

# Oracle TimesTen 製品とテクノロジー

オラクル・ホワイト・ペーパー

2007 年 2 月

# Oracle TimesTen 製品とテクノロジー

概要 .....	3
ミリ秒単位の重要性.....	3
リアルタイムなアプリケーションの成長.....	4
リアルタイムな業界.....	4
リアルタイムなエンタープライズ.....	4
リアルタイムなデータ管理ソフトウェア .....	4
アプリケーション層の配置.....	5
製品情報 .....	5
Oracle TimesTen In-Memory Database .....	5
Replication – TimesTen to TimesTen.....	6
Cache Connect to Oracle.....	6
インメモリー・データベース・テクノロジー.....	6
Oracle TimesTenの物理的な構造 .....	7
アプリケーション層の共有ライブラリ .....	7
メモリー常駐のデータ構造.....	8
システム・プロセス.....	9
管理プログラム.....	9
チェックポイントとログ・ファイル.....	9
データ・レプリケーションのテクノロジー.....	9
キャッシングのテクノロジー.....	10
IMDBテクノロジーの詳細 .....	11
クエリーの最適化.....	12
バッファ・プールの管理.....	12
索引構造.....	13
差異の理由.....	14
コストパフォーマンスにおける新しい様式.....	14
卓越したパフォーマンス .....	15
スケーラビリティ.....	15
レスポンス時間.....	16
リアルタイム機能 .....	17
データ管理.....	17
問合せ処理.....	20
データ・レプリケーション .....	22
キャッシング.....	24
イベント処理.....	27
結論 .....	28

## 概要

Oracle TimesTen In-Memory Database は、パフォーマンスが重要なシステムのために迅速なレスポンスと高いスループットを実現する、メモリーに最適化されたリレーショナル・データベースです。このデータベースは、アプリケーション層で、アプリケーションの近く、またはアプリケーションと強調して動作することを目的としています。また、レコードのデータベースや Oracle Database へのキャッシュとしても使用できます。

このホワイト・ペーパーでは、Oracle TimesTen 製品とテクノロジーならびにオラクル社の他の製品との統合ポイントを説明します。そのソフトウェアとマニュアルを使用に先立ち、紙面で評価するうえで役立ちます。

## ミリ秒単位の重要性

Oracle TimesTen 製品は、パフォーマンスが重要なシステムに対してアプリケーション層のデータ管理を提供し、迅速なレスポンスと Oracle データのリアルタイムのキャッシングを実現します。

Oracle TimesTen 製品は、パフォーマンスが重要なシステムに対してアプリケーション層のデータ管理を提供し、迅速なレスポンスと Oracle データのリアルタイムのキャッシングを実現します。企業は、Oracle TimesTen を使用してソフトウェアのインフラストラクチャを拡張し、次のようなシステムを作成できます。

- 即座にレスポンス可能
- 高拡張性
- 継続して使用可能

このシステムを使用して、次のことができます。

- 顧客のロイヤリティの向上
- 新しい顧客の取り込み
- 処理の合理化
- コスト高の独自ソフトウェア開発の回避

Oracle TimesTen は、1998 年から、ネットワーク、電話サービス、オペレーショナル・サポート・システム、コンタクト・センター、航空会社と予約システム、コマンド・アンド・コントロール・システム、証券取引などのスピードが重視される業界、ならびにリアルタイムなエンタープライズでの本稼動システムでの採用実績があります。Amdocs、Aspect、Avaya、Bombay Stock Exchange、Cisco、Ericsson、JP Morgan、Lucent、NEC、Nokia、Salesforce.com、Sprint など、何百社もの世界規模の企業が本番用のアプリケーションに Oracle TimesTen を使用しています。

## リアルタイムなアプリケーションの成長

### リアルタイムな業界

ネットワーク機器のメーカー、電話のオペレータ、証券取引所と証券会社、航空会社、物流管理企業、国防情報局などが、リアルタイムなアプリケーションを必要とする組織の典型的な例です。

多くの企業で、リアルタイムなアプリケーションは電子化されていません。これらのアプリケーションは、ビジネスにおいて不可欠です。ネットワーク機器のメーカー、電話のオペレータ、証券取引所と証券会社、航空会社、物流管理企業、国防情報局などが、その典型的な例です。従来、これらの業界で使用するリアルタイムのアプリケーションの作成には、リアルタイムなインフラストラクチャ・ソフトウェアの開発が必要でした。しかも、市販の選択肢はありませんでした。高速であるが柔軟性に掛けるこれらのシステムは、アプリケーションの変更が必要無い静的な場合に限り、使用できました。しかし、動的な業界はすぐに静的なアプリケーションを凌ぎ、市販の代替製品が出現すると、特別なインフラストラクチャ・ソフトウェアを開発、テスト、管理するためのコストには根拠がなくなります。

### リアルタイムなエンタープライズ

キー・イベントの取得、分析、レスポンスをインテリジェントに行うリアルタイム処理の使用が、すぐれた企業のベンチマーク（判断基準）になろうとしています。

ビジネス・ネットワーク間のメッセージ移動が高速になるにつれ、キー・イベントの取得、分析、レスポンスをインテリジェントに行うリアルタイム処理の使用が、すぐれた企業のベンチマーク（判断基準）になろうとしています。これは、クリティカルなビジネス・プロセスの実行と管理においてのみ重要なわけではありません。顧客は、きめ細かい対応に加え、取引を行う企業から最大の対応を期待しています。

ビジネス・アクティビティの監視、複雑なイベント処理、RFID/センサーベースのアプリケーション、Web ポータルと Web サービスは、最新のエンタープライズに対するアプリケーションの動向に寄与しています。内部関連ルーチンの動的な集合として構成されたこれらのアプリケーションは、サービス指向アーキテクチャ (SOA) として知られる総括的なアプローチの一部です。最新のエンタープライズに対する様々なアプリケーションの動向にも関わらず、データ・ソースの多くは今だにバックオフィスの中にあるうえ、ほとんどアクセスされない大量のレガシー・データで占められ、少量のアクティブな情報がその周囲に存在している状況です。自然な流れとして SOA 概念には、企業のデータ・ソースに接続してアクティブなデータにリアルタイムなパフォーマンスを提供する、アプリケーション層における軽量でリアルタイムなデータ管理が含まれています。

### リアルタイムなデータ管理ソフトウェア

アプリケーションに加えデータを単に収集しキャッシュするだけでは不十分で、企業のデータベースを同じプラットフォーム上に1つのアプリケーションとして配置しても実用的ではありません。

リアルタイム処理から多大なメリットを得るエンタープライズ・アーキテクチャは、アプリケーション層におけるイベント、データおよびトランザクションの管理を提供して、フロントラインのシステムで迅速なレスポンスと深い洞察を実現します。このような管理では、アプリケーションに加えデータを単に収集しキャッシュするだけでは十分ではありません。特に、社内で開発された第一世代の場合、顕著です。また、企業のデータベースを同じプラットフォーム上に1つのアプリケーションとして配置しても実用的ではありません。

必要なのは、使い慣れた強力なインタフェースと汎用な問合せ言語を備えた軽量のインフラストラクチャ・ソフトウェアです。このようなソフトウェアは、既存のバックオフィス・データベース、メッセージング・システム、アプリケーション・サーバーと簡単にインタフェースで接続でき、今日のネットワーク化された豊富なメモリーを持つコンピューティング・プラットフォームをフル活用します。これこそ、Oracle TimesTen が提供するリアルタイムのデータ管理用のインフラストラクチャ・ソフトウェアです。

## アプリケーション層の配置

今日の多くの新しいアプリケーション開発では、顧客との対話を改善し内部処理を合理化して、遅延や超過コストをなくすことを重視しています。これらは、ネットワークの近く、または場合によりネットワーク内にホスト・サービスとして存在する、リアルタイムなアプリケーションです。

それは、従来のバックオフィス・アプリケーションとは異なるプラットフォーム、パフォーマンス、要件を持った新しいエンタープライズ・アプリケーション層です。ビジネス・イベントは、アプリケーションが配布するネットワーク・メッセージに統合され、リアルタイム処理と追加のメッセージ・パブリッシングをトリガーとします。

これらのアプリケーションに対するレスポンス時間とスケーラビリティの目標を満たすためには、アプリケーションを制御するデータの一部または全部を含め、インフラストラクチャ・ソフトウェアとアプリケーションを同じプラットフォームに配置する必要があります。

Oracle TimesTen 製品は、これらの環境にシームレスに統合するよう設計されているため、アプリケーション層への配置と構成用に最適化されたアーキテクチャを備え、高度に作成されたソリューションを実現します。

## 製品情報

Oracle TimesTen のリアルタイムなデータ管理ソフトウェアは、インメモリー・データベース、データ・レプリケーション、キャッシングのテクノロジーに基づいた3つの製品で構成されています。このセクションでは、これらの製品とテクノロジーを概説し、以降のセクションで詳しく説明します。

## Oracle TimesTen In-Memory Database

*Oracle TimesTen In-Memory Database* は、メモリーに最適化されたリレーショナル・データベースで、通信、金融市場、防衛など、今日のリアルタイムな企業と業界が求める迅速なレスポンスと大量のスループットをアプリケーションで実現できます。キャッシュまたは組み込みのデータベースとしてアプリケーション層に配置された Oracle TimesTen In-Memory Database は、標準の SQL インタフェースを介して物理的なメモリーと完全適合したデータベースで機能します。

必要なのは、今日のネットワーク化された豊富なメモリーを持つコンピューティング・プラットフォームをフル活用する軽量のインフラストラクチャ・ソフトウェアです。Oracle TimesTen 製品は、アプリケーション層の配置用に最適化されたアーキテクチャにより、これらの環境にシームレスに統合します。

Oracle TimesTen は、インメモリー・データベース、データ・レプリケーション、キャッシングのテクノロジーに基づいた3つの製品で構成されています。

## Replication – TimesTen to TimesTen

*Replication – TimesTen to TimesTen* は、Oracle TimesTen In-Memory Database のオプションです。サーバー間のリアルタイムなデータ・レプリケーションにより、高可用性と負荷分散を実現します。データ・レプリケーションの構成は、非同期転送または同期転送で、アクティブ/スタンバイまたはアクティブ/アクティブが可能です。また、競合を検出および解決し、障害が発生したサーバーの復元後に自動的に再同期します。データ・レプリケーションは、Cache Connect to Oracle オプションと同時に使用することができます。

## Cache Connect to Oracle

*Cache Connect to Oracle* は Oracle TimesTen In-Memory Database のオプションで、アプリケーション層に存在する Oracle データに対してリアルタイムに更新可能なキャッシュを作成します。このオプションは、バックエンド・システムのコンピューティング・サイクルをオフロードし、リアルタイム・アプリケーションのレスポンス性と拡張性を飛躍的に向上させます。Cache Connect to Oracle には、Oracle データのサブセットを Oracle TimesTen に処理させる、更新を両方向に伝播する、キャッシュされていないデータに対する SQL 要求を自動的に通過させる、障害後にデータを自動的に再同期化させる、などの機能があります。Cache Connect to Oracle は、Replication – TimesTen to TimesTen オプションと同時に使用することができます。

## インメモリー・データベース・テクノロジー

インメモリー・データベース・テクノロジーは、すべての実行時データが RAM にあり、この特性をデータ構造とアクセスのアルゴリズムがブレイクスルー・パフォーマンスのために利用する、リレーショナル・データベースを実装しています。

インメモリー・データベース (IMDB) テクノロジーは、Oracle TimesTen の基盤テクノロジーです。IMDB テクノロジーは、実行時には全てのデータが RAM にあるリレーショナル・データベースを実装しており、データ構造とアクセスのアルゴリズムはブレイクスルー・パフォーマンスのためにこの特性を十分に活用しています。完全にキャッシュされた RDBMS と比較すると、IMDB テクノロジーでは CPU 使用が減少します。これはメモリー・バッファを管理するオーバーヘッドと複数のデータ保管場所 (ディスクおよびメモリー) の評価がなくなるためです。IMDB テクノロジーでは、磁気ディスクはデータベースのプライマリ・ストレージ・ロケーションとしてではなく、永続性とリカバリを実現するために使用されます。

Oracle TimesTen のメモリーに最適化されたパフォーマンスは、トランザクションの特性、永続メカニズム、システム障害からのリカバリに補完されます。ロッキング、マルチユーザーの分離およびロギングについて様々な選択肢があるため、一時的なロックアップ・キャッシュから取引トランザクションや課金システムの構築まで、多様なアプリケーションのシナリオに対応します。

Oracle TimesTen のパフォーマンスは、トランザクションの特性、永続メカニズム、システム障害からのリカバリによって補足されます。

永続性は、コミットされたトランザクションの変更をディスクにロギングし、データベースのイメージを定期的に更新することで実現します。これは「チェックポイント」と呼ばれます。ログの内容をディスクに書き込むタイミングはアプリケーションで設定でき、トランザクションの最後に同期的に行うこともできますし、後で実行してパフォーマンスを向上させることもできます。多くの場合、例えば、セル・ロケーションを 2~3 秒ごとに通信するネットワークでは、携帯電話の場所

の追跡など、トランザクションの金銭的価値が低いまたはトランザクション・データの寿命が短い場合は特に、同期ロギングよりも高いスループットが要求されます。

Oracle TimesTen がサポートするネイティブなインタフェースは標準に準拠しており、一般に標準準拠の他のリレーショナル・データベースと互換性があります。アプリケーションは、JDBC (Java Database Connectivity) または ODBC (Open Database Connectivity) インタフェースを介して、SQL (構造化問合せ言語) を発行します。データベースの定義、レプリケーションの構成およびキャッシュ・グループも、SQL の構文規則に従います。SNMP (Simple Network Management Protocol) は、標準化されたシステム管理の警告を発行するために使用されます。

トランザクション・ログの読取りには、標準の JMS インタフェースを備えたオープンな Transaction Log API (XLA) が用意されています。これは、データベースの更新に対処するアプリケーションを作成する場合に便利です。この観点では、XLA は軽量なトリガーといえます。また、XLA を使用して、Oracle TimesTen から他のデータベース・システムへカスタム・データ・レプリケーションを作成することもできます。

## Oracle TimesTen の物理的な構造

このセクションでは、Oracle TimesTen のベースにあるシステム・コンポーネントをインストール、実行、コンピューティング・リソースの消費の観点から説明します。

Oracle TimesTen は、次のコンポーネントを組み合わせて構成されています。

- 共有ライブラリ
- メモリー常駐のデータ構造
- システム・プロセス
- 管理プログラム
- ディスク上のチェックポイントおよびログ・ファイル

## アプリケーション層の共有ライブラリ

SQL の操作は、開発者がアプリケーションとリンクさせる一組の共有ライブラリに統合されています。

SQL 操作を実装し関数に関連付けられたルーチンは、共有ライブラリのセットに統合されています。開発者は、共有ライブラリをアプリケーションにリンクし、アプリケーションのプロセスの一部として実行します。アプリケーションの接続先である実行可能なプログラムの集合として実装された従来の RDBMS とは異なり、この共有ライブラリのアプローチは、一般的にクライアント/サーバーのネットワーク上で機能します。

通常、データ・マネージャ・ライブラリをアプリケーションに組み込んだ場合、アプリケーション・プロセスが異常終了するとデータベースが潜在的な破損を受けやすくなります。Oracle TimesTen はこの問題を解決しました。Oracle TimesTen のライブラリは、MicroLogging と呼ばれるアルゴリズム (特許出願中) を使用して、アプリケーション・プロセスの障害に対して自己防衛します。インメモリー・データベースは整合性を保ち、他のアプリケーションに影響を与えません。

IMDB テクノロジーは、インメモリーのデータベースとして実装され、アプリケーション、ユーティリティ・プログラム、システム・プロセスが共有ライブラリ・ルーチンを介してアクセスします。ログとチェックポイント用のディスク・ファイル（バックアップ・コピー）は、リカバリ用にオプションとして保持されます。

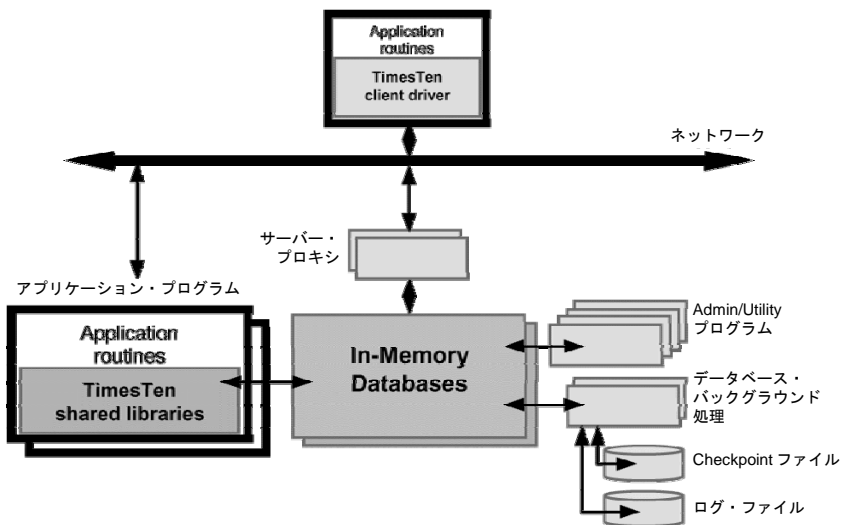


図 1: Oracle TimesTen In-Memory Database のコンポーネント

アプリケーションは、クライアント/サーバー接続を使用して、Oracle TimesTen データベースへアクセスすることもできますが、通常、最高のパフォーマンスは直接アプリケーションにリンクしたときに得られます。

### メモリー常駐のデータ構造

インメモリー・データベースは、オペレーティング・システムの共有メモリー・セグメントに保持され、すべてのユーザー・データ、索引、システム・カタログ、ログ・バッファ、ロック表、一時領域を含みます。複数のアプリケーションで1つの Oracle TimesTen インメモリー・データベースを共有できます。1つのアプリケーションで同じシステムの複数の Oracle TimesTen データベースにアクセスすることもできます。

メモリー常駐のデータベースは、オペレーティング・システムの共有メモリー・セグメントに保持され、ユーザー・データ、索引、システム・カタログ、ログ・バッファ、ロック表、一時領域を含みます。複数のアプリケーションで1つのデータベースを共有でき、1つのアプリケーションで複数のデータベースにアクセスすることもできます。

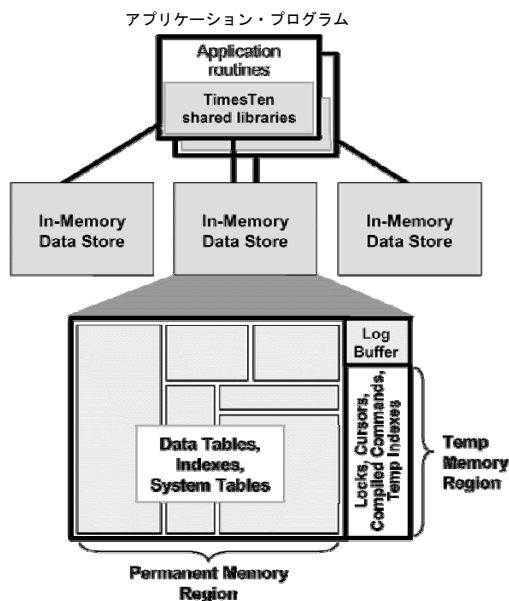


図 2: インメモリー・データベース

## システム・プロセス

バックグラウンド・プロセスは、システム・レベルの開始、停止およびアプリケーション障害の検出サービス、ならびにデータベース・レベルのロード、チェックポイントおよびデッドロック処理サービスを提供します。インスタンス・レベルの *TimesTen* デーモン (1つのシステムで複数のインスタンスを持つことも可能) は1つで、各データベースに対して個別のサブデーモンがあります。

## 管理プログラム

ユーティリティ・プログラムはユーザー、スクリプトまたはアプリケーションによって明示的に呼び出され、対話式 SQL、一括コピー、バックアップ/リストア、データベースの移行、システム監視などのサービスを実行します。

## チェックポイントとログ・ファイル

データベースに対する変更とトランザクション・ログは、定期的にディスクに書き込まれます。データベースのリカバリが必要な場合、Oracle TimesTen は、ディスク上のデータベースのチェックポイントをログ・ファイルにある完了済トランザクションとマージします。通常のディスク・ファイル・システムは、チェックポイントとログ・ファイル用に使用されます。

データベースに対する変更とトランザクション・ログは、定期的にディスクに書き込まれます。

## データ・レプリケーションのテクノロジー

ほぼノンストップの可用性の維持やワークロードの分散が必要な場合、複数のサーバー間で更新を送信するようにデータ・レプリケーションを構成できます。マスター・サーバーは更新を送信し、それをサブスライバ・サーバーが受信するように構成されますが、1つのサーバーをマスターとサブスライバの両方に設定することにより、双方向のレプリケーションが実現します。複数の場所で同時に同じデータが更新されるという稀なケースに対応するために、時間ベースの競合の検出とその解決法が使用されます。

ほぼノンストップで可用性の維持やワークロードの分散が必要な場合、複数のサーバー間で更新を送信するようにレプリケーションを構成できます。1つのサーバーをマスターとサブスライバの両方に設定することにより、双方向のレプリケーションが実現します。

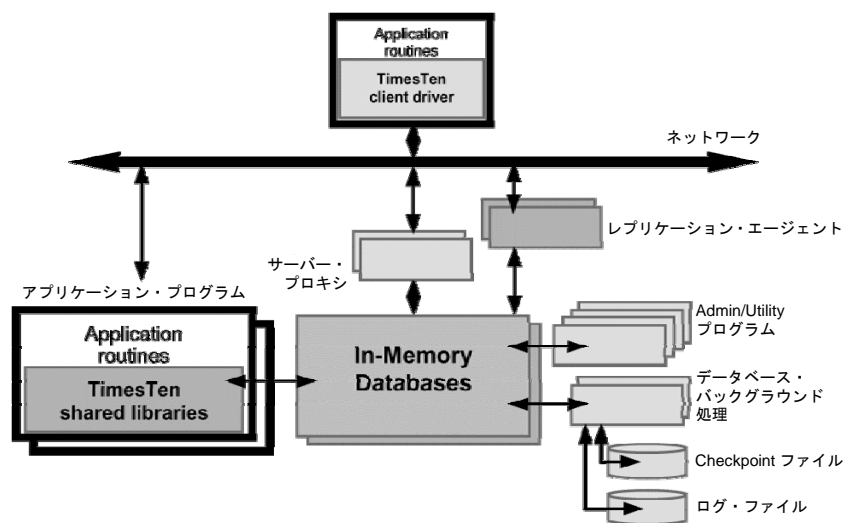


図 3: データ・レプリケーションの追加による可用性の向上

レプリケーションが構成されると、各データベースに対してレプリケーション・エージェント・プロセスが開始されます。複数のデータベースが、同じサーバー上でレプリケーション用に構成されている場合、各データベースに対して個別のレプリケーション・エージェントが存在します。各レプリケーション・エージェントは、1つ以上のサブスライバに対して更新を送信し、1つ以上のマスターから更新を受信できます。これらの接続はそれぞれ、レプリケーション・エージェント・プロセス内の個別の実行スレッドとして実装されます。レプリケーション・エージェントは、TCP/IP のストリーム・ソケットを介して通信します。

最高のパフォーマンスを得るために、レプリケーション・エージェントは、現在のトランザクション・ログを監視することによってデータベースの更新を検出し、可能な場合はバッチでそれをサブスライバに送信します。コミットされたトランザクションのみがレプリケートされます。サブスライバ・ノードでは、レプリケーション・エージェントが、ローレベルのインタフェースを通じてデータベースを更新し、SQL 層でのオーバーヘッドを回避します。

## キャッシングのテクノロジー

Oracle TimesTen が、インメモリー・データベース内で Oracle データベースの一部をキャッシュとして使うと、キャッシュ・グループというデータベース・コンストラクトが作成され、キャッシュされたデータを保持します。また、アプリケーションによって別の共有ライブラリ・ルーチンが呼び出され、システム・エージェントである、キャッシュ・エージェントが開始され、キャッシュと Oracle データベース間ですべての非同期的なデータ転送が実行されます。

Cache Connect to Oracle オプションを使用して、Oracle TimesTen データベース内の Oracle 表の一部をキャッシュすると、キャッシュ・グループのコンストラクトが作成され、キャッシュされたデータが保持されます。また、アプリケーションによって別の共有ライブラリ・ルーチンが呼び出され、キャッシュ・エージェントは、Oracle TimesTen キャッシュと Oracle データベース間でデータ転送を非同期に実行します。

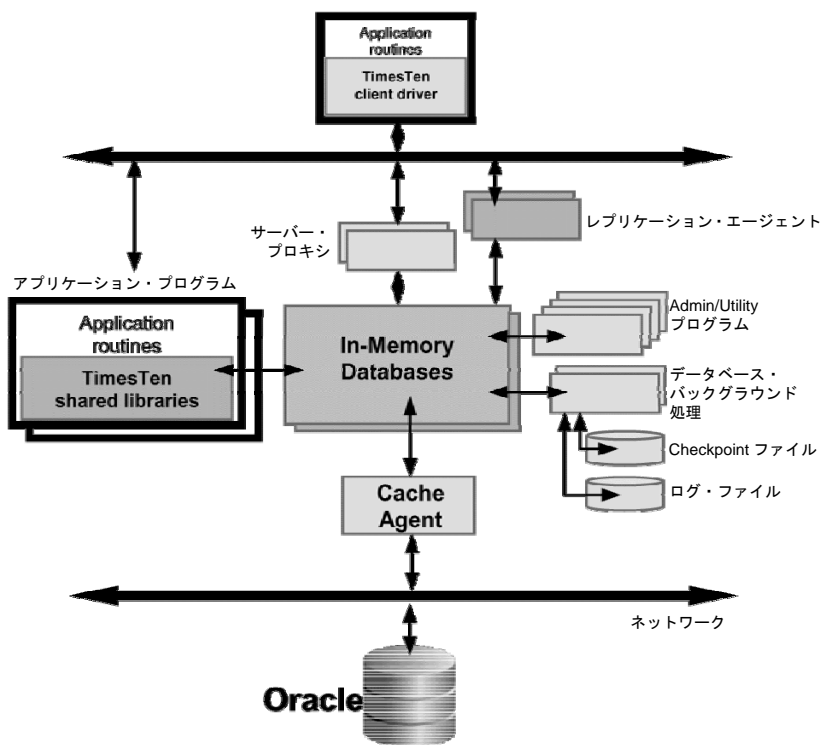


図 4: Oracle データに対するキャッシングの追加

キャッシュ・グループは、主キー/外部キーの関係により論理的な階層をなす 1 つ以上の表の集合です。1 つのキャッシュ・グループ内の各表は、1 つの Oracle のデータベース表と関連しています。

キャッシュ・グループは、主キー/外部キーの関係により論理的な階層をなす 1 つ以上の表の集合です。1 つのキャッシュ・グループ内の各表は、1 つの Oracle のデータベースの表と関連しています。キャッシュ・グループの表には、関連する Oracle の表のすべてまたは一部の行と列を含めることができます。キャッシュ・グループは、ブラウザ・ベースの *Cache Administrator* または SQL 文によって作成または修正できます。キャッシュ・グループは次の機能をサポートしています。

- アプリケーションはキャッシュ・グループ内の表から読取りと書き込みを実行できます。
- キャッシュ・グループを、自動または手動でリフレッシュ (Oracle データベースのデータをキャッシュ・グループに移行) できます。
- キャッシュ・グループを、自動または手動でフラッシュ (Oracle データベースの表に対するキャッシュの更新を伝播) できます。
- Oracle データベースの表またはキャッシュ・グループに対する変更を自動的に追跡できます。

キャッシュ・グループ内の行がアプリケーションで更新された場合、Oracle データベースの対応する行が、同じトランザクションの一部として同期的に更新されるか、または作成されたキャッシュ・グループのタイプにより直後に非同期的に更新されます。非同期の構成では、スループットが飛躍的に向上し、アプリケーションのレスポンス時間も大幅に短縮されます。

Oracle データベースに対する同期的なコミットが何らかの理由で失敗した場合、トランザクションはロールバックされ整合性が保持されます。一方、非同期の構成では、トランザクションはすでにコミットされているため、Oracle データベースの障害を記録するのみで、Oracle TimesTen のトランザクションは自動的にロールバックされません。ただし、パフォーマンスの他にも非同期構成のメリットがあります。それは、Oracle データベースへの接続が失われた場合でも継続して処理され、接続が回復した時点で Oracle データベースへ自動的に更新されることです。

Oracle データベースで発生した変更は、キャッシュ・エージェントを介してキャッシュへリフレッシュされます。キャッシュされたデータへの更新内容は追跡され、定期的にエージェントによりリフレッシュされます。

## IMDB テクノロジーの詳細

メモリー内のすべてのデータを管理し、その環境に対して最適化することによって、インメモリー・データベースのテクノロジーはより効率的に処理を実行でき、結果としてパフォーマンスが大幅に向上しました。

インメモリー・データベースのテクノロジーでは、実行時にデータがどこに存在するかの仮定を変更することにより、パフォーマンスを飛躍的に向上させます。メモリー内のすべてのデータを管理し、その環境に対して最適化することによって、インメモリー・データベースのテクノロジーはより効率的に処理を実行し、結果としてレスポンス性とスループットが大幅に向上しました。

ところで、このパフォーマンスの中核にあるのは何でしょうか。他のすべての RDBMS データをメイン・メモリーに配置するだけでは、アプリケーションで同じ結果は得られないのでしょうか。

RDBMS で実行されるほとんどの操作は、「データは主にディスクに存在する」という仮定のもとに行われます。最適化のアルゴリズム、バッファ・プールの管理、索引を利用した検索方法は、この基本的な仮定を重視します。

一方、Oracle TimesTen ではデータはメイン・メモリーにあるため、データに対してより直接的なルートをとることができ、コード・パスが短くなり、アルゴリズムも構造も簡潔になります。

これらの2つのアーキテクチャを比較すると、多数のパフォーマンスの改善がどのように実現できるかを明確に示す重要な違いがいくつかあります。たとえば、クエリーの最適化アルゴリズム、バッファ・プールの管理、索引構造における違いなどです。

## クエリーの最適化

ディスクの I/O はメモリーのアクセスよりもコスト高であるため、ディスクベースの RDBMS は、データがディスク上に存在すると仮定する必要があります。

クエリーの最適化アルゴリズムは、ディスクベースのシステムとメモリーベースのシステムでは異なります。RDBMS の最適化では、データは主にディスク上に存在すると仮定しています。ある任意のタイミングでデータがディスク上にある場合や、データがメイン・メモリーにキャッシュされた動的な実行時環境においては、ディスクベース・システムに対して最悪の事態を想定する必要があります。ディスクの I/O はメモリーのアクセスよりもコスト高であるため、ディスクベースの RDBMS は、データがディスク上に存在すると仮定します。データは、パフォーマンスのボトルネックのポイント、つまりディスクの I/O を減らすために最適化されています。この仮定に基づく限り、ディスクベースの最適化では、主に（または完全に）メイン・メモリーに存在するデータに対して常に最適なプランを提供することはできません。

一方、IMDB のテクノロジーでは、データがメイン・メモリーにあることがわかっているため、より単純な仮定のもとでクエリーを最適化します。このテクノロジーでは、データがディスクに存在するという最悪の事態を想定しなくてよいため、コストの評価もより簡潔になり、さらに高い整合性が得られます。

## バッファ・プールの管理

バッファ・プールは、ディスクベースのデータ管理ソリューションでは必要ですが、メモリーベースのデータ管理では必要ありません。データはすでにメモリー内にあるためです。

従来の RDBMS アーキテクチャでは、メイン・メモリーにキャッシュされたデータに対しバッファ・プールを保持することが必要でした。SQL クエリー・プロセッサでデータのページが必要になった場合、アクセス・メソッドでメモリー内のデータに対するバッファ・プールが最初に検索されます。データがバッファ・プールにあっても、ほとんどの場合には以降の処理にプールのコピーが必要です。追加のデータ・コピーを伴う、このバッファ・プールのメンテナンスと管理では、アプリケーションでデータを使用可能にする最初の負荷と比較して、負荷が大幅に増加します。バッファ・プールは、ディスクベースのデータ管理ソリューションでは必要ですが、メモリーベースのデータ管理では必要ありません。Oracle TimesTen では、バッファ・プールを必要としません。データはすでにメイン・メモリーに存在しているためです。このため、コード・パスとエンジンのフットプリントが小さくなる、コピーの処理が不要になる、アルゴリズムが簡潔になる、アプリケーションに対してより迅速にデータが提供される、などが実現します。

さらに、これらの RDBMS 製品では、実行時にディスクベースの仮定を動的に転換する設計がされていません。ディスクベースの仮定は、システムのコード・ベース内で複雑に組み合わせられ、適当に配置された少数の if-then-else 文を使用して再形成します。この組み合わせの例は、2つのアーキテクチャのツリー構造を索引付けする場合のディスクベースの仮定です。

## 索引構造

### B<sup>+</sup>ツリー

データがいったんメモリー内に保持されると、索引付けスキームの目的は、I/O の削減ではなく CPU サイクルの削減になります。

従来のRDBMSのB<sup>+</sup>ツリー索引ページでは、キー値とデータに対するポインタが、B<sup>+</sup>ツリー・エントリに保持されています。B<sup>+</sup>ツリー・ノード（ディスク上のページ）は複数のB<sup>+</sup>ツリー・エントリで構成されています。各エントリは、索引のデータ値と該当する索引ノードの次のページ番号、または検索されたデータ行を持つディスク・ブロックのページ番号を保持しています。この構造により、ディスクのI/O減少にとって理想的な、非常にフラットでワイドなツリーが実現します。B<sup>+</sup>ツリーの構造での目的は主に、データ・ファイルの索引付けの検索に必要なディスクI/Oの回数を削減することです。B<sup>+</sup>ツリーは、(1)キー値をB<sup>+</sup>ツリーのノード自身に保持する（ディスクI/Oを減らす）、(2)ノード内にできるだけ多くの索引エントリを保持する（1回のI/OでサービスできるB<sup>+</sup>ツリーのエントリ数を増やす）ことで、これを実現します。

この構造は、データと索引のファイルがディスク上にある場合は有効ですが、データがメモリーにある場合は適していません。データがメモリーに保持されると、索引付けのスキームの目的は、I/Oの削減ではなくCPUサイクルの削減になります。CPUサイクルは、圧縮された索引値をB<sup>+</sup>ツリー内で展開し比較することによって消費されます。ディスクからメモリーへ読取り済のデータと索引を保持したバッファの管理も、CPUサイクルを増加させます。

### Tツリー

Oracle TimesTenでは、処理を表す図は簡単です。Tツリーは、メイン・メモリーのアクセスに対して最適化されています。Tツリーは、サイズとアルゴリズムの両面で、B<sup>+</sup>ツリーに比べ低コストな索引エントリを備えています。Tツリー・ノード間の接続は、「小なりイコール (≤)」と「大なり (>)」のポインタによって達成されます。これらのポインタは、ディスク上のページ・アウトではなくメモリーの場所を参照します。2つを比較すると、Tツリーの検索アルゴリズムでは、検索している値が対象のノードか、あるいはメモリー内の他の場所にあるのかが分かります。新しい索引ノード・ポインタの分岐ごとに、その検索領域は半分になります（正しいバイナリ・サーチの定義）。

T ツリー索引は、データがメモリーに常駐するアーキテクチャに対して最適化されています。すべての行はメモリーに存在するため、T ツリーではキー値を持つ必要はなく、単に実際の行をポイントするだけです（この例では説明のためにキー値を使用しています）。

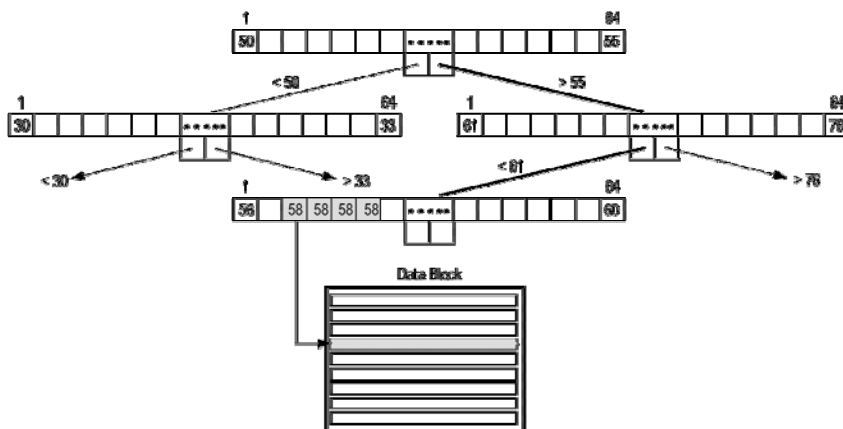


図 5: メモリー常駐データベースに対する T ツリー索引

データがディスクに常駐するという仮定を排除すると、すべてが単純化され洗練され、よりコンパクトで高速になります。

メイン・メモリーのデータ管理では、その目的は必要な領域の削減、ディスク I/O の除去、アルゴリズム、コード・パスおよびフットプリントの縮小です。T ツリーは、キー値を索引ノード自体に保持するのではなく、データ行のメモリーに存在している値を単にポイントすることによって、必要な領域を減少させます。T ツリーでは、索引ツリーの構造はすべてメモリーに存在するため、ディスクの I/O は発生しません。T ツリーでは、アルゴリズムが非常に簡潔になるため、コードが単純化されコード・パスの長さが減少します。また T ツリーを使用すると、B<sup>+</sup> ツリーベースのアクセスのすべてのメリットが得られるだけでなく、処理とサイズもきわめて経済的になります。

### 差異の理由

アーキテクチャの違いに関する前述の例では、データがディスク上に存在する（またはデータの場所が曖昧）という仮定を排除すると、複雑さが大幅に減少します。マシン命令の数は少なくとも 10 分の 1 になり、バッファ・プールの管理がなくなる、余分なデータ・コピーが不要になる、索引ページが縮小される、構造がシンプルになる、などのメリットがあります。基本的にデータがメモリーに常駐すると仮定した場合、すべてが単純化して洗練され、よりコンパクトで高速になります。

### コストパフォーマンスにおける新しい様式

多くのアプリケーションでは、インメモリー・データベースのテクノロジーを利用することによって、競争に勝つ、ユーザーの満足度を向上させる、ROI（投資収益率）を上げる、などの新しいメリットが得られます。

処理時間の半分がデータ管理であるアプリケーションは、ほぼ 2 倍の能力を実現できます。

パフォーマンスの改善は、割合ではなく倍数で測定することができます。処理時間の半分以上をデータ管理に、残り半分をアプリケーションの領域またはビジネス・ロジックに費やしているアプリケーションが、インメモリーのデータ管理製品を使用した場合、2 倍近くの能力を達成できます。システムの能力が 2 倍になると、市場戦略、アーキテクチャの経済性、システムの実装の点で選択肢が大幅に増えます。

## 卓越したパフォーマンス

### スケーラビリティ

Oracle TimesTen のテクノロジーは、対称型マルチプロセッサ・コンピュータで複数の CPU のメリットを活かす設計がされています。次のグラフは、4-CPU Linux/Intel のシステムでの Oracle TimesTen 7.0 のトランザクション・スループットを示しています。各トランザクションは、図のように、1つの SQL の更新または選択（読取り）処理を実行します。その結果を、CPU が 1つ、2つおよび4つの場合で示します。

Oracle TimesTen は、トランザクションに単一の SQL 処理が含まれている場合、毎秒 10,000 件の更新トランザクションから 100,000 件の読取り処理まで、卓越したスループットを実現します。これらの結果は、3.0 GHz プロセッサで 4CPU Linux/Intel システムを使用した場合に得られました。

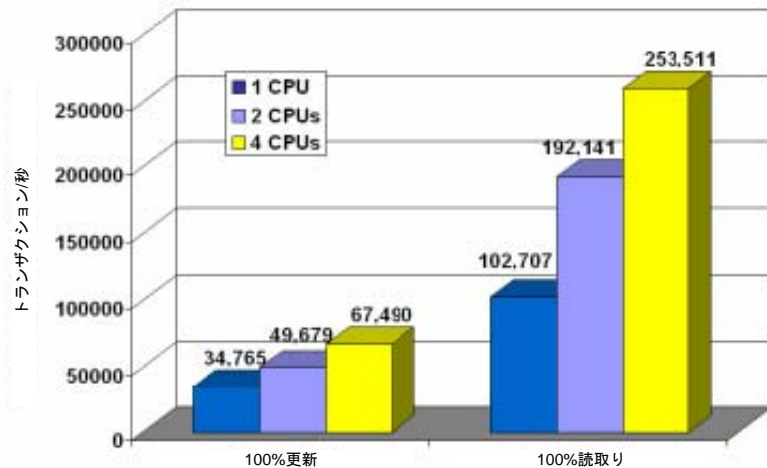


図 6: Oracle TimesTen のスループット

最も簡単で、そのため負荷が最も高い 100% read のテストでは、キー値の検索を使用して取得できる個々のレコードの数を測定します。この場合、平均で毎秒 253,511 件の読取りが行われます。1つの CPU のみの場合は、読取り速度は 1 秒あたり 102,707 でした。

更新処理では、読取り処理に比べ多くの処理が必要です。リカバリのために変更のログ（記録）が必要であることが、その理由のひとつです。このため、4 CPU では毎秒 67,000 件を超えるレコードが更新され、1 CPU ではその数は 34,000 を超えます。

これらの純粋なワークロードは特定のビジネス・アプリケーションについて表したのですが、この結果は Oracle TimesTen の効率が非常にすぐれていることを示しており、現実にはどのような SQL 処理が混在する場合でも、1 秒あたり最大で「1,000 倍から 10,000 倍」のスループットが得られます。さらにボリュームが必要な場合、Oracle TimesTen は 4CPU システム以上に拡張できます。性能結果は、プロセッサのプラットフォームごとに異なります。

## レスポンス時間

スループットの向上は、個々のトランザクションのレスポンス時間が短縮された結果として起こる2次的なメリットです。スループットが向上すると、平均のレスポンス時間が短縮されます。Oracle TimesTen の卓越したスループットにより、現行のシステムで、データ管理のワークロード全体でマイクロ秒レベルのレスポンス時間が得られます。マイクロ秒は、1秒の100万分の1です。前述のスループットのワークロードに対する平均のレスポンス時間を、次の2つのグラフに示します。

Oracle TimesTen の待機時間（レスポンス時間）は、一行の処理であればマイクロ秒（100万分の1秒）で測定されます。アプリケーションに対する全体のレスポンス時間には、データ管理の部分以外も含まれるため、データ処理集中型のアプリケーションで最大のメリットが得られます。これらの結果は、3.0 GHz の4CPU Linux/Intel システムを使用した場合に得られました。

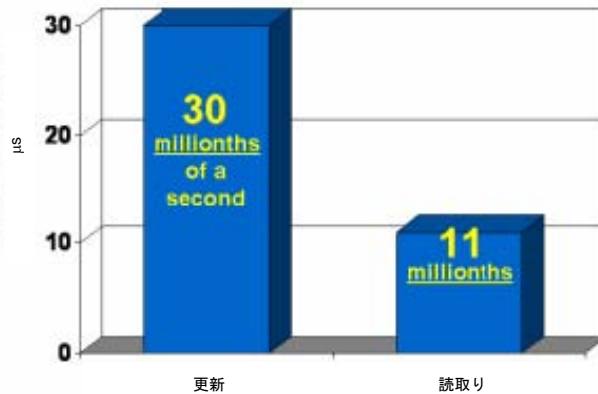


図 7: Oracle TimesTen 待機時間

トランザクションのレスポンス時間は、構成要素の処理ステップ（ビジネス・ロジック、ネットワークング、I/O、データ管理など）のレスポンス時間を平均したものである、ということをご認識しておいてください。Oracle TimesTen だけのパフォーマンスは、全体のレスポンス時間だけでは測定できません。場合によっては、1つのステップ（通常はI/Oまたはネットワークング）がレスポンス時間の大半を占めることがあります。

たとえば、アプリケーションとデータ・マネージャ間の直接接続の代わりにクライアント/サーバーのネットワークングを使用した場合、ネットワーク上の各ラウンド・トリップに対してレスポンス時間がミリ秒またはそれ以上長くなります。同様に、変更のロギングに対して同期的なI/Oを使用した場合、ディスク処理の遅延がレスポンス時間に反映されます。これらの2つのケースで、データ管理のコンポーネントがマイクロ秒単位で完了したとしても、アプリケーション全体のレスポンス時間は、ネットワークおよびI/Oの待機時間の要因となります。

主要なデータ管理コンポーネントを備えたアプリケーションでは、Oracle TimesTen のスピードアップのテクノロジーによりレスポンス時間を大幅に短縮することができます。高度なパフォーマンスを実現するアプリケーションは、直接接続のアプリケーションで、同期的なロギングを使用しデータ管理を集中して実行できます。検索集中型のオンライン予約システム、プレゼンスおよびロケーションベースの通信サービス、コンタクトセンターのルーチン・エンジンなどが、その例です。

通常、パフォーマンスは、トランザクションのサイズ、非データ・アクセス・ロジックの量、プラットフォームのパフォーマンスなど、様々な要因により大きく異なります。

## リアルタイム機能

このセクションでは Oracle TimesTen の機能概要を取り上げ、開発者がリアルタイム・アプリケーションの記述に使用するインタフェースと機能を説明します。

## データ管理

### リレーショナル・データベース・モデル

Oracle TimesTen のデータ・マネージャは、真のリレーショナル・データベース・モデルを実装しています。これは、ハイブリッド・モデルでも最上位に別のリレーショナル・ビューを結合させた別のモデルでもありません。一般的な RDBMS 製品に精通した開発者なら、すぐに生産性を高めることができます。

リレーショナル・モデルの場合、ユーザー・データは表内の行（レコード）に格納されます。行は列（データ・フィールド）のリストとして定義され、それぞれ特定のデータ型を持ちます。索引は、明示的または暗黙的に作成されて表内におけるキーベースまたは値範囲での検索を高速化します。Oracle TimesTen は、ハッシュ索引と（前述の）T ツリー索引をサポートしています。一意性や参照整合性（表間の親子関係）など、いくつかの制約が自動的に設定されます。

1つの表では、1つの主キーを設定できます。主キーは1つ以上の列で構成されず。主キーは行に対する一意の ID で、Oracle TimesTen では、同じ主キー値を持つ別の行を挿入することが自動的に禁止されます。表に対して、別の表の主キーに関連付けられた1つ以上の外部キーを設定できます。外部キーにより、2つの表間に親子関係が定義されます。たとえば、別の表で関連付けられた外部キーが同じ値を持っている場合、その値を持つ主キーの行は削除できません。

マテリアライズド・ビューの定義はオプションの構造です。マテリアライズド・ビューは読取り専用の表で、正規の（ベース）表から導出されます。マテリアライズド・ビューには、1つ以上の表の一部またはすべてのデータを含めることができ、計算値（合計や平均など）を含めることもできます。マテリアライズド・ビューは、ベース表に対する変更が必要な場合にはいつでもシステムによって自動的に更新されるため、アプリケーションは頻繁に導出されるデータにより高速にアクセスできます。マテリアライズド・ビューを使用しないと、アプリケーションでデータを読取るたびに、結合などの計算のオーバーヘッドが発生します。マテリアライズド・ビューは、イベント処理（後述の説明を参照）をサポートするうえで特に有効です。

キャッシュ・グループは、対応する Oracle 表のセットにマッピングされた階層的な表で、同じデータの一部またはすべてが含まれています。キャッシュ・グループは読取りと書き込み両方ができ、データに対する更新は、キャッシュ・グループと Oracle データベース間で自動的に伝播されます。

Cache Connect to Oracle を使用すると、インメモリー・データベース内で1つ以上のキャッシュ・グループを定義できます。キャッシュ・グループは、対応する一組の Oracle データベースの表にマッピングされた一組の階層的な表で（外部キーによって制約される）、同じデータの一部またはすべてが含まれています。キャッシュ・グループは読取りと書き込み両方ができ、データに対する更新は、Oracle TimesTen と Oracle データベース間で自動的に伝播されます。

データ管理の処理は、業界標準の構造化問合せ言語（SQL）によって表現されず。主な演算は、SELECT（read）、UPDATE、INSERT、DELETE です。1つ以上の表の列を結果セットに組み合わせるリレーショナルな結合処理も、基本的な SQL92 の標準定義で制定された SQL オプションとしてサポートされています。

表、索引、マテリアライズド・ビュー、キャッシュ・グループは、データベース内で作成され保守されます。アプリケーションは Oracle TimesTen データベースに接続し、その中のデータに対してデータ管理処理を要求します。同じコンピュータ・システム上に複数のデータベースを定義し、同じアプリケーションでアクセスすることもできます。ただし、1つ以上のデータベースにまたがって単一の処理（2つの表間での結合など）を行うことはできません。アプリケーションは、2フェーズ・コミットに対する Oracle TimesTen の標準 XA（または JTA for Java）インタフェースのサポートにより、分散トランザクションに関与できます。

表、索引、マテリアライズド・ビュー、キャッシュ・グループは、データベース内で作成され保守されます。同じコンピュータ・システム上に複数のデータベースを定義し、同じアプリケーションでアクセスすることもできます。

前述の Oracle TimesTen の IMDB テクノロジーでは、データ管理サービスがいかにリアルタイムに行われるかを説明しました。繰返し述べますが、単にすべてのデータベースがメモリー内に存在するだけでなく、I/O が消滅します（実際には、ごく少数の I/O になる）。データのメモリー内存在が保証されることにより、データの場所が曖昧であることを説明する処理命令は必要ありません。処理命令には間接的な検索アルゴリズム、バッファ管理のオーバーヘッド、データ・コピー、最適化された索引構造、実行計画が含まれています。

また、Oracle TimesTen のデータベース内の情報のレイアウトは、高速処理とメモリー保持のために最適化されます。ディスクベースの RDBMS では、ディスクとメモリーの構造はほとんど同じです。メモリー・ページは I/O のブロック・サイズと同じで、I/O の遅延を最少にします。Oracle TimesTen では、処理命令を最少にするメモリー構造を設計できるため、これは重要ではありません。

## 永続性と並行性

Oracle TimesTen は、データ管理システムの「ACID」特性（原子性、整合性、独立性、永続性）に準拠しています。

Oracle TimesTen は、データ管理システムの「ACID」特性（原子性、整合性、独立性、永続性）に準拠しています。これらの特性により、マルチユーザーのシステムでは、各トランザクションが一度に1つのトランザクションのみが実行されるかのように機能し、システムではコミットされたトランザクションの結果が失われません。これらは、データ・マネージャにとって最も重要な特性であり、Oracle TimesTen はこの内容に完全準拠します。

一般に、インメモリー・データ・マネージャでは、システム障害によるデータの損失は避けられないと考えられていますが、これは誤解です。実際に、Oracle TimesTen では従来型のデータベースでトランザクションとデータを永続させるためのテクニックと同じテクニックが使用されています。主な違いは、Oracle TimesTen では「永続性の程度」と「全体のスループット」とのトレードオフでアプリケーションを制御する点です。

一般に、インメモリー・データ・マネージャでは、システム障害によるデータの損失は避けられないと考えられていますが、これは誤解です。実際に、Oracle TimesTen では従来型のデータベースでトランザクションとデータを永続させるためのテクニックと同じテクニックが使用されています。トランザクション指向のすべてのシステムでは、変更のロギングとディスクに存在するデータベースの内容の定期的なリフレッシュを組み合わせることによって永続性を実現します。

主な違いは、Oracle TimesTen では「永続性の程度」と「全体のスループット」とのトレードオフで、アプリケーションを制御する点です。Oracle TimesTen では、この永続性またはスループットを重視することによって選択肢を提供します。

## ディスクの操作

アプリケーションですべての変更を維持する必要がある場合、トランザクションのコミット処理の一環として、ログ・レコードがディスクへフラッシュされます。いくつかのトランザクションの喪失よりもパフォーマンスを最大化する方が重要な場合には、ディスクに対するログ・レコードの書込み回数を各トランザクションのコミットから非同期に減少できます。いずれの場合でも、Oracle TimesTen のデータ・マネージャは、複数のトランザクションをまとめてグループ・コミットし、ディスクへの書込みを最少にします。

データベースは、定期的にユーザー指定の頻度およびログ容量に基づいて、自動的にチェックポイントされます。Oracle TimesTen では、ファジーなチェックポイント操作もサポートしているため、実行中のアプリケーションに対する影響を最少にしつつバックグラウンドで実行することもできます。システム障害からのリカバリは、ログ・レコードと最新のチェックポイント・ファイルのマージによって行われます。

Oracle TimesTen のデータベースは、永続的または一時的に作成することができ、排他または共有アクセス用に設計することができます。

Oracle TimesTen のデータベースは、永続的または一時的なものとして作成でき、排他（つまりシングルユーザー）または共有（マルチユーザー）アクセス用に設計できます。永続的なデータベース（チェックポイント・ファイル）は、使用されないときはディスク上にあります。一時的データベースは単にメモリー内に存在し、使用されないときは消去されます。

トランザクション・ロギング・サーバーには2つの目的があります。1つはロギングによって、システム障害後、永続的なデータベースに対してトランザクションをリカバリすることです。もう1つは、ロギングにより Oracle TimesTen のデータ・マネージャが共有アクセス・モードで使用された場合にデッドロックの検出と絞込みを行うことです。ロギングされていないデータベースでは、優れたパフォーマンスが得られます。ただし、ロギングが無効の場合はトランザクションのロールバックまたはリカバリが不可能になり、データベース・レベルで設定できるのはロックのみです（行レベルまたは表レベルのロッキングはできません）。ロギングの無効化は、最初から再作成できるシングルユーザーの処理で障害が発生した場合にのみ適しています。

永続的な共有ディスク・ロギング機能を備えたデータベースは、最も一般的に配置される構成です。データベースが一時的で、システム障害の発生後にデータベースをリカバリしても意味がないようなアプリケーションでは、一時的なデータベースが一般的です。コール・センターのアプリケーションにおける非常に変わりやすいステート情報が、その例です。

## ロッキングと分離

ロックは Oracle TimesTen によって適用され、あるユーザーが読取り中または変更中のデータが他のユーザーによって変更されるのを防止します。ロックは、データベース、表または行レベルでデータベース・オブジェクトに設定される「予約」です。

ロックは、データベース、表または行レベルでデータベース・オブジェクトに設定されています。行レベルのロッキングによって、マルチユーザーの最大同時処理性が得られます。Oracle TimesTen では、行レベルのロッキングがデフォルトです。

行レベルのロッキングによって、マルチユーザーの最大同時処理性が得られます。Oracle TimesTen データベースでは、行レベルのロッキングがデフォルトです。アプリケーションはプロシージャをコールして、実行時にロッキングのレベルを行またはデータベースに変更できます。Oracle TimesTen のオプティマイザがメリットがあると判断した場合、またはアプリケーションがオプティマイザを操作するプロシージャをコールして、トランザクションの処理中に表レベルのロッキングを適用する場合、表レベルのロッキングが使用されます。

分離により、処理を読取るためのレスポンスに適用されるロックの動作が指定されます。アプリケーションでは、**SERIALIZABLE** (直列可能) または **READ-COMMITTED** (読取り専用、デフォルト) の 2 つの分離レベルから選択できます。

アプリケーションでは、**SERIALIZABLE** と **READ-COMMITTED** の 2 つの分離レベルから選択できます。

**SERIALIZABLE** 分離は、全トランザクションを通じて一貫した予測可能なデータ値を要求するトランザクションで使用されます。**SERIALIZABLE** 分離を使用すると、あるユーザーが読取り中のレコードは、トランザクションのコミット後またはロールバック後でなければ、他のユーザーは更新または削除できません。また、この期間中、他のユーザーは結果セットの一部であっても、これらの読取り処理から新しいレコードを挿入することはできません。つまり、**SERIALIZABLE** 分離では、読取り処理を何回繰り返しても同じ結果が保証されます。データベース・レベルのロッキングでは、トランザクションは、デフォルトの **SERIALIZABLE** 分離によって効率よく機能します。

**READ-COMMITTED** 分離では、マルチユーザーの最大同時処理性が得られます。この分離では、読取りを実行したユーザーのみがコミットされたデータ値を参照できますが、書込みを実行するユーザーはロックの解除を待ったり読取り中のレコードへのロックの設定を待ったりすることはできません。

**READ-COMMITTED** 分離では、マルチユーザーの最大同時処理性が得られます。この分離では、読取りを実行したユーザーのみがコミットされたデータ値を参照できますが、書込みを実行するユーザーはロックの解除を待ったり読取り中のレコードに対するロックの設定を待ったりすることはありません。この分離の効果を実現するために、Oracle TimesTen では、更新中のレコードについて 2 つのコピーを作成します。1 つは更新前のもの、つまり、読取りを実行したユーザーにとっては「コミット」された値で、もう 1 つは書込みを実行するユーザーが更新可能なものです。読取りを実行したユーザーは「読取り」バージョンへのアクセスをブロックされず、書込みを実行するユーザーは読取りを実行したユーザーを待機しません。

## 問合せ処理

Oracle TimesTen では、適切なオープン標準の採用が基本的な方針です。SQL、JDBC、ODBC、JMS は、アプリケーションが Oracle TimesTen のデータベースとインターフェース接続するために使用するインターフェースです。これらのインターフェースの実装は、Oracle TimesTen のアーキテクチャに対して高度にチューニングされています。

高度なパフォーマンスのために設計された特殊な製品では独自の API を使用しますが、Oracle TimesTen ではその必要はありません。Oracle TimesTen では、適切なオープン標準の採用を基本的な方針としてきました。主な標準としては SQL (構造化問合せ言語)、JDBC (Java Database Connectivity)、ODBC (Open Database Connectivity)、JMS (Java Message Service) があり、データベースへのアクセス

には必ずこれらを使用します。これらのインタフェースの実装は、Oracle TimesTen  
のアーキテクチャに対して高度にチューニングされています。

SQL は、リレーショナル・データベース用の問合せ言語として、長い間広く使用されてきました。ベースとなるストレージおよび索引付けされた詳細の抽象化が、SQL の主なメリットのひとつです。また、SQL はどのように行うかではなく何を行うかを表す、使いやすい言語です。SQL には、索引、データ型、物理的なレイアウトに対する参照がなく、表、列および検索条件のみがあります。Oracle TimesTen のオプティマイザ機能は、索引の存在やキー値の分散などの要因に基づいて、問合せに対する最速のレスポンス方法を判断します。

この抽象レベルにより、ベースとなるデータ・モデルをチューニングすることも拡張することも、既存のアプリケーションに影響を与えずにできます。アプリケーション・モジュールを追加し、必要なデータ表と列を追加するだけで、ただちに新しいサービスを本番環境に追加できます。SQL などの問合せ言語を使用してデータ・マネージャの内部からアプリケーションを遮蔽しないと、ほとんどのアプリケーションは、新しいデータ・フィールドの追加によって影響を受けます。

## データ・レプリケーション

レプリケーションは SQL 文によって構成され、指定された表またはデータベース全体へ適用できます。効率の向上とオーバーヘッドの低減のために、Oracle TimesTen ではトランザクションログ・ベースのレプリケーション・スキームを使用します。

レプリケーションは、データベース間でデータをコピーするプロセスです。Oracle TimesTen のレプリケーションの基礎をなす基本的な方針は、ミッション・クリティカルなアプリケーションに対してパフォーマンスが受ける影響を最少にしたうえで、データを継続して使用可能にすることです。複数のデータベースに対して最大のパフォーマンスを維持しつつ、オンラインでの更新と保守をスムーズに行ってユーザーの負荷を分散させるためには、障害時のリカバリのロールだけでなくレプリケーションも有効です。

Oracle TimesTen は、マスター/サブスライバのレプリケーション・モデルに従います。これによって、コミットされた変更は、ソースから 1 つ以上のサブスライバ・データベースへコピーされます。レプリケーションは SQL 文によって構成され、指定された表またはデータベース全体へ適用できます。効率の向上とオーバーヘッドの低減には、トランザクションログ・ベースのレプリケーション・スキームを使用します。

各マスターおよびサブスライバ・データベースにおけるレプリケーションは、TCP/IP ストリーム・ソケットを介して通信するレプリケーション・エージェントにより制御されます。マスター・データベースのレプリケーション・エージェントは、トランザクション・ログからレコードを読み込み、レプリケートしている要素に対し変更が検出された場合、それをサブスライバ・データベースのレプリケーション・エージェントへ転送します。サブスライバのレプリケーション・エージェントは、データベースに対して更新を適用します。マスターが更新を転送するときにサブスライバ・エージェントが動作していない場合、サブスライバに更新が適用されるまでマスターはトランザクション・ログ内で更新を保持します。

## パフォーマンスと整合性のバランス

マスターとサブスライバのデータベースには、サブスライバが更新を正しく受け取りコミットしたことを確認するための内部メカニズムがあります。これらのメカニズムにより、更新はサブスライバにおいて 1 回のみ適用されますが、それはアプリケーションでは見えません。

Oracle TimesTen のレプリケーションは、デフォルトでは非同期のメカニズムです。非同期のレプリケーションの使用により、アプリケーションはマスター・データベースを更新し、サブスライバが更新を受け取るのを待つことなく作業を継続します。マスターとサブスライバのデータベースには、サブスライバが更新を正しく受け取りコミットしたことを確認する内部メカニズムがあります。これらのメカニズムにより、更新はサブスライバにおいて 1 回のみ適用されますが、それはアプリケーションでは見えません。

非同期のレプリケーションにより最高のパフォーマンスが得られますが、アプリケーションは、サブスライバ上でレプリケートされた要素を受け取るプロセスから完全に分離されます。レプリケートされたデータがマスターとサブスライバのデータベース間で高い確率で整合性を保持する必要があるアプリケーションに対して、Oracle TimesTen は、次の 2 つのリターン・サービスを提供します。

- *リターン・レシート・サービス*は、サブスライバが更新を受け取ったことをレプリケーションが確認するまでアプリケーションをブロックすることによって、レプリケーションのメカニズムとアプリケーションをゆるやかに結合します（つまり同期します）。
- *リターン・2 セーフ・サービス*は、サブスライバが更新を受け取りコミットしたことをレプリケーションが確認するまで、アプリケーションをブロックすることによって、レプリケーションを完全に同期化します。

非同期のレプリケーションでは、最高のパフォーマンスが得られません。Oracle TimesTen は、レプリケートされたデータがマスターとサブスライバのデータベース間で整合性を高い確率で保持する必要があるアプリケーションに対して、2 つのリターン・サービスを提供します。

リターン・サービスを使用するアプリケーションでは、マスターとサブスライバのデータベース間で、データの高度な整合性と一貫性を保証するために多少のパフォーマンスを犠牲にします。マスターで障害が発生した場合、アプリケーションは、マスターでコミットされたトランザクションがサブスライバされたデータベースでも保持されることを高い確率で保証します。リターン・レシート・レプリケーションでは、リターン・2 セーフのように完全に同期をとらない分だけ、パフォーマンスに関する影響が減少します。

## レプリケーションのトポロジ

マスターおよびサブスライバとしてデータベースを設計することにより、双方向のレプリケーションを構成できます。また、Oracle TimesTen ではマルチノードの「多方向」レプリケーションが可能です。これによって、ホット/スタンバイとアクティブ/アクティブ両方の構成を対象とした広範囲なレプリケーション・トポロジを提供します。後者は、分割ワークロード（レプリケートされた各表に 1 つだけマスターがある）と分散ワークロード（1 つの表に複数のマスターがある）に分けることもできます。分散ワークロードの構成では、アプリケーションは 2 つのシステム間で処理を分割するため、レプリケーションの衝突は発生しません。イベントの衝突が発生した場合、タイムスタンプベースの衝突検出と解決のメカニズムにより整合性のとれないレプリカの生成が防止されます。

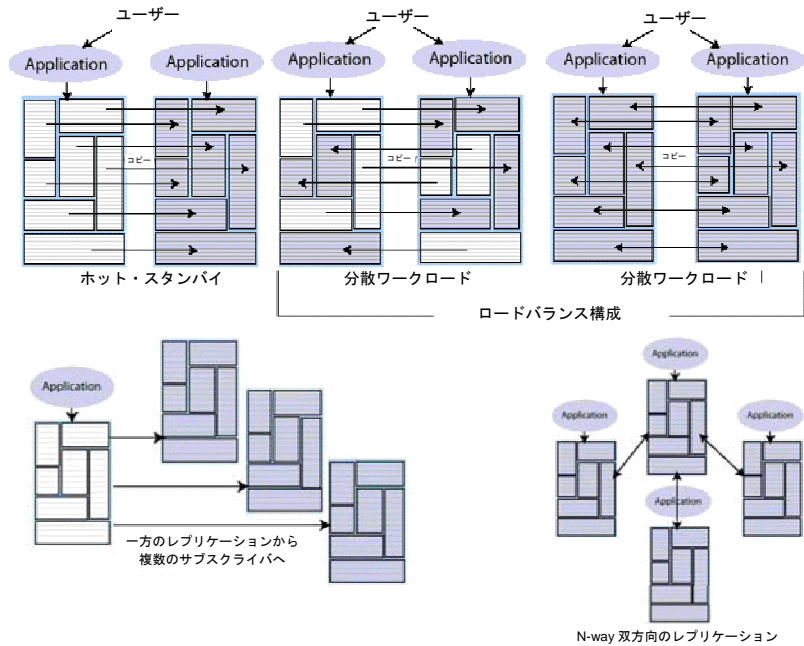


図 8: レプリケーション構成オプション

サブスライバをプロパゲーターとして機能するよう定義することもできます。プロパゲーターは、レプリケートされた更新をマスターから受け取り、それをサブスライバへ渡します。プロパゲーターは、イントラネット内のサーバー間など、帯域幅が低いネットワーク接続でレプリケーションのパフォーマンスを最適化する場合に有効です。

## キャッシング

### カスタマイズされた統合

「主要なサービス・プロバイダは、我々の製品を使用して、スピードが重視される請求データを管理しているため、ボトルネックと停止時間は許されません。Oracle TimesTen と Oracle Real Application Clusters を使用することで、信頼性と拡張性に富んだ実証済のリアルタイム・データ管理の基盤を実現し、顧客によりすぐれた環境を提供できます。」

Ed McKee 氏  
Director of Applications, Interact, Inc.

一般的に、Oracle TimesTen は、ディスクベースの RDBMS とともに配置され、データをリアルタイムで取得し処理が必要な場合に使用されます。データがリアルタイムではないステートへ遷移すると（株式取引が終了した、または通話の詳細レコードが処理された場合など）、Oracle TimesTen からバックエンド RDBMS へ情報が転送されます。この統合を実現する方法は複数あります。

アプリケーションは Oracle TimesTen とバックエンド・データベース両方に接続でき、正規の API 要求を介してデータを移行します。これは最も柔軟なアプローチですが、アプリケーションに対する透過性が最も低く複雑なプログラミングが要求されます。たとえば、アクティブなすべてのサブスライバをキャッシュする必要があるアプリケーションでは、最初に Oracle TimesTen 内のレコードを要求できます。結果として見つからなかった場合は、バックエンド RDBMS へ接続し同じ要求を繰り返して、結果を Oracle TimesTen へ挿入します。以降の情報へのアクセスは非常に速くなります。データに対して何らかの変更があった場合、アプリケーションにより両方のデータベースでこれを繰り返す必要があります。このメカニズムの問題点は、バックエンド RDBMS のデータに直接変更が加えられると、キャッシュで一貫性の問題が発生することです。

このアプローチの派生的な方法として、パブリッシュおよびサブスクライブ・メッセージ・バスによって Oracle TimesTen とバックエンド RDBMS へ接続し、既存のアプリケーションとは別に、「変更をリスニングしてバスへパブリッシュし、バスからピックアップした変更を複製する」ことを新しいモジュールに記述する方法があります。アプリケーションは、Oracle TimesTen のトランザクション・ログ API (XLA) を使用して、データベースに対して行われた更新の通知を登録、受信できます。大半の RDBMS は、変更の通知に使用できるトリガー機能を備えているか、または XLA に類似したトランザクション・ログの API を提供しています。

## Cache Connect to Oracle

読取り専用のほとんどのキャッシュとは異なり、Cache Connect to Oracle は Oracle データの読取り/書き込みのキャッシュをサポートしているため、Oracle TimesTen と Oracle データベース間で更新を両方向に伝播できます。

Oracle TimesTen のデータベースと Oracle データベースを統合するうえで、より簡単に透過性の高い方法は、ビルトイン接続により Cache Connect to Oracle を追加することです。

読取り専用のほとんどのキャッシュとは異なり、Cache Connect to Oracle は Oracle データの読取り/書き込みのキャッシュをサポートしているため、Oracle TimesTen と Oracle データベース間で更新を両方向に伝播できます。もう 1 つの違いはキャッシュ・グループの概念で、これにより Oracle データベースのすべてまたは一部の表へマッピングする Oracle TimesTen 表のグループが記述されます。キャッシュ・グループの表は、これらの Oracle データベース表のすべてまたは一部の行と列で構成されます。

Cache Connect to Oracle は、Oracle データベースからのデータをキャッシュ内に保持する期間を指定できます。自動で設定できるだけでなく、SQL 文を使用して要求に応じてキャッシュ・グループをロード、リフレッシュ、フラッシュおよびアンロードできます。

Cache Connect to Oracle は Oracle データベースと対話して、キャッシュ・グループの作成、Oracle からのキャッシュ・グループのロード、キャッシュ・グループと Oracle 間での更新の伝播など、同期的なキャッシュ・グループのすべての処理を実行します。また、Oracle エージェントと呼ばれるバックグラウンド・プロセスがあります。これは、Oracle データベースから Oracle TimesTen 内のキャッシュ・グループへ更新を自動的に伝播するなど、非同期的なすべてのキャッシュ処理を実行します。

Cache Connect to Oracle アプリケーションは、1 つの接続を介してキャッシュ・グループまたは Oracle データベースに SQL 文を送信できます。単一接続の機能は、パススルー機能により実現されます。パススルー機能は、SQL 文がキャッシュされた表によってローカルで処理できるか、または Oracle データベースへリダイレクトする必要があるかをチェックします。キャッシュされたデータは、Oracle TimesTen のキャッシュ・グループまたは Oracle データベースへ更新できます。Cache Connect to Oracle には、キャッシュ・グループから Oracle データベースへ、また Oracle データベースから Oracle TimesTen 内のキャッシュ・グループへ、更新を自動的に伝播する機能があります。

アプリケーションは、1 つの接続を介してキャッシュ・グループまたは Oracle のいずれにも SQL 文を送信できます。単一接続の機能は、パススルー機能により実現されます。パススルー機能は、SQL 文がキャッシュ表によってローカルで処理できるか、または Oracle へリダイレクトする必要があるかをチェックします。

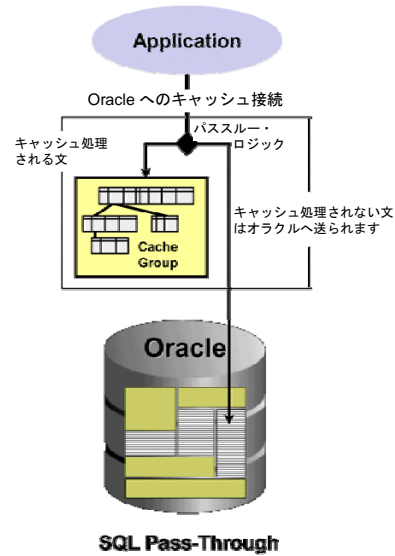


図 9: Cache Connect to Oracle の単一接続機能

キャッシュ・グループには、2 つの基本的なカテゴリがあります。

- **システム管理**のキャッシュ・グループ。Cache Connect to Oracle によって全体的に管理される決定前のキャッシング動作を提示します。
- **ユーザー管理**のキャッシュ・グループ。ユーザーはこれを使用して、すべての属性と SQL 文から選択してカスタマイズされたキャッシング動作を定義できます。ユーザーは、ロード、アサイン、伝播のメカニズムを完全に制御します。

システム管理のキャッシュ・グループには、次の 2 つの基本的なタイプがあります。

- **Writethrough** キャッシュ・グループ。このキャッシュ・グループ作成後は、キャッシュされた表のデータを Oracle データベースからロードします。それ以降は、キャッシュ・グループに対するすべての更新は、Oracle に自動的に伝播されます。Writethrough キャッシュ・グループは、非同期 (AWT) または同期 (SWT) に設定できます。SWT のキャッシュ・グループは、キャッシュのコミット前に Oracle データベースでコミットが実行されるのを待機します。AWT のキャッシュ・グループは Oracle データベースでのコミットを待たずに、キャッシュでの変更をコミットします。
- **Readonly** キャッシュ・グループは、読取り専用のデータを保持しているため、キャッシュ・グループ内で表を更新することはできません。更新は Oracle データベースで実行する必要があります。これらの更新は、パススルー機能を使用して Oracle データベースへリダイレクトすることにより、Oracle データベースで直接または Oracle TimesTen を介して実行できます。デフォルトでは、Oracle データベースで更新が発生した場合、READONLY のキャッシュ・グループが自動的にリフレッシュされます。リフレッシュの頻度は、アプリケーションにより制御されます。

キャッシュ・グループの動作は事前定義できます。動作には、Cache Connect to Oracle により定義される「システム管理」と、高い柔軟性が実現できる「ユーザー管理」があります。システム管理のキャッシュ・グループは、事前定義された Readonly または Writethrough の構成をインストールします。

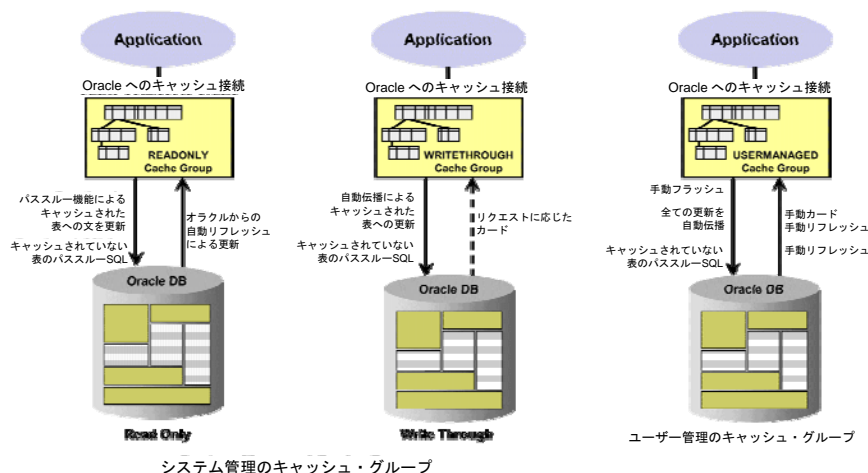


図 10: システム管理およびユーザー管理のキャッシュ・グループ

## イベント処理

イベント処理は、ビジネスに関連すると思われるイベントを「検知および応答」する機能として定義されます。

リアルタイム・エンタープライズ (RTE) に関するGartnerの最新の調査<sup>1</sup>によると、「RTEは、根本原因および成功にとって重要である明白なイベントについて、イベントが発生する瞬間を監視、取得、分析し、新しい機会を特定し、事故を回避して、コア・ビジネス・プロセスの遅延を最少にします。RTEはこの情報を利用して、クリティカルなビジネス・プロセスの管理と実行における遅延を随時解消します。」と報告されています。

Oracle TimesTen 製品でイベント処理アプリケーションを実現する方法は複数あります。最も基本的な方法では、ネットワークの終端に位置するアプリケーションをサポートする設計がされています。ネットワークの終端では、メッセージおよびビジネス・イベントが発生し、最初に検出されます。様々なコンピューティング・サーバー上でアプリケーションと共存する機能によって、イベント処理アプリケーションを柔軟にサポートできます。

ただし、簡単なイベントだけでは、その妥当性が明白にならないことがよくあります。また、このようなイベントは、イベントのパターンが明白になるまで他の情報に関連付けられるか待機されます。このような場合、他のデータ・ソースから呼び出した参照データと比較して、イベントのデータをデータ・マネージャで取得する必要があります。アクションが必要なイベントが特定された場合、ただちに適切なレスポンスを実行する必要があります。このような処理は、概念的には Oracle TimesTen 製品の機能に適しています。

Oracle TimesTen のトランザクション・ログ API (XLA) では、データベースに対する更新を検出できます。アプリケーションは XLA を使用して、Oracle TimesTen データベースの変更を監視し、これらの変更に基づいてアクションを実行します。たとえば、次のような場合には通知の実行が必要です。

<sup>1</sup> Gartnerによる、リアルタイム・エンタープライズの定義の更新、K. McGee氏、2004年3月25日

- 顧客の1日あたりの取引金額が1,000,000ドルを超えた場合
- 「A」リストの顧客が購入する場合
- 前払いの残高を使い切った場合
- ネットワークの要素がオフラインの場合
- フライトの出航ゲートが変更になった場合

複数のアプリケーションで、同時にトランザクション・ログの更新を読取ることができます。各アプリケーションは、そのポジションを保持するために自身のブックマークをログ・フィルで管理できます。ブックマークはデータベースの接続、停止およびシステム障害において永続的であるため、イベントの通知はそのイベントが停止した場所から回復できます。

XLA を使用して、Oracle TimesTen 以外のデータベースに対してカスタム・データのレプリケーション・ソリューションを作成することがよくあります。

Oracle TimesTen のイベント処理機能は、トリガーとストアド・プロシージャの使用により従来の RDBMS で実現できます。ただし、XLA はオーバーヘッドを減少してパフォーマンスを向上させ、リアルタイムで期待する内容に一致させることを目的として設計されています。また、XLA のイベントをマテリアライズド・ビューと併用することにより、データベースの細かいレベルのサブセットまでターゲットにすることができます。従来のデータベースのトリガーは、表内のレコードで変更が行われるたびに自身のロジックを実行していました。

マテリアライズド・ビューは、XLA 機能と組み合わせて、きめ細かいイベント通知を実現する効果的な方法を提供します。つまり、マテリアライズド・ビューは、実行が必要なアクションについて特定のイベントに関連するデータを一箇所に収集します。XLA アプリケーションに必要なことは、1つのマテリアライズド・ビュー表から得られる更新レコードの監視のみです。マテリアライズド・ビューがなければ、XLA アプリケーションは、アプリケーションに関係ない行と列に対する更新を反映するレコードも含め、すべてのベース表のすべての更新レコードを監視しなければなりません。

## 結論

ビジネス・ネットワーク間を移動するメッセージが高速化し、キー・イベントの取得、分析ならびにレスポンスをインテリジェントに行うリアルタイム処理の使用が、すぐれた企業のベンチマーク（判断基準）になろうとしています。これは、クリティカルなビジネス・プロセスの実行と管理でのみ重要なわけではありません。顧客は、きめ細かく作成された対話処理に加え、取引を行う企業から最大の対応を期待しています。

必要なのは、使い慣れた強力なインタフェースと汎用な問合せ言語を備えた軽量のインフラストラクチャ・ソフトウェアです。このようなソフトウェアは、既存のバックオフィス・データベース、メッセージング・システム、アプリケーション・サーバーと簡単にインタフェースで接続でき、今日のネットワーク化された豊富なメモリーを持ったコンピューティング・プラットフォームをフル活用しま

す。これこそ、Oracle TimesTen が提供するリアルタイムのデータ管理用のインフラストラクチャ・ソフトウェアです。

Oracle TimesTen の製品は、パフォーマンスが重要なシステムに対してアプリケーション層の管理を提供するもので、迅速なレスポンスと Oracle データのリアルタイム・キャッシュのために最適化されています。Amdocs、Aspect、Avaya、Bombay Stock Exchange、Cisco、Ericsson、JP Morgan、Lucent、NEC、Nokia、Salesforce.com および Sprint など何百社もの世界規模の企業が、本稼動用のアプリケーションに Oracle TimesTen を使用しています。



Oracle TimesTen 製品とテクノロジー  
2007 年 2 月

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

海外からのお問合せ窓口:  
電話: +1.650.506.7000  
ファックス: +1.650.506.7200  
[www.oracle.com](http://www.oracle.com)

Copyright © 2007, Oracle. 無断転載を禁ず。

この文書はあくまで参考資料であり、掲載されている情報は予告なしに変更されることがあります。オラクル社は、本ドキュメントの無謬性を保証しません。また、本ドキュメントは、法律で明示的または暗黙的に記載されているかどうかに関係なく、商品性または特定の目的に対する適合性に関する暗黙の保証や条件を含む一切の保証または条件に制約されません。オラクル社は、本書の内容に関していかなる保証もいたしません。また、本書により、契約上の直接的および間接的義務も発生しません。本書は、事前の書面による承諾を得ることなく、電子的または物理的に、いかなる形式や方法によっても再生または伝送することはできません。

Oracle、JD Edwards、PeopleSoft および Siebel は、Oracle Corporation および関連会社の登録商標です。他の製品名は、それぞれの所有者の商標です。