Enhancing Execution Plan Stability in Oracle SE2

Speaker:

- **Urh Srečnik** <urh.srecnik@abakus.si>
  OCP DBA, OCP J2SE, OCA OCI, OCIS Exadata
  DBA Team Lead @ Abakus Plus d.o.o.

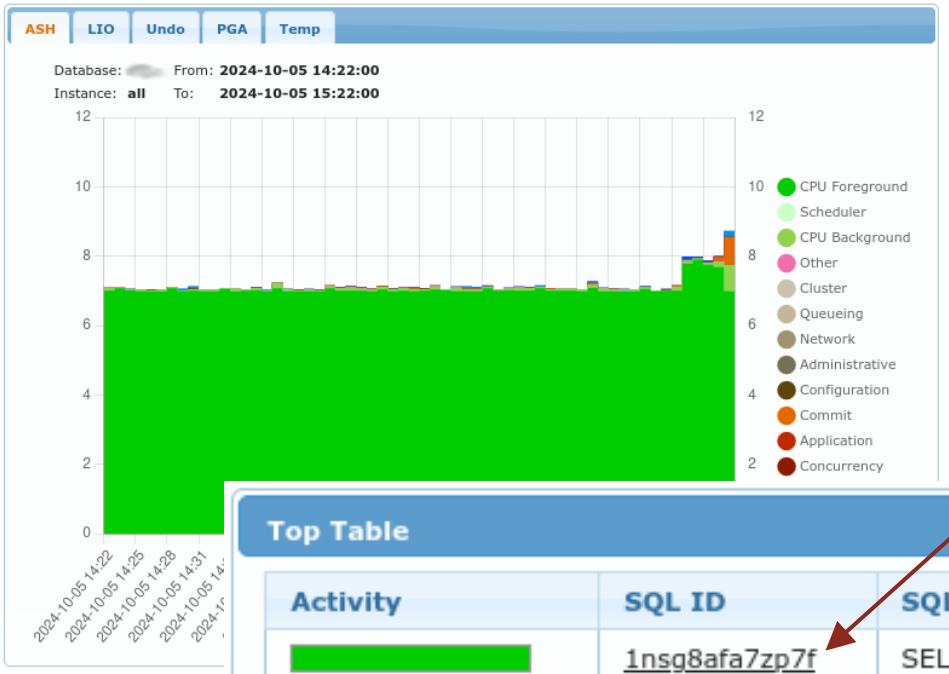
Oracle Optimizer

- 5-table join:
  - Join Order
    - 1*2*3*4*5 = 120 possible permutations.
  - Join method = NL/HJ/MJ
    - 3*3*3*3*3 = 81 (81*120 ~ 10.000)
  - Data Access = FTS/IUqS/I(Rg)S/ISkipS/IFFS/…
    - 5*5*5*5*5 = 3.000 (*10.000)
- 30.000.000 possible access paths
- 6-Table join is 2.733.750.000 possible access paths …
- Limited number of gueses (2000).

https://chandlerdba.com/presentations/why-has-my-plan-changed-top-7-plan-stability-pitfalls-and-how-to-avoid-them/


Photo by tom_kelley_archive on Freeimages.com

Identifying Poorly Performing SQL



This one!

Top Table

Activity	SQL ID	SQL Text	Duration
	1nsg8afa7zp7f	SELECT /*+ FIRST_ROWS(100) */	00d 06:00:06
	8cs6ppmzcqtg9	SELECT /*+ FIRST_ROWS(100) */	00d 01:00:01
	n/a		00d 00:06:06
	31tsppjhtvuvx	SELECT /*+ FIRST_ROWS(100) INDEX_ASC(a0	00d 00:02:59
	9c65jctq4z9p4	select	00d 00:00:42
	8q4g6azcjmu86	SELECT * FROM	00d 00:00:11
	74nvkgd6kd7xr	selec	00d 00:00:07
	61r3zcbj1nz2j	SELECT /*+ FIRST_ROWS(100) INDEX(	00d 00:00:05
	b9ynrqvgpf7x7	select * from	00d 00:00:04
	3xfg4hg8m0xm2	SELECT	00d 00:00:03

APPM

Download DEB | Download RPM | View On GitHub

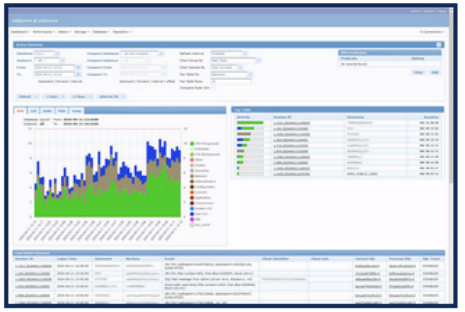
Abakus

Screenshots I

Following screenshots represent data which is available from APPM repository (postgres database). This data is periodically collected and stored into the APPM repository by APPM collector. Thus, all data presented here is available even when the Oracle database is not accessible.

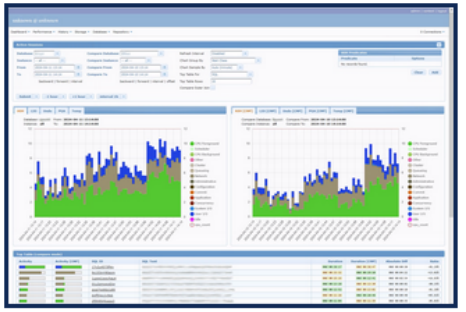
Dashboard

ASH of databases on same host/pdb/cluster



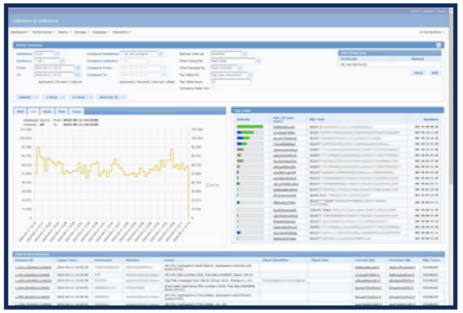
ASH

Active Session History for specific database



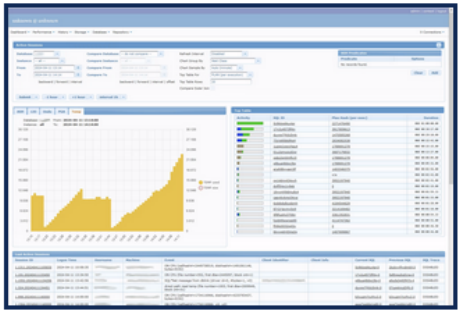
ASH Compare

Compare ASH and sql between different periods and/or databases



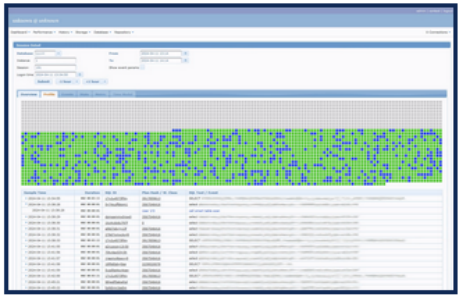
LIO & Top SQL

Amount of LIO per instance and top SQL statements



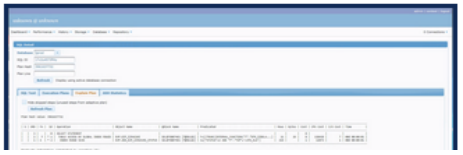
Temp TS & Top Execution Plans

Top 20 execution plans based on ASH.



Session Profiling

What session was doing every second



Free Edition!

https://appm.abakus.si/

Abakus Plus d.o.o.

- Infrastructure Team
  - HW & Services
    - OS & NET admin
    - DBA, Programming
  - Applications
    - **APPM**
    - Backup Server & Deja Vu
    - Arbiter
- Development Team
  - Enterprise Applications
  - Document Management
  - Newspaper Distribution
  - Flight Information System

Abakus
As na disku.

Customers

Modifying Execution Plans

(because sometimes even Oracle optimizer needs a bit of help)

- Hints
- ~~SQL Profiles~~ (not available in SE2)
- ~~Stored Outlines~~ (deprecated since 11.1)
- **SQL Patches**
- **Baselines** (partially available in SE2)

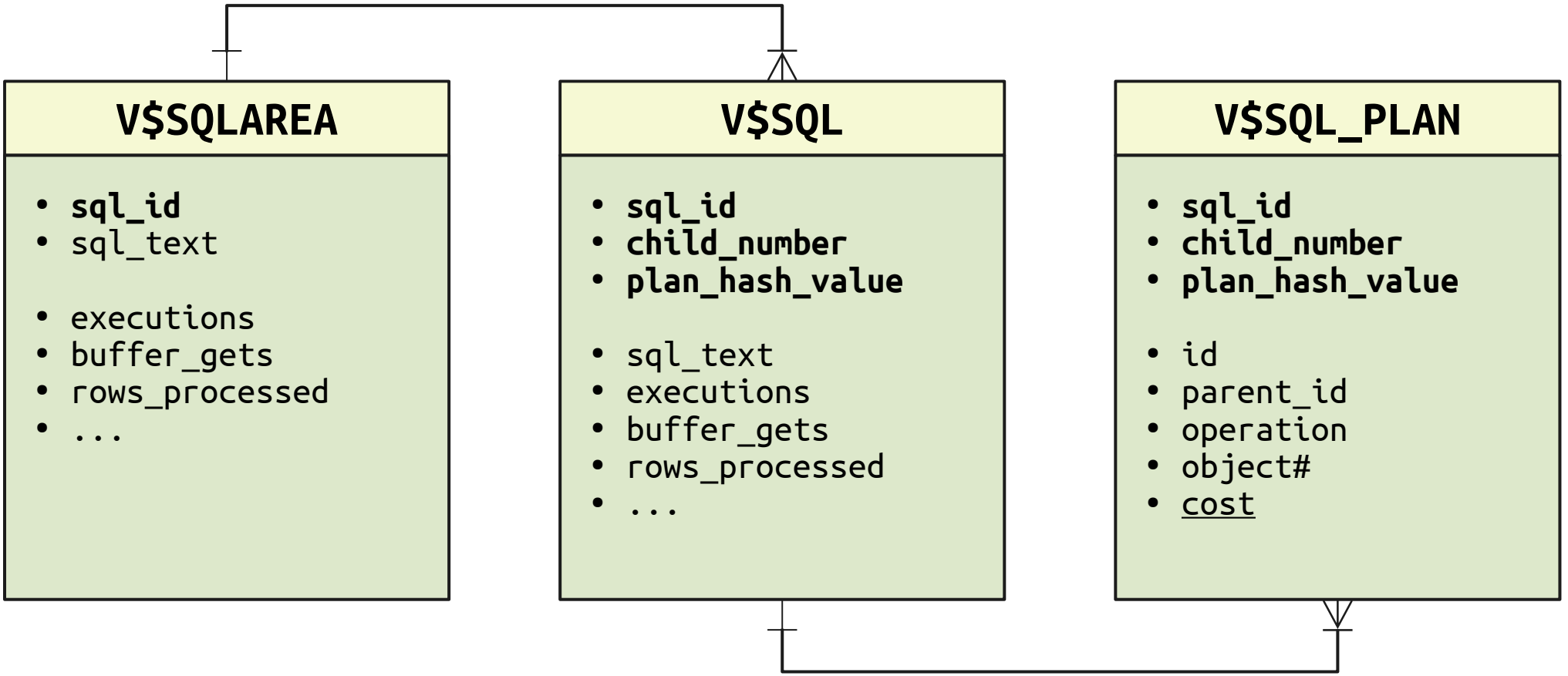
Hints

```
select /*+ ORDERED USE_HASH(t) FULL(o) INDEX(t) */ *
   from tab_one o
   join tab_two t on t.one_id = o.one_id;


Plan hash value: 2815624801

-------------------------------------------------------------------------------
| Id  | Operation                    | Name       | Rows | Bytes | Cost | Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |            |    1 |    47 |    5 | 00:00:01 |5
|*  1 |  HASH JOIN                   |            |    1 |    47 |    5 | 00:00:01 |4 USE_HASH(t)
|   2 |   TABLE ACCESS FULL          | TAB_ONE    |    1 |    30 |    3 | 00:00:01 |1 FULL(o)
|   3 |   TABLE ACCESS BY INDEX … | TAB_TWO    |    1 |    43 |    2 | 00:00:01 |3 INDEX(t)
|   4 |    INDEX FULL SCAN           | TAB_TWO_PK |    1 |       |    1 | 00:00:01 |2
-------------------------------------------------------------------------------
```

SQL Profiles

- Not available in SE2
- Mostly applies statistics-related hints to (e.g. `OPT_ESTIMATE`)
- We could use ***SQL Patch*** to mimic this to some degree.
- Statistics in those hints may become obsolete

V$SQL%



SQL Patch

- … adds hints to any SQL Statement (sql_id)
- available in SE2, no additional licence required

- >= 12.2
  - dbms_sqldiag.create_sql_patch
  - hint_text is CLOB.

- 11g, 12.1 (unofficial):
  - sys.dbms_sqldiag_internal.i_create_patch
  - hint_text is limited to 500 bytes

Creating a SQL Patch

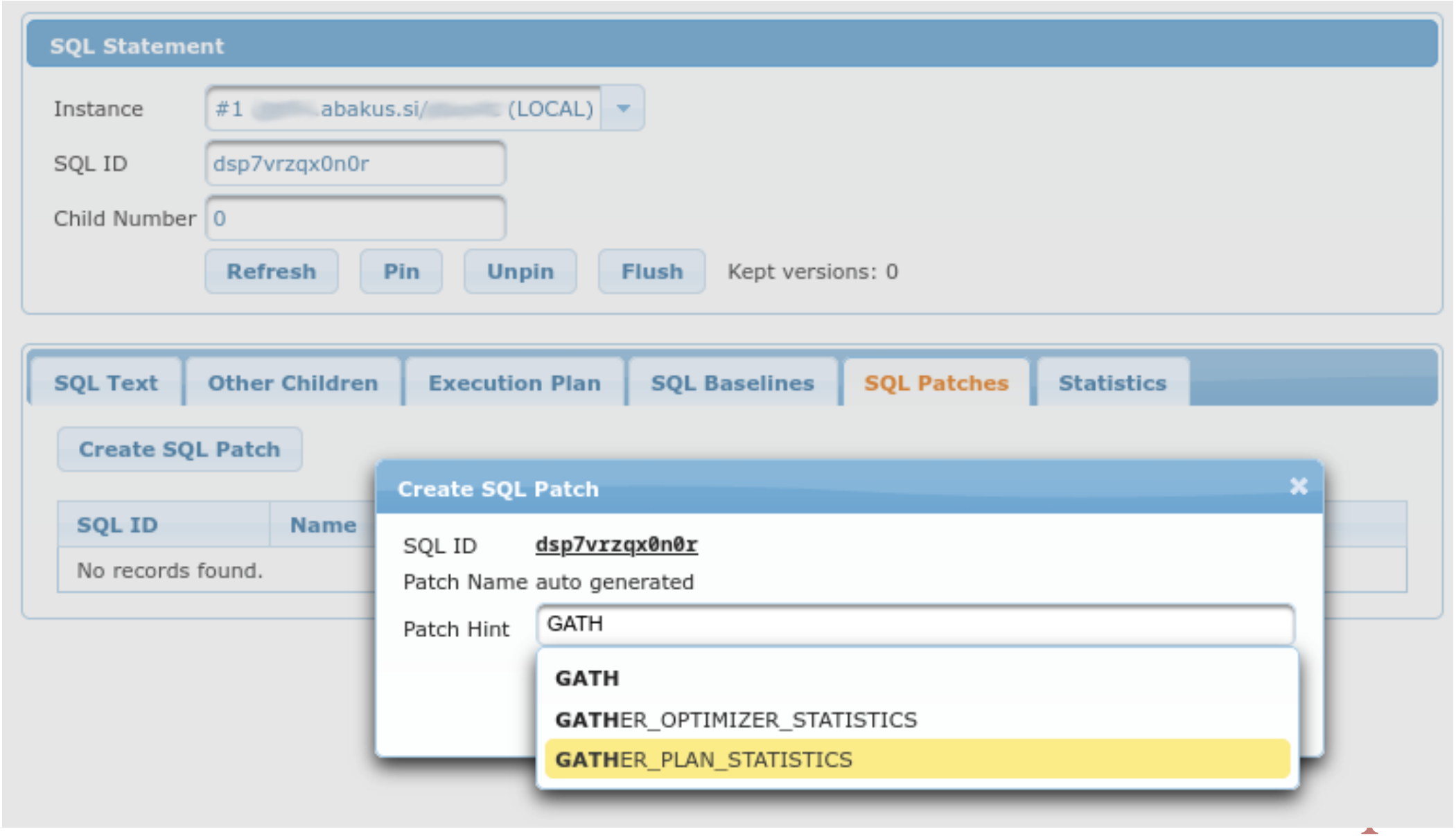
```
DBMS_SQLDIAG.CREATE_SQL_PATCH (
    sql_id          IN   VARCHAR2,
    hint_text       IN   CLOB,
    name            IN   VARCHAR2    := NULL,
    description     IN   VARCHAR2    := NULL,
    category        IN   VARCHAR2    := NULL,
    validate        IN   BOOLEAN     := TRUE)
RETURN VARCHAR2;
```

```
select * from dba_sql_patches;
```

Script Example

```
-- create_sql_patch.sql [sql_id] [hint] [patch_name]

var l_sql_id varchar2(13);
var l_hint varchar2(512);
var l_name varchar2(30);

DECLARE
   l_signature dba_sql_patches.name%TYPE;
   l_ret VARCHAR2(128);
BEGIN
   :l_sql_id := '&1.';
   :l_hint := '&2.';
   :l_name := '&3.';
   l_ret := dbms_sqldiag.create_sql_patch(sql_id => :l_sql_id,
      hint_text => :l_hint,
      name => :l_name,
      description => 'sql_id="' || :l_sql_id || '"',
      category => 'DEFAULT');
   sys.dbms_sqldiag.alter_sql_patch(:l_name, 'STATUS', 'ENABLED');
   dbms_output.put_line ('Created patch name: [' || l_ret || ']');
   FOR x_rec IN (SELECT DISTINCT hash_value, address FROM v$sql WHERE sql_id = :l_sql_id)
   LOOP
      sys.dbms_shared_pool.purge(x_rec.address || ' ' || x_rec.hash_value, 'C');
      dbms_output.put_line('Purged [' || x_rec.address || '/' || x_rec.hash_value || ']');
   END LOOP;
END;
/
```

APPM GUI Example



https://appm.abakus.si/

When To Use SQL Patches

- To add **a single or a few hints** to alter the statement execution plan
- To force one exact execution plan (outline data)
- To gather more info about the execution of specific statement.

Some Interesting Examples

- `ignore_optim_embedded_hints`
- `all_rows/first_rows_nnn`
- `ordered`
- `optimizer_features_enable`
- `gather_plan_statistics`
- `no_gather_optimizer_statistics`
- `use_invisible_indexes`
- ***Outline Data***

Stored Outlines

- Deprecated since 11.1
- We can mostly mimic this using **SQL Patch** by using Outline Data:

```
SQL> select * from
         table(dbms_xplan.display_cursor(
            '65furs9rva73q', 0, 'ADVANCED'));
```

Outline Data

```
SQL_ID  65furs9rva73q, child number 0
-------------------------------------
select /* URH_TEST3 */ *    from tab_one o    join tab_two t on
t.one_id = o.one_id

Plan hash value: 3311267762

-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |            |       |       |     4 (100)|          |
|   1 |  NESTED LOOPS                |            |     1 |    73 |     4   (0)| 00:00:01 |
|   2 |   NESTED LOOPS               |            |     1 |    73 |     4   (0)| 00:00:01 |
|   3 |    TABLE ACCESS FULL         | TAB_TWO    |     1 |    43 |     3   (0)| 00:00:01 |
|*  4 |    INDEX UNIQUE SCAN         | TAB_ONE_PK |     1 |       |     1   (0)| 00:00:01 |
|   5 |   TABLE ACCESS BY INDEX ROWID| TAB_ONE    |     1 |    30 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$58A6D7F6
   3 - SEL$58A6D7F6 / T@SEL$1
   4 - SEL$58A6D7F6 / O@SEL$1
   5 - SEL$58A6D7F6 / O@SEL$1

Outline Data
-------------

  /*+
      BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('19.1.0.1')
      DB_VERSION('19.1.0')
      OPT_PARAM('optimizer_index_cost_adj' 20)
      OPT_PARAM('optimizer_index_caching' 80)
      ALL_ROWS
      OUTLINE_LEAF(@"SEL$58A6D7F6")
      MERGE(@"SEL$1" >"SEL$2")
      OUTLINE(@"SEL$2")
      OUTLINE(@"SEL$1")
      FULL(@"SEL$58A6D7F6" "T"@"SEL$1")
      INDEX(@"SEL$58A6D7F6" "O"@"SEL$1" ("TAB_ONE"."ONE_ID"))
      LEADING(@"SEL$58A6D7F6" "T"@"SEL$1" "O"@"SEL$1")
      USE_NL(@"SEL$58A6D7F6" "O"@"SEL$1")
      NLJ_BATCHING(@"SEL$58A6D7F6" "O"@"SEL$1")
      END_OUTLINE_DATA
  */

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

   4 - access("T"."ONE_ID"="O"."ONE_ID")

Column Projection Information (identified by operation id):
-----------------------------------------------------------

   1 - "T"."TWO_ID"[NUMBER,22], "T"."ONE_ID"[NUMBER,22],
       "T"."TWO_TEXT"[VARCHAR2,30], "O"."ONE_ID"[NUMBER,22], "O"."ONE_TEXT"[VARCHAR2,30]
   2 - "T"."TWO_ID"[NUMBER,22], "T"."ONE_ID"[NUMBER,22],
       "T"."TWO_TEXT"[VARCHAR2,30], "O".ROWID[ROWID,10], "O"."ONE_ID"[NUMBER,22]
   3 - "T"."TWO_ID"[NUMBER,22], "T"."ONE_ID"[NUMBER,22],
       "T"."TWO_TEXT"[VARCHAR2,30]
   4 - "O".ROWID[ROWID,10], "O"."ONE_ID"[NUMBER,22]
   5 - "O"."ONE_TEXT"[VARCHAR2,30]

Query Block Registry:
---------------------

  <q o="2"><n><![CDATA[SEL$1]]></n><f><h><t><![CDATA[O]]></t><s><![CDATA[SEL$1]]></s>
       </h><h><t><![CDATA[T]]></t><s><![CDATA[SEL$1]]></s></h></f></q>
  <q o="18" f="y" h="y"><n><![CDATA[SEL$58A6D7F6]]></n><p><![CDATA[SEL$2]]></p><i><o>
       <t>VW</t><v><![CDATA[SEL$1]]></v></o></i><f><h><t><![CDATA[O]]></t><s><![CDATA[SEL$
       1]]></s></h><h><t><![CDATA[T]]></t><s><![CDATA[SEL$1]]></s></h></f></q>
  <q o="2"><n><![CDATA[SEL$2]]></n><f><h><t><![CDATA[from$_subquery$_003]]></t><s><![
       CDATA[SEL$2]]></s></h></f></q>
```

But...

- What if »better« plan is no longer in SGA?
- Enterprise Edition has AWR which contains old execution plans
  - On SE, we can mimic AWR-like functionality by capturing the same data from V$ views ourselves
  - trust me; we did it not because it is easy to do, but becase we *thought* it would be

Here's What We've Done:

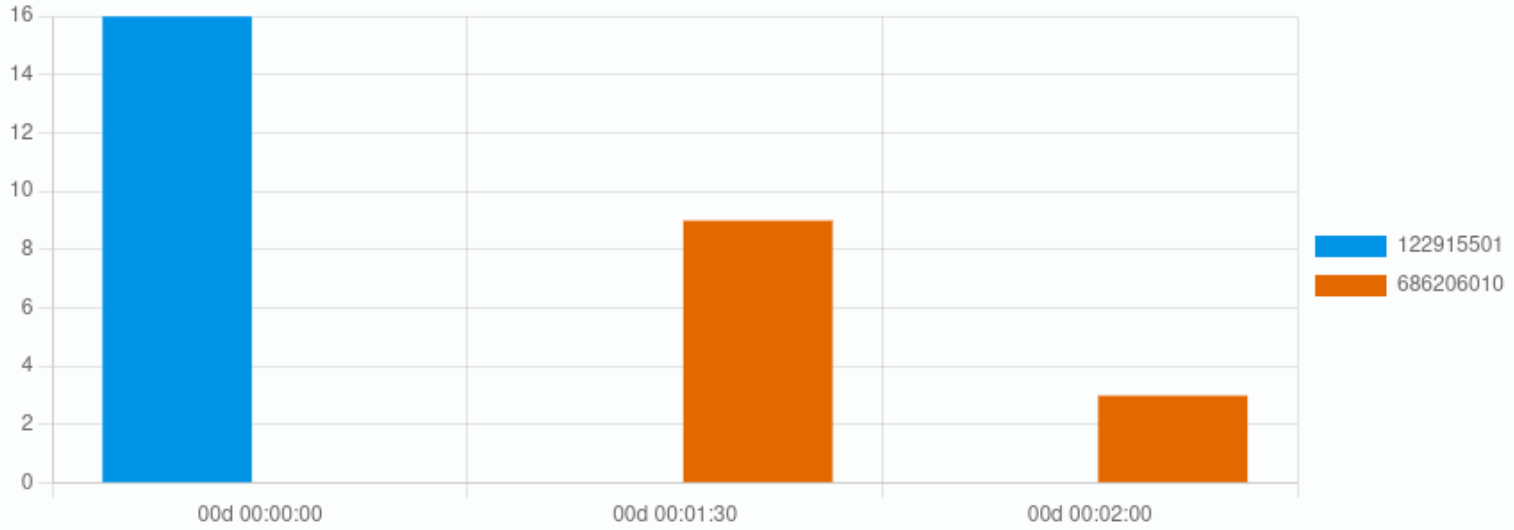
SQL Text | Execution Plans | Explain Plan | ASH Statistics

Hash Value	Full Hash Value	First Seen	Cost	Time
122915501	1405501415	2024-10-10 10:43:54	536	00d 00:00:01
686206010	3717389126	2024-10-10 10:10:17	384130	00d 00:00:16

Date From 2024-10-10 09:00 Compare From 2024-10-10 10:15

Date To 2024-10-10 12:00 Compare To 2024-10-10 11:15

Run Report

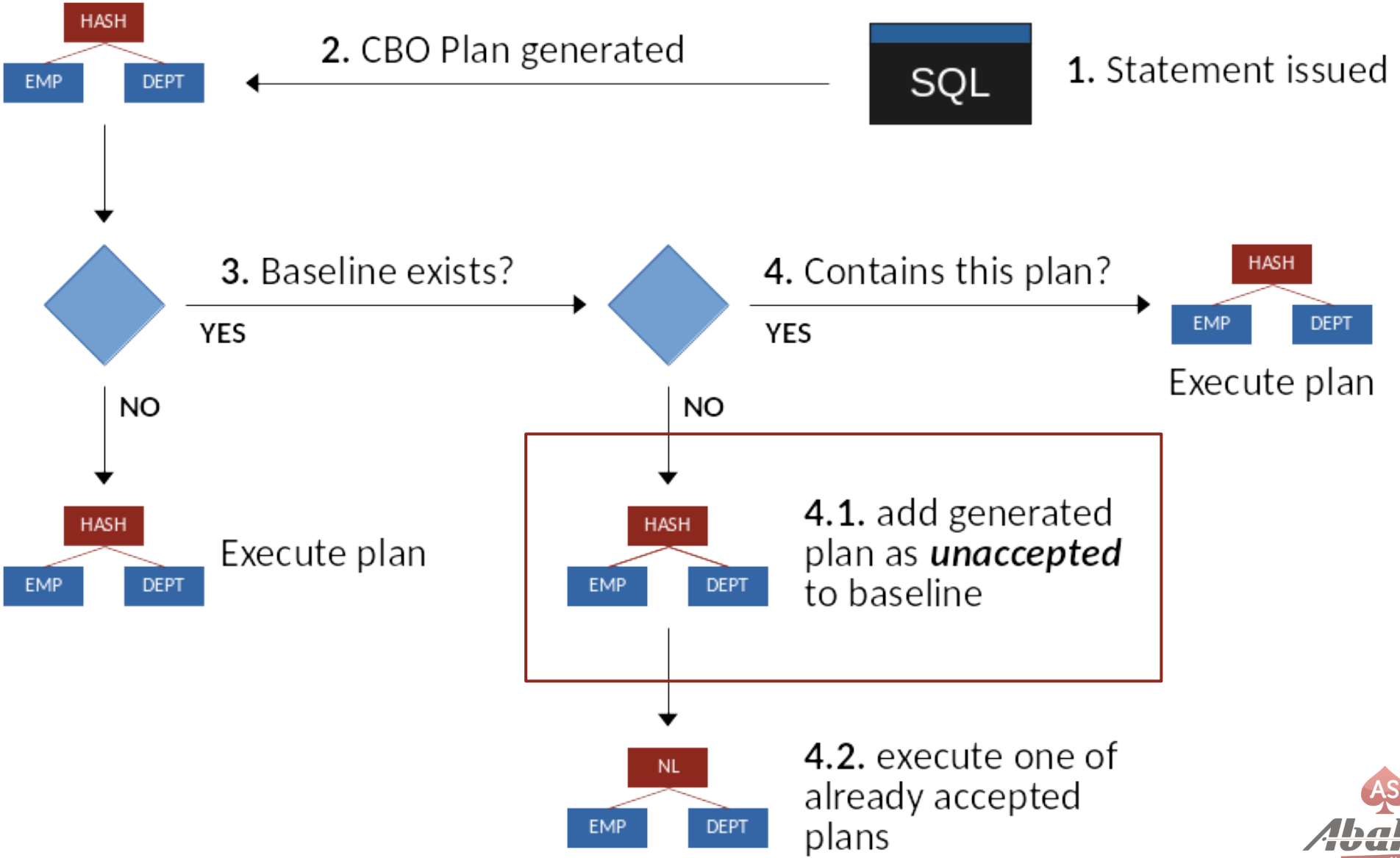


SQL ID	Plan Hash Value	Elapsed Time	Executions
f4zctq31w2cva	122915501	00d 00:00:00	16
f4zctq31w2cva	686206010	00d 00:01:30	9
f4zctq31w2cva	686206010	00d 00:02:00	3

plan_hash_value and each has **Outline Data** available

https://appm.abakus.si/

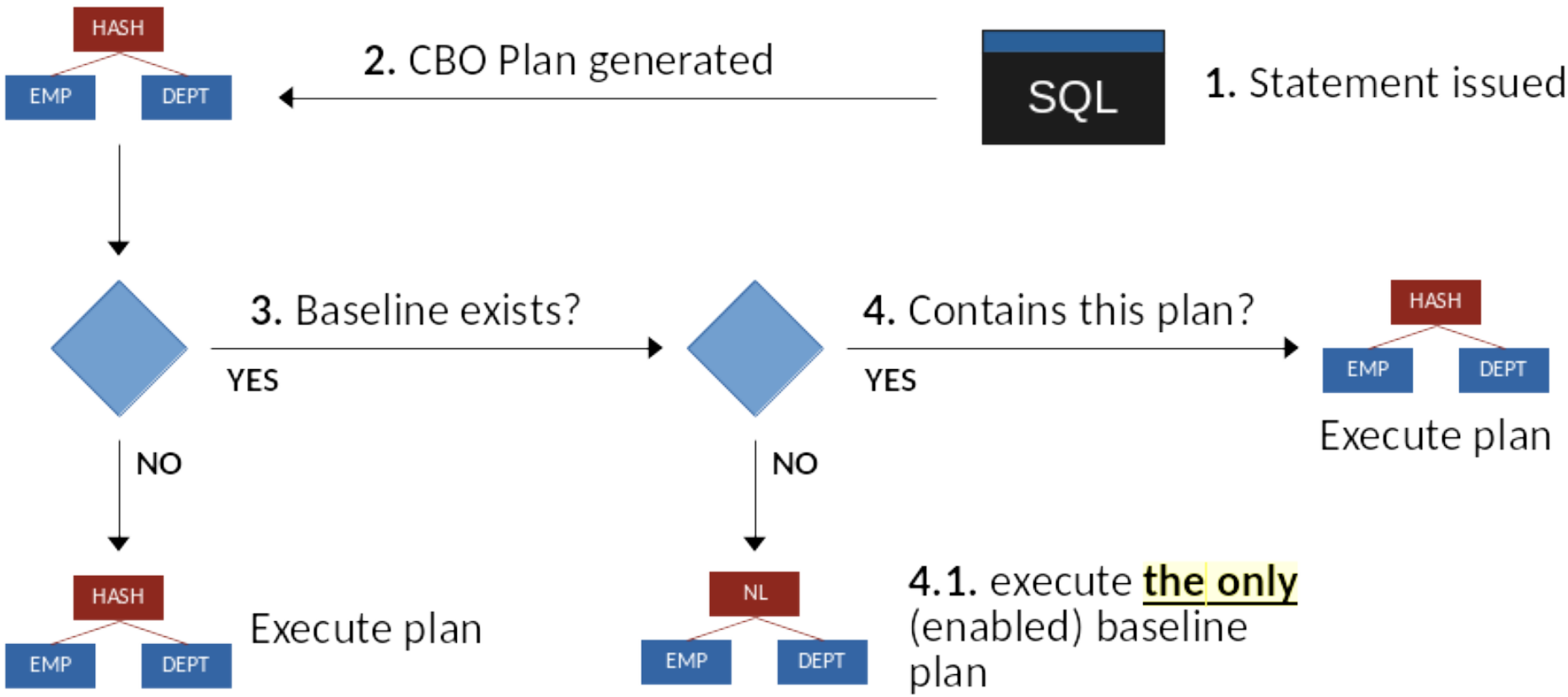
Baselines Introduction (EE!)



SE2 Limitations

- SE2 only allows to store a single SQL plan baseline per SQL statement
- The license guide remains the single source of truth

Baselines Intro in SE2



SQL Plan Baselines

```
SQL> select sql_handle, plan_name, optimizer_cost
          from dba_sql_plan_baselines;

SQL_HANDLE            PLAN_NAME                          OPTIMIZER_COST
--------------------- ---------------------------------- --------------
SQL_6d77b9dff78b24bb SQL_PLAN_6uxxtvzvsq95v5fc82e83                 55
SQL_b85e793d85b59754 SQL_PLAN_bhrmt7q2vb5undd079936                  4
```

sql_id

plan_hash_value

SMB

- SQL Plan Baselines are stored in SMB (*SQL Plan Management Base*)
- SMB is stored in **data dictionary** (contents exposed via `dba_sql_plan_baselines`)

Capture *Repeatable* Statements

```
SQL> alter system set
     OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES=true;
```

- Use `DBMS_SPM.CONFIGURE` to only capture statements for specific schema/action/module.

Capture Plans From SGA

```
set serveroutput on;
declare
   l_plan_count number := 0;
begin
   for l_plan in (select distinct sql_id, plan_hash_value
                     from v$sql
                     where parsing_schema_name='MY_APP')
   loop
         l_plan_count := l_plan_count + dbms_spm.load_plans_from_cursor_cache (
            sql_id => l_plan.sql_id,
            plan_hash_value => l_plan.plan_hash_value,
            fixed => 'NO',
            enabled => 'NO');
   end loop;
   dbms_output.put_line('Loaded [' || l_plan_count || '] plans.');
end;
```

Displaying Plans From SMB

```
select * from table(
   dbms_xplan.DISPLAY_SQL_PLAN_BASELINE(
      sql_handle => 'SQL_b85e793d85b59754',
      plan_name => 'SQL_PLAN_bhrmt7q2vb5undd079936'));
```

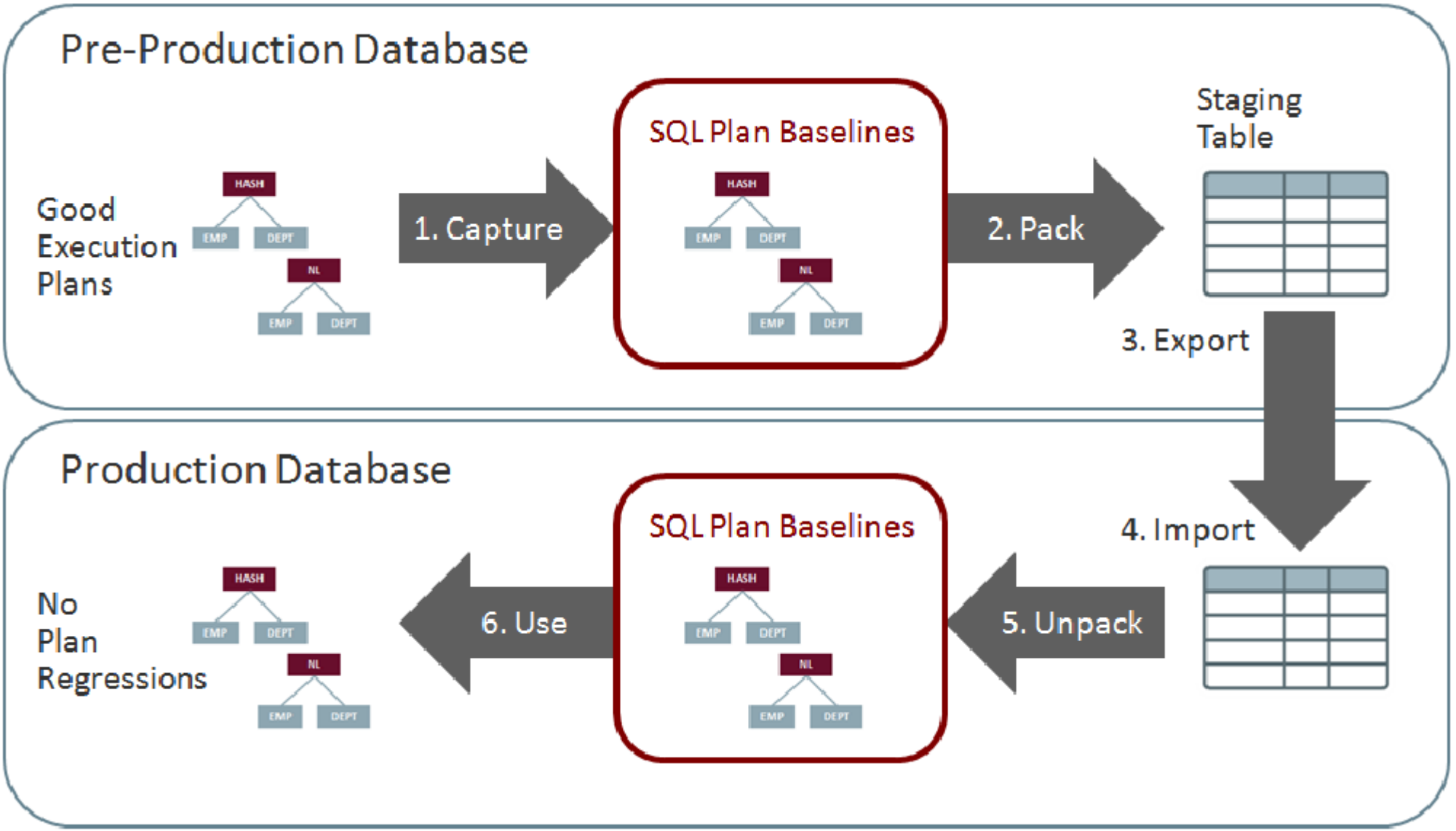
Staging (Exporting) Baselines

```
declare
   l_plan_count number := 0;
begin
   dbms_spm.create_stgtab_baseline (
      table_owner => 'DEMO_OWNER',
      table_name => 'DEMO_STGTAB');

   l_plan_count := l_plan_count + dbms_spm.pack_stgtab_baseline (
      table_owner => 'DEMO_OWNER',
      table_name => 'DEMO_STGTAB');
end;
```

Staged Baselines

```
SQL> select sql_handle, obj_name, max(optimizer_cost) as cost
   from spm_test.spm_export_tab
   group by sql_handle, obj_name
   order by max(optimizer_cost) desc;


SQL_HANDLE                OBJ_NAME                                  COST
------------------------- ------------------------------ ----------
SQL_9832fdfb8497d4c2  SQL_PLAN_9hcrxzf29gp6202feb565      19089
SQL_9832fdfb8497d4c2  SQL_PLAN_9hcrxzf29gp62cea8bf8c        759
```

»Transporting« Baselines



https://www.oracle.com/technetwork/database/bi-datawarehousing/twp-sql-plan-mgmt-19c-5324207.pdf

My Recommendations

- Collect baselines before upgrades
- Deploy baselines as part of your software installation
- The right tools make tasks significantly easier
  - Consider the free [https://appm.abakus.si/]

APPM

https://appm.abakus.si/

http://www.abakus.si/