

一、查看执行计划的方法有多种，依次如下：

1、打开熟悉的查看工具:PL/SQL Developer。


在 PL/SQL Developer 中写好一段 SQL 代码后，按 F5，PL/SQL Developer 会自动打开执行计划窗口，显示该 SQL 的执行计划。

2、explain plan for 命令

在 sql*plus 或者 PL/SQL Developer 打开的 command window 中，执行如下命令：

1)explain plan for select * from dual; +回车

2)select * from table(dbms_xplan.display); +回车

数据分析可视化工具完全跟踪用户访问行为的分析工具,持续优化转化率,提升产品粘度!

二、使用执行计划进行 SQL 调优

1、查看总 COST，获得资源耗费的总体印象

一般而言，执行计划第一行所对应的 COST(即成本耗费)值，反应了运行这段 SQL 的总体估计成本，单看这个总成本没有实际意义，但可以拿它与相同逻辑不同执行计划的 SQL 的总体 COST 进行比较，通常 COST 低的执行计划要好一些。

2、按照从左至右，从上至下的方法，了解执行计划的执行步骤

执行计划按照层次逐步缩进，从左至右看，缩进最多的那一步，最先执行，如果缩进量相同，则按照从上而下的方法判断执行顺序，可粗略认为上面的步骤优先执行。每一个执行步骤都有对应的 COST,可从单步 COST 的高低，以及单步的估计结果集(对应 ROWS/基数)，来分析表的访问方式，连接顺序以及连接方式是否合理。

3、 分析表的访问方式

表的访问方式主要是两种:全表扫描(TABLE ACCESS FULL)和索引扫描(INDEX SCAN)，如果表上存在选择性很好的索引，却走了全表扫描，而且是大表的全表扫描，就说明表的访问方式可能存在问题;若大表上没有合适的索引而走了全表扫描，就需要分析能否建立索引，或者是否能选择更合适的表连接方式和连接顺序以提高效率。

4、 分析表的连接方式和连接顺序

表的连接顺序:就是以哪张表作为驱动表来连接其他表的先后访问顺序。

表的连接方式:简单来讲，就是两个表获得满足条件的数据时的连接过程。主要有三种表连接方式，**嵌套循环(NESTED LOOPS)**、**哈希连接(HASH JOIN)**和**排序-合并连接(SORT MERGE JOIN)**。

我们常见得是嵌套循环和哈希连接。

嵌套循环:最适用也是最简单的连接方式。类似于用两层循环处理两个游标，外层游标称作驱动表，Oracle 检索驱动表的数据，一条一条的代入内层游标，查找满足 WHERE 条件的所有数据，因此**内层游标表中可用索引的选择性越好，嵌套循环连接的性能就越高。**

哈希连接:先将驱动表的数据按照条件字段以散列的方式放入内存，然后在内存中匹配满足条件的行。哈希连接需要有**合适的内存**，而且必须在**CBO 优化模式**下，**连接两表的 WHERE 条件有**

等号的情况下才可以使用。哈希连接在表的数据量较大，表中没有合适的索引可用时比嵌套循环的效率要高。

总结:

1、这里看到的执行计划，只是 SQL 运行前可能的执行方式，实际运行时可能因为软硬件环境的不同，而有所改变，而且 cost 高的执行计划，不一定在实际运行起来，速度就一定差，我们平时需要结合执行计划，和实际测试的运行时间，来确定一个执行计划的好坏。

2、对于表的连接顺序，多数情况下使用的是嵌套循环，尤其是在索引可用性好的情况下，使用嵌套循环式最好的，但当 ORACLE 发现需要访问的数据表较大，索引的成本较高或者没有合适的索引可用时，会考虑使用哈希连接，以提高效率。排序合并连接的性能最差，但在存在排序需求，或者存在非等值连接无法使用哈希连接的情况下，排序合并的效率，也可能比哈希连接或嵌套循环要好。

Oracle 中的执行计划 查看和使用

Oracle 中查看执行计划

Oracle 作者: stotf 时间: 2019-02-11 18:15:20 5452 0

方法一、通过使用工具 PLSQL Developer 中的 Explain Plan Window 窗口查看 SQL 执行计划。具体参考

<https://www.cnblogs.com/Dreamer-1/p/6076440.html>

方法二、通过 SQL*PLUS 中的 autotrace 命令查看

1. 登录拥有 dba 权限的用户，分别执行

脚本 \${ORACLE_HOME}/RDBMS/ADMIN/utlxplan.sql 和

脚本 \${ORACLE_HOME}/sqlplus/admin/plustrce.sql

然后通过 SQL*PLUS 就可以查看执行计划了

2. 查看执行计划有下面四种选项

1> set autotrace on -- (得到执行计划，并输出结果)

2> set autotrace traceonly -- (得到执行计划，但不输出结果)

3> set autotrace traceonly explain -- (得到执行计划，不输出统计信息和结果，仅展现执行计划部分)

4> set autotrace traceonly statistics -- (不输出执行计划和结果，仅展现统计信息)

3. 优缺点

优点：

1> 可以输出运行时的相关统计信息 (产生多少逻辑读，多少递归调用，多少次物理读的

情况)

2> 虽然必须要等语句执行完毕后才可执行计划，但是可以有 traceonly 开关来控制返回结果不打屏输出

缺点：

1> 必须等语句执行完毕后，才可以出结果

2> 无法看到表被访问了多少次

方法三、explain plan for 获取

1. 执行步骤如下：

SQL> set linesize 200;

SQL> set pagesize 2000;

SQL> explain plan for select * from emp;

Explained

```
SQL> select * from table(dbms_xplan.display());
```

PLAN_TABLE_OUTPUT

Plan hash value: 3956160932

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	532	3 (0)	00:00:01
1	TABLE ACCESS FULL	EMP	14	532	3 (0)	00:00:01

8 rows selected

2.优缺点

优点:

1> 无需真正执行，快捷方便

缺点:

1> 没有输出相关的统计信息（产生多少次逻辑读，多少次物理读，多少次递归调用）

2> 无法判断处理了多少行

3> 无法判断表被访问了多少次

方法四、statistics_level = all 或者 /*+ gather_plan_statistics */

1.执行步骤

1> 通过 statistics_level = all

```
SQL> set linesize 500;
```

```
SQL> set pagesize 1000;
```

```
SQL> alter session set statistics_level = all;
```

会话已更改。

```
SQL> select count(*) from emp;
```

```
COUNT(*)
```

```
-----
```

```
14
```

```
SQL> select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

```
PLAN_TABLE_OUTPUT
```

```
-----
```

```
SQL_ID g59vz2u4cu404, child number 0
```

```
-----
```

```
select count(*) from emp
```

```
Plan hash value: 2937609675
```

```
-----
```

```
| Id | Operation          | Name | Starts | E-Rows | A-Rows | A-Time   | Buffers |
```

```
-----
```

```
| 0 | SELECT STATEMENT |      |      1 |      1 |      1 | 00:00:00.01 |      1 |
```

```
| 1 |  SORT AGGREGATE  |      |      1 |      1 |      1 | 00:00:00.01 |      1 |
```

```
| 2 |  INDEX FULL SCAN| PK_EMP |      1 |     14 |     14 | 00:00:00.01 |      1 |
```

```
-----
```

```
已选择 14 行。
```

```
2> 通过 /*+ gather_plan_statistics*/
```

```
SQL> set linesize 200;
```

```
SQL> set pagesize 500;
```

```
SQL> select /*+gather_plan_statistics*/ count(*) from emp;
```

```
COUNT(*)
```

```
-----
```

```
SQL> select * from table(dbms_xplan.display_cursor(null,null,'allstats last'));
```

```
PLAN_TABLE_OUTPUT
```

```
-----  
SQL_ID 537ffv2mq5375, child number 0
```

```
-----  
select /*+gather_plan_statistics*/ count(*) from emp
```

```
Plan hash value: 2937609675
```

```
-----  
| Id | Operation          | Name      | Starts | E-Rows | A-Rows |   A-Time   | Buffers |  
-----  
| 0 | SELECT STATEMENT    |           |       1 |       1 |       1 | 00:00:00.01 |       1 |  
| 1 |  SORT AGGREGATE      |           |       1 |       1 |       1 | 00:00:00.01 |       1 |  
| 2 |   INDEX FULL SCAN| PK_EMP    |       1 |      14 |      14 | 00:00:00.01 |       1 |  
-----
```

已选择 14 行。

2.关键字解读

1> Starts: 该 SQL 执行的次数

2> E-Rows: 执行计划预计的行数。

3> A-Rows: 实际返回的行数。**A-Rows 跟 E-Rows 做比较**, 就可以确定哪一步执行计划出了问题。

4> A-Time: 每一步实际执行的时间 (HH : MM :SS.FF), 根据这一行可以知道该 **SQL 耗时** 在了哪个地方

5> Buffers: 每一步执行的逻辑读或一致性读

3.优缺点

优点

1> 可以清晰的从 **Starts** 得出表被访问多少

2> 可以清晰的从 **E-Rows** 和 **A-Rows** 中得到预测的行数和真实的行数,从而可以准确判断 **Oracle** 评估是否正确

3> 虽然没有专门的输出运行时的相关统计信息,但是执行计划中的 **buffers** 就是真实的逻辑读的多少

缺点

1> 必须等语句真正执行完毕后,才可以得出结果

2> 无法控制记录输屏打出,不像 **autotrace** 由 **traceonly** 可以不将结果打屏输出

3> 看不出递归调用次数,看不出物理读的多少(不过逻辑读才是重点)

方法五、通过 **dbms_xplan.display_cursor(&sql_id)**输入 **sql_id** 参数获取

1.执行步骤

```
SQL> select sql_id from sys.v_$sql t where t.sql_text like '%select count(*) from emp';
```

```
SQL_ID
```

```
-----
```

```
g59vz2u4cu404
```

```
SQL> select * from table(dbms_xplan.display_cursor('g59vz2u4cu404'));
```

```
PLAN_TABLE_OUTPUT
```

```
-----
```

```
SQL_ID  g59vz2u4cu404, child number 0
```

```
-----
```

```
select count(*) from emp
```

```
Plan hash value: 2937609675
```

```
-----
```

```
| Id | Operation          | Name | Rows | Cost (%CPU)| Time     |
```



```

-----
| 0 | SELECT STATEMENT |          |          | 1 (100)|          |
| 1 |  SORT AGGREGATE   |          | 1 |          |          |
| 2 |  INDEX FULL SCAN| PK_EMP | 14 | 1 (0)| 00:00:01 |
-----

```

已选择 14 行。

2.优缺点

优点:

1> 知道 `sql_id` 立即可得到执行计划, 和 `explain plan for` 一样无需执行

2> 可以得到真实的执行计划

缺点:

1> 没有输出运行的相关统计信息 (产生的物理读, 逻辑读, 递归调用次数)

2> 无法判断处理了多少行

3> 无法判断表被访问了多少次

方法六、10046 trace 跟踪

1.执行步骤

Step1: `alter session setevents '10046 trace name context forever,level 12';` (开启跟踪)

Step2: 执行 sql

Step3: `alter session setevents '10046 trace name context off';` (关闭跟踪)

Step4: 步骤 4: 找到跟踪后产生的文件

Step5: `tkprof trc 文件 目标文件 sys=no sort=prsela,exeela,fchela` (格式化命令)

```
SQL> set autot off
```

```
SQL> alter session set statistics_level=typical;
```

```
Session altered.
```

```
SQL> alter session set events '10046 trace name context forever,level 12';
```

Session altered.

```
SQL> select count(*) from test;
```

```
COUNT(*)
```

```
-----
```

```
7
```

```
SQL> alter session set events '10046 trace name context off';
```

Session altered.

```
SQL> select d.value
```

```
2 || '/'
```

```
3 || LOWER (RTRIM(i.INSTANCE, CHR(0)))
```

```
4 || '_ora_'
```

```
5 || p.spid
```

```
6 || '.trc' trace_file_name
```

```
7 from (select p.spid
```

```
8      from v$mystat m,v$session s, v$process p
```

```
9      where m.statistic#=1 and s.sid=m.sid and p.addr=s.paddr) p,
```

```

10      (select t.INSTANCE
11
12      FROM v$thread t,v$parameter v
13
14      WHERE v.name='thread'
15
16      AND(v.VALUE=0 OR t.thread#=to_number(v.value))) i,
17
18      (select value
19
20      from v$parameter
21
22      where name='user_dump_dest') d;

```

TRACE_FILE_NAME

/u01/app/oracle/diag/rdbms/ora12c/ora12c/trace/ora12c_oracle_12195.trc

SQL> host

[oracle@ora12c ~]\$ tkprof

/u01/app/oracle/diag/rdbms/ora12c/ora12c/trace/ora12c_oracle_12195.trc /home/oracle/10046.txt

sys=no sort=prsela,exeela,fchela

TKPROF: Release 12.1.0.1.0 - Development on Fri Jan 20 08:22:25 2017

Copyright (c) 1982, 2013, Oracle and/or its affiliates. All rights reserved.

SQL ID: 7b2twsn8vgfsc Plan Hash: 784602781

select count(*) from test

call	count	cpu	elapsed	disk	query	current	rows
------	-------	-----	---------	------	-------	---------	------

Parse	1	0.00	0.00	3	3	2	0
-------	---	------	------	---	---	---	---

Execute	1	0.00	0.00	0	0	0	0
---------	---	------	------	---	---	---	---

Fetch	2	0.00	0.00	4	66	0	1
-------	---	------	------	---	----	---	---

total	4	0.00	0.00	7	69	2	1
-------	---	------	------	---	----	---	---

Misses in library cache during parse: 1

Optimizer mode: ALL_ROWS

Parsing user id: 103

Number of plan statistics captured: 1

Rows (1st) Rows (avg) Rows (max) Row Source Operation

1 1 1 SORT AGGREGATE (cr=66 pr=4 pw=0 time=298 us)

7 7 7 PARTITION RANGE ALL PARTITION: 1 3 (cr=66 pr=4
pw=0 time=397 us cost=39 size=0 card=11)

7 7 7 TABLE ACCESS FULL TEST PARTITION: 1 3 (cr=66
pr=4 pw=0 time=290 us cost=39 size=0 card=11)

Elapsed times include waiting on following events:

Event waited on	Times	Max. Wait	Total Waited
-----------------	-------	-----------	--------------

-----	Waited	-----	-----
-------	--------	-------	-------

db file sequential read	4	0.00	0.00
-------------------------	---	------	------

SQL*Net message to client	2	0.00	0.00
---------------------------	---	------	------

db file scattered read	1	0.00	0.00
------------------------	---	------	------

2.优缺点

优点:

- 1> 可以看出 SQL 语句对应的等待事件
- 2> 如果 SQL 语句中有函数, SQL 中有 SQL, 将会都被列出, 无处遁形
- 3> 可以方便的看出处理的行数, 逻辑物理读
- 4> 可以跟踪整个程序包

缺点:

- 1> 步骤繁琐, 比较麻烦
- 2> 无法判断表被访问了多少次
- 3> 执行计划中的谓词部分不能清晰的展现出来

方法七、awrsqldrpt.sql

1.执行步骤

Step1: @?/rdbms/admin/awrsqldrpt.sql

Step2: 选择你要的断点 (begin snap 和 end snap)

Step3: 输入 sql_id

适用场合分析

1.如果某 SQL 执行非常长时间才会出结果, 甚至慢到返回不了结果, 这时候看执行计划就只能用方法 explain plan for;

2.跟踪某条 SQL 最简单的方法是方法 explain plan for , 其次就是方法 autotrace;

3.如果想观察到某条 SQL 有多条执行计划的情况, 只能用方法 dbms_xplan.display_cursor(sql_id) 和方法 awrsqldrpt.sql;

4.如果 SQL 中含有多函数, 函数中套有 SQL 等多层递归调用, 想准确分析, 只能使用方法 10046 trace;

5.要想确保看到真实的执行计划, 不能用方法 plsql developer 和方法 explain plan for;

6.要想获取表被访问的次数, 只能使用方法 statistics_level (/*+ gather_plan_statistics */);

