第22回 夜もよか！！オラクル勉強会

ORACLE

性能トラブル対策ガイド!
オプティマイザ/統計情報収集について



日本オラクル株式会社
九州支社
2014年10月15日

ORACLE DATABASE

よかたい オラクル倶楽部

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。

OracleとJavaは、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。
文中の社名、商品名等は各社の商標または登録商標である場合があります。

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

目的とゴール

目的

オプティマイザ統計情報の特徴を理解し運用方法を身につける

ゴール

- 統計情報収集の各種パラメータを理解する
- システムにあった選択ができるようになる
- 課題発生時の改善ができるようになる

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

Program Agenda

- システム品質の重要性
- オプティマイザ統計情報(Optimizer Statistics)
- 統計収集の効率向上
- その他の関連機能
- 課題のパターン別 統計情報収集のポイント

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

システム品質の重要性



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

システム品質

低い品質によって発生する問題

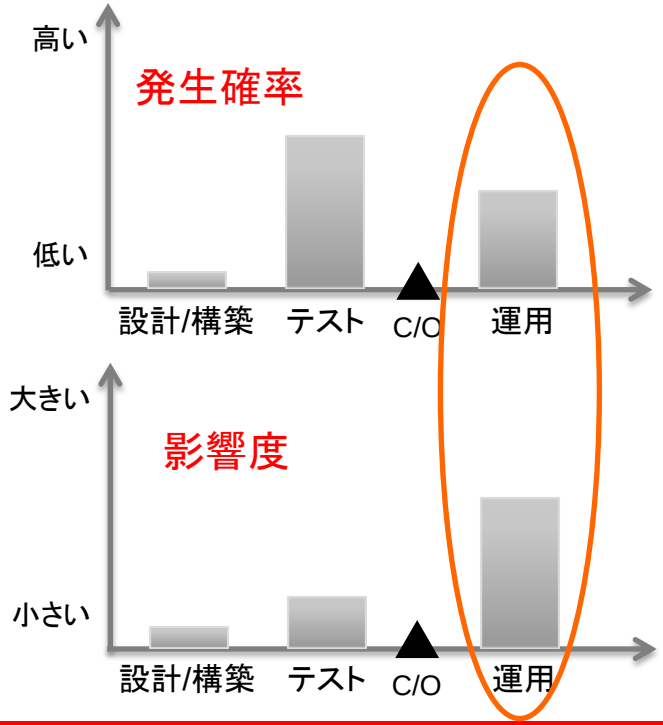
- 経営観点から見たリスク
  - 修正・手戻りが増えることによる開発コストの増大
  - 自社・お客様の本業務への悪影響が発生
  - 社会的名声へのダメージ、行政処分/注意

- 開発現場から見たリスク
  - 開発スケジュールが混乱し、修正作業に足が引っ張られる
  - 要件定義から見直しが必要な場合も

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

問題の発生確率と影響度の関係

運用開始後の性能問題が多い傾向





開発コストの約80%は
欠陥の発見と修正に
費やされている

* Source: IBM



リリース後に発見された
不具合は、設計段階より
100倍以上の修正コスト
を要する

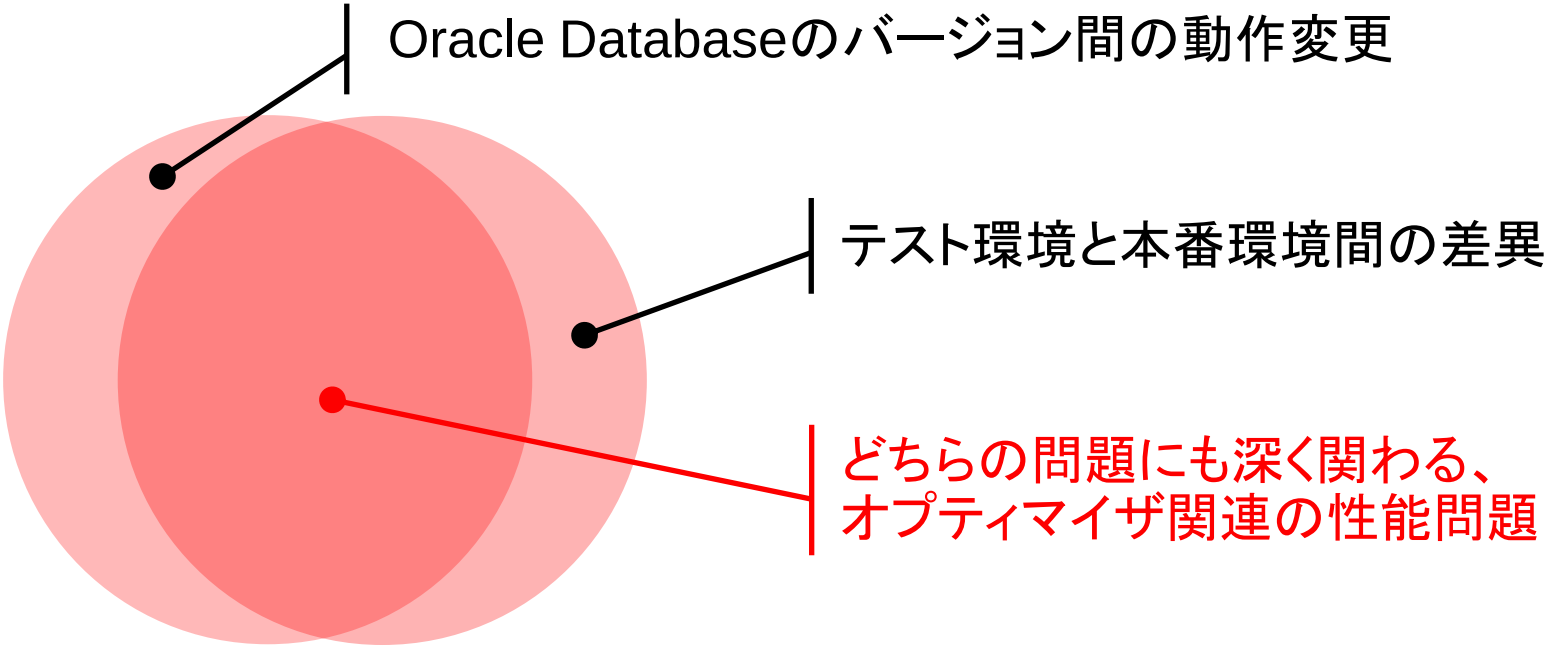
* Source: NIST

運用開始後の業務への影響を
未然に防ぐための設計/構築とテストが重要

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Databaseにおける性能問題

主な原因は？



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ関連の性能問題
実際に発生したトラブル・ケース

- Oracle 9i Databaseで使用していた初期化パラメータの値を Oracle Database 11gでも流用した結果、SQL性能が劣化

- Oracle Database 11gではRule-based Optimizer（RBO）ではなく Cost-based Optimizer（CBO）がデフォルトである情報を知らず、 統計情報収集の不備により不適切な実行計画を生成

- Oracle Database 11g の新機能を全く活用できておらず性能向上が 実現しない

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ関連の性能問題

運用フェーズでの発生を未然に防止する為には？

- 新環境での正しい知識をもとに適切な設計/設定を行う
  - 新しいバージョンにおける推奨設定や新機能を上手に活用する
- 本番環境と同じリアルなデータとSQLでテストを行う
  - データ量が異なれば、SQLの実行計画は異なる



Cost-based Optimizerの基本を理解し、
最新の正しいオプティマイザ統計情報の収集を行う



性能テストの観点を（少し）習得する

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

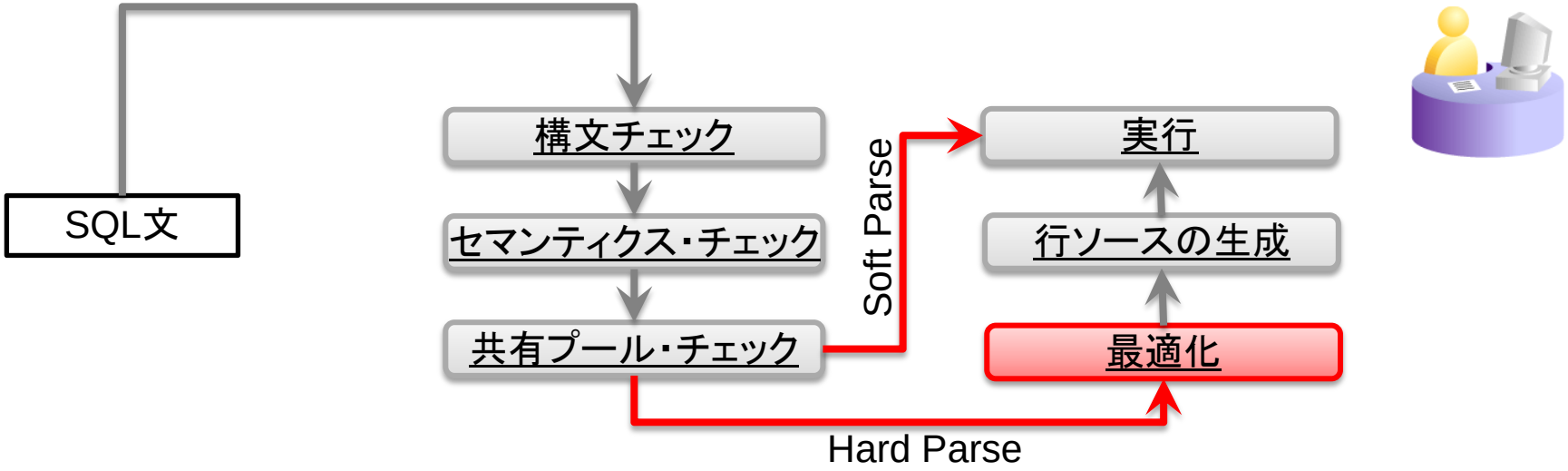
オプティマイザ統計情報
(Optimizer Statistics)



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

SQL実行の内部動作
処理ステップ

- クライアントから発行されたSQL文は、
  通常、以下の処理ステップで実行される



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

SQL文の処理ステップ
解析フェーズ（1）

- 構文チェック
  - SQL文の構文上の妥当性を確認
    - 例) SELECT * FORM employee;

- セマンティクス・チェック
  - SQL文内のオブジェクト及び列が存在するか等の文の意味の有効性を確認
    - 例) SELECT * FROM employee;

      ORA-00942: table or view does not exist

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

SQL文の処理ステップ
解析フェーズ（2）

- 共有プール・チェック
  - 共有プール（ライブラリ・キャッシュ）に同一SQL文の実行計画が存在するか否かを確認
    - 有 → Soft Parse
    - 無 → Hard Parse

ライブラリ・キャッシュではLRUアルゴリズムにより、
使用頻度が低いSQL文の実行計画が自動的に削除される
削除されたSQL文を再実行する場合には再度Hard Parseを行う必要がある為、
頻繁に使用されるSQL文の共有カーソルを全てキャッシュ可能な共有プールのサイズが理想

→ 自動メモリ管理の利用やメモリー・アドバイザを使用した最適なサイズ設定で解決可能

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

SQL文の処理ステップ
最適化フェーズ

- 問い合わせ処理の最適化
  - 解析済のSQL文を受け取り、最も効率的な実行計画を生成
  - 必要に応じてSQL文自体を変換
  - 各表へのアクセス方法や表同士の結合方法や結合順序を決定

- オプティマイザの種類
  - Rule-Based Optimizer（RBO）
  - Cost-Based Optimizer（CBO）

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザの進化

Rule-Based Optimizer → Cost-Based Optimizer

	RBO (Rule-based Optimizer)	CBO (Cost-based Optimizer)
概要	使用可能なアクセスパスを順序づけるランキングに基づいて実行計画を作成（OLTP向き）	統計情報に基づきコストを見積もり、最もコストの低い実行計画を作成（OLTP、DSS共に有効）
メリット	• アプリケーション開発者にとってRBOの考え方は理解しやすい • SQL実行計画の変動がほとんど起こらない	• データの変動に追随できる • 機能強化の恩恵を受けられる • データ偏りや量に基づいて実行計画を作成できる
デメリット	• データの変動に追随できない • データの偏りや量は考慮されない • 機能強化の恩恵を全く受けられない • Oracle Database 10g~ 未サポート	• 統計情報の取得が必要です • 統計情報の再収集によって性能が変化するリスクがある

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

Cost-based Optimizer

統計情報からコストを見積もり、最適な実行計画を生成

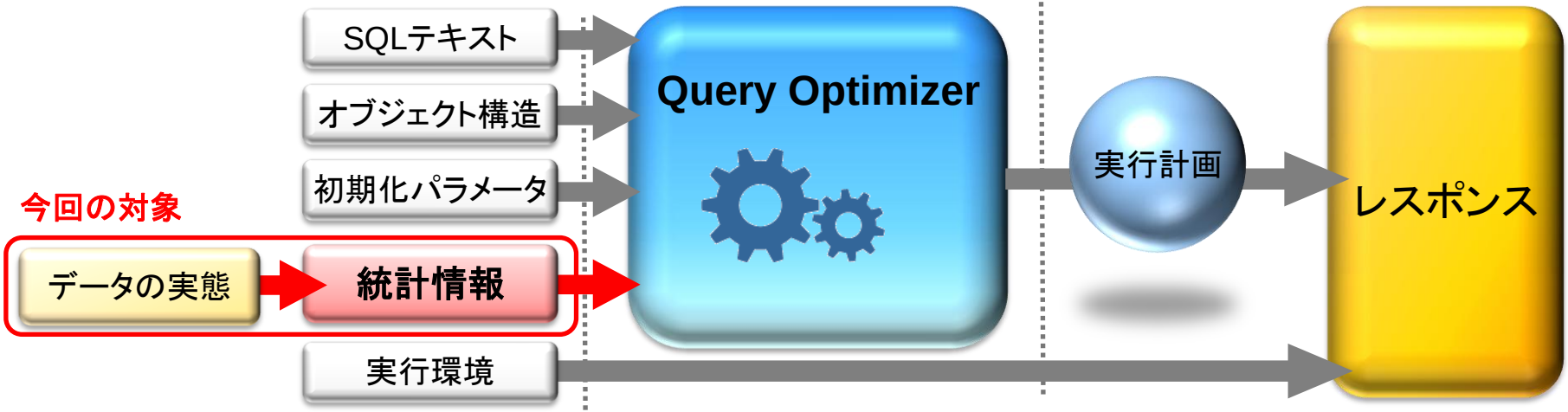
- 必要に応じてSQL文自体を変換
- 統計情報を基に各表へのアクセスコストを計算
  - 全表スキャンや索引スキャンなど各アクセスパスのコストと予想される行数を算出
- 結合順序と各結合方法のコストを計算
  - 各パターンの結合順序で全ての結合操作のコストを算出
- コスト評価を行い、最適なアクセスパスを決定
  - 最も低コストのアクセスパスを決定

ORACLE

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計
Optimizer Statistics

- オプティマイザ統計とは、データベースおよびデータベース内のオブジェクト（表や列、索引等）を特徴づける情報であり、Query オプティマイザは、これらの統計を使用して各SQL文に最適な実行計画を選択する。 ここで統計情報以外のインプットは固定的である事が多いため、**統計情報の変動がSQLのレスポンスに及ぼす影響は大きい**と言えます。

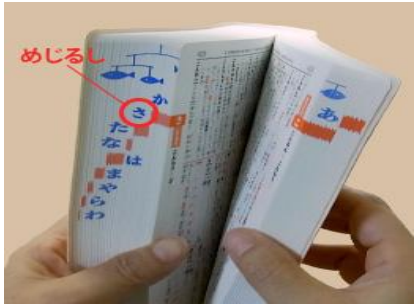


Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

全表スキャンや索引スキャン

- 検索における行の取り出し方は索引スキャンと全表スキャンがある

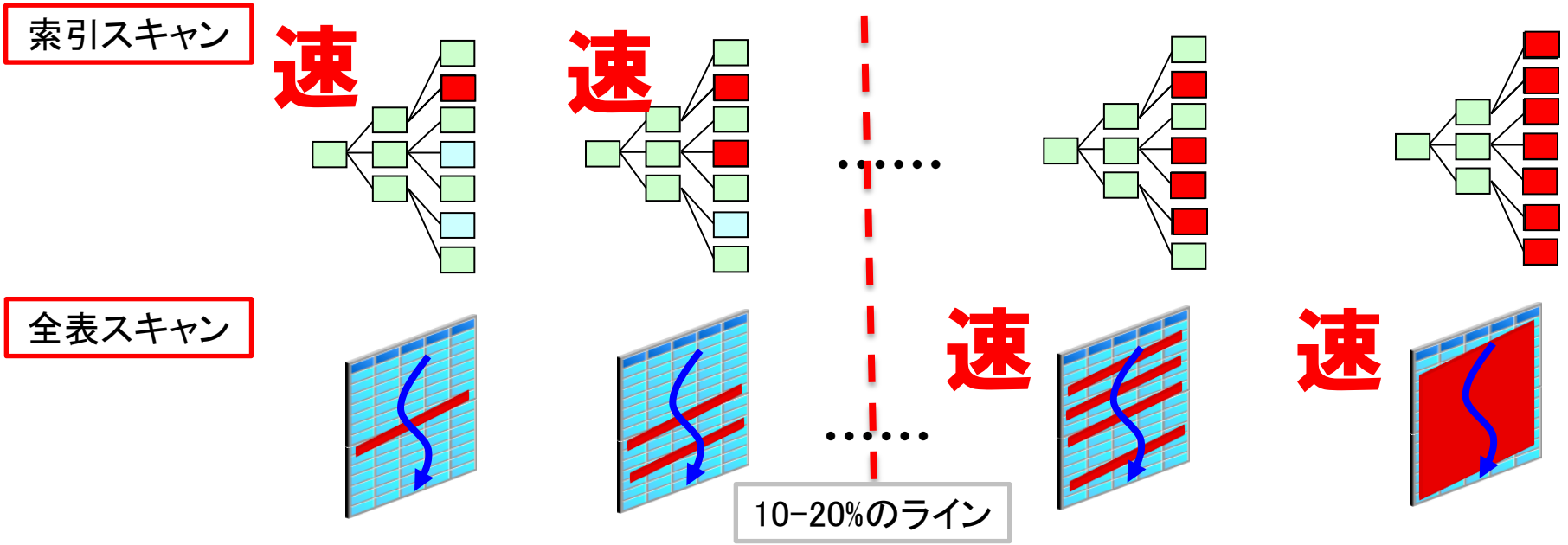
- 索引スキャン
  - 辞書から単語を探すとき、横の目印から探していくような方法
  - 特定の単語をピンポイントで探したいときに有効
- 全表スキャン
  - 該当する行を表の最初から最後まで検索する方法
  - 辞書の例でいえば、言葉の最後が”あ”で終わる用語を探したいなどヒット数が多数になるような場合に有効



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

アクセスする行数に応じて性能の違い

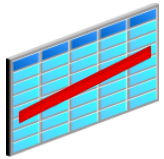
- 一般的に表の10-20%以上の行数にアクセスするならば全表スキャンの方が高速といわれる



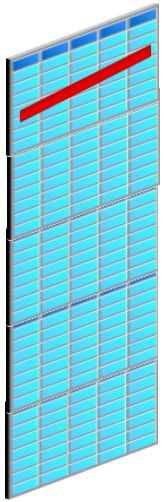
ORACLE
Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

アクセスする行数に応じて実行計画が変わる効果

- アクセスする行数が表の何%であるかは対象となる行数だけでなく、表全体の行数によっても変化



100行の表の中の25行
なら全体の25%なので
全表スキャンが有効

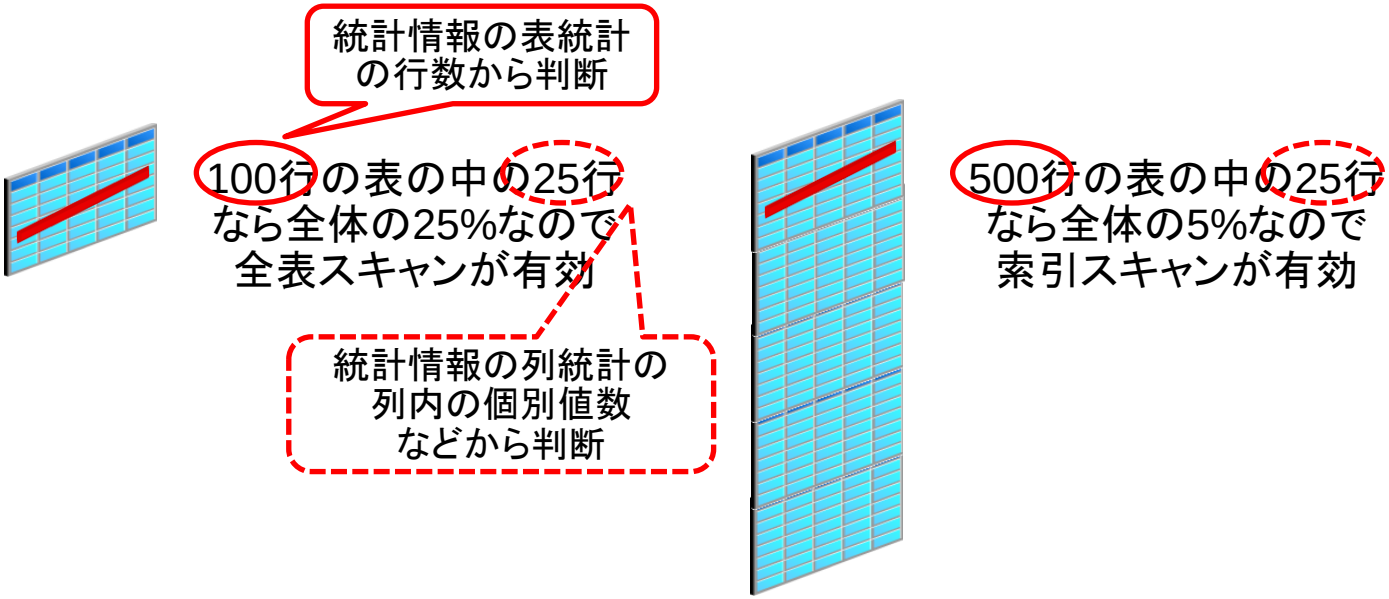


500行の表の中の25行
なら全体の5%なので
索引スキャンが有効

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

アクセスする行数に応じて実行計画が変わる効果

- 以下の2種の赤丸の情報は統計情報から得ることができる
- 統計情報に実データの情報を正確に反映させることが非常に重要



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計
表統計と列統計



- 表統計：USER_TAB_STATISTICS
  - 行数、データ・ブロック数、平均行長

- 列統計：USER_TAB_COL_STATISTICS
  - 列内の個別値（NDV：Number of Distinct Values:個別値）数
  - 列内のNULL数
  - 最小値/最大値
  - データ配分（ヒストグラム）
  - 拡張統計



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計
表統計と列統計のサンプル

```
SQL> exec print_table('select * from USER_TAB_STATISTICS');

TABLE_NAME                     : TAB_WITH_100_ROWS
PARTITION_NAME                 :
PARTITION_POSITION             :
SUBPARTITION_NAME              :
SUBPARTITION_POSITION          :
OBJECT_TYPE                    : TABLE
NUM_ROWS                       : 100
BLOCKS                         : 5
EMPTY_BLOCKS                   : 0
AVG_SPACE                      : 0
CHAIN_CNT                      : 0
AVG_ROW_LEN                    : 3
AVG_SPACE_FREELIST_BLOCKS      : 0
NUM_FREELIST_BLOCKS            : 0
AVG_CACHED_BLOCKS              :
AVG_CACHE_HIT_RATIO            :
SAMPLE_SIZE                    : 100
LAST_ANALYZED                  : 14-3月 -2014 15:02:10
GLOBAL_STATS                   : YES
USER_STATS                     : NO
STATTYPE_LOCKED                :
STALE_STATS                    : NO
```

```
SQL> exec print_table('select * from USER_TAB_COL_STATISTICS');

TABLE_NAME                     : TAB_WITH_100_ROWS
COLUMN_NAME                    : COL_NDV_10
NUM_DISTINCT                   : 9
LOW_VALUE                      : C102
HIGH_VALUE                     : C10A
DENSITY                        : .111111111111111
NUM_NULLS                      : 10
NUM_BUCKETS                    : 1
LAST_ANALYZED                  : 14-3月 -2014 15:02:10
SAMPLE_SIZE                    : 90
GLOBAL_STATS                   : YES
USER_STATS                     : NO
AVG_COL_LEN                    : 3
HISTOGRAM                      : NONE
```

+ “print_table” procedure

http://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11_QUESTION_ID:1035431863958

http://asktom.oracle.com/pls/apex/f?p=100:11:0::::P11_QUESTION_ID:1035431863958

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計
表統計と列統計からアクセス対象の行数を見積もり

- 例：100レコードの表で、NDV=10の列に対する等価条件のSELECT
  - オプティマイザは、”均等なデータ分布と仮定”し、次のようにCardinalityを計算

```
SQL> create table TAB_WITH_100_ROWS (COL_NDV_10 number);
SQL> insert into TAB_WITH_100_ROWS select case mod(LEVEL,10) when 0 then NULL else mod(LEVEL,10) end from DUAL connect by LEVEL <= 100;
SQL> commit;
SQL> exec dbms_stats.gather_table_stats('SIBA','TAB_WITH_100_ROWS');

SQL> select count(*) from TAB_WITH_100_ROWS where COL_NDV_10=2;
SQL> select * from table(dbms_xplan.display_cursor());
------------------------------------------------------------------------------------------
| Id  | Operation          | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |                   |       |       |     3 (100)|          |
|   1 |  SORT AGGREGATE    |                   |     1 |     3 |            |          |
|*  2 |   TABLE ACCESS FULL| TAB_WITH_100_ROWS |    10 |    30 |     3   (0)| 00:00:01 |
------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - filter("COL_NDV_10"=2)
```

［表の行数］/［NDV］
= 100 / 10 = 10

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計
追加の列統計

- 基本的な列統計では、列データの偏りや同一表内の列間の相互関係が存在する場合、最適な実行計画を選択できない
  下記の機能で統計を拡張することが可能

  - ヒストグラム
    - 頻度ヒストグラム（Frequency Histograms）
    - 高さ調整済ヒストグラム（Height balanced Histograms）

  - 拡張統計：Oracle Database 11g Release 1 ~　　⇒　Appendixに記載
    - 複数列の統計（Column Groups）
    - 式の統計（Expression Statistics）

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

ヒストグラム

- ヒストグラムとは？

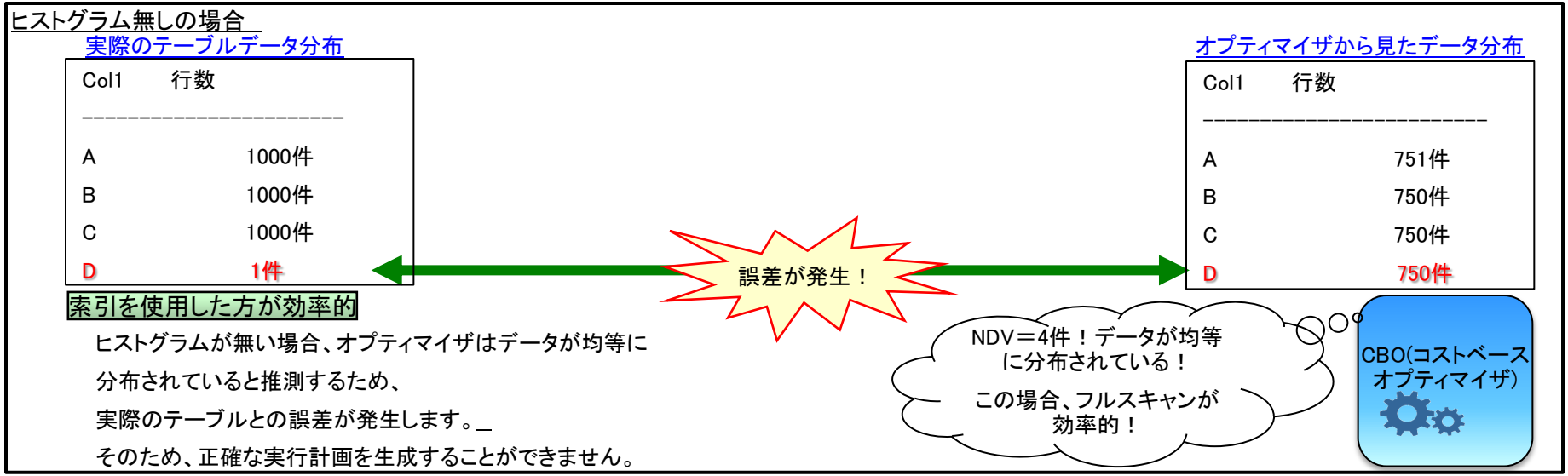
ヒストグラムとは、列データの分布です。

列データの分布の正確な見積もりを提供するため、

データが偏っている場合の選択性の見積もりがこれにより改善されます。

その結果最適な実行計画を生成することができます。

なお、Oracle Databaseには2種類のヒストグラムが存在します。



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

ヒストグラム

- ヒストグラムは列データの分布の状態をオプティマイザに対して提供
  - ヒストグラムが無ければ、オプティマイザはデータに偏りの存在を知ることができない
    - データが均等に分布していると推測する為、実データとの乖離が生まれ、
      正確な実行計画を選択できない

- 作成方法（DBMS_STATS.GATHER_XX_STATS）
  - 手動選定
    - 全列：　METHOD_OPT= 'FOR ALL COLUMNS SIZE [1~254]'
    - 特定列：　METHOD_OPT= 'FOR COLUMNS [列名] SIZE [1~254]'
  - 自動選定
    - METHOD_OPT= 'FOR ALL COLUMNS SIZE AUTO'
      - Oracle Databaseは列のデータ分布とワークロードに基づいて、
        ヒストグラムを収集する列を自動的に決定

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

頻度分布ヒストグラム

頻度分布ヒストグラムとは？

頻度分布ヒストグラムとは、列値の差分からデータ分布を正確に計算し記録するものです。

USER_HISTOGRAMディクショナリー・ビューから参照できます。

COL1(ENDPOINT_VALUE)	ENDPOINT_NUMBER
A	1000
B	2000
C	3000
D	3001

“A”が1000行

“B”が1000行

“C”が1000行

“D”が1行

行数累計の差分算出し、その差分がその列値の行数になる。

・ENDPOINT_NUMBERは行数の累計を表します。

・ENDPOINT_VALUEは列値を表します。

頻度ヒストグラムのメリット

・列の種類（NDV）が少ないが、他の値よりも極端に少ない値を検索する場合は索引アクセスが選択される可能性が高くなる

・列の値の差分を算出するため、正確なデータ分布が得られる

頻度ヒストグラムのデメリット

・データ量が多くなる

・最大255種類の値までしかヒストグラムの作成ができない

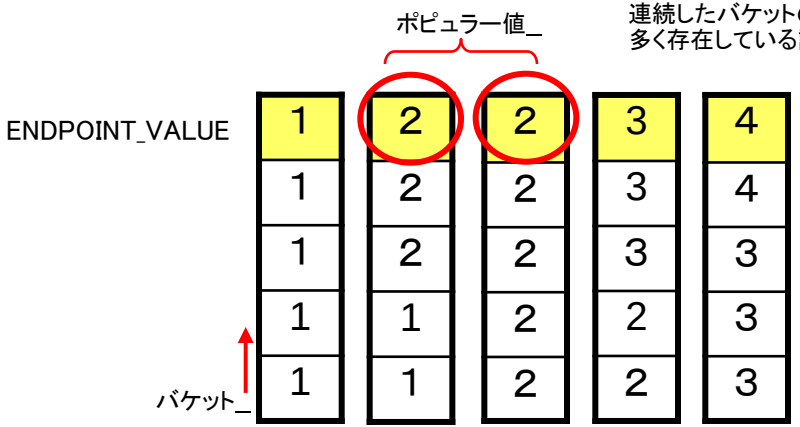
Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

高さ調整ヒストグラム

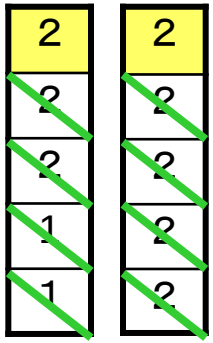
高さ調整ヒストグラムとは？

指定されたバケットと呼ばれるものに対し、ソートしたデータを順番に入れ、それぞれのバケットの最後の値（ENDPOINT_VALUE）を記録していき、その値によって列値の偏りを検出します。

USER_HISTOGRAMSディクショナリー・ビューから参照できます。



連続したバケットのENDPOINT_VALUEが同じであれば、その値が他の値より多く存在している証拠になり、そこからデータの偏りを算出します。



領域使用効率向上のため、ポピュラー値が存在した場合、最後のENDPOINT_VALUEを格納し、残りのバケット内の情報を省略します。

ポピュラー値とは･･･

複数のバケットのENDPOINT_VALUEとなる値。

ある値が他の値に比べて偏って多く存在する場合に発生する。

高さ調整ヒストグラムのメリット

･厳密な統計がなくても正しい実行計画の選択ができる

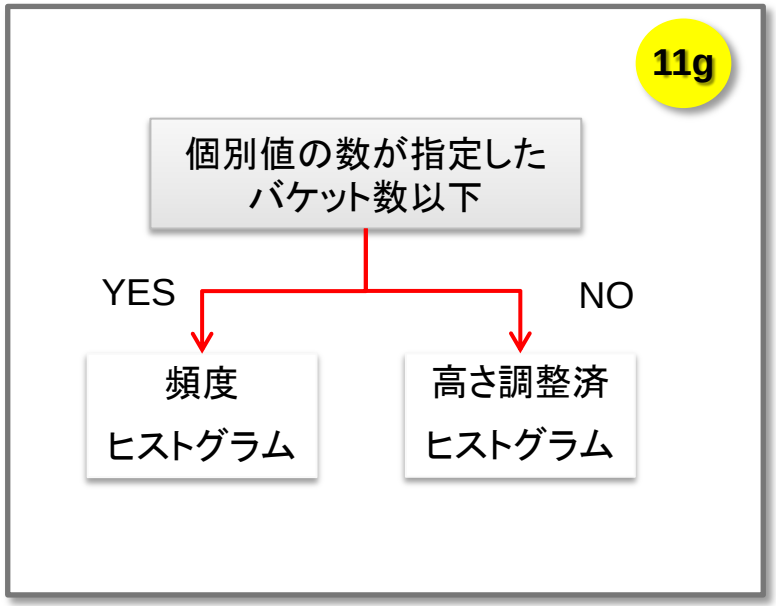
･列の個別値数が多い場合でも使用可能

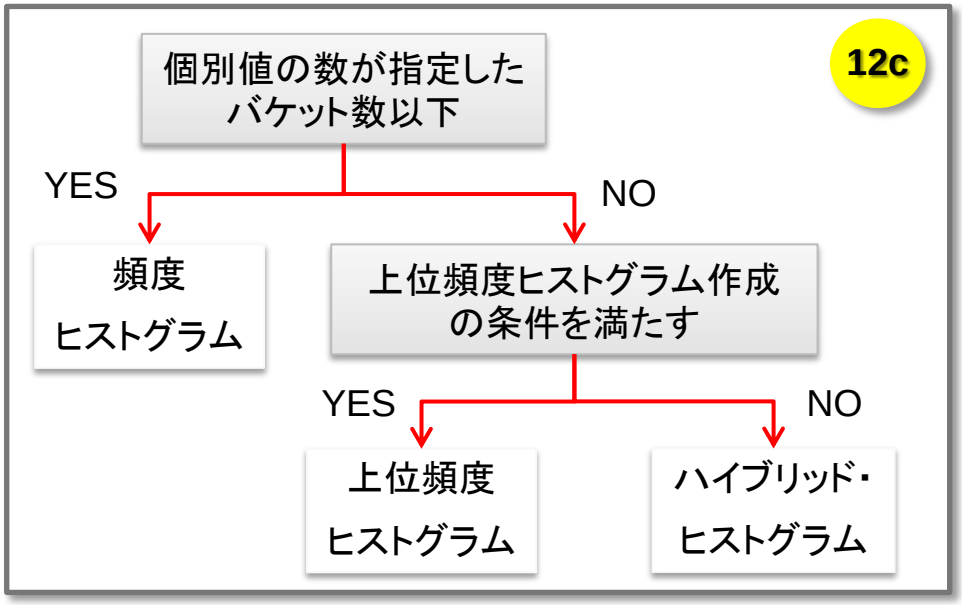
高さ調整ヒストグラムのデメリット

･ポピュラー値で偏りを図るため、等価条件ではポピュラー値以外は偏りをみつけられない

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

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ヒストグラムを作成する

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計
その他の統計

- 索引統計
- システム統計
- ディクショナリ表の統計
- 固定オブジェクトの統計

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の詳細(索引統計)

- 索引統計

  各索引ごとに収集された統計情報です。主に以下の項目が存在します。

  - リーフ・ブロック数

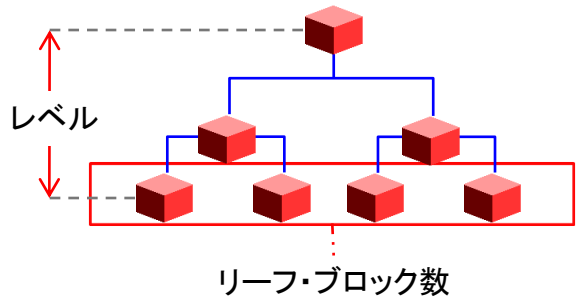
    索引に含まれるリーフ・ブロックの数を表します。

  - レベル

    ルート・ブロックからリーフ・ブロックまでの
    階層数を表します。

  - クラスタ化係数

    索引が付けられている列のデータが表内でどの程度分布しているかを表します。



補足)クラスタ化係数

クラスタ化係数は索引内の隣り合ったレコードが異なるデータ・ブロックへのポインターを持つ場合にカウント・アップされます。**この値が大きいほど、索引が付けられた列のデータが表全体に分布している事を意味します。**

取得する行数が少ない場合でもクラスタ化係数が大きい場合は、表内の大半のデータ・ブロックにアクセスする必要があるため、INDEX SCANよりFULL SCANの方が効率的です。

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計
索引統計

- 索引内の個別値（NDV）数
- 索引の深さ（BLEVEL）
  - ルートからリーフ・ブロックまでの階層数
- リーフ・ブロック数
- クラスタ化係数
  - 索引内の隣り合ったレコードが異なるデータ・ブロックへのポインタを持つ総数
  - 列データが表全体に分散している場合、大きな値を示す傾向

```
SQL> exec print_table('select * from USER_IND_STATISTICS');

INDEX_NAME                    : PRODUCT_NAME
TABLE_OWNER                   : JPETSTORE
TABLE_NAME                    : PRODUCT
PARTITION_NAME                :
PARTITION_POSITION            :
SUBPARTITION_NAME             :
SUBPARTITION_POSITION         :
OBJECT_TYPE                   : INDEX
BLEVEL                        : 2
LEAF_BLOCKS                   : 1954
DISTINCT_KEYS                 : 500000
AVG_LEAF_BLOCKS_PER_KEY       : 1
AVG_DATA_BLOCKS_PER_KEY       : 1
CLUSTERING_FACTOR             : 12500
NUM_ROWS                      : 500000
AVG_CACHED_BLOCKS             :
AVG_CACHE_HIT_RATIO           :
SAMPLE_SIZE                   : 500000
LAST_ANALYZED                 : 19-3月 -2014 16:32:14
GLOBAL_STATS                  : NO
USER_STATS                    : NO
STATTYPE_LOCKED               :
STALE_STATS                   : NO
```

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計
システム統計

- I/OとCPUのパフォーマンスおよび使用率などの情報
  - 各問合せに必要なI/O、CPUリソースを計算する際に使用
  - DBMS_STATS.GATHER_SYSTEM_STATSで取得可能（デフォルト値がある）
    - その際、任意の期間のアクティビティを分析するか、作業負荷をシミュレート
    - 基本的には一度取得するだけで良い(ワークロードが流れている時など)
  - 自動統計取得ジョブのデフォルト設定では自動取得されない
  - 取得した値は、DBMS_STATS.GET_SYSTEM_STATSプロシージャで参照

【参考】 Oracle Databaseパフォーマンス・チューニング・ガイド11gリリース2（11.2） 13.4 システム統計

10g以降では、CPU速度など、データベース起動時に自動的に取得される「NOWORKLOAD統計」と呼ばれるものと、明示的に設定しなければ取得されない「WORKLOAD統計」と呼ばれる、より詳細な統計情報の二種類があります。
WORKLOAD統計に関しては、システムの負荷特性を考慮しワークロードが流れている時の負荷がかかっている時が好ましい

ORACLE
Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計
ディクショナリ表の統計

- SYSTEMやSYSAUX表領域に格納されているディクショナリのスキーマ（SYS、SYSTEM及びRDBMSコンポーネントのスキーマ）の統計

  - 対象スキーマ名は、DBA_REGISTRYビューを参照（Doc ID 276358.1）
  - デフォルトでは、自動統計収集ジョブにより更新
    - AUTOSTATS_TARGETパラメータの値を「ORACLE」に変更することで、この統計のみの取得に限定することが可能（デフォルトは「AUTO」）
      - その場合、DBMS_STATS.GATHER_DICTIONARY_STATSを使用して、通常スキーマと同じく定期的にメンテナンスを推奨

    ⇒ 定期的にディクショナリ統計を取得しないシステムでは、構築後だけでなく、大幅なシステム変更により管理オブジェクトが増大した場合にもディクショナリ統計を取得してください。

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計
固定オブジェクトの統計

- V$ビューの元となるX$表のような動的パフォーマンス表や索引（固定オブジェクト）の統計であり、次の2点の理由から取得を強く推奨
  - これらの統計が無い場合、オプティマイザはデフォルト値を使用することになり、不適切な実行計画が選択される可能性有り
  - X$表の統計情報が適切でない場合でも、Dynamic Samplingは機能しない
  - 自動統計収集ジョブでは取得されない
    - DBMS_STATS.GATHER_FIXED_OBJECTS_STATSプロシージャで取得
      - GATHER_TABLE_STATSと同じように統計を取得するが、X$表の特性上、いつでもBLOCKSの値は「0」となる点に注意
  - 固定オブジェクト統計は、表などのオブジェクト統計のように日々変動するものではない為、構築後や、PSR適用時などメジャーアップデートを行った際のメジャーバージョンアップの際には、再取得を推奨

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計
統計が欠落している場合のデフォルト値

- 表のデフォルト値

表統計	デフォルト値
Cardinality	ブロック数×（ブロック・サイズ - キャッシュ層）÷行の平均の長さ
行の平均長	100バイト
ブロック数	100、またはエクステント・マップに基づく実際の値
Remote Cardinality	2000行
Remote 行の平均長	100バイト

- 索引のデフォルト値

表統計	デフォルト値
レベル	1
リーフ・ブロック	25
リーフ・ブロック/キー	1
データ・ブロック/キー	1
個別キー	100
クラスタ化係数	800

【参考】 Oracle Databaseパフォーマンス・チューニング・ガイド11gリリース2（11.2） 13.5.8 統計の欠落の処理

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

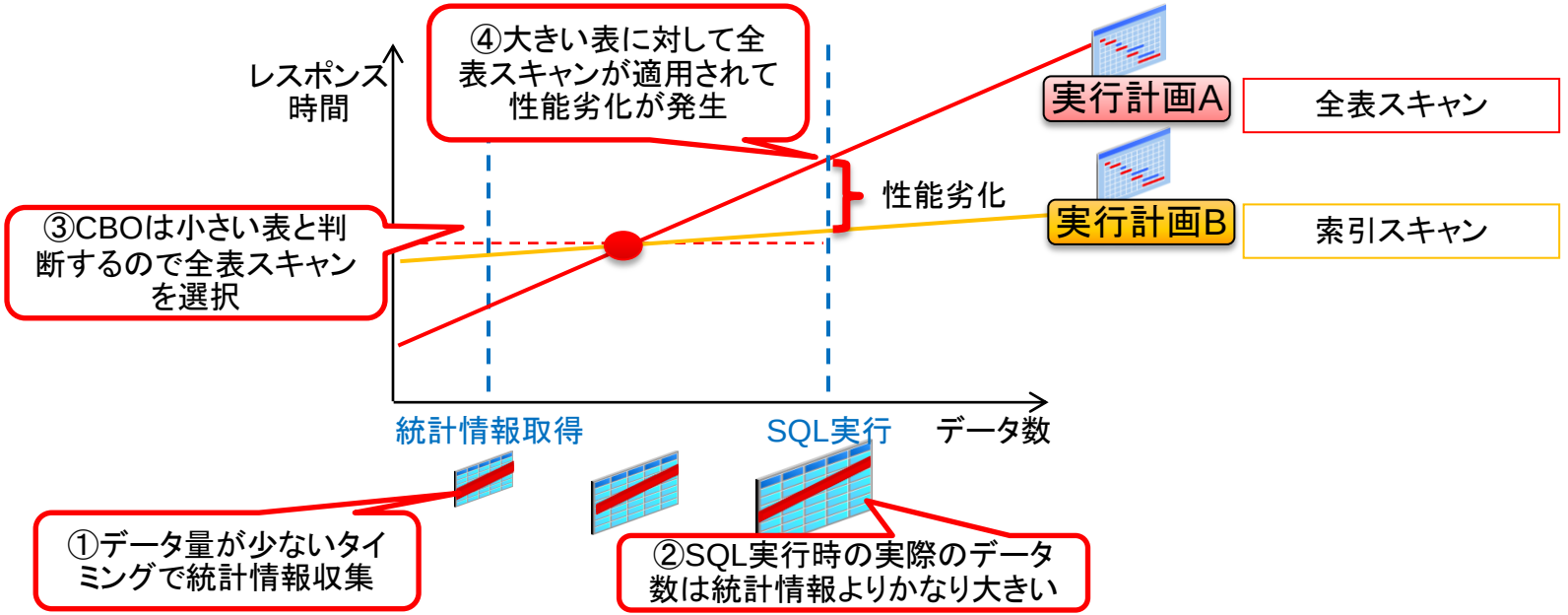
統計収集の効率向上



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

実データと統計情報のかい離のリスク発生メカニズム

- 統計情報による表サイズ見積もりと実際のサイズが大幅に異なる場合
  不適切な実行計画による性能劣化リスクが発生

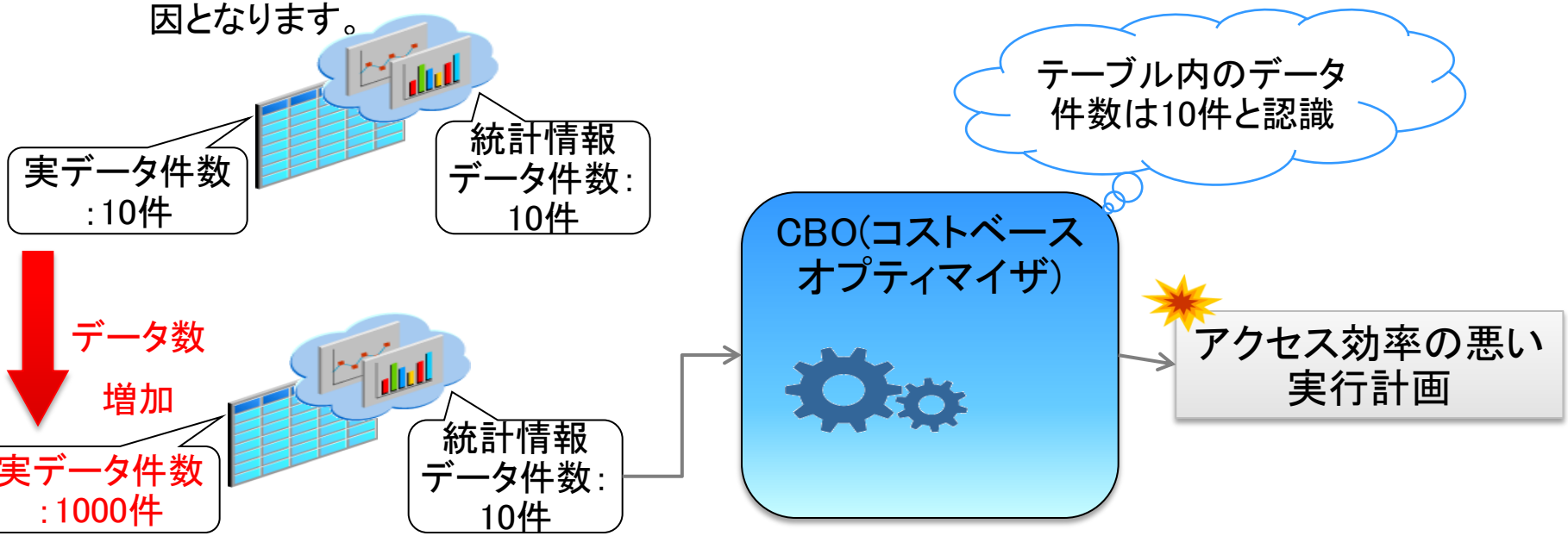


Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報と実データ間の差異(1)

データ件数の差異

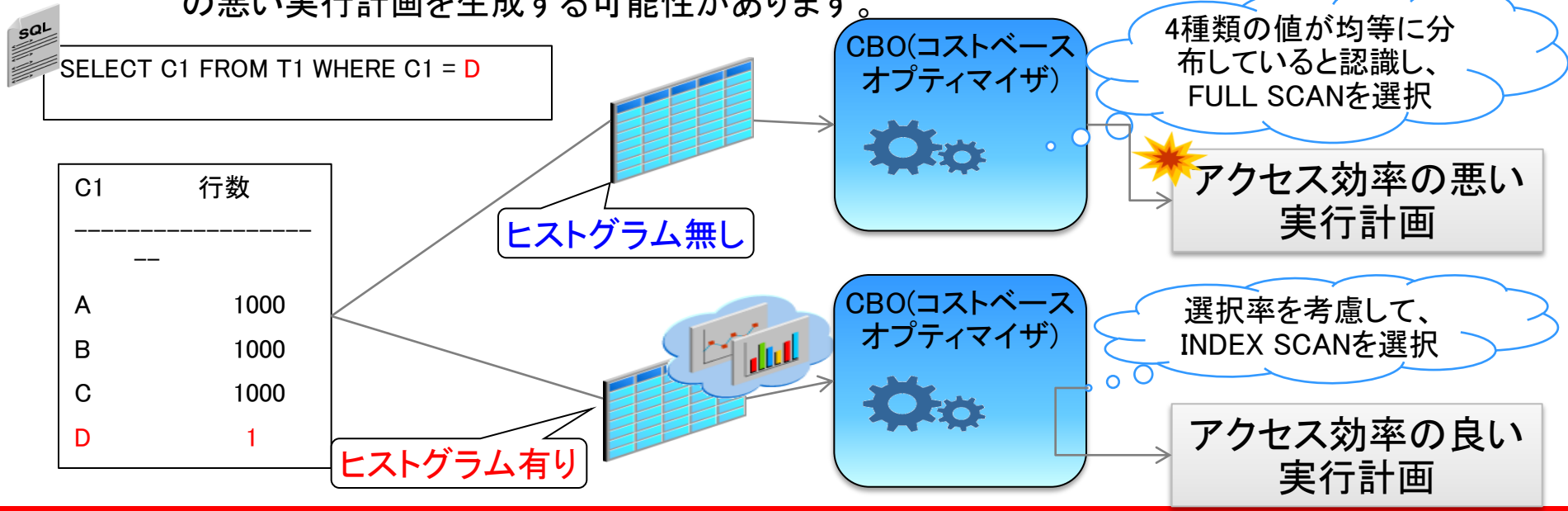
CBOは統計情報からテーブルのデータ件数を認識しています。統計情報が更新されていない場合、実際のデータは変化しているにも関わらず、統計情報上ではデータが変化していないため、結果的にCBOがアクセス効率の悪い実行計画を生成する原因となります。



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報と実データ間の差異(2)
ヒストグラムの有無

テーブル内のデータ件数が同じであっても、データの種類が多い場合と少ない場合では最適な実行計画が異なる場合があります。
ヒストグラムを取得していないとCBOはこの違いを認識する事ができず、アクセス効率の悪い実行計画を生成する可能性があります。

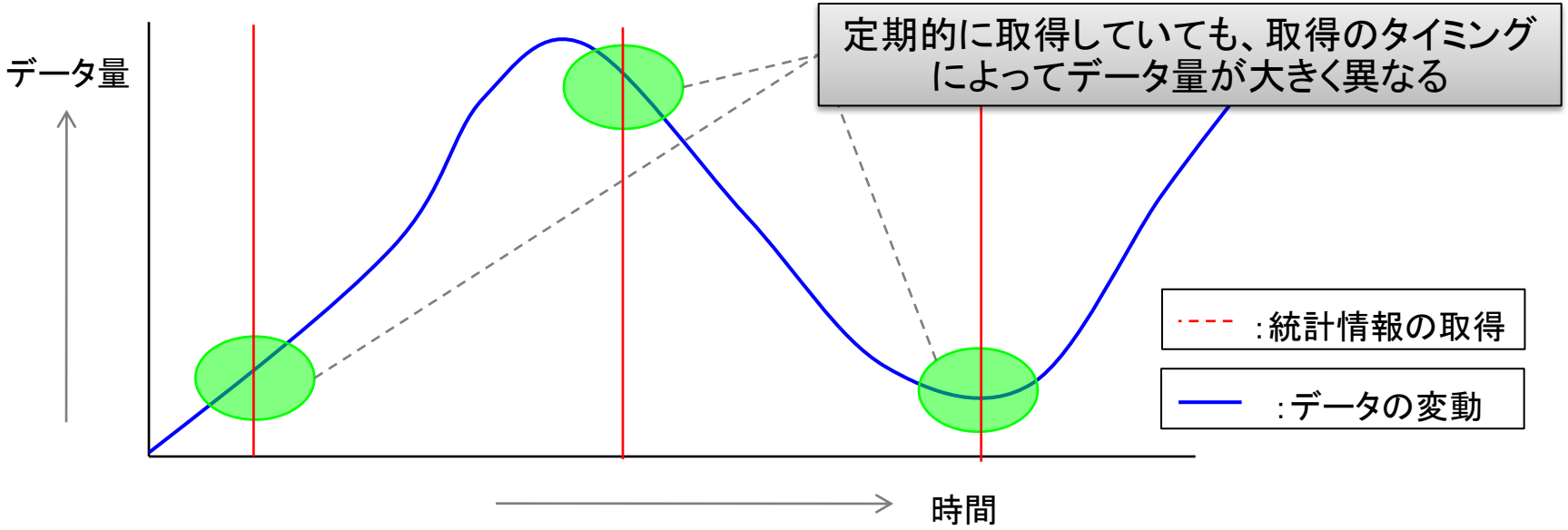


Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

「適切なタイミング」での統計情報取得

定期的な取得≠適切な取得

実データと統計情報間の差異を少なくするためには、「適切なタイミング」での統計情報の取得が重要です。「適切なタイミング」というのは、定期的に統計情報を取得する事ではありません。統計情報の再取得によって、パフォーマンスが悪化する可能性もあるため、どの時点の統計情報を取得するかを十分に考慮する必要があります。



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計情報の収集方法
手動実行コマンド

- DBMS_STATSパッケージ
  - 表単位での統計情報を収集

```
EXECUTE DBMS_STATS.GATHER_TABLE_STATS('SCOTT', 'EMP');
```

  - スキーマ単位（スキーマ内の全てのオブジェクトの統計情報を収集）

```
EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS('SCOTT');
```

  - データベース内のすべてのオブジェクトの統計を収集

```
EXECUTE DBMS_STATS.GATHER_DATABASE_STATS();
```

- ANALYZEコマンド(11gでは非推奨：下位互換性のためにのみサポート)

```
ANALYZE TABLE 表名 COMPUTE STATISTICS;
```

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計情報の収集方法
手動実行時に悩むポイントは？

- プロシージャの実行時間
- オプティマイザ統計情報の精度（正確性）
  - サンプリングサイズは？
  - 今、収集しなければならないセグメントはどれ？
  - 数多くの引数パラメータを正確に理解しているか？
    - ESTIMATE_PERCENT/METHOD_OPT/ DEGREE/NO_INVALIDATE/ DEGREE/CASCADE/OPTIONS/…
    - これらのパラメータは DBMS_STATS.SET_***_PREFSでも定義可能

最適化された、
自動オプティマイザ統計収集
を使用すれば悩まない！

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の効率向上
ESTIMATE_PERCENTパラメータ

- 統計収集する際にサンプリングする表のレコード数の割合を指定パラメータ
  - 次の2つの相反する要件を満たす必要性有り
    - 統計情報としての精度（正確性）
    - 統計収集に要する時間の短縮
  - Oracle Database 10g ~
    - サンプルサイズを表毎に自動的に調整するAUTO_SAMPLE_SIZEが登場
    - ただし、データの偏りが激しい表における精度に問題あり
  - Oracle Database 11g ~
    - 正確な統計情報を短時間で収集するAUTO_SAMPLE_SIZEの改善
    - ”100%のサンプリングの精度”、”10%サンプリングの所要時間”を両立

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の効率向上
ESTIMATE_PERCENTパラメータ（AUTO_SAMPLE_SIZE in 11g ~）

- Oracle Database 11gのAUTO_SAMPE_SIZEは、新しいハッシュ・ベースのサンプリング・アルゴリズム
  - ESTIMATE_PERCENTパラメータのデフォルト設定（DBMS_STATS.AUTO_SAMPLE_SIZE）の場合のみ適用されるアルゴリズム

		1% SAMPLE	100% SAMPLE	AUTO_SAMPLE_SIZE
Elapsed Time (sec)		797	18,772	1,908
NDV	L_ORDERKEY	225,000,000	450,000,000	450,000,000
	L_COMMENT	7,244,885	181,122,127	177,499,684

LINEITEM table(230GB) from the TPC-H benchmark with a scale factor of 300

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の効率向上
METHOD_OPTパラメータ

- 統計収集を実施する際に、ヒストグラムの作成を制御するパラメータ
  - デフォルト値（FOR ALL COLUMNS AUTO）の場合、自動的にヒストグラムを作成すべき列を決定するが、その際に使用される情報は、次の2つ
    - 列のNDV
    - 列の使用実績：Where句で使用された回数（=, <, >, like, is Null…）
  - 列の使用実績は、DBMS_STATS.REPORT_COL_USAGEファンクションで確認可能
    - 初期化パラメータstatistics_levelでモニタリング制御
      - BASIC：全ての表に関してSQL文の監視をしない
      - TYPICAL(デフォルト)/ALLの場合、一時表以外の表へのSQL文を監視

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の効率向上
DEGREEパラメータ

- 統計収集で内部的に実行されるSQL文のパラレル度を指定するパラメータ
  - デフォルト設定（NULL）では、対象表のDEGREE属性に従う
    - 表のDEGREE属性のデフォルト設定は「1」の為、シリアル実行がデフォルト
  - DBMS_STATS.AUTO_DEGREEを設定した場合
    - 表サイズを基に、自動的にパラレル度を決定
    - 選択されうるパラレル度は、1～ DBMS_STATS.DEFAULT_DEGREE
      - DEFAULT_DEGREE = （PARALLEL_THREADS_PER_CPU × CPU_COUNT）

    → 各表のサイズを分析して個別設定（SET_TABLE_PREFS）を行うより、SET_GLOBAL_PREFSでデフォルト設定を変更する方が効率的

ORACLE
Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の効率向上

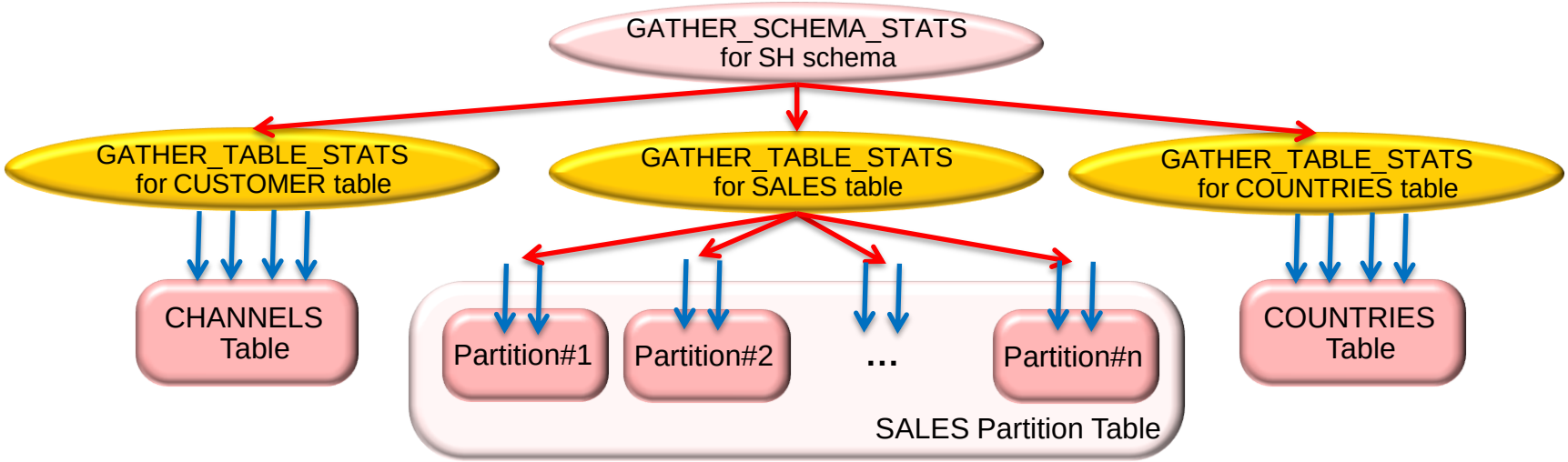
CONCURRENT in Oracle Database 11g Release 2(11.2.0.2)~

- 複数オブジェクトの統計を並列で収集するか否かを指定するパラメータ
  - SET_GLOBAL_PREFSで「TRUE」に設定（デフォルト：FALSE）した場合、複数のオブジェクトの統計を並列で収集し、総所要時間を大幅に短縮することが可能
    - 1つの表の統計をパラレル実行で収集するDEGREEより上位での並列化
      - 特に、多数の小さなサイズの表があるデータベースでは、DEGREEよりもCONCURRENTによる恩恵を受け易い（組合せ使用は可能）
  - 制限
    - 並列可能な数は、JOB_QUEUE_PROCESS初期化パラメータに依存
    - デッドロックの可能性排除の為、並列化が許可されるパーティション表は1つ
    - 自動統計収集ジョブは、この設定に関わらず表毎にシリアルで収集

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の効率向上
CONCURRENT & DEGREEパラメータ

- 各パラメータが影響する階層が異なる為、共存させることが可能
  - CONCURRENT： 複数の表を同時に統計収集（→）
  - DEGREE： ある一つの表をパラレル実行で統計収集（→）



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の効率向上
STALE_PERCENTパラメータ

- 表の統計情報が古い（失効）していると判断する閾値を指定するパラメータ
  - デフォルト値は10（%）であり、SET_*_PREFSで設定変更可能
  - 次の場合にこのパラメータの値が使用される
    - 自動統計収集ジョブの実行時
    - GATHER_DATABASE/SCHEMA_STATSプロシージャ実行時
      - ただし、OPTIONS='GATHER AUTO' or 'GATHER STALE'が必要
  - 統計情報が失効している表の確認方法
    - DBMS_STATS.GATHER_DATABASE/SCHEMA_STATS
    - Doc ID 252597.1
      - USER_TAB_MODIFICATIONSディクショナリ・ビュー
- STATISTICS_LEVELがTYPICALまたはALLに設定されている場合にデフォルトで有効

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の効率向上
CASCADE

- 表の統計収集の一環で索引統計を収集するか否かを指定するパラメータ
  - デフォルト値（DBMS_STATS.AUTO_CASCADE）の場合、
    統計が古い表の索引の統計情報が再収集される
  - CASCADEパラメータの設定を「FALSE」へ変更した方が良いケース有り
    - 大量データ挿入を高速化する為に一時的に索引を無効化するような表
      - 大量データ挿入が完了後に索引を再作成する際、
        自動的に索引の統計情報は収集される為
      - 表の統計収集の一環で、再度索引の統計を収集ことは二度手間

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の効率向上
NO_INVALIDATE

- 統計を収集した表を使用する実行計画（依存カーソル）を直ぐに無効化するか否かを指定するパラメータ
  - デフォルト値（DBMS_STATS.AUTO_INVALIDATE）の場合、直ぐに無効化せず、そのタイミングは自動的に決定される
    - デフォルト1800秒までの範囲内でランダムに存続期間を決定
      - この範囲を超えた後、カーソルが再利用される際に、新しい統計情報を使用した実行計画が解析（ハード・パース）される
  - ハード・パースに要するCPU消費量は大きい為、全てのカーソルが無効化された場合、ハード・パースが集中的に実行され、性能劣化が発生し易い
  - ランダム値を用いてハード・パースのタイミングを分散し、CPU消費量を平準化することで、性能への影響を極小化

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の効率向上
GRANULARITYパラメータ

- パーティション表の場合に収集する統計情報の粒度を指定するパラメータ
  - 粒度のレベルは、表（グローバル）、パーティション、サブ・パーティションの3つ
  - デフォルト値（AUTO）の場合、パーティション方法に依存して粒度を自動決定
    - パーティション・レベルの統計は、パーティション方法を問わず収集
    - サブ・パーティション・レベルの統計は、サブ・パーティションの方法が”LIST” or ”RANGE”の場合のみ収集
  - Oracle Database 11g Release 2 ~
    - ”APPROX_GLOBAL AND PARTITION”は、一部のパーティションのみの統計を再収集した際、表全体を読み込まずに、表レベルの統計情報を更新
  - 各設定値における統計収集のマトリックス：KROWN# 149661

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

GRANULARITYパラメータ
PARTITIONとAPPROX_GLOBAL AND PARTITIONの動作比較

- granularity => ‘PARTITION’
  - 表レベルの統計を取得していない場合、
    全パーティションの統計を取得すると、表レベルの統計を計算
  - 表レベルの統計を明示的に取得している場合、一部のパーティションの統計を再取得しても、表レベルの統計は更新されない

- granularity => ‘APPROX_GLOBAL AND PARTITION’
  - 表レベルの統計を明示的に取得している場合でも、一部のパーティションの統計を再取得した場合、表レベルの統計が自動的に再計算、更新される

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の効率向上
INCREMENTALパラメータ

- 表全体ではなく、変更されたパーティションのみをスキャンすることで、グローバル統計を更新可能か否かを指定するパラメータ
  - 使用条件（次の4つのパラメータが設定されていること）
    - SET_*_PREFSで事前設定
      - INCREMENTAL → TRUE
      - PUBLISH → TRUE
    - GATHER_*_STATS実行時に指定 or SET_*_PREFSで事前設定
      - ESTIMATE_PERCENT → AUTO_SAMPLE_SIZE
      - GRANULARITY → AUTO(GLOBALを含む)
  - 注意：INCREMENTALパラメータはサブ・パーティションには適用されない

ORACLE
Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

自動オプティマイザ統計収集ジョブ
手動実行(GATHER_DATABASE_STATS)との違い

- 自動統計収集ジョブは、自動メンテナンス・タスクの1つとして、定められたウィンドウの間、プライオリティの高いオブジェクトから統計収集を実施
  - Oracle Database 11g Release 2でのデフォルト・ウィンドウは次の通り
    - 月~金曜日:22時~26時(翌2時までの4時間)
    - 土~日曜日:6時~26時(翌2時までの20時間)
  - 内部プロシージャのGATHER_DATABASE_STATS_JOB_PRCをコール

→OPTIONS=>'GATHER AUTO'が設定されたGATHER_DATABASE_STATSによる統計収集とほとんど同じ

→ただし、最も統計収集を必要とするオブジェクトから実施するプライオリティ付けが自動統計収集ジョブでは実装されている

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報収集時間帯の変更

Enterprise Managerからウィンドウの編集



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報収集時間帯の変更

Enterprise Managerからタスクの実行曜日を編集



オプティマイザ統計収集を
指定された曜日別に実行指定

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

自動オプティマイザ統計収集ジョブ
RAC環境においてジョブを実行するノード(サービス)を設定

- 自動メンテナンスタスク(別名、AUTOTASKクライアント)に関する管理は、DBMS_AUTO_TASK_ADMINパッケージを使用
  - 自動統計収集ジョブはAUTOTASKクライアントの1つであり、名称は「auto optimizer stats collection」で登録されている
  - DBMS_AUTO_TASK_ADMIN.SET_CLIENT_SERVICEプロシージャで指定したサービスに関連付け

→ RAC環境において自動統計収集ジョブ用のサービスを作成して関連付けることで、自動統計収集ジョブが優先的に実行されるノードを設定可能

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の管理
DBMS_STATSパッケージ機能一覧

- オプティマイザ統計情報の収集
- 統計情報の設定または取得
- 統計情報の削除
- 統計情報の転送
- 統計情報のロックまたはロック解除
- 統計履歴のリストアおよびパージ
- 保留中の統計情報の削除、抽出、公開
- 統計情報の比較
- 統計情報の拡張

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の保留（Pending Statistics）

収集した統計情報を直ぐに公開しない

- デフォルトでは、統計収集直後に新しい統計情報をディクショナリ表へ公開
  - SET_*_PREFSプロシージャでPUBLISHパラメータを”FALSE”（デフォルト：TRUE）に設定することで、公開を保留させることが可能
  - この機能により、新しい統計情報を評価後に公開するステップが実現可能

  - 保留中の統計は、USER_*_PENDING_STATSディクショナリ・ビューで確認
  - 初期化パラメータOPTIMIZER_USE_PENDING_STATISTICSを”TRUE”に設定することで、保留中の統計を使用した実行計画を作成することが可能

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の比較（Comparing Statistics）
統計情報の変化を確認

- DBMS_STATS.DIFF_TABLE_STATS_* ファンクションで統計情報を比較
  - DIFF_TABLE_STATS_IN_STATTAB
    - 2つの異なるユーザー統計表
    - statidで識別可能な2つの統計セットが含まれている単一のユーザー統計表
    - ユーザー統計表とディクショナリ履歴
  - DIFF_TABLE_STATS_IN_HISTORY
    - 過去の2つのタイムスタンプにおける統計情報
  - DIFF_TABLE_STATS_IN_PENDING
    - 保留中の統計情報と過去もしくは現時点の統計情報

※ 依存オブジェクト（索引、列、パーティション）の統計も比較し、統計情報の違いが閾値（デフォルト10% or 引数で指定）を超えた場合に、両方のソースからオブジェクトの統計情報を表示

ORACLE
Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計
まとめ

- 最適な実行計画を選択する為には精度の高い統計情報が必要不可欠
  - これを実現する為に、DBMS_STATSパッケージの機能は進化し続けている
  - 自動統計収集ジョブは、それらの機能を簡単に最大限に活用する仕組み



  - 自動統計収集ジョブの使用を推奨
    - DBA業務は次の作業ポイントを押さえるのみ
      - ウィンドウ設定で、実行される時間帯を制御
      - バッチ処理で一時的に使用される表の統計をロック
      - SQL計画管理の使用は最小限にすることで運用コスト低減

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

課題のパターン別
統計情報収集のポイント



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報収集のポイント
発生頻度の高い障害と防止策

問題発生パターン	発生原因	対処	未然防止
① 夜間バッチ実行時間帯の負荷が高い			

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

問題発生パターン①

バッチ処理実行時間帯の負荷が高い

- 自動統計情報収集のデフォルト収集時間帯は平日夜間と休日であり、夜間バッチや休日バッチとぶつかる可能性が高い
  - バッチ処理時間帯と統計情報収集時間帯をずらすことで、バッチの性能を安定させることが可能
- デフォルトの収集タイミング

10g
月-金曜日の毎日22-6時の間
土曜日0時-月曜日0時の間

11g
- 月-金曜日の毎日22-2時の間
- 土曜日6時-日曜日2時の間

- デフォルトの収集条件
  - 統計情報をまだ収集していないオブジェクト
  - 10%以上更新され、既存統計情報では最新状態を表せていない可能性のあるオブジェクト

ORACLE
Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報収集のポイント

発生頻度の高い障害と防止策

問題発生パターン	発生原因	対処	未然防止
① 夜間バッチ実行時間帯の負荷が高い	自動統計情報収集の収集時間帯とのバッティング	暫定対応として、自動統計情報収集を止めて負荷を減らす。その後、収集タイミングの変更	自動統計情報収集の時間帯設定を変更

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報収集のポイント
発生頻度の高い障害と防止策

問題発生パターン	発生原因	対処	未然防止
① 夜間バッチ実行時間帯の負荷が高い	自動統計情報収集の収集時間帯とのバッティング	暫定対応として、自動統計情報収集を止めて負荷を減らす。その後、収集タイミングの変更	自動統計情報収集の時間帯設定を変更
② バッチ処理が突然遅くなった			

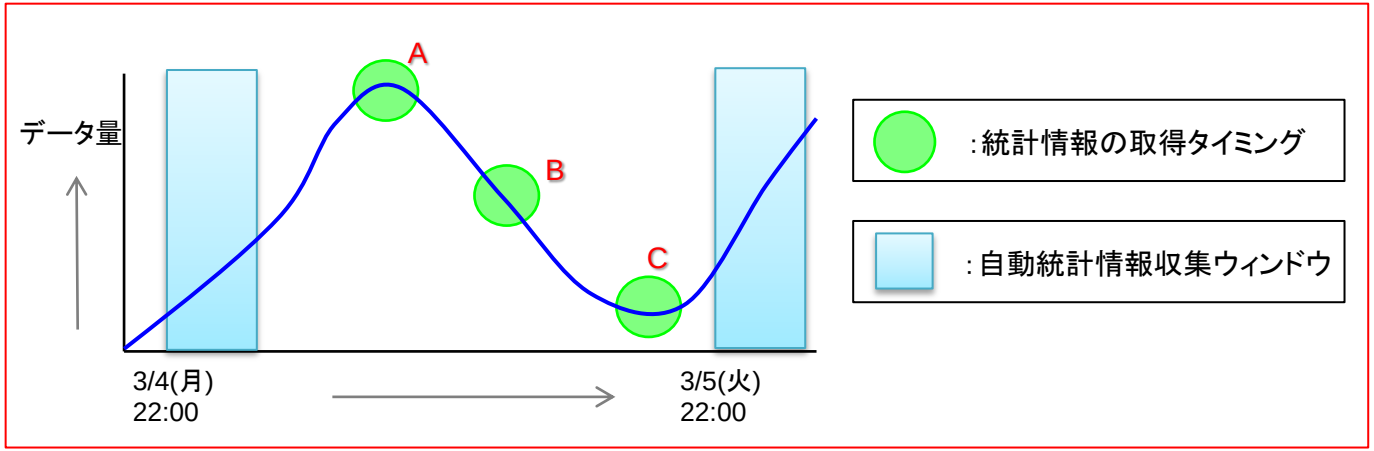
Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

問題発生パターン②

バッチ処理が突然遅くなった

- データ変動のある表(洗い替え等) において、データが空の際に統計情報が収集され、不適切な実行計画(例：全表スキャン)が立てられた

A ✓ データの件数が最も増加
B ✓ データ件数が平均値
C ✓ データ件数が最も減少

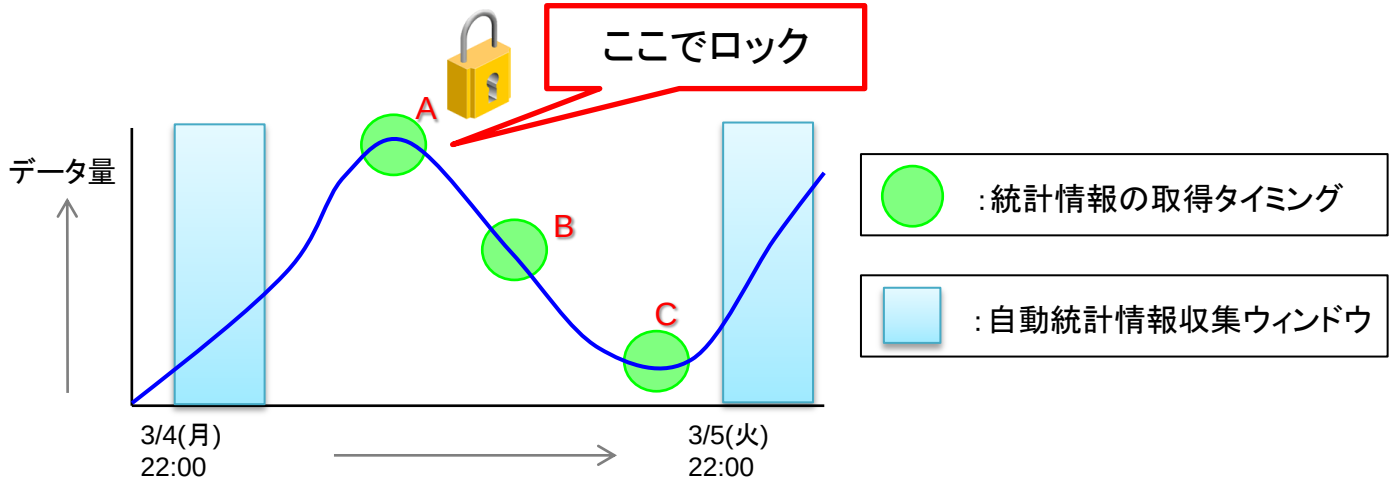


最適なタイミングは A での取得

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報のロック(固定化)
DBMS_STATS.LOCK_TABLE_STATS

- この図のように1日の間にデータ量がガラリと変わるテーブルとしては、バッチ処理の中でだけ使われるワークテーブルが代表的



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報のロック(固定化)
DBMS_STATS.LOCK_TABLE_STATS

- 例) 特定表の統計収集を止める

```
EXECUTE DBMS_STATS.LOCK_TABLE_STATS('SCOTT','EMP');
```

- 例) スキーマ内のすべてのオブジェクトの統計収集を止める

```
EXECUTE DBMS_STATS.LOCK_SCHEMA_STATS('SCOTT');
```

- 例) 表の統計がロックされているかどうかの確認

```
SELECT owner,table_name,stattype_locked FROM dba_tab_statistics;
```

ORACLE

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報収集のポイント
発生頻度の高い障害と防止策

問題発生パターン	発生原因	対処	未然防止
① 夜間バッチ実行時間帯の負荷が高い	自動統計情報収集の収集時間帯とのバッティング	暫定対応として、自動統計情報収集を止めて負荷を減らす。その後、収集タイミングの変更	自動統計情報収集の時間帯設定を変更
② バッチ処理が突然遅くなった	統計情報の収集タイミングが不適切	統計情報を再収集してロック。 該当SQLの実行計画をヒント/SPMで固める	バッチ処理の中でのみ使うようなワークテーブルは統計情報を最大量でロック 該当SQLの実行計画をヒント/SPMで固める

ORACLE
Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報収集のポイント
発生頻度の高い障害と防止策

問題発生パターン	発生原因	対処	未然防止
① 夜間バッチ実行時間帯の負荷が高い	自動統計情報収集の収集時間帯とのバッティング	暫定対応として、自動統計情報収集を止めて負荷を減らす。その後、収集タイミングの変更	自動統計情報収集の時間帯設定を変更
② バッチ処理が突然遅くなった	統計情報の収集タイミングが不適切	統計情報を再収集してロック。 該当SQLの実行計画をヒント/SPMで固める	バッチ処理の中でのみ使うようなワークテーブルは統計情報を最大量でロック 該当SQLの実行計画をヒント/SPMで固める
③ 統計情報は取得後に性能がダウン			

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

問題発生パターン③
統計情報取得後に性能ダウン

- 統計情報を取得する時間を短縮したい為に、ESTIMATE_PERCENTパラメータでサンプリング・サイズが小さく設定
  - これにより、精度が低い（現実と乖離がある）統計情報が取得され、適切ではない実行計画が選択されてしまったケース

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

AUTO_SAMPLE_SIZEで統計情報を取得
ESTIMATE_PERCENTパラメータで指定（デフォルト）

- 統計収集する際にサンプリングする表のレコード数の割合をESTIMATE_PERCENTパラメータで指定可能
  - 統計情報としての精度（正確性）と統計収集に要する時間の短縮の2つの相反する要件を満たす必要性有り
  - Oracle Database 10g ~
    - サンプルサイズを表毎に自動的に調整するAUTO_SAMPLE_SIZEが登場
    - ただし、データの偏りが激しい表における精度に問題あり
  - Oracle Database 11g ~
    - 正確な統計情報を短時間で収集するAUTO_SAMPLE_SIZEの改善

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報の自動バックアップ
統計情報取得後に性能問題が発生した場合の復旧方法①

- 統計情報取得による性能劣化に備えて過去の統計情報を保存
- DBMS_STATS で取得した場合、前の統計をSYSAUX 表領域に自動保存
  - デフォルトで31日間分保存 / 統計情報取得による性能劣化時にはリストア可能
  - DBMS_STATS で統計情報のリストアや消去、保存期間の変更が可能
    - データベース内の全ての表の統計情報を指定したタイムスタンプでリストア
      - DBMS_STATS.RESTORE_DATABASE_STATS プロシージャ
    - 特定スキーマ内の全ての表の統計情報を指定したタイムスタンプでリストア
      - DBMS_STATS.RESTORE_SCHEMA_STATS プロシージャ
    - 指定した表の統計情報を指定したタイムスタンプでリストア
      - DBMS_STATS.RESTORE_TABLE_STATS プロシージャ

※過去の統計はDBA_TAB_STATS_HISTORYにて確認可能

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報のexport/import
統計情報取得後に性能問題が発生した場合の復旧方法②

- 統計情報をSTAT_TABLEへExportしておくことで、
  後々Importすることにより、当時と同じプランを選択させることが可能

① 統計格納する表の作成
```
begin
DBMS_STATS.CREATE_STAT_TABLE (
  ownname  =>'SCOTT',
  stattab  =>'STAT_TAB',
  tblspace =>'USERS');
end;
/
```

② 統計情報のExport
```
begin
DBMS_STATS.EXPORT_TABLE_STATS (
  ownname   => 'SCOTT',
  tabname   => 'EMP',
  partname  => NULL,
  stattab   => 'STAT_TAB',
  statid    => '2004/01/14',
  cascade   => TRUE,
  statown   => NULL);
end;
/
```

③ 統計情報のImport
```
begin
DBMS_STATS.IMPORT_TABLE_STATS (
  ownname   => 'SCOTT',
  tabname   => 'EMP',
  partname  => NULL,
  stattab   => 'STAT_TAB',
  statid    => '2004/01/14',
  cascade   => TRUE,
  statown   => NULL);
end;
/
```

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

統計情報収集のポイント
発生頻度の高い障害と防止策

問題発生パターン	発生原因	対処	未然防止
① 夜間バッチ実行時間帯の負荷が高い	自動統計情報収集の収集時間帯とのバッティング	暫定対応として、自動統計情報収集を止めて負荷を減らす。その後、収集タイミングの変更	自動統計情報収集の時間帯設定を変更
② バッチ処理が突然遅くなった	統計情報の収集タイミングが不適切	統計情報を再収集してロック。 該当SQLの実行計画をヒント/SPMで固める	バッチ処理の中でのみ使うようなワークテーブルは統計情報を最大量でロック 該当SQLの実行計画をヒント/SPMで固める
③ 統計情報は取得後に性能がダウン	不適切なサンプルサイズで統計情報を収集した	暫定対応として、統計情報の自動バックアップからの戻し作業	AUTO_SAMPLE_SIZEを利用した統計情報の取得 PENDING統計を使用した事前評価

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計情報収集のポイント
まとめ

問題の未然防止

1. 自動統計情報取得の取得時間帯を負荷が低い時間帯に設定する
2. バッチ処理時にしかデータを入れないテーブルなど短時間の間にデータ量の増減が激しいテーブルは最大量で統計情報を固定
3. 精度の高い統計情報を取得する

問題発生時の復旧

1. 自動統計情報は31日間バックアップが取られているため、バックアップからリストアを行う
2. 開発環境等で安定した実行計画を生成することを確認済の統計情報をexport/import で移行する

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

まとめ



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

目的とゴール

目的

オプティマイザ統計情報の特徴を理解し運用方法を身につける

ゴール

- 統計情報収集の各種パラメータを理解する
- システムにあった選択ができるようになる
- 課題発生時の改善ができるようになる

ORACLE

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

まとめ

- 実行計画とはSQL実行時に内部的に使われるアクセスパスのこと
- 実行計画はCBOにより統計情報を重要なファクターとして計算される
- 統計情報は実データとかい離がないことが重要
- 適切な実行計画を得るためにはリスクのない統計情報収集タイミングの見極めが重要
- リスクは統計情報の持つ表サイズが実際の表サイズより小さい値になる場合に生じます
- リスクを最小化するには、テーブルサイズが極大になるところでの統計情報収集する方針を推奨
- データ変動に伴う実行計画の最適化より安定化を望む重要SQLについてはSPMにより実行計画を制御・管理
- 基本は統計情報を定期的に取得し全体的なSQLの実行計画の最適化し、重要SQLに関しては別途SPMで制御し管理する運用を推奨

  ※実行計画の変化による性能劣化に備えて統計情報のバックアップは定期的に取得

ORACLE

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

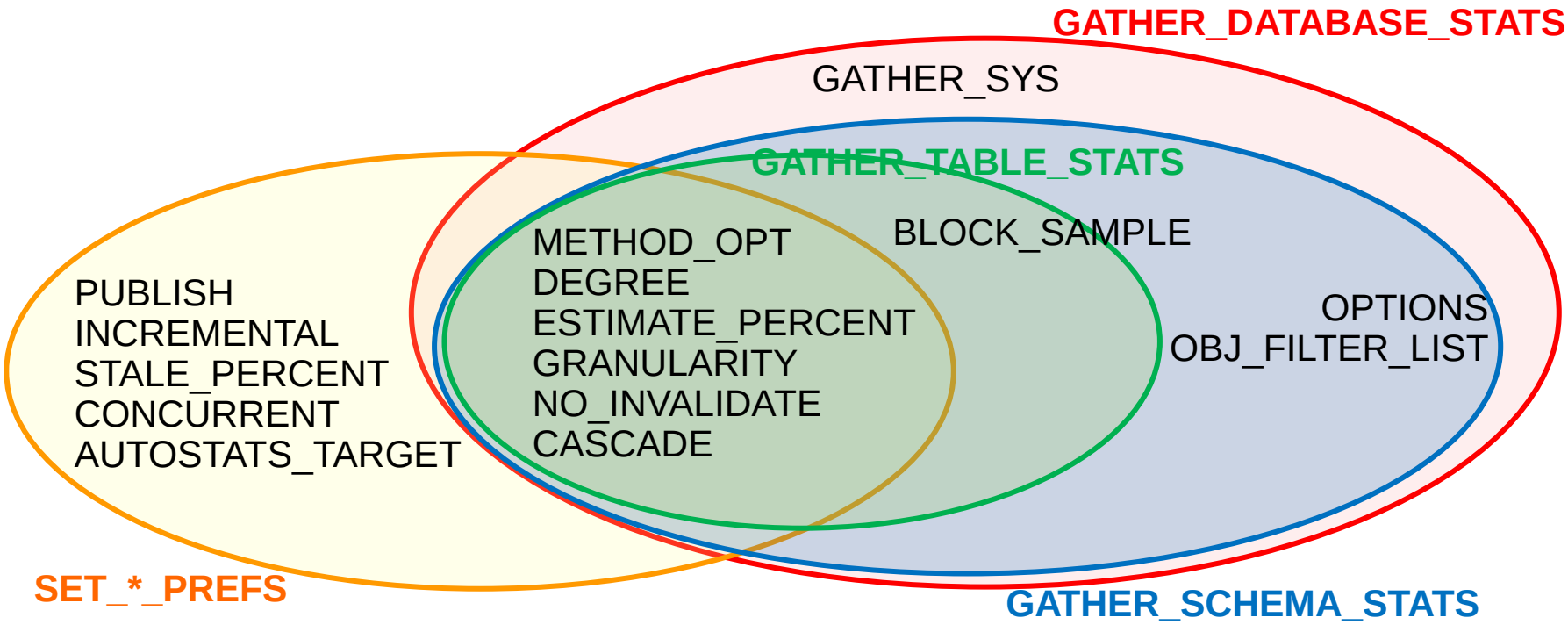
Appendix

- プロシージャ毎の設定可能なパラメータ(DBMS_STATSパッケージ)
- ヒストグラムに関する追加情報
  - 12c 位頻度ヒストグラム
  - 12c ハイブリッド・ヒストグラム
  - 11g R1 以降の拡張統計
- その他関連機能

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

プロシージャ毎の設定可能なパラメータ

DBMS_STATSパッケージ



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

上位頻度ヒストグラム

- 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

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

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

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

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



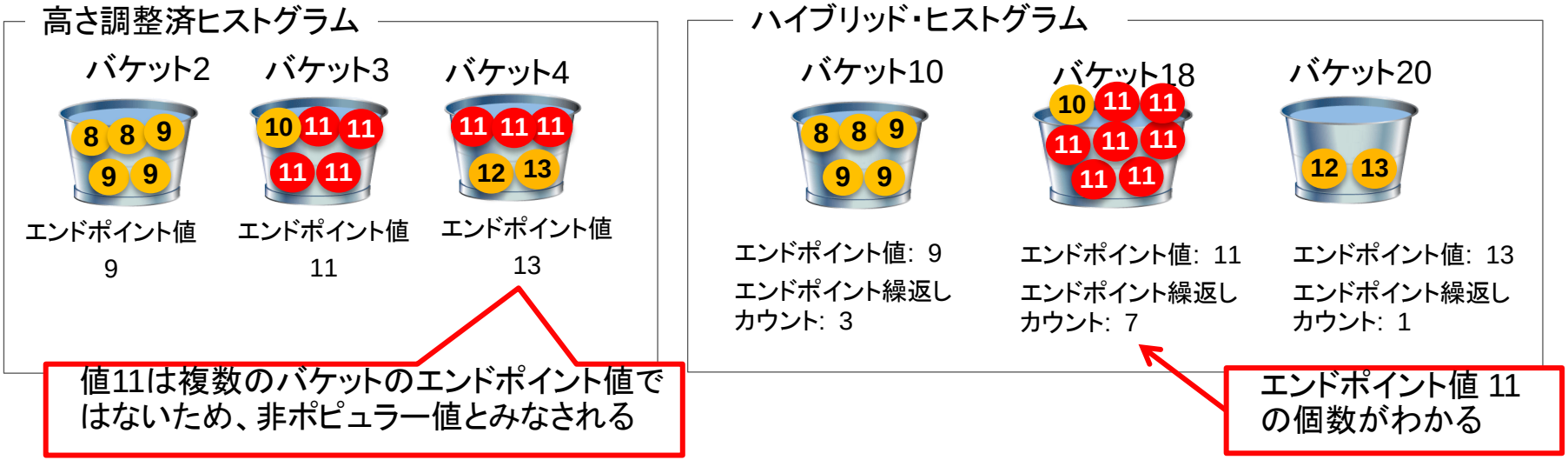
バケット20

エンドポイント値: 13
エンドポイント繰返しカウント: 1

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

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



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

オプティマイザ統計

拡張統計：Oracle Database 11g Release 1 ~

- 複数列の統計（Column Groups）
  - 同一表内の複数列に跨る統計を保持することで、列データ間の相関関係を考慮したCardinalityの計算を可能とする
  - 例：CUSTOMERS表のCUST_STATE_PROVINCE列とCOUNTRY_ID列

- 式の統計（Expression Statistics）
  - 関数を含めた統計を保持することで、関数に組み込まれた列を持つWhere句におけるCardinalityの計算を可能する
  - 例：Where UPPER（CUST_LAST_NAME) = :B1

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

拡張統計：複数列統計の取得

- デフォルトで、ヒストグラムと同様に、ワークロードにも基づいて列グループが作成される
- DBMS_STATS.CREATE_EXTENDE_STATSファンクションを使用して手動で列グループを作成することも可能
  - 一度、列グループを作成すると、その表の統計を取得する際に自動的に更新

```
SQL> SELECT DBMS_STATS.CREATE_EXTENDED_STATS(null, 'ACCOUNT', '(CITY, COUNTRY)') FROM dual;

DBMS_STATS.CREATE_EXTENDED_STATS(NULL,'ACCOUNT','(CITY,COUNTRY)')
------------------------------------------------------------------------------
SYS_STU3UJN5IT#IIA5ASKY72Q8V6Z

SQL>
SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS(null,'ACCOUNT');

PL/SQL procedure successfully completed.
```

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

拡張統計：複数列統計の管理

- 複数列統計の確認
  - USER_TAB_COL_STATISTICS

```
SQL> select COLUMN_NAME, NUM_DISTINCT, NUM_NULLS, HISTOGRAM from USER_TAB_COL_STATISTICS;

COLUMN_NAME                    NUM_DISTINCT  NUM_NULLS HISTOGRAM
------------------------------ ------------ ---------- ---------------
SYS_STU3UJN5IT#IIA5ASKY72Q8V6Z          300          0 NONE
```

- 列グループの定義の確認
  - USER_STAT_EXTENTIONS

```
SQL> select TABLE_NAME, EXTENSION_NAME, EXTENSION from USER_STAT_EXTENSIONS;

TABLE_NAME   EXTENSION_NAME                 EXTENSION
------------ ------------------------------ ------------------------
ACCOUNT      SYS_STU3UJN5IT#IIA5ASKY72Q8V6Z ("CITY","COUNTRY")
```

- 列グループの削除
  - DBMS_STATS.DROP_EXTENDED_STATSファンクション

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

拡張統計：式の統計の取得

- 作成/取得方法、および管理方法は、全て複数列統計と同じ

```
SQL> SELECT DBMS_STATS.CREATE_EXTENDED_STATS(null, 'ACCOUNT', '(UPPER(LASTNAME))') FROM dual;

DBMS_STATS.CREATE_EXTENDED_STATS(NULL,'ACCOUNT','(UPPER(LASTNAME))')
--------------------------------------------------------------------------------
SYS_STU3C9HWMAGNUMG9TQRL9SVKEK


SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS(null, 'ACCOUNT');


SQL> select COLUMN_NAME, NUM_DISTINCT, NUM_NULLS, HISTOGRAM from USER_TAB_COL_STATISTICS;

COLUMN_NAME                    NUM_DISTINCT  NUM_NULLS HISTOGRAM
------------------------------ ------------ ---------- ---------------
SYS_STU3C9HWMAGNUMG9TQRL9SVKEK          100          0 NONE


SQL> select TABLE_NAME, EXTENSION_NAME, EXTENSION from USER_STAT_EXTENSIONS;

TABLE_NAME   EXTENSION_NAME                 EXTENSION
------------ ------------------------------ ------------------------
ACCOUNT      SYS_STU3C9HWMAGNUMG9TQRL9SVKEK (UPPER("LASTNAME"))
```

ORACLE

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

拡張統計：制限

- 拡張統計はWhere句で「=」と「INリスト」が使用されている場合のみ適用
- 使用上の注意(参考：PL/SQLパッケージ 「CREATE_EXTENDED_STATS」)
  - 拡張統計に仮想列を含めることはできない
  - SYSが所有する表には拡張を作成できない
  - クラスタ表、索引構成表、一時表または外部表には拡張統計を作成できない
  - 列グループ内の列の数は、2から32の範囲
  - 1つの列を1つの列グループ内で複数回使用することはできない
  - 列グループに式を含められない
  - 式には1つ以上の列が含まれている必要がある
  - 式に副問合せを含めることはできない
  - COMPATIBLEは、11.0.0.0.0以上

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

その他の関連機能



Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

Dynamic Sampling / Dynamic Statistics

- 最適な実行計画を選択する際、標準のオプティマイザ統計が不適切な場合に対象データを一部サンプリングすることで統計情報を拡充する為の機能
  - 標準のオプティマイザ統計の代わりではない
  - 初期化パラメータ:OPTIMIZER_DYNAMIC_SAMPLIGで設定(デフォルト:2)
    - 設定可能な値の範囲は「0~10」であり、各レベルでは次の2つが制御される
      - どのような場面でDynamic Samplingを使用するのか
      - サンプリングのサイズ
    - Oracle 11g Release 2 ~ 次の条件を満たす際にレベルを自動設定
      - OPTIMIZER_DYNAMIC_SAMPLINGを明示設定していない
      - パラレル実行で、大きな表へアクセスする、複雑なWhere句を含むSQL
  - 動的サンプリングは11.2.0.4/12cから動的統計(Dynamic Statistics)という名称に変更
    - 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 OPTIMIZER_DYNAMIC_SAMPLING=11

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

Dynamic Sampling / Dynamic Statistics

OPTIMIZER_DYNAMIC_SAMPLINGの設定値毎の動作

Level	When	Samle Size(Blocks)
0	動的サンプリングをしない	N/A
1	問合せ対象表に非パーティション表で索引がなく統計情報が取得されていない表がある	32
2	少なくとも1つの表に統計がない場合。デフォルトの設定	64
3	2で対象となる表に加えて、WHERE句で列に関数や演算子を使用した式で条件を指定されている表	64
4	3で対象となる表に加えて、2つ以上の列に対する条件がWHERE句で指定されている表	64
5	動的サンプリングが行われる表は4と同様	128
6	動的サンプリングが行われる表は4と同様	256
7	動的サンプリングが行われる表は4と同様	512
8	動的サンプリングが行われる表は4と同様	1024
9	動的サンプリングが行われる表は4と同様	4086
10	動的サンプリングが行われる表は4と同様	All Blocks
11	オプティマイザが必要と判断した場合は、常に自動的に動的統計が使用	AUTO

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

Dynamic Sampling / Dynamic Statistics

拡張統計が使用できないケースで効果有り

- 相関関係が存在する二つの列に対する範囲検索を含むSELECT文
  - 標準の統計情報が不適切であり、拡張統計も使用できない（範囲検索の為）

```
SQL> select count(*) FROM SH.SALES where CUST_ID < 2222 AND PROD_ID > 5 ;
```

```
* OPTIMIZER_DYNAMIC_SAMPLING is set to 0
-------------------------------------------------------------------------------------
| Id  | Operation                    | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |       |       |       |   528 (100)|          |
|   1 |  SORT AGGREGATE              |       |     1 |     9 |            |          |
|   2 |   PARTITION RANGE ALL        |       | 20197 |   177K|   528   (2)| 00:00:07 |
|*  3 |    TABLE ACCESS STORAGE FULL| SALES | 20197 |   177K|   528   (2)| 00:00:07 |
-------------------------------------------------------------------------------------
```

```
* OPTIMIZER_DYNAMIC_SAMPLINT is set to 6
-------------------------------------------------------------------------------------
| Id  | Operation                    | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |       |       |       |   528 (100)|          |
|   1 |  SORT AGGREGATE              |       |     1 |     9 |            |          |
|   2 |   PARTITION RANGE ALL        |       |  206K |  1813K|   528   (2)| 00:00:07 |
|*  3 |    TABLE ACCESS STORAGE FULL| SALES |  206K |  1813K|   528   (2)| 00:00:07 |
-------------------------------------------------------------------------------------
- dynamic sampling used for this statement (level=6)
```

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

Cardinality Feedback

Oracle Database 11g Release 2 ~

- オプティマイザが実行計画作成（ハード・パース）時に見積もったCardinality（行数）と実際にSQL文を実行した際のCardinalityの間に大きな乖離

→ 実際のCardinalityを記録

→ その後、同じSQL文の実行が要求された際、
記録済みCardinalityを使用して実行計画を再作成

  - ハード・パースによるCPUコストが追加されると解釈されがちだが、
不適切な実行計画でSQLを再実行するよりも、低コストに抑えることが可能

- オプティマイザの進化
  - Rule-Based Optimizer → Cost-Based Optimizer（統計情報からの最適化）
→ Feedback-Based Optimizer (実行結果からの最適化)

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

Adaptive Cursor Sharing

Oracle Database 11g Release 1 ~

- 一つのSQL文においても、バインド変数の範囲毎に複数の実行計画を保持
  - 正確には・・・
    - バインド変数を含むSQL文をモニタリングし、バインド変数値に応じて複数の子カーソルを保持するExtended Cursor Sharing機能(Oracle Database 10g Release 2~:一部適用)が必要か否か判断し適用する機能
      - SQL文が複数回実行される場合、実行計画が適切かを自動的に判断
      - 生成される子カーソルの数を最小限に抑える(カーソルのマージ機能)

→ バインド変数の先読み機能(Bind Peek)問題を解消する機能

- 詳細は、KROWN#127064 [11g新機能] 優れたカーソル共有

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

SQL Plan Management(SQL計画管理/SPM)

Oracle Database 11g Release 1 ~

- 実行計画を自動で管理し、既知の計画又は検証済みの計画だけを使用
  - ハード・パース時に新しい実行計画が生成された場合でも、DBAが評価/承認が完了するまでは現在の実行計画を使用し続けることで、突然の性能劣化を防止
    - SQL実行が要求される度ではない。従来通りキャッシュ済みの実行計画を使用する(ソフト・パース)際は、SPMによる追加コストは発生しない
    - SQL個別に新しい実行計画を記録するか否かを制御可能
  - 初期化パラメータ
    - OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES
      - TRUE(デフォルト:FALSE) → SQL計画ベースラインの自動取得を開始
    - OPTIMIZER_USE_SQL_PLAN_BASELINES
      - TRUE(デフォルト:TRUE) → SQL計画ベースライン上の実行計画を使用

Copyright© 2014, Oracle and/or its affiliates. All rights reserved.

Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®