[**SQL语句优化技术分析**](http://www.cnblogs.com/wxj1020/archive/2008/04/27/1173638.html)

最近几周一直在进行数据库培训，老师精湛的技术和生动的讲解使我受益匪浅。为了让更多的新手受益，我抽空把SQL语句优化部分进行了整理，希望大家一起进步。

**一、操作符优化**

**1、IN 操作符**

用IN写出来的SQL的优点是比较容易写及清晰易懂，这比较适合现代软件开发的风格。但是用IN的SQL性能总是比较低的，从Oracle执行的步骤来分析用IN的SQL与不用IN的SQL有以下区别：

ORACLE 试图将其转换成多个表的连接，如果转换不成功则先执行IN里面的子查询，再查询外层的表记录，如果转换成功则直接采用多个表的连接方式查询。由此可见用 IN的SQL至少多了一个转换的过程。一般的SQL都可以转换成功，但对于含有分组统计等方面的SQL就不能转换了。

**推荐方案：**在业务密集的SQL当中尽量不采用IN操作符，用EXISTS 方案代替。

**2、NOT IN操作符**

此操作是强列不推荐使用的，因为它不能应用表的索引。

**推荐方案：**用NOT EXISTS 方案代替

**3、IS NULL 或IS NOT NULL操作**（判断字段是否为空）

判断字段是否为空一般是不会应用索引的，因为索引是不索引空值的。

**推荐方案**：用其它相同功能的操作运算代替，如：a is not null 改为 a>0 或a>’’等。不允许字段为空，而用一个缺省值代替空值，如申请中状态字段不允许为空，缺省为申请。

**4、> 及 < 操作符（大于或小于操作符）**

大 于或小于操作符一般情况下是不用调整的，因为它有索引就会采用索引查找，但有的情况下可以对它进行优化，如一个表有100万记录，一个数值型字段A，30 万记录的A=0，30万记录的A=1，39万记录的A=2，1万记录的A=3。那么执行A>2与A>=3的效果就有很大的区别了，因为A& gt;2时ORACLE会先找出为2的记录索引再进行比较，而A>=3时ORACLE则直接找到=3的记录索引。

**5、LIKE操作符**

LIKE操作符可以应用通配符查询，里面的通配符组合可能达到几乎是任意的查询，但是如果用得不好则会产生性能上的问题，如LIKE ‘%5400%’ 这种查询不会引用索引，而LIKE ‘X5400%’则会引用范围索引。

一 个实际例子：用YW\_YHJBQK表中营业编号后面的户标识号可来查询营业编号 YY\_BH LIKE ‘%5400%’ 这个条件会产生全表扫描，如果改成YY\_BH LIKE ’X5400%’ OR YY\_BH LIKE ’B5400%’ 则会利用YY\_BH的索引进行两个范围的查询，性能肯定大大提高。

**6、UNION操作符**

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表UNION。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。

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

**二、SQL书写的影响**

**1、同一功能同一性能不同写法SQL的影响。**

如一个SQL在A程序员写的为  Select \* from zl\_yhjbqk

B程序员写的为 Select \* from dlyx.zl\_yhjbqk（带表所有者的前缀）

C程序员写的为 Select \* from DLYX.ZLYHJBQK（大写表名）

D程序员写的为 Select \*  from DLYX.ZLYHJBQK（中间多了空格）

以 上四个SQL在ORACLE分析整理之后产生的结果及执行的时间是一样的，但是从ORACLE共享内存SGA的原理，可以得出ORACLE对每个SQL 都会对其进行一次分析，并且占用共享内存，如果将SQL的字符串及格式写得完全相同，则ORACLE只会分析一次，共享内存也只会留下一次的分析结果，这 不仅可以减少分析SQL的时间，而且可以减少共享内存重复的信息，ORACLE也可以准确统计SQL的执行频率。

**2、WHERE后面的条件顺序影响**

WHERE子句后面的条件顺序对大数据量表的查询会产生直接的影响。如：  
Select \* from zl\_yhjbqk where dy\_dj = '1KV以下' and xh\_bz=1  
Select \* from zl\_yhjbqk where xh\_bz=1 and dy\_dj = '1KV以下'  
以 上两个SQL中dy\_dj（电压等级）及xh\_bz（销户标志）两个字段都没进行索引，所以执行的时候都是全表扫描，第一条SQL的dy\_dj = '1KV以下'条件在记录集内比率为99%，而xh\_bz=1的比率只为0.5%，在进行第一条SQL的时候99%条记录都进行dy\_dj及xh\_bz的 比较，而在进行第二条SQL的时候0.5%条记录都进行dy\_dj及xh\_bz的比较，以此可以得出第二条SQL的CPU占用率明显比第一条低。

**3、查询表顺序的影响**

在 FROM后面的表中的列表顺序会对SQL执行性能影响，在没有索引及ORACLE没有对表进行统计分析的情况下，ORACLE会按表出现的顺序进行链接， 由此可见表的顺序不对时会产生十分耗服物器资源的数据交叉。（注：如果对表进行了统计分析，ORACLE会自动先进小表的链接，再进行大表的链接）

**三、SQL语句索引的利用**

**1、操作符优化（同上）**

**2、对条件字段的一些优化**

**采用函数处理的字段不能利用索引，**如：

substr(hbs\_bh,1,4)=’5400’，优化处理：hbs\_bh like ‘5400%’

trunc(sk\_rq)=trunc(sysdate)， 优化处理：sk\_rq>=trunc(sysdate) and sk\_rq<trunc(sysdate+1)

进行了显式或隐式的运算的字段不能进行索引，如：ss\_df+20>50，优化处理：ss\_df>30

‘X’ || hbs\_bh>’X5400021452’，优化处理：hbs\_bh>’5400021542’

sk\_rq+5=sysdate，优化处理：sk\_rq=sysdate-5

hbs\_bh=5401002554，优化处理：hbs\_bh=’ 5401002554’，注：此条件对hbs\_bh 进行隐式的to\_number转换，因为hbs\_bh字段是字符型。

**条件内包括了多个本表的字段运算时不能进行索引**，如：ys\_df>cx\_df，无法进行优化  
qc\_bh || kh\_bh=’5400250000’，优化处理：qc\_bh=’5400’ and kh\_bh=’250000’

**四、其他**

ORACLE 的提示功能是比较强的功能，也是比较复杂的应用，并且提示只是给ORACLE执行的一个建议，有时如果出于成本方面的考虑ORACLE也可能不会按提示进 行。根据实践应用，一般不建议开发人员应用ORACLE提示，因为各个数据库及服务器性能情况不一样，很可能一个地方性能提升了，但另一个地方却下降 了，ORACLE在SQL执行分析方面已经比较成熟，如果分析执行的路径不对首先应在数据库结构（主要是索引）、服务器当前性能（共享内存、磁盘文件碎 片）、数据库对象（表、索引）统计信息是否正确这几方面分析。