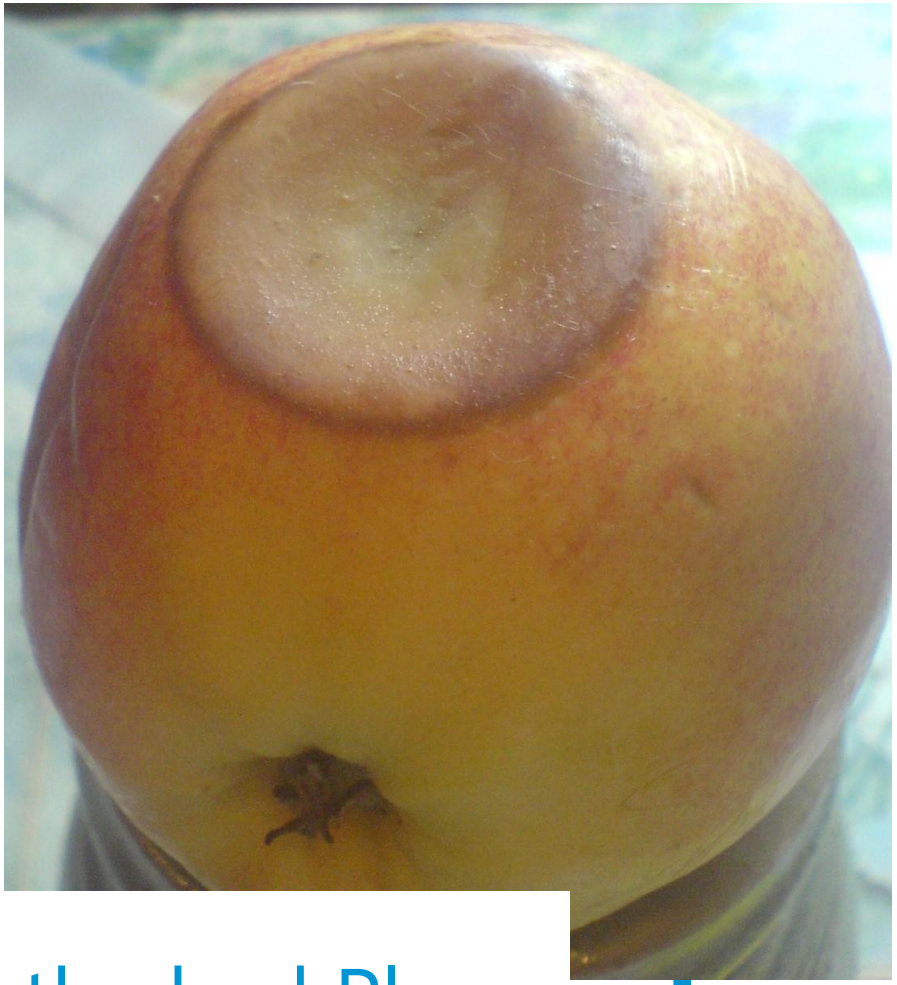

The good Plan and the bad Plan

Lothar Flatz – Senior Principal Consultant

Ich stelle mich vor...

Wer bin ich?

- Über 25 Jahre Oracle Database Erfahrung (beginnend mit Version 5)
- 15 Jahre Oracle Mitarbeiter
- Oak Table Mitglied
- Ex-Real World Performance Group
- Oracle ACE
- Signatur Project: PVSS (CERN)
- Patent US 8103658 B2 zusammen mit Björn Engsig

Über die Diso AG

- Gründung 1996, heute 40 Mitarbeiter
- Sitz in Gümligen bei Bern, Schweiz
- Portfolio:
  - Cloud-Computing
  - Entwicklung
  - IT-Beratung
  - Managed Services
  - Projektumsetzung
  - Systemoptimierung

ORACLE Gold Partner vmware amazon web services sas Atlassian . . .

Startpunkt: AWR

SQL ordered by Elapsed Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100
- %Total - Elapsed Time as a percentage of Total DB time
- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Captured SQL account for 87.6% of Total DB Time (s): 3,286
- Captured PL/SQL account for 87.9% of Total DB Time (s): 3,286

Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	PDB Name	SQL Text
2,883.96	31	93.03	87.76	99.82	0.00	0c049bj3n2rqy	SQL*Plus	PDBORCL	DECLARE CURSOR C1 IS SELECT di...
2,841.80	1,560,726	0.00	86.48	99.89	0.00	d1y981pwzh43u	SQL*Plus	PDBORCL	SELECT MIN(LINE_NR) AS MIN_LNR...
12.65	32	0.40	0.39	99.42	0.00	ac1nz27fr9xf6	SQL*Plus	PDBORCL	SELECT DISTINCT ID FROM MIN_MA...
9.33	1	9.33	0.28	98.17	0.70	572fbaj0fdw2b	SQL*Plus	PDBORCL	select output from table(dbms_...
7.96	126	0.06	0.24	99.97	0.00	7u3jnmmaw8qq1	SQL*Plus	PDBORCL	SELECT SUM(VALUE), COUNT(INSTA...
4.06	1	4.06	0.12	96.57	2.45	8n1jdtxkp83qq	SQL*Plus	PDBORCL	BEGIN dbms_stats.gather_table_...
2.32	5	0.46	0.07	99.63	0.00	1657jvvvkprq9	DBMS_SCHEDULER	PDBORCL	/* SQL Analyze(307, 1) */ SELE...
2.01	1	2.01	0.06	99.49	0.00	5jqsqqybn17sd	SQL*Plus	PDBORCL	select /*+ no_parallel_index(t...
1.60	1	1.60	0.05	58.51	50.76	fkhq9qz4uw1yu		PDBORCL	SELECT /* DS_SVC */ /*+ dynami...
1.37	1	1.37	0.04	99.04	0.00	5pd0tp8xbk5sf		PDBORCL	SELECT /* DS_SVC */ /*+ dynami...

Welcher Plan ?


```
select snap_id,
       plan_hash_value,
       round(buffer_gets_delta/executions_delta,2)
  from dba_hist_sqlstat
 where sql_id ='d1y981pwzh43u';
```

SNAP_ID	PLAN_HASH_VALUE	ROUND(BUFFER_GETS_DELTA/EXECUTIONS_DELTA,2)
1040	2627092648	67,37
1041	1246450326	6,04
1042	2627092648	67,32

Pläne vergleichen


```
SQL> Select * from table (dbms_xplan.display_awr('d1y981pwzh43u'))
  2  /


Plan hash value: 1246450326

-------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |              |       |       |     6 (100)|          |
|   1 |  SORT AGGREGATE                          |              |     1 |     8 |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| MIN_MAX_TEST |    20 |   160 |     6   (0)| 00:00:01 |
|   3 |    INDEX RANGE SCAN                      | FK           |    20 |       |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------


Plan hash value: 2627092648

-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |      |       |       |    22 (100)|          |
|   1 |  SORT AGGREGATE              |      |     1 |     8 |            |          |
|   2 |   FIRST ROW                  |      |     1 |     8 |    22   (0)| 00:00:01 |
|   3 |    INDEX FULL SCAN (MIN/MAX)| PK   |     1 |     8 |    22   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

Planwechsel

Guter Plan

```
-----------------------------------------------------------------------------------------------------------------
Plan hash value: 1246450326

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                             | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                      |              |       |       |     6 (100)|          |
|   1 |  SORT AGGREGATE                       |              |     1 |     8 |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| MIN_MAX_TEST |    20 |   160 |     6   (0)| 00:00:01 |
|   3 |    INDEX RANGE SCAN                   | FK           |    20 |       |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------
```

Schlechter Plan

```
Plan hash value: 2627092648

-------------------------------------------------------------------------------------
| Id  | Operation                    | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |      |       |       |    22 (100)|          |
|   1 |  SORT AGGREGATE              |      |     1 |     8 |            |          |
|   2 |   FIRST ROW                  |      |     1 |     8 |    22   (0)| 00:00:01 |
|   3 |    INDEX FULL SCAN (MIN/MAX)| PK   |     1 |     8 |    22   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

Reperaturvarianten

- SQL Plan Baseline
- SQL Profile
- SQL_Patch
- Nachdenken ?????

SQL Plan Baseline / Profiles

- SQL Profiles
  - Statistische Korrekturen (fudge factors)
  - Über SQL Tuning Advisor
  - Verwendet den opt_estimate hint
  - Kann auch manuell erzeugt
- Baselines
  - Soll negative Planveränderung verhindern
  - Erlaubt mehrere Pläne
  - Evolutionsmechanismus
  - Kennt plan_hash_value
  - Quelle: Kerry Osborne

SQL Plan Baseline

- Übertragung von Plänen von einem Statement auf ein anderes
- Neuer Plan muss im Cursor Cache sein
- Eventuell Statements mit Hints neu laufen lassen
- Script coe_load_sql_baseline.sql von Carlos Sierra

- https://github.com/carlos-sierra/cscripts/blob/master/sql/spm/coe_load_sql_baseline.sql

SQL Plan Baseline

- Neues Script
- Kann Plan auch aus dem AWR holen
- Zeigt eine Übersicht der Pläne
- Script create_sql_plan_baseline.sql von Carlos Sierra

- https://carlos-sierra.net/2017/12/01/creating-a-sql-plan-baseline-from-cursor-cache-or-awr/

SQL Profile

- Zwingender und einfacher als Baseline
- Kann Plan auch aus dem AWR holen
- Script coe_xfr_sql_profile.sql von Carlos Sierra

- https://github.com/carlos-sierra/cscripts/blob/master/sql/spm/coe_xfr_sql_profile.sql

Sql_patch <= 12.1

- Interne Funktion
- Hints können nur eine Zeile umfassen
- Erlaubt auch in SE

Outline generieren AWR


```
select * from table(dbms_xplan.display_awr('d1y981pwzh43u', 1246450326, null,'BASIC +OUTLINE'))

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
SQL_ID d1y981pwzh43u
--------------------
SELECT MIN(LINE_NR) AS MIN_LNR FROM MIN_MAX_TEST T WHERE ID = :B1

Plan hash value: 1246450326

-------------------------------------------------------------
| Id  | Operation                           | Name         |
-------------------------------------------------------------
|   0 | SELECT STATEMENT                    |              |
|   1 |  SORT AGGREGATE                     |              |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| MIN_MAX_TEST |
|   3 |    INDEX RANGE SCAN                 | FK           |
-------------------------------------------------------------

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

  /*+
      BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('12.1.0.2')
      DB_VERSION('12.1.0.2')
      ALL_ROWS
      OUTLINE_LEAF(@"SEL$1")
      INDEX_RS_ASC(@"SEL$1" "T"@"SEL$1" ("MIN_MAX_TEST"."ID"))
      BATCH_TABLE_ACCESS_BY_ROWID(@"SEL$1" "T"@"SEL$1")
      END_OUTLINE_DATA
  */
```

Outline generieren Statspack


```
select * from table(dbms_xplan.display (table_name   => 'perfstat.stats$sql_plan', statement_id => null, filter_preds =>
'plan_hash_value =  1246450326', format      =>'BASIC +OUTLINE'));

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

-------------------------------------------------------------
| Id  | Operation                           | Name         |
-------------------------------------------------------------
|   0 | SELECT STATEMENT                    |              |
|   1 |  SORT AGGREGATE                     |              |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| MIN_MAX_TEST |
|   3 |    INDEX RANGE SCAN                 | FK           |
--------------------------------------------------------------------------------------------------------------------------------------------------------
```

- Leider wird kein outline generiert, aber immerhin ein Plan
- Ein kleiner Trick ist nötig (Danke Oraculix) :

https://oraculix.com/2016/06/21/historical-sql-plan-from-statspack-using-dbms_xplan-in-12c/

Sql_patch <= 12.1


```
DECLARE
  v_sqltext VARCHAR2(10000);
BEGIN
  SELECT sql_fulltext
    INTO v_sqltext
    FROM v$sql
    WHERE sql_id='d1y981pwzh43u';
  SYS.DBMS_SQLDIAG_INTERNAL.i_create_patch(
    sql_text  => v_sqltext,
    hint_text => 'INDEX_RS_ASC(@"SEL$1" "T"@"SEL$1"
("MIN_MAX_TEST"."ID")) BATCH_TABLE_ACCESS_BY_ROWID(@"SEL$1"
"T"@"SEL$1")',
    name      => 'index_id_sql_patch');
END;
/
```

Sql_patch >= 12.1

- regulare Funktion
- Hints können mehr eine Zeile umfassen
- Erlaubt auch in SE
- Mit sql_text oder sql_id

Sql_patch >= 12.2


```
BEGIN
  SYS.DBMS_SQLDIAG.create_sql_patch(
    sql_id  => 'd1y981pwzh43u',
    hint_text => 'INDEX_RS_ASC(@"SEL$1" "T"@"SEL$1"
("MIN_MAX_TEST"."ID")) BATCH_TABLE_ACCESS_BY_ROWID(@"SEL$1"
"T"@"SEL$1")',
    name       => 'index_id_sql_patch');
END;
/
```

Nachdenken

- Es fehlt ein geeigneter Index
- Index erzeugen
- Plan ändert sich
- Plan stabil und schneller als bisher

Vielen Dank für Ihre Aufmerksamkeit!

Für Rückfragen stehen wir Ihnen gerne zur Verfügung.

Lothar Flatz
Diso AG
Morgenstrasse 1
3073 Gümligen

Tel.: +41 31 958 90 90
info@diso.ch
www.diso.ch