ORACLE

Oracle Database In-Memory を使用するケース

アプリケーションの加速を目指すユースケースの特定

2022年3月、バージョン2.1

公開

本書の目的

本書では、Database In-Memoryの概要を説明したのち、おおまかなユースケースをいくつか挙げ、パフォーマンス向上を実現するシナリオを説明します。本書の目的は、一般的なガイドラインを提供することで、読者のユースケースがこの優れた新テクノロジーに適しているかどうかを判断できるようにすることのみにあります。

免責事項

本文書には、ソフトウェアや印刷物など、いかなる形式のものも含め、オラクルの独占的な所有物である占有情報が含まれます。この機密文書へのアクセスと使用は、締結および遵守に同意したOracle Software License and Service Agreementの諸条件に従うものとします。本文書と本文書に含まれる情報は、オラクルの事前の書面による同意なしに、公開、複製、再作成、またはオラクルの外部に配布することはできません。本文書は、ライセンス契約の一部ではありません。また、オラクル、オラクルの子会社または関連会社との契約に組み込むことはできません。

本書は情報提供のみを目的としており、記載した製品機能の実装およびアップグレードの計画を支援することのみを意図しています。マテリアルやコード、機能の提供をコミットメント（確約）するものではなく、購買を決定する際の判断材料にするものでもありません。本書に記載されている機能の開発、リリースおよび時期については、オラクルの裁量により決定されます。製品アーキテクチャの性質上、コードが大幅に不安定化するリスクなしに、本書に記載されているすべての機能を安全に含めることができない場合があります。

目次

概要

Oracle Database In-Memoryは、Oracle Databaseのパフォーマンスに先例のない飛躍をもたらし、さまざまなワークロードのパフォーマンスを驚異的に引き上げます。Oracle Database In-Memoryは、分析ワークロードのパフォーマンスを桁違いに向上させるだけでなく、混合ワークロードのエンタープライズOLTPアプリケーションのパフォーマンスも大幅に高めます。本書では、Database In-Memoryについて簡単に説明したのち、おおまかなユースケースをいくつか挙げ、パフォーマンス向上を実現するシナリオを説明します。本書の目的は、一般的なガイドラインを提供することで、読者のユースケースがこの優れた新テクノロジーに適しているかどうかを判断できるようにすることです。

Oracle Database In-Memoryについて

Database In-Memoryは、高度に最適化されたインメモリ列ストア（IM列ストア）を備えており、図1に示すように、既存のバッファ・キャッシュとともに保持されます。IM列ストアの**おもな目的**は、**分析処理**による列指向のデータ・アクセスを高速化することです。従来の（分析用）索引を表内のすべての列に作成するようなものですが、従来の索引よりもずっと軽量で、ロギングやその他のデータベース書込みは必要ありません。従来の索引によるアプリケーション・パフォーマンスの向上は、その索引が付いた表に含まれるデータへのアクセスにどれだけの時間がかかっているかによって異なりますが、IM列ストアによるメリットも、同様に、分析処理用のデータへのアクセスにかかる時間に依存します。このため、Database In-Memoryからメリットを得るかどうかを判断するには、アプリケーションの基本特性を理解しておくことが重要になります。

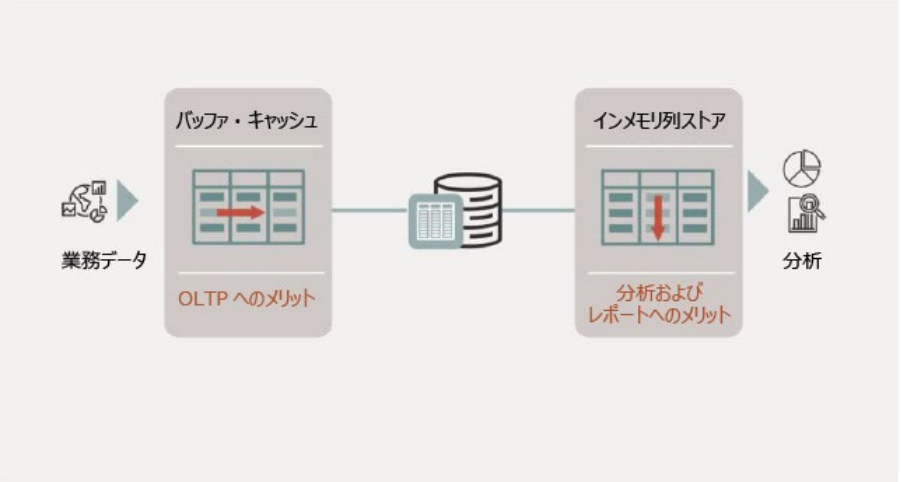

図1：新しいインメモリ列ストアを含むデュアル・フォーマットのインメモリ・データベース

Database In-Memoryによってパフォーマンスが向上する仕組み

IM列ストアには、問合せ処理を高速化するための最適化が複数含まれています。これらの詳しい説明はDatabase In-Memory技術概要に記載されているため、ここでは簡単に述べます。

分析問合せ処理を飛躍的に高速化する列ストアの基本アーキテクチャ要素は4つあります。

1. **圧縮された列形式ストレージ**：圧縮された列ユニットにデータが連続して保存されるため、分析問合せで、必要な列内のデータのみをスキャンできます。対照的に、行メジャー形式の場合は、その他の列に含まれる不要なデータをスキップする必要があります。列形式ストレージにより、問合せで高効率の連続メモリ参照が可能になるとともに、圧縮により、使用可能なシステム（プロセッサからメモリ）の帯域幅使用を最適化できます。

2. **ベクトル処理**：列編成ストレージではデータを連続処理できるだけでなく、ベクトル処理も利用できます。最近のCPUは、ベクトル命令と呼ばれる高度に並列化された命令を備えています。ベクトル命令では、複数の値を1つの命令で処理できます。たとえば、1つの命令内で、複数の値を特定の1つの値と比較することができます（例：State = “California”の売上を探す）。圧縮列形式をベクトル処理することで、列形式ストレージで得られるネイティブのスキャン速度をさらに上げることができ、CPUコアあたり毎秒数百億行を超えるスキャン速度が実現します。

3. **インメモリ・ストレージ索引**：特定の表のIM列ストアは、インメモリ圧縮ユニット（IMCU）と呼ばれる複数のユニットで構成されています。IMCUには通常、多数の行（約50万行）が格納されます。それぞれのIMCUには、そこに含まれる列内のデータの最小値と最大値に加えて、データに関するその他のサマリー情報が自動的に記録されます。このメタデータは、インメモリ・ストレージ索引の役割を果たします。たとえば、スキャン条件から、そのIMCU内には一致する値が見つからないことが判明している場合、スキャン中にIMCU全体をスキップできます。

4. **In-Memoryによる結合およびレポートの最適化**：スキャン速度が飛躍的に向上した結果、オプティマイザによってBloomフィルタ（以前にOracle Database 10gで導入）が選択されることが多くなります。Bloomフィルタの最適化を使用すると、外部（ディメンション）表のスキャンでコンパクトなBloomフィルタが生成されます。これを使用することで、内部（ファクト）表スキャン時の結合によって処理されるデータ量を大幅に削減できます。さらにDatabase In-Memoryでは、ベクトル処理によってBloomフィルタの評価を桁違いに高速に実行できます。同様に、*VECTOR GROUP BY*と呼ばれる最適化を使用すると、標準的なスター・スキーマに対する複雑な集計問合せを、ディメンション表とファクト表に対する一連のフィルタされたスキャンに変換することができます。

5. **Exadataの独自機能**：Oracle ExadataのDatabase In-Memoryでは、データベース層で使用されるものと同じインメモリ列形式を使用して、ストレージ・サーバーのSmart Flash Cacheで自動的にデータをエンコードできます。Exadataにはインメモリ・フォルト・トレランスと呼ばれる機能もあり、Database In-Memoryの列ストア内のオブジェクトを複製できます。ノード障害の発生時にも、複製されたオブジェクトを存続ノードの列ストア内で使用できるため、分析問合せのパフォーマンスを維持できます。さらに、プライマリ・データベースまたはActive Data Guardスタンバイ・データベースのいずれかがExadata Database Machine上にある場合、Database In-Memoryを使用してActive Data Guardスタンバイ上での問合せレポートを高速化することもできます。Exadata Database In-Memoryに固有のこれら3つの機能に加えて、Exadata独自のハードウェア機能を利用できるため、ExadataはDatabase In-Memoryを実行するための最善のプラットフォームと言えます。

Database In-Memoryでは、問合せの高速化に加えて、**分析用索引の置換機能**により、データベースのDML操作と書込みを高速化することができます。IM列ストアで超高速の分析が可能になるため、分析問合せの高速化だけに使用される従来の索引を削除することができます。コストの大きい索引メンテナンスを回避することで、更新/挿入/削除操作が桁違いに速くなります。前述したように、IM列ストアは完全なインメモリ構造であるため、メンテナンスのオーバーヘッドが非常に低く抑えられます。

以下の表に、Database In-Memoryから最大のメリットを得るためのおもなアプリケーション設計原則をまとめます。どれも目新しいものではありませんが、Database In-Memoryは分析データへのアクセスを飛躍的に高速化するので、その使用時にはこれらの重要性がさらに高くなります。

表1：Database In-Memoryによるメリットを最大化するための一般的なガイドライン

ルール1	**アプリケーションではなくデータベース内でデータを処理する**	合計や平均などの指標を計算する場合、データベースからアプリケーションに行を読み取るのではなく、その計算をデータベースにプッシュする方がはるかに効率的です。Database In-Memoryではデータベース内処理のメリットが非常に大きいため、これが特に当てはまります。
ルール2	**1行ごとではなく、セットで処理する**	データベースの出入りにかかるコストは処理行数によって相殺されるため、このルールはどのような分析ワークロードにも広く該当します。Database In-Memoryオプションでは、1秒あたりに何十億行も処理できるため、呼出しごとに処理する十分な量のデータをデータベースで指定することが重要です。
ルール3	**代表的なオプティマイザ統計を使用する**	インメモリ・アクセス・パスによって桁違いの高速化が実現する場合は特に、実行計画の違いが問合せパフォーマンスに非常に大きく影響する可能性があります。最適な実行計画を確実に使用するため、オラクルが推奨するベスト・プラクティスに従って、典型的な統計情報を収集してください。
ルール4	**可能な限りパラレルSQLを使用する**	Database In-MemoryではI/Oボトルネックが軽減されており、全般的な実行プロファイルはCPUタイムに左右されます。使用できるすべてのCPUコアをインメモリ処理に使用する並列処理は、パフォーマンスを最大化するために欠かせません。Oracle Real Application Clusters環境では、すべての使用可能なCPUコアをクラスタ全体で十分活用するために自動DOPが必要になるので、特にこれが当てはまります。

Database In-Memoryのおおまかなユースケース

下の図2に示すように、Database In-Memoryは、エンタープライズOLTPシステム内とリアルタイム分析用データウェアハウス内の両方で使用できます。

データウェアハウス・システム

データウェアハウスの場合、Database In-Memoryにより、IM列ストアに格納できるデータ（比較的短期のデータを表す表パーティションなど）に関する分析およびレポートのパフォーマンスが大幅に向上します。

- 基盤レイヤーとアクセス・レイヤーにある表では、Database In-Memoryを活用できます。基盤レイヤー内のインメモリ表に対する問合せパフォーマンスが飛躍的に向上するため、索引やその他のサマリー・オブジェクト（事前計算されたキューブなど）をアクセス・レイヤーから削除できる可能性があります。
- Database In-Memoryは特にデータ・マートに適しています。通常、事前計算されたサマリーと集計（キー・パフォーマンス・インディケータなど）は容易にメモリ内に格納できます。

注：データの読取りと書込みが一度だけになりがちなETLやステージング・レイヤーでは、Database In-Memoryは効果的ではありません。

エンタープライズOLTPシステム

エンタープライズOLTPシステム（Siebel、Peoplesoft、JD Edwardsなどのパッケージ化されたERP/CRM/HCMアプリケーションを実行）には通常、OLTPトランザクションと定期的な分析レポートの両方が混在しています。これらのシステムでは、ベースとなるOLTPデータのリアルタイム・レポート処理にDatabase In-Memoryを使用できます。すでに説明したように、IM列ストアはOLTPのDML操作を大幅にスピードアップするため、こういったシステムに含まれる分析用索引を置換できる可能性があります。

分析用索引を削除すると、システム全体にわたる利点が多くもたらされます。たとえば、データベースの合計サイズが小さくなるため、ストレージ要件が低く、バックアップ時間が短くなります。また、分析用索引の削除により、バッファ・キャッシュのヒット率が上がり、全体的なREDOおよびUNDO生成率が下がり、合計データベースI/Oが少なくなります。

注：リアルタイム取引/通信アプリケーションなどの、分析コンポーネントを持たない特化型の純粋なOLTPシステムでは、Database In-Memoryによる効果はありません。こういったシステムでデータをメモリ内に保存できる場合は、Oracle TimesTen In-Memory Databaseの方がより良い選択肢になります。

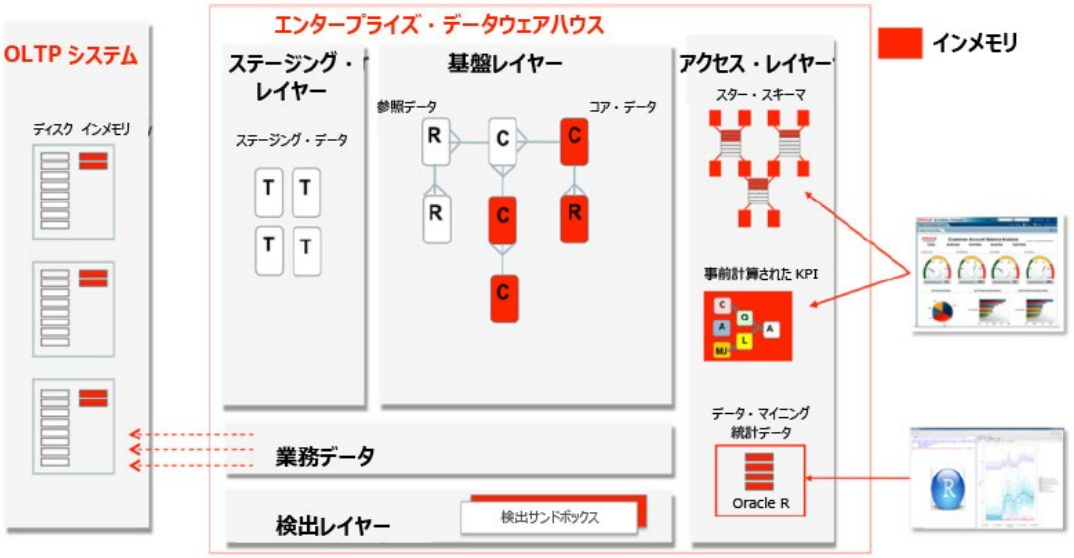

図2：企業内のDatabase In-Memoryユースケース

アプリケーションについて

実際のユースケースがDatabase In-Memoryに大体適合していることがわかったら、Database In-Memoryから得られる全体的なメリットを見積もるためには、アプリケーションのボトルネック（アプリケーション時間の大半が消費されている場所）を理解することが重要です。下の図3に、標準的なアプリケーション時間プロファイルを要約した円グラフを示します。

Database In-Memoryからメリットを得られる領域

上で述べたように、Database In-Memoryがもたらす最適化により、分析問合せを劇的に高速化できます。これにより、以下のワークロード時間要素は、Database In-Memoryから利益を得られる可能性があります（円グラフ内に*マークを付加）。

1. 分析およびレポートのためのデータ・アクセス：これは、Database In-Memoryにとって中心となる価値提案であり、分析用データへのアクセスを桁違いに高速化します。
2. 分析用索引のメンテナンス：Database In-Memoryによって分析用索引を削除できることは少なくありません。これらの索引のメンテナンスが解消されることで、アプリケーション全体のパフォーマンスが向上します。

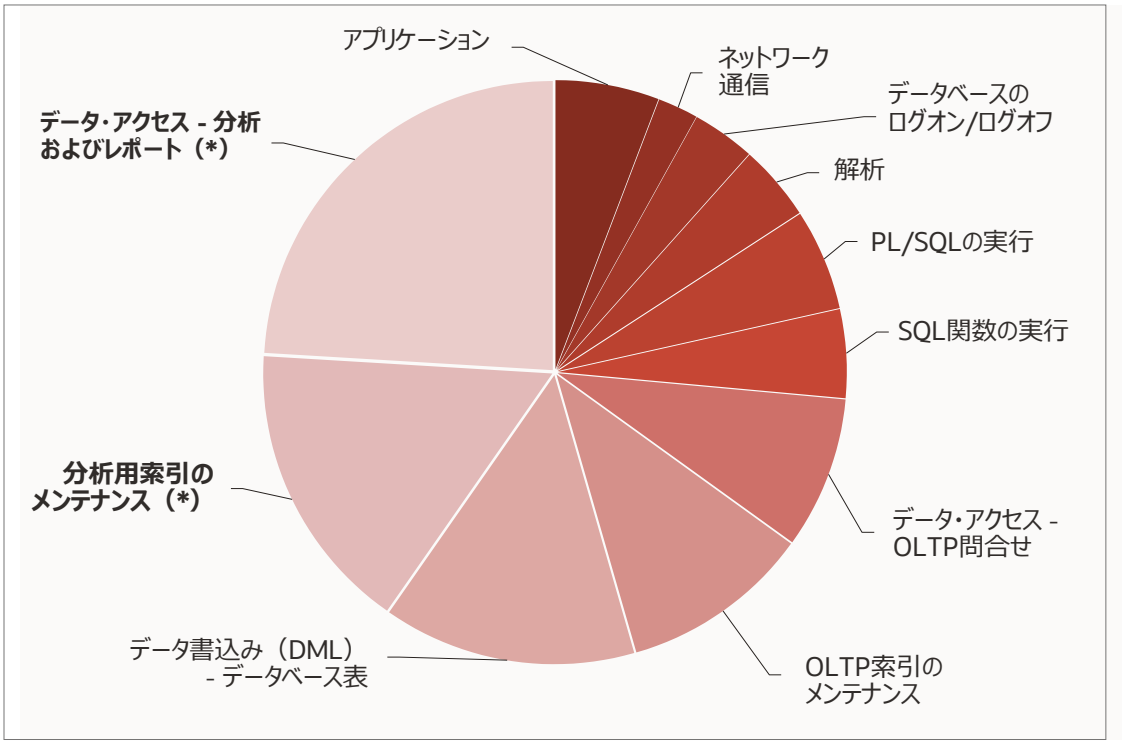

図3：標準的なアプリケーションでの時間プロファイルの要約

Database In-Memoryからメリットを得られない領域

円グラフからわかるように、Database In-Memoryによるメリットのないその他のワークロード要素は数多くあります。これらは分析用データへのアクセスだけでなく、SQL実行のどの側面にも関係していません。このような領域に費やされる時間を最小化するには、Database In-Memoryの登場前に普及していたのと同じOracle Databaseベスト・プラクティスを引き続き適用します。

1. アプリケーション時間：アプリケーション内で費やされる時間は、データベース内を最適化しても変わりません。この時間を最小化するには、（表1で説明したように）データベースでの処理を増やすなど、アプリケーションに特化した最適化を行う必要があります。
2. クライアント/データベース間のネットワーク通信：データベース実行がどれだけ高速になっても、ここには影響しません。通信時間を短縮するための標準テクニックには、可能な限りバッチ実行や配列実行を使用して、データベースのラウンドトリップ・コストを相殺する方法があります。
3. ログオンとログオフ：データベースの接続/再接続にかかるコスト（認証、プロセス作成/破棄コストを含む）は高くなりがちです。この時間はデータ処理とはまったく無関係です。これを短縮するための標準テクニックには、接続をオープンにしたままで可能な限り再利用する方法（接続プールなど）が含まれます。

4. 解析時間：Database In-MemoryはSQL文の実行を大幅に高速化しますが、SQL文の解析と最適化にかかる時間は変わりません。解析時間を短縮するには、バインド変数を使用する、カーソルをオープンのままにする、セッション・カーソル・キャッシュを使用する、などの一般的なベスト・プラクティスを適用する必要があります。

Database In-Memoryによるメリットのないその他の領域は、アプリケーションにとってより基本的な領域であり、アプリケーションの設計を変更せずに対処できる可能性は低いと思われます。

5. *PL/SQL*および*SQL*関数の実行：アプリケーション時間の大半が、PL/SQLプロシージャ、関数、組込み/ユーザー定義SQL関数内で費やされている場合、そのボトルネックは分析用データへのアクセスではなく計算部分にあります。Database In-Memoryでは、多数の問合せ条件式と集計関数（MIN()、MAX()、SUM()など）の評価が高速になるのは事実ですが、一般的には、計算が非常に多いアプリケーションがDatabase In-Memoryから得られるメリットは大きくありません。

6. *OLTP*問合せによるデータ・アクセス：選択性が非常に高い検索（主キーなど）や、単純な主キー/外部キー結合を特徴とするOLTP問合せは、Database In-Memoryからメリットを得られません。ワークロードの大半がこのタイプのデータ・アクセスである場合は、おそらくOracle TimesTen In-Memory Databaseの方が適しています（ただし、アクセスするデータをメモリ内に格納できる場合）。

7. *OLTP*索引のメンテナンス：OLTP問合せを高速化するためだけに存在する索引（主キー/外部キー索引など）や、参照整合性の維持に必要な索引は、Database In-Memoryを使用する場合も引き続き必要です。このため、Database In-Memoryを使用しても、これらの索引のメンテナンスに費やされる時間は変わりません。この場合も、このタイプの索引アクセスが大半となるユースケースでは、Oracle TimesTenの方が適しているでしょう（ただし、アクセスするデータをメモリ内に格納できる場合）。

8. データベース表への書込み：アプリケーション表に対する更新/挿入/削除のDMLにかかる時間は、表がメモリ内にあってもなくても変わりません。書込み処理がきわめて多く、DMLがボトルネックとなっているアプリケーションが、Database In-Memoryからメリットを得る可能性は高くありません。

Database In-Memoryからメリットを得られる問合せの種類

ここまでに、Database In-Memoryのユースケースをおおまかに説明し、アプリケーション実行時間の簡単な内訳を示すことで、Database In-Memoryからメリットを得られる領域を明らかにしました。もう1つ理解する必要があるのは、IM列ストアから最大のメリットを得るのはどのような種類の分析問合せなのかです。すべての問合せに同様に効果がもたらされるわけではありません。

一般的な経験則から言うと、問合せで実際に処理されるデータのうち、問合せによってアクセスされる合計データの割合が大きいほど、*Database In-Memory*による効果が大きい可能性があります。

下の表2では、問合せがDatabase In-Memoryからどれだけメリットを得られるかに影響する各種の問合せ特性を挙げています。それぞれの特性について、指定された問合せ特性だけが異なる2つの問合せを提示しており、一方はDatabase In-Memoryによるメリットが小さく、もう一方はより大きくなります。これは包括的なリストではなく、インメモリ実行から得る利益を見積もる際に、何を基準とするかを理解するためのおおまかな目安として提示されています。

ここでは、オンライン・マーケットプレイスでの売上を表す単純なスター・スキーマがあるものとします。1つのSALES（売上）ファクト表と、STORES（店舗）、PRODUCTS（製品）、SHIPMENTS（発送）、CUSTOMERS（顧客）などの各種ディメンション表で構成されています。

表2：標準的な問合せ特性とDatabase In-Memoryから得るメリットへの影響

問合せ特性	内容	問合せ例	
		メリットは小さい	メリットはより大きい
選択する列の数	問合せで選択される列の数が多いほど、列をまとめるコストが大きくなり、メリットは小さくなります。	SELECT * FROM Sales; 問合せですべての列を選択しているため、メリットは小さい	SELECT revenue FROM Sales; 問合せで*1*列だけを選択しているため、メリットはより大きい
返される値の数	クライアントに返されるデータ量がコストを左右するため、問合せで返される値の数が多いほどIMによるメリットは小さくなります。	SELECT revenue FROM Sales; 問合せによって表内のすべての行の値がアプリケーションに返されるため、メリットは小さい	SELECT SUM(Revenue) FROM Sales; 問合せは表内のすべての行をスキャンするが、*1*つの値だけを返すため、メリットはより大きい
列条件の選択性	列条件の選択性が高いほど、スキャン結果に対するフィルタリングが細かくなり、中間の問合せ計画ノードで処理が必要なデータ量が少なくなります。選択性の高い条件では、インメモリ・ストレージ索引も利用できます。	SELECT MEDIAN(revenue) FROM Sales WHERE revenue > 2; ほとんどの行が対象になり（大半の商品価格は*2*ドル超）、問合せ実行時間の大部分が中央値の計算に費やされるため、メリットは小さい。	SELECT MEDIAN(revenue) FROM Sales WHERE revenue < 2; 対象になる行は少なく、問合せ実行時間のごく一部しか中央値の計算に費やされないため、メリットはより大きい。
結合条件の選択性	上の特性と同様に、結合条件の選択性が高いほど、結合によって処理されるデータ量は少なくなります。Bloomフィルタの最適化が使用されると、パフォーマンスが大幅に向上します。	SELECT S1.id, S.revenue, P.id FROM Sales S, Products P WHERE S.prod_id=P.id; 上の結合では、どの売上にも対応する製品があるため、すべての売上レコードに対して*1*行ずつ返される。その結果、問合せは結合結果の処理に多くの時間を費やすことになる。	SELECT S.id, S.revenue, P.id FROM Sales S, Products P WHERE S.prod_id=P.id AND P.type='Footwear'; この結合では、売上全体のうち、履物製品の売上に相当する行だけが返される。また、上の問合せでは、*Bloom*フィルタの最適化を利用することで、履物製品*ID*のコンパクトなビット・ベクトルを構築し、売上表のスキャン中に適用することができる。
結合される表の数	結合に含まれる表の数が多いほど、結合処理にかかる時間の割合が大きくなります。	SELECT <select list> FROM Sales, Products, Customers, Shipments, Stores, Suppliers, Warehouses WHERE <Join Condition> 上の問合せには*7*つの表の結合が含まれており、合計実行時間のうち結合処理に費やす時間が多くなるため、インメモリ表から得るメリットは小さい。	SELECT <select list> FROM Sales, Products, Customers WHERE <join condition> 上の問合せには*3*つの表の結合が含まれており、合計実行時間のうち結合処理に費やす時間は少ないため、インメモリ表から得るメリットはより大きい。
SQL関数の複雑さ	計算コストの高いSQL関数を含む問合せは、インメモリ表によるメリットが小さくなります。	SELECT I.id, sum(revenue) FROM Sales S, Items I WHERE S.item_id=I.id AND MyMatch(I.name,"LED TV")=1 GROUP BY I.id; 上の問合せは、ユーザー定義関数*MyMatch( )*を使用する条件評価に多くの時間を費やしており、*Database In-Memory*によるメリットは小さい。	SELECT I.id, sum(revenue) FROM Sales S, Items I WHERE S.item_id=I.id AND I.name LIKE "%LED%TV" GROUP BY I.id; この問合せは、文字列マッチングに組込みの*LIKE*演算子を適用することで、条件処理にかかる時間を短くしており、*Database In-Memory*から得るメリットはより大きい。

Oracle Database In-Memoryの情報

リアルタイム・エンタープライズを強化

透過的かつ飛躍的な分析の高速化	Oracle Database In-Memoryは、列のインメモリ・テクノロジーを使用して、業界をリードするOracle Database 12cを透過的に拡張しています。高度に最適化されたインメモリ列形式とSIMDベクトル処理により、CPUコアごとに1秒あたり数十億行の速さで分析を実行できるため、ユーザーは、これまで数時間かかっていたビジネス上の疑問に対する回答を即座に得ることができます。
独自のアーキテクチャで分析をリアルタイムに実行しながら、混合ワークロードのOLTPを加速	分析には列形式が最適ですが、OLTPには行形式が最適です。Oracle Database In-Memoryは両方の形式を同時に使用するため、データウェアハウスとOLTPデータベースの双方でリアルタイムに分析を実行できます。それまで分析に必要であった索引は削除できるため、混合ワークロードのOLTP速度が向上します。
既存のすべてのアプリケーションとの互換性	Oracle Databaseと互換性のあるあらゆる既存アプリケーションとともにOracle Database In-Memoryをデプロイすることは、スイッチを切り替えるように容易であり、アプリケーションを変更する必要もありません。Oracleの幅広い機能、データ型、APIはすべて、引き続き透過的に機能します。
業界をリードするスケールアップ	Oracleの極めて成熟したスケールアップ・テクノロジーにより、最大で数十テラバイトのメモリと数千のCPUスレッドを搭載したSMPコンピュータで、アプリケーションに透過的なインメモリのスケールアップが実現します。データは、1秒あたり数千億行というおびただしい速さで、極めて効率的に分析され、制限される機能もありません。
業界をリードするスケールアウト	Oracleの極めて成熟したスケールアウト・テクノロジーにより、数百テラバイトのメモリと数千のCPUスレッドを搭載した大規模なコンピュータ・クラスタ全体で、アプリケーションに透過的なインメモリのスケールアウトが実現します。データは、1秒あたり数兆行というおびただしい速さで分析され、制限される機能もありません。
業界をリードする高可用性とセキュリティ	Oracleの定評ある高可用性およびセキュリティのテクノロジーはすべて、Oracle Database In-Memoryと透過的に連携するため、ミッション・クリティカルなアプリケーションにおいて極めて高い安全性が確保されます。Oracle Engineered Systemsでは、インメモリ・フォルト・トレランスにより、ノード全体にインメモリ・データが複製されるため、1つのノードに障害が発生しても、問合せではインメモリのデータ・コピーを即座に使用できます。
最大規模のデータベースにおいても優れたコスト効率性を実現	Oracle Database In Memoryでは、すべてのデータをインメモリに格納する必要はありません。頻繁にアクセスされるデータをインメモリに格納し、あまりアクティブでないデータをはるかに低コストのフラッシュやディスクに保存できます。
リアルタイム・エンタープライズを強化	企業は、既存のすべてのアプリケーションで、リアルタイムのデータ分析とリアルタイムのトランザクション処理を容易に同時実行できる能力を持つことで、リアルタイム・エンタープライズへと変革できるようになります。リアルタイム・エンタープライズとは、データに基づく意思決定を素早く行い、顧客の要求に瞬時に対応し、重要なすべての業務プロセスを継続的に最適化できる企業のことです。

Connect with us

+1.800.ORACLE1までご連絡いただくか、**oracle.com**をご覧ください。北米以外の地域では、**oracle.com/contact**で最寄りの営業所をご確認いただけます。

blogs.oracle.com facebook.com/oracle twitter.com/oracle