Oracle® Database
Database In-Memory ガイド
19c

F16109-08(原本部品番号:E96137-10)

2023年7月

タイトルおよび著作権情報

Oracle Database Database In-Memoryガイド, 19c

F16109-08

原本著者: Lance Ashdown

原本協力著者: Frederick Kush、Maria Colgan、Vineet Marwah、Andy Rivenes、Randy Urbano

原本協力者:Yasin Baskan、Nigel Bayliss、Eric Belden、Larry Carpenter、Shasank Chavan、William Endress、Michael Gleeson、Allison Holloway、Katsumi Inoue、Jesse Kamp、Chinmayi Krishnappa、Vasudha Krishnaswamy、Hariharan Lakshmanan、Sue Lee、Teck Hua Lee、Huagang Li、Yunrui Li、Yuehua Liu、Roger Macnicol、Aurosish Mishra、Ajit Mylavarapu、Khoa Nguyen、Jay Patel、Kathy Rich、Beth Roeser、Rich Strohm、Dina Thomas、Qiuhong Wang、Bob Zebian

目次

はじめに

このマニュアルでは、Oracle Database In-Memory機能セットに関連付けられているアーキテクチャおよびタスクを説明します。

この章の内容は次のとおりです。

対象読者

このドキュメントは、インメモリー列ストア(IM列ストア)を管理するデータベース管理者、およびOracle Database In-Memoryの機能を使用する分析問合せを最適化する開発者を対象としています。

ドキュメントのアクセシビリティ

Oracleのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWebサイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracleサポートへのアクセス

サポートを購入したオラクル社のお客様は、My Oracle Supportを介して電子的なサポートにアクセスできます。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info)か、聴覚に障害のあるお客様は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

関連ドキュメント

このマニュアルは、『Oracle Database概要』を十分に理解していることを前提としています。次のガイドを頻繁に参照します。

- Oracle Databaseデータウェアハウス・ガイド
- Oracle Database VLDBおよびパーティショニング・ガイド
- Oracle Database SQLチューニング・ガイド
- Oracle Database SQL言語リファレンス
- Oracle Databaseリファレンス

このマニュアルに記載されている多数の例は、Oracle Databaseで「基本インストール」オプションを選択した場合にデフォルトでインストールされるサンプル・スキーマを使用しています。それらのスキーマの作成方法および使用方法の詳細は、『Oracle Databaseサンプル・スキーマ』を参照してください。

表記規則

このドキュメントでは次の表記規則を使用します。

表記	意味
太字	太字は、操作に関連する Graphical User Interface 要素、または本文中で定義されている用語および用語集に記載されている用語を示します。

表記	意味
イタリック	イタリックは、ユーザーが特定の値を指定するプレースホルダー変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、または入力するテキストを示します。

Oracle Database In-Memoryガイドのこのリリースでの変更点

ここでは、Oracle Database 19c、Oracle Database 18cおよびOracle Database 12cのIn-Memory機能をまとめます。

この項では、次の項目について説明します。

Oracle Databaseリリース19c, バージョン19.20の変更点

Oracle Database 19cバージョン19.20用の『*Oracle Database In-Memory*ガイド』には、新しい`DBMS_INMEMORY_ADVISE` PL/SQLパッケージが含まれています。

`DBMS_INMEMORY_ADVISE`には、データベース・ワークロードに対してデータベース・インメモリーの効果があるかどうかを迅速に評価するために実行できる、`IS_INMEMORY_ELIGIBLE`プロシージャが含まれています。

参照:

unresolvable-reference.html#GUID-D1EDC6A8-4187-4B25-8654-CBC0BA8952D9

Oracle Databaseリリース19c, バージョン19.8の変更点

Oracle Database 19c, バージョン19.8向けの*Oracle Database In-Memory*ガイドには、次のような変更点があります。

新機能

今回のリリースの主な新機能は次のとおりです。

- Database In-Memoryベース・レベル

  Oracle Database release 19c, バージョン19.8以降、`INMEMORY_FORCE`初期化パラメータを`BASE_LEVEL`に設定することで、Database In-Memoryベース・レベルを有効にできます。ベース・レベルでは、Oracle Database In-Memoryオプションを購入することなくインメモリー機能を試すことができます。

  ベース・レベルが有効である場合、IM列ストアのサイズは、Oracle RACデータベースの各データベース・インスタンスおよびCDBまたは非CDBで16 GBに制限されます。また、すべてのオブジェクトおよび列の圧縮レベルが自動的かつ透過的に`QUERY LOW`に設定され、自動インメモリーが無効になります。

- Oracle Database release 19c, バージョン19.8以降、`INMEMORY_FORCE=CELLMEMORY_LEVEL`および`INMEMORY_SIZE=0`を設定することで、IM列ストアを有効にしなくてもCellMemory機能を使用できます。これらの設定では、IM列ストアは有効にならず、問合せでCellMemoryを使用してオブジェクトをスキャンできます。

  以前のリリースのOracle Databaseでは、CellMemory機能を使用するために、IM列ストアを使用する予定がない場合でも、有効にする必要がありました。これにより、何の利点もないIM列ストアの有効化によるオーバーヘッドが発生しました。

関連項目:

- データベースに対するIM列ストアの有効化を参照してください。
- 様々なエディションとサービスでサポートされる機能の詳細は、『Oracle Databaseライセンス情報ユーザー・マニュアル』を参照

Oracle Databaseリリース19c, バージョン19.1の変更点

Oracle Databaseリリース19c, バージョン19.1向け*Oracle Database In-Memory*ガイドには、次のような変更点があります。

新機能

今回のリリースの主な新機能は次のとおりです。

- Database In-Memoryベース・レベル

  Database In-Memoryベース・レベルを有効にするには、INMEMORY_FORCE初期化パラメータにBASE_LEVELを設定します。ベース・レベルでは、Oracle Database In-Memoryオプションを購入することなくインメモリー機能を試すことができます。

  ベース・レベルが有効である場合、IM列ストアのサイズは、Oracle RACデータベースの各データベース・インスタンスおよびCDBまたは非CDBで16 GBに制限されます。また、すべてのオブジェクトおよび列の圧縮レベルが自動的かつ透過的にQUERY LOWに設定され、自動インメモリーが無効になります。

  データベースに対するIM列ストアの有効化およびOracle Databaseライセンス情報ユーザー・マニュアルを参照してください。

- Database In-Memoryの移入待機

  DBMS_INMEMORY_ADMIN.POPULATE_WAIT関数は、優先度が指定された優先度以上のすべてのINMEMORYオブジェクトの移入を開始し、移入のステータス値を返します。ユーザー指定の間隔は、値がコール元に戻されるまで関数が待機する最大時間を指定します。

  「インメモリー・オブジェクトの初期移入の強制」を参照してください。

- インメモリー外部表のビッグ・データおよびパフォーマンスの向上

  このリリースでは、インメモリー外部表機能に対する管理性とパフォーマンスの改善が導入されています。

  - ORACLE_HIVEドライバとORACLE_BIGDATAドライバがサポートされています。
  - パラレル問合せがサポートされています。
  - 全表スキャンでは、インメモリー外部表が移入されます。以前のリリースでは、DBMS_INMEMORYのPOPULATEまたはREPOPULATEプロシージャを使用して移入する必要がありました。
  - In-Memoryのバックグラウンド・プロセス(フォアグラウンド・プロセスではない)で、IMセグメントを削除するようになりました。

  「インメモリー外部表」を参照してください。

- ハイブリッド・パーティション表

  パーティションは、Oracle Databaseセグメント内と、外部ファイルおよびソース内の両方に存在できます。この機能は、

表の大部分が外部パーティションに存在できるBig Data SQLのパーティション化を大幅に拡張します。ハイブリッド・パーティション表の内部パーティションのみがINMEMORY属性を継承します。

「インメモリー表」を参照してください。

- Database In-Memoryに対するOracle Database Resource Managerの自動有効化

  INMEMORY_SIZEが0より大きい場合は、リソース・マネージャが自動的に有効になります。

  「IM動的スキャンのインタフェース」を参照してください。

- Oracle Data Guard Multi-Instance Redo ApplyによるIM列ストアのサポート

  初期化パラメータENABLE_IMC_WITH_MIRAをTRUEに設定すると、IM列ストアとData Guard Multi-Instance Redo ApplyがActive Data Guardスタンバイ・データベースで同時に有効になります。デフォルトでは、ENABLE_IMC_WITH_MIRAはFALSEです。

  ENABLE_IMC_WITH_MIRAについてさらに学習するには、『Oracle Databaseリファレンス』参照してください。

関連項目:

様々なエディションとサービスでサポートされる機能の詳細は、『Oracle Databaseライセンス情報ユーザー・マニュアル』を参照

Oracle Databaseリリース18c, バージョン18.1の変更点

Oracle Databaseリリース18c, バージョン18.1向け*Oracle Database In-Memory*ガイドには、次のような変更点があります。

新機能

今回のリリースの主な新機能は次のとおりです。

- 自動インメモリー

  この機能は、セグメントおよび列の使用状況の統計を使用して、IM列ストアの内容を自動的に管理します。IM列ストアがいっぱいであるために移入ジョブが失敗した場合、自動インメモリーは非アクティブなセグメントを除去して、アクティブなセグメントのためのスペースを作成します。

  「自動インメモリーの構成」を参照してください。

- In-Memory動的スキャン

  IM動的スキャンは、軽量プロセス・スレッドを使用して表スキャンを自動的かつ透過的にパラレル化します。Oracle Resource Managerは、CPUリソースがアイドル状態であり問合せを高速化するために利用できることを認識したときに、これらのスレッドを割り当てます。

  「In-Memory動的スキャン」を参照してください。

- IM式ウィンドウの取得

  任意の長さの式取得ウィンドウを定義できます。これにより、このウィンドウ内で発生する式のみがインメモリーのマテリアライズ用に考慮されます。このメカニズムは、ワークロード全体で代表的な小さい間隔がわかっている場合に特に便利ですたとえば、取引時間帯の間に証券会社が一連の式を収集し、IM列ストアでそれらをマテリアライズして、ワークロード全体の将来の問合せ処理を高速化できます。

「式の取得間隔」を参照してください。

- 外部表のインメモリー・サポート

  外部表をIM列ストアに移入できます。この機能は、内部データと外部データを組み合せた分析問合せに役立ちます。

  「インメモリー外部表」および「DBMS_INMEMORY.POPULATEを使用したインメモリー外部表への移入: 例」を参照してください。

- インメモリー最適化算術

  QUERY LOWで表を圧縮する場合、NUMBER列は、ハードウェアでのネイティブ計算が有効になる最適化された形式を使用してエンコードされます。最適化されたこの形式を使用する集計および算術演算のSIMDベクター処理により、パフォーマンスが大幅に向上します。この機能は、INMEMORY_OPTIMIZED_ARITHMETICをENABLEに設定すると有効になります。

  「インメモリー算術の最適化」を参照してください。

- ラージ・オブジェクト(LOB)のパフォーマンスの向上

  以前のリリースでは、LOBとLOBポインタがIM列ストアに移入されましたが、データベースはバッファ・キャッシュを使用して問合せを満たしました。このリリースでは、範囲述語をスカラー列に適用するインメモリー問合せ、またはSQL演算子をLOB列に適用するインメモリー問合せで、SIMDベクター処理による利益が得られます。

  IM列ストアは、IMCU内の4KB未満のLOBであるインラインLOBに対して連続した記憶域を提供します。アウトラインLOBの場合、IM列ストアは40バイトのLOBロケータのみを格納します。前述のルールには、例外が1つあります。IMEUは、LOBデータ型として定義されたJSON列に対して最大32KBの連続した記憶域を割り当てることができます。IMEUは、これらの列をOSON (バイナリJSON)形式で格納します。

  「CPUアーキテクチャ: SIMDベクター処理」を参照してください。

- 1つの列にあるインメモリー結合グループ

  次の構文を使用して、単一の列に自己結合の結合グループを作成できます。CREATE INMEMORY JOIN GROUP jg_name(table_name(column_name))

  「結合グループによる結合の最適化」を参照してください。

Oracle Database 12cリリース2 (12.2.0.1)での変更点

Oracle Database 12cリリース2 (12.2.0.1)向け*Oracle Database In-Memory*ガイドには、次のような変更点があります。

新機能

今回のリリースの主な新機能は次のとおりです。

- インメモリー列ストア(IM列ストア)の動的サイズ変更

  データベースを再度開くことなく、インメモリー領域のサイズを動的に拡大できるようになりました。

  「IM列ストアのサイズの動的な増加」を参照してください。

- インメモリー式(IM式)

  Oracle Databaseにより、IM列ストアへの移入の候補となる、頻繁に使用される(ホットな)式が自動的に特定され

ます。式の候補としては、(monthly_sales*12)/52が考えられます。IM式は、計算集約的な式を使用し、大規模なデータ・セットにアクセスする分析型問合せのパフォーマンスを大幅に向上させます。

「インメモリー式による問合せの最適化」を参照してください。

- インメモリー仮想列(IM仮想列)

  IM仮想列により、IM列ストアで、表内の一部またはすべての仮想列をマテリアライズできます。

  「インメモリー表に対する列の有効化および無効化」を参照してください。

- IMファスト・スタート

  IMファスト・スタートでは、IMCUをディスクに直接格納することで、IM列ストアへのデータベース・オブジェクトの移入が最適化されます。

  「IM列ストアのIMファスト・スタートの管理」を参照してください。

- サービスに対するオブジェクトレベルのサポート

  個々のオブジェクトの場合、INMEMORY ... DISTRIBUTE句には、このサービスを実行できるデータベース・インスタンスへの移入を制限するFOR SERVICE副句があります。たとえば、INMEMORYオブジェクトをインスタンス1のみ、インスタンス2のみ、または両方のインスタンス内のIM列ストアに移入するよう構成できます。

  「オブジェクトレベルのサービス制御」を参照してください。

- スタンバイ・データベース上のIM列ストア

  Oracle Active Data Guardスタンバイ・データベース上でIM列ストアを有効にできます。アプリケーションで使用できるインメモリー列ストアのサイズを効率的に2倍にして、プライマリおよびスタンバイ・データベース上で、インメモリー列ストア内に完全に異なるデータのセットを移入できます。

  「Oracle Active Data GuardでのIM列ストアのデプロイ」を参照してください。

- IM列ストアでのADOのサポート

  自動データ最適化(ADO)ポリシーを使用して、表、パーティションまたはサブパーティションなどのオブジェクトを、ヒート・マップ統計に基づいてIM列ストアから除去できます。正常なポリシー完了により、指定されたオブジェクトに対してNO INMEMORYが設定されます。

  「IM列ストアに対するADOの有効化」を参照してください。

- 結合グループ

  結合グループは、効果的に結合できる2つの列をリストするユーザー作成オブジェクトです。特定の問合せで、結合グループにより、データベースで列値を解凍およびハッシュするパフォーマンス上のオーバーヘッドを排除できます。結合グループには、IM列ストアが必要となります。

  「結合グループによる結合の最適化」を参照してください。

第I部 Oracle Database In-Memoryの概念

この部では、Oracle Database In-Memory (Database In-Memory)の機能セットを紹介し、インメモリー列ストア(IM列ストア)の基本アーキテクチャを説明します。

1 Oracle Database In-Memoryの概要

Oracle Database In-Memory (Database In-Memory)は、リアルタイム分析と混合ワークロードのパフォーマンスを大幅に改善する一連の機能です。インメモリー列ストア(IM列ストア)は、Database In-Memoryの主要機能です。

ノート:

Database In-Memory の機能には、Oracle Database In-Memory オプションが必要となります。Database In-Memory ベース・レベルの場合、IM 列ストアのサイズは CDB レベルで 16 GB に制限されます。異なるエディションとサービスでサポートされる機能の詳細は、『Oracle Database ライセンス情報ユーザー・マニュアル』を参照してください。

1.1 分析アプリケーションの課題

従来、分析問合せでの優れたパフォーマンスの獲得は、いくつかの要件を満たすことを意味していました。

一般的なデータ・ウェアハウスまたは多目的データベースでは、次のような要件があります。

- ユーザー・アクセス・パターンを理解する必要があります。
- 優れたパフォーマンスを提供する必要があります。それには、一般に、索引、マテリアライズド・ビューおよびOLAPキューブの作成が必要となります。

たとえば、OLTPアプリケーションに対して優れたパフォーマンスを提供するために、表に1つから3つまでの索引(主キーが1つと外部キー索引が2つ)を作成する場合は、分析問合せに対して優れたパフォーマンスを提供するために索引の追加作成が必要になる場合があります。

図1-1 複数の索引

前述の要件を満たすと、管理性およびパフォーマンスに関する問題が生まれます。追加のアクセス構造は、それらを作成、管理および調整する必要があることから、パフォーマンスのオーバーヘッドをもたらします。たとえば、単一の行を表に挿入すると、この表内のすべての索引を更新する必要が生じ、応答時間が長くなります。

リアルタイム分析に対する需要は、より多くの分析問合せが混合ワークロード・データベースで実行されていることを意味します。従来のアプローチでは持ちこたえることはできません。

1.2 単一形式アプローチ

従来、リレーショナル・データベースには、行形式か列形式のどちらかでデータが格納されます。メモリーおよびディスクには、同じ形式でデータが格納されます。

Oracleデータベースでは、行はデータ・ブロック内に隣接して格納されます。たとえば、3つの行がある表の場合、Oracleデータ・ブロックには、まず1つ目の行、次に2つ目の行、次に3つ目の行が格納されます。各行には、その行のすべての列値が含まれます。行形式で格納されたデータは、トランザクション処理のために最適化されます。たとえば、少数の行ですべての列を更新すると、少数のブロックのみが変更されます。

分析問合せに関する問題に対処するために、一部のデータベース・ベンダーでは、列形式が採用されています。列データベースには、行ではなく、選択された列が隣接して格納されます。たとえば、大きなsales表では、ある列に販売IDが存在し、別の列に販売地域が存在します。

分析ワークロードでは、スキャン中に少数の列にしかアクセスしませんが、データ・セット全体をスキャンします。このため、列形式は、分析には最も効率的となります。列は別々に格納されるため、分析問合せで、必要な列のみにアクセスでき、不要なデータを読み取らずにすみます。たとえば、地域別販売額に関するレポートで、少数の列のみにアクセスしながら、多数の行を迅速に処理できます。

データベース・ベンダーは、一般に、顧客に列形式と行ベース形式のどちらかを選択することを強います。たとえば、データ形式が列の場合、データベースでは、メモリー内とディスク上の両方で列形式でデータが格納されます。一方の形式の利点を得ることは、もう一方の形式の利点を失うことを意味します。アプリケーションでは、迅速な分析または迅速なトランザクションのどちらかが実現されますが、両方ではありません。多目的データベースのパフォーマンス問題は、単一形式でのデータの格納では解決されません。

1.3 Oracle Database In-Memoryソリューション

Oracle Database In-Memory (Database In-Memory)機能セットには、インメモリー列ストア(IM列ストア)、高度な問合せ最適化、および可用性のソリューションが含まれています。

Database In-Memoryの最適化により、データ・ウェアハウスおよび多目的データベース上で、分析問合せを桁違いの速さで実行できるようになります。

1.3.1 Database In-Memoryとは

Database In-Memory機能セットには、IM列ストア、高度な問合せ最適化、および可用性のソリューションが含まれています。

Database In-Memory機能を組み合せることで、OLTPのパフォーマンスまたは可用性を犠牲にすることなく、分析問合せが桁違いに高速化されます。

関連項目:

Database In-Memoryオプションについて学習するには、Oracle Databaseライセンス情報ユーザー・マニュアルを参照

1.3.1.1 IM列ストア

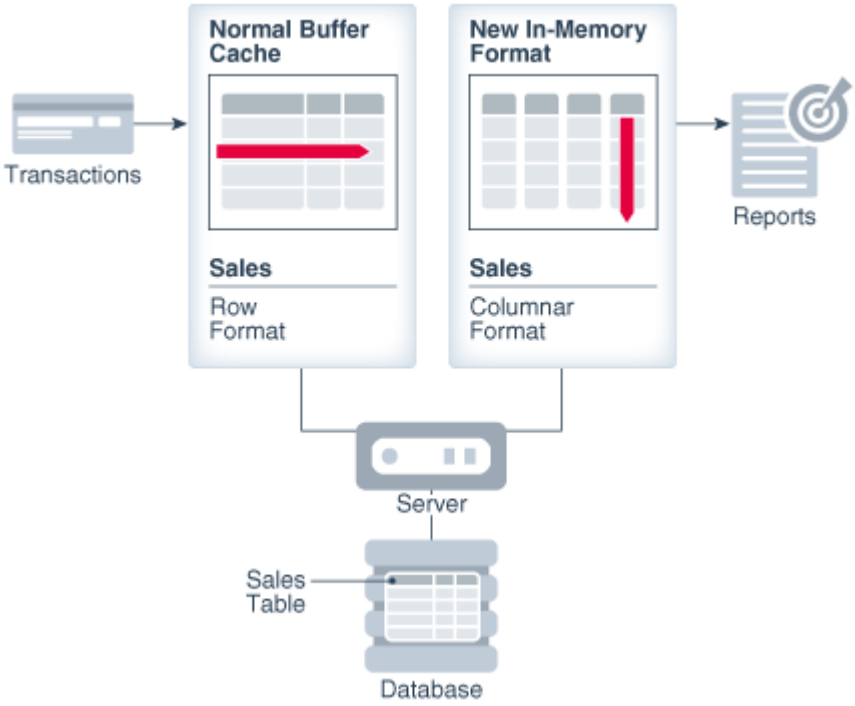
IM列ストアでは、表、パーティションおよび個別の列のコピーが、高速スキャン向けに最適化された圧縮列形式で保持されます。

ビデオ:

IM列ストアは、システム・グローバル領域(SGA)のオプション部分である、インメモリー領域に存在します。IM列ストアは行ベー

ス・ストレージまたはデータベース・バッファ・キャッシュに代わるものではありませんが、それを補完します。データベースでは、データは、行ベースおよび列形式の両方でメモリー内に存在できるようになり、両方の長所が提供されます。IM列ストアにより、ディスク形式とは無関係な、トランザクションの一貫性がある、表データのコピーがさらに提供されます。

図1-2 デュアルフォーマット・データベース



ノート:

IM 列ストアに移入するオブジェクトをバッファ・キャッシュにもロードする必要はありません。

DDL文でINMEMORY句を使用して、次のレベルのいずれかで、IM列ストアを有効にします。

- 列(非仮想または仮想)
- 表(内部または外部)、マテリアライズド・ビューまたはパーティション

ノート:

INMEMORY 属性をハイブリッド・パーティション表に適用する場合、属性は内部パーティションにのみ適用されます。

- 表領域

INMEMORY属性が表領域レベルで指定されている場合、デフォルトで、表領域内のすべての新しい表およびマテリアライズド・ビューがIM列ストアに対して有効になります。Database In-Memoryとの関連では、移入とは、ディスク上の行ベースのデータからIM列ストア内の列データへの自動変換です。IM列ストアへの移入のためにデータベース・オブジェクトの列のすべてまたはサブセットを構成できます。同様に、パーティション化された表またはマテリアライズド・ビューの場合は、パーティションのすべてまたはサブセットを移入のために構成できます。

たとえば、IM列ストアに移入するために、shスキーマからcustomers、productsおよびsalesという3つの表を構成できます。IM列ストアでは、行ではなく列で各表のデータが格納され、各列が別々の行サブセットに分割されます。インメモリー圧縮単位(IMCU)という特別なコンテナに、表セグメント内の行のサブセットのすべての列が格納されます。

関連項目:

- 「インメモリー列ストアのアーキテクチャ」
- 「データベースに対するIM列ストアの有効化」
- INMEMORY句の詳細は、『Oracle Database SQL言語リファレンス』を参照

1.3.1.2 高度な問合せ最適化

Database In-Memoryには、分析問合せのためのいくつかのパフォーマンス最適化が含まれています。

最適化には、次のものがあります。

- 式は、1つ以上の値、演算子、および値を解決するSQL関数(DETERMINISTICのみ)の組合せです。デフォルトでは、インメモリー式(IM式)最適化により、DBMS_INMEMORY_ADMIN.IME_CAPTURE_EXPRESSIONSプロシージャでの、IM列ストア内のホットな式の特定および移入が可能になります。IM式は、非表示の仮想列としてマテリアライズされますが、非仮想列と同じ方法でアクセスされます。
- 結合グループとは、表のセットを結合するために頻繁に使用される列のセットを指定するユーザー定義のオブジェクトです。特定の問合せで、結合グループにより、データベースで列値を解凍およびハッシュするパフォーマンス上のオーバーヘッドを排除できます。
- 小さいディメンション表を大きいファクト表に結合する集計問合せの場合、インメモリー集計(IM集計)では、VECTOR GROUP BY操作を使用してパフォーマンスを向上させます。この最適化では、後でではなくファクト表のスキャン中に、データが集計されます。
- IM列ストアでは、再移入とは、IMCU内のデータが著しく変更された後のIMCUの自動更新です。IMCUに失効エントリがあるが失効しきい値に満たない場合、バックグラウンド・プロセスにより、IM列ストアの段階的な再移入である、トリクル再移入が引き起こされることがあります。

関連トピック

- インメモリー問合せの最適化

1.3.1.3 高可用性のサポート

可用性は、要求に応じてアプリケーション、サービスまたは機能がどの程度使用可能であるかを表しています。

Database In-Memoryでは、次の可用性機能がサポートされています。

- インメモリー・ファスト・スタート(IMファスト・スタート)では、データベース・インスタンス再起動時のIM列ストアへのデータ移入時間が短縮されます。IMファスト・スタートは、IM列ストアに現在移入されているデータのコピーを、ディスク上に圧縮された列形式で定期的に保存することによって、移入時間を削減します。
- Oracle Real Application Clusters (Oracle RAC)環境の各ノードには、独自のIM列ストアがあります。完全に異なるオブジェクトを各ノードに移入させることや、大きなオブジェクトをクラスタ内のすべてのIM列ストア間で分散させることが可能です。Engineered Systemsでは、同じオブジェクトを各ノードのIM列ストアに存在させることも可能です。
- Oracle Database 12cリリース2 (12.2)以降、IM列ストアは、Active Data Guard環境内のスタンバイ・データベース上でサポートされています。

関連トピック

- 高可用性とIM列ストア

1.3.2 分析問合せのパフォーマンスの向上

圧縮された列形式によって、高速スキャン、問合せ、結合および集計が可能になります。

1.3.2.1 データ・スキャンのパフォーマンスの向上

列形式では、大量のデータをスキャンするために高速なスループットが提供されます。

IM列ストアを使用すると、データをリアル・タイムで分析して、様々な可能性を探り、反復を実行できます。具体的には、IM列ストアでは、次を実行する問合せのパフォーマンスが大幅に向上します。

- 多数の行をスキャンし、<、>、=および`IN`などの演算子を使用するフィルタを適用する
- 表、または100列のうちの5列にアクセスする問合せなど、多数の列を持つマテリアライズド・ビューから、わずかな列を選択する
- SQL演算子を使用してLOB列を選択する

ビデオ:

列形式は、大部分の数値および短い文字列のデータ型で、固定幅の列を使用します。この最適化によって高速ベクター処理が可能になり、これにより、データベースでは問合せの応答が速くなります。

IM列ストアのスキャンは、次の理由から、行ベースのデータのスキャンよりも高速になります。

- バッファ・キャッシュのオーバーヘッドの排除

  IM列ストアには、純粋なインメモリー列形式でデータが格納されます。データは、データ・ファイル内に存在し続ける(または、REDOを生成する)ことはありません。そのため、データベースにより、ディスクからバッファ・キャッシュへのデータの読取りのオーバーヘッドがなくなります。

- データ・プルーニング

  データベースでは、データの行全体ではなく、問合せに必要な列のみがスキャンされます。また、データベースでは、記憶域索引および内部ディクショナリを使用して、特定の問合せのための必要なIMCUのみが読み取られます。たとえば、問合せで、店舗IDが8未満の店舗のすべての売上高がリクエストされた場合、データベースでは、IMCUプルーニングを使用して、この値を含まないIMCUを除外できます。

- 圧縮

  従来は、圧縮の目的は領域の節約です。IM列ストアでは、圧縮の目的はスキャンの高速化です。データベースは、圧縮された形式に対して`WHERE`句の述部が適用されるアルゴリズムを使用して、自動的に列データを圧縮します。Oracle Databaseでは、適用された圧縮のタイプによっては、データを最初に解凍することなく、その圧縮された形式でスキャンできます。したがって、データベースで、IM列ストア内でスキャンする必要があるデータの量は、データベース・バッファ・キャッシュ内の対応する量より少なくなります。

- ベクター処理

  各CPUコアにより、ローカルのインメモリー列がスキャンされます。データを配列(セット)として処理するために、スキャンでは、SIMDベクター命令が使用されます。たとえば、問合せでは、値を1つずつ読み取るのではなく、単一のCPU命令で一連の値を読み取ることができます。CPUコアによるベクター・スキャンは、行スキャンよりも桁違いに高速です。

たとえば、次の非定型の問合せを実行するとします。

```
SELECT cust_id, time_id, channel_id
FROM   sales
WHERE  prod_id BETWEEN 14 and 29
```

バッファ・キャッシュを使用する際、データベースでは通常、索引をスキャンして製品IDを検索し、ROWIDを使用してディスクからバッファ・キャッシュに行をフェッチしてから、不必要な列値を廃棄します。行形式でバッファ・キャッシュ内のデータをスキャンすると、多くのCPU命令が必要となり、CPU効率が最善ではなくなります。

IM列ストアを使用する場合は、データベースで、ディスク全体ではなく、リクエストされたsales列のみスキャンできます。列形式でデータをスキャンすると、必要な列のみをCPUにパイプライン処理し、効率が向上します。各CPUコアは、SIMDベクター命令を使用してローカル・インメモリー列をスキャンします。

関連トピック

- CPUアーキテクチャ: SIMDベクター処理
- デュアルフォーマット: 列と行
- IM列ストアの構成および移入

1.3.2.2 結合のパフォーマンスの向上

ブルーム・フィルタは、セット内のメンバーシップをテストする、低メモリーのデータ構造です。IM列ストアでは、ブルーム・フィルタを利用して、結合のパフォーマンスを向上させます。

ブルーム・フィルタは、小さなディメンション表上の述語を大規模なファクト表上のフィルタに変換して、結合を高速化します。この最適化は、1つの大きなファクト表で複数ディメンションの結合を実行する場合に有効です。ファクト表のディメンション・キーには、繰返しの値が数多くあります。スキャンのパフォーマンスと繰返し値の最適化により、結合の速度は桁違いに向上します。

関連トピック

- インメモリー結合について

関連項目:

「インメモリー結合について」

1.3.2.3 集計のパフォーマンスの向上

分析の重要な側面は、データの集計によって、パターンと傾向を判断することです。データがIM列ストアに格納されていると、集計と複合的なSQL問合せはより高速に実行されます。

Oracle Databaseでは、一般に、集計にはGROUP BY句が関係します。従来は、データベースではSORTおよびHASH操作が使用されていました。Oracle Database 12cリリース1 (12.1)以降、データベースでは、効率的な配列ベースのインメモリー集計を可能にするために、VECTOR GROUP BY変換を提供していました。

ファクト表のスキャン中に、データベースにより、集計値がインメモリー配列に蓄積され、効率的なアルゴリズムを使用して集計が実行されます。主キーと外部キーの関係に基づく結合が、スター・スキーマとスノーフレーク・スキーマの両方に対して最適化されます。

関連項目:

- 「VECTOR GROUP BYを使用したインメモリー集計の最適化」
- SQL集計についてさらに学習するには、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照

1.3.3 混合ワークロードのパフォーマンスの向上

OLTPアプリケーションには、IM列ストア内のデータへのアクセスによる恩恵はありませんが、デュアルメモリー形式により、間接的にOLTPパフォーマンスを改善できます。

すべてのデータが行に格納されている場合、分析問合せのパフォーマンスの改善には、アクセス構造の作成が必要になります。標準的なアプローチは、分析索引、マテリアライズド・ビューおよびOLAPキューブの作成です。たとえば、ある表で、OLTPアプリケーションのパフォーマンスを改善するために3つの索引(主キー1つと外部キー索引2つ)が必要となり、分析問合せのパフォーマンスを改善するために10個から20個の索引がさらに必要となるとします。この方法では、分析問合せのパフォーマンスを改善できますが、OLTPのパフォーマンスが低下します。表に1行挿入するには、その表のすべての索引を変更する必要があります。索引の数が増えると、挿入速度が低下します。

データをIM列ストアに移入する場合は、分析アクセス構造を使用しないようにすることができます。この方法では、必要な索引、マテリアライズド・ビューおよびOLAPキューブが少なくなるため、記憶域領域および処理のオーバーヘッドが減少します。たとえば、1回の挿入では、11から23個ではなく、1から3個の索引が変更されます。

IM列ストアにより、ビジネス・アプリケーションにおける分析問合せのパフォーマンスを大幅に向上させることができる一方で、索引参照を使用して短いトランザクションを実行するその場かぎりの分析問合せ、およびデータ・ウェアハウスのワークロード、純粋なOLTPデータベースには、利益が少なくなります。次のタイプの問合せの場合、IM列ストアによってパフォーマンスは向上しません。

- 述語が複雑な問合せ
- 多数の列を選択する問合せ
- 多数の行を返す問合せ

関連項目:

物理的なデータ・ウェアハウス設計についてさらに学習するには、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

1.3.4 Exadataフラッシュ・キャッシュのインメモリー・サポート

`INMEMORY`とマークされたすべてのオブジェクトが同時にDRAMメモリーに収まるわけではありません。Oracle Exadata Storage Serverソフトウェアを使用する場合は、Exadataスマート・フラッシュ・キャッシュを補足メモリーとして使用できます。

IM列ストアが有効になっている場合、Exadataスマート・フラッシュ・キャッシュは、自動的にインメモリー列形式にデータを再フォーマットします。以前のExadataリリースでは、ハイブリッド列圧縮データのみがIM列形式のフラッシュ記憶域の対象でした。再フォーマットは、圧縮された表(OLTP圧縮を含む)と圧縮されていない表の両方で行われます。

ノート:

Database In-Memory ベース・レベルが有効である場合は、Exadata スキャンおよび `CELLMEMORY` 表への移入が無効になります。

この形式では、結合や集計など、Database In-Memoryのほとんどのパフォーマンス拡張機能がSmart Scanでサポートさ

れています。また、非圧縮およびOLTP圧縮のデータ・ブロックをIM列形式に再フォーマットすると、必要なフラッシュ・メモリーの量を大幅に削減できます。

Exadataスマート・フラッシュ・キャッシュは、次の段階でデータを変換します。

1. Oracle Exadataは、データがすぐに利用できるように、対象のスキャンから従来の列形式でデータをキャッシュします。この形式は列形式ですが、IM列ストアで使用する形式と同じではありません。
2. バックグラウンドでは、Oracle Exadataはデータをより低い優先度で純粋なIM列ストア形式に再フォーマットします。バックグラウンドの書込みは、メイン・ワークロードへの干渉を防止します。

データベースがOLTPワークロードを実行していない場合、データ・ウェアハウス・ワークロードがフラッシュ・キャッシュの100%を消費する可能性があります。ただし、OLTPワークロードでは、データ・ウェアハウスのワークロードがフラッシュ・キャッシュの50%以下に制限されます。この最適化により、分析スキャンでOLTPワークロードのパフォーマンスが犠牲になることはありません。

デフォルトでは、Exadataスマート・フラッシュ・キャッシュは、レベル`MEMCOMPRESS FOR CAPACITY LOW`を使用してデータを圧縮します。圧縮レベルを変更するか、列形式を完全に無効にするには、`ALTER TABLE ... NO CELLMEMORY`文を使用します。

関連項目:

- 「データベースに対するIM列ストアの有効化」
- 「CPUアーキテクチャ: SIMDベクター処理」
- `CELLMEMORY`属性についてさらに学習するには、Oracle Exadata Database Machineシステム概要を参照してください
- 各種エディションおよびサービスでサポートされる機能の詳細は、『Oracle Databaseライセンス情報ユーザー・マニュアル』を参照

1.3.5 高可用性のサポート

IM列ストアは、Oracle Databaseに十分に組み込まれています。すべての高可用性機能がサポートされています。

列形式によって、Oracleデータベースのディスク・ストレージ形式が変更されることはありません。したがって、バッファ・キャッシュの変更およびREDOロギング機能も同様です。RMAN、Oracle Data GuardおよびOracle ASMなどの機能は、十分にサポートされています。

Oracle Real Application Clusters (Oracle RAC)環境では、デフォルトで、各ノードに固有のIM列ストアがあります。要件に応じて、様々な方法でオブジェクトを移入できます。

- 様々な表がすべてのノードに移入されます。たとえば、あるノード上に`sales`ファクト表があるのに対して、別のノード上には`products`ディメンション表があるなどです。
- 単一の表が様々なノード間で分散されます。たとえば、ハッシュパーティション化された同じ表の様々なパーティションが様々なノード上にある場合や、パーティション化されていない単一の表の様々なROWID範囲が様々なノード上にある場合などがあります。
- 一部のオブジェクトは、すべてのノード上のIM列ストア内に存在します。たとえば、`products`ディメンション表はすべてのノードに移入するが、`sales`ファクト表のパーティションは様々なノードにわたり分散するなどです。

関連項目:

「高可用性とIM列ストア」

1.3.6 導入の容易性

Database In-Memoryは実装が単純であり、アプリケーション変更は必要になりません。

Database In-Memoryの導入の重要側面を次に示します。

- デプロイの容易性

  ユーザーの管理によるデータ移行は必要ありません。データベースでは、ディスク上に行形式でデータが格納され、IM列ストアの移入時に、自動的に行データが列形式に変換されます。

- 既存のアプリケーションとの互換性

  アプリケーションの変更は不要です。オプティマイザにより、自動的に列形式が利用されます。ご使用のアプリケーションで、データベースに接続してSQLを発行する場合、それにはDatabase In-Memoryの機能が役立ちます。

- 完全なSQL互換性

  Database In-Memoryでは、SQLに関して制限はありません。分析問合せには、Oracleの分析機能を使用するか、カスタマイズされたPL/SQLコードを使用するかに関係なく、利点があります。

- 使いやすさ

  複雑な設定は必要ありません。INMEMORY_SIZE初期化パラメータで、IM列ストアで使用するために予約するメモリーの量が指定されます。DDL文におけるINMEMORY句は、オブジェクトまたは列をIM列ストアに移入することを指定します。IM列ストアを構成すると、既存の分析ワークロードと非定型問合せのパフォーマンスがただちに向上します。

関連項目:

- IM列ストアを有効化する方法を学習するには、「IM列ストアの有効化およびサイズ設定」を参照
- INMEMORY_SIZEおよびINMEMORY_FORCE初期化パラメータについて学習するには、『Oracle Databaseリファレンス』を参照

1.4 Database In-Memoryの要件

Oracle Database In-Memoryオプションは、すべてのDatabase In-Memory機能に必要となります。Database In-Memoryベース・レベルは、16 GB以下のIM列ストアに使用できます。

要件を次に示します。

- Database In-Memoryベース・レベルを使用するには、CDBレベルの初期化パラメータ・ファイルで、INMEMORY_FORCE初期化パラメータにBASE_LEVELを設定する必要があります。このパラメータは動的に設定できず、PDBレベルで設定することもできません。BASE_LEVELを設定すると、次のような動作になります。
  - すべてのINMEMORYオブジェクトおよび列の圧縮レベルにQUERY LOWが自動的かつ透過的に使用されます。
  - 自動インメモリーが無効になります。
- ベース・レベルの場合は、IM列ストアのサイズが16 GBを超えないようにする必要があります。

- IM列ストアには、最小で100 MBのメモリーが必要となります。ストア・サイズは、MEMORY_TARGETに含まれています。
- Oracle RACデータベースの場合は、次の追加の要件が適用されます。
  - DUPLICATEおよびDUPLICATE ALLオプションでは、Oracleエンジニアド・システムが必要となります。
  - INMEMORY_FORCE初期化パラメータにBASE_LEVELを設定すると、各データベース・インスタンスのサイズは16 GBに制限されます。

IM列ストアには、特別なハードウェアは必要ありません。

関連項目:

- 「IM列ストアの必須サイズの推定」
- 「Oracle RACでのIM列ストアのデプロイ」
- Database In-Memoryのライセンス関連のすべての情報は、Oracle Databaseライセンス情報ユーザー・マニュアルを参照してください

1.5 Database In-Memoryのための主要な作業

IM列ストアによる利益を得られる問合せの場合、必要なタスクは、IM列ストアのサイズの設定、および移入のためのオブジェクトの指定のみです。問合せの最適化および可用性の機能には、さらに構成が必要となります。

IM列ストアの構成のための主要な作業

次の表には、主要構成タスクが示されます。

表1-1 構成タスク

タスク	ノート	さらに学習するには
サイズを指定することで、IM 列ストアを有効にします。	INMEMORY_SIZE に少なくとも 100 MB を設定します。インメモリー Free Tier のみの場合は、非 CDB または CDB のサイズが 16 GB 以下である必要があります。Oracle RAC データベースでは、各インスタンスのサイズが 16 GB 以下である必要があります。 COMPATIBLE 初期化パラメータは 12.1.0 以上に設定されている必要があります。	「データベースに対する IM 列ストアの有効化」
Database In-Memory ベース・レベルの場合は、追加の構成を実行します。	Database In-Memory ベース・レベルのみの場合、CDB レベルで INMEMORY_FORCE 初期化パラメータに BASE_LEVEL を設定し、	

タスク	ノート	さらに学習するには
	INMEMORY_SIZE を 16 GB 以下にする必要があります。	
インメモリーFree Tier のみの場合は、追加の構成を実行します。	INMEMORY_FORCE 初期化パラメータを FREE に設定する必要があります。	「データベースに対する IM 列ストアの有効化」
IM 列ストアに移入するための表(内部または外部)、列(非仮想列または仮想列)、表領域またはマテリアライズド・ビューを指定します。	INMEMORY 句は、IM 列ストアのオブジェクトを有効にしますが、即座に移入はしません。	「インメモリー移入に対するオブジェクトの有効化」
必要な場合は、自動データ最適化(ADO)ポリシーを作成して、IM 列ストア内のオブジェクトに対して INMEMORY 属性を設定します。	たとえば、ポリシーにより、10 日間アクセスがなかった場合に IM 列ストアから sales 表を除去できます。インメモリーADO 機能では、HEAT_MAP=ON を ON に設定し、INMEMORY_SIZE をゼロ以外の値に設定する必要があります。	「IM 列ストアに対する ADO の有効化」
オプションで、自動インメモリーを構成してコールド・セグメントを除去し、作業データ・セットが常に移入されるようにします。	たとえば、移入ジョブは表パーティションの半分のみロードします。IM 列ストアがほぼ最大容量であるため、自動インメモリーがトリガーされてコールド・セグメントが除去されるためです。新しいパーティションの次の移入ジョブで、完全に移入されます。インメモリー ADO 機能では、AUTOMATIC_INMEMORY_LEVEL を LOW に、または MEDIUM および INMEMORY_SIZE をゼロ以外の値に設定する必要があります。	自動インメモリーの構成

インメモリー問合せの最適化のための主要な作業

インメモリー問合せの最適化は、IM列ストアを機能させるために必要なわけではありません。次の最適化作業はオプションです。

表1-2 問合せ最適化タスク

タスク	ノート	さらに学習するには
DBMS_INMEMORY_ADMIN パッケージを使用することで、IM 列ストア内の	たとえば、IME_CAPTURE_EXPRESSIONS プロ	「INMEMORY_EXPRESSIONS_US

タスク	ノート	さらに学習するには
IM 式の自動検出を管理します。	シージャを呼び出して、データベースでのホットな式を特定できる期間を定義してから、それらを段階的に移入します。INMEMORY_EXPRESSIONS_USAGE 初期化パラメータは、データベースでの移入可能な IM 式のタイプ(静的、動的、またはそれら両方)を制御します。	AGE」
CREATE INMEMORY JOIN GROUP 文を使用して結合グループを定義します。	候補は、ファクト表およびディメンション表を結合する列など、結合述語で頻繁に組み合せられる列です。	「結合グループの作成」
問合せブロックに必要な場合は、VECTOR_TRANSFORM ヒントを指定してインメモリー集計を有効にするか、NO_VECTOR_TRANSFORM で無効にします。	インメモリー集計は、初期化パラメータや DDL で制御できない、自動的に有効になる機能です。	「IM 集計の制御」
初期化パラメータ INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT を設定することで、2 分間にトリクル再移入によって更新される IMCU の数を制限します。	この初期化パラメータを 0 に設定すると、トリクル再移入を無効にできます。	「しきい値ベース再移入とトリクル再移入」

可用性管理のための主要作業

次の表に、主要タスクを示します。

表1-3 可用性タスク

タスク	ノート	さらに学習するには
DBMS_INMEMORY_ADMIN.ENABLE_FASTSTART プロシージャを使用してインメモリー・ファスト・スタート(IM ファスト・スタート)表領域を指定します。	IM ファスト・スタートでは、データベースの再起動時に IM 列ストアへのデータベース・オブジェクトの移入が最適化されます。IM ファスト・スタートは、高速に IM 列ストアの移入を行うために、ディスク上に情報を格納します。	「IM 列ストアに対する IM ファスト・スタートの有効化」
オブジェクトまたは表領域の場合は、DDL 文で INMEMORY を DISTRIBUTE または DUPLICATE キーワードとともに指定し	デフォルトでは、各インメモリー・オブジェクトは、Oracle RAC インスタンス間で分散され、IM 列ストアのシェアード・ナッシ	「Oracle RAC での IM 列ストアのデプロイ」

タスク	ノート	さらに学習するには
て Oracle RAC でのデータの分散を制御します。	ング・アーキテクチャが効率的に使用されます。	
Oracle Data Guard 環境では、プライマリまたはスタンバイ・データベース上で同じ Database In-Memory 初期化パラメータおよび文を使用できます。	たとえば、INMEMORY_SIZE を設定することで、プライマリおよびスタンバイ・データベースの両方で、IM 列ストアを有効にできます。必要な場合は、DDL で INMEMORY DISTRIBUTE FOR SERVICE 句を使用して、プライマリ・データベースおよびスタンバイ・データベース上で IM 列ストアに異なるデータ・セットを移入します。	「インメモリー移入に対するオブジェクトの手動による有効化について」

1.6 IM列ストアのためのツール

IM列ストアまたはその他のDatabase In-Memory機能の管理には、特別なツールやユーティリティは必要ありません。SQL*Plus、SQL DeveloperおよびOracle Enterprise Manager (Enterprise Manager)などの管理ツールが十分にサポートされています。

この項では、特定のデータベース・インメモリー機能でサポートされているツールについて説明します。

1.6.1 インメモリー適格性テスト

インメモリー適格性テストは、データベース・ワークロードについて、データベース・インメモリー機能を使用する効果があるかどうかを判別するのに役立ちます。

多くのワークロードにはデータベース・インメモリーが有効ですが、そうでないワークロードもあります。インメモリー適格性テストは、指定のワークロードに対してデータベース・インメモリー機能の効果があるかどうかを判別し、この機能の使用について、その適格性を評価します。適格性は、ワークロードにおける分析アクティビティの割合で判別されます。データベース・インメモリーを実装する予定の場合は、このツールを使用すると、分析アクティビティが少なくデータベース・インメモリーを使用しても実質的なメリットを見込めない不適格なデータベースを迅速に特定し除外できます。それにより、ワークロードに強力な分析アクティビティがより多く含まれているため効果が大きいデータベースを、データベース・インメモリー導入の対象にできます。ワークロードにおける分析アクティビティの割合が大きければ大きいほど、データベース・インメモリーによって得られる効果が高くなります。

インメモリー適格性テストは、PL/SQLパッケージDBMS_INMEMORY_ADVISE内のIS_INMEMORY_ELIGIBLEプロシージャです。このパッケージはOracle Databaseに組み込まれています。それをダウンロードしインストールする必要はありません。

インメモリー適格性テストの実行は、データベース・インメモリーを有効にすることを検討しているデータベースでインメモリー・アドバイザを実行する前に必要な準備手順です。これによって時間を節約できます。これは、インメモリー・アドバイザ(時間がかかる)を実行する前に、インメモリー適格性テストで迅速な分析を実行して、指定のワークロードにデータベース・インメモリーが適切かどうかを把握しておくことができるためです。データベース・インメモリーの効果を保証するには分析アクティビティのレベルが不十分であるとインメモリー適格性テストで示されたワークロードに対しては、インメモリー・アドバイザの実行をスキップする必要があります。

データベースに対してデータベース・インメモリーを有効にするかどうかを検討している場合は、次の手順を実行します。

1. 候補となるデータベース・ワークロードに対してインメモリー適格性テストを実行して、データベース・インメモリーを効果的に使用できるワークロードとそうでないワークロードを特定します。
2. インメモリー適格性テストで不適格と判別されたワークロードを除き、すべてのワークロードに対してインメモリー・アドバイザを実行します。

関連項目:

- 『Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』では、データベース・インメモリーについて適格性をテストするための`IS_INMEMORY_ELIGIBLE`プロシージャを含む、DBMS_INMEMORY_ADVISEパッケージについて説明しています。
- AWRスナップショット・データのエクスポート方法については、『Databaseパフォーマンス・チューニング・ガイド』のAWRデータのエクスポートを参照してください。
- 「インメモリー・アドバイザ」では、インメモリー・アドバイザをダウンロードして使用する方法について説明します。

1.6.2 インメモリー・アドバイザ

インメモリー・アドバイザは、データベース内の分析処理ワークロードを分析する、ダウンロード可能なPL/SQLパッケージです。

インメモリー・アドバイザでは、SQL計画のカーディナリティ、アクティブ・セッション履歴(ASH)、パラレル問合せの使用、およびその他の統計に基づいて、分析処理が他のデータベース・アクティビティと区別されます。インメモリー・アドバイザでは、統計およびヒューリスティックな圧縮要因に基づいて、IM列ストア内のオブジェクトのサイズが推定されます。

アドバイザでは、次の内容に基づいて、分析処理のパフォーマンスの向上要因が推定されます。

- ユーザーのI/Oの待機、クラスタの転送の待機、およびバッファ・キャッシュのラッチの待機などの待機イベントの推定
- 特定の圧縮タイプに関連する、問合せ処理の利点
- 特定の圧縮タイプに対する、解凍コストの経験則
- SQL計画のカーディナリティ、結果セット内の列の数など

出力は、インメモリー移入による利益を得られる、IM列ストアのサイズ、およびオブジェクトのリストをお薦めするレポートです。アドバイザでは、推奨されたオブジェクトを`INMEMORY`句で変更するSQL*Plusスクリプトも生成されます。

インメモリー・アドバイザは、ストアドPL/SQLパッケージには含まれません。Oracle Supportからパッケージをダウンロードする必要があります。

関連項目:

インメモリー・アドバイザについてさらに学習するには、My Oracle Supportノート1965343.1

1.6.3 IM列ストア用のCloud Controlページ

Enterprise Manager Cloud Control (Cloud Control)では、インメモリー列ストア中央ホーム・ページが用意されています。このページでは、IM列ストアへのダッシュボード・インタフェースが提供されます。

このページを使用して、表、索引、パーティション、表領域などのデータベース・オブジェクトに対するインメモリー・サポートを監視します。オブジェクトに対するインメモリー機能を表示し、インメモリーの使用状況統計を監視できます。特に指定のないかぎり、このマニュアルでは、Database In-Memoryの機能へのコマンドライン・インタフェースについて説明します。

関連トピック

- Cloud ControlでのIM列ストアの使用

関連項目:

「Cloud ControlでのIM列ストアの使用」では、Cloud Controlを使用してIM列ストアを管理する方法について説明します。

1.6.4 Oracle Compression Advisor

Oracle Compression AdvisorではMEMCOMPRESS句を使用してユーザーが実感できる圧縮率を推定します。アドバイザはDBMS_COMPRESSIONインタフェースを使用します。

関連項目:

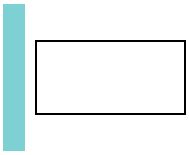
- 「Oracle Compression Advisor」
- DBMS_COMPRESSIONについてさらに学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照

1.6.5 Oracle Data PumpとIM列ストア

impdpコマンドのTRANSFORM=INMEMORY:yオプションを使用して、IM列ストアに対して有効になっているデータベース・オブジェクトをインポートできます。

このオプションを使用すると、Oracle Data Pumpにより、IM列ストア句が設定されたすべてのオブジェクトについて、その句が維持されます。TRANSFORM=INMEMORY:nオプションを指定すると、データ・ポンプは、IM列ストア句が設定されたすべてのオブジェクトからその句を削除します。

また、TRANSFORM=INMEMORY_CLAUSE:stringオプションを使用して、インポート時にダンプ・ファイルのデータベース・オブジェクトに対するIM列ストアを上書きできます。たとえば、このオプションを使用して、インポートされたデータベース・オブジェクトについて、IM列ストアの圧縮を変更できます。



ビデオ:

関連項目:

TRANSFORM impdbパラメータの詳細は、『Oracle Databaseユーティリティ』を参照してください。

2 インメモリー列ストアのアーキテクチャ

インメモリー列ストア(IM列ストア)では、高速スキャン用に最適化された列形式を使用して、表およびパーティションがメモリーに格納されます。Oracle Databaseでは、高機能なアーキテクチャを使用して、列形式と行形式で同時にデータを管理します。

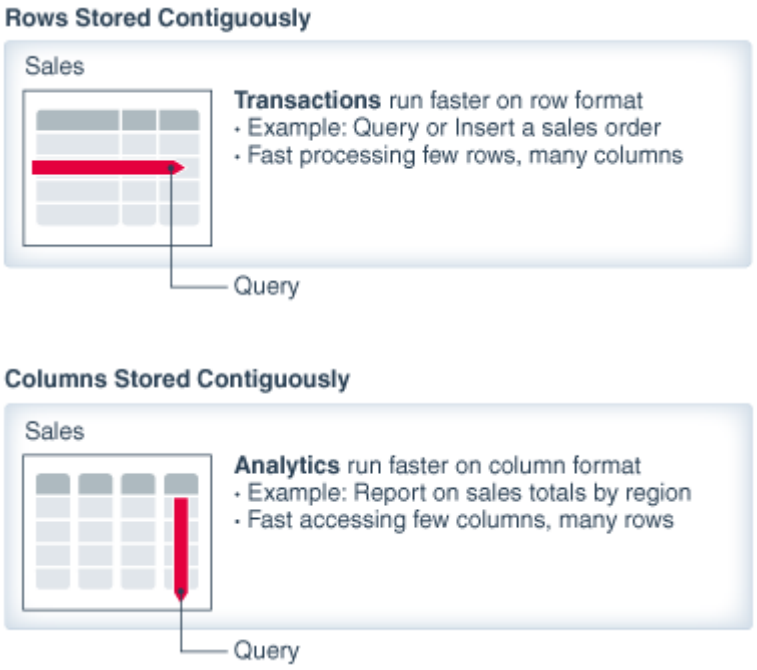
2.1 デュアルフォーマット: 列と行

IM列ストアを有効にした場合は、SGAにより、インメモリー領域とデータベース・バッファ・キャッシュという別々の場所でデータが管理されます。

IM列ストアでは、データが列形式でエンコードされます。各列は独立した構造となります。それらの列は隣接して格納され、分析問合せのために最適化されます。データベース・バッファ・キャッシュでは、IM列ストアにも移入されるオブジェクトを変更できます。ただし、バッファ・キャッシュでは、データが従来の行形式で格納されます。データ・ブロックでは、行はトランザクションのために最適化され、隣接して格納されます。

次の図では、行ベースのストレージと列ストレージとの違いを示します。

図2-1 列および行ベースのストレージ



2.1.1 インメモリー領域内の列データ

インメモリー領域は、IM列ストアが含まれる、オプションのSGAコンポーネントです。

2.1.1.1 インメモリー領域のサイズ

インメモリー領域は、INMEMORY_SIZE初期化パラメータによって制御されます。デフォルトでは、インメモリー領域のサイズは0であり、これは、IM列ストアが無効になっていることを意味します。

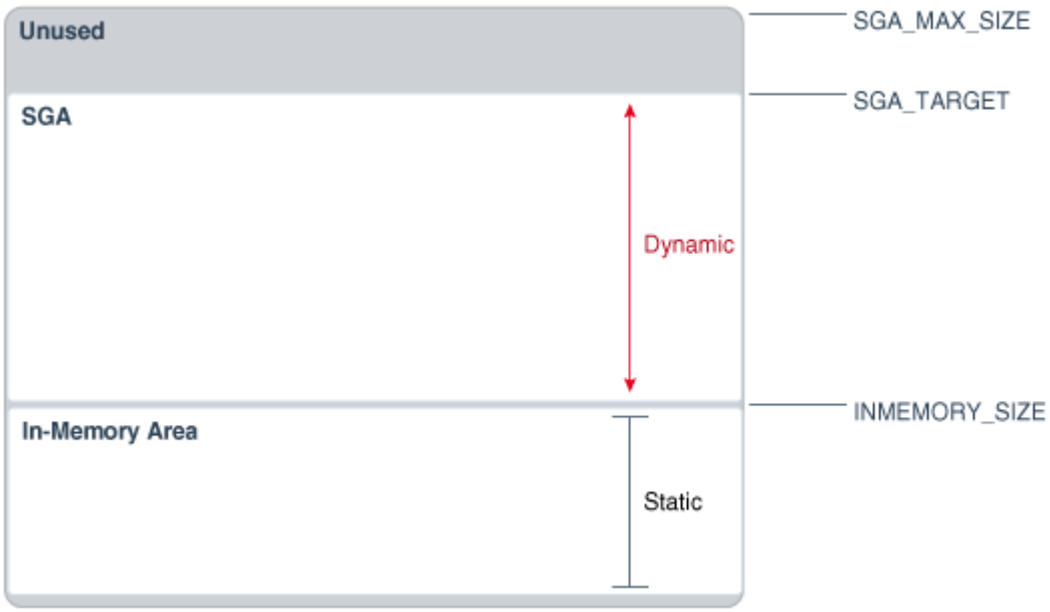
IM列ストアを有効にするには、インメモリー領域を少なくとも100 MBに設定します。サイズは、V$SGAで示されます。

インメモリー領域とSGA_TARGET

インメモリー領域は、SGA_TARGET初期化パラメータの設定から差し引かれます。たとえば、SGA_TARGETを10 GBに設定し、INMEMORY_SIZEを4 GBに設定した場合は、SGA_TARGET設定の40%がインメモリー領域に割り当てられます。次の図

に、この関係を示します。

図2-2 INMEMORY_SIZEとSGA_TARGET



バッファ・キャッシュおよび共有プールなど、SGAのその他のコンポーネントと異なり、インメモリー領域のサイズは、自動メモリー管理によって制御されません。データベースでは、バッファ・キャッシュまたは共有プールでさらにメモリーが必要な場合に自動的にインメモリー領域が縮小されることや、インメモリー領域が領域不足の場合に拡大されることはありません。

インメモリー領域の動的サイズ変更

Oracle Database 12cリリース2 (12.2)以降は、ALTER SYSTEM文を使用することで、INMEMORY_SIZEを動的に拡大できます。次の条件が満たされた場合に、データベースにより、増加したメモリーが割り当てられます。

- SGAに空きメモリーがある。
- INMEMORY_SIZEの新しいサイズが、現在の設定よりも128 MB以上大きい。

ノート:

ALTER SYSTEM を使用して INMEMORY_SIZE を縮小することはできません。

V$INMEMORY_AREAおよびV$SGAビューには、即座にその変更内容が反映されます。

マルチテナント環境のインメモリー領域

CDBでは、IM列ストアのサイズはCDBルートのINMEMORY_SIZEパラメータによって設定されます。デフォルトでは、IM列ストアはPDB間で共有されます。その結果、使用可能なメモリーを消費することによって、PDBが他のPDBを枯渇させる場合があります。

PDB内では、ALTER SYSTEM SET INMEMORY_SIZEを使用してメモリー使用量を制限することができます。たとえばCDBレベルでは、INMEMORY_SIZEを20Gに設定し、PDBを次のように構成します。

- hrpdbで、INMEMORY_SIZEを0に設定します
- salespdbで、INMEMORY_SIZEを10Gに設定します
- oepdbで、INMEMORY_SIZEを11Gに設定します

前述の例では、CDBレベルのINMEMORY_SIZEが20Gのみであっても、PDBレベルのINMEMORY_SIZE設定は21Gに追加

されます。オーバーサブスクリプションは、PDBが停止または切断されている場合に、IM列ストアの貴重な領域が消費されないようにします。

関連項目:

- 「IM列ストアのサイズの動的な増加」
- 自動メモリー管理についてさらに学習するには、『Oracle Database管理者ガイド』を参照
- INMEMORY_SIZE、V$INMEMORY_AREAおよびV$SGAについて学習するには、『*Oracle Database*リファレンス』を参照

2.1.1.2 インメモリー領域内のメモリー・プール

インメモリー領域は、列データおよびメタデータのために複数のサブプールに分割されます。

インメモリー領域は、さらに次のサブプールに分割されます。

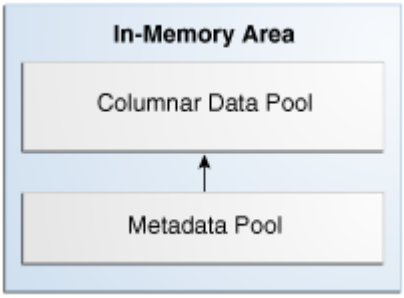
- 列データ・プール

  このサブプールには、列データを含むIMCUが格納されます。例2-1で示すように、`V$INMEMORY_AREA.POOL`列では、このサブプールは`1MB POOL`として識別されます。

- メタデータ・プール

  このサブプールには、IM列ストア内に存在するオブジェクトについてのメタデータが格納されます。例2-1で示すように、`V$INMEMORY_AREA.POOL`列では、このサブプールは`64KB POOL`として識別されます。

図2-3 インメモリー領域内のサブプール



データベースでは、2つのサブプールの相対的なサイズは内部ヒューリスティックを使用して決まります。データベースでは、インメモリー領域内のほとんどの領域が列データ・プール(1 MBプール)に割り当てられます。

ノート:

Oracle Database により、サブプール・サイズが自動的に決定されます。領域割当てを変更することはできません。

例2-1 V$INMEMORY_AREAビュー

この例では、`V$INMEMORY_AREA`ビューを問い合せ、各サブプール内のメモリーの空き容量を測定します(出力例が含まれています)。

```
COL POOL FORMAT a9
COL POPULATE_STATUS FORMAT a15
SSELECT POOL, TRUNC(ALLOC_BYTES/(1024*1024*1024),2) "ALLOC_GB",
```

```
        TRUNC(USED_BYTES/(1024*1024*1024),2) "USED_GB",
        POPULATE_STATUS
FROM    V$INMEMORY_AREA;
POOL      ALLOC_GB   USED_GB    POPULATE_STATUS
--------- ---------- ---------- ---------------
1MB POOL  7.99       0          DONE
64KB POOL 1.98       0          DONE
```

インメモリー領域の現在のサイズはV$SGAで表示可能です。

```
SELECT NAME, VALUE/(1024*1024*1024) "SIZE_IN_GB"
FROM   V$SGA
WHERE  NAME LIKE '%Mem%';
NAME                 SIZE_IN_GB
-------------------- ----------
In-Memory Area       10
```

この例では、サブプールに割り当てられているメモリーは9.97 GBであるのに対して、インメモリー領域のサイズは10 GBです。データベースでは、内部管理構造のためにメモリーのごく一部が使用されます。

関連項目:

V$INMEMORY_AREAについて学習するには、『Oracle Databaseリファレンス』を参照してください。

2.1.2 データベース・バッファ・キャッシュ内の行データ

データベース・バッファ・キャッシュでは、IM列ストアが有効でも無効でも同じようにデータ・ブロックが格納および処理されます。バッファI/Oおよびバッファ・プールは、同じ機能を果たします。

IM列ストアによって、データは従来の行形式(バッファ・キャッシュ)および列形式の両方でSGAに同時に移入されます。データベースは透過的にOLTP問合せ(主キー参照など)をバッファ・キャッシュに送信し、分析問合せおよびレポート問合せをIM列ストアに送信します。データをフェッチする場合は、Oracle Databaseで、同じ問合せ内で両方のメモリー領域からデータを読み取ることもできます。

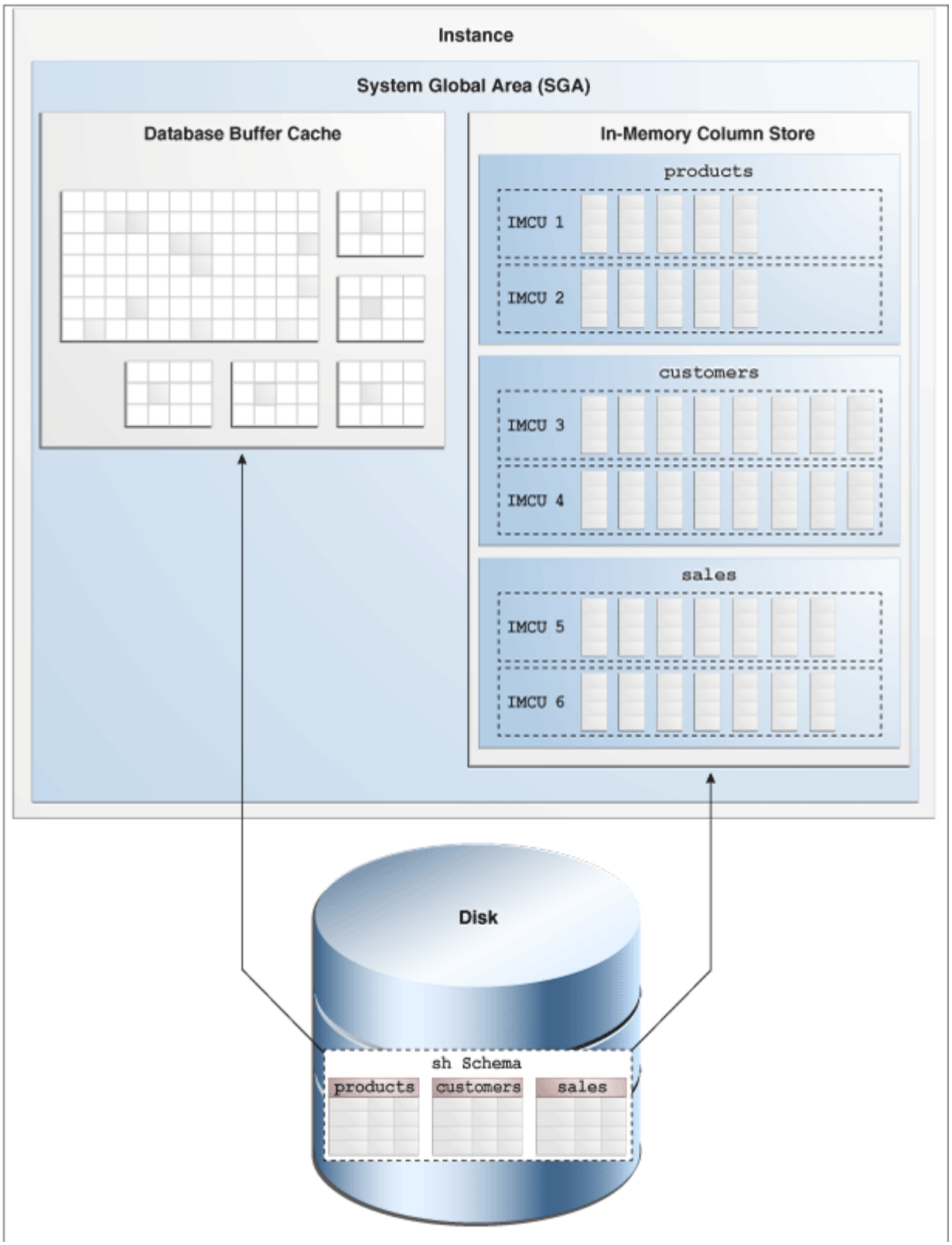
ノート:

実行計画では、操作 TABLE ACCESS IN MEMORY FULL は、IM 列ストア内で一部またはすべてのデータがアクセスされることを示します。

デュアル形式アーキテクチャではメモリー要件は倍増しません。バッファ・キャッシュは、データベースのサイズよりもはるかに小さいサイズで稼働するように最適化されます。

次の図に、IM列ストアのサンプルを示します。データベースで、sh.sales表がディスクに従来の行形式で格納されます。SGAはデータをIM列ストアに列形式で格納し、データベース・バッファ・キャッシュに行形式で格納します。

図2-4 IM列ストア



永続的なヒープ構成表のどのディスク上データ形式もみな、IM列ストアでサポートされます。列形式は、データ・ファイルまたはバッファ・キャッシュに格納されているデータの形式には影響せず、データのUNDO、オンラインREDOロギングにも影響しません。

データベースでは、バッファ・キャッシュ、オンラインREDOログおよびUNDO表領域を更新することで、IM列ストアが有効になっているかどうかにかかわらず、DML変更が同じ方法で処理されます。ただし、データベースは内部メカニズムを使用して変更を追跡し、IM列ストアと残りのデータベースとの一貫性を保ちます。たとえば、sales表がIM列ストアに移入されており、アプリケーションによってsales内の行が更新される場合、データベースでは、自動的にIM列ストア内のsales表のコピーに関してトランザクションの一貫性が保たれます。IM列ストアにアクセスする問合せは常に、バッファ・キャッシュにアクセスする問合せと同じ結果を返します。

関連項目:

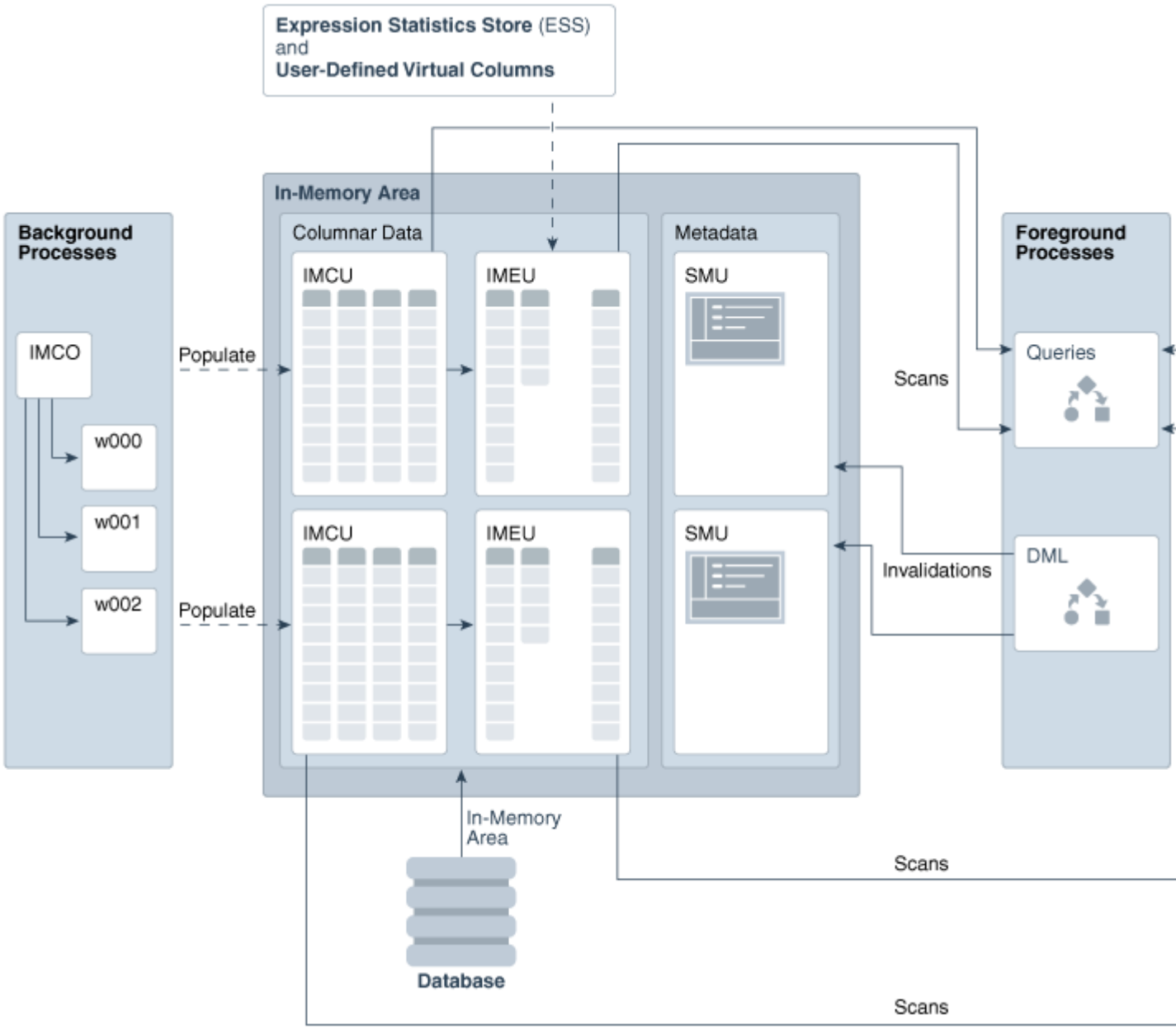
データベース・バッファ・キャッシュについてさらに学習するには、Oracle Database概要を参照してください。

2.2 インメモリー記憶域単位

IM列ストアでは、データとメタデータの両方が、従来のOracleデータ・ブロックではなく、最適化された記憶域単位で管理されます。

Oracle Databaseでは、記憶域単位はインメモリー領域内で保持されます。次の図では、インメモリー領域の概要、およびそれとやり取りするデータベース・プロセスを示します。残りの章では、様々なメモリー・コンポーネントを説明します。

図2-5 IM列ストア: メモリーおよびプロセスのアーキテクチャ



2.2.1 インメモリー圧縮ユニット(IMCU)

インメモリー圧縮単位(IMCU)は、圧縮された、読取り専用の記憶域単位であり、1つ以上の列のためにデータを含みます。

IMCUは、表領域エクステントに似ています。IMCUには、一連の列圧縮単位(CU)、およびIM記憶域索引などのメタデータを含むヘッダーという、2つの部分があります。

2.2.1.1 IMCUとスキーマ・オブジェクト

IM列ストアでは、一連のIMCUに、単一のオブジェクト(表、パーティション、マテリアライズド・ビュー)のデータが格納されます。IMCUには、1つのオブジェクトについてのみ列データが格納されます。

2.2.1.1.1 IMCUとINMEMORY列

INMEMORYとして指定されたオブジェクトの場合は、INMEMORY句でリストされたすべての列がすべてのIMCUに含まれます。

たとえば、sh.sales表に7つの列があるとします。次のDDL文では、その表がINMEMORYとして指定されます。これは、salesのためのすべてのIMCUに、これら7列の列データが含まれることを意味します。

```
ALTER TABLE sh.sales INMEMORY MEMCOMPRESS FOR QUERY LOW;
```

INMEMORYオブジェクトにINMEMORY列がありません

INMEMORY表のすべてではなく一部の列のみにINMEMORY属性を指定できます。たとえば、sh.customers表に23個の列があるとします。次のDDL文では、sh.customersの15列にNO INMEMORY属性があることを指定します。これは、表の他の8列にINMEMORY属性があることを意味します。

```
ALTER TABLE sh.customers INMEMORY
  MEMCOMPRESS FOR QUERY LOW
  NO INMEMORY ( cust_gender, cust_year_of_birth, cust_marital_status,
                cust_postal_code, cust_city, cust_state_province,
                cust_main_phone_number, cust_income_level, cust_credit_limit,
                cust_email, cust_total, cust_total_id, cust_eff_from,
                cust_eff_to, cust_valid );
```

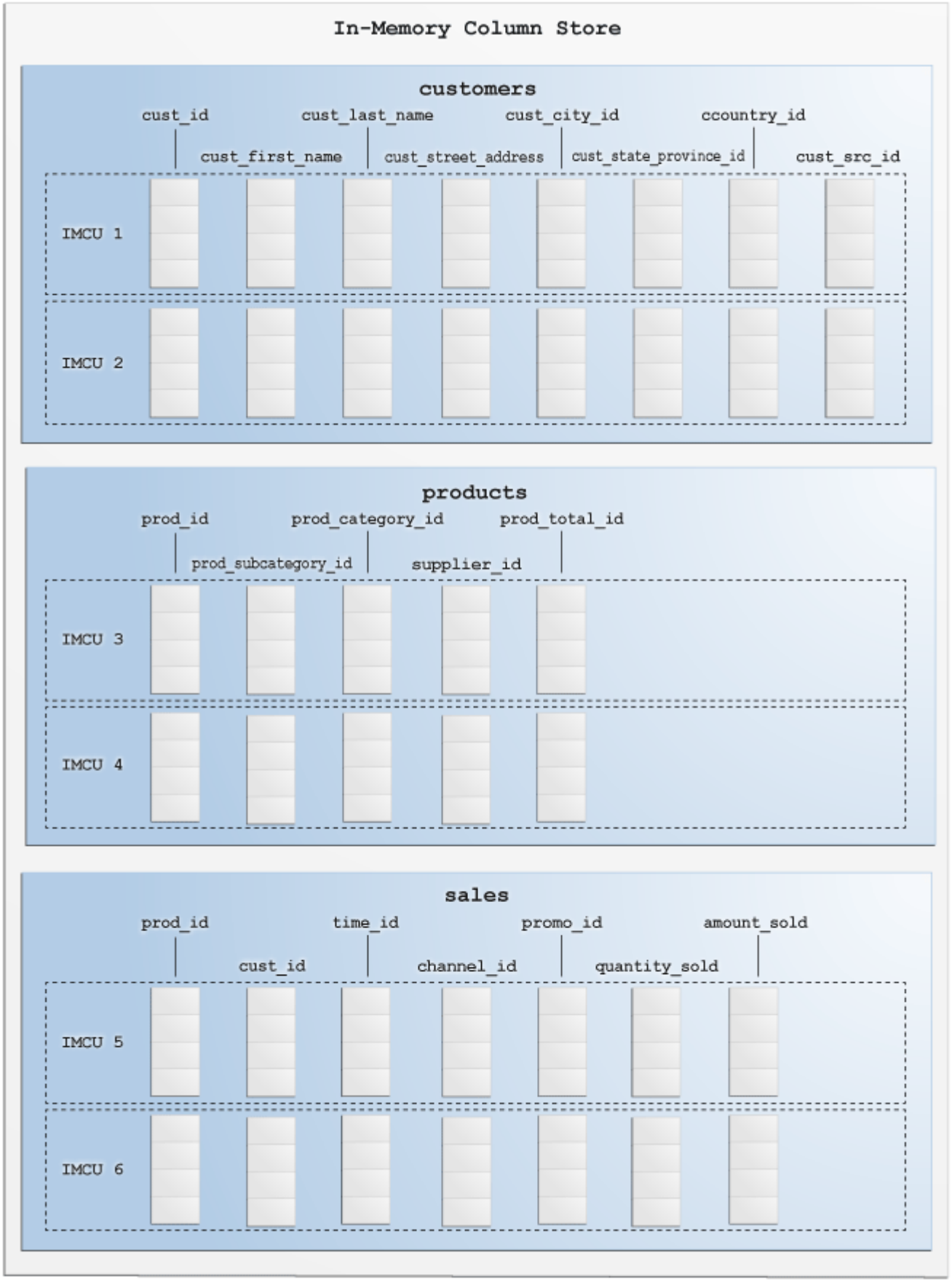
次の問合せでは、sh.customersの列の圧縮レベルが表示され、どの列がNO INMEMORYかが示されます。

```
SET LINESIZE 200
COL TABLE_NAME FORMAT a25
COL SEG_COL_ID FORMAT 999
COL COLUMN_NAME FORMAT a25
COL INMEMORY_COMPRESSION FORMAT a11
SELECT SEGMENT_COLUMN_ID AS "SEG_COL_ID", COLUMN_NAME, INMEMORY_COMPRESSION
FROM   V$IM_COLUMN_LEVEL WHERE TABLE_NAME = 'CUSTOMERS'
ORDER BY SEG_COL_ID;
SEG_COL_ID COLUMN_NAME               INMEMORY_CO
---------- ------------------------- -----------
         1 CUST_ID                   DEFAULT
         2 CUST_FIRST_NAME           DEFAULT
         3 CUST_LAST_NAME            DEFAULT
         4 CUST_GENDER               NO INMEMORY
         5 CUST_YEAR_OF_BIRTH        NO INMEMORY
         6 CUST_MARITAL_STATUS       NO INMEMORY
         7 CUST_STREET_ADDRESS       DEFAULT
         8 CUST_POSTAL_CODE          NO INMEMORY
         9 CUST_CITY                 NO INMEMORY
        10 CUST_CITY_ID              DEFAULT
        11 CUST_STATE_PROVINCE       NO INMEMORY
        12 CUST_STATE_PROVINCE_ID    DEFAULT
        13 COUNTRY_ID                DEFAULT
        14 CUST_MAIN_PHONE_NUMBER    NO INMEMORY
        15 CUST_INCOME_LEVEL         NO INMEMORY
        16 CUST_CREDIT_LIMIT         NO INMEMORY
        17 CUST_EMAIL                NO INMEMORY
        18 CUST_TOTAL                NO INMEMORY
        19 CUST_TOTAL_ID             NO INMEMORY
```

```
        20 CUST_SRC_ID                 DEFAULT
        21 CUST_EFF_FROM               NO INMEMORY
        22 CUST_EFF_TO                 NO INMEMORY
        23 CUST_VALID                  NO INMEMORY
```

次の図では、customers、productsおよびsalesという、IM列ストアに移入されたshスキーマからの3つの表を示します。この例では、各表で、異なる数の列にINMEMORYが指定されています。各表のIMCUには、INMEMORY列のデータのみが含まれています。

図2-6 列とIMCU



NO INMEMORY列を参照する問合せ

問合せがNO INMEMORY列を参照する場合、表スキャンによってIM列ストア内のIMCUではなく、行ストアからデータを取得します。行ストアのアクセスは、問合せで参照される他の列すべてにINMEMORY列が移入されている場合でも発生します。

たとえば、IM列ストアにcustomers表が移入されるとします。cust_idおよびcust_last_name列にはINMEMORYが指

定されていますが、cust_postal_code列はNO INMEMORYと指定されています。次の問合せを発行します。

```
SELECT cust_id, cust_last_name, cust_postal_code
FROM   customers
WHERE  cust_id < 5001
ORDER BY cust_id;
```

この場合、cust_postal_codeが問合せで参照されている唯一のNO INMEMORY列であっても、IM列ストアではなく行ストアにアクセスします。次の問合せでは、述語にcust_postal_codeがあってSELECTリストがなく、行ストアにもアクセスする必要があります。

```
SELECT cust_id, cust_last_name
FROM   customers
WHERE  cust_postal_code = 77501
ORDER BY cust_id;
```

関連項目:

- INMEMORY列の有効化について
- IM列ストアで移入されていない列にアクセスするブログ・エントリの場合は、https://blogs.oracle.com/in-memory/what-happens-if-a-column-is-not-populated
- ALTER TABLE文について学習するには、『Oracle Database SQL言語リファレンス』を参照してください。

2.2.1.1.2 インメモリー圧縮

IM列ストアでは、記憶域の削減よりはむしろアクセス速度の面で最適化された特別な圧縮形式が使用されます。列形式によって問合せを、圧縮された列に対して直接実行できます。

圧縮により、スキャン操作およびフィルタ操作で処理するデータ量を非常に少なくできるため、問合せパフォーマンスが向上します。Oracle Databaseでは結果セットに必要な場合のみ、データが圧縮解除されます。

IM列ストアで適用される圧縮は、ハイブリッド列圧縮と密接に関係しています。どちらの技術も列ベクターを処理します。主な違いは、IM列ストアの列ベクターがSIMDベクター処理の面で最適化されているのに対し、ハイブリッド列圧縮の列ベクターはディスク記憶域の面で最適化されている点です。

オブジェクトをIM列ストアに移入できるようにする場合は、INMEMORY句で、FOR DML、FOR QUERY (LOWまたはHIGH)、FOR CAPACITY (LOWまたはHIGH)またはNONEの圧縮タイプを指定します。

関連項目:

- 「インメモリー・オブジェクトの制御」
- ハイブリッド列圧縮についてさらに学習するには、『Oracle Database概要』を参照

2.2.1.1.3 IMCUと行

各IMCUには、表セグメント内の行のサブセットのすべての列値(nullを含む)が含まれます。行のサブセットは、グラニュルと呼ばれます。

指定されたセグメントのすべてのIMCUには、ほぼ同数の行が含まれます。Oracle Databaseでは、データ型、データ形式および圧縮タイプに応じて、グラニュルのサイズが自動的に判断されます。圧縮レベルが高いと、IMCU内の行数が多くなります。

IMCUとデータベース・ブロックのセットの間には1対多マッピングが存在します。例2-2で示すように、各IMCUには、異なるブロッ

ク・セットの列の値が格納されます。

IMCU内の列はソートされません。Oracle Databaseでは、それらはディスクから読み取られた順に移入されます。

IMCU内の行の数により、IMCUに使用される容量が決定されます。対象行数によって、1 MBプール内で使用可能な隣接する1 MBのエクステントの量を超えてIMCUが拡大する場合、IMCUにより、残りの列CUを保持するための追加のエクステント(ピース)が作成されます。IMCUでは、必ず1 MBずつ領域が割り当てられます。

例2-2 IMCUと行のサブセット

この単純化された例では、customers表のcust_id、cust_first_name、cust_last_nameおよびcust_genderという4列のみにINMEMORY属性があります。5行のみがその表に存在し、2つのデータ・ブロックに格納されています。概念的には、最初のデータ・ブロックにはその行が次のように格納されています。

```
82,Madeline,Li,F;37004,Abel,Embrey,M;1714,Hardy,Gentle,M
```

2つ目のデータ・ブロックには、行が次のように格納されています。

```
100439,Uma,Campbell,F;3047,Lucia,Downey,F
```

IMCU 1には最初のデータ・ブロックのデータが格納されると仮定します。このケースでは、このデータ・ブロック・ストア内のそれら3行のcust_id列値は、CU内で次のように垂直に格納されます。

```
82
37004
1714
```

IMCU 2には、2つ目のデータ・ブロックからデータが格納されます。これら2行のcust_id列値は、CU内で次のように格納されます。

```
100439
3047
```

cust_id値はデータ・ブロック内の各行の最初の値であるため、cust_id列は、IMCU内で先頭の位置にあります。列は必ず同じ位置を使用します。そのため、Oracle Databaseでは、セグメントのIMCUを読み取ることで、行を再構成できます。

関連トピック

- インメモリー・オブジェクトの制御

2.2.1.2 列圧縮単位(CU)

列圧縮単位(CU)は、IMCU内の単一列のための隣接する記憶域です。すべてのIMCUに1つ以上のCUがあります。

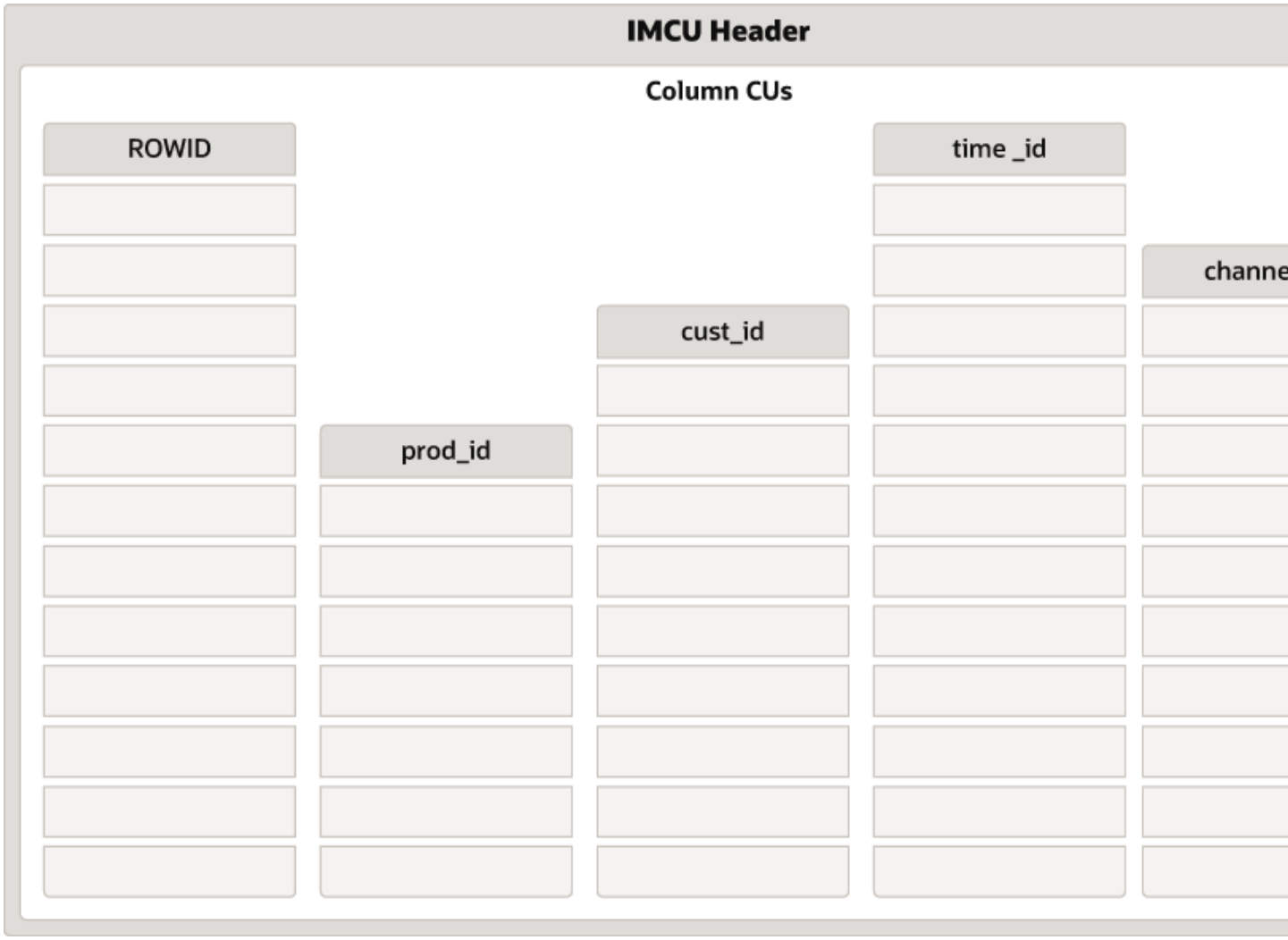
2.2.1.2.1 CUの構造

CUは、本体とヘッダーに分けられます。

すべてのCUの本体には、IMCUに含まれている行の範囲で列値が格納されます。ヘッダーには、CU本体に格納されている値に関してメタデータが含まれます。たとえば、CU内の最小値および最大値です。また、ローカル・ディクショナリが含まれる場合もあります。これは、その列内の非重複値のソート済リスト、およびそれらに対応するディクショナリ・コードです。

次の図では、sales表のprod_id、cust_id、time_idおよびchannel_idという4つのCUがあるIMCUを示します。各CUには、IMCUに含まれている行の範囲で列値が格納されます。

図2-7 IMCU内のCU



CUには、ROWIDの順序で値が格納されます。このため、データベースは、行を元どおりにまとめることで問合せに答えることができます。たとえば、アプリケーションによって次のような問合せが発行されるとします。

```
SELECT cust_id, time_id, channel_id
FROM   sales
WHERE  prod_id =5;
```

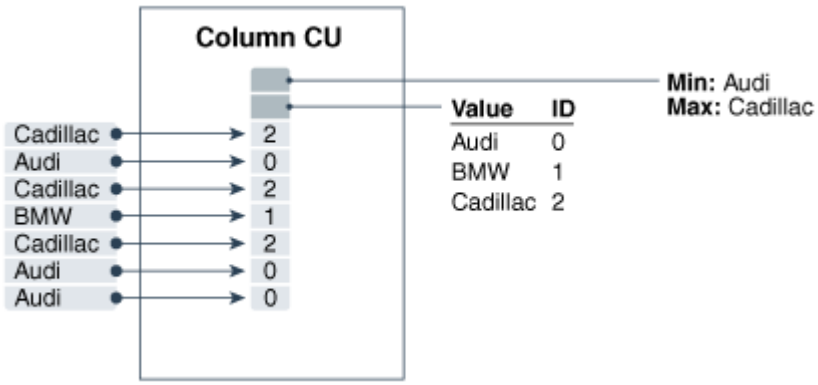
データベースでは、まず、prod_id列に値が5のエントリがあるかどうかが調べられます。データベースで、prod_id列内の位置2に5が見つかったとします。データベースは、次に、この行の対応するcust_id、time_idおよびchannel_idを見つける必要があります。CUでは行順でデータが格納されるため、データベースは、対応するcust_id、time_idおよびchannel_id値をそれらの列内の位置2で見つけることができます。したがって、この問合せに答えるには、データベースは、cust_id、time_idおよびchannel_id列内の位置2から値を抽出し、行を元どおりにまとめ、それをエンド・ユーザーに返す必要があります。

2.2.1.2.2 ローカル・ディクショナリ

CUでは、ローカル・ディクショナリに、非重複値のリスト、およびそれらの対応するディクショナリ・コードが含まれます。

ローカル・ディクショナリには、列に含まれている記号が格納されます。次の図に、CUによるvehicles表のname列の格納方法を示します。

図2-8 ローカル・ディクショナリ



上の図では、CUには7行のみが含まれています。CadillacやAudiなど、このCU内のすべての非重複値には、Cadillacの場合は2、Audiの場合は0というように、異なるディクショナリ・コードが割り当てられます。CUには、元の値ではなくディクショナリ・コードが格納されます。

ノート:

データベースで結合グループのために共通ディクショナリが使用される場合、ローカル・ディクショナリには、記号ではなく共通ディクショナリへの参照が含まれます。たとえば、ローカル・ディクショナリには、vehicles.name列の値 Audi、BWM および Cadillac が格納されるのではなく、101、220 および 66 などのディクショナリ・コードが格納されます。

CUヘッダーには、列の最小値および最大値が含まれます。この例では、最小値はAudiで、最大値はCadillacです。ローカル・ディクショナリには、Audi、BMWおよびCadillacという非重複値のリストが格納されます。それらの対応するディクショナリ・コード(0、1および2)は暗黙的です。各IMCU内のCUのローカル・ディクショナリは、他のIMCU内のローカル・ディクショナリから独立しています。

問合せによってAudi社の自動車でフィルタする場合、データベースでは、0のコードのみについて、このIMCUがスキャンされます。

関連トピック

- 結合グループでどのように共通ディクショナリが使用されるか

関連項目:

「結合グループでどのように共通ディクショナリが使用されるか」

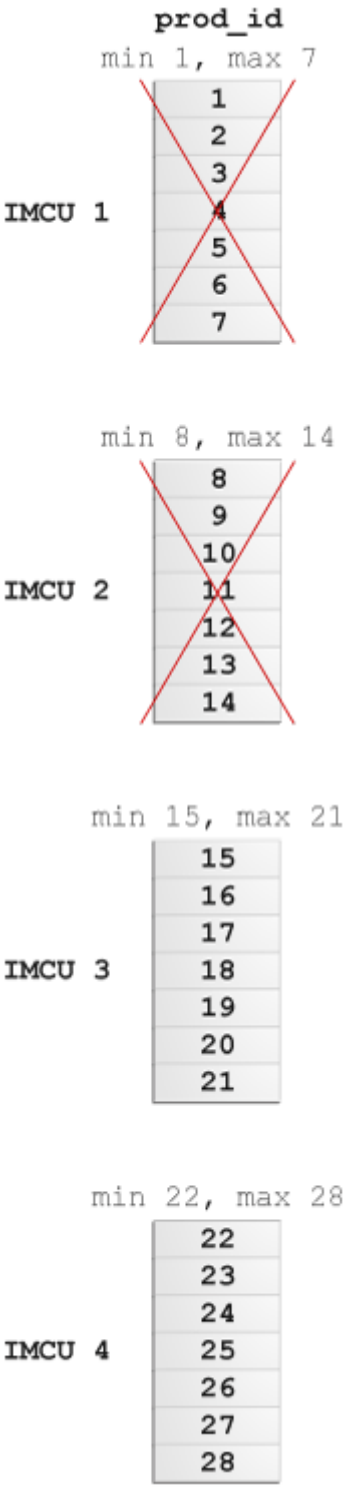
2.2.1.3 インメモリー記憶域索引

すべてのIMCUヘッダーでは、そのCUのためのインメモリー記憶域索引(IM記憶域索引)が自動的に作成および管理されます。IM記憶域索引では、IMCU内のすべての列の最小値および最大値が格納されます。

たとえば、salesがIM列ストアに移入されるとします。この表のためのすべてのIMCUには、すべての列が含まれています。sales.prod_id列は、すべてのIMCU内の別個のCUに格納されます。IMCUヘッダーには、各prod_id CU (および他のすべてのCU)の最小値および最大値が含まれています。

不要なスキャンをなくすには、データベースで、SQLフィルタ述語に基づいてIMCUプルーニングを実行します。次の図のWHERE prod_id > 14 AND prod_id < 29例に示すように、データベースでは問合せの述語を満たすIMCUのみをスキャンします。

図2-9 列データの記憶域索引



2.2.2 スナップショット・メタデータ単位(SMU)

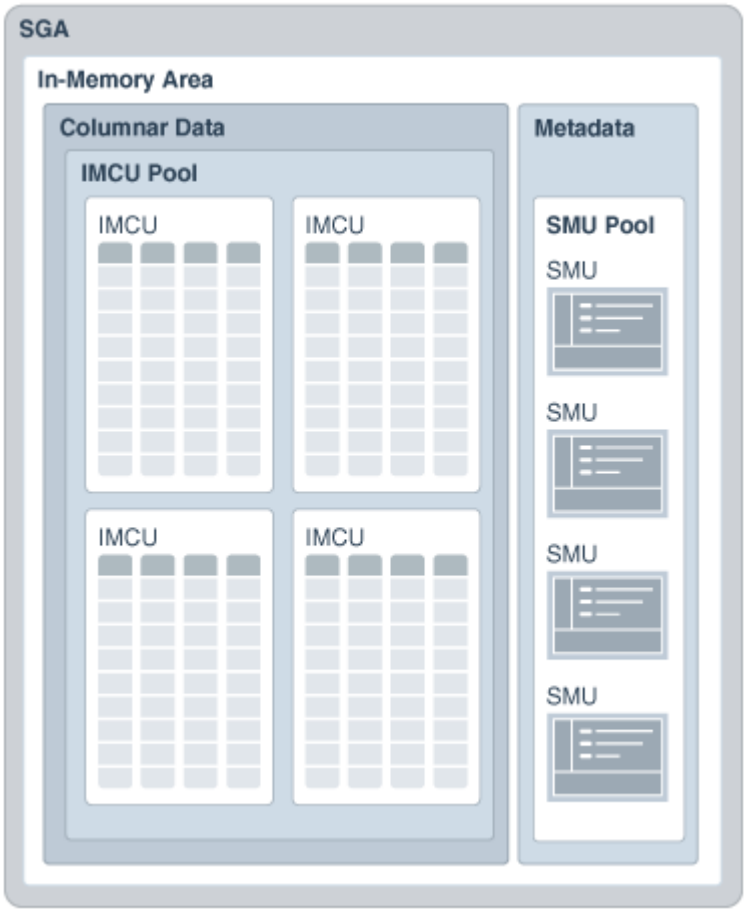
スナップショット・メタデータ単位(SMU)には、関連するIMCUのメタデータおよびトランザクション情報が含まれます。

2.2.2.1 IMCUとSMU

インメモリー領域の列プールには、IMCUおよびIMEUという実データが格納されます。インメモリー領域内のメタデータ・プールには、SMUが格納されます。

図2-10 IMCUとSMU

この図には、データ・プール内のIMCU、およびメタデータ・プール内のSMUを示します。



すべてのIMCUは、別個のSMUにマップされます。したがって、列データ・プールに100個のIMCUが含まれる場合、メタデータ・プールには100個のSMUが含まれます。SMUには、関連付けられているIMCUについて、次のようないくつかのタイプのメタデータが格納されます。

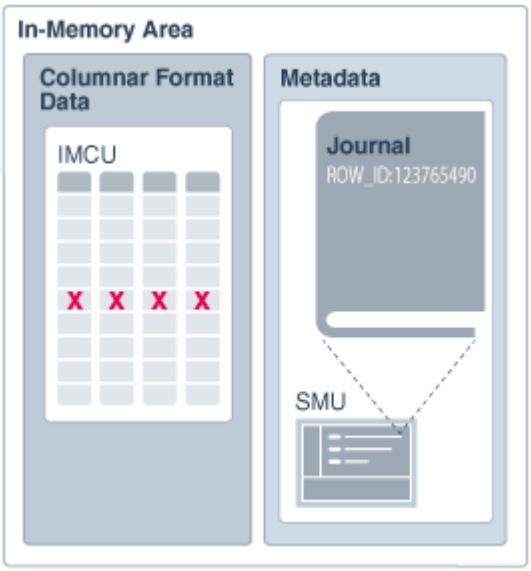
- オブジェクト番号
- 列番号
- 行のマッピング情報

2.2.2.2 トランザクション・ジャーナル

すべてのSMUには、トランザクション・ジャーナルが含まれます。データベースでは、トランザクション・ジャーナルを使用して、IMCUに関するトランザクションの一貫性が保たれます。

データベースでは、IM列ストアが有効になっていない場合と同様に、バッファ・キャッシュを使用してDMLが処理されます。たとえば、UPDATE文で、IMCU内の行を変更するとします。このケースでは、データベースにより、変更した行のROWIDがトランザクション・ジャーナルに追加され、それがDML文のSCNの時点で失効とマークされます。問合せで新しいバージョンの行にアクセスする必要がある場合は、データベースにより、データベース・バッファ・キャッシュからその行が取得されます。

図2-11 トランザクション・ジャーナル



データベースでは、列、トランザクション・ジャーナルおよびバッファ・キャッシュの内容をマージすることで、読取り一貫性が実現されます。再移入中にIMCUがリフレッシュされた場合は、問合せで、IMCUから直接、最新の行にアクセスできます。

関連項目:

どのようにIM列ストアでトランザクションの一貫性が維持されるかの詳細は、IM列ストアの再移入の最適化を参照してください。

2.2.3 インメモリー式単位(IMEU)

インメモリー式単位(IMEU)は、マテリアライズされたインメモリー式(IM式)、およびユーザーが定義した仮想列のための記憶域コンテナです。

データベースではマテリアライズされた式が、IMCUの他の列と同様に取り扱われます。概念的には、IMEUは、その親IMCUの論理拡張です。IMCUに複数の列を含むことができるのと同様に、IMEUには複数の仮想列を含むことができます。

すべてのIMEUは1つのIMCUのみにマップされ、同じ行セットにマップされます。IMEUには、関連付けられているIMCU内に含まれたデータについて式結果が含まれます。IMCUが移入されると、関連付けられているIMEUも移入されます。

一般的なIM式には、1つ以上の列が含まれており、定数が使用されている場合もあります。また、表内の行と1対1でマップされています。たとえば、`employees`表のIMCUに、列`weekly_salary`の行1から1000が含まれているとします。このIMCUに格納されている行について、IMEUで、自動検出されたIM式`weekly_salary*52`、および`weekly_salary*12`として定義されているユーザー定義の仮想列`quarterly_salary`が計算されます。IMCU内の3番目の行は、IMEU内の3番目の行にマップされます。

IMEUは、特定セグメントのIMCUの論理拡張です。デフォルトでは、IMEUは、基本セグメントから、`DISTRIBUTE`や`DUPLICATE`などのOracle Real Application Clusters (Oracle RAC)プロパティを含む`INMEMORY`句プロパティを継承します。IMEU内の記憶域に対して仮想列を有効または無効にすることを選択できます。様々な列に対して圧縮レベルを指定することもできます。

関連トピック

- INMEMORY列の有効化について
- インメモリー・ビュー

2.3 式統計ストア(ESS)

式統計ストア(ESS)は、式評価についての統計を格納するためにオプティマイザによって保持されるリポジトリです。ESSはSGAにあり、ディスクに保持されます。

IM列ストアが有効になっている場合、データベースでは、そのインメモリー式(IM式)機能のためにESSが利用されます。ただし、ESSは、IM列ストアから独立しています。ESSは、データベースの永続的コンポーネントであり、無効にはできません。

データベースでは、ESSを使用して、式がホット(頻繁にアクセスされている)かどうか、ひいてはIM式の候補が判断されます。問合せのハード解析の間には、ESSにより、SELECTリスト、WHERE句、GROUP BY句などに含まれているアクティブな式が探されます。

ESSにより、セグメントごとに次のような式統計が保持されます。

- 実行頻度
- 評価コスト
- タイムスタンプ評価

オプティマイザにより、コストおよび評価された回数に基づいて、各式に重み付きのスコアが割り当てられます。値は、正確ではなく概算となります。アクティブな式ほど、高いスコアとなります。ESSでは、最も頻繁にアクセスされた式について内部リストが保持されます。

DBMS_INMEMORY_ADMINパッケージを使用して、IM式の挙動を制御します。たとえば、IME_CAPTURE_EXPRESSIONSプロシージャは、データベースに、データベース内の最もホットな式を特定して徐々に移入するよう求めます。IME_POPULATE_EXPRESSIONSプロシージャは、データベースに、すぐに式を移入するよう強制します。

ESS情報は、データ・ディクショナリに格納され、DBA_EXPRESSION_STATISTICSビューで公開されます。このビューでは、オプティマイザによってESSに収集されたメタデータが示されます。IM式は、SYS_IMEという文字列で接頭辞が付けられ、DBA_IM_EXPRESSIONSビューで、システム生成の仮想列として公開されます。

関連項目:

- 「IM式について」
- ESSについてさらに学習するには、Oracle Database SQLチューニング・ガイドを参照
- DBA_EXPRESSION_STATISTICSビューについてさらに学習するには、Oracle Databaseリファレンスを参照
- DBMS_INMEMORY_ADMINパッケージについてさらに学習するには、Oracle Database PL/SQLパッケージおよびタイプ・リファレンスを参照

2.4 インメモリー・プロセスのアーキテクチャ

サーバー・プロセスは、問合せおよびDMLに応答して、列データをスキャンし、SMUメタデータを更新します。バックグラウンド・プロセスは、ディスクからIM列ストアに行データを移入します。

2.4.1 インメモリー・コーディネータ・プロセス(IMCO)

インメモリー・コーディネータ・プロセス(IMCO)では、IM列ストアのための多くのタスクが管理されます。その主要タスクは、列データのバックグラウンド移入および再移入の開始です。

移入はストリーミング・メカニズムであり、行データを列形式に変換してから、それを圧縮します。IMCOにより、NONE以外の任意の優先順位で、自動的にINMEMORYオブジェクトの移入が開始されます。優先順位がNONEのオブジェクトがアクセスされた場合は、IMCOにより、領域管理ワーカー・プロセス(Wnnn)のプロセスを使用してそれらが移入されます。

また、IMCOバックグラウンド・プロセスでは、IM列ストア・オブジェクトが失効しきい値に達したときに、それらのしきい値ベース再移入が開始されます。IMCOでは、失効エントリが存在するが失効しきい値に満たないIM列ストア内の任意のIMCUに対して、トリクル再移入を引き起こす場合があります。

トリクル再移入はバックグラウンドで自動的に行われます。ステップは次のとおりです。

1. IMCOが起動します。
2. IMCOでは、IMCUに失効エントリが存在するかどうかも含めて、移入タスクを実行する必要性の有無が判断されます。
3. IMCOで失効エントリが見つかった場合、領域管理ワーカー・プロセスがトリガーされてこれらのエントリがIMCUで再移入されます。
4. IMCOは2分間スリープしてから、ステップ1に戻ります。

関連項目:

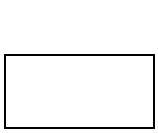
- 「IM列ストアの再移入の最適化」
- バックグラウンド・プロセスについてさらに学習するには、『Oracle Databaseリファレンス』を参照

2.4.2 領域管理ワーカー・プロセス(Wnnn)

領域管理ワーカー・プロセス(Wnnn)は、IMCOにかわってデータを移入または再移入します。

移入中に、Wnnnプロセスは、IMCU、SMUおよびIMEUを作成する役割を担います。IMEUの作成時は、ワーカー・プロセスにより、次のようなタスクが実行されます。

- 移入のための仮想列を特定する
- 仮想列値を作成する
- 各行の値を計算し、データを列形式に変換し、それを圧縮する
- オブジェクトを領域レイヤーに登録する
- IMEUをそれらの対応するIMCUに関連付ける



ノート:

IMEU を作成する間も、親 IMCU は、引き続きキューで使用可能です。

再移入中は、Wnnnプロセスにより、既存のIMCUおよびトランザクション・ジャーナルに基づいて、IMCUの新しいバージョンが作成され、古いバージョンは一時的に保持されます。このメカニズムは、ダブル・バッファリングと呼ばれます。

データベースは、IM列ストアの内外にIM式を迅速に移動できます。たとえば、IMCUがIMEUなしで作成された場合、データベースは、IMCUに再移入メカニズム全体を強制的に経験させることなく、後でIMEUを追加できます。

INMEMORY_MAX_POPULATE_SERVERS初期化パラメータは、移入のために開始できる、ワーカー・プロセスの最大数を制御します。INMEMORY_TRICKLE_REPOPULATE_PERCENT初期化パラメータは、ワーカー・プロセスがトリクル再移入を実

行できる回数の最大パーセンテージを制御します。

関連項目:

- 「インメモリー移入に対するオブジェクトの手動による有効化について」
- 「IM列ストアの再移入について」
- 「インメモリー初期化パラメータ」
- バックグラウンド・プロセスについてさらに学習するには、『Oracle Databaseリファレンス』を参照

2.4.3 In-Memory動的スキャン

In-Memory動的スキャン(IM動的スキャン)は、軽量スレッドを使用して、インメモリー表スキャンをパラレル化します。

2.4.3.1 IM動的スキャンの目的

追加のCPUが使用可能な場合、IM動的スキャンは、CPUにバインドされたインメモリー表スキャンを高速化します。

IM動的スキャンは、自動的にアイドル状態のCPUリソースを使用してIMCUを並行してスキャンし、CPU使用率を最大化します。CPUリソースが使用可能な場合、アプリケーションはより高速な分析問合せ結果を自動的に取得できます。スキャンは動的なため、既存のワークロードに影響を与えずに余分なCPU帯域幅を使用できます。

IM動的スキャンは、従来のOracleのパラレル実行よりも柔軟性がありますが、相互に排他的ではありません。動的スキャンは、プロセス内で複数の軽量スレッドの実行を使用します。通常、動的スキャンのパフォーマンス・オーバーヘッドは少なくなっています。

関連項目:

リソース・マネージャについてさらに学習するには、Oracle Database管理者ガイドを参照

2.4.3.2 IM動的スキャンの仕組み

IM動的スキャンは、IMCUを並行して読み取ることで最適なパフォーマンスを実現します。

2.4.3.2.1 軽量スレッドについて

軽量スレッドは、全表スキャンをパラレル化するために役立つ実行エンティティです。Oracleプロセスのメモリー・オーバーヘッドが増えないため「軽量」です。

ノート:

IM 動的スキャンで使用される軽量スレッドは、マルチスレッド Oracle Database モデルの通常スレッドと同じではありません。

軽量スレッドは、一連のIMCUのスキャンを調整する、表スキャンプロセスという親フォアグラウンドまたはPQプロセスのリソースを共有します。スレッドは独自の独立した実行フローを維持します。データベースは、スレッドの優先度付けと非同期実行によってスキャンをパラレル化できます。

対象となる問合せの場合、プロセスはスレッドのプールを割り当てます。リソース・マネージャは、データベース・ホストのCPU数およびシステムでの現在の負荷に基づいてプール内のスレッド数を自動的に決定します。アイドル時間が内部しきい値に達して、

その時点でデータベースがスレッドを終了しないかぎり、スレッドのプールは後続の問合せのためにセッションで使用可能なままです。

スレッド間の通信は、プロセス内で排他的に発生します。このため、データベース・インスタンス・レベルで競合は発生しません。

関連項目:

マルチスレッドOracle Databaseモデルについて学習するには、Oracle Database概要を参照

2.4.3.2.2 データベースがIM動的スキャンを考慮する場合

軽量スレッドは、CPUリソース・プランが有効で(たとえば、RESOURCE_MANAGER_PLAN=DEFAULT_PLAN)、データベースのCPU使用率が低い場合に有効になります。

軽量スレッドが有効な場合、データベースは、現在IM列ストアに移入されているオブジェクトをアプリケーションが問い合せたときに、IM動的スキャンを考慮します。通常、シリアルまたはパラレル問合せは、次の特性を持つ場合にIM動的スキャンの候補となります。

- 多数のIMCUまたは列にアクセスする
- 表内のすべての行を消費する
- CPUに負担がかかる

Oracle Database Resource Manager (リソース・マネージャ)は、INMEMORY_SIZEが0より大きい場合に自動的に有効化されます。また、IM動的スキャンに必要です。リソース・マネージャは、軽量スレッドを使用するタイミングと方法を決定します。軽量スレッドは、未使用のリソースを利用しているため、データベース内で最も優先度の低い操作です。

ノート:

IM 動的スキャンを実行するには、CPU_COUNT が 24 以上である必要があります。

2.4.3.2.3 IM動的スキャンの仕組み

リソース・マネージャが軽量スレッドを割り当てて、IMCUのスキャンをパラレル化します。

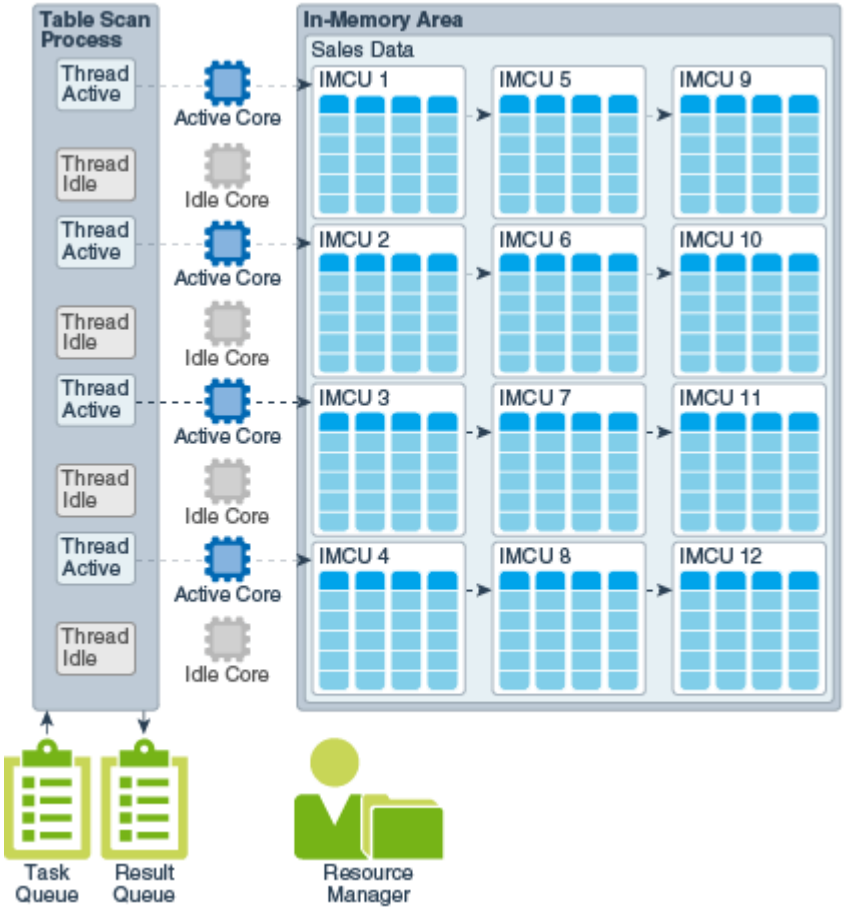
問合せがIM動的スキャンによる利益を得られるとデータベースが判断した場合、通常は次のように進みます。

1. 表スキャン・プロセスが、軽量スレッドのプールを生成します。
2. 表スキャン・プロセスは、スキャンが必要なすべてのIMCU用に個別のタスクを作成した後、各タスクをタスク・キューに追加します。
3. リソース・マネージャが、表スキャンに参加できるスレッド数を決定します。
4. アクティブ・スレッドがタスク・キューからタスクを選択し、表スキャン・プロセスは完了したタスクの結果を消費します。

データベースの負荷に応じて、リソース・マネージャは問合せの実行中にアクティブな軽量スレッドの数を継続して調整します。 CPUリソースが使用できない場合、表スキャン・プロセスは軽量スレッドを使用せずにスキャンを実行します。

次の図は、sales表内の12個のIMCUのIM動的スキャンを示しています。

図2-12 IM動的スキャン



前の図では、データベース・ホストに8つのCPUコアがあります。内部アルゴリズムに基づいて、リソース・マネージャは4つのスレッドを割り当てて表スキャン・プロセスを支援します。このシナリオでは、4つのCPUコアが他の同時データベース操作で使用するためにアイドル状態のままになります。

2.4.3.3 IM動的スキャンのインタフェース

IM動的スキャンは透過的です。これは、アプリケーションの変更を必要としないことを意味し、リソース・マネージャによって自動的に制御されます。

IM動的スキャンでは、リソース・マネージャが必要です。これは、INMEMORY_SIZEが0より大きい場合に自動的に有効になります。特定のリソース・プランは必要ありません。

いくつかの新しいセッション統計が、IM動的スキャンの使用状況を追跡します。各スレッドは、トレース・データを別のトレース・ファイルに書き込みます。

実行計画は変更されていません。次の図に、実行計画のサンプルを示します。

```
SQL> SELECT MAX(l_quantity) largest_order FROM lineitem;
LARGEST_ORDER
-------------
           50
Elapsed: 00:00:03.41
Execution Plan
----------------------------------------------------------
Plan hash value: 1885658499
-------------------------------------------------------------------------------------------
------
|Id| Operation                    | Name     |Rows| Bytes |Cost(%CPU)| Time   |Pstart|
Pstop|
-------------------------------------------------------------------------------------------
------
|0| SELECT STATEMENT             |          |    1 |    3| 116K  (4)| 00:00:05 |    |
|
```

```
|1|  SORT AGGREGATE               |          |    1 |    3|          |          |    |
|
|2|   PARTITION RANGE ALL         |          | 600M |1716M| 116K  (4)| 00:00:05 |  1 |
84 |
|3|    TABLE ACCESS INMEMORY FULL| LINEITEM | 600M |1716M| 116K  (4)| 00:00:05 |  1 |
84 |
-------------------------------------------------------------------------------------
------
NAME                                                  VALUE
-----------------------------------------------------------
IM scan CUs memcompress for query low                  1147
IM scan bytes in-memory                           5.1790E+10
IM scan bytes uncompressed                        7.6722E+10
IM scan CUs columns accessed                           1147
IM scan rows                                       600037902
IM scan rows projected                                    29
IM scan (dynamic) rows                             600037902
IM scan (dynamic) multi-threaded scans                     1
IM scan (dynamic) tasks processed by thread             1146
```

計画の特性を考えてみます。

1. 実行計画は変更されていません。

   計画では、ステップ3でIM動的スキャンに言及していないので注意してください。ただし、SQLモニター・レポートの双眼鏡アイコンをクリックすると、「スレッドの動的スキャン・タスク」が表示されます。

2. `IM scan (dynamic) multi-threaded scans`がゼロ以外の場合、データベースがIM動的スキャンを使用したことを意味します。

3. `IM scan CUs memcompress for query low`は、`lineitem`表に1147個のIMCUが存在することを示しています。

4. `IM scan (dynamic) tasks processed by thread`は、並列処理されたIMCUの数を示しています。

   その数は1146で、`IM scan CUs memcompress for query low`で示されている1147の合計数より少ない数です。データベースは、パラレル化なしで最初のIMCUを分析して、パラレル化に価値があるかどうかを判断しました。答えは「はい」であったため、データベースは残りの1146個のIMCUを並行してスキャンしました。

5. `IM scan (dynamic) rows`および`IM scan rows are equal`は、スレッドで問合せのすべての行が取得されたことを意味します。

関連項目:

- リソース・マネージャについてさらに学習するには、『Oracle Database管理者ガイド』を参照
- インメモリー統計の詳細は、Oracle Databaseリファレンスを参照

2.5 CPUアーキテクチャ: SIMDベクター処理

IM列ストアに移入されるデータについては、データベースはSIMD (単一命令、複数データ)処理を使用します。

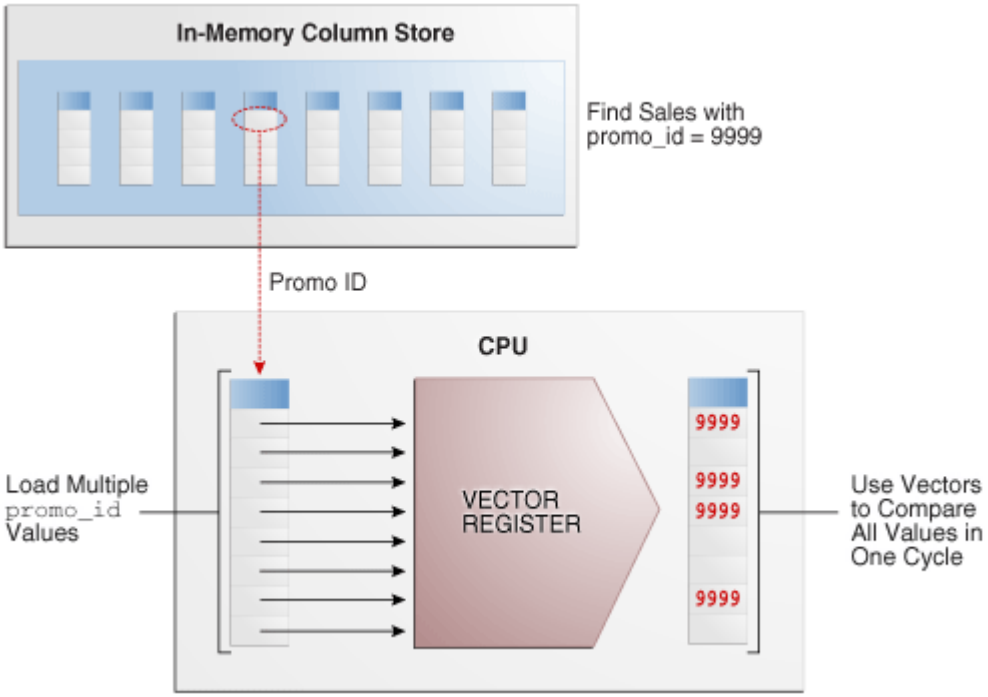
SIMDユニットは、データを複数の別々の命令で処理するのではなく、単一の命令で、**ベクトル**という1つの単位としてデータを処理できるようにするプロセッサです。たとえば、ループを使用して4つの追加操作を実行するのではなく、SIMDでは、4つの数値セットをベクトルにロードし1つの追加操作を実行できます。SIMD処理は、**ベクトル化**と呼ばれることもあります。

IM列ストアでは、CPUでベクター・レジスタにロードして評価できる、列エントリの数が最大化されます。列内の各エントリを1回

に1つずつ評価するかわりに、データベースは単一のCPU命令で列値のセットを評価します。SIMDベクター処理により、データベースで、1秒当たり何十億もの行をスキャンできるようになります。

たとえば、promo_id値に9999を使用した、sales表内の注文の総数を確認する問合せをアプリケーションが発行します。sales表は、IM列ストア内に存在します。問合せでは、まず、次の図に示すようにsales.promo_id列のみがスキャンされます。

図2-13 SIMDベクター処理



CPUではデータが次のように評価されます。

1. promo_id列の最初の8つの値(値の数はデータ型と圧縮モードによって異なる)がSIMDレジスタにロードされ、単一の命令で9999という値と比較されます。
2. エントリが破棄されます。
3. 別の8つの値がSIMDレジスタにロードされ、すべてのエントリが評価されるまでこのように続きます。

2.5.1 SIMDおよびOracle LOB

Oracleデータベース18cは、特定のLOB列に対するSQL演算子を含む問合せに対するSIMDベクター・サポートを提供します。

サポートの内容は、LOBの種類によって異なります。

- インラインLOB

  IM列ストアは、IMCU内の4KB未満のLOBであるインラインLOBに対して連続した記憶域を提供します。列型記憶域により、データベース・バッファ・キャッシュからLOBデータをアセンブルするオーバーヘッドを排除することで、より高速な問合せ処理が可能です。

- アウトラインLOB

  この場合、IM列ストアは40バイトのLOBロケータのみを格納します。アウトライン列は、列の最適化による利益を得られません。

前述のルールには、例外が1つあります。IMEUは、LOBデータ型として定義されたJSON列に対して最大32KBの連続した記

憶域を割り当てることができます。IMEUは、これらの列をOSON形式で格納します。これにより、SIMD処理を使用してより高速な問合せのパフォーマンスを提供できます。

関連項目:

LOBについてさらに学習するには、Oracle Database SecureFilesおよびラージ・オブジェクト開発者ガイドを参照してください。

2.5.2 SIMDおよびOracle Numbers

QUERY LOWで表を圧縮する場合、NUMBER列は、ハードウェアでのネイティブ計算が有効になる最適化された形式を使用してエンコードされます。

SIMDベクター処理では、単純な集計、GROUP BY集計、算術演算を使用でき、大きいメリットがあります。パフォーマンスの向上は、集計が算術計算に費やす時間によって異なります。集計によっては、最大9倍までの効果があります。

関連項目:

- 「インメモリー算術の最適化」
- Oracle Database SQL言語リファレンス

2.5.3 SIMDおよびExadataスマート・フラッシュ・キャッシュ

ハイブリッド列圧縮形式でデータを格納する他、Exadataスマート・フラッシュキャッシュはデータを純粋な列形式で格納できます。

Exadata Smart ScanはSIMD述語をサポートします。利点は、インメモリーのパフォーマンスがDRAM記憶域から二次記憶域に及ぶことです。

デフォルトでは、Exadataスマート・フラッシュ・キャッシュは、レベルMEMCOMPRESS FOR CAPACITY LOWを使用してデータを圧縮します。圧縮レベルを変更するか、列形式を完全に無効にするには、ALTER TABLE ... NO CELLMEMORY文を使用します。

関連項目:

- 「Exadataフラッシュ・キャッシュのインメモリー・サポート」
- Oracle Exadata Database Machineシステム概要

第II部 IM列ストアの構成および移入

インメモリー列ストア(IM列ストア)を有効化およびサイズ設定できます。また、オブジェクトのIn-Memory設定を構成し、これらのオブジェクトをIM列ストアに移入することもできます。

3 IM列ストアの有効化およびサイズ設定

IM列ストアを有効化または無効化するには、INMEMORY_SIZE初期化パラメータの値を指定します。

3.1 IM列ストアの有効化の概要

IM列ストアのサイズを有効にするには、INMEMORY_SIZE初期化パラメータを設定します。

デフォルトでは、INMEMORY_SIZEは0に設定されており、これは、IM列ストアが無効になっていることを意味します。IM列ストアを有効にするには、データベース・インスタンスを再起動する前に、INMEMORY_SIZE初期化パラメータに最小値の100 MBを設定します。ALTER SYSTEM文を使用することで、INMEMORY_SIZEサイズ設定を動的に増加させることができます。

Database In-Memoryベース・レベルのみの場合、CDBのサイズは16 GB以下にする必要があります。Oracle RACデータベースでは、すべてのインスタンスが16 GB以下である必要があります。

デフォルトでは、表、表領域またはマテリアライズド・ビューに対してCREATEまたはALTER文のINMEMORY句を使用して、IM列ストアに移入するための候補を指定する必要があります。

関連項目:

- 「インメモリー初期化パラメータ」
- INMEMORY_SIZE初期化パラメータについてさらに学習するには、『Oracle Databaseリファレンス』を参照
- INMEMORY句の詳細は、『Oracle Database SQL言語リファレンス』を参照
- 各種エディションおよびサービスでサポートされる機能の詳細は、『Oracle Databaseライセンス情報ユーザー・マニュアル』を参照

3.2 IM列ストアの必須サイズの推定

要件に基づいてIM列ストアのサイズを推定してから、それらの要件に合わせてIM列ストアのサイズを変更します。圧縮の適用により、メモリー・サイズを削減できます。

IM列ストアで必要なメモリー量は、それに格納するデータベース・オブジェクト、および各オブジェクトに適用する圧縮方法によって異なります。INMEMORYオブジェクトに対する圧縮方法を選択する際には、使用可能なメモリー量に対するパフォーマンス利益のバランスを取ります。

- メモリー・サイズを最大限に削減するには、FOR CAPACITY HIGHまたはFOR CAPACITY LOW圧縮方法を選択します。ただし、これらのオプションでは、データを解凍するための問合せ実行中にさらにCPUが必要となります。
- 最善の問合せパフォーマンスを得るには、FOR QUERY HIGHまたはFOR QUERY LOW圧縮方法を選択します。ただし、これらのオプションでは、より多くのメモリーが消費されます。

IM列ストアのサイズ設定時には、次のガイドラインを考慮してください。

1. IM列ストアに移入するすべてのオブジェクトについて、消費するメモリー量を推定します。

   Oracle Compression AdvisorではMEMCOMPRESS句を使用してユーザーが実感できる圧縮率を推定します。アドバイザはDBMS_COMPRESSIONインタフェースを使用します。

2. 個々の容量を加算します。

ノート:

移入後は、V$IM_SEGMENTS によって、ディスク上のそれらのオブジェクトの実サイズ、および IM 列ストアでのそれらのサイズが示されます。この情報を使用して、移入したオブジェクトの圧縮率を計算できます。ただし、オブジェクトがディスク上で圧縮された場合、この問合せでは正しい圧縮率は示されません。

3. In-Memory最適化算術を構成し、インメモリー表がFOR QUERY LOW圧縮を使用する場合は、NUMBER列の二重記憶を考慮して約15%を追加します。
4. データベース・オブジェクトの拡大を考慮するため、およびDML操作後に行の最新版を格納するために、領域を追加します。

   動的サイズ変更の最小容量は128 MBです。

関連項目:

- 「IM列ストアの圧縮方法」
- 「データベースに対するIM列ストアの有効化」
- 「IM列ストアのサイズの動的な増加」
- 「インメモリー最適化算術について」
- 圧縮アドバイザを使用した圧縮率の推定方法を学習するには、『Oracle Database管理者ガイド』を参照
- V$IM_SEGMENTSについて学習するには、『Oracle Databaseリファレンス』を参照

3.3 データベースに対するIM列ストアの有効化

IM列ストアに表またはマテリアライズド・ビューを移入する前に、データベースのIM列ストアを有効にする必要があります。

このコンテキストでは、「データベース」は非CDB、CDBまたはPDBです。CDBでは、IM列ストアの全体のサイズがCDBルートのINMEMORY_SIZE設定によって決まります。デフォルトでは、すべてのPDBがIM列ストアにアクセスできます。

個々のPDB内では、INMEMORY_SIZEを別の値に設定することで、共有インメモリー領域へのアクセスを制限できます。たとえば、PDBが100個のCDBでは、CDBレベルでINMEMORY_SIZEを16Gに設定し、あるPDBではINMEMORY_SIZEを10Gに設定し、2番目のPDBでは6Gに、残りのPDBでは0に設定できます。

前提条件

この作業では、次のことを想定しています。

- データベースはオープンしています。
- COMPATIBLE初期化パラメータが12.1.0以上に設定されています。
- INMEMORY_SIZE初期化パラメータが0(デフォルト)に設定されています。
- Database In-Memoryベース・レベルを使用する。

IM列ストアを有効にするには:

1. SQL*PlusまたはSQL Developerで、管理者権限を持つユーザーとしてデータベースにログインします。
2. INMEMORY_SIZE初期化パラメータをゼロ以外の値に設定します。

   最小設定は、100Mです。

   ALTER SYSTEM文を使用してサーバー・パラメータ・ファイル(SPFILE)でこの初期化パラメータを設定する場合、SCOPE=SPFILEを指定する必要があります。

   たとえば、次の文では、インメモリー領域サイズを16 GBに設定します。

   ```
   ALTER SYSTEM SET INMEMORY_SIZE = 16G SCOPE=SPFILE;
   ```

3. Database In-Memoryベース・レベルの場合は、INMEMORY_FORCE初期化パラメータにBASE_LEVELを設定します。

   たとえば、次の文はベース・レベルを指定しています。

   ```
   ALTER SYSTEM SET INMEMORY_FORCE=BASE_LEVEL SCOPE=SPFILE;
   ```

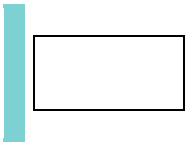
   PDBレベルでは、INMEMORY_FORCE=BASE_LEVELを設定できません。また、このパラメータを動的に設定することはできません。

4. データベースを停止してから再度開きます。

   データベースを再度開いてSGA内のIM列ストアを初期化する必要があります。

5. 必要な場合は、IM列ストアに現在割り当てられているメモリーの量を確認します。

   ```
   SHOW PARAMETER INMEMORY_SIZE
   ```



ノート:

IM 列ストアを有効にした後は、データベースを再度開くことなく、そのサイズを動的に増加できます。

例3-1 IM列ストアの有効化

INMEMORY_SIZE初期化パラメータが0に設定されていることが前提です。次のSQL*Plusの例では、INMEMORY_SIZEを16 GBに設定し、その変更内容が有効になるよう、データベース・インスタンスを停止してからデータベースを再度開きます。

```
SQL> SHOW PARAMETER INMEMORY_SIZE
NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
inmemory_size                        big integer 0
SQL> ALTER SYSTEM SET INMEMORY_SIZE=16G SCOPE=SPFILE;
System altered.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.
Total System Global Area   11525947392 bytes
Fixed Size                     8213456 bytes
Variable Size                754977840 bytes
Database Buffers              16777216 bytes
Redo Buffers                   8560640 bytes
In-Memory Area             10737418240 bytes
Database mounted.
Database opened.
SQL> SHOW PARAMETER INMEMORY_SIZE
```

```
NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
inmemory_size                        big integer 16G
```

関連項目:

- 「複数のIM列ストア」
- データベースの互換性レベルの設定の詳細は、『Oracle Databaseアップグレード・ガイド』を参照
- INMEMORY_SIZE初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照

3.4 IM列ストアのサイズの動的な増加

IM列ストアにより多くのメモリーが必要な場合は、そのサイズを動的に増加できます。

IM列ストアのサイズを動的に減少することはできません。INMEMORY_SIZEを現在の設定よりも小さい値に設定する場合は、ALTER SYSTEM文にSCOPE=SPFILEを指定する必要があります。SCOPE=SPFILEを指定してこのパラメータを設定した場合は、変更を有効にするためにデータベースを再起動する必要があります。

前提条件

IM列ストアのサイズを動的に増加させるには、次の前提条件を満たしている必要があります。

- 列ストアが有効になっている必要があります。
- 互換性レベルが12.2.0以上である必要があります。
- データベース・インスタンスをSPFILEで起動する必要があります。
- IM列ストアの新しいサイズが、現在のINMEMORY_SIZE設定よりも128 MB以上大きい必要があります。

1. SQL*PlusまたはSQL Developerで、管理者権限でデータベースにログインします。
2. 必要な場合は、IM列ストアに現在割り当てられているメモリーの量を確認します。

   ```
   SHOW PARAMETER INMEMORY_SIZE
   ```

3. SCOPE=BOTHまたはSCOPE=MEMORYが指定されたALTER SYSTEM文で、INMEMORY_SIZE初期化パラメータを現在のIM列ストアのサイズよりも大きい値に設定します。

   このパラメータを動的に設定する場合は、現在の値よりも高い値に設定する必要があり、IM列ストアのサイズを新しい値に動的に増加するには、SGAに使用可能な十分なメモリーが存在する必要があります。

   たとえば、次の文は、INMEMORY_SIZEを500Mに動的に設定します。

   ```
   ALTER SYSTEM SET INMEMORY_SIZE = 500M SCOPE=BOTH;
   ```

関連項目:

- 「データベースに対するIM列ストアの有効化」
- INMEMORY_SIZE初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照

3.5 IM列ストアの無効化

INMEMORY_SIZE初期化パラメータをゼロに設定してからデータベースを再度開くことで、IM列ストアを無効にできます。

前提

この作業では、開いているデータベースでIM列ストアが有効になっていることを想定しています。

IM列ストアを無効にするには:

1. サーバー・パラメータ・ファイル(SPFILE)でINMEMORY_SIZE初期化パラメータを0に設定します。
2. データベースを停止します。
3. データベース・インスタンスを起動してから、データベースを開きます。

関連項目:

INMEMORY_SIZE初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

4 インメモリー移入に対するオブジェクトの有効化

この章では、圧縮や優先順位のオプションの設定を含め、IM列ストアへの移入に対してオブジェクトを有効および無効にする方法を説明します。

この章のトピックは、次のとおりです:

4.1 インメモリー移入に対するオブジェクトの手動による有効化について

INMEMORY句のあるオブジェクトのみが、IM列ストアへの移入の対象になります。この句を手動で適用するには、CREATE TABLEやALTER TABLEなどのDDL文を使用する必要があります。

4.1.1 インメモリー移入に対するオブジェクトの有効化の目的

デフォルトでは、オブジェクトにNO INMEMORY属性が暗黙的に含まれており、これはオブジェクトがIM列ストアへの移入の対象ではないことを意味します。

オブジェクトは、DDLを使用してINMEMORYとして指定した場合のみ、移入の対象となります。オブジェクトをINMEMORYとして有効にするということは、オブジェクトがIM列ストアに存在可能であると指定するということです。インメモリー移入は個別のステップであり、データベースがディスクから既存の行形式データを読み取って、そのデータを列形式に変換してIM列ストアに格納するときに発生します。

移入は、ディスク上の既存のデータを列形式に変換することであり、新しいデータを列形式に変換する再移入とは異なります。IMCUは読取り専用の構造であるため、Oracle Databaseでは、それらは行の変更時に移入されません。正確に述べると、データベースでは、行変更がトランザクション・ジャーナルに記録されてから、再移入の一環として新しいIMCUが作成されます。

関連トピック

- IM列ストアの再移入の最適化
- インメモリー圧縮単位(IMCU)

4.1.2 インメモリー移入はどのように機能するか

データベース・インスタンス起動時またはINMEMORYオブジェクトへのアクセス時のどちらかにデータベースによってオブジェクトがIM列ストアに移入されるよう指定できます。

移入アルゴリズムは、単一インスタンスまたはOracle RACのどちらを使用するかによっても異なります。

この項では、次の項目について説明します。

4.1.2.1 インメモリー移入の優先順位付け

DDL文には、移入キューをより細かく制御できるようにするINMEMORY PRIORITY副句が含まれています。

ノート:

INMEMORY PRIORITY 副句は、移入の速度ではなく、移入の優先順位を制御します。

優先度レベル設定は、別々の列サブセットではなく、全体の表、パーティションまたはサブパーティションに適用されます。オブジェクト上でINMEMORY属性を設定することは、このオブジェクトがIM列ストアへの移入の候補であることを意味します。データベー

スでそのオブジェクトがすぐに移入されることを意味するわけではありません。

ノート:

ディスク上のセグメントが 64KB 以下の場合、IM 列ストアに移入されません。したがって、IM 列ストアに対して有効になっている小規模データベース・オブジェクトは、移入されないことがあります。

Oracle Databaseでは、次のように優先順位付けが管理されます。

- オンデマンド移入

  デフォルトでは、INMEMORY PRIORITYパラメータはNONEに設定されています。この場合、データベースでは、オブジェクトは表の全体スキャンを通じてアクセスされるときのみ移入されます。オブジェクトがアクセスされることがないか、索引スキャンまたはROWIDによるフェッチを通じてしかアクセスされない場合は、移入が起こることはありません。

- 優先度ベース移入

  PRIORITYがNONE以外の値に設定されている場合、Oracle databaseでは、内部的に管理される優先順位キーを使用して、オブジェクトが自動的に移入されます。この場合、全体スキャンは、移入に必要な条件ではありません。データベースでは、次のことが実行されます。

  - データベース・インスタンスの再起動後に、列データをIM列ストアに自動的に移入します
  - 指定された優先度レベルに基づいて、INMEMORYオブジェクトの移入を問い合せます

    たとえば、INMEMORY PRIORITY CRITICALで変更された表はINMEMORY PRIORITY HIGHで変更された表よりも優先され、この表もまた、INMEMORY PRIORITY LOWで変更された表よりも優先されます。IM列ストアに十分な領域がない場合、Oracle Databaseは領域が使用可能になるまで追加オブジェクトを移入しません。

  - オブジェクトに対する変更がIM列ストアで記録されるまで、ALTER TABLEまたはALTER MATERIALIZED VIEW文からの返答を待機します

セグメントがIM列ストアに移入された後、データベースでそれが除去されるのは、そのセグメントが削除または移動されるか、NO INMEMORY属性で更新されるときのみです。セグメントは手動またはADOポリシーによって除去できます。

例4-1 IM列ストアへのオブジェクトの移入

この例を完了する前に、IM列ストアをデータベースに対して有効にする必要があります。

1. データベースに管理者としてログインしてから、次のようにcustomers表を問い合せます。

```
SELECT cust_id, cust_last_name, cust_first_name
FROM   sh.customers
WHERE  cust_city = 'Hyderabad'
AND    cust_income_level LIKE 'C%'
AND    cust_year_of_birth > 1960;
```

2. 問合せの実行計画を表示します。

```
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(FORMAT=>'+ALLSTATS'));
SQL_ID  frgk9dbaftmm9, child number 0
-------------------------------------
SELECT cust_id, cust_last_name, cust_first_name FROM   sh.customers
WHERE  cust_city = 'Hyderabad' AND    cust_income_level LIKE 'C%' AND
 cust_year_of_birth > 1960
Plan hash value: 2008213504
```

```
-------------------------------------------------------------------------------
| Id| Operation         | Name      |Starts|E-Rows|A-Rows|   A-Time   |Buffers|
-------------------------------------------------------------------------------
|  0| SELECT STATEMENT  |           |     1|      |    6 |00:00:00.01 |   1523|
|* 1|  TABLE ACCESS FULL| CUSTOMERS |     1|    6 |    6 |00:00:00.01 |   1523|
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter(("CUST_CITY"='Hyderabad' AND "CUST_YEAR_OF_BIRTH">1960 AND
              "CUST_INCOME_LEVEL" LIKE 'C%'))
```

3. sh.customers表をIM列ストアへの移入のために有効にします。

```
ALTER TABLE sh.customers INMEMORY;
```

上の文では、デフォルトの優先順位であるNONEが使用されます。優先順位なしでオブジェクトを移入するには、全体スキャンが必要となります。

4. sh.customers表からのデータがIM列ストアに移入されているかどうかを判断するには、次の問合せを実行します(出力例が含まれています)。

```
SELECT SEGMENT_NAME, POPULATE_STATUS
FROM   V$IM_SEGMENTS
WHERE  SEGMENT_NAME = 'CUSTOMERS';
no rows selected
```

この場合、sh.customers表はまだスキャンされていないため、IM列ストアに移入されているセグメントはありません。

5. ステップ1と同じ文を使用してsh.customersを問い合せます。

```
SELECT cust_id, cust_last_name, cust_first_name
FROM   sh.customers
WHERE  cust_city = 'Hyderabad'
AND    cust_income_level LIKE 'C%'
AND    cust_year_of_birth > 1960;
```

6. カーソルの問合せでは、データベースで全体スキャンを実行しIM列ストアにアクセスしたことが示されます。

```
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(FORMAT=>'+ALLSTATS'));
SQL_ID  frgk9dbaftmm9, child number 0
-------------------------------------
SELECT cust_id, cust_last_name, cust_first_name FROM   sh.customers
WHERE  cust_city = 'Hyderabad' AND    cust_income_level LIKE 'C%' AND
 cust_year_of_birth > 1960
Plan hash value: 2008213504
-------------------------------------------------------------------------------
--
| Id| Operation           | Name            |Starts|E-Rows|A-Rows|A-
Time|Buffers|
-------------------------------------------------------------------------------
--
|  0| SELECT STATEMENT           |           |    1|     | 6 |00:00:00.02| 1523
|
|* 1|  TABLE ACCESS INMEMORY FULL| CUSTOMERS |    1|    6| 6 |00:00:00.02| 1523
|
-------------------------------------------------------------------------------
--
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - inmemory(("CUST_CITY"='Hyderabad' AND "CUST_YEAR_OF_BIRTH">1960 AND
              "CUST_INCOME_LEVEL" LIKE 'C%'))
       filter(("CUST_CITY"='Hyderabad' AND "CUST_YEAR_OF_BIRTH">1960 AND
              "CUST_INCOME_LEVEL" LIKE 'C%'))
```

7. V$IM_SEGMENTSを再度問い合せます(出力例が含まれています)。

```
COL SEGMENT_NAME FORMAT a20
SELECT SEGMENT_NAME, POPULATE_STATUS
FROM   V$IM_SEGMENTS
WHERE  SEGMENT_NAME = 'CUSTOMERS';
SEGMENT_NAME         POPULATE_STATUS
-------------------- --------------
CUSTOMERS            COMPLETED
```

POPULATE_STATUS内の値COMPLETEDは、表がIM列ストアに移入されていることを意味します。

8. DBA_FEATURE_USAGE_STATISTICSビューでは、データベースでIM列ストアを使用して結果を取得したことを確認します。

```
COL NAME FORMAT a25
SELECT ul.NAME, ul.DETECTED_USAGES
FROM   DBA_FEATURE_USAGE_STATISTICS ul
WHERE  ul.VERSION= (SELECT MAX(u2.VERSION)
                    FROM   DBA_FEATURE_USAGE_STATISTICS u2
                    WHERE  u2.NAME = ul.NAME
                    AND    ul.NAME LIKE '%Column Store%');
NAME                      DETECTED_USAGES
------------------------- --------------
In-Memory Column Store    1
```

関連項目:

「インメモリー・オブジェクトの移入の優先度オプション」

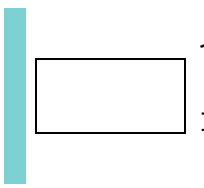
INMEMORY PRIORITY句について学習するには、『Oracle Database SQL言語リファレンス』を参照

4.1.2.2 バックグラウンド・プロセスによるIMCUの移入方法

移入中、データベースは、行形式でディスクからデータを読み取り、列を作成するために行をピボットし、インメモリー圧縮ユニット(IMCU)にデータを圧縮します。

ワーカー・プロセス(Wnnn)は、IM列ストアにデータを移入します。各ワーカー・プロセスは、オブジェクトのデータベース・ブロックのサブセットで動作します。移入は、データを同時に圧縮して列形式に変換する、ストリーミング・メカニズムです。

INMEMORY_MAX_POPULATE_SERVERS初期化パラメータにより、IM列ストアの移入に使用するワーカー・プロセスの最大数を指定します。デフォルト設定は、CPU_COUNTの2分の1です。このパラメータは環境に適した値に設定します。ワーカー・プロセスの数を増加すると移入は高速化しますが、より多くのCPUリソースが使用されます。ワーカー・プロセスの数を削減すると、移入が低速化し、これによりCPUオーバーヘッドは低減します。



ノート:

INMEMORY_MAX_POPULATE_SERVERS が 0 に設定されている場合、移入は無効化されます。

関連項目:

INMEMORY_MAX_POPULATE_SERVERS初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

4.1.3 インメモリー・オブジェクトの制御

INMEMORY句をDDL文で使用して、IM列ストアへの移入の対象となるオブジェクトを指定します。表領域、表(内部および外部)、パーティションおよびマテリアライズド・ビューを使用可能にできます。

この項では、次の項目について説明します。

4.1.3.1 INMEMORY副句

INMEMORYはセグメントレベルの属性で、列レベルの属性ではありません。ただし、INMEMORY属性を、指定したオブジェクト内の列のサブセットに適用できます。

IM列ストアに対するオブジェクトを有効化または無効化するには、表領域、表およびマテリアライズド・ビューのDDL文にINMEMORY句を指定します。DBA_TABLESビューのINMEMORY列は、どの表にINMEMORY属性が設定されている(ENABLED)か、または設定されていない(DISABLED)かを示します。

次のオブジェクトは、IM列ストアへの移入の対象ではありません。

- 索引
- 索引構成表
- ハッシュ・クラスタ
- SYSユーザーに所有されているか、SYSTEMまたはSYSAUX表領域内に格納されているオブジェクト

4.1.3.1.1 インメモリー表

ヒープ構成表を移入の対象にするには、CREATE TABLE文またはALTER TABLE文で、INMEMORYを指定します。

デフォルトでは、IM列ストアは表内のすべての非仮想列を移入します。内部表のすべての列または一部の列を指定できます。たとえば、oe.product_informationのweight_class列およびcatalog_url列を適格性から除外する場合があります。

パーティション表の場合、IM列ストア内のパーティション(すべてまたは一部)を移入できます。デフォルトでは、パーティション表内のすべてのパーティションはINMEMORY属性を継承します。内部パーティションと外部パーティションの両方を含むハイブリッド・パーティション表の場合、内部パーティションのみがINMEMORY属性を継承します。

Oracle Exadata Storage Serverでは、CELLMEMORYキーワード(デフォルト)により、フラッシュ・キャッシュはインメモリー形式でデータを格納できます。ALTER TABLEを使用して、FOR QUERYまたはFOR CAPACITY圧縮を選択できます。NO CELLMEMORYを指定すると、フラッシュ・キャッシュ内の列型記憶域が無効になります。

表をIM列ストアに対して有効化した場合に、その表に次のいずれかのタイプの列が含まれていても、これらはIM列ストアに移入されません。

- 表外格納列(VARRAY、ネストした表の列、およびアウトラインLOB)

ノート:

インライン LOB 列の場合、IM 列ストアは連続した最大 4KB のバッファ記憶域を割り当て、インライン LOB に OSON (バイナリ JSON)データが含まれている場合は最大 32KB を割り当てます。アウトライン LOB の場合、IM 列ストアはロケータに対して最大 40 バイトを割り当てますが、LOB 自体は格納しません。

- LONGまたはLONG RAWデータ型を使用する列
- 拡張データ型の列

例4-2 表のINMEMORYとしての指定

データベースにユーザーshとして接続していると仮定します。FOR QUERY LOWのデフォルト圧縮レベルを使用して、IM列ストアへの移入のためにcustomers表を有効化します。

```
SQL> SELECT TABLE_NAME, INMEMORY FROM USER_TABLES WHERE TABLE_NAME = 'CUSTOMERS';
TABLE_NAME INMEMORY
---------- --------
CUSTOMERS  DISABLED
SQL> ALTER TABLE customers INMEMORY;
Table altered.
SQL> SELECT TABLE_NAME, INMEMORY, INMEMORY_COMPRESSION FROM USER_TABLES WHERE
TABLE_NAME='CUSTOMERS';
TABLE_NAME INMEMORY INMEMORY_COMPRESS
---------- -------- -----------------
CUSTOMERS  ENABLED  FOR QUERY LOW
```

関連項目:

- 「インメモリー圧縮」
- 「データベースに対するIM列ストアの有効化」
- ALTER TABLE ... CELLMEMORYについてさらに学習するには、Oracle Exadata System Softwareユーザーズ・ガイドを参照
- CREATE TABLE文のINMEMORY句の詳細は、Oracle Database SQL言語リファレンスを参照

4.1.3.1.2 インメモリー外部表

外部表を移入の対象にするには、CREATE TABLEまたはALTER TABLEでEXTERNAL ... INMEMORY句を指定します。

インメモリー外部表の目的

インメモリー外部表は、次のような場合に役立ちます。

- 短期間に繰り返しスキャンする必要があり、Oracle Databaseに保存する必要がない短期データ
- 高速分析処理のためにリレーショナル・データに結合する必要がある外部データ
- Oracle Databaseと外部ツールの両方で分析問合せによってアクセスされ、データベース記憶域でマテリアライズする必要がないデータ

パーティション化されたインメモリー外部表の制限

IM列ストアは、ヒープ構成表の場合と同じ方法で外部表のデータを管理します。たとえば、全表スキャンでは、内部表と外部表の両方をIM列ストアに移入します。外部表でサポートされるドライバは、インメモリー外部表でもサポートされます。ただし、次の相違点に注意してください。

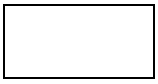
- column句、distribute句およびpriority句を含む、外部表の一部のINMEMORY副句は有効ではありません。
- インメモリー最適化算術は、外部表をサポートしていません。
- インメモリー外部表では、パーティション化はサポートされていません。内部パーティションと外部パーティションの両方を

含むハイブリッド・パーティション表の場合は、内部パーティションのみ(外部パーティションではない)がINMEMORY属性をサポートします。

- インメモリー外部表では、結合グループはサポートされていません。
- インメモリー外部表では、IM式はサポートされていません。
- インメモリー外部表は、Oracle Active Data GuardインスタンスのDISTRIBUTE ... FOR SERVICE句をサポートしていません。

ノート:

インメモリーの外部表を問い合せるセッションでは、初期化パラメータ QUERY_REWRITE_INTEGRITY を stale_tolerated に設定する必要があります。



外部表が変更された場合、IM 列ストアからの結果は未定義であることに注意してください。パーティションが(値を削除または追加して)変更された場合も、結果は未定義です。これにより、IM ベースと IM 以外のスキャンの結果に違いが生じる可能性があります。DBMS_INMEMORY.REPOPULATE を実行して IM ストアをリフレッシュし、表データと再同期化できます。

関連項目:

- 「DBMS_INMEMORY.POPULATEを使用したインメモリー外部表の移入: 例」
- CREATE TABLE ... EXTERNAL文のINMEMORY句の詳細は、Oracle Database SQL言語リファレンスを参照
- 様々なエディションとサービスでサポートされる機能の詳細は、『Oracle Databaseライセンス情報ユーザー・マニュアル』を参照

4.1.3.1.3 インメモリー・マテリアライズド・ビュー

マテリアライズド・ビューを移入の対象にするには、CREATE MATERIALIZED VIEW文またはALTER MATERIALIZED VIEW文でINMEMORYを指定します。

パーティション・マテリアライズド・ビューの場合、IM列ストア内のパーティションのすべて、またはサブセットを移入できます。

関連項目:

ALTER MATERIALIZED VIEWの構文およびセマンティクスについては、Oracle Database SQL言語リファレンスを参照してください

4.1.3.1.4 インメモリー表領域

表領域を移入の対象にするには、CREATE TABLESPACE文またはALTER TABLESPACE文で、INMEMORYを指定します。

デフォルトで、表領域内の表およびマテリアライズド・ビューはすべてIM列ストアに対して有効です。表領域内の個々の表およびマテリアライズド・ビューには、異なるINMEMORY属性がある場合があります。個々のデータベース・オブジェクトの属性は、表領域の属性をオーバーライドします。

ノート:

一時表領域はインメモリー移入の対象ではありません。

関連項目:

ALTER TABLESPACEの構文およびセマンティクスについては、『Oracle Database SQL言語リファレンス』を参照してください

4.1.3.2 インメモリー・オブジェクトの移入の優先順位オプション

IM列ストアに対してオブジェクトを有効にする場合、オブジェクトを移入するタイミングをOracle Databaseで制御することも(デフォルト)、移入キューでのオブジェクトの優先度を決定するレベルを指定することもできます。

Oracle SQLには、移入のためにキューをより細かく制御できるようにするINMEMORY PRIORITY句が組み込まれています。たとえば、あるデータベース・オブジェクトのデータを移入してから他のデータベース・オブジェクトのデータを移入することが、重要になったり重要でなくなったりすることがあります。

ビデオ:

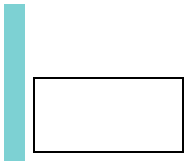
次の表では、サポートされている優先度レベルを説明します。

表4-1 IM列ストアにデータベース・オブジェクトを移入するための優先度レベル

CREATE/ALTERの構文	説明
PRIORITY NONE	データベースでは、要求された場合のみオブジェクトが移入されます。データベース・オブジェクトの全体スキャンによって、IM 列ストアへのオブジェクトの移入がトリガーされます。 これは、PRIORITY が INMEMORY 句に指定されていない場合のデフォルト・レベルです。
PRIORITY LOW	データベースにより、オブジェクトに低い優先順位が割り当てられ、キュー内のその位置に基づいて、起動後にそれが移入されます。移入には、オブジェクトがアクセスされるかどうかは関係ありません。 オブジェクトは、優先度レベルが NONE のデータベース・オブジェクトより前に、IM 列ストアに移入されます。データベース・オブジェクトのデータは、優先度レベルが MEDIUM、HIGH または CRITICAL のデータベース・オブジェクトより後に、IM 列ストアに移入されます。
PRIORITY MEDIUM	データベースにより、オブジェクトに中間の優先順位が割り当てられ、キュー内のその位置に基づいて、起動後にそれが移入されます。移入には、オブジェクトがアクセスされるかど

CREATE/ALTERの構文	説明
	うかは関係ありません。 データベース・オブジェクトは、優先度レベルが NONE または LOW のデータベース・オブジェクトより前に、IM 列ストアに移入されます。データベース・オブジェクトのデータは、優先度レベルが HIGH または CRITICAL のデータベース・オブジェクトより後に、IM 列ストアに移入されます。
PRIORITY HIGH	データベースにより、オブジェクトに高い優先順位が割り当てられ、キュー内のその位置に基づいて、起動後にそれが移入されます。移入には、オブジェクトがアクセスされるかどうかは関係ありません。 データベース・オブジェクトのデータは、優先度レベルが NONE、LOW または MEDIUM のデータベース・オブジェクトより前に、IM 列ストアに移入されます。データベース・オブジェクトのデータは、優先度レベルが CRITICAL のデータベース・オブジェクトより後に、IM 列ストアに移入されます。
PRIORITY CRITICAL	データベースにより、オブジェクトに低い優先順位が割り当てられ、キュー内のその位置に基づいて、起動後にそれが移入されます。移入には、オブジェクトがアクセスされるかどうかは関係ありません。 データベース・オブジェクトのデータは、優先度レベルが NONE、LOW、MEDIUM または HIGH のデータベース・オブジェクトより前に、IM 列ストアに移入されます。

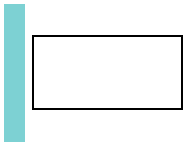
複数のデータベース・オブジェクトの優先度レベルがNONE以外である場合、Oracle Databaseでは、優先度レベルに基づいて移入するオブジェクトのデータをキューに入れます。CRITICAL優先度レベルのデータベース・オブジェクトが最初に移入され、HIGH優先度レベルのデータベース・オブジェクトが次に移入される、といったようになります。IM列ストアに領域が残っていない場合、領域が使用できるようになるまで、他のオブジェクトは移入されません。



ノート:

すべてのオブジェクトを CRITICAL として指定した場合、データベースでは、どのオブジェクトも他より重要とはみなされません。

データベースを再起動すると、優先度レベルがNONE以外のデータベース・オブジェクトのデータはすべて起動時にIM列ストアに移入されます。優先度レベルがNONE以外のデータベース・オブジェクトの場合、そのデータベース・オブジェクトが含指定されているALTER TABLEまたはALTER MATERIALIZED VIEW DDL文は、IM列ストアでDDLの変更が記録されるまで戻りません。



ノート:

- 優先度レベル設定は、表全体または表パーティションに適用する必要があります。表内の列のサブ

セットごとに異なる IM 列ストア優先度レベルを指定することはできません。

- ディスク上のセグメントが 64KB 以下の場合、IM 列ストアに移入されません。したがって、IM 列ストアに対して有効になっている小規模データベース・オブジェクトは、移入されないことがあります。

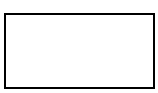
関連項目:

- 「インメモリー移入の優先順位付け」
- CREATE TABLE ... INMEMORY PRIORITYの構文およびセマンティクスについては、『Oracle Database SQL言語リファレンス』を参照

4.1.3.3 IM列ストアの圧縮方法

要件に応じて、様々なレベルでインメモリー・オブジェクトを圧縮できます。

一般には、圧縮は領域を節約するメカニズムです。ただし、IM列ストアでは、問合せパフォーマンスも向上させる新しい一連のアルゴリズムを使用して、データを圧縮できます。FOR DMLまたはFOR QUERYオプションを使用して列データが圧縮される場合、SQL問合せは、圧縮されたデータに対して直接実行します。そのため、スキャンおよびフィルタ操作は、非常に少量のデータに対して行います。データベースでは、結果セットに必要な場合のみ、データが解凍されます。


ビデオ:

V$IM_SEGMENTSおよびV$IM_COLUMN_LEVELビューでは、現在の圧縮レベルが示されます。適切なALTERコマンドを使用することで、圧縮レベルを変更できます。表が現在IM列ストアに移入されており、PRIORITY以外の表のINMEMORY属性を変更した場合は、データベースでその表がIM列ストアから除去されます。再移入の動作は、PRIORITY設定によって異なります。

次の表に、IM列ストアでサポートされているデータ圧縮方法をまとめます。

表4-2 IM列ストアの圧縮方法

CREATE/ALTERの構文	説明
NO MEMCOMPRESS	データは圧縮されません。
MEMCOMPRESS FOR DML	この方法では、DML のパフォーマンスが最適になります。 この方法では、NO MEMCOMPRESS を例外として、IM 列ストアが最も小さく圧縮されます。 ノート: この圧縮方法は、Exadata フラッシュ・キャッシュの CELLMEMORY 記憶域ではサポートされません。

CREATE/ALTERの構文	説明
MEMCOMPRESS FOR QUERY LOW	この方法は、問合せパフォーマンスが最適になります。 IM 列ストア・データの圧縮は、MEMCOMPRESS FOR DML を上回りますが、MEMCOMPRESS FOR QUERY HIGH を下回ります。 この方法は、CREATE または ALTER SQL 文に圧縮方法を指定せずに INMEMORY 句が指定されている場合または LOW または HIGH のいずれかを指定せずに MEMCOMPRESS FOR QUERY が指定されている場合のデフォルトです。 ノート: INMEMORY_FORCE 初期化パラメータに BASE_LEVEL が設定されている場合は、INMEMORY オブジェクトおよび列で QUERY LOW 圧縮が自動的に使用されます。データ・ディクショナリ・ビューには既存の圧縮設定が引き続き表示される場合がありますが、ベース・レベルでは常にオブジェクトおよび列が QUERY LOW レベルで透過的に圧縮されます。
MEMCOMPRESS FOR QUERY HIGH	この方法では、問合せパフォーマンスが向上し、領域が節約されます。 IM 列ストアデータの圧縮は、MEMCOMPRESS FOR QUERY LOW を上回りますが、MEMCOMPRESS FOR CAPACITY LOW を下回ります。
MEMCOMPRESS FOR CAPACITY LOW	この方法では、領域節約になるような方向で、領域節約と問合せパフォーマンスのバランスがとられます。 IM 列ストア・データの圧縮は、MEMCOMPRESS FOR QUERY HIGH を上回りますが、MEMCOMPRESS FOR CAPACITY HIGH を下回ります。この方法では、Oracle Zip (OZIP)という独自の圧縮技術が適用されます。これにより、Oracle Database 専用に調整されている、非常に高速な解凍が提供されます。そのデータは、スキャンするより前に解凍する必要があります。 この方法は、LOW または HIGH のいずれかを指定せずに MEMCOMPRESS FOR CAPACITY が指定されている場合のデフォルトです。
MEMCOMPRESS FOR CAPACITY HIGH	この方法では、最も優れた領域節約がもたらされます。 IM 列ストア・データを最も大きく圧縮します。

SQL文では、MEMCOMPRESSキーワードの前にINMEMORYキーワードがある必要があります。

関連項目:

- 「IM列ストアの再移入について」
- ALTER TABLE ... CELLMEMORYについてさらに学習するには、Oracle Exadata System Softwareユーザーズ・ガイドを参照
- CREATE TABLE ... INMEMORY PRIORITYの構文およびセマンティクスについては、『Oracle Database SQL言語リファレンス』を参照

4.1.3.4 Oracle Compression Advisor

Oracle Compression AdvisorではMEMCOMPRESS句を使用してユーザーが実感できる圧縮率を推定します。アドバイザはDBMS_COMPRESSIONインタフェースを使用します。

表に対してDBMS_COMPRESSION.GET_COMPRESSION_RATIOを実行すると、Oracle Databaseにより、行のサンプルが分析されます。このため、Oracle Compression Advisorでは、IM列ストアに移入された後に表が得られる圧縮結果について、適切な見積りが提供されます。

関連項目:

DBMS_COMPRESSION.GET_COMPRESSION_RATIOについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

4.2 IM列ストアに対する表の有効化および無効化

IM列ストアに対して表を有効にするには、CREATE TABLEまたはALTER TABLE文でINMEMORY句を指定します。IM列ストアに対して表を無効にするには、CREATE TABLEまたはALTER TABLE文でNO INMEMORY句を指定します。

4.2.1 インメモリー列ストアに対する新しい表の有効化

新しい表をIM列ストアに対して有効にするには、CREATE TABLE文でINMEMORY句を指定します。

IM列ストアに対して内部表または外部表のいずれかを有効にできます。列および優先度の副句を含む、一部のINMEMORY副句は、外部表に対して有効ではありません。

前提条件

IM列ストアがデータベースに対して有効になっていることを確認します。「データベースに対するIM列ストアの有効化」を参照してください。

新しい表をIM列ストアに対して有効にするには:

1. 表の作成に必要な権限があるユーザーとして、データベースにログインします。
2. INMEMORY句を指定してCREATE TABLE文を実行します。

関連項目:

- 「IM列ストアに対する表の有効化および無効化：例」
- 「IM列ストアに対する列のサブセットの有効化：例」
- CREATE TABLE文のINMEMORY句の詳細は、Oracle Database SQL言語リファレンスを参照

4.2.2 IM列ストアに対する既存の表の有効化および無効化

既存の表をIM列ストアに対して有効または無効にするには、ALTER TABLE文でINMEMORY句またはNO INMEMORY句を指定します。

前提条件

IM列ストアがデータベースに対して有効になっていることを確認します。「データベースに対するIM列ストアの有効化」を参照してください。

IM列ストアに対して既存の表を有効または無効にするには:

1. ALTER TABLE権限のあるユーザーとしてデータベースにログインします。
2. INMEMORY句またはNO INMEMORY句を指定してALTER TABLE文を実行します。
3. 必要な場合は、インメモリー・セグメントに関するメタデータ(サイズ、優先順位、圧縮レベル)を表示するために、V$IM_SEGMENTSを問い合せます。

関連項目:

- 「IM列ストアに対する表の有効化および無効化：例」
- 「IM列ストアに対する列のサブセットの有効化：例」
- ALTER TABLE文の詳細は、『Oracle Database SQL言語リファレンス』を参照
- V$IM_SEGMENTSビューの詳細は、『Oracle Databaseリファレンス』を参照

4.2.3 IM列ストアに対する表の有効化および無効化：例

次の例では、IM列ストアに対して表を有効化または無効化する方法を示します。

関連項目:

「IM列ストアに対する表の有効化および無効化：例」

4.2.3.1 インメモリー表の作成：例

この例では、test_inmem表を作成し、IM列ストアに対して有効にします。

SQL*Plusで、表を所有するユーザーとしてデータベースにログインし、次のSQL文を実行します。

```
CREATE TABLE test_inmem (
     id         NUMBER(5) PRIMARY KEY,
     test_col   VARCHAR2(15))
  INMEMORY;
```

上の文では、INMEMORY句のデフォルトであるMEMCOMPRESS FOR QUERYおよびPRIORITY NONEが使用されます。PRIORITYがNONEになっているため、データベースは自動的に表を移入しません。

4.2.3.2インメモリー・パーティションを使用した表の作成: 例

この例では、パーティションのサブセットをINMEMORYとして指定して、range_salesという名前のパーティション表を作成します。

新しい表を所有するユーザーとしてSQL*Plusにログインし、次のDDL文を実行します。

```
CREATE TABLE range_sales
    ( prod_id        NUMBER(6)
    , cust_id        NUMBER
    , time_id        DATE
    , channel_id     CHAR(1)
    , promo_id       NUMBER(6)
    , quantity_sold  NUMBER(3)
    , amount_sold    NUMBER(10,2)
    )
PARTITION BY RANGE (time_id)
  (PARTITION SALES_Q4_1999
      VALUES LESS THAN (TO_DATE('01-JAN-2015','DD-MON-YYYY'))
      INMEMORY MEMCOMPRESS FOR DML,
   PARTITION SALES_Q1_2000
      VALUES LESS THAN (TO_DATE('01-APR-2015','DD-MON-YYYY'))
      INMEMORY MEMCOMPRESS FOR QUERY,
   PARTITION SALES_Q2_2000
      VALUES LESS THAN (TO_DATE('01-JUL-2015','DD-MON-YYYY'))
      INMEMORY MEMCOMPRESS FOR CAPACITY,
   PARTITION SALES_Q3_2000
      VALUES LESS THAN (TO_DATE('01-OCT-2015','DD-MON-YYYY'))
      NO INMEMORY,
   PARTITION SALES_Q4_2000
      VALUES LESS THAN (MAXVALUE));
```

前述のSQLでは、IM列ストア内の最初の3つのパーティションに対して、異なる圧縮レベルを指定しています。最後の2つのパーティションは、IM列ストアへの移入の対象ではありません。

4.2.3.3 インメモリー外部表の作成: 例

この例では、INMEMORYオプションを使用して外部表を作成します。

この例では、ホストにディレクトリ/tmp/data/、/tmp/log/および/tmp/bad/があることを前提としています。

次のSQLスクリプトは、sh.sales表からカンマ区切りのフラット・ファイル/tmp/data/sh_sales.csvを作成します。ユーザーshとしてスクリプトを実行します。

```
SET HEAD OFF
SET TRIMSPOOL ON
SET PAGES 0
SET FEEDBACK OFF
SET TERMOUT OFF
SPOOL /tmp/data/sh_sales.csv
SELECT prod_id       || ',' || cust_id  || ',' || time_id || ',' ||
       channel_id    || ',' || promo_id || ',' ||
       quantity_sold || ',' || amount_sold
FROM   sales;
SPOOL OFF
```

sh_sales.csvファイルを使用して、次のSQLスクリプトはINMEMORYオプションで外部表sh.admin_ext_salesを作成します。

```
CONNECT  /  AS SYSDBA;
-- Set up directories and grant access to sh
CREATE OR REPLACE DIRECTORY admin_dat_dir
    AS '/tmp/data';
```

```
CREATE OR REPLACE DIRECTORY admin_log_dir
    AS '/tmp/log';
CREATE OR REPLACE DIRECTORY admin_bad_dir
    AS '/tmp/bad';
GRANT READ ON DIRECTORY admin_dat_dir TO sh;
GRANT WRITE ON DIRECTORY admin_log_dir TO sh;
GRANT WRITE ON DIRECTORY admin_bad_dir TO sh;
-- sh connects. Provide the user password (sh) when prompted.
CONNECT sh
-- create the external table
DROP TABLE admin_ext_sales;
CREATE TABLE admin_ext_sales
     (  prod_id           NUMBER,
        cust_id           NUMBER,
        time_id           DATE,
        channel_id        NUMBER,
        promo_id          NUMBER,
        quantity_sold     NUMBER(10,2),
        amount_sold       NUMBER(10,2)
     )
     ORGANIZATION EXTERNAL
     ( TYPE ORACLE_LOADER
       DEFAULT DIRECTORY admin_dat_dir
       ACCESS PARAMETERS
       ( records delimited by newline
         badfile admin_bad_dir:'empxt%a_%p.bad'
         logfile admin_log_dir:'empxt%a_%p.log'
         fields terminated by ','
         missing field values are null
         ( prod_id, cust_id,
           time_id char date_format date mask "dd-mon-yy",
           channel_id, promo_id, quantity_sold, amount_sold
         )
       )
       LOCATION ('sh_sales.csv')
     )
     REJECT LIMIT UNLIMITED
     INMEMORY;
```

ALL_EXTERNAL_TABLESの次の問合せでは、admin_ext_salesがINMEMORYに対して有効になっていることを示しています。

```
COL OWNER FORMAT A10
COL TABLE_NAME FORMAT A15
SELECT OWNER, TABLE_NAME,
       INMEMORY, INMEMORY_COMPRESSION
FROM   ALL_EXTERNAL_TABLES
WHERE  TABLE_NAME = 'ADMIN_EXT_SALES';
OWNER      TABLE_NAME      INMEMORY INMEMORY_COMPRESS
---------- --------------- -------- -----------------
SH         ADMIN_EXT_SALES ENABLED  FOR QUERY LOW
```

関連するビューには、ALL_XTERNAL_PART_TABLES、ALL_XTERNAL_TAB_PARTITIONSおよびALL_XTERNAL_TAB_SUBPARTITIONSが含まれます。

関連項目:

- 「DBMS_INMEMORY.POPULATEを使用したインメモリー外部表の移入：例」
- ALL_EXTERNAL_TABLESおよび関連する外部表のビューについて学習するには、『Oracle Database Reference』を参照

4.2.3.4 ハイブリッド外部表の作成および移入：例

この例では、INMEMORYオプションを使用してハイブリッド外部表を作成し、内部と外部の両方のパーティションに移入します。

この例では、sh.sales表が存在することを前提としています。目標は、2つの内部パーティションのあるハイブリッド・パーティション表sales_hptを作成することです。そのうちの1つはsh.salesのデータを使用し、次に外部パーティションを1つ追加します。INMEMORY属性をsales_hptに適用すると、この属性がすべてのパーティションに適用されます。

1. Linuxで一時ディレクトリを作成し、1行の売上データを含むテキスト・ファイルを作成します。

```
rm -rf /tmp/sales_data
mkdir /tmp/sales_data
echo "1002,110,19-MAR-2016,12,18,150,4800" > /tmp/sales_data/sales2016_data.txt
```

2. SQL*Plusで、管理者権限でログインして売上データ用のディレクトリ・オブジェクトを作成します。

```
CONNECT / AS SYSDBA
CREATE DIRECTORY sales_data AS '/tmp/sales_data';
GRANT READ,WRITE ON DIRECTORY sales_data TO sh;
```

3. ユーザーshとしてログインし、sales_hpt表を作成します。

```
CONNECT sh
DROP TABLE sales_hpt;
CREATE TABLE sales_hpt
  ( prod_id       NUMBER        NOT NULL,
    cust_id       NUMBER        NOT NULL,
    time_id       DATE          NOT NULL,
    channel_id    NUMBER        NOT NULL,
    promo_id      NUMBER        NOT NULL,
    quantity_sold NUMBER(10,2)  NOT NULL,
    amount_sold   NUMBER(10,2)  NOT NULL
  )
    EXTERNAL PARTITION ATTRIBUTES (
      TYPE ORACLE_LOADER
      DEFAULT DIRECTORY sales_data
       ACCESS PARAMETERS(
         FIELDS TERMINATED BY ','
         (prod_id,cust_id,time_id DATE 'dd-mm-
yyyy',channel_id,promo_id,quantity_sold,amount_sold)
       )
      REJECT LIMIT UNLIMITED
     )
    PARTITION BY RANGE (time_id)
    (
     PARTITION sales_2014 VALUES LESS THAN (TO_DATE('01-01-2015','dd-mm-yyyy')),
     PARTITION sales_2015 VALUES LESS THAN (TO_DATE('01-01-2016','dd-mm-yyyy')),
     PARTITION sales_2016 VALUES LESS THAN (TO_DATE('01-01-2017','dd-mm-yyyy'))
       EXTERNAL LOCATION ('sales2016_data.txt')
);
```

前述の文では、表にsales_2014、sales_2015およびsales_2016の3つのパーティションがあります。sales_2016のみが外部として指定されています。

4. データ・ディクショナリを問い合せて、表がハイブリッドになっていることを確認します(出力例も示します)。

```
COL TABLE_NAME FORMAT a25
SELECT TABLE_NAME, HYBRID FROM USER_TABLES WHERE HYBRID = 'YES';
TABLE_NAME                HYB
------------------------- ---
SALES_HPT                 YES
```

5. 内部パーティションのsales_2014とsales_2015に行を挿入します。

```
INSERT INTO sh.sales_hpt (SELECT * FROM sales);
INSERT INTO sh.sales_hpt
  VALUES (30, 21086, TO_DATE('2015-12-30','SYYYY-MM-DD'), 2, 999, 1, 10.19);
COMMIT;
```

前述の文の最初の部分で、すべての行がsales表から挿入されます。salesのすべての日付は2002より前です。そのため、salesのすべての行がsales_2014パーティションに挿入されます。2番目の文は、sales_2015パーティションに1行を挿入します。

6. パーティションを問い合せて、正しいデータが存在することを確認します。

```
SQL> SELECT COUNT(*) FROM sales_hpt PARTITION(sales_2014);
  COUNT(*)
----------
    918843
SQL> SELECT COUNT(*) FROM sales_hpt PARTITION(sales_2015);
  COUNT(*)
----------
         1
SQL> SELECT COUNT(*) FROM sales_hpt PARTITION(sales_2016);
  COUNT(*)
----------
         1
```

7. INMEMORY属性を表レベルで適用し、データベースに対して強制的に表をIM列ストアに移入させます。

```
ALTER TABLE sales_hpt INMEMORY;
EXEC DBMS_INMEMORY.POPULATE('SH', 'SALES_HPT');
```

8. sales_hptパーティションの移入ステータスを問い合せます。

```
COL OWNER FORMAT a3
COL SEGMENT FORMAT a18
COL PARTITION FORMAT a13
COL STATUS FORMAT a9
COL BNP FORMAT 99999
SELECT OWNER, SEGMENT_NAME SEGMENT, PARTITION_NAME PARTITION,
       IS_EXTERNAL AS EXT, POPULATE_STATUS STATUS,
       BYTES_NOT_POPULATED AS "BNP"
FROM   V$IM_SEGMENTS
WHERE  SEGMENT_NAME = 'SALES_HPT'
ORDER BY PARTITION;
OWN SEGMENT            PARTITION     EXT   STATUS       BNP
--- ------------------ ------------- ----- --------- ------
SH  SALES_HPT          SALES_2014    FALSE COMPLETED      0
SH  SALES_HPT          SALES_2015    FALSE COMPLETED      0
SH  SALES_HPT          SALES_2016    TRUE  COMPLETED      0
```

問合せによって、すべてのパーティション(内部と外部の両方)が移入されていることが示されます。

4.2.3.5 IM列ストアに対する既存の表の有効化: 例

この例では、IM列ストアに対して既存のsh.sales表を有効にします。

SQL*Plusで、shユーザーとしてデータベースにログインし、次のDDL文を実行します。

```
ALTER TABLE sales INMEMORY;
```

上の文では、INMEMORY句のデフォルトであるMEMCOMPRESS FOR QUERYおよびPRIORITY NONEが使用されます。

4.2.3.6 インメモリー圧縮をFOR CAPACITY LOWに設定する方法: 例

この例では、IM列ストアに対して既存のoe.product_information表を有効にし、圧縮方法FOR CAPACITY LOWを指定します。

SQL*Plusで、oeユーザーとしてデータベースにログインし、次のDDL文を実行します。

```
ALTER TABLE product_information
  INMEMORY
  MEMCOMPRESS FOR CAPACITY LOW;
```

上の文では、NONEのPRIORITY句にデフォルトを使用しています。次のように全表スキャンを強制的に実行して表に移入します(出力例も示します)。

```
SELECT /*+ FULL(p) NO_PARALLEL(p) */ COUNT(*)
FROM   product_information p;
  COUNT(*)
----------
       288
```

別のセッションで、管理権限を持つユーザーとしてログインし、次の問合せを実行して圧縮率を計算します(出力例も示します)。

```
COL OWNER FORMAT a5
COL SEGMENT_NAME FORMAT a19
SET PAGESIZE 50000
SELECT OWNER, SEGMENT_NAME, BYTES ORIG_SIZE,
       INMEMORY_SIZE IN_MEM_SIZE,
       ROUND (BYTES / INMEMORY_SIZE, 2) COMP_RATIO
FROM   V$IM_SEGMENTS
WHERE  SEGMENT_NAME LIKE 'P%'
ORDER BY 4;
OWNER SEGMENT_NAME         ORIG_SIZE IN_MEM_SIZE COMP_RATIO
----- ------------------- ---------- ----------- ----------
OE    PRODUCT_INFORMATION      98304     1310720        .08
```

4.2.3.7インメモリー優先度をHIGHに設定する方法:例

この例では、IM列ストアに対してoe.product_information表を有効にし、表データをIM列ストアに移入するためにPRIORITY HIGHを指定します。

SQL*Plusで、oeユーザーとしてデータベースにログインし、次のDDL文を実行します。

```
ALTER TABLE
  product_information
  INMEMORY
  PRIORITY HIGH;
```

4.2.3.8 インメモリー表の圧縮および優先順位の設定の変更: 例

この例では、FOR CAPACITY HIGH表圧縮とLOW優先順位設定を使用するように、oe.product_information表を変更します。

SQL*Plusで、管理ユーザーとしてデータベースにログインし、次の問合せを実行して、oe.product_information表の現在の優先度と圧縮設定を表示します。

```
COL OWNER FORMAT a5
COL SEGMENT_NAME FORMAT a19
SET PAGESIZE 50000
SELECT v.OWNER, v.SEGMENT_NAME, v.INMEMORY_PRIORITY,
       v.INMEMORY_COMPRESSION
FROM   V$IM_SEGMENTS v
```

```
WHERE  SEGMENT_NAME LIKE 'P%';
OWNER SEGMENT_NAME        INMEMORY INMEMORY_COMPRESS
----- ------------------- -------- ----------------
OE    PRODUCT_INFORMATION HIGH     FOR CAPACITY LOW
```

次のDDL文では、oe.product_informationが、FOR CAPACITY HIGH表圧縮およびPRIORITY LOWを使用するよう変更されます。

```
ALTER TABLE oe.product_information
  INMEMORY
  MEMCOMPRESS FOR CAPACITY HIGH
  PRIORITY LOW;
```

4.2.3.9 IM列ストアに対する表の無効化: 例

IM列ストアに対して表を無効にするには、NO INMEMORY句を指定します。

ユーザーoeとしてデータベースにログインし、次の文を実行して、IM列ストアのproduct_information表を無効にします。

```
ALTER TABLE oe.product_information NO INMEMORY;
```

V$IM_SEGMENTSビューは、IM列ストアに移入されているデータベース・オブジェクトをリストします。

4.2.3.10 Exadataスマート・フラッシュ・キャッシュでの列形式の無効化: 例

この例では、Exadataスマート・フラッシュ・キャッシュ記憶域のoe.product_informationの列形式を無効にします。

デフォルトでは、Exadataスマート・フラッシュ・キャッシュは、レベルMEMCOMPRESS FOR CAPACITY LOWを使用してデータを圧縮します。圧縮レベルを変更するか、列形式を完全に無効にするには、ALTER TABLE ... NO CELLMEMORY文を使用します。

ユーザーoeとしてデータベースにログインし、次のDDL文を実行します。

```
ALTER TABLE product_information NO CELLMEMORY;
```

4.3 インメモリー表に対する列の有効化および無効化

内部表内の個々の列にINMEMORY句を指定できます。外部表は、列レベルでのINMEMORYの指定をサポートしません。

4.3.1 INMEMORY列の有効化について

内部表の場合、インメモリー仮想列(IM仮想列)および非仮想列の両方がIM移入の対象です。外部表の場合、適格なのは非仮想列のみです。

INMEMORYオブジェクトにINMEMORY列がありません

デフォルトでは、内部オブジェクトにINMEMORY属性がある場合、このオブジェクトのすべての列が、IM列ストアへの移入の対象になります。ただし、INMEMORY表の一部の列はNO INMEMORY属性を持つように指定できます。この場合は、INMEMORY列のみが移入の対象になります。

INMEMORY属性を列のサブセットに適用するには、ALTER TABLE table_name INMEMORY ... NO INMEMORY excluded_columnsを指定します。ここで、excluded_columnsには、NO INMEMORYとして指定される列をリストします。NO INMEMORYとして指定されない列、つまり除外される列のリストにない列のみが、セグメント・レベルのINMEMORY属性を継承します。

INMEMORY表でNO INMEMORY列を指定すると、重要な結果になります。NO INMEMORY列を参照する問合せは、行スト

アを排他的に使用します。たとえば、salesが7列のINMEMORY表で、promo_id列にのみNO INMEMORY属性がある場合、SELECTリストにpromo_idが含まれる問合せは、IM列ストアではなく行ストアからデータを取得します。同様に、promo_idがSELECTリストに含まれているかどうかに関係なく、述語がpromo_idを参照する問合せはすべて、行ストアから排他的にデータを取得します。

仮想列

IM仮想列は、式を評価することによってその値が導出されること以外は、他の任意の列と同じです。事前に計算されたIM仮想列の値をIM列ストアに格納すると、問合せのパフォーマンスが向上する可能性があります。式には、同じ表からの列、制約、SQL関数およびユーザー定義PL/SQL関数(DETERMINISTICのみ)を含めることができます。IM仮想列に明示的に書き込むことはできません。

ノート:

仮想列または IM 式は、移入されたオブジェクトごとの列の上限である 1000 個に数えられます。

IM列ストアにIM仮想列を移入するには、INMEMORY_VIRTUAL_COLUMNS初期化パラメータを次のいずれかの値に設定します。

- MANUAL (デフォルト): 表がIM列ストアに対して有効になっている場合、この表で定義されているIM仮想列は、明示的にINMEMORYとして設定されていないかぎり、移入の対象ではありません。
- ENABLE: 表がIM列ストアに対して有効になっている場合、この表で定義されているすべてのIM仮想列は、明示的にNO INMEMORYとして設定されていないかぎり、移入の対象となります。

  デフォルトでは、IM列ストア内の列の圧縮レベルは、それが格納されている表またはパーティションと同じです。ただし、IM仮想列に対して異なる圧縮レベルが指定されている場合、それは指定された圧縮レベルで移入されます。

IM列ストアにIM仮想列が移入されないよう指定するには、この初期化パラメータをDISABLEに設定します。

IM仮想列およびIM式の基礎となる記憶域構造は同じです。ただし、IM式およびIM仮想列は異なるメカニズムで制御されます。

ノート:

- IM 列ストアでは、INMEMORY とマークされている表の仮想列のみが移入されます。
- IM 列ストアに IM 仮想列を移入するには、COMPATIBLE 初期化パラメータの値が 12.1.0 以上に設定されている必要があります。

関連項目:

- 「IM列ストアに対する表の有効化および無効化」
- 「IM式のインフラストラクチャ」
- 「インメモリー初期化パラメータ」
- INMEMORY句の構文およびセマンティクスについては、『Oracle Database SQL言語リファレンス』を参照

- 移入されていない列にアクセスするブログ・エントリの場合は、https://blogs.oracle.com/in-memory/what-happens-if-a-column-is-not-populated

4.3.2 IM仮想列の有効化

IM仮想列により、計算の繰返しをなくすことで、問合せのパフォーマンスが向上します。また、データベースはSIMDベクター処理などの手法を使用してIM仮想列をスキャンおよびフィルタリングできます。

前提条件

IM仮想列を有効にするには、次の条件を満たしている必要があります。

- IM列ストアがデータベースに対して有効になっている。

  「データベースに対するIM列ストアの有効化」を参照してください。

- 仮想列を含む表は内部であり、INMEMORY属性が指定されていること。

  「IM列ストアに対する表の有効化および無効化」を参照してください。

- INMEMORY_VIRTUAL_COLUMNS初期化パラメータがDISABLEに設定されていない。
- 初期化パラメータCOMPATIBLEの値が12.1.0以上に設定されている。

IM仮想列を有効にするには:

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。
2. INMEMORY_VIRTUAL_COLUMNS初期化パラメータをENABLEに設定するか、特定の仮想列をIM列ストアに対して有効にします。

例4-3 IM列ストアに対する仮想列の有効化

この例では、SYSTEMとしてデータベースにログインしています。IM列ストアは有効化されていますが、仮想列の移入は現在無効化されています。

```
SQL> SHOW PARAMETER INMEMORY_SIZE
NAME                                 TYPE        VALUE
------------------------------------ ----------- -----
inmemory_size                        big integer 200M
SQL> SHOW PARAMETER INMEMORY_VIRTUAL_COLUMNS
NAME                                 TYPE        VALUE
------------------------------------ ----------- -------
inmemory_virtual_columns             string      DISABLE
```

仮想列をhr.employees表に追加してから、その表がINMEMORYであると指定します。

```
SQL> ALTER TABLE hr.employees ADD (weekly_sal AS (ROUND(salary*12/52,2)));
Table altered.
SQL> ALTER TABLE hr.employees INMEMORY;
Table altered.
```

この段階では、weekly_salは移入の対象ではありませんが、hr.employees内の非仮想列は移入の対象となっています。次の文により、weekly_sal、およびhr.employees内の他の仮想列が移入されるようになります。

```
SQL> ALTER SYSTEM SET INMEMORY_VIRTUAL_COLUMNS=ENABLE SCOPE=BOTH;
System altered.
```

SCOPE=SPFILEを使用することもできますが、その場合、変更は次にデータベースを再起動するまで有効になりません。SCOPE=BOTHを使用すると、変更はただちに行われ、再起動は必要ありません。

例4-4 IM列ストアに対する特定のIM仮想列の有効化

この例では、INMEMORY_VIRTUAL_COLUMNS初期化パラメータがMANUALに設定され、IM列ストアにIM仮想列を明示的に追加する必要があることを前提としています。この例では、最初にhr.admin_emp表を作成します。

```
CREATE TABLE hr.admin_emp (
      empno       NUMBER(5) PRIMARY KEY,
      ename       VARCHAR2(15) NOT NULL,
      job         VARCHAR2(10),
      sal         NUMBER(7,2),
      hrly_rate   NUMBER(7,2) GENERATED ALWAYS AS (sal/2080),
      deptno      NUMBER(3) NOT NULL)
   INMEMORY;
```

この段階では、hrly_rate仮想列は、移入の対象ではありません。次の文では、その仮想列がINMEMORYとして明示的に指定されます。

```
ALTER TABLE hr.admin_emp INMEMORY(hrly_rate);
```

4.3.3 IM列ストアに対する列のサブセットの有効化: 例

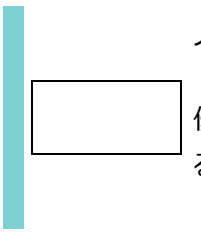
この例では、IM列ストアに対してoe.product_information表のすべての列を有効にしますが、weight_classおよびcatalog_urlは例外です。

また、次の文は、IM列ストアに対して有効な列に、別のIM列ストア圧縮方法を指定します。

```
ALTER TABLE oe.product_information
   INMEMORY MEMCOMPRESS FOR QUERY (
      product_id, product_name, category_id, supplier_id, min_price)
   INMEMORY MEMCOMPRESS FOR CAPACITY HIGH (
      product_description, warranty_period, product_status, list_price)
   NO INMEMORY (
      weight_class, catalog_url);
```

次の点に注意してください。

- product_id、product_name、category_id supplier_idおよびmin_priceの各列は、MEMCOMPRESS FOR QUERY圧縮方法を使用してIM列ストアに対して有効になります。
- product_description、warranty_period、product_statusおよびlist_priceの各列は、MEMCOMPRESS FOR CAPACITY HIGH圧縮方法を使用してIM列ストアに対して有効になります
- weight_class列およびcatalog_url列は、IM列ストアに対して有効になりません。したがって、SELECTリストと述語のいずれかでこれら2つの列を参照する問合せは、IM列ストアではなく行ストアを使用する必要があります。
- 表では、PRIORITY句のデフォルトであるPRIORITY NONEが使用されます。



ノート:

優先度レベル設定は、表全体またはパーティションに適用する必要があります。表の列のサブセットごとに異なる IM 列ストアの優先度レベルを指定することはできません。

データベース・オブジェクトに対して定義されている、選択された列圧縮レベルを判断するには、次の例で示すようにV$IM_COLUMN_LEVELビューを問い合せます。

```
COL TABLE_NAME FORMAT a20
COL COLUMN_NAME FORMAT a20
```

```
SELECT TABLE_NAME, COLUMN_NAME, INMEMORY_COMPRESSION
FROM   V$IM_COLUMN_LEVEL
WHERE  TABLE_NAME = 'PRODUCT_INFORMATION'
ORDER BY COLUMN_NAME;
TABLE_NAME           COLUMN_NAME          INMEMORY_COMPRESSION
-------------------- -------------------- -------------------------
PRODUCT_INFORMATION  CATALOG_URL          NO INMEMORY
PRODUCT_INFORMATION  CATEGORY_ID          FOR QUERY LOW
PRODUCT_INFORMATION  LIST_PRICE           FOR CAPACITY HIGH
PRODUCT_INFORMATION  MIN_PRICE            FOR QUERY LOW
PRODUCT_INFORMATION  PRODUCT_DESCRIPTION  FOR CAPACITY HIGH
PRODUCT_INFORMATION  PRODUCT_ID           FOR QUERY LOW
PRODUCT_INFORMATION  PRODUCT_NAME         FOR QUERY LOW
PRODUCT_INFORMATION  PRODUCT_STATUS       FOR CAPACITY HIGH
PRODUCT_INFORMATION  SUPPLIER_ID          FOR QUERY LOW
PRODUCT_INFORMATION  WARRANTY_PERIOD      FOR CAPACITY HIGH
PRODUCT_INFORMATION  WEIGHT_CLASS         NO INMEMORY
```

関連項目:

- 「インメモリー圧縮」
- V$IM_COLUMN_LEVELビューの詳細は、『Oracle Databaseリファレンス』を参照

4.3.4 NO INMEMORY表でのINMEMORY列属性の指定: 例

Oracle Database 12cリリース2 (12.2)以降では、まだINMEMORYとして指定されていないオブジェクト上で、列レベルでINMEMORY句を指定できます。

以前のリリースでは、列レベルのINMEMORY句は、INMEMORY表またはパーティションで指定されている場合のみ有効でした。この制限は、表またはパーティションにINMEMORY句を関連付ける前に、列にINMEMORY句を関連付けることはできないことを意味します。

Oracle Database 12cリリース2 (12.2)以降では、INMEMORY句を列レベルで指定した場合、データベースで、指定された列の属性が記録されます。表がNO INMEMORY (デフォルト)の場合、表またはパーティションがINMEMORYとして指定されるまで、列レベルの属性は、表を問い合せる方法には影響しません。表自体をNO INMEMORYとしてマークした場合は、データベースにより、既存の列レベル属性が削除されます。

この例での目的は、パーティション表内の列c3がIM列ストアに移入されないようにすることです。次のステップを実行します。

1. 次のように、パーティション表tを作成します。

```
CREATE TABLE t (c1 NUMBER, c2 NUMBER, c3 NUMBER)
  NO INMEMORY -- this clause specifies the table itself as NO INMEMORY
  PARTITION BY LIST (c1)
    ( PARTITION p1 VALUES (0),
      PARTITION p2 VALUES (1),
      PARTITION p3 VALUES (2) );
```

   表tはNO INMEMORYです。この表は、列c1上のリストによってパーティション化されており、p1、p2およびp3という3つのパーティションを含んでいます。

2. 表内の列の圧縮を問い合せます(出力例が含まれています)。

```
COL TABLE_NAME FORMAT a20
COL COLUMN_NAME FORMAT a20
SELECT TABLE_NAME, COLUMN_NAME, INMEMORY_COMPRESSION
FROM   V$IM_COLUMN_LEVEL
WHERE  TABLE_NAME = 'T'
```

```
ORDER BY COLUMN_NAME;
no rows selected
```

出力で示されているように、列レベルのINMEMORY属性は設定されていません。

3. 列c3が移入されないようにするには、NO INMEMORY属性を列c3に適用します。

```
ALTER TABLE t NO INMEMORY (c3);
```

4. 表内の列の圧縮を問い合せます(出力例が含まれています)。

```
SELECT TABLE_NAME, COLUMN_NAME, INMEMORY_COMPRESSION
FROM   V$IM_COLUMN_LEVEL
WHERE  TABLE_NAME = 'T'
ORDER BY COLUMN_NAME;
TABLE_NAME           COLUMN_NAME          INMEMORY_COMPRESSION
-------------------- -------------------- --------------------
T                    C1                   DEFAULT
T                    C2                   DEFAULT
T                    C3                   NO INMEMORY
```

データベースでは、c3のNO INMEMORY属性が記録されています。他の列では、デフォルトの圧縮が使用されます。

5. パーティションp3をINMEMORYとして指定します。

```
ALTER TABLE t
  MODIFY PARTITION p3
    INMEMORY PRIORITY CRITICAL;
```

列c3は前にNO INMEMORYとして指定されているため、パーティションp3の最初の移入には、列c3は含まれません。

6. 表全体をINMEMORYとして指定します。

```
ALTER TABLE t INMEMORY;
```

7. 表内の列の圧縮を問い合せます(出力例が含まれています)。

```
SELECT TABLE_NAME, COLUMN_NAME, INMEMORY_COMPRESSION
FROM   V$IM_COLUMN_LEVEL
WHERE  TABLE_NAME = 'T'
ORDER BY COLUMN_NAME;
TABLE_NAME           COLUMN_NAME          INMEMORY_COMPRESSION
-------------------- -------------------- --------------------------
T                    C1                   DEFAULT
T                    C2                   DEFAULT
T                    C3                   NO INMEMORY
```

データベースでは、列c3のNO INMEMORY設定が保持されています。他の列では、デフォルトの圧縮が使用されます。

8. 異なる圧縮レベルを列c1およびc2に適用します。

```
ALTER TABLE t
  INMEMORY MEMCOMPRESS FOR CAPACITY HIGH (c1)
  INMEMORY MEMCOMPRESS FOR CAPACITY LOW (c2);
```

9. 表内の列の圧縮を問い合せます(出力例が含まれています)。

```
SELECT TABLE_NAME, COLUMN_NAME, INMEMORY_COMPRESSION
FROM   V$IM_COLUMN_LEVEL
WHERE  TABLE_NAME = 'T'
ORDER BY COLUMN_NAME;
TABLE_NAME           COLUMN_NAME          INMEMORY_COMPRESSION
-------------------- -------------------- --------------------------
T                    C1                   FOR CAPACITY HIGH
```

```
T                    C2                   FOR CAPACITY LOW
T                    C3                   NO INMEMORY
```

これで、各列に異なる圧縮レベルが設定されました。

10. 表全体をNO INMEMORYとして指定します。

```
ALTER TABLE t NO INMEMORY;
```

11. 表内の列の圧縮を問い合せます(出力例が含まれています)。

```
SELECT TABLE_NAME, COLUMN_NAME, INMEMORY_COMPRESSION
FROM   V$IM_COLUMN_LEVEL
WHERE  TABLE_NAME = 'T'
ORDER BY COLUMN_NAME;
no rows selected
```

表全体がNO INMEMORYとして指定されたため、データベースにより、列レベルのすべてのINMEMORY属性が削除されました。

関連項目:

ALTER TABLEの構文およびセマンティクスは、『Oracle Database SQL言語リファレンス』を参照してください。

4.4 IM列ストアに対する表領域の有効化および無効化

IM列ストアに対する表領域を有効化または無効化できます。

INMEMORY句を指定したCREATE TABLESPACE文を使用して表領域を作成する際に、IM列ストアに対して表領域を有効にします。また、INMEMORY句を指定したALTER TABLESPACE文を使用して、IM列ストアに対して有効になるように表領域を変更できます。

IM列ストアに対して表領域を無効にするには、CREATE TABLESPACE文またはALTER TABLESPACE文にNO INMEMORY句を指定します。

IM列ストアに対して表領域が有効になっている場合、表領域内の表およびマテリアライズド・ビューはすべて、デフォルトでIM列ストアに対して有効になります。INMEMORY句は、表、マテリアライズド・ビューおよび表領域の場合と同じです。DEFAULT記憶域句は、IM列ストアに対して表領域を有効にする場合はINMEMORY句の前に、IM列ストアに対して表領域を無効にする場合はNO INMEMORY句の前に必要です。

IM列ストアに対して表領域が有効になっている場合、表領域内の個々の表およびマテリアライズド・ビューに異なるインメモリー設定を指定でき、個々のデータベース・オブジェクトの設定は表領域の設定を上書きします。たとえば、メモリーにデータを移入するために表領域がPRIORITY LOWに設定されているのに対し、表領域内の表がPRIORITY HIGHに設定されている場合、その表ではPRIORITY HIGHを使用します。

前提条件

IM列ストアがデータベースに対して有効になっていることを確認します。「データベースに対するIM列ストアの有効化」を参照してください。

IM列ストアに対する表領域を有効化または無効化にします。

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。
2. INMEMORY句またはNO INMEMORY句を指定して、CREATE TABLESPACEまたはALTER TABLESPACE文を

実行します。

例4-5 表領域の作成とIM列ストアに対する有効化

次の例では、users01表領域を作成し、IM列ストアに対して有効にします。

```
CREATE TABLESPACE users01
   DATAFILE 'users01.dbf' SIZE 40M
   ONLINE
   DEFAULT INMEMORY;
```

この例では、INMEMORY句にデフォルトを使用しています。したがって、MEMCOMPRESS FOR QUERYおよびPRIORITY NONEが使用されます。

例4-6 表領域の変更とIM列ストアに対する有効化

次の例では、IM列ストアに対して有効になるようにusers01表領域を変更し、表領域内のデータベース・オブジェクトに対してFOR CAPACITY HIGH圧縮を指定し、メモリーにデータを移入するためにPRIORITY LOWを指定します。

```
ALTER TABLESPACE users01 DEFAULT INMEMORY
   MEMCOMPRESS FOR CAPACITY HIGH
   PRIORITY LOW;
```

4.5 IM列ストアに対するマテリアライズド・ビューの有効化および無効化

IM列ストアに対してマテリアライズド・ビューを有効および無効にすることができます。

IM列ストアに対してマテリアライズド・ビューを有効にするには、CREATE MATERIALIZED VIEWまたはALTER MATERIALIZED VIEW文でINMEMORY句を使用します。IM列ストアに対してマテリアライズド・ビューを無効にするには、CREATE MATERIALIZED VIEWまたはALTER MATERIALIZED VIEW文でNO INMEMORY句を使用します。

前提条件

IM列ストアがデータベースに対して有効になっていることを確認します。「データベースに対するIM列ストアの有効化」を参照してください。

IM列ストアに対してマテリアライズド・ビューを有効または無効にするには:

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。
2. INMEMORY句またはNO INMEMORY句を指定して、CREATE MATERIALIZED VIEWまたはALTER MATERIALIZED VIEW文を実行します。

例4-7 マテリアライズド・ビューの作成とIM列ストアに対する有効化

次の文では、oe.prod_info_mvマテリアライズド・ビューを作成し、IM列ストアに対して有効にします。

```
CREATE MATERIALIZED VIEW oe.prod_info_mv INMEMORY
  AS SELECT * FROM oe.product_information;
```

この例では、INMEMORY句のデフォルトであるMEMCOMPRESS FOR QUERY LOWおよびPRIORITY NONEを使用します。

例4-8 HIGHデータ移入優先度を指定したIM列ストアに対するマテリアライズド・ビューの有効化

次の文では、IM列ストアに対してoe.prod_info_mvマテリアライズド・ビューを有効にします。

```
ALTER MATERIALIZED VIEW oe.prod_info_mv INMEMORY PRIORITY HIGH;
```

この例では、デフォルトの圧縮であるMEMCOMPRESS FOR QUERY LOWを使用します。

関連項目:

CREATEまたはALTER MATERIALIZED VIEW文についてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照してください。

5 IM列ストアの手動での移入

データベースはPRIORITY設定がNONEのインメモリー・オブジェクトを自動的に移入しません。これらのオブジェクトに移入するには、SQLまたはPL/SQLを実行する必要があります。

この章のトピックは、次のとおりです:

5.1 インメモリー・オブジェクトの手動移入について

全表スキャン、DBMS_INMEMORYプログラム・ユニット、またはDBMS_INMEMORY_ADMIN.POPULATE_WAITのいずれかを使用して、手動でオブジェクトを移入します。

PRIORITYをNONEに設定してオブジェクトを有効にし、そのオブジェクトをすぐに追加する場合は、次のセクションで説明するオプションを使用できます。

5.1.1 SELECTを使用した移入

表スキャンを強制するSELECT文を発行して、移入を開始できます。

この場合、データベースはオブジェクトの各行を読み取り、それを列形式に変換します。次の文は全表スキャンを保証しません。

```
SELECT COUNT(*) FROM object
```

これは、オプティマイザが索引のスキャンを選択するためです。したがって、次の例に示すように、SELECT COUNT(*)問合せのFULLヒントを使用して全表スキャンを強制することをお薦めします。

```
SELECT /*+ FULL(customers) NO_PARALLEL(customers) */ COUNT(*) FROM customers;
```

関連項目:

- 「全表スキャンを使用したインメモリー表の移入：例」
- FULLヒントについてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照

5.1.2 DBMS_INMEMORY.POPULATEを使用した移入

DBMS_INMEMORY.POPULATEプロシージャは、全体スキャンと同じ結果を出します。

データベースは指定されたオブジェクトの各行を読み取り、行形式から列形式に変換して、IM列ストアに移入します。次のPL/SQLブロックは、customer表の移入を開始します。

```
BEGIN
  DBMS_INMEMORY.POPULATE( schema_name => 'SH', table_name => 'CUSTOMERS');
END;
/
```

関連項目:

- 「POPULATEプロシージャを使用した表の移入：例」
- DBMS_INMEMORYパッケージについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレ

ンス』を参照

5.1.3 DBMS_INMEMORY_ADMIN.POPULATE_WAITを使用した移入

DBMS_INMEMORY_ADMIN.POPULATE_WAIT関数は、優先度が指定された優先度以上のすべてのINMEMORYオブジェクトの移入を開始し、移入のステータス値を返します。ユーザー指定の間隔は、値がコール元に戻されるまで関数が待機する最大時間を指定します。

オブジェクトが移入されるかどうかを確認する使用例には、次のものが含まれます。

- データベースが閉じられたら、管理者のみだけがデータベースにアクセスできるようにSTARTUP RESTRICTでデータベースを開き、希望するタイムアウト設定でPOPULATE_WAITを実行します。POPULATE_WAITが、タイムアウトを示す-1を返した場合は、POPULATE_WAITを再実行します。関数が0を返したときに、制限付きセッションを無効にして、管理者以外のユーザーがデータベースに問合せできるようにします。
- サービスまたはアプリケーション層の手法を使用して、データベース接続をブロックします。分析索引が存在しない場合や、アプリケーションが適切なパフォーマンスを提供するためにIM列ストアに依存している場合は、これらの手法がリソース集中型の問合せを防ぎます。

POPULATE_WAIT関数は、表名を入力として受け入れません。かわりに、ファンクションは指定されたpriority以上のPRIORITY設定(デフォルトはLOW)を使用して、すべてのINMEMORYオブジェクトの移入タスクを送信します。優先度がNONEの場合、このファンクションはすべてのINMEMORYオブジェクトの移入を開始します。POPULATE_WAITは、優先度設定のない外部表には適用しません。

関数は移入パーセンテージを入力(デフォルトは100)およびタイムアウト間隔(デフォルトは99999999秒(115.74日))としてを受け入れます。関数を実行すると、データベースは指定されたPRIORITY基準を満たすオブジェクトの移入をタイムアウト間隔内で試行し、次に移入ステータスを示す値を返します。

次の表に、POPULATE_WAITで想定される戻り値を示します。この関数は、timeoutで指定された間隔の終了前に条件が満たされた場合にのみ、値0、1、2および3を返します。たとえば、timeoutが600の場合に、600秒経過する前にメモリー不足エラーが発生した場合にのみ、この関数は1を返します。このファンクションは、データベースがリクエストされた操作を完了する前に、タイムアウト間隔が終了になった場合にのみ-1を返します。

表5-1 POPULATE_WAITの戻り値

定数	値	説明
POPULATE_TIMEOUT	-1	移入の完了を待機中に関数がタイムアウトしました。 現在の移入ジョブは、-1 が返された後もバックグラウンドで引き続き実行されます。-1 が返された後で POPULATE_TIMEOUT を再発行すると、移入が再開されます。すでに移入されているセグメントは削除されません。
POPULATE_SUCCESS	0	priority 基準を満たすすべてのオブジェクトが、指定した完了の percentage に移入されました。
POPULATE_OUT_OF_MEMORY	1	インメモリー・プールに、指定した完了の percentage の priority 基準を満たすオブジェクトを移入するための十分なメモリーがありません。

定数	値	説明
POPULATE_NO_INMEMORY_OBJECTS	2	指定した priority 基準を満たす INMEMORY オブジェクトがありません。
POPULATE_INMEMORY_SIZE_ZERO	3	インメモリー列ストアが有効になっていません。

関連項目:

- 「POPULATE_WAIT関数を使用したタイムアウトの設定: 例」
- DBMS_INMEMORY_ADMINパッケージについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

5.1.4 DBMS_INMEMORY.REPOPULATEを使用した移入

DBMS_INMEMORY.REPOPULATEは、現在IM列ストアに移入されている表、パーティションまたはサブパーティションを強制的に再移入します。

このプロシージャを現在移入されていないインメモリー・オブジェクトに使用する場合、このプロシージャはDBMS_INMEMORY.POPULATEと機能的に同等です。

関連項目:

- 「REPOPULATEプロシージャを使用したインメモリー外部表のリフレッシュ: 例」
- DBMS_INMEMORYパッケージについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照
- FULLヒントについてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照

5.2 インメモリー・オブジェクトの初期移入の強制

全表スキャン、POPULATEプロシージャ、POPULATE_WAIT関数またはREPOPULATEプロシージャを使用して、オブジェクトの移入を強制できます。

前提

このタスクでは、次のことを想定しています。

- IM列ストアが有効になっている。
- 表をインメモリー移入できるようにする。
- IM列ストアへの表の即時移入を強制的に実行する必要がある。

INMEMORY表の移入を強制的に実行するには:

1. SQL*PlusまたはSQL Developerで、管理者権限を持つユーザーとしてデータベースにログインします。
2. INMEMORY属性を表に適用します。

たとえば、次のようにsh.customersをIM移入のために有効にします。

```
ALTER TABLE sh.customers INMEMORY;
```

前述の例では、デフォルトの優先度はNONEです。

3. 必要な場合は、V$IM_SEGMENTSを問い合せて移入ステータスを確認します。

たとえば、次の文を使用します(出力例が含まれています)。

```
COL OWNER FORMAT a10;
COL NAME FORMAT a25;
COL STATUS FORMAT a10;
SELECT OWNER, SEGMENT_NAME NAME,
       POPULATE_STATUS STATUS
FROM   V$IM_SEGMENTS
WHERE  SEGMENT_NAME = 'CUSTOMERS';
no rows selected
```

上の出力では、オブジェクトがまだIM列ストアに移入されていないことが示されています。

4. 次のいずれかの方法で移入を開始します。

- FULLヒントを指定したSELECTを使用して、表のすべての行を問い合せます。

  たとえば、次の文を発行します。

  ```
  SELECT /*+ FULL(customers) NO_PARALLEL(customers) */ COUNT(*) FROM
  sh.customers;
  ```

- DBMS_INMEMORY.POPULATEプロシージャを実行します。

  たとえば、このプロシージャを次のように、sh.customersに対して実行します。

  ```
  EXEC DBMS_INMEMORY.POPULATE('SH', 'CUSTOMERS');
  ```

- DBMS_INMEMORY.REPOPULATEプロシージャを実行します。

  移入されていない表の場合、このプロシージャはPOPULATEと機能的に同じです。たとえば、このプロシージャを次のように、sh.customersに対して実行します。

  ```
  EXEC DBMS_INMEMORY.REPOPULATE('SH', 'CUSTOMERS');
  ```

- DBMS_INMEMORY_ADMIN.POPULATE_WAIT関数を実行します。

  次のコード例では、PRIORITY設定に関係なく、すべてのINMEMORYオブジェクトを移入します。例では、すべてのオブジェクトが100%移入されるまで関数が待機するように指定します。また、1800秒(30分)以内に正常終了しなかった場合はエラーでタイムアウトする必要があります。

  ```
  VARIABLE b_pop_status NUMBER
  BEGIN
     SELECT DBMS_INMEMORY_ADMIN.POPULATE_WAIT(
               priority   => 'NONE' ,
               percentage => 100    ,
               timeout    => 1800   ,
               force      => FALSE  )
       INTO :b_pop_status
     FROM   dual;
  END;
  /
  PRINT b_pop_status
  ```

5. 必要な場合は、移入ステータスを確認するために、V$IM_SEGMENTSを問い合せます。

   たとえば、次の文を使用します(出力例が含まれています)。

```
SELECT OWNER, SEGMENT_NAME NAME,
       POPULATE_STATUS STATUS
FROM   V$IM_SEGMENTS
WHERE  SEGMENT_NAME = 'CUSTOMERS';
OWN NAME        STATUS
--- ---------- --------------------
SH  CUSTOMERS  COMPLETED
```

   これで、表がIM列ストアに移入されました。

関連項目:

- V$IM_SEGMENTSについて学習するには、『Oracle Databaseリファレンス』を参照
- DBMS_INMEMORYサブプログラムについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照

5.3 インメモリー表の手動での移入: 例

次の例は、インメモリー表を手動で移入する方法を示しています。

関連項目:

「IM列ストアに対する表の有効化および無効化: 例」

5.3.1 全表スキャンを使用したインメモリー表の移入: 例

この例では、全表スキャンを使用してsh.sales表をIM列ストアに移入します。

管理者としてデータベースにログインし、次のDDL文を発行してsh.sales表にINMEMORY句を追加したと仮定します。

```
ALTER TABLE sh.sales INMEMORY;
```

上の文では、INMEMORY句のデフォルトであるMEMCOMPRESS FOR QUERYおよびPRIORITY NONEが使用されます。PRIORITYがNONEであるため、データベースは表をIM列ストアに自動的に移入しません。次の問合せは、sh.sales表が現在移入されていないことを確認します。

```
COL OWNER FORMAT a10;
COL NAME FORMAT a25;
COL STATUS FORMAT a10;
SELECT OWNER, SEGMENT_NAME NAME,
       POPULATE_STATUS STATUS
FROM   V$IM_SEGMENTS
WHERE  SEGMENT_NAME = 'SALES';
no rows selected
```

次の問合せはFULLヒントを使用してsalesの全表スキャンを強制し、それによって移入を開始します(出力例を含む)。

```
SELECT /*+ FULL(sales) NO_PARALLEL(sales) */ COUNT(*) FROM sh.sales;
  COUNT(*)
----------
    918843
```

次の問合せでは、salesの移入ステータスが示されます(出力例が含まれています)。

```
SET PAGESIZE 50000
COL OWNER FORMAT a3
COL NAME FORMAT a10
COL STATUS FORMAT a20
SELECT OWNER, SEGMENT_NAME NAME,
       POPULATE_STATUS STATUS
FROM   V$IM_SEGMENTS
WHERE  SEGMENT_NAME = 'SALES';
OWN NAME       STATUS
--- ---------- --------------------
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
SH  SALES      COMPLETED
16 rows selected.
```

次の問合せは表の圧縮率を計算します。この問合せは、表がディスク上でさらに圧縮されていないことが前提となります。

```
COL OWNER FORMAT a5
COL SEGMENT_NAME FORMAT a5
SET PAGESIZE 50000
SELECT v.OWNER, v.SEGMENT_NAME, v.BYTES ORIG_SIZE,
       v.INMEMORY_SIZE IN_MEM_SIZE,
       ROUND(v.BYTES / v.INMEMORY_SIZE, 2) COMP_RATIO
FROM   V$IM_SEGMENTS v
ORDER BY 4;
OWNER SEGME  ORIG_SIZE IN_MEM_SIZE COMP_RATIO
----- ----- ---------- ----------- ----------
SH    SALES     851968     1310720        .65
SH    SALES     835584     1310720        .64
SH    SALES     925696     1310720        .71
SH    SALES     958464     1310720        .73
SH    SALES     950272     1310720        .73
SH    SALES     786432     1310720         .6
SH    SALES     876544     1310720        .67
SH    SALES     753664     1310720        .58
SH    SALES    1081344     1310720        .83
SH    SALES     901120     1310720        .69
SH    SALES     925696     1310720        .71
SH    SALES     933888     1310720        .71
SH    SALES     843776     1310720        .64
SH    SALES     999424     1310720        .76
SH    SALES     581632     1507328        .39
SH    SALES     696320     1507328        .46
16 rows selected.
```

5.3.2 OPULATEプロシージャを使用した表の移入：例

この例ではDBMS_INMEMORY.POPULATEを使用して、sh.customers表のIM列ストアへの移入を開始します。

管理者としてデータベースにログインし、次のDDL文を発行してsh.customers表にINMEMORY句を追加したと仮定します。

```
ALTER TABLE sh.customers INMEMORY;
```

上の文では、INMEMORY句のデフォルトであるMEMCOMPRESS FOR QUERYおよびPRIORITY NONEが使用されます。PRIORITYがNONEであるため、データベースは表をIM列ストアに自動的に移入しません。次の問合せは、sh.customers表が現在移入されていないことを確認します。

```
COL OWNER FORMAT a10;
COL NAME FORMAT a25;
COL STATUS FORMAT a10;
SELECT OWNER, SEGMENT_NAME NAME,
       POPULATE_STATUS STATUS
FROM   V$IM_SEGMENTS
WHERE  SEGMENT_NAME = 'CUSTOMERS';
no rows selected
```

次のPL/SQLコードはイニシアティブの移入にPOPULATEプロシージャを使用します。

```
EXEC DBMS_INMEMORY.POPULATE('SH', 'CUSTOMERS');
```

次の問合せでは、customersの移入ステータスが示されます(出力例が含まれています)。

```
SELECT OWNER, SEGMENT_NAME NAME,
       POPULATE_STATUS STATUS
FROM   V$IM_SEGMENTS
WHERE  SEGMENT_NAME = 'CUSTOMERS';
OWN NAME       STATUS
--- ---------- --------------------
SH  CUSTOMERS  COMPLETED
```

5.3.3 POPULATE_WAIT関数を使用したタイムアウトの設定: 例

この例では、優先度設定に関係なく、DBMS_INMEMORY_ADMIN.POPULATE_WAITを使用してすべてのインメモリー表を移入します。

例5-1 インメモリー移入のタイムアウト間隔の指定

この例では、データベースに様々な優先順位設定のあるインメモリー表が数多く含まれています。目標は、制限付きデータベース・セッションですべてのインメモリー表に100%の完了を移入し、制限付きセッションを無効にして、アプリケーションがインメモリー表現のみの問合せを行うことを保証できるようにすることです。

データベースは停止されていると仮定します。SQL*Plusでアイドル・インスタンスにSYSDBAとして接続し、次のコマンドを実行します(出力例を含む)。

```
SQL> STARTUP RESTRICT
ORACLE instance started.
Total System Global Area 1157624280 bytes
Fixed Size                  8839640 bytes
Variable Size             754974720 bytes
Database Buffers           16777216 bytes
Redo Buffers                7933952 bytes
In-Memory Area            369098752 bytes
Database mounted.
Database opened.
```

データベースが開きますが、アクセスできるのは管理ユーザーのみです。SQL*Plusで次の文を実行します(出力例は太字で表示)。

```
VARIABLE b_pop_status NUMBER
```

```
SELECT DBMS_INMEMORY_ADMIN.POPULATE_WAIT(
         priority   => 'NONE' ,
         percentage => 100    ,
         timeout    => 300    )
  INTO b_pop_status
FROM   DUAL;
PRINT b_pop_status
-1
```

5分後、関数が–1を返します。このコードは、移入か完了するのを待機している間に関数がタイムアウトしたことを示します。5分は、すべてのINMEMORY表を移入するのに十分な長さではありません。タイムアウトを30分に指定して、SELECT文を再実行します。

```
SELECT DBMS_INMEMORY_ADMIN.POPULATE_WAIT(
         priority   => 'NONE' ,
         percentage => 100    ,
         timeout    => 1800   )
  INTO b_pop_status
FROM   DUAL;
PRINT b_pop_status
0
```

8分後、この関数は数値0を返します。このコードは、すべての表が完全に移入されていることを示します。制限付きセッションを無効にします。これによりアプリケーションは、インメモリー表現のみにアクセスするという完全な信頼度でインメモリー・オブジェクトの問合せを開始できます。

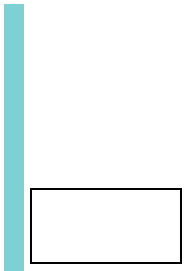
```
ALTER SYSTEM DISABLE RESTRICTED SESSION;
```

5.3.4 DBMS_INMEMORY.POPULATEを使用したインメモリー外部表への移入: 例

この例では、INMEMORYオプションを持つ外部表を移入します。

この例では、sh_sales.csvファイルを使用して、INMEMORYオプションを指定した外部表sh.admin_ext_salesを作成していると仮定します。

admin_ext_sales表はまだ移入されていません。Oracle Database19c以降、全表スキャンでは標準表への移入と同様に外部表が移入されます。ただし、このシナリオでは、DBMS_INMEMORYパッケージを使用して移入の開始を選択します。



ノート:

インメモリーの外部表を問い合せるセッションでは、初期化パラメータ QUERY_REWRITE_INTEGRITY を stale_tolerated に設定する必要があります。

外部表が変更された場合、IM 列ストアからの結果は未定義であることに注意してください。パーティションが(値を削除または追加して)変更された場合も、結果は未定義です。これにより、IM ベースと IM 以外のスキャンの結果に違いが生じる可能性があります。DBMS_INMEMORY.REPOPULATE を実行して IM ストアをリフレッシュし、表データと再同期化できます。

表に移入するには、次のステップを実行します。

1. ユーザーshとしてログインします。

2. データベース・セッションでQUERY_REWRITE_INTEGRITYを設定します。

```
ALTER SESSION SET QUERY_REWRITE_INTEGRITY=stale_tolerated;
```

3. 次のPL/SQLプログラムを実行します。

```
EXEC DBMS_INMEMORY.POPULATE('SH', 'ADMIN_EXT_SALES');
```

4. V$IM_SEGMENTSを問い合せて、移入ステータスを確認します。

```
COL OWNER FORMAT a3
COL NAME FORMAT a15
COL STATUS FORMAT a9
SELECT OWNER, SEGMENT_NAME NAME,
       POPULATE_STATUS STATUS
FROM   V$IM_SEGMENTS;
OWN NAME            STATUS
--- --------------- ---------
SH  ADMIN_EXT_SALES COMPLETED
```

前述の出力は、admin_ext_sales表が移入されたことを示しています

関連項目:

- 「インメモリー・ビュー」
- DBMS_INMEMORY.POPULATEについてさらに学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照
- QUERY_REWRITE_INTEGRITY初期化パラメータについて学習するには、『Oracle Databaseリファレンス』を参照

5.3.5 REPOPULATEプロシージャを使用したインメモリー外部表のリフレッシュ: 例

この例では、現在移入されているインメモリー外部表を再移入します。

この例では、カンマ区切りのフラット・ファイル/tmp/data/sh_sales.csvを使用して、INMEMORYオプションを指定してsh.admin_ext_sales表を作成し、この表をIM列ストアに移入したと仮定します。
次のように、レコードを/tmp/data/sh_sales.csvに追加したとします。

```
echo "148,8787,23-NOV-01,2,999,1,23.43" >> /tmp/data/sh_sales.csv
```

内部表とは異なり、外部表は自動再移入メカニズムを使用しません。外部表をリフレッシュするには、次の方法のいずれかを使用する必要があります。

- DBMS_INMEMORY.REPOPULATEプロシージャをコールします
- 表をNO INMEMORYとして指定し、それをINMEMORYとして指定して、全表スキャンを実行します

次の例ではREPOPULATEプロシージャを使用して、IM列ストアにadmin_ext_salesを強制的にリフレッシュさせます。

```
EXEC DBMS_INMEMORY.REPOPULATE('SH', 'ADMIN_EXT_SALES');
```

関連項目:

- 「インメモリー外部表の作成: 例」

- 「IM列ストアの再移入の制御」
- DBMS_INMEMORY.REPOPULATEについてさらに学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

6 インメモリー・オブジェクトの管理の自動化

自動データ最適化(ADO)および自動インメモリーを使用してIM列ストア内のオブジェクトを動的に管理できます。

ADOは、ブロックとセグメントのデータ・アクセス・パターンを追跡するヒート・マップを使用します。ADOおよびヒート・マップは、情報ライフサイクル管理(ILM)の一部であり、作成からアーカイブまたは削除までのデータを管理する一連のプロセスおよびポリシーです。この章は、ILM、ADOおよびヒート・マップをよく理解していることが前提となっています。

この章のトピックは、次のとおりです:

関連項目:

ILM、ADOおよびヒート・マップの概要は、Oracle Database VLDBおよびパーティショニング・ガイドを参照

6.1 IM列ストアに対するADOの有効化

自動データ最適化(ADO)では、ILM計画を実装するために、ポリシーが作成され、それらのポリシーに基づいてアクションが自動化されます。

この項では、次の項目について説明します。

6.1.1 ADOポリシーとIM列ストアについて

ADOは、**ADOポリシー**を使用してIM列ストアを管理します。セグメント・レベルで`INMEMORY`句を使用することによってのみ、ADOポリシーを作成できます。

データベースでは、ADOポリシーは、オブジェクトの属性のように扱われます。ADOポリシーは、インスタンス・レベルではなくデータベース・レベルとなります。Oracle Databaseでは、Database In-Memoryのために次のタイプのADOポリシーがサポートされています。

- `INMEMORY`ポリシー

  このポリシーは、オブジェクトを`INMEMORY`属性でマークし、それらをIM列ストアへの移入のために有効にします。表レベルで設定すると、`INMEMORY`属性は内部パーティションにのみ適用されます。そのため、ハイブリッド・パーティション表の外部パーティションはポリシーによって管理されません。

- 再圧縮ポリシー

  このポリシーは、`INMEMORY`オブジェクトに対する圧縮レベルを変更します。

- `NO INMEMORY`ポリシー

  このポリシーは、オブジェクトをIM列ストアから削除し、その`INMEMORY`属性を削除します。

Oracle Databaseでは、次の条件をサポートして、ポリシーを適用する時期を決定します。

- オブジェクトが変更されてからの指定日数

  `DBA_HEAT_MAP_SEGMENT`ビュー内の列`SEGMENT_WRITE_TIME`からこの値を取得します。

- オブジェクトがアクセスされてからの指定日数

  この値は、`DBA_HEAT_MAP_SEGMENT`ビューの列`SEGMENT_WRITE_TIME`、`FULL_SCAN`および

LOOKUP_SCAN内の大きいほうの値となります。

- オブジェクトが作成されてからの指定日数

  DBA_OBJECTS内のCREATED列からこの値を取得します。

- ブール値を返すユーザー定義関数

関連項目:

- DBA_HEAT_MAP_SEGMENTビューについて学習するには、『Oracle Databaseリファレンス』を参照
- INMEMORY句について学習するには、『Oracle Database SQL言語リファレンス』を参照

6.1.2 ADOとIM列ストアの目的

ADOは、IM列ストアを新しいデータ層として管理します。

ポリシーを作成して、IM列ストアからオブジェクトがあまり頻繁にアクセスされない場合にはそのオブジェクトを除去し、オブジェクトが頻繁にアクセスされる場合にはそのオブジェクトを移入することで、問合せのパフォーマンスを向上することができます。ADOでは、ヒート・マップ統計を使用してIM列ストアが管理されます。

INMEMORYポリシーの目的

多くのデータベースでは、セグメントは、作成後に大幅に変更されます。パフォーマンスを最大限に高めるには、ADOで、書込みアクティビティがおさまったときに、これらのセグメントをIM列ストアに移入します。たとえば、パーティションを表に毎日追加する場合は、作成翌日にsales_2016_d100パーティションを移入するポリシーを作成できます。

```
ALTER TABLE sales MODIFY PARTITION sales_2016_d100
  ILM ADD POLICY SET INMEMORY MEMCOMPRESS FOR QUERY
    PRIORITY HIGH
  AFTER 1 DAYS OF CREATION
```

同様に、作成の2か月後に表に対する書込みアクティビティがおさまることが判明しており、この時間条件が満たされたときにこのオブジェクトを移入する必要があるとします。

```
ALTER TABLE 2016_ski_sales
  ILM ADD POLICY SET INMEMORY MEMCOMPRESS FOR QUERY
    PRIORITY CRITICAL
  AFTER 60 DAYS OF CREATION
```

前述のポリシーでは、2016_ski_sales表のすべての既存および新規パーティションにポリシーを継承させます。セグメントがポリシーの対象となると、データベースにより、すべてのパーティションが、指定されたINMEMORY句で個別にマークされます。セグメントにすでにINMEMORYポリシーがある場合、データベースでは新しいポリシーは無視されます。

再圧縮ポリシーの目的

アクセス・パターンに基づいてIM列ストア内のデータを圧縮する必要がある場合があります。たとえば、セグメントに対するDMLアクティビティが終了した2日後に、セグメントをDML圧縮から問合せ圧縮に変更する必要があるとします。

```
ALTER TABLE lineorders
  ILM ADD POLICY MODIFY INMEMORY MEMCOMPRESS FOR QUERY HIGH
  AFTER 2 DAYS OF NO MODIFICATION
```

オブジェクトがIM列ストアに移入されていない場合、このポリシーでは、圧縮属性のみが変更されます。オブジェクトがIM列ストアに移入されている場合は、ADOにより、新しい圧縮レベルを使用してオブジェクトが再移入されます。セグメントにまだ

INMEMORY属性がない場合、データベースでは、ポリシーは無視されます。

NO INMEMORYポリシーの目的

IM列ストア内の領域を最適化するために、NO INMEMORYポリシーを使用して非アクティブなセグメントを除去できます。このポリシーは、不定期の問合せによる非アクティブなセグメントの移入を防止するためにも役立ちます。たとえば、特定のsalesパーティションに関するレポートが年内に頻繁に実行されるが、通常は毎週ではない場合に、このパーティションを、アクセスがなくなってから1週間後に除去する必要があるとします。

```
ALTER TABLE sales MODIFY PARTITION sales_2015_q1
  ILM ADD POLICY NO INMEMORY AFTER 7 DAYS OF NO ACCESS;
```

1998年用のsales表に問い合せることがほとんどない場合に、アクセスがなくなった翌日に除去する必要があるとします。

```
ALTER TABLE sales_1998
  ILM ADD POLICY NO INMEMORY AFTER 1 DAYS OF NO ACCESS;
```

除去されたセグメントの問合せがブロックされることはありません。データベースでは、常に、従来のバッファ・キャッシュ・メカニズムを使用してデータにアクセスできます。

関連項目:

- 「データベース・バッファ・キャッシュ内の行データ」
- 『Oracle Database VLDBおよびパーティショニング・ガイド』

6.1.3 ADOはどのように列データと連携するか

ADOの視点からは、IM列ストアは別の記憶域層となります。

6.1.3.1 ヒート・マップはどのように機能するか

有効になっている場合は、ヒート・マップにより、自動的にデータ・アクセス・パターンが検出されます。ADOでは、ヒート・マップ・データを使用して、ユーザーが定義したポリシーがデータベース・レベルで実装されます。

ヒート・マップでは、自動的に、行レベルおよびセグメント・レベルで使用情報が追跡されます。行レベルでは、ヒート・マップにより、データ変更回数が追跡されてから、これらの回数がブロック・レベルに集計されます。セグメント・レベルでは、ヒート・マップにより、変更、表の全体スキャンおよび索引参照の回数が追跡されます。

IM列ストアが有効になっている場合は、ヒート・マップにより、列データに対するアクセス・パターンが追跡されます。たとえば、sales表がホットであるのに対し、locations表がコールドだとします。ADOアルゴリズムは、行ベースのデータに対してと同じ方法で列データに対して機能します。

データベースでは、定期的にヒート・マップ・データがデータ・ディクショナリに書き込まれます。データベースにより、ヒート・マップ・データがデータ・ディクショナリ・ビューで公開されます。たとえば、インメモリー・オブジェクトに対する読取り回数と書込み回数を取得するには、ALL_HEAT_MAP_SEGMENTビューを問い合せます。

関連項目:

- ヒート・マップについてさらに学習するには、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照
- ALL_HEAT_MAP_SEGMENTビューについて学習するには、『Oracle Databaseリファレンス』を参照

6.1.3.2 ポリシー評価はどのように機能するか

IM列ストア・ポリシーのポリシー評価では、他のADOポリシーの評価と同じインフラストラクチャが使用されます。データベースでは、メンテナンス・ウィンドウの間に自動的にポリシーが評価および実行されます。

データベースでは、データ・ディクショナリに格納されているヒート・マップ統計を使用してポリシーが評価されます。INMEMORY属性の設定は、大部分はメタデータ操作となります。そのため、パフォーマンスへの影響は最小限となります。

ADOでは、ジョブ・スケジューラを使用して移入が実行されます。インメモリー・コーディネータ・プロセス(IMCO)により、移入が実行されます。

関連トピック

- インメモリー・コーディネータ・プロセス(IMCO)

6.1.4 ADOとIM列ストアの制御

HEAT_MAP初期化パラメータを使用してヒート・マップを有効にします。SQLおよびPL/SQLインタフェースを使用してADOを制御します。

DDL文のILM句

インメモリー・ポリシーの作成には新しいSQL文は必要ありませんが、ILM句には新しいオプションがあります。次の表では、ADOとIM列ストアのためのSQLオプションを説明します。

ノート:

INMEMORY 属性は、ハイブリッド・パーティション表の内部パーティションにのみ適用されます。

表6-1 ADOとIM列ストアのためのILM句

句	説明	例
SET INMEMORY	オブジェクトに対して INMEMORY 属性を設定します。	ALTER TABLE sh.sales ILM ADD POLICY SET INMEMORY MEMCOMPRESS FOR QUERY LOW PRIORITY HIGH SEGMENT AFTER 30 DAYS OF CREATION;
MODIFY INMEMORY	オブジェクトに対する圧縮レベルを変更します。	ALTER TABLE sh.customers ILM ADD POLICY MODIFY INMEMORY MEMCOMPRESS FOR QUERY HIGH PRIORITY CRITICAL SEGMENT AFTER 30 DAYS OF CREATION;
NO INMEMORY	オブジェクトに対して NO INMEMORY 属性を設定します。	ALTER TABLE sh.products ILM ADD POLICY NO INMEMORY SEGMENT AFTER 30 DAYS OF CREATION;

関連項目:

CREATE TABLEのilm_policy_clauseについてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照

初期化パラメータ

次の表では、ADOとIM列ストアに関連する初期化パラメータを説明します。

表6-2 ADOとIM列ストアのための初期化パラメータ

初期化パラメータ	説明
COMPATIBLE	互換性を維持する必要があるデータベースのリリースを指定します。ADO で IM 列ストアを管理する場合は、このパラメータを 12.2.0 以上に設定します。
HEAT_MAP	ヒート・マップ機能と ADO 機能を両方とも有効にします。ADO で IM 列ストアを管理する場合は、このパラメータを ON に設定します。
INMEMORY_SIZE	IM 列ストアを有効にします。このパラメータは、ゼロ以外の値に設定する必要があります。

PL/SQLパッケージ

次の表では、ADOとIM列ストアに関連するPL/SQLパッケージを説明します。

表6-3 ADOとIM列ストアのためのPL/SQLパッケージ

パッケージ	説明
DBMS_HEAT_MAP	表領域、セグメント、オブジェクト、エクステントおよびブロック・レベルでの詳細なヒート・マップ・データを表示します。
DBMS_ILM	ADO ポリシーを使用して ILM 計画を実装します。
DBMS_ILM_ADMIN	ADO ポリシーの実行をカスタマイズします。

関連項目:

DBMS_HEAT_MAPパッケージ、DBMS_ILMパッケージおよびDBMS_ILM_ADMINパッケージについてさらに学習するには、『*Oracle Database PL/SQL*パッケージおよびタイプ・リファレンス』を参照

V$およびデータ・ディクショナリ・ビュー

次の表では、ADOとIM列ストアに関連するビューを説明します。

表6-4 ADOとIM列ストアのためのビュー

ビュー	説明
DBA_HEAT_MAP_SEG_HISTOGRAM	ユーザーが参照できるすべてのセグメントのセグメント・アクセス情報を表示します。
DBA_HEAT_MAP_SEGMENT	ユーザーが参照できるすべてのセグメントの最新のセグメント・アクセス時刻を表示します。
DBA_HEATMAP_TOP_OBJECTS	デフォルトでは、オブジェクトの上位 10000 個についてヒート・マップ情報を示します。

ビュー	説明
DBA_HEATMAP_TOP_TABLESPACES	上位 10000 個の表領域についてヒート・マップ情報を示します。
DBA_ILMDATAMOVEMENTPOLICIES	データベース内の ADO ポリシーのデータ移動関連属性固有の情報を表示します。action_type 列では、IM 列ストアに関連するポリシーを説明します。使用可能な値は、COMPRESSION、STORAGE、EVICT および ANNOTATE です。
V$HEAT_MAP_SEGMENT	リアルタイムのセグメント・アクセス情報を表示します。

関連項目:

ビューについてさらに学習するには、『Oracle Databaseリファレンス』を参照

6.1.5 IM列ストアのためのADOポリシーの作成

ADOポリシーを使用することで、ヒート・マップ統計に基づいた、オブジェクトに対するINMEMORY句の設定、変更または削除が可能です。

ADO IM列ストア・ポリシーを作成するには、ALTER TABLE文でILM ADD POLICY句を指定し、その後に、次のいずれかの副句を指定します。

- SET INMEMORY ... SEGMENT

  このオプションは、DMLアクティビティがおさまっているときのみセグメントをINMEMORY属性でマークする必要がある場合に役立ちます。

- MODIFY INMEMORY ... MEMCOMPRESS ... SEGMENT

  圧縮されていないデータまたはMEMCOMPRESS FOR DMLレベルでのデータの格納は、それが頻繁に変更されるときに適しています。別の圧縮レベルは、問合せに適しています。セグメントに対するアクティビティの大部分が書込みから読取りに推移した場合は、MODIFY句を使用して異なる圧縮方法を適用できます。

- NO INMEMORY ... SEGMENT

  このオプションは、セグメントへのアクセスが時間とともに減少する(コールドになる)場合や、ランダム・アクセスの結果としてこのセグメントの移入を防ぐために役立ちます。

前提条件

ADO IM列ストア・ポリシーを使用する前に、次の前提条件を満たしている必要があります。

- INMEMORY_SIZE初期化パラメータにゼロ以外の値を設定し、データベースを再起動することにより、データベースのIM列ストアを有効化します。
- HEAT_MAP初期化パラメータをONに設定する必要があります。

  ヒート・マップは、セグメント・レベルのデータ・アクセス・トラッキングおよびセグメントおよび行レベルのデータ変更トラッキングを提供します。

- COMPATIBLE初期化パラメータは12.2.0以上に設定されている必要があります。

ADOポリシーを作成するには:

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。
2. ALTER TABLE文をILM ADD POLICY ... INMEMORY句とともに使用します。

例6-1 除去ポリシーの作成

この例では、3日間アクセスされなかったoe.order_items表がIM列ストアから除去されるよう指定するポリシーを作成します。ADO IM列ストア・ポリシーは、セグメント・レベルのポリシーである必要があります。

```
ALTER TABLE oe.order_items ILM ADD POLICY
   NO INMEMORY SEGMENT
   AFTER 3 DAYS OF NO ACCESS;
```

例6-2 DBMS_ILMの使用によるILMポリシーの実行

ポリシーを手動で評価および実行することもできます。したがって、オブジェクトを圧縮および階層化する必要がある時期をプログラムで決定できます。次の例では、sh.salesのためのADOタスクを手動で実行します。

```
DECLARE
  v_executonid NUMBER;
BEGIN
  DBMS_ILM.EXECUTE_ILM ( owner           => 'SH',
                         object_name     => 'SALES',
                         execution_mode  => DBMS_ILM.ILM_EXECUTION_OFFLINE,
                         task_id         => v_executionid);
END;
/
```

関連項目:

- CREATE TABLEの構文およびセマンティクスについては、『Oracle Database SQL言語リファレンス』を参照
- DBMS_ILMパッケージについてさらに学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照

6.2 自動インメモリーの構成

自動インメモリーは、アクセス・トラッキングと列統計を使用して、IM列ストア内のオブジェクトを管理します。

IM列ストアがいっぱいで、より頻繁にアクセスされる他のセグメントがIM列ストア内の移入による利益を得られる場合、IM列ストアは非アクティブなセグメントを除去します。ただし、IM列ストアがすべてのINMEMORYセグメントを保持するように構成されている場合、自動インメモリーは何も行いません。

ノート:

INMEMORY_FORCE 初期化パラメータに BASE_LEVEL が設定されている場合は、INMEMORY_AUTOMATIC_LEVEL が設定されていても自動インメモリーは無効になります。表の圧縮レベルが AUTO の場合でも、自動インメモリー・バックグラウンド操作は実行されません。

この章のトピックは、次のとおりです:

6.2.1 自動インメモリーの目的

作業データ・セットが常に移入されるように、自動インメモリーは自動的にコールド(アクセス頻度の低い)セグメントを除去します。

IM列ストアは、それが削除または移動された場合、INMEMORYオプションが削除された場合、またはIM ADOポリシーが適用された場合にのみ、移入されたセグメントを削除します。メモリー不足は、INMEMORYデータ・セットのサイズがIM列ストアの使用可能メモリーを超過し、移入された一部のセグメントが非アクティブになったときに発生します。最適パフォーマンスを得るために、IM列ストアは、最も頻繁に問合せされるセグメント(作業データ・セットとも呼ばれます)を含む必要があります。

通常、作業データ・セットは多くのアプリケーションで時間とともに変化します。したがって、最適化には、定期的な監視とDBAによる手動介入が必要です。これにより、IMストア要素を除去したり、ADO IMポリシーを作成できます。どちらのタスクでもワークロードをよく理解する必要があります。

コールド・セグメントを自動的に除去することで、自動インメモリーには次の利点があります。

- パフォーマンスの向上

  コールド・セグメントの除去によりメモリー不足を緩和することで、自動インメモリーでは作業データ・セットがIM列ストア内に存在しているという理由でワークロードのパフォーマンスが向上します。

- 管理が容易

  コールド・セグメントの除去によりメモリー不足を緩和するためのIM列ストアの管理には、かなりのユーザー介入が必要です。自動インメモリーが、ユーザーの介入を最小限に抑えてこれらの問題に対処します。

6.2.2 自動インメモリーはどのように機能するか

データ除去の単位は、INMEMORYセグメントです。

INMEMORYセグメントは、優先度がNONEの場合にのみ除去の対象となります。基本的なプロセスは次のとおりです。

1. 移入ジョブが失敗します。これは、IM列ストアの領域がすべて使用されたことを意味します。
2. データベースは、対象となる移入されたセグメントの内部統計を使用して、除去するオブジェクトのセットを定義します。統計はヒート・マップで使用されるものと似ていますが、ヒート・マップを有効にする必要はありません。
3. セット内の各セグメントについて、データベースはセグメントに対してADOポリシーが有効かどうかをチェックします。
   - 有効なポリシーでセグメントが移入されたままになっている必要がある場合、ADOポリシーは自動インメモリーをオーバーライドします。データベースは何も行いません。
   - どのポリシーも除去を妨げていない場合、自動インメモリーはセグメントを除去するタスクを送信します。
4. Wnnnプロセスは、前述のチェックを通過するセグメントをすべて除去し、IM列ストア内の領域を解放します。

   INMEMORY属性は、除去されたセグメントに保持されます。

たとえば、夜間のバッチ・ジョブがsalesパーティション(優先度NONE)をロードした後、移入をトリガーするパーティションに問い合せます。IM列ストアはほぼ最大容量になっているため、パーティションの行の半分のみが移入されます。新しいパーティションを完全に移入できないと、自動インメモリーがトリガーされ、コールド・セグメントが除去されます。新しいパーティションに対する後続のオンデマンド移入ジョブによって、新しいsalesパーティションが完全に移入されます。

関連項目:

「領域管理ワーカー・プロセス(Wnnn)」

6.2.3 自動インメモリーのユーザー・インタフェース

初期化パラメータINMEMORY_AUTOMATIC_LEVELを使用して、自動インメモリーを有効または無効にします。

初期化パラメータ

システム・レベルの初期化パラメータINMEMORY_AUTOMATIC_LEVELは、次の値を指定できます。

- OFF (デフォルト)

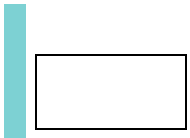
  このオプションは、自動インメモリーを無効にし、IM列ストアをOracle Database 12cリリース2 (12.2.0.1)の動作に戻します。

- LOW

  メモリー不足の場合、データベースはIM列ストアからコールド・セグメントを除去します。

- MEDIUM

  このレベルには、メモリー不足のために移入されなかったホット・セグメントが最初に投入されることを保証する追加の最適化が含まれます。



ノート:

自動インメモリーでは、HEAT_MAP 初期化パラメータを有効にする必要はありません。

IM列ストアに収まるように作業データ・セットに十分なメモリーをプロビジョニングすることをお薦めします。追加の自動インメモリー共有プール要件のサイズを設定するための経験則として、SGAメモリーのINMEMORYセグメントの数に5KBを掛けます。たとえば、10,000個のセグメントにINMEMORY属性がある場合は、自動インメモリー用に50MBの共有プールを確保します。

関連項目:

INMEMORY_AUTOMATIC_LEVELについてさらに学習するには、Oracle Databaseリファレンスを参照

DBMS_INMEMORY_ADMIN

DBMS_INMEMORY_ADMINパッケージを使用して、自動インメモリーが統計を考慮する時間枠を制御します。たとえば、自動インメモリーが過去1か月または過去1週間のみを考慮するように指定できます。

DBMS_INMEMORY_ADMIN.AIM_SET_PARAMETERプロシージャを使用して、AIM_STATWINDOW_DAYS定数を設定します。たとえば、スライディング統計ウィンドウを7日間に設定するには、次のプログラムを実行します。

```
EXEC DBMS_INMEMORY_ADMIN.AIM_SET_PARAMETER ( DBMS_INMEMORY_ADMIN.AIM_STATWINDOW_DAYS,
7 );
```

AIM_STATWINDOW_DAYSのデフォルト値は31日です。

対応するDBMS_INMEMORY_ADMIN.AIM_GET_PARAMETERプロシージャは、AIM_STATWINDOW_DAYSの現在の設定を取得します。

関連項目:

DBMS_INMEMORY_ADMIN.AIM_SET_PARAMETERおよびDBMS_INMEMORY_ADMIN.AIM_GET_PARAMETERにつ

いてさらに学習するには、Oracle Database PL/SQLパッケージおよびタイプ・リファレンスを参照

V$ VIEWS

V$IM_ADOTASKSおよびDBA_INMEMORY_AIMTASKSビューは、自動インメモリー・タスクによる決定を追跡できます。V$IM_ADOTASKDETAILSおよびDBA_INMEMORY_AIMTASKDETAILSビューには、タスクに関連する詳細が記述されています。

関連項目:

V$IM_ADOTASKSについてさらに学習するには、Oracle Databaseリファレンスを参照

6.2.4 自動インメモリーの制御

INMEMORY_AUTOMATIC_LEVEL初期化パラメータを使用して、自動インメモリーを制御します。

デフォルトでは、自動インメモリーはOFFに設定されています。INMEMORY_AUTOMATIC_LEVELをMEDIUMまたはLOWのいずれかに設定して有効にします。

前提条件

このパラメータをALTER SYSTEMで設定するには、ALTER SYSTEM権限を持っている必要があります。

INMEMORY_AUTOMATIC_LEVEL設定を変更するには、次のようにします。

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。
2. ALTER SYSTEM文を使用してINMEMORY_AUTOMATIC_LEVELを指定します。

   次の例では、自動インメモリーを無効にします。

```
ALTER SYSTEM SET INMEMORY_AUTOMATIC_LEVEL = 'OFF' SCOPE=BOTH;
```

関連項目:

INMEMORY_AUTOMATIC_LEVELについてさらに学習するには、Oracle Databaseリファレンスを参照

6.2.5 自動インメモリーの時間間隔の設定

DBMS_INMEMORY_ADMINパッケージを使用して、使用状況統計が自動インメモリーによってチェックされる時間間隔を設定します。

デフォルトでは、自動インメモリーは過去31日間の使用状況統計をチェックします。DBMS_INMEMORY_ADMIN.AIM_SET_PARAMETERにAIM_STATWINDOW_DAYSパラメータを指定して、現在の設定を変更できます。

前提条件

DBMS_INMEMORY_ADMIN.AIM_SET_PARAMETERおよびDBMS_INMEMORY_ADMIN.AIM_GET_PARAMETERプロシージャを実行するには、管理者権限が必要です。

前提

間隔を7日間に設定するとします。

自動インメモリー間隔の設定を変更するには、次のようにします。

1. SQL*PlusまたはSQL Developerで、管理者権限を持つユーザーとしてデータベースにログインします。
2. 必要に応じて、aim_statwindow_daysパラメータの現在の設定を確認します。

   次の例では、DBMS_INMEMORY_ADMIN.AIM_GET_PARAMETERプロシージャをコールします。

```
VARIABLE b_interval NUMBER
BEGIN
  DBMS_INMEMORY_ADMIN.AIM_GET_PARAMETER(
    DBMS_INMEMORY_ADMIN.AIM_STATWINDOW_DAYS, :b_interval);
END;
/
PRINT b_interval
B_INTERVAL
----------------------------
31
```

3. DBMS_INMEMORY_ADMIN.AIM_SET_PARAMETERプロシージャを使用してaim_statwindow_days設定を変更します。

   次のコードは、設定を7日間に変更します。

```
BEGIN
  DBMS_INMEMORY_ADMIN.AIM_SET_PARAMETER(
    DBMS_INMEMORY_ADMIN.AIM_STATWINDOW_DAYS, 7);
END;
/
```

関連項目:

DBMS_INMEMORY_ADMIN.AIM_SET_PARAMETERおよびDBMS_INMEMORY_ADMIN.AIM_GET_PARAMETERプロシージャについてさらに学習するには、Oracle Database PL/SQLパッケージおよびタイプ・リファレンスを参照

第III部 インメモリー問合せの最適化

この部では、インメモリー式、結合グループおよびインメモリー集計を使用して問合せを最適化する方法を説明します。また、変更したデータがIM列ストアでどのように再移入されるのかを説明します。

7 インメモリー式による問合せの最適化

IM列ストアとの関連においては、式は、1つ以上の値、演算子、および値を解決するSQLまたはPL/SQL関数(DETERMINISTICのみ)の組合せです。

式統計ストア(ESS)では、頻繁に評価される(ホットな)式の結果が自動的に追跡されます。DBMS_INMEMORY_ADMINパッケージを使用して、ホットな式を取得し、それらを非表示の仮想列として移入するか、それらの一部またはすべてを削除できます。

7.1 IM式について

デフォルトでは、DBMS_INMEMORY_ADMIN.IME_CAPTURE_EXPRESSIONSプロシージャは、**インメモリー式(IM式)**と呼ばれる、ホットな式を特定および移入します。

IM式は、ヒープ構成表における非表示の仮想列としてマテリアライズされますが、非仮想列と同じ方法でアクセスされます。マテリアライズされた式を格納するために、IM列ストアでは、固定幅ベクター、および固定幅コードでのディクショナリ・エンコーディングなど、特別な圧縮形式が使用されます。

ノート:

IM 式は外部表ではサポートされていません。

Oracle Databaseにより、IM列ストアへの移入の候補となる式が自動的に特定されます。DBA_IM_EXPRESSIONS.COLUMN_NAMEでは、IM式の列には接頭辞SYS_IMEがあります。SYS_IME列を直接作成することはできません。たとえば、weekly_salおよびann_compという別名が付けられた2つの式を指定する、次のような問合せを考察します。

```
SELECT employee_id, last_name, salary, commission_pct,
       ROUND(salary*12/52,2) as "weekly_sal",
       12*(salary*NVL(commission_pct,0)+salary) as "ann_comp"
FROM   employees
ORDER BY ann_comp;
```

ROUND(salary*12/52,2)および12*(salary*NVL(commission_pct,0)+salary)の計算式は、計算が集中的に行われて頻繁にアクセスされ、それにより、それらは非表示のIM式列の候補になります。

DBMS_INMEMORY_ADMINパッケージは、IM式を管理するための主要インタフェースです。

- 次回の再移入中にデータベースでIM式を特定してそれらがそれぞれの表に追加されるようにするには、IME_CAPTURE_EXPRESSIONSを使用します。
- IM式の即時移入を強制的に実行するには、IME_POPULATE_EXPRESSIONSを使用します。
- SYS_IME列を削除するには、DBMS_INMEMORY_ADMIN.IME_DROP_ALL_EXPRESSIONSまたはDBMS_INMEMORY.IME_DROP_EXPRESSIONSを使用します。

関連項目:

- インメモリー外部表
- 式についてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照

- DBMS_INMEMORY_ADMINについてさらに学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照
- DBA_IM_EXPRESSIONSビューについてさらに学習するには、『Oracle Databaseリファレンス』を参照

7.1.1 IM式の目的

IM式により、計算が集中的に行われる式を事前計算することで、データ・セットが大きい問合せが高速化されます。

IM式は、頻繁に実行される表結合、投影および述語評価に特に役立ちます。IM式の主な利点は、次のとおりです。

- 問合せで、式を毎回再計算する必要がありません。IM列ストアが式結果を移入しないと、データベースでは行ごとに計算が必要になり、これはリソースが集中的に消費されます。データベースで、移入中にCPUのオーバーヘッドが発生します。
- IM式のマテリアライズによって、SIMDベクター処理やIMCUプルーニングなどのパフォーマンス強化機能をデータベースで利用できるようになります。
- ユーザーではなく、データベースが、ユーザー指定の式取得ウィンドウ内で最もアクティブな式を追跡します。

IM式とマテリアライズド・ビューは、式の評価の繰返しをどのように回避するかという同じ問題に対応しています。ただし、IM式にはマテリアライズド・ビューを上回る利点があります。

- IM式では、永続的に格納されるわけではないデータを取得できます。

  たとえば、IM列ストアでは、問合せ内の式に基づいて、内部計算を自動的にキャッシュできます。
- 効率的に使用するために、マテリアライズド・ビューには問合せでリストされるすべての列が含まれるか、問合せがビューおよびベース表を結合する必要があります。対照的に、IM式が含まれる問合せはいずれも利点を享受できます。
- ユーザーが作成したオブジェクトであるマテリアライズド・ビューとは異なり、データベースでは、IM式は自動的に特定および作成されます。

関連項目:

- 「インメモリー記憶域索引」
- 「式統計ストア(ESS)」
- マテリアライズド・ビューについてさらに学習するには、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照

7.1.2 IM式の仕組み

式をIM式の候補として特定するために、データベースでは、ESSに問い合せます。オプティマイザでは、ESSを使用して、ヒープ構成表に対する式評価についての統計が保持されます。

7.1.2.1 IM式のインフラストラクチャ

IM式のインフラストラクチャは、IM式の結果、IM仮想列、およびIM列ストア内のその他の役立つ内部計算を計算および移入する役割を担います。これらの最適化は、主に分析問合せに役立ちます。

移入された結果には、投影、スキャンまたは結合式で使用される列上の関数評価が含まれる可能性があります。IM列ストアでは、問合せ評価中に、SQLランタイム・エンジンによって評価される式に基づいて、内部計算を自動的にキャッシュできます。

仮想列

IM式の移入の他に、IM列ストアでは、インメモリー仮想列を移入できます。基礎となるメカニズムは同じです。IM式は仮想列です。ただし、IM仮想列がユーザーによって作成され公開されるのに対して、IM式はデータベースによって作成され非表示にされます。

関連項目:

「インメモリー表に対する列の有効化および無効化」

静的式: バイナリJSON列

IM式のインフラストラクチャでは、動的式(IM式および仮想列)と静的式の両方がサポートされています。

Oracle Database 12cリリース2 (12.2)以降、IM列ストアでは、バイナリJSON形式のOSONがサポートされています。これは、行指向のJSONテキスト・ストレージよりもパフォーマンスに優れています。問合せでは、実際のJSONデータにアクセスしますが、アクセスの高速化のために、最適化された仮想列を使用します。

データベースでは、IM式のインフラストラクチャを使用して、JSONテキスト列の効率的なバイナリ表現が仮想列としてロードされます。MAX_STRING_SIZE初期化パラメータが、VARCHAR2データ型のEXTENDEDに設定されている場合は、IM列ストアに32KBまでのOSON仮想列を格納できます。

Oracle Databaseでは、JSON_TABLE、JSON_VALUEおよびJSON_EXISTSという複数のJSON関数がサポートされています。INMEMORY_EXPRESSIONS_USAGE初期化パラメータは、動的式と静的式の両方の挙動を制御します。

関連項目:

- Database In-MemoryでのJSONの使用についてさらに学習するには、Oracle Database JSON開発者ガイドを参照
- INMEMORY_EXPRESSIONS_USAGE、ALL_JSON_COLUMNSおよびMAX_STRING_SIZEについて学習するには、『*Oracle Database*リファレンス』を参照してください

7.1.2.2 IM式の取得

IME_CAPTURE_EXPRESSIONSプロシージャを呼び出すと、データベースでESSに問合せが行われ、指定された時間範囲で最も頻繁にアクセスされている(最もホットな)式が20個特定されます。

時間範囲は、ユーザー定義の時間枠(過去24時間)またはデータベース作成以降のどちらかです。データベースでは、IM列ストアに少なくとも一部分が移入されている表の式のみが考慮されます。

7.1.2.2.1 式の取得間隔

式の取得間隔は、取得可能な式をデータベースが評価する期間です。

Oracle Database 18c以降、DBMS_INMEMORY_ADMIN.IME_CAPTURE_EXPRESSIONSプロシージャのsnapshotパラメータでは、式の取得間隔を定義する次の値を使用できます。

- CUMULATIVE

  データベースでは、データベース作成以降のすべての式統計が考慮されます。

- CURRENT

  データベースでは、過去24時間の式統計のみが考慮されます。

- WINDOW

データベースは、最新のユーザー指定式取得ウィンドウで追跡された式に非表示の仮想列を追加します。このウィンドウは、IME_OPEN_CAPTURE_WINDOWプロシージャを手動で起動して開き、IME_CLOSE_CAPTURE_WINDOWプロシージャを手動で起動して閉じます。

取得ウィンドウが現在開いている場合、データベースはこの時点までに現在のウィンドウで追跡されたすべての式を考慮し、最もホットな式をマテリアライズします。現在のウィンドウで追跡された式をリストするには、SNAPSHOT = 'WINDOW'でDBA_EXPRESSION_STATISTICSを問い合せます。

ユーザー定義の時間間隔(snapshot='WINDOW')は、このウィンドウ内で発生する式のみをマテリアライズ用に考慮する場合に役立ちます。このメカニズムは、短い間隔がワークロード全体で代表的な場合に特に便利です。たとえば、取引時間帯の間に証券会社が一連の式を収集し、IM列ストアでそれらをマテリアライズして、ワークロード全体の将来の問合せ処理を高速化できます。

関連項目:

IME_OPEN_CAPTURE_WINDOW、IME_CLOSE_CAPTURE_WINDOWおよびIME_CAPTURE_EXPRESSIONSについてさらに学習するには、*Oracle Database PL/SQL*パッケージおよびタイプ・リファレンスを参照

7.1.2.2.2 非表示のSYS_IME仮想列

取得中、データベースにより最もホットな20個の式がそれぞれの表に非表示のSYS_IME仮想列として追加され、デフォルトのINMEMORY列圧縮句が適用されます。

前の呼出し中に追加されたSYS_IME列が最新の式リストに表示されなくなった場合、その属性はNO INMEMORYに変わります。

図7-1 非表示のSYS_IME仮想列の定義

属性がINMEMORYかどうかに関係なく、表のSYS_IME列の最大数は50個です。表の式が上限の50個に達した後は、データベースにより、新しいSYS_IME列は追加されません。新しい式を許可するには、SYS_IME列をDBMS_INMEMORY.IME_DROP_EXPRESSIONSまたはDBMS_INMEMORY_ADMIN.IME_DROP_ALL_EXPRESSIONSプロシージャを使用して削除する必要があります。

SYS_IME仮想列およびユーザー定義の仮想列のどちらも、表での列の上限である1000個に数えられます。たとえば、表に980個の非仮想(ディスク上)列が含まれる場合は、20個の仮想列のみを追加できます。

関連項目:

DBMS_INMEMORY_ADMINについてさらに学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照

7.1.2.3 ESSの仕組み

ESSは、式評価についての統計を格納するためにオプティマイザによって保持されるリポジトリです。

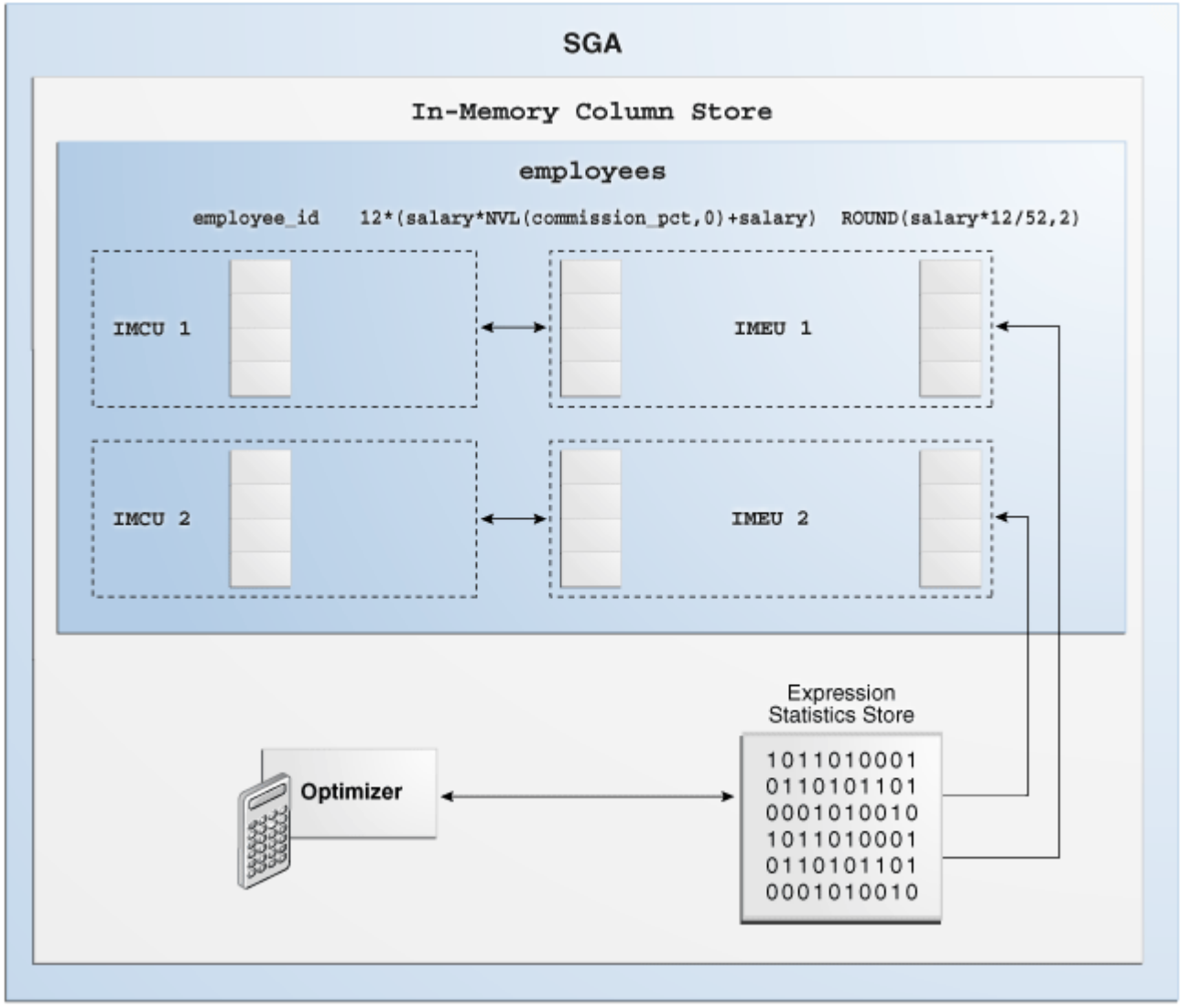
表ごとに、ESSで実行頻度や評価コストなどの式統計が保持されます。述語の評価時は、Oracle Databaseにより、式の評価回数および動的コストに関して実行時フィードバックが追跡および提供されます。ESS統計に基づいて、データベースで、特定の式がIM式であれば問合せのパフォーマンスが向上すると判断される場合があります。

ノート:

特定の表の ESS にキャッシュされた式には、この表の列のみが含まれます。このルールは、Oracle Database で決定性の PL/SQL 関数が IM 式の候補として特定された場合は、特に重要となります。

図7-2 ESSとIM式

この図で、ESSではemployees表で一般的に使用される2つの式、ROUND(salary*12/52,2)および12*(salary*NVL(commission_pct,0)+salary)が決定されました。データベースによってemployeesがIM列ストアに移入されると、2つのIMCUに列データが格納されます。各IMCUはそのIMEUのみと関連付けられ、これには、そのIMCU内の行に対して一般的に使用される2つの式の導出された値が含まれています。



すべての式がIM式の候補であるわけではありません。データベースでは、頻繁にアクセスされる式のみが考慮されます。IM式は非表示の仮想列として実装されるため、仮想列の制限事項を満たしている必要もあります。

IM列ストアはESSのクライアントですが、ESSはDatabase In-Memoryの機能から独立しています。オプティマイザ自体など、他のクライアントもESS統計を使用できます。

関連項目:

- 「式統計ストア(ESS)」
- 仮想列についてさらに学習するには、『Oracle Database管理者ガイド』を参照

7.1.2.4 データベースではどのようにIM式が移入されるか

インメモリー・コーディネータ・プロセス(IMCO)の指示に従い、領域管理ワーカー・プロセス(Wnnn)で、IM式がIMEUに自動的にロードされます。

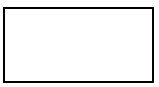
データベースで、ユーザー定義またはIM式のどちらの仮想列を移入するかに関する情報とともに、すべてのインメモリー圧縮単位(IMCU)の移入または再移入タスクが増大されます。決定は、INMEMORY_EXPRESSION_USAGEおよびINMEMORY_VIRTUAL_COLUMNS初期化パラメータの設定によって異なります。

ノート:

DBMS_INMEMORY.IME_CAPTURE_EXPRESSIONS プロシージャは、自動検出した式を非表示の仮想列として追加します。

Wnnnプロセスにより、IMCUが作成されます。IMEUを作成するために、それらのプロセスで、次のようなステップがさらに実行されます。

1. 式値を作成します
2. 値を列形式に変換し、それらをインメモリー式単位(IMEU)に圧縮します
3. 各IMEUをその関連するIMCUにリンクさせます



ノート:

IMEU に格納する式の数が増加すると、ワーカー・プロセスで、式値を計算するために少し多くの CPU が使用される場合があります。このオーバーヘッドにより、移入時間が長くなることがあります。

関連項目:

- 「式統計ストア(ESS)」
- 「領域管理ワーカー・プロセス(Wnnn)」
- 「インメモリー式単位(IMEU)」
- DBMS_INMEMORY.IME_CAPTURE_EXPRESSIONSプロシージャについてさらに学習するには、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照

7.1.2.5 IMEUはどのようにIMCUに関連するか

どの行についても、物理的な列はIMCU内に存在し、仮想列は関連するIMEU内に存在します。IMEUは、IMCUと同様に、読取り専用であり、列となります。

IMEUは、特定のINMEMORYセグメントのために作成されたIMCUの論理拡張であるため、デフォルトでは、INMEMORY句、およびDISTRIBUTEおよびDUPLICATEなどのOracle Real Applications Cluster (Oracle RAC)プロパティを継承します。IMEUは、1つのIMCUのみと関連付けられます。データベースでは、追加および削除が容易になるよう、IMEUは別個の構造として管理されます。

ノート:

IMEU には、ユーザーが作成した IM 仮想列も含まれます。

ソース・データが変更された場合は、データベースにより、再移入中に、IM式で得られたデータが変更されます。たとえば、トランザクションで表内の100個の給与値が更新された場合は、領域管理ワーカー・プロセス(Wnnn)により、変更されたこれら100個の値から導出されるすべてのIM式値が自動的に更新されます。データベースでは、最初にすべてのIMCUを再移入してからすべてのIMEUを再移入するのではなく、IMCUおよびその関連するIMEUが一緒に再移入されます。IMEUは、IMCUの再移入中も問合せに使用できます。

7.1.3 IM式のユーザー・インタフェース

DBMS_INMEMORY_ADMINパッケージ、DBMS_INMEMORYパッケージおよびINMEMORY_EXPRESSIONS_USAGE初期化パラメータは、IM式の挙動を制御します。

7.1.3.1 INMEMORY_EXPRESSIONS_USAGE

INMEMORY_EXPRESSIONS_USAGE初期化パラメータは、どのタイプのIM式を移入するかを決定します。INMEMORY_VIRTUAL_COLUMNS初期化パラメータは、通常の(非表示ではない)仮想列の移入を制御します。

IM列ストアが有効になっている(INMEMORY_SIZEがゼロ以外)場合、INMEMORY_EXPRESSIONS_USAGEは、データベースで移入されるIM式のタイプを制御します。INMEMORY_EXPRESSIONS_USAGE初期化パラメータには、次のオプションがあります。

- ENABLE

  データベースでは、静的IM式と動的IM式の両方がIM列ストアに移入されます。この値を設定すると、一部の表でインメモリー・フットプリントが増加します。これはデフォルトです。

- STATIC_ONLY

  静的構成では、IM列ストアでOSON (バイナリJSON)列をキャッシュできます。それらは、IS_JSONチェック制約でマークされます。内部的には、OSON列はSYS_IME_OSONという名前の非表示の仮想列です。

- DYNAMIC_ONLY

  データベースでは、SYS_IME非表示仮想列として表に追加された、頻繁に使用されているかホットな式のみが移入されます。この値を設定すると、一部の表でインメモリー・フットプリントが増加します。

- DISABLE

  データベースでは、静的か動的かに関係なく、IM式はIM列ストアに移入されません。

INMEMORY_EXPRESSIONS_USAGEの値の変更により、現在IM列ストアに移入されているIM式への影響はすぐにはありません。たとえば、INMEMORY_EXPRESSIONS_USAGEをDYNAMIC_ONLYからDISABLEに変更した場合は、データベースによって、格納されているIM式はすぐには削除されません。正確に述べると、次の再移入で、無効になっているIM式が除外されます。それにより、効率的にそれらが削除されます。

関連項目:

- Database In-MemoryでのJSONの使用についてさらに学習するには、Oracle Database JSON開発者ガイドを参照
- INMEMORY_EXPRESSIONS_USAGEおよびALL_JSON_COLUMNSについて学習するには、*Oracle Database*リファレンスを参照

7.1.3.2 DBMS_INMEMORY_ADMINおよびDBMS_INMEMORY

IM式を管理するには、DBMS_INMEMORY_ADMINおよびDBMS_INMEMORYパッケージを使用します。

IM式を管理するためのPL/SQLプロシージャ

パッケージ	プロシージャ	説明
DBMS_INMEMORY_ADMIN	IME_OPEN_CAPTURE_WINDOW	このプロシージャは、式取得ウィンドウの開始を知らせます。
DBMS_INMEMORY_ADMIN	IME_CLOSE_CAPTURE_WINDOW	このプロシージャは、現在の式取得ウィンドウの終了を知らせます。
DBMS_INMEMORY_ADMIN	IME_GET_CAPTURE_STATE	このプロシージャは、式取得ウィンドウの現在の取得状況および最新の変更のタイムスタンプを返します。
DBMS_INMEMORY_ADMIN	IME_CAPTURE_EXPRESSIONS	このプロシージャは、指定された時間範囲でデータベース内の最も頻繁にアクセスされている(最もホットな)式を 20 個取得します。
DBMS_INMEMORY_ADMIN	IME_POPULATE_EXPRESSIONS	このプロシージャは、IME_CAPTURE_EXPRESSIONS プロシージャの最後の呼出しで取得された IM 式の移入を強制的に実行します。
DBMS_INMEMORY_ADMIN	IME_DROP_ALL_EXPRESSIONS	このプロシージャは、データベース内のすべての SYS_IME 仮想列を削除します。
DBMS_INMEMORY	IME_DROP_EXPRESSIONS	このプロシージャは、指定された一連の SYS_IME 仮想列を表から削除します。

関連項目:

DBMS_INMEMORYおよびDBMS_INMEMORY_ADMINパッケージについてさらに学習するには、*Oracle Database PL/SQLパッケージおよびタイプ・リファレンス*を参照

7.1.4 IM式のための基本作業

INMEMORY_EXPRESSIONS_USAGEのデフォルト設定では、データベースで、動的および静的IM式の両方を使用できます。IM列ストアに式を移入するには、DBMS_INMEMORY_ADMINを使用する必要があります。

通常は、次の順序でIM式の作業を実行します。

1. 必要な場合は、データベースで使用可能なIM式のタイプを変更します。

   「IM式の使用の構成」を参照してください。

2. IM式を取得および移入します。

   「IM式の取得および移入」を参照してください。

3. 必要な場合は、IM式の一部またはすべてを削除します。

   「IM式の削除」を参照してください。

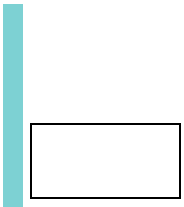
7.2 IM式の使用の構成

必要な場合は、INMEMORY_EXPRESSIONS_USAGEを使用して、移入の対象となるIM式のタイプを選択するか、すべてのIM式の移入を無効にします。

前提条件

データベースでのIM式の使用を可能にするには、次の条件を満たしている必要があります。

- INMEMORY_SIZE初期化パラメータがゼロ以外の値に設定されている。
- 初期化パラメータCOMPATIBLEの値が12.2.0以上に設定されている。



ノート:

Oracle Real Applications Cluster (RAC)データベースでは、INMEMORY_EXPRESSIONS_USAGE初期化パラメータは、すべてのデータベース・インスタンスで同じ値である必要はありません。各 IMCU は、独立して仮想列をリストします。各 IMCU で、IMCU が移入または再移入されたときに存在していた初期化パラメータ値および仮想列に基づいて、様々な式をマテリアライズできます。

IM式の使用を構成するには:

1. 適切な権限があるユーザーとしてデータベースにログインします。
2. IM式の使用を構成するには、ALTER SYSTEM文を使用して、INMEMORY_EXPRESSIONS_USAGEを次のいずれかの値に設定します。
   - ENABLE (デフォルト)—動的および静的IM式を有効にします
   - STATIC_ONLY—静的IM式のみを有効にします
   - DYNAMIC_ONLY—動的IM式のみを有効にします

- DISABLE—すべてのIM式を無効にします

例7-1 IM式の無効化

次の文は、IM列ストア内のIM式の記憶域を無効にします。

```
ALTER SYSTEM SET INMEMORY_EXPRESSIONS_USAGE='DISABLE' SCOPE=BOTH;
```

関連項目:

INMEMORY_EXPRESSIONS_USAGEについてさらに学習するには、『Oracle Databaseリファレンス』を参照してください。

7.3 IM式の取得および移入

IME_CAPTURE_EXPRESSIONSプロシージャは、指定された時間間隔でデータベース内の最も頻繁にアクセスされている(最もホットな)式を20個取得します。IME_POPULATE_EXPRESSIONSプロシージャは、DBMS_INMEMORY_ADMIN.IME_CAPTURE_EXPRESSIONSの最後の呼出しで取得された式の移入を強制的に実行します。

IME_CAPTURE_EXPRESSIONSプロシージャを呼び出したときは必ず、データベースで、式統計ストア(ESS)に問合せが行われ、IM列ストアに少なくとも一部分が移入されているヒープ構成表の式のみが考慮されます。データベースにより、最もホットな20個の式がそれぞれの表に、文字列SYS_IMEという接頭辞が付いた非表示の仮想列として追加され、デフォルトのINMEMORY列圧縮句が適用されます。前の呼出し時に追加されたSYS_IME列が、最新の上位20件のリスト内になくなると、その属性はNO INMEMORYに変わります。

IME_POPULATE_EXPRESSIONSを呼び出さない場合、データベースでは、SYS_IME列は、それらの親IMCUが再移入されるときに段階的に再移入されます。表が再移入されない場合、データベースでは、IME_CAPTURE_EXPRESSIONSプロシージャによって取得された新しいSYS_IME列は再移入されません。IME_POPULATE_EXPRESSIONSは、再移入を強制的に実行することで、この問題を解決します。

IME_POPULATE_EXPRESSIONSプロシージャは、内部的に、INMEMORY属性が指定されているSYS_IME列があるすべての表に対して、DBMS_INMEMORY.REPOPULATEを呼び出します。指定された表サブセット内のSYS_IME列を移入するには、DBMS_INMEMORY_ADMIN.IME_POPULATE_EXPRESSIONSではなくDBMS_INMEMORY.REPOPULATEを使用します。

前提条件

データベースでのIM式の取得を可能にするには、次の条件を満たしている必要があります。

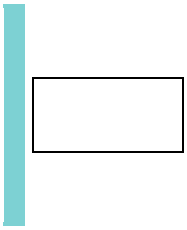
- INMEMORY_EXPRESSIONS_USAGE初期化パラメータがDISABLE以外の値に設定されている必要があります。
- INMEMORY_SIZE初期化パラメータがゼロ以外の値に設定されている。
- 初期化パラメータCOMPATIBLEの値が12.2.0以上に設定されている必要があります。
- (事前に決められた24時間またはそれより長い時間範囲ではなく)任意の長さの式取得ウィンドウを指定するには、DBMS_INMEMORY_ADMIN.IME_OPEN_CAPTURE_WINDOWを使用してウィンドウを開き、IME_CLOSE_CAPTURE_WINDOWを使用して閉じる必要があります。ウィンドウは、Oracle RACデータベースのすべてのインスタンスにわたってグローバルです。

IM式を取得および移入するには:

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。

2. (事前に決められた時間範囲ではなく)ウィンドウの長さを指定する場合は、次のように DBMS_INMEMORY_ADMIN.IME_OPEN_CAPTURE_WINDOWを実行します。

```
EXEC DBMS_INMEMORY_ADMIN.IME_OPEN_CAPTURE_WINDOW();
```



ノート:

式取得ウィンドウが現在開いているかどうかを確認するには、DBMS_INMEMORY_ADMIN.IME_GET_CAPTURE_STATE を実行します。

3. 前述のステップでIME_OPEN_CAPTURE_WINDOWを使用してウィンドウを開いた場合、次のように IME_CLOSE_CAPTURE_WINDOWを使用して閉じます。

```
EXEC DBMS_INMEMORY_ADMIN.IME_CLOSE_CAPTURE_WINDOW();
```

4. DBMS_INMEMORY_ADMIN.IME_CAPTURE_EXPRESSIONSを実行し、intervalパラメータを次のいずれかの値に設定します。

- CUMULATIVE

  データベースでは、データベース作成以降のすべての式統計が考慮されます。

- CURRENT

  データベースでは、過去24時間の式統計のみが考慮されます。

- WINDOW

  データベースは、最新のユーザー指定式取得ウィンドウで追跡された式に非表示の仮想列を追加します。このウィンドウは、IME_OPEN_CAPTURE_WINDOWプロシージャを手動で起動して開き、IME_CLOSE_CAPTURE_WINDOWプロシージャを手動で起動して閉じます。

  取得ウィンドウが現在開いている場合、データベースはこの時点までに現在のウィンドウで追跡されたすべての式を考慮し、最もホットな式をマテリアライズします。現在のウィンドウで追跡された式をリストするには、SNAPSHOT = 'WINDOW'でDBA_EXPRESSION_STATISTICSを問い合せます。

5. 必要に応じて、取得されたすべてのIM式の即時移入を強制的に実行するには、次のように DBMS_INMEMORY_ADMIN.IME_POPULATE_EXPRESSIONSを実行します。

```
EXEC DBMS_INMEMORY_ADMIN.IME_POPULATE_EXPRESSIONS();
```

例7-2 ユーザー定義ウィンドウでの式の取得

この例は、WINDOW取得モードの使用方法を示しています。目標は、式取得ウィンドウを開いて閉じ、この時間帯の間にデータベースが追跡したすべての式を取得することです。次のステップを実行します。

1. 式取得ウィンドウを開き、式を生成してからウィンドウを閉じます。

```
EXEC DBMS_INMEMORY_ADMIN.IME_OPEN_CAPTURE_WINDOW();
-- Generate expressions for the database to track
EXEC DBMS_INMEMORY_ADMIN.IME_CLOSE_CAPTURE_WINDOW();
```

2. 問合せDBA_EXPRESSION_STATICS (出力例を含む)。

```
COL OWNER FORMAT A6
COL TABLE_NAME FORMAT A9
COL COUNT FORMAT 99999
```

```
COL CREATED FORMAT A10
COL EXPRESSION_TEXT FORMAT A29
SELECT OWNER, TABLE_NAME, EVALUATION_COUNT AS COUNT,
       CREATED, EXPRESSION_TEXT
FROM   DBA_EXPRESSION_STATISTICS
WHERE  SNAPSHOT = 'WINDOW'
AND    OWNER = 'SH';
OWNER  TABLE_NAM  COUNT CREATED    EXPRESSION_TEXT
------ --------- ------ ---------- -------------------------
SH     SALES       4702 09-OCT-17  "QUANTITY_SOLD"
SH     SALES       4702 09-OCT-17  "QUANTITY_SOLD"*"AMOUNT_SOLD"
SH     SALES       4702 09-OCT-17  "PROD_ID"
SH     SALES       4702 09-OCT-17  "CUST_ID"
SH     SALES       4702 09-OCT-17  "CHANNEL_ID"
SH     SALES       4702 09-OCT-17  "AMOUNT_SOLD"
```

前述の問合せは、ESSで追跡された列と、shスキーマ内の問合せの時間帯の間に取得された式の両方を示します。最新の時間帯の間に、データベースは1つの式(QUANTITY_SOLD*AMOUNT_SOLD)を取得しました。

3. IME_CAPTURE_EXPRESSIONSを使用して、現在のウィンドウ内のすべての式がマテリアライズできるかをデータベースに検討させます。

```
EXEC DBMS_INMEMORY_ADMIN.IME_CAPTURE_EXPRESSIONS('WINDOW');
```

4. 問合せDBA_IM_EXPRESSIONS (出力例を含む)。

```
COL OWNER FORMAT a6
COL TABLE_NAME FORMAT a9
COL COLUMN_NAME FORMAT a25
SET LONG 50
SET LINESIZE 150
SELECT OWNER, TABLE_NAME, COLUMN_NAME, SQL_EXPRESSION
FROM   DBA_IM_EXPRESSIONS;
OWNER  TABLE_NAM COLUMN_NAME               SQL_EXPRESSION
------ --------- ------------------------- ------------------------------
SH     SALES     SYS_IME000100000025201B   "QUANTITY_SOLD"*"AMOUNT_SOLD"
```

前述の出力には、表に追加され、最新のIME_CAPTURE_EXPRESSIONS呼出しの一部としてINMEMORYとマークされたすべての仮想列が表示されます。データベースは、表の様々なIMCUを再移入するときに、取得した式をIM列ストアに段階的に移入します。

5. 次の手順を実行して、取得されたすべてのIM式の移入を明示的に強制します。

```
EXEC DBMS_INMEMORY_ADMIN.IME_POPULATE_EXPRESSIONS();
```

forceパラメータをTRUEに設定してDBMS_INMEMORY.REPOPULATEプロシージャを実行することにより、特定の表からIM式を移入できることに注意してください。

例7-3 式取得ウィンドウの状態の判断

この例では、式取得ウィンドウを開き、その取得状態を判断します。

```
EXEC DBMS_INMEMORY_ADMIN.IME_OPEN_CAPTURE_WINDOW();
VARIABLE b_state VARCHAR2(25)
VARIABLE b_time  VARCHAR2(10)
EXECUTE DBMS_INMEMORY_ADMIN.IME_GET_CAPTURE_STATE(:b_state, :b_time)
PRINT b_state b_time
```

次の出力例は、式取得ウィンドウが現在開いていることを示しています。

```
B_STATE
--------------------------------------------------
```

```
OPEN
B_TIME
-------------------------------------------------
09-OCT-17
```

例7-4 過去24時間の上位20件のIM式の取得

この例では、最後の日に収集された統計のみを使用してIM式を取得してから、即時移入を強制的に実行します。

```
EXEC DBMS_INMEMORY_ADMIN.IME_CAPTURE_EXPRESSIONS('CURRENT');
EXEC DBMS_INMEMORY_ADMIN.IME_POPULATE_EXPRESSIONS();
```

次のDBA_IM_EXPRESSIONSの問合せでは、INMEMORYとマークされているすべてのIME仮想列を示しています(出力例を提供)。

```
COL OWNER FORMAT a6
COL TABLE_NAME FORMAT a9
COL COLUMN_NAME FORMAT a25
SET LONG 50
SET LINESIZE 150
SELECT OWNER, TABLE_NAME,
       COLUMN_NAME, SQL_EXPRESSION
FROM   DBA_IM_EXPRESSIONS;
OWNER  TABLE_NAM COLUMN_NAME               SQL_EXPRESSION
------ --------- ------------------------- ----------------------------------------
---
HR     EMPLOYEES SYS_IME00010000001746FD
12*("SALARY"*NVL("COMMISSION_PCT",0)+"SALARY")
HR     EMPLOYEES SYS_IME00010000001746FE   ROUND("SALARY"*12/52,2)
```

関連項目:

- IME_CAPTURE_EXPRESSIONSおよびIME_POPULATE_EXPRESSIONSについてさらに学習するには、『*Oracle Database PL/SQL*パッケージおよびタイプ・リファレンス』を参照
- DBA_IM_EXPRESSIONSビューについてさらに学習するには、『Oracle Databaseリファレンス』を参照

7.4 IM式の削除

DBMS_INMEMORY_ADMIN.IME_DROP_ALL_EXPRESSIONSプロシージャは、データベース内のすべてのSYS_IME式仮想列を削除します。DBMS_INMEMORY.IME_DROP_EXPRESSIONSプロシージャは、指定された一連のSYS_IME仮想列を表から削除します。

SYS_IME列を削除する一般的な理由は、領域およびパフォーマンスです。属性がINMEMORYかNO INMEMORYかに関係なく、表のSYS_IME列の最大数は50個です。1つの表に対して式が上限の50個に達すると、データベースにより、新しいSYS_IME列は追加されなくなります。新しい式のために領域を作り出すには、DBMS_INMEMORY.IME_DROP_EXPRESSIONSまたはDBMS_INMEMORY_ADMIN.IME_DROP_ALL_EXPRESSIONSプロシージャを使用して、SYS_IME列を手動で削除する必要があります。

IME_DROP_ALL_EXPRESSIONSプロシージャは、INMEMORY属性の有無に関係なく、すべての表からすべてのSYS_IME列を削除します。事実上、このプロシージャは、データベース規模のリセット・ボタンの役割を果たします。

IME_DROP_ALL_EXPRESSIONSを使用すると、SYS_IME列があるセグメントのすべてのIMEUおよびIMCUの削除がトリガーされます。たとえば、移入された50個の表にそれぞれ1つのSYS_IME列がある場合は、

IME_DROP_ALL_EXPRESSIONSによって、IM列ストアから50個すべての表が削除されます。これらのセグメントを再度移入するには、DBMS_INMEMORY.POPULATEプロシージャを使用するか、表の全体スキャンを実行する必要があります。

前提条件

IM式を削除するには、次の条件を満たしている必要があります。

- INMEMORY_EXPRESSIONS_USAGE初期化パラメータは、DISABLE以外の値に設定されています。
- INMEMORY_SIZE初期化パラメータがゼロ以外の値に設定されています。
- COMPATIBLE初期化パラメータが12.2.0以上に設定されています。

IM式を削除するには:

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。
2. DBMS_INMEMORY_ADMIN.IME_DROP_ALL_EXPRESSIONSまたはDBMS_INMEMORY.IME_DROP_EXPRESSIONSのどちらかを実行します。

   IME_DROP_EXPRESSIONSを実行する場合は、次のパラメータを指定します。

   - schema_name — インメモリー表を含むスキーマの名前
   - table_name — インメモリー表の名前
   - column_name — SYS_IME列の名前。デフォルトでこの値はnullで、この表内のすべてのSYS_IME列を指定します。

例7-5 表内のすべてのIM式の削除

この例では、hr.employees表内のすべてのIM式を削除します。

```
EXEC DBMS_INMEMORY.IME_DROP_EXPRESSIONS('hr', 'employees');
```

関連項目:

- INMEMORY_EXPRESSIONS_USAGE初期化パラメータについてさらに学習するには、『Oracle Databaseリファレンス』を参照
- DBMS_INMEMORY.IME_DROP_EXPRESSIONSおよびDBMS_INMEMORY_ADMIN.IME_DROP_ALL_EXPRESSIONSについてさらに学習するには、『*Oracle Database PL/SQL*パッケージおよびタイプ・リファレンス』を参照
- DBA_IM_EXPRESSIONSデータ・ディクショナリ・ビューについてさらに学習するには、『Oracle Databaseリファレンス』を参照

8 結合グループによる結合の最適化

結合グループは、効果的に結合できる1つ以上の列をリストするユーザー作成ディクショナリです。

この章のトピックは、次のとおりです:

8.1 インメモリー結合について

結合は、データ・ウェアハウジング・ワークロードの不可欠な一部です。結合される表がメモリーに格納されている場合、IM列ストアにより、結合のパフォーマンスが向上します。

スキャンおよび結合処理が高速化されるため、ブルーム・フィルタを使用する複雑な複数表結合および単純な結合には、IM列ストアが役立ちます。データ・ウェアハウジング環境では、最も頻繁に使用される結合には、1つのファクト表と1つ以上のディメンション表が関与しています。

次の結合は、表がIM列ストアに移入される場合に、より高速に実行されます。

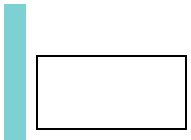
- ブルーム・フィルタを使用できる結合
- 複数の小規模なディメンション表の、1つのファクト表への結合
- 主キーと外部キーの関係がある2つの表の間の結合

8.2 結合グループについて

IM列ストアが有効になっている場合、データベースでは、結合グループを使用して、IM列ストアに移入された表の結合を最適化できます。

結合グループは、一連の表が頻繁に結合される列のセットです。列セットには、1つ以上の列が含まれ、最大255の列が含まれます。表セットには、1つ以上の内部表が含まれます。外部表はサポートされていません。

結合グループ内の列は、同じ表内または異なる表内にある可能性があります。たとえば、`sales`と`times`が`time_id`列で頻繁に結合される場合は、`(times(time_id), sales(time_id))`の結合グループを作成できます。`employees`表が`employee_id`列でそれ自体に結合されることが多い場合、結合グループは`(employees(employee_id))`である可能性があります。



ノート:

同じ列が複数の結合グループのメンバーになることはできません。

結合グループを作成すると、データベースにより、その結合グループで参照されている表の現在のインメモリー・コンテンツが無効になります。後続の再移入により、データベースで、共通ディクショナリによる表のIMCUの再エンコードが引き起こされます。このため、最初に結合グループを作成し、その後、表に移入することをお薦めします。

`CREATE INMEMORY JOIN GROUP`文を使用して結合グループを作成します。結合グループに列を追加するか、結合グループから列を削除するには、`ALTER INMEMORY JOIN GROUP`文を使用します。`DROP INMEMORY JOIN GROUP`文を使用して結合グループを削除します。

ノート:

Oracle Active Data Guard では、スタンバイ・データベースで結合グループ定義が無視されます。スタンバイ・データベースでは、共通ディクショナリは使用されず、結合グループが存在しないかのように問合せが実行されます。

例8-1 結合グループの作成

この例では、hr.employees表内およびhr.departments表内のdepartment_idを含む、deptid_jgという名前の結合グループを作成します。

```
CREATE INMEMORY JOIN GROUP deptid_jg
(hr.employees(department_id),hr.departments(department_id));
```

8.3 結合グループの目的

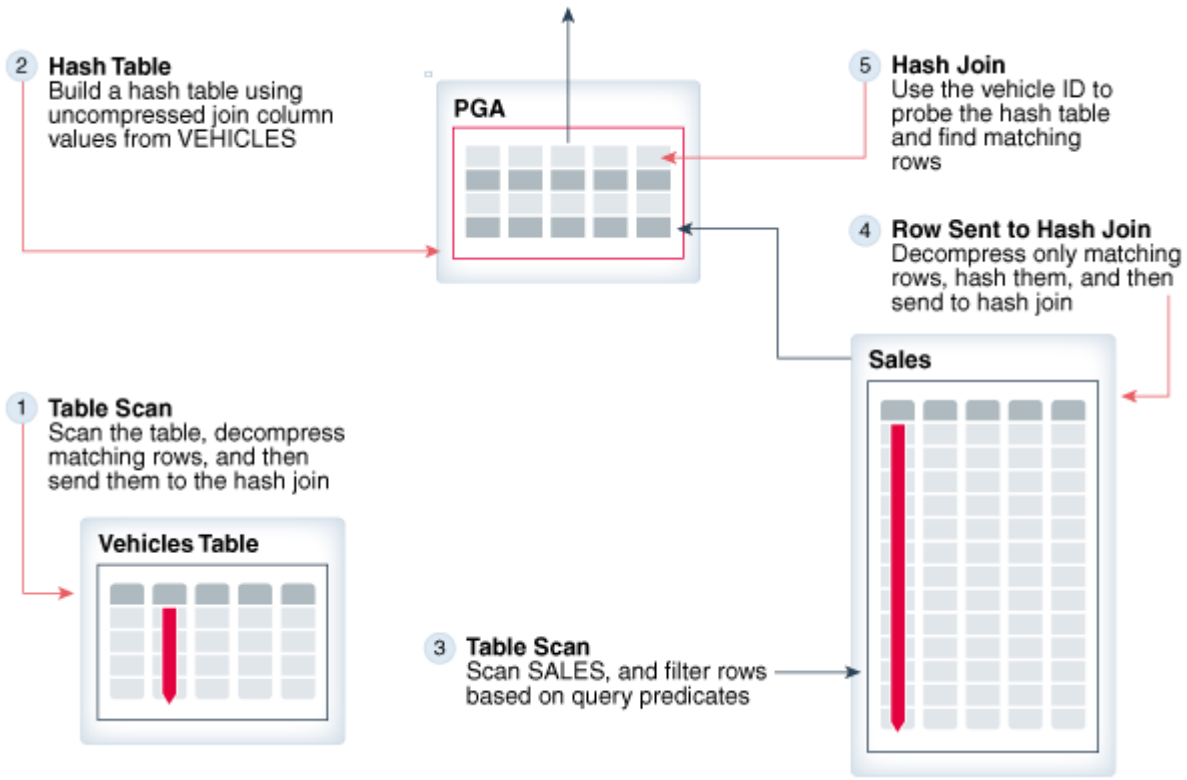
特定の問合せでは、結合グループにより、列値の解凍およびハッシュについてパフォーマンスのオーバーヘッドがなくなります。

結合グループを使用しないときは、オプティマイザでハッシュ結合を使用するがブルーム・フィルタを使用できないか、ブルーム・フィルタで行が効率的にフィルタされない場合に、データベースで、IMCUを解凍し、コストのかかるハッシュ結合を使用する必要があります。問題を示すため、スター・スキーマにsalesファクト表およびvehiclesディメンション表があると仮定します。次の問合せでは、これらの表が結合されますが、出力はフィルタされません。これは、データベースでブルーム・フィルタを使用できないことを意味します。

```
SELECT v.year, v.name, s.sales_price
FROM   vehicles v, sales s
WHERE  v.name = s.name;
```

次の図に、データベースによる2つのデータ・セットの結合方法を示します。

図8-1 結合グループなしのハッシュ結合



データベースは、次のようにハッシュ結合を実行します。

1. vehicles表をスキャンし、述語を満たす行(このケースでは、フィルタが存在しないため、すべての行が述語を満たします)を解凍し、行をハッシュ結合に送信します。
2. 解凍した行に基づいてPGAでハッシュ表を作成します
3. sales表をスキャンし、フィルタを適用します(この場合、問合せはフィルタを指定しません)
4. IMCUから一致する行を処理して、その行を結合に送信します

   調査側(この例では、sales表)の行セットがハッシュ結合で利用できる場合は、表スキャンによって送信される行セットは圧縮された形式になります。構築側から一致する行を検索するためにローカル・ディクショナリまたは結合グループが利用されるかどうかに応じて、ハッシュ結合で行が解凍されるか、未圧縮のままにされます。
5. 結合列(この場合、乗物名)を使用してハッシュ表を調べます

結合グループがv.name列およびs.name列に存在する場合、データベースは前述のステップをより効率的に実行します(解凍とフィルタリングのオーバーヘッドがなくなります)。結合グループの利点は次のとおりです。

- データベースは、圧縮されたデータに対して機能します。
- データベースは、調査行とハッシュされた行の結合キーの比較が必要になる結合キーのハッシングとハッシュ表の調査を回避します。

  結合グループが存在する場合、データベースは結合列値ごとのコードを共通ディクショナリに格納します。データベースは、ディクショナリのコードを使用して結合グループの配列を作成します。すべての配列要素は、ハッシュ領域(通常は、PGAメモリー)に格納されている構築側の行をポイントしています。調査時に、各調査行は結合キーに関連付けられたコードを保持しています。データベースは、このコードを使用して配列を検索し、配列要素にポインタが存在しているかどうかを調べます。ポインタが存在する場合は一致があり、それ以外の場合は一致がありません。
- ディクショナリ・コードは稠密および固定長であるため、領域が効率的に使用されます。
- 結合グループによる問合せの最適化は、ブルーム・フィルタを使用できないときに可能な場合があります。

8.4 結合グループの仕組み

結合グループでは、データベースによって、同じ共通ディクショナリを使用して結合グループ内のすべての列が圧縮されます。

8.4.1 結合グループでどのように共通ディクショナリが使用されるか

共通ディクショナリは、表レベルの、インスタンス固有の一連のディクショナリ・コードです。

データベースでは、基になる列で結合グループが定義されている場合に、IM列ストア内に共通ディクショナリが自動的に作成されます。共通ディクショナリにより、複数の結合列で同じディクショナリ・コードを共有できるようになります。

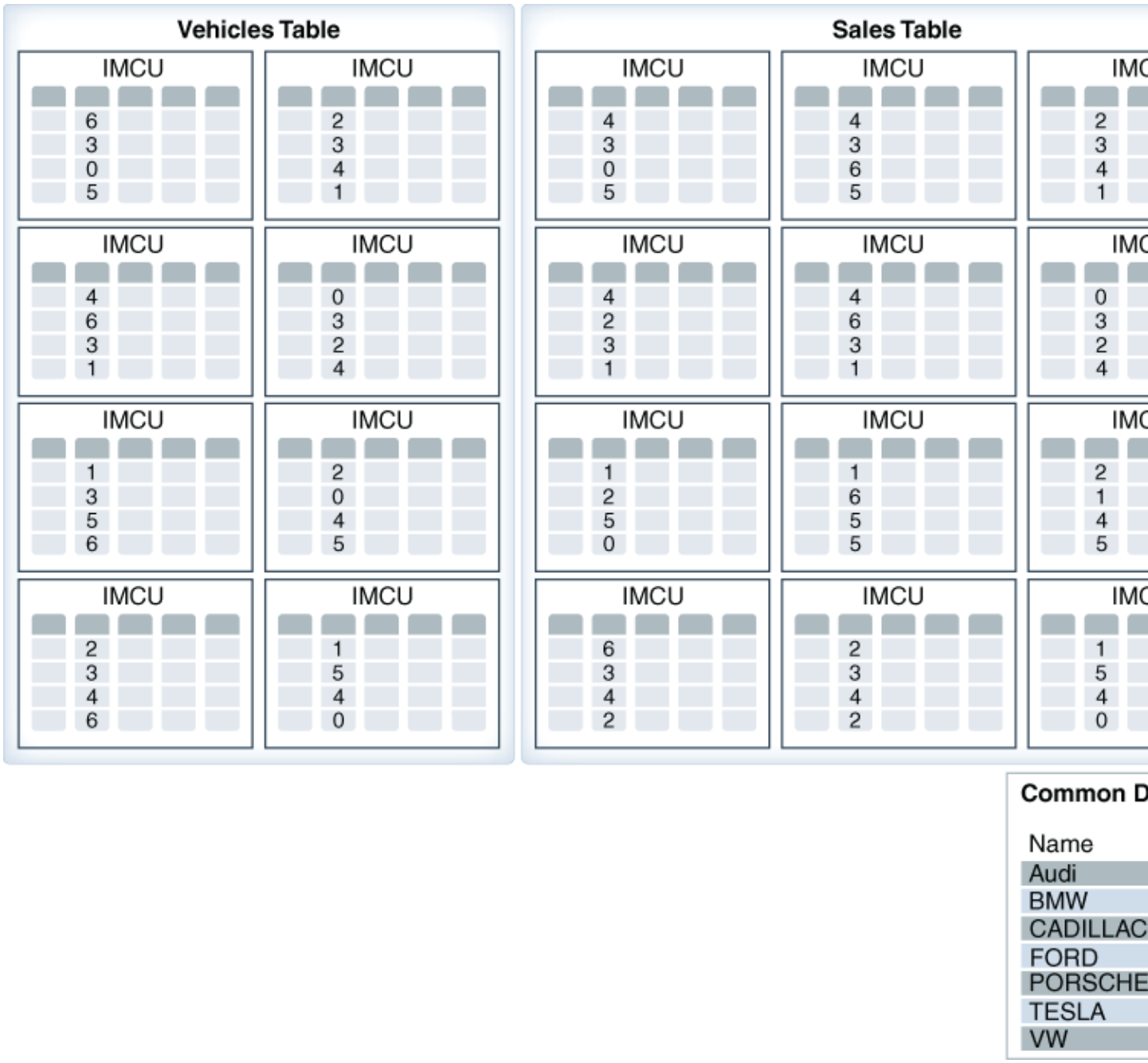
共通ディクショナリには、次のような利点があります。

- 共通ディクショナリからのコードでローカル・ディクショナリ内の値をエンコードします。それにより、圧縮が提供され、IMCUのキャッシュ効率が向上します
- 結合でディクショナリ・コードを使用して、ハッシュ結合中に使用されるデータ構造を構成および調査できます
- オプティマイザにより、カーディナリティ、列値の分布などの統計を取得できます

次の図は、sales.name列とvehicles.name列で作成された結合グループに対応する、共通ディクショナリを示していま

す。

図8-2 結合グループのための共通ディクショナリ



データベースで共通ディクショナリが使用される場合、各CUのローカル・ディクショナリには、元の値(AUDI、BMW、CADILLAC、FORDなど)は格納されません。かわりに、ローカル・ディクショナリには、共通ディクショナリに格納されている値への参照が格納されます。たとえば、ローカル・ディクショナリではAudiに値101、BMWには220を格納できます。共通ディクショナリでは、Audiに0、BMWに1を格納できます。ローカル・ディクショナリの101 (AUDI)は、共通ディクショナリの0 (AUDI)へのポインタです。

8.4.2 結合グループでどのようにスキャンが最適化されるか

主要な最適化は、列値ではなく共通ディクショナリ・コードでの結合であり、それによって、結合にハッシュ表を使用しないようにします。

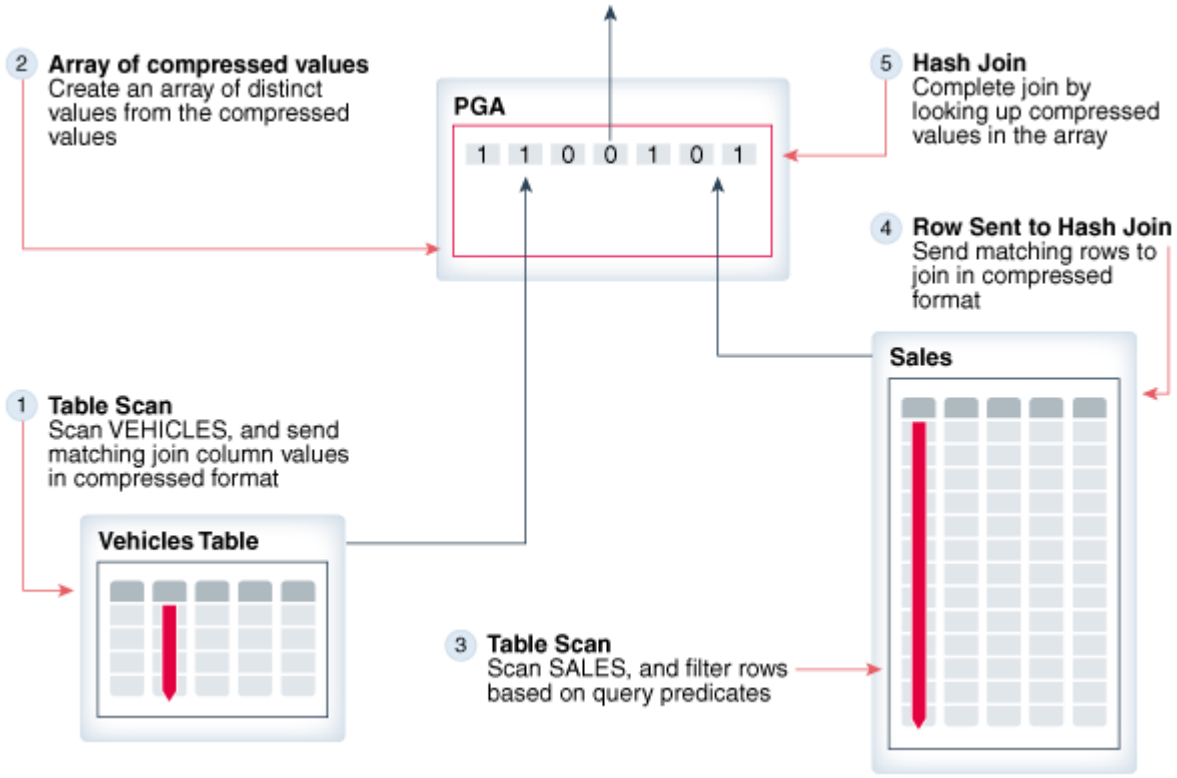
結合グループを使用してvehiclesとsalesをname列で結合する、次のような問合せについて考察します。

```
SELECT v.year, v.name, s.sales_price
FROM   vehicles v, sales s
```

```
WHERE  v.name = s.name
AND    v.name IN ('Audi', 'BMW', 'Porsche', 'VW');
```

次の図では、結合グループで作成された共通ディクショナリがどのように結合に役立つかを示します。

図8-3 結合グループありのハッシュ結合



前述の図に示されているように、データベースは次のように圧縮されたデータのハッシュ結合を実行します。

1. vehicles表をスキャンし、0 (Audi)、1 (BMW)、2 (Cadillac)などのディクショナリ・コード(元の列値ではない)をハッシュ結合に送信します。
2. PGA内で非重複共通ディクショナリ・コードの配列を構築します。
3. sales表をスキャンし、フィルタを適用します(このケースでは、ドイツ車のみのフィルタとなります)
4. 一致する行を、圧縮された形式で結合に送信します。
5. ハッシュ表のプロービングではなく、配列内の対応する値を探します。それにより、結合キー列上のハッシュ関数を計算する必要がなくなります。

この例では、vehicles表には7行のみがあります。vehicles.name列には、次のような値があります。

```
Audi
BMW
Cadillac
Ford
Porsche
Tesla
VW
```

共通ディクショナリにより、ディクショナリ・コードが各非重複値に割り当てられます。概念的には、共通ディクショナリは、次のようになります。

```
Audi     0
BMW      1
```

```
Cadillac 2
Ford     3
Porsche  4
Tesla    5
VW       6
```

データベースで、vehicles.nameがスキャンされます。最初のIMCU内の最初のディクショナリ・コードから始まり、最後のIMCU内の最後のコードで終わります。フィルタ(ドイツ車のみ)に適合するすべての行については1、フィルタに適合しないすべての行については0が格納されます。概念上、配列は次のようになります。

```
array[0]: 1
array[1]: 1
array[2]: 0
array[3]: 0
array[4]: 1
array[5]: 0
array[6]: 1
```

データベースでは、次に、salesファクト表がスキャンされます。この例を単純化するために、sales表には6行のみがあると仮定します。データベースでは、次のように行がスキャンされます(値ごとの共通ディクショナリ・コードを括弧で囲んで示します)。

```
Cadillac (2)
Cadillac (2)
BMW      (1)
Ford     (3)
Audi     (0)
Tesla    (5)
```

データベースでは、次に、vehicles.name配列内を進み、一致を探します。行が一致すると、データベースにより、一致する行がその関連する共通ディクショナリ・コードとともに送信され、対応する列値がvehicles.nameおよびsales.name IMCUから取得されます。

```
2  -> array[2] is 0, so no join
2  -> array[2] is 0, so no join
1  -> array[1] is 1, so join
3  -> array[3] is 0, so no join
0  -> array[0] is 1, so join
5  -> array[5] is 0, so no join
```

8.5 ハッシュ結合で共通ディクショナリ・エンコーディングが使用される場合

通常、結合グループ内の列を結合することでパフォーマンスが向上します。

結合グループの作成時に、次の処理がデータベースによって実行されます。

- 結合キー列に対応するディクショナリ値のハッシュがキャッシュされます
- 結合キー列に対応するNUMBERデータのバイナリ表現がキャッシュされます
- 同じ共通ディクショナリで列がエンコードされます

結合グループ内の列の結合では、最初の2つの最適化が常に使用されるためパフォーマンスが向上します。たとえば、オプティマイザがハッシュ結合を選択すると、問合せはブルーム・フィルタの調査にハッシュ値を使用するようになります。問合せでIM集計結合が使用される場合、問合せはキー・ベクターへの索引付けにキャッシュされた2進数を使用します。

ハッシュ結合には、ディクショナリ・エンコーディングが使用されることも使用されないこともあります。ハッシュ結合の少なくとも1つの列にディクショナリ・エンコーディングが存在する場合、問合せでは次の方法でエンコーディングを活用できます。

- 結合グループ対応のハッシュ結合

実行時に、ハッシュ結合の両方の列が共通ディクショナリ・エンコーディング・データを保持します。実行計画は、ハッシュ結合の両側が関与する分散のないパラレル・ハッシュ結合計画か、シリアル・ハッシュ結合計画のどちらかを必ず示します。

- エンコーディング対応のハッシュ結合

  実行時に、ハッシュ結合内の1つのファクト表列がディクショナリ・エンコーディング・データを保持します。実行計画は、ハッシュ結合の右側からの分散のないパラレル・ハッシュ結合か、シリアル・ハッシュ結合計画のどちらかを必ず示します。共通ディクショナリの圧縮率が良好な場合や、パラレル・ハッシュ結合計画では結合グループ対応のハッシュ結合を活用できない場合(並列ブロードキャストなしの計画など)、問合せでは共通ディクショナリにエンコーディング対応ハッシュ結合を使用することもできます。

SQL監視レポートでは、ディクショナリの使用が`Columnar Encodings Observed`フィールドと`Columnar Encodings Leveraged`フィールドで示されます。この統計は累積されます。パラレル・ハッシュ結合の場合、このフィールドには、行ソースの実行に関連するすべてのスレーブ・プロセスから収集した統計が集約されます。IMCU内のローカル・ディクショナリの観点では、この統計には、右側の子行ソースで観測されたエンコーディングIDの数と、結合で利用されたエンコーディングの数が示されます。単一プロセスのハッシュ結合が共通ディクショナリを利用する場合、`Columnar Encodings Leveraged`には、結合で利用されたエンコーディングの数が示されます。

次の表に、`Columnar Encodings Observed`と`Columnar Encodings Leveraged`に示される可能性のある値と、その組合せが意味する内容を示します。

表8-1 SQL監視レポートでの結合グループの使用

観測された列形式エンコーディング	**利用された列形式エンコーディング**	**エンコーディング対応ハッシュ結合の使用**	**結合グループ対応ハッシュ結合の使用**
存在しない	存在しない	なし	なし
正の値	存在しない	なし	なし
正の値	正の値	あり	なし
存在しない	正の値	なし	はい

たとえば、`Columnar Encodings Leveraged`フィールドに4が示されていているときに(並列度が4であると仮定)、`Columnar Encodings Observed`フィールドが存在していない場合、問合せはハッシュ結合に結合グループを利用したことを意味します。`Columnar Encodings Observed`フィールドが4のときに、`Columnar Encodings Leveraged`フィールドが存在しない場合、ディクショナリ・エンコーディングが存在していても、問合せでは使用されなかったことを意味します。

様々な要因によって、問合せでエンコーディング対応ハッシュ結合が使用されなくなることがあります。こうした要因には次のものがあります。

- 共通ディクショナリの圧縮率が最適でない。
- 共通ディクショナリなしで表スキャンから渡された行セットが多すぎることが問合せで観測された。
- 構築側の行の長さが長すぎる。

- 構築側の行がPGAメモリーに収まりきらない。
- 構築側に重複した結合キーがある。

関連項目:

「結合グループの使用の監視」

8.6 結合グループの作成

CREATE INMEMORY JOIN GROUP文を使用して結合グループを定義します。

結合グループの候補は、結合述語で頻繁にペアにされる列です。代表的な例に、ファクト表とディメンション表を結合する列、または表をそれ自体に結合する列などがあります。

CREATE INMEMORY JOIN GROUP文は、結合グループをただちに定義します。これは、そのメタデータがデータ・ディクショナリに表示されることを意味します。データベースでは、共通ディクショナリはすぐには構成されません。正確に述べると、データベースでは、次回、結合グループ内で参照されている表が、IM列ストアに移入されるか再移入されるときに共通ディクショナリが構築されます。

ガイドライン

結合グループを作成、変更または削除すると、通常は、その結合グループ内で参照されている、基にあるすべての表が無効になります。そのため、表を最初に移入する前に結合グループを作成することをお薦めします。

結合グループを作成するには:

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。
2. 次の形式の文を使用して、結合グループを作成します。

```
CREATE INMEMORY JOIN GROUP join_group_name ( table1(col1), table2(col2) );
```

たとえば、次の文は、sales_products_jgという結合グループを作成します。

```
CREATE INMEMORY JOIN GROUP sales_products_jg (sales(prod_id),
products(prod_id));
```

3. オプションで、データ・ディクショナリを問い合せて、結合グループの定義を表示します(出力例も示します)。

```
COL JOINGROUP_NAME FORMAT a18
COL TABLE_NAME FORMAT a8
COL COLUMN_NAME FORMAT a7
SELECT JOINGROUP_NAME, TABLE_NAME, COLUMN_NAME, GD_ADDRESS
FROM   DBA_JOINGROUPS;
JOINGROUP_NAME     TABLE_NA COLUMN_ GD_ADDRESS
------------------ -------- ------- ----------------
SALES_PRODUCTS_JG  SALES    PROD_ID 00000000A142AE50
SALES_PRODUCTS_JG  PRODUCTS PROD_ID 00000000A142AE50
```

4. 結合グループ内で参照されている表を移入します。または、現在移入されている場合は、それらを再移入します。

例8-2 結合グループの使用による問合せの最適化

この例では、データベースにSYSTEMとしてログインしてから、まだIM列ストアに移入されていないsalesおよびproductsのprod_id列で結合グループを作成します。

```
CREATE INMEMORY JOIN GROUP
```

```
  sh.sales_products_jg (sh.sales(prod_id), sh.products(prod_id));
```

IM列ストアへの移入のためにsh.sales表およびsh.products表を有効にします。

```
ALTER TABLE sh.sales INMEMORY;
ALTER TABLE sh.products INMEMORY;
```

次の問合せでは、それらの表がまだIM列ストアに移入されていないことが示されます(出力例が含まれています)。

```
COL OWNER FORMAT a3
COL NAME FORMAT a10
COL STATUS FORMAT a20
SELECT OWNER, SEGMENT_NAME NAME,
       POPULATE_STATUS STATUS
FROM   V$IM_SEGMENTS;
no rows selected
```

両方の表を問い合せて、それらをIM列ストアに移入します。

```
SELECT /*+ FULL(s) NO_PARALLEL(s) */ COUNT(*) FROM sh.sales s;
SELECT /*+ FULL(p) NO_PARALLEL(p) */ COUNT(*) FROM sh.products p;
```

次の問合せでは、それらの表が現在はIM列ストアに移入されていることが示されます(出力例が含まれています)。

```
COL OWNER FORMAT a3
COL NAME FORMAT a10
COL PARTITION FORMAT a13
COL STATUS FORMAT a20
SELECT OWNER, SEGMENT_NAME NAME, PARTITION_NAME PARTITION,
       POPULATE_STATUS STATUS, BYTES_NOT_POPULATED
FROM   V$IM_SEGMENTS;
OWN NAME       PARTITION     STATUS               BYTES_NOT_POPULATED
--- ---------- ------------- -------------------- -------------------
SH  SALES      SALES_Q3_1998 COMPLETED            0
SH  SALES      SALES_Q4_2001 COMPLETED            0
SH  SALES      SALES_Q4_1999 COMPLETED            0
SH  PRODUCTS                 COMPLETED            0
SH  SALES      SALES_Q1_2001 COMPLETED            0
SH  SALES      SALES_Q1_1999 COMPLETED            0
SH  SALES      SALES_Q2_2000 COMPLETED            0
SH  SALES      SALES_Q2_1998 COMPLETED            0
SH  SALES      SALES_Q3_2001 COMPLETED            0
SH  SALES      SALES_Q3_1999 COMPLETED            0
SH  SALES      SALES_Q4_2000 COMPLETED            0
SH  SALES      SALES_Q4_1998 COMPLETED            0
SH  SALES      SALES_Q1_2000 COMPLETED            0
SH  SALES      SALES_Q1_1998 COMPLETED            0
SH  SALES      SALES_Q2_2001 COMPLETED            0
SH  SALES      SALES_Q2_1999 COMPLETED            0
SH  SALES      SALES_Q3_2000 COMPLETED            0
```

DBA_JOINGROUPSを問い合せ、結合グループに関する情報を取得します(出力例が含まれています)。

```
COL JOINGROUP_NAME FORMAT a18
COL TABLE_NAME FORMAT a8
COL COLUMN_NAME FORMAT a7
SELECT JOINGROUP_NAME, TABLE_NAME, COLUMN_NAME, GD_ADDRESS
FROM   DBA_JOINGROUPS;
JOINGROUP_NAME     TABLE_NA COLUMN_ GD_ADDRESS
------------------ -------- ------- ----------------
SALES_PRODUCTS_JG  SALES    PROD_ID 00000000A142AE50
SALES_PRODUCTS_JG  PRODUCTS PROD_ID 00000000A142AE50
```

上の出力では、結合グループsales_products_jgが同じ共通ディクショナリ・アドレスで結合されていることが示されていま

す。

関連項目:

- CREATE INMEMORY JOIN GROUP文について学習するには、『Oracle Database SQL言語リファレンス』を参照してください
- DBA_JOINGROUPSビューについて学習するには、『Oracle Databaseリファレンス』を参照

8.7 結合グループの使用の監視

問合せで結合グループが使用されているかどうかを判断するには、グラフィカルなSQL監視レポート(推奨)を使用することも、DBMS_SQLTUNE.REPORT_SQL_MONITOR_XMLファンクションを使用するSQL問合せを使用することもできます。

「ハッシュ結合で共通ディクショナリ・エンコーディングが使用される場合」では、SQL監視の出力を解釈する方法について説明しています。

前提条件

結合グループを監視するには、次の前提条件を満たしている必要があります。

- 結合グループが存在する必要があります。
- 結合グループによって参照される列が、結合グループの作成後に移入されている必要があります。
- 結合グループを使用可能な結合問合せを実行する必要があります。

結合グループの使用を監視するには:

1. 必要な権限があるユーザーとしてデータベースにログインします。
2. 次のようにして、SQL IDを格納するSQL*Plus変数を作成します。

```
VAR b_sqlid VARCHAR2(13)
```

3. 結合グループ内の列で結合する問合せを実行します。
4. 次のどちらかの方法を使用します。
    - グラフィカルなSQL監視レポート

      SQL監視レポートはEnterprise Managerで使用できます。SQL*Plusでは、次のようにDBMS_SQL_MONITOR.REPORT_SQL_MONITORを使用すると、SQL監視レポートを生成できます。

```
SET TRIMSPOOL ON
SET TRIM ON
SET PAGES 0
SET LINESIZE 1000
SET LONG 1000000
SET LONGCHUNKSIZE 1000000
SPOOL /tmp/long_sql.htm
SELECT DBMS_SQL_MONITOR.REPORT_SQL_MONITOR(
         sql_id       => :b_sqlid,
         report_level => 'ALL',
         TYPE         => 'active')
FROM   DUAL;
SPOOL OFF
```

      ブラウザでレポートにアクセスし、ハッシュ結合の双眼鏡アイコンをクリックして、結合グループの統計を表示しま

す。

- コマンドラインでの問合せ

  問合せで、DBMS_SQLTUNE.REPORT_SQL_MONITOR_XMLファンクションを使用します。次に例を示します。

```
SELECT
  encoding_hj.rowsource_id row_source_id,
    CASE
      WHEN encoding_hj.encodings_observed IS NULL
      AND encoding_hj.encodings_leveraged IS NOT NULL
      THEN
        'join group was leveraged on ' || encoding_hj.encodings_leveraged
|| ' processes'
      ELSE
        'join group was NOT leveraged'
    END columnar_encoding_usage_info
FROM
  (SELECT DBMS_SQLTUNE.REPORT_SQL_MONITOR_XML(session_id=>-
1,sql_id=>:b_sqlid).
          EXTRACT(q'#//operation[@name='HASH JOIN' and @parent_id]#')
xmldata
   FROM   DUAL) hj_operation_data,
  XMLTABLE('/operation'
    PASSING hj_operation_data.xmldata
    COLUMNS
     "ROWSOURCE_ID"        NUMBER PATH '@id',
     "ENCODINGS_LEVERAGED" NUMBER PATH 'rwsstats/stat[@id="9"]',
     "ENCODINGS_OBSERVED"  NUMBER PATH 'rwsstats/stat[@id="10"]')
encoding_hj;
```

8.7.1 SQL監視レポートを使用した結合グループの監視: 例

グラフィカルなSQL監視レポートを使用して、問合せで結合グループが利用されたかどうかを判断できるようにします。

この例では、sh.products表およびsh.sales表のprod_id列で結合グループを作成してから、これらの表をこの列で結合します。shアカウントに管理権限を付与します。

例8-3 SQL監視レポートを使用した結合グループの監視

1. SQL*Plusで、ユーザーshとしてデータベースにログインします。
2. SQL IDを格納するためのSQL*Plus変数を次のように作成します。

```
VAR b_sqlid VARCHAR2(13)
```

3. 次のようにINMEMORY属性をsh.products表およびsh.sales表に適用します。

```
ALTER TABLE sales NO INMEMORY;
ALTER TABLE products NO INMEMORY;
ALTER TABLE sales INMEMORY MEMCOMPRESS FOR QUERY;
ALTER TABLE products INMEMORY MEMCOMPRESS FOR QUERY;
```

4. prod_idで結合グループを作成します。

```
CREATE INMEMORY JOIN GROUP jgrp_products_sales (products(prod_id),
sales(prod_id));
```

5. それらの表をスキャンしてIM列ストアに移入します。

```
SELECT /*+ FULL(s) */ COUNT(*) FROM sales s;
SELECT /*+ FULL(p) */ COUNT(*) FROM products p;
```

6. prod_id列で結合する問合せを実行してから、商品売上を集計します。

```
SELECT /*+ USE_HASH(sales) LEADING(products sales) MONITOR */ products.prod_id,
       products.prod_category_id, SUM(sales.amount_sold)
FROM   products, sales
WHERE  products.prod_id = sales.prod_id
GROUP BY products.prod_category_id, products.prod_id;
```

7. DBMS_SQLTUNE.REPORT_SQL_MONITORを使用して、HTMLベースのSQL監視レポートを生成します。

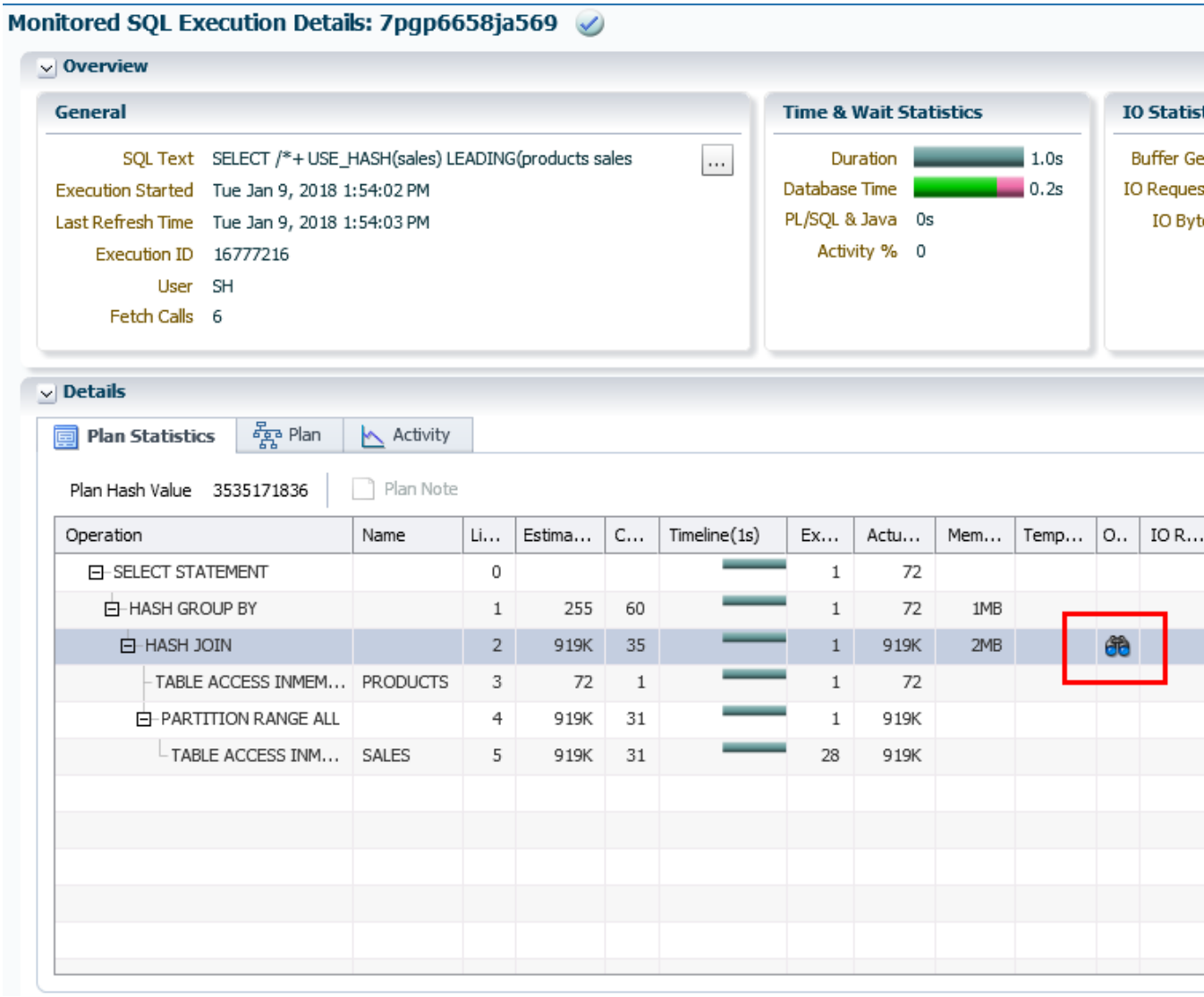
たとえば、次に示す内容のSQLスクリプトを作成して、SQL*Plusで実行します。

```
SET TRIMSPOOL ON
SET TRIM ON
SET PAGES 0
SET LINESIZE 1000
SET LONG 1000000
SET LONGCHUNKSIZE 1000000
SPOOL /tmp/jg_report.htm
SELECT DBMS_SQL_MONITOR.REPORT_SQL_MONITOR(
         sql_id       => :b_sqlid,
         report_level => 'ALL',
         TYPE         => 'active')
FROM   DUAL;
SPOOL OFF
```

8. ブラウザで、HTMLレポートを開きます。

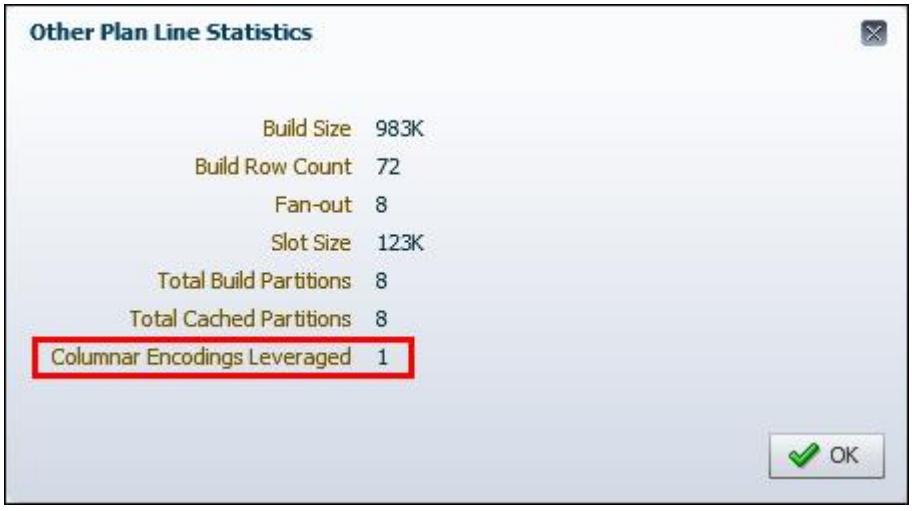
次のレポート例は、結合の実行計画を示しています。ハッシュ結合の双眼鏡アイコンで、追加の統計が示されるウィンドウを開きます。

図8-4「監視されたSQL実行の詳細」ページ



9. 双眼鏡アイコンをクリックして、結合グループの統計を表示するウィンドウを開きます。

次に、統計が表示されたウィンドウの例を示します。



`Columnar Encodings Leveraged`が正の値であり、`Columnar Encodings Observed`が存在していないことから、結合グループが利用されたことがわかります。

10. 必要に応じて、この例の後でクリーン・アップを実行します。

```
DROP INMEMORY JOIN GROUP jgrp_products_sales;
ALTER TABLE sales NO INMEMORY;
```

```
ALTER TABLE products NO INMEMORY;
```

関連項目:

- 「ハッシュ結合で共通ディクショナリ・エンコーディングが使用される場合」
- DBMS_SQLTUNE.REPORT_SQL_MONITOR_XML関数について学習するには、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照
- V$SESSIONビューについて学習するには、『Oracle Databaseリファレンス』を参照

8.7.2 コマンドラインからの結合グループの監視: 例

コマンドライン・ツールを使用して、問合せで結合グループが利用されたかどうかを判断できるようにします。

この例では、sh.products表およびsh.sales表のprod_id列で結合グループを作成してから、これらの表をこの列で結合します。shアカウントに管理権限を付与します。

例8-4 コマンドラインからの結合グループの監視

1. shとしてデータベースにログインします。
2. SQL IDを格納するためのSQL*Plus変数を次のように作成します。

```
VAR b_sqlid VARCHAR2(13)
```

3. 次のようにINMEMORY属性をsh.products表およびsh.sales表に適用します。

```
ALTER TABLE sales NO INMEMORY;
ALTER TABLE products NO INMEMORY;
ALTER TABLE sales INMEMORY MEMCOMPRESS FOR QUERY;
ALTER TABLE products INMEMORY MEMCOMPRESS FOR QUERY;
```

4. prod_idで結合グループを作成します。

```
CREATE INMEMORY JOIN GROUP jgrp_products_sales (products(prod_id),
sales(prod_id));
```

5. それらの表をスキャンしてIM列ストアに移入します。

```
SELECT /*+ FULL(s) */ COUNT(*) FROM sales s;
SELECT /*+ FULL(p) */ COUNT(*) FROM products p;
```

6. prod_id列で結合する問合せを実行してから、商品売上を集計します。

```
SELECT /*+ USE_HASH(sales) LEADING(products sales) MONITOR */ products.prod_id,
       products.prod_category_id, SUM(sales.amount_sold)
FROM   products, sales
WHERE  products.prod_id = sales.prod_id
GROUP BY products.prod_category_id, products.prod_id;
```

7. 前述の集計問合せのSQL IDを取得します。

```
BEGIN
  SELECT PREV_SQL_ID
    INTO :b_sqlid
  FROM   V$SESSION
  WHERE  SID=USERENV('SID');
END;
```

8. DBMS_SQLTUNE.REPORT_SQL_MONITOR_XMLを使用して、データベースで結合グループが使用されたかどうかを判断します。

   たとえば、次の問合せを実行します。

```
COL row_source_id FORMAT 999
COL columnar_encoding_usage_info FORMAT A40
SELECT
  encoding_hj.rowsource_id row_source_id,
    CASE
      WHEN encoding_hj.encodings_observed IS NULL
      AND encoding_hj.encodings_leveraged IS NOT NULL
      THEN
        'join group was leveraged on ' || encoding_hj.encodings_leveraged || '
processes'
      ELSE
        'join group was NOT leveraged'
    END columnar_encoding_usage_info
FROM
  (SELECT DBMS_SQLTUNE.REPORT_SQL_MONITOR_XML(session_id=>-1,sql_id=>:b_sqlid).
          EXTRACT(q'#//operation[@name='HASH JOIN' and @parent_id]#') xmldata
   FROM   DUAL
  ) hj_operation_data,
  XMLTABLE('/operation'
    PASSING hj_operation_data.xmldata
    COLUMNS
     "ROWSOURCE_ID"       NUMBER PATH '@id',
     "ENCODINGS_LEVERAGED" NUMBER PATH 'rwsstats/stat[@id="9"]',
     "ENCODINGS_OBSERVED"  NUMBER PATH 'rwsstats/stat[@id="10"]'
     ) encoding_hj;
```

   次の出力例は、問合せで結合グループが利用されたことを示しています。

```
ROW_SOURCE_ID COLUMNAR_ENCODING_USAGE_INFO
------------- ----------------------------------------
            2 join group was leveraged on 1 processes
```

9. 必要に応じて、例の後にクリーン・アップを実行します。

```
DROP INMEMORY JOIN GROUP jgrp_products_sales;
ALTER TABLE sales NO INMEMORY;
ALTER TABLE products NO INMEMORY;
```

関連項目:

- 「ハッシュ結合で共通ディクショナリ・エンコーディングが使用される場合」
- DBMS_SQLTUNE.REPORT_SQL_MONITOR_XML関数について学習するには、『Oracle Database PL/SQL パッケージおよびタイプ・リファレンス』を参照
- V$SESSIONビューについて学習するには、『Oracle Databaseリファレンス』を参照

9 集計の最適化

Database In-Memoryは、集計と算術の時間を短縮する最適化を提供します。

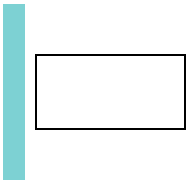
9.1 VECTOR GROUP BYを使用したインメモリー集計の最適化

Oracle Database 12cリリース1 (12.1.0.2)以降、**インメモリー集計(IM集計)**では、問合せにおいて、スキャン処理しながら集計できます。

9.1.1 IM集計について

IM集計では、集計に関与する問合せブロックが最適化され、1つの大きい表から複数の小さい表に結合されます。

KEY VECTORおよびVECTOR GROUP BY操作では、結合および集計のために効率的な配列が使用されます。オプティマイザにより、コストに基づくGROUP BY操作のためにVECTOR GROUP BYが選択されます。オプティマイザでは、GROUP BY ROLLUP、GROUPING SETSまたはCUBE操作のためにVECTOR GROUP BY集計が選択されることはありません。



ノート:

IM 集計は、ベクター集計および VECTOR GROUP BY 集計とも呼ばれます。

IM集計では、INMEMORY_SIZEがゼロ以外の値に設定される必要があります。ただし、IM集計では、参照される表がIM列ストアに移入されている必要はありません。

関連項目:

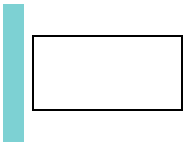
- 「データベースに対するIM列ストアの有効化」
- SQL集計についてさらに学習するには、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照

9.1.2 IM集計の目的

IM集計では、大きい表で実行される、行ごとの処理を高速化するために、小さい表が前処理されます。

一般的な分析問合せでは、ファクト表から集計し、それをディメンション表に結合します。このタイプの問合せは、大量のデータをスキャンし、オプションでフィルタ処理を行い、1列から40列までのGROUP BYを実行します。ファクト表での最初の集計で、ほとんどの行が処理されます。

Oracle Database 12cより前は、GROUP BY操作はHASHおよびSORTのみでした。VECTOR GROUP BYは、ディメンション表とファクト表との結合をフィルタに変換する、追加のコストベースの変換です。データベースでは、ファクト表のスキャン中にこのフィルタを適用できます。結合では、ブルーム・フィルタに似たキー・ベクターが使用され、集計では、VECTOR GROUP BYが使用されます。



ノート:

ベクター変換は IM 列ストアから独立していますが、SIMD ベクター処理によってそれらをインメモリー・データに

非常に効率的に適用できます。

IM集計により、ベクトル結合およびGROUP BY操作を、大きな表のスキャンと同時に行うことができるようになります。したがって、これらの操作では、スキャンしながら集計されるため、表のスキャンおよび結合操作の完了を待つ必要はありません。IM集計では、CPU使用率、特にCPUキャッシュが最適化されます。

IM集計により、問合せのパフォーマンスを大幅に改善できます。データベースでは、動的にレポート概要を作成し、ファクト表のスキャン中にレポート詳細を入力できます。

関連項目:

- 「CPUアーキテクチャ: SIMDベクター処理」
- 問合せ変換についてさらに学習するには、Oracle Database SQLチューニング・ガイドを参照

9.1.2.1 IM集計が役立つ場合

IM集計により、比較的小さい表を比較的大きい表に結合してファクト表内のデータを集計する問合せのパフォーマンスが向上します。これは通常、スター・クエリーまたはスノーフレーク・クエリーで起こります。

行ストア表およびIM列ストア内の表の両方に、IM集計が役立ちます。

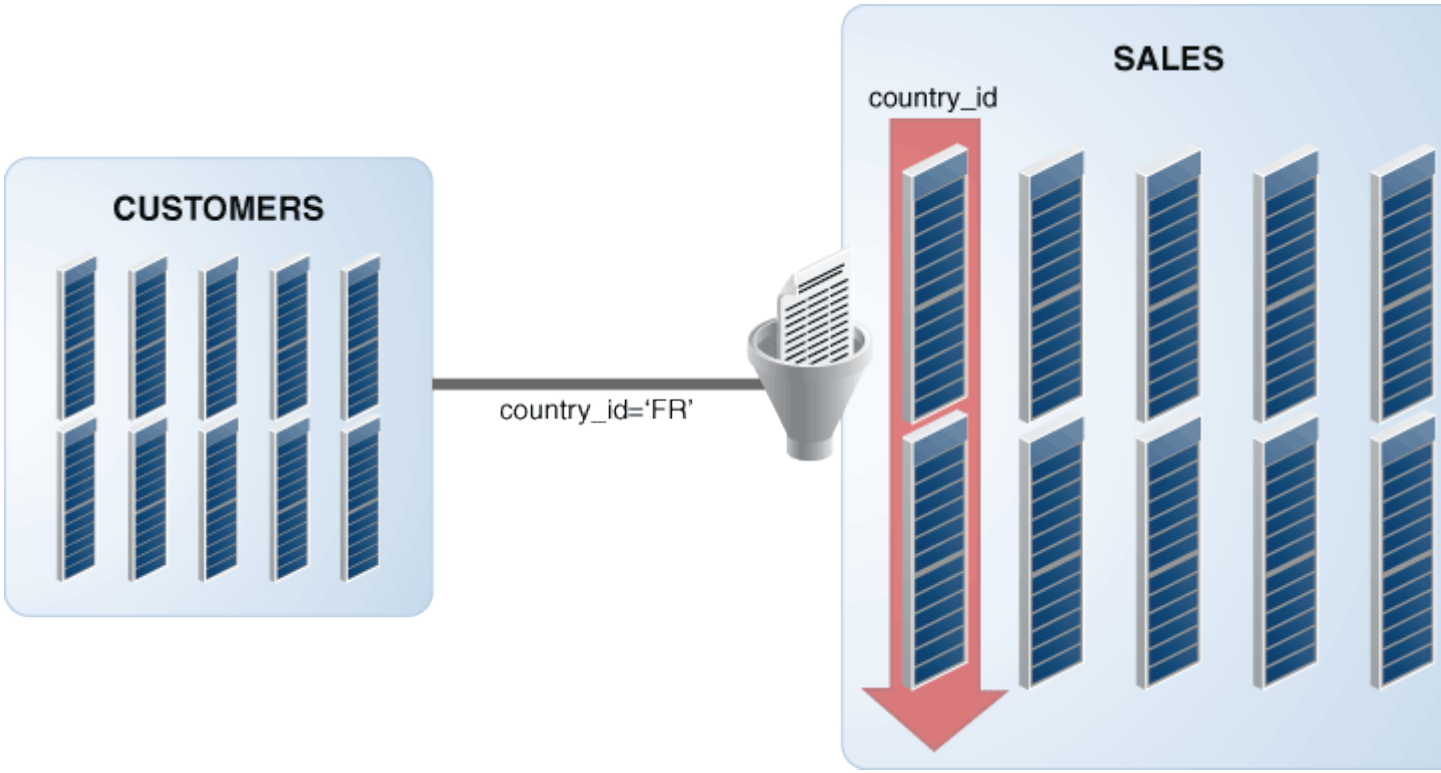
例9-1 VECTOR GROUP BY

customersディメンション表とsalesファクト表の結合を実行する、次のような問合せを想定します。

```
SELECT c.customer_id, s.quantity_sold, s.amount_sold
FROM   customers c, sales s
WHERE  c.customer_id = s.customer_id
AND    c.country_id = 'FR';
```

両方の表がIM列ストアに移入される場合、データベースでは、SIMDベクター処理を使用して、行セットをスキャンし、フィルタを適用できます。次の図では、ベクトル結合が問合せでどのように使用されるかを示します。オプティマイザにより、customers表の述語、c.country_id='FR'が、salesファクト表上のフィルタに変換されます。フィルタはcountry_id='FR'です。salesは列形式で格納されるため、問合せでは、結果を決定するために1列のスキャンしか必要になりません。

図9-1 インメモリー列ストアを使用したベクトル結合



9.1.2.2 IM集計が役立たない場合

IM集計は、十分なシステム・リソースが存在する場合に、特定のスター・クエリーに役立ちます。他の問合せには、ほとんどまたはまったく利益がない場合があります。

VECTOR GROUP BY集計が不利な状況

具体的に述べると、VECTOR GROUP BY集計は、次のシナリオにおいてはパフォーマンス向上に役立ちません。

- 2つの非常に大きな表の間で結合が実行される。

  デフォルトでは、比較的小さい表が比較的大きい表に結合される場合のみ、オプティマイザでVECTOR GROUP BY変換が選択されます。

- ディメンションに、20億以上の行が含まれる。

  ディメンションに20億以上の行が含まれる場合、VECTOR GROUP BY変換は使用されません。

- システムに十分なメモリーがない。

  IM列ストアを使用するほとんどのデータベースには、IM集計が役立ちます。

9.1.3 IM集計の仕組み

一般的な分析問合せでは、複数の処理ステージ間で行が分散されます。

次のようなステージがあります。

1. 表のフィルタ処理と行セットの生成
2. 行セットの結合
3. 行の集計

VECTOR GROUP BY変換では、様々なステージでの処理を組み合せて、結合をフィルタに変換し、ファクト表をスキャンしなが

ら集計します。

ステージ間の作業単位をデータ・フロー演算子(DFO)と呼びます。VECTOR GROUP BY集計では、ディメンションごとにDFOを使用して、キー・ベクター構造と一時表を作成します。ファクト表のメジャー列を集計する場合、データベースではこのキー・ベクターを使用して、ファクト結合キーをその稠密グループ化キーに変換します。遅延マテリアライズ・ステップは、稠密グループ化キーで一時表に結合します。

9.1.3.1 オプティマイザでIM集計が選択される場合

オプティマイザでは、キー・ベクター(つまり、非重複結合キー)のサイズ、非重複グループ化キーの数、およびその他の要因に基づいて、ベクター変換を使用するかどうかが決定されます。

ディメンション結合キーのカーディナリティが低い場合に、オプティマイザはこの変換を選択する傾向があります。Oracle Databaseは、次の条件が満たされた場合、VECTOR GROUP BY集計を使用してデータ集計を実行します。

- 問合せまたは副問合せは、ファクト表からのデータを集計し、ファクト表を1つ以上のディメンションに結合します。

  これらのファクト表が結合のみを介してディメンションに接続されているという想定では、同じディメンションに結合された複数のファクト表もサポートされます。この場合、VECTOR GROUP BYはファクト表を別々に集計し、グループ化キー上で結果を結合します。

- ディメンションとファクト表は、結合列を介してのみ互いに接続されます。

  特に、問合せには、複数のディメンションを通じて、またはディメンションとファクト表から、列を参照する述語が他にないようにする必要があります。問合せが2つ以上の表の間の結合を実行し、結果をファクトに結合する場合、VECTOR GROUP BY集計は、複数のディメンションを1つのディメンションとみなします。

ノート:

問合せブロックのヒントまたは表ヒントを使用して、問合せに対して VECTOR GROUP BY 集計を使用するようにデータベースに指示できます。

VECTOR GROUP BY集計は、次の場合をサポートしていません。

- 複数のディメンションの間、またはディメンション表とファクト表の間の半結合と逆結合
- 複数のディメンションの間の等価結合
- DISTINCT関数を使用して実行される集計

ノート:

ブルーム・フィルタと VECTOR GROUP BY 集計は、互いに矛盾します。そのため、問合せでブルーム・フィルタを使用して行セットを結合する場合は、この問合せの処理に VECTOR GROUP BY 集計を適用できません。

関連項目:

SQL集計についてさらに学習するには、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照

9.1.3.2 キー・ベクター

キー・ベクターは、稠密結合キーと稠密グループ化キー間をマップするデータ構造です。

稠密キーは、ネイティブ整数として格納され、値の範囲を持つ数値キーです。稠密結合キーは、結合列が特定のファクト表またはディメンションから取得されるすべての結合キーを表します。稠密グループ化キーは、グループ化列が特定のファクト表またはディメンションから取得されるすべてのグループ化キーを表します。キー・ベクターは、高速参照を可能にします。

例9-2 キー・ベクター

hr.locations表に、次のようなcountry_idの値があるとします(結果のうち冒頭の一部のみを示しています)。

```
SQL> SELECT country_id FROM locations;

CO
--
IT
IT
JP
JP
US
US
US
US
CA
CA
CN
```

複合分析問合せは、フィルタWHERE country_id='US'をlocations表に適用します。このフィルタのキー・ベクターは、次の1次元配列のようになります。

```
0
0
0
0
1
1
1
1
0
0
0
```

前の配列では、1はcountry_id='US'の稠密グループ化キーです。0の値は、このフィルタに一致しないlocationsの行を示します。問合せでフィルタWHERE country_id IN ('US','JP')を使用する場合、配列は次のようになります。ここで、2はJPの稠密グループ化キー、1はUSの稠密グループ化キーです。

```
0
0
2
2
1
1
1
1
0
0
0
```

9.1.3.3 インメモリー集計の2つのフェーズ

一般に、VECTOR GROUP BY集計では、各ディメンションが順々に処理されてから、ファクト表が処理されます。

インメモリー集計を実行する場合、データベースは次のように処理を行います。

1. 各ディメンションを次の順序で処理します。

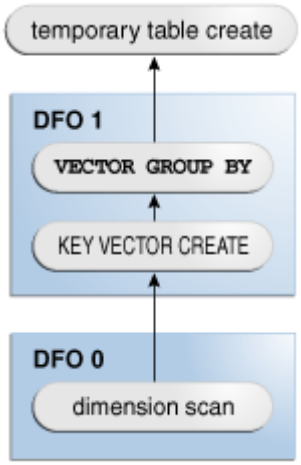
    a. 一意の稠密グループ化キーを検索します。

    b. キー・ベクターを作成します。

    c. 一時表(CURSOR DURATION MEMORY)を作成します。

    次の図では、DFO 0のディメンション表のスキャンで開始し、一時表の作成で終了する、このフェーズのステップを示します。最も単純な形式のパラレルGROUP BYまたは結合処理では、データベースは自身のDFO内で各結合またはGROUP BYを処理します。

    図9-2 インメモリー集計のフェーズ1



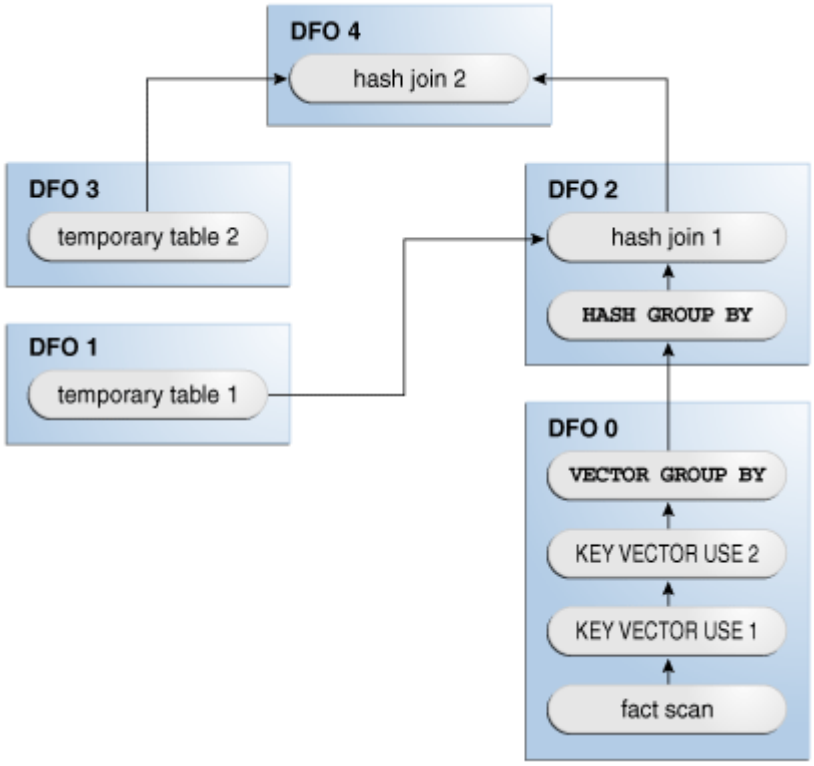
2. ファクト表を処理します。

    a. 前のフェーズで作成したキー・ベクターを使用して、すべての結合と集計を処理します。

    b. 結果を各一時表に再び結合します。

    図9-3に、2つのディメンションがあるファクト表の結合のフェーズ2を示します。DFO 0では、データベースはファクト表の全スキャンを実行し、各ディメンションのキー・ベクターを使用して不一致の行をフィルタで除外します。DFO 2は、DFO 0の結果をDFO 1と結合します。DFO 4は、DFO 2の結果をDFO 3と結合します。

    図9-3 インメモリー集計のフェーズ2



9.1.3.4 IM集計: シナリオ

この項では、VECTOR GROUP BY集計の仕組みの概念的な例を示します。

ノート:

シナリオに使用しているスキーマ表はサンプルではなく、表示されている実行計画も実際のものではありません。

9.1.3.4.1 スター・スキーマのサンプル分析問合せ

このシナリオのサンプル・スター・スキーマには、sales_onlineファクト表と、2つのディメンション表geographyおよびproductsが含まれます。

geographyの各行は、geog_id列で一意に識別されます。productsの各行は、prod_id列で一意に識別されます。sales_onlineの各行は、geog_id列、prod_id列および販売量で一意に識別されます。

表9-1 geography表のサンプル行

国	州	市	geog_id
USA	WA	seattle	2
USA	WA	spokane	3
USA	CA	SF	7
USA	CA	LA	8

表9-2 products表のサンプル行

製造者	カテゴリ	サブカテゴリ	prod_id
Acme	スポーツ	バイク	4
Acme	スポーツ	ボール	3
Acme	電気	電球	1
Acme	電気	スイッチ	8

表9-3 sales_online表のサンプル行

prod_id	geog_id	量
8	1	100
9	1	150
8	2	100
4	3	110
2	30	130
6	20	400
3	1	100
1	7	120
3	8	130
4	3	200

仕事上でマネージャに、「各サブカテゴリのAcme製品が、ワシントンでオンライン販売された数と、カリフォルニアで販売された数はいくつだったか」と聞かれたとします。この質問に答えるには、sales_onlineファクト表の分析問合せでproductsおよびgeographyディメンション表を次のように結合します。

```
SELECT p.category, p.subcategory, g.country, g.state, SUM(s.amount)
FROM   sales_online s, products p, geography g
WHERE  s.geog_id = g.geog_id
AND    s.prod_id = p.prod_id
AND    g.state IN ('WA','CA')
AND    p.manuf = 'ACME'
```

```
GROUP BY category, subcategory, country, state
```

9.1.3.4.2 ステップ1: geographyディメンションのキー・ベクターと一時表の作成

この問合せのVECTOR GROUP BY集計の最初のフェーズでは、データベースはワシントンまたはカリフォルニア州の市について、各市/州の組合せに対する稠密グループ化キーを作成します。

表9-6では、1はUSA,WAグループ化キーで、2はUSA,CAグループ化キーです。

表9-4 geographyの稠密グループ化キー

国	州	市	geog_id	dense_gr_key_geog
USA	WA	seattle	2	1
USA	WA	spokane	3	1
USA	CA	SF	7	2
USA	CA	LA	8	2

geography表のキー・ベクターは、表9-5の最後の列で表される配列のようになります。値は、geography稠密グループ化キーです。したがって、キー・ベクターは、sales_onlineのどの行がgeography.stateフィルタ基準(CAまたはWAの州の販売)を満たし、各行がどの国/州グループ(USA、WAグループまたはUSA、CAグループ)に属するかを示します。

表9-5 オンライン販売

prod_id	geog_id	量	geographyのキー・ベクター
8	1	100	0
9	1	150	0
8	2	100	1
4	3	110	1
2	30	130	0
6	20	400	0
3	1	100	0
1	7	120	2

prod_id	geog_id	量	geographyのキー・ベクター
3	8	130	2
4	3	200	1

内部的に、データベースは次のような一時表を作成します。

```
CREATE TEMPORARY TABLE tt_geography AS
SELECT MAX(country), MAX(state), KEY_VECTOR_CREATE(...) dense_gr_key_geog
FROM   geography
WHERE  state IN ('WA','CA')
GROUP BY country, state
```

表9-6に、tt_geography一時表の行を示します。USA、WAの組合せに対する稠密グループ化キーは1で、USA、CAの組合せに対する稠密グループ化キーは2です。

表9-6 tt_geography

国	州	dense_gr_key_geog
USA	WA	1
USA	CA	2

9.1.3.4.3 ステップ2: productsディメンションのキー・ベクターと一時表の作成

データベースでは、Acme製品の個々のカテゴリ/サブカテゴリの組合せに対して稠密グループ化キーが作成されます。

たとえば、表9-7では、4がAcme electric switchの稠密グループ化キーです。

表9-7 products表のサンプル行

製造者	カテゴリ	サブカテゴリ	prod_id	dense_gr_key_prod
Acme	スポーツ	バイク	4	1
Acme	スポーツ	ボール	3	2
Acme	電気	電球	1	3
Acme	電気	スイッチ	8	4

products表のキー・ベクターは、表9-8の最後の列で表される配列のようになります。値は、products稠密グループ化キーを表します。たとえば、4はAcme electric switchのオンライン販売を表します。したがって、キー・ベクターは、sales_onlineのどの行がproductsフィルタ基準(Acme製品の販売)を満たすかを示します。

表9-8 キー・ベクター

prod_id	geog_id	量	productsのキー・ベクター

prod_id	geog_id	量	productsのキー・ベクター
8	1	100	4
9	1	150	0
8	2	100	4
4	3	110	1
2	30	130	0
6	20	400	0
3	1	100	2
1	7	120	3
3	8	130	2
4	3	200	1

内部的に、データベースは次のような一時表を作成します。

```
CREATE TEMPORARY TABLE tt_products AS
SELECT MAX(category), MAX(subcategory), KEY_VECTOR_CREATE(...) dense_gr_key_prod
FROM   products
WHERE  manuf = 'ACME'
GROUP BY category, subcategory
```

表9-9に、この一時表の行を示します。

表9-9 tt_products

カテゴリ	サブカテゴリ	dense_gr_key_prod
スポーツ	バイク	1
スポーツ	ボール	2
電気	電球	3
電気	スイッチ	4

9.1.3.4.4 ステップ3: キー・ベクター問合せ変換

このフェーズでは、データベースはファクト表を処理します。

オプティマイザは、元の問合せを、キー・ベクターにアクセスする次の同等の問合せに変換します。

```
SELECT KEY_VECTOR_PROD(prod_id),
       KEY_VECTOR_GEOG(geog_id),
       SUM(amount)
FROM   sales_online
WHERE  KEY_VECTOR_PROD_FILTER(prod_id) IS NOT NULL
AND    KEY_VECTOR_GEOG_FILTER(geog_id) IS NOT NULL
GROUP BY KEY_VECTOR_PROD(prod_id), KEY_VECTOR_GEOG(geog_id)
```

前の変換は、はるかに複雑な内部SQLの正確なレンディションではなく、基本概念を示すように設計された概念的な表現です。

9.1.3.4.5 ステップ4: ファクト表からの行フィルタ

このフェーズでは、グループ化キーの各組合せの販売量を取得します。

データベースは、キー・ベクターを使用して、ファクト表から不要な行をフィルタで除外します。表9-10では、最初の3列が`sales_online`表を表します。最後の2列は、`geography`および`products`表の稠密グループ化キーを提供します。

表9-10 sales_online表の稠密グループ化キー

prod_id	geog_id	量	dense_gr_key_prod	dense_gr_key_geog
7	1	100	4	
9	1	150		
8	2	100	4	1
4	3	110	1	1
2	30	130		
6	20	400		
3	1	100	2	
1	7	120	3	2
3	8	130	2	2
4	3	200	1	1

表9-11に示すように、データベースは`sales_online`から、どちらの稠密グループ化キーもnull値でない行のみを取得しており、すべてのフィルタ基準を満たす行が示されています。

表9-11 sales_online表からフィルタされた行

geog_id	prod_id	量	dense_gr_key_prod	dense_gr_key_geog
2	8	100	4	1
3	4	110	1	1
3	4	200	1	1
7	1	120	3	2
8	3	130	2	2

9.1.3.4.6 ステップ5: 配列を使用した集計

データベースでは、多次元配列を使用して集計を実行します。

表9-12では、geographyグループ化キーは水平で、productsグループ化キーは垂直です。データベースは、各稠密グループ化キーの組合せの交差部分の値を加算します。たとえば、geographyグループ化キー1とproductsグループ化キー1の交差部分では、110と200の合計は310です。

表9-12 集計配列

dgkp/dgkg	1	2
1	110,200	
2		130
3		120
4	100	

9.1.3.4.7 ステップ6: 一時表への再結合

処理の最終ステージでは、データベースが稠密グループ化キーを使用して行を一時表に再結合し、地域とカテゴリの名前を取得します。

結果は次のようになります。

```
CATEGORY SUBCATEGORY COUNTRY STATE AMOUNT
-------- ----------- ------- ----- ------
electric bulb        USA     CA    120
electric switch      USA     WA    100
sport    ball        USA     CA    130
sport    bike        USA     WA    310
```

9.1.4 IM集計の制御

IM集計は、オプティマイザと統合されます。

新しいSQLまたは初期化パラメータが必要となります。IM集計には、追加の索引、外部キーまたはディメンションは不要です。

IM集計を手動で制御するために、次のヒントのペアを使用できます。

- 問合せブロック・ヒント

  VECTOR_TRANSFORMでは、コストにかかわらず、指定した問合せブロックに対してベクター変換を有効にします。NO_VECTOR_TRANSFORMは、指定した問合せブロックからのベクター変換を無効にします。

- 表ヒント

  次のヒントのペアを使用できます。

  - VECTOR_TRANSFORM_FACTでは、ベクター変換で生成されたファクト表に、指定したFROM式が含まれます。NO_VECTOR_TRANSFORM_FACTでは、ベクター変換で生成されたファクト表から、指定したFROM式が除外されます。
  - VECTOR_TRANSFORM_DIMSでは、ベクター変換で生成された有効なディメンションに、指定したFROM式が含まれます。NO_VECTOR_TRANSFORM_DIMSでは、ベクター変換で生成された有効なディメンションから、指定したFROM式が除外されます。

関連項目:

VECTOR_TRANSFORM_FACTおよびVECTOR_TRANSFORM_DIMSヒントについてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照してください。

9.1.5 インメモリー集計: 例

この例では、仕事上の問いは「各カレンダ年に販売されたカテゴリごとの製品数はいくつか」というものです。

times、productsおよびsales表を結合する、次のような問合せを記述します。

```
SELECT t.calendar_year, p.prod_category, SUM(quantity_sold)
FROM   times t, products p, sales s
WHERE  t.time_id = s.time_id
AND    p.prod_id = s.prod_id
GROUP BY t.calendar_year, p.prod_category;
```

例9-3 VECTOR GROUP BY実行計画

次の例に、現在のカーソルに含まれる実行計画を示します。ステップ4と8では、ディメンション表timesおよびproductsのキー・ベクターを作成しています。ステップ17と18では、前に作成したキー・ベクターを使用しています。ステップ15では、VECTOR GROUP BY操作を行っています。

```
----------------------------------------------------------
Execution Plan
----------------------------------------------------------
Plan hash value: 2093829546
-------------------------------------------------------------------------------------------
----------------------------
|Id| Operation                                 | Name
|Rows|Bytes|Cost(%CPU)|Time |Pstart| Pstop |
-------------------------------------------------------------------------------------------
----------------------------
```

```
|0 |SELECT STATEMENT                          |                          | 18 |
1116|302(90)|0:00:01|       |       |
|1 | TEMP TABLE TRANSFORMATION                |                          |    |     |
|       |       |       |
|2 |  LOAD AS SELECT (CURSOR DURATION MEMORY)|SYS_TEMP_0FD9D6608_F6A13|    |     |
|       |       |       |
|3 |   HASH GROUP BY                          |                          |  5 |  80 | 3
(67)|0:00:01|       |       |
|4 |    KEY VECTOR CREATE BUFFERED            | :KV0000                  |  5 |  80 | 2
(50)|0:00:01|       |       |
|5 |     TABLE ACCESS INMEMORY FULL           | TIMES                    |1826|21912| 2
(50)|0:00:01|       |       |
|6 |  LOAD AS SELECT (CURSOR DURATION MEMORY)|SYS_TEMP_0FD9D6607_F6A13|    |     |
|       |       |       |
|7 |   HASH GROUP BY                          |                          |  5 | 125 | 2
(50)|0:00:01|       |       |
|8 |    KEY VECTOR CREATE BUFFERED            | :KV0001                  |  5 | 125 | 1
(0)|0:00:01|       |       |
|9 |     TABLE ACCESS INMEMORY FULL           | PRODUCTS                 | 72 |1512 | 1
(0)|0:00:01|       |       |
|10|  HASH GROUP BY                           |                          | 18 |1116
|297(91)|0:00:01|       |       |
|11|   HASH JOIN                              |                          | 18 |1116
|296(91)|0:00:01|       |       |
|12|    HASH JOIN                             |                          | 18 | 666
|294(91)|0:00:01|       |       |
|13|     TABLE ACCESS FULL                    |SYS_TEMP_0FD9D6608_F6A13|  5 |  80 | 2
(0)|0:00:01|       |       |
|14|     VIEW                                 | VW_VT_0737CF93           | 18 | 378
|291(92)|0:00:01|       |       |
|15|      VECTOR GROUP BY                     |                          | 18 | 414
|291(92)|0:00:01|       |       |
|16|       HASH GROUP BY                      |                          | 18 | 414
|291(92)|0:00:01|       |       |
|17|        KEY VECTOR USE                    | :KV0000                  |918K| 20M
|285(92)|0:00:01|       |       |
|18|         KEY VECTOR USE                   | :KV0001                  |918K| 16M
|284(92)|0:00:01|       |       |
|19|          PARTITION RANGE ITERATOR        |                          |918K| 13M
|282(92)|0:00:01|:KV0000|:KV0000|
|20|           TABLE ACCESS INMEMORY FULL     | SALES                    |918K| 13M
|282(92)|0:00:01|:KV0000|:KV0000|
|21|    TABLE ACCESS FULL                     |SYS_TEMP_0FD9D6607_F6A13|  5 | 125 | 2
(0)|0:00:01|       |       |
-------------------------------------------------------------------------------------
-----------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
11 - access("ITEM_9"=INTERNAL_FUNCTION("C0"))
12 - access("ITEM_8"=INTERNAL_FUNCTION("C0"))
20 - inmemory(SYS_OP_KEY_VECTOR_FILTER("S"."PROD_ID",:KV0001) AND
SYS_OP_KEY_VECTOR_FILTER("S"."TIME_ID",:KV0000))
       filter(SYS_OP_KEY_VECTOR_FILTER("S"."PROD_ID",:KV0001) AND
SYS_OP_KEY_VECTOR_FILTER("S"."TIME_ID",:KV0000))
Note
-----
   - vector transformation used for this statement
Statistics
-----------------------------------------------------------
         26  recursive calls
         13  db block gets
        124  consistent gets
         67  physical reads
       2200  redo size
       1454  bytes sent via SQL*Net to client
        634  bytes received via SQL*Net from client
          3  SQL*Net roundtrips to/from client
```

```
          2  sorts (memory)
          0  sorts (disk)
         20  rows processed
```

9.2 インメモリー算術の最適化

インメモリー最適化算術では、SIMDハードウェアを使用して、最適化されたNUMBERフォーマットを使用して高速計算を行います。

9.2.1 インメモリー最適化算術について

インメモリー最適化されたNUMBER形式により、SIMDハードウェアを使用した高速計算が可能になります。

QUERY LOWで表を圧縮する場合、NUMBER列は、ハードウェアでのネイティブ計算が有効になる最適化された形式を使用してエンコードされます。SIMDベクター処理では、単純な集計、GROUP BY集計、算術演算を使用でき、大きいメリットがあります。パフォーマンスの向上は、集計が算術計算に費やす時間によって異なります。集計によっては、最大9倍までの効果があります。

問合せ処理エンジンのすべての行ソースで、インメモリー最適化数値形式がサポートされているわけではありません。したがって、IM列ストアには、Oracle Databaseの従来のNUMBERデータ型とインメモリー最適化数値型の両方を格納する必要があります。この二重記憶域は、領域のオーバーヘッドを場合によっては最大15%まで増加させます。

関連項目:

「IM列ストアの必須サイズの推定」

9.2.2 インメモリー最適化算術の有効化と無効化

初期化パラメータINMEMORY_OPTIMIZED_ARITHMETICをDISABLE (デフォルト)またはENABLEに設定することにより、機能を制御します。

ENABLEに設定すると、Oracle Databaseは、FOR QUERY LOW圧縮を使用する表のNUMBER列のインメモリー最適化コーディングを使用します。DISABLEに設定すると、データベースは、最適化されたエンコーディングを使用しません。

ENABLEからDISABLEに切り替えても、既存のIMCUの最適化された数値エンコーディングはすぐには削除されません。かわりに、IM列ストアがIMCUを再移入するので、新しいIMCUは最適化されたエンコーディングを使用しません。

インメモリー最適化算術を有効または無効にするには:

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。
2. ALTER SYSTEM文を使用して、INMEMORY_OPTIMIZED_ARITHMETICを指定します。

   次の例では、インメモリー最適化算術を有効にしています。

   ```
   ALTER SYSTEM SET INMEMORY_OPTIMIZED_ARITHMETIC = 'ENABLE' SCOPE=BOTH;
   ```

関連項目:

INMEMORY_OPTIMIZED_ARITHMETIC初期化パラメータについてさらに学習するには、Oracle Databaseリファレンスを参照

10 IM列ストアの再移入の最適化

IM列ストアでは、変更されたオブジェクトが定期的にリフレッシュされます。初期化パラメータおよびDBMS_INMEMORYパッケージを使用して、この動作を制御できます。

10.1 IM列ストアの再移入について

大幅な変更の後の列データの自動リフレッシュは、再移入と呼ばれます。

10.1.1 行の変更とトランザクション・ジャーナル

インメモリー圧縮単位(IMCU)は、内部表でDMLが発生したときに、その場でデータを変更しない読取り専用構造です。

各IMCUに関連付けられたスナップショット・メタデータ単位(SMU)は、トランザクション・ジャーナルの行の変更を追跡します。問合せで、データにアクセスし、変更された行を見つけた場合は、対応するROWIDをトランザクション・ジャーナルから取得し、その後、変更された行をバッファ・キャッシュから取得できます。

変更数が増加すると、SMUのサイズ、およびトランザクション・ジャーナルまたはデータベース・バッファ・キャッシュからフェッチする必要があるデータの量も増加します。ジャーナル・アクセスにより問合せパフォーマンスが低下しないようにするために、バックグラウンド・プロセスで、変更されたオブジェクトを再移入します。

10.1.2 自動再移入

IM列ストア内のオブジェクトに対してDMLが発生すると、データベースは自動的に再移入を実行します。

自動再移入は、次の形式をとります。

- しきい値ベース再移入

  この形式は、IMCUのトランザクション・ジャーナル内の失効エントリの割合によって異なります。

- トリクル再移入

  この形式は、しきい値に達していない場合でも失効列データを定期的にリフレッシュすることで、しきい値ベース再移入を補完します。

自動再移入中は、従来のアクセス・メカニズムを使用可能です。データには、常にバッファ・キャッシュまたはディスクからアクセスできます。また、IM列ストアは、ディスク上のデータとトランザクションの一貫性が常に保たれています。問合せでアクセスするデータの場所を問わず、データベースでは、常に、一貫性のある結果が返されます。

関連項目:

- 「トランザクション・ジャーナル」
- 「インメモリー・プロセスのアーキテクチャ」

10.1.3 外部表の手動再移入

外部表は自動再移入の対象ではありません。

IM列ストアは、外部表を内部表と異なる方法で管理します。外部表は読取り専用であるため、DMLでは更新されず、した

がってトランザクション・ジャーナルに依存しません。このため、データベースは外部表を自動的に再移入しません。ただし、DBMS_INMEMORY.REPOPULATEを使用することにより外部表を手動でリフレッシュできます。外部表のインメモリー・スキャンは、表がIM列ストアに完全に移入されている場合にのみサポートされています。

ノート:

インメモリーの外部表を問い合せるセッションでは、初期化パラメータ QUERY_REWRITE_INTEGRITY を stale_tolerated に設定する必要があります。

外部表が変更された場合、IM 列ストアからの結果は未定義であることに注意してください。パーティションが(値を削除または追加して)変更された場合も、結果は未定義です。これにより、IM ベースと IM 以外のスキャンの結果に違いが生じる可能性があります。DBMS_INMEMORY.REPOPULATE を実行して IM ストアをリフレッシュし、表データと再同期化できます。

関連項目:

- 「DBMS_INMEMORY.POPULATEを使用したインメモリー外部表の移入：例」
- QUERY_REWRITE_INTEGRITY初期化パラメータについてさらに学習するには、Oracle Databaseリファレンスを参照

10.2 データ・ロードはどのようにIM列ストアと連携するか

IM列ストアでは、従来型DML、ダイレクト・パス・ロードおよびパーティション交換ロードといったデータ・ロード・タイプに応じて、様々なメカニズムが使用されます。

10.2.1 従来型DMLはどのようにIM列ストアと連携するか

従来型DMLでは、一度に1つの行または行配列が処理され、最高水位標より下の行が挿入されます。IM列ストアが有効になっているかどうかに関係なく、データベースでは、バッファ・キャッシュを使用してDMLが処理されます。

IMCUは読取り専用です。文によってIMCU内の行が変更されると、IM列ストアにより、関連するSMU内のROWIDが記録されます。

列圧縮単位(CU)エントリは、その値が対応するジャーナル・エントリの値と異なる場合に失効になります。たとえば、トランザクションによって従業員の週給が1000から1200に変更されますが、IMCU内の実際の値は1000のままです。トランザクション・ジャーナルにより、失効行のROWIDおよびそのSCNが記録されます。

ノート:

トランザクション・ジャーナルでは、新しい値は記録されません。正確に述べると、対応する行が、特定の SCN の時点で失効として示されます。

10.2.1.1 失効しきい値

IMCU内の失効エントリの数が増えるほど、IMCUのスキャンの速度は低下します。

データベースでは、変更された行をIM列ストアからではなくバッファ・キャッシュまたはディスクからフェッチする必要があるため、パフォーマンスが低下します。このため、IMCU内の失効エントリの数が内部失効しきい値に達すると、Oracle Databaseにより、IMCUが再移入されます。

データベースでは、IMCUアクセスの頻度および失効行の数を考慮する経験則を使用して、そのしきい値が決定されます。頻繁にアクセスされる、または失効行の割合が高いIMCUでは、再移入がより頻繁に行われます。

関連項目:

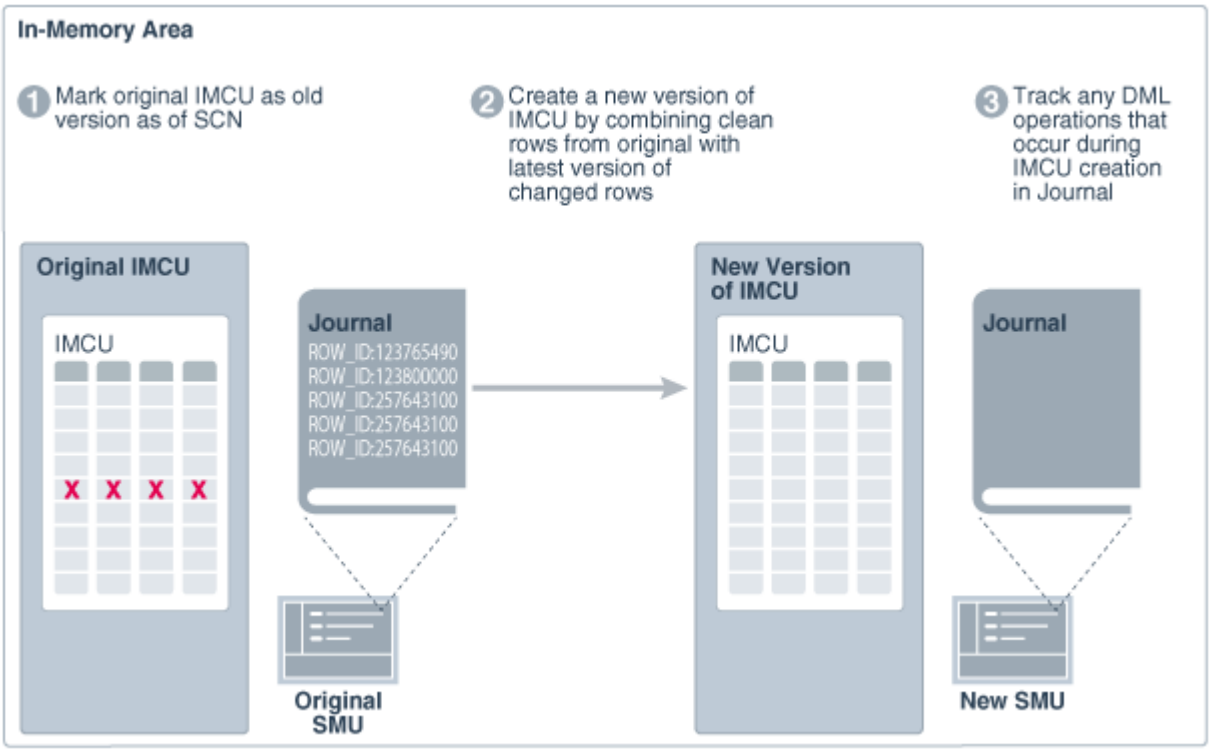
- 「インメモリー圧縮単位(IMCU)」
- データベース・バッファ・キャッシュについてさらに学習するには、Oracle Database概要を参照してください。

10.2.1.2 ダブル・バッファリング

ダブル・バッファリングでは、バックグラウンド・プロセスにより、元の行と最後に変更された行を組み合せることで、新しいIMCUバージョンが作成されます。

データベースでしきい値ベース再移入またはトリクル再移入のどちらかが開始されると、IM列ストアでダブル・バッファリングが使用されます。次の図で示すように、IM列ストアでは、同時に2つのバージョンのIMCUが保持され、元の失効IMCUは引き続き問合せでアクセス可能となります。

図10-1 ダブル・バッファリング



ダブル・バッファリングの基本ステップは、次のとおりです。

1. 元のSMUでは、データベースにより、特定のSCNの時点で既存のIMCUが元のバージョンとしてマークされます。
2. バックグラウンド・プロセスにより、変更された行の最新バージョンと元の行を組み合せることで、IMCUの新しいバージョンが作成されます。
3. 新しいSMUのジャーナルでは、データベースにより、IMCU作成中に起こるDML操作が追跡されます。

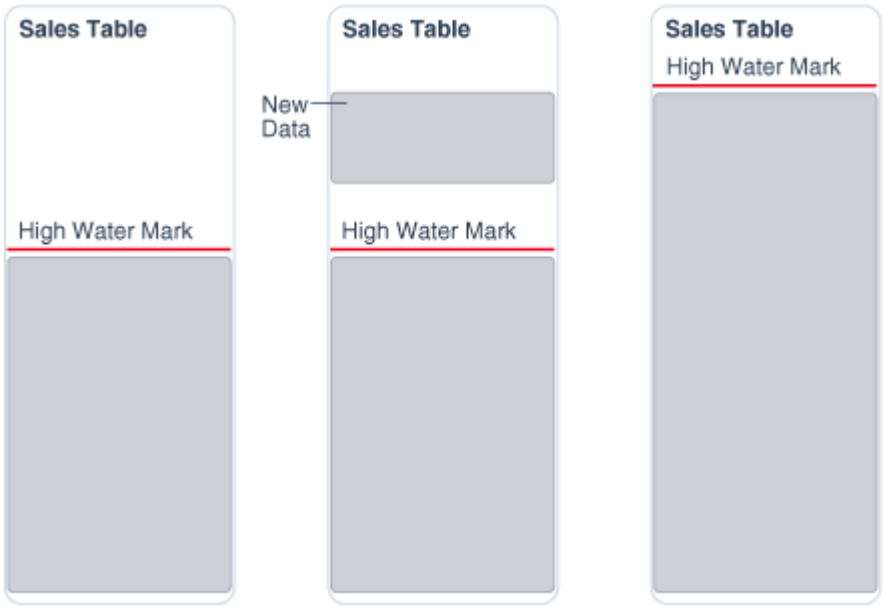
このようにして、元のIMCUはオンラインのままになります。データベースでは、IMCUの古いバージョンと新しいバージョンは、それらが役立つかぎり、またはIM列ストアに領域が必要になるまで、どちらも保持されます。

10.2.2 ダイレクト・パス・ロードはどのようにIM列ストアと連携するか

ダイレクト・パス・ロードは、`INSERT /*+APPEND*/`文、または`DIRECT=true`でのSQL*Loader操作です。

ダイレクト・パス・ロードでは、データベース・バッファ・キャッシュを迂回し、データベースにより、フォーマットされたデータ・ブロックがデータ・ファイルに直接書き込まれます。データベースにより、セグメント内の使用領域と未使用領域との境界である最高水位標より上のデータが付加されます。ダイレクト・パス・ロードは、すべてのデータを挿入するかまったく挿入しないという、オール・オア・ナッシングの操作です。

図10-2 ダイレクト・パス・ロードと最高水位標



セグメントがIM列ストアに移入される場合、ダイレクト・パス・ロードは次のように機能します。

1. `CREATE TABLE AS SELECT`または`INSERT /*+APPEND*/`文を使用してデータをロードします。現在のセッションのみが、DMLを認識しています。
2. 文をコミットします。
3. 最高水位標が、新しいデータを含むよう移動します。それにより、データが欠落しているIMCUが警告されます。`V$IM_SEGMENTS.BYTES_NOT_POPULATED`で、新しく挿入されたデータのサイズが示されるようになります。
4. IM列ストアで、次のアルゴリズムに基づいて再移入が管理されます。
    - 影響を受けたオブジェクトの`PRIORITY`が`NONE`以外の値に設定されている場合は、データベースにより、データが再移入されます。
    - 影響を受けたオブジェクトの`PRIORITY`が`NONE`に設定されている場合は、データベースにより、次回のオブジェクトの全体スキャンで再移入されます。

10.2.3 パーティション交換ロードはどのようにIM列ストアと連携するか

パーティション交換ロードは、表をパーティションと交換する技術です。交換ロードは、データではなくメタデータを変更するため、ほとんど一瞬です。

交換ロードを実行するには、次のステップに従います。

1. ソース表と呼ばれる、パーティション化されていない表を作成します。
2. ソース表に行をロードします。
3. ターゲット・パーティションと呼ばれる既存の表パーティションをその表と交換します。

交換後にターゲット・パーティションをIM列ストアに移入する場合は、ソース表を、交換前にIM列ストアに移入する必要があります。ターゲット・パーティションが移入されているかどうかに応じて、次のシナリオが可能となります。

- 交換前に、ターゲット・パーティションがIM列ストアに移入されていません。たとえば、そのパーティションが空だとします。

  交換後は、ソース表は、もうIM列ストアに移入されていません。ソースIMCUは、現在はターゲット・パーティションに関連付けられています。

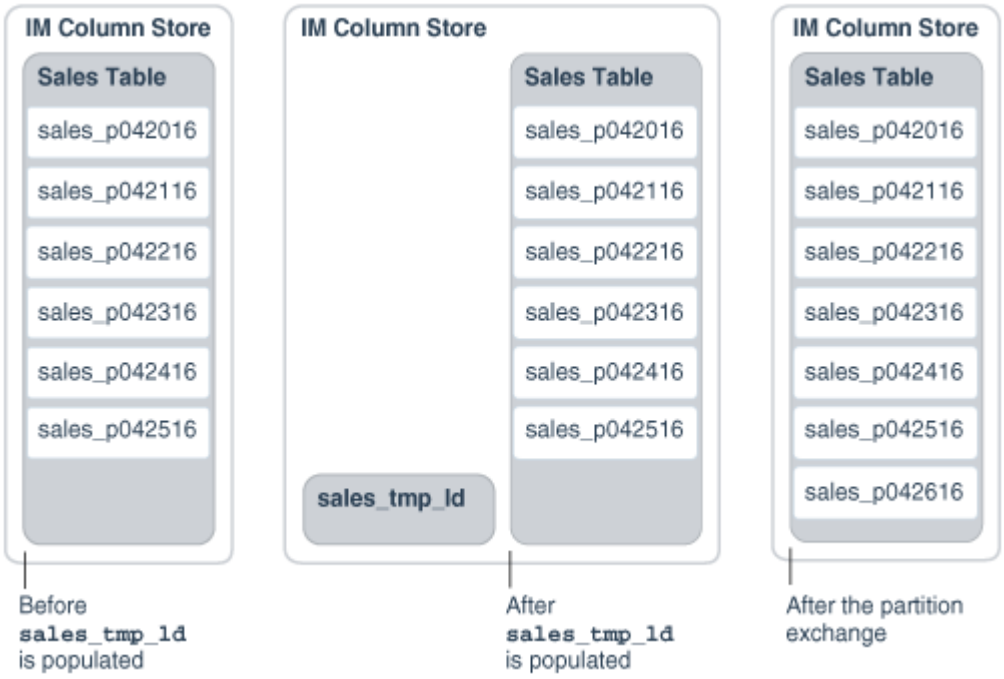
- 交換前に、ターゲット・パーティションがIM列ストアに移入されています。

  交換後は、ソース表は、IM列ストアに移入されたままとなります。

例10-1 INMEMORYパーティション交換ロード

この例では、パーティション化されているsales表のINMEMORY属性は表レベルで設定されています。この表内の空でないすべてのパーティションは、現在移入されています。sales_p042616パーティションは、現在は空になっています。目的は、空のパーティションsales_p042616に、テキスト・ファイルに含まれているデータを移入することです。次の図では、シナリオの前と後を示します。

図10-3 パーティション交換



この交換を実行するには、次の手順を実行します。

1. CREATE TABLE ... ORGANIZATION EXTERNAL文を使用して、外部表sales_tmp_extを作成します。

   この外部表は、データベース内には存在せず、アクセス・ドライバが提供されている任意の形式にすることができます。この表は、読取り専用です。

2. CREATE TABLE ... AS SELECT * FROM sales_tmp_extを使用して、sales_tmp_ldというパーティション化されていない表を作成します。

   sales_tmp_ld表は外部ではなく、これは、それにデータ・ファイル内の行が格納されることを意味します。

3. ALTER TABLE文を使用して、sales_tmp_ld内にINMEMORY属性を設定します。

   sales_tmp_ld表は、現在はINMEMORYとしてマークされていますが、まだIM列ストアには移入されていません。

4. 表の全体スキャンを強制的に実行することで、sales_tmp_ldをIM列ストアに移入します。

   たとえば、次の問合せは、全体スキャンを強制的に実行します。

   ```
   SELECT /*+ FULL(s) NO_PARALLEL(s) */ COUNT(*) FROM sales_tmp_ld s;
   ```

5. sales_p042616パーティションをsales_tmp_ld表と交換します。

   たとえば、次のようにsales表を変更します。

   ```
   ALTER TABLE sales EXCHANGE PARTITION sales_p042616 WITH TABLE sales_tmp_ld;
   ```

交換の完了後は、sales_p042616パーティションはIM列ストアに移入されており、sales_tmp_ldはもう移入されていません。

関連項目:

パーティション交換ロードについてさらに学習するには、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

10.3 データベースでIM列ストアが再移入される時期

データベースでは、内部アルゴリズムに従って、IM列ストアが自動的に再移入されます。手動で再移入を無効にして、その積極性に影響を与えることができます。

ノート:

この項では、自動再移入について説明します。DBMS_INMEMORY.REPOPULATE プロシージャを使用することで、再移入を手動で強制実行できます。

関連項目:

DBMS_INMEMORY.REPOPULATEプロシージャについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

10.3.1 しきい値ベース再移入とトリクル再移入

自動再移入には、しきい値ベース再移入とトリクル再移入という2つの形式があります。

自動再移入では、必ず、失効ジャーナル・エントリがチェックされ、ダブル・バッファリングが使用されます。ただし、再移入には、様々なトリガーがあります。

- しきい値ベース再移入

  データベースでは、トランザクション・ジャーナルで記録された変更の数が内部失効しきい値に達したときに、IMCUが移入されます。しきい値ベース再移入は、INMEMORY_MAX_POPULATE_SERVERS初期化パラメータが0以外の値に

設定されている場合は、自動的に起こります。

- トリクル再移入

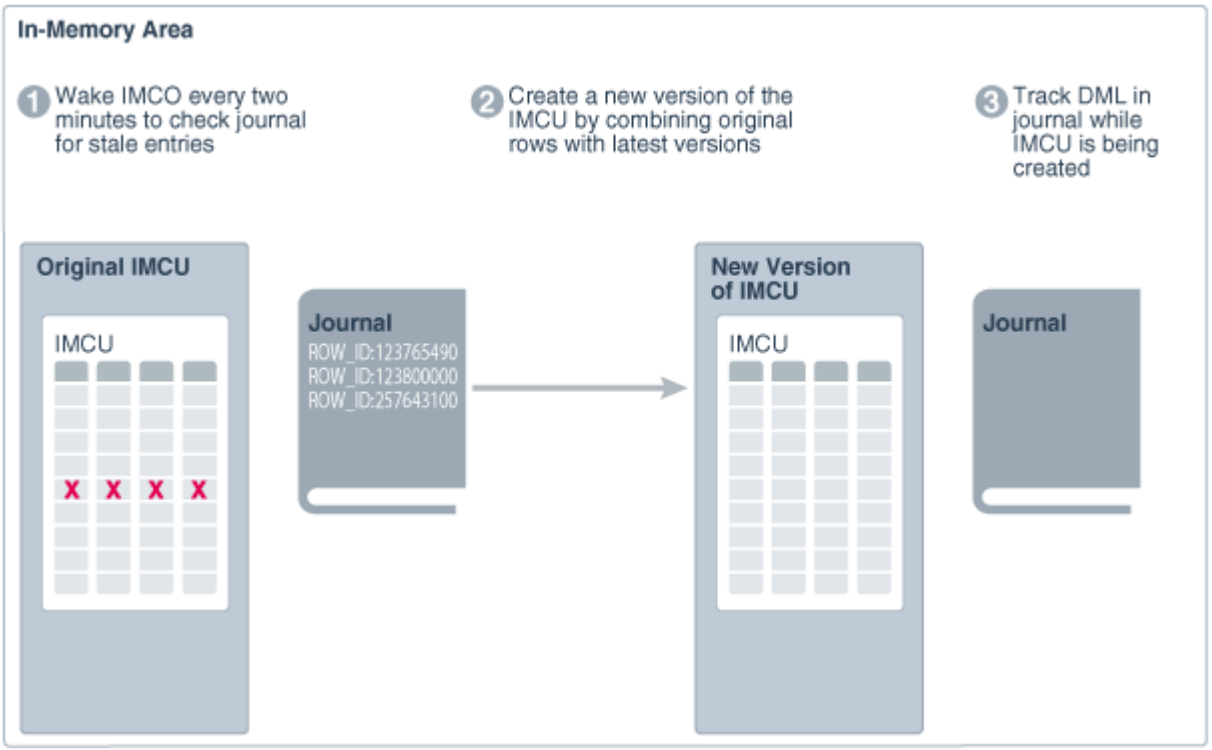
  IMCO (インメモリー・コーディネータ)バックグラウンド・プロセスでは、定期的に、失効行が存在するかどうかがチェックされてから、IMCUが再移入キューに追加されます。このメカニズムは、失効しきい値に達しているかどうかには依存しません。INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT初期化パラメータは、トリクル再移入に使用されるバックグラウンド・プロセスの数を制限します。このパラメータを0に設定すると、トリクル再移入が無効になります。

トリクル再移入は、Javaガベージ・コレクションに似ています。このメカニズムは、次のように機能します。

1. IMCOが起動します。
2. IMCOでは、IMCUに関連付けられているトランザクション・ジャーナルに失効エントリが存在するかどうかも含めて、移入タスクを実行する必要性の有無が判断されます。
3. IMCOによって失効エントリが見つかった場合は、領域管理ワーカー・プロセス(Wnnn)がトリガーされてIMCUの新しいバージョンが作成されます。

   IMCUの作成中に、データベースにより、変更された行のROWIDがトランザクション・ジャーナルに記録されます。
4. IMCOは2分間スリープしてから、ステップ1に戻ります。

図10-4 トリクル再移入



たとえば、データベースは、1日当たり8時間ビジー状態になるとします。ほとんどのSMUに、少数のトランザクション・ジャーナル・エントリが含まれています(失効しきい値未満)。データベースが静止状態のときに、IMCOが起動し、ジャーナルをチェックしてどのIMCUに失効エントリがあるかを判断してから、トリクル再移入を使用してそれらのIMCUをリフレッシュします。

関連項目:

- 「インメモリー・プロセスのアーキテクチャ」
- 「トランザクション・ジャーナル」

- インメモリー・バックグラウンド・プロセスについて学習するには、『Oracle Databaseリファレンス』を参照

10.3.2 再移入に影響する要素

再移入をトリガーするアルゴリズムは、内部的であり、いくつかの要素によって異なります。

再移入に影響する主な要素は、次のとおりです。

- DML変更の率

  変更された行の数が増えると、失効した列データの割合が増加します。トランザクション・ジャーナルが拡大し、問合せを満たすためにバッファ・キャッシュを使用する必要性が増加します。

- DML操作のタイプ

  一般的に、挿入は新しいデータ・ブロックに対して行われることが多いため、挿入には、削除や更新よりもパフォーマンス・オーバーヘッドはありません。

- データ・ブロック内での変更された行の位置

  同じデータベース・ブロックまたは表パーティション内でグループ化された変更は、表全体にわたり分散された変更よりも影響は少なくなります。すべてのIMCUのバージョニングは、少数のIMCUをバージョニングするよりも影響が大きくなります。

- `INMEMORY`オブジェクトに適用された圧縮レベル

  ダブル・バッファリングにより、表に適用された圧縮レベルが高いほど、再移入中に問合せおよびDMLのオーバーヘッドが大きくなります。たとえば、`MEMCOMPRESS FOR CAPACITY HIGH`では、`MEMCOMPRESS FOR DML`よりも大きなオーバーヘッドがもたらされます。

- アクティブなワーカー・プロセスの数

  ワーカー・プロセスの数が増加すると、並列で起こる処理が多くなります。それにより、再移入率が増加します。

関連項目:

- 「IM列ストアの圧縮方法」
- `INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT`初期化パラメータについて学習するには、『Oracle Databaseリファレンス』を参照

10.4 IM列ストアの再移入の制御

デフォルトでは、再移入は自動的に起こりますが、その積極性を制御することや、完全に無効にすることができます。

初期化パラメータ

次の初期化パラメータは、バックグラウンド・プロセスの挙動に影響します。

- `INMEMORY_MAX_POPULATE_SERVERS`

  このパラメータは、移入および再移入(しきい値ベースおよびトリクル)に使用可能なWnnnプロセスの最大数を制限します。デフォルト値は、`CPU_COUNT`の半分です。このパラメータは、スロットルの役割を果たし、これらのサーバー・プロセスで残りのデータベースがオーバーロードされないようにします。このパラメータを`0`に設定すると、移入および再移入が無効になります。

注意:

このパラメータの値の設定が高くなりすぎないように注意してください。コア数に近い値またはそれより大きい値に設定すると、残りのシステムの実行に使用可能な CPU がなくなる可能性があります。

- INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT

  このパラメータは、トリクル再移入を実行する、移入プロセスと再移入プロセスの合計パーセンテージを制限します。その趣旨は、2分間にトリクル再移入によって再移入されるIMCUの数を制限することです。

  このパラメータの値は、INMEMORY_MAX_POPULATE_SERVERS値のパーセンテージです。たとえば、INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENTが5パーセントで、INMEMORY_MAX_POPULATE_SERVERSが20の場合、IM列ストアでは、平均値である1コア(.05 * 20)がトリクル再移入のために使用されます。

  バックグラウンドCPUの増加という犠牲を払いスループットを向上させるには、このパラメータを5や10などの大きい値に設定します。INMEMORY_MAX_POPULATE_SERVERSプロセスの半分以上を他のタスクで使用可能なよう、50より大きい値は許可されていません。

  このパラメータを0に設定すると、トリクル移入が無効になります。

関連項目:

- INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENTについて学習するには、『Oracle Databaseリファレンス』を参照
- INMEMORY_MAX_POPULATE_SERVERSについて学習するには、『Oracle Databaseリファレンス』を参照

DBMS_INMEMORY.REPOPULATEプロシージャ

表、パーティションまたはサブパーティションを手動で再移入するには、DBMS_INMEMORY.REPOPULATEプロシージャを使用します。IM列ストアに現在移入されているオブジェクトのみが、再移入の対象となります。

次の値をforceパラメータで使用可能です。

- FALSE—データベースにより、変更された行を含むIMCUのみが再移入されます。これはデフォルトです。
- TRUE—データベースにより、セグメントが削除されてから再構築されます。データベースにより、統計が増分され、最初の移入に関連するその他すべてのタスクが実行されます。

たとえば、IMCU 1には行1から500,000までが含まれ、IMCU 2には行500,001から1,000,000までが含まれています。文により、行600,000が変更されます。forceがFALSEの場合は、データベースにより、IMCU 2のみが再移入されます。forceがTRUEの場合は、データベースにより、両方のIMCUが再移入されます。

さらにINMEMORY_VIRTUAL_COLUMNS初期化パラメータがENABLEに設定されており、アプリケーションによって新しい仮想列が作成されるとします。forceがFALSEの場合は、データベースにより、新しい列とともにIMCU 2のみが再移入されます。forceがTRUEの場合は、データベースにより、新しい列とともに両方のIMCUが再移入されます。

関連項目:

DBMS_INMEMORY.REPOPULATEについてさらに学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リ

ファレンス』を参照してください。

10.5 トリクル再移入の最適化: チュートリアル

このチュートリアルでは、トリクル再移入に使用可能なバックグラウンド・プロセスのパーセンテージを増加させます。

前提

このチュートリアルでは、次のことが前提となっています。

- IM列ストアが有効になっている。
- トリクル再移入を実行する領域管理ワーカー・プロセス(Wnnn)に、より多くのCPUを充てる必要があります。
- データベース・サーバーには、12個のCPUコアがあります。

再移入の積極性を向上させるには:

1. SQL*PlusまたはSQL Developerで、管理者権限を持つユーザーとしてデータベースにログインします。
2. 再移入に関連する初期化パラメータについて設定を表示します(出力例が含まれています)。

```
SHOW PARAMETER POPULATE_SERVERS
NAME                                         TYPE        VALUE
-------------------------------------        ----------- -----------
inmemory_max_populate_servers                integer     12
inmemory_trickle_repopulate_servers_percent integer     1
```

   上の出力では、12個のコアを移入タスクおよび再移入タスクに使用可能であることが示されています。INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENTは、INMEMORY_MAX_POPULATE_SERVERS値の1%です。移入タスクと再移入タスクに使用可能なサーバー・プロセスについては、IM列ストアでは、トリクル再移入に最大0.12個のCPUコア(.01 * 12)を使用できます。

3. トリクル再移入の最大値を、INMEMORY_MAX_POPULATE_SERVERS初期化パラメータ値の25%に増加させます。

   たとえば、次の文を使用します。

```
ALTER SYSTEM
  SET INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT=25
  SCOPE=BOTH;
```

   結果として、IM列ストアで、移入処理と再移入処理に使用可能な合計12個のうち、最大3個のCPUコア(.25 * 12)をトリクル再移入に使用するようになりました。

関連項目:

- INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENTについて学習するには、『Oracle Databaseリファレンス』を参照
- INMEMORY_MAX_POPULATE_SERVERSについて学習するには、『Oracle Databaseリファレンス』を参照

第IV部 高可用性とIM列ストア

この部では、インメモリー・ファスト・スタート(IMファスト・スタート)、Oracle Data GuardおよびOracle Real Application Clusters (Oracle RAC)などの高可用性機能とともにIM列ストアを使用する方法を説明します。

この項では、次の項目について説明します。

関連項目:

MAAホワイト・ペーパーOracle Database In-Memoryの高可用性のベスト・プラクティス

11 IM列ストアに対するIMファスト・スタートの管理

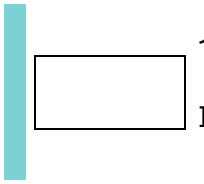
IM列ストアが有効になっている場合は、インメモリー・ファスト・スタート(IMファスト・スタート)により、列データをディスクに格納することでデータベースをより迅速に開くことができます。

この章のトピックは、次のとおりです:

11.1 IMファスト・スタートについて

IMファスト・スタートでは、IMCUをディスクに直接格納することで、IM列ストアへのデータベース・オブジェクトの移入が最適化されます。

データベースは、インスタンスの障害およびリカバリの後、または異なるOracle RACインスタンスへの複製中に、IM FastStart領域から読み取ることができます。



ノート:

IM ファスト・スタートは、読取り専用であるスタンバイ・データベースではサポートされていません。

この項では、次の項目について説明します。

11.1.1 IMファスト・スタートの目的

IM列ストアは、データベース・インスタンスを再起動すると必ず移入されます。これは、I/OおよびCPUに負担がかかる低速な操作となる可能性があります。

IMファスト・スタートが有効になっている場合、データベースでは、インスタンス再起動中の再移入を高速化するために、定期的に列データのコピーがディスクに保存されます。データベースを閉じてから再度開くと、データベースによって、FastStart領域から列データが読み取られ、それがIM列ストアに移入され、すべてのトランザクション一貫性が確保されます。

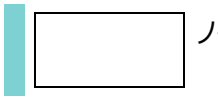
IMファスト・スタート表領域では、データベースが開かれて使用可能になっている間、断続的なI/Oが必要となります。データベースでCPUに負荷のかかるデータ圧縮およびフォーマットが行われなくなるため、データベースを再度開いたときにパフォーマンスが向上します。

11.1.2 IMファスト・スタートの仕組み

FastStart領域は、IMファスト・スタートにより`INMEMORY`オブジェクトのデータが格納および管理される、指定された表領域です。Oracle Databaseでは、FastStart表領域はDBAの介入なしで管理されます。

1つのFastStart領域、および指定された1つのFastStart表領域のみが、各PDBに対して許可されます。表領域は、指定されたIM FastStart表領域である間は、変更も削除もできません。Oracle RACデータベースでは、すべてのノードでFastStartデータが共有されます。

`DBMS_INMEMORY_ADMIN.FASTSTART_ENABLE`プロシージャを使用してFastStart表領域を有効にします。領域管理ワーカー・プロセス(Wnnn)で、`SYSDBinstance_name_LOBSEG$`という名前で空のSecureFiles LOBが作成されます。



ノート:

FastStart 領域を作成するには、IM FastStart 領域の有効化では不十分です。データの移入または再移入が必要となります。

関連項目:

- 「領域管理ワーカー・プロセス(Wnnn)」
- 「IM列ストアの再移入について」
- DBMS_INMEMORY_ADMIN.FASTSTART_ENABLEプロシージャについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照

11.1.2.1 データベースではどのようにFastStart領域が管理されるか

FastStart領域を有効にした後の最初の移入または再移入中に、データベースによってFastStart領域が作成されます。

データベースでは、次のように、自動的にFastStart領域が管理されます。

- オブジェクトの移入または再移入が行われるときは必ず、データベースによって、その列データがFastStart領域に書き込まれます。

  ノート:

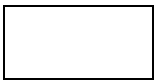
  

  データベースでは、FastStart 表領域も暗号化されている場合のみ、暗号化された表領域から FastStart 領域にセグメントが書き込まれます。

  領域管理ワーカー・プロセス(Wnnn)により、SYSDBinstance_name_LOBSEG$という名前のSecureFiles LOBにIMCU (IMEUやSMUではない)が書き込まれます。データベースにより、FastStartメタデータがSYSAUX表領域に書き込まれます。それらの表領域は、オンラインである必要があります。

  CUに対してどのくらいのDMLアクティビティが行われているかによって、FastStart領域内のCUとIM列ストア内のCUとの間でラグが存在することがあります。CUがホットなほど、データベースでそれをIM列ストアに移入してFastStart領域に書き込む頻度は少なくなります。データベースがクラッシュした場合は、IM列ストアに移入されたいくつかのCUがFastStart領域に存在しない場合があります。

  ノート:

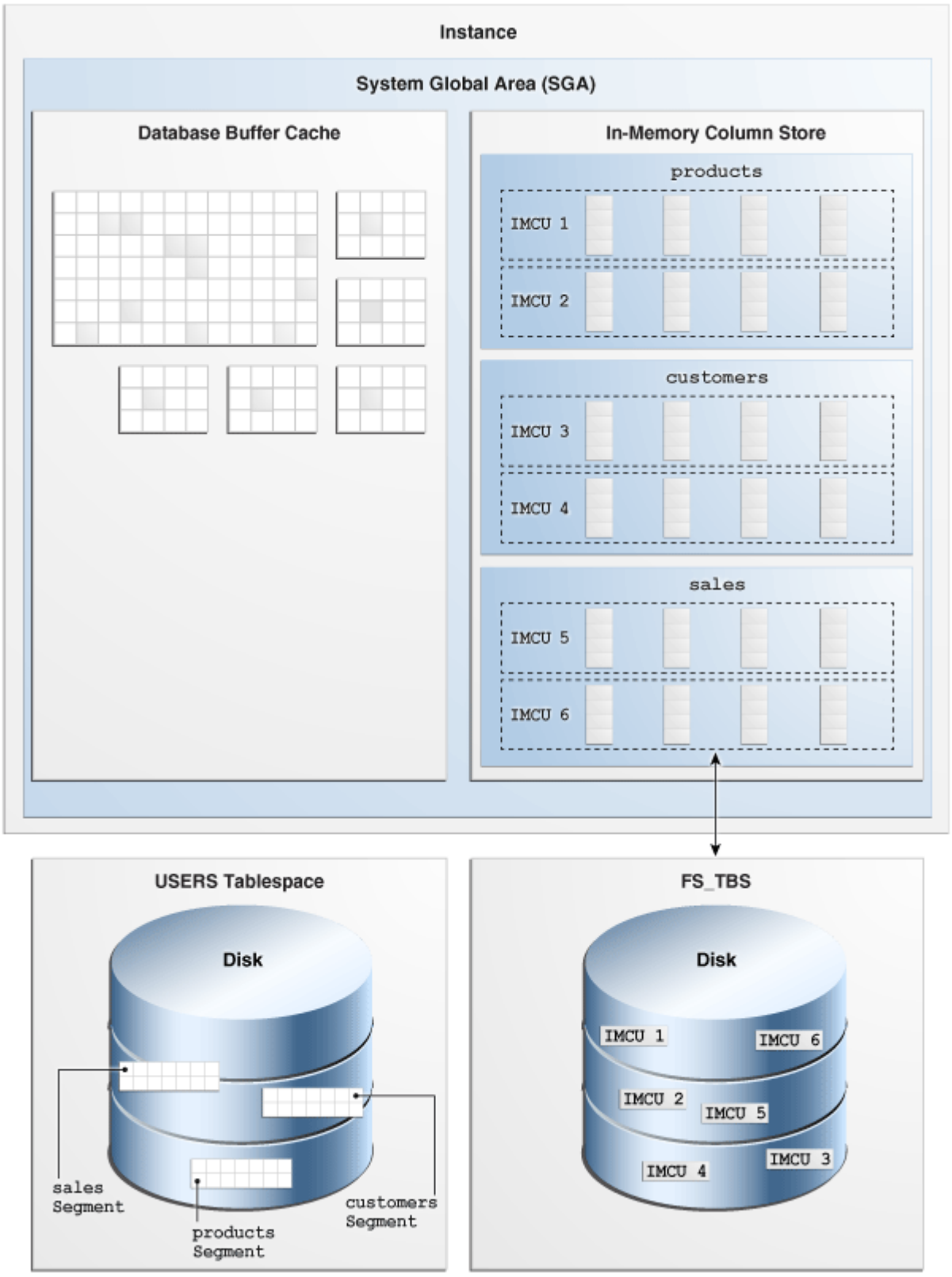
  FastStart 領域が一時的にアクセス不可になった場合、インメモリー移入は影響を受けません。

- セグメントに対してADOポリシーを定義した場合は、データベースで、そのポリシー内のルールに基づいて、FastStart領域内のセグメントが管理されます。たとえば、ADOにより、オブジェクトでポリシーに基づいてその属性がNO INMEMORYに変更されるよう指定した場合、IM列ストアで、そのデータがFastStart領域から削除されます。
- 移入されたオブジェクトの属性がNOINMEMORYに変更されると、データベースで、自動的にそのIMCUがFastStart領域から削除されます。
- FastStart表領域が領域不足になった場合は、データベースで、内部アルゴリズムを使用して最も古いセグメントが削除され、FastStart領域への書込みが続行されます。領域が残っていない場合は、データベースで、FastStart領域

への書込みが停止されます。

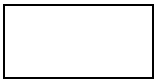
次の図では、IM列ストアに移入されたproducts、customersおよびsalesを示します。

図11-1 FastStart領域



FastStart領域が有効になっている場合は、データベースで、これらのセグメントのIMCUがfs_tbs内のFastStart領域にも書き込まれます。データベースを再度開くかインスタンスを再起動すると、データベースで、変更できるようIMCUを有効にしてトランザクションの一貫性を確保し、IMCUを再使用できます。FastStart領域が有効になっているかどうかに関係なく、データベースにより、データ・ブロックおよびセグメントがディスク上のusers表領域内に格納されます。

ノート:



データを FastStart 表領域に書き込むよう、IM 列ストアに手動で強制することはできません。

関連項目:

- 「IM列ストアに対するADOの有効化」
- 「Oracle RACでのFastStart領域」
- 「IM列ストアの再移入について」
- ADOについてさらに学習するには、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照

11.1.2.2 データベースでのFastStart領域からの読取り方

FastStart領域では、データベースを再度開いたときにどのデータをロードするかは定義されますが、いつロードするかは定義されません。移入は、優先順位設定によって制御されます。

データベースを再度開いたときに、標準のPRIORITYルールによって移入が決定されます。たとえば、データベースでは、要求に応じて、PRIORITY NONEが指定されているオブジェクトが移入されます。優先順位CRITICALが指定されているオブジェクトは、優先順位LOWが指定されているオブジェクトよりも、自動移入キュー内で高い位置にあります。

たとえば、単一インスタンスのデータベースで、sales、customersおよびproduct表が、PRIORITY NONEでIM列ストアに移入されます。どの再移入においても、データベースによって、これらの表のIMCUがFastStart領域に保存されます。そのインスタンスが予期せず終了したとします。データベースを再度開いたとき、IM列ストアは空になっています。問合せでsales、customersまたはproduct表をスキャンした場合は、データベースにより、この表のIMCUがFastStart領域からIM列ストアにロードされます。

ほとんどの場合、FastStart領域により、移入の速度が向上します。ただし、FastStart領域に格納されているCUがDMLアクティビティの内部しきい値に達した場合は、データベースにより、FastStart領域からではなくデータ・ファイルから行データが移入されます。

関連項目:

- 「インメモリー移入の優先順位付け」
- 「Oracle RACでのFastStart領域」
- INMEMORY句セマンティクスについては、『Oracle Database SQL言語リファレンス』を参照

11.2 IM列ストアに対するIMファスト・スタートの有効化

DBMS_INMEMORY_ADMIN.FASTSTART_ENABLEプロシージャを使用してFastStart領域のための表領域を指定します。

必要な場合は、FastStart領域のために作成されたLOBのロギング・モードを設定します。nologgingパラメータがTRUE(デフォルト)に設定されている場合は、データベースにより、NOLOGGINGオプションを指定してLOBが作成されます。nologgingがFALSEに設定されている場合は、データベースにより、LOGGINGオプションを指定してFastStart LOBが作成されます。

前提条件

FastStart領域を作成するには、次の前提条件を満たしている必要があります。

- FastStart領域として指定される表領域が存在する必要があります。
- この表領域には、IM列ストアのデータを格納するための十分な領域がある必要があり、FastStart領域として指定される前に他のデータが含まれていてはなりません。INMEMORY_SIZE設定の2倍の大きさでFastStart表領域を作成することをお薦めします。
- 管理者権限が必要です。

IM FastStart領域を作成するには:

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。
2. DBMS_INMEMORY_ADMIN.FASTSTART_ENABLEプロシージャを使用します。

例11-1 IM FastStart領域の指定

この例では、表領域を作成し、それをFastStart領域として指定します。

1. SQL*PlusまたはSQL Developerで、管理者権限を持つユーザーとしてデータベースにログインします。
2. fs_tbsという名前の表領域を作成します。

```
CREATE TABLESPACE fs_tbs
  DATAFILE 'fs_tbs.dbf' SIZE 500M REUSE
  AUTOEXTEND ON NEXT 500K MAXSIZE 1G;
```

3. IMファスト・スタートを有効にし、デフォルトのNOLOGGINGオプションをFastStart LOBに対して使用して、fs_tbs表領域をFastStart領域として指定します。

```
EXEC DBMS_INMEMORY_ADMIN.FASTSTART_ENABLE('fs_tbs');
```

4. FastStart領域のステータスおよびサイズを問い合せます。

```
COL TABLESPACE_NAME FORMAT a15
SELECT TABLESPACE_NAME, STATUS,
       ( (ALLOCATED_SIZE/1024) / 1024 ) AS ALLOC_MB,
       ( (USED_SIZE/1024) / 1024 ) AS USED_MB
FROM   V$INMEMORY_FASTSTART_AREA;
TABLESPACE_NAME STATUS                  ALLOC_MB    USED_MB
--------------- -------------------- ---------- ----------
FS_TBS          ENABLE                        500      .0625
```

この段階では、FastStart領域にユーザー・データはありません。

5. FastStart LOBのロギング・モードを問い合せます。

```
COL SEGMENT_NAME FORMAT a20
SELECT SEGMENT_NAME, LOGGING
FROM   DBA_LOBS
WHERE  TABLESPACE_NAME = 'FS_TBS';
SEGMENT_NAME         LOGGING
-------------------- -------
SYSDBIMFS_LOBSEG$    NO
```

6. 現在移入されているオブジェクトの再移入をIM列ストアに強制します。

次の問合せでは、sales、productsおよびcustomers表の再移入が強制的に実行されます。

```
SELECT /*+ FULL(s) NO_PARALLEL(s) */ COUNT(*) FROM sh.sales s;
```

```
SELECT /*+ FULL(p) NO_PARALLEL(p) */ COUNT(*) FROM sh.products p;
SELECT /*+ FULL(c) NO_PARALLEL(c) */ COUNT(*) FROM sh.customers c;
```

7. FastStart領域のサイズを問い合せます。

```
COL TABLESPACE_NAME FORMAT a15
SELECT TABLESPACE_NAME, STATUS,
       ( (ALLOCATED_SIZE/1024) / 1024 ) AS ALLOC_MB,
       ( (USED_SIZE/1024) / 1024 ) AS USED_MB
FROM   V$INMEMORY_FASTSTART_AREA;
TABLESPACE_NAME STATUS                 ALLOC_MB    USED_MB
--------------- -------------------- ---------- ----------
FS_TBS          ENABLE                      500       2.25
```

ここでは、同じ問合せで、FastStart領域の2.25 MBが使用されていることが示されています。

関連項目:

DBMS_INMEMORY_ADMINパッケージについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

11.3 現在のIMファスト・スタート表領域の名前の取得

V$INMEMORY_FASTSTART_AREAビューを問い合せることで、現在FastStart領域として指定されている表領域の名前を取得します。

有効になっているFastStart表領域がない場合は、STATUS列にNOT ENABLEDと示されます。そうでない場合、この列には、表領域名が示されます。

前提条件

FastStart表領域の名前を取得するには、管理者権限が必要です。

FastStart表領域の名前を取得するには:

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。
2. V$INMEMORY_FASTSTART_AREAビューを問い合せます。

例11-2 現在のIMファスト・スタート表領域の名前の取得

この例では、FastStart表領域の名前およびステータスを問い合せます(出力例が含まれています)。

```
COL TABLESPACE_NAME FORMAT a20
SELECT TABLESPACE_NAME, STATUS
FROM   V$INMEMORY_FASTSTART_AREA;
TABLESPACE_NAME      STATUS
-------------------- ----------
FS_TBS               ENABLE
```

関連項目:

V$INMEMORY_FASTSTART_AREAビューについて学習するには、『Oracle Databaseリファレンス』を参照してください。

11.4 別の表領域へのFastStart領域の移行

DBMS_INMEMORY_ADMINパッケージ内のFASTSTART_MIGRATE_STORAGEプロシージャを実行することにより、別の表領域にFastStart領域を移行できます。

非CDBまたはPDBでは、一度に1つの表領域のみをFastStart領域として指定できます。

前提条件

FastStart領域を移行するには、次の前提条件を満たしている必要があります。

- 新しいFastStart領域として指定される表領域が存在する必要があります。
- この表領域には、IM列ストアのデータを格納するための十分な領域がある必要があり、FastStart領域として指定される前に他のデータが含まれていてはなりません。
- 管理者権限が必要です。

IM FastStart領域を移行するには:

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。
2. DBMS_INMEMORY_ADMIN.FASTSTART_MIGRATE_STORAGEプロシージャを実行します。

例11-3 別の表領域へのFastStart領域の移行

この例では、IM FastStart領域をnew_fs_tbs表領域に移行します。

1. SQL*PlusまたはSQL Developerで、管理者権限を持つユーザーとしてデータベースにログインします。
2. 現在のFastStart表領域の名前を問い合せます。

```
COL TABLESPACE_NAME FORMAT a15
SELECT TABLESPACE_NAME, STATUS
FROM   V$INMEMORY_FASTSTART_AREA;
TABLESPACE_NAME STATUS
--------------- -----------
FS_TBS          ENABLE
```

3. new_fs_tbsという名前の表領域を作成します。

```
CREATE TABLESPACE new_fs_tbs
  DATAFILE 'new_fs_tbs.dbf' SIZE 500M REUSE
  AUTOEXTEND ON NEXT 500K MAXSIZE 1G;
```

4. FastStart領域を新しい表領域に移行します。

```
EXEC DBMS_INMEMORY_ADMIN.FASTSTART_MIGRATE_STORAGE('new_fs_tbs');
```

5. 現在のFastStart表領域の名前を問い合せます。

```
TABLESPACE_NAME      STATUS
-------------------- --------------------
NEW_FS_TBS           ENABLE
```

関連項目:

FASTSTART_MIGRATE_STORAGEプロシージャについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

11.5 IM列ストアに対するIMファスト・スタートの無効化

IMファスト・スタートを無効にした場合は、データベースでFastStart領域が保持されなくなります。データベースを再度開いたとき、データベースでIM列ストアの移入にIMファスト・スタートは使用されません。

前提条件

FastStart領域を無効にするには、次の条件を満たしている必要があります。

- FastStart領域が有効になっている必要があります。
- 管理者権限が必要です。

FastStart表領域を無効にするには:

1. SQL*PlusまたはSQL Developerで、必要な権限を持つユーザーとしてデータベースにログインします。
2. V$INMEMORY_FASTSTART_AREAを問い合せて、IM FastStart領域が有効になっていることを確認します。
3. DBMS_INMEMORY_ADMIN.FASTSTART_DISABLEプロシージャを実行します。
4. 必要な場合は、FastStart表領域を削除します。

例11-4 IMファスト・スタートの無効化

この例では、IM FastStart領域を無効にしてから、fs_tbs表領域を削除します。

1. SQL*PlusまたはSQL Developerで、管理者権限を持つユーザーとしてデータベースにログインします。
2. FastStart領域のステータスを問い合せます。

```
COL TABLESPACE_NAME FORMAT a15
SELECT TABLESPACE_NAME, STATUS
FROM   V$INMEMORY_FASTSTART_AREA;
TABLESPACE_NAME STATUS
--------------- -----------
FS_TBS          ENABLE
```

3. FastStart領域を無効にします。

```
EXEC DBMS_INMEMORY_ADMIN.FASTSTART_DISABLE;
```

4. FastStart領域のステータスを問い合せます。

```
SELECT TABLESPACE_NAME, STATUS
FROM   V$INMEMORY_FASTSTART_AREA;
TABLESPACE_NAME      STATUS
-------------------- --------------------
INVALID_TABLESPACE   DISABLE
```

IMファスト・スタートが有効になっている場合、TABLESPACE_NAMEの値はINVALID_TABLESPACEであり、STATUSの値はDISABLEです。

5. 前のFastStart表領域を削除します。

```
DROP TABLESPACE fs_tbs INCLUDING CONTENTS AND DATAFILES;
```

関連項目:

FASTSTART_DISABLEプロシージャについて学習するには、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

12 Oracle RACでのIM列ストアのデプロイ

この章では、Oracle Real Application Clusters (Oracle RAC)環境でIM列ストアを使用可能にする方法、およびオブジェクトを移入のために構成する方法について説明します。

この項では、次の項目について説明します。

12.1 Database In-MemoryおよびOracle RACの概要

すべてのOracle RACノードには、独自のインメモリー(IM)列ストアがあります。デフォルトでは、移入されたオブジェクトは、クラスタ内のすべてのIM列ストアにわたり分散されます。

すべてのOracle RACノードでIM列ストアを同じサイズにすることをお薦めしますOracle RACノードがIM列ストアを必要としない場合、INMEMORY_SIZEパラメータを0に設定します。

完全に異なるオブジェクトを各ノードに移入させたり、より大きなオブジェクトをクラスタ内のすべてのIM列格納間で分散させることが可能です。Oracle Engineered Systemsでは、同じオブジェクトを各ノードのIM列ストアに表示させることも可能です。クラスタ内のIM列ストア間のオブジェクトの分散は、INMEMORY属性への追加の副句(DISTRIBUTEおよびDUPLICATE)により制御されます。

Oracle RAC環境では、指定されたINMEMORY属性のみを含むオブジェクトが、クラスタ内のIM列ストア間で自動的に分散されます。DISTRIBUTE句を使用して、クラスタ間でのオブジェクトの分散方法を指定できます。デフォルトで、使用されるパーティション化のタイプ(ある場合)によりオブジェクトの分散方法が決定されます。オブジェクトがパーティション化されないと、ROWID範囲に分散されます。あるいは、DISTRIBUTE句を指定して、デフォルトの動作をオーバーライドできます。

Oracle Engineered Systemでは、クラスタ内のIM列ストア間で移入されたオブジェクトを複製またはミラー化できます。この技法により最高レベルの冗長性が提供されます。DUPLICATE句は、オブジェクトの複製方法を制御します。DUPLICATEのみを指定すると、データのミラー化されたコピーが1つ、クラスタ内のIM列ストア間で分散されます。各IM列ストア内のすべてのオブジェクトを複製するには、DUPLICATE ALLを指定します。

ノート:

非エンジニアド・システムで Oracle RAC にデプロイする場合、DUPLICATE 句は NO DUPLICATE として扱われます。

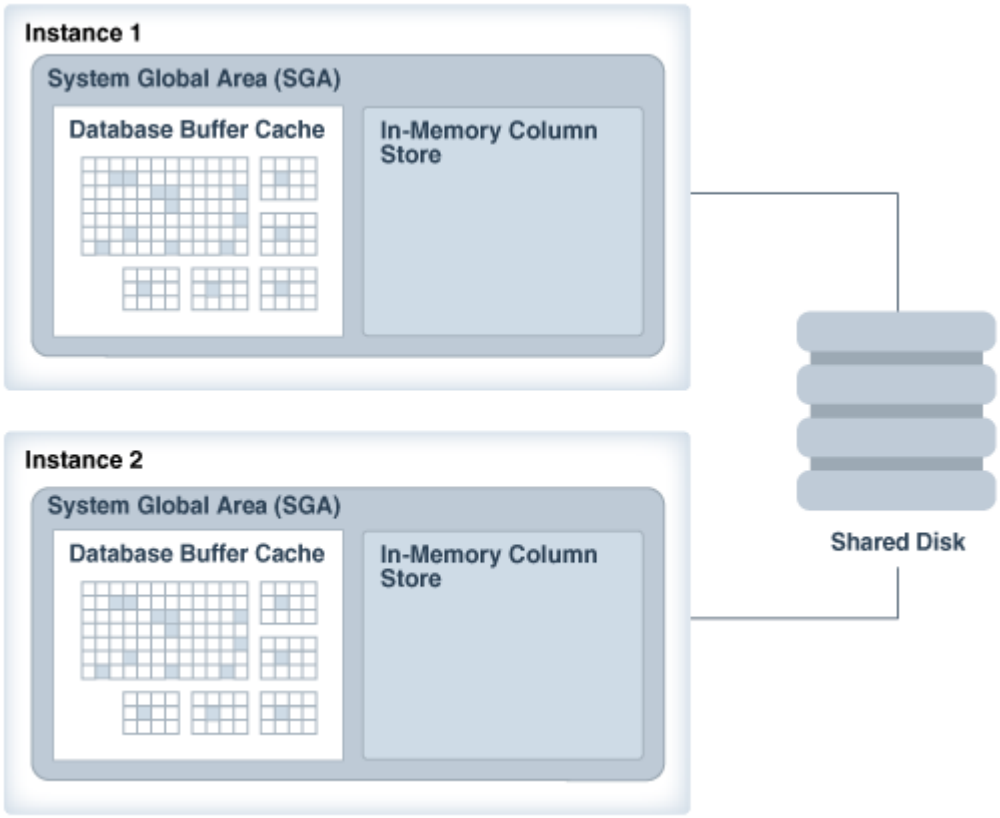
12.1.1 複数のIM列ストア

Oracle RACでは、各データベース・インスタンスには固有のIM列ストアがあります。

概念上、Oracle RAC環境内のIM列ストアでは、シェアード・ナッシング・アーキテクチャが使用されます。各データベース・インスタンスで、別個にIM列ストアをサイズ設定および管理します。データベース・インスタンスで、キャッシュ・フュージョンを使用してIMCUがあちこちへ転送されることはありません。

図12-1 Oracle RACデータベース内のIM列ストア

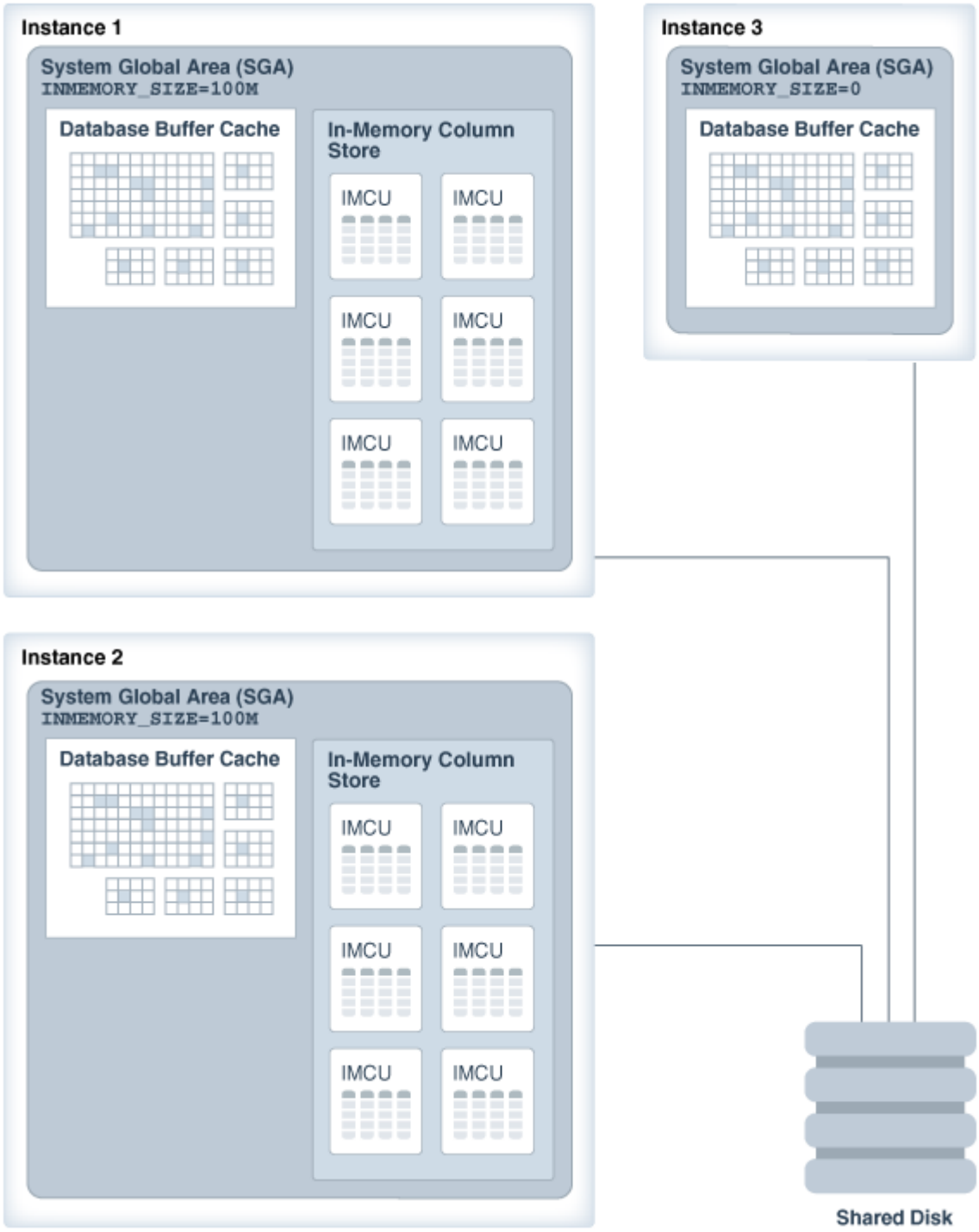
この図は、ノード2つのOracle RACクラスタを示しています。各インスタンスには、別々に構成されたIM列ストアがあります。



すべてのOracle RACノードでIM列ストアのサイズを等しく設定することをお薦めします。たとえば、どのIM列ストアにも100 GBのメモリーを割り当てることができます。IM列ストアを必要としないノードの場合は、このノードに対して`INMEMORY_SIZE`初期化パラメータを0に設定します。

図12-2 2つのIM列ストアがある、ノード3つのOracle RACデータベース

この例では、インスタンス1およびインスタンス2にIM列ストアがあります。インスタンス3ではIM列ストアは必要ないため、このノードに対する`INMEMORY_SIZE`初期化パラメータは0に設定されています。



関連項目:

- 「データベースに対するIM列ストアの有効化」
- INMEMORY_SIZE初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照
- Oracle RACの概要は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照

12.1.2 Oracle RACでの列データの分散および複製

INMEMORYが指定されると、DISTRIBUTEおよびDUPLICATEキーワードはオブジェクトの分散を制御します。

Oracle RACには、複数の分散オプションが用意されています。ノードごとに異なるオブジェクトを移入することや、大きなオブジェクトをOracle RACクラスタ内のすべてのIM列ストアにわたり分散することができます。同じオブジェクトを各ノード(Oracle

Engineered Systems上のみ)のIM列ストアに移入することもできます。

ノート:

表が現在 IM 列ストアに移入されており、PRIORITY 以外の表の INMEMORY 属性を変更した場合は、データベースでその表が IM 列ストアから除去されます。再移入の動作は、PRIORITY 設定によって異なります。

この項では、次の項目について説明します。

12.1.2.1 Oracle RACでの列データの分散

INMEMORYのDISTRIBUTE句は、IM列ストア内の表データを複数のOracle RACインスタンスにわたりどのように分散するかを制御します。

デフォルト・オプションAUTOが設定されている場合は、Oracle RACインスタンスにより、データが自動的に分散されます。セグメントの移入中に、領域管理スレーブ・プロセス(Wnnn)により、各インスタンスに同量のデータの配置が試みられます。分散は、アクセス・パターンおよびオブジェクト・サイズによって異なります。もう1つの方法として、データベースでパーティション、サブパーティションまたはROWID範囲をインスタンス間で分散するやり方を手動で指定することもできます。

パフォーマンスのためには、同量のデータ分散が重要となります。目的は、パラレル問合せプロセスで等しいデータ・セット・サイズを処理することで、それらすべてが最小限の時間で終了するようにすることです。データ分散が偏ると、実行時間の長いプロセスによって問合せの完了が遅くなります。

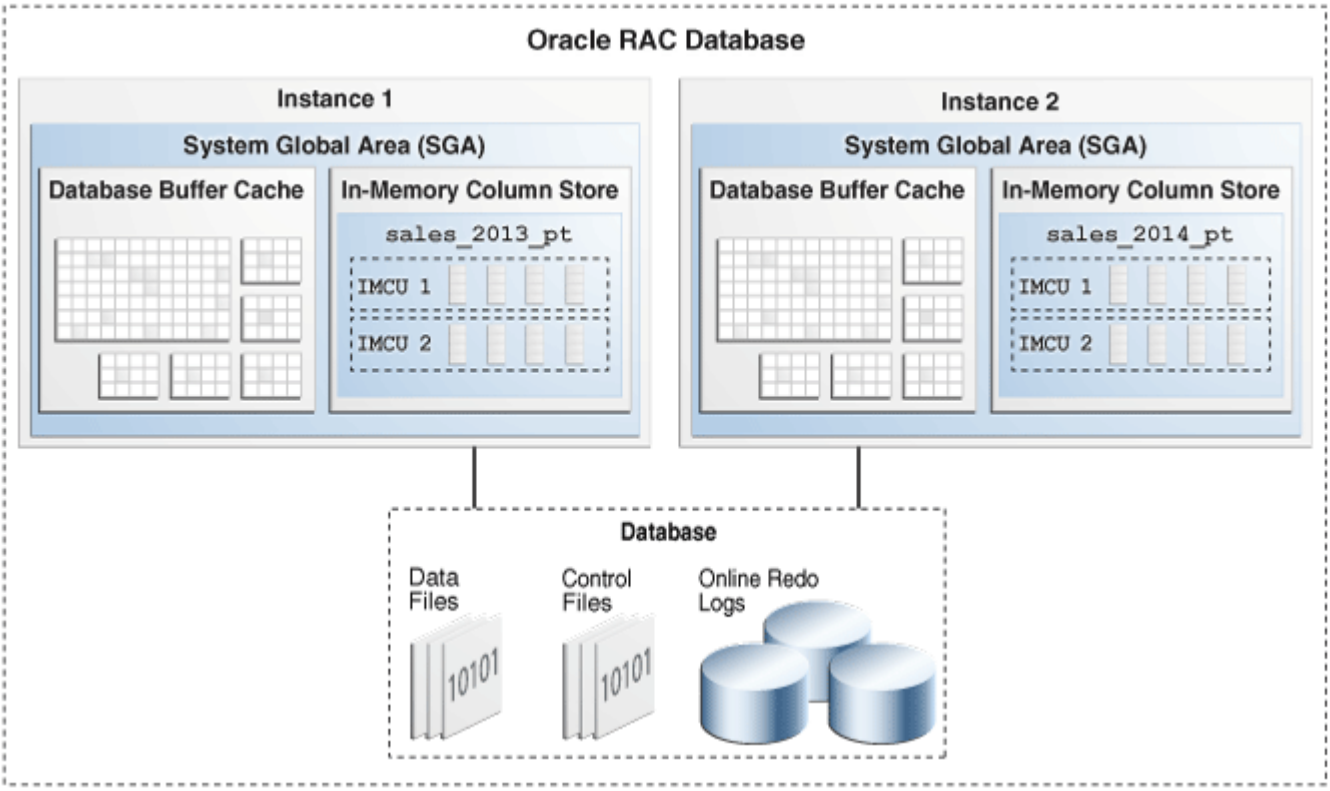
Oracle RACインスタンスに障害が発生すると、障害が発生したインスタンス上のIMCUが利用不可になります。その結果、アクセス不可のIMCUに格納されているデータを必要とする問合せでは、データベース・バッファ・キャッシュ、フラッシュ・ストレージ、ディスク、または他のIM列ストア内のミラー済IMCUなど、別の場所からそれを読み取ることが必要になります。

DBA_TABLES.INMEMORY_DISTRIBUTE列は、IMCUをどのように分散するかを示します。AUTOオプションが設定されている場合、この列値はAUTO-DISTRIBUTEとなります。

例12-1 デフォルトの分散

この例では、パーティションsales_2013_ptおよびsales_2014_ptのみを含むsales表を分散するデータベースを示します。データベースでは、自動的にsales_2013_ptパーティションがインスタンス1に、sales_2014_ptがインスタンス2に配置されます。

図12-3 Oracle RACでの自動的なインメモリー分散



この項では、次の項目について説明します。

関連項目:

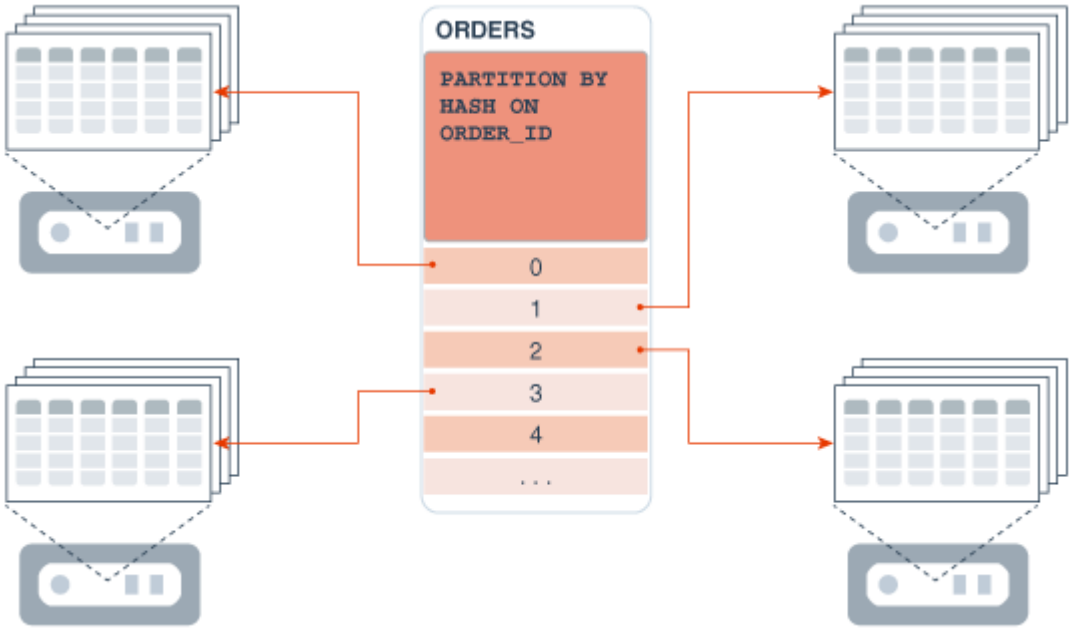
- 「領域管理ワーカー・プロセス(Wnnn)」
- Oracle RACでのIM列ストアの管理方法を学習するには、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照
- DISTRIBUTE句についてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照
- Wnnnバックグラウンド・プロセスについて学習するには、『Oracle Databaseリファレンス』を参照

12.1.2.1.1 パーティションによる分散

DISTRIBUTE BY PARTITION句を使用して、パーティション内のデータを様々なOracle RACインスタンスに分散できます。

この手法は、ハッシュ・パーティションに適しています。たとえば、orders表内のパーティションを等しく分散するには、order_id列でハッシュによって分配します。次の図に示すように、Oracle Databaseでは、order_id列でハッシュすることで4つのインスタンスにわたりパーティションが分散されます。

図12-4 ハッシュによるパーティションの分散



この手法は、複数パーティションが均一にアクセスされる場合のその他のパーティション化スキームに適しています。DISTRIBUTE BY PARTITION句では、パーティション・ワイズ結合もサポートされています。

ノート:

パーティション化戦略によって大規模なデータ・スキューが生じる場合、つまり、あるパーティションが他のパーティションよりも非常に大きい場合は、DISTRIBUTE BY ROWID RANGE を手動で指定することでデフォルトの分散(BY PARTITION)をオーバーライドすることをお薦めします。

関連項目:

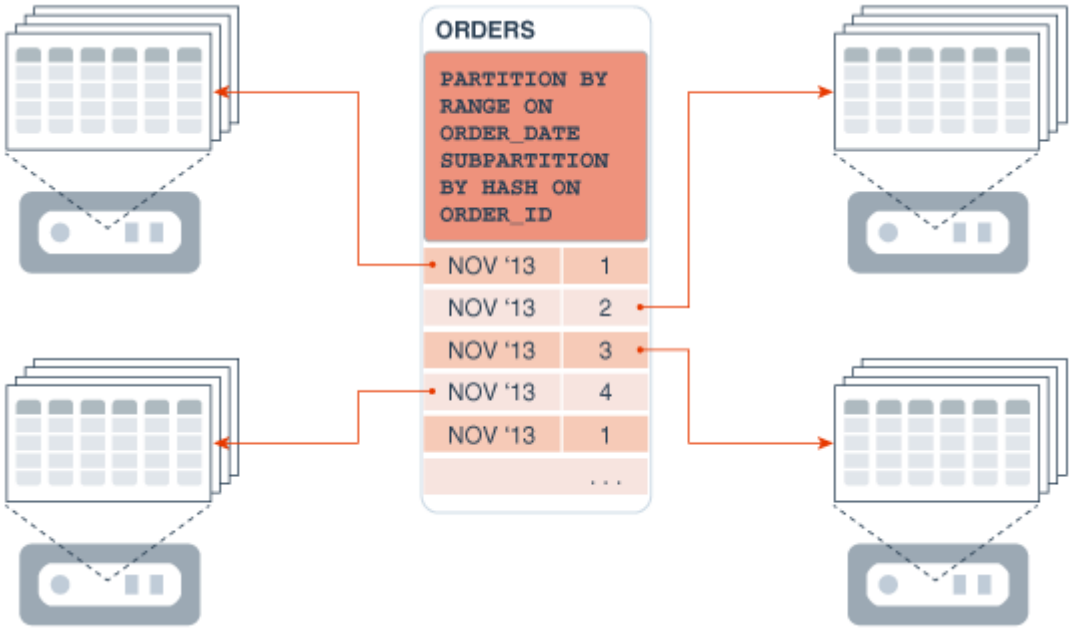
- DISTRIBUTE BY PARTITION副句についてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照
- パーティション表の概要は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照

12.1.2.1.2 サブパーティションによる分散

コンポジット・パーティション化スキームを使用する表では、DISTRIBUTE BY SUBPARTITION句を使用して、サブパーティション内のデータを様々なインスタンスに分散できます。

この手法は、ハッシュ・サブパーティションに適しています。たとえば、orders表内のパーティションを等しく分散するには、order_date列で範囲によって、またはorder_id列でハッシュによって分配します。

図12-5 範囲によるパーティションの分散とハッシュによるサブパーティションの分散



この手法は、複数サブパーティションが均一にアクセスされる場合のその他のパーティション化スキームに適しています。

`DISTRIBUTE BY PARTITION ... SUBPARTITION`句では、パーティション・ワイズ結合もサポートされています。

関連項目:

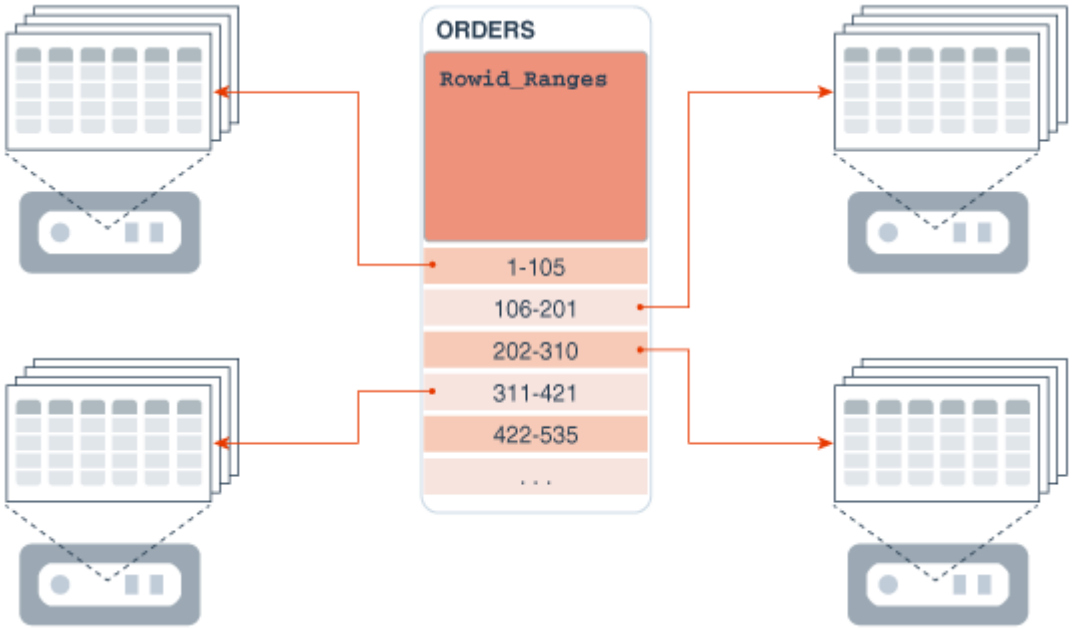
- コンポジット・パーティション化についてさらに学習するには、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照
- `DISTRIBUTE BY PARTITION ... SUBPARTITION`副句についてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照

12.1.2.1.3 ROWID範囲による分散

`DISTRIBUTE BY ROWID RANGE`句を使用して、特定のROWID範囲内のデータを様々なOracle RACインスタンスに分散できます。

この手法では、最初のROWIDに対する均一ハッシュによってIMCUが分散されます。たとえば、Oracle Databaseで`orders`表内の行1から105までをあるデータベース・インスタンスに分散し、行106から121までを別のインスタンスに分散するといったことができます。

図12-6 ROWID範囲による分散



ROWIDの分散手法は、パーティション化されていない表に最も役立ちます。ただし、パーティション化戦略によって大規模なデータ・スキューが生じる場合、たとえば、あるパーティションが他のパーティションよりも非常に大きい場合は、DISTRIBUTE BY ROWID RANGEを手動で指定することでデフォルトの分散(BY PARTITION)をオーバーライドすることをお薦めします。

関連項目:

DISTRIBUTE BY ROWID RANGE副句についてさらに学習するには、Oracle Database SQL言語リファレンスを参照してください。

12.1.2.2 Oracle RACでの列データの複製

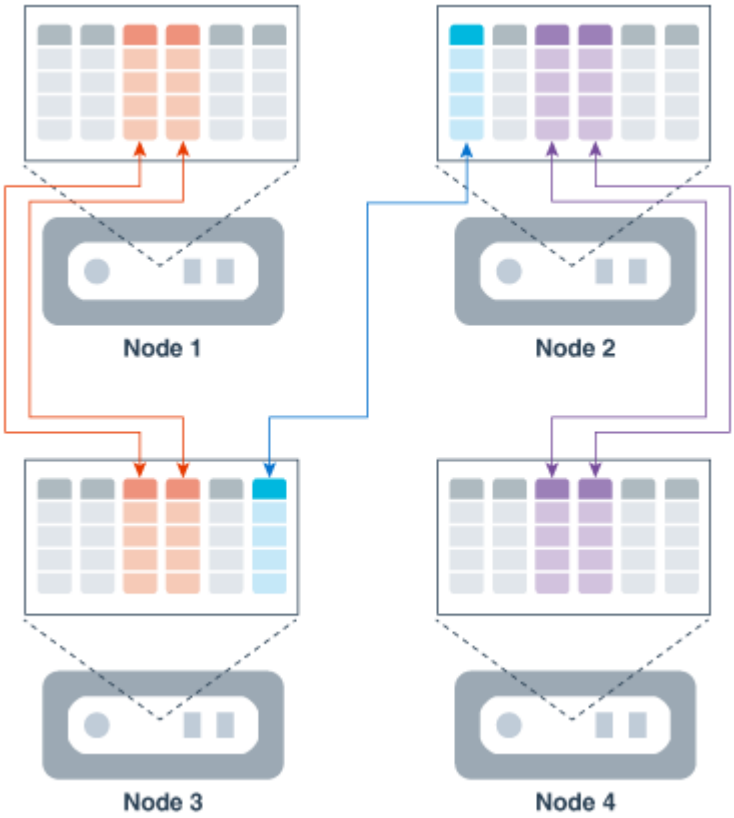
DUPLICATE句は、どのようにOracle RACデータベースによって複数のOracle RACインスタンスにわたり列データを複製するかを制御します。

ノート:

DUPLICATE 句は、Oracle Engineered System 上の Oracle RAC でのみ使用可能です。Oracle RAC が Oracle Engineered System で稼働しないとき、DUPLICATE 句は NO DUPLICATE と機能的に同等になります。

IM列ストアのフォルト・トレランスを提供するため、IMCUのミラー化を選択できます。IMCUミラー化では、同じIMCUが複数のIM列ストアに存在します。この手法は、ストレージのミラー化に似ています。

図12-7 Oracle RACでのIMCUの複製



DUPLICATE句を使用して表領域またはオブジェクト(表、パーティションまたはサブパーティション)レベルでデータをミラー化すると、次のような利点があります。

- あるノードで障害が発生した場合に、ミラー化した列データに異なるノードからアクセスできるため、フォルト・トレランスが提供されます。
- 問合せでローカルでデータにアクセスでき、それによってバッファ・キャッシュまたはディスクへのアクセスが行われなくなるため、パフォーマンスが向上します。

  たとえば、スター・クエリーでは、ディメンション表がDUPLICATE ALLを使用するのに対して、ファクト表はパーティション化できます。このシナリオでは、すべての結合は、完全にローカル・ノードで行われます。
- オブジェクトのサブセットを複製できるため、管理性が向上します。

  たとえば、今年のパーティションを複製し、同じ表の他のパーティションを複製しないでおくことができます。

IMCUのミラー化の短所は、オブジェクトがn回複製されると、そのメモリー要件がn倍増加することです。たとえば、500 MBの表が4つのインスタンスに分散されている場合は、合計で2000 MBのメモリーを使用します。

この項では、次の項目について説明します。

関連項目:

DUPLICATE句についてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照してください。

12.1.2.2.1 Oracle RACでのDUPLICATE句

DUPLICATE句は、データベースですべてのIMCUのコピーを2つ目のデータベース・インスタンスに保持することを指定します。したがって、同じセグメントが正確に2つのOracle RACインスタンスに移入されます。

各オブジェクトで、1つのIMCUがプライマリとなります。セカンダリIMCUは、異なるデータベース・インスタンスに存在します。データベースでは、問合せの成立のためにどちらか一方のコピーを使用できます。IMCUのプライマリ・コピーがあるデータベース・インスタ

ンスに障害が発生した場合、データベースでは、残存するIMCUを使用して問合せを成立させることができます。

たとえば、パーティションsales_q1_2014に対してDUPLICATEを指定するとします。インスタンス1内およびインスタンス2内のIM列ストアには、同一のデータ・コピーがあります。インスタンス1が終了した場合は、インスタンス2上のIM列ストアにより、sales_q1_2014に対するリクエストを成立させることができます。

関連項目:

- 「インメモリー圧縮単位(IMCU)」
- DUPLICATE句についてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照

12.1.2.2.2 Oracle RACでのDUPLICATE ALL句

DUPLICATE ALL句は、すべてのインメモリー・オブジェクトをすべてのデータベース・インスタンスにミラー化することを指定します。

この設定により、高いレベルの冗長性が提供され、問合せが単一ノード内で完全に実行できることから線形の拡張性も実現されます。たとえば、sales表のすべてのIMCUが、インスタンス1、インスタンス2およびインスタンス3内のIM列ストアに移入されます。したがって、どのデータベース・インスタンスでも、salesの問合せでリクエストされたデータを取得できます。

DUPLICATE ALL句の結果、オブジェクトのすべてのIMCUが分散されるため、DISTRIBUTE副句の有用性はなくなります。複製はオブジェクト・レベルで指定します。これは、IM列ストア内のすべてのオブジェクトでDUPLICATE ALL句が必要となるわけではないことを意味します。

DUPLICATE ALL手法の主な利点は、次のとおりです。

- 高可用性

  すべてのインメモリー・オブジェクトに対してDUPLICATE ALL句を使用した場合、n個のインスタンスがあるOracle RACデータベースは、n-1個のOracle RACインスタンスの障害に耐えることができます。あるデータベース・インスタンスを保守のために稼働停止する必要がある場合に、重要なデータを少なくとも1つのIM列ストアで使用可能です。すべてのデータにアクセスできなくなる唯一のシナリオは、クラスタ内のすべてのデータベース・インスタンスの障害です。

- スター・クエリーのパフォーマンス

  問合せによって小さいディメンション表をパーティション化された大きいファクト表に結合する場合は、DUPLICATE ALLを使用して、ディメンション表をすべてのOracle RACインスタンスにミラー化できます。そのファクト表は、パーティションまたはサブパーティションによって分散されます。この計画では、すべてのデータベース・インスタンス内のIM列ストアに、スター型結合に必要なデータが含まれています。この手法は、ディメンション表全体がすべてのIM列ストアに移入されるため、パーティション・ワイズ結合に似ています。

関連項目:

DUPLICATE ALL句についてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照してください。

12.1.2.2.3 Oracle RACでのNO DUPLICATE句

デフォルトのNO DUPLICATE句は、データベースでオブジェクトの1つのコピーのみが保持されることを指定します。

たとえば、3つのノードを持つOracle RACデータベースは、インスタンス1にsales表の2012パーティションを格納し、インスタンス2に2013パーティションを格納し、インスタンス3に2014パーティションを格納できます。各表パーティションは、1つのデータベース・インスタンスにのみ存在します。

Oracle RACノードが列データを複製しない場合、障害発生ノード上の列データは、クラスタ上のIM列ストアで使用できません。不足しているデータに対して発行される問合せでは、エラーは発生しません。かわりに、データベース・バッファ・キャッシュまたは永続記憶域のいずれかのデータに問合せがアクセスするため、パフォーマンスが低下するおそれがあります。ノードがしばらく停止したままであり、残存するIM列ストア内に空き領域が存在する場合、Oracle RACでは、不足しているオブジェクトまたはオブジェクトの一部がクラスタ内の残りのノードに移入されます。

関連項目:

`NO DUPLICATE`句についてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照してください。

12.1.3 Oracle RACでの並列処理

データベース・インスタンスは、それらが存在するIM列ストア内のIMCUにアクセスする必要があります。Oracle RAC内のIM列ストアの移入およびアクセスは、すべてのIM列ストアにどのインスタンスからでもアクセスできるよう、並列で起こる必要があります。

12.1.3.1 Oracle RACでのシリアル問合せとパラレル問合せ

Oracle RAC内のDatabase In-Memoryは、シェアード・ナッシング・アーキテクチャです。少なくとも1つのパラレル・サーバー・プロセスがすべてのアクティブ・インスタンス上で実行されていないかぎり、問合せによる、IM列ストアからの必要なすべてのデータへのアクセスは保証されません。

シリアル問合せ

Oracle RACデータベース内の特定のノードで実行されるシリアル問合せでは、他のIM列ストア内のIMCUにアクセスできません。たとえば、インスタンス1で実行されているシリアル問合せで、`sales`の全体スキャンがリクエストされます。いくつかの`sales`パーティションはインスタンス1内のIM列ストアに移入されますが、他のパーティションはインスタンス2内のIM列ストアに移入されます。この問合せでは、インスタンス1上のIM列ストア内のIMCUにのみアクセスできます。残りのデータは、ディスク・ストレージから取得する必要があります。

パラレル問合せ

パラレル実行が有効でも、`PARALLEL_DEGREE_POLICY`初期化パラメータが`AUTO`に設定されていない場合、状況はシリアル問合せの場合と似ています。問合せコーディネータは、問合せを実行するデータベース上で動作します。PQプロセスにより、そのコーディネータにデータが送信されます。このケースでは、データベースによって複数のPQプロセスが開始されます。ただし、DOPが、問合せ内で参照されるオブジェクトのために移入されたIMCUを含むIM列ストアの数以上でないかぎり、すべてのデータがIM列ストアからアクセスできるわけではありません。自動DOPが有効になっていない場合は、少なくともDOPの大きさが問合せ内の移入されたオブジェクト用のIMCUを含むIM列ストアと同じになるようにします。

In-Memory動的スキャン

シリアル問合せとパラレル問合せの両方がIn-Memory動的スキャン(IM動的スキャン)を実行し、軽量スレッド・インフラストラクチャを使用できます。パラレル実行のインフラストラクチャは、新しいスレッド・インフラストラクチャと共存し、Oracle Database Resource Manager (リソース・マネージャ)によって動的に管理されます。`INMEMORY_SIZE`が`0`より大きい場合、リソース・マネージャはデフォルトで有効になります。

表スキャン・プロセスは、シリアル問合せのフォアグラウンド・プロセスまたはパラレル問合せのパラレル・サーバー・プロセスのいずれかになります。パラレル問合せがIM動的スキャンを実行するとき、すべての表スキャン・プロセスがスレッドのプールを所有できます。

関連項目:

- In-Memory動的スキャン
- リソース・マネージャについてさらに学習するには、『Oracle Database管理者ガイド』を参照
- Oracle RACでのパラレル問合せについてさらに学習するには、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照
- PARALLEL_DEGREE_POLICY初期化パラメータについてさらに学習するには、『Oracle Databaseリファレンス』を参照

12.1.3.2 Oracle RACでの自動DOP

自動並列度(自動DOP)により、オプティマイザはコストベースで計算を実行してSQL文の並列度を決定します。

PARALLEL_DEGREE_POLICY初期化パラメータをAUTOに設定することで、自動DOPを有効にします。オプティマイザでSQL文を解析するときには、実行時間が推定されます。この推定をPARALLEL_MIN_TIME_THRESHOLD初期化パラメータの設定に対して確認します。これは、IM列ストアが使用可能な場合に自動的に設定されます。その後、オプティマイザによって、次のようなコストベースの決定が行われます。

- 推定時間がPARALLEL_MIN_TIME_THRESHOLDより短い場合、その文は順次実行されます。
- 推定時間がPARALLEL_MIN_TIME_THRESHOLDより長い場合、その文は並列で実行されます。

  オプティマイザにより、リソース要件に基づいて並列度が計算されます。この計算は、PARALLEL_DEGREE_LIMIT初期化パラメータ、および構成されている場合はDatabase Resource Managerによって制限されます。

Oracle RAC環境でIM列ストアを使用している場合、この目的は、ディスクまたはバッファ・キャッシュへのアクセスを行わないようにすることです。この目的を達成するためには、すべてのアクティブ・データベース・インスタンス上で少なくとも1つのパラレル・サーバー・プロセスが実行されることを保証する必要があります。自動DOPは、この目的を達成するための推奨される方法です。

ノート:

自動 DOP を使用しない場合は、DOP が、問合せで要求される IMCU を含む IM 列ストアの数以上になるようにする必要があります。

自動DOPにより、すべての共有プールに、すべてのIMCUが存在する場所やそれらの大きさなどを示すメタデータが格納されるため、プロセスの適切な分散が保証されます。すべての共有プール内に同じマップが存在します。問合せがクラスタ内のどこで生じても、パラレル問合せコーディネータは、IMCUのホーム・ロケーション(それが存在するインスタンス)を認識しています。

たとえば、PQコーディネータは、2016年のsalesパーティションがインスタンス1にあるのに対して、2015年のパーティションがインスタンス2にあることを認識しています。インスタンス1上で実行されている問合せによって2015年と2016年のパーティションがリクエストされた場合、問合せコーディネータでは、そのホーム・ロケーションを使用してアクセス先のIM列ストアを決定します。DOPが十分に高い値に設定されている場合は、コーディネータによって自動的に両方のインスタンス上でPQプロセスが開始され、それらのプロセスにより、リクエストされたデータが問合せコーディネータに送り返されます。

関連項目:

自動DOPについて学習するには、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください。

12.1.4 Oracle RACでのFastStart領域

FastStart領域は、すべてのOracle RACノードにわたり共有されます。この機能により、クラスタ全体にわたり最大限の共有および再利用性が実現されます。

FastStart領域には、IMCUのコピーが1つのみ存在します。たとえば、ノード4つのクラスタ内で1つのオブジェクトに対して`DUPLICATE ALL`が指定されている場合、IM列ストアには、オブジェクトのコピーが4つ存在します。しかしながら、データベースにより、1つのコピーのみがFastStart領域に保存されます。

Oracle RACクラスタ内のどのデータベース・インスタンスでも、FastStart領域内のIMCUを使用できます。この機能により、Oracle RAC環境でのインスタンスの再起動のパフォーマンスが向上します。

たとえば、`sales`表に`sales_2014`、`sales_2015`および`sales_2016`という3つのパーティションがあり、各パーティションは異なるインスタンスに移入されているとします。インスタンス障害が発生し、1つのインスタンスを再起動できません。IM列ストア内に十分な空き領域がある場合は、残存するインスタンスで、アクセス不可のインスタンスに以前に移入されたIMCUを読み取ることができます。したがって、3つすべての`sales`表パーティションをアプリケーションで使用できます。

関連項目:

- 「Oracle RACでのFastStart領域」
- `DUPLICATE ALL`句についてさらに学習するには、『Oracle Database SQL言語リファレンス』を参照してください。

12.2 Oracle RACでのインメモリー・サービスの構成

サービスは、一連のインスタンスを意味します。Oracle RACでは、サービスを使用して、接続またはアプリケーションをクラスタ内のノードのサブセットに向けることができます。

関連項目:

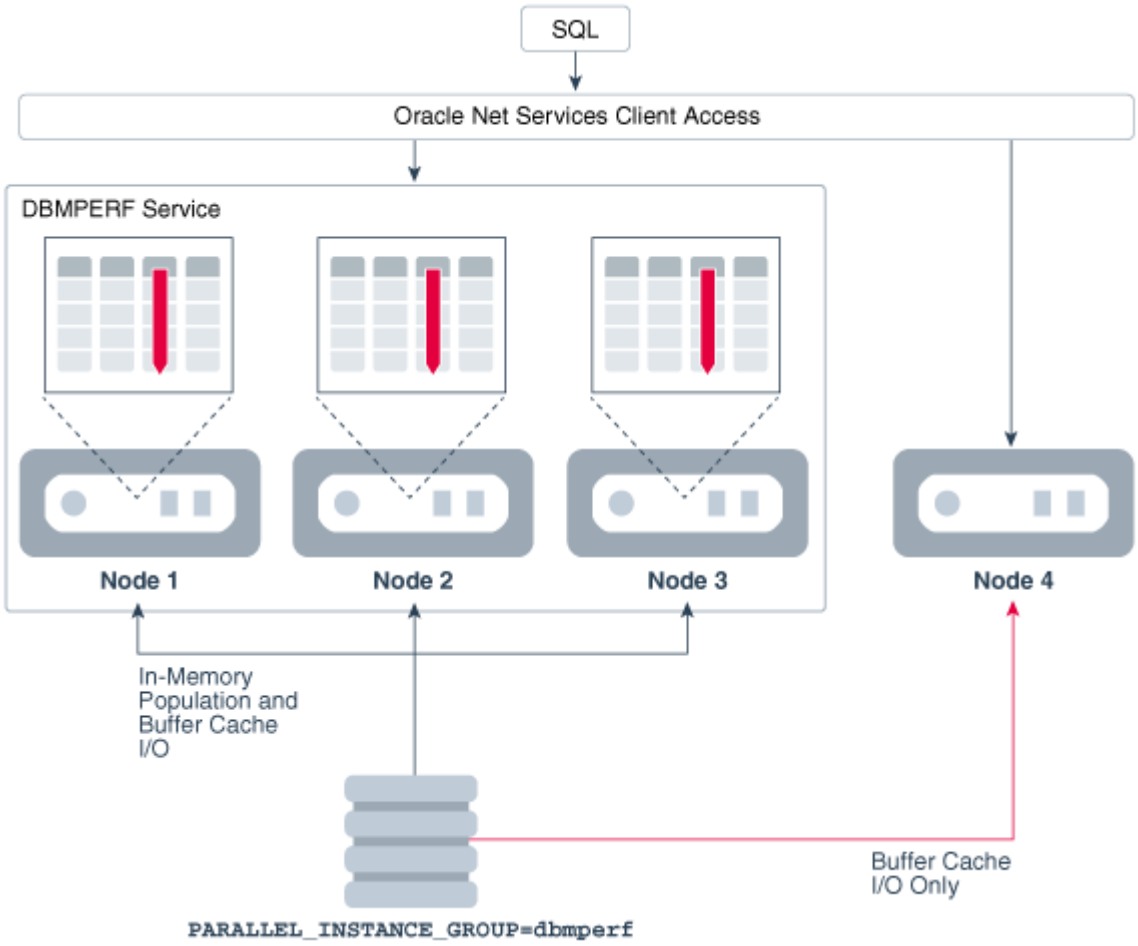
Oracle RACでのサービスについてさらに学習するには、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

12.2.1 インスタンスレベルのサービス制御

Oracle RACでは、IM列ストアの移入およびアクセスは、すべてのIM列ストアにどのデータベース・インスタンスからでもアクセスできるよう、並列で起こる必要があります。

`PARALLEL_INSTANCE_GROUP`初期化パラメータは、指定したサービスにパラレル問合せ操作を限定します。たとえば、クラスタ内の4つのデータベース・インスタンスのうちの3つにIM列ストアがある場合に、`dbmperf`という名前のサービスを作成し、`PARALLEL_INSTANCE_GROUP`を使用してこれら3つのインスタンスをこのサービスに割り当てるとします。その後、すべてのクライアント接続をその`dbmperf`サービスに限定できます。パラレル操作では、サービスで定義されたインスタンスに対してのみ、パラレル実行プロセスが生成されます。

図12-8 サービスへのインスタンスのサブセットの割当て



関連項目:

PARALLEL_INSTANCE_GROUP初期化パラメータについてさらに学習するには、『Oracle Databaseリファレンス』を参照してください。

12.2.2 オブジェクトレベルのサービス制御

個々のオブジェクトの場合、INMEMORY ... DISTRIBUTE句には、このサービスを実行できるデータベース・インスタンスへの移入を制限するFOR SERVICE副句があります。

PARALLEL_INSTANCE_GROUP初期化パラメータは、サービス・レベルでセグメントを制御します。ここでは、サービスは1つ以上のインスタンスを意味します。その一方、INMEMORY ... DISTRIBUTE FOR SERVICEは、セグメント・レベルで分散を制御します。たとえば、INMEMORYオブジェクトをインスタンス1のみ、インスタンス2のみ、または両方のインスタンス内のIM列ストアに移入するよう構成できます。

DISTRIBUTE FOR SERVICEのオプションは次のとおりです。

- DEFAULT - PARALLEL_INSTANCE_GROUPが設定されている場合、オブジェクトは、PARALLEL_INSTANCE_GROUPで指定されたIM列ストアがあるすべてのデータベース・インスタンスに移入されます。PARALLEL_INSTANCE_GROUPが設定されていない場合、オブジェクトは、IM列ストアがあるすべてのインスタンスに移入されます。

  FOR SERVICEを指定することは、FOR SERVICE DEFAULTを指定することと同じです。

- ALL - データベースにより、オブジェクトが、IM列ストアがあるすべてのインスタンスに移入されます。

ノート:

PARALLEL_INSTANCE_GROUP が設定されていない場合、DEFAULT と ALL は機能的に等しくなります。

- service_name - IMCOでは、その任務の一部として、前のサービスに割り当てられているデータベース・インスタンスからのオブジェクトの削除がトリガーされ、新しいサービスに割り当てられているインスタンスにそれが移入されます。

  セグメントを再分散する場合、データベースでは、必要な最小限の処理が実行されます。たとえば、サービスdbmperfがインスタンス1およびインスタンス2に割り当てられているとします。salesパーティションは、インスタンス1とインスタンス2の間で均等に分散されています。インスタンス3をこのサービスに追加します。均等な分散に必要になる場合は、データベースによって、IMCUがインスタンス3のみに移入され、それらがインスタンス1またはインスタンス2から削除されます。一部のIMCUは、それらの元の場所に残ります。

- NONE - IMCOにより、現在指定してあるサービスのIM列ストアからオブジェクトが削除されます。

オブジェクトのPRIORITYの値がNONE以外の場合は、Wnnnプロセスにより、次のIMCOサイクルの間のDDL実行またはサービス起動の後にオブジェクトが移入されます。ただし、オブジェクトのPRIORITYがNONEに設定されている場合、オブジェクトは、表の全体スキャンが実行中にのみ移入されます。スキャンによって、表に対する指定したサービスがアクティブでありブロックされていない、すべてのデータベース上でインメモリー移入がトリガーされます。このサービスは発行側サービスのスキャンとは異なる場合があることに注意してください。

インメモリー移入に使用されるサービスが停止した場合は、データベースにより、このサービスによって表されているIM列ストアからセグメントが削除されます。この点において、サービスの停止は、インスタンスのシャットダウンと同様です。このオブジェクトのINMEMORY属性は変わりません。サービスが再度起動されると、データベースによって、オブジェクトがそのINMEMORY属性に従って移入されます。IM列ストアからオブジェクトを削除するには、DDL文でNO INMEMORYを指定します。

DUPLICATEをDISTRIBUTE FOR SERVICEと組み合せることができます。たとえば、ノード4つのうち3つに割り当てられているサービスdbmperfに対してオブジェクトでDUPLICATE ALLを使用することを指定するとします。このケースでは、これら3つの各ノード上のIM列ストアにオブジェクトのコピーがあります。

関連項目:

- 「インメモリー移入の優先順位付け」
- DISTRIBUTE FOR SERVICE副句についてさらに学習するには、Oracle Database SQL言語リファレンスを参照

12.2.3 Oracle RACでのDatabase In-Memoryのサービスの利点

サービスとDUPLICATEの組合せで、ノード・アクセスおよびインメモリー移入を制御できます。

サービスには、次の利点があります。

- ローリング・パッチおよびアップグレード

  IM列ストアを含むインスタンスに対してクライアント問合せを行うよう、Oracle RACサービスを設定するとします。DUPLICATE句を使用する場合は、問合せ応答時間に影響を与えることなくインスタンスを削除することを選択できます。この方法は、削除したインスタンスのワークロードを処理するためにサービス内の他のインスタンス上に十分なリソー

スが存在することが前提となっています。

たとえば、ノード4つのクラスタでは、各ノードを順次削除すること、パッチを適用すること、および再度使用可能にすることができます。一時的にアクセス不可になっているノードのIMCUは、DUPLICATE句またはDUPLICATE ALL句を使用することで、1つ以上の他のノードで使用できます。したがって、列データへのアプリケーション・アクセスが中断されることはありません。

- アプリケーション・アフィニティ

  サービス名に基づいてアプリケーション・アクセスを単一ノードに限定できます。たとえば、サービスdbmperf1はノード1に限定され、サービスdbmperf2はノード2に限定されるなどです。アプリケーションが特定のサービスに接続してパラレル問合せを発行する場合、その問合せでは、同一サービスに属するノード上のプロセスが使用されます。たとえば、サービスdbmperf1に接続するアプリケーションでは、ノード1上のプロセスのみが使用されます。

  アプリケーションは、Oracle RACデータベース内に独立して共存して、列データにアクセスできます。完全に異なるオブジェクトは、各ノードに移入できます。たとえば、HRアプリケーションをサービスdbmperf1に向け、販売履歴アプリケーションをサービスdbmperf2に向けることができます。

関連項目:

- 「インメモリー移入に対するオブジェクトの手動による有効化について」
- DUPLICATEセマンティクスについては、『Oracle Database SQL言語リファレンス』を参照

12.2.4 ノードのサブセットのためのインメモリー・サービスの構成: 例

この作業では、インメモリー・サービスをOracle RACデータベース内のノードのサブセットに割り当てる方法を知ります。

目的は次のとおりです。

- RACデータベース内のノードのサブセット上でIM列ストアを作成します
- IM列ストアがあるノードのみにアクセスできるよう、サービスを定義します

前提

このタスクでは、次のことを想定しています。

- dbmmという名前のOracle RACデータベースにdbm1、dbm2、dbm3およびdbm4という4つのインスタンスがあります。「図12-8」を参照してください。
- dbm4以外のすべてのインスタンスで、INMEMORY_SIZEがゼロ以外の値に設定されています。
- dbmperfという名前のサービスを追加し、それをIM列ストアがある3つのノードに割り当てる必要があります。
- そのサービスに関連付けられているIM列ストアにsales表を移入する必要があります。

ノードのサブセットのためにインメモリー・サービスを構成するには:

1. IM列ストアを実行する3つのノードを表すサービスを作成します。

   オペレーティング・システムのコマンド・ラインで、次の形式でsrvctlコマンドを使用します。

   ```
   srvctl add service -db db_name –s service_name
     -preferred "instance_names"
   ```

   たとえば、次のコマンドを入力します。

```
srvctl add service –db dbmm –s dbmperf –preferred "dbm1, dbm2, dbm3"
```

2. サービスを開始します。

   たとえば、dbmperfサービスを起動するには、次のコマンドを使用します。

   ```
   srvctl start service -db dbmm -service "dbmperf"
   ```

3. サービスに接続するためのネット・サービス名を作成します。

   たとえば、次のようにtnsnames.oraファイルを更新します。

   ```
   DBMPERF =
     (DESCRIPTION =
        (ADDRESS =
          (PROTOCOL = TCP)
          (HOST = host_name)
          (PORT = listener_port))
        (CONNECT_DATA =
          (SERVER = DEDICATED)
          (SERVICE_NAME = DBMPERF)
        )
     )
   ```

4. DISTRIBUTE FOR SERVICE副句を使用して、移入する予定の表にINMEMORY属性を割り当てます。

   たとえば、次のようにsalesを変更します。

   ```
   ALTER TABLE sales INMEMORY DISTRIBUTE FOR SERVICE "dbmperf";
   ```

   上の文では、デフォルトのPRIORITY設定であるNONEがsales表に対して使用されます。そのため、この表の移入は、自動ではなく要求に応じて行われます。

5. sales表を移入するには、dbmperfサービスに接続してから、表の全体スキャンを開始します。

   たとえば、次のようにsalesを問い合せることで、全体スキャンを強制的に実行します。

   ```
   SELECT /*+ FULL(s) */ COUNT(*) FROM sales s;
   ```

関連項目:

- ALTER TABLEのINMEMORY DISTRIBUTE FOR SERVICE句についてさらに学習するには、Oracle Database SQL言語リファレンスを参照
- srvctlユーティリティについてさらに学習するには、Oracle Database管理者ガイドを参照
- ネット・サービス名についてさらに学習するには、Oracle Database Net Services管理者ガイドを参照

13 Oracle Active Data GuardでのIM列ストアのデプロイ

この章では、どのようにDatabase In-Memoryの機能がOracle Active Data Guard環境で動作するかを説明します。

13.1 Database In-MemoryおよびActive Data Guardについて

Oracle Database 12cリリース2 (12.2.0.1)以降、Oracle Database In-MemoryはOracle Engineered SystemsまたはOracle Cloud Platform as a Serviceを使用してOracle Active Data Guard環境内でサポートされています。

関連項目:

Oracle Active Data Guardの概要は、Oracle Data Guard概要および管理を参照してください。

13.1.1 Oracle Active Data GuardでのIM列ストアの目的

IM列ストアは、プライマリ・データベース上のみ、スタンバイ・データベース上のみ、またはプライマリおよびスタンバイ・データベースの両方で構成できます。

IM列ストアを両方のデータベースに対して構成する場合は、同一または異なるオブジェクト・セットをそれら2つのインスタンスに移入できます。この手法では、IM列ストアのサイズが効率的に増加します。

13.1.1.1 プライマリおよびスタンバイ・データベース内に同一のIM列ストア

最も単純なシナリオでは、プライマリおよびスタンバイ・データベースのどちらにも、同じサイズでIM列ストアが含まれます(必須ではない)。それらのIM列ストアには、同じオブジェクトが含まれています。

このシナリオの利点は、分析問合せにより、どちらのデータベース上のIM列ストアにもアクセスできることです。したがって、分析問合せをスタンバイ・データベースに指示し、プライマリ・データベースではリソースを消費しないようにすることができます。結果として、スタンバイ・データベースが分析ワークロードをサポートする一方で、プライマリ・データベースがトランザクション・ワークロードをサポートできます。

主な作業は、次のとおりです。

- INMEMORY_SIZE初期化パラメータをプライマリおよびスタンバイ・データベース・インスタンスの両方で設定します。
- 必ずスタンバイ・データベース・インスタンスでINMEMORY_ADG_ENABLED初期化パラメータがTRUE (デフォルト)に設定されるようにします。
- それら2つのIM列ストアに移入されるすべてのオブジェクトに対して、INMEMORY属性を設定します。

オブジェクトのINMEMORY属性を変更した場合は、プライマリ・データベースにより、その変更内容がスタンバイ・データベースに伝えられます。たとえば、NO INMEMORY属性をsales表で設定した場合は、両方のIM列ストアでsalesが強制的に除去されます。

プライマリ・データベースでは、IM列ストアへの移入のために表の列のサブセットを有効にできます。異なる列に対して異なる圧縮レベルを指定することもできます。特定列の有効化には、ディクショナリ変更が含まれます。プライマリ・データベース上のDDLは、Oracle Active Data Guardデータベースに伝搬されます。

関連項目:

- 「データベースに対するIM列ストアの有効化」
- CREATE TABLE文の詳細は、『Oracle Database SQL言語リファレンス』を参照
- INMEMORY_SIZEおよびINMEMORY_ADG_ENABLED初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照

13.1.1.2 スタンバイ・データベース内のみにIM列ストア

このシナリオでは、IM列ストアは、スタンバイ・データベースには存在しますが、プライマリ・データベースには存在しません。

このシナリオでは、プライマリ・データベースは純粋なOLTPデータベースとして機能できます。IM列ストアのためにプライマリ・データベースで余分なメモリーは必要になりません。パフォーマンスを犠牲にすることやプライマリ・データベース上のリソースを消費することなく、分析レポート・アプリケーションをスタンバイ・データベースに向けることができます。

主な作業は、次のとおりです。

- INMEMORY_SIZE初期化パラメータをスタンバイ・データベース・インスタンスでゼロ以外の値に設定し、プライマリ・データベース・インスタンスでは0に設定します。
- 必ずスタンバイ・データベース・インスタンスでINMEMORY_ADG_ENABLED初期化パラメータがTRUE (デフォルト)に設定されるようにします。
- スタンバイ・データベース内のIM列ストアに移入されるすべてのオブジェクトに対して、DISTRIBUTE FOR SERVICE句でINMEMORY属性を設定します。

  たとえば、プライマリ・データベースにログインし、INMEMORY属性をsh.sales表で設定した場合、IM列ストアはこのデータベースに存在しないため、この表はプライマリ・データベース上のIM列ストアに移入されません。ただし、スタンバイ・データベースでは、sh.sales表のINMEMORY属性が継承されます。この表は、スタンバイ・データベース内のIM列ストアに移入されます。

13.1.1.3 プライマリおよびスタンバイのIM列ストア内に異なるオブジェクト

最も柔軟性のあるシナリオは、IM列ストアをプライマリおよびスタンバイ・データベースに対して別々に構成するというものです。

このシナリオの利点は、各データベースで異なるワークロードを実行できるということです。たとえば、HRアプリケーションがプライマリ・データベースでレポートを実行する一方で、販売履歴アプリケーションがスタンバイ・データベースでレポートを実行します。したがって、どちらのデータベースも、分析レポートのすべての負荷に耐えることはありません。

主な作業は、次のとおりです。

- スタンバイおよびプライマリ・データベース・インスタンスで、INMEMORY_SIZE初期化パラメータをゼロ以外の値に設定します。それらの値を同一にする必要はありません。
- 必ずスタンバイ・データベース・インスタンスでINMEMORY_ADG_ENABLED初期化パラメータがTRUE (デフォルト)に設定されるようにします。
- それら2つのIM列ストアに移入されるすべてのオブジェクトに対して、INMEMORY ... DISTRIBUTE FOR SERVICE句を設定します。サービスにより、オブジェクトを移入するインスタンスが指定されます。

サービス3つの構成

一般的な構成では、スタンバイのみ、プライマリのみ、およびプライマリとスタンバイという3つのサービスを作成します。たとえば、プ

ライマリ・インスタンスでは最新月のsalesファクト表データが必要だが、スタンバイ・インスタンスでは前のsalesデータが必要だとします。両方のインスタンスに移入されたディメンション表が必要です。salesパーティションごとに、INMEMORY ... DISTRIBUTE FOR SERVICEを使用して、スタンバイまたはプライマリ・サービスのどちらかを指定します。ディメンション表ごとに、プライマリおよびスタンバイ・データベース・インスタンスを両方とも含むサービスを指定します。

ノート:

サービス名がプライマリおよびスタンバイ・インスタンスの両方に対して定義されているかぎり、DISTRIBUTE FOR SERVICE で同じサービス名を指定して、プライマリおよびスタンバイ・データベースに同じ表を移入できます。

Oracle RACおよびOracle Active Data Guard

Oracle RACでは、移入のためのインスタンスを指定するFOR SERVICE句を、IMCUの分散を制御するDISTRIBUTE AUTO句またはDISTRIBUTE BY句と組み合せることができます。ただし、Oracle Active Data Guardでは、FOR SERVICE句は、指定されたオブジェクトを移入するプライマリまたはスタンバイ・インスタンスを指定します。DISTRIBUTE AUTOまたはDISTRIBUTE BYを使用してプライマリ・インスタンスとスタンバイ・インスタンスの間でIMCUを分散することはできません。たとえば、IMCUの半分がプライマリ・インスタンス内に、IMCUのもう半分がスタンバイ・インスタンス内に存在するように、プライマリ・インスタンスとスタンバイ・インスタンスとの間でsales表の移入を分割することはできません。

関連項目:

- 「オブジェクトレベルのサービス制御」
- DISTRIBUTE FOR SERVICE副句についてさらに学習するには、Oracle Database SQL言語リファレンスを参照

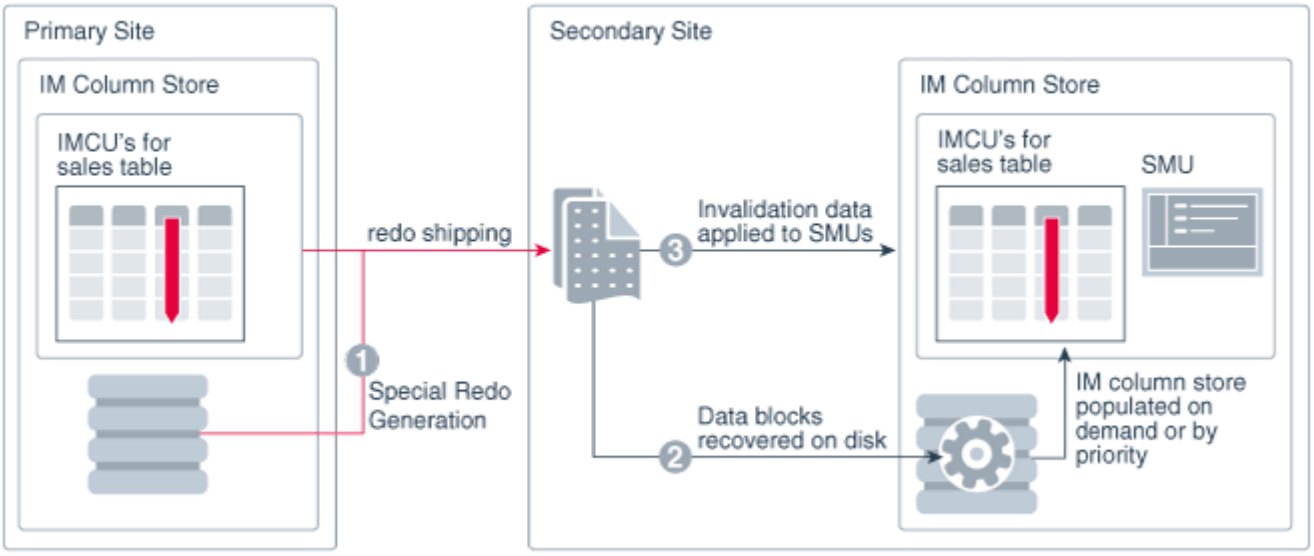
13.1.2 IM列ストアのOracle Active Data Guardでの機能

Oracle Active Data Guard環境では、オブジェクトレベルのPRIORITY属性によって移入が管理されます。オブジェクトは、サービスがアクティブなデータベース・インスタンスにのみ移入されます。

移入は、PRIORITY値に応じて、オンデマンドまたは優先順位ベースのどちらかとなります。ロール変更またはスイッチオーバーがあった場合、データベースでは、サービスを新たにマップするデータベース・インスタンスのセットに従って、表が再移入されます。

次の図では、プライマリ・データベースからのREDOによるスタンバイ・データベースの更新の内部メカニズムを示します。

図13-1 スタンバイ・データベースの更新



このプロセスは、次のとおりです。

1. プライマリ・データベースでREDOが生成されてから、そのREDOがスタンバイ・データベースに転送されます。

   すべてのDML文に対してプライマリ・データベース上で生成されたREDOには、変更が`INMEMORY`オブジェクトに対するものであるかどうかを示すメタデータが含まれています。

2. スタンバイ・データベースにより、ディスクに格納されたデータ・ブロックにREDOが適用されます。

   プライマリ・データベースで行われている操作から生成されたREDOがスタンバイ・データベースによって適用されるため、スタンバイ・データベースにより、それらのトランザクションの一貫性が保たれます。

3. `INMEMORY`オブジェクトが変更される場合は、スタンバイ・データベースにより、プライマリ・データベースで行われるのと同様に、変更された行が無効になり、トランザクション・ジャーナルおよびスナップショット・メタデータ単位(SMU)を使用して変更が追跡されます。

再移入のメカニズムは、プライマリ・データベースと同じようにスタンバイ・データベースで機能します。十分な数のDMLがオブジェクト上で発生して内部しきい値に達すると、スタンバイ・データベースにより、そのオブジェクトがIM列ストアに再移入されます。

関連項目:

- 「IM列ストアの再移入について」
- 「結合グループについて」
- 複数インスタンスのREDO適用についてさらに学習するには、Oracle Data Guard概要および管理を参照

13.1.3 Active Data GuardでのIn-Memoryの制限

Active Data Guardでは、In-Memoryのほとんどの機能がサポートされています。

スタンバイ・データベースでは、次に示すIn-Memoryの機能はサポートされていません。

- IMファスト・スタート
- 結合グループ
- IM式の取得
- ヒート・マップ(プライマリ・データベースのみを反映します)

Data Guard Multi-Instance Redo ApplyはActive Data Guardスタンバイ・データベースのIM列ストアでサポートされていますが、初期化パラメータENABLE_IMC_WITH_MIRAがTRUEに設定されている場合のみです。デフォルトでは、ENABLE_IMC_WITH_MIRAはFALSEです。

関連項目:

ENABLE_IMC_WITH_MIRAについてさらに学習するには、『Oracle Databaseリファレンス』を参照

13.2 Oracle Active Data Guard環境でのIM列ストアの構成

Oracle Active Data GuardでのIM列ストアの構成には、INMEMORY_SIZEの設定、および移入されるオブジェクトに対する適切なINMEMORY属性の設定が必要となります。

この作業は、Oracle Active Data Guardの概要および手順について知識があることが前提となっています。

前提条件

次の要件を満たす必要があります。

- スタンバイ・データベースは、Oracle Engineered System上で、またはOracle Cloud Platform as a Service内で実行する必要があります。
- COMPATIBLE初期化パラメータは、12.2.0以上にする必要があります。
- データベースごとに異なるオブジェクトを移入するには、適切なサービスを構成します。

Oracle Active Data Guard環境でIM列ストアを構成するには:

1. IM列ストアを含むデータベース・インスタンス上でINMEMORY_SIZE初期化パラメータを設定します。

   次のガイドラインに従ってください。

   - プライマリおよびスタンバイ・データベース上のIM列ストアを構成するには、両方のデータベース・インスタンス上でINMEMORY_SIZEを設定します。
   - スタンバイ・データベースのみでIM列ストアを構成するには、スタンバイ・データベース・インスタンス上でINMEMORY_SIZEを設定します。
2. 必ずスタンバイ・データベース・インスタンスでINMEMORY_ADG_ENABLED初期化パラメータがTRUE (デフォルト)に設定されるようにします。
3. オプションで、IM列ストアでMulti-Instance Redo Applyを有効にする場合は、ENABLE_IMC_WITH_MIRA初期化パラメータをTRUEに設定します。
4. プライマリ・データベースで、INMEMORY属性を使用してDDL文を実行します。

   タスクはIM列ストアが存在する場所、およびオブジェクトを移入するIM列ストアによって異なります。

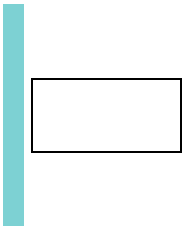
   - オブジェクトをスタンバイ・データベースにのみ移入するには、スタンバイ・データベースでのみ実行される有効なサービスを指定するDISTRIBUTE FOR SERVICE句をINMEMORY属性に設定します。

     REDO転送中に、スタンバイ・データベースはこのDDL文をプライマリ・データベースから受信します。スタンバイ・データベース上で通常の方法で移入が行われます。たとえば、salesにINMEMORY属性および優先順位NONEが設定されている場合は、移入を行うためにその表に全体スキャンを実行する必要があります。
   - オブジェクトをプライマリ・データベースにのみ移入するには、プライマリ・データベースでのみ実行される有効な

サービスを指定するDISTRIBUTE FOR SERVICE句をINMEMORY属性に設定します。

- プライマリ・データベースとスタンバイ・データベースの両方にオブジェクトを移入するには、次のいずれかのアクションを実行します。
  - DISTRIBUTE FOR SERVICE句を設定しません。
  - DISTRIBUTE FOR SERVICE servicenameを設定します。ここで、servicenameは、プライマリ・データベースとスタンバイ・データベースの両方で実行されているサービスです。
  - DISTRIBUTE FOR SERVICE DEFAULTを設定して、オブジェクトがPARALLEL_INSTANCE_GROUP初期化パラメータで指定されたすべてのインスタンスに対する移入の対象となるようにします。
  - DISTRIBUTE FOR SERVICE ALLを設定して、PARALLEL_INSTANCE_GROUP初期化パラメータの設定にかかわらず、オブジェクトがすべてのインスタンスに対する移入の対象となるようにします。

オブジェクトの移入は、優先度設定に従って、プライマリ・データベースまたはスタンバイ・データベースで行われます。たとえば、スタンバイ・データベース上のsalesの優先順位がNONEの場合は、salesの全体スキャンをトリガーするスタンバイ・データベースの問合せにより、この表がスタンバイIM列ストアに移入されます。



ノート:

スタンバイ・データベース上の sales の全体スキャンによって、この表がプライマリ・データベース内の IM 列ストアに移入されることはありません。

関連項目:

- 「インメモリー移入の優先順位付け」
- 『Oracle Data Guard概要および管理』
- DISTRIBUTE FOR SERVICE副句についてさらに学習するには、Oracle Database SQL言語リファレンスを参照
- INMEMORY_SIZE、INMEMORY_ADG_ENABLEDおよびENABLE_IMC_WITH_MIRA初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照

第V部 Database In-Memoryリファレンス

この部には、インメモリー列ストア(IM列ストア)に関連する初期化パラメータおよびビューが含まれています。

14 インメモリー初期化パラメータ

IM列ストアの動作を制御するいくつかの初期化パラメータを示します。

この章は要約のみです。Oracle Databaseリファレンスには、すべてのデータベース・ビューについての完全な情報が含まれています。

表14-1 IM列ストアに関連する初期化パラメータ

初期化パラメータ	説明
INMEMORY_ADG_ENABLED	スタンバイ・データベース上で Oracle Active Data Guard のインメモリーが有効(TRUE)になっているか無効(FALSE)になっているかを示します。デフォルトは TRUE です。 Active Data Guard の場合、メディア・リカバリでは、REDO を適用するときにインメモリー・オブジェクトを取得して、問合せの進行後に関連オブジェクトを無効にする必要があります。このパラメータは、メディア・リカバリで取得および無効化を実行するかどうかを制御します。 スタンバイ・リカバリが実行されていない場合のみ、このシステムレベル・パラメータを変更できます。スタンバイ・データベースで Oracle RAC が使用されている場合は、このパラメータをすべてのインスタンス上で同じ値に設定する必要があります。
INMEMORY_AUTOMATIC_LEVEL	作業データセットが常に IM 列ストア内にあるように保証することによって、IM 列ストアの管理を自動化します。 次の値を設定できます。 ● OFF: これがデフォルトです。この値が設定されている場合、自動インメモリーは無効になります。この値は、IM 列ストアを Oracle データベース 18c より前の動作に戻します安定した作業データ・セットがないと思われる場合は、パラメータを DISABLE に設定します。 ● LOW:この値が設定されている場合、データベースはメモリー不足のときに IM 列ストアからコールド・セグメントを除去します。 ● MEDIUM:このレベルでは、メモリー不足のために

初期化パラメータ	説明
	移入されなかったホット・セグメントが最初に移入されるように保証する追加の最適化が含まれます。
INMEMORY_CLAUSE_DEFAULT	新しい表およびマテリアライズド・ビューに対してデフォルトの IM 列ストア句を指定します。 このパラメータでは、次のオプションがサポートされています。 ● 新しい表およびマテリアライズド・ビューに対してデフォルトの IM 列ストア句がないことを指定するには、このパラメータを未設定のままにするか、空の文字列に設定します。このパラメータを NO INMEMORY に設定すると、デフォルト値(空の文字列)に設定したのと同じ結果となります。 ● すべての新しい表およびマテリアライズド・ビューに対して句がデフォルトであることを指定するには、このパラメータを有効な INMEMORY 句に設定します。句には、IM 列ストアの圧縮方法およびデータ移入オプションに関する有効な句を含めることができます。オプションは次のとおりです。 ● 句が INMEMORY で始まる場合、INMEMORY 句が指定されていないものも含め、すべての新しい表およびマテリアライズド・ビューが IM 列ストアに移入されます。 ● 句から INMEMORY が省略されている場合、作成時に INMEMORY 句を指定して IM 列ストアに対して有効になっている新しい表およびマテリアライズド・ビューにのみ適用されます。
INMEMORY_EXPRESSIONS_USAGE	どの IM 式が IM 列ストアへの移入の対象かを制御します。 このパラメータでは、次のオプションがサポートされています。

初期化パラメータ	説明
	● ENABLE データベースでは、静的 IM 式と動的 IM 式の両方が IM 列ストアに移入されます。この値を設定すると、一部の表でインメモリー・フットプリントが増加します。これはデフォルトです。 ● STATIC_ONLY 静的構成では、IM 列ストアで OSON (バイナリ JSON)列をキャッシュできます。それらは、IS_JSON チェック制約でマークされます。内部的には、OSON 列は、SYS_IME_OSON という名前の非表示仮想列です。 ● DYNAMIC_ONLY データベースでは、SYS_IME 非表示仮想列として表に追加された、頻繁に使用されているかホットな式のみが移入されます。この値を設定すると、一部の表でインメモリー・フットプリントが増加します。 ● DISABLE データベースでは、静的か動的かに関係なく、IM 式は IM 列ストアに移入されません。 ノート: IM 式では、NLS 依存のデータ型はサポートされていません。
INMEMORY_FORCE	IM 列ストアに対して表およびマテリアライズド・ビューを有効化または無効化します。 このパラメータでは、次のオプションがサポートされています。 ● 移入を決定するために INMEMORY または NO INMEMORY 属性を使用可能にするには、このパラメータを DEFAULT (デフォルト値)に設定し

初期化パラメータ	説明
	ます。この値は動的に設定できます。 ● IM 列ストアに対してすべての表およびマテリアライズド・ビューを無効にするには、このパラメータを OFF に設定します。この値は動的に設定できます。 ● Database In-Memory ベース・レベルを有効にするには、CDB ルートの初期化パラメータ・ファイル(PDB レベルではなく)でこのパラメータに BASE_LEVEL を設定します。この値は動的に設定できません。 CDB でベース・レベルが使用されている場合、自動インメモリーは無効になり、INMEMORY オブジェクトおよび列の圧縮レベルは、QUERY LOW に自動的に設定されます。 ● IM 列ストアの作成によるオーバーヘッドを発生させることなく CellMemory 機能を使用するには、このパラメータを CELLMEMORY_LEVEL に設定します。 この値は動的に設定できません。 INMEMORY_SIZE の値が 0 より大きい場合、INMEMORY_FORCE=CELLMEMORY_LEVEL 設定は、INMEMORY_FORCE=DEFAULT 設定と同等であることに注意してください。CellMemory のみを使用している場合でも、IM 列ストアは有効になります。
INMEMORY_MAX_POPULATE_SERVERS	領域管理ワーカー・プロセス(Wnnn)によって他のシステムに負荷がかかりすぎないよう、移入に使用するそれらのプロセスの数を最大で指定します。 システムのコア数に基づいて、このパラメータを適切な値に設定します。デフォルトは、有効 CPU スレッド数の半分、または PGA_AGGREGATE_TARGET 値を 512 MB で割った値の、どちらか少ないほうです。 ノート: INMEMORY_MAX_POPULATE_SERVERS が 0 に設定されている場合、IM 列ストアにオブジェクトを移

初期化パラメータ	説明
	入できません。
INMEMORY_OPTIMIZED_ARITHMETIC	NUMBER 列が、インメモリー最適化された形式で格納されるかどうかを制御します。 このパラメータでは、次のオプションがサポートされています。 ● DISABLE (デフォルト)は、最適化されたエンコーディングを使用しません。 ● ENABLE は、SIMD ハードウェアを使用したネイティブ計算を可能にする最適化された形式で数値をエンコードします。この最適化は、FOR QUERY LOW 圧縮を含む表で使用できます。
INMEMORY_QUERY	インメモリー問合せを許可するかどうかを指定します。 このパラメータでは、次のオプションがサポートされています。 ● ENABLE (デフォルト)は、移入されたオブジェクトに問合せがアクセスすることを許可します。 ● DISABLE は、移入されたオブジェクトへのアクセスをブロックします。
INMEMORY_SIZE	データベース・インスタンスでの IM 列ストアのサイズを設定します。 デフォルト値は 0 であり、これにより、IM 列ストアは無効になります。ゼロ以外の最小設定は 100M です。 インメモリーFree Tier オプションの場合、非 CDB または CDB のサイズが 16 GB を超えないようにする必要があります。Oracle RAC データベースでは、各インスタンスの INMEMORY_SIZE 設定が 16 GB を超えないようにする必要があります。
INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT	トリクル再移入を実行する、移入プロセスと再移入プロセスの合計パーセンテージを制限します。

初期化パラメータ	説明
	このパラメータの値は、INMEMORY_MAX_POPULATE_SERVERS 初期化パラメータの値の割合になります。たとえば、INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT が 5 パーセントで、INMEMORY_MAX_POPULATE_SERVERS が 20 の場合、IM 列ストアでは、平均値である 1 コア(.05 * 20)がトリクル再移入のために使用されます。
INMEMORY_VIRTUAL_COLUMNS	どの式を IM 列ストアに移入するかを制御します。 このパラメータでは、次のオプションがサポートされています。 ● MANUAL (デフォルト)にすると、明示的に INMEMORY として指定されている IM 仮想列の移入が許可されます。 ● ENABLE にすると、INMEMORY 表内のすべての IM 仮想列について、明示的に IM 列ストアから除外されていないかぎり、移入が許可されます。 ● DISABLE にすると、移入の対象となる IM 仮想列がないことが指定されます。
OPTIMIZER_INMEMORY_AWARE	Database In-Memory のためのオプティマイザ・コスト・モデル機能拡張を制御します。 このパラメータでは、次のオプションがサポートされています。 ● TRUE (デフォルト)にすると、INMEMORY オブジェクトを参照する SQL 文が最適化されます。 ● FALSE にすると、最適化中に表の INMEMORY 属性が無視されます。

15 インメモリー・ビュー

このトピックでは、インメモリー列ストア(IM列ストア)に関連するデータ・ディクショナリおよび動的パフォーマンス・ビューについて説明します。

表15-1 インメモリー・ビュー

初期化パラメータ	説明
DBA_EXPRESSION_STATISTICS	データベース内のすべての表について式の使用状況の追跡統計を示します。
DBA_EXTERNAL_TABLES	データベース内の外部表をすべて示します。INMEMORY 列は、表が INMEMORY および INMEMORY_COMPRESSION に有効かどうかを示し、圧縮レベルを示しています。 関連するビューには、ALL_XTERNAL_PART_TABLES、ALL_XTERNAL_TAB_PARTITIONS および ALL_XTERNAL_TAB_SUBPARTITIONS が含まれます。
DBA_FEATURE_USAGE_STATISTICS	データベース機能の使用状況の統計情報を示します。IM 列ストアにアクセスしている場合は、NAME 列に In-Memory Column Store と示されます。
DBA_HEAT_MAP_SEGMENT	すべてのセグメントの最新のセグメント・アクセス時刻を示します。ビュー内のタイムスタンプには、1 日のヒート・マップのフラッシュ回数が粗粒度で反映されます。
DBA_INMEMORY_AIMTASKS	自動インメモリーによって作成されたタスクのステータスを表示します。
DBA_INMEMORY_AIMTASKDETAILS	自動インメモリーが実行するオブジェクトとアクションについて説明します。
DBA_ILMDATAMOVEMENTPOLICIES	データベース内の自動データ最適化ポリシーのデータ移動関連属性固有の情報を示します。
DBA_IM_EXPRESSIONS	列の接頭辞として SYS_IME が付けられている、INMEMORY 属性がある IM 式を示します。
DBA_JOINGROUPS	データベース内の結合グループを示します。結合グループは、効果的に結合できる 2 つの列をリストするユーザー作成オブジェクトです。
DBA_SEGMENTS	データベース内のすべてのセグメントに割り当てられた記憶域を示します。INMEMORY および INMEMORY_PRIORITY を含め、いくつかの列で

初期化パラメータ	説明
	は、セグメントのインメモリー属性が示されます。
DBA_TABLES	どの表に INMEMORY 属性が設定されているか(INMEMORY 列が ENABLED)または設定されていないか(DISABLED)を示します。
V$ACTIVE_SESSION_HISTORY	サンプル化されたセッション・アクティビティを示します。INMEMORY_QUERY および INMEMORY_POPULATE を含め、いくつかの列では、サンプリング時にインメモリー列ストアに関連するセッション・アクティビティが示されます。
V$HEAT_MAP_SEGMENT	リアルタイムのセグメント・アクセス情報を表示します。
V$IM_COLUMN_LEVEL	CREATE TABLE 文の INMEMORY 句を使用して定義されている、選択された列圧縮レベルを示します。SYS_IME 非表示仮想列は、このビューでは示されません。
V$IM_SEGMENTS	データベース内のすべてのインメモリー・セグメントに関する情報を示します。インメモリー表現があるセグメントのみが表示されます。セグメントが IM 列ストアのためにマークされているが移入されていない場合、ビューには、このセグメントのための対応する行は含まれません。
V$INMEMORY_AREA	インメモリー領域内の領域割当てに関する情報を示します。
V$INMEMORY_FASTSTART_AREA	インメモリー・ファスト・スタート(IM ファスト・スタート)領域に関する情報を示します。
V$SGA	インメモリー領域のサイズを示します。

A Cloud ControlでのIM列ストアの使用

Oracle Enterprise Manager Cloud Control (Cloud Control)でIM列ストアを構成および管理できます。

A.1 Cloud ControlでのIM列ストアの使用のための前提条件を満たす方法

IM列ストアの使用のためにデータベースを有効にする前に、COMPATIBLEが12.1.0.0以上に設定されていることを確認します。

互換性レベルを設定するには、次のステップを実行します。

1. Enterprise Managerで「データベース・ホーム」ページから「管理」メニューの「初期化パラメータ」を選択し、「初期化パラメータ」ページに移動します。

   このページを使用すると、互換性レベルを設定または変更できます。

2. COMPATIBLE初期化パラメータを探します。

   パラメータのカテゴリは「その他」です。

3. 値を12.1.0.0に変更し、「適用」をクリックします。

   Cloud Controlにより、データベースを再起動するよう求めるメッセージが表示されます。データベースが再起動されると、新しい値が有効になります。

IM列ストアのサイズを設定または変更するには、次のステップに従います。

1. Enterprise Managerで「データベース・ホーム」ページから「管理」メニューの「初期化パラメータ」を選択し、「初期化パラメータ」ページに移動します。
2. パラメータINMEMORY_SIZEを検索します。パラメータのカテゴリは「インメモリー」です。
3. 値を変更し、「適用」をクリックします。

   この値は、最小サイズである100 MBよりも大きな値に設定できます。

   データベースを再起動するよう求められます。

A.2 インメモリー列ストアの中央ホーム・ページを使用したデータベース・オブジェクトのインメモリー・サポートの監視

インメモリー列ストア中央ホーム・ページを使用して、表、索引、パーティション、表領域などのデータベース・オブジェクトに対するインメモリー・サポートを監視します。オブジェクトに対するインメモリー機能を表示し、インメモリーの使用状況統計を監視できます。

インメモリー列ストアの中央ホーム・ページで、次のアクションを完了できます。

- 「インメモリー・オブジェクトのアクセス・ヒート・マップ」には、IM列ストア内の上位100個のオブジェクトと、それらの相対サイズが表示され、オブジェクトのアクセス頻度が様々な色で表されて示されます。ヒート・マップをアクティブにするには、初期化パラメータ・ファイルでヒート・マップのためのオプションをオンにする必要があります。一般に、マップがアクティブ化される前の待機期間は1日です。日付セレクタを使用して、ヒート・マップに表示されるオブジェクトの日付範囲を選択できます。スライダを使用して色の粒度を制御することもできます。

- 「構成」セクションを使用して、「インメモリー問合せ」、「インメモリー強制」、「デフォルトのインメモリー句」などのステータス設定を表示します。「編集」をクリックして、このセクションに表示される値と設定を変更できる初期化パラメータ・ページに移動します。「パフォーマンス」セクションを使用して、「アクティブ・セッション」のメトリックを表示します。
- 「オブジェクト・サマリー」セクションを使用して「圧縮係数」と移入オブジェクトに使用したメモリーに関するデータを表示します。「インメモリー対応オブジェクト統計」は、ページの「インメモリー対応オブジェクト統計の表示」リンクからドリルダウンするとポップアップ・ウィンドウで表示されます。
- 「インメモリー・オブジェクトの分散」セクションを使用して、メモリー内で使用されている様々なオブジェクトのパーセント基準の分散を表示します。このセクションには、「パーティション」、「サブパーティション」、「パーティション化されていない表」および「パーティション化されていないマテリアライズド・ビュー」の分散を示すグラフが含まれます。それぞれの数値はグラフの上に表示されます。
- 「インメモリー・オブジェクトの検索」セクションを使用して、インメモリーで使用するために指定されたオブジェクトを検索します。検索に使用するパラメータを入力した後で「検索」をクリックします。結果の表には、見つかった各オブジェクトの「名前」が「サイズ」、メモリー内サイズ、ディスク上のサイズ、インメモリーの割合およびその「インメモリー」パラメータとともに表示されます。また、インメモリー、インメモリー以外いずれかのアクセス・オブジェクトも検索できます。ヒート・マップが有効になっている場合は、「インメモリー・オブジェクトの検索」ボックスの「ビュー」フィールドにあるドロップダウン・リストに「アクセス済オブジェクト」オプションが表示されます。アクセス・オブジェクトを選択すると、インメモリー、インメモリー以外いずれかのアクセス・データを含む上位100オブジェクトをフィルタ処理できます。時間範囲を選択し、その範囲内でオブジェクトを検索できます。インメモリーのすべてのオブジェクト・オプションを選択すると、インメモリー・サイズに基づくインメモリーの上位100オブジェクトのリストを表示できます。

Oracle RAC環境を使用している場合は、ヒート・マップの右上にある「インスタンス」選択ボックスでインスタンスを選択することで、インスタンス間をすばやく移動できます。

A.3 表またはパーティションを作成する場合のインメモリー詳細の指定

表またはパーティションを作成するときにIM列ストアの詳細を指定できます。

1. 「スキーマ」メニューから、「データベース・オブジェクト」、「表」オプションの順に選択します。
2. 「作成」をクリックして、表を作成します。

   表の作成ページが表示されます。インメモリー列ストア・タブを選択して、表のインメモリー・オプションを指定します。

3. 列が指定されている表の下部で、列レベルのインメモリー詳細をオーバーライドすることを選択します(必要な場合)。
4. 必要に応じて、「パーティション」タブをクリックできます。
5. 必要に応じ、ウィザードを使用して表パーティションを作成します。

   パーティションにIM列ストアの詳細を指定するには、「パーティション」タブの表から選択して「拡張オプション」をクリックします。

6. 表レベル、列レベルおよびパーティション・レベルで必要なIM列ストアの詳細をすべて入力したら、「SQL表示」をクリックして、生成されたSQLを確認します。「OK」をクリックして表を作成します。

A.4 表のIM列ストアの詳細の表示または編集

表のIM列ストアの詳細を表示または編集できます。

1. 「スキーマ」メニューから、「データベース・オブジェクト」、「表」オプションの順に選択します。
2. 目的の表を検索し、「表示」をクリックして詳細を表示します。

3. 「編集」をクリックして、表の編集ページを開きます。

   別の方法としては、「検索」ページで「編集」をクリックします。「インメモリー列ストア」タブをクリックし、表に対してインメモリーのオプションを指定します。

4. 必要な詳細情報を編集します。
5. 「適用」をクリックします。

A.5 パーティションのIM列ストアの詳細の表示または編集

パーティションのIM列ストアの詳細を表示または編集できます。

1. 「スキーマ」メニューから、「データベース・オブジェクト」、「表」オプションの順に選択します。
2. 目的のパーティションが含まれる表を検索して選択し、「表示」をクリックします。
3. 「編集」をクリックして、表の編集ページを開きます。

   別の方法としては、表検索ページで「編集」をクリックします。

4. 「パーティション」タブをクリックしてから、目的のパーティションを選択します。
5. 「拡張オプション」をクリックします。
6. 必要な詳細情報を編集します。
7. 「OK」をクリックして「パーティション」タブに戻ります。
8. 表の必要なパーティションすべてに同様の変更を行い、「適用」をクリックします。

A.6 表領域作成時におけるIM列ストアの詳細の指定

表領域を作成するときにIM列ストアの詳細を指定できます。

1. 「管理」メニューから、「記憶域」、「表領域」の順に選択します。
2. 「作成」をクリックして、表領域を作成します。
3. 「一般」タブに表示する詳細を入力します。
4. インメモリー列ストアタブをクリックします。
5. 表領域のための必要なすべてのIM列ストア詳細情報を入力します。
6. 「SQL表示」をクリックします。
7. 「SQL表示」ページで、「戻る」をクリックします。
   別のページが表示されます。
8. 「OK」をクリックします。
9. 「OK」をクリックして表領域を作成します。

表領域のIM列ストアの設定は、表領域で作成されるすべての新しい表に適用されます。表で表領域の構成を上書きする必要がある場合は、個々の表レベルでIM列ストアの構成の詳細を指定する必要があります。

A.7 表領域のIM列ストアの詳細の表示および編集

表領域のIM列ストアの詳細を表示または編集できます。

1. 「管理」メニューから、「記憶域」を選択し、「表領域」オプションを選択します。
2. 目的の表領域を検索して選択し、「表示」をクリックします。
3. 「編集」をクリックして表領域の編集ページを開き、インメモリー列ストアタブをクリックします。
4. 必要な詳細を編集し、「適用」をクリックします。

A.8 マテリアライズド・ビュー作成時におけるIM列ストアの詳細の指定

マテリアライズド・ビューの作成時に、IM列ストアの詳細を指定できます。

1. 「スキーマ」メニューから、「マテリアライズド・ビュー」を選択し、「マテリアライズド・ビュー」オプションを選択します。
2. 「作成」をクリックして、マテリアライズド・ビューを作成します。
3. マテリアライズド・ビューの名前を入力し、問合せを指定します。
4. インメモリー列ストアタブをクリックして、マテリアライズド・ビューにIM列ストアのオプションを指定します。
5. 必要なIM列ストアの詳細をすべて入力したら、「SQL表示」をクリックします。「SQL表示」ページから「戻る」をクリックしてから、結果ページで「OK」をクリックします。
6. 「OK」をクリックしてマテリアライズド・ビューを作成します。

A.9 マテリアライズド・ビューのIM列ストアの詳細の表示または編集

マテリアライズド・ビューのIM列ストアの詳細を表示または編集できます。

1. 「スキーマ」メニューから、「マテリアライズド・ビュー」を選択し、「マテリアライズド・ビュー」オプションを選択します。
2. 目的のマテリアライズド・ビューを検索し、「表示」をクリックして詳細を表示します。
3. 「編集」をクリックして、マテリアライズド・ビューの編集ページを開きます。
4. インメモリー列ストアタブをクリックして、マテリアライズド・ビューにIM列ストアのオプションを指定します。
5. 必要な詳細を編集し、「適用」をクリックします。

用語集

ADOポリシー

自動データ最適化(ADO)のためのルールおよび条件を指定するポリシー。たとえば、ADOポリシーにより、オブジェクトが作成の30日後(条件)にNOINMEMORYとマークされる(アクション)よう指定できます。CREATE TABLE文およびALTER TABLE文のILM句を使用してADOポリシーを指定します。

自動データ最適化(ADO)

情報ライフサイクル管理(ILM)計画を実装するために、ポリシーを作成し、それらのポリシーに基づいてアクションを自動化するテクノロジ。

自動インメモリー

IM列ストアからコールド(アクセス頻度の低い)セグメントを自動的に削除し、作業データ・セットが常に移入されるようにする機能。

可用性

要求に応じてアプリケーション、サービスまたは機能を使用可能な程度。

ブルーム・フィルタ

セット内のメンバーシップをテストする低メモリー・データ構造。データベースでは、ハッシュ結合のパフォーマンスを高めるためにブルーム・フィルタが使用されます。

列圧縮単位(CU)

インメモリー圧縮単位(IMCU)内の列のための隣接する記憶域。

列データ・プール

列データを格納する、インメモリー領域内のサブプール。1 MBプールとも呼ばれます。

列形式

インメモリー列ストアにあるオブジェクト用の列ベースの形式。データ・ブロックで使用される、行形式と対照を成す列形式。

共通ディクショナリ

ローカル・ディクショナリから作成された、セグメント・レベルの、インスタンス固有の共通ディクショナリ・コードのセット。ローカル・ディクショナリは、列圧縮単位(CU)固有のディクショナリ・コードのソート済リストです。結合グループでは、結合を最適化するために共通ディクショナリが使用されます。

圧縮階層化

アクセス・パターンに基づく、データへの様々なレベルの圧縮の適用。たとえば、管理者は、アクセスがより低速になるという代償

を払い、より高い圧縮率で、非アクティブなデータを圧縮できます。

データ・フロー演算子(DFO)

パラレル問合せのデータ再分散ステージ間の作業単位。

稠密グループ化キー

グループ化列が特定のファクト表またはディメンションから取得されるすべてのグループ化キーを表すキー。

稠密結合キー

結合列が特定のファクト表またはディメンションから取得されるすべての結合キーを表すキー。

稠密キー

ネイティブ整数として格納され、値の範囲を持つ数値キー。

ダブル・バッファリング

バックグラウンド・プロセスで、変更した最新の行を元の行と組み合せることで新しいインメモリー圧縮単位(IMCU)バージョンを作成する再移入メカニズム。再移入中は、古いIMCUを引き続き問合せに使用できます。

式

1つ以上の値、演算子および値を評価するSQL機能の組合せ。

式の取得間隔

データベースが、取得可能なIM式を検討する時間間隔。

式取得ウィンドウ

`DBMS_INMEMORY_ADMIN`パッケージの`IME_OPEN_CAPTURE_WINDOW`および`IME_OPEN_CAPTURE_WINDOW`プロシージャの呼出しによって定義される式の取得間隔。

式統計ストア(ESS)

式評価に関する統計を格納する、オプティマイザによって保持されるリポジトリ。セグメントごとに、ESSによって、実行頻度、評価コスト、タイムスタンプ評価などの統計が監視されます。ESSは本来、永続的で、式の高速参照のためのSGA表現を持っています。

FastStart領域

データベースがインメモリー列データを定期的に書き込む指定表領域。

ヒート・マップ

ヒート・マップでは、データ・ブロックおよび行の需要が示されます。異なるストレージ層に移動する候補となるセグメントを決定する、自動データ最適化(ADO)。

ホーム・ロケーション

IMCUが存在するデータベース・インスタンス。Oracle RACで自動DOPが有効になっている場合、パラレル問合せコーディネータでは、ホーム・ロケーションを使用して、各IMCUが存在する場所、その大きさなどが判断されます。

ハイブリッド・パーティション表

パーティションの一部がデータ・ファイル・セグメントに格納されたり、外部データ・ソースに格納される表。

IM集計

単一の大きい表から複数の小さい表に結合する複数の問合せに対して集計を促進する最適化。変換では、KEY VECTORおよびVECTOR GROUP BY演算子が使用されます。これが、VECTOR GROUP BY集計とも呼ばれる理由です。

IM列ストア

迅速スキャン用に最適化された表とパーティションのコピーを、列形式で格納するオプションのSGA領域。

IM動的スキャン

インメモリー表スキャンを自動的にパラレル化する軽量スレッドの使用。

IM式

インメモリー列ストアに結果が格納されるSQL式。last_nameが、IM列ストアに格納される列である場合は、IM式がUPPER(last_name)になる場合があります。

IMCUミラー化

Oracle RACでは、複数のIM列ストアでのIMCUの複製。たとえば、インスタンス1およびインスタンス2上のIM列ストアが、同じsales表を使用して移入されます。

IMCUプルーニング

インメモリー列ストアの問合せにおける、各IMCUの高い値および低い値に基づくIMCUの排除。たとえば、100を上回る製品IDが文でフィルタリングされる場合、100を下回る値が含まれるIMCUのスキャン処理がデータベースで回避されます。

IM記憶域索引

IMCU内のすべての列の最小値および最大値を格納する、IMCUヘッダー内のデータ構造。

インメモリー・アドバイザ

データベース内の分析処理ワークロードを分析する、ダウンロード可能なPL/SQLパッケージ。このアドバイザは、インメモリー移入による利益を得られる、IM列ストアのサイズ、およびオブジェクトのリストを推薦します。

インメモリー集計

IM集計を参照してください。

インメモリー領域

IM列ストアが含まれる、オプションのSGAコンポーネント。

インメモリー列ストア

IM列ストアを参照してください。

インメモリー圧縮単位(IMCU)

高速スキャン用に最適化されたインメモリー列ストア内の記憶域単位。インメモリー列ストアでは、各列が表内に別々に格納され、圧縮されます。各IMCUでは、ある行サブセットのすべての列が特定の表セグメント内に含まれます。

IMCUとデータベース・ブロックのセットの間には1対多マッピングが存在します。たとえば、表に列c1およびc2が含まれ、その行がディスク上の100個のデータベース・ブロックに格納されている場合、IMCU 1にはブロック1から50までの両方の列の値が格納され、IMCU 2にはブロック51から100までの両方の列の値が格納されます。

インメモリー・コーディネータ・プロセス(IMCO)

主要タスクが列データのバックグラウンド移入および再移入の開始である、バックグラウンド・プロセス。

In-Memory動的スキャン

IM動的スキャンを参照してください。

インメモリー式

IM式を参照してください。

インメモリー式単位(IMEU)

インメモリー式(IM式)の計算結果を格納するコンテナ。各IMEUは、それぞれに独自の親のインメモリー圧縮単位(IMCU)にリンクされています。

インメモリー・ファスト・スタート

データベース・インスタンス再起動時のIM列ストアへのデータ移入時間を大幅に削減する機能。

インメモリー移入

移入を参照してください。

インメモリー仮想列

インメモリー列ストアに移入される候補となる仮想列。

情報ライフサイクル管理(ILM)

データをその有用期間中ずっと管理するためのプロセスおよびポリシーのセット。

結合グループ

同じ表または異なる表から頻繁に結合された列を指定するユーザー定義オブジェクト。外部表はサポートされていません。

一般的な結合グループ候補は、ファクト表およびディメンション表を結合するために使用される列のセットです。結合グループは、INMEMORY_SIZEがゼロ以外の値の場合のみサポートされます。

キー・ベクター

稠密結合キーと稠密グループ化キー間をマップするデータ構造。

ローカル・ディクショナリ

列圧縮単位(CU)固有のディクショナリ・コードのソート済リスト。

軽量スレッド

In-Memory動的スキャンで使用される実行エンティティ。軽量スレッドは、IMCUのスキャンをパラレル化するために役立ちます。

メタデータ・プール

IM列ストア内に存在するオブジェクトについてのメタデータを格納する、インメモリー領域のサブプール。メタデータ・プールは、64 KBプールとも呼ばれます。

オンデマンド移入

INMEMORY PRIORITYがNONEに設定されている場合、IM列ストアでは、全体スキャンを通じてアクセスされるときのみオブジェクトが移入されます。オブジェクトは、アクセスされることがないか、索引スキャンまたはROWIDによるフェッチを通じてしかアクセスされない場合、移入されることはありません。

OSON

Oracleの最適化されたバイナリJSON形式。OSONを使用すると、Oracleデータベース・サーバーおよびデータベース・クライアントでのJSONデータ・モデルの高速な問合せおよび更新が可能になります。

OZIP

非常に高速な解凍を提供する、独自の圧縮技術。OZIPは、Oracle Database向けに特別に調整されています。

パーティション交換ロード

表を作成し、データをそれにロードしてから、既存の表パーティションをその表と交換する技術。この交換プロセスは、実際のデータの移動を伴わないDDL操作です。

移入

データ・ファイルから既存のデータ・ブロックを読み取り、行を列形式に変換してから、その列データをIM列ストアに書き込む操作。一方、ロードとは、DMLまたはDDLを使用して新しいデータをデータベースに移すことを指します。

優先度ベース移入

PRIORITYがNONE以外の値に設定されている場合、Oracle Databaseでは、オブジェクトは優先順位付けされた移入キューに追加されます。データベースでは、CRITICALからLOWまで、それらのキュー位置に基づいてオブジェクトが移入されます。優先順位ベースと呼ばれる理由は、IM列ストアではデータベースが再度開かれるときは必ず、優先順位付けられたリストを使用して自動的にオブジェクトが移入されるためです。オンデマンド移入と異なり、オブジェクトは、移入に全体スキャンを必要としません。

再移入

現在移入されているインメモリー圧縮単位(IMCU)の自動リフレッシュ。そのデータが大幅に変更された後に行われます。一方、移入は、IM列ストアでの最初のIMCU作成です。

サービス

同じ属性、パフォーマンスしきい値および優先順位を共有する、アプリケーション・ワークロードの論理表現。単一のサービスをOracle RACデータベースの1つ以上のインスタンスに関連付けると、単一のインスタンスで複数のサービスをサポートできます。

スナップショット・メタデータ単位(SMU)

関連するインメモリー圧縮単位(IMCU)のメタデータおよびトランザクション情報を含む、インメモリー領域内の記憶域単位。

領域管理ワーカー・プロセス(Wnnn)

インメモリー・コーディネータ・プロセス(IMCO)にかわってIM列ストアにデータを移入または再移入するプロセス。

失効しきい値

再移入を開始する、IMCUのトランザクション・ジャーナル内のエントリについて内部的に設定されたパーセンテージ。

ストレージ階層化

アクセス・レベルに応じた、ストレージの様々な層でのデータのデプロイメント。たとえば、管理者は、非アクティブなデータを高パフォーマンスで高コストのストレージから低コストのストレージに移行します。

表スキャン・プロセス

IM動的スキャンを調整するフォアグラウンドまたはPQプロセス。

しきい値ベース再移入

IMCU内の失効エントリの数が内部失効しきい値に達したときのIMCUの自動再移入。

トランザクション・ジャーナル

IM列ストアのトランザクションの一貫性を保つ、スナップショット・メタデータ単位(SMU)内のメタデータ。

トリクル再移入

しきい値ベース再移入の補完機能。インメモリー・コーディネータ・プロセス(IMCO)では、失効エントリが存在するが失効しきい

値に満たないIM列ストア内の任意のIMCUに対して、トリクル再移入を自動的に引き起こす場合があります。

ベクター集計

IM集計を参照してください。

仮想列

ディスク上に格納されない列。データベースは、必要に応じて、一連の式またはファンクションを計算することによって仮想列の値を導出します。

作業データ・セット

指定時間にアクティブに問合せされているINMEMORYオブジェクトのサブセット。通常、作業の作業データ・セットは時間の経過とともに変化します。

索引

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

V

W