ORACLE

オラクル・コンサルが語る！ SQL 実行性能の安定化方式

日本オラクル株式会社
テクノロジーコンサルティング統括本部
テクニカルアーキテクト本部
DB コアテクノロジー部
プリンシパルコンサルタント
鈴木 健吾

#odddtky

日本オラクル、今年最大の技術トレーニング・イベント

Oracle DBA & Developer Day 2013

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。

Program Agenda

- 実行計画と SPM 概要
- プロジェクト事例
- SPM 機能詳細
- 運用における SPM 障害事例
- まとめ

本セミナーの内容に関する注意

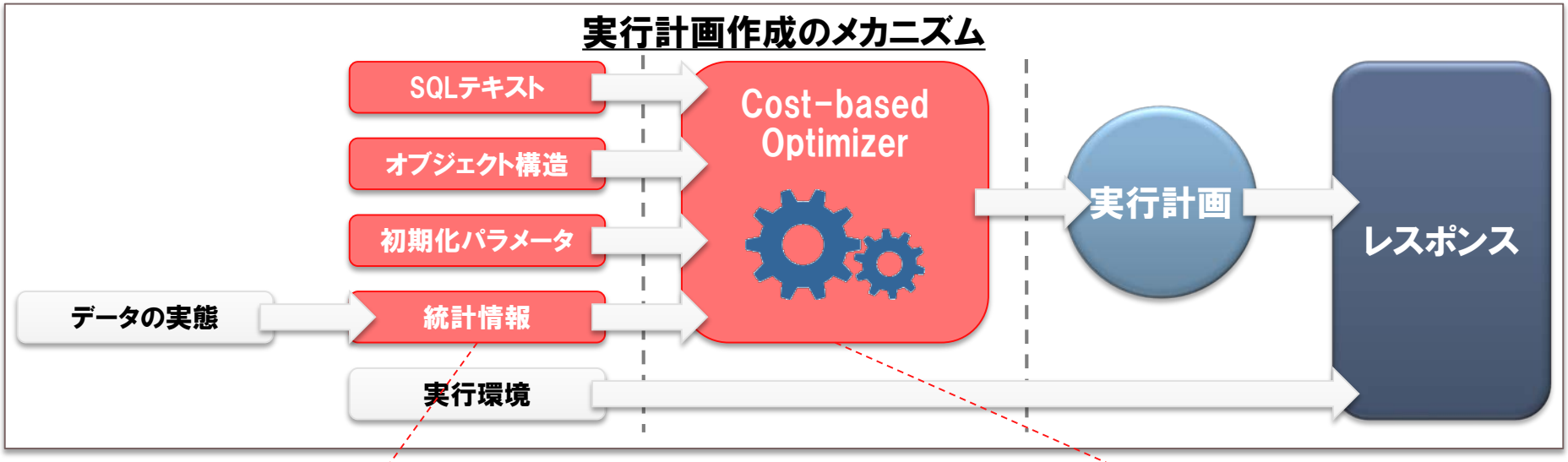
- 本資料に記載の機能詳細は、製品仕様に関する正式な情報ではなく、プロジェクトで調査をして把握した製品動作を記載しています。そのため、参考情報と考えてください。
- DB のバージョンアップを行う際に、必ずしも実行計画が変化して性能が低下するわけではありません。このプロジェクトではお客様との打ち合わせを重ねて行くなかで、アプリケーションの重要機能の性能低下が発生することを抑止したい、というご要望に対してプロアクティブに対応するため SPM を利用するという選択をしています。
- 製品紹介のセミナーではなく、プロジェクトで動作検証し利用した内容のフィードバックとなります。
- 製品仕様や基本的な使用方法は製品マニュアルを合わせてご活用ください。

  『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)』(B56312-04)

  - 「15 SQL計画の管理の使用方法」
- 実際に SPM を利用される際には、十分な技術検証を実施してください。

実行計画と SPM 概要

SQL 文の実行計画が決まる要因



統計情報が再収集される機能の例として以下があります。
- ✓自動オプティマイザ統計情報収集
- ✓手動統計情報収集 (DBMS_STATS, ANALYZE 文)
- ✓索引再作成
- ✓索引リビルド(再構成)

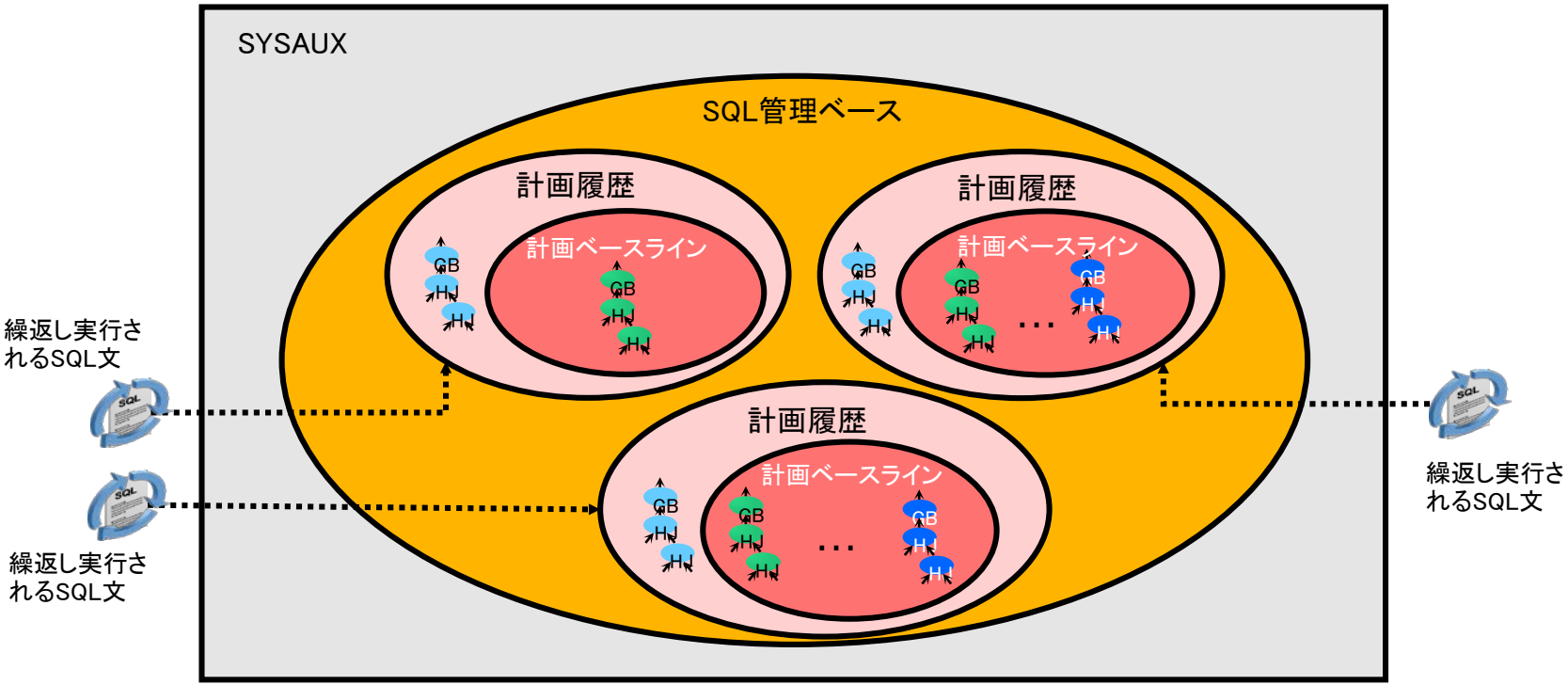
オプティマイザのバージョンによって、生成される実行計画が変化する可能性があります。また、オプティマイザは以下の機能により、動的に実行計画を変化させる可能性があります。
- ✓バインド・ピーク / 適応カーソル共有
- ✓カーディナリティ・フィードバック
- ✓統計情報の動的サンプリング

SPM とは

- SQL文の実行計画を時間の経過とともに記録し、評価を行う予防メカニズム
- データベースの変更にかかわらず、対応するSQL文のパフォーマンスを維持
- 承認されたSQL文の計画セット、SQL計画ベースラインを構築
- SQL計画ベースラインには承認された計画のみが含まれる
- 計画履歴は、時間の経過とともにSQL文に対して生成された承認済および未承認の計画のセット
- SQL計画ベースラインの改良フェーズでは、新しい計画のパフォーマンスを評価し、より優れたパフォーマンスの計画をSQL計画ベースラインに組み込む

『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)』(B56312-04)
「15 SQL計画の管理の使用方法」よりキーワードを抜粋

SPM のアーキテクチャ



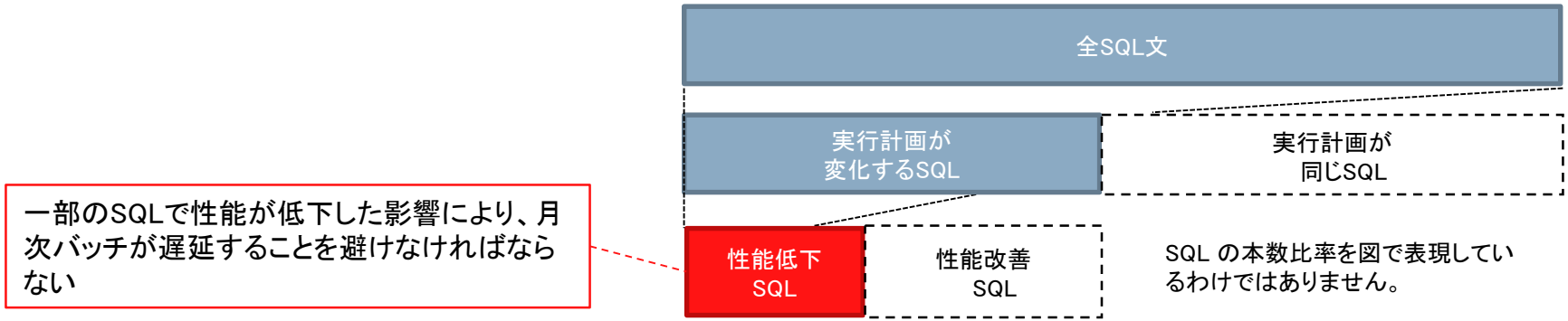
プロジェクト事例

プロジェクト概要

- 某社内基幹システム老朽化対応プロジェクト
  - DB を含むミドルウェアのアップグレードを伴う
- 現行システムの性能に関する問題点
  - 月次バッチ処理が長時間化（バッチ・ウインドウを超過）
  - DB 処理時間の 60% がディスク I/O の待機時間
  - 業務領域単位にディスク分散しており、特定の領域のバッチ処理は特定のディスクに I/O が集中
- 新システムの特徴
  - ASM を利用し全ディスクにデータを分散（全体最適化）
  - 業務データは全て EFD(SSD) を採用
- 現行システム・バージョン
  - Oracle E-Business Suite 11i(11.5.8) / Oracle9i Database Release 2(9.2.0.5) on Solaris
- 新システム・バージョン
  - Oracle E-Business Suite 11i(11.5.10.2) / Oracle Database 11g Release 2(11.2.0.2) on Linux x86(64bit)

SPM 利用の背景

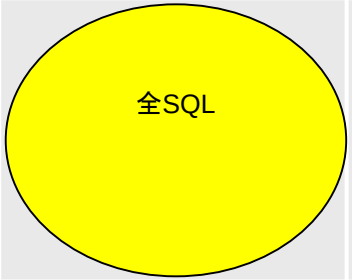
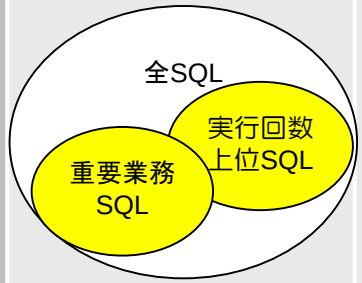
- SQL 文の性能低下による月次バッチ処理の遅延は避けなければならない。
- Oracle Database のバージョンが変わるため、同じ SQL 文でも実行計画は変化する可能性がある。（多くの SQL は性能が改善すると想定されるが、ごく一部の SQL で性能が低下する可能性がある）
- 複雑な SQL が多く、実行計画が変化したときの性能影響を事前に把握することが困難である。
- 新システムの I/O 性能は現行システムと比較して改善することが想定されるため、同じ実行計画であれば性能改善が見込める。



SPM 利用の目的

1. バージョン・アップにおいて、SQLの実行性能を低下させない。
   - 現行システムの重要 SQL の実行性能を低下させず、新システムへ移行する。
2. アプリケーションの改修なしで、SQLの実行性能を改善させる。
   - テスト期間中に SQL の性能低下を検出した場合、SPM の機能を使用して性能改善する。
   - 本番稼働後に SQL の性能低下を検出した場合、SPM の機能を使用して性能改善する。

SPM 利用範囲

	対象SQL	メリット	デメリット	備考
案1 **全ての SQL を管理対象とする**		全ての SQL の実行計画を現行システムと同一にすることで、カットオーバ後の性能問題の発生を抑止できる。	現行システムで流れている全 SQL を取得する必要があり、そのために現行システムの設定や運用方法を変更する必要がある。	**利用範囲を決定する設計フェーズで、SPM で多数の実行計画を管理した場合の性能影響や領域使用量が把握できなかった。**
案2 **管理対象とする SQL を選定する**		重要業務の SQL や実行回数の多い SQL の実行計画を現行システムと同一にすることで、カットオーバ後のクリティカルな性能問題の発生数を抑制できる。	SPM 管理対象外の SQL は性能問題が発生する可能性がある。	**重要業務はこのフェーズで既に選定されていた。**

プロジェクトでは案2 を選択

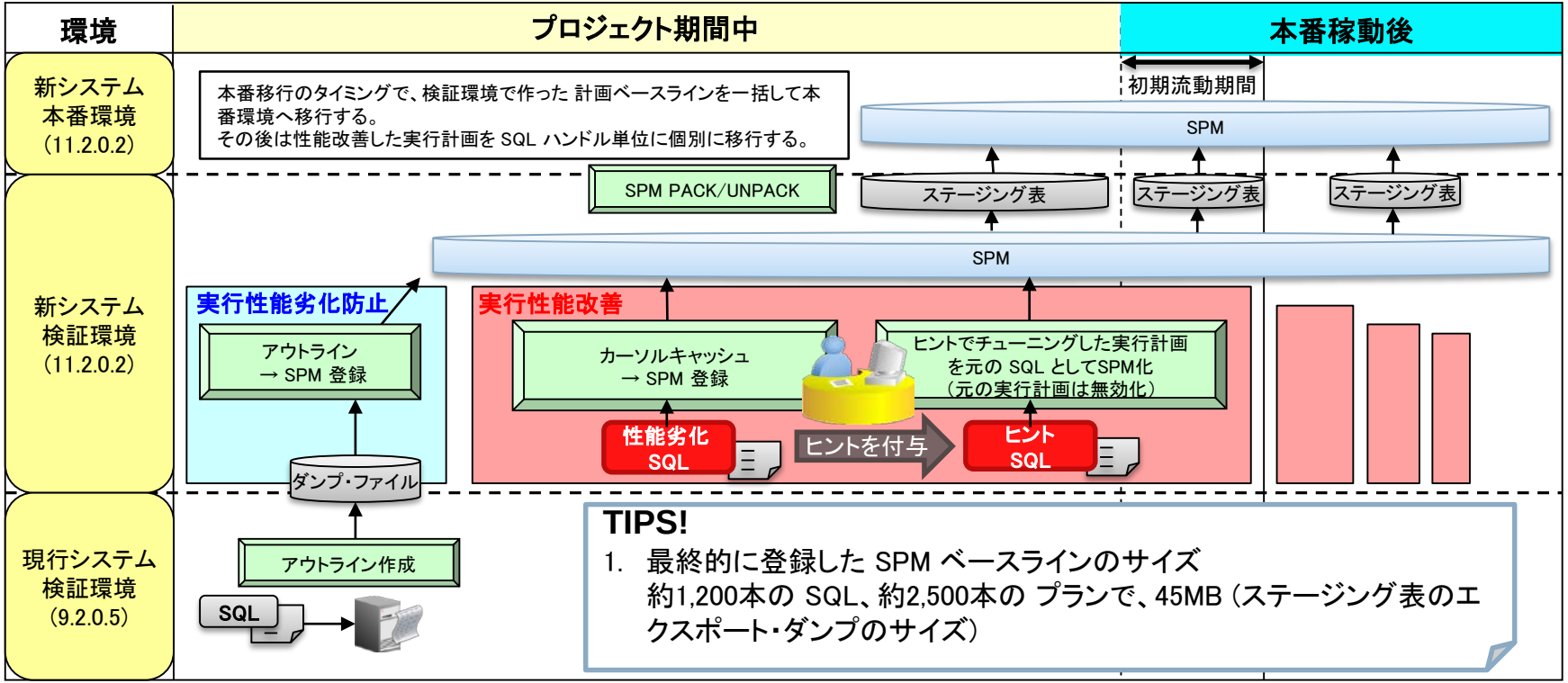
SPM 初期登録 SQL の選定

- 重要業務 SQL
  - 実行性能が低下した場合の業務的な影響が大きな機能をアプリケーション側の観点から選定
  - 夜間バッチのクリティカル・パスになっている機能や、バッチ・ウィンドウに必ず収めなければならない機能など
- 実行回数上位 SQL
  - SQL実行回数が多いほど、SQL 単体性能劣化の影響を大きく受けるため

「初期登録 SQL 」の本数と、「9i と同じ実行計画で 11g へ SPM 登録できた SQL」本数の割合（参考情報）

	初期登録 SQL		実行計画一致割合	
重要業務 SQL	約 200 本	約 600 本	約 80%	約 90%
実行回数上位 SQL	約 400 本		約 95%	

プロジェクトの各フェーズにおける SPM 運用



SPM を利用した性能改善方法例 ①

KROWN#134329 [11g新機能]ヒントを指定した際の実行計画を SQL計画の管理にロードする方法

作業内容	元の実行計画の登録(ベース登録)	ヒントを利用して実行計画を編集
SQL実行	`select * from tab2 where c2 > 999;`	`select /*+ FULL (TAB2) */ * from tab2 where c2 > 999;`
カーソルのSQL_ID, PHVを確認(V$SQL)	SQL_ID PLAN_HASH_VALUE 3m7d74pkw543z 2200541503	SQL_ID PLAN_HASH_VALUE 1j82gtnc0n4ps 2156729920
カーソルの実行計画を確認(dbms_xplan.display_cursor)	Plan hash value: 2200541503 | Id | Operation | Name | | 0 | SELECT STATEMENT | | | 1 | TABLE ACCESS BY INDEX ROWID | TAB2 | |* 2 | INDEX RANGE SCAN | IND2 |	Plan hash value: 2156729920 | Id | Operation | Name | | 0 | SELECT STATEMENT | | |* 1 | TABLE ACCESS FULL | TAB2 |

カーソルから SPM 登録	`var res number`
	`exec :res := dbms_spm.load_plans_from_cursor_cache( -`
	`sql_id => '3m7d74pkw543z', -`
	`plan_hash_value => 2200541503);`
		`var res number`
		`exec :res := dbms_spm.load_plans_from_cursor_cache( -`
		`sql_id => '1j82gtnc0n4ps', -`
		`plan_hash_value => 2156729920, -`
		`sql_handle => 'SYS_SQL_2506061412f95de3');`

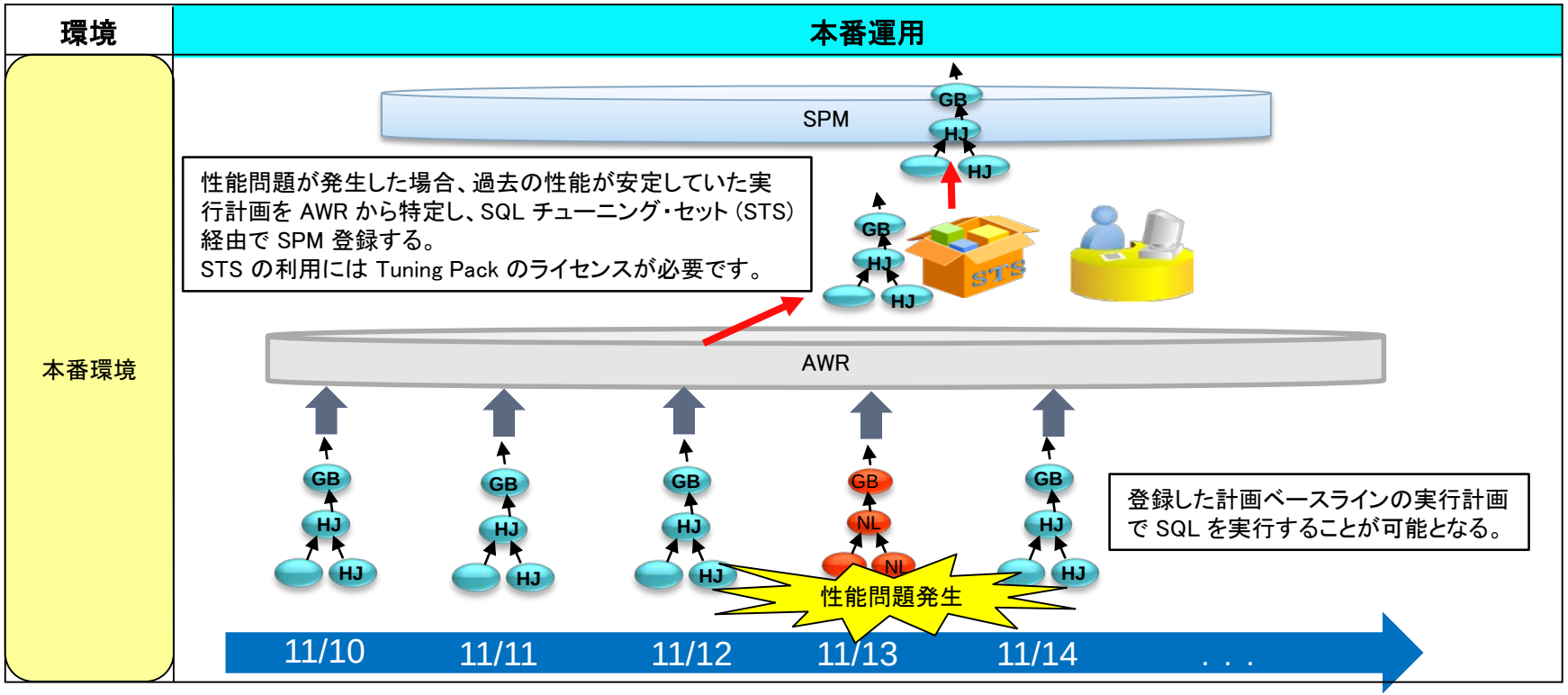
SPM 登録

元の SQL_HANDLE に、ヒントで編集した実行計画を結び付けることで、
元の SQL の実行計画を変えることが可能
元の実行計画を今後使わないように、無効化 or 削除を行う

SPM 登録

SQL_HANDLE	PLAN_NAME	ORIGIN	ENA	ACC	FIX	AUT	LAST_EXECUTED
SQL_2506061412f95de3	SQL_PLAN_2a1h62h9gkrg371befc37	MANUAL-LOAD	YES	YES	NO	YES	09/06/03 08:35:32
SQL_2506061412f95de3	SYS_SQL_PLAN_12f95de3b157e6ce	MANUAL-LOAD	YES	YES	NO	YES	09/06/03 08:39:06

SPM を利用した性能改善方法例 ②



運用における DESCRIPTION 列の活用

TIPS!

- DBA_SQL_PLAN_BASELINES.DESCRIPTION 列を活用し、台帳としての利用が可能 (11.2.0.3 ～)
- DBA_SQL_PLAN_BASELINES.DESCRIPTION 列に 500 バイトまでの任意の文字列を格納することが可能
- フォーマットを決めておき、後から参照したときに登録した目的を把握できるようにする
- 例) <MODULE>, <VERSION>, <PURPOSE>, <DATE>

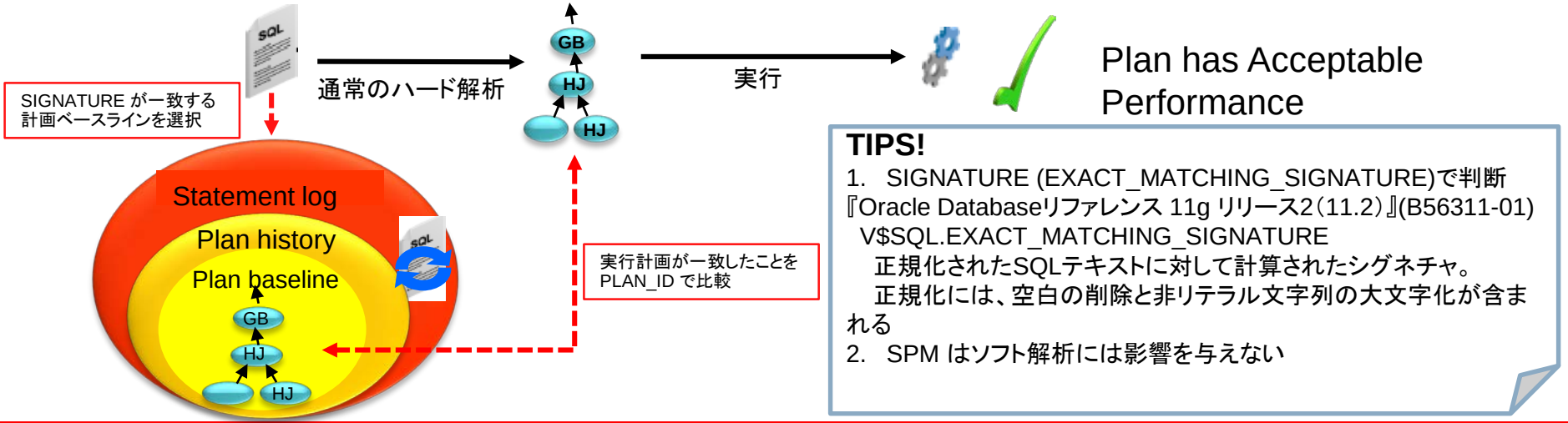
```
declare
  ret number;
begin
  ret := DBMS_SPM.ALTER_SQL_PLAN_BASELINE(
             sql_handle => 'SQL_50541272d7b8921d',
             plan_name => 'SQL_PLAN_50p0kfbbvj4hxa59ae91f',
             attribute_name => 'DESCRIPTION',
             attribute_value => 'MODULE03, 11.2.0.3, PT-011, 2013/11/14');
  DBMS_OUTPUT.PUT_LINE('RETURN CODE = ' || ret);
end;
/
```

SQL_HANDLE	PLAN_NAME	DESCRIPTION
SQL_07670b0d6bcca0a6	SQL_PLAN_0ftsb3ppwt856a59ae90f	MODULE01, 9.2.0.5, MIGRATE, 2013/07/04
SQL_4942d06e8207e3e4	SQL_PLAN_4khqhdu00gsz4bad3e404	MODULE02, 11.2.0.3, PT-010, 2013/11/14
SQL_a9478876f5f75f24	SQL_PLAN_akjw8fvuzfrt49a5b843c	MODULE03, 11.2.0.3, PT-011, 2013/11/14

SPM 機能詳細

ハード解析フェーズの流れ ①

- 実行する SQL 文と SIGNATURE が一致する計画ベースラインを選択
- 計画ベースラインが存在する場合でも、最初にオプティマイザによる通常のハード解析を実施
- ハード解析結果の実行計画と、計画ベースラインの実行計画を PLAN_ID で比較
- PLAN_ID が一致した場合は、ハード解析結果の実行計画で SQL 文を実行

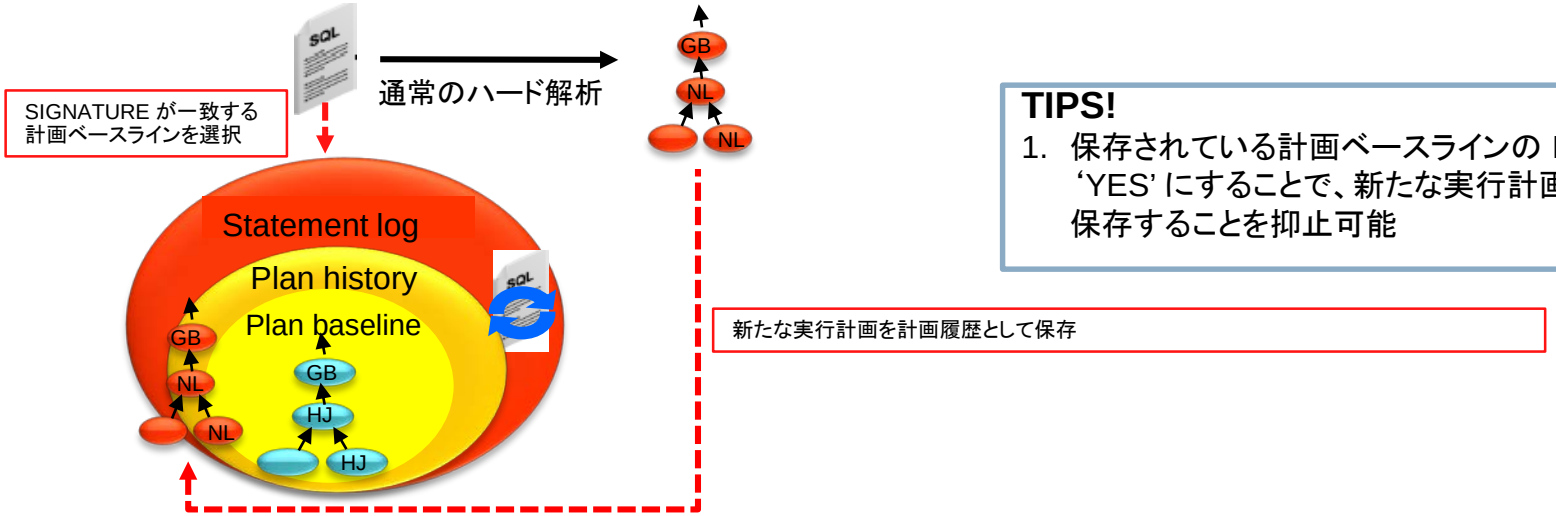


TIPS!

1. SIGNATURE (EXACT_MATCHING_SIGNATURE)で判断
『Oracle Databaseリファレンス 11g リリース2（11.2）』(B56311-01)
V$SQL.EXACT_MATCHING_SIGNATURE
正規化されたSQLテキストに対して計算されたシグネチャ。
正規化には、空白の削除と非リテラル文字列の大文字化が含まれる
2. SPM はソフト解析には影響を与えない

ハード解析フェーズの流れ ② - 1 (① で PLAN_ID が一致しない場合)

- 環境の変化（統計情報の再収集/オブジェクトの変化など）により、オプティマイザの通常のハード解析で生成される実行計画は変化
- 計画ベースラインと異なる実行計画が生成されると、その実行計画を計画履歴として保存
- 新たな実行計画は、ベースラインではないため、SQL 実行には不採用

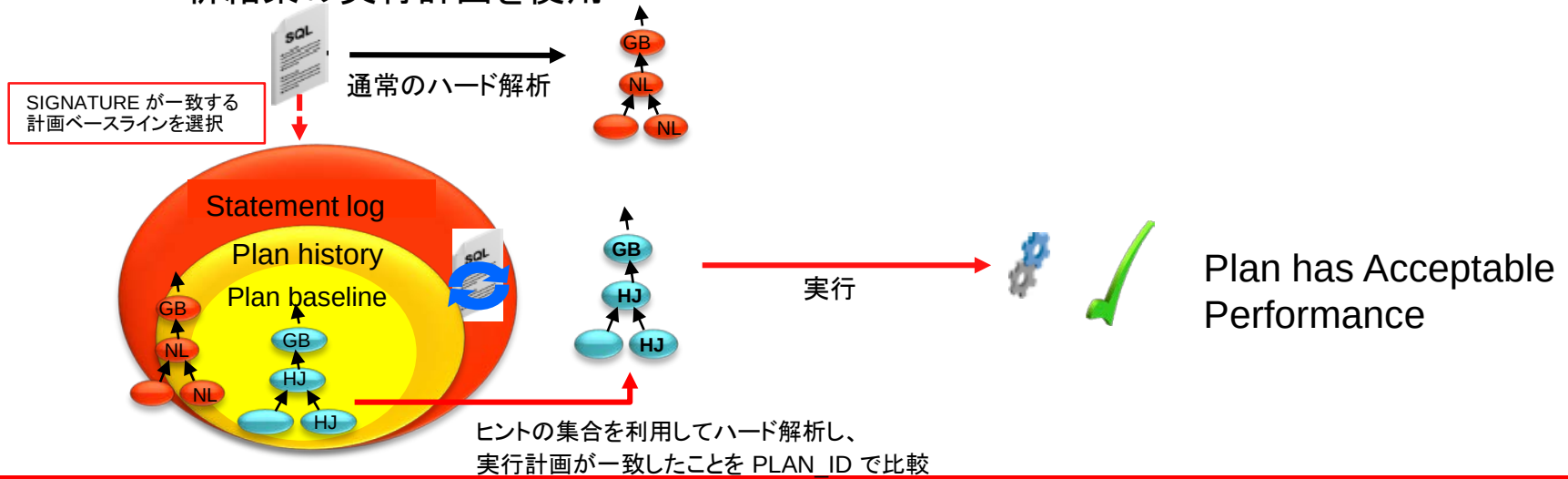


TIPS!

1. 保存されている計画ベースラインの FIXED 属性を ‘YES’ にすることで、新たな実行計画を計画履歴に保存することを抑止可能

ハード解析フェーズの流れ ② - 2 (① で PLAN_ID が一致しない場合)

- 計画ベースラインの実体であるヒントの集合を利用して、再度ハード解析を実施
- ハード解析結果の実行計画と、計画ベースラインの実行計画を PLAN_ID で比較
- PLAN_ID が一致した場合は、ハード解析結果の実行計画で SQL 文を実行
- PLAN_ID が一致しない場合は、計画ベースラインは利用不可と判断し、通常のハード解析結果の実行計画を使用



実行計画固定のための基礎情報

TIPS!

- 実行された SQL に対する計画ベースラインが登録されているかどうかを SIGNATURE で判断
- 元の実行計画と同じ実行計画を再現できたかどうかを PLAN_ID で比較
- SPM は実行計画をそのまま保持しているのではなく、元の実行計画を再現するためのヒントの集合を保持

	概要	確認方法
SIGNATURE	正規化した SQL のハッシュ値 (SIGNATURE が一致する場合に該当計画ベースラインを利用)	DBA_SQL_PLAN_BASELINES.SIGNATURE SQLOBJ$.SIGNATURE SQLOBJ$DATA.SIGNATURE
PLAN_ID	元の実行計画と一致したかどうかを比較するため、PLAN_ID を保持	SQLOBJ$.PLAN_ID SQLOBJ$DATA.PLAN_ID
ヒントの集合	元の実行計画と一致したかどうかを比較するため、PLAN_ID を保持	SQLOBJ$DATA.COMP_DATA

実行計画固定のための基礎情報を確認

TIPS!

- SQL の一致を SIGNATURE で識別し、実行計画の一致を PLAN_ID で識別

```
SQL> select spm.SIGNATURE, spm.SQL_HANDLE, spm.PLAN_NAME, spm.ENABLED, spm.ACCEPTED, spm.REPRODUCED, so.PLAN_ID
  2    from dba_sql_plan_baselines spm, sqlobj$ so
  3    where spm.SIGNATURE = 2667826512551042531 and spm.SIGNATURE = so.SIGNATURE and spm.PLAN_NAME = so.NAME;
```

SIGNATURE	SQL_HANDLE	PLAN_NAME	ENA	ACC	REP	PLAN_ID
2667826512551042531	SQL_2506061412f95de3	SQL_PLAN_2a1h62h9gkrg371befc37	YES	YES	YES	1908341815

- 実行計画を固定するために SPM ではヒントの集合を保持

```
SQL> select spm.SIGNATURE, spm.SQL_HANDLE, spm.PLAN_NAME, sod.COMP_DATA
  2    from dba_sql_plan_baselines spm, sqlobj$ so, sqlobj$data sod
  3    where spm.SIGNATURE = 2667826512551042531 and spm.PLAN_NAME = 'SQL_PLAN_2a1h62h9gkrg371befc37'
  4      and spm.SIGNATURE = so.SIGNATURE and spm.SIGNATURE = sod.SIGNATURE
  5      and spm.PLAN_NAME = so.NAME and so.PLAN_ID = sod.PLAN_ID;
```

SIGNATURE	SQL_HANDLE	PLAN_NAME
COMP_DATA		
2667826512551042531	SQL_2506061412f95de3	SQL_PLAN_2a1h62h9gkrg371befc37

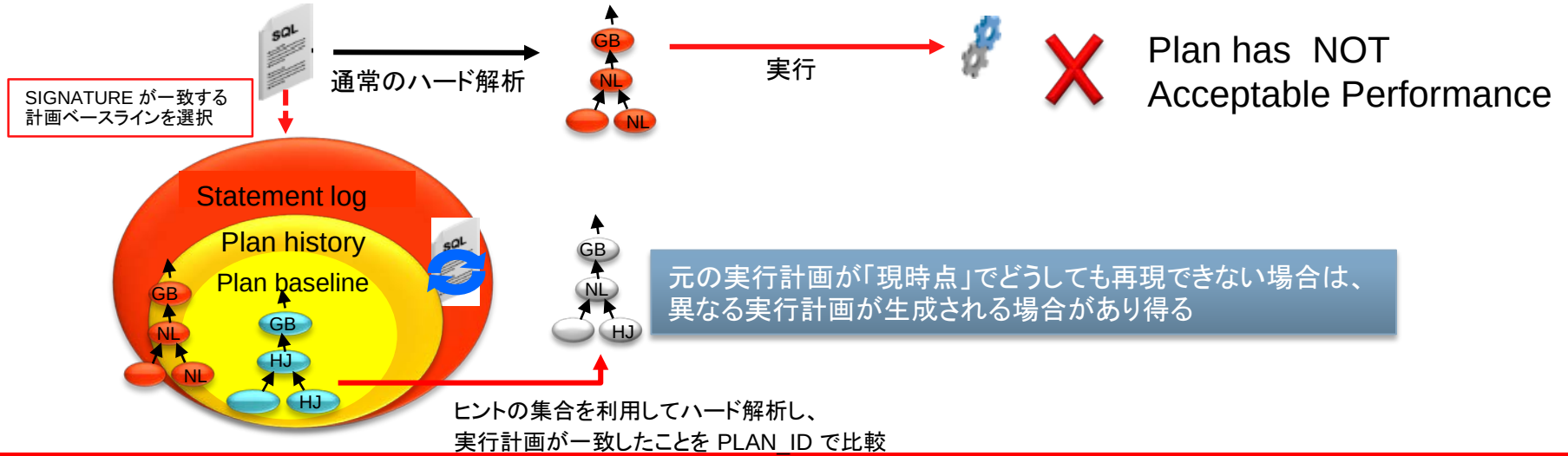
<outline_data><hint><![CDATA[IGNORE_OPTIM_EMBEDDED_HINTS]]></hint><hint><![CDATA[OPTIMIZER_FEATURES_ENABLE('11.2.0.2')]]></hint><hint><![CDATA[DB_VERSION('11.2.0.2')]]></hint><hint><![CDATA[OPT_PARAM('_replace_virtual_columns' 'false')]]></hint><hint><![CDATA[OPT_PARAM('_optimizer_use_feedback' 'false')]]></hint><hint><![CDATA[OPT_PARAM('_fix_control' '7426911:0')]]></hint><hint><![CDATA[ALL_ROWS]]></hint><hint><![CDATA[OUTLINE_LEAF(@"SEL$1")]]></hint><hint><![CDATA[INDEX_RS_ASC(@"SEL$1" "TAB2"@"SEL$1" ("TAB2"."C2"))]]></hint></outline_data>

運用における SPM 障害事例

計画ベースラインが利用できない？

- 計画ベースラインを利用することで、オプティマイザがハード解析時にヒントの集合を考慮し、通常は元の実行計画を再現させることが可能
- SPM では実行計画が再現できたことを PLAN_ID というハッシュ値で比較し確認
- PLAN_ID が一致しない場合は、計画ベースラインは利用不可と判断し、通常のハード解析結果の実行計画を使用（計画ベースラインは REPRODUCED = NO となる）

※ DBA_SQL_PLAN_BASELINES の REPRODUCED 列は、11.2 から追加された列



SPM が利用できていることの確認 ①

TIPS!

- V$SQL (GV$SQL) による確認

```
SQL> select EXACT_MATCHING_SIGNATURE, SQL_ID, CHILD_NUMBER, PLAN_HASH_VALUE,
SQL_PLAN_BASELINE,
  2         substr(SQL_TEXT, 1, 100) SQL_TEXT
  3  from v$sql
  4  where EXACT_MATCHING_SIGNATURE = 2667826512551042531;
```

計画ベースラインを利用している場合、V$SQL.SQL_PLAN_BASELINE 列にプラン名が格納される

EXACT_MATCHING_SIGNATURE	SQL_ID	CHILD_NUMBER	PLAN_HASH_VALUE	SQL_PLAN_BASELINE	SQL_TEXT
2667826512551042531	3m7d74pkw543z	0	2200541503	SQL_PLAN_2a1h62h9gkrg371befc37	select * from tab2 where c2 > 999

- DBA_SQL_PLAN_BASELINES による確認

TIPS!

1. LAST_EXECUTED が 6.5 日間更新されない

```
SQL> set lines 120
SQL> col SQL_HANDLE for a20
SQL> col LAST_EXECUTED for a20
SQL> select spm.SIGNATURE, spm.SQL_HANDLE, spm.PLAN_NAME, spm.ENABLED, spm.ACCEPTED,
spm.REPRODUCED,
  2         to_char(spm.LAST_EXECUTED, 'YYYY/MM/DD HH24:MI:SS') LAST_EXECUTED
  3    from dba_sql_plan_baselines spm
  4    where spm.SIGNATURE = 2667826512551042531
  5      and spm.PLAN_NAME = 'SQL_PLAN_2a1h62h9gkrg371befc37';
```

計画ベースラインが利用された場合、DBA_SQL_PLAN_BASELINES.LAST_EXECUTED 列に利用された日時が格納され、REPRODUCED 列は YES となっている。
利用しようとしたができなかった場合、REPRODUCED 列が NO となる。

SIGNATURE	SQL_HANDLE	PLAN_NAME	ENA	ACC	REP	LAST_EXECUTED
2667826512551042531	SQL_2506061412f95de3	SQL_PLAN_2a1h62h9gkrg371befc37	YES	YES	YES	2012/06/06 00:38:16

SPM が利用できていることの確認 ②

TIPS!

```
SQL> select * from table(dbms_xplan.display_cursor('3m7d74pkw543z', 0));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------
SQL_ID  3m7d74pkw543z, child number 0
-------------------------------------
select * from tab2 where c2 > 999

Plan hash value: 2200541503

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |      |       |       |    12 (100)|          |
|   1 |  TABLE ACCESS BY INDEX ROWID| TAB2 |    10 |  2280 |    12   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | IND2 |    10 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

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

   2 - access("C2">999)

Note
-----
   - dynamic sampling used for this statement (level=2)
   - SQL plan baseline SQL_PLAN_2a1h62h9gkrg371befc37 used for this statement

24 rows selected.
```

DBMS_XPLAN パッケージで実行計画を表示すると、利用された計画ベースラインの名前を確認することができる。

実行計画が再現できない例

No.	分類	状況	例
1	オブジェクトの存在	•索引削除 •マテリアライズド・ビュー削除	•元の実行計画のアクセス・パスで利用するオブジェクトが削除されている場合。
2	オブジェクトの有効/無効	•トリガーの無効化 •外部キー制約の無効化(もしくはNOT VALIDATED)	•バッチ処理の時間帯のみトリガーを無効化する場合。 •データ移行の高速化のために、外部キー制約を NOT VALIDATED にした場合。
3	計画ベースライン登録方法の違い	•SQL*Plus から直接実行される SQL と PL/SQL から実行される SQL の実行計画の違い	•PL/SQL で実行された SQL の実行計画を計画ベースラインとして登録し、その SQL 文を SQL*Plus から(PL/SQL ブロック内ではなく直接) 実行した場合。 •もしくはその逆の場合。

実行計画が再現できない事例 ① 1/2

トリガーの無効化

事象

- ✓ 夜間バッチ処理の性能改善のため、アプリケーションを改修せず、ヒントで実行計画を変更して元のSQL 文の SQL 計画ベースラインとして登録した。
- ✓ 夜間バッチ処理を実行したところ、登録しているSQL 計画ベースラインの実行計画を利用することができず、性能が悪い実行計画が利用された。
- ✓ それにより、バッチ処理の性能が低下し、バッチ・ウィンドウに納まらず、バッチ処理を中断しなければならなかった。

原因調査の状況

- ✓ 該当の SQL で利用している表は、トリガーが作成されており、オンライン処理時間帯はトリガーが有効で、バッチ処理時間帯は処理の高速化のためにトリガーを無効にしていた。
- ✓ 登録されている SQL 計画ベースラインの実行計画は、トリガーが有効なオンライン処理時間帯に作成された実行計画であった。

実行計画が再現できない事例 ① 2/2

トリガーの無効化

- トリガーの定義によっては、トリガーの有効/無効によって、同じ DML 文の実行計画が異なることがあり得る。
- トリガーが無効な場合は、有効な場合と同じ実行計画を再現できない。
- そのため、トリガーが有効な場合に作成した計画ベースラインは、トリガーが無効な場合に利用することができない。

トリガーの例

```
create or replace trigger emp_trig01
    after update of ename on emp
    for each row
declare
    old_ename varchar2(100);
begin
    old_ename := :OLD.ename;
end;
/
```

実行 SQL 文

```
update emp set ename = 'SCOTT' where empno =
7788
```

トリガー有効

```
Plan hash value: 3891354354

-----------------------------------------------
| Id  | Operation                     | Name   |
-----------------------------------------------
|   0 | UPDATE STATEMENT              |        |
|   1 |  UPDATE                       | EMP    |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP    |
|*  3 |    INDEX UNIQUE SCAN          | PK_EMP |
-----------------------------------------------
```

トリガー無効

```
Plan hash value: 3659136155

-------------------------------------
| Id  | Operation           | Name   |
-------------------------------------
|   0 | UPDATE STATEMENT    |        |
|   1 |  UPDATE             | EMP    |
|*  2 |   INDEX UNIQUE SCAN| PK_EMP |
-------------------------------------
```

実行計画が再現できない事例 ② 1/2

NOT VALIDATED の外部キー制約

事象

- ✓ 検証環境で作成した SQL 計画ベースラインを、本番環境へ移行した（ステージング表のエクスポート/インポート）。
- ✓ 検証環境で利用できていた SQL 計画ベースラインが本番環境では利用できず、性能の悪い実行計画が採用された。

原因調査の状況

- ✓ 本番環境へのデータの移行時に、移行時間短縮のため外部キー制約を NOT VALIDATED にしていた。(整合性は保証されているデータを移行)
- ✓ 検証環境はデータ量が少ないため、VALIDATED にしていた。
- ✓ 該当の SQL は、外部キー制約の親表と子表を結合する SQL であった。

実行計画が再現できない事例 ② 2/2

NOT VALIDATED の外部キー制約

- 親表に対する存在確認のための結合は、外部キー制約が定義されている場合にオプティマイザの最適化によって除外される。
- 外部キー制約が NOT VALIDATED となっている場合、オプティマイザの最適化は行われないため、親表との結合が行われ、VALIDATED な場合の実行計画を再現することができない。

```
SELECT E.ENAME
FROM EMP E, DEPT D
WHERE E.DEPTNO = D.DEPTNO;
```

VALIDATED

```
Plan hash value: 3956160932

-------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | C
-------------------------------------------------
|   0 | SELECT STATEMENT  |      |    12 |   108 |
|*  1 |  TABLE ACCESS FULL| EMP  |    12 |   108 |
-------------------------------------------------

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

   1 - filter("E"."DEPTNO" IS NOT NULL)
```

NOT VALIDATED

```
Plan hash value: 3074306753

-----------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | C
-----------------------------------------------------------
|   0 | SELECT STATEMENT   |         |    12 |   144 |
|   1 |  NESTED LOOPS      |         |    12 |   144 |
|   2 |   TABLE ACCESS FULL| EMP     |    12 |   108 |
|*  3 |   INDEX UNIQUE SCAN| PK_DEPT |     1 |     3 |
-----------------------------------------------------------

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

   3 - access("E"."DEPTNO"="D"."DEPTNO")
```

実行計画が再現できない事例 ③ 1/2

SQL*Plus と PL/SQL から実行される SQL の実行計画の違い

事象

- ✓ SQL の性能改善のため、アプリケーションを改修せず、ヒントで実行計画を変更して元の SQL 文の SQL 計画ベースラインとして登録した。
- ✓ SQL*Plus で該当 SQL 文を実行すると、その計画ベースラインを利用することができるが、アプリケーションから SQL が実行された場合にはその計画ベースラインを利用することができず、性能改善ができていない。

原因調査の状況

- ✓ 該当アプリケーションでは、PL/SQL のパッケージから SQL 文を実行していた。
- ✓ 該当 SQL 文では同じ名前のバインド変数が複数回利用されていた。
- ✓ ヒントで実行計画を変更して計画ベースラインへ登録する際には、PL/SQL 内で SQL 文を実行するのではなく、SQL*Plus に SQL 文を張り付けて実行していた。

実行計画が再現できない事例 ③ 2/2

SQL*Plus と PL/SQL から実行される SQL の実行計画の違い

- 同じ名前のバインド変数が複数回利用されていると、PL/SQL ブロック内から実行されると FILTER というステップが実行計画に含まれる。(～ 11.2.0.3)
- SQL*Plus ではこのバインド変数同士を比較する FILTER が含まれることはないため、両者が同じ実行計画をとることができない。

```
SQL 文
SELECT E.ENAME,D.DNAME
FROM EMP E, DEPT D
WHERE E.DEPTNO = D.DEPTNO
  AND E.DEPTNO = :B1 AND D.DEPTNO
= :B1)
```

同じ名前のバインド変数が複数回利用されている

PL/SQL ブロックで実行

```
-----------------------------------------------------
| Id  | Operation           | Name | Rows  | Bytes | Co
-----------------------------------------------------
|   0 | SELECT STATEMENT    |      |       |       |
|*  1 |  FILTER             |      |       |       |
|*  2 |   HASH JOIN         |      |     1 |  1030 |
|*  3 |    TABLE ACCESS FULL| EMP  |     1 |   515 |
|*  4 |    TABLE ACCESS FULL| DEPT |     1 |   515 |
-----------------------------------------------------

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

   1 - filter(:B1=:B1)
   2 - access("E"."DEPTNO"="D"."DEPTNO")
   3 - filter(("E"."DEPTNO"=:B1 AND "E"."DEPTNO"=:B1))
   4 - filter(("D"."DEPTNO"=:B1 AND "D"."DEPTNO"=:B1))
```

SQL*Plus で実行

```
-----------------------------------------------------
| Id  | Operation           | Name | Rows  | Bytes | Co
-----------------------------------------------------
|   0 | SELECT STATEMENT    |      |       |       |
|*  1 |  HASH JOIN          |      |     1 |  1030 |
|*  2 |   TABLE ACCESS FULL| EMP  |     1 |   515 |
|*  3 |   TABLE ACCESS FULL| DEPT |     1 |   515 |
-----------------------------------------------------

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

   1 - access("E
   2 - filter("E
   3 - filter("D
```

TIPS!

1. PL/SQL から実行される SQL をヒントでチューニングする場合は、PL/SQL ブロック内で SQL を実行する必要がある。

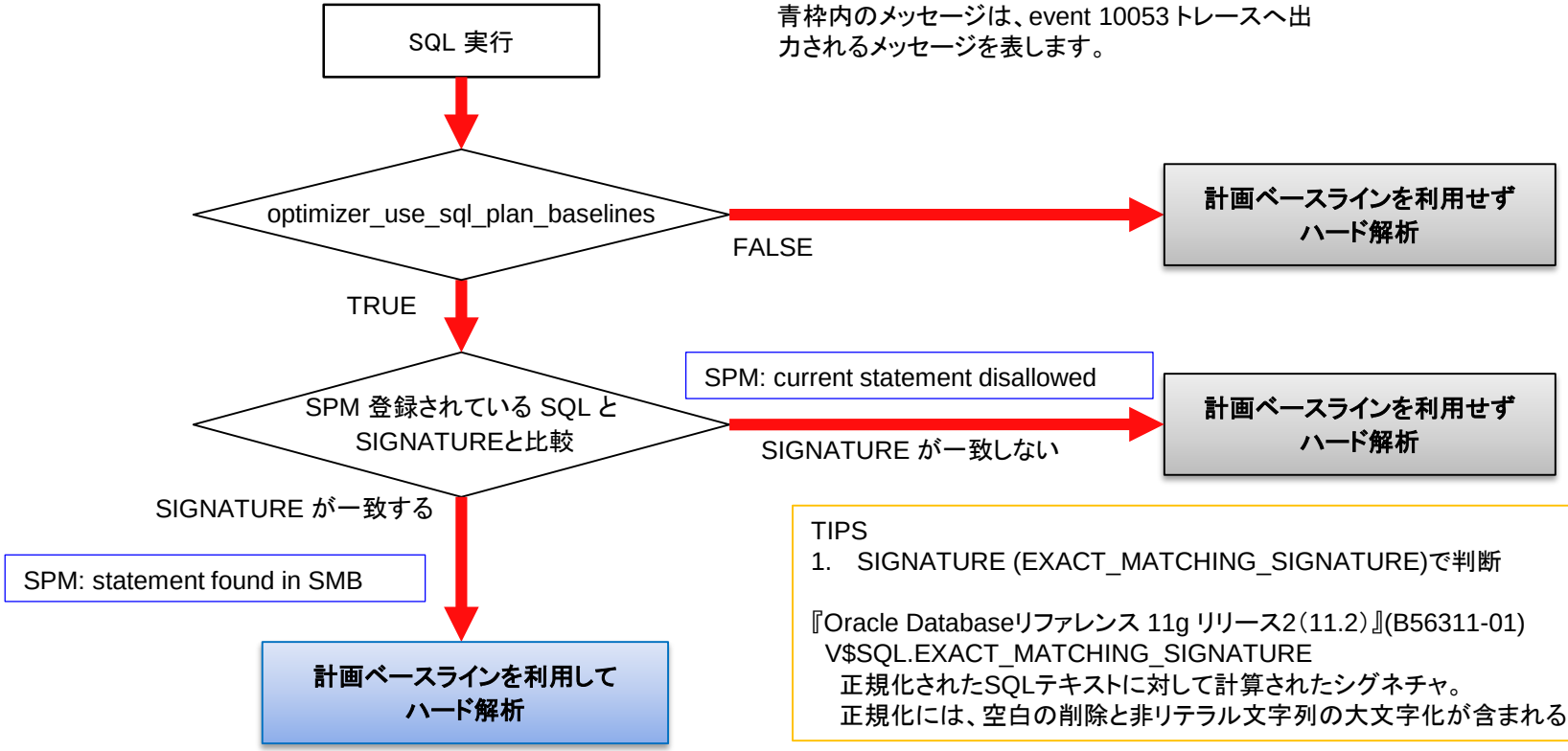
まとめ

まとめ

- アップグレード案件において、実行計画の変化に伴う性能低下を最小限に抑えることができる。
- 性能障害への対応速度が向上する。
  - ✓ ヒントと SPM を活用することで、アプリケーションに修正を加えることなく性能の改善/安定化が実現できる。
  - ✓ 突然の性能問題に対して、過去の実行計画を利用した性能の安定化が実現できる。
- SPM は比較的新しい機能であり、機能改善が行われているため、できるだけ新しい製品バージョンの利用を検討する。
- SPM で実行計画を固定できない場合、最終的にアプリケーションを修正することを検討する。
- より SPM を活用するために、製品マニュアルとともに本資料を参考に SPM に関する理解を深めてください！

Appendix

A-1. ハード解析で SPM を利用するかどうかの判断



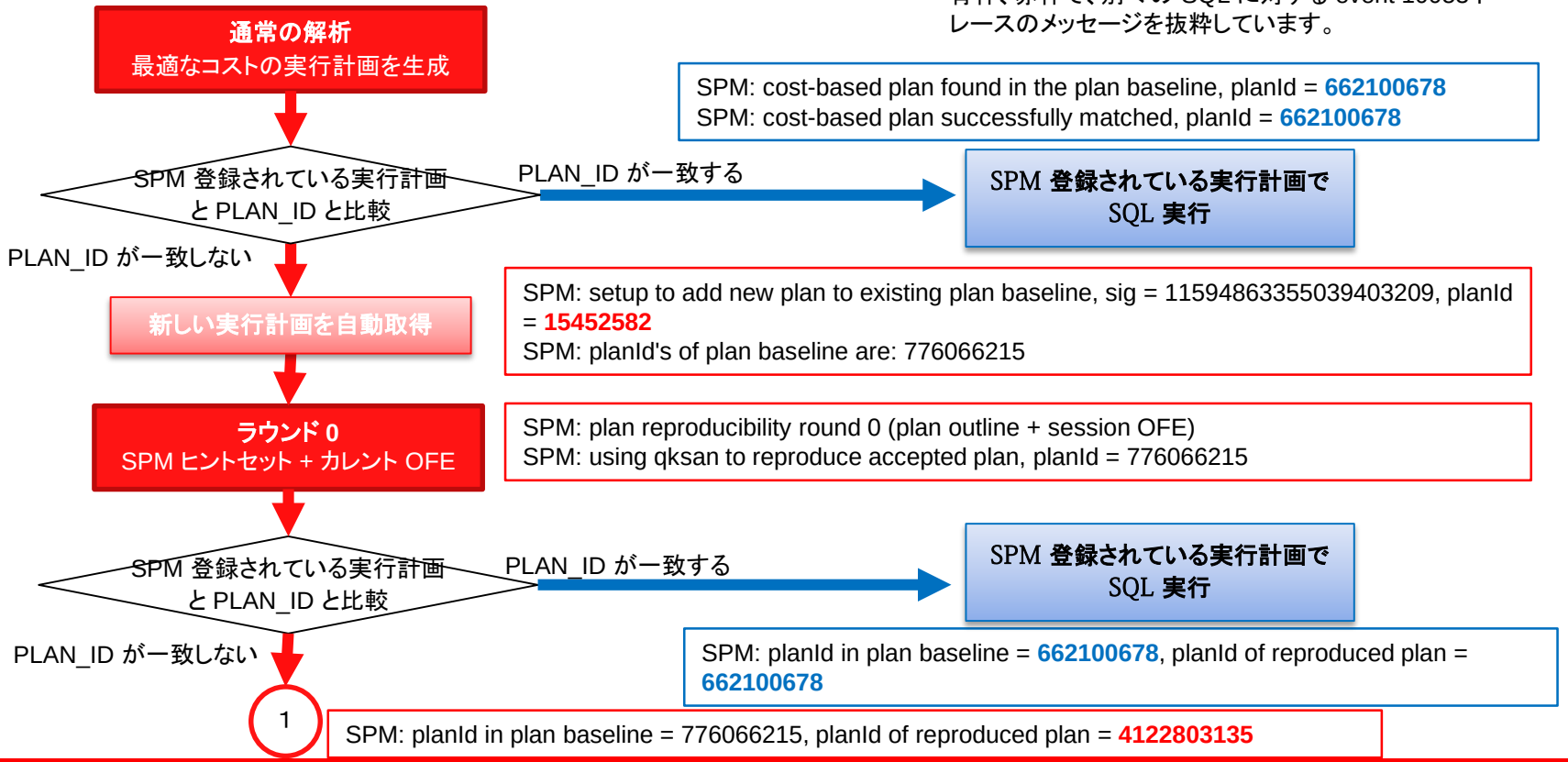
青枠内のメッセージは、event 10053 トレースへ出力されるメッセージを表します。

TIPS
1. SIGNATURE (EXACT_MATCHING_SIGNATURE)で判断

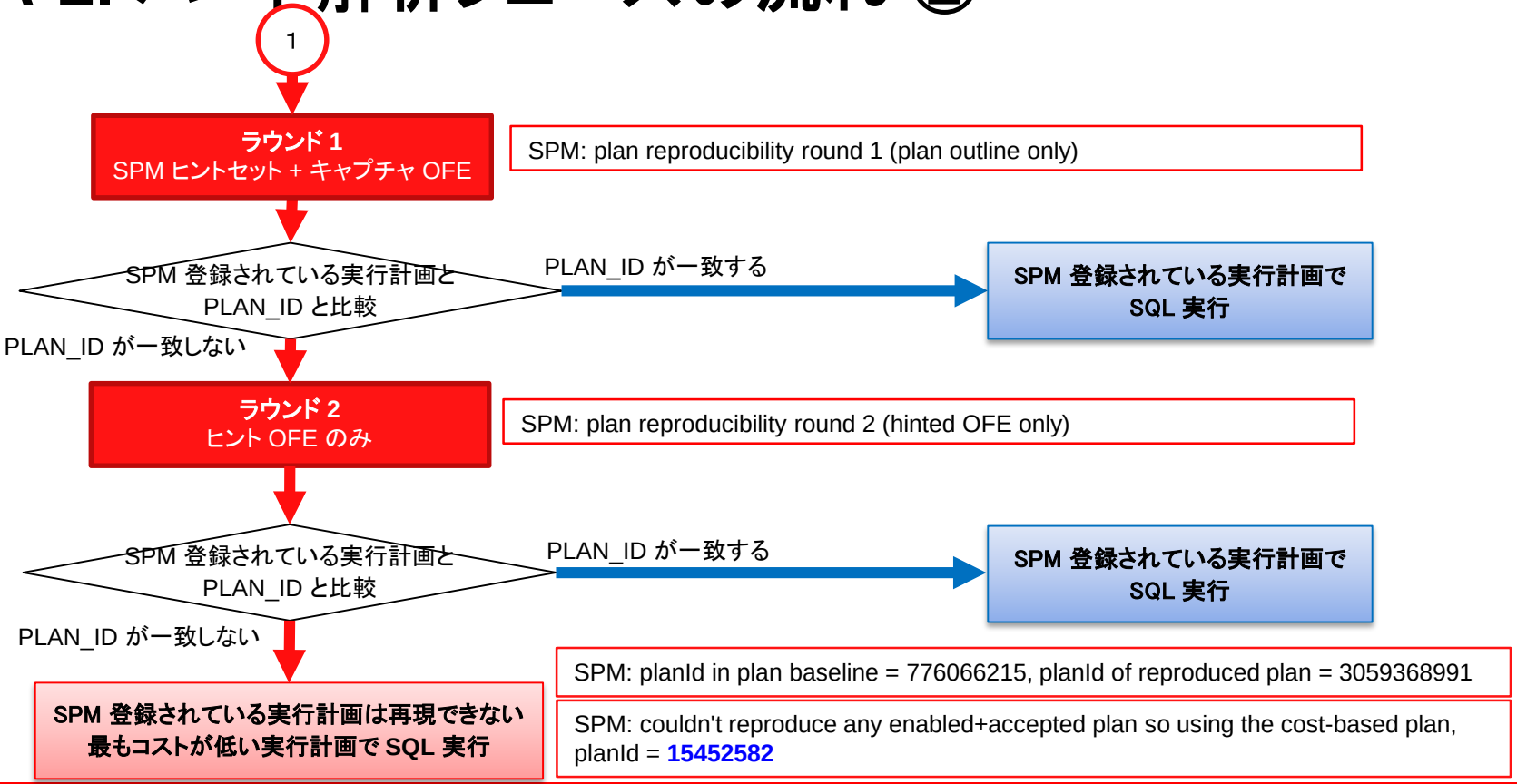
『Oracle Databaseリファレンス 11g リリース2（11.2）』(B56311-01)
V$SQL.EXACT_MATCHING_SIGNATURE
正規化されたSQLテキストに対して計算されたシグネチャ。
正規化には、空白の削除と非リテラル文字列の大文字化が含まれる

A-2. ハード解析フェーズの流れ ①

青枠、赤枠で、別々の SQL に対する event 10053 トレースのメッセージを抜粋しています。



A-2. ハード解析フェーズの流れ ②

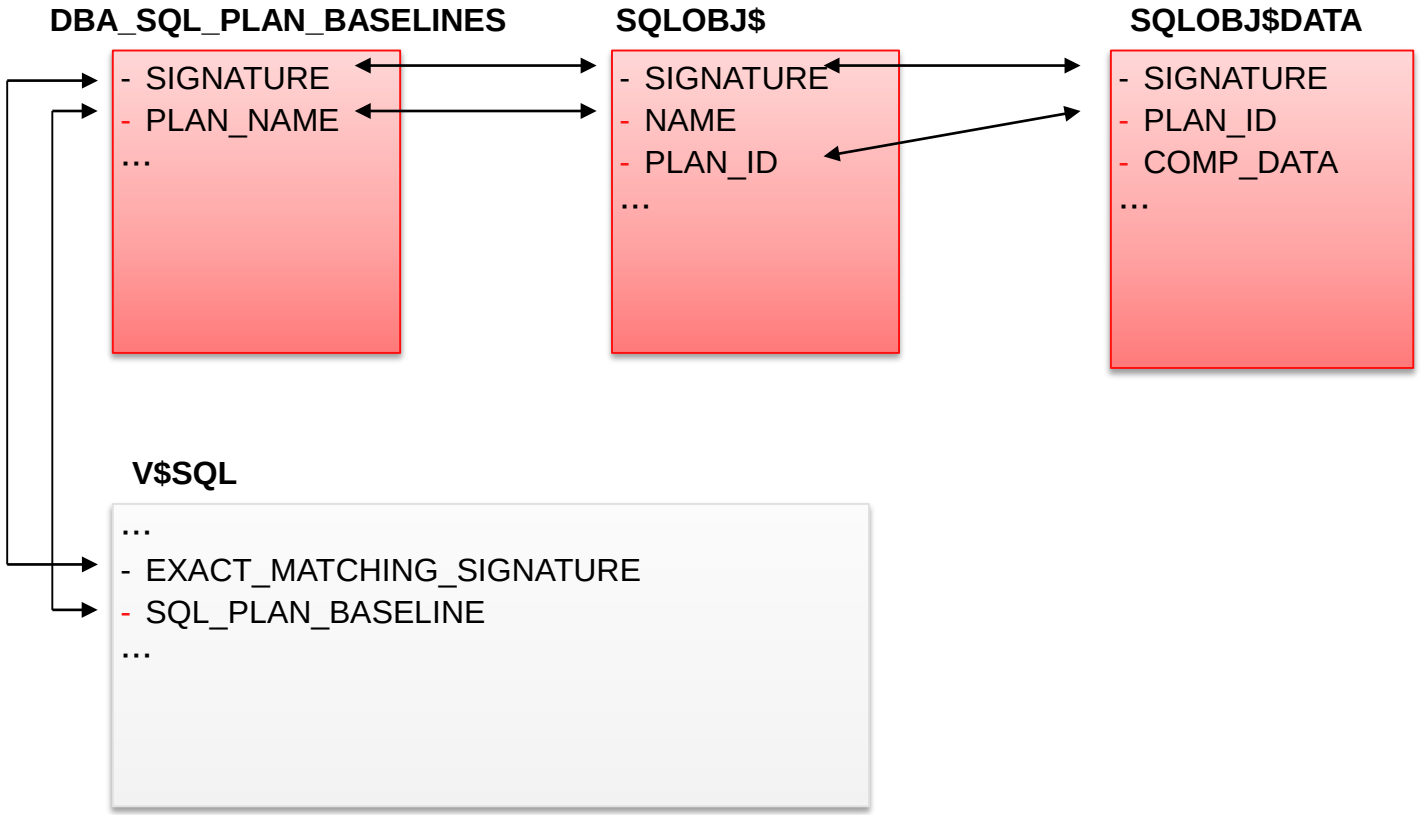


A-3. 主要なビュー DBA_SQL_PLAN_BASELINES

SPM 管理のための主要なディクショナリ・ビュー (一部の列のみ抜粋)

列名	説明
SIGNATURE	正規化した SQL のハッシュ値 (SIGNATURE が一致する場合に該当計画ベースラインを利用)
SQL_HANDLE	SPM で管理する SQL に対する名前
SQL_TEXT	SQL 本文
PLAN_NAME	各実行計画に対する名前 (DBMS_SPM パッケージにて変更可能)
ORIGIN	MANUAL-LOAD / AUTO-CAPTURE / MANUAL-SQLTUNE / AUTO-SQLTUNE / STORED-OUTLINE (UNKNOWN)
DESCRIPTION	デフォルト NULL (DBMS_SPM パッケージにて500バイト以内で設定可能)
CREATED	計画ベースラインの作成日 (PACK / UNPACK でコピーした場合は UNPACK 実行日)
LAST_EXECUTED	前回この実行計画が利用された日 (前回の利用から 6.5 日間経過するまでは更新されない)
LAST_MODIFIED	何らかの属性が変更された日 (REPRODUCED 列の YES / NO が切り替わっただけでも日付は更新される)
REPRODUCED	通常 YES (NO の場合、その実行計画は有効に利用できていない。)

A-4. 主要なビューの関連



Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®