
ORACLE®

Oracle Database 12c Release 2 CoreTech Seminar

12.2.0.1
Database In-Memory

日本オラクル株式会社
クラウド・テクノロジー事業統括
Database & Exadataプロダクトマネジメント本部
井上 克己
2016/10

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

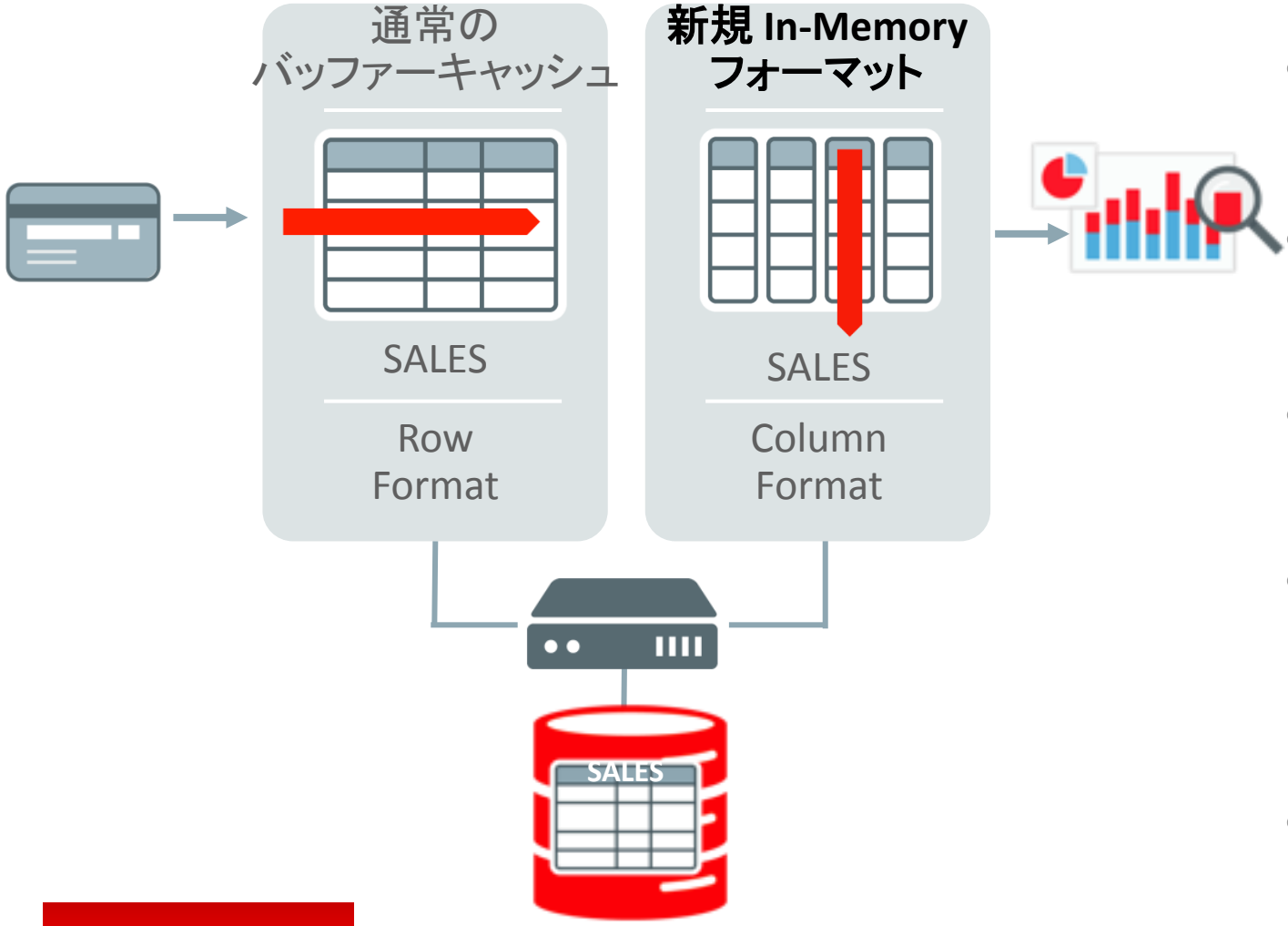
Agenda

1. 12c R1 機能
2. 12c R2 新機能概要
3. 構成の多彩化
4. オプティマイザー/実行計画
5. クエリー高速化

Agenda

1. 12c R1 機能
2. 12c R2 新機能概要
3. 構成の多彩化
4. オプティマイザー/実行計画
5. クエリー高速化

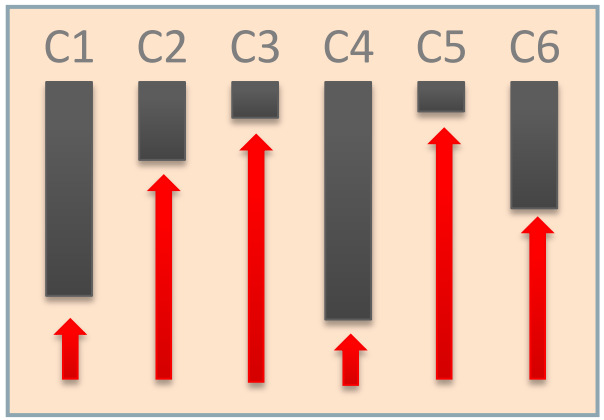
What is Oracle Database In-Memory?



- Breakthrough: デュアル フォーマット データベース
- 一つの表につき 行 および 列 フォーマットの**両方**
- 同時にアクティブ、および、トランザクション一貫性
- 分析& レポーティングは新しい in-memory 列フォーマットを使う
- OLTPは実績のある行フォーマットを使用する

カラム型表は何故分析用クエリーが高速か?

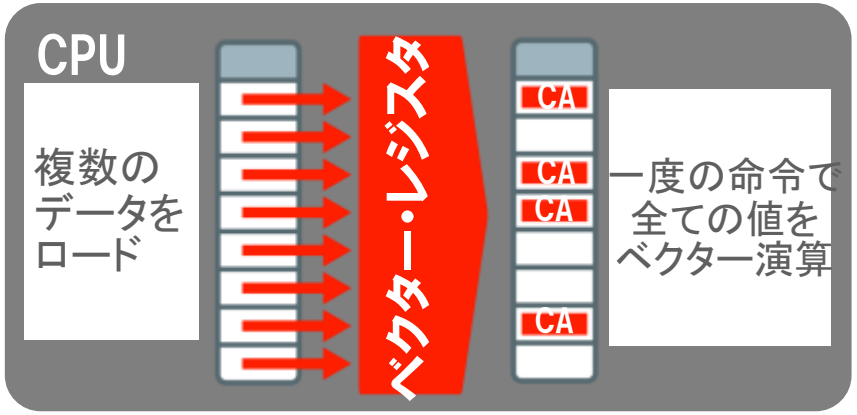
ポイント1:
集計に必要なカラムのみアクセス＋効果的な圧縮技術により圧縮した状態で検索が可能 (ディクショナリ圧縮)



ポイント2:
インメモリ・ストレージ索引により最小限のIMCUのみスキャン

例) where storeid > 8

ポイント3:
最新のプロセッサで搭載されているSIMDにより高速フィルター



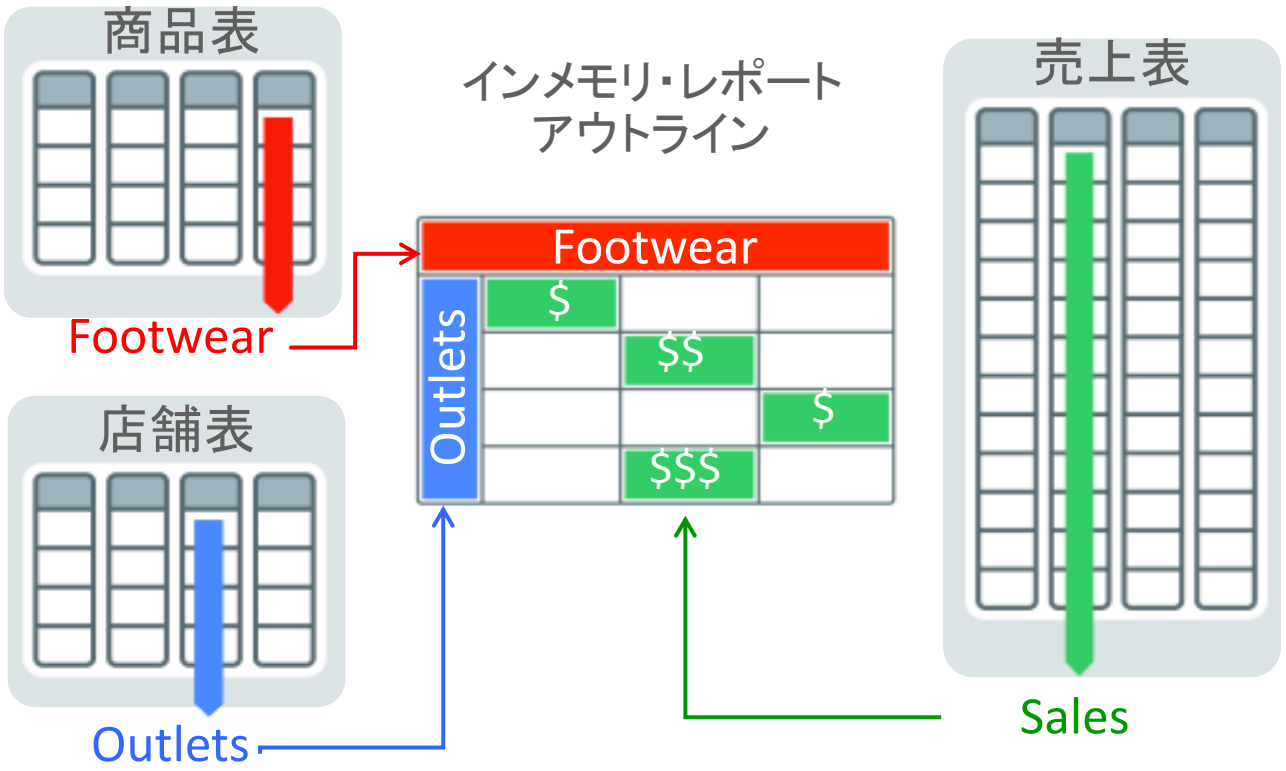
ポイント4:
パラレル・クエリーとパーティション表によりさらに高速化可能

インメモリ検索による表の集計処理の高速化

ベクター・Group By(Vector Group By)

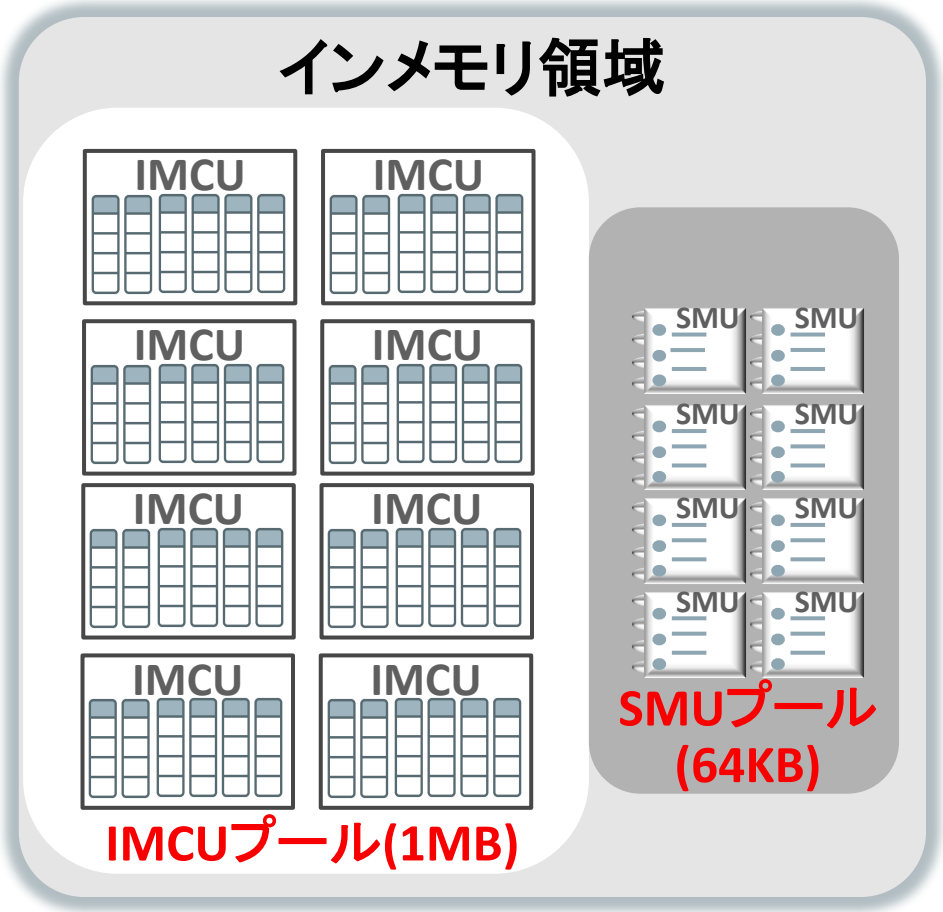
インメモリ固有の機能ではないがインメモリ検索で非常に効果的

例: アウトレットでの靴の売上を集計



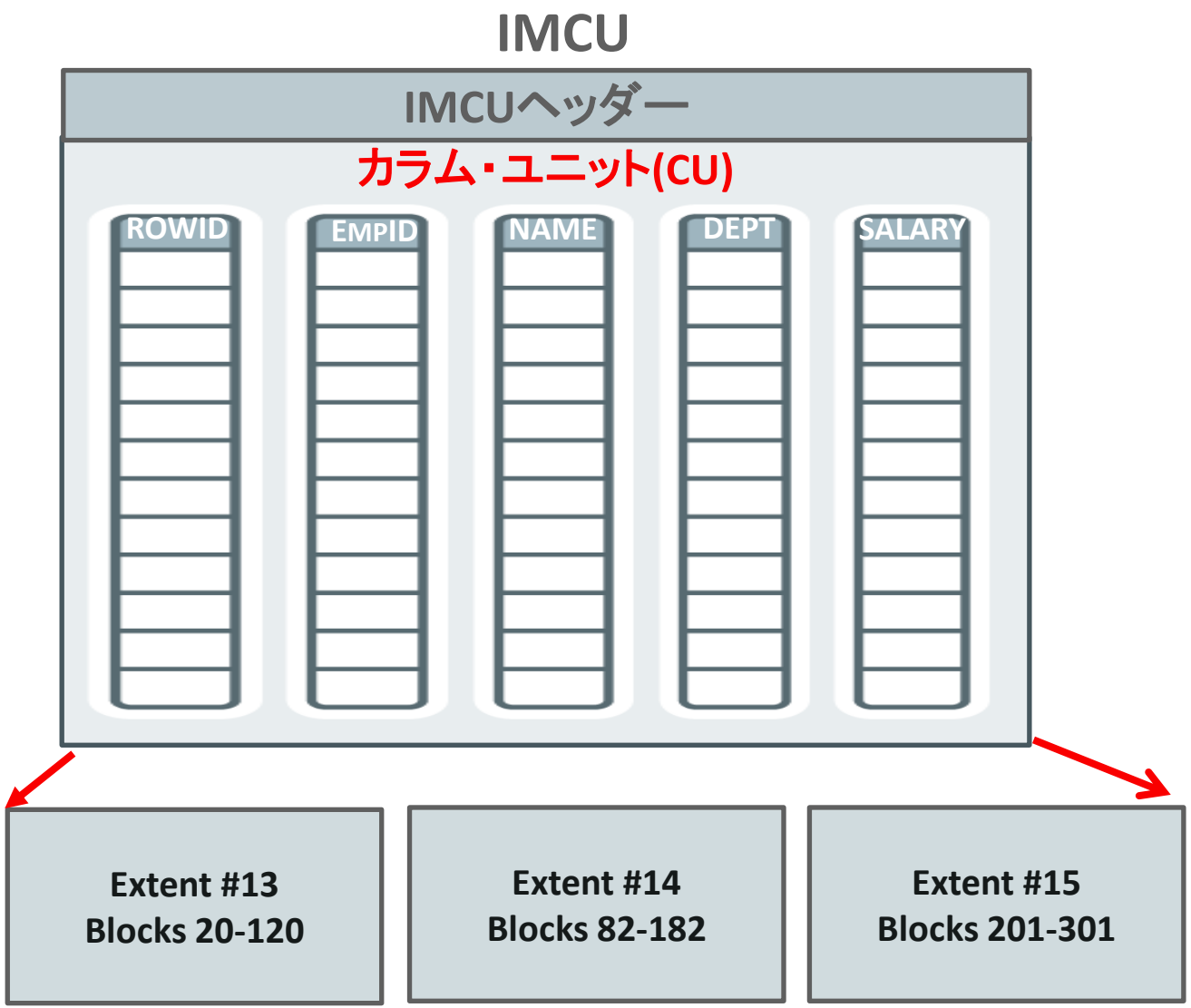
- レポート・アウトラインをメモリー上に動的に作成(インメモリ配列)
- レポート内の集計値はファクト表のスキャン中に展開
- 事前定義された多次元キューブを使わずに高速化

インメモリ領域: 構成



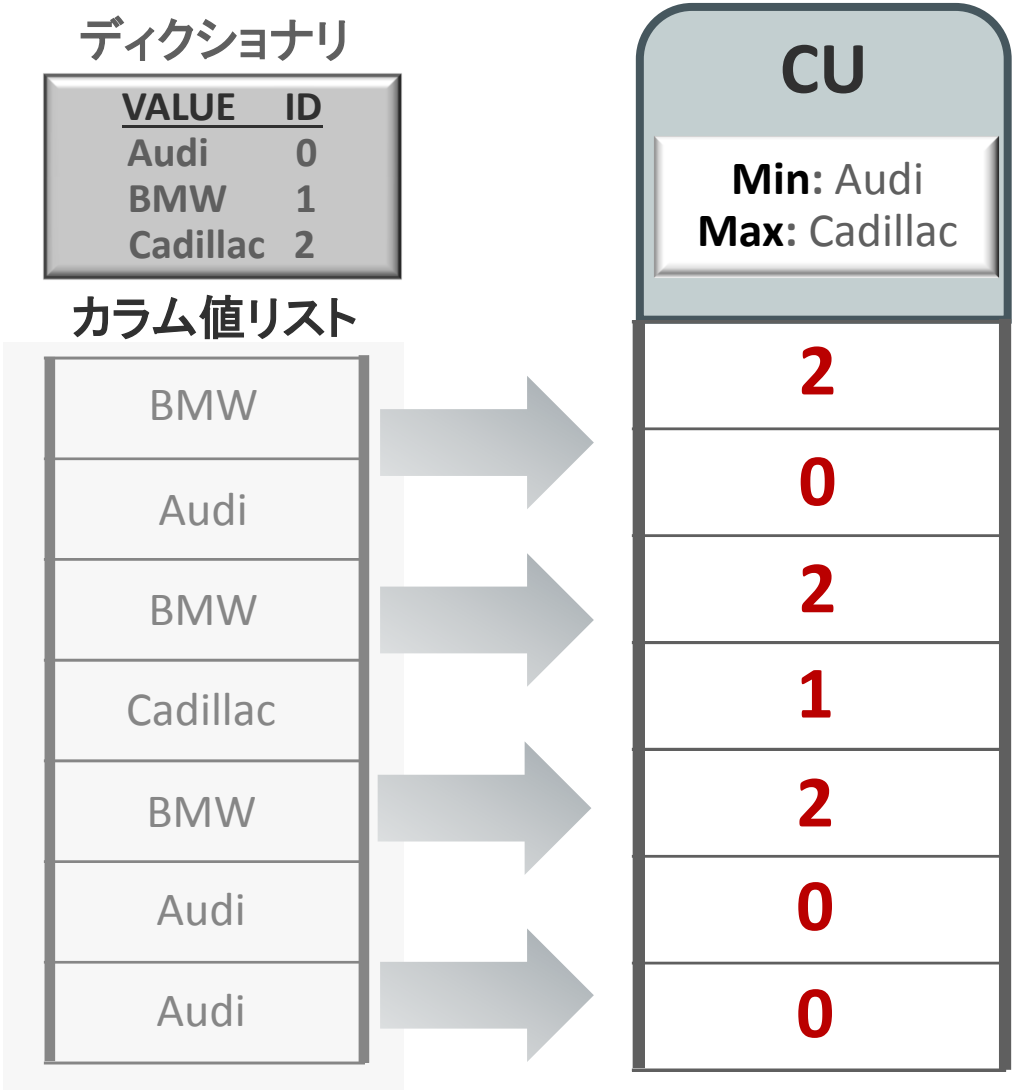
- 2つのサブプールで構成:
  - IMCU(1MB)プール:
    IMCU(In-Memory Compression Units)を格納
  - SMU(64KB)プール:
    SMU(Snapshot Metadata Units)を格納
- IMCUはカラム書式のデータを格納
- SMUはメタデータとトランザクション情報を格納

IMCU : In-Memory Compression Unit



- カラム型オブジェクトの管理単位
  - カラム型データをある程度の行数のセットで保持(例:50万行程度)
  - 格納される行は1つ以上の表エクステントから取得
- IMCUの実サイズは行サイズ、圧縮率等により変化(固定値ではない)
- カラム毎に分離/近接したカラム・ユニット(CU)として保存
  - Rowidも同様に1つのCUとして保存

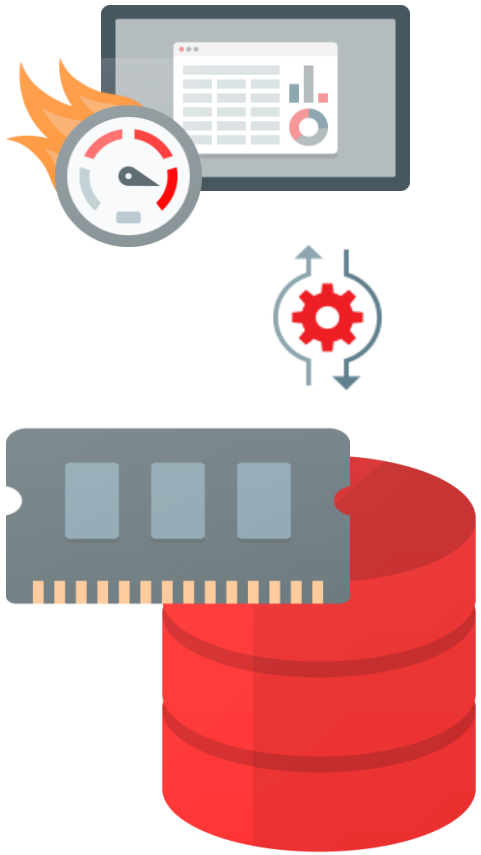
CU：カラムユニット (Column Unit)



- IMCUに格納されている各カラム値の管理単位
- 全CUは自動的に最小／最大値を保存
  - インメモリ・ストレージ索引
- 圧縮フォーマット
  - 例）ディクショナリ圧縮
    CUは実際の値ではなく、サイズの小さいディクショナリIDをデータ値として格納
    → 圧縮した状態で検索が可能
  - 他の圧縮方式と組合わせることも可能

12.1.0.2 Bundle Patch

DMLパフォーマンス改良
SPARC M7 対応

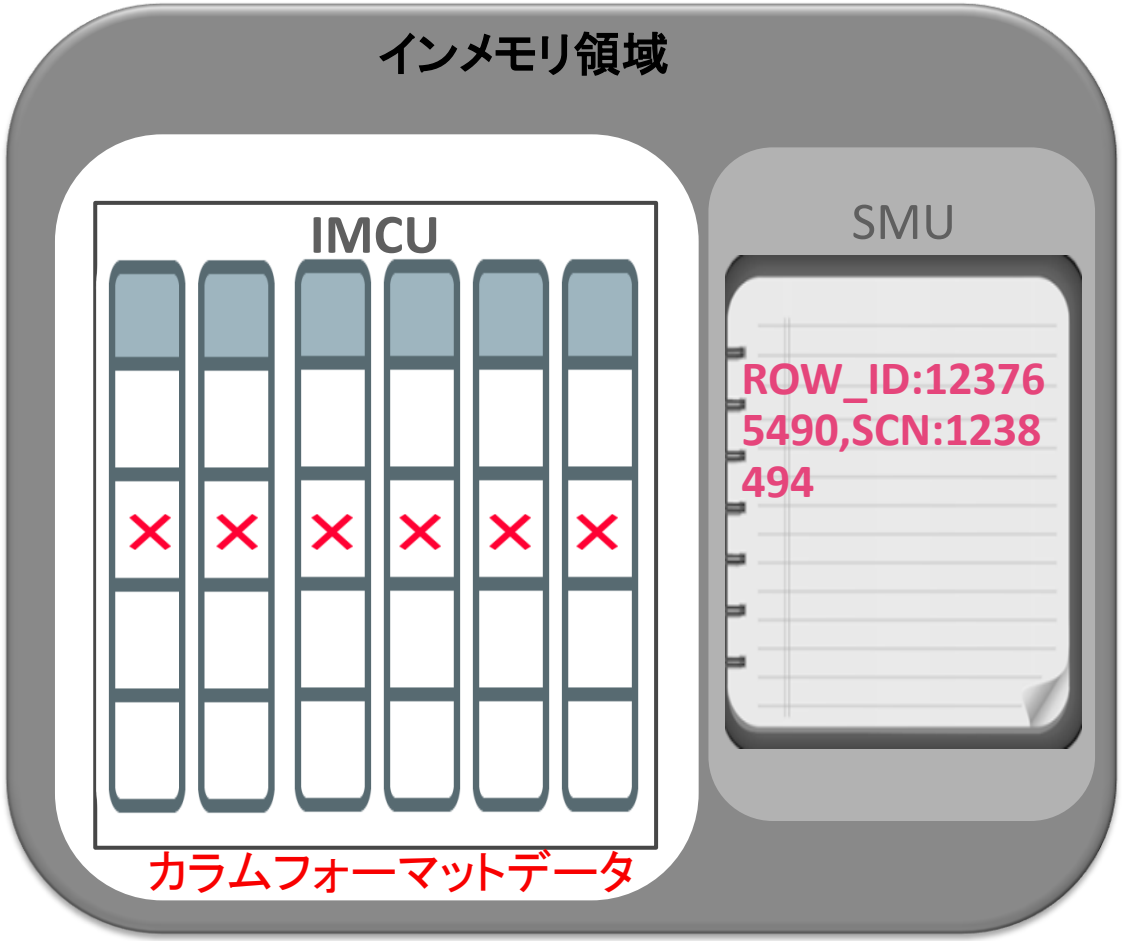


12.1.0.2 Bundle Patchによるパフォーマンス向上も

- MOS Note:1454618.1
  Quick Reference to Patch Numbers for PSU, SPU(CPU), BPs and Patchsets
- 2014年7月以降90個程度のIn-Memory関連のバグが修正
- 新機能の実装も
- パフォーマンス向上
  - DML後のIMCUメンテナンス
    3倍高速
  - 混合ワークロード(Mixed Workload)2倍程度高速
  - ハードパース時間短縮

12.1.0.2				
Description	**PSU**	**GI PSU**	**Proactive Bundle Patch**	**Bundle Patch** (Windows 32bit & 64bit)
APR2016	22291127 (12.1.0.2.160419)	22646084 (12.1.0.2.160419)	22899531	22809813 (12.1.0.2.160419)
JAN2016	21948354 (12.1.0.2.160119)	22191349 (12.1.0.2.160119)	22243551	22310559 (12.1.0.2.160119)
OCT2015	21359755 (12.1.0.2.5)	21523234 (12.1.0.2.5)	21744410 (12.1.0.2.13)	21821214 (12.1.0.2.10)
JUL2015	20831110 (12.1.0.2.4)	20996835 (12.1.0.2.4)	21188742 (12.1.0.2.10)	21126814 (12.1.0.2.7)
APR2015	20299023 (12.1.0.2.3)	20485724 (12.1.0.2.3)	20698050 (12.1.0.2.7)	20684004 (12.1.0.2.4)
JAN2015	19769480 (12.1.0.2.2)	19954978 (12.1.0.2.2)	20141343 (12.1.0.2.4)	19720843 (12.1.0.2.1)
OCT2014	19303936 (12.1.0.2.1)	19392646 (12.1.0.2.1)	19404326 (12.1.0.2.1)	N/A

DMLとインメモリ・カラムストアのメンテナンス(repopulate)



- DML操作は現状通りロー(行)・ストアで処理される
- カラムストア内の該当エントリがそのSCN時点で「stale」とマークされる
- トランザクションジャーナル内に該当行のROWIDが保存される
- あるタイミングでIMCU再作成
- 12.1.0.2 GA 時点はディスクから全行読み取り
- BPで差分による更新に変更

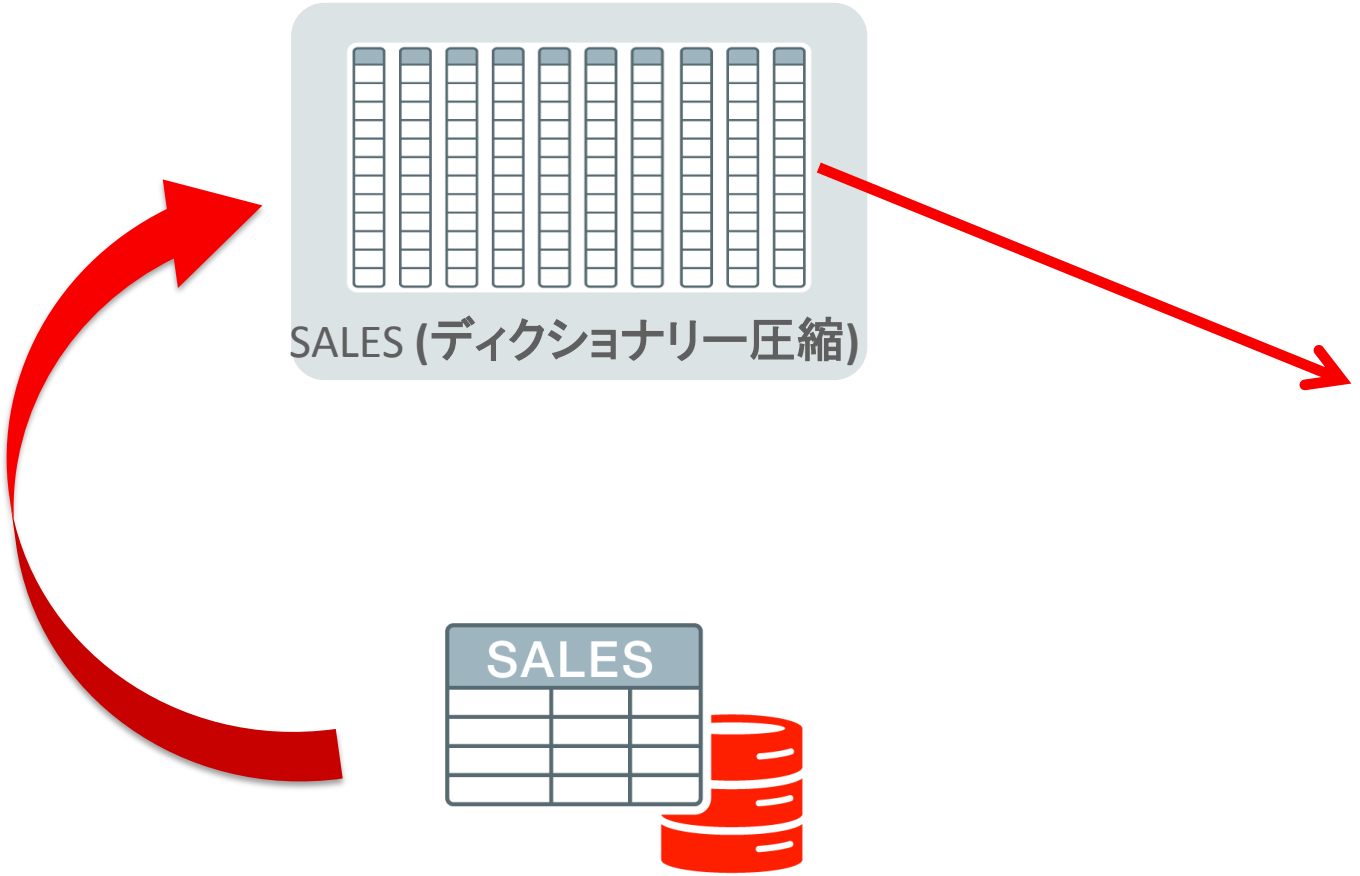
12.1 Bundle PatchでのSPARC M7 Software in Silicon対応

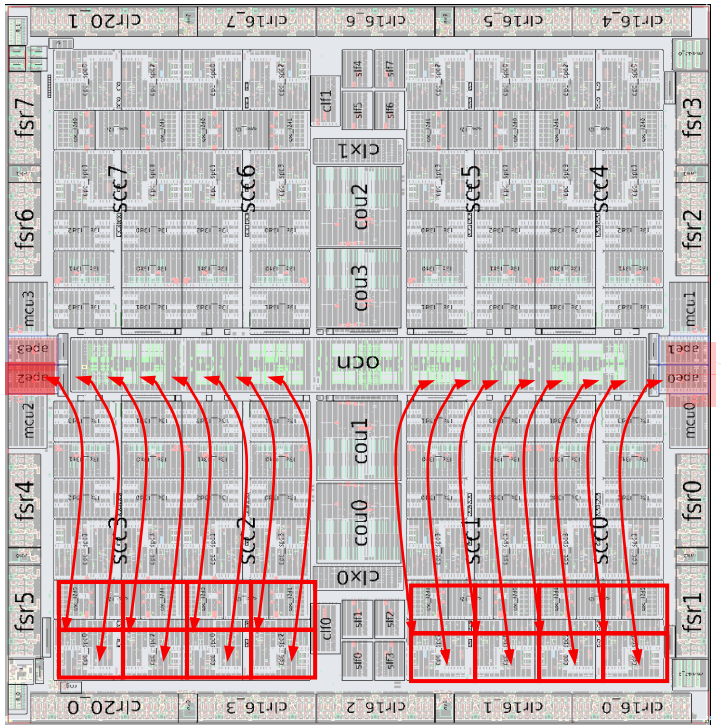
Oracle Database In-Memoryの内部処理をCPUチップ上に実装

- 従来の行ベースのデータベースのアルゴリズムをCPUチップ上に実装することは複雑で困難
- Oracle Database In-Memoryによりインメモリ・カラム型フォーマットに対応
  → より多くの内部処理をシンプル化
  → CPUチップ上に実装可能
- 5年前からオラクルはこの革新的なプロジェクトを開始
  - 従来型プロセッサで最速プロセッサの構築
    - 32 cores, 16 DDR4 チャネル, 160 GB/sec 測定帯域
  - **インメモリ・データベース** 処理を直接CPUチップ上で実行
- SQL本来の処理に最適化された最高クラスのCPU
  - ムーアの法則で拡張し続ける高速データ処理性能

ソフトウェア・イン・シリコンによるインメモリーデータ処理

DAX scan processing in DAX frees most of the cores for transactional processing





限界テスト時スループット最大計測時のAWR比較

- 多重検索処理最大スループットを達成した際の2つのAWRレポートを比較
- CPU使用率に差
  - X4-2

Host CPU

CPUs	Cores	Sockets	Load Average Begin	Load Average End	%User	%System	%WIO	%Idle
40	20	2	0.02	14.59	95.2	0.3	0.0	4.5

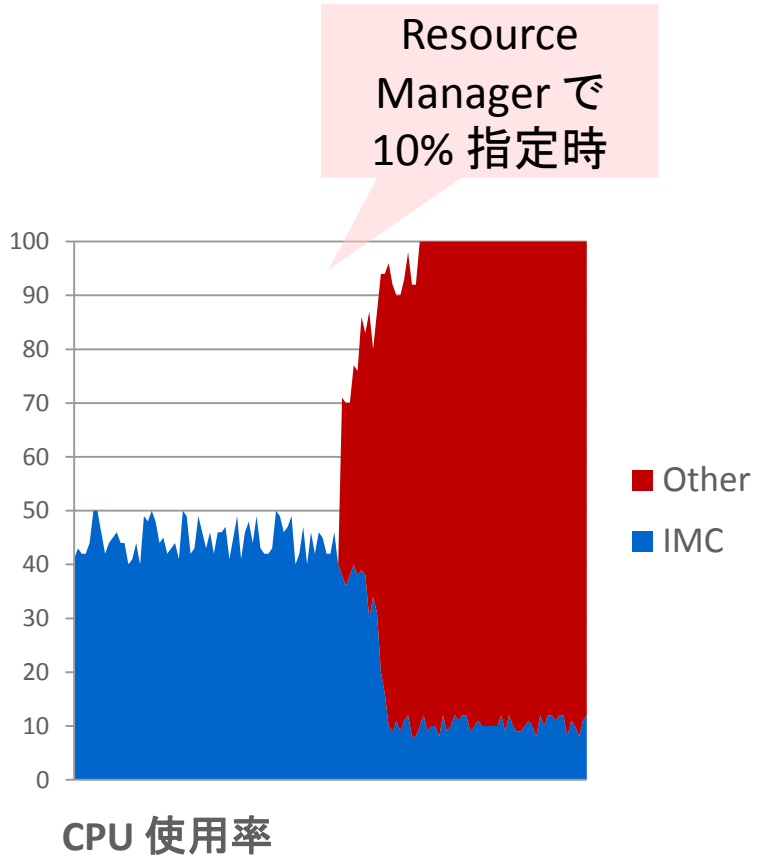
  - M7

Host CPU

CPUs	Cores	Sockets	Load Average Begin	Load Average End	%User	%System	%WIO	%Idle
256	32	1	0.05	15.86	58.1	0.1	0.0	41.8

DB In-Memory: 12.1.0.2 -> 12.2.0.1 でもほぼ同等な機能

- 圧縮レベル指定方法と種類
- それぞれのレベルでの圧縮率
- 初回ポピュレーション速度
  - FastStart 不使用時
  - 再ポピュレーションについては高速化
    - 12.1 Bundle Patchで多くが既に実装されリリース済み
- マルチ・テナント、プラガブルDBでの設定、動作
- Resource Manager で可能な設定
  - リソース使用量の上限設定
- インメモリ領域のプールの割合



In-Memory基本設定

- 12c R1 から変更無

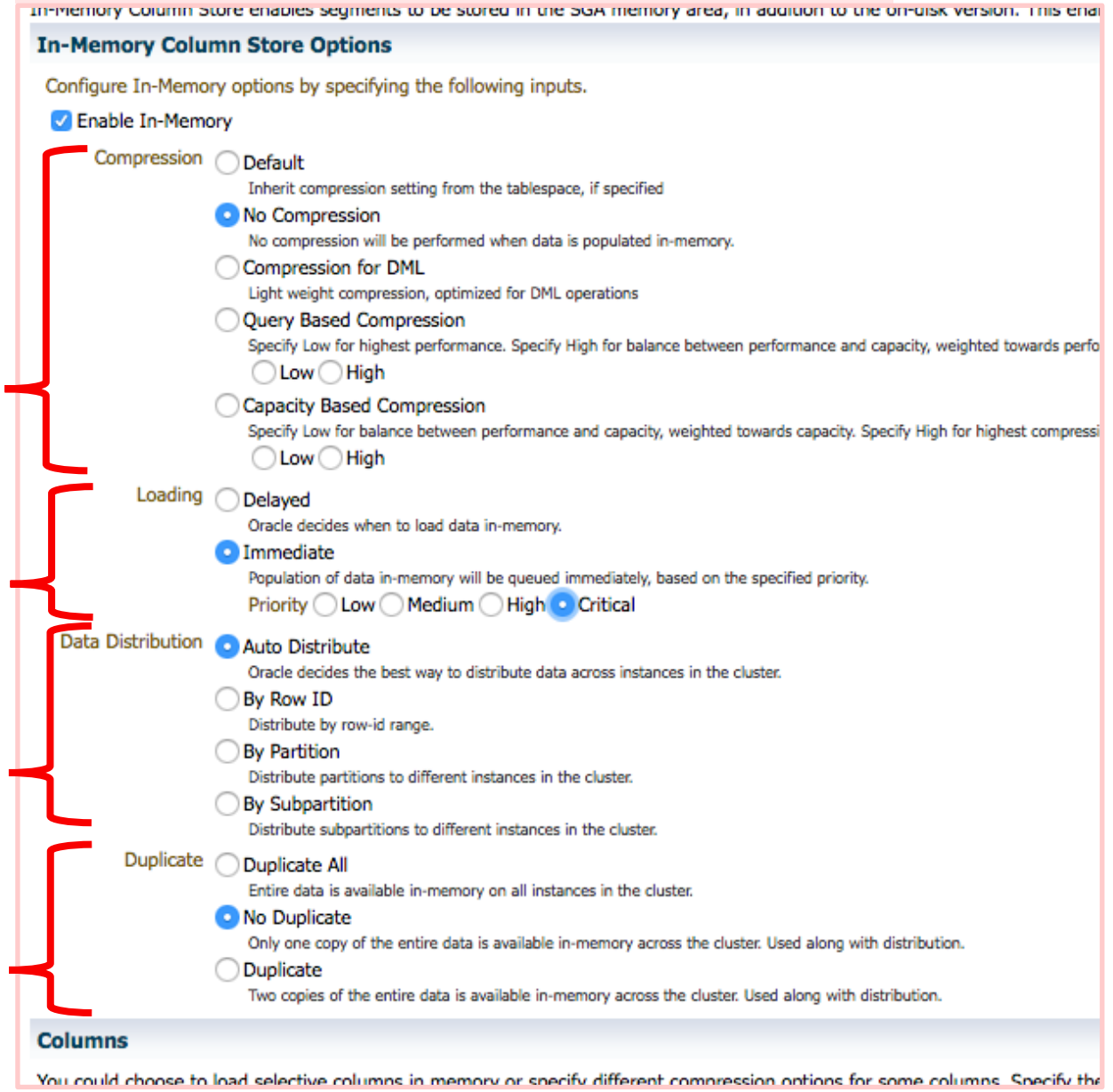
6段階の圧縮レベル

5段階のプライオリティ

RACノード間の分散方法

RACノード間のHA

Enterprise Manager 13c Cloud Control画面

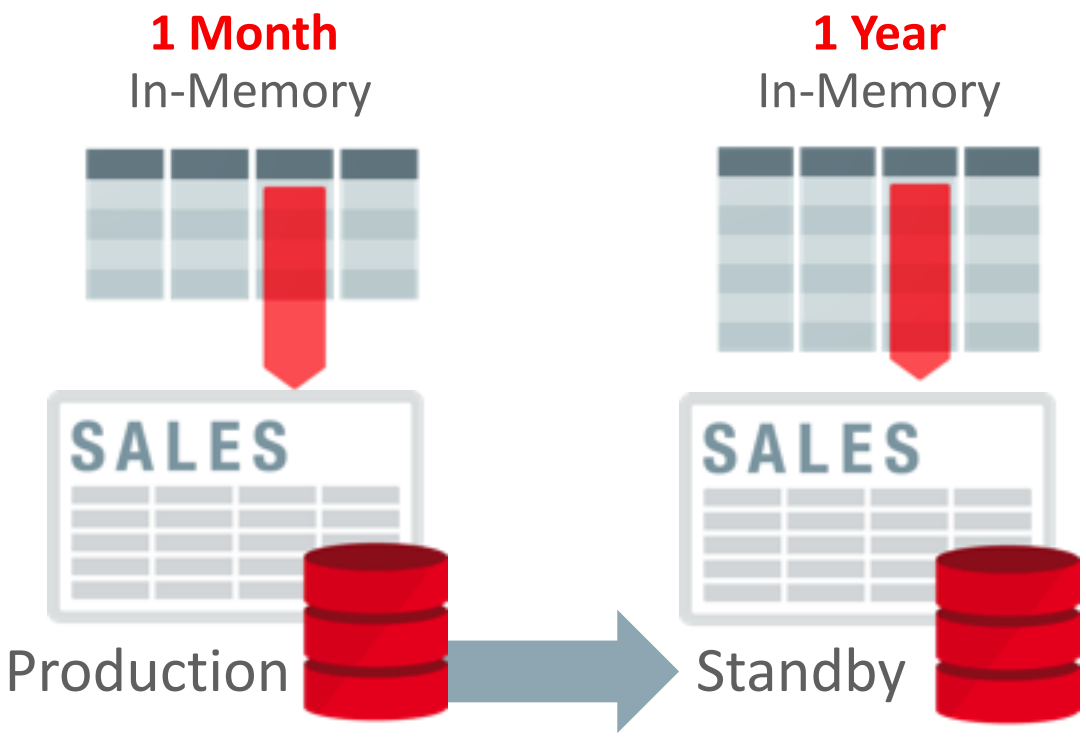


Agenda

1. 12c R1 機能
2. 12c R2 新機能概要
3. 構成の多彩化
4. オプティマイザー/実行計画
5. クエリー高速化

混合ワークロード: Active Data GuardでのIn-Memory

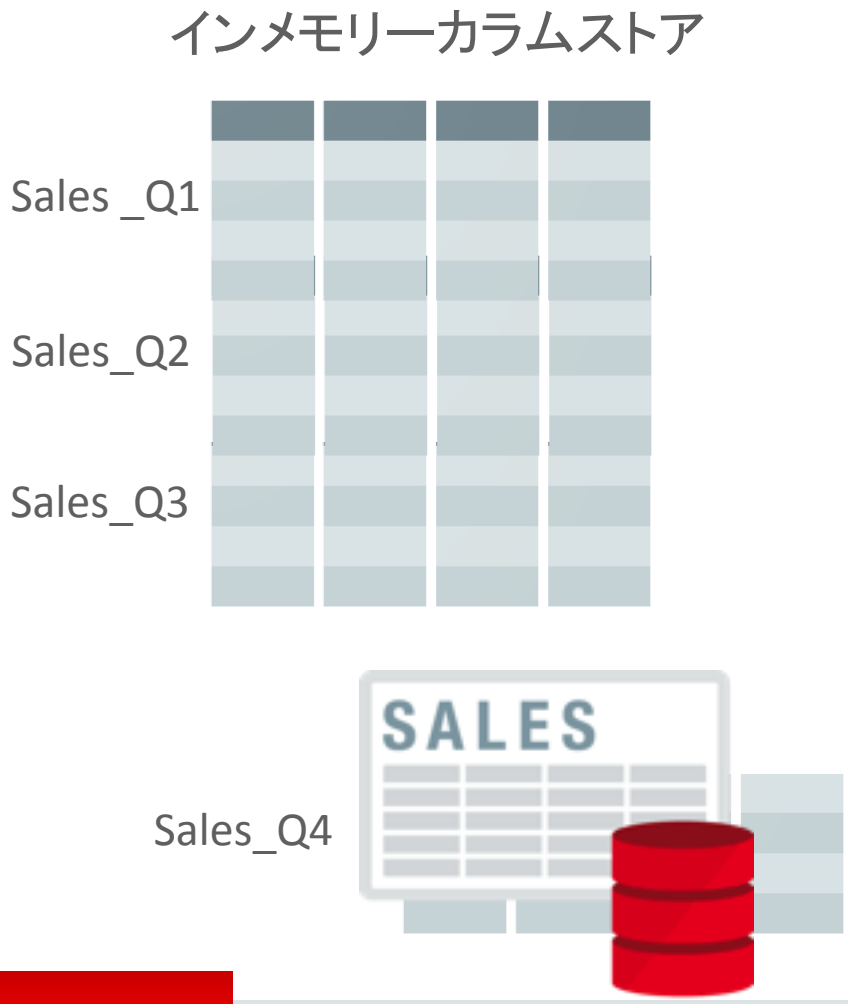
NEW IN 12.2



- プロダクションDBに何ら影響を与えることなくリアルタイム分析
- スタンバイデータベースのリソースを活用
- プロダクションDBとは異なるデータをポピュレーション可能
  - 新規の "DISTRIBUTE BY SERVICE" 句によりどちらに表をポピュレートするか指定する
  - カラムナー領域のトータルなサイズは増える

自動化: In-Memory データ自動ポピュレーションポリシー

NEW IN 12.2




```
ALTER TABLE SALES ILM ADD POLICY
NO INMEMORY AFTER 9 months OF CREATION
```

- ADOのインメモリーへの拡張
- Heat map はデータアクセス頻度をトラックする
- ポリシーにより以下を定義可能:
  - IMカラムストアへデータをポピュレーション
  - データがクールダウンすると圧縮レベルを上げる
  - IMカラムストアからデータを排出(evict)

リアルタイム分析: In-Memory Expressions

NEW IN 12.2

例: 売り上げの総額を計算

Net = Price + Price * Tax

インメモリーカラムストア

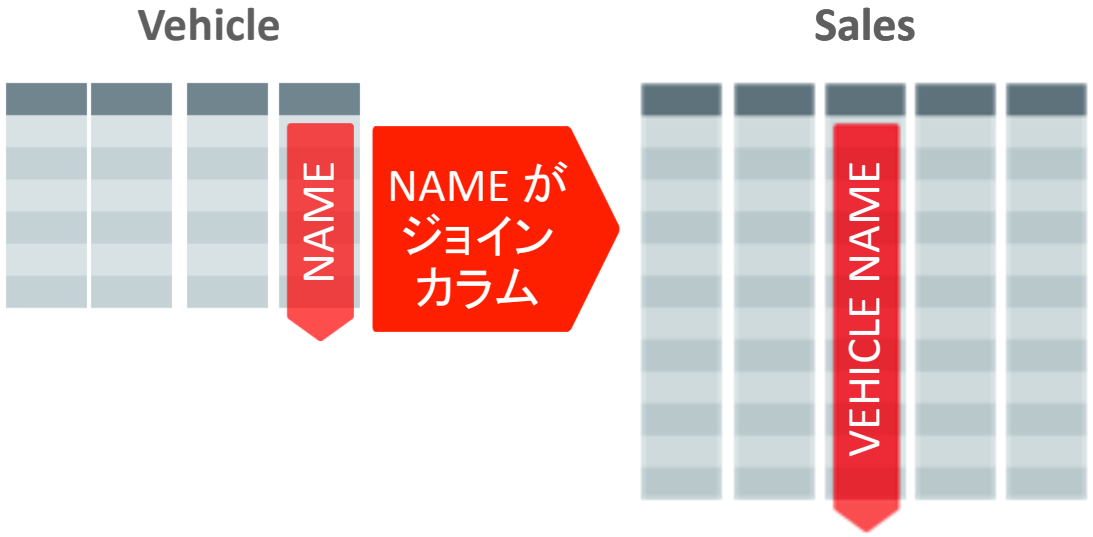

```
CREATE TABLE SALES (
       PRICE NUMBER, TAX NUMBER, …,
       NET AS (PRICE + PRICE * TAX)
       ) INMEMORY;
```

- 分析的なクエリーが複雑な計算式(expression)を含んでいる
  - 通常は各行につき計算される
- Expression(表現、計算式)は事前に計算されin-memoryに格納可能に
  - 仮想カラムでユーザが明示的に指定する
  - または expression が自動的に検出される
- 全ての In-Memory に最適化された処理が適用可能(SIMDなど)
- 複雑なクエリー: **3-5倍高速に**

リアルタイム分析: より高速な In-Memory ジョイン

NEW IN 12.2

例: 各車種の販売価格を調べる



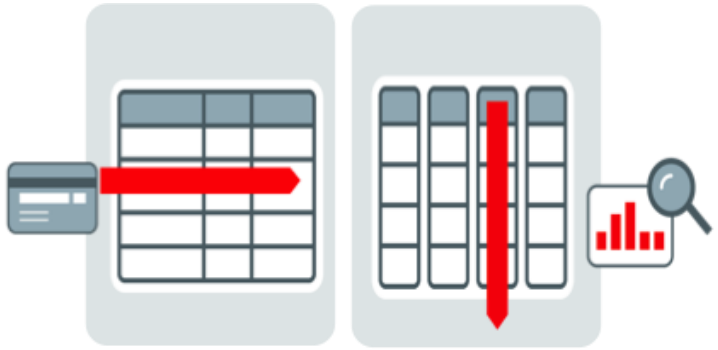

```
CREATE INMEMORY JOIN GROUP V_name_jg
(VEHICLES(NAME),SALES(NAME));
```

- 分析的クエリーにJOINがあり、フィルター述語が無い場合
- "ジョイン・グループ" でテーブルをJOINするカラムを指定
  - それらのカラムは圧縮ディクショナリーを共有する
  - 実際のカラム値データそのものではなくディクショナリー 値によりJOINされる
- **2-3倍高速な** join 処理が可能に

Oracle Database 12*c* In-Memory

12.1 & 12.2新機能概要

Real Time Analytics



12.1 OLTP & Analytics on same single server or RAC database

12.2 Offload analytics to Active Data Guard standby

Performance

Sub-second reporting & analytics

SQL in Silicon

3x faster joins, 10x faster expressions, 60x faster JSON

実装が簡単

アプリケーションの変更不要

Dynamic data movement between storage & memory

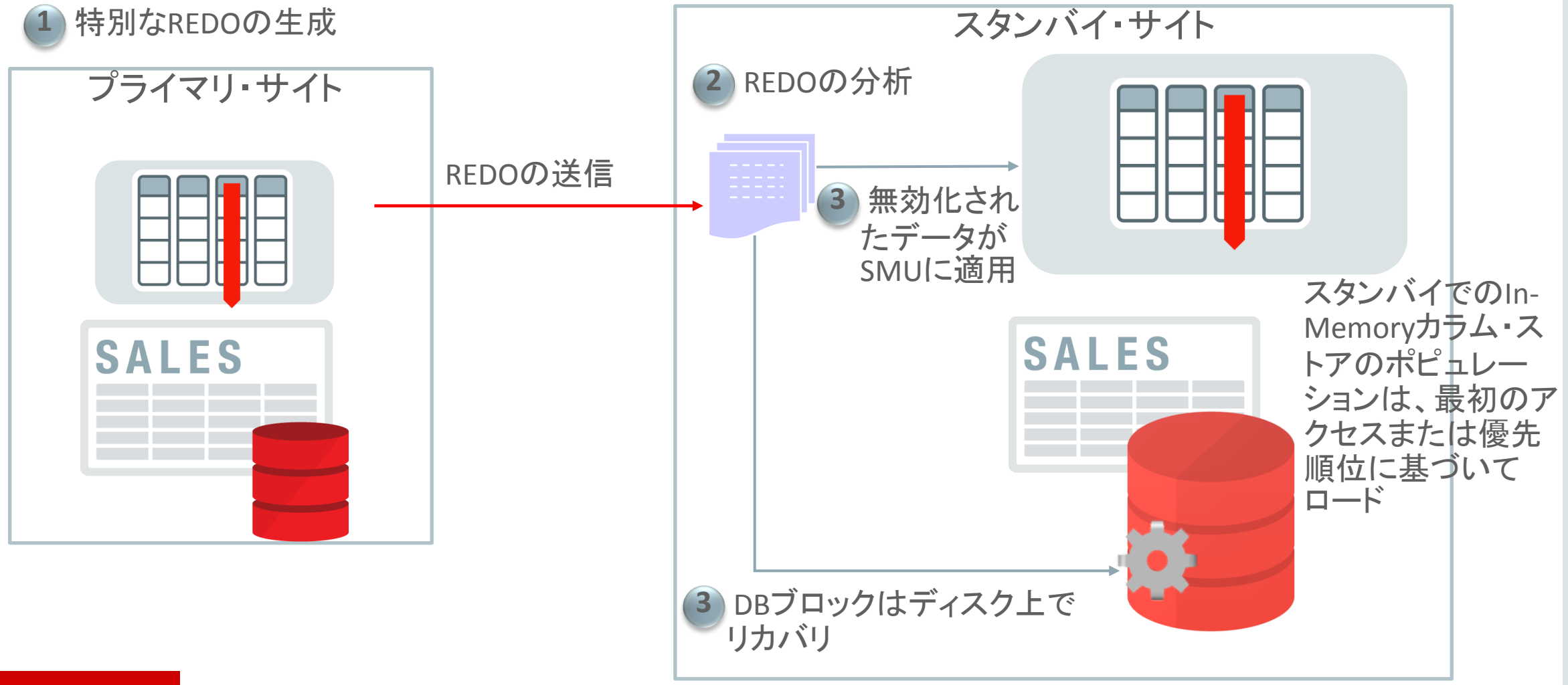
Agenda

1. 12c R1 機能
2. 12c R2 新機能概要
3. **構成の多彩化**
4. オプティマイザー/実行計画
5. クエリー高速化

Active Data Guard

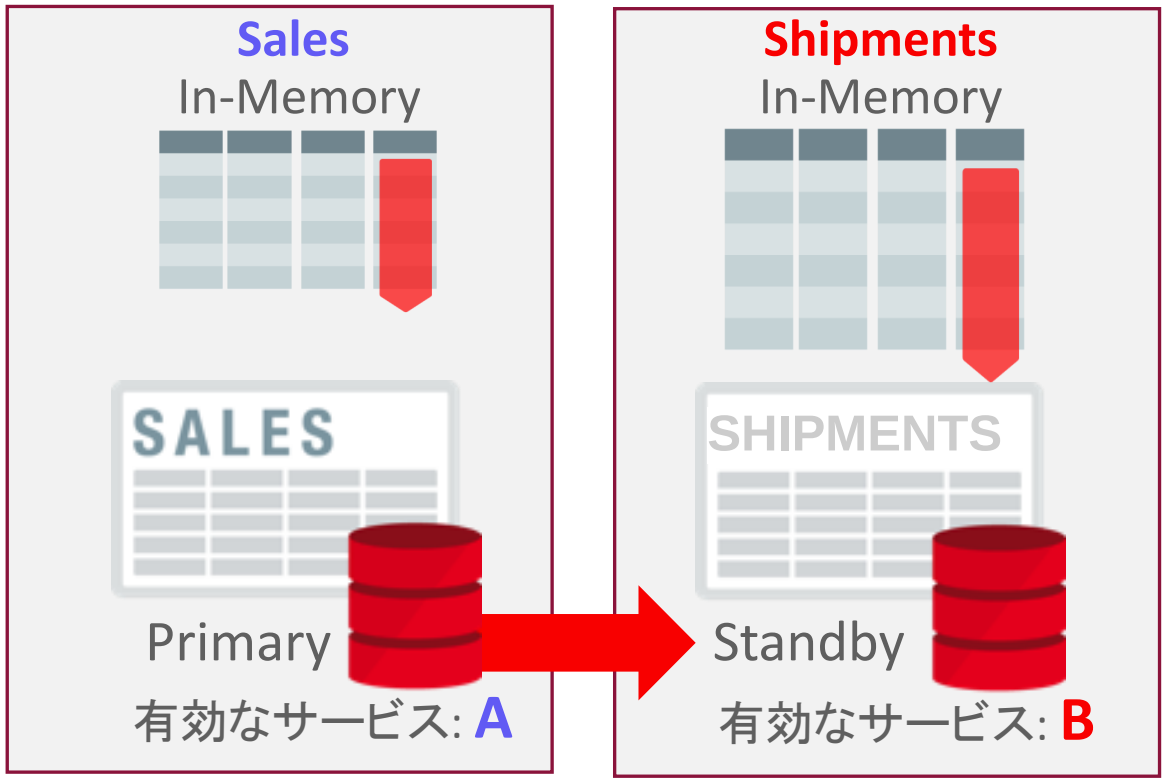
スタンバイでの分析ワークロードが可能に

Active Data GuardでのDBIM： High-Level Architecture



Capacity + Availability: In-Memory on Active Data Guard

NEW IN 12.2




```
ALTER TABLE
SALES INMEMORY
DISTRIBUTE
FOR SERVICE A
```

```
ALTER TABLE
SHIPMENTS INMEMORY
DISTRIBUTE
FOR SERVICE B
```

- **DISTRIBUTE FOR SERVICE** 句によりプライマリ、スタンバイ、両方でメモリーにポピュレーションするかどうかを指定
- スタンバイが複数ある場合、各々で異なる表をインメモリー化可能
- ロールベースサービス使用可能

srvctl add **service** -db db_unique_name -service **サービス名** [-eval]
-serverpool server_pool [-cardinality {UNIFORM | SINGLETON}]
[-edition edition_name] [-netnum network_number]
[**-role** "[**PRIMARY**][,**PHYSICAL_STANDBY**][,LOGICAL_STANDBY][,...]"

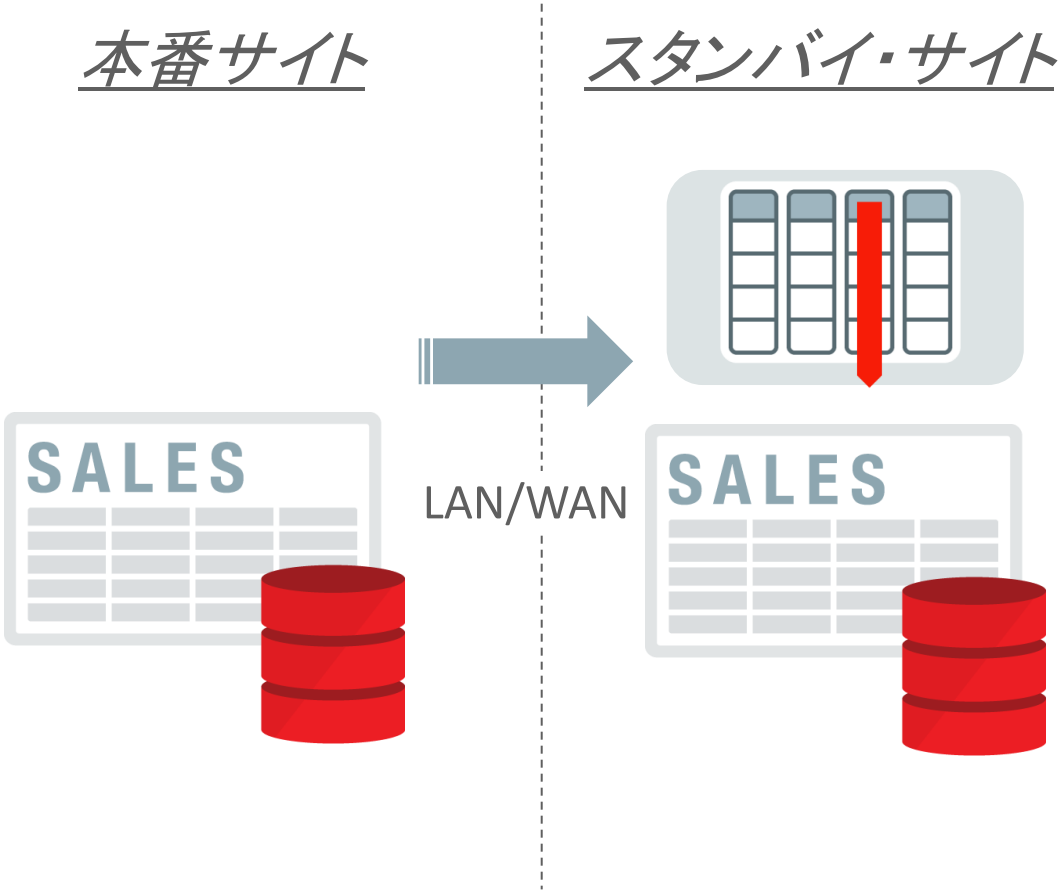
- 非Grid Infrastructure環境では dbms_service.start_service() 使用

考慮点

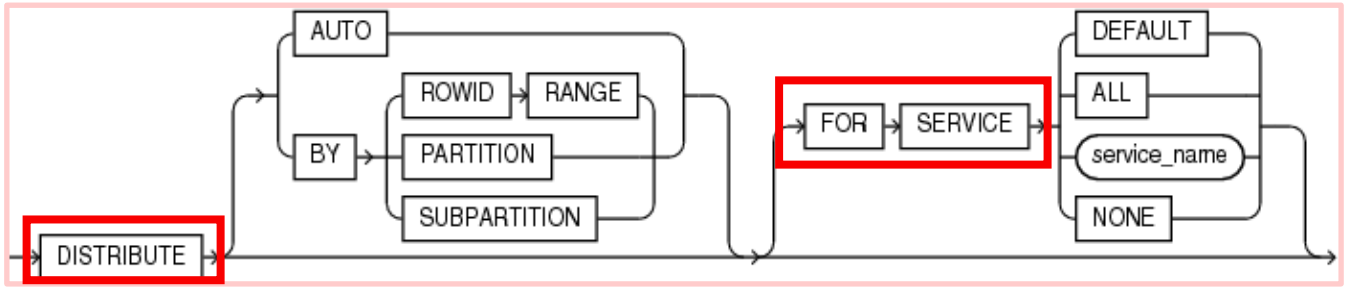
NEW IN 12.2

- オペレーション的な制限
  - In-Memory Expressionはプライマリーのみでキャプチャー可能
  - ADO: In-Memory Information Lifecycle Management (ILM)ポリシーはプライマリーのみでアクセス回数などがカウントされ、かつ、アクションが実行される
- より基本的な使用上の制限
  - In-Memory FastStart および In-Memory Join-Group はActive Data Guardのスタンバイ側では動作せず、それらの設定が存在しない場合と同じ動作になる
  - Multi Instance Redo Apply(複数インスタンスREDO適用,MIRA) 未サポート
  - `DISTRIBUTE FOR SERVICE` 句を使用して表の一部をプライマリー、残りをスタンバイにポピュレーションする設定は不可能
    - プライマリーまたはスタンバイがRAC構成の場合はRACノード間でこのような設定は可能

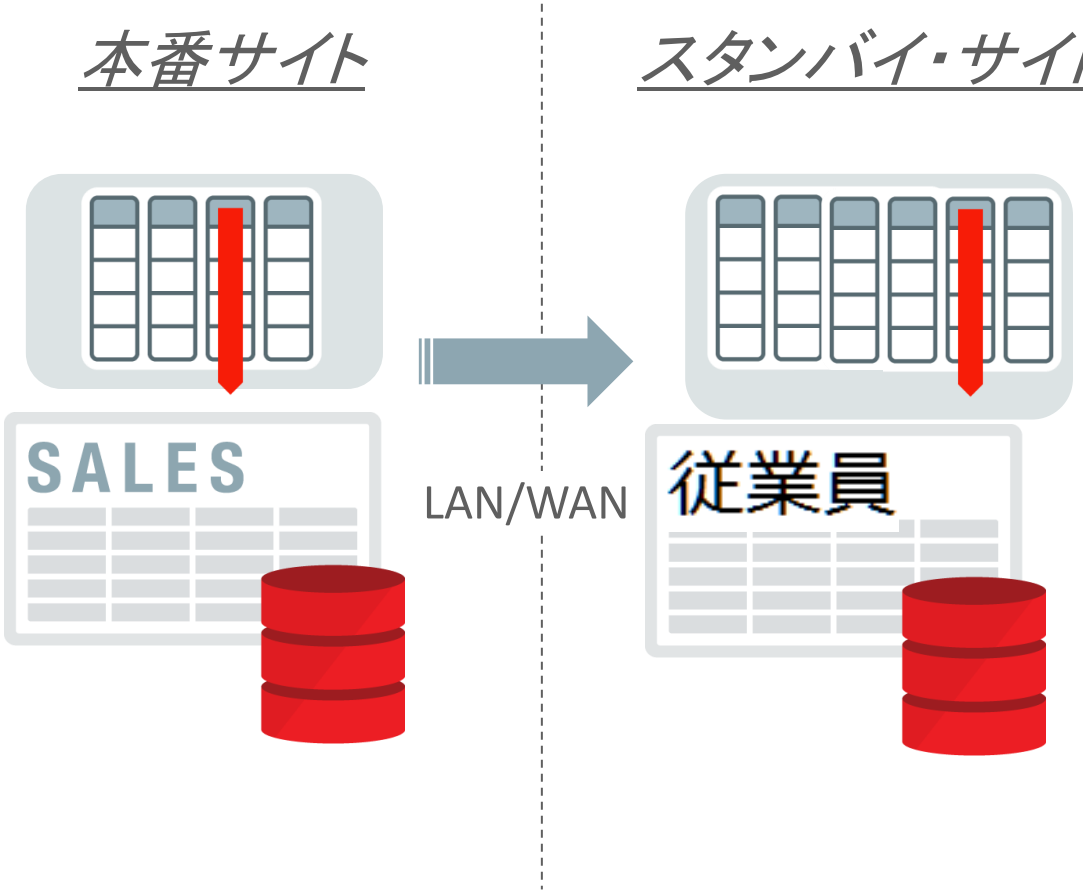
In-Memory on Active Data Guard:スタンバイのみでIn-Memory



- INMEMORY_SIZE パラメータ
  - スタンバイ・サイトのみで設定
- 表にDISTRIBUTE FOR SERVICE句なしでINMEMORYを設定
- 分析問合せ
  - セカンダリ・サイトに接続 & In-Memoryデータにアクセス



IM on Active Data Guard: サイト間で異なるIn-Memory データ



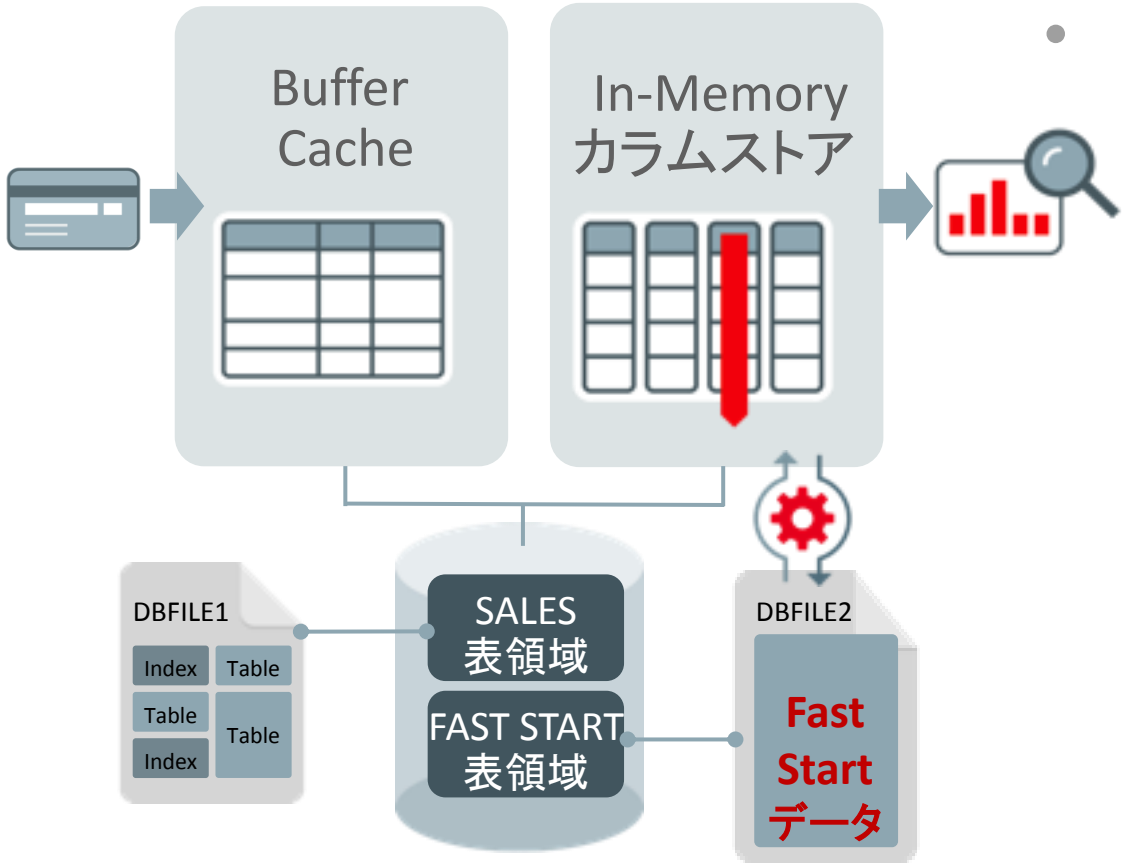
- INMEMORY_SIZE パラメータ
  - プライマリ&スタンバイ・サイトの両方で設定
  - 異なる値も可能
- 全てのオブジェクトに以下に設定
  INMEMORY DISTRIBUTE FOR SERVICE
  - プライマリにサービスを設定
    または
  - スタンバイにサービスを設定
  - 両方にサービスを設定
- 分析問合せ
  - 適切なサービスを指定して接続

FastStart

ポピュレーションの高速化

ポピュレーション速度の向上: In-Memory Fast-Start

NEW IN 12.2



- IM 列フォーマットのストレージへの永続化
  - In-Memory カラムストアの内容がポピュレーション時にSecureFile LOBへチェックポイントされる
  - DB 再起動時のポピュレーションは列フォーマットを直接ストレージから読み出してくるのでより高速
  - データの再フォーマットの必要がないためカラムストアのリストアはより高速 (2-5倍)

In-Memory FastStart

NEW IN 12.2

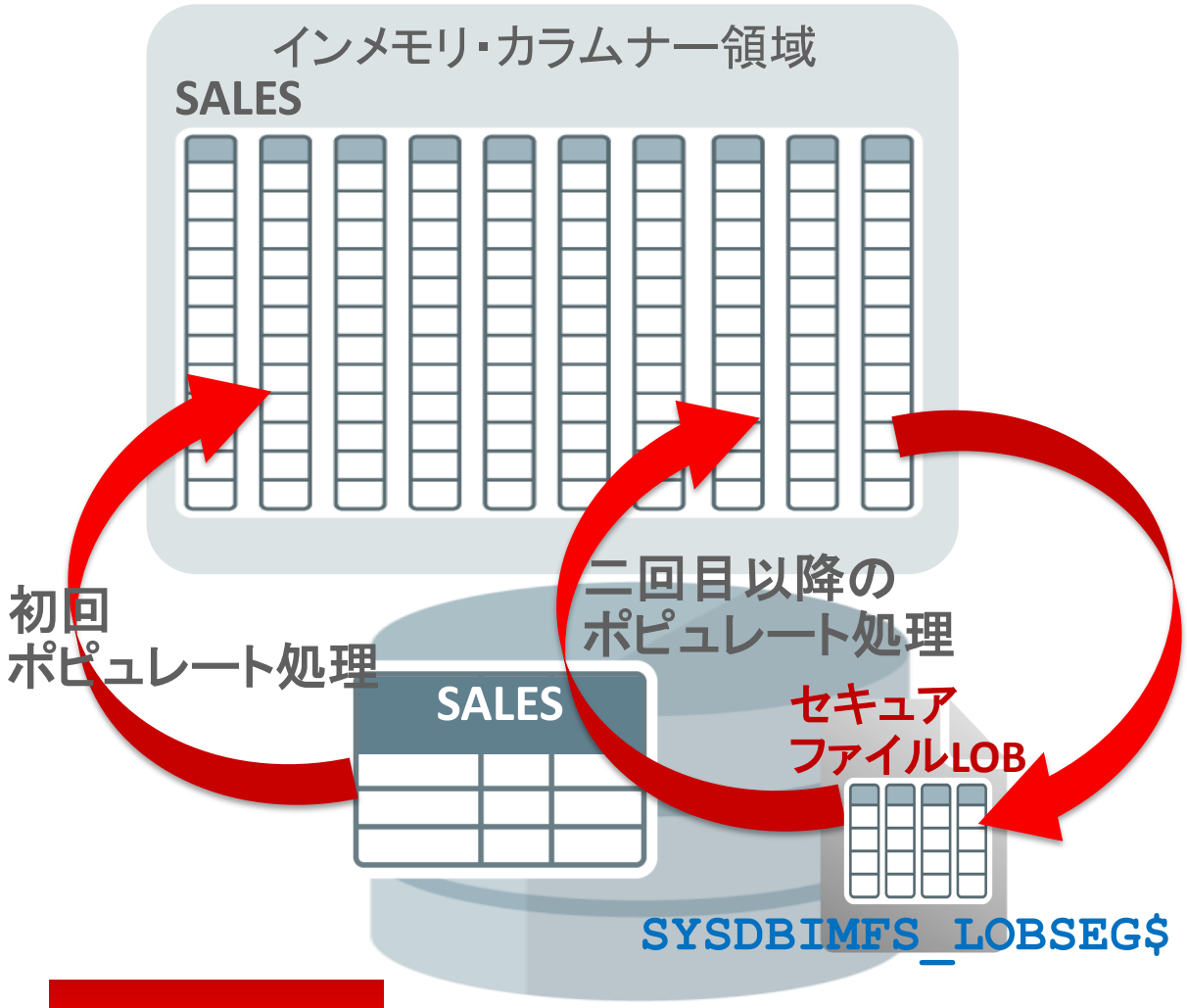
```
BEGIN
   dbms_inmemory_admin.Enable_faststart('TS_FASTSTART');
END;
```

```
SQL>
SQL> Select      l.owner, l.segment_Name, sum(s.bytes)/1024/1024 MB
  2  From        dba_lobs l, dba_segments s
  3  Where       l.segment_name = s.segment_name
  4  And         l.tablespace_name = 'TS_FASTSTART'
  5  Group by l.owner, l.segment_Name;

OWNER      SEGMENT_NAME            MB
---------- --------------- ----------
SYS        DBIMFS_LOBSEG$        .125
```

1. FastStart用表領域の割り当て
   - 推奨は表領域のサイズは INMEMORY_SIZEパラメーターの 2倍
     - データ更新に備えるため。一時的にインメモリIMCU合計値以上の領域が必要になる
2. データはLOB SYSDBIMFS_LOBSEG$に保存
   - メタデータはSYSAUX表領域に保持
   - SYSAUX 表領域はオンラインであること
3. 基表のいずれかが暗号化表領域にある場合FS用表領域も暗号化が必要

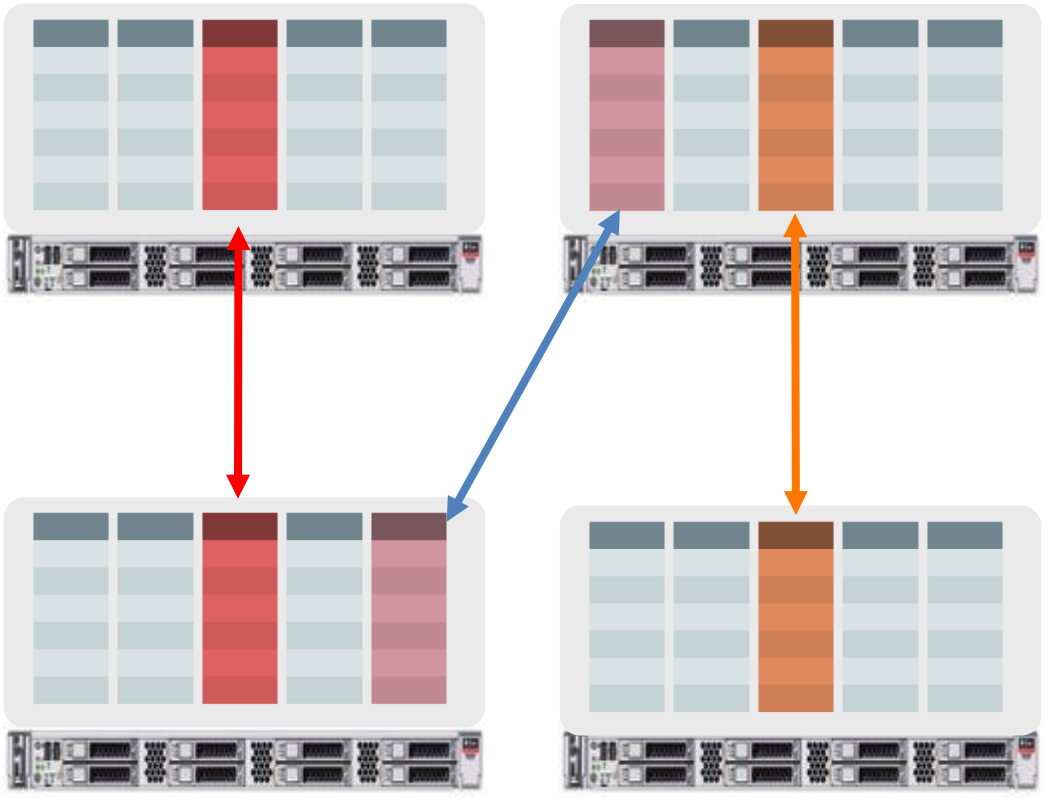
In-Memory FastStart

NEW IN 12.2



- ポピュレート処理の都度にデータはFastStart領域に書き出される
  - 手動で書き出す方法は提供されない
- データベース再起動時はFastStart領域から直接ポピュレートする
  - フォーマットと圧縮が不要
  - 標準ポピュレーション・ルールを適用
- オブジェクトのインメモリ属性を外すと、FastStart領域から削除される
- 作成後の表領域の移動も可能
  - DBMS_INMEMORY_ADMIN.FASTSTART_MIGRATE_STORAGE
- In-Memory Expression,Join Groupは未サポート
- LOBデータはプラットフォーム固有

RACなどでのFastStart機能詳細



Engineered Systemsでのみ設定可能な
インメモリー領域の High Availability

- 左の絵のようなExadata、ODAのDUPLICATE時には共通のFaststartイメージが使用される
- Active Data Guardは非対応
  - IMCUデータのLOBへの書き出しはNOLOGGING
  - Faststartデータは伝播されない

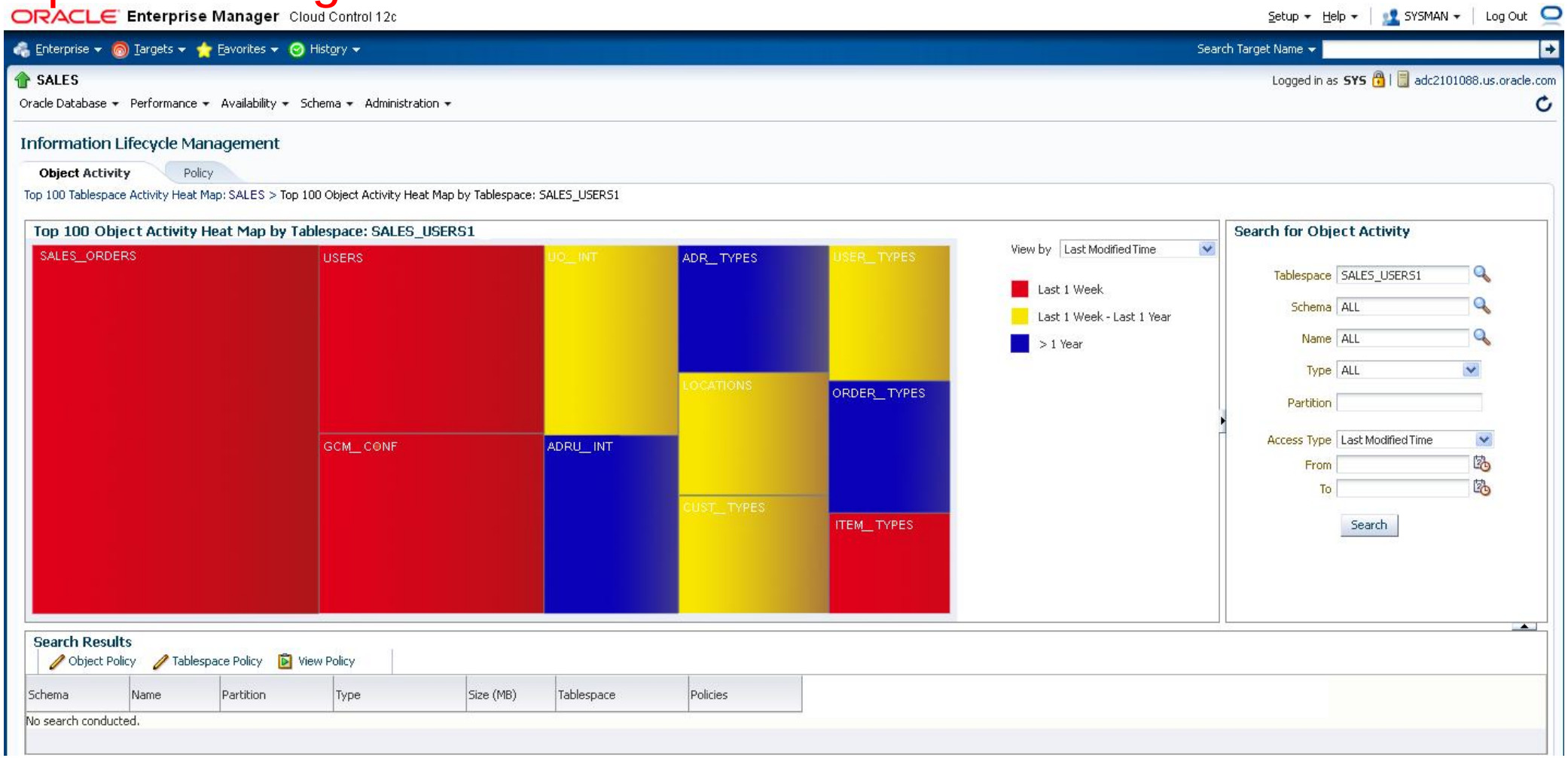
ADO/ILM 対応

Automatic Data Optimization(自動データ最適化)
Information Lifecycle Management（ILM）
Heatmap(ヒートマップ)

ヒート・マップ

- “ヒート・マップ” 追跡
  - 表やパーティションに対するアクセス回数や変更回数が追跡される
  - データベース・ブロックに対する変更回数も追跡される
- 広範囲な追跡
  - セグメント・レベルの読み込み回数、書き込み回数が分かる
  - テーブル・フル・スキャンと索引アクセスを区別する
  - 統計情報収集、DDL、表再定義などの操作は自動的に除外される
- 低いオーバーヘッド
  - オブジェクト・レベルでは、オーバーヘッドは非常に小さい
  - ブロック・レベルでは、5%以下のコスト

ヒート・マップ
Enterprise Manager



12.1.0.1 : 自動データ最適化(ADO/Automatic Data Optimization)

Powerful Policy Specification

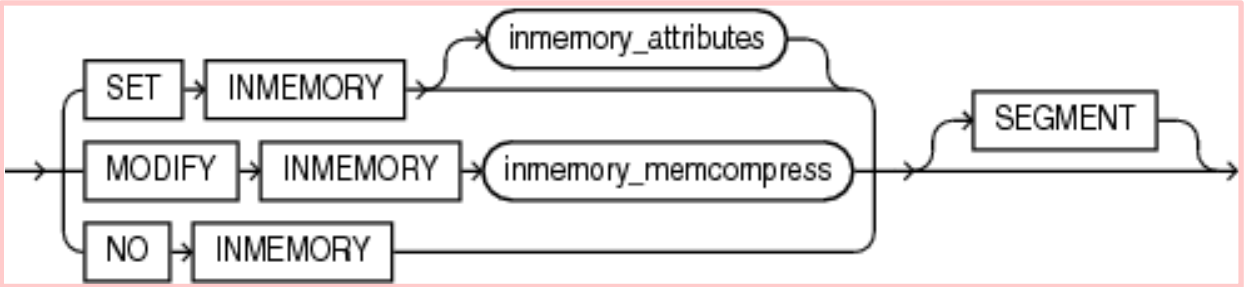
- 宣言型のポリシー設定方法
  - alter table sales ilm policy **row store compress advanced segment after 3 days of no modification;**
  - 条件は、データに対する”creation”や”access”、”modification”が実行されてからの時間を指定
  - アクションは、圧縮形式の変更や表領域の変更が可能
- ポリシーは、表領域ごとや表単位で引き継がれる
  - 新しい表の定義は、表領域の定義が引き継がれる。既存の表へも適用される
  - 新しいパーティション（インターバル・パーティションも含む）は表の定義が引き継がれる

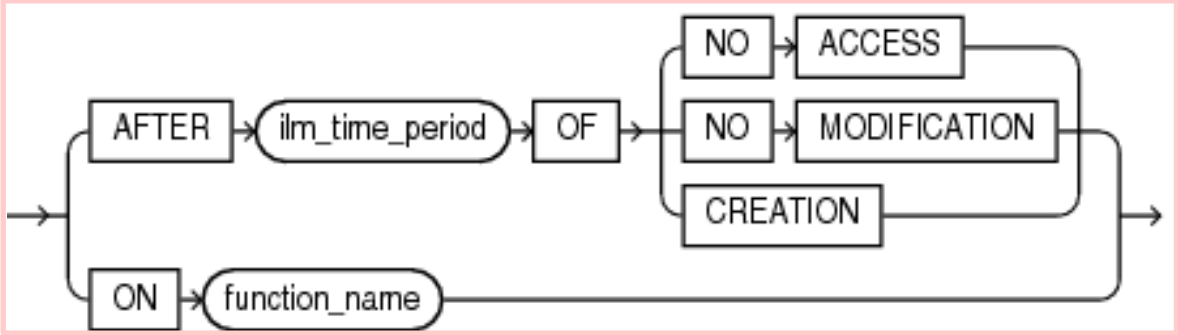
自動データ最適化のインメモリ対応

NEW IN 12.2

ADO IM ポリシー

- ポリシー・クライテリア(条件)
  - [set|modify|no] inmemory after <time spec> of [no access | creation | no modification]
  - [set|modify|no] inmemory on <function_name>
- ポリシー適用が成功すると、3種のアクションのうち1つが適用される
- ポリシーは、表領域、もしくは、表の定義を引き継ぐ
- ポリシーは、メンテナンスウィンドウ中に自動で実行される
- 手動でのポリシー実行 – dbms_ilm.execute_ilm プロシージャ

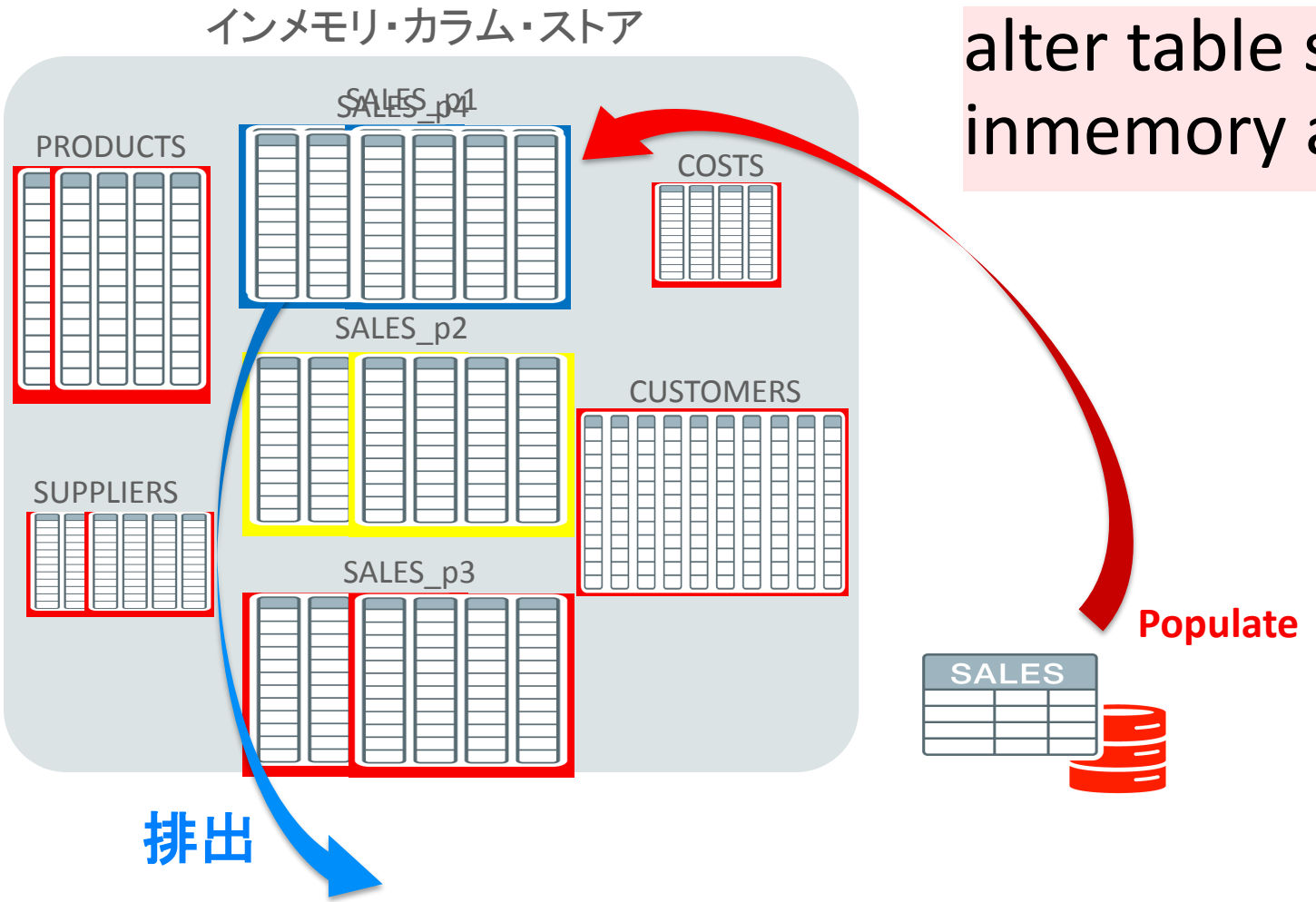




自動データ最適化のインメモリ対応

ポリシー・モードの例 – 作成

NEW IN 12.2



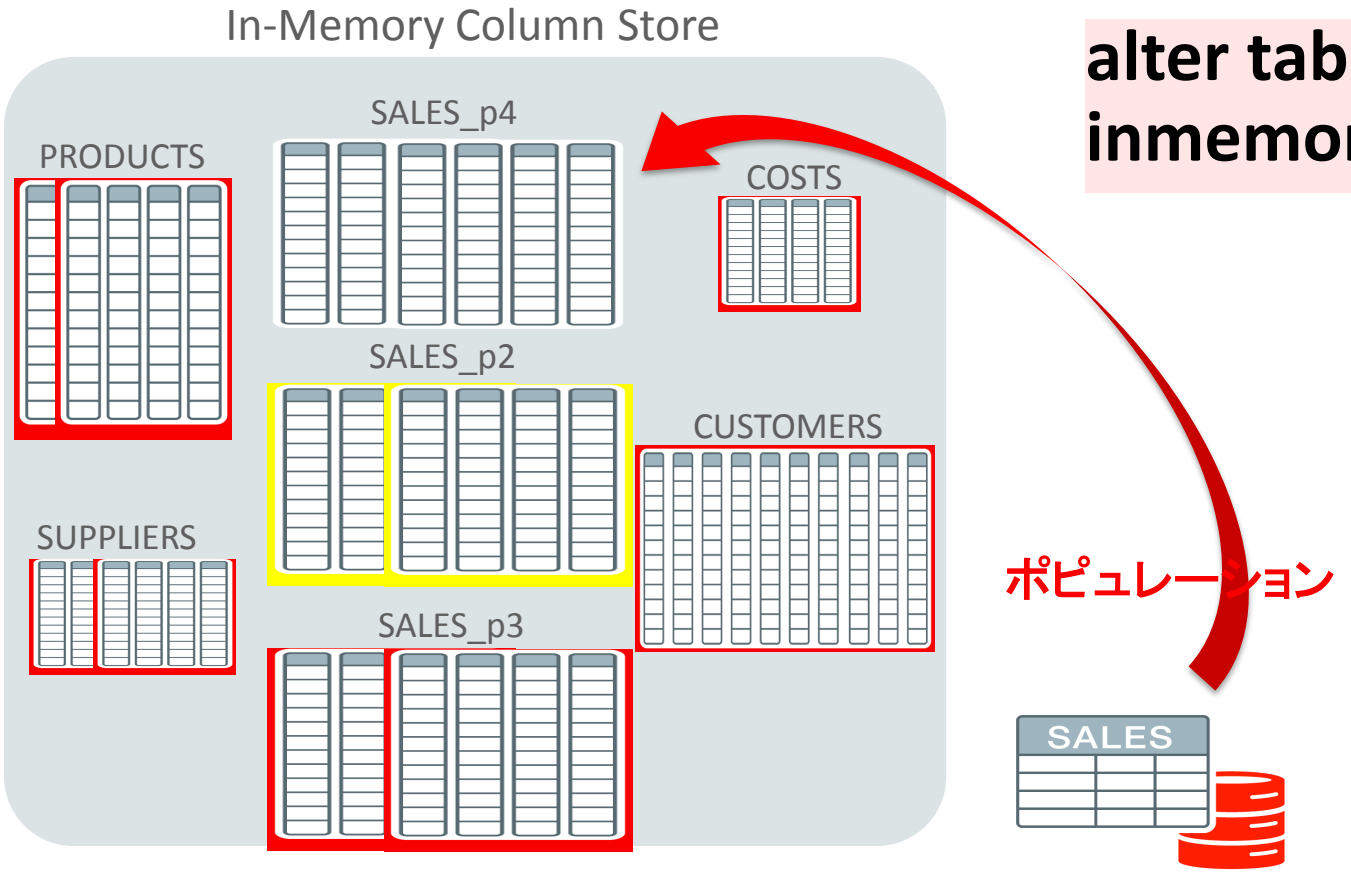

```
alter table sales ilm add policy no
inmemory after 90 days of creation
```

- 定期的に追加作成されるパーティション例
- “Cold” パーティションは、作成されてからの経過日数に従い、エビクト（排出）される
- 新しい月になると、インターバル・パーティションによって、新しいパーティションが作成され、ポピュレートされる

Automatic Data Optimization with Database In-Memory

ポリシーでインメモリー化する例

NEW IN 12.2



alter table sales ilm add policy set inmemory after 10 days of creation;

- 時間差ポピュレーションの例
- 表は作成されてから10日後にインメモリーにポピュレーションされる
- 作成直後に更新が集中するユースケースが実際に多い

RAC

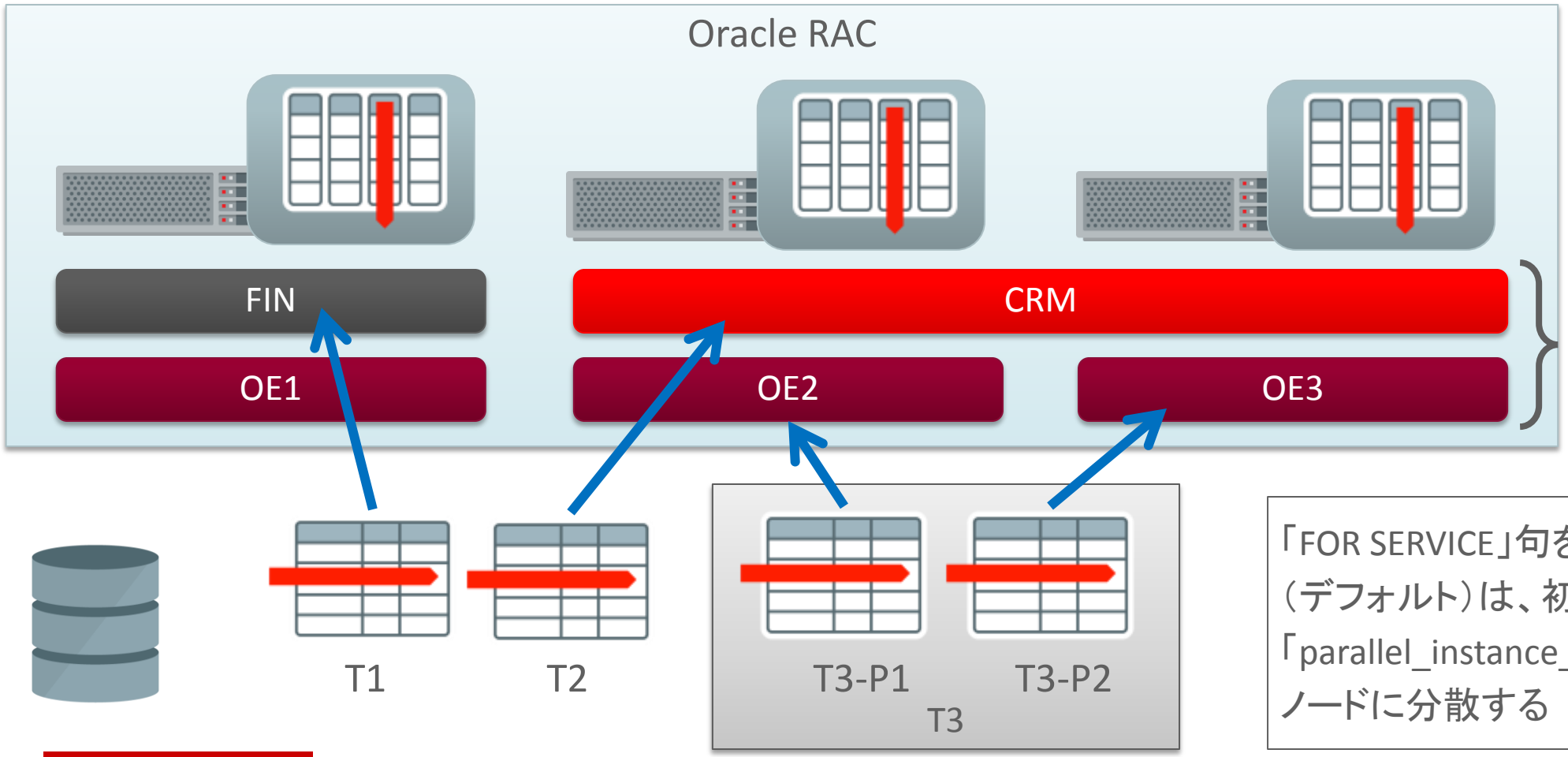
”サービス”対応

RACの設定、構成方法などの変更

- サービス指定は Active Data Guard と共通
  - DISTRIBUTE FOR SERVICE
  - アプリケーションパーティショニングがより容易に
- parallel_instance_group は非推奨に
  - DISTRIBUTE FOR SERVICEが代替手段
- Auto DOP の設定が不要に
  - PARALLEL_DEGREE_POLICY=AUTO
  - Parallel Statement Queuing が有効になってしまう副作用なしに
- Flex Cluster サポート

RAC上のどのインスタンスにポピュレートするか指定可能

ALTER TABLE <Table Name> INMEMORY …. DISTRIBUTE **FOR SERVICE <service name>** ;



「FOR SERVICE」句を指定しない場合（デフォルト）は、初期化パラメータ「parallel_instance_group」、もしくは全ノードに分散する

RACサービス作成方法など

- 12.2でアプリケーションアフィニティー（アプリケーションパーティショニング）構成が容易に
- RAC services should be created with srvctl (register them with the cluster)
- 3つのサービス作成、起動例:

```
srvctl add service -d IMRACDB -service IM1 -preferred IMRACDB1, IMRACDB2
srvctl add service -d IMRACDB -service IM2 -preferred IMRACDB3
srvctl add service -d IMRACDB -service IM3 -preferred IMRACDB4
srvctl start service -d IMRACDB -service "IM1,IM2,IM3"
```

- Corresponding TNS entry example:

```
IM1 =
   (DESCRIPTION=
     (ADDRESS=(PROTOCOL=TCP)(HOST=MySCAN)(PORT=1521))
     (CONNECT_DATA=
        (SERVICE_NAME=IM1)))
```

インメモリ・カラムストアの動的なサイズ変更

NEW IN 12.2

```
ALTER SYSTEM SET
  inmemory_size = 300m scope=both;
```

```
SQL> SELECT *
  2  FROM   v$inmemory_area;

POOL            ALLOC_BYTES USED_BYTES POPULATE_STATUS      CON_ID
--------------- ----------- ---------- ---------------- ----------
1MB POOL          124780544  102760448 DONE                      0
64KB POOL          16777216     851968 DONE                      0

SQL>
SQL> ALTER SYSTEM SET inmemory_size = 300M;

System altered.

SQL>
SQL> SELECT *
  2  FROM   v$inmemory_area;

POOL            ALLOC_BYTES USED_BYTES POPULATE_STATUS      CON_ID
--------------- ----------- ---------- ---------------- ----------
1MB POOL          216006656  102760448 DONE                      0
64KB POOL          32374784     851968 DONE                      0
```

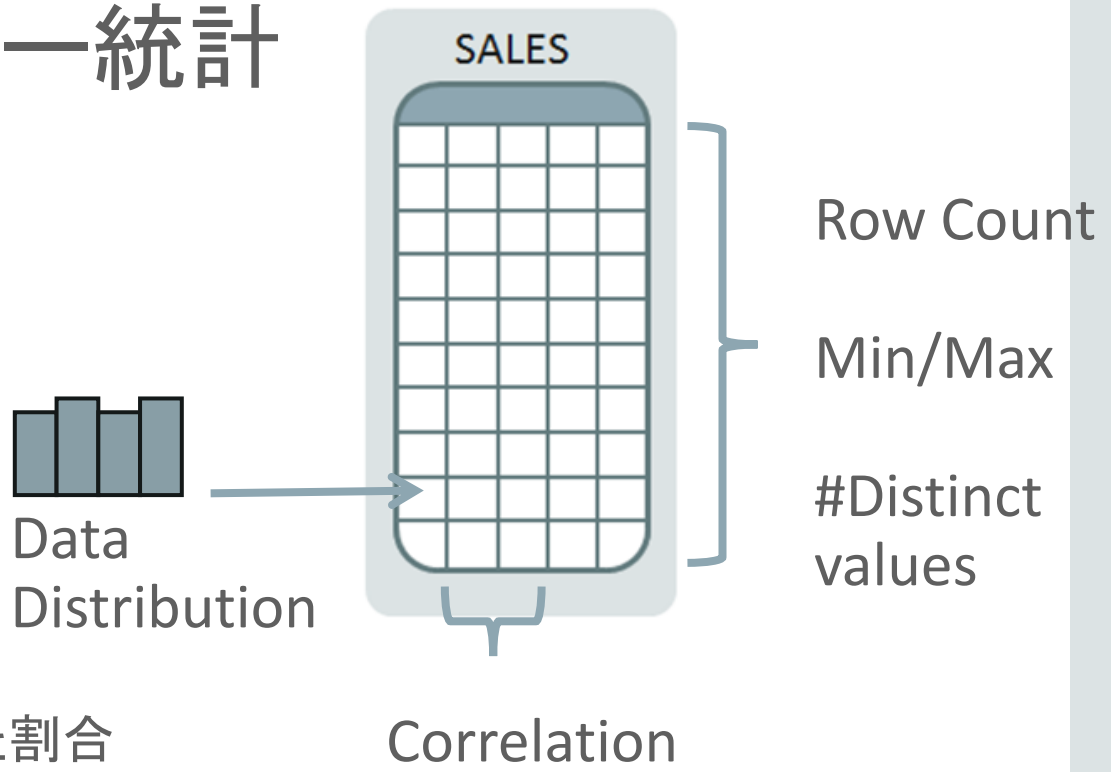
- IMカラムストアは動的に変更可能
  - データベースの再起動をせずにIMカラムストアのサイズ拡張ができる
  - IMカラムストアを小さくはできない
  - 変更するサイズが現在のINMEMORY_SIZEより128MB以上大きいこと
  - 注意点：ホットなBuffer Cache部分が割り当てられることもある
  - 変更はV$INMEMORY_AREAに即座に反映
- 最大はSGA_TARGETの70%

Agenda

1. 12c R1 機能
2. 12c R2 新機能概要
3. 構成の多彩化
4. オプティマイザー/実行計画
5. クエリー高速化

12.1: In-Memory表オプティマイザー統計

- ハードパースのタイミングで取得される
- セグメントレベルの計算 ― 表, (サブ)パーティション
  - **# IMCUs**
  - **# IM Blocks**
  - **IM Quotient**
    - 該当表のインメモリ・カラムストアにポピュレートされた割合
    - 0 から 1 の間の値
  - **# IM Rows**
  - **# IM Transaction Journal Rows**
- **インメモリ統計はRACを認識する(DUPLICATEとDISTRIBUTE)**



10053 Optimizerトレース: SALES表が部分的にインメモリ化されている

```
***************************************
BASE STATISTICAL INFORMATION
***********************
Table Stats::
 Table: SALES  Alias: SALES  (Using composite stats)
 #Rows: 960  SSZ: 0  LGR: 0  #Blks:  12  AvgRowLen:  30.00  NEB: 0  ChainCnt:  0.00  SPC: 0  RFL: 0  RNF: 0  CBK: 0  CHR: 0  KQDFLG: 1
 #IMCUs: 7  IMCRowCnt: 560  IMCJournalRowCnt: 14  #IMCBlocks: 7  IMCQuotient: 0.583333
Index Stats::
 Index: SALES_CHANNEL_BIX  Col#: 4
  USING COMPOSITE STATS
 LVLS: 0  #LB: 12  #DK: 5  LB/K: 2.00  DB/K: 12.00                    00 SSZ: 0.00 LGR: 0.00 CBK: 0.00 GQL: 0.00 CHR: 0.00 KQDFLG: 1
       BSZ: 1
 KKEISFLG: 1
  ALL PARTITIONS USABLE
 Index: SALES_CUST_BIX  Col#: 2
  USING COMPOSITE STATS
 LVLS: 0  #LB: 12
       KQDFLG:
 KKEISFLG: 1
  ALL PARTI
 Index: SALE
  USING CO
 LVLS: 0  #LE
       KQDFL
 KKEISFLG: 1
  ALL PARTI
 Index: SALE
  USING CO
 LVLS: 0  #LE
       KQDFL
 KKEISFLG: 1
  ALL PARTI
 Index: SALE
  USING CO
 LVLS: 0  #LE
       KQDFL
 KKEISFLG: 1
  ALL PARTITI
IMC Implied pred
 Table SALES [SALES]: (null)"SALES"."QUANTITY_SOLD">30 OR "SALES"."QUANTITY_SOLD"<10
...
```

```
SELECT Count(*)
FROM   sales
WHERE quantity_sold > 30
  OR (quantity_sold < 10
      AND prod_id = 5000);
```

```
Table: SALES  Alias: SALES  (Using composite stats)
  #Rows: 960  SSZ: 0  LGR: 0  #Blks:  12  AvgRowLen:  30.00
NEB: 0  ChainCnt:  0.00  SPC: 0  RFL: 0  RNF: 0  CBK: 0  CHR: 0
KQDFLG: 1
 #IMCUs: 7  IMCRowCnt: 560 #IMCBlocks: 7  IMCQuotient: 0.583333
```

Oracle Database In-Memory用の統計

NEW IN 12.2

- 表、(サブ)パーティションに対する[ALL|DBA|USER]_TAB_STATISTICS ビュー 内に新しい列:
  - **IM_IMCU_COUNT**: In-Memory compression units (IMCUs)数
  - **IM_BLOCK_COUNT**: In-Memory ブロック数
  - **IM_STAT_UPDATE_TIME**: インメモリ統計が更新された時間
- Oracle Database In-Memoryがサポートしている統計情報の可視化
  - 透過性の改善
  - ディスクに保存され、シングル or RACに関係なく統計情報を単一セットとして保持
  - インメモリのポピュレーション、非インメモリ化の情報を元に自動で管理(IMCO)
  - インメモリ統計に対してもDBMS_STATS関数を利用可能
- Active Data Guard スタンバイでは使用されない

Expression Statistics Store

NEW IN 12.2

Expressions 統計ストア

- オプティマイザ層で管理
- クエリのハードパースで演算式を特定
  - SELECTリスト、WHERE句、GROUP BY 等
  - ユニーク演算式ID(カノニカル表現、オブジェクト番号)
- 演算式評価統計のメンテナンス
  - 実行頻度(ヒューリスティック:行ソースの行の出入り)
  - コスト評価
  - タイムスタンプ評価
  - 「温度」スコア(アクセス頻度)
- ディスクに保存 (より高速なアクセスのために SGAにもキャッシュされる)

```
SELECT prod_list_price - prod_min_price
FROM   products;
```

Expression Tracking 例

NEW IN 12.2

```
SELECT prod_list_price - prod_min_price
FROM   products;

SELECT expression_text, evaluation_count,
       fixed_cost
FROM   user_expression_statistics
WHERE  table_name = 'PRODUCTS';

EXPRESSION_TEXT                    EVALUATION_COUNT FIXED_COST
---------------------------------- ---------------- ----------
"PROD_LIST_PRICE"-"PROD_MIN_PRICE” 766              .000041667
```

Agenda

1. 12c R1 機能
2. 12c R2 新機能概要
3. 構成の多彩化
4. オプティマイザー/実行計画
5. クエリー高速化

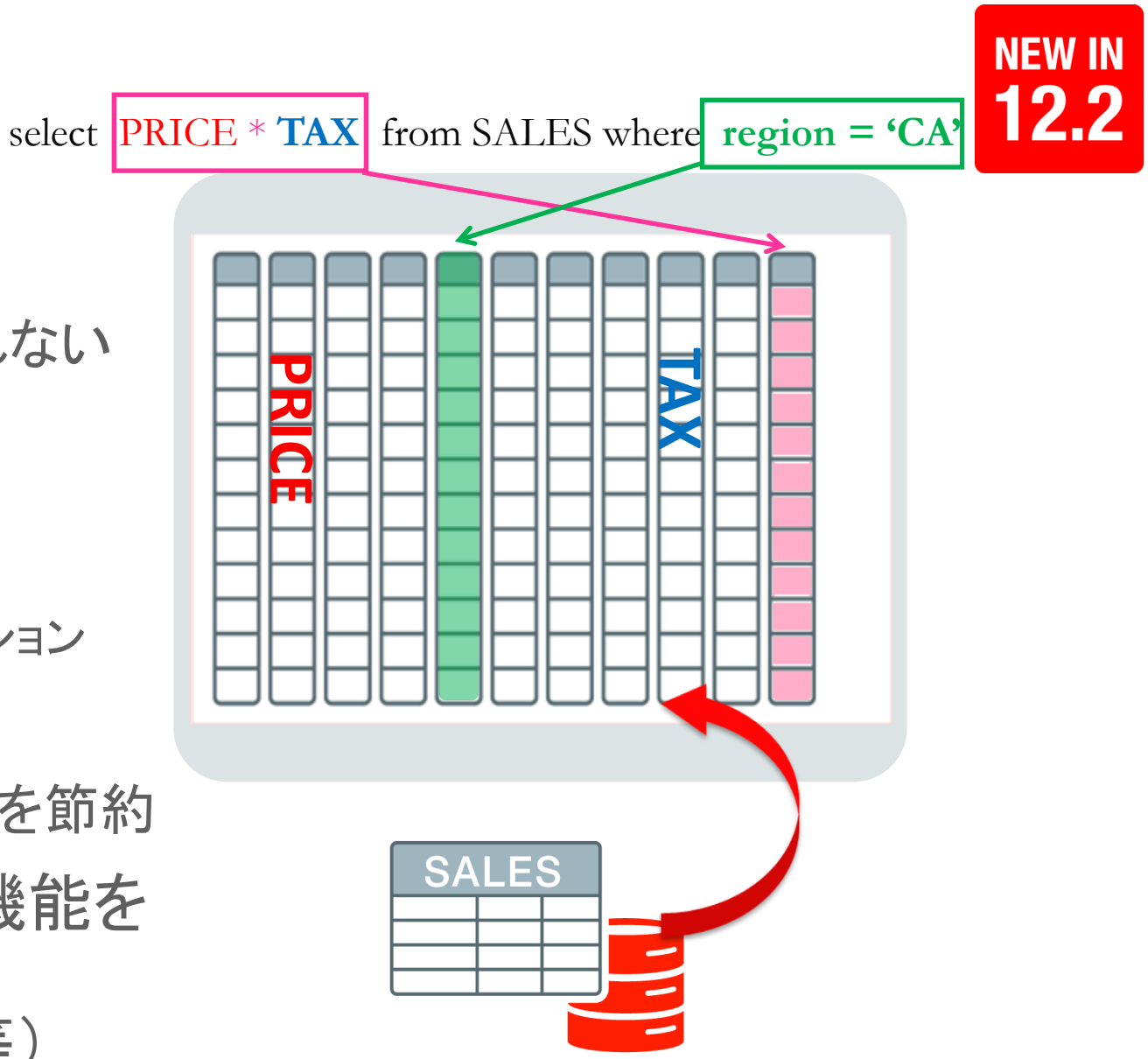
In-Memory Expression(IME)

仮想カラム
事前計算列
実行履歴からのキャプチャー

In-Memory Expressions
どのように動作するか？

NEW IN 12.2

- インメモリのみの列
  - ディスクでは実体化(materialize)されない
  - ユーザ定義仮想列
  - 自動キャプチャー
    - 頻繁に評価される演算式
    - 頻出関数、文字列操作、PL/SQLファンクション
- 繰返し評価を減らす
  - 一度だけ演算することでCPUの利用を節約
- IMEであってもインメモリ最適化機能をサポート
  (min/maxプルーニング、SIMD　等）



In-Memory Expressions

Types

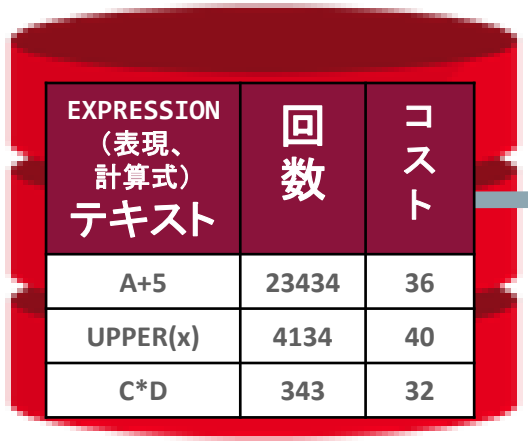
NEW IN 12.2

```
CREATE TABLE t (
        a NUMBER,
        b NUMBER,
        v AS ( a + b ) );

SELECT a * b
FROM   t
WHERE  a / b = 1;
```

- 仮想列
  - 表内のカラムを含んだ演算式表現
  - ディスク上にはデータなし
  - その他のカラムと同様に検索可能 (オン・ザ・フライ評価)
- 自動In-Memory Expression
  - "Hot" な演算式を自動検知
  - 表内の単一行の1つ以上の列で定義され、定数も含むことも可
  - 表内の行と1対1でマッピングされる

In-Memory Expressions：自動キャプチャー

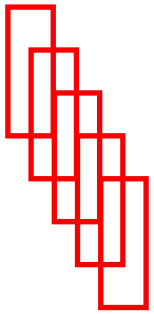
Expression 統計ストア(ESS)



EXPRESSION（表現、計算式）テキスト	回数	コスト
A+5	23434	36
UPPER(x)	4134	40
C*D	343	32

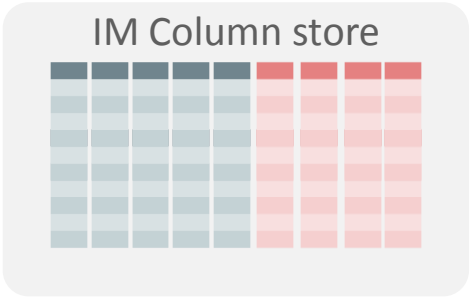
- Expressions Statistics Store (ESS) は常にワークロードをモニター
  - “Hot”な計算式や関数呼び出しをパース時に頻度やコストに応じて累積的に記録

上位20個のexpression (計算式)をキャプチャー



- 12.2の新規のprocedureのIME_CAPTURE_EXPRESSIONS()を実行するとインスタンス中の上位20個の計算式が判別されキャプチャーされる
  - DBMS_INMEMORY_ADMINパッケージの1プロシージャー

IMEは インメモリー・カラム・ストアに 自動的に作成される



- IME_POPULATE_EXPRESSIONS()を実行すると隠し仮想カラムが作成され計算結果、関数実行結果の値がインメモリー領域にポピュレーションされる
  - DBMS_INMEMORY_ADMINパッケージの1プロシージャー

In-Memory Expressions

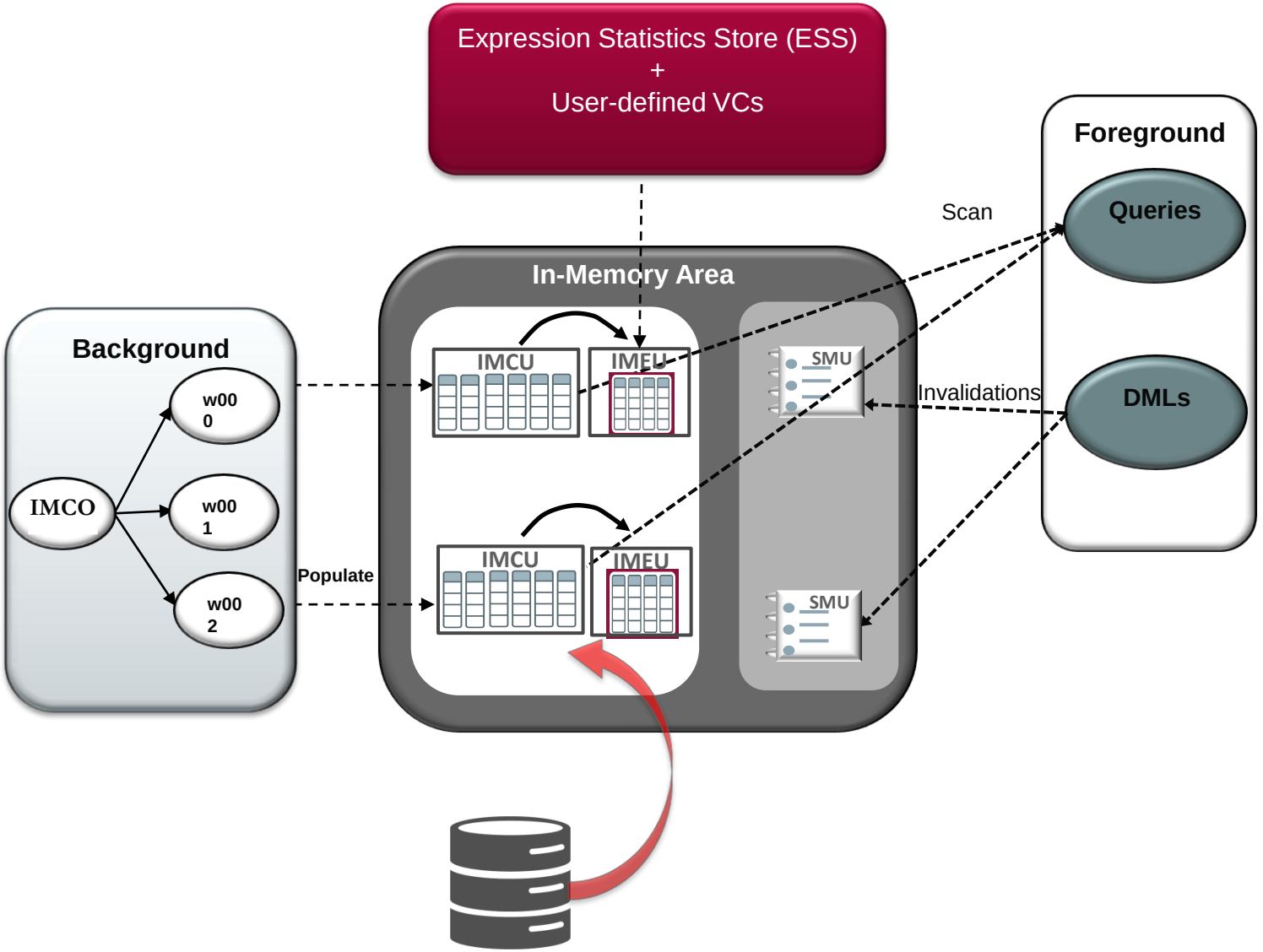
NEW IN 12.2

IMEをIMカラムストアのどこに格納するか？

- In-Memory Expression Unit (IMEU)
  - –IMEと仮想列を格納するインメモリ・ユニット
    - IMCUの隣接した領域
    - IMCUの行と1対1でマッピングされる
  - –微妙な差異があるがIMCUと同じような構造
  - –IMCU あたりに1つのIMEU
    - 親IMCUのライフスパンと連動
  - –IMCU 1 MB pool のメモリ領域

IME: ポピュレーションからクエリー実行まで

NEW IN 12.2



- IMCO → population framework
- IMCUとSMUの構築
- クエリに対してIMCUを有効化
- IMEU構築
  - ✓ ポピュレートする仮想列の特定
  - ✓ 仮想列値の作成
  - ✓ カラムフォーマット化と圧縮
  - ✓ スペース層で登録
  - ✓ IMEUをIMCUにリンク
- クエリに対してIMEUを有効化

仮想カラムIn-Memory Expressionを有効化

NEW IN 12.2

初期化パラメータ

- INMEMORY_VIRTUAL_COLUMN
  - ユーザ定義のカラム（仮想列）のインメモリポピュレーションを制御
  - 自動IMEには影響を与えない
  - 入力可能値:
    - ENABLE – 「NO INMEMORY」指定で除外しない限り、インメモリ化されている表やパーティション上に定義されたユーザ定義の全仮想列をポピュレートする
    - MANUAL – (デフォルト) ユーザ定義仮想列が明示的に「INMEMORY」とマークされている場合だけポピュレートする
    - DISABLE – ユーザ定義仮想列をポピュレートしない

```
alter table foo inmemory memcompress for capacity high (vc1) no inmemory (vc2);
```

自動キャプチャーIn-Memory Expressions有効化

NEW IN 12.2

初期化パラメータ

- INMEMORY_EXPRESSIONS_USAGE
  - 自動 In-Memory Expressionのポピュレートを制御
  - ユーザ定義仮想列には影響を与えない
  - 入力可能値
    - ENABLE – JSONと自動In-Memory Expressionの両方をポピュレートする
    - DISABLE – 自動In-Memory Expressionをポピュレートしない
    - STATIC_ONLY – Oracle JSON バイナリ表現のみポピュレートする
    - DYNAMIC_ONLY – 頻繁に利用され、コストが高いIn-Memory Expressionを ESSから判断し、ポピュレートする

Real-Time Analytics: In-Memory Expressions

NEW IN 12.2

- 12.1 パフォーマンス
  - (Parallel 8)

```
SQL> select  /*+ parallel(8) */
        lo_shipmode,
        sum(lo_quantity) as sum_qty,
        sum(lo_extendedprice) as sum_base_price,
        sum(lo_extendedprice * (1 - lo_discount)) as sum_disc_price,
        sum(lo_extendedprice * (1 - lo_discount) * (1 + lo_tax)) as sum_charge,
        avg(lo_quantity) as avg_qty,
        avg(lo_extendedprice) as avg_price,
        avg(lo_discount) as avg_disc,
        count(*) as count_order
from
        lineorder
group by
        lo_shipmode
order by
        lo_shipmode;
  2    3    4    5    6    7    8    9    10   11   12   13   14   15   16

7 rows selected.

Elapsed: 00:07:35.09
```

- 12.2 パフォーマンス
  - (Parallel 8)

```
SQL> select  /*+ parallel(8) */
        lo_shipmode,
        sum(lo_quantity) as sum_qty,
        sum(lo_extendedprice) as sum_base_price,
        sum(lo_extendedprice * (1 - lo_discount)) as sum_disc_price,
        sum(lo_extendedprice * (1 - lo_discount) * (1 + lo_tax)) as sum_charge,
        avg(lo_quantity) as avg_qty,
        avg(lo_extendedprice) as avg_price,
        avg(lo_discount) as avg_disc,
        count(*) as count_order
from
        lineorder
group by
        lo_shipmode
order by
        lo_shipmode;
  2    3    4    5    6    7    8    9    10   11   12   13   14   15   16

7 rows selected.

Elapsed: 00:02:05.31
```

3.5倍 パフォーマンス高速化

紫色バックグラウンド部分の計算式を仮想カラムとして追加

自動In-Memory Expressionキャプチャーおよび注意点など

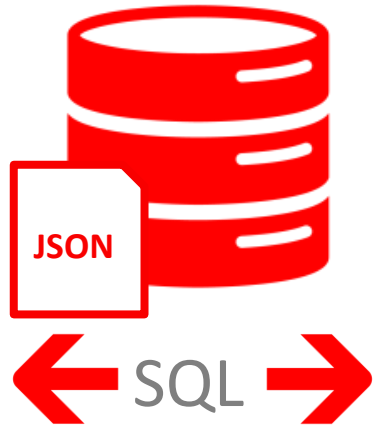
- DBAによるキャプチャー実行

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> EXEC DBMS_INMEMORY_ADMIN.IME_CAPTURE_EXPRESSIONS('CURRENT')
PL/SQL procedure successfully completed.
SQL> EXEC dbms_inmemory_admin.ime_populate_expressions()
```

```
SQL> SELECT column_name, sql_expression FROM v$im_imecol_cu;
COLUMN_NAME   SQL_EXPRESSION
-----------   --------------------
VC1           "C1"+"C2"
VC2           "C1"*2
```

直近24時間

- カラム名変更にはIME機能は追従しない
- ユーザー定義PL/SQLファンクションもDETERMINISTIC宣言されていればIME化可能



JSON

仮想カラム
バイナリ形式

マニュアル構成

- 新規のJSON独立のマニュアル
- In-Memoryの章

ORACLE Help Center Welcome Katsum

Home / Database / Oracle Database 12c Release 2 / Development

Database JSON Developer's Guide

26 In-Memory JSON Data

A column of JSON data can be stored in the In-Memory Column Store (IM column store) to improve query performance.

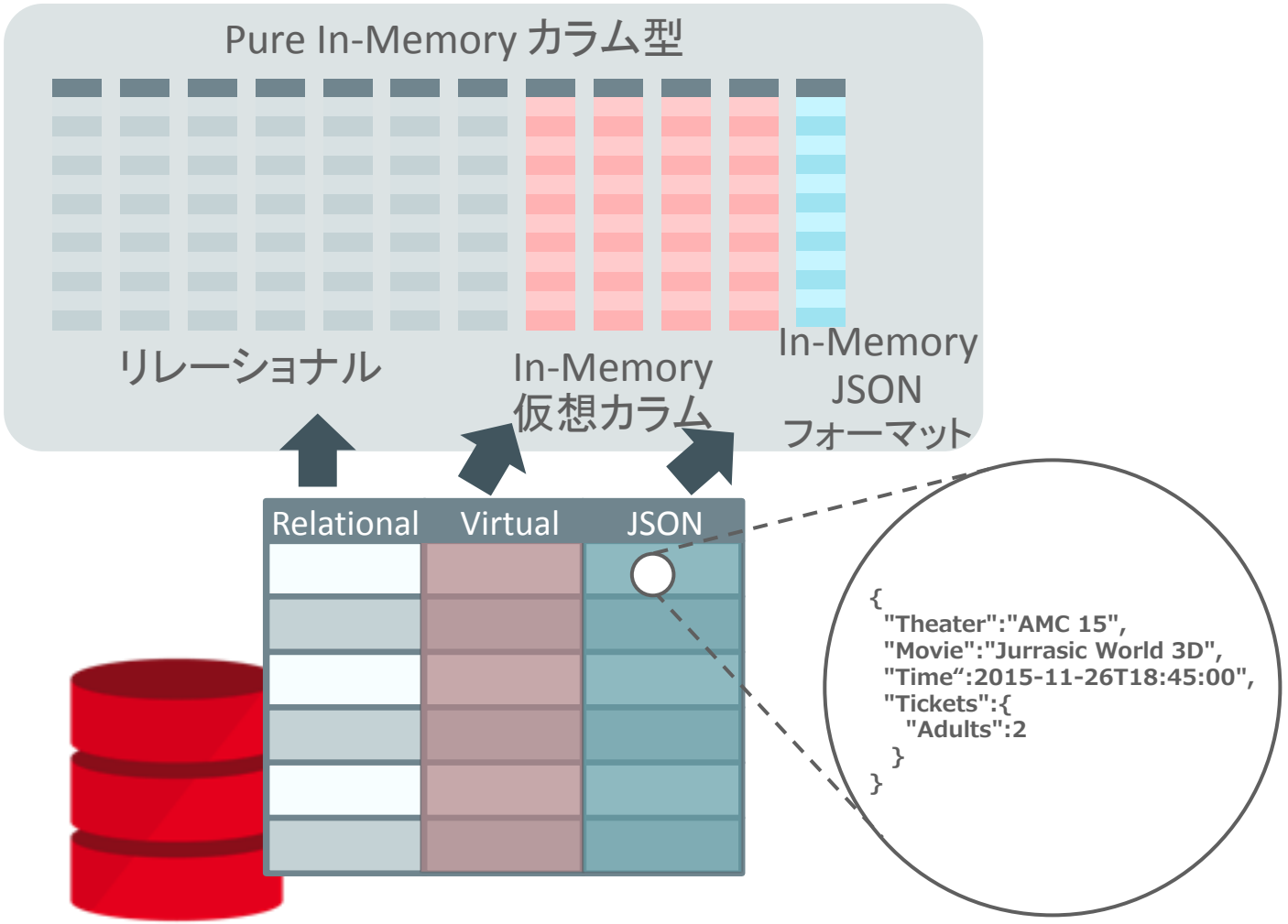
- Overview of In-Memory JSON Data
- Populating JSON Data Into the In-Memory Column Store
- Upgrading Tables With JSON Data For Use With the In-Memory Column Store

See Also:

Oracle Database In-Memory Guide

リアルタイムなマルチモデル分析: In-Memory JSON

NEW IN 12.2



- JSONドキュメント全体を最適化されたバイナリーフォーマットでポピュレート可能
- JSON列上のexpression(表現、計算式)の事前処理結果をカラムストアにポピュレーション可能
  - 例) JSON_VALUE(…,…)
- JSONデータへのクエリーでは自動的にIn-Memoryフォーマットが使われる
  - 例) "movie.name"属性が“Jurassic”文字列を含む映画を検索
- **60倍** の高速化も可能(場合による)

Real-Time Multi-Model Analytics: In-Memory JSON

NEW IN 12.2

- 12.1 パフォーマンス
  - (Parallel 4)

```
SQL> SELECT  count(*) FROM retail o,
JSON_TABLE(o.c FORMAT json, '$'
COLUMNS (Country_Code VARCHAR2(5) PATH '$.CountryCode',
         Trans_Date   NUMBER      PATH '$.StoreTransaction.EndDateTime.Ddate',
         Store_no     NUMBER      PATH '$.StoreTransaction.StoreNo',
         NESTED PATH '$.LineItems[*]'
            COLUMNS (lineitem_code     VARCHAR2(20)  PATH  '$.Identity.Code',
                     lineitem_amount   VARCHAR2(10)  PATH  '$.Amount',
                     lineitem_quantity VARCHAR(6)    PATH  '$.Units.Quantity'
                    )
       )
     ) lineitems;
  2    3    4    5    6    7    8    9   10   11   12

           COUNT(*)
--------------------
          351617024

Elapsed: 00:16:16.93
```

- 12.2 パフォーマンス
  - (Parallel 4)

```
SQL> SELECT  count(*) FROM retail o,
JSON_TABLE(o.c FORMAT json, '$'
COLUMNS (Country_Code VARCHAR2(5) PATH '$.CountryCode',
         Trans_Date   NUMBER      PATH '$.StoreTransaction.EndDateTime.Ddate',
         Store_no     NUMBER      PATH '$.StoreTransaction.StoreNo',
         NESTED PATH '$.LineItems[*]'
            COLUMNS (lineitem_code     VARCHAR2(20)  PATH  '$.Identity.Code',
                     lineitem_amount   VARCHAR2(10)  PATH  '$.Amount',
                     lineitem_quantity VARCHAR(6)    PATH  '$.Units.Quantity'
                    )
       )
     ) lineitems;  2    3    4    5    6    7    8    9   10   11   12

  COUNT(*)
----------
 351617024

Elapsed: 00:00:41.14
SQL>
```

23倍 パフォーマンス高速化

JSONドキュメント全体がインメモリー化されている例

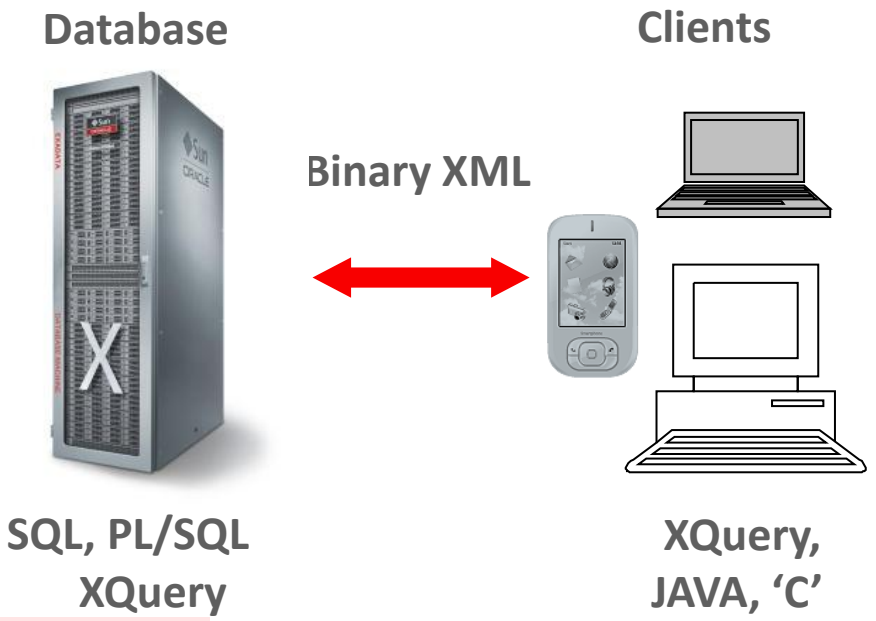
JSON対応: 詳細

- VARCHAR2サイズ32KB化が必要
  - max_string_size=extended
- 最大サイズ 32KB
- 右のサンプルJSONドキュメントは改行、タブなどを除くと716バイトなので右が約45個分が最大
- "IS JSON" 制約が該当カラムに必要


```
{
        "PONumber": 1600,
        "Reference": "ABULL-20140421",
        "Requestor": "Alexis Bull",
        "User": "ABULL",
        "CostCenter": "A50",
        "ShippingInstructions": {
                "name": "Alexis Bull",
                "Address": {
                        "street": "200 Sporting Green",
                        "city": "South San Francisco",
                        "state": "CA",
                        "zipCode": 99236,
                        "country": "United States of America"
                },
                "Phone": [
                        {
                                "type": "Office",
                                "number": "909-555-7307"
                        }, {
                                "type": "Mobile",
                                "number": "415-555-1234"
                        }
                ]
        },
        "SpecialInstructions": null,
        "AllowPartialShipment": false,
        "LineItems": [
                {
                        "ItemNumber": 1,
                        "Part": {
                                "Description": "One Magic Christmas",
                                "UnitPrice": 19.95,
                                "UPCCode": 13131092899
                        },
                        "Quantity": 9.0
                }, {
                        "ItemNumber": 2,
                        "Part": {
                                "Description": "Lethal Weapon",
                                "UnitPrice": 19.95,
                                "UPCCode": 85391628927
                        },
                        "Quantity": 5.0
                }
        ]
}
```

Binary XML との共通点と違い

- 共通点
  - 要素の高速な取り出しが可能
  - サイズ小
- 違い
  - Binary XMLはその形式でDBに永続化できる
  - クライアントでBinary XML化できるライブラリがありその形式でOracle DBと直接やり取り可能

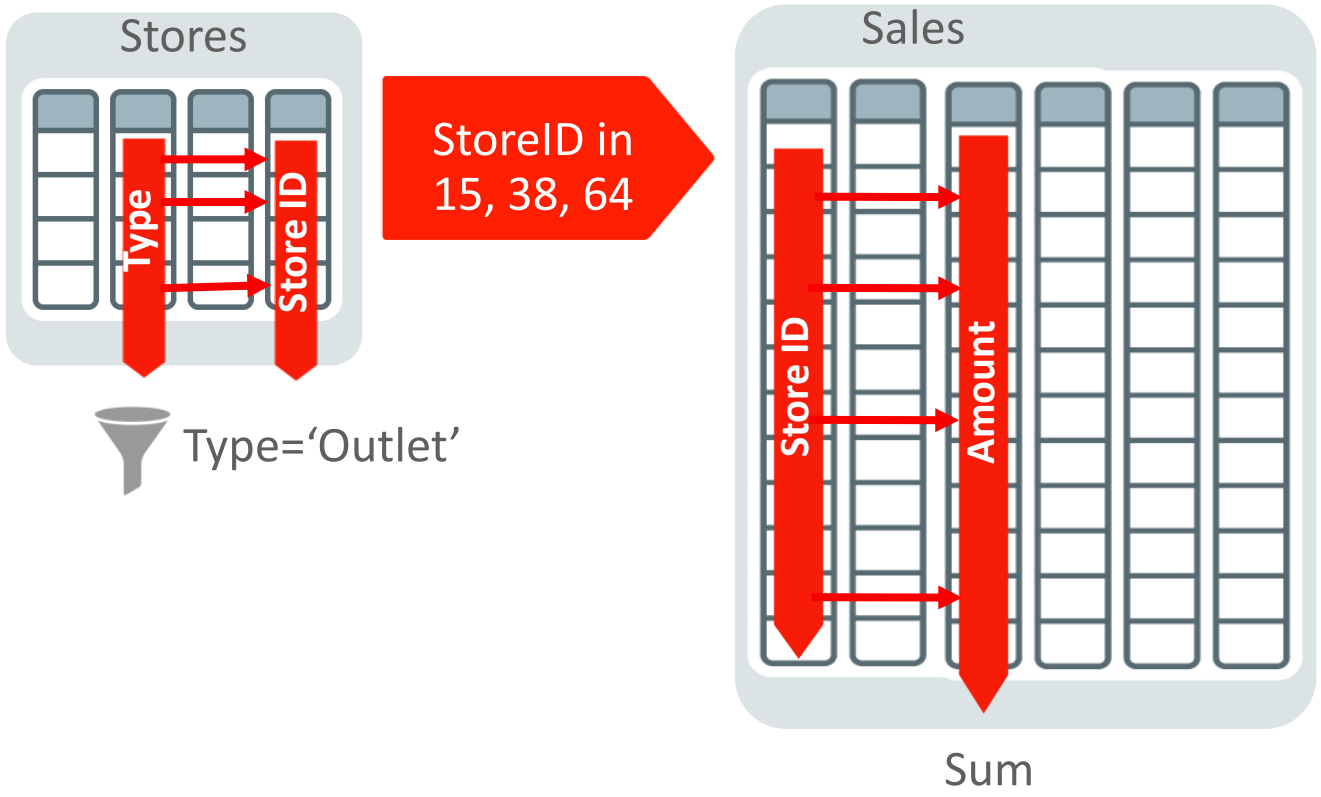



```
SQL> create table PURCHASEORDER of XMLTYPE
     XMLTYPE store as SECUREFILE BINARY XML;
```

Join Groups

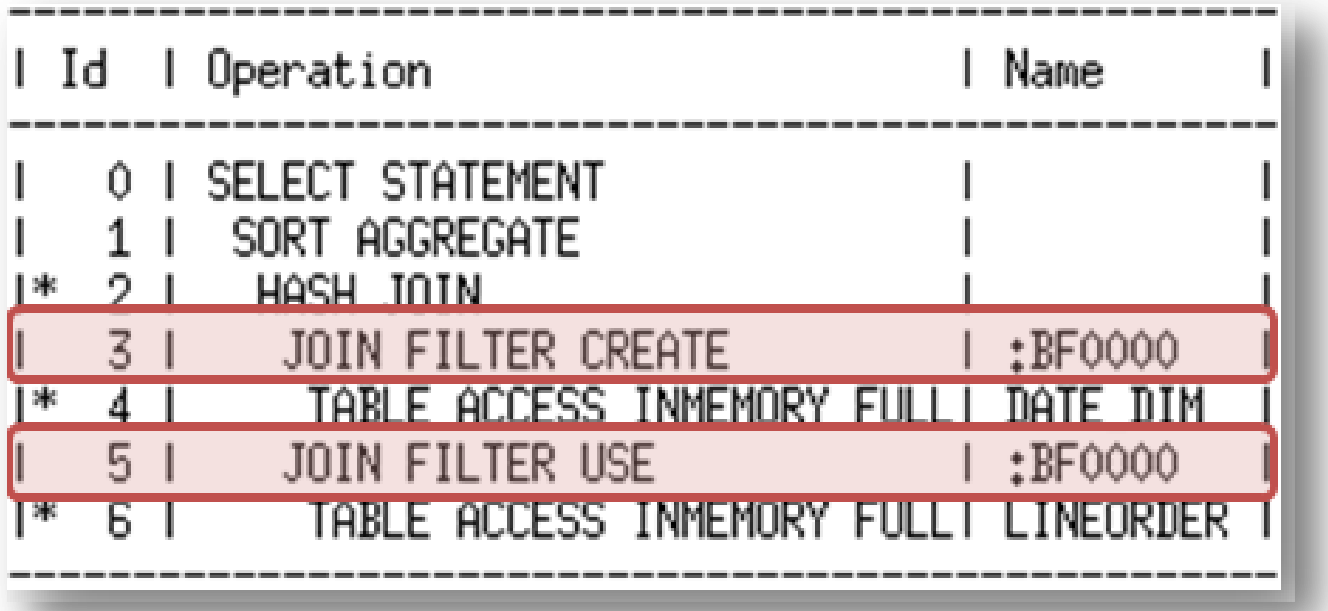
ジョイン および データの結合

例: アウトレット店舗での総売り上げを問い合わせる



- 複数の表のデータのジョインを高速な列 のスキャンへと変換
- WHERE句フィルターがある場合、ブルームフィルターにより表のジョインが **10倍**高速

確認方法：INMEMORY のジョインでのBloom Filter使用



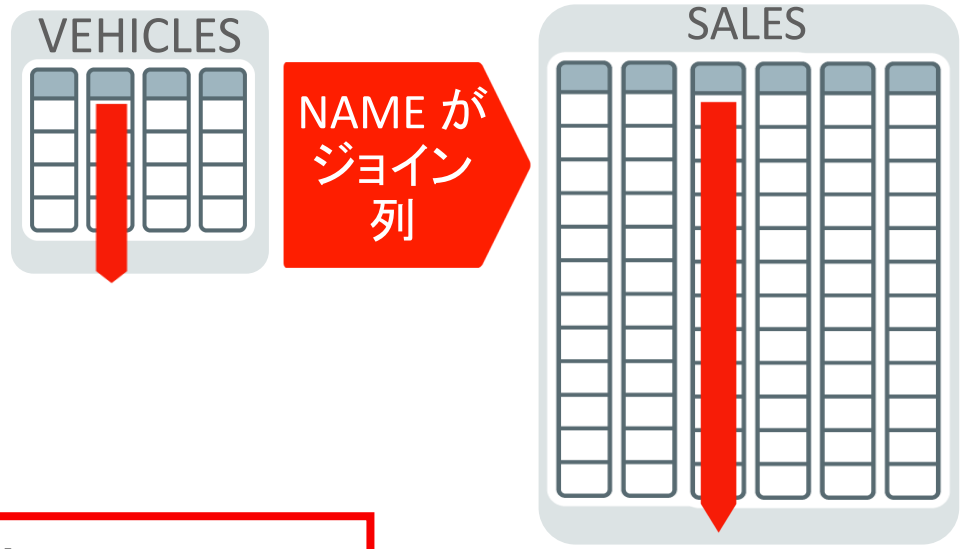
```
-----------------------------------------------------
| Id  | Operation                      | Name       |
-----------------------------------------------------
|   0 | SELECT STATEMENT               |            |
|   1 |  SORT AGGREGATE                |            |
|*  2 |   HASH JOIN                    |            |
|   3 |    JOIN FILTER CREATE          | :BF0000    |
|*  4 |     TABLE ACCESS INMEMORY FULL | DATE_DIM   |
|   5 |    JOIN FILTER USE             | :BF0000    |
|*  6 |     TABLE ACCESS INMEMORY FULL | LINEORDER  |
-----------------------------------------------------
```

- ブルーム・フィルターがジョインを高速な列スキャンに変換することを可能にする
- 10g から存在する実績のあるテクノロジー
- 近い技術がExadataへのオフロード・ジョインでも使われている

Bloom Filterが使われない Hash Join

```
SELECT v.year,
       v.name,
       s.sales_price
FROM   Vehicles v, Sales s
Where  v.name = s.name;
```

例: 各々の車輌の販売価格を問い合わせ

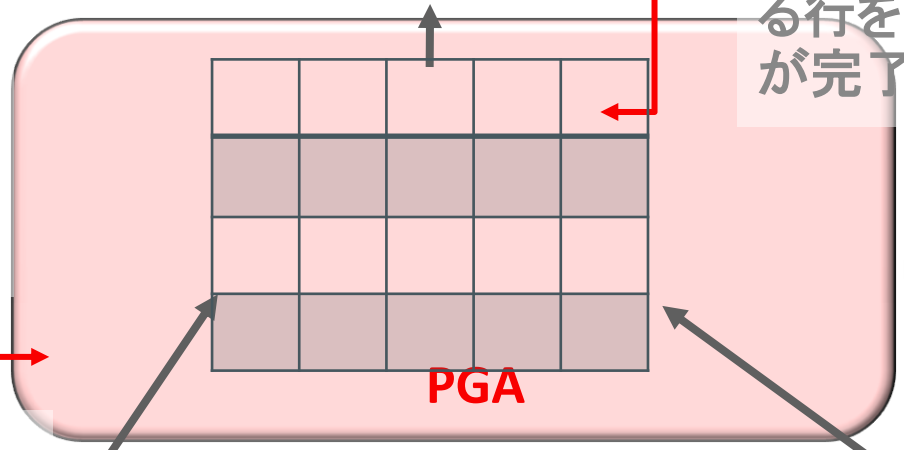


12.1では圧縮値の伸張とハッシュ値計算が実行時間の大部分を占めるためあまり高速化されない

Hash Join

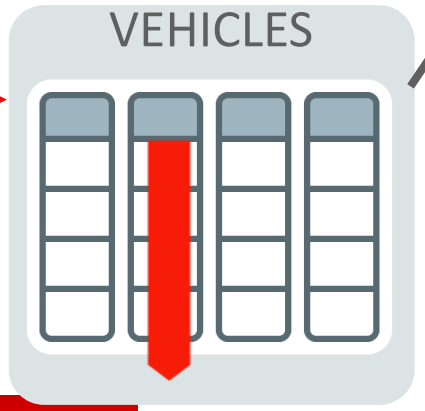
5 **Hash Join: stord_idからのハッシュテーブルをprobeし実際にマッチしている行を見つけることによりジョイン処理が完了**

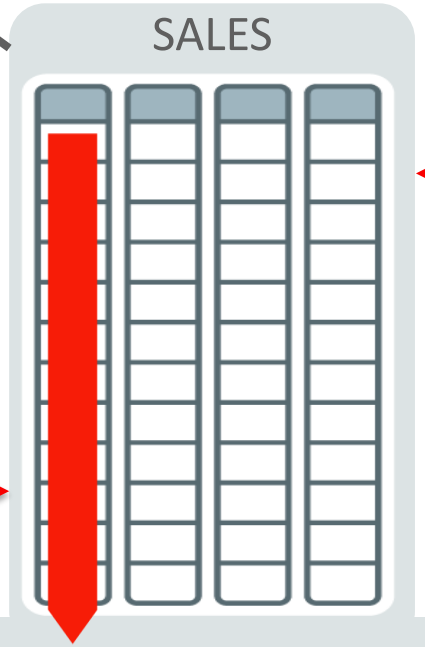
2 **Hash Table:**
VEHICLE表の伸張されたジョインカラムの値を基にメモリー中にhash tableが作成される



4 **行をHash Joinへ送出:**
マッチする行のみがdecompressされ、および、hashされ hash joinへ渡される

1 **表走査:**
VEHICLES表がスキャンされ、一致する行がdecompressされ、および、hash joinへ渡される





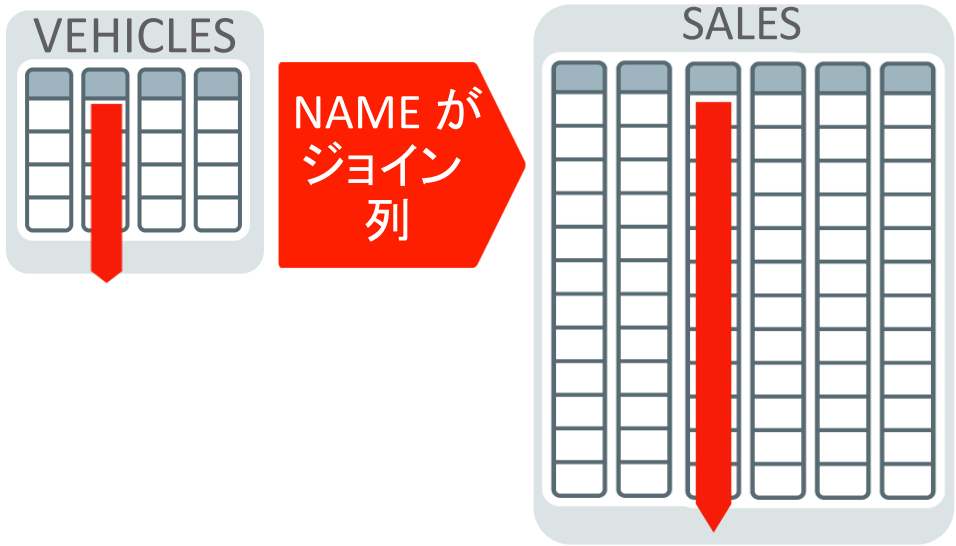
3 **表走査:** SALES 表がスキャンされ、および、行がクエリーの述語でフィルターされる

ジョイン・グループ: より高速な Hash ジョイン

NEW IN 12.2

- 両方の表からのジョイン列を共通の”ディクショナリー”で圧縮
- ジョインは実際の列の値データではなく”ディクショナリー値”で行われる
  - 圧縮を伸張する処理を省略できる
  - データのHash処理も省略
- "表名(カラム名) "グループ化最大数は 255
- 同一カラムは2つ以上のJOIN GROUPに属することはできない
  - ORA-00957: duplicate column name

例:各々の車輌の販売価格を問い合わせ



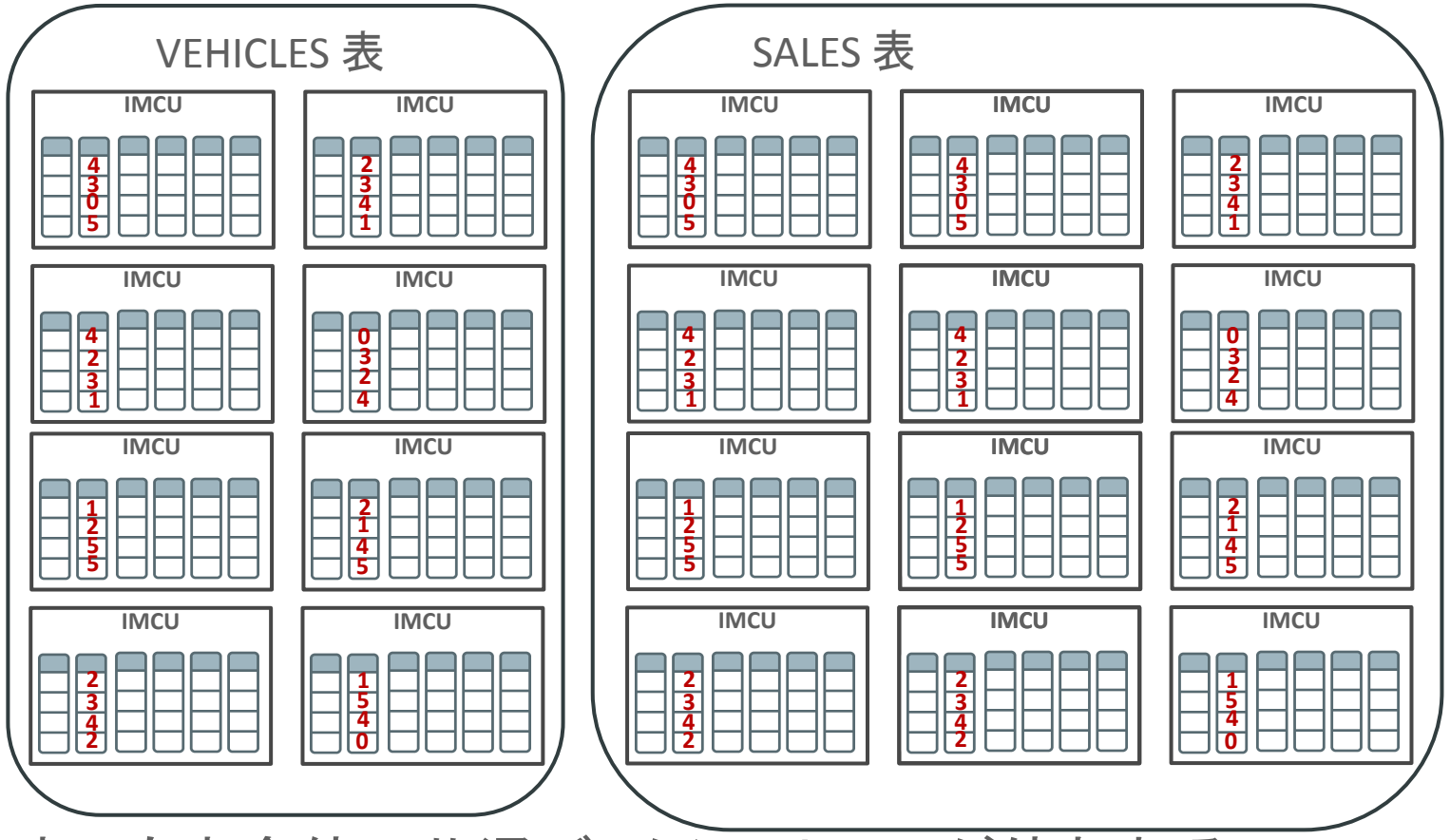

```
CREATE INMEMORY JOIN GROUP
   V_name_jg
   (VEHICLES(NAME),SALES(NAME));
```

ジョイン・グループ: 共通のディクショナリーが両方の表で使われる

NEW IN 12.2

共通ディクショナリー

NAME	ID
AUDI	0
BMW	1
CADILLAC	2
PORSCHE	3
TESLA	4
VW	5



パーティション化されている表でも表全体で共通ディクショナリーが使われる

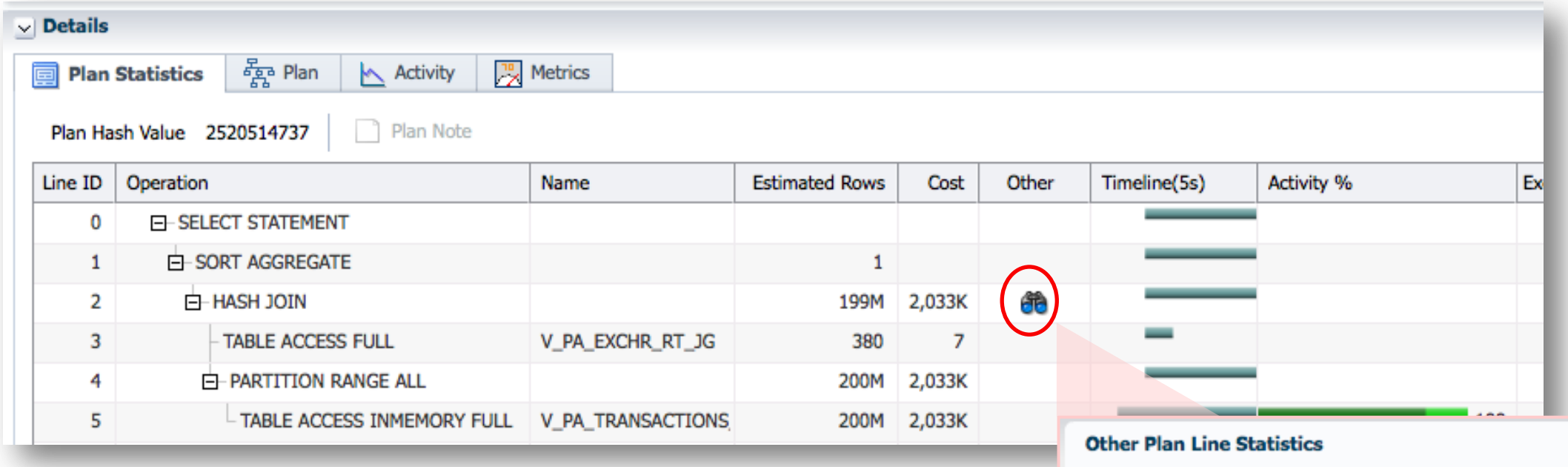
作成からポピュレーションの確認、および、注意点

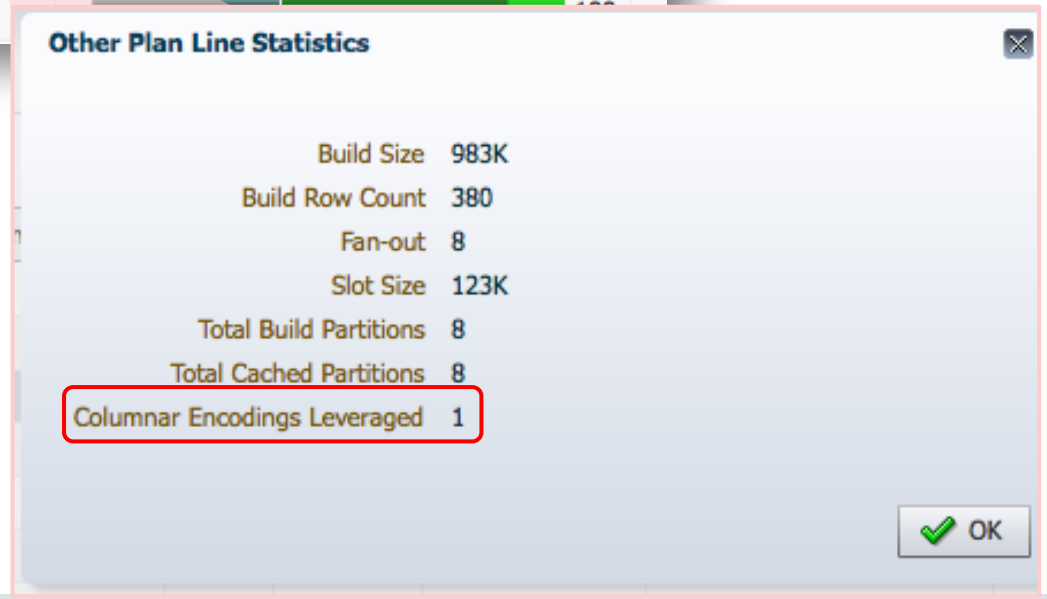
```
CREATE INMEMORY JOIN GROUP V_name_jg
(VEHICLES(NAME),SALES(NAME));

SELECT joingroup_name, table_name,
    column_name
FROM   user_joingroups;

SELECT o.object_name, c.column_name ,
       cd.head_address "Common Dict"
FROM   user_objects o,
       user_tab_columns c,
       v$im_segdict cd
WHERE  cd.objn = o.object_id
AND    o.object_name = c.table_name
AND    cd.column_number = c.column_id;
```

- 速くなるケース
  - ブルームフィルターが適用できない
  - ブルームフィルターで除外される行が少ない
- ディクショナリー共通化の影響
  - メモリ使用量減少
  - ポピュレーション時間延長
- Active Data Guardスタンバイでの動作は未サポート
- 12.2.0では再ポピュレーションが必要

ジョイン・グループ :管理、および、モニタリング

NEW IN 12.2



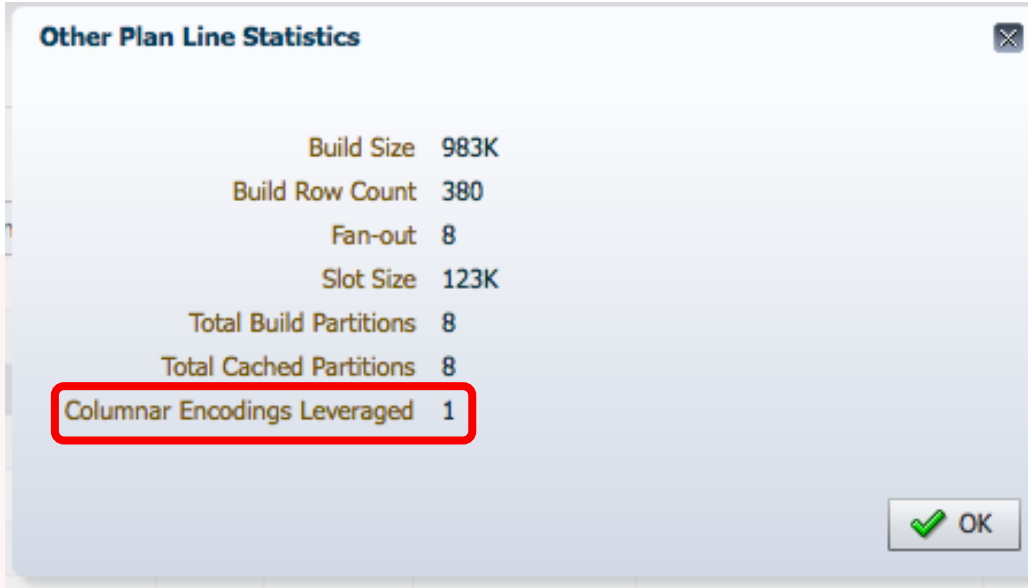


SQLモニターでの確認方法

ジョイン・グループ :管理、および、モニタリング

NEW IN 12.2

- 下記と同等の情報をSQLで取得可能




```
BEGIN
  SELECT PREV_SQL_ID
  INTO   :B_SQLID
  FROM   V$SESSION
  WHERE  SID=USERENV('SID');
END;
SELECT DBMS_SQLTUNE.REPORT_SQL_MONITOR_XML(sql_id=>:B_SQLID).
      EXTRACT(q'#//operation[@name='HASH JOIN']/rwsstats/stat[@id='9']#').
      GETCLOBVAL(2,2) join_group_usage   FROM DUAL;
JOIN_GROUP_USAGE
------------------------------------------------
<stat id="9">1</stat>
```

  - SQLモニターの他の表示情報も同じように取得可能

Real-Time Analytics: より高速な In-Memory ジョイン

NEW IN 12.2

- 12.1 パフォーマンス
  - (Serial)

```
SQL> Select /*+ NO_VECTOR_TRANSFORM */
       sum(lo_extendedprice * lo_discount) revenue
From   LINEORDER l, DATE_DIM d
Where  l.lo_orderdate = d.d_datekey
And    l.lo_discount between 2 and 3
And    l.lo_quantity < 24
And    d.d_date='June 3, 1998';
  2    3    4    5    6    7

            REVENUE
--------------------
       884260172118

Elapsed: 00:00:11.40
```

- 12.2 パフォーマンス
  - (Serial)

```
SQL> Select /*+ NO_VECTOR_TRANSFORM */
       sum(lo_extendedprice * lo_discount) revenue
From   LINEORDER l, DATE_DIM d
Where  l.lo_orderdate = d.d_datekey
And    l.lo_discount between 2 and 3
And    l.lo_quantity < 24
And    d.d_date='June 3, 1998';  2    3    4    5    6    7

            REVENUE
--------------------
       884260172118

Elapsed: 00:00:06.14
```

Create join Group on
Lineorder.lo_orderdate, date_dim.d_datekey

2倍 パフォーマンス高速化

Cursor Duration Temp Table(CDT/CDTT)

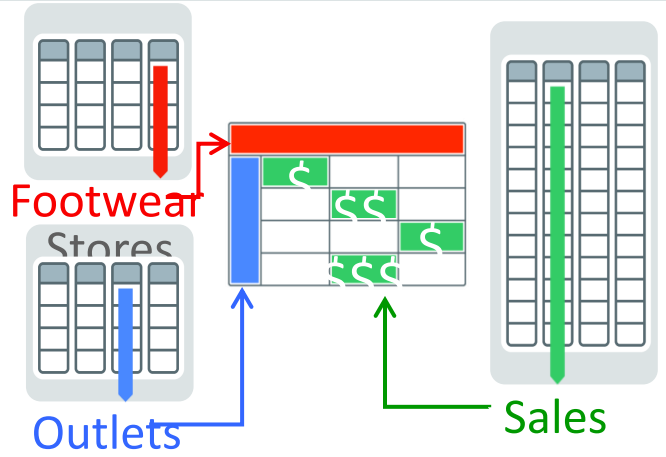
クエリー変換時の一時表が全てメモリー上で完結

Cursor Duration Temp Table

- カーソルが存在する期間のみの一時表の完全インメモリー化
- 実行計画上はTEMP TABLE TRANSFORMATION
  - コストベース変換の一種で以前から存在
- DBIMのVECTOR GROUP BY実行計画の場合に必要になる
- 非DBIMでも例えばスター変換でも使用される
- WITH 句を使用したサブクエリー・ファクタリングでも使われる
- シリアル実行の場合はPGA内に作られる
- パラレル実行の場合は新しい名称MGAと呼ばれるメモリー領域
  - プロセス間で共有される領域
  - PGA_AGGREGATE_TARGETに加算される

一時表

```
-------------------------------------------------------------------------------
| Id  | Operation                                | Name                       |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                            |
|   1 |  TEMP TABLE TRANSFORMATION               |                            |
|   2 |   LOAD AS SELECT(CURSOR DURATION MEM)    | SYS_TEMP_0FD9DADAD_9873DD  |
|   3 |    VECTOR GROUP BY                       |                            |
|   4 |     KEY VECTOR CREATE BUFFERED           | :KV0000                    |
|   5 |      PARTITION RANGE ALL                 |                            |
|   6 |       TABLE ACCESS INMEMORY FULL         | TIME_DIM                   |
|   7 |   LOAD AS SELECT(CURSOR DURATION MEM)    | SYS_TEMP_0FD9DADAE_9873DD  |
|   8 |    VECTOR GROUP BY                       |                            |
|   9 |     KEY VECTOR CREATE BUFFERED           | :KV0001                    |
|  10 |      TABLE ACCESS INMEMORY FULL          | CUSTOMER_DIM               |
|  11 |   HASH GROUP BY                          |                            |
|  12 |    HASH JOIN                             |                            |
|  13 |     HASH JOIN                            |                            |
|  14 |      TABLE ACCESS FULL                   | SYS_TEMP_0FD9DADAE_9873DD  |
|  15 |      VIEW                                | VW_VT_AF278325             |
|  16 |       VECTOR GROUP BY                    |                            |
|  17 |        HASH GROUP BY                     |                            |
|  18 |         KEY VECTOR USE                   | :KV0001                    |
|  19 |          KEY VECTOR USE                  | :KV0000                    |
|  20 |           PARTITION RANGE SUBQUERY       |                            |
|  21 |            TABLE ACCESS INMEMORY FULL    | SALES_FACT                 |
|  22 |     TABLE ACCESS FULL                    | SYS_TEMP_0FD9DADAD_9873DD  |
-------------------------------------------------------------------------------
```



NEW IN 12.2

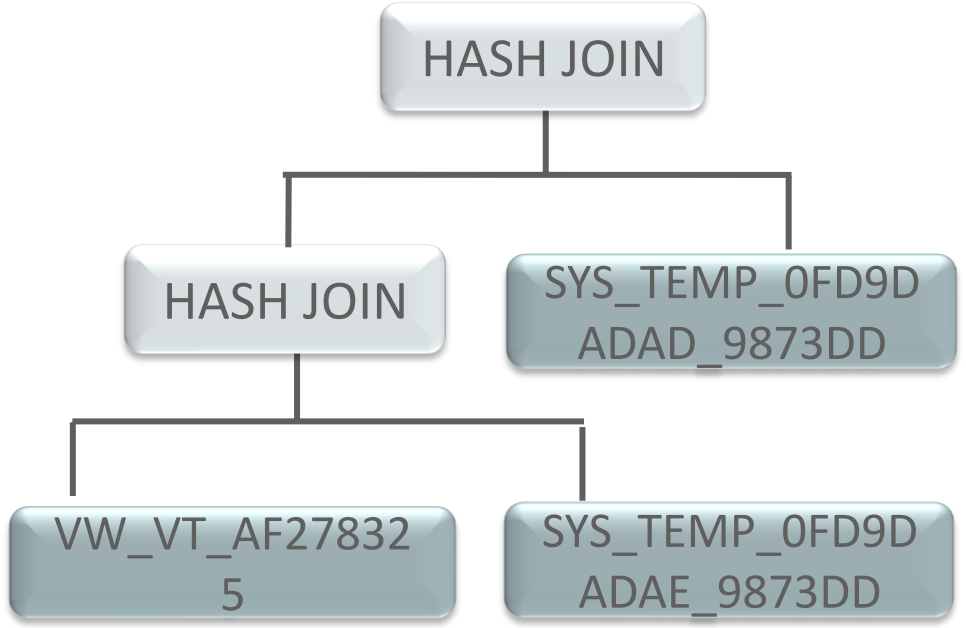
REGION_NAME	DENSE_GROUPING_KEY
Europe	39
North America	17
Asia	91
Africa	84
South America	26

YEAR_NAME	QUARTER_NAME	DENSE_GROUPING_KEY
CY2012	Q1-CY2012	62
CY2012	Q2-CY2012	78
CY2012	Q3-CY2012	36
CY2012	Q4-CY2012	47

In-Memory 一時表 が作成され付加的なペイロード列を格納する。ジョインバックの際参照される。

最終結果の生成

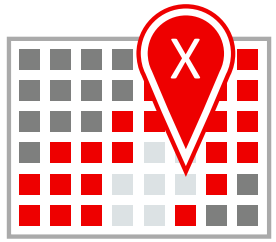
```
-------------------------------------------------------------------------
| Id  | Operation                                | Name                   |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                        |
|   1 |  TEMP TABLE TRANSFORMATION               |                        |
|   2 |   LOAD AS SELECT(CURSOR DURATION MEM)    | SYS_TEMP_0FD9DADAD_9873DD |
|   3 |    VECTOR GROUP BY                       |                        |
|   4 |     KEY VECTOR CREATE BUFFERED           | :KV0000                |
|   5 |      PARTITION RANGE ALL                 |                        |
|   6 |       TABLE ACCESS INMEMORY FULL         | TIME_DIM               |
|   7 |   LOAD AS SELECT(CURSOR DURATION MEM)    | SYS_TEMP_0FD9DADAE_9873DD |
|   8 |    VECTOR GROUP BY                       |                        |
|   9 |     KEY VECTOR CREATE BUFFERED           | :KV0001                |
|  10 |      TABLE ACCESS INMEMORY FULL          | CUSTOMER_DIM           |
|  11 |   HASH GROUP BY                          |                        |
|  12 |    HASH JOIN                             |                        |
|  13 |     HASH JOIN                            |                        |
|  14 |      TABLE ACCESS FULL                   | SYS_TEMP_0FD9DADAE_9873DD |
|  15 |      VIEW                                | VW_VT_AF278325         |
|  16 |       VECTOR GROUP BY                    |                        |
|  17 |        HASH GROUP BY                     |                        |
|  18 |         KEY VECTOR USE                   | :KV0001                |
|  19 |          KEY VECTOR USE                  | :KV0000                |
|  20 |           PARTITION RANGE SUBQUERY       |                        |
|  21 |            TABLE ACCESS INMEMORY FULL    | SALES_FACT             |
|  22 |    TABLE ACCESS FULL                     | SYS_TEMP_0FD9DADAD_9873DD |
-------------------------------------------------------------------------
```



ペイロード列を参照するため一時表へジョイン・バック

高速化その他

ZoneMapが使用可能に



- データ格納情報をユーザー・オブジェクトとして保持
- 12.1.0.2ではExadataのみで使用可能
- 12c R2からインメモリー表にZONEMAP作成が可能に

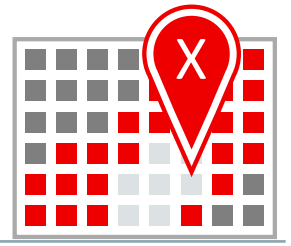
```
CREATE MATERIALIZED ZONEMAP
lineitem_zmap ON LINEITEM
(
  orderkey,
  shipdate,
  receiptdate
);
```

LINEITEM表

ブロック	orderkey	shipdate	receiptdate	destination	quantity
1	1	**1-1-2011**	1-10-2011	San_Fran	100
1	2	1-2-2011	1-10-2011	San_Fran	200
2	3	1-3-2011	1-5-2011	San_Fran	100
2	4	**1-5-2011**	1-10-2011	San_Diego	100
3	5	**1-10-2011**	1-15-2011	San_Fran	100
3	6	1-12-2011	1-16-2011	San_Fran	200
4	7	1-13-2011	1-20-2011	San_Fran	100
4	8	**1-15-2011**	1-30-2011	San_Jose	100

Block Range	min orderkey	max orderkey	min shipdate	max shipdate	min receiptdate	max receiptdate
1-2	1	4	**1-1-2011**	**1-5-2011**	1-9-2011	1-10-2011
3-4	5	8	**1-10-2011**	**1-15-2011**	1-15-2011	1-30-2011

ZoneMapが使用可能に



- 指定されたカラムデータの最小値・最大値の保持
- Zone mapsは 対象外データの除外(pruning,プルーニング)
  - 索引は対象データの検出用途
- In-MemoryでもMIN/MAXインデックスとの組み合わせでさらに高速化
- Attribute Clusteringの併用も効果的

Block Range	min orderkey	max orderkey	min shipdate	max shipdate	min receiptdate	max receiptdate
1-2	1	4	**1-1-2011**	**1-5-2011**	1-9-2011	1-10-2011
3-4	5	8	**1-10-2011**	**1-15-2011**	1-15-2011	1-30-2011

```
-------------------------------------------------------------
| Id  | Operation                                 | Name    |
-------------------------------------------------------------
|   0 | SELECT STATEMENT                          |         |
|   1 |  SORT AGGREGATE                           |         |
|*  2 |   TABLE ACCESS INMEMORY FULL WITH ZONEMAP| SALES4  |
-------------------------------------------------------------
```

```
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - inmemory("QUANTITY_SOLD">11)
       filter(SYS_ZMAP_FILTER('/* ZM_PRUNING */ SELECT zm."ZONE_ID$",
            CASE WHEN BITAND(zm."ZONE_STATE$",1)=1 THEN 1 ELSE CASE WHEN
```

その他12.2 でのパフォーマンス改良点

- ベクター集計がより多くの場合で実行可能に
  - 12.1.0.2では比較的保守的に実行方法を選択し、HASH GROUP BYになる場合が多かった
  - 12.1.0.2 では /*+ VECTOR_TRANSFORM */ヒントをつけてもIMA実行方法にならない場合があった
- インメモリー領域からのサンプリングが可能に
  - ダイナミック・サンプリング(Dynamic Statistics,ダイナミック統計)の高速化
- 基本性能の向上
  - ドキュメントに記載されない小さい改良
- SPARC M7固有最適化

参考資料

その他の製品、コンポーネントとの関係など

- 12.2からDB In-Memoryマニュアルが独立
  - ただしJSONガイド、Data WarehouseガイドにもIn-Memoryの説明あり
- Oracle Text 対応
  - R 表などテキストインデックスにより内部的に作成されるテーブル
- Spatial & Graph オプションのプロパティー・グラフ機能対応
  - 12.2 の新機能
- Sharding のduplicated table未対応

ORA-02584: INMEMORY options not supported for duplicated tables

- DBMS_INMEMORY.REPOPULATE()のパーティション表対応
  - 12.1ではパーティション名指定が必要

Mixed Workload改良点

DML後のメンテナンス

Enhancements for DML Processing In-Memory

New in bundle patches since 12.1.0.2 release

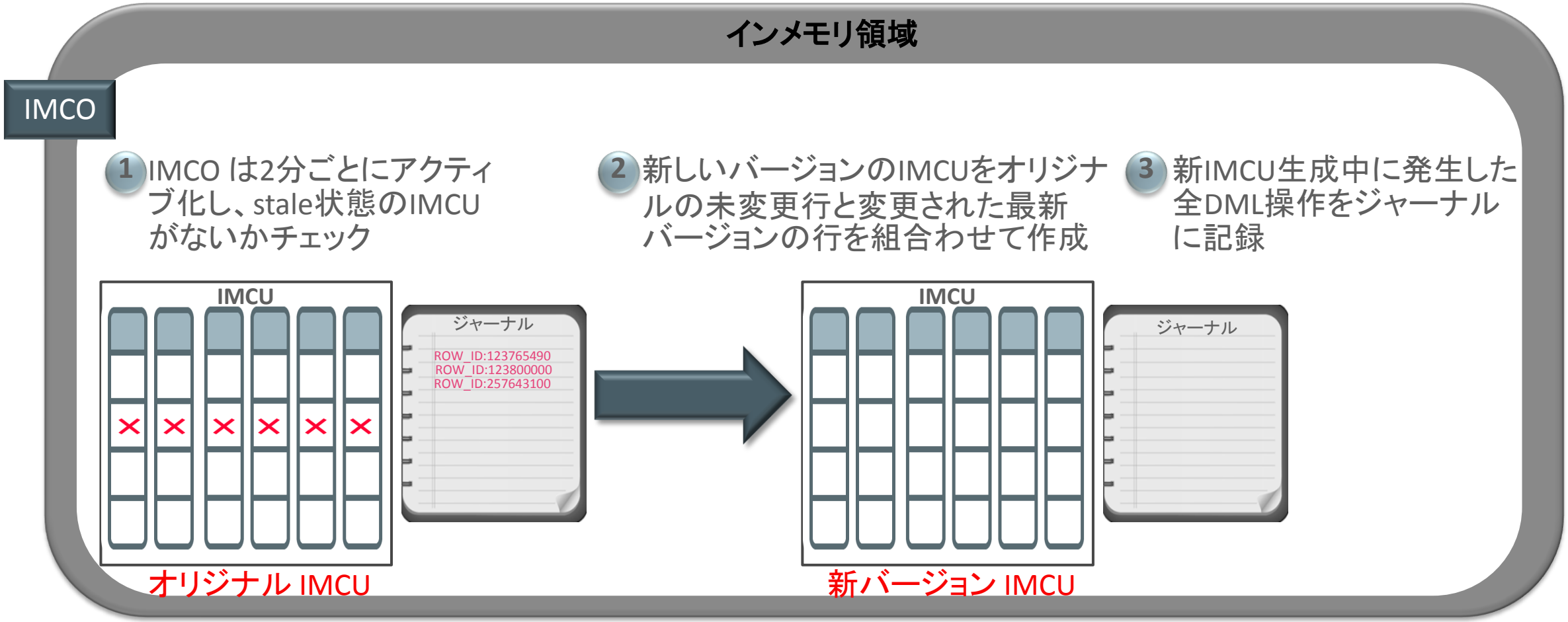
- Double buffering
- Incremental repopulation
- カラムレベルの無効化
- CRクローン
  - SCN毎の有効行リストのビットマップを保持
  - インスタンス統計(V$SYSSTAT)で利用回数などを確認可能
- ガーベッジ・コレクション(Garbage Collection)
- Support for complex predicate push down
  - Direct evaluation on compressed data

カラムレベルの無効化対応

- 12.1.0.2 GA(2016 6月)時点では全て 行 (row)　の粒度での処理
- その後の機能改良によりいくつかの処理の粒度が 列(column)に
- セッション統計によりカラムレベルの処理が発生したか確認可能

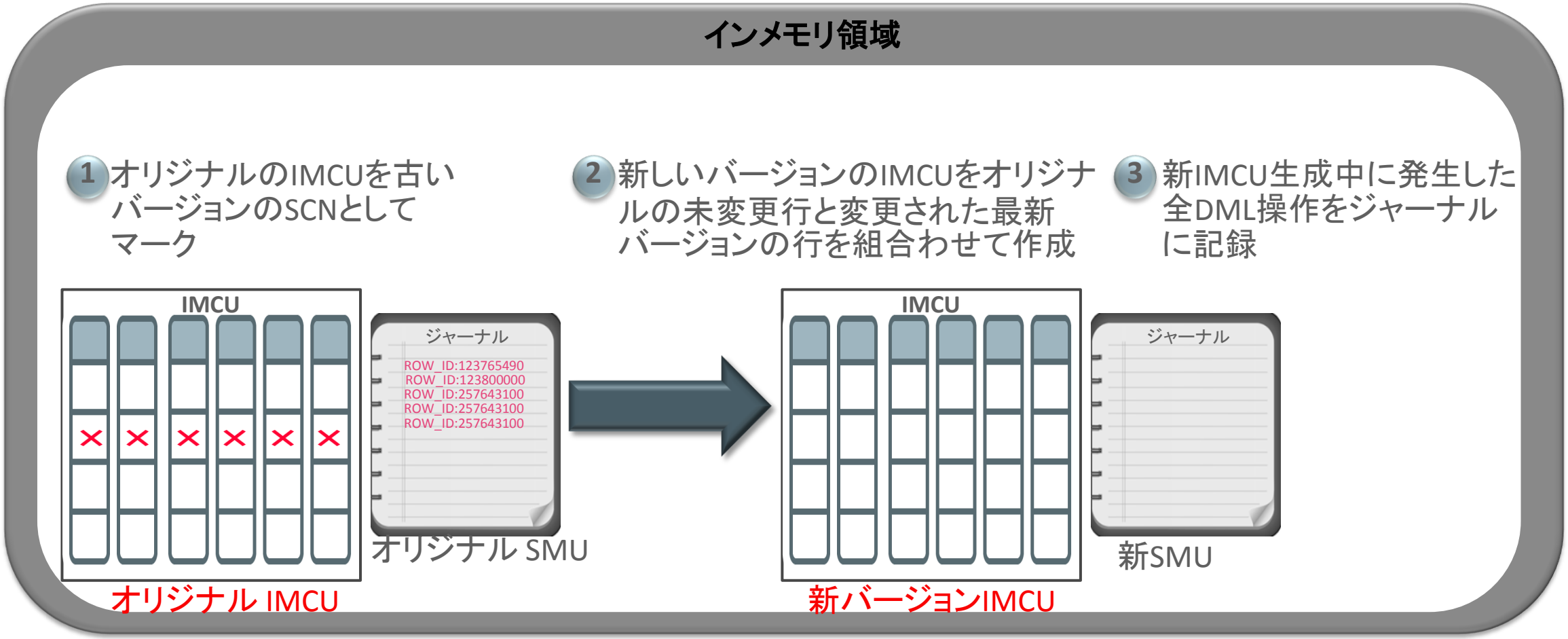
```
SQL> SELECT * FROM v$sysstat WHERE name LIKE '%smucolmap%';

STATISTIC# NAME CLASS VALUE STAT_ID CON_ID
---------- ---- ----- ----- ------- ----------
 709 IM repopulate smucolmap used     128 0 1834967311 0
 710 IM scan smucolmap hit           128 0 4103052601 0
 711 IM scan smucolmap miss due TO inserts    128 0 3403739177 0
 712 IM scan smucolmap miss due TO deletes     128 0 611634101 0
 713 IM scan smucolmap miss due TO invalid blocks   128 0 2271019354 0
```

トリクル再ポピュレーションとインメモリ・カラムストア



トリクル再ポピュレーションが使えるCPUリソースの割合は初期化パラメータ
INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENTで指定

ダブル・バッファリングとインメモリ・カラムストア



両バージョンのIMCUは、オリジナルが2度と利用されない、または領域が逼迫するまで、IMカラムストア内に保持される

本編の補足

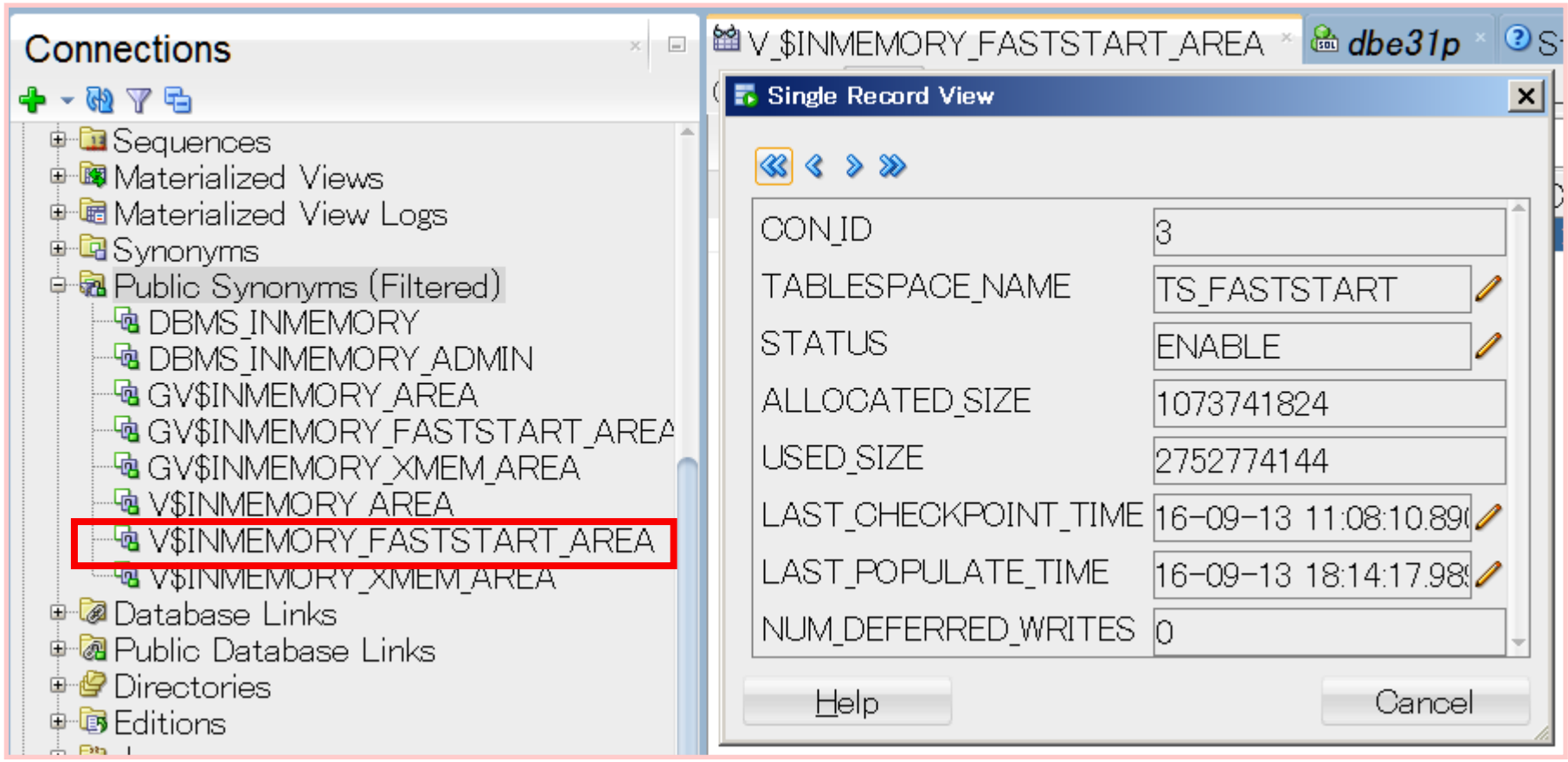
新規セッション統計
新規インスタンス統計
関連するビュー

12c R2 でのセッション統計: SIMD 関連が見れるように

- 12.1.0.2 ではSIMDが使用されたかを確認するのは非常に困難
- SPARC M7 固有のものも追加
  - 以下で "HW" とあるもの

IM simd compare calls	6232
IM simd compare HW offload calls	6227
IM simd decode calls	966
IM simd decode HW offload calls	891
IM simd decode unpack calls	1
IM simd rle burst calls	2084
IM simd rle burst HW offload calls	2074

FASTSTART 関連ビュー



FASTSTART 関連インスタンス統計

- 代表的なもの

```
IM faststart read CUs requested                                  1368
IM faststart read CUs                                               0
IM faststart read CUs not accessible                                0
IM faststart read bytes                                             0
IM faststart read CUs incompatible                                  0
IM faststart read CUs invalid                                       0
IM faststart write bytes                                  2.6759E+10
IM faststart write CUs                                            1489
IM faststart write CUs requested                                  1575
IM faststart write CUs problems                                     0
IM faststart write accumulated time (ms)                      14172391
IM faststart write CUs encryption mismatch                          0
IM faststart write CUs evicted                                      0
IM faststart write CUs too dirty                                    9
IM faststart write deferred CUs requested                         1890
IM faststart write deferred CUs                                   1489
IM faststart write deferred accumulated (ms)                  18096354
```

自動データ最適化のインメモリ対応

NEW IN 12.2

ポリシー関連のビュー

- ILM ポリシー関連のビュー:
  - DBA|USER_ILMDATAMOVEMENTPOLICIES – ADOがポリシーに従い移動したデータの情報を表示
  - DBA|USER_ILMEVALUATIONDETAILS – ADOのポリシー評価に関する詳細情報を表示
  - DBA|USER_ILMOBJECTS – ADOの全ポリシーと全オブジェクトを表示
  - DBA_ILMPARAMETERS – ADOに関連するパラメータを表示
  - DBA|USER_ILMPOLICIES – ADOポリシーの詳細を表示
  - DBA|USER_ILMRESULTS – ADOのデータ移動ジョブに関する情報を表示
  - DBA|USER_ILMTASKS – ADOタスクに関する詳細を表示

自動データ最適化のインメモリ対応

NEW IN 12.2

新しいビュー

- ADOポリシーによって実行されるインメモリ・タスクの追跡するために:
  - v$im_adotasks – 自動モードのタスク一覧
  - v$im_adotaskdetails – 自動モードのタスク詳細
  - v$im_adoelements – 自動モードの評価指標一覧

自動データ最適化のインメモリ対応

NEW IN 12.2

ADO IM ポリシーの例

- ポリシーの手動実行:

```
declare
  v_executionid number;
begin
  dbms_ilm.execute_ilm (
    owner=>'SSB',
    object_name=>'LINEORDER',
    execution_mode=>dbms_ilm.ilm_execution_offline,
    task_id=>v_executionid);
end;
/
```

In-Memory Expressions 監視

NEW IN 12.2

新しい統計

- 全般的にCU関連の統計と同じものに準じる
  - IM populate EUs ...
  - IM prepopulate EUs ...
  - IM repopulate EUs ...
  - IM scan EUs ...

In-Memory Expressions 監視

NEW IN 12.2

新しい統計

- よく利用される統計:
  - IM scan EU bytes in-memory
  - IM scan EU bytes uncompressed
  - IM scan EU rows
  - IM scan EUs columns accessed
  - IM scan EUs columns theoretical max
  - IM scan EUs memcompress for query low
  - IM scan EUs split pieces

In-Memory Expressions 監視

NEW IN 12.2

新しいビュー

- v$imeu_header
  - IMEUに関わるセグメント情報を提供
- v$im_imecol_cu
  - IMEU内に保存されている演算式の詳細情報を提供

リファレンス
マニュアル・ドキュメント

- Oracle Database In-Memory ガイド, 12*c* リリース 2 (12.2)
  - 全般

    http://docs.oracle.com/cd/E82638_01/INMEM/toc.htm
- Oracle Database JSON Developer's Guide, 12*c* Release 2 (12.2)
  - **In-Memory JSON Data**

    http://docs.oracle.com/cd/E82638_01/ADJSN/in-memory-json-data.htm#ADJSN-GUID-857A2E95-28AB-4314-B654-4CB05C50D9CF

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Integrated Cloud

Applications & Platform Services


ORACLE®