**数据库优化方案**

1. 高效地进行SQL语句设计：

通常情况下，可以采用下面的方法优化SQL对数据操作的表现：

（1）减少对数据库的查询次数，即减少对系统资源的请求，使用快照和显形图等分布式数据库对象可以减少对数据库的查询次数。

（2）尽量使用相同的或非常类似的SQL语句进行查询，这样不仅充分利用SQL共享池中的已经分析的语法树，要查询的数据在SGA中命中的可能性也会大大增加。

（3）避免不带任何条件的SQL语句的执行。没有任何条件的SQL语句在执行时，通常要进行FTS，数据库先定位一个数据块，然后按顺序依次查找其它数据，对于大型表这将是一个漫长的过程。

（4）如果对有些表中的数据有约束，最好在建表的SQL语句用描述完整性来实现，而不是用SQL程序中实现。

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。在业务密集的SQL当中尽量不采用IN操作符。

优化sql时，经常碰到使用in的语句，一定要用exists把它给换掉，因为Oracle在处理In时是按Or的方式做的，即使使用了索引也会很慢。

2、 NOT IN操作符

强列推荐不使用的，因为它不能应用表的索引。用NOT EXISTS或（外连接+判断为空）方案代替

3、IS NULL或IS NOT NULL操作

判断字段是否为空一般是不会应用索引的，因为B树索引是不索引空值的。

用其它相同功能的操作运算代替，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，如业扩申请中状态字段不允许为空，缺省为申请。

避免在索引列上使用IS NULL和IS NOT NULL 避免在索引中使用任何可以为空的列，ORACLE将无法使用该索引．对于单列索引，如果列包含空值，索引中将不存在此记录.对于复合索引，如果每个列都为空，索引中同样不存在此记录.如果至少有一个列不为空，则记录存在于索引中．举例:如果唯一性索引建立在表的A 列和B 列上,并且表中存在一条记录的A,B 值为(123,null) , ORACLE 将不接受下一条具有相同A,B值（123,null）的记录(插入).然而如果所有的索引列都为空，ORACLE将认为整个键值为空而空不等于空.因此你可以插入1000 条具有相同键值的记录,当然它们都是空!因为空值不存在于索引列中,所以WHERE子句中对索引列进行空值比较将使ORACLE停用该索引.

低效: (索引失效)

SELECT …FROM DEPARTMENT WHERE DEPT\_CODE ISNOTNULL;

高效: (索引有效)

SELECT …FROM DEPARTMENT WHERE DEPT\_CODE >=0;

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>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。

用>=替代>

高效:

SELECT …FROM DEPARTMENT WHERE DEPT\_CODE >=0;

低效:

SELECT\*FROM EMPWHERE DEPTNO >3

两者的区别在于, 前者DBMS将直接跳到第一个DEPT等于4的记录而后者将首先定位到DEPT NO=3的记录并且向前扫描到第一个DEPT大于3的记录.

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、用EXISTS替换DISTINCT：

当提交一个包含一对多表信息(比如部门表和雇员表)的查询时,避免在SELECT子句中使用DISTINCT. 一般可以考虑用EXIST 替换, EXISTS使查询更为迅速,因为RDBMS核心模块将在子查询的条件一旦满足后,立刻返回结果.

例子：

(低效):

SELECTDISTINCT DEPT\_NO,DEPT\_NAMEFROM DEPT D , EMP EWHERE D.DEPT\_NO = E.DEPT\_NO

(高效):

SELECT DEPT\_NO,DEPT\_NAMEFROM DEPT D WHEREEXISTS

　　(SELECT'X'FROM EMP EWHERE E.DEPT\_NO = D.DEPT\_NO);

如：

用EXISTS 替代IN、用NOT EXISTS替代NOT 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.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.

例子：

(高效):

SELECT\*FROM EMP (基础表)WHERE EMPNO >0ANDEXISTS

　　(SELECT'X'FROM DEPTWHERE DEPT.DEPTNO= EMP.DEPTNO AND LOC='MELB')

(低效):

SELECT\*FROM EMP (基础表)WHERE EMPNO >0AND DEPTNOIN

　　(SELECT DEP TNOFROM DEPT WHERE LOC ='MELB')

7、用UNION替换OR (适用于索引列)

通常情况下, 用UNION替换WHERE 子句中的OR 将会起到较好的效果.对索引列使用OR 将造成全表扫描. 注意,以上规则只针对多个索引列有效.如果有column 没有被索引, 查询效率可能会因为你没有选择OR而降低. 在下面的例子中, LOC\_ID和REGION 上都建有索引.

(高效):

SELECT LOC\_ID,LOC\_DESC,REGIONFROM LOCATION WHERE LOC\_ID =10

　　UNIONSELECT LOC\_ID , LOC\_DESC , REGIONFROM LOCATION WHERE REGION ='MELBOURNE'

(低效):

SELECT LOC\_ID,LOC\_DESC,REGIONFROM LOCATION WHERE LOC\_ID= 10OR REGION = 'MELBOURNE'

如果你坚持要用OR, 那就需要返回记录最少的索引列写在最前面.

8、用IN来替换OR

这是一条简单易记的规则，但是实际的执行效果还须检验，在ORACLE8i下，两者的执行路径似乎是相同的．

低效:

SELECT….FROM LOCATION WHERE LOC\_ID =10OR LOC\_ID=20OR LOC\_ID=30

高效:

SELECT…FROM LOCATION WHERE LOC\_IN IN (10,20,30);

二、SQL语句结构优化

1、SELECT子句中避免使用‘ \* ‘：

2、用TRUNCATE替代DELETE ：

用TRUNCATE替代DELETE删除全表记录：（大数据量的表用次方法）

当删除表中的记录时,在通常情况下,回滚段(rollback segments )用来存放可以被恢复的信息. 如果你没有COMMIT 事务,ORACLE会将数据恢复到删除之前的状态(准确地说是恢复到执行删除命令之前的状况)而当运用TRUNCATE 时, 回滚段不再存放任何可被恢复的信息.

3、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. (非oracle中)on、where、having这三个都可以加条件的子句中，on是最先执行，where 次之，having 最后，因为on是先把不符合条件的记录过滤后才进行统计，它就可以减少中间运算要处理的数据，按理说应该速度是最快的， where 也应该比having快点的

4、sql语句用大写

因为oracle 总是先解析sql语句，把小写的字母转换成大写的再执行。

5、在Java代码中尽量少用连接符“＋”连接字符串！

6、避免改变索引列的类型.:

当比较不同数据类型的数据时, ORACLE自动对列进行简单的类型转换. 假设EMPNO 是一个数值类型的索引列.

SELECT … FROM EMP WHERE EMPNO = ‘123'实际上,经过ORACLE类型转换, 语句转化为:

SELECT …FROM EMP WHERE EMPNO = TO\_NUMBER(‘123')

幸运的是,类型转换没有发生在索引列上,索引的用途没有被改变.现在,假设EMP\_TYPE是一个字符类型的索引列.

SELECT …FROM EMP WHERE EMP\_TYPE =123

这个语句被ORACLE转换为:

SELECT …FROM EMP WHERETO\_NUMBER(EMP\_TYPE)=123

因为内部发生的类型转换, 这个索引将不会被用到! 为了避免ORACLE对你的SQL 进行隐式的类型转换,最好把类型转换用显式表现出来.注意当字符和数值比较时, ORACLE会优先转换数值类型到字符类型

7、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:

1SELECT JOB,AVG(SAL)FROM EMP GROUPby JOBHAVING JOB= 'PRESIDENT' OR JOB ='MANAGER'

高效:

1SELECT JOB,AVG(SAL)FROM EMP WHERE JOB ='PRESIDENT'OR JOB='MANAGER'GROUPby JOB

数据库优化方案

1. 利用表分区

分区将数据在物理上分隔开，不同分区的数据可以制定保存在处于不同磁盘上的数据文件里。这样，当对这个表进行查询时，只需要在表分区中进行扫描，而不必进行全表扫描，明显缩短了查询时间，另外处于不同磁盘的分区也将对这个表的数据传输分散在不同的磁盘I/O，一个精心设置的分区可以将数据传输对磁盘I/O竞争均匀地分散开。对数据量大的时时表可采取此方法。可按月自动建表分区。

1. 别名的使用

别名是大型数据库的应用技巧，就是表名、列名在查询中以一个字母为别名，查询速度要比建连接表快1.5倍。

1. 索引Index的优化设计

索引可以大大加快数据库的查询速度，索引把表中的逻辑值映射到安全的RowID，因此索引能进行快速定位数据的物理地址。对一个建有索引的大型表的查询时，索引数据可能会用完所有的数据块缓存空间，ORACLE不得不频繁地进行磁盘读写来获取数据，因此在对一个大型表进行分区之后，可以根据相应的分区建立分区索引。但是个人觉得不是所有的表都需要建立索引，只针对大数据量的表建立索引。

缺点： 第一，创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。第二，索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。第三，当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。

索引需要维护：为了维护系统性能，索引在创建之后，由于频繁地对数据进行增加、删除、修改等操作使得索引页发生碎块，因此，必须对索引进行维护。

1. 调整硬盘I/O

这一步是在信息系统开发之前完成的。数据库管理员可以将组成同一个表空间的数据文件放在不同的硬盘上，做到硬盘之间I/O负载均衡。在磁盘比较富裕的情况下还应该遵循以下原则：

* 将表和索引分开；
* 创造用户表空间，与系统表空间（system）分开磁盘；
* 创建表和索引时指定不同的表空间；
* 创建回滚段专用的表空间，防止空间竞争影响事务的完成；
* 创建临时表空间用于排序操作，尽可能的防止数据库碎片存在于多个表空间中。

我们在使用物化视图的过程中基本可以“把它当作一个实际的数据表来看待”，不用再担心视图本身的基础表的效率、优化等

物化视图

1.对于复杂而高消耗的查询，如果使用频繁，应建成物化视图

2.物化视图是一种典型的以空间换时间的性能优化方式

3.对于更新频繁的表慎用物化视图

4.选择合适的刷新方式

一般的视图是虚拟的，而物化视图是实实在在的数据区域，是要占据存储空间的。

当然，物化视图在创建和管理上和一般的视图有不同的地方。相比来讲，物化视图占用了一定的存储空间，另外系统刷新物化视图也需要耗费一定的资源，但是它却换来了效率和灵活性。

减少IO与网络传输次数

1.尽量用较少的数据库请求，获取到需要的数据，能一次性取出的不分多次取出

2.对于频繁操作数据库的批量操作，应采用存储过程，减少不必要的网络传输

死锁与阻塞

1.对于需要频繁更新的数据，尽量避免放在长事务中，以免导致连锁反应

2.不是迫不得已，最好不要在ORACLE锁机制外再加自己设计的锁

3.减少事务大小，及时提交事务

4.尽量避免跨数据库的分布式事务，因为环境的复杂性，很容易导致阻塞

5.慎用位图索引，更新时容易导致死锁

自动增加表分区：

该程序可以做为一个Oracle的JOB执行在每月的28日前执行(考虑2月28天的原因),自动为该用户下的分区表增加分区.

create or replace procedure guan\_add\_partition

/\*

/\*为一个用户下所有分区表自动增加分区.分区的列为date类型,分区名类似:p200706.

/\*create by David

\*/

as

v\_table\_name varchar2(50);

v\_partition\_name varchar2(50);

v\_month char(6);

v\_add\_month\_1 char(6);

v\_sql\_string varchar2(2000);

v\_add\_month varchar2(20);

cursor cur\_part is select distinct u.table\_name,max(p.partition\_name) max\_part\_name from user\_tables u,user\_tab\_partitions p

where u.table\_name=p.table\_name and u.partitioned = 'YES'

group by u.table\_name;

Begin

select to\_char(sysdate,'yyyymm') into v\_month from dual;

select to\_char(add\_months(sysdate,1),'yyyymm') into v\_add\_month\_1 from dual;

select to\_char(add\_months(trunc(sysdate,'mm'),2),'yyyy-mm-dd') into v\_add\_month from dual;

open cur\_part;

loop

fetch cur\_part into v\_table\_name,v\_partition\_name;

exit when cur\_part%notfound;

if to\_number(substr(v\_partition\_name,2)) <=to\_number(substr(v\_month,1)) then

v\_sql\_string :='alter table '||v\_table\_name||' add partition p'||v\_add\_month\_1||

' VALUES LESS THAN ( to\_date('''||v\_add\_month||''',''yyyy-mm-dd'') ) tablespace users';

execute immediate v\_sql\_string;

else

null;

end if;

end loop;

close cur\_part;

end;