Oracle 中查看执行计划的方法

1. explain plan
2. dbms_xplan
3. SQLPLUS中的 AUTOTRACE开关
4. 10046 事件
5. 10053 事件
6. AWR 报告或 Statspack 报告
7. 一些现成的脚本

1. explain plan

Note: PL/SQL Develper 中的快捷键 F5 只是在 explain plan 命令上的一层封装。

语法 ：

explain plan for + 目标SQL

select * from table(dbms_xplan.display)

eg:

explain plan for select empno, ename, dname from scott.emp, scott.dept where emp.deptno=dept.deptno;

select * from table(dbms_xplan.display);

note: 10g 及以上版本，执行 explain plan 命令，则 Oracle 就将解析目标SQL所产生的执行计划的具体步骤写入 PLAN_TABLE$, select * from table(dbms_xplan.display) 从 PLAN_TABLE$中将这些具体执行步骤一格式化的方式显示出来。

PLAN_TABLE$ 是一个 ON COMMIT PRESERVE ROWS 的 GLOBAL_TEMPORARY TABLE.

eg: 查看 PLAN_TABLE$ 的 DDL

select dbms_metadata.get_ddl('TABLE','PLAN_TABLE$','SYS') from dual;

2. dbms_xplan

语法:

1) select * from table(dbms_xplan.display);

Note: 此方法需要与 explain plan 命令配合使用。

2) select * from table(dbms_xplan.display_cursor(null,null,'advanced'));

or

select * from table(dbms_xplan.display_cursor(null,null,'all'));

Note: 此方法用于在SQLPLUS 中查看刚刚执行过的SQL 的执行计划。advanced 比 all 显示的信息更详细。

3) select * from table(dbms_xplan.display_cursor('sql_id/hash_value',child_cursor_number,'advanced'));

Note: 此方法用于查看指定SQL 的执行计划。第一个参数是指定 SQL 的 SQL ID 或者 SQL HASH VALUE, 第二个参数是要查看的执行计划所在的 Child Cursor Number。

eg: 只要目标SQL 所对应的 Child Cursor 还在 Library Cache中，就可以从 V$SQL 中查到目标 SQL 的 Child Cursor 的详细信息。

select sql_text, sql_id, hash_value, child_number from v$sql where sql_text like 'select empno, ename%';

eg: 只要目标 SQL 的执行计划所在的 Child Cursor 还没有被 age out 出 Shared Pool, 就可用法3来查看该SQL 的执行计划。

select * from table(dbms_xplan.display_cursor('3yfu3wh150aqt',0,'advanced'));

4) select * from table(dbms_xplan.display_awr('sql_id'));

Note: 此方法用于查看指定SQL的所有历史执行计划。如果该 SQL 的执行计划已经被 age out 出 Shared Pool, 那么只要改 SQL 的执行计划被 Oracle 采集到 AWR Repository 中，就可用此法查看该 SQL 的所有历史执行计划。

eg: 可以查到结果

select sql_text, sql_id, version_count, executions from v$sqlarea where sql_text like 'select count(*) from t1%';

eg: 手工采集一下 AWR 报告，采集完后清空 Shared Pool

exec dbms_workload_repository.create_snapshot();

alter system flush shared_pool;

eg: 查询不到结果

select sql_text, sql_id, version_count, executions from v$sqlarea where sql_text like 'select count(*) from t1%';

eg: 用之前的方法（是查不到的）

select * from table(dbms_xplan.display_cursor('79g1p9197x4u',0,'advanced'));

select * from table(dbms_xplan.display_cursor('79g1p9197x4u',1,'advanced'));

eg: 用此法（可以查到）

select * from table(dbms_xplan.display_awr('79g1p9197x4u'));

Note: dbms_xplan.display_awr 相对于 dbms_xplan.display_cursor 的不足是 显示的执行计划中看不到执行步骤对应的为此条件。

3. SQLPLUS中的 AUTOTRACE开关

语法：

set autotrace {off | on | traceonly} [explain] [statistics]

note:

set autotrace on : 显示SQL 执行结果的具体内容，显示SQL 执行计划和资源消耗情况。

set autotrace off : 默认off，只显示SQL 执行结果

set autotrace traceonly : 不显示SQL 执行结果的具体内容（只显示执行结果的数量），显示SQL 执行计划和资源消耗情况。

set autotrace traceonly explain : 只显示SQL 执行计划，不显示SQL 的执行结果和资源消耗量

set autotrace traceonly statistics : 只显示SQL 的资源消耗量和执行结果的数量，不显示SQL 的执行计划。

autotrace = autot

traceonly = trace

explain = exp

statistics = stat

eg:

set autotrace on

select statement;

set autotrace traceonly

select statement;

set autotrace traceonly explain

select statement;

select atutotrace traceonly statistics

select statement;

set autotrace off

select statement;

4. 10046 事件

Note: 此方法所得到的执行计划中明确显示了目标SQL实际执行计划中每一个执行步骤所消耗的逻辑读、物理读和化肥的时间。这种细粒度的明细显示在诊断复杂SQL 的性能问题是尤为有用，也是其他三种方法所不能提供的。（gather_plan_statistics hint 配合 dbms_xplan包一起使用也可达到类似10046事件的之中细粒度明细显示效果。）

步骤：

1） 在当前 Session 中激活 10046 事件；

2） 在此 Session 中执行目标 SQL;

3） 在此 Session 中关闭 10046 事件。

eg: 在当前 Session 中激活 10046 事件

alter session set events '10046 trace name context forever, level 12'

or

oradebug event 10046 trace name context forever, level 12

Note: 除了 level 值，其他部分都不可修改。level 值通常为12，标识产生的 trace 文件中除了有目标 SQL 的执行计划和资源消耗明细，还会包含目标SQL 所使用的绑定变量的值以及该 Session 所经历的等等事件。

eg: 得到当前 Session 所对应的 trace 文件的具体路径和名称

oradebug tracefile_name

eg: 在当前 Session 中关闭 10046 事件

alter session set events '10046 trace name context off'

or

oradebug event 10046 trace name context off

Note: tkprof 命令 可将 10046 事件所产生的 裸trace 文件翻译得更直观容易读懂。

eg:

eg: 准备对当前 Session 使用 oradebug 命令

oradebug setmypid

eg:

oradebug event 10046 trace name context forever, level12

eg:

select statement;

eg: 查看对应的trace 文件的路径和名称

oradebug tracefile_name

eg: 关闭当前 Session 的10046事件

oradebug event 10046 trace name context off

eg: 用 tkprof 命令翻译一下裸trace 文件

tkprof c:xx\xxx.trc e:yy\yyy.trc

5. 10053 事件

6. AWR 报告或 Statspack 报告

@$ORACLE_HOME/rdbms/admin/awrsqrpt.sql : 获取 AWR SQL 报告（10g 及以上版本）

@$ORACLE_HOME/rdbms_admin/sprepsql : 获取 Statspack SQL 报告 （9i 及以上）

7. 一些现成的脚本

如display_cursor_9i.sql 和存储过程 printsql (来源：www.dbsnake.net/books)

Note: 使用 display_cursor_9i.sql 脚本的前提条件是目标 SQL 的执行计划还在 Shared Pool 中。

eg: desc dbms_xplan; (in 9i)

可以看到 9i 的 dbms_xplan 包里是没有 display_cursor方法的，so 需要该脚本

eg: 该脚本使用样例

@'E:\display_cursor_9i.sql' 1771670138 0

存储过程 printsql 是在脚本 display_cursor_9i.sql 上的封装。

eg: printsql使用样例

exec printsql(1212576,'SPID');

Note: 1212576 为oracle 的一个 进程的 spid (通过在服务器上执行 topas命令得到)