# InfiniDB®
# 概要

Release：4.5
Document Version：4.5-2

InfiniDB 概要
2014 年 5 月

# 目次

# 1 はじめに

『InfiniDB概要』へようこそ。本書では、分析用データベースInfiniDBの概要と、InfiniDBのアーキテクチャおよびこのデータベースの機能について説明します。このマニュアルを読み終えると、InfiniDBの理解を深め、管理および開発の両方の観点から、このデータベースを有効に利用する準備ができるようになります。

## 1.1 対象読者

本書は、以下に示す様々な役割の読者を対象としています。

- データベース管理者
- アプリケーションおよびWebの開発者
- データの設計者
- システムおよびWebの管理者

## 1.2 マニュアルリスト

InfiniDB データベースプラットフォームのマニュアルは、様々な読者を対象とした複数のガイドで構成されています。次の表を参照してください。

| マニュアル | 説明 |
|---|---|
| 『InfiniDB 管理者ガイド』 | InfiniDB を管理するための詳細な手順について説明します。 |
| 『InfiniDB 最小推奨仕様ガイド』 | InfiniDB の実装に必要なハードウェアおよびソフトウェアの最小の推奨仕様を示します。 |
| 『InfiniDB インストレーションガイド』 | InfiniDB をインストールするために必要な手順の概要について説明します。 |
| 『InfiniDB マルチ UM 構成ガイド』 | マルチユーザーモジュールの構成情報について説明します。 |
| 『InfiniDB SQL 構文ガイド』 | InfiniDB に固有の構文について説明します。 |
| 『InfiniDB パフォーマンスチューニングガイド』 | 分析用データベース InfiniDB をパラレル化および拡張するためのチューニングに役立つ情報について説明します。 |

## 1.3 マニュアルの入手

英語版のマニュアルは、http://www.infinidb.coで入手することができます。追加の支援が必要な場合はinfinidb_doc@ashisuto.co.jpにご連絡ください。

## 1.4 マニュアルへのフィードバック

マニュアルの改善に向けて、フィードバック、コメントおよび提案をいただけますようお願いいたします。マニュアル名、バージョンおよびページ番号を添えてコメントをinfinidb_doc@ashisuto.co.jpにご送付ください。

## 1.5 追加リソース

InfiniDBのインストールおよびチューニング、またはInfiniDBを使用したデータの問合せに関して支援が必要な場合はinfinidb_doc@ashisuto.co.jpまでご連絡ください。

# 2 分析用データベース InfiniDB の概要

分析用データベースInfiniDBは、大規模な収集情報の管理および取得に特化して設計されたソフトウェアソリューションで、通常、データウェアハウス、データマート、分析用データベースまたは分析用データストアと呼ばれているものです。従来のリレーショナルデータベースは分析操作またはビジネスインテリジェンス操作用ではなくトランザクション処理操作用に設計されていますが、InfiniDBは、この従来のリレーショナルデータベースに見られる制約をなくすように設計されています。InfiniDBのマルチスレッドの列指向アーキテクチャによって、安価な汎用ハードウェアを利用し、マシンにCPUおよびRAMを追加した拡張構成でテラバイトレベルのデータを効率的に管理することができます。また、従来のリレーショナルデータベース管理システム(RDBMS)では完了するのに時間がかかっていた問合せの応答時間を大幅に短縮することもできます。さらに、大規模パラレル処理(MPP)方法で展開するためのスケールアウト設計を実現できます。この構成では、すべての参加マシン間で処理がパラレル化されるため、実際よりも多いノードがシステムに追加されたかのようにパフォーマンスをリニアに向上させることができます。

さらに最近は、InfiniDBで固有のHDFS統合が追加され、InfiniDBをSQL-on-Hadoop問合せエンジンとして使用できるようになっています。これは、バッチ処理に集中するHadoopプラットフォームの上に増分アドオンとして構築された他のSQL-on-Hadoopソリューションに比べ、InfiniDBは、大幅にパフォーマンスが優れた高性能な分析を行うことを目的に構築されているためです。

ビジネスインテリジェンスおよびデータ分析は、日々の重要な意思決定および競争力の維持を目的として使用するビジネスの主要なメカニズムとなっているため、InfiniDBは多くの方にこの強力な分析用データベース機能をご利用いただけるものと確信しています。そのため、InfiniDBは分析用データベースInfiniDBの無償オープンソース版(InfiniDB)を提供します。だれでもGPLv2ライセンスのもと無償でダウンロードして使用できます。InfiniDBは高度な機能を提供し、技術サポートを週7日、24時間体制で行う分析用データベースInfiniDBの製品版(エンタープライズ版)も提供しています(日本でのサポートは9:00-17:00です(2014年6月現在))。

## 2.1 分析用データベースの概要

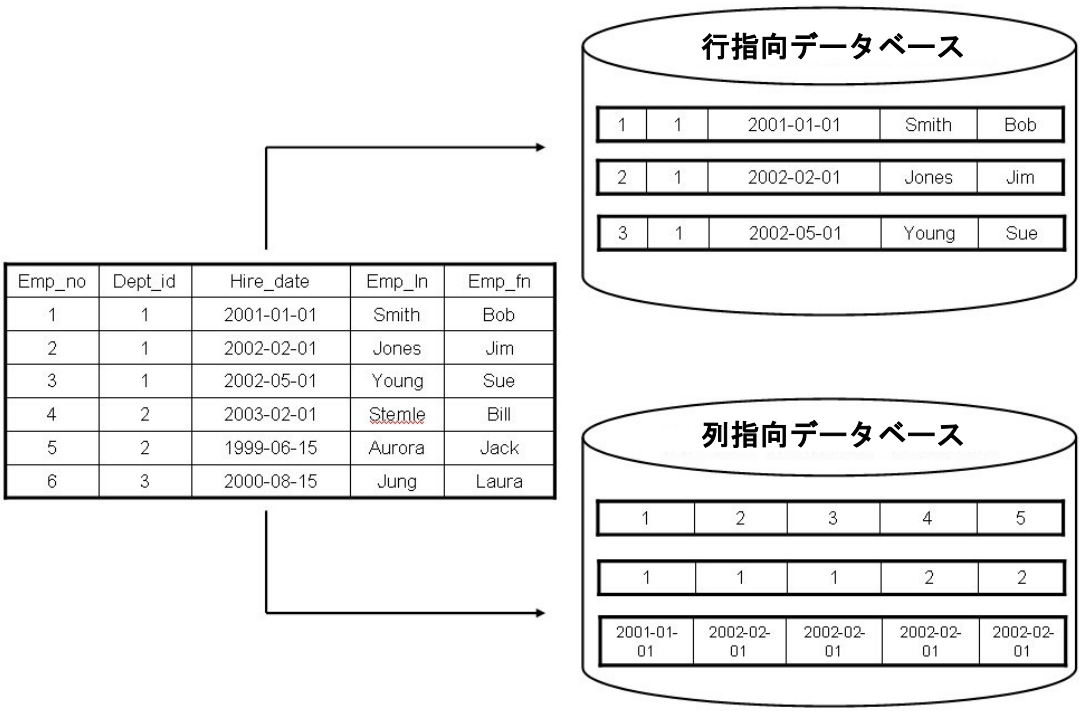
現在提供されているほぼすべての従来のリレーショナルデータベースは、オンライントランザクション処理（OLTP）のワークロードに対処することを主な目的として設計されています。通常、トランザクション（オンラインで製品を注文する行為など）は、リレーショナルデータベースの1つ以上の行に対応し、大部分のRDBMSの設計は行単位のパラダイムに基づいています。トランザクションベースのシステムの場合、このアーキテクチャは受信データの入力および情報の行ごとの削除を処理するのに適しています。

ただし、読取りが非常に集中していて、リクエストされる情報に対して選択問合せを実行するアプリケーションの場合、OLTPデータベースの設計が最適なモデルとはいえません。トランザクションは行ベースですが、ほとんどの場合、データベースの問合せは列ベースです。トランザクションデータの挿入と削除は行ベースのシステムで適切に処理できますが、表の少数の列のみを対象とする選択問合せは列指向アーキテクチャの方がはるかに適切に処理できます。平均的に、行ベースのシステムは、列ベースのデータベースで同じ情報を取得する場合に比べて5倍から10倍の物理I/Oを消費します。問合せで最も時間がかかる処理は通常は物理I/Oであること、また分析問合せでは一般的なトランザクションでのデータベース操作よりはるかに多い行数のデータを処理することを考えると、行指向アーキテクチャと列指向アーキテクチャのパフォーマンスの差は、データベースが拡大するにつれて広がることがよくあります。

選択問合せの効率を低下させないため、行ベースのRDBMSは索引付け、水平パーティション化、マテリアライズドビュー、サマリー表、パラレル処理を活用しています。これらの技法はすべて、複雑な問合せの実行には有益ですが、それぞれに欠点もあります。たとえば、索引付けは確かに問合せをより迅速に完了させるために役立つ場合がありますが、それにはより多くの格納領域を必要とし、挿入、更新、削除、バルクロードの操作の妨げになり（元になる表だけでなく索引も保持する必要があるため）、断片化が進むとパフォーマンスが低下する可能性があります。

さらに、ビジネスインテリジェンスと経営分析の場合は、非定型の問合せが実行されるため、どの列に索引を付ける必要があるか予測することはほぼ不可能です。そのため表に索引を付けすぎるか（そして負荷と保持の問題を引き起こすか）、または適切に索引を付けられず、多くの問合せでは期待する時間よりも終了がかなり遅くなります。

特に分析用に設計された列指向データベースは、行ではなく列に基づいてデータを保存、管理、照会することで、従来型のRDBMSの制約をなくします。行全体ではなく、問合せで指定された必要な列だけにアクセスするため、I/Oアクティビティも全体的な問合せ応答時間も削減できます。その結果、中程度の情報量（数十または数百GB）でも大量のデータ（数TB）に対しても、標準的なRDBMSに必要な時間より少ない時間で照会し問合せ結果を戻せます。

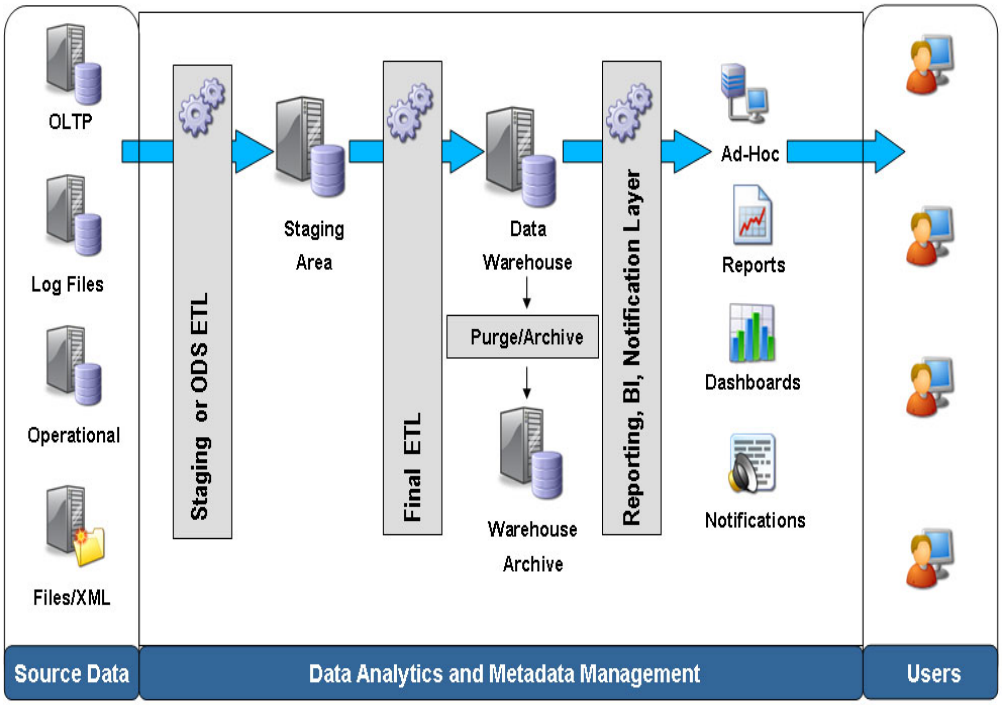



## 2.2 基本的な分析用データベースの使用事例

分析用データベースを使用するメリットがあるアプリケーションおよびシステムには、一般に次の3つの例があります。

1. データウェアハウス、データマート、その他のビジネスインテリジェンス(BI)データストア
2. 汎用のレポート作成データベース
3. アプリケーションの読取り/検索/参照を行う部分が分析/問合せデータベースによって処理され、トランザクションセグメント(注文入力など)が従来のリレーショナルデータベースで管理される、アプリケーション全体のうちの読取り集中セグメント

ビジネスインテリジェンス(BI)およびデータ分析は、ビジネスの重要な意思決定を行うための情報の円滑な利用に関係します。通常、BIおよびデータ分析のライフサイクルは次に示すとおりです。




この図では、業務データおよびトランザクションデータが抽出-変換-ロード(ETL)フェーズに入り、最終的に分析用データベースまたはデータウェアハウスに移入される仕組みを示しています。その後、分析用データベースは様々なビジネスインテリジェンス処理で使用され、非定型の問合せ、事前にビルドされたダッシュボードとレポート、または主要なデータ変更が発生したときに自動送信される通知を介してデータにアクセスする意思決定者をサポートします。

現在提供されているほぼすべてのリレーショナルデータベースは、オンライントランザクション処理(OLTP)のワークロードに対処することを主な目的として設計されています。通常、トランザクション(Amazonなど、Webベースで本を販売する業者から本をオンラインで注文する行為など)は、リレーショナルデータベースの1つ以上の行に対応し、大部分のRDBMSの設計は行単位のパラダイムに基づいています。トランザクションベースのシステムの場合、このアーキテクチャは受信データの入力を処理するのに適しています。

ただし、読取りが非常に集中していて、リクエストされる情報に対して選択問合せを実行するアプリケーションの場合、OLTPデータベースの設計は一般的に適していません。トランザクションは行ベースですが、ほとんどの場合、データベースの問合せは列ベースです。トランザクションデータの挿入と削除は行ベースのシステムで適切に処理できますが、表の少数の列のみを対象とする選択問合せは列指向アーキテクチャの方がはるかに適切に処理できます。平均的に、行ベースのシステムは、列ベースのデータベースで同じ情報を取得する場合に比べて5倍から10倍の物理I/Oを実行します。問合せで最も時間がかかる処理は通常は物理I/Oであること、また分析問合せでは一般的なトランザクションでのデータベース操作よりはるかに多い行数のデータを処理することを考えると、行指向アーキテクチャと列指向アーキテクチャのパフォーマンスの差は、データベースが拡大するにつれて広がることがよくあります。

選択問合せの効率を低下させないため、行ベースのRDBMSは索引付け、水平パーティション化、マテリアライズドビュー、サマリー表、パラレル処理を活用しています。これらの技法はすべて、集中的な問

合せには有益ですが、それぞれに欠点もあります。たとえば、索引付けは確かに問合せをより迅速に完了させるために役立つ場合がありますが、それにはより多くの格納領域を必要とし、挿入、更新、削除、バルクロードの操作の妨げになり(元になる表だけでなく索引も保持する必要があるため)、断片化が進むとパフォーマンスが低下する可能性があります。さらに、ビジネスインテリジェンスと経営分析の場合は、非定型の問合せが実行されるため、どの列に索引を付ける必要があるか予測することはほぼ不可能です。そのため表に索引を付けすぎるか(そして負荷と保持の問題を引き起こすか)、または適切に索引を付けられず、多くの問合せでは期待する時間よりも終了がかなり遅くなります。

特に分析用に設計された列指向データベースは、行ではなく列に基づいてデータを保存、管理、照会することで、従来型のRDBMSの制約をなくします。行全体ではなく、問合せで指定された必要な列だけにアクセスするため、I/Oアクティビティも全体的な問合せ応答時間も大幅に削減されます。その結果、中程度の情報量(数十または数百GB)でも大量のデータ(数TB)に対しても、標準的なRDBMSに必要な時間の何分の1かの時間で照会し問合せ結果を戻せます。

従来の行ベースのRDBMSではなく分析用に設計されたデータベースを選択することで得られる明らかなメリットがいくつかあります。列指向データベースについてのレポートで、Philip Howard氏(Bloor Researchのリサーチディレクタ)は次のように述べています。

> 「列のほうが低コストであり、かつ少ない占有領域で、優れたパフォーマンスを実現する。問合せのパフォーマンスに深く関心を持っている企業が、列ベースのソリューションをなぜ検討しないのか理解し難い。」[1]

## 2.3 InfiniDB の主な機能

分析用データベースInfiniDBは、読取りのみのWebアプリケーション、データマート、データウェアハウス、その他のビジネスインテリジェンス、分析、または問合せに重点を置いた使用事例など、集中的な読取りが発生する環境でパフォーマンスを向上させることを目的とした、様々な機能を提供します。

InfiniDBは、マルチCPUやマルチコアを搭載する今日のマシンのメリットを活用して、問合せと書込みを並行してマルチスレッド方式で処理できるように設計されています。InfiniDBサーバーの主な機能は次のとおりです。

- **列指向アーキテクチャ**:InfiniDBでは、従来の行ベースの形式ではなく、列ベースのアーキテクチャを採用しています。そのため、InfiniDBは、200-300GBから数TBもの情報を、行指向データベースより高速に任意の場所から読み取る必要がある問合せに適しています。
- **マルチスレッド設計**:InfiniDBはマルチスレッド設計であるため、今日の新しいマルチCPUやマルチコアベースのハードウェアを活用できます。CPUやコアの数が増えるほどInfiniDBのパフォーマンスは向上し、アプリケーションを修正する必要はありません。
- **垂直および水平の自動パーティション化**:列指向のInfiniDBは、垂直パーティション化を透過的に使用してデータを格納し、問合せに必要な列だけを読み取ります。ただし、ストレージの特別な配置や設計を必要とせずに、論理的な水平レンジパーティション化も使用できます。垂直パーティション化と論理的な水平レンジパーティション化の両方を使用することで、InfiniDBはどちらの方向(列と行)でもI/Oを削減できます。垂直および水平のパーティション化は、いずれもInfiniDBデータベースによって自動的に処理されるため、ユーザーの介入は不要です。

---

[1] Philip Howard、『What's Cool About Columns』、Bloor Research、2008年3月。

- **高レベルの同時実行機能**：同時実行に関しては、InfiniDBはサーバーマシンの処理能力によってのみ制限され、理論上の制限はありません。
- **高速データローダー**：InfiniDBは従来型の挿入、更新、削除処理をサポートしていますが、大量のデータをロードするには高速ロードユーティリティを利用できます。InfiniDBのローダーを使用すると、TB単位のデータを従来型の方法よりはるかに高速にInfiniDBデータベースにロードできます。
- **DMLのサポート**：データの高速バルクロードのサポートに加えて、InfiniDBはDML（挿入、更新、削除）操作もすべてサポートします。
- **トランザクションのサポート**：InfiniDBデータベースはACID準拠のトランザクションをサポートします。トランザクションは簡単にコミットまたはロールバックすることができ、競合の解決手段としてデッドロック検出もサポートします。
- **クラッシュからのリカバリ**：InfiniDBにはクラッシュからのリカバリ機能があります。システムクラッシュが発生すると、InfiniDBは自動的にデータの整合性を保持し、システム再起動時にロールフォワードとロールバックの操作によってデータベースを一貫した状態に戻すようサポートします。
- **MVCC設計**：InfiniDBは多版型同時実行制御（MVCC）つまり「スナップショット読取り」機能をサポートしているため、問合せ操作が表でブロックされることはありません。表は常に、その問合せを発行したときに存在していたデータを参照します。
- **索引付けが不要**：InfiniDBは垂直パーティション化と論理的な水平パーティション化の両方を透過的に使用するため、索引付けは不要です。本質的に、列指向データベースのデータが索引になります。また、InfiniDBはエクステントマップという小さい重要な構造を自動的に保持して、I/Oを削減するために利用します。エクステントマップによって手動によるデータのパーティション化も不要になります。
- **保守の軽減**：表の索引付けが不要になるほか、InfiniDBには、マテリアライズドビュー、サマリー表などの、パフォーマンスを向上させるためのオブジェクトも不要です。これは、一般的なデータベースで必要とされる管理と保守の作業を不要にし、システム全体の複雑さを軽減するためにも役立ちます。
- **ディクショナリの文字列の圧縮**：InfiniDBはトークンディクショナリ方式によって9バイト以上の固定長文字列と8バイト以上の可変長文字列を管理します。列ファイルには、長い文字列を含む個別のディクショナリファイルへのトークン（ポインタ）が格納されます。表内で重複した文字列値はディクショナリの1つの場所に、共有文字列エントリを指す重複したトークンとともに格納されます。
- **表定義変更のサポート**：ALTER TABLEコマンドを使用して表に対して列を追加および削除することもでき、オンライン状態で列を追加する機能も特別にサポートされています。
- **パフォーマンス診断**：パフォーマンスチューニングを支援するため、InfiniDBには監視と診断のユーティリティが付属しています。これは、ユーザーがデータベースを監視し、実行が遅いSQLのトラブルシューティングを行うために役立ちます。
- **MySQLフロントエンド**：InfiniDBはフロントエンドにMySQLを利用します。そのため、MySQLに精通したユーザーであれば、InfiniDBで、すぐに生産性を向上させることができます。MySQLをよく知らないユーザーも、MySQLは標準SQL操作のほぼすべてをサポートしているため、容易に習得できます。さらに、MySQLと他のベンダーによって無償のGUIツールが多数提供されており、InfiniDBデータベースの開発と管理に使用できます。
- **圧縮とリアルタイム解凍**：InfiniDBは、優れた圧縮機能を提供するとともに、ディスクからの読取りのパフォーマンスを最適化する圧縮方法を採用しています。これによって、I/Oバウンドのシステ

ムで、ディスクからの読取りのパフォーマンスを向上させることができます。圧縮はデフォルトで有効になっていますが、インスタンスレベル、表レベル、列レベルで有効と無効を切り替えることができます。列指向のストレージでは、類似したデータが各列ファイルに格納されるため、優れた圧縮性が実現されます。

- **パーティションの無効化とドロップ**：InfiniDBは、DBAがパーティションにあらかじめ名前を付けたり定義したりする必要がないように、自動パーティション化を採用しています。パーティションをドロップするときは、`show partitions`コマンドによって、データの各パーティションに存在する値の範囲を表示します。業務に関連する行がなくなったパーティションは、無効にしてデータを論理的に削除するか、またはドロップして関連するファイルとともに削除します。
- **ユーザー定義関数（UDF）**：InfiniDBでは、InfiniDBエンジン内に統合された関数として実行できる、完全な並行分散型のUDFを作成できます。これによって、InfiniDBの統合マップリダクション機能を最大限に活用したカスタム関数を作成し、分散レイヤー内の利用可能なすべてのコアにカスタマイズした機能を分散できます。
- **大規模パラレル処理（MPP）対応**：InfiniDBは複数の汎用ハードウェアマシンを使用して、総合的なパフォーマンスをリニアに向上させることができます。InfiniDB構成に安価なハードウェアを追加してデータベースの処理能力を向上させることが可能で、新しいノードを追加するだけで応答時間を半分に短縮できる場合もあります。これによって、増大するデータ量への対応、増加するユーザーアクティビティへの対応、または総合的なパフォーマンス目標達成のための調整が容易になります。さらに、InfiniDBシステムを停止またはオフラインにする必要がない、動的な方法で新しいノードを追加できます。
- **シェアードナッシング分散データキャッシュ**：複数ノードによるInfiniDB構成では、データは様々なノードとそのデータキャッシュに分散されます。ノード間でデータを共有することはありませんが、問合せのためにデータを読み取るときは、InfiniDB MPPアーキテクチャ内のすべてのノードがアクセスされます。つまり、InfiniDBは、すべての参加ノードが並行して分散形式でアクセスする、1つの大きな論理データキャッシュを作成すると考えられます。これによって、十分な数のノードと豊富なメモリーがあれば、InfiniDBは文字通り、大規模なデータベースをキャッシュできます。
- **自動フェイルオーバー**：InfiniDBのアーキテクチャには、ユーザー問合せの管理と実際のデータベースI/O操作を処理するソフトウェアモジュール用に、組み込みの自動フェイルオーバー機能が含まれています。
- **同時実行の自動スケールアウト**：構成にノードを追加すると、自動的に1台以上のマシンに同時実行をスケールアウトできます。
- **ソフトウェアパッチの自動管理**：複数のInfiniDBサーバーにパッチまたはアップグレードを適用する必要がある場合、自動処理がアップグレードが適用された最初のノードからソフトウェアを取得し、他のすべての参加ノードを自動的にアップグレードします。

ここで説明したすべての機能が、分析用データベースやデータウェアハウスを設計し実装するための専門知識を持たない初心者でも、高速に動作するデータマート、データウェアハウス、その他の分析用データベースを作成し管理できるように支援します。

## 2.4 サポートするハードウェアとソフトウェア

InfiniDBデータベースは次のハードウェアとソフトウェアによる構成をサポートします。

- IntelおよびAMDの汎用ハードウェア、64ビット
- 次のLinux
    - Red Hat Enterprise
    - CentOS
    - Debian
    - Ubuntu
    - AWS（Amazon）EC2

# 3 InfiniDB のアーキテクチャ

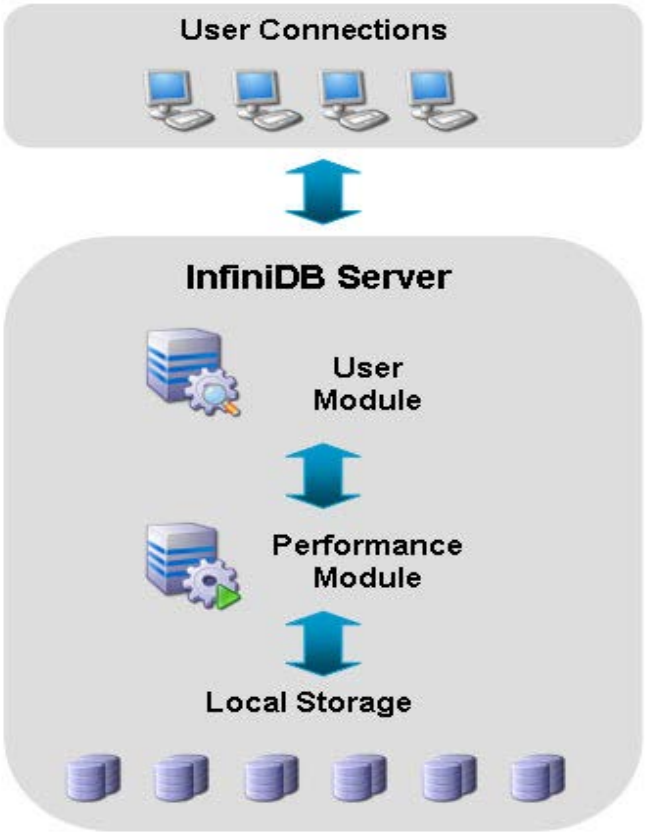
この項では、InfiniDBのアーキテクチャおよびデータベースを構成する様々なコンポーネントについて説明します。

## 3.1 InfiniDB の概要

InfiniDBのアーキテクチャは様々なコンポーネントで構成されており、すべてのコンポーネントが連携してInfiniDBデータベースを構成しています。これらのコンポーネントには次のものがあります。

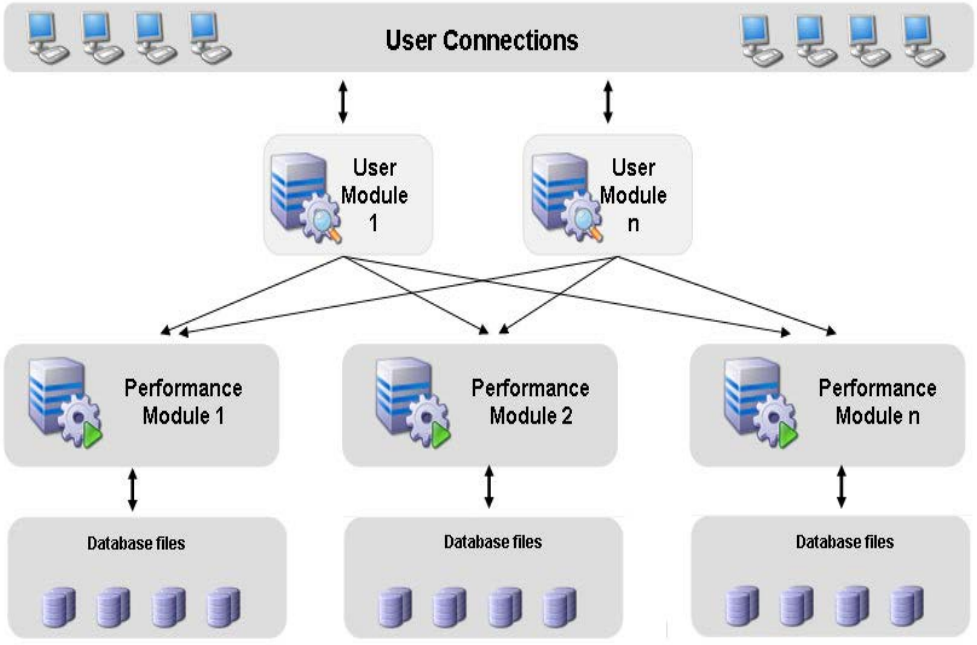
- **ユーザーモジュール**:ユーザーモジュールは、小さいMySQLインスタンス(個別にディレクタモジュールと呼ばれることもある)、および同時実行スケーリングを処理する多くのInfiniDBプロセスで構成されています。また、ユーザーモジュールは、SQLリクエストを分割して1つ以上のパフォーマンスモジュールに分散します。パフォーマンスモジュールによって、リクエストされたデータがメモリーキャッシュまたはディスクのいずれかから実際に取得されます。最後に、ユーザーモジュールは、関連するパフォーマンスモジュールからのすべての問合せ結果をまとめ、包括的な問合せ結果セットを形成してユーザーに返します。ユーザーモジュールは複数存在させることができます。この構成では、高可用性に対応した構成および問合せのワークロードバランシング用の方法を実現できます。
- **パフォーマンスモジュール**:パフォーマンスモジュールは、データの格納、取得および管理、問合せ操作に対するブロックリクエストの処理、および問合せリクエストをファイナライズするためにユーザーモジュールへのデータの引渡しを行います。パフォーマンスモジュールは、ディスクからデータを選択して、パフォーマンスモジュールが存在するサーバーの一部であるシェアードナッシング方式のデータキャッシュにキャッシュします。MPPは、ユーザーが必要な数のパフォーマンスモジュールを構成できるようにすることで実現されます。追加された各パフォーマンスモジュールによって、データベース全体にキャッシュが追加され、処理能力が高まります。
- **ストレージ**:InfiniDBは、ストレージシステムとしては非常に柔軟です。オンプレミスで実行する場合、ローカルストレージまたは共有ストレージ(SANなど)のいずれかを使用してデータを格納できます。Amazon EC2環境では、InfiniDBはエフェメラルボリュームまたはElastic Block Store(EBS)ボリュームを使用できます。シェアードナッシング展開にデータの冗長性が必要な場合、InfiniDBはGlusterFSおよびApache Hadoop分散ファイルシステム(HDFS)と統合するように設計されます。

InfiniDBデータベースには、設計可能な多くの構成があります。InfiniDBのデータベースは単一サーバーまたはMPPスケールアウトで構成されます。単一サーバーは次のように示すことができます。




前述の構成では、すべてのモジュールが同じサーバーで動作し、マシン上のすべてのCPU処理能力を活用してスケールアップした処理能力を利用できます。

MPPスケールアウトオプションには、設計可能な多くの構成があります。その1つがシェアードナッシングディスクアーキテクチャです。



前述の構成では、同時実行スケールアウトはユーザーモジュールノードを追加することで実現されていますが、I/Oスケールアウトはパフォーマンスモジュールノードを追加することで実現されています。

ユーザーモジュールとパフォーマンスモジュールは互いに近接した場所にある場合もあれば、異なるデータセンターおよび地理的な場所に分けられている場合もあることに注意してください。

## 3.2 InfiniDB のコンポーネント

この項では、InfiniDBデータベースの主要な各コンポーネントおよびデータベースの動作全体に影響する関連構造について詳しく説明します。

### 3.2.1 ユーザーモジュール

簡単に説明すると、InfiniDBユーザーモジュールは、エンドユーザーの問合せ操作の管理および制御を行います。各問合せの状態を維持し、問合せのかわりに1つ以上のパフォーマンスモジュールにリクエストを発行して作業を行い、すべての関連パフォーマンスモジュールの結果セットを集計して最終的にエンドユーザーに返す結果セットにする問合せを実行します。

#### 3.2.1.1 ディレクタモジュールおよび MySQL インタフェース

InfiniDBユーザーモジュールの個別のコンポーネントはディレクタモジュールと呼ばれる場合があります。基本的に、これはInfiniDBが次の基本機能の実行に利用するMySQLインスタンスです。

- 接続の検証
- SQL文の解析
- 一般的なSQL計画の生成
- 最終結果セットの配付

このコンポーネント内で、InfiniDBはサーバーを管理するためのいくつかのプロセスを提供します。プロセスマネージャ（ProcMgr）はすべてのInfiniDBプロセスの起動、監視および再起動を行います。また、各マシン上でプロセスモニター（ProcMon）と呼ばれる別のプロセスを使用してInfiniDBプロセスを追跡します。

InfiniDBでは、MySQL形式の問合せ計画を利用する多くの接続ライブラリを使用して問合せ計画をInfiniDB形式に変換します。InfiniDB形式も基本的には解析ツリーですが、ユーザーモジュールで解析ツリーをInfiniDBジョブリストに変換するときに役立つようにオプティマイザから実行ヒントが追加されます。また、InfiniDB用のMySQLへのMySQLストレージエンジンAPI（ハンドラ）インタフェースを提供するMySQLコネクタライブラリ（libcalmysql.so）もあります。

#### 3.2.1.2 主要なユーザーモジュール操作

ユーザーモジュールは、InfiniDBデータベースに対して次の主要な機能を実行します。

- InfiniDBジョブリストへのMySQL計画の変換
- InfiniDBシステムカタログからのInfiniDB OID（オブジェクトID）の検索の実行
- 不要なエクステントの除外によって実現されるI/Oの削減のためのエクステントマップ（詳細はこのマニュアルの後半を参照）の検査
- パフォーマンスモジュールへの指示（「プリミティブ操作」とも呼ばれる）の発行

- 必要に応じたハッシュ結合の実行(結合での小さい方の表のサイズに基づく)。処理が必要なハッシュマップをパフォーマンスモジュールに送信することによる、分散されたハッシュ結合の管理の支援
- ハッシュ結合の後に発生する表をまたいだ関数および式の実行
- パフォーマンスモジュールからのデータの受信および必要に応じたパフォーマンスモジュールへの返信
- すべての集計処理および重複行の削除処理に対する補足ステップの実行
- MySQLインタフェースへのデータの返信

ユーザーモジュールの主要なジョブは同時実行スケーリングの処理です。データベースファイルに対して直接処理が行われることはないため、データベースファイルの可視性も必要とされません。マシンのRAMを一時的に使用して部分的な問合せ結果を集計し、最終的にユーザーに返される包括的な応答にします。この時点で、問合せは1つのユーザーモジュールにのみ存在しますが、利用可能なすべてのパフォーマンスモジュールに問合せをブロードキャストして実際のI/O動作をスケールアウトできます。

#### 3.2.1.3 ユーザーモジュールのプロセス

ユーザーモジュールには、実行マネージャ(ExeMgr)、DMLマネージャ(DMLProc)、DDLマネージャ(DDLProc)およびインポート分散マネージャ(cpimport)という、いくつかのプロセスが含まれています。ExeMgrは、TCP/IPポートでディレクタモジュール(MySQLインスタンス)の問合せ解析ツリーをリスニングします。ExeMgrは、問合せ解析ツリーをInfiniDBジョブリスト(問合せへの応答に必要な一連の指示)に変換します。ExeMgrは、問合せ解析ツリーを移動しながら、ジョブステップを繰返し生成してジョブリストの最適化および再最適化を行います。ジョブステップの主要なカテゴリは、列フィルタの適用、表結合の処理および返された列の投影です。DMLProcおよびDDLProcは、適切なパフォーマンスモジュールにDMLおよびDDLを分散します。cpimportは、ユーザーモジュールで実行されている場合に、パフォーマンスモジュールにソースファイルを分散します。

計画内の各ノードは、ジョブリスト自体によって並行して実行されます。ノードは、ユーザーモジュール全体、パフォーマンスモジュール全体またはこれらの組合せに対して動作できます。各ノードでは、エクステントマップを使用して、作業指示の送信先となるパフォーマンスモジュールを決定します(この使用方法の詳細は、後述のエクステントマップに関する説明を参照してください)。これらのすべてのノードはユーザーレベルのスレッドです。各スレッドで、データおよび利用可能なシステムリソースの配分に基づいて追加のスレッドを生成できます。

#### 3.2.1.4 ユーザーモジュールのスケールアウト

InfiniDBでは、透過的なユーザーモジュールのスケールアウトを実現できます。管理者は、システムをスケーリングしてより多くのユーザー接続をサポートできるように、複数のユーザーモジュールが含まれるようにInfiniDBシステムを構成できます。

### 3.2.2 パフォーマンスモジュール

この項では、InfiniDBのパフォーマンスモジュールの様々な側面について説明します。

#### 3.2.2.1 一般的な操作

パフォーマンスモジュールは、問合せ処理および書込み処理においてI/O操作を実行します。また、実行する作業に関してユーザーモジュールから指示を受信します。パフォーマンスモジュールは、問合せ自体は確認せず、ユーザーモジュールによって与えられた一連の指示のみを確認します。この点にお

いて、パフォーマンスモジュールでは、標準的な意味での問合せの実行ではなく、ブロック指向のI/O操作の実行に重点が置かれています。

大規模パラレル処理またはスケールアウト機能は、InfiniDB構成にパフォーマンスモジュールを追加することで実現されます。特定の設定にノードを追加すると、異なるパフォーマンスモジュール間で問合せがパラレル化されるため、パフォーマンスのリニアな向上が実現されます。パフォーマンスモジュールには、データベースの動作のスケールアウトに必要となる3つの重要な動作（スキャンの分散、ハッシュ結合の分散および集計の分散）があります。これら3つの動作を組み合せることによって、問合せが集中する環境でのMPPの動作をチューニングできます。

パフォーマンスモジュールで実行されるプロセスには、PrimProc（問合せ実行を処理するプロセス）、WriteEngineServer（パラレル書込みを調整するプロセス）およびcpimport（実際のデータベースファイルの更新を実行するプロセス）の3つがあります。

#### 3.2.2.2 ロード処理および書込み処理

パフォーマンスモジュールのノードは、基礎となる永続ストレージへのロードおよび書込みを実行するタスクを割り当てられています。WriteEngineServerは、各パフォーマンスモジュールで、DML、DDLおよびインポートを調整します。cpimportは、どのモジュールで実行されているかを認識し、パフォーマンスモジュールで実行されている場合には、データベースディスクファイルの実際の更新を処理します。このように、InfiniDBは完全に並行なロード機能をサポートします。DDLの変更は、すべてのInfiniDBメタデータを追跡するInfiniDBシステムカタログ内で維持されます。InfiniDBでは、高可用性構成をサポートするために書込みノードに対してフェイルオーバー機能が提供されています。

#### 3.2.2.3 シェアードナッシング方式のデータキャッシュ

InfiniDBでは、複数のパフォーマンスモジュールでシェアードナッシング方式のデータキャッシュが使用されています。初めてデータへのアクセスが行われたときに、パフォーマンスモジュールはユーザーモジュールによって指示されたデータの量に基づいて動作し、後続のアクセス用にLRUベースのキャッシュにデータをキャッシュします。InfiniDB専用マシンでは、コンピュータのRAMの大部分をパフォーマンスモジュールのデータキャッシュ専用にできます。

パフォーマンスモジュールのキャッシュはシェアードナッシング方式で設計されているため、パフォーマンス上の主要なメリットが少なくとも2つあります。1つは、関連パフォーマンスモジュールノード間でデータブロックのpingが発生しないことです。これは複数インタンスまたは共有ディスクのデータベースシステムでは発生することがあります。もう1つは、多くのパフォーマンスモジュールノードがシステムに追加されているため、データベースの全体的なキャッシュサイズが大幅に大きくなることです。これによって、（場合によっては）データベース全体をRAMにキャッシュして非常に高速なアクセスを行うことができます。

InfiniDBがスタンドアロンサーバーで使用される場合、データキャッシュは他の標準LRUベースのデータキャッシュと同様に動作することに注意してください。単一サーバーの実装では、存在するパフォーマンスモジュールは1つのみのため、データベースからリクエストされたすべてのデータがキャッシュを介して渡されます。

#### 3.2.2.4 パフォーマンス向上のための機能

パフォーマンスモジュールの追加は、追加サーバー用のユーザーモジュールで一連の基本構成手順を実行すると実現されます。この処理はInfiniDBシステム全体で透過的であり、通常新しいパフォーマンスモジュールノードは60秒以内に追加されます。

#### 3.2.2.5 パフォーマンスモジュールのフェイルオーバー

複数ノードのパフォーマンスモジュール構成では、すべてのノードがオンラインであり、特定のパフォーマンスモジュールでエラーが発生した場合に透過的なフェイルオーバーが行われることがハートビートメカニズムによって保証されます。パフォーマンスモジュールが異常終了した場合、処理中の問合せでエラーが発生します。パフォーマンスモジュールでエラーが発生したためエラーを受信したユーザーは、問合せを再発行して、残りのパフォーマンスモジュールを使用して作業を実行できます。

フェイルオーバーが発生した場合、基礎となるストレージデータが外部にマウントされている場合（EC2 EBS、SANなど）、動作しているパフォーマンスモジュール間でパフォーマンスモジュールのデータブロックのマッピングが再編成され、ユーザーモジュールのエクステントマップが再評価されることにより、残りの適切なパフォーマンスモジュールに問合せが送信されます。この処理はユーザーに対して透過的であり、手動操作は必要ありません。

エラーが発生したパフォーマンスモジュールがオンラインに戻ると、InfiniDBはそのパフォーマンスモジュールを自動的に構成に戻し、作業に使用し始めます。

#### 3.2.2.6 ユーザーモジュールおよびパフォーマンスモジュールを使用した問合せ処理の概要

InfiniDBによるエンドユーザーの問合せの処理方法のトップダウンの見解は次のとおりです。

1. MySQLインタフェースを介してリクエストを受信します。MySQLによって、リクエストの処理に必要なすべての表に対する表操作が実行され、最初の問合せ実行計画が取得されます。
2. InfiniDBで、MySQLストレージエンジンインタフェースを使用してMySQLの表オブジェクトがInfiniDBオブジェクトに変換されます。その後、これらのオブジェクトはユーザーモジュールに送信されます。
3. ユーザーモジュールで、MySQL実行計画が変換され、これらのオブジェクトがInfiniDB実行計画に最適化されます。ユーザーモジュールで、問合せの実行に必要な手順およびそれらを実行するタイミングが決定されます。
4. ユーザーモジュールで、問合せに必要なデータの場所に関してエクステントマップが参照され、エクステントマップ内に含まれている情報に基づいてエクステントの除外が実行されます。
5. ユーザーモジュールで、1つ以上のパフォーマンスモジュールにコマンドが送信されてブロックのI/O操作が実行されます。
6. パフォーマンスモジュールで、条件のフィルタ処理、結合処理、データの初期集計が実行され、最終結果セットの処理のためにユーザーモジュールにデータが送信されます。
7. ユーザーモジュールで、最終結果セットの集計が実行され、問合せの最終結果セットが作成されます。
8. ユーザーモジュールから、ユーザーに結果セットが返されます。

### 3.3 InfiniDB のストレージ概念

この項では、InfiniDBデータベースの基本的なストレージ概念について説明します。

### 3.3.1 InfiniDB データの一般的なストレージ概念

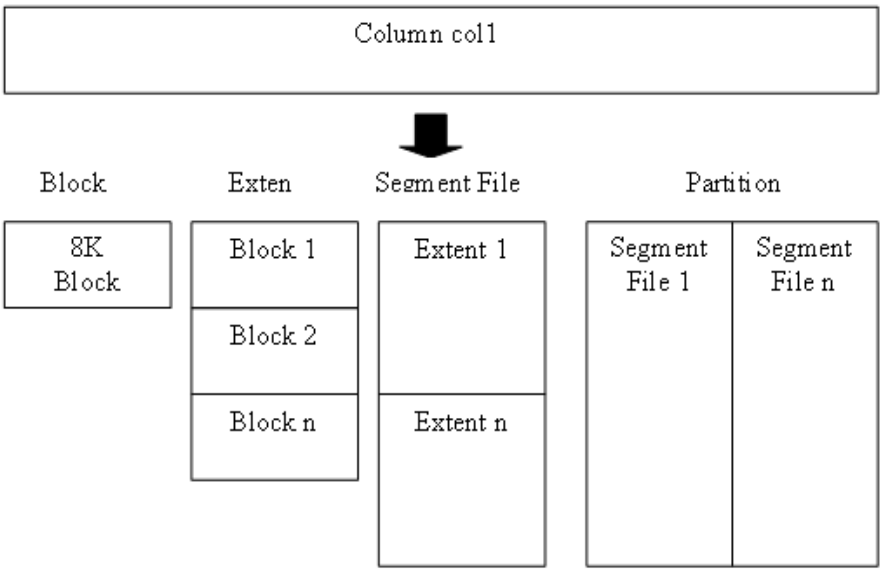
InfiniDBは列指向データベースであるため、表が作成されると、システムによって表の列ごとに1つ以上のファイルが作成されます。したがって、たとえば、3つの列を含む表が作成されると、個別にアドレス指定可能な論理オブジェクトが少なくとも3つ、SANまたはサーバーのディスクドライブに作成されます。

次に、InfiniDBデータベースのストレージビルディングブロックを下位から上位の順に示します。

- **ブロック**:ブロック(一部のRDBMSではページとも呼ばれる)は、サイズが8KBの物理オブジェクトです。データベース内のすべてのデータベースブロックは論理ブロック識別子(LBID)で一意に識別されます。InfiniDBのブロックサイズは設定変更できませんが、InfiniDBの先読みメカニズムで読み取られるブロックの数はカスタマイズできます。
- **エクステント**:エクステントは、1つの物理的なセグメントファイル内に存在する800万行の論理的測定単位です。エクステントは、1バイトのデータ型では8MB、2バイトのデータ型では16MB、4バイトのデータ型では32MB、8バイトのデータ型では64MB、可変サイズのデータ型では64MBが消費されます。エクステントが一杯になると、新しいエクステントが自動的に作成されます。
- **セグメントファイル**:セグメントファイルは、列のデータを保持するディスク上の物理ファイルです。セグメントファイルがエクステントの最大数に到達すると、新しいセグメントファイルが自動的に作成されます。
- **パーティション**:パーティションは、表または索引に存在するデータを水平に分割するためにパーティション化を使用する従来の行ベースのデータベースの概念とは異なる概念です。InfiniDBでは、パーティションは論理オブジェクトであり、1つ以上のセグメントファイルで構成されます。1つのパーティションに含めることができるセグメントファイルの数は`FilesPerColumnPartition`パラメータによって制御されます。列に含めることができるパーティションの数に制限はありません。

InfiniDBのストレージ概念は、次のような図で表すことができます。

```
create table t1 (col1 int, col2 varchar(20), col3 date) engine=infinidb;
```




ディスクのサブシステムへのセグメントファイルの配置に関して、InfiniDBストレージに使用するディスクリソースをInfiniDB構成ファイル内にディレクトリで指定できます。InfiniDBでは、データの物理的な読取り率が最大化されるように、指定したすべてのディスクリソースにすべてのセグメントファイルが自動的に分散されます。

### 3.3.2 圧縮とリアルタイム解凍

InfiniDBは、優れた圧縮機能を提供するとともに、ディスクからの読取りのパフォーマンスを最適化する圧縮方法を採用しています。これによって、I/Oバウンドのシステムで、ディスクからの読取りのパフォーマンスを向上させることができます。

圧縮はデフォルトで有効になっていますが、表レベルまたは列レベルで有効と無効を切り替えることができます。また、変数*infinidb_compression_type*を設定してセッションレベルで制御することもできます。デフォルト設定では、リリース2.0より後のリリースで作成されたすべての表に対して圧縮が有効になります。以前の既存の表は変更されないことに注意してください。圧縮されていない以前の表を圧縮するには、新しい表を圧縮して作成した後、データを選択し、高速インポートメソッド（cpimport）を使用して圧縮した新しい表にロードします。その後、表の名前を変更し、圧縮されていない表を削除できます。

列指向のストレージでは、類似したデータが各列ファイルに格納されるため、優れた圧縮性が実現されます。ほとんどのデータセットで65-95%の領域を節約する優れた圧縮率が示されますが、実際に節約される領域は、データのランダムさおよび存在する個別値の数によって異なります。

InfiniDBの圧縮方法は、解凍速度を高速化してディスクからの読取りパフォーマンスが最大化されるようにチューニングされています。これは、通常WORM（Write Once Read Many）であるシステムでの標準的なデータウェアハウスの使用に理想的です。ディスクからの読取りの高速化は、ベストプラクティスに従い、既存のデータ操作を最小化するシステムに適しています。初期ロード後に頻繁に更新される列については、その列が未圧縮と定義されている場合のほうが、InfiniDBでは更新のパフォーマンスが向上することがあります。列の追加または削除に、列データベースの表を再構築する必要はありません。したがって、複数のオプションについて並行して評価されます。

### 3.3.3 バージョンバッファおよびファイル

InfiniDBは、変更対象のディスクブロックを格納するためにバージョンバッファと呼ばれる構造を使用します。このバージョンバッファは、InfiniDBがデータベースの問合せで一貫したビューを提供できるように、トランザクションのロールバックアクティビティを管理し、データベースのMVCC（多版型同時実行制御）または「スナップショット読取り」機能を提供するためにも使用されます。InfiniDBのすべての文は、システムによってシステム変更番号（SCN）として参照されるデータベースの特定のバージョン（またはスナップショット）で実行されます。これはバージョンバッファと呼ばれますが、メモリー構造とディスク構造の両方で構成されています。

バージョンバッファでは、メモリー内のハッシュ表を使用して処理中のトランザクションの情報へのメモリーアクセスが提供されます。スタートアップ時の初期サイズは4MBです。メモリー領域は、トランザクションによって変更されたブロックを処理するためにこのサイズから増えていきます。ハッシュ表の各エントリは、変更された8Kのブロックに対する40バイトの参照情報です。

更新対象の行数はバージョンバッファに対する制限要因にはなりません。むしろ更新対象のディスクブロック数に注意する必要があります。このサイズは大きくできますが、更新するディスクブロック数が増えるとUPDATE文またはDELETE文の実行時間が長くなるうえ、問題が発生した場合のロールバックにかかる時間も長くなるため、注意する必要があります。

バージョンバッファファイルは、各DBRootでサイズが1GBのファイルにデフォルト設定されます。これは、`VersionBufferFileSize`パラメータを使用して構成できます。バージョンバッファファイルは、システムの各DBRootに分散されます。

**注意**：InfiniDBがHDFSでHadoop問合せエンジンとして実行するように構成されている場合、MVCC機能（およびバージョンバッファファイルの使用）は無効になります。HDFSは書込み専用のファイルシステムであるため、MVCCモデルにおけるブロックレベルのバージョニングの運用は実用的ではありません。InfiniDBは、HDFSで実行する場合に、DML操作に対する文レベルの追跡およびロールバック機能をサポートしています。HDFSでの問合せは、「結果整合性」モードより有効です。結果整合性モードでの問合せは、更新が完了した時点で正しい結果を得ることに集中しますが、HDFSでの問合せでは、更新と同時に発行された問合せは、ブロックレベルでの実行が行われた時点で最新のブロックを使用します。

### 3.3.4 トランザクションログ

トランザクションを管理し、クラッシュリカバリの状況が発生した場合にデータベースに対してロールフォワード操作を実行するために、InfiniDBではトランザクションログが使用されます。トランザクションログには、DDL文とDML文のエントリおよびバルクロード操作のマーカーが含まれています。LinuxでのInfiniDBの現行の実装では、トランザクションログは`data_mods.log`という名前で`/var`ファイルシステムにインストールされています。

バックアップおよびリカバリの目的で、トランザクションログのアーカイブが時間ベースの間隔で実行されます。これは、InfiniDB構成ファイル（`TransactionArchivePeriod`）で構成できます。各トランザクションログダンプの末尾には、マシン上で一意のファイルになるようにタイムスタンプが追加されます。トランザクションログがダンプされた後、アーカイブログは切り捨てられ、新しいエントリに対応できるようになります。

トランザクションログの情報を表示するために、ユーザーは実行されたDDL文およびDML文についてレポートする`ReplayTransactionLog`ユーティリティを使用することができます。バルクロード操作はトランザクションログには含まれないことに注意してください。

## 3.4 エクステントマップ

問合せの処理速度を高めるために、InfiniDBではエクステントマップと呼ばれるスマート構造が使用されます。これによって、行ベースのデータベースでそのアーキテクチャによる制約をなくすために実装が必要な索引付け、手動での表のパーティション化、マテリアライズドビュー、サマリー表、およびその他の構造とオブジェクトが必要なくなります。

前述のとおり、エクステントは物理的なセグメントファイル内に存在する領域の論理ブロックで、サイズは8-64MBの間です。各エクステントはディスク上のより少ないスペースを使用して、より小さいデータ型で、同じ数の行をサポートしています。エクステントマップには、すべてのエクステントおよびそれらに対応するブロック（LBID）がカタログ化されます。また、エクステントマップには、エクステント内の列のデータの最小値および最大値が保持されます。

InfiniDBでは、問合せリクエストを処理するためにエクステントマップおよびバージョンバッファの両方が使用されることに注意してください。これら2つの構造の組合せはブロック解決マネージャ（BRM）と呼ばれます。

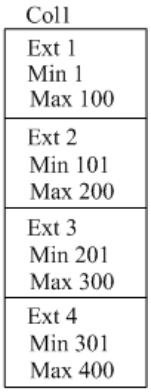
エクステントマップのマスターコピーはプライマリのパフォーマンスモジュールに存在します。システムのスタートアップ時、ファイルはメモリーに読み取られた後で、ディザスタリカバリおよびフェイルオーバーに備えて関連するその他すべてのユーザーモジュールおよびパフォーマンスモジュールに物理的にコピーされます。すばやくアクセスできるように、すべてのノードでエクステントマップがメモリーに保持されます。エクステントが変更されると、すべての参加ノードにも更新がブロードキャストされます。

### 3.4.1 エクステントマップの機能

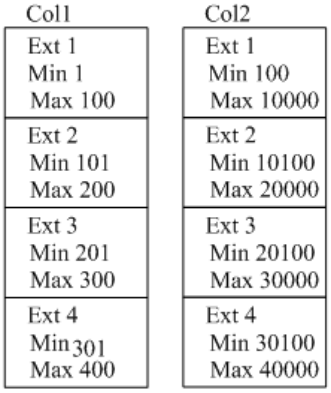
エクステントマップを使用すると、InfiniDBで問合せへの対応に必要なブロックのみを取得できますが、それとは別に論理的レンジパーティション化というメリットもあります。これは、エクステントマップ内に含まれている各エクステントの最小値および最大値によって実現されます。InfiniDBでは、まず（必要な列のみがスキャンされる）列指向のアーキテクチャによってエクステントの除外が実行されますが、エクステントマップで実装される論理的水平パーティション化のため、この処理は高速化されます。

この自動エクステント除外の動作は、データが頻繁にロードされ、時間に基づいて参照されることが多い一連のデータ、順序付けられたデータ、パターン化されたデータまたは時間ベースのデータに適しています。データの先頭に対する問合せでほぼリアルタイムでロードすることによって、すべての日時列および昇順のキー値に適したエクステント除外を簡単に示すことができます。クラスタ化された値を含む列はエクステント除外に適しています。

列スキャンにフィルタが設定されているときにエクステントを除外する場合、そのフィルタの値は列の各エクステントに格納されている最小値および最大値と比較されます。




前述の図では、「COL1 BETWEEN 220 AND 250」というWHERE列フィルタが指定されている場合、InfiniDBではエクステント1、2および4がスキャンから除外されるため、I/Oの4分の3が節約され、多くの比較操作が省略されます。エクステントにフィルタと一致する行が含まれている可能性がない場合、そのエクステント全体が無視されます。また、1つの列に対して除外された各エクステントによって表内の他のすべての列の同じエクステントも除外されるため、使用できないフィルタチェーンをI/Oの必要なく簡単に識別できます。たとえば、次の2つの列およびそれらのエクステントマップ情報について考えてみます。

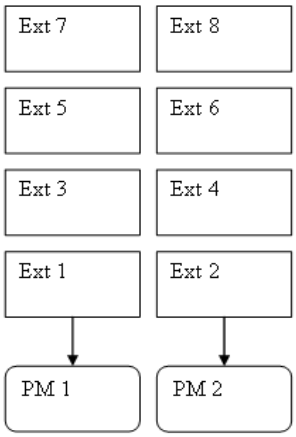



「COL1 BETWEEN 220 AND 250 AND COL2 < 10000」というWHEREフィルタが指定されている場合、InfiniDBでは、エクステント1、2および4が最初の列フィルタから除外されます。その後、COL2（つま

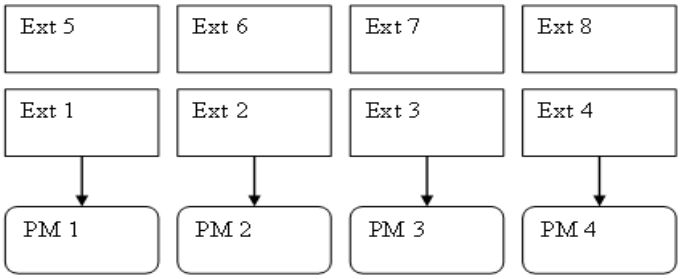
りエクステント3のみ）に対して一致するエクステントのみが検索され、一致するエクステントが存在しないことが確認されるとI/Oを実行することなく0行が返されます。

### 3.4.2 I/O 処理およびワークロードの分散

ブロックはエクステント内に割り当てられており、エクステントはディスク上の連続領域（セグメントファイル）にマップされるため、任意の数のパフォーマンスモジュールに作業を均等に分散できます。たとえば、2つのパフォーマンスモジュールシステムでは、8つのエクステントが含まれている列のスキャンは次のように分散されます。




4つのパフォーマンスモジュールシステムでは次のようになります。




このアーキテクチャでは、InfiniDBによって作業がすべての関連パフォーマンスモジュールに適切に分散されるため、多くの問合せにおいてパフォーマンスをリニアに向上させることができます。

大規模なスキャン操作のために、InfiniDBでは、`ColScanReadAheadBlocks`パラメータによって構成される先読みメカニズムを採用しています。このパラメータは、現在512にデフォルト設定されています。ただし、問合せへの対応に必要でない可能性があるデータの読取りを回避するために、InfiniDBではプリフェッチしきい値パラメータ（`PrefetchThreshold`、デフォルト設定は5）も使用されます。これによって、問合せに必要なブロックの数がしきい値を下回っていることがシステムで確認された場合、必要なブロックのみが読み取られます。

もう1つの注意点は、InfiniDBではハッシュ結合がサポートされているという点です。このマニュアルの作成時点では、ハッシュ結合はMySQLではサポートされていません。また、InfiniDBのハッシュ結合は、複数ノードのMPP構成が使用されている場合、分散して並行で処理できます。

# 4 データベースの管理

InfiniDBには、InfiniDBデータベースに管理コマンドを発行するために使用されるInfiniDB管理(コマンド)コンソールが付属します。管理者は、これを使用して1つ以上のInfiniDBサーバーの構成、監視および管理を行うことができます。ユーティリティ名は`cmconsole`で、InfiniDBバイナリに対する実行権限を持つ任意のオペレーティングシステムユーザーが起動できます。

サポートされているコマンドに関するヘルプを参照するには、**`help`**と入力してコマンドのリストを確認します。特定のコマンドのヘルプは、**`help`**に続けて目的のコマンドを入力すると参照できます。

```
InfiniDB> help
help   Fri Oct 26 14:11:12 2012

List of commands:
Note: the command must be the first entry entered on the command line

Command                          Description
-------------------------------  ------------------------------------------------------------
?                                Get help on the Console Commands
addDbroot                        Add DBRoot Disk storage to the Calpont InfiniDB System
addExternalDevice                Add External Device to Configuration file
addModule                        Add a Module within the Calpont InfiniDB System
alterSystem-disableModule        Disable a Module and Alter the Calpont InfiniDB System
alterSystem-enableModule         Enable a Module and Alter the Calpont InfiniDB System
assignDbrootPmConfig             Assign unassigned DBroots to Performance Module
assignElasticIPAddress           Assign Amazon Elastic IP Address to a module
disableLog                       Disable the levels of process and debug logging
enableLog                        Enable the levels of process and debug logging
exit                             Exit from the Console tool
getActiveAlarms                  Get Active Alarm list
getActiveSQLStatements           Get List Active SQL Statements within the System
getAlarmConfig                   Get Alarm Configuration Information
getAlarmHistory                  Get system alarms
getAlarmSummary                  Get Summary counts of Active Alarm
getCalpontSoftwareInfo           Get the Calpont InfiniDB RPM detailed information
getExternalDeviceConfig          Get External Device Configuration Information
getLogConfig                     Get the System log file configuration
getModuleConfig                  Get Module Name Configuration Information
getModuleCpu                     Get a Module CPU usage
getModuleCpuUsers                Get a Module Top Processes utilizing CPU
getModuleDisk                    Get a Module Disk usage
getModuleMemory                  Get a Module Memory usage
getModuleMemoryUsers             Get a Module Top Processes utilizing Memory
getModuleResourceUsage           Get a Module Resource usage
getModuleTypeConfig              Get Module Type Configuration Information
getProcessConfig                 Get Process Configuration Information
getProcessStatus                 Get Calpont InfiniDB Process Statuses
getStorageConfig                 Get System Storage Configuration Information
getStorageStatus                 Get System Storage Status
getSystemConfig                  Get System Configuration Information
getSystemCpu                     Get System CPU usage on all modules
getSystemCpuUsers                Get System Top Processes utilizing CPU
getSystemDisk                    Get System Disk usage on all modules
getSystemInfo                    Get the Over-all System Statuses
getSystemMemory                  Get System Memory usage on all modules
getSystemMemoryUsers             Get System Top Processes utilizing Memory
getSystemNetworkConfig           Get System Network Configuration Information
getSystemResourceUsage           Get System Resource usage on all modules
getSystemStatus                  Get System and Modules Status
help                             Get help on the Console Commands
monitorAlarms                    Monitor alarms in realtime mode
movePmDbrootConfig               Move DBroots from one Performance Module to another
quit                             Exit from the Console tool
```

InfiniDBコマンドの全リストについては、『InfiniDB管理者ガイド』を参照してください。

## 4.1 SQL インタフェース

InfiniDBデータベースに対するSQLコマンドは、`idbmysql`ユーティリティを使用して入力できます。このコマンドラインツールは、用途と機能の点で、標準MySQLの`mysql`コマンドラインユーティリティに類似しています。すべての標準SQL文(`SELECT`、`INSERT`、`UPDATE`、`DELETE`、`GRANT`など)は、このユーティリティを介して発行できます。

## 4.2 InfiniDB インスタンスの起動および停止

関連するすべてのユーザーモジュールおよびパフォーマンスモジュールの起動および停止は、`cmconsole`ユーティリティを使用して非常に簡単に実行できます。主な3つのコマンドを次に示します。

- `startsystem`
- `stopsystem`
- `restartsystem`

管理者は、基本的なオペレーティングシステムのサービス起動コマンドを活用することもできます。たとえば、管理者は、個々のLinuxマシンで`service infinidb start`コマンドを発行してInfiniDBインスタンスを起動できます。`service infinidb stop`を実行した場合はInfiniDBインスタンスが停止されます。

## 4.3 セキュリティの概念

この項では、InfiniDBのセキュリティの概念およびプラクティスについて簡単に説明します。

### 4.3.1 オペレーティングシステムのセキュリティ

InfiniDBは、実行可能ファイルおよび基礎となるデータベースファイルの保護に関して、オペレーティングシステムの標準的なセキュリティを使用します。このレベルでは、特殊なものは使用されません。InfiniDBデータベースをインストールして管理する管理者は、InfiniDBバイナリまたはデータベースファイルを含むファイルシステムやディレクトリへのアクセス権をどのオペレーティングシステムアカウントに付与するかという点に関して、標準のセキュリティ手順に従う必要があります。

### 4.3.2 データベースのログインアクセス権

InfiniDBでは、標準MySQLのセキュリティコマンドを使用して、InfiniDBデータベースに対するユーザーの追加および削除を行います。ユーザーアカウントには、システム全体および作業する特定のデータベースへのアクセス権が付与されます。

たとえば、「rms」という名前の新しいユーザーに、システムへのアクセス権および「db1」という名前のデータベースに対する完全な権限を付与する場合、管理者は次のコマンドを入力します。

```
mysql> grant all on db1.* to rms identified by ‘password’;
Query OK, 0 rows affected (0.00 sec)
```

InfiniDBデータベースからユーザーアカウントを削除する場合、管理者は`drop user`コマンドを使用します。

ユーザーアカウントの追加および削除の詳細は、最新オンライン版のMySQLのリファレンスマニュアルを参照してください。

### 4.3.3 オブジェクトのセキュリティ

データベースやシステムのアクセスセキュリティと同様に、InfiniDBでは標準MySQLのセキュリティコマンドを使用します。これには、選択されたユーザーアカウントに対してオブジェクトレベルの権限を付与および削除する、標準SQLのGRANTおよびREVOKEコマンドが含まれます。たとえば、「db1」データベースの「t1」という名前の表に対するすべての権限を「rms」という名前のユーザーに付与する場合、管理者は次のコマンドを発行します。

```
mysql> grant all on db1.t1 to rms;
Query OK, 0 rows affected (0.00 sec)
```

GRANTおよびREVOKEの機能の全リストについては、最新オンライン版のMySQLのリファレンスマニュアルを参照してください。

## 4.4 InfiniDB 構成ファイル

InfiniDB構成ファイルには、InfiniDBデータベースの操作を制御するパラメータが含まれます。これはXMLベースのファイルで、システムのスタートアップおよびシャットダウン時に、InfiniDBシステム内の関連するすべてのノードで自動的にレプリケートされます（または明示的なコマンドを発行してファイルをレプリケートします）。構成ファイルのマスターコピーは、プライマリのパフォーマンスモジュールに保持されます。システムは、スタートアップ時に構成ファイルから情報を読み取って、システムのアクティビティに必要なリソース（データキャッシュなど）を割り当てます。

構成ファイルをテキストエディタで直接編集することはお薦めしません。かわりに、ファイルを変更するためのコマンドラインツールconfigxmlが提供されています。スクリプトに対して各種パラメータを使用して、構成情報を表示および設定します。

たとえば、アクティブなトランザクションログのアーカイブログがディスクに書き込まれる時間間隔の値（分単位で設定）を表示する場合、DBAは次を入力します。

```
[root@server1 bin]$ ./configxml.sh getconfig SystemConfig
TransactionArchivePeriod
Current value of SystemConfig / TransactionArchivePeriod is 10
```

アーカイブログの時間間隔を60分に変更するには、setconfigオプションを使用して次を入力します。

```
[root@server1 bin]$ ./configxml.sh setconfig SystemConfig
TransactionArchivePeriod 60
Old value of SystemConfig / TransactionArchivePeriod is 10

/usr/local/Calpont/etc/Calpont.xml backed up to /usr/local/
Calpont/etc/Calpont.xml.1255027253

Old value of SystemConfig / TransactionArchivePeriod is 10
SystemConfig / TransactionArchivePeriod now set to 60
```

InfiniDBで使用される構成変数の全リストについては、『InfiniDB管理者ガイド』を参照してください。

## 4.5 新しいデータベースの作成

InfiniDBのパラダイムは、サーバーのインスタンスが実行されて、データベースがそのインスタンスの下で管理されるという点で、MySQLおよびMicrosoft SQL Serverに類似しています。インスタンスとデータベースには1対多の関係があります。これは、インスタンスとデータベースが通常1対1であるOracleと異なります。

InfiniDBでは、新しいデータベース(MySQLではスキーマと呼ばれることもある)を非常に簡単に作成できます。

```
mysql> create database db1;
Query OK, 1 row affected (0.00 sec)
```

作成後、次の例のように、データベースに表を作成できます。

```
mysql> use db1
Database changed
mysql> create table t1 (c1 int, c2 date) engine=infinidb;
Query OK, 0 rows affected (0.03 sec)

mysql> desc t1;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| c1    | int(11) | YES  |     | NULL    |       |
| c2    | date    | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

## 4.6 管理ユーティリティ

InfiniDBでは、様々なコマンドライン管理ユーティリティが提供されています。よく使用されるユーティリティを以下に示します。

- **colxml**:バルクロードユーティリティcpimportによって使用されるジョブファイルを作成します
- **configxml.sh**:構成ファイルのパラメータの読取りおよび更新に使用されます
- **cpimport**:データベースへのデータのバルクロードに使用されます
- **dumpcol**:列データを出力します
- **edititem**:エクステントマップのレポートおよび編集に使用されます
- **edittxn**:システム変更番号(SCN)を出力し、複数ノード構成での一貫性の検証に役立ちます
- **file2oid.sh**:列のいずれかのデータファイルが指定されている場合に、その列のオブジェクトIDを出力します
- **oid2file**:オブジェクトの最初のファイルのディレクトリ情報を表示します
- **replaytransactionlog**:アーカイブされたトランザクションログおよび現在のトランザクションログのすべてのトランザクション情報を含むファイルを作成します

## 4.7 システムログ

InfiniDBは、次のようないくつかのログにシステム情報を書き込みます。

- サーバーの操作に関する一般情報を含む情報ログ(`info.log`)
- サーバーで発生したエラーに関するデータを含むエラーログ(`err.log`)
- 将来的なシステムエラー(サーバーのディスク領域の不足など)を示す可能性があるサーバーの状態を通知する警告ログ(`warning.log`)

InfiniDBでは、これらのファイルは一般的なロギングディレクトリに格納されます(たとえば、Linuxの場合、この場所は`/var/log/Calpont`です)。

## 4.8 バックアップおよびリカバリ

この項では、InfiniDBのバックアップおよびリカバリの操作について説明します。

### 4.8.1 バックアップ手順の概要

InfiniDBシステムの全体バックアップを作成する場合の基本的な手順は次のとおりです。

- システムでの書込みアクティビティを一時的に停止します(`cmconsole`ユーティリティを使用して実行)
- MySQLインスタンスをバックアップします
- InfiniDBデータベースファイルをバックアップします
- システムでの書込みアクティビティをリストアします(`cmconsole`ユーティリティを使用して実行)

### 4.8.2 データベースの書込みの一時停止

InfiniDBシステムでデータベースの書込みを一時停止するには、`cmconsole`内で次のコマンドを発行します。

```
calpont> suspendDatabaseWrites
suspenddatabasewrites   Wed Sep 30 08:58:06 2009

This command suspends the DDL/DML writes to the InfiniDB Database
          Do you want to proceed:(y or n) [n]: y

Suspend InfiniDB Database Writes Request successfully completed
```

### 4.8.3 MySQL インスタンスのバックアップ

MySQLデータベースは、いくつかの方法を使用してバックアップできます。

方法1:
MySQLが提供するユーティリティ(`mysqldump`や`mysqlhotcopy`など)を使用します。InfiniDBではMySQL内のデータベースデータは格納されないため、MySQLインスタンスのバックアップは迅速に実行する必要があることに注意してください。

MySQLのバックアップおよびリカバリの詳細な処理については、最新オンライン版のMySQLのリファレンスマニュアルを参照してください。

方法2:
MySQLフロントエンドにはInfiniDB固有の実装はされないため、通常の手順のかわりに、次のディレクトリをバックアップします。

**/usr/local/Calpont/mysql/db**

例:

```
cp -rp /usr/local/Calpont/mysql/db /mnt/InfiniDB/backup/frontend
```

注意:-rpオプションは、ディレクトリを再帰的にコピーし、所有権情報を保存するためのものです。

選択したデータベースのみをバックアップする場合は、かわりに前述のディレクトリ内のデータベースディレクトリをバックアップします。

### 4.8.4 InfiniDB データファイルのバックアップ

InfiniDBデータファイルをバックアップするには、そのファイルをバックアップ場所にコピーするか、システムまたはベンダーが提供するバックアップまたはスナップショットのユーティリティを使用してInfiniDBデータが存在するディレクトリのスナップショットを作成します。これらの場所の次のファイルをバックアップする必要があります。

- すべてのデータファイルおよびDBRMファイル。たとえば、Linuxの標準インストールの場合、すべてのファイルはPMごとに/usr/local/Calpont/data*N*(*N*はdata1、data2などの一意のディレクトリを表す)にあります。

注意:InfiniDBがHDFSで実行するように構成されている場合、/usr/local/Calpont/data*N*ディレクトリはHDFSにあり、Linuxホストファイルシステムにはありません。

### 4.8.5 データベースの書込みの再開

バックアップの完了後にデータベースの書込みアクティビティを再開するには、cmconsole内で次のコマンドを発行します。

```
calpont> resumeDatabaseWrites
resumedatabasewrites   Wed Sep 30 08:58:23 2009

This command resumes the DDL/DML writes to the InfiniDB Database
          Do you want to proceed:(y or n) [n]: y

Resume InfiniDB Database Writes Request successfully completed
```

## 4.9 リカバリ手順の概要

InfiniDBデータベースの完全リストアを行うには、次の手順を実行します。

- MySQLインスタンスをリストアします
- InfiniDBデータベースをリストアします

## 4.10 MySQL インスタンスのリストア

MySQLデータベースは様々な方法でリストアできますが、最も一般的な方法は、`mysqldump`によって作成したバックアップファイルを取得し、`mysql`コマンドラインユーティリティからそのファイルを実行することです。これによって、そのファイル内に含まれるすべてのSQL文が実行されます。

MySQLのバックアップおよびリカバリの詳細な処理については、最新オンライン版のMySQLのリファレンスマニュアルを参照してください。

### 4.10.1 InfiniDB データファイルのリストア

InfiniDBデータファイルをリストアするには、そのファイルをバックアップ場所からコピーするか、システムまたはベンダーが提供するスナップショットユーティリティを使用して、InfiniDBデータが存在したディレクトリのスナップショットをリストアします。リストアするファイルには次のものがあります。

- すべてのデータファイルおよびDBRMファイル。たとえば、Linuxの標準インストールの場合、すべてのファイルはPMごとに`/usr/local/Calpont/data`*`N`*(*N*はdata1、data2などの一意のディレクトリを表す)に戻されます。

## 4.11 GUI ツールのサポート

現在、InfiniDB は、InfiniDB データベースで動作する GUI ツールを提供していません。一般的なデータモデリング、問合せの開発、一般的なMySQL のセキュリティおよびオブジェクト管理用に、開発者または DBA は MySQL から無償の GUI ツール(MySQL Workbench および古い GUI ツールのバンドル)をダウンロードできます。

また、開発者または DBA が InfiniDB データベースオブジェクトに対して問合せや管理を実行できるツールが多数のサードパーティのソフトウェアベンダーによって作成されています。SQLYog、Quest Software (TOAD for MySQL)、Mentat Technologies(DreamCoder for MySQL)など、多数のベンダーが、無償版の GUI ツールおよび高度な機能を含む有償エディションの両方を提供しています。

# 5 オブジェクト管理

この項では、InfiniDBデータベースオブジェクトの作成および一般的な管理について説明します。

## 5.1 表

リレーショナルデータベースであるInfiniDBは論理表構造でデータを格納します。InfiniDBと他の多くのRDBMSの主な違いは、InfiniDBでは列指向の構造が使用されることです。これは、従来の行ベースの実装より、分析環境またはデータウェアハウス環境に適しています。ただし、データベース表の作成に関しては、InfiniDBと他のリレーショナルデータベースに表面上の違いはありません。表内のデータの作成、移入、操作および表示には、標準のSQL文、DDL文またはDML文が使用されます。

たとえば、列が3つの「db1」というデータベース内に新しい表を作成する場合、ユーザーは次のように入力できます。

```
mysql> use db1
Database changed
mysql> create table t1 (c1 int, c2 date, c3 varchar(20)) engine=infinidb;
Query OK, 0 rows affected (3.03 sec)
```

InfiniDBによって各表に連続した領域が事前に割り当てられるため、`create table`コマンドの実行では、表が空で作成されている場合でも表に領域が事前に割り当てられない他のデータベースシステムより少し時間がかかることに注意してください。

作成後、ユーザーはInfiniDBのバルクロードユーティリティを使用して大量のデータを迅速に表にロードしたり、MySQLユーティリティを使用してデータを表にロードしたりできます。また、1つのDML文を使用してデータの挿入、更新および削除を行うこともできます。表への領域割当てに関してユーザーによる操作は必要ありません。InfiniDBによる領域の割当て方法の詳細は、InfiniDBのストレージ概念に関する前述の項を参照してください。

InfiniDBの現行の実装では、主キー、外部キーまたは列レベルの制約はサポートされていません。このサポートは今後のリリースのサーバーで追加されます。

CREATE TABLEコマンドの処理の詳細は、『InfiniDB SQL構文ガイド』またはMySQLのリファレンスマニュアルを参照してください。

### 5.1.1 表の変更および削除

InfiniDBでは、表の列の追加および削除に対して標準のALTER TABLEコマンドがサポートされています。表の変更は、標準のMySQL（または他のデータベース）よりInfiniDBでの方が短時間で実行されます。これは、InfiniDBでは索引付けが使用されないので操作時に索引の保守を実行する必要がないためです。

また、InfiniDBでは、特別な方法を使用してオンラインで表に新しい列を追加することもできます。操作時、表の読取り操作は中断なく続行されます。オンラインで表に新しい列を追加するために`calonlinealter`関数コールが使用された後、操作が完了するとMySQLデータディクショナリが同期化されます。たとえば、新しい整数列（c7）を表（t1）にオンラインで追加するには、ユーザーは次のコマンドを発行します。

```
select calonlinealter('alter table t1 add column c7 int;');
alter table t1 add column c7 int comment 'schema sync only';
```

表の削除は、単純なDROP TABLEコマンドを使用して実行します。InfiniDBでは、表の削除時に、基礎となる物理データファイルもすべてサーバーから削除されるため、後で管理者が手動クリーンアップを実行する必要はありません。

ALTER TABLEおよびDROP TABLEコマンドの処理の詳細は、『InfiniDB SQL構文ガイド』またはMySQLのリファレンスマニュアルを参照してください。

## 5.2 ストアドプロシージャおよび SQL 関数

InfiniDBでは、MySQLのストアドプロシージャおよびSQL関数がサポートされています。ストアドプロシージャは、サーバー自体に格納されているプロシージャコードオブジェクトであり、データのプログラムによる処理の実行および機密データに対するセキュリティの強化のために使用される場合があります。DBAは、ストアドプロシージャに実行権限を付与してアクセスする表内の列およびそのアクセス方法を制限できますが、基礎となる表自体は制限できません。

たとえば、ユーザーが顧客番号を入力したときに「customer」という表から2つの列を取得するストアドプロシージャを作成する必要がある場合、DBAは次のようなプロシージャを作成できます。

```
delimiter //
create procedure p_cust (p1 int)
begin
            select c_name,c_address from customer
            where c_custkey = p1;
end
//
delimiter ;
```

ストアドプロシージャのコールは次のように実行します。

```
mysql> call p_cust(1);
+---------------------------+--------------+
| c_name                    | c_address    |
+---------------------------+--------------+
| Customer#000000001        | IVhzIApeRb   |
+---------------------------+--------------+
1 row in set (0.26 sec)
```

ストアドプロシージャおよびInfiniDBを使用する場合の現在の制限では、一度に使用できるオープンカーソルは1つだけです。

また、SQL関数も作成して使用できます。簡単な関数の例を次に示します。

```
mysql> CREATE FUNCTION hello (s CHAR(20))
    -> RETURNS CHAR(50) DETERMINISTIC
    -> RETURN CONCAT('Hello, ',s,'!');
Query OK, 0 rows affected (0.00 sec)

mysql> select hello('world');
+----------------+
| hello('world') |
+----------------+
| Hello, world! |
+---------------+
1 row in set (0.03 sec)
```

ストアドプロシージャおよび関数の詳細は、最新版のMySQLのリファレンスマニュアルを参照してください。

## 5.3 サポートされていないデータベースオブジェクト

現行バージョンのInfiniDBでは次のオブジェクトはサポートされていません。

- トリガー
- 主キー
- 外部キー
- NOT NULLとデフォルト値以外の列制約

## 5.4 InfiniDB システムカタログ

InfiniDBをインストールすると、InfiniDBオブジェクトに関連付けられているメタデータの追跡に使用される小さいシステムカタログが構成されます。このシステムカタログは、MySQLの`mysql`および`information_schema`データベース内にある標準オブジェクトとともに使用されます。データベース名またはスキーマ名は`calpontsys`です。

InfiniDBシステムカタログデータベース内に含まれているオブジェクトは次のとおりです。

```
mysql> desc systable;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| tablename     | varchar(128) | YES  |     | NULL    |       |
| schema        | varchar(128) | YES  |     | NULL    |       |
| objectid      | int(11)      | YES  |     | NULL    |       |
| createdate    | date         | YES  |     | NULL    |       |
| lastupdate    | date         | YES  |     | NULL    |       |
| init          | int(11)      | YES  |     | NULL    |       |
| next          | int(11)      | YES  |     | NULL    |       |
| numofrows     | int(11)      | YES  |     | NULL    |       |
| avgrowlen     | int(11)      | YES  |     | NULL    |       |
| numofblocks   | int(11)      | YES  |     | NULL    |       |
| autoincrement | int(11)      | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
11 rows in set (0.00 sec)


mysql> desc syscolumn;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| schema          | varchar(128) | YES  |     | NULL    |       |
| tablename       | varchar(128) | YES  |     | NULL    |       |
| columnname      | varchar(128) | YES  |     | NULL    |       |
| objectid        | int(11)      | YES  |     | NULL    |       |
| dictobjectid    | int(11)      | YES  |     | NULL    |       |
| listobjectid    | int(11)      | YES  |     | NULL    |       |
| treeobjectid    | int(11)      | YES  |     | NULL    |       |
| datatype        | int(11)      | YES  |     | NULL    |       |
| columnlength    | int(11)      | YES  |     | NULL    |       |
| columnposition  | int(11)      | YES  |     | NULL    |       |
| lastupdate      | date         | YES  |     | NULL    |       |
| defaultvalue    | varchar(64)  | YES  |     | NULL    |       |
| nullable        | int(11)      | YES  |     | NULL    |       |
| scale           | int(11)      | YES  |     | NULL    |       |
```

```
| prec            | int(11)      | YES  |     | NULL    |       |
| autoincrement   | char(1)      | YES  |     | NULL    |       |
| distcount       | int(11)      | YES  |     | NULL    |       |
| nullcount       | int(11)      | YES  |     | NULL    |       |
| minvalue        | varchar(64)  | YES  |     | NULL    |       |
| maxvalue        | varchar(64)  | YES  |     | NULL    |       |
| compressiontype | int(11)      | YES  |     | NULL    |       |
| nextvalue       | bigint(20)   | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
22 rows in set (0.00 sec)
```

# 6 データのロードおよび操作

この項では、InfiniDBデータベース内でのデータのロードおよび操作について説明します。

## 6.1 InfiniDB のバルクローダー

InfiniDBでは、InfiniDBの表にデータを迅速に効率よくインポートする高速のバルクロードユーティリティが提供されています。このユーティリティは、データのフィールド（表内の列など）がデリミタで区切られたデータを含む任意のフラットファイルを入力することができます。デフォルトのデリミタはパイプ（|）文字ですが、カンマなど、その他のデリミタも使用できます。

InfiniDBのバルクロードユーティリティ（`cpimport`と呼ばれる）は、InfiniDBデータベースへのデータのインポート時に次の操作を実行します。

- 指定されたフラットファイルからデータを読み取ります
- InfiniDBの列指向のストレージ設計に合うようにデータを変換します
- 冗長データをトークン化し、論理的に圧縮します
- データをディスクに書き込みます

`cpimport`を使用する最も一般的な方法には次の2つがあります。

1. UMから:`cpimport`がすべてのパフォーマンスモジュールに行を分散します
2. PMから:`cpimport`が起動元のPMにのみインポートされた行をロードします

これらほど一般的ではありませんが、`cpimport`には他にも多くのデータのロード方法があります。`cpimport`の使用方法の詳細については、『InfiniDB管理者ガイド』を参照してください。

`cpimport`ユーティリティの2つの主な使用手順は次のとおりです。

1. フラットファイルから複数の表にデータをロードするのに使用するジョブファイルを作成します（オプション）
2. `cpimport`ユーティリティを実行して、データのインポートを実行します

バルクロードは表への追加操作であるため、処理中も既存のデータの読取りが可能であり、既存のデータは影響を受けないことに注意してください。また、バルクロードでは、データ操作がトランザクションログに書き込まれません。これらの操作にトランザクションの特性はなく、この時点ではアトミック操作と見なされます。ただし、情報マーカーがトランザクションログに配置されるため、DBAはバルク操作が発生したことを確認できます。

ロード操作が完了すると、各列ファイルの最高水位標がアトミック操作で移動されます。このアトミック操作により、新たにロードされたデータを後続の問合せで読み取ることができるようになります。この追加操作では、読取り一貫性を実現できますが、データのロギングのオーバーヘッドは発生しません。

### 6.1.1 シンプルバルクロードの例

DBAが「db1」データベースの「t1」表をロードし、「t1.tbl」という名前のロードファイルを使用することを想定します。このファイルには次のデータが含まれ、パイプデリミタを使用してデータが区切られています。

```
1|Smith|2000-01-01
2|Jones|2000-02-01
3|Fields|2000-03-01
4|Johnson|2000-04-01
5|Simms|2000-05-01
```

InfiniDBの現在のバージョンでは、`-f`パス名オプションが使用されていない場合、このファイルの場所は、デフォルトで現在のディレクトリに設定されます。現在、ユーティリティでは`<table name>.tbl`のネーミング形式がデフォルトで設定されます。

表の構造は次のとおりです。

```
mysql> desc t1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| c1    | int(11)     | YES  |     | NULL    |       |
| c2    | varchar(20) | YES  |     | NULL    |       |
| c3    | date        | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.03 sec)
```

1つの表のみをロードする場合は、`cpimport`ユーティリティを直接コールできます。DBAは、`cpimport`ユーティリティ（InfiniDBの`bin`ディレクトリにある）を実行して、実際にデータを表にロードします。

```
[root@server1 test]# pwd
/home/test

[root@server1 bin]# /usr/local/Calpont/bin/cpimport db1 t1 t1.dbl

Bulkload root directory : /home/test
job description file : Job_501.xml
2009-10-06 15:08:16 (17995) INFO : successfully load job file /usr/
local/Calpont/data/bulk/job/Job_501.xml
2009-10-06 15:08:16 (17995) INFO : PreProcessing check starts
2009-10-06 15:08:16 (17995) INFO : PreProcessing check completed
2009-10-06 15:08:16 (17995) INFO : preProcess completed, total run time :
0 seconds
2009-10-06 15:08:16 (17995) INFO : No of Read Threads Spawned = 1
2009-10-06 15:08:16 (17995) INFO : No of Parse Threads Spawned = 3
2009-10-06 15:08:17 (17995) INFO : For table db1.t1: 5 rows processed and
5 rows inserted.
2009-10-06 15:08:17 (17995) INFO : Bulk load completed, total run time : 1
seconds
```

複数の表をロードする場合は、後述する複数の表のバルクロード例を参照してください。

#### 6.1.1.1 STDIN のバルクロードの例

InfiniDBでは、cpimportを簡単に使用した、標準入力からのデータのバルクロードがサポートされています。前述の同じロードファイルを使用して、ファイルをcpimportプロセスに送信できます。たとえば、最初の100,000,000行のみをインポートするようにt1.tblファイルを制限でき、このファイルをcpimportに送信します。

```
head -100000000 t1.tbl | /usr/local/Calpont/bin/cpimport db1 t1   -s '\t'
```

#### 6.1.1.2 表から選択した内容のバルク挿入

標準入力は、任意のSELECT文からの出力を高速のバルクローダーに直接送信するために使用することもできます。ここでは、データ以外の書式設定を削除するために-Nフラグを使用してdb2.source_tableから選択しています。

```
idbmysql -e 'select * from source_table;' -N db2 | /usr/local/
Calpont/bin/cpimport db1 t1 -s '\t'
```

### 6.1.2 複数の表のバルクロード

インポートユーティリティを使用して複数の表を同時にインポートできます。シンプルバルクロードでは、インポートごとに表が一意であるかぎり、複数のジョブを同時に発行できます。従来のバルクロードでは、スキーマ内のすべての表をインポートするか、またはcolxmlで-tオプションを使用して特定の表をリストすることで複数の表をインポートできます。表およびジョブIDが一意であるかぎり、従来のインタフェースを使用して複数のジョブを発行できます。

### 6.1.3 従来のバルクロードの例

DBAは、従来のバルクロードを使用して、スキーマ全体または複数の表を1つのバルクロードジョブでロードすることもできます。たとえば、DBAが、「tpch2」という名前のデータベースにある次の表にデータをロードするとします。

```
mysql> show tables;
+---------------+
| Tables_in_rms |
+---------------+
| customer      |
| lineitem      |
| nation        |
| orders        |
| part          |
| partsupp      |
| region        |
| supplier      |
+---------------+
8 rows in set (0.00 sec)
```

ロードファイルでデフォルトのパイプ (|) フィールドデリミタを使用し、インポートディレクトリにすべての表のロードファイルを配置してから、「tpch2」データベース自体に対するロードジョブ用にcolxmlユーティリティを実行できます。

```
[user1@server1 bin]$ ./colxml tpch2 -j500
Running colxml with the following parameters:
2009-10-07 15:14:20 (9481) INFO :
        Schema: tpch2
        Tables:
        Load Files:
```

```
        -b      0
        -c      1048576
        -d      |
        -e      10
        -f      CSV
        -j      500
        -n
        -p      /usr/local/Calpont/data/bulk/job/
        -r      5
        -s
        -u
        -w      10485760
        -x      tbl

    File completed for tables:
        tpch2.customer
        tpch2.lineitem
        tpch2.nation
        tpch2.orders
        tpch2.part
        tpch2.partsupp
        tpch2.region
        tpch2.supplier

     Normal exit.
```

その後、実際のバルクロード操作を実行できます。

```
[user1@server1 bin]$ ./cpimport -j 500

Bulkload root directory : /usr/local/Calpont/data/bulk
job description file : Job_500.xml
2009-10-07 15:14:59 (9952) INFO : successfully load job file /usr/
local/Calpont   data/bulk/job/Job_500.xml
2009-10-07 15:14:59 (9952) INFO : PreProcessing check starts
2009-10-07 15:15:04 (9952) INFO : PreProcessing check completed
2009-10-07 15:15:04 (9952) INFO : preProcess completed, total run time : 5
seconds
2009-10-07 15:15:04 (9952) INFO : No of Read Threads Spawned = 1
2009-10-07 15:15:04 (9952) INFO : No of Parse Threads Spawned = 3
2009-10-07 15:15:06 (9952) INFO : For table tpch2.customer: 150000 rows
processed and 150000 rows inserted.
2009-10-07 15:16:12 (9952) INFO : For table tpch2.nation: 25 rows processed
and 25 rows inserted.
2009-10-07 15:16:12 (9952) INFO : For table tpch2.lineitem: 6001215 rows
processed and 6001215 rows inserted.
2009-10-07 15:16:31 (9952) INFO : For table tpch2.orders: 1500000 rows
processed and 1500000 rows inserted.
2009-10-07 15:16:33 (9952) INFO : For table tpch2.part: 200000 rows processed
and 200000 rows inserted.
2009-10-07 15:16:44 (9952) INFO : For table tpch2.partsupp: 800000 rows
processed and 800000 rows inserted.
2009-10-07 15:16:44 (9952) INFO : For table tpch2.region: 5 rows processed
and 5 rows inserted.
2009-10-07 15:16:45 (9952) INFO : For table tpch2.supplier: 10000 rows
processed and 10000 rows inserted.
2009-10-07 15:16:45 (9952) INFO : Bulk load completed, total run time : 106
seconds
```

### 6.1.4 STDIN のバルクロードの例

InfiniDBでは、標準入力からのデータのバルクロードがサポートされています。この際、複数表のcpimportで-fSTDINフラグを指定します。前述の同じロードファイルを-j501コマンドで使用して、ファイルをcpimportプロセスに送信できます。たとえば、最初の100,000,000行のみをインポートするようにt1.tblファイルを制限でき、このファイルをcpimportに送信します。

```
head -100000000 t1.tbl | /usr/local/Calpont/bin/cpimport -j501 -s '\t' -fSTDIN
```

### 6.1.5 表から選択した内容のバルク挿入

標準入力は、任意のSELECT文からの出力を高速のバルクローダーに直接送信するために使用することもできます。ここでは、データ以外の書式設定を削除するために-Nフラグを使用してdb2.source_tableから選択しています。

```
idbmysql -e 'select * from source_table;' -N db2 | /usr/local/
Calpont/bin/cpimport -j501 -s '\t' -fSTDIN
```

### 6.1.6 バイナリソースのバルク挿入

固定長のレコードを含むバイナリファイルは、cpimportへの入力としても使用できます。これは、-I1（NULLの許容）または-I2（NULLの飽和演算）オプションを使用して実行できます。NULLを含むデータをロードする場合の、バイナリファイルのロードによるシンプルなインポート方法の例を次に示します。

```
[root@server1 bin]# /usr/local/Calpont/bin/cpimport -I1 mydb mytable mytablesource.bin
```

## 6.2 データロード用の MySQL ユーティリティ

MySQLでは、データベースの表にデータをロードするload data infileコマンドが提供されています。このコマンド（ユーティリティ）はInfiniDBでサポートされており、使用可能ですが、InfiniDBのバルクロードユーティリティcpimportを使用したほうが高速です。

次の表を例とします。

```
mysql> desc t3;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| c1    | int(11) | YES  |     | NULL    |       |
| c2    | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
```

次のファイルをロードします。

```
[root@server1 local]# more t3.tst
1,2
2,3
3,4
```

次のように実行してロードできます。

```
mysql> load data infile '/usr/local/t3.tst' into table t3 fields terminated
by ',';
Query OK, 3 rows affected (0.96 sec)
```

## 6.3 DML のサポート

InfiniDBでは標準DMLがサポートされているため、ユーザーは、InfiniDBの表に対してSQLのINSERT、UPDATEおよびDELETE文を発行できます。InfiniDBは列指向データベースであるため、InfiniDBでINSERT文およびDELETE文を実行した場合、行指向データベースで実行した場合よりも、通常は低速になることに注意してください（より多くのブロックにアクセスする必要があります）。ただし、UPDATE文はかなり高速に実行されます。

INSERT文の場合、InfiniDBではMySQLの複数挿入操作がサポートされることに注意してください。次の表を例とします。

```
mysql> desc t3;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| c1    | int(11) | YES  |     | NULL    |       |
| c2    | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
```

ユーザーは次のINSERT文を発行できます。

```
mysql> insert into t3 values (1,2),(3,4),(5,6);
Query OK, 3 rows affected (0.34 sec)
Records: 3  Duplicates:0  Warnings:0
```

DML構文の詳細は、『InfiniDB SQL構文ガイド』を参照してください。

## 6.4 トランザクション管理

InfiniDBでは、完全なコミット、ロールバックおよびデッドロック検出機能とともに、ACID準拠のトランザクションがサポートされています。InfiniDBのデフォルトモードは、すべてのDML文が自動的に確定される自動コミットです。この設定は、InfiniDBのセッションごとに簡単に変更できます。たとえば、次の例では自動コミットモードをオフに設定し、表を更新して変更をロールバックし、再度表を更新して、データベースに対しトランザクションをコミットします。

```
mysql> set autocommit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select c1 from t3;
+------+
| c1   |
+------+
|    1 |
|    2 |
|    3 |
|    1 |
|    1 |
|    3 |
|    5 |
```

```
+------+
7 rows in set (0.26 sec)

mysql> update t3 set c1=4;
Query OK, 7 rows affected (0.26 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> rollback;
Query OK, 0 rows affected (0.23 sec)

mysql> select c1 from t3;
+------+
| c1   |
+------+
|    1 |
|    2 |
|    3 |
|    1 |
|    1 |
|    3 |
|    5 |
+------+
7 rows in set (0.26 sec)

mysql> update t3 set c1=5;
Query OK, 7 rows affected (0.14 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> commit;
Query OK, 0 rows affected (0.21 sec)

mysql> select c1 from t3;
+------+
| c1   |
+------+
|    5 |
|    5 |
|    5 |
|    5 |
|    5 |
|    5 |
|    5 |
+------+
7 rows in set (0.14 sec)
```

## 6.5 同時実行の制御

InfiniDBでは、ページレベルのロックが使用され、分離レベルでのコミット読取りロックが採用されており、コミットされたデータのみが読み取られるようにしています。

InfiniDBでは、InfiniDBにバージョニングインフラストラクチャを提供する多版型同時実行制御（MVCC）も使用されます。これは、他のデータベースでは「スナップショット読取り」と呼ばれることもあります。スナップショット読取りは、問合せが常に、問合せが発行された時点のデータを参照することを意味します。これによって、問合せに関係するすべての表のすべての列で読取り一貫性が実現されます。

InfiniDBのバージョニングの最も重要なメリットは、*読取りが遮断されない*ことです。つまり、バージョニングされた（古い）ブロックを、あるセッションが更新しているときに、同時に別のセッションが読み取ることができます。InfiniDBのすべての文は、InfiniDBによってシステム変更番号（SCN）として参照されるデー

タベースの特定のバージョン（またはスナップショット）で実行されます。循環バージョンバッファ、バージョン置換構造（VSS）およびバージョンバッファブロックマネージャ（VBBM）構造を使用して、リクエストされたブロックの適切なバージョンが検索されます。これらのオブジェクトのサイズは、予測負荷でのバッファオーバーランを回避できるように構成可能です。

バージョニングがトランザクションのDML文のみに適用され、追加操作を使用して行われるバルクロードには適用されないことに注意してください。

# 7 アプリケーションおよびビジネスインテリジェンスの開発

この項では、アプリケーション開発およびビジネスインテリジェンス環境でのInfiniDBとの連携動作について簡単に説明します。

## 7.1 アプリケーションツールおよび BI ツールを使用した InfiniDB への接続

InfiniDBではMySQLフロントエンドが使用されるため、InfiniDBへの接続には任意の標準MySQLコネクタを使用できます。無償で入手できるコネクタには、次のものがあります。

- ADO.NET
- ODBC
- JDBC
- C++
- C
- PHP
- Perl
- Python
- Ruby

MySQLコネクタは、MySQL Webサイトから自由にダウンロードして使用できます。現在のURLの場所は、http://www.mysql.com/products/connector/です。

## 7.2 アプリケーション開発ツールおよびビジネスインテリジェンスツール

InfiniDBのMySQLフロントエンドにより、InfiniDBでは、現在MySQLをサポートするアプリケーション開発ツールまたはビジネスインテリジェンスツールが動作します。Eclipse、Microsoft Visual Studioなど、多くのアプリケーション開発ツールがMySQLをサポートしており、InfiniDBで正常に動作します。

現在、MySQLに接続するビジネスインテリジェンスツールには、Business Objects、Cognos、Microsoft Analysis Servicesなど、主要ベンダー独自仕様のものがすべて含まれます。Pentaho、Jaspersoft、TalendなどのオープンソースのBIツールもInfiniDBで動作します。

# 8 パフォーマンスのチューニング

この項では、InfiniDBデータベースの監視およびチューニングについて説明します。

## 8.1 監視および診断ユーティリティ

cmconsoleユーティリティを使用すると、DBAまたはsysadminは、単一または複数ノードのInfiniDB構成で様々な監視操作を実行できます。DBAは、cmconsoleユーティリティを使用して、ユーザーモジュールおよびパフォーマンスモジュールが動作するデータベースとサーバーの両方を監視できます。

各種の監視および診断コマンドのリストを次に示します。これらのコマンドの多くは、その名前から内容がわかるようになっています。

| コマンド | 説明 |
|---|---|
| ? | コンソールコマンドに関するヘルプを表示します |
| addDbroot | DBRoot ディスクストレージを Calpont InfiniDB システムに追加します |
| addExternalDevice | 外部デバイスを構成ファイルに追加します |
| addModule | Calpont InfiniDB システム内でモジュールを追加します |
| alterSystem-disableModule | モジュールを無効にして Calpont InfiniDB システムを変更します |
| alterSystem-enableModule | モジュールを有効にして Calpont InfiniDB システムを変更します |
| assignDbrootPmConfig | 割り当てられていない DBroot をパフォーマンスモジュールに割り当てます |
| assignElasticIPAddress | Amazon Elastic IP アドレスをモジュールに割り当てます |
| disableLog | プロセスおよびデバッグのロギングのレベルを無効にします |
| enableLog | プロセスおよびデバッグのロギングのレベルを有効にします |
| exit | コンソールツールを終了します |
| getActiveAlarms | アクティブなアラームリストを取得します |
| getActiveSQLStatements | システム内でアクティブな SQL 文のリストを取得します |
| getAlarmConfig | アラームの構成情報を取得します |
| getAlarmHistory | システムアラームを取得します |
| getAlarmSummary | アクティブなアラームの集計数を取得します |
| getCalpontSoftwareInfo | Calpont InfiniDB RPM の詳細情報を取得します |
| getExternalDeviceConfig | 外部デバイスの構成情報を取得します |
| getLogConfig | システムログファイルの構成を取得します |
| getModuleConfig | モジュール名の構成情報を取得します |
| getModuleCpu | モジュールの CPU 使用率を取得します |
| getModuleCpuUsers | CPU を利用している上位のモジュールプロセスを取得します |
| getModuleDisk | モジュールのディスク使用率を取得します |
| getModuleMemory | モジュールのメモリー使用率を取得します |
| getModuleMemoryUsers | メモリーを利用している上位のモジュールプロセスを取得します |
| getModuleResourceUsage | モジュールのリソース使用率を取得します |
| getModuleTypeConfig | モジュールタイプの構成情報を取得します |
| getProcessConfig | プロセスの構成情報を取得します |
| getProcessStatus | Calpont InfiniDB プロセスの状態を取得します |
| getStorageConfig | システムのストレージ構成情報を取得します |
| getStorageStatus | システムのストレージステータスを取得します |
| getSystemConfig | システムの構成情報を取得します |
| getSystemCpu | すべてのモジュールにおけるシステムの CPU 使用率を取得します |
| getSystemCpuUsers | CPU を利用している上位のシステムプロセスを取得します |
| getSystemDisk | すべてのモジュールにおけるシステムのディスク使用率を取得します |
| getSystemInfo | システム全体の状態を取得します |

| | |
|---|---|
| getSystemMemory | すべてのモジュールにおけるシステムのメモリー使用率を取得します |
| getSystemMemoryUsers | メモリーを利用している上位のシステムプロセスを取得します |
| getSystemNetworkConfig | システムのネットワーク構成情報を取得します |
| getSystemResourceUsage | すべてのモジュールにおけるシステムのリソース使用率を取得します |
| getSystemStatus | システムおよびモジュールの状態を取得します |
| help | コンソールコマンドに関するヘルプを表示します |
| monitorAlarms | リアルタイムモードでアラームを監視します |
| movePmDbrootConfig | DBRoot をパフォーマンスモジュールから別のモジュールに移動します |
| quit | コンソールツールを終了します |
| removeDbroot | DBRoot ディスクストレージを Calpont InfiniDB システムから削除します |
| removeExternalDevice | 外部デバイスを構成ファイルから削除します |
| removeModule | モジュールを Calpont InfiniDB システムから削除します |
| resetAlarm | アクティブなアラームをリセットします |
| restartSystem | 停止またはシャットダウンされた Calpont InfiniDB システムを再起動します |
| resumeDatabaseWrites | Calpont InfiniDB データベースへの書込みの実行を再開します |
| setAlarmConfig | アラームの構成パラメータを設定します |
| setExternalDeviceConfig | 外部デバイスの構成パラメータを設定します |
| setModuleTypeConfig | モジュールタイプの構成パラメータを設定します |
| setProcessConfig | プロセスの構成パラメータを設定します |
| setSystemConfig | システムの構成パラメータを設定します |
| shutdownSystem | Calpont InfiniDB システムをシャットダウンします |
| startSystem | 停止またはシャットダウンされた Calpont InfiniDB システムを起動します |
| stopSystem | Calpont InfiniDB システムの処理を停止します |
| suspendDatabaseWrites | Calpont InfiniDB データベースへの書込みの実行を一時停止します |
| switchParentOAMModule | アクティブな親 OAM モジュールを別のパフォーマンスモジュールに切り替えます |
| system | システムシェルコマンドを実行します |
| unassignElasticIPAddress | Amazon Elastic IP アドレスの割り当てを解除します |

コマンドのヘルプを参照するには、「**help**」に続けてコマンド名を入力します

cmconsoleを使用して、多数のシステム診断を監視および分析できます。問合せアクティビティ、システム全体の状態、ディスク使用率、CPUとメモリーのリソース消費などの問合せおよびレポートが可能です。たとえば、DBAまたはsysadminは、次のコマンドを実行して、構成全体の状態をすばやく取得できます。

```
calpont> getSystemStatus
getsystemstatus   Wed Oct 26 12:34:28 2012

System qperfd01

System and Module statuses

Component     Status              Last Status Change
------------  ------------------  ------------------------
System        ACTIVE              Wed Oct  7 15:10:47 2009
Module um1    ACTIVE              Tue Oct  6 15:50:09 2009
Module pm1    ACTIVE              Tue Oct  6 15:50:19 2009
Module pm2    ACTIVE              Wed Oct  7 15:10:45 2009
Module pm3    MAN_DISABLED        Tue Oct  6 13:25:09 2009
Module pm4    MAN_DISABLED        Tue Oct  6 13:25:39 2009
Module pm5    MAN_DISABLED        Tue Oct  6 11:19:38 2009
Module pm6    MAN_DISABLED        Tue Oct  6 11:20:10 2009
Module pm7    MAN_DISABLED        Tue Oct  6 11:20:44 2009
Module pm8    MAN_DISABLED        Tue Oct  6 11:21:18 2009
```

```
  Active Parent OAM Performance Module is 'pm1'
```

この出力は、現在、1つのユーザーモジュールおよび2つのパフォーマンスモジュールがオンラインのため、問合せに利用可能であるのに対し、パフォーマンスモジュール3-8が現在オフラインであることを示しています。DBAまたはsysadminは、構成内にあるいずれかのノードに関するCPU、ディスクおよびメモリーの消費を表示する場合、次のようなコマンドを入力できます。

```
calpont> getModuleCpu pm1
getmodulecpu   Wed Oct 26 12:35:36 2012

Module 'pm1' CPU Usage % = 47

calpont> getModuleDisk pm1
getmoduledisk   Wed Oct 26 12:36:02 2012
Module 'pm1' Disk Usage (in 1K blocks)

Mount Point               Total Blocks  Used Blocks  Usage %
------------------------  ------------  ------------ -------
/                         66719320      57662764         87
/boot                     101086        19987            20
/usr/local/Calpont/data1  1124868224    830641000        74
/usr/local/Calpont/data2  1124868224    835241732        75
/usr/local/Calpont/data3  1124868224    834809896        75
/usr/local/Calpont/data4  1124868224    833940516        75
/usr/local/Calpont/data5  1124868224    835826024        75
/usr/local/Calpont/data6  1124868224    833620112        75
/usr/local/Calpont/data7  1124868224    832053424        74
/usr/local/Calpont/data8  1124868224    832324408        74
/home/qa/srv              235332176     88043896         38

calpont> getModuleMemory pm1
getmodulememory   Wed Oct 26 12:37:20 2012

Module Memory Usage (in K bytes)

Module  Mem Total  Mem Used cache   Mem Usage % Swap Total Swap Used Swap
                                                                Usage%
------  ---------  -------  ------- ----------  ---------- --------- ----
pm1     16440436   15472352  136236      94       2031608     0        0
```

cmconsoleを使用してローカルマシンおよびリモートマシンを監視することで時間を節約することができます。さらに、これによって、DBAおよびsysadminは、1つのツールでデータベースと物理サーバーの両方の診断を実行できます。

## 8.2 アラームおよび通知

InfiniDBでは、DBAまたはsysadminは、データベースまたはシステムで障害が発生する前に問題の通知を受けるために使用可能な各種のシステムアラームを構成することもできます。管理者は、ディスク、CPU、メモリー、システムのスワップ領域、非互換のソフトウェアバージョンなどに対してしきい値を設定できます。

アラームおよび通知は、cmconsoleユーティリティを使用して管理します。管理者は、次のコマンドを入力すると、構成可能なアラームのリストを表示できます。

```
  calpont> getAlarmConfig
  getalarmconfig   Wed Oct  7 15:45:50 2009
```

```
Alarm Configuration

Alarm ID #1 Configuration information
BriefDesc = CPU_USAGE_HIGH
DetailedDesc = The usage on the indicated CPU has exceeded its high threshold
Severity = CRITICAL
Threshold = 100

Alarm ID #2 Configuration information
BriefDesc = CPU_USAGE_MED
DetailedDesc = The usage on the indicated CPU has exceeded its medium threshold
Severity = MAJOR
Threshold = 70

Alarm ID #3 Configuration information
BriefDesc = CPU_USAGE_LOW
DetailedDesc = The usage on the indicated CPU has exceeded its low threshold
Severity = MINOR
Threshold = 50

Alarm ID #4 Configuration information
BriefDesc = DISK_USAGE_HIGH
DetailedDesc = The usage on the indicated Disk Drive has exceeded its high
threshold
Severity = CRITICAL
Threshold = 100
.
.
.
```

管理者は、setAlarmConfigコマンドを使用して、アラーム番号とそのしきい値を指定することによって、アラームの現在の設定を変更できます。たとえば、約95%に達したCPUに関して通知するCPU_USAGE_HIGHアラームを設定する場合、管理者は次のコマンドを実行します。

```
calpont> setAlarmConfig 1 Threshold 95
setalarmconfig   Wed Oct  7 15:55:47 2009

   Successfully set Threshold = 95
```

トリガーされたアラームは、getActiveAlarmsコマンドを使用して表示できます。

```
calpont> getActiveAlarms

AlarmID           = 1
Brief Description = CPU_USAGE_HIGH
Alarm Severity    = CRITICAL
Time Issued       = Tue Oct  6 14:50:06 2009
Reporting Module  = pm2
Reporting Process = ServerMonitor
Reported Device   = CPU

AlarmID           = 30
Brief Description = CONN_FAILURE
Alarm Severity    = CRITICAL
Time Issued       = Tue Oct  6 15:48:08 2009
Reporting Module  = um1
Reporting Process = ExeMgr
Reported Device   = 10.100.7.5 PrimProc
```

Nagios、HP OpenViewまたはTivoliなどのグローバルシステム監視ソフトウェアにSNMPトラップを送信するようにアラーム通知を構成することもできます。

# 9 SQL の監視および診断ユーティリティ

適切に記述されていないSQLは、多くの場合、データベースでパフォーマンスの問題が発生する1番の原因となります。適切に実行されていないSQL文を特定して修正できるように、InfiniDBでは多数のコマンドおよびユーティリティが提供されています。

まず、一般的なMySQLサポートの観点では、ユーザーは、ログオンしているユーザーとそのユーザーが現在実行しているSQLに関する簡単な情報を`show processlist`コマンドを使用して表示できます。

```
mysql> show processlist;
+----+------+-----------+-------+---------+------+-------+--------------+
| Id | User | Host      | db    | Command | Time | State | Info         |
+----+------+-----------+-------+---------+------+-------+--------------+
| 73 | root | localhost | ssb10 | Query   |    0 | NULL  | show processlist
+----+------+-----------+-------+---------+------+-------+--------------+
1 row in set (0.01 sec)
```

InfiniDBの`cmconsole`ユーティリティでも、データベースで現在実行されているSQLを表示する`getActiveSQLStatements`コマンドが提供されています。

```
calpont> getActiveSQLStatements
getactivesqlstatements   Wed Oct  7 08:38:32 2009

Get List of Active SQL Statements
=================================

Start Time        Time (hh:mm:ss)   Session ID             SQL Statement
---------------   ---------------   --------------------
-------------------------------------------------------------
Oct  7 08:38:30    00:00:03     73               select c_name, sum(lo_revenue)
from customer, lineorder where lo_custkey = c_custkey and c_custkey = 6 group by
c_name
```

個々のSQL文のパフォーマンスに対してトラブルシューティングを行う場合、使用可能なユーティリティは3つあります。1つ目の標準MySQLの`EXPLAIN`ユーティリティは、InfiniDBでは索引やMySQL I/O機能が使用されないため、若干有用性に欠けます。ただし、次の例に示すように、`EXPLAIN`でいくつかの情報を取得できます。

```
mysql> explain select select count(*) from dateinfo,customer, part, supplier, lineorder
where s_suppkey = lo_suppkey and d_datekey = lo_or and p_partkey = lo_partkey and
c_custkey = lo_custkey and s_nation = 'BRAZIL';

+----+-------------+-----------+------+---------------+------+---------+-----
-+------+-------------------------------------------------------+
| id | select_type | table     | type | possible_keys | key  | key_len | ref  |
rows | Extra                                                    |
+----+-------------+-----------+------+---------------+------+---------+-----
-+------+-------------------------------------------------------+
|  1 | SIMPLE      | dateinfo  | ALL  | NULL          | NULL | NULL    | NULL |
2000 |                                                          |
|  1 | SIMPLE      | customer  | ALL  | NULL          | NULL | NULL    | NULL |
2000 | Using join buffer                                        |
|  1 | SIMPLE      | part      | ALL  | NULL          | NULL | NULL    | NULL |
2000 | Using join buffer                                        |
|  1 | SIMPLE      | supplier  | ALL  | NULL          | NULL | NULL    | NULL |
2000 | Using where with pushed condition; Using join buffer |
```

```
|  1 | SIMPLE      | lineorder | ALL  | NULL          | NULL | NULL    | NULL |
2000 | Using where; Using join buffer                       |
+----+-------------+-----------+------+---------------+------+---------+-----
-+------+-----------------------------------------------------+
```

InfiniDBが提供するSQL統計コマンド(calgetstats)およびトレースユーティリティ(caltrace)を使用すると、SQL文のパフォーマンスに関して、より高度な診断情報を取得できます。

DBAまたは開発者は、calgetstatsをコールするだけで、SQLインタフェースを介して実行された問合せのSQLパフォーマンスの統計を表示できます。次に例を示します。

```
mysql> select c_name, sum(l_quantity) from customer, orders, lineitem where
c_custkey = o_custkey and l_orderkey = o_orderkey and l_shipdate = '1992-01-02'
group by c_name;
+--------------------+-----------------+
| c_name             | sum(l_quantity) |
+--------------------+-----------------+
| Customer#000094199 |           35.00 |
| Customer#000009263 |           19.00 |
.
.
17 rows in set (1.21 sec)

mysql> select calgetstats()\G
*************************** 1. row ***************************
calgetstats(): Query Stats: MaxMemPct-0; NumTempFiles-0; TempFileSpace-0MB;
ApproxPhyI/O-0; CacheI/O-7977; BlocksTouched-7977; PartitionBlocksEliminated-0;
MsgBytesIn-5MB; MsgBytesOut-0MB; Mode-Distributed| 1255540221 457489
1 row in set (0.00 sec)
```

統計出力には、次の情報が含まれます。

- MaxMemPct:大規模なハッシュ結合操作のサポート時など、ユーザーモジュールのピーク時のメモリー利用率。
- NumTempFiles:一時ファイルに関するレポート。一時ファイルは、利用可能なメモリーを超える大きさの問合せ操作(通常は、比較的小さい表結合カーディナリティが、構成可能な何らかのしきい値を超える特殊な結合操作)をサポートするために作成されます。
- TempFileSpace:一時ファイルが使用する領域に関するレポート。一時ファイルは、利用可能なメモリーを超える大きさの問合せ操作(通常は、比較的小さい表結合カーディナリティが、構成可能な何らかのしきい値を超える特殊な結合操作)をサポートするために作成されます。
- PhyI/O:ディスク、SSD、その他の永続ストレージから読み取られる8Kブロックの数。ほとんどの場合、個々のI/O操作の数はブロックの数に比べて非常に少ないため、InfiniDBでは1回のI/O操作で512個のブロックが同時に読み取られます。
- CacheI/O:必要な個別の物理I/Oコールの数により下方調整されている、メモリー内で処理された8Kブロックの概数。
- BlocksTouched:メモリー内で処理された8Kブロックの概数。
- PartitionBlocksEliminated:InfiniDBがエクステントマップによって除外したブロックアクセス数。
- MsgBytesIn、MsgByteOut:問合せをサポートするためにノード間で送信されるメッセージのバイト数(MB単位)。

InfiniDBのSQLトレースユーティリティ`caltrace`の基本的な使用手順は次のとおりです。

1. 特定のセッションのトレースを開始します
2. 問題のSQL文を実行します
3. この文に対して収集されたトレースを確認します

例として、次のセッションでトレースを開始して、6,000,000行のファクト表および300,000行のディメンション表に対して問合せを発行し、トレースからの出力を確認します。

```
mysql> select calsettrace(1);
+----------------+
| calsettrace(1) |
+----------------+
|              1 |
+----------------+
1 row in set (0.04 sec)

mysql> select c_name, sum(lo_revenue)
    -> from customer, lineorder
    -> where lo_custkey = c_custkey and
    -> c_custkey = 5
    -> group by c_name;
+--------------------+-----------------+
| c_name             | sum(lo_revenue) |
+--------------------+-----------------+
| Customer#000000005 |    552483078.00 |
+--------------------+-----------------+
1 row in set, 0 warning (1.23 sec)

mysql> select calgettrace();
+-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------------+
| calgettrace()
|
+-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------------+
|
Desc Mode Table     TableOID ReferencedOIDs PIO  LIO   PBE Elapsed Rows
BPS  PM   customer  3327     (3328,3336)    0    150   0   0.017   1
BPS  PM   lineorder 3278     (3281,3291)    4096 29808 0   0.766   63
HJS  PM   lineorder 3278     -              -    -     -   0.000   -
ADS  UM   -         -        -              -    -     -   0.724   -
 |
+-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------------+
1 row in set (0.02 sec)
```

各ステップの物理および論理I/O、経過時間などに関する情報が表示されます。

# 10 InfiniDB への移行

この項では、他のデータベースからInfiniDBデータベースに移行する方法について説明します。

## 10.1 一般的な移行の概念

通常、InfiniDBへの移行では、別のデータベースからのスキーマまたはデータの移動、および場合によってはストアドプロシージャなどのストアドコードオブジェクトの移動を実行します。通常、次のロードマップに従います。

1. ソースデータベースオブジェクトのドキュメント化
2. InfiniDBオブジェクトの設計
3. ソースシステムからInfiniDBへのデータの移動
4. InfiniDBへのコードオブジェクトの変換（オプション）

既存のデータベース構造のドキュメント化は、通常は手動のデータディクショナリ検索などを実行する標準の手動処理で試行すると困難な場合があります。ほとんどのデータベースには操作に役立つメタデータディクショナリが含まれていますが、手動ですべてのメタデータ（表、列など）を取得する処理は、大規模なデータベースでは非常に時間がかかるイベントとなる場合があります。

ほとんどの場合の最良の方法は、変換に必要で適用可能なすべてのメタデータを自動的にカタログ化する、コンピュータによるリバースエンジニアリング処理を使用する方法です。これには、PowerDesigner、ERWinおよびER/Studioなどの適切なサードパーティのデータモデリングツールを使用できます。これらのすべてのツールで複数のデータベースタイプのリバースエンジニアリングがサポートされています。

InfiniDBではほとんどの重要なデータ型がサポートされているため、通常、InfiniDBへのスキーマ構造の移動は困難なタスクではありません。InfiniDBでのデータ型のサポートの詳細は、『InfiniDB SQL構文ガイド』を参照してください。

ソースデータベースのメタデータを取得して理解したら、次の手順はInfiniDBターゲットデータベースの設計です。これには、ソースオブジェクトおよびそのプロパティ（列のデータ型など）をInfiniDBの対応するものに変換することが含まれます。一部のモデリングツールでは、ユーザーは基礎となるデータベースを物理的な設計で変更するオプションを選択することで、あるデータベーススキーマを別のデータベーススキーマに自動的に変換できます。すべてのデータ型が自動的に変更されますが、データ型が1対1で一致しない場合、一部のデータ型に手動での調整が必要なことがあります。

ストアドプロシージャなどのコードオブジェクトの変換は別問題です。Oracle、Sybase、SQL Serverなどのストアドプロシージャから MySQLのストアドプロシージャに変換する、IspirerのSQLWays製品などのいくつかのサードパーティツールがあります。

スキーマオブジェクトの変換の場合は、可能性がある1つの方法として、無償のMySQL Migration Toolkitの使用があります。MySQL Migration Toolkitを使用すると、ツールによってリバースエンジニアリングされたソースデータベースがMySQLベースの対応するものに自動的に変換され、すべてのデータ型変換が行われます。ユーザーは、ソースデータベースから移行するオブジェクトおよびMySQLの特定の指定方法に関して包括的に制御できます。

InfiniDB用の無償のMySQL Migration Toolkitを使用する場合、通常、ユーザーは次の手順に従います。

1. Migration Toolkitをダウンロードしてインストールします。MySQLのWebサイトのGUIツールのダウンロードセクション（現在、http://dev.mysql.com/downloads/gui-tools/5.0.html）にあります。
2. ツールを使用してソースデータベースに接続します。
3. ツールを使用してInfiniDBターゲットデータベースに接続します。
4. InfiniDBに移行するスキーマおよびオブジェクトを選択します。
5. オブジェクトマッピングの画面で、ユーザー定義のラジオボタンを選択して、作成するSQLスクリプトに「**engine=infinidb**」と入力します。
6. オブジェクトをオンラインで自動的に作成するのではなく、SQLの作成をスクリプトに送信することを選択します。
7. エディタで作成スクリプトを確認し、InfiniDBで必要なデータ型またはその他の変更を行います。
8. InfiniDB SQLエディタでスクリプトを実行してInfiniDB表オブジェクトを作成します。

## 10.2 MySQL のデータの移行方法

汎用的なMySQLデータベースのデータをInfiniDBに移動するには、いくつかの異なる方法を使用できます。1つ目では、ユーザーはMySQLの`SELECT INTO OUTFILE`コマンドを使用して一連の表出力をフラットファイルにスプールできます。たとえば、ユーザーは、2つの列がある「t1」という表から、パイプ（|）文字で区切られたファイルにデータを書き出す必要がある場合、次のように行うことができます。

```
mysql> select c1, c2
    -> from t1
    -> into outfile 't1.tbl' fields terminated by '|';
```

MySQLファイルへの書込みが行われたら、そのファイルをInfiniDBインポートディレクトリに移動し、`cpimport`ユーティリティを使用してInfiniDBにインポートできます。

もう1つの方法では、無償のオープンソースETL（抽出-変換-ロード）ツールを使用して、あるMySQLサーバーから、あるInfiniDBサーバーにデータを移動します。ツールはPentaho、Jaspersoft、Talendなどから入手できます。

## 10.3 Oracle のデータの移行方法

OracleのデータをInfiniDBに移動する方法は多くあります。MySQLに接続可能なすべてのデータ移動ツールがInfiniDBへのデータの移動に適しています。たとえば、DBAは、必要に応じてデータを移動し、変換を実行するのに、TalendまたはPentahoのKettleなどのオープンソースETLツールを自由にダウンロードして使用できます。

Oracleではデータベース用のアンロードツールは提供されていないため、DBAはOracleデータをアンロードするためのサードパーティのアンロードツールを所有していないかぎり、SQL*Plusセッションからフラットファイルをスプールアウトする必要がある場合があります。たとえば、DBAは、3つの列がある「workorder」という表を含むInfiniDBにロードするためにOracleからフラット表を作成し、パイプ（|）文字でデータを区切る場合、次のようなスクリプトを使用できます。

```
SET TERMOUT OFF
set newpage 0
set space 0
set linesize 500
set pagesize 0
set echo off
set feedback off
set heading off
```

```
set trimspool on
col wonum     format a10
col leadcraft format a8
col location  format a50
spool c:\workorder.txt
SELECT
WORKNUM || '|' ||
CONTACT || '|' ||
PRIMARY_LOCATION
FROM SCHEMA1.WORKORDER;
spool off
exit
```

## 10.4 Microsoft SQL Server のデータの移行方法

Microsoftでは、SQL Server製品との統合サービスが提供されています。これは、バンドルされたデータの移動およびBIのプラットフォームです。SQL ServerのDBAは、これを使用してInfiniDBに接続し、InfiniDBにデータを移動できます。

Oracleとは異なり、SQL Serverにはデータベース用の`bcp`という高速アンロードツールがあります。`bcp`ユーティリティを使用してSQL Server表データのフラットファイルを作成し、`cpimport`ユーティリティを使用してInfiniDBにロードできます。大量のデータに対しては、SQL Serverデータを`bcp`で出力し、`cpimport`でInfiniDBにインポートする方法が最良の方法であると考えられます。