

# 数据库第十七周作业

19336035 陈梓乐

1. 使用 3.7.2 的方法，编码下列位图：
- 01100000010000000100  
→ (1, 0, 6, 7) → 0100110110110111
  - 100000001000001001010001  
→ (0, 7, 5, 2, 1, 3) → 001101111101011010011011
  - 0001000000000001000010000  
→ (3, 11, 4) → 101111101011110100
2. 构造一个表和有关查询，用执行计划证明建立普通的 B+树索引不能优化此查询，但建立位图索引则可以。提示：在 Oracle 中建立位图索引的命令是 `create bitmap index ...`

我们先在 emp.ename, emp.empno 建立位图索引，来执行以下命令：

```
select * from emp where ename = 'SCOTT' or empno = '12345';
```

下图将表明 Oracle 采用的是位图索引优化而非普通的 B+ 树优化：

终端 问题 输出 调试控制台

SQL = select \* from emp where ename = 'SCOTT' or empno = '12345';

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	LOC
7788	SCOTT	ANALYST	7566	19-4月 -87	3210		20	DALLAS

执行计划

Plan hash value: 3682317689

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	47	2 (0)	00:00:01
1	TABLE ACCESS BY INDEX ROWID	EMP	1	47	2 (0)	00:00:01
2	BITMAP CONVERSION TO ROWIDS					
3	BITMAP OR					
4	BITMAP CONVERSION FROM ROWIDS					
* 5	INDEX RANGE SCAN	NAME			1 (0)	00:00:01
6	BITMAP CONVERSION FROM ROWIDS					
* 7	INDEX RANGE SCAN	PK_EMP			0 (0)	00:00:01

3. 读完第 17 周课程幻灯片未讲完的部分，试举出 3 个自然连接的例子，分别使用 nestedloop、hash join、sort merge 来进行连接（展示其执行计划）。提示：可以使用 hints，在《SQL Reference》一书里有详细的 hints 列表和用途解释
- nestedloop

```
select /*+use_nl(dept, emp)*/
  ename,
  dname
from   emp
natural join dept;
```

```
SQL = select /*+use_nl(dept, emp)*/
2         ename,
3         dname
4   from   emp
5  natural join dept;
```

ENAME	DNAME
CLARK	ACCOUNTING
KING	ACCOUNTING
MILLER	ACCOUNTING
SMITH	RESEARCH
JONES	RESEARCH
SCOTT	RESEARCH
ADAMS	RESEARCH
FORD	RESEARCH
ALLEN	SALES
WARD	SALES
MARTIN	SALES
BLAKE	SALES
TURNER	SALES
JAMES	SALES

已选择14行。

执行计划

Plan hash value: 4192419542

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		6	222	10 (0)	00:00:01
1	NESTED LOOPS		6	222	10 (0)	00:00:01
2	TABLE ACCESS FULL	DEPT	4	80	3 (0)	00:00:01
* 3	TABLE ACCESS FULL	EMP	2	34	2 (0)	00:00:01

- hash join

```
select ename, dname from emp natural join dept;
```

```
SQL = select ename, dname from emp natural join dept;
```

ENAME	DNAME
SMITH	RESEARCH
ALLEN	SALES
WARD	SALES
JONES	RESEARCH
MARTIN	SALES
BLAKE	SALES
CLARK	ACCOUNTING
SCOTT	RESEARCH
KING	ACCOUNTING
TURNER	SALES
ADAMS	RESEARCH
JAMES	SALES
FORD	RESEARCH
MILLER	ACCOUNTING

已选择14行。

执行计划

Plan hash value: 615168685

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		6	222	7 (15)	00:00:01
* 1	HASH JOIN		6	222	7 (15)	00:00:01
2	TABLE ACCESS FULL	DEPT	4	80	3 (0)	00:00:01
3	TABLE ACCESS FULL	EMP	14	238	3 (0)	00:00:01

- sort merge

```
select /*+use_merge(dept, emp) */
       ename,
       dname
from   emp
natural join dept;
```

```
SQL = select /*+use_merge(dept, emp) */
2      emp.ename,
3      dept.dname
4  from emp
5  natural join dept;
```

ENAME	DNAME
BLAKE	SALES
TURNER	SALES
ALLEN	SALES
MARTIN	SALES
WARD	SALES
JAMES	SALES
SCOTT	RESEARCH
JONES	RESEARCH
SMITH	RESEARCH
ADAMS	RESEARCH
FORD	RESEARCH
KING	ACCOUNTING
MILLER	ACCOUNTING
CLARK	ACCOUNTING

已选择14行。

执行计划

Plan hash value: 1407029907

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		6	222	8 (25)	00:00:01
1	MERGE JOIN		6	222	8 (25)	00:00:01
2	SORT JOIN		4	80	4 (25)	00:00:01
3	TABLE ACCESS FULL	DEPT	4	80	3 (0)	00:00:01
* 4	SORT JOIN		14	238	4 (25)	00:00:01
5	TABLE ACCESS FULL	EMP	14	238	3 (0)	00:00:01

4. 关于连接次序的测试。构造一个 t1、t2、t3 的三表连接，通过设计数据使其中 t1 和 t2 的连接会产生较多的结果，t2 和 t3 的连接产生较少的结果，理论上先连接 t2 和 t3 是较为明智的执行计划，看下 Oracle 是否会自动选择这样的计划？改变三个表的书写次序，观察对执行计划是否有影响？我们的目标是希望让 Oracle 生成一个愚蠢的执行计划，然后用 hints 直接指定连接次序，人为强制Oracle按最有次序执行连接。对于安装了MySQL或PostgreSQL的同学，也可以在相应环境中做一下这个实验

我们准备了三张表，每张表都有两列，参数如下：

表名	行数	列名01	类型01	列名02	类型02
user_info	1500628	account	varchar2(10)	passwd	number(10)
account_admin	1	account	varchar2(10)	admin	number(2)
register_order	100001	account	varchar2(10)	ordered	number(7)

我们来把这三张表连接，对比以下三段代码：

```
select account, passwd, admin, ordered
from user_info
natural join register_order
natural join account_admin;

select account, passwd, admin, ordered
from user_info
natural join account_admin
natural join register_order;

select account, passwd, admin, ordered
from account_admin
natural join user_info
natural join register_order;
```

我们知道连接时最好以 `account_admin` 为中心连接，我们来看看各自性能分析：

```
SQL = select account, passwd, admin, ordered
2 from user_info
3 natural join account_admin
4 natural join register_order;
```

ACCOUNT	PASSWD	ADMIN	ORDERED
dprtQ4S70L	1680975342	1	0

执行计划

Plan hash value: 1983531489

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	58	111 (2)	00:00:02
1	NESTED LOOPS					
2	NESTED LOOPS		1	58	111 (2)	00:00:02
* 3	HASH JOIN		1	40	107 (2)	00:00:02
4	TABLE ACCESS FULL	ACCOUNT_ADMIN	1	20	3 (0)	00:00:01
5	TABLE ACCESS FULL	REGISTER_ORDER	83914	1638K	103 (1)	00:00:02
* 6	INDEX RANGE SCAN	ACCOUNT	1		2 (0)	00:00:01
7	TABLE ACCESS BY INDEX ROWID	USER_INFO	1	18	4 (0)	00:00:01

Predicate Information (identified by operation id):

```
3 - access("ACCOUNT_ADMIN"."ACCOUNT"="REGISTER_ORDER"."ACCOUNT")
6 - access("USER_INFO"."ACCOUNT"="ACCOUNT_ADMIN"."ACCOUNT")
```

```
SQL = select account, passwd, admin, ordered
2 from user_info
3 natural join register_order
4 natural join account_admin;
```

ACCOUNT	PASSWD	ADMIN	ORDERED
dprtQ4S70L	1680975342	1	0

执行计划

Plan hash value: 1983531489

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	58	111 (2)	00:00:02
1	NESTED LOOPS					
2	NESTED LOOPS		1	58	111 (2)	00:00:02
* 3	HASH JOIN		1	40	107 (2)	00:00:02
4	TABLE ACCESS FULL	ACCOUNT_ADMIN	1	20	3 (0)	00:00:01
5	TABLE ACCESS FULL	REGISTER_ORDER	83914	1638K	103 (1)	00:00:02
* 6	INDEX RANGE SCAN	ACCOUNT	1		2 (0)	00:00:01
7	TABLE ACCESS BY INDEX ROWID	USER_INFO	1	18	4 (0)	00:00:01

Predicate Information (identified by operation id):

```
3 - access("REGISTER_ORDER"."ACCOUNT"="ACCOUNT_ADMIN"."ACCOUNT")
6 - access("USER_INFO"."ACCOUNT"="REGISTER_ORDER"."ACCOUNT")
```

```
SQL = select account, passwd, admin, ordered
2 from account_admin
3 natural join user_info
4 natural join register_order;
```

ACCOUNT	PASSWD	ADMIN	ORDERED
dprtQ4S70L	1680975342	1	0

执行计划

Plan hash value: 1559136647

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	58	111 (2)	00:00:02
* 1	HASH JOIN		1	58	111 (2)	00:00:02
2	NESTED LOOPS					
3	NESTED LOOPS		1	38	7 (0)	00:00:01
4	TABLE ACCESS FULL	ACCOUNT_ADMIN	1	20	3 (0)	00:00:01
* 5	INDEX RANGE SCAN	ACCOUNT	1		2 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	USER_INFO	1	18	4 (0)	00:00:01
7	TABLE ACCESS FULL	REGISTER_ORDER	83914	1638K	103 (1)	00:00:02

Predicate Information (identified by operation id):

```
1 - access("USER_INFO"."ACCOUNT"="REGISTER_ORDER"."ACCOUNT")
5 - access("ACCOUNT_ADMIN"."ACCOUNT"="USER_INFO"."ACCOUNT")
```



可以看到，上图反映了如下事实：

- 更改书写顺序将改变表的连接顺序
- Oracle 并不会自动选择连接先后顺序，或只能在一定范围内智能选择。
- 最差的写法是第三种写法，它先决定连接最多行数的两个表，再来连接第三个表

这样，我们来修改第三种写法，加上hints，使得它先连接记录较少的表：

```
select /*+ordered use_hash(account_admin, register_order)*/
      account, passwd, admin, ordered
from   account_admin
natural join user_info
natural join register_order;
```

下图将说明我们的修改有助于 Oracle 先连接 account\_admin 和 register\_order。

SQL = select /\*+ordered use\_hash(account\_admin, register\_order)\*/  
2 account, passwd, admin, ordered  
3 from account\_admin  
4 natural join user\_info  
5 natural join register\_order;

ACCOUNT PASSWD ADMIN ORDERED  
-----  
dprrtQ4S70L 1680975342 1 0

执行计划  
-----  
Plan hash value: 1559136647

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	58	111 (2)	00:00:02
* 1	HASH JOIN		1	58	111 (2)	00:00:02
2	NESTED LOOPS					
3	NESTED LOOPS		1	38	7 (0)	00:00:01
4	TABLE ACCESS FULL	ACCOUNT_ADMIN	1	20	3 (0)	00:00:01
* 5	INDEX RANGE SCAN	ACCOUNT	1		2 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	USER_INFO	1	18	4 (0)	00:00:01
7	TABLE ACCESS FULL	REGISTER_ORDER	83914	1638K	103 (1)	00:00:02

Predicate Information (identified by operation id):  
-----  
1 - access("USER\_INFO"."ACCOUNT"="REGISTER\_ORDER"."ACCOUNT")  
5 - access("ACCOUNT\_ADMIN"."ACCOUNT"="USER\_INFO"."ACCOUNT")