第七届

数据技术嘉年华

Data Technology Carnival

云·数据·智能 - 数聚价值智胜未来

关注公众号回复help，可获取更多经典学习资料和文档，电子书

黄廷忠（认真就输）

- 就职于云和恩墨
- 7年专职ORACLE数据库经验
- 一直服务于电信运营商
- 6年电信行业驻场经验
- 个人技术博客:www.htz.pw

子查询优化的最佳实践

tingzhong.huang@enmotech.com

18081072613

目录

SQL优化的核心思想

SQL优化的核心思想

核心思想

减少IO次数（物理和逻辑）

优化目的

减少SQL执行时间

SQL慢在那里

SQL慢在那里-常用方法

- 查看SQL语句
- 查看执行计划
- 查看10053
- 查看10046
- 11g以后sql monitor查看
- dbms_profile（pl/sql）
- hint(gather_plan_statistics)
- autotrace

SQL慢在那里-最佳方法

如果SQL语句运行大于6S，可以利用sql monitor来快速定位SQL语句慢在什么地方

如果小于6S或者是毫秒级别的SQL语句呢？

/*+ sql_monitor*/ /*+ gather_plan_statistics*/这样的提示？

其实可以利用执行计划于ASH采样的数据结合，来快速定位SQL语句疼点

SQL慢在那里-最佳方法

Id	Operation	Name	Rows
0	SELECT STATEMENT		
1	SORT ORDER BY		1
* 2	FILTER		
3	NESTED LOOPS		1
4	NESTED LOOPS		1
5	NESTED LOOPS		1
6	NESTED LOOPS		103
7	INLIST ITERATOR		
8	TABLE ACCESS BY INDEX ROWID	SYS_DOMAIN	1
* 9	INDEX RANGE SCAN	IDX_SYS_DOMAIN_1	1
10	TABLE ACCESS FULL	STAFF	266K
* 11	TABLE ACCESS BY INDEX ROWID	A_PAY_ASSISTANT	1
* 12	INDEX RANGE SCAN	IDX_A_P_A_002	33
* 13	INDEX UNIQUE SCAN	PK_BSS_ORG	1
14	TABLE ACCESS BY INDEX ROWID	BSS_ORG	1

SQL慢在那里-最佳方法

Id	Operation	Name	Rows	
0	SELECT STATEMENT			CPU(3)(.04%)
1	SORT ORDER BY		1	
* 2	FILTER			
3	NESTED LOOPS		1	
4	NESTED LOOPS		1	
5	NESTED LOOPS		1	CPU(9)(.12%)
6	NESTED LOOPS		103	CPU(16)(.21%)
7	INLIST ITERATOR			
8	TABLE ACCESS BY INDEX ROWID	SYS_DOMAIN	1	
* 9	INDEX RANGE SCAN	IDX_SYS_DOMAIN_1	1	
10	TABLE ACCESS FULL	STAFF	266K	CPU(860)(11.07%)
* 11	TABLE ACCESS BY INDEX ROWID	A_PAY_ASSISTANT	1	gc current block congeste(1)(.01%)
				CPU(3039)(39.12%)
* 12	INDEX RANGE SCAN	IDX_A_P_A_002	33	latch: cache buffers chai(1)(.01%)
				CPU(3840)(49.43%)
* 13	INDEX UNIQUE SCAN	PK_BSS_ORG	1	
14	TABLE ACCESS BY INDEX ROWID	BSS_ORG	1	

SQL慢在那里-最佳方法

Id	Operation	Name	Rows
0	SELECT STATEMENT		
1	SORT ORDER BY		1
* 2	FILTER		
3	NESTED LOOPS		1
4	NESTED LOOPS		1
5	NESTED LOOPS		1
6	NESTED LOOPS		103
7	INLIST ITERATOR		
8	TABLE ACCESS BY INDEX ROWID	SYS_DOMAIN	1
* 9	INDEX RANGE SCAN	IDX_SYS_DOMAIN_1	1
10	TABLE ACCESS FULL	STAFF	266K
* 11	TABLE ACCESS BY INDEX ROWID	A_PAY_ASSISTANT	1
* 12	INDEX RANGE SCAN	IDX_A_P_A_002	33
* 13	INDEX UNIQUE SCAN	PK_BSS_ORG	1
14	TABLE ACCESS BY INDEX ROWID	BSS_ORG	1

Id	Operation	Name	Rows	
0	SELECT STATEMENT			CPU(3)(.04%)
1	SORT ORDER BY		1	
* 2	FILTER			
3	NESTED LOOPS		1	
4	NESTED LOOPS		1	
5	NESTED LOOPS		1	CPU(9)(.12%)
6	NESTED LOOPS		103	CPU(16)(.21%)
7	INLIST ITERATOR			
8	TABLE ACCESS BY INDEX ROWID	SYS_DOMAIN	1	
* 9	INDEX RANGE SCAN	IDX_SYS_DOMAIN_1	1	
10	TABLE ACCESS FULL	STAFF	266K	CPU(860)(11.07%)
* 11	TABLE ACCESS BY INDEX ROWID	A_PAY_ASSISTANT	1	gc current block congeste(1)(.01%) CPU(3039)(39.12%)
* 12	INDEX RANGE SCAN	IDX_A_P_A_002	33	latch: cache buffers chai(1)(.01%) CPU(3840)(49.43%)
* 13	INDEX UNIQUE SCAN	PK_BSS_ORG	1	
14	TABLE ACCESS BY INDEX ROWID	BSS_ORG	1	

SQL慢在那里-最佳方法-限制

对正在执行的且在ASH里面只有当前正在执行的数据的SQL，只能定位已经执行过的步骤的最慢位置，未执行的步骤不能定位，SQL MONITOR也存在同样问题。

OR遇到子查询的改写

OR遇到子查询的改写

```
create table system.test1 as select * from dba_objects;
create table system.test2 as select * from dba_objects;
create table system.test3 as select * from dba_objects;
```

```
SELECT owner, object_type
  FROM SYSTEM.test1 a
 WHERE     (    a.object_id IN (51577, 51575,51352)
            AND a.owner IN (SELECT OWNER
                              FROM SYSTEM.test3
                             WHERE object_type = 'TABLE'))
       OR (    a.object_type = 'TABLE'
           AND a.DATA_OBJECT_ID IN (SELECT DATA_OBJECT_ID
                                      FROM SYSTEM.test2
                                     WHERE object_name IN ('EMP', 'BONUS','DEPT')))
/
```

OR遇到子查询的改写-定位瓶颈

```
SYS                            JAVA CLASS
SYS                            JAVA CLASS
SCOTT                          TABLE
SCOTT                          TABLE
SCOTT                          TABLE

| Id  | Operation          | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
|   0 | SELECT STATEMENT   |       |    76 |  4104 |   291   (1)| 00:00:04 |
|*  1 |  FILTER            |       |       |       |            |          |
|   2 |   TABLE ACCESS FULL| TEST1 | 75597 |  3986K|   291   (1)| 00:00:04 |
|*  3 |   TABLE ACCESS FULL| TEST3 |    31 |   868 |    11   (0)| 00:00:01 |
|*  4 |   TABLE ACCESS FULL| TEST2 |     1 |    79 |   290   (1)| 00:00:04 |

Predicate Information (identified by operation id):

  1 - filter("A"."OBJECT_ID"=51352 OR "A"."OBJECT_ID"=51575 OR
              "A"."OBJECT_ID"=51577) AND  EXISTS (SELECT 0 FROM "SYSTEM"."TEST3"
              "TEST3" WHERE "OWNER"=:B1 AND "OBJECT_TYPE"='TABLE') OR
              "A"."OBJECT_TYPE"='TABLE' AND  EXISTS (SELECT 0 FROM "SYSTEM"."TEST2"
              "TEST2" WHERE "DATA_OBJECT_ID"=:B2 AND ("OBJECT_NAME"='BONUS' OR
              "OBJECT_NAME"='DEPT' OR "OBJECT_NAME"='EMP')))

Statistics

    2669291  consistent gets
          5  rows processed
```

```
SYS@htz1123 > @sql_monitor_by_sqlid.sql
Enter value for sqlid: 2h68pg6atawkj

Global Stats

| Elapsed |   Cpu   |    IO    |  Other   | Fetch | Buffer | Read | Read  |
| Time(s) | Time(s) | Waits(s) | Waits(s) | Calls |  Gets  | Reqs | Bytes |
|    6.47 |    6.43 |     0.00 |     0.04 |     1 |     2M |   61 |   6MB |

SQL Plan Monitoring Details (Plan Hash Value=599902274)

| Id |      Operation       | Name  | Execs | Rows (Actual) | Activity (%) | Activity Detail (# samples) |
|  0 | SELECT STATEMENT     |       |     1 |             5 |              |                             |
|  1 |   FILTER             |       |     1 |             5 |              |                             |
|  2 |     TABLE ACCESS FULL| TEST1 |     1 |         74510 |              |                             |
|  3 |     TABLE ACCESS FULL| TEST3 |     2 |             1 |              |                             |
|  4 |     TABLE ACCESS FULL| TEST2 |  2504 |             3 |       100.00 | Cpu (8)                     |
```

OR遇到子查询的改写-改写UNION

OR是否改写成UNION?

```
SYS@htz1123 > SELECT owner, object_type
  2    FROM SYSTEM.test1 a
  3   WHERE    (    a.object_id IN (51577, 51575,51352)
  4             AND a.owner IN (SELECT OWNER
  5                               FROM SYSTEM.test3
  6                              WHERE object_type = 'TABLE'))
  7  union
  8  SELECT owner, object_type
  9    FROM SYSTEM.test1 a
 10         WHERE (    a.object_type = 'TABLE'
 11             AND a.DATA_OBJECT_ID IN (SELECT DATA_OBJECT_ID
 12                                        FROM SYSTEM.test2
 13                                       WHERE object_name IN ('EMP', 'BONUS','DEPT')))
 14  /

OWNER                          OBJECT_TYPE
------------------------------ -----------
SCOTT                          TABLE
SYS                            JAVA CLASS
```

```
Statistics
----------------------------------------------------------
         17  recursive calls
          0  db block gets
       4531  consistent gets
          0  physical reads
          0  redo size
        671  bytes sent via SQL*Net to client
        519  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          5  sorts (memory)
          0  sorts (disk)
          2  rows processed
```

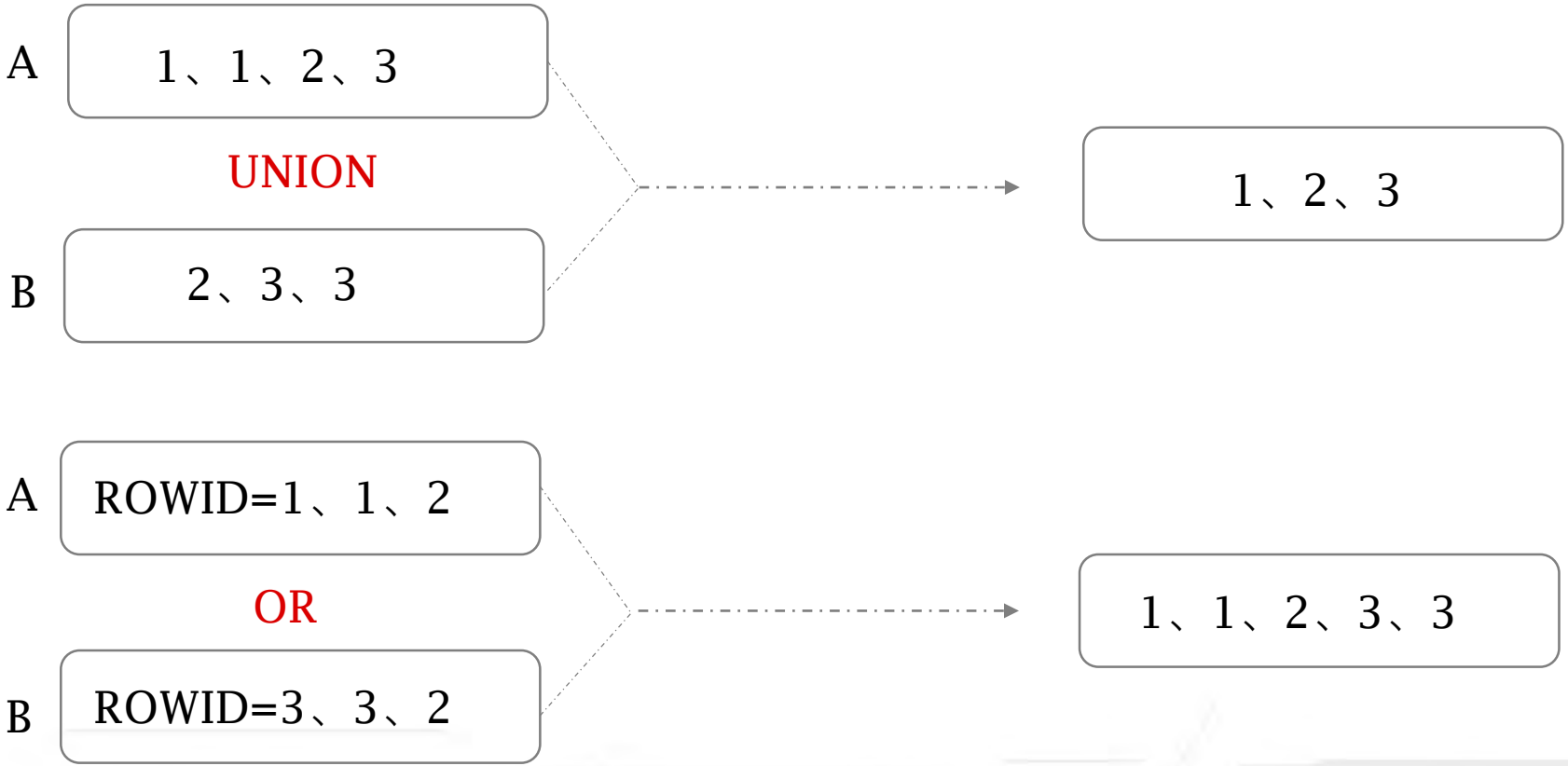
OR遇到子查询的改写-改写UNION

1 OR

两个结果集求和，并且在两结果集之间的重复的物理行记录只保留一份

2 UNION

两个结果集求和，求和后结果集的重复记录只保留一份


A
1、1、2、3
UNION
B
2、3、3
1、2、3
A
ROWID=1、1、2
OR
B
ROWID=3、3、2
1、1、2、3、3

OR遇到子查询的改写-改写方案一

```
SELECT owner, object_type
 FROM (SELECT ROWID, owner, object_type
         FROM SYSTEM.test1 a
        WHERE (    a.object_id IN (51577, 51575, 51352)
               AND a.owner IN (SELECT OWNER
                                 FROM SYSTEM.test3
                                WHERE object_type = 'TABLE'))
       UNION
       SELECT ROWID, owner, object_type
         FROM SYSTEM.test1 a
        WHERE     a.object_type = 'TABLE'
              AND a.DATA_OBJECT_ID IN
                     (SELECT DATA_OBJECT_ID
                        FROM SYSTEM.test3
                       WHERE object_name IN ('EMP', 'BONUS', 'DEPT')))
```

OR遇到子查询的改写-改写方案一

下面这条SQL语句那应该怎么改写呢？

```
SQL> SELECT COUNT (*)
  2    FROM (SELECT a.object_type
  3            FROM SYSTEM.test1 a, SYSTEM.test2 b
  4           WHERE     a.object_type = b.object_type
  5                 AND (a.owner = 'SYS' OR a.owner = 'SYSTEM'));

  COUNT(*)
----------
  21402120
```

OR遇到子查询的改写-改写方案一

改写方法如下：

```
SQL> SELECT COUNT (*)
  2    FROM (SELECT a.object_type
  3            FROM SYSTEM.test1 a, SYSTEM.test2 b
  4           WHERE     a.object_type = b.object_type
  5                 AND (a.owner = 'SYS' OR a.owner = 'SYSTEM'));

  COUNT(*)
----------
  21402120
```

```
SQL> SELECT COUNT (*)
  2    FROM (SELECT a.ROWID, b.ROWID, a.object_type
  3            FROM SYSTEM.test1 a, SYSTEM.test2 b
  4           WHERE a.object_type = b.object_type AND a.owner = 'SYS'
  5          UNION
  6          SELECT a.ROWID, b.ROWID, a.object_type
  7            FROM SYSTEM.test1 a, SYSTEM.test2 b
  8           WHERE a.object_type = b.object_type AND a.owner = 'SYSTEM');

  COUNT(*)
----------
  21402120
```

OR遇到子查询的改写-改写方案二

```
SELECT owner, object_type
  FROM SYSTEM.test1 a
 WHERE (    a.object_id IN (51577, 51575,51352)
        AND a.owner IN (SELECT OWNER
                          FROM SYSTEM.test3
                         WHERE object_type = 'TABLE'))
UNION ALL
SELECT owner, object_type
  FROM SYSTEM.test1 a
 WHERE     a.object_type = 'TABLE'
       AND a.DATA_OBJECT_ID IN (SELECT DATA_OBJECT_ID
                                  FROM SYSTEM.test3
                                 WHERE object_name IN ('EMP', 'BONUS','DEPT'))
       AND (    LNNVL (a.object_id = 51352)
            AND LNNVL (a.object_id = 51575)
            AND LNNVL (a.object_id = 51577))
```

OR遇到子查询的改写-改写方案三

1 **其它改写方案：**

还有其它很多的改写方案：如利用分析函数来求rowid的最小值，最大值都是可以的

2 **OR在什么情况下可以直接改写成union**

结果集中无物理重复行的时

案例2：BUG永远爱你

BUG永远爱你-OR扩展遇BUG

```
select *
  from (select row_.*, rownum rownum_
          from (SELECT ( SELECT MAX(D.GJSJ)
                           FROM FW_DHSQ D
                          WHERE C.KHSQBS = D.KHSQBS) AS PARAM2
                       A.KFGZDBS AS PARAM7
                  FROM FW_KFGDXX B, FW_KHSQ C, FW_QXYWFJXX A, FW_DHSQ D
                 WHERE B.KHSQBS = C.KHSQBS
                   AND A.KFGZDBS = B.KFGZDBS
                   AND C.KHSQBS = D.KHSQBS
                   AND B.ZFBZ = '0'
                   AND B.YWLBDM IN ('01')
                   AND B.SLZZBM IN
                       (SELECT ZZBM
                          FROM XT_ZZ Z
                         WHERE Z.ZTBZ = '1'
                         START WITH Z.ZZBM = '0609'
                        CONNECT BY Z.SJZZBM = PRIOR Z.ZZBM)
                   AND B.GZDZTDM = '3'
                   AND B.GDSJ >= to_date('2016-01-01','yyyy-mm-dd')
                   AND B.GDSJ <= sysdate
                   AND (((A.DDXCSJ - D.GJSJ) * 24 * 60 > 45 AND A.YDQY = '1') OR
                       ((A.DDXCSJ - D.GJSJ) * 24 * 60 > 90 AND A.YDQY = '2') OR
                       ((A.DDXCSJ - D.GJSJ) * 24 * 60 > 120 AND A.YDQY = '3'))
                   AND A.DDXCSJ IS NOT NULL
                   AND D.GJSJ IS NOT NULL) row_
         where rownum <= 10)
 where rownum_ > 0
```

BUG永远爱你-OR扩展遇BUG

Id	Operation	Name	Rows	Pstart	Pstop
0	SELECT STATEMENT				
1	SORT AGGREGATE		1		
2	TABLE ACCESS BY GLOBAL INDEX ROWID	FW_DHSQ	1	ROW LOCATION	ROW L
3	INDEX RANGE SCAN	IDX_FW_DHSQ_KHSQBS	1		
4	VIEW		3		
5	COUNT STOPKEY				
6	CONCATENATION				
7	FILTER				
8	FILTER				
9	NESTED LOOPS SEMI		1		
10	NESTED LOOPS		1		
11	NESTED LOOPS		1		
12	NESTED LOOPS		5		
13	PARTITION LIST ALL		1	1	12
14	TABLE ACCESS FULL	FW_DHSQ	1	1	12
15	TABLE ACCESS FULL	FW_QXYWFJXX	62		
16	TABLE ACCESS BY GLOBAL INDEX ROWID	FW_KFGDXX	1	ROW LOCATION	ROW L
17	INDEX UNIQUE SCAN	PK_FW_KFGDXX	1		
18	INDEX UNIQUE SCAN	PK_FW_KHSQ	1		
19	VIEW	VW_NSO_1	1		
20	FILTER				
21	CONNECT BY WITH FILTERING				
22	TABLE ACCESS BY INDEX ROWID	XT_ZZ	1		
23	INDEX UNIQUE SCAN	PK_XT_ZZ	1		
24	NESTED LOOPS		3		
25	CONNECT BY PUMP				
26	TABLE ACCESS BY INDEX ROWID	XT_ZZ	3		
27	INDEX RANGE SCAN	IDX_XT_ZZ_SJZZBM	3		

1 2 3 4

BUG永远爱你-OR扩展遇BUG

```
Predicate Information:
----------------------
3 - access("D"."KHSQBS"=:B1)
4 - filter("ROWNUM_">0)
5 - filter(ROWNUM<=10)
7 - filter(ROWNUM<=10)
8 - filter(TO_DATE(' 2016-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')<=SYSDATE@!)
14 - filter("D"."GJSJ" IS NOT NULL)                                    2
15 - filter(("A"."YDQY"='3' AND "A"."DDXCSJ" IS NOT NULL AND ("A"."DDXCSJ"-"D"."GJSJ")*24*60>120))   1
16 - filter(("B"."YWLBDM"='01' AND "B"."GZDZTDM"='3' AND "B"."ZFBZ"='0' AND "B"."GDSJ">=TO_DATE(' 2
17 - access("A"."KFGZDBS"="B"."KFGZDBS")
18 - access("C"."KHSQBS"="D"."KHSQBS")
18 - filter("B"."KHSQBS"="C"."KHSQBS")
19 - filter("B"."SLZZBM"="ZZBM")
20 - filter("Z"."ZTBZ"='1')
21 - access("Z"."SJZZBM"=PRIOR NULL)
23 - access("Z"."ZZBM"='0609')
27 - access("Z"."SJZZBM"="connect$_by$_pump$_009"."PRIOR Z.ZZBM")
```

BUG永远爱你-10053分析原因

OR扩展时
10053的
部分信息

```
Access path analysis for FW_DHSQ
***************************************
SINGLE TABLE ACCESS PATH
  Single Table Cardinality Estimation for FW_DHSQ[D]
  Column (#7): GJSJ(                    1
    AvgLen: 8 NDV: 536768 Nulls: 28243 Density: 0.000002 Min: 2457393 Max: 2457542
  Table: FW_DHSQ Alias: D                    2                    3
    Card: Original: 15238.000000 Rounded: 1 Computed: 0.00 Non Adjusted: 0.00


Access path analysis for FW_DHSQ
***************************************
SINGLE TABLE ACCESS PATH (First K Rows)
  Single Table Cardinality Estimation for FW_DHSQ[D]
  Table: FW_DHSQ Alias: D                    4
    Card: Original: 15238.000000 Rounded: 1 Computed: 0.00 Non Adjusted: 0.00
  Access Path: TableScan

  ****** trying bitmap/domain indexes ******
  ****** finished trying bitmap/domain indexes ******
  Best:: AccessPath: TableScan
```

BUG永远爱你-OR扩展遇BUG

OR不扩展时的
10053部分信息

```
Access path analysis for FW_DHSQ
**************************************
SINGLE TABLE ACCESS PATH
  Single Table Cardinality Estimation for FW_DHSQ[D]
  Column (#7): GJSJ(
    AvgLen: 8 NDV: 536768 Nulls: 28243 Density: 0.000002 Min: 2457393 Max: 2457542
  Table: FW_DHSQ Alias: D
    Card: Original: 587562.000000 Rounded: 559319 Computed: 559319.00 Non Adjusted: 559319.00
  Access Path: TableScan
    Cost: 4477.31 Resp: 4477.31 Degree: 0
      Cost_io: 4443.00 Cost_cpu: 287184596
      Resp_io: 4443.00 Resp_cpu: 287184596
  ****** trying bitmap/domain indexes ******
  ****** finished trying bitmap/domain indexes ******
  Best:: AccessPath: TableScan
         Cost: 4477.31 Degree: 1 Resp: 4477.31 Card: 559319.00 Bytes: 0
```

BUG永远爱你-OR扩展遇BUG

10053的对比结果：

OR展开时遇到ROWNUM出现Card值错误

通过MOS搜索：OR expansion发现下面bug

Bug 15996520 Suboptimal execution plan OR expansion and First K rows

This note gives a brief overview of bug 15996520.
The content was last updated on: 14-FEB-2017
Click here for details of each of the sections below.

Affects:

Product (*Component*)	Oracle Server (Rdbms)
Range of versions *believed* to be affected	(*Not specified*)
Versions *confirmed* as being affected	• 12.1.0.1 (Base Release) • 11.2.0.3
Platforms affected	Generic (all / most platforms affected)

Fixed:

The fix for 15996520 is first included in	• 12.2.0.1 (Base Release) • 12.1.0.2 (Server Patch Set) • 11.2.0.4 (Server Patch Set) • 11.2.0.3 Patch 16 on Windows Platforms

一个分享交流的地方

微信号：eyygle

Long Press QR Code To
Identify The Concern

长按二维码识别关注

扫一扫，加入我们，分享更多知识

THANKS