# 1 选择最有效的表名顺序

## 1.1 无关联的表

Oracle的解析器按照从右到左的处理顺序处理from子句中的表名，在最后的表预先被处理，所以在要选择记录条数最少的表作为基础表。当oracle处理多个表时，会运用排序及合并的方式链接它们，首先，扫描第一个表(FROM子句中最后的那个表)并对进序，然后扫描第二个表(FROM子句中最后第二个表)，最后将所有从第二个表中检索出的记录与第一个表中合适记录进行合并。

选择TAB2作为基础表(最好的方法)：

select count(\*) from tab1, tab2; --执行时间0.96秒

选择TAB2作为基础表(不佳的方法)：

select count(\*) from tab2, tab1; --执行时间26.09秒

## 1.2 有关联的表

如果有3个以上的表连接查询，那就需要选择关联表作为基础表（就是排在from子句的最右面），交叉表是指那个被其他表所引用的表。

例：

EMP表描述了LOCATION表和CATEGORY表的交集：

select \* from location l, category c, emp e where e.emp\_no between 1000 and 2000 and e.cat\_no = c.cat\_no and e.locn = l.locn;

将比下列SQL更有效率：

select \* from emp e, location l, category c where e.cat\_no = c.cat\_no and e.locn = l.locn and e.emp\_no between 1000 and 2000;

# 2 where子句中的连接顺序

1 oracle采用自上而下的顺序解析where子句，根据这个原理，表之间的连接必须必须写在where 条件之前，那些可以过滤掉最大数量记录的条件必须写在where子句的末尾。

例如：

--低效，执行时间156.3秒

select \* from emp e where sal > 50000 and job = 'MANAGER' and 25 < (select count(\*) from emp where mgr = e.empno);

--高效,执行时间10.6秒

select \* from emp e where 25 < (select count(\*) from emp where

mgr = e.empno)

# 3 减少数据库的访问次数

低效：

select emp\_name, salary, grade from emp where emp\_no = 342;

高效：

select a.emp\_name, a.salary, a.grade, b.emp\_name, b.salary, b.grade from emp a, emp b where a.emp\_no = 342 and b.emp\_no = 291;

# 4 计算记录条数

通过索引检索，对索引列的计数时最快的count(\*)

# 5 尽量多使用COMMIT

只要有可能，在程序中尽量多使用COMMIT，这样程序的性能得到提高，需求也会因为COMMIT所释放的资源而减少：COMMIT所释放的资源：

• 回滚段上用于恢复数据的信息

• 被程序语句获得的锁

• redo log buffer 中的空间

• ORACLE为管理上述3种资源中的内部花费

# 6 用Where子句替换HAVING子句

避免使用HAVING子句，HAVING只会在检索出所有记录之后才对结果集进行过滤。这个处理需要排序，总计等操作。如果能通过WHERE子句限制记录的数目，那就能减少这方面的开销

例：

低效：

select region, avg(log\_size) from location group by region having region region != 'SYDNEY' and region != 'PERTH';

高效： select region, avg(log\_size) from location where region region != 'SYDNEY' and region != 'PERTH' group by region;

# 7 使用表的别名(Alias)

当在SQL语句中连接多个表时，请使用表的别名并把别名前缀于每个Column上。这样一来，就可以减少解析的时间并减少那些由Column歧义引起的语法错误

# 8 用EXISTS替代IN

在许多基于基础表的查询中，为了满足一个条件，往往需要对另一个表进行联接。在这种情况下，使用EXISTS(或NOT EXISTS)通常将提高查询的效率,因为not exits在执行时发现满足条件后会直接返回false;

在子查询中，NOT IN子句将执行一个内部的排序和合并。无论在哪种情况下，NOT IN都是最低效的，因为它对子查询中的表执行了一个全表遍历。为了避免使用NOT IN，我们可以把它改写成外连接Outer Joins，或NOT EXISTS。

低效：

select \* from emp –-基础表 where empno > 0 and deptno in (select deptno from dept where loc = 'MELB');

高效：

select \* from emp –-基础表 where empno > 0 and exists (select 'X' from dept where dept.deptno = emp.deptno and loc = 'MELB');

# 9 用表连接替换EXISTS

通常来说，采用表连接的方式比EXISTS更有效率

例如：

低效： select ename from emp e where exists (select 'X' from dept where dept\_no = e.dept\_no and dept\_cat = 'A');

高效： select ename from dept d, emp e where e.dept\_no = d.dept\_no and dept\_cat = 'A';

在RBO的情况下，前者的执行路径包括FILTER，后者使用NESTED LOOP。

# 10 避免使用耗费资源的操作

带有DISTINCT，UNION，MINUS，INTERSECT，ORDER BY的SQL语句会启动SQL引擎执行耗费资源的排序(SORT)功能。DISTINCT需要一次排序操作，而其他的至少需要执行两次排序。 例如，一个UNION查询，其中每个查询都带有GROUP BY子句，GROUP BY会触发嵌入排序(NESTED SORT); 这样，每个查询需要执行一次排序，然后在执行UNION时，又一个唯一排序(SORT UNIQUE)操作被执行而且它只能在前面的嵌入排序结束后才能开始执行。嵌入的排序的深度会大大影响查询的效率。

# 11 优化GROUP BY

提高GROUP BY 语句的效率，可以通过将不需要的记录在GROUP BY 之前过滤掉。下面两个查询返回相同结果但第二个明显就快了许多。

低效： select job, avg(sal) from emp group by job having job = 'PRESIDENT' or job = 'MANAGER';

高效： select job, avg(sal) from emp where job = 'PRESIDENT' or job = 'MANAGER' group by job;

# 12 like操作符

遇到 需要用到 LIKE 过滤的SQL语句，完全可以用 instr 代替。处理速度将显著提高。

# 13 union

union在进行表链接后会筛选掉重复的记录，所以在表链接后会对所产生的结果集进行排序运算，删除重复的记录再返回结果。实际大部分应用中是不会产生重复的记录，最常见的是过程表与历史

如：

select \* from gc\_dfys union select \* from ls\_jg\_dfys

这个SQL在运行时先取出两个表的结果，再用排序空间进行排序删除重复的记录，最后返回结果集，

如果表数据量大的话可能会导致用磁盘进行排序。

推荐方案：采用union ALL操作符替代union，因为union ALL操作只是简单的将两个结果合并后就返回