (part_info->pmaplen) を 2 または 8192 のどちらかにする必要があります。そうでなければ、SQLCODE -6038 が戻されます。分散キーを定義しなければなりません。つまり、part_info パラメーターの sqld フィールド (part_info->sqld) をゼロより大きくしてください。そうでなければ、SQLCODE -2032 が戻されます。

NULL 値不可のパーティション列に NULL 値が割り当てられている場合には、SQLCODE -6039 が戻されます。

入力文字ストリングの先行ブランクと後書きブランクは、すべて除去されます。ただし、CHAR、VARCHAR、GRAPHIC、および VARGRAPHIC データ・タイプの場合は、後書きブランクのみが除去されます。

第 30 章 コマンド

REDISTRIBUTE DATABASE PARTITION GROUP

データベース・パーティション・グループ内のデータベース・パーティション間でデータを再分散します。データの配分先は、均一 (デフォルト) にするか、またはシステム固有の必要を満たすためにユーザー指定にすることができます。

REDISTRIBUTE DATABASE PARTITION GROUP コマンドは、データベース・パーティション・グループ内のすべてのパーティション間でデータを再配分します。これはデータベース・パーティション・グループ内に存在するすべてのオブジェクトに影響を与え、1 つのオブジェクトのみに制限することはできません。

このコマンドは、カタログ・データベース・パーティションからしか発行できません。どのデータベース・パーティションが各データベースのカタログ・データベース・パーティションになっているかを判別するには、 LIST DATABASE DIRECTORY コマンドを使用します。

有効範囲

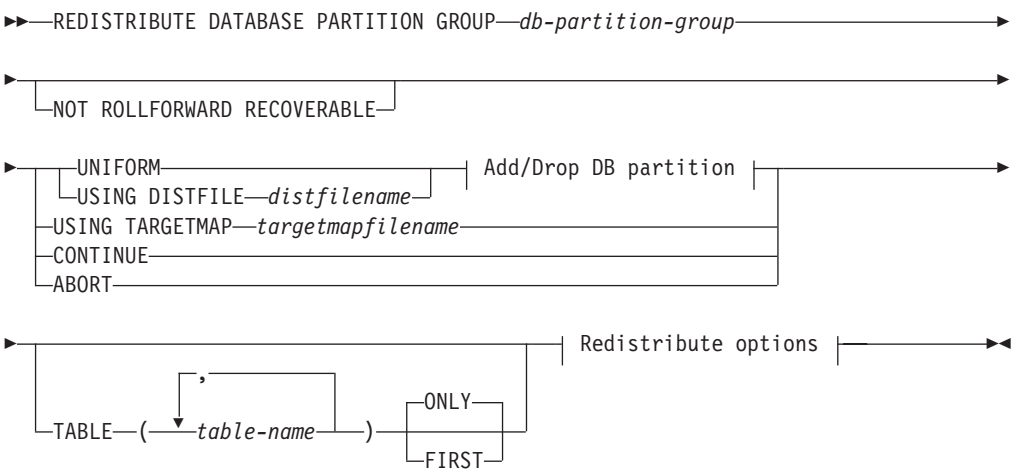
このコマンドは、データベース・パーティション・グループ内のすべてのデータベース・パーティションに影響を与えます。

許可

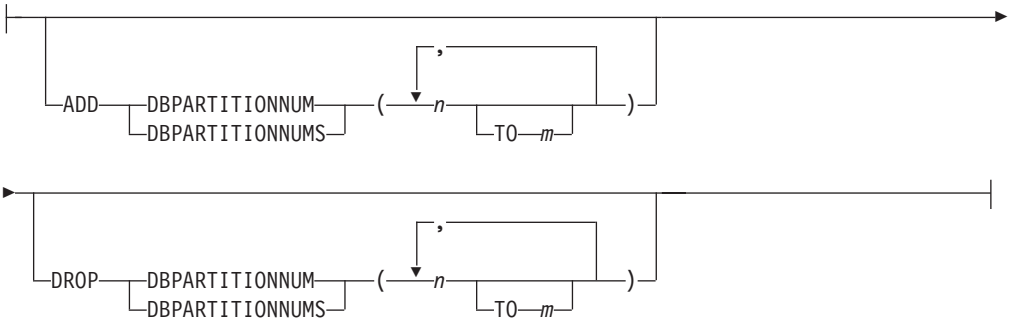
以下のいずれか。

- SYSADM
- SYSCTRL
- DBADM

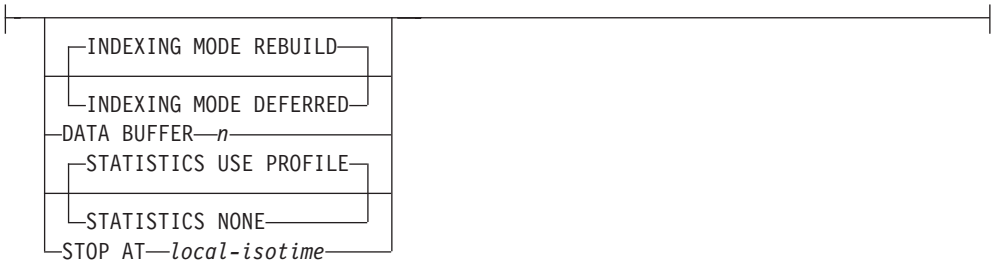
コマンド構文



Add/Drop DB partition:



Redistribute options:



コマンド・パラメーター

DATABASE PARTITION GROUP *db-partition-group*
データベース・パーティション・グループの名前。この 1 部構成の名前は、SYSCAT.DBPARTITIONGROUPS カタログ表に記述されたデータベース・パーティション・グループを識別します。データベース・パーティション・グループは、現在再配布を受けることはできません。

注: IBMCATGROUP および IBMTEMPGROUP データベース・パーティション・グループ内の表を再配分することはできません。

NOT ROLLFORWARD RECOVERABLE
このオプションは、DB2 9.5 フィックスパック 1 がインストールされている場合にのみ使用できます。このオプションを使用すると、REDISTRIBUTE DATABASE PARTITION GROUP コマンドはロールフォワード・リカバリー可能ではありません。

- データは、内部での挿入および削除操作によってではなく、一括して移動されます。これにより、表のスキャンおよびアクセスの回数が減り、パフォーマンスが向上します。
- 挿入および削除操作それぞれに対するログ・レコードは必要ではなくなりました。このため、データの再配分を実行するときに、システム内で大容量のアクティブ・ログ・スペースおよびログ・アーカイブ・スペースを管理する必要がなくなりました。過去に、大容量のアクティブ・ログ・スペースとストレージの要件のために、単一のデータ再配分操作を複数の小規

模の再配分タスクに分割することを強制され、その結果、エンドツーエンドのデータ再配分操作にさらに時間が要求されていた場合には、これは特に有益です。

このオプションが使用されない 場合には、すべての行移動に関する詳細なロギングが実行されるため、中断、エラー、または他のビジネスの必要が生じた場合に、データベースを後からリカバリーすることができます。

UNIFORM

データがハッシュ・パーティションにわたって均等に配分されることを指定します (つまり、それぞれのハッシュ・パーティションが同じ数の行を持つことが想定されます)。しかし、それぞれのデータベース・パーティションに同じ数のハッシュ・パーティションはマップされません。再配分後、データベース・パーティション・グループのすべてのデータベース・パーティションは、ほぼ同じ数のハッシュ・パーティションを持っています。

USING DISTFILE *distfilename*

分散キー値の分散に偏りがある場合、このオプションを使用して、データベース・パーティション・グループのデータベース・パーティション全体にわたるデータの均一な再分散を行います。

distfilename を使用して、4096 個のハッシュ・パーティションにわたる現行のデータの配分を指示します。

行カウント、バイト・ボリューム、または他の任意の尺度を使用して、各ハッシュ・パーティションで表示されたデータ量を示します。ユーティリティーは、パーティションに関連する整数値をそのパーティションの重みとして読み取ります。 *distfilename* を指定した場合、ユーティリティーはターゲット分散マップを生成し、データベース・パーティション・グループのデータベース・パーティション全体にデータをできるだけ均一に再配分するために使用します。再配分の後、データベース・パーティション・グループ中の各データベース・パーティションの重みは、ほぼ同じになります (データベース・パーティションの重みは、データベース・パーティションにマップしたすべてのハッシュ・パーティションの重みの合計です)。

たとえば、入力配布ファイルに以下の項目があるとします。

```
10223
1345
112000
0
100
...
```

例の中で、ハッシュ・パーティション 2 は 112000 の重みを持ち、パーティション 3 (重さは 0) には、マッピングするデータがまったくありません。

distfilename には、4096 の正整数値が文字形式で入っていなければなりません。値の合計は、4294967295 以下である必要があります。

distfilename のパスが指定されていない場合、現行ディレクトリーが使用されます。

USING TARGETMAP *targetmapfilename*

targetmapfilename で指定されたファイルは、ターゲット分散マップとして使

用されます。データの再配分はこのファイルに従って行われます。パスを指定していない場合、現行ディレクトリーが使用されます。

ターゲット・マップに含まれるデータベース・パーティションがデータベース・パーティション・グループ中に存在しないと、エラーが戻されます。REDISTRIBUTE DATABASE PARTITION GROUP コマンドを実行する前に、ALTER DATABASE PARTITION GROUP ADD DBPARTITIONNUM ステートメントを実行してください。

ターゲット・マップから除外されたデータベース・パーティションが、データベース・パーティション・グループにある 場合、そのデータベース・パーティションはパーティションの中に含まれていません。このようなデータベース・パーティションは、 REDISTRIBUTE DATABASE PARTITION GROUP コマンドの前か後に ALTER DATABASE PARTITION GROUP DROP DBPARTITIONNUM ステートメントを使用することによってドロップできます。

CONTINUE
直前に失敗または停止した REDISTRIBUTE DATABASE PARTITION GROUP 操作を継続します。何も起こらなければ、エラーが戻されます。

ABORT
直前に失敗または停止した REDISTRIBUTE DATABASE PARTITION GROUP 操作をアボートします。何も起こらなければ、エラーが戻されます。

ADD

DBPARTITIONNUM *n*

TO *m*

n または *n* **TO** *m* では、データベース・パーティション・グループに追加するデータベース・パーティション番号のリストを指定します。指定するパーティションは、データベース・パーティション・グループにすでに定義済みであってはなりません (SQLSTATE 42728)。ADD DBPARTITIONNUM 節を指定して ALTER DATABASE PARTITION GROUP ステートメントを実行する操作と等価です。

DBPARTITIONNUMS *n*

TO *m*

n または *n* **TO** *m* では、データベース・パーティション・グループに追加するデータベース・パーティション番号のリストを指定します。指定するパーティションは、データベース・パーティション・グループにすでに定義済みであってはなりません (SQLSTATE 42728)。ADD DBPARTITIONNUM 節を指定して ALTER DATABASE PARTITION GROUP ステートメントを実行する操作と等価です。

注: このオプションを使用してデータベース・パーティションを追加すると、表スペースのコンテナーは、データベース・パーティション・グループ内で最も小さい番号の既存のパーティション内の対

応する表スペースに基づくことになります。その結果、コンテナーの名前の競合が発生する場合は、このオプションを使用しないでください (新しいパーティションが既存のコンテナーと同じ物理マシンにあると、そのような名前の競合が発生する可能性があります)。そのような場合は、WITHOUT TABLESPACES オプションを指定して ALTER DATABASE PARTITION GROUP ステートメントを使用してから、REDISTRIBUTE DATABASE PARTITION GROUP コマンドを実行してください。その後、適切な名前を指定して、表スペース・コンテナーを手動で作成できます。

DROP

DBPARTITIONNUM *n*

TO *m*

n または *n* **TO** *m* では、データベース・パーティション・グループからドロップするデータベース・パーティション番号のリストを指定します。指定するパーティションは、データベース・パーティション・グループにすでに定義されている必要があります (SQLSTATE 42729)。DROP DBPARTITIONNUM 節を指定して ALTER DATABASE PARTITION GROUP ステートメントを実行する操作と等価です。

DBPARTITIONNUMS *n*

TO *m*

n または *n* **TO** *m* では、データベース・パーティション・グループからドロップするデータベース・パーティション番号のリストを指定します。指定するパーティションは、データベース・パーティション・グループにすでに定義されている必要があります (SQLSTATE 42729)。DROP DBPARTITIONNUM 節を指定して ALTER DATABASE PARTITION GROUP ステートメントを実行する操作と等価です。

TABLE *tablename*
再配分処理の表の順序を指定します。

ONLY 表の順序の後に **ONLY** キーワード (デフォルト) を使用すると、指定した表だけが再配分の対象になります。残りの表は、後から REDISTRIBUTE CONTINUE コマンドによって処理できます。これはデフォルトです。

FIRST 表の順序の後に **FIRST** キーワードを使用すると、指定した表が指定の順序で再配分処理を受け、データベース・パーティション・グループ内の残りの表は、ランダムな順序で再配分処理を受けることになります。

INDEXING MODE
NOT ROLLFORWARD RECOVERABLE オプションが指定されている場合、このパラメーターで、再配分の際に索引を維持する方法を指定します。有効な値は以下のとおりです。

REBUILD
索引は、最初から再作成されます。このオプションを使用する場合は、索引が有効である必要はありません。このオプションを使用すれば、索引ページはディスク上で一緒にクラスター化されます。

DEFERRED
再配分では、索引の保守作業を実行しません。リフレッシュが必要であることを示すマークが索引に付けられます。そのような索引に最初にアクセスした時点で再作成が強制実行されるか、データベースの再始動時に索引が再作成されることになります。

注: MDC 表以外の場合は、INDEXING MODE DEFERRED を指定しなくても、表に無効な索引があれば、**REDISTRIBUTE DATABASE PARTITION GROUP** コマンドによって、そのような索引の再作成が自動的に実行されます。MDC 表の場合は、**INDEXING MODE DEFERRED** を指定しても、表の再配分の開始前に無効な複合索引が再作成されます。ユーティリティーは、MDC 表を処理するために複合索引を必要とするからです。

DATA BUFFER *n*
ユーティリティー内でデータを転送するためのバッファー・スペースとして使用する 4 KB ページ数を設定します。指定された値がサポートされている最小値よりも低い場合には、最低の値が使用され、警告は戻されません。このメモリーは、ユーティリティー・ヒープから直接に割り当てられ、そのサイズは **util_heap_sz** データベース構成パラメーターで修正可能です。値を指定しないと、実行時に各表の処理を開始する時点でユーティリティーによって適切なデフォルトが計算されます。具体的には、表の再配分の開始時点でユーティリティー・ヒープで使用可能になっているメモリーの 50% を使用することを基本にしながら、さまざまな表プロパティーを考慮に入れることによって、デフォルトを計算することになります。

STOP AT *local-isotime*
このオプションを指定すると、各表のデータ再配分を開始する前に、*local-isotime* と現在のローカル・タイム・スタンプが比較されます。指定した *local-isotime* が現在のローカル・タイム・スタンプと同じか、それよりも早いと、ユーティリティーは処理を停止して、警告メッセージを生成します。停止時に進行中であった表のデータ再配分の処理は中断されずに完了します。表の新規のデータ再配分の処理は開始されません。未処理の表の再配分を実行するには、**CONTINUE** オプションを使用します。この *local-isotime* 値は、日付と時刻の組み合わせを識別する 7 部構成の文字ストリングのタイム・スタンプとして指定します。形式は、現地時間の *yyyy-mm-dd-hh.mm.ss.nnnnnn* (年、月、日、時、分、秒、マイクロ秒) です。

STATISTICS
このオプションでは、ユーティリティーが、統計プロファイルのある表の統計を収集するかどうかを指定します。このオプションを指定するほうが、データ再配分の完了後に RUNSTATS コマンドを別途実行するよりも効率的です。

USE PROFILE
統計プロファイルのある表の統計を収集します。統計プロファイルのない表については、何も実行されません。これはデフォルトです。

NONE 表の統計を収集しません。

例: 再配分の手順

ノード・グループからノードを追加またはドロップするとします。新規ノードをノード・グループに追加してデータを再配分する手順を以下に示します。追加されたデータベース・パーティションは分散マップにありませんが、データベース・パーティション・グループ内の表スペース用のコンテナーは作成されています。データベース・パーティション・グループの再配分操作が正常に完了すると、データベース・パーティションは分散マップに追加されます。

1. 再配分する必要のあるノード・グループを確認します。本書では、再配分する必要のあるノード・グループを **sampleNodegrp** という名前にします。
2. 再配分する前に使用不可にする必要がある、または削除する必要のあるオブジェクトを確認します。
   a. 複製 MQT: このタイプの MQT は REDISTRIBUTE ユーティリティーの一部としてサポートされていません。再配分を実行する前にドロップし、後ほど再作成する必要があります。

      ```
      SELECT tabschema, tabname FROM syscat.tables WHERE partition_mode = 'R'
      ```

   b. 表書き込みイベント・モニター: 自動開始する表書き込みイベント・モニターで、再配分するノード・グループにその表があるものがあれば、それらのイベント・モニターは無効にする必要があります。

      ```
      SELECT distinct evmonname FROM syscat.eventtables E
      JOIN syscat.tables T on T.tabname = E.tabname AND T.tabschema = E.tabschema
      JOIN syscat.tablespaces S on S.tbspace = T.tbspace AND S.ngname = 'sampleNodegrp'
      ```

   c. Explain 表: Explain 表は単一パーティション・ノード・グループ内に作成することをお勧めします。しかし、再配分を必要とするノード・グループ内で定義される場合、今までに生成されたデータを維持する必要がないなら、再配分する前にそれらをドロップして、再配分が完了した時に再定義することも可能です。
   d. 表アクセス・モードおよびロード状態: 再配分するノード・グループ内のすべての表がフル・アクセス・モードであること、および、ロード・ペンディングまたはロード進行中状態でないことを確認してください。

      ```
      SELECT distinct trim(T.creator) || ¥'.¥' || trim(T.name) as name, T.access_mode, A.load_status
       FROM sysibm.systables T, sysibm.sysnodegroups N, sysibmadm.admintabinfo A
       WHERE T.pmap_id = N.pmap_id
        AND A.tabschema = T.creator
        AND A.tabname = T.name
        AND N.name = 'sampleNodegrp'
        AND (T.access_mode <> 'F' or A.load_status is not null)
      ```

   e. 統計プロファイル: 統計プロファイルが表に定義されている場合、表統計は再配分プロセスの一環として更新することができます。REDISTRIBUTE ユーティリティーを使用して表統計を更新すると、データすべてが再配分のためにスキャンされて RUNSTATS 用のデータの追加スキャンが必要なくなるため、入出力を削減できます。

      ```
      RUNSTATS on table schema.table USE PROFILE runstats_profile SET PROFILE ONLY
      ```

3. データベース構成を確認します。**util_heap_sz** は、データベース・パーティション間のデータ移動処理にとって重要です。再配分時のために **util_heap_sz** にできるだけ多くのメモリーを割り振ってください。索引の再作成が再配分の一環として行われる場合、十分な **sortheap** が必要です。 **util_heap_sz** および **sortheap** を必要に応じて増やして、再配分のパフォーマンスを向上させてください。
4. 新規データベース・パーティションに使用されるデータベース構成設定を取得します。データベース・パーティションを追加する時は、デフォルトのデータベース設定が使用されます。そのため、REDISTRIBUTE コマンドを発行する前に新規ノード上でデータベース構成を更新することにより、ウェアハウス全体が均衡のとれた構成になるようにすることが重要です。

```
SELECT name,
CASE WHEN deferred_value_flags = 'AUTOMATIC' THEN deferred_value_flags ELSE substr(deferred_value,1,20) END as deferred_value
FROM sysibmadm.dbcfg
WHERE dbpartitionnum = existing-node
AND deferred_value != ''
AND name not in ('hadr_local_host','hadr_local_svc','hadr_peer_window',
          'hadr_remote_host','hadr_remote_inst','hadr_remote_svc',
          'hadr_syncmode','hadr_timeout','backup_pending',
          'codepage','codeset','collate_info','country',
          'database_consistent','database_level',
          'hadr_db_role','log_retain_status',
          'loghead','logpath','multipage_alloc','numsegs',
          'pagesize','release','restore_pending','restrict_access',
          'rollfwd_pending','territory','user_exit_status','number_compat',
          'varchar2_compat','database_memory')
```

5. 再配分を開始する前に、データベース (または再配分するノード・グループ内の表スペース) をバックアップして、最新のリカバリー・ポイントを確保します。
6. db2nodes.cfg ファイルを更新して新規データ BCU データベース・パーティションの指定を追加することにより、新規データ BCU を DB2 に定義します。そして ADD NODE WITHOUT TABLESPACES コマンドを使用して、新規のデータベース・パーティションを DB2 に定義します。

   ```
   db2start nodenum x export DB2NODE=x
   db2 add node without tablespaces
   db2stop nodenum x
   ```

   注: それがデータ BCU 上に最初の論理ポートでない場合は、後続の論理ポートのために、上記のコマンド手順を実行する前後に、最初の論理ポート番号の始動および停止を実行します。
7. 新たに定義されたデータベース・パーティションに SYSTEM TEMPORARY 表スペース・コンテナーを定義します。

   ```
   ALTER TABLESPACE tablespace_name ADD container_information ON dbpartitionnums (x to y)
   ```

8. 新規の論理データベース・パーティションを複数のデータ BCU にまたがるデータベース・パーティション・グループに追加します。

   ```
   ALTER DATABESE PARTITION GROUP partition_group_name ADD dbpartitionnums (x to y) WITHOUT TABLESPACES
   ```

9. 新たに定義されたデータベース・パーティションに永続データの表スペース・コンテナーを定義します。

   ```
   ALTER TABLESPACE tablespace_name ADD container_information ON dbpartitionnums (x to y)
   ```

10. ステップ 4 で取得したデータベース構成設定を新規のデータベース・パーティションに適用します。 (または、新規の DB2 9.5 の構成サポートの単一ビューを使用して、すべてのデータベース・パーティションに対して単一の UPDATE DB CFG コマンドを発行します。)

11. 再配分するデータベース・パーティション・グループに複製 MQT がある場合、それらの定義をキャプチャーしてから、複製 MQT をドロップします。

    ```
    db2look -d dbname -e -z schema -t replicated_MQT_table_names -o repMQTs.clp
    ```

12. 再配分するデータベース・パーティション・グループにある表書き込みイベント・モニターを使用不可にします。

    ```
    SET EVENT MONITOR monitor_name STATE 0
    ```

13. REDISTRIBUTE ユーティリティーを実行してすべてのデータベース・パーティションに均等に再配分します。以下に単純な再配分コマンドを示します。

    ```
    REDISTRIBUTE DATABASE PARTITION GROUP sampleNodegrp NOT ROLLFORWARD RECOVERABLE uniform;
    ```

また、REDISTRIBUTE コマンドへの入力として表リストを指定することによって、表が処理される順序を強制することも考慮する必要があります。REDISTRIBUTE ユーティリティーはデータを移動させます (圧縮および短縮)。オプションで、索引の再編成、および統計プロファイルが定義されている場合は統計の更新が行われます。それで、以前のコマンドの代わりに以下のスクリプトを実行できます。

```
REDISTRIBUTE DATABASE PARTITION GROUP sampleNodegrp
  NOT ROLLFORWARD RECOVERABLE uniform TABLE (tab1, tab2,...) FIRST;
```

NOT ROLLFORWARD RECOVERABLE オプションの使用結果

REDISTRIBUTE DATABASE PARTITION GROUP コマンドが発行され、NOT ROLLFORWARD RECOVERABLE オプションが指定されると、移動される各行のログ・レコードの書き込みを最小化する、最小限のロギング方式が使用されます。再配分操作のユーザビリティーという観点からすれば、このタイプのロギングは重要です。大規模なシステムですべてのデータ移動を完全にログに記録する方式を採用すれば、アクティブな永続ログ・スペースが大量に必要になり、通常はパフォーマンス特性が悪化してしまいます。一方、この最小限のロギング・モデルの結果として、REDISTRIBUTE DATABASE PARTITION GROUP コマンドはロールフォワード・リカバリー可能ではない ということをユーザーが忘れないようにするのは重要です。つまり、再配分操作の途中でデータベースのロールフォワードが必要になるような操作を実行すると、再配分操作の対象になったすべての表は、UNAVAILABLE 状態のままになってしまいます。そのような表については、ドロップ操作しか実行できません。つまり、そのような表のデータをリカバリーする方法はない、ということです。したがって、リカバリー可能なデータベースについては、REDISTRIBUTE DATABASE PARTITION GROUP ユーティリティーが、NOT ROLLFORWARD RECOVERABLE オプションを指定して発行されると、対象になるすべての表スペースを BACKUP PENDING 状態に設定し、成功した再配分操作の最後に、再配分を受けたすべての表スペースのバックアップをユーザーが実行するように強制しています。再配分操作の後に作成したバックアップがあれば、ユーザーが再配分操作の途中でロールフォワードを実行する必要はなくなるからです。

再配分ユーティリティーがロールフォワード・リカバリー可能でないことの結果として、ユーザーが注意しなければならない重要な点が 1 つあります。再配分操作の実行中に (再配分の最後に再配分の対象になった表スペースのバックアップをユーザーが作成している時間も含めてですが)、ユーザーがそのデータベースの表に対する更新を認めると、その表が再配分対象のデータベース・パーティション・グループに含まれていない場合でも、データベース・コンテナーの破壊などの重大な障害が発生すれば、その更新内容が失われてしまう可能性があります。その更新内容が失われてしまうのは、再配分操作がロールフォワード・リカバリー可能でないから

です。再配分操作の前に作成したバックアップからデータベースをリストアしなければならない場合、再配分操作の実行中に更新された内容を再生するために、ログに基づくロールフォワードを実行することは不可能です。すでに触れたとおり、再配分のロールフォワードが発生すると、再配分の対象になった表は UNAVAILABLE 状態になるからです。したがって、その状況で実行できるのは、ロールフォワードのない再配分の前に作成したバックアップからデータベースをリストアする操作だけになります。その後、再配分操作を再度実行できます。しかし、元の再配分操作の実行中に更新された内容はすべて失われてしまいます。

この点の重要性は、いくら強調しても強調しすぎるということはありません。再配分操作の実行中に更新内容を失わないようにするために、以下のいずれかの条件を満たすようにする必要があります。

- REDISTRIBUTE DATABASE PARTITION GROUP コマンドの操作の実行中は、コマンドの完了後に操作対象の表スペースのバックアップを作成する時間も含めて、ユーザーが更新を実行しないようにすること。
- 再配分操作中に適用する更新内容を反復可能なソースに入れておくこと。その場合は、後からいつでも更新を再適用できます。例えば、ファイルに格納したデータを更新のソースとして使用し、バッチ処理でその更新を適用するようにすれば、データベースのリストアを必要とするような障害が発生したとしても、その更新内容は失われません。その更新内容を後から再び適用すればよいからです。

再配分操作中のデータベース更新を認めるかどうかについては、データベースのリストアが必要になった場合に更新処理を繰り返せるかどうかに基づいて、それぞれのシナリオでそのような更新が適切かどうかを判断する必要があります。

注: REDISTRIBUTE DATABASE PARTITION GROUP コマンドの操作中に発生するあらゆる障害がこのような問題につながるわけではありません。実際のところ、ほとんどの障害では、このような問題は発生しません。REDISTRIBUTE DATABASE PARTITION GROUP コマンドは完全に再始動可能であり、ユーティリティーの処理が途中で失敗したとしても、**CONTINUE** または **ABORT** オプションで簡単に続行/アボートできます。ここで取り上げてきた障害は、あくまでも、再配分操作の前に作成したバックアップからユーザーがデータベースをリストアしなければならなくなるような障害です。

使用上の注意

- **NOT ROLLFORWARD RECOVERABLE** オプションが指定され、データベースがリカバリー可能である場合、ユーティリティーが最初に表スペースにアクセスした時点で、表スペースは BACKUP PENDING 状態になります。表スペースのバックアップが作成されるまで、その表スペースに含まれているすべての表は読み取り専用になります。表スペースのバックアップは、その表スペース内のすべての表の再配分処理が完了したときにのみ可能になります。
- 再配分操作の実行中に、イベント・ログ・ファイルが生成されます。そのファイルには、再配分操作に関する一般情報と、各表の処理の開始時刻と終了時刻などの情報が組み込まれます。このイベント・ログ・ファイルは、以下の場所に書き込まれます。
  - Linux システムと UNIX オペレーティング・システムの場合は、homeinst/sqllib/redist ディレクトリー。サブディレクトリーとファイル名の形式は、*database-name.database-partition-group-name.timestamp*.log になります。

– Windows オペレーティング・システムの場合は、**DB2INSTPROF***instance***redist ディレクトリー (**DB2INSTPROF** は、**DB2INSTPROF** レジストリー変数の値です)。サブディレクトリーとファイル名の形式は、*database-name.database-partition-group-name.timestamp*.log になります。
– タイム・スタンプ値は、コマンドが発行された時の時刻です。

再配分イベント・ログについて詳しくは、『再配分イベント・ログ・ファイル』のトピックを参照してください。

- このユーティリティーは、処理の途中に断続的 COMMIT を実行します。
- 再配分を受けた表と従属関係があるすべてのパッケージは無効にされます。再配分データベース・パーティション・グループ操作が完了した後に、そのようなパッケージを明示的に再バインドすることをお勧めします。明示の再バインドによって、無効パッケージの最初の SQL 要求の実行を初期遅延させなくします。再配分メッセージ・ファイルには、再配分を受けたすべての表のリストが入っています。
- 統計プロファイルがある表については、再配分ユーティリティーの実行時に、デフォルトで統計が更新されます。統計プロファイルがない表の場合は、表や索引の統計を別途更新することをお勧めします。そのためには、再配分操作の完了後に、db2Runstats API を呼び出すか、RUNSTATS コマンドを実行できます。
- 複製されたマテリアライズ照会表や、DATA CAPTURE CHANGES で定義されている表が含まれているデータベース・パーティション・グループでは、再配分を実行できません。
- ユーザー TEMPORARY 表スペースと既存の宣言済み一時表とがデータベース・パーティション・グループに存在する場合、再配分は許可されません。
- **INDEXING MODE** などのオプションは、適用対象にならない表では無視されます (警告も生成されません)。例えば、**INDEXING MODE** は、索引のない表では無視されます。
- 再配分操作を開始する前に、ロード・ペンディング状態の表がないことを確認してください。表状態は LOAD QUERY コマンドを使って調べることができます。

互換性

バージョン 8 より前のバージョンとの互換性 :

- キーワード **NODEGROUP** は、**DATABASE PARTITION GROUP** に置き換えられます。

db2nchg - データベース・パーティション・サーバー構成の変更

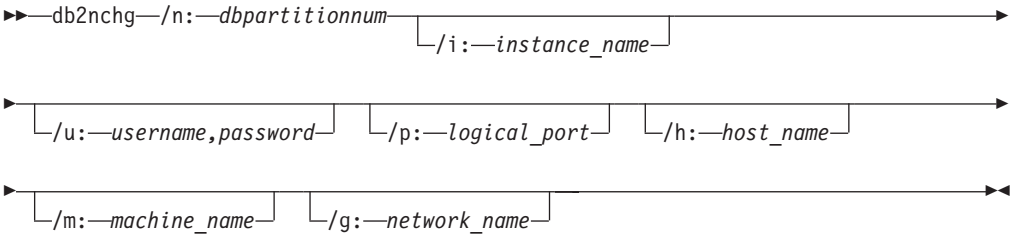
データベース・パーティション・サーバー構成を変更します。これには、あるマシンから別のマシンへのデータベース・パーティション・サーバー (ノード) の移動、マシンの TCP/IP ホスト名の変更、データベース・パーティション・サーバー (ノード) 用の別の論理ポート番号または別のネットワーク名の選択も含まれます。このコマンドが使用できるのは、データベース・パーティション・サーバーが停止している場合だけです。

このコマンドは、 Windows のオペレーティング・システムのみで使用できます。

許可

ローカル管理者

コマンド構文



コマンド・パラメーター

/n:*dbpartitionnum*
変更するデータベース・パーティション・サーバー構成のデータベース・パーティション番号を指定します。

/i:*instance_name*
このデータベース・パーティション・サーバーが参加するインスタンスを指定します。パラメーターが指定されていない場合、デフォルトは現行のインスタンスになります。

/u:*username,password*
ユーザー名およびパスワードを指定します。パラメーターが指定されない場合、既存のユーザー名とパスワードが設定されます。

/p:*logical_port*
データベース・パーティション・サーバー用の論理ポートを指定します。データベース・パーティション・サーバーを別のマシンに移動させるには、このパラメーターを指定する必要があります。パラメーターが指定されない場合、論理ポート番号は変更されません。

/h:*host_name*
内部通信用に FCM によって使用される TCP/IP ホスト名を指定します。パラメーターが指定されない場合、ホスト名は変更されません。

/m:*machine_name*
データベース・パーティション・サーバーが常駐するマシンを指定します。データベース・パーティション・サーバーを移動できるのは、インスタンス内にデータベースが 1 つもない場合だけです。

/g:*network_name*
データベース・パーティション・サーバーのネットワーク名を変更します。このパラメーターは、マシンに複数の IP アドレスがある場合に、特定の IP アドレスをデータベース・パーティション・サーバーに適用するために使用できます。ネットワーク名または IP アドレスを入力できます。

例

インスタンス TESTMPP に参加する、データベース・パーティション 2 に割り当てられている論理ポートを論理ポート 3 に変更するには、以下のコマンドを入力します。

```
db2nchg /n:2 /i:TESTMPP /p:3
```

db2ncrt - インスタンスへのデータベース・パーティション・サーバーの追加

データベース・パーティション・サーバー (ノード) をインスタンスに追加します。

このコマンドは Windows オペレーティング・システムでのみ使用できます。

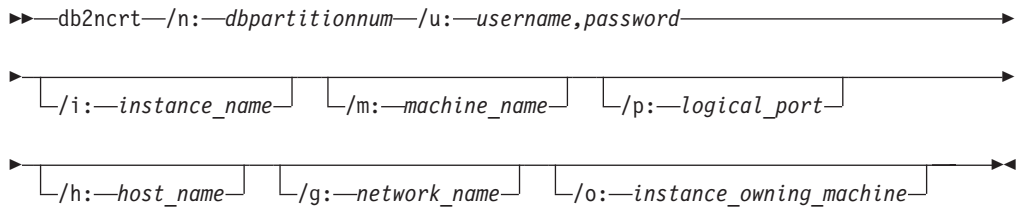
有効範囲

既にインスタンスが存在しているコンピューターにデータベース・パーティション・サーバーが追加される場合には、データベース・パーティション・サーバーはコンピューターへの論理データベース・パーティションとして追加されます。インスタンスが存在していないコンピューターにデータベース・パーティション・サーバーが追加される場合には、インスタンスが追加され、そのコンピューターは新しい物理データベース・パーティション・サーバーになります。インスタンスにデータベースがある場合には、このコマンドを使用してはなりません。代わりに、START DATABASE MANAGER コマンドを ADD DBPARTITIONNUM オプションを指定して発行してください。上記のコマンドにより、新しいデータベース・パーティション・サーバーにデータベースを正しく追加することができます。データベースが作成されたインスタンスにデータベース・パーティション・サーバーを追加することも可能です。 db2nodes.cfg ファイルは編集するべきではありません。このファイルを変更すると、パーティション・データベース環境に不整合が生じる可能性があるためです。

許可

新しいデータベース・パーティション・サーバーが追加されるコンピューターに対するローカル管理者権限。

コマンド構文



コマンド・パラメーター

/n:*dbpartitionnum*

データベース・パーティション・サーバーを識別する固有のデータベース・パーティション番号。 1 から 999 の範囲の番号を指定できます。

/u:*username,password*
DB2 のログオン・アカウント名およびパスワードを指定します。

/i:*instance_name*
インスタンス名を指定します。パラメーターが指定されていない場合、デフォルトは現行のインスタンスになります。

/m:*machine_name*
データベース・パーティション・サーバーが常駐する Windows ワークステーションのコンピューター名を指定します。データベース・パーティション・サーバーをリモート・コンピューター上に追加している場合、このパラメーターは必須です。

/p:*logical_port*
データベース・パーティション・サーバーに使用する論理ポート番号を指定します。このパラメーターが指定されていない場合、割り当てられる論理ポート番号は 0 です。論理データベース・パーティション・サーバーを作成する際には、このパラメーターを指定しなければならず、使用していない論理ポート番号を選択しなければなりません。以下の制限事項に注意してください。

- すべてのコンピューターには、論理ポートが 0 のデータベース・パーティション・サーバーがなければなりません。
- ポート番号は、`x:¥winnt¥system32¥drivers¥etc¥` ディレクトリーで FCM 通信に予約されているポートの範囲内でなければなりません。例えば、4 個のポートが現行のインスタンスに予約されている場合には、最大のポート番号は 3 になります。ポート 0 は、デフォルトの論理データベース・パーティション・サーバー用に使用されます。

/h:*host_name*
内部通信用に FCM によって使用される TCP/IP ホスト名を指定します。データベース・パーティション・サーバーをリモート・コンピューター上に追加する場合、このパラメーターは必須です。

/g:*network_name*
データベース・パーティション・サーバーのネットワーク名を指定します。パラメーターが指定されていない場合、システムで検出される最初の IP アドレスが使用されます。このパラメーターは、コンピューターに複数の IP アドレスがある場合に、特定の IP アドレスをデータベース・パーティション・サーバーに適用するために使用できます。ネットワーク名または IP アドレスを入力できます。

/o:*instance_owning_machine*
インスタンスを所有しているコンピューターのコンピューター名を指定します。デフォルトはローカル・コンピューターです。このパラメーターは、インスタンス所有コンピューターでないコンピューターで db2ncrt コマンドを呼び出すときに必須です。

例

インスタンス所有のコンピューター SHAYER 上で、インスタンス TESTMPP に新しいデータベース・パーティション・サーバーを追加する場合、新しいデータベー

ス・パーティション・サーバーがデータベース・パーティション 2 で、論理ポート 1 を使用する場合には、次のコマンドを入力します。

```
db2ncrt /n:2 /u:QBPAULZ¥paulz,g1reeky /i:TESTMPP /m:TEST /p:1 /o:SHAYER /h:TEST
```

db2ndrop - インスタンスからのデータベース・パーティション・サーバーのドロップ

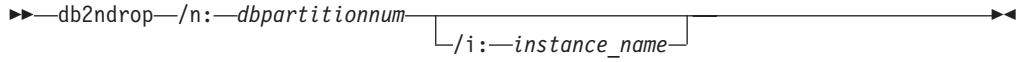
データベースのないインスタンスからデータベース・パーティション・サーバー (ノード) をドロップします。データベース・パーティション・サーバーがドロップされた場合には、このデータベース・パーティション番号を新しいデータベース・パーティション・サーバーで再使用できます。このコマンドが使用できるのは、データベース・パーティション・サーバーが停止している場合だけです。

このコマンドは、 Windows のオペレーティング・システムのみで使用できます。

許可

データベース・パーティション・サーバーをドロップするマシンに対するローカル管理者権限。

コマンド構文



コマンド・パラメーター

/n:*dbpartitionnum*
データベース・パーティション・サーバーを識別する固有のデータベース・パーティション番号。

/i:*instance_name*
インスタンス名を指定します。パラメーターが指定されていない場合、デフォルトは現行のインスタンスになります。

例

```
db2ndrop /n:2 /i=KMASCI
```

使用上の注意

インスタンスの所有するデータベース・パーティション・サーバー (dbpartitionnum 0) がインスタンスからドロップされると、このインスタンスは使用できなくなります。インスタンスをドロップするには、db2idrop コマンドを使用します。

このインスタンスにデータベースがある場合には、このコマンドを使用してはなりません。代わりに、db2stop drop nodenum コマンドを使用する必要があります。こうすると、パーティション・データベース環境からデータベース・パーティション・サーバーを確実に除去することができます。データベースが存在するインスタンスでデータベース・パーティション・サーバーをドロップすることも可能です。

db2nodes.cfg ファイルは編集するべきではありません。このファイルを変更すると、パーティション・データベース環境に不整合が生じる可能性があるためです。

複数の論理データベース・パーティション・サーバーを実行しているマシンから、論理ポート 0 に割り当てられたデータベース・パーティション・サーバーをドロップするには、他の論理ポートに割り当てられている他のすべてのデータベース・パーティション・サーバーを最初にドロップする必要があります。各データベース・パーティション・サーバーには、論理ポート 0 に割り当てられているデータベース・パーティション・サーバーが必ず必要です。

第 31 章 SQL 言語エレメント

データ・タイプ

データベース・パーティション互換データ・タイプ

データベース・パーティションの*互換性* は、分散キーの対応する列どうしのそれぞれの基本データ・タイプを対象に定義されます。データベース・パーティション互換データ・タイプには、型は異なるものの同じ値をもつ 2 つの変数が、同じデータベース・パーティション関数によって同じ分散マップ索引にマップされるという特性があります。

440 ページの表 41 は、データベース・パーティションのデータ・タイプの互換性を示しています。

データベース・パーティションの互換性には、次の特性があります。

- DATE、TIME、および TIMESTAMP には内部フォーマットが使用されます。これらは相互に互換性はなく、どれも文字データ・タイプまたはグラフィック・データ・タイプとの互換性はありません。
- パーティションの互換性は、列の NULL 可能性の影響を受けません。
- パーティションの互換性は、照合の影響を受けます。ロケールに依存する UCA ベースの照合は、照合の強さ属性が無視される以外、照合のときに完全な一致を必要とします。他のすべての照合は、パーティションの互換性を判別する目的で同等とみなされます。
- ロケールに依存する UCA ベースの照合以外の照合が使用されるとき、FOR BIT DATA で定義される文字列は、FOR BIT DATA のない文字列とのみ互換性があります。
- 互換データ・タイプの NULL 値は同じように取り扱われます。互換性のないデータ・タイプの NULL の場合は異なる結果が生じることがあります。
- UDT の基本データ・タイプは、データベース・パーティションの互換性を分析する場合に使用されます。
- 分散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) の末尾のブランクは、システムにより提供されるハッシュ関数によって無視されます。
- ロケールに依存する UCA ベースの照合が使用されるとき、CHAR、VARCHAR、GRAPHIC、および VARGRAPHIC は互換データ・タイプです。他の照合が使用される場合、CHAR と VARCHAR は互換タイプであり、GRAPHIC と VARGRAPHIC は互換タイプですが、CHAR と VARCHAR は GRAPHIC と VARGRAPHIC との互換タイプではありません。長さが異なる CHAR または VARCHAR は、互換データ・タイプです。

- 等しい DECFLOAT 値は、精度が異なっていても同一として取り扱われます。数値的に等しい DECFLOAT 値は、異なる数の有効桁数を持っていても同一として扱われます。

表 41. データベース・パーティションの互換性

オペランド	2 進整数	10 進数	浮動小数点	10 進浮動小数点	文字ストリング	GRAPHIC ストリング	日付	時刻	TIMESTAMP	特殊タイプ	構造化タイプ
2 進整数	あり	なし	なし	なし	なし	なし	なし	なし	なし	[1]	なし
10 進数	なし	あり	なし	なし	なし	なし	なし	なし	なし	[1]	なし
浮動小数点	なし	なし	あり	なし	なし	なし	なし	なし	なし	[1]	なし
10 進浮動小数点	なし	なし	なし	あり	なし	なし	なし	なし	なし	[1]	なし
文字ストリング[4]	なし	なし	なし	なし	あり[2]	2, 3	なし	なし	なし	[1]	なし
GRAPHIC ストリング[4]	なし	なし	なし	なし	2, 3	あり[2]	なし	なし	なし	[1]	なし
日付	なし	なし	なし	なし	なし	なし	あり	なし	なし	[1]	なし
時刻	なし	なし	なし	なし	なし	なし	なし	あり	なし	[1]	なし
Timestamp	なし	なし	なし	なし	なし	なし	なし	なし	あり	[1]	なし
特殊タイプ	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	なし
構造化タイプ[4]	なし	なし	なし	なし	なし	なし	なし	なし	なし	なし	なし

注:

[1] ユーザー定義特殊タイプ (UDT) の値には、UDT のソース・タイプ、もしくはデータベース・パーティション互換ソース・タイプをもったその他の UDT とのデータベース・パーティション互換性があります。

[2] 照合に互換性のある場合、文字およびグラフィック・ストリング・タイプは互換性があります。

[3] ロケールに依存する UCA ベースの照合が有効である場合、文字およびグラフィック・ストリング・タイプは互換性があります。そうでない場合、これらは互換タイプではありません。

[4] ユーザー定義構造化タイプとデータ・タイプ LONG VARCHAR、LONG VARGRAPHIC、CLOB、DBCLOB、および BLOB は、分散キーでサポートされていないので、データベース・パーティション互換性には該当しません。

特殊レジスター

CURRENT DBPARTITIONNUM

CURRENT DBPARTITIONNUM 特殊レジスターは、ステートメントのコーディネーター・ノード番号を識別する INTEGER 値を指定します。アプリケーションから発行されるステートメントの場合は、アプリケーションの接続先のデータベース・パーティションがコーディネーターになります。ルーチンから発行されるステートメントの場合は、ルーチンが呼び出されるデータベース・パーティションがコーディネーターになります。

ルーチン内部の SQL ステートメントで使用する場合、呼び出しステートメントからの CURRENT DBPARTITIONNUM の継承はありません。

データベース・インスタンスがデータベース・パーティション分割をサポートするように定義されていない場合、CURRENT DBPARTITIONNUM は 0 を戻します。(これはつまり、db2nodes.cfg ファイルが存在しない場合です。パーティション・データベースの場合は、db2nodes.cfg ファイルがあり、そこにデータベース・パーティションの定義が入っている場合です。)

CURRENT DBPARTITIONNUM は、ある一定の条件に該当する場合に限り、CONNECT ステートメントで変更できます。

バージョン 8 以前のバージョンと互換性を持たせるため、DBPARTITIONNUM の部分はキーワード NODE に置き換えられます。

例: 以下の例では、アプリケーションが接続しているデータベース・パーティションの番号をホスト変数 APPL_NODE (整数) に設定しています。

```
VALUES CURRENT DBPARTITIONNUM
  INTO :APPL_NODE
```

第 32 章 SQL 関数

DATAPARTITIONNUM

スキーマは SYSIBM です。

DATAPARTITIONNUM 関数は、行が置かれているデータ・パーティションのシーケンス番号 (SYSDATAPARTITIONS.SEQNO) を戻します。データ・パーティションは範囲別にソートされ、シーケンス番号は 0 から始まります。たとえば、範囲が最低のデータ・パーティションに置かれている行の場合、DATAPARTITIONNUM 関数から 0 が戻されます。

引数は、表内の任意の列の修飾された名前または無修飾の名前でなければなりません。行レベルの情報が戻されるので、どの列が指定されるかに関係なく、結果は同じです。該当の列は、どのようなデータ・タイプであっても構いません。

column-name がビューの列を参照する場合、そのビューの列の式は、基礎となる基本表の列を参照する必要があり、そのビューは削除可能でなければなりません。ネストされているか、または共通の表式は、ビューと同じ規則に従います。

結果のデータ・タイプは INTEGER であり、NULL 値にはなりません。

この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべてのデータ・タイプを引数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。

DATAPARTITIONNUM 関数は、チェック制約内、または生成された列の定義で使用することはできません (SQLSTATE 42881)。 DATAPARTITIONNUM 関数は、マテリアライズ照会表 (MQT) 定義の中でも使用できません (SQLSTATE 428EC)。

例:

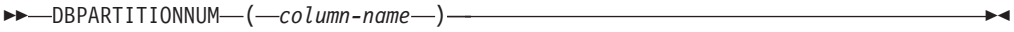
- ```
  SELECT DATAPARTITIONNUM (EMPNO)
  FROM EMPLOYEE
  ```

  DATAPARTITIONNUM によって戻されたシーケンス番号 (たとえば 0) を、他の SQL ステートメント (たとえば、ALTER TABLE...DETACH PARTITION) 内で使用できるデータ・パーティション名に変換するときは、SYSCAT.DATAPARTITIONS カタログ・ビューを照会することができます。以下の例で説明されているとおり、DATAPARTITIONNUM から取得された SEQNO を WHERE 節に組み込みます。

  ```
  SELECT DATAPARTITIONNAME
  FROM SYSCAT.DATAPARTITIONS
  WHERE TABNAME = 'EMPLOYEE' AND SEQNO = 0
  ```
  ```

上記の結果は、値 'PART0' になります。

DBPARTITIONNUM



スキーマは SYSIBM です。

DBPARTITIONNUM 関数は、行のデータベース・パーティション番号を戻します。たとえば、SELECT 節で使用すると、結果セット内の各行のデータベース・パーティション番号を戻します。

引数は、表内の任意の列の修飾された名前または無修飾の名前でなければなりません。行レベルの情報が戻されるので、どの列が指定されるかに関係なく、結果は同じです。該当の列は、どのようなデータ・タイプであっても構いません。

column-name がビューの列を参照する場合、そのビューの列の式は、基礎となる基本表の列を参照する必要があり、そのビューは削除可能でなければなりません。ネストされているか、または共通の表式は、ビューと同じ規則に従います。

DBPARTITIONNUM 関数によってデータベース・パーティション番号が戻される特定の行 (および表) は、この関数を使用する SQL ステートメントのコンテキストから判別されます。

遷移変数および表に戻されるデータベース・パーティション番号は、分散キー列の現行遷移値から得られます。たとえば、挿入前トリガーにおいて、新しい遷移変数の現行値があれば、関数は予想データベース・パーティション番号を戻します。ただし、分散キー列の値はそれ以後の挿入前トリガーによって変更される場合があります。したがって、データベースに挿入される時点での行の最終データベース・パーティション番号は、予測値とは異なるかもしれません。

結果のデータ・タイプは INTEGER であり、NULL 値にはなりません。`db2nodes.cfg` ファイルがない場合、結果は 0 になります。

この関数は、ユーザー定義関数の作成時にソース関数として使用することはできません。この関数は、すべてのデータ・タイプを引数として受け入れるので、ユーザー定義特殊タイプをサポートするための追加のシグニチャーを作成する必要はありません。

DBPARTITIONNUM 関数は、複製された表、チェック制約内、または生成された列の定義で使用することはできません (SQLSTATE 42881)。

以前のバージョンのDB2 との互換性: DBPARTITIONNUM の代わりに NODENUMBER を指定できます。

次に例を示します。

- EMPLOYEE 表内の指定された従業員の行が、DEPARTMENT 表内の従業員の部門についての記述とは異なるデータベース・パーティションにあるインスタンス数をカウントします。

```
SELECT COUNT(*) FROM DEPARTMENT D, EMPLOYEE E
  WHERE D.DEPTNO=E.WORKDEPT
  AND DBPARTITIONNUM(E.LASTNAME) <> DBPARTITIONNUM(D.DEPTNO)
```

- 2 つの表の行が同じデータベース・パーティションにあるようにするため、EMPLOYEE および DEPARTMENT の表を結合します。

```
SELECT * FROM DEPARTMENT D, EMPLOYEE E
  WHERE DBPARTITIONNUM(E.LASTNAME) = DBPARTITIONNUM(D.DEPTNO)
```

- EMPLOYEE 表で BEFORE トリガーを使用して、EMPINSERTLOG1 という表に、EMPLOYEE 表の従業員番号と新しい行の予想データベース・パーティション番号を記録します。

```
CREATE TRIGGER EMPINSLOGTRIG1
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AW NEWTABLE
FOR EACH ROW
INSERT INTO EMPINSERTLOG1
VALUES(NEWTABLE.EMPNO, DBPARTITIONNUM
(NEWTABLE.EMPNO))
```

第 33 章 SQL ステートメント

ALTER DATABASE PARTITION GROUP

ALTER DATABASE PARTITION GROUP ステートメントは、以下の目的で使用されます。

- データベース・パーティション・グループに 1 つ以上のデータベース・パーティションを追加する。
- データベース・パーティション・グループから 1 つ以上のデータベース・パーティションをドロップする。

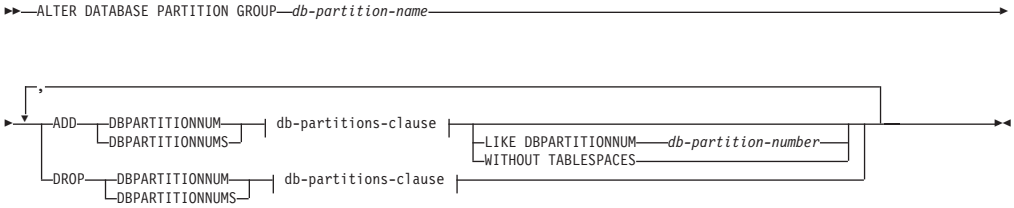
呼び出し方法

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

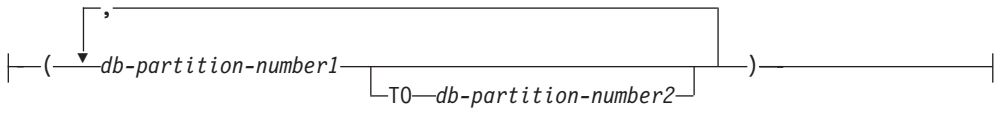
許可

このステートメントの許可 ID には、SYSCTRL 権限または SYSADM 権限がなければなりません。

構文



db-partitions-clause:



説明

db-partition-name
データベース・パーティション・グループの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。データベース・パーティション・グループはカタログに記述されている必要があります。IBMCATGROUP および IBMTEMPGROUP は指定できません (SQLSTATE 42832)。

ADD DBPARTITIONNUM
データベース・パーティション・グループに特定の 1 つまたは複数のデータベース・パーティションを追加することを指定します。 DBPARTITIONNUMS は DBPARTITIONNUM の同義語です。指定するデータベース・パーティションは、データベース・パーティション・グループにすでに定義済みであってはなりません (SQLSTATE 42728)。

DROP DBPARTITIONNUM
データベース・パーティション・グループから特定の 1 つまたは複数のデータベース・パーティションをドロップすることを指定します。
DBPARTITIONNUMS は DBPARTITIONNUM の同義語です。指定するデータベース・パーティションは、データベース・パーティション・グループにすでに定義されている必要があります (SQLSTATE 42729)。

db-partitions-clause
追加またはドロップする 1 つまたは複数のデータベース・パーティションを指定します。

db-partition-number1
特定のデータベース・パーティション番号を指定します。

TO *db-partition-number2*
データベース・パーティション番号の範囲を指定します。
db-partition-number2 の値は、 *db-partition-number1* の値以上でなければなりません (SQLSTATE 428A9)。

LIKE DBPARTITIONNUM *db-partition-number*
データベース・パーティション・グループの既存の表スペースのコンテナーが、指定した *db-partition-number* のコンテナーと同じであることを指定します。指定するデータベース・パーティションは、このステートメントの前にデータベース・パーティション・グループに存在しており、同じステートメントの DROP DBPARTITIONNUM 節に含まれていないパーティションである必要があります。

自動ストレージを使用するよう定義されている表スペース (つまり、CREATE TABLESPACE ステートメントの MANAGED BY AUTOMATIC STORAGE 節を使用して作成されているか、それに対してまったく MANAGED BY 節が指定されていない表スペース) については、コンテナーは必ずしも、指定のパーティションのものと一致するとは限りません。その代わり、コンテナーは、データベースに関連するストレージ・パスに基づいて、データベース・マネージャーによって自動的に割り当てられ、この結果として同じコンテナーが使用される場合もあれば、使用されない場合もあります。各表スペースのサイズは、表スペース作成時に指定された初期サイズに基づいて決まり、指定のパーティション上の表スペースの現行サイズと一致しない場合もあります。

WITHOUT TABLESPACES
データベース・パーティション・グループの既存の表スペースのコンテナーが、新規に追加された 1 つまたは複数のデータベース・パーティション上に作成されないことを指定します。 *db-partitions-clause* を用いた ALTER TABLESPACE ステートメントを使用して、このデータベース・パーティション・グループに対して定義される表スペースで使用するコンテナーを定義する必要があります。このオプションの指定がない場合、そのデータベース・パーティ

ション・グループに対して表スペースが定義されるたびに、新たに追加されるデータベース・パーティションにデフォルトのコンテナーが指定されます。

自動ストレージを使用するよう定義されている表スペース (つまり、CREATE TABLESPACE ステートメントの MANAGED BY AUTOMATIC STORAGE 節を使用して作成されているか、それに対してまったく MANAGED BY 節が指定されていない表スペース) については、このオプションは無視されます。こうした表スペースに関して、コンテナーの作成を後に延ばすということはできません。コンテナーは、データベース・マネージャーにより、データベースに関連するストレージ・パスを基に自動的に割り当てられます。各表スペースのサイズは、表スペース作成時に指定された初期サイズに基づいて決まります。

規則

- 番号によって指定するそれぞれのデータベース・パーティションは、db2nodes.cfg ファイルに定義されていなければなりません (SQLSTATE 42729)。
- *db-partitions-clause* にリストされる *db-partition-number* は、それぞれ固有のデータベース・パーティションに対する番号でなければなりません (SQLSTATE 42728)。
- 有効なデータベース・パーティション番号は、0 から 999 まで (0 と 999 を含む) です (SQLSTATE 42729)。
- 1 つのデータベース・パーティションを ADD と DROP の両方の節に指定することはできません (SQLSTATE 42728)。
- データベース・パーティション・グループには少なくとも 1 つのデータベース・パーティションが残っている必要があります。最後のデータベース・パーティションをデータベース・パーティション・グループからドロップすることはできません (SQLSTATE 428C0)。
- データベース・パーティションを追加する際に、 LIKE DBPARTITIONNUM 節も WITHOUT TABLESPACES 節も指定されていない場合、デフォルト解釈により、データベース・パーティション・グループの既存のデータベース・パーティションの最も小さいデータベース・パーティション番号 (ここでは *2* とします) が使用され、LIKE DBPARTITIONNUM *2* が指定された場合と同様の処理が行われます。既存のデータベース・パーティションをデフォルト値として使用する場合、そのデータベース・パーティションではデータベース・パーティション・グループ内のすべての表スペースに対してコンテナーが定義されている必要があります (SYSCAT.DBPARTITIONGROUPDEF の列 IN_USE が 'T' でない)。

注

- データベース・パーティションがデータベース・パーティション・グループに追加されると、そのデータベース・パーティションに対するカタログ項目が作成されます (SYSCAT.DBPARTITIONGROUPDEF を参照)。以下のいずれかの場合には、分散マップは直ちに変更され、新しいデータベース・パーティションが、そのデータベース・パーティションが分散マップにあることを示す標識 (IN_USE) を伴って組み込まれます。
  - データベース・パーティション・グループに表スペースが定義されていない、または
  - データベース・パーティション・グループに定義されている表スペースに表が定義されておらず、 WITHOUT TABLESPACES 節が指定されていない

以下のいずれかの場合は、分散マップは変更されず、標識 (IN_USE) はそのデータベース・パーティションが分散マップに組み込まれていないことを示すように設定されます。

– データベース・パーティション・グループの表スペースに表が存在する、または
– データベース・パーティション・グループに表スペースが存在し、WITHOUT TABLESPACES 節が指定された (すべての表スペースが自動ストレージを使用するよう定義されている場合を除く。その場合、WITHOUT TABLESPACES 節は無視される)。

分散マップを変更するには、REDISTRIBUTE DATABASE PARTITION GROUP コマンドを使用する必要があります。このコマンドは、任意のデータを再配分し、分散マップを変更し、標識を変更します。 WITHOUT TABLESPACES 節が指定された場合は、データを再配分する前に表スペース・コンテナーを追加する必要があります。

- データベース・パーティションがデータベース・パーティション・グループからドロップされると、そのデータベース・パーティションのカタログ項目 (SYSCAT.DBPARTITIONGROUPDEF を参照) が更新されます。データベース・パーティション・グループに定義された表スペースに表が定義されていない場合、分散マップが直ちに変更され、ドロップされたデータベース・パーティションを除外し、データベース・パーティション・グループのそのデータベース・パーティションに関する項目がドロップされます。表が存在する場合は、分散マップは変更されず、標識 (IN_USE) はそのデータベース・パーティションがドロップを待機していることを示すように設定されます。 REDISTRIBUTE DATABASE PARTITION GROUP コマンドは、データを再配分し、データベース・パーティション・グループからそのデータベース・パーティションに関する項目をドロップする場合に、使用しなければなりません。
- ***互換性***
  – 以前のバージョンの DB2 との互換性:
    - DBPARTITIONNUM の代わりに NODE を指定できます。
    - DBPARTITIONNUMS の代わりに NODES を指定できます。
    - DATABASE PARTITION GROUP の代わりに NODEGROUP を指定できます。

例

0、1、2、5、7、および 8 というデータベース・パーティションを持つ、 6 つのパーティションのデータベースがあると想定します。 2 つのデータベース・パーティション (3 と 6) をシステムに追加します。

- MAXGROUP という名前のデータベース・パーティション・グループに、データベース・パーティション 3 と 6 を追加し、データベース・パーティション 2 と同種の表スペース・コンテナーを設定するとします。その場合、ステートメントは以下のようになります。

```
ALTER DATABASE PARTITION GROUP MAXGROUP
  ADD DBPARTITIONNUMS (3,6)LIKE DBPARTITIONNUM 2
```

- データベース・パーティション 1 をドロップし、データベース・パーティション 6 をデータベース・パーティション・グループ MEDGROUP に追加するとしま

す。 ALTER TABLESPACE を使用して、データベース・パーティション 6 に対して別個に表スペース・コンテナーを定義します。必要なステートメントは以下のようになります。

```
ALTER DATABASE PARTITION GROUP MEDGROUP
  ADD DBPARTITIONNUM(6)WITHOUT TABLESPACES
  DROP DBPARTITIONNUM(1)
```

CREATE DATABASE PARTITION GROUP

CREATE DATABASE PARTITION GROUP ステートメントは、データベースに新しいデータベース・パーティション・グループを定義し、データベース・パーティションをデータベース・パーティション・グループに割り当て、データベース・パーティション・グループ定義をシステム・カタログに記録します。

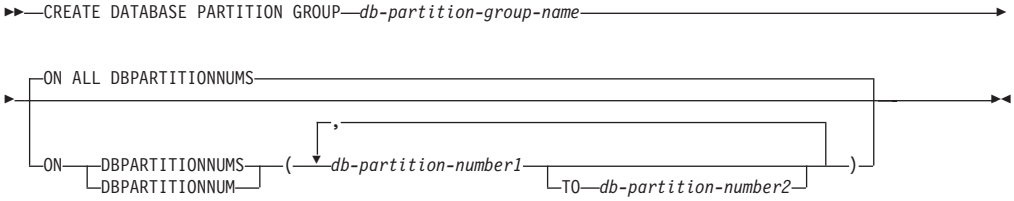
呼び出し方法

このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。 DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

許可

このステートメントの許可 ID が持つ特権には、SYSCTRL または SYSADM 権限が含まれている必要があります。

構文



説明

db-partition-group-name
データベース・パーティション・グループの名前を指定します。これは、1 部構成の名前です。これは、SQL ID です (通常 ID または区切り ID)。 *db-partition-group-name* (データベース・パーティション・グループ名) は、すでにカタログに存在しているデータベース・パーティション・グループを指定するものであってはなりません (SQLSTATE 42710)。 *db-partition-group-name* を文字 'SYS' または 'IBM' で始めることはできません (SQLSTATE 42939)。

ON ALL DBPARTITIONNUMS
データベース・パーティション・グループの作成時に、データベース (db2nodes.cfg ファイル) に定義されているすべてのデータベース・パーティションにわたってデータベース・パーティション・グループを定義することを指定します。

データベース・システムにデータベース・パーティションが追加された場合、ALTER DATABASE PARTITION GROUP ステートメントを実行して、この新しいデータベース・パーティションをデータベース・パーティション・グループ (IBMDEFAULTGROUP を含む) に組み込む必要があります。さらに、REDISTRIBUTE DATABASE PARTITION GROUP コマンドを実行して、そのデータベース・パーティションにデータを移す必要があります。

ON DBPARTITIONNUMS
データベース・パーティション・グループに入れるデータベース・パーティションを指定します。 DBPARTITIONNUM は DBPARTITIONNUMS の同義語です。

db-partition-number1
データベース・パーティション番号を指定します。(前のバージョンとの互換性を保つため、形式 `NODE`*nnnnn* の *node-name* も指定できます。)

TO *db-partition-number2*
データベース・パーティション番号の範囲を指定します。*db-partition-number2* の値は、 *db-partition-number1* の値以上でなければなりません (SQLSTATE 428A9)。指定したデータベース・パーティション番号の範囲 (指定した番号を含む) のすべてのデータベース・パーティションが、データベース・パーティション・グループに入れられます。

規則

- 番号によって指定するそれぞれのデータベース・パーティションは、`db2nodes.cfg` ファイルに定義されていなければなりません (SQLSTATE 42729)。
- ON DBPARTITIONNUMS 節にリストするそれぞれの *db-partition-number* は、同じであってはなりません (SQLSTATE 42728)。
- 有効な *db-partition-number* は、0 から 999 (両端を含む) です (SQLSTATE 42729)。

注

- このステートメントは、データベース・パーティション・グループに対する分散マップを作成します。それぞれの分散マップごとに、分散マップ ID (PMAP_ID) が生成されます。この情報はカタログに記録され、SYSCAT.DBPARTITIONGROUPS と SYSCAT.PARTITIONMAPS から検索することができます。 分散マップのそれぞれの項目は、ハッシュされたすべての行が常駐するターゲット・データベース・パーティションを指定します。単一パーティションのデータベース・パーティション・グループの場合、対応する分散マップの項目は 1 つだけです。複数パーティションのデータベース・パーティション・グループの場合、対応する分散マップには 4096 の項目があり、データベース・パーティション番号がマップ項目にラウンドロビン方式 (デフォルト) で割り当てられます。
- ***互換性***
  - 以前のバージョンの DB2 との互換性:
    - DBPARTITIONNUM の代わりに NODE を指定できます。
    - DBPARTITIONNUMS の代わりに NODES を指定できます。
    - DATABASE PARTITION GROUP の代わりに NODEGROUP を指定できます。

例

0、1、2、5、7、および 8 として定義された 6 つのデータベース・パーティションを持つパーティション・データベースがあると想定します。

- 6 つのデータベース・パーティションすべてに対して、 MAXGROUP という名前のデータベース・パーティション・グループを作成すると想定します。必要なステートメントは以下のようになります。

```
CREATE DATABASE PARTITION GROUP MAXGROUP ON ALL DBPARTITIONNUMS
```

- データベース・パーティション 0、1、2、5、および 8 に対して、 MEDGROUP と呼ばれるデータベース・パーティション・グループを作成すると想定します。必要なステートメントは以下のようになります。

```
CREATE DATABASE PARTITION GROUP MEDGROUP
  ON DBPARTITIONNUMS( 0 TO 2, 5, 8)
```

- データベース・パーティション 7 に対して、単一パーティションのデータベース・パーティション・グループ MINGROUP を作成すると想定します。必要なステートメントは以下のようになります。

```
CREATE DATABASE PARTITION GROUP MINGROUP
  ON DBPARTITIONNUM (7)
```

第 34 章 サポートされる管理 SQL ルーチンおよび管理ビュー

ADMIN_CMD ストアード・プロシージャーおよび関連する管理 SQL ルーチン

ADMIN_CMD プロシージャーを使用する GET STMM TUNING DBPARTITIONNUM コマンド

ユーザー設定のセルフチューニング・メモリー・マネージャー (STMM) の調整データベース・パーティション番号、および現在の STMM 調整データベース・パーティション番号を報告するカタログ表の読み取りに使用します。

許可

SYSADM または DBADM 権限

必要な接続

データベース

コマンド構文

例

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

以下はこの照会の出力例です。

```
Result set 1
  --------------

  USER_PREFERRED_NUMBER CURRENT_NUMBER
  --------------------- --------------
                      2              2

  1 record(s) selected.

  Return Status = 0
```

使用上の注意

ユーザー設定のセルフチューニング・メモリー・マネージャー (STMM) の調整データベース・パーティション番号 (USER_PREFERRED_NUMBER) は、ユーザーにより設定され、メモリー・チューナーの実行対象にするデータベース・パーティションを指定します。データベースの稼働中に、調整パーティションは 1 時間に数度、非同期に更新されます。結果として、戻される CURRENT_NUMBER および USER_PREFERRED_NUMBER は、ユーザー設定の STMM パーティション番号の更新後にも同期がとれていない可能性があります。これを解決するために、

CURRENT_NUMBER が非同期に更新されるのを待機するか、またはデータベースをいったん停止してから開始し、CURRENT_NUMBER の更新を強制します。

ADMIN_CMD プロシージャーを使用する UPDATE STMM TUNING DBPARTITIONNUM コマンド

ユーザー設定のセルフチューニング・メモリー・マネージャー (STMM) の調整データベース・パーティションを更新します。

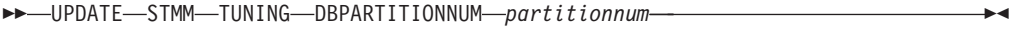
許可

SYSADM または DBADM 権限

必要な接続

データベース

コマンド構文



コマンド・パラメーター

partitionnum

partitionnum は整数です。-1 または存在しないデータベース・パーティション番号が使用される場合、DB2 は STMM メモリー・チューナーを実行する適切なデータベース・パーティションを自動的に選択します。

例

ユーザー設定のセルフチューニング・メモリー・マネージャー (STMM) の調整データベース・パーティションを更新して、データベース・パーティション 3 にします。

```
CALL SYSPROC.ADMIN_CMD( 'update stmm tuning dbpartitionnum 3' )
```

使用上の注意

STMM 調整プロセスは、ユーザー設定の STMM 調整データベース・パーティション番号の値の変更を定期的に検査します。 STMM 調整プロセスは、*partitionnum* が存在しており、それがアクティブなデータベース・パーティションであれば、ユーザー設定の STMM 調整データベース・パーティションに移ります。 このコマンドが STMM 調整データベース・パーティション番号を変更すると、現在の STMM 調整データベース・パーティション番号は即時に変更されます。

コマンドの実行状況は、CALL ステートメントからの結果である SQLCA で戻されます。

このコマンドは、その変更内容を ADMIN_CMD プロシージャーでコミットします。

構成管理 SQL ルーチンおよびビュー

DB_PARTITIONS

DB_PARTITIONS 表関数は、表形式の db2nodes.cfg ファイルの内容を戻します。

構文

スキーマは SYSPROC です。

許可

DB_PARTITIONS 表関数に対する EXECUTE 特権。

表関数パラメーター

関数には入力パラメーターはありません。

例

3 つのロジカル・パーティションを持つデータベースから情報を取り出します。

```
SELECT * FROM TABLE(DB_PARTITIONS()) AS T
```

以下はこの照会の出力例です。

```
PARTITION_NUMBER HOST_NAME                 PORT_NUMBER SWITCH_NAME
---------------- --------------------...- ----------- -----------...-
               0 jessicae.torolab.ibm.com           0 jessicae
               1 jessicae.torolab.ibm.com           1 jessicae
               2 jessicae.torolab.ibm.com           2 jessicae

  3 record(s) selected.
```

戻される情報

表 42. DB_PARTITIONS 表関数によって戻される情報

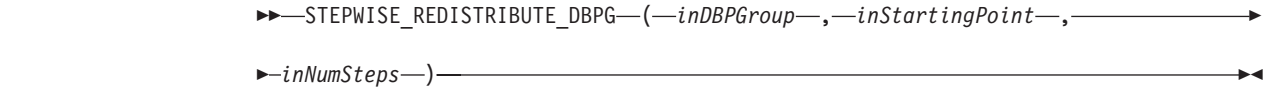
列名	データ・タイプ	説明
PARTITION_NUMBER	SMALLINT	パーティション・データベース環境内のデータベース・パーティション・サーバーを識別する 0 から 999 までの固有番号。
HOST_NAME	VARCHAR(128)	データベース・パーティション・サーバーの TCP/IP ホスト名。
PORT_NUMBER	SMALLINT	データベース・パーティション・サーバーのポート番号。
SWITCH_NAME	VARCHAR(128)	データベース・パーティション通信用の高速相互接続 (スイッチ) の名前。

段階的な再配分管理 SQL ルーチン

STEPWISE_REDISTRIBUTE_DBPG プロシージャー - データベース・パーティション・グループの一部の再配分

STEPWISE_REDISTRIBUTE_DBPG プロシージャーは、このプロシージャーに指定された入力と、SET_SWRD_SETTINGS プロシージャーによって作成または更新された設定ファイルに従って、データベース・パーティション・グループの一部を再配分します。

構文



スキーマは SYSPROC です。

プロシージャー・パラメーター

inDBPGroup
ターゲット・データベース・パーティション・グループの名前を指定する、タイプ VARCHAR (128) の入力引数。

inStartingPoint
使用する開始点を指定する、タイプ SMALLINT の入力引数。このパラメーターが NULL 以外の正の整数に設定されると、STEPWISE_REDISTRIBUTE_DBPG プロシージャーは、設定ファイルで指定された *nextStep* 値を使用しないで、この値を使用します。このオプションは、STEPWISE_REDISTRIBUTE_DBPG プロシージャーを特定のステップから再実行するときに役立ちます。このパラメーターを NULL に設定した場合は、*nextStep* 値が使用されます。

inNumSteps
実行するステップ数を指定する、タイプ SMALLINT の入力引数。このパラメーターが NULL 以外の正の整数に設定されると、STEPWISE_REDISTRIBUTE_DBPG プロシージャーは、設定ファイルで指定された *stageSize* 値を使用しないで、この値を使用します。このオプションは、設定で指定された内容とは異なるステップ数を指定して STEPWISE_REDISTRIBUTE_DBPG プロシージャーを再実行するときに役立ちます。たとえば、スケジュール済みステージに 5 つのステップがあり、再配分処理がステップ 3 で失敗した場合、エラー状態が訂正されたら STEPWISE_REDISTRIBUTE_DBPG プロシージャーを呼び出すことによって、残りの 3 つのステップを実行することができます。このパラメーターを NULL に設定した場合は、*stageSize* 値が使用されます。このプロシージャーに値 -2 を使用することにより、この数が無制限であることを表すことができます。

注: REDISTRIBUTE DATABASE PARTITION GROUP コマンドに **NOT ROLLFORWARD RECOVERABLE** オプションと同等のものを指定するパラメー

ターはありません。 STEPWISE_REDISTRIBUTE_DBPG プロシージャーの使用時には、行データ再配布に対してロギングが必ず実行されます。

許可

- STEPWISE_REDISTRIBUTE_DBPG プロシージャーに対する EXECUTE 特権
- SYSADM、SYSCTRL、または DBADM

例

SET_SWRD_SETTINGS プロシージャーによってレジストリーに保管された再配分プランに従って、データベース・パーティション・グループ「IBMDEFAULTGROUP」を再配分します。データの再配分をステップ 3 から開始して、再配分プランの 2 つのステップを完了します。

```
CALL SYSPROC.STEPWISE_REDISTRIBUTE_DBPG('IBMDEFAULTGROUP', 3, 2)
```

ステップ単位の再配分プロシージャーの詳細な使用例については、『STEPWISE_REDISTRIBUTE_DBPG プロシージャー』を参照してください。

使用上の注意

STEPWISE_REDISTRIBUTE_DBPG プロシージャーの実行開始後に SET_SWRD_SETTINGS プロシージャーを使用して processState のレジストリー値を 1 に更新すると、処理は次のステップの先頭で停止し、警告メッセージが戻されます。

再配分処理で SQL COMMIT ステートメントが呼び出されるため、タイプ 2 接続での再配分処理の実行はサポートされていません。

第 6 部 付録

付録 A. 非ルート・ユーザーとしてのインストール

非ルート・ユーザーとしての DB2 製品のインストール

ほとんどの DB2 データベース製品は、非ルート・ユーザーとしてインストールできます。

始める前に

非ルート・ユーザーとして何らかの DB2 データベース製品をインストールする前に、ルート・インストールと非ルート・インストールの違い、および非ルート・インストールの制限を知っておく必要があります。 **非ルート・インストール**について詳しくは、『非ルート・インストールの概要 (Linux および UNIX)』を参照してください。

非ルート・ユーザーとしての DB2 データベース製品のインストールの前提条件は、以下のとおりです。

- インストール DVD をマウントできるか、あるいはマウントを代行してもらう必要があります。
- DB2 インスタンスの所有者として使用できる正当なユーザー ID を持っている必要があります。

  ユーザー ID には、以下の制限と要件があります。
  - guests、admins、users、および local を除く 1 次グループがなければなりません。
  - 英小文字 (a から z)、数字 (0 から 9)、および下線文字 (_) を使用できます。
  - 長さが 8 文字を超えることはできません。
  - IBM、SYS、SQL、または数字から始まることはできません。
  - DB2 予約語 (USERS、ADMINS、GUESTS、PUBLIC、または LOCAL) あるいは SQL 予約語であってはなりません。
  - DB2 インスタンス ID、DAS ID または fenced ID の root 特権を持つユーザー ID は使用できません。
  - アクセント付き文字は使用できません。
  - 新しいユーザー ID を作成する代わりに既存のユーザー ID を指定する場合は、そのユーザー ID について以下を確認してください。
    - ロックされていない
    - パスワードが有効期限切れでない
- インストールする製品に存在するハードウェアおよびソフトウェア前提条件は、ルート・ユーザーに適用される場合と全く同様に非ルート・ユーザーにも適用されます。
- AIX バージョン 5.3 では、非同期入出力 (AIO) が有効になっている必要があります。

- ホーム・ディレクトリーは、有効な DB2 パスでなければなりません。

  DB2 インストール・パスには、以下の規則があります。
  - 英小文字 (a から z)、英大文字 (A から Z)、および下線文字 (_) を使用できます。
  - 128 文字を超えることはできません。
  - スペースは使用できません。
  - 英語以外の文字は使用できません。

このタスクについて

非ルート・ユーザーとしての DB2 データベース製品のインストールは、非ルート・ユーザーであることを意識せずに行われます。言い換えると、非ルート・ユーザーとしてログインすること以外は、非ルート・ユーザーが DB2 データベース製品をインストールするために特別に行う必要のあることはありません。

手順

非ルート・インストールを実行するには:

1. 非ルート・ユーザーとしてログインします。
2. 使用可能な方法のいずれかを使用して、DB2 データベース製品をインストールします。 以下のオプションがあります。
   - DB2 セットアップ・ウィザード（GUI インストール)
   - db2_install コマンド
   - 応答ファイルを使った db2setup コマンド (サイレント・インストール)

     注: 非ルート・ユーザーは、DB2 データベース製品がインストールされるディレクトリーを選択できないので、応答ファイル内に FILE キーワードがあっても無視されます。
3. DB2 データベース製品がインストールされた後に、非ルート DB2 インスタンスを使用するために、新しいログイン・セッションを開く必要があります。あるいは、*$HOME*/sqllib/db2profile (Bourne シェルおよび Korn シェル・ユーザーの場合) または *$HOME*/sqllib/db2chsrc (C シェル・ユーザーの場合) によって DB2 インスタンス環境を提供する場合は、同じログイン・セッションを使用することができます。ここで、*$HOME* は非ルート・ユーザーのホーム・ディレクトリーです。

次の作業

DB2 データベース製品がインストールされた後に、オペレーティング・システムのユーザー・プロセス・リソース限界 (ulimit) を検査してください。最小 ulimit 値に収まっていない場合、DB2 エンジンは、予期せぬオペレーティング・リソース不足エラーに遭遇する可能性があります。そうしたエラーによって、DB2 の停止にいたる場合があります。

付録 B. バックアップの使用

バックアップの使用

オリジナルで障害または損傷が起こる場合のために、BACKUP DATABASE コマンドを使用してデータベースのデータのコピーを取り、別のメディアに保管します。データベース全体、またはデータベース・パーティションをバックアップすることもでき、選択された表スペースのみをバックアップすることもできます。

始める前に

バックアップを作成しようとしているデータベースに接続する必要はありません。指定したデータベースへの接続はデータベース・バックアップ・ユーティリティーにより自動的に確立され、この接続はバックアップ操作が完了すると終了します。バックアップする予定のデータベースに接続している場合、接続を切断してから BACKUP DATABASE コマンドを発行し、バックアップ操作を進めます。

データベースは、ローカルとリモートのいずれかです。 Tivoli Storage Manager (TSM) または DB2 拡張コピー・サービス (ACS) などのストレージ管理製品を使用していない限り、バックアップ・イメージはデータベース・サーバーに残ります。

オフライン・バックアップを実行する予定であり、ACTIVATE DATABASE ステートメントを使用してデータベースを活動化した場合は、オフライン・バックアップを実行する前にデータベースを非活動化する必要があります。データベースへのアクティブな接続がある場合、データベースの非活動化を成功させるには、SYSADM 権限を持つユーザーがデータベースに接続し、以下のコマンドを発行する必要があります。

```
CONNECT TO database-alias
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;
UNQUIESCE DATABASE;
TERMINATE;
DEACTIVATE DATABASE database-alias
```

パーティション・データベース環境では、 BACKUP DATABASE コマンドを使用してデータベース・パーティションを個別にバックアップすること、 ON DBPARTITIONNUM コマンド・パラメーターを使用してデータベース・パーティションのいくつかを一度にバックアップすること、または ALL DBPARTITIONNUMS パラメーターを使用してデータベース・パーティションの全部を同時にバックアップすることができます。 LIST NODES コマンドを使用して、バックアップするユーザー表のあるデータベース・パーティションを識別できます。

シングル・システム・ビュー (SSV) バックアップを使用しておらず、パーティション・データベース環境でオフライン・バックアップを実行する場合には、他のすべてのデータベース・パーティションとは別にカタログ・パーティションのバックアップを取る必要があります。たとえば、最初にカタログ・パーティションのバックアップを取り、次に、他のデータベース・パーティションのバックアップを取ることができます。この必要がある理由は、バックアップ操作の際にはカタログ・パーティションに対する排他的データベース接続が必要で、その間その他のデータベー

ス・パーティションは接続できないからです。オンライン・バックアップを実行する場合は、すべてのデータベース・パーティション (カタログ・パーティションを含む) のバックアップを同時に取ったり、任意の順序で取ったりできます。

コマンド・エディターを使用してデータベース・パーティションのバックアップを取ることもできます。この方法の場合はロールフォワード・リカバリーはサポートされないので、これらのノード上のデータベースのバックアップを定期的に取ってください。作成する任意のバックアップ・コピーと共に db2nodes.cfg ファイルのコピーも保存する必要があります。これは、このファイルに関する損傷に対する保護となります。

分散要求システムでは、バックアップ操作は、当該データベース・カタログに保管されている分散要求データベースおよびメタデータ (ラッパー、サーバー、ニックネームなど) に適用されます。データ・ソース・オブジェクト (表およびビュー) は、分散要求データベースに保管されていないかぎり、バックアップされません。

過去のリリースのデータベース・マネージャーでデータベースを作成し、そのデータベースをマイグレーションしていない場合は、データベースをマイグレーションしてからでなければバックアップをとることはできません。

このタスクについて

バックアップ・ユーティリティーには、以下の制限が適用されます。

- 別々の表スペースであっても、表スペースのバックアップ操作と表スペースのリストア操作とを同時に実行することはできません。
- パーティション・データベース環境でロールフォワード・リカバリーを使用できるようにする場合は、定期的にノード・リストについてデータベースのバックアップをとる必要があり、また、システム内の残りのノードのバックアップ・イメージも少なくとも 1 つは作成する必要があります (該当するデータベースに関するユーザー・データを含んでいない場合でも)。データベースに関するユーザー・データを含んでいないデータベース・パーティション・サーバーで、データベース・パーティションのバックアップ・イメージが必要となるのは、次の 2 つの場合です。
  - 最後のバックアップを作成した後にデータベース・システムにデータベース・パーティション・サーバーを追加し、このデータベース・パーティション・サーバーについて順方向リカバリーを実行する必要がある場合。
  - 特定時点のリカバリーを使用する場合。この場合は、システム内のすべてのデータベース・パーティションがロールフォワード・ペンディング状態でなければなりません。
- DMS 表スペースのオンライン・バックアップ操作は、以下の操作との互換はありません。
  - ロード
  - 再編成 (オンラインおよびオフライン)
  - 表スペースのドロップ
  - 表の切り捨て
  - 索引作成

  - NOT LOGGED INITIALLY (CREATE TABLE および ALTER TABLE ステートメントと共に使用)
- 現在アクティブになっているデータベースのオフライン・バックアップを実行しようとすると、エラーを受け取ります。オフライン・バックアップを実行する前に、DEACTIVATE DATABASE コマンドを発行してデータベースがアクティブではないことを確認してください。

バックアップ・ユーティリティーは、コマンド行プロセッサー (CLP)、コントロール・センターにある「データベースのバックアップ」ウィザード、BACKUP DATABASEパラメーターを使用した ADMIN_CMD プロシージャーの実行、または *db2Backup* アプリケーション・プログラミング・インターフェース (API) を通して起動できます。

CLP によって発行する BACKUP DATABASE コマンドの例を以下に示します。

```
db2 backup database sample to c:¥DB2Backups
```

手順

「データベースのバックアップ」ウィザードをオープンするには、次のようにします。

1. コントロール・センターから、バックアップするデータベースまたは表スペース・オブジェクトが見つかるまでオブジェクト・ツリーを展開します。
2. オブジェクトを右クリックして、ポップアップ・メニューから「バックアップ」を選択します。「データベースのバックアップ」ウィザードがオープンします。

次の作業

詳細情報は、コントロール・センター内のコンテキスト・ヘルプ機能を使用して入手できます。

オフライン・バックアップを実行した場合は、バックアップの完了後に、次のようにしてデータベースを再び活動化させる必要があります。

```
ACTIVATE DATABASE sample
```

付録 C. パーティション・データベース環境カタログ・ビュー

SYSCAT.BUFFERPOOLDBPARTITIONS

各行は、バッファー・プールとデータベース・パーティションの組み合わせのうち、パーティション上のバッファー・プールのサイズが、同じデータベース・パーティション・グループ内の他のパーティションにおけるバッファー・プールのデフォルト・サイズ (SYSCAT.BUFFERPOOLS で表記されている) と異なっているものを表します。

表 43. SYSCAT.BUFFERPOOLDBPARTITIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
BUFFERPOOLID	INTEGER		内部バッファー・プール ID。
DBPARTITIONNUM	SMALLINT		データベース・パーティション番号。
NPAGES	INTEGER		このデータベース・パーティションに対するバッファー・プールのページ数。

SYSCAT.DATAPARTITIONEXPRESSION

各行は、表パーティション・キーの一部に入る式を表します。

表 44. SYSCAT.DATAPARTITIONEXPRESSION カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
TABSCHEMA	VARCHAR(128)		パーティション化された表のスキーマ名。
TABNAME	VARCHAR(128)		パーティション化された表の非修飾名。
DATAPARTITIONKEYSEQ	INTEGER		式キー部分のシーケンス ID (1 から始まる)。
DATAPARTITIONEXPRESSION	CLOB (32K)		シーケンス内のこの項目の式 (SQL 構文)。
NULLSFIRST	CHAR (1)		• N = この式で NULL 値を比較するときには高いものとして扱う • Y = この式で NULL 値を比較するときには低いものとして扱う

SYSCAT.DATAPARTITIONS

各行はデータ・パーティションを表します。

表 45. SYSCAT.DATAPARTITIONS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
DATAPARTITIONNAME	VARCHAR(128)		データ・パーティションの名前。
TABSCHEMA	VARCHAR(128)		このデータ・パーティションが属する表のスキーマ名。

表 45. SYSCAT.DATAPARTITIONS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
TABNAME	VARCHAR(128)		このデータ・パーティションが属する表の非修飾名。
DATAPARTITIONID	INTEGER		データ・パーティションの ID。
TBSPACEID	INTEGER	Y	このデータ・パーティションが保管されている表スペースの ID。STATUS が 'I' の場合、NULL。
PARTITIONOBJECTID	INTEGER	Y	表スペース内のデータ・パーティションの ID。
LONG_TBSPACEID	INTEGER	Y	長形式データが保管されている表スペースの ID。 STATUS が 'I' の場合、NULL。
ACCESS_MODE	CHAR (1)		データ・パーティションのアクセス制限の状態。これらの状態は、SET INTEGRITY によりペンディング状態にあるオブジェクト、または SET INTEGRITY ステートメントによって処理されたオブジェクトにのみ適用されます。可能な値は以下のとおりです。 • D = データの移動不可 • F = フル・アクセス権限 • N = アクセス不可 • R = 読み取り専用アクセス
STATUS	VARCHAR (32)		• A = データ・パーティションは新規にアタッチされた • D = データ・パーティションはデタッチされた • I = 非同期の索引クリーンアップ中にのみカタログ内の項目が維持される、デタッチされたデータ・パーティション。デタッチされたパーティションを参照するすべての索引レコードの削除時に、STATUS 値が 'I' の行は除去される。 • 空ストリング = データ・パーティションは表示できる (通常の状況)。 バイト 2 から 32 は、将来の使用のために予約されています。
SEQNO	INTEGER		データ・パーティションのシーケンス番号 (0 から始まる)。
LOWINCLUSIVE	CHAR (1)		• N = 低い方のキー値はそれ自身を含まない • Y = 低い方のキー値はそれ自身を含む
LOWVALUE	VARCHAR(512)		このデータ・パーティションの低い方のキー値 (SQL 値のストリング表記)。
HIGHINCLUSIVE	CHAR (1)		• N = 高い方のキー値はそれ自身を含まない • Y = 高い方のキー値はそれ自身を含む

表 45. SYSCAT.DATAPARTITIONS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
HIGHVALUE	VARCHAR(512)		このデータ・パーティションの高い方のキー値 (SQL 値のストリング表記)。

SYSCAT.DBPARTITIONGROUPDEF

各行は、データベース・パーティション・グループに属するデータベース・パーティションを表します。

表 46. SYSCAT.DBPARTITIONGROUPDEF カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
DBPGNAME	VARCHAR(128)		データベース・パーティションの入ったデータベース・パーティション・グループの名前。
DBPARTITIONNUM	SMALLINT		データベース・パーティション・グループに属するデータベース・パーティションのパーティション番号。有効なパーティション番号は、0 以上 999 以下の間です。
IN_USE	CHAR (1)		データベース・パーティションの状況。 • A = 新しく追加されたデータベース・パーティションは分散マップに入っていないが、データベース・パーティション・グループ内の表スペースにコンテナーが作成された。データベース・パーティションは、データベース・パーティション・グループ再配分操作が正常に完了したときに分散マップに追加される。 • D = データベース・パーティションは、データベース・パーティション・グループ再配分操作が正常に完了したときにドロップされる。 • T = 新しく追加されたデータベース・パーティションは、分散マップに入っておらず、WITHOUT TABLESPACES 節を使用して追加された。データベース・パーティション・グループの表スペースにコンテナーを追加する必要がある。 • Y = データベース・パーティションは分散マップに入っている。

SYSCAT.DBPARTITIONGROUPS

各行はデータベース・パーティション・グループを表します。

表 47. SYSCAT.DBPARTITIONGROUPS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
DBPGNAME	VARCHAR(128)		データベース・パーティション・グループの名前。
OWNER	VARCHAR(128)		データベース・パーティション・グループ作成時の許可 ID。
OWNERTYPE	CHAR (1)		• S = 所有者はシステム • U = 所有者は個々のユーザー
PMAP_ID	SMALLINT		SYSCAT.PARTITIONMAPS カタログ・ビュー内の分散マップの ID。
REDISTRIBUTE_PMAP_ID	SMALLINT		現在再配分のために使用されている分散マップの ID。再配分が現在進行中でない場合、-1。
CREATE_TIME	TIMESTAMP		データベース・パーティション・グループの作成時刻。
DEFINER[1]	VARCHAR(128)		データベース・パーティション・グループ作成時の許可 ID。
REMARKS	VARCHAR (254)	Y	ユーザー提供のコメントまたは NULL 値。

注:

1. DEFINER 列は、後方互換性のために含まれています。OWNER を参照してください。

SYSCAT.PARTITIONMAPS

各行は、表の分散キーのハッシュに基づいて、データベース・パーティション・グループ内のパーティションの間で表の行を分散するために使用される分散マップを表します。

表 48. SYSCAT.PARTITIONMAPS カタログ・ビュー

列名	データ・タイプ	NULL 可能	説明
PMAP_ID	SMALLINT		分散マップの ID。
PARTITIONMAP	BLOB (8192)		分散マップ。複数のパーティション・データベースのデータベース・パーティション・グループの場合は、4096 個の 2 バイト整数から成るベクトル。単一のパーティション・データベースのデータベース・パーティション・グループの場合は、単一パーティションのパーティション番号を示す項目が 1 つあります。

付録 D. DB2 技術情報の概説

DB2 技術情報は、以下のツールと方法を介して利用できます。

- *DB2 インフォメーション・センター*
  - トピック (タスク、概念、およびリファレンス・トピック)
  - DB2 ツールのヘルプ
  - サンプル・プログラム
  - チュートリアル
- DB2 資料
  - PDF ファイル (ダウンロード可能)
  - PDF ファイル (DB2 PDF DVD に含まれる)
  - 印刷資料
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ

注: *DB2 インフォメーション・センター* のトピックは、PDF やハードコピー資料よりも頻繁に更新されます。最新の情報を入手するには、資料の更新が発行されたときにそれをインストールするか、ibm.com® にある *DB2 インフォメーション・センター* を参照してください。

技術資料、ホワイト・ペーパー、IBM Redbooks® 資料などのその他の DB2 技術情報には、オンライン (ibm.com) でアクセスできます。 DB2 Information Management ソフトウェア・ライブラリー・サイト (http://www.ibm.com/software/data/sw-library/) にアクセスしてください。

資料についてのフィードバック

DB2 の資料についてのお客様からの貴重なご意見をお待ちしています。DB2 の資料を改善するための提案については、db2docs@ca.ibm.com まで E メールを送信してください。DB2 の資料チームは、お客様からのフィードバックすべてに目を通しますが、直接お客様に返答することはありません。お客様が関心をお持ちの内容について、可能な限り具体的な例を提供してください。特定のトピックまたはヘルプ・ファイルについてのフィードバックを提供する場合は、そのトピック・タイトルおよび URL を含めてください。

DB2 お客様サポートに連絡する場合には、この E メール・アドレスを使用しないでください。資料を参照しても、DB2 の技術的な問題が解決しない場合は、お近くの IBM サービス・センターにお問い合わせください。

IBM Information Management 製品の使用をより容易にするために IBM にご協力いただける場合は、コンシューマビリティーに関する次のアンケートにご回答ください。http://www.ibm.com/software/data/info/consumability-survey/

DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

以下の表は、DB2 ライブラリーについて説明しています。DB2 ライブラリーに関する詳細な説明については、www.ibm.com/shop/publications/order にある IBM Publications Center にアクセスしてください。英語の DB2 バージョン 9.5 のマニュアル (PDF 形式) とその翻訳版は、 www.ibm.com/support/docview.wss?rs=71&uid=swg2700947 からダウンロードできます。

この表には印刷資料が入手可能かどうかが示されていますが、国または地域によっては入手できない場合があります。

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

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

表 49. DB2 の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
管理 API リファレンス	SC88-4431-02	入手可能	2009 年 4 月
管理ルーチンおよびビュー	SC88-4435-02	入手不可	2009 年 4 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 1 巻	SC88-4433-02	入手可能	2009 年 4 月
コール・レベル・インターフェース ガイドおよびリファレンス 第 2 巻	SC88-4434-02	入手可能	2009 年 4 月
コマンド・リファレンス	SC88-4432-02	入手可能	2009 年 4 月
データ移動ユーティリティー ガイドおよびリファレンス	SC88-4421-02	入手可能	2009 年 4 月
データ・リカバリーと高可用性 ガイドおよびリファレンス	SC88-4423-02	入手可能	2009 年 4 月
データ・サーバー、データベース、およびデータベース・オブジェクトのガイド	SC88-4259-02	入手可能	2009 年 4 月
データベース・セキュリティー・ガイド	SC88-4418-02	入手可能	2009 年 4 月
ADO.NET および OLE DB アプリケーションの開発	SC88-4425-02	入手可能	2009 年 4 月

表 49. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
組み込み SQL アプリケーションの開発	SC88-4426-02	入手可能	2009 年 4 月
Java アプリケーションの開発	SC88-4427-02	入手可能	2009 年 4 月
Perl および PHP アプリケーションの開発	SC88-4428-02	入手不可	2009 年 4 月
SQL および外部ルーチンの開発	SC88-4429-02	入手可能	2009 年 4 月
データベース・アプリケーション開発の基礎	GC88-4430-02	入手可能	2009 年 4 月
DB2 インストールおよび管理 概説 (Linux および Windows 版)	GC88-4439-02	入手可能	2009 年 4 月
国際化対応ガイド	SC88-4420-02	入手可能	2009 年 4 月
メッセージ・リファレンス 第 1 巻	GI88-4109-01	入手不可	2009 年 4 月
メッセージ・リファレンス 第 2 巻	GI88-4110-01	入手不可	2009 年 4 月
マイグレーション・ガイド	GC88-4438-02	入手可能	2009 年 4 月
Net Search Extender 管理およびユーザーズ・ガイド	SC88-4630-02	入手可能	2009 年 4 月
パーティションおよびクラスタリングのガイド	SC88-4419-02	入手可能	2009 年 4 月
Query Patroller 管理およびユーザーズ・ガイド	SC88-4611-01	入手可能	2009 年 4 月
IBM データ・サーバー・クライアント機能 概説およびインストール	GC88-4441-02	入手不可	2009 年 4 月
DB2 サーバー機能 概説およびインストール	GC88-4440-02	入手可能	2009 年 4 月
Spatial Extender および Geodetic Data Management Feature ユーザーズ・ガイドおよびリファレンス	SC88-4629-02	入手可能	2009 年 4 月
SQL リファレンス 第 1 巻	SC88-4436-02	入手可能	2009 年 4 月
SQL リファレンス 第 2 巻	SC88-4437-02	入手可能	2009 年 4 月

表 49. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
システム・モニター ガイドおよびリファレンス	SC88-4422-02	入手可能	2009 年 4 月
Text Search ガイド	SC88-4424-01	入手可能	2009 年 4 月
問題判別ガイド	GI88-4108-02	入手不可	2009 年 4 月
データベース・パフォーマンスのチューニング	SC88-4417-02	入手可能	2009 年 4 月
Visual Explain チュートリアル	SC88-4449-00	入手不可	
新機能	SC88-4445-02	入手可能	2009 年 4 月
ワークロード・マネージャー ガイドおよびリファレンス	SC88-4446-02	入手可能	2009 年 4 月
pureXML ガイド	SC88-4447-02	入手可能	2009 年 4 月
XQuery リファレンス	SC88-4448-02	入手不可	2009 年 4 月

表 50. DB2 Connect 固有の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2® Connect™ Personal Edition 概説およびインストール	GC88-4443-02	入手可能	2009 年 4 月
DB2 Connect サーバー機能 概説およびインストール	GC88-4444-02	入手可能	2009 年 4 月
DB2 Connect ユーザーズ・ガイド	SC88-4442-02	入手可能	2009 年 4 月

表 51. Information Integration の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
Information Integration: フェデレーテッド・システム 管理ガイド	SC88-4166-01	入手可能	2008 年 3 月
Information Integration: レプリケーションおよびイベント・パブリッシングのための ASNCLP プログラム・リファレンス	SC88-4167-02	入手可能	2008 年 3 月

表 51. Information Integration の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
Information Integration: フェデレーテッド・データ・ソース 構成ガイド	SC88-4185-01	入手不可	
Information Integration: SQL レプリケーション ガイドおよびリファレンス	SC88-4168-01	入手可能	2008 年 3 月
Information Integration: レプリケーションとイベント・パブリッシング 概説	GC88-4187-01	入手可能	2008 年 3 月

DB2 の印刷資料の注文方法

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

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

注: 最新で完全な DB2 資料は、DB2 インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2luw/v9r5) で参照することができます。

DB2 の印刷資料は以下の方法で注文することができます。

- 日本 IBM 発行のマニュアルはインターネット経由でご購入いただけます。詳しくは http://www.ibm.com/shop/publications/order の「ご注文について」をご覧ください。資料の注文情報にアクセスするには、お客様の国、地域、または言語を選択してください。その後、各ロケーションにおける注文についての指示に従ってください。
- DB2 の印刷資料を IBM 営業担当員に注文するには、以下のようにします。
  1. 以下の Web サイトのいずれかから、営業担当員の連絡先情報を見つけてください。
     - IBM Directory of world wide contacts (www.ibm.com/planetwide)
     - IBM Publications Web サイト (http://www.ibm.com/shop/publications/order) 国、地域、または言語を選択し、お客様の所在地に該当する Publications ホ

ーム・ページにアクセスしてください。このページから、「このサイトについて」のリンクにアクセスしてください。

2. 電話をご利用の場合は、DB2 資料の注文であることをご指定ください。
3. 担当者に、注文する資料のタイトルと資料番号をお伝えください。タイトルと資料番号は、 474 ページの『DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)』でご確認いただけます。

コマンド行プロセッサーから SQL 状態ヘルプを表示する

DB2 は、SQL ステートメントの結果の原因になったと考えられる条件の SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

SQL 状態ヘルプを呼び出すには、コマンド行プロセッサーを開いて以下のように入力します。

```
? sqlstate or ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。
例えば、`? 08003` を指定すると SQL 状態 08003 のヘルプが表示され、 `? 08` を指定するとクラス・コード 08 のヘルプが表示されます。

異なるバージョンの DB2 インフォメーション・センターへのアクセス

DB2 バージョン 9.5 のトピックを扱っている DB2 インフォメーション・センターの URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/です。

DB2 バージョン 9 のトピックを扱っている DB2 インフォメーション・センターの URL は http://publib.boulder.ibm.com/infocenter/db2luw/v9/です。

DB2 バージョン 8 のトピックについては、バージョン 8 のインフォメーション・センターの URL http://publib.boulder.ibm.com/infocenter/db2luw/v8/にアクセスしてください。

DB2 インフォメーション・センターでの希望する言語でのトピックの表示

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

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

注: 言語を追加しても、特定の言語でトピックを表示するのに必要なフォントがコンピューターに備えられているとはかぎりません。

– リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「**上に移動**」ボタンをクリックします。

3. ブラウザー・キャッシュを消去してから、ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。

- Firefox または Mozilla Web ブラウザーの場合に、希望する言語でトピックを表示するには、以下のようにします。
  1. 「**ツール**」 -> 「**オプション**」 -> 「**詳細**」 ダイアログの「**言語**」セクションにあるボタンを選択します。「設定」ウィンドウに「言語」パネルが表示されます。
  2. 該当する言語が、言語リストの先頭の項目に指定されていることを確認します。
     – リストに新しい言語を追加するには、「**追加...**」ボタンをクリックしてから、「言語を追加」ウィンドウで言語を選択します。
     – リストの先頭に新しい言語を移動するには、その言語を選択してから、その言語が言語リストに先頭に行くまで「**上に移動**」ボタンをクリックします。
  3. ブラウザー・キャッシュを消去してから、ページを最新表示します。希望する言語で DB2 インフォメーション・センターが表示されます。

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

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

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

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

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

   注: ご使用の環境において、インターネットに接続されていないマシンに *DB2 インフォメーション・センター* の更新をインストールする必要がある場合は、

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

3. スタンドアロンのインフォメーション・センターを停止し、コンピューター上の *DB2 インフォメーション・センター* を再始動します。

注: Windows Vista の場合、下記のコマンドは管理者として実行する必要があります。完全な管理者特権でコマンド・プロンプトまたはグラフィカル・ツールを起動するには、ショートカットを右クリックしてから、 「**管理者として実行**」を選択します。

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

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

     ```
     /etc/init.d/db2icdv95 stop
     ```
2. インフォメーション・センターをスタンドアロン・モードで開始します。
   - Windows の場合:
     a. コマンド・ウィンドウを開きます。
     b. インフォメーション・センターがインストールされているパスにナビゲートします。*DB2 インフォメーション・センター* は、デフォルトで *Program_files*¥IBM¥DB2 Information Center¥Version 9.5 ディレクトリーにインストールされます。ここで、*Program_files* は Program Files ディレクトリーのロケーションを表します。
     c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。
     d. 次のように help_start.bat ファイルを実行します。

        ```
        help_start.bat
        ```
   - Linux の場合:
     a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2 インフォメーション・センター* は /opt/ibm/db2ic/V9.5 ディレクトリーにインストールされています。
     b. インストール・ディレクトリーから doc/bin ディレクトリーにナビゲートします。
     c. 次のように help_start スクリプトを実行します。

        ```
        help_start
        ```

システムのデフォルト Web ブラウザーが起動し、スタンドアロンのインフォメーション・センターが表示されます。

3. 「**更新**」ボタン () をクリックします。インフォメーション・センターの右側のパネルで、「**更新の検索 (Find Updates)**」をクリックします。 既存の文書に対する更新のリストが表示されます。
4. インストール・プロセスを開始するには、インストールする更新をチェックして選択し、「**更新のインストール**」をクリックします。
5. インストール・プロセスが完了したら、「**完了**」をクリックします。
6. 次のようにして、スタンドアロンのインフォメーション・センターを停止します。
   - Windows の場合は、インストール・ディレクトリーの doc¥bin ディレクトリーにナビゲートしてから、次のように help_end.bat ファイルを実行します。

     ```
     help_end.bat
     ```

     注: help_end バッチ・ファイルには、help_start バッチ・ファイルを使用して開始したプロセスを安全に終了するのに必要なコマンドが含まれています。help_start.bat は、Ctrl-C や他の方法を使用して終了しないでください。
   - Linux の場合は、インストール・ディレクトリーの doc/bin ディレクトリーにナビゲートしてから、次のように help_end スクリプトを実行します。

     ```
     help_end
     ```

     注: help_end スクリプトには、help_start スクリプトを使用して開始したプロセスを安全に終了するのに必要なコマンドが含まれています。他の方法を使用して、help_start スクリプトを終了しないでください。
7. *DB2 インフォメーション・センター* を再始動します。
   - Windows では、「**スタート**」 → 「**コントロール パネル**」 → 「**管理ツール**」 → 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**開始**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv95 start
     ```

更新された *DB2 インフォメーション・センター* に、更新された新しいトピックが表示されます。

DB2 チュートリアル

DB2 チュートリアルは、DB2 製品のさまざまな機能について学習するのを支援します。この演習をとおして段階的に学習することができます。

はじめに

インフォメーション・センター (http://publib.boulder.ibm.com/infocenter/db2help/) から、このチュートリアルの XHTML 版を表示できます。

演習の中で、サンプル・データまたはサンプル・コードを使用する場合があります。個々のタスクの前提条件については、チュートリアルを参照してください。

DB2 チュートリアル

チュートリアルを表示するには、タイトルをクリックします。

「*pureXML ガイド* 」の『**pureXML™**』
XML データを保管し、ネイティブ XML データ・ストアに対して基本的な操作を実行できるように、DB2 データベースをセットアップします。

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

DB2 トラブルシューティング情報

DB2 データベース製品を使用する際に役立つ、トラブルシューティングおよび問題判別に関する広範囲な情報を利用できます。

DB2 ドキュメンテーション
トラブルシューティング情報は、「DB2 問題判別ガイド」、またはDB2 インフォメーション・センターの『データベースの基本』セクションにあります。ここには、DB2 診断ツールおよびユーティリティーを使用して、問題を切り分けて識別する方法、最も頻繁に起こる幾つかの問題に対するソリューションについての情報、および DB2 データベース製品を使用する際に発生する可能性のある問題の解決方法についての他のアドバイスがあります。

DB2 Technical Support の Web サイト
現在問題が発生していて、考えられる原因とソリューションを検索したい場合は、DB2 Technical Support の Web サイトを参照してください。
Technical Support サイトには、最新の DB2 資料、TechNotes、プログラム診断依頼書 (APAR またはバグ修正)、フィックスパック、およびその他のリソースへのリンクが用意されています。この知識ベースを活用して、問題に対する有効なソリューションを探し出すことができます。

DB2 Technical Support の Web サイト (http://www.ibm.com/software/data/db2/support/db2_9/) にアクセスしてください。

ご利用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。 ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

付録 E. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

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

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

本書は、IBM 以外の Web サイトおよびリソースへのリンクまたは参照を含む場合があります。 IBM は、本書より参照もしくはアクセスできる、または本書からリンクされた IBM 以外の Web サイトもしくは第三者のリソースに対して一切の責任を負いません。 IBM 以外の Web サイトにリンクが張られていることにより IBM が当該 Web サイトを推奨するものではなく、またその内容、使用もしくはサイトの所有者について IBM が責任を負うことを意味するものではありません。 また、IBM は、お客様が IBM Web サイトから第三者の存在を知ることになった場合にも (もしくは、IBM Web サイトから第三者へのリンクを使用した場合にも)、お客様と第三者との間のいかなる取引に対しても一切責任を負いません。 従って、お客様は、IBM が上記の外部サイトまたはリソースの利用について責任を負うものではなく、また、外部サイトまたはリソースからアクセス可能なコンテンツ、サービス、

製品、またはその他の資料一切に対して IBM が責任を負うものではないことを承諾し、同意するものとします。 第三者により提供されるソフトウェアには、そのソフトウェアと共に提供される固有の使用条件が適用されます。

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

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

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

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

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

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

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

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

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

著作権使用許諾:

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

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

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

商標

IBM、IBM ロゴ、ibm.com は、International Business Machines Corporation 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。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml の「Copyright and trademark information」をご覧ください。

以下は、それぞれ各社の商標または登録商標です。

- Linux は、Linus Torvalds の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標です。
- UNIX は The Open Group の米国およびその他の国における登録商標です。
- Intel、Intel (ロゴ)、Intel Inside、Intel Inside (ロゴ)、Intel Centrino、Intel Centrino (ロゴ)、Celeron、Intel Xeon、Intel SpeedStep、Itanium、Pentium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。
- Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

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です。

索引

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

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[数字]

A

B

C

D

F

G

H

I

L

M

N

R

S

U

V

W

Printed in Japan

SC88-4419-02

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

Spine information:

DB2 Version 9.5 for Linux, UNIX, and Windows

パーティションおよびクラスタリングのガイド