IBM DB2 10.1
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

2013 年 1 月更新版

SA88-4697-01
(英文原典：SC27-3882-01)

IBM DB2 10.1
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

2013 年 1 月更新版

SA88-4697-01
(英文原典：SC27-3882-01)

ご注意

本書および本書で紹介する製品をご使用になる前に、513 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。

原典： SC27-3882-01
IBM DB2 10.1
for Linux, UNIX, and Windows
Partitioning and Clustering Guide
Updated January, 2013

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 2012.12

目次

本書について

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 データベース・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

本書は主に、ローカルまたはリモート・クライアントによってアクセスされるパーティション・データベースやクラスター・データベースを設計、インプリメント、および保守する必要のある、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	*データベース: 管理の概念および構成リファレンス* *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	*データベース: 管理の概念および構成リファレンス* *DB2 サーバー機能 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* *データベース: 管理の概念および構成リファレンス* *DB2 バージョン 10.1 へのアップグレード* *データベースのモニタリング ガイドおよびリファレンス* *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	*データベース: 管理の概念および構成リファレンス* *問題判別およびデータベース・パフォーマンスのチューニング* *Visual Explain チュートリアル*
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、PHP、Python および Ruby on Rails 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*

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 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 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 オペレーティング・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 オペレーティング・システムでのインストール済み環境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、データベース・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 root ユーザーとしてのインストール』では、DB2 データベース製品を非 root ユーザーとして UNIX および Linux オペレーティング・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。

強調表記規則

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のある名前または値 (変数) • 一般的な強調 • 新しい用語の紹介 • 他の情報源の参照
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。

このタスクについて

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。db2nodes.cfg ファイルはデータベース・パーティション・サーバー構成ファイルです。

Windows オペレーティング・システム環境の場合を除いて、データベース・パーティション・サーバー構成ファイル (db2nodes.cfg) の内容を表示および更新するために任意のエディターを使用できます。 Windows オペレーティング・システム環境では、**db2ncrt** および **db2nchg** コマンドを使ってデータベース・パーティション・サーバー構成ファイルを作成および変更します。

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有されます。

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 **start_stop_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・パーティションにまたがってデータを柔軟に拡散させる操作を可能にします。

ユーザーは分散キーを宣言することにより、データを分散する方法を選択できます。また、データの保管場所となるデータベース・パーティション・グループと表スペースを選択することにより、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。各ローカル・データベース・パーティションでは、そこに保管されるデータのローカル索引を作成して、パフォーマンスよくローカル・データにアクセスできます。

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。(再分散中に、データベース・パーティション・グループ内の表は異なる分散マップを使用する可能性があります。これらは連結されません。)
- 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、指定された 1 つ以上のデータベース・パーティションのセットです。

複数のデータベース・パーティションが含まれるデータベース・パーティション・グループを*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。

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ます。

- データベース・パーティション・グループ 1 には、1 つを除きすべてのデータベース・パーティションが含まれています。
- データベース・パーティション・グループ 2 には、データベース・パーティションが 1 つ含まれています。
- データベース・パーティション・グループ 3 には、2 つのデータベース・パーティションが含まれています。
- グループ 2 のデータベース・パーティションは、グループ 1 によって共有されて (および、オーバーラップして) います。
- グループ 3 の 1 つのデータベース・パーティションは、グループ 1 によって共有されて (および、オーバーラップして) います。

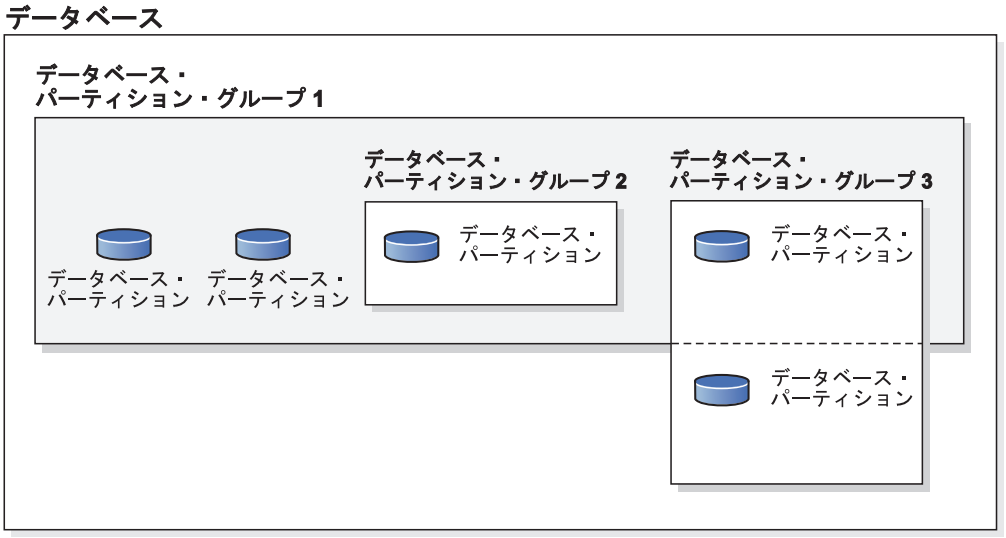


図 1. データベース内のデータベース・パーティション・グループ

データベースが作成されるときに、db2nodes.cfg という名前のデータベース・パーティション*構成ファイル* に指定されているすべてのデータベース・パーティションも同時に作成されます。**ADD DBPARTITIONNUM** または **DROP DBPARTITIONNUM VERIFY** コマンドを使用して、その他のデータベース・パーティションを追加または削除することができます。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・パーティション間で分割されます。

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* がそのグループに関連付けられます。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行を保管するかを決定するために、データベース・マネージャーは分散マップを*分散キー* およびハッシュ・アルゴリズムと共に使用します。

デフォルト・データベース・パーティション・グループ

データベース作成時に、以下の 3 つのデータベース・パーティション・グループが自動的に定義されます。

- 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 (ユーザー表と索引を保持します)。宣言済み一時表または作成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・グループの表スペース

表スペースが、(CREATE TABLESPACE ステートメントの実行時に) 複数パーティションのデータベース・パーティション・グループに関連付けられている場合、その表スペース内のすべての表は、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、パーティション化されます。あるデータベース・パーティション・グループに関連付けられている表スペースを、後から別のデータベース・パーティション・グループに関連付けることはできません。

データベース・パーティション・グループの作成

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。このステートメントは、以下のアクションも実行します。

- 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。
- 分散マップ ID を生成します。
- 以下のカタログ・ビューにレコードを挿入します。
  - SYSCAT.DBPARTITIONGROUPDEF
  - SYSCAT.DBPARTITIONGROUPS
  - SYSCAT.PARTITIONMAPS

データベース・パーティション・グループの変更

データベース・パーティション・グループでデータベース・パーティションを追加したりドロップしたりするには、ALTER DATABASE PARTITION GROUP ステートメントを使用します。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後、データベース・パーティション・グループにおけるデータベース・パーティション・セットの中でデータを再配分するために **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・グループの設計の考慮事項

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。より効率的な結合ストラテジーを使用するために、データベース・マネージャーはコロケーションによって連結された表を活用できます。このような表は、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、それらの表はコロケーションによって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。

中間的なサイズの表を、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てください。例えば、100 MB の表の場合、32 パーティションから成るデータベース・パーティション・グループよりも、16 パーティションから成るデータベース・パーティション・グループの方がパフォーマンスが良くなる可能性があります。

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) の表と分離します。これによって、OLTP トランザクションのパフォーマンス低下を防ぐことができます。

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、索引が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。
- 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・グループに含まれる場合があるため、各データベース・パーティションに固有の番号を割り当てる必要があります。
- 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。 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。

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、必要なデータがある場所を認識している必要があります。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 32 768 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・パーティションがマップ全体にわたって均等に配分されるような方法で、順々に使用されます。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、(0 から 3 までの番号が付けられている) 4 つのデータベース・パーティション上に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 1 および 2 を使ってデータベース内にデータベース・パーティション・グループが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次のようになります。

```
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 32 767 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。

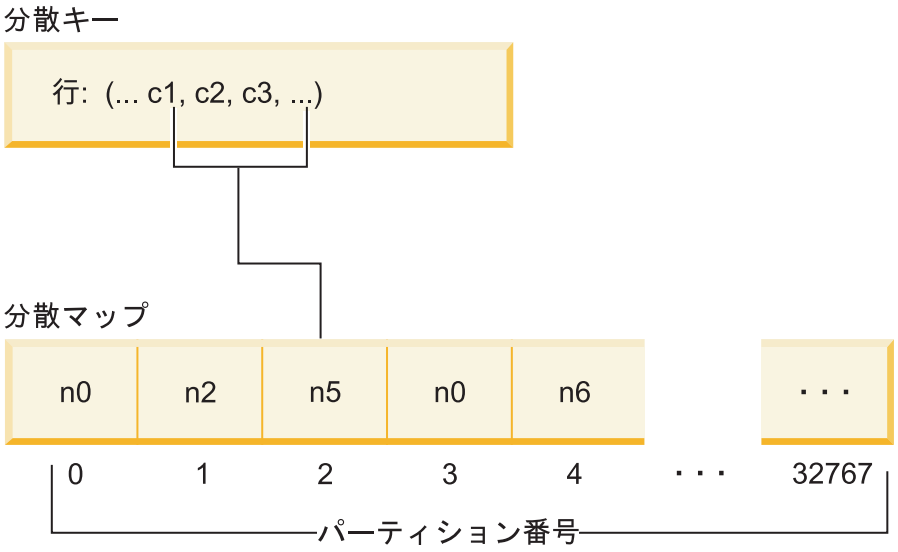


図 2. 分散マップを使用したデータ分散

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。データベース内の複数のデータベース・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できます。このユーティリティーによって、データ配分のリバランスを行ったり、データ配分に歪度を導入したりすることができます。

db2GetDistMap API を使用すると、表示可能な分散マップのコピーを取得できます。分散情報を取得するために sqlugtpi API を引き続き使用した場合、この API は 4096 個の項目を含む分散マップだけを取得できるため、エラー・メッセージ SQL2768N が戻される可能性があります。

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・システムで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。分布が不均一なデータを持つ列、または異なる値の数が少ない列を分散キーに選択しないでください。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。不必要な列を分散キーの中に含めないでください。

分散キーを定義するときには、以下の点を考慮してください。

- BLOB、CLOB、DBCLOB、LONG VARCHAR、LONG VARGRAPHIC、XML、または構造化データ型だけを含む複数パーティション表の作成はサポートされていません。
- 分散キーの定義は変更できません。
- 最も頻繁に結合される列を分散キーに含めます。
- GROUP BY 節に頻繁に含まれる列を分散キーに含めます。

- 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、等号述部を使用して、分散キーのすべての列を必ずトランザクションに含める必要があります。例えば、次のようにトランザクションでよく使用される従業員番号列 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ます。

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 32 767 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

ある種の照会への応答で特定の複数の表のデータが頻繁に使われるような場合には、これらの表の互いに関連するデータを物理的にできるだけ近接して配置するのが適切です。パーティション・データベース環境では、この処理を*表コロケーション* といいます。

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分かれても差し支えありませんが、それら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る必要があります。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。

結合または副照会で複数の表にアクセスするとき、データベース・マネージャーは、結合される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かどうかを判別します。そのようなケースに該当する場合、データベース・パーティション間でデータを移動する代わりに、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が実行されます。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、同じ方法で扱われます (非互換データ・タイプの NULL 値はそのように扱われない可能性があります)。
- 「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。
- 分散キーの同一値の小数部は、位取りおよび精度が異なっている場合であっても、同一として取り扱われます。
- 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

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。 PARTITION BY 節を使って表が作成される場合、その表はパーティション化されます。

指定されるすべての表スペースは、同じページ・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)、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じデータベース・パーティション・グループに属する必要があります。

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 32,767 個のデータ・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。

パーティション表に対して、パーティション化または非パーティション化された索引を設定できます。 1 つのパーティション表に対して、非パーティション化された索引とパーティション化された索引の両方を共存させることもできます。

制約事項

パーティション化された階層表または一時表、範囲がクラスター化された表、およびパーティション化されたビューを、パーティション化された表で使用することはサポートされていません。

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したがって、表データが、データ・パーティション と呼ばれる複数のストレージ・オブジェクト間で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率的に使用したい。

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、この効率性を実現できます。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。

DB2 バージョン 9.7 フィックスパック 1 以降のリリースでは、DETACH PARTITION 節を指定した ALTER TABLE ステートメントを使用して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間、ソースのパーティション表は、RS、CS、または UR 分離レベルで実行されている動的照会によってアクセス可能な状態を維持します。同様に、DB2 V10.1 以降のリリースでは、ATTACH PARTITION 節を指定した ALTER TABLE ステートメントを使用してパーティション表にデータ・パーティションをアタッチする間、ターゲットのパーティション表は、RS、CS、または UR 分離レベルで実行されている動的照会によってアクセス可能な状態を維持します。

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。この設計には、以下のような利点があります。

- 索引をドロップしたり、オンラインで索引を作成したりする際のパフォーマンスが向上します。
- 表の各索引の間で、表スペース特性ごとに異なる値を使用できます (例えば、スペースの使用効率をより良くするには、各索引のページ・サイズに異なる値を使用する方が適している場合があります)。
- 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。この照会の処理を「データ・パーティション除去」といい、意思決定を支援する多くの照会で役立ちます。

以下の例では CUSTOMER という名前の表が作成されます。ここで、l_shipdate >= '01/01/2006' かつ l_shipdate <= '03/31/2006' である行が表スペース TS1 に保管され、l_shipdate >= '04/01/2006' かつ l_shipdate <= '06/30/2006' である行が表スペース TS2 に保管されて、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

パーティション表では、表の 1 つ以上の表パーティション・キー列の値にしたがって、表データが、データ・パーティション と呼ばれる複数のストレージ・オブジェクト間で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。この値はシステムによって生成されます。さまざまなユーティリティーからの出力でこの値が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導入により、DB2 データベースは 3 つのレベルのデータ編成スキームを提供します。データの編成方法を示すアルゴリズムを含む 3 つの節が CREATE TABLE ステートメントにあります。

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。データベース・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・パーティショニング)
- 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たり (マルチディメンション・クラスタリング)、挿入操作の時刻に従って行をグループ化したり (挿入時クラスタリング表) します。

この構文を使用すると節間の整合性を取ることができ、将来のデータ編成アルゴリズムのため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。この方法を使用すると、Informix® Dynamic Server および Informix Extended Parallel Server ハイブリッドでも同様の動作が可能になります。

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、パーティション・データベース環境 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。

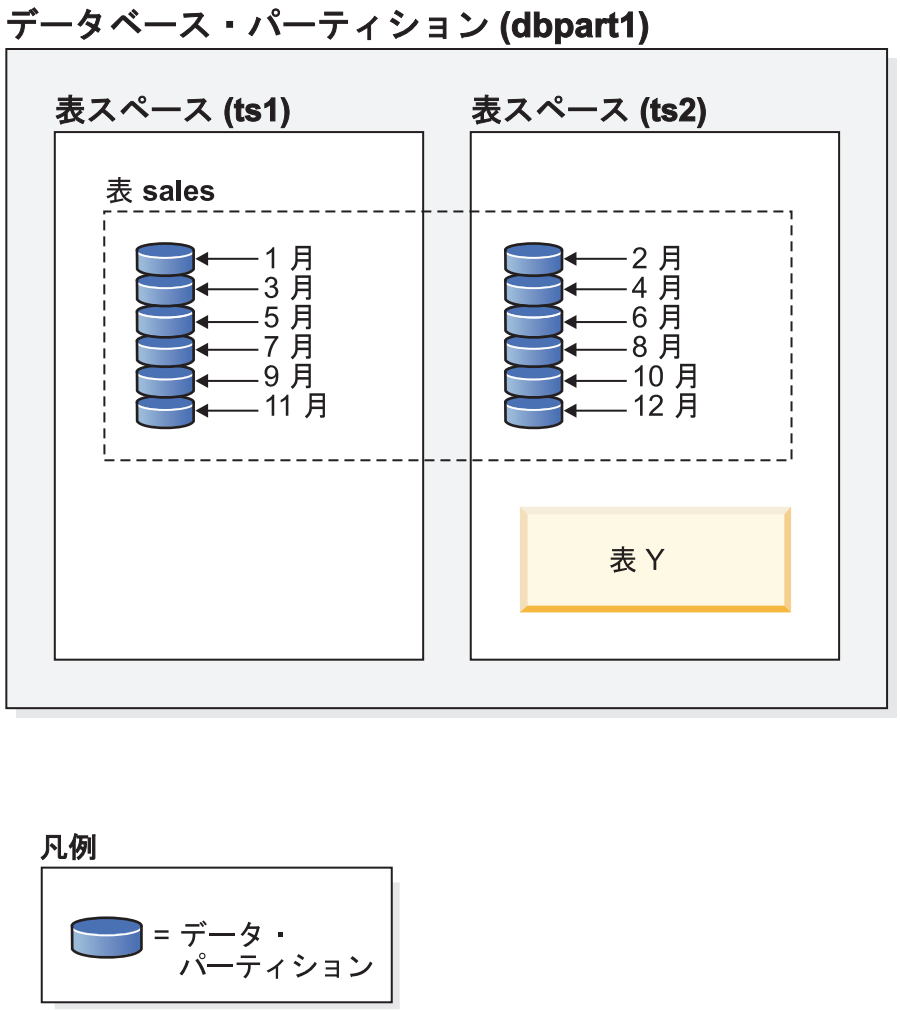


図 3. 月間売上高データを示す 1 つの表が複数のデータ・パーティションにパーティション化される表パーティション編成スキームを示しています。この表は、2 つの表スペースにまたがっています (ts1 と ts2)。

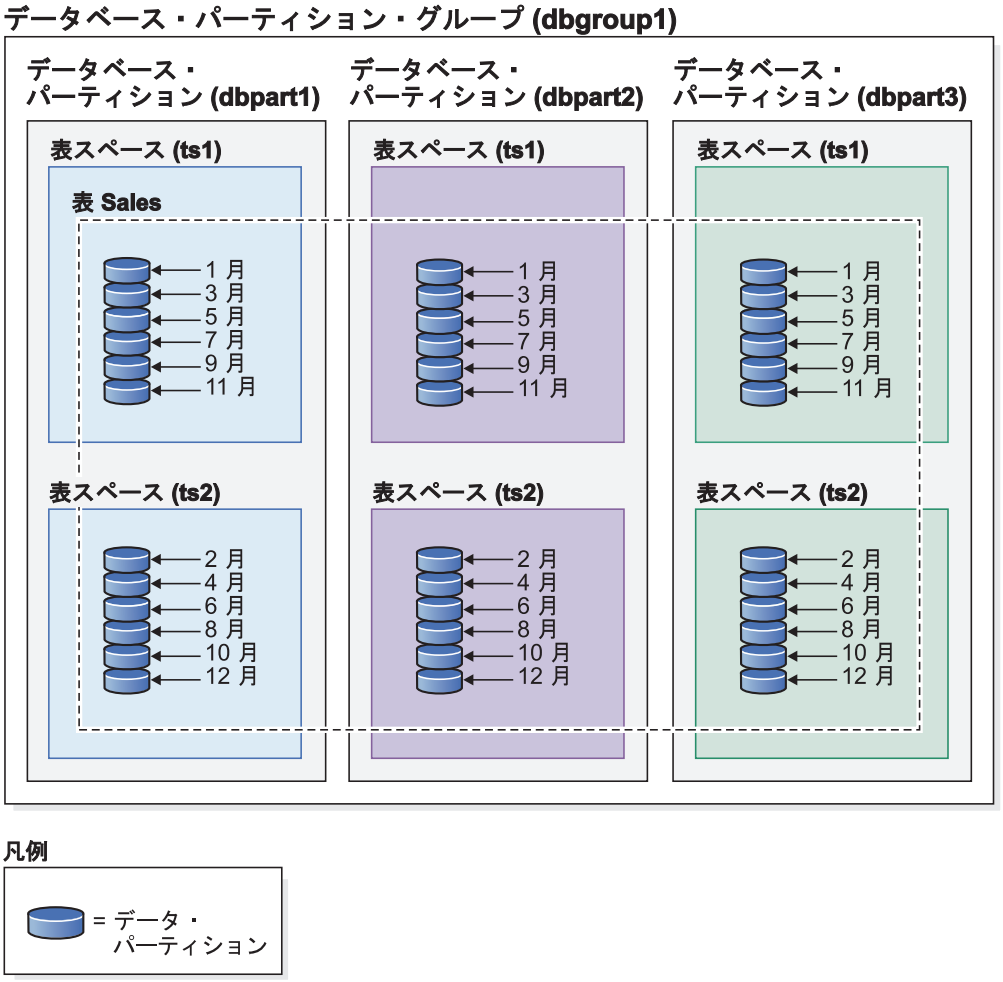


図 4. データベース・パーティションと表パーティションの補完的編成スキームを示しています。月間売上高データを示す 1 つの表は複数のデータ・パーティションにパーティション化されます。データは 2 つの表スペース (ts1 と ts2) にわたり、この 2 つの表スペースは、1 つのデータベース・パーティション・グループ (dbgroup1) の複数のデータベース・パーティション (dbpart1、dbpart2、dbpart3) に配分されます。

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持つ表) 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効率的に機能します。 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。

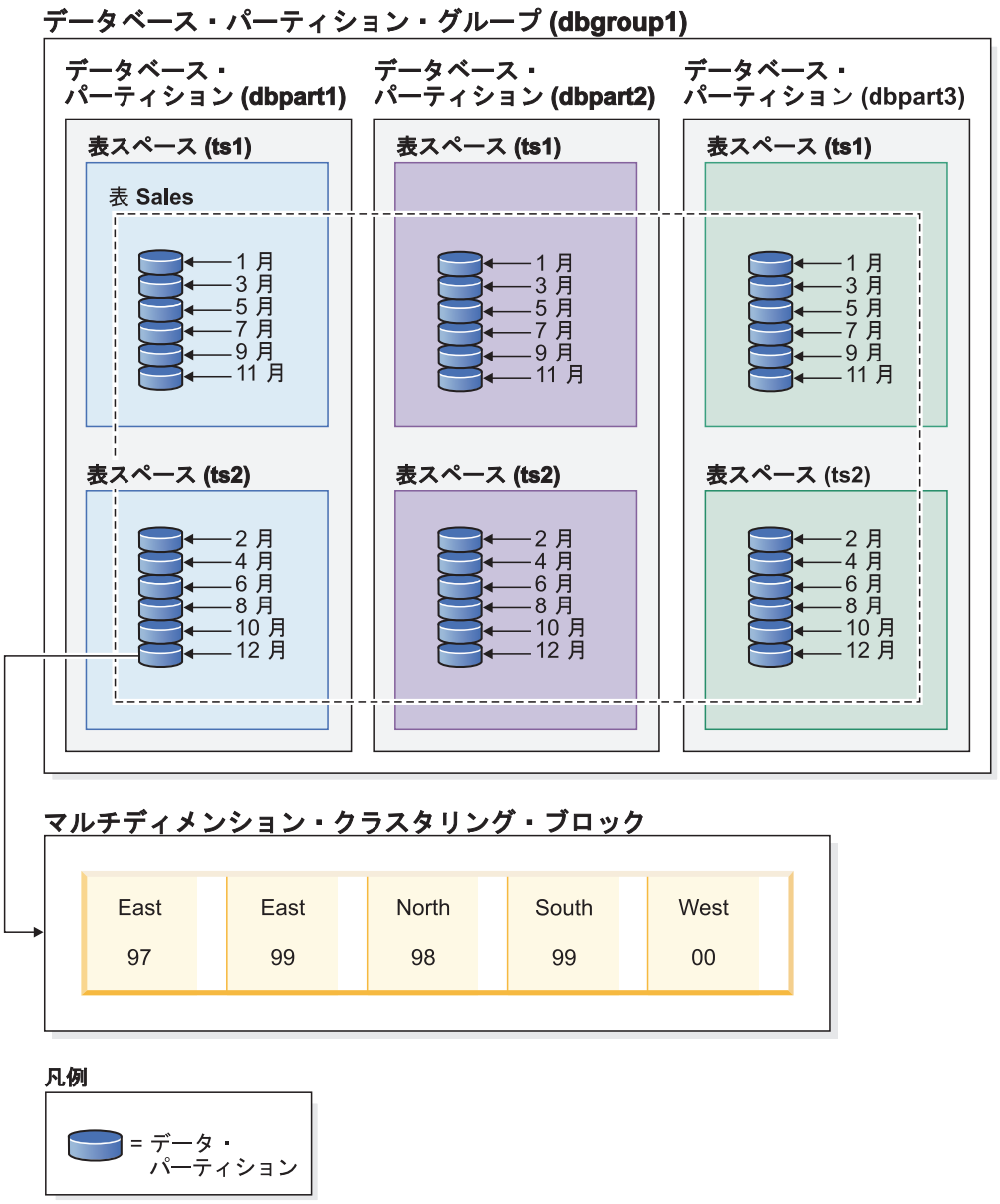


図 5. SALES 表からのデータが複数のデータベース・パーティションにまたがって配分され、表スペース ts1 と ts2 にパーティション化されるだけでなく、類似値を持つ行が Date ディメンションおよび Region ディメンションの両方にグループ化される、データベース・パーティション編成スキーム、表パーティション編成スキーム、およびマルチディメンション編成スキームの表示。

上記でリストされたスキームと共に使用できない、別のデータ編成スキームがあります。そのスキームは ORGANIZE BY KEY SEQUENCE です。各レコードを、表の作成時 (範囲がクラスター化された表) 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。

挿入時クラスタリング (ITC)
ORGANIZE BY INSERT TIME 節を指定することによって、行の挿入時刻に基づいてデータが物理的にクラスタリングされた表。

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。

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
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。
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したがって、表データが、データ・パーティション と呼ばれる複数のストレージ・オブジェクト間で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 DIMENSIONS

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関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。

- データのロールアウトと一致する範囲の細分度を定義する。週、月、または四半期が最もよく使用されます。
- 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。
- 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	DECFLOAT
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で使用できますが、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
- XML

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 または ITC 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。
- ターゲット表に XML 列がある場合、例外表を指定することはできません。
- 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 ロード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、SET INTEGRITY 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 SET INTEGRITY 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 ロード操作の場合は、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。

重要: バージョン 10.1 フィックスパック 1 以降、ALLOW READ ACCESS パラメーターは非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『LOAD コマンドの ALLOW READ ACCESS パラメーターが非推奨になった』を参照してください。

また、INGEST ユーティリティーはパーティション表をサポートしていて、LOAD コマンドで ALLOW READ ACCESS パラメーターを指定するよりもこのユーティリティーの方が、データ並行性と可用性に適しています。ターゲット表をロックすることなく、ファイルおよびパイプから大量のデータを移動することができます。また、経過時間または行数に基づいて、コミット後すぐにデータはアクセス可能になります。

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 を定義する照会は、表の中のデータの判別も行います。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。マテリアライズ照会表を使って照会の一部分を解決できるとデータベース・マネージャーが判断した場合、マテリアライズ照会表を使用するように照会が書き換えられることがあります。

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* の基になる表は、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ものの、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・パーティションに複製するのが適切な表です。複製されたマテリアライズ照会表を作成するには、REPLICATED オプションを指定して CREATE TABLE ステートメントを使用します。

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。必要とされる余分なストレージを最小限に抑えて、すべてのレプリカの更新による影響を最小にするためには、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の複製についても考慮する必要があります。この場合、一回の複製に大き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はコストを相殺します。

複製される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り、選択した列、選択した行、またはその両方を複製できます。

複製されたマテリアライズ照会表を使用する場合、不確定な操作が含まれる DELETE ステートメントや UPDATE ステートメントはサポートされていません。

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

マルチディメンション・クラスタリングとデータ・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マルチディメンション・クラスタリング (MDC) キーの両方で使用されます。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表に行うと、どちらかの機能を単独で使用するよりも、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 データウェアハウスの特性

以下の推奨事項は、DB2 V9.1 にとっては新しく典型的で主流であった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します。

- 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。
- パーティション・データベース環境が使用される (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単位にロードされる。
- 毎日の INGEST 量は、1 万から 1000 万以内のレコードである。
- データ量が変化する。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 (例えば 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。

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
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。
データの日次ロード (**LOAD** コマンドまたは **INGEST** コマンド)	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 を指定した **LOAD** コマンド、または **INGEST** コマンド)	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。

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
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、表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、 38 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);
```

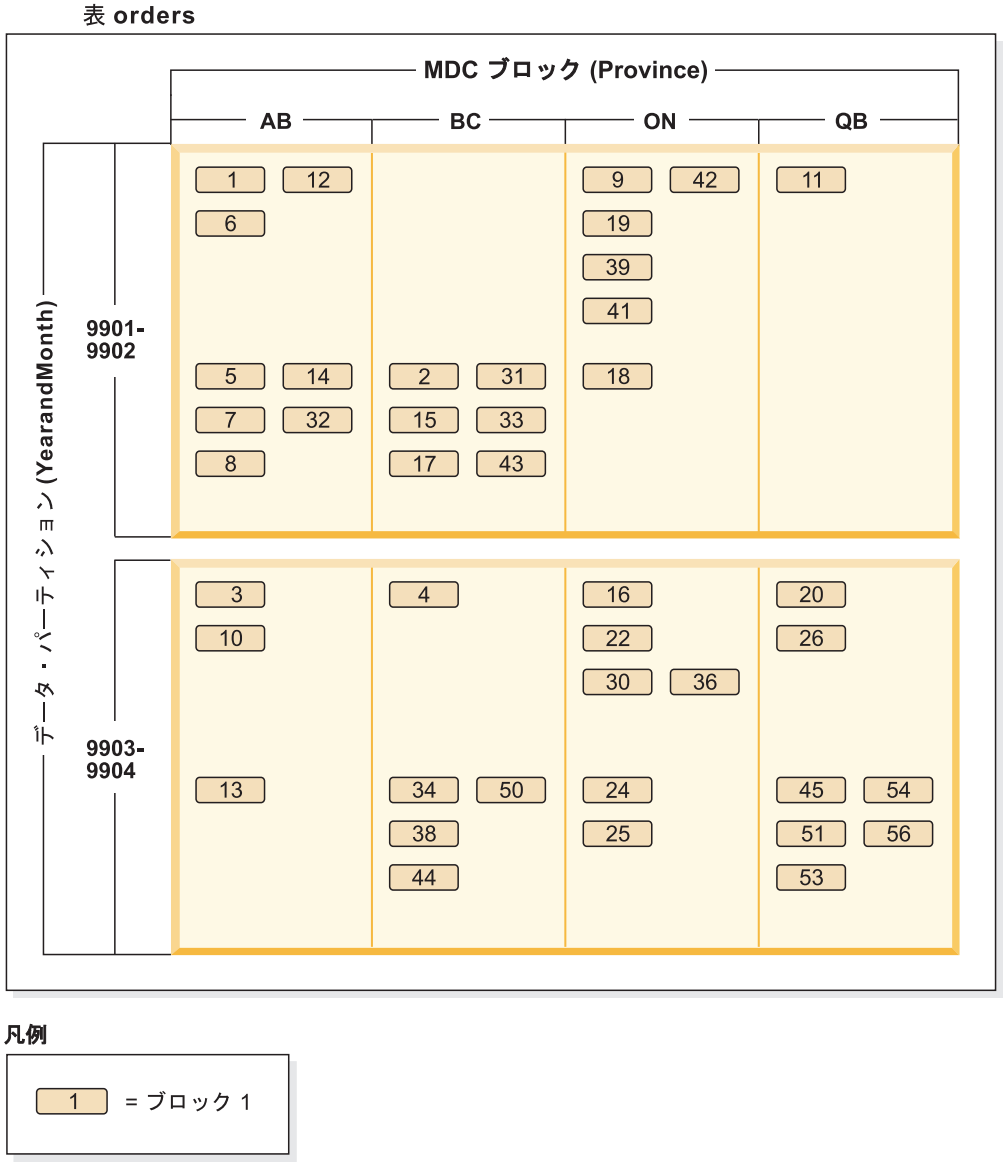


図 6. YearAndMonth によってパーティション化され、Province によって編成される表

例 2:

39 ページの図 7 に示されているように、YearAndMonth を ORGANIZE BY DIMENSIONS 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);
```

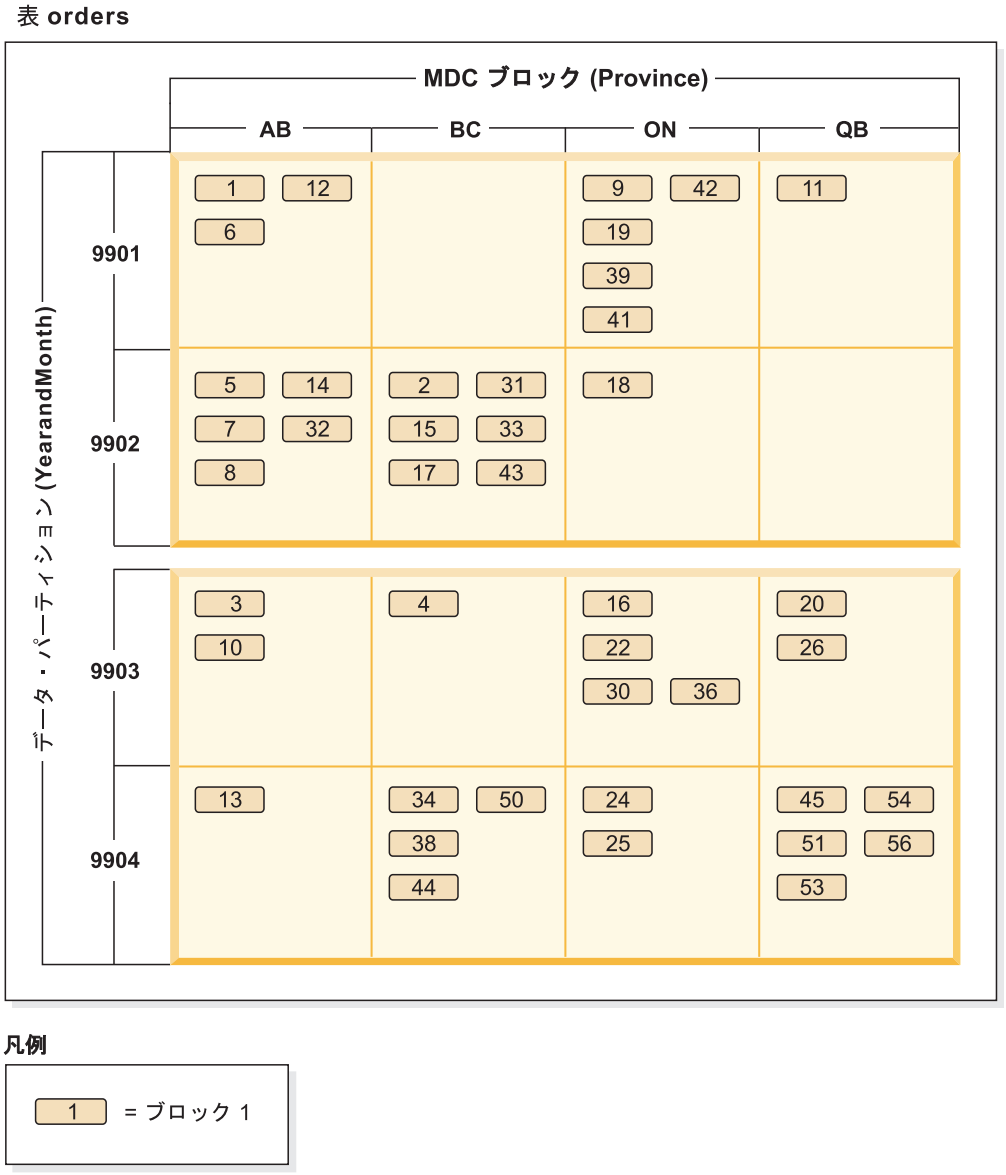


図 7. YearAndMonth によってパーティション化され、Province および YearAndMonth によって編成される表

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。
- DB2 V9.7 フィックスパック 1 以降のリリースで作成したデータ・パーティション化 MDC 表では、MDC ブロック索引はパーティション化されます。 DB2

V9.7 以前のリリースで作成したデータ・パーティション化 MDC 表では、MDC ブロック索引はパーティション化されません。

DB2 pureScale環境での表パーティション化

パフォーマンスを向上させるために、DB2 pureScale®で表パーティションを使用して、大規模な表オブジェクトを複数のパーティション間で分割することができます。

DB2 pureScaleの表には表パーティションを使用できます。これには、PARTITION BY RANGE 節を使用する表などがあります。また、表パーティションに関連するコマンドも DB2 pureScale環境で使用できます。

これは、例えば、以下の操作のすべてがサポートされることを意味します。

- ALTER TABLE ステートメントを介して実行できるパーティションのロールインおよびロールアウト操作
- CREATE INDEX ステートメントの PARTITIONED 節および NOT PARTITIONED 節
- パーティション索引に対する、REORG TABLE ステートメントおよび REORG INDEXES ALL ステートメントの ON DATA PARTITION 節

また、MON_GET_PAGE_ACCESS_INFO 表関数がパーティション表を処理するように更新されています。データ・パーティションに対して機能する既存のモニター関数はすべて、DB2 pureScale の表にも有効です。

既に DB2 pureScale Feature を使用している場合は、表パーティションを使用するとページ競合問題の解決に役立つ場合があります。より大きな範囲に競合を分散させることによって、データ・ページの競合を削減できます。同様に、パーティション索引を使用すると索引ページの競合を削減できます。

注: DB2 pureScaleのパフォーマンスの観点から、使用メモリーの量は、表パーティションと索引の数に依存します。メンバー上でパーティション化に使用されるメモリー・リソースは、**dbheap** 構成パラメーターから取得されます。 CF では、メモリー・リソースは **cf_sca_sz** 構成パラメーターによって定義されます。

第 2 章 範囲がクラスター化された表

範囲がクラスター化された表 (RCT) の表レイアウト体系では、表の各レコードが、事前定義されたレコード ID (RID) を持ちます。 RID は、表の中のレコードを見つけるために使用される内部的な ID です。

レコードのキー値と、特定の表の行のロケーションとを関連付けるために、アルゴリズムが使用されます。このアプローチにより、特定の表の行に対して非常に高速なアクセスが提供されます。このアルゴリズムは、ハッシュを使用しません。ハッシュは、キーと値の順序を保持しないからです。この順序を保持することにより、時間が経過しても表データを再編成する必要がなくなります。

表内の各レコードのキー値は、以下の条件を満たす必要があります。

- 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。 (必要な場合は、CREATE TABLE ステートメントで ALLOW OVERFLOW オプションを使用して、定義されている値の範囲を越えるキー値を持つ行が許可されるようにします。)

範囲がクラスター化された表には、特定の表の行への直接アクセス以外にも、以下の利点があります。

- 保守の必要性が低い。挿入、更新、あるいは削除のたびに更新する必要のある、B+ ツリー索引などの 2 次構造がない。
- 類似のサイズの B+ ツリー索引を持つ REGULAR 表と比較した場合、RCT でのロギング回数が少なくて済む。
- 必要なバッファー・プール・メモリーが少ない。 B+ ツリー索引などの 2 次構造を格納するための追加のメモリーが不要。

RCT のためのスペースは事前割り振りされていて、レコードがまだない場合でも、表で使用するために予約されています。そのため、範囲がクラスター化された表には、フリー・スペース制御レコード (FSCR) が不要です。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。 VARCHAR のような可変長フィールドが定義されている場合は、フィールドの最大長が使用され、レコード全体のサイズは固定長になります。そのため、スペースを最適に使用できなくなります。キー値がまばらである場合は、未使用のスペースにより、範囲スキャンのパフォーマンスが低下します。範囲スキャンは、範囲内で可能性のあるすべての行をスキャンする必要があり、データがまだ含まれていない行も対象になります。

範囲がクラスター化された表に対してスキーマの変更が必要になった場合は、新しいスキーマ名を使用して表を再作成し、前の表からデータを取り込む必要がありま

す。例えば、表の範囲を変更する必要がある場合は、新しい範囲を持つ表を作成し、以前の表からデータを取り込みます。

RCT がオーバーフロー・レコードを許可する場合、新規レコードのキー値が、定義されている値の範囲を越えると、そのレコードはオーバーフロー域に配置されます。オーバーフロー域は動的に割り振られます。オーバーフロー域に追加されるレコードの数が増加すると、このオーバーフロー域に関連する表に対する操作の処理時間が増加します。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。これが問題になる場合は、より大きな範囲を持つ新規の RCT にデータをエクスポートして、オーバーフロー域のサイズを削減することを考慮してください。

範囲がクラスター化された表に関する制約事項

範囲がクラスター化された表を使用できない状況があります。また、一部のユーティリティーは、範囲がクラスター化された表を操作できません。

範囲がクラスター化された表には、以下の制限が適用されます。

- 範囲がクラスター化された表は、DB2 pureScale 環境では指定できません (SQLSTATE 42997)。
- パーティション表は、範囲がクラスター化された表にはできません。
- 宣言済み一時表および作成済みの一時表は、範囲がクラスター化された表にはできません。
- 自動サマリー表 (AST) は、範囲がクラスター化された表にはできません。
- ロード・ユーティリティーはサポートされません。インポート・ユーティリティーまたは並列の挿入アプリケーションを使用して、範囲がクラスター化された表にデータを挿入できます。
- **REORG** ユーティリティーはサポートされません。DISALLOW OVERFLOW オプションとして定義されている、範囲がクラスター化された表を再編成する必要はありません。 ALLOW OVERFLOW オプションを指定して定義されている、範囲がクラスター化された表で、オーバーフロー領域内のデータを再編成することはできません。
- 範囲がクラスター化されたマテリアライズ照会表である場合は、CREATE TABLE ステートメントに DISALLOW OVERFLOW 節を指定できません。
- 設計アドバイザーでは、範囲がクラスター化された表は勧められていません。
- マルチディメンション・クラスタリングおよびクラスタリング索引は、範囲がクラスター化された表と同時には使用できません。
- 値とデフォルトの圧縮はサポートされません。
- 範囲がクラスター化された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** パラメーター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。

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。

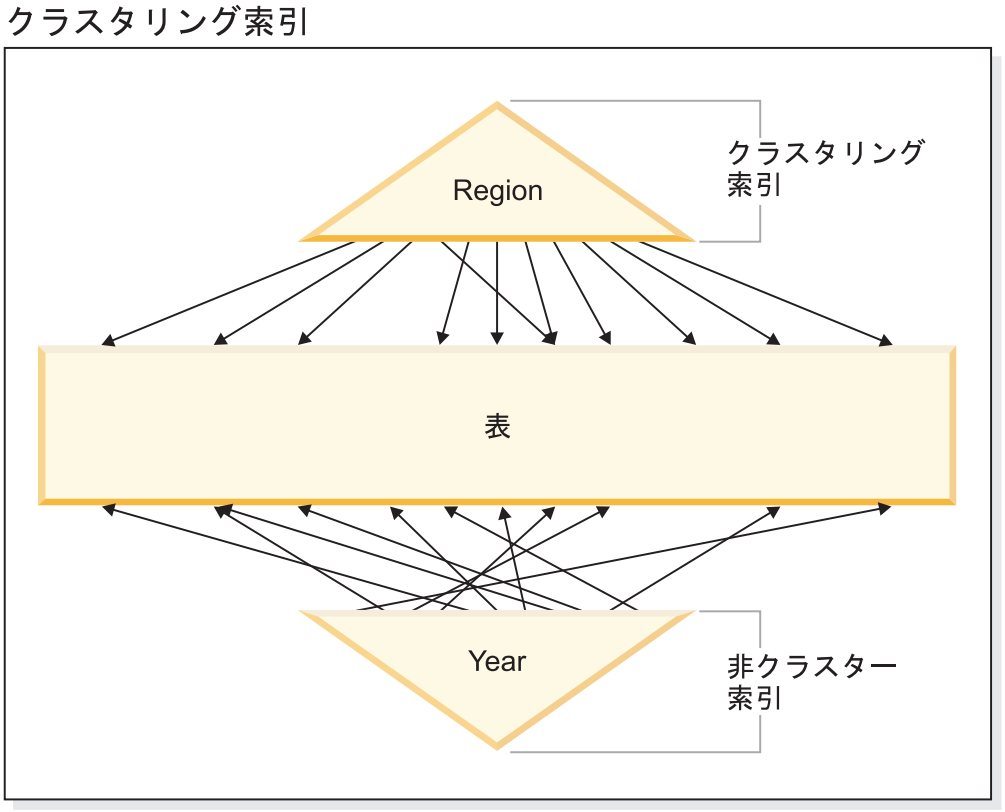


図 8. クラスター索引を伴う通常表

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を決定した後、どんなディメンションを選択するかは、それらの表を使ってブロック・レベルのクラスタリングを活用する照会の種類に依存するだけでなく、より重要なこととして、実際のデータの量と分布に依存します。

MDC 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。等式述部または範囲述部を含む照会に関係する列 (特にカーディナリティーの低い列) は、クラスタリング・ディメンションの利点を最大限に活用できます。ディメンション表を持つスター型結合に含まれる MDC ファクト表内の外部キーのディメンションを作成することを考慮してください。複数のディメンションで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'
```

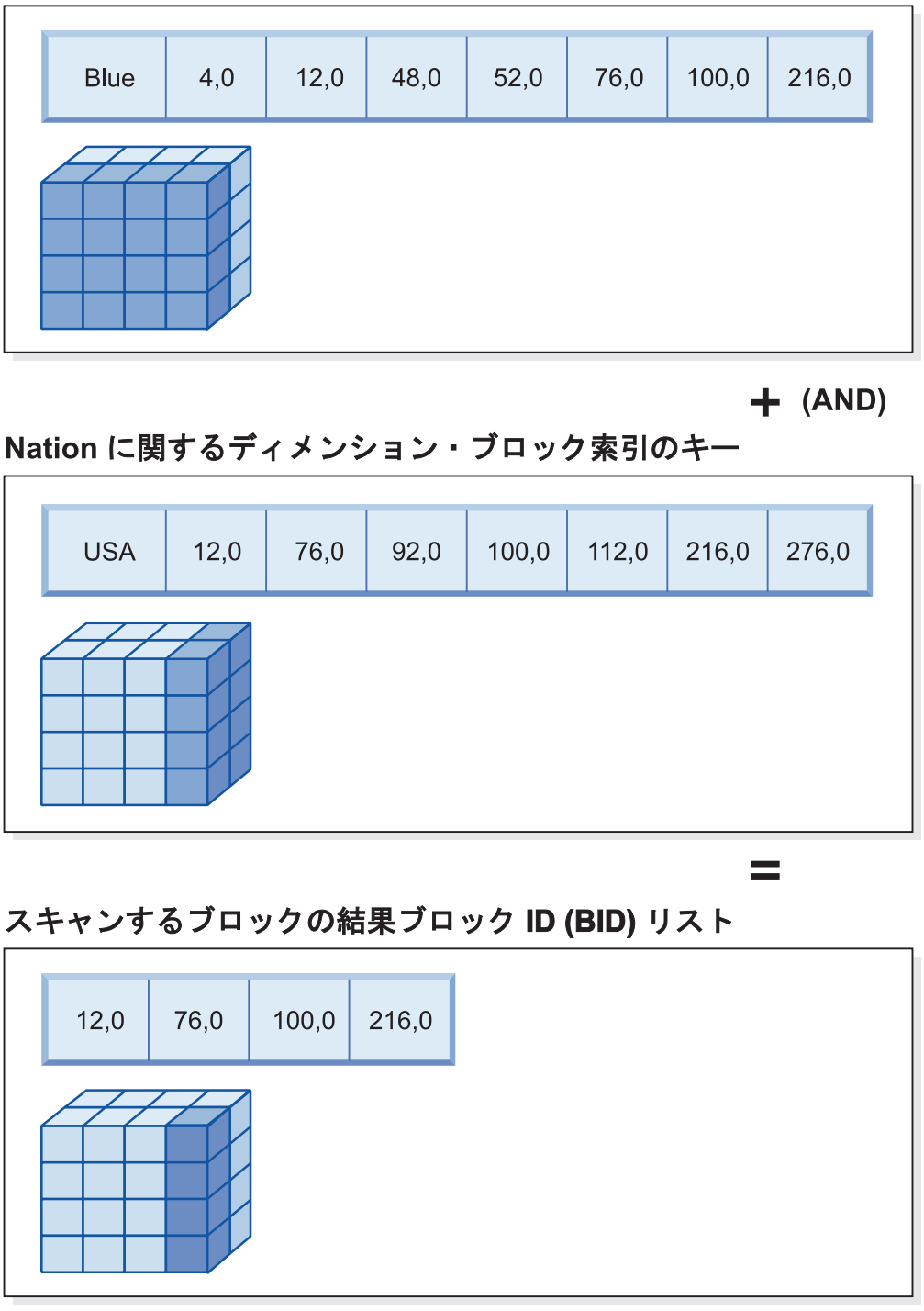


図 9. 2 個のブロック索引との論理 AND 演算を使用する照会要求

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、および論理 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
```

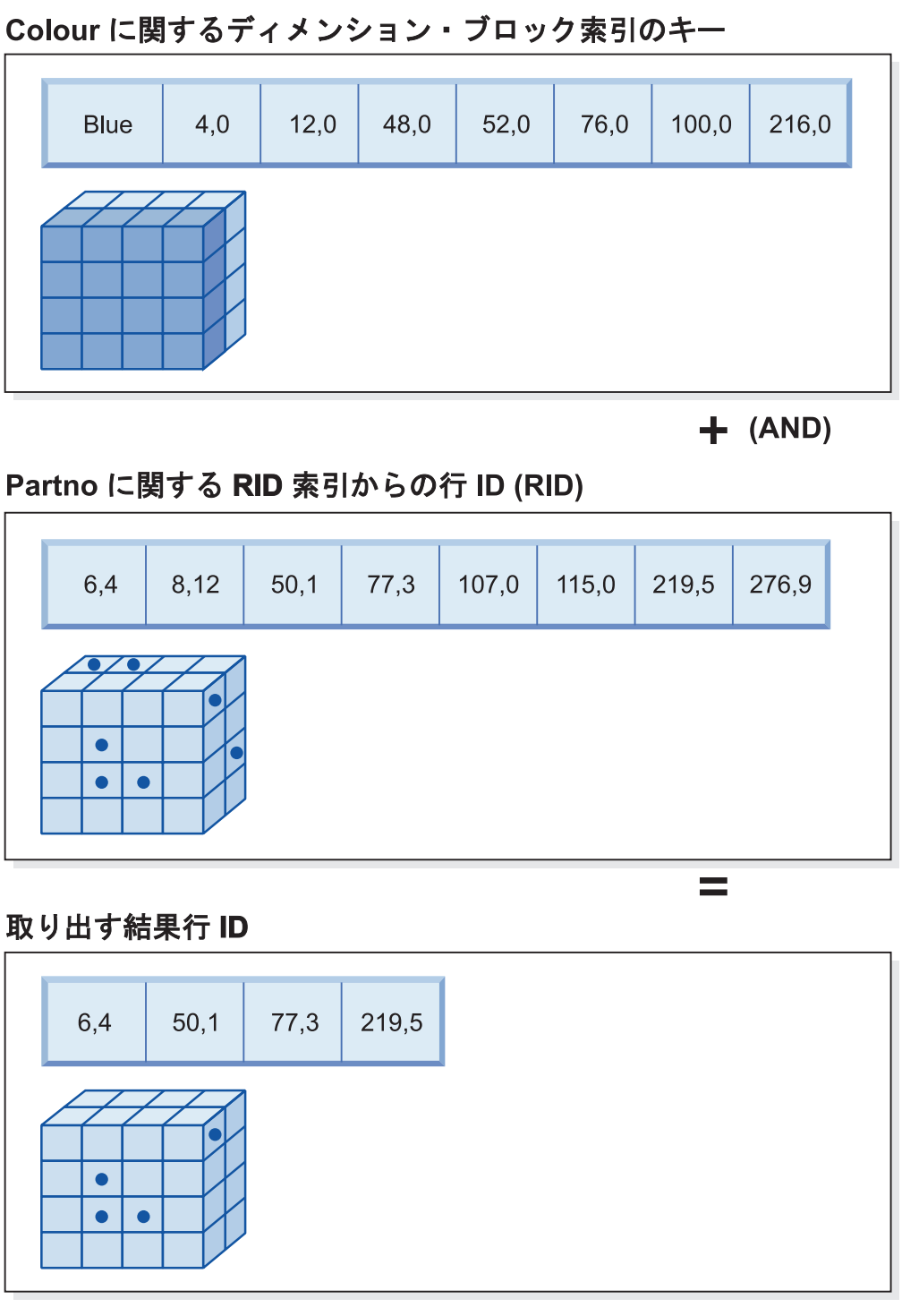


図 10. 1 つのブロック索引および行 ID (RID) 索引に対する論理 AND 演算を使用する照会要求

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
```

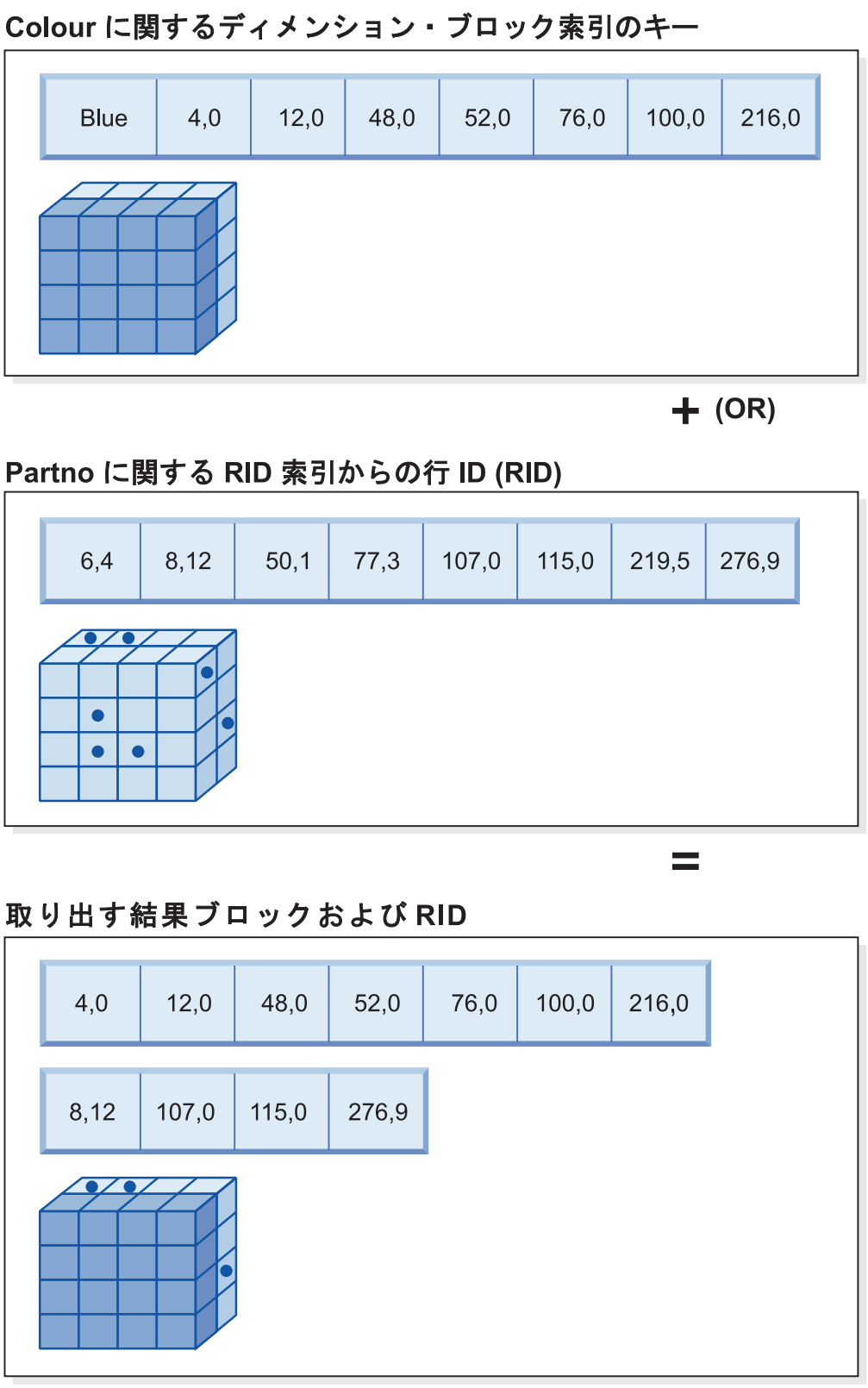


図 11. 論理 OR 演算を使用したブロック索引と行 ID の動作

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、ディメンション c1 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、および論理 OR 演算が含まれています。 c4 列に関する RID 索引がある場合には、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 索引を使用して論理 OR 演算を行うように、これを内部的に書き直すことができます。 c4 に関する索引がない場合には、すべてのレコードを検査する必要があるため、代わりに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、および c4 に対する値 100 の RID 索引スキャンが、論理 OR 演算で使用されます。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 表である必要はありません。
region、year、および product は、(ディメンション表が MDC 表である場合に) 通常の (正規) 索引またはブロック索引を使って索引化できる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。少ない数のレコードだけを含んでいるセルでも、そのレコードを保管するために、やはりページのブロック全体を必要とします。ブロック・サイズが大きい場合、この表は、必要なサイズよりもかなり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。表が存在するならば、その表のディメンションとなる各列の中で異なる値の数を選択することにより、現在のデータに関する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ば、データ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判別してください。
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・パーティションでの 1 セル当たりのレコード数を考慮してください。データベース・パーティションに基づいてデータにブロックが割り振られるためです。パーティション・データベース環境で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・パーティションでの 1 セル当たり平均レコード数を考慮してください。

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。

  関係する列に対して頻繁に使われる述部を考慮に入れてください。例えば、Month of Date、Quarter、あるいは Day 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日付上にあり、Month 上に表をクラスター化した場合、DB2 for Linux, UNIX, and Windows は、YearAndMonth 上のブロック索引を使用して、必要な日付が含まれている月を素早く限定し、関連するそれらのブロックだけ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、(12 個の異なる値が入っている) Region 列を Regions

West、North、South、East にロールアップするような方法で、セルの密度を増やすために他のいずれか 1 つのディメンションをロールアップすることを考慮できます。

MDC または ITC 表を作成する際の考慮事項

MDC または ITC 表を作成する際には、さまざまな要因を考慮する必要があります。現在のデータベース環境 (例えばパーティション・データベースが存在するかどうか) に応じて、またディメンションの選択に応じて、MDC または ITC 表を作成、配置、および使用する方法の決定が異なることがあります。

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、データ保守操作の要件を少なくするため、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 表にデータを移動するには、以下のようにします。

1. データをエクスポートする。
2. 元の表をドロップする (オプション)。
3. (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) 表を作成する。
4. MDC 表にデータをロードする。

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手順を実行します。

1. 表の従属オブジェクトをすべてドロップします。
2. 表の名前を変更します。
3. 新しい定義で表を作成します。
4. 表の従属オブジェクトをすべて再作成します。
5. 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、古い表のデータを選択して新しい表にそのデータをロードする際に、列関数を使って古いデータ・タイプから新しいデータ・タイプに変換することができます。

既存の表から ITC 表へのデータの移動

データ保守操作の必要性を減らすため、通常の表のデータを、挿入時クラスタリング (ITC) 表に移動することができます。既存の表から ITC 表にデータを移動するには、オンライン表移動のストアード・プロシージャーを使用します。

ExampleBank のシナリオは、既存の表から ITC 表にデータを移動する方法を示します。また、ITC 表を使用する際にスペースを再利用することの利便性についてもシナリオで示されています。詳しくは、関連概念のリンクを参照してください。

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**) 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 (異なる値の数) を少なくすることに関する数学用語です。一般的な例としては、日付、週、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ります。エクステント・サイズは 1 個のブロックまたはセルに入れることのできるレコード数に影響を与えるためです。エクステント・サイズはセルの密度に直接影響します。

表に関して単一ディメンションまたは複数ディメンションのどちらも推奨される可能性がありますが、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・タイプ例外には CHAR、VARCHAR、GRAPHIC、および VARGRAPHIC 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利点だけでなく、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 フィーチャーを実行するには、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 DIMENSIONS 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

パーティション・データベース環境でマルチディメンション・クラスタリングを使用できます。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データベース・パーティション間で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・パーティション上に、Yellow スライスがもう 1 つのパーティション上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てセル占有度を見積もる際には、平均的に、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、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。 ORGANIZE BY DIMENSIONS 節を使用してこのことを行えます。

例 1:

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

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

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・ブロック索引があります。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 つがあります。この 2 つの方法の主な違いは、例 1 の場合、 c4 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 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。この方法で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 が列 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` である生成列 month に関するディメンションを持つ MDC 表を作成するとします。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含めます。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 表および ITC 表のロード考慮事項

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、マルチディメンション・クラスタリング (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、削除 (コミット済み) によって空になったブロックを再使用できます。また、据え置きクリーンアップの MDC ロールアウト・フィーチャーを使用するよう選択することもできます。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。挿入時クラスタリング (ITC) 表では、表を拡張する前に、使用されていないブロックを可能な限り再使用します。これには、再利用されたブロックも含まれます。 ITC 表では、ロールアウトはサポートされていません。

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で、表に複数のディメンションがある場合は、以下を検討してください。

- **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ます。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。バージョン 9.5 から、**LOAD** コマンドの **DATA BUFFER** パラメーターの値は、システムにさらに使用可能メモリーがある場合に、 **util_heap_sz** を一時的に超えられるようになりました。
- 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 MDC または ITC 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** パラメーター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 または ITC 表には `anyorder` ファイル・タイプ修飾子が必要です。 `anyorder` 修飾子を使用せずに MDC 表または ITC 表へのロードを実行すると、この修飾子はユーティリティーによって明示的に使用可能にされます。

MDC または ITC 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、新しいセルに属するデータ、表の初期データ、および ITC 表への追加データなどのために使用するのが最善です。

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

MDC 表および ITC 表のためのロギング考慮事項

索引の保守およびロギングは、RID 索引を使用する場合と比較して、ディメンションと (それに伴って) ブロック索引を使用することにより削減できます。

データベース・マネージャーは、ブロック全体の最後のレコードが削除された場合のみ、ブロック索引から BID を除去します。この索引操作もまた、このときログに記録されます。同様に、データベース・マネージャーは、新規ブロックに新規レコードが挿入されたときのみ、ブロック索引に BID を挿入します。そのレコードは、論理セルの最初のレコードであるか、現在いっぱいになっているブロックの論理セルに対する挿入である必要があります。この索引操作もまた、このときログに記録されます。

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 表および ITC 表のブロック索引の考慮事項

MDC 表のディメンションを定義すると、ディメンション・ブロック索引が作成されます。さらに、マルチディメンションが定義されている場合には、複合ブロック索引も作成されることがあります。しかし、MDC 表にディメンションを 1 つだけ定義した場合、または表が挿入時クラスタリング (ITC) である場合、ブロック索引が 1 つだけ作成されます。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。データ・パーティション化 MDC または ITC 表では、表の MDC または ITC ブロック索引がパーティション化されます。

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、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・ブロック索引が作成されます。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。

MDC 表の場合、複合ブロック索引は、照会処理において、表内にある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用されます。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ります。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とします。

```
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) に関して作成されます。 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ます。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が作成されます。

```
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。

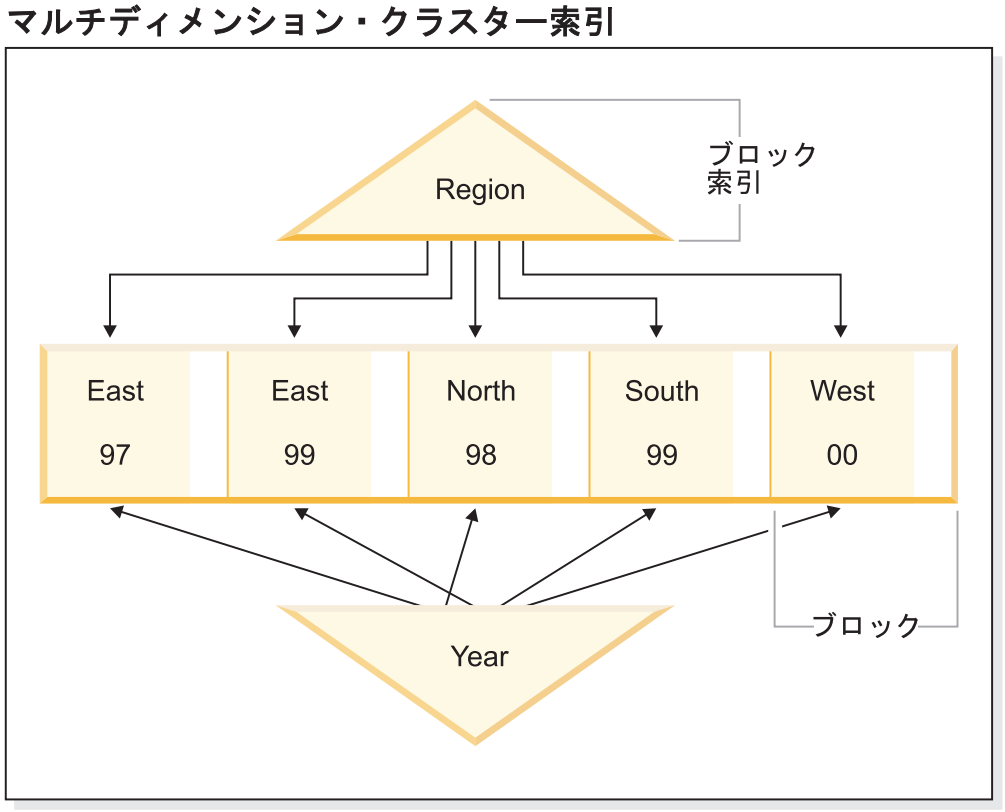


図 12. マルチディメンション・クラスター化表

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 56 ページの『MDC または ITC 表を作成する際の考慮事項』 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 DIMENSIONS 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 表および ITC 表のブロック索引の考慮事項』 を参照してください。

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

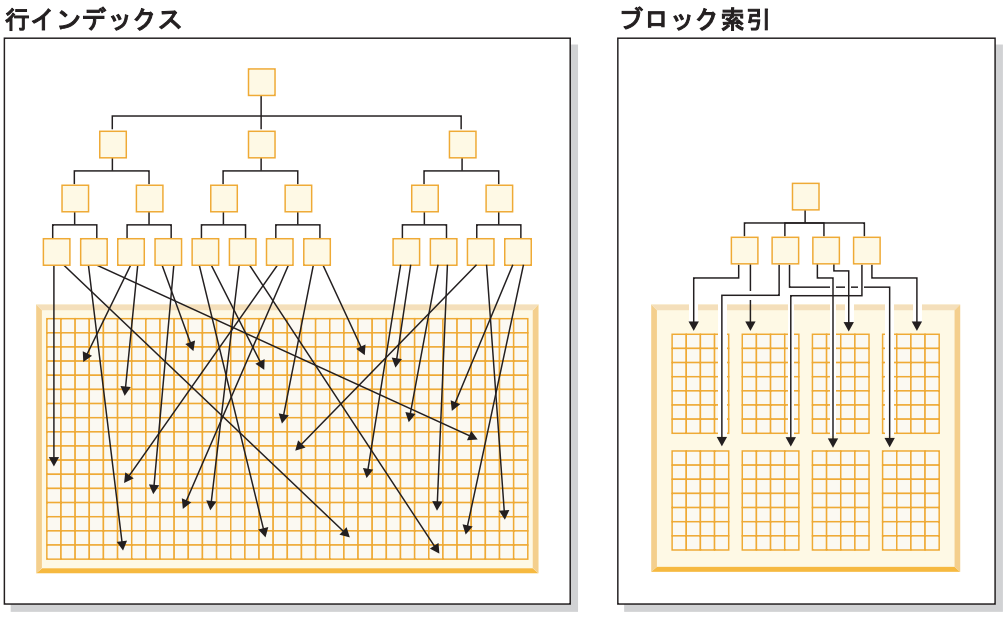


図 13. 行索引とブロック索引の違い

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。

MDC 表はパーティション化できます。パーティション化 MDC 表のブロック索引は、パーティション化、または非パーティション化のどちらも可能です。

- DB2 V9.7 フィックスパック 1 以降のリリースで作成したパーティション化 MDC 表では、表のブロック索引はパーティション化されます。
- DB2 V9.7 以前のリリースで作成したパーティション化 MDC 表では、表のブロック索引はパーティション化されません。

非パーティション化ブロック索引は、データベースをDB2 V9.7 フィックスパック 1 以降のリリースにアップグレードした後も、サポートされます。

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。

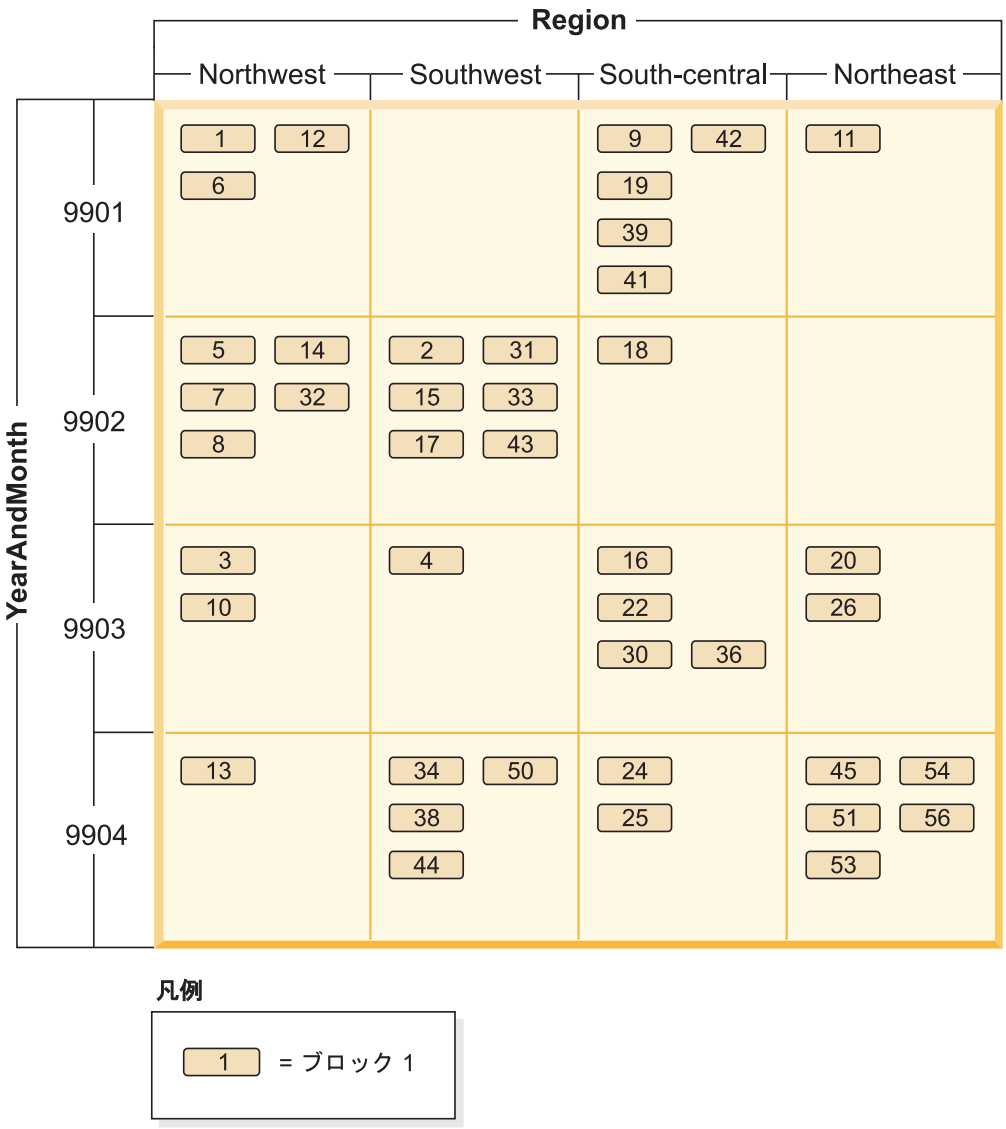


図 14. ディメンション 'Region' および 'YearAndMonth' が含まれるマルチディメンション表 Sales

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。

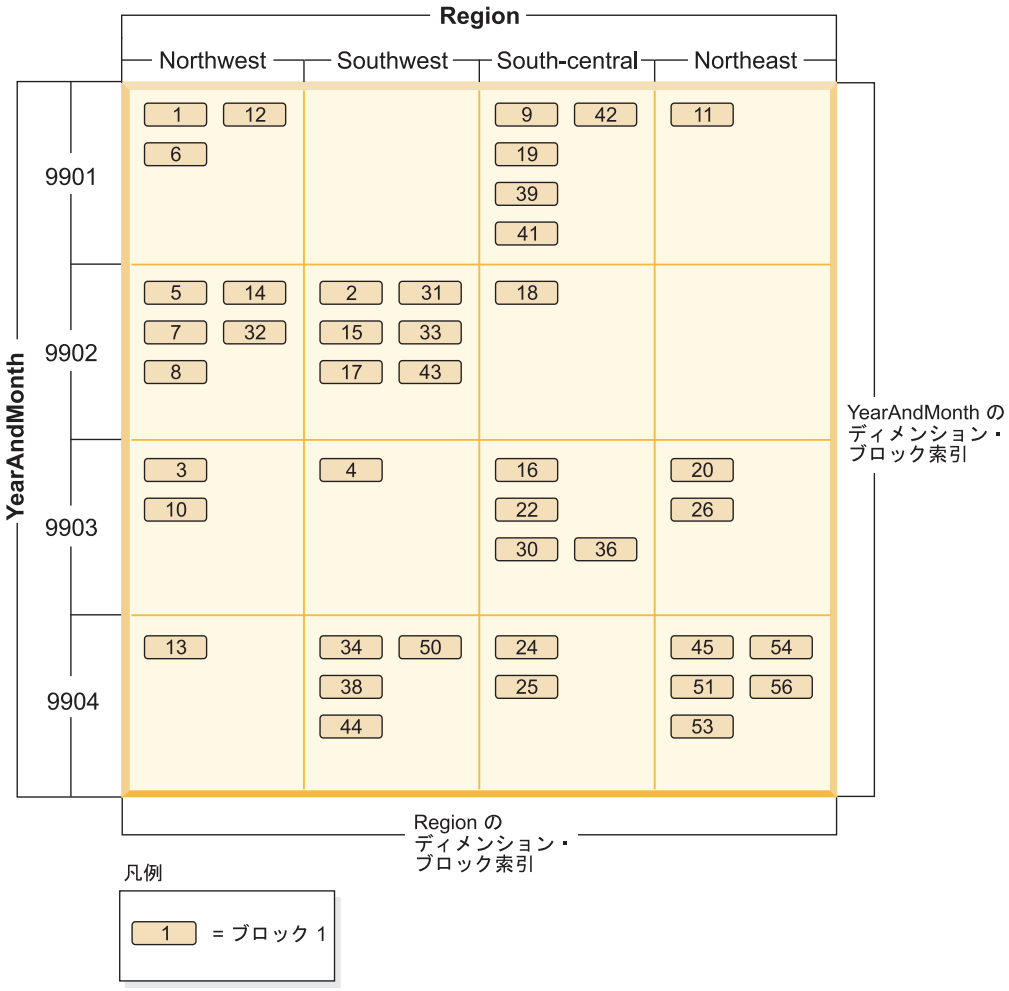


図 15. 'Region' および 'YearAndMonth' のディメンションを含む Sales 表とディメンション・ブロック索引

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。

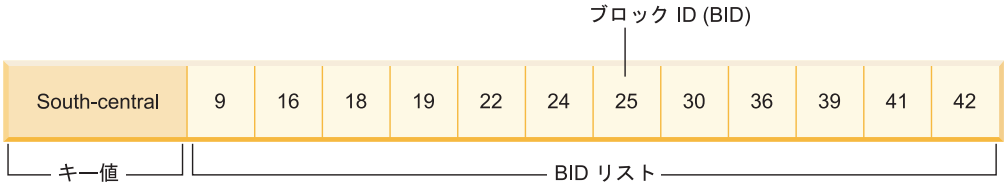


図 16. 'Region' に関するディメンション・ブロック索引のキー

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)。

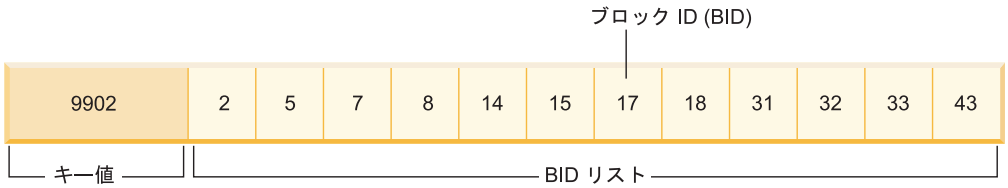


図 17. 'YearAndMonth' に関するディメンション・ブロック索引のキー

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。

- 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。
- 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 データベース・マネージャーは索引を先読みし、大ブロック入出力を使用してデータのブロック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コストが発生しないようにします。
- 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・サイズと等しい場合、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。
- 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。この方が、通常は、RID ベースの取り出しを使用してデータをスキャンするよりも高速で行われます。

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。

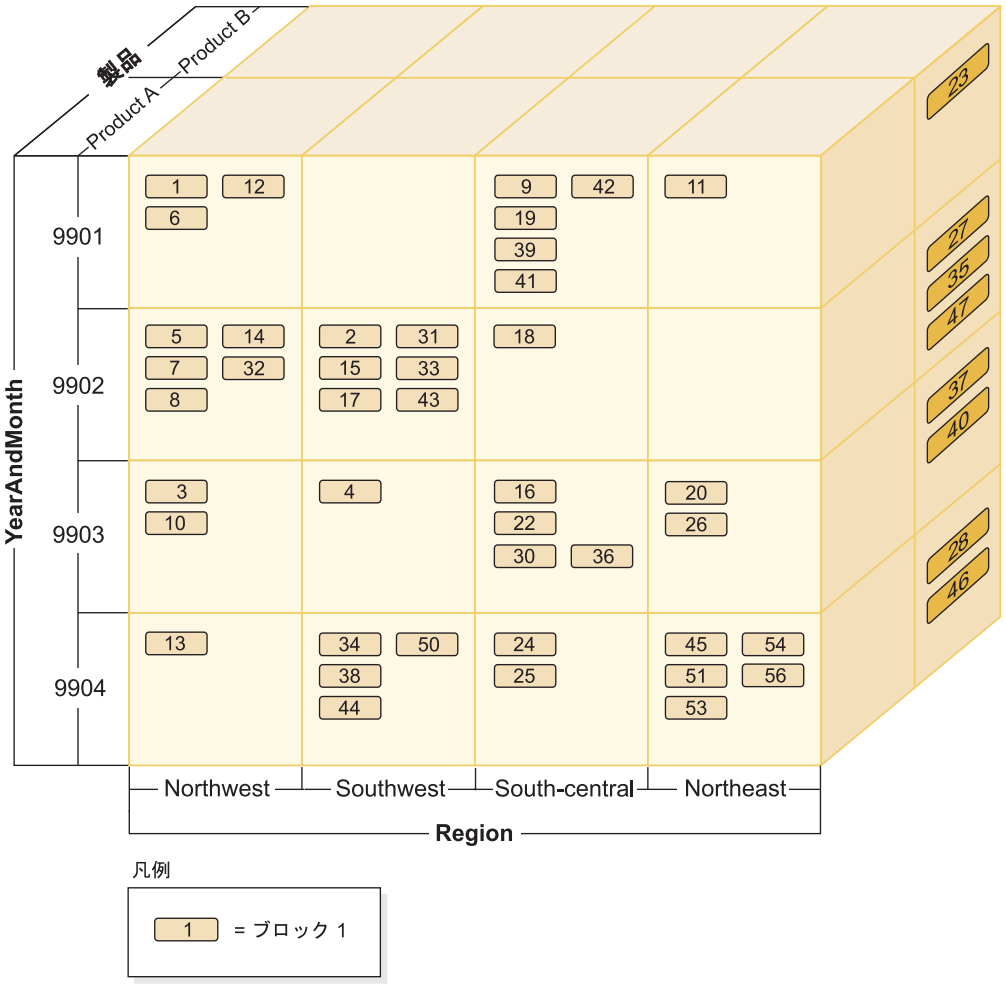


図 18. ディメンション 'Region'、'YearAndMonth'、および 'Product' が含まれるマルチディメンション表

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。(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

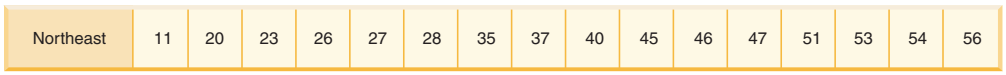


図 20. 'Region' に関するディメンション・ブロック索引のキー

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。

上記の例で、 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。

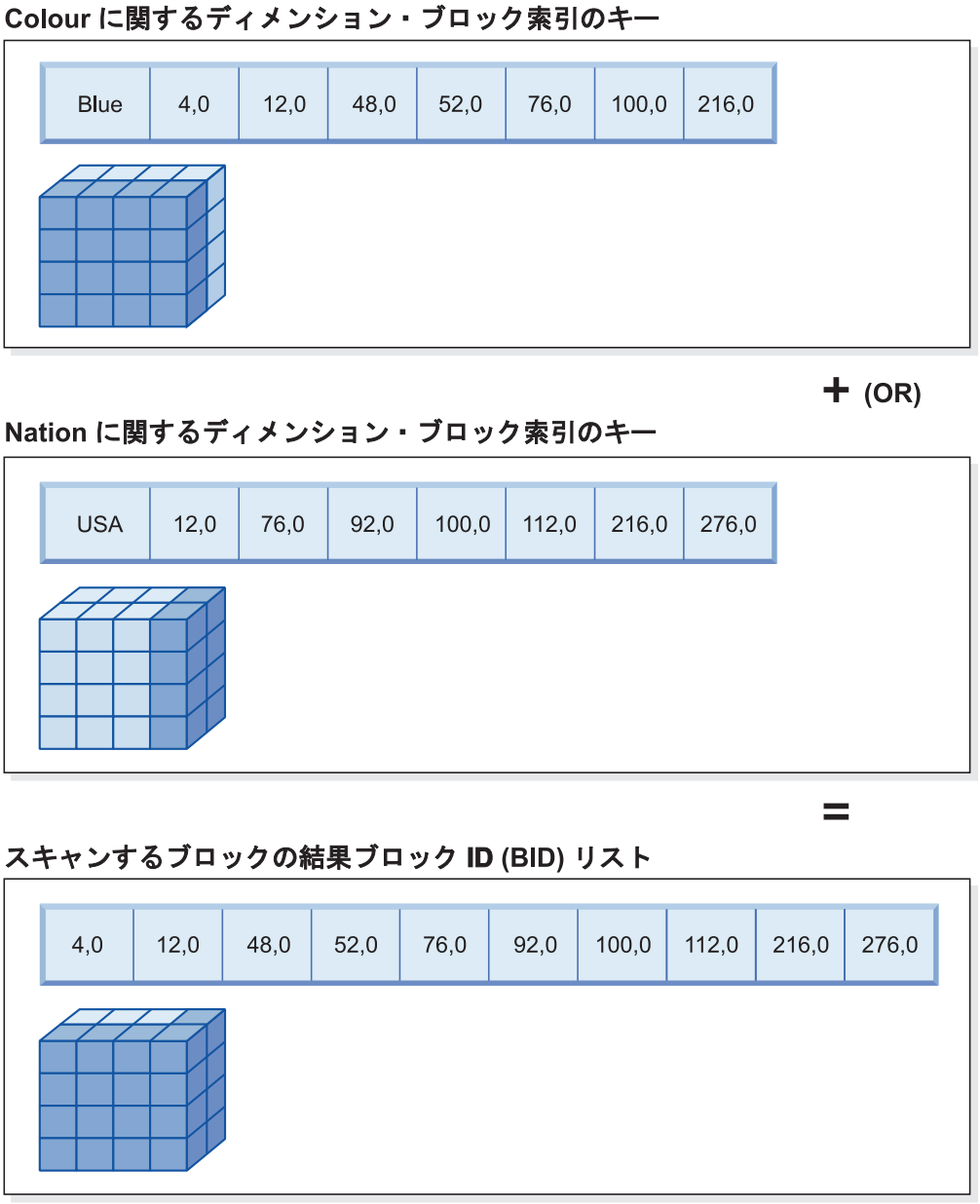


図 21. ブロック索引での論理 OR 演算の使用方法

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。

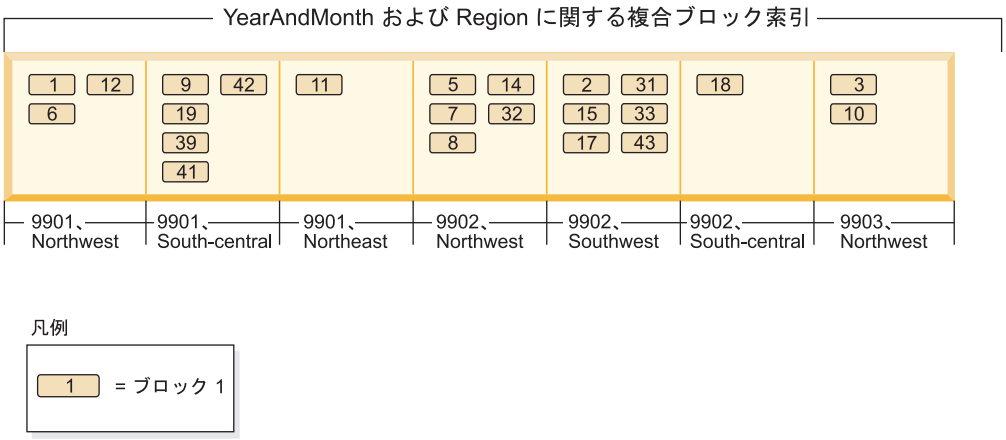


図 22. 'YearAndMonth' および 'Region' に関する複合ブロック索引

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。

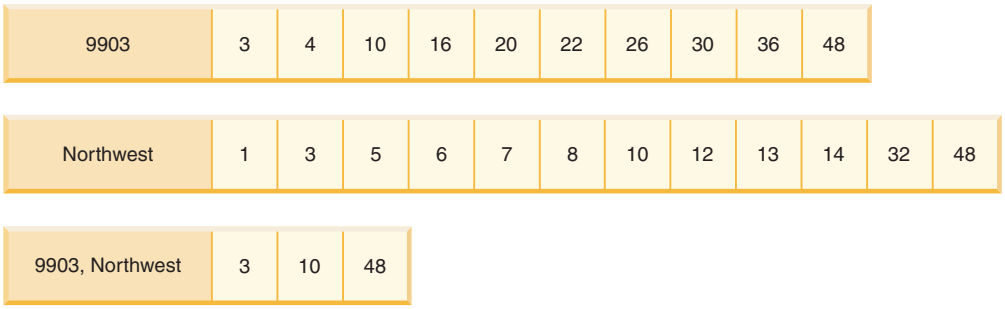


図 24. ブロック 48 追加後のディメンション・ブロック索引のキー

MDC 表および ITC 表のブロック・マップ

MDC 表では、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。 ITC 表では、すべてのブロックが単一のセルと関連付けられます。セル内でブロックを解放すると、そのブロックは後続の挿入で再利用できます。この再利用により、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 または ITC 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。

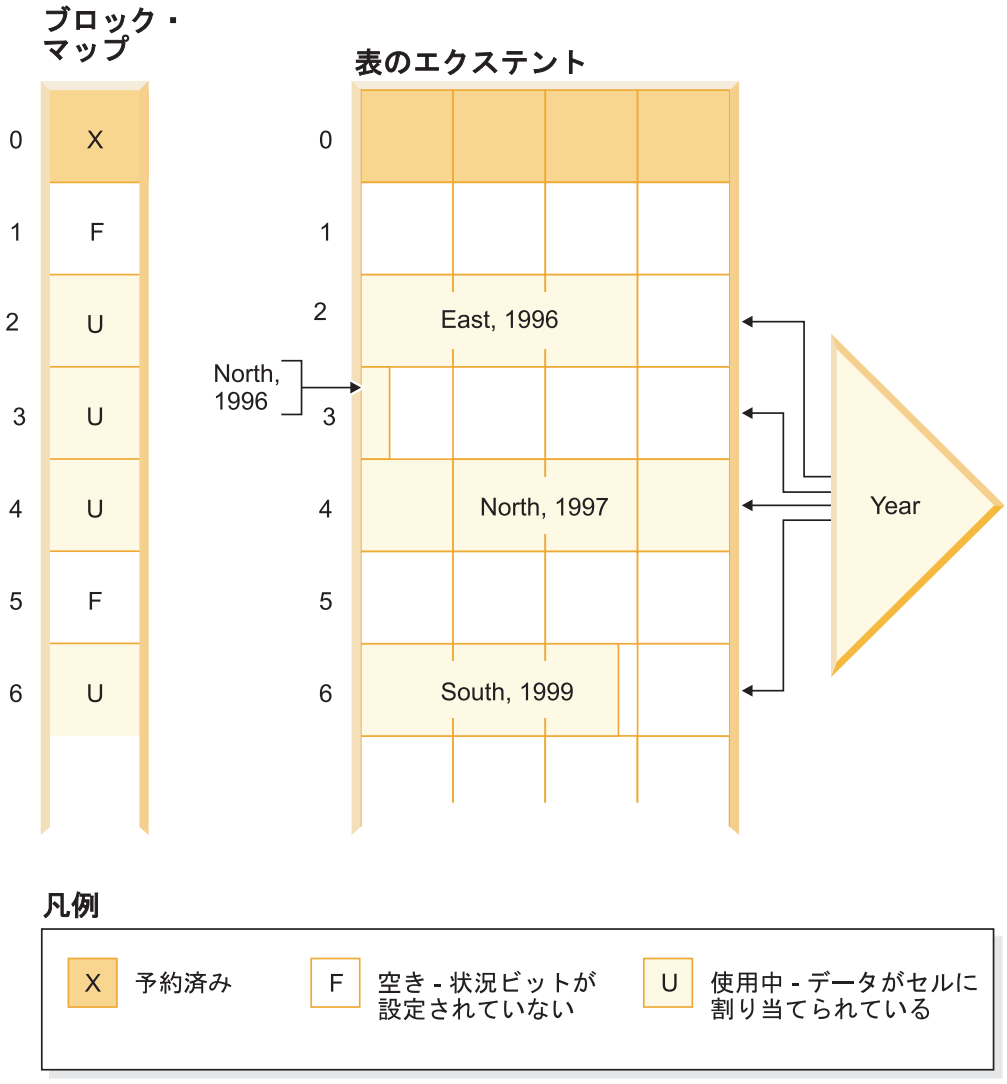


図 25. ブロック・マップの動作

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・ページには触れない)、その次からスキャンを続けます。

MDC 表および ITC 表からの削除

MDC または ITC 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。

削除によりブロック内の最後のレコードが除去されると、データベース・マネージャーはそのブロックを解放します。ブロックは、IN_USE 状況ビットを変更し、すべてのブロック索引からブロックの BID を除去することにより解放されます。また、レコード・ベースの索引も存在する場合、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 回実行されます。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 表および ITC 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。 MDC 表または ITC 表のレコードを更新することでそのレコードが長くなり、ページに収まらなくなる場合は、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。

ITC 表で、更新された行を配置するためのスペースがブロック内にない場合、その行は新規ブロックに移動されます。この移動により、その行は、同時期に挿入された他の行とクラスター化されなくなります。

MDC 表のみに関する考慮事項

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。

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。

MDC 表および ITC 表に関する考慮事項

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ときと、ブロックから最後のレコードを削除するときだけです。したがって、ブロック索引に関連した、保守とロギングのための索引リソースおよび要件は、通常の索引に比べてずっと少なくなります。通常の索引をブロック索引に置き換えれば、置き換えの件数に比例して、保守とロギングのためのリソースと要件が大幅に削減されます。

最近空になったブロックを再使用する場合は、現在そのブロックが UR スキャナーによってスキャン中ではないことを確実にするために、ブロックに対して条件付き Z ロックを使用する必要があります。

マルチディメンション・クラスタリングおよび挿入時クラスタリングのエクステント管理

マルチディメンション・クラスタリング (MDC) 表、または挿入時クラスタリング (ITC) 表内部からのデータ・エクステントの解放は、表を再編成することによって行われます。

MDC 表および ITC 表の中のブロック・マップは、表に属するすべてのデータ・エクステントを追跡し、データを持つブロックおよびエクステントとデータを持たないブロックおよびエクステントを示します。データを持つブロックは「使用中」としてマーク付けされます。 MDC または ITC 表の削除または MDC 表のロールアウトが発生すると、ブロック・マップを持つブロック項目の「使用中」というマークが解除され、表による再使用のために解放されます。

ただし、表スペース内の他のオブジェクトはこれらのブロックとエクステントを使用できません。表を再編成することにより、これらの空きデータ・エクステントを表から解放することができます。 **RECLAIM EXTENTS** パラメーターを指定して **REORG TABLE** コマンドを使用すると、スペースを再利用しつつ、ユーザーに対して表を使用可能およびオンラインにすることができます。 MDC または ITC 表からエクステントを解放できるのは、表が DMS 表スペースに入っている場合だけです。

REORG TABLE コマンドで **RECLAIM EXTENTS** パラメーターを使用することにより、MDC または ITC 表での排他使用からエクステントを解放して、表スペース内の他のデータベース・オブジェクトがこのスペースを使用できるようにします。

また、このオプションを使用すると、エクステントが解放されるときの MDC または ITC 表への並行アクセスを制御できます。書き込みアクセスがデフォルトですが、読み取りアクセスおよびアクセス権限なしを並行アクセス制御として選ぶこともできます。

MDC または ITC 表が範囲パーティション化またはデータベース・パーティション化されている場合、デフォルトでは、すべてのデータ・パーティションまたはデータベース・パーティションでエクステントの解放が発生します。パーティション名 (データ・パーティションの場合) またはパーティション番号 (データベース・パーティションの場合) を指定することにより、特定のパーティションのエクステントだけを解放するようコマンドを実行できます。

エクステントの解放には、**REORG TABLE** コマンドと db2Reorg API のどちらでも使用可能です。

データベースの自動保守アクティビティーの一部として、自動的にエクステントを解放する操作が可能です。 MDC または ITC 表のエクステントを解放するために再編成を有効にするには、データベース構成パラメーター **auto_maint**、**auto_tbl_maint**、および **auto_reorg** の値がすべて ON でなければなりません。これらのデータベース構成パラメーターの構成は、コマンド行を使用して実行できます。データベース・パーティション・フィーチャーが有効になった DB2 インスタンスでは、カタログ・パーティションでパラメーターの構成を発行する必要があります。

使用されていないエクステントを解放するための MDC または ITC 表の自動再編成をいつ実行するかは、保守ポリシーによって制御されます。この保守ポリシーを設定するには DB2 システム・ストアード・プロシージャー AUTOMAINT_SET_POLICY および AUTOMAINT_SET_POLICYFILE を使用します。自動保守ポリシーは XML を使って保管されます。

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

マルチディメンション・クラスタリングとデータ・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マルチディメンション・クラスタリング (MDC) キーの両方で使用されます。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表に行うと、どちらかの機能を単独で使用するよりも、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 データウェアハウスの特性

以下の推奨事項は、DB2 V9.1 にとっては新しく典型的で主流であった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します。

- 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。

- パーティション・データベース環境が使用される (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単位にロードされる。
- 毎日の INGEST 量は、1 万から 1000 万以内のレコードである。
- データ量が変化する。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 (例えば 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。

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
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。
データの日次ロード (**LOAD** コマンドまたは **INGEST** コマンド)	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 を指定した **LOAD** コマンド、または **INGEST** コマンド)	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、表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、 38 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);
```

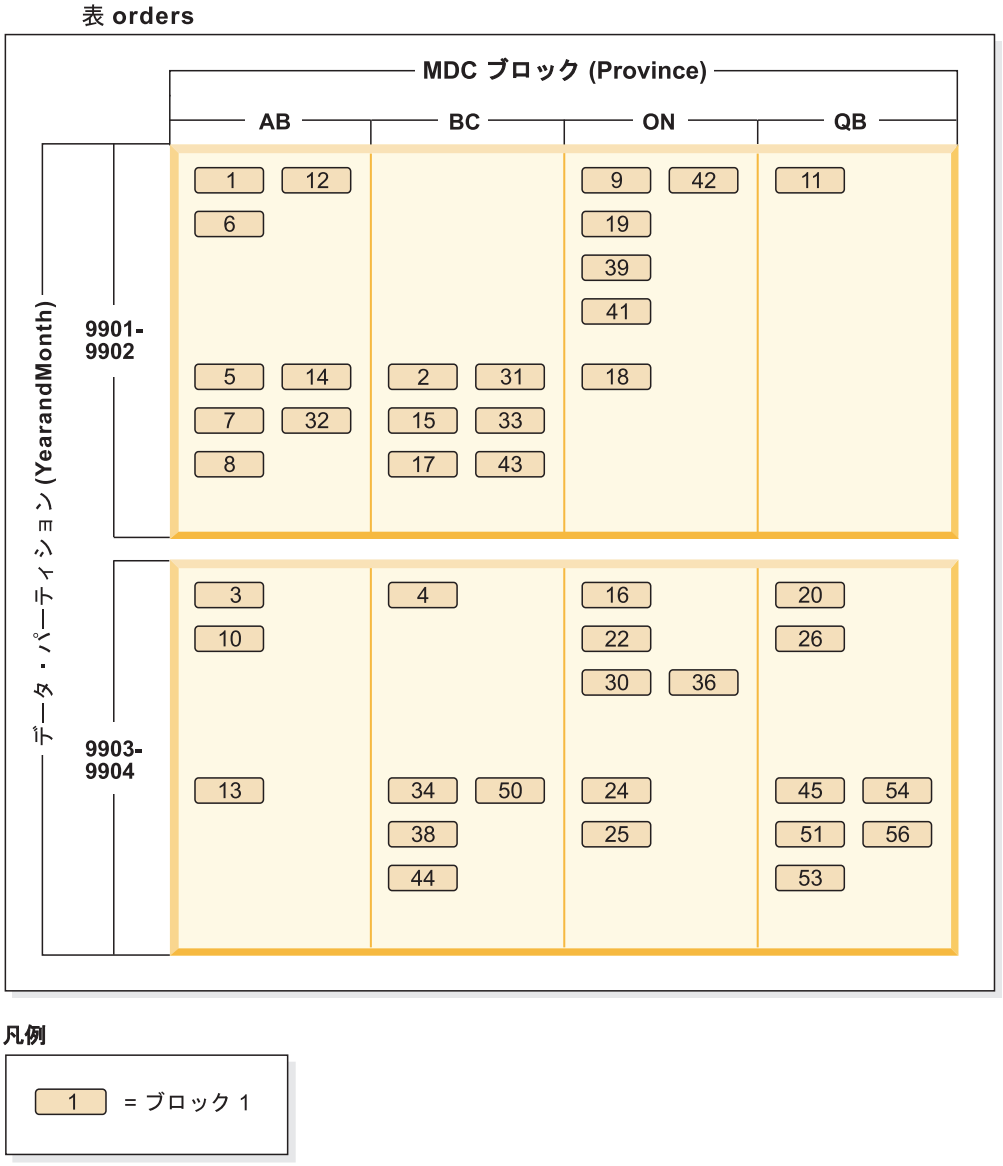


図 26. YearAndMonth によってパーティション化され、Province によって編成される表

例 2:

39 ページの図 7 に示されているように、YearAndMonth を ORGANIZE BY DIMENSIONS 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);
```

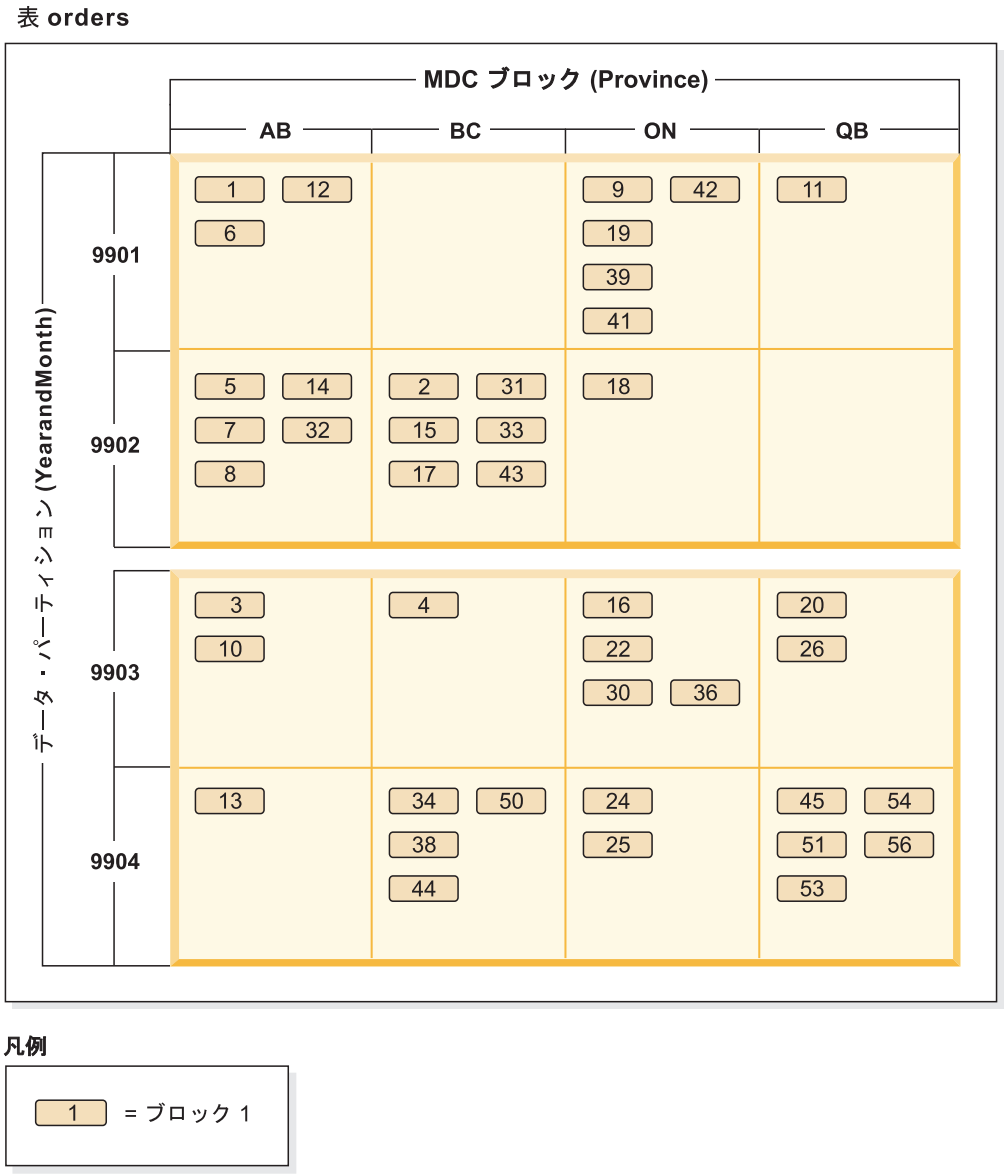


図 27. YearAndMonth によってパーティション化され、Province および YearAndMonth によって編成される表

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。
- DB2 V9.7 フィックスパック 1 以降のリリースで作成したデータ・パーティション化 MDC 表では、MDC ブロック索引はパーティション化されます。 DB2 V9.7 以前のリリースで作成したデータ・パーティション化 MDC 表では、MDC ブロック索引はパーティション化されません。

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。

これらの要因は互いに関連しています。データベースを物理的および論理的に設計する際には、これらすべてを考慮してください。 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* を使用できます。並列入出力とは、複数の入出力装置との間で書き込み/読み取り処理を同時に行うことです。その結果、スループットが大幅に改善される可能性があります。

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) を使って 1 つの照会の各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、SQL 照会など) 通常は 1 つの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、それらのほとんど (またはすべて) を 1 つのデータベース・パーティション内で並列して実行することができます。

88 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

を使用するには、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。

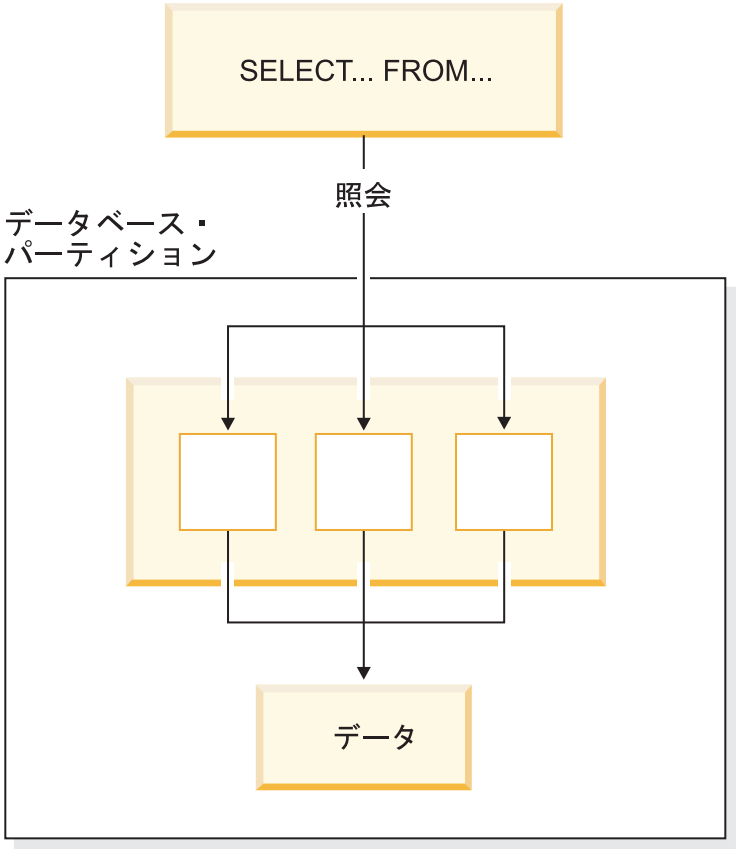


図 28. パーティション内並列処理

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、SQL 照会など) 通常は 1 つの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、1 つまたは複数のマシン上で、それらのほとんど (またはすべて) をパーティション・データベースの複数パーティションの間で並列して実行することができます。

89 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。

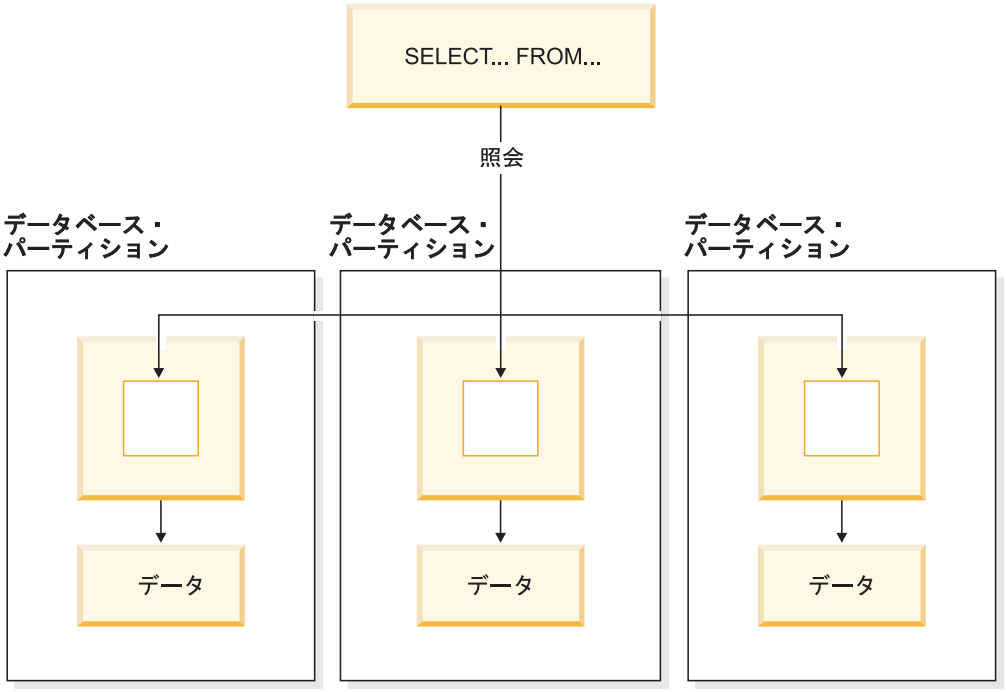


図 29. パーティション間並列処理

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。

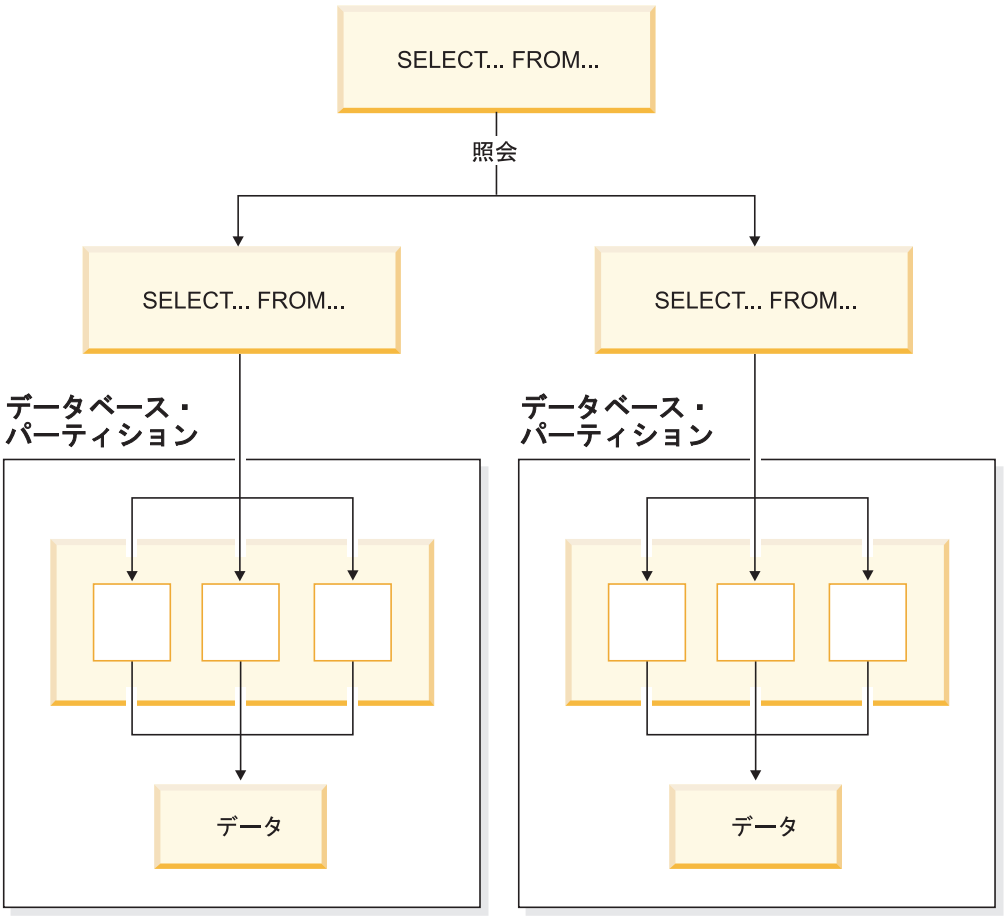


図 30. パーティション間並列処理とパーティション内並列処理の同時使用

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。また、パーティション間並列処理も利用できます。複数のデータベース・パーティションが存在する場合、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的に実行されます。

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・データベース環境では、**LOAD** コマンドを、表が存在する各データベース・パーティションで並列的に呼び出すことによって、パーティション内、パーティション間、および入出力の各並列処理を利用することができます。

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

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。

- *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。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、データの物理的な場所を認識しません。

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。また、ユーザーは、データの保管場所となる表スペースおよび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り、いくつの、そしてどのデータベース・パーティションに自分のデータを配分するかを決定できます。 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、大きな表の場合は複数パーティション・データベースにワークロードを分散させ、より小さい表は 1 つ以上のデータベース・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。`db2nodes.cfg` ファイルを手動で更新しないでください。手動更新するとエラー (SQL6031N) が戻されます。

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

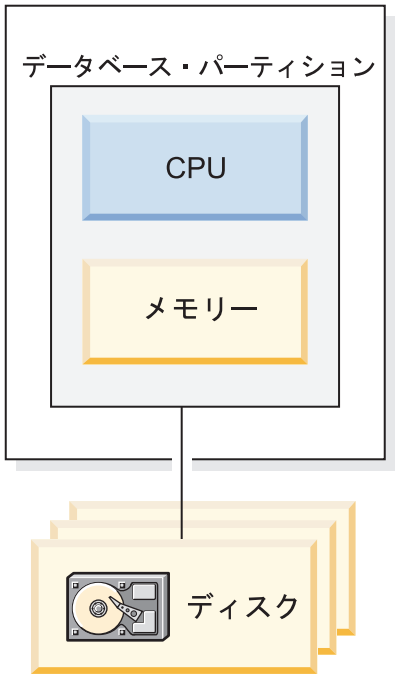


図 31. シングル・プロセッサー上での単一データベース・パーティション

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て、(メモリーやディスクなど) 他のコンポーネントを追加するかどうかに関わらず、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

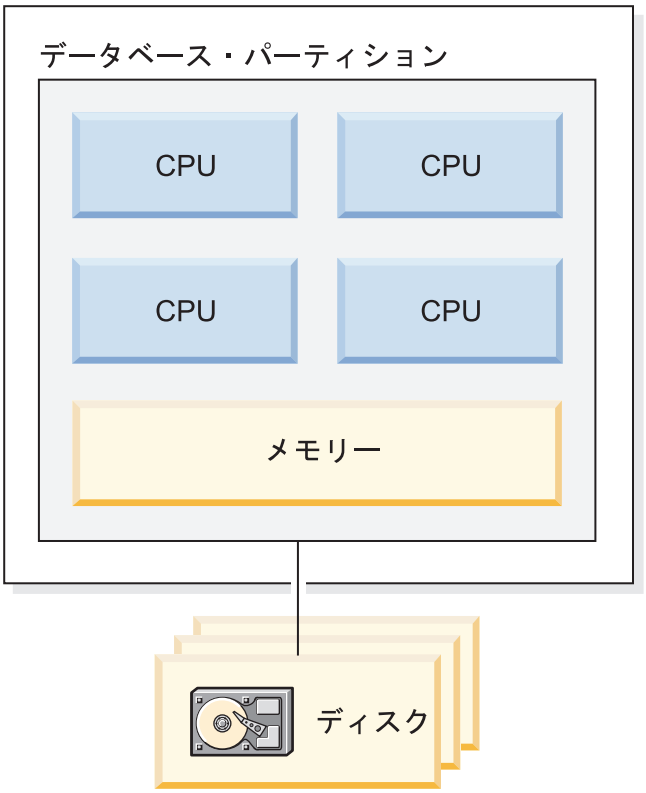


図 32. 単一パーティション・データベース対称型マルチプロセッサー環境

容量および拡張容易性

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 (96 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。

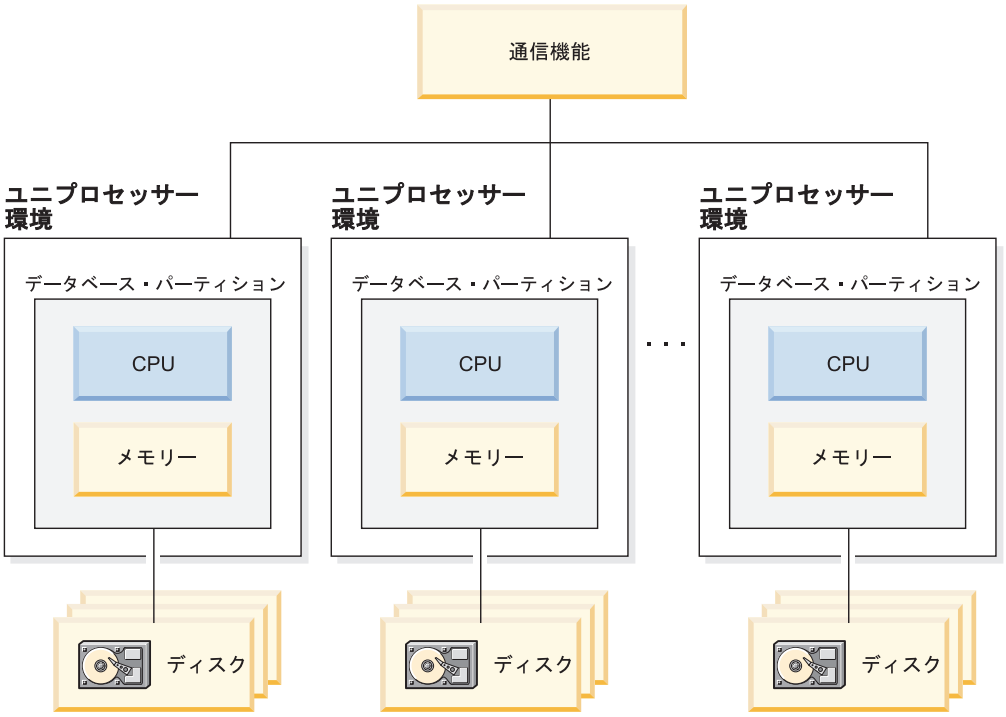


図 33. 超並列処理 (MPP) 環境

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では、最大数は 512 データベース・パーティションです。ただし、多数のマシンとインスタンスの管理に関する実質的な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 (97 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。

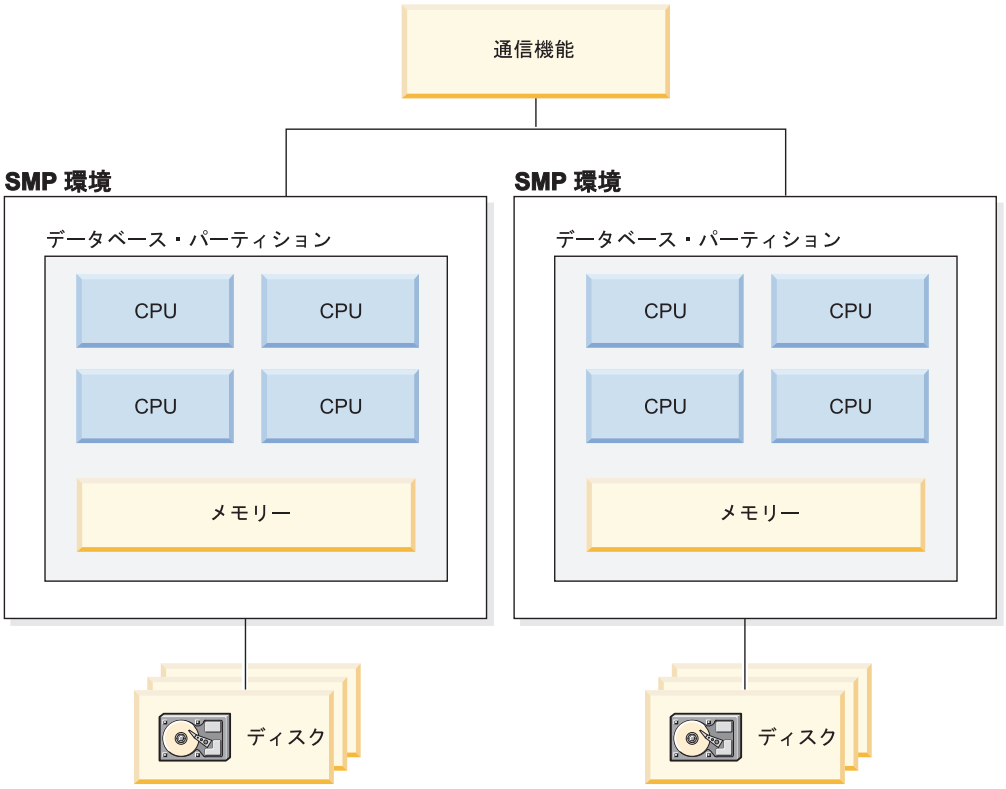


図 34. 複数の対称型マルチプロセッサー (SMP) 環境からなるクラスター

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・マネージャーに比べて、利用可能なリソースを最大限に使用できます。98 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。これは、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。

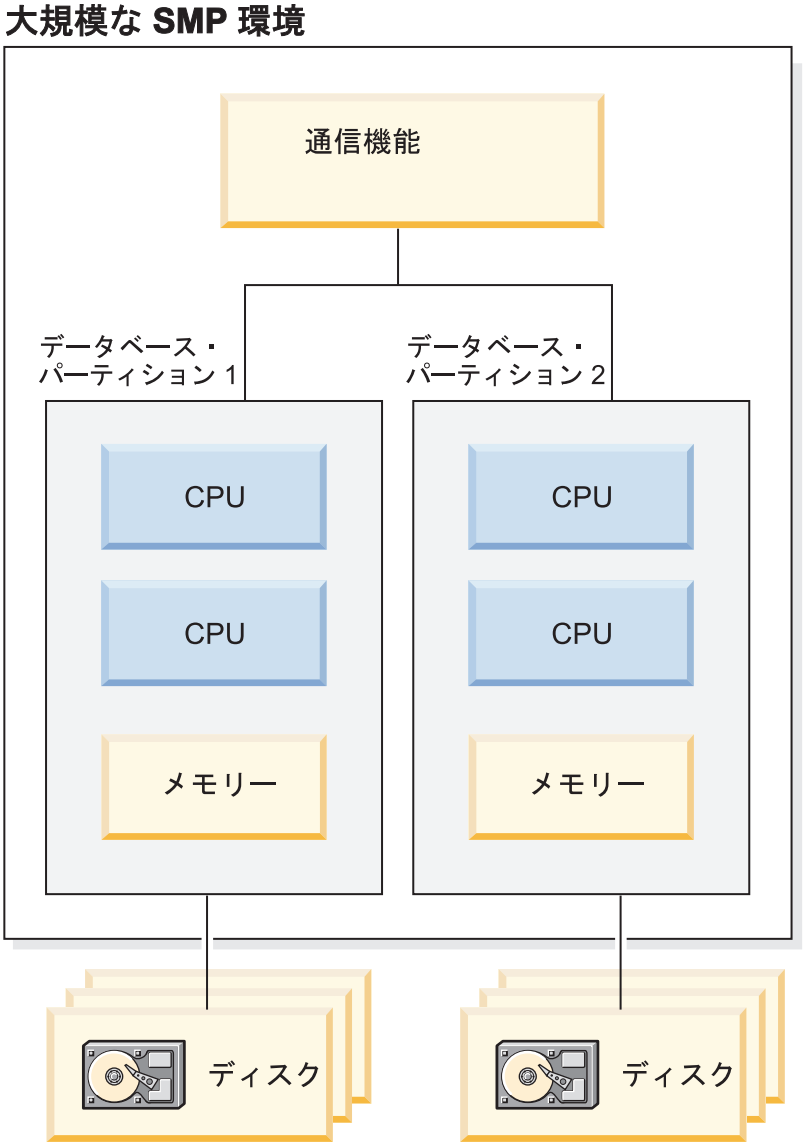


図 35. 対称型マルチプロセッサー環境でのパーティション・データベース

99 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。

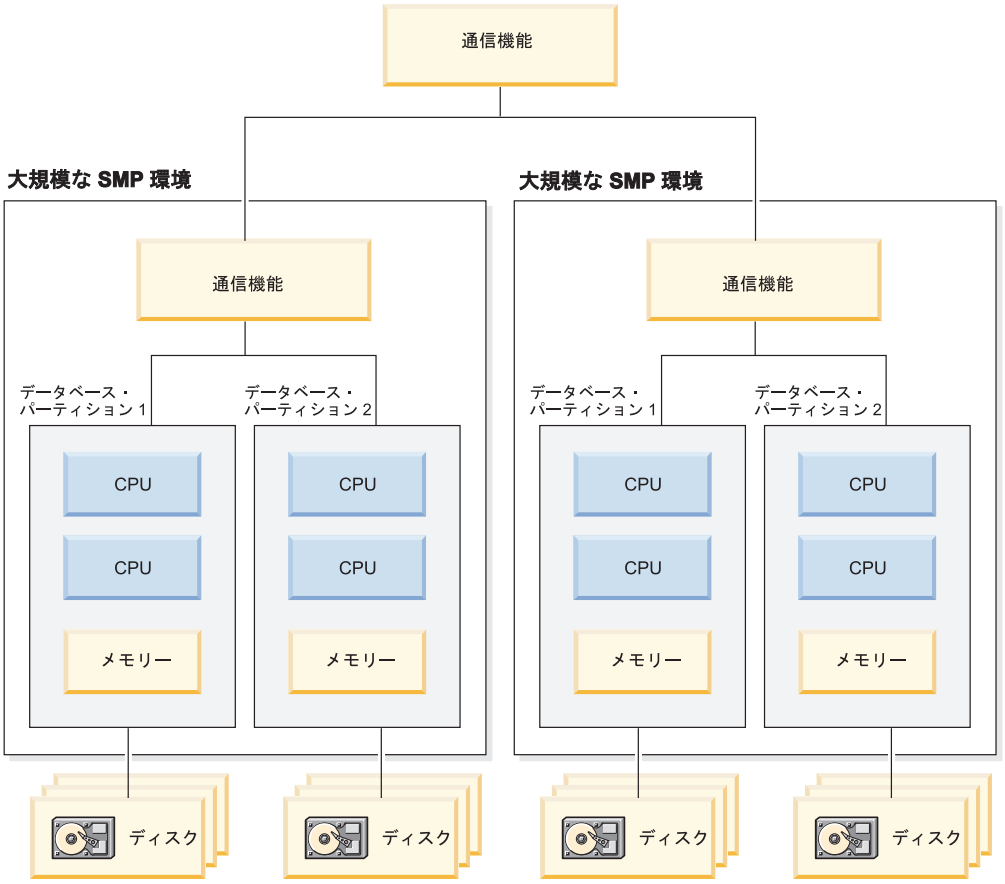


図 36. クラスターを構成する複数の対称型マルチプロセッサー環境を含むパーティション・データベース

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可能な並列処理の種類

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	はい	いいえ [1]	いいえ
単一データベース・パーティション、複数プロセッサー (SMP)	はい	はい	いいえ

表 8. それぞれのハードウェア環境で可能な並列処理の種類 (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)	はい	いいえ [1]	はい
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)	はい	はい	はい
論理データベース・パーティション	はい	はい	はい
[1] (いずれかの構成パラメーターを使って) 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・システムでも該当し、(入出力制約などにより) 照会が 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 セットアップ・ウィザードを使用した DB2 データベース・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。
- LDAP を使用して、DB2 サーバーを Windows オペレーティング・システムの 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・スキーマを拡張します。そうでない場合は、手動でノードを登録し、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 VS2010 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 VS2010 ランタイム・ライブラリーが必要です。 VS2010 ランタイム・ライブラリーは、Microsoft ランタイム・ライブラリーのダウンロード 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。
- DB2 製品を仮想ドライブまたはマップされていないネットワーク・ドライブ (例えば、Windows エクスプローラで *¥¥hostname¥sharename* と表示されるもの) からインストールすることはサポートされていません。 DB2 製品のインストールを試行する前に、ネットワーク・ドライブを Windows ドライブ名 (例えば、Z:) にマップする必要があります。

制約事項

- どのユーザー・アカウントでも、DB2 セットアップ・ウィザードの複数のインスタンスを実行することはできません。
- 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 コピー名は、文字 A から Z、a から z および 0 から 9 で構成される 64 英文字に制限されています。
- 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 か Windows 2008、またはそれ以降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 コピーの更新、既存の DB2 コピーへの機能追加、既存の DB2 バージョン 9.5 またはバージョン 9.7 のコピーのアップグレード、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。
7. DB2 セットアップ・ウィザードを使用する際のサンプル・パネルからインストール・プロセスに進みます。関連リンクを参照してください。

タスクの結果

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。

次のタスク

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

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 2012.log)。

これが Vista 64 ビット上の新しい DB2 製品インストールであり、32 ビットの OLE DB プロバイダーを使用する予定の場合は、IBMDADB2 DLL を手動で登録する必要があります。この DLL を登録するには、次のコマンドを実行します。

```
c:¥windows¥SysWOW64¥regsvr32 /s c:¥Program_Files¥IBM¥SQLLIB¥bin¥ibmdadb2.dll
```

Program_Files は Program Files ディレクトリーの場所を表します。

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 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 セットアップ・ウィザードを実行して、IBM Data Studio をインストールすることができます。

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 64 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 9.5 または 9.7 からアップグレードする際の重要な注意事項:

- 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 データベース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。

始める前に

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。

手順

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. 1 次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 データベース製品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 データベース・システム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、特権は既にアカウントに明示的に追加されています。

高速コミュニケーション・マネージャー (Windows)

複数のメンバーからなる環境では、各メンバーに、エージェント要求に関係するメンバー間の通信をサポートする一対の FCM デーモンがあります。1 つは通信を送信するためのデーモン、もう 1 つは受信するためのデーモンです。これらのデーモンとサポート・インフラストラクチャーは、インスタンスの開始時にアクティブにされます。FCM 通信は、同じメンバー内で動作するエージェントにも使用されます。このタイプの通信は、メンバー内通信としても知られています。

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** を使用して指定できます。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** を使用して指定できます。デフォルトでは、**fcm_num_buffers** および **fcm_num_channels** データベース・マネージャー構成パラメーターは AUTOMATIC に設定されます。 AUTOMATIC に設定されている場合 (これが推奨される設定です)、FCM はリソースの使用状況をモニターし、ワークロードの需要に対応できるようにリソースを調整します。

DB2 データベース・サーバー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。

手順

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 のアップグレード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
   - 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 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

DB2 データベース製品をインストールする方式は複数あります。それぞれのインストール方式は特定の環境に適したものです。

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	はい	はい
応答ファイル・インストール	はい	はい
db2_install コマンド	いいえ	はい
ペイロード・ファイルのデプロイメント	いいえ	はい

重要: コマンド**db2_install** は推奨されておらず、今後のリリースで除去される可能性があります。代わりに、**db2setup** コマンドまたは応答ファイルによるインストール方式を使用してください。

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 データベース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 オペレーティング・システム上の非 root 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 root インスタンスを自動的に作成します。

Linux および UNIX オペレーティング・システムでは、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
応答ファイル生成プログラムを使用して、既存のインストールを複製する応答ファイルを作成することができます。例えば、IBM Data Server Client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・ウィザードを使用する方法
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 データベース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 データベース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。 **db2cfimp** コマンドを使用して、プロファイルをインポートします。 **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 データベース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 データベース製品 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 オペレーティング・システムのみ)

db2_install コマンドは、指定した DB2 データベース製品のすべての 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クリアすることはできません。

db2_install コマンドは、指定した DB2 データベース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。インストール中に DB2 データベース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。

要確認: コマンド **db2_install** は推奨されておらず、将来のリリースで削除される予定です。

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 データベース製品のインストール・パッケージは、Linux および UNIX 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 データベース製品のインストール環境とのインターフェースや照会に使用する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 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。
- サポートされるブラウザーがインストールされていることを確認します。
- DB2 データベース・サーバーは、root 権限と非 root 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 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

- どのユーザー・アカウントでも、DB2 セットアップ・ウィザードの複数のインスタンスを実行することはできません。
- 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。
- Itanium ベースの HP Integrity Series システム上の HP-UX 11i V2 の場合、DB2 インスタンス所有者のセットアップ・ウィザードで作成されたユーザー、fenced ユーザー、または DAS には DB2 セットアップ・ウィザードで指定されたパスワードを使ってアクセスすることはできません。セットアップ・ウィザードが終了した後、それらのユーザーのパスワードを再設定する必要があります。これは、セットアップ・ウィザードを使ったインスタンスまたは DAS の作成には影響しません。したがって、インスタンスまたは DAS を再作成する必要はありません。

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`) 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。追加された最新の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をお勧めします。
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。

6. DB2 セットアップ・ウィザードを使用する際のサンプル・パネルからインストール・プロセスに進みます。関連リンクを参照してください。

インストールを開始した後、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。

タスクの結果

非 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 root ユーザーのホーム・ディレクトリーを表します。

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/V10.1

Linux /opt/ibm/db2/V10.1

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。
  - 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 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 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。
  - 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 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 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。

次のタスク

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

DB2 セットアップ・ウィザードを実行して、IBM Data Studio をインストールすることができます。

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 DB2 関連製品の資料が収録されています。

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 64 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 9.5 または 9.7 からアップグレードする際の重要な注意事項:

- バージョン 9.5 または 9.7 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アップグレード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、パーティション・データベース環境の通信サポートを提供します。

複数のメンバーからなる環境では、各メンバーに、エージェント要求に関係するメンバー間の通信をサポートする一対の FCM デーモンがあります。1 つは通信を送信するためのデーモン、もう 1 つは受信するためのデーモンです。これらのデーモンとサポート・インフラストラクチャーは、インスタンスの開始時にアクティブにされます。FCM 通信は、同じメンバー内で動作するエージェントにも使用されます。このタイプの通信は、メンバー内通信としても知られています。

FCM デーモンは、通信アクティビティーに関する情報を収集します。FCM 通信に関する情報は、 データベース・システム・モニターを使用することによって取得できます。メン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 デーモンはこの情報でモニター・エレメントを更新します。このイベントに対し、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** を使用して指定できます。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** を使用して指定できます。デフォルトでは、**fcm_num_buffers** および **fcm_num_channels** データベース・マネージャー構成パラメーターは AUTOMATIC に設定されます。 AUTOMATIC に設定されている場合 (これが推奨される設定です)、FCM はリソースの使用状況をモニターし、ワークロードの需要に対応できるようにリソースを調整します。

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。

手順

以下のようにして、AIX 環境設定を更新します。

1. root ユーザー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
   ```

   各要素の意味は以下のとおりです。

   - *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 データベース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 Enterprise Server Edition 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 (1 台のコンピューター上の biod 数) × # of computers in the instance (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、上記の公式に従えば 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。

複数の AIX ノードにコマンドを配布する一括作業のセットアップ

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 System p® 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。

始める前に

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 ユーザー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している必要があります。

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する必要があります。

手順

以下のようにして、ワークステーション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. 一括作業に関与するすべてのワークステーションのホスト名をリストする、`nodelist.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 つのワークステーションを指定して、一括作業を作成しようとしているとします。 nodelist.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 DSH_NODE_LIST=path/nodelist.txt
   ```

   ここで *path* は、nodelist.txt が作成された場所になります。 nodelist.txt は、一括作業に組み込まれた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 /nodelist.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。

手順

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・システムの場合:

  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 データベース製品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・システムの場合:

  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 データベース製品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・システムの場合:

  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 データベース製品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。

始める前に

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 ユーザー権限が必要です。

手順

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例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、1 次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。

次のタスク

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 データベース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・データベース・システム用のファイル・システムの作成 (Linux)

Linux オペレーティング・システムでのパーティション・データベース・システムのセットアップの一部として、DB2 ホーム・ファイル・システムを作成する必要が

あります。次に、ホーム・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・システムに関与する各コンピューターからそれをマウントする必要があります。

このタスクについて

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。

手順

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。

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/exportfs -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. 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。 以下のコマンドを入力します。

   ```
   mount /db2home
   ```

   mount 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。例えば、

   ```
   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。

タスクの結果

これらのステップを実行して、以下のタスクを完了します。

1. パーティション・データベース環境内の 1 つのコンピューターで、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。
2. 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。
3. 関与するコンピューターそれぞれに、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)

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として、DB2 ホーム・ファイル・システムを作成する必要があります。次に、ホーム・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・システムに関与する各コンピューターからそれをマウントする必要があります。

始める前に

DB2 データベース製品 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 (KB 単位) を検査することができます。

```
du -sk DVD_mounting_point
```

DB2 インスタンスは、最低 200 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 データベース製品 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置かれているボリューム・グループを作成済みであることが必要です。

手順

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。

1. DB2 ホーム・ファイル・システムを作成します。

   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。

   a. **smit jfs** コマンドを入力します。
   b. 「**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。
   c. 「**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。
   d. そのファイル・システムが物理的に常駐す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。
   e. 「**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。
   f. 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。
   g. 「**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。
   h. 「**OK**」をクリックします。

2. DB2 ホーム・ファイル・システムをエクスポートします。

/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。

3. ログアウトします。
4. 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・システムをマウントします。

   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。

   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. 「**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。
   d. 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。
   e. 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。
   f. 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。

g. 「**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。
h. 「**/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。
i. 「**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。
j. 「**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「ハード (hard)」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、システムが破損した場合に問題が生じることがあります。このフィールドを「ハード (hard)」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。

k. 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。 これがデフォルトの設定です。
l. 「**OK**」をクリックします。
m. ログアウトします。

DB2 pureScale Feature のインストールに必要なユーザー (Linux)

Linux オペレーティング・システム上の DB2 データベース環境を運用するには、2 つのユーザーおよびグループが必要です。

始める前に

- ユーザーおよびグループを作成するためには、root ユーザー権限が必要です。
- 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。

このタスクについて

DB2 pureScale インスタンスを作成するには、次の 2 ユーザーが必要です。

- インスタンス所有者としての 1 ユーザー
- fenced ユーザーとしての 1 ユーザー

2 ユーザーをそれぞれ、異なる 2 つのグループで使用する必要があります。 2 ユーザーそれぞれの UID、GID、グループ名、ホーム・ディレクトリーが、すべてのホストで同じでなければなりません。使用するユーザーのいずれかがホストのいずれかに存在する場合は、プロパティーが一致しなければなりません。これらの必要なユーザーをインストール開始前に作成する必要はありません。これらのユーザーは、DB2 セットアップ・ウィザードのパネルを進んでいく過程で作成することも、応答ファイルで指定することもできます。既存のユーザーを使用する場合は、すべてのホストに存在し、ここに記載した要件を満たしていなければなりません。

この後の手順で使用するユーザー名とグループ名はデフォルトです。これらを次の表に示します。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。

表 10. デフォルトの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	db2sdin1	db2iadm1
fenced ユーザー	db2sdfe1	db2fadm1

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 データベース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。

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 データベース・システムの命名規則に沿ったものでなければなりません。

各ホストに作成する同一ユーザー名の HOME ディレクトリーは、同じでなければなりません。ただし、ユーザー名はどのホストにもまだ存在していてはなりません。既存のユーザー名を使用する場合は、そのユーザー名がすべてのホストに存在し、ユーザー ID (uid)、グループ ID (gid)、グループ名、HOME ディレクトリーが同じでなければなりません。

手順

これらのユーザーを作成するには、以下のようなステップを実行します。

1. ホストにログオンします。
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) と、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) を作成します。

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

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有した 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
```

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
```

5. ログアウトします。
6. 作成した各ユーザー (db2inst1 および db2fenc1) 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. 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)

AIX オペレーティング・システム上のパーティション・データベース環境で DB2 データベースを操作するには、3 つのユーザーおよびグループが必要です。

始める前に

- ユーザーおよびグループを作成するためには、root ユーザー権限が必要です。
- 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。

このタスクについて

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 データベース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

表 11. 必要なユーザーおよびグループ (続き)

必要なユーザー	ユーザー名	グループ名
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。

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 データベース・システムの命名規則に沿ったものでなければなりません。

手順

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有した 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 root 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 InfoSphere® 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に関する考慮事項』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 と 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。

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。
   - 「**インストールおよび応答ファイルの作成を選択**」ウィンドウで、「**インストール設定を応答ファイルに保存する**」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)
7. Windows 2000 またはそれ以降のパーティション・データベース環境の場合、DB2 リモート・コマンド・サービス・セキュリティー・フィーチャーを開始して、データとリソースを保護します。

   完全にセキュア化するには、コンピューター (サービスが LocalSystem アカウントのコンテキストのもとで実行される場合) またはユーザー (サービスがユーザーのログオン・コンテキストのもとで実行される場合) を委任に対して開始します。

   DB2 リモート・コマンド・サービス・セキュリティー・フィーチャーを開始するには、次のようにします。

   a. ドメイン・コントローラーで「Active Directory ユーザーとコンピュータ」ウィンドウをオープンします。つまり、「**スタート**」をクリックし、「**プログラム**」 > 「**管理ツール**」 > 「**Active Directory ユーザーとコンピュータ**」を選択します。

b. 右側のウィンドウ・パネルで、コンピューターまたはユーザーを右クリックして開始し、「**プロパティ**」を選択します。

c. 「**全般**」タブをクリックし、「**コンピュータを委任に対して信頼する**」チェック・ボックスを選択します。ユーザーの設定の場合には、「**アカウント**」タブをクリックして、「**アカウント オプション**」グループ内の「**アカウントは委任に対して信頼されている**」チェック・ボックスを選択します。「**アカウントは重要なので委任できない**」ボックスがチェックされていないことを確認します。

d. 「**OK**」をクリックして、コンピューターまたはユーザーを委任に対して開始します。

開始する必要のあるコンピューターまたはユーザーごとに、上記のステップを繰り返します。セキュリティーに関する変更を有効にするには、コンピューターを再始動する必要があります。

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 (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。

始める前に

- 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。

手順

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 データベース製品 DVD が入っているディレクトリーに変更します。 例:

   ```
   cd c:¥db2dvd
   ```

   ここで、db2dvd は、DB2 データベース製品 DVD が入っている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 データベース製品は、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。

タスクの結果

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。

次のタスク

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。

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。

始める前に

- 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 データベース製品をインストールし、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が作成されている必要があります。
- 関与するコンピューターに対して root ユーザー権限を持っている必要があります。

手順

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 データベース製品 DVD の内容をコピーしたディレクトリーに移動します。 例えば、

   ```
   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 は /db2home ディレクトリーに保存されています。この状態のコマンドは、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。

タスクの結果

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。

次のタスク

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 データベースの資料に 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 データベース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

インストールの検証

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 データベース・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・データを取得し、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。

始める前に

すべてのインストール・ステップを完了していること。

手順

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。

次のタスク

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)

db2val ツールを使用して、インストール・ファイル、インスタンス、データベース作成、そのデータベースへの接続、およびパーティション・データベース環境の状態を検証することにより、DB2 コピーの中核となる機能を検査します。

詳しくは、『DB2 コピーの検証』を参照してください。少なくとも 2 つのノードがある場合にのみ、パーティション・データベース環境の状態が検証されます。さらに、DB2 データベース・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・データを取得し、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。

始める前に

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。

手順

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

次のタスク

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。

このタスクについて

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。db2nodes.cfg ファイルはデータベース・パーティション・サーバー構成ファイルです。

Windows オペレーティング・システム環境の場合を除いて、データベース・パーティション・サーバー構成ファイル (db2nodes.cfg) の内容を表示および更新するために任意のエディターを使用できます。 Windows オペレーティング・システム環境では、**db2ncrt** および **db2nchg** コマンドを使ってデータベース・パーティション・サーバー構成ファイルを作成および変更します。

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有されます。

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 **start_stop_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。

このタスクについて

データベース・パーティションを使用可能にするためには、データベース・マネージャーを開始する前に、db2nodes.cfg ファイルが、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に配置されている必要があります。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有されます。

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。例えば、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。

各ホスト名について、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。この動作は、**db2profile** スクリプト内の **DB2NODE** 環境変数、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。

次の例は、あるシステムの場合に考えられるノード構成ファイルを示しています。このシステムでは、SP2EN1 が複数の TCP/IP インターフェースと 2 つの論理パーティションを持ち、DB2 データベース・インターフェースとして SP2SW1 を使用します。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 **START DBM** を出したときにロックされ、**STOP DBM** 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、**START DBM** 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 **START DBM** を発行することができます

注: STOP DBM 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 **STOP DBM 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 インスタンスに参加するデータベース・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も、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。

dbpartitionnum hostname logicalport netname resourcesetname

dbpartitionnum, *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。

dbpartitionnum hostname computername logicalport netname resourcesetname

Windows オペレーティング・システムでは、**db2ncrt** または START DBM ADD DBPARTITIONNUM コマンドによって db2nodes.cfg にこれらの項目が追加されます。項目は **db2nchg** コマンドによって変更することもできます。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。

dbpartitionnum

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 *dbpartitionnum* 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、*dbpartitionnum* 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。正規のホスト名を強く 推奨します。

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権限を付与します。

```
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 を共有し、MLN 当たり 24% の CPU を共有し、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、**numactl** 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. 以下のコマンドを発行します。

```
$ 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、データベース・パーティション構成ファイル db2nodes.cfg から取得されます。

このタスクについて

注: Windows の場合、データベース・パーティション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。

手順

db2nodes.cfg に含まれているコンピューターのリストをオーバーライドするには、以下のようにします。

- 環境変数 **RAHOSTFILE** をエクスポート (Linux および UNIX オペレーティング・システムの場合) 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を格納するファイルのパス名を指定します。
- 環境変数 **RAHOSTLIST** をエクスポート (Linux および UNIX オペレーティング・システムの場合) 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します。

  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。

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。

このタスクについて

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 オペレーティング・システムでは、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  (for Korn shell syntax)
```

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。 **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 **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)

DB2 パーティション・データベース環境では、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。

始める前に

- 関与するコンピューターのすべてに DB2 データベース製品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。

このタスクについて

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 データベース・システム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
```

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。

手順

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。 例えば、この一連のステップでは db2inst1 がインスタンス所有者です。
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。

複数の論理パーティションのセットアップ

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方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、それらのパーティションが同じインスタンスに含まれるならば、「そのコンピューターは*複数の論理パーティション* を実行している」といいます。それらのパーティションが複数の異なるインスタンスに含まれる場合には、このコンピューターは複数の論理パーティションを提供していません。

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 つだけ用意されている、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)、 **START DBM DBPARTITIONNUM** 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使用できることです。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。

このタスクについて

- 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 内で論理データベース・パーティションを構成するには、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
```

IBM DB2 pureScale Feature では、"nodenumber 0" となっているメンバーが存在することを確認してください。

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。

このタスクについて

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。

手順

- データのロード中に並列処理を使用可能にするには、以下のようにします。

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には、以下のようにします。
  - 複数の宛先メディアを使用します。
  - 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。
  - **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。
  - **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。
    - バックアップする表スペースの大きさを、少なくとも (エクステント・サイズ * コンテナー数) の積の最大の大きさにする。
- 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。
  - リストアする表スペースの大きさを、少なくとも (エクステント・サイズ * コンテナー数) の積の最大の大きさにする。
  - 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

照会のパーティション内並列処理を使用可能にするには、1 つ以上のデータベース構成パラメーターまたはデータベース・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。あるいは、 CREATE ステートメントまたは ALTER WORKLOAD ステートメントで MAXIMUM DEGREE オプションを使用するか、ADMIN_SET_INTRA_PARALLEL プロシージャーを使用して、トランザクション・レベルにおいてパーティション内並列処理を使用可能または使用不可にします。

このタスクについて

特定のデータベースまたはインスタンス構成ファイルに含まれる個々の項目の値を検出するには、**GET DATABASE CONFIGURATION** コマンドまたは **GET DATABASE MANAGER CONFIGURATION** コマンドを使用します。それらの項目を 1 つ以上修正するには、**UPDATE DATABASE CONFIGURATION** コマンドまたは **UPDATE DATABASE MANAGER CONFIGURATION** コマンドを使用します。

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。デフォルトは NO で、この場合、このインスタンス内のアプリケーションはパーティション内並列処理なしで実行されます。例えば、以下のようにします。

```
update dbm cfg using intra_parallel yes;
get dbm cfg;
```

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実行するときにこの値を超える値は使用しません。デフォルトは -1 で、この場合には、システムでは、ユーザー指定の値ではなく、オプティマイザーによって決められたパーティション内並列処理の度合いが使用されます。例えば、以下のようにします。

```
update dbm cfg using max_querydegree any;
get dbm cfg;
```

さらに、**max_querydegree** の値を使用するためには、**intra_parallel** データベース・マネージャー構成パラメーターを YES に設定する必要もあります。

dft_degree
DEGREE プリコンパイル・オプションまたは 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値を指定するデータベース構成パラメーター。デフォルトは 1 です。値 -1 (または ANY) は、システムではオプティマイザーによって決められたパーティション内並列処理の度合いが使用されることを意味します。例えば、以下のようにします。

```
connect to sample;
update db cfg using dft_degree -1;
get db cfg;
connect reset;
```

DEGREE 対称型マルチプロセッシング (SMP) システムで静的 SQL ステートメントを実行するためのパーティション内並列処理の度合いを指定する、プリコンパイル・オプションまたは BIND オプション。例えば、以下のようにします。

```
connect to prod;
prep demoapp.sqc bindfile;
bind demoapp.bnd degree 2;
...
```

CURRENT DEGREE
動的 SQL ステートメントを実行するためのパーティション内並列処理の度合いを指定する特殊レジスター。SET CURRENT DEGREE ステートメントを使用して、CURRENT DEGREE 特殊レジスターに値を割り当てます。例えば、以下のようにします。

```
connect to sample;
set current degree = '1';
connect reset;
```

さらに、パーティション内並列処理を使用するには、**intra_parallel** データベース・マネージャー構成パラメーターを YES に設定する必要があります。NO に設定すると、この特殊レジスターの値は無視され、ステートメントではパーティション内並列処理は使用されません。**intra_parallel** データベース・マネージャー構成パラメーターと CURRENT DEGREE 特殊レジスターの値は、MAXIMUM DEGREE ワークロード属性を設定するとワークロード内でオーバーライドできます。

MAXIMUM DEGREE
ワークロードの並列処理の最大実行時度合いを指定する CREATE WORKLOAD ステートメント (または ALTER WORKLOAD ステートメント) のオプション。

例えば、bank_trans が主に短い OLTP トランザクションを実行するパッケージ・アプリケーションで、bank_report がビジネス・インテリジェンス (BI) レポートを生成する複合照会を実行する別のパッケージ・アプリケーションであるとします。どちらのアプリケーションも変更できず、両方ともデータベースに対して度合い 4 でバインドされています。bank_trans が実行中の場合、ワークロード trans に割り当てられ、この場合にはパーティション内並列処理は使用できません。この OLTP アプリケーションは、パーティション内並列処理のオーバーヘッドに関連した性能低下を生じさせることなく実行されます。bank_report が実行中の場合、ワークロード bi に割り当てられ、その場合には、パーティション内並列処理が使用可能になり、最大実行時度合いは 8 に指定されます。パッケージのコンパイル度合いは 4 なので、このアプリケーション内の静的 SQL ステートメントが実行される場合の度合いは 4 にしかなりません。この BI アプリケーションに動的 SQL ステートメントが含まれる場合、CURRENT DEGREE 特殊レジスターが設定されていないと、これらのステートメントは度合い 8 で実行されます。

```
connect to sample;

create workload trans
  applname('bank_trans')
  maximum degree 1
  enable;

create workload bi
  applname('bank_report')
  maximum degree 8
  enable;

connect reset;
```

ADMIN_SET_INTRA_PARALLEL

データベース・アプリケーションのパーティション内並列処理を使用可能にしたり使用不可にしたりするプロシージャー。このプロシージャーは現行トランザクション内で呼び出されますが、次のトランザクションから適用されます。例えば、以下のコードが、静的 SQL ステートメントと動的 SQL ステートメントの両方を持つ ADMIN_SET_INTRA_PARALLEL プロシージャーを使用する demoapp アプリケーションの一部であるとします。

```
EXEC SQL CONNECT TO prod;

// Disable intra-partition parallelism:
EXEC SQL CALL SYSPROC.ADMIN_SET_INTRA_PARALLEL('NO');
// Commit so that the effect of this call
// starts in the next statement:
EXEC SQL COMMIT;

strcpy(stmt, "SET CURRENT DEGREE='1'");
// Set the degree of parallelism to 1:
EXEC SQL EXECUTE IMMEDIATE :stmt;

// All statements in the next two transactions run
// without intra-partition parallelism:
strcpy(stmt, "SELECT deptname FROM org");
EXEC SQL PREPARE rstmt FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR rstmt;
EXEC SQL OPEN c1;
EXEC SQL FETCH c1 INTO :deptname;
EXEC SQL CLOSE c1;
...
```

```
// New section for this static statement:
EXEC SQL SELECT COUNT(*) INTO :numRecords FROM org;
...
EXEC SQL COMMIT;

// Enable intra-partition parallelism:
EXEC SQL CALL SYSPROC.ADMIN_SET_INTRA_PARALLEL('YES');
// Commit so that the effect of this call
// starts in the next statement:
EXEC SQL COMMIT;

strcpy(stmt, "SET CURRENT DEGREE='4'");
// Set the degree of parallelism to 4:
EXEC SQL EXECUTE IMMEDIATE :stmt;

// All dynamic statements in the next two transactions
// run with intra-partition parallelism and degree 4:
strcpy(stmt, "SELECT deptname FROM org");
EXEC SQL PREPARE rstmt FROM :stmt;
EXEC SQL DECLARE c2 CURSOR FOR rstmt;
EXEC SQL OPEN c2;
EXEC SQL FETCH c2 INTO :deptname;
EXEC SQL CLOSE c2;
...
// All static statements in the next two transactions
// run with intra-partition parallelism and degree 2:
EXEC SQL SELECT COUNT(*) INTO :numRecords FROM org;
...
EXEC SQL COMMIT;
```

動的 SQL ステートメントのパーティション内並列処理の度合いは CURRENT DEGREE 特殊レジスターを介して指定され、静的 SQL ステートメントの場合には DEGREE BIND オプションを介して指定されます。以下のコマンドは、demoapp アプリケーションを準備してバインドするために使用します。

```
connect to prod;
prep demoapp.sqc bindfile;
bind demoapp.bnd degree 2;
...
```

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、論理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。単一パーティション・データベース構成で複数の

プロセッサーを使用する場合 (SMP)、プロセッサーはメモリーとストレージ・システム・リソースを共有します。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、システム間の通信や、システム間のタスクの調整はワークロードを考慮する上での要因にはなりません。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。

増設コンピューターの追加

パーティション・データベース環境が既に存在する場合、増設のコンピューター (シングル・プロセッサーまたはマルチプロセッサー) およびストレージ・リソースを環境に追加することにより、処理能力とデータ・ストレージ容量を増強することができます。メモリー・リソースとストレージ・リソースはコンピューター間で共有されません。この方法を選択した場合、ストレージとコンピューターの全体にわたってデータとユーザー・アクセスのバランスが取れるという利点があります。

新しいコンピューターとストレージを追加した後、**START DATABASE MANAGER** コマンドを使用して、新規データベース・パーティション・サーバーを新しいコンピューターに追加することになります。追加したそれぞれの新規データベース・パーティション・サーバー上のインスタンス内のデータベースごとに、新規データベース・パーティションが作成されて構成されます。多くの場合、新規データベース・パーティション・サーバーの追加後にインスタンスを再始動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

高速コミュニケーション・マネージャー (Windows)

複数のメンバーからなる環境では、各メンバーに、エージェント要求に関係するメンバー間の通信をサポートする一対の FCM デーモンがあります。1 つは通信を送信するためのデーモン、もう 1 つは受信するためのデーモンです。これらのデーモンとサポート・インフラストラクチャーは、インスタンスの開始時にアクティブに

されます。FCM 通信は、同じメンバー内で動作するエージェントにも使用されます。このタイプの通信は、メンバー内通信としても知られています。

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** を使用して指定できます。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** を使用して指定できます。デフォルトでは、**fcm_num_buffers** および **fcm_num_channels** データベース・マネージャー構成パラメーターは AUTOMATIC に設定されます。 AUTOMATIC に設定されている場合 (これが推奨される設定です)、FCM はリソースの使用状況をモニターし、ワークロードの需要に対応できるようにリソースを調整します。

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、パーティション・データベース環境の通信サポートを提供します。

複数のメンバーからなる環境では、各メンバーに、エージェント要求に関係するメンバー間の通信をサポートする一対の FCM デーモンがあります。1 つは通信を送信するためのデーモン、もう 1 つは受信するためのデーモンです。これらのデーモンとサポート・インフラストラクチャーは、インスタンスの開始時にアクティブにされます。FCM 通信は、同じメンバー内で動作するエージェントにも使用されます。このタイプの通信は、メンバー内通信としても知られています。

FCM デーモンは、通信アクティビティーに関する情報を収集します。FCM 通信に関する情報は、 データベース・システム・モニターを使用することによって取得できます。メン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 デーモンはこの情報でモニター・エレメントを更新します。このイベントに対し、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** を使用して指定できます。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** を使用して指定できます。デフォルトでは、**fcm_num_buffers** および **fcm_num_channels** データベース・マネージャー構成パラメーターは AUTOMATIC に設定されます。 AUTOMATIC に設定されている場合 (これが推奨される設定です)、FCM はリソースの使用状況をモニターし、ワークロードの需要に対応できるようにリソースを調整します。

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 を使用可能にして、他のデータベース・パーティションと通信できるようにするには、このセクションの後の部分で示すように、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・パーティションを既に定義している場合、このセクションの後の部分で示すように、ある範囲のポートを定義する必要があります。

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

TCP/IP のポート範囲は、以下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。インスタンス名が DB2PUSER である場合には、DB2_db2puser と指定することになります。

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 ファイルが共有されている場合、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。そう

するには、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 番目の行は複数ポートから成るブロックの終わりを示します。以下の例では、SALES 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。以下に例を示します。

```
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 (_) を必ず含める必要があります。

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。

始める前に

root ユーザー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。

このタスクについて

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、Tivoli System Automation や IBM PowerHA® SystemMirror for AIX 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。

手順

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
```

各要素の意味は以下のとおりです。

- *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

データベース・パーティションを管理する

パーティションの開始または停止、パーティションのドロップ、およびパーティションのトレースを実行できます。

始める前に

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。SECADM または ACCESSCTRL 権限を持つユーザーから、特定のインスタンスへのアクセス権限を付与してもらうことができます。

手順

- 特定のデータベース・パーティションを開始または停止するには、**DBPARTITIONNUM** パラメーターを指定した **START DATABASE MANAGER** コマンドまたは **STOP DATABASE MANAGER** コマンドを使用します。
- 特定のデータベース・パーティションを db2nodes.cfg 構成ファイルからドロップするには、**DROP DBPARTITIONNUM** パラメーターを指定した **STOP DATABASE MANAGER** コマンドを使用します。 **DROP DBPARTITIONNUM** パラメーターを使用する前に、**DROP DBPARTITIONNUM VERIFY** コマンドを実行して、このデータベース・パーティションにユーザー・データが存在しないことを確認してください。
- データベース・パーティションでのアクティビティーをトレースするため、IBM サポートが指定するオプションを使用します。

  重要: トレース・ユーティリティーは、IBM サポートまたは技術サポート担当者によって指示された場合のみ使用してください。
  トレース・ユーティリティーは、DB2 for Linux, UNIX, and Windows の操作に関する情報を記録し、フォーマット設定します。詳しくは、『db2trc - トレース・コマンド』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停止中のどちらの時点でも、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できます。新しいサーバーの追加には時間がかかる可能性があるため、データベース・マネージャーが既に実行しているときにこれを行うのが適切かもしれません。

データベース・パーティションをシステムに追加するには、**ADD DBPARTITIONNUM** コマンドを使用します。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。

- **START DBM** コマンドのオプションとして
- **ADD DBPARTITIONNUM** コマンドと共に
- sqleaddn API と共に
- sqlepstart API と共に

システムが停止している場合は、**START DBM** コマンドを使用してください。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。

ADD DBPARTITIONNUM コマンドを使って新しいデータベース・パーティションをシステムに追加すると、インスタンス内の既存の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)。
- 別のデータベース・パーティション用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。

注: 新しいデータベース・パーティションの追加時には、カタログが作成されていないデータベースは認識されません。カタログが作成されていないデータベースは、新しいデータベース・パーティションには含まれません。新しいデータベース・パーティション上のデータベースに接続しようとすると、エラー・メッセージ SQL1013N が戻されます。

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。

データベース・パーティションをシステムに追加することで、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・データベースに変え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には、db2nodes.cfg ファイルを編集して新しいデータベース・パーティション定義を追加してください。ここで説明されている手順はデータベースが存在する場合にのみ該当しますから、どれも使用しないでください。

Windows での考慮事項: Windows オペレーティング・システムで Enterprise Server Edition を使用している場合、インスタンス内にデータベースが存在しなければ、**db2ncrt** コマンドを使ってデータベース・システムを拡大/縮小してください。ただしデータベースが既に存在する場合には、**START DBM ADD DBPARTITIONNUM** コマンドを使用することで、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 オペレーティング・システムでは、データベース・パーティション構成ファイル (db2nodes.cfg) を手動で編集しないでください。手動で編集した場合、ファイルの内容が矛盾するようになる可能性があります。

オンラインのデータベース・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、オンラインの新しいデータベース・パーティションをパーティション・データベース環境に追加することができます。

手順

コマンド行を使用してオンライン・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、**START DBM** 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。

   例えば、既存のデータベースに 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。

   ```
   START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME HOSTNAME3
    PORT PORT3;

   START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME HOSTNAME4
    PORT PORT4;

   START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME HOSTNAME5
    PORT PORT5;
   ```

2. オプション: 新しいデータベース・パーティションを取り込むためにデータベース・パーティション・グループを変更する。 新しいデータベース・パーティションにデータを再配分する際にも、オプションとしてこのアクションを実行できます。
3. 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。 新しいデータベース・パーティションを活用したい場合には、このアクションはオプションではありません。また、データベース・パーティション・グループの変更オプションを再配分操作に含めることもできます。そうしない場合は、新しいデータベース・パーティションにデータを再配分する前に、別個のアクションとして、新しいデータベース・パーティションを取り込むためのデータベース・パーティション・グループの変更を行う必要があります。
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。 これはオプションですが、特に新旧両方のデータベース・パーティションにわたってデータを再配分した場合には、新しいデータベース・パーティションおよび他のデータベース・パーティションに対してこの操作を行うと役立つでしょう。

データベース・パーティションの追加をオンラインで行う場合の制約事項

インスタンスに追加された新しいデータベース・パーティションの状況は、元のデータベース・パーティションの状況に応じて異なります。アプリケーションが WITH HOLD カーソルを使用する場合、インスタンスに追加された新しいデータベース・パーティションがアプリケーションに認識される場合と、認識されない場合があります。

単一パーティション・データベース・インスタンスに新しいデータベース・パーティションを追加するとき、

- データベース・パーティションの追加時に元のデータベース・パーティションが稼働中であれば、データベース・パーティションの追加操作の完了時に新しいデータベース・パーティションは非稼働になります。
- データベース・パーティションの追加時に元のデータベース・パーティションが非稼働であれば、データベース・パーティションの追加操作の完了時に新しいデータベース・パーティションは稼働中になります。

データベース・パーティション追加操作の実行前に開始された WITH HOLD カーソルを使用しているアプリケーションは、データベース・パーティションの追加操作の完了時に新しいデータベース・パーティションを認識しません。データベース・パーティション追加操作の実行前に WITH HOLD カーソルがクローズされた場合、アプリケーションはデータベース・パーティションの追加操作の完了時に新しいデータベース・パーティションを認識します。

オフラインのデータベース・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。

始める前に

- 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。
- **DB2_FORCE_OFFLINE_ADD_PARTITION** レジストリー変数のデフォルト値を TRUE に設定し、追加されたデータベース・パーティションが強制的にオフラインになるようにします。

手順

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. **STOP DBM** 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. **START DBM** 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。 新しいサーバーのインストール中に、新しいサーバーを含めるようにデータベース・パーティション構成ファイルがデータベース・マネージャーによって既に更新されていることに注意してください。
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、**START DBM** コマンドを実行します。

      DBPARTITIONNUM、**ADD DBPARTITIONNUM**、**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。

      例えば、既存のデータベースに 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。

      ```
      START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME HOSTNAME3
       PORT PORT3;

      START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME HOSTNAME4
       PORT PORT4;

      START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME HOSTNAME5
       PORT PORT5;
      ```

      この **START DBM** コマンドが完了すると、新しいサーバーは停止します。
   b. **STOP DBM** 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。 **STOP DBM** 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 (**START DBM** コマンドに **ADD DBPARTITIONNUM** 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. **START DBM** 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 **START DBM** 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: 新しいデータベース・パーティションを取り込むためにデータベース・パーティション・グループを変更する。 新しいデータベース・パーティションにデータを再配分する際にも、オプションとしてこのアクションを実行できます。
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。 新しいデータベース・パーティションを活用したい場合には、このアクションはオプションではありません。また、データベース・パーティション・グループの変更オプションを再配分操作に含めることもできます。そうしない場合は、新しいデータベース・パーティションにデータを再配分する前に、別個のアクションとして、新しいデータベース・パーティションを取り込むためのデータベース・パーティション・グループの変更を行う必要があります。
8. 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。 これはオプションですが、特に新旧両方のデータベース・パーティションにわたってデータを再配分した場合には、新しいデータベース・パーティションおよび他のデータベース・パーティションに対してこの操作を行うと役立つでしょう。

オフラインのデータベース・パーティションの追加 (Linux および UNIX)

パーティション・データベース・システムに、オフラインの新規データベース・パーティションを追加できます。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。

始める前に

- データベース・パーティションを作成する前に、新しいサーバーがまだ存在しなければ、インストールします。
- 共有ファイル・システム・マウントまたはローカル・コピーを使用して、実行可能モジュールをアクセス可能にします。
- 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・ファイルと同期化します。
- sqllib ディレクトリーがファイル共有システムとしてアクセス可能であることを確認します。
- 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します。
- すべてのデータベース・パーティションにおいて、ホスト名をネーム・サーバーに、あるいは /etc ディレクトリーの hosts ファイルに登録します。**rsh** または **rah** を使ってリモート・コマンドを実行するには、コンピューターのホスト名が .rhosts に登録済みでなければなりません。

- **DB2_FORCE_OFFLINE_ADD_PARTITION** レジストリー変数のデフォルト値を TRUE に設定し、追加されたデータベース・パーティションが強制的にオフラインになるようにします。

手順

- 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. **STOP DBM** 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. **START DBM** 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。 新しいサーバーのインストール中に、新しいサーバーを含めるようにデータベース・パーティション構成ファイル (db2nodes.cfg) がデータベース・マネージャーによって既に更新されていることに注意してください。
  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、**START DBM** コマンドを実行します。

        DBPARTITIONNUM、**ADD DBPARTITIONNUM**、**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。

        例えば、既存のデータベースに 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。

```
START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME HOSTNAME3
 PORT PORT3;

START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME HOSTNAME4
 PORT PORT4;

START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME HOSTNAME5
 PORT PORT5;
```

        この **START DBM** コマンドが完了すると、新しいサーバーは停止します。
     b. **STOP DBM** 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。 **STOP DBM** 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 (**START DBM** コマンドに **ADD DBPARTITIONNUM** 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. **START DBM** 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 **START DBM** 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: 新しいデータベース・パーティションを取り込むためにデータベース・パーティション・グループを変更する。 新しいデータベース・パーティションにデータを再配分する際にも、オプションとしてこのアクションを実行できます。
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。 新しいデータベース・パーティションを活用したい場合には、このアクションはオプションではありません。また、データベース・パーティション・グループの変更オプションを再配分操作に含めることもできます。そうしない場合は、新しいデータベース・パーティションにデータを再配分する前に、別個のアクションとして、新しいデータベース・パーティションを取り込むためのデータベース・パーティション・グループの変更を行う必要があります。
8. 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。 これはオプションですが、特に新旧両方のデータベース・パーティションにわたってデータを再配分した場合には、新しいデータベース・パーティションおよび他のデータベース・パーティションに対してこの操作を行うと役立つでしょう。

- 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。
     START DBM 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 *partitionnum* 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 DBPARTITIONNUM** 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、**START DBM** 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

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。その理由は、すべてのバッファー・プール・ページ・サイズに対するデフォルト自動サポートを提供する目的で、データベース・マネージャーがシステム・バッファー・プールを作成するためです。

ただし、これらのシステム・バッファー・プールのいずれかが使用される場合、これらのシステム・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可能性があります。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 表スペースを持つパーティション・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、**START DBM** コマンドを使用します。

     ```
     START DBM 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。

  これらの問題を防ぐ 1 つの方法は、**ADD DBPARTITIONNUM** または **START DBM** コマンドで WITHOUT TABLESPACES 節を指定することです。これを行った後、適切な SIZE および PAGESIZE 値を指定した CREATE BUFFERPOOL ステートメントを使ってバッファー・プールを作成し、ALTER TABLESPACE ステートメントを使って SYSTEM TEMPORARY 表スペースをバッファー・プールに関連付けます。

- 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。

    ```
    START DBM
    CONNECT TO mpp1
    ALTER DATABASE PARTITION GROUP ng1 ADD DBPARTITIONNUM (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。

    ```
    START DBM
    CONNECT TO mpp1
    CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
    CONNECT RESET
    CONNECT TO mpp1
    ALTER DATABASE PARTITION GROUP ng1 ADD DBPARTITIONNUM (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。

始める前に

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 DATABASE PARTITION GROUP** 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。

手順

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、以下のようにします。

DROP DBPARTITIONNUM パラメーターを指定した **STOP DBM** 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、**START DBM** 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)

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。

このタスクについて

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。

このタスクについて

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、**START DBM ADD DBPARTITIONNUM** 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:partition_number
        /u:username,password
        /p:logical_port
```

/n:partition_number
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:username,password
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
```

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。

このタスクについて

- 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、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。インスタンスに既存のデータベースがない場合にのみ、データベース・パーティション・サーバーを移動させることができます。

/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。

手順

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。 *database_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。

例

以下の例では、UNIX 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。

このタスクについて

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、**STOP DBM DROP DBPARTITIONNUM** 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:dbpartitionnum /i:instance_name
```

/n:*dbpartitionnum*
: 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 (*dbpartitionnum*)。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 (*instance_name*)。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。

シナリオ: 新規データベース・パーティションでのデータの再配分

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。

このタスクについて

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。

バージョン 9.7 以降、データベース・パーティションの追加は、データベースの実行中にアプリケーションが接続していても行えるようになりました。ただし、このシナリオでは、**DB2_FORCE_OFFLINE_ADD_PARTITION** レジストリー変数のデフォルト値を TRUE に変更することにより、操作をオフラインで行えます。

手順

DBPG1 内のデータベース・パーティション間のデータを再配分するには、次のようにします。

1. 再配分の前に無効または除去する必要のあるオブジェクトを識別します。
   a. 複製 MQT: このタイプの MQT は、再配分操作の一部としては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、後で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: 表書き込みイベント・モニターでは、データベース・パーティション・グループ内の表が再配分されるため、自動的にアクティブにされる表書き込みイベント・モニターを使用不可にします。

      ```
      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 = 'DBPG_1'
      ```

   c. Explain 表: Explain 表は単一パーティション・データベース・パーティション・グループ内に作成することをお勧めします。ただし、再配分を必要とするデータベース・パーティション・グループでそれらが定義されていて、その時点までに生成されたデータを維持する必要がなければ、それらをドロップすることを考慮してください。再配分が完了してから、Explain 表を再定義できます。
   d. 表アクセス・モードおよび状態: 再配分するデータベース・パーティション・グループ内のすべての表がフルアクセス・モードであり、通常の表の状態であることを確認します。

```
SELECT DISTINCT TRIM(T.OWNER) || ¥'.¥' || TRIM(T.TABNAME)
  AS NAME, T.ACCESS_MODE, A.LOAD_STATUS
 FROM SYSCAT.TABLES T, SYSCAT.DBPARTITIONGROUPS
  N, SYSIBMADM.ADMINTABINFO A
 WHERE T.PMAP_ID = N.PMAP_ID
  AND A.TABSCHEMA = T.OWNER
  AND A.TABNAME = T.TABNAME
  AND N.DBPGNAME = 'DBPG_1'
  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。再配分ユーティリティーで表の統計を更新すると、すべてのデータが再配分のためにスキャンされ、**RUNSTATS** のためのデータの追加スキャンが不要になるので、入出力が少なくなります。

```
RUNSTATS on table schema.table
  USE PROFILE runstats_profile
  SET PROFILE ONLY
```

2. 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。再配分の間は、できるだけ多くのメモリーを **util_heap_sz** に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は、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。

3. 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ときは、デフォルトのデータベース構成が使用されます。結果として、**REDISTRIBUTE DATABASE PARTITION GROUP** コマンドを発行する前に新規データベース・パーティションのデータベース構成を更新することが重要です。この一連のイベントにより、構成のバランスがとれていることが保証されます。

```
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')
```

4. 再配分プロセスを開始する前に、データベース (または当該データベース・パーティション・グループ内の表スペース) をバックアップします。 このアクションにより、新しいリカバリー・ポイントが確保されます。

5. 3 つの新規データベース・パーティションをデータベースに追加します。 以下のコマンドを発行します。

```
START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME HOSTNAME3
 PORT PORT3 WITHOUT TABLESPACES;
```

```
START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME HOSTNAME4
 PORT PORT4 WITHOUT TABLESPACES;

START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME HOSTNAME5
 PORT PORT5 WITHOUT TABLESPACES;
```

DB2_FORCE_OFFLINE_ADD_PARTITION を TRUE に設定した場合、インスタンスは、シャットダウンされて再始動されるまで、新しいデータベース・パーティションを認識しません。以下に例を示します。

```
STOP DBM;
START DBM;
```

6. 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 (3 to 5)
   ```

7. 新規のデータベース・パーティションをデータベース・パーティション・グループに追加します。 次のコマンドは、DBPG_1 定義を (0, 1) から (0, 1, 3, 4, 5) に変更します。

   ```
   ALTER DATABASE PARTITION GROUP DBPG_1
     ADD dbpartitionnums (3 to 5)
     WITHOUT TABLESPACES
   ```

8. 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 (3 to 5)
   ```

9. データベース構成設定を、新しいデータベース・パーティションに適用します (または、全データベース・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)。
10. 再配分されるデータベース・パーティション・グループ内に存在する複製 MQT すべての定義を収集し、ドロップします。

    ```
    db2look -d DBPG1 -e -z
      schema -t replicated_MQT_table_names
      -o repMQTs.clp
    ```

11. 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・モニターをすべて使用不可にします。

    ```
    SET EVENT MONITOR monitor_name STATE 0
    ```

12. 再配分ユーティリティーを実行して、すべてのデータベース・パーティションにわたって均一に再配分します。

    ```
    REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
     UNIFORM 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。

    ```
    REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1
    NOT ROLLFORWARD RECOVERABLE ABORT;
    ```

エラーまたは中断が発生したために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ことができます。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。

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 UNIFORM 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。

```
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:

- ステップ 7 (193 ページ) および 8 (193 ページ) で ALTER DATABASE PARTITION GROUP および ALTER TABLESPACE ステートメントを実行する代替手段として、**REDISTRIBUTE DATABASE PARTITION GROUP** コマンドで **ADD DBPARTITIONNUM** パラメーターを指定できます。このコマンド・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。
- この例の **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。
- **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操作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。

詳細については、『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してください。再配分ユーティリティーはデータ (圧縮およびコンパクト) を移動します。統計プロファイルが定義されていれば、必要に応じて索引が再作成され、統計が更新されます。したがって、前述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 DBPG_1
  NOT ROLLFORWARD RECOVERABLE uniform
  TABLE (t1, t2,...) FIRST;
```

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。

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。

Linux および UNIX オペレーティング・システムでは、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。

また、Linux および UNIX オペレーティング・システムでは、**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。 **DB2RSHCMD** が設定されない場合、rsh (HP-UX では remsh) が使用されます。 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。

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データベース・パーティション・サーバーで **db2trc** を実行する場合は、**rah** を使用してください。

注: **db2_all** コマンドは、対話式ユーザー入力が必要なコマンドをサポートしていません。

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 オペレーティング・システ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 オペレーティング・システムでは、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、データベース・パーティション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 UPDATE DB CFG FOR sample USING LOGFILSIZ 100"
```

注: db2_all コマンドは、対話式ユーザー入力が必要なコマンドをサポートしていません。

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。

次のような特殊文字がコマンドに含まれる場合は、プロンプト方式を使用してください。

```
| & ; < > ( ) { } [ ] 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 オペレーティング・システムでは、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 ¥ を含める必要がある場合は、円記号を 2 つ (¥¥) 入力してください。

注: Linux および UNIX オペレーティング・システムでは、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)。これらの文字のいずれかをコマンドに含める必要がある場合は、その前に円記号を 3 つ (¥¥¥) 付ける必要があります。例えば ¥ をコマンドに含める必要がある場合、円記号を 4 つ (¥¥¥¥) 入力する必要があります。

二重引用符 (") をコマンドに含める必要がある場合は、その前に 3 つの円記号を付けて ¥¥¥" のようにする必要があります。

注:

1. Linux および UNIX オペレーティング・システムでは、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。

バックグラウンドで stdin から読み取るロジックを含む Korn シェルのシェル・スクリプトを実行する場合には、端末上で処理が停止することなく読み取れるようなソースに stdin を明示的にリダイレクトしてください (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、バックグラウンドで **db2_all** を実行する際には常に </dev/null を指定してください。以下に例を示します。

```
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。

これに代わる方法として、リモート・コマンドからの出力を考慮する必要がない場合には、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 オペレーティング・システ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** または $**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データベース・パーティションの数を検査し、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データベース・パーティションに送信します。

それぞれのデータベース・パーティションでは、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データベース・パーティション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、**RAHTREETHRESH** 環境変数で指定できます。これを指定しないと、デフォルトの 15 になります。

物理データベース・パーティションに対して複数の論理データベース・パーティションが存在するシステムの場合、**db2_all** は個別の物理データベース・パーティションに再帰的な呼び出しを送った後、その同じ物理データベース・パーティション上の他の論理データベース・パーティションに rsh するため、物理データベース・パーティション間のトラフィックもまた削減されます。 (この点は **db2_all** にのみ該当します。rah は常に個別の物理データベース・パーティションにのみ送信するため、これは rah には当てはまりません。)

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・サーバーで非対話式コマンドを実行します。**db2_all** は、対話式ユーザー入力が必要なコマンドをサポートしていません。

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。 IBM ソフトウェア・サポートから指示された場合、または維持されたトラップから回復するよう指示された場合を除いて、このコマンドを発行しないでください。

db2_call_stack

Linux および UNIX オペレーティング・システムでは、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 オペレーティング・システムでは、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 および db2_all 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 オペレーティング・システムでは、以下のようにします。

```
rah "};ps -F pid,ppid,etime,args -u $USER"
db2_all "};ps -F pid,ppid,etime,args -u $USER"
```

- Windows オペレーティング・システムでは、以下のようにします。

```
rah "||db2 get db cfg for sample"
db2_all "||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 オペレー

ティング・システムの場合) またはバックグラウンド・プロセス (Windows オペレーティング・システムの場合) 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 オペレーティング・システムで & 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、以前に |& 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 オペレーティング・システムで & 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。

; ||& と同じ。これは、省略形です。

注: Linux および UNIX オペレーティング・システムで ; 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 オペレーティング・システ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 オペレーティング・システ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 オペレーティング・システ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 オペレーティング・システ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 (データベース・パーティション番号 *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 は、db2nodes.cfg ファイルの *nodenum* 値に対応する 1、2、または 3 桁のデータベース・パーティション番号です。

<<-nnn< を適用できるのは、**db2_all** だけです。

<<+*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。

nnn は、db2nodes.cfg ファイルの *nodenum* 値に対応する 1、2、または 3 桁のデータベース・パーティション番号です。

<<+nnn< を適用できるのは、**db2_all** だけです。

(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。

- 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、データベース・パーティション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、データベース・パーティション構成ファイル内の 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 オペレーティング・システムの場合、データベース・パーティション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。

- " 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。

使用上の注意

- 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
$**RAHBUFDIR** **注:** Linux および UNIX オペレーティング・システムに限り使用できます。	バッファーのディレクトリー	/tmp/$USER
$**RAHBUFNAME** **注:** Linux および UNIX オペレーティング・システ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 オペレーティング・システムの場合)、**RAHOSTFILE** (Windows オペレーティング・システムの場合)	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 オペレーティング・システムの場合)、**RAHOSTLIST** (Windows オペレーティング・システムの場合)	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 オペレーティング・システ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 オペレーティング・システムの場合)、**RAHSLEEPTIME** (Windows オペレーティング・システムの場合)	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 オペレーティング・システムの場合)、**RAHWAITTIME** (Windows オペレーティング・システムの場合)	Windows オペレーティング・システムの場合、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するインターバルの秒数。 Linux および UNIX オペレーティング・システムの場合、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。 オペレーティング・システムに関係なく、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 オペレーティング・システ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 オペレーティング・システムの場合)、**RAHUSER** (Windows オペレーティング・システムの場合)	Linux および UNIX オペレーティング・システムの場合、リモート・コマンドがその下で実行されるユーザー ID。 Windows オペレーティング・システムの場合、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 オペレーティング・システムでは、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 オペレーティング・システ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 に書き出すことはできません。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 オペレーティング・システ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 /etc/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** または $**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。

このタスクについて

注: この節の情報は Linux および UNIX オペレーティング・システ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。これを指定する方法について詳しくは、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 または UNIX オペレーティング・システムでは、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を判別します。) 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。

このタスクについて

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。

始める前に

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。

このタスクについて

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、8 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。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 AMS)、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。

始める前に

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)
- DBADM 権限

このタスクについて

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。

手順

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。

このタスクについて

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。

データ・パーティションのデータ、索引、およびロング・データの配置

本質的に、パーティション表の作成によって、表のさまざまな部分、および関連する表オブジェクトを特定の表スペースに配置することができます。

表を作成するとき、どの表スペースに表データ全体および関連する表オブジェクトを配置するかを指定することができます。あるいは、表の索引、ロング・データまたはラージ・データ、または表パーティションを特定の表スペースに配置することができます。すべての表スペースは、同一のデータベース・パーティション・グループになければなりません。

CREATE TABLE ステートメントには、表データおよび関連する表オブジェクトを特定の表スペース内に配置するこの機能を示す、以下の節があります。

```
CREATE TABLE table_name IN table_space_name1
   INDEX IN table_space_name2
   LONG IN table_space_name3
   PARTITIONED BY ...
     PARTITION partition_name | boundary specification | IN table_space_name4
       INDEX IN table_space_name5
       LONG IN table_space_name6
```

パーティション表の各パーティションは、異なる表スペースに配置できます。

また、CREATE INDEX ... IN *table_space_name1* ステートメントを使用することにより、ユーザー作成の非パーティション化索引用の表スペースをパーティション表に指定することもできます。これは、CREATE TABLE ... INDEX IN *table_space_name2* ステートメントで指定される索引表スペースとは異なるものにすることが可能です。 CREATE INDEX ステートメントの IN 節はパーティション表でのみ使用されます。CREATE TABLE ステートメントまたは CREATE INDEX ステートメントで INDEX IN 節を指定しない場合は、表の最初の可視パーティションまたは最初に追加されたパーティションと同じ表スペースに索引が配置されます。

システムにより生成される非パーティション化索引 (例えば XML 列パス索引) は、CREATE TABLE ステートメントの INDEX IN 節で指定された表スペースに配置されます。

XML データを持つパーティション表において、XML 領域索引は表データと同じ方法で常にパーティション化されます。パーティション化索引の表スペースは、パーティション・レベルで定義されます。

XML データは、表のロング・データによって使われる表スペースに格納されます。パーティション表での XML データの配置は、ロング・データの配置規則に従います。

ロング・データ用の表スペースを明示的に指定できます。または、データベース・マネージャーによって暗黙的に決定することもできます。パーティション表の場合、表レベルの LONG IN 節をパーティション・レベルの LONG IN 節と共に使用

できます。この両方が指定される場合、パーティション・レベルの LONG IN 節が表レベルのすべての LONG IN 節よりも優先されます。

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

空のパーティション表に、非パーティション表または UNION ALL ビューをマイグレーションすることができます。

始める前に

特定の列の SYSCAT.COLUMNS.IMPLICITVALUE がソース列とターゲット列の両方で非ヌル値であり、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。

- アタッチ操作時にソース表から IMPLICITVALUE フィールドが伝搬される
- デタッチ操作時にソース表から 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)。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。

ATTACH 操作に関係するソース表およびターゲット表は、必ず同じ列を定義して作成してください。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おけるミスマッチの回避策については、246 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。

このタスクについて

通常表からのマイグレーション時、**EXPORT** コマンドまたは High Performance Unload を使用してソース表をアンロードします。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そのパーティション表にデータを取り込みます。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には、**LOAD FROM CURSOR** コマンドを使用します (ステップ 1 を参照)。

UNION ALL ビューにある非パーティション化データをパーティション表に変換することができます (ステップ 2 を参照)。 UNION ALL ビューは、ブランチを除去することによる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。ALTER TABLE...ATTACH PARTITION ステートメントを使用すると、基本表のデータを移動せずに変換を行えます。変換の後、非パーティション化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。UNION ALL ビューをパーティション表に変換する際は、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が推奨されています。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。

手順

1. パーティション表に通常表をマイグレーションします。 中間ステップを回避するため、**LOAD FROM CURSOR** コマンドを使用します。以下の例は、表 T1 を SALES_DP 表にマイグレーションする方法を示しています。
   a. 通常表 T1 を作成し、データを取り込みます。

      ```
      CREATE TABLE t1 (c1 int, c2 int);
      INSERT INTO t1 VALUES (0,1), (4, 2), (6, 3);
      ```

   b. 空のパーティション表を作成します。

      ```
      CREATE TABLE sales_dp (c1 int, c2 int)
        PARTITION BY RANGE (c1)
        (STARTING FROM 0 ENDING AT 10 EVERY 2);
      ```

   c. **LOAD FROM CURSOR** コマンドを使用して、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;
      ```

2. UNION ALL ビューにある非パーティション化データを、パーティション表に変換します。 以下の例は、ALL_SALES という名前の UNION ALL ビューを、SALES_DP 表に変換する方法を示しています。
   a. UNION ALL ビューを作成します。

      ```
      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'
       );
      ```

   b. 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してください。

      ```
      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');
      ```

   c. 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;
      ```

   d. 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;
```

e. 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;
```

f. SET INTEGRITY ステートメントを使用して、新規にアタッチしたパーティションのデータを照会からアクセス可能にします。

```
SET INTEGRITY FOR sales_dp IMMEDIATE CHECKED
FOR EXCEPTION IN sales_dp USE sales_ex;
```

ヒント: データ・サーバーから独立したアプリケーション・ロジックを使用して、アタッチ操作の前に、範囲の妥当性その他の制約検査を含むデータ保全性検査を実行できる場合、新規にアタッチされたデータはずっと迅速に使用可能になります。SET INTEGRITY...ALL IMMEDIATE UNCHECKED ステートメントを使用して、範囲および制約の違反に対する検査をスキップすることにより、データのロールイン処理を最適化できます。この場合、表は SET INTEGRITY 保留状態を脱します。そして、ターゲット表上に非パーティション・ユーザー索引が存在しない限り、即時に新規データがアプリケーションから使用可能になります。

g. 必要に応じて索引を作成します。

既存の索引をパーティション索引に変換する

システム作成による索引やユーザー作成の索引を、非パーティション化からパーティション化にマイグレーションする必要が生じることがあります。ほとんどのマイグレーションにおいて、表および索引を使用可能にしたまま、ユーザー作成の索引を変換することができます。主キー制約やユニーク制約を実施するために使われるシステム作成索引の場合、変換中に制約を維持することはできません。

始める前に

以前のリリースの製品で作成された索引はパーティション化されていない場合があります。これには、ユーザーが独自に作成した索引と、データベース・マネージャーによって作成されたシステム作成索引が含まれる可能性があります。システム作成索引の例として、ユニーク制約や主キー制約を実施するための索引、および MDC 表のブロック索引があります。

このタスクについて

索引を使用してデータを利用できるようにしたまま、ユーザー作成の索引を非パーティション化からパーティション化に変換できます。対応する非パーティション索引と同じキーを使用して、パーティション索引を作成できます。パーティション索引の作成時には、現在の索引と、索引が生成される表を引き続き使用できます。パーティション索引を作成した後、必要であれば、対応する非パーティション索引をドロップして新しいパーティション索引の名前を変更することができます。

タスクの結果

以下の例は、既存の非パーティション索引をパーティション索引に変換する方法を示しています。

例

ユーザー作成による非パーティション索引をパーティション索引に変換する例を示します。

```
UPDATE COMMAND OPTIONS USING C OFF;
CREATE INDEX data_part ON sales(sale_date) PARTITIONED;
DROP INDEX dateidx;
RENAME INDEX data_part TO dateidx;
COMMIT;
```

データベース・マネージャーによって作成された非パーティション索引をパーティション索引に変換する例を示します。この場合、元の制約をドロップしてから新しい制約を作成するまでに時間がかかります。

```
ALTER TABLE employees DROP CONSTRAINT emp_uniq;
ALTER TABLE employees ADD CONSTRAINT emp_uniq UNIQUE (employee_id);
```

DB2 バージョン 9.7 以前のリリースを使用して作成された MDC 表は、非パーティション化ブロック索引を持ちます。データのロールイン、ロールアウト、表のデータおよび索引のパーティション・レベルでの再編成など、パーティション表のデータ可用性フィーチャーを利用するためには、 DB2 V9.7 以前のリリースを使用して作成されたマルチディメンション・クラスタリング (MDC) 表のデータを、 DB2 V9.7 フィックスパック 1 以降のリリースを使用して作成された、パーティション化ブロック索引を持つパーティション化 MDC 表に移動する必要があります。

パーティション化ブロック索引を使用するためのパーティション化 MDC 表のオンライン移動

オンライン表移動を使用して、非パーティション化ブロック索引を持つ MDC 表から、パーティション化ブロック索引を持つ MDC 表に、データを移動できます。

以下の例では、company1.parts 表が MDC キー列として **region** および **color** を持ち、対応するブロック索引はパーティション化されていません。

```
CALL SYSPROC.ADMIN_MOVE_TABLE(
 'COMPANY1',  -- 表のスキーマ
 'PARTS',     -- 表名
 ' ',         -- null (列定義に変更なし)
 ' ',         -- null (追加のオプションなし)
 'MOVE');     -- 表を 1 ステップで移動させる
```

パーティション化ブロック索引を使用するためのパーティション化 MDC 表のオフライン移動

データ移動を最小限に抑えるために、表がオフラインの時に、非パーティション化ブロック索引を持つ MDC 表から、パーティション化ブロック索引を持つ MDC 表に、データを移動できます。この処理には、以下の手順を使用します。

1. 変換する表と同じ定義の単一パーティション MDC 表を新規作成します。パーティションの範囲指定には、変換するパーティション化 MDC 表の範囲の外側を使用します。

新規の単一パーティション MDC 表のブロック索引は、パーティション化されます。範囲を指定して作成したパーティションは、後の手順でデタッチします。

2. MDC 表の各パーティションをデタッチします。各パーティションはスタンドアロン MDC 表になります。

   パーティションをデタッチすると、パーティション・データは、パーティション内のデータを移動させることなく、新規のターゲット表にアタッチされます。

   注: MDC 表の最後のパーティションをデタッチすることはできません。これは、非パーティション化ブロック索引を持つ単一パーティション MDC 表であるためです。

3. MDC 表のパーティションのデタッチによって作成された各スタンドアロン表、および非パーティション化ブロック索引を持つ単一パーティション MDC 表を、手順 1 で作成した新規パーティション MDC 表にアタッチします。

   表をアタッチすると、表データは、データを移動させることなく新規パーティション化 MDC 表にアタッチされ、ブロック索引はパーティション化ブロック索引として作成されます。

4. スタンドアロン MDC 表を最初にアタッチしたら、新規 MDC 表の作成時に生成した空のパーティションはデタッチ可能です。

5. SET INTEGRITY ステートメントを新規パーティション化 MDC 表に対して実行します。

次のタスク

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表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。

- ALTER TABLE ... DETACH PARTITION ステートメントを実行すると、DETACH 操作によって、デタッチされるパーティション・データのターゲット表が作成されます。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)、デタッチされた従属表に対して SET INTEGRITY ステートメントを実行して、表を増分的に保守する必要があります。 DB2 V9.7 フィックスパック 1 以降のリリースでは、デタッチされたすべての従属表に対して SET INTEGRITY ステートメントを実行した後、非同期のパーティション・デタッチ操作によって、このデータ・パーティションはスタンドアロンのターゲット表に変換されます。非同期のパーティション・デタッチ操作が完了するまで、ターゲット表は使用できません。ターゲット表は、SYSCAT.TABLES カタログ・ビューの TYPE 列に 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、ターゲット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、データ・パーティションはソース表から論理的にデタッチされ、非同期のパーティション・デタッチ操作によって、ソース表からデタッチされてターゲット表に変換されます。非同期のパーティション・デタッチ操作が完了するまで、ターゲット表は使用できません。

- 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表に対して 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。
- SYSCAT.DATAPARTITIONS の STATUS が「D」であるデタッチされたデータ・パーティションを持つ表に非パーティション索引を作成すると、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妥当性検査されるまでは、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、ユーザーによって妥当性検査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

範囲がクラスター化された表の使用のガイドライン

このトピックでは、範囲がクラスター化された表 (RCT) を扱うときに従う必要がある、いくつかのガイドラインをリストします。

- 範囲がクラスター化された表の作成処理では、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、そのスペースが使用可能でなければなりません。
- 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。負の最小値は、明示的に指定しなければなりません。以下に例を示します。

  ```
  ORGANIZE BY KEY SEQUENCE (f1 STARTING FROM -100 ENDING AT -10)
  ```

- 範囲がクラスター化された表の定義に使用されたのと同じキー値に通常の索引を作成することはできません。
- 表の物理構造に影響を与える ALTER TABLE ステートメントのオプションは使用できません。

シナリオ: 範囲がクラスター化された表

範囲がクラスター化された表は 1 列または複数列のキーを持ち、定義されている値の範囲を越えるキー値を持つ行を許可または禁止することができます。このセクションには、そのような表を作成する方法を示す、いくつかのシナリオが含まれています。

シナリオ 1: 範囲がクラスター化された表の作成 (オーバーフローが許可される)

次の例は、特定の学生に関する情報を取得するために使用できる、範囲がクラスター化された表を示します。各学生のレコードには、以下の情報が含まれています。

- 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)

```
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
;
```

この例では、表キーとして機能する STUDENT_ID 列を使用して、学生のレコードを追加、更新、および削除します。

各レコードのサイズは、列の長さの合計に基づいて決定されます。この例では、各レコードの長さは 97 バイトです (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))。ページ・サイズが 4 KB (すなわち 4096 バイト) の場合、オーバーヘッドを考慮すると、1 ページあたり 4038 バイト (41 レコード分) が使用可能です。100 万人の学生のレコードを保管するには、このようなページが合計 24391 ページ必要です。この表が作成されるときには、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 ページを想定して、4 KB のページが 24384 ページ、事前割り振りされます。(エクステント・マッピングは、この表が 3 ページの単一コンテナーに保持されることを想定しています。)

シナリオ 2: 範囲がクラスター化された表の作成 (オーバーフローが許可されない)

次の例では、教育委員会が 200 の学校を管理し、各学校には、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。

```
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、SCHOOL_ID、CLASS_ID、および STUDENT_NUM の各列を組み合わせた表キーを使用して、学生のレコードを追加、更新、および削除します。

教育委員会の方針により、クラスルームあたりの学生数は制限されています。また、この教育委員会が管理する学校数およびクラスルーム数は固定されています。そのため、オーバーフローは許可されません。一部の、比較的小規模な学校 (大きな学校よりもクラスルームの数が少ない) では、表の事前割り振りされたスペースの中に、決して使用されないスペースが存在します。

MDC または ITC 表を作成する際の考慮事項

MDC または ITC 表を作成する際には、さまざまな要因を考慮する必要があります。現在のデータベース環境 (例えばパーティション・データベースが存在するかどうか) に応じて、またディメンションの選択に応じて、MDC または ITC 表を作成、配置、および使用する方法の決定が異なることがあります。

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、データ保守操作の要件を少なくするため、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 表にデータを移動するには、以下のようにします。

1. データをエクスポートする。
2. 元の表をドロップする (オプション)。
3. (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) 表を作成する。
4. MDC 表にデータをロードする。

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手順を実行します。

1. 表の従属オブジェクトをすべてドロップします。
2. 表の名前を変更します。
3. 新しい定義で表を作成します。
4. 表の従属オブジェクトをすべて再作成します。
5. 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、古い表のデータを選択して新しい表にそのデータをロードする際に、列関数を使って古いデータ・タイプから新しいデータ・タイプに変換することができます。

既存の表から ITC 表へのデータの移動

データ保守操作の必要性を減らすため、通常の表のデータを、挿入時クラスタリング (ITC) 表に移動することができます。既存の表から ITC 表にデータを移動するには、オンライン表移動のストアード・プロシージャーを使用します。

ExampleBank のシナリオは、既存の表から ITC 表にデータを移動する方法を示します。また、ITC 表を使用する際にスペースを再利用することの利便性についてもシナリオで示されています。詳しくは、関連概念のリンクを参照してください。

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**) 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 (異なる値の数) を少なくすることに関する数学用語です。一般的な例としては、日付、週、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ります。エクステント・サイズは 1 個のブロックまたはセルに入れることのできるレコード数に影響を与えるためです。エクステント・サイズはセルの密度に直接影響します。

表に関して単一ディメンションまたは複数ディメンションのどちらも推奨される可能性がありますが、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・タイプ例外には CHAR、VARCHAR、GRAPHIC、および VARGRAPHIC 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利点だけでなく、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 フィーチャーを実行するには、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 DIMENSIONS 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

パーティション・データベース環境でマルチディメンション・クラスタリングを使用できます。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データベース・パーティション間で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・パーティション上に、Yellow スライスがもう 1 つのパーティション上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てセル占有度を見積もる際には、平均的に、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、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。 ORGANIZE BY DIMENSIONS 節を使用してこのことを行えます。

例 1:

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

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

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・ブロック索引があります。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 つがあります。この 2 つの方法の主な違いは、例 1 の場合、 c4 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 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。この方法で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 が列 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` である生成列 month に関するディメンションを持つ MDC 表を作成するとします。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含めます。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

複数のデータベース・パーティションにおけるデータベース構成の変更

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。

このタスクについて

SQL コンパイラーは、データベース・パーティション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

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・パーティションの追加、新規データ・パーティションのロールイン (アタッチ)、および既存のデータ・パーティションのロールアウト (デタッチ) が可能になります。

始める前に

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 PARTITION 操作を実行するユーザーは、ソース表に対する ALTER、SELECT、および DELETE 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。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。
  - 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 DATAACCESS 権限または SELECT 特権、およびソース表のスキーマに対する DBADM 権限または DROPIN 特権
- ソース表に対する CONTROL 特権

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
- DBADM

- 表スキーマの ALTERIN 特権

このタスクについて

- 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。

手順

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。

SYSCAT.DATAPARTITIONS カタログ・ビューの STATUS 列には、表のパーティションの状況に関する情報が含まれています。

- STATUS が空ストリングの場合、パーティションは通常の状況であり表示可能です。
- STATUS が「A」の場合、パーティションは新規にアタッチされており、アタッチされたパーティションを通常の状況にするには、SET INTEGRITY ステートメントを実行する必要があります。
- STATUS が「D」、「L」、または「I」の場合、パーティションはデタッチ中であり、デタッチ操作はまだ完了していません。
  - 状態「D」のパーティションの場合、パーティションを論理的にデタッチされた状態にするには、すべてのデタッチされた従属表に対して SET INTEGRITY ステートメントを発行する必要があります。
  - 状態「L」のパーティションの場合、パーティションは論理的にデタッチされたパーティションであり、非同期のパーティション・デタッチ・タスクによって、パーティションのデタッチが完了しつつあります (DB2 バージョン 9.7 フィックスパック 1 以降のリリース)。
  - 状態「I」のパーティションの場合、非同期のパーティション・デタッチ・タスクは完了し、非同期の索引クリーンアップによって、パーティションに定義された非パーティション索引を更新中です。

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。状態「D」または「L」のデタッチされたパーティションがパーティション表に含まれるときに主キー制約またはユニーク制約を追加しようとすると、制約の実施のためにシステムが新しいパーティション索引を生成

する必要がある場合にはエラーが戻されます。パーティションが状態「L」の場合、この操作は SQL20285N (SQLSTATE 55057) を戻します。パーティションが状態「D」の場合、この操作は SQL20054 (SQLSTATE 55019) を戻します。

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。状態「I」の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。

状態「L」または「D」のデタッチされたパーティションの場合、この操作は失敗しエラーが戻されます。パーティションが状態「L」の場合、この操作は SQL20285N (SQLSTATE 55057) を戻します。パーティションが状態「D」の場合、この操作は SQL20296N (SQLSTATE 55057) を戻します。

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。

パーティションがデタッチされて状態「L」または「D」であるときに、列をドロップまたはリネームしようとすると、この操作は失敗しエラーが戻されます。パーティションが状態「L」の場合、この操作は SQL20285N (SQLSTATE 55057) を戻します。パーティションが状態「D」の場合、この操作は SQL0270N (SQLSTATE 42997) を戻します。

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。

非パーティション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・パーティションが状態「D」であり、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表またはステージング表を持っている場合を除きます。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。

パーティション索引の追加または変更

アタッチされたデータ・パーティションが存在する場合にパーティション索引を作成する際には、アタッチされたデータ・パーティションごとに 1 つ

の索引パーティションが作成されます。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ために SET INTEGRITY ステートメントを実行するまでは、アタッチされたデータ・パーティション上の索引パーティションに関する索引項目は表示されません。索引の作成ではアタッチされたデータ・パーティションが含まれるため、パーティション化されたユニーク索引の作成時に、アタッチされたデータ・パーティション内に重複キー値を持つ行が見つかって索引の作成が失敗する可能性があります。アタッチされたパーティションが存在する場合には、この問題を防ぐために、ユーザーがパーティション索引の作成を試みないよう推奨します。

デタッチされた従属表が表に存在する場合には、デタッチされた従属表を持つパーティション表でパーティション索引を作成することはできません。このような状況でパーティション索引を作成しようとすると、SQLSTATE 55019 が結果として発生します。状態「L」のパーティションを持つ表にパーティション索引を作成しようとすると、この操作は SQL20285N (SQLSTATE 55057) を戻します。

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

同じトランザクションでのデータ・パーティションの作成とアクセス
表に非パーティション索引がある場合、その表に新しいデータ・パーティションを作成した追加操作またはアタッチ操作と同じトランザクションにおいてその表が排他モードでロックされていないと、そのトランザクションでその新しいパーティションにアクセスすることはできません (SQL0668N、理由コード 11)。

パーティションの ADD、ATTACH、または DETACH 操作によって表を変更する場合の XML 索引に関する特別な考慮事項

非パーティション化されたリレーショナル索引の場合と同様に、XML 列に対する非パーティション化索引は、1 つのパーティション表のすべてのデータ・パーティション間で共有される独立したオブジェクトです。パーティションの追加、アタッチ、またはデタッチ操作によって表を変更するとき、XML 領域索引と列パス索引が影響を受けます。 XML 列パスの索引は、常にパーティション化されません。一方、XML データの索引はデフォルトでパーティション索引として生成されます。

XML 領域索引

ADD PARTITION 操作では、追加される新しい空のデータ・パーティション用の新しい領域索引パーティションが作成されます。領域索引パーティションに関する新しい項目が SYSINDEXPARTITIONS 表に追加されます。新しいパーティション上のパーティション化索引オブジェクト用の表スペースは、ADD PARTITION 節の INDEX IN <table space> によって決定されます。 ADD PARTITION 節で INDEX IN <table space> が指定されない場合、パーティション化索引オブジェクト用の表スペースは、デフォルトで対応するデータ・パーティションによって使われる表スペースと同じになります。

パーティション表のシステム生成 XML 領域索引は常にパーティション化されます。パーティション化索引で使われる索引編成スキームは、表のパーティション・スキームに従って、(索引パーティションという) 複数のストレージ・オブジェクトに索引データを分割します。それぞれの索引パーティションは、対応するデータ・パーティション内の表の行だけを参照します。

ATTACH の場合、XML 列を持つパーティション表の領域索引は常にパーティション化されるため、ソース表の領域索引を、新しい表パーティション用の新しい領域索引パーティションとして ATTACH 操作の完了後に保持できます。データ・オブジェクトおよび索引オブジェクトは移動されないため、カタログ表の項目を更新する必要があります。ソース表の領域索引に関するカタログ表の項目は ATTACH によって除去され、1 つの領域索引パーティションが SYSINDEXPARTITIONS 表に追加されます。プール ID とオブジェクト ID は、ソース表と同じものになります。索引 ID (IID) は、ターゲット上の領域索引に対応して変更されます。

DETACH 操作の完了後に、領域索引はデタッチされた表に保持されます。デタッチされるパーティションに関連した索引パーティション項目は、SYSINDEXPARTITIONS 表から除去されます。デタッチされる表に関して SYSINDEXES カタログ表に 1 つの新しい領域索引項目が追加されます。そのプール ID とオブジェクト ID は DETACH 前の領域索引パーティションと同じです。

XML データに対する索引

DB2 バージョン 9.7 フィックスパック 1 から、パーティション表の XML データの索引は、パーティション索引にも非パーティション索引にもできます。デフォルトは、パーティション索引です。

ATTACH および DETACH 操作中に、XML データに対するパーティション索引および非パーティション索引は、他のリレーショナル索引と同じように扱われます。

ソース表の索引は ATTACH 操作中にドロップされます。論理的および物理的 XML 索引のどちらにもこれが適用されます。システム・カタログ内の該当する項目は ATTACH 操作中に除去されます。

ターゲット表の XML データに対する非パーティション化索引を保守するために、ATTACH 後に SET INTEGRITY を実行する必要があります。

DETACH の場合、ソース表の XML 列に対する非パーティション索引はターゲット表に継承されません。

XML 列パス索引

XML 列パスの索引は、常に非パーティション索引です。ソース表とターゲット表の XML 列パス索引は、ロールインおよびロールアウト操作中に保守されます。

ATTACH の場合、ターゲット表上の非パーティション化 XML 列パス索引は DB2 データベース・マネージャーによって保守されます (これとは対照的に、他の非パーティション化索引は ATTACH 完了後の SET INTEGRITY 操作中に保守されます)。その後、ソース表の XML 列パス索引がドロップされて、それに対応するカタログ項目が除去されます。これは、ターゲット表の列パス索引がパーティション化されないためです。

ロールアウトの場合、XML 列パス索引がパーティション化されないことに注意してください。非パーティション化索引はスタンドアロン・ターゲット表に持ち越されません。しかし、XML 列を持つ表に外部ユーザーがアクセスするには、(列ごとに 1 つの) XML 列パス索引が表に既に存在する必要があります。このため、ターゲット表を使用する前に XML 列パス索引がターゲット表に作成済みでなければなりません。列パス索引が作成される時点は、DETACH 操作中にデタッチされる従属表があるかどうかに応じて異なります。デタッチされる従属表がない場合は、DETACH 操作中にパス索引が作成されます。存在する場合は、デタッチ従属オブジェクトを保守する SET INTEGRITY または MQT リフレッシュによって作成されます。

DETACH 後、ターゲット表に作成された XML 列パス索引は、その表の他のすべての索引と同じ索引オブジェクトに格納されます。

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。ALTER TABLE ステートメントで ATTACH PARTITION 節を使用すると、データのロールインが簡単になります。

始める前に

データ・サーバーから独立したアプリケーション・ロジックを使用して、アタッチ操作の前に、範囲の妥当性その他の制約検査を含むデータ保全性検査を実行できる場合、新規にアタッチされたデータはずっと迅速に使用可能になります。SET INTEGRITY...ALL IMMEDIATE UNCHECKED ステートメントを使用して、範囲および制約の違反に対する検査をスキップすることにより、データのロールイン処理を最適化できます。この場合、ターゲット表のユーザー索引がすべてパーティション索引である限り、表は SET INTEGRITY 保留状態ではなくなり、即時に新規データがアプリケーションから使用可能になります。

アタッチ操作の後で維持する必要のある非パーティション索引 (ただし、XML 列パス索引を除く) が表にある場合、SET INTEGRITY...ALL IMMEDIATE UNCHECKED ステートメントは、SET INTEGRITY...IMMEDIATE CHECKED ステートメントであるかのように振る舞います。すべての整合性処理、非パーティション索引の保守、および表の状態の遷移は、SET INTEGRITY...IMMEDIATE CHECKED ステートメントが発行された場合のように実行されます。この動作により、SET INTEGRITY...ALL IMMEDIATE UNCHECKED を使用するロールイン・ス

クリプトによる処理が、そのロールイン・スクリプトの使用開始後に非パーティション索引がターゲット表に作成された場合でも停止しないようにします。

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。

- 表に対する SELECT 特権、および表のスキーマに対する DROPIN 特権
- 表に対する CONTROL 特権
- DATAACCESS 権限

このタスクについて

データ・パーティションのアタッチにより、既存の表 (ソース表) が新しいデータ・パーティションとしてターゲット表にアタッチされます。 DB2 V10.1 以降のリリースで、ALTER TABLE ステートメントに ATTACH PARTITION 節を付けて使用することにより、パーティション表にデータ・パーティションをアタッチする場合、ターゲット・パーティション表はオンライン状態を維持し、この表に対して RS、CS、または UR の分離レベルで実行されている動的照会は実行を継続します。

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対象のターゲット表が、既存のパーティション表で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 つ持ち、ATTACHED データ・パーティションまたは 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。
- ソース表は型付き表になることができません。
- ソース表は、範囲がクラスター化された表になることができません。
- ソース表の定義とターゲット表の定義は一致していなければなりません。
- 列の数、タイプ、および順序は、ソースとターゲットで一致していなければなりません。
- デフォルト値があるかどうかについて、ソース表の列とターゲット表の列で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、ソース表の列とターゲット表の列で一致していなければなりません。
- ソース表とターゲット表で、VALUE COMPRESSION 節および COMPRESS SYSTEM DEFAULT 節などの圧縮の指定が、一致していなければなりません。
- ソース表とターゲット表で、DATA CAPTURE、ACTIVATE NOT LOGGED INITIALLY、および APPEND オプションに関する指定が、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、対応するソース列は生成列ではない場合でも、データ・パーティションのアタッチは可能です。次のステートメントは、アタッチされた行の生成列に、値を生成します。

  ```
  SET INTEGRITY FOR table-name
    ALLOW WRITE ACCESS
    IMMEDIATE CHECKED FORCE GENERATED
  ```

  生成列に対応するソース表の列は、タイプと NULL 可能性について一致しなければなりません。ただし、デフォルト値については必要ありません。推奨されるアプローチは、アタッチ操作のソース表が、生成列に正しい生成値を確実に持つようにするというものです。推奨されるアプローチに従う場合は、FORCE GENERATED オプションを使用する必要はなく、次のステートメントを使用できます。

  ```
  SET INTEGRITY FOR table-name
    GENERATED COLUMN
    IMMEDIATE UNCHECKED
  ```

  このステートメントは、生成列の検査を省略するように指定しています。

  ```
  SET INTEGRITY FOR table-name
    ALLOW WRITE ACCESS
    IMMEDIATE CHECKED
    FOR EXCEPTION IN table-name USE table-name
  ```

  このステートメントは、アタッチされたデータ・パーティションの整合性検査を実行しますが、生成列検査は実行しません。
- ターゲット列が ID 列であり、ソース列は ID 列ではない場合でも、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)。
- データ・パーティション名を指定する場合、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。
- 非パーティション表を使用する場合、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の間で、タイプ (DMS または SMS)、ページ・サイズ、エクステント・サイズ、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。プリフェッチ・サイズが一致しない場合、警告が戻されます。ソース表の索引表スペースと、ターゲット表のパーティション索引によ

って使用される索引表スペースとの間で、タイプ、データベース・パーティション・グループ、ページ・サイズ、エクステント・サイズが一致している必要があります。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、およびページ・サイズが一致している必要があります。パーティション表を使用する場合、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の間で、タイプ、ページ・サイズ、エクステント・サイズ、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。

- 構造化、XML、または LOB 列を持つパーティション表への ALTER TABLE ATTACH ステートメントを発行する場合、ソース表における構造化、XML、または LOB 列の INLINE LENGTH は、ターゲット表における対応する構造化、XML、または LOB 列の INLINE LENGTH と一致する必要があります。
- ATTACH PARTITION 節と共に REQUIRE MATCHING INDEXES 節を使用する場合、ソース表と一致しないパーティション索引がターゲット表に存在するなら、SQL20307N が戻されます。
- ターゲット表上のパーティション化された各ユニーク索引に対して一致する索引を持たないソース表をアタッチした場合、アタッチ操作は失敗して、エラー SQL20307N、理由コード 17 が返されます。
- 索引の据え置きクリーンアップ・メカニズムを使用する MDC ロールアウトをパーティション索引に対して実行することはできません。MDC ロールアウトの結果として索引の据え置きクリーンアップ操作が表で実行されている場合、索引の据え置きクリーンアップ操作の影響を受ける RID 索引がソース表に保持されているなら、アタッチ操作は実行できません。アタッチ操作中に保持されるのではなくて、再作成される索引については、この制限を受けません。
- ターゲット表の XML データ形式とは異なる XML データ形式を持つソース表をアタッチする操作は、サポートされません。
- バージョン 9.5 以前の XML レコード形式を使用する XML 列が表に含まれる場合、バージョン 9.7 以降のレコード形式を使用する XML 列を持つパーティション表に表をアタッチする操作はサポートされません。

  表をアタッチする前に、ターゲット・パーティション表のレコード形式と一致するように表の XML レコード形式を更新する必要があります。以下の 2 つの方法のいずれかによって、表の XML レコード形式を更新できます。

  – ADMIN_MOVE_TABLE プロシージャーを使用して、オンライン表移動を表に対して実行します。
  – 以下の手順を実行します。
    1. EXPORT コマンドを使って表データのコピーを作成します。
    2. TRUNCATE ステートメントを使って表のすべての行を削除し、表に割り振られたストレージを解放します。
    3. LOAD コマンドを使って表の中にデータを追加します。

  表の XML レコード形式を更新した後、ターゲット・パーティション表に表をアタッチします。
- 表に非パーティション索引がある場合、その表に新しいデータ・パーティションを作成した追加操作またはアタッチ操作と同じトランザクションにおいてその表

が排他モードでロックされていないと、そのトランザクションでその新しいパーティションにアクセスすることはできません (SQL0668N、理由コード 11)。

アタッチ操作を実行する前に、ターゲット表のそれぞれのパーティション化索引に一致する索引をソース表に作成します。パーティション化索引を一致させることで、ロールイン操作の効率が上がり、必要なアクティブ・ログ・スペースが減ります。ソース表の索引を適切に準備しない場合、データベース・マネージャーはそれらを自動的に保守する必要があります。ロールインによってパーティション化索引の保守コストを増やさないために、パーティションのアタッチ操作で REQUIRE MATCHING INDEXES を指定することができます。REQUIRE MATCHING INDEXES を指定すると、ターゲット上のパーティション索引に一致する索引がソース表にない場合、アタッチ操作は失敗します。その後、修正アクションを行ってアタッチ操作を再発行することができます。

さらに、アタッチ操作を実行する前に、ソース表にある余分な索引をすべてドロップしてください。ソース表における余分な索引とは、ターゲット表の索引と一致しない索引、またはターゲット表の非パーティション索引と一致する索引です。アタッチ操作を実行する前に余分な索引をドロップすると、操作の実行が速くなります。

例えば、ORDERS というパーティション表に、(一年の各月に対応する) 12 個のデータ・パーティションがあるとします。毎月終わりに NEWORDERS という別の表が、パーティション化された ORDERS 表にアタッチされます。

1. パーティション化索引を ORDERS 表に作成します。

   ```
   CREATE INDEX idx_delivery_date ON orders(delivery) PARTITIONED
   CREATE INDEX idx_order_price ON orders(price) PARTITIONED
   ```

2. アタッチ操作の準備として、対応する索引を NEWORDERS 表に作成します。

   ```
   CREATE INDEX idx_delivery_date_for_attach ON neworders(delivery)
   CREATE INDEX idx_order_price_for_attach ON neworders(price)
   ```

3. アタッチ操作には、以下の 2 つのステップが含まれます。

   a. ATTACH。 ORDERS 表のパーティション化索引に一致する NEWORDERS 表の索引は保持されます。

      ```
      ALTER TABLE orders ATTACH PARTITION part_jan2009
         STARTING FROM ('01/01/2009')
         ENDING AT ('01/31/2009') FROM TABLE neworders
      ```

      ORDERS 表は自動的に SET INTEGRITY ペンディング状態に置かれます。アタッチ操作の完了後、idx_delivery_date_for_attach 索引および idx_order_price_for_attach 索引の両方が ORDERS 表に含まれるようになります。この操作中にデータの移動は発生しません。

   b. SET INTEGRITY (保全性の設定)。新しくアタッチされたパーティションに対して範囲検査が行われます。制約が存在する場合は、それらがすべて実施されます。完了すると、新しくアタッチされたデータは、データベース内で表示できるようになります。

      ```
      SET INTEGRITY FOR orders IMMEDIATE CHECKED
      ```

ターゲット表に非パーティション索引が存在する場合、SET INTEGRITY ステートメントでは、新しくアタッチされたパーティションのデータに対する制約の検査や範囲の妥当性検査といった他のタスクに加えて、索引の保守も実行されなければな

りません。非パーティション索引の保守では、新しくアタッチされたパーティションのデータ量、それぞれの非パーティション索引のキー・サイズ、および非パーティション索引の数に比例して、大容量のアクティブ・ログ・スペースが必要になります。

ソース表の表スペース ID とオブジェクト ID を使用して、新しいデータ・パーティション上のそれぞれのパーティション索引に関する項目が SYSINDEXPARTITIONS カタログ表に追加されます。ID 情報は (非パーティション表の場合は) SYSINDEXES 表から、または (パーティション表の場合は) SYSINDEXPARTITIONS 表から取得されます。索引 ID は、対応するターゲット表のパーティション索引から取得されます。

ソース表がパーティション化されている場合、ターゲット表のパーティション化索引に一致するソース表のパーティション化索引はアタッチ操作の一部として保持されます。 SYSINDEXPARTITIONS 表の索引パーティション項目が更新されて、新しい索引 ID を持つ新しいターゲット表の索引パーティションとして示されます。

データ・パーティションのアタッチ時に、索引およびデータに関するいくつかの統計が、新規パーティションのソース表からターゲット表に持ち越されます。特に、ターゲット上の新規パーティションに関する SYSDATAPARTITIONS 表および SYSINDEXPARTITIONS 表のすべてのフィールドがソースから移植されます。ソース表が非パーティション化されている場合、これらの統計は SYSTABLES 表および SYSINDEXES 表から得られます。ソース表が単一パーティションから成るパーティション表である場合、これらの統計は単一ソース・パーティションの SYSDATAPARTITIONS 表および SYSINDEXPARTITIONS 表から得られます。

注: 持ち越される統計は SYSINDEXES 表および SYSTABLES 表の集約統計に影響を与えないため、アタッチ操作の完了後に RUNSTATS 操作を実行してください。

SET INTEGRITY...ALL IMMEDIATE UNCHECKED での非パーティション索引の保守。パーティション表に対して SET INTEGRITY...ALL IMMEDIATE UNCHECKED を発行して、新規にアタッチされたパーティションの範囲検査をスキップする場合、XML 列パス索引以外の非パーティション索引が表にあると、SET INTEGRITY...ALL IMMEDIATE UNCHECKED は以下のように振る舞います。

- SET INTEGRITY...ALL IMMEDIATE UNCHECKED ステートメントが 1 つのターゲット表を参照している場合、SET INTEGRITY...ALLOW WRITE ACCESS...IMMEDIATE CHECKED ステートメントが発行されたかのように振る舞います。SET INTEGRITY...ALL IMMEDIATE UNCHECKED ステートメントは、すべての非パーティション索引 (ただし、XML 列パス索引を除く) を保守し、その他のすべての整合性処理を実行し、SYSCAT.TABLES カタログ・ビューの CONST_CHECKED 列内にある制約チェック・フラグの値を更新し、制約違反が検出されるとエラーを戻して即時に停止します。
- SET INTEGRITY...ALL IMMEDIATE UNCHECKED ステートメントが複数のターゲット表を参照している場合、エラーが戻されます (SQL20209N および理由コード 13)。

SET INTEGRITY での無効なパーティション索引の再作成。SET INTEGRITY ステートメントは、新規にアタッチされたパーティションのパーティション索引オブジ

ェクトが無効であるかどうかを検出できます。無効の場合、必要に応じて、パーティション索引の再作成を実行します。

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、『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
```

デタッチされたパーティションは、以下の各ステップを完了するまでデタッチされたままになります。

1. 必要な SET INTEGRITY ステートメントを実行して、デタッチされた従属を増分リフレッシュします。
2. バージョン 9.7.1 以降では、デタッチが非同期的に完了するのを待ちます。この処理を迅速に行うには、デタッチ操作より前に開始した表へのすべてのアクセスを完了または終了させます。
3. ソース表に非パーティション索引がある場合は、非同期索引クリーンアップの完了を待ちます。この処理を迅速に行う上で、ソース表上の非パーティション索引をドロップすることは 1 つの選択肢になり得ます。

すぐにアタッチ操作を実行する場合に 1 つの選択肢となり得るのは、新しいソース表を作成することです。

ソース表が、システム表、ビュー、型付き表、表 ORGANIZED BY KEY SEQUENCE、作成済み一時表、または宣言済み一時表である。 (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)
```

アタッチ操作中に索引が欠落していて、エラーが戻される (SQL20307N 理由コード 18)

アタッチ操作では、ターゲット表のパーティション化索引に対応する、ソース表では欠落している索引が暗黙的に作成されます。欠落している索引の暗黙的な作成は、完了するまでに時間がかかります。オプションとして、アタッチ操作で欠落する索引が検出された場合にエラー条件を作成することもできます。このオプションは ERROR ON MISSING INDEXES と呼ばれ、アタッチ操作のオプションの 1 つです。これが発生した場合に戻されるエラーは SQL20307N、SQLSTATE 428GE、理由コード 18 です。マッチしない索引に関する情報は管理ログに記録されます。

アタッチ操作では、ターゲット表のパーティション化索引に対応しないソース表上の索引がドロップされます。このようなマッチしない索引を識別してドロップする操作が完了するには、時間がかかります。アタッチ操作を試みる前に、これらの索引をドロップする必要があります。

ターゲット表上のマッチしない索引がユニーク索引である場合、または REJECT INVALID VALUES 節を使って XML 索引が定義されている場合には、アタッチ操作中にエラーが戻されます (SQL20307N 理由コード 17)

ターゲット表上のパーティション化索引に対応する索引がソース表に存在しない場合、ERROR ON MISSING INDEXES が使用されていなければ、次のような結果が発生する可能性があります。

1. ターゲット表上のマッチしない索引がユニーク索引である場合、または REJECT INVALID VALUES 節を使って XML 索引が定義されている場合には、アタッチ操作が失敗してエラー・メッセージ SQL20307N、SQLSTATE 428GE、理由コード 17 が戻されます。
2. ターゲット表上のマッチしない索引が上記の条件を満たさない場合、アタッチ操作中にソース表の索引オブジェクトが無効であるとマーク付けされます。アタッ

チ操作は正常に完了しますが、新しいデータ・パーティション上の索引オブジェクトは無効とマーク付けされます。新しくアタッチされたパーティション上で索引オブジェクトを再作成するには、SET INTEGRITY 操作を使用します。データ・パーティションをアタッチした後には、次の操作として、通常、これを実行します。索引の再作成には時間がかかります。

ソース表とターゲット表の間で索引が対応しない場合、管理ログにその詳細が記述されます。

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
```

ATTACH PARTITION 中にソース表の索引をターゲット表のパーティション索引にマッチさせるための条件

ターゲット表のパーティション索引のすべての索引キー列は、ソース表の索引の索引キー列にマッチする必要があります。索引の他のプロパティーがすべて同じである場合、ソース表の索引はターゲット表のパーティション索引にマッチすると見なされます。つまり、ソース表の索引をターゲット表の索引として使用することができます。以下の表を使用して、索引がマッチするかどうかを判別できます。

次の表は、ターゲット索引がパーティション化されている場合にのみ該当します。ソースとターゲットがマッチすると見なされるすべての場合において、ターゲット索引プロパティーはソース索引によって想定されます。

表 14. ターゲット索引プロパティーがソース索引プロパティーと異なる場合にソース索引がマッチするかどうかを判別する

規則番号	ターゲット索引プロパティー	ソース索引プロパティー	ソース索引がマッチするかどうか
1.	非ユニーク	ユニーク	はい (索引が XML 索引でない場合)。
2.	ユニーク	非ユニーク	いいえ
3.	列 X が降順	列 X が昇順	いいえ
4.	列 X が昇順	列 X が降順	いいえ
5.	パーティション	非パーティション	いいえ (注: ソース表がパーティション化されていることを想定)
6.	pctfree n1	pctfree n2	はい
7.	level2pctfree n1	level2pctfree n2	はい
8.	minpctused n1	minpctused n2	はい
9.	反転スキャンを許可しない	反転スキャンを許可する	はい (反転スキャンを許可するかどうかに関わらず、物理的な索引構造は同じ)
10.	反転スキャンを許可する	反転スキャンを許可しない	はい (9 と同じ理由)
11.	pagesplit [L|H|S]	pagesplit [L|H|S]	はい
12.	サンプル統計	詳細な統計	はい
13.	詳細な統計	サンプル統計	はい
14.	非クラスター	CLUSTER	はい

表 14. ターゲット索引プロパティーがソース索引プロパティーと異なる場合にソース索引がマッチするかどうかを判別する (続き)

規則番号	ターゲット索引プロパティー	ソース索引プロパティー	ソース索引がマッチするかどうか
15.	CLUSTER	非クラスター	はい。索引はクラスタリング索引になりますが、データが再編成されるまでは、この索引に従ってデータがクラスター化されません。この索引パーティションに従ってデータをクラスターにアタッチした後にパーティション・レベルの再編成を使用できます。
16.	無効値を無視	無効値を拒否	はい
17.	無効値を拒否	無効値を無視	いいえ。無効値を拒否するというターゲット索引プロパティーに従う必要があり、この索引制約に違反する行がソース表に含まれる可能性があります。
18.	索引の圧縮が有効	索引の圧縮が無効	はい。注: 索引が再作成されるまでは、基礎となる索引データの圧縮は発生しません。
19.	索引の圧縮が無効	索引の圧縮が有効	はい。注: 索引が再作成されるまでは、索引データの圧縮解除は発生しません。

注: DB2 バージョン 9.7 以前のリリースを使用して作成したマルチディメンション・クラスタリング (MDC) 表 (非パーティション化ブロック索引を持つ) を、DB2 バージョン 9.7 フィックスパック 1 以降のリリースを使用して作成した新規の MDC パーティション表 (パーティション化ブロック索引を持つ) に、ERROR ON MISSING INDEXES 節を指定してアタッチしようとすると、規則番号 5 によって ALTER TABLE ... ATTACH PARTITION ステートメントは失敗しエラー・メッセージ SQL20307N、SQLSTATE 428GE が戻ります。 ERROR ON MISSING INDEXES 節を除去すると、アタッチ操作中に索引がデータベース・マネージャーによって保守されるため、アタッチが完了します。エラー・メッセージ SQL20307N、SQLSTATE 428GE を受け取った場合には、ERROR ON MISSING INDEXES 節を除去することを考慮してください。

別の方法として、オンライン表移動の手順を使用すると、非パーティション化ブロック索引を持つ MDC パーティション表を、パーティション化ブロック索引を持つ表に変換できます。

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。

始める前に

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 PARTITION 操作を実行するユーザーは、ソース表に対する ALTER、SELECT、および DELETE 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。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。
  - 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 権限か DATAACCESS 権限があるユーザーのみです。

このタスクについて

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。

DB2 バージョン 9.7 フィックスパック 1 以降のリリースでは、DETACH PARTITION 節を指定した ALTER TABLE ステートメントを使用して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間、ソースのパーティション表はオンラインのままです。この表に対する実行中の照会は、継続して実行されます。デタッチされるデータ・パーティションは、以下の 2 フェーズのプロセスで、スタンドアロン表に変換されます。

1. ALTER TABLE...DETACH PARTITION 操作は、論理的にデータ・パーティションをパーティション表からデタッチします。
2. 論理的にデタッチされたパーティションは、非同期パーティション・デタッチ・タスクによって、スタンドアロン表に変換されます。

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)、デタッチされた従属表のすべてに SET INTEGRITY ステートメントが実行された後にのみ、非同期パーティション・デタッチ・タスクが開始されます。

デタッチされた従属表がない場合は、ALTER TABLE...DETACH PARTITION ステートメントを発行したトランザクションがコミットされた後に、非同期のパーティション・デタッチ・タスクは開始されます。

制約事項

ソース表が、 DB2 バージョン 9.7 以前のリリースを使用して作成された MDC 表である場合、ブロック索引はパーティション化されません。ALTER TABLE...DETACH PARTITION 操作と同じ作業単位の中で、新しくデタッチされた表にアクセスすることはできません。MDC 表ではパーティション化ブロック索引はサポートされません。その場合、ブロック索引は、ALTER TABLE...DETACH PARTITION 操作がコミットされた後、表への最初のアクセス時に作成されます。デタッチより前の時点でソース表に他のパーティション索引が存在していた場合、ブロック索引の作成を可能にするために、ターゲット表の索引オブジェクトは無効とマーク付けされます。その結果、ブロック索引が作成されてパーティション索引が再作成される間はアクセス時間が長くなります。

ソース表が、DB2 V9.7 フィックスパック 1 以降のリリースを使用して作成された MDC 表である場合、ブロック索引はパーティション化されており、パーティション化された索引は、再作成の必要もなく、デタッチのターゲット表の索引になります。

DETACH PARTITION 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 PARTITION 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 PARTITION が許可されません。 RI のリレーションシップが施行される表があり、その親表からデータ・パーティションをデタッチする場合、回避策を使用できます。以下の例では、すべてのステートメントが同一の作業単位 (UOW) 内で実行され、並行更新がロックアウトされます。

```
// Change the RI constraint to informational:
ALTER TABLE child ALTER FOREIGN KEY fk NOT ENFORCED;
```

```
ALTER TABLE parent DETACH PARTITION p0 INTO TABLE pdet;

SET INTEGRITY FOR child OFF;

// Change the RI constraint back to enforced:
ALTER TABLE child ALTER FOREIGN KEY fk ENFORCED;

SET INTEGRITY FOR child ALL IMMEDIATE UNCHECKED;
// Assuming that the CHILD table does not have any dependencies on partition P0,
//   and that no updates on the CHILD table are permitted until this UOW is complete,
//   no RI violation is possible during this UOW.

COMMIT WORK;
```

- 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)、デタッチされた従属表に対して SET INTEGRITY ステートメントを実行して、表を増分的に保守する必要があります。 DB2 V9.7 フィックスパック 1 以降のリリースでは、デタッチされたすべての従属表に対して SET INTEGRITY ステートメントを実行した後、非同期のパーティション・デタッチ操作によって、このデータ・パーティションはスタンドアロンのターゲット表に変換されます。非同期のパーティション・デタッチ操作が完了するまで、ターゲット表は使用できません。

手順

1. 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、DETACH PARTITION 節を指定して ALTER TABLE ステートメントを発行します。
2. オプション: 新しくデタッチされたスタンドアロン表で同じ制約を使用するには、デタッチ操作の完了後に、ターゲット表に対して ALTER TABLE ... ADD CONSTRAINT を実行します。

   ソース表で索引がパーティション化されていた場合には、制約を満たすために必要なすべての索引がターゲット表に既に存在します。

タスクの結果

デタッチされたパーティションの名前は、後続のアタッチですぐに再使用できるように、システム生成名 (SQL*yymmddhhmmssxxx* という形式を使用) に変更されます。

デタッチ対象のデータ・パーティションに関してソース表で定義されているそれぞれの索引パーティションは、ターゲット表の 1 つの索引になります。パーティションのデタッチ操作中に索引オブジェクトが物理的に移動されることはありません。ただし、デタッチされる表パーティションの索引パーティションに関するメタデータは、カタログ表 SYSINDEXPARTITIONS から削除されます。新しい表に関する新しい索引項目が、パーティションのデタッチ操作の結果として SYSINDEXES に追加されます。元の索引 ID (IID) は保持され、ソース表と同じように引き続きユニークになります。

ターゲット表に引き続き存続する索引の索引名は、システムによって生成されます (使用される形式は SQL*yymmddhhmmssxxx*)。これらの索引のスキーマはターゲット表のスキーマと同じですが、パス索引、領域索引、MDC ブロック索引、および ITC ブロック索引は例外です (これらは SYSIBM スキーマに含まれます)。その他のシステム作成索引 (例えばユニーク・キー制約や主キー制約を実施するための索引) は

ターゲット表のスキーマを使用します。これは、索引はデタッチされた表に持ち越されるものの、制約は持ち越されないためです。 RENAME ステートメントを使用すると、SYSIBM スキーマに含まれない索引を名前変更できます。

ソース表の作成時に指定された表レベルの INDEX IN オプションは、ターゲット表に継承されません。その代わりに、パーティション・レベルの INDEX IN (指定されている場合) またはデタッチ・パーティションのデフォルト索引表スペースが、引き続きターゲット表の索引表スペースになります。

データ・パーティションのデタッチ操作では、いくつかの統計がデタッチ対象パーティションからターゲット表に持ち越されます。特に、パーティション索引に関する SYSINDEXPARTITIONS の統計が、新しくデタッチされた表に関する項目 SYSINDEXES に持ち越されます。 SYSDATAPARTITIONS の統計は、新しくデタッチされた表に関する SYSTABLES にコピーされます。

次のタスク

パーティションのデタッチ操作の完了後に持ち越されない統計が多数あるため、DETACH PARTITION 操作の完了後、新しくデタッチされた表とソース表の両方に対して **RUNSTATS** を実行してください。

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と、このデータ・パーティションは、スタンドアロンの、非パーティション化された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。

DETACH 後のターゲット表は、ソース表で定義されたパーティション化索引をすべて継承します。このような索引には、システム生成による索引およびユーザー定義の索引が含まれます。デタッチ操作中に索引オブジェクトは物理的に移動されません。デタッチされるデータ・パーティションの索引パーティション・メタデータは SYSINDEXPARTITIONS カタログから除去されます。新しい表に関して SYSINDEXES に新しい項目が追加されます。ソース表のパーティション化索引に関する索引 ID (IID) は、ターゲット表の索引の IID になります (表において IID は引き続きユニークになるため、デタッチ中に変更されません)。

新しい表に存続する索引に対して、SQLyymmddhhmmssxxx という形式の索引名がシステムにより生成されます。パス索引、領域索引、および MDC または ITC 索引は SYSIBM スキーマに含まれるようになります。その他のすべての索引は、新しい表のスキーマに含まれるようになります。システム生成による索引 (例えばユニーク・キー制約や主キー制約を実施するための索引) は、新しい表に持ち越されるため、新しい表のスキーマに含まれます。 DETACH 後、ソース表に対する制約はターゲット表に継承されません。

RENAME ステートメントを使用すると、SYSIBM スキーマに含まれない索引を別の時点で名前変更できます。

デタッチ操作の完了後に新しい表に対して ALTER TABLE ... ADD CONSTRAINT ステートメントを使用すると、ソース表と同じ制約を新しい表に対して実施できます。

ソース表に対する表レベルの INDEX IN 節で指定される表スペースの場所は、新しいターゲット表に継承されません。代わりに、パーティション・レベルの INDEX IN 節で指定される表スペースの場所、または新しい表のデフォルト索引表スペースが、引き続き新しい表の索引表スペースの場所になります。

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
  - INLINE LENGTH (インライン長)
  - 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 (MDC) または挿入時クラスタリング (ITC) 表である場合、ターゲット表も同じディメンション列で定義される MDC または ITC 表になります。
- 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。
- ソース・データ・パーティションに関する SYSDATAPARTITIONS 内の INDEX_TBSPACEID 列値 (パーティション・レベルの索引表スペース) は表スペ

ース名に変換され、ターゲット表に関する SYSTABLES の INDEX_TBSPACE 値になります。 CREATE TABLE ステートメントで表レベルの INDEX IN <table space> によって指定される索引表スペースは、ターゲット表に継承されません。
- 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表または一時表、範囲がクラスター化された表、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。
- 表レベルの索引表スペース (SYSCAT.TABLES カタログ・ビューの INDEX_TBSPACE 列)。 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
- 主キー制約とユニーク・キー制約
- 非パーティション化索引に関する統計は継承されません。
- 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。

データ・パーティションのデタッチ・フェーズ

DB2 バージョン 9.7 フィックスパック 1 以降のリリースでは、データ・パーティション表からのデータ・パーティションのデタッチ処理は、 2 つのフェーズで構成されています。最初のフェーズで、パーティションを表から論理的にデタッチし、2 番目のフェーズで、そのデータ・パーティションをスタンドアロン表に変換します。

デタッチ処理は、ALTER TABLE...DETACH PARTITION ステートメントが発行された時に開始されます。

1. ALTER TABLE...DETACH PARTITION 操作は、論理的にデータ・パーティションをパーティション表からデタッチします。
2. 論理的にデタッチされたパーティションは、非同期パーティション・デタッチ・タスクによって、スタンドアロン表に変換されます。

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)、デタッチされた従属表のすべてに SET INTEGRITY ステートメントが実行された後にのみ、非同期パーティション・デタッチ・タスクが開始されます。

デタッチされた従属表がない場合は、ALTER TABLE...DETACH PARTITION ステートメントを発行したトランザクションがコミットされた後に、非同期のパーティション・デタッチ・タスクは開始されます。

DETACH 操作

ALTER TABLE...DETACH PARTITION 操作は、以下のように振る舞います。

- DETACH 操作は、非コミット読み取り (UR) 分離レベルの動的照会のために、処理の開始を待機することはありません。また、実行中の UR 動的照会を中断させることもありません。デタッチするパーティションに対して UR 照会がアクセスしている場合でさえも、このように振る舞います。
- UR ではない動的照会 (読み取りまたは書き込み照会) であっても、デタッチするパーティションをロックしなかった場合は、表に対する非 UR 動的照会の実行中に DETACH 操作を完了することができます。
- UR ではない動的照会が、デタッチするパーティションをロックした場合は、そのロックが解放されるまで DETACH 操作は待機します。
- DETACH 操作を進める前に、表に従属するすべての静的パッケージに対して、ハードな無効化を実行する必要があります。
- DETACH 操作ではカタログの更新が必要となるため、データ定義言語 (DDL) のステートメントに適用される以下の制約事項は、 DETACH 操作にも適用されます。
  - 表に対する新規の照会はコンパイルできません。
  - 表に対して実行される照会に関しては、バインドも再バインドもできません。

  これらの制約事項の影響を最小化するには、DETACH 操作の直後に COMMIT を発行します。

DETACH 操作の間、データ・パーティションの名前は、 SQL*yymmddhhmmssxxx*という形式のシステム生成名に変更され、 SYSCAT.DATAPARTITIONS のパーティションの STATUS は、デタッチされた従属表がない場合は L に設定され、ある場合は D に設定されます。

DETACH 操作の間、 SYSCAT.TABLES にターゲット表に関する項目が作成されます。デタッチされた従属表がある場合、表の TYPE は L に設定されます。デタッチされた従属表すべてに対して SET INTEGRITY が実行された後、TYPE は T に設定されますが、ターゲット表はまだ使用不可です。非同期パーティション・デタッチ・タスクによって、デタッチが完了してターゲット表が使用可能になります。

デタッチ操作中の動的 SQL のソフトな無効化によって、 ALTER TABLE...DETACH PARTITION ステートメントの前に開始されていた動的 SQL 照会は、 DETACH 操作と並行して、実行を継続できます。 ALTER TABLE...DETACH PARTITION ステートメントは、パーティション表に対して IX ロックを取得し、デタッチするデータ・パーティションに対して X ロックを取得します。

非同期パーティション・デタッチ・タスク

DETACH 操作がコミットされ、デタッチされた従属表がすべてリフレッシュされた後、非同期パーティション・デタッチ・タスクによって、論理的にデタッチされたパーティションが、スタンドアロン表に変換されます。

非同期パーティション・デタッチ・タスクは、デタッチ操作のフェーズ 1 より前に開始されていたパーティション表に対するアクセスがすべて完了するまで待機します。パーティション表に非パーティション索引がある場合、非同期パーティション・デタッチ・タスクは、据え置き索引クリーンアップとして、非同期索引クリーンアップ・タスクを作成します。アクセスの完了後、非同期パーティション・デタッチ・タスクは、論理的にデタッチされたパーティションをスタンドアロン表に変換して、デタッチ操作のフェーズ 2 を完了させます。

LIST UTILITIES コマンドを使用して、非同期パーティション・デタッチ・タスクの進行をモニターできます。**LIST UTILITIES** コマンドは、非同期パーティション・デタッチ・タスクが、以下のいずれかの状態にあるかを示します。

- パーティション表への既存アクセスの完了を待機中
- デタッチ操作の完了中およびターゲット表の使用可能化を実行中

データ・パーティション表に対する非同期パーティション・デタッチ

DB2 バージョン 9.7 フィックスパック 1 以降のリリースでは、 ALTER TABLE ... DETACH 操作によって開始されるデータ・パーティションのパーティション表からのデタッチは、非同期パーティション・デタッチ・タスクによって完了されます。このタスクは、パーティションが論理的にデタッチされたパーティションになった後に開始される非同期のバックグラウンド・プロセス (ABP) です。

非同期パーティション・デタッチ・タスクによって、パーティション表からのデータ・パーティションのデタッチ処理が高速化されます。パーティション表に従属マテリアライズ照会表 (MQT) がある場合、その MQT に対して SET INTEGRITY ステートメントが実行されるまで、このタスクは開始されません。

データ・パーティションのデタッチを非同期で実行することによって、 ALTER TABLE...DETACH PARTITION ステートメントの実行前に開始されていたパーティション表にアクセスする照会は、パーティションが速やかにデタッチされている間も、実行を継続することができます。

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)、デタッチされた従属表のすべてに SET INTEGRITY ステートメントが実行された後にのみ、非同期パーティション・デタッチ・タスクが開始されます。

デタッチされた従属表がない場合は、ALTER TABLE...DETACH PARTITION ステートメントを発行したトランザクションがコミットされた後に、非同期のパーティション・デタッチ・タスクは開始されます。

非同期パーティション・デタッチ・タスクは、以下の操作を実行します。

- ALTER TABLE...DETACH 操作がソフトな無効化を実行したキャッシュ・ステートメントに対して、ハードな無効化を実行します。
- ソースのパーティション表、およびスタンドアロンのターゲット表に関するカタログ項目を更新し、ターゲット表を使用可能にします。
- 非パーティション化ブロック索引を持ち、パーティション索引を持たないマルチディメンション・クラスタリング (MDC) 表の場合、ターゲット表に索引オブジェクトを作成します。ブロック索引は、非同期パーティション・デタッチ・タスクがコミットされた後、初めてターゲット表がアクセスされた時に作成されます。
- XML 列を持つ表の場合、ターゲット表にシステム・パス索引を作成します。
- デタッチされたパーティションを含む表スペースの最小リカバリー時間 (MRT) を更新します。
- 非パーティション索引の場合、非同期索引クリーンアップ (AIC) タスクを作成します。 AIC タスクは、非同期パーティション・デタッチの完了後に、索引のクリーンアップを実行します。
- 非パーティション索引が表にない場合は、データ・パーティション ID をリリースします。

非同期パーティション・デタッチ・タスクがパフォーマンスに与える影響

非同期パーティション・デタッチ・タスクがパフォーマンスに与える影響は最小限のものです。 ALTER TABLE...DETACH 操作がソフトな無効化を実行したキャッシュ・ステートメントに対して、ハードな無効化を実行することによって、デタッチされたパーティションに対するすべてのアクセスが完了するまで、このタスクは待機します。その後、タスクは、表およびパーティションに対して必要なロックを取得し、デタッチされたパーティションをスタンドアロン表に変換する処理を続行します。

非同期パーティション・デタッチ・タスクのモニター

分散デーモン、および非同期パーティション・デタッチ・タスクのエージェントは、 **LIST APPLICATIONS** コマンドの出力に、それぞれ **db2taskd** と **db2apd** というアプリケーション名で表示される内部システム・アプリケーションです。予期しな

い中断を防止するため、システム・アプリケーションを強制終了することはできません。分散デーモンは、データベースがアクティブである間、オンラインのままです。タスクは、デタッチが完了するまでアクティブのままです。デタッチの進行中にデータベースが非アクティブ化した場合、非同期パーティション・デタッチ・タスクは、データベースが再度アクティブ化した時に、再開されます。

LIST UTILITIES コマンドは、非同期パーティション・デタッチ・タスクが、以下のいずれかの状態にあるかを示します。

- パーティション表への既存アクセスの完了を待機中
- デタッチ操作の完了中およびターゲット表の使用可能化を実行中

以下の **LIST UTILITIES SHOW DETAIL** コマンドの出力例は、WSDB データベースの非同期パーティション・デタッチ・タスク活動を示しています。

```
ID                   = 1
Type                 = ASYNCHRONOUS PARTITION DETACH
Database Name        = WSDB
Partition Number     = 0
Description          = Finalize the detach for partition '4' of table 'USER1.ORDERS'.
Start Time           = 07/15/2009 14:52:14.476131
State                = Executing
Invocation Type      = Automatic
Progress Monitoring:
    Description      = Waiting for old access to the partitioned table to complete.
    Start Time       = 07/15/2009 14:52:51.268119
```

この LIST UTILITIES コマンドの出力にある、非同期パーティション・デタッチ・タスクのメインの「Description」から、デタッチされるデータ・パーティション、およびデタッチ操作によって作成されるターゲット表を特定できます。「Progress Monitoring」の記述には、非同期パーティション・デタッチ・タスクの現在の状態に関する情報が示されます。

注: 非同期パーティション・デタッチ・タスクは、非同期処理です。デタッチ操作のターゲット表が、いつ使用可能になるかを調べるために、SYSCAT.DATAPARTITIONS カタログ・ビューの STATUS 列を照会してデタッチ操作の完了時間を戻すストアード・プロシージャーを作成できます。

パーティション・データベース環境での非同期パーティション・デタッチ処理

パーティション・データベース環境では、データベース・パーティションの数に関わらず、1 つの DETACH 操作につき 1 つの非同期パーティション・デタッチ・タスクが作成されます。タスクはカタログ・データベース・パーティション上に作成され、必要に応じて、他のデータベース・パーティションに処理が配布されます。

非同期パーティション・デタッチ・タスクのエラー処理

非同期パーティション・デタッチ・タスクは、トランザクション・ベースです。タスクが失敗した場合、タスクによって実行されたすべての変更は、内部的にロールバックされます。非同期パーティション・デタッチ・タスクの処理中に発生したエラーは、すべて **db2diag** ログ・ファイルに記録されます。失敗したタスクは、システムによって後で再試行されます。

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。

このタスクについて

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場合、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。

特定の表スペース場所に新しいデータ・パーティションを追加するには、ALTER TABLE ADD PARTITION ステートメントでオプションとして IN 節を追加します。

データ・パーティションの表スペース場所とは別の特定の表スペース場所に新しいデータ・パーティションのパーティション化索引を追加するには、ALTER TABLE ADD PARTITION ステートメントでパーティション・レベルの INDEX IN 節をオプションとして追加します。 INDEX IN オプションを指定しない場合、デフォルトでは、新しいデータ・パーティションのすべてのパーティション索引はデータ・パーティションと同じ表スペースに格納されます。パーティション表にパーティション化索引が存在する場合、ADD PARTITION 節によって、新しいパーティション用の対応する空の索引パーティションが作成されます。それぞれのパーティション索引に対して、新しい項目が SYSCAT.INDEXPARTITIONS カタログ・ビューに挿入されます。

データ・パーティションの表スペース場所とは別の特定の表スペース場所に、新しいデータ・パーティションの LONG、LOB、または XML データを追加するには、ALTER TABLE ADD PARTITION ステートメントでパーティション・レベルの LONG IN 節をオプションとして追加します。

DB2 V10.1 以降のリリースでは、ALTER TABLE ステートメントに ADD PARTITION 節を指定してパーティション表にデータ・パーティションを追加するときに、ターゲット・パーティション表はオンライン状態を維持し、この表に対して RS、CS、または UR 分離レベルで実行されている動的照会は、実行を継続します。

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、217 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、INDEX 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。
- 表に非パーティション索引がある場合、その表に新しいデータ・パーティションを作成した追加操作またはアタッチ操作と同じトランザクションにおいてその表が排他モードでロックされていないと、そのトランザクションでその新しいパーティションにアクセスすることはできません (SQL0668N、理由コード 11)。

ADD 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.
```

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。SALES 表が値 900 までの 9 つの範囲 (0 - 100、101 -

200 など) を保持しているとします。この例では、表の最後に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・パーティションの表スペース場所とは別の特定の表スペース場所に新しいデータ・パーティションのパーティション化索引を追加するには、ALTER TABLE ADD PARTITION ステートメントでパーティション・レベルの INDEX IN 節をオプションとして追加します。 INDEX IN オプションを指定しない場合、デフォルトでは、新しいデータ・パーティションのすべてのパーティション化索引はデータ・パーティションと同じ表スペースに格納されます。パーティション表にパーティション化索引が存在する場合、ADD PARTITION によって、新しいパーティション用の対応する空の索引パーティションが作成されます。それぞれのパーティション索引に対して、新しい項目が SYSCAT.INDEXPARTITIONS カタログ・ビューに挿入されます。

例 2: ロング・データと索引をデータ・パーティションの残りの部分から分離させて、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。

```
ALTER TABLE newbusiness ADD PARTITION IN tsnewdata
INDEX IN tsnewindex LONG IN tsnewlong
```

データ・パーティションのドロップ

データ・パーティションをドロップするには、パーティションをデタッチし、デタッチ操作によって作成される表をドロップします。DETACH PARTITION 節を指定した ALTER TABLE ステートメントを使用して、パーティションのデタッチおよびスタンドアロン表の作成を行い、DROP TABLE ステートメントを使用して、その表をドロップします。

始める前に

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、ソース表に対して ALTER、SELECT、および DELETE を行う権限が必要です。
- また、ユーザーには、ターゲット表を作成する権限も必要です。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。
  - 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。
  - 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を持っていなければなりません。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。

手順

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、DETACH PARTITION 節を指定して ALTER TABLE ステートメントを発行します。

例

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。非同期パーティション・デタッチ・タスクによってターゲット表 junk が使用可能になったことを確認したら、表 junk をドロップします。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;
 -- ターゲット表が使用可能になったら、DROP TABLE ステートメントを実行します。
DROP TABLE junk;
```

次のタスク

DB2 バージョン 9.7 フィックスパック 1 以降のリリースでは ALTER TABLE ... DETACH を可能な限り高速化するために、非同期のパーティション・デタッチ・タスクによって、デタッチ操作が非同期で実行されます。デタッチされた従属表が存在する場合、非同期パーティション・デタッチ・タスクは開始されず、デタッチされたデータ・パーティションはスタンドアロン表になりません。この場合、すべてのデタッチされた従属表に対して、SET INTEGRITY ステートメントを発行する必要があります。 SET INTEGRITY の完了後、非同期パーティション・デタッチ・タスクが開始され、ターゲット表が使用可能になります。ターゲット表がアクセス可能になったら、表のドロップを実行できます。

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 (パーティションのデタッチ操作) した後、新規データを追加 (パーティションのアタッチ操作) 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。

始める前に

別の方法として、デタッチしたパーティションをアーカイブして、アタッチ操作を実行する前に、新規データを別のソース表にロードすることができます。次のシナリオでは、デタッチ操作がその他のステップに先行しますが、特定の要件に応じて、この操作を最後のステップにすることもできます。

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 に次の特権および権限が付与されていなければなりません。

- デタッチされるパーティションのターゲット表に対して、以下に示す権限の 1 つ以上。
  - データベースに対する CREATETAB 権限、表によって使用される表スペースに対する USE 特権、および以下の権限または特権のいずれか。
    - データベースに対する IMPLICIT_SCHEMA 権限 (新しい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 (新しい表のスキーマ名が既存のスキーマを指す場合)
  - DBADM 権限
- ソース表に対して、以下に示す特権および権限の 1 つ以上。
  - 表に対する SELECT、ALTER、および DELETE 特権
  - 表に対する CONTROL 特権
  - DATAACCESS 権限

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 に次の特権および権限が含まれていなければなりません。

- ソース表に対して、以下に示す権限または特権の 1 つ以上。
  - 表に対する SELECT 特権、および表のスキーマに対する DROPIN 特権
  - 表に対する CONTROL 特権
  - DATAACCESS 権限
- ターゲット表に対して、以下に示す権限または特権の 1 つ以上。
  - 表に対する ALTER および INSERT 特権
  - 表に対する CONTROL 特権
  - DATAACCESS 権限

手順

パーティション表でデータを入れ替えるには、ALTER TABLE ステートメントを発行します。 次の例は、2008 年 12 月のデータを除去し、それを 2010 年 12 月の最新データで置き換えることにより、STOCK 表を更新する方法を示しています。

1. STOCK 表から古いデータを除去する。

   ```
   ALTER TABLE stock DETACH PARTITION dec08 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** コマンドを使用すると、既存のデータが上書きされます。

   ```
   LOAD FROM data_file OF DEL REPLACE INTO newtable
   ```

注: デタッチされた従属表がある場合、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表に対して SET INTEGRITY ステートメントを実行する必要があります。SQL20285N が戻された場合は、非同期のパーティション・デタッチ・タスクが完了するのを待ってから、再度 SET INTEGRITY ステートメントを発行します。

3. 必要に応じて、データ・クレンジング・アクティビティーを実行します。これには、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異なる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*: 詳細すぎてウェアハウスに保管できないロー・データは、ロールインする前に集約する必要があります。
4. データを新規範囲としてアタッチする。

```
ALTER TABLE stock
  ATTACH PARTITION dec10
    STARTING '12/01/2008' ENDING '12/31/2010'
  FROM newtable;
```

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

データウェアハウスでの一般的な管理操作として、定期的に行う新規データのロールインおよび不要データのロールアウトがあります。これらの操作について、以下のシナリオで説明します。

シナリオ 1: データ・パーティションのデタッチによる不要データのロールアウト

次の例は、不要なデータ・パーティション (DEC01) を、STOCK という名前のパーティション表からデタッチする方法を示しています。デタッチされたデータ・パーティションは、そのデータを移動させることなく、STOCK_DROP という名前の表の作成に使用されます。

```
ALTER TABLE stock DETACH PART dec01 INTO stock_drop;
COMMIT WORK;
```

デタッチ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。デタッチされる従属表がソース表に対して定義されていない場合は、デタッチ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。

新規表は、削除したり、別の表にアタッチしたりできます。また、データを切り捨てて新規データをロードしてから、このソース表に再度アタッチすることもできます。これらの操作は、ソース表が従属表をデタッチした場合を除いて、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アクセス可能かどうかを判別するには、SYSCAT.TABDETACHEDDEP カタログ・ビューを照会します。デタッチされた表がアクセス不能であると判明した場合は、すべてのデタッチされた従属表に対して、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ます。デタッチされた従属表がすべて保守される前に、デタッチされた表にアクセスしようとすると、エラー・コード SQL20285N が戻されます。

シナリオ 2: 空の範囲の新規作成

次の例は、空のデータ・パーティション (DEC02) を、STOCK という名前のパーティション表に追加する方法を示しています。 STARTING FROM 節および ENDING AT 節は、この新規データ・パーティションの値の範囲を指定しています。

```
ALTER TABLE stock ADD PARTITION dec02
 STARTING FROM '12/01/2002' ENDING AT '12/31/2002';
```

この ALTER TABLE...ADD PARTITION ステートメントは、STOCK 表を対象として実行中の静的照会、または反復読み取り可能照会の完了を待って、関連パッケージを無効にします。そのような実行中の照会は、STOCK 表に対する追加操作が開始される前に、正常に完了します。STOCK 表に対して実行中の動的な反復読み取り不可照会は実行を継続し、追加操作と並行して実行することもできます。ALTER TABLE...ADD PARTITION ステートメントが STOCK 表に対して実行を開始した後は、その表にアクセスする新規の照会は、追加操作が完了するまで待機しなければなりません。

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 ステートメントを発行して、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。定義されている制約に違反する行はすべて、例外表 STOCK_EX に移動されます。

```
SET INTEGRITY FOR stock
 ALLOW READ ACCESS
 IMMEDIATE CHECKED
 FOR EXCEPTION IN stock USE stock_ex;

COMMIT WORK;
```

シナリオ 3: ロード済みのデータ・パーティションのアタッチによる新規データのロールイン

次の例は、アタッチ操作を使用して、新規のデータ範囲を既存のパーティション表 (STOCK という名前のターゲット表) に簡単にロードする方法を示しています。データをロードするのは新規の空の表 (DEC03) です。そこで検査を行い、必要な場合はクレンジングします。ターゲット表への影響はありません。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
- 正規化または集約によるデータの変換
  - *正規化*: 同じ値を異なる方法で表現している別のソースからのデータは、ロールイン処理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、ロールイン中に集約する必要があります。

このようにしてデータを整えておくと、新規のロード済みデータ・パーティションを、ターゲット表にアタッチすることができます。

```
CREATE TABLE dec03(...);
LOAD FROM data_file OF DEL REPLACE INTO dec03;
(data cleansing, if necessary)
ALTER TABLE stock ATTACH PARTITION dec03
 STARTING FROM '12/01/2003' ENDING AT '12/31/2003'
 FROM dec03;
```

アタッチ操作においては、STARTING FROM 節、ENDING AT 節のいずれかまたは両方を指定する必要があります。また、下限 (STARTING FROM 節) は、上限 (ENDING AT 節) より小さいか等しい必要があります。新たにアタッチするデータ・パーティションは、ターゲット表の既存のデータ・パーティション範囲に重なり合ってはなりません。既存の一番上の範囲の上限が MAXVALUE で定義されている場合は、新しく高範囲をアタッチしようとしても、新しい範囲が既存の高範囲と重なり合ってしまうためアタッチできません。同様の制限が、MINVALUE で終わる低位の範囲についても適用されます。さらには、中間に新規データ・パーティションを追加することもアタッチすることもできません。ただし、新規範囲が既存の各範囲のギャップに収まる場合は別です。ユーザーが境界を指定しない場合は、表の作成時に決定されます。

ALTER TABLE...ATTACH PARTITION ステートメントは、STOCK 表を対象として実行中の静的照会、または反復読み取り可能照会の完了を待って、関連パッケージを無効にします。そのような実行中の照会は、STOCK 表に対するアタッチ操作が開始される前に、正常に完了します。STOCK 表に対して実行中の動的な反復読み取り不可照会は実行を継続し、アタッチ操作と並行して実行することもできます。ALTER TABLE...ATTACH PARTITION ステートメントが STOCK 表に対して実行を開始した後は、その表にアクセスする新規の照会は、アタッチ操作が完了するまで待機しなければなりません。

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。SET INTEGRITY ステートメントは、新規にアタッチされたデータが定義された範

囲内にあることを検証するために必要です。また、索引および MQT などの他の従属オブジェクトに関する必要な保守アクティビティーを行います。新規データは、SET INTEGRITY ステートメントがコミットされるまで表示されません。ただし、SET INTEGRITY ステートメントがオンラインで実行されている場合、STOCK 表の既存のデータは、読み取り操作および書き込み操作用に完全にアクセス可能です。

ヒント: データ・サーバーから独立したアプリケーション・ロジックを使用して、アタッチ操作の前に、範囲の妥当性その他の制約検査を含むデータ保全性検査を実行できる場合、新規にアタッチされたデータはずっと迅速に使用可能になります。SET INTEGRITY...ALL IMMEDIATE UNCHECKED ステートメントを使用して、範囲および制約の違反に対する検査をスキップすることにより、データのロールイン処理を最適化できます。この場合、表は SET INTEGRITY 保留状態を脱します。そして、ターゲット表上に非パーティション・ユーザー索引が存在しない限り、即時に新規データがアプリケーションから使用可能になります。

注: SET INTEGRITY ステートメントの実行中は、その表に対して、データ定義言語 (DDL) のステートメントまたはユーティリティー操作を実行できません。このような操作には、以下のステートメントおよびコマンドが該当します。ただし、それらに限られているわけではありません。

- LOAD コマンド
- REDISTRIBUTE DATABASE PARTITION GROUP コマンド
- REORG INDEXES/TABLE コマンド
- ALTER TABLE ステートメント
  - ADD COLUMN
  - ADD PARTITION
  - ATTACH PARTITION
  - DETACH PARTITION
- CREATE INDEX ステートメント

SET INTEGRITY ステートメントは、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。

```
SET INTEGRITY FOR stock
 ALLOW WRITE ACCESS
 IMMEDIATE CHECKED
 FOR EXCEPTION IN stock USE stock_ex;
```

このトランザクションをコミットすると、表は使用可能になります。

```
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。この表を照会し、行を修正してから STOCK 表に挿入することができます。

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 37 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 38 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。

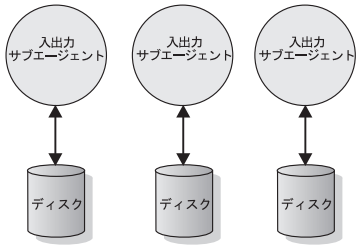


図 37. データ・ロード時に入出力の並列処理を利用する

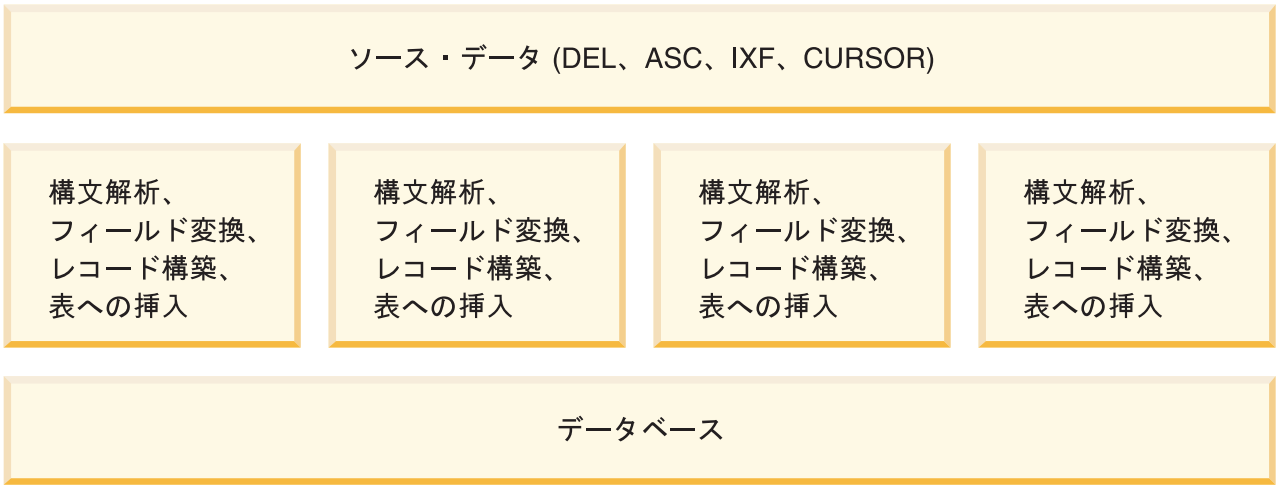


図 38. データ・ロード時のパーティション内の並列処理の利用

MDC および ITC 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) 表および挿入時クラスタリング (ITC) 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 または ITC 表には anyorder ファイル・タイプ修飾子が必要です。anyorder 修飾子を使用せずに MDC 表または ITC 表へのロードを実行すると、この修飾子はユーティリティーによって明示的に使用可能にされます。

MDC または ITC 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

複数のディメンションがある MDC 表をロードする場合のロード・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の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。バージョン 9.5 から、**LOAD** コマンドの DATA BUFFER オプションの値は、システムにさらに使用可能メモリーがある場合に、*util_heap_sz* を一時的に超えられるようになりました。.

すべての MDC および ITC 表にはブロック索引があるため、MDC または ITC 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 および ITC 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。 ITC 表には 1 つのセルしかなく、2 バイトのディメンション・キーを使用します。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。
- ターゲット表に XML 列がある場合、例外表を指定することはできません。
- 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 ロード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、SET INTEGRITY 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 SET INTEGRITY 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 ロード操作の場合は、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。

重要: バージョン 10.1 フィックスパック 1 以降、ALLOW READ ACCESS パラメーターは非推奨となっており、将来のリリースで除去される可能性があります。詳しくは、「*DB2 バージョン 10.1 の新機能*」の『LOAD コマンドの ALLOW READ ACCESS パラメーターが非推奨になった』を参照してください。

また、INGEST ユーティリティーはパーティション表をサポートしていて、LOAD コマンドで ALLOW READ ACCESS パラメーターを指定するよりもこのユーティリティーの方が、データ並行性と可用性に適しています。ターゲット表をロックすることなく、ファイルおよびパイプから大量のデータを移動することができます。また、経過時間または行数に基づいて、コミット後すぐにデータはアクセス可能になります。

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。

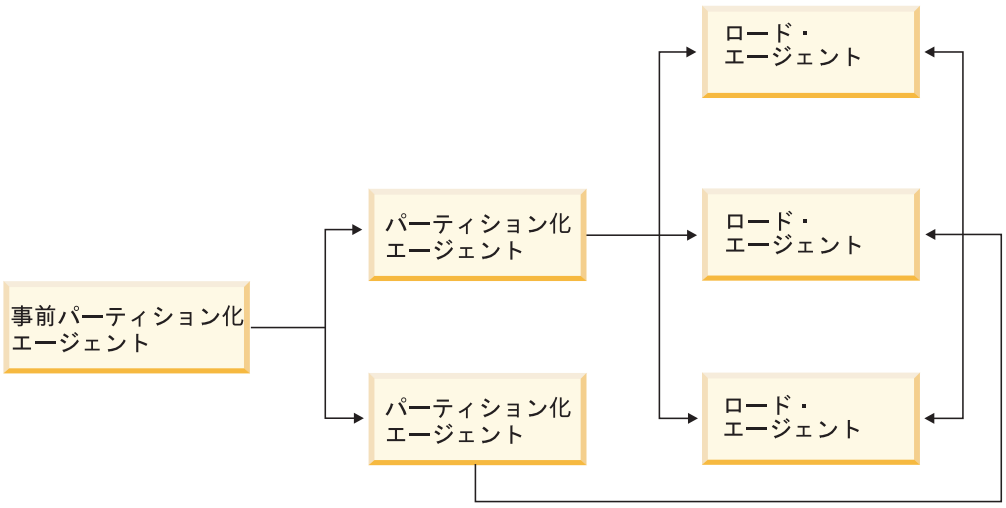


図 39. パーティション・データベース・ロードの概説： 事前パーティション化エージェントによりソース・データを読み、2 つのパーティション化エージェントそれぞれに約半分のデータが送られます。パーティション化エージェントはデータを分散し、各パーティションを 3 つのデータベース・パーティションのいずれかに送ります。各データベース・パーティションのロード・エージェントがデータをロードします。

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、適切なデータベース・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、*問題判別およびデータベース・パフォーマンスのチューニング*の『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。

始める前に

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 :

- 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、このステップは重要です。
- 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ので、ロード操作を開始する前に COMMIT または ROLLBACK ステートメントを発行して、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いてください。 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。
  1. **CLIENT** パラメーター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。
  2. 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。
- 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、「*問題判別*およびデータベース・パフォーマンスのチューニング」の『設計アドバイザー』を参照してください。

制約事項

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 パラメーター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 パラメーターを指定する際には、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 パラメーターを指定する際には、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。 **PARTITIONING_DBPARTNUMS** オプションによって指定されたデータベース・パーティションがオーバーラップしている場合、対象の表でロード・パーティション化サブエージェントがまだ実行されていないとロード操作で自動的に **PARTITIONING_DBPARTNUMS** パラメーターが選択され、使用可能なデータベース・パーティションがない場合には LOAD が失敗します。

  バージョン 9.7 フィックスパック 6 以降では、**PARTITIONING_DBPARTNUMS** オプションによって指定されたデータベース・パーティションがオーバーラップしている場合、表でロード・パーティション化サブエージェントがまだ実行されていないと、ロード・ユーティリティーは、**OUTPUT_DBPARTNUMS** で示されているデータベース・パーティションから自動的に **PARTITIONING_DBPARTNUMS** パラメーターの選択を試みます。使用可能なデータベース・パーティションがない場合には失敗します。

  PARTITIONING_DBPARTNUMS オプションを使用してパーティションを明示的に指定する予定の場合は、そのオプションをすべての並行 **LOAD** コマンドで使用し、各コマンドで別々のパーティションを指定することを強くお勧めします。並行ロード・コマンドの一部のみで **PARTITIONING_DBPARTNUMS** を指定した場合、または、オーバーラップするパーティションを指定した場合は、 **LOAD** コマンドは並行ロードの少なくとも一部で代替パーティション化ノードを選択しなければならず、その場合まれなケースとしてコマンドが失敗することがあります (SQL2038N)。
- 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。

例

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。

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 です。 **OUTPUT_DBPARTNUMS** パラメーターのデフォルトは、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)
```

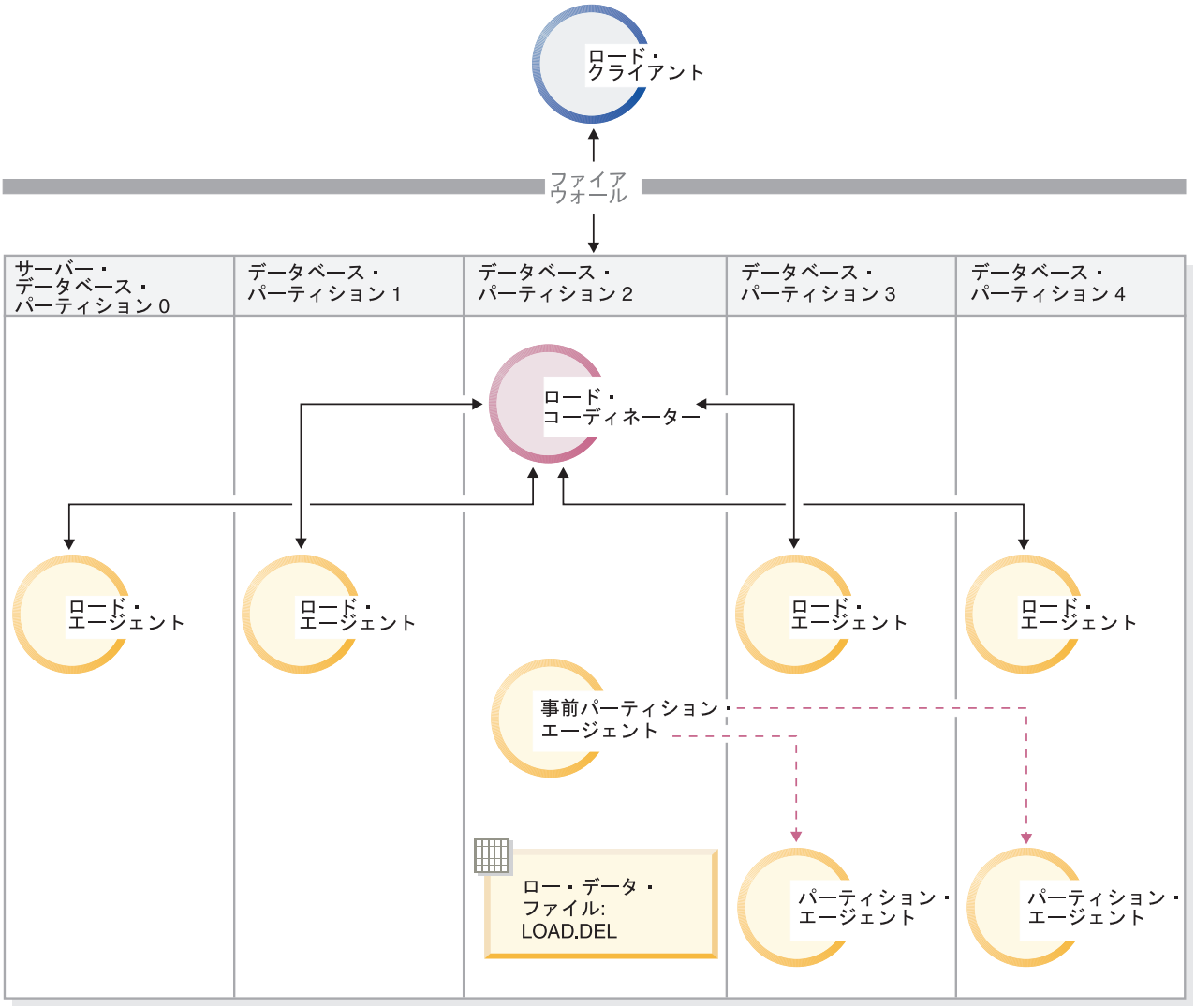


図 40. データベース・パーティション 3 および 4 へのデータのロード。 : この図では、前のコマンドが発行された結果の動作を説明します。データは、データベース・パーティション 3 および 4 にロードされます。

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)
```

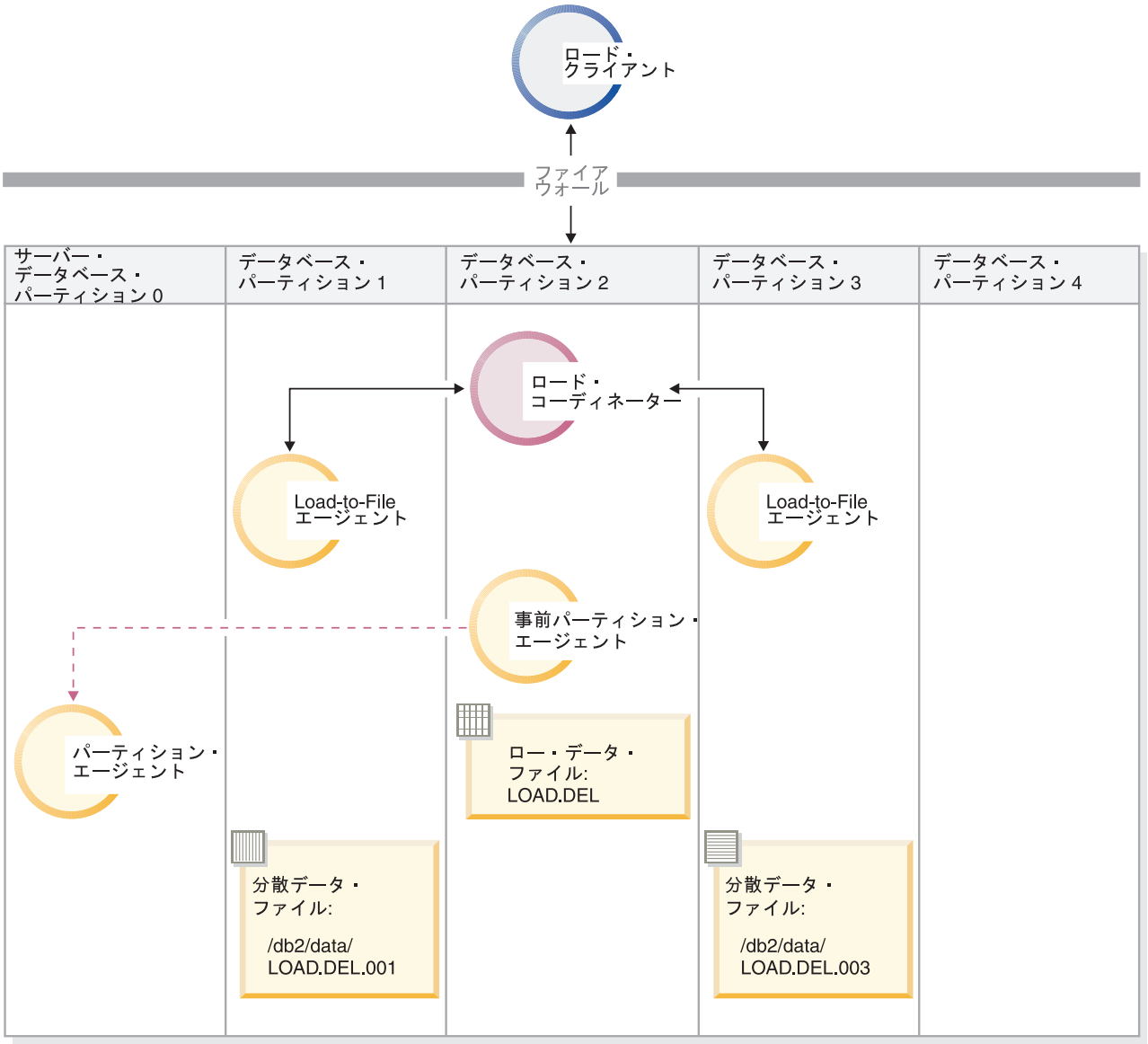


図 41. 1 つのパーティション化エージェントを使用したデータベース・パーティション 1 および 3 へのデータのロード。 : この図では、前のコマンドが発行された結果の動作を説明します。データは、データベース・パーティション 0 で実行する 1 パーティション化エージェントを使用して、データベース・パーティション 1 および 3 にロードされます。

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
```

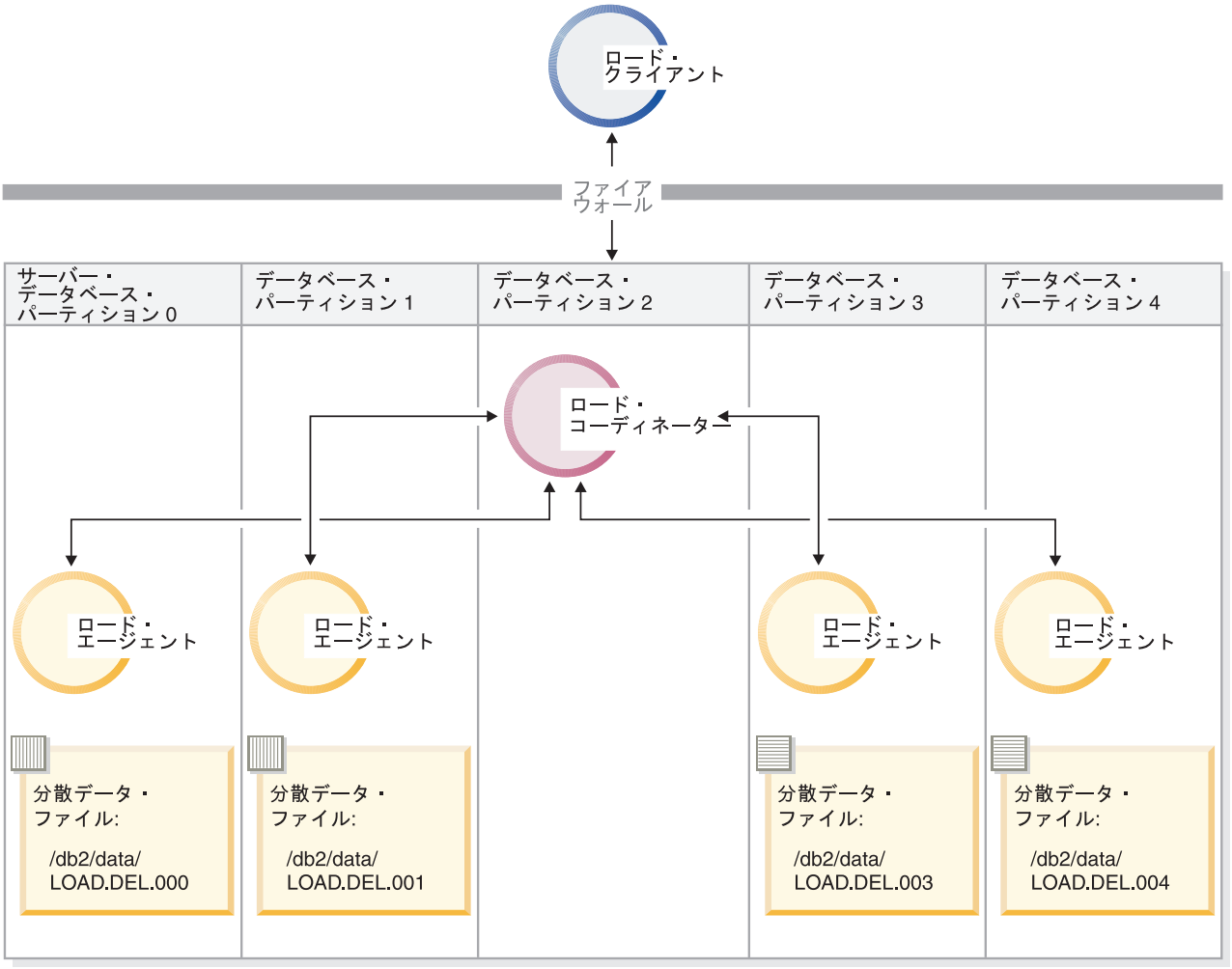


図 42. 特定の表が定義されているすべてのデータベース・パーティションへのデータのロード。 : この図では、前のコマンドが発行された結果の動作を説明します。分散データは、TABLE1 が定義されているすべてのデータベース・パーティションにロードされます。

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** パラメーターを指定する必要があります。このパラメーターは、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します。例えば、以下のようにします。

```
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** パラメーター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** パラメーター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次の表にリストします。

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。

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。 **LOAD** 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 (次の例を参照)。 **LOAD**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

パーティション・データベース環境でのロード操作を変更するために使用できる構成オプションが、多数存在します。

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** パラメーター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** パラメーター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** パラメーター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** パラメーター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 49152 から 65535 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 32 768 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歪度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 USE PROFILE** 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

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: DB2_PARTITIONEDLOAD_DEFAULT レジストリー変数は非推奨で、将来のリリースでは除去され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・サーバーにデータベース製品の最新のリリースをインストールし、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。

このタスクについて

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。

索引再作成の必要性の識別

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。

例

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
```

例 6: 表の REORG 情報には再編成操作の一部としてエクステントを再利用することに関する情報が含められています。 続く例は関係する出力を示しています。

```
db2 -v "get snapshot for tables on wsdb"

 Table Reorg Information:
   Reorg Type        =
        Reclaim Extents
        Allow Write Access
   Reorg Index       = 0
   Reorg Tablespace  = 0
   Start Time        = 10/22/2008 15:49:35.477532
   Reorg Phase       = 12 - Release
   Max Phase         = 3
```

注: SQLM_DBMON_VERSION9_7 以前のモニターのバージョンからのスナップショット要求はどれも要求クライアントに再利用の REORG 状況を戻しません。

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、スナップショット・モニターを使用して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、または DB2 pureScale 環境内のデータベース用のイベント・モニターの作成

一般的に、パーティション・データベース・システムまたは DB2 pureScale 環境のイベント・モニターは、単一メンバー・データベースで実行されるイベント・モニターと動作が似ています。ただし、それらの環境用のイベント・モニターを作成する際には、注意すべき相違点がいくつかあります。

手順

- モニター対象のパーティションを指定します。

```
CREATE EVENT MONITOR tabmon FOR TABLES
                    WRITE TO FILE '/tmp/tabevents'
                    ON PARTITION 3
```

  tabmon はイベント・モニターの名前を表します。

  /tmp/tab events 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。

- 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。 例えば、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
```

  注: 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。

- 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 TABLES
                    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。

- 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
```

タスクの結果

イベント・モニターが作成されてアクティブ化されると、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。例えば、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、ディスクに書き込ま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 43)。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。

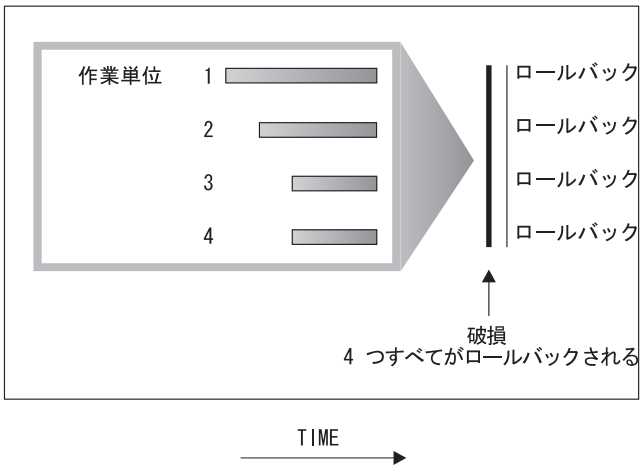


図 43. 作業単位のロールバック (クラッシュ・リカバリー)

IBM DB2 pureScale Feature を使用している場合は、メンバー・クラッシュ・リカバリー およびグループ・クラッシュ・リカバリー という、2 種類の固有のクラッシュ・リカバリー方法があることに注意してください。メンバー・クラッシュ・リカバリーとは、メンバーに障害が発生した場合に、単一のメンバーのログ・ストリームを使用して、データベースの一部をリカバリーするプロセスのことです。メンバー・クラッシュ・リカバリーは、通常、メンバー再始動の一環として自動的に開始されるオンライン操作です。つまり、この操作中、他のメンバーは、データベースにアクセスできます。複数のメンバーで、同時にメンバー・クラッシュ・リカバリーが行われる場合もあります。グループ・クラッシュ・リカバリーとは、障害によって、クラスター内に使用可能なクラスター・キャッシング・ファシリティーが存在しなくなった場合に、複数メンバーのログ・ストリームを使用して、データベースをリカバリーするプロセスのことです。グループ・クラッシュ・リカバリーも、通常は (グループ再始動の一環として) 自動的に開始されるものであり、 DB2 pureScale 環境以外の DB2 クラッシュ・リカバリー操作と同様に、このリカバリーの進行中は、データベースにアクセスできません。

データベースまたはデータベース・マネージャーに障害が発生すると、データベースは不整合な状態のままとなる可能性があります。障害発生時に未完了であったトランザクションによる変更が、データベースの内容に含まれている可能性があります。また、障害前に完了したトランザクションによって行われた変更で、まだディスクにフラッシュされていなかったものが、データベースから失われている可能性もあります。クラッシュ・リカバリー操作を実行して、部分的にしか完了しなかったトランザクションをロールバックし、メモリーにしか書き込まれていなかった完了トランザクションによる変更をディスクに書き込む必要があります。

クラッシュ・リカバリーが必要となる状況には、次のようなものがあります。

- 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、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 インスタンスの異常終了を引き起こす、重大なオペレーティング・システム・エラー。

データベース・マネージャーにクラッシュ・リカバリーを自動的に実行させるには、データベース構成パラメーター (**autorestart**) を ON に設定することによって、自動再始動を有効にします。 (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。

ロールフォワード・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 に設定されていない) データベースで、クラッシュ・リカバリーが発生し、そのクラッシュ・リカバリー中に個別の表スペースが原因のエラーが発生した場合、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・リカバリーは他の表スペースで続行します。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。この動作は、 DB2 pureScale 環境 には当てはまりません。メンバー・クラッシュ・リカバリーまたはグループ・クラッシュ・リカバリー中にエラーが発生した場合、そのクラッシュ・リカバリー操作は失敗します。

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。

データベース・リカバリーには、次の 2 つのタイプがあります。

- クラッシュ・リカバリーは、障害状態が修正された後に、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、トランザク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、最初にトランザク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。例えば、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がそうです。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。

1. SQLCA を調査することにより、障害を検出したパーティション・サーバーを見つけます。 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。)
2. 手順 1 で見つかったサーバーの管理通知ログで、障害が発生したサーバーのノード番号を調べます。

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

障害の原因となった問題を特定して訂正することによって、障害が発生したデータベース・パーティション・サーバーからリカバリーできます。

手順

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。

このタスクについて

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。BK*xy* 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。

手順

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
   ```

次のタスク

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

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行えます。このリカバリーについて、いくつかの異なる Tivoli Storage Manager (TSM) 環境における例で示します。

以下の例では、コンピューター 1 (`bar` という名前) において AIX オペレーティング・システムが実行されています。このマシンのユーザーは `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。このコンピューターでも AIX オペレーティング・システムが実行され、ユーザーは regress9 です。

例 1: TSM サーバーがパスワードを自動管理する (PASSWORDACCESS オプションを GENERATE に設定)

このノード間リカバリーの例では、ログ・アーカイブとバックアップが TSM サーバー上に保管され、PASSWORDACCESS=GENERATE オプションを使ってパスワードが管理されるような場合に、1 つのコンピューターから別のコンピューターにデータをリカバリーする目的で 2 つのコンピューターをセットアップする方法を示します。

注: データベース構成を更新した後に、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。

1. bar コンピューターのデータベースのログを TSM サーバーにアーカイブできるようにするには、以下のコマンドを使用して、zample データベースに関する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.
   ```

2. 以下のコマンドを使用して、すべてのユーザーとアプリケーションをデータベースから切断します。

   ```
   db2 force applications all
   ```

3. 以下のコマンドを使用して、データベースに接続しているアプリケーションが存在しないことを確認します。

   ```
   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. 以下のコマンドを使用して、TSM サーバー上にデータベースのバックアップを作成します。

   ```
   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 imagge is : 20090216151025
   ```

   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、 491 ページの『データのバックアップ』を参照してください。

5. 以下のコマンドを使用して、zample データベースに接続します。

   ```
   db2 connect to zample
   ```

6. データベースに関する新しいトランザクション・ログを生成します。そうするには、以下のコマンドを使って表を作成し、TSM サーバーにデータをロードします。

```
bar:/home/roecken> db2 load from mr of del modified by noheader replace
   into employee copy yes use tsm
```

この例では、表の名前は employee であり、区切り文字で区切られている mr という名前の ASCII ファイルからデータをロードします。**COPY** YES オプションを指定することにより、ロードされるデータのコピーを作成します。**USE TSM** オプションは、TSM サーバーにデータのコピーを保管することを指定します。

注: COPY YES オプションを指定できるのは、データベースのロールフォワード・リカバリーが使用可能になっている場合だけです。つまり、**logarchmeth1** データベース構成パラメーターが USEREXIT、 LOGRETAIN、DISK、または TSM に設定されている必要があります。

進行状況を示すために、ロード・ユーティリティーは次のような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 "02/16/2009
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 "02/16/2009
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
```

7. データが表にロードされた後、zample データベースに対して以下のような照会を実行して、TSM サーバー上に 1 つのバックアップ・イメージ、1 つのロード・コピー・イメージ、および 1 つのログ・ファイルが存在することを確認します。

```
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: 20090216151025  Oldest log: S0000000.LOG  Log stream: 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: 20090216151213

Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
   Taken at: 2009-02-16-15.10.38
```

8. ノード間リカバリーを使用可能にするには、bar コンピューターに関連するオブジェクトへのアクセス権限を、別のコンピューターおよびアカウントに与える必要があります。この例では、以下のコマンドを使用して、コンピューター dps およびユーザー regress9 にアクセス権限を付与します。

```
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.
```

注: db2adutl 付与操作の結果を確認するには、以下のコマンドを発行して、現在のノードでの現在のアクセス権限リストを取得することができます。

```
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
```

9. この例では、コンピューター 2 (つまり dps) は zample データベースのノード間リカバリー用にまだセットアップされていません。以下のコマンドを使用して、このユーザーおよびコンピューターに関連付けられたデータが TSM サーバー上に存在しないことを確認します。

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
```

次の情報が戻されます。

```
  --- 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.
```

10. 以下のコマンドを使用して、ユーザー roecken およびコンピューター bar に関連付けられた zample データベースのオブジェクトのリストを TSM サーバーに照会します。

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename
     bar owner roecken
```

次の情報が戻されます。

```
--- Database directory is empty ---

Query for database ZAMPLE

Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 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: 20090216151213

Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
   Taken at: 2009-02-16-15.10.38
```

この情報は既に生成された TSM 情報と一致するため、このイメージを dps コンピューターにリストアできることが確認されます。

11. 以下のコマンドを使用して、zample データベースを TSM サーバーから 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。

12. 新しい表を作成して新しいデータをロードしたときに zample データベース・ログ・ファイルに記録されたトランザクションを適用するために、ロールフォワード操作を実行します。この例では、次のようなロールフォワード操作の試行は失敗します。ユーザーとコンピューターの情報が指定されないため、ロールフォワード・ユーティリティーはログ・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
```

次のようなエラーがコマンドによって戻されます。

```
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".
```

適切な **logarchopt** 値を使用して、別のコンピューターに関連付けられたログ・ファイルをロールフォワード・ユーティリティーに強制的に検索させます。この例では、次のようなコマンドを使って **logarchopt1** データベース構成パラメーターを設定し、ユーザー roecken およびコンピューター bar に関連付けられたログ・ファイルを検索します。

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
"'-fromnode=bar -fromowner=roecken'"
```

13. 以下のコマンドを使って **vendoropt** データベース構成パラメーターを設定することにより、ロールフォワード・ユーティリティーがバックアップおよびロード・コピー・イメージを使用できるようにします。

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
"'-fromnode=bar -fromowner=roecken'"
```

14. 以下のコマンドを使用して、zample データベース・ログ・ファイルに記録されたトランザクションを適用することにより、ノード間のデータ・リカバリーを完了できます。

```
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 members have returned status = 1


Member number  Rollforward  Next log to  Log files processed        Last committed transaction
               status       be read
-------------  -----------  -----------  -------------------------  ------------------------------
        0      not pending               S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC


DB20000I  The ROLLFORWARD command completed successfully.
```

ユーザー regress9 の下でのコンピューター dps 上のデータベース zample が、ユーザー roecken の下でのコンピューター bar 上のデータベースと同じポイントにリカバリーされました。

例 2: パスワードがユーザーによって管理される (PASSWORDACCESS オプションを PROMPT に設定)

このノード間リカバリーの例では、ログ・アーカイブとバックアップが TSM サーバー上に保管され、パスワードがユーザーによって管理されるような場合に、1 つのコンピューターから別のコンピューターにデータをリカバリーする目的で 2 つのコンピューターをセットアップする方法を示します。これらの環境では、余分の情報 (特に、オブジェクトの作成場所となったコンピューターの TSM ノード名とパスワード) が必要です。

1. コンピューター bar はソース・コンピューターの名前であるため、クライアント dsm.sys ファイルを更新して次のような行を追加します。

```
NODENAME bar
```

注: Windows オペレーティング・システムでは、このファイルの名前は dsm.opt です。このファイルを更新した場合、変更内容を有効にするにはシステムをリブートする必要があります。

2. 以下のコマンドを使用して、ユーザー roecken およびコンピューター bar に関連付けられたオブジェクトのリストを TSM サーバーに照会します。

```
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: 20090216151025  Oldest log: S0000000.LOG  Log stream: 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: 20090216151213

Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
   Taken at: 2009-02-16-15.10.38
```

3. zample データベースがコンピューター dps に存在しない場合は、以下の手順を実行します。
   a. 以下のコマンドを使用して、空の zample データベースを作成します。

      ```
      dps:/home/regress9> db2 create db zample
      ```

   b. 以下のコマンドを使用して、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
      ```

   c. 以下のコマンドを使用して、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. 以下のコマンドを使用して、zample データベースのリストアを試行します。

   ```
   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.
   ```

5. 以下のコマンドを使用して、ロールフォワード操作を実行します。

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   この例では、リストア操作によってデータベース構成ファイルが置き換えられたためにロールフォワード・ユーティリティーが適切なログ・ファイルを見つけられず、次のようなエラー・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".
   ```

   以下の TSM データベース構成値を適切な値に再設定します。
   a. 以下のコマンドを使用して、**tsm_nodename** 構成パラメーターを設定します。

      ```
      dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
      ```

   b. 以下のコマンドを使用して、**tsm_password** データベース構成パラメーターを設定します。

      ```
      dps:/home/regress9> db2 update db cfg for zample using tsm_password *******
      ```

c. ロールフォワード・ユーティリティーが適切なログ・ファイルを見つけられるようにするために、以下のコマンドを使って **logarchopt1** データベース構成パラメーターを設定します。

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
"'-fromnode=bar -fromowner=roecken'"
```

d. さらに、ロールフォワード操作中にロード・リカバリー・ファイルも使われるようにするために、以下のコマンドを使用して **vendoropt** データベース構成パラメーターを設定します。

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
"'-fromnode=bar -fromowner=roecken'"
```

6. 以下のコマンドを使ってロールフォワード操作を実行することにより、ノード間リカバリーを完了できます。

```
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 members have returned status = 1


Member number  Rollforward  Next log to  Log files processed        Last committed transaction
               status       be read
-------------  -----------  -----------  -------------------------  ------------------------------
        0      not pending               S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC


DB20000I  The ROLLFORWARD command completed successfully.
```

ユーザー regress9 の下でのコンピューター dps 上のデータベース zample が、ユーザー roecken の下でのコンピューター bar 上のデータベースと同じポイントにリカバリーされました。

例 3: クライアント・プロキシー・ノードを使用するよう TSM サーバーを構成する

このノード間リカバリーの例では、ログ・アーカイブとバックアップが TSM サーバー上に保管され、PASSWORDACCESS=GENERATE オプションを使ってパスワードが管理されるような場合に、1 つのコンピューターから別のコンピューターにデータをリカバリーする目的で 2 つのコンピューターをプロキシー・ノードとしてセットアップする方法を示します。

注: データベース構成を更新した後に、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。

この例では、コンピューター bar および dps がプロキシー名 clusternode の下で登録されます。これらのコンピューターはプロキシー・ノードとして既にセットアップされています。

1. 以下のコマンドを使用して、TSM サーバー上でコンピューター bar および dps をプロキシー・ノードとして登録します。

```
REGISTER NODE clusternode mypassword
GRANT PROXYNODE TARGET=clusternode AGENT=bar,dps
```

2. データベースのログを TSM サーバーにアーカイブできるようにするには、以下のコマンドを使用して、zample データベースに関する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 logarchopt1 "'-asnodename=clusternode'"
```

次の情報が戻されます。

```
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
```

3. 以下のコマンドを使用して、すべてのユーザーとアプリケーションをデータベースから切断します。

   ```
   db2 force applications all
   ```

4. 以下のコマンドを使用して、データベースに接続しているアプリケーションが存在しないことを確認します。

   ```
   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。

5. 以下のコマンドを使用して、TSM サーバー上にデータベースのバックアップを作成します。

   ```
   db2 backup db zample use tsm options "'-asnodename=clusternode'"
   ```

   次のような情報が戻されます。

   ```
   Backup successful. The timestamp for this backup image is : 20090216151025
   ```

   BACKUP DATABASE コマンドで **-asnodename** オプションを指定する代わりに、**vendoropt** データベース構成パラメーターを更新できます。

   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。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、 491 ページの『データのバックアップ』を参照してください。

6. 以下のコマンドを使用して、zample データベースに接続します。

   ```
   db2 connect to zample
   ```

7. データベースに関する新しいトランザクション・ログを生成します。そうするには、以下のコマンドを使って表を作成し、TSM サーバーにデータをロードします。

   ```
   bar:/home/roecken> db2 load from mr of del modified by noheader
    replace into employee copy yes use tsmwhere
   ```

   この例では、表の名前は employee であり、区切り文字で区切られている mr という名前の ASCII ファイルからデータをロードします。**COPY** YES オプションを指定することにより、ロードされるデータのコピーを作成します。**USE TSM** オプションは、TSM サーバーにデータのコピーを保管することを指定します。

   注: COPY YES オプションを指定できるのは、データベースのロールフォワード・リカバリーが使用可能になっている場合だけです。つまり、**logarchmeth1** データベース構成パラメーターが USEREXIT、 LOGRETAIN、DISK、または TSM に設定されている必要があります。

進行状況を示すために、ロード・ユーティリティーは次のような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 "02/16/2009
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 "02/16/2009
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
```

8. データが表にロードされた後、zample データベースに対して以下のような照会を実行して、TSM サーバー上に 1 つのバックアップ・イメージ、1 つのロード・コピー・イメージ、および 1 つのログ・ファイルが存在することを確認します。

```
bar:/home/roecken/sqllib/adsm> db2adutl query db zample
   options "-asnodename=clusternode"
```

次の情報が戻されます。

```
Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 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: 20090216151213

Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
   Taken at: 2009-02-16-15.10.38
```

9. この例では、コンピューター 2 (つまり dps) は zample データベースのノード間リカバリー用にまだセットアップされていません。以下のコマンドを使用して、このユーザーおよびコンピューターに関連付けられたデータが存在しないことを確認します。

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
```

次の情報が戻されます。

```
  --- 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.
```

10. 以下のコマンドを使用して、プロキシー・ノード clusternode に関連付けられた zample データベースのオブジェクトのリストを TSM サーバーに照会します。

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
  options="-asnodename=clusternode"
```

次の情報が戻されます。

```
--- Database directory is empty ---

Query for database ZAMPLE

Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 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: 20090216151213

Retrieving LOG ARCHIVE information.
   Log file: S0000000.LOG, Chain Num: 0, Log stream: 0,
   Taken at: 2009-02-16-15.10.38
```

この情報は既に生成された TSM 情報と一致するため、このイメージを dps コンピューターにリストアできることが確認されます。

11. 以下のコマンドを使用して、zample データベースを TSM サーバーから 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
   "'-asnodename=clusternode'" 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。

12. 新しい表を作成して新しいデータをロードしたときに zample データベース・ログ・ファイルに記録されたトランザクションを適用するために、ロールフォワード操作を実行します。この例では、次のようなロールフォワード操作の試

行は失敗します。ユーザーとコンピューターの情報が指定されないため、ロールフォワード・ユーティリティーはログ・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
```

次のようなエラーがコマンドによって戻されます。

```
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".
```

適切な **logarchopt** 値を使用して、別のコンピューター上のログ・ファイルをロールフォワード・ユーティリティーに強制的に検索させます。この例では、次のようなコマンドを使って **logarchopt1** データベース構成パラメーターを設定し、ユーザー roecken およびコンピューター bar に関連付けられたログ・ファイルを検索します。

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-asnodename=clusternode'"
```

13. 以下のコマンドを使って **vendoropt** データベース構成パラメーターを設定することにより、ロールフォワード・ユーティリティーがバックアップおよびロード・コピー・イメージを使用できるようにします。

    ```
    dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
       "'-asnodename=clusternode'"
    ```

14. 以下のコマンドを使用して、zample データベース・ログ・ファイルに記録されたトランザクションを適用することにより、ノード間のデータ・リカバリーを完了できます。

    ```
    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 members have returned status = 1


 Member number  Rollforward  Next log to  Log files processed        Last committed transaction
                status       be read
 -------------  -----------  -----------  -------------------------  ------------------------------
         0      not pending               S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC

DB20000I  The ROLLFORWARD command completed successfully.
```

ユーザー regress9 の下でのコンピューター dps 上のデータベース zample が、ユーザー roecken の下でのコンピューター bar 上のデータベースと同じポイントにリカバリーされました。

例 4: DB2 pureScale環境でクライアント・プロキシー・ノードを使用するよう TSM サーバーを構成する

この例では、ログ・アーカイブとバックアップが TSM サーバー上に保管され、PASSWORDACCESS=GENERATE オプションを使ってパスワードが管理されている場合に、一方のメンバーから、もう一方のメンバーにデータをリカバリーできるように、 2 つのメンバーをプロキシー・ノードとしてセットアップする方法を示します。

注: データベース構成を更新した後に、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。

この例では、メンバー member1 および member2 がプロキシー名 clusternode の下で登録されます。 DB2 pureScale環境では、任意のメンバーからバックアップまたはデータ・リカバリー操作を実行できます。この例では、member2 からデータをリカバリーします。

1. 以下のコマンドを使用して、TSM サーバー上でメンバー member1 および member2 をプロキシー・ノードとして登録します。

   ```
   REGISTER NODE clusternode mypassword
   GRANT PROXYNODE TARGET=clusternode AGENT=member1,member2
   ```

2. データベースのログを TSM サーバーにアーカイブできるようにするには、以下のコマンドを使用して、zample データベースに関するデータベース構成パラメーター **logarchmeth1** を更新します。

   ```
   member1:/home/roecken> db2 update db cfg for zample using
      LOGARCHMETH1 tsm logarchopt1 "'-asnodename=clusternode'"
   ```

   注: DB2 pureScale環境では、グローバル **logarchmeth1** データベース構成パラメーターを、任意のメンバーから一度設定するだけで済みます。

   次の情報が戻されます。

   ```
   DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
   ```

3. 以下のコマンドを使用して、すべてのユーザーとアプリケーションをデータベースから切断します。

   ```
   db2 force applications all
   ```

4. 以下のコマンドを使用して、データベースに接続しているアプリケーションが存在しないことを確認します。

   ```
   db2 list applications global
   ```

   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。

5. 以下のコマンドを使用して、TSM サーバー上にデータベースのバックアップを作成します。

   ```
   db2 backup db zample use tsm options '-asnodename=clusternode'
   ```

   次のような情報が戻されます。

   ```
   Backup successful. The timestamp for this backup image is : 20090216151025
   ```

   BACKUP DATABASE コマンドで **-asnodename** オプションを指定する代わりに、**vendoropt** データベース構成パラメーターを更新できます。

   注: DB2 pureScale環境では、このコマンドを任意のメンバーから実行して、データベースの全データをバックアップできます。

6. 以下のコマンドを使用して、zample データベースに接続します。

   ```
   db2 connect to zample
   ```

7. データベースに関する新しいトランザクション・ログを生成します。そうするには、以下のコマンドを使って表を作成し、TSM サーバーにデータをロードします。

   ```
   member1:/home/roecken> db2 load from mr of del modified by noheader replace
         into employee copy yes use tsmwhere
   ```

この例では、表の名前は employee であり、区切り文字で区切られている mr という名前の ASCII ファイルからデータをロードします。**COPY** YES オプションを指定することにより、ロードされるデータのコピーを作成します。**USE TSM** オプションは、TSM サーバーにデータのコピーを保管することを指定します。

注: COPY YES オプションを指定できるのは、データベースのロールフォワード・リカバリーが使用可能になっている場合だけです。つまり、**logarchmeth1** データベース構成パラメーターが USEREXIT、 LOGRETAIN、DISK、または TSM に設定されている必要があります。

進行状況を示すために、ロード・ユーティリティーは次のような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 "02/16/2009
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 "02/16/2009
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
```

8. データが表にロードされた後、zample データベースに対して以下のような照会を実行して、TSM サーバー上に 1 つのバックアップ・イメージ、1 つのロード・コピー・イメージ、および 1 つのログ・ファイルが存在することを確認します。

```
member1:/home/roecken/sqllib/adsm> db2adutl query db zample
   options "-asnodename=clusternode"
```

次の情報が戻されます。

```
Retrieving FULL DATABASE BACKUP information.
    1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 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: 20090216151213

Retrieving LOG ARCHIVE information.

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.01.10

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.01.11

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.01.19

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.02.49

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.02.49

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.02.49

   Log file: S0000002.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.03.15

   Log file: S0000002.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.03.15

   Log file: S0000002.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.03.16
```

9. 以下のコマンドを使用して、プロキシー・ノード clusternode に関連付けられた zample データベースのオブジェクトのリストを TSM サーバーに照会します。

```
member2:/home/regress9/sqllib/adsm> db2adutl query db zample
   options="-asnodename=clusternode"
```

次の情報が戻されます。

```
   --- Database directory is empty ---

   Query for database ZAMPLE

   Retrieving FULL DATABASE BACKUP information.
       1 Time: 20090216151025  Oldest log: S0000000.LOG  Log stream: 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: 20090216151213

Retrieving LOG ARCHIVE information.

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.01.10

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.01.11

   Log file: S0000000.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.01.19

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.02.49

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.02.49

   Log file: S0000001.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.02.49

   Log file: S0000002.LOG, Chain Num: 1, Log stream: 1, Taken at: 2009-02-16-13.03.15
```

```
Log file: S0000002.LOG, Chain Num: 1, Log stream: 2, Taken at: 2009-02-16-13.03.15

Log file: S0000002.LOG, Chain Num: 1, Log stream: 0, Taken at: 2009-02-16-13.03.16
```

この情報は既に生成された TSM 情報と一致するため、このイメージを member2 メンバーにリストアできることが確認されます。

10. 以下のコマンドを使用して、TSM サーバー上の zample データベースを member2 メンバーからリストアします。

```
member2:/home/regress9> db2 restore db zample use tsm options
   '-asnodename=clusternode' without prompting
```

次の情報が戻されます。

```
DB20000I  The RESTORE DATABASE command completed successfully.
```

注: member2 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。

11. 以下のコマンドを使って **vendoropt** データベース構成パラメーターを設定することにより、ロールフォワード・ユーティリティーがバックアップおよびロード・コピー・イメージを使用できるようにします。

```
member2:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-asnodename=clusternode'"
```

注: DB2 pureScale環境では、グローバル **vendoropt** データベース構成パラメーターを、任意のメンバーから一度設定するだけで済みます。

12. 以下のコマンドを使用して、zample データベース・ログ・ファイルに記録されたトランザクションを適用することにより、メンバー間のデータ・リカバリーを完了できます。

```
member2:/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 members have returned status = 3


 Member number  Rollforward  Next log to  Log files processed        Last committed transaction
                status       be read
 -------------  -----------  -----------  -------------------------  ------------------------------
             0  not pending               S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC
             1  not pending               S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC
             2  not pending               S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC

DB20000I  The ROLLFORWARD command completed successfully.
```

ユーザー regress9 の下でのメンバー member2 上のデータベース zample が、ユーザー roecken の下でのメンバー member1 上のデータベースと同じポイントにリカバリーされました。

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、および SQLOGCTL.LFH ファイルに格納されている仮想タイム・スタンプを使用します。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

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。

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。

Linux および UNIX オペレーティング・システムでは、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。

また、Linux および UNIX オペレーティング・システムでは、**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。 **DB2RSHCMD** が設定されない場合、rsh (HP-UX では remsh) が使用されます。 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。

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データベース・パーティション・サーバーで **db2trc** を実行する場合は、**rah** を使用してください。

注: **db2_all** コマンドは、対話式ユーザー入力が必要なコマンドをサポートしていません。

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

マルチディメンション・クラスタリングとデータ・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マルチディメンション・クラスタリング (MDC) キーの両方で使用されます。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表に行うと、どちらかの機能を単独で使用するよりも、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 データウェアハウスの特性

以下の推奨事項は、DB2 V9.1 にとっては新しく典型的で主流であった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します。

- 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。
- パーティション・データベース環境が使用される (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単位にロードされる。
- 毎日の INGEST 量は、1 万から 1000 万以内のレコードである。
- データ量が変化する。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 (例えば 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 15. 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。
データの日次ロード (**LOAD** コマンドまたは **INGEST** コマンド)	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 15. 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 を指定した **LOAD** コマンド、または **INGEST** コマンド)	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、表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、 38 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);
```

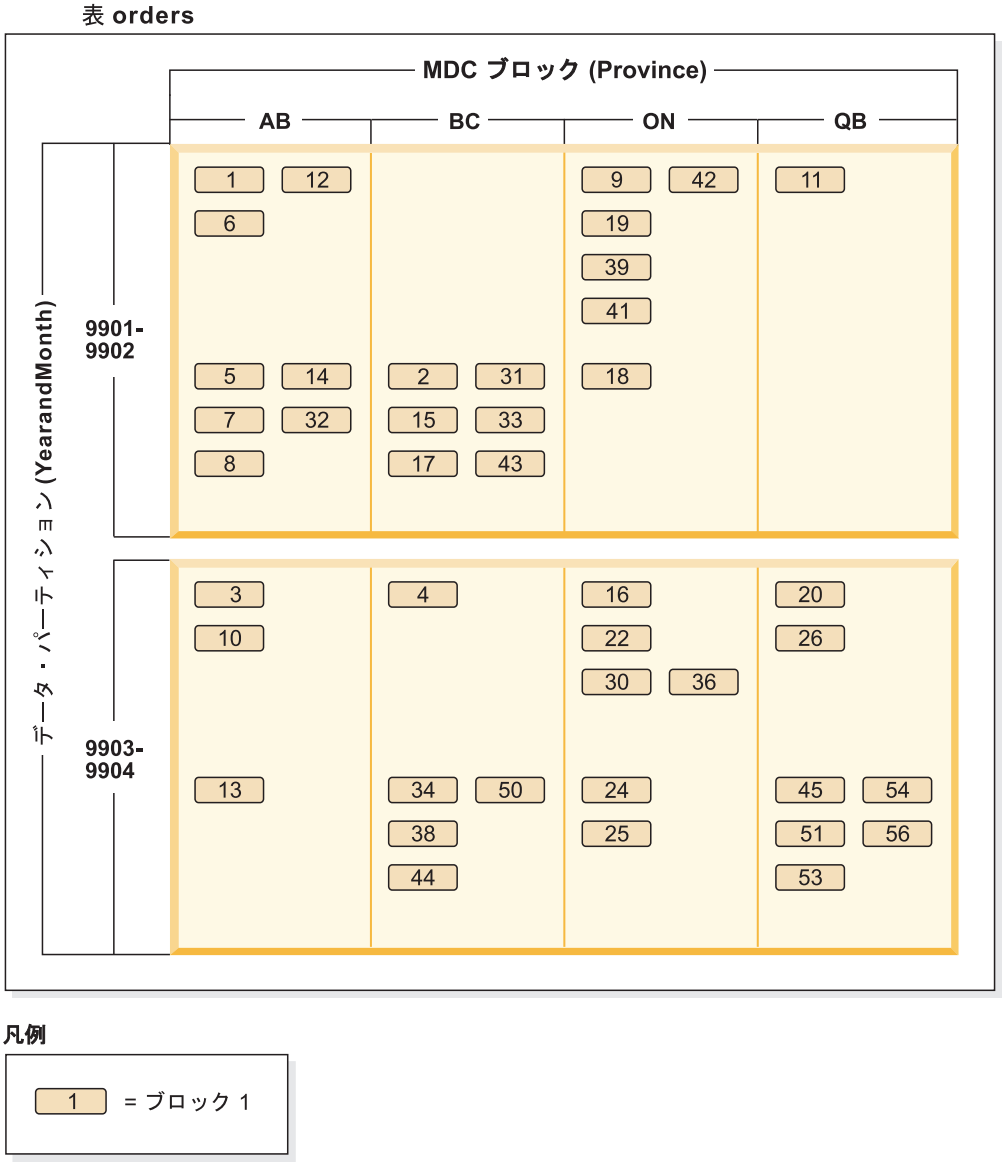


図 44. YearAndMonth によってパーティション化され、Province によって編成される表

例 2:

39 ページの図 7 に示されているように、YearAndMonth を ORGANIZE BY DIMENSIONS 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);
```

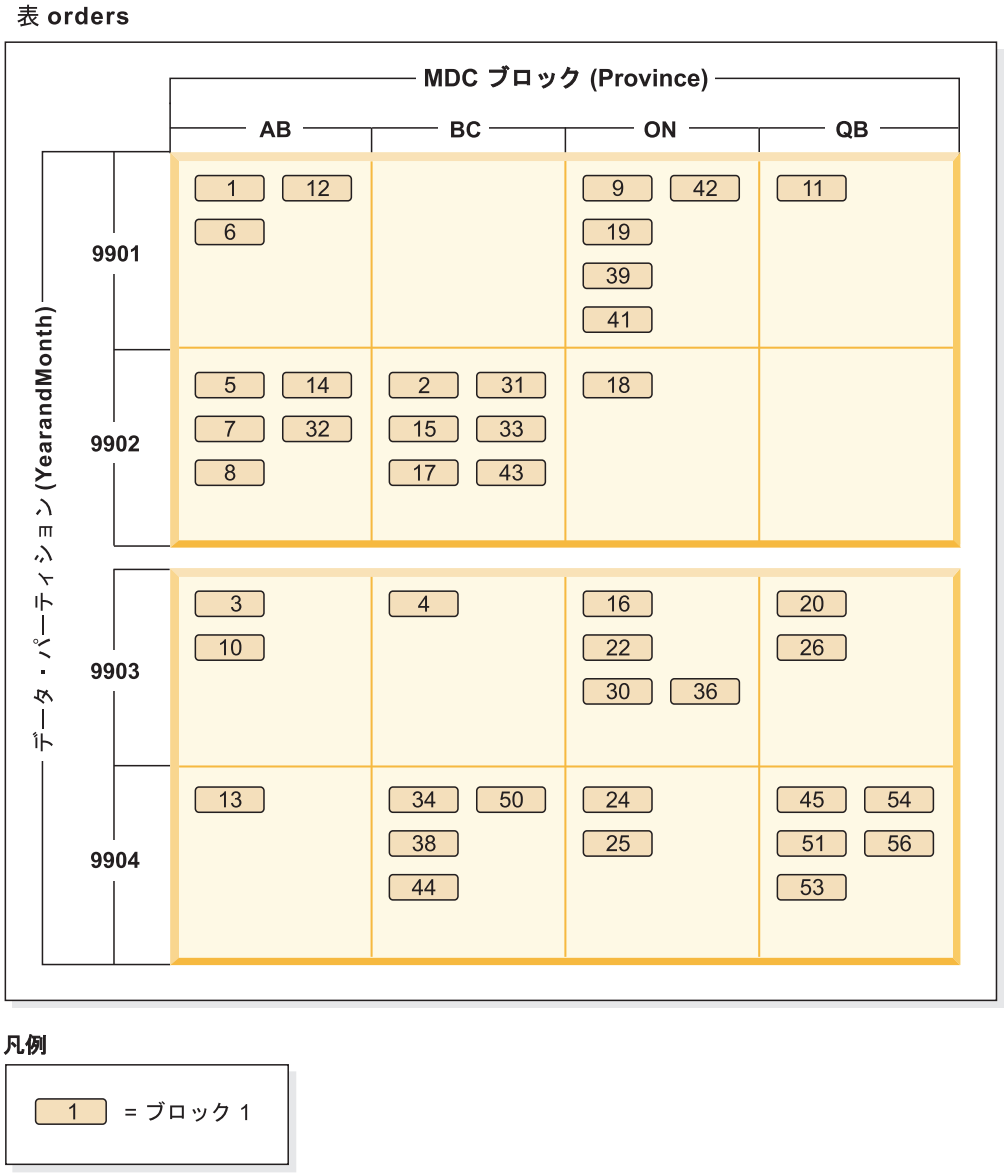


図 45. YearAndMonth によってパーティション化され、Province および YearAndMonth によって編成される表

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。
- DB2 V9.7 フィックスパック 1 以降のリリースで作成したデータ・パーティション化 MDC 表では、MDC ブロック索引はパーティション化されます。 DB2

V9.7 以前のリリースで作成したデータ・パーティション化 MDC 表では、MDC ブロック索引はパーティション化されません。

パーティション表の最適化ストラテジー

データ・パーティションの除去とは、照会述部に基づき、照会に応答するためにアクセスする必要があるのは表のデータ・パーティションのサブセットのみであると判断するデータベース・サーバーの機能のことです。データ・パーティションの除去は、パーティション表に対して意思決定支援照会を実行する際に特に役立ちます。

パーティション表は、データ・パーティションまたは範囲と呼ばれる複数のストレージ・オブジェクトに表データを分割するというデータ編成スキームを使用します。分割は、表の 1 つ以上の表パーティション・キー列の値に従って行われます。表のデータは、CREATE TABLE ステートメントの PARTITION BY 節で提供された仕様に基づいて、複数のストレージ・オブジェクトにパーティション化されます。このストレージ・オブジェクトは異なる表スペース、同じ表スペース内、またはその両方に配置することができます。

以下の例は、データ・パーティションの除去のパフォーマンス上の利点について示しています。

```
create table custlist(
  subsdate date, province char(2), accountid int)
  partition by range(subsdate) (
    starting from '1/1/1990' in ts1,
    starting from '1/1/1991' in ts1,
    starting from '1/1/1992' in ts1,
    starting from '1/1/1993' in ts2,
    starting from '1/1/1994' in ts2,
    starting from '1/1/1995' in ts2,
    starting from '1/1/1996' in ts3,
    starting from '1/1/1997' in ts3,
    starting from '1/1/1998' in ts3,
    starting from '1/1/1999' in ts4,
    starting from '1/1/2000' in ts4,
    starting from '1/1/2001'
    ending '12/31/2001' in ts4)
```

2000 年の顧客情報にのみ関心があるとします。

```
select * from custlist
  where subsdate between '1/1/2000' and '12/31/2000'
```

349 ページの図 46 に示されているように、データベース・サーバーはこの照会を解決するために、表スペース TS4 の 1 つのデータ・パーティションのみにアクセスする必要があると判断します。

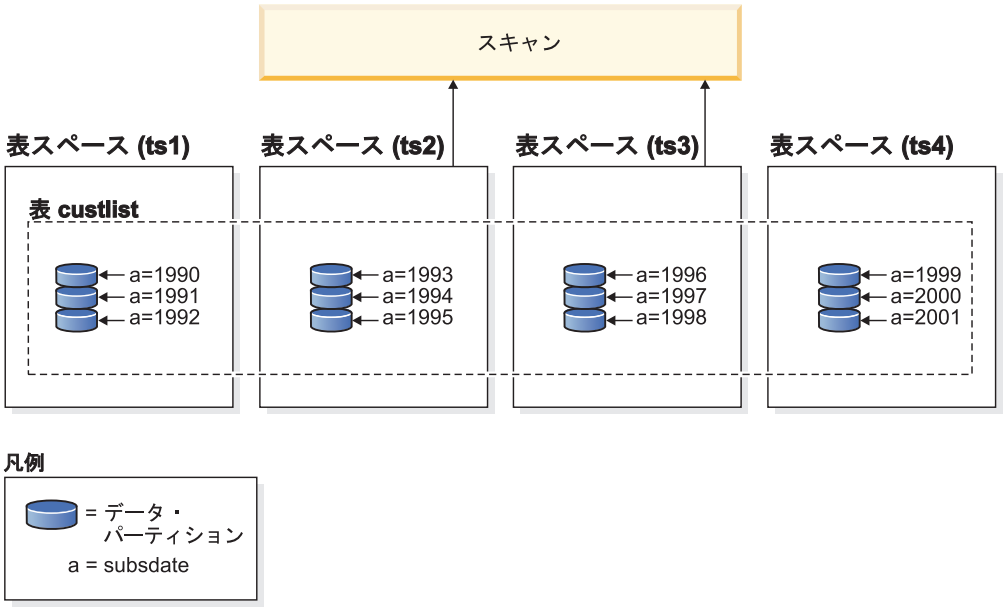


図 46. データ・パーティションの除去のパフォーマンス上の利点

データ・パーティションの除去の別の例は、以下のスキームに基づいています。

```
create table multi (
  sale_date date, region char(2))
  partition by (sale_date) (
    starting '01/01/2005'
    ending '12/31/2005'
    every 1 month)

create index sx on multi(sale_date)

create index rx on multi(region)
```

以下の照会を発行するとします。

```
select * from multi
  where sale_date between '6/1/2005'
    and '7/31/2005' and region = 'NW'
```

表のパーティション化を使用しないで考えられる 1 つのプランは、索引 ANDing です。索引 ANDing は、以下のタスクを実行します。

- 各索引から関連するすべての索引項目を読み取る
- 行 ID (RID) の両方のセットを保管する
- RID を突き合わせて、両方の索引のどちらで生じたかを判別する
- RID を使用して行を取り出す

350 ページの図 47 で示されているように、表のパーティション化を使用すると、REGION と SALE_DATE の両方での一致を検出するために索引が読み取られ、それにより一致する行を素早く取得することができます。

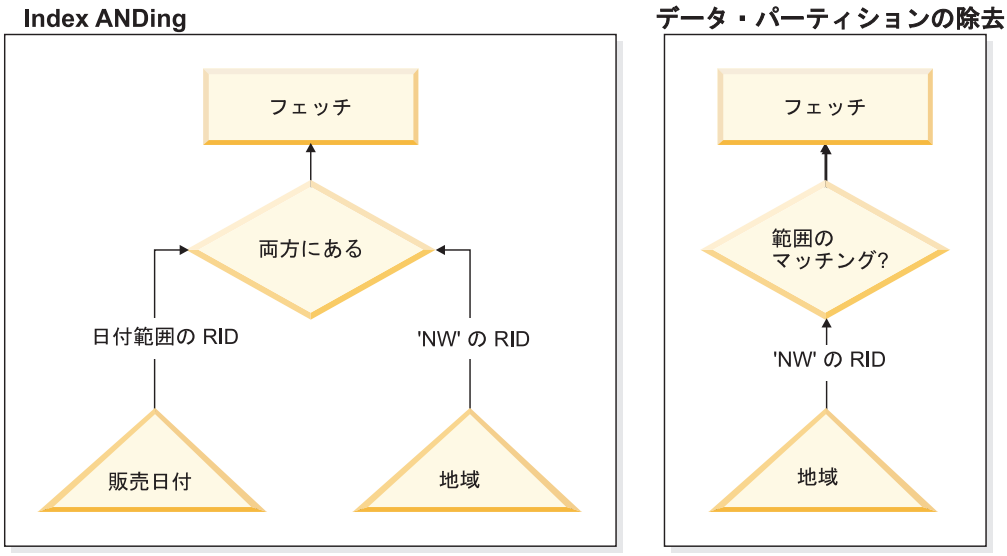


図 47. 表のパーティション化と索引 ANDing の両方のオプティマイザーの決定パス

DB2 Explain

さらに、Explain 機能を使用して、照会オプティマイザーによって選択されたデータ・パーティションの除去プランを判別できます。 『DP Elim Predicates』情報には、以下の照会を解決するためにどのデータ・パーティションがスキャンされるかが示されます。

```
   select * from custlist
     where subsdate between '12/31/1999' and '1/1/2001'

Arguments:
---------
DPESTFLG: (Number of data partitions accessed are Estimated)
    FALSE
DPLSTPRT: (List of data partitions accessed)
    9-11
DPNUMPRT: (Number of data partitions accessed)
   3

DP Elim Predicates:
------------------
Range 1)
  Stop  Predicate: (Q1.A <= '01/01/2001')
 Start Predicate: ('12/31/1999' <= Q1.A)



Objects Used in Access Plan:
 --------------------------

  Schema: MRSRINI
  Name:    CUSTLIST
  Type:    Data Partitioned Table
   Time of creation:      2005-11-30-14.21.33.857039
   Last statistics update:    2005-11-30-14.21.34.339392
    Number of columns:      3
    Number of rows:      100000
   Width of rows:      19
   Number of buffer pool pages:    1200
    Number of data partitions:    12
    Distinct row values:      No
    Tablespace name:      <VARIOUS>
```

複数列のサポート

データ・パーティションの除去は、表パーティション・キーとして複数列が使用されている場合に向いています。例えば、

```
create table sales (
     year int, month int)
    partition by range(year, month) (
       starting from (2001,1)
       ending at (2001,3) in ts1,
      ending at (2001,6) in ts2,
      ending at (2001,9) in ts3,
      ending at (2001,12) in ts4,
      ending at (2002,3) in ts5,
      ending at (2002,6) in ts6,
      ending at (2002,9) in ts7,
      ending at (2002,12) in ts8)

   select * from sales where year = 2001 and month < 8
```

照会オプティマイザーは、この照会を解決するために TS1、TS2 および TS3 内のデータ・パーティションのみにアクセスする必要があると推論します。

注: 表パーティション・キーが複数列から構成されている場合、複合キーの先頭列に述部がある場合に限ってデータ・パーティションの除去が可能です。表パーティション・キーとして使用される非先頭列は独立していないからです。

複数範囲のサポート

複数範囲 (OR で結ばれたもの) を持つデータ・パーティションでデータ・パーティションの除去を行うことが可能です。前述の例で作成された SALES 表を使用して、以下の照会を実行します。

```
   select * from sales
     where (year = 2001 and month <= 3)
       or (year = 2002 and month >= 10)
```

データベース・サーバーは、2001 年の最初の四半期と 2002 年の最後の四半期のデータのみにアクセスします。

生成列

生成列を表パーティション・キーとして使用できます。例えば、以下のようにします。

```
   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。例えば、

```
select * from sales where a > 35
```

データベース・サーバーは、a (a > 35) から b (b > 7) に追加の述部を生成し、データ・パーティションの除去が可能になります。

結合述部

結合述部が表アクセス・レベルに下がると、結合述部をデータ・パーティションの除去に使用することもできます。結合述部は、ネストされたループ結合 (NLJN) の内部結合の表アクセス・レベルにのみ下がります。

次の表を考慮してください。

```
create table t1 (a int, b int)
  partition by range(a,b) (
    starting from (1,1)
    ending (1,10) in ts1,
    ending (1,20) in ts2,
    ending (2,10) in ts3,
    ending (2,20) in ts4,
    ending (3,10) in ts5,
    ending (3,20) in ts6,
    ending (4,10) in ts7,
    ending (4,20) in ts8)

create table t2 (a int, b int)
```

次の 2 つの述部を使用できます。

```
P1: T1.A = T2.A
P2: T1.B > 15
```

この例では、結合の外部値が不明なため、コンパイル時にアクセスされるデータ・パーティションを正確に判別することはできません。この場合、ホスト変数またはパラメーター・マーカーが使用される場合と同様に、必要な値が結合される実行時にデータ・パーティションの除去が生じます。

実行時に T1 が NLJN の内部にある場合、述部に基づいて、T2.A のすべての外部値に対してデータ・パーティションの除去が動的に生じます。実行時に、述部 T1.A = 3 および T1.B > 15 が外部値 T2.A = 3 に対して適用され、表スペース TS6 のデータ・パーティションにアクセスする資格を与えます。

表 T1 と T2 の列 A に以下の値があるとします。

外部表 T2: 列 A	内部表 T1: 列 A	内部表 T1: 列 B	内部表 T1: データ・パーティション・ロケーション
2	3	20	TS6
3	2	10	TS3
3	2	18	TS4
	3	15	TS6
	1	40	TS3

(内部表に対する表スキャンを前提として) ネスト・ループ結合を実行するために、データベース・マネージャーは以下のステップを実行します。

1. T2 から最初の行を読み取ります。A の値は 2 です。
2. T2.A の値 (2) を結合述部 T1.A = T2.A で列 T2.A にバインドします。述部は T1.A = 2 となります。

3. 述部 T1.A = 2 および T1.B > 15 を使用して、データ・パーティションの除去を適用します。これにより、表スペース TS4 のデータ・パーティションが資格を得ます。
4. T1.A = 2 および T1.B > 15 の適用後に行が検出されるまで、表 T1 の表スペース TS4 のデータ・パーティションをスキャンします。資格にかなう最初の検出行は、T1 の行 3 です。
5. 一致した行を結合します。
6. 次の一致 (T1.A = 2 および T1.B > 15) が検出されるまで、表 T1 の表スペース TS4 のデータ・パーティションをスキャンします。それ以上、行は見つかりません。
7. T2 のすべての行が処理されるまで、T2 の次の行 (A の値を 3 に置換する) に対してステップ 1 から 6 までを繰り返します。

XML データの索引

DB2 バージョン 9.7 フィックスパック 1 以降では、パーティション表の XML データに対する索引を、パーティション索引と非パーティション索引のどちらとしても作成できます。デフォルトは、パーティション索引です。

パーティションおよび非パーティション XML 索引は、表の挿入、更新、および削除の各操作時にデータベース・マネージャーによって保守され、その方法はパーティション表の他のリレーショナル索引の保守方法と同じです。パーティション表における XML データの非パーティション索引は、非パーティション表の XML データの索引と同様の仕方で照会処理の速度を速めるために使用されます。照会述部を使用すると、照会に応答するために、パーティション表のデータ・パーティションのサブセットのみにアクセスする必要があることを判別できます。

XML 列のデータ・パーティションの除去および索引は、連携して照会パフォーマンスを向上させることができます。以下のパーティション表について考慮してください。

```
create table employee (a int, b xml, c xml)
  index in tbspx
  partition by (a) (
    starting 0 ending 10,
    ending 20,
    ending 30,
    ending 40)
```

次に以下の照会について考慮します。

```
select * from employee
  where a > 21
  and xmlexist('$doc/Person/Name/First[.="Eric"]'
    passing "EMPLOYEE"."B" as "doc")
```

オプティマイザーは、述部 a > 21 に基づいて最初の 2 つのパーティションを直ちに除去できます。照会プランでオプティマイザーが列 B の XML データの非パーティション索引を選択すると、XML データの索引を使用する索引スキャンでは、オプティマイザーが実行したデータ・パーティションの除去を活用でき、リレーショナル・データ・パーティションの除去述部によって除去されなかったパーティションに属する結果だけを戻すことができます。

MDC 表の最適化ストラテジー

マルチディメンション・クラスタリング (MDC) 表を作成した場合、オプティマイザーは追加の最適化ストラテジーを適用できるので、多くの照会のパフォーマンスが向上する可能性があります。これらのストラテジーは主にブロック索引の効率が改善されたことに基づいていますが、複数ディメンションでのクラスタリングによる利点もデータ検索の高速化を可能にしています。

MDC 表の最適化ストラテジーは、パーティション内並列処理およびパーティション間並列処理によるパフォーマンス上の利点も活用することができます。 MDC 表による以下の特定の利点を検討してください。

- ディメンション・ブロック索引の参照数により、表の必要な部分を識別して必要なブロックだけを高速にスキャンすることができます。
- ブロック索引はレコード ID (RID) 索引よりも小さいので、参照はより高速になります。
- 索引の ANDing 操作および ORing 操作をブロック・レベルで実行して、 RID と結合することができます。
- データはエクステント上でクラスター化されることが保証されているので、検索がより高速になります。
- ロールアウトを使用できるときに行の削除が高速になります。

SALES という名前で、ディメンションが REGION および MONTH 列に定義されている MDC 表についての次の簡単な例を検討してください。

```
select * from sales
  where month = 'March' and region = 'SE'
```

この照会では、オプティマイザーはディメンション・ブロック索引のルックアップを実行して、 month が March で region が SE のブロックを検索することができます。その後、こうしたブロックのみをスキャンして、結果セットを迅速にフェッチできます。

ロールアウト削除

ロールアウトを使用する削除が許可される条件になると、MDC 表から行を削除するための、さらに効果的なこの方法が使用されます。必要な条件は以下のとおりです。

- DELETE ステートメントは検索条件付き DELETE であって、位置指定 DELETE ではない (このステートメントは WHERE CURRENT OF 節を使用しない)。
- WHERE 節がない (すべての行が削除される) または WHERE 節の条件だけがディメンションに適用される。
- 表が DATA CAPTURE CHANGES 節で定義されていない。
- 表が参照整合性リレーションシップで親でない。
- 表に ON DELETE トリガーが定義されていない。
- 表が即時に更新される MQT で使用されない。
- カスケード削除操作がロールアウトに適格になる可能性がある (その外部キーがその表のディメンション列のサブセットである場合)。

- DELETE ステートメントは、(CREATE TRIGGER ステートメント上の OLD TABLE AS 節により指定される) トリガー SQL 操作の前に、影響を受ける一連の行を識別する一時表に対して実行される SELECT ステートメントには入れることができません。

ロールアウト削除の際、削除レコードはログに記録されません。その代わり、レコードの入ったページはページのパーツを再フォーマットすることによって空にされます。再フォーマットされたパーツに対する変更はログに記録されますが、レコード自体はログに記録されません。

デフォルトの動作である*即時クリーンアップ・ロールアウト* は、削除時に RID 索引をクリーンアップするためのものです。このモードは、**DB2_MDC_ROLLOUT** レジストリー変数を IMMEDIATE に設定するか、または IMMEDIATE を SET CURRENT MDC ROLLOUT MODE ステートメントで指定することによって、指定することもできます。標準の削除操作と比較して、索引の更新のロギングに変更がない場合、パフォーマンスの向上は RID 索引の数によって異なります。 RID 索引が少ないと、合計時間およびログ・スペースの割合が改善されます。

節約されるログ・スペースの見積もりは、以下の公式によって作成できます。

```
S + 38*N - 50*P
```

ここで、*N* は削除されるレコードの数であり、*S* は削除されるレコードの合計サイズ (NULL 標識および VARCHAR の長さなどのオーバーヘッドを含む) であり、*P* は削除されるレコードの入ったブロックのページ数です。この数値は、実際のログ・データを縮約したものです。アクティブ・ログ・スペースで必要な節約量はその値の倍です。ロールバック用に予約されたスペースの節約量があるためです。

代替方法として、*据え置きクリーンアップ・ロールアウト* を使用して、トランザクション・コミット後に RID 索引を更新できます。このモードは、**DB2_MDC_ROLLOUT** レジストリー変数を DEFER に設定するか、または DEFERRED を SET CURRENT MDC ROLLOUT MODE ステートメントで指定することでも指定できます。据え置きロールアウトで、RID 索引は削除コミット後に、バックグラウンドで非同期にクリーンアップされます。このロールアウトのメソッドは、非常に大規模な削除の場合、または表に多数の RID 索引が存在する場合は結果としてかなり高速な削除となります。即時索引クリーンアップでは索引内の各行は 1 つずつクリーンアップされるのに対し、据え置き索引クリーンアップ中に索引は並列でクリーンアップされるので、クリーンアップ操作全体の速度は向上します。さらに、非同期索引クリーンアップでは、索引キーではなく索引ページにより索引更新をログ記録するので、DELETE ステートメントのトランザクション・ログスペース所要量は大幅に削減されます。

注: 据え置きクリーンアップ・ロールアウトには、データベース・ヒープから取られる追加のメモリー・リソースが必要です。データベース・マネージャーが必要とするメモリー構造を割り振れない場合、据え置きクリーンアップ・ロールアウトは失敗し、メッセージが管理通知ログに書き込まれます。

据え置きクリーンアップ・ロールアウトを使用する状況

削除のパフォーマンスが最も重要な要因であり、RID 索引が表に定義されている場合は、据え置きクリーンアップ・ロールアウトを使用してください。索引クリーン

アップの前に、ロールアウトされたブロックの索引ベースのスキャンは、ロールアウトされたデータの量に応じてパフォーマンスがやや低下します。即時索引クリーンアップと据え置き索引クリーンアップとを決定するときは、以下の問題についても考慮してください。

- 削除操作のサイズ

  非常に大規模な削除に対しては、据え置きクリーンアップ・ロールアウトを選択します。ディメンション DELETE ステートメントが数多くの小さな MDC 表に対して発行される場合は、非同期に索引オブジェクトをクリーンアップするためのオーバーヘッドが、削除操作中の時間の節約という利点を上回ってしまう可能性があります。

- 索引の数およびタイプ

  表に行レベルの処理を必要とする数多くの RID 索引が含まれている場合は、据え置きクリーンアップ・ロールアウトを使用します。

- ブロック可用性

  DELETE ステートメントのコミット直後に、その削除操作により解放されるブロック・スペースを使用できるようにするには、即時クリーンアップ・ロールアウトを使用します。

- ログ・スペース

  ログ・スペースが制限されている場合、大規模削除の据え置きクリーンアップ・ロールアウトを使用します。

- メモリー制約

  据え置きクリーンアップ・ロールアウトは、据え置きクリーンアップが保留中のすべての表の追加のデータベース・ヒープ・スペースを消費します。

削除でロールアウト動作を無効にするには、**DB2_MDC_ROLLOUT** レジストリー変数を OFF に設定するか、または NONE を SET CURRENT MDC ROLLOUT MODE ステートメントに指定します。

注: DB2 バージョン 9.7 以降のリリースでは、パーティション化された RID 索引を含むデータ・パーティション MDC 表において据え置きクリーンアップ・ロールアウトはサポートされません。サポートされるのは、NONE モードと IMMEDIATE モードだけです。 **DB2_MDC_ROLLOUT** レジストリー変数が DEFER に設定されている場合、または **DB2_MDC_ROLLOUT** の設定をオーバーライドするために CURRENT MDC ROLLOUT MODE 特殊レジスターが DEFERRED に設定されている場合、クリーンアップ・ロールアウト・タイプは IMMEDIATE になります。

MDC 表に非パーティション RID 索引のみが存在する場合は、据え置き索引クリーンアップ・ロールアウトがサポートされます。

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

パーティション表の索引は、非パーティション表の索引と同様に機能します。ただし、パーティション表の索引は、索引をパーティション化するかどうかによって、異なるストレージ・モデルを使用して格納されます。

通常の非パーティション表の索引はすべて、共有索引オブジェクト内にありますが、パーティション表の*非パーティション索引* は、データ・パーティションが複数の表スペースにまたがっている場合であっても、単一の表スペースの独自の索引オブジェクト内に作成されます。データベース管理スペース (DMS) とシステム管理スペース (SMS) 表スペースは両方とも、表データとは別の場所にある索引の使用をサポートしています。各非パーティション索引は、LARGE 表スペースを含めて独自の表スペースに配置できます。各索引表スペースは、データ・パーティションとして DMS か SMS のどちらかの同一のストレージ・メカニズムを使用しなければなりません。 LARGE 表スペース内の索引は、最大 2^{29} ページまで含めることが可能です。すべての表スペースは、同一のデータベース・パーティション・グループになければなりません。

パーティション索引 は、複数の*索引パーティション* に索引データを分割するという索引編成スキームを使用します。分割は、表のパーティション・スキームに従って行われます。各索引パーティションは、対応するデータ・パーティション内の表の行のみを参照します。特定のデータ・パーティション用の索引パーティションはすべて、同じ索引オブジェクト内にあります。

DB2 バージョン 9.7 フィックスパック 1 以降では、パーティション表内の XML 列の XML データに対するユーザー作成の索引は、パーティション索引と非パーティション索引のどちらにもすることができます。デフォルトはパーティション索引です。システム生成の XML 領域索引は必ずパーティション化され、システム生成の列パス索引は常に非パーティションです。 DB2 V9.7 では、XML データに対する索引はすべて非パーティションです。

非パーティション索引の利点には、以下のものがあります。

- 各索引に異なる表スペース特性を定義できます (例えば、スペースの使用効率をより良いものにするには、ページ・サイズを異ならせると役立つ場合があります)。
- 索引は、相互に独立して再編成できます。
- 索引のドロップ操作に関するパフォーマンスが向上します。
- 入出力競合を削減して、索引データにより効率的な同時アクセスを提供するのに役立ちます。
- 個々の索引をドロップすると、索引再編成を行わなくてもシステムがスペースをすぐに使用できます。

パーティション索引の利点には、以下のものがあります。

- データのロールインおよびロールアウトのパフォーマンスが向上します。
- 索引がパーティション化されるため、索引ページ上の競合が少なくなります。
- 各索引パーティションの索引 B ツリー構造により、以下の利点が得られる場合があります。
  - 挿入、更新、削除、およびスキャンのパフォーマンスが向上する。これは、表のすべてのデータを参照する索引に比べて、通常、索引パーティションの B ツリーの中に含まれるレベルの数が少ないためです。
  - パーティションの除去が有効である場合、スキャンのパフォーマンスおよび並行性が向上する。パーティションの除去は、パーティション索引および非パーティション索引の両方のスキャンで使用可能ですが、より効果的なのはパーティション索引のスキャンの場合です。それは、各索引パーティションには、対応するデータ・パーティション以外のキーは含まれないためです。この構成により、非パーティション索引での同様の照会に比べて、スキャンするキーと索引ページの数が少なくて済む場合があります。

非パーティション索引は索引列上の順序を常に保持しますが、パーティション索引は、一部のシナリオ (例えば、パーティション列が索引列と一致しない場合、および複数のパーティションがアクセスされる場合) において、複数のパーティションにまたがる順序を失うことがあります。

オンラインの索引作成中は、表への同時読み取り/書き込みアクセスが許可されます。オンライン索引が作成された後、索引作成中に表に行われた変更が、新しい索引に適用されます。表への書き込みアクセスは、索引の作成が完了し、トランザクションがコミットするまで、ブロックされます。パーティション索引の場合、各データ・パーティションは、そのデータ・パーティションに対して (索引パーティションの作成中に) 加えられた変更を適用する期間中に限り、静止されて読み取り専用アクセスを受け入れます。

パーティション索引サポートは、ALTER TABLE...ATTACH PARTITION ステートメントを使用してデータをロールインするときに、特に役立ちます。非パーティション索引が存在している場合 (表に XML データがある場合、XML 列パスの索引は含まない)、パーティションのアタッチの後に、SET INTEGRITY ステートメントを発行します。このステートメントは、非パーティション索引の保守、範囲妥当性検査、制約チェック、およびマテリアライズ照会表 (MQT) の保守に必要です。非パーティション索引の保守は、時間がかかり、大容量のログ・スペースを必要とする可能性があります。パーティション索引を使用して、この保守のコストを回避してください。

アタッチ操作の後に、保守対象の表の非パーティション索引 (XML 列パス索引を除く) が存在する場合、SET INTEGRITY...ALL IMMEDIATE UNCHECKED ステートメントは SET INTEGRITY...IMMEDIATE CHECKED ステートメントのように機能します。すべての整合性処理、非パーティション索引の保守、および表の状態の遷移は、SET INTEGRITY...IMMEDIATE CHECKED ステートメントが発行された場合のように実行されます。

359 ページの図 48 のダイアグラムは、 1 つのパーティション表に対する 2 つの非パーティション索引を示しており、それぞれの索引は別々の表スペースにありま

す。

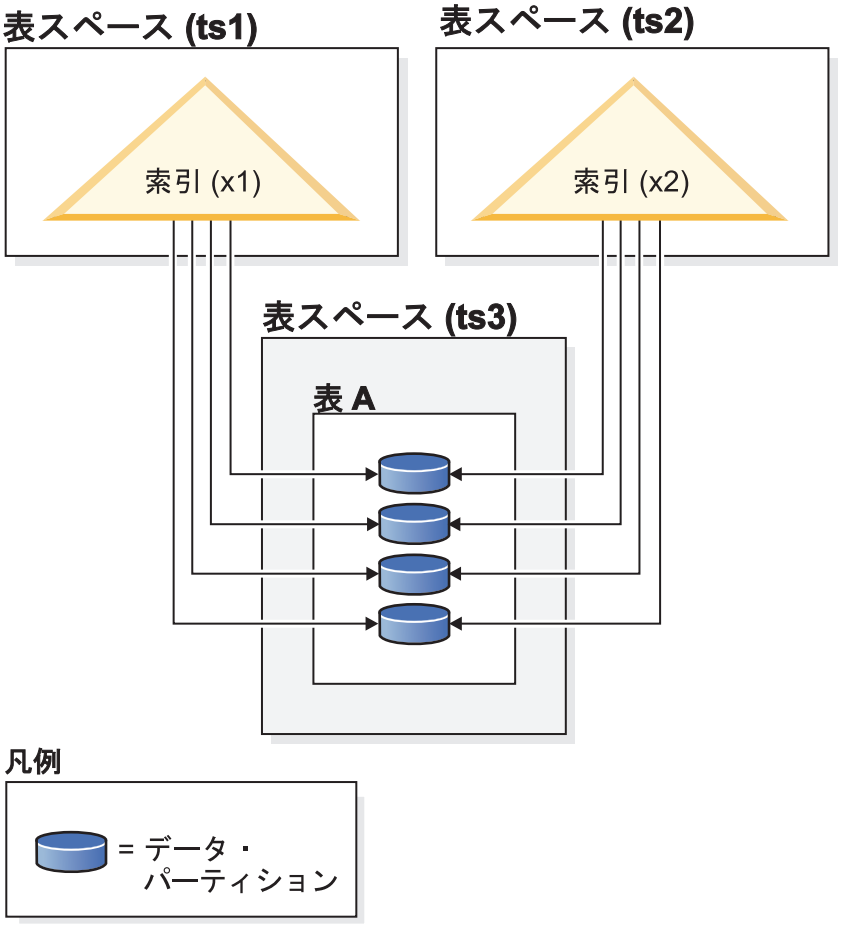


図 48. パーティション表上の非パーティション索引

360 ページの図 49 のダイアグラムは、 2 つのデータベース・パーティションにわたり、なおかつ 1 つの表スペースに内在するパーティション表の、非パーティション索引を示しています。

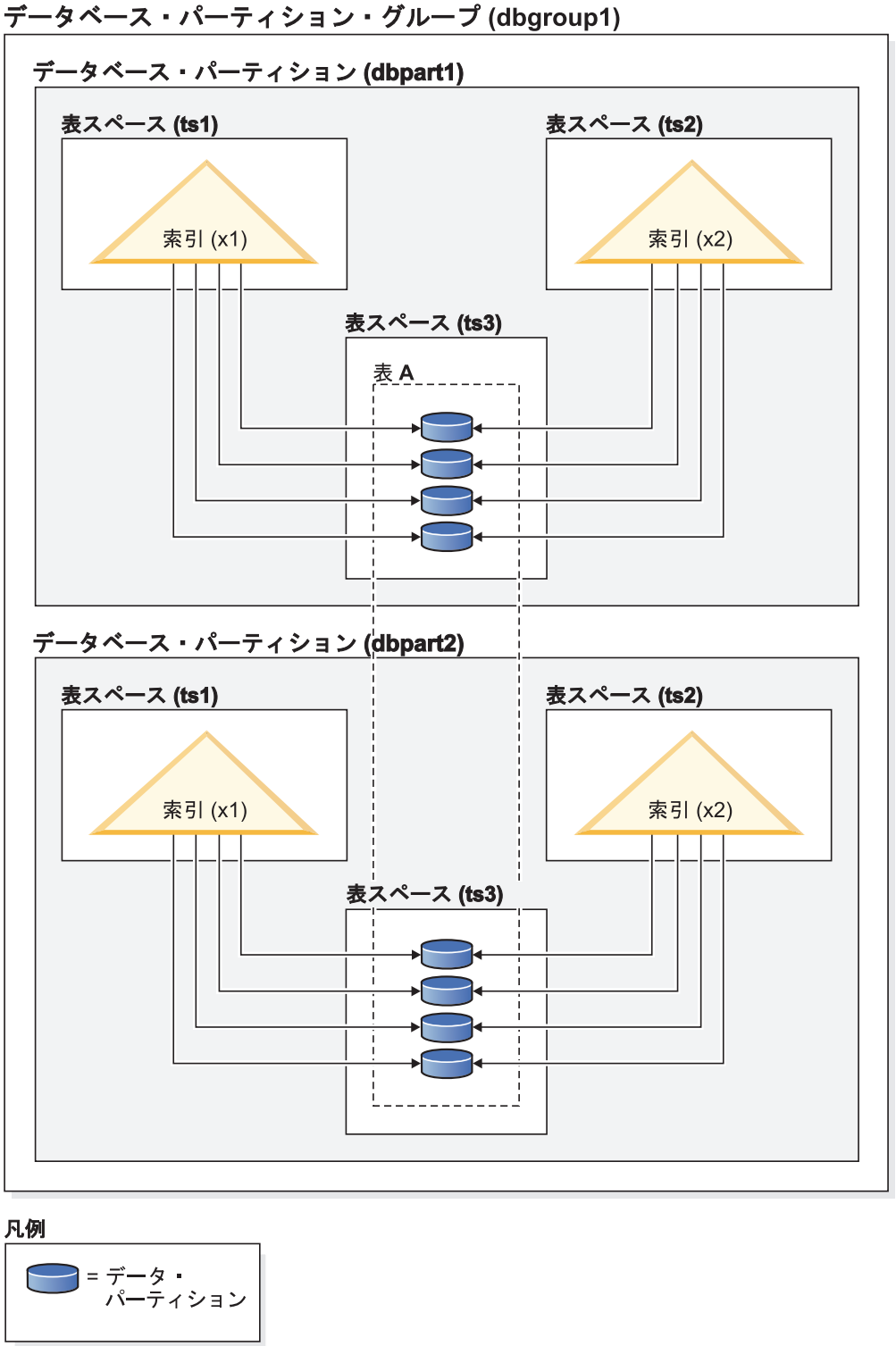


図 49. 分散とパーティション化の両方が行われている表における非パーティション索引

361 ページの図 50 のダイアグラムは、 1 つのパーティション表に対してパーティション索引と非パーティション索引が混在する状態を示しています。

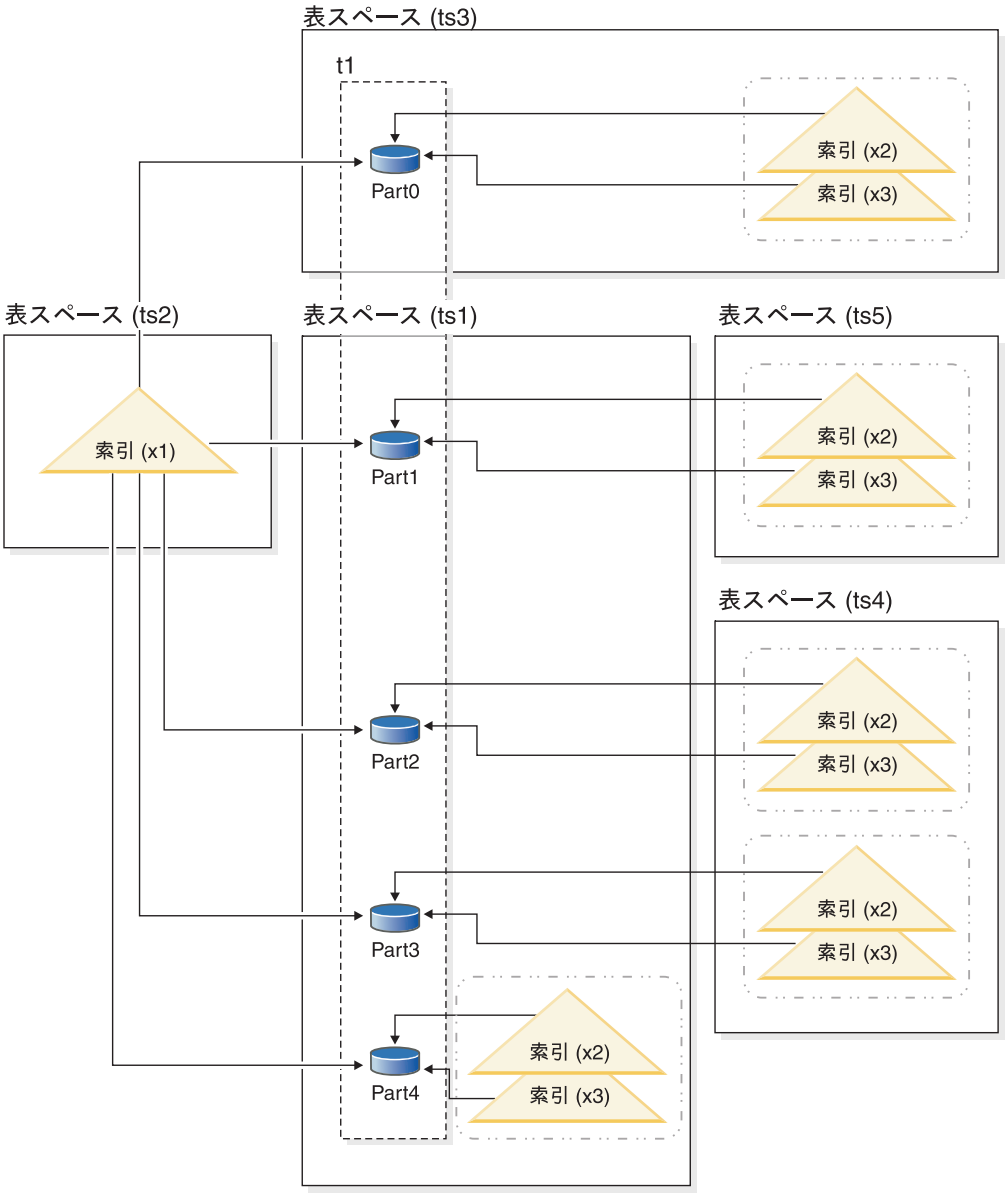


図 50. パーティション表上のパーティション索引と非パーティション索引

非パーティション索引 X1 は、すべてのデータ・パーティションの行を参照します。対照的に、パーティション索引 X2 および X3 は、関連するデータ・パーティションの行のみを参照します。表スペース TS3 は、関連するデータ・パーティションの表スペースを共有する索引パーティションも示しています。この構成は、パーティション索引のデフォルトです。

方法はそれぞれ異なりますが、非パーティション索引およびパーティション索引のデフォルトの位置をオーバーライドすることができます。非パーティション索引の場合、索引を作成するときに表スペースを指定できます。パーティション索引の場合は、表を作成するときに、索引パーティションを格納する表スペースを決めておく必要があります。

非パーティション索引

非パーティション索引の索引位置をオーバーライドするには、 CREATE INDEX ステートメントで IN 節を使用して、代わりの表スペース・ロケーションを索引用に指定できます。必要に応じて、別々の索引を別々の表スペースに配置することができます。パーティション化されていない索引を配置する場所を指定しないでパーティション表を作成し、表スペースを指定しない CREATE INDEX ステートメントを使用して索引を作成する場合、最初のアタッチされたデータ・パーティションまたは表示可能なデータ・パーティションの表スペースに索引が作成されます。次の 3 つの考え得る各ケースをケース 1 から順番に評価して、索引が作成される場所を判別します。この評価により、一致したケースが見つかると停止して、索引の表スペースの配置場所を判別します。

ケース 1:
索引表スペースが CREATE INDEX...IN *tbspace* ステートメントで
指定される場合、この索引用に指定された表スペースを使用します。

ケース 2:
索引表スペースが CREATE TABLE...INDEX IN *tbspace* ステートメントで
指定される場合、この索引用に指定された表スペースを使用します。

ケース 3:
表スペースが指定されない場合、
最初のアタッチされたデータ・パーティション
または可視のデータ・
パーティションによって使用される表スペースを選択します。

パーティション索引

デフォルトでは、索引パーティションは、参照するデータ・パーティションと同じ表スペースに配置されます。このデフォルト動作をオーバーライドするには、CREATE TABLE ステートメントを使用して定義するデータ・パーティションごとに INDEX IN 節を使用する必要があります。つまり、パーティション表でのパーティション索引の使用を計画している場合、表を作成するときに、索引パーティションを保管する場所をあらかじめ決めておく必要があります。パーティション索引を作成するときに INDEX IN 節を使用しようとすると、エラー・メッセージを受け取ります。

例 1: パーティション表 SALES (a int、b int、c int) を想定し、ユニーク索引 A_IDX を作成します。

```
create unique index a_idx on sales (a)
```

表 SALES がパーティション化されるため、索引 a_idx もパーティション索引として作成されます。

例 2: 索引 B_IDX を作成します。

```
create index b_idx on sales (b)
```

例 3: パーティション索引の索引パーティションのデフォルトの位置をオーバーライドするには、パーティション表を作成するときに定義するパーティションごとに INDEX IN 節を使用します。以下の例では、表 Z の索引が表スペース TS3 に作成されます。

```
create table z (a int, b int)
      partition by range (a) (starting from (1)
      ending at (100)  index in ts3)

   create index c_idx on z (a) partitioned
```

パーティション表上の非パーティション索引のクラスタリング

クラスタリング索引には、REGULAR 表に対するのと同じ利点がパーティション表に関してもあります。しかし、クラスタリング索引を選択する際には表パーティション・キー定義に関して注意が必要です。

任意のクラスタリング・キーを使用して、パーティション表にクラスタリング索引を作成できます。データベース・サーバーは、クラスタリング索引を使用して、各データ・パーティション内でデータをローカルにクラスター化しようとします。クラスター化された挿入操作の際、適切なレコード ID (RID) を検出するために索引検索が実行されます。この RID は、表内でレコードを挿入するスペースを検索する際の開始点として使用されます。効率が良く、最適化されたクラスタリングを実行するには、索引列と表パーティション・キー列間に相関がなければなりません。そのような相関を確保する 1 つの方法は、以下の例に示されているように、表パーティション・キー列を使用して索引列を先頭に付けることです。

```
partition by range (month, region)
create index...(month, region, department) cluster
```

データベース・サーバーがこの相関を強制することはありませんが、適切なクラスタリングを行うために索引中のすべてのキーがパーティション ID ごとに互いにグループ化されていることが期待されています。例えば、QUARTER 上にパーティションされた表が 1 つあり、クラスタリング索引が DATE で定義されているとします。 QUARTER と DATE の間には関連があり、データ・パーティションのすべてのキーが索引内で相互にグループ化されているので、効率の良い最適なデータのクラスタリングを行うことが可能です。 364 ページの図 51 は、クラスタリングが表パーティション・キーと相互に関連がある場合にのみ、最適なスキャン・パフォーマンスが得られることを示しています。

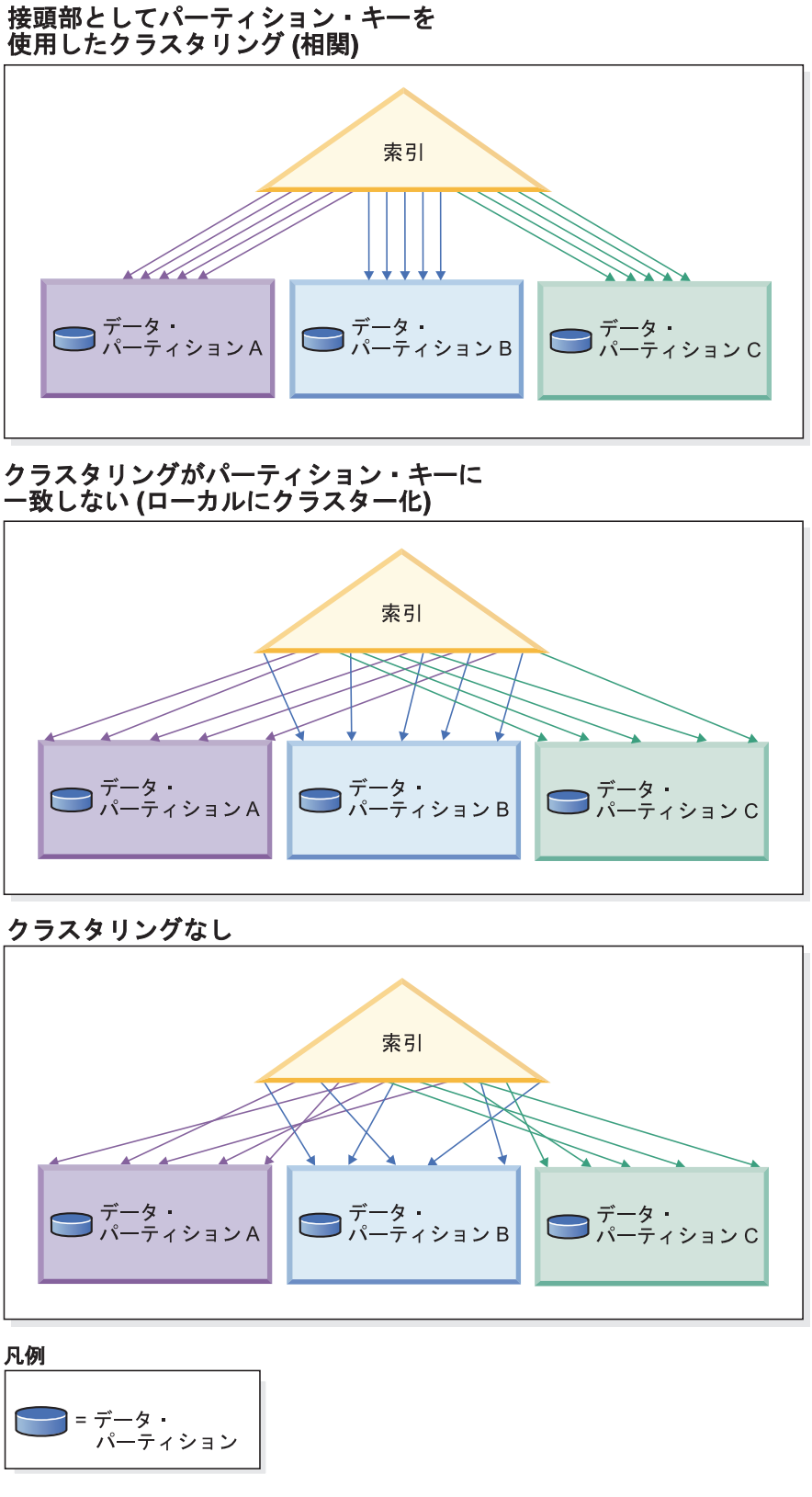


図 51. パーティション表でのクラスター化された索引の見込まれる効果。

クラスタリングの利点には、以下のものがあります。

- 各データ・パーティション内で、行はキーの順序になります。

- クラスタリング索引は、キーの順序で表内をトラバースするため、スキャンのパフォーマンスが向上します。なぜなら、スキャナーは最初のページの最初の行を取り出し、その後その同じページの各行を取り出してから次のページに移動するからです。つまり、どの時点でも表の 1 ページのみがバッファー・プール内になければならないという意味です。対照的に、表がクラスター化されていない場合、異なるページから各行が取り出される確率が高くなります。バッファー・プールが表全体を保持できる場合を除いて、ほとんどのページが何度も取り出される可能性が高くなり、スキャンの速度がとても遅くなります。

クラスタリング・キーが表パーティション・キーと相関していないものの、データがローカルにクラスター化される場合、バッファー・プールに各データ・パーティションの 1 ページを保持できるスペースが十分にあれば、クラスター化された索引の利点を十分に生かすことが依然として可能です。これは、特定のデータ・パーティションから取り出される各行が、同じパーティションから以前に取り出された行の近くにあるためです (364 ページの図 51 の 2 番目の例を参照)。

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・データベースへの変換

単一パーティション・データベースから複数パーティション・データベースへの変換の際に、設計アドバイザーを役立てることができます。

このタスクについて

新規の索引、マテリアライズ照会表 (MQT)、およびマルチディメンション・クラスタリング (MDC) 表についての推奨に加えて、設計アドバイザーはデータの分散についても推奨できます。

手順

1. **db2licm** コマンドを使用して、パーティション・データベース環境のライセンス・キーを登録します。
2. 複数パーティション・データベースのパーティション・グループに少なくとも 1 つの表スペースを作成します。

   注: 設計アドバイザーは、既存の表スペースへのデータ再配分のみ推奨できます。
3. **db2advis** コマンドにパーティション・オプションを指定して、設計アドバイザーを実行します。
4. **db2advis** 出力ファイルを若干変更してから、設計アドバイザーによって生成された DDL ステートメントを実行します。 設計アドバイザーによって生成された DDL スクリプトを実行できるようにするには、まずデータベース・パーティションをセットアップする必要があるため、返されるスクリプトで推奨内容はコメント化されています。推奨内容にしたがって表をトランスフォームする操作は、ご自分で行ってください。

第 23 章 並行性の管理

MDC および ITC 表と RID 索引スキャンのロック・モード

表または RID 索引のスキャン中に、マルチディメンション・クラスタリング (MDC) 表または挿入時クラスタリング (ITC) 表で取得されるロックのタイプは、有効になっている分離レベル、および使用中のデータ・アクセス・プランによって決まります。

以下の表では、種々のアクセス・プランごとに、各分離レベルにおいて MDC 表および ITC 表で取得されるロックのタイプをリストします。各項目には、表ロック、ブロック・ロック、および行ロックの 3 つの部分があります。ハイフンは、特定のロック細分性が使用できないことを示します。

表 9 から 14 で示すロックのタイプは、データ・ページの読み取りの据え置き時に RID 索引スキャンにおいて取得されるものです。 UR 分離レベルにおいて、索引の組み込み列に述部がある場合、分離レベルは CS にアップグレードされ、ロックは IS 表ロック、IS ブロック・ロック、または NS 行ロックにアップグレードされます。

- 表 1. 述部なしの表スキャンのロック・モード
- 表 2. ディメンション列上の述部のみでの表スキャンのロック・モード
- 表 3. 索引と他の述部 (sargs、resids) での表索引スキャンのロック・モード
- 表 4. 述部なしの RID 索引スキャンのロック・モード
- 表 5. 単一修飾行での RID 索引スキャンのロック・モード
- 表 6. 開始述部と停止述部のみでの RID 索引スキャンのロック・モード
- 表 7. 索引述部のみでの RID 索引スキャンのロック・モード
- 表 8. 他の述部 (sargs、resids) での RID 索引スキャンのロック・モード
- 表 9. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 述部なしでの RID 索引スキャン
- 表 10. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 述部なしでの RID 索引スキャン後
- 表 11. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 述部 (sargs、resids) での RID 索引スキャン
- 表 12. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 述部 (sargs、resids) での RID 索引スキャン後
- 表 13. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 開始述部と停止述部のみの RID 索引スキャン
- 表 14. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 開始述部と停止述部のみの RID 索引スキャン後

注: ロック・モードは、SELECT ステートメントの *lock-request-clause* を使用して明示的に変更できます。

表 16. 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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	S/-/-	U/-/-	SIX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/U	IX/X/-	IX/I/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

表 17. 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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/X/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/U/-	X/X/-

表 18. 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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 19. 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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	S/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

表 20. 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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/IS/S	IX/IX/U	IX/IX/X	X/X/X	X/X/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

表 21. 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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/IS/S	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

表 22. 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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 23. 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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 24. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S/S	IX/IX/S		X/-/-	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

表 25. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 述部なしでの RID 索引スキャン後

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IN/IN/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

表 26. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 述部 (sargs、resids) での RID 索引スキャン

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S/-	IX/IX/S		IX/IX/S	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

表 27. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 述部 (sargs、resids) での RID 索引スキャン後

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 28. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 開始述部と停止述部のみの RID 索引スキャン

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/IS/S	IX/IX/S		IX/IX/X	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

表 29. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 開始述部と停止述部のみの RID 索引スキャン後

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IS/-/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

MDC ブロック索引スキャンのロック・モード

ブロック索引スキャン中に、マルチディメンション・クラスタリング (MDC) 表で取得されるロックのタイプは、有効である分離レベル、および使用中のデータ・アクセス・プランによって決まります。

以下の表では、種々のアクセス・プランごとに、各分離レベルにおいて MDC 表で取得されるロックのタイプをリストします。各項目には、表ロック、ブロック・ロック、および行ロックの 3 つの部分があります。ハイフンは、特定のロック細分性が使用できないことを示します。

表 5 から 12 で示すロックのタイプは、データ・ページの読み取りの据え置き時にブロック索引スキャンにおいて取得されるものです。

- 表 1. 述部なしでの索引スキャンのロック・モード
- 表 2. ディメンション列上の述部のみでの索引スキャンのロック・モード
- 表 3. 開始述部と停止述部のみでの索引スキャンのロック・モード
- 表 4. 述部での索引スキャンのロック・モード
- 表 5. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 述部なしでのブロック索引スキャン
- 表 6. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 述部なしでのブロック索引スキャン後
- 表 7. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: ディメンション列上の述部のみのブロック索引スキャン

- 表 8. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: ディメンション列上の述部のみのブロック索引スキャン後
- 表 9. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 開始述部と停止述部のみのブロック索引スキャン
- 表 10. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 開始述部と停止述部のみのブロック索引スキャン後
- 表 11. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 他の述部 (sargs、resids) でのブロック索引スキャン
- 表 12. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 他の述部 (sargs、resids) でのブロック索引スキャン後

注: ロック・モードは、SELECT ステートメントの *lock-request-clause* を使用して明示的に変更できます。

表 30. 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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	S/--/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/--	X/X/--

表 31. ディメンション列上の述部のみでの索引スキャンのロック・モード

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

表 32. 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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S/-	IX/IX/S	IX/IX/S	IX/IX/S	IX/IX/S
RS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
CS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-

表 33. 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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 34. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 述部なしでの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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S/--	IX/IX/S		X/--/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

表 35. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 述部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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IN/IN/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	X/X/--	X/X/--

表 36. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: ディメンション列上の述部のみでのブロック索引スキャン

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S/--	IX/IX/--		IX/S/--	
RS	IS/IS/NS	IX/--/--		IX/--/--	
CS	IS/IS/NS	IX/--/--		IX/--/--	
UR	IN/IN/--	IX/--/--		IX/--/--	

表 37. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: ディメンション列上の述部のみでのブロック索引スキャン後

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/S/--	IX/X/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--

表 38. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 開始述部と停止述部のみの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		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S/--	IX/IX/--		IX/X/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

表 39. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 開始述部と停止述部のみのブロック索引スキャン後

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IN/IN/--	IX/IX/X		IX/X/--	
RS	IS/IS/NS	IN/IN/--		IN/IN/--	
CS	IS/IS/NS	IN/IN/--		IN/IN/--	
UR	IS/--/--	IN/IN/--		IN/IN/--	

表 40. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 他の述部 (sargs、resids) でのブロック索引スキャン

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IS/S/--	IX/IX/--		IX/IX/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

表 41. 据え置きデータ・ページ・アクセスに使用される索引スキャンのロック・モード: 他の述部 (sargs、resids) でのブロック索引スキャン後

分離レベル	読み取り専用および未確定のスキャン	カーソル操作		検索条件付き UPDATE または DELETE	
		スキャン	現在の場所	スキャン	更新または削除
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

パーティション表での動作をロックする

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。データ・パーティション・ロックは、**db2pd** コマンド、イベント・モニター、管理ビュー、および表関数からの出力によって識別されます。

表がアクセスされると、まず表ロックが取得され、その後データ・パーティション・ロックが必要に応じて取得されます。アクセス方式および分離レベルは、結果セットに示されていないデータ・パーティションのロックを必要とする場合があります。これらのデータ・パーティション・ロックが取得されると、表ロックと同じだけ長く保持される可能性があります。例えば、カーソル固定 (CS) の索引のスキャンは、以前にアクセスされたデータ・パーティションでロックを維持し、後にデータ・パーティション・ロックを再取得するコストを削減する可能性があります。

さらにデータ・パーティション・ロックは、表スペースへのアクセスを確保するコストを担います。非パーティション表の場合、表スペースのアクセスは表ロックによって処理されます。表レベルに排他ロックまたは共有ロックがある場合でも、データ・パーティション・ロックは発生します。

より良い細分性によって、1 つのトランザクションは、他のトランザクションが他のデータ・パーティションにアクセスしている間に、特定のデータ・パーティションへ排他的にアクセスでき、行ロックを避けることができます。これは、大量更新用に選択されるプランの結果として、またはデータ・パーティション・レベルへのロックのエスカレーションによって生じることがあります。データ・パーティションが共有または排他モードでロックされている場合であっても、多数のアクセス方式の表ロックは、通常意図的ロックです。これにより、並行性が増大します。しかし、非意図的ロックがデータ・パーティション・レベルで必要とされ、すべてのデータ・パーティションがアクセスされる可能性があることをプランが示す場合には、並行トランザクション間のデータ・パーティション・デッドロックが発生しないようにするために、表レベルで非意図的ロックが選択されることがあります。

LOCK TABLE ステートメント

パーティション表の場合、LOCK TABLE ステートメントによって取得される唯一のロックは表レベル・ロックです。これにより、後続のデータ操作言語 (DML) ステートメントによって行がロックされることが回避され、行、ブロック、またはデータ・パーティションの各レベルでのデッドロックを避けられます。 IN EXCLUSIVE MODE オプションを使用すると、索引を更新するときに排他的アクセスを保証するのに使用でき、大きな更新の間に索引が増大するのを制限するのに役立ちます。

ALTER TABLE ステートメントの LOCKSIZE TABLE オプションの影響

LOCKSIZE TABLE オプションを使用すると、意図的ロックを用いないで共有モードまたは排他モードで表をロックします。パーティション表の場合、このロック計画は、表ロックとデータ・パーティション・ロックの両方に適用されます。

行レベルおよびブロック・レベルのロックのエスカレーション

パーティション表の行レベルおよびブロック・レベルのロックは、データ・パーティション・レベルにエスカレートできます。これが生じる場合、データ・パーティションが共有、排他、または超排他モードにエスカレートされる場合にも、他のデータ・パーティションは影響を受けないので、表が他のトランザクションによりさらにアクセスできます。エスカレーションの通知ログ項目には、影響を受けるデータ・パーティションとその表の名前が含まれます。

ロック・エスカレーションによって非パーティション索引への排他的アクセスを確保することはできません。排他的アクセスの場合、以下のいずれかの条件が TRUE でなければなりません。

- ステートメントは排他表レベルのロックを使用する必要があります
- 明示的 LOCK TABLE IN EXCLUSIVE MODE ステートメントが発行される必要があります
- 表には LOCKSIZE TABLE 属性がなければなりません

パーティション索引の場合、データ・パーティションの排他的アクセス・モードまたは超排他アクセス・モードへのロック・エスカレーションにより、索引パーティションへの排他的アクセスを確保します。

ロック情報の解釈

SNAPLOCK 管理ビューは、パーティション表から戻されるロック情報を解釈する際に役立ちます。以下の SNAPLOCK 管理ビューは、オフラインでの索引再編成中にキャプチャーされたものです。

```
SELECT SUBSTR(TABNAME, 1, 15) TABNAME, TAB_FILE_ID, SUBSTR(TBSP_NAME, 1, 15) TBSP_NAME,
       DATA_PARTITION_ID, LOCK_OBJECT_TYPE, LOCK_MODE, LOCK_ESCALATION L_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 L_ESCALATION
------- ----------- ---------- ----------------- ---------------- --------- ------------
TP1           32768 -                          -1 TABLE_LOCK       Z                   0
```

```
TP1              4 USERSPACE1                0 TABLE_PART_LOCK  Z                    0
TP1              5 USERSPACE1                1 TABLE_PART_LOCK  Z                    0
TP1              6 USERSPACE1                2 TABLE_PART_LOCK  Z                    0
TP1              7 USERSPACE1                3 TABLE_PART_LOCK  Z                    0
TP1              8 USERSPACE1                4 TABLE_PART_LOCK  Z                    0
TP1              9 USERSPACE1                5 TABLE_PART_LOCK  Z                    0
TP1             10 USERSPACE1                6 TABLE_PART_LOCK  Z                    0
TP1             11 USERSPACE1                7 TABLE_PART_LOCK  Z                    0
TP1             12 USERSPACE1                8 TABLE_PART_LOCK  Z                    0
TP1             13 USERSPACE1                9 TABLE_PART_LOCK  Z                    0
TP1             14 USERSPACE1               10 TABLE_PART_LOCK  Z                    0
TP1             15 USERSPACE1               11 TABLE_PART_LOCK  Z                    0
TP1              4 USERSPACE1                - TABLE_LOCK       Z                    0
TP1              5 USERSPACE1                - TABLE_LOCK       Z                    0
TP1              6 USERSPACE1                - TABLE_LOCK       Z                    0
TP1              7 USERSPACE1                - TABLE_LOCK       Z                    0
TP1              8 USERSPACE1                - TABLE_LOCK       Z                    0
TP1              9 USERSPACE1                - TABLE_LOCK       Z                    0
TP1             10 USERSPACE1                - TABLE_LOCK       Z                    0
TP1             11 USERSPACE1                - TABLE_LOCK       Z                    0
TP1             12 USERSPACE1                - TABLE_LOCK       Z                    0
TP1             13 USERSPACE1                - TABLE_LOCK       Z                    0
TP1             14 USERSPACE1                - TABLE_LOCK       Z                    0
TP1             15 USERSPACE1                - TABLE_LOCK       Z                    0
TP1             16 USERSPACE1                - TABLE_LOCK       Z                    0

  26 record(s) selected.
```

この例では、パーティション表 TP1 に対するアクセスと並行性の制御に、タイプ TABLE_LOCK のロック・オブジェクトと -1 の DATA_PARTITION_ID を使用しています。タイプ TABLE_PART_LOCK のロック・オブジェクトは、各データ・パーティションのほとんどのアクセスおよび並行性の制御に使用されます。

この出力では、他にもタイプ TABLE_LOCK のロック・オブジェクトがありますが (TAB_FILE_ID 4 から 16)、これらのロック・オブジェクトには DATA_PARTITION_ID の値がありません。この種のロック (オブジェクトがデータ・パーティションまたはパーティション表の索引に対応する TAB_FILE_ID と TBSP_NAME を持つ) は、オンライン・バックアップ・ユーティリティーとの並行性を制御するのに使用される場合があります。

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、またはパーティション内並列処理が使用可能になっている環境では、各データベース・パーティションが独自のエージェント・プールを持っていて、そこからサブエージェントを引き出すことができます。

このプールがあるので、必要になったり作業を終了したりするたびに、サブエージェントを作成したり破棄したりする必要がありません。サブエージェントはプール内に関連エージェントとして残ることができ、それらが関連付けされたアプリケーションから、または新規のアプリケーションから新しい要求が出された場合には、データベース・マネージャーでそれらのサブエージェントを使用できます。

システム内のパフォーマンスとメモリー消費量への影響は、エージェント・プールのチューニング方法に強く関係しています。エージェント・プール・サイズに関するデータベース・マネージャー構成パラメーター (**num_poolagents**) は、1 つのデータベース・パーティションでアプリケーションとの関連付けを保持できるエージェントとサブエージェントの合計数に影響します。プール・サイズが小さすぎるので、プールが満杯になった場合には、サブエージェントは作業を行っているアプリケーションと自分自身との関連付けを切り離し、終了します。サブエージェントを作成してアプリケーションに再度関連付けするということを常に行わなければならないため、パフォーマンスが低下します。

デフォルトでは、**num_poolagents** は AUTOMATIC (値 100) に設定され、データベース・マネージャーはプールするアイドル・エージェントの数を自動的に管理します。

手動で設定した **num_poolagents** の値が小さすぎた場合には、ある 1 つのアプリケーションが関連サブエージェントによってプールを満杯にしてしまう場合があります。その後、別のアプリケーションが新しいサブエージェントを必要としているときに、そのエージェント・プール内にサブエージェントがない場合、そのアプリケーションは、他のアプリケーションのエージェント・プールにあるアクティブでないサブエージェントをリサイクルします。この動作により、リソースは完全に使用されます。

手動で設定した **num_poolagents** の値が大きすぎる場合には、関連するサブエージェントは、長い間未使用のままプール内に置かれ、他のタスクでは使用できないデータベース・マネージャー・リソースが使用される可能性があります。

接続コンセントレーターが使用可能である場合、**num_poolagents** の値は、同時にプール内でアイドル状態のままであるエージェントの正確な数を必ずしも反映するわけではありません。さらに多くのワークロード・アクティビティーを処理するために、一時的にエージェントが必要になる可能性があります。

データベース・マネージャーが独自のプロセスまたはスレッドとして実行する非同期アクティビティーは、データベース・エージェント以外にもあります。例えば、次のようなアクティビティーがあります。

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

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ページをバッファー・プールに取り出すのに必要な入出力の数を見積もります。この見積もりには、バッファー・プール使用率の予測も含まれています。既にバッファー・プールの中にあるページから行を読み取るために追加の入出力が必要ないためです。

索引スキャンの場合、オプティマイザーは、システム・カタログの情報を使用して、データ・ページをバッファー・プール内に読み込むための入出力コストを見積もります。それは、SYSCAT.INDEXES ビューの以下の列の情報を使用します。

- CLUSTERRATIO 情報はこの索引に関連して表データのクラスター化の程度を示します。数が大きいほど、行は索引キーの順序に並んでいます。表の行がほとんど索引キーの順序に並んでいれば、いくつもの行を 1 つのデータ・ページがバッファーにある内にそのページから読み取ることができます。この列の値が -1 の場合、オプティマイザーは、使用可能なら PAGE_FETCH_PAIRS および CLUSTERFACTOR 情報を使用します。
- PAGE_FETCH_PAIRS 列には CLUSTERFACTOR 情報と共に、データ・ページをさまざまなサイズのバッファー・プールに読み込むのに必要な入出力の数をモデル化するための数値の対がいくつか含まれています。これらの列のデータは、DETAILED 節を指定した **RUNSTATS** コマンドを索引に対して実行した場合だけ集められます。

索引のクラスタリング統計が使用不可である場合、オプティマイザーはデフォルト値を使います。この値は、索引に関するデータのクラスタリング率が低いことを想定しています。データがどの程度クラスター化されているかによってパフォーマンスに重大な影響を与える可能性があるため、表に対して定義する索引の 1 つについては、クラスター化が 100% 近くに維持されるようにしてください。一般的に、索引のキーがクラスター索引のキーのスーパーセットを表す場合と 2 つの索引のキー列間に実際の相関がある場合を除いて、100% クラスタリングできるのは 1 つの索引だけです。

表を再編成するときに、行をクラスター化するために使用する索引を指定し、挿入処理中にそのクラスター化を保持することができます。更新および挿入操作を行うと索引に対する表のクラスター化が低下する場合があるため、定期的に表を再編成することが必要な場合があります。挿入、更新、または削除操作が頻繁に発生する表に対する再編成の回数を少なくするには、ALTER TABLE ステートメントで PCTFREE 節を指定します。

MDC および ITC 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) 表と挿入時クラスタリング (ITC) 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きます。

標準の表と同様に、MDC 表と ITC 表は列に分割されたデータの行を含むページに編成されます。各ページの行は、レコード ID (RID) で識別されます。ただし、

MDC 表と ITC 表のページは、エクステント・サイズのブロックにグループ化されます。例えば、図 52 はエクステント・サイズが 4 の表を示しています。 0 から 3 の番号が付けられた最初の 4 ページは表の最初のブロックとなります。 4 から 7 の番号が付けられた次の 4 ページは、表の 2 番目のブロックとなります。

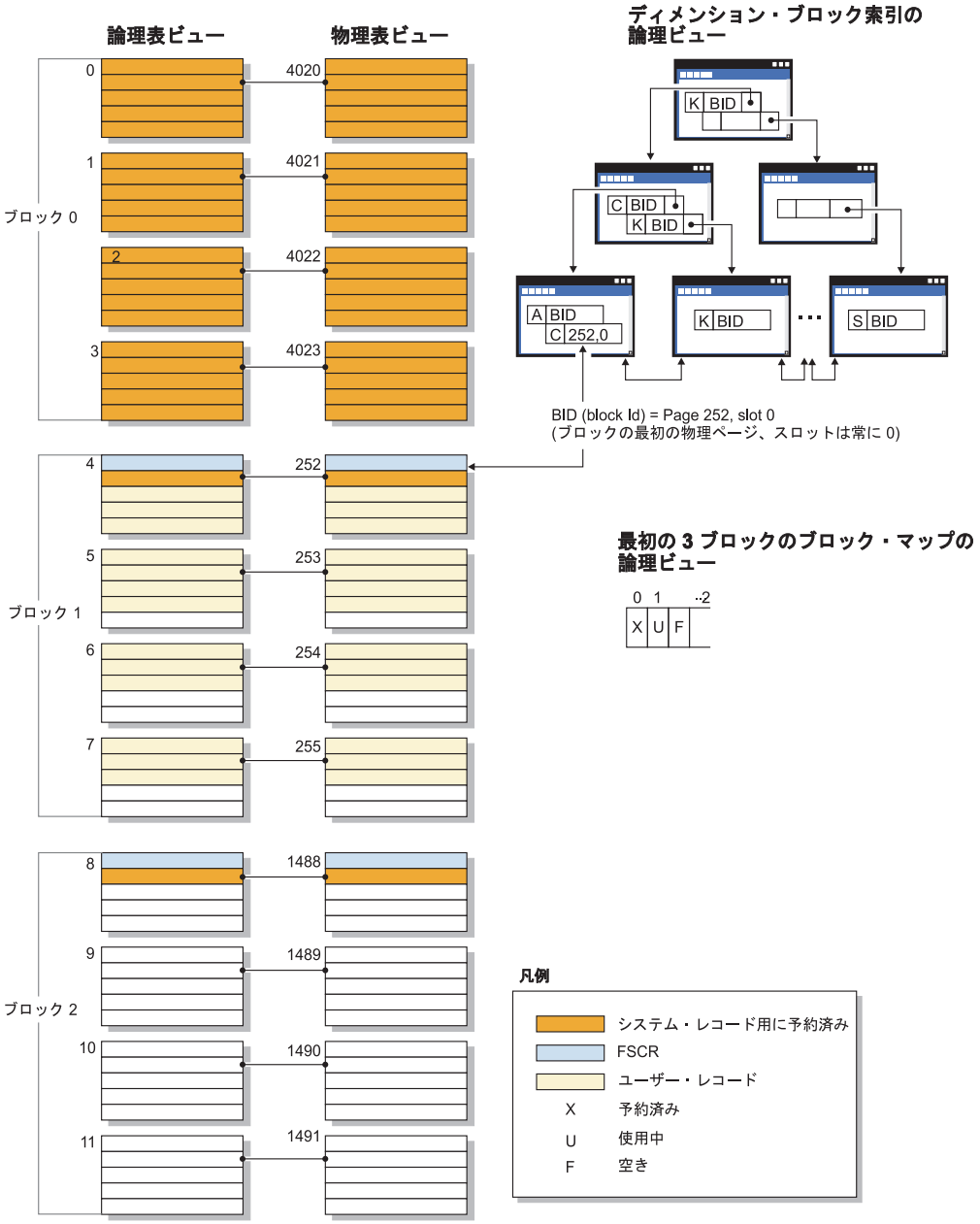


図 52. MDC および ITC 表の論理表、レコード、および索引構造

最初のブロックには、DB2 サーバーが表を管理するために使用する、フリー・スペース制御レコード (FSCR) を含む特殊な内部レコードが含まれます。続くブロックでは、最初のページに FSCR が含まれます。 FSCR はブロック内の各ページに存在する新規のレコード用にフリー・スペースをマップします。この使用可能なフリー・スペースは、表にレコードを挿入する際に使用されます。

名前が暗黙に示すように、MDC 表は複数のディメンションのデータをクラスター化します。各ディメンションは、CREATE TABLE ステートメントの ORGANIZE BY DIMENSIONS 節で指定した列または列のセットによって決まります。 MDC 表を作成するとき、以下の 2 つの索引が自動的に作成されます。

- 単一のディメンションの各ブロックに対するポインターを含む、ディメンション・ブロック索引。
- 複合ブロック索引。これには、すべてのディメンションのキー列が含まれ、挿入と更新アクティビティーの際にクラスタリングを保守するために使用されます。

オプティマイザーは、特定の照会に最適のアクセス・プランを判別する際にディメンション・ブロック索引を使用するアクセス・プランを検討します。照会にディメンション値についての述部があるとき、オプティマイザーはディメンション・ブロック索引を使用して、これらの値を含むエクステントを識別し、そこからフェッチします。エクステントはディスク上の物理的に連続したページなので、これにより入出力が最小になり、パフォーマンスが向上します。

さらに、データ・アクセス・プランの分析によって特定の RID 索引が照会のパフォーマンスを改善することが示された場合、その索引を作成することができます。

その名前が示すとおり、ITC 表は、データを行の挿入時刻に基づいてクラスター化します。 MDC 表と ITC 表の違いは次のとおりです。

- どのようなデータ・アクセスでも、ブロック索引は使用されない。
- 表に対して単一の複合ブロック索引しか作成されず、その索引は 1 つの仮想ディメンションから構成される。
- 索引がオプティマイザーによってプランに選択されることはない。これは、索引に含まれる列を SQL ステートメントによって参照することができないためです。

MDC 表と ITC 表は、表スペースに空のブロックを解放できます。

パーティション内並列処理の最適化ストラテジー

SQL ステートメントのコンパイル時に並列処理の多重度が指定された場合、オプティマイザーは、シングル・データベース・パーティション内で並列して照会を実行するアクセス・プランを選択します。

実行時には、サブエージェントと呼ばれる複数のデータベース・エージェントが作成されて、照会を実行します。サブエージェントの数は、SQL ステートメントのコンパイル時に指定された並列処理の多重度以下になります。

オプティマイザーは、アクセス・プランを並列化するため、プランを各サブエージェントによって実行される部分とコーディネーター・エージェントによって実行される部分とに分割します。サブエージェントは、表キューを介して、データをコーディネーター・エージェントか他のサブエージェントに渡します。パーティション・データベース環境では、サブエージェントは、表キューを介して、他のデータベース・パーティションのサブエージェントとの間でデータの送受信を行うことができます。

パーティション内の並列スキャン方式

リレーショナル・スキャンおよび索引スキャンは、同じ表または索引上で並列して実行することができます。並列リレーショナル・スキャンの場合、表はページ範囲または行範囲に分割され、それらはサブエージェントに割り当てられます。サブエージェントは割り当てられた範囲をスキャンし、その現行の範囲での作業が完了した時点で別の範囲が割り当てられます。

並列索引スキャンの場合、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、複数の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。

並列表スキャンは範囲パーティション表に対して実行できます。同様に、並列索引スキャンは、パーティション索引に対して実行できます。並列索引スキャンの場合には、パーティション索引は、索引キー値およびキー値あたりのキー項目数に基づいて、レコード範囲に分割されます。並列スキャンが開始すると、サブエージェントにレコード範囲が割り当てられ、サブエージェントは 1 つ範囲を完了するたびに新しい範囲が割り当てられます。索引パーティションは任意の時点で、予約されていない索引パーティションをスキャンしている可能性があるサブエージェントを使用して順次スキャンされます。これらのサブエージェントは、互いが完了するのを待ちません。データ・パーティションの除去分析に基づく照会に関連した索引パーティションのサブセットのみがスキャンされます。

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

このソートは、ソートが各サブエージェントごとに作成されるという点では、ラウンドロビン・ソートに似た働きをします。このソートでは、サブエージェントはハッシュ関数をソート列に適用して、行をどのソートに挿入するかを判別します。例えば、マージ結合の内部表と外部表がパーティション・ソートの場合、サブエージェントは、マージ結合を使用することによって、対応する表の部分を結合し並列で実行できます。

- 複製ソート

  このソートは、各サブエージェントがすべてのソート出力を必要とする場合に使用されます。あるソートが作成されると、サブエージェントは、そのソートに行が挿入されるときに同期化されます。ソートが完了すると、各サブエージェントがソート全体の読み取りを行います。このソートは、行数が少ないとき、データ・ストリームのバランスをとり直すのに使用できます。

- 共有ソート

  このソートは、複製ソートと同様の働きをしますが、共有ソートの場合は、ソートされた結果に対してサブエージェントが並列スキャンをオープンし、ラウンドロビン・ソートと同様の方法でサブエージェント間にデータが分配されます。

パーティション内並列一時表

サブエージェントが共同して同じ表に行を挿入することによって、一時表を生成できます。この表は、*共有一時表*と呼ばれます。サブエージェントは、データ・ストリームが複製されるか分割されるかに応じて、専用スキャンまたは並列スキャンのいずれかを共有一時表上でオープンします。

パーティション内の並列集約方式

集約操作は、サブエージェントによって並列に実行することができます。集約操作では、データをグループ化列上に配列する必要があります。サブエージェントがグループ化列の値の集合に関する行をすべて確実に受け取ることができれば、集約を最後まで完全に実行できます。これは、以前のパーティション・ソートのためにグループ化列上のストリームがすでに分割されている場合に生じます。

上記以外の場合は、サブエージェントは部分的に集約を実行し、別の方式を使用して集約を完了させます。その方式は以下のとおりです。

- 表キューをマージして、部分的に集約されたデータをコーディネーター・エージェントに送る。コーディネーター・エージェントにより集約が完全に行われます。
- 部分的に集約データをパーティション・ソートに挿入します。このソートは、グループ化列上で分割されるため、グループ化列の集合に関するすべての行が確実に 1 つのソート・パーティションに入れられます。
- 処理のバランスをとるためにストリームの複製が必要な場合は、部分的に集約データを複製ソートに挿入できます。個々のサブエージェントは複製ソートを使用して集約を完成させ、集約の結果と同じ内容のコピーを受け取ります。

パーティション内の並列結合方式

結合操作は、サブエージェントによって並列に実行することができます。並列結合の方式は、データ・ストリームの特性によって決められます。

結合は、結合の内部表または外部表でデータ・ストリームをパーティションに分割または複製 (あるいは、その両方) することによって、並列化できます。例えば、ネスト・ループ結合は、外部のストリームが並列スキャンのためにパーティション化され、さらに内部のストリームが各サブエージェントで別々に再評価されると並列化できます。マージ結合は、内部ストリームと外部ストリームがパーティション・ソートのために値でパーティション化されると並列化できます。

データのフィルタリングとデータの偏りによって、照会の実行中にサブエージェント間のワークロードに不均衡が生じる可能性があります。ワークロードの不均衡による効率の悪さは、結合その他、コンピューターに負荷を掛ける操作によって拡大します。オプティマイザーは照会のアクセス・プランの中で不均衡の原因を探し、バランシング方針を適用して、作業がサブエージェント間で均等に分割されるようにします。順序付けられていない外部データ・ストリームの場合、オプティマイザーはこの外部データ・ストリームに REBAL 演算子を使用して、結合のバランスをとります。順序付けられているデータ・ストリームの場合 (順序付けられているデータは索引のアクセスやソートにより生成されます)、オプティマイザーは共有ソートを使用してデータのバランスをとります。ソートが一時表にオーバーフローした場合、ソート・オーバーフローによる負担が大きいため、共有ソートは使用されません。

結合

結合 とは、情報の何らかの共通の領域に基づいて複数の表のデータを組み合わせるプロセスのことです。 1 つの表の行は、対応する行にある情報が結合基準 (*結合述部*) に基づいて合致する場合に、別の表の行と組になります。

例えば、次の 2 つの表を考えてみてください。

TABLE1

PROJ	PROJ_ID
A	1
B	2
C	3
D	4

TABLE2

PROJ_ID	NAME
1	Sam
3	Joe
4	Mary
1	Sue
2	Mike

表の PROJ_ID 列が同じ値になるように TABLE1 と TABLE2 を結合するには、次に示した SQL ステートメントを使用します。

```
select proj, x.proj_id, name
  from table1 x, table2 y
  where x.proj_id = y.proj_id
```

この場合、適切な結合述部は `where x.proj_id = y.proj_id` です。

照会により、次の結果セットが生成されます。

PROJ	PROJ_ID	NAME
A	1	Sam
A	1	Sue
B	2	Mike
C	3	Joe

PROJ	PROJ_ID	NAME
D	4	Mary

結合述部の特性、また表と索引の統計に基づいて判別されるコストに応じて、オプティマイザーは以下の結合方式のいずれかを選択します。

- ネスト・ループ結合
- マージ結合
- ハッシュ結合

2 つの表を結合する場合、1 つの表は結合の外部表として選択され、もう一方の表は内部表として選択されます。外部表は最初にアクセスされ、1 回だけスキャンされます。内部表が複数回スキャンされるかどうかは、結合の種類および使用できる索引によって異なります。照会によって 3 つ以上の表が結合されるとしても、オプティマイザーは 1 回に 2 つの表だけを結合します。必要なら中間結果を保持するために一時表が作成されます。

INNER または LEFT OUTER JOIN のような明示的な結合演算子を指定して、結合で表をどう使用するかを決定することができます。ただし、この方法で照会を変更する前に、表の結合方法をオプティマイザーで判別してみると良いでしょう。それから、照会のパフォーマンスを分析して結合演算子を追加するかどうかを決定します。

照会の最適化に影響を与えるデータベース・パーティション・グループ

パーティション・データベース環境では、オプティマイザーは、照会に対する最適のアクセス・プランを判別する際に表のコロケーションを認識し、使用します。

表が頻繁に結合照会に関係する場合、それらの表は、結合される各表にある行が同じデータベース・パーティションに置かれるように、データベース・パーティション間で分割する必要があります。結合操作を実行するときに、結合される両方の表にあるデータのコロケーションによって、データをあるデータベース・パーティションから別のデータベース・パーティションに移動できなくなります。同じデータベース・パーティション・グループに両方の表を置き、そのデータが確実に連結されるようにしてください。

データをより多くのデータベース・パーティションに配分すると、表のサイズに応じて照会の実行にかかる見積時間が減少します。表の数、表のサイズ、それらの表のデータがある場所、および結合が必要かどうかといった照会のタイプはすべて照会のコストに影響を与えます。

パーティション・データベースでの結合ストラテジー

パーティション・データベース環境での結合ストラテジーは、非パーティション・データベース環境でのストラテジーとは異なる場合があります。パフォーマンスを改善するために、標準の結合方式に加えて別の技法を適用することができます。

頻繁に結合が行われる表では、表コロケーションを考慮する必要があります。パーティション・データベース環境では、*表コロケーション* とは、互換性のあるパーテ

ィション・キーの数が同じである 2 つの表が、同じデータベース・パーティション・グループに保管されている場合に生じる状態のことです。この状態になると、結合処理はそのデータが保管されているデータベース・パーティションで実行できるようになり、結果セットをコーディネーター・データベース・パーティションに移動するだけで済みます。

表キュー

パーティション・データベース環境での結合技法の説明は以下の用語を使用します。

- *表キュー* (TQ と呼ばれることもある) は、データベース・パーティション間 (または、単一パーティション・データベースの場合はプロセッサー間) で行を転送するための機構です。
- *指示表キュー* (DTQ と呼ばれることもある) は、行が受信データベース・パーティションの 1 つにハッシュされる表キューです。
- *ブロードキャスト表キュー* (BTQ と呼ばれることもある) は、行がすべての受信データベース・パーティションに送信されるが、ハッシュは行われない表キューです。

表キューは、次の方法で表データを渡す場合に使用されます。

- パーティション間並列処理の使用時に、あるデータベース・パーティションから別のデータベース・パーティションに渡す
- パーティション内並列処理の使用時に、データベース・パーティション内で渡す
- 単一パーティション・データベースの使用時に、データベース・パーティション内で渡す

各表キューは単一方向にデータを渡します。コンパイラーはどこで表キューが必要とされているかを判断し、それらをプランに組み込みます。プランが実行されると、データベース・パーティション間の接続を行うとその表キューが開始されます。表キューがクローズされるのは、処理が終了したときです。

表キューには、以下に示すようにいくつかの種類があります。

- 非同期表キュー

  これらの表キューが非同期と呼ばれるのは、アプリケーションからフェッチ要求が出される前に、行の読み取りを行うためです。 FETCH ステートメントが出されたときには、行はこの表キューから取り出されます。

  非同期表キューは、SELECT ステートメントに FOR FETCH ONLY 節を指定した場合に使用されます。行の取り出しだけを行う場合には、非同期表キューが他よりも速い方法になります。
- 同期表キュー

  これらの表キューが同期と呼ばれるのは、アプリケーションによって FETCH ステートメントが出されるたびに行を 1 行読み取るためです。各データベース・パーティションでは、カーソルが、そのデータベース・パーティションから次に読み取られる行に位置づけられます。

同期表キューは、 SELECT ステートメントに FOR FETCH ONLY 節が指定されていない場合に使用されます。パーティション・データベース環境では、行の更新を行う場合には、データベース・マネージャーは同期表キューを使用します。

- マージ表キュー

  これらの表キューは、順序を保存します。

- 非マージ表キュー

  これらの表キューは*正規表*キュー とも呼ばれ、順序を保持しません。

- Listener 表キュー (LTQ と呼ばれることもある)

  これらの表キューは、相関副照会とともに使用されます。相関値が副照会に渡された後、このタイプの表キューを使用して、結果が親照会ブロックに戻されます。

パーティション・データベースでの結合方式

パーティション・データベース環境では、コロケーテッド結合、外部表のブロードキャスト結合、外部表の指示結合、内部表および外部表の指示結合、内部表のブロードキャスト結合、内部表の指示結合といった、いくつかの結合方式を使用できます。

以下の図中の q1、q2、および q3 は、表キューを指しています。参照される表は 2 つのデータベース・パーティションに分割されていて、矢印は、表キューが送られる方向を示します。なお、コーディネーター・データベース・パーティションはデータベース・パーティション 0 です。

コンパイラーによって選択された結合方式がハッシュ結合の場合、リモート・データベース・パーティションごとに作成されるフィルターを使用して、ハッシュ結合が処理されるデータベース・パーティションにタプルが送信される前にタプルを除去することができ、これによってパフォーマンスが向上します。

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。 392 ページの図 53 に例を示します。

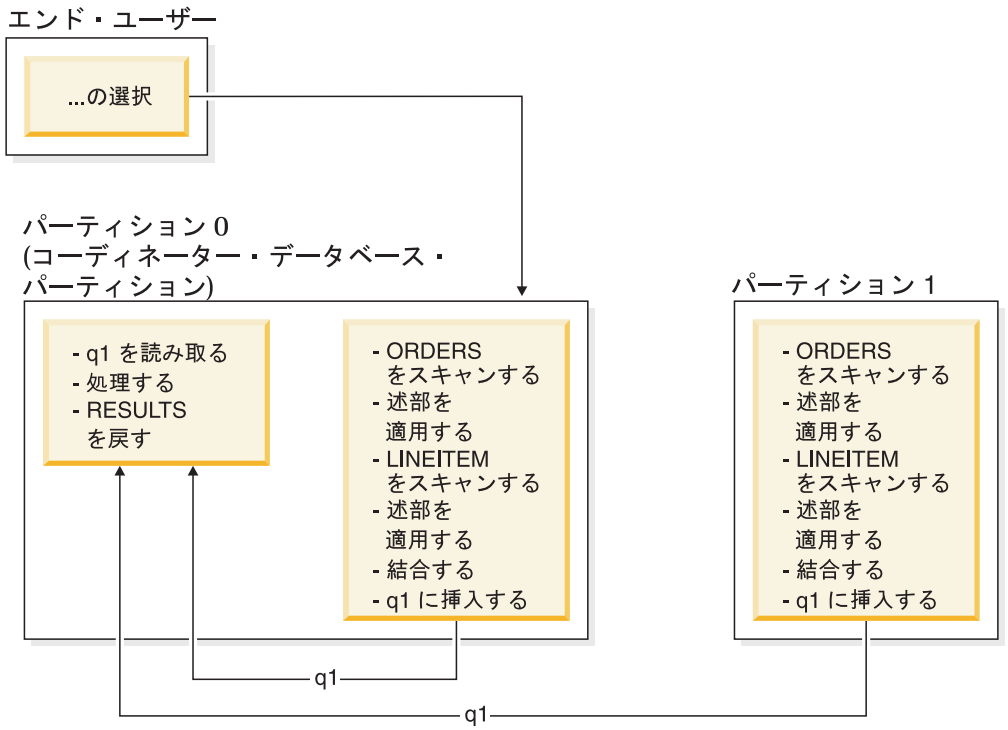


図 53. コロケーテッド結合の例

LINEITEM および ORDERS 表は ORDERKEY 列上でパーティション化される。結合は、各データベース・パーティションでローカルに行われる。この例では、結合述部は次のように想定されている。`orders.orderkey = lineitem.orderkey`

複製マテリアライズ照会表 (MQT) はコロケーテッド結合の可能性を高めます。

外部表のブロードキャスト結合

外部表のブロードキャスト結合は、結合される表の間に等価結合述部がない場合に使用できる並列結合方式を示します。また、この結合方式は、コスト面での効果においてこれが最良の結合方式となるその他の状況でも使用できます。例えば、外部表のブロードキャスト結合は、非常に大きな表が 1 つと非常に小さな表が 1 つあり、どちらの表も結合述部列上で分割されていない場合に使用されます。両方の表をパーティションに分割するよりも、小さな表を大きな表にブロードキャストするほうがコストがかからない可能性があります。 393 ページの図 54 に例を示します。

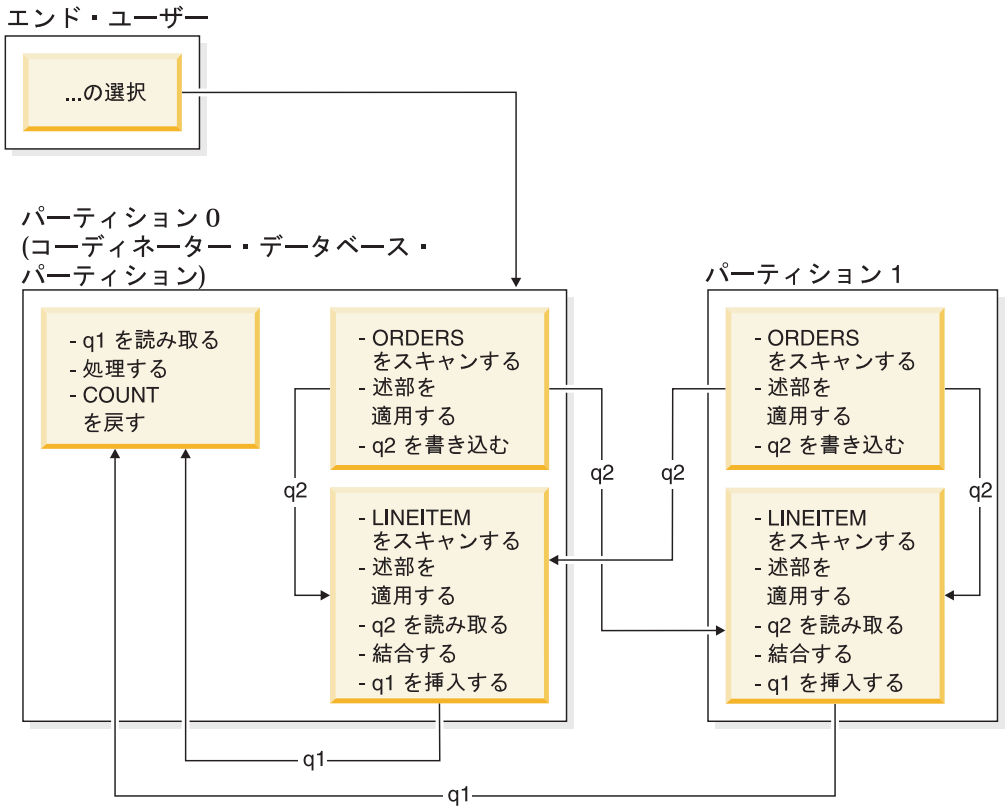


図 54. 外部表のブロードキャスト結合の例

ORDERS 表は、LINEITEM 表を持つデータベース・パーティションすべてに送られる。表キュー q2 は、内部表のデータベース・パーティションすべてにブロードキャストされる。

外部表の指示結合

外部表の指示結合方式では、外部表の各行を内部表の分割属性に基づいて内部表の一部分に送ります。結合は、このデータベース・パーティション上で行われます。394 ページの図 55 に例を示します。

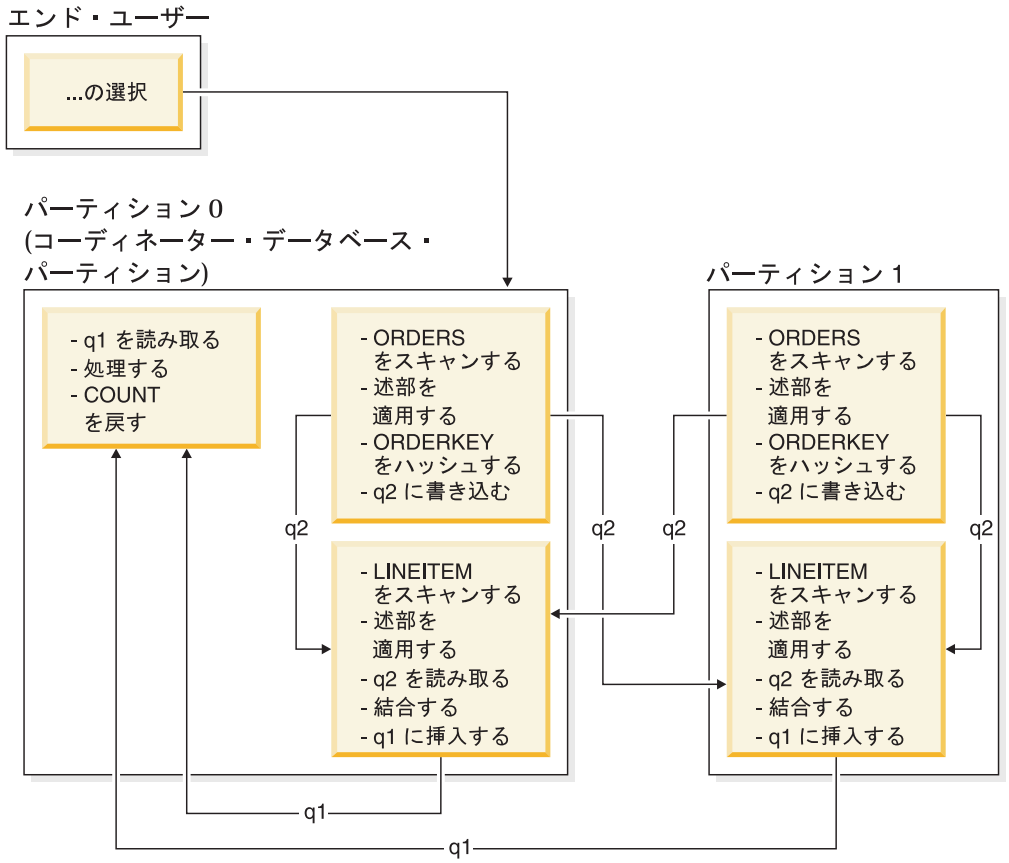


図 55. 外部表の指示結合の例

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`

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。 395 ページの図 56 に例を示します。

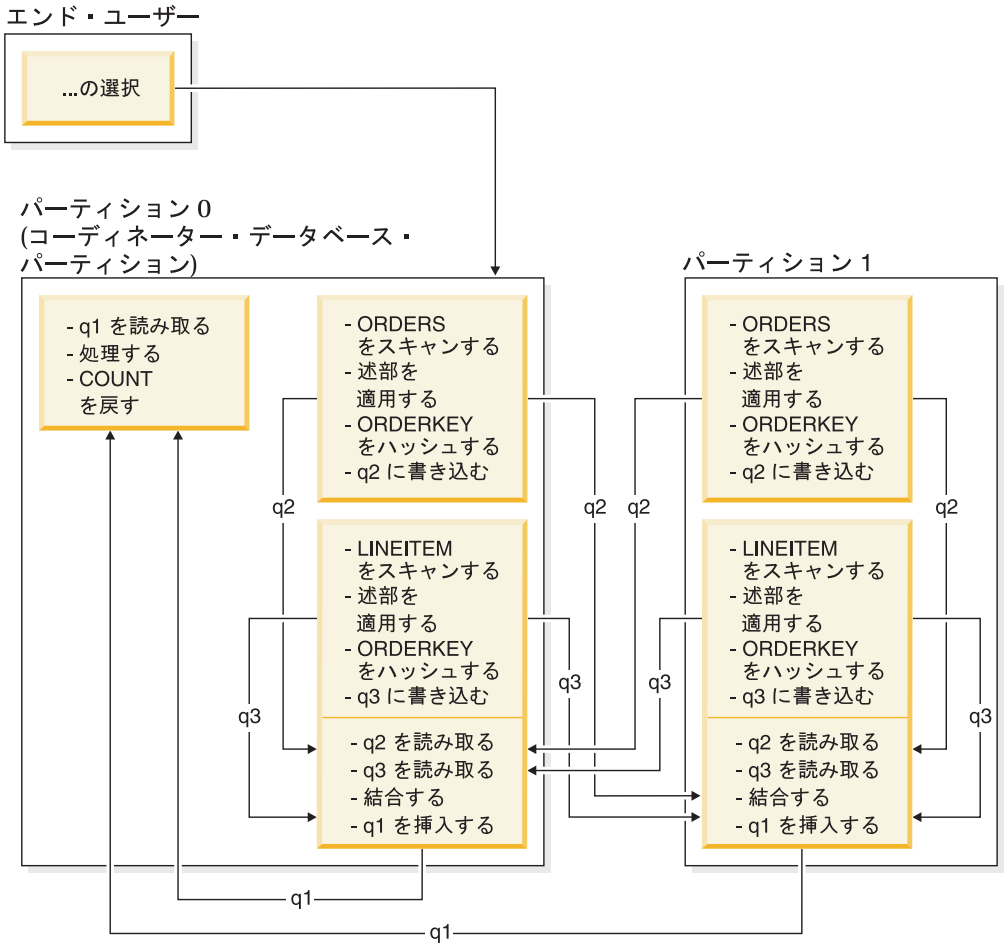


図 56. 内部表および外部表の指示結合の例

いずれの表も、ORDERKEY 列上ではパーティション化されない。どちらの表もハッシュされ、新しいデータベース・パーティションに送られて、そこで結合される。両方の表キュー q2 と q3 が送られる。この例では、結合述部は次のように想定されている。`orders.orderkey = lineitem.orderkey`

内部表のブロードキャスト結合

内部表のブロードキャスト結合方式では、内部表が外部表のすべてのデータベース・パーティションに対してブロードキャストされます。 396 ページの図 57 に例を示します。

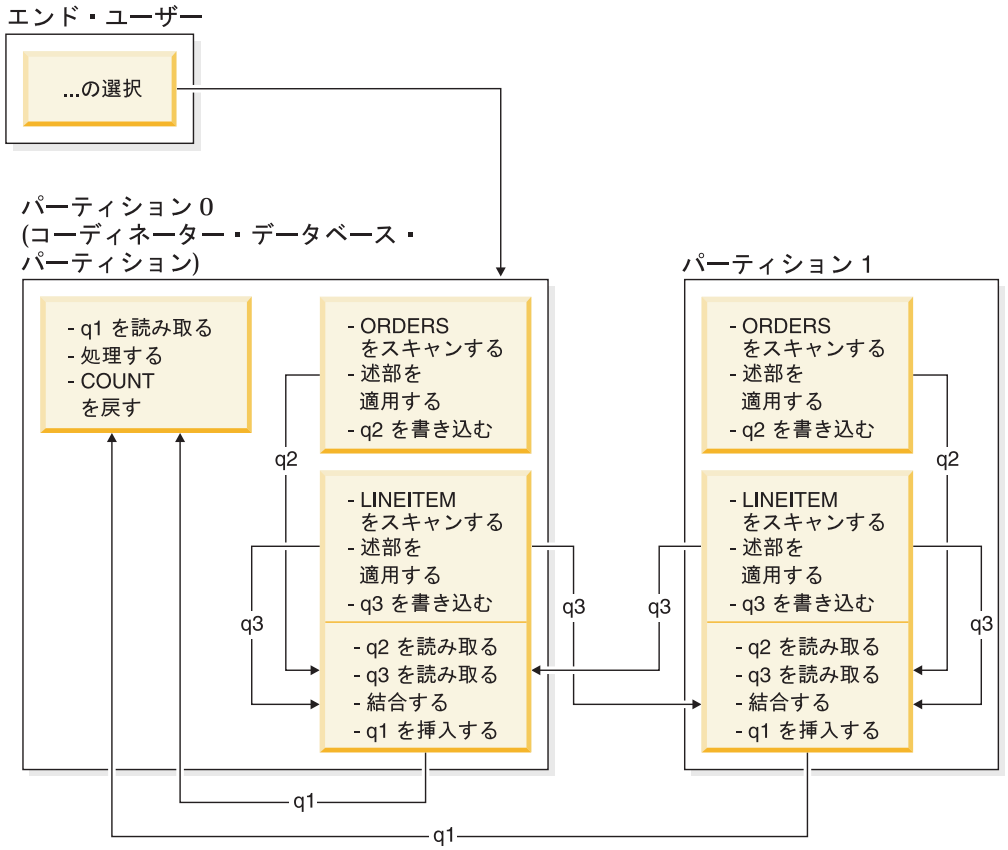


図 57. 内部表のブロードキャスト結合の例

LINEITEM 表は、ORDERS 表を持つデータベース・パーティションすべてに送られる。表キュー q3 は、外部表のデータベース・パーティションすべてにブロードキャストされる。

内部表の指示結合

内部表の指示結合方式では、内部表の各行を、外部表の分割属性に基づいて外部表のデータベース・パーティションの 1 つに送ります。結合は、このデータベース・パーティション上で行われます。 397 ページの図 58 に例を示します。

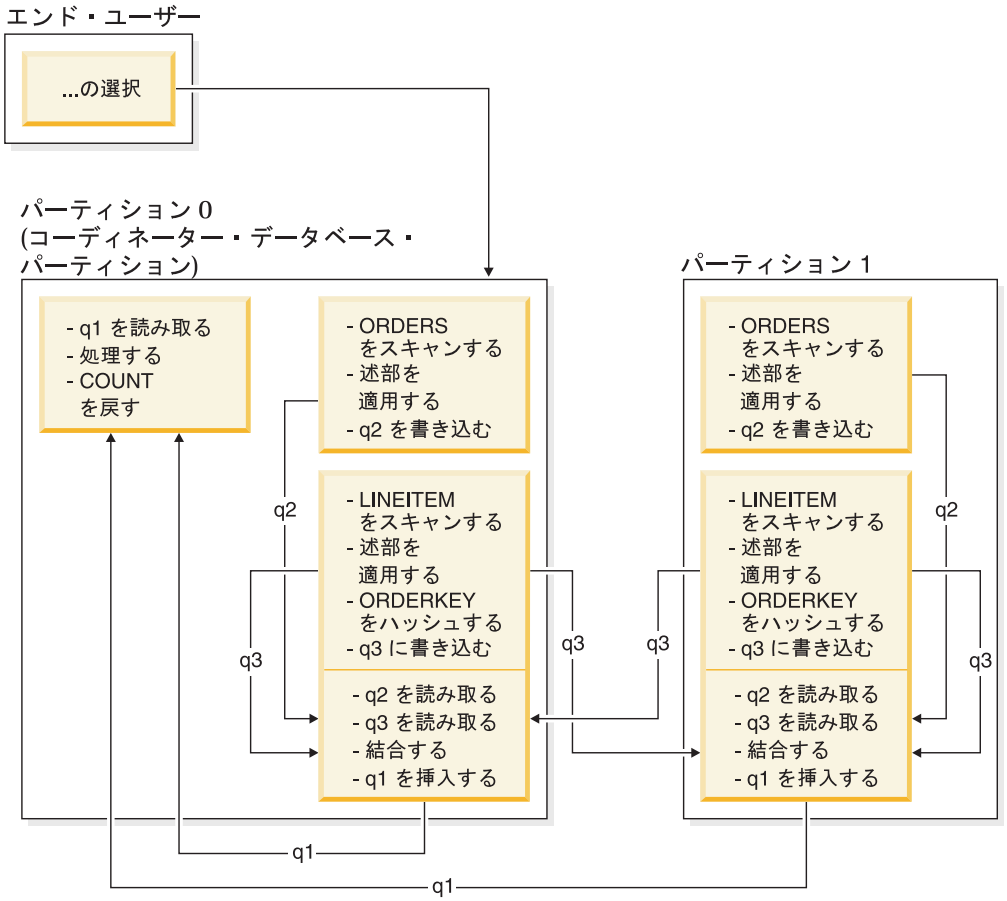


図 58. 内部表の指示結合の例

ORDERS 表は ORDERKEY 列上でパーティション化される。 LINEITEM 表は別の列でパーティション化される。 LINEITEM 表がハッシュされて、適切な ORDERS 表のデータベース・パーティションに送られる。この例では、結合述部は次のように想定されている。`orders.orderkey = lineitem.orderkey`

パーティション・データベース環境の複製されたマテリアライズ照会表

複製マテリアライズ照会表 (MQT) は、データベースが表データの事前計算された値を管理できるようにすることによって、パーティション・データベース環境で頻繁に実行される結合のパフォーマンスを改善します。

この文脈での複製された MQT とは、データベース内複製に関連した用語である点に注意してください。データベース間複製はサブスクリプション、コントロール表、異なったデータベース、および異なったオペレーティング・システムに配置されたデータと関連しています。

次の前提条件で以下の例を考えます。

- SALES 表は、REGIONTABLESPACE という名前の複数パーティション表スペースにあり、REGION 列で分割されています。
- EMPLOYEE 表および DEPARTMENT 表が単一パーティションのデータベース・パーティション・グループにあります。

EMPLOYEE 表の情報に基づき、複製 MQT を作成します。

```
create table r_employee as (
  select empno, firstnme, midinit, lastname, workdept
    from employee
  )
  data initially deferred refresh immediate
  in regiontablespace
  複製済み
```

複製された MQT の内容を次のように更新します。

```
refresh table r_employee
```

REFRESH ステートメントを使用した後は、複製表に対して (他の表に対して行う場合と同様に) runstats ユーティリティーを実行する必要があります。

次の照会は、従業員ごとの売上、部署の合計、総合計を計算します。

```
select d.mgrno, e.empno, sum(s.sales)
  from department as d, employee as e, sales as s
  where
    s.sales_person = e.lastname and
    e.workdept = d.deptno
  group by rollup(d.mgrno, e.empno)
  order by d.mgrno, e.empno
```

データベース・マネージャーは、1 つのデータベース・パーティションにしか存在しない EMPLOYEE 表を使用するのではなく、SALES 表が保管されている各データベース・パーティションで複製される MQT である R_EMPLOYEE を使用します。結合を行うときに、ネットワークを介して従業員の情報をそれぞれのデータベース・パーティションに移動させる必要はないので、パフォーマンスが向上します。

コロケーテッド結合における複製マテリアライズ照会表

複製 MQT は結合のコロケーションでも助けになります。例えば、スター・スキーマに 20 のデータベース・パーティションにまたがる大規模なファクト表がある場合、ファクト表とディメンション表の結合はこれらの表が連結されていると最も効率的です。同一のデータベース・パーティション・グループにすべての表があれば、多い場合でも 1 つのディメンション表がコロケーテッド結合のために正しくパーティション化されます。他のディメンション表はコロケーテッド結合では使用できません。それは、ファクト表の結合列がファクト表の分散キーに対応していないためです。

C1 で分割された FACT (C1、C2、C3、...) という表、C1 で分割された DIM1 (C1、dim1a、dim1b、...) という表、C2 で分割された DIM2 (C2、dim2a、dim2b、...) という表 (以下同様に続く) がある場合を考えます。この場合、述部 `dim1.c1 = fact.c1` は連結できるので、FACT と DIM1 の間の結合は完全です。これらの表は両方とも C1 列で分割されています。

しかし、DIM2 と述部 `dim2.c2 = fact.c2` が関係する結合は連結できません。これは FACT が、C2 列ではなく C1 列で分割されているからです。この場合、DIM2 をファクト表のデータベース・パーティション・グループに複製して、データベース・パーティションごとにローカルに結合するようにできます。

複製 MQT を作成する場合、ソース表はデータベース・パーティション・グループの単一パーティション表でも複数パーティション表でも構いません。ほとんどの場合、複製される表は小さく、単一パーティション・データベース・パーティション・グループ内に配置することができます。表からの列のサブセットだけを指定することにより、または述部を使用して適格となる行数を制限することにより、複製されるデータを制限できます。

ソース表のコピーをすべてのデータベース・パーティションに作成するため、複数パーティションのデータベース・パーティション・グループに複製 MQT を作成することもできます。すべてのデータベース・パーティションに対しソース表をブロードキャストするよりも、大規模なファクト表とディメンション表間の結合は、この環境内でローカルで行う方が良いです。

複製された表の索引は、自動的に作成されません。ソース表のものとは異なる索引を作成することができます。しかし、ソース表になかった制約違反を防ぐため、複製された表に対してはユニーク索引の作成および制約の定義は行えません (同じ制約がソース表にある場合でも定義できません)。

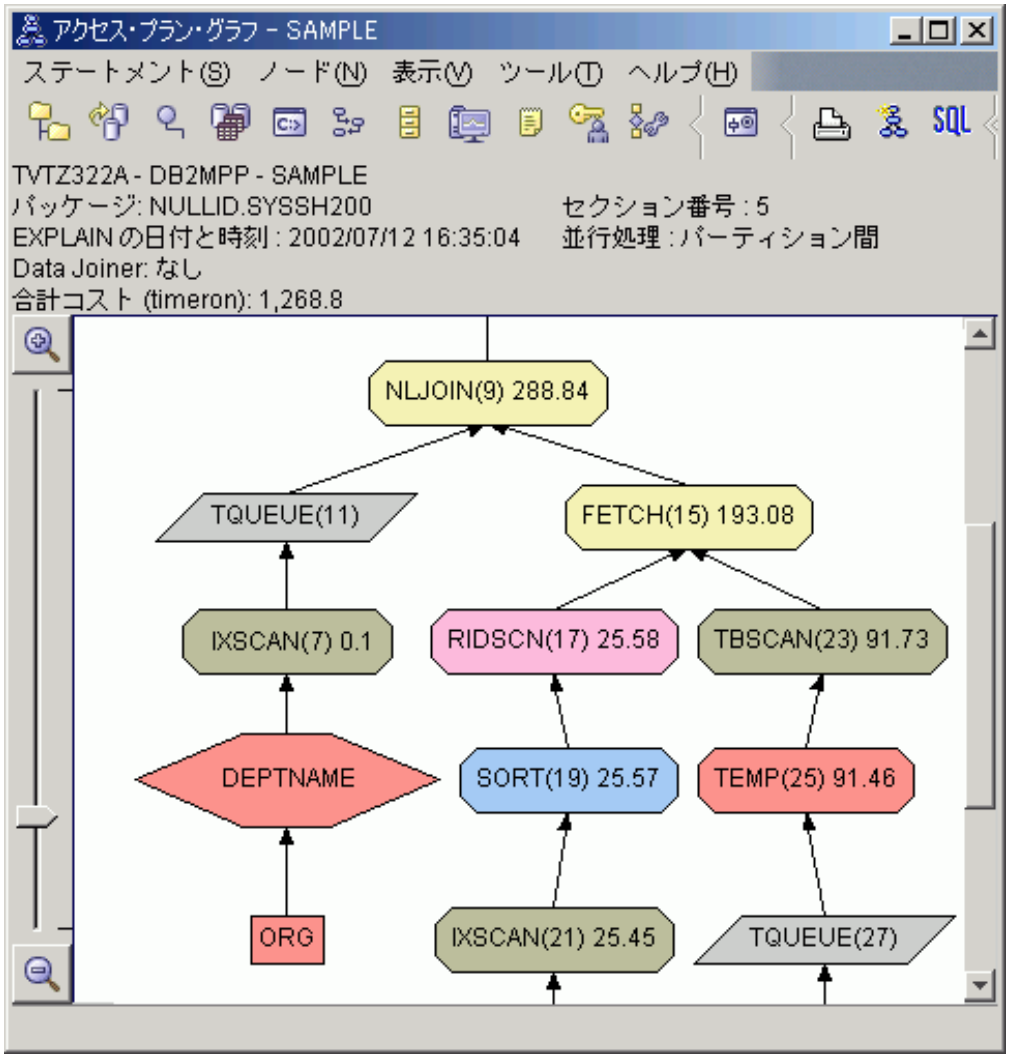
複製された表は照会で直接参照できますが、複製された表で DBPARTITIONNUM スカラー関数を使用して特定のデータベース・パーティション上の表データを参照することはできません。

複製された MQT が照会のためのアクセス・プランで使用されたかどうかを調べるには、DB2 Explain 機能を使用します。複製された MQT がオプティマイザーで選択されたアクセス・プランで使用されるかどうかは、結合されるデータによって異なります。オプティマイザーがオリジナル・ソース表をデータベース・パーティション・グループの他のデータベース・パーティションにブロードキャストするほうがコストがかからないと判断した場合、複製された MQT が使用されない場合もあります。

パーティション・データベース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。)

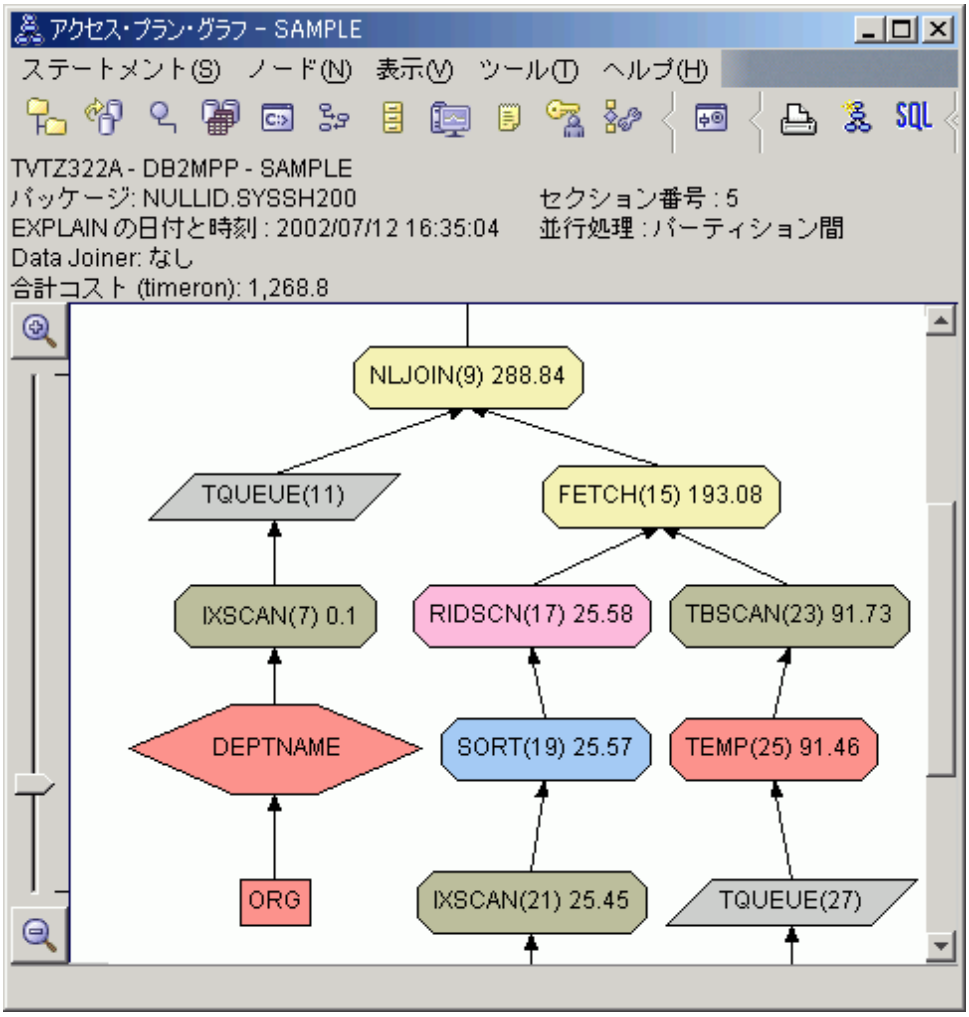
この照会 (Query 4) のアクセス・プラン・グラフを表示するには、「EXPLAIN されたステートメント履歴」ウィンドウで、Query Number 4 という名前の項目をダブルクリックします。このようにステートメントを実行するための「アクセス・プラン・グラフ」ウィンドウがオープンします。



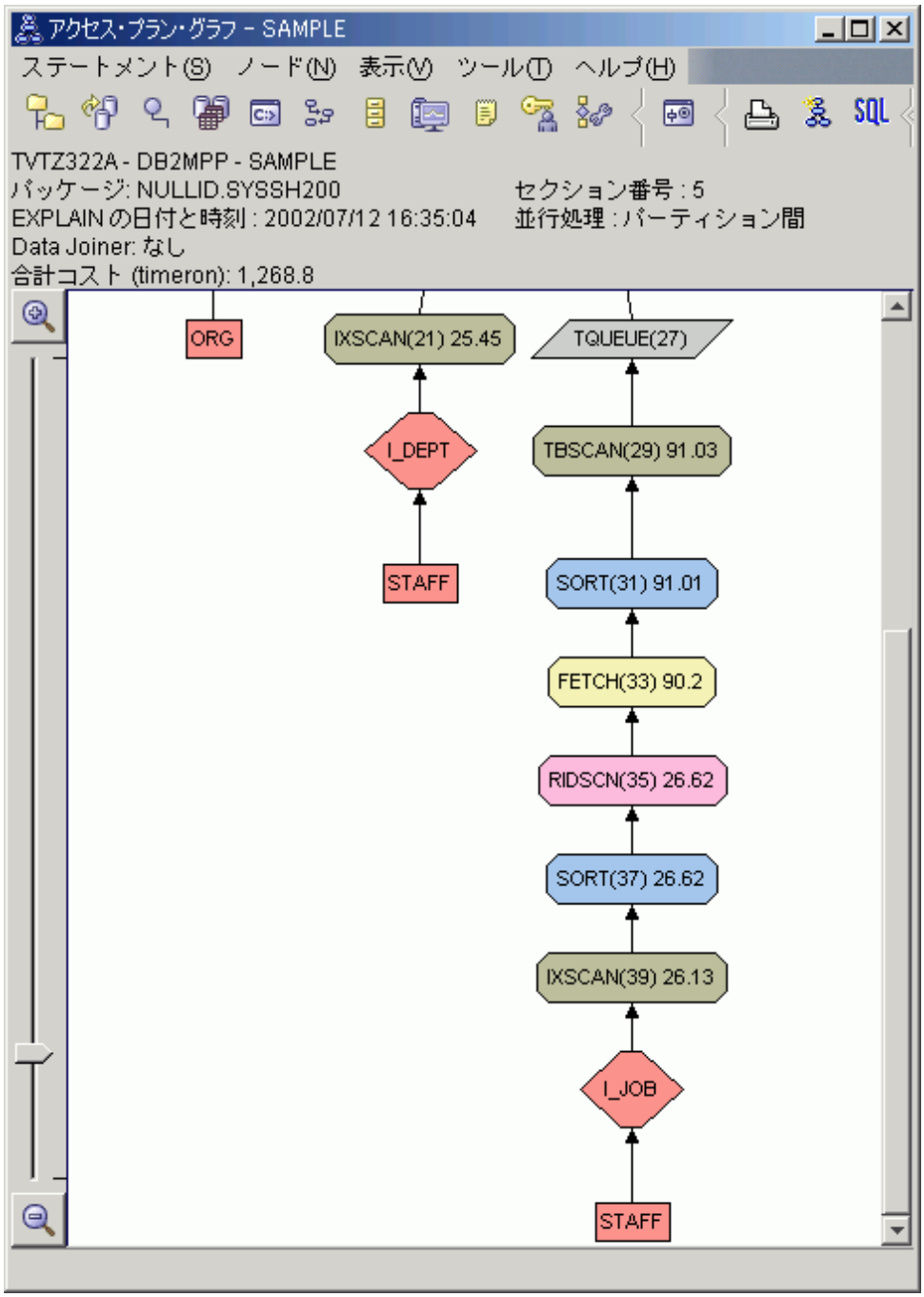
以下の質問に答えることにより、照会を改善する方法を学習します。

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、パーティションが追加または除去されるときに、主としてパーティション・データベース環境内でデータを移動するために実行できるデータベース管理操作です。この操作の目的は通常、ストレージ・スペースの使用のバランスを取ること、データベースのシステム・パフォーマンスを高めること、または他のシステム要件を満たすことです。

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 組み込みプロシージャー
- STEPWISE_REDISTRIBUTE_DBPG 組み込みプロシージャー
- 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、複数パーティション・データベースのパーティション・グループ内に表が作成されるときに自動的に生成されます。 CREATE TABLE または ALTER TABLE ステートメントを使用して、明示的に分散キーを定義することもできます。データの再配分時のデフォルトでは、特定のデータベース・パーティション・グループ内の表ごとに、データベース・パーティション間で表データが均等に分割されて再配分されます。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。

ログ記録リカバリー可能再配分と最小ログ記録ロールフォワード・リカバリー不可再配分との比較

REDISTRIBUTE DATABASE PARTITION GROUP コマンドまたは ADMIN_CMD 組み込みプロシージャーを使用して、データ再配分を実行する際、データ再配分の方式を、次の 2 つから選ぶことができます。1 つは、ログ記録リカバリー可能再配分で、もう 1 つは、最小ログ記録ロールフォワード・リカバリー不可再配分です。後者の方式は、**NOT ROLLFORWARD RECOVERABLE** コマンド・パラメーターを使用して指定できます。

容量が増えるシナリオで、ロード・バランシング中またはパフォーマンス調整中にデータ再配分を行うと、貴重な保守時間枠の時間、相当量の計画時間、ログ・スペースや余分のコンテナー・スペースが必要になって、大きなコストが発生する可能性があります。再配分の方式を選択する基準は、リカバリー可能性とスピードのどちらを優先するかです。

- ログ記録リカバリー可能再配分の方式を使用すると、どのような割り込み、エラー、その他の業務上の必要が発生した場合にもデータベースをリカバリーできるようにするため、すべての行移動の詳細なロギングが実行されます。
- ロールフォワード・リカバリー不可再配分の方式を使用すると、パフォーマンスが向上します。これは、データが一括して移動されることにより、挿入および削除操作のためのログ・レコードが不要になるためです。

従来、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、後者の方式が特に有益です。

ほとんどの場合、ロールフォワード・リカバリー不可再配分の方式がベスト・プラクティスです。その理由は、データ再配分が短時間で完了し、エラーが起こりにくく、システム・リソースの消費が少ないためです。結果として、データ再配分を実行する際の合計コストが減るので、他のビジネス・オペレーションのために時間とリソースが解放されます。

最小ログ記録ロールフォワード・リカバリー不可再配分

REDISTRIBUTE DATABASE PARTITION GROUP コマンドが発行されて **NOT ROLLFORWARD RECOVERABLE** パラメーターが指定された場合、移動される各行のログ・レコードの書き込みを最小にする、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。

最小ログ記録ロールフォワード・リカバリー不可再配分の方式を選択した場合にのみ使用可能なフィーチャー、およびオプション・パラメーターもあります。例えば、この再配分の方式はデフォルトでデータベースを静止し、事前チェックを実行して、前提条件が満たされていることを確認します。また、オプションで、再配分操作の一部として、索引の再作成および表統計の収集を指定することもできます。

これらのタスクを組み合わせて自動化すると (組み合わせて自動化しないと手動タスクになる)、エラーが起きにくくなり、速度が上がり、効率も上がり、操作を制御できる割合が増えます。

ロールフォワード・リカバリー不可再配分の方式は、表を自動的に再編成し、これによってディスク・スペースが解放されます。この表の再編成により、再配分操作にパフォーマンス・コストが追加されることはありません。クラスタリング索引を持つ表では、再編成時にクラスタリングの保守は試行されません。完全なクラスタリングが必要であれば、データ再配分の完了後に、クラスタリング索引を持つ表に対して **REORG TABLE** コマンドを実行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 (MDC) 表の場合、再編成では表のクラスタリングが保守されて、未使用ブロックが再使用のために解放されます。ただし、再配分後の表の合計サイズは変わらないように見えます。

注: このタイプの再配分操作によるロールフォワードにより、再配分されたすべての表が無効とマークされることになるので、再配分操作の完了時に個々の影響を受ける表スペースかデータベース全体のバックアップを取ることは重要です。そうした表はドロップのみが可能です。つまり、それらの表の中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 状態に設定します。この状態は、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ます。再配分操作の後に作成したバックアップがあれば、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ありません。

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 (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が作成される時間も含む) に、ユーザーにそのデータベースの表への更新を認めると、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、更新を実行しないようにすること。

- **QUIESCE DATABASE** コマンド・パラメーターを YES に設定して再配分操作を実行すること。その場合でも、静止状態のデータベースにアクセスを許可されているアプリケーションやユーザーが、更新を行っていないことを確認する必要があります。
- 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。

再配分操作中にデータベースへの更新を許可することについて、ユーザーはそうした更新が適切かどうかを、データベースのリストア後に必要に応じて更新を反復できるかどうかに基づいて決定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。

ログ記録リカバリー可能再配分

この方式は、**REDISTRIBUTE DATABASE PARTITION GROUP** コマンドのオリジナルのデフォルト・バージョンで、標準の SQL 挿入および削除を使用してデータを再配分します。すべての行移動の詳細なログが記録されるため、**RESTORE DATABASE** コマンドを使用してデータベースをリストアした後、**ROLLFORWARD DATABASE** コマンドを使用してすべての変更をロールフォワードすることにより、データベースをリカバリーできます。

データを再配分した後のソース表には空白が含まれています。これは、行が削除されて、新規データベース・パーティションに送られたためです。この空白を解放するには、表を再編成する必要があります。表を再編成するためには、再配分完了してから、別の操作を使用する必要があります。この方式のパフォーマンスを向上させるには、再配分の完了後に、索引をドロップしてから再作成します。

データ再配分の前提条件

データベース・パーティション・グループ内の表集合に対してデータの再配分を正常に実行するには、特定の前提条件に従う必要があります。

必須の前提条件を以下に示します。

- サポートされているデータ再配分インターフェースの選択項目からデータの再配分を実行する権限。
- システム・アクティビティーが少ない期間中に、再配分操作を実行するための相当の時間があること。

- データの再配分操作の一部として再配分されるデータの入ったすべての表は、NORMAL 状態でなければならない。例えば、表が LOAD PENDING 状態またはその他のアクセス不能な表のロード状態であってはなりません。表の状態を確認するには、データベース・パーティション・グループの各パーティションへの接続を確立し、**LOAD QUERY** コマンドを発行します。このコマンドの出力に、表の状態についての情報が含まれています。**LOAD QUERY** コマンドの資料は、それぞれの表の状態の意味、および表をある状態から別の状態へ変更する方法について説明しています。
- 再配分されるデータベース・パーティション内のすべての表は、分散キーで定義されている必要がある。新規のデータベース・パーティションが単一パーティション・システムに追加される場合、パーティション内のすべての表が分散キーを持つまで、データの再配分を実行することはできません。分散キーを含まない定義を持つ CREATE TABLE ステートメントを使用して作成された表の場合、データの再配分の前に ALTER TABLE ステートメントを使用して表を変更し、分散キ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・グループに含まれている複製されたマテリアライズ照会表は、データを再配分する前にドロップする必要がある。マテリアライズ照会表の定義のコピーを保管して、データの再配分の完了後に、それらを再作成できるようにします。
- 均等でない再配分を行う場合、再配分インターフェースに対するパラメーターに使用するターゲット分散マップとして分散マップを作成する必要がある。
- **BACKUP DATABASE** コマンドを使用して、データベースのバックアップを作成する必要がある。このバックアップは必須の前提条件ではありませんが、実行することを強くお勧めします。
- カタログ・データベース・パーティションからデータベースへの接続を確立する必要がある。
- データ再配分中またはその後にすべての索引を再作成するために十分なスペースが使用可能でなければならない。 **INDEXING MODE** コマンド・パラメーターは索引がいつ再作成されるかに影響を与えます。
- **NOT ROLLFORWARD RECOVERABLE** コマンド・パラメーターを指定する場合は、IBM サービスが問題判別に使用できるよう、制御ファイルに状況情報を書き込むための十分なスペースが使用可能でなければならない。制御ファイルは以下のパスに生成されます。データ再配分操作が完了したときに、このファイルを手動で削除する必要があります。
  - 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*¥

  以下の数式を使用して、制御ファイルのスペース所要量をバイト単位で計算できます。

  ```
  (number of pages for all tables in the database partition group) * 64 bytes
  + number of LOB values in the database partition group) * 600 bytes
  ```

  データベース・パーティション・グループ内の *LOB 値の数* を見積もるには、表内の LOB 列の数を加算し、これを最大の表の行数で乗算してください。

- **NOT ROLLFORWARD RECOVERABLE** コマンド・パラメーターを指定して**いない**場合、データの再配分中に実行される INSERT 操作および DELETE 操作に関連したログ項目を収容できるように、十分なログ・ファイル・スペースが必要です。そうでない場合、データの再配分は中断されるか失敗します。

util_heap_sz データベース構成パラメーターは、データベース・パーティション間でのデータ移動を処理する上で重要です。したがって、再配分操作を行う間は、できるだけ多くのメモリーを **util_heap_sz** に割り振ってください。また、再配分操作の一環として索引が再作成される場合は、十分な **sortheap** も必要です。再配分のパフォーマンスを向上させるために、必要に応じて **util_heap_sz** および **sortheap** データベース構成パラメーターの値を引き上げてください。

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。

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。
  - 同じデータベース・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** コマンド・パラメーターが指定された **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** コマンド・パラメーターが指定される場合、以下を検討します。
  - 再配分中に行われるデータ再配分の変更は、ロールフォワード・リカバリー可能ではありません。
  - データベースがリカバリー可能である場合、表スペース内の最初の表へのアクセス後、表スペースは BACKUP PENDING 状態になります。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** パラメーターに値 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。
- 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。ただし、パーティション表の場合、次の両方とも該当しなければ、データ・パーティション表の範囲間のデータの移動は制限されます。
  – 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** パラメーターと適切な表リストを指定して **REDISTRIBUTE DATABASE PARTITION GROUP** コマンドを実行します。

データ再配分が必要かどうかの判別

データベース・パーティション・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するのに役立ちます。現行のデータ配分に関する詳細情報は、データの配分方法を指定するカスタム分散マップを作成する際にも使用できます。

このタスクについて

データベース・パーティション・グループに新規データベース・パーティションを追加した場合、またはデータベース・パーティション・グループから既存のデータベース・パーティションをドロップした場合は、すべてのデータベース・パーティション間でデータのバランスを取るためにデータの再配分を実行します。

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。

手順

データの再配分が必要かどうかを判別するには、以下のようにします。

1. データベース・パーティション・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の配分に関する情報を取得します。

   データベース・パーティション・グループ内の最大の表 (あるいは代表的な表) に対して、次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。

   この照会の出力は、各データベース・パーティションにある *table_name* のレコード数を示します。データベース・パーティション間のデータの配分が望みどおりでない場合は、次のステップに進んでください。

2. ハッシュ・パーティション間のデータの配分に関する情報を取得します。

   前のステップで使用したのと同じ *column_name* と *table_name* を指定して、次の照会を実行します。

   ```
   SELECT HASHEDVALUE(column_name), COUNT(*) FROM table_name
         GROUP BY HASHEDVALUE(column_name)
         ORDER BY HASHEDVALUE(column_name) DESC
   ```

   REDISTRIBUTE DATABASE PARTITION GROUP コマンドの **USING DISTFILE** パラメーターを指定すると、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。配分ファイルのフォーマットの説明は、**REDISTRIBUTE DATABASE PARTITION GROUP** コマンド・リファレンスを参照してください。

3. オプション: データの再配分を必要とする場合、その操作をシステム保守の機会に行うように計画することができます。

   USING DISTFILE パラメーターを指定すると、**REDISTRIBUTE DATABASE PARTITION GROUP** コマンドは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・グループの新規パーティション・マップを生成します。この操作により、データベース・パーティション間のデータの配分が均等になります。

   均等な配分を望まない場合は、再配分操作のための独自のターゲット・パーティション・マップを構成できます。ターゲット・パーティション・マップは、**REDISTRIBUTE DATABASE PARTITION GROUP** コマンドの **USING TARGETMAP** パラメーターを使用して指定できます。

タスクの結果

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。

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

REDISTRIBUTE DATABASE PARTITION GROUP コマンドは、データ再配分を行うための推奨インターフェースです。

手順

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. オプション: データベースのバックアップを実行します。 **BACKUP DATABASE** コマンドを参照してください。

   ロールフォワード・リカバリー可能でないデータ再配分を行う前にデータベースのバックアップ・コピーを作成することを強くお勧めします。
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 ステートメントを参照してください。
3. **REDISTRIBUTE DATABASE PARTITION GROUP** コマンドを出す。

   注: DB2 データベース製品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することを指定します。 **UNIFORM** は配分タイプが指定されない場合のデフォルトであるため、その他の配分タイプが指定されていない場合には、このオプションは省略して構いません。

   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歪度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・データ再分散マップを使用すること、およびターゲット再分散マップが含まれる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。 コマンドが完了してデータ再配分が正常に行われた場合は、以下のアクションを実行します。

- 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あり、**REDISTRIBUTE DATABASE PARTITION GROUP** コマンドで **NOT ROLLFORWARD RECOVERABLE** コマンド・パラメーターが使用された場合のみ、表スペースは BACKUP PENDING 状態になります。
- 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。
- 以下の状態が該当する場合は、**RUNSTATS** コマンドを実行します。
  - **REDISTRIBUTE DATABASE PARTITION GROUP** コマンドで **STATISTICS NONE** コマンド・パラメーターが指定された、または **NOT ROLLFORWARD RECOVERABLE** コマンド・パラメーターが省略された。これらの状態は両方とも、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。
  - 統計プロファイルを保有する表がデータベース・パーティション・グループにある。

  RUNSTATS コマンドは、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使用するデータ分散統計を収集します。
- **NOT ROLLFORWARD RECOVERABLE** コマンド・パラメーターを指定した場合、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*¥

タスクの結果

データの再配分が完了し、データの再配分処理に関する情報が再配分ログ・ファイルにあります。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作成し、データを再配分するには、**REDISTRIBUTE DATABASE PARTITION GROUP** コマンドを発行するか、または sqludrdt API を呼び出します。

始める前に

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、SYSCTRL、または DBADM 権限が必要です。

手順

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。

- コマンド行プロセッサー (CLP) で **REDISTRIBUTE DATABASE PARTITION GROUP** コマンドを発行します。
- ADMIN_CMD プロシージャーを使用して **REDISTRIBUTE DATABASE PARTITION GROUP** コマンドを発行します。
- sqludrdt API を呼び出します。

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、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。**NOT ROLLFORWARD RECOVERABLE** コマンド・パラメーターを指定する場合、ログ・スペース所要量はそれほど気にする必要はありません。このタイプのデータ再配分のときは最小限のロギングが行われるからです。

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。

データの再配分がロールフォワード・リカバリー可能なサポートされるインターフェースから再配分が実行される場合は、以下の点に注意してください。

- 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 値が含まれる場合のような、不均等なデータ配分について考えてください。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 1024 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。例えば、見積もりがアクティブ・ログの限界よりどれだけ大きいかに比例するいくつかのステップで STEPWISE_REDISTRIBUTE_DBPG プロシージャーを使用します。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。

データの再配分がロールフォワード・リカバリー可能でないサポートされるインターフェースから再配分が実行される場合は、以下の点に注意してください。

- 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。この動作により、ログ・ファイルのスペース所要量が大幅に小さくなります。ただし、データベース・ロールフォワード・リカバリーでこのオプションを使用した場合、再配分操作のログ・レコードをロールフォワードできず、ロールフォワード操作の一部として処理される表はすべて UNAVAILABLE 状態のままになります。
- 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んでいる場合、データの各行にログ・レコードが作成されるので、データ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 と予測されます。

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、後でエラー・リカバリーを実行するのに使用できるイベント・ログ・ファイルに記録されます。

データの再配分が実行されると、処理される各表に関する情報が、1 つのペアになっているイベント・ログ・ファイルに記録されます。イベント・ログ・ファイルの名前は、*database-name.database-partition-group-name.timestamp*.logと *database-name.database-partition-group-name.timestamp*です。

ログ・ファイルは、以下のように配置されます。

- Linux および UNIX オペレーティング・システムでは *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。

```
SAMPLE.IBMDEFAULTGROUP.2012012620240204
SAMPLE.IBMDEFAULTGROUP.2012012620240204.log
```

これらのファイルは、IBMDEFAULTGROUP というデータベース・パーティション・グループの SAMPLE というデータベースに対する再配分操作のためのものです。ファイルは 2012 年 1 月 26 日 8:24 PM (現地時間) に作成されました。

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判別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。

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、組み込みプロシージャーを使用して実行できます。

手順

STEPWISE_REDISTRIBUTE_DBPG プロシージャーを使用してデータベース・パーティション・グループを再配分するには、以下のようにします。

1. ANALYZE_LOG_SPACE プロシージャーを使用して、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 プロシージャーは、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 プロシージャーを使用して、特定の表のデータ配分ファイルを作成します。

   GENERATE_DISTFILE プロシージャーは、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 プロシージャーを使用して、データベース・パーティション・グループの段階的再配分プランの内容を作成し、報告します。
4. GET_SWRD_SETTINGS プロシージャーおよび SET_SWRD_SETTINGS プロシージャーを使用して、特定の表のデータ配分ファイルを作成します。

   GET_SWRD_SETTINGS プロシージャーは、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 プロシージャーは、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 プロシージャーは、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。

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
# ------------------------------------------------------------------------------------
db2start
db2 -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セルフチューニング・メモリー

パーティション・データベース環境でセルフチューニング・メモリー・フィーチャーを使用する場合、このフィーチャーがシステムを適切に調整するかどうかを決定するいくつかの要因があります。

パーティション・データベースでセルフチューニング・メモリーが使用可能にされると、単一のデータベース・パーティションがチューニング・パーティションとして指定され、メモリーのチューニングに関するすべての決定は、そのデータベース・パーティションのメモリーおよびワークロード特性に基づいて行われます。チューニングに関する決定がそのパーティションで行われると、メモリーの調整が他のデータベース・パーティションに分散され、すべてのデータベース・パーティションが同様の構成を保守することが保証されます。

この単一チューニング・パーティション・モデルでは、すべてのデータベース・パーティションでメモリー要件が類似している場合のみ、このフィーチャーが使用されると想定します。以下に示すガイドラインは、パーティション・データベースでセルフチューニング・メモリーを使用可能にするかどうかを判別する際に使用します。

パーティション・データベースでセルフチューニング・メモリーが推奨される場合

すべてのデータベース・パーティションでメモリー要件が類似していて、それらのパーティションが類似のハードウェア上で稼働している場合、変更を加えずにセルフチューニング・メモリーを使用可能にすることができます。これらのタイプの環境では、以下の特性が共通しています。

- すべてのデータベース・パーティションが同一のハードウェア上に存在し、複数の論理データベース・パーティションが複数の物理データベース・パーティションに均一に分散されている
- 完璧かほぼ完璧に分散されているデータがある
- 複数のデータベース・パーティションでワークロードが均一に分散されている。つまり、他のデータベース・パーティションと比べて、1 つ以上のヒープに関するメモリー要件が高いパーティションは存在しません。

そのような環境では、すべてのデータベース・パーティションが同等に構成されていると、セルフチューニング・メモリーはシステムを適切に構成します。

パーティション・データベースでセルフチューニング・メモリーが制限付きで推奨される場合

環境内のほとんどのデータベース・パーティションでメモリー要件が類似していて、それらのパーティションが類似のハードウェア上で稼働している場合は、初期構成にいくらかの注意を払うかぎり、セルフチューニング・メモリーを使用することができます。これらのシステムには、データ用のデータベース・パーティション

のセット 1 つと、少数のコーディネーター・パーティションおよびカタログ・パーティションのセットが存在する場合があります。このような環境では、コーディネーター・パーティションとカタログ・パーティションを、データを含むデータベース・パーティションとは異なった構成にすると便利です。

データを含むすべてのデータベース・パーティションでセルフチューニング・メモリーを使用可能にし、これらのデータベース・パーティションの 1 つをチューニング・パーティションとして指定する必要があります。加えて、コーディネーター・パーティションとカタログ・パーティションが異なる構成になっている可能性があるため、セルフチューニング・メモリーをこれらのパーティションで使用不可にする必要があります。コーディネーター・パーティションおよびカタログ・パーティションでセルフチューニング・メモリーを使用不可にするには、これらのパーティションで **self_tuning_mem** データベース構成パラメーターを OFF に設定してください。

パーティション・データベースでセルフチューニング・メモリーが推奨されない場合

各データベース・パーティションのメモリー要件が異なっている場合や、さまざまなデータベース・パーティションが大幅に異なるハードウェア上で稼働している場合では、セルフチューニング・メモリー・フィーチャーを使用不可にすることをお勧めします。このフィーチャーを使用不可にするには、すべてのパーティションで **self_tuning_mem** データベース構成パラメーターを OFF にします。

異なるデータベース・パーティションのメモリー要件の比較

複数の異なるデータベース・パーティションのメモリー要件が十分に類似しているかどうかを判別する最良の方法は、スナップショット・モニターを参照することです。以下のスナップショット・エレメントがすべてのデータベース・パーティションで類似していれば (差が 20% 以下)、それらのデータベース・パーティションのメモリー要件はかなり類似していると見なせます。

以下のデータを収集するには、コマンド `get snapshot for database on <dbname>` を発行します。

```
Locks held currently                        = 0
Lock waits                                  = 0
Time database waited on locks (ms)          = 0
Lock list memory in use (Bytes)             = 4968
Lock escalations                            = 0
Exclusive lock escalations                  = 0

Total Shared Sort heap allocated            = 0
Shared Sort heap high water mark            = 0
Post threshold sorts (shared memory)        = 0
Sort overflows                              = 0

Package cache lookups                       = 13
Package cache inserts                       = 1
Package cache overflows                     = 0
Package cache high water mark (Bytes)       = 655360

Number of hash joins                        = 0
Number of hash loops                        = 0
Number of hash join overflows               = 0
Number of small hash join overflows         = 0
Post threshold hash joins (shared memory)   = 0
```

```
Number of OLAP functions                   = 0
Number of OLAP function overflows          = 0
Active OLAP functions                      = 0
```

以下のデータを収集するには、コマンド `get snapshot for bufferpools on <dbname>` を発行します。

```
Buffer pool data logical reads             = 0
Buffer pool data physical reads            = 0
Buffer pool index logical reads            = 0
Buffer pool index physical reads           = 0
Total buffer pool read time (milliseconds) = 0
Total buffer pool write time (milliseconds)= 0
```

パーティション・データベース環境でのセルフチューニング・メモリーの使用

パーティション・データベース環境でセルフチューニング・メモリーが有効な場合、データベース・パーティションが 1 つ (チューニング・パーティション と呼ばれる) 存在します。これは、メモリー構成をモニターして、構成変更があればそれを他のすべてのデータベース・パーティションに伝搬し、すべての関連するデータベース・パーティション間で構成の整合性を保持します。

チューニング・パーティションは、パーティション・グループのデータベース・パーティションの数およびバッファー・プールの数などの、幾つかの特性に基づいて選択されます。

- チューニング・パーティションとして現在指定されているデータベース・パーティションを判別するには、**ADMIN_CMD** プロシージャーを以下のように呼び出します。

  ```
  CALL SYSPROC.ADMIN_CMD('get stmm tuning dbpartitionnum')
  ```

- チューニング・パーティションを変更するには、**ADMIN_CMD** プロシージャーを以下のように呼び出します。

  ```
  CALL SYSPROC.ADMIN_CMD('update stmm tuning dbpartitionnum <partitionnum>')
  ```

  チューニング・パーティションは、非同期で更新されるか、次回のデータベース始動時に更新されます。メモリー・チューナーが自動的にチューニング・パーティションを選択するようにするには、*partitionnum* 値に -1 を入力します。

パーティション・データベース環境でのメモリー・チューナーの開始

パーティション・データベース環境では、メモリー・チューナーが開始されるのは **ACTIVATE DATABASE** コマンドによって明示的にデータベースがアクティブにされる場合に限ります。セルフチューニング・メモリーでは、すべてのパーティションがアクティブである必要があるからです。

指定のデータベース・パーティションでセルフチューニング・メモリーを無効にする

- データベース・パーティションのサブセットでセルフチューニング・メモリーを無効にするには、こうしたデータベース・パーティションに対して **self_tuning_mem** データベース構成パラメーターを OFF に設定します。

- 特定のデータベース・パーティションの構成パラメーターによって制御された、メモリー・コンシューマーのサブセットに対してセルフチューニング・メモリーを無効にするには、関連する構成パラメーターの値またはバッファー・プール・サイズの値を MANUAL またはそのデータベース・パーティションの特定の値に設定します。セルフチューニング・メモリーの構成パラメーターの値は、すべての稼働パーティション間で整合させることをお勧めします。
- 特定のデータベース・パーティションの特定のバッファー・プールでセルフチューニング・メモリーを無効にするには、ALTER BUFFERPOOL ステートメントを発行し、サイズ値と、セルフチューニング・メモリーを無効にするパーティションを指定します。

  特定のデータベース・パーティション上のバッファー・プールのサイズを指定する ALTER BUFFERPOOL ステートメントは、そのバッファー・プールに関する例外項目を SYSCAT.BUFFERPOOLDBPARTITIONS カタログ・ビューに作成します (または既存の項目を更新します)。バッファー・プールの例外項目が存在する場合に、デフォルトのバッファー・プール・サイズが AUTOMATIC に設定されていると、そのバッファー・プールはセルフチューニング操作に関与しません。例外項目を除去して、バッファー・プールをセルフチューニング用に有効にするには、以下のようにします。

  1. ALTER BUFFERPOOL ステートメントを発行して、バッファー・プール・サイズを特定の値に設定することにより、このバッファー・プールのセルフチューニングを使用不可にします。
  2. 別の ALTER BUFFERPOOL ステートメントを発行し、このデータベース・パーティション上のバッファー・プールのサイズをデフォルトに設定します。
  3. 別の ALTER BUFFERPOOL ステートメントを発行して、バッファー・プール・サイズを AUTOMATIC に設定することにより、このバッファー・プールのセルフチューニングを使用可能にします。

非均一環境でセルフチューニング・メモリーを有効にする

データをすべてのデータベース・パーティションに均一に配分し、各パーティションで実行されるワークロードが同様のメモリー要求量を持つのが理想的です。データ配分に偏りがあり、1 つ以上のデータベース・パーティションに他のデータベース・パーティションよりもかなり多い (または非常に少ない) データが含まれる場合、これらの変則的なデータベース・パーティションをセルフチューニング用に有効にするべきではありません。メモリー所要量がデータベース・パーティション間で偏っている場合にも同じことが当てはまります。これが生じる可能性があるのは、例えば、リソースを多く使用するソートが 1 つのパーティションでのみ実行される場合、または一部のデータベース・パーティションが別のハードウェアと関連付けられており、他のデータベース・パーティションよりも使用可能メモリーが多い場合です。このタイプの環境にある一部のデータベース・パーティションでは、セルフチューニング・メモリーを有効にすることができます。偏りがある環境でメモリーのセルフチューニングを活用するには、同様のデータおよびメモリー所要量のデータベース・パーティションのセットを識別し、セルフチューニング用にそれらを有効にします。残りのパーティションのメモリーは手動で構成してください。

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 9.7 にアップグレードする場合、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、データベースのアップグレード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 9.7 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、「*DB2 バージョン 10.1 へのアップグレード*」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。アップグレード後のデータベースに対するデータベース構成の更新またはリセット要求は、デフォルトではすべてのデータベース・パーティションに適用されます。

既存の更新コマンド・スクリプトまたはリセット・コマンド・スクリプトの場合、上記の同じ規則がすべてのデータベース・パーティションに適用されます。 **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 以降の命令を使用してください。バージョン 9.5 より前の動作を使用する場合、**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 以降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

パーティション・データベース環境変数を使用すると、許可、フェイルオーバー、ネットワーク動作などの、パーティション・データベース環境におけるデフォルト動作を制御できます。

DB2CHGPWD_EEE

- オペレーティング・システム: AIX、Linux、および Windows 上の DB2 ESE
- 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
- デフォルト =YES。値:
  - FCM_MAXIMIZE_SET_SIZE:[YES|TRUE|NO|FALSE]。FCM_MAXIMIZE_SET_SIZE のデフォルト値は YES です。
  - FCM_CFG_BASE_AS_FLOOR: [YES|TRUE|NO|FALSE]。FCM_CFG_BASE_AS_FLOOR のデフォルト値は NO です。
- **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 4 GB のスペースを事前割り振りできます。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_FCM_SETTINGS レジストリー変数を FCM_CFG_BASE_AS_FLOOR オプションとともに設定すると、*fcm_num_buffers* および *fcm_num_channels* のデータベース・マネージャー構成パラメーターの下限となる基本値を設定できます。 FCM_CFG_BASE_AS_FLOOR オプションが YES または TRUE に設定されている場合、前述のパラメーターの設定値が AUTOMATIC で、かつ初期値または開始時の値があるなら、DB2 がそれより低い値にチューニングすることはありません。

DB2_FORCE_OFFLINE_ADD_PARTITION

- オペレーティング・システム: すべて
- デフォルト = FALSE。値: FALSE または TRUE
- この変数を使用すると、データベース・パーティション・サーバーの追加操作をオフラインで実行するように指定することができます。デフォルト設定の FALSE は、データベースをオフラインにしなくても DB2 データベース・パーティション・サーバーを追加できることを示します。しかし、操作をオフラインで実行したい場合、または何らかの制限のために、データベースがオンラインになっているときにデータベース・パーティション・サーバーの追加ができない場合は、**DB2_FORCE_OFFLINE_ADD_PARTITION** を TRUE に設定してください。この変数が TRUE に設定されると、新しい DB2 データベース・パーティション・サーバーはバージョン 9.5 およびそれ以前のバージョンの動作に従って追加されます。つまり、新しいデータベース・パーティション・サーバーは、インスタンスをシャットダウンして再始動するまで、そのインスタンスには認識されません。

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
- **DB2_NUM_FAILOVER_NODES** を設定して、フェイルオーバーの際にマシン上に開始する必要があるであろう追加のデータベース・パーティションの数を指定します。

  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 2 に設定されています。

START DBM 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、その障害が発生したマシンの 2 つのデータベース・パーティションをもう一方のマシンで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、DB2 メンバー間でネットワーク接続が確立されるまでの許容経過時間 (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 DB2 pureScale サーバー (複数の DB2 メンバーを含む)

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・サーバーまたは DB2 pureScale 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 [895 - 65300]

64 ビット・プラットフォーム

Automatic [895 - 524288]

- ローカル・クライアントとリモート・クライアントを持つデータベース・サーバー: 1024
- ローカル・クライアントを持つデータベース・サーバー: 895
- ローカル・クライアントとリモート・クライアントを持つパーティション・データベース・サーバーまたは DB2 pureScaleデータベース・サーバー: 4096

高速コミュニケーション・マネージャー (FCM) バッファーは、デフォルトでメンバー間通信とメンバー内通信の両方に使用されます。

重要: **fcm_num_buffers** パラメーターのデフォルト値は、データベースの初回作成後に、DB2 構成アドバイザーによって変更される場合があります。

fcm_num_buffers 構成パラメーターには、初期値と AUTOMATIC 値の両方を設定できます。パラメーターを AUTOMATIC に設定すると、FCM はリソース使用状況をモニターして、リソースを増やしたり、減らしたり (リソースが 30 分間使用されていない場合) することができます。リソースの増加量または減少量は、オペレーティング・システムによって異なります。 Linux オペレーティング・システムでは、バッファー数は開始値の 25% までしか増加させることができません。データベース・マネージャーがインスタンスを開始しようとしたときに指定バッファー数を割り振れない場合、インスタンスを開始できるまでその数が減らされます。

fcm_num_buffers パラメーターに特定の値と AUTOMATIC の両方を設定し、システム・コントローラー・スレッドによるリソース調整が指定値より下にならないようにする場合は、**DB2_FCM_SETTINGS** レジストリー変数の FCM_CFG_BASE_AS_FLOOR オプションを YES または TRUE に設定します。 **DB2_FCM_SETTINGS** レジストリー変数の値は動的に調整されます。

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** パラメーターの値に物理マシン上の論理ノード数を乗算することによって決定できます。使用する値を確認して、複数の論理ノードがある単一マシンまたは複数マシン上で割り振られる FCM バッファーの数を考慮してください。同一マシン上に複数の論理ノードがある場合、**fcm_num_buffers** 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。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、またはアプリケーションの複雑さによっては、システムがメッセージ・バッファーを使い果たす可能性があります。

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・サーバーまたは DB2 pureScale 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、または 2048 [128 - 120000]

UNIX 64 ビット・プラットフォーム

自動。開始値は 256、512、または 2048 [128 - 524288]

Windows 32 ビット

自動。開始値は 10000 [128 - 120000]

Windows 64 ビット

自動。開始値は 256、512、または 2048 [128 - 524288]

各種サーバーのデフォルトの開始値は、以下のとおりです。

- 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 2048 です。

高速コミュニケーション・マネージャー (FCM) バッファーは、デフォルトでメンバー間通信とメンバー内通信の両方に使用されます。非クラスター・データベース・システムが FCM サブシステムと **fcm_num_channels** パラメーターを使用できるようにするには、**intra_parallel** パラメーターを YES に設定する必要があります。

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、メンバー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 DB2 メンバー間でのネットワーク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

DB2 pureScale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 DB2 メンバー間で通信を確立する試みが失敗すると (例えば **conn_elapse** パラメーターで指定された値に達した場合)、**max_connretries** が DB2 メンバーに対して実行できる接続再試行の回数を指定します。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 - メンバー間最大時差

このパラメーターは、ノード構成ファイル内にリストされる DB2 pureScale環境のメンバー間で許可される最大時差を指定します。

構成タイプ
データベース・マネージャー

適用 ローカル・クライアントおよびリモート・クライアントを持つメンバー

パラメーター・タイプ
構成可能

デフォルト [範囲]

DB2 pureScale環境内
1 [1 - 1 440]

DB2 pureScale環境外
60 [1 - 1 440]

単位 分

各メンバーには、独自のシステム・クロックがあります。 2 つ以上のメンバー間のシステム・クロックの時差が周期的に検査されます。システム・クロック間の時差が **max_time_diff** パラメーターで指定された時間より大きい場合、警告が db2diag ログ・ファイルに記録されます。

DB2 pureScale環境の場合、メンバーが相互に同期から外れないようにするために、Network Time Protocol (NTP) のセットアップが必要になり、各メンバーで周期的に検査されます。 NTP デーモンが検出されない場合、警告が db2diag ログ・ファイルに記録されます。

SQLOGCTL.LFH ログ制御ファイルに保存されている仮想タイム・スタンプ (VTS) とシステム・クロックが比較されるパーティション・データベース環境では SQL1473N エラー・メッセージが返されます。 .LFH ログ制御ファイルのタイム・スタンプが

システム時刻より小さい場合、データベース・ログの時刻は VTS に設定されます。VTS とシステム・クロックが一致するまでこの設定になります。

DB2 データベース・マネージャーは協定世界時 (UTC) を使用するので、**max_time_diff** パラメーターを設定する際、異なる時間帯は考慮されません。 UTC 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 **START DBM** または **STOP DBM** コマンドに応答する必要があります。また、**ADD DBPARTITIONNUM** および **DROP 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

指定した時間内に **db2start** コマンドまたは **db2stop** コマンドに対して反応しないメンバーまたはノードは、複数メンバー/ノード・インスタンスの **db2start** または **db2stop** によって自動的に強制終了し、クリーンアップされます。診断メッセージは、データベース・マネージャー構成で定義される **diagpath**、またはそのデフォルト値 (例えば UNIX オペレーティング・システムの場合は `sqllib/db2dump/ $m`) に記録されます。

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・パーティションは自動的に強制終了し、クリーンアップされます。 **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**、または **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照会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。

注: デフォルト値は、データベースの初回作成後に、DB2 構成アドバイザーによって変更される場合があります。

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。
- **intra_parallel** 設定は、アプリケーション内で ADMIN_SET_INTRA_PARALLEL プロシージャーの呼び出しによってオーバーライドすることができます。 **intra_parallel** 設定と、ADMIN_SET_INTRA_PARALLEL プロシージャーによってアプリケーションで設定される値のどちらも、ワークロード定義で MAXIMUM DEGREE 属性を設定することによってワークロードでオーバーライドできます。

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 はシステムにより決定されることを意味する)

注: デフォルト値は、データベースの初回作成後に、DB2 構成アドバイザーによって変更される場合があります。

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・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 ステートメントのコンパイルの度合い

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・パーティションの追加

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・サーバーに追加します。

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

以下の権限のいずれか。

- SYSADM
- SYSCTRL

必要な接続

なし

API 組み込みファイル

```
sqlenv.h
```

API およびデータ構造構文

```
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。この定義は作成される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

この API を使用できるのは、データベース・パーティション・サーバーに 1 つのデータベースがあり、そのデータベースがパーティションの追加操作時にカタログされていない環境に追加される場合に限ります。この状況では、データベースはカタログされないため、パーティションの追加操作ではデータベースを認識しません。また、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のデータベース・パーティションを作成しません。新規データベース・パーティション・サーバー上でデータベース・パーティションに接続しようとすると、エラーが発生します。データベースを最初にカタログしてから、sqleaddn API を使用して、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のデータベース・パーティションを作成する必要があります。

この API は、複数のデータベースがあり、そのデータベースの少なくとも 1 つがパーティションの追加操作時にカタログされていない環境では使用してはなりません。この状況では、sqlecran API を使用して、パーティション追加操作時にカタログされなかったデータベースごとにデータベース・パーティションを作成します。アンカタログされたデータベースをそれぞれ最初にカタログしてから、sqlecran API を使用して、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のデータベース・パーティションを作成する必要があります。

新規データベース・パーティションを追加する前に、コンテナーを作成するだけの十分なストレージがあることを確認してください。

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。

注: 新規データベース・パーティションを追加する際に、アンカタログされたデータベースは認識されません。アンカタログされたデータベースは、新規データベース・パーティションには存在しません。新規データベース・パーティション上でデータベースに接続しようとすると、エラー・メッセージ SQL1013N が戻されます。

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。操作が完了したら、API をもう一度呼び出すことができます。

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るときに、ストレージ・グループのストレージ・パス定義を検索します。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

以下の権限のいずれか。

- 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 およびデータ構造構文

```
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

以下の権限のいずれか。

- 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 およびデータ構造構文

```
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。将来の利用のために予約済み。

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

以下の権限のいずれか。

- SYSADM
- SYSCTRL

API 組み込みファイル

```
sqlenv.h
```

API およびデータ構造構文

```
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

バージョン 9.7 からは、この API は使用すべきではありません。db2GetRowPartNum (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) API を使用して、特定の行のデータベース・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。

sqlugrpn API を呼び出す場合に **DB2_PMAP_COMPATIBILITY** レジストリー変数が OFF に設定されていると、エラー・メッセージ SQL2768N が戻されます。

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 およびデータ構造構文

```
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必要があります。

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 が呼び出される対応するシステムのコード・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・グループ内のパーティション間でデータを再配分します。このコマンドはデータベース・パーティション・グループにあるすべてのオブジェクトに影響を及ぼし、1 つのオブジェクトだけに限定することはできません。

このコマンドを発行できるのは、カタログ・データベース・パーティションからのみです。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

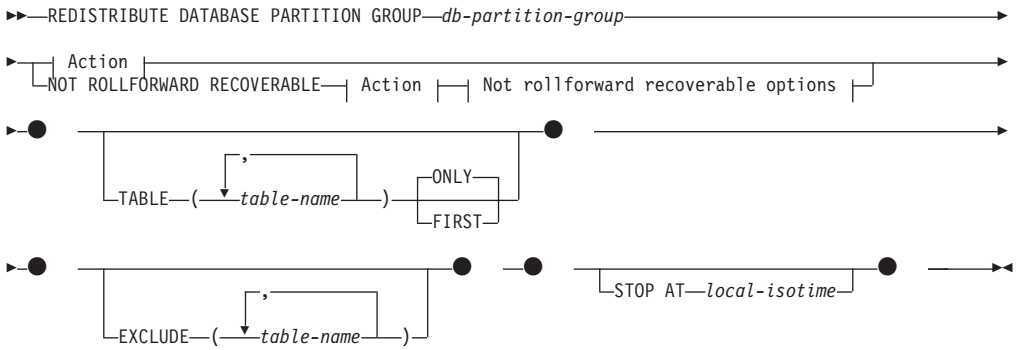
以下のいずれかの権限が必要です。

- SYSADM
- SYSCTRL
- DBADM

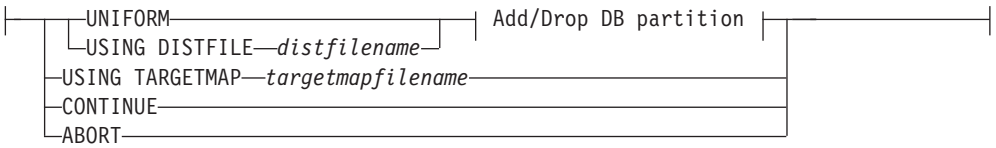
さらに、以下の権限のグループのいずれかも必要です。

- 再配分されるデータベース・パーティション・グループ内のすべての表に対する DELETE、INSERT、および SELECT 特権。
- DATAACCESS 権限

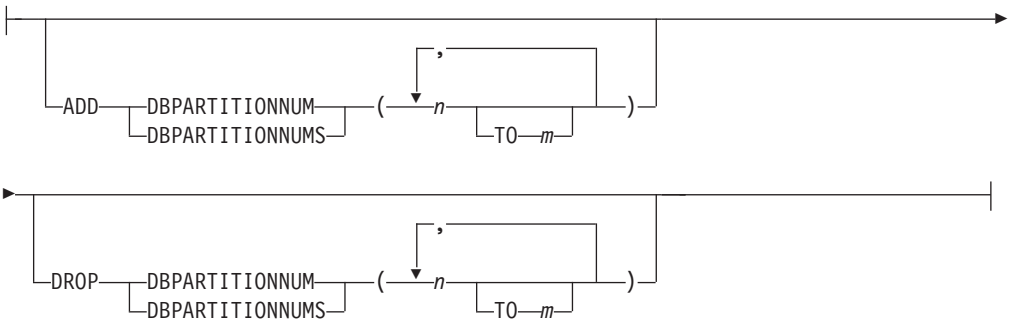
コマンド構文



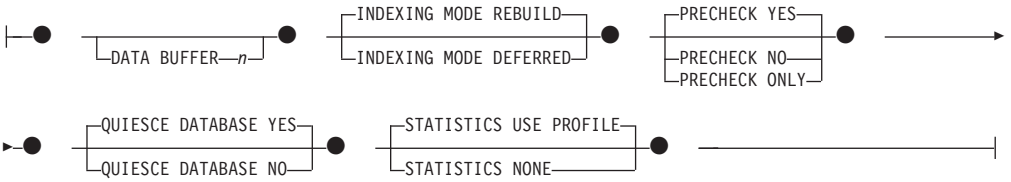
アクション:



Add/Drop DB partition:



Not rollforward recoverable 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、現在再配分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。

- **REDISTRIBUTE DATABASE PARTITION GROUP** コマンドを **NOT ROLLFORWARD RECOVERABLE** オプションとともに使用する場合、再配分操作は XML 列の入った表に対して **INDEXING MODE DEFERRED** オプションを使用します。表に XML 列が含まれていない場合、再配分操作はコマンドの発行時に指定された索引付けモードを使用します。

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、中断やエラーが起きた場合、またはほかにビジネス上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・パーティション間で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、32 768 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ます。このマップは、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おいてデータをできるだけ均一に再配分するために使用されます。再配分した後は、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・パーティションの重みが、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、そのデータベース・パーティションにマップするすべてのハッシュ・パーティションの重みの合計です)。

例えば、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、32 768 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、4 294 967 295 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。

targetmapfilename には 32 768 個の整数を格納する必要があり、それぞれは有効なデータベース・パーティション番号を表します。各行の番号によって、ハッシュ値が特定のデータベース・パーティションにマップされます。つまり、行 *X* に値 *Y* が入っている場合、HASHEDVALUE() が *X* のすべてのレコードはデータベース・パーティション *Y* に配置されます。

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:

1. 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を持つ既存のパーティション内の、対応する表スペースのコンテナー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。
2. **ADD DBPARTITIONNUMS** パラメーターを指定すると、データ再配分によって、すべての新規データベース・パーティションに表スペースが作成される可能性があります。

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*
再配分処理する表の順番を指定します。

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、**REDISTRIBUTE CONTINUE** コマンドによって後で処理できます。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。

EXCLUDE *tablename*
再配分処理をしない表を指定します。例えば、データ再配分の要件を表が満たすように構成できるようになるまで、その表を一時的に対象外にできます。対象から外した表は、**REDISTRIBUTE CONTINUE** コマンドを使用して後で処理できます。

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。

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 ページの数を指定します。このコマンド・パラメーターは、**NOT ROLLFORWARD RECOVERABLE** パラメーターも指定された場合のみ使用できます。

指定された値がサポートされている最小値よりも小さい場合には、最小値が使用され、警告は戻されません。 **DATA BUFFER** 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。

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。システムにさらに使用可能なメモリーがある場合、**REDISTRIBUTE DATABASE PARTITION GROUP** コマンドの **DATA BUFFER** パラメーターの値は、一時的に **util_heap_sz** を超える場合があります。

INDEXING MODE
再配分時の索引の保守方法を指定します。このコマンド・パラメーターは、**NOT ROLLFORWARD RECOVERABLE** パラメーターも指定された場合のみ使用できます。

有効な値は以下のとおりです。

REBUILD
索引が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る結果として、「索引」ページがディスク上で一緒にクラスター化されます。

DEFERRED
再配分で索引の維持を試行しません。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: 非 MDC 表および非 ITC 表の場合、表に無効な索引があると、**INDEXING MODE DEFERRED** が指定されていなければ、**REDISTRIBUTE DATABASE PARTITION GROUP** コマンドによって自動的に索引が再作成されます。 MDC 表または ITC 表の場合、**INDEXING MODE DEFERRED** が指定されている場合でも、無効な複合索引は表の再配分が始まる前に再作成されます。なぜなら、ユーティリティーは MDC 表または ITC 表を処理するために複合索引を必要とするからです。

PRECHECK
データベース・パーティション・グループを再配分できるかどうかを検査します。このコマンド・パラメーターは、**NOT ROLLFORWARD RECOVERABLE** パラメーターも指定された場合のみ使用できます。

YES
これはデフォルト値です。再配分操作が開始されるのは、検査が正常に終了した場合のみです。検査に失敗すると、このコマンドは終了して、失敗した最初の検査に関連したエラー・メッセージを返します。

NO 再配分操作はすぐに開始されます。検査は行われません。

ONLY
検査実行後に、コマンドは終了します。再配分は行われません。デフォルトでは、データベースは静止されません。**QUIESCE DATABASE** コマンド・パラメーターが YES に設定された場合、またはデフォルト値の YES の場合には、データベースは静止状態のままになります。データベースへの接続を復元するには、再配分操作を実行するか、**UNQUIESCE DATABASE** コマンドを発行します。

QUIESCE DATABASE
データベースからすべてのユーザーを強制的に切断して静止モードにするように指定します。このコマンド・パラメーターは、**NOT ROLLFORWARD RECOVERABLE** パラメーターも指定された場合のみ使用できます。

YES
これはデフォルト値です。SYSADM、SYSMAINT、または SYSCTRL の権限を持つユーザーと、QUIESCE_CONNECT 権限を付与されたユーザーだけが、データベースとそのオブジェクトにアクセスできるようになります。再配分が正常に完了すると、データベースは静止解除されます。

NO 再配分操作によって、データベースが静止状態になることはありません。ユーザーがデータベースから強制的に切断されることもありません。

詳しくは、**QUIESCE DATABASE** コマンドを参照してください。

STATISTICS
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ように指定します。このコマンド・パラメーターは、**NOT ROLLFORWARD RECOVERABLE** パラメーターも指定された場合のみ使用できます。

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

データ分散ファイル distfile_for_dbpg_1 によって現在のデータ分散を提供することにより、データベース・パーティション・グループ DBPG_1 を再配分します。データを 2 つの新しいデータベース・パーティション 6 および 7 に移動します。

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1
   USING DISTFILE /home/user1/data/distfile_for_dbpg_1
   ADD DATABASE PARTITION (6 TO 7)
```

データベース・パーティション・グループ DBPG_2 に対して以下のような再配分を行います。

- 再配分はロールフォワード・リカバリー可能ではない。
- データは複数のハッシュ・パーティションにわたって一様に配分される。
- 索引は最初から再作成される。
- 統計は収集されない。
- データ転送のためのバッファー・スペースとして、180,000 個の 4 KB ページが使用される。

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_2
 NOT ROLLFORWARD RECOVERABLE
 UNIFORM
 INDEXING MODE REBUILD
 DATA BUFFER 180000
 STATISTICS NONE
```

この再配分操作では、**QUIESCE DATABASE** および **PRECHECK** コマンド・パラメーターのデフォルト値により、データベースの静止および事前検査も実行されます。

使用上の注意

- 再配分操作を開始する前に、表が通常状態にあり、「ロード・ペンディング」状態でも「REORG ペンディング」状態でも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。
- **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、その再配分操作に関する一般情報、および各表の処理開始時刻と処理終了時刻などの情報を含むイベント・ログ・ファイルが作成されます。このイベント・ログ・ファイルは、以下のように書き込まれます。
  - Linux および 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。
- このユーティリティーは、処理中に断続的な 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なります。データベース・パーティション・グループの再配分操作が完了した後で、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。明示的な再バインドにより、無効パッケージに対する最初の SQL 要求の実行での初期遅延がなくなります。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ります。
- 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 を用いて定義された表を含むデータベース・パーティション・グループは、再配分することができません。
- データベース・パーティション・グループに、既存の宣言済み一時表または作成済み一時表を含む USER TEMPORARY 表スペースがある場合、再配分を行うことはできません。
- **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。
- **REDISTRIBUTE DATABASE PARTITION GROUP** コマンドは、データベース・パーティションの追加サーバー要求が保留中または進行中の場合は、失敗する可能性があります (SQLSTATE 55071)。さらに、新規データベース・パーティション・サーバーがオンラインでインスタンスに追加された場合、およびすべてのアプリケーションが新規データベース・パーティション・サーバーを認識しているわけではない場合、このコマンドは失敗する可能性があります (SQLSTATE 55077)。

互換性

DB2 バージョン 9.5 以前の XML レコード・フォーマットを使用する XML 列が入った表は、再配分できません。そうした表を新しいフォーマットに移行するには、ADMIN_MOVE_TABLE ストアード・プロシージャーを使用します。

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。

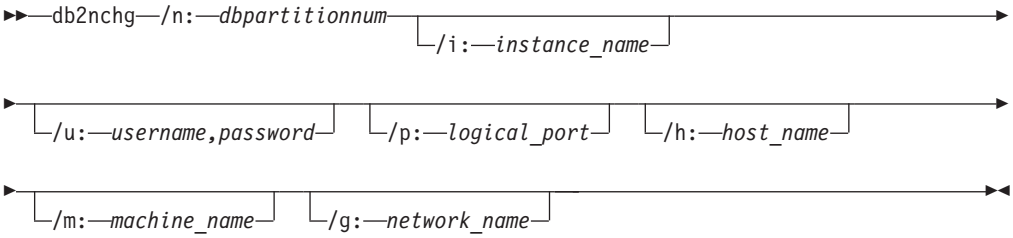
このコマンドが使用できるのは、データベース・パーティション・サーバーが停止している場合だけです。

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。インスタンスに既存のデータベースがない場合にのみ、データベース・パーティション・サーバーを移動させることができます。

/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。

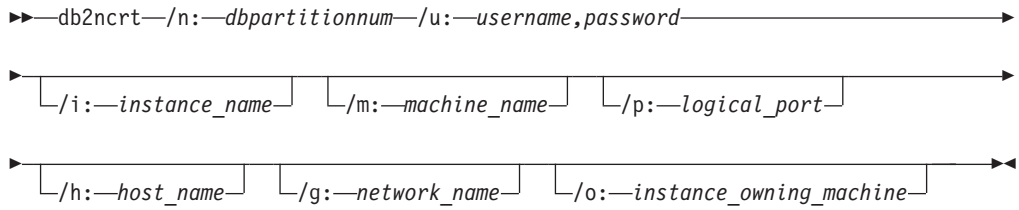
有効範囲

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。こうすると、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が確実に正しく追加されます。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。入力できる数値の範囲は 1 から 999 までです。

/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 通信用に予約されているポート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。インスタンス所有コンピューターではない任意のコンピューターで **db2ncrt** コマンドが呼び出される場合、このパラメーターは必須です。

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。

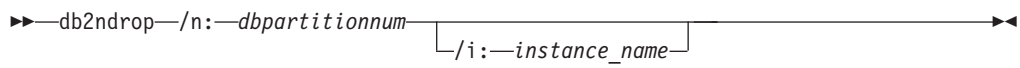
このコマンドが使用できるのは、データベース・パーティション・サーバーが停止している場合だけです。

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 dbpartitionnum** 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持つ 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。

466 ページの表 42 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。内部フォーマットは相互に互換性がなく、文字またはグラフィック・データ・タイプとの互換性がありません。
- 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場合、FOR BIT DATA で定義される文字の列は、FOR BIT DATA なしの文字の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。
- 分散キーの同一値のタイム・スタンプは、そのタイム・スタンプの精度が異なっている場合であっても、同一として取り扱われます。
- 分散キーの同一値の小数部は、位取りおよび精度が異なっている場合であっても、同一として取り扱われます。
- 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。
- 分散キーの一部としてサポートされていないデータ・タイプは、データベース・パーティションの互換性には適用できません。この対象には、データ・タイプが BLOB、CLOB、DBCLOB、XML、またはこれらのタイプのいずれかに基づく特殊タイプ、構造化タイプの列が含まれます。

表 42. 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	浮動小数点数	10 進浮動小数点数	文字ストリング	GRAPHIC ストリング	日付	時刻	TIMESTAMP	特殊タイプ
2 進整数	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	[1]
10 進数	いいえ	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	[1]
浮動小数点数	いいえ	いいえ	はい	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	[1]
10 進浮動小数点数	いいえ	いいえ	いいえ	はい	いいえ	いいえ	いいえ	いいえ	いいえ	[1]
文字ストリング	いいえ	いいえ	いいえ	いいえ	はい[2]	2, 3	いいえ	いいえ	いいえ	[1]
GRAPHIC ストリング	いいえ	いいえ	いいえ	いいえ	2, 3	はい[2]	いいえ	いいえ	いいえ	[1]
日付	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	いいえ	いいえ	[1]
時刻	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	いいえ	[1]
Timestamp	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	いいえ	はい	[1]
特殊タイプ	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]

注:

[1] 特殊タイプの値は、その特殊タイプのソース・データ・タイプと互換性があるか、または同じソース・データ・タイプの他の特殊タイプと互換性があるデータベース・パーティションです。特殊タイプのソース・データ・タイプは、分散キーの一部としてサポートされているデータ・タイプでなければなりません。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。特殊タイプは BLOB、CLOB、DBCLOB、または XML に基づくことはできません。

[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。

特殊レジスター

CURRENT MEMBER

CURRENT MEMBER 特殊レジスターは、ステートメントのコーディネーター メンバー 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メンバー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メンバー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 MEMBER 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または IBM DB2 pureScale Feature をサポートするように定義されていない場合、CURRENT MEMBER は 0 を戻します。 db2nodes.cfg ファイルが存在しない場合、データベース・インスタンスはこれらの環境をサポートするようには定義されません。パーティション・データベース環境または DB2 pureScale 環境の場合、db2nodes.cfg ファイルが存在し、そこにはデータベース・パーティション定義およびメンバー定義があります。

CURRENT MEMBER 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。

以前のバージョンの DB2 およびその他のデータベース製品との互換性のため、MEMBER の代わりに NODE を指定できます。

例

例 1: 以下の例では、アプリケーションが接続しているメンバーの番号をホスト変数 APPL_NODE (整数) に設定しています。

```
VALUES CURRENT MEMBER
  INTO :APPL_NODE
```

例 2: 以下のコマンドは、パーティション・データベース環境の 4 メンバー・システムのメンバー 0 で実行しています。この照会は、現在接続しているデータベースのメンバー番号を取得します。

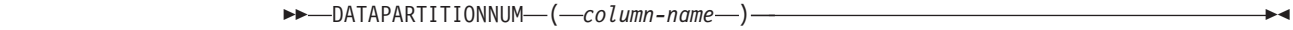
```
db2 "values current member"

1
-----------
          0
```

第 32 章 SQL 関数

DATAPARTITIONNUM

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。



スキーマは SYSIBM です。

column-name
表内の任意の列の修飾された名前または無修飾の名前。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。

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。例えば、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。

結果のデータ・タイプは INTEGER であり、NULL 値にはなりません。

注

- 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

- *例 1:* EMPLOYEE.EMPNO の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を取得します。

```
SELECT DATAPARTITIONNUM (EMPNO)
FROM EMPLOYEE
```

- *例 2:* DATAPARTITIONNUM によって戻されたシーケンス番号 (例えば 0) を、他の SQL ステートメント (例えば 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

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。例えば、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。

スキーマは SYSIBM です。

column-name
表内の任意の列の修飾された名前または無修飾の名前。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。例えば、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。

注

- DBPARTITIONNUM 関数は、複製された表、チェック制約内、または生成列の定義で使用することはできません (SQLSTATE 42881)。
- DBPARTITIONNUM 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。
- ***代替構文:*** 以前のバージョンの DB2 製品との互換性を確保するため、関数名 NODENUMBER は、DBPARTITIONNUM のシノニムとなっています。

例

- *例 1*: 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)
```

- *例 2*: 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)
```

- *例 3*: 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、1 つ以上のデータベース・パーティションを、データベース・パーティション・グループに追加したりデータベース・パーティション・グループからドロップしたりする場合に使用します。

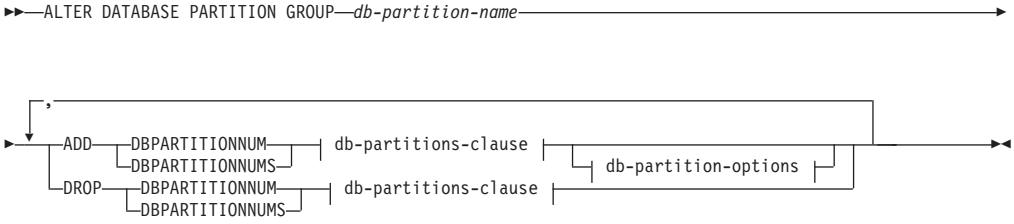
呼び出し

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。これは、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)。

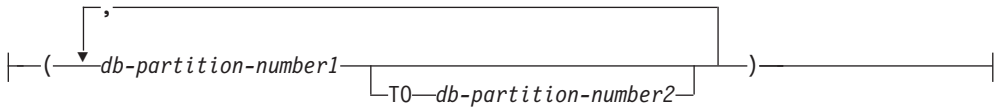
許可

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。

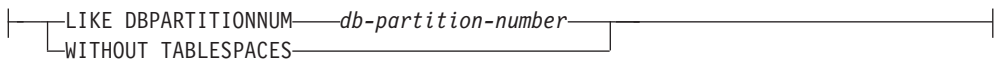
構文



db-partitions-clause:



db-partition-options:



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・グループに既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・グループに既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)。

db-partition-options

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* または MANAGED BY AUTOMATIC STORAGE 節を用いた 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)。
- データベース・パーティションの追加サーバー要求が保留中または進行中の場合、ALTER DATABASE PARTITION GROUP ステートメントに障害が起こる可能性があります (SQLSTATE 55071)。また、このステートメントは、新規データベース・パーティション・サーバーがオンラインでインスタンスに追加され、す

べてのアプリケーションがこの新規データベース・パーティション・サーバーについて認識しているわけではない場合にも失敗する可能性があります (SQLSTATE 55077)。

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。
- ***代替構文***: DB2 の以前のバージョンおよび他のデータベース製品との互換性のために、以下の代替の構文がサポートされています。これらの代替は非標準であり、使用すべきではありません。
  - 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。

- *例 1*: 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
```

- *例 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

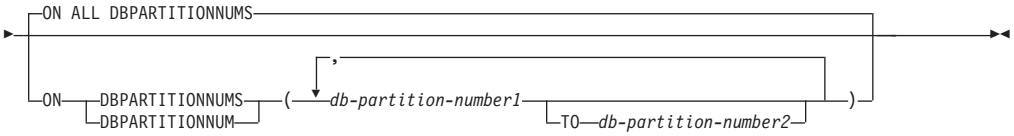
このステートメントは、アプリケーション・プログラムに組み込むか、あるいは対話式に発行することができます。これは、DYNAMICRULES の実行動作がパッケージに効力を持つ場合にのみ、動的に準備できる実行可能ステートメントです (SQLSTATE 42509)。

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

```
►►—CREATE DATABASE PARTITION GROUP—db-partition-group-name———————————————————►
```



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、既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)。

- データベース・パーティションの追加サーバー要求が保留中または進行中の場合、CREATE DATABASE PARTITION GROUP ステートメントに障害が起こる可能性があります (SQLSTATE 55071)。また、このステートメントは、新規データベース・パーティション・サーバーがオンラインでインスタンスに追加され、すべてのアプリケーションがこの新規データベース・パーティション・サーバーについて認識しているわけではない場合にも失敗する可能性があります (SQLSTATE 55077)。

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 32768 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。
- ***代替構文***: DB2 の以前のバージョンおよび他のデータベース製品との互換性のために、以下の代替の構文がサポートされています。これらの代替は非標準であり、使用すべきではありません。
  - 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

以下の例は、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・データベースに基づいています。

- *例 1*: 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
```

- *例 2*: 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)
```

- *例 3*: 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 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) の調整メンバー番号、および現在の STMM 調整メンバー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

ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかの権限または特権が含まれていなければなりません。

- DBADM
- SECADM
- SQLADM
- ACCESSCTRL
- DATAACCESS
- SYSIBM.SYSTUNINGINFO に対する SELECT

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 member' )
```

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) の調整メンバー番号 (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メンバーを指定します。データベースの稼働中に、調整メンバーは 1 時間に数度適用されます。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 メンバー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。

互換性

以前のバージョンとの互換性:

- **DB2_ENFORCE_MEMBER_SYNTAX** レジストリー変数が ON に設定されている場合を除き、**DBPARTITIONNUM** を **MEMBER** の代わりに使用できます。

ADMIN_CMD プロシージャーを使用する UPDATE STMM TUNING 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・メンバー番号を更新します。

許可

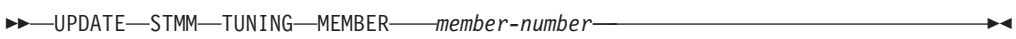
ステートメントの許可 ID によって保持されている特権には、少なくとも以下のいずれかの権限が含まれていなければなりません。

- DBADM
- DATAACCESS
- SQLADM

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

member-number

member-number は整数です。パーティション・データベース環境では、-1 または存在しないメンバー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メンバーを自動的に選択します。DB2 pureScale環境では、-1 または存在しないメンバー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メンバーをランダムに選択します。

例

パーティション・データベース環境で、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、メンバー 3 にします。

```
CALL SYSPROC.ADMIN_CMD( 'update stmm tuning member 3' )
```

使用上の注意

- STMM 調整プロセスは、ユーザー設定の STMM 調整メンバー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、*member-number* が存在しており、それがアクティブなメンバーであれば、ユーザー設定の STMM 調整メンバーに移ります。このコマンドが STMM 調整メンバー番号を変更すると、現在の STMM 調整メンバー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。

互換性

以前のバージョンとの互換性:

- **DB2_ENFORCE_MEMBER_SYNTAX** レジストリー変数が ON に設定されている場合を除き、**DBPARTITIONNUM** を **MEMBER** の代わりに使用できます。

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。

注: この表関数は使用すべきではなく、DB2_MEMBER 管理ビュー、DB2_CF 管理ビュー、および DB2_GET_INSTANCE_INFO 表関数に置き換えられました。

構文

スキーマは SYSPROC です。

許可

このルーチンを実行するには、以下のいずれかの権限が必要です。

- ルーチンに対する EXECUTE 特権
- DATAACCESS 権限
- DBADM 権限
- SQLADM 権限

デフォルトの PUBLIC 特権

制限のないデータベースでは、この関数が自動的に作成されると、EXECUTE 特権が PUBLIC に付与されます。

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

4 つのメンバーを持つパーティション・データベース・インスタン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 SWITCHNAME
---------------- --------- ----------- -------
               0 so1                 0 so1-ib0
               1 so2                 0 so2-ib0
               2 so3                 0 so3-ib0
               3 so4                 0 so4-ib0

    4 record(s) selected.
```

DB2 pureScale環境では、3 つのメンバーおよび 1 つのクラスター・キャッシング・ファシリティー DB2 pureScaleインスタン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 SWITCHNAME
---------------- ---------- ----------- ----------
               0 so1                  0 so1-ib0
               0 so2                  0 so2-ib0
               0 so3                  0 so3-ib0

    3 record(s) selected.
```

使用上の注意

DB2 Enterprise Server Edition およびパーティション・データベース環境では、DB_PARTITIONS 表関数は、db2nodes.cfg ファイル内の項目ごとに 1 行ずつ戻します。

DB2 pureScale環境では、DB_PARTITIONS 表関数は、以下の情報が列に入った複数の行を返します。

- PARTITION_NUMBER 列は常に 0 です。
- 残りの列は、db2nodes.cfg ファイル内の、現在のメンバーの項目に関する情報を表示します。

戻される情報

表 43. DB_PARTITIONS 表関数によって戻される情報

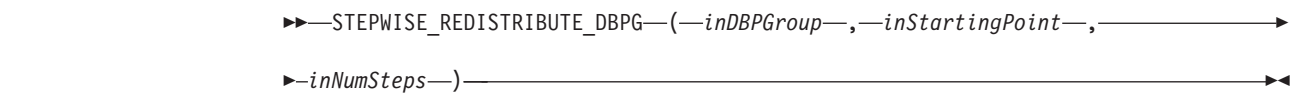
列名	データ・タイプ	説明
PARTITION_NUMBER	SMALLINT	partition_number - パーティション番号モニター・エレメント
HOST_NAME	VARCHAR(256)	host_name - ホスト名モニター・エレメント
PORT_NUMBER	SMALLINT	データベース・パーティション・サーバーのポート番号。
SWITCH_NAME	VARCHAR(128)	データベース・パーティション通信用の高速相互接続 (スイッチ) の名前。

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。例えば、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

デフォルトの PUBLIC 特権

制限のないデータベースでは、このプロシージャーが自動的に作成されると、EXECUTE 特権が PUBLIC に付与されます。

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、「*パーティションおよびクラスタリングのガイド*」の『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』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 データベース・サーバーのインストール

ほとんどの DB2 データベース製品は、非 root ユーザーとしてインストールできます。

始める前に

非 root 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、root インストールと非 root インストールの違い、および非 root 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 root 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、root ユーザーに適用される場合と全く同様に非 root 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。入出力完了ポート (IOCP) がシステムで使用可能になっていることが強く推奨されています。

- 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 root 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 root 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 root ユーザーとしてログインすること以外は、非 root 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 root インストールを実行するには:

1. 非 root 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)
   - 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 root 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 root 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セットアップする場合は、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 root ユーザーのホーム・ディレクトリーです。

次のタスク

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 データベース・システム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

データのバックアップ

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。このバックアップ・データは、元のデータに障害や損傷が発生した場合に使用できます。

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** コマンド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** コマンドを使ってデータベース・パーティションを個別にバックアップできます。または、**ON DBPARTITIONNUM** コマンド・パラメーターを使っていくつかのデータベース・パーティションを一度にバックアップしたり、**ALL DBPARTITIONNUMS** パラメーターを使ってすべてのデータベース・パーティションを同時にバックアップすることができます。 **LIST DBPARTITIONNUMS** コマンドを使用すると、バックアップ対象となるユーザー表を含んでいるデータベース・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。例えば、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。このアクションが必要な理由は、バックアップ操作の際にカタログ・

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な場合があり、その間は他のデータベース・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。

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アップグレードしていない場合は、データベースをアップグレード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。

制約事項

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す。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ノードでも)。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
  - 索引の作成
  - 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** コマンドを発行すると、データベースがアクティブ状態ではないことを確認できます。

手順

バックアップ・ユーティリティーを起動するには、次のようにします。

- コマンド行プロセッサー (CLP) で **BACKUP DATABASE** コマンドを発行します。
- BACKUP DATABASE パラメーターを使って ADMIN_CMD プロシージャーを実行します。
- 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) を使用します。
- **BACKUP DATABASE** コマンドに関する IBM Data Studio のタスク・アシストを開きます。

例

以下は、CLP を介して発行される **BACKUP DATABASE** コマンドの例です。

```
db2 backup database sample to c:¥DB2Backups
```

次のタスク

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・プールとメンバーの組み合わせのうち、メンバー上のバッファー・プールのサイズが、同じデータベース・パーティション・グループ内の他のメンバー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 44. 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		メンバー番号。
NPAGES	INTEGER		このメンバーにおけるこのバッファー・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 45. 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。複数のデータベース・パーティションに対して表を作成する場合、データ・パーティション統計は 1 つのデータベース・パーティションを表すことに注意してください。

表 46. 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。

表 46. SYSCAT.DATAPARTITIONS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
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。
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 値。
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 値。
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ており、デタッチされた従属はこのパーティションの内容に関して増分保守される。 • 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。 • L = データ・パーティションは論理的にデタッチされた • 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)。

表 46. SYSCAT.DATAPARTITIONS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
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
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)。
CARD	BIGINT		データ・パーティション内の行の総数。統計が収集されていない場合は -1。
OVERFLOW	BIGINT		データ・パーティション内のオーバーフロー・レコードの総数。統計が収集されていない場合は -1。
NPAGES	BIGINT		データ・パーティションの行が存在しているページの総数。統計が収集されていない場合は -1。
FPAGES	BIGINT		データ・パーティション内のページの総数。統計が収集されていない場合は -1。
ACTIVE_BLOCKS	BIGINT		データ・パーティション内のアクティブ・ブロックの総数、または -1。マルチディメンション・クラスタリング表 (MDC) のみに適用されます。
INDEX_TBSPACEID	INTEGER		このデータ・パーティションのパーティション索引すべてが保持されている表スペースの ID。
AVGROWSIZE	SMALLINT		このデータ・パーティションの中の圧縮された行と圧縮されていない行の両方の長さの平均 (バイト単位)。統計が収集されていない場合は -1。
PCTROWSCOMPRESSED	REAL		データ・パーティションの中の行の総数に対する圧縮された行のパーセンテージ。統計が収集されていない場合は -1。
PCTPAGESAVED	SMALLINT		行の圧縮の結果、データ・パーティションに保存されるページの概算パーセンテージ。この値はデータ・パーティション内の各ユーザー・データ行のオーバーヘッド・バイトを含んでいますが、ディクショナリー・オーバーヘッドによって消費されるスペースは含みません。統計が収集されない場合は -1 です。
AVGCOMPRESSEDROWSIZE	SMALLINT		このデータ・パーティションの中の圧縮された行の長さの平均 (バイト単位)。統計が収集されていない場合は -1。

表 46. SYSCAT.DATAPARTITIONS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
AVGROWCOMPRESSIONRATIO	REAL		データ・パーティションの中の圧縮された行の場合、これは行の平均の圧縮率です。つまり、圧縮されていない行の平均の長さを、圧縮された行の平均の長さで除算したものです。統計が収集されていない場合は、-1 です。
STATS_TIME	TIMESTAMP	Y	このオブジェクトについて記録されている統計値が最後に変更された時刻。統計が収集されていない場合は、NULL 値。
LASTUSED	DATE		データ・パーティションが DML ステートメントまたは LOAD コマンドによって最後に使用された日付。データ・パーティションが HADR スタンバイ・データベースで使用される場合、この列は更新されません。デフォルト値は '0001-01-01' です。この値は、多くて 24 時間に 1 回の頻度で非同期に更新されるため、ここ 15 分以内の使用は反映されていない可能性があります。

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 47. 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。

表 47. SYSCAT.DBPARTITIONGROUPDEF カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
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 48. 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・グループの所有者の許可 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・グループの所有者の許可 ID。

表 48. SYSCAT.DBPARTITIONGROUPS カタログ・ビュー (続き)

列名	データ・タイプ	NULL 可能	説明
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 49. 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 (65536)		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、32768 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、さまざまな方法でアクセスすることが可能な、各種形式で入手できます。

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)
  - サンプル・プログラム
  - チュートリアル
- DB2 資料
  - PDF ファイル (ダウンロード可能)
  - PDF ファイル (DB2 PDF DVD に含まれる)
  - 印刷資料
- コマンド行ヘルプ
  - コマンド・ヘルプ
  - メッセージ・ヘルプ

注: DB2 インフォメーション・センターのトピックは、PDF やハードコピー資料よりも頻繁に更新されます。最新の情報を入手するには、資料の更新が発行されたときにそれをインストールするか、ibm.com にある DB2 インフォメーション・センターを参照してください。

技術資料、ホワイト・ペーパー、IBM Redbooks® 資料などのその他の DB2 技術情報には、オンライン (ibm.com) でアクセスできます。 DB2 Information Management ソフトウェア・ライブラリー・サイト (http://www.ibm.com/software/data/sw-library/) にアクセスしてください。

資料についてのフィードバック

DB2 の資料についてのお客様からの貴重なご意見をお待ちしています。DB2 の資料を改善するための提案については、db2docs@ca.ibm.com まで E メールを送信してください。DB2 の資料チームは、お客様からのフィードバックすべてに目を通しますが、直接お客様に返答することはありません。お客様が関心をお持ちの内容について、可能な限り具体的な例を提供してください。特定のトピックまたはヘルプ・ファイルについてのフィードバックを提供する場合は、そのトピック・タイトルおよび URL を含めてください。

DB2 お客様サポートに連絡する場合には、この E メール・アドレスを使用しないでください。資料を参照しても、DB2 の技術的な問題が解決しない場合は、お近くの IBM サービス・センターにお問い合わせください。

DB2 テクニカル・ライブラリー (ハードコピーまたは PDF 形式)

以下の表は、IBM Publications Center (www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss) から利用できる DB2 ライブラリーについて説明しています。英語および翻訳された DB2 バージョン 10.1 のマニュアル (PDF 形式) は、www.ibm.com/support/docview.wss?rs=71&uid=swg27009474 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 50. 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	SA88-4671-00	入手可能	2012 年 4 月
管理ルーチンおよびビュー	SA88-4672-01	入手不可	2013 年 1 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	SA88-4676-01	入手可能	2013 年 1 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	SA88-4677-01	入手可能	2013 年 1 月
コマンド・リファレンス	SA88-4673-01	入手可能	2013 年 1 月
データベース: 管理の概念および構成リファレンス	SA88-4662-01	入手可能	2013 年 1 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	SA88-4693-01	入手可能	2013 年 1 月
データベースのモニタリング ガイドおよびリファレンス	SA88-4663-01	入手可能	2013 年 1 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	SA88-4694-01	入手可能	2013 年 1 月
データベース・セキュリティー・ガイド	SA88-4695-01	入手可能	2013 年 1 月

表 50. DB2 の技術情報 (続き)

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 ワークロード管理ガイドおよびリファレンス	SA88-4685-01	入手可能	2013 年 1 月
ADO.NET および OLE DB アプリケーションの開発	SA88-4665-01	入手可能	2013 年 1 月
組み込み SQL アプリケーションの開発	SA88-4666-01	入手可能	2013 年 1 月
Java アプリケーションの開発	SA88-4669-01	入手可能	2013 年 1 月
Perl、PHP、Python および Ruby on Rails アプリケーションの開発	SA88-4670-00	入手不可	2012 年 4 月
IBM データ・サーバー用の RDF アプリケーション開発	SA88-5083-00	入手可能	2013 年 1 月
SQL および外部ルーチンの開発	SA88-4667-01	入手可能	2013 年 1 月
データベース・アプリケーション開発の基礎	GI88-4279-01	入手可能	2013 年 1 月
DB2 インストールおよび管理 概説 (Linux および Windows 版)	GI88-4280-00	入手可能	2012 年 4 月
グローバリゼーション・ガイド	SA88-4696-00	入手可能	2012 年 4 月
DB2 サーバー機能 インストール	GA88-4679-01	入手可能	2013 年 1 月
IBM データ・サーバー・クライアント機能 インストール	GA88-4680-00	入手不可	2012 年 4 月
メッセージ・リファレンス 第 1 巻	SA88-4688-01	入手不可	2013 年 1 月
メッセージ・リファレンス 第 2 巻	SA88-4689-01	入手不可	2013 年 1 月
Net Search Extender 管理およびユーザーズ・ガイド	SA88-4691-01	入手不可	2013 年 1 月
パーティションおよびクラスタリングのガイド	SA88-4697-01	入手可能	2013 年 1 月
Preparation Guide for DB2 10.1 Fundamentals Exam 610	SC27-4540-00	入手不可	2013 年 1 月

表 50. 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
Preparation Guide for DB2 10.1 DBA for Linux, UNIX, and Windows Exam 611	SC27-4541-00	入手不可	2013 年 1 月
pureXML ガイド	SA88-4686-01	入手可能	2013 年 1 月
Spatial Extender ユーザーズ・ガイドおよびリファレンス	SA88-4690-00	入手不可	2012 年 4 月
SQL プロシージャー言語: アプリケーションのイネーブルメントおよびサポート	SA88-4668-01	入手可能	2013 年 1 月
SQL リファレンス 第 1 巻	SA88-4674-01	入手可能	2013 年 1 月
SQL リファレンス 第 2 巻	SA88-4675-01	入手可能	2013 年 1 月
Text Search ガイド	SA88-4692-01	入手可能	2013 年 1 月
問題判別およびデータベース・パフォーマンスのチューニング	SA88-4664-01	入手可能	2013 年 1 月
DB2 バージョン 10.1 へのアップグレード	SA88-4678-01	入手可能	2013 年 1 月
DB2 バージョン 10.1 の新機能	SA88-4684-01	入手可能	2013 年 1 月
XQuery リファレンス	SA88-4687-01	入手不可	2013 年 1 月

表 51. DB2 Connect 固有の技術情報

資料名	資料番号	印刷資料が入手可能かどうか	最終更新
DB2 Connect Personal Edition インストールおよび構成	SA88-4681-00	入手可能	2012 年 4 月
DB2 Connect サーバー機能 インストールおよび構成	SA88-4682-01	入手可能	2013 年 1 月
DB2 Connect ユーザーズ・ガイド	SA88-4683-01	入手可能	2013 年 1 月

コマンド行プロセッサーから SQL 状態ヘルプを表示する

DB2 製品は、SQL ステートメントの結果として生じる可能性がある状態に対応した SQLSTATE 値を戻します。 SQLSTATE ヘルプは、SQL 状態および SQL 状態クラス・コードの意味を説明します。

手順

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

```
? sqlstate または ? class code
```

ここで、*sqlstate* は有効な 5 桁の SQL 状態を、*class code* は SQL 状態の最初の 2 桁を表します。
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

他のバージョンの DB2 製品の資料は、ibm.com® のそれぞれのインフォメーション・センターにあります。

このタスクについて

DB2 バージョン 10.1 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v10r1 です。

DB2 バージョン 9.8 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/v9r8/ です。

DB2 バージョン 9.7 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/ です。

DB2 バージョン 9.5 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9r5 です。

DB2 バージョン 9.1 のトピックを扱っている *DB2 インフォメーション・センター* の URL は、http://publib.boulder.ibm.com/infocenter/db2luw/v9/ です。

DB2 バージョン 8 のトピックについては、*DB2 インフォメーション・センター* の URL (http://publib.boulder.ibm.com/infocenter/db2luw/v8/) を参照してください。

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

ローカルにインストールした DB2 インフォメーション・センターは、定期的に更新する必要があります。

始める前に

DB2 バージョン 10.1 インフォメーション・センターが既にインストール済みである必要があります。詳しくは、「*DB2 サーバー機能 インストール*」の『DB2 セットアップ・ウィザードによる DB2 インフォメーション・センターのインストール』のトピックを参照してください。インフォメーション・センターのインストールに適用されるすべての前提条件と制約事項は、インフォメーション・センターの更新にも適用されます。

このタスクについて

既存の DB2 インフォメーション・センターは、自動で更新することも手動で更新することもできます。

- 自動更新は、既存のインフォメーション・センターのフィーチャーと言語を更新します。自動更新を使用すると、手動更新と比べて、更新中にインフォメーション・センターが使用できなくなる時間が短くなるというメリットがあります。さらに、自動更新は、定期的に実行する他のバッチ・ジョブの一部として実行されるように設定することができます。
- 手動更新は、既存のインフォメーション・センターのフィーチャーと言語の更新に使用できます。自動更新は更新処理中のダウン時間を減らすことができますが、フィーチャーまたは言語を追加する場合は手動処理を使用する必要があります。例えば、ローカルのインフォメーション・センターが最初は英語とフランス語でインストールされており、その後ドイツ語もインストールすることにした場合、手動更新でドイツ語をインストールし、同時に、既存のインフォメーション・センターのフィーチャーおよび言語を更新できます。しかし、手動更新ではインフォメーション・センターを手動で停止、更新、再始動する必要があります。更新処理の間はずっと、インフォメーション・センターは使用できなくなります。自動更新処理では、インフォメーション・センターは、更新を行った後に、インフォメーション・センターを再始動するための停止が発生するだけで済みます。

このトピックでは、自動更新のプロセスを詳しく説明しています。手動更新の手順については、『コンピューターまたはイントラネット・サーバーにインストールされた DB2 インフォメーション・センターの手動更新』のトピックを参照してください。

手順

コンピューターまたはイントラネット・サーバーにインストールされている DB2 インフォメーション・センターを自動更新する手順を以下に示します。

1. Linux オペレーティング・システムの場合、次のようにします。
   a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`/opt/ibm/db2ic/V10.1` 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 `update-ic` スクリプトを実行します。

      ```
      update-ic
      ```
2. Windows オペレーティング・システムの場合、次のようにします。
   a. コマンド・ウィンドウを開きます。
   b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、DB2 インフォメーション・センターは、`<Program Files>¥IBM¥DB2 Information Center¥バージョン 10.1` ディレクトリーにインストールされています (`<Program Files>` は「Program Files」ディレクトリーのロケーション)。

c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。

d. 次のように update-ic.bat ファイルを実行します。

```
update-ic.bat
```

タスクの結果

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

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

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

このタスクについて

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

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

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

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

手順

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

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

     ```
     /etc/init.d/db2icdv10 stop
     ```
2. インフォメーション・センターをスタンドアロン・モードで開始します。
   - Windows の場合:
     a. コマンド・ウィンドウを開きます。
     b. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2* インフォメーション・センター は、*Program_Files*¥IBM¥DB2 Information Center¥バージョン 10.1 ディレクトリーにインストールされています (*Program_Files* は Program Files ディレクトリーのロケーション)。
     c. インストール・ディレクトリーから doc¥bin ディレクトリーにナビゲートします。
     d. 次のように help_start.bat ファイルを実行します。

        ```
        help_start.bat
        ```
   - Linux の場合:
     a. インフォメーション・センターがインストールされているパスにナビゲートします。デフォルトでは、*DB2* インフォメーション・センター は、/opt/ibm/db2ic/V10.1 ディレクトリーにインストールされています。
     b. インストール・ディレクトリーから doc/bin ディレクトリーにナビゲートします。
     c. 次のように help_start スクリプトを実行します。

        ```
        help_start
        ```

   システムのデフォルト Web ブラウザーが開き、スタンドアロンのインフォメーション・センターが表示されます。
3. 「**更新**」ボタン () をクリックします。(ブラウザーで JavaScript が有効になっている必要があります。) インフォメーション・センターの右側のパネルで、「**更新の検索**」をクリックします。 既存の文書に対する更新のリストが表示されます。
4. インストール・プロセスを開始するには、インストールする更新をチェックして選択し、「**更新のインストール**」をクリックします。
5. インストール・プロセスが完了したら、「**完了**」をクリックします。

6. 次のようにして、スタンドアロンのインフォメーション・センターを停止します。
   - Windows の場合は、インストール・ディレクトリーの doc¥bin ディレクトリーにナビゲートしてから、次のように help_end.bat ファイルを実行します。

     ```
     help_end.bat
     ```

     注: help_end バッチ・ファイルには、help_start バッチ・ファイルを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。help_start.bat は、Ctrl-C や他の方法を使用して停止しないでください。
   - Linux の場合は、インストール・ディレクトリーの doc/bin ディレクトリーにナビゲートしてから、次のように help_end スクリプトを実行します。

     ```
     help_end
     ```

     注: help_end スクリプトには、help_start スクリプトを使用して開始したプロセスを安全に停止するのに必要なコマンドが含まれています。他の方法を使用して、help_start スクリプトを停止しないでください。
7. *DB2 インフォメーション・センター* を再開します。
   - Windows では、「**スタート**」 > 「**コントロール パネル**」 > 「**管理ツール**」 > 「**サービス**」をクリックします。次に、「**DB2 インフォメーション・センター**」サービスを右クリックして「**開始**」を選択します。
   - Linux では、以下のコマンドを入力します。

     ```
     /etc/init.d/db2icdv10 start
     ```

タスクの結果

更新された *DB2 インフォメーション・センター* に、更新された新しいトピックが表示されます。

DB2 チュートリアル

DB2 チュートリアルは、DB2 データベース製品のさまざまな機能について学習するための支援となります。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/db2luw/v10r1/) 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。

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 の資料

トラブルシューティング情報は、「*問題判別およびデータベース・パフォーマンスのチューニング*」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、以下の情報が記載されています。

- DB2 診断ツールおよびユーティリティーを使用した、問題の切り分け方法および識別方法に関する情報。
- 最も一般的な問題のうち、いくつかの解決方法。
- DB2 データベース製品で発生する可能性のある、その他の問題の解決に役立つアドバイス。

IBM サポート・ポータル

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見つけるには、IBM サポート・ポータル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。

IBM サポート・ポータル (http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows) 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。

適用度: これらのご利用条件は、IBM Web サイトのあらゆるご利用条件に追加で適用されるものです。

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。

権利: 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。

IBM の商標: IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

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。IBM 以外の製品に関する情報は、本書の最初の発行時点で入手可能な情報に基づいており、変更される場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

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

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

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

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

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

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADA

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

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

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

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

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

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

著作権使用許諾:

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

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

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

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

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

以下は、それぞれ各社の商標または登録商標です。

- Linux は、Linus Torvalds の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連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。
- 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。
- インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Celeron、Intel SpeedStep、Itanium、Pentium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。
- Microsoft、 Windows、 Windows NT、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

索引

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

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[数字]

A

B

C

D

F

G

H

I

L

M

N

R

S

T

U

V

W

X

Printed in Japan

SA88-4697-01

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

Spine information:

IBM DB2 10.1 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