
ORACLE®

ORACLE

Oracle Database 12c Release 1 CoreTech Seminar

Performance(optimizer)

日本オラクル株式会社
大池伸幸

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

Agenda

- 実行計画最適化の強化
- 統計情報に関する新機能
- SQL計画管理(SPM)の新機能
- パラレル実行に関する新機能

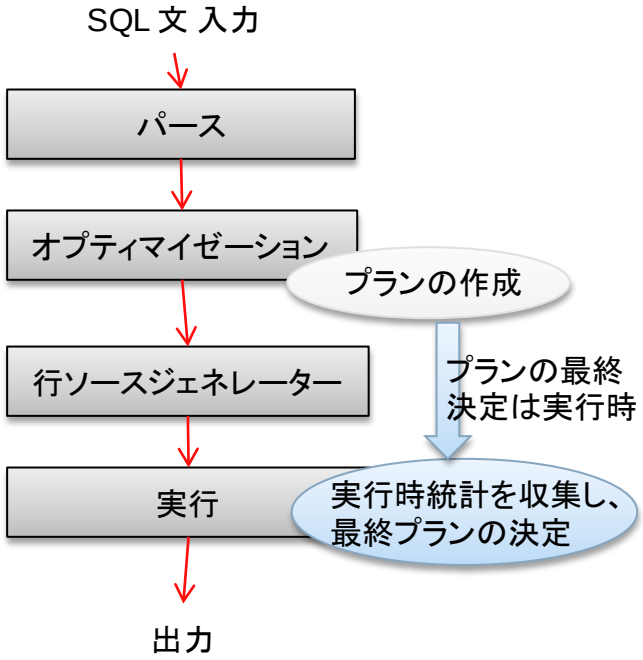
実行計画最適化の強化

実行計画最適化の強化

- SQL 実行時に統計を収集し、より適切なプラン(実行計画)を選択する
- 最適なプランを生成するために既存の統計では十分でない場合に有用
- 以下の二つの仕組みでプランを適応
  - 適応計画(Adaptive Plans): SQL 実行時にプランの適応を行う
  - 自動再最適化(Automatic Reoptimization): SQL 実行時に得た統計を次回の実行時にフィードバックし、プランの適応を行う

適応計画

- 問合わせ実行時に収集した統計を基に最終的なプランを決定
  - 既存の統計を基にプランを作成 (デフォルトプラン)
  - 実行時に収集した統計を基に、プランの一部を実行時の条件に適したサブプランに切り替える
- 設定
  - 初期化パラメータで有効/無効の設定
  - デフォルト有効
- レポート表示
  - 適応計画によるプランの変化を確認可能



適応計画の仕組み

実際の条件に応じてサブプランを切り替える

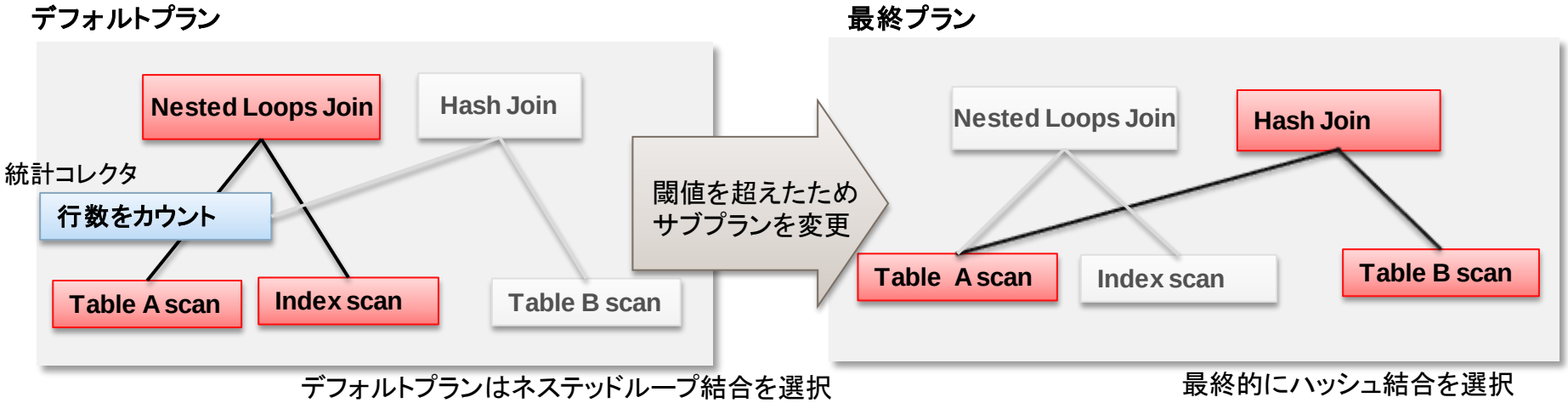
- サブプラン
  - 切り替え可能なプランの一部分
  - サブプランの種類
    - 結合方法
    - パラレル分散方法
- 切り替えの閾値はオプティマイザが決定
  - 例: order_items表から取り出す行数が10 以上なら Hash 結合を選択

```
SELECT product_name FROM order_items o,
product_information p WHERE o.unit_price = 15
AND p.product_id = o.product_id
```

適応計画の仕組み

動作イメージ

1. 既存の統計情報を基にデフォルトプランを作成
2. 統計コレクタが実際の統計(行数など)を取得
3. 取得した統計を基に使用するサブプランを選択し、最終プランを決定



適応計画の確認

- v$SQL ビューの IS_RESOLVED_ADAPTIVE_PLAN 列
  - 実行したSQL文のプランが適応計画の場合は値が入る
  - 適応計画ではない場合は NULL

```
SELECT sql_id, is_resolved_adapvtive_plan AP FROM v$sql WHERE sql_id=' 7zbft64brn1n4';
SQL_ID          AP
-------------   ---
7zbft64brn1n4   Y
```

- プランの Note セクション
  - 適応計画の場合は Note セクションに表示

```
Note
-----
   - this is an adaptive plan
```

適応計画の制御

OPTIMIZER_ADAPTIVE_REPORTING_ONLY

- 初期化パラメータ OPTIMIZER_ADAPTIVE_REPORTING_ONLY で有効/無効を設定
  - デフォルトは有効 (設定値: FALSE)
  - ALTER SYSTEM、ALTER SESSION による設定が可能
- OPTIMIZER_ADAPTIVE_REPORTING_ONLY を TRUE に設定した場合
  - レポートのためのモードとなり、プランの変更は行わない
  - 適応計画に必要な統計の収集は実施

適応計画 実行例

デフォルトプランの確認

- SQL文

```
SELECT /*+DEMO */ product_name FROM order_items o,
product_information p WHERE o.unit_price = 15 AND p.product_id =
o.product_id;
```

- デフォルトプランの確認

```
EXPLAIN PLAN FOR <SQL文>;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY(FORMAT=>'BASIC +NOTE'));

Plan hash value: 1255158658
-------------------------------------------------------------------
| Id  | Operation                    | Name                       |
-------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                            |
|   1 |  NESTED LOOPS                |                            |
|   2 |   NESTED LOOPS               |                            |
|   3 |    TABLE ACCESS FULL         | ORDER_ITEMS                |
|   4 |    INDEX UNIQUE SCAN         | PRODUCT_INFORMATION_PK     |
|   5 |   TABLE ACCESS BY INDEX ROWID| PRODUCT_INFORMATION        |
-------------------------------------------------------------------
Note
-----
   - this is an adaptive plan
```

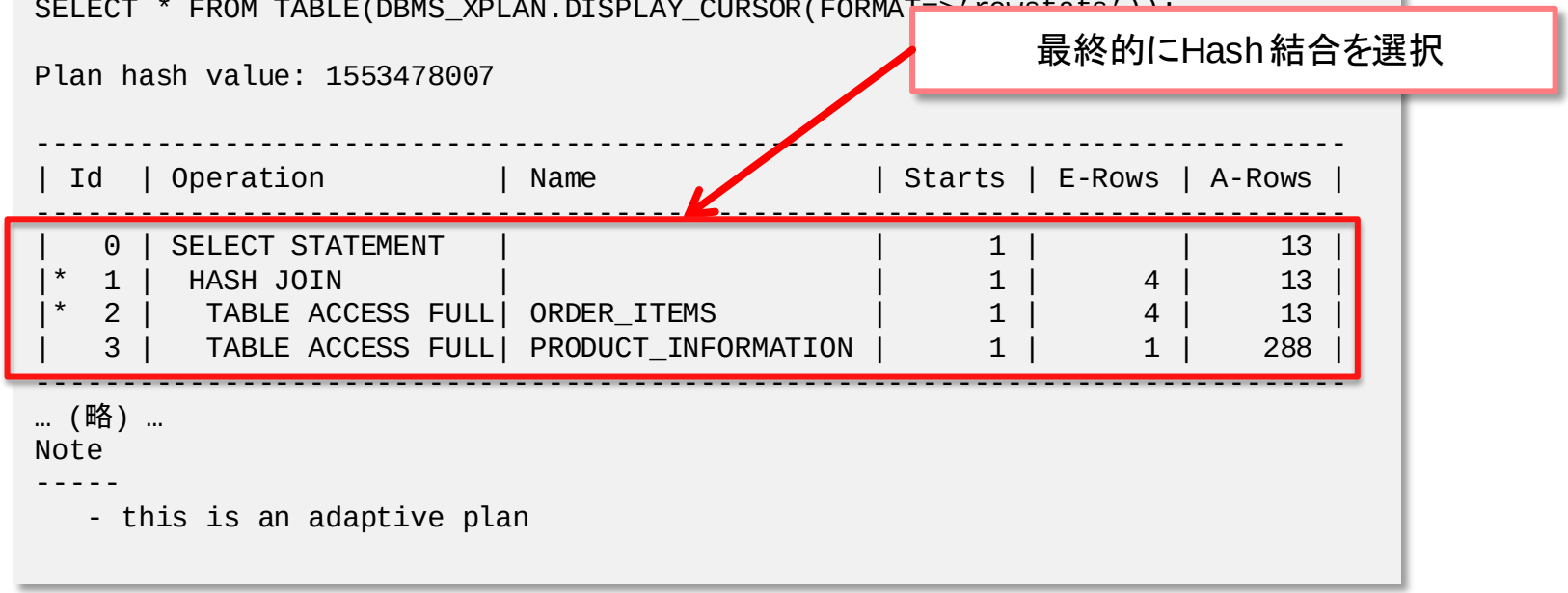
デフォルトプランはネステッド・ループ結合を選択

適応計画 実行例

SQL 実行とファイナルプランの表示

- SQL 文の実行後、ファイナルプランを確認

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(FORMAT=>'rowstats'));

Plan hash value: 1553478007

-------------------------------------------------------------------------------
| Id  | Operation          | Name                | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |                     |      1 |        |     13 |
|*  1 |  HASH JOIN         |                     |      1 |      4 |     13 |
|*  2 |   TABLE ACCESS FULL| ORDER_ITEMS         |      1 |      4 |     13 |
|   3 |   TABLE ACCESS FULL| PRODUCT_INFORMATION |      1 |      1 |    288 |
-------------------------------------------------------------------------------
… (略) …
Note
-----
   - this is an adaptive plan
```



適応計画のレポート出力

- プランの適応内容を表示
  - デフォルトプランと適応後のプランを表示
- 実行時のプランを変更せずに、適応によるプランの変化を確認可能
- 表示方法
  1. レポート・モードに設定
     - 初期化パラメータOPTIMIZER_ADAPTIVE_REPORTING_ONLYをTRUEにセット
  2. SQL文 を実行
  3. DBMS_XPLAN.DISPLAY_CURSOR ファンクションの FORMAT パラメータに 'REPORT' をセットしてプランを表示

```
select * from table(dbms_xplan.display_cursor(format=>'report'));
```

適応計画のレポート出力

```
-------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name                    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                         |       |       |            |          |
|   1 |  NESTED LOOPS                 |                         |       |       |            |          |
|   2 |   NESTED LOOPS                |                         |     4 |   94  |     2   (0)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL          | ORDER_ITEMS             |     4 |   72  |     2   (0)| 00:00:01 |
|*  4 |    INDEX UNIQUE SCAN          | PRODUCT_INFORMATION_PK  |     1 |       |     0   (0)|          |
|   5 |   TABLE ACCESS BY INDEX ROWID | PRODUCT_INFORMATION     |     1 |   219 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

…(略)…

Adaptive plan:
-------------
This cursor has an adaptive plan, but adaptive plans are enabled for
reporting mode only.  The plan that would be executed if adaptive plans
were enabled is displayed below.

Plan hash value: 1255158658

-------------------------------------------------------------------------------------------------
| Id  | Operation           | Name                    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                         |       |       |     7 (100)|          |
|*  1 |  HASH JOIN          |                         |     4 |   128 |     7   (0)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL | ORDER_ITEMS             |     4 |    48 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL | PRODUCT_INFORMATION     |     1 |    20 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

デフォルトプランと適応した場合のプランを表示

適応計画のサブプランを出力

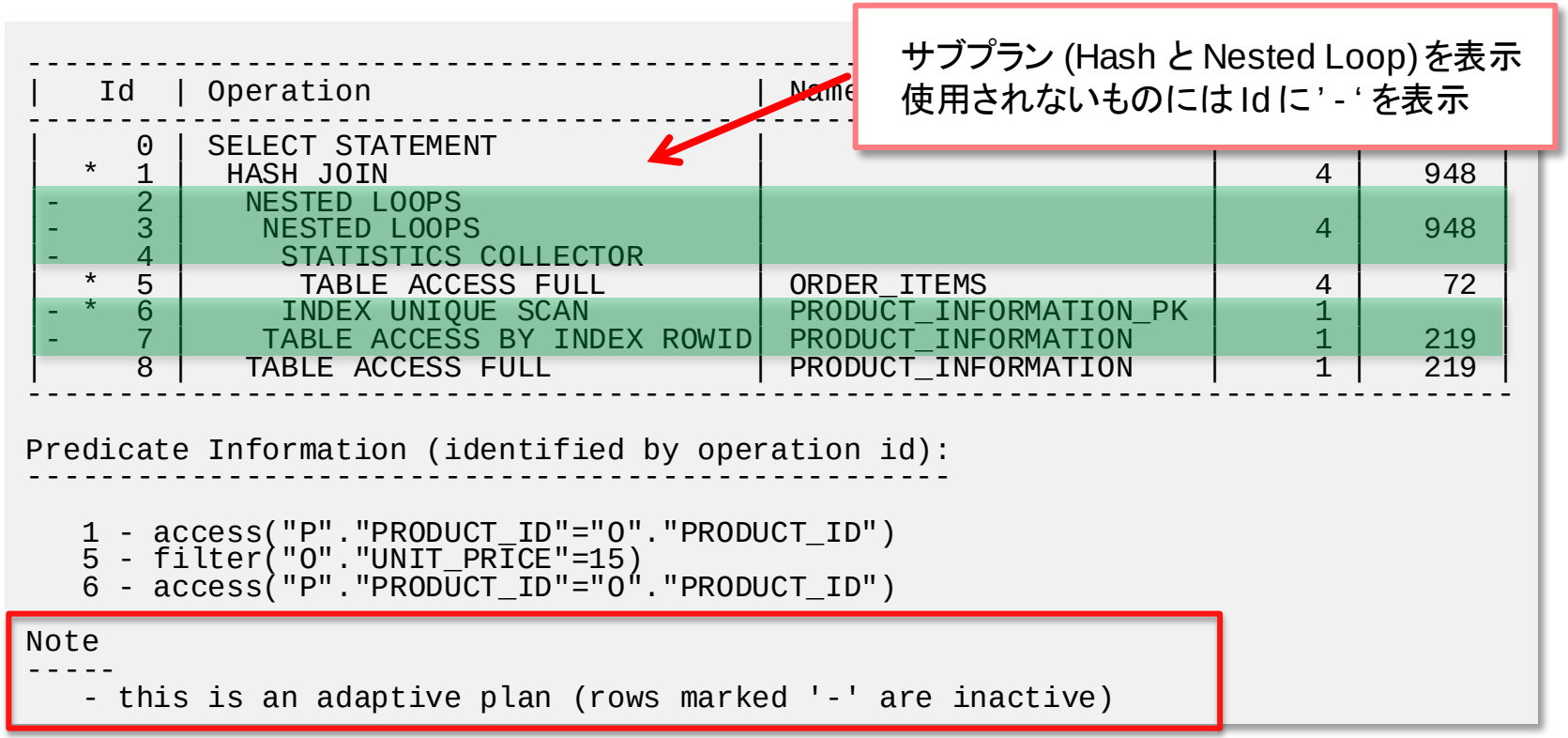
- サブプランを含んだプランを表示
  - 置き換えられる可能性のあるサブプランも表示
  - 使用されないサブプランを示す印 (-) を表示
- 作成されたすべてのサブプランを確認可能
- 表示方法
  - DBMS_XPLANファンクションのFORMATパラメータに'ADAPTIVE'をセット

```
select * from table(dbms_xplan.display_cursor(format=>'adaptive'));
```

サブプランの出力

```
-------------------------------------------------------------------------------------
|   Id  | Operation                     | Name                     | Rows  | Bytes |
-------------------------------------------------------------------------------------
|     0 | SELECT STATEMENT              |                          |       |       |
|   *  1 |  HASH JOIN                    |                          |     4 |   948 |
|-    2 |   NESTED LOOPS                |                          |       |       |
|-    3 |    NESTED LOOPS               |                          |     4 |   948 |
|-    4 |     STATISTICS COLLECTOR      |                          |       |       |
|   *  5 |      TABLE ACCESS FULL        | ORDER_ITEMS              |     4 |    72 |
|- *  6 |     INDEX UNIQUE SCAN         | PRODUCT_INFORMATION_PK   |     1 |       |
|-    7 |    TABLE ACCESS BY INDEX ROWID| PRODUCT_INFORMATION      |     1 |   219 |
|     8 |   TABLE ACCESS FULL           | PRODUCT_INFORMATION      |     1 |   219 |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("P"."PRODUCT_ID"="O"."PRODUCT_ID")
   5 - filter("O"."UNIT_PRICE"=15)
   6 - access("P"."PRODUCT_ID"="O"."PRODUCT_ID")

Note
-----
   - this is an adaptive plan (rows marked '-' are inactive)
```

サブプランの出力

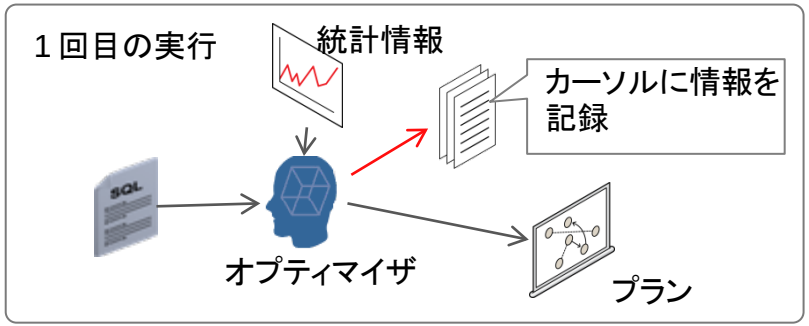


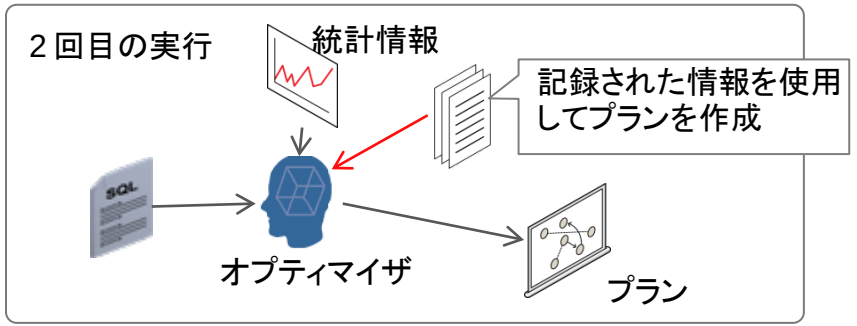
```
-------------------------------------------------------------------------------------------
|   Id  | Operation                      | Name                    | Starts | E-Rows |
-------------------------------------------------------------------------------------------
|     0 | SELECT STATEMENT               |                         |        |        |
|  *  1 |  HASH JOIN                     |                         |      4 |    948 |
|-    2 |   NESTED LOOPS                 |                         |        |        |
|-    3 |    NESTED LOOPS                |                         |      4 |    948 |
|-    4 |     STATISTICS COLLECTOR       |                         |        |        |
|  *  5 |      TABLE ACCESS FULL         | ORDER_ITEMS             |      4 |     72 |
|- *  6 |     INDEX UNIQUE SCAN          | PRODUCT_INFORMATION_PK  |      1 |        |
|-    7 |    TABLE ACCESS BY INDEX ROWID | PRODUCT_INFORMATION     |      1 |    219 |
|     8 |   TABLE ACCESS FULL            | PRODUCT_INFORMATION     |      1 |    219 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("P"."PRODUCT_ID"="O"."PRODUCT_ID")
   5 - filter("O"."UNIT_PRICE"=15)
   6 - access("P"."PRODUCT_ID"="O"."PRODUCT_ID")

Note
-----
   - this is an adaptive plan (rows marked '-' are inactive)
```


DEMONSTRATION

自動再最適化

- オプティマイザは次回以降の問合せ実行時にプランを適応
  - 実行時に収集した統計が見積もりと大きく異なる場合、その情報を記録
  - 次回の問合せ実行時に記録した統計を使用してプランを作成
- 適応計画では解決できない最適化も可能
  - 結合の順番など





自動再最適化

- 再最適化機能の種類
  - 統計フィードバック
    - 問い合わせ実行時に収集した統計を使用してプランを作成
    - 11g Release2 では カーディナリティ・フィードバックの名称
  - パフォーマンス・フィードバック
    - 収集した統計(例えば CPU time)を使用してパラレル度を改善
- 設定
  - 初期化パラメーターで有効/無効の設定
  - デフォルト有効
- レポートの表示
  - 自動再最適化によるプランの変化を確認可能

自動再最適化の確認

- v$SQLビューのIS_REOPTIMIZABLE列
  - 次回の問合せ実行時に再最適化による適応が行われるかを示す
    - Y: 次回にプランを再最適化する
    - N: 次回にプランを再最適化しない
    - R: レポートモードでコンパイルされたため、再最適化しない

- v$SQL_SHARED_CURSORビューのUSE_FEEDBACK_STATS列
  - 11g Release2と同様の動作
  - 次回の問合せ実行時に再解析を行う場合はYを格納

自動再最適化の制御

- 初期化パラメータ OPTIMIZER_ADAPTIVE_REPORTING_ONLY で有効/無効を設定
  - デフォルトは有効 (設定値: FALSE)
  - ALTER SYSTEM、ALTER SESSION による設定が可能
  - 適応計画と同じ初期化パラメーター
- OPTIMIZER_ADAPTIVE_REPORTING_ONLY を TRUE に設定した場合
  - レポートのためのモードとなり、プランの変更は行わない
  - 自動再最適化に必要な統計の収集は実施

自動再最適化の実行例

1回目の問合せ実行時に統計の乖離を認識

- SQL文の実行

```
SELECT /*TEST*/ o.order_id, v.product_name FROM   orders o,
       (  SELECT order_id, product_name  FROM   order_items o, product_information p
          WHERE  p.product_id = o.product_id   AND  list_price < 50  AND    min_price
< 40  ) v
WHERE  o.order_id = v.order_id;
```

見積もりと実際の統計が離れている

- プランを確認

```
-------------------------------------------------------------------------------------
| Id   | Operation             | Name                | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------------------------
|    0 | SELECT STATEMENT      |                     |        |        |    269 |
|    1 |  NESTED LOOPS         |                     |      1 |      1 |    269 |
|    2 |   MERGE JOIN CARTESIAN|                     |      1 |      4 |   9135 |
|*   3 |    TABLE ACCESS FULL  | PRODUCT_INFORMATION |      1 |      1 |     87 |
|    4 |    BUFFER SORT        |                     |     87 |    105 |   9135 |
|    5 |     INDEX FULL SCAN   | ORDER_PK            |      1 |    105 |    105 |
|*   6 |   INDEX UNIQUE SCAN   | ORDER_ITEMS_UK      |   9135 |      1 |    269 |
-------------------------------------------------------------------------------------
```

自動再最適化の実行例

2回目の問合せ実行時にプランを再最適化

- 2回目のSQL実行時のプラン

```
-------------------------------------------------------------------------------
| Id  | Operation             | Name                | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                     |      1 |        |    269 |
|   1 |  NESTED LOOPS         |                     |      1 |    269 |    269 |
|*  2 |   HASH JOIN           |                     |      1 |    313 |    269 |
|*  3 |    TABLE ACCESS FULL  | PRODUCT_INFORMATION |      1 |     87 |     87 |
|   4 |    INDEX FAST FULL SCAN| ORDER_ITEMS_UK     |      1 |    665 |    665 |
|*  5 |   INDEX UNIQUE SCAN   | ORDER_PK            |    269 |      1 |    269 |
-------------------------------------------------------------------------------

  …(略) …
Note
-----
   - statistics feedback used for this statement
```

実際の条件に適応した見積もりになった

プランが変更された

Note部分にstatistics feedbackの出力

自動再最適化の実行例

再最適化を行った子カーソルを作成

- 2回目のSQL実行後のカーソルの状態

```
select sql_id, PLAN_HASH_VALUE, child_number, is_reoptimizable
from v$sql where sql_text like '%ARTEST%';

SQL_ID                                 PLAN_HASH_VALUE CHILD_NUMBER IS_
-------------------------------------- --------------- ------------ ---
46cxwf85u43ga                            1906736282            0 Y
46cxwf85u43ga                              35479787            1 N
```

再最適化が終了した子カーソルは
IS_REOPTIMIZABLE列をNと表示

自動再最適化のレポート出力

- プランの再最適化内容を表示
  - 変更前と再最適化後のプランを表示
- 実行時のプランを変更せずに、再最適化によるプランの変化を確認可能
- 表示方法
  1. レポート・モードに設定
     - 初期化パラメータOPTIMIZER_ADAPTIVE_REPORTING_ONLYをTRUEにセット
  2. SQL文 を実行
  3. DBMS_XPLAN.DISPLAY_CURSOR ファンクションの FORMAT パラメータに 'REPORT' をセットしてプランを表示

```
select * from table(dbms_xplan.display_cursor(format=>'report'));
```

適応計画のレポート出力

```
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                         |      1 |     40 |            |
|   1 |  NESTED LOOPS               |                         |      1 |     40 |            |
|   2 |   MERGE JOIN CARTESIAN      |                         |      4 |    128 |            |
|*  3 |    TABLE ACCESS FULL        | PRODUCT_INFORMATION     |      1 |     28 |            |
|   4 |    BUFFER SORT              |                         |    105 |    420 |            |
|   5 |     INDEX FULL SCAN         | ORDER_PK                |    105 |    420 |            |
|*  6 |   INDEX UNIQUE SCAN         | ORDER_ITEMS_UK          |      1 |      8 |     0   (0)|            |
-------------------------------------------------------------------------------------------
--(略)--

Reoptimized plan:
-----------------
This cursor is marked for automatic reoptimization, but automatic
reoptimization is enabled for reporting mode only.  The plan that would
be selected on the next execution if automatic reoptimization were
enabled is displayed below.

Plan hash value: 35479787

-------------------------------------------------------------------------------------------
| Id  | Operation                   | Name                    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                         |   269 | 10760 |     7   (0)| 00:00:01 |
|   1 |  NESTED LOOPS               |                         |   269 | 10760 |     7   (0)| 00:00:01 |
|*  2 |   HASH JOIN                 |                         |   313 | 11268 |     7   (0)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL        | PRODUCT_INFORMATION     |    87 |  2436 |     5   (0)| 00:00:01 |
|   4 |    INDEX FAST FULL SCAN     | ORDER_ITEMS_UK          |   665 |  5320 |     2   (0)| 00:00:01 |
|*  5 |   INDEX UNIQUE SCAN         | ORDER_PK                |     1 |     4 |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

変更前と再最適化した場合のプランを表示

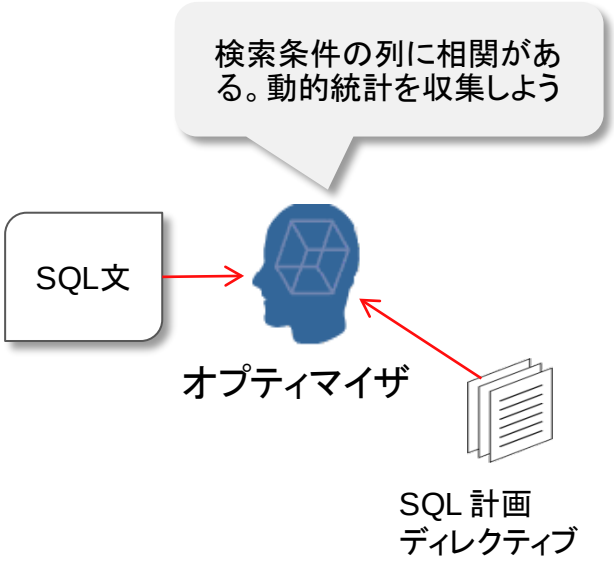
統計情報に関する新機能

統計情報に関する新機能

- SQL 計画ディレクティブ
- 動的統計
- バルク・ロードのオンライン統計収集
- 同時統計収集
- グローバル一時表のセッション固有統計
- ヒストグラムの強化

SQL 計画ディレクティブ

- オプティマイザがより適したプランを作成するための追加情報や指示
  - 動的統計の収集、列グループ統計の作成など
- SQL 実行時にカーディナリティが誤って見積もられた場合、ディレクティブを作成
- 作成したディレクティブは SYSAUX 表領域に格納し、永続化
- DBMS_SPD パッケージで管理
  - ディレクティブ保持期間の変更など



SQL 計画ディレクティブ

- 似たパターンのSQL文に対してディレクティブを使用
  - 特定のSQL文やSQL_IDに紐づきではない
  - 問合せ式で定義

ディレクティブを作成したSQL文

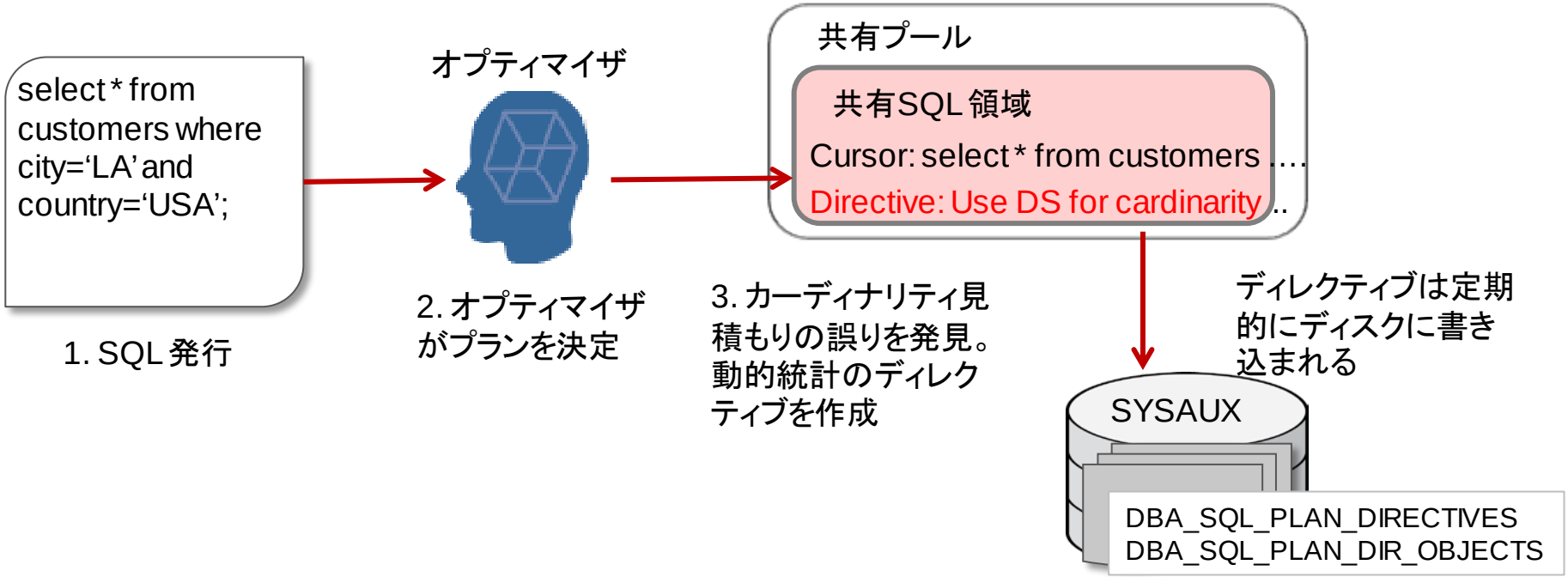
```
SELECT * FROM customers WHERE cust_state_province='CA' AND country_id=52790;
```

次のSQL文は作成済みのディレクティブを使用する

```
SELECT * FROM customers WHERE cust_state_province='MA' AND country_id=52790;
SELECT cust_city FROM customers WHERE cust_state_province='MA' AND country_id=52790;
```

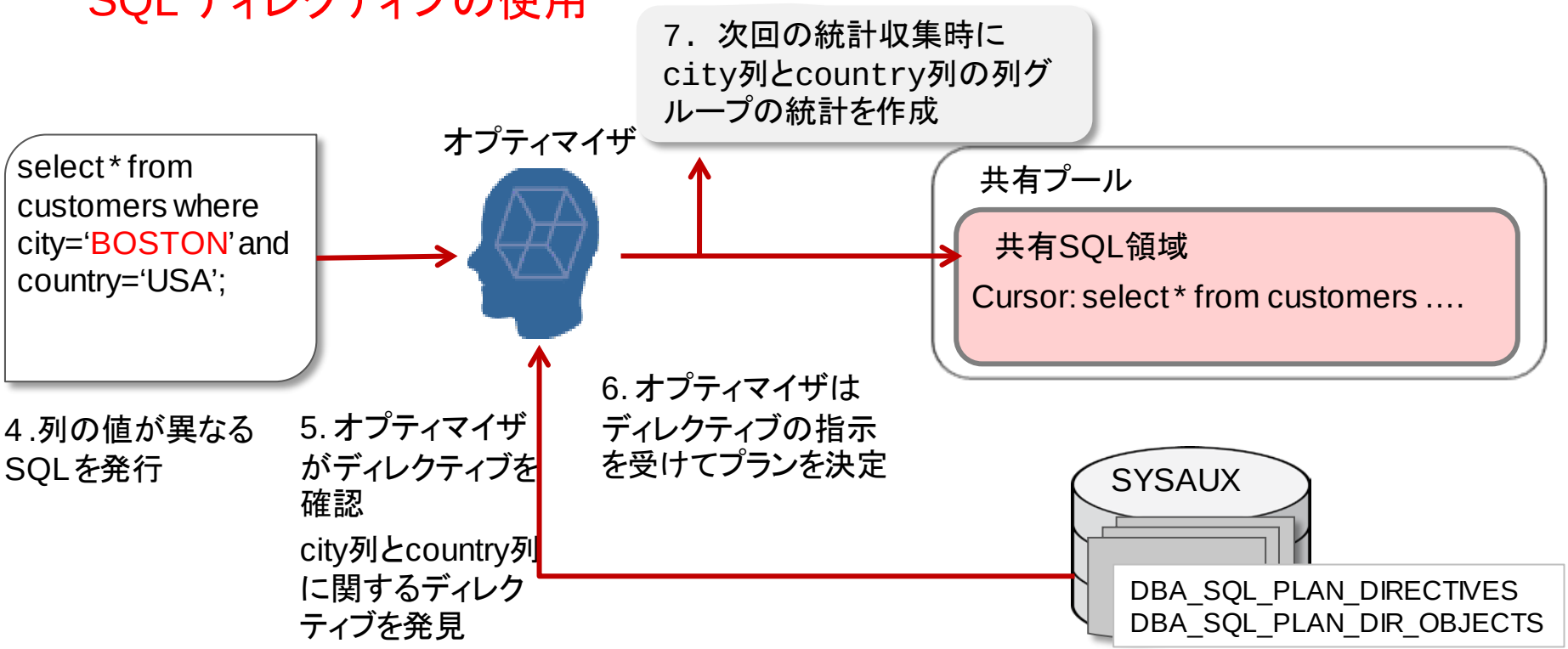
SQL 計画ディレクティブ 動作イメージ

SQL ディレクティブの作成



SQL 計画ディレクティブ 動作イメージ

SQL ディレクティブの使用



SQL 計画ディレクティブの確認

- ディレクティブの確認
  - DBA_SQL_PLAN_DIRECTIVES
  - DBA_SQL_PLAN_DIR_OBJECTS
- オブジェクト名、ディレクティブ作成理由、状態、最後の使用時間などを格納

```
OBJECT_NA SUBOBJECT_NAME      OBJECT TYPE              STATE REASON
--------- ------------------- ------ ----------------  ----- --------------------------------
CUSTOMERS CUST_STATE_PROVINCE COLUMN DYNAMIC_SAMPLING  NEW   SINGLE TABLE CARDINALITY MISESTIMATE
CUSTOMERS COUNTRY_ID          COLUMN DYNAMIC_SAMPLING  NEW   SINGLE TABLE CARDINALITY MISESTIMATE
CUSTOMERS                     TABLE  DYNAMIC_SAMPLING  NEW   SINGLE TABLE CARDINALITY MISESTIMATE
```

SQL 計画ディレクティブの管理

DBMS_SPDパッケージ

- ディレクティブの保持期間設定
  - 使用されないディレクティブは53週後に自動的にパージ(デフォルト)
  - DBMS_SPD.SET_PREFSプロシージャで期間を設定

例: 保持期間を1カ月(4週間)に設定

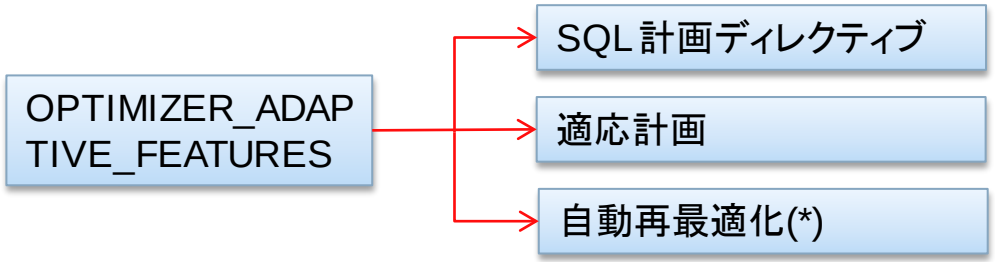
```
DBMS_SPD.SET_PREFS('SPD_RETENTION_WEEKS', '4');
```

- ディレクティブの削除
  - DROP_SQL_PLAN_DIRECTIVEプロシージャでdirective_idを指定

```
DBMS_SPD.DROP_SQL_PLAN_DIRECTIVE(<directive_id>);
```

SQL 計画ディレクティブの制御

- 初期化パラメータ OPTIMIZER_ADAPTIVE_FEATURES で有効/無効を設定
  - デフォルトは有効 (設定値: TRUE)
  - ALTER SYSTEM、ALTER SESSION による設定が可能
- 無効に設定した場合、他の適応機能も無効化される
  - 適応計画
  - 自動再最適化 (*)



(*) 現在のリリースでは自動再最適化は無効になりません

SQL 計画ディレクティブ実行例

SQL文の実行とプランの確認

- SQLの実行

```
SELECT /*+gather_plan_statistics*/ * FROM customers WHERE cust_state_province='CA' AND
country_id=52790;
```

- プランの確認

見積もりと実際の統計が大きく離れている
→ SQL 計画ディレクティブを作成

```
-------------------------------------------------------------------------------------
| Id  | Operation         | Name      | Starts | E-Rows | A-Rows |   A-Time    | Buffers |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |      1 |        |   3341 |00:00:00.02 |    1734 |
|*  1 |  TABLE ACCESS FULL| CUSTOMERS |      1 |     20 |   3341 |00:00:00.02 |    1734 |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter(("CUST_STATE_PROVINCE"='CA' AND "COUNTRY_ID"=52790))
```

SQL 計画ディレクティブ実行例

作成されたディレクティブを確認

- ディレクティブを確認

```
select o.object_name, o.subobject_name, o.object_type, d.type, d.state, d.reason
from dba_sql_plan_directives d, dba_sql_plan_dir_objects o where d.directive_id=o.directive_id and
o.owner = 'SH';


OBJECT_NA SUBOBJECT_NAME      OBJECT TYPE              STATE REASON
--------- ------------------- ------ ----------------  ----- --------------------------------
CUSTOMERS CUST_STATE_PROVINCE COLUMN DYNAMIC_SAMPLING  NEW   SINGLE TABLE CARDINALITY MISESTIMATE
CUSTOMERS COUNTRY_ID          COLUMN DYNAMIC_SAMPLING  NEW   SINGLE TABLE CARDINALITY MISESTIMATE
CUSTOMERS                     TABLE  DYNAMIC_SAMPLING  NEW   SINGLE TABLE CARDINALITY MISESTIMATE
```

SQL 計画ディレクティブ実行例

変数の値が異なるSQL文を実行

- 変数の値が異なるSQL文を実行

```
SELECT /*+gather_plan_statistics*/ * FROM customers WHERE cust_state_province='MA' AND
country_id=52790;
```

値をCAからMAに変更

```
-------------------------------------------------------------------------------------------
| Id  | Operation         | Name      | Starts | E-Rows | A-Rows |   A-Time    | Buffers |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |      1 |        |    181 |00:00:00.03 |    1535 |
|*  1 |  TABLE ACCESS FULL| CUSTOMERS |      1 |    208 |    181 |00:00:00.03 |    1535 |
-------------------------------------------------------------------------------------------
… (略) …
Note
-----
   - dynamic statistics used: dynamic sampling (level=3)
   - 1 Sql Plan Directive used for this statement
```

動的統計を実施してプランを作成

SQL 計画ディレクティブ実行例

列統計の作成

- 統計の収集

```
exec dbms_stats.gather_table_stats(user, 'CUSTOMERS');
```

- 統計の確認

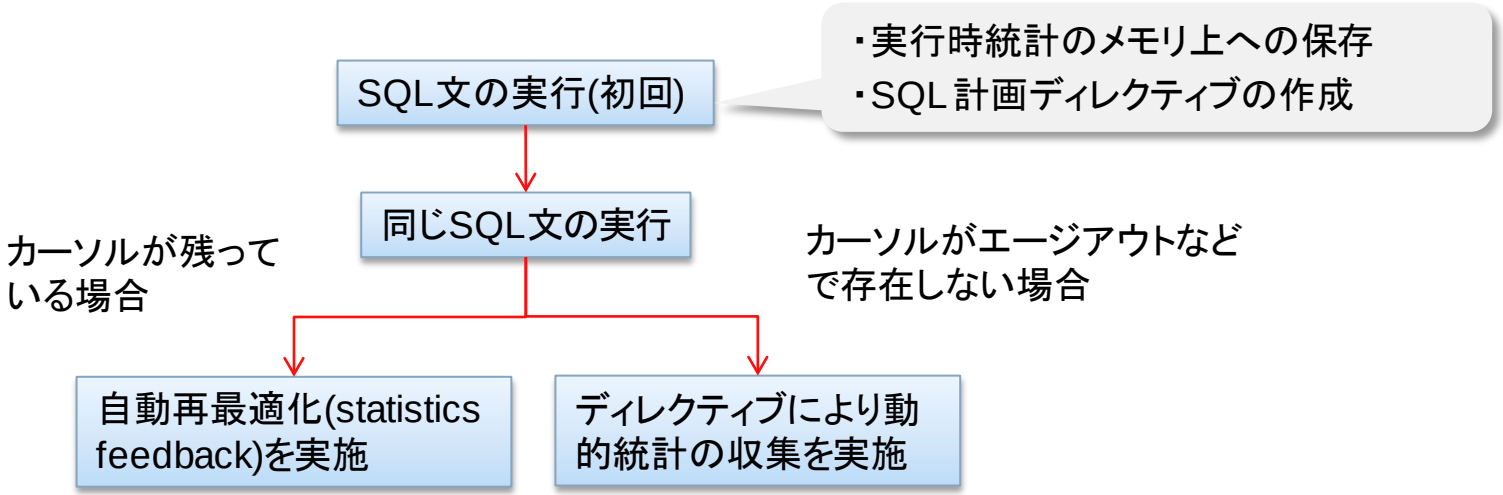
```
select table_name, extension_name, extension from user_stat_extensions;

CUSTOMERS
SYS_STS#S#WF25Z#QAHIHE#MOFFMM_
("CUST_STATE_PROVINCE","COUNTRY_ID")
```

ディレクティブにより列統計が作成された

SQL 計画ディレクティブによる永続化

SQL 計画ディレクティブと自動再最適化

- SQL文実行時に認識した統計が不十分であるという情報はディレクティブを使用して永続化される
- 自動再最適化の統計情報を持ったカーソルがエージアウトなどで存在しない場合、ディレクティブを使用して統計を補完



DEMONSTRATION

動的統計

- 動的サンプリングは12cから動的統計(Dynamic Statistics)という名称に変更
- 設定パラメータとヒントの名前は変更なし
  - 初期化パラメータ: OPTIMIZER_DYNAMIC_SAMPLING
  - ヒント: DYNAMIC_SAMPLING

動的統計の強化

- 以前のリリースの動的サンプリングでは、統計情報が欠落しているか、不十分な場合に統計情報を補完
- 12c では全てのSQLに対して動的統計が有用かどうか、どのレベルで使用するかを自動的に決定することが可能
- 収集した統計は永続的であり、他のクエリからも使用可能
- 設定
  - サンプリングレベルに11を導入

    初期化パラメーター OPTIMIZER_DYNAMIC_SAMPLING を 11 に設定
- 実行計画のNote部分にlevel=AUTOを表示

```
Note
-----
   - dynamic sampling used for this statement (level=AUTO)
```

バルク・ロードのオンライン統計収集

- 12cからバルク・ロード操作の中で統計の収集を自動的に実施
  - ユーザーによる統計収集を必要としない
  - 表統計収集のための追加の表スキャンをしない
- 以前のリリースではバルク・ロード後に手動で統計収集
- 以下のSQL文により実行
  - CREATE TABLE AS SELECT
  - INSERT INTO … SELECT (空テーブルへのダイレクト・パス・インサート)
- 表統計、列統計を収集
  - インデックス統計、ヒストグラムは収集しない

バルク・ロードのオンライン統計収集の設定

- 設定
  - デフォルト設定で有効
  - 無効にする場合、ヒントを設定

    NO_GATHER_OPTIMIZER_STATISTICS

```
CREATE TABLE T1 AS SELECT /*+NO_GATHER_OPTIMIZER_STATISTICS */ * FROM
SH.CUSTOMERS;
```

バルク・ロードのオンライン統計収集に関する注意

- 以下の条件の表はオンライン統計収集が行われない
  - SysのようなOracle所有のスキーマ
  - ネステッド表
  - 索引構成表(IOT)
  - 外部表
  - グローバルー時表(トランザクション固有ー時表)
  - バーチャル列を持つ表
  - 表プリファレンスのPUBLISHがFALSEにセットされた表
  - 表プリファレンスのINCREMENTALがTRUEにセットされたパーティション表に、パーティションを指定せずにバルク・ロードを実施した場合
    - 収集されない: INSERT INTO salses SELECT …
    - 収集される: INSERT INTO sales PARTITION (sales_q4_2000) SELECT …

バルク・ロードのオンライン統計収集 実行例

- SQLの実行

```
CREATE TABLE TEST AS SELECT * FROM SH.CUSTOMERS;
```

- 統計の確認

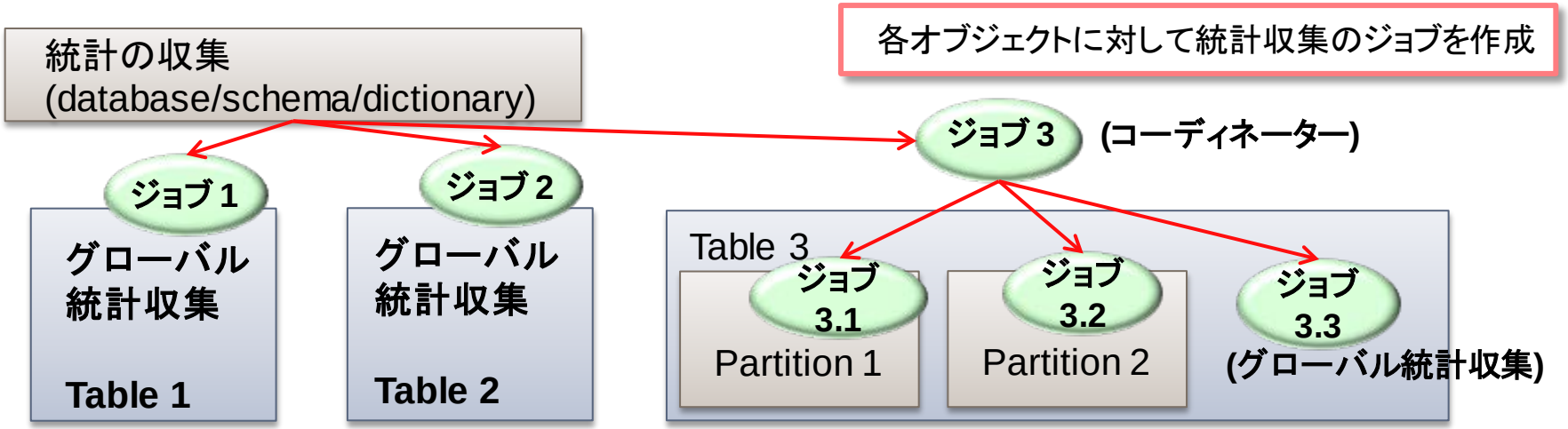
```
SELECT NUM_ROWS,BLOCKS FROM USER_TAB_STATISTICS WHERE TABLE_NAME='TEST';

 NUM_ROWS     BLOCKS
 ------------ ----------
        55500       1550

SELECT COLUMN_NAME, NUM_DISTINCT, NOTES FROM USER_TAB_COL_STATISTICS;

COLUMN_NAME  NUM_DISTINCT    NOTES
-----------  --------------  -------------------
CUST_ID               55500  STATS_ON_LOAD
```

自動的に統計を収集

DBA_TAB_COL_STATISTICSビューのNOTES列にSTATS_ON_LOADの値が入る

オプティマイザ統計の同時収集

- 複数のオブジェクトに対し統計情報を並行収集
  - 表、パーティション、サブパーティション
- マルチCPU環境では総収集時間の短縮が可能



オプティマイザ統計の同時収集

- 以前のリリース (11.2.0.2) から統計情報の同時収集を導入
- 12cでは以下の場合も並行収集が可能
  - 複数のパーティション表
  - 自動統計収集ジョブ
- データベースのジョブ機能を使用して統計を収集
  - Oracle Scheduler および Advanced Queuing を使用してジョブの作成、管理を実施
  - サイズが非常に小さい、あるいは空のパーティションは別の小さいオブジェクトと一つのジョブにまとめられる
- ジョブの数の制御にはパラメータ JOB_QUEUE_PRCESSESを使用可能

オプティマイザ統計の同時収集の設定

- 設定
  - DBMS_STATS.SET_GLOBAL_PREFファンクションを用いて、CONCURRENTプリファレンスを設定 (デフォルト 無効: OFF)

```
DBMS_STATS.SET_GLOBAL_PREFS('CONCURRENT',‘ALL’);
```

    - MANUAL: 手動統計収集時に同時収集
    - AUTOMATIC: 自動統計収集時に同時収集
    - ALL: 手動、自動統計収集時に同時収集
    - OFF: 同時収集しない (デフォルト)
- 注意
  - 同時統計収集の実行によるシステム・リソースの消費に注意
    - リソース・マネージャを使用したリソース管理

オプティマイザ同時統計収集の監視

- DBA_OPTSTAT_OPERATION_TASKS
  - ステータス、開始時間、終了時間の監視が可能

```
TARGET            TARGET_TYPE                JOB_NAME        STATUS       TO_CHHAR(START_T
----------------- -------------------------- --------------- ------------ ----------------
OPT.PT1           TABLE (COORDINATOR JOB)    ST$SD88_1       IN PROGRESS    03:08:06
OPT.PT2           TABLE                      ST$SD88_2_B2    IN PROGRESS    03:08:09
OPT.PT2.P3        TABLE PARTITION            ST$SD88_2_B2    IN PROGRESS    03:08:43
OPT.PT1.P4        TABLE PARTITION            ST$T128_2_B2    IN PROGRESS    03:08:07
OPT.PT1.P2        TABLE PARTITION            ST$T128_1_B3    IN PROGRESS    03:08:45
 …
```

グローバル一時表のセッション固有統計

グローバル一時表の概要

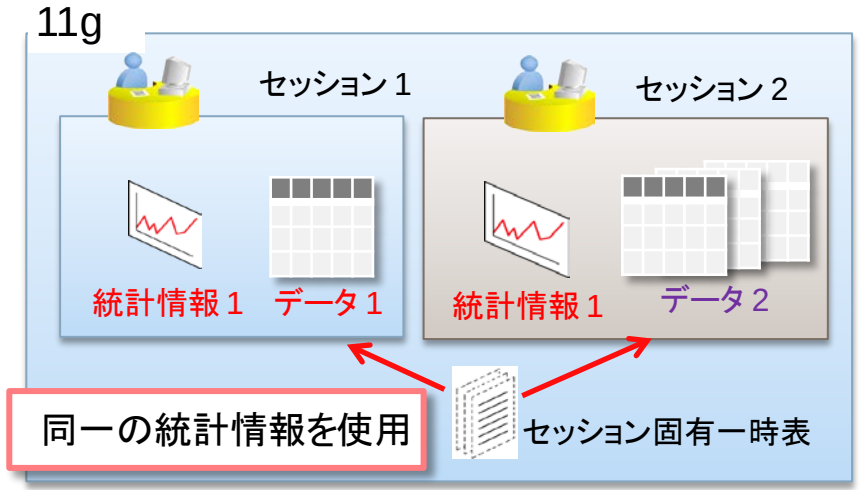
- グローバル一時表には二つの種類がある
  - セッション固有一時表
  - トランザクション固有一時表

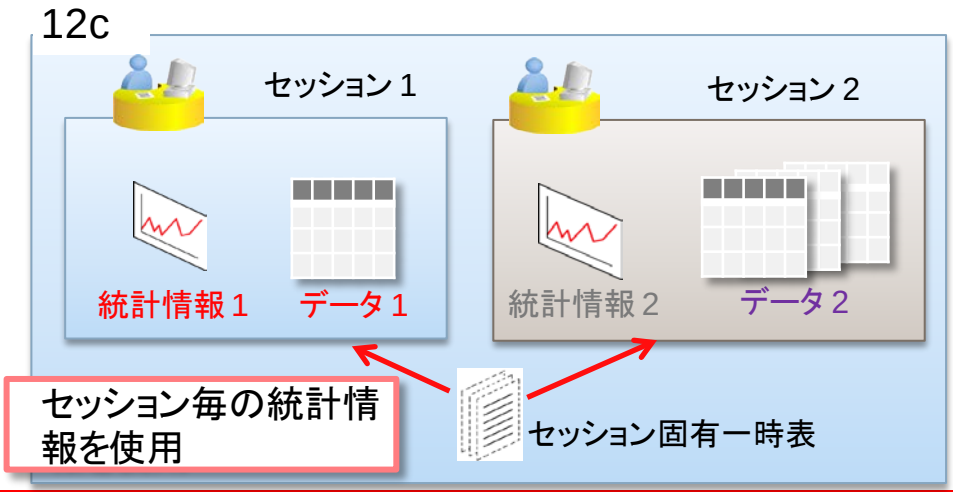
- セッション固有一時表はセッション中のみデータを保持する表
  - 一時表内のデータを参照、変更できるのはそのセッション自身のデータのみ
  - セッションの終了により、表内のデータを失う

- トランザクション固有一時表はトランザクション中のみデータを保持する表
  - トランザクションの終了により、表内のデータを失う

セッション固有一時表の統計収集

- 以前のリリースでは全てのセッションで一つの統計情報を共有して使用
- 12cではそれぞれのセッションで異なる統計情報を保持(デフォルト)
  - そのセッションで使用するデータに合わせた統計を収集可能





セッション固有一時表の統計収集

- デフォルトはセッション固有の統計を収集
- セッション固有の統計は他のセッションと共有されない
- 以前のリリース同様に一つの統計情報を全てのセッションで使用する設定も可能
  - GLOBAL_TEMP_TABLE_STATS プリファレンスで設定

```
DBMS_STATS.SET_GLOBAL_PREFS('GLOBAL_TEMP_TABLE_STATS', 'SHARED');
```

設定値

GLOBAL_TEMP_TABLE_STATSの値	有効化の設定
SESSION	セッション毎の統計情報を保持(デフォルト)
SHARED	セッション間で統計情報を共有

セッション固有一時表の統計収集

- 収集した統計は DBA_TAB_STATISTICSなどのディクショナリ・ビューから確認可能
  - 収集した統計の種類を表示するSCOPE列を導入

セッション固有統計のみ収集した状態 (デフォルト)

```
SELECT TABLE_NAME, NUM_ROWS, SCOPE FROM USER_TAB_STATISTICS WHERE TABLE_NAME
= 'TEST';

TABLE_NAME        NUM_ROWS     BLOCKS SCOPE
--------------- ---------- ---------- -------------------
TEST                                  SHARED
TEST                     5          1 SESSION
```


セッション固有一時表の統計収集 実行例

セッション毎に異なる統計を使用

- 異なるデータを持つセッション固有一時表での実行計画
  - セッション1に 5行、セッション2に 2行をINSERTし、統計収集を実施
  - それぞれのセッションでプランを確認

セッション1

```
-------------------------------------------------
| Id  | Operation          | Name  | Rows  |
-------------------------------------------------
|   0 | SELECT STATEMENT   |       |     5 |
|   1 |  TABLE ACCESS FULL| TEST  |     5 |
-------------------------------------------------

Note
-----
   - Global temporary table session private
statistics used
```

セッション2

```
-------------------------------------------------
| Id  | Operation          | Name  | Rows  |
-------------------------------------------------
|   0 | SELECT STATEMENT   |       |     2 |
|   1 |  TABLE ACCESS FULL| TEST  |     2 |
-------------------------------------------------

Note
-----
   - Global temporary table session private
statistics used
```

Noteセクションに出力

トランザクション固有一時表の統計収集

- 以前のリリースでは統計収集操作によりデータを失う動作
  - DBMS_STATS.GATHER_TABLE_STATSがCOMMIT処理を発行
  - COMMIT処理により表内のデータを失った状態で統計を収集
- 12cではDBMS_STATS.GATHER_TABLE_STATSを実行してもデータを失わずに統計を収集可能(デフォルト)
- 設定
  - DBMS_STATS.SET_GLOBAL_PREFSファンクションから設定
    - セッション固有一時表と同じ設定

```
DBMS_STATS.SET_TABLE_PREFS ('GLOBAL_TEMP_TABLE_STATS','SESSION');
```

トランザクション固有一時表の統計収集 実行例

- トランザクション固有一時表の統計収集
  - トランザクション固有一時表にデータをINSERTし、統計収集を実施
  - 統計情報の確認

12c のトランザクション固有一時表

```
TABLE_NAME   NUM_ROWS SCOPE
---------- ---------- -------
TEST                5 SESSION
```

11g のトランザクション固有一時表

```
TABLE_NAME   NUM_ROWS
---------- ----------
TEST                0
```

格納したデータを用いた統計が収集される

トランザクション固有一時表の統計収集

- トランザクション固有一時表に対してコミットを行わないよう拡張されたプロシージャ

- GATHER_TABLE_STATS
- DELETE_TABLE_STATS
- DELETE_COLUMN_STATS
- DELETE_INDEX_STATS
- SET_TABLE_STATS
- SET_COLUMN_STATS
- SET_INDEX_STATS
- GET_TABLE_STATS
- GET_COLUMN_STATS
- GET_INDEX_STATS

ヒストグラム

概要

- ヒストグラムは列データの分布状況を表し、データの偏りをオプティマイザに伝える
  - オプティマイザはヒストグラムを使用して最適なプランを選択
  - ヒストグラムが無い場合、オプティマイザはデータが均等に分布すると考えるため、適切な実行計画を選択できないことがある

C1列のデータ分布

C1	件数
A	1000件
B	1000件
C	1件

SQL文

```
SELECT .. FROM T
WHERE C1='C'
```

ヒストグラムが無い場合

個別値はA、B、C の3つ
データC は全体の 1/3 を占めていると推測
→ 表スキャンを選択

オプティマイザ

11g のヒストグラム

- 以前のリリースでは二種類のヒストグラムを使用
  - 頻度ヒストグラム (Frequency Histogram)
  - 高さ調整済ヒストグラム (Height-Balanced Histogram)

11g 頻度ヒストグラム

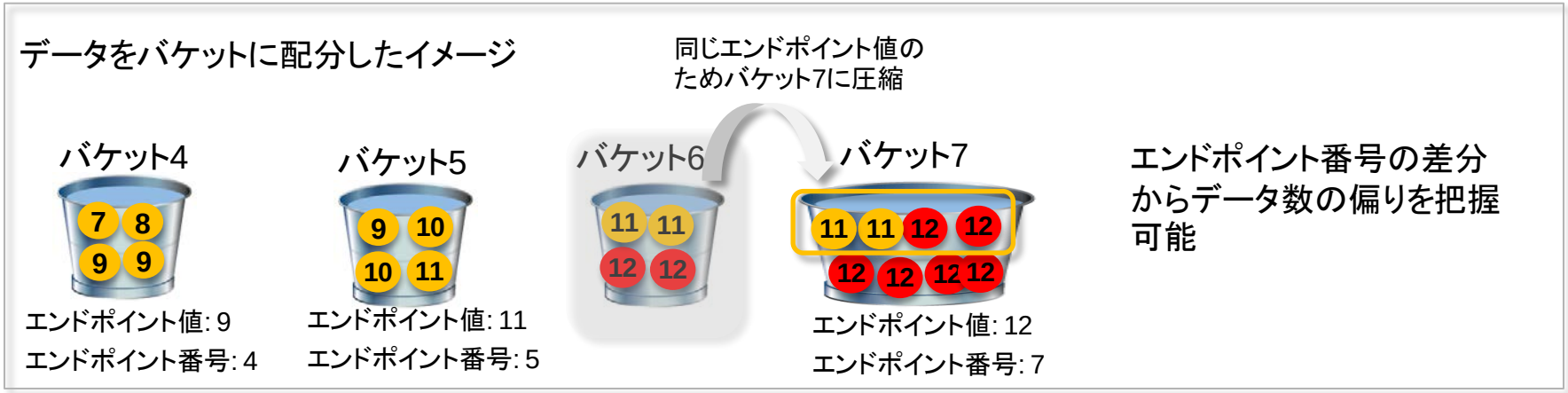
- 個別値の数がヒストグラム収集時に指定したバケット数以下の場合に生成
  - バケットは1から254まで指定可能
- 各バケットに同一のデータを格納
- データの個数はDBA_TAB_HISTOGRAMSビューのENDPOINT_NUMBER(エンドポイント番号)列の差分から把握

データをバケットに配分したイメージ

エンドポイント番号の差分から個数の把握が可能
e.g. 値3の個数は 12-5=7個

11g 高さ調整済みヒストグラム

- 個別値の数がバケット数より多い場合に作成
- 頻度ヒストグラムほど正確ではないがデータ数の偏りを把握可能
  - 各バケットにほぼ同数にデータを分配
  - 連続するバケットのエンドポイント値が同じ場合、最後のバケットに圧縮
  - データ数の偏りはエンドポイント番号列の差分から把握

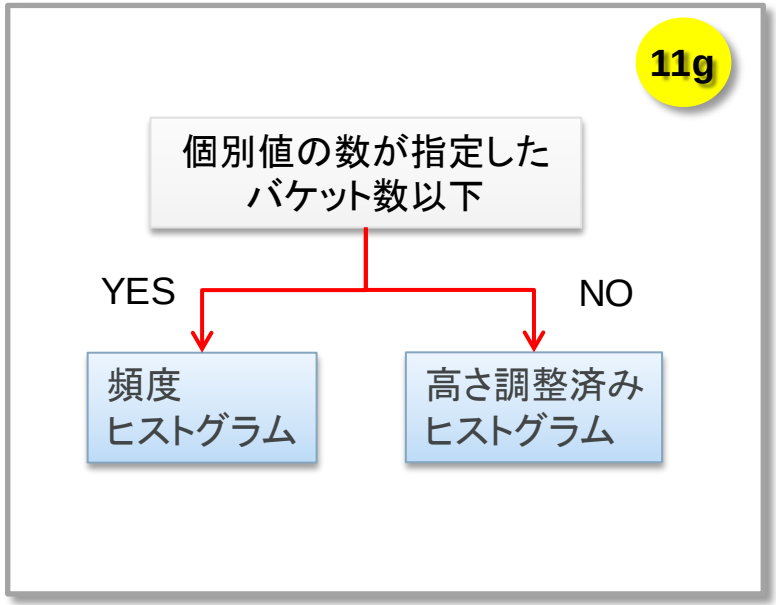


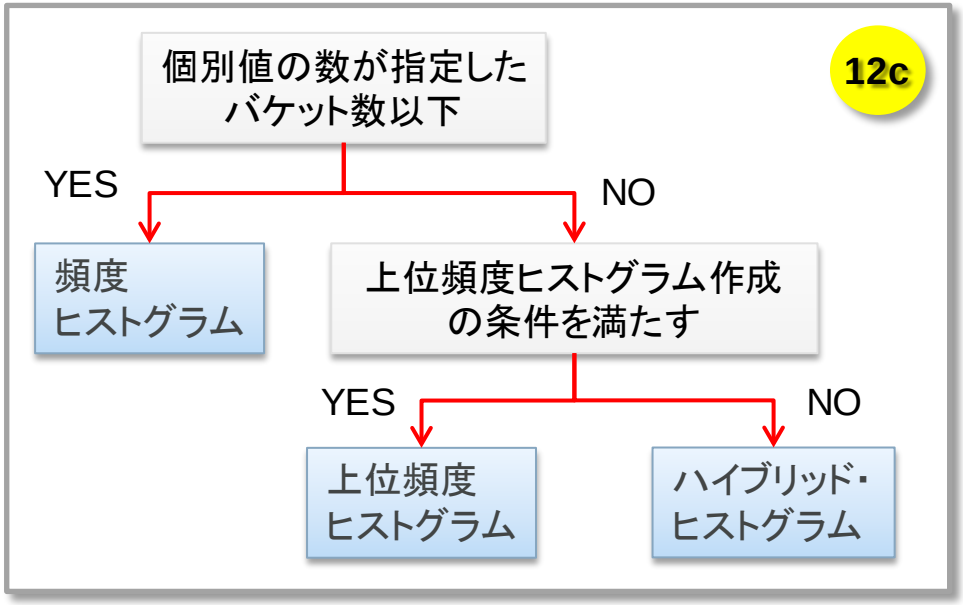
12c のヒストグラム

- より正確な見積もりを得るために新しいヒストグラムを導入
- 二つのヒストグラム
  - 上位頻度ヒストグラム
  - ハイブリッド・ヒストグラム

- バケットの数の最大値を2048に拡張
  - 以前のリリースでは254が最大値

作成されるヒストグラム





- 12c でも統計収集時のサンプリング率 (estimate_percent) をデフォルト値以外に設定した場合は高さ調整済みヒストグラムを作成する

上位頻度ヒストグラム

- 以前のリリースでは個別値の数がバケットの数以下の場合に頻度ヒストグラムを作成
- 12c では個別値の数がバケットの数より多くても、少数の個別値がデータの大部分を占める場合には上位頻度ヒストグラムを作成
- 上位頻度ヒストグラムでは上位n個の個別値に対してその個数を把握可能 (n=バケット数)
- データの大部分を占める上位n個の個別値に対してよりよい見積もりが可能になる

データをバケットに配分するイメージ

•各バケットに個別値が分配される
•個別値の数はエンドポイント番号列から把握が可能

•2の値は非ポピュラー値であり、統計上有意ではないため無視される

上位頻度ヒストグラム

- 作成条件
  - 統計収集のサンプリング率がデフォルト設定(AUTO_SAMPLE_SIZE)
  - 指定したバケット数より多い個別値を持つ
  - 上位 n 個のデータが占める割合(%)が閾値 p 以上になる(n=バケット数)
    - 閾値(p)はバケットの数をnとした場合、 p=(1-(1/n))*100

例. バケットの数が 254 (デフォルト) の場合
p = (1-(1/254))*100 → 99.6 %

上位 254 個の個別値の占める割合が上記 p の値以上の場合は、上位頻度ヒストグラムを作成

上位頻度ヒストグラム 作成例

- sh.countries表のcountry_subregion_id列
- 個別値の数 8個
- バケット数 6 で統計を収集
  - 閾値 p = (1-(1/6))*100 = 83.3%
  - 上位 6 個の占める割合 91.3%

上位頻度ヒストグラムを作成

COUNTRY_SUBREGION_ID列の個別値の数

```
COUNTRY_SUBREGION_ID COUNT(*)
-------------------- ----------
52792                         1
52793                         5
52794                         2
52795                         1
52796                         1
52797                         2
52798                         2
52799                         9
```

作成されたヒストグラム

USER_TAB_COL_STATISTICSビュー

```
COLUMN_NAME  NUM_BUCKETS HISTOGRAM
-----------  ----------- --------------
COUNTRY_SUB            6 TOP-FREQUENCY
REGION_ID
```

USER_HISTOGRAMSビュー

```
ENDPOINT_NUMBER ENDPOINT_VALUE
--------------- --------------
              1          52792
              6          52793
              8          52794
             10          52797
             12          52798
             21          52799
```

ハイブリッド・ヒストグラム

- 上位頻度ヒストグラムの条件から外れる場合に作成
- 高さ調整済ヒストグラムと頻度ヒストグラムの両方の特徴を組み合わせる
  - 高さ調整済みヒストグラム同様に各バケットに値を割り振る
  - その後、同じ値が複数のバケットに入らないように値を移動
  - 各バケットのエンドポイント値の個数をエンドポイント繰返しカウント列 (ENDOPOINT_REPEAT_COUNT) に記録
- オプティマイザはエンドポイント繰返しカウントを使用してエンドポイント値の個数を把握

データをバケットに分配したイメージ

バケット10

エンドポイント値: 9
エンドポイント繰返しカウント: 3

バケット18

エンドポイント値: 11
エンドポイント繰返しカウント: 7

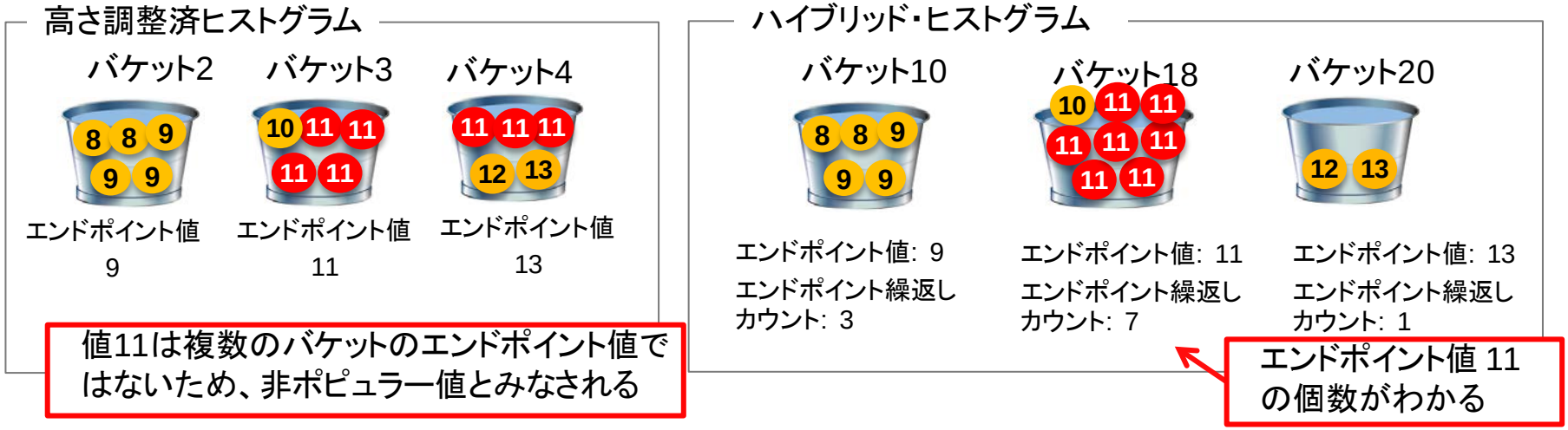
バケット20

エンドポイント値: 13
エンドポイント繰返しカウント: 1

ハイブリッド・ヒストグラム

高さ調整済ヒストグラムとの比較

- 高さ調整済ヒストグラムは、歪みがあってもポピュラー値と認識されない場合がある
  - (例) 二つのバケットの大部分を一つのデータが占めるが、エンドポイント値は一つ
- ハイブリッド・ヒストグラムではこのような値に対してよりよい見積もりが可能



ハイブリッド・ヒストグラム 作成例

- sh.products表のprod_subcategory_id列
- 個別値の数 22個
- バケット数 10で統計を収集
  - 閾値 p = (1-(1/10))*100 = 90%
  - 上位10個の占める割合 75%

ハイブリッド・ヒストグラムを作成

COUNTRY_SUBREGION_ID列の個別値の数

```
PROD_SUBCATEGORY_ID   COUNT(*)
------------------- ----------
               2011          1
               2012          2
               2013          2
               2014          8
                  …          …
               2056          5
```

作成されたヒストグラム

USER_TAB_COL_STATISTICSビュー

```
COLUMN_NAME NUM_BUCKETS HISTOGRAM
----------- ----------- --------------
PROD_SUBCAT  10 HYBRID
EGORY_ID
```

USER_HISTOGRAMSビュー

```
ENDPOINT_NUMBER ENDPOINT_VALUE ENDPOINT_REPEAT_COUNT
--------------- -------------- ---------------------
              1           2011                     1
             13           2014                     8
             26           2032                     6
             36           2036                     4
              …              …                     …
             72           2056                     5
```

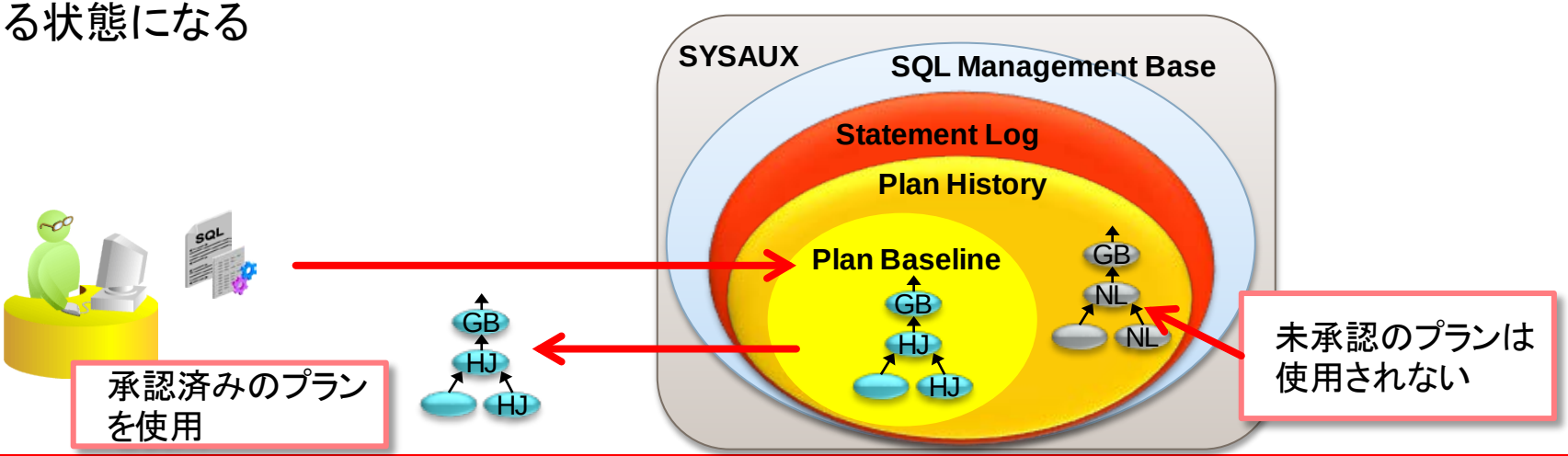
DEMONSTRATION

SQL 計画管理 (SPM) の新機能

SQL 計画管理

SQL Plan Management (SPM) 概要

- プランの変更による実行性能の低下を防止する
  - 使用するプランは計画ベースラインに登録
  - 新規のプランは計画履歴に追加
- 検証を受け、承認されたプランは計画ベースラインに登録され使用できる状態になる



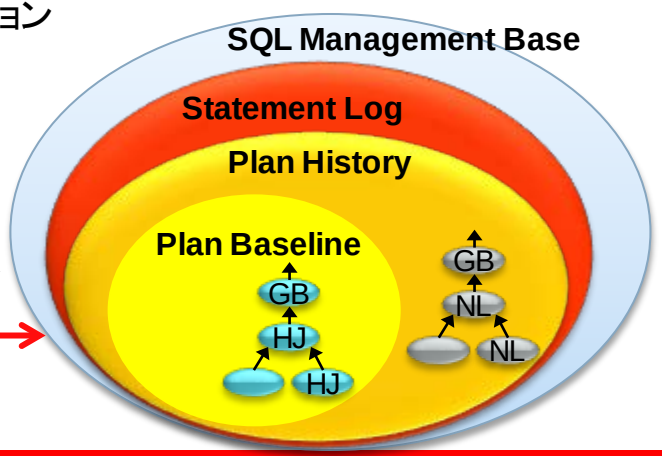
SQL 計画管理

11gのSQL計画ベースラインの改良(evolve)

- 新しいプランが検出された場合、承認済みプランとなる前に検証が必要
  - 新しく追加されたプランのパフォーマンスを評価
  - 検証されたプランを承認するかを管理者が決定
- 検証の実行
  - DBMS_SPM.EVOLVE_SQL_PLAN_BASELINEファンクション
  - SQL Tuning Advisor

パフォーマンスの比較検証

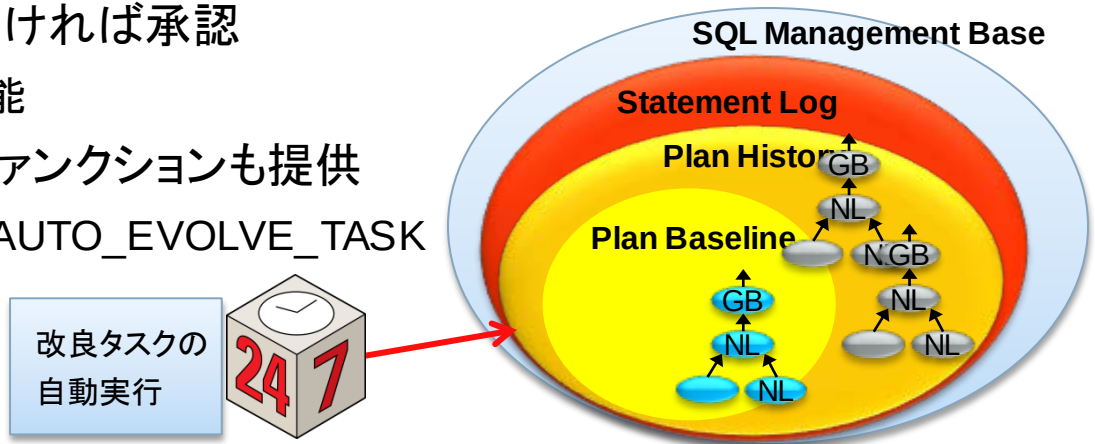
未承認のプランの方がよいパフォーマンスの場合、承認され、計画ベースラインに登録



適応 SQL 計画管理

SPM Evolve Advisor タスクの導入

- 12c ではプラン改良に関する新しいアドバイザータスクを導入 (SYS_AUTO_SPM_EVOLVE_TASK)
  - 自動メンテナンス・ウィンドウで検証を実施
  - 未承認のプランに対して検証プロセスを実行
  - 既存のプランより性能がよければ承認
    - 承認を行わない設定も可能
  - 評価レポートを確認するファンクションも提供
    - DBMS_SPM.REPORT_AUTO_EVOLVE_TASK



手動によるプラン改良

プラン改良タスクのファンクションを提供

- DBMS_SPMパッケージに手動実行可能なファンクションを提供
- 改良タスクの手動実行や自動実行のスケジュールを作成することが可能
  - e.g. 週次で改良タスクを実行し、レポートを確認してからプランを承認する

改良タスクのファンクション抜粋

ファンクション名	内容
CREATE_EVOLVE_TASK	改良タスクの作成、SQLハンドルやプラン名を指定可能
EXECUTE_EVOLVE_TASK	改良タスクの実行
REPORT_EVOLVE_TASK	改良タスクのレポートを表示
IMPLEMENT_EVOLVE_TASK	レポートの推奨するプランを承認

パラレル実行に関する新機能

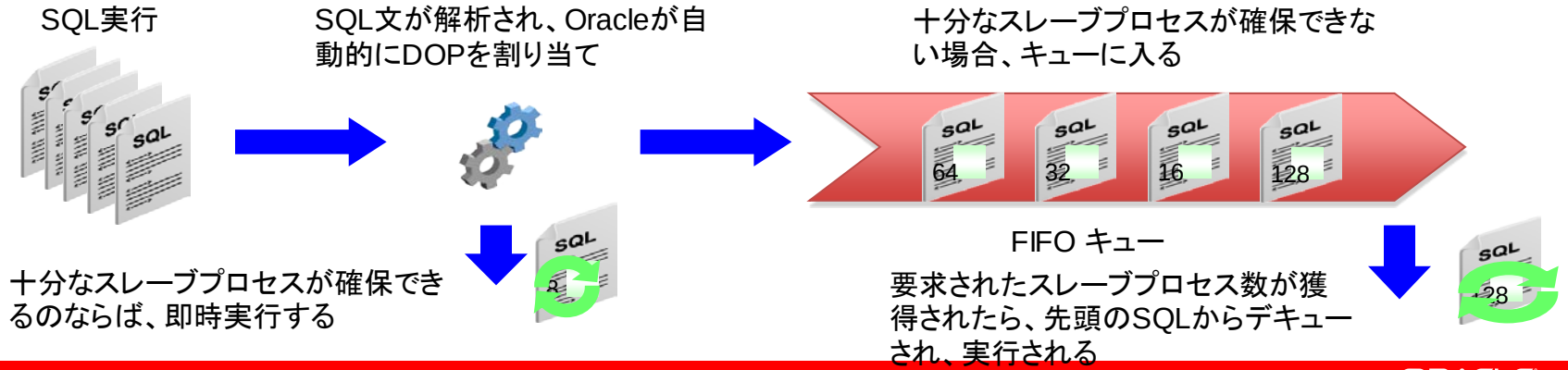
パラレル実行の強化

- パラレル文のキューイングの強化
- パラレル分散方法の適応
- パラレル度の再最適化

パラレル文のキューイング

11g Release2のキューイング

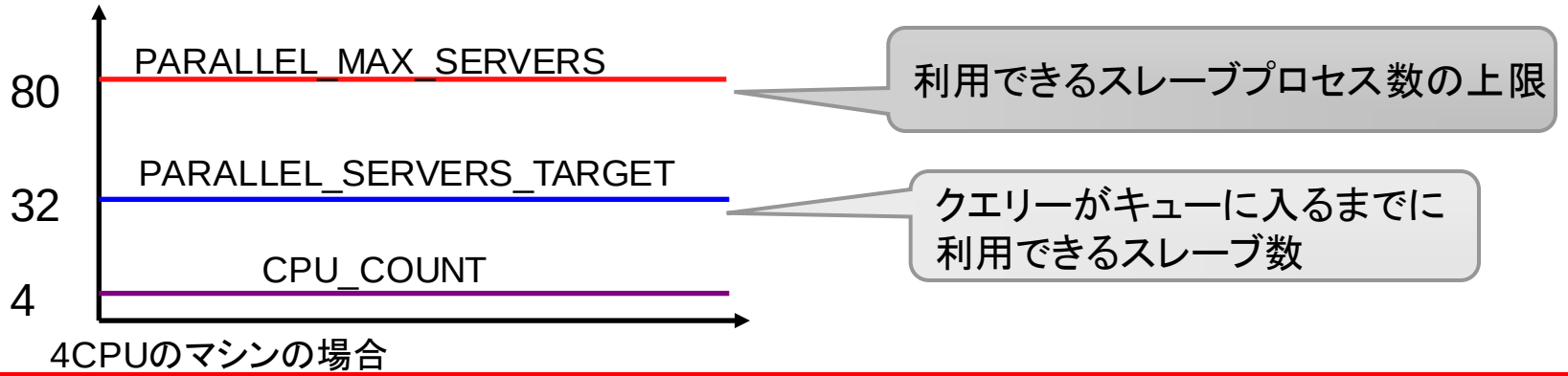
- 大量のパラレル実行が行われている場合に、パラレル実行のダウングレードを防止
- リソース使用状況によるパラレル度のダウングレードを回避するために実装
- 設定方法
  - ✓ PARALLEL_DEGREE_POLICYをAUTOに設定



パラレル文のキューイング

11g Release2のキューイング

- キューイングは一定の閾値を超える場合に行われる
  - 初期化パラメータPARALLEL_SERVERS_TARGETの値が閾値
  - 使用されているパラレル実行プロセスがこの値を超えるクエリーから、キューイングが行われる



パラレル文のキューイングの強化

12c クリティカルなパラレル文を優先的に実行

- 12c ではクリティカルなパラレル文の優先実行が可能
  - キューをバイパスして即時実行
  - parallel_servers_targetを超えて実行可能
- 設定
  - 特定のコンシューマー・グループをクリティカルとマーク
  - プラン・ディレクティブで設定

```
PARALLEL_STMT_CRITICAL => 'BYPASS_QUEUE'
```

- 確認
  - DBA_RSRC_PLAN_DIRECTIVES ビューのPARALLEL_STMT_CRITICAL列にBYPASS_QUEUEと表示

パラレル文のキューイングの強化

キューイング設定例

- コンシューマーグループの作成

```
BEGIN
  DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
  DBMS_RESOURCE_MANAGER.CREATE_PLAN('PQ_TEST_PLAN');
  DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP('PQ_CRITICAL_GROUP');

--  クリティカルなSQL実行用のプランを作成
  DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
    'PQ_TEST_PLAN','PQ_CRITICAL_GROUP', parallel_stmt_critical =>
'BYPASS_QUEUE');


 DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
    'PQ_TEST_PLAN', 'OTHER_GROUPS', 'no degree limit');
  DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
END;
/
```

パラレル文のキューイングの強化

キューイング設定例

- コンシューマー・グループの設定確認

```
SELECT PLAN, GROUP_OR_SUBPLAN, PARALLEL_STMT_CRITICAL FROM
DBA_RSRC_PLAN_DIRECTIVES WHERE PARALLEL_STMT_CRITICAL = 'BYPASS_QUEUE';

PLAN            GROUP_OR_SUBPLAN    PARALLEL_STMT_CRITICAL
--------------- ------------------- ----------------------------------
PQ_TEST_PLAN    PQ_CRITICAL_GROUP   BYPASS_QUEUE
```

パラレル分散方法の適応

実行時に収集した統計をベースに分散方法を決定

- パラレル実行ではソート、集計、結合などの操作のためにパラレル実行サーバー間でデータを再分配する
- Oracleデータベースには複数のパラレル分散方法が存在
  - e.g. ハッシュ方法、ブロードキャスト方法など
- 12c ではハイブリッド・ハッシュ分散方法を導入
  - 実行時に収集した統計から最終的な分散方法を決定
    - 実際の行数が閾値以上の場合はハッシュを使用
    - 閾値より少ない場合はブロードキャストに切り替わる
  - 閾値は並列度の2倍で定義

パラレル度の再最適化

パフォーマンス統計をフィードバックしてパラレル度を改善

- SQL実行時に収集した統計を使用して、次回実行時の自動パラレル度(AutoDOP) をより適切に決定
- 動作
  - 初回実行時にオプティマイザが並列度を決定
  - 実行の最後に以下を比較
    - オプティマイザが選択したパラレル度
    - 実行時に収集したパフォーマンス統計(例えばCPU Time)に基づいて計算したパラレル度

    両者が大きく異なる場合はその文を再解析の対象とし、統計を記録
  - 次回のSQL実行時に、記録した統計を使用してより適切なパラレル度を決定

パラレル度の再最適化

- 初期化パラメーターPARALLEL_DEGREE_POLICY を ADAPTIVE に設定
  - デフォルトは無効 (設定値: MANUAL)
  - ALTER SESSION, ALTER SYSTEMで設定の変更が可能

PARALLEL_DEGREE_POLICYの設定値

	MANUAL	LIMITED	AUTO	ADAPTIVE
自動パラレル度設定	×	○	○	○
In-Memory Parallel実行	×	×	○	○
Parallel Statementキューイング	×	×	○	○
Performance Feedback	×	×	×	○

Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®