Oracle **Database** Technology Night
～集え！オラクルの力（チカラ）～

Technical Discussion Night

- オラクル・コンサルが語る！-
SQL性能を最大限に引き出す
DB 12c クエリー・オプティマイザ
新機能活用 と 統計情報運用の戦略

ORACLE DATABASE 12c

Plug into the Cloud

日本オラクル株式会社
コンサルティングサービス事業統括
クラウド・テクノロジーコンサルティング事業本部
プリンシパルコンサルタント 畔勝 洋平
プリンシパルコンサルタント 柴田 歩

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。

自己紹介(柴田 歩)

- 日本オラクル株式会社
  クラウド・テクノロジーコンサルティング事業本部
  DBソリューション
  プリンシパルコンサルタント
  柴田　歩(しばた あゆむ)
- シバタツ(柴田 竜典)さん(※2017年8月ご卒業)、
  しばちょう(柴田 長)さんに続く、第3の柴田！
- 2007年4月に中途で日本オラクルに入社
- DBの製品コンサルとして、DB関連のプロジェクトを歴任

自己紹介代わりのコンテンツ類(柴田)

- ブログ「ねら～ITエンジニア雑記」
- http://d.hatena.ne.jp/gonsuke777/

- DDD 2013 SQLチューニングに必要な考え方と最新テクニック
http://www.oracle.com/technetwork/jp/ondemand/ddd-2013-2051348-ja.html

- Oracle DDD 2017
- オラクル・コンサルが語る！- SQL性能を最大限に引き出す
DB 12c クエリー・オプティマイザ 新機能活用 と 統計情報運用の戦略
http://www.oracle.com/technetwork/jp/ondemand/ddd-2017-3373953-ja.html

- SQLチューニングと対戦格闘ゲームの類似性について語る。- JPOUG Advent Calendar 2017 Day 15 -
- http://d.hatena.ne.jp/gonsuke777/20171215/1481795088

- Bind Peek をもっと使おうぜ！
-JPOUG Advent Calendar 2014-
- http://d.hatena.ne.jp/gonsuke777/20141205/1417710300

- JPOUG Tech Talk Night #6
「固定化か？最新化か？オプティマイザ統計の運用をもう一度考える。」
- http://d.hatena.ne.jp/gonsuke777/20170226/1456488499

- まだ統計固定で消耗してるの？
-JPOUG Advent Calendar 2015-
- http://d.hatena.ne.jp/gonsuke777/20151208/1449587953

自己紹介（畔勝）

- 畔勝 洋平（あぜかつ ようへい）
- 2010年中途入社（6年目）
  ネットベンチャー→フリーランス→ドワンゴ→日本オラクル
- Webデザイナー（HTML・JSコーダー）としてキャリアをスタート
- DBコンサルとしてミッションクリティカルシステムを支援
- トラブルシューティングではOSカーネル（Linux/商用UNIXなど）に Deep Dive することも

自己紹介（畔勝）

Twitter: @yoheia

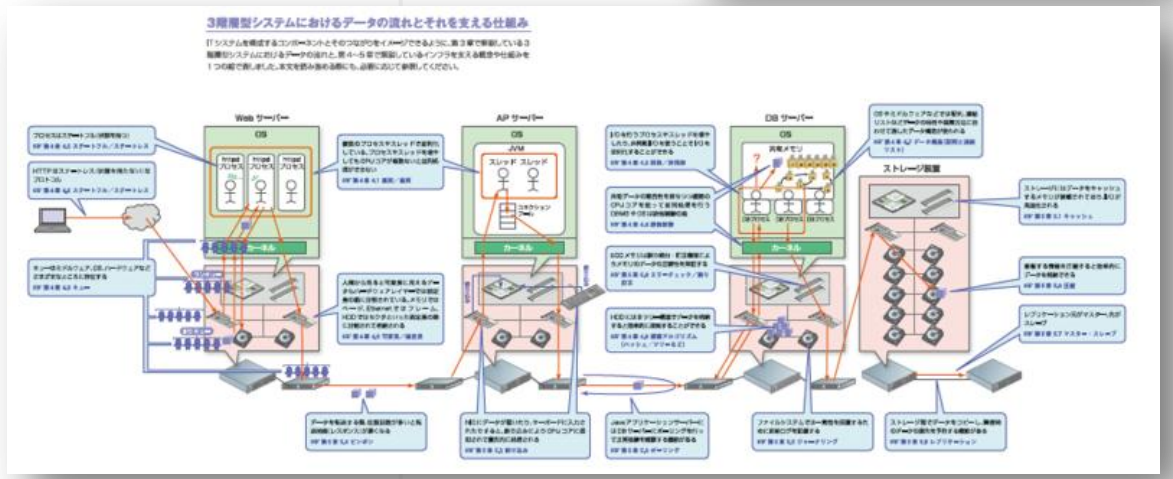
Blog: http://d.hatena.ne.jp/yohei-a/

著書（共著）：絵で見てわかるITインフラの仕組み

JPOUG(Japan Oracle User Group)の運営に関わってます

2013年ジュンク堂池袋本店コンピュータ書
売上冊数ランキング第5位
http://compbook.g.hatena.ne.jp/compbook/20140107/p1

カバーを裏返すとシステムの全貌がわかる解剖図



本セミナーの目的

- SQLと云う言語の特徴と、SQL性能を改善するための原理／原則を理解すること
- Oracle Database 12c の 各種クエリー・オプティマイザ機能の概要をざっくりと把握すること
- 様々なシステムにおける、統計情報運用／クエリー・オプティマイザ機能の適切な組み合わせを、考える切っ掛けになること

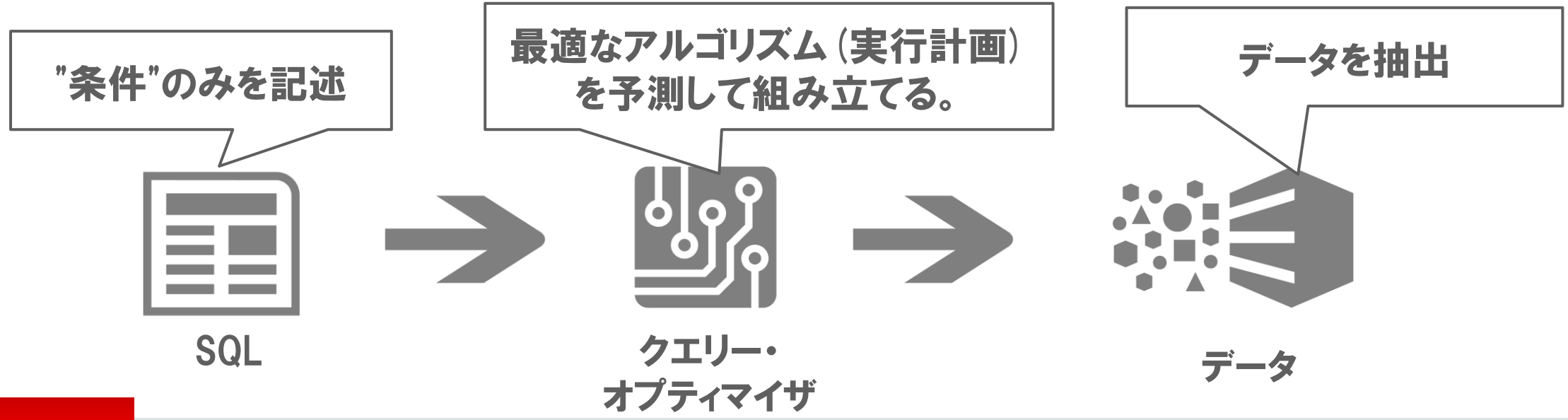
本日の内容

1. 前提知識
   (各種クエリーオプティマイザ機能のご紹介)
2. クエリーオプティマイザ機能 と 統計情報運用 の
   組み合わせ戦略
3. 統計情報運用のデザイン と
   事例紹介(良いパターン／アンチパターン)
4. まとめ
5. Appendix. 関連情報

1章. 前提知識
(各種クエリー・オプティマイザ機能のご紹介)

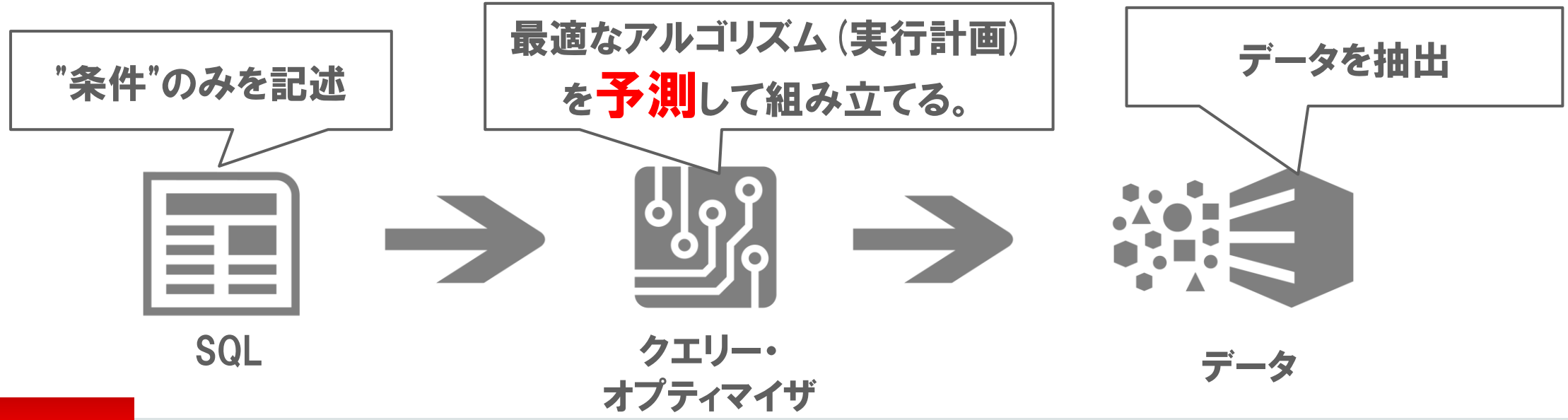
そもそも クエリー・オプティマイザ とは？

- SQL と云う言語は、アルゴリズムを書かずにデータ抽出の条件のみを書けば良いという特徴があります。
- 効率の良い アルゴリズム＝実行計画 を予測して組み立てるのが、クエリー・オプティマイザ の 役割です。



そもそも クエリー・オプティマイザ とは？

- SQL と云う言語は、アルゴリズムを書かずにデータ抽出の条件のみを書けば良いという特徴があります。
- 効率の良い アルゴリズム＝実行計画 を予測して組み立てるのが、クエリー・オプティマイザ の 役割です。

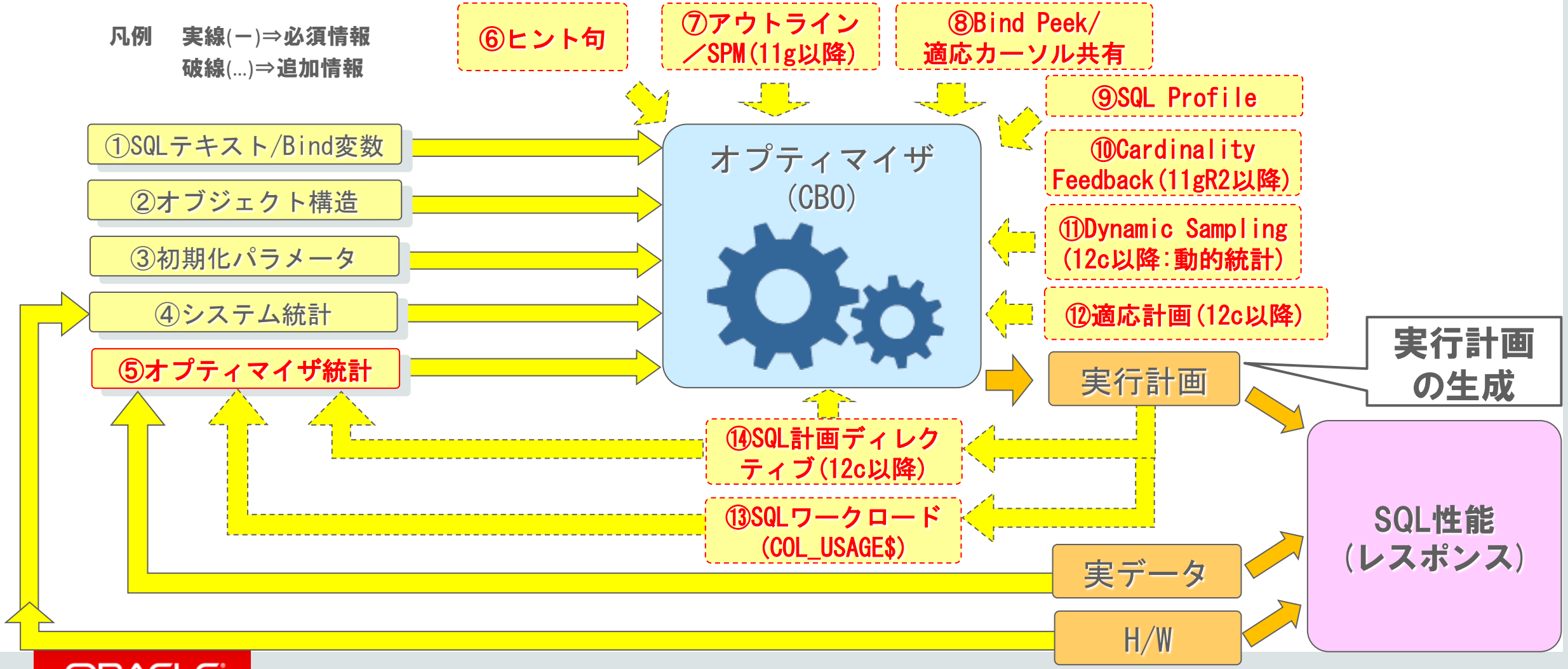


キーワード

「予測」

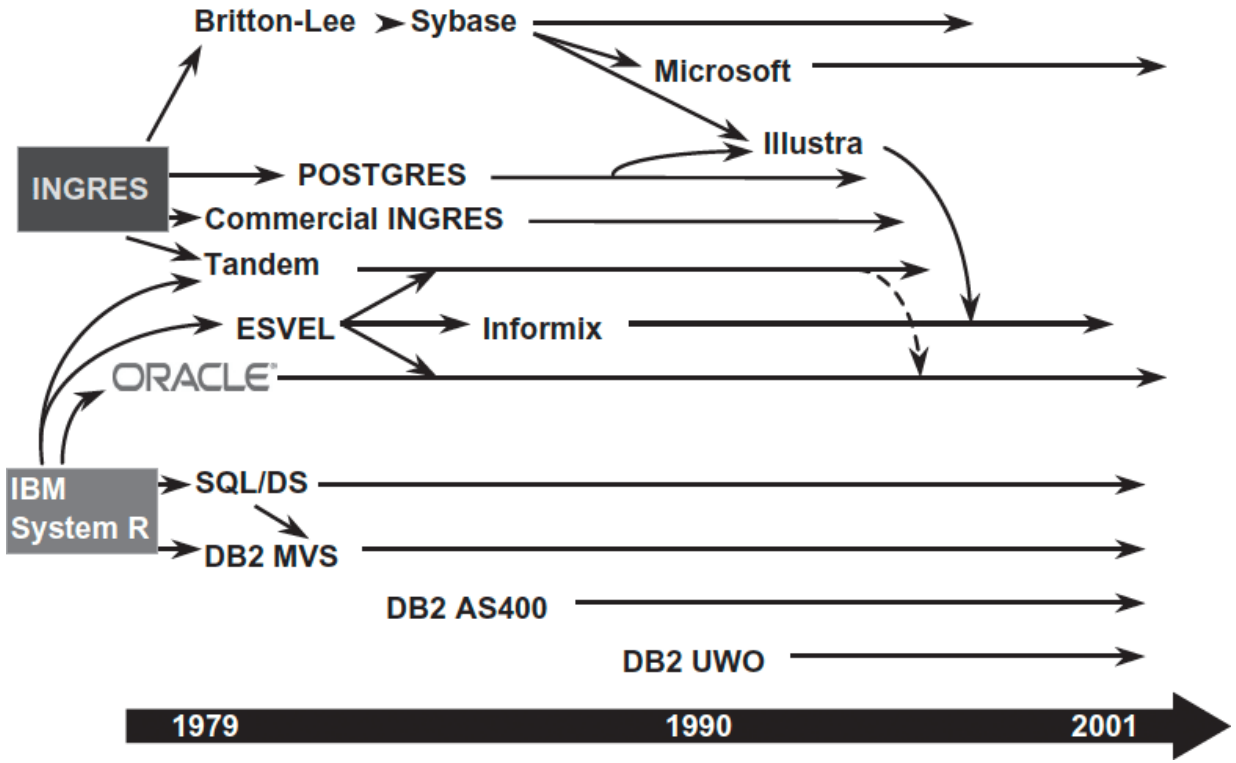
Oracle Database 12cR1 における 実行計画生成の全体像

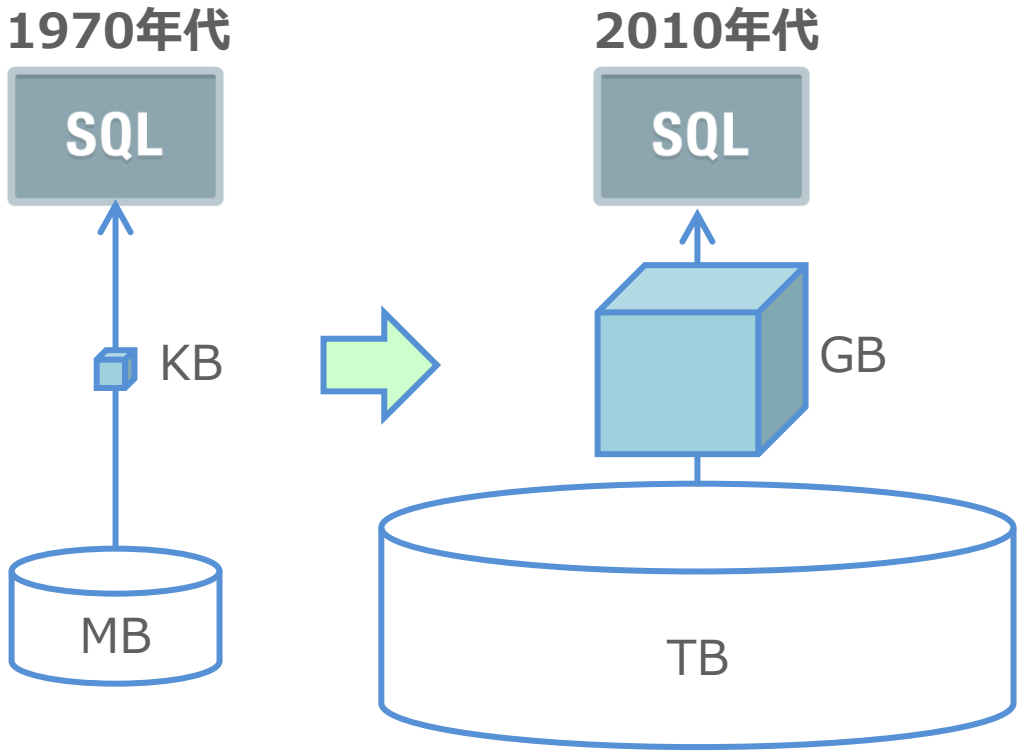
- 予測精度向上のための、様々なクエリー・オプティマイザ機能をご紹介



[背景]SQLがアクセスするデータサイズはKBからGBに

1970年代にRDBMS登場後、データベースのサイズはMBからTBに、
SQLが1回の実行でアクセスするデータサイズはKBからGBに

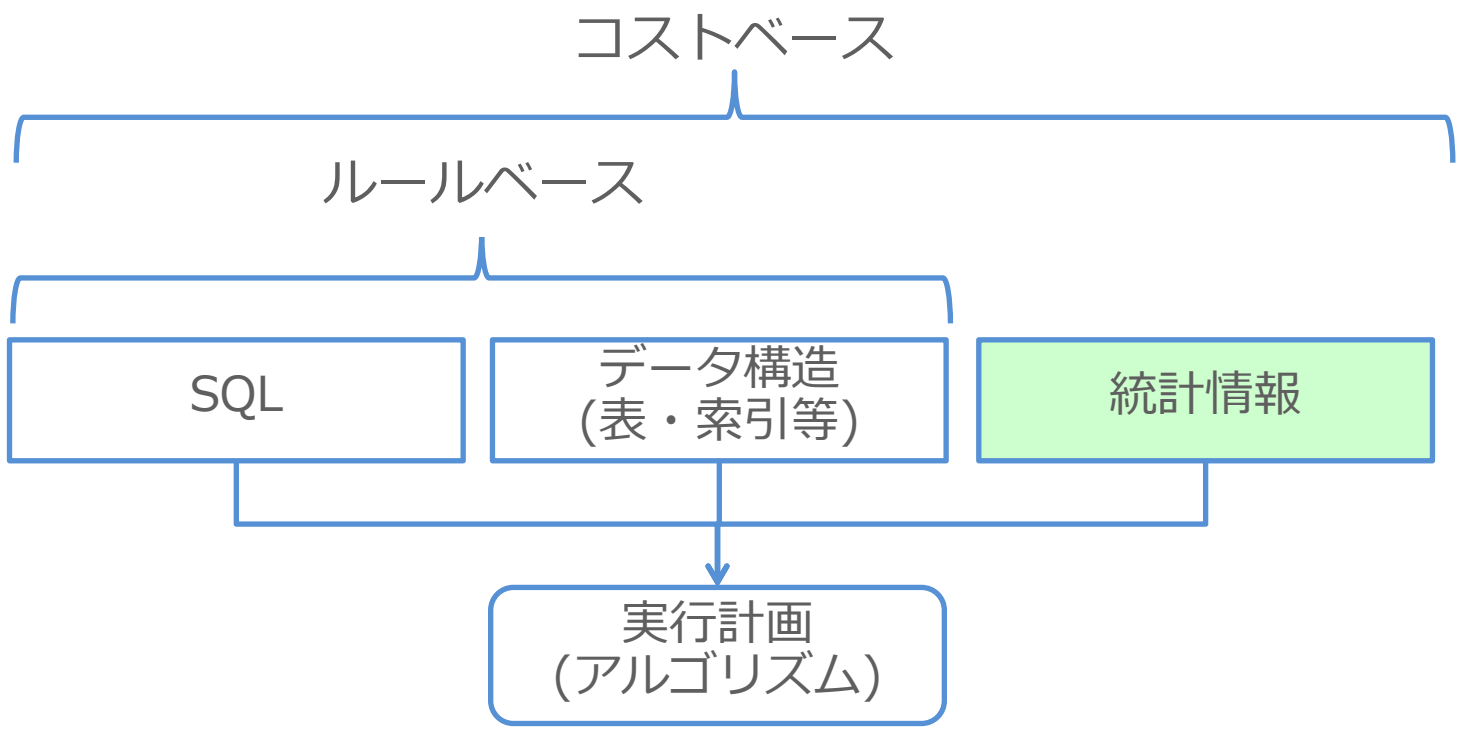




https://www.computer.org/csdl/mags/an/2013/02/man2013020010-abs.html

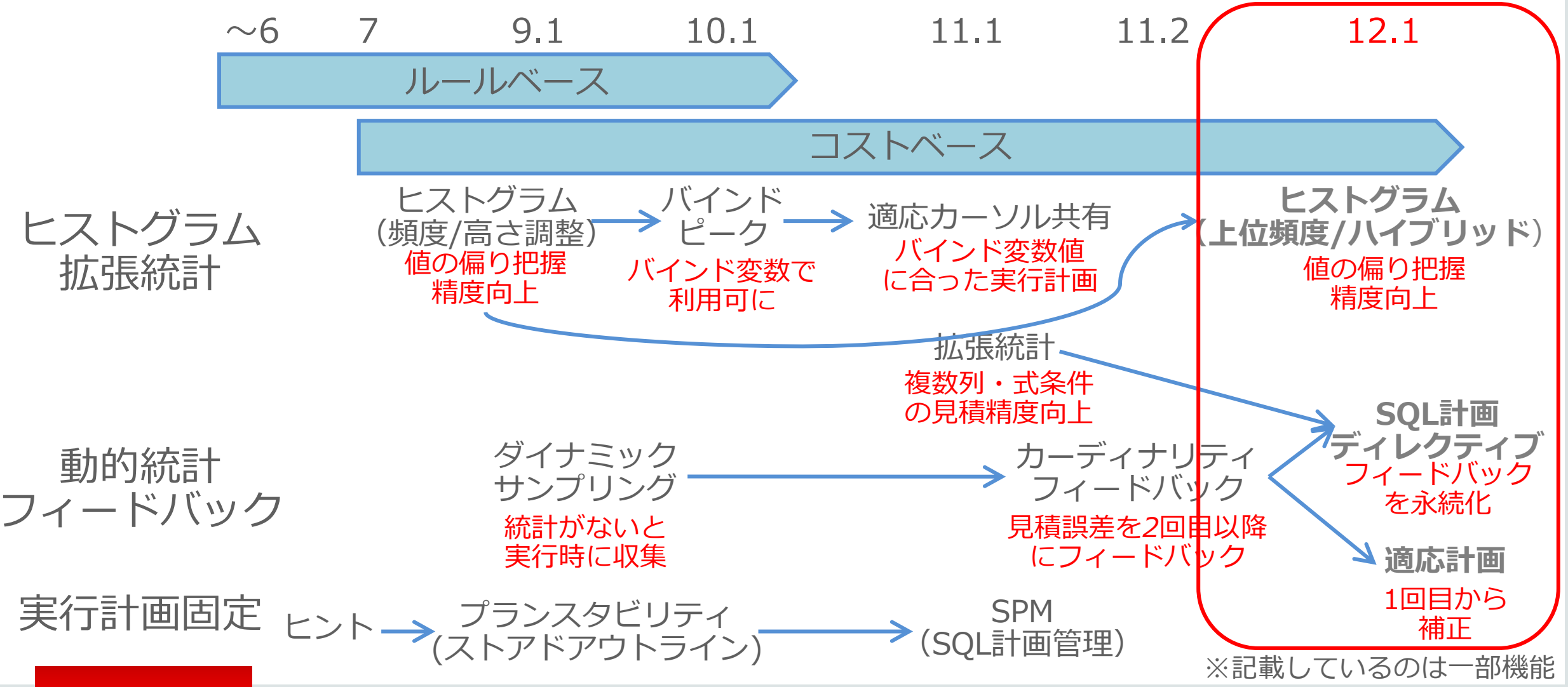
ルールベースからコストベースへ

1990年代前半に大量データ処理を睨み、Oracle Database のクエリー・オプティマイザはルールベースからコストベースへ



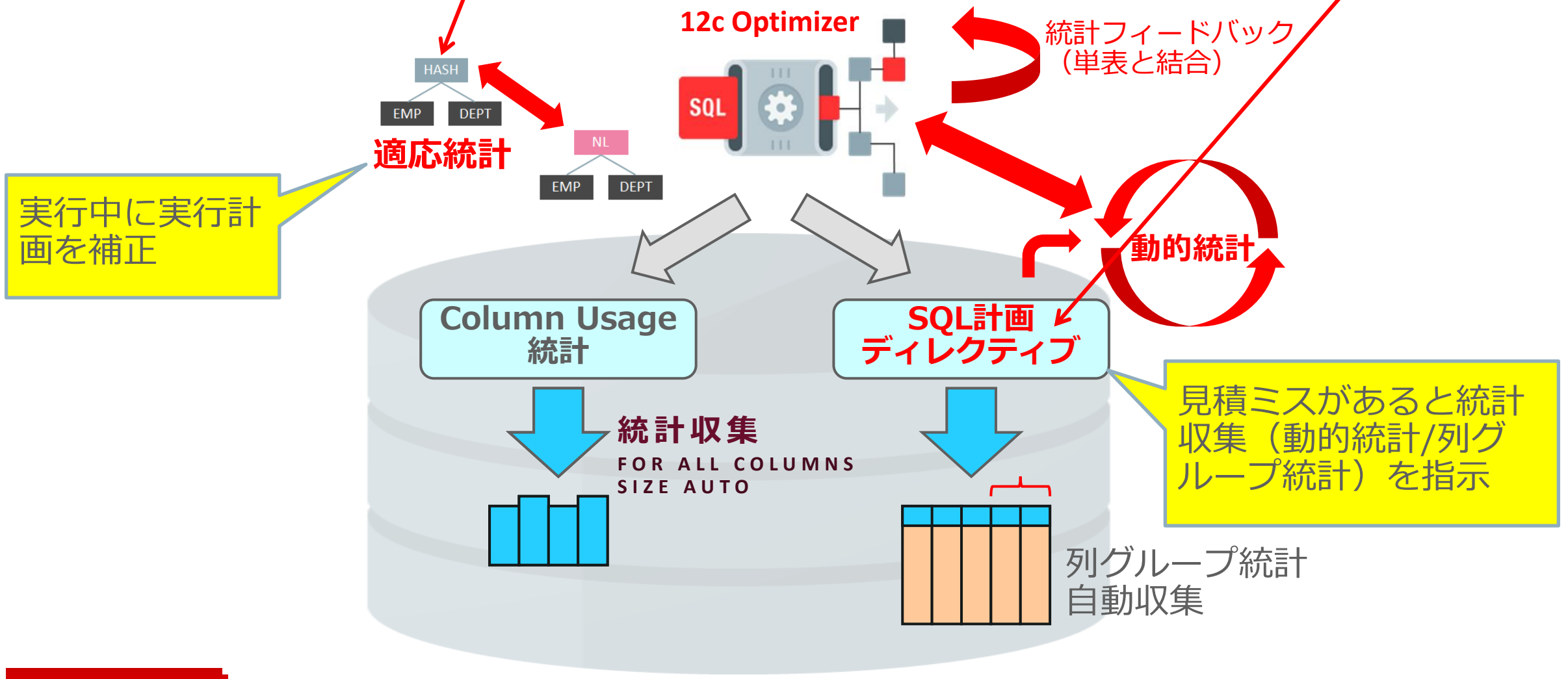
- コストベース
  - System-R(1974～)
  - Ingres(1974～)
  - DB2(1983～)
  - Sybase(1984～)
  - SQL Server(1989～)
  - MySQL(1995～)
  - PostgreSQL(1986～)
  - Oracle(1992～)
    Version 7～
- ルールベース
  - Oracle(1979～)
    Version 2～10.2

コストベースオプティマイザ（CBO） の進化



※記載しているのは一部機能

12.1の「適応計画」と「SQL計画ディレクティブ」

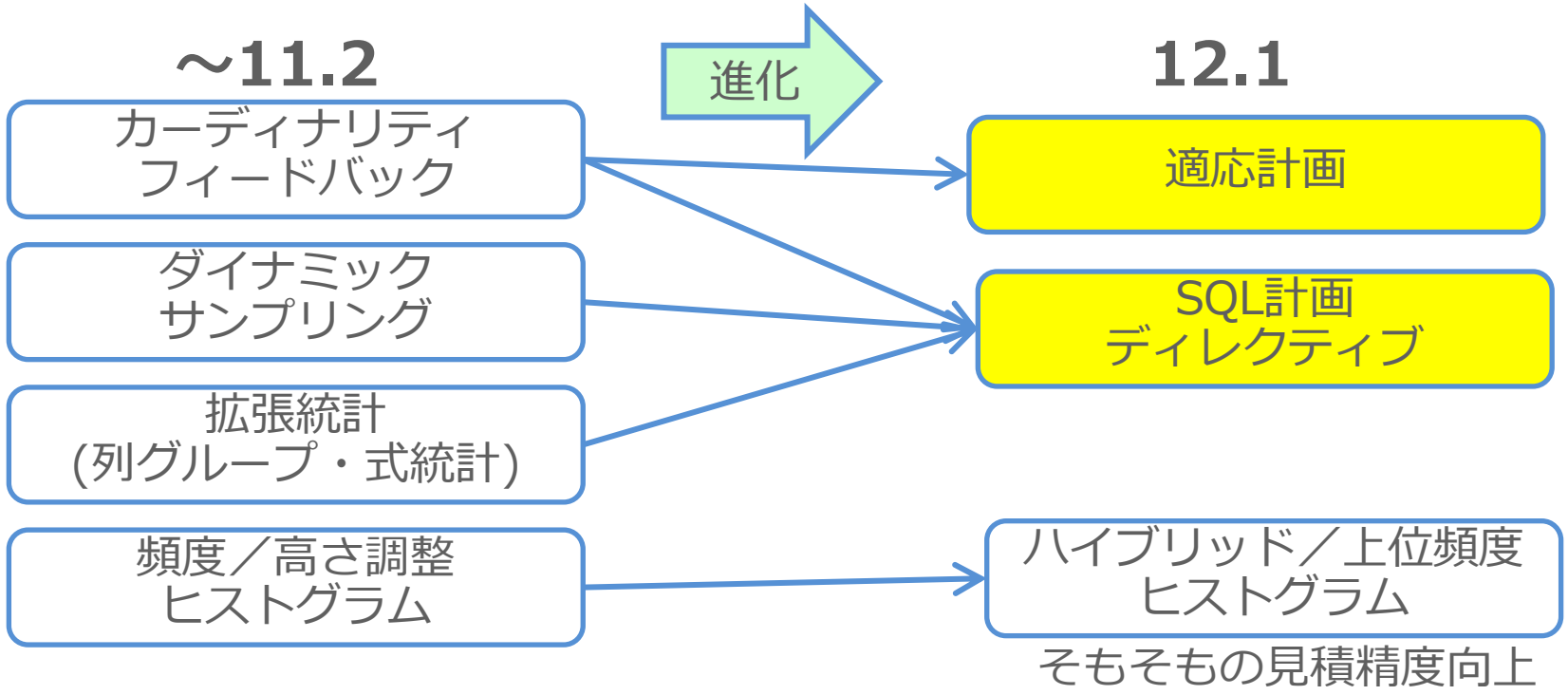


12.1 で CBO のミッシングピースが埋まってきた？

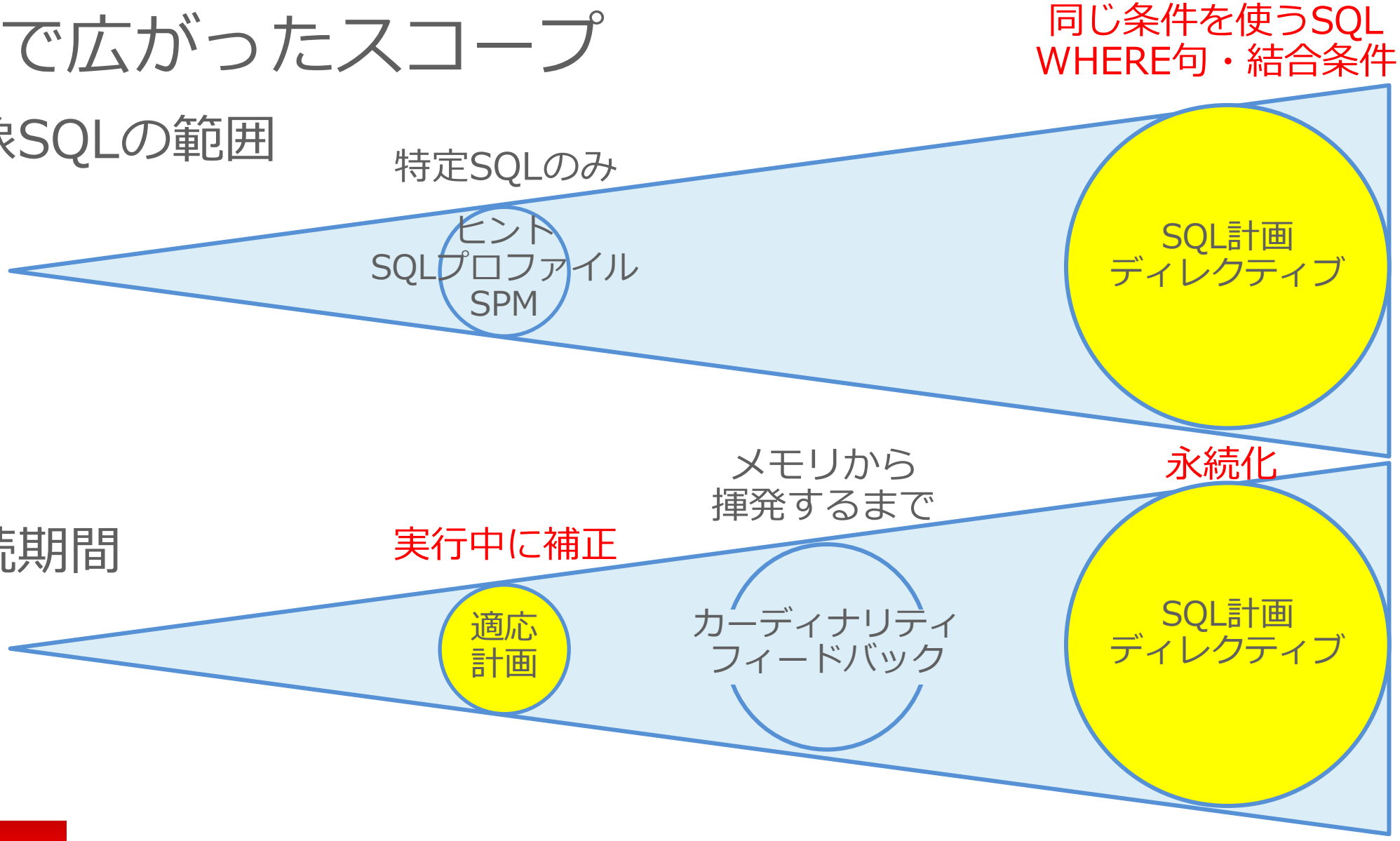
課題：カーディナリティフィードバックは2回目から補正、永続化されない

→適応計画：実行中に実行計画補正

→ SQL計画ディレクティブ：揮発せずに永続化



12.1 で広がったスコープ

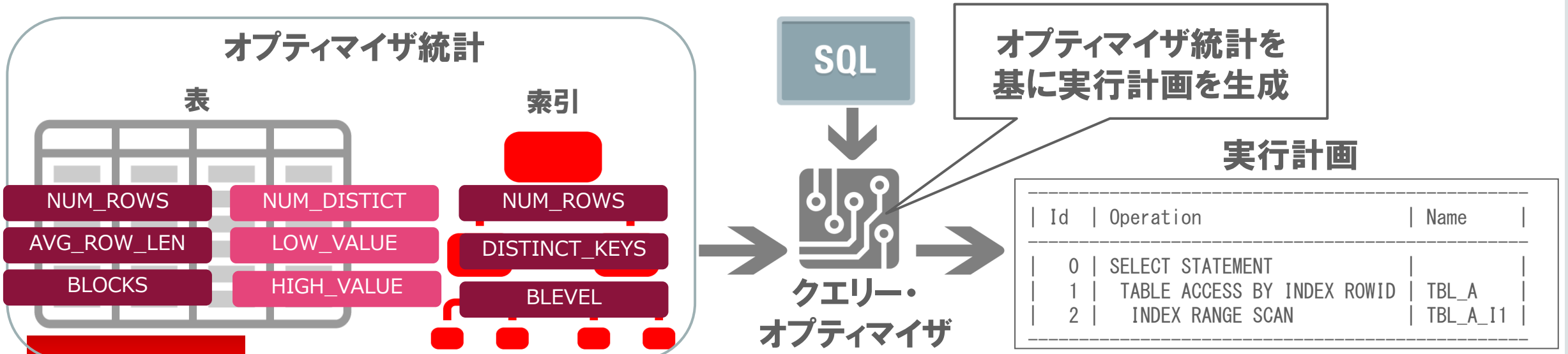


Oracle Database の クエリー・オプティマイザ機能

- オプティマイザ統計(CBO導入時)
- ヒストグラム／拡張統計(11g)
- SQLワークロード(col_usage$,10g／col_group_usage$,11g)
- Bind Peek(9i)
- 適応カーソル共有(11g)
- Dynamic Sampling(9i)／Dynamic Statistics(動的統計・12c)
- Statistics(Cardinarity) Feedback(11g)
- SQL計画ディレクティブ(12c)
- Adaptive Plan(適応計画・12c)
- その他、各種機能……SQL Profile(10g), SPM(11g), etc...

オプティマイザ統計の役割

- オプティマイザ統計は、オプティマイザが適切な実行計画を予測する為に必要となる、基礎的な情報となります。
  - 表の件数、平均レコード長、ブロック数、etc...
  - 索引の列値の種類、件数、Bツリーの高さ、etc...
  - 列統計(列値の種類、LOW_VALUE、HIGH_VALUE、etc...)



ヒストグラムの概要

- ヒストグラムは、列内のデータ配分が均一ではない場合にオプティマイザの予測を補正するための追加の情報です。
  - 以下のように列Xの値に偏りがある場合、ヒストグラムを採取するとカーディナリティ(取得される行数)の予測が正確になります。

列A(PK)	…	列X(FLG列)	…
1	…	0	…
2	…	0	…
3	…	0	…
:		:	
99	…	0	…
100	…	1	…

大半が"0"で
ごく一部が"1"

拡張統計(複数列統計／式統計)の概要

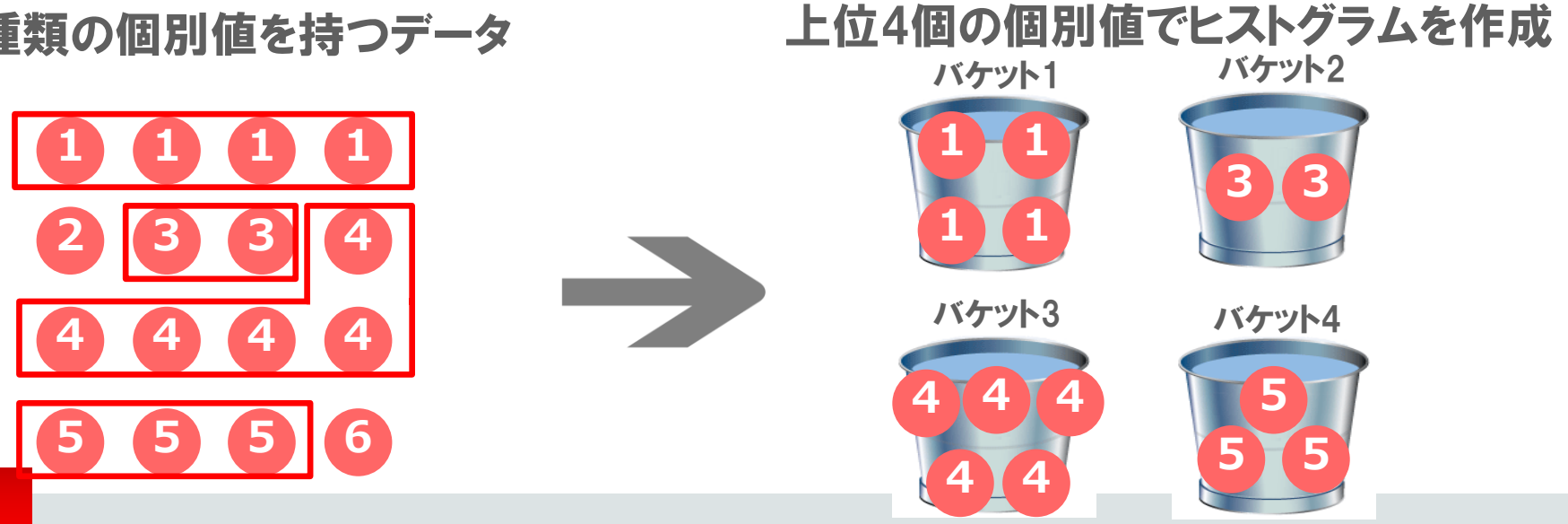
- 拡張統計はヒストグラムを機能拡張したもので、複数列(列グループ)統計と式統計の2種類があります。
  - 以下のように列同士の値に相関があるケースで、複数列統計を採取すると、カーディナリティの予測が正確になります。

列A(PK)	…	列X	列Y	…
1	…	0 →	A	…
2	…	0 →	A	
3		1 →	B	
4	…	1 →	B	…
:	…	…	…	…
100	…	25 →	Z	

列X が "0" の時に 列Y は "A"、
列X が "1" の時に 列Y は "B"、
……というように、
列同士の値に相関があるケース

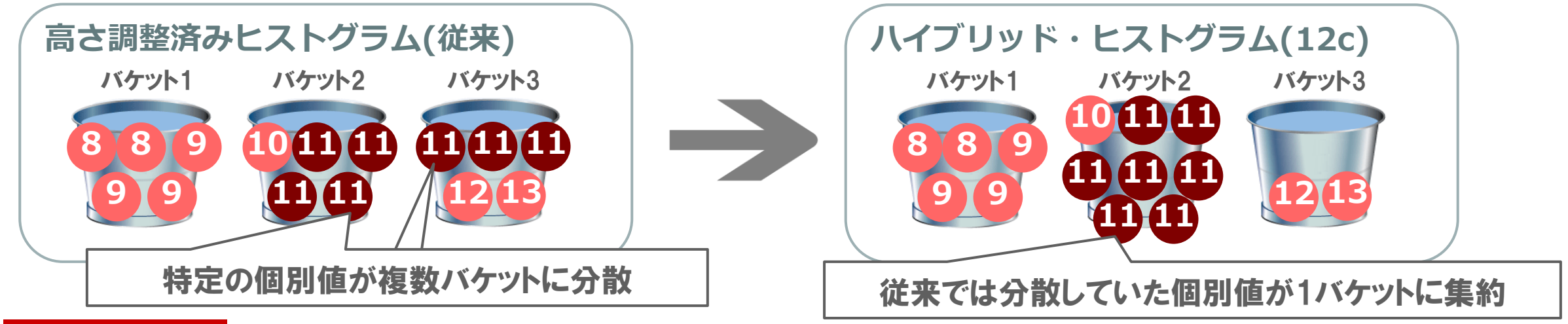
12c新機能：上位頻度ヒストグラム

- 上位頻度ヒストグラムは、個別値の種類がバケット数(※ヒストグラムを格納する内部領域)より少ないケースで、一部の個別値がデータの大部分を占める場合に作成されます。
  - 上位n個の個別値に対するカーディナリティの予測精度が向上します。
  - 下記は上位頻度ヒストグラムの作成例となります。



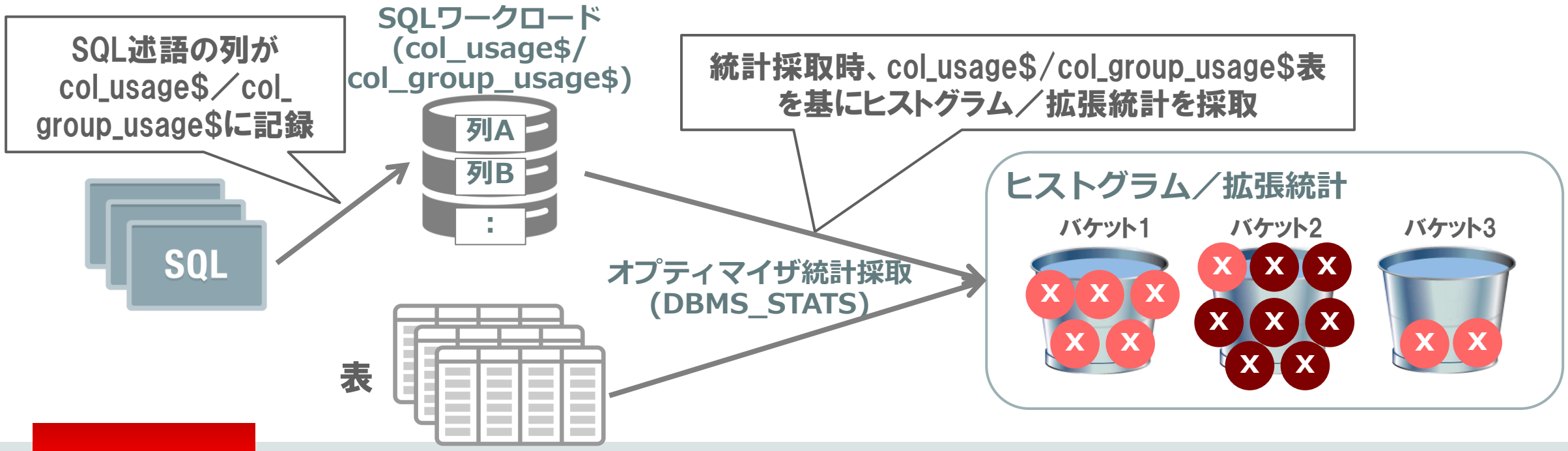
12c新機能：ハイブリッド・ヒストグラム

- 上位頻度ヒストグラムが作成できないケースで、ハイブリッド・ヒストグラムが作成されます。
  - 従来の高さ調整済みヒストグラムでは複数バケットに分散していた個別値が、1バケットに集約されるように調整されます。
  - そのような個別値に対するカーディナリティの予測精度が向上します。



SQLワークロード(col_usage$/col_group_usage$)の概要

- SQLの述語(WHERE句の条件)で使われた列／列グループをマークして、col_usage$/col_group_usage$表に格納する機能です。
  - SQLワークロード(col～usage$)にマークされた列／列グループは、統計採取時に参照されて、それらのヒストグラム／拡張統計が採取されます。



SQLワークロード(col_usage$/col_group_usage$)の出力例

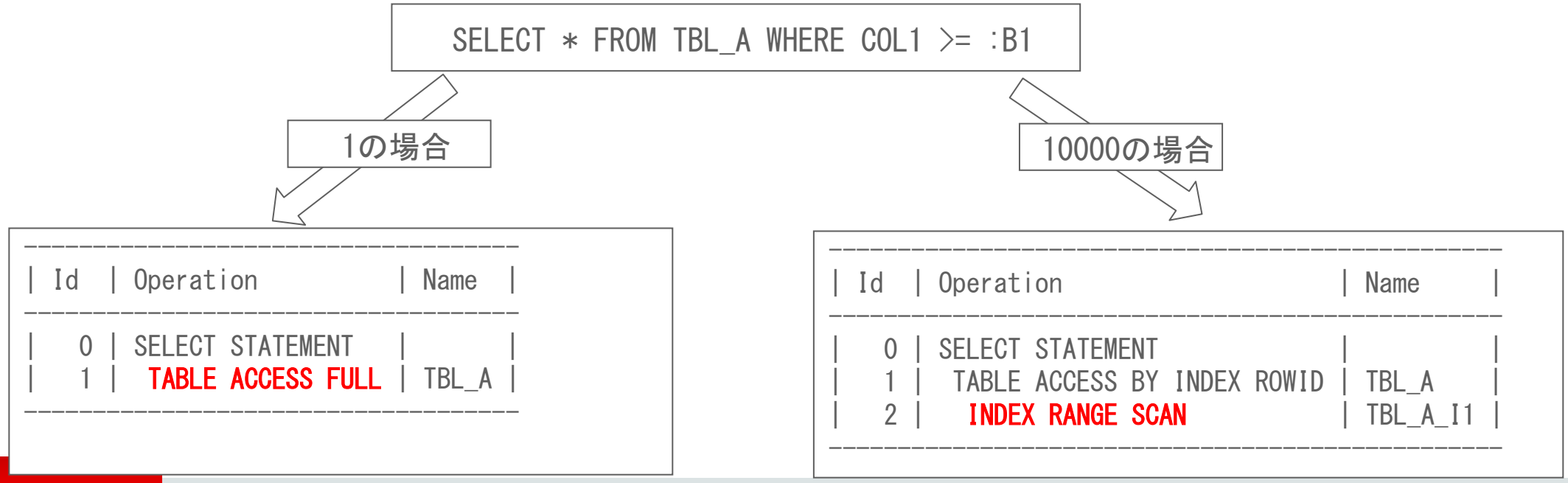
- SQLワークロード(col_usage$/col_group_usage$)でマークされた列は、DBMS_STATS.REPORT_COL_USAGEで参照可能です。

```
SET LONG 1000000;
SET LONGC 1000000;
SET LINESIZE 1000;
SET PAGESIZE 0;
SELECT DBMS_STATS.REPORT_COL_USAGE(NULL, NULL) FROM DUAL;
 :
################################################################################

COLUMN USAGE REPORT FOR STDBYPERF.STATS$FILE_HISTOGRAM
.....................................................

1. DB_UNIQUE_NAME                          : EQ EQ_JOIN
2. FILE#                                   : EQ EQ_JOIN
3. INSTANCE_NAME                           : EQ EQ_JOIN
4. SINGLEBLKRDTIM_MILLI                    : EQ_JOIN
5. SNAP_ID                                 : EQ
################################################################################
 :
```

SQLの述語で使われた列や、等価条件や結合条件等の情報が出力される。

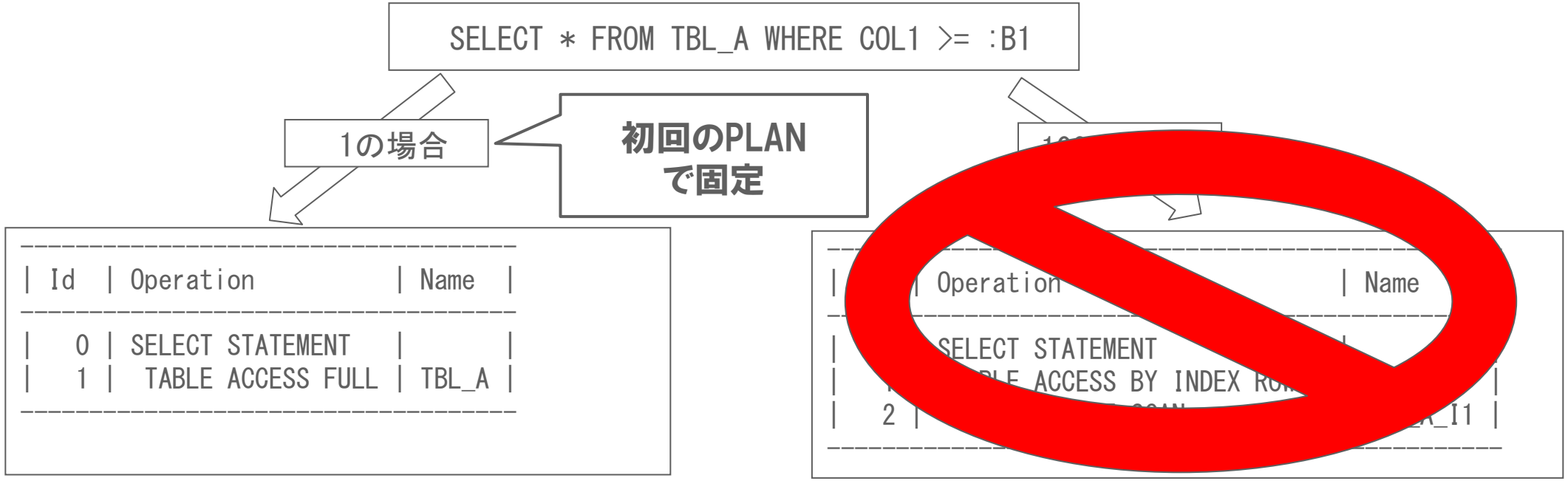
Bind Peekの概要

- バインド変数が使用されているSQLで、与えられたバインド変数値によって実行計画を使い分ける(最適化する)機能
  - Bind Peek を有効にするとバインド変数使用時に 列統計/ヒストグラム/拡張統計が使用されるため、実行計画の予測精度が向上します。



10gR2までの Bind Peekの動作は...

- 10gR2までは、初回ハード・パース時のバインド変数値によって作成された実行計画で固定されてしまいます。

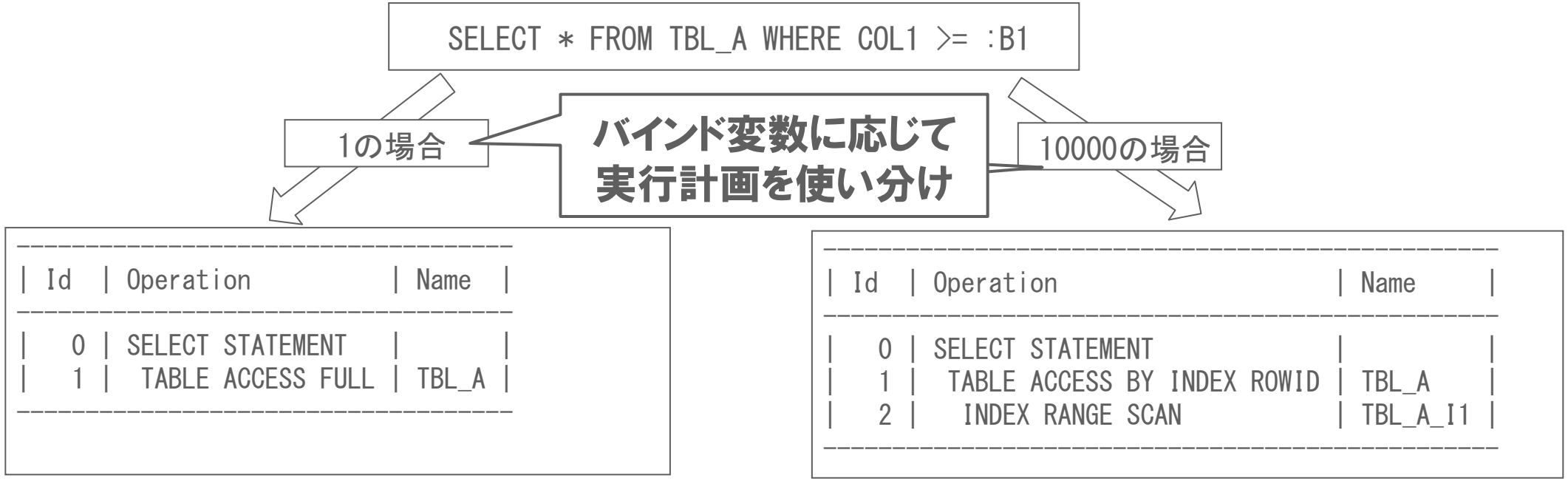


このような動作をしていたため、10gR2 までは
Bind Peek を無効化するシステムが多かった。

※列統計／ヒストグラム／拡張統計を見なくなるので、実行計画の予測精度は低下する。

11gR1で「適応カーソル共有」が導入

- 11gR1からは「適応カーソル共有(Adaptive Cursor Sharing)」が導入されて、複数の実行計画を併用するようになりました。

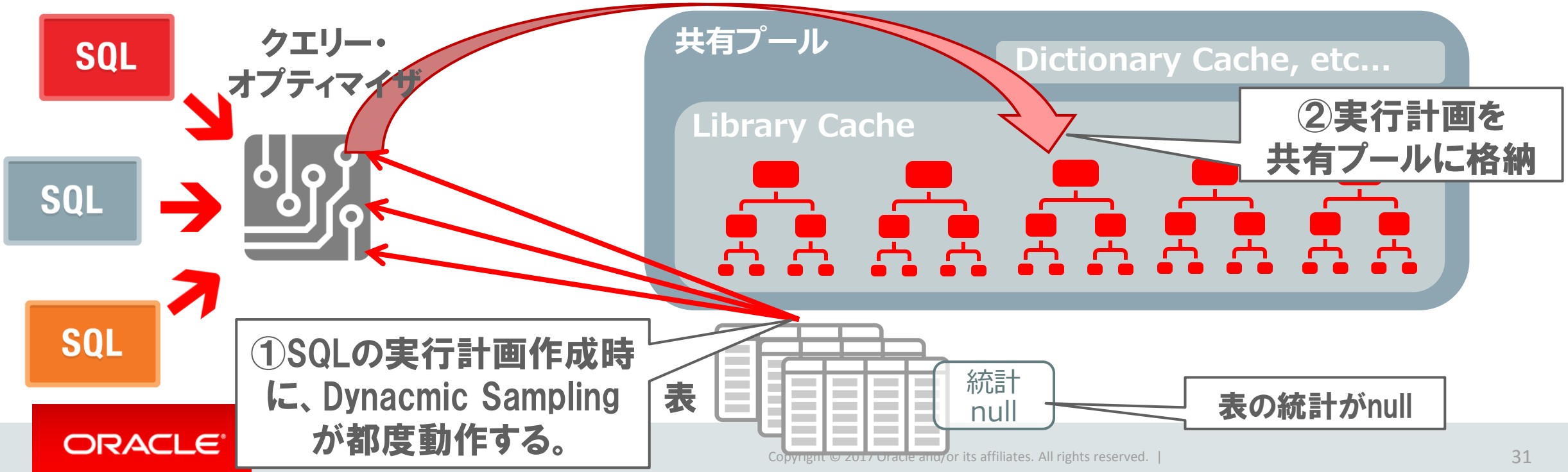


10gR2までの欠点は、11gR1以降は概ね解消されている。

※Bind Peek を無効化すると、適応カーソル共有も無効化されます。

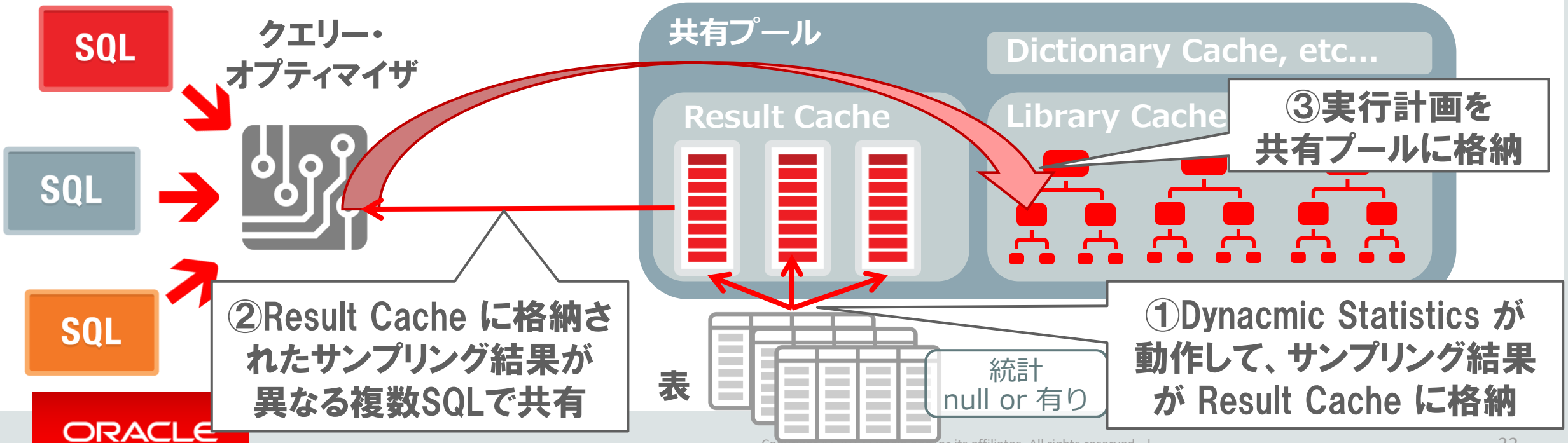
Dynamic Samplingの概要

- 表や索引のオプティマイザ統計 が null の場合に、簡易的な統計を動的に採取(Dynamic Sampling)して、実行計画の予測精度を向上させる機能です。
  - 簡易的な統計のため、結果の精度は通常の統計よりは低くなります。
  - 本機能によるサンプリング結果は、異なるSQLでは共有されません。



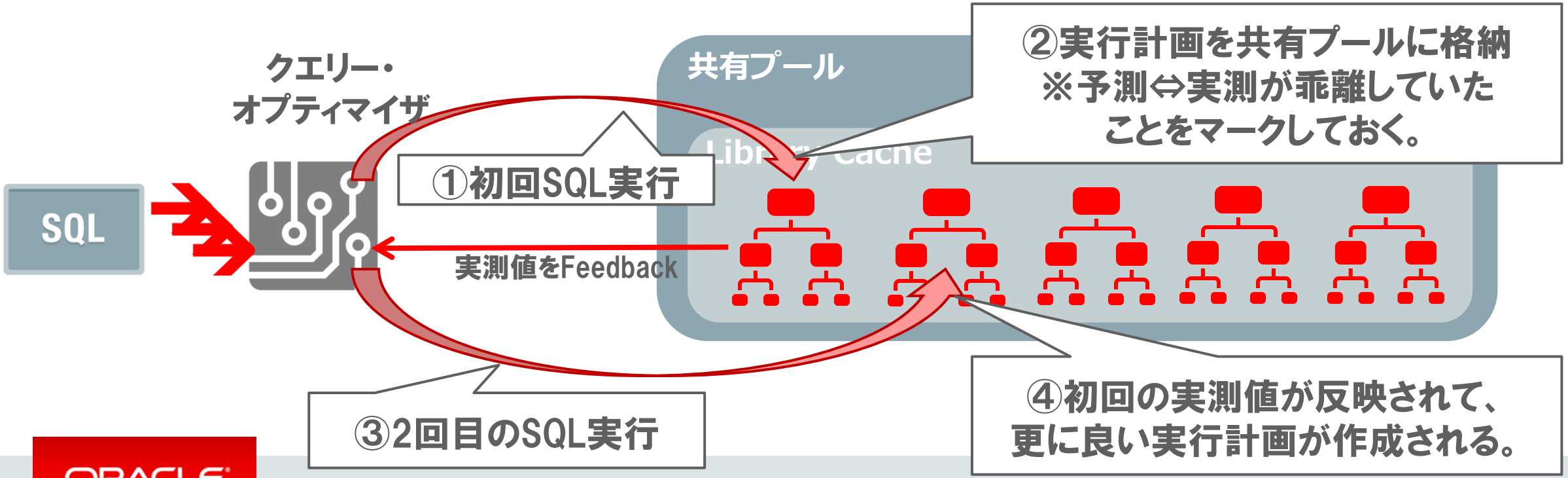
12c新機能：Dynamic Statistics(動的統計)

- Dynamic Sampling は 12c で機能強化されて、Dynamic Statistics(動的統計) と云う名前になりました。
  - OPTIMIZER_DYNAMIC_SAMPLING を 11 にセットすると、サンプリング結果が Result Cache に格納されて、複数SQLで共有されるようになります。
  - 更にSQL計画ディレクティブ(後述)との組み合わせでも、動作します。



Statistics(Cardinality)Feedbackの概要

- 初回SQL実行時の予測と実測が乖離しているケースで、2回目のSQL実行時に初回実測値をFeedbackする機能です。
  - 初回の実測値で予測の乖離を補正して、実行計画を最適化します。



Statistics(Cardinality)Feedbackによる性能改善例

- 下記はStatistics Feedbackによる性能改善例となります。

初回SQL実行時

```
10:48:08 SQL> SELECT /*+ MONITOR */
10:48:08   2          A.*
10:48:08   3    FROM TEST_TABLE_A A
10:48:08   4       , TBL_B B
10:48:08   5   WHERE A.P_NO2    = B.P_NO
10:48:08   6     AND A.P_CHAR = B.P_CHAR
10:48:08   7     AND TO_CHAR(B.P_DATE, 'YYYYMMDD') = '20120801';
 :
 :
 :
 :
 :
1102 rows selected.

Elapsed: 00:00:04.71
 :
Statistics
----------------------------------------------------------

      8994  consistent gets
        59  physical reads
```

2回目のSQL実行時

```
21:12:15 SQL> SELECT /*+ MONITOR */
21:12:15   2          A.*
21:12:15   3    FROM TEST_TABLE_A A
21:12:15   4       , TBL_B B
21:12:15   5   WHERE A.P_NO2    = B.P_NO
21:12:15   6     AND A.P_CHAR = B.P_CHAR
21:12:15   7     AND TO_CHAR(B.P_DATE, 'YYYYMMDD') = '20120801';
 :
Note
-----
   - statistics feedback used for this statement
 :
1102 rows selected.
 :
Elapsed: 00:00:00.14
 :
Statistics
----------------------------------------------------------

      1301  consistent gets
         1  physical reads
```

Statistics (Cardinality) Feedback が動作

仕事量 (consistent gets) が減って性能改善！

Statistics(Cardinality)Feedback動作前後のSQL監視

- **Before**
  初回SQL
  実行時

```
=============================================================================
| Id |      Operation        |     Name     | Rows    | Rows     | Activity Detail |
|    |                       |              | (Estim) | (Actual) |   (# samples)   |
=============================================================================
|  0 | SELECT STATEMENT      |              |         |     1102 |                 |
|  1 |   HASH JOIN           |              |   30012 |     1102 | Cpu  (5)        |
|  2 |    TABLE ACCESS FULL  | TBL_B        |     300 |       11 |                 |
|  3 |    TABLE ACCESS FULL  | TEST_TABLE_A |      3M |       3M |                 |
=============================================================================
```

結合の駆動表（外部表）の実測（Actual・11件）と予測（Estim・300件）がズレている。

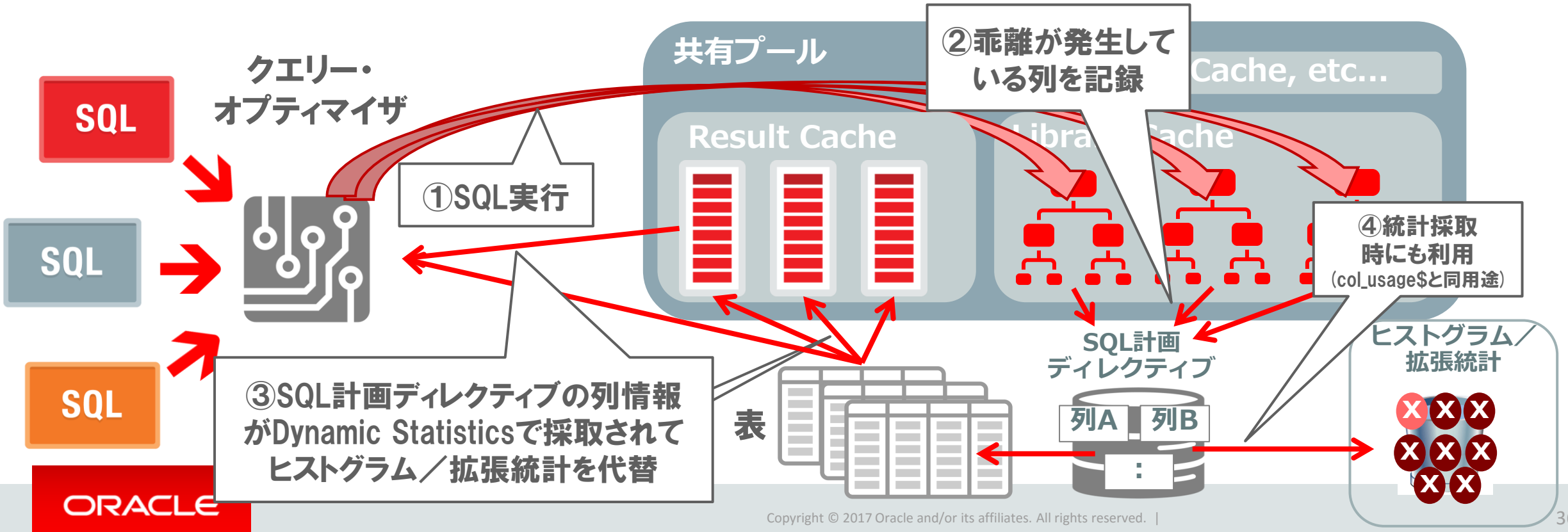
- **After**
  2回目の
  SQL
  実行時

```
=============================================================================================
| Id |          Operation          |      Name      | Rows    | Rows     | Activity Detail |
|    |                             |                | (Estim) | (Actual) |   (# samples)   |
=============================================================================================
|  0 | SELECT STATEMENT            |                |         |     1102 |                 |
|  1 |   NESTED LOOPS              |                |         |     1102 |                 |
|  2 |    NESTED LOOPS             |                |    1106 |     1102 |                 |
|  3 |     TABLE ACCESS FULL       | TBL_B          |      11 |       11 |                 |
|  4 |     INDEX RANGE SCAN        | TEST_TABLE_A_IY |     100 |     1102 |                 |
|  5 |    TABLE ACCESS BY INDEX ROWID | TEST_TABLE_A |     101 |     1102 |                 |
=============================================================================================
```

予測（Estim）と 実測（Actual）の差が無くなり、適切な実行計画が選択されている。

12c新機能：SQL計画ディレクティブの概要

- SQL実行時の予測と実測が乖離しているケースで、乖離が発生している列／列グループの情報を記録する機能です。
  - 記録された列情報はSQL実行時と統計採取時の両方で利用されます。



SQL計画ディレクティブの内容を参照

- DBA_SQL_PLAN_DIRECTIVES／DBA_SQL_PLAN_DIR_OBJECTS の両ディクショナリから、SQL計画ディレクティブの内容を参照できます。
  - 乖離(MISESTIMATE)が発生した表名／列名や理由が記録されています。

```
SELECT DIRECTIVE_ID, REASON FROM DBA_SQL_PLAN_DIRECTIVES ORDER BY DIRECTIVE_ID;

          DIRECTIVE_ID REASON
---------------------- ----------------------------------
:
   17285060759984228 65 GROUP BY CARDINALITY MISESTIMATE
   17594243734095478 47 JOIN CARDINALITY MISESTIMATE
   18673680948264460 21 SINGLE TABLE CARDINALITY MISESTIMATE
:

SELECT DIRECTIVE_ID, OWNER, OBJECT_NAME, SUBOBJECT_NAME FROM DBA_SQL_PLAN_DIR_OBJECTS
WHERE OWNER = 'AYSHIBAT' ORDER BY DIRECTIVE_ID;

          DIRECTIVE_ID OWNER      OBJECT_NAME     SUBOBJECT_NAME
---------------------- ---------- --------------- --------------------
   50736901807991617 71 AYSHIBAT   SALES_DETAIL
:
  11717631835078839377 AYSHIBAT   SALES_DETAIL    RECEIPT_NUM
  16570908913880212990 AYSHIBAT   SALES           SALES_DATE
:
```

乖離 (MISESTIMATE) の理由

乖離 (MISESTIMATE) が発生している表名／列名

SQL計画ディレクティブによるSQL性能改善例

- 下記はSQL計画ディレクティブによる性能改善例です。
  - SQL計画ディレクティブからヒストグラム／拡張統計が不足している列／列グループ を判断して、Dynamic Statistics(動的統計)で補っています。

SQL計画ディレクティブ無効時

```
09:21:12 SQL> SELECT /*+ MONITOR */
09:21:12   2         DTL.*
09:21:12   3    FROM SALES        SAL
09:21:12   4       , SALES_DETAIL DTL
09:21:12   5   WHERE SAL.RECEIPT_NUM = DTL.RECEIPT_NUM
09:21:12   6     AND TO_CHAR(SAL.SALES_DATE, 'YYYYMMDD')
09:21:12   7         = '20151101';
:
:
:
:
:
:
:
:
:
統計
----------------------------------------------------------
:
      4301  consistent gets
         0  physical reads
```

SQL計画ディレクティブ有効時

```
09:22:36 SQL> SELECT /*+ MONITOR */
09:22:36   2         DTL.*
09:22:36   3    FROM SALES        SAL
09:22:36   4       , SALES_DETAIL DTL
09:22:36   5   WHERE SAL.RECEIPT_NUM = DTL.RECEIPT_NUM
09:22:36   6     AND TO_CHAR(SAL.SALES_DATE, 'YYYYMMDD')
09:22:36   7         = '20151101';
:
:
Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - 1 Sql Plan Directive used for this statement
:
:
統計
----------------------------------------------------------
:
        58  consistent gets
         0  physical reads
```

SQL計画ディレクティブと動的統計が同時に動作

仕事量(consistent gets)が大幅減少して性能改善！

SQL計画ディレクティブ動作前後の実行計画

- **Before**
  SQL計画ディレクティブ無効時

```
| Id | Operation                    | Name         | Rows (Estim) | Rows (Actual) | Activity Detail (# samples) |
|  0 | SELECT STATEMENT             |              |              |             1 |                             |
|  1 |   NESTED LOOPS               |              |            1 |             1 | Cpu (2)                     |
|  2 |    NESTED LOOPS              |              |            1 |          870K | Cpu (12)                    |
|  3 |     TABLE ACCESS FULL        | SALES_DETAIL |            1 |          870K |                             |
|  4 |     INDEX UNIQUE SCAN        | SALES_PK     |            1 |          870K |                             |
|  5 |    TABLE ACCESS BY INDEX ROWID | SALES      |            1 |             1 |                             |
```

結合の駆動表（外部表）の予測（Estim・1件）と実測（Actual・870,000件）が大幅にズレている。

- **After**
  SQL計画ディレクティブ有効時

```
| Id | Operation                    | Name         | Rows (Estim) | Rows (Actual) | Activity Detail (# samples) |
|  0 | SELECT STATEMENT             |              |              |             1 |                             |
|  1 |   NESTED LOOPS               |              |            1 |             1 |                             |
|  2 |    NESTED LOOPS              |              |            1 |             1 |                             |
|  3 |     TABLE ACCESS FULL        | SALES        |            1 |             1 |                             |
|  4 |     INDEX RANGE SCAN         | SALES_DETAIL_I |            1 |             1 |                             |
|  5 |    TABLE ACCESS BY INDEX ROWID | SALES_DETAIL |            1 |             1 |                             |
```

予測（Estim）と 実測（Actual）の差が無くなり、適切な実行計画が選択されている。

12c新機能：適応計画(Adaptive Plan)

- SQL実行時の予測と実測が乖離しているケースで、実行計画を予め用意されたサブプランに"動的"に切り替える機能
  - NESTED LOOPS ⇒ HASH JOIN に切り替えるケース
  - PQ Distribute(パラレル配分方法)を切り替えるケース

```
-----------------------------------------------------
| Id  | Operation                    | Name         |
-----------------------------------------------------
|   0 | SELECT STATEMENT             |              |
|   1 |  NESTED LOOPS                |              |
|   2 |   NESTED LOOPS               |              |
|   3 |    TABLE ACCESS FULL         | SALES_DETAIL |
|*  4 |    INDEX UNIQUE SCAN         | SALES_PK     |
|*  5 |   TABLE ACCESS BY INDEX ROWID| SALES        |
-----------------------------------------------------
```

実行時に動的に切替

```
-----------------------------------------------------
| Id | Operation                    | Name         |
-----------------------------------------------------
|  0 | SELECT STATEMENT             |              |
|  1 |  HASH JOIN                   |              |
|  2 |   TABLE ACCESS FULL          | SALES_DETAIL |
|  3 |   TABLE ACCESS FULL          | SALES        |
-----------------------------------------------------
```

適応計画(Adaptive Plan)による性能改善例

- 適応計画(Adaptive Plan)により、SQL性能が改善

適応計画無効時

```
06:51:22 SQL> SELECT /*+ MONITOR */
06:51:22   2         DTL.*
06:51:22   3    FROM SALES        SAL
06:51:22   4       , SALES_DETAIL DTL
06:51:22   5   WHERE SAL.RECEIPT_NUM = DTL.RECEIPT_NUM
06:51:22   6     AND TO_CHAR(SAL.SALES_DATE, 'YYYYMMDD')
06:51:22   7         = '20151102';

870000行が選択されました。
:
:
:
:
:
:
:
統計
----------------------------------------------------------

 :
     178364  consistent gets
       1885  physical reads
 :
```

適応計画有効時

```
06:51:35 SQL> SELECT /*+ MONITOR */
06:51:35   2         DTL.*
06:51:35   3    FROM SALES        SAL
06:51:35   4       , SALES_DETAIL DTL
06:51:35   5   WHERE SAL.RECEIPT_NUM = DTL.RECEIPT_NUM
06:51:22   6     AND TO_CHAR(SAL.SALES_DATE, 'YYYYMMDD')
06:51:22   7         = '20151102';

870000行が選択されました。
:
Note
-----
   - this is an adaptive plan

統計
----------------------------------------------------------
:
     3879  consistent gets
     1962  physical reads
:
```

適応計画(Adaptive Plan)が有効

仕事量(consistent gets)が大幅減少して性能改善！

適応計画(Adaptive Plan) の サブプラン切替時の実行計画

- **Before**
  適応計画
  無効時

```
-------------------------------------------------------------------------------
| Id  | Operation                    | Name         | E-Rows | A-Rows |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |        |      1 |
|   1 |  NESTED LOOPS                |              |      1 |      1 |
|   2 |   NESTED LOOPS               |              |      1 |    870K|
|   3 |    TABLE ACCESS FULL         | SALES_DETAIL |      1 |    870K|
|*  4 |    INDEX UNIQUE SCAN         | SALES_PK     |      1 |    870K|
|*  5 |   TABLE ACCESS BY INDEX ROWID| SALES        |      1 |      1 |
-------------------------------------------------------------------------------
```

結合の駆動表（外部表）は予測（Estim）では1件だが、実測（Actual）では870,000件で、内部表に870,000回アクセスしている。

- **After**
  適応計画
  有効時

```
-------------------------------------------------------------------------------
|   Id  | Operation                    | Name         | E-Rows | A-Rows |
-------------------------------------------------------------------------------
|     0 | SELECT STATEMENT             |              |        |      1 |
|  *  1 |  HASH JOIN                   |              |      1 |      1 |
|-    2 |   NESTED LOOPS               |              |      1 |    870K|
|-    3 |    NESTED LOOPS              |              |      1 |    870K|
|-    4 |     STATISTICS COLLECTOR     |              |        |    870K|
|     5 |      TABLE ACCESS FULL       | SALES_DETAIL |      1 |    870K|
|- *  6 |     INDEX UNIQUE SCAN        | SALES_PK     |      1 |      0 |
|- *  7 |    TABLE ACCESS BY INDEX ROWID| SALES       |      1 |      0 |
|  *  8 |   TABLE ACCESS FULL          | SALES        |      1 |      1 |
-------------------------------------------------------------------------------

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

結合の駆動表（外部表）が 実測（Actual・870,000件）と予測（Estim・1件）で大幅にズレている。

NESTED LOOPS は動作していない。
('-' are inactive)

HASH JOIN が動作して、内部表に1回だけアクセス

適応計画発動で'-'部分は動作しない

本章(1章)のまとめ

- 本章では、次章以降の理解に必要となるクエリー・オプティマイザの機能について、その概要を解説しました。
- 次章ではオプティマイザ統計の運用と、本章で紹介したクエリー・オプティマイザ機能との組み合わせをご紹介したいと思います。

2章.クエリー・オプティマイザ機能 と 統計情報運用 の 組み合わせ戦略

オプティマイザ機能の前に「コスト」とは？

SQLチューニングでよくあるやり取り

ベンダA様

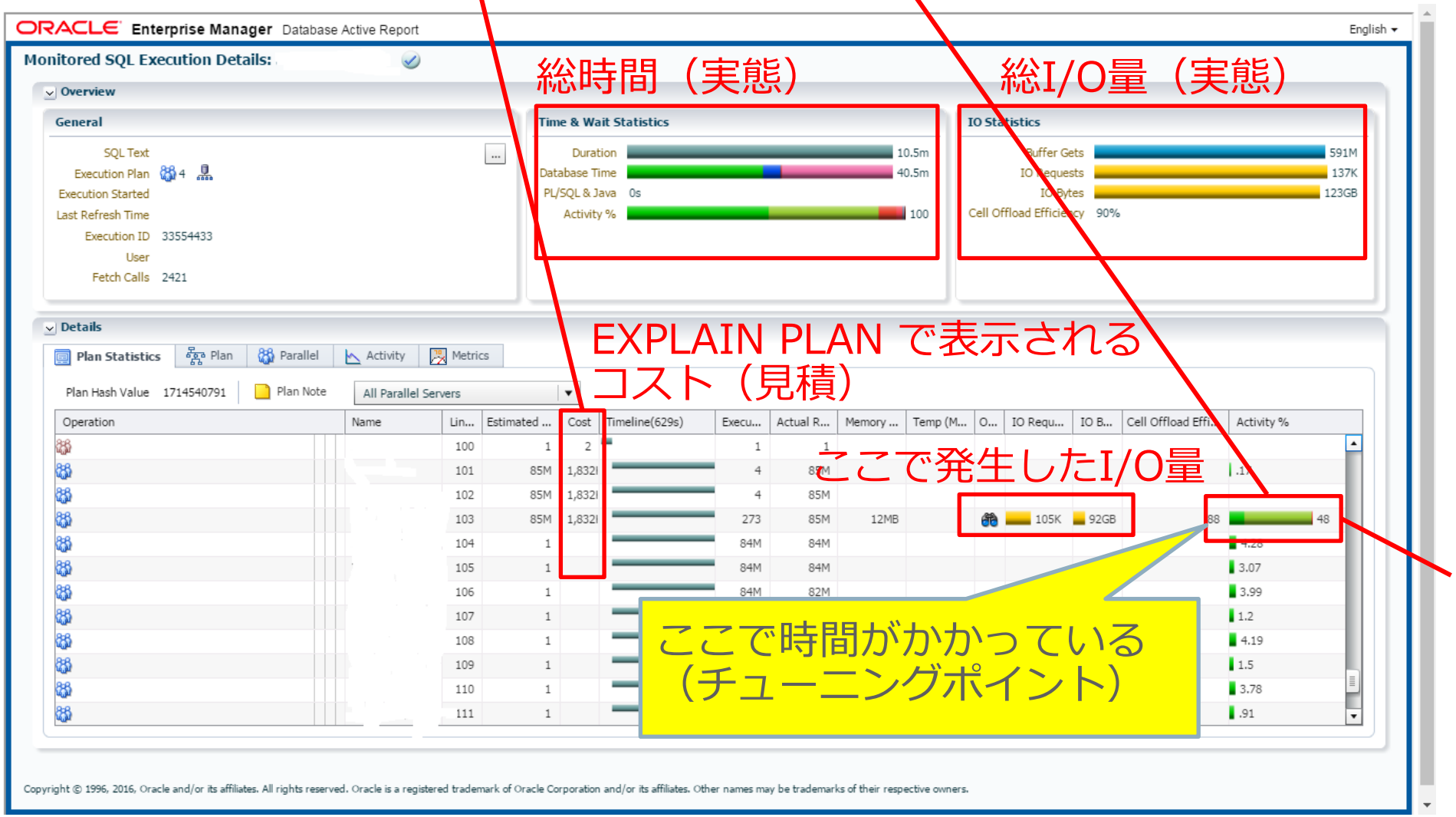
遅いSQL の EXPLAIN PLAN とったら、実行計画のこの行の「コスト」が高いのでチューニングが必要です。

そのコストは「見積」です。
「何に時間がかかったか」を見るのが、パフォーマンス分析・チューニングでは大切です。

私（畔勝）

コストは見積であり、時間のかかった箇所ではない

SQL監視で見積ではなく実態を見る

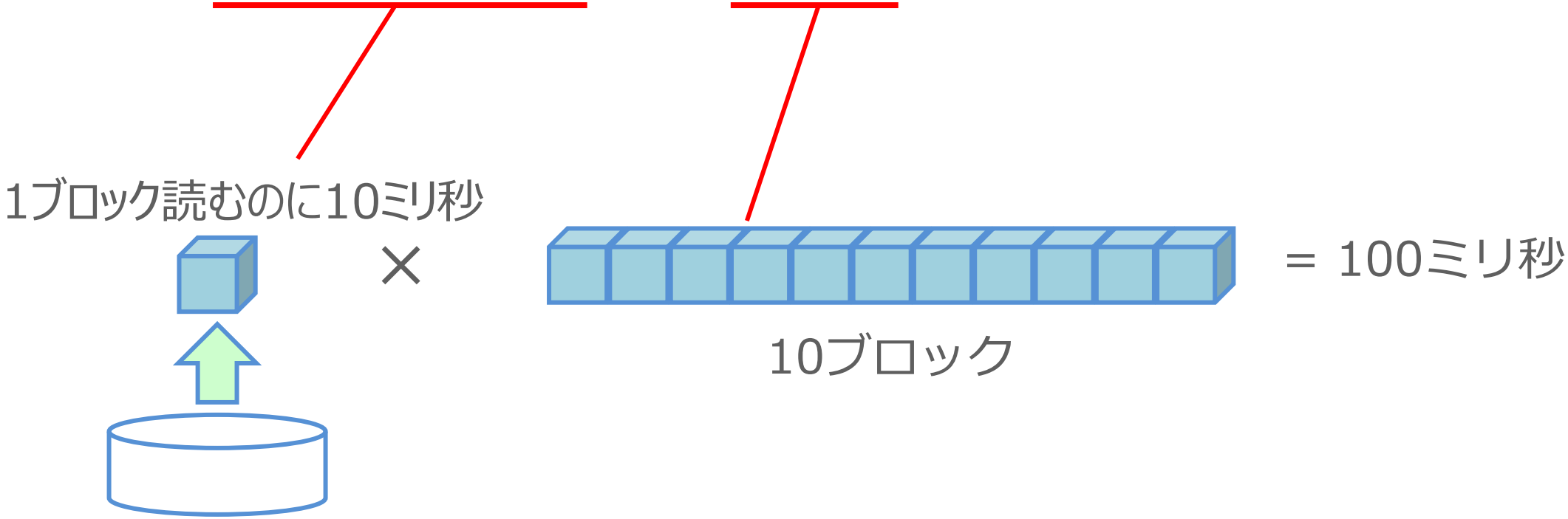


SQL監視は Enterprise Edition の Diagnostic&Tuning Pack のライセンスが必要。DBMS_XPLAN.DISPLAY_CURSORで代用可

コストとは CBO が予測した仕事量

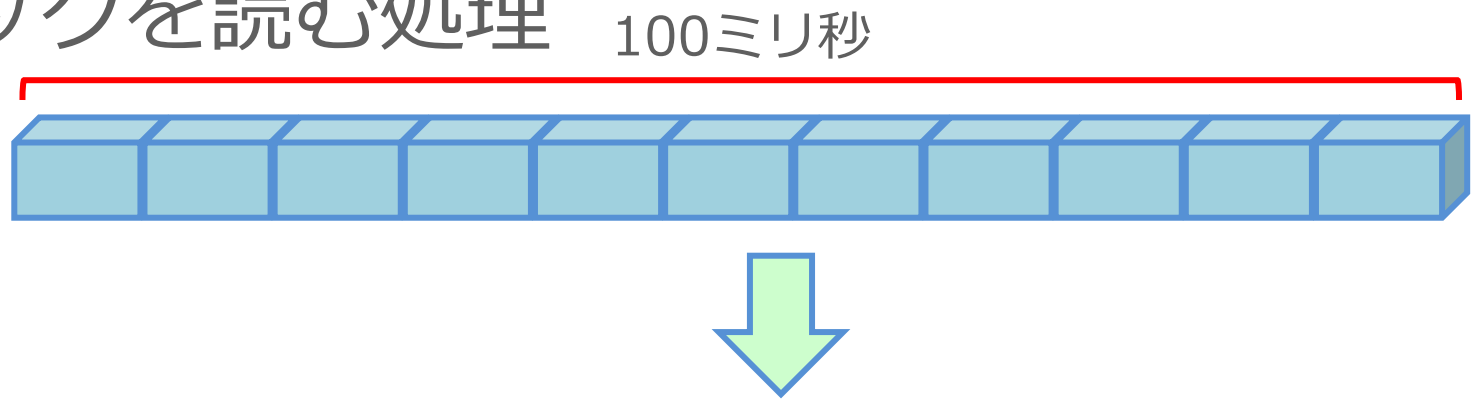
コストベースオプティマイザ（CBO）は仕事量が最小になると「予測」した実行計画を導出する

単価（時間）× 回数 = 仕事量（予測時間）

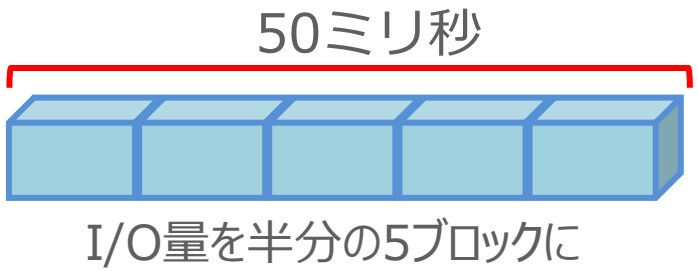


処理を速くする方法は３つ=CBOが考えるのもこの3つ

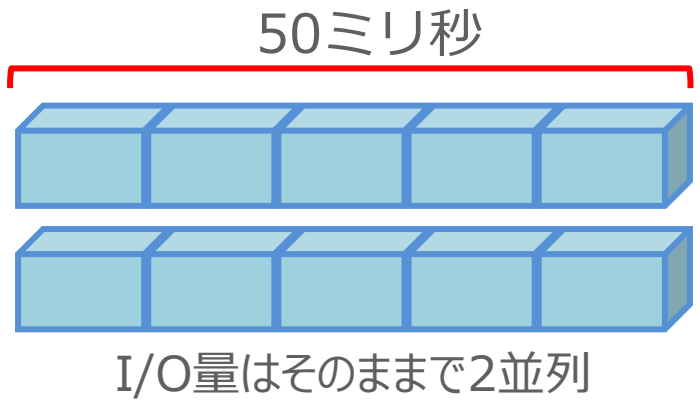
10ブロックを読む処理



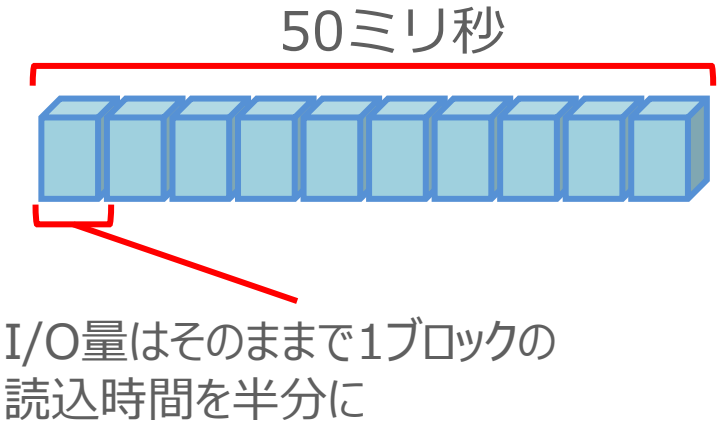
1）仕事量を減らす



2）並列化

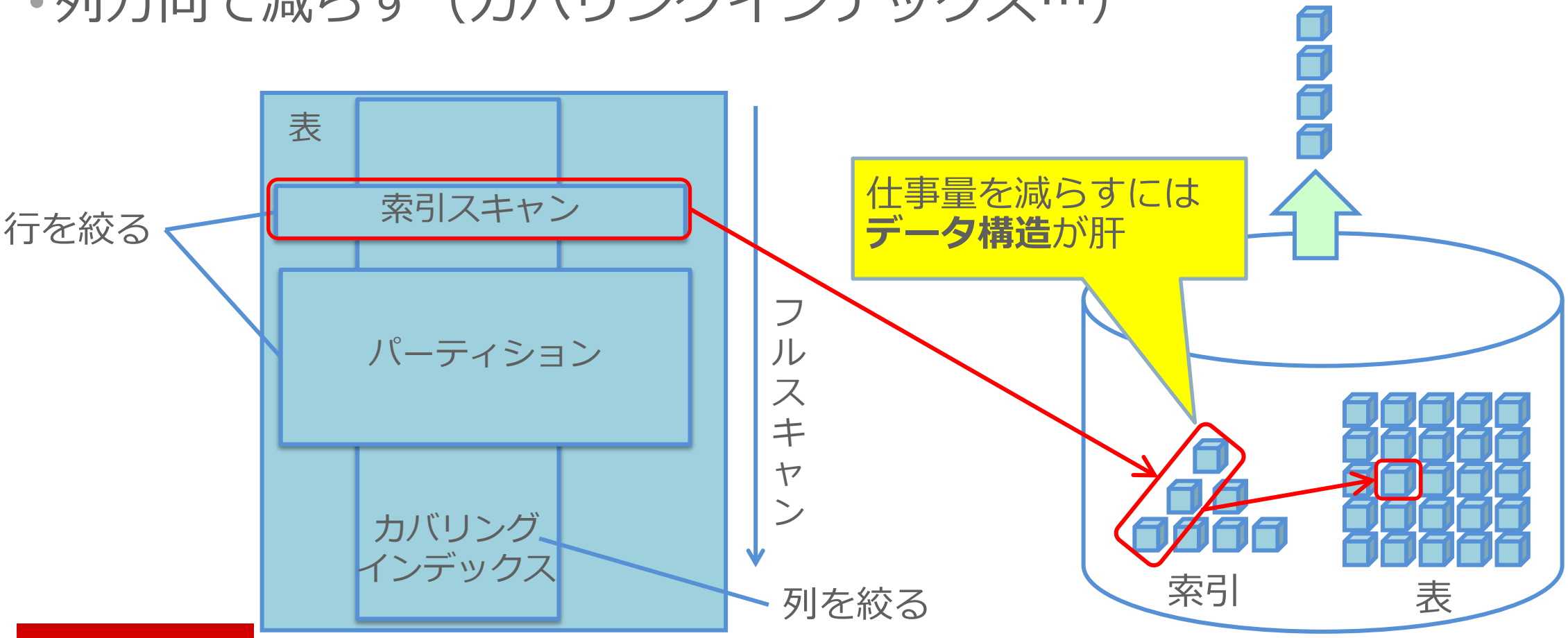


3）高速化



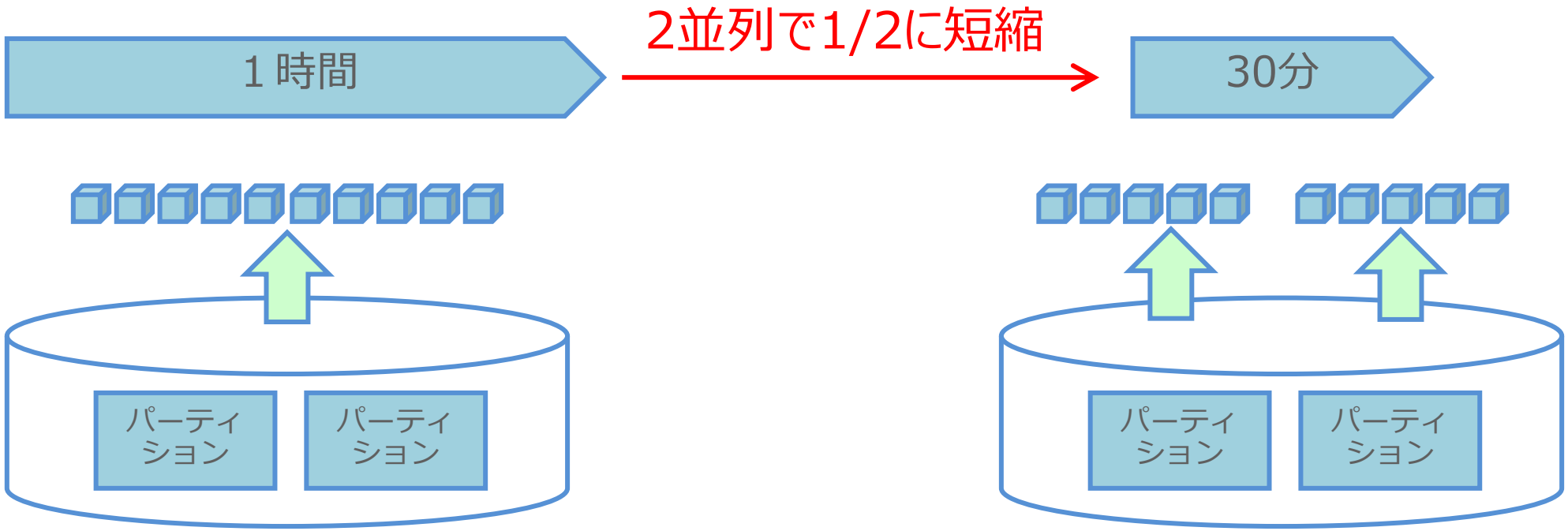
1）仕事量を減らす

- 行方向で減らし（索引・パーティション…）
- 列方向で減らす（カバリングインデックス…）



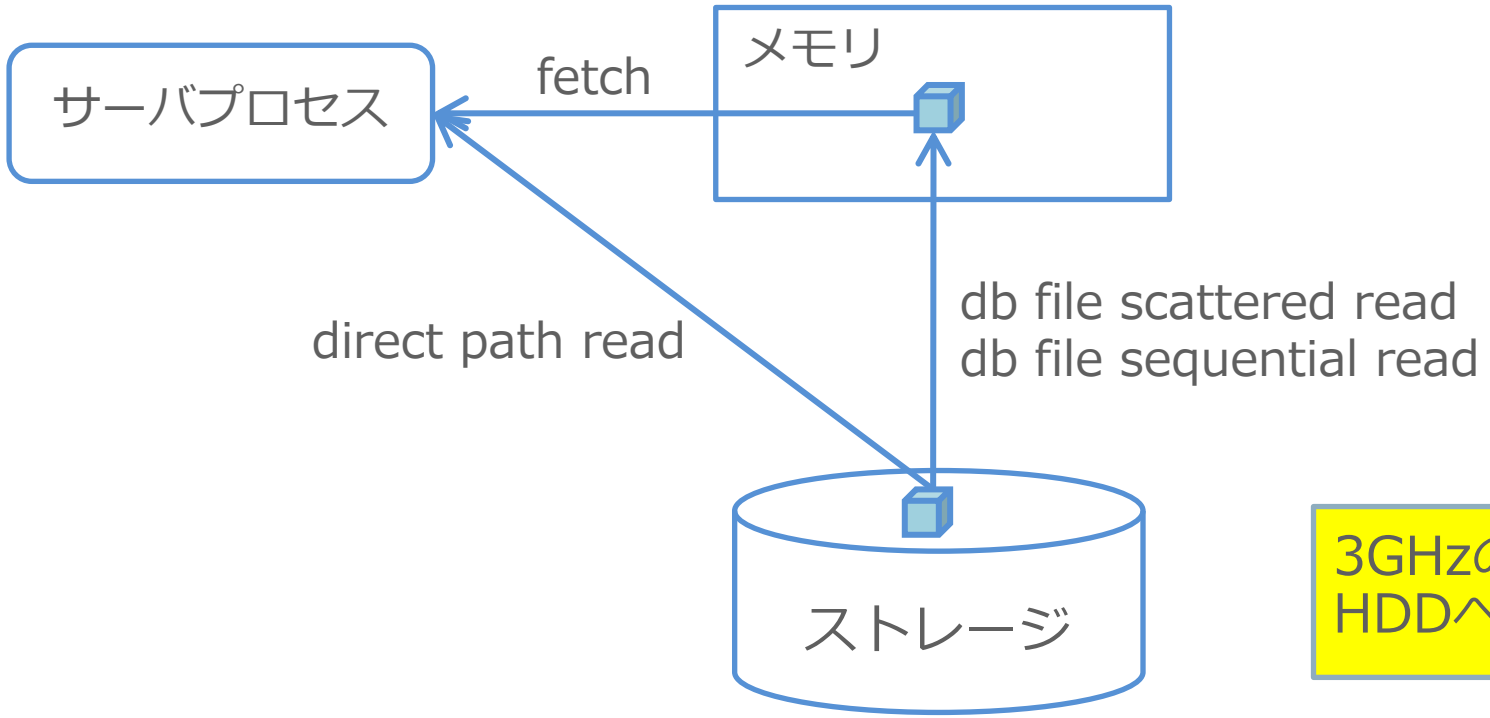
2）並列化

- 「1 / 並列度」に短縮できる
- シンプルなタスクでないと並列化が難しい（効果が出ない）
- 並列処理するリソースが必要（CPU数、ストレージのスループット等）



3）高速化

- データをメモリに置く（KEEPプール、DB In-Memory…）
- 速いストレージを使う（フラッシュ、SCM…）
- 速いCPUを使う（クロック周波数、専用の命令…）



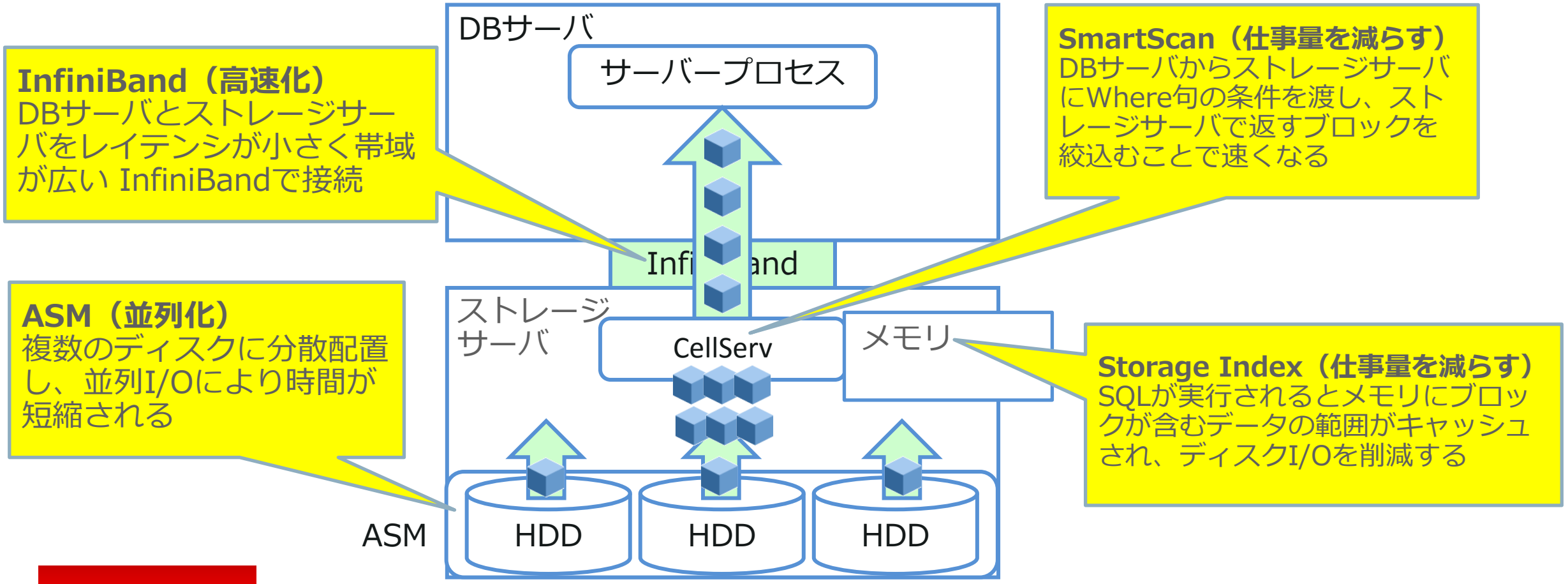
CPUの1サイクルを1秒とすると…

デバイス	レイテンシ	比較
1CPUサイクル	0.3ns	1秒
メモリ	120ns	6分
SSD	50-150μs	2-6日
HDD	1-10ms	1-12ヶ月

3GHzのCPUの1サイクルを1秒とすると、
HDDへのアクセスは1～12ヶ月にもなる

Exadata の中の「処理を速くする3つの方法」

仕事量を減らす、並列化、高速化はあらゆるレイヤーで有効な考え方



DD2-3 しばちょう先生の特別講義！！ ストレージ管理のベストプラクティス
～ASMからExadataまで～ より

時間↓ ＝ 処理量↓ / （速度 * 並列度）↑

じかん = みちのり ÷ はやさ

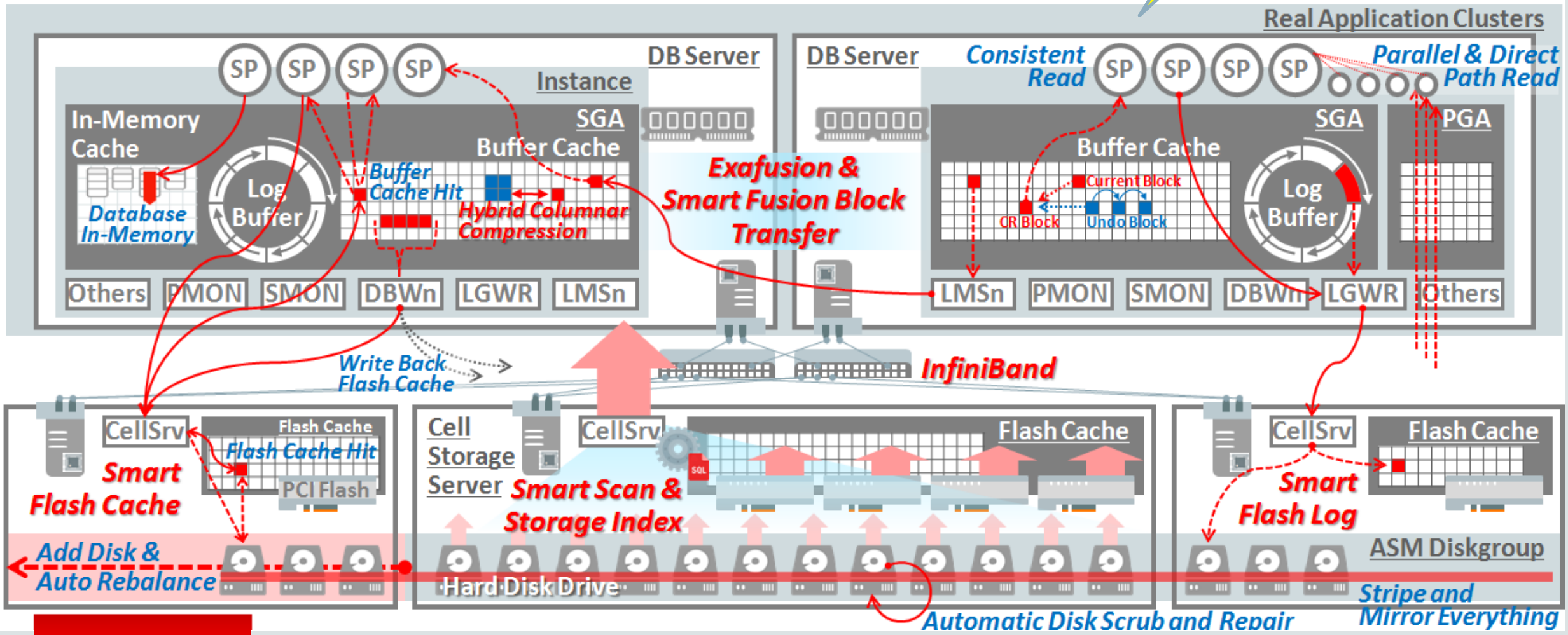
- 処理量を減らす
  - Index, Partitioning, Compression, Exadata Smart Scan/Storage Index …
- 高速化
  - Database In-Memory, Flash Device, InfiniBand, Exafusion, …
- 並列化
  - Parallel Query, Multi-Core, RAC, **ASM**, …

処理を速くする３つの方法

1つ前のスライドの話

DD2-3 しばちょう先生の特別講義！！
ストレージ管理のベストプラクティス
～ASMからExadataまで～ より

Exadata のあらゆる箇所に処理を速くする3つの工夫が実装



RDBMS のコストモデル

- オプティマイザはSQL処理を 単位処理 に分解
- 単位処理の 単価（時間） × 回数 でコスト算出
- 総コストが最小になる実行計画を導出する

例）10ミリ秒　　例）100回

総コスト = ランダムI/O単価 * 回数 + シーケンシャルI/O単価 * 回数

単位処理 =1秒

+ 1行の演算に必要なCPUサイクル × 行数 …

Oracle Database のコストモデル

システム統計を加味したCPUコストモデル（I/Oコスト + CPUコスト）

単価：システム統計　　回数：統計情報をベースにCBOが算出

総コスト =
SREADTIM（ミリ秒）
単一ブロック読込にかかる時間（単価）
＊ 10（回）

+ MREADTIM（ミリ秒）
複数ブロック読込にかかる時間（単価）
＊ 100（回）

} I/Oコスト

+ 必要なCPUサイクル数 …

} CPUコスト

Oracle Database の実際のコスト計算式ではなく単純化した式

実際の Oracle Database の CPU コストモデル式

ミリ秒で総コスト算出後にSREADTIMで割っているので、コストを「シングルブロックリードの回数」という単位で表現

```
COST = (#SRds * SREADTIM +
        #MRds * MREADTIM +
        #CPUCycles / (CPUSPEED * 1000)) / SREADTIM
```

時間の単位をミリ(1/1,000)秒に合わせている

#SRds: 単一ブロック読込み数
#MRds: 複数ブロック読込み数
#CPUCycles: CPUサイクル数（MHz）
SREADTM: 単一ブロック読込み時間（1/1,000秒）
MREADTM: 複数ブロック読込み時間（1/1,000秒）
CPUSPEED: 1秒あたりのCPUサイクル（MHz）

Oracle9i データベース・パフォーマンス・チューニング・ガイドおよびリファレンス リリース 2（9.2） 部品番号：J06248-02
http://otndnld.oracle.co.jp/document/oracle9i/920/generic/server/J06248-02.pdf

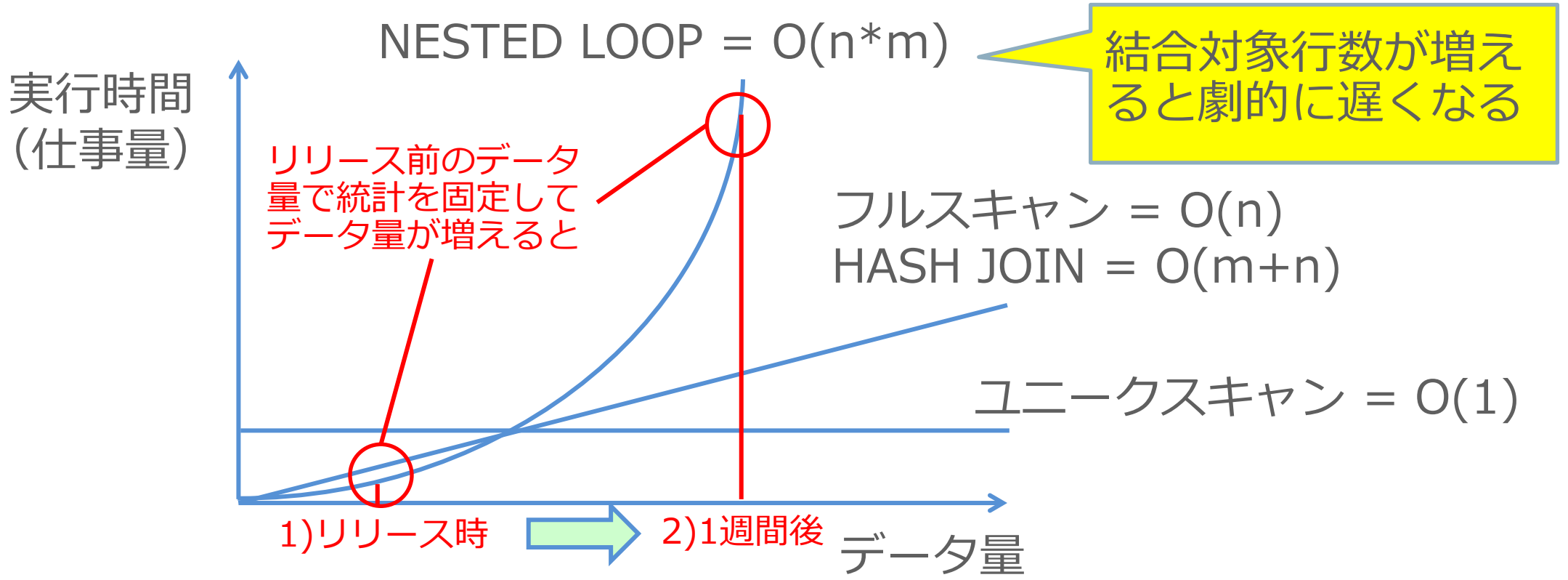
この予測が EXPLAIN PLAN の「コスト」

```
EXPLAIN PLAN FOR
SELECT * FROM TBL_A WHERE C1 <= :B1;

Explained.

SET LINESIZE 170;
SET PAGESIZE 1000;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY('PLAN_TABLE', NULL, 'ALL'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------

Plan hash value: 3263267004

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |           |   500 |  6000 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| TBL_A     |   500 |  6000 |     3   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | TBL_A_PK  |    90 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

あくまで見積
実態ではない

将来の仕事量を予測する

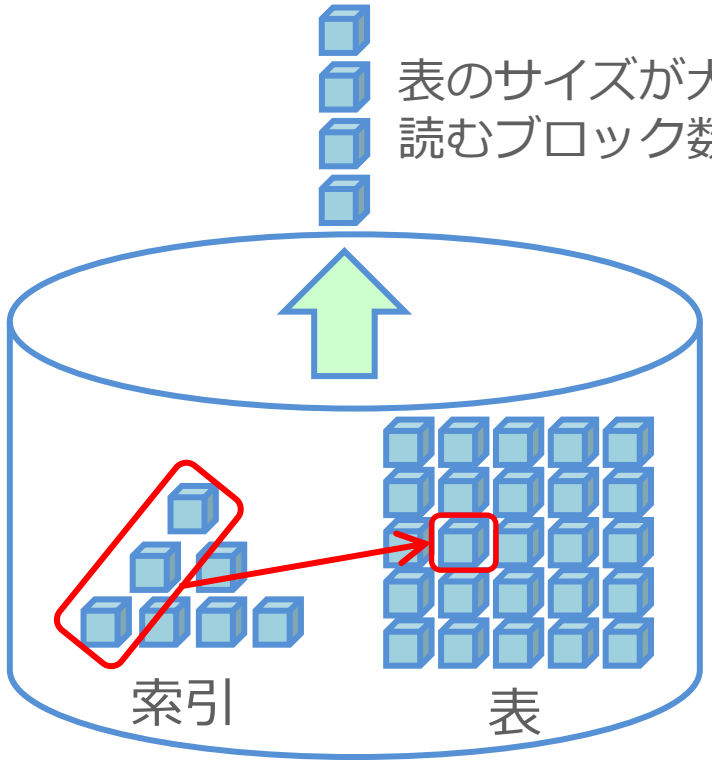
実行計画の種類によりデータ増加時の遅くなり方が違う

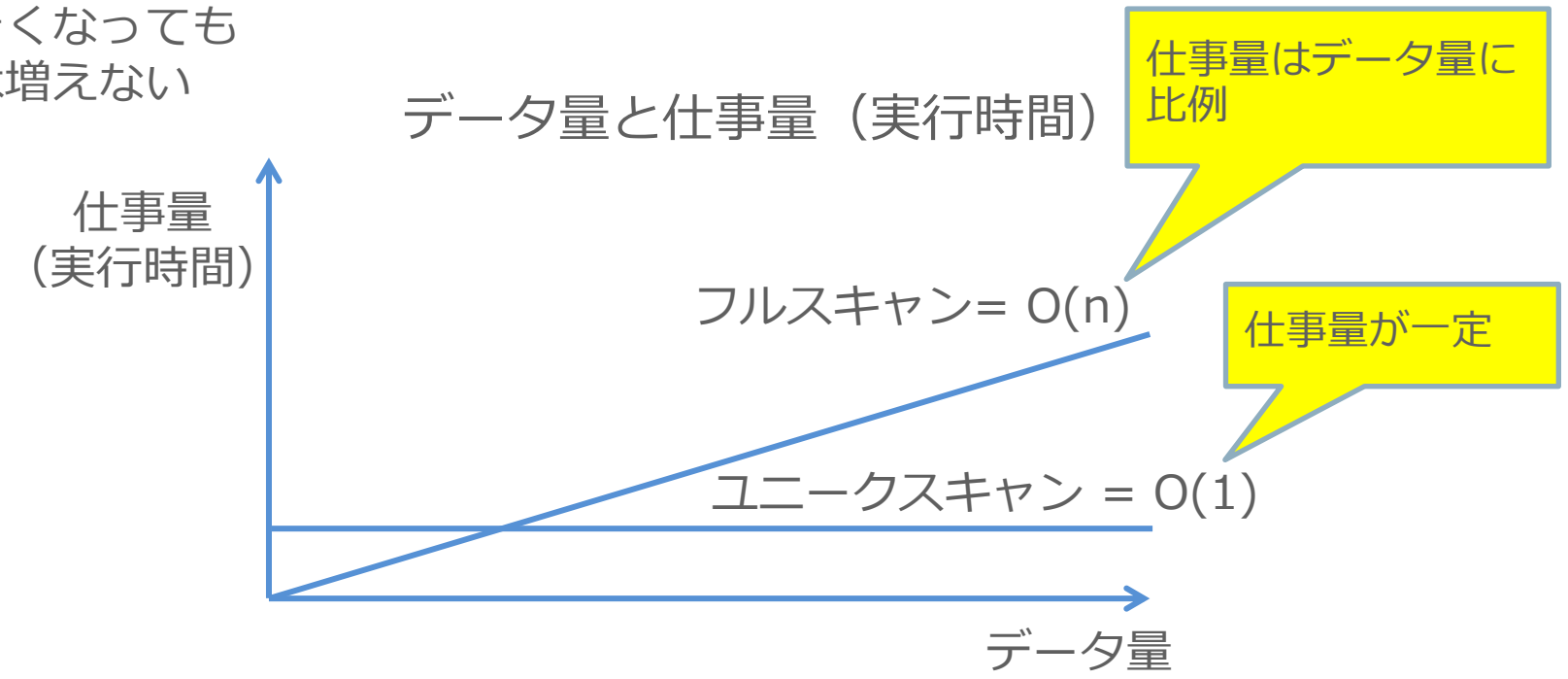


「O(1)」などはO（Order）記法と呼ばれるアルゴリズムで入力に対する仕事（計算）量の関係を表す記法

仕事量が一定の索引ユニークスキャン

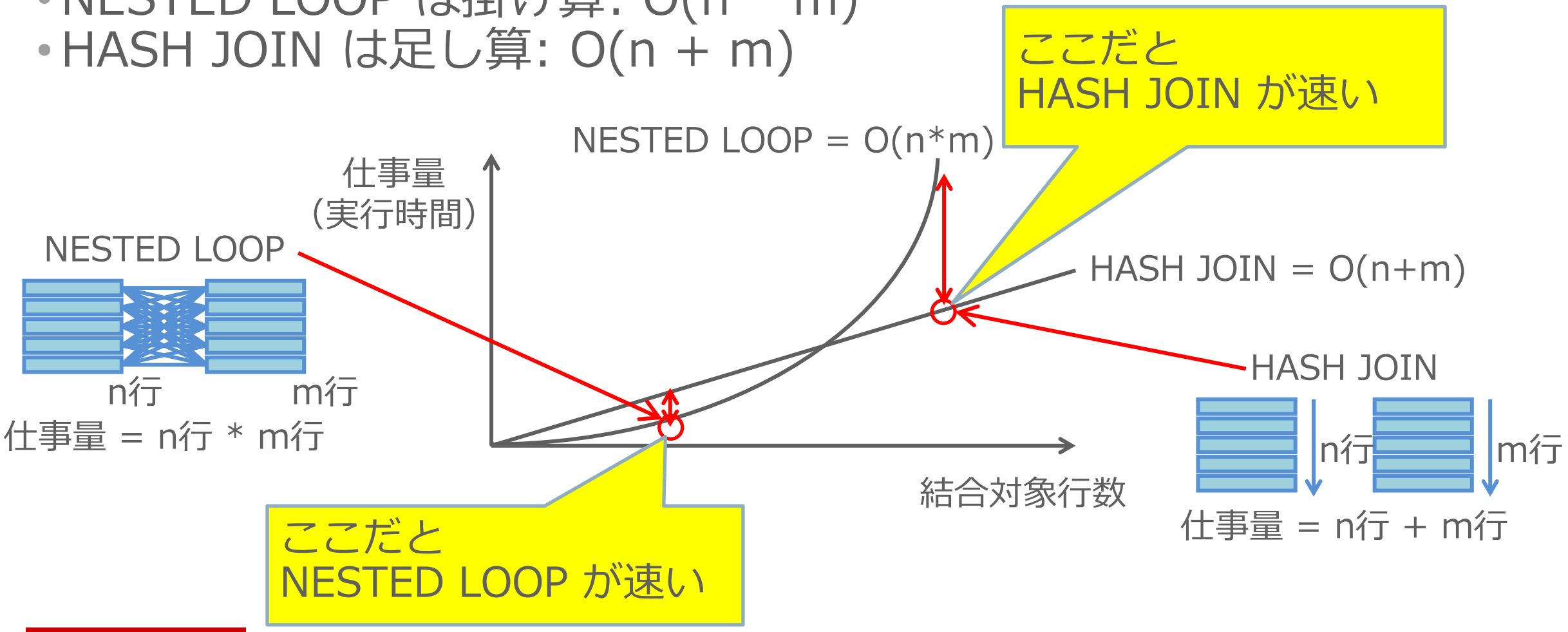
- 索引ユニークスキャンはデータ量が増えても遅くならない
- フルスキャンはデータ量に比例して遅くなる





NESTED LOOP は掛け算、HASH JOIN は足し算

- NESTED LOOP は掛け算: O(n * m)
- HASH JOIN は足し算: O(n + m)



Exadata/SuperCluster のシステム統計

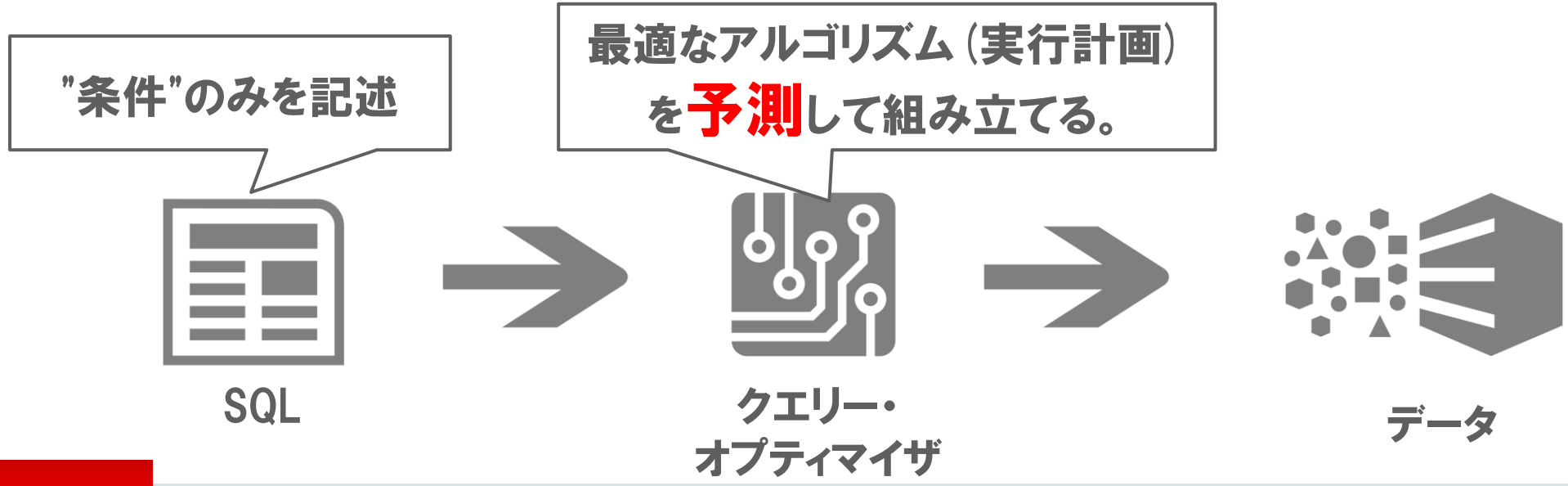
- Exadata/SuperCluster は FULL SCANが速い。
  ⇒ FULL SCANのコストが低く見積もられる。
    ⇒ FULL SCANになり易い。

- Exadata/SuperCluster用の特殊な実行計画が有るわけではない。
  ⇒ CBOのロジック／エンジンは、
    通常のOracle Database EE/SEと共通

- Exadata/SuperClusterで速い実行計画
  ⇒ FULL SCAN + HASH JOIN + PARALLEL
    ⇒ OLTP的な実行計画(INDEX + NL + SINGLE)との対称性

SQL の アルゴリズム は「予測」で組み立てられる。

- SQL と云う言語は、アルゴリズムを書かずにデータ抽出の条件のみを書けば良いという特徴があります。
- 効率の良い アルゴリズム＝実行計画 を予測して組み立てるのが、クエリー・オプティマイザ の 役割



SQLの実行計画はハズレを引く可能性が常に有る。

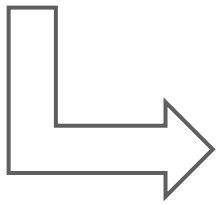
- SQL の アルゴリズム=実行計画 は、クエリー・オプティマイザが最適と考えられるものを「予測」して組み立てます。
- そして「予測」である以上、必ずハズレが出てきます。
  - アルゴリズムを書かないと云うSQLの特徴に由来する、世間一般のRDBMS に共通した本質的な困難
  - Oracle Database は 様々な機能でこのハズレを補正している。

キーワード

「ハズレ」

「ハズレ」が常に有り「予測」の精度が重要になる。

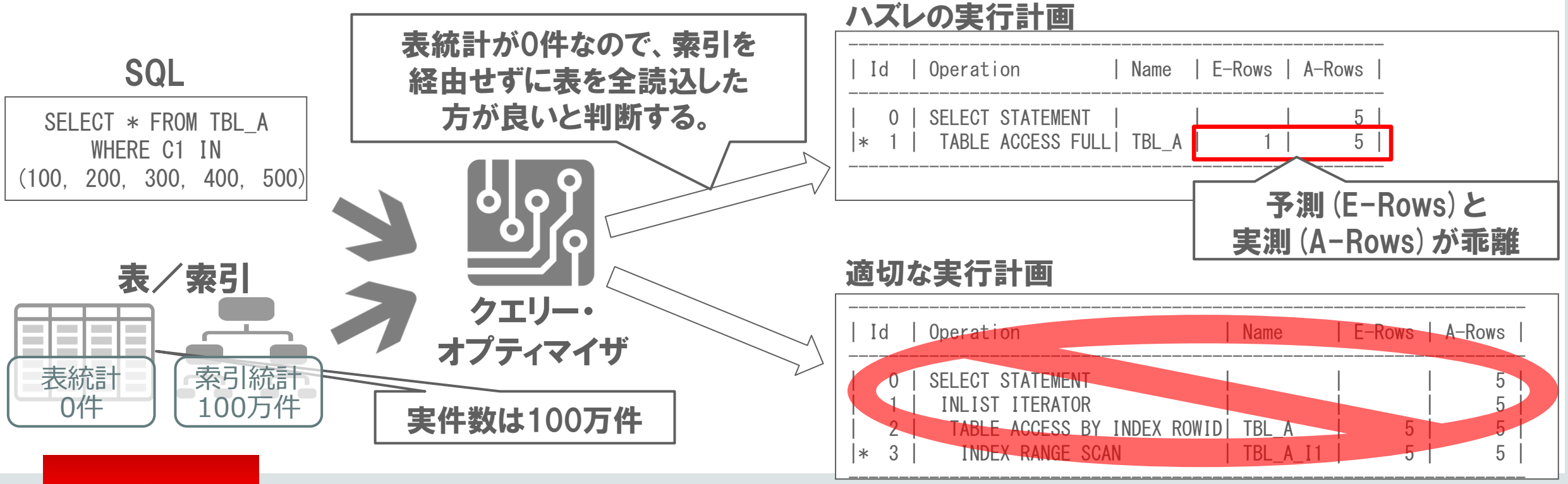
- SQL はアルゴリズムを書く必要が無く、誰でも比較的簡単に書けると云う長所があります。
- 一方 SQL はアルゴリズム＝実行計画 が「予測」で組み立てられるため、「ハズレ」の実行計画を引く可能性と、常に隣合わせと云う短所も併せ持ちます。



「ハズレ」を減らして「予測」精度を上げると云う考え方が必要

参考：ハズレの実行計画例・統計と実態の乖離

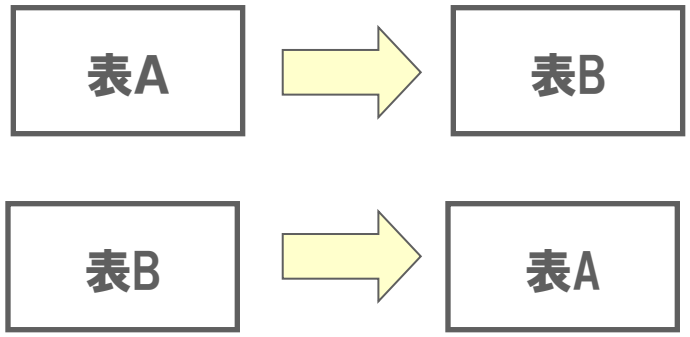
- 下記の例は、表／索引の実態(実件数)と統計が乖離して、性能が悪いハズレの実行計画が選択されるケースです。
  - 表統計が0件で実態と乖離しており、索引を経由せずに表を全読込した方が仕事量が少ないとオプティマイザが判断しています。



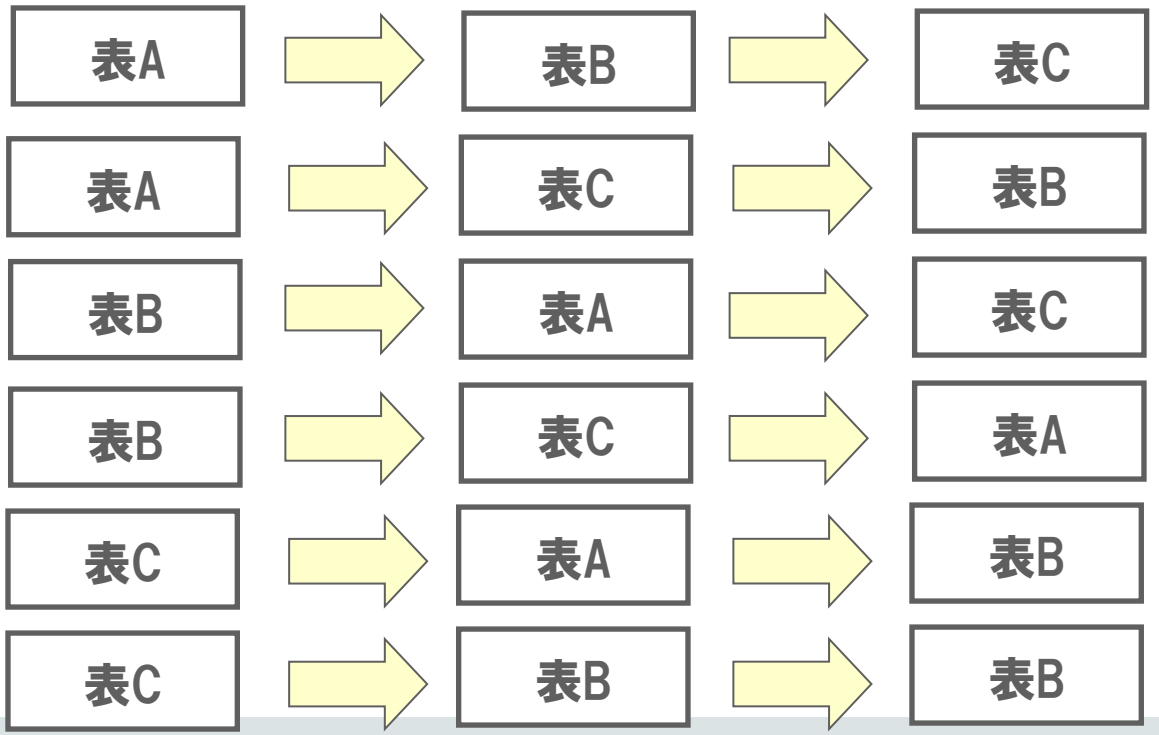
参考：ハズレの実行計画例・結合表が多い(1/2)

- Oracleのオプティマイザは、実行計画作成時に表の適切な結合順序を見つけるために全ての組み合わせを解析しようとします。

結合する表が2つの場合
⇒ 2！（2の階乗）= 2×1＝2通り



結合する表が3つの場合
…3！（3の階乗）=3× 2×1＝6通り



参考：ハズレの実行計画例・結合表が多い(2/2)

- 結合する表が多い場合は、Oracleのオプティマイザは結合順序の解析を途中で止めます。
  - –全組み合わせの解析には、時間が掛かり過ぎるからです。
  - –デフォルトで2000通りまでの結合順序を解析します。
- 結合表が多過ぎるSQLは、適切な結合順序に辿り着けず、ハズレの実行計画を引くリスクが高いと言えます。

結合する表が8つの場合 ⇒ 8!（8の階乗）＝8*7*6*5*4*3*2*1＝40320通り

参考：ハズレの実行計画例・列値同士に相関が有るケース

- 下記は列の値同士の相関によって予測と実測が乖離して、ハズレの実行計画が選択されてしまうケースです。
  - この例では拡張統計(複数列統計)が有効で、結合順序が適切になります。

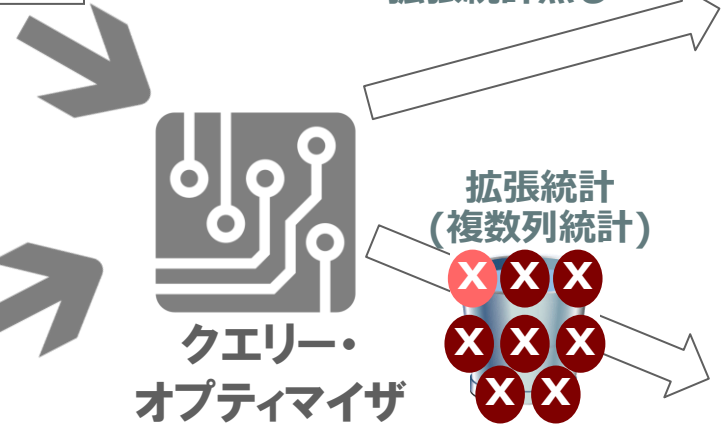
SQL

```
SELECT * FROM TBL_A
WHERE (C2, C3) IN (
  SELECT DISTINCT C2, C3
    FROM TBL_B
   WHERE C2 = 0 AND C3 = 'A');
```

テーブル

...	C2	C3
...	0	A
...	0	A
	1	B
...	1	B
...	...	...

列の値同士に相関



複数列統計無し(ハズレの実行計画)

```
-------------------------------------------------------------------
| Id  | Operation          | Name  | E-Rows | A-Ro
-------------------------------------------------------------------
|   0 | SELECT STATEMENT   |       |        |   3846
|*  1 |  HASH JOIN SEMI    |       |    148 |   3846
|*  2 |   TABLE ACCESS FULL| TBL_A |    148 |   3846
|*  3 |   TABLE ACCESS FULL| TBL_B |     74 |      1
-------------------------------------------------------------------
```

予測(E-Rows)と実測(A-Rows)が 乖離

複数列統計有り(適切な実行計画)

```
-------------------------------------------------------------------
| Id  | Operation          | Name  | E-Rows | A-Row
-------------------------------------------------------------------
|   0 | SELECT STATEMENT   |       |        |   3846
|*  1 |  HASH JOIN RIGHT SEMI|     |   3846 |   3846
|*  2 |   TABLE ACCESS FULL| TBL_B |   1923 |   1923
|*  3 |   TABLE ACCESS FULL| TBL_A |   3846 |   3846
-------------------------------------------------------------------
```

適切な予測と結合順序

フィルタ条件が多いと行数見積が小さくなりやすい

1行に絞れると予測したが、実は5,000行ヒットした。。。

```
SELECT …
    FROM T1
    WHERE COL1 =:バインド変数1
        AND COL2 = :バインド変数2
        AND COL3 = :バインド変数3
        AND COL4 = :バインド変数4
```

実は値に偏りがあり、フィルタ後行数が5,000の場合は見積が大きくハズれる

Number of Distinct Values=列値の種類数

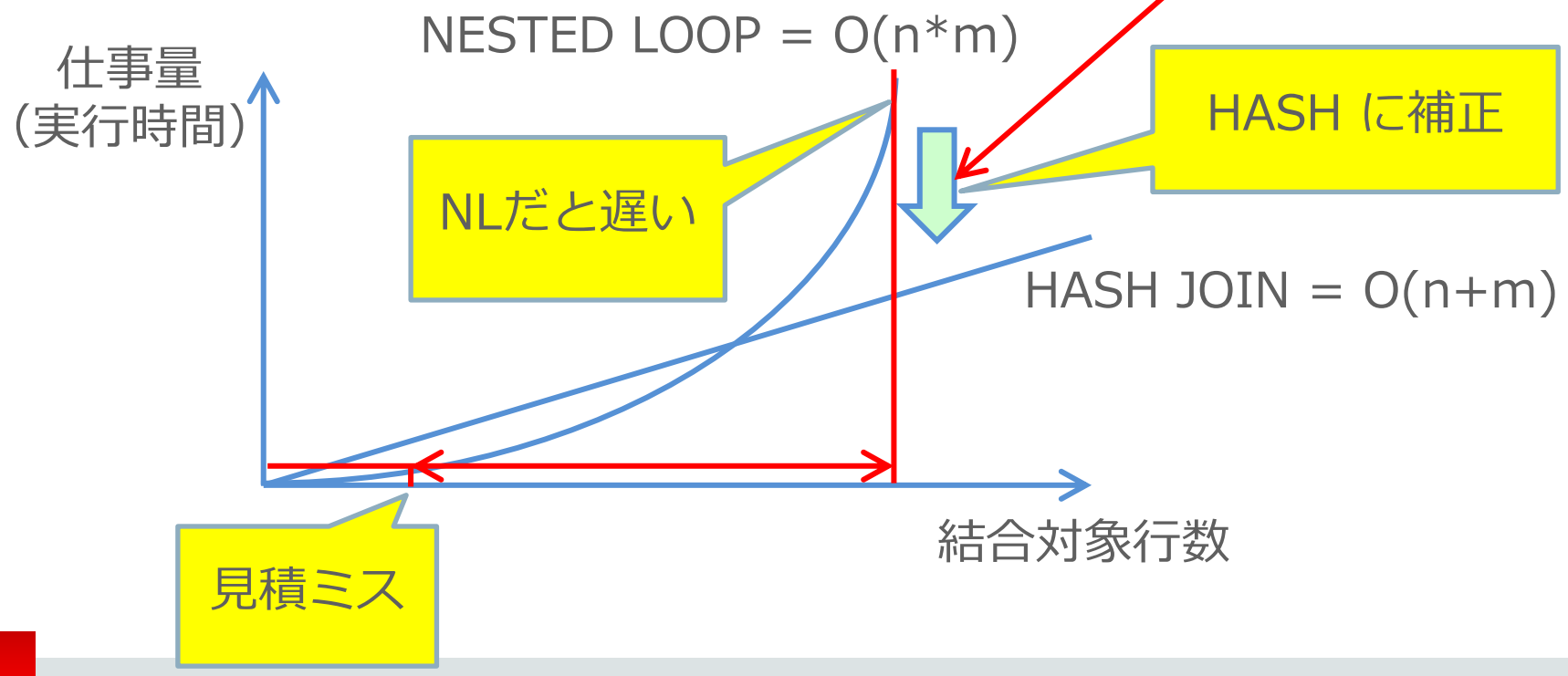
T1表のフィルタ後行数 = T1の行数 * (1 / COL1のNDV * 1 / COL2のNDV * …)
= 10,000行 * (1 / 10 * 1/10 * 1/10 * 1/10 = 1/10,000)
= 1行

COL1~4 の列値が10種類とすると

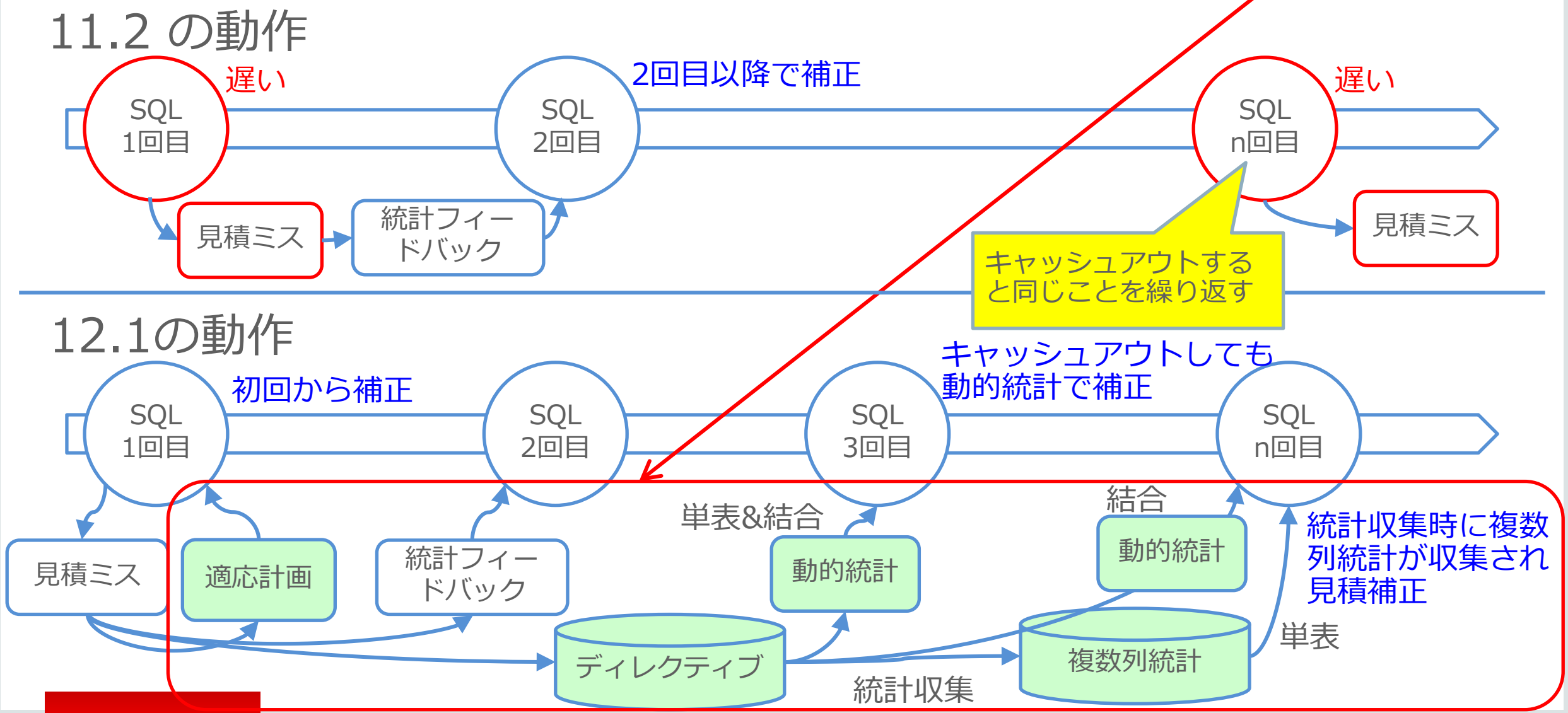
※T1表の行数はNULLを除いたものとする。バインドピーク無効化またはヒストグラム・複数列統計がない場合。

適応計画・SQL計画ディレクティブはこれを補正

- 統計が最新でも、複雑なクエリ（結合数・フィルタ条件が多い集計クエリなど）では正確なコスト見積が困難。
- 適応計画・SQL計画ディレクティブはこれを補正。

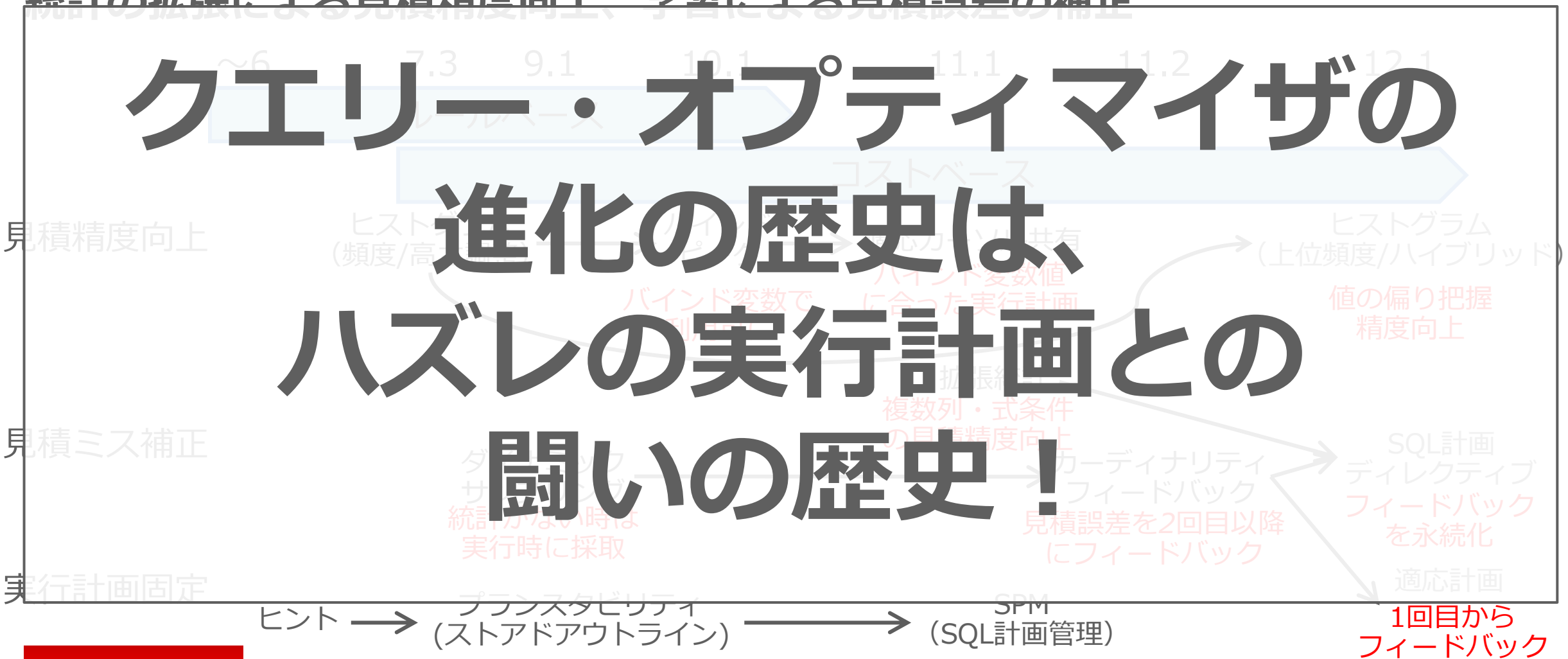


適応計画・SQL計画ディレクティブの補正イメージ



Oracle のクエリー・オプティマイザの進化

統計の拡張による見積精度向上、学習による見積誤差の補正

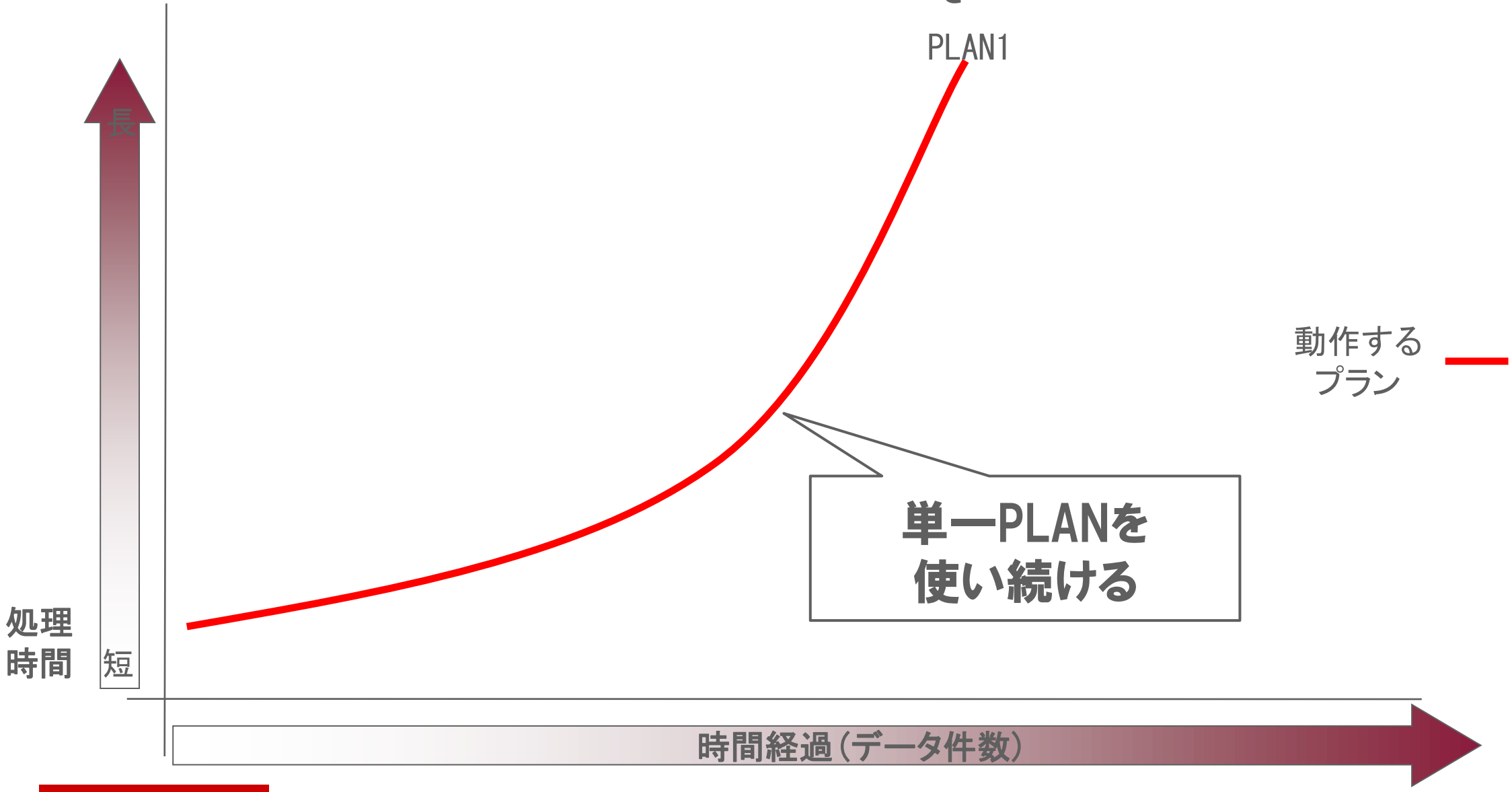


クエリー・オプティマイザの進化の歴史は、ハズレの実行計画との闘いの歴史！

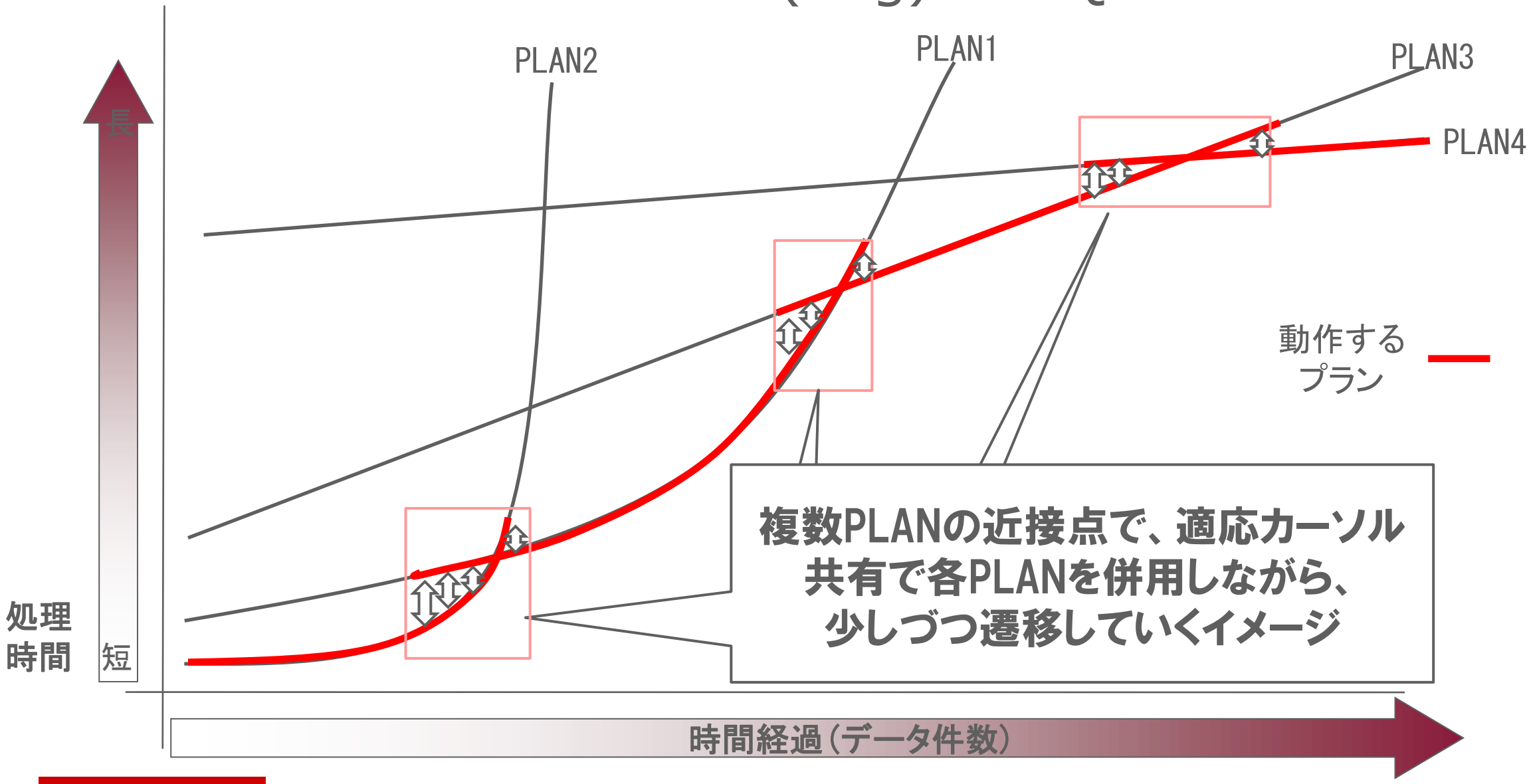
オプティマイザ統計の運用(固定化 or 最新化) と 最適化機能 との 組み合わせ で 考えてみます。

- オプティマイザ統計 と 最適化機能からのインプットは、実行計画の予測精度を上げる為の極めて重要な要素です。
- オプティマイザ統計運用を、2大潮流である「固定化運用」「最新化運用」と、クエリー・オプティマイザの各種最適化機能との組み合わせモデルで考えてみます。
  - ① 固定化運用＋最適化機能無し の モデル
  - ② 最新化運用＋最適化機能有り(11g) の モデル
  - ②'最新化運用＋最適化機能有り(12c) の モデル
  - ③ 最新化運用＋最適化機能無し の モデル

①固定化＋最適化無しモデル の SQL処理時間イメージ



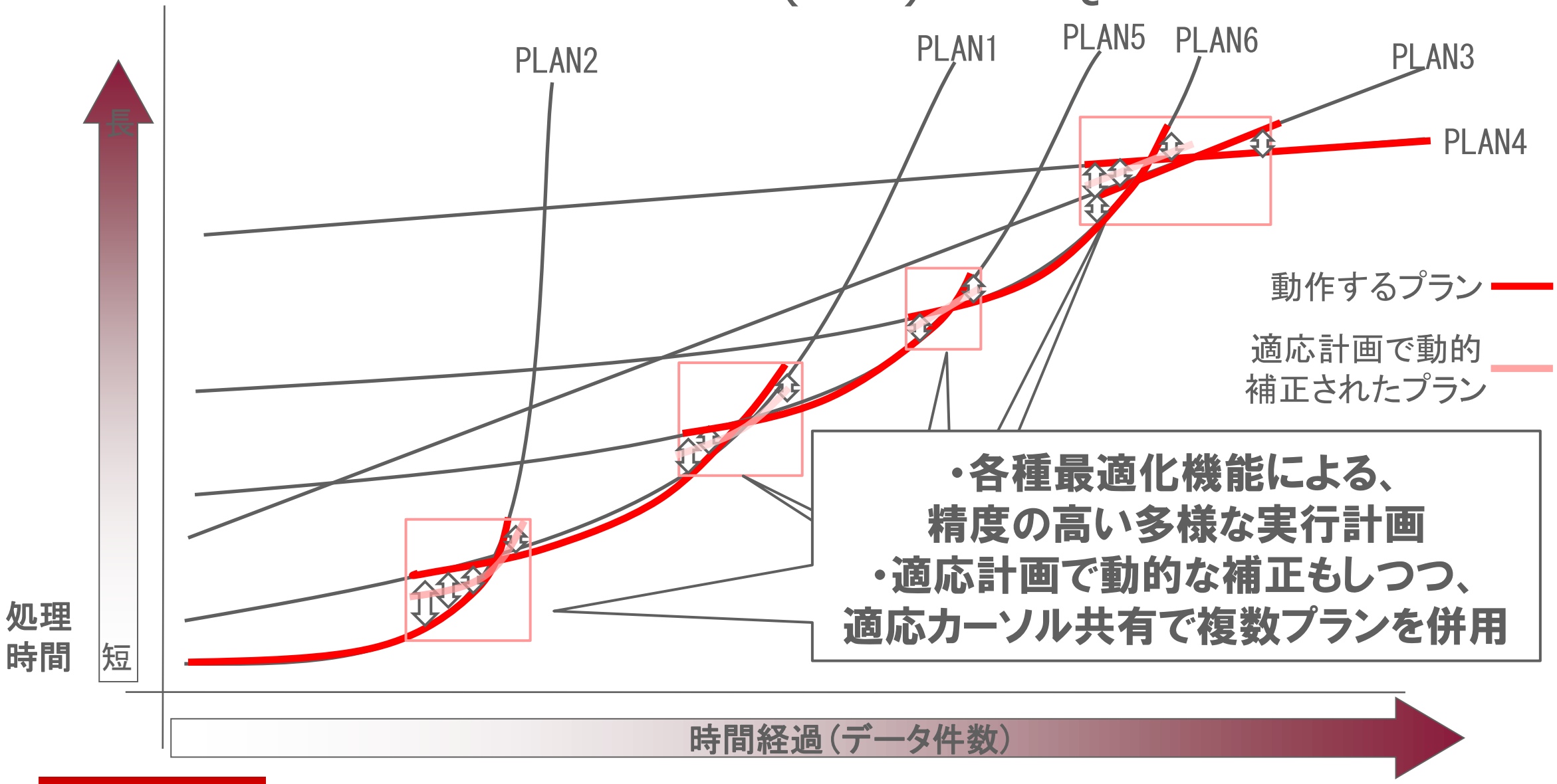
②最新化＋最適化有りモデル(11g) の SQL処理時間イメージ



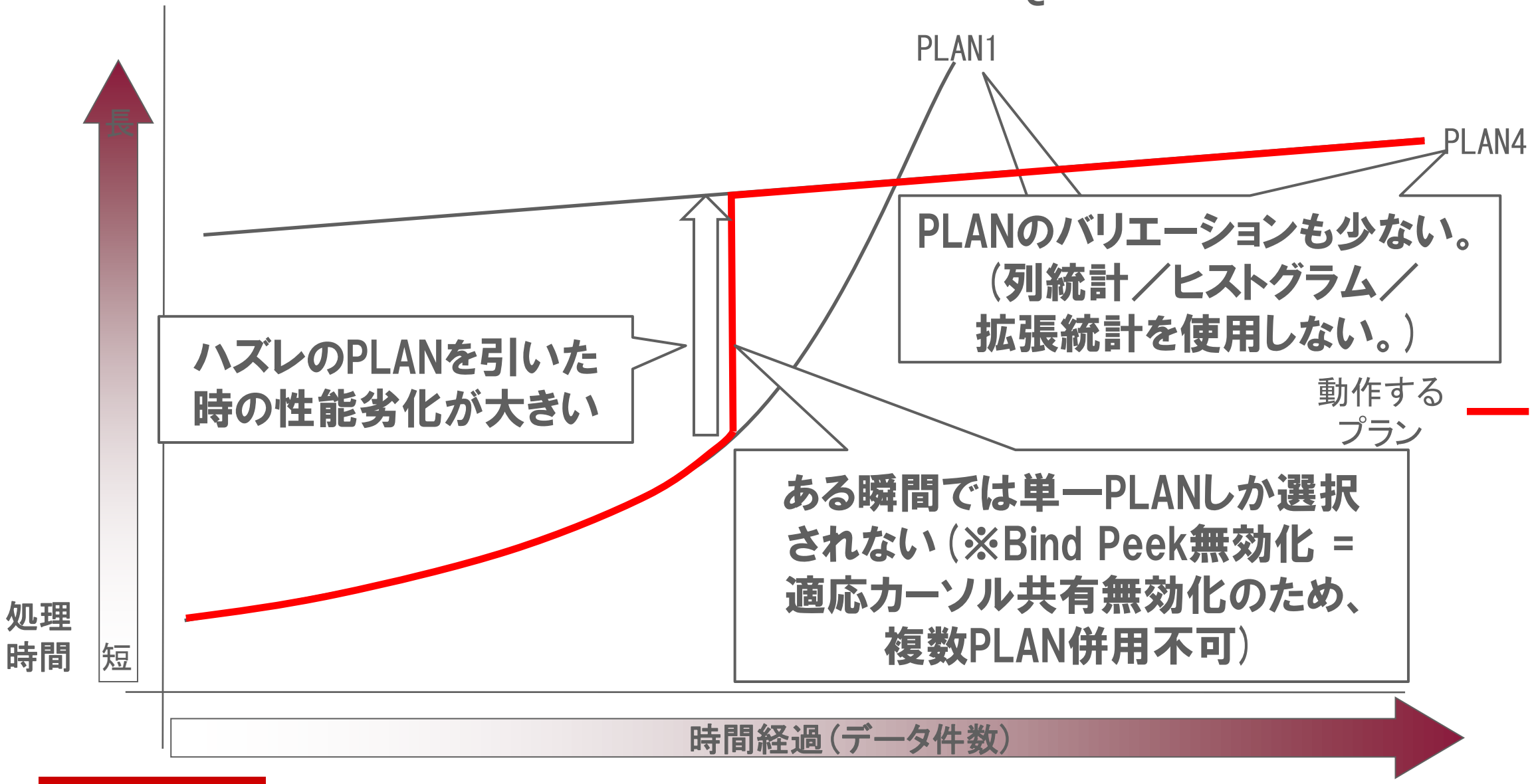
これに Oracle DB 12c の最適化機能が加わると…

- SQL計画ディレクティブ
- 適応計画(Adaptive Plan)
- 動的統計(Dynamic Statistics)

②'最新化＋最適化有りモデル(12c) の SQL処理時間イメージ



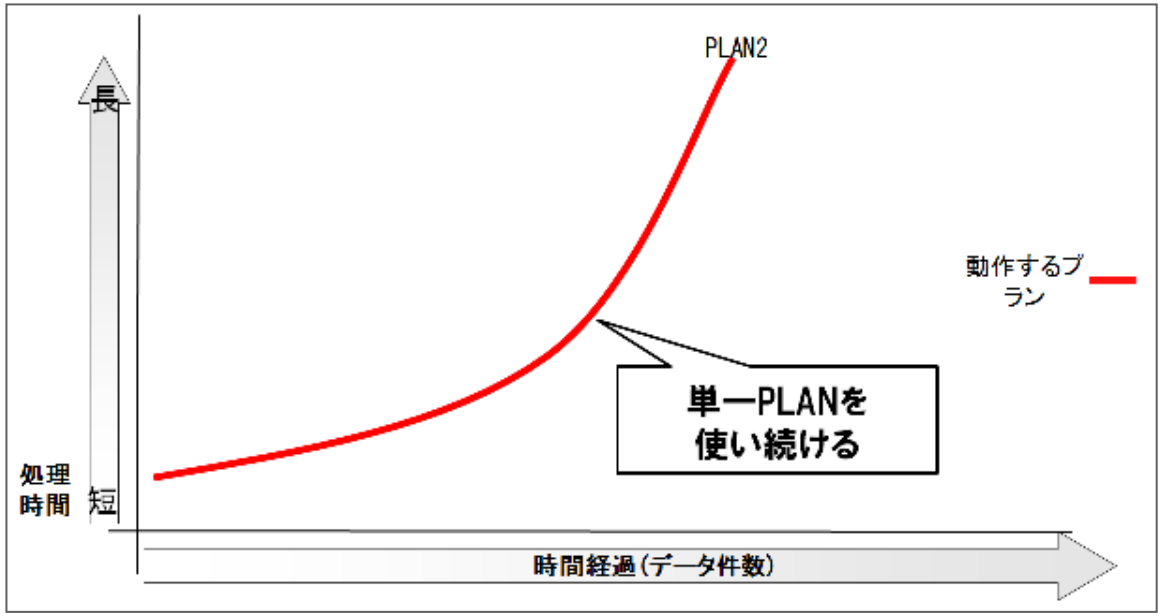
③最新化＋最適化機能無し モデルのSQL処理時間イメージ



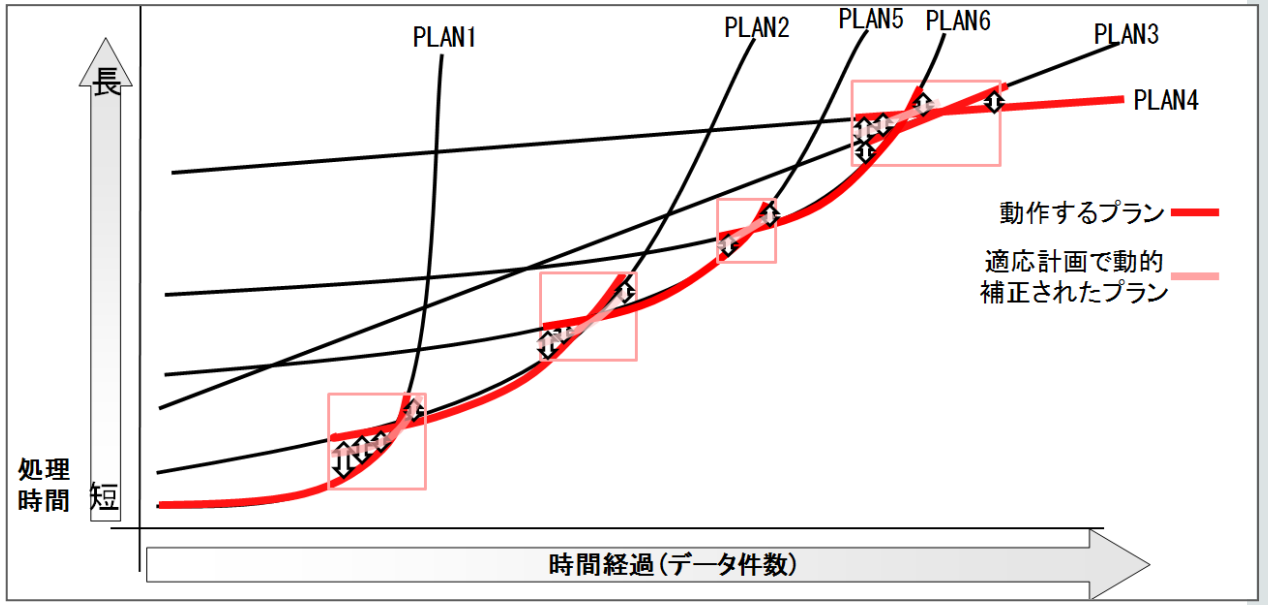
① と ②' の性能変動モデルケース比較

- 「赤線」が直線に近いほど、リスクは低い。
- どちらもリスクは低いが、性能は ②' の方が良い。

①固定化＋最適化無し



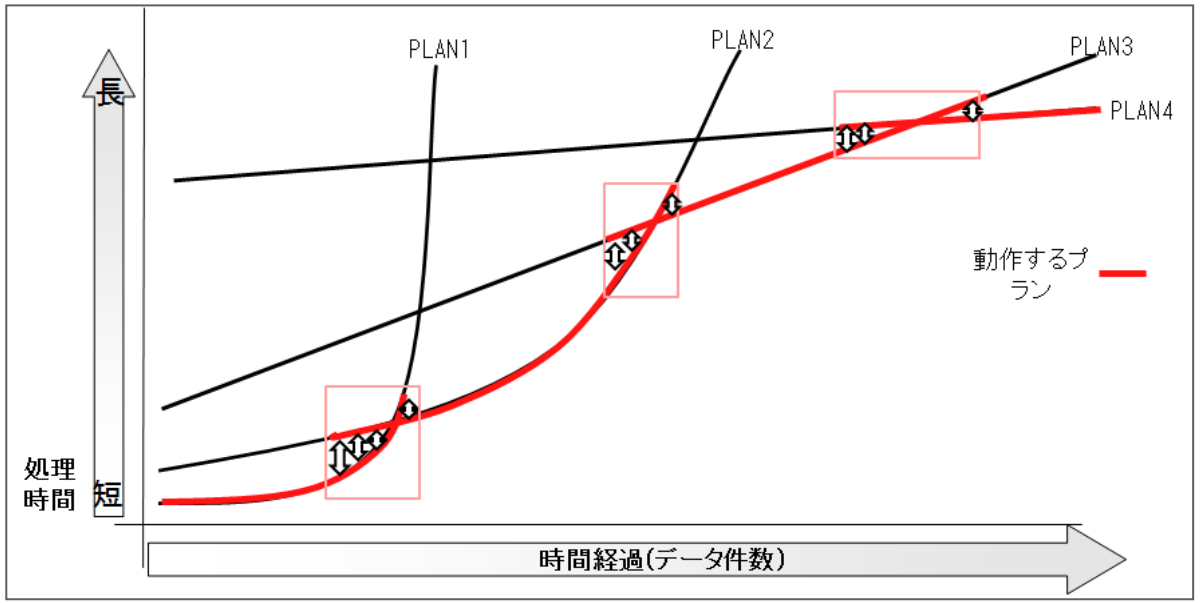
②' 最新化＋最適化有り(12c)



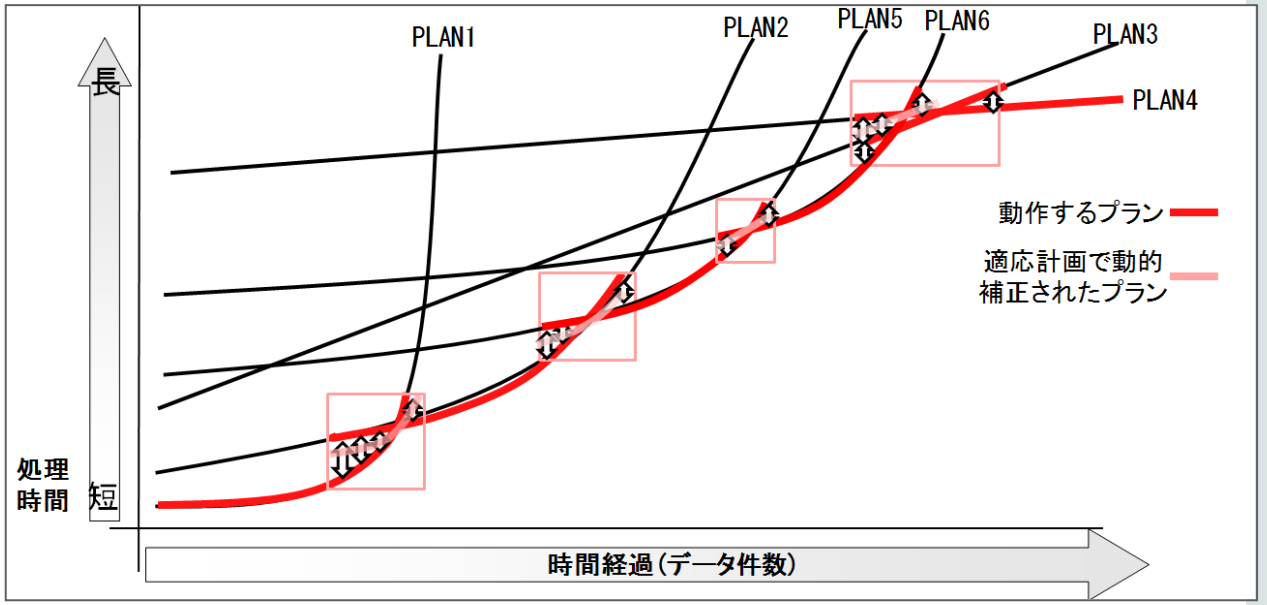
② と ②' の性能変動モデルケース比較

- 「赤線」が直線に近いほど、リスクは低い。
- 12c新機能によって更に直線に近づいた ②' の組み合わせ

②最新化＋最適化有り



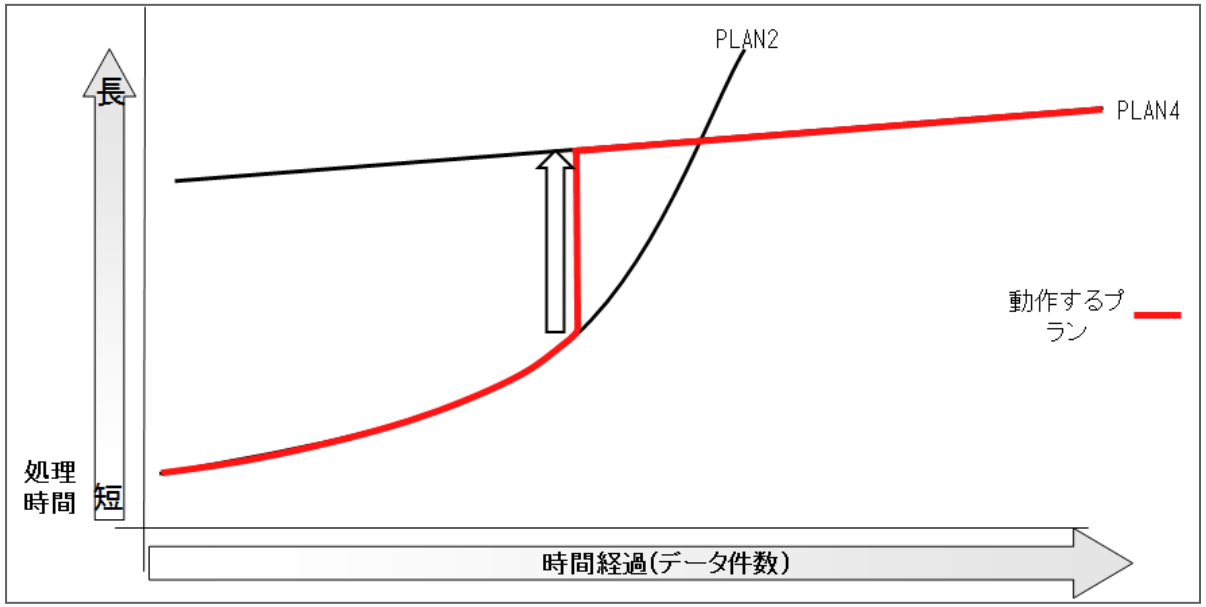
②' 最新化＋最適化有り(12c)



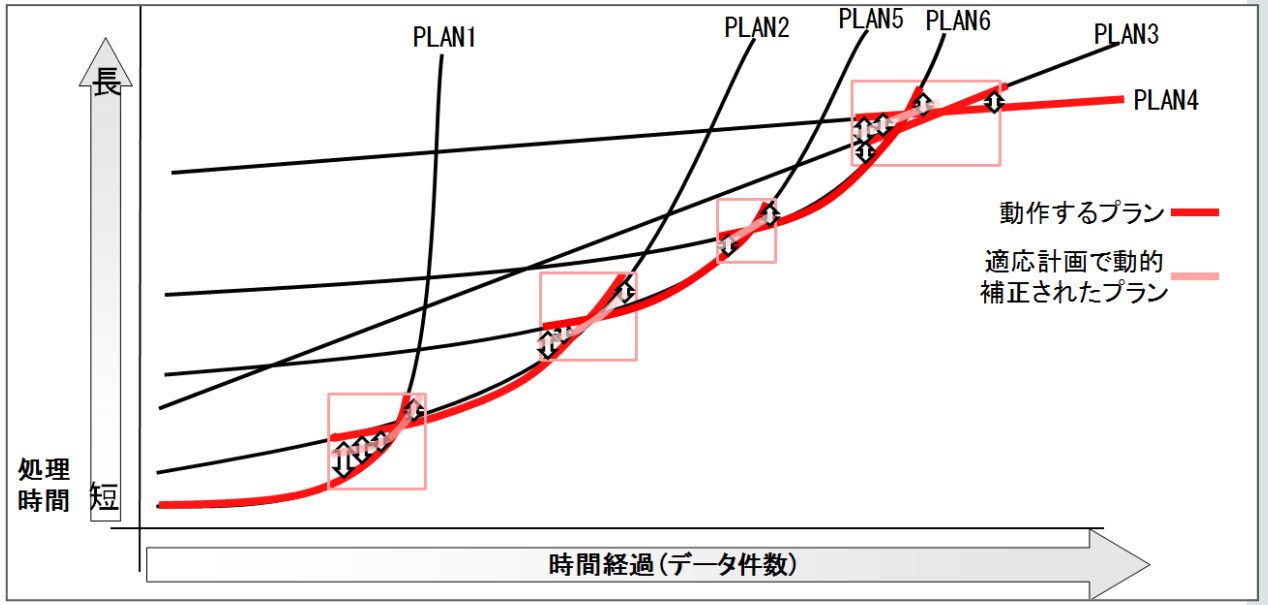
③ と ②' の性能変動モデルケース比較

- 「赤線」が直線に近いほど、リスクは低い。
- ③ よりも ②' の方が圧倒的に直線に近い。

③最新化＋最適化無し



②' 最新化＋最適化有り(12c)



"SQL性能"に加えて、更に評価軸を増やしてみる

- 前ページまでは "SQL性能" を中心に評価していますが、更に "運用負荷", "性能劣化リスク" を加えて、3軸で各モデルを評価してみます。

SQL性能

運用負荷

性能劣化リスク

"SQL性能" "運用負荷" "性能劣化リスク" による 評価例

- 下記は前ページで挙げた 3軸による評価例となります。

No	統計運用／最適化機能の組合わせ	SQL性能	運用負荷	性能劣化リスク
①	固定化運用＋最適化無し	× 精度の低い統計に加え最適化機能無し	× 新アプリや新環境リリース時のメンテナンス負荷	◎ 性能変動の要素がデータ増以外に無く、低リスク
②	最新化運用＋最適化有り(11g)	◎ 新鮮で高精度な統計と最適化機能の組合せ	◎ 自動化運用を重要視したモデル	△ ハズレの実行計画の可能性をゼロにはできない
③	最新化運用＋最適化無し	△ 最適化機能無しのため②よりも性能は低い	◎ C/O後の運用負荷は②と同等	× 最適化機能無しのため②より劣化リスクは高い

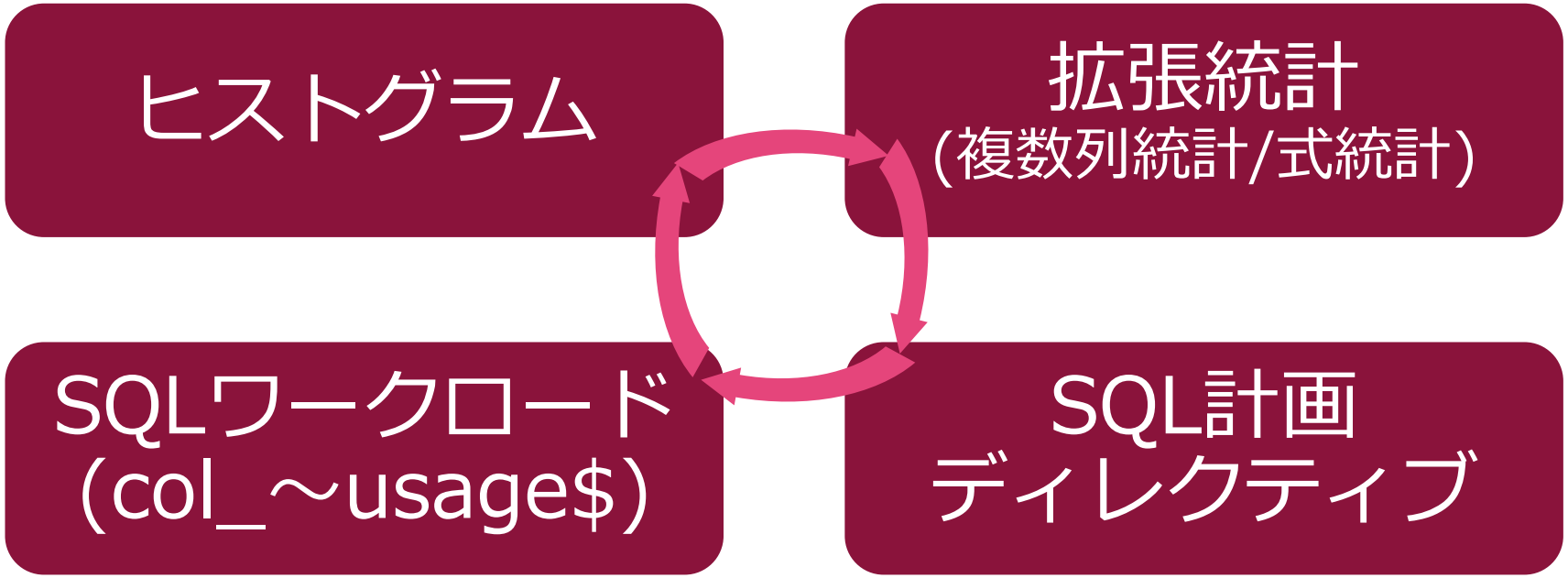
Oracle DB 12c の 新機能 が 加わると、更に良くなる。

- ②' は 12c新機能の恩恵で、性能劣化リスクが更に低くなります。

No	統計運用／最適化機能の組合わせ	SQL性能	運用負荷	性能劣化リスク
①	固定化運用＋最適化無し	✕ 精度の低い統計に加え最適化機能無し	✕ 新アプリや新環境リリース時のメンテナンス負荷	◎ 性能変動の要素がデータ増以外に無く、低リスク
②'	最新化運用＋最適化有り(12c)	◎ 新鮮で高精度な統計と最適化機能の組合せ	◎ 自動化運用を重要視したモデル	○ 12cの各種最適化機能により、リスクが更に低下
③	最新化運用＋最適化無し	△ 最適化機能無しのため②よりも性能は低い	◎ C/O後の運用負荷は②と同等	✕ 最適化機能無しのため②より劣化リスクは高い

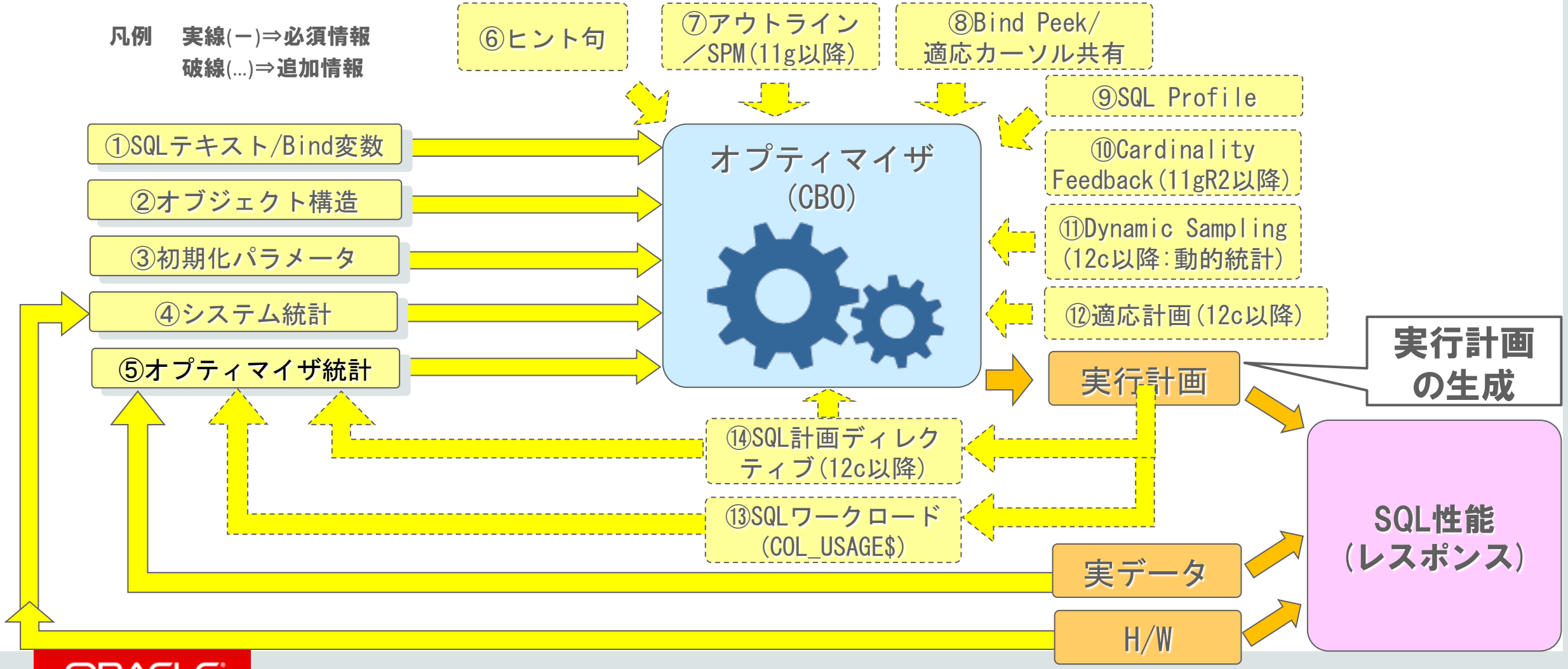
12cの新機能が切り拓く、新しい世界

- 更にこの(②')モデル、新しい世界が待っています。
- 今までに紹介した実行計画の各種最適化機能と、オプティマイザ統計の最新化運用が組み合わさると…



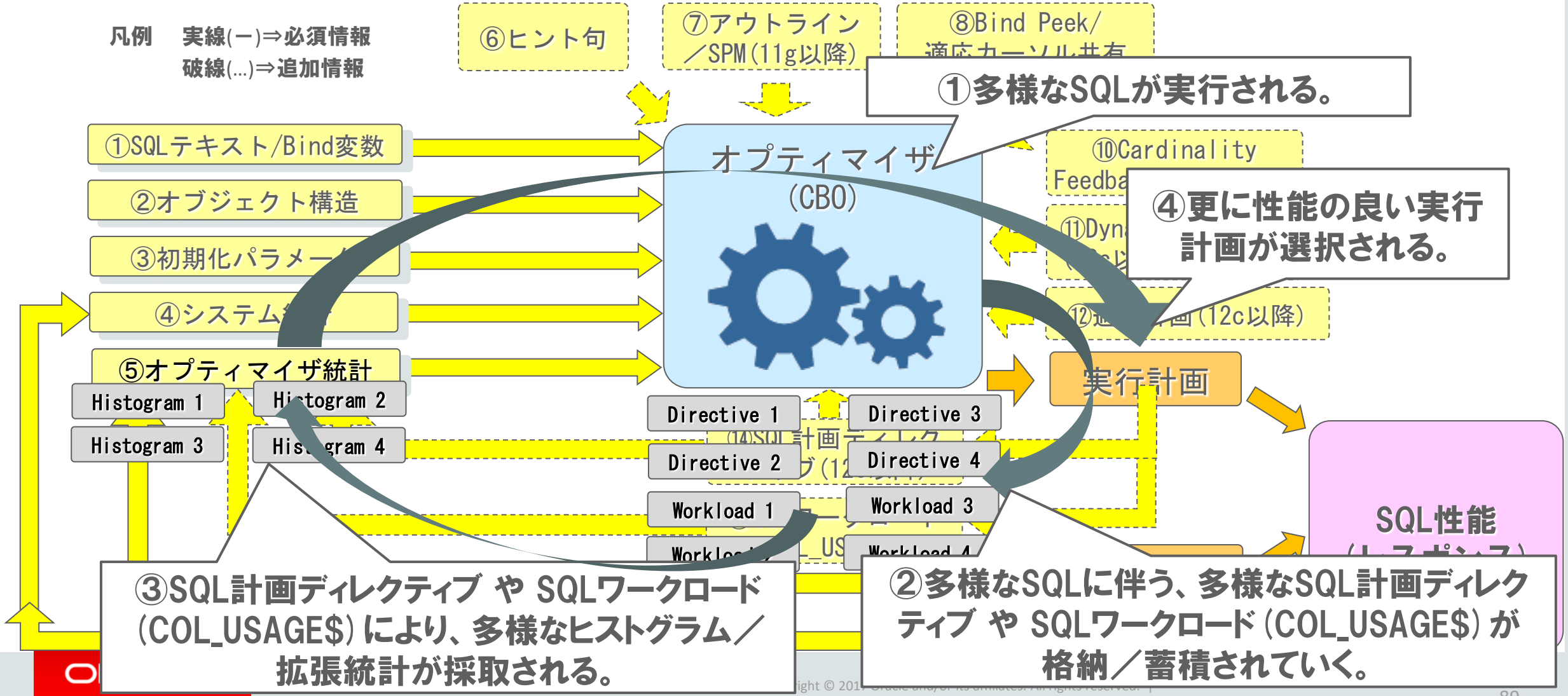
Oracle Database 12cR1 における 実行計画生成の全体像

- オプティマイザは様々なインプットを元にして実行計画を生成



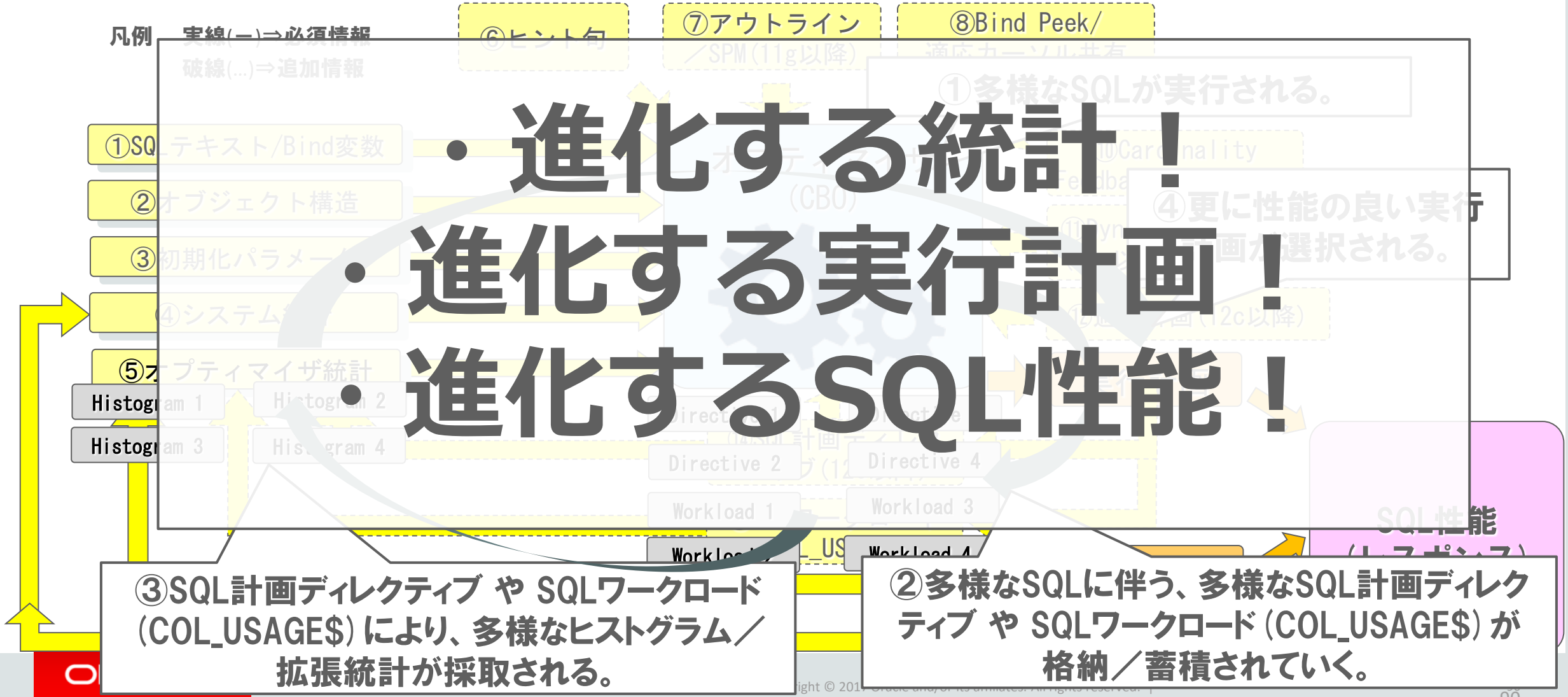
実行計画の各種最適化と最新化運用が組み合わさると…

- ①様々なSQLが実行 ⇒ ②多様なSQL計画ディレクティブ／SQLワークロードが蓄積
  ⇒ ③多様なヒストグラム／拡張統計が採取 ⇒ ④更に性能の良い実行計画が選択！



実行計画の各種最適化と最新化運用が組み合わさると…

- ①様々なSQLが実行 ⇒ ②多様なSQL計画ディレクティブ／SQLワークロードが蓄積
  ⇒ ③多様なヒストグラム／拡張統計が採取 ⇒ ④更に性能の良い実行計画が選択！



・進化する統計！
・進化する実行計画！
・進化するSQL性能！

本章(2章)のまとめ

- クエリー・オプティマイザの進化によって、昔よりも機能や運用の選択／組合せの幅は広がっています。
  - 安易な現行踏襲 や 旧バージョンの陳腐化した推奨に縛られていないでしょうか？
- 次章では、具体的な統計情報運用のデザインと実際の事例(良いパターン／アンチパターン)をご紹介したいと思います。

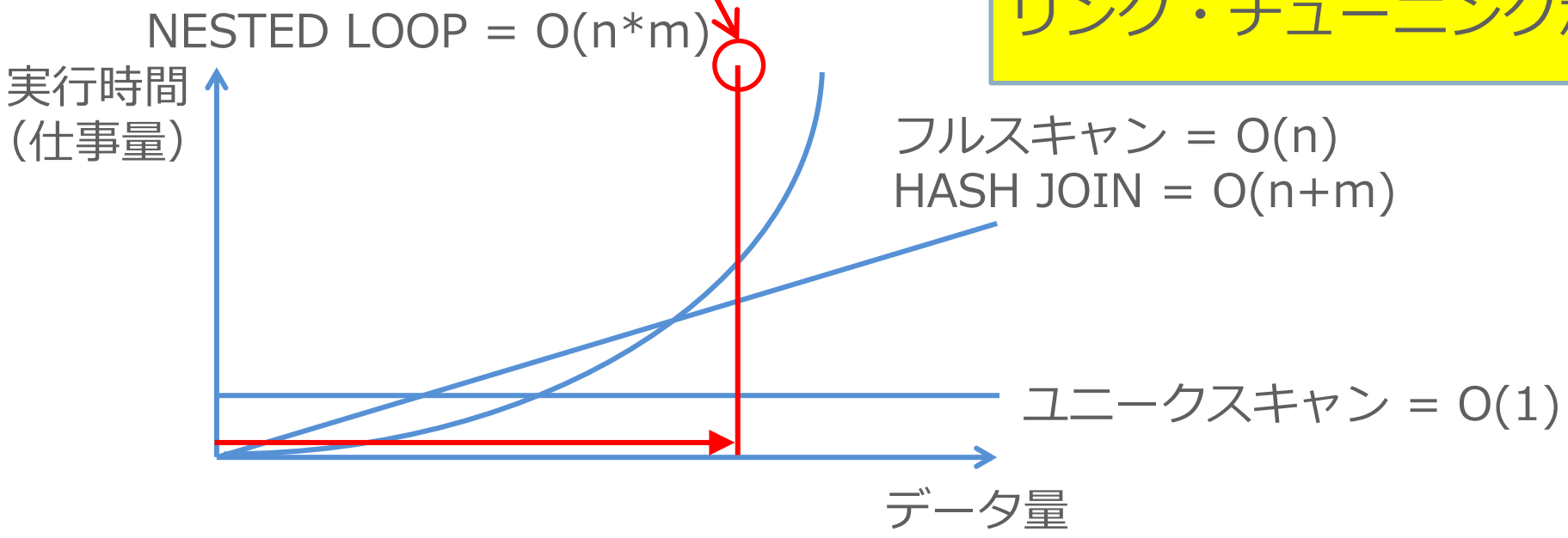
3章. 統計情報運用のデザイン と 事例紹介 (良いパターン／アンチパターン)

統計運用デザイン(固定化)

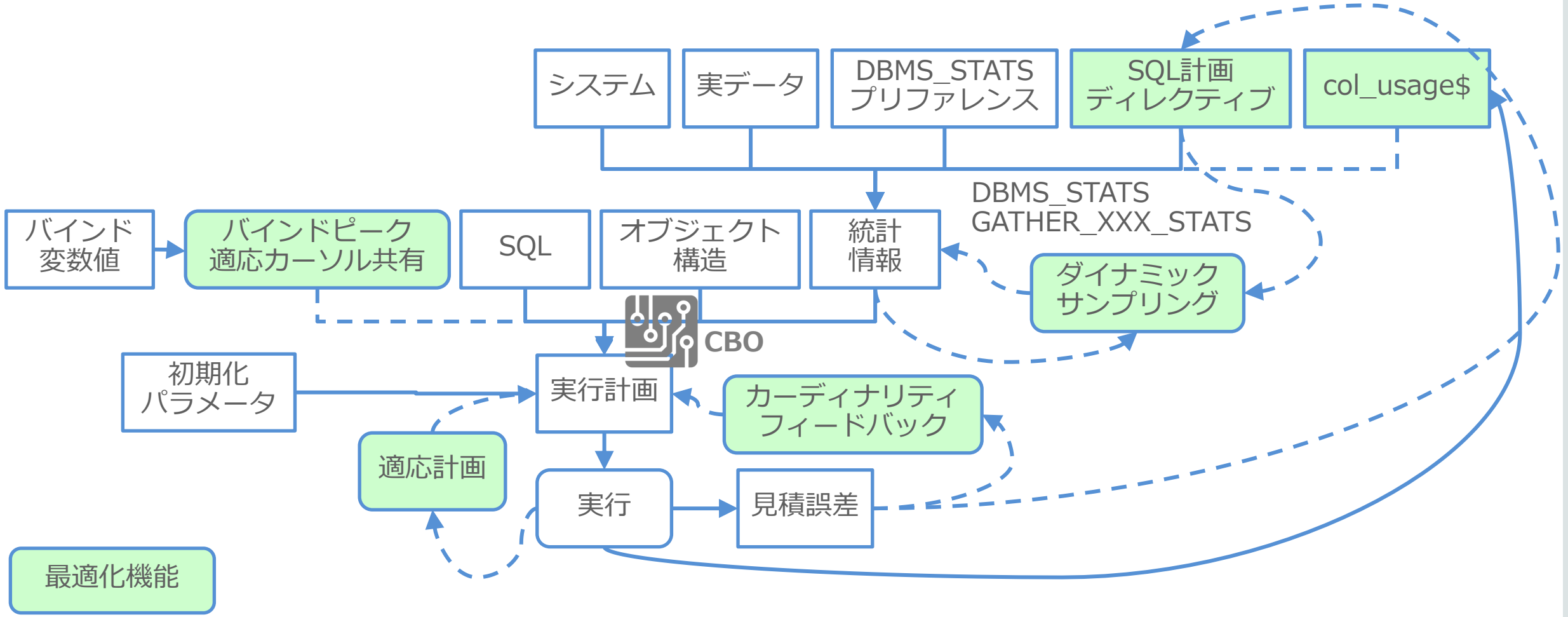
統計情報固定運用（実行計画固定）のポイント

1. 将来想定最大データ量・バリエーションで統計収集・固定
2. 最適化機能を無効化
3. SQL性能をモニタリング

1が完璧な場合は3は不要だが、想定外のデータ増加による性能劣化を防ぐためにはSQL性能のモニタリング・チューニングが必要



実行計画固定のため CBO へのインプットを固定



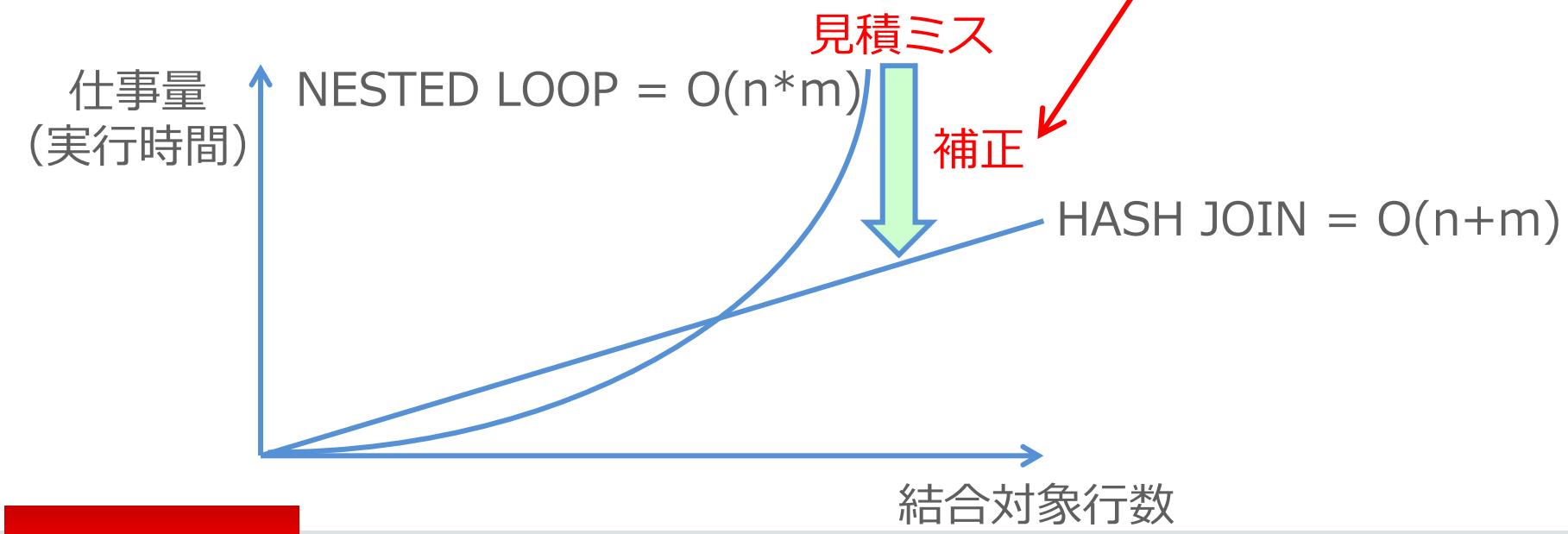
統計情報固定のつもりで実行計画が変動する要因

- 統計情報を収集せずに固定（デフォルト統計）
  - 表のサイズが変わるとカーディナリティ見積が変わる。(DocID:1714202.1参照)
- LOW_VALUE / HIGH_VALUE の範囲外の値が使用
  - 列統計の LOW_VALUE～HIGH_VALUE の範囲外の値がフィルタ条件で使用されるとカーディナリティ（行数）見積が小さくなる
  - SYSDATEなどはハードパース時に展開されリテラル同等の評価になる
- 最適化機能を無効化していない
  - バインドピーク/適応カーソル共有/統計フィードバック/動的統計/適応計画/SQL計画ディレクティブなどが有効なまま
- ヒストグラムを収集
  - 意味的に同じSQL（リテラル値のみ異なる）の実行計画を同一にしたい
    → ヒストグラムを取得しない
  - ヒストグラムを取得しない方法
    → DBMS_STATS の METHOD_OPT=>'FOR ALL COLUMNS SIZE 1'

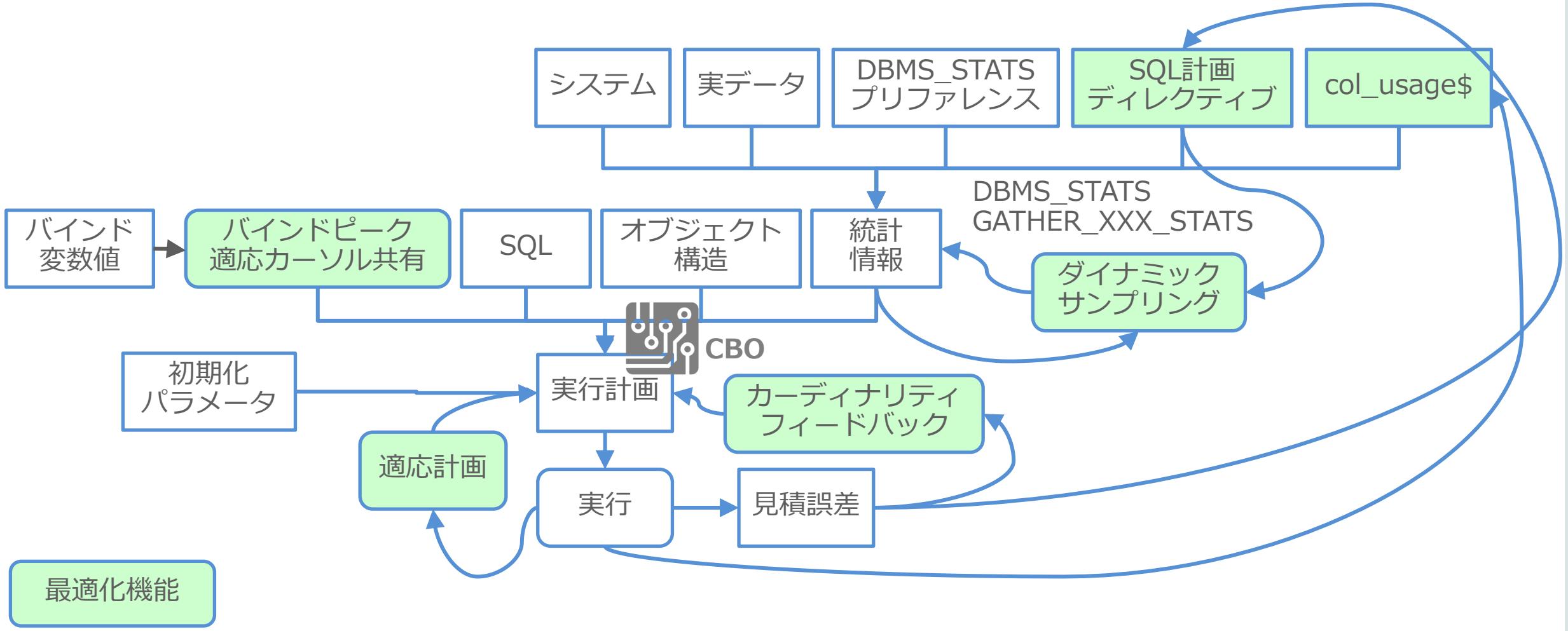
統計運用のデザイン(統計最新化+最適化有)

大量データ・複雑／Ad-HocなSQLに向いている

- DWH・BIの複雑なSQLはコスト見積が難しい
  - 集計などで大量行へアクセス
  - 結合数・フィルタ条件が多い
- 適応計画・SQL計画ディレクティブなどが活きる
  - 見積精度向上：ヒストグラム、拡張統計
  - 見積ミス補正：適応計画、SQL計画ディレクティブ

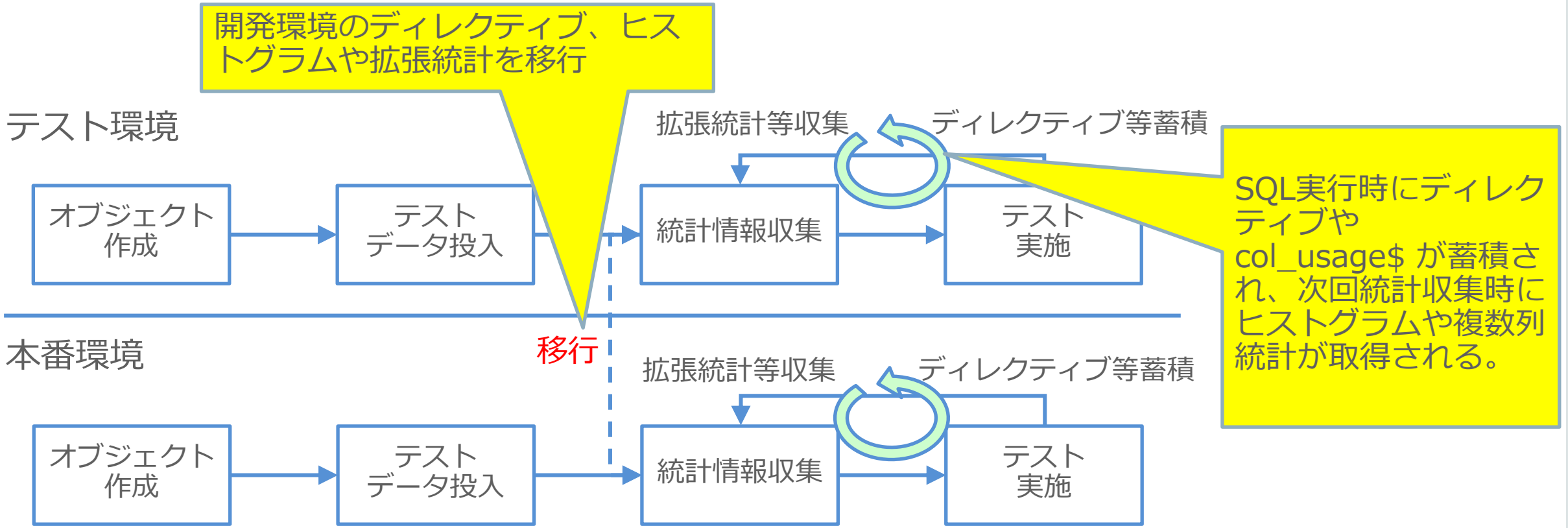


統計最新化+最適化有でCBOへのインプットを最大化



統計最新化+最適化有の運用の注意点

- 本番環境でワークロードが流れるまでテスト環境同等の性能にならない
- 「過去になぜその実行計画になったか」の原因追究・再現が困難（割切りが必要）



https://static.rainfocus.com/oracle/oow16/sess/1464801405362001BEoy/ppt/OOWUpgradingTo12cWithoutPain_final.pdf

OOW 2017 の Optimizer PM の Nigel のスライドより

Upgrading to Oracle Database 12c Without Pain

And How Oracle Database 12c Release 2 Optimizer Features Will Help

ORACLE OPEN WORLD

September 18–22, 2016
San Francisco

12.2 ではどうなるか？

Nigel Bayliss
Optimizer Product Manager
@vldbb
http://blogs.oracle.com/optimizer

Data Warehousing
Product Management Team

Maria Colgan
Master Product Manager

ORACLE

Oracle Database 12c Release 1

Controlling adaptive features

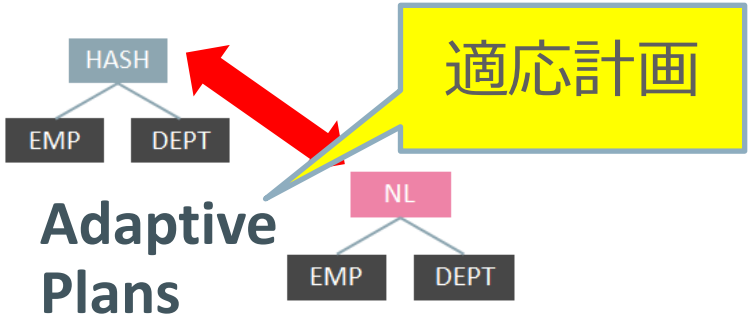
OPTIMIZER_ADAPTIVE_FEATURES

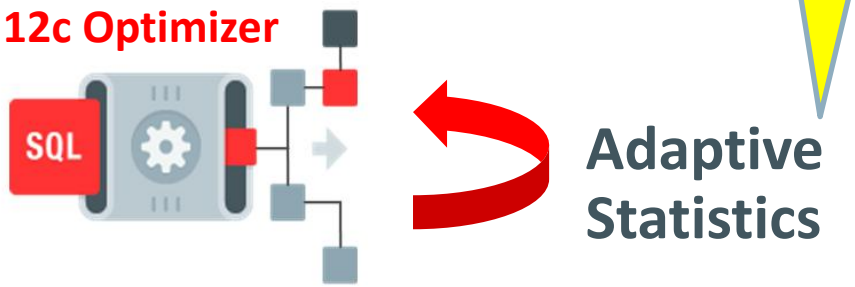
Optimizer Adaptive Features

- SQL計画ディレクティブ
- 動的統計
- 統計フィードバック

Change plans at runtime

Learn from previous executions





From Oracle Database 12c Release 2

NEW IN 12.2

Finer control of adaptive features – new database parameters

12.2では廃止され2つのパラメータに分割

~~OPTIMIZER_ADAPTIVE_FEATURES~~ Obsolete

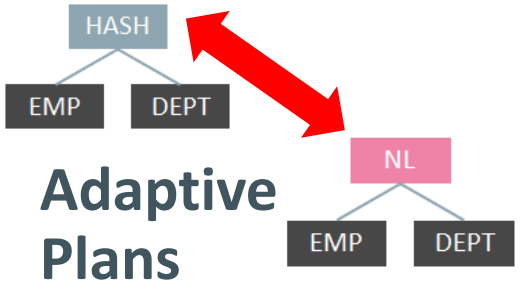
Optimizer Adaptive Features

OPTIMIZER_ADAPTIVE_PLANS

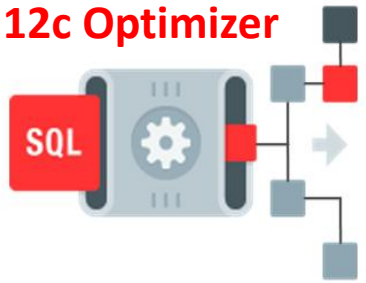
Change plans at runtime

OPTIMIZER_ADAPTIVE_STATISTICS

Learn from previous executions



Adaptive Plans

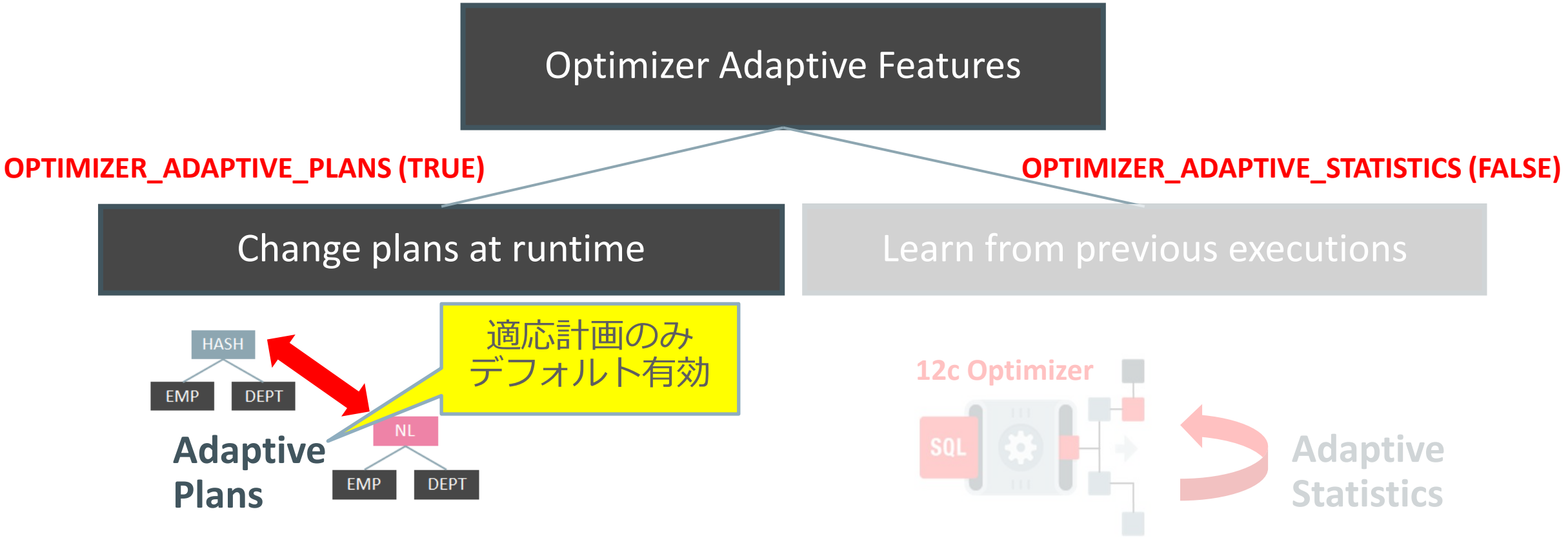


Adaptive Statistics

From Oracle Database 12c Release 2

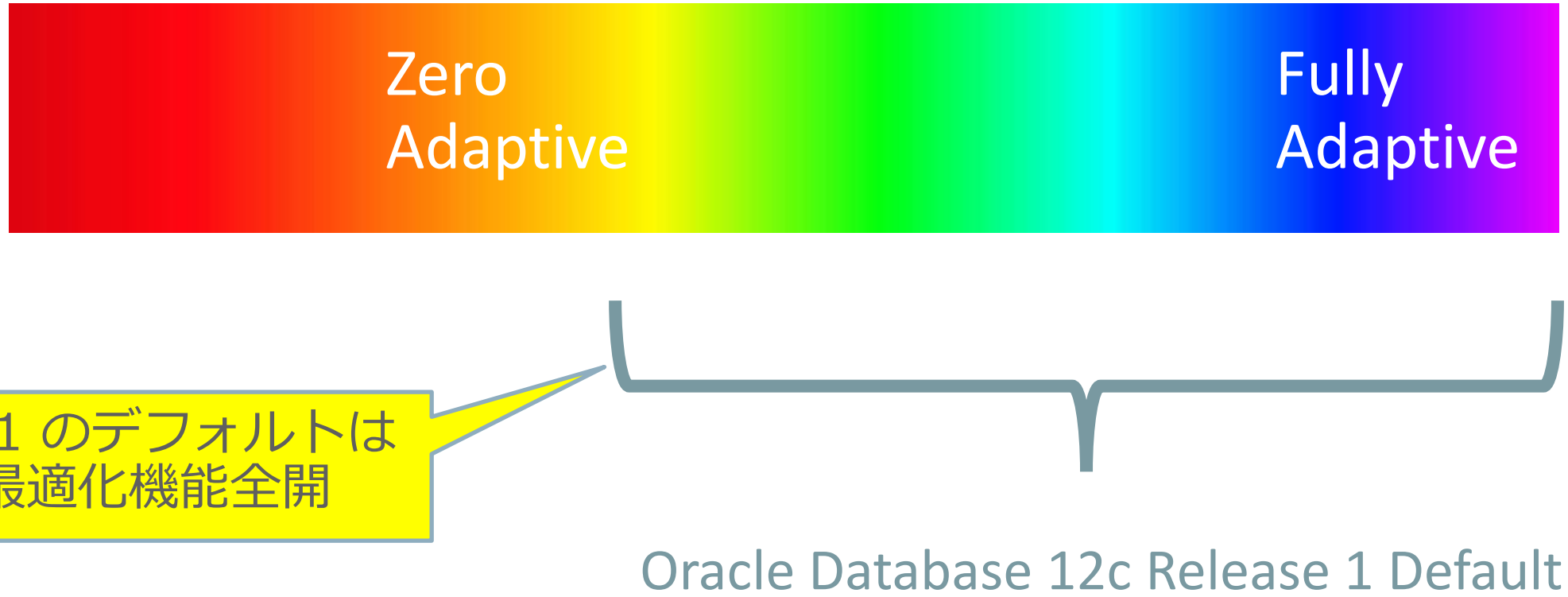
New default behavior

NEW IN 12.2



Oracle Database 12c Release 1

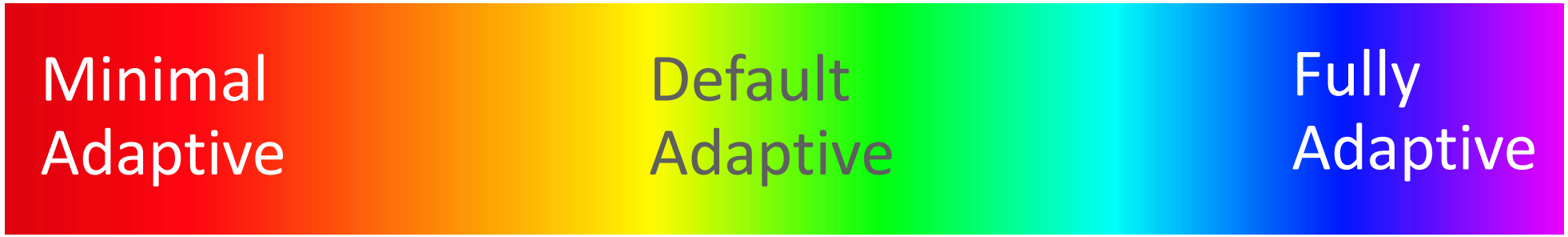
Adaptive features controlled with OPTIMIZER_ADAPTIVE_FEATURES



Oracle Database 12c Release 2

A new default and finer control

NEW IN 12.2

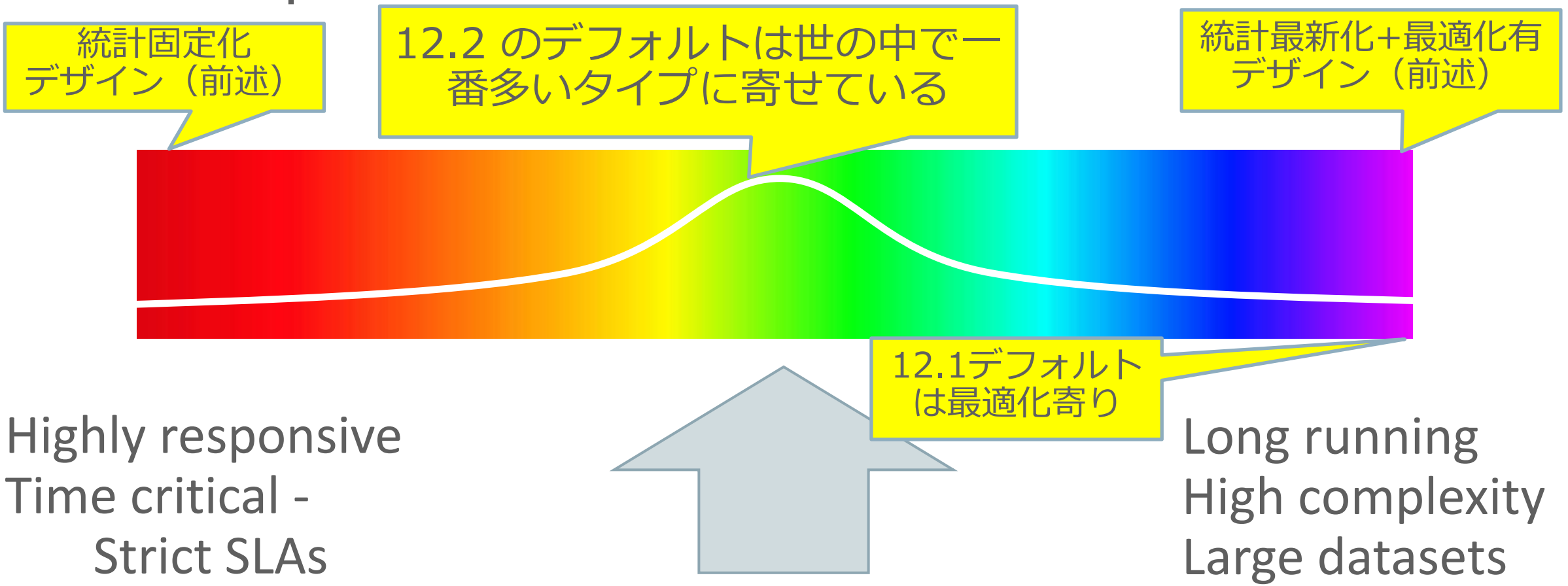


12.2 の
デフォルトは中間

Oracle Database 12c Release 2 Default

ORACLE

A Wide Spectrum of Oracle Databases

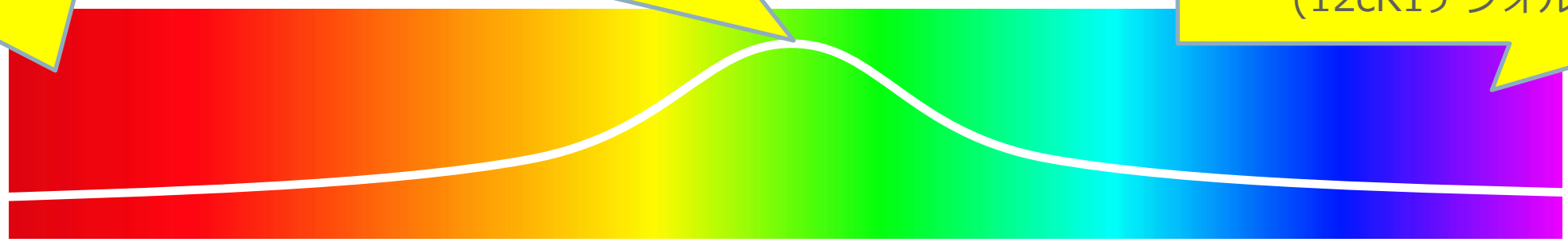


それぞれのモデルが適合するシステム

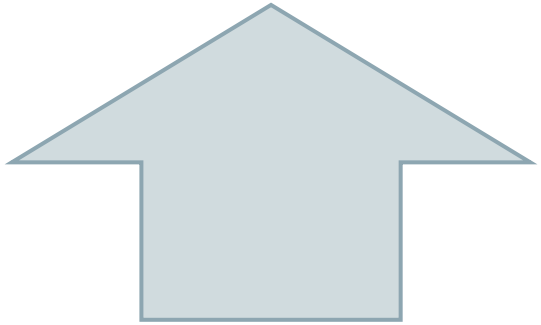
安定性重視(コンサバ)、ハイスキルなDBAが常駐、OLTP、ミッション・クリティカルなシステム

汎用的な動作、中庸、様々なシステムに適応
(12cR2デフォルト)

データ量が多い、結合表が多い、アドホックなクエリ、DWH＆情報系なシステム
(12cR1デフォルト)



Highly responsive
Time critical -
Strict SLAs



Long running
High complexity
Large datasets

Most Databases

Oracle Database 12c Release 1 - Options

- If you want the new adaptive parameters in Oracle Database 12c Release 1 request patch for bug#22652097
- To control auto column group creation using `DBMS_STATS` preference `AUTO_STAT_EXTENSIONS`, apply patch for bug#21171382
- Recommendations for Adaptive Features in Oracle Database 12c Release 1 (12.1) Doc ID 2187449.1

パッチで12.2同等の
動作に変更が可能

CBO機能のデフォルト動作まとめ(from Doc ID 2187449.1)

凡例 ◎…デフォルトで動作　○…明示指定で動作　－…機能無し

Adaptive Statistics

機能名	11gR1	11gR2	12cR1	12cR2
適応計画 (Adaptive Plan)	－	－	◎	◎
統計 (Cardinality) Feedback	－	◎	◎/○※2	○
SQL計画ディレクティブ	－	－	◎/○※2	○
動的統計 (Dynamic Sampling = 11)	－	－/○※1	○※2	○
拡張統計 (式統計/複数列統計)	○	○	◎/○※3	○※3

※1…動的統計は11.2.0.4以降で利用可能
※2…Patch 22652097 で 12cR2と同様の制御が可能
※3…Patch 21171382で12cR2と同様の制御が可能
12cR2以降は統計プリファレンスで表毎に制御

統計運用の事例紹介(アンチパターン)

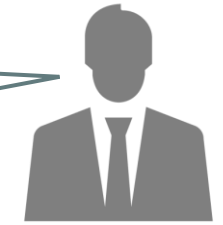
あるシステムでのSQL性能トラブルのやり取り

- 過去のあるシステムのSQL性能トラブル対応やり取り ※全て統計固定運用

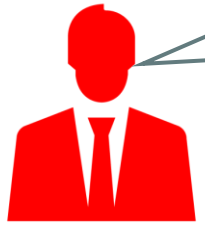
私（柴田）

MView実体表や索引の統計が
0件で実行計画が悪いです。

すみません。MView や 索引って
統計有るんでしたっけ？？？



ベンダA様



私（柴田）

エンドユーザ様が使う研修環境の統計が0件/Null
統計の嵐で、SQL性能が全く出ていません。

研修環境の統計は管理していません。研修環境の
データ量は少ないのに、変な実行計画を選ぶ
Oracle Database が悪いんじゃないですか？

ベンダB様

統計固定ではなく、統計が”放置”されている。

なぜこうなってしまうのか？それは……

- 性能劣化リスクでしか、統計運用を評価していないから。

No	統計運用／最適化機能の組合わせ	性能劣化リスク
①	固定化運用＋最適化無し	◎ 性能変動の要素がデータ増以外に無く、低リスク
②	最新化運用＋最適化有り(12c)	○ 12cの各種最適化機能により、リスクが更に低下
③	最新化運用＋最適化無し	× 最適化機能無しのため②より劣化リスクは高い

採用！

本来は様々な軸での評価が必要

- "SQL性能" "運用負荷" "性能劣化リスク" による 評価例(再掲載)

No	統計運用／最適化機能の組合わせ	SQL性能	運用負荷	性能劣化リスク
①	固定化運用＋最適化無し	✕ 精度の低い統計に加え最適化機能無し	✕ 新アプリや新環境リリース時のメンテナンス負荷	◎ 性能変動の要素がデータ増以外に無く、低リスク
②'	最新化運用＋最適化有り(12c)	◎ 新鮮で高精度な統計と最適化機能の組合せ	◎ 自動化運用を重要視したモデル	○ 12cの各種最適化機能により、リスクが更に低下
③	最新化運用＋最適化無し	△ 最適化機能無しのため②よりも性能は低い	◎ C/O後の運用負荷は②と同等	✕ 最適化機能無しのため②より劣化リスクは高い

①のモデルを上手く運用するために必要な要素

- ①の統計固定化するモデルを上手く運用するには、
  下記のような要素が必要になります。

Oracle Database
の 有識者

有識者を貼り付ける
体制／コスト

アプリ／インフラ
双方への周知＆習熟

統計を固定／管理／
運用する為の仕組み

これらの要素が欠けた、プアな運用だと……

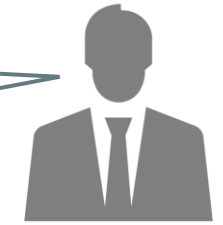
こうなります。(再掲)

- 過去のあるシステムのSQL性能トラブル対応やり取り ※全て統計固定運用

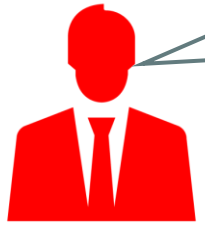
私（柴田）

MView実体表や索引の統計が
0件で実行計画が悪いです。

すみません。MView や 索引って
統計有るんでしたっけ？？？



ベンダA様



私（柴田）

エンドユーザ様が使う研修環境の統計が0件/Null
統計の嵐で、SQL性能が全く出ていません。

研修環境の統計は管理していません。研修環境の
データ量は少ないのに、変な実行計画を選ぶ
Oracle Database が悪いんじゃないですか？

ベンダB様

統計”放置”でリスクヘッジになっていない。

①は運用がプア(有識者不在／スキーム無し)だと…

- ①は運用がプア(有識者不在／スキーム無し)だと、悲惨！

No	統計運用／最適化機能の組合わせ	SQL性能	運用負荷	性能劣化リスク
①	固定化運用＋最適化無し	✕ 精度の低い統計に加え最適化機能無し	✕ 新アプリや新環境リリース時のメンテナンス負荷	✕ 0件/Null統計の頻発で、リスクヘッジにならない
②'	最新化運用＋最適化有	◎		◎ 種最適化機能クが更に低下
③	最新化運用＋最適化無し	△ 最適化機能無しのため②よりも性能は低い	◎ C/O後の運用負荷は②と同等	✕ 最適化機能無しのため②より劣化リスクは高い

運用がプアだと、全てが✕

このようなシステムは②'のモデルの方がフィット

- 運用がプアなシステムは、②'のモデルの方がフィットする。

No	統計運用／最適化機能の組合わせ	SQL性能	運用負荷	性能劣化リスク
①	固定化運用＋最適化無し	✕ 精度の低い統計に加え最適化機能無し	✕ 新アプリや新環境リリース時のメンテナンス負荷	✕ 0件/Null統計の頻発で、リスクヘッジにならない
②'	最新化運用＋最適化有り(12c)	◎ 新鮮で高精度な統計と最適化機能の組合せ	◎ 自動化運用を重要視したモデル	○ 12cの各種最適化機能により、リスクが更に低下
③	最新化運用＋最適化無	▲	◎	✕ のため は高い

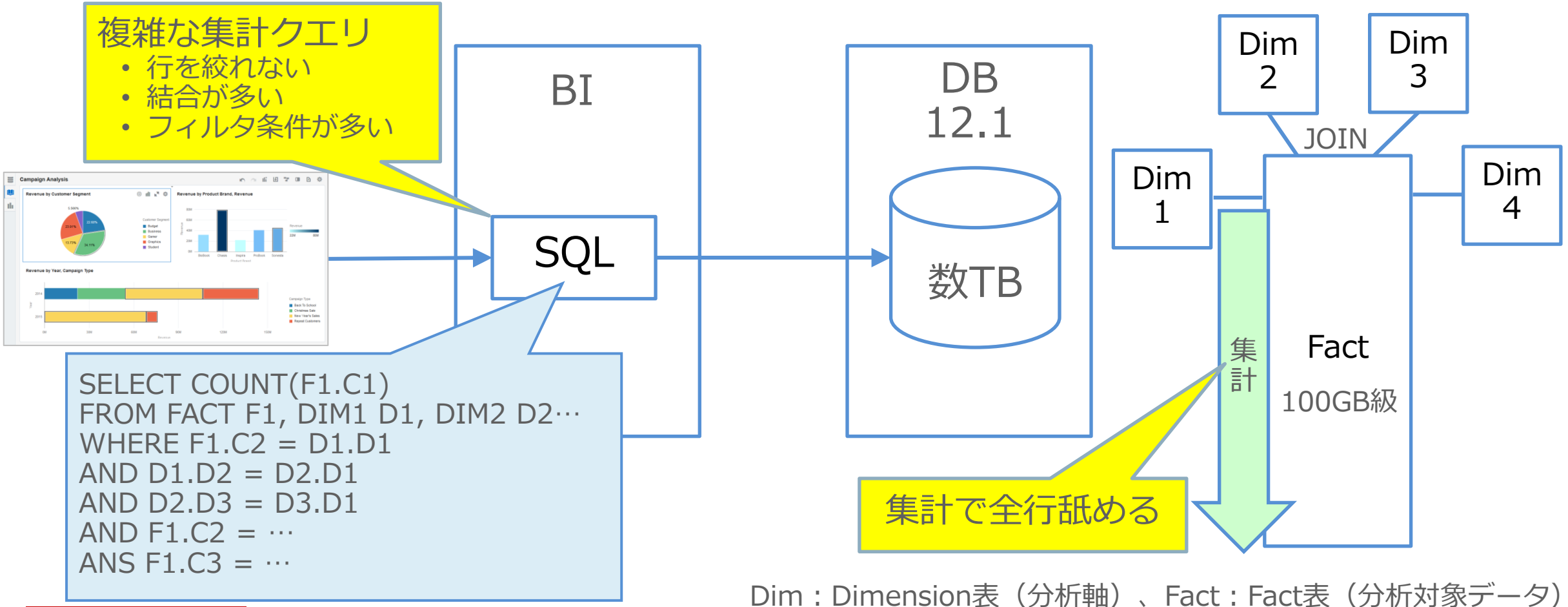
②'のモデルの方がフィット

（※性能劣化リスクをゼロには出来ない。）

統計運用の事例紹介(良いパターン)
統計最新化 ＋ 最適化有

複雑な集計クエリで100GB級の表をフルスキャン

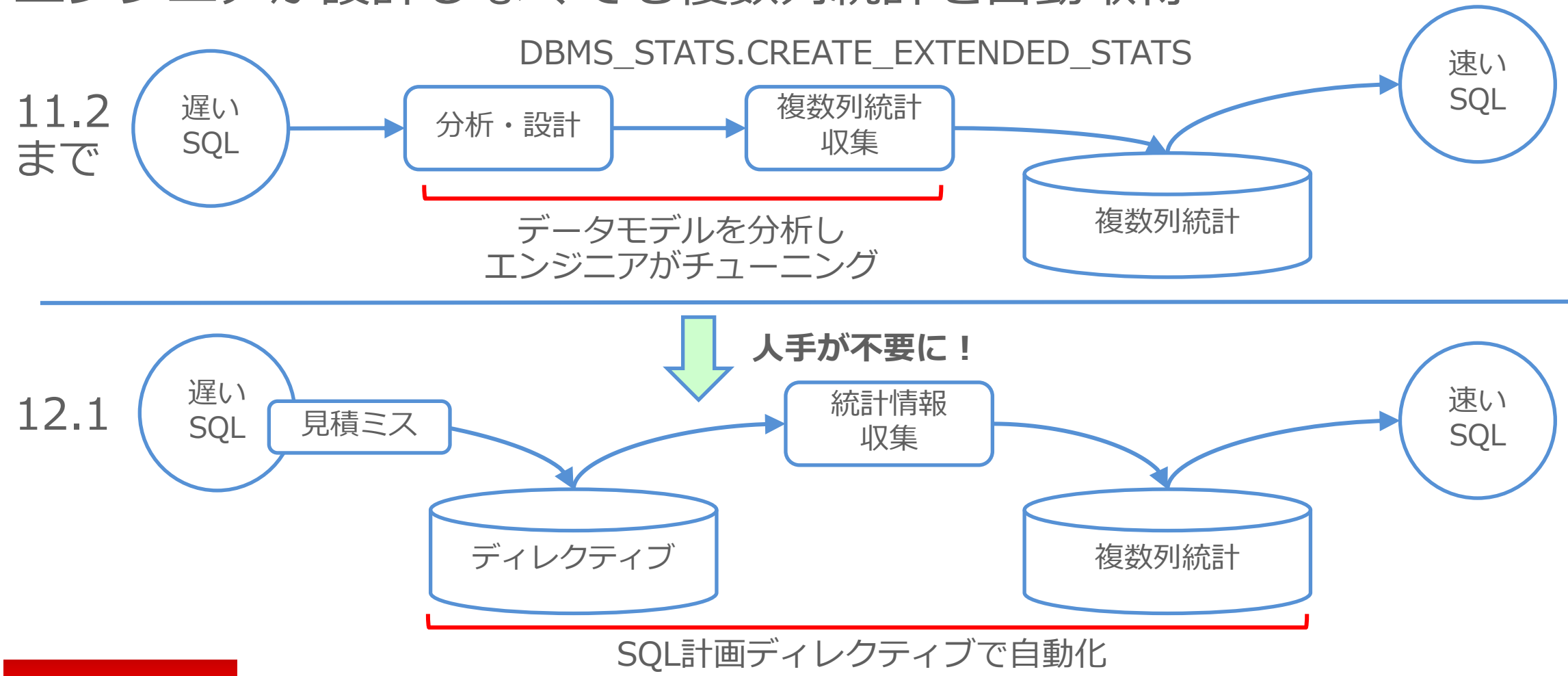
BIで複雑な集計クエリ



Dim：Dimension表（分析軸）、Fact：Fact表（分析対象データ）

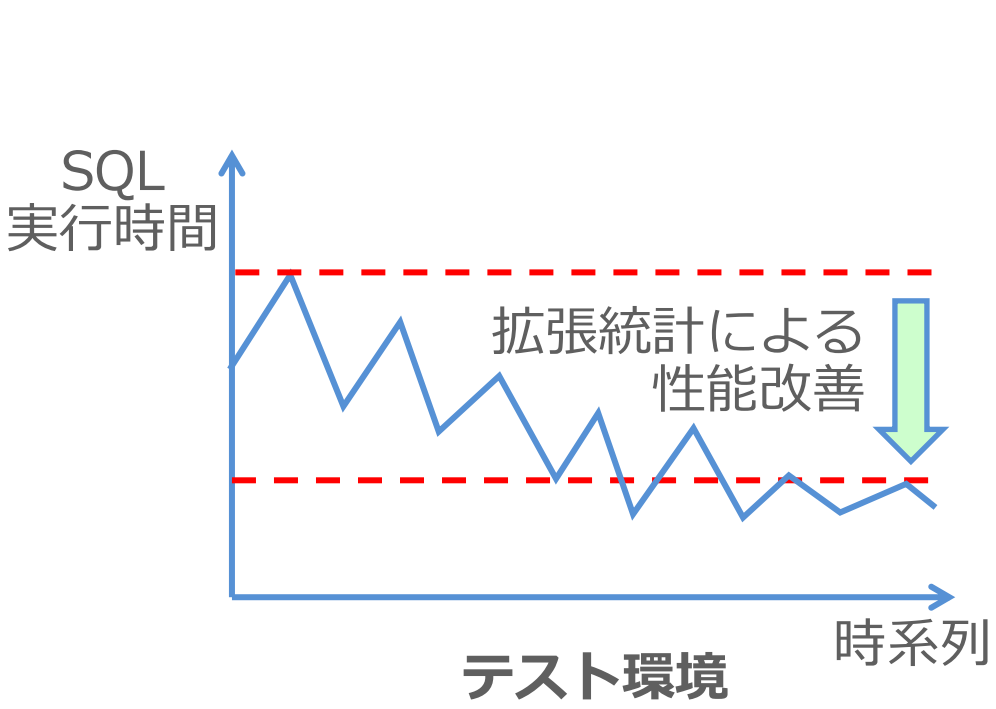
SQL計画ディレクティブでチューニング工数削減

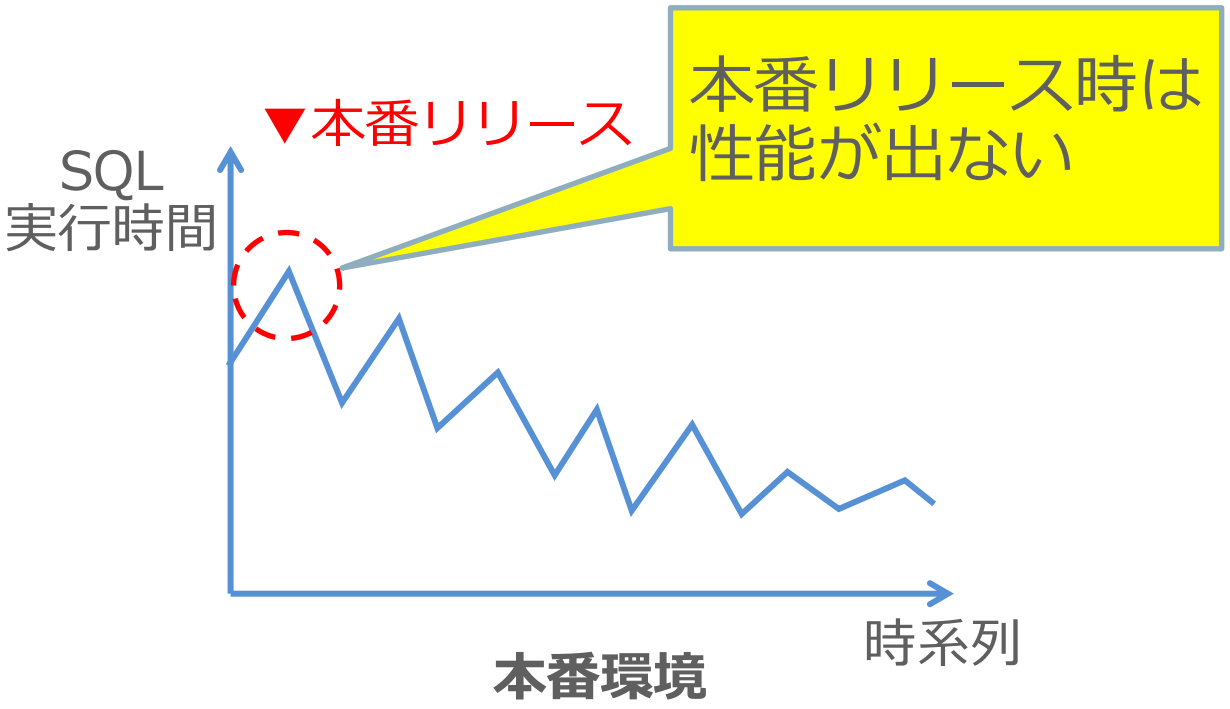
エンジニアが設計しなくても複数列統計を自動取得



テスト環境で速くても本番環境では

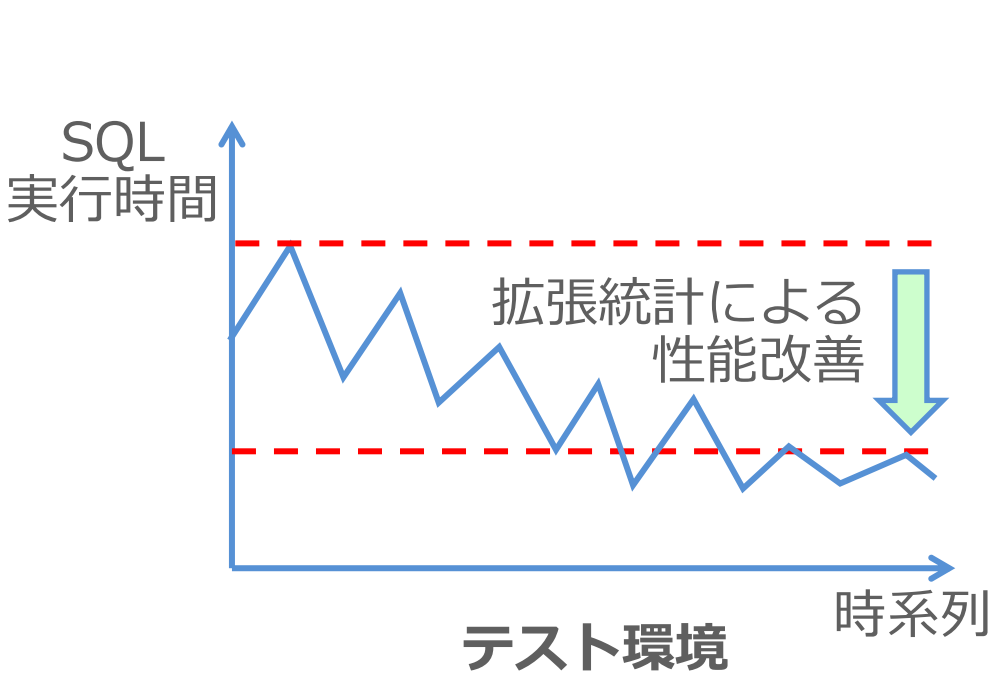
- テスト環境は学習機能で性能改善
- 本番リリース後、テスト環境相当の学習が進むまで性能が出ない





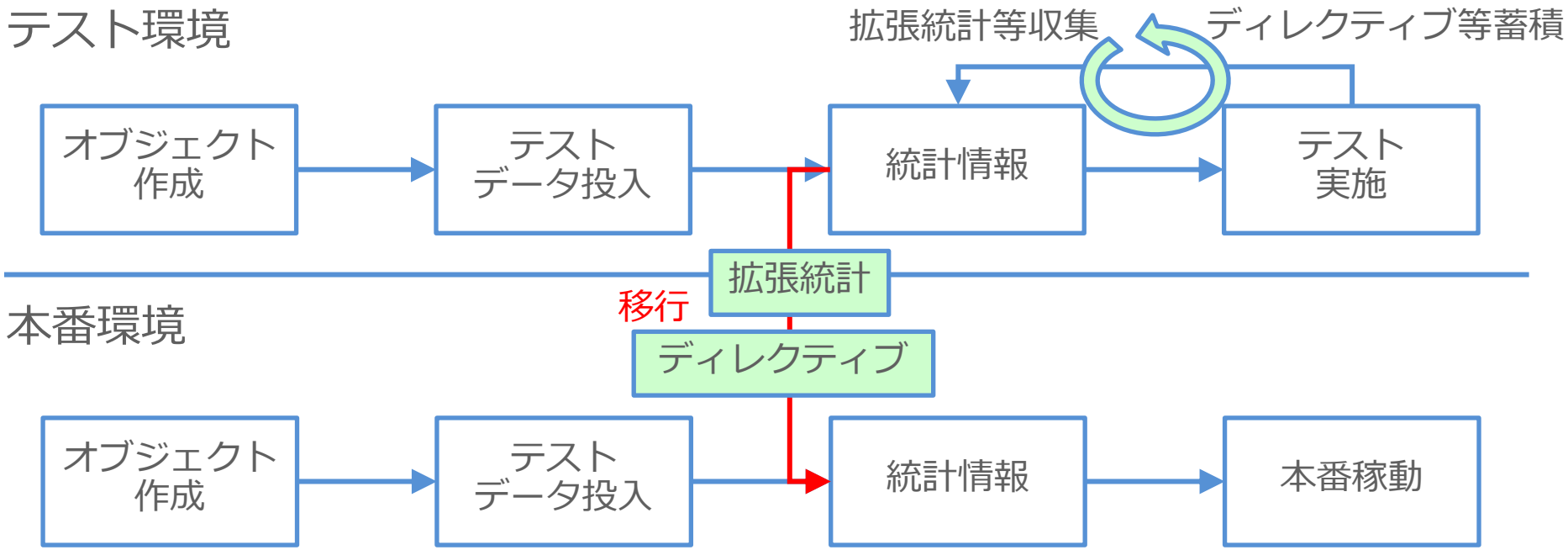
最初からテスト環境同等の性能を出すには

1. 本番環境でSQL実行して学習
   OR
2. テスト環境から本番環境に移行（ディレクティブ・拡張統計）



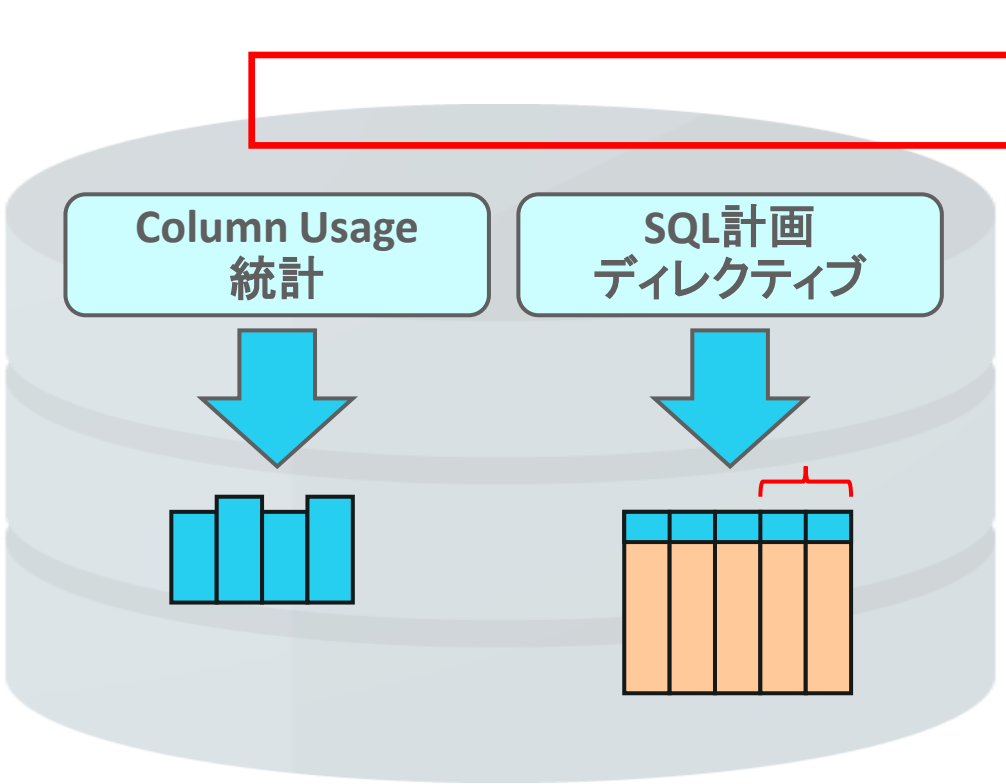
移行時の注意

- ディレクティブ移行時は拡張統計をセットで
  ディレクティブは複数列統計のステータスを持つため
- ディレクティブを移行しないと結合カーディナリティ補正されない
  結合カーディナリティ補正は拡張統計ではなく動的統計のみで補正されるため



[移行]拡張統計は Data Pump

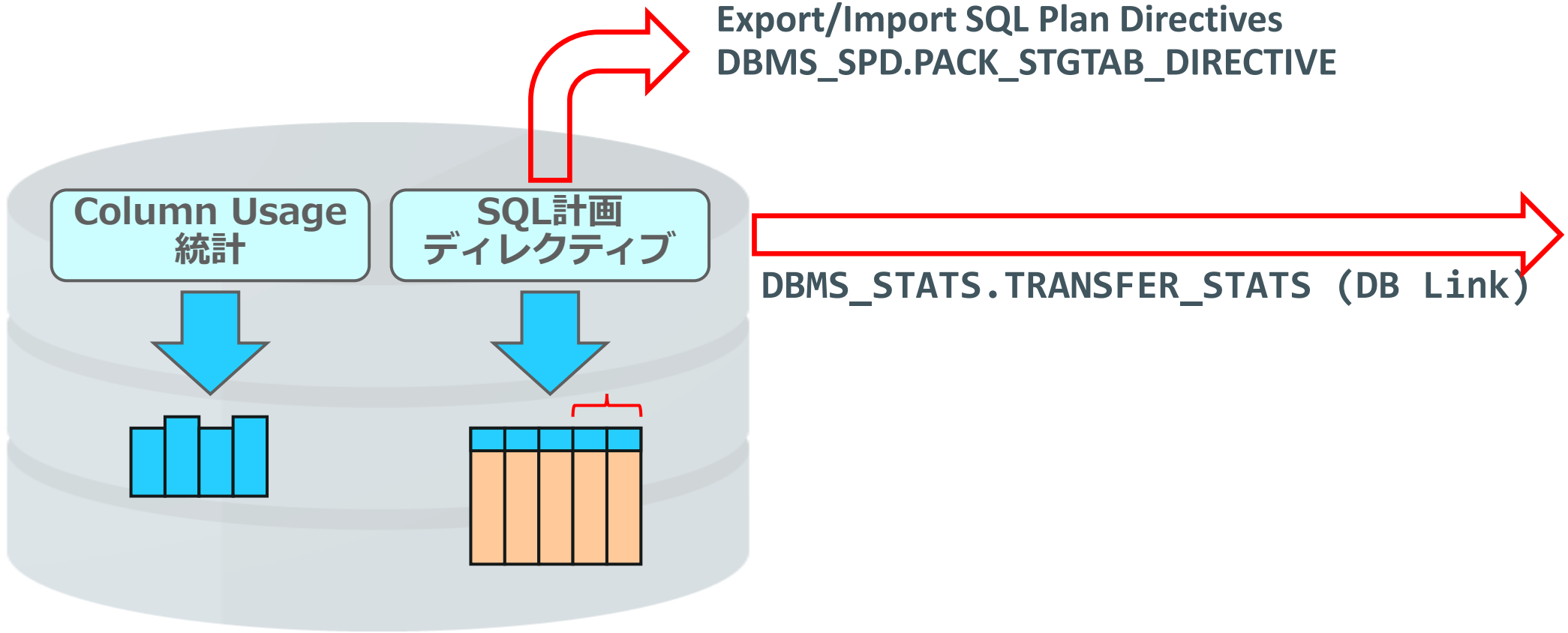
- 拡張統計は Data Pump で移行できる
- SQL計画ディレクティブは含まれない




```
expdp content=metadata_only
                include=statistics

impdp table_exists_action=skip
                include=statistics
                remap_schema=s1:s2
```

[移行]SQL計画ディレクティブはステージング表経由

SQL計画ディレクティブは DBMS_SPD パッケージを使用しステージング表経由で Data Pump Export/Import で移行

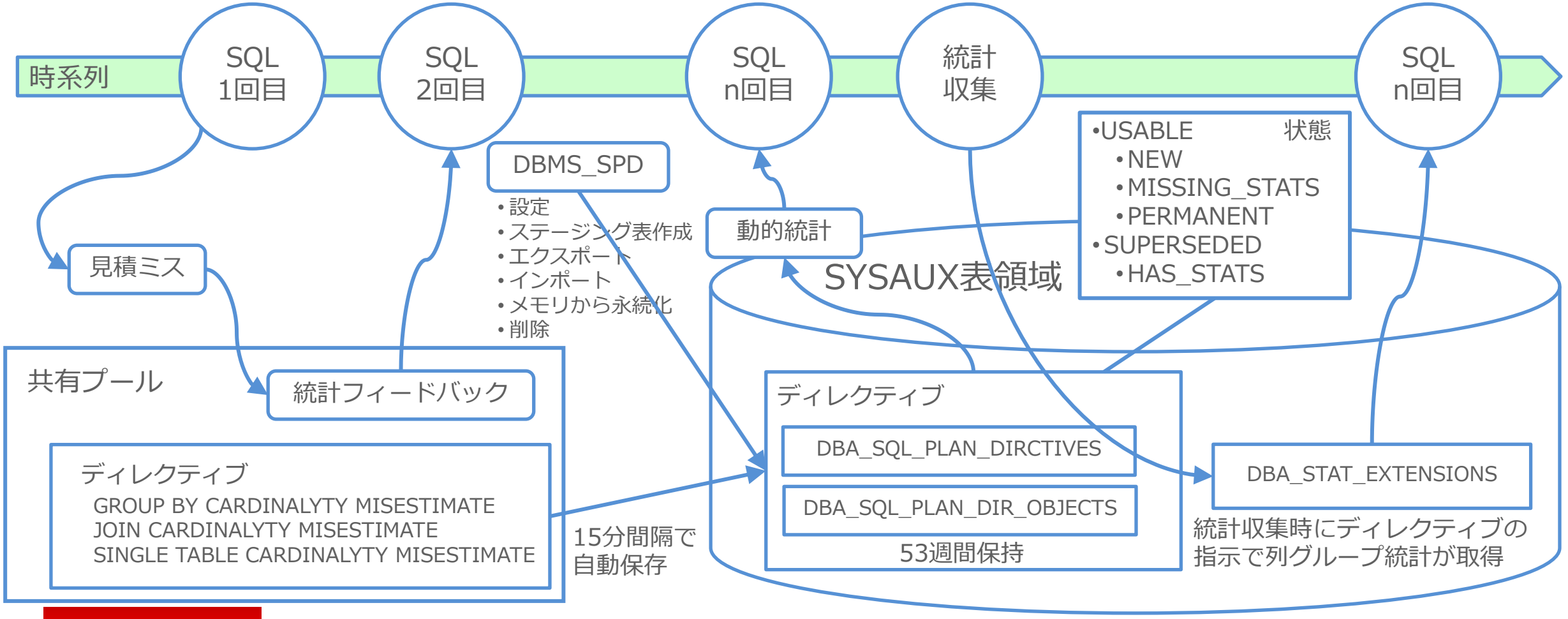


まとめ（良いパターン）… 統計最新化+最適化有

- 向いているシステム
  - 複雑なクエリで大量データにアクセス
- メリット
  - チューニング工数削減
- 注意事項（SQL計画ディレクティブ・拡張統計について）
  - ✓リリース時
    本番環境で、リリース直後にテスト環境同等の性能が要求される場合は本番でも学習させるか、テスト環境から移行する
  - ✓ストレージサイジング
    ディレクティブはSYSAUX表領域に53週分保持

（参考）SQL計画ディレクティブの時系列の動作

見積ミス→フィードバック→キャッシュアウト→動的統計→拡張統計取得



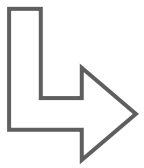
本章(3章)のまとめ

- まず重要視する要素が何なのか？を検討しましょう。
- その重要視する要素に沿った仕組みを作り込み、運用していく事こそが、統計運用の"戦略"そのものなのです。
  - アンチパターンは仕組み／運用を検討せず、放置して失敗している。
  - 良いパターンでは、環境／フェーズ毎の仕組みを作り上げて、Oracle Database 12c の機能／性能を上手く引き出している。

4章. まとめ

まとめ

- SQLにはアルゴリズムが予測で組み立てられると云う特徴があり、予測ゆえのハズレと常に隣合わせです。
- Oracle Database のオプティマイザは常に進化を続けており、その進化に伴って選択の幅は広がっています。
- 自分達のシステムで何を重要視するかを考えて、運用モデルの選択や仕組みを構築することが重要となります。



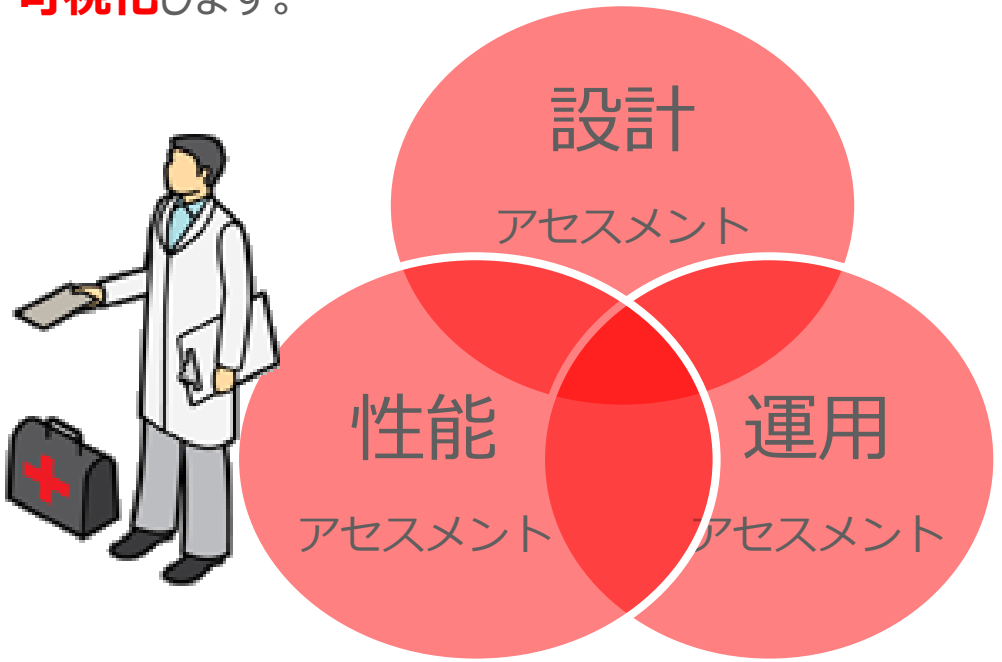
重要視する要素を見定めて、運用や体制、仕組みを作り込んでいく事が”戦略”です！

Appendix. 関連情報

Oracle Consulting Service による DBアセスメントサービスのご紹介

こんなお悩みはありませんか？

製品を熟知し、豊富な経験を持つOracleコンサルタントが**性能・設計・運用の3つの観点で、貴社システムの課題を可視化**します。



Oracle Consulting Service に 是非ご相談下さい！

表のオプティマイザ統計を参照可能なディクショナリ

- (DBA | ALL | USER)_TAB_STATISTICS で表のオプティマイザ統計を参照可能です。

```
SQL> DESC DBA_TAB_STATISTICS
 Name                          Null?    Type
 ----------------------------- -------- --------------
 OWNER                                  VARCHAR2(128)
 TABLE_NAME                             VARCHAR2(128)
 PARTITION_NAME                         VARCHAR2(128)
 PARTITION_POSITION                     NUMBER
 SUBPARTITION_NAME                      VARCHAR2(128)
 SUBPARTITION_POSITION                  NUMBER
 OBJECT_TYPE                            VARCHAR2(12)
 NUM_ROWS                               NUMBER
 BLOCKS                                 NUMBER
 EMPTY_BLOCKS                           NUMBER
 AVG_SPACE                              NUMBER
 CHAIN_CNT                              NUMBER
 AVG_ROW_LEN                            NUMBER
 AVG_SPACE_FREELIST_BLOCKS              NUMBER
 NUM_FREELIST_BLOCKS                    NUMBER
 AVG_CACHED_BLOCKS                      NUMBER
 AVG_CACHE_HIT_RATIO                    NUMBER
 SAMPLE_SIZE                            NUMBER
 LAST_ANALYZED                          DATE
 GLOBAL_STATS                           VARCHAR2(3)
 USER_STATS                             VARCHAR2(3)
 STATTYPE_LOCKED                        VARCHAR2(5)
 STALE_STATS                            VARCHAR2(3)
 SCOPE                                  VARCHAR2(7)
```

NUM_ROWS, LAST_ANALYZED 等で 統計の採取状況を確認

索引のオプティマイザ統計を参照可能なディクショナリ

- (DBA | ALL | USER)_IND_STATISTICS で索引のオプティマイザ統計を参照可能です。

```
SQL> DESC DBA_IND_STATISTICS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OWNER                                              VARCHAR2(128)
 INDEX_NAME                                         VARCHAR2(128)
 TABLE_OWNER                                        VARCHAR2(128)
 TABLE_NAME                                         VARCHAR2(128)
 PARTITION_NAME                                     VARCHAR2(128)
 PARTITION_POSITION                                 NUMBER
 SUBPARTITION_NAME                                  VARCHAR2(128)
 SUBPARTITION_POSITION                              NUMBER
 OBJECT_TYPE                                        VARCHAR2(12)
 BLEVEL                                             NUMBER
 LEAF_BLOCKS                                        NUMBER
 DISTINCT_KEYS                                      NUMBER
 AVG_LEAF_BLOCKS_PER_KEY                            NUMBER
 AVG_DATA_BLOCKS_PER_KEY                            NUMBER
 CLUSTERING_FACTOR                                  NUMBER
 NUM_ROWS                                           NUMBER
 AVG_CACHED_BLOCKS                                  NUMBER
 AVG_CACHE_HIT_RATIO                                NUMBER
 SAMPLE_SIZE                                        NUMBER
 LAST_ANALYZED                                      DATE
 GLOBAL_STATS                                       VARCHAR2(3)
 USER_STATS                                         VARCHAR2(3)
 STATTYPE_LOCKED                                    VARCHAR2(5)
 STALE_STATS                                        VARCHAR2(3)
 SCOPE                                              VARCHAR2(7)
```

DISTINCT_KEYS, NUM_ROWS, LAST_ANALYZED 等で 統計の採取状況を確認

列統計を参照可能なディクショナリ

- (DBA | ALL | USER)_TAB_COL_STATISTICS ,
  (DBA | ALL | USER)_PART_STATISTICS,
  (DBA | ALL | USRE)_SUBPART_STATISTICS
  で、列統計を参照できます。

```
SQL> DESC DBA_TAB_COL_STATISTICS
Name                  Null?    Type
 --------------- -------- ---------------------------
 OWNER                          VARCHAR2(128)
 TABLE_NAME                     VARCHAR2(128)
 COLUMN_NAME                    VARCHAR2(128)
 NUM_DISTINCT                   NUMBER
 LOW_VALUE                      RAW(1000)
 HIGH_VALUE                     RAW(1000)
 DENSITY                        NUMBER
 NUM_NULLS                      NUMBER
 NUM_BUCKETS                    NUMBER
 LAST_ANALYZED                  DATE
 SAMPLE_SIZE                    NUMBER
 GLOBAL_STATS                   VARCHAR2(3)
```

```
 USER_STATS                     VARCHAR2(3)
 NOTES                          VARCHAR2(63)
 AVG_COL_LEN                    NUMBER
 HISTOGRAM                      VARCHAR2(15)
 SCOPE                          VARCHAR2(7)
```

NUM_DISTINCT, LOW_VALUE, HIGH_VALUE等で列値の特性を、必要に応じて確認します。

本資料で紹介したオプティマイザ機能・一覧表(1)

機能名	対応パラメータ	関連ディクショナリ	参考ドキュメント
Bind Peek	"_optim_peek_user_binds"(FALSE で無効化)	V$SQL_BIND_CAPTURE, V$SQL_BIND_DATA など	ドキュメントID 1725161.1 ドキュメントID 1725161.1 など
適応カーソル共有(優れたカーソル共有)	"_optimizer_adaptive_cursor_sharing" (FALSE で無効化) ※Bind Peek無効化でも一緒に無効化される。	VSQL, VSQL_PLAN, V$SQL_SHARED_CURSOR など	ドキュメントID 1741388.1 ドキュメントID 2096561.1 マニュアル Oracle Database SQL チューニング・ガイド 12c リリース1 -適応カーソル共有- など
ヒストグラム	非公開	DBA_HISTOGRAMS, DBA_PART_HISTOGRAMS, DBA_SUBPART_HISTOGRAMS など	ドキュメントID 1708172.1 マニュアル Oracle Database SQL チューニング・ガイド 12c リリース1 -ヒストグラム- など

本資料で紹介したオプティマイザ機能・一覧表(2)

機能名	対応パラメータ	関連ディクショナリ	参考ドキュメント
拡張統計 (複数列統計／式統計)	非公開	DBA_STAT_EXTENSIONS, DBA_HISTOGRAMS, DBA_PART_HISTOGRAMS, DBA_SUBPART_HISTOGRAMS など	ドキュメントID 1743890.1 ドキュメントID 2110955.1 マニュアル Oracle Database SQL チューニング・ガイド 12c リリース1 -拡張統計の管理- など
SQL ワークロード	非公開	col_usage$, col_group_usage$	マニュアル Oracle Database SQL チューニング・ガイド 12c リリース1 - ヒストグラム - ヒストグラムが作成される場合- など
Dynamic Sampling／Dynamic Statistics (動的統計)	OPTIMIZER_DYNAMIC_SAMPLING (11 で 動的統計が有効化)	V$RESULT_CACHE_OBJECTS など(パラメータに 11 を セットして動的統計を有効化した場合)	ドキュメントID 2002108.1 マニュアル Oracle Database SQL チューニング・ガイド 12c リリース1 -動的統計の制御- など

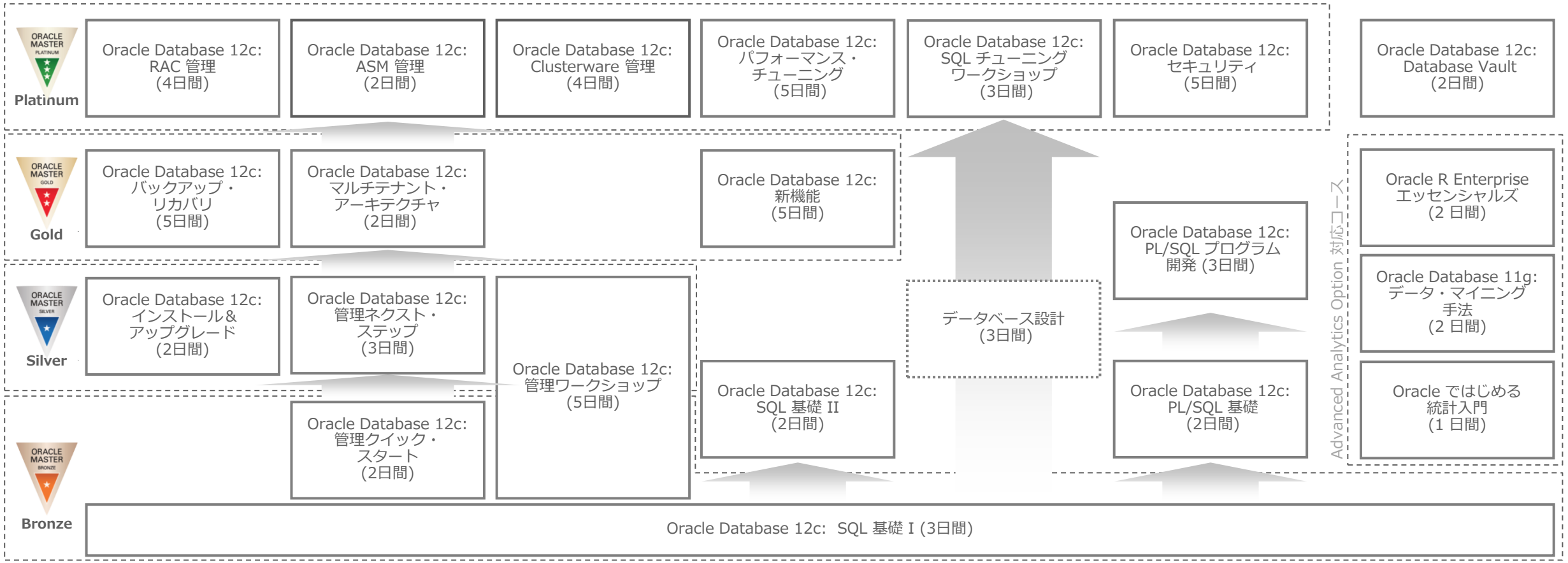
本資料で紹介したオプティマイザ機能・一覧表(3)

機能名	対応パラメータ	関連ディクショナリ	参考ドキュメント
Statistics (Cardinarity) Feedback	"_optimizer_use_feedback"(FALSE で無効化※)	VSQL, VSQL_PLAN, V$SQL_SHARED_CURSOR など	ドキュメントID 1344937.1 ドキュメントID 1752584.1 など マニュアル Oracle Database SQLチューニング・ガイド 12cリリース1 -再最適化: 統計フィードバック- など
SQL計画ディレクティブ	"_optimizer_dsdir_usage_control"(0 で無効化※)	DBA_SQL_PLAN_DIRECTIVES/DBA_SQL_PLAN_DIR_OBJECTS	ドキュメントID 2059121.1 マニュアル Oracle Database SQLチューニング・ガイド 12cリリース1 -SQL計画ディレクティブについて- など
Adaptive Plan (適応計画)	"_optimizer_adaptive_plans "(FALSE で無効化※)	VSQL, VSQL_PLAN など	ドキュメントID 1945816.1 マニュアル Oracle Database SQLチューニング・ガイド 12cリリース1 -適応計画- など

※本ページで提示した上記3つのオプティマイザ機能は、
OPTIMIZER_ADAPTIVE_FEATURES=FALSE でも無効化されます。
OPTIMIZER_ADAPTIVE_FEATURES の 12cR1 (12.1.0.2.0) のデフォルト値 は TRUE となります。

12c Oracle Database 12c 対応研修コースのご案内

基礎から上級スキルまで。Oracle Database 12c の製品機能を学習できる多彩な研修コースでスキルアップを



※ Oracle Database 12cR2 対応研修は順次提供予定です。詳しくはオラクルユニバーシティまでお問い合わせください。

オラクルユニバーシティ お問合わせ窓口
ORACLE UNIVERSITY
TEL 0120-155-092
URL http://www.oracle.com/jp/education/

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 / 13:00~18:00(祝日・年末年始休業日を除く)

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

ORACLE


ORACLE®