

Cardinality Analysis

– a life saver for DBAs and Developers

Michelle Deng
sanofi aventis
Q306

Speaker Qualifications

- Michelle Deng
Sr. Oracle DBA consultant at sanofi aventis

Overview

- What is cardinality?

CBO estimate of the number of rows produced by a row source or combination of row sources.

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		38	2508	10228 (28)	00:02:24
1	PX COORDINATOR					
2	PX SEND QC (RANDOM)	:TQ10002	38	2508	10228 (28)	00:02:24
3	SORT GROUP BY		38	2508	10228 (28)	00:02:24
4	PX RECEIVE		38	2508	10228 (28)	00:02:24
5	PX SEND HASH	:TQ10001	38	2508	10228 (28)	00:02:24
6	SORT GROUP BY		38	2508	10228 (28)	00:02:24
* 7	HASH JOIN		36M	2299M	8338 (12)	00:01:57
8	BUFFER SORT					
9	PX RECEIVE		39	1638	39 (6)	00:00:01
10	PX SEND BROADCAST	:TQ10000	39	1638	39 (6)	00:00:01
* 11	TABLE ACCESS FULL	PROD_FCTR_MV	39	1638	39 (6)	00:00:01
12	PX BLOCK ITERATOR		36M	841M	8127 (10)	00:01:54
* 13	TABLE ACCESS FULL	WEEK_FACT	36M	841M	8127 (10)	00:01:54

Overview cont'd

- How important is cardinality?

Cardinality drives

- Access Paths

Full Table Scan, Index Lookup, Rowid

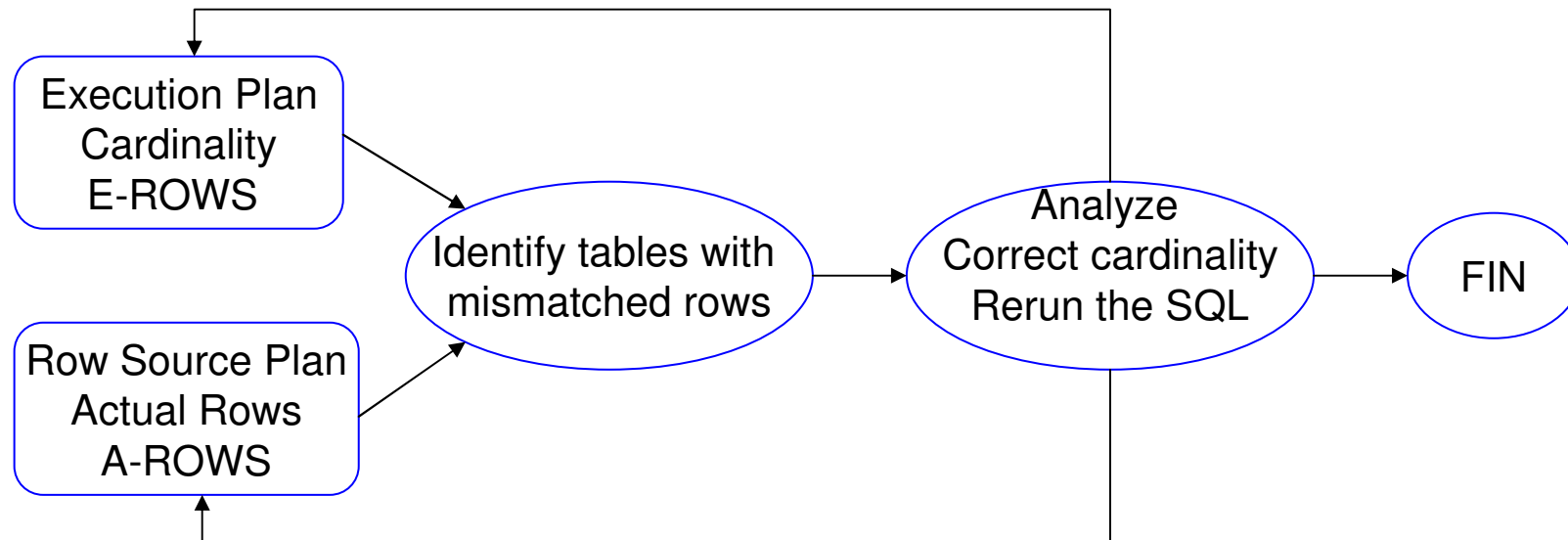
- Join Method

Nested loop, Sort merge, Hash join, Cartesian

- Join Order

Overview cont'd

- What is cardinality analysis?



SQL Plan/Statistics Utilities

E-ROWS

A-ROWS

Explain plan
Autotrace

Dynamic Views
SQLTXPLAN
DBMS_XPLAN

Tkprof
TRCANLZR
Manual select count

SQL Plan/Statistics Utilities cont'd

Dynamic Views

- V\$SQL, V\$SESSION
sql_id, child_number
- V\$SQL_PLAN_STATISTICS_ALL
cardinality -> E-ROWS
last_output_rows -> A-ROWS
 - 10046 EVENT
 - alter session set statistics_level=all
 - /*+ gather_plan_statistics */

SQL Plan/Statistics Utilities cont'd

DBMS_XPLAN

- `select * from table(dbms_xplan.display_cursor(null,null, 'ALL'));`
 - plan of the last executed SQL
- `select * from TABLE(dbms_xplan.display_cursor('&SQL_ID', &CHILD_NUMBER));`
 - plan of particular SQL in memory
- `select * from TABLE(dbms_xplan.display_cursor('&SQL_ID', &CHILD_NUMBER, 'ALLSTATS LAST'));`
 - E-ROWS and A-ROWS with `gather_plan_statistics` or `alter session set statistics_level=all`

Case Study - Issue

```
SQL> SELECT 5576, 'ERTS', 777778, SYSDATE, 'WEEKLY', '00006',
 2 'WEEKLY_SECTOR1', SUM (nrx), SUM (trx), SUM (nun),
 3 SUM (tun), COUNT (1), qtr
 4 FROM WEEKLY_SECTOR1_VIEW
 5 GROUP BY 5576,
 6 'ERTS',
 7 777778,
 8 SYSDATE,
 9 'WEEKLY',
10 '00006',
11 'WEEKLY_SECTOR1',
12 qtr;
```

Elapsed: 21:55:38.08

```
CREATE OR REPLACE FORCE VIEW WEEKLY_SECTOR1_VIEW AS
SELECT col1, col2, ..., SUM(col11), SUM(col12), ...
FROM   week_dim wk, prod_fctr_mv p, mkt_channel channel_lookup,
      mkt_period mktgui, cust_period_bridge sib, cust_addr dfca,
      week_fact f,
      (SELECT max(week_rowid) AS max_week_rowid
       FROM week_fact WHERE mkt_ssk = '00006') maxweek
WHERE  f.mkt_ssk = '00006' AND f.week_rowid = wk.week_rowid
      AND f.week_rowid >= (maxweek.max_week_rowid - 51) AND p.mkt_id = 3
      AND f.mkt_ssk = p.extract_mkt_ssk AND p.hier_type = 'CORP'
      AND f.prod_ssk = p.prod_ssk
      AND channel_lookup.channel_rowid = f.channel_rowid
      AND mktgui.extract_type_cd = 'EARLYVIEW'
      AND channel_lookup.mkt_prd_gui_rowid = mktgui.mkt_prd_gui_rowid
      AND mktgui.mkt_ssk = p.mkt_ssk AND mktgui.mkt_ssk = '00006'
      AND sib.cust_rowid = f.cust_rowid AND dfca.cust_cct_id = sib.cct_id
      AND dfca.period_cd = sib.period_cd
      AND mktgui.extract_prd_cd = sib.period_cd
GROUP BY col1, col2, ...;
```

Case Study – Analysis

Id	Operation	Name	Rows	Bytes	TempSp	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT		3	177		11187 (8)	00:02:37		
1	SORT GROUP BY		3	177		11187 (8)	00:02:37		
2	VIEW	WEEKLY_SECTOR1_VIEW	3607	207K		11185 (8)	00:02:37		
3	SORT GROUP BY		3607	1025K	2224K	11185 (8)	00:02:37		
4	NESTED LOOPS		3607	1025K		11184 (8)	00:02:37		
* 5	HASH JOIN		3526	919K		10201 (8)	00:02:23		
6	TABLE ACCESS FULL	WEEK_DIM	522	6264		3 (0)	00:00:01		
7	NESTED LOOPS		3526	878K		10197 (8)	00:02:23		
8	NESTED LOOPS		3610	828K		9190 (9)	00:02:09		
* 9	HASH JOIN		1	183		8171 (10)	00:01:55		
10	NESTED LOOPS		1	156		8169 (10)	00:01:55		
11	NESTED LOOPS		1	67		8167 (10)	00:01:55		
12	VIEW		1	13		8127 (10)	00:01:54		
13	SORT AGGREGATE		1	10					
14	PX COORDINATOR								
15	PX SEND QC (RANDOM)	:TQ10000	1	10					
16	SORT AGGREGATE		1	10					
17	PX BLOCK ITERATOR		36M	350M		8127 (10)	00:01:54	5	5
* 18	TABLE ACCESS FULL	WEEK_FACT	36M	350M		8127 (10)	00:01:54	5	5
* 19	TABLE ACCESS FULL	PROD_FCTR_MV	1	54		39 (6)	00:00:01		
* 20	TABLE ACCESS BY INDEX ROWID	MKT_PERIOD	1	89		2 (0)	00:00:01		
* 21	INDEX RANGE SCAN	PK_MKT_IDX	4			1 (0)	00:00:01		
22	TABLE ACCESS FULL	MKT_CHANNEL	199	5373		2 (0)	00:00:01		
23	PARTITION RANGE SINGLE		12249	622K		1019 (2)	00:00:15	5	5
* 24	TABLE ACCESS BY LOCAL INDEX ROWID	WEEK_FACT	12249	622K		1019 (2)	00:00:15	5	5
* 25	INDEX RANGE SCAN	WEEK_FACT_IDX_IE3	330K			51 (6)	00:00:01	5	5
26	PARTITION LIST ITERATOR		1	20		1 (0)	00:00:01	KEY	KEY
27	TABLE ACCESS BY LOCAL INDEX ROWID	CUST_PERIOD_BRIDGE	1	20		1 (0)	00:00:01	KEY	KEY
* 28	INDEX UNIQUE SCAN	CUST_PERIOD_BRIDGE_QTR	1			0 (0)	00:00:01	KEY	KEY
29	PARTITION LIST ITERATOR		1	24		1 (0)	00:00:01	KEY	KEY
30	TABLE ACCESS BY LOCAL INDEX ROWID	CUST_ADDR	1	24		1 (0)	00:00:01	KEY	KEY
* 31	INDEX UNIQUE SCAN	CUST_ADDR_IDX1	1			0 (0)	00:00:01	KEY	KEY

Predicate Information (identified by operation id):

```

5 - access("P"."WEEK_ROWID"="WK"."WEEK_ROWID")
9 - access("CHANNEL_LOOKUP"."MKT_PRD_GUI_ROWID"="MKTGUI"."MKT_PRD_GUI_ROWID")
18 - filter("MKT_SSK"='00006')
19 - filter("P"."MKT_ID"=3 AND "P"."EXTRACT_MKT_SSK"='00006' AND "P"."MKT_SSK"='00006' AND "P"."HIER_TYPE"='CORP')

```

Case Study – Analysis cont'd

Table Name	Cardinality (autotrace)	Actual Rows (manual select)
WEEK_DIM	522	522
WEEK_FACT	36M	36641302
PROD_FCTR_MV	1	42
MKT_CHANNEL	199	199

```
SQL> select count(*) from PROD_FCTR_MV p
  2  where P.MKT_ID=3 AND P.EXTRACT_MKT_SSK='00006'
  3  AND P.MKT_SSK='00006' AND P.HIER_TYPE='CORP'
  4  /

COUNT(*)
-----
         42
```

```
-----
| Id | Operation                | Name                | Rows  | Bytes | C
-----
|  0 | SELECT STATEMENT         |                     |      1 |    24 | 
|  1 |   SORT AGGREGATE         |                     |      1 |    24 | 
|*  2 |    TABLE ACCESS FULL    | PROD_FCTR_MV        |      1 |    24 | 
-----
```

Predicate Information (identified by operation id):

```
2 - filter("P"."MKT_ID"=3 AND "P"."EXTRACT_MKT_SSK"='00006' AND
          "P"."MKT_SSK"='00006' AND "P"."HIER_TYPE"='CORP')
```

Case Study – Analysis cont'd

Table/Column Statistics – PROD_FCTR_MV

Table Name	Number of Rows	Blocks	Empty Blocks	Average Space	Chain Count	Average Row Len
PROD_FCTR_MV	4,965	113	0	0	0	305

Column Name	Column Details	Distinct Values	Density	Number Buckets	Number Nulls	HISTOGRAM
EXTRACT_MKT_SSK	VARCHAR2 (5)	100	.00010	100	0	FREQUENCY
MKT_SSK	VARCHAR2 (20)	73	.00011	73	296	FREQUENCY
MKT_ID	NUMBER (16,0)	91	.00010	91	0	FREQUENCY
HIER_TYPE	VARCHAR2 (20)	5	.00010	5	0	FREQUENCY

Case Study – Analysis cont'd

```
SQL> select count(*) from PROD_FCTR_MV p
2  where P.MKT_ID=3
3  ;
```

```
COUNT(*)
-----
      169
```

Execution Plan

Plan hash value: 1269706947

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		1	5
1	SORT AGGREGATE		1	5
* 2	TABLE ACCESS FULL	PROD_FCTR_MV	169	845

```
SQL> select count(*) from PROD_FCTR_MV p
2  where P.EXTRACT_MKT_SSK='00006'
3  ;
```

```
COUNT(*)
-----
      169
```

Execution Plan

Plan hash value: 1269706947

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		1	6
1	SORT AGGREGATE		1	6
* 2	TABLE ACCESS FULL	PROD_FCTR_MV	169	1014

```
SQL> select count(*) from PROD_FCTR_MV p
2  where P.MKT_SSK='00006'
3  ;
```

```
COUNT(*)
-----
      169
```

Execution Plan

Plan hash value: 1269706947

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		1	6
1	SORT AGGREGATE		1	6
* 2	TABLE ACCESS FULL	PROD_FCTR_MV	169	1014

```
SQL> select count(*) from PROD_FCTR_MV p
2  where P.HIER_TYPE='CORP'
3  /
```

```
COUNT(*)
-----
     2441
```

Execution Plan

Plan hash value: 1269706947

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		1	7
1	SORT AGGREGATE		1	7
* 2	TABLE ACCESS FULL	PROD_FCTR_MV	2441	17087

Case Study – Analysis cont'd

- Strong column correlation
- Limitation of CBO – assume no column correlation
- CBO overestimate the selectivity

```
SELECT distinct mkt_id, mkt_ssk, extract_mkt_ssk  
FROM PROD_FCTR_MV
```

where mkt_id = 3;

One record returned.

Case Study – Solution

- Hints – use_hash ...
- DBMS_STATS.SET_COLUMN_STATS
density=>1 for 'extract_mkt_ssk', 'mkt_ssk'
- optimizer_dynamic_sampling = 4

optimizer_dynamic_sampling = 4

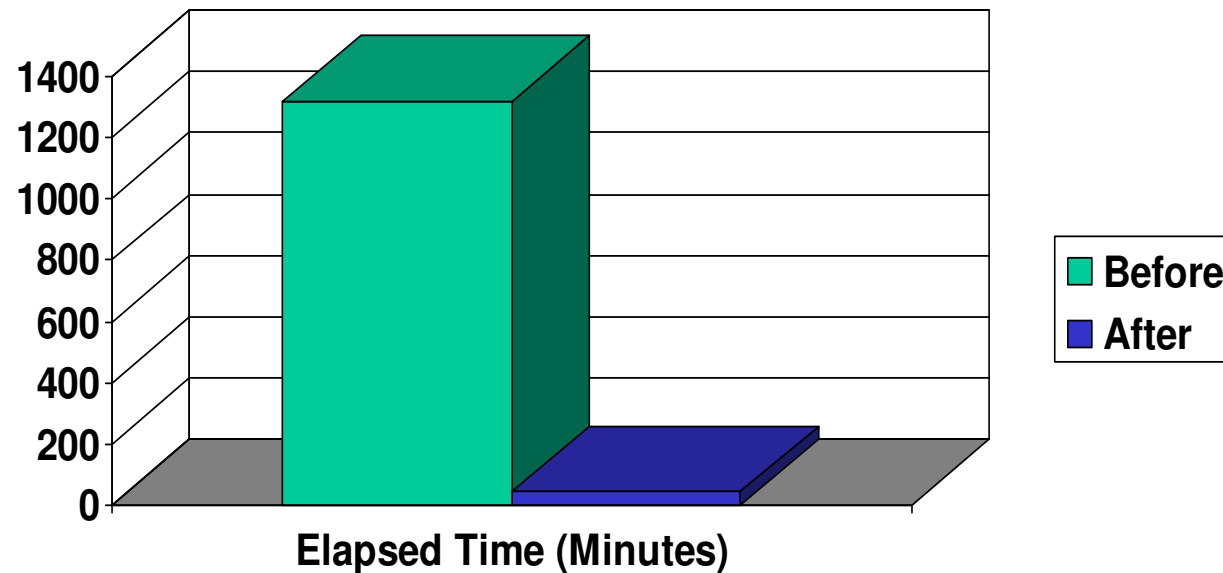
Elapsed: 00:43:31.94

Case Study – Solution cont'd

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT		3	177		16633 (10)	00:03:53		
1	PX COORDINATOR								
2	PX SEND QC (RANDOM)	:TQ20005	3	177		16633 (10)	00:03:53		
3	SORT GROUP BY		3	177		16633 (10)	00:03:53		
4	PX RECEIVE		3	177		16633 (10)	00:03:53		
5	PX SEND HASH	:TQ20004	3	177		16633 (10)	00:03:53		
6	SORT GROUP BY		3	177		16633 (10)	00:03:53		
7	VIEW	WEEKLY_SECTOR1_VIEW	1117K	62M		16587 (10)	00:03:53		
8	SORT GROUP BY		1117K	310M	658M	16587 (10)	00:03:53		
9	PX RECEIVE		1117K	310M		16542 (10)	00:03:52		
10	PX SEND HASH	:TQ20003	1117K	310M		16542 (10)	00:03:52		
* 11	HASH JOIN		1117K	310M		16542 (10)	00:03:52		
12	PX BLOCK ITERATOR		7327K	167M		4480 (8)	00:01:03	1	5
13	TABLE ACCESS FULL	CUST_ADDR	7327K	167M		4480 (8)	00:01:03	1	5
14	BUFFER SORT								
15	PX RECEIVE		1092K	278M		12005 (10)	00:02:49		
16	PX SEND BROADCAST LOCAL	:TQ20002	1092K	278M		12005 (10)	00:02:49		
* 17	HASH JOIN		1092K	278M		12005 (10)	00:02:49		
18	BUFFER SORT								
19	PX RECEIVE		522	6264		3 (0)	00:00:01		
20	PX SEND BROADCAST	:TQ20000	522	6264		3 (0)	00:00:01		
21	TABLE ACCESS FULL	WEEK_DIM	522	6264		3 (0)	00:00:01		
* 22	HASH JOIN		1092K	265M		11996 (10)	00:02:48		
23	PX BLOCK ITERATOR		12M	229M		2716 (9)	00:00:39	KEY(SQ)	KEY(SQ)
24	TABLE ACCESS FULL	CUST_PERIOD_BRIDGE	12M	229M		2716 (9)	00:00:39	KEY(SQ)	KEY(SQ)
25	BUFFER SORT								
26	PX RECEIVE		1118K	250M		9216 (9)	00:02:10		
27	PX SEND BROADCAST LOCAL	:TQ20001	1118K	250M		9216 (9)	00:02:10		
* 28	HASH JOIN		1118K	250M		9216 (9)	00:02:10		
29	TABLE ACCESS FULL	MKT_CHANNEL	199	5373		2 (0)	00:00:01		
* 30	HASH JOIN		1826K	362M		9205 (9)	00:02:09		
* 31	TABLE ACCESS BY INDEX ROWID	MKT_PERIOD	1	89		2 (0)	00:00:01		
* 32	INDEX RANGE SCAN	PK_MKT_IDX	4			1 (0)	00:00:01		
* 33	HASH JOIN		1826K	207M		9194 (9)	00:02:09		
* 34	TABLE ACCESS FULL	PROD_FCTR_MV	39	2106		39 (6)	00:00:01		
35	NESTED LOOPS		1837K	113M		9146 (9)	00:02:09		
36	VIEW		1	13		8127 (10)	00:01:54		
37	SORT AGGREGATE		1	10					
38	PX COORDINATOR								
39	PX SEND QC (RANDOM)	:TQ10000	1	10					
40	SORT AGGREGATE		1	10					
41	PX BLOCK ITERATOR		36M	350M		8127 (10)	00:01:54	5	5
* 42	TABLE ACCESS FULL	WEEK_FACT	36M	350M		8127 (10)	00:01:54	5	5
43	PARTITION RANGE SINGLE		1837K	91M		1019 (2)	00:00:15	5	5
* 44	TABLE ACCESS BY LOCAL INDEX ROWID	WEEK_FACT	1837K	91M		1019 (2)	00:00:15	5	5
* 45	INDEX RANGE SCAN	WEEK_FACT_IDX_IES	330K			51 (6)	00:00:01	5	5

Case Study – Summary

- Optimizer_dynamic_sampling changed from 2 to 4
- Query elapsed time improved from ~22 hours to ~44 minutes



Items Learned in this Session

- Quick problem identification of suboptimal query plan using cardinality analysis
- Approaches to correct cardinality
- Usage of Oracle query tuning utilities

Questions?

Thank You

Please complete the evaluation form.

Speaker Name: Michelle Deng

Session name: Cardinality Analysis – a life saver
for DBAs and Developers

Session#: Q306