**ORACLE高效SQL书写规范**

**佰瑞管理咨询有限公司**

**Oracle DBA**

**赵磊**

**2013-09-16**

**V-20130916001**

**本规范适用于oracle DBA和应用开发人员参考使用**

**目录**

1. 书写格式规范

1-1、大小写字符及空格的统一

1-2、日期格式明确化

1-3、Bind变量的使用

1-4、表别名的使用

1-5、检索时尽量避免检索不需要的列

1-6、ORDER BY列明确指定

1-7、插入列明确指定

1-8、关联表数目的限制

1-9、子查询中不使用视图

1-10、Hint的写法

1-11、命名规范

1. 索引应用规范

2-1、WHERE子句中尽量不用OR

2-2、WHERE子句中尽量不用LIKE ‘%c%’, LIKE ’%c’

2-3、WHERE子句中尽量不用NOT

2-4、WHERE子句中尽量不用IS NULL, IS NOT NULL

2-5、在WHERE中慎用<>,!=

2-6、WHERE子句中IS NOT NULL使用的补充

2-7、尽量用EXISTS代替DISTINCT

2-8、ORDER BY使用上的注意点

2-9、Index列尽量不参加计算

2-10、复合Index中前面的列应在条件中指定

1. 其他规范

3-1、列比较时尽量保持类型一致

3-2、尽量避免使用子查询

3-3、对子查询层次限定

3-4、尽量用NOT EXISTS代替NOT IN + 子查询

3-5、用表连接替换EXISTS

3-5、尽量不使用HAVING子句

3-6、为FROM子句中的表指定顺序

3-7、WHERE子句中的连接顺序

3-8、ROWID的使用

3-9、用ROWNUM判断纪录是否存在

3-10、翻页SQL(下N件记录取得)的写法

3-11、MERGE的使用

3-12、多表插入的使用

3-13、DBlink使用的限制

3-14、尽量用DECODE代替集合函数

3-15、删除重复记录

3-16、减量减少对表的查询

3-17、避免使用耗费资源的操作

当数据库从客户应用程序接收到一条SQL语句时，它首先检查是否该语句在共享池中。若在，那么不重新对其分析而是重复使用已经在共享池中的形式及执行方案；若不在，则对该语句进行分析，得到新的分析形式与执行方案并进行存储，覆盖共享池中以前的内容。由此可以看出，编写可重用共享池中语句的SQL语句就显得十分必要，因为避免不必要的重新分析，会很大程度上减小服务器所承担的工作量。要想重复使用共享池中的语句，就应该编写与其格式一致的语句，包括字母的大小写、标点符号、换行的位置等都要一致。

1. 书写格式规范

1-1、大小文字及空格的统一

数据库对象名（表、索引等）和变量全半角小写，其他的SQL文全半角大写，避免不必要的空格，如果使用空格只使用半角空格，既保持SQL文的可读性，又尽可能的减少SQL的解析时间。

因为ORACLE的共享SQL机制，只有在共享池中找到完全匹配的(字符级比较，包括空格，字符大小写)SQL才不需要再次解析(parsing)被执行SQL，所以应尽量保持统一风格的SQL。

错：select E.EMP\_NO FROM emp E;

对：SELECT e.emp\_no FROM emp e;

1-2、日期格式明确化

如果不明确的指定时间的格式，默认使用NLS\_DATE\_FORMAT参数指定的时间格式，就有可能得不到你想要的结果，所以需要用TO\_CHAR或TO\_DATE函数明确转换再使用。

错：SELECT e.ename, e.hire\_date FROM emp e WHERE e.hire\_date > ‘20100726’

对：SELECT e.ename, TO\_CHAR(e.hire\_date,’YYYYMMDD’) FROM emp e WHERE e.hire\_date > TO\_DATE(‘20100726’,’YYYYMMDD’)

1-3、Bind变量的使用

对于相同条件不同条件值的SQL文，采用bind变量使其变成同一SQL文，从而减少ORACLE对它的解析时间。(Java语言中一般用?来作为bind变量)

错：SELECT e.ename FROM emp e WHERE e.emp\_no = 123;

SELECT e.ename FROM emp e WHERE e.emp\_no = 567;

对：SELECT e.ename FROM emp e WHERE e.emp\_no = ?;

1-4、表别名的使用

在进行多表关联查询时，检索列前加上表别名，这样既明确列的定义，又减少了SQL文的解析时间。

错：SELECT ename, dname FROM emp, dept WHERE emp.deptno=dept.deptno AND sal >1000;

对：SELECT e.ename, d.dname FROM emp e, dept d WHERE e.deptno=d.deptno AND e.sal >1000;

1-5、检索时尽量避免检索不需要的列

在检索纪录时，特别是纪录的列比较多时，尽量不要使用‘\*’来代替所有列，这样不仅增加处理时间(ORACLE在解析的过程中，会将‘\*’ 依次转换成所有的列名，这个工作是通过查询数据字典完成的，这意味着将耗费更多的时间，同时也增加了Ｉ／Ｏ的量)，而且当表结构变化时，原来的列顺序有可能完全改变而导致不必要的bug或修改。

错：SELECT \* FROM emp WHERE eadrs = “上海市”;

对：SELECT e.empno e.ename FROM emp WHERE e.eadrs = “上海市”;

1-6、ORDER BY列明确指定

虽然使用ORDER BY字句时可以根据SELECT句中检索列的列号来指定所需排序的列，但是从代码的维护性考虑，应该尽量不使用列号而是明确指定排序列来排序。

错：SELECT e.emp\_no, e.ename, e.sal FROM emp e ORDER BY 2;

对：SELECT e.emp\_no, e.ename, e.sal FROM emp e ORDER BY e.ename;

1-7、插入列明确指定

在对对象表进行全列插入时，SQL文法上允许省略对列的指定，但是考虑到因表结构（列的增删改）有可能发生变化，而减少不要的修改，即使全列插入也需要明确指定插入列。

错：INSERT INTO emp VALUES(100, ‘AAA’);

对：INSERT INTO emp(emp\_no, ename) VALUES(100, ‘AAA’);

1-8、关联表数目的限制

在进行多表关联查询时，随着关联表的增多ORACLE访问路径也会变得越来越复杂，从而导致ORACLE执行了不合理的实行计划，所以要避免不必要的关联。

错：SELECT ~ FROM emp1 e1, emp2 e2, emp3 e3; //emp3为不需要关联的表

对：SELECT ~ FROM emp1 e1, emp2 e2;

1-9、子查询中不使用视图

在子查询中使用视图会使ORACLE的访问路径变得很复杂，从而导致ORACLE执行了不合理的实行计划，所以尽量不要在子查询中使用视图。

CREATE VIEW emp\_v AS SELECT e.name, e.sal\*12 sal FROM emp e WHERE e.deptno < 120;

错：SELECT t1.dept\_no, t2.ename FROM dept1 t1, (SELECT v.ename, FROM emp\_v WHERE v.sal < 3000) t2 WHERE t1.dept\_no = t2.dept\_no;

对：SELECT t1.dept\_no, t2.ename FROM dept1 t1, (SELECT v.ename, FROM emp e WHERE e.salary < (3000/12) AND e.dept\_no < 120) t2 WHERE t1.dept\_no = t2.dept\_no;

1-10、Hint的写法

一般的SQL中不考虑使用hint句，除非在性能试验等试验结果性能很差的时候才考虑使用。（使用hint之后会改变原有的实行计划）

错：SELECT /\*+FIRST\_ROWS\*/ e.ename FROM emp e WHERE e.emp\_no = 'SCOTT';

对：SELECT e.ename FROM emp e WHERE e.emp\_no = 'SCOTT';

1-11、命名规范

表别名，列别名命名时，尽量按照原表名和列名的省略缩写形式，保持SQL的可读性。

错：[表名]employee -->[别名]a

对：[表名]employee -->[别名]emp

1. 索引应用规范

2-1、WHERE子句中尽量不用OR

使用了OR之后，有的情况下(涉及到索引的扫描方式)会使对索引的检索变成无效，从而降低检索的性能，这这种情况下，可以考虑是否可以用UNION或IN来代替。

RBO（基于规则）下，OR就像UNION ALL文的展开一样，在有外部结合、CONNECT BY句的情况下就不能被执行。CBO（基于成本）下，OR根据各结合顺序的基础结合成本，在再估算阶段，根据成本对INLIST和OR文能不能再扩展进行判断。

错1：SELECT name FROM emp WHERE deptno = 99 OR deptno = 999;

对1：SELECT name FROM emp WHERE deptno = 99

UNION ALL

SELECT name FROM emp WHERE deptno = 999;

错2：~ FROM emp WHERE deptno = 10 OR deptno = 20 OR teptno = 30;

对2：~ FROM emp WHERE deptno IN(10,20,30);

2-2、WHERE子句中尽量不用LIKE ‘%c%’, LIKE ’%c’

在有索引的列上使用LIKE ‘%c%’, LIKE ’%c’之后，索引就会失效，因而性能也会下降。

错：~ WHERE name LIKE ‘%c%’;

错：~ WHERE name LIKE ’%c’;

对：~ WHERE name LIKE ’ c%’;

2-3、WHERE子句中尽量不用NOT

在有索引的列上使用NOT、!=、 <>之后，索引检索就会失效，因而性能也会下降。

例1：job列大部分值为NULL或’SALESMAN’，并且job列上建有B\*Tree索引。

错：~ WHERE job != ‘SALESMAN’

对：~ WHERE job > ‘SALESMAN’ OR job < ‘SALESMAN’

例2：dname为建有B\*Tree索引的文字列，并且基数比较少（值比较少，譬如性别列，只有男和女两个值），且不经常被更新。

错：~ WHERE dname IS NOT NULL;

对：DROP INDEX btree\_dname\_idx;

CREATE BITMAP INDEX bitmap\_dname\_idx ON emp(dname);

SELECT ~ WHERE dname IS NOT NULL;

2-4、WHERE子句中尽量不用IS NULL, IS NOT NULL

在用了IS NULL、IS NOT NULL 之后，对应检索列的B\*Tree索引就会失去索引功能，从而性能会大幅下降。

在使用IS NULL的时候也应该考虑一下是否真的需要NULL值。IS NOT NULL在CBO下统计情报取得的情况下，索引的检索也有可能被使用（参照IS NOT NULL使用的补充）。即使使用IS NULL、IS NOT NULL，BITMAP索引还是照样能被使用。

错：Hiredate列为日期型，在hiredate列上建有B\*Tree索引，并且’9999-12-31’是系统中不可能出现的日期，执行下面的SQL

SELECT ~ WHERE hiredate IS NULL;

对：CREATE INDEX function\_hiredate\_idx ON emp(NVL(hiredate, TO\_DATE(‘9999-12-31’)));

SELECT ~ WHERE NVL(hiredate, TO\_DATE(‘9999-12-31’)) = TO\_DATE(‘9999-12-31’, ‘YYYYMMDD’);

2-5、在WHERE中慎用<>,!=

记住索引只能告诉你什么存在于表中，而不能告诉你什么不存在于表中。

在下面的例子中‘!=' 将不使用索引。

不使用索引：SELECT account\_name FROM transaction WHERE amout !=0;

使用索引：SELECT account\_name FROM transaction WHERE amout>0;

2-6、WHERE子句中IS NOT NULL使用的补充

在NULL值比率较低的列上使用IS NOT NULL，会使对B\*Tree索引变得无效，从而降低检索性能，相反，在NULL值比率相当高的列上使用IS NOT NULL，B\*Tree索引会使用全表扫描(full scan)从而提高性能。（这是在CBO中统计情报已经取得的条件下）

例：Comm为建有B\*Tree索引的数值型列，且NULL值比率相当高（CBO并且统计情报已经取得）

错：~ WHERE comm >= 0 OR comm < 0;

对：~ WHERE comm IS NOT NULL;

2-7、尽量用EXISTS代替DISTINCT

DEPT表和EMP表是一对多的关系，根据EMP表从DEPT表中取出纪录，这种情况要避免使用DISTINCT，因为用DISTINCT后满足条件的所有纪录将都会被检索、排序、重复行删除，从而影响了性能。取而代之的可以用EXSITS子查询，EXSITS子查询当有一件符合条件的纪录存在时马上就返回不处理剩下的纪录，因而速度很快。

错：SELECT DISTINCT d.dept\_code, d.dept\_name FROM dept d, emp e WHERE e.dept\_code = d.dept\_code;

对：SELECT d.dept\_code, d.dept\_name FROM dept d WHERE EXSITS(SELECT ‘X’ FROM emp e WHERE e.dept\_code = d.dept\_code);

2-8、ORDER BY使用上的注意点

如果想要ORDER BY句的排序用到索引，就必须满足下面的两个条件

1）、ORDER BY句中的列顺序要完全包含在同一顺序的一个复合索引中

2）、ORDER BY句中的列要全部在表中定义为NOT NULL列

2-9、Index列尽量不参加计算

检索条件中索引列被参与计算，或被用作函数的参数，那么就会失去该列的索引功能，从而导致性能急剧下降。

可以通过建函数索引的方法，计算结果或函数值事前计算好作为所引来用。

错：~ WHERE sal\*1.1 > 950;

对：~ WHERE sal > 950/1.1;

错：~ WHERE name || type =’XXXY’ ;

对：~ WHERE name = ‘XXX’ AND type = ’Y’ ;

错：~ WHERE TO\_CHAR(hiredate, ‘YYYYMMDD’) = ‘20100722’;

对：~ WHERE hiredate = TO\_DATE( ‘20100722’ , ‘YYYYMMDD’) ;

错：~ WHERE SUBSTR(name, 1, 7) = ‘CAPTIAL’;

对：~ WHERE name LIKE ‘CAPTIAL%’;

错：~ WHERE TRUNC(trans\_date) = TRUNC(SYSDATE);

对：~ WHERE trans\_date BETWEEN TRUNC(SYSDATE) AND TURNC(SYSDATE) + .99999

注：在日期上加上超过5位小数的数是，日期就自动变为下一日的日期。

TO\_DATE(‘2010-7-22’)+.99999 🡪 ‘2010-7-22 23:59:59’

TO\_DATE(‘2010-7-22’)+.999999 🡪 ‘2010-7-23 00:00:00’

2-10、复合Index中前面的列应在条件中指定

在使用复合索引时，复合索引中前面的列要在条件中指定。如果前面列在条件中没指定的情况下，虽然通过索引的SKIP SCAN机能有可能会使用索引，但是一定要确认索引是否真的被使用。

(depno, job)列顺序的复合索引

对：~ WHERE depno = 20 AN job = ‘MANAGER’;

对：~ WHERE job = ‘MANAGER’ AND depno = 20;

对：~ WHERE depno = 20; //索引前方部分使用

对：~ WHERE job = ‘MANAGER’; //这种情况下，一定要确认索引是否被使用

2-11、in 和 not in 也要慎用，否则会导致全表扫描，如：   
select id from t where num in(1,2,3)   
对于连续的数值，能用 between 就不要用 in 了：   
select id from t where num between 1 and 3

1. 其他规范

3-1、列比较时尽量保持类型一致

数值型列和文字列比较的时候，文字列会自动的转换成数值型，但是这样的处理会使基于索引的检索无效，从而降低性能。

例：

对：~ WHERE emp\_no = 123 (emp\_no建有索引的数值列)

可能不对：~ WHERE emp\_no = ‘123’ (emp\_no建有索引的数值列)

注）ORACLE在实际执行中会做~ WHERE TO\_CHAR(emp\_no) = ‘123’的转换而可能导致索引不被使用

对：~ WHERE emp\_type = ‘123’ (emp\_tepe建有索引的文字列)

错：~ WHERE emp\_type = 123 (emp\_tepe建有索引的文字列)

注）ORACLE在实际执行中会做~ WHERE TO\_NUMBER(emp\_no) = ‘123’的转换而导致索引使用无效

3-2、尽量避免使用子查询

不使用子查询的SQL访问路径上自由度增大，处理性能上也提升不少，所以原则上不使用子查询而通过考虑是否能用通常的结合处理来代替。

错：SELECT e.ename FROM emp e, (SELECT d1.deptno FROM dept d1 WHERE d1.deptno = ‘A’) d WHERE e.deptno = d.deptno;

对：SELECT e.ename FROM emp e, dept d WHERE e.deptno = d.deptno AND d.deptno = ‘A’;

3-3、对子查询层次限定

在子查询中再嵌套子查询会使ORACLE的实行计划变得更加复杂，随着子查询的重度越高执行不确切的实行计划的可能性越高。因此，尽量使避免使用嵌套子查询。

3-4、尽量用NOT EXISTS代替NOT IN + 子查询

在子查询中使用了NOT IN演算后，会发生内部排序、合并处理，为了提高性能，可以用NOT EXISTS来代替NOT IN + 子查询。

错：SELECT e.ename FROM emp e WHERE e.deptno NOT IN(SELECT d.deptno FROM dept d WHERE d.deptno = e.deptno AND d.dept\_cat = ’A’);

对：SELECT e.ename FROM emp e WHERE NOT EXISTS(SELECT ‘X’ FROM dept d WHERE d.deptno = e.deptno AND d.dept\_cat = ’A’);

3-5、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。

注）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。

低效：SELECT DISTINCT dept\_no,dept\_name FROM deptd,empe

   WHERE d.dept\_no = e.dept\_no;

高效：SELECT dept\_no,dept\_name FROM deptd

   WHERE EXISTS ( SELECT ‘X’ FROM empe

WHERE e.dept\_no = d.dept\_no);

3-6、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

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

低效：SELECT ename

   FROM empe

   WHERE EXISTS (SELECT ‘X’ FROM dept

                   WHERE dept\_no = e.dept\_no AND DEPT\_CAT = ‘A’);

　高效：SELECT ename FROM deptd, empe

      WHERE e.dept\_no = d.dept\_no AND dept\_cat = ‘A’ ;

3-7、尽量不使用HAVING子句

HAVING子句是把全部的纪录选中之后，对选择的纪录再按条件进行过滤，相比之下用WHERE子句来对选择的纪录进行过滤性能更胜一筹，所以在用HAVING子句的时候考虑一下是不是能够移到WHERE子句中实现。

错：

SELECT e.deptno AVG(e.sal) FROM emp e GROUP BY e.deptno HAVING e.deptno > 10;

对：

SELECT e.deptno AVG(e.sal) FROM emp e WHERE e.deptno > 10 GROUP BY e.deptno;

3-8、基础表的选择

基础表（Driving Table，也被叫做驱动表）是指被最先访问的表（通常以全表扫描的方式被访问）。根据优化器的不同，SQL语句中基础表的选择是不一样的。

CBO模式下，优化器会检查SQL语句中的每个表的物理大小，索引的状态，然后选用花费最低的执行路径。

RBO模式下，表结合的条件全部相同的情况下，FROM子句中从右到左结合，所FROM子句最右边的表为基表。因为基表纪录数越少性能越好，所以FROM子句中按纪录数从大到小顺序指定结合表。

注）但是条件指定后件数变化的情况下，按照变化后的件数从大到小来指定。

CBO例：SELECT a.name，b.manager FROM　worker a，lodging b

WHERE　a.loding = b.loding;

由于lodging表的loding列上有一个索引， 而且worker表中没有相比较的索引，worker表将被作为查询中的基础表。

RBO例：

错：~ FROM a,b,c; (纪录数 c < b < a)

对：~ FROM c,b,a; (纪录数 c < b < a)

3-9、WHERE子句中的连接顺序

ORACLE采用自下而上的顺序解析WHERE子句，根据这个原理，表之间的连接必须写在其他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。

错：SELECT …

FROM emp e

WHERE sal> 50000

AND    job = ‘MANAGER’

AND    25 < (SELECT COUNT(\*)FROM emp WHERE mgr=e.empno);

对：SELECT …

FROM empe

WHERE 25 < (SELECT COUNT(\*) FROM emp WHERE mgr=e.empno)

AND    sal > 50000

AND    job = ‘MANAGER’;

3-10、ROWID的使用

在通过SELECT结果作为条件来DELTE或者UPDATE的场合，使用ROWID作为条件能减轻负荷。

注）ROWID是数据库中实际的ORACLE数据块对应的物理纪录番号，通过ROWID检索是最快的检索方式。

错：SELECT e.ename INTO:emp\_ename FROM emp e WHERE e.emp\_no = 123 FOR UPDATE OF e.ename;

UPDATE emp e SET e.ename = ‘XXX’WHERE e.emp\_no = 123;

对：SELECT e.ename e.rowid INTO:emp\_ename,:emp\_rowid FROM emp e WHERE e.emp\_no = 123 FOR UPDATE OF e.ename;

UPDATE emp e SET e.ename = ‘XXX’WHERE ROWID = e.emp\_rowid;

3-11、用ROWNUM判断纪录是否存在

记录存在与否用ROWNUM<=1来判断效率更高，因为当指定ROWNUM<=1的时候，只要一条记录被找到SQL的执行就结束了，所以能很好的提升执行的速度。

错：SELECT ‘X’FROM dual WHERE EXISTS(SELECT ‘X’FROM emp WHERE sal > 100);

对：SELECT ‘X’FROM dual WHERE sal > 100 AND ROWNUM<=1;

3-12、翻页SQL(下N件记录取得)的写法

在取下N件纪录的时候，在FORM子句中用ROWNUM限定结果集，然后再在WHERE句中的指定所取件数，这样会大大减少response的性能。

错：SELECT row\_num,empno,name FROM (SELECT ROWNUM row\_num, empno,name WHERE emp) WHERE row\_num >= 6 AND row\_num <= 10;

对：SELECT row\_num,empno,name FROM (SELECT ROWNUM row\_num, empno,name WHERE emp ROWNUM <= 10) WHERE row\_num >= 6;

3-11、MERGE的使用

MERGE是ORACLE 9i之后才用的新机能，对应条件的纪录存在的时候执行UPDATE,不存在的时候执行INSERT处理。使用MERGE可以简化编程工作量及复杂度。

错：DECLARE

CURSOR dept\_cur IS SELECT \* FROM dept FOP UPDATE;

wk dept\_cur%ROWTYPE;

BEGIN

FOR wk IN dept\_cur LOOP

UPDATE dept2 SET dname = wk.dname WHERE deptno = wk.deptno;

IF SQL%NOTFOUND THEN

INSERT INTO dept2 VALUES(wk.deptno,dname,wk.loc);

END IF

END LOOP;

END;

对：MERGE INTO dept2 d2

USING dept d

ON (d2.deptno = d.deptno)

WHERE MATCHED THEN

UPDATE SET d2.dname = d.dname

WHEN NO MATCHED THEN

INSERT(d2.deptno,d2.dname,d2.loc) VALUES(d.deptno,d.name,d.loc);

3-13、多表插入的使用

同一数据源插入复数个表，这在ORACLE 9i之前需要执行复数个单独的INSERT ~

SELECT语句，但是在ORACLE 9i以后我们可以用多表插入功能，只要执行一次INSERT文就可以向复数的表中插入数据。

过去：

INSERT INTO sales\_month\_a(month,sales) SELECT month,SUM(sales) FROM sales WHERE prod\_id = ‘PROD\_A’ GROUP BY month;

INSERT INTO sales\_month\_b(month,sales) SELECT month,SUM(sales) FROM sales WHERE prod\_id = ‘PROD\_B’ GROUP BY month;

现在：

INSERT INTO FIRST

WHEN prod\_id = ‘PROD\_A’INTO sales\_month\_a(month,sales) VALUES(month,slaes)

WHEN prod\_id = ‘PROD\_B’INTO sales\_month\_b(month,sales) VALUES(month,slaes)

SELECT month,SUM(sales) FROM sales GROUP BY month;

3-14、DBlink使用的限制

分散DB环境中，通过DBlink使用不同DB中的表进行结合处理，会因为传送数据量的过多而造成性能严重低下。所以尽量用其他方法来替换不同DB中的表结合处理。

3-15、尽量用DECODE代替集合函数

为了减轻集合函数中的总数或合计的计算的负荷，尽量使用DECODE来减少处理时间。使用DECODE函数可以避免重复扫描相同记录或重复连接相同的表。

错：SELECT COUNT(\*),SUM(sal) FROM emp WHERE deptno=10;

SELECT COUNT(\*),SUM(sal) FROM emp WHERE deptno=20;

对：SELECT COUNT(DECODE(deptno,10,’D10’)) “COUNT-10”,

COUNT(DECODE(deptno,20,’D20’)) “COUNT-20”,

SUM(DECODE(deptno,10,sal,NULL) “SUM-10”,

SUM(DECODE(deptno,20,sal,NULL) “SUM-20”

FROM emp;

3-16、删除重复记录

最高效的删除重复记录方法（因为使用了ROWID）

DELETE FROM emp e

WHERE E.ROWID > (SELECT MIN(x.ROWID)

                   FROM empx

                   WHERE x.emp\_no = e.emp\_no);

3-17、减量减少对表的查询

在含有子查询的SQL语句中，要特别注意减少对表的查询。

错：SELECT tab\_name

       FROM tables

        WHERE tab\_name = ( SELECT tab\_name

FROM tab\_colums

                                WHERE version = 604)

          AND　db\_ver= ( SELECT db\_ver

                           FROM tab\_colums

                           WHERE version= 604);

        对：SELECT tab\_name

          FROM tables

          WHERE (tab\_name,db\_ver)= ( SELECT tab\_name,db\_ver)

                   FROM tab\_colums

                   WHERE version = 604);

3-18、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。通常, 带有UNION, MINUS , INTERSECT的SQL语句都可以用其他方式重写。