Oracle Database Technology Night

津島博士のパフォーマンス講座
SQLパフォーマンスの基礎

日本オラクル株式会社
ソリューション・エンジニアリング統括
クラウド・インフラストラクチャー本部
津島 浩樹

2019/01/23

ORACLE

- 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。

アジェンダ

1. SQLとオプティマイザ
2. オプティマイザの動作
3. 最適化の戦略

SQLとオプティマイザ
SQLとは

プログラムとデータがそれぞれ独立性をもつことができるのが特長

- データを意識する必要がない
- パフォーマンスについてはそうはいかない

SQLとオプティマイザ
SQLパフォーマンスに影響するもの

- リレーショナル・データベース設計の中で、次の項目がパフォーマンスに大きな影響を与える（最適な実行計画の作成に影響する）
  - スキーマ設計
    - 単一列に複数の属性が含んでいるなど
  - SQL設計
    - 複雑さ（結合が多いなど）、Row by Row（1行ずつの処理）、使用するBIツールなど
  - 実行戦略（どう実行するか）
    - パラレル実行するかなど
  - 最適化テクニック（オプティマイザの機能）
    - どの機能を使用するか

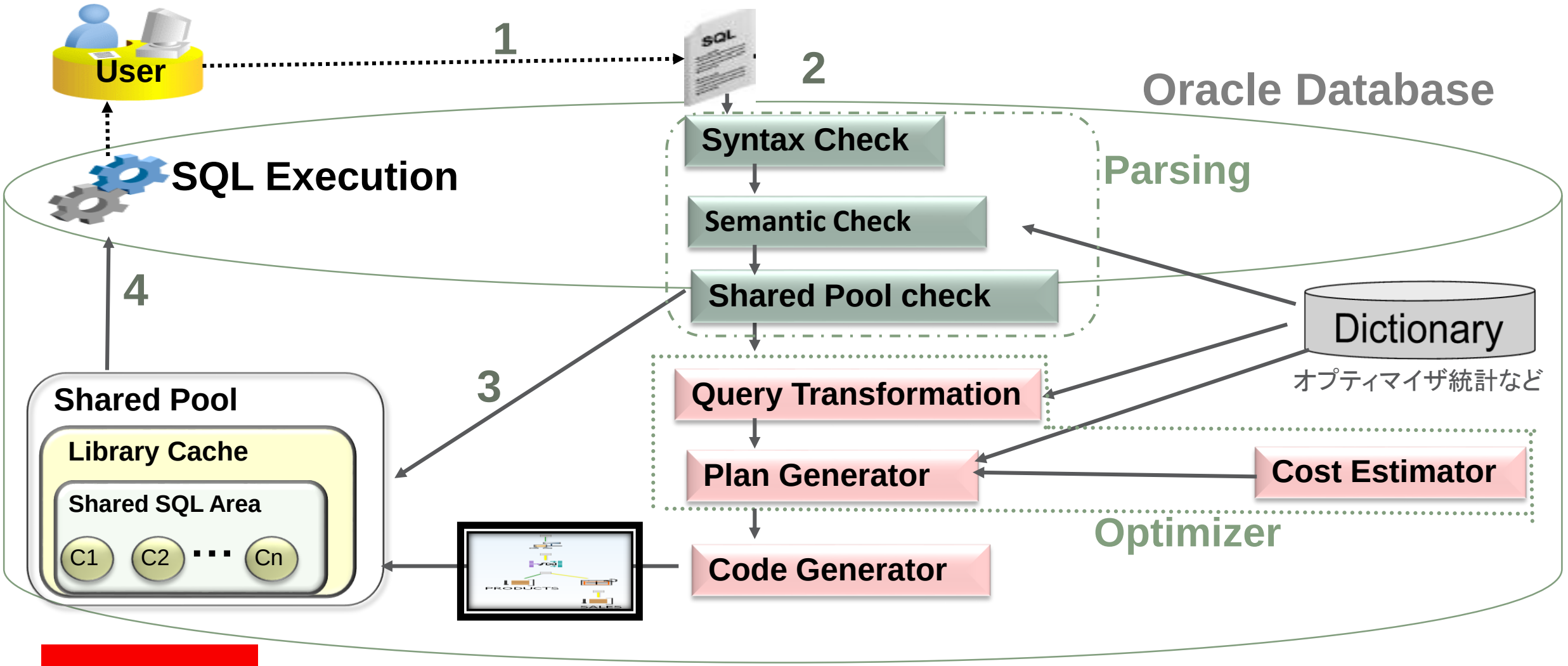
SQLとオプティマイザ
オプティマイザ

コストベース・オプティマイザはSQL実行のための戦略を選択する

- 重要な（パフォーマンスに影響する）決定
  - アクセス方法
  - 結合方法
  - 結合順序
  - パラレル実行の分散方法

SQLとオプティマイザ

SQLが発行されたときの動作



アジェンダ

1. SQLとオプティマイザ
2. オプティマイザの動作
3. 最適化の戦略

オプティマイザの動作
アクセス方法の決定

- 索引スキャン
  - 大規模表に対する少ない行のアクセス
    - カーディナリティ（述語を適用した行数）が少ないか
      - データの偏りや述語の複雑さが正確な見積りを難しくする
- 全表スキャン
  - 表の多くの行のアクセス
    - ダイレクト・パス・リード/非ダイレクト・パス・リード
      - 表サイズとバッファキャッシュ・サイズから

以下の時間で判断（カーディナリティ見積りが重要）
・全表のマルチ・ブロック・リード
・対象データのシングル・ブロック・リード
11gR2からのカーディナリティ・フィードバックで2回目から正確に

- 全索引スキャン（ソートが必要なとき）
- 高速全索引スキャン（索引のみのアクセスのとき）

SQL文で決まる

オプティマイザの動作
アクセス方法の決定（必要な情報）

- 表内の行数
- 索引の存在
- 表がパーティション化されている場合、アクセスするパーティションの数
- 表から取得する行数（カーディナリティ）の見積り
  - 列のNDV（Number of Distinct Values）
  - 列のヒストグラム（ない場合は最小値から最大値までの均等分布）
  - フィルター列のセレクティビティ（行の選択率）
    - NDVとヒストグラムから求める
      - ヒストグラムがないときの最小値から最大値間に対する均等条件のセレクティビティ= 1/NDV
      - 異なる値（NDV）が多いとフィルター述語の行数は少なくなり、NDVが少ないとフィルター述語の行数は多くなる

オプティマイザの動作
結合方法の決定

- ネステッド・ループ結合
  - 索引を使用する（**カーディナリティ**が少ない）結合
    - アクセス方法と同じ判断が必要
- ハッシュ結合
  - 索引を使用しない等価結合
  - ブルーム・フィルターが使用できる
- ソート・マージ結合
  - 索引を使用しない**非等価結合**
  - ソート処理が必要な等価結合でも

最も多いのが等価結合のため、この判断が重要。
12cからの適応結合方法で動的な判断が可能に

オプティマイザの動作
結合順序の決定

行数を早く減らして、実行中の処理を少なくする

1. それぞれの表の見積り行数（**カーディナリティ**）を求める
2. まず表カーディナリティの小さい順をベスト・コストの結合順序とする
3. ベストな結合順序のコストより少ない結合順序がないか調べる

- 結合列の重複値の数で結合後の行数（**結合カーディナリティ**）が増減する
  - 結合カーディナリティ=直積の行数*1/MAX(NDV(T1.c1),NDV(T2.c1))
- 数が多いと時間が掛かるので評価する結合順序数の最大値がある
  - OPTIMIZER_MAX_PERMUTATIONS（10g以降は2000固定）<=第24回
    - 7表では全組合せができない（7!=5040）
- そのため評価する数を少なく、コストが低いものが先になるようにしている
  - 結合する表の途中でコストが大きくなるとそれ以降は調べない
  - クロス結合（直積）を考慮する表数の上限を超えるとクロス結合を後回しにする
    - optimizer_search_limit（8i以降は5固定）

3表以上の結合は、結合カーディナリティの少ない順序を求める必要があるが、結合カーディナリティはオプティマイザ統計にない（この式で求めるだけ）

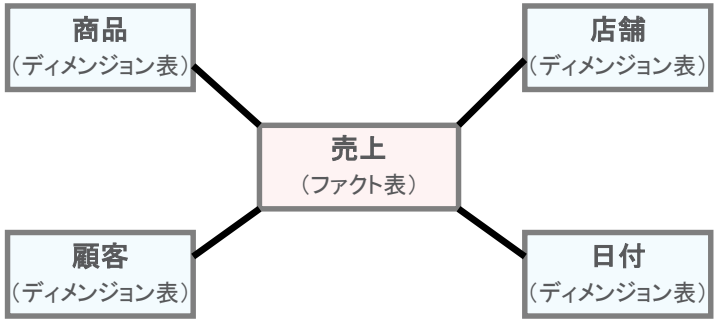
あまり問題にならない

オプティマイザの動作
結合順序の決定

- 途中でコストが大きくなるとそれ以降は調べない

t1→t2→t3→t4→t5→t6→t7（これがベスト・コストとして）
t4→t5→t6→t7→**t3**
t4→t5→t6→t3→**t7**
…
t4→**t6** (これ以降のt3,t5,t7の組合せは調べない)
t4 …
…

- クロス結合を後回し
  - クロス結合はコストが高くなるという考えから
  - 全ての組合せに結合条件を指定している訳ではない

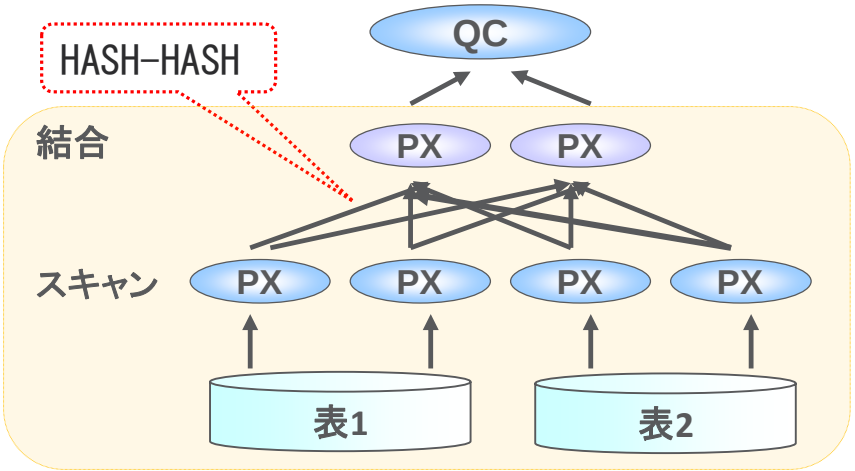


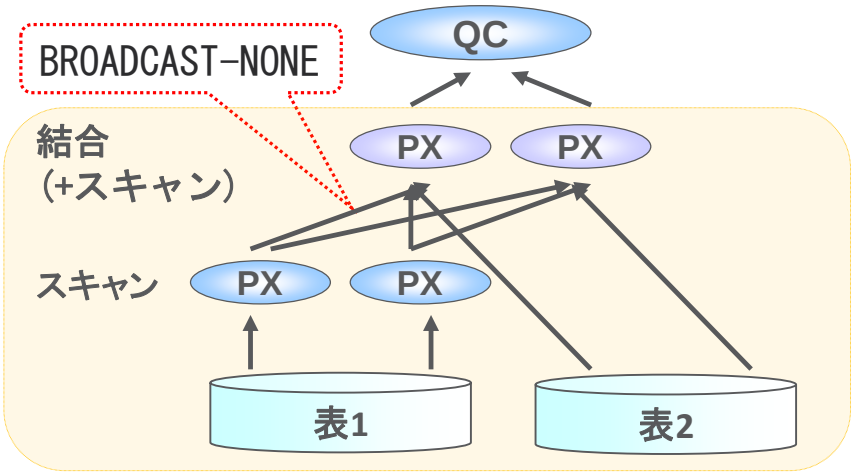
オプティマイザの動作
シリアル実行とパラレル実行

- シリアル実行
  - SQLは一つのプロセスで実行される
  - どのようなときに有効
    - 小さなデータ・セットを参照している問合せ
    - 同時実行性が高い
    - 効率が重要
- パラレル実行
  - SQLは連携して動作する多くのプロセスで実行される
  - どのようなときに有効
    - 大規模データ・セットを参照している問合せ（PGAを多く使用できる）
    - 同時実行性が低い
    - 経過時間が重要

オプティマイザの動作
パラレル実行（何が問題になるのか）

- データの偏りの問題
  - 非常に少ない値しかない場合のDISTINCT句など
    - 対応できない
  - ほとんどの行が特定（1つまたは2つ）の値を持つハッシュ結合
    - BROADCAST分散で対応
- 分散方法の問題
  - HASHとBROADCASTの選択ミス
    - 正しくない行数（**カーディナリティ**）の見積り
    - 値が少なすぎるとHASH分散が偏る



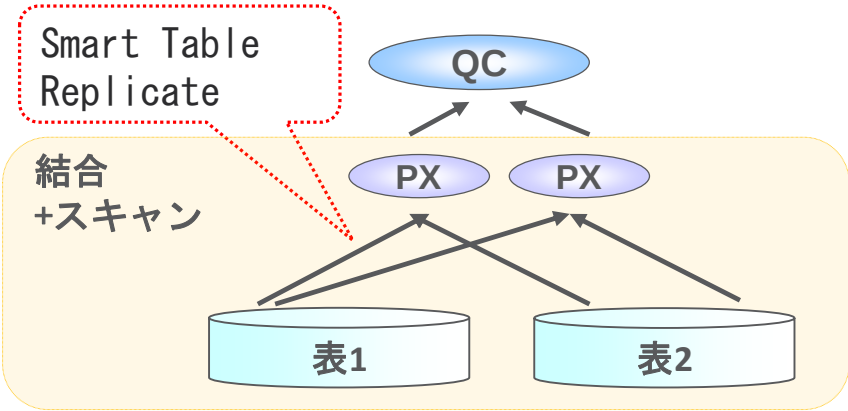


オプティマイザの動作
パラレル分散方法の決定（ハッシュ結合のとき）

- HASH-HASH
  - 重複値が少ないときに均等に分散
- BROADCAST-NONE
  - 重複値が多いときでも均等に分散
  - ただし、片方の結合対象が非常に少ないとき
    - 12cの適応パラレル分散方法で改善

パラレル実行で最も多いのがハッシュ結合のため、この判断が重要。
12cからの適応パラレル分散方法で動的な判断が可能に

- Smart Table Replicate（12cから）<=**第56回**
  - すべてのPXスキャン・プロセスが全表スキャンをする
  - 小さい表はこれになる（**この判断は難しくない**）
    - 表サイズは大きいが結合対象が少ないときはBROADCAST



オプティマイザの動作
パラレル分散方法の決定(HASH)

```
SQL> SELECT COUNT(*) FROM (SELECT * FROM t3,t1 WHERE t3.col1 = t1.col1);

実行計画
----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name     | Starts | E-Rows |  |    TQ  |IN-OUT| PQ Distrib | A-Rows |
----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |          |      1 |        |  |        |      |            |      1 |
|   1 |  SORT AGGREGATE          |          |      1 |      1 |  |        |      |            |      1 |
|   2 |   PX COORDINATOR         |          |      1 |        |  |        |      |            |      2 |
|   3 |    PX SEND QC (RANDOM)   | :TQ10002 |      0 |      1 |  | Q1,02  | P->S | QC (RAND)  |      0 |
|   4 |     SORT AGGREGATE       |          |      2 |      1 |  | Q1,02  | PCWP |            |      2 |
|*  5 |      HASH JOIN           |          |      2 |  50000 |  | Q1,02  | PCWP |            |    100K|
|   6 |       PX RECEIVE         |          |      2 |      5 |  | Q1,02  | PCWP |            |      5 |
|   7 |        PX SEND HASH      | :TQ10000 |      0 |      5 |  | Q1,00  | P->P | HASH       |      0 |
|   8 |         PX BLOCK ITERATOR|          |      2 |      5 |  | Q1,00  | PCWC |            |      5 |
|*  9 |          TABLE ACCESS FULL| T1       |      1 |      5 |  | Q1,00  | PCWP |            |      5 |
|  10 |       PX RECEIVE         |          |      2 |    100K|  | Q1,02  | PCWP |            |    100K|
|  11 |        PX SEND HASH      | :TQ10001 |      0 |    100K|  | Q1,01  | P->P | HASH       |      0 |
|  12 |         PX BLOCK ITERATOR|          |      2 |    100K|  | Q1,01  | PCWC |            |    100K|
|* 13 |          TABLE ACCESS FULL| T2       |     26 |    100K|  | Q1,01  | PCWP |            |    100K|
```

オプティマイザの動作
パラレル分散方法の決定（BROADCAST）

```
SQL> SELECT COUNT(*) FROM (SELECT * FROM t3,t1 WHERE t3.col1 = t1.col1);

実行計画
----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name     | Starts | E-Rows |  |    TQ  |IN-OUT| PQ Distrib | A-Rows |
----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |          |      1 |        |  |        |      |            |      1 |
|   1 |  SORT AGGREGATE          |          |      1 |      1 |  |        |      |            |      1 |
|   2 |   PX COORDINATOR         |          |      1 |        |  |        |      |            |      2 |
|   3 |    PX SEND QC (RANDOM)   | :TQ10001 |      0 |      1 |  | Q1,01  | P->S | QC (RAND)  |      0 |
|   4 |     SORT AGGREGATE       |          |      2 |      1 |  | Q1,01  | PCWP |            |      2 |
|*  5 |      HASH JOIN           |          |      2 |  50000 |  | Q1,01  | PCWP |            |   100K |
|   6 |       PX RECEIVE         |          |      2 |      5 |  | Q1,01  | PCWP |            |     10 |
|   7 |        PX SEND BROADCAST | :TQ10000 |      0 |      5 |  | Q1,00  | P->P | BROADCAST  |      0 |
|   8 |         PX BLOCK ITERATOR|          |      2 |      5 |  | Q1,00  | PCWC |            |      5 |
|*  9 |          TABLE ACCESS FULL| T1      |      1 |      5 |  | Q1,00  | PCWP |            |      5 |
|  10 |       PX BLOCK ITERATOR  |          |      2 |   100K |  | Q1,01  | PCWC |            |   100K |
|* 11 |        TABLE ACCESS FULL | T2       |     26 |   100K |  | Q1,01  | PCWP |            |   100K |
----------------------------------------------------------------------------------------------------------------------
```

行数×パラレル度

スキャンだけ分散（スキャンと結合のプロセスが同じ）

オプティマイザの動作
パラレル分散方法の決定（Smart Table Replicate）

```
SQL> SELECT COUNT(*) FROM (SELECT * FROM t3,t1 WHERE t3.col1 = t1.col1);

実行計画
----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name     | Starts | E-Rows |  |    TQ  |IN-OUT| PQ Distrib | A-Rows |
----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |          |      1 |        |  |        |      |            |      1 |
|   1 |  SORT AGGREGATE          |          |      1 |      1 |  |        |      |            |      1 |
|   2 |   PX COORDINATOR         |          |      1 |        |  |        |      |            |      2 |
|   3 |    PX SEND QC (RANDOM)   | :TQ10000 |      0 |      1 |  | Q1,00  | P->S | QC (RAND)  |      0 |
|   4 |     SORT AGGREGATE       |          |      2 |      1 |  | Q1,00  | PCWP |            |      2 |
|*  5 |      HASH JOIN           |          |      2 |  50000 |  | Q1,00  | PCWP |            |    100K|
|*  6 |       TABLE ACCESS FULL  | T1       |      2 |      5 |  | Q1,00  | PCWP |            |     10 |
|   7 |       PX BLOCK ITERATOR  |          |      2 |    100K|  | Q1,00  | PCWC |            |    100K|
|*  8 |        TABLE ACCESS FULL | T2       |     26 |    100K|  | Q1,00  | PCWP |            |    100K|
```

スキャンも分散していない

行数×パラレル度

オプティマイザの動作
良くない実行計画

- 良くない実行計画の多くは不十分なカーディナリティ見積りに起因する
- カーディナリティ見積りの検証は常に診断の出発点となる
  - 大きさの違いを確認する（E-RowsとA-Rows）
- 新しい統計を収集することが問題の範囲に適しているどうかを検討する
  - 問題のある問合せはひとつか、たくさんあるか
- 全ての不十分なカーディナリティ見積りが良くない実行計画になる訳ではない

オプティマイザの動作
重要な決定（まとめ）

- アクセス方法
  - データの偏りや複雑な述語に注意
  - 11gR2からのカーディナリティ・フィードバックで2回目から対応
- 結合方法
  - 12cからの適応結合方法で対応
- 結合順序
  - 結合数と結合カーディナリティに注意
- パラレル実行時の分散方法
  - 12cからの適応パラレル分散方法で対応

アジェンダ

1. SQLとオプティマイザ
2. オプティマイザの動作
3. 最適化の戦略

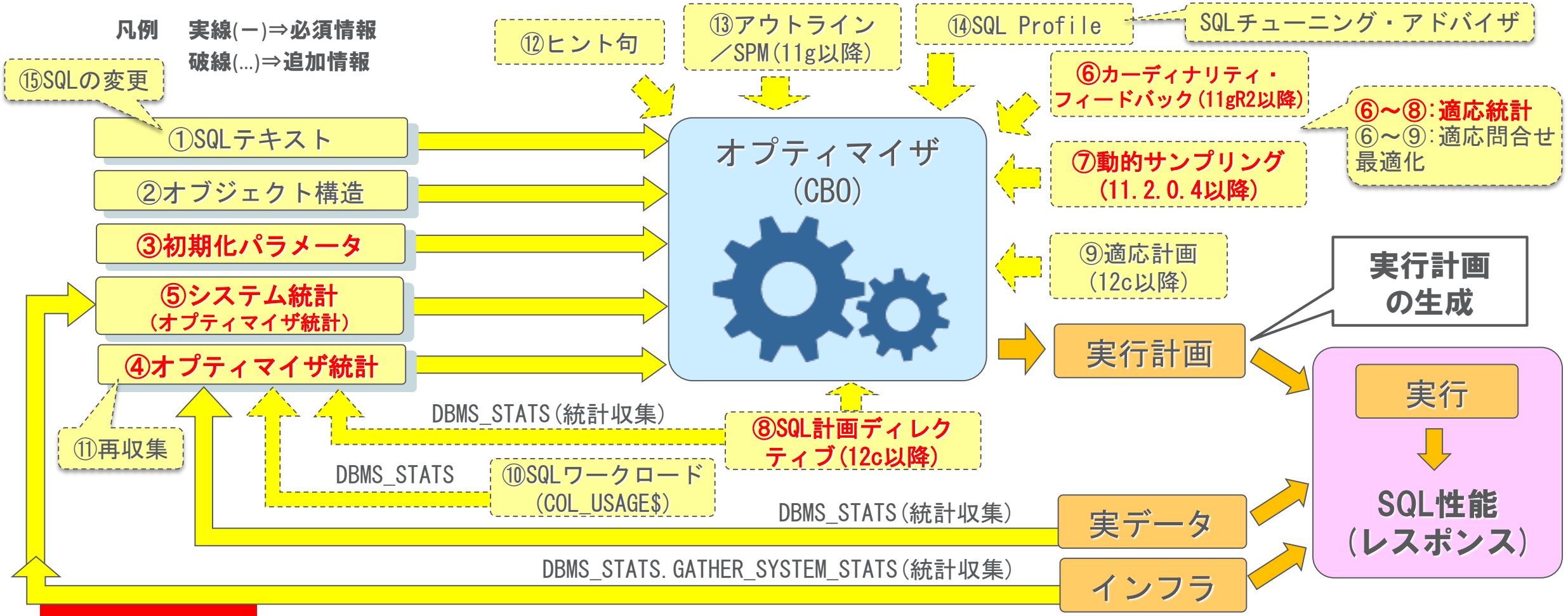
最適化の戦略
オプティマイザ

- オプティマイザの用途はSQL文の実行計画を策定すること
  - データとSQL文に関する情報
    - オプティマイザ統計、動的サンプリング、SQLプロファイルなど
- オプティマイザが適切な実行計画を選択しやすくする機能がいくつかある
  - オプティマイザ統計を正確にする機能など
  - どれを使用するかも重要

最適化の戦略

オプティマイザのイメージ(実行計画生成の仕組み)

- Oracle Databaseにおいては、Cost Base Optimizer (CBO) にSQLテキストや
  オプティマイザ統計等の様々な情報がインプットされて、実行計画が生成されます。



最適化の戦略
オプティマイザ

- オプティマイザはリリースごとに動的になっている
  - 見積りを正確にするため
- オプティマイザが使用する情報は次のカテゴリに分類できる
  - 初めて実行計画を立てるときに役立つもの
    - オプティマイザ統計（収集方法、ヒストグラム、拡張統計）、動的サンプリング、MView
  - 実行時に適切な実行計画を立てるのに役立つもの
    - 適応計画（適応結合方法、適応パラレル分散方法）
  - 後続の処理で正しい実行計画を立てるのに役立つもの
    - 適応統計（統計フィードバック、パフォーマンス・フィードバック、SQL計画ディレクティブ）
  - 適切な実行計画を立てられないときに役立つもの
    - ヒント、SQL計画管理（SPM）、SQLプロファイル
      - オプティマイザ統計では解決できないものやデフォルトがFALSEのものがある

最適化の戦略
オプティマイザの主な機能

バージョン	機能	備考
10g（コストベースのみに）	システム統計	CPUコスト・モデル（I/O回数のみから）
	自動オプティマイザ統計収集	多く変更した表だけ収集するように
	ヒストグラムの自動収集	col_usage$
	動的サンプリングの変更	デフォルトを2に（無いと動作）、一時表などに有効
	SQLプロファイル	SQLチューニング・アドバイザ
	プラン・スタビリティ	実行計画の固定化
11g	拡張統計	列グループ統計（col_group_usage$）、式統計
	AUTO_SAMPLE_SIZEの拡張	統計収集の精度と速さを向上
	増分統計	パーティション表の高速収集
	SPM	プラン・スタビリティの拡張

最適化の戦略
オプティマイザの主な機能（補正）

バージョン	機能	備考
11gR2	カーディナリティ・フィードバック	実行時の行数を記録
	動的サンプリングの拡張	パラレル実行時にレベル自動調整
12c	適応計画（適応問合せ最適化）	実行時に
	適応統計（適応問合せ最適化）	単一表カーディナリティ以外も可能に
	列統計の拡張	列サイズ（32バイトから64バイト） ヒストグラムの種類とバケット数（254から2048）
	増分統計の拡張	Exchange Partitionが可能に
	SPMの拡張	SPM展開アドバイザ
12cR2	適応統計の拡張	分離してデフォルトをFALSE（適応計画のデフォルトはTRUE）
	AUTO_STAT_EXTENSIONS	列グループ統計の自動作成を適応統計から分離
18c	パフォーマンス・フィードバック	適応統計から分離

最適化の戦略
オプティマイザ統計（収集方法）

精度の良い統計をできるだけ短時間で収集できるように

- DBMS_STATSパッケージを使用
- AUTO_SAMPLE_SIZEを使用
  - より高速で正確なNDVアルゴリズム
  - パーティション表の増分統計の使用を許可する
  - 同時統計収集の使用を許可する
  - 12cからの新しいヒストグラム・タイプを使用できるようにする
- 自動オプティマイザ統計収集
  - 必要なものだけを収集
  - 基本はこれを使用する

最適化の戦略
オプティマイザ統計（収集方法）

精度の良い統計をできるだけ短時間で収集できるように

- バルク・ロード
  - バルクロード（ダイレクト・パス）時にオンライン統計収集
- 増分統計
  - パーティション表のグルーバル（表レベル）統計
  - 表全体をアクセスしないで求める
    - パーティション・レベルのシノプシスを使用してNDVを求める
- 同時統計収集
  - パラレル収集が有効でないとき（サイズが小さい表など）

最適化の戦略
オプティマイザ統計（ヒストグラム）

- データ分布の詳細を提供するように設計されている
- SQLの使用法に基づいて自動的に作成される
  - オプティマイザは使用状況を確認する必要がある
  - 述語と結合で使用される列を追跡してsys.col_usage$に格納
    - 次回の統計収集時にヒストグラムが作成される
- 12cの新機能
  - 最大バケット数が2048（デフォルトは254のまま）
  - 全表スキャンを使用したAUTO_SAMPLE_SIZEで作成する（少ない値がより正確に）
    - ハッシュ・ベースから近似値（Approximate）のNDVアルゴリズムに変更することで全表スキャンが可能に

最適化の戦略
オプティマイザ統計（ヒストグラム）

- 種類（バケット数がデフォルトのとき）
  - 頻度ヒストグラム（Frequency）
    - NDV≦254の列の場合
  - 上位頻度ヒストグラム（Top-Frequency）<=12cから
    - NDV>254だが行の大部分の値の数≦254の列の場合（行の大部分の値の割合≧内部しきい値）
      - 内部しきい値：(1-(1/<バケット数>))*100
  - ハイブリッド・ヒストグラム（Hybrid）<=12cから
    - 上位頻度ヒストグラム以外の場合
  - 高さ調整ヒストグラム（Height Balanced）
    - NDV>254のときAUTO_SAMPLE_SIZE以外で作成された場合
- 見積もり行数が正しくないときにはバケット数を明示的に指定する
  - FOR ALL COLUMNS SIZE <バケット数>

最適化の戦略
オプティマイザ統計（拡張統計）

- 2種類の拡張統計
  - 列グループ統計
    - 同じ表の異なる列に格納されているデータ間の相関関係をオプティマイザに提供する
      - 例：'顧客'表の'住所'列と'年代'列など
      - 列グループ統計がないと相関関係がないものとして計算される
        - 等価条件のカーディナリティ=行数*1/NDV(住所)*1/NDV(年代)
    - SQL計画ディレクティブから自動生成（12cから）、自動列グループ検出（11gR2から）<=第49回
  - 式統計
    - 式の中に列が埋め込まれている述語のカーディナリティ見積りに役立つ
      - Where UPPER（氏名) = :B1
    - ファンクション索引を作成するか明示的に式統計を作成する必要がある（式の影響が分からない）
      - 仮想列を作成しても式統計が収集される（12cR2の自動インメモリ式は自動的に行う）
    - BIツールの使用でよく見る（BIツールで仮想列を使用しているなど）

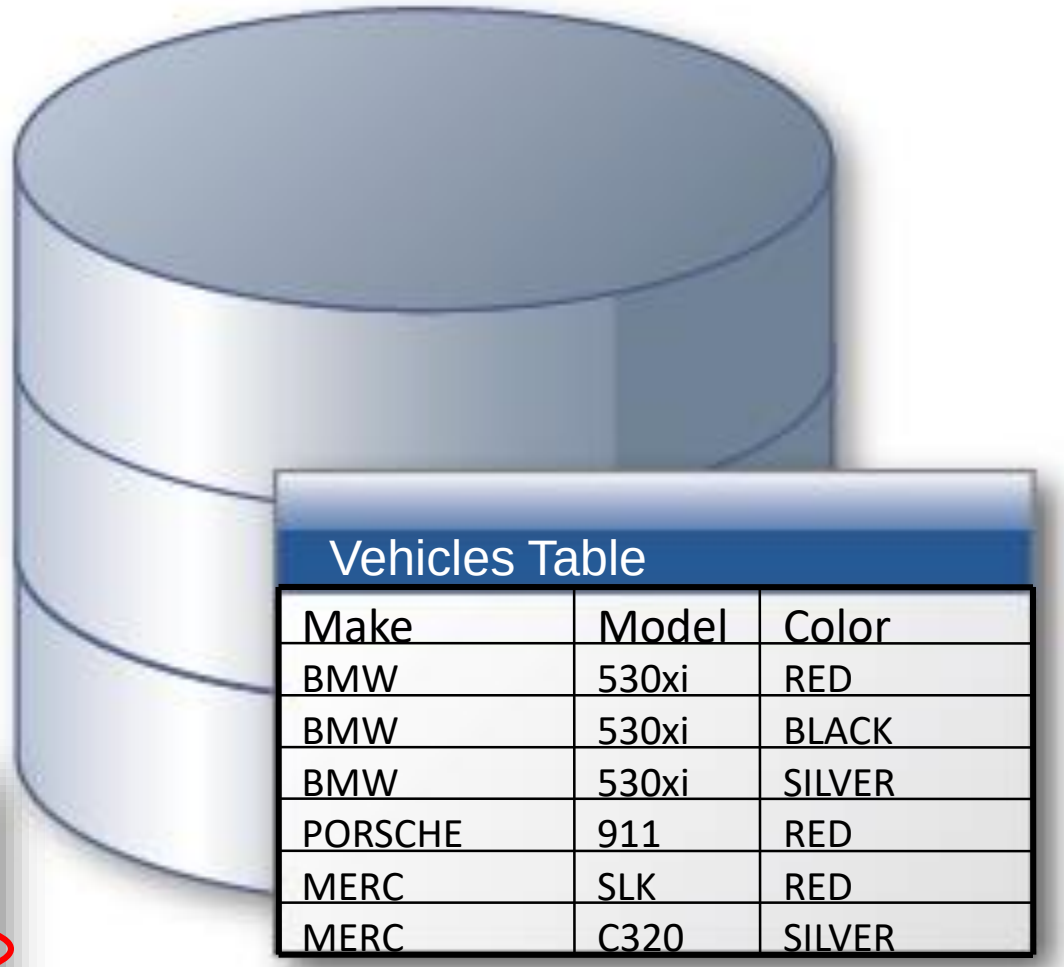
最適化の戦略
オプティマイザ統計（拡張統計）

```
SELECT * FROM vehicles
WHERE model = '530xi'
AND      make = 'BMW';
```

MAKE(c2)	MODEL(c1)	COLOR
BMW	530xi	RED
BMW	530xi	BLACK
BMW	530xi	SILVER

Cardinality= #ROWS * $\frac{1}{NDV(c1)}$ * $\frac{1}{NDV(c2)}$ => $12 * \frac{1}{4} * \frac{1}{3} = 1$

```
-------------------------------------------------------------------
| Id  | Operation         | Name     | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------
|   0 | SELECT STATEMENT  |          |      1 |        |      3 |
|*  1 |  TABLE ACCESS FULL| VEHICLES |      1 |      1 |      3 |
-------------------------------------------------------------------
```



最適化の戦略
オプティマイザ統計（拡張統計）

- 制限事項（列グループ統計）
  - 等価条件またはIN-LISTの場合のみに使用される
    - 単一表複数列のGROUP BYカーディナリティ見積りにも使用する
  - 基になる列にヒストグラムが存在し、列グループにヒストグラムが存在しない場合には使用されない
    - 問合せで使用されないとヒストグラムが作成されない（col_group_usage$に登録されないと）

最適化の戦略
オプティマイザ統計（初期統計収集の戦略）

1. テーブルの作成
2. 自動列グループ検出を使用する場合（ディレクティブから自動生成しない）
   - 空テーブルの統計を収集
   - dbms_stats.seed_col_usageを実行
3. 空テーブルに問合せを実行する
   - 作成する列グループやヒストグラムをオプティマイザに検出させる
4. 列グループの作成
   - dbms_stats.report_col_usage
   - dbms_stats.create_extended_stats
5. 増分統計を有効にする
   - 大きなパーティション表の場合
6. データのロード
7. 統計の収集
   - デフォルトを使用（ヒストグラムも作成）
8. 索引の作成（統計も収集される）
   - もし必要なら

最適化の戦略
動的統計（動的サンプリング）

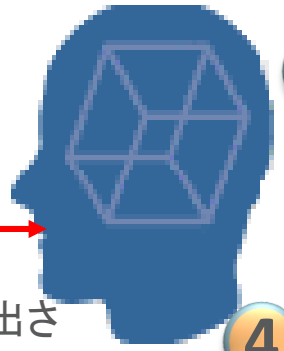
- 既存のオプティマイザ統計を補うために解析時に収集された統計
- 通常のオプティマイザ統計では正しいカーディナリティ見積りを得ることができない場合に使用する
  - 表レベルの統計は、結合カーディナリティとGroup-By（問合せブロック）カーディナリティを含むように12cで強化された
    - 問合せブロック・カーディナリティはビュー・マージなどの判断に使用
- OPTIMIZER_DYNAMIC_SAMPLINGパラメータにより制御
- 動的統計は統計の品質を大幅に向上させ、より良い実行計画につながる
- しかし、コストが増加する
  - より良い統計は通常より大きなサンプル・サイズで得られ、解析時間が長くなる

最適化の戦略
動的統計（動的サンプリング）

- 動的サンプリング（11gR2まで）
  - デフォルトでパラレル実行すると自動調整
    - レベル：3から10（WHERE句の複雑さと表のブロック数から）
- 動的統計/適応動的サンプリング（12cから）
  - optimizer_dynamic_samplingパラメータ/dynamic_samplingヒント=11
  - デフォルトでパラレル実行すると自動調整
    - optimizer_adaptive_statistics=FALSEのとき（11gR2と同様）
    - optimizer_adaptive_statistics=TRUEのとき（レベル：11）
  - SQL計画ディレクティブからキックされたとき（optimizer_adaptive_statistics=TRUE）
    - optimizer_dynamic_samplingパラメータ/dynamic_samplingヒント=0以外
  - 他のSQLと共有される

最適化の戦略
適応動的統計サンプリング

```
Select *
From product_information
Where list_price-min_price=29
And category_id not in (11,22)
And prod_name like 'Smart%';
```



1 SQL文が提出された

2 オプティマイザはデータ・ディクショナリ内の既存の統計をチェックする

3 統計は見つかったが、複雑な述語のために増強する必要がある

4 動的サンプリングは表からの少数のブロックで行われる
結果の動的統計はキャッシュに保管される

5 オプティマイザは動的統計を使用して実行計画を決定する

SGA

共有動的統計

prod_infoの表カーディナリティ見積り

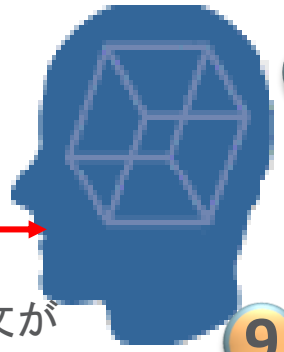
SHARED_POOL

Cursor 0: Select * from product_information …

DATA DICTIONARY

最適化の戦略
適応動的統計サンプリング

```
Select supplier_id, prod_name
From product_information
Where list_price-min_price=29
And category_id not in (11,22)
And prod_name like 'Smart%';
```



6 異なるSQL文が同じ述語を使用して実行依頼された

10 オプティマイザは動的統計を使用して実行計画を決定する

9 共有キャッシュに必要な動的統計が見つかった

7 オプティマイザはデータ・ディクショナリ内の既存の統計をチェックする

8 統計は見つかったが、複雑な述語のために増強する必要がある

SGA

共有動的統計

prod_infoの表カーディナリティ見積り

SHARED_POOL

Cursor 0: Select supplier_id, prod_name …

DATA DICTIONARY

最適化の戦略
適応動的サンプリング（Adaptive dynamic sampling）

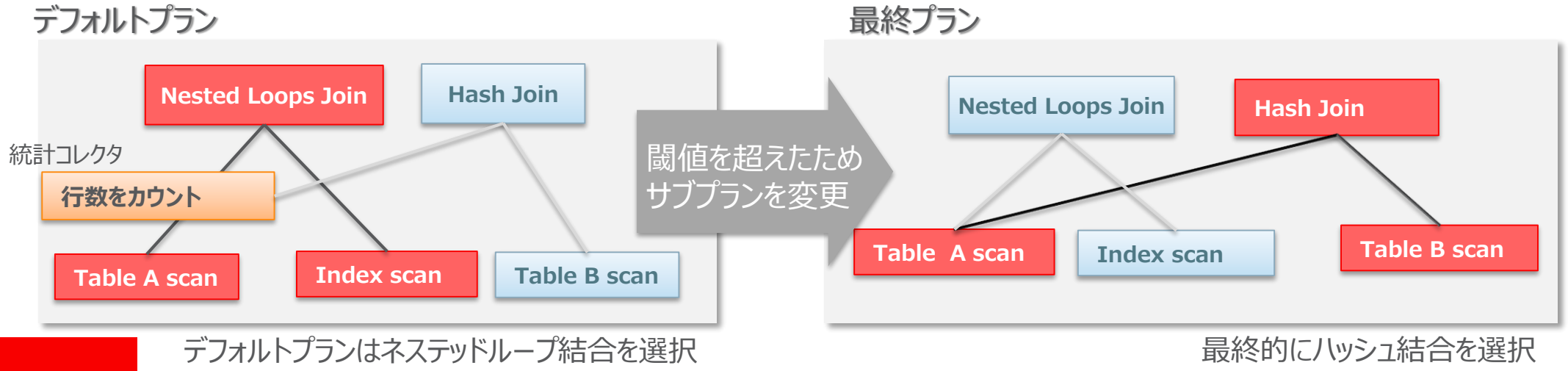
- 旧式の動的サンプリングでは、オプティマイザは単一のサンプリング問合せを実行する
  - サンプリング率は動的サンプリング・レベルに依存
- 適応動的サンプリングでは、オプティマイザは複数のサンプリング問合せを実行する
  - この数は、問合せの複雑さと利用可能なインデックスの数によって異なる
    - たとえ少ない表にアクセスする問合せであっても、1つの解析に対して10～30個のサンプリング問合せを行うこともある
  - この数値は、データベースのバージョンによっても異なる
    - 新しいリリースでは通常、より多くのサンプリング問合せが実行される
- そのため12cR2からデフォルトがFALSEになっている

最適化の戦略
マテリアライズド・ビュー（Mview）

- 問合せの結果を格納
  - 一般的で長時間実行される問合せ結果を素早く取得するために使用できる
  - 定期的に更新される
- オプティマイザが関連表のカーディナリティ見積りをより良くするための手法として使用できる
  - スノーフレーク・スキーマには、通常静的な関連ディメンションが含まれている
    - オプティマイザが正確な見積りを得るのは難しい（結合カーディナリティが存在しないため）
  - 関連表のMViewを使用すると、オプティマイザはMViewの統計を使用して、より優れたカーディナリティ見積りを取得できる

最適化の戦略
適応計画（適応結合方法、適応パラレル分散方法）

- 既存の統計情報から複数のサブプランを生成し、SQL実行中に収集した実行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機能（デフォルトがTRUE）
- 適応計画の動作イメージ
  1. 既存の統計情報を基に複数のサブプラン（実行計画）を作成
  2. 最適と思われる実行計画（デフォルトプラン)にて、SQL文を実行
  3. 統計コレクタ（最初の表）により収集された実行統計を基に、必要に応じて実行計画の一部を変更
     - 適応結合方法（NL結合orHASH結合）の例



最適化の戦略
適応パラレル分散方法（HYBRID HASH）

- 実行時の収集コレクタの行数によってHASHとBROADCASTを決定
  - 行数がパラレル度の2倍以上の場合はHASH、それ以外はBROADCAST
- 重複値も分散可能（**第39回**）
  - 結合列にヒストグラムが存在するかPQ_SKEWヒント
    - 「PX SEND HYBRID HASH (SKEW)」と出力される
  - 重複値をBROADCASTする

最適化の戦略
適応パラレル分散方法（HYBRID HASH）

```
SQL> SELECT COUNT(*) FROM (SELECT * FROM t3,t1 WHERE t3.col1 = t1.col1);
```

実行時に行数を求めてパラレル度によってBROADCAST / HASHを決定

重複データを分散したときに'(SKEW)'が出力される

BROADCASTだと行数×パラレル度

実行計画

Id	Operation	Name	Starts	E-Rows	TQ	IN-OUT	PQ Distrib	A-Rows
0	SELECT STATEMENT		1					1
1	SORT AGGREGATE		1	1				1
2	PX COORDINATOR		1					2
3	PX SEND QC (RANDOM)	:TQ10002	0	1	Q1,02	P->S	QC (RAND)	0
4	SORT AGGREGATE		2	1	Q1,02	PCWP		2
* 5	HASH JOIN		2	50000	Q1,02	PCWP		100K
6	PX RECEIVE		2	5	Q1,02	PCWP		5
7	PX SEND HYBRID HASH	:TQ10000	0	5	Q1,00	P->P	HYBRID HASH	0
8	STATISTICS COLLECTOR		2		Q1,00	PCWC		5
9	PX BLOCK ITERATOR		2	5	Q1,00	PCWC		5
* 10	TABLE ACCESS FULL	T1	1	5	Q1,00	PCWP		5
11	PX RECEIVE		2	100K	Q1,02	PCWP		100K
12	PX SEND HYBRID HASH (SKEW)	:TQ10001	0	100K	Q1,01	P->P	HYBRID HASH	0
13	PX BLOCK ITERATOR		2	100K	Q1,01	PCWC		100K
* 14	TABLE ACCESS FULL	T2	26	100K	Q1,01	PCWP		100K

最適化の戦略
統計フィードバック（カーディナリティ・フィードバック）

見積りと実行時の統計が大きく異なると記録して2回目から使用する

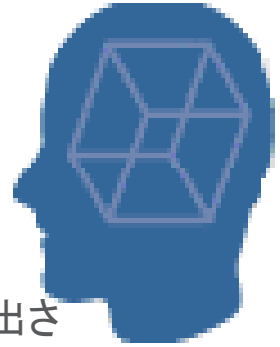
- カーディナリ・ティフィードバック（11gR2まで）
  - 単一表カーディナリティ
- 統計フィードバック（12cから）
  - optimizer_adaptive_statistics=FALSEのとき
    - 11gR2と同様
  - optimizer_adaptive_statistics=TRUEのとき
    - **結合カーディナリティ**、問合せブロック・カーディナリティも対応
- 適応統計（統計フィードバックや動的統計）は結合カーディナリティや問合せブロック・カーディナリティを補正できる唯一の機能

最適化の戦略
SQL計画ディレクティブ

- ディレクティブは、最適な計画を生成するために最適化中に使用される追加情報
  - 例えば、表T1がT2に結合されている場合は、動的統計を使用して正確なカーディナリティ見積りを取得する
  - ディレクティブはSQL文レベルではなく問合せの述語で収集される
    - 複数のSQL文にディレクティブを使用することを許可する
  - ディレクティブはSYSAUX表領域に格納され自動的に維持される
  - DBMS_SPDパッケージを使用して管理
    - EXEC dbms_spd.alter_sql_plan_directive(<ディレクティブID>,'ENABLED','NO');

最適化の戦略
統計フィードバックとSQL計画ディレクティブ

```
Select *
FROM customers
where state ='CA'
AND country='USA';
```



1 SQL文が提出された

2 オプティマイザが実行計画を決定する

3 実行中にカーディナリティ見積りが間違っていることがわかりディレクティブが作成された

SGA

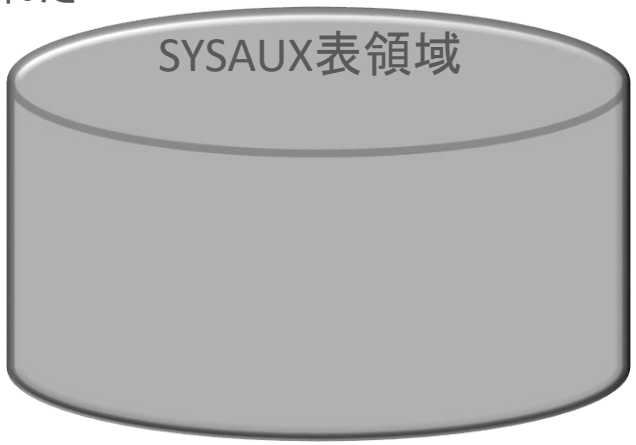
ディレクティブ・キャッシュ

Directive: 列state, countryが使用されている場合にcustomers表にDSを使用する

SHARED_POOL

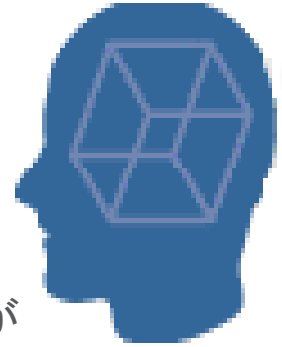
Cursor 0: Select * from customers ……..

IS_REOPTIMIZABLE = Y



最適化の戦略
統計フィードバックとSQL計画ディレクティブ

```
Select *
FROM customers
where state='CA'
AND country='USA';
```



4 同じSQL文が再度提出された

5 オプティマイザは実行統計を使用して新しい実行計画を決定する

6 次回からの実行では子カーソル1が使用される

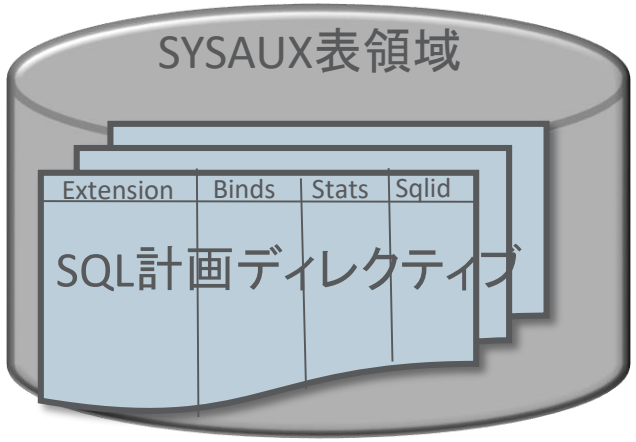
SGA

ディレクティブ・キャッシュ

SHARED_POOL

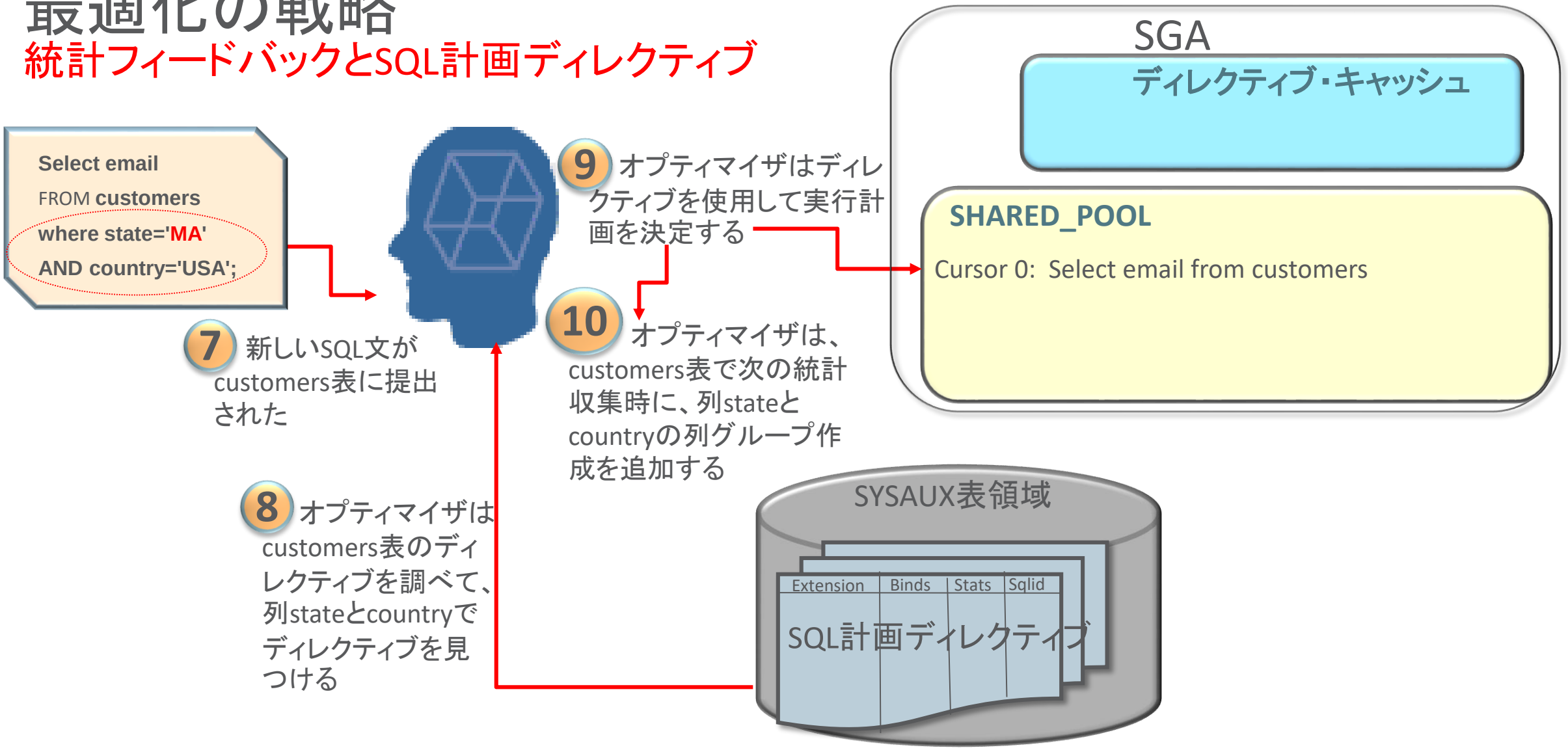
Cursor 0: Select * from customers .....
IS_REOPTIMIZABLE = Y

Cursor 1: Select * from customers .....
IS_REOPTIMIZABLE = N



ディレクティブはSYSAUX表領域内のディスクに定期的に格納する

最適化の戦略
統計フィードバックとSQL計画ディレクティブ



最適化の戦略
適応統計

- 12cR2の変更（第63回）
  - デフォルト値がFALSE
    - 適応計画（optimizer_adaptive_plans）と適応統計（optimizer_adaptive_statistics）に分離
  - SQL計画ディレクティブからの列グループ統計自動作成
    - プリファレンスAUTO_STAT_EXTENSIONSだけで制御（デフォルトはFALSE）
  - 小さい問合せでの自動再最適化
    - アクセス数が少ない問合せは補正しない（100未満のアクセス・ブロック数）
      - もともと遅くないので補正する必要がない（動的統計のオーバーヘッドが大きくなってしまう）
  - 動的統計の共有
    - SQL計画ディレクティブ・リポジトリ（12cは結果キャッシュ）

最適化の戦略
適応統計

- 18cの変更
  - パフォーマンス・フィードバック
    - 問合せ時間に基づいて選択した並列度の調整
      - 収集した統計（CPU Timeなど）を使用してパラレル度を改善
    - 12cR2までは適応統計の機能
    - 適応統計から分離
      - parallel_degree_policy=adaptiveだけで制御

最後に

- 次の項目がパフォーマンスに大きな影響を与えることを忘れないで、オプティマイザと上手くつきあうようにしてください
  - スキーマ設定
  - SQL設計
  - 実行戦略
  - 最適化テクニック

- よろしければ「津島博士のパフォーマンス講座」を読んでみてください

津島博士の質問コーナー（ask 津島博士）

津島博士の質問コーナー（ask 津島博士）

津島博士からの質問

- この問合せのアクセス方法は以下のどれが正しいでしょうか。列c1に索引があり、対象件数は表t1（100万件）に対して10件とする。

```
SELECT * FROM t1 WHERE substr(c1,1,10) = ’1234567890’
```

1. 列c1の索引だけで索引スキャンを行う
2. 仮想列があれば索引スキャンを行う
3. 索引スキャンするにはファンクション索引が必要
4. 何を行っても全表スキャンになる

津島博士の質問コーナー（ask 津島博士）
津島博士からの質問（答え）

- この問合せのアクセス方法は以下のどれが正しいでしょうか。列c1に索引があり、対象件数は表t1（100万件）に対して10件とする。

```
SELECT * FROM t1 WHERE substr(c1,1,10) = ’1234567890’
```

1. 列c1の索引だけで索引スキャンを行う
2. 仮想列があれば索引スキャンを行う
3. **索引スキャンするにはファンクション索引が必要**
4. 何を行っても全表スキャンになる

述語に関数がある時に索引は使用されません。ファンクション索引を作成すれば索引を使用することができます（仮想列はオプティマイザ統計を正確にしますが、索引は使用されません）。

津島博士の質問コーナー（ask 津島博士）

津島博士からの質問

- 以下の実行計画の結合時のデータ分散方法は最終的に何になると思いますか？

1. HASH-HASH
2. BROADCAST-NONE
3. PARTITION-NONE
4. NONE-NONE

```
SQL> SELECT /*+ parallel(8) */ COUNT(*) FROM (SELECT * FROM t3,t1 WHERE t3.col1 =
t1.col1);
-------------------------------------------------------------------------------------
| Id  | Operation                          | Name     | Starts | E-Rows |  | A-Rows |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |          |      1 |        |  |      1 |
|   1 |  SORT AGGREGATE                    |          |      1 |      1 |  |      1 |
|   2 |   PX COORDINATOR                   |          |      1 |        |  |      2 |
|   3 |    PX SEND QC (RANDOM)             | :TQ10002 |      0 |      1 |  |      0 |
|   4 |     SORT AGGREGATE                 |          |      8 |      1 |  |      2 |
|*  5 |      HASH JOIN                     |          |      8 |  50000 |  |   100K |
|   6 |       PX RECEIVE                   |          |      8 |      5 |  |      X |
|   7 |        PX SEND HYBRID HASH         | :TQ10000 |      0 |      5 |  |      0 |
|   8 |         STATISTICS COLLECTOR       |          |      8 |        |  |      5 |
|   9 |          PX BLOCK ITERATOR         |          |      8 |      5 |  |      5 |
|* 10 |           TABLE ACCESS FULL        | T3       |      1 |      5 |  |      5 |
|  11 |       PX RECEIVE                   |          |      8 |   100K |  |   100K |
|  12 |        PX SEND HYBRID HASH (SKEW)  | :TQ10001 |      0 |   100K |  |      0 |
|  13 |         PX BLOCK ITERATOR          |          |      8 |   100K |  |   100K |
|* 14 |          TABLE ACCESS FULL         | T1       |     26 |   100K |  |   100K |
```

津島博士の質問コーナー(ask 津島博士)
津島博士からの質問(答え)

- 以下の実行計画の結合時のデータ分散方法は最終的に何になると思いますか？

1. HASH-HASH
2. **BROADCAST-NONE**
3. PARTITION-NONE
4. NONE-NONE

行数＜パラレル度＊2のため（5＜8＊2）

```
SQL> SELECT /*+ parallel(8) */ COUNT(*) FROM (SELECT * FROM t3,t1 WHERE t3.col1 =
t1.col1);
-------------------------------------------------------------------------------------
| Id  | Operation                          | Name     | Starts | E-Rows |  | A-Rows |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |          |      1 |        |  |      1 |
|   1 |  SORT AGGREGATE                    |          |      1 |      1 |  |      1 |
|   2 |   PX COORDINATOR                   |          |      1 |        |  |      2 |
|   3 |    PX SEND QC (RANDOM)             | :TQ10002 |      0 |      1 |  |      0 |
|   4 |     SORT AGGREGATE                 |          |      8 |      1 |  |      2 |
|*  5 |      HASH JOIN                     |          |      8 |  50000 |  |   100K|
|   6 |       PX RECEIVE                   |          |      8 |      5 |  |     40 |
|   7 |        PX SEND HYBRID HASH         | :TQ10000 |      0 |      5 |  |      0 |
|   8 |         STATISTICS COLLECTOR       |          |      8 |        |  |      5 |
|   9 |          PX BLOCK ITERATOR         |          |      8 |      5 |  |      5 |
|* 10 |           TABLE ACCESS FULL        | T3       |      1 |      5 |  |      5 |
|  11 |       PX RECEIVE                   |          |      8 |   100K|  |   100K|
|  12 |        PX SEND HYBRID HASH (SKEW)  | :TQ10001 |      0 |   100K|  |      0 |
|  13 |         PX BLOCK ITERATOR          |          |      8 |   100K|  |   100K|
|* 14 |          TABLE ACCESS FULL         | T1       |     26 |   100K|  |   100K|
```

津島博士の質問コーナー（ask 津島博士）
津島博士からの質問

- 12cR2から適応統計（optimizer_adaptive_statistics）のデフォルトはFALSEになりましたが、皆さんはこの機能を今後使用しますか。

1. デフォルトのまま（使わない）
2. 有効なときは使用する
3. どちらとも言えない

津島博士の質問コーナー（ask 津島博士）
津島博士からの質問（コメント）

- 12cR2から適応統計（optimizer_adaptive_statistics）のデフォルトはFALSEになりましたが、皆さんはこの機能を今後使用しますか。

1. デフォルトのまま（使わない）
2. 有効なときは使用する
3. どちらとも言えない

以下のようなメリットがあることを忘れないで下さい

- 有効なところ
  - 長時間実行の問合せ
  - 複雑な問合せ
  - 複雑なスキーマ
  - 複雑なデータ
- Ad-hoc問合せ環境では
  - SQL計画ディレクティブにより、オプティマイザはSQLから学習し、その情報を他と共有することができる

テック・ナイトアーカイブ資料と お役立ち情報

各回テック・ナイトセッション資料 ダウンロードサイト

oracle technight

技術コラム しばちょう先生の 試して納得！ DBAへの道

技術コラム 津島博士の パフォーマンス講座

もしも みなみんが DBをクラウドで 動かしてみたら

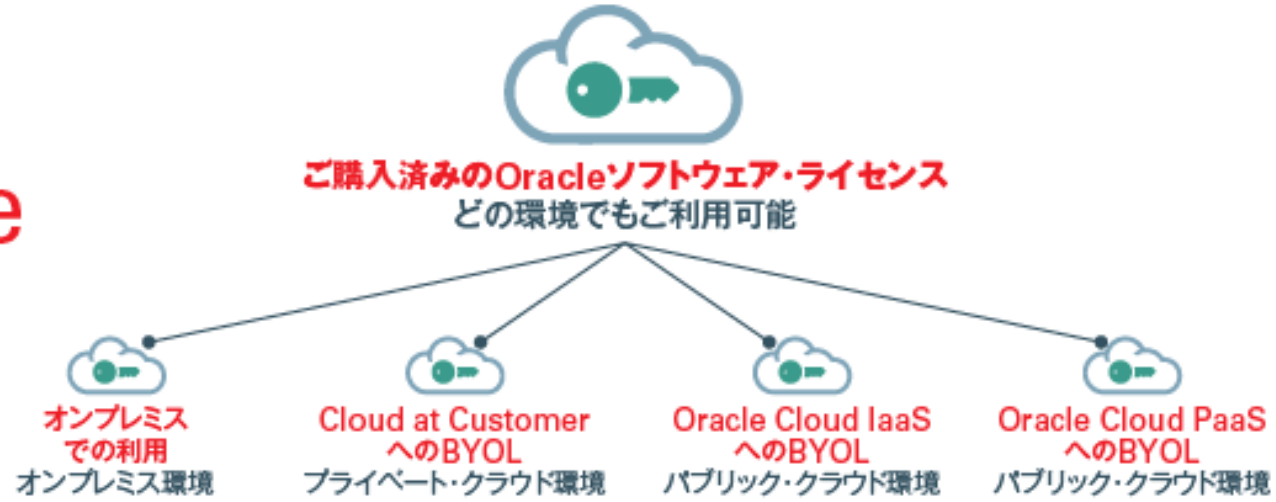
～ みなさまの投稿をお待ちしております ～

Twitter

#OracleTechNight

Bring Your Own License

既存のオラクル・ライセンスを
柔軟にクラウド環境で活用



300ドル分の無料トライアルでOracle Cloudを体験!

https://cloud.oracle.com/ja_JP/tryit

Oracle Cloudでは各種クラウドサービスを300ドル分無料でお試しいただけるトライアルサービスをご提供しております。無料トライアルのお申込み方法の詳細は、左のQRコード、またはURLにアクセスしてください。

Oracle Cloudのユースケース、導入事例、資料、価格などの詳細情報は、下記URLにアクセスしてください。
http://www.oracle.com/jp/cloud/platform/overview/index.html

ORACLE

こんな時、かけこむ会社が増えています。

Oracle Digitalは、オラクル製品の導入をご検討いただく際の総合窓口。
電話とインターネットによるダイレクトなコニュニケーションで、どんなお問い合わせにもすばやく対応します。
もちろん、無償。どんなことでも、ご相談ください。

Oracle Digital

お問い合わせは電話またはWebフォーム

☎ 0120-155-096
受付時間　月～金　9:00-12:00 / 13:00-17:00
（祝日および年末年始休業日を除きます）

http://www.oracle.com/jp/contact-us

ORACLE

Integrated Cloud

Applications & Platform Services


ORACLE®