第三章

第3章 访问和联结方法

3.1.1 如何选择全扫描操作

eg:

```
create table t1 as
select trunc((rownum-1)/100) id, rpad(rownum,100) t_pad
from dba_source
where rownum <= 10000;
create index t1_idx1 on t1(id);
exec dbms_stats.gather_table_stats(user,'t1',method_opt=>'FOR ALL COLUMNS SIZE 1',cascade=>TRUE);
create table t2 as
select mod(rownum,100) id,
rpad(rownum,100) t_pad
from dba_source
where rownum <10000;
create index t2_idx on t2(id);
exec dbms_stats.gather_table_stats(user,'t2',method_opt=>'FOR ALL COLUMNS SIZE 1',cascade=>TRUE);
```

eg: 对比两个表的执行计划

```
set autotrace on
select count(*) ct from t1 where id = 1;
select count(*) ct from t2 where id = 1;
```

Note: 对于表1，仅需访问很少的几个数据块；而对于表2，数据行在物理上是零散地分布的。

3.1.2

eg: 表T1和T2的数据行和数据块统计信息

```
select table_name, num_rows, blocks from user_tables where table_name = 'T2';
```

3.1.3

db_file_multiblock_read_count 参数制定一个单独的IO调用最多可以读几个块。

3.1.4

eg: list number of allocated blocks (table has 800000 rows). The highwater mark is the last block containing data.

```
select blocks from user_segments where segment_name = 'T2';
```

eg: list how many blocks contain data

```
select count(distinct (dbms_rowid.rowid_block_number(rowid))) block_ct from t2;
```

eg: list the lowest and highest block numbers for this table

```
select min(dbms_rowid.rowid_block_number(rowid)) min_blk,
max(dbms_rowid.rowid_block_number(rowid)) max_blk from t2;
```

eg: check the space usage in the table

get space_usage.sql

```
declare
    l_tabname varchar2(30) := '&1';
    l_fs1_bytes number;
    l_fs2_bytes number;
    l_fs3_bytes number;
    l_fs4_bytes number;
    l_fs1_blocks number;
    l_fs2_blocks number;
    l_fs3_blocks number;
    l_fs4_blocks number;
    l_full_bytes number;
    l_full_blocks number;
```

```
    l_unformatted_bytes number;
    l_unformatted_blocks number;
begin
    dbms_space.space_usage(
        segment_owner => user,
        segment_name => l_tabname,
        segment_type => 'TABLE',
        fs1_bytes => l_fs1_bytes,
        fs1_blocks => l_fs1_blocks,
        fs2_bytes => l_fs2_bytes,
        fs2_blocks => l_f2_blocks,
        fs3_bytes => l_fs3_bytes,
        fs3_blocks => l_fs3_blocks,
        fs4_bytes => l_fs4_bytes,
        fs4_blocks => l_fs4_blocks,
        full_bytes => l_full_bytes,
        full_blocks => l_full_blocks,
        unformatted_blocks => l_unformatted_blocks,
        unformatted_bytes => l_unformatted_bytes
    );
    dbms_output.put_line('0-25% Free = '||l_fs1_blocks||' Bytes = '||l_fs1_bytes);
    dbms_output.put_line('25-50% Free = '||l_fs2_blocks||' Bytes = '||l_fs2_bytes);
    dbms_output.put_line(50-70% Free = '||l_fs3_blocks||' Bytes = '||l_fs3_bytes);
    dbms_output.put_line('70-100% Free = '||l_fs4_blocks||' Bytes = '||l_fs4_bytes);
    dbms_output.put_line('Full Blocks = '||l_full_blocks||' Bytes = '||l_full_bytes);
end;
@space_usage T2
Note that most blocks are full, now delete the data.
delete from t2;
commit;
@space_usage T2
Note that blocks are now free but the same space is still consumed
eg: list number of blocks
select blocks from user_segments where segment_name='T2';
eg: list how many blocks contain data
select count(distinct (dbms_rowid.rowid_block_number(rowid))) block_ct from t2;
eg: execute a full table scan and note the consistent gets (logical block reads)
set autotrace traceonly
select * from t2;
set autotrace off
eg: truncate the table to deallocate the space and reset the HWM
truncate table t2;
eg: check the space usage after table is truncated
@space_usage T2
Note that the space has been deallocated
eg: list number of blocks
select blocks from user_segments where segment_name='T2';
set autotrace traceonlly
select * from t2;
set autotrace off

3.2
eg: 行编号解码
column filen format a50 head 'File Name'
select e.rowid, (select file_name from dba_data_files where file_id = dbms_rowid.rowid_to_absolute_fno(e.rowid, user, 'EMP'))
filen,
dbms_rowid.rowid_block_number(e.rowid) block_no,
dbms_rowid.rowid_row_number(e.rowid) row_no
```

```
from emp e where e.ename = 'KING';
```

3.2.2

eg: 索引聚簇因子

```
select t.table_name||'.'||i.index_name idx_name, i.clustering_factor, t.blocks, t.num_rows
from user_indexes i, user_tables t
where i.table_name = t.table_name and t.table_name in ('T1','T2')
order by t.table_name, i.index_name;
```

eg: 计算索引的局促因子

```
select t.table_name||'.'||i.index_name idx_name, i.clustering_factor, t.blocks, t.num_rows
from all_indexes i, all_tables t
where i.table_name = t.table_name
and t.table_name = 'EMPLOYEES'
and t.owner = 'HR'
and i.index_name = 'EMP_DEPARTMENT_ID'
order by t.table_name, i.index_name;
```

eg:

```
select department_id, last_name, blk_no,
lag(blk_no, 1, blk_no) over (order by department_id) prev_blk_no,
case when blk_no != lag(blk_no, 1, blk_no) over (order by department_id)
or rownum = 1
then '*** + 1'
else numm
end cluf_ct
from (
select department_id, last_name, dbms_rowid.rowid_block_num(rowid) blk_no
from hr.employees
where department_id is not null
order by department_id
);
```

3.2.3 索引唯一扫描 （INDEX UNIQUE SCAN）

eg:

```
set autotrace traceonly explain
select * from hr.employees where employee_id = 100;
```

3.2.4 索引范围扫描（INDEX UNIQUE SCAN）

eg:

```
set autotrace traceonly explain
select * from hr.employees where department_id = 60;
```

eg: 使用索引范围扫来避免排序

```
set autotrace traceonly explain
select * from hr.employees where department_id in (90, 100) order by department_id desc;
```

3.2.5 索引全扫描（INDEX FULL SCAN）

eg:

```
set autotrace traceonly explain
select email from hr.employees;
select first_name, last_name from hr.employees where first_name like 'A%';
select * from hr.employees order by employee_id;
select * from hr.employees order by employee_id desc;
```

eg: 索引全扫描求最小/最大值的最优办法（INDEX FULL SCAN (MIN/MAX)）

```
set autotrace traceonly explain
select min(department_id) from hr.employees;
select max(department_id) from hr.employees;
select min(department_id), max(department_id) from hr.employees;
select (select min(department_id) from hr.employees) min_id, (select max(department_id) from hr.employees) max_id from dual;
```

3.2.6 索引跳跃扫描（INDEX SKIP SCAN）

当谓语中包含谓语索引中非引导列上的条件，并且引导列的值是唯一的时候会选择索引跳跃扫描。

eg:

```
create index emp_jobfname_ix on employees(job_id, first_name, salary);
set autotrace traceonly
select * from employees where first_name = 'William';
select /*+ full(employees) */ * from employees where first_name = 'William';
```

3.2.7 索引快速全扫描（INDEX FAST FULL SCAN? ）

eg:

```
alter table hr.employees modify (email null);
set autotrace traceonly explain
select email from hr.employees;
set autotrace off
alter table hr.employees modify (email not null);
set autotrace traceonly explain
select email from hr.employees;
```

3.3 联结方法

嵌套循环联结

散列联结

排序-合并联结

笛卡尔联结

驱动表：访问的第一张表

内层表or被驱动表：访问的第二张表

3.3.1 嵌套循环联结（NESTED LOOPS）

使用一次访问运算所得的结果集中的每一行来与另一个表进行对碰。如果结果集的大小是有限的且在用来联结的列上有索引的话，这种联结的效率通常最高。

eg:

```
select empno, ename, dname, loc from emp, dept
where emp.deptno = dept.deptno
eg:
for each row in (select empno, ename, deptno from emp) loop
    for (select dname, loc from dept where deptno = outer.deptno) loop
        IF match then pass the row on to the next step
        If inner join and no match then discard the row
        If outer join and no match set inner column values to null
            and pass the row on to the next step
    end loop
end loop
```

eg: 嵌套循环联结顺序比较

```
set autotrace traceonly statistics
select empno, ename, dname, loc from scott.emp, scott.dept where emp.deptno  dept.deptno;
eg:
select/*+ ordered use_nl (dept emp) */ empno, ename, dname, loc
from scott.dept, scott.emp where emp.deptno = dept.deptno;
```

Note: 后面的例子使用了hint 来强制优化器选择将dept表作为驱动表的执行计划。

3.3.2 排序-合并联结（SORT JOIN）

独立地读取需要联结的两张表，对每张表中的数据行按照连接件进行排序，然后对排序后的数据行集进行合并。对这种联结方法来说排序的开销是非常大的。但合并的过程非常快。

eg:

```
select empno, ename, dname, loc from emp, dept where emp.deptno = dept.deptno
eg:
select empno, ename, deptno from emp order by deptno
select dname, loc, deptno from dept order by deptno
compare the rowsets and return rows where deptno in both lists match
```

for an outer join, compare the rowsets and return all rows from the first list
setting column values for the other table to null
eg:
select /*+ ordered */ empno, ename, dname, loc from scott.dept, scott.emp
where emp.deptno = dept.deptno;

3.3.3 散列联结（HASH JOIN）
与排序-合并联结类似，首先应用WHERE子句中的筛选标准来独立读取要进行联结的两个表。基于表和索引的统计信息，被确定为返回最少行数的表被完全散列化到内存中，这个散列表包含了原表的所有数据行并被基于将联结键转化为散列值的随机函数载入到散列桶中。下一步就是读取另一张较大的表并对联结键列应用散列函数。然后利用得到的散列值对较小的在内存中的散列表进行探测以寻找匹配的第一个表的行数据所在的散列桶。较大的包是驱动表，只读取一次，较小的表被多次读取。
eg:
select empno, ename, dname, loc from emp, dept where emp.deptno = dept.deptno
eg:
determine the smaller row set, or in the case of an outer join, use the outer joined table
select dname, loc, deptno from dept
hash the deptno column and build a hash table
select empno, ename, deptno from emp
hash the deptno column and probe the hash table
if match made, check bitmap to confirm row match
if no match made, discard the row
eg: 生成散列值
select distinct deptno, ora_hash(deptno, 1000) hv from scott.emp order by deptno;
eg:
select deptno from (select distinct deptno, ora_hash(deptno, 100) hv from scott.emp order by deptno) where hv between 100 and 500;
eg:
select distinct deptno, ora_hash(deptno,1000,50) hv from scott.emp order by deptno;
eg:
select deptno from (select dstinct deptno, ora_hash(deptno,1000,50) hv from scott.emp order by deptno) where hv between 100 and 500;
Note: 此处用ora_hash函数来说明散列值是如何生存的。ora_hash函数有三个参数：一个任何基本类型的输入，最大散列通知，一个中资质。如，ora_hash(10.1000) 将会返回一个0~1000之间的整数值。

3.3.4 笛卡尔联结（MERGE JOIN CARTESIAN）
笛卡尔联结发生在当一张表中的所有行与另一张表中的所有行联结的时候。
eg:
select empno, ename, dname, loc from emp, dept
eg:
determine the smaller table
select dname, loc from dept
select empno, ename from emp
for each row in dept match it to every row in emp retaining all rows

3.3.5 外联结 (NESTED LOOPS OUTER)
外联结返回一张表的所有行以及另一张联结表中满足联结条件的行数据。
（+）表示外联结
eg: query to show customers with total orders between $0 and $5000
select c.cust_last_name, nvl(sum(o.order_total),0) tot_orders
from custoemrs c, orders o
where c.customer_id = o.customer_id
group by c.cust_last_name
having nvl(sum(o.order_total),0) between 0 and 5000
order by c.cust_last_name;
eg: to produce just a count, modify the query slightly
select count(*) ct
from
(

```
select c.cust_last_name, nvl(sum(o.order_total),0) tot_orders
from customers c, orders o
where c.customer_id = o.customer_id
group by c.cust_last_name
having nvl(sum(o.order_total),0) between 0 and 5000
order by c.cust_last_name
);
```

eg: change the query to an outer join to include customers without orders

```
select count(*) ct
from
(
select c.cust_last_name, nvl(sum(o.order_total),0) tot_orders
from customers c, orders o
where c.customer_id = o.customer_id(+)
group by c.cust_last_name
having nvl(sum(o.order_total),0) between 0 and 5000
order by c.cust_last_name
);
set autotrace traceonly explain
/
```

eg: 使用ANSI联结语法的外联结

```
select count(*) ct
from
(
select c.cust_last_name, nvl(sum(o.order_total),0) tot_orders
from customers c
left outer join
orders o
on (c.customer_id = o.customer_id)
group by c.cust_last_name
having nvl(sum(o.order_total),0) between 0 and 5000
order by c.cust_last_name
);
```

eg: 使用ANSI联结语法的全外联结

```
create table e1 as select * from emp where deptno in (10,20);
create table e2 as select * from emp where deptno in (20,30);
select e1.ename, e1.deptno, e1.job, e2.ename, e2.deptno, e2.job
from e1 full outer join e2 on (e1.empno =  e2.empno);
set autotrace traceonly explain
/
```

eg: 全外联结功能的Oracle 等价语法

```
select e1.ename, e1.deptno, e1.job, e2.ename, e2.deptno, e2.job
from e1, e2 where e1.empno (+) = e2.empno
union
select e1.ename, e1.deptno, e1.job, e2.ename, e2.deptno, e2.job
from e1, e2 where e1.empno = e2.empno (+);
set autotrace traceonly explain
/
```