
ORACLE®

Oracle Database 12c Release 2 CoreTech Seminar

12.2.0.1
コア機能

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. パーティション
2. Analytic View
3. 近似値計算
4. パフォーマンス/Optimizer

Agenda

1. パーティション
2. Analytic View
3. 近似値計算
4. パフォーマンス/ Optimizer

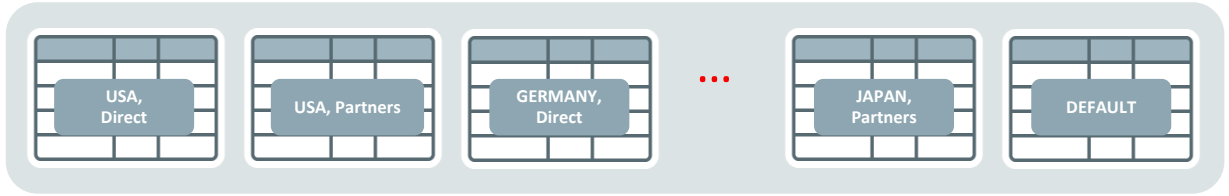
Oracleのパーティション新機能追加の実績

Oracle 8.0 から Oracle 12c R1まで

	コア系機能	パフォーマンス	管理性
Oracle 8.0	Range partitioning Global Range indexes	Static partition pruning	基本メンテナンス機能: ADD, DROP, EXCHANGE
Oracle 8*i*	Hash partitioning Range-Hash partitioning	Partition-wise joins ダイナミックパーティションプルーニング	Expanded maintenance: MERGE
Oracle 9*i*	List partitioning		Global index maintenance
Oracle 9*i* R2	Range-List partitioning	高速パーティションSPLIT	
Oracle 10*g*	Global Hash indexes		Local Index maintenance
Oracle 10*g* R2	表あたり100万パーティション	Multi-dimensional pruning	高速 DROP TABLE
Oracle 11g	Virtual column based partitioning コンポジットの組み合わせの多様化 Reference partitioning		- Interval partitioning - Partition Advisor - インクリメンタルoptimizer統計
Oracle 11g R2	Hash-* partitioning Expanded REF partitioning	“AND” pruning	Multi-branch execution
Oracle 12c R1	Interval-REF partitioning	- Partition Maintenance on multiple partitions - Partial local and global indexes	- Asynchronous global index maintenance for DROP/TRUNCATE - Online partition MOVE - Cascading TRUNCATE/EXCHANGE

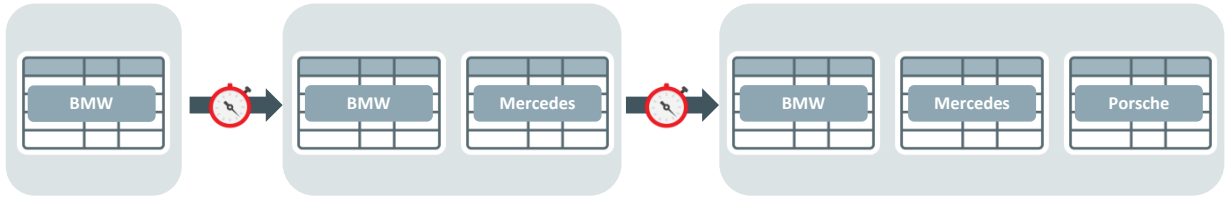
Oracle Partitioning:
3種の新規パーティション方法

NEW IN 12.2

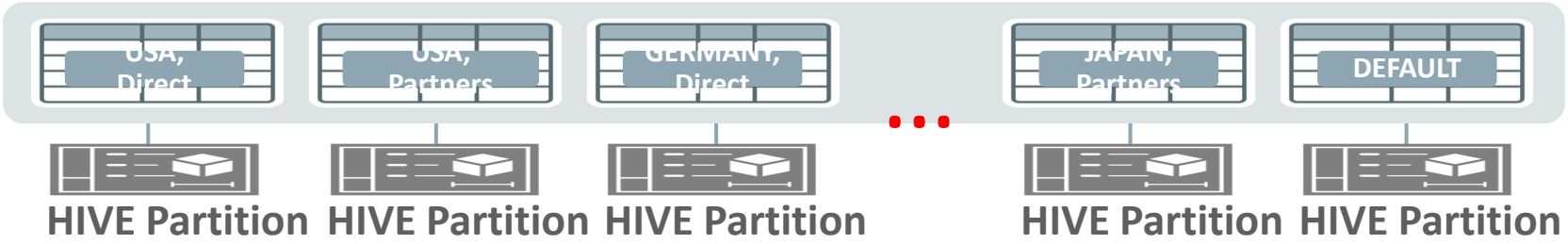


複数列リスト [sub] Partitioning

More than one column as partition key
Ideal for accessing HIVE partitioned tables



自動リストパーティショニング



外部表のパーティショニング

自動リストパーティション

Auto-List Partitioning

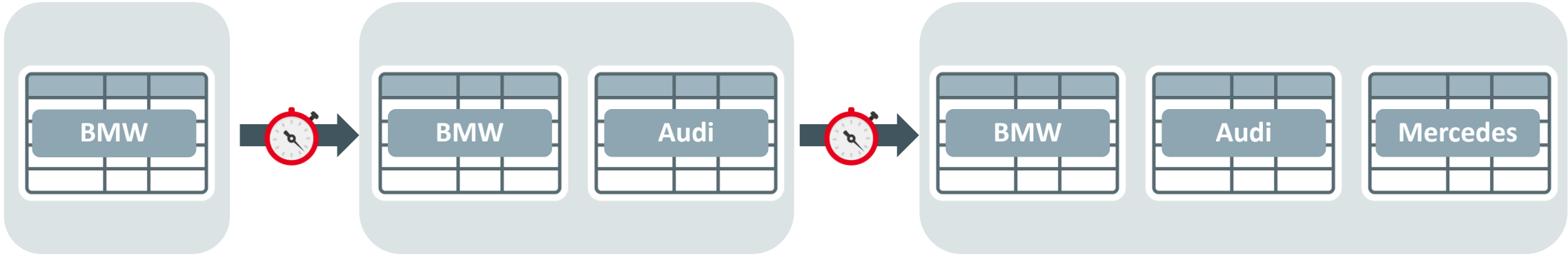
手動 vs 自動、範囲 vs 値

	手動設定	自動
値の範囲	レンジ	インターバル(*)
値	リスト	自動リスト

*) 範囲は等間隔という制限あり

Auto-List Partitioning

NEW IN 12.2



- パーティションキーに指定した列に新規のユニークな値が入るとパーティションが自動で追加される
  - 旧来のリストパーティションの機能はそのまま維持
  - 既存のリストパーティションを”自動リスト”にEVOLVE（進化、発展、変換）させることも可能

Auto-List Partitioning
構文サンプル

```
CREATE TABLE sales_auto_list (
  salesman_id NUMBER(5),
  salesman_name VARCHAR2(30),
  sales_state VARCHAR2(20),
  sales_amount NUMBER(10),
  sales_date DATE )
  PARTITION BY LIST (sales_state) AUTOMATIC
    ( PARTITION P_CAL VALUES ('CALIFORNIA') )
```

リストパーティション宣言時に
AUTOMATIC句を宣言する

Auto-List Partitioning

詳細・制約

- テーブル作成時に最低でも1つのパーティションが必要
  - インターバルパーティションと同様
- *_PART_TABLESの「AUTOLIST 」列よりAuto-List Partitionが設定されているのか確認可能(*はALL、DBA、USER)
- サブパーティションには設定できない

> ORA-14160: この物理属性は表のサブパーティションに指定できません。

- 最大パーティション数: 1,000,000

Auto-List Partitioning

詳細・制約

- 新規で作成されるパーティション名はシステム側で決定される(SYS_P***)
  - 特定のパーティションを指定するには "FOR VALUES" 句使用
- 通常のList Partition表をAuto-List表に変更することは可能
- DEFAULT パーティションがある表には設定できない
  - CREATE時にDEFAULTを設定
  - DEFAULTがあるリストパーティション表をAutoにする場合

ORA-14851: AUTOLIST[サブ]パーティション・オブジェクトには DEFAULT[サブ]パーティションを指定できません。

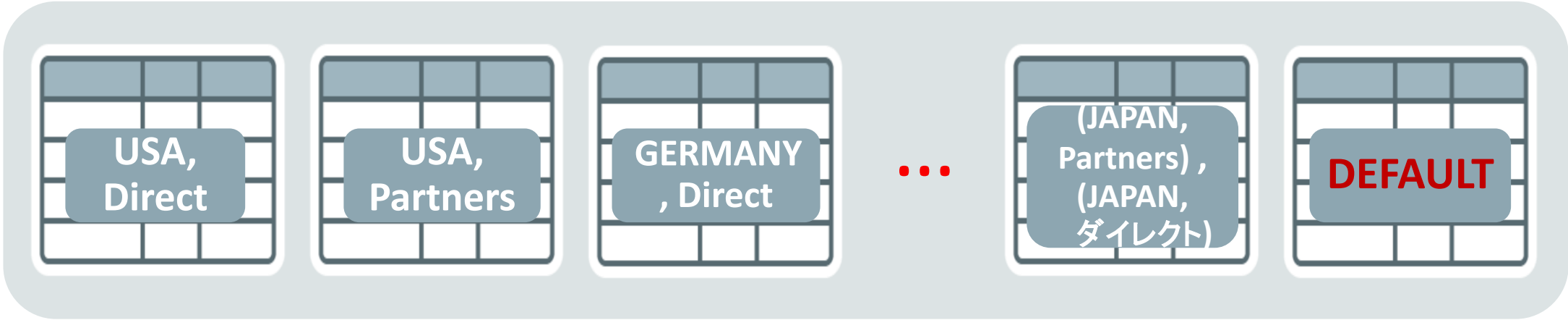
ORA-14852: SET [SUB]PARTITIONING AUTOMATICは この表では無効です。

- 新規のパーティションへのローカル索引は自動で作成される(USABLE状態で)

複数列リストパーティション

Multi-Column List Partitioning

Multi-Column List Partitioning



[概要]

複数の列の組み合わせを指定したリスト・パーティションの作成が可能。

[メリット]

より複雑な業務モデルに対応できるようになる。

Multi-Column List Partitioning

構文サンプル

```
CREATE TABLE sales ( region VARCHAR2(50),
channel VARCHAR2(50), …)
PARTITION BY LIST (region, channel)
( partition p1 values (‘USA’,’Direct’),
partition p2 values (‘USA’,’Partners’),
partition p3 values (‘GERMANY’,’Direct’),
…
partition p44 values ((‘Japan’,’Partners’),(‘Japan’,’W                ));
partition p45 values (DEFAULT));
```

リストパーティション
宣言時に
2つ以上のキー列を指定

ひとつの列の値のみを設定して
デフォルトパーティションを
設定できない

NG例)　partition p46 values
(‘JAPAN’,**DEFAULT**)

Multi-Column List Partitioning

詳細・制約など

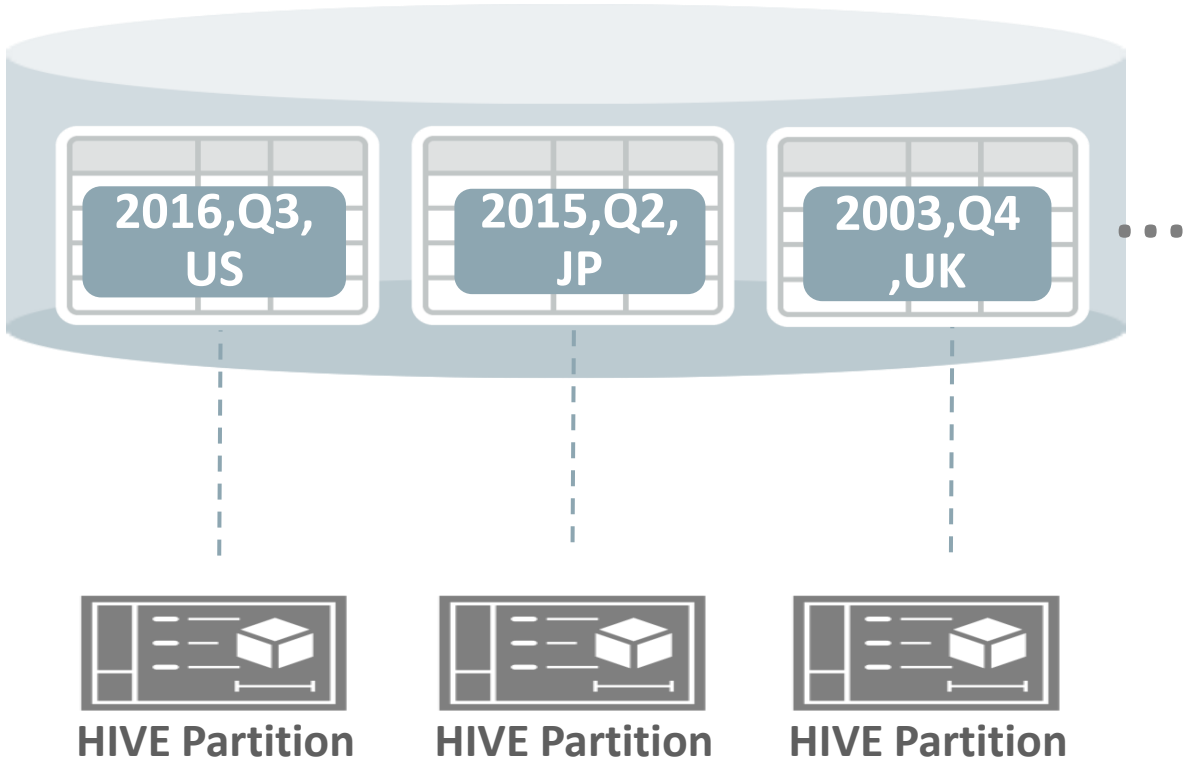
- リスト-リストとは異なるパーティション方法
- 複数カラムで選択可能なキー(列)値は最大で16個まで
  - 実行時エラー: ORA-14014: パーティション化列の最大数は16です。
- サブパーティションにも設定可能
- ヒープ表、及び外部表に設定可能
- 参照パーティション、Auto-Listパーティションに対応

外部表のパーティション化

Partitioned External Table

New in 12.2　外部表のパーティション化

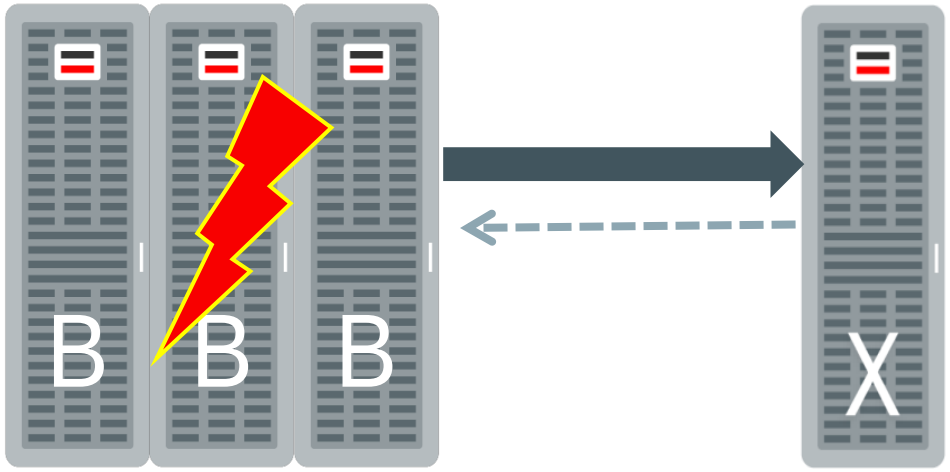
NEW IN 12.2

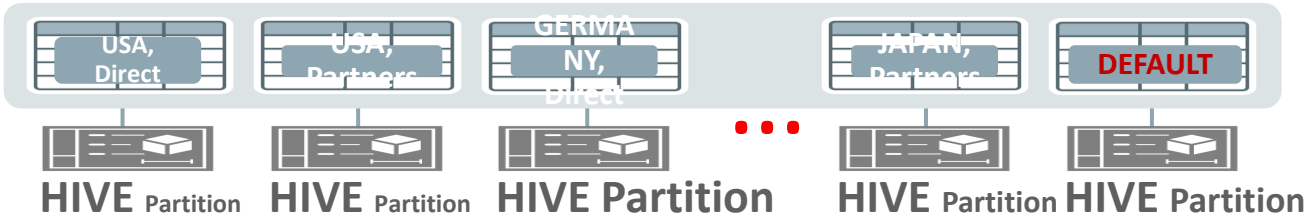


- パーティション化された Hive, HDFS表を Oracleエコシステムにマッピング
- Oracle Partitioning機能を外部の HDFSベースデータストアに 対しても適用
  - パーティションプルーニングとパーティションメンテナンス
  - Big Data に対応したオプティマイザ
- 高速化

Partitioned External Tables

NEW IN 12.2





- いくつかのパーティション化方法をサポート。Range, List
  - 外部表がHiveの場合はMulti-Column Partitionが適する場合が多くなると想定
- Partition pruning、および、いくつかの Partition Maintenanceオペレーションをサポート
  - ADD partition, DROP partition

Partitioned External Table

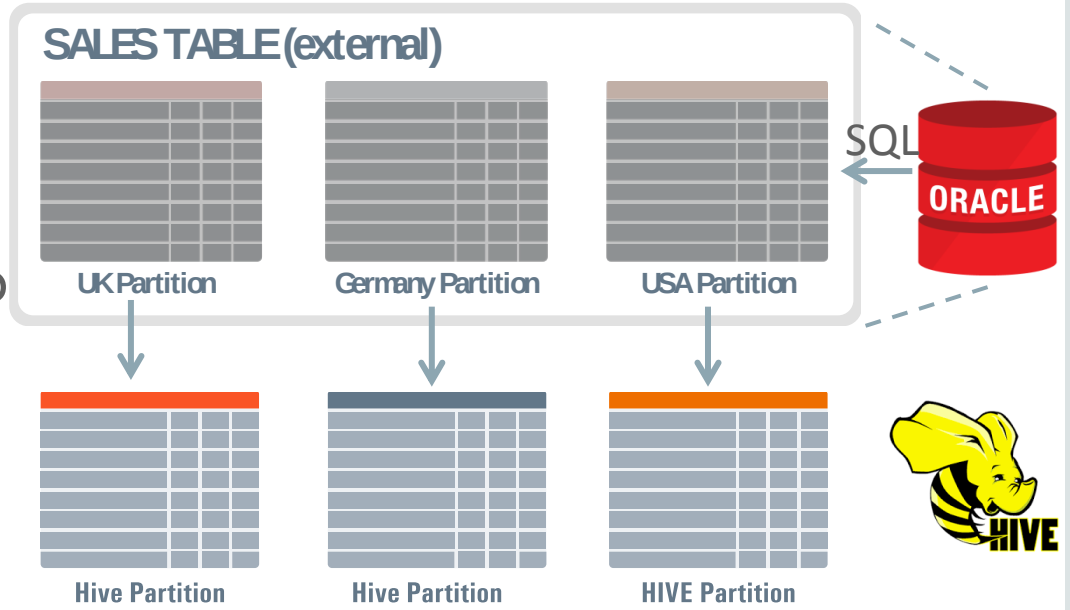
構文サンプル: ローカルファイルシステムのcsvファイル例

```
CREATE TABLE pet (col1 number, col2 number)
organization external(
TYPE oracle_loader
DEFAULT DIRECTORY d1
ACCESS PARAMETERS  ( records delimited by newline characterset us7ascii
nobadfile    logfile l1:'pet.log'    fields terminated by ","   ))
REJECT LIMIT unlimited
partition by range (col1)(
partition p1 values less than (1000) location ('file1.txt'),
partition p2 values less than (2000) default directory d2 location ('file2.txt'),
partition p3 values less than (3000) location ('file3.txt','file4.txt'));
```

LOCATION句はパーティション単位で記載する。
※サブパーティションがある場合はサブパーティション単位で記載

LOCATION句には複数のファイルを選択可能

Partitioned External Table
詳細・制約



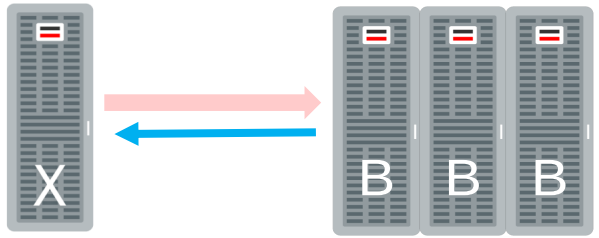
- パーティションキーの値に実際の外部表に含まれるデータが則しているかどうかのチェックはしない
- リスト,レンジのコンポジットパーティション設定可能
- ハッシュ、自動系(Auto-list、Interval)、参照パーティションは非対応
- NOT NULL、UNIQUE、PRIMARY KEY、FOREING KEYの設定が可能
  - 制約がRELY DISABLE に設定されている場合のみ
- DROP PARTITIONを実行しても外部のデータは削除されない
- MODIFY、MERGE、SPLIT、MOVE、EXCHANGE 未サポート
- LOCATION句はパーティション/サブパーティション単位で記載する
  - LOCATION句が省略された場合は空のパーティションが作成される
- デフォルトディレクトリはパーティション単位でも記載可能

実行計画ノート部分より

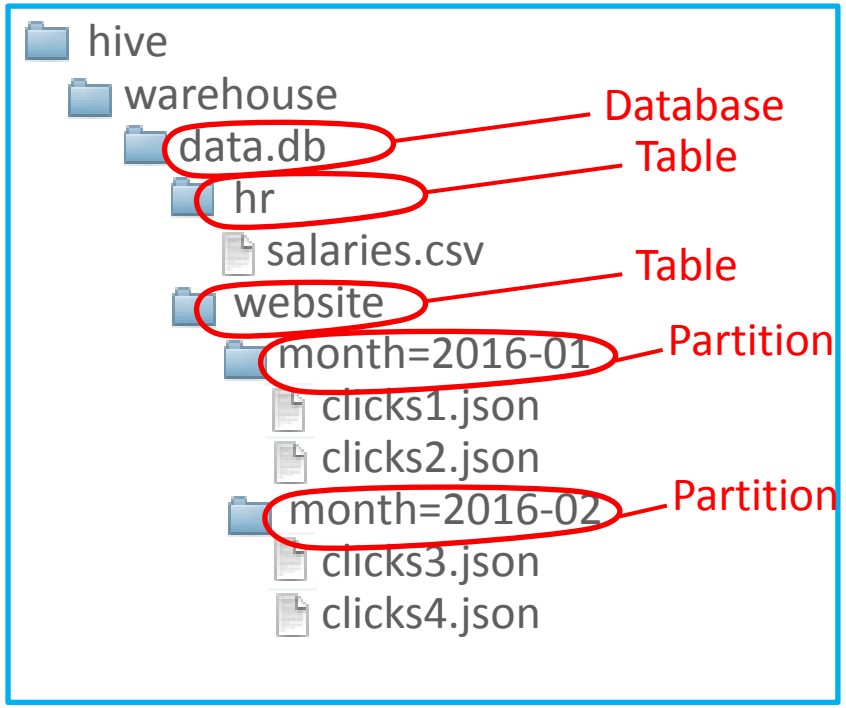
```
Note
-----
   - rely constraint used for this statement
```

DBMS_HADOOP パッケージ

- Hive,HDFSを外部パーティション表にする場合DBMS_HADOOPを使用
- 12.2からデフォルトでロードされる
- Big Data SQL の一部
- PL/SQLパッケージマニュアルに詳細




```
declare
 ddltxt varchar2(4000);
begin
 DBMS_HADOOP.create_extddl_for_hive(
        cluster_id=>'hadoop_cl_1',
        db_name=>'default',
         hive_table_name=>'website',
         hive_partition=>true,
         table_name=>'hive_pet_xt11',
         perform_ddl=>true,
         text_of_ddl=>ddltxt
   );
 dbms_output.put_line(ddltxt);
end;
```



パーティション - その他

オンライン、Partition Exchange用の表作成 など

オンライン操作

[概要]

Online状態でパーティション表に対して実行可能なDDL(SPLIT)が追加

[メリット]

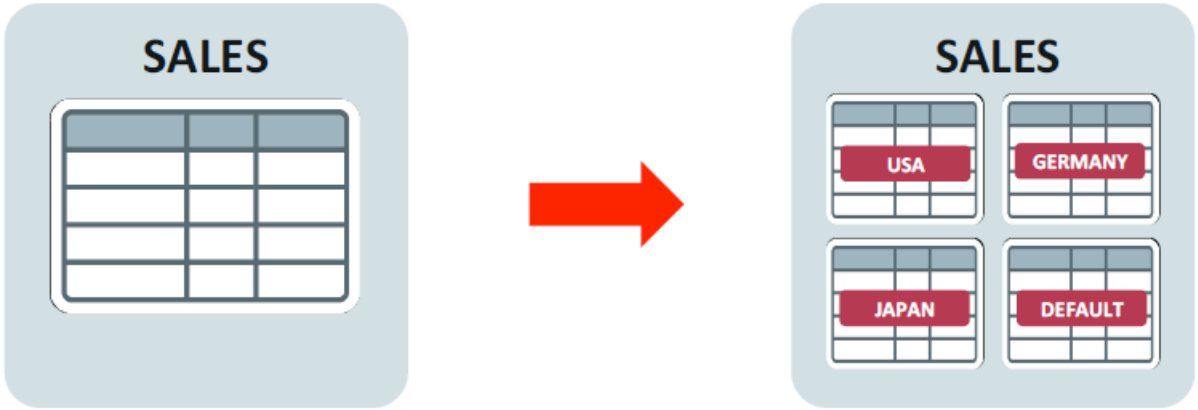
発行中のDMLに影響を与えることなくパーティションのメンテナンスが可能

```
ALTER TABLE EMP
  SPLIT PARTITION p_all
  INTO
   ( PARTITION p_manager VALUES ('MANAGER'),
     PARTITION p_president) ONLINE;
```

非パーティション表をパーティション化

[概要]

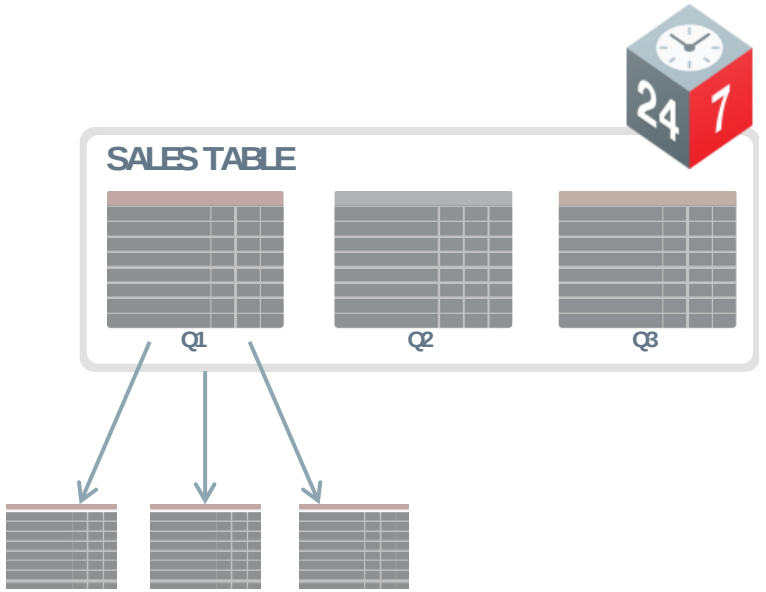
標準的SQLで非パーティション表をパーティション表に変更可能になった。




```
CREATE TABLE sales ( order_num NUMBER,
region VARCHAR2 (10), … );

ALTER TABLE sales MODIFY
PARTITION BY LIST (region)
(partition p1 values (‘USA’),
partition p2 values (‘Germany’),
partition p3 values (‘Japan’),
partition p4 values (DEFAULT)) ONLINE;
```

Conversion to Partitioned Table

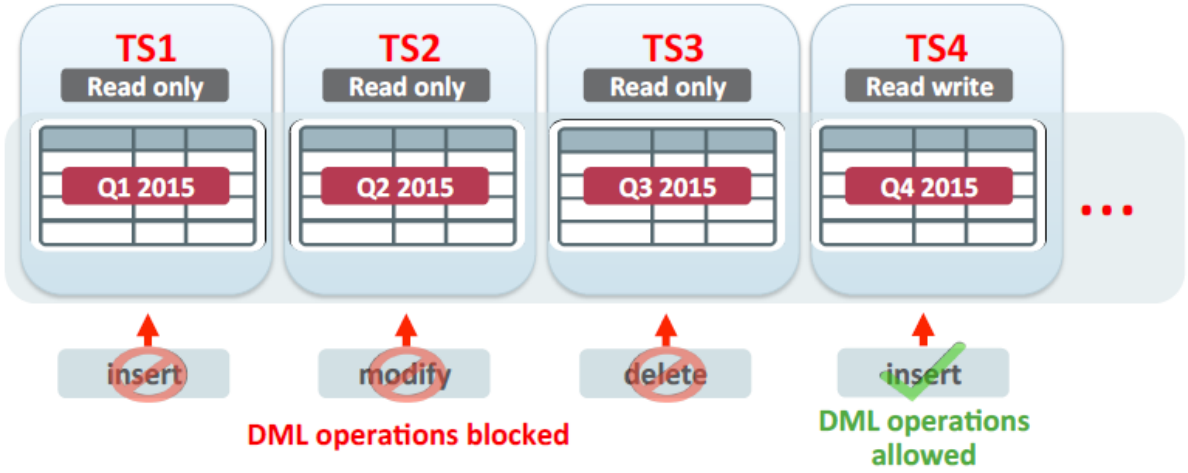
詳細・制約



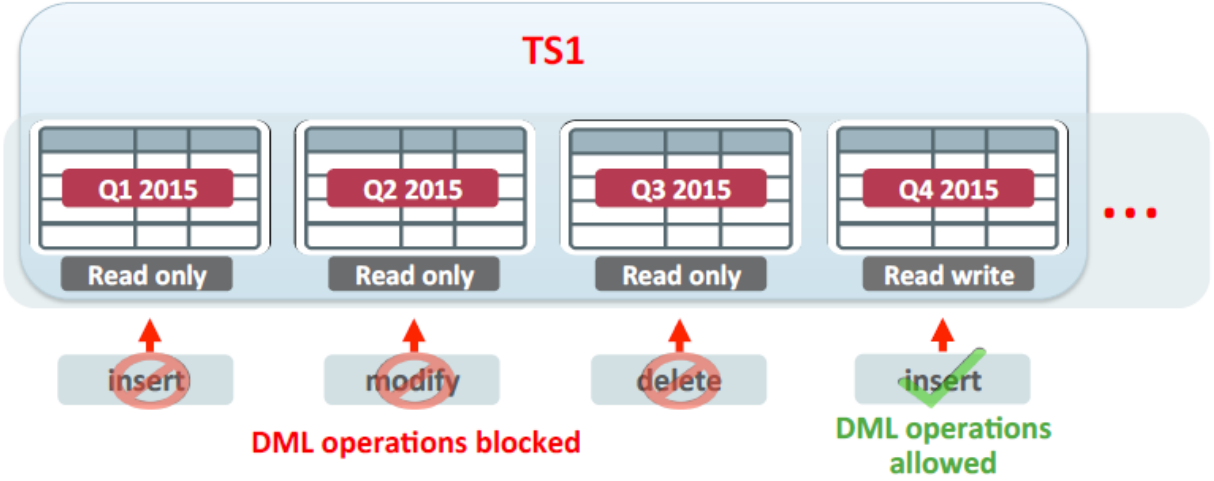
- DDLにONLINE句やUPDATE INDEX句を指定可能
  - ONLINEは"インライン"変換ではなく作業領域が必要
- UPDATE INDEX句を宣言しなかった場合のデフォルト動作
  - パーティション定義とキー(列)が同じ索引 ⇒ ユニークローカル索引
  - パーティション定義とキー(列)が異なる索引
    ⇒ グローバル(非パーティション)ユニーク索引
  - ビットマップ索引 ⇒ ローカル索引
  - グローバルパーティション索引 ⇒ グローバルパーティション索引(変わらない)
- UPDATE INDEX の注意事項
  - 索引の属性(ユニーク等)を変更できない
  - 表領域を指定しない場合の索引のデフォルト表領域
    - コンバート後ローカル索引になる索引：パーティションと同じ表領域
    - コンバート後グローバル索引になる索引：定義変更前のグローバル索引と同じ表領域

リード・オンリー パーティション

- 12c R1以前: 読み取り専用表領域



- 12c R2: 読み取り専用パーティション



ORA-14466:読取り専用のパーティションまたはサブパーティション内のデータは変更できません。

- サブパーティション対応

Read Only Partition
構文サンプル - レンジパーティションの例

```
CREATE TABLE orders ( order_id number,
order_date DATE, … ) READ WRITE
PARTITION BY RANGE(order_date)
( partition q1_2015 values less than (‘2014-10-01’) READ ONLY,
partition q2_2015 values less than (‘2015-01-01’) READ ONLY,
partition q3_2015 values less than (‘2015-04-01’),
partition q4_2015 values less than (‘2015-07-01’)
);
```

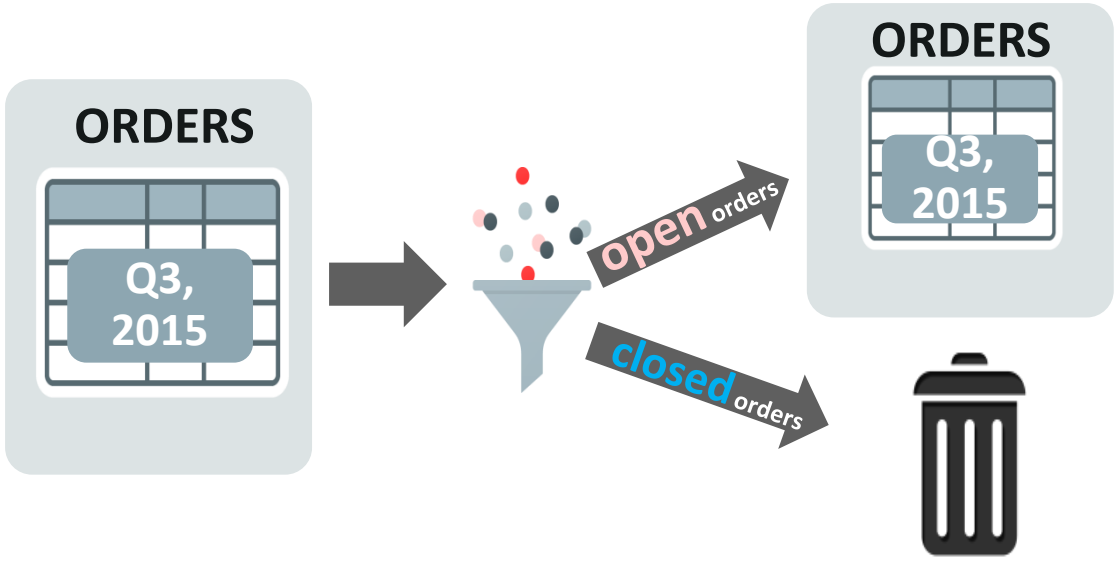
Read Only Partition

詳細・制約

- *_PART_TABLES,、*_TAB_PARTITIONS、*_TAB_SUBPARTITIONS に READ_ONLY 列が追加された(*はALL、DBA、USER)
- Read-only ステータスを設定したパーティションにサブパーティションがある場合は設定がカスケードされる
- データの中身に影響しないDDLは実行可能
  - カラム追加 (ALTER TABLE … ADD …)
  - 圧縮
  - VARCHAR2(32) -> VARCHAR2(64)
- SPLITした場合は元表の設定が引き継ぐ
  Read-Only ⇒ Read-Only + Read-Only
  Read-Write ⇒ Read-Write + Read-Write

⇐ **読み取り専用表領域との大きな違い**

フィルター付パーティションメンテナンス操作



“パーティションを低コストディスクに圧縮して再配置したい。中のデータもきれいにしたい。1年以上前のオーダーは”オープン”ステータスなもの以外は消したい。”

- Combines data maintenance with partition maintenance
  - パーティション形状変更
  - パーティション属性変更
  - パーティションのデータ変更
- パーティションメンテナンスとデータのメンテナンスが同時に行える
- Available for all online and offline partition maintenance operations
  - MOVE, MERGE, SPLIT

NEW IN 12.2

Filtered Partition Maintenance Operation
構文サンプル

```
ALTER TABLE orders MOVE PARTITION q3_2015
TABLESPACE archive
INCLUDING ROWS WHERE order_state = 'open';
```

Filtered partition maintenance operations

詳細・制約

- ONLINE句を指定可能
- MOVE,SPLIT,MERGE,MODIFY操作をサポート
- フィルタリングはDDL発行時のSCNイメージに対して実行。
  フィルタリングしながらDDL(Online)を発行している最中にDMLによってフィルタ条件に該当するデータが入ってもフィルタの対象にならない
- DDL中にパーティションの条件にさえ当てはまっていれば、フィルタリング条件外のデータが挿入される
- DDL中に既存のデータが途中でフィルタの条件にマッチしてもそれは反映されない

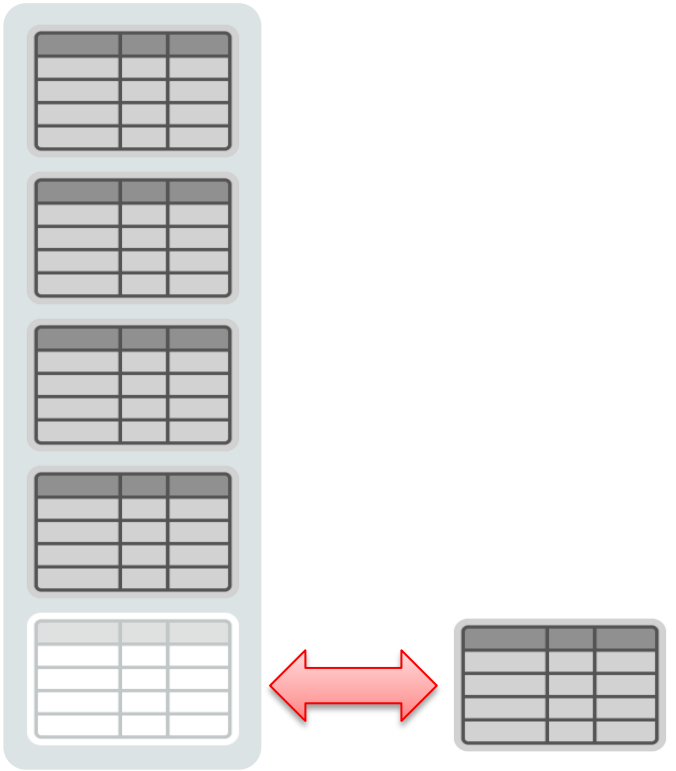
後々PARTITION EXCHANGEするための非パーティション表を簡単に作成

[概要]

EXHANGE目的で表を作成する場合、「FOR EXCHANGE WITH」句をつけて表を作成可能

[メリット]

CTAS(CREATE TABLE AS SELECT)より確実にEXCHANGE目的の表が作成可能



```
CREATE TABLE sales_exchange
TABLESPACE my_sales_tblspace
FOR EXCHANGE WITH TABLE sales;
```

Create Table for Exchange

詳細・制約

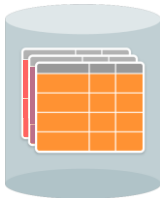
- 索引はコピーされない
- コピー元の表の列名・列の順番・属性を引き継ぐ
- コピー元の列の以下の属性を引き継ぐ

⇦CTASでは不可能

  - unusable columns, invisible columns, virtual expression columns, functional index expression columns
- 「FOR EXCHANGE WITH」宣言時の同時にパーティションの宣言が可能
- 「FOR EXCHANGE WITH」宣言時の同時にセグメント属性の指定が可能
  - 表領域、PCTFREE等

Oracle Partitioning in Oracle Database 12.2

更に進化



	コア機能	パフォーマンス	管理性
Oracle 10g R2 以前	Range partitioning, Hash partitioning, Local and global Range indexing, Range-Hash partitioning, List partitioning, Range-List partitioning, Global Hash indexing, 1M partitions per table	Static partition pruning, Partition-wise joins, Dynamic partition pruning, Fast partition SPLIT, Multi-dimensional pruning	Basic maintenance: ADD, DROP, EXCHANGE, MERGE Global index maintenance, Local Index maintenance, Fast DROP TABLE
Oracle 11g	Virtual column based partitioning, コンポジット組み合わせ可能種類増加 Reference partitioning, Hash-* partitioning, Expanded Reference partitioning	“AND” pruning	Interval partitioning, Partition Advisor, Incremental stats management, Multi-branch execution (aka table or-expansion)
Oracle 12c R1	Interval-Reference partitioning	Partition Maintenance on multiple partitions, Asynchronous global index maintenance	Online partition MOVE, Cascading TRUNCATE, Partial indexing,
Oracle 12c R2	- 自動リスト - 複数列リスト - 外部表	- Online partition maintenance operations - Online table conversion to partitioned table - DDLによるカーソル無効化をオプションにより抑制可能に	- Filtered partition maintenance operations - Read only partitions - Create table for exchange

Agenda

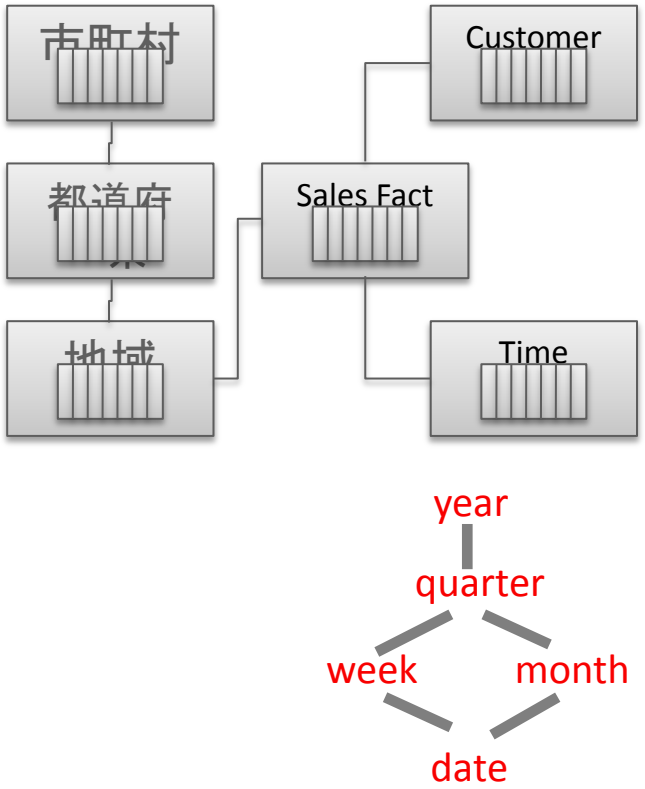
1. パーティション
2. Analytic View
3. 近似値計算
4. パフォーマンス/ Optimizer

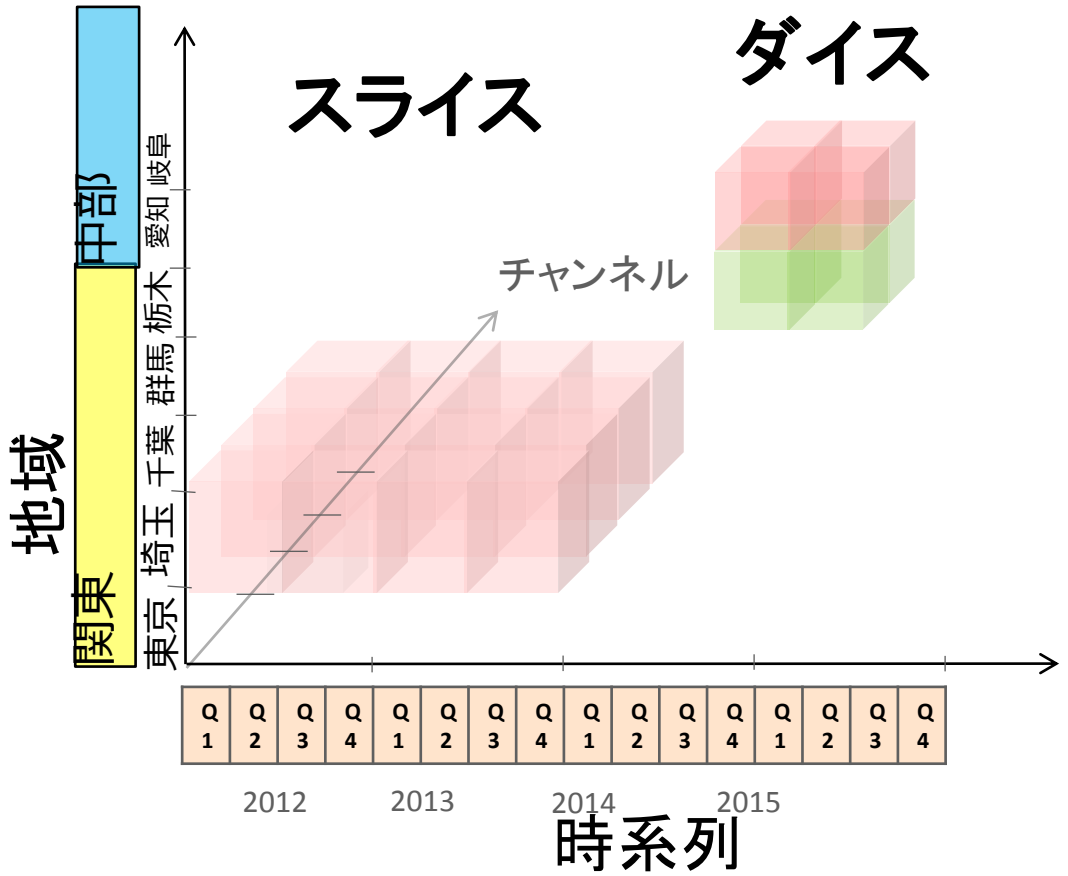
New in 12.2 Analytic View

NEW IN 12.2

- OLAP, BI, スタークエリーなどで出てくるコンセプト、ビジネスロジックをデータベースに取り込む
  - 集計(Aggregations)
  - 階層(Hierarchy,ハイアラーキー)
  - 計算(Calculation)
- 簡単なSQLで複雑な分析が可能に
  - JOIN と "GROUP BY" が不要に
- 基にするテーブル、ビューに制限無
  - 外部表(含むBigData), In-Memoryなど可
  - "Analytic View" 自体は通常VIEWと同様にストレージ領域を使わない
- APEXのサンプルアプリも提供予定

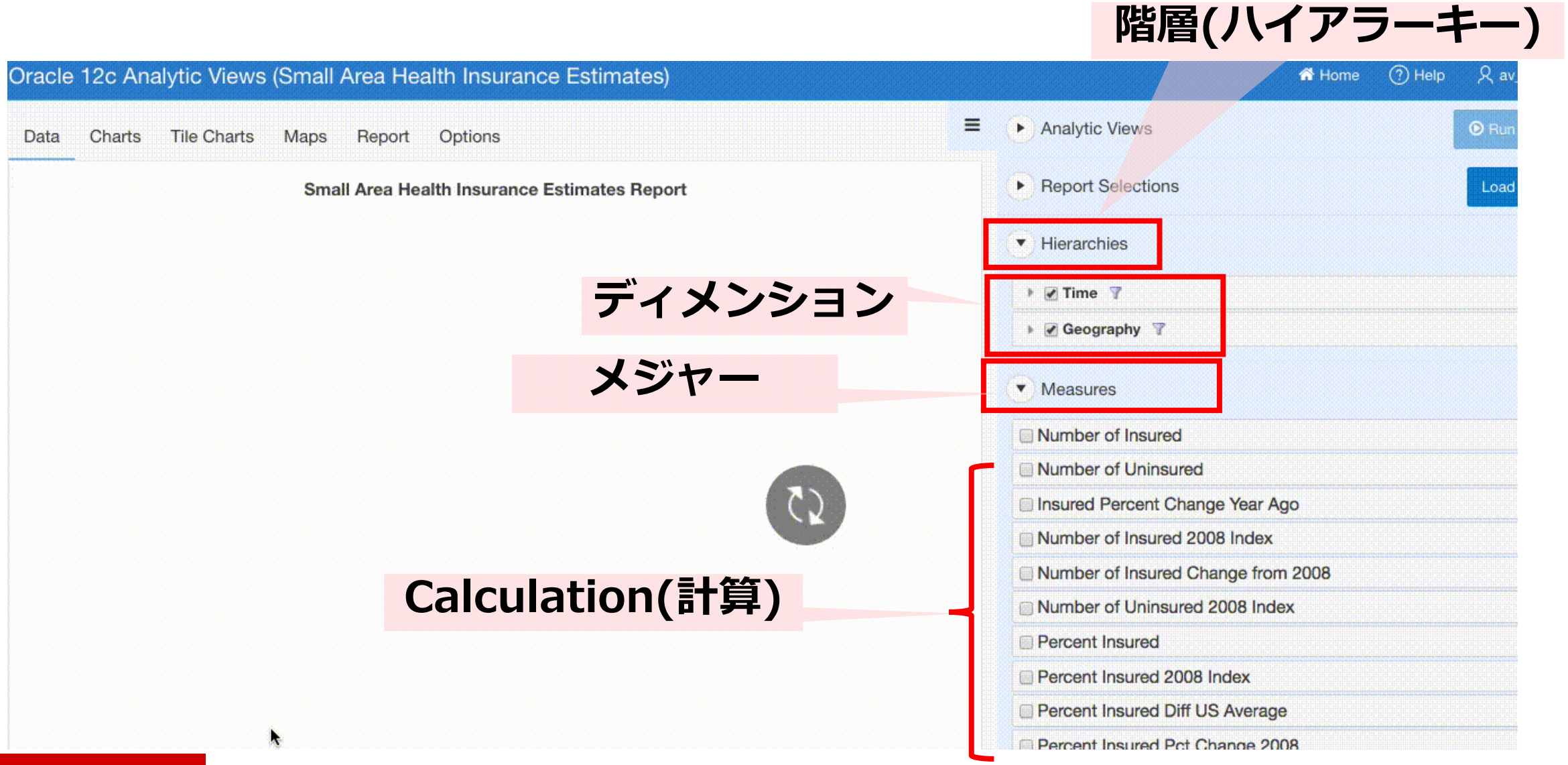
典型的な分析的クエリー
Simple Tables and Complex Queries






```
WITH
  fact_dense AS
      (
        SELECT
       b.year_name,
       b.year_end_date,
       a.department_name,
       a.region_name,
       a.sales
     FROM
       (
         SELECT
           d1.year_name,
           d1.year_end_date,
           d2.department_name,
           d3.region_name,
           SUM(f.sales) sales
         FROM
           time_dim d1,
           product_dim d2,
           geography_dim d3,
           sales_fact f
         WHERE
           d1.month_id            = f.month_id
         AND d2.category_id       = f.category_id
         AND d3.state_province_id = f.state_province_id
         AND year_end_date       IN (add_months('31-DEC-15', - 12),'31-DEC-15')
         GROUP BY
           d1.year_name,
           d1.year_end_date,
           d2.department_name,
           d3.region_name
       )
       a PARTITION BY (a.department_name, a.region_name)
     RIGHT OUTER JOIN
       (
         SELECT DISTINCT
           year_name,
           year_end_date
         FROM
           time_dim
         WHERE
           year_end_date IN (add_months('31-DEC-15', - 12), '31-DEC-15')
       )
       b
     ON
       (
         a.year_name = b.year_name
       )
   )
SELECT
  c.year_name,
  c.year_end_date,
  c.department_name,
 c.region_name,
 c.sales,
 d.sales sales_yr_ago,
 (c.sales       - d.sales) sales_chg_yr_ago,
 ROUND(((c.sales - d.sales) / d.sales), 2) AS sales_pctchg_yr_ago
FROM
  fact_dense c
LEFT OUTER JOIN fact_dense d
ON
  (
    add_months(c.year_end_date, - 12) = d.year_end_date
  AND c.department_name                = d.department_name
  AND c.region_name                    = d.region_name
  )
WHERE
  c.year_name      = 'CY2015'
ORDER BY
  c.year_name,
  c.year_end_date,
  c.department_name,
  c.region_name;
```

Analytic Viewを使用するアプリケーション - APEXの例



"Analytic View" APEX サンプルアプリケーション

- SQLは 5 本(statement)のみ
  - 階層(HIERARCHY) 2 個。SQL 4本
  - "Analytic View" 1個。SQL 1 本
- アプリの全ての構成物はデータベース中に存在

階層SQL x4本

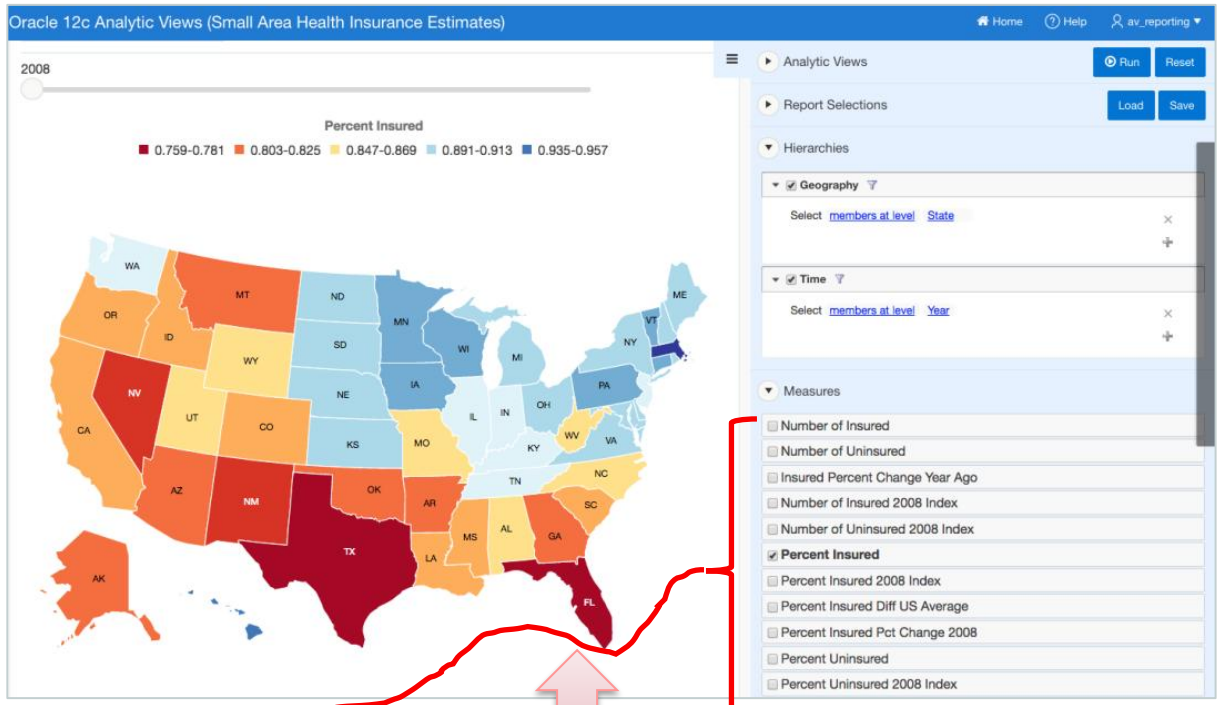
```
CREATE OR REPLACE FORCE HIERARCHY "AV_DEMO"."SAHIE_GEOG_HIER"
    CLASSIFICATION "CAPTION" VALUE 'Geography'
    CLASSIFICATION "DESCRIPTION" VALUE 'Geography'
    CLASSIFICATION "AV_TYPE" VALUE 'GEOGRAPHY'
USING "SAHIE_GEOG_ATTR_DIM"
("COUNTY"
 CHILD OF "STATE"
 CHILD OF "COUNTRY")
```

Analytic View

```
CREATE OR REPLACE FORCE ANALYTIC VIEW "AV_DEMO"."SAHIE_GEOG_AV"
    CLASSIFICATION "CAPTION" VALUE 'Small Area Health Insurance Estimates'
    CLASSIFICATION "DESCRIPTION" VALUE 'US Cenus Bureau - Model-based Smal
    CLASSIFICATION "DATA_SET_ORIGIN" VALUE 'www.census.gov/did/www/sahie/'
USING "SAHIE_GEOG_FACT_V" AS "SAHIE_GEOG_FACT_V"
DIMENSION BY
( "SAHIE_TIME_ATTR_DIM" AS "SAHIE_TIME_ATTR_DIM"
      KEY ("YEAR")
```

計算 x20

```
"INSURED_SHARE_2008" AS
  (SHARE_OF(num_insured HIERARCHY sahie_time_hier MEMBER year['2008']))
  CLASSIFICATION "CAPTION" VALUE 'Number of Insured 2008 Index'
  CLASSIFICATION "DESCRIPTION" VALUE 'Number of Insured 2008 Index'
  CLASSIFICATION "FORMAT_STRING" VALUE '9,999.99',
```



Simple SQL

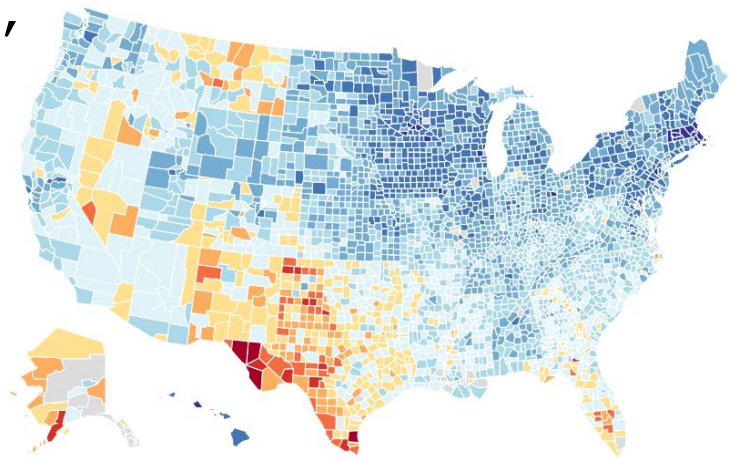
Analytic View

Data
Tables, Views, etc.

Analytic View

簡潔なSQLで様々な切り口の集計結果を返すことが可能

```
SELECT time_hier.member_name AS TIME,
geog_hier.member_name        AS GEOGRAPHY,
  percent_insured
FROM INSURED_ANALYTIC_VIEW
     HIERARCHIES(time_hier,geog_hier)
WHERE time_hier.level_name = 'YEAR'
AND   geog_hier.level_name = 'COUNTY'
ORDER BY
 time_hier.hier_order,
 geog_hier.hier_order
```



ドリルダウンに必要な変更は階層の"level_name"の変更のみ
STATE(州) -> COUNTY(郡)

計算(calculation)カラムは自動的に選択されたレベルで集計し直される

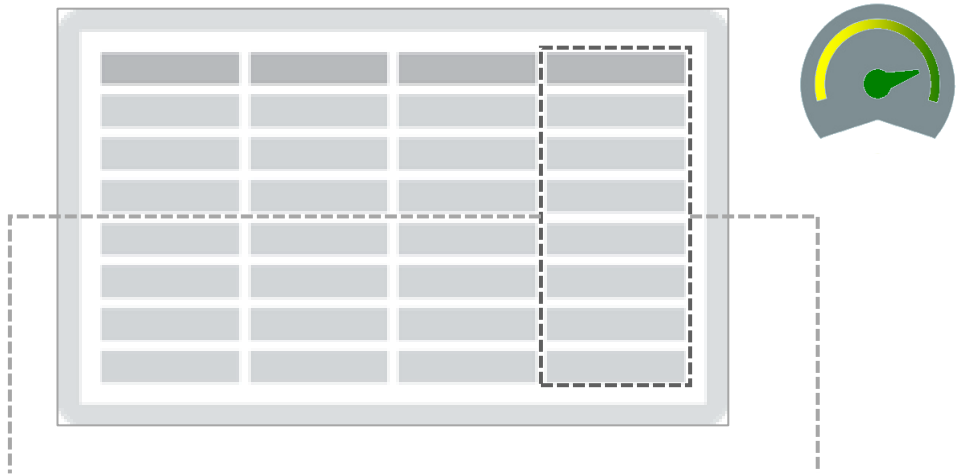
Agenda

1. パーティション
2. Analytic View
3. 近似値計算
4. パフォーマンス/ Optimizer

Approximate Query Processing

正確さよりも応答時間が重視されるインタラクティブで繰り返しが多いデータ探索に非常に高速に分析結果を返す。集計数を求めるクエリーに完全性を求めないケースで有効

例：先週1週間広告Webサイトを訪れた個人の年齢の中間値



- 高価でリソースを多く使う集計計算の近似値算出処理を実装

  APPROX_COUNT_DISTINCT (12.1)

  APPROX_PERCENTILE
  APPROX_MEDIAN

  NEW IN 12.2

  - 6-13倍 高速。正確さは多くの場合 1% 以内
- アプリを全く変更することなく利用可能
  - "approx_for_aggregation = TRUE" パラメーター
- Accuracy および Error Rate も返す
- マテリアライズド・ビュー作成可能
  - クエリー・リライト可能

近似値処理詳細

NEW IN 12.2

- Percentile(パーセンタイル)
  - 例： 日本人の年齢の percentile 0.727 は65歳
    - 昇順時、降順も指定可能
  - MEDIAN: メジアン、中間値
  - PERCENTILE(0.5) は MEDIAN() と同意
- count(distinct <カラム>) などを自動で近似値バージョンに置き換えることが可能に
  - alter session set approx_for_aggregation=true
    - または "ALTER SYSTEM"
- Materialized View例
  - 各市町村でのpercentile近似値のマテリアライズド・ビュー
  - 県レベルでのメジアン近似値をもとめるクエリーは高速化

```
create materialized view
pctl_mview enable query
rewrite as
Select    県名,市町村名,
approx_percentile_detail(age)
detail
From      年齢表
Group by  県名,市町村名
```

参考 : Approx有り無しでの比較 - SQLモニター

- Without approx query processing:
  - 実行計画に SORT
  - 8GB of memory (PGA)
  - 164GB of temp
- With approx query processing:
  - コストの高いSORT無
  - 540MB of memory (PGA)
  - 0 GB temp

50x 高速
15X 少ないメモリー
一時領域必要無

正確なメジアンを求めるクエリー

Details

Plan Statistics | Plan | Parallel | Activity | Metrics

Plan Hash Value 707829080 | Plan Note | All Parallel Servers

Operation	Name	L...	Estim...	C...	Timeline(...	Ex...	Actu...	Mem...	Tem...	O..	IO ...	I...	Cell Offlo...	Activity %
SELECT STATEMENT		0				1	1							.03
SORT GROUP BY		1	1			1	1							
PX COORDINATOR		2				161	454							.01
PX SEND QC (RAN...	:TQ10001	3	1			80	454							
SORT GROUP BY		4	1			80	454	8GB	164GB		1,418K	32...		86
PX RECEIVE		5	1			80	48G							1.73
PX SEND H...	:TQ10000	6	1			80	48G							7.78

メジアンの近似値を求めるクエリー

Details

Plan Statistics | Plan | Parallel | Activity | Metrics

Plan Hash Value 2037788262 | Plan Note

Operation	Name	Li...	Estim...	C...	Timeline(...	Ex...	Actu...	Mem...	Tem...	O..	IO ...	I...	Cell Offlo...	Activity %
SELECT STATEMENT		0				1	1							
SORT AGGREGATE APP...		1	1			1	1							
PX COORDINATOR		2				81	80							
PX SEND QC (RAN...	:TQ10000	3	1			80	80							
SORT AGGREGAT...		4	1			80	80							77
PX PARTITION...		5	6,000M	16K		80	6,000M							
TABLE ACCES...	LINEITEM	6	6,000M	16K		13K	6,000M	542MB			588K	192...	-12.36	23

Agenda

1. パーティション
2. Analytic View
3. 近似値計算
4. パフォーマンス/ Optimizer

コスト・ベース "OR Expansion" 変換の新規選択肢

NEW IN 12.2

- CONCATENATIONと異なりUNION-ALL以下の枝(Branch)はパラレル動作可能

```
SELECT count(*) FROM t_4k T1, h_4k T2, t_10k T3 WHERE T1.ten = T2.ten AND
T1.hundred = T3.hundred AND (T1.unique1 = 10 OR  T2.unique2 = 20);
```

12.1以前の plan:

```
-----------------------------------------------------
| Id  | Operation                     | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |               |
|   1 |  SORT AGGREGATE               |               |
|   2 |   CONCATENATION               |               |
|*  3 |    HASH JOIN                  |               |
|   4 |     TABLE ACCESS FULL         | T_4K          |
|   5 |     MERGE JOIN CARTESIAN      |               |
|*  6 |      TABLE ACCESS FULL        | H_4K          |
|   7 |      BUFFER SORT              |               |
|   8 |       INDEX FAST FULL SCAN    | T_10K_HUNDRED |
|*  9 |    HASH JOIN                  |               |
|  10 |     NESTED LOOPS              |               |
|* 11 |      TABLE ACCESS FULL        | T_4K          |
|* 12 |      INDEX RANGE SCAN         | T_10K_HUNDRED |
|* 13 |     TABLE ACCESS FULL         | H_4K          |
-----------------------------------------------------
```

12.2 plan:

```
-----------------------------------------------------------
| Id  | Operation                     | Name              |
-----------------------------------------------------------
|   0 | SELECT STATEMENT              |                   |
|   1 |  SORT AGGREGATE               |                   |
|*  2 |   HASH JOIN                   |                   |
|   3 |    VIEW                       | VW_JF_SET$4869194F|
|   4 |     UNION-ALL                 |                   |
|*  5 |      HASH JOIN                |                   |
|*  6 |       TABLE ACCESS FULL       | T_4K              |
|   7 |       TABLE ACCESS FULL       | H_4K              |
|*  8 |      HASH JOIN                |                   |
|*  9 |       TABLE ACCESS FULL       | H_4K              |
|* 10 |       TABLE ACCESS FULL       | T_4K              |
|  11 |     INDEX FAST FULL SCAN      | T_10K_HUNDRED     |
-----------------------------------------------------------
```

再帰的なWITHクエリーのパラレル実行が可能に

- "CONNECT BY"句使用により行と行の関連性をたどる検索が可能
- ORACLE独自構文
- **注意点:**
  再帰的なWITHに書き換えることによりパラレル実行が可能に

```
SELECT fid, tid, level iters,
 SYS_CONNECT_BY_PATH(fid, '/')||'/'||tid path
FROM tedges
START WITH fid < tid
CONNECT BY PRIOR tid = fid and fid < tid;
```

FID	TID	ITERS	PATH
1	2	1	/1/2
2	4	2	/1/2/4
4	5	3	/1/2/4/5
1	3	1	/1/3
2	4	1	/2/4
4	5	2	/2/4/5
4	5	1	/4/5

```
WITH rw (fid, tid, iters, path) as
(  SELECT fid, tid, 1, to_char(fid)
   FROM tedges_tw WHERE fid < tid
   UNION ALL
   SELECT r.fid, t.tid, r.iters + 1, r.path||'/'||t.fid
   FROM rw r, tedges_tw t
   WHERE r.tid = t.fid and t.fid < t.tid  )
SELECT fid, tid, iters, path||'/'||tid path FROM rw;
```

RECURSIVE WITH のパラレル実行計画

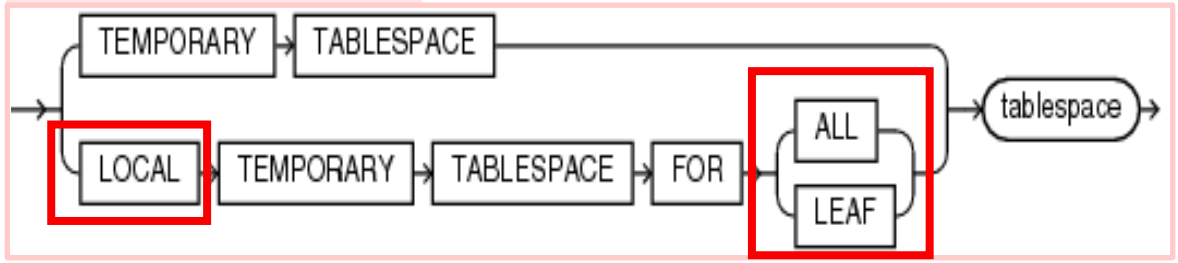
- 右はWITH構文使用時
- RECURSIVE WITH 修飾子
- データによるが数倍程度の高速化
- 右では"CURSOR DURATIO TABLE"も12.2新機能

```
Plan hash value: 3642156913
-------------------------------------------------------------------------------
| Id  | Operation                                          | Name                |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                                   |                     |
|   1 |  TEMP TABLE TRANSFORMATION                         |                     |
|   2 |   LOAD AS SELECT (CURSOR DURATION MEMORY)          | SYS_TEMP_0FD9D6620_E5776 |
|   3 |    UNION ALL (RECURSIVE WITH) BREADTH FIRST|       |                     |
|   4 |     PX COORDINATOR                                 |                     |
|   5 |      PX SEND QC (RANDOM)                           | :TQ20000            |
|   6 |       LOAD AS SELECT (CURSOR DURATION MEMORY)      | SYS_TEMP_0FD9D6620_E5776 |
|   7 |        PX BLOCK ITERATOR                           |                     |
|*  8 |         TABLE ACCESS FULL                          | TEDGES              |
|   9 |     PX COORDINATOR                                 |                     |
|  10 |      PX SEND QC (RANDOM)                           | :TQ10000            |
|  11 |       LOAD AS SELECT (CURSOR DURATION MEMORY)|     | SYS_TEMP_0FD9D6620_E5776 |
|* 12 |        HASH JOIN                                   |                     |
|  13 |         BUFFER SORT (REUSE)                        |                     |
|* 14 |          TABLE ACCESS FULL                         | TEDGES              |
|  15 |         PX BLOCK ITERATOR                          |                     |
|* 16 |          TABLE ACCESS FULL                         | SYS_TEMP_0FD9D6620_E5776 |
|  17 |   PX COORDINATOR                                   |                     |
|  18 |    PX SEND QC (RANDOM)                             | :TQ30000            |
|  19 |     VIEW                                           |                     |
|  20 |      PX BLOCK ITERATOR                             |                     |
|  21 |       TABLE ACCESS FULL                            | SYS_TEMP_0FD9D6620_E5776 |
-------------------------------------------------------------------------------
```

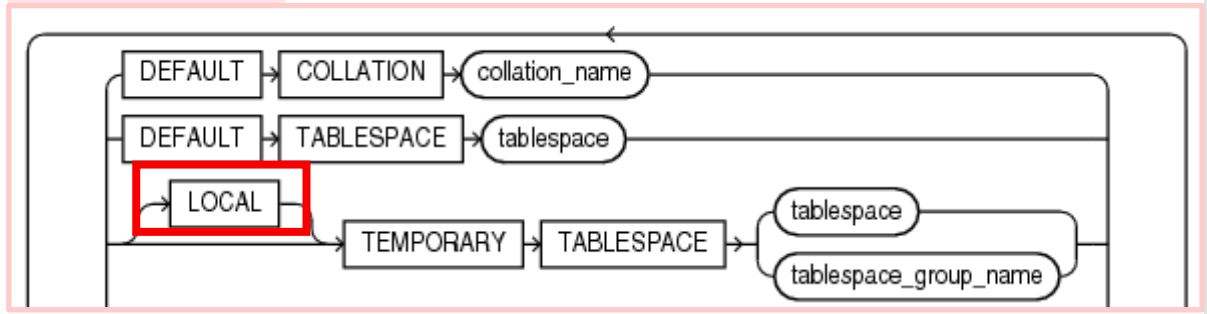
ローカル一時表領域

- RAC環境で各ノードでローカルに使用する表領域
  - 全ノードのローカルディスクで共通のパスにデータファイル作成
  - BIGFILE 表領域のみ
  - I/O高速化によりパフォーマンス向上につながる場合も
- ソート、ハッシュ系処理で使用

CREATE TABLESPACE



CREATE USER




```
SQL> SELECT default_tablespace, temporary_tablespace, local_temp_tablespace from
        DBA_USERS  where username = 'SCOTT';
DEFAULT_TABLESPACE     TEMPORARY_TABLESPACE     LOCAL_TEMP_TABLESPACE
SCOTT_DATA             SCOTT_TEMP               LTT_ALL
```

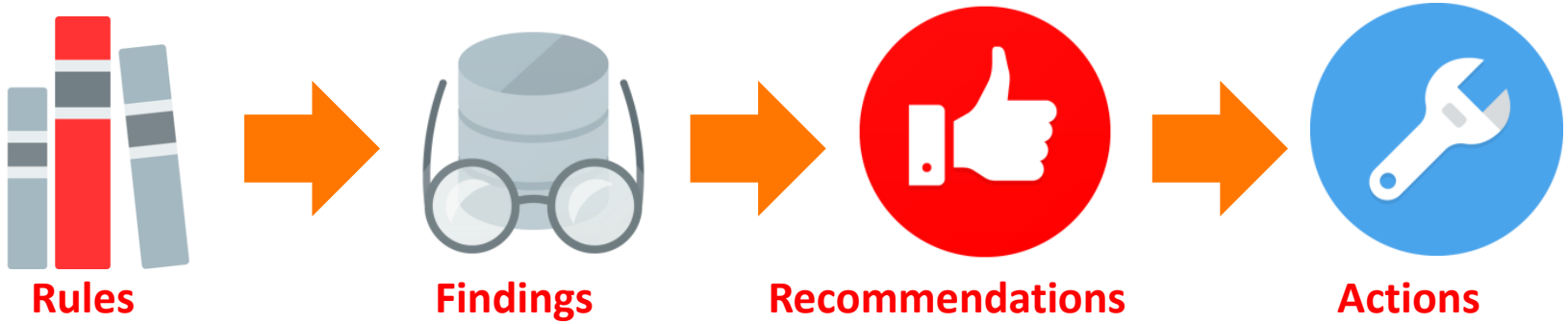
オプティマイザ統計アドバイザー

(Optimizer Statistics Advisor)

Optimizer Statistics Advisor

NEW IN 12.2

- 過去から引き継いでいる資産（スクリプト）が使われていることがよくあり、最適ではない設定でオプティマイザ統計が収集されている
- データベースがオプティマイザ統計を収集するために現在のベストプラクティスに従っていることを確認するための**ルール**セットが適用される
- **調査結果**と**推奨事項**を提示するレポートを生成
- **アクション**を使用して推奨事項を実施するためのSQLスクリプトを生成
- メンテナンスウィンドウ中のジョブでの実行がデフォルトで設定される



12個のアドバイザー "ルール"

```
SELECT RULE_ID AS ID, NAME, RULE_TYPE, DESCRIPTION FROM   V$STATS_ADVISOR_RULES
WHERE  RULE_ID BETWEEN 1 AND 12 ORDER BY RULE_ID;

ID NAME                           RULE_TYPE DESCRIPTION
-- ------------------------------ --------- ----------------------------------------------------
 1 UseAutoJob                     SYSTEM    Use Auto Job for Statistics Collection
 2 CompleteAutoJob                SYSTEM    Auto Statistics Gather Job should complete successfully
 3 MaintainStatsHistory           SYSTEM    Maintain Statistics History
 4 UseConcurrent                  SYSTEM    Use Concurrent preference for Statistics Collection

[…以下略…] 全部で 12 のルール […]
```

- その他のルールからの抜粋
  - dbms_stats.set_global_prefs() を以前に使ったことがあればデフォルトに戻す
  - SQL Plan Directive(SPD) を有効化
  - SET_{COLUMN|INDEX|TABLE|SYSTEM}_STATS は使わないように
  - "estimate_percent" などはデフォルトを使用する
  - 不要な統計収集は停めましょう

カスタマイズ

NEW IN 12.2

- アドバイザーのスコープは3種類のフィルターによって狭めることが可能
  - オブジェクト、オペレーション、ルール
- 特定のオブジェクトを除外する例

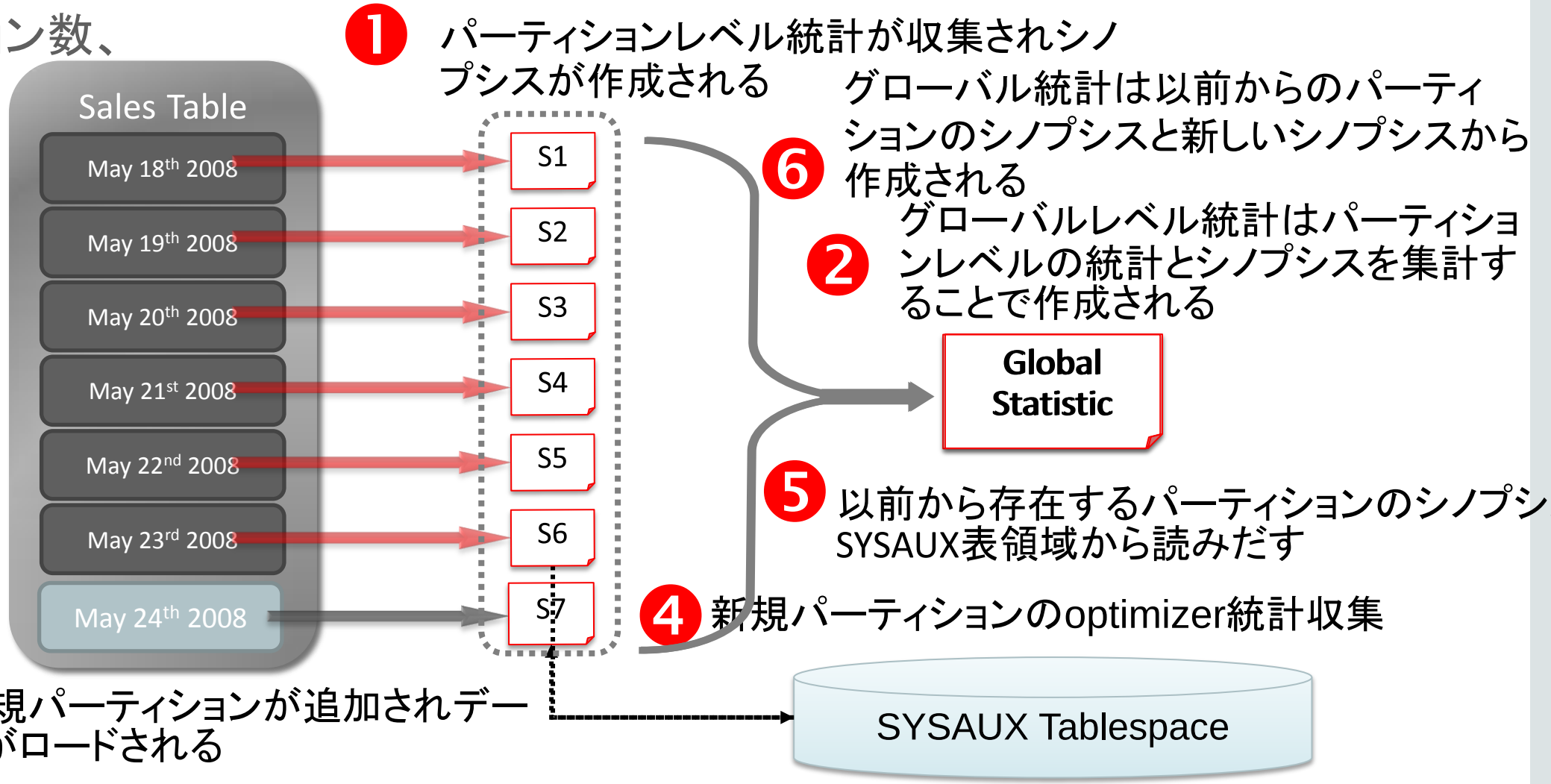
```
-- Turn off validation/reporting... for table SH.COUNTRIES
declare
   tname   VARCHAR2(32767) := 'demo';   -- タスク名
   filter_report clob;                  -- report of operation
begin
  filter_report :=
    dbms_stats.CONFIGURE_ADVISOR_OBJ_FILTER(tname, NULL, NULL, 'SH', 'COUNTRIES',
                                            'DISABLE');
end;
```

オプティマイザその他

パーティション表の統計収集
SQL Plan Management

パーティション表でのインクリメンタル統計収集

- シノプシス: パーティション毎、カラム毎のNDV(一意値)の情報
  - パーティション数、カラム数、NDVによりサイズが大きくなる傾向があった

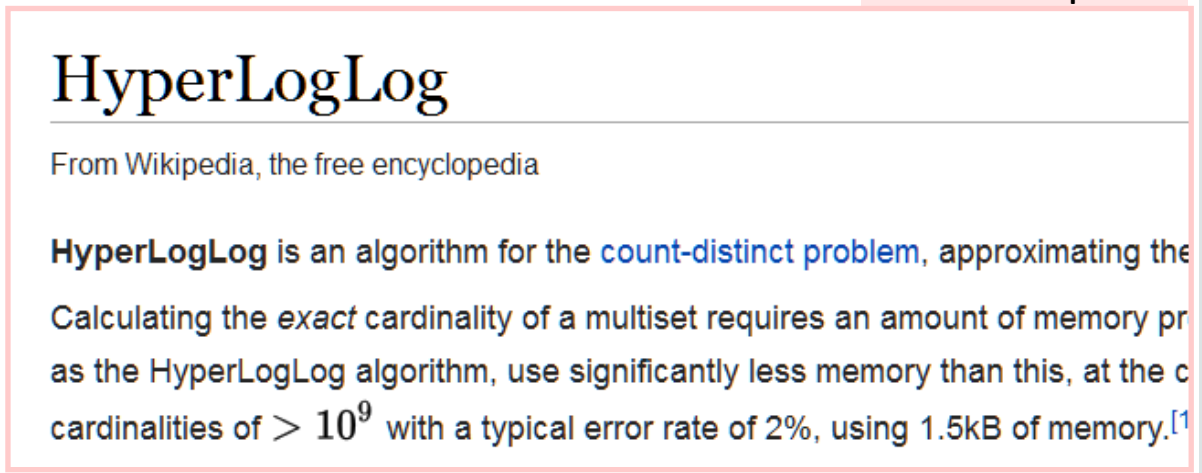


シノプシスのサイズ削減が可能に

NEW IN 12.2

- DBMS_STATS のプレファレンスの新しいエントリーにNDV収集アルゴリズムを指定
  - Preference 名: **APPROXIMATE_NDV_ALGORITHM**
    - `'REPEAT OR HYPERLOGLOG'` [Default]
      - 古いフォーマットのシノプシスが存在する場合古いアルゴリズムを使い続ける
      - シノプシスが存在しない、または、新しいフォーマットのものがある場合、新アルゴリズムを使用
    - `'ADAPTIVE SAMPLING'`
      - 強制的に古いアルゴリズムを使用
    - `'HYPERLOGLOG'`
      - 強制的に新しいアルゴリズムを使用

出典: Wikipedia



HyperLogLog

From Wikipedia, the free encyclopedia

HyperLogLog is an algorithm for the count-distinct problem, approximating the

Calculating the *exact* cardinality of a multiset requires an amount of memory pr

as the HyperLogLog algorithm, use significantly less memory than this, at the c

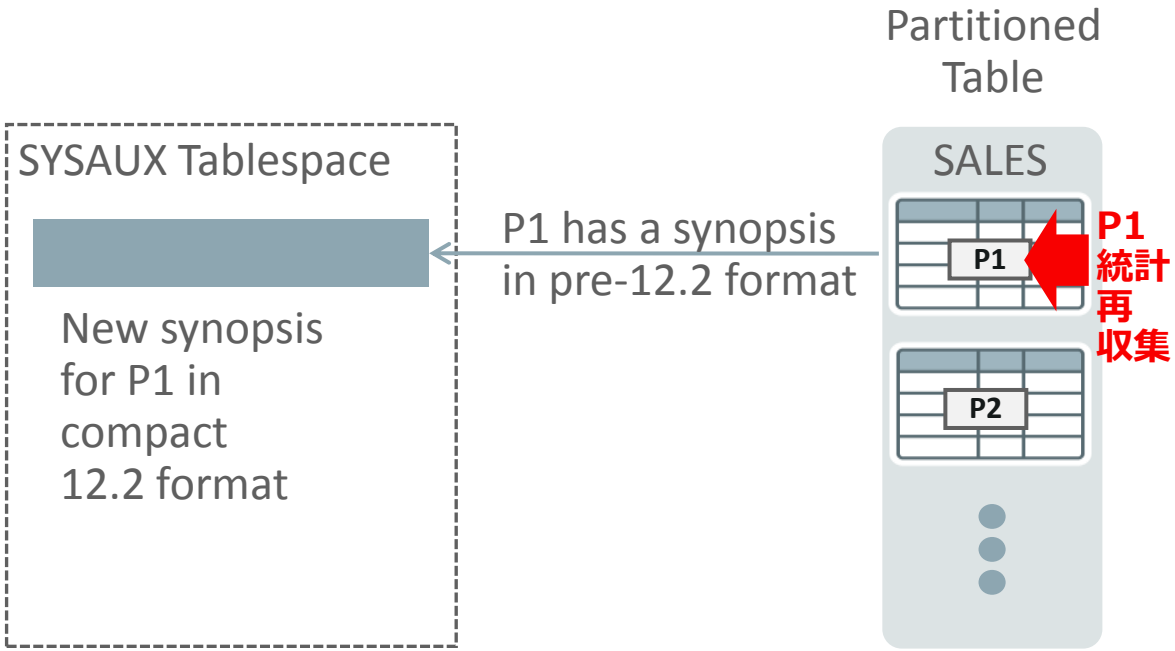
cardinalities of $> 10^9$ with a typical error rate of 2%, using 1.5kB of memory.[1

  - 削減例:
    - 8TB table with 84 partitions ➔ Size in Oracle 12.1.0.2: 167GB ➔ Size in Oracle 12.2.0.1: 7GB

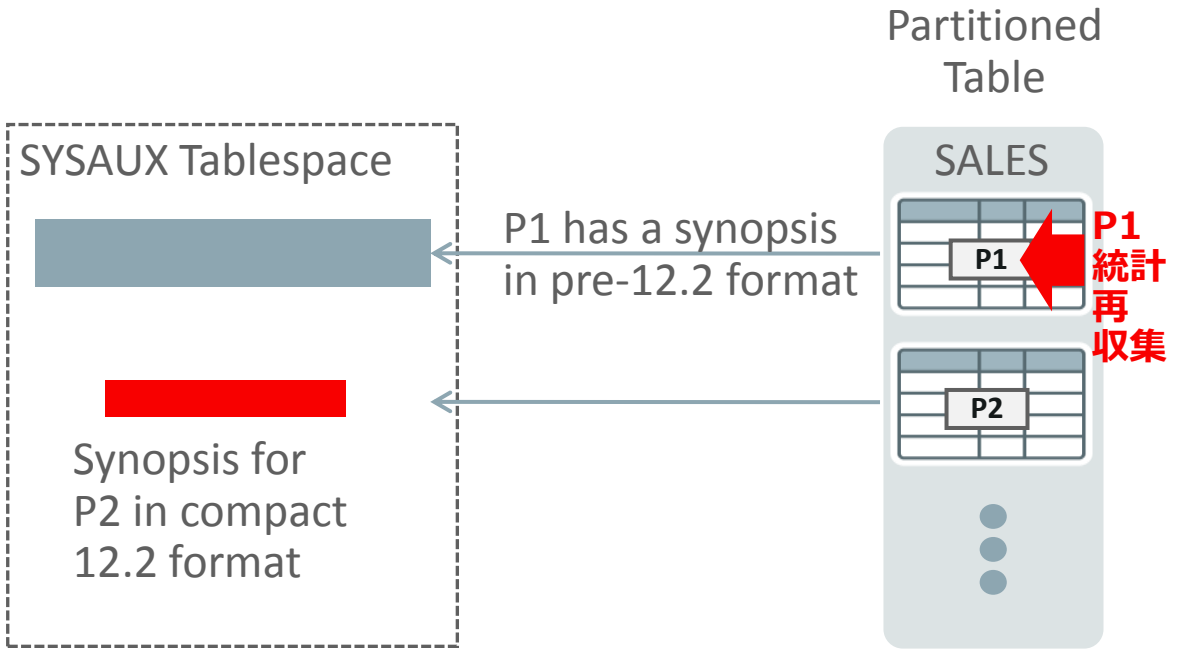
Things to do right after upgrade

NEW IN 12.2

- Preference: *HYPERLOGLOG*
  - *古い形式のシノプシスを新しい形式で置き換える*



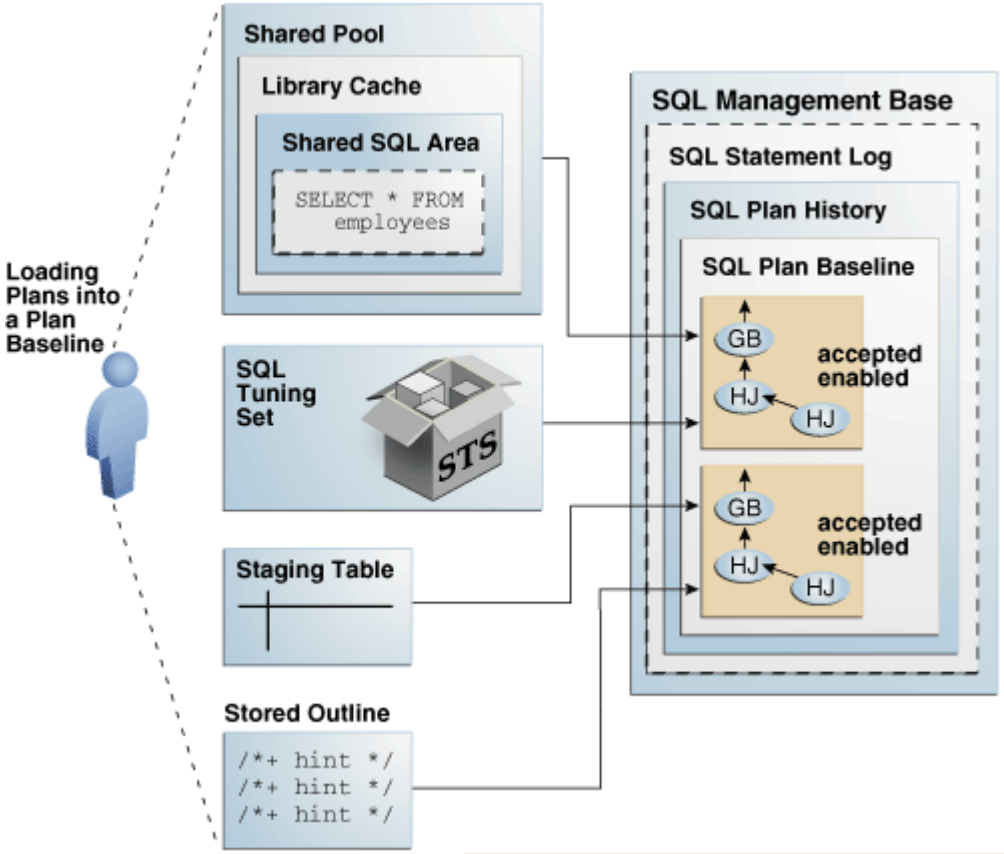
- Preference: *REPEAT OR HYPERLOGLOG*
  - *古い形式のシノプシスと新しい形式のシノプシスが共存*

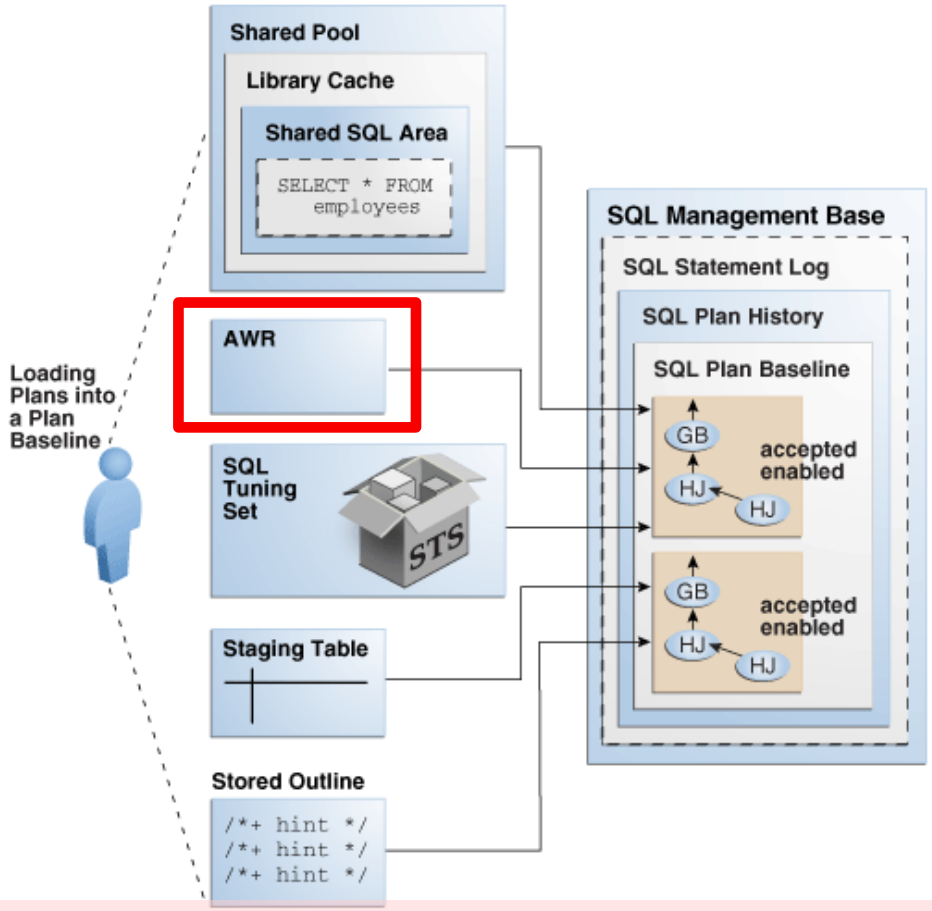


SPM（SQL Plan Management）新機能

AWRからプランをキャプチャできるように

NEW IN 12.2






```
SQL> exec DBMS_SPM.LOAD_PLANS_FROM_AWR(begin_snap=>29,
                                        end_snap=>69)
```

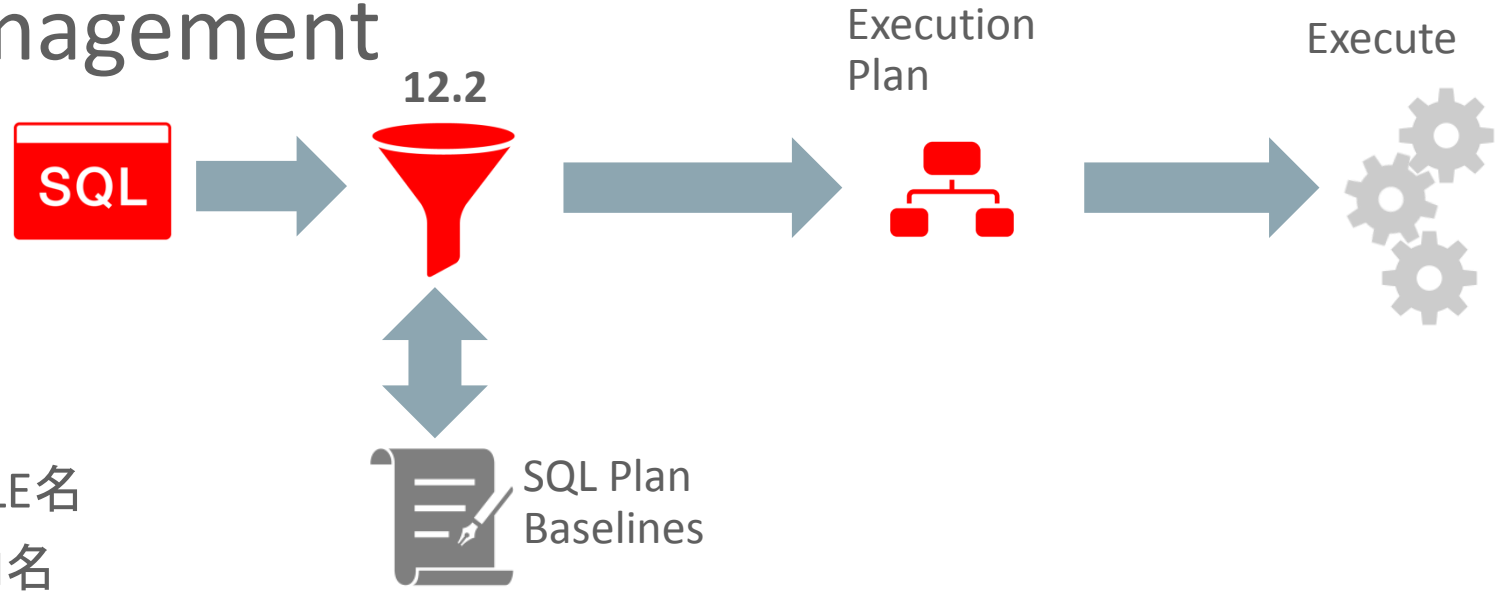
SQL Plan ベースライン自動キャプチャー - 12.2

NEW IN 12.2

- 以前のリリースでは、自動キャプチャは全ての繰り返し可能なクエリーに適用された。
  - 無視できるようなクエリーなども含んでおりSYSAUX 表領域を圧迫するケースがあった
- 12.2から OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES = TRUE 設定で**自動取得される計画をフィルタ可能**に
  → 計画ベースラインの肥大化を抑制できる
- フィルター作成例

```
SQL> exec DBMS_SPM.CONFIGURE('AUTO_CAPTURE_PARSING_SCHEMA','SCOTT', TRUE)
```

Enhanced SQL Plan Management



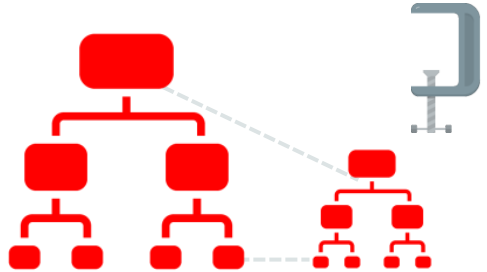
- フィルタ条件に使用できる要素
  - SQLが解析（実行）されたスキーマ
  - SQLを発行したプログラムのMODULE名
  - SQLを発行したプログラムのACTION名
  - 発行されたSQL文字列（ LIKE条件が適用される）
- 想定される利用シナリオ
  → 特定の条件を満たすSQLのみSQL計画管理に登録
  - 特定のスキーマ（アプリケーション用スキーマなど）のSQLのみ
  - 特定のプログラム（バッチ用など）から発行されたSQLのみ
  - 特定のコメントを含むSQLのみ（要注意SQLなど）
  - 特定の表にアクセスするSQLのみ（大データを格納する表など）

参考資料

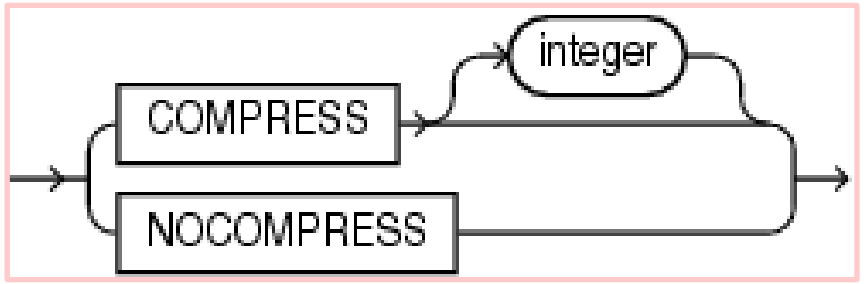
インデックス圧縮

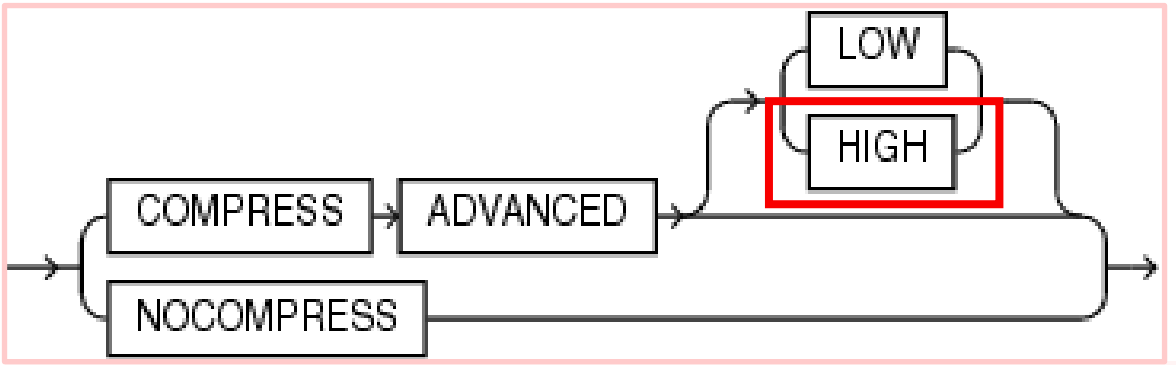
COMPRESS ADVANCED HIGH

Advanced インデックス圧縮



- 11g 以前: Prefix圧縮
  - Ver 8.1.3 から
  - 別名 Key 圧縮
- 12c R1(12.1.0.2): ADVANCED LOW
  - ブロック毎に圧縮法調節
  - 圧縮率の目安: 2.5 x
  - バッファ・キャッシュにより多くのるため高速になる場合も
- 12c R2: ADVANCED HIGH
  - より高い圧縮率
  - 単一カラムユニークインデックスにも対応





NEW IN 12.2

要 : Advanced Compression Option(ACO)

Advanced インデックス圧縮 - 圧縮率見積もり

- DBMS_COMPRESSION パッケージ
- 旧インデックス圧縮は対応していない
- COMPTYPE に COMP_INDEX_ADVANCED_LOW または COMP_INDEX_ADVANCED_HIGH を指定

DBMS_COMPRESSION.GET_COMPRESSION_RATIO

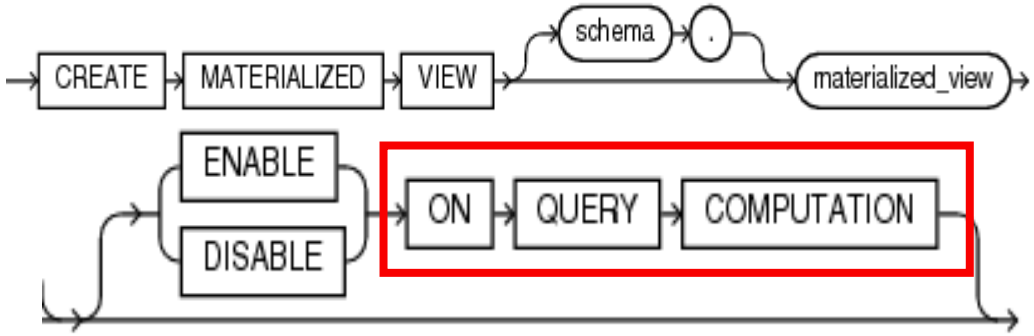
```
(
scratchtbsname => 'USERS',
ownname => 'U1',
objname => 'IDX_T1_OBJECT_ID',
subobjname => NULL,
comptype => dbms_compression.COMP_INDEX_ADVANCED_LOW,
blkcnt_cmp => blkcnt_cmp,
blkcnt_uncmp => blkcnt_uncmp,
row_cmp => row_cmp,
row_uncmp => row_uncmp,
cmp_ratio => cmp_ratio,
comptype_str => comptype_str,
subset_numrows => dbms_compression.COMP_RATIO_MINROWS,
objtype => dbms_compression.OBJTYPE_INDEX
);
DBMS_OUTPUT.PUT_LINE('Block count compressed = ' || blkcnt_cmp);
DBMS_OUTPUT.PUT_LINE('Block count uncompressed = ' || blkcnt_uncmp);
DBMS_OUTPUT.PUT_LINE('Row count per block compressed = ' || row_cmp);
DBMS_OUTPUT.PUT_LINE('Row count per block uncompressed = ' || row_uncmp);
DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype_str);
DBMS_OUTPUT.PUT_LINE('Compression ratio org= '||cmp_ratio);
Block count compressed = 171
Block count uncompressed = 170
Row count per block compressed = 447
Row count per block uncompressed = 450
Compression type = "Compress Advanced Low"
Compression ratio org= 3
```

マテリアライズド・ビュー

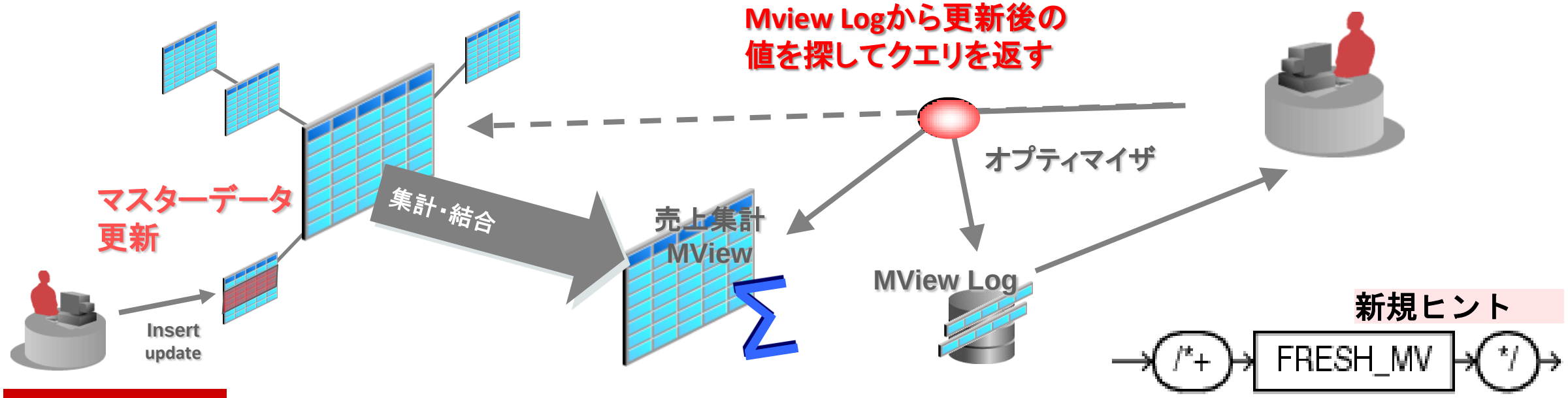
12.2 から新規の2種のタイプ

Real-time Materialized Views

別名： ON QUERY COMPUTATION



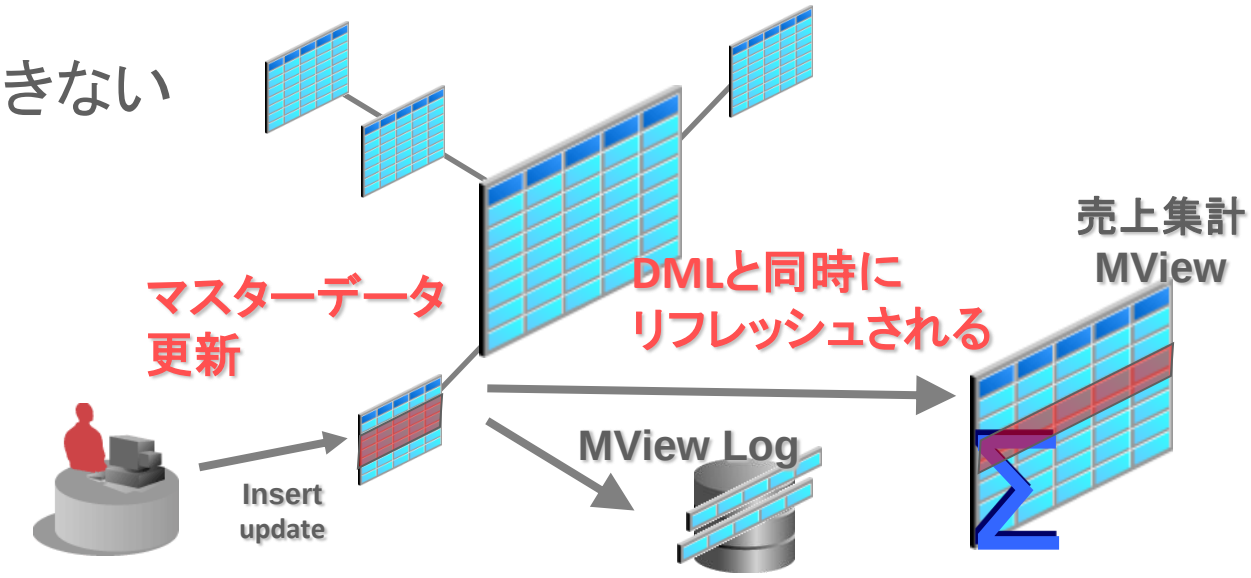
- マテビューが最新ではない場合、マテビューログへ差分を見に行く
- マテビューログの値からマテビューの差分をマージし結果を返す(On Query Compute)



ON STATEMENT リフレッシュ

- マテビューの更新頻度を指定する句に「ON STATEMENT」が指定可能になった
- 元表への更新(DML)があると即座に(commit前に)マテビューがリフレッシュされる
- 更新をcommit/rollbackするとマテビューのリフレッシュもcommit/rollbackされる
- StarまたはSnowflake Schema 上のスター・クエリー使用必須
- REFRESH FAST 必須
- CREATE時の指定必要。ALTER文で指定できない

```
SQL> select mview_name, refresh_mode from
    user_mviews;

MVIEW_NAME                     REFRESH_MODE
------------------------------ ------------------------
FACTMV                         STATEMENT
```



その他

Resource Manager

- セッション毎の最大使用PGAの制限
- 越えると ORA-10260 発生
- プランディレクティブに新規追加
- 数100MB から 1.5GB ぐらいが想定される設定

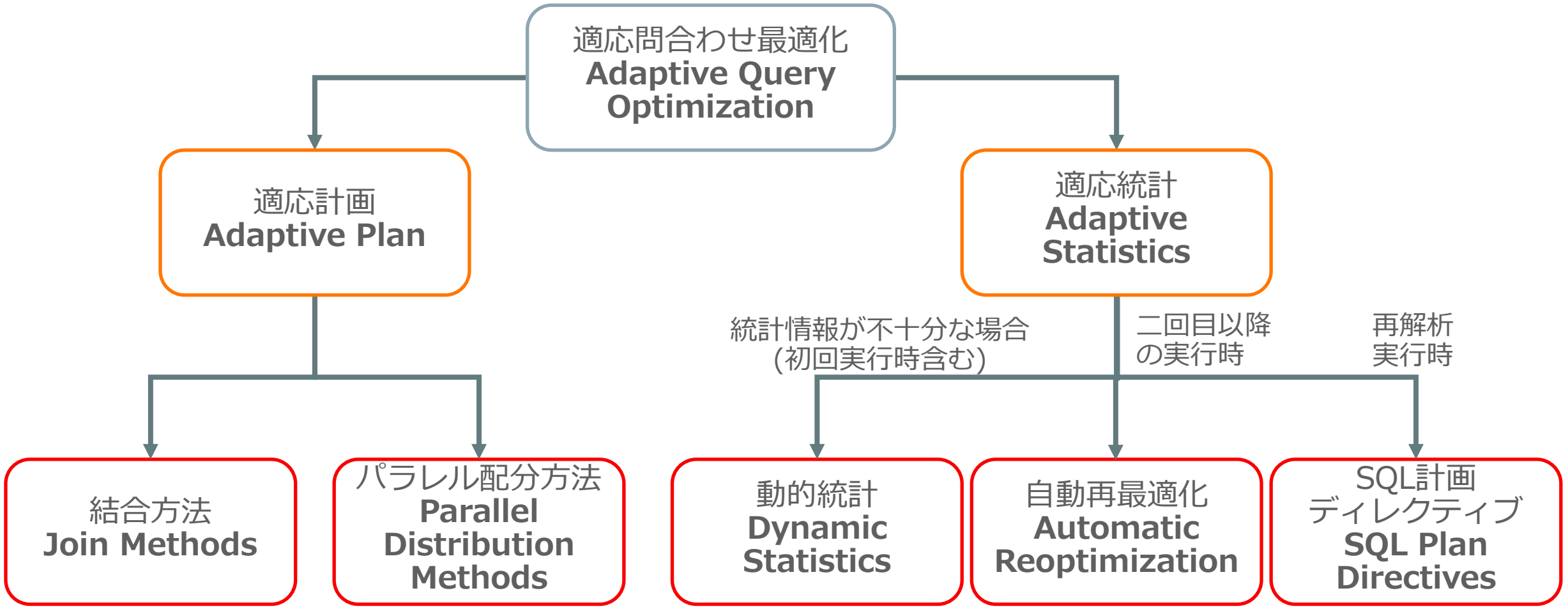
12c R2プランディレクティブ一覧

ACTIVE_SESS_POOL_P1
MAX_EST_EXEC_TIME
MAX_IDLE_BLOCKER_TIME
MAX_IDLE_TIME
PARALLEL_DEGREE_LIMIT_P1
QUEUEING_P1
SESSION_PGA_LIMIT
SWITCH_ESTIMATE
SWITCH_FOR_CALL
SWITCH_GROUP
SWITCH_IO_MEGABYTES
SWITCH_IO_REQS
SWITCH_TIME
UNDO_POOL
UTILIZATION_LIMIT

スケジューラー(DBMS_SCHEDULER)ジョブ高速オプション

- In-Memory ジョブ
  - DBMS_SCHEDULER.CREATE_JOBに新規ジョブスタイル
    - IN_MEMORY_RUNTIME
    - IN_MEMORY_FULL
      - 繰り返し指定不可
  - 実行時オーバーヘッドを小さくする
  - Active Data Guardスタンバイでの実行は非サポート

12c R1のオプティマイザ機能



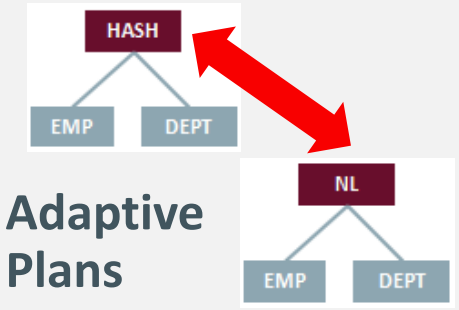
12.1 Optimizer 新機能 Adaptive 機能のパラメーター変更

NEW DEFAULT

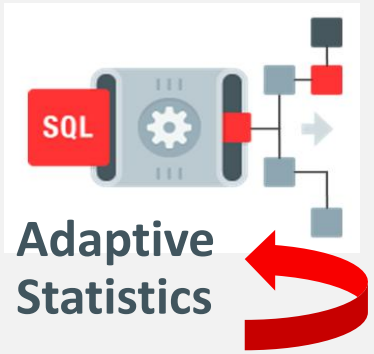
Oracle 12.1 `optimizer_adaptive_features`

Oracle 12.2

optimizer_adaptive_plans
Default: **TRUE**



optimizer_adaptive_statistics
Default: **FALSE**

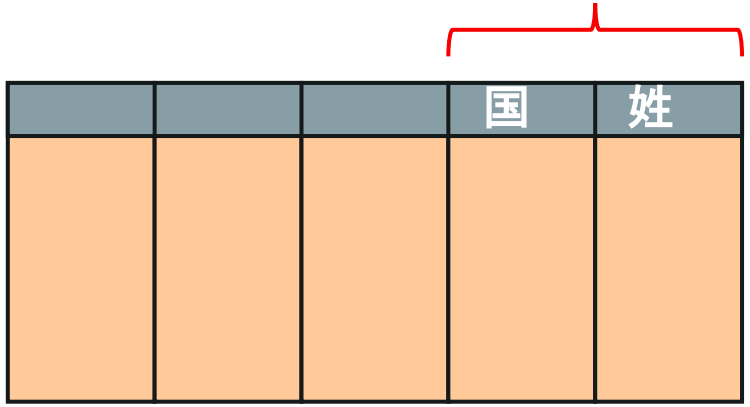


	何よりも**パフォーマンス安定性**重視	やや**保守的**な**Adaptive** アプローチ	新規の**複雑な**クエリーにも**ベスト**パフォーマンス
optimizer_adaptive_plans	FALSE	TRUE	TRUE
optimizer_adaptive_statistics	FALSE	FALSE	TRUE

拡張統計収集

- AUTO_STAT_EXTENSIONS=OFF がデフォルトに
- 12.1 でもパッチを当てれば同様

相関関係



- Things to consider before upgrade to Oracle Database 12.1.0.2 to avoid Poor Performance or Wrong Results: MOS Note:2034610.1

No PSU	PSU 1	2	3	4	5	160119	160419	Bugs Fixed
Patch 21171382 for 12.1.0.2.0								Document 21171382.8 Enh: AUTO_STAT_EXTENSIONS preference on DBMS_STATS

外部テーブル統計収集

- HDFS, oracle_datapump ,oracle_loader 等
- パーティションサポート
- 現時点で通常表とくらべると全く同等ではない
  - オプティマイザ統計の差分(incremental)による更新は未サポート

Windows

- ReFS (Resilient File System) サポート
  - より高パフォーマンスに
- Group Managed Services Account (gMSA)
  - Windows Server 2012以上
- Virtual Account
  - Windows Server 2008以上
  - パスワード不要
- 新規の Microsoft Management Console(MMC) スナップインにより管理可能に
  - ORADIM.exeコマンドラインツールのMMC版

本編の補足

Conversion to Partitioned Table

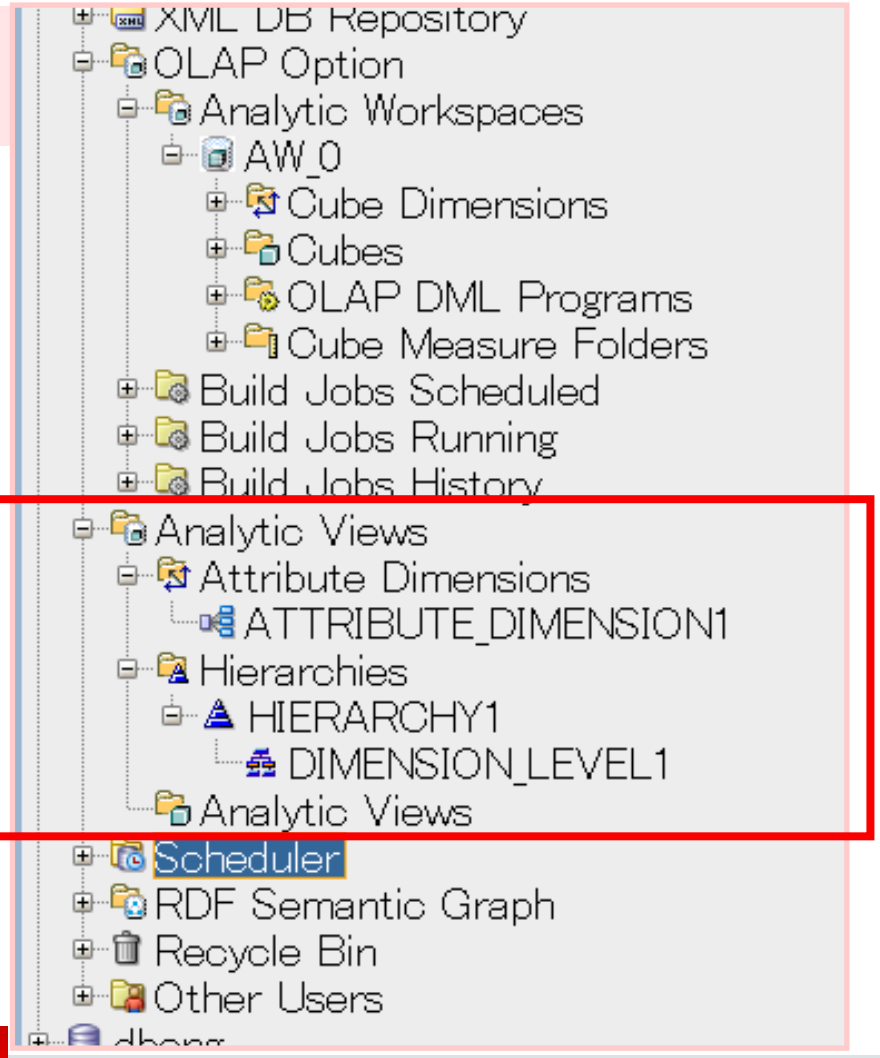
参考 (ONLINE句の有無で実行時間を比較)

```
select count(*) from customers;
  COUNT(*)
----------
   1000000

set timing on

ALTER TABLE customers MODIFY
  PARTITION BY RANGE (customer_id)
  (PARTITION p_1 VALUES LESS THAN (500001),
  PARTITION p_2 VALUES LESS THAN (99999999))
[ONLINE ] ;
```

	ONLINE	OFFLINE
1回目	00:00:13.35	00:00:11.76
2回目	00:00:13.34	00:00:11.74

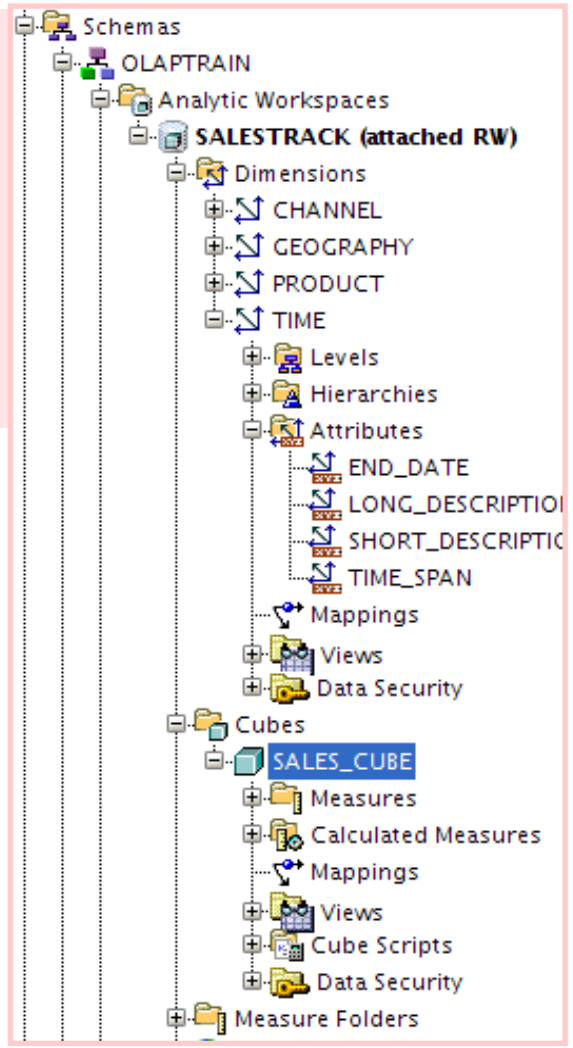
SQL DeveloperでのAnalytic View作成

SQL Developer UIも開発中



ご参考:
旧来の Oracle OLAP Option ツール

Analytics Workspace Manager(awm)



近似値の Materialized View

- MVIEWを用いた近似値処理の高速化

```
CREATE MATERIALIZED VIEW mv0_ymd
enable query rewrite  AS
 SELECT y, m, d,
 APPROX_COUNT_DISTINCT_DETAIL(name1)
apx1
 FROM tbl0 GROUP BY y, m, d;
```

```
SELECT y, m, d,
 TO_APPROX_COUNT_DISTINCT(apx1)
FROM mv0_ymd;
```

MVIEWとして事前計算しバイナリー形式で保持

元データの表

y	m	d	name1
2016	1	1	abcdeff
2016	1	1	abcdeff
2016	1	2	有限YY
2016	1	2	有限YX
2016	1	2	有限YZ
2016	1	3	越後屋本店
2016	1	3	越後屋本締
2016	1	3	越後屋本店
2016	1	4	株)X商会
2016	1	4	株)X商会

MVIEW

y	m	d	apx1
2016	1	1	(BLOB)
2016	1	2	(BLOB)
2016	1	3	(BLOB)
2016	1	4	(BLOB)

問合せ結果

y	m	d	cnt
2016	1	1	1
2016	1	2	3
2016	1	3	1
2016	1	4	1

```
SELECT y, m, d,
APPROX_COUNT_DISTINCT
(name1) cnt
  FROM tbl0 GROUP BY y, m, d;
```

元データの表に直接APPROX_COUNT_DISTINCTを実行するのと同等

3種の集計近似値表関連関数

NEW IN 12.2

近似値群を格納する表を作る関数、および、近似値を取り出す関数
マテリアライズド・ビュー作成に使用可能

1. **APPROX_xxxxxx_DETAIL**(カラム名など)
   - **GROUP BY** 句の全てのディメンションにつきサマリーテーブルを作る
   - 例: approx_count_distinct_detail(volume)
2. **APPROX_xxxxxx_AGG** (expr)
   - 下位レベルのサマリーを**ロールアップ**し上位レベルのサマリー作成
   - 例: approx_percentile_agg(detail)
3. **TO_APPROX_xxxxxx(detail, …)**
   - 集計サマリーから結果を取り出す
   - 例: to_approx_percentile(approx_percentile_agg(detail), 0.78)

NEW IN 12.2

まとめ: Approximate(近似値) Query Processing

- 12.1.0.2で導入されたApproximate Query Processing(近似値を高速に返すSQL処理)を強化

1. Approx系ファンクションの追加

Version	ファンクション	相当する機能
12.1.0.2	APPROX_COUNT_DISTINCT()	COUNT(DISTINCT <列>)
12.2	APPROX_FOR_PERCENTILE(), APPROX_MEDIAN()	PERCENTILE_CONT(), PERCENTILE_DISC()

2. 非Approx系ファンクションをApprox処理に自動的に置換するパラメータの追加

初期化パラメータ	置換処理
APPROX_FOR_AGGREGATION	下記2つをまとめて設定するumbrellaパラメーター
APPROX_FOR_COUNT_DISTINCT	COUNT(DISTINCT <列>) → APPROX_COUNT_DISTINCT()
APPROX_FOR_PERCENTILE= {all|percentile_disc|percentile_cont}	PERCENTILE_CONT() PERCENTILE_DISC() → APPROX_FOR_PERCENTILE()

ORACLE

OPT_PARAMで近似値関連パラメーターも指定可能に

- APPROX_FOR_AGGREGATION, APPROX_FOR_COUNT_DISTINCT, APPROX_FOR_PERCENTILE, OPTIMIZER_DYNAMIC_SAMPLING, OPTIMIZER_INDEX_CACHING, OPTIMIZER_INDEX_COST_ADJ, OPTIMIZER_SECURE_VIEW_MERGING, STAR_TRANSFORMATION_ENABLED

SQL Planベースラインキャプチャー: フィルタ条件の指定

EXEC DBMS_SPM.CONFIGURE('<フィルタ用の属性>','<属性値>',<enable>)

enable: {true または false}

フィルタ用の属性名	フィルタ条件
AUTO_CAPTURE_PARSING_SCHEMA_NAME	SQLが解析（実行）されたスキーマ
AUTO_CAPTURE_MODULE	SQLを発行したプログラムのMODULE名
AUTO_CAPTURE_ACTION	SQLを発行したプログラムのACTION名
AUTO_CAPTURE_SQL_TEXT	発行されたSQL文字列（LIKE条件が適用）

DBMS_SPM.CONFIGUREに与える引数と適用されるフィルタ条件

属性	適用されるフィルタ条件	
	enable=true	enable=false
AUTO_CAPTURE_SQL_TEXT以外の3つ	= <属性値>	<> <属性値>
AUTO_CAPTURE_SQL_TEXT	LIKE <属性値>	NOT LIKE <属性値>

フィルタ条件の確認

- DBA_SQL_MANAGEMENT_CONFIGから指定したフィルタ条件を確認可能

```
SQL> SELECT PARAMETER_NAME, PARAMETER_VALUE
       FROM DBA_SQL_MANAGEMENT_CONFIG
       WHERE  PARAMETER_NAME LIKE '%AUTO%';

PARAMETER_NAME                          PARAMETER_VALUE
--------------------------------------- ---------------------------------
AUTO_CAPTURE_PARSING_SCHEMA_NAME        parsing_schema IN (SCOTT, HR)
AUTO_CAPTURE_MODULE
AUTO_CAPTURE_ACTION
AUTO_CAPTURE_SQL_TEXT                   (sql_text LIKE %SPM_CAPTURE%)
```

リファレンス
マニュアル・ドキュメント

- Oracle Database VLDB and Partitioning ガイド, 12*c* リリース 2 (12.2)
  - パーティショニング、インデックス圧縮
    http://docs.oracle.com/cd/E82638_01/VLDBG/toc.htm
- Oracle Database Data Warehousing ガイド, 12*c* リリース 2 (12.2)
  - Analytic View, 近似値クエリー, マテリアライズド・ビュー
    https://docs.oracle.com/cd/E82638_01/DWHSG/toc.htm
- Oracle Database SQL チューニングガイド, 12c リリース 2 (12.2)
  - オプティマイザ統計アドバイザー、パフォーマンス
    https://docs.oracle.com/cd/E82638_01/TGSQL/toc.htm

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®