Oracle SQL性能优化40条

**1. SQL语句执行步骤**

语法分析> 语义分析> 视图转换 >表达式转换> 选择优化器 >选择连接方式 >选择连接顺序 >选择数据的搜索路径 >运行“执行计划”

**2. 选用适合的Oracle优化器**

RULE（基于规则）、 COST（基于成本） 、CHOOSE（选择性）

**3. 访问Table的方式**

**全表扫描**

全表扫描就是顺序地访问表中每条记录，ORACLE采用一次读入多个数据块(database block)的方式优化全表扫描。

**通过ROWID访问表**

ROWID包含了表中记录的物理位置信息，ORACLE采用索引实现了数据和存放数据的物理位置(ROWID)之间的联系，通常索引提供了快速访问ROWID的方法，因此那些基于索引列的查询就可以得到性能上的提高。

**4. 共享 SQL 语句**

Oracle提供对执行过的SQL语句进行高速缓冲的机制。被解析过并且确定了执行路径的SQL语句存放在SGA的共享池中。

Oracle执行一个SQL语句之前每次先从SGA共享池中查找是否有缓冲的SQL语句，如果有则直接执行该SQL语句。

可以通过适当调整SGA共享池大小来达到提高Oracle执行性能的目的。

**5. 选择最有效率的表名顺序**

ORACLE的解析器按照从右到左的顺序处理FROM子句中的表名，因此FROM子句中写在最后的表(基础表 driving table)将被最先处理。

当ORACLE处理多个表时，会运用排序及合并的方式连接它们，并且是从右往左的顺序处理FROM子句。首先，扫描第一个表(FROM子句中最后的那个表)并对记录进行排序，然后扫描第二个表(FROM子句中倒数第二个表)，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。

只在基于规则的优化器中有效。

举例：

表 TAB1 16,384 条记录

表 TAB2 1 条记录

/\*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TAB1作为基础表 (不佳的方法)\*/SELECT COUNT(\*) FROM TAB2,TAB1 /\*执行时间26.09秒\*/

如果有3个以上的表连接查询, 那就需要选择交叉表(intersection table)作为基础表, 交叉表是指那个被其他表所引用的表。

/\*高效的SQL\*/

SELECT \* FROM LOCATION L, CATEGORY C, EMP E WHERE E.EMP\_NO BETWEEN 1000 AND 2000AND E.CAT\_NO = C.CAT\_NOAND E.LOCN = L.LOCN

将比下列SQL更有效率

/\*低效的SQL\*/SELECT \* FROM EMP E, LOCATION L, CATEGORY CWHERE E.CAT\_NO = C.CAT\_NOAND E.LOCN = L.LOCNAND E.EMP\_NO BETWEEN 1000 AND 2000

**6. Where子句中的连接顺序**

Oracle采用自下而上或自右向左的顺序解析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。

/\*低效,执行时间156.3秒\*/SELECT Column1,Column2 FROM EMP E WHERE E.SAL > 50000 AND E.JOB = 'MANAGER'AND 25 < (SELECT COUNT(\*) FROM EMP WHERE MGR = E.EMPNO)

/\*高效,执行时间10.6秒\*/SELECT Column1,Column2FROM EMP EWHERE 25 < (SELECT COUNT(\*) FROM EMPWHERE MGR=E.EMPNO)AND E.SAL > 50000 AND E.JOB = 'MANAGER'

**7. SELECT子句中避免使用“\*”**

Oracle在解析SQL语句的时候，对于“\*”将通过查询数据库字典来将其转换成对应的列名。

如果在Select子句中需要列出所有的Column时，建议列出所有的Column名称，而不是简单的用“\*”来替代，这样可以减少多于的数据库查询开销。

**8. 减少访问数据库的次数**

当执行每条SQL语句时, ORACLE在内部执行了许多工作：解析SQL语句 > 估算索引的利用率 > 绑定变量 > 读数据块等等

由此可见, 减少访问数据库的次数 , 就能实际上减少ORACLE的工作量。

**9. 整个简单无关联的数据库访问**

如果有几个简单的数据库查询语句，你可以把它们整合到一个查询中（即使它们之间没有关系），以减少多于的数据库IO开销。

虽然采取这种方法，效率得到提高，但是程序的可读性大大降低，所以还是要权衡之间的利弊。

**10. 使用Truncate而非Delete**

Delete表中记录的时候，Oracle会在Rollback段中保存删除信息以备恢复。Truncate删除表中记录的时候不保存删除信息，不能恢复。因此Truncate删除记录比Delete快，而且占用资源少。

删除表中记录的时候，如果不需要恢复的情况之下应该尽量使用Truncate而不是Delete。

Truncate仅适用于删除全表的记录。

**11. 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

**12. 计算记录条数**

Select count(\*) from tablename;

Select count(1) from tablename;

Select count(column) from tablename;

一般认为，在没有主键索引的情况之下，第二种COUNT(1)方式最快。如果只有一列且无索引COUNT(\*)反而比较快， 如果有索引列，当然是使用索引列COUNT(column)最快。

**13. 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，就能减少这方面的开销。

**14. 减少对表的查询操作**

在含有子查询的SQL语句中，要注意减少对表的查询操作。

/\*低效SQL\*/SELECT TAB\_NAME FROM TABLESWHERE TAB\_NAME =(SELECT TAB\_NAME FROM TAB\_COLUMNS WHERE VERSION = 604)AND DB\_VER =(SELECT DB\_VER FROM TAB\_COLUMNSWHERE VERSION = 604)

/\*高效SQL\*/SELECT TAB\_NAME FROM TABLES WHERE (TAB\_NAME，DB\_VER)=(SELECT TAB\_NAME，DB\_VERFROM TAB\_COLUMNSWHERE VERSION = 604)

**15. 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。

Column歧义指的是由于SQL中不同的表具有相同的Column名,当SQL语句中出现这个Column时,SQL解析器无法判断这个Column的归属。

**16. 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。

/\*低效SQL\*/SELECT \* FROM EMP WHERE EMPNO > 0 AND DEPTNO IN (SELECT DEPTNO FROM DEPT WHERE LOC = 'MELB')

/\*高效SQL\*/SELECT \* FROM EMPWHERE EMPNO > 0 AND EXISTS (SELECT 1FROM DEPT WHERE DEPT.DEPTNO = EMP.DEPTNO AND LOC = 'MELB')

**17. 用NOT EXISTS替代NOT IN**

在子查询中，NOT IN子句将执行一个内部的排序和合并，对子查询中的表执行一个全表遍历，因此是非常低效的。

为了避免使用NOT IN，可以把它改写成外连接（Outer Joins）或者NOT EXISTS。

/\*低效SQL\*/SELECT \* FROM EMP WHERE DEPT\_NO NOT IN (SELECT DEPT\_NO FROM DEPT WHERE DEPT\_CAT='A')

/\*高效SQL\*/SELECT \* FROM EMP EWHERE NOT EXISTS (SELECT 1FROM DEPT DWHERE D.DEPT\_NO = E.DEPT\_NOAND DEPT\_CAT ='A')

**18. 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 。

/\*低效SQL\*/SELECT ENAMEFROM EMP EWHERE EXISTS (SELECT 1FROM DEPTWHERE DEPT\_NO = E.DEPT\_NOAND DEPT\_CAT = 'A')

/\*高效SQL\*/SELECT ENAMEFROM DEPT D，EMP EWHERE E.DEPT\_NO = D.DEPT\_NOAND D.DEPT\_CAT = 'A'

**19. 用EXISTS替换DISTINCT**

当提交一个包含对多表信息（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。

/\*低效SQL\*/SELECT DISTINCT D.DEPT\_NO，D.DEPT\_NAME FROM DEPT D，EMP EWHERE D.DEPT\_NO = E.DEPT\_NO

/\*高效SQL\*/SELECT D.DEPT\_NO，D.DEPT\_NAMEFROM DEPT D WHERE EXISTS (SELECT 1 FROM EMP EWHERE E.DEPT\_NO = D.DEPT\_NO)

**20. 识别低效的SQL语句**

下面的SQL工具可以找出低效SQL，前提是需要DBA权限，否则查询不了。

SELECT EXECUTIONS, DISK\_READS, BUFFER\_GETS,ROUND ((BUFFER\_GETS-DISK\_READS)/BUFFER\_GETS, 2) Hit\_radio,ROUND (DISK\_READS/EXECUTIONS, 2) Reads\_per\_run, SQL\_TEXTFROM V$SQLAREAWHERE EXECUTIONS>0AND  BUFFER\_GETS > 0 AND (BUFFER\_GETS-DISK\_READS)/BUFFER\_GETS < 0.8ORDER BY 4 DESC

另外也可以使用SQL Trace工具来收集正在执行的SQL的性能状态数据，包括解析次数，执行次数，CPU使用时间等 。

**21. 用Explain Plan分析SQL语句**

EXPLAIN PLAN 是一个很好的分析SQL语句的工具, 它甚至可以在不执行SQL的情况下分析语句. 通过分析, 我们就可以知道ORACLE是怎么样连接表, 使用什么方式扫描表(索引扫描或全表扫描)以及使用到的索引名称。

**22. SQL PLUS的TRACE**

SQL> list SELECT \*FROM dept, empWHERE emp.deptno = dept.deptnoSQL> set autotrace traceonly /\*traceonly 可以不显示执行结果\*/SQL> /rows selected.Execution Plan----------------------------------------------------------SELECT STATEMENT Optimizer=CHOOSE0 NESTED LOOPS1 TABLE ACCESS (FULL) OF 'EMP' 1 TABLE ACCESS (BY INDEX ROWID) OF 'DEPT'3 INDEX (UNIQUE SCAN) OF 'PK\_DEPT' (UNIQUE)

**23. 用索引提高效率**

**（1）特点**

优点：提高效率 主键的唯一性验证

代价：需要空间存储 定期维护

重构索引：

ALTER INDEX <INDEXNAME> REBUILD <TABLESPACENAME>

**（2）Oracle对索引有两种访问模式**

索引唯一扫描 (Index Unique Scan)

索引范围扫描 (Index Range Scan)

**（3）基础表的选择**

基础表(Driving Table)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。

如果你使用的是CBO (COST BASED OPTIMIZER)，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。

如果你用RBO (RULE BASED OPTIMIZER)， 并且所有的连接条件都有索引对应，在这种情况下，基础表就是FROM 子句中列在最后的那个表。

**（4）多个平等的索引**

当SQL语句的执行路径可以使用分布在多个表上的多个索引时，ORACLE会同时使用多个索引并在运行时对它们的记录进行合并，检索出仅对全部索引有效的记录。

在ORACLE选择执行路径时，唯一性索引的等级高于非唯一性索引。然而这个规则只有当WHERE子句中索引列和常量比较才有效。如果索引列和其他表的索引类相比较。这种子句在优化器中的等级是非常低的。

如果不同表中两个相同等级的索引将被引用，FROM子句中表的顺序将决定哪个会被率先使用。FROM子句中最后的表的索引将有最高的优先级。

如果相同表中两个相同等级的索引将被引用，WHERE子句中最先被引用的索引将有最高的优先级。

**（5）等式比较优先于范围比较**

DEPTNO上有一个非唯一性索引，EMP\_CAT也有一个非唯一性索引。

SELECT ENAME FROM EMP WHERE DEPTNO > 20 AND EMP\_CAT = 'A'

这里只有EMP\_CAT索引被用到,然后所有的记录将逐条与DEPTNO条件进行比较. 执行路径如下:

TABLE ACCESS BY ROWID ON EMPINDEX RANGE SCAN ON CAT\_IDX

即使是唯一性索引，如果做范围比较，其优先级也低于非唯一性索引的等式比较。

**（6）不明确的索引等级**

当ORACLE无法判断索引的等级高低差别，优化器将只使用一个索引,它就是在WHERE子句中被列在最前面的。

DEPTNO上有一个非唯一性索引，EMP\_CAT也有一个非唯一性索引。

SELECT ENAME FROM EMPWHERE DEPTNO > 20AND EMP\_CAT > 'A'

这里, ORACLE只用到了DEPT\_NO索引. 执行路径如下:

TABLE ACCESS BY ROWID ON EMPINDEX RANGE SCAN ON DEPT\_IDX

**（7）强制索引失效**

如果两个或以上索引具有相同的等级，你可以强制命令ORACLE优化器使用其中的一个(通过它,检索出的记录数量少) 。

SELECT ENAMEFROM EMPWHERE EMPNO = 7935AND DEPTNO + 0 = 10 /\*DEPTNO上的索引将失效\*/AND EMP\_TYPE || '' = 'A' /\*EMP\_TYPE上的索引将失效\*/

**（8）避免在索引列上使用计算**

WHERE子句中，如果索引列是函数的一部分。优化器将不使用索引而使用全表扫描。

/\*低效SQL\*/SELECT \* FROM DEPTWHERE SAL \* 12 > 25000;

/\*高效SQL\*/SELECT \* FROM DEPTWHERE SAL > 25000/12;

**（9）自动选择索引**

如果表中有两个以上（包括两个）索引，其中有一个唯一性索引，而其他是非唯一性索引。在这种情况下，ORACLE将使用唯一性索引而完全忽略非唯一性索引。

SELECT ENAME FROM EMP WHERE EMPNO = 2326AND DEPTNO = 20;

这里，只有EMPNO上的索引是唯一性的，所以EMPNO索引将用来检索记录。

TABLE ACCESS BY ROWID ON EMPINDEX UNIQUE SCAN ON EMP\_NO\_IDX

**（10）避免在索引列上使用NOT**

通常，我们要避免在索引列上使用NOT，NOT会产生在和在索引列上使用函数相同的影响。当ORACLE遇到NOT，它就会停止使用索引转而执行全表扫描。

/\*低效SQL: (这里，不使用索引)\*/SELECT \* FROM DEPTWHERE NOT DEPT\_CODE = 0

/\*高效SQL: (这里，使用索引)\*/SELECT \* FROM DEPTWHERE DEPT\_CODE > 0

**24. 用 >= 替代 >**

如果DEPTNO上有一个索引

/\*高效SQL\*/SELECT \* FROM EMPWHERE DEPTNO >=4

/\*低效SQL\*/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DEPTNO等于3的记录并且向前扫描到第一个DEPT大于3的记录.

**25. 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。

/\*高效SQL\*/SELECT LOC\_ID , LOC\_DESC , REGIONFROM LOCATIONWHERE LOC\_ID = 10UNION SELECT LOC\_ID , LOC\_DESC , REGIONFROM LOCATIONWHERE REGION = 'MELBOURNE'

/\*低效SQL\*/SELECT LOC\_ID,LOC\_DESC,REGIONFROM LOCATIONWHERE LOC\_ID = 10 OR REGION = 'MELBOURNE'

**26. 用IN替换OR**

/\*低效SQL\*/SELECT \* FROM LOCATIONWHERE LOC\_ID = 10OR LOC\_ID = 20OR LOC\_ID = 30

/\*高效SQL\*/SELECT \* FROM LOCATIONWHERE LOC\_IN IN (10，20，30)

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。

**27. 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。

/\*低效SQL：（索引失效）\*/SELECT \* FROM DEPARTMENTWHERE DEPT\_CODE IS NOT NULL;

/\*高效SQL：（索引有效）\*/SELECT \* FROM DEPARTMENTWHERE DEPT\_CODE >=0;

**28. 总是使用索引的第一个列**

如果索引是建立在多个列上， 只有在它的第一个列(leading column)被where子句引用时， 优化器才会选择使用该索引。

SQL> create index multindex on multiindexusage(inda,indb);

Index created.  
SQL> select \* from multiindexusage where indb = 1;

Execution Plan---------------------------------------------------------- SELECT STATEMENT Optimizer=CHOOSE0 TABLE ACCESS (FULL) OF 'MULTIINDEXUSAGE‘

很明显, 当仅引用索引的第二个列时,优化器使用了全表扫描而忽略了索引。

**29. 使用UNION ALL替代UNION**

当SQL语句需要UNION两个查询结果集合时，这两个结果集合会以UNION-ALL的方式被合并，然后在输出最终结果前进行排序。如果用UNION ALL替代UNION，这样排序就不是必要了，效率就会因此得到提高。

由于UNION ALL的结果没有经过排序，而且不过滤重复的记录，因此是否进行替换需要根据业务需求而定。

**30. 对UNION的优化**

由于UNION会对查询结果进行排序，而且过滤重复记录，因此其执行效率没有UNION ALL高。UNION操作会使用到SORT\_AREA\_SIZE内存块，因此对这块内存的优化也非常重要。

可以使用下面的SQL来查询排序的消耗量 ：

select substr（name，1，25） "Sort Area Name"，substr（value，1，15） "Value"from v$sysstatwhere name like 'sort%'

**31. 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EMP\_TYPE是一个字符类型的索引列.\*/SELECT \*FROM EMPWHERE EMP\_TYPE = 123  
/\*这个语句被ORACLE转换为:\*/SELECT \*FROM EMPWHERE TO\_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。

几点注意：

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。

如果在索引列上面进行了隐式类型转换，在查询的时候将不会用到索引。

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。

为了避免ORACLE对SQL进行隐式的类型转换，最好把类型转换用显式表现出来。

**32. 使用提示（Hints）**

FULL hint 告诉ORACLE使用全表扫描的方式访问指定表。

ROWID hint 告诉ORACLE使用TABLE ACCESS BY ROWID的操作访问表。

CACHE hint 来告诉优化器把查询结果数据保留在SGA中。

INDEX Hint 告诉ORACLE使用基于索引的扫描方式。

**其他的Oracle Hints**

ALL\_ROWS

FIRST\_ROWS

RULE

USE\_NL

USE\_MERGE

USE\_HASH 等等。

这是一个很有技巧性的工作。建议只针对特定的，少数的SQL进行hint的优化。

**33. 几种不能使用索引的WHERE子句**

（1）下面的例子中，‘!=’ 将不使用索引 ，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。

/\*不使用索引\*/SELECT ACCOUNT\_NAMEFROM TRANSACTIONWHERE AMOUNT !=0；

/\*使用索引\*/SELECT ACCOUNT\_NAMEFROM TRANSACTIONWHERE AMOUNT > 0；

（2）下面的例子中，‘||’是字符连接函数。就象其他函数那样，停用了索引。

/\*不使用索引\*/SELECT ACCOUNT\_NAME，AMOUNTFROM TRANSACTIONWHERE ACCOUNT\_NAME||ACCOUNT\_TYPE='AMEXA'；

/\*使用索引\*/SELECT ACCOUNT\_NAME，AMOUNTFROM TRANSACTIONWHERE ACCOUNT\_NAME = 'AMEX'AND ACCOUNT\_TYPE='A'；

（3）下面的例子中，‘+’是数学函数。就象其他数学函数那样，停用了索引。

/\*不使用索引\*/SELECT ACCOUNT\_NAME，AMOUNTFROM TRANSACTIONWHERE AMOUNT + 3000 >5000；

/\*使用索引\*/SELECT ACCOUNT\_NAME，AMOUNTFROM TRANSACTIONWHERE AMOUNT > 2000 ;

（4）下面的例子中，相同的索引列不能互相比较，这将会启用全表扫描。

/\*不使用索引\*/SELECT ACCOUNT\_NAME, AMOUNT FROM TRANSACTION WHERE ACCOUNT\_NAME = NVL(:ACC\_NAME, ACCOUNT\_NAME)

/\*使用索引\*/SELECT ACCOUNT\_NAME，AMOUNTFROM TRANSACTIONWHERE ACCOUNT\_NAME LIKE NVL(:ACC\_NAME, ’%’)

**34. 连接多个扫描**

如果对一个列和一组有限的值进行比较，优化器可能执行多次扫描并对结果进行合并连接。

举例：

SELECT \* FROM LODGING WHERE MANAGER IN ('BILL GATES','KEN MULLER')

优化器可能将它转换成以下形式：

SELECT \* FROM LODGINGWHERE MANAGER = 'BILL GATES'OR MANAGER = 'KEN MULLER'

**35. CBO下使用更具选择性的索引**

基于成本的优化器（CBO，Cost-Based Optimizer）对索引的选择性进行判断来决定索引的使用是否能提高效率。

如果检索数据量超过30%的表中记录数，使用索引将没有显著的效率提高。

在特定情况下，使用索引也许会比全表扫描慢。而通常情况下，使用索引比全表扫描要块几倍乃至几千倍！

**36. 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。

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

**37. 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。

/\*低效SQL\*/SELECT JOB,AVG（SAL）FROM EMP GROUP BY JOBHAVING JOB = 'PRESIDENT''OR JOB = 'MANAGER'

/\*高效SQL\*/SELECT JOB,AVG（SAL）FROM EMP WHERE JOB = 'PRESIDENT'OR JOB = 'MANAGER'GROUP BY JOB

**38. 使用日期**

当使用日期时，需要注意如果有超过5位小数加到日期上，这个日期会进到下一天!

SELECT TO\_DATE（'01-JAN-93'+.99999）FROM DUAL结果：'01-JAN-93 23:59:59'  
SELECT TO\_DATE（'01-JAN-93'+.999999）FROM DUAL结果：'02-JAN-93 00:00:00'

**39. 使用显示游标(CURSORS)**

使用隐式的游标，将会执行两次操作。第一次检索记录，第二次检查TOO MANY ROWS 这个exception。而显式游标不执行第二次操作。

**40. 分离表和索引**

总是将你的表和索引建立在不同的表空间内（TABLESPACES）。

决不要将不属于ORACLE内部系统的对象存放到SYSTEM表空间里。

确保数据表空间和索引表空间置于不同的硬盘上。

好了，关于Oracle SQL优化的内容，这一篇应该满足常规大部分的应用优化要求。就先到这里了。