收获，不止 SQL 优化

第 四 章

左右 Oracle 执行计划之妙法

E-Mail:45240040@qq.com

目录

1.说说你对 hint 的认识

1.1 hint 可分类总结

Hint 是 oracle 提供的一种 SQL 语法，通过在 SQL 语句中设置 hint 从而影响 SQL 的执行计划，通过 v$sql_hint(11G)视图查询所有的 hint，他们可分为一下几类：

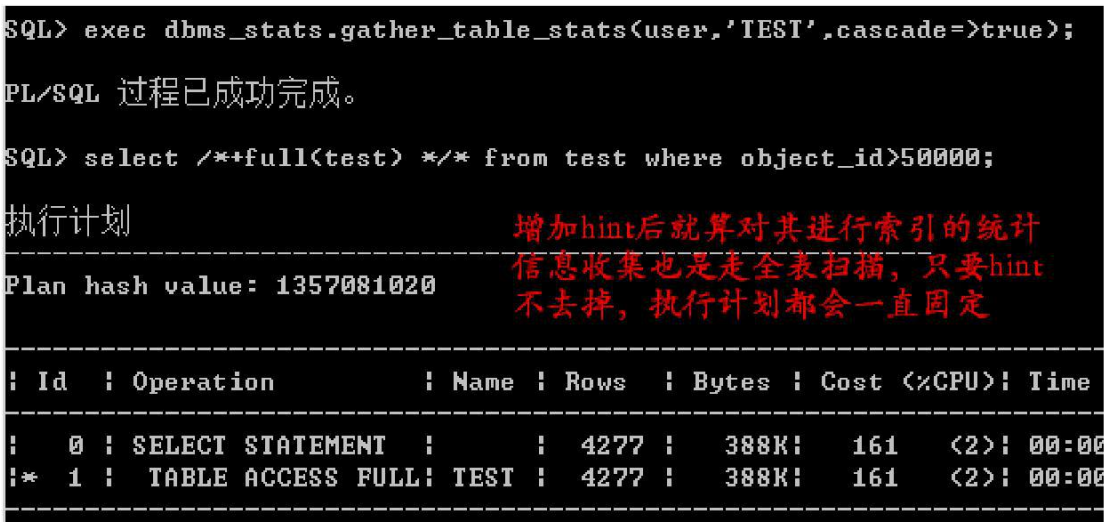
- 初始化参数 hint 如 all_rows, first_rows ,rule 等
- 查询转化 hint 如 outer_join_inner, merge 等
- 访问路径 hint 如 full,hash,index 等
- 连接提示 hint 如 leading, use_nl 等
- 并行处理 hint 如 parallel, parallel_index 等
- 其他 hint 如 append, cache,driving_site 等

1.2 hint 可改变执行计划

通过 hint 可以改变 SQL 的执行计划，来

```
SQL> create table test as select * from dba_objects;

表已创建。

SQL> create index idx_test on test(object_id);

索引已创建。

SQL> set autot traceonly exp
SQL> select * from test where object_id>2000;

执行计划
----------------------------------------------------------
Plan hash value: 1357081020

--------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      | 56214 |  9716K|   163   (4)| 00:00:02 |
|*  1 |  TABLE ACCESS FULL| TEST | 56214 |  9716K|   163   (4)| 00:00:02 |
--------------------------------------------------------------------------------
```

1.3 hint 可固定执行计划



1.4 hint 有作用范围

对于简单的 SQL 语句一般只有一个查询块，那么在其上设置 hint 其作用范围就是该语句块，而对于复杂的有多个查询语句的 SQL 语句，例如查询中用到了子

查询、内联视图、集合等操作时，各个 hint 的作用域是不同的。

```
SQL> conn livan/livan
已连接。
SQL> create table emp as select * from scott.emp;

表已创建。

SQL> create index idx_emp_deptno on emp(deptno);

索引已创建。

SQL> create index idx_emp_empno on emp(empno);

索引已创建。

SQL> create table dept as select * from scott.dept;

表已创建。

SQL> create index idx_dept_deptno on dept(deptno);

索引已创建。
```

```
SQL> set lines 1000
SQL> set autot traceonly exp
SQL> with emps as (select deptno,count(*) as cnt  from emp
  2                 where empno in (7369,7782,7499)group by deptno)
  3  select dept.dname,emps.cnt
  4   from dept,emps
  5   where dept.deptno=emps.deptno;                    正常情况的执行计划

执行计划
----------------------------------------------------------
Plan hash value: 1813372126

-------------------------------------------------------------------------------------------
| Id  | Operation                      | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                 |     3 |   144 |     4  (25)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID   | DEPT            |     1 |    22 |     1   (0)| 00:00:01 |
|   2 |   NESTED LOOPS                 |                 |     3 |   144 |     4  (25)| 00:00:01 |
|   3 |    VIEW                        |                 |     3 |    78 |     3  (34)| 00:00:01 |
|   4 |     HASH GROUP BY              |                 |     3 |    78 |     3  (34)| 00:00:01 |
|   5 |      INLIST ITERATOR           |                 |       |       |            |          |
|   6 |       TABLE ACCESS BY INDEX ROWID| EMP           |     3 |    78 |     2   (0)| 00:00:01 |
|*  7 |        INDEX RANGE SCAN        | IDX_EMP_EMPNO   |     1 |       |     1   (0)| 00:00:01 |
|*  8 |    INDEX RANGE SCAN            | IDX_DEPT_DEPTNO |     1 |       |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

```
SQL> with emps as (select /*+full(emp)*/ deptno,count(*) as cnt
  2                 from emp where empno in (7369,7782,7499)
  3                 group by deptno)
  4  select /*+full(dept)*/ dept.dname,emps.cnt
  5   from dept,emps
  6   where dept.deptno=emps.deptno;

执行计划                                              可以看到两个hint有效区域都被控制在他们的语句块中，hint才起到了效果
----------------------------------------------------------
Plan hash value: 2415981340

-------------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |     3 |   144 |     8  (25)| 00:00:01 |
|*  1 |  HASH JOIN           |      |     3 |   144 |     8  (25)| 00:00:01 |
|   2 |   VIEW               |      |     3 |    78 |     4  (25)| 00:00:01 |
|   3 |    HASH GROUP BY     |      |     3 |    78 |     4  (25)| 00:00:01 |
|*  4 |     TABLE ACCESS FULL| EMP  |     3 |    78 |     3   (0)| 00:00:01 |
|   5 |   TABLE ACCESS FULL  | DEPT |     4 |    88 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

1.5 hint 也有无法生效的时候

Hint 无法生效的原因大概有一下几种：

- 算法不能够支持
- 组合 hint 有矛盾
- 依据 hint 结果错
- 书写出现了错误

现在示例一个常见的场景：

```
SQL> create table test2(name varchar(30),addre varchar(30));

表已创建。

SQL> declare
  2  begin
  3  for i in 1..1000 loop
  4  insert into test2 values('张三'||i,'北京'||i);
  5  end loop;
  6  end;
  7  /

PL/SQL 过程已成功完成。

SQL> update test2 set name='' where rownum=1;

已更新 1 行。

SQL> commit;

提交完成。

SQL> create index idx_test2 on test2(name);

索引已创建。
```

```
SQL> set autot traceonly exp
SQL> select /*+ index(test2) */count(*) from test2;
                          虽然指定走索引，因为索引中共有空值导致count错误，
执行计划                   所以hint没有起作用
----------------------------------------------------------
Plan hash value: 634289536

-------------------------------------------------------------------------
| Id  | Operation          | Name  | Rows  | Cost (%CPU)| Time     |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |       |     1 |     3   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE    |       |     1 |            |          |
|   2 |   TABLE ACCESS FULL| TEST2 |  1000 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement

SQL> select /*+ index(test2) */count(*) from test2 where name is not null;
                                  通过屏蔽掉空值，系统开始走索引
执行计划
----------------------------------------------------------
Plan hash value: 2968702229

-----------------------------------------------------------------------------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |     1 |    17 |     8   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE   |           |     1 |    17 |            |          |
|*  2 |   INDEX FULL SCAN | IDX_TEST2 |   999 | 16983 |     8   (0)| 00:00:01 |
-----------------------------------------------------------------------------
```

2.说说你对 rownum 实体化视图优化方法表连接的认识

2.1 案例演示

```
SQL> create table t1
  2  ( prc_chk_key number(9) not null,
  3    prod_key number(12) not null,
  4    cmpt_loc_key number(5) not null,
  5    loc_key number(5) not null,
  6    prc_chk_dt date
  7  )
  8  /

表已创建。

SQL> create table t2
  2  ( prc_chk_key number(9) not null,
  3    prc_chk_typ_desc varchar2(35) not null,
  4    cmpt_loc_key number(5),
  5    loc_key number(5) not null
  6    )
  7  /

表已创建。
```

```
SQL> insert into t1 select 2, 3, 4, 5, sysdate
  2  from all_objects where ROWNUM <= 500;

已创建500行。

SQL> insert into t2 select 2, 'x', 4, 5
  2  from all_objects where ROWNUM <= 500;

已创建500行。

SQL> commit;

提交完成。

SQL> CREATE OR REPLACE function F
  2  (v_prod_key IN number default NULL,
  3   v_prc_chk_key IN number default NULL,
  4   v_return IN varchar2 default NULL,
  5   v_want_sr IN varchar2 default NULL,
  6   v_version IN number ) RETURN varchar2
  7   as
  8   begin
  9   dbms_application_info.set_client_info
 10   (userenv('client_info')+1);
 11   return 'x';
 12   end;
 13  /

函数已创建。
```

```
SQL> set timing on
SQL> exec dbms_application_info.set_client_info(0);

PL/SQL 过程已成功完成。

已用时间:  00: 00: 00.00
SQL> set autotrace traceonly
SQL> select a12.prc_chk_typ_desc prc_chk_typ_desc,
  2         a11.prc_chk_dt prc_chk_dt,
  3         a11.cmpt_loc_key cmpt_loc_key,
  4         a11.prod_key upc_prod_key,
  5         a11.loc_key loc_key,
  6         max(f(a11.prod_key, a11.prc_chk_key, 'QTY', 'D', 1)),
  7         max(f(a11.prod_key, a11.prc_chk_key, 'AMT', 'D', 1)),
  8         max(f(a11.prod_key, a11.prc_chk_key, 'CODE', 'D', 1)),
  9         max(f(a11.prod_key, a11.prc_chk_key, 'PRC', 'D', 1))
 10    from t1 a11, t2 a12
 11   where a11.cmpt_loc_key = a12.cmpt_loc_key
 12     and a11.loc_key = a12.loc_key
 13     and a11.prc_chk_key = a12.prc_chk_key
 14   group by a12.prc_chk_typ_desc,
 15            a11.prc_chk_dt,
 16            a11.cmpt_loc_key,
 17            a11.prod_key,
 18            a11.loc_key;

已用时间:  00: 00: 10.04
```

```
执行计划
----------------------------------------------------------
Plan hash value: 2197463864

-------------------------------------------------------------------------------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |      | 38073 |  4424K|    11  (64)| 00:00:01 |
|   1 |  SORT GROUP BY      |      | 38073 |  4424K|    11  (64)| 00:00:01 |
|*  2 |   HASH JOIN         |      | 38073 |  4424K|     5  (20)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| T1   |   500 | 30500 |     2   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| T2   |   500 | 29000 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("A11"."CMPT_LOC_KEY"="A12"."CMPT_LOC_KEY" AND
              "A11"."LOC_KEY"="A12"."LOC_KEY" AND
              "A11"."PRC_CHK_KEY"="A12"."PRC_CHK_KEY")

Note
-----
   - dynamic sampling used for this statement

SQL> set autotrace off
SQL> select userenv('client_info' ) data from dual;
                          执行时间达到了10秒多，递归调用达到了100万次
DATA
--------------------------------------------------------------------------------
1000000
```

```
SQL> set timing on
SQL> exec dbms_application_info.set_client_info(0);

PL/SQL 过程已成功完成。

已用时间:  00: 00: 00.00
SQL> set autotrace traceonly exp
SQL> select
  2    a12.prc_chk_typ_desc prc_chk_typ_desc,
  3    a11.prc_chk_dt prc_chk_dt,
  4    a11.cmpt_loc_key cmpt_loc_key,
  5    a11.prod_key upc_prod_key,
  6    a11.loc_key loc_key,
  7    max(a),
  8    max(b),
  9    max(c),
 10    max(d)
 11     from (select a11.*,
 12                  F(a11.PROD_KEY, a11.PRC_CHK_KEY, 'QTY', 'D', 1) a,
 13                  F(a11.PROD_KEY, a11.PRC_CHK_KEY, 'AMT', 'D', 1) b,
 14                  F(a11.PROD_KEY, a11.PRC_CHK_KEY, 'CODE', 'D', 1) c,
 15                  F(a11.PROD_KEY, a11.PRC_CHK_KEY, 'PRC', 'D', 1) d,
 16                  ROWNUM r
 17             from t1 a11) a11,
 18          T2 a12
 19    where a11.cmpt_loc_key = a12.cmpt_loc_key
 20      and a11.loc_key = a12.loc_key
 21      and a11.prc_chk_key = a12.prc_chk_key
 22    group by a12.prc_chk_typ_desc,
 23             a11.prc_chk_dt,
 24             a11.cmpt_loc_key,
 25             a11.prod_key,
 26             a11.loc_key;
已用时间:  00: 00: 00.16
```

```
执行计划
----------------------------------------------------------
Plan hash value: 2566946263

-------------------------------------------------------------------------------------
| Id  | Operation              | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |      |  1530 |    11M|     8  (25)| 00:00:01 |
|   1 |  SORT GROUP BY         |      |  1530 |    11M|     8  (25)| 00:00:01 |
|*  2 |   HASH JOIN            |      |  1530 |    11M|     7  (15)| 00:00:01 |
|   3 |    TABLE ACCESS FULL   | T2   |   500 | 29000 |     3   (0)| 00:00:01 |
|   4 |    VIEW                |      |   500 |  3939K|     3   (0)| 00:00:01 |
|   5 |     COUNT              |      |       |       |            |          |
|   6 |      TABLE ACCESS FULL | T1   |   500 | 30500 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("A11"."CMPT_LOC_KEY"="A12"."CMPT_LOC_KEY" AND
              "A11"."LOC_KEY"="A12"."LOC_KEY" AND
              "A11"."PRC_CHK_KEY"="A12"."PRC_CHK_KEY")

Note
-----
   - dynamic sampling used for this statement

SQL> set autotrace off
SQL> select userenv('client_info' ) data from dual;

DATA
--------------------------------------------------------------------------------
2000
```

```
执行计划
----------------------------------------------------------
Plan hash value: 2566946263

-------------------------------------------------------------------------------------
| Id  | Operation              | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |      |  1530 |    11M|     8  (25)| 00:00:01 |
|   1 |  SORT GROUP BY         |      |  1530 |    11M|     8  (25)| 00:00:01 |
|*  2 |   HASH JOIN            |      |  1530 |    11M|     7  (15)| 00:00:01 |
|   3 |    TABLE ACCESS FULL   | T2   |   500 | 29000 |     3   (0)| 00:00:01 |
|   4 |    VIEW                |      |   500 |  3939K|     3   (0)| 00:00:01 |
|   5 |     COUNT              |      |       |       |            |          |
|   6 |      TABLE ACCESS FULL | T1   |   500 | 30500 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("A11"."CMPT_LOC_KEY"="A12"."CMPT_LOC_KEY" AND
              "A11"."LOC_KEY"="A12"."LOC_KEY" AND
              "A11"."PRC_CHK_KEY"="A12"."PRC_CHK_KEY")

Note                          增加了rownum后执行时间不到1秒且递归调用明显降低
-----
   - dynamic sampling used for this statement

SQL> set autotrace off
SQL> select userenv('client_info' ) data from dual;

DATA
--------------------------------------------------------------------------------
2000
```

2.2 案例总结

通过上面的案例可以总结如下：

第一次没有加 rownum 执行时执行计划显示了对 T1 和 T2 表进行了全表扫描，并且先对 T1 和 T2 表扫描的结果进行 HASH 运算，然后调用的 F 函数，这样就会直接导致递归掉用的多达 100 万次【（500*500）*4】，造成 SQL 执行的时间严重消耗在 SQL 的递归调用上。

第二次在加上 rownum 执行时执行计划显示了对 T1 表是一个单独的操作，因为加上了 rownum oracle 为了保证结果集是对的，就必须对 T1 表进行单独运算，由此对 F 函数的对递归就变更了 2000 次【500*4】，由此整个 SQL 的执行时间也有原来的 10.04 秒降到 0.16 秒。

3.简要谈谈你对非 hint 方式影响执行计划的认识

3.1SQL 写法上的差异导致执行计划的不同

- 通过 with 语句改写 SQL
- Insert all 代替两个 insert
- 通过 rownum 分页
- 通过 rownum 改写实体化视图
- 直接通过 rowid 访问定位数据
- 缓存复杂 SQL 结果集
- 增加分区表分区列分区条件

3.2 利用数据库设计特性影响执行计划

- ✧ 对大表进行分区
- ✧ 对关联表建立簇表
- ✧ 使用索引组织表
- ✧ 为 SQL 建立物化视图
- ✧ 设置表的并行度
- ✧ 索引里空值对执行计划的影响
- ✧ 在外键上建立索引

3.3 使用表统计信息来影响执行计划

自从 10G 开始正式使用 CBO 时，统计信息有无、多少就会对 SQL 的执行计划的影响变得越来越重要，创建一张表及为该表创建索引，当表上的数据量较大需要收集表的统计信息，不同数量的统计信息对执行计划的影响显著，如对表收集10%的统计信息和对表收集 80%的统计信息，显然后者更能使 ORACLE 选择正确的执行计划，由此当然后者所需要耗费的资源也越多。当对所上的表建立索引时，索引建在选择性不高的列和选择性较高的列上同样出现天壤之别。

3.4 利用特殊手段预估索引效果

```
SQL> conn scott/tiger
已连接。
SQL> create table t as select * from dba_objects;

表已创建。

SQL> alter session set "_use_nosegment_indexes"=true;

会话已更改。

SQL> create index ix_t_id on t(object_id) nosegment;

索引已创建。

SQL> set linesize 1000
SQL> select index_name from user_indexes where table_name='T';

未选定行

SQL> select index_name from dba_indexes where index_name ='IX_T_ID';

未选定行
```

创建虚拟索引，在索引字典里查不到该索引

```
SQL> set lines 120
SQL> explain plan for select * from t where object_id=1;

已解释。

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

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 206018885

-------------------------------------------------------------------------------------
| Id  | Operation                   | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |         |     8 |  1416 |     5   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T       |     8 |  1416 |     5   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | IX_T_ID |   221 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

Predicate Information (identified by operation id):

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

   2 - access("OBJECT_ID"=1)

Note
-----
   - dynamic sampling used for this statement

已选择18行。
```

虽然查不到该索引，但在执行计划里可以看到该索引。
利用虚拟索引的技术可以预估你的SQL是否可以走索引扫描，然后再真正建立索引。

3.5 利用存储大纲改变或稳固执行计划

```
SQL> create table emp3 as select * from emp;

表已创建。

SQL> alter table emp3 add constraints pk_emp3 primary key(empno);

表已更改。

SQL> set autot trace exp
SQL> select * from emp3;

执行计划
----------------------------------------------------------
Plan hash value: 2425169977

--------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |    14 |  1218 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| EMP3 |    14 |  1218 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement

SQL> select /*+index(emp3) */* from emp3;

执行计划
----------------------------------------------------------
Plan hash value: 319922758

-----------------------------------------------------------------------------------------
| Id  | Operation                   | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |         |    14 |  1218 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMP3    |    14 |  1218 |     2   (0)| 00:00:01 |
|   2 |   INDEX FULL SCAN           | PK_EMP3 |    14 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```

创建emp3表，正常情况下执行计划走全表扫描，增加hint时使其走索引

```
SQL> set autot off
SQL> create or replace private outline pri_outline_emp3_1
  2  on select * from emp3;

大纲已创建。

SQL> create or replace private outline pri_outline_emp3_2
  2  on select /*+index(emp3)*/* from emp3;

大纲已创建。

SQL> select ol_name from  ol$hints;

OL_NAME
------------------------------------------------------------------
PRI_OUTLINE_EMP3_1
PRI_OUTLINE_EMP3_1
PRI_OUTLINE_EMP3_1
PRI_OUTLINE_EMP3_1
PRI_OUTLINE_EMP3_1
PRI_OUTLINE_EMP3_2
PRI_OUTLINE_EMP3_2
PRI_OUTLINE_EMP3_2
PRI_OUTLINE_EMP3_2
PRI_OUTLINE_EMP3_2

已选择10行。

SQL> update ol$hints set ol_name=decode(ol_name,'PRI_OUTLINE_EMP3_1',
  2  'PRI_OUTLINE_EMP3_2','PRI_OUTLINE_EMP3_2','PRI_OUTLINE_EMP3_1')
  3  where ol_name in ('PRI_OUTLINE_EMP3_1','PRI_OUTLINE_EMP3_2');

已更新10行。

SQL> commit;

提交完成。
```

创建两个私有的存储大纲并交换他们

```
SQL> alter session set use_private_outlines=true;

会话已更改。                                    刷新存储大纲

SQL> execute dbms_outln_edit.refresh_private_outline('PRI_OUTLINE_EMP3_1');

PL/SQL 过程已成功完成。

SQL> execute dbms_outln_edit.refresh_private_outline('PRI_OUTLINE_EMP3_2');

PL/SQL 过程已成功完成。

SQL> set autot traceonly exp
```

```
SQL> set autot traceonly exp
SQL> select * from emp3;
                              正常应该走全表扫描，现在走索引，并提示到了存储大纲
执行计划
----------------------------------------------------------
Plan hash value: 319922758

-------------------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |    14 |   518 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMP3    |    14 |   518 |     2   (0)| 00:00:01 |
|   2 |   INDEX FULL SCAN            | PK_EMP3 |    14 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

Note
-----
   - outline "PRI_OUTLINE_EMP3_1" used for this statement

SQL> select /*+index(emp3)*/* from emp3;

执行计划                      正常应该走索引，现在走全表扫描，并提示用到了存储大纲
----------------------------------------------------------
Plan hash value: 2425169977

--------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |    14 |   518 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| EMP3 |    14 |   518 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Note
-----
   - outline "PRI_OUTLINE_EMP3_2" used for this statement
```