Oracle9i

数据库系统优化与调整

目　　录

第一部分　ORACLE 系统优化基本知识

第 1 章　ORACLE 结构回顾

为了使读者对本资料所描述的内容有直接的理解，这里从总结的角度出发，给出了深入了解 Oracle8i/9i 的管理所需的准备知识小结，如果读者对基本的概念已经很熟悉，则可以跳过本章。

§1.1 Oracle 数据库结构

主要介绍 Oracle 数据库结构，包括：

- Oracle 数据字典
- 表空间与数据文件
- Oracle 实例(Instance)

§1.1.1 Oracle 数据字典

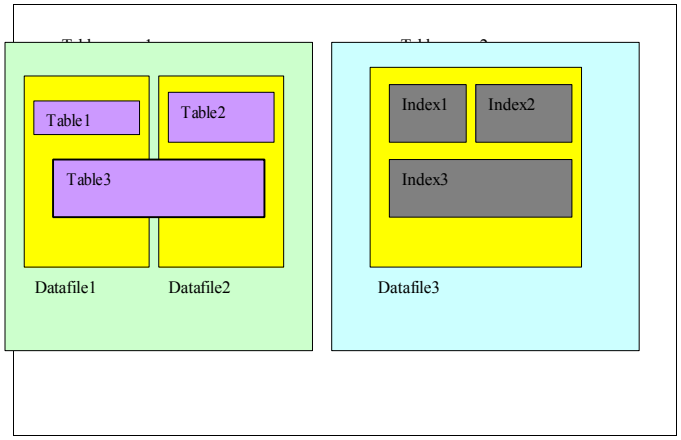
Oracle数据库的重要部分是数据字典。它存放有Oracle数据库所用的有关信息，对用户来说是一组只读的表。数据字典内容包括：

- 数据库中所有模式对象的信息，如表、视图、簇、及索引等。
- 分配多少空间，当前使用了多少空间等。
- 列的默认值。
- 约束信息的完整性。
- Oracle用户的名字。
- 用户及角色被授予的权限。
- 用户访问或使用的审计信息。
- 其它产生的数据库信息。

数据库数据字典是一组表和视图结构。它们存放在SYSTEM表空间中。
数据库数据字典不仅是每个数据库的中心。而且对每个用户也是非常重要的信息。用户可以用SQL语句访问数据库数据字典。

§1.1.2 表空间与数据文件

Oracle 以表空间来存储逻辑数据并以物理数据相连。如图：



数据库被分成一个或多个逻辑部件称作***表空间。***而表空间又被分成称作***段***（segment）的逻辑部件。这些段再细分就叫***扩展（extents）。***Oracle 一般有如下表空间：

- SYSTEM 表空间
- 用户多个表空间
- 工具表空间
- 只读表空间
- 临时表空间
- 回滚段表空间

§1.1.3 Oracle 实例(Instance)

Oracle实例是由一组后台进程和内存结构组成。每个运行的数据库系统都与实例有关。

Oracle 进程

Oracle 有两种类型的进程： 服务器进程和后台进程（server processes and background processes)。

服务器进程

Oracle 服务器进程是处理用户与实例连接的事务。任务是：

- 分析和执行应用所发出的SQL语句。
- 从数据文件读必要的数据到SGA区的共享数据区。
- 返回必要信息给应用。

后台进程

Oracle系统使用一些附加的进程来处理系统的必须的工作。这些进程叫后台进程：

- 数据库写 (DBW0 或DB*W*n)
- 日志写 (LGWR)
- 检测点 (CKPT)
- 系统监视 (SMON)
- 进程监视 (PMON)
- 归档 (AR*C*n)
- 恢复 (RECO)
- 锁(LCK0)
- 工作队列 (SN*P*n)
- 队列监视 (QM*N*n)
- 发布（Dispatcher） (*Dnn*n)
- 服务器 (*Snn*n)

Oracle 的内存结构(SGA)

SGA 结构为：

- 数据高速缓冲区
- 重做日志缓冲区
- 共享池
- 大的共享池(可选)
- 数据字典缓冲区
- 其它数据缓冲区

§1.2 Oracle 文件

§1.2.1 数据文件

- Oracle 的数据文件是用来存放实际数据的物理文件；
- Oracle 数据文件必须对应于一个表空间；
- Oracle 系统安装完成并创建数据库成功后，Oracle 会自动创建几个数据库文件。它们被分配给 SYSTEM、USERS、TEMP 等表空间；
- 用户可以根据需要创建表空间时创建一个或多个数据文件；
- 对已经创建完成的数据文件可以用 ALTER ADTABASE … AUTOEXTEND 、RESIZE 来改变数据文件的大小。
- 数据文件可用下面命令查到：

```
select tablespace_name,file_name ,bytes from dba_data_files;
```

§1.2.2 控制文件

- Oracle 数据库至少有一个控制文件；
- 一般数据库系统安装完成后，自动创建两个以上控制文件；
- 为了安全一般建议创建多个控制文件；
- 控制文件可用下面命令查到：

```
select name,value from v$parameter where name like  'control_files';
```

§1.2.3 重做日志文件

- 重做日志是 Oracle 的日记帐，负责记录所有用户对象或系统变更的信息；
- 安装完成后有多个重做日志文件，它们是几个分为一组，组内的重做日志文件大小要一样；
- 为了使系统性能更好可以在创建多重做日志文件组；
- 重做日志文件可以名下面命令查到：

```
select * from v$logfile;
```

§1.2.4 其它支持文件

除了上面的三类文件外，还有：

- INITsid.ORA 参数文件；
- Sqlnet.ora 文件；
- Tnsnames.ora 文件；

- Listener.ora 文件等。

§1.3 数据块、区间和段

Oracle 系统的数据块(block)和区间(extent)及段（segment）存在一种关系。了解它们的关系对于管理和优化都有好处。

§1.3.1 数据块（data block）

- Oracle 的数据块也叫 Oracle 块；
- Oracle 系统在创建表空间时将数据文件格式化成若干个 Oracle 块；
- 每个 Oracle 块是 Oracle 系统处理的最小单位；
- 块的大小在安装系统时确定，可以选择“自定义安装”来设置大小；
- 块的大小一旦确定下来就不能改动；
- 块的大小可以从 2k 至 64k 不等；
- 块的大小可以用下面命令查到：

```
select name,value from v$parameter where name like 'db_block_size';
```

§1.3.2 区间（extent）

- 分配给对象（如表）的任何连续块叫区间；
- 区间也叫扩展，因为当它用完已经分配的区间后，再有新的记录插入就必须在分配新的区间（即扩展一些块）；
- 区间的大小由 next 决定；
- 一旦区间分配给某个对象（表、索引及簇），则该区间就不能再分配给其它的对象；
- 一个对象所用去多少区间可用下命令查到：

```
select segment_name,tablespace_name,count(*) from dba_extents
 having count(*)>1 group by segment_name,tablespace_name;
```

§1.3.3 段(segment)

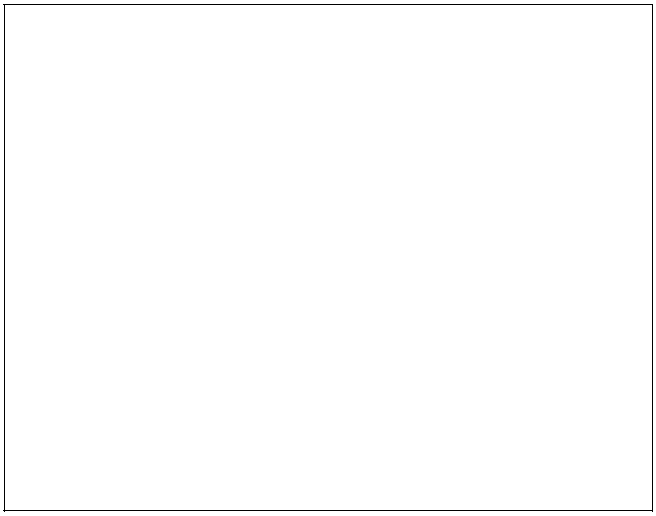
- 段是指占用数据文件空间的通称，或数据库对象使用的空间的集合；
- 段可以有表段、索引段、回滚段、临时段和高速缓存段等；
- 段使用同表空间中的若干 Oracle 块（可以位于不同数据文件中）。

例：

```
CREATETABLE abc ( empno number(4),ename varchar2(20),sal number(9,2))
TABLESPACEuser_data storage(initial 500k next 256k pctincrease 0);
```

１）段被分配＝初始区间＝500k;
２）当开始的 500k 用完后就再分配 256k; 此时段＝500k+256k;

3）如果所分配的区间又用完后，就再分配 256k, …



段（segment）、区间（extent）及块的关系(seg_ext_block)

§1.4 SQL 语句处理

SQL(Structured Query Language)是结构查询语言，它用于操纵和检索 Oracle 数据库的数据。而 SQL 语句分为 DDL(Data Define Language)和 DML(Data Manipulation Language)两类。在本教材里主要讨论 DML 语言。DML 语言包括：SELECT 、INSERT、UPDATE、DELETE、EXPLAIN 及 LOCK TABLE。还要讨论 COMMIT 语句。

§1.4.1 SQL 语句处理顺序

SQL 语句处理分两个或三个阶段，每个语句从用户进程传给服务器进程进行分析然后执行。如果是 select 语句，则还需要将结果返回给用户。

1．分析（Parse）

分析是 SQL 语句处理的第一步。主要进行：

- 检查语法和根据字典来检查表名、列名。
- 确定用户执行语句的权限。
- 为语句确定最优的执行计划。
- 从 SQL 区中找出语句。

2．执行(Execute）

Oracle 执行阶段执行的是被分析过的 语句。对于 UPDATE、DELETE 语句，Oracle 先锁住有关的行。Oracle 还要查找数据是否在数据缓冲区里。如果不在还得从数据文件中将数据读到数据缓冲区里来。

3．检索(Fetch ）

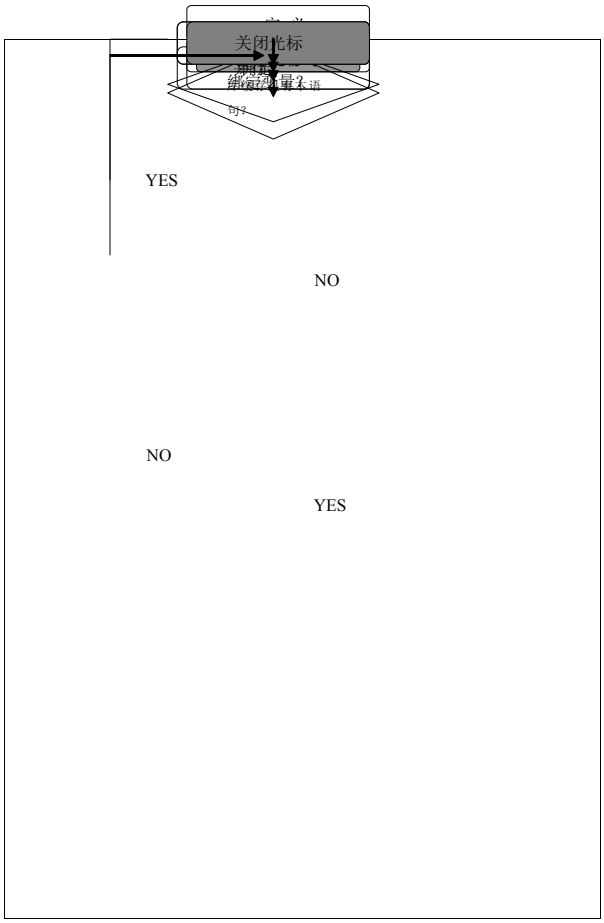
如果是 select 语句，还要进行检索操作。执行结束，将数据返回给用户。

4．Select 语句处理：

一般 select 语句处理要经过下面步骤：
执行顺序是：
1）建立光标。
2）分析语句。
3）定义输出：指定位置，类型，结果集的数据类型。
4）绑定变量：如果查询使用变量的话，Oracle 就要知道变量的值。
5）是否能并行运行。
6）执行查询。
7）检索出数据。
8）关闭光标。

5．DML 语句处理：

一般 INSERT,UPADTE,DELETE 语句处理要经过下面步骤：



执行顺序是：
1)建立光标：Oracle 建立一个隐含的光标。
2)分析语句。
3)绑定变量：如果语句用了变量，Oracle 要知道变量的值。
4)看语句是否能以并行方式运行（如果有多个服务器时）。
5)执行语句。
6)通知用户，语句已执行完毕。

7)关闭光标。

§1.4.2 COMMIT 语句处理顺序

当事务提交时，Oracle 分配一个唯一的顺序号 SCN(System Change Number)给事务。数据库恢复总是基于该 SCN 号来进行处理。SCN 号是记录在控制文件、数据文件、块头及重做日志文件中。

1．COMMIT 处理步骤：

Oracle 在下面情况提交事务：

- 发出一个 COMMIT 语句。
- 执行 DDL 语句时。
- 离开 Oracle 时。

Oracle 处理 COMMIT 的顺序是：

1）服务器为每个 COMMIT 产生一个 SCN。使改变永久化。
2）LGWR 进程将日志缓冲区数据并带有 SCN 一起写到重做日志文件。
3）服务器释放表级和行级锁。
4）用户被提示 COMMIT 完成。
5）服务器使事务已完成。

Oracle 处理 ROLLBACK 的顺序是：

当下面情况发生时执行回滚：

- 发出 ROLLBACK 命令。
- 服务器进程放弃地终止。
- 会话被 DBA 终止。

ROLLBACK 是对数据库的操作进行撤消，步骤有：

1)服务器进程不做任何的改变。
2)服务器释放表级和行级锁。
3) 服务器使事务已完成。

§1.5 共享池

Oracle 的系统全局区 SGA 是由三个部分组成：

- 重做日志缓冲区
- 数据库缓高速冲区
- 共享 SQL 区（包括数据字典 cache 和 Shared SQL Pool ）

SGA 中的共享池由两个部分组成，即共享 SQL 高速缓存（即库高速缓存）和数据字典高速缓存。共享 SQL 高速缓存用于在 SGA 中保存当前一次执行的查询的存储过程和其它 SQL 操作。频繁使用的 SQL 语句都存放在该区内。 而数据字典高速缓存存放系统的数据字典。
共享池高速缓存的使用效率取决于命中率。命重率高表示不需要从硬盘读数据。但是这样的情况是很不现实的。

例：查询执行次数和失效次数：

```
select sum(pins) pins,  sum(reloads) reloads  from  v$librarycache;
```

如果 ratio = (reloads / pins) * 100 大于 1 或更大。就需要加大共享池的大小。

类似地，数据字典高速缓存取决于数据库访问的用户数、权限、数据表、索引等。数据库系统会重复使用相同的数据库对象。如果程序频繁地访问硬盘，就说明数据字典高速缓存过快失效造成。

例：查询用户可以获得 gets(找到对象)次数和 getmisses(高速缓存失效)的次数：

```
select sum(gets) gets , sum(getmisses) getmisses
from  v$rowcache;
```

如果 ratio = (getmisses / gets) * 100 大于 10%，就要考虑加大 SHARED_POOL_SIZE 参数值。

§1.6 块缓存(数据高速缓冲区)

块缓存即数据高速缓冲区。它是 SGA 区的一个主要部分。用于存放从数据文件读入的数据块。它的大小有 INITsid.ORA 文件的 DB_BLOCK_BUFFERS 参数来设置(Oracle9i 参数名称为 DB_CACHE_SIZE)。这个区内容再分为：

- Dirty Buffers 已经改变但还没有存盘的缓冲区。
- Pinned Buffers 当前正在访问的缓冲区。
- Free Buffers 当前尚未使用的缓冲区。

数据高速缓冲区也有命中率的问题。如果需要的数据能在数据高速缓冲区中找到。就叫命中。下面语句查询命中率。

例：查询数据高速缓冲区的命中率

```
select name, value  from  v$sysstat
where  name in ('consistent gets','db block gets','physical reads' ) ;
```

如果 ratio = 1- (physical reads / (db block gets + consistent gets)) 低于 70%。 则应该加大 INITsid.ORA 文件中的 DB_BLOCK_BUFFERS 参数值。

§1.7 数据库写入进程

数据库写入进程（DBWR）的任务是将修改后的数据块写回数据库文件中。在某些操作系统中，Oracle 有两个 BDWR 进程。
当用户向表中增加数据或从表中读数据，都要用到数据高速缓冲区。当对缓冲区的数据进行过改动，就叫脏（dirty），如果多次修改（弄脏多个块）缓冲区数据，空闲的缓冲区就减少。用户就找不到有效的缓冲区来存放查出（读出）的数据。

§1.8 日志写进程

数据库在处理每个事务中，先将所涉及的记录记在日志缓冲区内。日志缓冲区的是按照先进先出原则进行工作的。在 Oracle 系统中，有多个日志文件，日志进程以循环的方式将日志缓冲区内的数据写到日志文件中。
日志写入进程的规则是：

- LGWR 在收到用户进程的提交语句时被唤醒。
- LGWR 每 3 秒被唤醒一次并写重做日志缓冲区。
- 当重做日志缓冲区使用达 1/3 时，LGWR 写重做日志缓冲区。
- 当 DBWR 将数据缓冲区的数据写到数据文件时，LGWR 确保相应的重做日志缓冲区也写入到日志文件中。

§1.9 数据库检查点

当把 SGA 高速缓冲区的数据写回数据文件时就发生了检查点。检查点事件将定期发生以保证数据库和控制文件同步。然而，由于硬盘的 I/O 速度比内存的处理速度慢，所以数据库系统一方面要保持内存数据文件的同步，又要保证这种同步不能过频繁。这样就要在 INITsid.ORA 文件中设置两个参数： LOG_CHECKPOINT_INTERVAL 和

LOG_CHECKPOINT_TIMEOUT。
LOG_CHECKPOINT_INTERVAL 设置块的数量间隔。
LOG_CHECKPOINT_TIMEOUT 设置时间间隔。
如果 DBA 增大重做日志文件，则重做日志交换的频率降低。
如果要强制产生检查点，就要用下面命令来完成：

ALTER system switch logfile;

当 DBA 执行重做日志的管理时，如将日志文件从一个硬盘移到另一个硬盘时，就需要强制检查点发生。

§1.10 归档处理

当数据库环境设置成归档模式时，归档进程（ARCH）就被启动。LGWR 以循环的方式对日志文件进行写。当 LGWR 进程写满一个日志文件并要覆盖写另外的日志文件时，LGWR 通知 ARCH 进程把想将将被覆盖的日志文件拷贝到一个归档日志中。ARCH 就负责将在线的日志文件的副本写到指定的磁带或硬盘上。
要设置数据库为归档方式，可以有两种方法：

ALTER SYSTEM ARCHIVE LOG START;

或

在 INITsid.ORA 文件中设置 LOG_ARCHIVE_START=true 并重启动数据库实例。

§1.11 程序全局区（PGA）

当每个用户与 Oracle 系统连接成功后，Oracle 创建一个叫做程序或私有全局区（Program or Private Gloable Area ）。这个程序全局区一般来说是一个较小的服务器内存区域，程序全局区包含有数据和单进程的（服务器或后台）控制信息。它包含有特定的会话信息。例如，一个客户端的服务器进程使用它的 PGA 来存储该会话的程序变量及包的状态等。

程序全局区是一个非共享的程序可以写入的内存区。当某个 PGA 被分配给每个服务器进程；则该 PGA 对于服务器进程来说是独占的，并且只有 Oracle 代码才进行可读写。

第 2 章　警告日志与跟踪日志

§2.1 警告与日志文件

警告日志是 ORACLE 的一种基本警告机制，其作用是提醒数据库管理员注意数据库的错误、异常环境及一些永久性操作。比如建立表空间，增加数据文件等。每个事件都在影响着数据库的性能和可用性。警告日志主要记录下面信息：

- 重作日志转换
- 检查点（可选）
- 关键性错误
- 数据库的启动与关闭
- 对数据库结构的改变

§2.1.1 经常查看警告日志文件内容

要经常使用编辑器或其它工具对警告日志内容进行阅读。如使用下面命令进行跟踪：
$ tail –f alert_sid.log

§2.1.2 熟悉警告日志的事件或消息

- 经常性提示
- 不常见的提示
- 非关键的错误或警告
- 关键性的错误

对于 ORACLE 错误，可以使用 oerr ora xxxx 来查阅该错误是否严重。如：
oerr ora 1578

1.数据库启动信息

Sarting ORACLE instance (normal)
LICENSE_MAX_SESSION=0
LICENSE_SESSION_WARNING=0
LICENSE_MAX_USER=0
Starting up ORACLE RDBMS Version:8.0.5.1.0.
System parameters with non_default values:

Processes = 200
Shared_pool_size =18000000
。 。 。 。 。 。
Completed: **alter database open.**

2.数据库关闭

Sat Mar 20 07:59:12 1999
Shutting Down instance (immediate)
License hight water mark = 78
。 。 。 。 。 。

Completed:ALTER DATABASE DISMOUNT

3.日志转换

Sun May 23 11:52 2000
Thread 1 advanced to log sequence 1836
Current log# 3 seq# 1836 mem# 0 : /opt/apps/oracle/ORAKRNL/DT08/log03DT.log
Current log# 3 seq# 1836 mem# 1 : /opt/apps/oracle/cdata1/DT08/ ORAKRNL /log03DT.log
Sun May 23 11:54:04 1999
Thread 1 advanced to log sequence 1837
Current log# 1 seq# 1837 mem# 0: /opt/apps/oracle/ORAKRNL/DT08/log03DT.log
Current log# 1 seq# 1836 mem# 1 : /opt/apps/oracle/cdata1/DT08/ ORAKRNL /log03DT.log

4.检查点信息(初始化参数 LOG_CHECKINTS_TO_ALERT)

Thu Jul 1 16:32:11 2000
Beginning log swich checkpoint up to RBA[0x762.2.10]，SCN:0x0000.076683b5
Thread 1 advanced to log sequence 1890
Current log# 1 seq# 1890 mem# 0: /opt/apps/oracle/ORAKRNL/DT08/log03DT08.log
Current log# 1 seq# 1890 mem# 0: /opt/apps/oracle/odata1/DT08/ORAKRNL/log03DT08.log
。 。 。 。 。 。
Completed checkpoint up to RBA [0x763.2.10]，SCN:0x0000.076684d5

5.回滚段生成、联机/脱机和撤销

Thu Jul 1 16:37:46 2000
Create rollback segment rbs7 tablespace rbs storage(initial 1m next 1m optimal 5m maxextents 999)
Thu Jul 1 16:37:46 2000
Completed:create rollback segment rbs7 tablespace rbs storage
。 。 。 。 。 。

6.表空间及数据文件

Thu Jul 1 16:55:27 2000
Create tablespace des2k_data datafile
‘/opt/apps/oracle/odata5/DT08/usr_dbf/des2_data01.dbf’ size 10m;
default storage(initial 16k next 16k pctincrease 0)

thu Jul 1 16:55:31 2000
Completed:create tablespace DES2k_DATAdatafile '/opt/apps/oracle …
Thu Jul 1 16:59:26 2000
Alter tablespace DES2k_DATAoffline
。 。 。 。 。 。

7.控制文件备份确认

Thu Jul 1 17:05:11 2000
Alter database backup controlfile to trace
Completed:alter database backup controlfile to trace

8.系统崩溃启动信息

Thu Jul 13 19:42:34 2000
Alter database open
Beginning crash **recovery** of 1 threads
Thu mar 13 19:44:01 2000
Crash recovery completed successfully
Thu mar 13 19:44:02 2000
Thread 1 advance to lof sequence 1943
Current log# 2 seq# 1943 mem# 0: /opt/apps/oracle/ORAKRNL/DT08/log02DT08.log
Current log# 2 seq# 1943 mem# 1: /opt/apps/oracle/odata9/DT08/ORAKRNL/log02DT08.log
Successful open of redo thread 1.
Thu Mar 13 19:44:04 2000
SMON:enabling cache recovery
SMON:enabling tx recovery
Thu Mar 13 19:44:07 2000
Completed:alter database open

9 介质恢复

Sun Jul 04 21:32:58 2000
Alter database open
Sun Jul 04 21:32:59 2000
Error on file c:\ORANT\RDBMS73\trace\orclDBWR.TRC:
ORA-01157:canot identify data file 2 – file not found
ORA-01110:data file 2: E:\ORANT\ORCL\TOOLS01.DBF'
。 。 。 。 。 。

media Recovery complete

10.重作日志组产生、添加新成员/撤消等

Thu Jul 1 21:54:04 2000
Alter database add logfile group 4('/opt/apps/oracle/ORAKRNL/DT08/redog4f1.log') size 2m
Thu Jul 1 21:54:05 2000
Completed:alter database add logfile group 4 ('/opt/apps/oracle…')
Thu Jul 1 21:59:05 2000
Alter database add logfile member '/opt/apps/oracle/odata5/DT08/ORAKRNL/redog4f2.log' to group 4
Thu Jul 1 22:01:05 2000
Completed : alter database add logfile member '/opt/apps/oracle
。 。 。 。 。 。

11.数据文件和重作日志文件的重命名

Thu Jul 1 22:14:50 2000
Alter database mount
Thu Jul 1 22:14P:55 2000
Successful mount of redo thread 1，with mount id 3686055391
Thu Jul 1 22:14:55 2000
Database mounted in Exclusive Mode.
Completed: alter database mount
Thu Jul 1 22;16:30 2000
Alter database rename file '/opt/apps/oracle/odata2/DT08/sys_dbf/tolls01.dbf ' to
'/opt/apps/oracle/odata2/Dt08/sys_dbf/tools_01.dbf'
Thu Jul 1 22:16:30 2000
。 。 。 。 。 。

12.数据库改为 ARCHIVELOG 模式后的日志记录:

The Jul 1 22:26:45 2000
Alter database mount
The Jul 1 22:26;50 2000
Successful mount of redo thread 1，with mount id 3686048170
The Jul 1 22:26:50 2000
Database mounted in exclusive Mode.
Completed:alter database mount
Thu Jul 1 22:26:59 2000
Alter database archivelog
Completed:alter database archivelog

§2.1.3 分析警告日志错误

关键性、非关键性、普通、异常的错误信息和警告有时回产生一个或多个跟踪文件。这些文件需要 DBA 进行分析，并对主要问题采取相应的措施。下面是一些错误可能原因分析。

(1)由于空间不足而无法扩展段

- 故障现象：用户屏幕和警告日志中出现错误。
- 可能的固故障原因：
  1） 空间确实不足；
  2） 可能空间碎片太多而无法扩展
- 可能解决方案：
  1） 直接设置段的物理参数以减少存储碎片
  2） 添加其它数据文件
  3） 周期性重构段或表空间碎片

下面是缺乏段扩展空间的提示：

Thu Mar 13 13:37:49 1999
OER-1653:Unable to extend **table**/cluster MIKI.MIKI_ACCT by 25600 in tablespace MIKI_DATA
Sun Jan 26;18:29:35 1999
OER-1654:Unable to extend **index** MIKI.MIKI_ACCTS_IDX02 by 3200 in tablespace MIKI_INDX
Sat Mar 29 04:34:22 1999
ORA-1652:unable to extend temp segment by 1280 in tablespace TEMP
Sat Mar 29 05:24:39 1999
ORA-1652:Unable to extend temp segment by 1280 in tablespace TEMP

Mon Aug 10 13:44:52 1999
ORA-1650:unable to extend temp rollback segment RBS by 640 in tablespace RBS1
Failure to extend rollback segment 2 because of 1650 condition
ORA-1650:unable to extend rollback segment RBS by 640 in tablespace RBS1
Failure to extend rollback segment 2 because of 1650 condition

(2)缺乏足够的用户许可证

- 故障现象：
  1) 用户无法登录到数据库
  2) 用户屏幕和/或警告日志出现错误信息
- 可能的故障原因：
  访问的用户或会话过多（检查 V$LICENSE 中允许本站点的最大用户数），其原因可能是用户的数量（或每个用户需要的会话数）随时间不断增长，或者可能是由于用户在获取了所需数据后仍然保持着与数据库的连接。
- 可能解决方案

1)如果确实数量问题，则需要增加用户或会话数，可通过 Oracle 公司获取必需的许可证；

2)如果问题是由于用户作业完成后没有退出而引起，则需要提供必要的指导帮助用户。如果仍无效，则设置某些策略，断开那些超过空闲时间阀值的会话，并将这些策略应用与每个用户。

下面是许可证问题提示：

Mon jan 6 16:54:22 1999
LICENSE_MAX_USERS = 16
Tue Jan 7 09:58:46 1999
LICENSE_MAX_SESSION = 8
LICENSE_SESSION_WARNING = 7
Non-DBA logon denied;current logons equal maximum (8)
Thu Jan 7 10:52:56 1999
License warning limit (7) exceeded
Tue jan 7 11:07:22 1999
Non-DBA logon denied;current logon equal maximum (8)

(3)因等待空间管理资源超时

- 故障现象： 在警告日志和相关跟踪文件组中出现错误信息 ORA-1575;
- 可能的故障原因：
  1） Oracle 本身错误；
  2） 由于空间管理活动使 SMON 过忙。
- 在下面的"数据库会话挂起"

Thu Dec 3 13:13:28 1999
Errors in file /opt/apps/oracle/admin/Dt08/bdump/smon_dt08_11196.trc:
ORA-01575:timeout waiting for space management resource
Thu Dec 3 13:13:34 1999
Errors in file /opt/apps/oracle/admin/DT08/bdump/smon_dt08_11196.trc:
ORA-01575:timeout waiting for space management resource

(4) 无法分配新日志
故障现象：警告日志出现消息

可能故障原因：

1） 联机重作日志过小
2） 联机重作日志组数量不足
3） DBWR 吞吐率不足
4） ARCH 因故障中断（或已启动存档）

故障提示：

Thu Mar 13: 10:29:40 1999
Thread 1 advance to log sequence 1880
Current log# 2 seq# 1880 mem# 0: /opt/apps/oracle/ORAKRNL/DT08/log02DT08.log
Current log# 2 seq# 1880 mem# 1: /opt/apps/oracle/odata9/DT08/ORAKRNL/log02DT08.log

Thu Mar 13 10:33:57 1999
Thread 1 cannot allocate new log，sequence 1881
Checkpoint not complete
Current log# 2 seq# 1880 mem# 0: /opt/apps/oracle/ORAKRNL/DT08/log02DT08.log
Current log# 2 seq# 1880 mem# 1: /opt/apps/oracle/odata9/DT08/ORAKRNL/log02DT08.log

(5)存档器故障

故障现象：
- **数据库被挂起**
- **警告日志出现错误信息**

可能故障原因：
- 存档日志目录已满
- Oracle 本身故障

可能的解决方案：
- 使存档日志的空间可用
- 若怀疑是 Oracle 的 BUG，与 Oracle 公司联系

故障日志：

Sat Dec 12 16:21:07 1999
ORA-00255: error archiving log 2 of thread 1，sequence # 57582
ORA-00312:online log 2 thread 1:'/opt/apps/oracle/odata7/DT08/ORAKRNL/log02DT08.log'
ORA-00312:online log 2 thread 1:'/opt/apps/oracle/odata8/DT08/ORAKRNL/log02DT08.log'
ORA-00272:error writing archive log
ARCH:
ORA-00255:error archiving log 2 of thread 1， sequence #57582
ORA-00312:online log 2 thread 1:'/opt/apps/oracle/odata7/DT08/ORAKRNL/log02DT08.log'
ORA-00312:online log 2 thread 1:'/opt/apps/oracle/odata8/DT08/ORAKRNL/log02DT08.log'
ORA-00272:error writing archiving log
。。。。。。

(6) 数据库中断
- **故障现象：见第 27 章**
- **可能的故障原因：见第 27 章**
- **可能的解决方案：见第 27 章**

故障日志：

Mon Jul 7 23:18:48 1999
Error in file /opt/apps/oracle/admin/DT08/bdump/pmon_DT08_16593.trc:

ORA-00600 : internal error code，argments:[6856]，[0]，[0]，[]，[]，[]，[]
Mon Jul 7 23:18:57 1999
Errors in file /opt/apps/oracle/admin/DT08/bdump/pmon_DT08_16593.trc:
ORA-01578 :ORACLE data block corrupted (file # 10，block # 32379)
ORA-01110:data file 10:'/opt/apps/oracle/odata8/DT08/usr_dbf/miki_data01.dbf'
ORA-00600 : internal error code，argments:[6856]，[0]，[0]，[]，[]，[]，[]
…… ……

(7)应用程序死锁

- **故障现象：**

１）警告日志出现错误信息
２）应用程序事务失败（该事务由 Oracle 执行回滚）

- **可能的故障原因：**

1)应用程序包涉及到非默认锁机制，例如对不同的代码程序随意使用显示表级锁或者非法锁定顺序。这里指 modul_a 先锁定 Row_1/table_1，然后锁定 row_1/table_2，相反 Modul-b 要锁定 row_1/table_2，然后锁定 row-1/table-1，当 Modul-a 与 modul-b 并发执行时，可能导致死锁。
2)Oracle Bug。

- **可能的解决方案：**

１）检测并解决应用程序问题。参见 Oracle 有关如何解决相关应用程序死锁问题文档
２）如果怀疑 Oracle bug，与 Oracle 公司联系

故障日志：

Tue Jun 22 00:02:27 1999
Errors in file /opt/apps/oracle/admin/DT08/udump/ora_DT08_9949.trc:
ORA-00600:internet error code，argments: [4406]，[2306963552]，
[2306961104]，[]，[]，[]，[]
ORA-00060:deadlock detected while waiting for resource

§2.2 对警告日志进行归档

随着时间的流逝和数据库的活动，警告日志文件是不断的增长，因而警告日志文件可能变的很大。有可能影响系统的使用。为了对警告日志文件进行有效的管理，需要定期对警告日志进行归档是非常必要的。

- 需要转存，即将整个文件存到另外的磁盘上；
- 修剪日志，在先存档的情况下可以对日志文件进行修剪，即删除前面部分；

- 保留近期的日志内容，比如保留大约 1000 行的日志信息；
- 在数据库运行或启动后都可以对日志文件进行编辑；
- 建议不要直接删除日志文件，虽然可以直接删除日志文件，但最好不要这样；
- 跟踪文件的删除，可以删除不需要的跟踪文件，跟踪文件是在系统发生异常时随日志文件一起产生，日志文件包括了错误事件的概述，而随之产生的跟踪文件则是记录了错误事件的详细内容。

§2.3 跟踪文件

跟踪文件是 Oracle 在系统异常时有 Oracle 自动创建的文件，它与警告文件一起构成完整的故障信息。警告日志包含错误事件的概述；而随之产生的**跟踪文件**记录了该错误的详细信息。所以我们要想得到错误的解决办法的话，要认真分析跟踪文件的内容。

§2.3.1 跟踪文件的产生

跟踪文件有后台进程的“跟踪书写器（TRWR）”进行写入。Oracle 在运行时有许多后台进程，可以用下面命令查询所有后台进程名字：

```
SQL>select name,  description from v$bgprocess;

NAME                  DESCRIPTION
--------------------- ----------------------------------------------------------
PMON                  process cleanup
. . .                 . . .
TRWR                  trace writer process
DBWR                  db writer  process
ARCH                   Archival
LGWR                   Redo …
. . .  . . .
```

- 后台进程在运行过程中，如果遇到例外就产生跟踪文件信息；
- 可以用 ALTER SESSION SET SQL_TRACE=TRUE 来由前台进程产生会话跟踪文件；
- 在初始化参数文件中加 SQL_TRACE=TRUE 来产生数据库级的跟踪文件；
- 可以在初始化参数文件中加 TIMED_STATISTICS=TRUE 来实现有意义的跟踪文件；
- 后台进程产生的跟踪文件所在目录由 BACKGROUND_DUMP_DEST 参数确定；
- 前台进程产生的跟踪文件所在目录由 USER_DUMP_DEST 参数确定。

§2.3.2 分析跟踪文件的信息

跟踪文件包含下面信息：

- 内存转储信息，有 SGA、PGA 或上层栈内容组成；
- 对调用栈的跟踪信息；
- 错误影响到的数据块。

此外，可以从跟踪文件获得数据文件的块、线程号及字节地址等。

建议用户：

1)先从跟踪文件中数据块地址识别出错误区域；
2)该错误所涉及的数据文件；
3)检查错误。

1.定位跟踪文件所列出的数据块地址：

2. 使用 Oracle 提供的包解释数据块：
使用 Oracle 提供的包 DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE 和 DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK 解释每个数据块的地址。如：

SQL>EXEC DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE(<dba>);
SQL>EXEC DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK(<dba>);

这里<dba>是数据块地址。

3.从数据字典中查数据块：

从数据字典中查出数据块属于范围（哪个文件号和块号）：

SQL>select segment_name, segment_type from dba_segments
Where file_id=<file_number>
And <blk_number> between block_id and block_id + (block –1) ;

§2.3.3 跟踪文件的内容

一般出现例外所记录的跟踪信息可能包含下面主要部分：

1.等待表空间资源超时（ORA-01575 ）：

在警告文件和跟踪文件出现：ORA-01575 错误。可能原因：
- Oracle 本身故障；
- 由于表空间活动使 SMON 过忙；

可能的解决办法：

2.内存或空间泄漏(ORA-0600)：

在警告文件和跟踪文件出现：ORA-0600 错误。可能原因：
- 每个进程所占用的内存不断增加而又没有释放；
- Oracle 本身故障。

可能的解决办法：
- 确认引起问题的部件，需要 Oracle 修补应用；
- 若修补不能解决，则考虑版本降级到某个版本。

3.数据库或会话崩溃

故障现象：
- 例程崩溃；
- 会话在无明显原因下被异常终止。

可能原因：
- Oracle 本身故障；
- 硬件或操作系统故障；
- 连接了单任务的 Oracle 实用程序运行在多任务平台上；
- 用 EVENT 和设置了不正确的参数；
- 用户或管理员的失误；
- 数据库中断；
- 在 Rollback 时进行了不适当的改写，可能恢复出现错误。

可能解决方案：
- 用 ALTER SYSTEM DUMP LOGFILE <log_file_to_do_dump>;将信息转储到文件上；

4.数据块中断

故障现象：
- 在警告日志和相关的跟踪文件中出现 ORA-01578 ; ORA-00600);
- 在 DML 下，错误信息提示在屏幕上。

可能解决办法：

5.数据库或会话挂起

故障现象：
- 数据库出现或挂起症状（无响应），可能　其他例程还在运行。
- 一个多个会话出现卡壳现象。

可能的原因：
- Oracle 本身原因；
- 内部错误；
- 归档目录已满；

- 资源集中的任务可能导致数据库出现冻结；
- 操作系统级修改了 Oracle 进程的优先级（由 nice,renice 命令）；
- 在 Oracle 不支持的环境下创建 Oracle 文件；
- 特地的操作系统错误或硬件错误。

可能的解决方案：

- 关闭机器再重新启动；
- 要求 Oracle 技术支持。

§2.4 关于 NET 的日志与跟踪

Oracle 系统在运行中，如果 NET 系统出现错误，NET 将问题信息写到日志文件和跟踪文件中。日志和跟踪错误信息的过程可以帮助你解决网络方面的问题。

- 对于服务器的监听器，在 UNIX 下，日志文件默认记录在 ORACLE_HOME/network/log 目录下；在 WINDOWS 平台下，日志文件默认记录在 ORACLE_HOME\network\log 目录下；
- 对于客户端，日志和跟踪文件就放在当前的工作目录下。

****
Oracle8/8i NET 名称为 NET8，Oracle9i 名称为 Oracle9i NET Service
****

§2.4.1 日志文件(LOG File)

§2.4.1.1 日志中的错误信息

在Oracle网络产品中遇到的错误都记录在网络相关的日志文件中。这些日志文件为管理员提供了一些附加的信息。要想得到所有的错误信息，需要在客户端或名称服务器上也要记录信息。此外，管理员可以替换或删除日志文件。
监听器上的跟踪文件也可以包括审计，对每个请求连接的客户端进行跟踪等。

§2.4.1.2 日志文件的命名

每个 NET 都有自己的日志文件，下面是默认的命名方法：

日志文件	包含的错误信息
sqlnet.log	客户/服务器
listener.log	监听器
names.log	Oracle名称服务器
cman_ pid.log（UNIX） cman pid.log（NT）	Oracle连接管理器CMGW网关进程
cmadm_ pid.log（UNIX）	Oracle连接管理器CMADMIN管理进程

cmadm pid.log（NT）	

§2.4.1.3 设置日志参数

控制文件的参数，包括日志信息类型、日志信息总量及日志文件所存放的位置。

与。。。相关的日志参数	需要设置的配置文件
客户端（Client）	sqlnet.ora
服务器（Server）	sqlnet.ora
监听器（Listener）	listener.ora
Oracle 名称服务器（Oracle Names Server）	names.ora
Oracle 连接管理器（Oracle Connection Manager processe）	cman.ora

1.sqlnet.ora 参数：

Sqlnet.ora 参数	NET 辅助字段	说明
LOG_DIRECTORY_CLIENT	客户信息日志目录	为日志文件建立目标目录
LOG_FILE_CLIENT	客户信息日志文件	默认名为 sqlnet.log
LOG_DIRECTORY_SERVER	服务器信息日志目录	为服务器日志文件建立目标目录，默认： $ORACLE_HOME/network/log（UNIX） ORACLE_HOME\network\log（NT）
LOG_FILE_SERVER	n/a	设置服务器日志文件名字，默认为： sqlnet.log

2.listener.ora 参数：

listener.ora 参数	NET 辅助字段	说明
LOG_DIRECTORY_*listener_name* LOG_FILE_*listener_name*	日志文件	为日志文件建立目标目录和日志文件名。默认： $ORACLE_HOME/network/log（UNIX） ORACLE_HOME\network\log (NT)

3.names.ora 参数：

names.ora 参数	NET 辅助字段	说明
NAMES.LOG_DIRECTORY	日志目录	为日志文件建立目标目录，默认： $ORACLE_HOME/network/log（UNIX） ORACLE_HOME\network\log（NT）
NAMES.LOG_FILE	日志文件	设置客户端的日志文件名，默认： names.log

4. cman.ora 参数：

cman.ora 参数	说明
LOG_LEVEL	建立日志级别： 0 – 不记录； 1- 基本报告； 2- RULE_LIST 匹配查找报告； 3- 传递块报告； 4- 传递 I/O 次数报告 ● CMGW 网关在 UNIX 下建立 cman_pid.log 日志文件；在 NT 下建立 pid.log 日志文件； ● CMADMIN 管理进程在 UNIX 下建立 cmadm_ pid.log 日志文件；在 NT 下建立 pid.log 日志文件。

§2.4.1.4 在配置文件中设置日志参数

sqlnet.ora、listener.ora 、names.ora 日志参数可以在NET配置助理中进行设置。而cman.ora 文件的参数必须手工设置。

设置参数步骤如下：

1. 启动 Net Assistant:

- 在UNIX, 在$ORACLE_HOME/bin下运行 netasst
- 在 NT, 选择 Start > Programs > Oracle - *HOME_NAME*> Network Administration > Net Assistant.

2.指定日志参数：

日志文件	设置日志参数
sqlnet.log	**1.** 在导航方框, 展开 Local > Profile. **2.** 从右边列表中 选择 General. **3.** 点击 Logging 钮 **4.** 指定设置。
listener.log	**1.** 在导航方框, 展开 Local > Listeners. **2.** 选择一个listener. **3.** 从右边列表中 选择 General **4.** 点击 Logging和 Tracing 钮 **5.** 指定设置。
names.log	**1.** 在导航方框, 展开 Oracle Names Servers. **2.** 选择Oracle Names server. **3.** 从右边列表中 选择 Configure Server. **4.** 点击 Adv. tab. **5.** 指定日志目录和文件名

3. 选择 File > Save Network Configuration.
退出The Net Assistant 应用。

§2.4.1.5 在运行控制实用程序中设置日志参数

可以在运行 Lsnrctl 时进行日志参数的设置。比如：

- 监听器, 可以在LSNRCTL中使用SET LOG_FILE和SET LOG_DIRECTORY 进行设置。
- 对于连接管理器（ Oracle Connection Manager ），可以在 LSNRCTL 中使用 SET LOG_LEVEL 进行设置。
- 对于名称服务器（ Oracle Names server ），可以在 NAMESCTL 中使用 SET LOG_FILE_NAME 进行设置或者通过NET 配置助理来设置：

a. 启动Net Assistant:
- UNIX, 在 $ORACLE_HOME/bin下运行netasst
-在 NT, 选择 Start > Programs > Oracle - *HOME_NAME*> Network Administration > Net Assistant.
b. 用导航器扩展 Oracle Names Servers .
c. 选择 Oracle Names server.
d. 从右边列表中选择 Manage Server.
e.点击Logging
f. 指定日志目录和文件名.
g. 选择 File > Save Network Configuration.

§2.4.1.6 理解监听日志中信息

1.监听器的跟踪信息
包含：

- 客户端的连接请求；
- 启动、停止、状态、加载或服务命令的发送。

2. 监听器跟踪的格式

*Timestamp * Connect Data * [Protocol Info] * Event * [SID | Service] * Return Code*
跟踪属性如下：
- 每个字段以星号隔开(*)
- 协议地址信息和 SID或服务名信息在连接时出现
- 成功连接或者返回0
- 失败返回错误代码

例1：成功地加载请求：

14-SEP-1999 00:29:54 *
(CONNECT_
DATA=(CID=(PROGRAM=)(HOST=dlsun1013)(USER=jdoe))(COMMAND=stop)(ARGUMENTS=64)(SER
VICE=LISTENER)(VERSION=135290880))
* stop * 0

例2：成功地完成请求：

10-AUG-1999 15:28:58 *
(CONNECT_DATA=(service_name=sales.us.acme.com)(CID=(PROGRAM=)(HOST=dlsun1013)
(USER=jdoe)))
* (ADDRESS=(PROTOCOL=tcp)(HOST=144.25.185.246)(PORT=41349)) * establish
* sales.us.acme.com * 0

3. 服务注册事件的信息

服务注册信息由PMON进程提供，包括：

- 运行的实例的服务名；
- 数据库实例名；
- 服务操纵者（发布的或专门的服务器）；
- 发送程序、实例和节点信息。

下面是记录在listener.log 文件中的相关的注册事件：

1.事件说明(Event Description)

- SERVICE_REGISTER 监听器收到实例的注册信息；
- SERVICE_UPDATE 监听器收到特殊实例（如发送程序加信息）更新注册信息；
- SERVICE_DIED 监听器丢失与PMON的连接，所有注册信息被忽略。客户不 能连接到实例直到PMON再次注册为止。

2.服务注册信息格式：

*Timestamp * Event * Instance Name * Return Code*

- 每个字段由星号隔开；
- 在一个实例中出现多个时间是正常的；
- 一个成功的注册返回的是0，表示客户可以连接到实例；
- 失败的过程返回一个错误信息。

例1。在一个成功的SERVICE_REGISTER的事件后，是如何收到客户的请求。但到了SERVICE_DIED还不能收到客户的请求。

10-AUG-1999 15:28:43 * **service_register** * sales * 0

10-AUG-1999 15:28:43 * service_register * sales * 0
10-AUG-1999 15:28:58 *
(CONNECT_DATA=(service_name=sales.us.acme.com)(CID=(PROGRAM=)(HOST=dlsun1013)
(USER=jdoe)))
* (ADDRESS=(PROTOCOL=tcp)(HOST=144.25.185.246)(PORT=41349)) * establish
* sales.us.acme.com * 0
10-AUG-1999 15:38:44 * service_update * sales * 0
10-AUG-1999 15:38:44 * service_update * sales * 0
10-AUG-1999 15:48:45 * service_update * sales * 0
10-AUG-1999 15:48:45 * service_update * sales * 0
10-AUG-1999 15:50:57 *
(CONNECT_DATA=(service_name=sales.us.acme.com)(CID=(PROGRAM=)(HOST=dlsun1013)(U
SER=jdoe)))
* (ADDRESS=(PROTOCOL=tcp)(HOST=144.25.185.246)(PORT=41365)) * establish
* sales.us.acme.com * 0
10-AUG-1999 15:51:26 * **service_died** * sales * 12537
10-AUG-1999 15:51:26 * service_died * sales * 12537
10-AUG-1999 15:52:06 *
(CONNECT_DATA=(service_name=sales.us.acme.com)(CID=(PROGRAM=)(HOST=dlsun1013)(U
SER=jdoe)))
* (ADDRESS=(PROTOCOL=tcp)(HOST=144.25.185.246)(PORT=41406)) * establish
* sales.us.acme.com * 12514
TNS-12514: TNS:listener could not resolve SERVICE_NAME given in connect
descriptor

3.直接传递信息

监听器对MTS调度程序直接记录信息。这些信息的格式为：
*Timestamp * Presentation * Handoff * Error Code*

- 每个字段由星号隔开；
- 一个成功的注册返回的是0；
- 失败返回错误信息。

例2。在日志文件中直接传递：
21-MAY-1999 10:54:55 * oracle.aurora.net.SALESHttp2 * handoff * 0

§2.4.1.7 理解连接管理器信息

Oracle连接管理器（Oracle Connection Manager） CMGW网关进程在UNIX上建立一个日志文件叫cman_ pid.log（NT叫pid.log）。而CMADMIN 管理进程在UNIX上建立一个日志文件叫cmadm_ pid.log。在UNIX下，该文件放在$ORACLE_HOME/network/log;在NT下，该文件放在ORACLE_HOME\network\log。

例3. cman_pid.log

(TIMESTAMP=30-OCT-98 18:03:10)(EVENT=10)(VERSION=8.1.6.0.0)
(TIMESTAMP=30-OCT-98 18:03:10)(EVENT=36)(rule_list=
(rule=(src=spcstn)(dst=x)(srv=x)(act=accept)))
(TIMESTAMP=30-OCT-98 18:03:10)(EVENT=32)(PARAMETER_LIST=(MAXIMUM_
RELAYS=1024)(RELAY_STATISTICS=no)(AUTHENTICATION_LEVEL=0)(LOG_LEVEL=1)(SHOW_TNS_
INFO=no)(ANSWER_TIMEOUT=0)(MAXIMUM_CONNECT_DATA=1024)(USE_ASYNC_
CALL=yes)(TRACING=no)(TRACE_DIRECTORY=default)(MAX_FREELIST_BUFFERS=0))
(TIMESTAMP=30-OCT-98 18:03:10)(EVENT=34)(ADDRESS_LIST=
(ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1610)(QUEUESIZE=32)))
(TIMESTAMP=30-OCT-98 18:03:12)(EVENT=38)(COMMAND=2)
(TIMESTAMP=30-OCT-98
18:03:27)(EVENT=26)(RLYNO=0)(SRC=(ADDRESS=(PROTOCOL=tcp)(HOST=spcstn.us.oracle.c
om)(PORT=34758)))(DST=(ADDRESS=(PROTOCOL=tcp)(HOST=144.25.187.89)(PORT=1581)))
(TIMESTAMP=30-OCT-98 18:03:43)(EVENT=28)(RLYNO=0)(SINCE=30-OCT-98
18:03:27)(STATISTICS=(IN=(BYTES=0)(PACKETS=0)(DCDS=0)(OUT=(BYTES=0)(PACKETS=0)(D
CDS=0)))

例4. cmadm_pid.log

(TIMESTAMP=30-OCT-98 18:03:09)(EVENT=Sent Admin Status to UI)
(TIMESTAMP=30-OCT-98 18:03:10)(EVENT=CMan Registration)

§2.4.2 跟踪文件(Trace File)

一般的 NET跟踪可以分为：

- 客户层；
- 应用层（中间层）；
- 数据库层（后台）。

NET 可以在任何层上进行跟踪。可以支持的层次跟踪有：

- 0 或无跟踪（设置为 OFF）；
- 4 或基本跟踪（可设置为 USER）；
- 10 或管理员级跟踪（也可设置为 ADMIN）；
- 16 或由 Oracle 技术分析员为调试问题而设置的详细跟踪(也可设置为 SUPPORT)。

注意，由于跟踪会引起性能下降，特别是高级的跟踪会消耗 CPU 和 I/O，建议在有问题时在进行跟踪。

§2.4.2.1 跟踪文件的命名：

跟踪文件	包含的错误信息

sqlnet.trc	客户端
svr_ pid.trc	服务器
listener.trc	监听器
names.trc	Oracle名称服务器
cman_ pid.trc 在UNIX cman pid.trc 在NT	Oracle连接管理器CMGW网关进程
cmadm_ pid.trc在UNIX cmadm pid.trc在NT	Oracle连接管理器CMADMIN管理进程

§2.4.2.2 参数设置与初始化文件：

跟踪参数对应的	要设置的文件
客户端	sqlnet.ora
服务器	sqlnet.ora
监听器	listener.ora
Oracle连接管理器进程	cman.ora
Oracle名称服务器	names.ora

1.sqlnet.ora 文件的参数

下面的参数可以在sqlnet.ora文件中进行设置：

sqlnet.ora参数	Net辅助字段	说明
TRACE_LEVEL_CLIENT	客户端信息，跟踪级	共0-16级： 0—OFF 5- USER 6- ADMIN 16- SUPPORT
TRACE_DIRECTORY_CLIENT	客户端信息，跟踪目录	建立跟踪文件的目标目录，默认： UNIX为$ORACLE_HOME/network/trace NT为：ORACLE_HOME\network\trace
TRACE_FILE_CLIENT	客户端信息，跟踪文件	设置客户日志文件名字，默认为： sqlnet.trc
TRACE_UNIQUE_CLIENT	客户端信息，唯一的跟踪文件名	在建立客户文件是否唯一。ON 则将进程标识符加到文件中；OFF则可以覆盖存在的文件。 唯一跟踪文件命名为：sqlnet pid.trc
TRACE_LEVEL_SERVER	服务器信息，跟踪级	共0-16级： 0—OFF 7- USER 8- ADMIN 16- SUPPORT
TRACE_DIRECTORY_SERVER	服务器信息，跟踪目录	建立跟踪文件的目标目录，默认： UNIX为$ORACLE_HOME/network/trace NT为：ORACLE_HOME\network\trace
TRACE_FILE_SERVER	服务器信息，跟踪文件	设置服务器的日志文件名字，默认为： svr_ pid.trc

2. LISTENER.ORA 文件的参数

要设置监听程序的跟踪，需要在 listener.ora 文件中设置：

- TRACE_LEVEL_<listener_name>=(0—16)
- TRACE_DIRECTORY_<listener_name>=/ …/，指定跟踪文件目录
- TRACE_FILE_<listener_name>= /…/ ， 指 定 日 志 文 件 ， 默 认 写 到 $ORACLE-HOME/network/trace 目录中;
- 如果所指定的参数无效，可以用 lsnrctl trace 来进行跟踪参数设置。

从上面看，监听程序除了跟踪外，还包含常规日志文件。

另外就是，跟踪文件是在系统运行时自动进行的，因而随着时间的退役，跟踪文件会不断增加，所以，希望管理员经常对跟踪文件进行管理，包括编辑剪裁掉一些内容。但是不要物理地删除这些文件。

3.names.ora文件的参数设置

- NAMES.TRACE_DIRECTORY 跟 踪 文 件 的 目 标 目 录 。 默 认 为 $ORACLE_HOME/network/trace （ UNIX ）；NT在 ORACLE_HOME\network\trace
- NAMES.TRACE_FILE 跟踪文件 名。默认为 names.trc
- NAMES.TRACE_LEVEL 跟踪级别。见前面。
- NAMES.TRACE_UNIQUE 使跟踪文件唯一。ON为每个会话建立跟踪文件；允许多个文件共存。默认文件名：pid.trc.

4.cman.ora文件的参数设置

- TRACING 是否跟踪Oracle连接管理器：

YES 启用跟踪Oracle连接管理器; 在UNIX下， CMGW网关进程建立的跟踪文件为cman_ pid.trc；在NT下，跟踪文件为pid.trc。在UNIX下，CMADMIN管理进程建立的跟踪文件为 cmadm_ pid.trc；在NT下，CMADMIN管理进程建立的跟踪文件为cmadm pid.trc。

- TRACE_DIRECTORY 建立跟踪文件的目标目录。

在UNIX下，默认目录为 $ORACLE_HOME/network/trace
在NT下，默认目录为ORACLE_HOME\network\trace

5.在配置文件中设置跟踪参数：

sqlnet.ora, listener.ora 和 names.ora 文件的参数可以在NET配置助理来设置。但是cman.ora必须手工设置。

设置步骤如下：

1. 启动Net Assistant:

- UNIX, 在 $ORACLE_HOME/bin. 下运行netasst

- NT,选择 Start > Programs > Oracle - *HOME_NAME*> Network Administration > Net Assistant.

2. 指定跟踪参数:
3. 选择 File > Save Network Configuration.

sqlnet.trc (客户端)
svr_ pid.trc (服务器端)
1. 导航器，扩展 Local > Profile.
2.从右边选择 General.
3. 点击Tracing
4. 选择settings.

listener.trc
1. 导航器，扩展 Local > Listeners.
2. 选择一个监听器
3. 从右边选择 General.
4. 点击T Logging and Tracing.
5. 指定设置:

names.trc
1.导航并扩展Oracle Names Servers
2. 选择一个 Oracle Names server.
3. 从有边窗选择 Configure Server.
4. 点击Adv.
5. 指定跟踪目录及文件

§2.4.2.3 在控制实用程序运行中设置跟踪参数

跟踪参数可以在运行控制程序过程中设置。注意，这样的设置只能在会话期间有效，它并不设置所有的*.ora文件。

- 对于监听器，在 LSNRCTL 中使用 SET TRC_FILE, SET TRC_DIRECTORY 和 SET TRC_LEVEL 来设置监听器;
- 对于名称服务器，可以在 NAMESCTL 中使用 SET TRACE_FILE_NAME 和 SET TRACE_LEVEL来设置名称服务器。

通过NET配置助理进行设置的步骤如下：

1. 启动 Net Assistant:
-在UNIX, 在$ORACLE_HOME/bin 目录下运行netasst
-在 NT, 选择Start > Programs > Oracle - *HOME_NAME*> Network Administration > Net Assistant.

2. 在导航窗内，扩展 Oracle Names Servers.
3. 选择一个 Oracle Names server.
4. 从右边选择Manage Server.
5. 点击Logging
6. 指定跟踪级别 trace level,目录及文件名
7. 选择File > Save Network Configuration.

§2.4.2.4 服务器 / 客户包跟踪

- NET 包跟踪可以在服务器端进行设置；
- NET 包跟踪也可以在连接到服务器上的某个客户机进行设置；
- 在服务器端设置包跟踪，要设置：

1)TRACE_LEVEL_SERVER=跟踪级 （跟踪级=16 才能进行包跟踪,低于 16 只能一般跟踪）
2)TRACE_DIRECTORY_SERVER 设置跟踪子目录

- 在 SALNET.ORA 文件中设置 TRACE_FILE_SERVER 设置跟踪文件；
- 在客户机进行跟踪，要在 SQLNET.ORA 文件上设置：

1)TRACE_LEVEL_CLIENT
2)TRACE_DIRECTORY_CLIENT
3)TRACE_FILE_CLIENT
4)TRACE_UNIQUE_CLIENT 设置每个进程对应的跟踪文件

ORACLE 跟踪文件小结：

1. ORACLE 跟踪文件在 $ORACLE_HOME/... /trace/sidalrt.log
2. SQL*net 跟踪文件在 $ORACLE_HOME /net/trace 目录,而 SQL*Net 跟踪分为两种情况:
   1) 客户端跟踪,在 SQLNET.ORA 文件中应设为:
      trace_level_client=admin
      trace_file_client=cli
      trace_directory_cli8ent=$ORACLE_HOME/NET/TRACE
      trace_unique_client=true

      跟踪文件结果数据写到 cli_<pid>.trc 文件中.

   2) 跟踪服务器端,则在服务器端的 SQLNET.ORA 文件中加上以下参数:
      trace_level_server=admin
      trace_file_server=serv
      trace_directory_server=$ORACLE_HOME/NET8TRACE

      跟踪文件结果数据写到 serv_<pid>.trc 文件中.

§2.4.2.5 评估 NET 跟踪信息

无论是自动的或是设置所引起跟踪文件所记录的信息，都对分析错误有很重要的意义。特别是设置跟踪级别（）为SUPPORT时，可以浏览NET的实践内容，包括发送和接收数据包的顺序。

1.数据包的格式

跟踪文件的每一行以一个程序开始，后面是十六进制数据。下面是Net包关键字的列表和说明：

关键字（Keyword）	包类型（Packet Type）
NSPTCN	Connect
NSPTAC	Accept
NSPTRF	Refuse
NSPTRS	Resend
NSPDA	Data
NSPCNL	Control
NSPTMK	Marker

注：如果使用Oracle的网络产品做过加密的话，则这些数据是不可阅读的。

例子。下面是一个叫“nscon”过程在网络上发送NSPTCN包的情况：

nscon: sending NSPTCN packet

所有的包都以NSP开始：

```
nscon: entry
nscon: doing connect handshake...
nscon: sending NSPTCN packet
nspsend: entry
nspsend: plen=187, type=1
nspsend: 187 bytes to transport
nspsend:packet dump
nspsend:00 BB 00 00 01 00 00 00 |........|
nspsend:01 33 01 2C 0C 01 08 00 |.3.,....|
nspsend:7F FF 7F 08 00 00 00 01 |........|
nspsend:00 99 00 22 00 00 08 00 |..."....|
nspsend:01 01 28 44 45 53 43 52 |..(DESCR|
nspsend:49 50 54 49 4F 4E 3D 28 |IPTION=(|
nspsend:43 4F 4E 4E 45 43 54 5F |CONNECT_|
nspsend:44 41 54 41 3D 28 53 49 |DATA=(SI|
```

```
nspsend:44 3D 61 70 33 34 37 64 |D=ap347d|
nspsend:62 31 29 28 43 49 44 3D |b1)(CID=|
nspsend:28 50 52 4F 47 52 41 4D |(PROGRAM|
nspsend:3D 29 28 48 4F 53 54 3D |=)(HOST=|
nspsend:61 70 32 30 37 73 75 6E |ap207sun|
nspsend:29 28 55 53 45 52 3D 6D |)(USER=m|
nspsend:77 61 72 72 65 6E 29 29 |warren))|
nspsend:29 28 41 44 44 52 45 53 |)(ADDRES|
nspsend:53 5F 4C 49 53 54 3D 28 |S_LIST=(|
nspsend:41 44 44 52 45 53 53 3D |ADDRESS=|
nspsend:28 50 52 4F 54 4F 43 4F |(PROTOCO|
nspsend:4C 3D 74 63 70 29 28 48 |L=tcp)(H|
nspsend:4F 53 54 3D 61 70 33 34 |OST=ap34|
nspsend:37 73 75 6E 29 28 50 4F |7sun)(PO|
nspsend:52 54 3D 31 35 32 31 29 |RT=1521)|
nspsend:29 29 29 00 00 00 00 00 |)))....|
nspsend: normal exit
nscon: exit (0)
```

2.理解相关的错误输出

每个问题的发生都与Net的连接有关，错误代码记录在跟踪文件中。都以<ERROR>开头或以<FATAL>开头。如：

例2. 跟踪文件中错误信息

```
nspsend: entry
nspsend: plen=244, type=6
ntpwr: entry
ntpwr: exit
-<ERROR>- nspsend: transport write error
nspsend: error exit
nserror: entry
-<ERROR>- nserror: nsres: id=0, op=65, ns=12541, ns2=12560; nt[0]=511,
nt[1]=61,nt[2]=0
-<ERROR>- nsopen: unable to open transport
nricdt: Call failed...
nricdt: exit
-<ERROR>- osnqper: error from nricall
-<ERROR>- osnqper: nr err code: 12203
-<ERROR>- osnqper: ns main err code: 12541
-<ERROR>- osnqper: ns (2) err code: 12560
```

-<ERROR>- osnqper: nt main err code: 511
-<ERROR>- osnqper: nt (2) err code: 61
-<ERROR>- osnqper: nt OS err code: 0
osnqme: entry
osnqme: reporting nr (1) error: (12203) as rdbms err (12203)
osnqme: exit
-<ERROR>- onstns: Couldn't connect, returning 12203
nricall: Exiting NRICALL with following termination result -1
nricall: exit
osnqme: entry
osnqme: reporting nr (1) error: (12203) as rdbms err (12203)
osnqme: exit
-<ERROR>- onstns: Couldn't connect, returning 12203
-<ERROR>- osnqper: error from nricall

最有效的办法是找出最近的错误代码，因为许多错误代码都是会话层的错误，关键的错误要放在文件的底部。这些最新的错误才能反映当前系统的情况。
有关错误代码的查找，在UNIX下，可以直接用下面命令进行查找：

oerr tns error_number

§2.4.2.6 使用跟踪助理(Trace Assistant)检查跟踪文件

Net提供了一个叫跟踪助理（Trace Assistant）的工具能帮助用户理解跟踪文件的信息。它可以将跟踪文件的文本转换成易读的段落。特别指出的是：跟踪助理只能运行在跟踪级别是16级的跟踪文件下。
用下面命令来启动跟踪助理：

trcasst [options] filename

选件所们如下：
-o 显示可连接的和两任务信息，-o 可以用于：
n c (连接信息)
n d (详细连接信息)
n u (TTC小结信息)
n t (详细的 TTC信息)
n q (显示SQL命令)

-p 仅Oracle内部使用
-s 显示统计信息
-e 启动错误信息显示
n 0 不做任何事
n 1 (从nserror函数中显示 NS 错误信息)
n 2 (显示错误号)
如果不给出参数，则默认为：-odt -e –s

例。选件为-e0 or -e1 的跟踪文件转换情况：

Trace File

nsc2addr: normal exit Error found. Error Stack follows:
nsopen: entry id: 00000
nsmal: 404 bytes at Operation code: 00065
0x10d5a48 NS Error 1: 12541
nsopen: opening NS Error 2: 12560
transport... NT Generic Error: 00511
-<ERROR>- ntus2err: sd=13, Protocol Error: 00146
op=1, resnt[0]=511, OS Error: 00000
resnt[1]=2, resnt[2]=0
-<ERROR>- nserror: nsres:
id=0, op=65, ns=12541,
ns2=12560; nt[0]=511,
nt[1]=2, nt[2]=0
-<ERROR>- nsopen: unable
to open transport

经过转换后，显示出下面可读的信息：

Error found. Error Stack follows:
id: 00000
Operation code: 00065
NS Error 1: 12541
NS Error 2: 12560
NT Generic Error: 00511
Protocol Error: 00146
OS Error: 00000
NS & NT Errors Translation
12541, 00000, "TNS:no listener"
// *Cause: The connection request could not be completed because the
listener
// is not running.
// *Action: Ensure that the supplied destination address matches one of
// the addresses used by the listener - compare the tnsnames.ora entry
with
// the appropriate listener.ora file (or tnsnav.ora if the connection
is to
// go by way of an Interchange). Start the listener on the remote machine.
/
12560, 00000, "TNS:protocol adapter error"
// *Cause: A generic protocol adapter error occurred.
// *Action: Check addresses used for proper protocol specification. Before
// reporting this error, look at the error stack and check for lower level

// transport errors. For further details, turn on tracing and re-execute the
// operation. Turn off tracing when the operation is complete.
/
00511, 00000, "No listener"
// *Cause: The connect request could not be completed because no
application
// is listening on the address specified, or the application is unable to
// service the connect request in a sufficiently timely manner.
// *Action: Ensure that the supplied destination address matches one of
// the addresses used by the listener - compare the tnsnames.ora entry
with
// appropriate listener.ora file.
// Start the listener on the remote machine.

第 3 章　初始化参数、SQL 脚本文件

Oracle数据库系统的一个重要部分是初始化参数，这些参数对于一个数据库实例来说是必须的，同时也是很重要的。下面给出有关Oracle系统的参数的简要说明。包括：

- 初始化参数文件；
- 在参数文件中指定参数值；
- 阅读参数说明；
- 参数说明。

§3.1 初始化参数文件

初始化参数文件(initsid.ora)是一个包括一系列参数的文本文件。这个文件的参数在安装时以默认的方式创建。比如：

PROCESSES = 100
OPEN_LINKS = 12
GLOBAL_NAMES = TRUE

初始化参数文件的名字与操作系统有关，有的操作可能以大小写混合来取名。较早的版本参数文件的名字是init.ora， 但现在名字的结构一般是INITsid.ORA。这里的sid是数据库实例的名字。比如安装Oracle系统时，命名的数据库实例的名字sid为ora817，则对应的初始化文件名就是 init**ora817**.ora 。

数据库管理员使用初始化参数文件可以进行下面的工作：

- 优化内存结构，如数据缓冲区的大小；
- 设置数据库范围的默认值，如为文本区建立初始分配区的大小；
- 设置数据库限制，如数据库用户的最大数；
- 指定文件名。

有些参数可以通过调整来改善数据库的性能，但有些参数必须在Oracle公司全球支持人员的指导下进行。

Oracle9i 参数存放在 SPFILE(SERVER PARAMETER FILE)文件中。SPFILE 是一个二进制的文件，只能由 Oracle 系统自己进行读写，如果对参数进行过修改，则所修改过的参数自动写到 SPFILE 文件中。

§3.2 在参数文件中指定参数值

本节讲述下面内容：

- 参数文件中的规则控制
- 在参数值使用特别字符
- 改变参数值
- 显示当前参数值
- 参数的使用
- 参数的类型
- 不该在参数文件指定的参数
- 当参数设置不正确怎么办?

§3.2.1 参数文件中的规则控制

下面规则在参数文件中支配着参数的说明：

- 所有的参数都是可选的。对每个参数服务器都有一个默认值。参数可以与操作系统有关；
- 参数包含参数和注释（用 # 进行注释 ）；
- 各个参数可以按任何顺序进行指定；
- 参数文件的大小写与操作系统有关；在一行输入几个参数，用空格分开，如：

PROCESSES = 100 CPU_COUNT = 1 OPEN_CURSORS = 10

- 有些参数，如ROLLBACK_SEGMENTS 允许多个值，输入多个值时要用豆号分开，如：

ROLLBACK_SEGMENTS=(seg1,seg2,seg3,seg4)
或
ROLLBACK_SEGMENTS=seg1 seg2 seg3 seg4

- **多次说明同一参数时只有最后一条有效，如：**

ROLLBACK_SEGMENTS=seg1 seg2
OPEN_CURSORS=20
ROLLBACK_SEGMENTS= seg3 seg4

结果是 seg1,seg2 无效。

- 续行用'\'符号，如：

ROLLBACK_SEGMNETS=(seg1,seg2,\
Seg3,seg4,seg5)

- 调用另外参数文件时，用 IFILE 关键字
- **含有空格的参数要用双引号指定，如：**

NLS_TERRITORY=”CZECH REPUBLIC”

- 使用特殊字符时要用双引号指定。

§3.2.2 在参数值中使用特殊字符

如果参数值包含一个特殊的字符，则必须用**转义符**反斜杠（\）来指定或用双引号来说明，如：

DB_DOMAIN = "JAPAN.ACME#.COM"
或
DB_DOMAIN = JAPAN.ACME\#.COM

下表给出初始化文件中使用的特殊字符的列表：

字符	说　　明
#	注释
(	列表值的开始
)	列表值的结束
"	字符的开始或结束
’	引号字符的开始或结束
=	关键字与值的分隔符
,	元素的分隔符
-	UNIX风格关键字的引导符
\	转义符

引号的用法：

有三种方法：
1）在嵌套字符前后用引号，如：

NLS_DATE_FORMAT = """Today is"" MM/DD/YYYY"

2)交替使用单引号和双引号，如：

NLS_DATE_FORMAT = ’"Today is" MM/DD/YYYY’

3)使用转义符方法：

NLS_DATE_FORMAT = "**\"Today is\"** MM/DD/YYYY"

§3.2.3 修改参数值

可以通过编辑参数文件来改变参数值，也可以在会话中改变参数值。

动态参数：

动态参数可以 用**ALTER SESSION 或 ALTER SYSTEM 命令**在**实例运行期间**进行修改，语法如下：

ALTER SESSION SET parameter_name = value
ALTER SYSTEM SET parameter_name = value [DEFERRED]

下表是可以用 **ALTER SESSION 命令**进行修改的参数：

CURSOR_SHARING
DB_BLOCK_CHECKING
DB_FILE_MULTIBLOCK_READ_COUNT
FAST_START_IO_TARGET
GLOBAL_NAMES
HASH_AREA_SIZE
HASH_JOIN_ENABLED
HASH_MULTIBLOCK_IO_COUNT
LOG_ARCHIVE_DEST_n
LOG_ARCHIVE_DEST_STATE_n
LOG_ARCHIVE_MIN_SUCCEED_DEST
MAX_DUMP_FILE_SIZE
NLS_CALENDAR
NLS_COMP
NLS_CURRENCY
NLS_DATE_FORMAT
NLS_DATE_LANGUAGE
NLS_DUAL_CURRENCY
NLS_ISO_CURRENCY
NLS_LANGUAGE
NLS_NUMERIC_CHARACTERS
NLS_SORT
NLS_TERRITORY
OBJECT_CACHE_MAX_SIZE_PERCENT

OBJECT_CACHE_OPTIMAL_SIZE
OPTIMIZER_INDEX_CACHING
OPTIMIZER_INDEX_COST_ADJ
OPTIMIZER_MAX_PERMUTATIONS
OPTIMIZER_MODE
OPTIMIZER_PERCENT_PARALLEL
PARALLEL_BROADCAST_ENABLED
PARALLEL_INSTANCE_GROUP
PARALLEL_MIN_PERCENT
PARTITION_VIEW_ENABLED
PLSQL_V2_COMPATIBILITY
QUERY_REWRITE_ENABLED
QUERY_REWRITE_INTEGRITY
REMOTE_DEPENDENCIES_MODE
SESSION_CACHED_CURSORS
SORT_AREA_RETAINED_SIZE
SORT_AREA_SIZE
SORT_MULTIBLOCK_READ_COUNT
STAR_TRANSFORMATION_ENABLED
TIMED_STATISTICS

下表是可以用 **ALTER SYSTEM 命令**进行修改的参数：

AQ_TM_PROCESSES
BACKGROUND_DUMP_DEST
CONTROL_FILE_RECORD_KEEP_TIME
CORE_DUMP_DEST
CURSOR_SHARING
DB_BLOCK_CHECKSUM
DB_BLOCK_MAX_DIRTY_TARGET
DB_FILE_MULTIBLOCK_READ_COUNT
FAST_START_IO_TARGET
FAST_START_PARALLEL_ROLLBACK
FIXED_DATE
GC_DEFER_TIME
GLOBAL_NAMES
HASH_MULTIBLOCK_IO_COUNT
HS_AUTOREGISTER
JOB_QUEUE_PROCESSES
LICENSE_MAX_SESSIONS
LICENSE_MAX_USERS
LICENSE_SESSIONS_WARNING
LOG_ARCHIVE_DEST

LOG_ARCHIVE_DEST_n
LOG_ARCHIVE_DEST_STATE_n
LOG_ARCHIVE_DUPLEX_DEST
LOG_ARCHIVE_MAX_PROCESSES
LOG_ARCHIVE_MIN_SUCCEED_DEST
LOG_ARCHIVE_TRACE
LOG_CHECKPOINT_INTERVAL
LOG_CHECKPOINT_TIMEOUT
MAX_DUMP_FILE_SIZE
MTS_DISPATCHERS
MTS_SERVERS
OPTIMIZER_MAX_PERMUTATIONS
PARALLEL_ADAPTIVE_MULTI_USER
PARALLEL_INSTANCE_GROUP
PARALLEL_THREADS_PER_CPU
PLSQL_V2_COMPATIBILITY
QUERY_REWRITE_ENABLED
QUERY_REWRITE_INTEGRITY
REMOTE_DEPENDENCIES_MODE
RESOURCE_LIMIT
RESOURCE_MANAGER_PLAN
STANDBY_ARCHIVE_DEST
TIMED_OS_STATISTICS
TIMED_STATISTICS
USER_DUMP_DEST

下表是可以用 **ALTER SYSTEM . . . DEFERRED** 命令进行修改的参数：

参数名字	说明
BACKUP_TAPE_IO_SLAVES	
DB_BLOCK_CHECKING	
DB_FILE_DIRECT_IO_COUNT	
MAX_DUMP_FILE_SIZE	
OBJECT_CACHE_MAX_SIZE_PERCENT	
OBJECT_CACHE_OPTIMAL_SIZE	
PLSQL_V2_COMPATIBILITY	
SORT_AREA_RETAINED_SIZE	
SORT_AREA_SIZE	
SORT_MULTIBLOCK_READ_COUNT	
TRANSACTION_AUDITING	

§3.2.4 显示当前参数值

可以在SQL下用下面命令来显示当前系统的所有参数名字及其值：

SQL>SHOW PARAMETERS

也可以显示单项的参数值，如：

SQL>SHOW PARAMETERS BLOCK
等价于：
SQL> show parameter db_block

SQL>select name,value from v$parameter;

§3.2.5 参数的使用

初始化参数按照功能可以分成以下几组：

- 设置整个数据库的限制
- 设置用户或进程的限制
- 命名数据库系统需要的文件或目录
- 设置数据库资源的限制
- 影响性能的参数（也叫可变参数）

§3.2.6 参数的类型

共有下面几种类型：

- 起源参数
- 带GC前缀的全局高速缓冲参数
- 操作系统有关的参数
- 可变参数
- 异类服务参数

1. 起源参数(Derived Parameters)

有些参数叫起源参数，意思是这些参数是由另外的参数计算而得。所以这些参数的值都不需

要在参数文件中改变或指定。但如果要改变，则要根据经过计算所得的结果来改变。
典型的起源参数如：
SESSIONS 参数是根据 **PROCESSES 参数**的值计算所得。如果PROCESSES参数改变，则**SESSIONS参数**也跟着改变。

2. 带GC前缀的全局高速缓冲参数

带GC开头的参数是全局高速缓存（Global Cache）的意思，它是在多个实例的并行环境下使用。

3. 操作系统有关的参数

这些参数的值与主机操作系统有关，如 DB_BLOCK_BUFFERS 参数就与主机的内存有关。这个参数值的大小与DB_BLOCK_SIZE参数的大小也有关。

4. 可变参数

这些可变的参数都与系统的性能有关。有些可变参数设置容量限制但不影响性能，例如当设置 OPEN_CURSORS = 10 ，用户试图打开第11个光标就会收到一个错误。另外的参数要影响到性能但又不是绝对的。比如减少**DB_BLOCK_BUFFERS 参数**的大小，可系统工作性能变慢。增加该参数的值可改善系统的性能。

5. 异类服务参数

这些异类服务参数可以用于设置**网关**的参数，如使用DBMS_HS包等。

§3.2.7 不能在参数文件中指定的参数

下面参数是不宜在参数文件中指定的参数，一般也不用指定。它们是：

- 不需要改变的参数，这些参数用于Oracle解决问题所用。
- 起源参数，这些参数由Oracle服务器自动计算。

§3.2.8 当参数指定错误时怎么办？

有些参数有它自己的最低限制，如果低于这些限制则可能Oracle实例不能启动。另一方面，设置过低或过高可能引起性能更坏。如果设置某些参数过低或过高而使系统提示错误，应该调整后重启系统。

§3.3 参数内容说明

一般参数说明包括下面几项：

参数名（parameter_name）
参数类型（Parameter type）：整型，布尔型，字符型等。
语法（Syntax）： 字符串，指定语法。
参数类（Parameter class）：说明参数是动态或静态。如果是动态，则可以用ALTER SYSTEM 或ALTER SESSION 命令进行修改。可以用ALTER SYSTEM 设置或改变整个会话的参数值。而用ALTER SESSION 设置只能覆盖当前会话的参数设置。所以最好要存储当前的参数设置的值。
默认值（Default value）：说明该参数不明确时的值。
值的范围（Range of values）： 说明 该参数的取值范围。
Oracle并行服务器（Oracle Parallel Server）：指定该参数对于多实例的并行环境的设置。

§3.4 DBA 常用参数说明

下面给出管理员常用的参数的说明。其它未提到的参数请参见后面的附录。

§3.4.1 跟踪文件路径（BACKGROUND_DUMP_DEST）

参数类型 ：字符型。
语法：BACKGROUND_DUMP_DEST = {pathname | directory}
参数类 ：动态。范围= ALTER SYSTEM 。
默认值 ：与操作系统有关
值的范围 ： 任何有效的路径。

BACKGROUND_DUMP_DEST 指定后台进程（如LGWR, DBWn 等）产生的跟踪文件的路径。警告文件也记录在该路径下。警告文件的默认名为ALERT_sid.LOG。这些文件增长比较慢，但要求用户要周期性删除这些文件。

§3.4.2 在缓冲区驻留对象（BUFFER_POOL_KEEP）

参数类型: 字符串
语法: BUFFER_POOL_KEEP = {integer |
(BUFFERS: integer [, LRU_LATCHES: integer]) }
这里 *integer*是缓冲区数，和LRU锁存器的数
参数类: 静态
默认值:无

BUFFER_POOL_KEEP 可以使你在DB_BLOCK_BUFFERS 下作为保留缓冲池来驻留对象。你也可以用分配一个LRU的一部分（用DB_BLOCK_LRU_LATCHES ）
可以指定5种格式，比如简单的：
BUFFER_POOL_KEEP = 5

或指定缓冲区的组合项和LRU锁存器，如：

BUFFER_POOL_KEEP = (BUFFERS: 400 [, LRU_LATCHES:3])

§3.4.3 版本兼容（COMPATIBLE）

参数类型: 字符串
语法: COMPATIBLE = release_number
参数类: 静态
默认值: 8.0.0
值的范围：默认为当前发行的版本。
并行服务器: 多个实例必须有相同的版本。

§3.4.4 控制文件路径（CONTROL_FILES ）

参数类型: 字符串
语法: CONTROL_FILES = filename [, filename [...]]
参数类: 静态
默认值: 操作系统有关
值的范围：1到8个文件名
并行服务器: 多个实例必须有相同的版本。
每个有一个控制文件，它包括一些数据库的实体（如名字，建立的邮戳、数据文件和日志文件的位置等）。CONTROL_FILES指定一个或多个控制文件名（用豆号来隔开）。

§3.4.5 CPU 个数（CPU_COUNT）

参数类型: 整数
参数类: 静态
默认值: 有Oracle系统自动设置
值的范围：0 到 n

一般由Oracle 安装程序自动设置。CPU_COUNT指定CPU的个数。Oracle使用此参数设置LOG_SIMULTANEOUS_COPIES的默认值。单个CPU时CPU_COUNT为1。根据经验，你可以修改SIMULTANEOUS_COPIES 为CPU个数的两倍。

§3.4.6 数据缓冲区块数（DB_BLOCK_BUFFERS）

参数类型: 整数
参数类: 静态
默认值: 48MB/块大小
值的范围：4 到 操作系统允许的最大
并行服务器：多个实例可以有不同的值。
DB_BLOCK_BUFFERS 指定高速缓冲区的数目。由几个参数一起构成SGA的大小。
缓冲区的实际的大小与DB_BLOCK_SIZE参数有关。有效地使用这个参数可以改善I/O的性能。这个缓冲区的保留（"keep"）和再利用（"recycle"）是这个缓冲区变过来的。另外，缓冲区中的每个LRU列表至少包含50个缓冲区。因此，你必须指定合适的BUFFER_POOL_KEEP 和 BUFFER_POOL_RECYCLE，确保BLOCK_BUFFERS符合下面公式:

DB_BLOCK_BUFFERS > #_buffers_in_"keep"_buffer_pool
+ #_BUFFERS_IN_"recycle"_pool
+ 50*(DB_BLOCK_LRU_LATCHES
- #_lru_latches_in_"keep"_buffer_pool
- #_lru_latches_in_"recycle"_buffer_pool)
如果你没有在BUFFER_POOL_KEEP和BUFFER_POOL_中指定任何LRU，则LRU默认为1。

§3.4.7 数据块大小（DB_BLOCK_SIZE）

参数类型: 整数
参数类: 静态
默认值: 与操作系统有关
值的范围：2048到32768
并行服务器：必须设置所有的实例有相同的数据块。

§3.4.8 读数据块数（DB_FILE_MULTIBLOCK_READ_COUNT）

参数类型: 整数
参数类: 动态，范围= ALTER SYSTEM, ALTER SESSION.
默认值: 8
值的范围：与操作系统有关

在扫描表时一次读的最小的数据块数目。在OLTP和批处理事务中，此参数可以是4到16。对于DSS类型应用，可以设置高些。

§3.4.9 数据文件的数目（DB_FILES）

参数类型: 整数
参数类: 静态
默认值: 与操作系统有关
值的范围： 当前数据库的数据文件的数目
最大值： 由 **MAXDATAFILES**指定。

§3.4.10 全局数据库名（GLOBAL_NAMES）

参数类型: Boolean
参数类: 动态，范围 = ALTER SESSION, ALTER SYSTEM.
默认值: TRUE
值的范围： TRUE | FALSE
用于知单数据库全局名称。如果是分布环境，要将此参数设置为TRUE。

§3.4.11 数据库实例名（INSTANCE_NAME）

参数类型: 字符型
语法： INSTANCE_NAME = instance_id
参数类: 静态
默认值: 实例的SID
指定数据库实例的名字，在安装完成后此参数已经设置好，用户一般不用设置此参数。

§3.4.12 许可的最大会话数（LICENSE_MAX_SESSIONS）

参数类型: 整数型
参数类: 动态，范围= **ALTER SYSTEM**
默认值: 0 到 许可的数
并行服务器: 多个实例可以有不同的值，但是所有实例的会话数之和应该等于所有数据库允许的会话总数。当会话达到最大值时，只有具有RESTRICTED_SESSION权限的用户才能进行登录到数据库。而一般的用户回得到提示信息。0 值表示当前的会话不受限制。如果 设置为非0的话，还要参考 LICENSE_SESSIONS_WARNING参数。
如果不希望使能使用许可和用户许可，可以设置 LICENSE_MAX_SESSIONS 或 LICENSE_MAX_USERS 任何一个为0即可。

§3.4.13 许可的最大用户数（LICENSE_MAX_USERS）

参数类型: 整数型

参数类: 动态，范围= ALTER SYSTEM
默认值: 0
值范围: 0 到 用户许可数
并行服务器: 多个实例有相同的值。如果指定了不同的值，则第1个实例的值优先。
LICENSE_MAX_USERS 指定数据库可以建立的用户数目，如果达到这个值，将不能再创建更多的用户。当然你可以加大这个值。

*注意，**不要同时使能会话（SESSION）和用户（USERS）两个参数。你可以设置两个参数之一为0.***

§3.4.14 许可的会话警告（LICENSE_SESSIONS_WARNING）

参数类型: 整数型
参数类: 动态，范围= ALTER SYSTEM
默认值: 0
值范围: 0 到LICENSE_MAX_SESSIONS
并行服务器: 多个实例可以有不同的值。LICENSE_SESSIONS_WARNING指定了当前用户会话的警告限。当达到这个值时，另外的用户也能登录。但Oracle会在警告文件中些一条信息。具有RESTRICTED SESSION权限的用户可以连接到数据库而且收到一个警告信息。
如果此参数设置为 0，则使用达到限制时没有警告。如果设置为非 0，则还要参考 LICENSE_MAX_SESSIONS。

§3.4.15 归档文件目标路径(LOG_ARCHIVE_DEST)

参数类型：字符
语法: LOG_ARCHIVE_DEST = filespec
参数类: 动态，范围 = ALTER SYSTEM
默认值: 无
值的范围: 正确的定义

§3.4.16 归档文件目标路径(LOG_ARCHIVE_DEST_ n)

参数类型：字符
语法: LOG_ARCHIVE_DEST_[1 | 2 | 3 | 4 | 5] =
"null_string" |
((SERVICE=tnsnames_service
| LOCATION=local_pathname)
[MANDATORY | OPTIONAL]
[REOPEN[=integer]])
参数类: 动态，范围 = ALTER SYSTEM, ALTER SESSION.
默认值: 无
值的范围: 正确的定义

企业版可以设置LOG_ARCHIVE_DEST_ n，标准版本只能设置LOG_ARCHIVE_DEST。
LOG_ARCHIVE_DEST_n
这里n = 1,2,3,4,5 。表示可以有5个归档目标目录。

§3.4.17 日志缓冲区大小(LOG_BUFFER)

参数类型: 整数型
参数类: 静态
默认值: 操作系统指定，最大：500K or 128K * CPU_COUNT
值范围: 操作系统有关

日志缓冲区的大小。一般如果系统较忙，可以设置比65536或更大些。

§3.4.18 检查点块数(LOG_CHECKPOINT_INTERVAL)

参数类型: 整数型
参数类: 动态， 范围= ALTER SYSTEM
默认值: 与操作系统有关
值范围: 无限制
并行服务器: 多个实例可以有不同的值。LICENSE_SESSIONS_WARNING指定检查点从最后一次写到日志文件后的时间间隔。它是操作系统物理块数，不是数据库块。
当忽略此参数时，当从一个日志切换到另外一个日志文件时总是出现检查点。频繁出现检查点会影响系统的性能。

注:

- *设置 LOG_CHECKPOINT_INTERVAL为0 与设置成无限大是一样的效果并且引起参数被忽略。只有非0才有意义。*
- *设置e FAST_START_IO_TARGET或LOG_CHECKPOINT_TIMEOUT时，恢复I/O可能受限制。*

§3.4.19 检查点间隔(LOG_CHECKPOINT_TIMEOUT)

参数类型: 整数型
参数类: 动态， 范围= ALTER SYSTEM
默认值: Oracle8i为900秒；企业版为1800秒。
值范围: 0到无限制
并行服务器: 多个实例可以有不同的值。LOG_CHECKPOINT_TIMEOUT指定检查点从最后一次写到日志文件后（日志尾部）的间隔(秒)。建议不要设置为0 。

§3.4.20 对大卸出文件大小(MAX_DUMP_FILE_SIZE)

参数类型: 字符型
语法: MAX_DUMP_FILE_SIZE = {integer | UNLIMITED}
参数类: 静态， ALTER SYSTEM [... DEFERRED], ALTER SESSION.
默认值: 无限制
值范围: 0到无限制

MAX_DUMP_FILE_SIZE指定跟踪文件的最大值（不包括警告文件）。

§3.4.21 对大回滚段数(MAX_ROLLBACK_SEGMENTS)

参数类型: 整数型
参数类: 动态
默认值: MAX(30, TRANSACTIONS/TRANSACTIONS_PER_ROLLBACK_SEGMENT)
值范围: 0到65535

MAX_ROLLBACK_SEGMENTS 指定回滚段在SGA中的最大联机数。

§3.4.22 打开的光标数(OPEN_CURSORS)

参数类型: 整数型
参数类: 静态
默认值: 50
值范围: 0到4294967295 (4 GB -1)

OPEN_CURSORS 指定 一次会话能打开光标的最大数 (以处理私有的 SQL区)，此参数也限制PL/SQL光标数。

§3.4.23 优化方式(OPTIMIZER_MODE)

参数类型: 字符型
语法: OPTIMIZER_MODE ={RULE | CHOOSE | FIRST_ROWS | ALL_ROWS}
参数类:动态，范围=ALTER SESSION
默认值: 选择

OPTIMIZER_MODE 指定SQL语句的优化目标。

- RULE 指定基于规则的优化，除非在查询中专门指定。
- CHOOSE指定基于代价的SQL优化，如果数据字典有统计书记就按照代价，否则按照规则。
- FIRST_ROWS 引起优化器使用代价优化。
- ALL_ROWS 引起优化器使用代价执行计划。

§3.4.24 进程数(PROCESSES)

参数类型: 整数型
参数类: 静态
默认值: 来源PARALLEL_MAX_SERVERS
值范围: 0到操作系统许可数
并行服务器: 多个实例可以有不同的值。
PROCESSES 指定了操作 系统用户能连接到Oracle的最大数。这个参数也涉及后台进程，如锁、作业进程、并行执行进程等。
SESSIONS 和TRANSACTIONS的默认值来源于这个参数。

§3.4.25 回滚段名称(ROLLBACK_SEGMENTS)

参数类型: 字符型
语法: ROLLBACK_SEGMENTS =(segment_name [, segment_name] ...)
参数类: 静态
默认值: 无
值范围: 由DBA_ROLLBACK_SEGS能列出的回滚段名字（除SYSTEM外）。
并行服务器: 多个实例可以有不同的值。

§3.4.26 服务名(SERVICE_NAMES)

参数类型: 字符型
语法: SERVICE_NAMES =db_service_name [, db_service_name [,...]]
参数类: 静态
默认值: DB_NAME.DB_DOMAIN
值范围: 任何用豆号隔开的字符名字

§3.4.27 会话的数(SESSIONS)

参数类型: 整数型
参数类: 静态
默认值: 来源数据： 1.1 * PROCESSES + 5
值范围: 1 到 2的31 次方

SESSIONS 有系统建立的最大会话数。用户不要设置此参数。

§3.4.28 共享池大小(SHARED_POOL_SIZE)

参数类型: 字符型
语法: SHARED_POOL_SIZE = integer [K | M]
参数类: 静态
默认值: 64位为64MB；否则16MB
值范围: 300KB到操作系统许可数

SHARED_POOL_SIZE以字节指定共享池大小。共享池包括光标、存储过程、控制结构及其它结构。PARALLEL_AUTOMATIC_TUNING = FALSE，Oracle也允许并行从共享池执行信息。较大的值可以改善性能。较小的值可以节约内存。

§3.4.29 分类区的大小(SORT_AREA_SIZE)

参数类型: 整数型
参数类: 动态 ， 范围= ALTER SESSION, ALTER SYSTEM ... DEFERRED
默认值: 操作系统有关
值范围: 最小也是6个数据库块。

SORT_AREA_SIZE 指定Oraclee用于分类的内存数（字节）。排序完成Oracle就释放这些内存。

§3.4.30 用户卸出文件的路径(USER_DUMP_DEST)

参数类型: 字符型
语法: USER_DUMP_DEST = {pathname | directory}
参数类: 动态 ， 范围= ALTER SYSTEM
默认值: 操作系统有关
值范围: 任何有效的路径。

§3.5 SQL 脚本文件

Oracle系统在安装完成后，在$ORACLE_HOME/rdbms/admin **目录**下保存了许多的脚本文件，这些脚文件有一部分已经在执行安装的过程中被运行过，有的未被运行过。那么Oracle系统将这些文件保留在该目录下有什么用呢。其实该目录下的许多脚本是留给用户来处理的。也就是说用户可以根据自己的需要来启动运行某些脚本。比如，如果你需要进行系统的**审计**的话，管理员需要登录到sys帐户来运行 **cataudit.sql** 脚本进行视图的建立，相反，可以用catnoaud.sql.脚来删除由cataudit.sql 脚本所创建的视图。为此我们专门给出脚本的文件名和它们的功能说明。供有兴趣的DBA参考。

§3.5.1 建立数据字典的脚本

CATALOG.SQL
创建所有数据库数据字典和公共同义词，并授权public 可访问这些同义词。

CATPROC.SQL
PL/SQL 运行所需的所有脚本。

§3.5.2 建立附加的数据字典

脚本名称	用于	运行帐户	说明
CATBLOCK.SQL	性能管理	SYS	建立动态显示视图
CATEXP7.SQL	导出数据到Oracle7	SYS	创建从Oracle8到Oracle7所需的字典视图。
CATHS.SQL	异种服务器	SYS	安装异种服务器所需的包
CATIO.SQL	性能管理	SYS	允许I/O跟踪表-表处理
CATOCTK.SQL	安全	SYS	创建Oracle密码工具包
CATPARR.SQL	Oracle并行服务器	SYS或SYSDBA	创建并行服务器数据字典视图
CATQUEUE.SQL	高级查询	SYS—需证实	为高级查询创建字典对象
CATREP.SQL	高级复制	SYS	使能数据库复制
CATRMAN.SQL	恢复管理器	RMAN 或 使 用 GRANT_RECOVERY_CATALOG_OWNER的用户	创建恢复管理器表和视图，用于建立备份、恢复、外部恢复目录等。
DBMSIOTC.SQL	存储管理	任何用户	在索引结构表中分析连接行
DBMSOTRC.SQL	性能管理	SYS或SYSDBA	有效或禁止Oracle的跟踪输出
DBMSPOOL.SQL	性能管理	SYS或SYSDBA	使DBA锁PL/SQL和SQL语句包有效，并使触发器进入共享池
USERLOCK.SQL	协调锁控制	Sys或sysdba	为用户命名的锁在顺序应用行动进行辅助
UTLBSTAT.SQL 和 UTLESTAT.SQL	性能监视	sys	分别启动和停止性能调整统计数据的收集
UTLCHN1.SQL	存储管理	任何用户	Oracle8i 下，为 analyze 命令带 CHAINED ROWS 选项分析连接行的输出创建存储表。可以处理物理和逻辑rowid
UTLCONST.SQL	2000年	任何用户	提供一个函数可在2000年列上进行检查（CHECK）限制
UTLDTREE.SQL	元数据管理	任何用户	创建表或视图以显示对象间的关系
UTLEXPT1.SQL	约束	任何用户	Oracle8i下，创建默认EXCEPTIONS表，用于存储约束所产生的数据。可以处理物理和逻辑rowid
UTLHTTP.SQL	Web访问	Sys或sysdba	借助HTTP协议从internet或intranet的web服务器取回数据
UTLIP.SQL	PL/SQL	SYS	用于移植、升级和降级操作。
UTLIRP.SQL	PL/SQL	SYS	用于从32位到64位的处理。在新环境下重新编译现有的脚本。
UTLLOCKT.SQL	**性能监视**	**Sys或sysdba**	**以曲线树结构显示锁等待。**
UTLPG.SQL	数据转换	Sys或sysdba	提供一个转换IBM/370 和COBOOL II的包
UTLPWDMG.SQL	安全	Sys或sysdba	创建用于复杂口令验证的PL/SQL函数。设置默认口令资源文件（profile）参数和口令管理功能。
UTLRP.SQL	PL/SQL	SYS	重新编译那些无效的PL/SQL模块。
UTLSAMPL.SQL	例子	Sys 或 DBA 角色	创建样例表，如EMP,DEPT等.

		的用户	
UTLSCLN.SQL	高级复制	任何用户	从其它快照点拷贝一个快照
UTLTKPROF.SQL	性能管理	SYS	建立TKPROFER角色以使非DBA用户能运行TKPROF实用程序。
UTLVALID.SQL	分区表	任何用户	建立ANALYZE TABLE ...VALIDATE STRUCTURE 输出所需的表
UTLXPLAN.SQL	性能管理	任何用户	建立PLAN_TABLE表，用于处理由EXPLAIN PLAN命令的输出结果。
脚本名称	**用于**	**运行帐户**	**说明**

§3.5.3 带“NO”的脚本

Oracle提供相应的带“NO”的SQL脚本，可以用它们来完成从数据字典中删除信息。

脚本名称	用于	运行帐户	说明
CATNOADT.SQL	对象	SYS	从数据字典中删除有关的视图和同义词
CATNOAUD.SQL	安全	SYS	删除审计元数据的视图和同义词
CATNOHS.SQL	异类服务器	SYS	移掉异类服务器的数据字典
CATNOPRT.SQL	分区	SYS	在元数据上删除有关分区表和索引的视图和同义词
CATNOQUEUE.SQL	高级队列	SYS	移掉高级队列数据字典。
CATNORMN.SQL	恢复管理器	进行恢复的人	移掉恢复目录模式
CATNOSVM.SQL	服务管理器	SYS	移掉Oracle7服务管理器视图和同义词
CATNSNMP.SQL	分布管理	SYS	删除DBSNMP 用户和SNMPAGENT 角色

§3.5.4 移植的脚本

脚本名称	用于	运行帐户	说明
DROPCAT6.SQL	移掉遗留的元数据	SYS	删除Oracle6数据字典目录视图
DROPCAT5.SQL	移掉遗留的元数据	SYS	删除Oracle5数据字典目录视图
R070304.SQL	复制	SYS	执行一个后CATREP.SQL复制升级
RM804812.SQL	恢复管理器	恢复目录表的用户	从8.0.4 or 8.0.5 to 8.1.3升级恢复目录
U703040.SQL	从Oracle7升级	SYS或SYSDBA	建立新的Oracle8u元数据

§3.5.5 JAVA脚本

在Oracle8i JServer平台上，需要下面的JAVA脚本：

脚本名称	说明
INITJVM.SQL	安装Java类库时初始话Jserver，并指定Java类库进入Oracle服务器及初始化Java相关的包和设置。
RMJVM.SQL	删除所有JServer的元素
UTLJAVARM.SQL	当升级8.1.4 时需要。见前面的升级部分。

第 5 章　性能优化基础知识

§5.1 理解 ORACLE 性能优化

- 响应时间与吞感吐量的折衷
- 临界资源
- 过度请求的影响
- 调整以解决问题

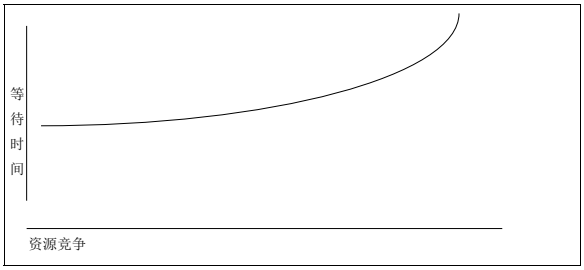
§5.1.1 响应时间与吞吐量的折衷

根据应用类型的不同，性能优化的目标不同：
在线事务处理（OLTP=Online Transaction Processing）应用程序把吞吐量定义为性能指标；
决策支持系统（DSS=Decision Support System）把响应时间定义为性能指标。

响应时间
响应时间=服务时间+等待时间
如任务 1 运行不许任何等待；任务 2 运行等待任务 1 完成后才能运行；任务 3 又必须等待任 1 和任务 2 完成后才能运行，。。。

系统吞吐量
系统吞吐量指在给定的时间内所完成的工作量。有以下两种技术：
- 以相同的资源来完成更多的工作（减少服务时间）；
- 通过减少整个响应时间来更快完成工作。

等待时间
当竞争增强的时候，某个任务的服务时间也许保持不变，但它的等待时间将增长。

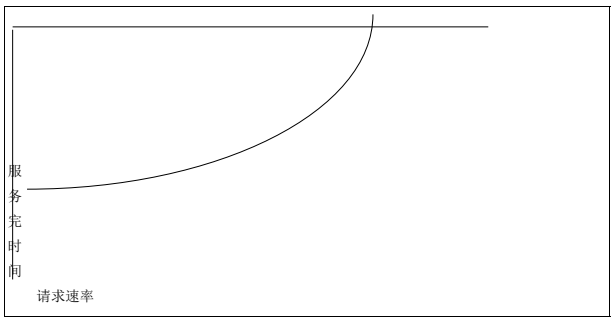


当对资源的竞争增强时，等待时间增长

§5.1.2 临界资源

诸如 CPU、内存、I/O 容量、网络带宽等资源，都是减少时间的关键因素。性能好坏取决于以下因素：

- 可用资源的数量
- 需要该资源的客户方的数目
- 客户方等待资源所消耗的时间
- 客户保持资源的时间长短



服务完成时间与请求速率的关系

随着请求单元的增加，服务时间也增加。为了处理这种情形，用户可以选择：

- 通过限制请求的速率，从而维护可接受响应时间
- 还可通过增加资源数目，如 CPU 和硬盘

§5.1.3 过度请求的影响

过度请求将会引起下面现象：

- 响应时间增长
- 降低吞吐量

§5.1.4 调整以解决问题

下面是可以解决的性能问题：

调整单元的消费量

通过使事务在执行时占用更少的资源，或减少服务时间；

调整功能请求

通过重新规划或重新分配工作；

调整容量

通过增加或重新分配资源。

§5.2 优化的执行者

在优化过程中，系统涉及到下面角色：

- **商业高级管理人员:**负责制定并重新考察商业规划和流程，从而为应用设计提供一种清晰而适当的模型。
- **应用设计人员：**必须饶过潜在的系统瓶颈进行设计，还应当与系统设计人员进行交流，从而得到每个人都可以理解应用模型。
- **应用开发人员：**必须与其所选择的实现策略进行充分的交互，使得在进行语句优化的时候，可以顺利地较快确定模块和 SQL 语句。
- **数据库管理员：**必须仔细地监视系统的活动，并将其归档，以此来识别和修正异常的系统性能。

§5.3 设置性能目标

无论用户是进行系统设计，还是进行系统维护，当需要进行优化时，都应当设定特定性能的目标。优化工作通常是由一系列的“折衷”所组成。一旦用户识别出瓶颈问题所在，那么就可能需要牺牲其他系统资源来获得预期的效果。例如 I/O 存在问题，就需购买更多的内存和硬盘。

第 6 章　系统优化方法

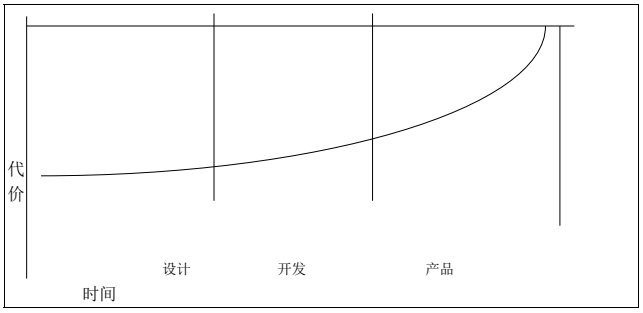
根据应用系统的类型，如 OLTP 或 DSS 来规划系统的优化方法。是性能优化的关键。

§6.1 何时优化效率最高

为了达到最佳的效果，优化工作应当从设计阶段进行，而不是在系统实施后进行。下面是各个阶段优化的代价。

§6.1.1 系统设计阶段和开发阶段的优化

最有效的优化方法是 proactive 方法，从下图可看出，进行优化的最有效阶段是是设计阶段，用户能以最低的代价获得最大的效益。



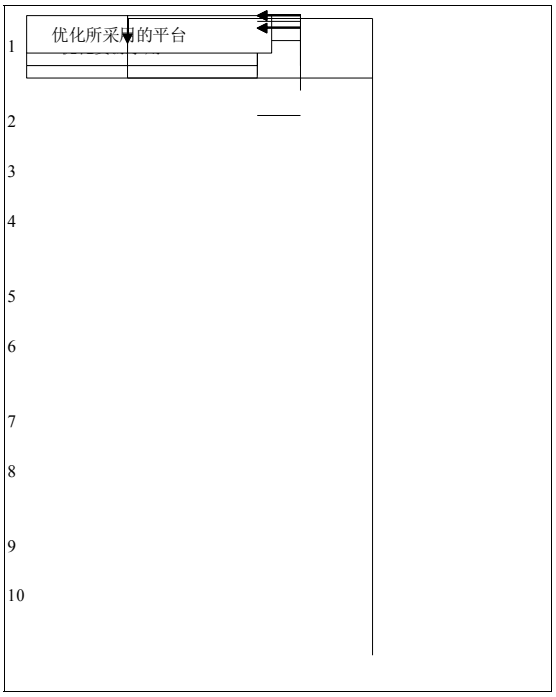
应用程序生命周期内不同阶段的优化代价

§6.1.2 改善产品系统的优化

当拥护抱怨系统的响应时间时，通常优化不是在者种情况下才进行。因为当响应时间比较慢时，再通过实现某些最有效的优化策略来解决，就已经太迟了，出现这种情况时，如果用户还不愿意**彻底重新设计**应用程序，那么就只能通过重新分配内存或优化 I/O 来或多或少地提高一点性能。

§6.2 优化的优先步骤

下面是对基于 ORACLE 应用的优化的推荐方法，它分为 10 个步骤。按照投资回报减少的顺序给出优化过程步骤，对性能影响最大就越靠前：



应用系统的优化方法

§6.2.1 步骤 1：优化商业规则

为获得最佳的系统性能，用户有时需要调整商业规则。主要考虑有关配置问题。比如所申请的线路；所购买的网络设备等。即如果在实际使用时已经发现超出物理的能力，则公司的上

层就应该追加另外的配置方案，如采用多层配置方案等。

§6.2.2 步骤 2：优化数据设计

数据库设计阶段通常要经历规范化阶段，此时需要对数据进行分析，以降低数据冗余，除了主键外，任何数据元素都应当在数据库中只能出现一次。有时又需打破这种规范形式，用户还需要保证数据库通过汇总值经常性地进行记忆。例如，在每次进行访问的时候，不应当强迫应用程序重新计算给定订单中的所有商品的总价。另外，为了更快地访问信息，用户应当建立主关键字和外部关键字索引。

数据设计阶段的另一个考虑是避免数据争用。也就是说，把对数据的访问进行定位，以避免任何请求特定数据范围的进程可以局限到特定的实例。
在 ORACLE 并行服务器中，需要寻找同步点，对设计不良的系统，就存在顺序命令编号的要求，也就是同步的问题。

为了避免数据争用，考虑：

- 将数据分区
- 使用局部或全局索引

§6.2.3 步骤 3：优化应用程序设计

对于某些带智能处理的设计而言，在战略上使用缓存数据技术。在某零售应用程序中，用户在每天开始的时候可以选择一次税率，并将其缓存在应用程序中，通过这种方式，就可以避免一天之中总是重复地获取相同的信息。

§6.2.4 步骤 4：优化数据库的逻辑结构

在设计完应用系统应用程序之后，就需要对数据库的逻辑结构进行规划，这一步主要是对索引的设计进行调整，以保证数据被正确索引。在逻辑结构设计阶段，则应当创建辅助索引来支持应用程序。

对于由于争用所引起的性能问题，经常会涉及到插入相同的块，或者是序号错误的使用。在索引的设计、使用和定位的时候，以及使用序号产生程序和簇的时候，应当格外小心。

§6.2.5 步骤 5：优化数据库操作

优化 ORACLE 服务器之前，应当确保用户应用程序充分利用了 SQL 语言的特性，以及 Oracle 为增强应用程序处理能力的相关特性。根据用户应用程序的要求，可以运用下述特性技术：

- 数组处理
- ORACLE 优化程序

- 行级锁管理器
- PL/SQL

无论用户在编写新的 SQL 语句，或是对应用程序中存在的疑问的语句进行优化，对数据库操作的优化本质上都是关心 CPU,磁盘 I/O 等资源情况。下面是所作的步骤。

1. 查找最消耗资源的语句

利用诸如 TKPROF、SQL TRACE、SQL Analyze、Oracle trace 和 Enterprise Manager Tuning Pack 等工具。可以查出存在问题的语句和存储过程。此外，用户还可以通过 V$SORT_USAGE 视图来查看与临时段关联的会话和 SQL 语句。

在优化工作中，最有可能提高性能的语句包括：

- 整体消耗资源最多的 语句
- 每行消耗资源最多的语句
- 执行频率高的语句

在 V$SQLAREA 视图中，用户可以发现仍然驻留在缓存的语句，这些语句进行了大量的磁盘 I/O 和缓存获取操作。

2．对这些语句进行优化

需要记住的是，应用程序的设计情况是性能好坏的基础。对于低效的应用程序设计方案，不能通过 SQL 语句的优化来弥补它的不足。如果用户遭遇到 SQL 语句的优化问题，那么也许就需要改应用程序设计方案。下面方法可以减少特定语句所消耗的资源：

- 使语句使用更少的资源
- 降低使用语句的频率

由于语句执行大量的事务处理，或者其工作效率低下，或者两者兼而有之，就可能消耗大量的资源。用户可以不改程序，而是更改索引结构；或只需改变 SQL 语句自身（不改环境逻辑）就可以完成任务。

§6.2.6 步骤 6：优化访问路径

为了确保数据库访问的效率，需要考虑使用簇、哈希簇、B*树索引、位图索引、以及优化程序提示。此外，还应当考虑对表进行分析，以及利用直方图表来分析。从而帮助优化程序确定最佳查询方案。

有效访问可能意味着增加索引，或增加特定应用程序的索引，随后再其撤消。还可能意味着建立数据库之后，再对设计结果进行再次分析。如果用户发现实际响应时间比必须响应时间要长，则需要寻找其他的方法来提高设计性能。

§6.2.7 步骤 7：优化内存分配

在 ORACLE 8I， 系统共享内存被动态地分配如下结构：

- 数据字典缓存
- 库缓存
- 上下文区域（如果运行多线程服务器）

用户可以设置下面内存结构：

- 缓冲区缓存
- 日志缓冲区
- 序列缓冲区

内存资源的适当分配可以提高缓存的性能，降低 SQL 语句的解析，同时可以减少分页（Paging）和叫换(Swapping)。

进程的本地区域包括：

- 上下文区域（如果运行多线程服务器）
- 排序区域
- 哈希区域

值得注意的是，对与大量的影响到分页和交换的机器物理内存。不要将其分配相同全局区（SGA）。

§6.2.8 步骤 8：优化 I/O 和物理结构

磁盘 I/O 操作会降低软件应用程序的性能。优化 I/O 涉及到：

- 调度数据，以使 I/O 分配时避免磁盘争用问题
- 最佳访问方式是将数据存储在数据块中：将自由列表设定为合适的大小，以及恰当的 PCTFREE 和 PCTUSED
- 为用户创建足够大的盘区，以避免表的动态扩展，它的负面影响到高容量 OLTP 应用程序的性能。
- 评测原设备（raw device）的使用情况。

§6.2.9 步骤 9：优化资源争用

对于多个 ORACLE 并发请求，会产生对 ORACLE 资源的争应。应避免下面的争用发生：

- 块争用
- 共享池争用
- 锁争用
- Pingping(并行环境)
- 锁存器（latch）争用

§6.2.10 步骤 10：优化所采用的平台

涉及以下方面：

- UNIX 缓冲区的大小

- 逻辑卷管理器
- 内存使用及进程的大小

§6.3 应用优化方法

§6.3.1 设定明确的优化目标

在没有建立明确的优化目标前，最好不要开始进行优化。“使其尽用户所能运转起来”听起来是一个目标，但很难确定实际情况是否已达到了目标。

一个更有用的目标如下：我们需要 20 名操作员，每名每小时输入 20 条 命令，在 30 分钟内必须组装列表。

此外，用户还应当记住，在获得目标后可能存在一些冲突。如为获得 SQL 语句的最佳性能，就会牺牲并发运行在数据库中的其他 SQL 语句性能。

§6.3.2 创建最少可重复测试

用户应当创建一系列的最少可重复测试，如，如果用户却定某条 SQL 语句影响性能。那么就在 SQL*PLUS 中（用 SQL Trace 或 ORACLE Trace）运行原始版本和 修订版本的语句，以便可通过察看统计结果发现性能的差别。

§6.3.3 测试假想

在创建了最少可重复测试后，可通过脚本来执行测试，可对结果进行汇总和报告。通过这种方法，用户可以对各种假设情况进行测试。

从 ORACLE 的缓存算法可以看出，当第一次把数据缓存到内存中，它的开销比以后（只从内存中访问数据）的都要大。因为第 2 次以后不需磁盘读数据到内存。

§6.3.4 记录和自动测试

1.通过分析执行每个脚本及所得的结果记录，此外，用户还应当是测试自动化，有下列优点：

2.可以根据优化程序的能力，更快地对测试的效能进行计算。

3.由于每次测试所使用的设备相同，可保证测试体系方法的一致性。

§6.3.5 避免常见错误

无经验的人经常犯下面的错误：

- 受预先设想的见解影响较大；

- 随机进行各种方案的测试；
- 无目的、无依据的修改环境。

我们应该通过编写用户认为问题出处的描述，仔细推敲分析过程，理清用户的思路，进而确定错误所在。还要请一些对应用程序有较好了解的人参与，验证 SQL 语句优化程序，设计出解决方案。
另外，要避免解决方案以外的**奇怪的想法**，如，通过猜测方法来更改系统的参数，用户对这种做法要十分谨慎，否则做出了某种假设，而用户并没有对这种假设有完整的理解，却急于实现这种想法。于是草率行事。由此导致性能严重下降以至于用户只好从备份中恢复某些系统环境。再就是**避免偏见**。当定位优化问题的时候，要避免偏见，而要用户描述性能问题所在。但也不要期望用户确切知道问题所在。

第二部分　ORACLE 应用系统设计优化

第 7 章 ORACLE 数据库系统优化安装

Oracle 数据库系统的安装与以后应用系统的运行有着密切的关系，如果一个中大型的应用系统没有充分设计和规划，而是采用默认的方法安装，则给以后应用系统的运行带来一定的影响。下面给出一些建议。

§7.1 应用系统环境规划和 Oracle 系统安装考虑

如果在分析阶段得到用户的初步资料，在与用户讨论确认之后就可以订购数据库服务器了。当数据库服务器到货后，就可以与操作系统人员一起规划服务器的操作系统的安装和 Oracle 数据库系统的安装等。

§7.1.1 操作系统安装考虑

当数据库服务器在开箱后，就开始规划如何安装操作系统软件。因为一般的小型机或多数服务器机器在出厂后是不安装任何软件的。所有安装操作系统和其他所需要的软件都是在机器安装完成后由供应商进行的。
为了使所安装的操作系统能满足 Oracle 系统的基本要求，有的服务器的操作系统需要注意某些 Oracle 的要求：

- 操作交换区

交换区是 Oracle 的一项基本的要求。可以根据 Oracle 的发行要求来确定。一般交换区大小的要求是该服务器内存的 2 倍至 4 倍之间。过小的交换区可能导致 Oracle 系统安装的失败，所以建议交换区最好是内存的 4 倍为佳。

- 硬盘格式化的考虑

在安装操作系统时，安装程序回提示将硬盘化分为不同大小的部分。在安装操作系统时就开始考虑哪个硬盘是用来安装 Oracle 系统的，哪个是用来存放数据文件的等。建议用于存放 Oracle 数据库系统的目录一定比 Oracle 系统发行要求的 2 倍以上；其次就是考虑 Oracle 数据库系统的数据文件的目录所对应的硬盘的大小。Oracle 系统所在硬盘最好不要与其他的软件混早一起。

§7.1.2 Oracle 系统安装考虑

当服务器平台已完成操作系统的安装后，就应该开始认真的考虑下面的问题：

- 操作系统的信号量

Oracle 在某些 UNIX 操作系统环境下安装需要合适的操作系统信号量。应该根据 Oracle 版本发行的要求进行设置，比如在 SUN 环境下，需要以 root 登录并根据 Oracle 安装手册的参数要求修改/etc 目录的 system 文件。然后在进行 Oracle RDBMS 的安装。

- 是否采用升级方案

如果应用是将旧的应用系统上进行升级的话，要考虑系统的性能问题。一般建议采用非升级安装，采用人工升级。因为系统自动升级安装会给应用带来性能问题。

- 安装类型方案

采用自定义安装进行 Oracle 数据库系统的安装，这样考虑根据需要定义包括字符集、数据库块的大小、数据文件的大小等。

- 安装点的考虑

Oracle 的安装点就是指数据文件、日志文件和控制文件的安置路径，为了使系统在以后运行性能达到优化，建议将数据文件、日志文件和控制文件的安置路径与数据库系统存放在不同的路径上。最好将数据文件、日志文件和控制文件分别存放在不同的路径。

- SYSTEM 表空间对应数据文件

在自定义安装会话中，建议你根据需要设置 system 表空间所对应的数据文件的大小。一般要设置比默认值的 2 倍。该数据文件的大小最好是在 300MB 至 500MB 间。因为数据文件太小不利于系统的运行。

- 临时表空间对应的数据文件

临时表空间对应的数据文件可以根据将来系统存放的应用的处理情况来定。比如系统将来可能要经常进程排序处理，则需要设置较大的临时表空间，也可能需要再建立新的临时表空间。这里建议临时表空间的数据文件在 100MB 至 300MB 左右。

- 回滚段表空间对应的数据文件

如果是 Oracle8i 及以前的版本，则考虑为 RBS 表空间建立较大的数据文件。最好数据文件在 300MB 至 500MB 之间，如果不够在完成安装后再进行扩展。但是不要采用默认值。

- 日志文件的大小

日志文件的大小对于 Oracle 系统的运行也是相当重要。默认值是太小。建议日志文件大小在 10MB 至 50MB 左右。

- 控制文件的大小

如果是 Oracle8 及以上版本，控制文件文件除了存放数据文件信息和日志文件信息外，换存放恢复信息等。所以控制文件所在目录应该有足够的扩展空间。一般建议在该目录应该有 200MB 以上空间。

- 数据库块的大小

如果你的应用系统是 OLTP 的话，可以采用较小的数据库块。如果是 DSS 类型的应用系统，则可以设置较大的数据库块，目前 Oracle 产品所允许的数据库块可以是 2KB 至 64KB 之间。无论你选择较大的块或较小的块，它的值都必须是 2 的整数倍，比如 2048,4096,8192 等。但需要注意的是，如果操作系统为 64 位，则可选择较大的块。

- 字符集的选择

字符集是 Oracle 系统专门支持的一项技术。详细请参考另外的章节。一般不要与另外的已经存放的 Oracle 系统的字符集产生冲突即可。但如果你的环境是一个新的平台，不需要与其它平台进行数据交换的话，建议选择默认的字符集。这样可以利于将来的修改。

§7.2 关于创建多个 Oracle 实例问题

一部分设计师和用户都这样认为，用户的应用系统有几个子系统，就应该建立几个数据库

（实例）。将每个应用系统建立在一个独立的数据库（实例）上。这样的考虑主要是对 Oracle 系统的结构或工作方式不够了解造成。一般来说，如果用户的应用系统不是非常庞大，服务器的内存也有限，建议不要在同一台服务器上创建两个以上的数据库（实例）。因为每个数据库（实例）在启动后都回占用大量的内存和 CPU 时间。如果有多个不同的应用系统，只要分别为不同的应用系统建立的表空间即可。

§7.3 Oracle 系统安装后的优化基础工作

一般在安装成功后，管理员确认 Oracle 系统正常启动和关闭没有问题后，除了要修改 SYS 和 SYSTEM 帐户的口令外，最好还要做下面的工作：

§7.3.1 Oracle 系统有关目录所有文件的保护

将所有文件，特别是数据文件、控制文件几次日志文件的设置为不可删除的状态。避免任何人有意无意的删除。如果你的环境是 UNIX 操作系统，建议将所有文件设置为不可删除状态。

§7.3.2 避免新用户使用默认 system 系统表空间

在修改了 SYS 和 SYSTEM 帐户的口令后，基本可避免任何人都可随意窗新用户的操作。这时，管理员自己应该在创建新用户时，一定要为用户指定默认表空间。

§7.4 Oracle 系统所在服务器的独立性

由于 Oracle 是一个消耗资源较大的大型软件系统，为了确保 Oracle 系统在运行期间不与其它的软件系统发生资源的竞争。建议将其它软件系统，包括 Oracle9i 的 iAS 软件，不要与 Oracle 系统所在的服务上安装这些软件。以保证服务器资源能满足 Oracle 系统的要求。

第 8 章 项目分析、设计与管理

在性能优化中，许多人把责任推到数据库管理员身上，这是不公平的，因为数据库的管理不好可以在运行一段时间后进行调整。而如果应用系统的数据库结构不合适，则就很难对其进行修改，原因是修改数据库结构会带来许多应用程序的改动。所以，优化的基础应该从项目分析和设计开始。下面给出一点简单总结。

§8.1 项目分析要点考虑

中国的许多应用系统的可性性分析、项目需求分析、概要设计等，一般都是开发方与用户来共同完成，说得明确点，就是一般的用户不会给出项目的需求说明，而是由开发方与用户在讨论中来总结出自己对用户需求的理解，再写出需求分析文档，所以项目的需求分析就变得非常关键了。

§8.1.1 对应用系统类型的认识

在分析阶段和设计阶段及安装 Oracle 系统时，要理解项目是属于哪种类型。要针对应用系统类型来设计和规划应用系统的环境。甚至购买硬件设备时都先考虑进行去。下面是不同类型的应用系统考虑的侧重点。

- 联机事务处理(OLTP)

如航空定票系统、超市收费系统、旅馆预定系统以及面向多用户的系统，这样的应用系统主要特点是同时进行 DML 操作高，则需要高并发。

- 决策支持系统(DSS)

大量的只读统计分析，如数据仓库系统，这样的应用系统特点是用户交互少，但一次所处理的数据量大。对于这样的应用系统考虑用户要少，因为不需并发，但需硬盘有一次读多块。

- 批作业系统

非交互的自动应用，不需高的并发或多线程。

§8.1.2 软件项目计划

在项目启动后，就该开始项目的分析、设计与开发计划。主要考虑：

- 任务分解

将项目分解成为几个独立的子系统；

- 进度安排(人月)

按照各个子系统的工作量、难易程度来估计工作量，从而得到预算费用；

- 关键问题(风险与可能的技术问题)分析

对于任何简单的应用系统，都要对任何可能的困难进行估计。这种关键问题可能是技术问题，也可能是协调问题或者是与第三方有关的问题。总之，要将可能影响应用系统实施进度的问题列出，以便在实施过程采取对策。

§8.1.3 开发环境资源的配置

项目一但落实和启动，就必须先配置开发环境。在本阶段需要考虑的问题有：数据库系统的要求，比如：标准版或企业版；系统的开发工具方面，要从工具的流行、性能、开发人员的熟悉程度等考虑。下面是主要考虑的问题：

1. 内部的开发用服务器内存与硬盘的要求
2. 网络环境的要求
3. 客户端 PC 机器的配置要求
4. 数据库系统的要求
6. 调试工具的要
5. 开发工具的要求

§8.1.4 各种人员的招募要求

人员的落实也很重要，主要考虑用户方配备人员的情况，也考虑开发方的人员的配备情况。有的项目考虑要有用户方的领导参加，使得在项目的推动方面得到支持。

§8.1.5 开发组工作的开始

项目开始后，要由双方主要领导考虑总设计师的人选，总设计人选考虑的要点：必需是近年从事软件设计和开发工作的人员，了解最新软件知识的、设计过大型项目的、用过几种流行语言和开发工具的、有开发成功经验的、工作严谨认真的人员来担任。在这样的基本条件都满足后，就可以进行：

1． 总设计师与项目小组的工作
2． 一般开发人员的任务与准备工作

§8.2 应用系统运行环境分析

一般来说，购置服务器，包括数据库服务器、应用服务器的内存、硬盘、主频、CPU 个数等都要从应用系统的实际出发。不要只是网络人员或主机人员决定，因为服务器的配置是否合适应该经过与用户分析和讨论后才能比较准确地估算出来。那么要考虑的运行环境有哪些呢？作者认为主要考虑下面的方面：

§8.2.1 数据库服务器性能的考虑

只要采用数据库来存放应用系统数据，就需要配置数据库服务器。可以根据应用系统的规模和资金情况选择不同档次的数据库服务器。比如小型应用系统可以采用 PC 服务器；而中型以上的应用系统，建议采用小型机。主要考虑的方面有：

1. 主频-要根据所采用的 Oracle 数据库系统的版本的最低要求来购买；
2. 内存的要求-由于 Oracle 数据库系统要消耗大量的内存，建议内存配置要比最低要求的 2 倍；
3. CPU 个数-如果采用的服务器支持多个 CPU，可考虑采用 2CPU 或 4CPU 等，这样考虑充调度挥 Oracle 的性能。

§8.2.2 数据库服务器硬盘空间的估计

对于没有实际经验的设计师来说，一般对硬盘的估计都偏小。如果这样，应用系统可能在投入运行一两年后就要再扩硬盘才行。在分析阶段完成后，一般可根据下面的几个方面来估计硬盘的空间要求：

- 操作系统的空间需要

一定要询问供应商操作系统的基本要求，包括操作系统总占空间量；

- 其他软件的占用量

比如高级语言、管理软件等；

- 数据库系统的空间占用量

要根据 Oracle 公司提供的软件发行中的说明来估计磁盘空间的使用量；

- 应用系统数据量

可根据在分析阶段的基本资料来估计应用系统的占用量。一般来说要估计的数据量上在乘 1.2 左右；

- 应用系统数据的索引的数据量

一般的设计师可能回忘记这一类数据的空间占用量。其实这类型的数据量也不小，一般要占数据的 25%到 30%左右。

§8.2.3 应用服务器的考虑

如果应用系统采用 B/S 结构，则可以考虑购买 Oracle8i/9i 的 iAS 软件。与上面有相同的要求，也要考虑 iAS 软件的空间要求问题、将来产生的应用程序的空间占用情况等。一般的应用服务器的配置可以考虑比数据服务器稍低些。

§8.2.4 网络带宽的考虑

无论是采用 C/S 结构或在于 B/S 结构，带宽是制约性能的瓶颈。因为大部分业务基本上是在 C/S 下进行的，建议还是采用带宽高的网络方案。

§8.4 数据库逻辑设计

数据库的逻辑设计应该由应用系统总设计师和数据库管理员共同担任，这样可以互相取长补短。要考虑的方面在下面给出。

§8.4.1 系统表空间

system 表空间用于存放 ORACLE 系统的有关信息，一般的用户建立的对象（object）不要放在 system 表空间中。

1. 保护 SYSTEM 表空间：

用下面方法可以避免：

```
alter user user_nane quota 0 on system;

create user use_nname identified by password
default tablespace tablespace_name;

alter user user_name default tablespace tablespace_name;
```

2. 增加 SYSTEM 表空间大小：

另外，由于 oracle 日常的运行会在 system 表空间中存储一些信息，如表名，索引名，以及审计信息等，所以需对 system 表空间增加一些新的空间，如：

```
alter tablespace system add datafile
'/orant /database/system01.dbf' size 100m;
```

3. 确认没有用户使用 SYSTEM 表空间作为缺省表空间：

```
SELECT USERNAME,DEFAULT_TABLESPACE FROM DBA_USERS
WHERE DEFAULT_TABLESPACE= 'SYSTEM';
```

如果已经有个别用户的用了 SYSTEM 表空间，则需要进行及时的处理。包括：
* 修改该用户的缺省表空间；
* 备份该用户的对象和移动数据。

4. 确认没有用户使用 SYSTEM 表空间作为临时表空间：

```
SELECT USERNAME,TEMPORARY_TABLESPACE FROM DBA_USERS
WHERE DEFAULT_TABLESPACE= 'SYSTEM';
```

如果有用户使用 SYSTEM 表空间当临时表空间，则要用下命令加以阻止：

```
ALTER USER <username> TEMPORARY TABLESPACE temp;
```

§8.4.2 数据表空间和索引空间分开

与前面提到的一样，较大的应用系统一般都需要将数据和索引分开存储，这样可以提高系统运行速度，比如为"产品设计制造系统"规划表空间，可以建如下数据表空间和索引表空间：

```
create tablespace Desg_Deve_data
datafile /oracle_home/…… Desg_Deve_data01.dbf
size 500m; /* 数据表空间*/

create tablespace Desg_Deve _indx datafile
    /oracle-home/… Desg_deve_indxd1.dbf
size 100m; /*索引表空间*/
```

如果已将索引与数据存放于一个表空间内，可以用下面命令将索引迁移到索引表空间中：

```
alter index emp_no rebuild
tablespace Desg_deve _indx
storage (initial 2m next 2m);
```

§8.4.3 回滚段设置

在缺省情况下，orade8i 有 24 个回滚段是建在 RBS 表空间下， 其中约有一半的回滚段是处于联机(online)状态，而且 RBS 表空间对应的数据文件也较小。在实际应用中，有时需处理大型的事务。其大小都会超出缺省回滚段的大小，当这样的情况发生时，事务往往不能成功。建议应用总设计师和 DBA 必须做以下工作：

1)扩展表空间 RBS

alter tablespace RBS add datafile
'/oracle-home/…/RBS02.dbf' size 100m ;

2)重新设置 Rb01, Rb02,…回滚段的参数：
alter rollback segment Rb01 shrink to 2M;
alter rollback segment Rb01
storage (next 2M optimal 10M);

3)为专门的巨型事务建立大的回滚段，如为产品设计与开发建立一个大的回滚段：
a) 先建立一个表空间 product_rbs:
create tablespace ptoduct_rbs
datafile /oraele-home/…/productrbs ' size 200M;
b) 在表空间 product_Rbs 上建立一个回滚段 prod_roll:
create roll back segment prod_roll
storage(initial 50m next 5M optimal 60M);

4)建立大型事务使用大型的回滚段：
set transaction use roll back segment prod_roll
注：必须在事务处理前使用 set tramsaction 语句，
如果当某一事务已提交或回滚后还想继续使用大型回滚段，则必须在 commit 或 Rollback 语句后加上 set transaction 语句。

§8.4.4 临时表空间设计规划

临时表空间主要用于查询操作中的 distinct , union , order by 以及 create index 操作。Oracle 缺省表空间为 Temp，其大小为 1MB，显然对于一个真正的中、大型应用系统是不够的，因此需总设计师或 DBA 作如下工作：

1.增加 Temp 表空间
alter table space temp datafile
'/oracle_hime/…/temp02.ddf' size 100M;

2.指定用缺省使用 temp 空间
1） 建立用户时指定该用户缺省使用表空间 Temp
create user user_name identified by pass_word
default tablespace tablespace_name
temporary tablespace TEMP;
2） 已建立的用户，将其缺省临时表空间为 Temp：
alter use user-name
temporary tsble space Temp;
3.为应用系统建立专门的大的临时表空间，用于进行系统的月报、季报、年报统计等。

处理比如为产品设计与开发系统建立一个大的临时表空间 prod_temp
1) create table space prod_temp
datafile '/…/…/prod_temp_data.dbf' size 200M;
2) alter tablespace prod_temp temporary;

§8.4.5 数据文件和日志文件在不同磁盘上

数据文件的写入是通 DBWR 后台进程实现，日志文件的写入是通 过 LGWR 后台进程实现，由于日志文件是连续的写入，因此无并发处理现象。而数据文件的写入相对是随机的，为避免在同一时间内 DBWR 和 LGWR 的冲突，应将日志文件和数据文件放在不同的硬盘上，如果一定要将数据文件与日志文件存在同一磁盘，则数据文件不应当属 system RBS 表空间，也不应当属于非常活跃的 DATA，IDEX 表空间。否则将可能导致与日志文件的直接冲突，并且增加日志文件被数据库操作破坏的可能性。

另外，在 ARCHIVELOG 模式时，数据库会在覆盖（第一个）日志文件之前将其进行拷贝（即写入磁盘或磁带），这些拷贝操作由 ARCH 后台进程执行。当数据库正在处理频繁的事务时也产生日志文件间的磁盘冲突，这是由于 LGWR 总是在当 ARCH 试图读取另一个日志文件时，正在同时写入一个日志文件，这种冲突的避免只能通过将日志文件分配到多个磁盘才能解决。因此在 ARCHIVELOG 模式运行一个面向事务的数据库时，要避免 LGWR 与 ARCH 之间的冲突，可以将日志文件分配到多个设备中去。

§8.5 数据库物理设计

逻辑设计考虑的是应用系统的逻辑结构，而物理设计则是在数据库系统上进行具体的考虑和操作的工作。

§8.5.1 定量估计

就数据库而言，定量估计主要是事务分析（有时称作容量分析）和大小分析。事务分析测量以下项目的最小、平均、最大值等。主要考虑：

* 并发用户数
* 响应时间
* 经过时间
* 事务数
* 并发程序数
* 读写字节数

应用为 OLTP，录入数据等，关心并发用户数，响应时间应用为批处理，如大量加载数据统计等，关心单位时间内 I / O 多少字节（不关心用户数）
这些数据的来源主要有二：人工分析估计；由 DBA 利用数据库提供的功能进行测量。数据库管理员负责测量结果。

1.定期定时测量并发用户数、事务数
（1） 高峰期用户数
（2） 低峰期用户数
（3） 平均用户数
方法有二；
（1） 用操作系统命令，如 PS 等
（2） 用数据库提供的方法，如
* 查询 v$sesson v$process, v$bgprocess 数据字典等
* 用 oracle 提供的脚本 utlbstat, utlestat 进行统计等

2.记录那些计算量大的应用所用的时间
主要任务有：
* 启动应用后起止时间
* 当中用户个数
* CPU 的情况
大小分析
（1） 逐个表进行估算，方法有二；
* 人工估算，按照字段本质、月记录，平均度来计算（具体算法参见后面的 storage 参数）
* 使用 oracle 提供的 analyze 命令进行统计
（2） 估计值乘以此例固定，一般为大小 120%

§8.5.2 表空间与数据文件

除了安装完成系统已经提供的 SYSTEM、RBS 等表空间外，建立中大型的应用系统时，还要规划和创建新的表空间。下面是规划要考虑的几个方面：

1.数据类型：
总设计师要了解表空间将要存放什么类型的数据。是表或索引。

2.不稳定性：
数据表空间在插入、修改与删除数据时变化的频率和数量。比如 OLTP 类型的应用，典型的有商场的交易、定单的录入类型的应用每时都在变；而雇员的信息的录入可能几天变化一次。也就是说不稳定（变化快）的数据会引起重做日志的长时间处理。

3.数据的重要性：
要知道系统中哪些数据是非常重要，比如银行的储户信息和存款明细。相对来说月报、季报数据就次要些。

4.数据是否可重建：
有些数据可能是来源于其它的数据。如电信的用户月终应交费用等。

5.表空间类型：
表空间要分类型，比如较固定的表和重要数据，应该创建在新的硬盘上，因为旧的硬盘出故

障的可能性要大。此外。要考虑创建只读表空间存储历史数据。

6.同类数据放在同类表空间：

变化频率高和相对固定的数据要分别存放于不同的表空间。为那些经常在程序中存放临时结果的应用建立临时表空间。

§8.5.3 物理设计原则

物理数据库实际上是逻辑设计的预优化设计（第二阶段）原则如下：
* 分而治之，划分分段和并行化
比如把具有大量运算，且需较长时间的应用分开处理或并行处理或者平时处理
* 预分配，根据经验估计来预分配资源，不要采用动态分配，困为而需在额外的计算，I / O 以及不合理性。
* 预激活，考虑到整个系统将来运行能出现的各种问题，预先作出一些设计，这种考虑没有统一的方法，比如预先估计可能需要的回滚段表的个数、临时段大小，光标个数等。
* 应用分段，考虑服务器，客户端以及网络通讯流量等，把一些具有连续（瀑布式）的放在服务器端。
总之，在物理设计阶段，要的下建议：
* 表和索引分开
* 将大表，大的索引进行分区或放在不同的磁盘上
* 将频繁进行连接查询的表用簇将它们聚在一起
* 将那些不常连接的表放在一个盘上，如数据字典等
* 数据字典与表索引分开
* 如果呆能将回滚表和重做日表放在不同的磁盘上，RAID 盘不考虑
- 使用 RAID3 或 RAID5 具有奇偶信息的分区存储表数据

§8.5.4 数据库物理设计内容和步骤

1、 设计和确定各分系统或子系统
把整个系统按类型分为不同的分系统或子系统，如：
财务系统
人事系统
成本计划与控制系统
2、 确定各分系统年数据量
用估算方法计算各子系统可能空间需求字节数
3、 确定各分系统对应的表空间及数据文件，根据第 2 步的估算字节，建立各表空间，根据物理设计原则将表、索引等分别建立表空间。
4、 为整个分系统建立相应 oracle 帐户
对每一个分系统建立相应的 oracle 帐户，这些 oracle 帐户将来成为该分系统的所有者，建立帐户时，要按 oracle 的要示指定相应的角色和缺省表空间等。
5、 以分系统方式分别建立各类对象

将所有数据库结构设计好，并经过认真核实后以分系统进行建立对象，开始建立的对象必须包括表、索引、同义词等，以后陆续完成视图、过程、、触发器的建立。
6、 核实以上各步的正确性，并对每一步的操作进行文档登记，不要操作完毕就忘了。
7、 规划开发人员，使用人员角色
以上建立的 oracle 帐户一般是以管理员方式进行的，对于开发人员，使用人员要求建立相应的 oracle 帐户，当然开发人员应具有与该分系统的所有者一样的权限。对于使用人员一定要认真进行登记和分类，根据不同的人员（比录人员、领导）进行划分角色，这样的工作必须写成文档方式，就象工资表一样，当确各类人员的角色后，在相应的分系统上建立不同权限的角色，并赋予不同的用户。

8、 进行相应的安全管理
规划安全管理机制，比如登录市场审计等，当系统提供使用后可以使安全审计有效。进行相应的日志。

9、 监督项目进展情况
在开发阶段，制定必要的目标管理定期检查各系统的情况，检查的内容包括，程序编制完成情况，程序规范情况，文档的完成情况，对于未按时完成的工作找出主要原因，及时调整计划。
10、 开发阶段自我测试和联合测试
每一个子系统的每一个模块应按照系统详细设计中的测试说明书进行测试，只有测试完成要求后，方可进行另外模块的开发。
在自我测试完成，还应该进行相应测试和联合测试。

11、 用户测试
无论开发人员是否作过多少次的测试，在实际数据输入运行前必须经过使用者的测试，用户测试包括，操作简单、美观以及正确性等，当用户提出改进意见或找出错误，开发者必须及时改正。

12、 数据加载和数据检查
当测试通过后，可以将实际数据加到系统中，但首先要消除原来的实验数据。在加载数据和开始录入新数据时，开发人员必须和使用者一道联合工作（肩并肩）。这样，可以及时发现问题和有时改正。
13、 完善使用说明书
在开发当中，各开发人员必须独完成各自模块的使用说明文档，当软件的改后要及时对文档进行补充。

§8.6 开发过程管理

在国内，应用系统从分析、设计、实现到维护的管理经常被忽略，特别是在开发编程阶段，由于开发人员的随意性，加上人员的流动没有给出完整的文档。给应该系统的进度控制和质量控制带来很多困难。在这里，我们给出一点管理控制的摘录和经验介绍。仅供参考。

§8.6.1 应用软件生命周期阶段的管理

虽然现在流行的是原型法的，但整个软件系统还是经过：

分析==》设计==》实现==》维护 这样的阶段。

分析阶段：

分析阶段要做的工作是：
* 用户需求的全面收集并进行文档编制；
* 分析各个需求的详细要求；
* 与具体使用人员接触，得到更接近将来系统需求的内容；
* 简单给出系统的将来功能性要求；
* 不要随便承诺将来系统的更多功能和性能。

设计阶段：

设计阶段要做的工作是：
* 按子系统进行设计，并给出包括算法、输入/输出等的初步考虑。并进行设计文档编制；
* 给出系统的性能上基本指标，如数据的精度、交互时系统的响应速度，但在响应速度上不要提得过高，避免将来用户拿这些指标来做验收；
* 给出系统的公共编码数据结构设计；
* 给出各个子系统的数据结构详细说明，建议按照 Oracle 的建表结构顺序给出，以便对文档的修改和双方认可后直接转换成 Oracle 能运行的脚本；
* 保证所有的数据结构是全局的（统一的）；
* 给出各个模块的测试要求和样本数据；
* 与用户和 DBA 一起进行审查，必要时要请具有实际开发经验的软件工程师进行评审。

实现阶段：

实现阶段要做的工作是：
* 进行编码前，总设计师要给出编程的要求，比如注释的多少、语句行的长度、程序模块的字节数等。最好给出一些样例；
* 选择能满足要求的开发工具或语言，建议一个系统不要采用过多过杂的开发工具；
* 明确各个模块必须采用的开发工具，比如月报、季报等统计要用 PL/SQL 来编写，网上查询要用 JAVA 来开发等；
* 明确各个模块之间的接口约定；
* 测试的明确要求，即每个编程人员必须对自己所开发的模块进行过自测，再让开发组进行互测。不要没有调试完毕就让用户来测试或提意见。否则给用户一个不好的印象；
* 每个开发人员完成自册后要写出操作文档，即根据自己的开发模块对设计阶段的操作文档进行修改，以便用户可以按照你的文档进行测试操作；
* 小组测试和联合测试；
* 组装测试和文档汇总；
* 交用户测试，并要求用户根据所测试的情况写出测试报告，报告中要明确那些模块需要优化

和修改等；
* 对测试的修改要求进行分工和修改，给出下次用户测试的时间；
* 用户测试，合格后进入试用和维护阶段。

维护阶段：

维护阶段要做的工作是：
* 交互使用时的头一两周，要求所有参加过开发的人员必须在场；
* 对所出现的各种错误分为立即可以修改和稍后修改，并进行记录。对立即可以修改的部分进行修改；
* DBA 对系统运行进行监视，特别是表空间的使用情况、表和索引的扩展情况；
* 对问题进行修改并记录，修改完成并确认没有问题后再让用户进行测试；
* 进行必要的文档整理和产品市场化的工作。

§8.6.2 成功的三要素

生命周期四个部分组成：规划（planning）、组成（creating）、监控（monitoring）、协调（tuning），而这一周期的成功与否取决于三个方面：培植过程（cultural processes），管理过程（management proeesses）和技巧（technalogy）。在数据库开发人员的管理，要求执行以下三个方面：
1)培植：团体的气氛及开发小组应全力支持 SBA 的需求。
2)管理：全体开发人员，对数据库生命周期的方法讨论保持认可与支持。
3)方法：DBA 及开发人员，必须定义相应的机制，以确保在开发的档同阶段采取的方法。

§8.6.3 培植过程

应用系统开发的成功与失败，很大的因素是三个方面决定的，即用户、设计与管理及开发人员的技术水平。建议：
* 要求有最终用户的参与，比如，具体的使用人员经常与开发方在一起讨论和提要求；
* 打破总设计师、DBA 与开发人员的区别；
* 如果可能，有专门的文档编制员；
* 所有的数据结构始终挂在墙上；
* 统一的数据库结构，不允许随意对数据库结构进行修改，必要时经过总设计师的批准；
* 培养良好的团对精神，责消除他们之间原有的势力范围之战，提高效率；
* 仅早发现技术难点，特别是不能或不易实现的技术；
* 挑选专人负责解决技术难点，并将解决的问题通知大家；
* 将公共的部分写成视图或标准代码供大家参考；
* 允许 DBA 在整个过程对应用系统程序的参与。

§8.6.3.1 定义环境

在应用系统的生命周期中，系统的设计和编码及测试都是同等重要，不能只注意设计和程序的编写，特别要重视如那机的测试工作。即要考虑下面的方面：

开发（developmant）
系统测试(system test)
验收测试(aceptance test)
强化测试(stess test)
产品(production)

§8.6.3.2 角色定义

虽然 Oracle 系统有多个角色，但管理员可以根据需要来规划和设计新的角色。比如，一般的程序员可以具有 resource 角色，个别人员有 DBA 角色等。不要所有的人员都具有 DBA 角色或过大的系统权限。

§8.6.3.3 方案报告

生成系统方案清单，以便在实际检查是否被遵守。清单内容应包括如下方面：

* 实体联系图
* 物理数据库图表
* 空间需求
* 协调目标
* 安全需求
* 数据需求
* 执行计划
* 接受性测试进程

1. 实体联系草图（E—R 图表）
（Entity Relationship disgram）
2. 物理数据库图表
数据定义语言描述的表结构等
3. 空间需求
数据库表、索引所需的初始化空间、年数据量等。年数据量计算公式为：

blocksize * (rows in 12 month * avg row length)/ (blocksize - 90) (1- pctfree/100)

4.协调目标
建立两套目标：管理目标和扩展目标

5.安全要求
. 确定开发中用列的帐户结构

. 拥有权、授权方式、所有角色和权利定义
. 将批处理帐户从联机帐户中分离
. DBA 在符合数据库安全的同时能满足应用需求

6.数据需求
明确定义数据输入、输出方法
在应用测试阶段，应同时测试数据变化的需求
备份与恢复要求的描述

7.执行计划
在执行查询请求时，数据库将要经历的步骤，可以用 explain plan 语句生成，记录数据库的重要询执行计划。

8.接受性测试过程
定义功能及执行目标，用于组成测试进程，表述如何处理，未及目标情况等。

§8.7 确定应用程序类型

在进行数据库系统的设计时，除了上面的建议外，一个不可忽视的问题是弄清你所面对的应用是属于那种类型。目前流行的应用类型有：
* 在线事务处理(OLTP=Online transaction Processing)应用程序
* 决策支持系统(DSS=Decision Supoort System)应用程序
* 多目的应用程序

§8.7.1 在线事务处理(OLTP)

在线事务处理(OLTP=Online transaction Processing)应用程序具有高的吞吐量，并且是Update、Insert 的密集型。如订票系统、订货系统等。在 OLTP 设计时，要考虑并发用户和系统性能问题，由于索引和簇会降低 Insert 和 Update 的速度。所以用户还要避免对这两种结构的过度使用。
对于 OLTP 系统，下面因素是关键：
* 回滚段
* 索引、簇、哈希
* 离散事务
* 数据块大小
* 缓冲区缓存大小
* 表和回滚段空间的动态分配
* 事务处理监视和多线程服务器
* 帮定变量的使用
* 共享池
* 分区

* 优化过的 SQL 语句
* 完整性约束
* 客户/服务器体系结构
* 可动态改变的初始化参数
* 流程、组件、功能

§8.7.2 决策支持系统(DSS)

典型情况下，决策支持系统大量的信息转化为用户定义的报告。DSS 对从 OLTP 应用程序收集的大量数据进行查询。
决策支持系统的关键目标有三个：1）响应时间；2）精确性；3）可用性。在设计 DSS 系统时，要确保大量数据的查询在合理的时间段内执行完毕。如在白天，决策者需要看报告，所以要保证在前一天晚上完成所需数据的统计。
对于 DSS 的优化，下面因素至关重要：

* 实体化的视图
* 索引（B*树和位图）
* 簇、哈希
* 数据块大小
* 星型查询
* 优化程序
* 使用查询提示
* SQL 语句中的 PL/SQL 功能
* 分区

§8.7.3 多用途应用程序

典型的 OLTP 和数据仓库系统的结合就是多目的应用程序。因为 OLTP 应用程序收集数据，随后将数据提供给数据仓库系统。
如果 OLTP 和数据仓库系统使用相同的数据库，就存在目标冲突，即可能回影响性能问题。为解决该问题。首先，OLTP 数据库在收集到数据后，就将数据复制进另一个数据库中。该数据库为数据仓库所查询。如果每天只复制一次，则这种配置可能会使数据仓库应用的精确性受到轻微的折衷。但是对于两个系统的整体来说，可获得更好的性能。

§8.8 注册应用程序

为了注册应用程序执行数据库操作或应用的名称，开发人员可以结合 Oracle trace、SQL Trace 实用工具使用 DBMS_APPLICATION_INFO 来进行。
通过对应用程序进行注册，DBA 可以根据模块来跟踪性能。当应用程序被注册到 ORACLE 系统后，可以从 V$SESSION 和 V$SQLAREA 视图中查到。

§8.9 Oracle 配置

取决于用户系统的硬件配置，可以对系统进行不同的配置。主要有：
* 分布式系统
* 多层系统
* Oracle 并行服务器
* 客户/服务

§8.9.1 分布式系统

分布式应用程序将多重数据库中的数据分布到多台机器上。在 Oracle 里，分布式系统还允许用户将数据存储到多个位置，并且站点都可以透明地访问全部数据。
分布式系统的目标包括 可用性、精确性、并发性和可恢复性。在进行分布式系统的设计时，主要考虑数据所在的位置。对于客户经常使用的数据，应当保证客户对它们能快速访问。此外，还保证不要经常做远程操作。复制是处理数据定位的一种方法。在设计分布式系统时主要考虑：
* 网络配置
* 分布式数据库设计
* 对称复制
* 表快照和快照日志
* 流程、组件、功能

§8.9.2 多层系统

多层系统具有以下部件：
* 启动操作的客户或初始部件
* 执行部分操作的单个或更多的应用服务器，应用服务器是一种进程，它可以为客户和执行某种处理的应用提供到数据的访问。
* 终端或数据库服务器
* 验证客户的信认证书（如浏览器）
* 建立到 Oracle 数据服务器的连接
* 代表客户来执行所请求的操作

§8.9.3 Oracle 并行服务器

在大型应用中，可以使用 Oracle 并行服务器，它允许多台机器通过独立实例来访问同一数据库。
在配置 Oracle 并行服务器时，关键是防止各个节点之间的争用。如果多个节点请求对相同的数据进行 DML 操作，那么在一个节点获得锁以前，必须首先将数据写磁盘。这将降低性能。为获得较好的性能，数据必须被有效地在各节点之间进行分区。

§8.10 Oracle 数据库增长的规划

数据库对空间的要求是随着时间的推延而增长的，在这方面没有特别的理论可以参考，一般都靠经验来确定。但是，作为管理员和设计师应该对各个应用系统有一个基本的估计。下面是一点参考。

§8.10.1 不同增长表的配置

管理员应该根据设计师给出的数据库结构脚本来规划数据文件和表空间。即根据建立表、索引和簇的 STORAGE 的存储参数来确定建立多大的数据文件。主要考虑：

1. 事务相关表

应用中的大部分表属于事务相关表。事务相关表存放事务相关的数据。事务相关表的大小与事务的大小相符。大的事务通常使用大的事务相关表。尽管事务相关表的大小在增涨（依赖于时间和事务变化）。事务相关表可以作调整，事务相关表的调整问题依赖于表的大小和增长率。

解决办法：
* 事务相关表一直很小，-->考虑建立完全索引和高速缓存；
* 表中列的顺序和索引列的顺序对查询有影响；

2. 应用相关表

应用相关表用来给应用提供相关的数据，例如：完整性引用的一系列编码值。如温度代码表 temperature_code。应用相关表的特点是：1）应用相关表存放的数据不一定唯一；2）应用相关表一直很小；3）应用相关表比较稳定。

应用相关表的调整问题：
　由于应用相关表通常很小，考虑建立完全索引和高速缓存。

3. 商业事务处理表

商业事务处理表中存放着应用中的大部分事务。如：定货系统中的商品明细表。商业事务处理表的特点是：1）数据不稳定；2）数据量大，增长快。

商业事务处理表的调整问题：
　由于从一个商业事务处理表中删除数据行，但该行所对应的索引不会被 ORACLE 重新使用，当商业事务处理表多次被删除和插入时，它所对应的索引会增涨（称索引停滞）。对于这种情况需要我们重新建立索引。如果商业事务处理表的增涨率很大，可考虑为这些大表建立专门的表空间。

4. 临时接口表

临时接口表在数据处理时临时使用的表，一般用户不直接访问这些表。由于临时接口表大小变化不确定性，因而也有调整问题：

临时接口表的解决办法：
临时接口表也有索引停滞，顶部标志问题。
当使用 SQL*loader direct path 或在 insert 中用 append 时，顶部数据用来确定数据从什么地方开始装载。
如果临时接口表的大小经常改变但它还不变小，则应用 truncate 来清除表中的所有记录。

§8.10.2 对增长表进行规划和分析

为了估计算数据库的增涨，应按前面方法对各种表进行分类，并用下面方法监视增长：

* 监视分配段的空间（从 DBA_SEGMENTS 得到）；
* 分析表，索引等（用 analyze ...）后，查询
  DBA_TABLES,DBA_INDEXES,DBA_IND_COLUMNS;
* 记录表中大小的变化，跟踪增长最快的商业事务处理表的变化率；
* 仔细观察可修改表的索引，索引中可修改的值越多越容易产生索引停滞；
* 定期使用

alter index inxex_name tablespace xxx storage(initial xxm next xxm)rebuild;
来重新创建索引。

第 9 章 数据库结构设计要点

任何应用系统的高性能运行，最基本的是数据库结构的设计。数据库结构是整个应用系统的根基，如果结构设计不好，只在数据库的参数来优化恐怕也不理想。下面给出关于 Oracle 环境的数据库结构设计的一点介绍，希望指出的是，这里的内容仅作者本人的一些经验和体会，不是理论和方法。仅供Oracle 应用开发人员参考。

§9.1 分析阶段的对表的理解

在系统需求分析阶段，一般需要有经验的系统分析员及编程人一起与用户进行交流。这个阶段主要是听用户对需求的描述。但是当我们对用户的需求有初步的了解后，就需要分析员将这些用户需求变为文档，即写出数据需求定义。
一般来说，对用户的复杂的表结构、表中再套小表这样的大表，需要将它们的数据之间理清，拆开成几个相互有关系的表结构。不要简单地将用户的表原封不动地进行转换。

§9.2 正确的主键字段的选择

选择主键字段的前提是，该列的值不能有重复，也不能空，这是基本的要求。此外，建议注意下面的问题：

该列的值不能过长，比如不能使用单位名称作为主键；
建议用字符型或数字型（整数）作为主键；
不要用日期型，浮点型之类的字段作为主键；
如果只用一个字段作为主键出现重复，可采用加校验位或选用两字段作为主键，但不推荐用 3 个以上的字段作为主键。

§9.3 字段类型及长度的选择

对于一个表的字段来说，不同的设计者可能给出不同的类型，有时字段确实可以定义成不同的字段。在这方面，目前从理论上没有明确的限制和规定。因而许多人就认为字段的定义只要能满足用户的要求即可。其实对于一个要求很高且复杂的应用系统。定义字段可以说是一项值得认真考虑的技术问题。本人多年的应用设计和开发中的一点经验，仅供参考。

§9.3.1 如果能用字符型就不要用数字型

在许多地方，有一些字段你可以用数字型也可以用字符型，比如员工的身份证号，从表面看，它的内容全是数字。就由于它的表面特点，所以大多数人认为用数字类型是肯定的。其实我们想一下，身份证的每一位都有其特殊的意义，在数据的输入和查询中，都有一套严密的核查方法。比如最后一位表示男（1）或女（2）。当我们用字符型时，可以在输入中用 where substr(per_id,15,1)='1' or substr(per_id,15,1)='2' 来检查数据的正确性。如果采用数字这样的判断就不那么容易了。
另外，应该提醒许多新的 Oracle 使用的是，在 Oracle 里，字符型也能进行运算。如：

```
SQL> desc abc
 名称                      空?     类型
 ----------------------------------------- -------- ----------------------
 NAME                                      VARCHAR2(20)
 SAL                                       VARCHAR2(5)
 COMM                                      VARCHAR2(5)
 TOT                                       NUMBER(6)

SQL> select * from abc;

NAME       SAL   COMM      TOT
---------- ---------- ---------- ----------
Jodan      2234  1000         0
Johan      2000  2000      3000

SQL> select sum(sal),sum(comm),sum(to_number(sal)+to_number(comm)) tot
  2  from abc;

SUM(SAL)  SUM(COMM)   TOT
----------  ----------      ----------
     3234      3000       6234
```

§9.3.2 相互产生运算的数字型字段长度和精度要一致

在字段定义中，除了要求数字型必须能容纳下以后可能变化的数据外，还需要注意的地方就是：凡是相互可能参加运算的字段，它们的长度及精度最好要一致。这样做，初看起来，好象有的数值小的字段很浪费空间，其实，不要担心空间浪费而专门考虑需要多少字节就够了的想法。因为如果数字型的字段长度和精度参差不齐，有可能在某些运算中产生不必要的错误结果。比如，我们可能用 PRO*C 来编程，当我们根据表中的数字字段定义变量时，C 编译不会检查这些变量与表中字段的一致性关系。举例来说，在表中字段说明很大，在 C 语言中可以说明的很小，但是当你将表中的字段的数据取出放到变量时，C 语言并不提示任何错误，

这样，如果我们不作任何转换就直接进行运算的话，可能出现另外的结果。

§9.3.2 不要为了节省空间而将字段的长度缩小或拆开

现在的存储硬件发展很快，不要将以前的教科书的一些过时的理论带到当前的应用设计中来。这里所说的是指，实际的用户需要就是字段的长度需要。比如，有人经常将日期分成三个字段来定义，可能写 nian char(2), 即两位的年； yue char(1), 即 1 位的月，ri char(2), 两位的日。这样在年份只能存放两位，2000 年存为 00，2001 年存为 01 ；而 月份就不同了，设计者要求用户 1 月输入成 1， 2 月输入成 2 ,...而 10 月输入成 a ； 11 月输入成 b 等。
这样的设计好象节省了空间，其实也没有节省多少，一条记录节省 2 个字节，几千万条记录才节省才节省几十兆字节。这样做无形中增加了程序的处理时间。对于优化和将来的移植很不利。

§9.4 将 LOB 类型的字段与其它的类型分开

Oracle8i 提供了许多可以用于存储大对象数据的类型，如 LONG、LONG RAW 等。从性能出发，建议在设计表结构时将这些大对象类型与其它的类型数据分开存储。比如职工的基本档案，档案上有职工的基本情况信息和照片，在设计最好将照这样的 LONG RAW 类型与职工基本信息分开。然后采用唯一关键字段进行连接。

§9.5 采用具有编码的设计方法

对于具有在多处被使用的值应采用编码来设计，如职工的单位名称，因为一个单位有多名职工，如果每个职工对应的记录都有单位名称的话，就出现所谓的冗余。编码一般有两种，除了前面提到的冗余以外，另外还考虑一些在应用中的使用的方便性的问题。比如银行的存款应用。可以考虑设计一个叫“交易代码”的代码，它表示分别表示“存入”、“取出”及“结息”。可以将该字段取名为：

```
tran_code   char(1)  check (tran_code='1' or tran_code='2' or tran_code='3') ,
```

在设计操作处理界时，只给操作操作者选择“存入”、“取出”或“结息”三种可能，这样可以避免让操作员直接输入字符所带来的不一致等的问题。

§9.6 建立公共字典表

除了上面提到的将具有共性的内容建立统一的代码表外，一个在设计中经常采用的方法就是：建立对象数据字典，与 Oracle 系统的数据字典类似。应用系统的数据字典也是为了各个

对象的命名标准而采用的方法。凡是表名、索引名、表中的列名以及过程所用名称等最好以数据字典的方式在数据库建立。这样可以让 Oracle 系统来帮助我们进行检查。当一个新的对象需要命名时，可将其建立在这个字典中。字典中的名称要经过设计小组和用户讨论通过，通过之后就不能随便改动。在必须改动的时候，一定要经过 D B A 的同意并且背案才能修改。另外一个建议的是。在建立对象名称时一定要避免与 Oracle 的保留字同名。为了做到这一点，建议先将 Oracle 的保留字（各种关键字）放入一个表里，然后再建立时进行比较，以达到避免重复的目的。

§9.7 哪种类型的表设为 cache 方式

Oracle 提供了一种方法，可以将表的数据驻留在内存的S G A 区内，这样叫做缓存（cache）。一般来说，把那些数据量不大，而且使用频率很高的表采用驻留的方式使 Oracle 系统完成启动后就被读到内存的 SGA 区中。这样的操作命令非常简单。比如：

```
ALTER TABLE emp CACHE;
```

§9.8 数据表和索引分开原则

按照 Oracle 的性能要求，建议在大型应用系统中，将表和索引分别存放在不同的表空间里，以得到较高的性能效果。一般在建立表和索引之前，先按照应用系统的年数据量估计表空间的大小，要考虑该应用系统的所有表的可能数据量来估计。如果在 Oracle 系统内要存放多个应用系统，如人事子系统，工资子系统等。则应该为每个子系统建立相应的表空间。在数据表空间规划好后，接着考虑规划对应索引的表空间。索引表空间一般要比数据的表空间小些，一般考虑是数据表空间的 1/3 或 1/2 即可。

在数据表空间和索引表空间都建立好后，要在建立表结构和建立索引的脚本上指定将它们存放在哪个表空间上。这样可避免缺省使用系统的 SYSTEM 表空间的问题。

§9.9 是否采用簇和分区

对于那些具有主－从关系的表，并且在处理时，经常是由主表来访问子表的相应记录的要求，建议采用簇 结构来建立表。比如会计到银行去给单位的所有职工发工资，就要找到该单位，然后对该单位的每个职工进行工资拨付处理。这样的应用使用簇进行设计要比用主键外部键要有效的多。

§9.10 表和索引的空间预分配

在设计 Oracle 应用时，我们经常采用在分析阶段得到的数据来估计空间的需要量。这样当然是由用户给出的，我们一般都是根据经验来决定。虽然是经验，但也有一个基本的规则。这

些规则就是：以用户给出的各种表的每个字段的类型及长度、各表的年数据量（不要以月来估计）、加上相应的余量后得到表及索引的预分配空间大小。这里所谓的预分配就是说先给每个对象分配一个估计的空间。在以后运行一段时间后在进行调整。
另外建议是：在对象的存储空间分配中，不要使用 pctincrease 这样的参数，最好是一次就分配够一年的数据量，而且下次扩展（NEXT 参数）也要分配以半年左右的数据量为基本值。不要过于小气。这样对于性能有好处。

§9.11 确定数据库对象存储大小

下面给出 Oracle8i 数据库（包括表、索引及簇）对象大小估计的方法简单介绍：

§9.11.1 非簇表的大小计算

一般在创建表、索引时，除了指定初始空间外，这应指定记录的年增长值。要估计其大小，需要考虑下面方面：

1．计算表或索引的块数：
数据库块＝2048bytes
块头信息＝90bytes(oracle8/8i)
块有效空间＝
2048－90＝1958 bytes available （有效字节）
用于更新的空间＝块有效空间*(pctfree/100)：
假如 pctfree=10，则
＝用于更新空间＝1958*(pctfree/100）
=1958*(10/100)
=196 bytes
即在有效的自由空间中，仅有 196 字节用于行扩展。

行记录使用的有效空间
＝1958-196=1762 bytes available

行平均长度

如果表中已有数据，则可以用下面命令计算（假定只有三列）：

```
Select  AVG( NVL (vsite (column 1 ),0 )+
        AVG(NUL  (vsite (column 2 ),0 )+
        AVG(NUL (vsite  (column 3 ),0 ) ) Avg_Row_len
        From  table_name;
```

每一行使用的空间：

如果平均长度 AVG_ROW_LEN＝24，这应加上每一列加 1 个字节，
如果表中有一个列（long 字段）宽度超出 250 个字符，则这应加上 1 个字节。所以有：

space_used_per_row=AVG_ROW_len
+3
+Number of columns
+Number of long columns

对于本列，则

space used per row=24
+ 3
+ 3
+ 0
=30 bytes per row

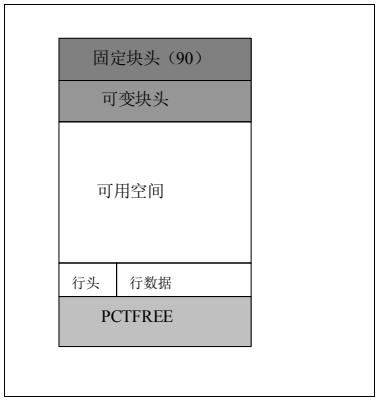
块中可以存放多少行

对于一个块有 1762 字节的自由空间，每行长度为 30 个字节

则该块中可以存放

rows per block =TRUNC(1762 free bybes/30 bytes per row)
=58 row per block

由此可以估算出一个表所需的块数。



块数据存储示意图（块存储图）

2．确定合适的 pctfree

由于 pctfree 值控制了一个数据库块中存储记录的数量，所以该值设置是否合适是至关重要，要了解 pctfree 设置是否合适，可用：

```
analyze table table_name compute statistics;
```

命令来对表进行分析，然后查询 user_tables 数据字典中的记录：

```
select Num_Rows, /*number of rows*/
blocks , /*mumber of block*/
Num_Rows/blocks /* number of blocks per block */
From user_tables
Where table_name= UPPER('table_name');
```

由 Num_Rows/Blocks 可以知道每一个块中行的数量；
模仿用户的实际应用，用 update 更新表中的记录；
用 analyze 对该表再次分析，查询块中行的数量；
如果块的行数量没有变化，一些行没有被移到新的块中去，则说明 pctfree 值是合适的。
如果 Avg_Space 值在 updade 后变大，则说明 pctfree 值还可以减少些。

3．正确设置 pctused 值

可用空间低于 pctused 值（百分比）时，则插入时可以使用刚被删除而空出的空间。它的缺省值是 40%，它表示数据库块中只有 40%的空间被使用。所以最好不要使用缺省值。一般建议将 pctfree 值和 pctused 的总和为 95 比较合适，比如将 pctused 75,则 pctfree 20 。

§9.11.2 索引大小计算

块头占 161 字节
块大小＝2048 bytes
可用空间＝2048－161＝1887bytes
用于更新的空间，假设 pctfree=10，
用于更新的空间＝1887*（pctfree/100）
=1887*0.1＝189 bytes
索引有效使用空间＝1887－189＝1698 bytes
索引列平均长度＝索引列中每一个值平均长度之和
如 table_name 有 column1 和 column2 列用作建立索引，则：

```
select AVG(NVL(vsite(column1),0)) +
    AVG(NVL(vsite(column2),0)) AVG_Row_length
    From table_name
```

行的平均空间（space used per row）：
每一索引列加 1 个字节；
有索引列超出 127 字符的加 1 个字节；
索引头加 8 个字节
space used per row=Avg_Row_Length

+Number of columns
+Number of long columns
+8 header bytes
假设索引到的平均长度为 16 字节，则
Space used per Row=16
+2
+0
+8
+26 bytes per index entry
如果索引是唯一索引：则再加 1 个字节

26＋1＝27 字节 / 每行

每块存放索引行：

对于 1689 字节自由空间的索引
Entries per block =TRUNC(1689 freebytes/27bytes per entry)
=62 Entries per block

§9.11.3 簇表的大小计算

用于管理簇的块头＝110 字节
有效字节 2048－110＝1938 available
用于更新的字节数，假设 pctfree=10 则
1938*(pctfree/100)=1938*0．1＝194 bytes
用于存放簇表的字节：
1938－194＝1744 bytes

有效自由空间：
bytes available=1744 bytes available
-4 bytes
-4*number of table（参加簇的表个数）

比如有两个表参加簇，则
bytes auaildble =1744bytes auailable
-4 bytes
-4*2
=1732 bytes available
每一个 2K 字节的块，可以有 1732 字节用于存储簇项。

每一个单行所需空间:

例子：比如有如下两个表共同组成簇项

```
create cluster emp_unit(dep_no varchar2(10))
storage(initial 10M next 2M)
/
create index emp_unit_idx on cluster emp_unit
/
create table unit_infor
(
dept_no varcharz(10), --单位编号（部门）
dept_name varcharz(30),--部门名称
tot_emp number(5),--职工人数
master varcharz(10),--负责人
tel_no varcharz(15),--电话
... ...
）
cluster emp_unit(dept_no)
/
create unique index unit_idx on unit_infor(dep_no)
storage (intial 1M next 1M)
/
create table emp_infor
(
dept_no uarcharz(10),---部门编号
emp_no uarcharz(10),---职工编号
name uarcharz(10),---姓名
birth_date date,---出生
)
cludter emp_unit(dept_no)
/
```

计算每一个表的非簇关键字的平均长度

```
select AVG(NVL(vsite(dept_name),0))+
    AVG(NVL(vsite(tot_emp),0))+
    AVG(NVL(vsite(master),0))+
    AVG(NVL(vsite(tel_no),0)) AVG_row_len1
From unit_inf;

select AVG(NVL(vsite(emp_no),0))+
    AVG(NVL(vsite(name),0))+
    AVG(NVL(vsite(birth_dater),0)) avg_row_len2
From emp_inf;
```

非簇关键字的平均长度

AVG_row_len1＋AVG_row_len2=Average_row_length

行首信息
簇中的每一行存有行首信息，行首信息包括：
Row header space=4 bytes
＋number of columns
＋number of long columns
行占用空间
平均长度＋行首长度
设 Avg_row_len = 60, Avg_row_len-2=27
space used per row =average row length + row header space
=60+27+7+0
=94 bytes

§9.11.4 位图索引的大小计算

如果建立了位图索引，则 Oracle 自动压缩所生成的位图，位图压缩可以节省空间。对于位图索引大小的估计与普通（B*树）的估计一样，可以采用前面介绍的公式进行计算得到位图的基本大小后在加 5%到 10%即可。

第 10 章　表、索引优化管理

§10.1 表和索引优化管理

```
/* 检查表已分配的次数，如果 count(*) (扩展次数)较大时，说明 next 设得太低。*/
col segment_name for a30
col tablespace_name for a20
col segment_type for a18
ttitle center 扩展次数较高的对象的情况信息
Select tablespace_name, segment_type, segment_name, sum(bytes), count(*) extents
From dba_extents Having count(*)>9
Group by tablespace_name, segment_type, segment_name;

/* count(*) =index 区间的数目，如果该数较大，则原 initial 和 next 设置过低。*/
/* 多数情况下，都希望区间数 count(*) 的结果小于 6 左右。理想的应该是 1 */
col segment_name for a20
select owner , segment_name, count(*)
from DBA_EXTENTS where segment_type='INDEX'
group by segment_name, owner ;
```

/* 在 Oracle8i 里检查表已分配的次数>1，则修改 next 大些。并自动产生脚本。*/

```
select ' alter  table '||table_name ||
' storage (next '||to_char(NEXT_EXTENT)|| ' pctincrease 0); '
 from user_tables ,
(select  segment_name,count(*) from user_segments
   having count(*)>1 group by segment_name ) seg
where seg.segment_name = user_tables.table_name;
```

§10.2 CLUSTER 优化管理

检查簇的存储情况主要看整个 簇 是否分配过多的片，下面查询列出已经分配空间的数目：

```
col segment_name  for a20

select  owner,  segment_name,  count(*)  from   dba_extents
where   segment_type='CLUSTER'
group  by   segment_name,  owner;
```

count(*) 列出 簇 片的数目，如果该数较大，则表明 initial 和 next 太低。

§10.3 碎片整理

产生**碎片的原因：**
删除表是造成表空间碎片的原因之一；

Oracle8i/9i 版本可以用 **Alter table ... MOVE 语句**将表移动到一个新的段或新表空间上，这样可以实现对不合理存储参数进行修改，包括用一般的 ALTER TABLE 不能修改的参数。如：

例 1：通过移动来实现存储参数的修改：

```
Alter table emp MOVE
STORAGE(INITIAL 1m  next 512k minextents 1 maxextents 999  pctincrease 0 );
```

例 2：将那些使用 system 表空间 的对象移动到合适的表空间中：
1)移动前表所使用的表空情况：

```
SQL> select tablespace_name,table_name,initial_extent from user_tables;

TABLESPACE_N       TABLE_NAME  INITIAL_EXTENT
```

```
------------- ------------  -------------
SYSTEM          ABC                65536
SYSTEM         BONUS               65536
SYSTEM         DEPT                65536
SYSTEM         EMP                 65536
SYSTEM         EMP2                65536
SYSTEM         EMP3                65536
SYSTEM         EMP4                65536
USERS          PAY_LST_DET        1048576
SYSTEM         PLAN_TABLE          65536
SYSTEM         SALGRADE            65536
USERS          UNIT_INF           1048576

11 rows selected.
```

2）用**Alter table . . . MOVE** 语句对表进行移动：

```
SQL> alter table emp move tablespace user_data
  2  storage(initial 128k next 128k minextents 1 pctincrease 0);

Table altered.

SQL> alter table dept move tablespace user_data
  2  storage(initial 128k next 128k minextents 1 pctincrease 0);

Table altered.

SQL> alter table BONUS  move tablespace user_data
  2  storage(initial 128k next 128k minextents 1 pctincrease 0);

Table altered.
```

3）移动后的表及表空间的情况：

```
SQL> select tablespace_name,table_name,initial_extent from user_tables;

TABLESPACE_N   ABLE_NAME       INITIAL_EXTENT
------------ --------------- --------------
SYSTEM          ABC                 65536
USER_DATA       BONUS              131072
USER_DATA       DEPT               131072
USER_DATA       EMP                131072
SYSTEM          EMP2                65536
SYSTEM          EMP3                65536
SYSTEM          EMP4                65536
```

USERS	PAY_LST_DET	1048576
SYSTEM	PLAN_TABLE	65536
SYSTEM	SALGRADE	65536
USERS	UNIT_INF	1048576

11 rows selected.

§10.4 行链接和行转移整理

- 行转移

行转移是在对行的修改引起的长度比块中可利用的空 间大时发生。当行转移发生时，行的一部分就转移到另一个新的数据块中。这个新的单元地址存储在原始的单元中。当 ORACLE 读这个行时，它首先读原始单元并找到一些数据和指向另一个块的指针。再在这个块中找到剩下的行的数据。

- 行链接

行链接是在一个行太长以致于不能存放到一个数据块时发生。这导致行被存储在多个数据块中的链中。产生行链接的大都是带有 long，long raw 或者 LOB 类型的数据。

既然链接和行转移都对数据库性能产生影响，那么我们如何知道当前系统是否存在链接和行转移呢？可以用下面命令来分析：

```
ANALYZE TABLE table_name LIST CHAINED into CHAINED_ROWS;
```

要建立 chained_rows 表，需执行 **$ORACLE_HOME/rdbms/admin/utlchain.sql** （或 utlchn1.sql)脚本。上面命令把 ROWID 和数据库中所有链和行转移的表存储到 chined_rows 表中。我们可以直接查询该表的数据即可。
当我们发现有行转移时，可用下面方法来解决：

1） 建立一临时表保存数据：

```
create table tmp_emp as select * from emp
where rowid in ( select rowid from chained_rows
 where table_name ='EMP' );
```

2） 从原表中删除有行转移的行：

```
delete from emp where rowid in
( select head_rowid from chained_rows
   where table_name = 'EMP' );
```

3） 从临时表中把行数据插入到原表中：

```
Insert into emp select * from tmp_emp ;
```

采用上面方法是可以把行转移给压缩了，但有可能产生另一个问题，就是空出来的空间可能永远没有被使用。这又取决于 PCTUSED 参数。所以要避免行转移的发生，要认真研究 **PCTFREE** 和 PCTUSED 参数。

第三部分　ORACLE 应用系统开发优化

第 11 章　诊断与调整工具

下面仅给出诊断工具的简要介绍，它们的详细使用说明参考后面章节。

§11.1 警告日志文件

§11.1.1 警告日志文件管理

- 警告文件所包含的内容和错误信息;
- 经常检查警告文件:

1) 内部错误(ORA-600)和块冲突(ORA-1578或ORA-1498);
2) 监视数据库操纵,如CREATE DATABASE,STARTUP,SHUTDOWN,ARCHIVE LOG及RECOVER,CREATE TABLESPACE等操作;
3) 查看非默认值,即在实例启动时查看初始化参数;

- 删除或裁剪掉内容.如:从参数文件的的BACKGROUD_DUMP_DEST参数得到ALERT.LOG文件的位置.

§11.1.2 参考警告日志文件调整

- 检查启动和结束时间:如果设置CHECKPOINTS_TO_ALERT=TRUE的话;
- 未完成的检查点;
- 归档性能;
- 失败恢复启动与完成时间

§11.2 后台进程与跟踪文件

- Oracle 服务器将出现各种错误都将其信息卸载到跟踪文件中;
- Oracle 支持工程师使用这些跟踪文件来诊断和解决问题;
- 这些文件一般不包含调整信息.

§11.3 用户跟踪文件

- 可在会话级或实例级来设置服务器进程跟踪;

1) 实例级的跟踪：
初始化文件设置 SQL_TRACE=TRUE;

2) 会话级的跟踪：
SQL>execute dbms_system.set_sql_trace_in_session(8,12,true);

这里的 8 为 sid (会话标识), 12 为 Serial#(会话序列编号) 。
DBMS——SYSTEM 包由 catproc.sql 建立：
$ORACLE_HOME/rdbms/admin
%ORACLE_HOME\rdbms\admin

- 用户跟踪包含在会话中 SQL 语句的统计;

SQL>ALTER SESSION SET sql_trace=TRUE;

- 用户跟踪文件在服务器进程来建立;
- 用户跟踪也可由“后台控制跟踪”或 SET EVENT 来建立。

§11.4 动态性能视图

- DBA_TABLES
- DBA_TAB_COLUMNS
- DBA_CLUSTERS
- DBA_INDEXES
- INDEX_STATS
- DBA_TAB_HISTOGRAMS
- INDEX_HISTOGRAM

§11.5 UTLBSTAT 和 UTLESTAT

§11.6 STATSPACK

§11.7 Oracle 等待事件

§11.8 Oracle 诊断和调整包

第 12 章　优化 SQL 语句

系统的运行性能如何，很大一部分是与编程人员有关。本章介绍基于开销的 SQL 语句程序设计，包括：

- SQL 语句优化方法
- 优化目标
- 最佳实例
- SQL 语句优化技巧
- 使用 EXISTS 和 IN

§12.1 SQL 语句的优化方法

在提高SQL语句效率上，可以考虑下面几个方面：

- 重新构造语句；
- 修改或禁止触发器；
- 重新构造数据；
- 及时统计CBO所用信息。

§12.1.1 重新构造语句

在重构索引之后，你可以试着重构语句。即重写那些效率差的语句。如果你对语句的意图了解的话，就很容易对语句进行修改。

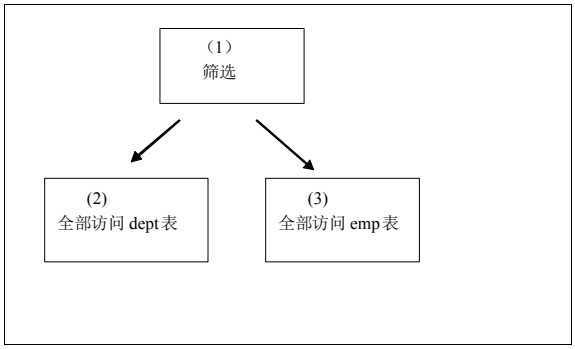
1.考虑可选择性的SQL语法:

由于SQL语言的灵活性，应用中多个语句更适合应用的需要。虽然两个SQL语句可以产生同样的结果，但Oracle处理一个语句比处理多个组合而成的语句更快。你可以使用EXPLAIN PLAN语句的结果进行比较，从而确定哪个语句更有效。

看一下的特殊语句：

```
SELECT dname, deptno
FROM dept
WHERE deptno NOT IN
(SELECT deptno FROM emp  );
```

实际是有两个语句组成，两个语句都是从dept 表中取数，执行的是同一个功能。两个语句返回在emp表中没有的所有dept行，每个语句都搜索emp子查询。假定在emp表中建立了索引为deptno_index的话，执行计划是：

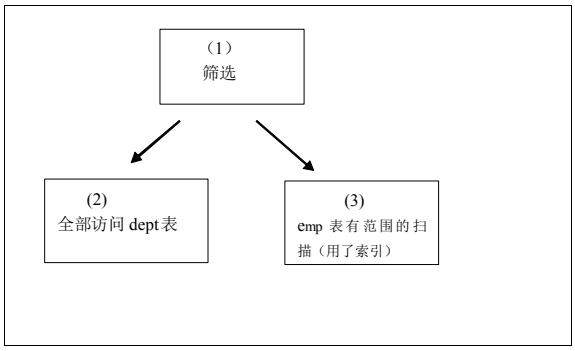


步骤3预示着Oracle要对emp表进行全表扫描。这些全表扫描可能是时间的操作。这里Oracle根本不使用索引，因为没有where 子句。

然而，下面的语句使用索引也能查询到同样的行：

```
SELECT dname, deptno
FROM dept
WHERE NOT EXISTS
(SELECT deptno
FROM emp
WHERE dept.deptno = emp.deptno);
```

这里的执行计划是全表扫描和索引扫描。



由于在查询emp表时使用了where 子句，所以它就使用了索引。它的索引使用在时间上要比全表扫描要省许多。而且，当第一次查到结果后，第二个SQL语句就比第一个语句快多了。
有此看出，如果你的应用中使用了 NOT IN 运算的话，请你将其换成 NOT EXISTS 运算。

2.使用 AND 和 = 谓词

随时都可以用等同连接(equijoin)，正常情况下，等同连接的执行在非转换列上是很容易进行调整的。

3．选择有利的连接顺序

连接顺序对执行很有意义。SQL语句调整的主要目标是避免执行不必要的工作，这样就考虑三个规则：

- 如果通过索引可以得到行的话，就要避免全表扫描；
- 如果你能使用的索引取到100行的话，就避免使用取到 10,000 行的索引；
- 选择连接要注意选择连接较少行的表。

下面的例子说明连接顺序的效率：

```
SELECT info
FROM taba a, tabb b, tabc c
WHERE a.acol BETWEEN :alow AND :ahigh
AND b.bcol BETWEEN :blow AND :bhigh
AND c.ccol BETWEEN :clow AND :chigh
AND a.key1 = b.key1
AND a.key2 = c.key2;
```

1). 选择驱动表和驱动索引。

前三个条件是对单个表进行筛选，后两个条件是连接条件。筛选条件主要是表和索引的选择。

2).选择正确的索引

当了解所要驱动的表后，就要选择有效的索引。否则就可能出现全表扫描。

3).选择最好的连接顺序

4.使用非转换的列值

可以在where 子句中使用非转换的列值，如：

```
WHERE a.order_no = b.order_no
```

这样的使用比下面的用法更好：

```
WHERE TO_NUMBER (SUBSTR(a.order_no, instr(b.order_no, '.') - 1))
= TO_NUMBER (SUBSTR(a.order_no, instr(b.order_no, '.') - 1))
```

建议不要在where 后使用SQL 函数。

5.避免混合类型表达式

避免使用混合的类型表达式和警惕不明确的类型转换。当你在VARCHAR2类型的索引列charcol列上使用索引时，可能将语句写成：

```
AND charcol = <numexpr>
```

这里的 numxpr 是数字类型表达式。Oracle会将该表达式转换成：

AND TO_NUMBER(charcol) = numexpr

这样就会有下面的结果：

- 任何在列上使用表达式，比如函数，都会引起优化被忽略。甚至是唯一索引也一样。
- 如果系统在处理字符串时不作转换，则要返回错误。

要避免这样的情况发生，接要进行转换，如：

AND charcol = TO_CHAR(<numexpr>)

作为选择，可以使所有类型直接转换，语句为：

numcol = charexpr

在numcol 上允许使用索引，因为默认的转换总是 字符到数字。

6.特殊的值写单独的语句

SQL语句不是过程语言，用一条语句来完成多个不同的事情不是一个好的主意。所以要完成两件工作，写两条语句比写一条要好。例如：

```
SELECT info  FROM tables
WHERE ...
AND somecolumn BETWEEN DECODE(:loval, 'ALL', somecolumn, :loval)
AND DECODE(:hival, 'ALL', somecolumn, :hival);
```

这样写，数据库不会使用索引，因为这里用了BETWEEN子句。
如果要使用索引，只要给出:loval 和:hival即可。然后重写逻辑语句：

```
SELECT /* change this half of union all if other half changes */ info
FROM tables
WHERE ...
AND somecolumn BETWEEN :loval AND :hival
AND (:hival != 'ALL' AND :loval != 'ALL')
UNION ALL
SELECT /* Change this half of union all if other half changes. */ info
FROM tables
WHERE ...
AND (:hival = 'ALL' OR :loval = 'ALL');
```

7.使用提示控制访问路径

使用优化器提示，即使用 /*+ORDERED */ 来控制访问路径。这样的方法要比传统的技术（比如默认要访问索引）要好。例如：

```
SELECT /*+ FULL(emp) */ e.ename
FROM scott.emp e
WHERE e.job = 'CLERK';


SELECT e.ename FROM emp e
WHERE e.job || '' = 'CLERK';
```

8.使用IN和NOT IN 要注意

使用 (NOT)EXISTS 是很有用。注意(NOT) EXISTS 不等于 NOT IN。

例1．反连接查询：

```
SELECT * FROM employees
WHERE department_id NOT IN
(SELECT department_id FROM departments
WHERE loc = 'HEADQUARTERS');
```

例2．存在查询：

```
SELECT * FROM departments
WHERE EXISTS
(SELECT * FROM employees
WHERE departments.employee_id = employees.employee_id
AND employees.salary > 25000);
```

9．在应用中嵌如数据列表要注意

数据列表值总是以单引号引起，如：

```
WHERE transport IN ('BMW', 'CITROEN', 'FORD', HONDA')
```

10.减少数据库的调用次数

使用适当地以单条语句 INSERT、UPDATE 或DELETE …RETURNING来查询和修改数据时，可以减少数据库的调用次数而提高性能。见“Oracle8i SQLReference Manual”。

11.谨慎使用管理视图

当对视图进行 连接视图、执行视图的外连接、重复利用视图，要谨慎地使用。

1)连接视图
在访问视图时可以通过SGA区的SQL区来达到减少解释的开销。但是Oracle建议不要某种类似联合或连接的视图操作。看下面的操作：

```
CREATE VIEW dx(deptno, dname, totsal)
AS SELECT d.deptno, d.dname, e.sum(sal)
FROM emp e, dept d
WHERE e.deptno = d.deptno
GROUP BY deptno, dname ;

SELECT *
FROM dx
WHERE deptno=10;
```

在上例中，不好的就是用了GROUP BY，而且有在查询中加条件（连接操作）。

2)子查询嵌套

可以通过设置参数 UNNEST_SUBQUERY=TRUE 来启用子查询嵌套操作。但是此参数不是基于开销的。在默认下是不需要设置的。可以用NO_UNNEST来禁止子查询嵌套操作。详细见“Oracle8i SQL”中UNNEST和NO_UNNEST参数。

3)执行视图的外连接

对于几个表视图的外连接，最好不要构造过于复杂的外连接。要尽量简单。比如可以建立下面视图：

```
CREATE VIEW empdept (empno, deptno, ename, dname)
AS SELECT e.empno, e.deptno, e.ename, d.dname
FROM dept d, emp e
WHERE e.deptno = d.deptno(+);
```

用简单的外连接查询来实现操作：

```
SELECT e.ename, d.loc
FROM dept d, empdept e
WHERE d.deptno = e.deptno(+)
AND d.deptno = 20;
```

4)不能重复利用的视图

要了解编写视图就是为了让另外的程序可以直接对其进行操作后得到需要的结果。而不是别的目的。如：

SELECT dname from dx where deptno = 10 ;
这里dx 是一个视图，这样的用法不如直接从原始表中进行查询有效。要通过dx 视图反而效率低下。

§12.1.2 调整或使触发器无效

使用触发器会消耗系统大量资源，如果系统使用了过多的触发器，则需要将部分触发器置为无效。

§12.1.3 重组数据

在重建索引和重建语句后，可以再考虑重组数据：

- 引入起源值，避免GROUP BY 从句;
- 实现丢失实体（missing entity）和交叉 (intersection)表
- 减少网络负载、迁移、复制及分区数据

作为数据分布策略的主要目的是定位每个数据的属性。比如使网络行程最小化。如果当前行程数过大，则移动（迁移）该数据是一个自然的解决方案。
然而，数据的非单一位置（数据放在多个地方）可以减少网络加载到接收层的时间或减少信息传输延迟。在这种情况下，可考虑支持多个拷贝（复制数据）或在不同的地点保持有数据的不同部分(对于本地就相当于丢失实体)。然而，要进行分布式查询，需要有效的代码与PL/SQL存储过程或用户接口代码结合一起使用。当考虑交叉连接时，你可以：1)从远程节点拿来数据并与本地执行连接；2)或将数据送到远程节点，再执行远程连接。这样的选择，需要你考虑不同节点数据卷的关系。

§12.2 优化目标

数据库开发人员都知道，结构查询语言（SQL）是用于执行数据库的操作。但要很好的进行使用，还需要注意许多地方：

- 调整序列SQL语句
- 调整并行执行
- 调整OLTP应用

一般来说，这样考虑应用的类型：

- 数据仓库操作处理很高的数据量，具有很高的相关性。
- OLTP应用有很大的用户数，并且它们都与一系列的操作有关。

§12.2.1 优化序列 SQL 语句

在孤立的环境下调整一个SQL语句的目的是：在执行中使资源的使用减少到最小。你可以选择不用修改应用而使用不同的SQL语法来得出语句的执行代价。使用EXPLAIN PLAN语句，比较各个语句的执行计划及代价。另外要提示的是，在调整完SQL 语句后要实际的运行一下应用才能了解到语句的效果。

§12.2.2 优化并行执行

调整并行执行的目的是：最大地发挥硬件的能力。如果你有一个高性能的系统，有高优先的SQL语句在运行，则并行语句就可以使用所有有效的资源。Oracle可以执行的下面的并行：

- 并行查询；
- 并行DML(包括 INSERT, UPDATE, DELETE; APPEND提示，并行索引扫描)；
- 并行 DDL；
- 并行恢复；
- 并行加载；
- 并行传播（复制）；
- 高数据量的行处理。

如果处理的行数较多，可以考虑将其进行分割(split)，使得单个进程只处理部分的行。关于这里的相关信息，请参考“Oracle8i数据仓库指南”中的下面内容：

1）设置并行度和使能多用户
2）调整并行执行参数
3）建立并行索引
4）分区索引扫描
5）使用大块插入、更新和删除

如果具备并行环境，可考虑下面的内容：

- 对称多处理器（SMP), 集群或强大的并行系统；
- 有效的I/O带宽；
- 低利用的或闲置的CPU（如CPU使用小于30%）；
- 对附加的内存无效，如分类、哈西索引及I/O缓冲区等。

如果你的系统缺少以上这些特点，则并行可能不会有多大改善。

§12.2.3 调整 OLTP 应用

调整OLTP应用大多涉及到调整SQL语句。建议考虑两种设计方案：

使用SQL 与共享 PL/SQL

将分析减少到最小，就要在SQL语句中使用绑定变量的方法老实现。这样，所有的用户都可以使用相同的SQL语句。

事务模式

如果性能已经达到极限并且这些用户愿意设计后应用的话，有经验的用户可以采用分散事务的方法来实现性能的改善。

触发器

如果过度的使用触发器的话就会影响系统的性能。所以要用ALTER TRIGGER 语句来设置触发器为脱机的状态。

§12.3 一般优化

§12.3.1 避免基于规则优化器技术

传统的基于规则的技术包括：

- 不用索引：

– col+0 或 col || ‘‘
– 在列外套函数，如 NVL (col, -999) 或TO_NUMBER等。

- 在FROM子句中工作表的次序

基于CBO选择最讲究连接顺序。所以要选择好主要工作表的次序。

§12.3.2 索引代价

如果选择的索引的代价过高，即可选择性太少=满足的行太多，可考虑采用全表扫描方法。

例。下面例子中有 employee_num 索引，但由于代价高而直接采用全表扫描：

```
SELECT employee_num, full_name NAME, employee_id
FROM mtl_employees_current_view
WHERE (employee_num LIKE '20%') AND
(organization_id = :1)
ORDER BY employee_num;
```

§12.3.3 分析统计数据

要分析的统计数据包括：

- 列统计
- 数据偏移
- 表统计
- 索引统计
- 分区统计

下面是基于CBO的例子：

```
SELECT item.expenditure_item_id
FROM pa_tasks t,
pa_expenditures exp,
pa_expenditure_types etype,
pa_expenditure_items item
WHERE
TRUNC(exp.expenditure_ending_date)<=TRUNC(NVL(TO_DATE(:b0),
exp.expenditure_ending_date))
AND exp.expenditure_status_code||''='APPROVED'
AND exp.expenditure_group=NVL(:b1,exp.expenditure_group)
AND exp.expenditure_id=item.expenditure_id
AND (NVL(item.request_id,(:b2+1))<>:b2 OR item.cost_dist_rejection_code IS
NULL )
AND item.cost_distributed_flag='N' and t.task_id=item.task_id
AND t.project_id=DECODE(:b4,0,T.project_id,:b4)
AND item.expenditure_type=etype.expenditure_type
AND etype.system_linkage_function||''=:b6
ORDER BY item.expenditure_item_date;

COST DISTRIBUTED FLAG
C          7
N          80,251
Y          16,534,822
```

基于规则的规划（Rule Plan）的情况：

Cost= SELECT STATEMENT
COUNT(*)
Cost= SORT ORDER BY
==========
Cost= NESTED LOOPS
Cost= NESTED LOOPS
Cost= NESTED LOOPS
Cost= TABLE ACCESS BY INDEX ROWID PA_EXPENDITURE_ITEMS_ALL
Cost= INDEX RANGE SCAN PA_EXPENDITURE_ITEMS_N3: COST_DISTRIBUTED_
FLAG
Cost= TABLE ACCESS BY INDEX ROWID PA_EXPENDITURE_TYPES
Cost= INDEX UNIQUE SCAN PA_EXPENDITURE_TYPES_U1: EXPENDITURE_TYPE
Cost= TABLE ACCESS BY INDEX ROWID PA_EXPENDITURES_ALL
Cost= INDEX UNIQUE SCAN PA_EXPENDITURES_U1: EXPENDITURE_ID
Cost= TABLE ACCESS BY INDEX ROWID PA_TASKS
Cost= INDEX UNIQUE SCAN PA_TASKS_U1: TASK_ID

基于开销的规划（CBO Plan-默认）的情况：

Cost=6503 SELECT STATEMENT
Cost=6503 SORT ORDER BY
Cost=6489 NESTED LOOPS
Cost=6487 NESTED LOOPS
Cost=6478 MERGE JOIN CARTESIAN
Cost=6477 TABLE ACCESS FULL PA_EXPENDITURES_ALL
Cost=1 SORT JOIN
Cost=1 TABLE ACCESS FULL PA_EXPENDITURE_TYPES
Cost=9 TABLE ACCESS BY INDEX ROWID PA_EXPENDITURE_ITEMS_ALL
Cost=4 INDEX RANGE SCAN PA_EXPENDITURE_ITEMS_N1: EXPENDITURE_ID
Cost=2 TABLE ACCESS BY INDEX ROWID PA_TASKS
Cost=1 INDEX UNIQUE SCAN PA_TASKS_U1: TASK_ID

使用提示强制使用规则的情况：

对于同样的要求，利用强制规则要比CBO要高出许多：
This illustrates that the cost of the RBO plan is significantly higher than that of the
the default CBO generated plan.
Cost=592532 SELECT STATEMENT
Cost=592532 SORT ORDER BY
Cost=592518 NESTED LOOPS
Cost=592516 NESTED LOOPS
Cost=587506 NESTED LOOPS
Cost=504831 TABLE ACCESS BY INDEX ROWID PA_EXPENDITURE_ITEMS_ALL

Cost=32573 INDEX RANGE SCAN PA_EXPENDITURE_ITEMS_N3:
Cost=1 TABLE ACCESS BY INDEX ROWID PA_EXPENDITURE_TYPES
Cost= INDEX UNIQUE SCAN PA_EXPENDITURE_TYPES_U1:
Cost=2 TABLE ACCESS BY INDEX ROWID PA_EXPENDITURES_ALL
Cost=1 INDEX UNIQUE SCAN PA_EXPENDITURES_U1:
Cost=2 TABLE ACCESS BY INDEX ROWID PA_TASKS
Cost=1 INDEX UNIQUE SCAN PA_TASKS_U1:

重写SQL语句：

为了避免全表扫描，可以用更有效的可选择性过滤器来实现重写SQL语句。下面的NVL函数是避免索引的使用：

```
SELECT item.expenditure_item_id
FROM pa_tasks t,
pa_expenditures exp,
pa_expenditure_types etype,
pa_expenditure_items item
WHERE
TRUNC(exp.expenditure_ending_date)<=TRUNC(NVL(TO_DATE(:b0),
exp.expenditure_ending_date))
AND exp.expenditure_status_code||''='APPROVED'
AND exp.expenditure_group=:b1
AND exp.expenditure_id=item.expenditure_id
AND (NVL(item.request_id,(:b2+1))<>:b2 OR item.cost_dist_rejection_code IS
NULL)
AND item.cost_distributed_flag='N' and t.task_id=item.task_id
AND t.project_id=DECODE(:b4,0,t.project_id,:b4)
AND item.expenditure_type=etype.expenditure_type
AND etype.system_linkage_function||''=:b6
ORDER BY item.expenditure_item_date
```

新的 CBO 规划：

Cost=32 SELECT STATEMENT
Cost=32 SORT ORDER BY
Cost=18 NESTED LOOPS
Cost=16 NESTED LOOPS
Cost=7 MERGE JOIN CARTESIAN
Cost=1 TABLE ACCESS FULL PA_EXPENDITURE_TYPES
Cost=6 SORT JOIN
Cost=6 TABLE ACCESS BY INDEX ROWID PA_EXPENDITURES_ALL
Cost=2 INDEX RANGE SCAN PA_EXPENDITURES_N3: EXPENDITURE_GROUP
Cost=9 TABLE ACCESS BY INDEX ROWID PA_EXPENDITURE_ITEMS_ALL

Cost=4 INDEX RANGE SCAN PA_EXPENDITURE_ITEMS_N1: EXPENDITURE_ID
Cost=2 TABLE ACCESS BY INDEX ROWID PA_TASKS
Cost=1 INDEX UNIQUE SCAN PA_TASKS_U1: TASK_ID

§12.3.4 避免复杂的表达式

在写SQL语句时，要尽量避免复杂的表达式，下面的例子的表达式就有点复杂：

- col1 = NVL (:b1,col1)
- NVL (col1,-999) = ….
- TO_DATE(), TO_NUMBER() 等。

例子:：

```
SELECT employee_num, full_name NAME, employee_id
FROM mtl_employees_current_view
WHERE (employee_num = NVL (:b1,employee_num)) AND (organization_id=:1)
ORDER BY employee_num;
```

§12.3.5 处理复杂的逻辑

复杂的逻辑处理建议采用Oracle的Forms 触发器、PL/SQL逻辑或C语言编码来处理。如：

```
SELECT *
FROM ar_addresses_v
WHERE (customer_id=:1)
==========
AR_ADDRESSES_V:
SELECT *
FROM AR_LOOKUPS L_CAT,
FND_TERRITORIES_VL TERR,
FND_LANGUAGES_VL LANG,
RA_SITE_USES SU_SHIP,
RA_SITE_USES SU_STMT,
RA_SITE_USES SU_DUN,
RA_SITE_USES SU_LEGAL,
RA_SITE_USES SU_BILL,
RA_SITE_USES SU_MARKET,
RA_ADDRESSES ADDR
```

对于这样的复杂语句，可以：

- 重写SQL语句和消除6个表的连接。
- 在Forms 加后查询触发器。
- 减少处理行数。

§12.3.6 一般的 SQL 语句优化

能简化的表达式就该简化：

1) 常数

sal > 24000/12
sal > 2000 -- 最优
sal * 12 > 24000

2)like
ename like 'SMITH'
ename = 'SMITH'
变长时第 2 语句最优；定长时第 1 最优。

1) IN
Ename in ('SIMTH','KING','JONES')
Ename='SIMTH' OR ename ='KING' or ename='JONES'
效率相同。

2) ANY 和 SOME
Sal > any (:first_sal, :second_sal)
Sal > :first_sal or sal > :second_sal
效率相同。

X > any (select sal from emp where job='ANALYST')
Exists (select sal from emp where job='ANALYST' and x> sal)
效率相同。

3) ALL
Sal > all (:first_sal, :second_sal)
Sal > :first_sal and sal > :second_sal
效率相同。

X > all (select sal from emp where deptno=10)
NOT (x <= ANY (select sal from emp where deptno=10))
效率相同。

4) Between

Sal Between 2000 and 3000
Sal >= 2000 and sal <= 3000
效率相同。

NOT deptno = (select deptno from emp where ename='TAYLOR')
Deptno <> (select deptno from emp where ename='TAYLOR')
效率相同。

§12.4 SQL 语句优化技巧

SQL 语句的优化的技巧很多，下面是一些常用的技巧列表：

SQL 优化技巧	注释
以更快的工作完成系统的工作	选择行数最少为目标，可以使 SQL 解释和操作更少。
分析连接的层次	逐一对连接进行分析，搞清具体的使用。
检查基础视图	如果用户访问了视图或与视图进行过连接，则应对视图进行彻底的检查，看视图是否还需要优化。
对于小表，可以进行全表扫描	通过全表扫描，可以使问题解决更加清楚。对于小表或可选择性差的索引，全表扫描更有效。
逐条检查执行规划	索引访问和嵌套连接（NL）可能并不是最优。对于特地的连接类型，查询不要返回太多的行。
对于运行时间较长的查询，执行下面数学运算：如： ● 某个运算希望 3 分钟完成； ● 查询需要将表 so_lines 和表 so_headers 连接起来。	验证下列条件： ● so_headers 的选择性为 5%; ● so_lines 的选择性为 15%; ● so_headers = 1GB,so_lines=25GB ● 数据工作集（结果集）=3.04GB ● 所需吞感吐量=22MB/每秒 可看出，速度取决于系统的配置，要求 3 分钟查出结果的期望太高了。
监视磁盘读和缓冲区的获取操作连接	见本章的“磁盘读和缓冲区获取”
连接： ● 检查外连接 ● 以子查询替代连接	见本章“选择有利的连接次序”
选择 EXISTS 或 IN	见后面“使用 EXISTS 和 IN”章节。
判定式失败	见“判定式崩溃”

§12.4.1 对所有 SQL 语句执行 EXPLAIN_PLAN

为了使每条 SQL 语句能达到优化，建议用户对所有的 SQL 语句执行 EXPLAIN_PLAN，并查看输出结果，然后调整相应的语句。如：

```
SQL> set autotrace on explain;
SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON


Execution Plan
----------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (FULL) OF 'DEPT'
```

§12.4.2 磁盘读和缓冲区获取

可以在命令方式下用 SET autotrace on stat 来设置 磁盘读和缓冲区获取的自动统计显示，如：

```
SQL> set autotrace on stat;
SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON


Statistics
----------------------------------------------------
          0  recursive calls
         12  db block gets
          6  consistent gets
          0  physical reads
          0  redo size
        678  bytes sent via SQL*Net to client
        424  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0    sorts (disk)
          4  rows processed
```

可从显示中找“一致性读（consistent read）”或“物理读（hpysical read）”的数量。如果这两值较高，表示查询的开销较大。比如，用户希望返回的记录行数为 1,000 行，但是一致性读的数目为 1,000,000；而物理读的数目为 10,000 ，则查询语句需要进行优化。

§12.4.3 判定式崩溃

在查询语句中，如果列包含有许多个绑定变量时，就可能发生判定式崩溃。例如，对某列使用 col=DECODE(:b1, ‘ ‘, :b3, col) 判定时，就表示：如果绑定变量 b1 为 null，则用绑定变量 b3 代替。否则，表达式就采用 col = col 。 这样的语句将会阻止系统优化程序使用 col 列的索引。当然也就产生全表扫描，从而影响性能。

例1.可能导致判定崩溃的例子：

```
SELECT delivery_id, planned_departure_id,
organization_id, status_code
FROM wsh_deliveries
WHERE delivery_id = NVL(:b1,delivery_id) AND
name = DECODE(:b1,'',:b3, NAME)
ORDER BY UPPER(HRE.full_name) ;
```

这样的语句，通过EXPLAIN PLAN解释计划，从而显示得到下面的结果：

```
Cost=2090 SELECT STATEMENT
Cost=2090 TABLE ACCESS FULL WSH_DELIVERIES
```

这个结果表明，选择了2090 条记录，全表扫描为2090条记录，说明这个语句不能使用索引。那么如何修改该语句才能达到使索引呢？我们可用Oracle提供的语句UNION关键字来实现。这个语句该写成：

```
SELECT delivery_id, planned_departure_id,
organization_id, status_code
FROM wsh_deliveries
WHERE delivery_id = :b1 AND (:b1 IS NOT NULL)
UNION
SELECT delivery_id, planned_departure_id,
organization_id, status_code
FROM wsh_deliveries
WHERE name = :b2 AND (:b1 is null) ;
```

这样就是两条语句，如果提供了绑定变量，系统就可执行UNION的第2条语句，即在任何一种情况下，UNION关键字都会使两条语句使用delivery_id 列或名称列所具有的索引列。这个语

句可用EXPLAIN PLAN解释计划显示，从而得到下面的结果：

Cost=34 SELECT STATEMENT
Cost=34 SORT UNIQUE
Cost= UNION-ALL
Cost= FILTER
Cost=3 TABLE ACCESS BY INDEX ROWID WSH_DELIVERIES
Cost=2 INDEX UNIQUE SCAN WSH_DELIVERIES_U1: DELIVERY_ID
Cost= FILTER
Cost=3 TABLE ACCESS BY INDEX ROWID WSH_DELIVERIES
Cost=2 INDEX UNIQUE SCAN WSH_DELIVERIES_U2: NAME

从显示结果来看，系统不再进行全表扫描了。所以语句的执行效率也就提高了。

§12.5 使用 EXISTS 和 IN

在 SQL 语句中，许多语句的可以完成同样的操作结果，但是它们的执行效率不尽相同。下面给出 EXISTS 和 IN 语句的用法说明。

在 SELECT 语句中使用EXISTS:

```
SELECT COUNT(*)
FROM so_picking_lines_all pl
WHERE (EXISTS (SELECT pld.picking_line_id
FROM so_picking_line_details pld
WHERE (pld.picking_line_id=pl.picking_line_id AND
pld.delivery_id=:b1))
AND nvl(PL.SHIPPED_QUANTITY,0)>0)
```

执行规划为：
Cost=97740 SELECT STATEMENT
Cost= SORT AGGREGATE
Cost= FILTER
Cost=97740 TABLE ACCESS FULL SO_PICKING_LINES_ALL
Cost=4 TABLE ACCESS BY INDEX ROWID SO_PICKING_LINE_DETAILS
Cost=3 INDEX RANGE SCAN SO_PICKING_LINE_DETAILS_N3:

这样的查询由于没有外部查询的选择标准，所以将导致全表扫描，由此看出用IN要好些。

在嵌套循环连接 SELECT 中使用IN:

```
SELECT COUNT(*)
FROM so_picking_lines_all pl
WHERE pl.picking_line_id in (SELECT pld.picking_line_id
FROM so_picking_line_details pld
WHERE pld.delivery_id=:b1)
AND PL.SHIPPED_QUANTITY>0
```

执行规划为：

```
Cost=265 SELECT STATEMENT
Cost= SORT AGGREGATE
Cost=265 NESTED LOOPS
Cost=19 VIEW
Cost=19 SORT UNIQUE
Cost=4 TABLE ACCESS BY INDEX ROWID SO_PICKING_LINE_DETAILS
Cost=3 INDEX RANGE SCAN SO_PICKING_LINE_DETAILS_N3:
Cost=2 TABLE ACCESS BY INDEX ROWID SO_PICKING_LINES_ALL
Cost=1 INDEX UNIQUE SCAN SO_PICKING_LINES_U1:
```

This is another example where IN is more appropriate than EXISTS.

在UPDATE 语句中使用EXISTS:

```
UPDATE so_sales_credits_interface sc
SET request_id=:b0
WHERE request_id IS NULL AND error_flag IS NULL AND
interface_status IS NULL AND
EXISTS (SELECT NULL
FROM so_headers_interface i
WHERE sc.original_system_reference=i.original_system_reference AND
sc.order_source_id=i.order_source_id AND i.request_id=:b0)
```

执行规划为：

```
Cost=1459 UPDATE STATEMENT
Cost= UPDATE SO_SALES_CREDITS_INTERFACE
Cost= FILTER
Cost=1459 TABLE ACCESS FULL SO_SALES_CREDITS_INTERFACE
Cost=2 TABLE ACCESS BY INDEX ROWID SO_HEADERS_INTERFACE_ALL
Cost=1 INDEX UNIQUE SCAN SO_HEADERS_INTERFACE_U1:
```

这样的查询由于没有外部查询的选择标准，所以将导致全表扫描，由此看出用IN要好些。

§12.6 分离事务（Discrete Transactions）

在非分布环境下可以用 BEGIN_DISCRETE_TRANSACTION 过程实现性能改善。

何时使用分离技术 在流线型事务中使用分离对事务而言是非常有用的。

1）只修改几个数据块
2）每次事务中不改变几个数据块
3）在很长的查询中不作修改
4）修改后不需要查看数据新值
5）不修改含 long 值的表

可以考虑在设计时使用约束来实现分离事务

分离事务如何工作

1）参数 DISCRETE_TRANSACTIONS_ENABLED TRUE
2）使用 BEGIN-DISCRETE_TRANSACTION 语句

分离事务例子

```
Create procedure checkout(bookno in number(10),
Status out varchar(5))
As
DeclareTol_books number(3);
Checked_out     number(3);
Res             number(3);

Begin
Dbms_transaction.begin_discrete transaction;

For I In 1 .. 2 loop
  Begin
     Select total, num_out, num_res
     Into tot_books, checked_out, res
     From books
     Where book_num=bookno
     For update;

If res >= ( tot_books - checked_out ) then status := 'res ';
Else update books set num_out=check_out+1
  Where book_num=bookno;
Status := 'AVAIL';
Endif ;
Commit;
Exit;
```

```
Exception
When  dbms_transaction.discrete_transaction_failed
Then  rollback;
End;
End  loop;
End ;
```

§12.7 测试 SQL 语句性能

Oracle 的 SQL TRACE 实用工具和 TKPROF 格式化程序是两种基本的性能诊断工具。他们可以在服务器端运行，用于帮助监视和优化。常用于测试语句性能的方法有：TKPROF 实用程序、EXPLAIN PLAN 及 AUTOTRACE 三种。

§12.7.1 SQL_Trace 实用工具

当设置 SQL_Trace 为有效，则系统对每条 SQL 语句的执行情况提供：

- 解析、执行、取数据的计数；
- CPU 时间和占用时间；
- 物理读和逻辑读；
- 处理行数目；
- 所解析的用户名；
- 每次提交和回滚的情况。

在初始化文件中设置跟踪参数见另外章节。

§12.7.2 设置跟踪初始化参数

要实现对 Oracle 系统使用的跟踪，先要 INITsid.ORA 文件中设置跟踪参数。
1.先检查：TIMED_STATISTICS、MAX_DUMP_FILE_SIZE、USER_DUMP_DEST。
2.给出跟踪的语句（方法）；
3.兼容的版本足够高；
4.设置所产生的跟踪文件可以被所有用户读写。

§12.7.3 启用 SQL_Trace 实用工具

1．只在 SQL 语句内启用 SQL_TRACE:

SQL>Alter session SQL_TRACE=TRUE;
或
SQL>Alter session SQL-TRACE=FALSE;

警告：由于 SQL_TRACE 实用程序会增加系统的开销，建议用完后及时设置为 FALSE。另外，如果将整个系统都进行跟踪的话，则在 INITsid.ORA 中设置 sql_trace=true 。这样会使系统付出更大的代价。
注意：在 INITsid.ORA 中没有设置 sql_trace=true 时，可以用 alter system set sql_trace=true，如果在 INITsid.ORA 中已经设置 sql_trace=true 时或在将其注释掉，都不能再用 alter system set sql_trace=true，而只能用 SQL>alter session set sql_trace=true 。

2．在整个实例内启用 SQL_TRACE:

如果数据库实例内启用 SQL_Trace，则需在初始化文件加 sql_trace=true 。也可以在命令方式来设置实例内启用 SQL-Trace， 如：
SQL>ALTER SYSTEM SET SQL_TRACE=TRUE;

如果不需要实例内启用 SQL-Trace，则：
SQL>ALTER SYSTEM SET SQL_TRACE=FALSE;

关于在 SQL>命令方式下使用 **ALTER SYSTEM SET SQL_TRACE=TRUE** 或 ALTER SYSTEM SET SQL_TRACE=FALSE 的问题。Oracle8i 的文档：

Oracle8 i
Designing and Tuning for Performance
Release 2 (8.1.6)
December 1999
Part No. A76992-01
中 P6-5 页和 P6-6 中的下面内容（拷贝自原稿）：

To enable the SQL trace facility for your *instanc*e, set the value of the SQL_TRACE initialization parameter to true. Statistics are collected for all sessions.

ALTER SYSTEM SET SQL_TRACE = true;
After the SQL trace facility has been enabled for the instance, you can disable it for the instance by entering:
ALTER SYSTEM SET SQL_TRACE = false;

可能有错。
因为当用 ALTER SYSTEM SET SQL_TRACE = true; 命令时，系统提示：

ORA-02095:Specified initialization parameter cannot be modified.
Cause : Specified initialization parameter cannot be moditied
Action:Check the Oracle8i Administrator Guids for information about under scope the parameter may be modified.

1）因为在Oracle8i Reference 中只提到ALTER SESSION set sql_trace=true 而没有提到Alter system set sql_trace=true 这样的用法。

2）另外在Oracle8i SQL Reference 中也只提到用Alter session 语句（**7-126** SQL Reference）：
Enabling SQL Trace Example To enable the SQL trace facility for your session, issue the following statement:

ALTER SESSION SET SQL_TRACE = TRUE;

§12.7.4 用 TKPROF 格式化跟踪文件

TKPROF 将跟踪文件作为输入文件，在经过格式化后产生输出文件。由于跟踪文件是一系列的文件，在使用 TKPROF 时可以对单个文件进行格式化，也可以将所有的跟踪文件串在一切在进行格式化。

- 逐一对单个文件进行 TKPROF，分别产生相应的输出文件。
- 对所有跟踪文件级联在一起在进行 TKPROF 处理，从而产生整个实例的格式化输出文件。

TKPROF 命令语法：

```
TKPROF  filename1, filename2 [ SORT  = [ opion][,option] ]
 [  PRINT = integer ]
 [ AGGREGATE  = [ YES | NO ] ]
 [ INSERT = filename3 ]
 [ SYS = [ YES | NO ]  ]
 [  [ TABLE = schema.table ] | [ EXPLAIN = user/password ]  ]
 [  RECORD = filename ]
```

filename1 指定的输入文件，可以是多个文件联起来。
Filename2 格式化输出文件。
SORT　　在输出到输出文件前，先进程排序。如果省去，则按照实际使用的顺序输出到文件中。排序选项有以下多种：
　PRSCNT 语句解析的数目
　PRSCPU 语句解析所占用的 CPU 时间
　PRSELA 语句解析所占用的时间
　PRSDSK 语句解析期间，从磁盘进行物理读的数目
　(暂略，见 原版)
PRINT　只列出输出文件的第一个 integer 的 SQL 语句。默认为所有的 SQL 语句。
AGGREGATE　如果= NO ，则不对多个相同的 SQL 进行汇总。
INSERT　SQL 语句的一种，用于将跟踪文件的统计信息存储到数据库中。在 TKPROF 创建脚本后，在将结果输入到数据库中。
SYS　　禁止或启用 将 SYS 用户所发布的 SQL 语句列表到输出文件中。
TABLE　在输出到输出文件前，用于存放临时表的用户名和表名。
EXPLAIN　对每条 SQL 语句确定其执行规划。并将执行规划写到输出文件中。

TKPROF 命令例子：

例 1：使用 TKPROF 中的 sort 和 print 参数 来对较大的跟踪文件进行处理，只输出物理 I/O 最繁忙的前 10 条语句：

$tkprof ora53269.trc 53269.prf sort=(PRSDSK,EXEDSK,FCHDSK) print=10

例 2：进行下面复杂的处理要求：

$tkprof sqlplus_007.trc out.prf
expalin=scott/tiger table=scott.temp_plan_table insert=storea.sql sys=no
sort=(EXECPU,FCHCPU)

- 由于用了 expalin，oracle 将 scott 与 Oracle 系统进行连接。并为各条 SQL 语句产生执行计划，此外，还可以利用它来获得访问路径和行源数目。
- 用 table=scott.temp … 来作为临时规划表。
- 由于使用了 insert，tkprof 就建立名为 storea.sql 的 SQL 脚本。该脚本用于将被跟踪的 SQL 语句的统计信息存储到数据库中。
- 由于 sys=no，tkprof 将忽略所有的递归 SQL 语句。
- 由于用了 sort , 则在将 SQL 语句写到输出文件前，先进行排序。

§12.7.5 解释 TKPROF 输出文件

经过 tkprof 处理输出的结构如：

```
select u.name , o.name from obj$ o , user$ u, trigger$ t
where t.baseobject = :1  and  t.obj#=o.obj#
and o.owner# = u.user# order by o.obj#

      call     count       cpu    elapsed       disk      query    current        rows
------------  ------  --------  ----------  ----------  ----------  ----------  ----------
parse             1      0.01       0.01          0           0           0           0
execute           1      0.01       0.01          0           0           0           0
getch             1      0.00       0.00          0           1           0           0
------------  ------  --------  ----------  ----------  ----------  ----------  ----------
total             3      0.02       0.02          0           1           0           0

Insert into employee(employee_id,last_name,first_name)
Values( 295, ‘Joe’, ‘Johnson’);

      call     count       cpu    elapsed       disk      query    current        rows
------------  ------  --------  ----------  ----------  ----------  ----------  ----------
parse             1      0.08       0.12          0           0           0           0
execute           1      0.04       0.20          5           3          11           1
getch             0      0.00       0.00          0           1           0           0
------------  ------  --------  ----------  ----------  ----------  ----------  ----------
total             2      0.12       0.32          5           3          11           1
```

```
select last_name, title, department_id
from employee;

      call     count       cpu    elapsed       disk      query    current        rows
---------- ---------- ---------- ---------- ---------- ---------- ----------  ----------
parse        1      0.01       0.01          0          0          0           0
execute      5      0.08       0.09          0          0          0           0
getch      250      0.09       0.10         54        202         64         245
---------- ---------- ---------- ---------- ---------- ---------- ----------  ----------
total      256      0.18       0.20         54        202         64         245
```

parse 分析是将 SQL 转成执行计划，包括安全授权及表等存在性检查
execute Oracle 执行的语句，如 insert,update,delete 及 select 等
fetch 查询所返回的行。

Count 语句的解释、执行或取数调用的次数。
CPU 执行某条语句所用 CPU 的时间。
Elapsed 解释语句、执行语句或取数所用去的时间（秒）。
Disk 语句解释、执行和取数前，从磁盘文件中物理读的块数。
Query 所有解释、执行和取数等的调用期间，从 SGA 区所读的回滚块数。
Current 所有解释、执行和取数等的调用期间，从 SGA 区所读的数据块数。
Rows SQL 语句所处理的行数。

第 3 个语句：
分析 1 次；执行 5 次；取数 250 行；总 CPU 是 0.18 秒；处理用去 0.20 秒；从磁盘读 54 数据块；从 SGA 中得到 266(202+64)；返回 245 行。
结论：花去 0.18 秒的 CPU 时间取到 245 行，还能接受。

识别需要调整的语句：

- 消耗超出 CPU 资源的能力；
- 分析、执行或取回的时间太长；
- 访问太多的数据块但只返回少量的行。

§12.7.6 解释计划(Explain Plan)策略

Oracle 的 explain Plan 用于确定特殊语句是如何处理。根据这些信息可以重写这些 SQL 语句。
有两种方法来产生解释计划：

1.用 TKPROF 生成解释计划：

在格式化用户跟踪文件时，TKPROF 可以用 EXPLAIN 参数来产生 SQL 语句解释计划，如果需要解释计划，则输出的结果文件格式多出语句的解释部分，如：

```
select  * from emp,dept
where  emp.deptno=dept.deptno;
```

call	count	cpu	elapsed	disk	query	current	rows
parse	1	0.16	0.29	3	13	0	0
execute	1	0.00	0.00	0	0	0	0
fetch	1	0.03	0.26	2	2	4	14

Misses in libaray cache during parse:1
Parse user id(8) SCOTT

Rows	Execution Plan
14	MEREGEJOIN
4	SORT JOIN
4	TABLE ACCESS(FULL)OF ‘DEPT’
14	SORT JOIN
14	TABLE ACCESS(FULL)OF ‘EMP’

上面的输出结果包括：

- SQL 语句的文本；
- 以表格形式输出 SQL 的跟踪统计信息；
- 在语句分析和执行时库缓存失败的数目；
- 对该语句解析的用户信息；
- EXPLAIN PLAN 解释的执行计划。

2.用 EXPLAIN PLAN FOR 生成解释计划：

对于运行性能不佳的 SQL 语句，可用 EXPLAIN PLAN FOR 命令来对该 SQL 语句进行解释。步骤如下：

- 创建 PLAN_TABLE 表以存放结果：
  在 UNIX 环境，运行$ORACLE_HOME/rdbms/admin/utlxplan.sql；
  在 NT 环境：运行%ORACLE_HOME%\rdbms\admin\utlxplan.sq。

SQL>@$ORACLE_HOME/rdbms/admin/utlxplan.sql；

Table created.

- 在 SQL 语句执行时用上 EXPLAIN PLAN FOR，如：

```
SQL> EXPLAIN PLAN FOR
2       select  * from emp,dept
3       where  emp.deptno=dept.deptno;
```

Explained.

§12.7.7 AUTOTRACE 实用程序

Oracle AUTOTRACE 实用程序集中了 TKPROF 和 EXPLAIN PLAN 的功能。与 TKPROF 不一样的是它不需要对跟踪文件进行格式化；与 EXPLAIN PLAN 不一样的是，AUTOTRACE 在产生执行计划前就实际执行了解释过的语句。

1．使用 AUTOTRACE 准备

- 每个需要运行 AUTOTRACE 的用户要有自己的 PLAN_TABLE 表（用 utlxplan.sql 建立）；
- 以 SYS 登录并运行 plustrace.sql 脚本来创建角色 PLUSTRACE；
- 授权 PLUSTRACE 角色给每个希望进行 AUTOTRACE 的用户。

2．使用 AUTOTRACE

AUTOTRACE 只能在 SQL*PLUS 下运行，所以只需要在 SQL 语句前加上 set autotrace on 即可，如：

```
SQL> connect scott/tiger@ora816
Connected.
SQL> start d:\oracle\rdbms\admin\utlxplan.sql

Table created.

SQL> connect internal
Connected.
SQL> show user
USER is "SYS"
SQL> start d:\oracle\sqlplus\admin\plustrce.sql (下面的提示是正常的)
SQL>
SQL> drop role plustrace;
drop role plustrace
         *
```

```
ERROR at line 1:
ORA-01919: role 'PLUSTRACE' does not exist


SQL> create role plustrace;

Role created.

SQL>
SQL> grant select on v_$sesstat to plustrace;

Grant succeeded.

SQL> grant select on v_$statname to plustrace;

Grant succeeded.

SQL> grant select on v_$session to plustrace;

Grant succeeded.

SQL> grant plustrace to dba with admin option;

Grant succeeded.

SQL>
SQL> set echo off
SQL> grant plustrace to scott;

Grant succeeded.

SQL> show user
USER is "SYS"
SQL> connect scott/tiger@ora816
Connected.

SQL> set autotrace on


SQL> select d.dname,e.ename,e.sal from dept d, emp e
  2  where d.deptno=e.deptno;
```

```
DNAME          ENAME           SAL
-------------- ----------      ----------
RESEARCH       SMITH            800
SALES          ALLEN           1600
SALES          WARD            1250
RESEARCH       JONES           2975
SALES          MARTIN          1250
SALES          BLAKE           2850
ACCOUNTING     CLARK           2450
RESEARCH       SCOTT           3000
ACCOUNTING     KING            5000
SALES          TURNER          1500
RESEARCH       ADAMS           1100
ACCOUNTING     MILLER          1300
RESEARCH       赵元杰          1234.5

13 rows selected.


Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   NESTED LOOPS
   2    1     TABLE ACCESS (FULL) OF 'EMP'
   3    1     TABLE ACCESS (BY INDEX ROWID) OF 'DEPT'
   4    3       INDEX (UNIQUE SCAN) OF 'PK_DEPT' (UNIQUE)


Statistics
----------------------------------------------------------
          0  recursive calls
          4  db block gets
         28  consistent gets
          0  physical reads
          0  redo size
       1207  bytes sent via SQL*Net to client
        424  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
         13  rows processed
```

SQL> set autotrace off

在设置 set autotrace on 时，也可以在后面加如下参数：

选件	说明
ON EXPALIN	显示查询结果和执行路径，但没有统计数据
TRACEONLY	显示执行路径和统计，但没有查询结果
TRACEONLY STATISTICS	仅显示统计，没有执行路径和结果
OFF	设置 autotrace 为无效

第 13 章　数据访问方法

在程序设计中，查询语句是使用频率最高的语句，也是最可以进行调整的语句。一般来说，由 Oracle 的优化器选择的路径是最好的路径。在极少的情况下（大约 5%至 10%），你可以提供更好的路径。也就是说，你可以给出更优化的 SQL 语句。下面是一些常用优化的用法。

§13.1 使用索引的访问方法

作为编程人员，除了理解表的结构关系外，在编写查询程序时，要知道所访问表中哪些列已经创建了哪些索引等，从而在编程中运用这些索引。只有这样，才能达到优化的目的。下面给出在创建索引的表的访问方法。

§13.1.1 何时创建索引

在利用索引的情况下，由于只从表中选择部分行，所以能够提高查询速度。对于只从表中查询出总行数的 2%到 4%行的表，可以考虑创建索引。下面是创建索引的基本原则：

- 对于所有分配给 表的数据块，经常以查询关键字为基础，并且表中的行遵从均匀分布；
- 以查询关键字为基础，表中的行为随机排序；
- 表只能感包含的列数相对比较少；
- 表中的大多数查询都包含相对简单的 WHERE 从句；
- 缓存命中率较低，并且不需要操作系统缓存。

§13.1.2 索引列和表达式的选择

在创建索引时，选择列和表达式是非常重要的，下面是一些基本的原则：

- WHERE 从句频繁使用的关键字；
- SQL 语句中频繁用于进行表连接的关键字；
- 可选择性高的关键字，可选择性值在另外章节介绍；
- 对于取值较少的关键字或表达式，不要采用标准的 B*树索引。可以考虑建立位图索引；
- 不要将那些频繁修改的列作为索引列；
- 不要使用包含函数或操作符的 WHERE 从句中的关键字作为索引。如果存在这样的需要的话，可以考虑建立函数索引;
- 如果大量的并发 INSERT、UPDATE、DELETE 语句访问了父表或子表，则考虑使用完整性约束的外部键作为索引；
- 在选择所有列，还要考虑该索引所引起的 INSERT、UPDATE、DELETE 操作是否值得。

§13.1.3 选择复合索引的主列

多列索引叫复合索引，复合索引有时比单列索引有更好的性能。如果在建立索引时采用了几个列作为索引。则在使用时也要按照建立时的顺序来描述。也就是说，主列是最先被选择的列。
例如：

CREATE INDEX comp_ind ON tab1(x, y, z);

这里的复合列变为 x,xy 和xyz 几个部分。如果索引被建立成复合型，则应该在查询语句中带有： where ... and 来使用复合键。

在选择作为组合索引的关键字时，要遵循下列原则：

- 应选择在WHERE 从句条件中频繁使用的关键字，且这些关键字由AND操作符连接；
- 如果几个查询都选择相同的关键集合，则考虑创建组合索引；
- 创建索引后使得WHERE从句所使用的关键字能够组成前导部分；
- 如果在某些关键字在WHERE 从句中的使用频率较高，则考虑创建索引；
- 如果某些关键字在WHERE从句中使用频率相当，则创建索引时考虑从高到低的顺序来说明关键字。

§13.1.4 要用可选择性索引

索引的“可选择性”是指在该索引列里存储不同值的数目和记录数的比。比如某个表的记录数是1000条，而该表的索引列的值只有900个不同的值（有100个是相同或是空）。这样索引的可选择性为900/1000为0.9 。这样当然效果就不好，最好的索引可选择性（如主键索引）是1.0 。索引的可选择性是衡量索引的利用率的方法，比如在极端的情况下，一个表记录数是1000，而索引列的值只有5个不同的值，则索引的可选择性很差（只有0.005）。这样的情形使用全表扫描要比采用索引还好。

§13.1.5 测量索引的可选择性

方法1：
可以采用命令方式来测量某个索引的可选择性值，例如COMPAY表的city 列和state列是被建立成索引，则：

COMPAY 表的city和state 列的不同值的数目为：
Select count(distinct ||%||state) from COMPAY;

用下面语句查出COMPAY表的记录数：
select count(*) from COMPAY;

按照前面的方法进行比可得到该索引的可选择性值。

方法2：
用analyze 对表进行统计，对表进行分析后，系统同样对索引进行了分析。在查询结果。

ANALYZE TABLE compay compute statistics;

一旦表及其索引被分析过，就可以查询USER_INDEXES数据字典中的数据：

查询不同索引的列值：
select distinct_keys from user_indexes where table_name='COMPAY';

查询表中的记录数：
select num_rows from user_tables where table_name ='COMPAY';

索引可选择性= distinct_keys / num_rows

§13.1.6 避免全表扫描

在应用的程序设计中，除了一些必要的情况下，如月报数据的统计。打印所有清单等可以允许使用全表扫描外，一般都尽量避免涉及全表扫描。全表扫描就是指不加任何条件的查询语句。下面情况可以 Oracle 就使用全表扫描：

1．所查询的表没有索引；
2．需要返回所有的行；
3．对索引主列有条件限制，但是使用了函数，则 Oracle 使用全表扫描，如：
where upper(city)='TOKYO';
这样的语句不会使用索引方法。所以就只能全表扫描。
4．带有 is null 和 is not null 及 != 等子句。如：

. . . where city is null ;
. . . where city is not null;
. . . where city != ‘TOKYO’;
5．带 like 并使用’%’这样的语句就使用全表扫描：

§13.1.7 编写避免使用索引的语句

有时，使用索引进行查询可能速度更慢，如果用户希望避免使用类似的访问路径。则采用全表扫描方法。下面是强制优化程序使用全表扫描的方法：

- 提示 NO_INDEX 能够提供 CBO 最大的灵活性，并且禁止对某些索引的使用；
- 提示 NULL 能够强制优化程序选择全表扫描，而不是索引扫描；
- 提示 INDEX,INDEX_COBINE 或 AND-EQUAL 能够强制优化程序使用某种索引，或者利用某种组合索引。

§13.1.8 编写使用索引的语句

在创建了索引后，Oracle 系统只能在 INSERT 和 UPDATE 时建立了索引数据项。优化程序并不自动使用索引。既然这样，作为编程人员就要在程序中指定使用索引。其实，使用索引是非常简单的事，只要在 WHERE 从句后将索引列写上即可。
那么，程序员在程序中使用索引没有，可以使用 EXPLAIN PLAN 语句来对各个语句进行分析就可看出。

§13.1.9 重新构造索引

为了清除那些由于删除表记录而无效的索引数据项。可以采用重建索引的方法来实现。可以采用重新创建索引或以修改方式来创建索引来实现。对于数据量比较大的应用来说，经常性地进行重新构造索引和数据是一个好的方法。可以进行下面的整理工作：

- 清除那些不具有可选择性的索引，以提高DML处理速度；
- 重新组织索引；

可用 ALTER INDEX . . . REBUILD 命令对索引进行重组。比如：

SQL>ALTER INDEX EMPNO REBUILD ONLINE;

在alter index 语句加online 选项，可最大限定地减少由于索引的重建所出现的任何加锁现象。在重建索引期间，用户可以继续在该索引所对应的表上进行DML处理。

此外，也可用alter index … tablespace rebuild 来使索引存放到另外的表空间上。如：

```
SQL>select owner,segment_name,segment_type,tablespace_name
 from dba_segments  where owner='SCOTT' ;
```

```
OWNER                SEGMENT_NAME       SEGMENT_TYPE   TABLESPACE_NAM
-------------------- ------------------ ------------------ --------
SCOTT                DEPT               TABLE          SYSTEM
SCOTT                EMP                TABLE          SYSTEM
SCOTT                BONUS              TABLE          SYSTEM
SCOTT                SALGRADE           TABLE          SYSTEM
SCOTT                PK_DEPT            INDEX          SYSTEM
SCOTT                PK_EMP             INDEX          SYSTEM
```

已选择6行。

```
SQL> alter index scott.pk_emp rebuild online tablespace indx storage(initial 20k next 20k);
```

索引已更改。

§13.1.10 压缩索引

可以用 ALTER INDEX … 后加 COALESCE 来实现对索引的压缩。

另外的方法是用alter index . . . 加coalesce 选项来重建索引，这高选项在重建索引期间不需要任何额外的空间。只是在重建时将索引页内的分枝拼接为一个整体。如：

```
SQL> alter index scott.pk_dept coalesce;
```

索引已更改。

§13.2 创建索引和使用索引

§13.2.1 使用函数索引

基于函数的索引就是表达式的索引。最新的 Oracle8i 支持函数索引，Oracle 公司也推荐使用基于函数的索引。

例 1.比如创建了下面的函数的索引：

```
CREATE INDEX idx ON  table_1  (a + b * (c - 1), a, b);
```

则可以在查询语句中使用函数索引：

```
SELECT a
FROM table_1
WHERE a + b * (c - 1) < 100;
```

例2. 创建一个查询时使用大写字母的函数索引：

```
CREATE INDEX uppercase_idx ON emp (UPPER(empname));
```

当创建这样的函数索引，在查询语句中应该采用函数来判断，如：

```
SELECT * FROM emp
WHERE  UPPER(empname) = 'MARK';
```

§13.2.2 使用位图索引-

位图索引是 Oracle 提供的比较新的功能，创建位图索引的列要求值有写讲究。详细见“SQL*PLUS”部分。

主要考虑：

- 性能方面的因素；
- 存储方面的因素；
- 维护方面的因素。

1.性能方面的因素：

- WHERE 从句中包含有多个判定式，并且这些判定式中使用了具有低的或中等的基数的列；
- 上述判定均选择大量的行；
- 对于部分或全部具有低或中等基数的列，创建相应的位图索引；
- 被查询的表中包含有许多行。

使用位图索引有下面限制：

- 对于直接加载的位图索引，SORTED_INDEX不能使用；
- 位图索引不能考虑基于规则的优化；
- 对于完整性约束不能使用位图索引。

§13.2.3 使用 B 树索引-

§13.2.4 使用反向键索引

有时应用需要对最新的一组记录进行操作比对旧的记录进行操作更频繁。也就是说，用户经

常操作的记录中，对最近的记录操作次数更多。这样的应用应该建立逆关键字索引。
建议在下面情况建立逆关键字索引：

- 被索引的列顺序是逆序递减的数值。
- 表中存在并发 INSERT 和 DELETE。
- 应用程序用于 Oracle 并行服务器环境。
- 查询中很少用索引范围，如在。。。之间等。

§13.3 使用范围索引

范围索引有用户定义的所有类型来创建。所有类型提供一种有效的机制来访问满足一定操作谓词的数据。特别是，用户定义的培训索引是Oracle的选件的一部分。比如，空间选件。
例如，空间数据所有类型有效的搜寻和返回给出范围的空间数据。

插件确定你指定的参数和管理范围索引。类似，性能和范围索引的存储特性都是以插件方式给出。
参考下列插件文档：

- 什么样的数据类型可以创建索引类型?
- 索引类型能提供什么?
- 什么样的需要索引类型支持?
- 如何创建和管理范围索引?
- 如何管理范围索引?
- 性能特点是什么?

可以参考CREATE INDEXTYPE 语句和 *Oracle Spatial User's Guide and Reference*.

§13.4 使用簇

聚合是一组表的集合，这些表有一个共同的列。对表进行聚合时需要注意下面的原则：

- 对于应用程序连接语句中经常用在一起访问的表可以考虑创建聚合；
- 那些很少进行连接的表不要创建聚合；
- 如果应用程序经常对某个表进行全表扫描也不要创建聚合；
- 如果经常选择主表和子表，则要创建聚合；
- 如果表中具有相同的键值，但是这些键值占空间超过1个或 2 个数据块时，不宜创建聚合；
- 如果各个键值对应的行差异太大就不宜创建聚合。

例1。下面语句创建一个聚合（cluster）名字为 personnel，它带的索引键列(cluster key column)为department_number。一个聚合大小是512字节：

```
CREATE CLUSTER personnel
( department_number NUMBER(2) )
SIZE 512
STORAGE (INITIAL 100K NEXT 50K);
```

例2。加表到建立好的聚合里去。这里将emp和dept表加到personnel聚合中：

```
dept tables to the cluster:
CREATE TABLE emp
(empno NUMBER PRIMARY KEY,
ename VARCHAR2(10) NOT NULL
CHECK (ename = UPPER(ename)),
job VARCHAR2(9),
mgr NUMBER REFERENCES scott.emp(empno),
hiredate DATE
CHECK (hiredate < TO_DATE ('08-14-1998', 'MM-DD-YYYY')),
sal NUMBER(10,2) CHECK (sal > 500),
comm NUMBER(9,0) DEFAULT NULL,
deptno NUMBER(2) NOT NULL )
CLUSTER personnel (deptno);

CREATE TABLE dept
(deptno NUMBER(2),
dname VARCHAR2(9),
loc VARCHAR2(9))
CLUSTER personnel (deptno);
```

例3. 下面语句在聚合键personnel上创建一个聚合索引(cluster index)

```
CREATE INDEX idx_personnel ON CLUSTER personnel;
```

当创建完聚合索引(cluster index)后，就可以往emp和dept表中插入数据了。

§13.5 使用 Hash 簇

哈希聚合（Hash Clusters）也是一种类似于聚合表的方法。但在等价查询中比一般的索引表或索引聚合（index cluster）还要好。有关HASH CLUSTER的具体解释见本教材的附录部分。哈希聚合由一个哈希函数对每行的键值来使用表数据。所有行的相同键值在硬盘中被存放在一起。

§13.5.1 何时创建 Hash 簇

是否创建哈希聚合，主要考虑下面原则：

- 对那些在使用中经常被访问的列，并且这些列经常使用等式的一组表可以创建哈希聚合。
- 对于非定的聚合键，如果空间能容纳下将来所存放的所有行才能创建哈希聚合。
- 如果空间不足，不要考虑创建哈希聚合。
- 如果在应用经常采用全表扫描，则不要创建哈希聚合。
- 那些经常被修改聚合键值的表不宜创建哈希聚合。

§13.5.2 创建 Hash 簇

使用CREATE INDEX 语句再后面加 HASHKEYS 保留字就可以创建哈希聚合。见下面例子：

例1.创建哈希聚合（Hash Cluster）：
下面语句创建一个哈希聚合名字为personnel，聚合键列为department_number，哈希键值最大为500字节；每个键值为512字节：

```
CREATE CLUSTER personnel
( department_number NUMBER )
SIZE 512 HASHKEYS 500
STORAGE (INITIAL 100K NEXT 50K);
```

这个例子忽略了 HASH IS 子句，这样Oracle就使用内部哈希函数来创建这个聚合。

例2.创建哈希聚合（Hash Cluster）：
下面语句创建一个哈希聚合名字为personnel，聚合键列由home_area_code和home_prefix组成，并且为这两个列使用了哈希函数：

```
CREATE CLUSTER personnel
( home_area_code NUMBER,
home_prefix NUMBER )
HASHKEYS 20
HASH IS MOD(home_area_code + home_prefix, 101);
```

§13.6 使用实体视图 -

§13.6.1 实体视图概念

§13.6.2 创建实体视图

§13.6.3 使用实体视图

§13.6.4 管理实体视图

第 14 章　优化器简介

Oracle 提供一个叫优化器(Optimizer)的软件，编程人员可根据情况选用某种优化模式。本章内容可参考《OCP:Oracle8iDBA 性能调整及网络管理》chapter 3 和《Oracle8iDesigning and Tuning for Performance》chapter 4.
或 **Oracle9i**Database Performance Tuning Guide and Reference
Release 2 (9.2) Part No. A96533-01 中的 Introductions to the Optimizer 1-**1**

§14.1 Oracle 优化器

Oracle 的优化器（Optimizer）实际上是数据库环境的参数设置。可以在 INITsid.ORA 文件内的 OPTIMZER_MODE=RULE 或 OPTIMZER_MODE=COST 或 OPTIMZER_MODE=CHOOSE 来设置优化目标。用户也可以在会话和查询方式下更改优化器的默认操作模式。

如果 OPTIMZER_MODE=RULE，则激活基于规则的优化器(RBO)。基于规则的优化器按照一系列的语法规则来推测可能执行路径和比较可替换的执行路径。

如果 OPTIMZER_MODE=COST，则激活基于成本的优化器(CBO)。它使用 ANALYZE 语句来生成数据库对象的统计数据。这些统计数据包括表的行数、平均长度及索引中不同的关键字数等。基于这些统计数据，成本优化器可以计算出可获得的执行路径的成本。并选择具有最小的成本执行路径。在 CBO 模式下，需要经常运行 ANALYZE 命令来确保数据的准确性。

如果 OPTIMZER_MODE=CHOOSE，则在表被分析的情况下激活基于成本的优化器。但当一个查询分析的表是未被 ANALYZE 分析统计过的时侯，CBO 优化器就决定进行全表扫描操作。所以为了减少可能的全表扫描，应该尽量避免使用 OPTIMZER_MODE=CHOOSE 选项。

下面介绍在编程序中常用到的语句优化。

本章讨论 SQL 处理技术、优化方法以及优化程序如何执行 SQL 语句。包括：

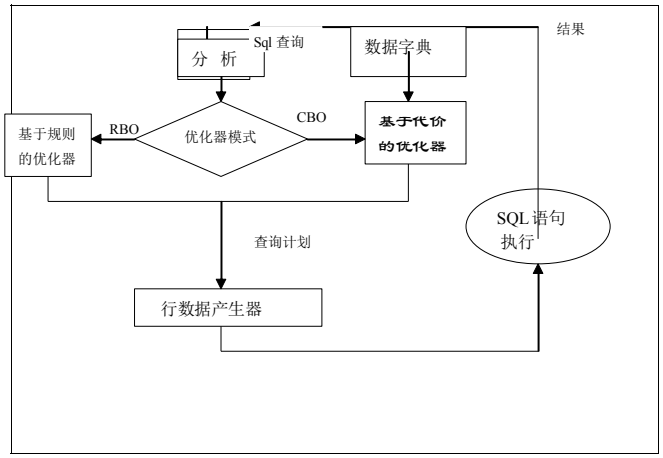
- SQL 处理体系结构
- EXPLAIN PLAN
- 优化程序的定义
- 选择优化程序的方法和目标
- 基于开销的优化程序(CBO=Cost_Base Optimizer)
- CBO 参数
- 可扩展的优化程序
- 基于规则的优花程序(RBO=Rule_Base Optimizer)
- 优化程序操作概述

- 优化连接（Join）
- 优化使用公共子表达式的语句
- 表达及条件的评价
- 转换及优化的语句

§14.2 SQL 处理体系结构

SQL 处理体系结构主要有：

- 解析程序
- 优化程序
- 行源产生程序
- SQL 执行



§14.2.1 解析程序

解析程序执行下面两种功能：

- 语法分析：检查 SQL 语句的语法正确性
- 语义分析：例如检查当前数据库对象和相应的对象属性，并判断其准确性。

§14.2.2 优化器

优化器是 SQL 处理引擎的核心，Oracle 服务器支持两种优化方法：基于规则的优化和基于开销的优化程序。

§14.2.3 行源程序产生器

行源程序产生器(Row Source generator)从优化程序接受优化规则，并输出 SQL 语句的执行规则。执行规则由行源的集合所组成，它呈现出树型结构。行源程序是一种迭代的控制结构，它以迭代的方式对执行集合进行处理，并且每次只处理一次。

§14.2.4 SQL 执行

SQL 执行（Execution）是操作执行规则时的组件，它与 SQL 语句紧密相连，并且可以给出查询的结果。

§14.3 EXPLAIN PLAN

通过使用 EXPLAIN PLAN 语句，用户可以对 SQL 语句优化程序所选择的执行规划进行检查。只要发出 EXPLAIN PLAN 语句，就可以对输出表进行查询，下面是输出表的数据描述先前所检查的语句：

ID	OPERATION	OPTION	OBJECT_NAME
0	SELECT STATEMENT		
1	FILTER		
2	NESTED LOOPS		
3	TABLE ACCESS	FULL	EMP
4	TABLE ACCESS	BY ROWID	DEPT
5	INDEX	UNIQUNE SCAN	PK_DEPTNO
6	TABLE ACCESS	FULL	SALGRADE

§14.4 选择优化器路径及目标

默认方式下，基于代价（CBO）是最好的方法。它使用最少的资源就能对所有行进行处理。Oracle也可以使用时间响应来优化语句。如： SQL语句并行执行。优化器可以选择最小共用资源消费开支时间。它可以由参数OPTIMIZER_PERCENT_PARALLEL来指定并行执行的优化器的个数。
执行计划由优化器根据目标来产生。**最大吞吐量优化**类似于宁愿不要索引的全表扫描的结果

或者相当于不要嵌套循环的分类合并连接。然而**最佳响应时间的优化**有点类似于索引扫描或嵌套循环连接。
例如你可以在任意的嵌套循环操作中或分类合并操作中使用连接语句。当嵌套循环操作快速返回第一行时，这些分类合并语句也可以快速返回整个结果。如果你的目标是改善吞吐量。那么优化器就有点象选择分类合并连接操作。如果你的目标是选择改善响应时间，则优化器更象选择一个嵌套连接操作。
综合上面的描述，选择优化器主要是基 于应用的需要：

- 对于批处理应用，如报表输出应用，需要采用吞吐量优化。因为吞吐量对于批处理来说更重要。
- 对于交互式应用，如Oracle Forms 应用或SQL*PLUS查询，需要采用最佳时间响应优化。因为交互式用户等着看到第一行的数据。
- 对于用ROWNUM来限制查询结果的查询，优化首先要考虑的响应时间。因为它要求的是得到最快的结果。

当选择最佳路径和目标时，优化器受到下面因素的影响：

- OPTIMIZER_MODE 初始参数
- 数据字典的统计数据
- ALTER SESSION 语句的OPTIMIZER_GOAL 参数
- 在提示中改变目标

§14.4.1 OPTIMIZER_MODE 初始参数

可以用OPTIMIZER_MODE来建立优化路径的默认值,它可以取下面的值：
choose 表示优化器在基于代价和基于规则两种之间进行选择。如果数据字典有访问表的至少一行的统计数据，则优化器使用的基于代价和最佳吞吐量方法。如果访问表没有统计数据，则优化使用基于规则的路径。默认为基于规则的路径。

All_rows 对整个SQL语句，优化器使用基于代价的路径。使用最小资源返回整个行。
FIRST_ROWS 对整个SQL语句，优化器使用基于代价的路径。使用最小资源返回第一行。
RULE 对整个SQL语句，优化器使用基于规则的路径。

如果优化使用基于代价的路径，而访问的表没有统计数据时，优化器就使用该表的一些近似值来代替。

§14.4.2 数据字典中的统计数据

Oracle 为 CBO 存储有列、表、簇 、索引及分区的统计数据。可以使用 ANALYZE 语句或 COMPUTE STATISTICS 子句和 DBMS_STATS 包来得到详细的统计结果。为了给优化提供最新的数据，你应该经常使用 ANALYZE 语句对表进行统计。

§14.4.3 ALTER SESSION 语句的 OPTIMIZER_GOAL 参数

ALTER SESSION 语句中带 OPTIMIZER_GOAL 参数可以越过初始化路径和 OPTIMIZER_MODE 参数所建立的目标。这个参数的结果影响到 SQL 语句的优化。但它对会话中所使用 SQL 语句的递归不起作用。OPTIMIZER_GOAL 可以有下面的值：
choose 表示优化器在基于代价和基于规则两种之间进行选择。如果数据字典有访问表的至少一行的统计数据，则优化器使用的基于代价和最佳吞吐量方法。如果访问表没有统计数据，则优化使用基于规则的路径。默认为基于规则的路径。
All_rows 对整个SQL语句，优化器使用基于代价的路径。使用最小资源返回整个行。
FIRST_ROWS 对整个SQL语句，优化器使用基于代价的路径。使用最小资源返回第一行。
RULE 对整个SQL语句，优化器使用基于规则的路径。

§14.4.4 关于提示的改变目标

可以在单个的 SQL 语句中，使用 FIRST_ROWS, ALL_ROWS, CHOOSE, 或 RULE 可以替代由 OPTIMIZER_MODE 初始参数和 ALTER SESSION 的 OPTIMIZER_GOAL 参数所设置的效果。在默认下，系统使用的是基于最佳吞吐量的代价的优化。但可以用下面方法来改变 CBO 的目标：

- 在会话中为CBO改变所有的SQL语句目标，使用：

ALTER SESSION SET OPTIMIZER_MODE 语句带 ALL_ROWS 或FIRST_ROWS子句。

- 为单个SQL语句指定CBO的目标。使用 ALL_ROWS 或 FIRST_ROWS 提示。如：

ALTER SESSION SET OPTIMIZER_MODE = FIRST_ROWS;

§14.5 基于代价优化器（CBO）

CBO在考虑访问路径和模式的统计数据（表或索引）来确定哪种执行计划最有效。同时 CBO 也考虑提示语句中哪种是最优的建议。CBO 考虑下面步骤：

1.优化器在它的有效路径和提示中为SQL语句产生一组潜在的计划。

2.优化器根据数据字典的统计估计每个计划的代价，并存储表、索引和分区的特征。
代价（cost）是预期资源要求的估计值。优化器计算每一种访问方法的代价并基于估计计算机资源（包括I/O和内存）的次序。从而得到执行该语句的需求。
较大的代价需要较多时间来执行，而较小的计划需要较少的时间来执行。当使用并行时，资源不直接与消失的时间有关。

3. 优化器比较这些代价然后选择较小的代价。

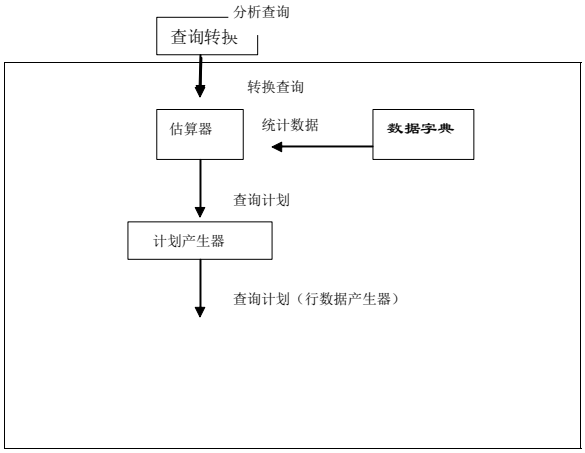
维护CBO的有效性，必须收集有关的统计并使它们准确才行。可以用下面方法来收集统计数据：

- 用ANALYZE 语句；
- 使用DBMS_STATS包。

§14.5.1 CBO 结构调整

CBO包括下面部件：

- 查询转换器
- 评价器
- 规划产生程序



§14.5.2 CBO 需求

要采用基于CBO的优化，要求下面的任何一种特征：

- 分区的表
- 索引构造的表
- 逆键字索引
- 基于函数的索引

- 在SELECT语句中的SAMPLE子句
- 并行执行和并行DML
- 星形转换
- 星形连接
- 可执行的优化器
- 查询重写(实体化视图)
- 进展仪表(Progress meter)
- HASH 连接
- 位图索引

§14.5.3 使用 CBO

要在语句中使用CBO，就要收集表的统计数据，并用下面方法来使CBO有效：

- 确认OPTIMIZER_MODE 初始参数设置为 CHOOSE.
- 在会话中启动CBO，要用 ALTER SESSION SET OPTIMIZER_MODE 语句并带 ALL_ROWS 或FIRST_ROWS 子句。
- 在一个单独的SQL语句中使用CBO，要用提示而不用RULE。

由CBO根据表的大小来产生规划，如果使用直方图情况下，还要产生的潜在的数据分布。当通过具有少量数据的CBO来进行应用程序的原型测试时，不要认为实际数据库选择的规划与原型数据库所选择的规划就一定一致。

§14.5.4 CBO 访问路径

明确地描述执行计划如何从数据库返回数据是优化器的最重要选择之一。对于任何表的任意行，该表的行的定位和返回都有许多的访问路径。优化器可从中选择一个。下面是Oracle访问数据的基本方法：

全表扫描
全表扫描是从表中返回所有的行。执行全表扫描，Oracle从该表读所有的行。检查每一行，看它是否满足where 子句的要求。实际上Oracle是读表中相继的数据块。所以它可以采用多块读。

样本表扫描

Oracle在Oracle8i Release 8.1.6 版本之后，提供了在select 语句后加 SAMPLE BLOCK 来对表进行部分的扫描查询。这种查询要求是只能对单表进行，不能在连接查询上使用样本扫描。也不能在远程进行样本扫描。当然，如果希望在远程上进行样本扫描的话，可以先用CREATE TABLE AS SELECT 语句将远程表复制到本地，然后在使用 SAMPLE BLOCK SAMPLE BLOCK 语句。

例：扫描emp表的1%的内容，则发出：

select count(*)*100 from emp sample block(1);

或

SELECT * FROM emp SAMPLE BLOCK (1);

用Rowid访问表

用Oracle的rowid 也能访问到表的记录。每个行的rowid 能确定数据文件、行所在数据块及该行所在的块位置。可以用rowid 来快速定位到一个单行。

簇(Cluster)扫描

对于以索引簇形式存放的表，簇扫描能够从中获得具有相同簇键(cluster key)值的行。在索引化的簇中，所有具有相同簇键值的行都被存储进相同的数据库块中。为了执行簇扫描，Oracle 首先通过扫描簇的索引，并从 中得到所选择的 rowid 值，然后再基于该 rowid 对所有选择行进行定位。

. . . . . .

§14.6 基于规则(RBO)的优化程序

尽管 Oracle 支持基于 RBO(Rule-Based Optimzer)的优化程序，但建议大多数系统还是采用基于 CBO 的优化。因为 CBO 支持 DSS 的某些新增强的功能，所以对于象数据仓库应用系统也要采用 CBO 方法。

如果满足：
1 ）设置 OPTIMIZER_MODE=CHOOSE;
2 ）数据字典中没有统计信息；
3 ）未给 SQL 语句加提示。
则可以采用 RBO。

如果是：
1 ） OPTIMIZER_MODE=FIRST_ROWS 或 ALL_ROWS ;
2 ）不存在可用的统计信息；
则 CBO 使用默认值。

§14.6.1 RBO 访问路径

如果使用基于 RBO，则在执行规则中，访问路径的等级具有启发性，即如果存在多种可执行的 SQL 语句，则 RBO 选择等级较低的操作，因为等级低的语句执行速度快。下面是 RBO 访问路径及等级：

- 路径 1：根据 rowid 访问行
- 路径 2：通过 cluster 联结访问行
- 路径 3：根据唯一性主键或主键的哈西簇访问单行
- 路径 4：根据唯一键或主键访问单行
- 路径 5：cluster 连接
- 路径 6：哈西 cluster 键
- 路径 7：索引化 cluster 键
- 路径 8：复合索引
- 路径 9：单列索引
- 路径 10：索引化有界搜索
- 路径 11：索引化无界搜索
- 路径 12：合并排序联结
- 路径 13：带有 MAX 或 MIN 的列
- 路径 14：带 ORDER BY 的查询
- 路径 15：全表扫描

路径1：根据rowid 直接访问到单行

SELECT * from emp where **rowid='AAAA7BAA5AAA1UAAA';**

```
OPERATION                 OPTIONS                 OBJECT_NAME
----------------          ---------------         --------------------
SELECT STATEMENT
TABLE                     ACCESS BY ROWID         EMP
```

路径 2: 根据cluster 联结访问单行

```
SELECT *
FROM emp, dept
WHERE emp.deptno = dept.deptno
AND emp.empno = 7900;
```

The EXPLAIN PLAN output for this statement might look like this:

```
OPERATION                 OPTIONS          OBJECT_NAME
---------------    -------------   ---------------------
SELECT STATEMENT
NESTED LOOPS
```

TABLE ACCESS BY ROWID EMP
INDEX UNIQUE SCAN PK_EMP
TABLE ACCESS CLUSTER DEPT

Pk_emp 为主键名

路径 3：根据唯一键或主键的哈西簇键访问单行

col column_name for a12
SQL>l
1 select a.table_name,a.index_name,a.uniqueness,b.column_name
2 from *user_indexes a, user_ind_columns b*
3* where a.index_name=b.index_name and a.table_name='EMP'
SQL> /

TABLE_NAME INDEX_NAME UNIQUENES COLUMN_NAME
——————— ——————— ——————— ———————
EMP PK_EMP UNIQUE EMPNO

从这里看出 emp 表的 empno 列被说明为唯一索引，现在可以对 emp 用带索引进行查询：

SQL> explain plan for select * from emp where empno=7369;

Explained.

SQL> col operation for a18
SQL> col options for a18
SQL> select operation,options,object_name from *plan_table* ;

OPERATION OPTIONS OBJECT_NAME
---------- ---------- ----------
SELECT STATEMENT
TABLE ACCESS BY INDEX ROWID EMP
INDEX UNIQUE SCAN PK_EMP

路径 4~路径 15 省去，详细见 原版《Designing and tuning for performance》4 Optimizaer

§14.7 优化器操作

Oracle 接收到一条 SQL 语句后，首先进行句法分析，检查文法并对该查询产生可执行的方案

（plan）。这个可执行的方案有 EXPLAIN PLAN 来处理，方案确定后语句就被存放在 SQL 共享区内。而 Oracle 的优化器是用来确当对特定语句的最佳执行路径。Oracle 的优化器可以有两种类型：即基于规则和基于代价。下面是优化程序能进行优化的 SQL 语句。

§14.7.1 可优化的 SQL 语句

- 简单的 SQL 语句，即只设计单个表的 insert,update,select
- 简单的查询
- 等式连接
- 非等式连接
- 外连接
- 笛卡尔乘积
- 复合语句
- 组合查询
- 访问视图
- 分布式语句

§14.7.2 优化程序操作

优化程序自动简化ＳＱＬ语句中的某些常用结构，如果结果简化执行的话，这些语句变得非常简单，如 2000/10 简化为 200 。也可能变复杂，如将带 OR 的运算语句转换为两个复合的子查询。对于前者可以随时进行，但后者则取决于 where 子句的列上是否有索引以及选择哪种优化方法。

此外，还有一些其它的转换，包括：

- 化简算术表达式
- 将 IN 算子转化为一系列的 OR 条件
- 将一个 BETWEEN … AND 转换为一对比较表达式
- 将 OR 转换为复合查询
- 视图的定义合并到条件语句中
- 将一个复杂语句转换成连接条件语句

§14.7.3 基于规则或基于代价优化方法

Oracle 按照若干准则对每个语句进行优化，包括：

- 引用的对象有一个并行度
- 语句中的提示
- OPTIMZER_GOAL 的会话设置
- 初始化参数 OPTIMZER_MODE 的值
- 被引用的对象的统计结果

通过优化程序的分析，决定执行是采用基于规则或基于代价方法。考察优化的首要因素是看语句是否存在一次全表扫描和是否有并行（是否包括 PARALLEL 选项）。如果都有的话，就使用基于代价的优化来创建一个包含并行的执行计划。
基于规则的方法不支持并行处理语句的执行计划。如果没有 rule 语句的提示，Oracle 都采用基于代价方法进行优化。

§14.8 优化连接*

Oracle 提供了 4 种连接操作：

- 嵌套循环连接（nested loop）
- 合并排序连接（merge join）
- 哈西连接(Hash join)
- 簇连接（cluster join）

下面分别给出简单介绍。

§14.8.1 嵌套连接-

嵌套循环连接（NESTED LOOPS）操作是将两个数据源连接起来。通常是在连接时系统提供了可用的索引。实际在处理过程中，Oracle 先处理第 1 行，接着下一行（不是等到整个处理完毕才返回第 1 行）。在处理嵌套连接时，Oracle 优化器首先为连接选择一个驱动表，可能对驱动表进行全表扫描。对驱动表的每行进行索引检查，以便了解在表之间是否存在匹配。如果有一个匹配存在，则通过 NESTED LOOPS 操作并将该记录返回给用户。
例如：

```
SQL> explain plan for
select a.deptno,a.dname,b.empno,b.ename,b.sal
  from dept a, emp b where a.deptno=b.deptno and a.deptno=20;

Explained.

SQL> select operation,options,object_name from plan_table;

   OPERATION                 OPTIONS                 OBJECT_NAME
------------------   ------------------ ------------
SELECT STATEMENT
MERGE JOIN
TABLE ACCESS                 BY INDEX ROWID       DEPT
INDEX                        UNIQUE SCAN          PK_DEPT
FILTER
TABLE ACCESS                 FULL                 EMP
```

6 rows selected.

```
SQL> explain plan for
select a.deptno,a.dname,b.empno,b.ename,b.sal
  from dept a, emp b where a.deptno=b.deptno and b.empno=7369;

Explained.

SQL> select operation,options,object_name from plan_table;

OPERATION            OPTIONS              OBJECT_NAME
-------------------  -------------------  ------------
SELECT STATEMENT
NESTED LOOPS
TABLE ACCESS             BY INDEX ROWID       EMP
INDEX                    UNIQUE SCAN          PK_EMP
TABLE ACCESS             BY INDEX ROWID       DEPT
INDEX                    UNIQUE SCAN          PK_DEPT

6 rows selected.
```

在看为等价连接使用索引的例子：

```
SQL> select dept_id, id
2 from s_emp, s_dept
3 where s_emp.dept_id = s_dept.id;
```

这样 Oracle 优化器使用嵌套循环连接替代排序合并连接。嵌套循环连接不需要任何的排序操作，所以操作步骤为：
1 执行S_EMP 表全表扫描
2.每个返回行都使用DEPT_ID值，以在PK_ID索引上进行唯一扫描
3.使用从索引中返回的ROWID来定位S_DEPT表的相应行.
4. 为S_DEPT组合从S_EMP返回的每个行。

§14.8.2 合并连接

排序合并连接用于从两个独立的数据源进行连接。HASH 连接可能执行比排序合并连接更好。但另一方面，排序连接可能在下面情况下执行比 HASH 连接更好。

- 行源数据已经做过排序；
- 一个排序也没有做。

然而，如果一个排序操作涉及到一个较慢的方法（如一个索引扫描对应全表扫描），则采用排序合并可能会失策。
当连接条件是在两个表之间使用不等式（如<、<=、>、>=）时，排序合并连接很有用。排序合并连接执行比对大量数据进行嵌套循环连接更好(不要用 HASH 连接，除非有等价条件)。
在合并连接里，没有驱动表概念，该连接主要考虑两步：
１．排序连接操作:连接键的输入端都是被排序过；
２．合并连接操作：排序列表要合并在一起。

例子：带有非等价的排序连接

```
SELECT SUM(l.<revenue_amount>), l2.<creation_date>
FROM order_items l, order_items l2
WHERE l.<creation_date> < l2.<creation_date>
AND l.order_id <> l2.order_id
GROUP BY l2.<creation_date>, l2.line_item_id ;

Plan
-------------------------------------------------
SELECT STATEMENT
  SORT GROUP BY
    MERGE JOIN
    SORT JOIN
    TABLE ACCESS FULL ORDER_ITEMS
FILTER
  SORT JOIN
  TABLE ACCESS FULL ORDER_ITEMS
```

第 15 章　使用优化器提示

Oracle8i 提供了一种新的优化方法叫优化程序提示的方法，它可以允许编程人员提示 Oracle 优化器按照什么样的执行规则去执行。下面给出一点简单介绍：
本章内容可参考《Oracle8iDesigning and Tuning for Performance》chapter 7.
或

Oracle9i Database Performance Tuning Guide and Reference

Release 2 (9.2) Part No. A96533-01 中的 Optimizer Hints **5-1**

§15.1 提示（Hint）概念

一般在优化时，无论采用基于规则的或是基于代价的方法，由 Oracle 系统的优化器来决定语句的执行路径。这样的选择的路径不要见得是最好的。所以，Oracle 提供了一种方法叫提示的方法。它可以让编程人员按照**自己的要求**来选择执行路径，即提示优化器该按照什么样的执行规则来执行当前的语句。这样可以在性能上比起 Oracle 优化自主决定要好些。
通常情况下，编程人员可以利用提示来进行优化决策。通过运用提示可以对下面内容进行指定：

- SQL 语句的优化方法；
- 对于某条 SQL 语句，基于开销优化程序的目标；
- SQL 语句访问的访问路径；
- 连接语句的连接次序；
- 连接语句中的连接操作。

§15.1.1 提示的指定

如果希望优化器按照编程人员的要求执行，则要在语句中给出提示。提示的有效范围有限制，即**有提示的语句块**才能按照提示要求执行。下面语句可以指定提示：

- 简单的 SELECT ,UPDATE ,DELETE 语句；
- 复合的主语句或子查询语句；
- 组成查询（UNION）的一部分。

提示的指定有原来的注释语句在加“+”构成。语法如下：

[SELECT | DELETE|UPDATE] /*+ [hint | text] */

或

[SELECT | DELETE|UPDATE] --+ [hint | text]

注意在“/*”后不要空就直接加“+”，同样 “**--+**”也是连着写。

警告：如果该提示语句书写不正确，则 Oracle 就忽略掉该语句。

指定完整的提示：

对于复杂的语句，要用/*+ */来指定，它可以指定多个提示语句。且可以换行。

```
SELECT /*+ ORDERED INDEX (b, jl_br_balances_n1) USE_NL (j b)
USE_NL (glcc glf) USE_MERGE (gp gsb) */
b.application_id ,
b.set_of_books_id ,
b.personnel_id,
p.vendor_id Personnel,
p.segment1 PersonnelNumber,
p.vendor_name Name
FROM jl_br_journals j,
jl_br_balances b,
gl_code_combinations glcc,
fnd_flex_values_vl glf,
gl_periods gp,
gl_sets_of_books gsb,
po_vendors p
WHERE . . . . . .
```

§15.2 使用提示

了解了 Oracle 提供的提示功能后，就可以在编写程序时进行使用。

§15.2.1 提示的指定

使用提示，用户可以在基于开销的优化方法和基于规则的优化方法之间选择，由此可以对要求“最佳吞吐量”与“最佳响应时间”优化目标进行选择。优化方法如下：

- ALL_ROWS
- FIRST_ROWS
- CHOOSE
- RULE

如果某条语句已经指定了优化方法和优化目标后，则 Oracle 的优化器就按照指定的优化方法和目标进行执行。并且不考虑：
1）是否存在统计信息；
2）初始化参数 OPTIMIZER_MODE 的取值；
3）ALTER SESSION 语句中 OPTIMIZER_MODE 参数值。

§15.2.1.1 ALL_ROWS

ALL_ROWS 表示对语句块选择**基于开销**的优化方法，并且获得最佳的吞吐量（资源消耗总量最小）作为目标进行优化。语法如下：

/*+ ALL_ROWS */

例子：在查询 EMP 表希望用基于开销的优化方法，并获得最佳吞吐量，则使用下面语句：

```
SELECT /*+ ALL_ROWS */empno,ename,sal,job
FROM emp  WHERE empno=7566;
```

§15.2.1.2 FIRST_ROWS

FIRST_ROWS 表示对语句块选择**基于开销**的优化方法，并且获得最佳的响应（返回首行的资源最小化）作为目标进行优化。

使用 FIRST_ROWS 优化方法，优化器可能要进行下面工作：

- 如果能利用索引扫描，则不进行全表扫描；
- 当关联表是嵌套循环的内部表且能用索引扫描，则优化器先优化嵌套循环联结。
- 如果通过 order by 使得索引扫描可用，则优化器选择索引扫描以避免排序操作。

语法如下：

/*+ FIRST_ROWS */

例子：选择基于开销的优化方法，并希望获得最佳的响应时间，则：

```
SELECT /*+ FIRST_ROWS */empno,ename,sal,job
FROM emp
WHERE empno=7566;
```

§15.2.1.3 CHOOSE

选择 CHOOSE 表示告诉优化器要在基于开销和基于规则之间进行选择。优化器的确定要建立在是否存在访问表的统计信息之上：

- 如果数据字典中存在该表的统计数据，则选择基于开销，并以最佳吞吐量作为目标。
- 如果数据字典中不存在该表的统计数据，则选择基于规则。

语法为：

/*+ CHOOSE */

例子：

```
SELECT /*+ CHOOSE */ empno,sal,job FROM emp WHERE empno=7566;
```

§15.2.1.4 RULE

表示要求优化器对语句块选择基于规则的优化方法。语法如下：

/*+ RULE */

例子：

```
SELECT --+ RULE empno,ename,sal,job
FROM emp  WHERE empno=7655;
```

或

```
SELECT /*+ RULE */ empno,ename,sal,job
FROM emp  WHERE empno=7655;
```

§15.2.2 关于访问方法的提示

访问方法共有：

- FULL
- ROWID
- CLUSTER
- HASH
- INDEX
- INDEX_ASC
- INDEX_COMBINE
- INDEX_JOIN
- INDEX_DESC
- INDEX_FFS
- NO_INDEX
- AND_EQUAL
- USE_CONCAT
- NO_EXPAND
- REWRITE
- NOREWRITE

如果在语句中指定了上面的提示，并且语句所涉及的索引或簇是可用时，优化器就使用所指定的访问路径。否则，优化器就忽略提示的要求。

§15.2.2.1 FULL

FULL 提示表示对表选择全表扫描的访问方法。语法如下：

/*+ FULL ([table_name]| [table_aliase]) */

例：

SELECT /*+ FULL(A) don't use the index on accno */ accno,bal
FROM accounts a
WHERE accno=7789;

虽然这里使用了带索引的条件句，优化器也得选择全表扫描。

§15.2.2.2 ROWID

ROWID 表示对指定表选择根据 rowid 进行表扫描，语法如下：

/*+ ROWID(table_name) */

例：

SELECT /*+ROWID(emp)*/ * FROM emp
WHERE rowid>'AAAATKAABAAAFNTAAA' AND empno=155;

§15.2.2.3 CLUSTER

CLUSTER 表示对指定表选择簇扫描的访问方法。它仅对 CLUSTER 对象有效。语法如下：

/*+CLUSTER(table_name) */

例：

SELECT --+CLUSTER emp.ename, deptno
FROM emp,dept
WHERE deptno=10 AND emp.deptno=dept.deptno;

§15.2.2.3 HASH

HASH 表示对指定的表选择 HASH 扫描访问方法，它只对 CLUSTER 中的表有效。语法如下：

/*+HASH(table_name)*/

§15.2.2.4 INDEX

INDEX 表示对指定表选择索引扫描的访问方法。用户可以对域、B*树和位图索引应用本提示。对于建立了位图的索引，建议用 INDEX_COMBINE 更为合适。语法如下：

/*+INDEX(table_name [index]) */

例：

```
SELECT /*+INDEX(patients sex_index)use sex_index because there are few male patients*/
Name,height,weight FROM patients  WHERE sex='m';
```

§15.2.2.5 INDEX_ASC

INDEX_ASC 表示对指定的表选择索引的访问方法，并按照升序进行扫描。语法如下：

```
/*+INDEX_ASC(table_name [index])*/
```

§15.2.2.6 INDEX_COMBINE

INDEX_COMBINE 表示对指定的表选择位图访问路径。

- 如果没有提供可参考的索引，则优化器以最低开销为目标，选择位图索引的布尔组合方式；
- 如果有可参考的索引，则优化器就使用该位图的某写布尔组合。

语法如下：

```
/*+INDEX_COMBINE(table_name[index])*/
```

例：

```
SELECT /*+INDEX_COMBINE(emp sal_bmi hiredate_bmi)*/*
FROM emp
WHERE sal<5000 AND hiredate ,'01-JAN-1990';
```

§15.2.2.7 INDEX_JOIN

INDEX_JOIN 表示使用索引连接作为访问路径。语法如下：

```
/*+INDEX_JOIN(table_name[index])*/
```

例：

```
SELECT /*+INDEX_JOIN(emp sal_bmi hiredate_bmi)*/sal,hirdate
FROM emp
WHERE sal<5000 ;
```

§15.2.2.8 INDEX_DESC

INDEX_DESC 表示对指定表选择索引访问方法。如果使用索引区域扫描，则按照降序进行扫描。语法如下：

```
/*+INDEX_DESC(table_name[index])*/
```

§15.2.2.9 INDEX_FFS

INDEX_DESC 表示对指定表选择快速索引访问方法（不是全表扫描）。语法如下：

/*+INDEX_FFS(table_name[index])*/

例：

```
SELECT /*+INDEX_FFS(emp emp_empnp)*/ empno
FROM emp
WHERE empno>200;
```

§15.2.2.10 NO_INDEX

NO_INDEX 表示对指定表禁止选择索引访问方法。语法如下：

/*+NO_INDEX(table_name[index])*/

例：

```
SELECT /*+NO_INDEX(emp emp_empnp)*/ empno
FROM emp
WHERE empno>200;
```

§15.2.2.11 AND_EQUAL

AND-EQUAL 表示要进行执行规则的选择。使几个但列的索引的扫描合并起来。语法如下：

/*+AND_EQUAL(table_name[index] [inex]…)*/

§15.2.2.12 USE_CONCAT

USE_CONCAT 提示强制对查询语句中的 WHERE 从句的 OR 条件进行转换，转化成由 UNION_ALL 集合操作符连接的组合查询。一般来说，如果采用连接查询比不用连接查询低，则转换为用连接查询。

/*+USE_COMCAT*/

例：

```
SELECT /*USE_CONCAT*/* FROM emp
WHERE empno>50 OR sal<50000;
```

§15.2.2.13 NO_EXPAND

NO_EXPAND 对于具有 OR 或 IN 查询语句，它将阻止基于开销的优化器对其进行 OR 扩展。语法如下：

/*+NO_EXPAND*/

例：

```
SELECT /*+NO_EXPAND*/*
FROM emp
WHERE empno=50  OR  empno=100;
```

§15.2.2.14 REWRITE

REWRITE 表示可以将视图列表作为参数来看，如果用户使用 REWRITE，并且该列表包含有符合条件的**实体化视图**，则 Oracle 优化器将利用该视图而不用基于开销的方法。而对于列表以外的视图将不被考虑。如果在 REWRITE 中没有给出视图列表，则 Oracle 将搜索符合条件的实体化视图。并且利用该视图。语法如下：

/*+REWRITE(view,[view]…)*/

§15.2.2.15 NOWRITE

NOWRITE 表示禁止对查询块的查询重写操作，从而避免参数 QUERY_REWRITE_ENABLE 的设置。语法如下：

/*+NOWRITE*/

§15.2.3 关于连接次序的提示

§15.2.3.1 ORDERED

根据出现在 FROM 中顺序，ORDERED 提示将使得 Oracle 依此次序对其进行连接。语法如下：

/*+ORDERED*/

例：

```
SELECT /*+ORDERED*/tab1.col1,tabl2.col2,tab3.col3
FROM tab1,tab2,tab3
WHERE  tab1.col1=tab2.col1 AND tab2.col1=tab3.col1;
```

§15.2.3.2 STAR

强行让优化器使用星型查询规划。星型规划拥有查询中最大的一个表，该表位于连接次序的最后，并与嵌套式循环连接的级联索引连接。如果满足下面 3 个条件：

1）至少存在 3 个表；

2）最大表的级联索引至少存在 3 列；
3）不存在冲突的访问或连接访问提示。

/*+ STAR */

第四部分　ORACLE 系统调整

第 16 章　调整信息的来源

我们在进行 Oracle 的性能调整时，并不是凭感觉或凭经验来行事，而是根据系统在运行过程所产生的问题的信息记录来进行的。主要依据的信息来源有：系统产生的警告日志文件和跟踪文件、动态数据视图及 Oracle 的 GUI 工具。

§16.1 警告日志文件

一般当 Oracle 安装完成后，系统的日常的运行的一些基本信息都被记录到警告文件中。一般警告文件可以从初始化文件的 BACKGROUND_DUMP_DEST 参数来得到。警告文件的文件名为 alert_sid.log，其中 sid 是数据库实例的名字。

§16.1.1 警告日志文件信息

比如下面的警告日志片段说明，表空间扩展失败：

```
Mon Nov 22 11:16:09 1999
Thread 1 advanced to log sequence 184
  Current log# 1 seq# 184 mem# 0: /oradata2/oradata/ora2/redoora201.log
  Current log# 1 seq# 184 mem# 1: /oradata2/oradata/ora2/redoora2011.log
Mon Nov 22 11:26:46 1999
ORA-1652: unable to extend temp segment by 5 in tablespace          TEMP_SHA_DATA
Mon Nov 22 11:26:46 1999
ORA-1652: unable to extend temp segment by 5 in tablespace          TEMP_SHA_DATA
Mon Nov 22 11:26:46 1999
ORA-1652: unable to extend temp segment by 5 in tablespace          TEMP_SHA_DATA
Mon Nov 22 11:26:47 1999
ORA-1652: unable to extend temp segment by 5 in tablespace          TEMP_SHA_DATA
Mon Nov 22 11:26:47 1999
ORA-1652: unable to extend temp segment by 5 in tablespace          TEMP_SHA_DATA
Mon Nov 22 11:28:24 1999
Thread 1 advanced to log sequence 185
  Current log# 2 seq# 185 mem# 0: /oradata2/oradata/ora2/redoora202.log
  Current log# 2 seq# 185 mem# 1: /oradata2/oradata/ora2/redoora2021.log
Mon Nov 22 11:36:49 1999
```

ORA-1652: unable to extend temp segment by 5 in tablespace TEMP_SHA_DATA
Mon Nov 22 11:39:09 1999
Thread 1 advanced to log sequence 186
  Current log# 3 seq# 186 mem# 0: /oradata2/oradata/ora2/redoora203.log
  Current log# 3 seq# 186 mem# 1: /oradata2/oradata/ora2/redoora2031.log
Mon Nov 22 11:45:13 1999
Thread 1 advanced to log sequence 187
  Current log# 1 seq# 187 mem# 0: /oradata2/oradata/ora2/redoora201.log
  Current log# 1 seq# 187 mem# 1: /oradata2/oradata/ora2/redoora2011.log
Mon Nov 22 11:56:24 1999
Thread 1 advanced to log sequence 188
  Current log# 2 seq# 188 mem# 0: /oradata2/oradata/ora2/redoora202.log
  Current log# 2 seq# 188 mem# 1: /oradata2/oradata/ora2/redoora2021.log
Mon Nov 22 12:04:18 1999

下面信息说明，存在资源等待情况发生：
. . . . . .
Wed Nov 24 13:39:12 1999
Thread 1 advanced to log sequence 246
  Current log# 3 seq# 246 mem# 0: /oradata2/oradata/ora2/redoora203.log
  Current log# 3 seq# 246 mem# 1: /oradata2/oradata/ora2/redoora2031.log
Wed Nov 24 14:29:14 1999
Errors in file /oracle2/oracle88/app/oracle/admin/ora2/bdump/smon_1672.trc:
ORA-01575: timeout waiting for space management resource
Wed Nov 24 14:34:44 1999
Errors in file /oracle2/oracle88/app/oracle/admin/ora2/bdump/smon_1672.trc:
ORA-01575: timeout waiting for space management resource
Wed Nov 24 14:40:15 1999
Errors in file /oracle2/oracle88/app/oracle/admin/ora2/bdump/smon_1672.trc:
ORA-01575: timeout waiting for space management resource
Wed Nov 24 14:45:45 1999
Errors in file /oracle2/oracle88/app/oracle/admin/ora2/bdump/smon_1672.trc:
ORA-01575: timeout waiting for space management resource
. . . . . .

与性能有关的信息有：

警告日志信息	信息说明
Unable to extend temp segment by n in Tablespace x。如： Sun Nov 21 00:18:47 1999 ORA-1652: unable to extend temp segment by 5	错误代码是 ORA-1652，表示在临时表空间中找不到满足要求的连续空间。 见“调整物理 I/O”

in tablespace TEMP_PEK_DATA	
Unable to Extend Table x by n in Tablespace x	错误代码是 ORA-01653，表示在表空间 中没有没有足够的连续空间用于扩展。
Unable to Extend Rollback Segment x by String in Tablespace x	错误代码是 ORA-01650。表示没有足够的连续空间用于回滚段的扩展。见“调整物理 I/O”
Max# Extend n Readched in Table x	错误代码是 ORA-01631，表示在表的扩展中已经超过了允许的最大值。见“调整物理 I/O”
Checkpoint Not Complete，如： Thread 1 advanced to log sequence 18 Current log# 3 seq# 18 mem# 0: /oradata2/oradata/ora2/redoora203.log Sat Nov 20 15:42:30 1999 Thread 1 cannot allocate new log, sequence 19 Checkpoint not complete	当数据库的一个检查点开始后还未结束，下一个检查就开始了。见“调整 Redo Log Buffer”
Thread n Advance to Log Sequence n	日志之间切换时出现。在日志切换过度时提示。见“调整物理 I/O”

§16.1.2 管理警告日志文件

警告日志文件是在数据库运行期间发生的事件的记录文件。它在数据库运行时，将警告信息以追加的方式写到警告日志的后面。如果不进行管理而放任它增长的话，时间一长，它可能变的很大。使数据库系统不能正常工作。所以建议管理员定期将日志文件备份并进行编辑和裁减，删除不前面的旧的信息部分。也可以在数据库系统关闭后将警告日志文件删除，当删除该文件，Oracle 系统会自动建立新的警告日志文件。

§16.2 后台、事件及用户跟踪文件

跟踪文件是由 Oracle 后台进程产生的一个文本文件，它包含有创建会话时的一些会话信息。跟踪文件可以在初始化参数文件 INITsid.ORA 文件中指定或由用户服务器指定。所有的跟踪文件都包含性能调整和故障解决的重要信息。

§16.2.1 后台跟踪文件

后台跟踪文件可以从初始化文件的 BACKGROUND_DUMP_DEST 参数中找到。如果系统在安装时遵循 Oracle 的 OFA 目录结构，则跟踪文件应该在 $OARCELE_BASE/admin/SID/bdump 目录下，NT 系统则在 %ORACLE_HOME\admin\SID\bdump 目录下。这里 SID 是 Oracle 实例名字。跟踪文件的名字与后台进程有一定的关系。见下表：

表：后台进程与跟踪文件名

进程名字	UNIX 系统	NT 系统

Process Monitor(PMON)	Pmon_nnnn.trc	SidPMON.trc
System Monitor(SMON)	Smon_nnnn.trc	SidSMON.trc
Database writer(DBWR)	Dbwo_nnnn.trc	SidDBWO.trc
Log Writer(LGWR)	Lgwr_nnnn.trc	SidLGWR.trc
Checkpoint(CHPT)	CKPT_nnnn.trc	SidCKPT.trc
Archive Process(ARCO)	Arco_nnnn.trc	SidARCO.trc

§16.2.2 事件跟踪

Oracle8i 服务器可以为特殊的**数据库事件**而设置成为收集详细的跟踪信息。数据库事件是一个特殊的活动或数据库中行为。你可以将数据库的这种活动信息的故障和性能部分向 Oracle 的全球支持中心（OWS=Oracle Worldwide Support）询问。

一般事件的跟踪需要在 INITsid.ORA 初始参数文件中指定。如：

EVENT=”10046 trace name content forever, level 12”

引号内的参数是实际的跟踪参数。它由两部分组成：
* 跟着事件的号（如 10046）；
* 执行的级别(如 12)；
现在这些在文档不提供，而是由 Oracle 的全球支持中心 提供。

§16.2.3 用户跟踪文件

用户的跟踪文件由初始化参数文件的 USER_DUMP_DEST 参数来指定。如果系统在安装时遵循 Oracle 的 OFA 目录结构，则跟踪文件应该在 $OARCELE_BASE/admin/SID/udump 目录下，NT 系统则在 %ORACLE_HOME\admin\SID\udump 目录下。这里 SID 是 Oracle 实例名。
用户跟踪文件的名字与实例名与连接时的进程序号有关，如：

* UNIX 系统：ora_prod_4327.trc
* NT 系统：ora00117.trc

可以从 V$SESSION 和 V$PROCESS 动态视图中查出用户当前进程所对应的跟踪文件名。如：

```
SQL> select s.username, p.spid from v$session s , v$process p
  2  Where  s.paddr = p.addr and  p.background is null;

USERNAME                       SPID
------------------------------ ---------
                               246
                               247
                               248
```

	249
SYSTEM	258

从查询可以看出，用户 system 所产生的跟踪文件是 ora_prod_258.trc，或 ora0258.trc。

会话跟踪也可以从多线程服务器、事物管理器中来监视。

用户跟踪：

Oracle 有时也在用户的服务处理出错时建立用户跟踪。比如出现死锁时可创建如下的用户跟踪文件：

。 。 。 。 。 。
*** 2000.08.15 09:49:21.799
*** SESSION ID ☹7.25) 2000.08.15 09:48.21.719
ksqdedl: deadlock detected via did
DEADLOCK DETECTED
Current SQL statement for thie session:
Update apps.customer set name='Acme' where name='XYZ'
The following deadlock is not an Oracle error .It is a deadlock
Due to user error in the design os an application or information
May aid in detectming the deadlock:
Deadlock grph:

实例级跟踪：

如果在初始化参数文件设置 SQL_TRACE=TRUE，则实例中的所有后台进程都建立各自的跟踪文件。这样的设置要谨慎，因为这样的处理要大量的资源。

用户级的自我跟踪：

可以在 SQL 下设置自我跟踪：
SQL>alter session set SQL_TRACE=TRUE
或
SQL>alter session set SQL_TRACE=FALSE

用户级的 DBA 跟踪：

可以使用 Oracle 的 PL/SQL 包 DBMS_SYSTEM 来跟踪用户的情况。步骤如下：

1. 从 V$SESSION 中查出用户的 sid 和 serial#
SQL>select username, sid, serial# from v$session where username is not null;

```
USERNAME                       SID     SERIAL#
------------------------------ --------- -------------------
SYSTEM                           6     2301
SCOTT                           10     2642
```

2.可以用下面命令来跟踪 scott 用户的会话：

```
SQL>EXECUTE sys.dbms_system.set_trace_in_session( 10,  2642 , TRUE );

PL/SQL procedure successfully completed.
```

3.现在，Scott 会话就产生一个跟踪文件到 USER_DUMP_DEST 所指的目录中。如果跟踪完成或随时停止跟踪的话，则用：

```
SQL>EXECUTE sys.dbms_system.set_trace_in_session( 10,  2642 ,  FALSE );

PL/SQL procedure successfully completed.
```

解释用户跟踪：

如果产生了用户跟踪文件，可以用 TXPROF 实用程序对其进行解释。见 OCP chapter 3。

§16.2.4 管理跟踪文件

跟踪文件随着时间的推移，可以变得很大，如果不加限制的话很可能用光硬盘空间。可以在初始化参数文件中加 MAX_DUMP_FILE_SIZE 参数来限制跟踪文件增长的大小。该参数可以是块或字节，看下表：

参数指定	用户跟踪文件的最大值
MAX_DUMP_FILE_SIZE=10000	表示 10000 个操作系统块
MAX_DUMP_FILE_SIZE=500K	表示 500,000 字节
MAX_DUMP_FILE_SIZE=10M	表示 10M 字节
MAX_DUMP_FILE_SIZE=unlimited	表示对文件大小无限制

与警告文件类似，跟踪文件也可以被裁剪或删除或移动地方，也可以重新命名。

§16.3 性能调整视图

有两种主要的视图，分别以 V$开头动态性能视图和以 DBA 开头的数据字典视图。下面是这两类视图的区别：

- V$ 开头的是单一的名字，而 DBA 都带有复数的名字。
- 多数的 V$开头的视图在数据库 nomount 或 mount 状态都有效，而 DBA 开头只是在数据库处于 open 状态才有效。
- 在 V$中的数据一般都是小写，而 DBA 中的数据都是大写。
- V$包含的动态性能数据在关闭系统后就丢失，它们总是包含自数据库系统启动以来的统计数据。而 DBA 包含固定的数据。

§16.3.1 常用性能优化视图 V$_

Oracle8i 的不同版本的 V$动态视图不一样多，8.1.5 有 175 个;8.1.6 有 183 个；所有的 V$视图可以从 V$的视图中列出。而 V$FIXED_TABLE 视图中的 X$开头的视图是可修改的内部数据结构。它可以被 SQL 语句修改。因此，只有在数据库处于 NOMOUNT 或 MOUNT 时才能对 X$表的内容进行查询。

视图名字	说明
V$FIXED_TABLE	列出当前发行的固定对象的说明
V$INSTANCE	显示当前实例的状态
V$LATCH	列出锁存器的统计数据
V$LIBRARYCACHE	有关库缓存性能的统计数据
V$ROLLSTAT	列出联机的回滚段的名字
V$ROWCACHE	显示活动数据字典的统计
V$SGA	有关系统全局区的总结信息

V$SGASTAT	有关系统全局区的详细信息
V$SORT_USAGE	显示临时段的大小及会话，可以看出哪些进程在进行硬盘排序
V$SQLAREA	列出共享区的统计。包括每个 SQL 串有一个行。提供 SQL 语句在内存、分析 及执行准备的统计。文本限制在 1000 个字符内， 整个文本从 V$SQLTEXT 中剪出 64 个有效的字节。
V$SQLTEXT	在 SGA 中属于共享 SQL 光标的 SQL 语句内容。
V$SYSSTAT	包括基本的实例统计数据
V$SYSTEM_EVENT	包括一个事件的总等待时间
V$WAITSTAT	列出块竞争统计数据，只有当时间统计参数被“使能”时系统才能对其更新。

§16.3.2 常用性能优化视图 DBA_

Oracle8i 的最新版本有近 190 个 DBA 数据字典视图，这些视图都是具体的物理表，而象 X$是个虚表。这些表无正规的文档说明，但确是意义深长的表，比如 OBJ$和 FILE$。但是，大多数情况下，我们只用带 DBA 的表。如：

视图名字	说明
DBA_TABLES	表的存储、行及块等的信息
DBA_INDEXES	索引的存储、行及块的信息
INDEX_STATS	索引深度及差量（dispersion ）信息
DBA_DATA_FILES	数据文件位置、名字及大小信息
DBA_SEGMENTS	数据库中任意能占用空间的对象的有关信息
DBA_HISTOGRAMS	定义信息的直方图

§16.3.3 视图查询例子

在 Oracle 的 V$视图中，许多视图都是互相有联系的。一般来说，一个视图存放每个事件的部分，而另一部分信息存放在另外的视图上，这样在查询信息时就要进行必要的连接才能得到完整的信息。如：

例 1：查询关于 scott 用户的统计信息：

```
select  s.username ,  n.name , t.value
from  v$session  s , v$statname  n , v$sesstat  t
where  s.sid = t.sid  and t.statistic# = n.statistic#  and s.username='SCOTT';
```

例 2：查询关于 scott 用户的表信息：

```
select  table_name , chain_cnt  from  dba_tables where  owner='SCOTT' and chain_cnt != 0 ;
```

§16.4 Oracle 支持的调整脚本

Oracle 支持几种脚本用于调整系统性能。在 UNIX 环境下，这些脚本在 $ORACLE_HOME/rdbms/admin 目录中。在 NT，这些脚本存放在 %ORACLE_HOME%\rdbms\admin 目录中。通常用于性能调整的脚本有：

- UTLBSTAT.SQL/UTLESTAT.SQL
- UTLLOCKT.SQL
- CATBLOCK.SQL
- CATPARR.SQL
- DBMSPOOL.SQL
- UTLCHAIN.SQL
- UTLXPLAN.SQL

有些脚本用于建立表或视图等，这样的脚本只能运行一次。而有些是用于进行数据的统计，这样的脚则需要经常运行。如 UTLBSTAT.SQL 和 UTLESTAT.SQL 就是用于统计系统的某个时间段的使用情况。要想得到当前某时间段的系统使用情况，则不能从旧的视图查结果，而是要重新运行这两个脚本。

§16.4.1 UTLBSTAT.SQL 与 UTLESTAT.SQL

Oracle 的 UTLBSTAT.SQL/UTLESTAT.SQL是两个最有用的脚本。UTLBSTAT.SQL 是用于开始启动系统的统计；而 UTLESTAT.SQL 则是用于结束系统的统计。

运行 UTLBSTAT.SQL

运行 UTLBSTAT.SQL 脚本将进行下面的动作：

- 删除原来由 UTLBSTAT.SQL 所创建的表，如果这些不存在，则提示表或视图不存在，并直接建立统计所用的表。
- 建立一些临时表以存放从 V$中拷贝过来的数据，比如建立 stat@begin_event 表来存放 V$system_event 表的拷贝。
- 用下面命令来启动 UTLBSTAT.SQL 脚本的运行：

SQL>@$ORACLE_HOME/rdbms/admin/utlbstat.sql

运行 UTLESTAT.SQL

通过一段时间运行 **UTLBSTAT.SQL** 脚本之后，一般至少 15 分钟，就可以启动运行 **UTLESTAT.SQL** 了。一般建议应该运行一天后在来运行 UTLESTAT.SQL。当 UTLESTAT.SQL运行后，它要进行能下面工作：

- 第一，删除任何由 UTLESTAT.SQL 创建的表。所以看到"Table or view does exist" 是正常的；
- 第二，UTLESTAT.SQL脚本创建一些临时的表，以存放从 V$中拷贝的数据。

- 第三，计算 stats$begin 和 stats$end 中的数据。
- 最后，产生一个文本文件叫 report.txt，该文件包含所以调整所需计算结果信息。

可以用下面命令来启动 UTLESTAT.SQL 脚本的运行：

SQL>@$ORACLE_HOME/rdbms/admin/utlestat.sql

提示：如果在运行 UTLESTAT.SQL 和 UTLESTAT.SQL 期间出现数据库关机，则原来的统计数据无效。

§16.4.2 解释 REPORT.TXT 内容

这里所给出的统计结构对于调整非常有用，但指的是对有着实际经验的 DBA 来说。如果一般的新手，可以从 http://www.oracle.com/oramag/oracle/00-Mar/o20tun.html 来得到详细的解释。

§16.5 Oracle 的 STATSPACK

§16.6 图形性能调整工具

Oracle8i 新版本提供基于图形的管理工具，这些工具包括性能调整显示，主要有三个：

- Oracle 企业管理器/Oracle DBA 管理包
- Oracle 诊断包
- Oracle 调整包

§16.6.1 Oracle 企业管理器/Oracle DBA 管理包

Oracle 的企业管理器与数据库管理员管理包一起组成 Oracle 管理数据库工具的核心。OEM 用于监视系统，而用 DBA 管理包工具来管理数据库。DBA 管理包(DBA Management Pack)由 DBA Studio 和 SQL*PLUS Worksheet 组成。下面是 DBA 管理包及其用途：

部件	目的
Instance manager	启动、停止和管理实例
Schema Manager	建立和管理 Oracle 对象
Security Manager	建立和管理 Oracle 用户和角色
Storage Manager	建立和管理表空间及数据文件
SQL*PLUS Worksheet	在图形下使用 SQL*PLUS

§16.6.2 OEM 使用*

Oracle 企业管理器提供了一些有关性能优化的管理功能，可以从图形界面中了解系统运行的情

况。具体的使用方法另见其它资料。这里略去。

第 17 章　STATSPACK 工具的使用

Oracle 8i/9i 版本支持 statspack 工具。

§17.1 STATSPACK 介绍

§17.1.1 STATSPACK 与 UTLBSTAT/UTLESTAT 比较

- Statspack搜集更多的信息;
- Statspack预先计算更多的调整有用的比率信息，如：cache hit ratios, rates, and transaction statistics;
- 由PERFSTAT拥有的永久表来存放性能信息。可替代每次都得创建/删除表；
- Statspack从报告中分开数据的搜集。当快照发生时数据被搜集；
- 数据的搜集可用DBMS_JOB或操作系统完成；
- Statspack 在COMMIT或ROLLBACK事务后计算，而BSTAT/ESTAT只在事务提交时考虑；

注，如果在关联到Statspack时，选择运行BSTAT/ESTAT则不要在同用户上进行，因为表名为STATS$WAITSTAT 冲突。

§17.1.2 STATSPACK 工作流程

当运行脚本时，自动创建 PERFSTAT 用户。PERFSTAT 用户拥有所有的对象，并被授权访问 V$_视图。PERFSTAT 变成快照的一个家族。专门搜集性能的数据。每个快照都由一个 ID 来标识。每次搜集都产生一个新的 SNAP_ID。SNAP_ID 带有数据库标识(DBID)和实例编号（INSTANCE_NUMBER）。

§17.2 配置 STATSPACK

所有的Statspack 的表和索引的存储参数中的INITIAL要为 100k， next 要为1M 或5M。安装需要约64MB 。

- 如果在字典类型管理表空间上安装，你要监视空间的使用；
- 如果在本地类型管理表空间上安装，不需要存储参数项。

§17.3 statspack 安装

§17.3.1 交互安装 STATSPACK

1.创建PERFSTAT用户；
该用户包含有所有的PL/SQL代码和数据库对象，及STATSPACK表、约束、及STATSPACK包。在安装过程中，你被提示选择用户的默认表空间等。

****

注:

- 不要指定system、TEMP表空间为PERFSTAT用户的默认表空间。否则会出现错误。
- 如果已指定，请运行SPDROP.SQL脚本删除PERFSTAT用户的对象。
- 在安装过程中，要建立DBMS_SHARED_POOL和 DBMS_JOB

****

安装 脚本SPCREATE.SQL文件在ORACLE_HOME/rdbms/admin目录下。以SYSDBA进入：

在 UNIX:
SQL> CONNECT / AS SYSDBA
SQL> @?/rdbms/admin/spcreate

在 NT:
SQL> CONNECT / AS SYSDBA
SQL> @%ORACLE_HOME%\rdbms\admin\spcreate

SPCREATE 脚本还需要调用3个脚本：

- SPCUSR:建立用户和授权；
- SPCTAB:建立表；
- SPCPKG建立包。

安装后，检查输出文件SPCUSR.LIS, SPCTAB.LIS。

§17.3.2 批模式安装 STATSPACK

运行SPCREATE脚本前，分配变量，指定默认表空间。

- DEFAULT_TABLESPACE:默认表空间
- TEMPORARY_TABLESPACE:临时表空间

在 UNIX:
SQL> CONNECT / AS SYSDBA
SQL> define default_tablespace='tools'
SQL> define temporary_tablespace='temp'
SQL> @?/rdbms/admin/spcreate

§17.4 使用 statspack

§17.4.1 取得 STATSPACK 快照

以PERFSTAT用户登录 并执行STATSPACK.SNAP存储过程。如：

SQL> CONNECT perfstat/perfstat
SQL> EXECUTE statspack.snap;

为了得到性能分析，需要在INIT.ORA文件设置：

TIMED_STATISTICS = true

也可SQL>下动态地用ALTER SYSTEM 命令修改TIMED_STATISTICS 参数。

注：为了安全，完成设置后，最好修改 PERFSTAT用户的口令。
另外，在RAC环境中，你必需连接到需要统计的实例中。

注: 在 Oracle9i Release 2 (9.2) 或更高的版本,如果当设置STATISTICS_LEVEL = TYPICAL 或 ALL 时，系统自动进行时间统计。如果当设置STATISTICS_LEVEL = BASIC,则你必修设置时间统计参数TIMED_STATISTICS=TRUE以启动时间统计 。如果你已明确设置 DB_CACHE_ADVICE, TIMED_STATISTICS或TIMED_OS_STATISTICS(无论是在INIT.ORA 或是用ALTER SYSTEM)。这些显式值都覆盖原来的STATISTICS_LEVEL的设置。

特别是，你可以在 PL/SQL 中调用 STATSPACK.SNAP 函数来显示 SNAP_ID，如：

下面是一个PL/SQL匿名块：

SQL> variable snap number;

```
SQL> begin :snap := statspack.snap; end;
begin :snap := sys.statspack.snap; end;
PL/SQL procedure successfully completed.

SQL> print snap
SNAP
----------
12
```

§17.4.2 自动进行统计搜集

要进行天、周、年的比较，你需要多个快照以接收时间周期。最好的方法是自动交替地搜集，这样，你要进行到底下面步骤：

- 在数据库使用 DBMS_JOB 存储过程度确定快照时间表。
- 使用操作系统实用程序，如 UNIX 的 cron 、NT 的 at 等。

1．使用DBMS_JOB搜集统计数据

用DBMS_JOB包是Oracle搜集统计数据的自动方法。一个类似的脚本是SPAUTO.SQL 。
为了使DBMS_JOB有效，要在INIT.ORA中设置JOB_QUEUE_PROCESSES = n 。n是一个大于0的数。

2.改变统计间隔时间

使用 DBMS_JOB.INTERVAL 过程可设置搜集统计数据的间隔。如：

```
EXECUTE DBMS_JOB.INTERVAL(1,'SYSDATE+(1/48)');
```

这里的 'SYSDATE+(1/48)' 每半小时搜集一次。

强行运行作业：

```
EXECUTE DBMS_JOB.RUN(<job number>);
```

删除自动搜集作业：

```
EXECUTE DBMS_JOB.REMOVE(<job number>);
```

§17.4.3 运行 statspack 性能报告

产生快照后，要给出两个ID号（开始和结束）和报告名。

- 运行SPREPORT.SQL产生报告
- 检查实例报告

在UNIX:

```
SQL> connect perfstat/perfstat
SQL> @?/rdbms/admin/spreport
```

在 NT:

```
SQL> connect perfstat/perfstat
SQL> @%ORACLE_HOME%\rdbms\admin\spreport
```

样本输出:

```
SQL> connect perfstat/perfstat

Connected.
SQL> @spreport

DB Id        DB Name   Inst Num   Instance
-----------  ------------  --------   ------------
2618106428 PRD1        1              prd1
Completed Snapshots

                           Snap                  Snap
Instance   DB Name Id Snap Started Level     Comment
------------ ------------ ----- ----------------- ----- ----------------
prd1          PRD1 1 11 May 2000 12:07 5
                        2 11 May  2000  12:08 5
Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 1
Begin Snapshot Id specified: 1
Enter value for end_snap: 2
End Snapshot Id specified: 2
Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~~~
The default report file name is sp_1_2 To use this name, press <return> to
continue, otherwise enter an alternative. Enter value for report_name:
<press return or enter a new name>
Using the report name sp_1_2
```

这个报告也可写到一个文件中，如 sp_1_2.lis

变量为:

- BEGIN_SNAP: 指定开始的快照 ID
- END_SNAP: 指定结束的快照ID
- REPORT_NAME: 指定报告名

例子：UNIX下建立报告的例子：

SQL> connect perfstat/perfstat
SQL> define begin_snap=1
SQL> define end_snap=2
SQL> define report_name=batch_run
SQL> @?/rdbms/admin/spreport

§17.4 删除 statspack

要删除 STATSPACK，要以 SYSDBA 权限进行登录：

SQL>CONNECT /AS SYSDBA
SQL>@spdrop

Spdrop 脚本调用另外的两个脚本，即：

- PDTAB – 删除表和公共同义词；
- SPDUSR—删除用户。

在删除完成后，请检查两个输出结果：

SPDTAB.LIS 和 SPDUSR.LIS 文件。

§17.5 statspack 支持的脚本和文档

Statspack安装的脚本

下面的脚本要以 SYSDBA 权限运行:

- SPCREATE.SQL:

建立整个STATSPACK (调用SPCUSR.SQL, SPCTAB.SQL, SPCPKG.SQL)

- SPDROP.SQL:

删除整个STATSPACK (调用SPDTAB.SQL,SPDUSR.SQL)

下面的脚本要以 SYSDBA 权限运行:

- SPDTAB.SQL: 删除 Statspack 表
- SPDUSR.SQL: 删除 Statspack 用户 (PERFSTAT)

下面的脚本要以PERFSTAT运行:

- SPCUSR.SQL: 建立 Statspack用户 (PERFSTAT)
- SPCTAB.SQL: 建立 Statspack 表
- SPCPKG.SQL: 建立 Statspack 包

Statspack报告和自动运行的脚本

下面的脚本要以PERFSTAT运行:

- SPREPORT.SQL: 产生一个Statspack 报告
- SPREPSQL.SQL: 产生一个 Statspack SQL报告(带 hash 值)
- SPREPINS.SQL: 产生一个 Statspack 数据库和实例的报告
- SPAUTO.SQL: 自动 Statspack 统计 (使用 DBMS_JOB 包)

Statspack升级脚本

下面的脚本要以SYSDBA运行:

- SPUP90.SQL: 从 9.0 模式到 9.2 模式，并备份存在的模式
- 如果从 8.1.7升级到9.2, 则先运行SPUP817.SQL
- 如果从 8.1.6升级到9.2, 则先运行SPUP816.SQL

Statspack数据字典管理的脚本

下面的脚本要以PERFSTAT运行:

- SPPURGE.SQL: 净化数据库实例的快照ID的限制范围；
- SPTRUNC.SQL: 清除Statspac表的所有数据；
- SPUEXP.PAR: 导出整个PERFSTAT用户的参数文件。

Statspack 文档

SPDOC.TXT
此文件包含Statspack包的使用介绍。

第 18 章　动态性能视图与性能诊断

动态性能视图（以V$开头的数据字典）对于鉴别实例级性能问题来说是非常有用。所有的带V$的视图都可以从 V$FIXED_TABLE视图中列出。而V$FIXED_TABLE视图中的X$开头的视图是可修改的内部数据结构。所以这些表只是在实例处在NOMOUNT 或 MOUNT 状态是才有效。

§18.1 当前会话状态视图

下面的视图都是在当连接成功后就开始收集会话级信息。

视图名字	说明
V$LOCK	列出当前由 Oracle8 服务器及请求的锁。
V$MYSTAT	显示当前会话的统计数据。
V$PROCESS	包括当前活动的信息。
V$SESSION	列出当前会话信息。包括行锁信息等。
V$SESSION_EVENT	列出会话中等待的信息。
V$SESSION_WAIT	列出会话正在等待的资源或事件，当 WAIT_TIME=0 时为当前会话。
V$SESSTAT	列出用户会话统计。需要与 V$STATNAME,V$SESSION 连接才能得到详细信息。

从V$SESSION_WAIT 容易查到实时的会话等待的信息。如：

```
SELECT SID,EVENT FROM V$SESSION_WAIT
WHERE WAIT_TIME = 0;
```

§18.2 计数和积累视图

这些视图自从数据库实例起动或会话开始以来一直跟踪和统计的积累数据视图。比如，由 Oracle 提供的 Statspack 和 BSTAT/ESTAT 等。

视图名	**说明**
V$DB_OBJECT_CACHE	**在共享池中对象级统计信息**
V$FILESTAT	**I/O活动文件级小结**
V$LATCH	**锁存器活动信息**
V$LATCH_CHILDREN	**子锁存器的锁存器活动信息**
V$LIBRARYCACHE	**共享池中空间级小结信息**
V$LIBRARY_CACHE_MEMORY	**库缓存当前内存信息**

V$MYSTAT	**会话资源使用情况信息**
V$ROLLSTAT	**回滚段活动小结信息**
V$ROWCACHE	**数据字典活动小结信息**
V$SEGMENT_STATISTICS	**段级实时监视DBA统计视图**
V$SEGSTAT	**段级实时监视统计高效视图**
V$SESSION_EVENT	**当前会话所有等待会话级小结**
V$SESSTAT	**从会话开始以来，资源使用会话级小结**
V$LIBRARY_CACHE_MEMORY	**共幸池LRU列表机制模拟**
V$SQL	**V$SQLAREA子光表详细信息**
V$SQLAREA	**语句/匿名块的共享池详细信息**
V$SYSSTAT	**资源使用小结**
V$SYSTEM_EVENT	**实例范围资源等待小结**
V$UNDOSTAT	**撤消使用的直方图,每性代表10 分钟**
V$WAITSTAT	**块类型的缓冲区等待停止**

§18.3 信息视图

这些视图在实例不是动态的。但是在调整中也需要查询。这些视图如下：

视图名字	说明
V$MTTR_TARGET_ADVICE	MTTR 顾问的建议信息，在启动 FAST_START_MTTR_TARGET 时用。
V$PARAMETER 和 V$SYSTEM_PARAMETER	实例范围的参数
V$PROCESS	服务器进程（前台或后台）
V$SEGSTAT_NAME	段级的特性统计视图
V$SQL_PLAN	当前执行的光执行计划
V$SQL_PLAN_STATISTICS	执行计划中的每个执行统计
V$SQL_PLAN_STATISTICS_ALL	在 V$SQL_PLAN 中来自 V$SQL_PLAN_STATISTICS 和 V$SQL_WORKAREA 执行统计的连接信息。
V$SQLTEXT	共享池中的 SQL 语句
V$STATISTICS_LEVEL	由 STATISTICS_LEVEL 初始参数设置的统计状态和建议控制
下面是数据库实例级的视图	
V$FIXED_TABLE	列出当前发行的固定对象的说明
V$INSTANCE	显示当前实例的状态
V$LATCH	列出锁存器统计数据
V$LIBRARYCACHE	有关库缓存性能的统计数据
V$ROLLSTAT	列出联机的回滚段的名字
V$ROWCACHE	显示活动数据字典的统计
V$SGA	有关系统全局区的总结信息
V$SGASTAT	有关系统全局区的详细信息
V$SORT_USAGE	显示临时段的大小及会话，可以看出哪些进程在进行硬盘排序
V$SQLAREA	列出共享区的统计。包括每个 SQL 串有一个行。提供 SQL 语句在内 存、分析 及执行准备的统计。文本限制在 1000 个字符内，整个文本 从 V$SQLTEXT 中剪出 64 个有效的字节。

V$SQLTEXT	在 SGA 中属于共享 SQL 光标的 SQL 语句内容。
V$SYSSTAT	包括基本的实例统计数据
V$SYSTEM_EVENT	包括一个事件的总等待时间
V$WAITSTAT	列出块竞争统计数据，只有当时间统计参数被“使能”时更新。

有关动态性能视图的解释见本教材附录部分。

§18.4 当前统计值与变化比率

在得到当前系统的一些使用统计数据后，还需要看它们的变化频率情况。

§18.4.1 当前统计值与变化比率

实例的统计的关键比率以项来表示。如：

```
COL name FORMAT a35
COL value FORMAT 999,999,990
SELECT name, value  FROM V$SYSSTATS
WHERE lower(NAME) LIKE lower('%&stat_name%') ;
```

一般都是将这样的脚本存为一个文件，然后在SQL>下启动运行即可。

§18.4.2 找出统计值的变化率

如果希望得到准确的统计数据，建议要重复同一个脚本多次(至少两次)，然后从中分析系统的各个时间段内的变化情况，比如将上面的脚本稍作优化。就得到：

```
SET VERI OFF
DEFINE secs=0
DEFINE value=0
COL value FORMAT 99,999,999,990 new_value value
COL secs FORMAT a10 new_value secs noprint
COL delta FORMAT 9,999,990
COL delta_time FORMAT 9,990
COL rate FORMAT 999,990.0
COL name FORMAT a30
SELECT name, value, TO_CHAR(sysdate,'sssss') secs,
(value - &value) delta,
(TO_CHAR(sysdate,'sssss') - &secs) delta_time,
```

```
(value - &value)/ (TO_CHAR(sysdate,'sssss') - &secs) rate
FROM v$sysstat
WHERE name = '&&stat_name'
/
```

§18.5 有计划地调整系统的因子

性能在整个系统中是相互有联系的，而不是孤立的。因此要对现有的系统进行调整，先要了解一些关键的因素。下表给出Oracle调整范围与资源限制情况：

ORACLE 调整范围	资 源 限 制				
	CPU	内存	I/O	网络	软件
1．Application					
设计/结构	x	x	x	x	x
DML SQL	x	x	x	x	x
查询 SQL	x	x	x	x	x
客户/服务	x			x	
2．Instance					
高速缓存	x	x	x		
共享池	x	x			
排序区	x	x	x		
数据及数据文件I/O的物理结构	x		x		
日志文件 I/O		x	x		
归档 I/O	x		x		
回滚段			x		x
锁	x	x	x		x
备份	x		x	x	x
3．Operating System					
内存管理	x	x	x		
I/O 管理	x	x	x		
进程管理	x	x			
网络管理		x		x	

§18.6 不足的 CPU

在有效的CPU范围内，CPU可以全部被分配，也可以整个服务时间都用光。无论哪一种情况，都需要改善系统的处理能力。要确定为什么CPU不够，就要识别整个系统是如何使用CPU。不要只依赖于CPU是如何被Oracle服务进程使用这样的观点。在工作一开始，可以给系统消耗大量的有效的CPU，然后在一天后，可以给系统较小的瓶颈并去掉CPU。当评估系统的CPU级的使用时，工作量是一个很重要的因素。工作量高峰的小时数，CPU使用90%而闲置为10%，则等待是可以接受，也可以理解。如果在某个时间内只有30%的工作量，则也是可以理解的。然而，如果你的系统在一个正常的工作量中有很高的利用，则表明你的系统在高峰期没有足够的空间了。如果在正常时间内或非高峰期，闲置时间和I/O等待时间 都接近于0 或小于5%，则

CPU 有问题。

§18.7 不足的内存

一般来说，如果服务器只安装操作系统和Oracle数据库系统，则只有两种类型的内存需求，即Oracle系统和操作系统。而Oracle系统的内存需求会影响到操作系统的需求。内存问题可以产生内存分页和页交换问题。所以要避免内存分页和页交换出现，就要使系统运行在实际内存的有限范围内。
操作系统的内存需求应该与Oracle的内存需求相当或略小些。要达到这一点，就要减少某些Oracle内存结构的大小。如高速缓存、共享池或重做日志缓冲区等。在系统级上，你可以减少进程数和/或每个进程所使用的内存量。你也可以识别出哪个进程使用内存最多，从而用共享SQL来减少内存的需求。

§18.8 I/O 限制

要确保所发布的I/O能均匀地通过硬盘和通道。I/O限制包括：

- 通道带宽：通道的数量。
- 设备带宽： 磁盘数量。
- 设备潜能：从初始请求到请求收到所用去的时间；潜能是等待时间的一部分。

I/O问题可以是由于硬件限制引起。你的系统需要足够的磁盘和SCSI总线接口以支持事务的吞感吐量。你也可以评估一下整个信息量及总线所支持的潜在能力，从而得到高峰时的工作量。
如果某个I/O时间响应变得过于慢，大多数都是等待时间增加（响应时间=服务时间+等待时间）。如果等待时间增加，则该设备有过多的请求。如果服务时间增加，则I/O请求较大，需要写更多的字节到硬盘。
不同的后台进程，如DBWR和ARCH等，执行不同类型的I/O，并且每个进程有不同的I/O特性。有些进程以数据库块大小进行读和写；有些进程则以更大块的数据量进行读和写。如果服务时间过高，则条纹（stripe）文件要通过不同的设备。镜像可能引起I/O瓶颈。除非数据只镜像到有相同的数量硬盘的目标数据库上。

§18.9 网络限制

网络也有类似的I/O问题，要考虑的有：

- 网络带宽：每个传输需要相应的包，如果知道每次传输的包的数量，则可以算出系统所支持的工作量的能力。
- 通讯速率：你可以在网络上以批方式减少包的数量。这样比多次发很小的包更好。

- 传输时间：

随着用户数的增加和需求的上升，网络可能变为应用的主要瓶颈。 你可以花大量时间来等待网络变为可用。可以使用操作系统工具查看你的系统是如何的忙。

§18.9 软件限制

操作系统软件可以确定：

- 可以支持的最大进程数；
- 可以连接的最大进程数。

当你有效的地对Oracle进行调整前，你应该确定操作系统是否处于最佳的运行状态。工作接近硬件的能力，然后软件系统管理员确保Oracle在合适的操作系统资源内来分配。

注：在NT没有最大进程数和连接进程数的概念。

第 19 章　调整内存分配

一般来说，在应用系统提供测试前就应该对系统环境进行必要的调整，这样主要是因为在安装完 Oracle 系统后，系统的环境默认值都是按照较小的参数来设置的。下面给出有关内存参数设置的基本介绍。
本章内容可参考《OCP:Oracle8iDBA 性能调整及网络管理》9 和《Oracle8iDesigning and Tuning for Performance》chapter 19.

§19.1 理解内存分配要求

每一个Oracle版本对内存都有特别的要求，一般在软件的发行上可以找到。Oracle的存储信息参数是指内存和硬盘的要求。但是，我们都知道，内存的存取速度比硬盘要快8至10倍。因而，用内存来存放数据更能满足快速请求的要求。可是，内存资源一般都比硬盘贵，另一方面，内存的配置也是有限的。所以调整内存的分配以有效地利用内存是我们的一项工作。
由于Oracle的内存要求与应用有关，所以一般内存的调整是在应用和SQL语句做完调整之后进行。但是如果你是在应用和SQL语句调整前就调整了内存分配，那么在你修改完应用和SQL语句后还需要对Oracle的内存结构进行再调整。
另外，也建议在调整I/O前先调整内存分配。调整内存分配以建立Oracle进行I/O操作时所必需的内存总量。

下面是本章要讨论的题目：
块（Block） 内存和硬盘交换数据的单位。
缓冲区（Buffer） 内存地址。
缓冲区池（buffer pool） 缓冲区的累积。
高速缓存或高速缓冲区（cache or buffer cache）所有的缓冲区和缓冲区池。
段（segment） 特殊数据库类型（表，索引，簇）的一组扩展分配。

§19.2 监测内存分配问题

在 Oracle 运行当中，我们可以测试 Oracle 的进程，比如，在 UNIX 下，可以用：
ps -efl 或 ps –aux 来显示 Oracle 进程的情况。从显示的数据可以知道各个进程的大小。如进程的大小，分配的共享内存，堆、可执行堆栈及实际给出分配数量等。如：
SZ 给出分配页的大小（通常是 4KB）

SZ　　+20,000
Minus SHM　-15,000
Minus EXEABLE　-1,000

Actual per-process memory 4,000

§19.3 调整操作系统内存需求

调整操作系统内存主要目标是：

- 减少页分配和交换
- 适宜的系统全局区(SGA)
- 分配足够的内存给每个用户

1．减少页分配和交换

操作系统在下面的区里存储信息：

- 实际内存（Real memory）
- 虚拟内存（Virtual memory）
- 扩展存储（Expanded storage）
- 硬盘（ Disk）

操作系统可以将信息从一个存储位置移到另一个存储位置，这样就叫页分配交换（*paging* 或 *swapping*）。多数操作通过页分配和交换在不分配到实际的物理内存就能以适应大量信息的要求。然而，**过度的页分配和交换会降低操作系统的性能**。是否存在过度的页分配和交换，可以使用操作系统实用程序来监示。

2．适宜的系统全局区

SGA区总是存放在主存中，它的用途就是存放快速访问的数据。如果SGA区被交换到硬盘。那么数据就不会被快速访问。值得注意的是，大多数操作系统下，如果SGA过大而产生页交换对系统来说是不好的。虽然在SGA中都存放的是数据，但还分为热和冷两部分。热的部分总是在内存里，而冷的部分总是被记录在内存外。当热的部分不被保留在SGA区时，性能问题就会发生。

数据被交换到硬盘是因为它不被引用。你可以修改参数文件的PRE_PAGE_SGA 为 YES使得在实例启动时，让Oracle将所有的SGA放到内存里。这样的设置会在启动时需要更长的时间。同时也延长进程的启动时间。因为每个进程启动都与SGA区挂钩。这样的成本是固定的。然而，对于大多数操作系统，你只能为每个进程分配20,000页，这样的数量比较接近大多数的应用。

PRE_PAGE_SGA（SGA区页大小）的好处可以提供页的大小，如SGA为80MB,页的大小为4KB，则 80,000,000/4,000=20,000 为页更新大小。

如果操作系统允许使用到4MB页大小，则80,000/4,000 = 20，即更新SGA区只能是20页。

你可以用 show SGA来显示SGA的大小。如：

```
Total System Global Area 18847360 bytes
Fixed Size               63104 bytes
```

```
Variable Size          14155776 bytes
Database Buffers        4096000 bytes
Redo Buffers             532480 bytes
```

如果你的Oracle版本为较低的版本，可用下面查询语句得到SGA的信息：

```
select * from V$SGA;
```

3．分配足够的内存给每个用户

在一些操作系统中，你可以控制分配给用户的内存数量。因而，作为DBA就要了解每个用户实际对内存的需求情况。关心下面参数：

- Oracle执行程序大小
- SGA大小
- Oracle 应用工具
- 应用使用的特殊数据

§19.4 调整 Redo Log Buffer

Log_Buffer 参数是存放日志缓冲区的地方。当该区被填满时由LGWR(日志写进程)来将数据写到硬盘中。由于这样的原因，较大的日志缓冲区就很少出现冲突。

§19.4.1 观察 Redo Log Buffer 是否有竞争

当LGWR进程将日志缓冲区（redo log buffer）的事件写到日志文件（log file）或硬盘时，用户可能在这时又拷贝了新的事件覆盖掉内存中的事件。这样就产生了等待的冲突。
我们可以从 V$SYSSTAT、V$SESSION_WAIT 中查询到冲突信息。下面是查询的方法介绍：

§19.4.1.1 使用 V$SYSSTAT

数据库在运行当中，系统自动将所有系统运行情况（包括日志数据）记录在 V$SYSSTAT 字典上，下面是数据字典的结构：

```
SQL> desc v$sysstat
 名称                      空?       类型
 ------------------------- --------- -------------
 STATISTIC#                          NUMBER
 NAME                                VARCHAR2(64)
 CLASS                               NUMBER
 VALUE                               NUMBER
```

STATISTIC# 为统计编号
NAME 统计项的名称，可能为 redo blocks written 等
CLASS 统计的类，1 为 user, 2 为 redo, 4 为 enquenue, 8 为 cache, 16 为 os,32 为 parallel Server,64 为 SQL,128 为 debug。
VALUE 统计值

在这个数据字典里，name 分别为不要同的参数和对应不同的值。你可以用：

SQL>select dsitinct name from v$sysstat;

来列出 name 列的所有字符串值。

如：

SQL> select distinct name,value from v$sysstat where class=2 ;

NAME	VALUE
redo blocks written	44
redo buffer allocation retries	0
redo entries	74
redo log space requests	**0**
redo log space wait time	0
redo log switch interrupts	0
redo ordering marks	0
redo size	17348
redo wastage	4472
redo write time	0
redo writer latching time	0
redo writes	18

已选择 12 行。

现在我们关心的是有关重写日志的空间要求，那么就要查看 name ='redo log space requests'对应的值。也就是说，如果 name='redo log space requests '对应的 value 不为 0，则应该增大 log_buffer 的设置以支持加载事务，避免事务被迫等待访问重做日志区。

例 1：

SQL> select name,value from v$sysstat where name='redo log space requests';

NAME	VALUE
redo log space requests	0

SQL>

如果此项查询 value 为 0，则说明没有日志空间请求。如果大于 0 则说明有日志空间请求。

例 2：

```
SQL>  select retries.value/entries.value "redo buffer retry ratio"
  2  from v$sysstat  retries,V$sysstat entries
  3  where retries.name='redo buffer allocation retries'
  4* and entries.name='redo entries';

redo buffer retry ratio
-----------------------
               .000345
```

§19.4.1.2 使用 V$SESSION_WAIT

与 V$SYSSTAT 不一样，V$SESSION_WAIT 反映的是整个实例的统计数据。V$SESSION_WAIT 反映的是事件，个别用户等待的时间多少。对于日志缓冲区来说，就是显示用户在会话中等待空间有多长。先看 V$SESSION_WAIT 字典的结构：

```
SQL> desc v$SESSION_WAIT
 名称                                      空?    类型
 ----------------------------------------- -------- ----------------
 SID                                       NUMBER
 SEQ#                                      NUMBER
 EVENT                                     VARCHAR2(64)
 P1TEXT                                    VARCHAR2(64)
 P1                                        NUMBER
 P1RAW                                     RAW(4)
 P2TEXT                                    VARCHAR2(64)
 P2                                        NUMBER
 P2RAW                                     RAW(4)
 P3TEXT                                    VARCHAR2(64)
 P3                                        NUMBER
 P3RAW                                     RAW(4)
 WAIT_TIME                                 NUMBER
 SECONDS_IN_WAIT                           NUMBER
 STATE                                     VARCHAR2(19)

SQL> select distinct state from v$session_wait;
```

STATE

WAITED UNKNOWN TIME
WAITING

关于各个列的意义请参考 《Oracle8i/9i Reference》

下面给出例子来说明是否存在等待的查询：

例 3：查看 V$SESSION 和 V$SESSION_WAIT 来显示用户等待的时间：

SQL>select username, wait_time, seconds_in_wait
From V$SESSION_WAIT, V$SESSION
Where v$SESSION_WAIT.SID=V$SESSION.SID
And event like 'log buffer space';

USERNAME	SECONDS_IN_WAIT	STATE
JOE	20	WAITING
MATT	213	WAITED SHORT TIME

从查询结果来看，只要 SECONDS_IN_WAIT 大于 0 就存在等待。

这里 state 列的值可以分别为：

WAITING	会话等待的时刻
WAITED UNKNOWN TIME	不确定的最后等待时间
WAITED SHORT TIME	最后等待的 1/100 秒
WAITED KONOWN TIME	在 WAIT_TIME 列中实际等待的时间。

§19.4.1.3 使用 REPORT.TXT 输出

Oracle 提供了 UTLBSTAT.SQL 和 UTLESTAT.SQL 两个脚本，用于统计 Oracle 系统在某个时间点内的运行情况。这里由于篇幅所限，请 见另外章节 。

§19.4.1.4 使用 OEM 监示 Redo Log Buffer*

重做日志缓冲区的性能可以使用 Oracle 提供的 OEM 诊断包(OEM Diagnostics pack)进行监示。点击开始=》。。。Oracle Performance Manager ； 然后选择 Memory ,选择 Memory Statstics Char 即可显示图表。

§19.4.1.5 使用 V$SYSTEM_EVENT

有时，检查点事件(checkpoint event)启动后，但可能不完全成功，这是因为在第 1 点快完前有第 2 个检查点产生。这样的情形，你可以查询 V$SYSTEM_EVENT 字典，看是否有检查点开始时请求还没有完成的情况。这个视图报告的是自从实例启动以来的事件变化：

```
SQL>select  event, total_waits, average_wait
From  V$system_event
Where  event  like  'log file switch%';
```

V$system_event 视图的列名解释如下：

列名	说明
EVENT	等待事件的名
TOTAL_WAITS	等待当前事件的时间
TIME_WAITED	等待事件总时间（百分之一秒）
AVERAGE_WAIT	平均等待事件的时间（百分之一秒）

§19.4.1.6 使用警告日志（Alert Log）

经过日志也包括很重要的检查点信息，当找到经过日志后，将光标定位到 redo log 上。下面信息表示**重做日志轮换过快**并且**没有足够的时间**正常完成工作：

```
Thread 1 advanced to log sequence 293
Current log# 1  seq# 293 mem# 0: /u03/oradata/PROD/log1.dbf
Current log# 1  seq# 293 mem# 1: /u04/oradata/PROD/log1.dbf
Thread  1 cannot allocate new log, sequence 294
Checkpoint not complete
```

§19.4.2 调整 Redo Log Buffer 性能

当我们知道如何监示和查询重做日志的竞争数据后，接下来就是调整相应的参数以提高系统的性能。下面是常用的几种方法：

§19.4.2.1 将 Redo Log Buffer 参数改大

最简单的改善性能的方法就是加大 redo log buffer 参数的大小。较大的 redo log buffer 可以使用户在写入缓冲区时很少等待。LOG_BUFFER 参数的大小是以字节计。默认值是 500k 或 128k*CPU 个数。

```
SQL> show parameter log_buffer

NAME                 TYPE        VALUE
-------------------------------- ----------- -----
log_buffer           integer     524288
```

在 Oracle9i,要修改 log_buffer 参数,则在 init.ora 文件中修改然后用 init.ora 启动.

提示：如果你给出的 LOG_BUFFER 的值过低，Oracle 服务器会跳过你的设置而自动使用默认值。

§19.4.2.2 改善检查点效率

检查点事件在数据库里是一个精深的 I/O 问题。一旦检查点被启动，你就应该让它完全成功。如果不成功，检查点事件就终止引起不必要的 I/O。
如果在 V$SYSTEM EVENT,REPORT.TXT 或 Alrt Log 显示出**日志没有完成**。请你试图**加大日志成员的大小**以使它们被填多些和轮换稍慢些。
你也可以使用参数文件 INITsid.ORA 中的 LOG_CHECK_POINT_TIMEOUT 和 LOG_CHECKPOINT_INTERVAL 参数来调节。LOG_CHECK_POINT_TIMEOUT 参数在 Oracle8i 中默认是 900 秒，在 Oracle8i 企业版是 1800 秒。省略该参数表示 0（即没有限制）。而 LOG_CHECKPOINT_INTERVAL 参数的值与操作系统有关。允许的范围是 1 到 无限。

§19.4.2.3 加速归档处理

如果是归档方式，则 LGWR 进程需要写日志。而日志信息又由 ARCO 归档进程写到归档文件中，如果 LGWR 要写的日志文件被 ARCO 归档进程正在使用，则 LGWR 要等到 ARCO 完成后才能对该日志文件进行日志写。为了减少这种可能的情况发生，LGWR 应该在系统需要它时自动启动。可以通过加大归档进程来加速归档的处理。归档进程的数量可以在 INITsid.ORA 文件的 LOG_ARCHIVE_MAX_PROCESSES 参数来设置。该值可以设 1 到 10 之间。

§19.4.2.4 减少重做日志产生

可以在 DML 语句中用下面保留字来设置要求，以减少重做日志。即：

- UNRECOVERABLE
- NOLOGGING

1. UNRECOVERABLE

这个命令是在建立表时进行设置。如：

SQL>create table employee_history as select * from employee UNRECOVERABLE;

这样做就表示 不产生任何 重做日志。在默认下这样的语句会产生不可恢复的日志。

提示：当在建立表时使用分区时就不能使用 UNRECOVERABLE 和 RECOVERABLE。

2. NOLOGGING

在 Oracle8i 里，**NOLOGGING 和 LOGGING** 可以作为选件来在 DML 里使用。然而，与 UNRECOVERABLE 不同，NOLOGGING 是表的属性的描述。它一旦被说明为 NOLOGGING 就一直有效直到你发出 disable 为止。看下面例子：

```
SQL>create table  employee
( employee_id number,
last_name varchar2(20),
first_name varchar2(20),
hirdate  date,
start_date   date,
end_date  date)
tablespace appl_tab
storage( initial 500k  next 500k pctincrease 0)
NOLOGGING;
```

在建立表结构时设置了 NOLOGGING 属性，可以在 DBA_TABLES 中查到，如：

```
SQL>select owner, table_name  from  dba_tables
Where  logging='NO';
```

§19.5 调整共享池

Oracle 系统中的共享池（SHARED POOL）是用来存放 SQL 语句、PL/SQL 语句、各种包及 Oracle 系统的数据字典数据。调整共享池与重做日志文件及检查点等有关。下面介绍共享池及相关参数的调整。

§19.5.1 理解共享池的用途

Shared pool 是 SGA 的一部分，它是用来存放由应用用户发出的 SQL 和 PL/SQL 语句。Shared Pool 由 LRU(Least Recently Used)算法来进行管理。一旦 Shared Pool 被填满，LRU 算法就对那些最近很少用的语句请出 Shared Pool 区。Oracle 服务器保留那些经常被使用的 SQL 和 PL/SQL 语句在 Shared Pool 区里。我们也可以将那些经常使用的 SQL 语句保留（或缓存--Caching）在 Shared Pool 区里。

§19.5.2 缓存语句的好处

当应用用户发出 SQL 或 PL/SQL 语句到 Oracle 实例时，服务器在运行前先对这些语句进行分析，如发出：

```
SQL>select ename,sal from scott.emp  where  empno=1234 ;
```

首先，Oracle 要做以下的工作：

- 检查语法的正确性；
- 检查所引用的对象在数据字典中是否存在；
- 收集数据字典对象；
- 准备和执行计划；
- 确定引用对象的安全性；
- 产生编译版本（叫 p-code）.

分析运算是相当费时的操作，但是分析也是可以避免的，只要确认被分析的语句已经在 Shared Pool 就可以。即在 Shared Pool 中找到已经缓存成功（cache hit）的语句。如果在机器里没有找到就认为是缓存失败（cache miss）。为了缓存成功，就必须保证两个 SQL 语句完全的匹配才能成功。即 Shared pool 中的语句和当前应用中发出的语句要一样。因此，如果两语句间只是存在大小写的差异也是匹配失败。
然而，在你进行 Shared pool 的调整前，先要了解 Shared Pool 的部件。

§19.5.3 Shared Pool 部件

Shared Pool 由三个部件组成，即库缓存（Library Cache），数据字典（Data Dictionary Cache）和用户全局区（User Global Area）。每个在 SQL 和 PL/SQL 运行中都担当一种角色。

1. 库缓存（Library Cache）

库缓存（library cache）用来存放由应用用户刚发出的 SQL 及 PL/SQL 语句，存储过程、函数、包、触发器、同义词 PL/SQL 包及 JAVA 类库等对象信息。应该有下面几个部分：

- 实际语句的 text 部分；
- 与语句相关的 Hash 值；
- 语句的 P-code ;
- 与语句相关的统计；
- 语句的执行计划。

上面语句的大部分可以从动态数据字典中查询到，如：

```
SQL>
select sql_test, hash_value, optimizer_mode  from V$sqlarea
Where  users_executing != 0 ;
```

这样的查询也可以用于查询当前用户实际发出的语句，每个语句发出多少次及有多少硬盘和缓冲区读相关的语句等。在 V$SQLAREA 视图中只包括 SQL 或 PL/SQL 语句的前 80 字符。SQL 和 PL/SQL 更复杂的文本要在 V$SQLTEXT 字典的 SQL_TEXT 列中找。

提示：由于 V$SQLAREA 和 V$SQLTEXT 存放的是动态信息，所以它们的内容也只能当前某个时刻的情况。

如果你只是看一下 SQL 语句的类型，也可以查看 V$SESSION 数据字典的内容。在 V$SESSION 字典中，COMMAND 列用一组数字（100 以内）来代表命令。请看下表：

Command	Description
2	Insert
3	Select
6	Update
7	Delete
26	Lock table
44	Commit
45	Rollback
47	PL/SQL

例：查询看当前是谁在进行 Update ，可以发出下面命令：

SQL>select username,terminal from v$session where command= 6 ;

2. 数据字典（Data Dictionary Cache）

为了确保表，列等信息的存在，不管是否发出 SQL 语句和 PL/SQL 语句。数据字典都是被检查的。与前面的语句一样，这些表等的信息也被缓存在 Shared Pool 的 **Data Dictionary Cache** 内存里。这里的内存也是按照 LRU 来进行管理。
由于使用 Data Dictionary Cache，所以 Oracle 服务器将数据字典与 SQL 语句分开存放。这样做的理由是：

- 对 Libaray Cache 和 **Data Dictionary Cache** 分别进行 LRU 操作；
- 子查询应用用户发出类似的语句，可能完全不同，但与某个表有关的，也可以得到一些好处（虽然语句可能被执行，但表被保留在里面）。

3.用户全局区（User Global Area）*

如果使用 MTS 选件时，用户全局区才有用。在 MTS 结构下，用户全局区用于存放用户会话的应用。这些信息必须以共享的方式来被访问，在非 MTS 结构下，这些会话信息由程序全局区（Process Global Area）来管理。

§19.5.4 测试 Shared Pool 的性能

测试 Shared Pool 的主要目的是看它的缓存命中率（cache –hit ratio ）。高的命中率说明你的应用用户经常在内存中找到它的 SQL 和 PL/SQL 语句。命中率可以由 Libaray Cache 和 Data

Dictionary Cache 来计算。

1. Libaray Cache

Libaray Cache 的性能可以由 命中率（Hit Ratio）来测量。命中率可以从两个地方来得到。第一种方法是从 V$LIBRARYCACHE 动态数据字典得到；另外是运行 UTLBSTAT.SQL 和 TULESTAT.SQL 来产生 REPORT.TXT 文件来得到。

1) 使用 V$LIBRARYCACHE 数据字典

V$LIBRARYCACHE 字典与命中率有关的列有：

GETHITRATION 是命中率；
RELOADS 重新加载次数；
INVALIDATIONS 无效的次数

```
SQL>l
  1  select  namespace , gethitratio, pinhitratio, reloads, invalidations
  2  From  v$librarycache
  3  Where  namespace  in  ( 'SQL AREA',
  4  'TABLE/PROCEDURE',
  5  'BODY',
  6* 'TRIGGER')
SQL> /

NAMESPACE        GETHITRATIO    PINHITRATIO      RELOADS   INVALIDATIONS
---------------  ---------------- -------------  ------------ ------------
SQL AREA           .993049235     .99359224              4            1
TABLE/PROCEDURE  .789262821     .419168591             0            0
BODY                  .25              0              0            0
TRIGGER                0              0              0            0

SQL>
```

Namespace 的代码关系如下：

Namespace	Code source
SQL AREA	SQL 语句
TABLE/PROCEDURE	PL/SQL 存储过程或函数
BODY	PL/SQL 包体
TRIGGER	PL/SQL 触发器

2) GETHITRATIO

每次语句被分析时，在 V$LIBRARYCACHE 中的 GETS 值就加 1， 而 GETHIT 列存放的是 SQL 语句或 PL/SQL 语句在内存中找到并被使用的次数。当语句被找到后，系统就不需要再分析而直接执行。分析语句的比率 GETS 与不需要分析 GETHITS 的比就得到 命中率 GETHITRATIO 值。如：

```
SQL>select  namespace , gethitratio, pinhitratio, reloads, invalidations ,gets,gethits
  2  From  v$librarycache
  3  Where  namespace  in  ( 'SQL AREA',
  4  'TABLE/PROCEDURE',
  5  'BODY',
  6* 'TRIGGER')
SQL> /

 NAMESPACE      GETHITRATIO   PINHITRATIO     RELOADS     INVALIDATIONS     GETS     GETHITS
------------ ---------------- ------------- ------------ ------------ ---------- ----------
SQL AREA        .993846377        .994289 815            4            1         13813     13728
TABLE/PROCEDURE  .789768185        .42645382            0            0          2502      1976
BODY                  .25                  0           0            0             4         1
TRIGGER                 0                  0           0            0             1         0

SQL>
```

3)PINHITRATIO

与 GET 类似，锁住（pins）是与锁（lockong）有关。然而 GETS 是与分析的次数有关，而 pins 是与执行次数有关。因为一般的 SQL 语句可以是每次语句被分析后一定执行一遍，但也可能执行多遍，每执行一遍， PINS 就加 1。所以 PINHITRATIO 和 GETHITRATIO 有时不相等。

4) RELOADS

RELOADS 列值显示是被执行语句重新分析的次数。我们可以计算出重新加载率：

```
SQL> select  sum(reloads)/sum(pins) "reloads ratio" from  v$librarycache;

reloads ratio
------------
.000107547

SQL>
```

5)INVALIDATIONS

表示无效的次数。

6)使用 REPORT.TXT 输出

REPORT.TXT 是由运行 UTLBSTAT.SQL 和 UTLESTAT.SQL 产生。它包括 GETHITRATIO, PINHITRATIO 及 RELOADS,INVALIDATIONS 信息。

由于库缓存的统计结果放在 STATS$LIB 字典里，字典里，我们可以用下面语句进行查询：

```
SQL>set charwidth 12
SQL>set numwidth 10
SQL>select  namespace library, gets ,
Round(decode(gethits,0,1,gethits)/decode(gets,0,1,gets), 3 )
 gethitratio,  pins,
Round(decode(pinhits,0,1,pinhits)/decode(pins,0,1,pins0,3) pinhitratio,
reloads, invalidations
From stats$lib;
```

注：stats$lib 表是由 …/rdbms/admin 目录下的 utlestat.sql 脚本创建，默认下该表是不存在的。

2.数据字典（Data Dictionary Cache）

与库缓存类似，检查数据字典的有效性也是看它的命中率。数据字典的命中率指的是频繁地被在内存里（不是从硬盘中读）找到。在 V$ROWCACHE 字典里包括了动态的信息。你也可以从 REPORT.TXT 中输出。下面是从数据字典中直接查的例子：

1)使用 V$ROWCACHE 字典：

```
SQL> select  1- (sum(getmisses)/sum(gets)) "data dictionary hit ratio"
  2  From v$rowcache;

data dictionary hit ratio
-------------------------
        .91005291

SQL>
```

2)使用 REPORT.TXT 查询：

当用 UTLBSTAT.SQL 和 UTLESTAT.SQL 执行完后，统计结果被存放在 STATS$DC 字典里，我们只要用下面语句查即可：

```
SQL>set numwidth 8
SQL>select * from stats$dc
where get_reqs != 0 or scan_reqs != 0 or mod_reqs !=0;
```

§19.5.5 用 OEM 测试 Shared Pool 的性能-

(暂略)

§19.5.6 改善 Shared Pool 的性能

改善 Shared Pool 性能可以有以下几种办法：

- 稍微加大 Shared Pool 参数值；
- 为 PL/SQL 开较大的空间；
- 在内存中导入 PL/SQL 代码；
- 鼓励代码重用；
- 建立大的 Pool

§19.5.6.1 较大的 Shared Pool

最简单的办法就是加大 Shared Pool 的大小。因为 Oracle 是动态地从 Shared Pool 中来管理 library 和 Data Dictionary 的大小。加大 Shared Pool 可以改善 Library cache 和 data dictionary Cache 的命中率。

提示：经常看到这样的情况，就是 Library cache 的命中率很好，但 data dictionary cache 很坏或反过来。这样的情况就是因为动态分配所产生的结果。

Shared Pool 的大小在 INITsid.ora 文件中的 **SHARED_POOL_SIZE 参数**指定。这个参数以字节指定，默认下，64 操作系统是 64MB, 32 为操作系统是 32MB。如果你使用的 Oracle 带有 JAVA 兼容（有 JAVA 类库）。一定要确保 SHARED_POOL_SIZE 大于等于 50MB 才行。因为在启动时有超过 4,000 个列库被加载，并且这些类库需要至少这样大的空间。默认时，Oracle8i 配置 JAVA_POOL_SIZE 的三分之一大小（10MB）来缓存 Java 声明和运行。

§19.5.6.2 给大的 PL/SQL 语句一个空间

如果应用调用了一个大的 PL/SQL 包或触发器，则可能由 LRU 算法而将其他的 SQL 语句逐出内存。这样一来，如果有应用需要再读这些语句的话，就影响了 Libaray Cache 的命中率。为避免这样的问题发生。Oracle 给你一种能力在 Library cache 旁设置一个区（portion）。这个区就叫共享池保留区(Shared Pool Reserved Areas)。
共享池保留区需要在 INITsid.ora 文件中的 SHARED_POOL_RESERVED_SIZE 参数来设置。它的大小以字节计。它可以是 SHARED_POOL_SIZE 的 50%。默认下它仅是

SHARED_POOL_SIZE 的5% 。
一旦确定了共享池保留区的大小后，就可以在 V$DB_OBJECT_CACHE 字典来查询相应的结果。它反映 PL/SQL 包的名字和大小，如：

```
SQL> select owner, name , sharable_mem from v$db_object_cache
     Where type in ( 'PACKAGE','PACKAGE BODY')
    Order by sharable_mem desc ;

OWNER     NAME                      SHARABLE_MEM
---------- ------------------------  --------------------
SYS       STANDARD                   232576
SYS       DBMS_JAVA                  48805
SYS       DBMS_STANDARD              20561
SYS       DBMS_OUTPUT                13683
SYS       DBMS_JAVA                  13149
SYS       DBMS_APPLICATION_INFO      12369
SYS       DBMS_OUTPUT                6255
SYS       DBMS_APPLICATION_INFO      3069
```

动态 V$SHARED_POOL_RESERVE 视图可以用于监示 Shared Pool Reserved Area 的使用情况，并帮助确定适当大小。提示你是否重新分配内存。包括：

- REQUEST_MISS 列表示一致的 0 或是静态的；
- FREE_MEMORY 列统计大于 50%的保留区。

非 0 或是稳定的值都表示保留区给得太小。

提示：Oracle 建议你的目的应该是在 V$SHARED_POOL_RESERVED 中的 REQUEST_MISS 和 REQUEST_FAILURES 都接近 0 才行。

§19.5.6.3 在内存中保持 PL/SQL 代码

PL/SQL 包和触发器或 Oracle 序列（Sequences）的大量使用可能改善在内存中永久缓存的频繁使用的 PL/SQL 的共享池(Shared Pool)的命中率直到实例关闭为止。这个进程就是著名的围巾（pinning）。并且它由 Oracle 支持的 DBMS_SHARED_POOL 包来完成。围巾包是由在 INITsid.ora 文件中的 SHARED_POOL_RESERVED_SIZE 设置而被存储。

1. 建立 DBMS_SHARED_POOL

DBMS_SGARED_POOL 包在运行 CATPROC.SQL 建立数据库时并不建立。因而需要运行 DBMSPOOL.SQL 脚本才能建立。该脚本在 UNIX 操作系统下是在 $ORACLE_HOME/rdbms/admin 目录下；在 NT 下是在 %ORACLE_HOME%\rdbms\admin 目录下。用下面命令来建立：

SQL>connect sys/password
SQL>@ORACLE_HOME/rdms/admin/dbmspool.sql

2. 使用 DBMS_SHARED_POOL

DBMS_SHARED_POOL 包包括两个存储过程：KEEP 和 UNKEEP 包。她们分别用于在内存中驻留住（pin）或不驻留(unpin)一个包、触发器或序列。

下面例子表示把 APPROVE_PO 包驻留在内存：

SQL>EXECUTE DBMS_SHARED_POOL.KEEP('APPROVE_PO');

3.识别所被驻留内存的对象

可以查询 V$DB_OBJECT_CACHE 字典的内容得到某个对象是否被驻留在内存中：

SQL>select owner , name , type from v$db_object_cache Where kept='YES';

4. 为什么用驻留

访问频率高的对象可以采用驻留在内存的方法来提高系统处理速度。

§19.5.6.4 鼓励代码重用

要想使 SQL 和 PL/SQL 语句达到命中率高，就要养成代码编写的规范。只有这样才能达到代码重用。请看下面例子：

SQL>select last_name, first_name from app.customer where customer_id='2204';

SQL>select last_name, first_name from app.customer where customer_id='2203';

两个语句分别查询 顾客代码为 2204 和 2203 ，除此以外，语句的其它部分是相同的，如果我们将这个使用频率很高的语句改为：

SQL>select last_name, first_name from app.customer where customer_id= :customer_id;

这样就可以达到代码重用的目的。

§19.5.6.5 建立大的 POOL

增加 SHARED_POOL 大小、设置保留区（RESERVRED_AREA）、在内存中驻留 PL/SQL 包等

都是改善共享池（SHARED_POOL）的有效方法。Oracle 现在给出一个能建立大的 POOL 的参数，Oracle 服务器使用大的池(LARGE_POOL_SIZE)来给 I/O 服务进程 RMAN 和 MTS 分配内存。如果你的系统采用 MTS 或采用 RMAN 进行备份，可在初始化文件 INITsid.ORA 中用一个叫 LARGE_POOL_SIZE 参数来指定。在默认下，该参数的值为 0（即没有 LARGE_POOL_SIZE 参数存在）。除非你说明了 PARALLEL_AUTOMATIC_TUNING 参数为 TRUE。 当手工配置该参数时，允许用最小的非 0 值是 600K 。一旦配置完成，对于 Shared Pool 来说，内存的分配就将不被进程所取消。要是这样设置，你可以查询 V$SGASTAT 字典来看动态性能：

```
SQL>SELECT  name, bytes  from  V$sgastat
Where  pool = 'large pool';
```

§19.6 调整数据缓冲区

我们关心的 SGA 区之一就是数据库缓冲区（Database buffer cache）。下面讨论：

- 理解数据库缓冲区（Database buffer cache）
- 测试数据库缓冲区（Database buffer cache）的性能
- 改善数据库缓冲区（Database buffer cache）的性能

§19.6.1 理解数据库缓冲区

数据库缓冲区（Database buffer cache）是 SGA 区的一部分。在 Oracle7、Oracle8、 Oracle8i 版本里，数据库缓冲区由初始化参数文件 INITsid.ORA 中的 DB_BLOCK_BUFFERS(以块为单位)参数来确定；而 Oracle9i 版本则用 DB_CACHE_SIZE(以字节为单位)参数来设置。它是用来存放有应用用户要访问的数据块。每个块都一样大，数据库缓冲区存放的对象有：

- 表
- 索引
- 簇
- 大对象段(Large Object segment)
- 回滚段
- 临时段

§19.6.1.1 LRU 列表

与共享池（shared pool）一样，数据库缓冲区也是使用 LRU 算法来淘汰那些最近不被使用的数据块。

§19.6.1.2 脏（Dirty）列表

在缓冲区中有三种类型的数据：

状态	说明
Free	缓冲区没有被用过
Pinned	缓冲区正在被使用
Dirty	缓冲区不在使用，但提交的数据还没有被数

§19.6.1.3 用户服务器进程

当用户进程在缓冲区中找不到所需要的数据时，它就从硬盘中读取。而从数据文件中数据的程序就是**用户服务器进程**。它负责将硬盘中的一个数据拷贝读到当前的缓冲区中。

§19.6.1.4 数据库写进程

数据库进程 DBW0 不仅将脏的数据写到硬盘里，而且进行另外的事件操作，下面给出当数据库写进程在进行写时所引起的事件：

事件(event)	DBW0 写入进程如何做
当脏的数据列表达到一定的长度时	DBW0 写脏数据列表
当 LRU 找到没有自由空间时	DBW0 从 LRU 列表中写脏数据缓冲区
当 3 秒过去时	DBW0 从 LRU 中将脏的数据移到脏的列表中，如果门槛的长度超过时，就写数据到硬盘。
在检查点	DBW0 从 LRU 列表中将所有脏数据移到脏列表中并且将它们写到硬盘中 。
在表空间备份时	DBW0 将表空间中的脏数据移到脏的列表中然后将它们写到硬盘。
在表空间临时脱机时	DBW0 从 LRU 列表中将脏数据移到脏列表内并将它们写到硬盘。
在 DROP segment 时	当 drop table 或 drop index 时引起 DBW0 首先写段的脏数据块到硬盘。

§19.6.2 测试数据库缓冲区

在调整数据库缓冲区前，建议先测试当前数据库缓冲区的性能情况。可以有下面的测试方法：

§19.6.2.1 查询 V$SYSSTAT

用户服务器在 SGA 区中没有找到相应的数据就出现缓冲失败（misses）。这样的结果就是要从硬盘中的数据文件读取数据块到 SGA 区来。我们主要关心下面指标：

物理读： 在实例启动时，从硬盘将数据块读到缓冲区（比如读表，索引及回滚段）。

取 DB 块： 数据请求的总块数，这些数据由缓冲区来满足。
一致性获取： 数据请求的总块数，这些数据由回滚段缓冲区来器满足。

```
SET VERI OFF
DEFINE secs=0
DEFINE value=0
COL value FORMAT 99,999,999,990 new_value value
COL secs FORMAT a10 new_value secs noprint
COL delta FORMAT 9,999,990
COL delta_time FORMAT 9,990
COL rate FORMAT 999,990.0
COL name FORMAT a30
SELECT name, value, TO_CHAR(sysdate,'sssss') secs,
(value - &value) delta,
(TO_CHAR(sysdate,'sssss') - &secs) delta_time,
(value - &value)/ (TO_CHAR(sysdate,'sssss') - &secs) rate
FROM v$sysstat
WHERE name = '&&stat_name'
/
```

§19.6.2.2 使用 V$SESS_IO 和 V$SESSION

可以从这两个字典中查看数据库缓冲区的情况，当数据库系统全面运行的时候是非常有用的。要得到完整的结果，还需要进行必要的连接，如：

```
select  username,  osuser,
(io.physical_reads/(io.block_gets+io.CONSISTENT_GETS)) "hit ratio"
from v$sess_io  io , v$session sess
where  io.sid = sess.sid
and (io.block_gets+io.CONSISTENT_GETS) != 0
and username is not null;
```

USERNAME	OSUSER	Hit Ratio
USERS	oracle	.693421
SYS	oracle	.90123
APPS	oracle	.87654

§19.6.2.3 使用 REPORT.TXT*

§19.6.2.4 使用 OEM 工具*

§19.6.3 改善数据库缓冲区性能*

现在我们可以用下面的方法来改善数据库缓冲区的大小以改善系统的性能。

§19.6.3.1 将参数改大些

最简单的办法就是将 database buffer cache 加大。加大主要是在 INITsid.ORA 文件中考虑两个参数的值，即 DB_BLOCK_SIZE 和 **DB_BLOCK_BUFFERs**。

§19.6.3.2 使用多个 Buffer Pool

在默认下，所有的数据库都使用同一个 Buffer Pool 。这样就可能带来一种情形，只有几个少数的用户访问几个表也被频繁地压到内存里来。为了避免这样的情形，oracle 提供了一种能力可以建立多大 3 个分开的缓冲池（叫 buffer pools）。可以由 DBA 把相应的段（segment）指定到适当的缓冲池中。

1.缓冲池的类型

在建立缓冲池前就要确定你需要的是那种类型的缓冲池。可以有：

KEEP POOL
RECYCLE POOL
DEFAULT POOL

在默认下，Oracle 只建立 DEFAULT POOL 缓冲池。

2.确定段需要那种类型的缓冲池

在建立好缓冲池后，接下来就是确定那些 segment 使用那些缓冲池，并且这些 Segment 是如何被访问等。可以从 V$BH 和 DBA_SEGMENTS 中查到缓冲池是如何分配的。V$BH 主要用于并行服务器（Oracle Parallel Server）的情况。但是它对于非并行也很有用。因为它显示当前的数据库中有多少个缓冲池。使用 Object_id 可以查出当前段被缓存的情况：

```
SQL>select obj.owner, obj.object_name, obj.object_type,
Count(distinct bh.block#) “num buffer”
From dbs_objects  obj ,v$bh  bh
Where   obj.object_id=bh.objd
And  owner != ‘SYS’
Group  by  obj.owner,
Obj.object-name,obj.object_type
```

```
Order by  4;

OWNER    OBJECT_NAME   OBJECT_TYPE  num buffer
-----------  ---------------------- --------------------  --------------
APPS        EMPLOYEE          TABLE               12
APPS        REGION             TABLE                8
```

另外的办法，可以从 V$CACHE 和 DBA_USERS 中查到类似的结果，如：

```
SQL.select username “Owner”,
Name “Seq.name”,
KIND “Seq Type”,
Count(distinct block#) “num buffer”
From  v$cache, dba_users
Where  v$cache.owner# = dba_users_user_id
Group by name,username,kind
Having count(distinct block#) >10
Order by 3  desc ;

Owner     Seq name     Seq Type     num buffer
------------ -------------   ------------------ -------------
APPS      EMPLOYEE   TABLE       12
APPS      REGION         TABLE       8
```

3.确定每个缓冲池的大小

可以查 DBA_TABLES,和 DBA_INDEXES 来确定对象的大小。从而的出每个缓冲池的大小。比如 Keep Pool 需要 120MB；Recycle Pool 需要 50MB；Default Pool 需要 80MB 等。

4.建立缓冲池（Buffer Pool）

接下来就是要在 INITsid.ORA 文件中来建立缓冲池。建立缓冲池需要说明下面参数：

参数	说明
DB_BLOCK_BUFFERS	数据库中每个 实例 缓冲池总数
DB_BLOCK_LRU_LATCHES	访问数据库中每个实例的锁存器（latch） 数
BUFFER_POOL_KEEP	DB_BLOCK_BUFFERS 和 DB_BLOCK_LRU_LATCHES 分配给 BUFFER_POOL_KEEP 的数
BUFFER_POOL_RECYCLE	DB_BLOCK_BUFFERS 和 DB_BLOCK_LRU_LATCHES 分配给 BUFFER_POOL_RECYCLE 的数

锁存器（latch）是用于保护所有数据库共享内存访问的区。

下面是建立**缓冲池的 INITsid.ORA 参数描述：**

```
db_name = "s450"
instance_name = s450
service_names = s450

control_files = ("/home/oracle/app/oracle/oradata/s450/control01.ctl",
"/home/oracle/app/oracle/oradata/s450/control02.ctl",
"/home/oracle/app/oracle/oradata/s450/control03.ctl")

db_block_size = 8192

open_cursors = 300
max_enabled_roles = 30
# db_block_buffers = 2048

#建立 64000 块=500MB
db_block_buffers = 64000

# 建立 DB_BLOCK_LRU_LATCHE 为 30
DB_BLOCK_LRU_LATCHES = 30

# 从 64000 块中拿出 19200 块给 KEEP POOL; DB_BLOCK_LRU_LATCHE 30 中 10 给 KEEP
POOL.
BUFFER_POOl_KEEP=( BUFFERS:19200,LRU_LATCHES:10)

# 使用 BUFFER 中剩下 44,800 块中 6400 块（=50MB）给 Recycle Pool;
# LRU 剩下的 20 中的 5 给 RECYCLE
BUFFER_POOL_RECYCLE=(BUFFERS : 6400,LRU_LATCHES:5 )

#shared_pool_size = 52428800
shared_pool_size=104857600

# large_pool_size = 614400
large_pool_size = 1228800
# java_pool_size = 20971520
java_pool_size = 41943040

log_checkpoint_interval = 10000
log_checkpoint_timeout = 1800

processes = 300
```

```
log_buffer = 163840

# audit_trail = false  # if you want auditing
# timed_statistics = false  # if you want timed statistics
# max_dump_file_size = 10000  # limit trace file size to 5M each

# Uncommenting the lines below will cause automatic archiving if archiving has
# been enabled using ALTER DATABASE ARCHIVELOG.
# log_archive_start = true
# log_archive_dest_1 = "location=/home/oracle/app/oracle/admin/s450/arch"
# log_archive_format = arch_%t_%s.arc


#DBCA uses the default database value (30) for max_rollback_segments
#100 rollback segments (or more) may be required in the future
#Uncomment the following entry when additional rollback segments are created and made online
#max_rollback_segments = 51
# If using private rollback segments, place lines of the following
# form in each of your instance-specific init.ora files:
rollback_segments = ( RBS1, RBS2, RBS3, RBS4, RBS5, RBS6, RBS7, RBS8, RBS9, RBS10,
RBS11, RBS12, RBS13, RBS14, RBS15, RBS16 )

# Global Naming -- enforce that a dblink has same name as the db it connects to
# global_names = false

# Uncomment the following line if you wish to enable the Oracle Trace product
# to trace server activity.  This enables scheduling of server collections
# from the Oracle Enterprise Manager Console.
# Also, if the oracle_trace_collection_name parameter is non-null,
# every session will write to the named collection, as well as enabling you
# to schedule future collections from the console.
# oracle_trace_enable = true

# define directories to store trace and alert files
background_dump_dest = /home/oracle/app/oracle/admin/s450/bdump
core_dump_dest = /home/oracle/app/oracle/admin/s450/cdump
#Uncomment this parameter to enable resource management for your database.
#The SYSTEM_PLAN is provided by default with the database.
#Change the plan name if you have created your own resource plan.# resource_manager_plan =
system_plan
user_dump_dest = /home/oracle/app/oracle/admin/s450/udump

remote_login_passwordfile = exclusive
```

```
os_authent_prefix = ""

# The following parameters are needed for the Advanced Replication Option
job_queue_processes = 4
job_queue_interval = 60
distributed_transactions = 10
open_links = 4

mts_dispatchers = "(PROTOCOL=TCP)(PRE=oracle.aurora.server.SGiopServer)"
# Uncomment the following line when your listener is configured for SSL
# (listener.ora and sqlnet.ora)
# mts_dispatchers = "(PROTOCOL=TCPS)(PRE=oracle.aurora.server.SGiopServer)"

compatible = "8.1.0"
sort_area_size = 65536
sort_area_retained_size = 65536
```

5.将段(segment)分配给池

当建立好各类缓冲池后，就可以用 ALTER 语句将段（即表，索引，簇）分配给这些池了。看下面例子：

```
SQL>alter table apps.employee
Storage(BUFFER_POOL KEEP);

SQL> alter table apps.region
Storage(BUFFER_POOL KEEP);

SQL> alter INDEX  apps.employee_id
Storage(BUFFER_POOL KEEP);
```

接下来，可以从 DBA_SEGMENTS 字典中查出那些段被 KEEP 过。如：

```
SQL> SELECT owner, segment_name, segment_type , buffer_pool
From dba_segments
Where  buffer_pool != 'DEFAULT';
```

6.监示 多缓冲区的性能

可以从动态数据字典 V$BUFFER_POOL_STATISTICS 中查到多缓冲区的性能情况。

7. 使用 V$BUFFER_POOL

V$BUFFER_POOL 动态数据字典包括一些关于多缓冲区的信息。可以象下面来查询：
SQL> SELECT NAME, SET_COUNT "latches", BUFFERS
From V$BUFFER_POOL where ID != 0 ;

8. 使用 V$BUFFER_POOL_STATISTICS

V$BUFFER_POOL_STATAISTICS 动态数据字典包括许多我们在 V$SYSSTAT 的关于多缓冲区的信息。可以象下面来查询：

SQL>SELECT name "Buffer Pool",
1 – (physical_reads/(db_block_gets+consistent_gets)) "buffer pool ratio"
from V$BUFFER_POOL_STATISTICS
order by name;

§19.6.3.3 将表缓存在内存中

可以在 Alter table 或 Create table 后加 cache 子句来使表成为**缓存表**(不被请出)。如： alter table ACCOUNT_CODE cache;

取消缓存设置为 nocache ,如：alter ACCTOUNT_CODE nocache;

例：
SQL> CREATE table phone_lst
(employee_Id number,
phone_no varchar2(15),
extension varchar2(5)
) tablespace users
storage(INITIAL 1m next 500k pctincrease 0)
CACHE;

对于已经将表设置成 Cache 的信息，可以从 DBA_TABLES 字典中查到，如：

SQL>select owner, table_name from dba_tables
Where LTRIM(cache) ='Y';

第 20 章　调整物理 I/O

本章介绍物理 I/O 的调整问题。主要包括下面内容：

- 数据库写入器（DBWR0）将数据缓冲区数据写到数据文件；
- 数据库写入器（DBWR0）写数据到回滚段；
- 用户服务器进程读数据块到数据缓冲区；
- 日志写入器（LGWR）从重做日志缓冲区将数据写事务恢复信息到联机重做日志；
- 归档器（ARC0）读重做日志内容并写到归档目的地。

本章内容可参考《OCP:Oracle8iDBA 性能调整及网络管理》chapter7 及
《OCP:Oracle9i Performance Tuning》D33513：Database configuration and I/O Issues.

§20.1 理解 I/O 问题

虽然现在的磁盘容量越来越大，但是在读/写速度上还是跟上CPU的速度。因此，许多软件应用的性能都由于硬盘的输入/输出（I/O）速度跟不上内存的速度而受到限制。通常CPU的活动必须暂停等到I/O完成为止。Oracle 应用可以设计成性能不被I/O所限制的应用。如果数据库文件在它能力内运行的话，调整I/O可以增强系统运行性能。

- 调整 I/O：自顶向下和自底向上
- 分析 I/O 需求
- 规划文件存储
- 选择合适的数据块大小
- 估算设备带宽

§20.1.1 调整 I/O: 自顶向下和自底向上

当设计一个新的系统时，为了满足性能要求，你应该从顶到低分析一下I/O需要，以确定需要什么样的资源。对于一个现成的系统，你应该自底向上动手处理 I/O。比如：

1. 确定系统硬盘数量
2. 确定Oracle所用的硬盘数量
3. 确定系统运行的I/O类型
4. 确定是否I/O 适应于文件系统或原始设备
5. 确定如何手工扩展对象到多个硬盘中
6. 计算你所希望的性能的层次

§20.1.2 分析 I/O 需求

如何确定你的可能的I/O需求：

1.计算应用总需求。找出每个事务读的数量和写的数量

每个事务可以涉及到对象的读写：

- 从对象A读1次
- 从对象B读1次
- 写1次到对象C

这样就得到一个事务需要2次读1次写。

2. 为该应用定义 I/O 性能目标

如系统必须**每秒处理事务**（tps）的个数，设计者可以指定100个tps 的性能级别。达到这样之后，系统就**应该**能够每秒运行300个I/O。

- 从对象A读100次
- 从对象B读100次
- 100 次写到对象C

3. 确定达到这样性能级别的硬盘需要的总数

之后，确定每个硬盘每秒的I/O数。这些数字与下面3个因子有关：

- 硬盘的速度
- 无论I/O是读或是写
- 无论你用文件系统或原始设备方式

一般硬盘速度有下列特性：

1)列出硬盘的I/O速度

硬盘速度	文件系统	原始设备*
每秒读	快	慢
每秒写	慢	快

2)列出硬盘I/O工作单样例

硬盘速度	文件系统	原始设备
每秒读	50	45
每秒写	20	50

4. 确定达到预期性能目标所需的硬盘总数，如：

对象	存储在文件系统中			存储在原始设备中		
	每秒所需R/W次数	每秒磁盘R/W能力	所需磁盘数	每秒所需R/W次数	每秒磁盘R/W能力	所需磁盘数
A	100次读	50次读	2个	100次读	45次读	3个
B	100次读	50次读	2个	100次读	45次读	3个
C	100次写	20次写	5个	100次写	45次写	2个
需要的磁盘数			9个			8个

§20.2 调整数据文件 I/O 性能

数据库的数据文件是由独立的数据库块组成。每个块存放各个段（segment）的数据。这些块由用户服务器进程读到数据缓冲区，再由 DBWR0 写回数据文件。

§20.2.1 测试数据文件 I/O

1．使用 SQL 语句查询动态视图：

可以从动态视图 V$FILESTAT 和 V$DATAFILE来查到数据文件的利用情况。也可以使用OEM工具的到类似的信息。

主要关心下面项目：

- 物理读写数
- 块的读写数
- 读写总的 I/O 次数

```
col name for  a40
col  PHYRDS for   999,999
col  PHYWRTS for 999,999
col  AVGIOTIM for 999,999
col  MINIOTIM for 999,999
col  MAXIOWTM for 999,999
col  MAXIORTM for 999,999
SELECT NAME,PHYRDS,PHYWRTS,AVGIOTIM,MINIOTIM,MAXIOWTM,MAXIORTM
FROM V$DATAFILE, V$FILESTAT
WHERE V$FILESTAT.FILE#= V$DATAFILE.FILE#;
```

```
NAME                                PHYRDS  PHYWRTS AVGIOTIM MINIOTIM MAXIOWTM MAXIORTM
----------------------------------- -------- -------- -------- -------- -------- --------
D:\ORACLE\ORADATA\ORA816\SYSTEM01.DBF     1,112       6       0       0       0       0
D:\ORACLE\ORADATA\ORA816\RBS01.DBF           18      30       0       0       0       0
D:\ORACLE\ORADATA\ORA816\USERS01.DBF          4       2       0       0       0       0
D:\ORACLE\ORADATA\ORA816\TEMP01.DBF           4       2       0       0       0       0
D:\ORACLE\ORADATA\ORA816\TOOLS01.DBF          4       2       0       0       0       0
D:\ORACLE\ORADATA\ORA816\INDX01.DBF           4       2       0       0       0       0
D:\ORACLE\ORADATA\ORA816\DR01.DBF             4       2       0       0       0       0
D:\ORACLE\ORADATA\ORA816\SYSTEM02.DBF         5       2       0       0       0       0
```

已选择8行。

主要分析下面参数：

列名	描述
PHYRDS	在数据文件的物理读次数
PHYWRTS	在数据文件的物理写次数
AVGIOTIM	数据文件的I/O性能平均时间，以1/100秒计。
MINIOTIM	数据文件的I/O性能最小时间，以1/100秒计。
MAXIOWTM	数据文件写的I/O性能最大时间，以1/100秒计。
MAXIORTM	数据文件读的I/O性能最大时间，以1/100秒计。

２．使用 REPORT.TXT：

可以用Oracle提供的UTLBSTAT.SQL和ＵTLESTAT.SQL及REPORT.TXT。

操作步骤如下：

1)在启动Oracle 实例后就启动运行UTLBSTAT.SQL;
2)在下班或适当时候运行UTLESTAT.SQL;
3)查询所统计的结果。

3．使用性能管理器（略）：

§20.2.2 改善数据文件 I/O

所有的数据库段都存放在表空间中，而表空间是由数据文件组成。实际上数据文件存放的是每个段的扩展。所以调整表空间和数据文件是调整工作内容：

1.调整数据文件I/O

简单的方法是平衡数据库中段之间的I/O。比如确定用户不能拥有SYSTEM 表空间的分配权。确定有过多的SYSTEM表空间的I/O。要确认下面问题：

表空间	描述
SYSTEM	必须是由SYS拥有的对象。
TOOLS	SYSTEM的默认表空间。
USERS	所有数据库用户拥有的表空间。
TEMP	用于排序的临时表空间。
RBS	回滚段表空间。
APPL_DATA	模式用的数据表空间。
APPL_INDX	模式用的索引表空间。

Oracle9i版本的默认表空间为：

表空间名	是否必须	说明
SYSTEM	要	必须是由SYS拥有的对象。
UNDOTBS1	要	撤消表空间
CWMLITE	不	OLAP目录元数据知识库(CWMLite)
DEMO	不	演示例子表空间
DRSYS	不	Oracle Text段所用表空间
EXAMPLE	不	例子用表空间
INDX	不	索引表空间
ODM	不	?
OEM_REPOSITORY	不	Oracle 企业管理器知识库所用表空间。
TEMP	要	临时表空间。
TOOLS	不	Oracle工具所用的表空间。
XDB	不	XDB表空间保持Oracle XML DB知识库的数据（SQL到协议，如HTTP和WebDAV等）
USERS	不	各种用户数据表空间

以上如果不符合要求，就需要进行调整。比如将用户的对象从SYSTEM表空间请出来；将数据和索引分开存放。

2.调整数据文件快速I/O

将数据文件调整硬盘位置。平衡每个数据文件的驱动器。再调整INITsid.ORA文件中的DB_FILE_MULTIBLOCK_READ_COUNT参数。
可以修改参数文件 INIsid.ora 中的 DB_FILE_MULTIBLOCK_READ_COUNT 参数来实现一次读多块。但是此参数与操作系统的能力有关，如果要修改，必须知道操作系统的一次读的字节数，比如：在 HP_UX 操作系统中，一次只允许读 64KB 的数据，如果数据库实例的 Oracle 块 DB_BLOCK_SIZE=4KB，则 DB_FILE_MULTIBLOCK_READ_COUNT<= 16。

3.数据文件和重作日志文件的分离

一般检验将所有数据文件和重做日志文件分离，这样可以避免后台进程间的访问冲突。特别是归档模式下尤为重要。

4.数据文件的条带化（striping）

由于在RAID 阵列盘中，不利于在不同驱动器里建立分开的数据文件。可以采用手工方式来实现**数据文件的条带化**。**条带化可以提高读写速度**。下面是手工处理的步骤：

1)建立表EMP，并分配适当的空间：

```
CREATE TABLE EMP
(empno  number
ename varchar2(20),
deptno  number,
sal  number(9,2)
)storage(initial 4m next 4m pctincrease 0)
tablespace  appl_data;
```

2)增加两个数据文件：

```
SQL>ALTER TABLESPACE appl_data
Add datafile '/u02/ordata/prod/appl_data02.dbf' size 5m;

SQL>ALTER TABLESPACE appl_data
Add datafile '/u02/ordata/prod/appl_data03.dbf' size 5m;
```

3)手工将EMP表分配到新的扩展里去：

```
SQL>ALTER TABLE emp
ALLOCATE EXTENT
(DADAFILE '/u02/ordata/prod/appl_data02.dbf'  SIZE 4M);

SQL>ALTER TABLE emp
ALLOCATE EXTENT
(DADAFILE '/u02/ordata/prod/appl_data03.dbf'  SIZE 4M);
```

§20.3 调整数据库写入器性能

与 I/O 有关的数据库写入器（DBWR0），它负责将数据缓冲区数据块(脏的数据块)写到数据文件的中。调整 DBWR0 是物理 I/O 调整内容之一。

§20.3.1 测试 DBWR0 I/O

1.使用 V$SYSTEM_EVENT 动态视图

```
select  event, total_waits,average_wait  from v$system_event
where  event  in (‘buffer busy wait’,’db file parallel write’
                  ‘free buffer waits’, ‘write complete waits’);

EVENT                          TOTAL_WAITS  AVERAGE_WAIT
------------------------------ ------------ ------------
db file parallel write                  239  0.034454825
. . .
```

如果 buffer busy wait、db file parallel write、 free buffer waits、write complete waits 的值很高，就说明 DBWR0 在数据文件写时存在 I/O 问题。

2.使用 V$SYSSTAT 动态视图

在 V$SYSSTAT 动态视图中，列 name 为 DBWR buffers scanned、redo log space requests、DBWR lru scans 的值也与 DBWR 进程的性能有关。比如 redo log space requests 的值不断增长，则表示 DBWR 不有效执行写操作。如：

```
select  name, value  from v$sysstat
where  name = ‘redo log space requests’ ;

NAME                           VALUE
------------------------------ ---------------
redo log space requests          5
```

3.使用 STATSPACK 检测 DBWR 性能

```
Top 5 Wait Events
~~~~~~~~~~~~~~~~~                                Wait      % Total
Event                          Waits     Time (cs)  Wt Time
---------------------------------------- ---------- --------- -------
control file sequential read          29         0     26.25
async disk IO                        139         0     25.27
control file parallel write          214         0     23.68
buffer busy wait                      56         0     20.84
log file parallel  write               9         0      2.01
        -------------------------------------------------------------
```

从显示结果来看，buffer busy wait 有大于 0 的值，因此，DBWR 存在性能问题。

§20.3.2 改善 DBWR0 I/O

可以调整 INITsid.ORA 参数文件中的 DBWR_IO_SLAVES 和 DB_WRITER_PROCESSES。

1.DBWR_IO_SLAVES 参数

可以在 INITsid.ORA 文件中指定 DBWR_IO_SLAVES 参数，以在数据库启动后自动启动数据库写入器伺服器进程。参数的默认值为 0。数据库写入器伺服器与 DBWR0 类似。它们只执行写操作，在数据库缓冲区中，不用从 LRU 列表中移动缓冲区内容到脏列表。
当设置 **DBWR_IO_SLAVES** 为非 0 时，**DBWR_IO_SLAVES** 中 SLAVES、ARC0、LGWR 及 RMAN 就自动被启动。伺服器进程的名字命名如下：

- 第一个 DBWR0 I/O 伺服器：ora_i101_PROD
- 第二个 DBWR0 I/O 伺服器：ora_i102_PROD
- 第 n 个 DBWR0 I/O 伺服器：ora_i10n_PROD

2.DB_WRITER_PROCESSES 参数

当 **DBWR_IO_SLAVES** 伺服器为 DBWR0 完成一些性能问题时，它不能完成所有的工作。因而要在 INITsid.ORA 文件中还要加 **DB_WRITER_PROCESSES** 参数。该参数的默认值是 1，最大为 1 0。如果指定了 **DBWR_IO_SLAVES** 为非 0 值时，则 **DB_WRITER_PROCESSES** 不起作用。当调整 **DB_WRITER_PROCESSES** 参数时也要同时调整 DB_BLOCK_LRU_LATCHES 参数才能实现调整。DB_BLOCK_LRU_LATCHES 参数表示锁存器的数量，它与服务器的 CPU 个数有关。参考 V$Buffer_pool 和 V$buffer_pool_statistics 视图进行设置。

§20.4 调整段的 I/O

Oracle 系统的段（segment），主要指的是表和索引及 CLUSTER 等占用空间的对象。Oracle 系统把段的数据存放在对应的表空间中。而表空间由一个或多个数据文件组成；而数据文件是由多个数据块组成。Oracle 块的大小在建立数据库时就被确定下来。块的大小可能是 2k,4k,8k,16k,32k 或 64k。在数据库中，所有的块都具有一样的大小。

提示：可以用 SHOW PARAMETER DB_BLOCK_SIZE 来看块的大小。

如果你 Oracle 版本是 Oracle8i 或更低的版本的话，数据块的大小在创建实例后就不能再改变。如果版本是 Oracle9i，则可以在原来块（叫主块—primary block size）基础上，再增加不同大小的数据库块（第 2 种块大小）。第 2 种块的大小由 INITsid.ORA 文件的

DB_nK_CACHE_SIZE 参数来指定(详细见《Oracle9i 新功能》)。

§20.4.1 理解 块 I/O

要优化 segment 块的 I/O，需要对块的结构有所了解。下面是块的结构。

块内容结构：

§20.4.2 改善段块 I/O

要改善段（segment）的块 I/O 性能。必须考虑下面的关键因素：

- 动态扩展分配的代价
- 扩展大小性能冲突
- 块大小性能冲突
- 行连接及行迁移对性能的影响
- 全表扫描的最高水位的角色

1．动态扩展分配：

当所有有效的扩展分配用完后，下一个插入将引起下一个分配。对于快速增长的表，动态扩展会带来不合需要的 I/O 超额要求。为了避免动态分配，你必须使该表的分配接近实际的需要。如：

例。下面语句查询表的使用空间情况，是否快用完，接近 1 表示快用完：

```
SQL>select owner, table_name ,
2    1-(empty_blocks/(empty_blocks+blocks)) “% blocks used”
3    from dba_tables
```

```
4    where owner !='SYS'  and
5    1-(empty_blocks/(empty_blocks+blocks)) > .95
6    order  by  1;

OWNER          TABLE_NAME                % Blocks Used
-------------- ------------------------- ---------------------
APPL           SALES                      1
APPL           EMPLOYEE                  .966543
APPL           DEPARTMENT                .968796
```

从查询结果来看，SALES 表使用了接近 1 ，所以赶紧再给 SALES **分配扩展**：

```
SQL>ALTER TABLE SALES  ALLOCATE  EXTENT;
```

2.扩展大小与性能：

如果采用动态扩展，则较大的扩展要比较小的扩展性能好得多。所以建议指定扩展参数也要大些为好。

3.块大小与性能：

OLTP 系统采用较小的块：

- 较小的块提供较好的性能；
- 较小的块减少块冲突，因为每个块包括较少的行；
- 较小的块可以存储较小的行。

然而，较小的块需要分配更多的数据库缓冲区。
DSS 系统需要较大的块：

- 大的块存储更大的数据和整个索引；
- 大的块支持频繁的 I/O。

4.减少行连接及行迁移对性能的影响

由于使用 UPDATE 语句会增加行的大小，甚至该行不能存放在一个块里。为了容纳整个行，Oracle 会查找有足够空间的数据块。如果新块能容纳该行则将整个行移到该块（migrating 迁移）。如果块不存放整个行，则将该行拆成几个片段，在将每个片段存放在不同的数据块中。这样就产生叫链（chaining）的情况。对于一个应用系统来说，不可避免这样的情况发生，但可以对链进行优化，即进行必要的调整。

注：

- *行链接：一行太长（如 LONG，大的 varchar2）一个块放不下，必须几个块来存放。这样不可避免。*
- *行迁移：在一个块内的行由于更新变长而不能在原来块上存放，这样产生行迁移。Oracle 将该行移出原来的块放到另外能放得下的块，但是 Oracle 仍在原来的块上维护*

旧的指针，使指针向新块的位置。

对于行链接和行迁移的存在的检测可以从 V$SYSSTAT 视图中的 ‘table fetch continued row’得到，如：

SELECT name, value from V$SYSSTAT
where name=‘table fetch continued row’ ;

如果 value 有值（行链接的数量），则说明存在行链接。
关于行链接的分析和解决另见其它章节。

§20.5 调整 checkpoint 和 CKPT 的 I/O

由于 LGWR 的调整与检查点的 CKPT 进程有关，数据库的检查点是一个 I/O 加强器。需要调整的参数有：

１．检查数据字典 V$SYSSTAT 中的数据；

２．修改 INITsid.ORA 文件中的检查点参数；

LOG_CHECKPOINT_TIMEOUT
LOG_CHECKPOINT_INTERVAL
FAST_START_IO_TARGET
DB_BLOCK_MAX_DIRTY_TARGET

- LOG_CHECKPOINT_INTERVAL 指定日志文件的操作系统的块数（不是数据库块）。
- LOG_CHECKPOINT_TIMEOUT 设置时间间隔的秒数，一般 900 秒；企业版可以 1800 秒（30 分钟）。如果 DBA 增大重做日志文件，则重做日志交换的频率降低。
- FAST_START_IO_TARGET 参数用于限制 I/O 的数量来换取实例恢复。此参数的默认值等于 DB_BLOCK_BUFFERS 参数所指定的数。该参数的有效值为 0（不使能此参数时）或任何 1000 和 DB_BLOCK_BUFFERS 参数所指定的数之间。
- DB_BLOCK_MAX_DIRTY_TARGET 用于指定脏缓冲区的最大数，这些缓冲区用于收集在 DBWR0 将数据写到硬盘之前的数据。然而，在 I/O 恢复数量上不对硬盘作任何限制，所以此参数要比 FAST_START_IO_TARGET 小。此参数的默认值等于 DB_BLOCK_BUFFERS 参数所指定的数。该参数的有效值为 0（不使能此参数时）或任何 1000 和 DB_BLOCK_BUFFERS 参数所指定的数之间。

§20.6 调整归档及 ARCO 的 I/O

如果你的 Oracle 系统采用归档模式运行的话，则有必要进行 ARC0 的调整。

§20.6.1 测量归档和 ARCO I/O

与归档进程有关的性能问题是：

- 归档目标用归档日志去填充，引起数据库进程暂停直到附加的空间有效为止。
- 由于日志切换使 ARC0 后台进程不再继续，在 ARC0 完成日志文件内容的归档前，引起一个等待发生。

1.填归档日志位置：

如果是归档模式，则需要在 INITsid.ORA 文件中设置 LOG_ARCHIVE_DEST 或 LOG_ARCHIVE_DEST_n，这里的 n 是一个 1 至 5 的数。

注：在 Oracle8i 企业版可以用 LOG_ARCHIVE_DEST_n，而标准版只能用 LOG_ARCHIVE_DEST。

当目标归档被填满时，数据库进程会暂停，并将错误信息写到警告日志文件中。如下图所示：

Tue Apr 18 22:05:09 2001/8/27
Thread 1 advance to log sequence 301
Current log# seq# 301 mem# 0 : /u03/oradata/seed/log1a.dbf
Current log# seq# 301 mem# 1 : /u04/oradata/seed/log1b.dbf
Tue Apr 18 22:10:09 2001/8/27
ARC0:Beginning to archive log# 2 seq# 300
ARC0:error creating arachivelog file '/u05/oradata/seed/seed_300.arc'
Arc0:Archiving not possible:error count exceeded
。 。 。 。 。 。

当出现错误时，数据库将暂停直到归档的空间有效为止。你可以用下面 命令来临时改变归档日志的位置：

SQL>alter system arachive log all to '/u04/oradata/prod';

2.在归档日志切换中 ARCO 经常等待

如果重做日志切换过频，LGWR 就试图写重做日志但 ARCO 进程还没有完成。则 LGWR 将被挂起直到归档完成为止。对于这样的情况，DBA 可以通过修改 INITsid.ORA 中的 LOG_ARCHIVE_MAX_PROCESSES 参数来增加一附加的归档进程。该参数的默认值是，即只有一个后台进程来写日志到归档文件中。该参数的取值可以 2 到 10 之间。
在进行这样的决定前，先要查询 V$ARCHIVE_PROCESSES 动态视图以得到性能信息。该记录有当前系统的归档进程及忙、闲的情况。此外，还要查询 V$SESSION_EVENT 或 V$SYSTEM_EVENT 动态视图。下表给出一点 ARCO I/O 瓶颈的识别方法：

在 **V$SYSSTAT** 中**存在等待**表示 **ARCO I/O 存在瓶颈**问题：

统计值	说明
日志文件切换（归档需要）	LGWR 等待，因为重做日志切换是归档还没完成
日志文件切换完成	重做日志切换完成需要多长时间，高或稳定增长表示带有 ARCO 问题。
日志缓冲区	当在日志缓冲区分配时是如何等待。LGWR 等待由 ARCO 产生。
日志并行写	LGWR I/O 需要多长时间。高或稳定增长表示 LGWR 和 ARCO 有冲突。

****
数据字典 V$LATCH 存放有恢复日志使用的信息.
Select name,(immediat_mosses/decode(immediate-
gets+immediate-misses),0-1)-
(immediate-gets+immediate-misses)))*100
immediate-contention,
(misses/decode((gets+misses),0-1))*100
wait-contention
from V$LATCH where name in('redo copy','redo allocation');

这里 name 为各种处理的文字说明，如 session allocation,
如果 immediate_contention 或 wait_contention 的值大于 1
则说明发生了竞争，如果发生了竞争则可以在 init.ora 文件中
log_small_entrry_max_size 设较小些。

注意: 有关 ORACLE 中参数的修改,可以从 V$parameter 数据字典
中得到, 当 ISDEFAULT='FALSE' 时可以在 init.ora 文件中设置
（修改）; 当 ISDEFAULT='TRUE' 时不能在 init.ora 文件 中设置
****

§20.6.2 改善归档和 ARCO I/O

1．加更多的日志组：

每个数据库必须有至少两个重做日志组。然而，还需要增加另外的日志组到你的数据库中。

2.使重做日志大些：

增加重做日志文件的大小可以减少日志的频繁切换和归档覆盖。为了检查点和恢复的目的，Oracle 建议日志切换的发生在每 20 至 30 分钟一次为好。我们可以使用下面参数来调整系统的切换频率： LOG_CHECKPOINT_TIMEOUT ， LOG_CHECKPOINT_INTERVAL ，FAST_IO_TARGET 及 DB_BLOCK_MAX_DIRTY_TARGET 参数。

§20.7 调整排序的 I/O

应用中经常用到排序，如果排序是在内存进行，则代价不高，如果是在硬盘上进行，则代价就高。因为要进行 I/O 操作。

§20.7.1 理解排序活动

下面语句可以引起排序：

- ORDER BY
- GROUP BY
- SELECT DISTINCT
- UNION
- INTERSECT
- MINUS
- ANALYZE
- CREATE INDEX
- 连接两个没有索引的表。

可以在 INITsid.ORA 文件中设置 SORT_AREA_SIZE 和 SORT_AREA_RETAINED_SIZE 参数来实现调整。

SORT_AREA_SIZE 参数

该参数的默认值都很小，可能有 6 个 Oracle 数据库块。一个会话的有效排序区受到 SORT_AREA_SIZE 和 SORT_AREA_RESERVED_SIZE 控制。如果内存区太小，则 Oracle 就用磁盘的临时表空间进行排序。为了确定一个实例在处理排序时所内存和临时表空间的比率，可以用下面语句进行查询：

```
select  a.value “disk sort” , b.value “memory sort” ,
round(a.value/(b.value+a.value)*100,2) “disk sort percentage”
from  V$SYSSTATa  , V$SYSSTATb
where  a.name = ‘sort ( disk ) ‘  and  b.name = ‘sort (memory)’ ;

  disk sort      memory sort       disk sort percentage
------------ -------------------- ------------------------
           1           1357864                       0
```

从上面结果来看，排序都是在内存里进行的，磁盘排序几乎没有。一般来说，Oracle 也建议大部分排序在内存里进行。对于大多数的系统来说，磁盘的排序的百分比不应该大于 5 为宜。

SORT_AREA_RETAINED_SIZE 参数

默认下，SORT_AREA_RETAINED_SIZE 等于 SORT_AREA_SIZE。最小为 2 个 Oracle 块；最

大受到 SORT_AREA_SIZE 限制。
当排序完成后，如果排序区还有需要返回给用户的以排好的数据行。该用户的服务器进程（ Server Process）就把内存保留降到 SORT_AREA_RETAINED_SIZE 参数所指的范围。

PGA_AGGREGATE_TARGET 参数（9i）

在使 SORT_AREA_SIZE 指定为每个会话分配多少排序内存后，可用 PGA_AGGREGATE_TARGET 参数设置用户进程在完成数据库活动（包括排序操作等）后，可继续占用内存的一个上限。PGA_AGGREGATE_TARGET 参数默认值为 0；有效值为 10MB 到 4000GB 。

WORKAREA_SIZE_POLICY 参数

WORKAREA_SIZE_POLICY 参数用来分配给用户进程和内存量是明确或是隐含的，当 WORKAREA_SIZE_POLICY 参数=MANUAL 时，每个用户的排序区大小等于 SORT_AREA_SIZE 参数的大小。当 WORKAREA_SIZE_POLICY 参数=AUTO 时，Oracle 要自动管理内存分配。使得每个用户所用排序超过 PGA_AGGREGATE_TARGET 参数的值。

§20.7.2 改善排序 I/O

排序是数据库系统消耗大量时间的一种算法。改善排序的主要工作是：

1．避免排序：

一个可以改善排序性能的方法是减少排序数目。方法如下：

- 在 UNION 中使用 UNION ALL
- 确认使用索引
- 使用 ORDER BY 或 GROUP BY 来创建索引
- 在合适的 NOSORT 创建索引
- 分析表和索引时用 COMPUTE

2．加大排序区：

可以根据内存情况适当加大 SORT_AREA_SIZE 参数的大小。一般视要处理的数量的大小来确定，一般可以先设置为几兆字节，如设置为 10MB，则：
SORT_AREA_SIZE=10240000

需要注意：如果该参数设置比较大时，如果所有用户都在进行排序操作，则系统不断分配内存给各用户，这样可能出现内存超支而出现负面作用。
建议：一般对于中小型的 OLTP 应用，SORT_AREA_SIZE 参数的默认值可能已够用。所以建议将该值设置为 1M 左右为最佳。

3．使用多个临时表空间：

如果排序多数是在磁盘进行，则建立多个临时表空间来使多个用户对临时表空间的争用。通过多个临时表空间来使排序分布在不同的磁盘上。
在 Oracle9i 版本里，可用 ALTER DATABASE 命令将一个临时表空间指定为数据的临时表空间。如：
例 1.建立临时表空间 dflt_temp_tbsp (第 1 步)：
CREATE TEMPORARY TABLESPACE dflt_temp_tbsp
TEMPORARY 'D:\oracle\oradata\hr_temp\dflt_temp_tbsp01.dbf' SIZE 40m ;

例 2.指定临时表空间 dflt_temp_tbsp 为默认表空间(第 2 步)：
ALTER DATABASE DEAULT TEMPORARY TABLESPACE dflt_temp_tbsp ;

4．查询 V$SORT_SEGMENT 监视排序：

V$SORT_SEGMENT 视图包含临时表空间中常驻排序段的大小和增长信息。如：

SQL>select tablespace_name, current_users,max_sort_blocks
FROM V$SORT_SEGMENT;

TABLESPACE_NAME	CURRENT_USERS	MAX_SORT_BLOCKS
TEMP	11	300

有 11 个用户使用 temp 临时表空间进行排序。 单个用户的最大排序量为 300 个 Oracle 块。MAX_SORT_BLOCKS 列的值对于确定 SORT_AREA_SIZE 有参考作用。

5．查询 V$SORT_USAGE 监视排序：

V$SORT_USAGE 可反映当前正在排序的用户的信息。如：

Select user,tablespace,blocks from **V$SORT_USAGE order by blocks;**

USER	TABLESPACE	BLOCKS
APPS	TEMP	200
JOE	TEMP	110

Blocks 显示了以块为单位的排序的大小。可根据此列的值来设置 SORT_AREA_SIZE 和 SORT_AREA_RETAINED_SIZE 参数值。

第 21 章　Oracle 系统运行中的资源竞争

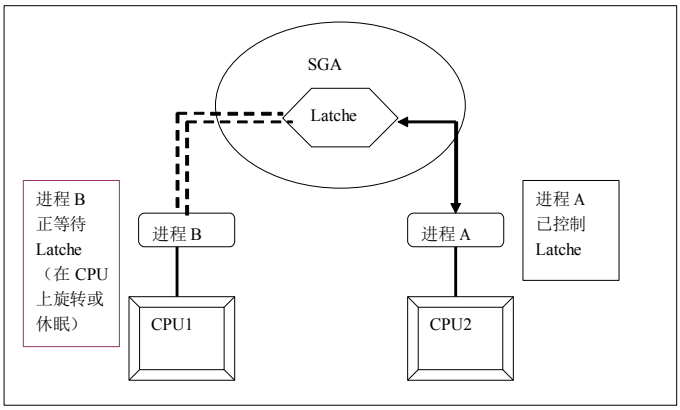
当多个进程试图同时访问相同的资源时，就会出现资源竞争问题。资源资源竞争也能影响到应用系统的性能。下面是资源竞争的调整介绍：

§21.1 理解锁存器竞争

锁存器(latch)是一个很重要的概念，理解它的原理，对于解决资源的竞争有帮助。下面介绍有关锁存器的内容。

§21.1.1 锁存器概念

锁存器（Latche）是 SGA 区中用于保护共享数据结构(Shared Data Structure)和内存的一种机制。比如，保护当前正在访问用户的数据列表和缓冲区中块的数据结构。一个用户进程和服务器进程必须要获得锁存器才能开始管理和查找缓冲区中的数据结构，并在完成处理后释放该锁存器。锁存器的执行是由平台上的操作系统来完成的。所以，长时间的处理会产生等待锁存器的情况。



使用锁存器的目的保证：

- 连续性访问：

1) 保护 SGA 区的数据结构；
2) 保护共享内存分配。

- 连续性执行：

1）预防临界的代码段执行；
2）防止冲突。

单 CPU:
如果锁存器被另外的用户所持有，而进程要请求锁存器的话，则该进程进入短时间的休眠。
多 CPU:
如果锁存器被另外的用户所持有并正在运行，则在下一步的几个指令**就可能**释放该锁存器。所以，在多 CPU 环境中，请求进程持锁存器，但要进行 轮换（计数到一个指定的数）。因此，可试图再次请求锁存器，如果还没有效，则再进入轮换。轮换的数和次数是由操作系统和硬件平台确定的。如果在轮换数到达后，锁存器还没有效，该进程就释放 CPU 并进入休眠 (与单 CPU 类似)。然而，由于锁存器是在一个很短的时间内被持有（以微秒），在多 CPU 下，一个成功的轮换可避免 CPU 代价的切换。

§21.1.2 锁存器的类型

锁存器以**愿意等待（Willing-to-wait）**或**立即（Immediate）**一种或两种模式来请求。

愿意等待（Willing-to-wait）:

- 请求进程等待一段短的时间再请求锁存器；
- 进程继续等待，并且直到锁存器有效为止。

立即（Immediate）

如果带立即请求的锁存器没有效，则请求进程不等待，而是继续处理其它的指令。例如，当 PMON 进程试图清除放弃的进程时，如果锁存器无效，则 PMON 继续处理后来的指令。

§21.2 检测锁存器竞争问题

资源竞争的征兆可以从 V$SYSTEM_EVENT 视图中找到。该视图揭示各种系统的可能竞争问题。问题可能是锁竞争、缓存竞争及 I/O 竞争等。它对于记忆问题的征兆（不是实际发生）是很重要的。例如，查看 V$SYSTEM_EVENT 可以注意到有大量的缓冲区在忙等待，它可能是由于有多个进程正在插入数据到同一个块。这样每个进程必须互相等待其它的进程完成后才能进行自己的插入。解决这样的问题可以是引入对象自由列表。
缓冲区忙等待也能引起某些锁自由等待。因为这样的等待多数是高速缓冲区不足引起。这样的情况可以增加SPINCOUNT以减少锁存的等待。你应该使这个对象能允许多个进程插入到自由块来减少竞争。

视图V$RESOURCE_LIMIT提供了关于系统资源的当前和最大全局资源使用的一些信息。

这些信息可以帮助你更好地作决定。
如果系统有闲置的时间，就检查V$SYSTEM_EVENT来分析系统的情况。检查事件的最高平均等待时间。然后采取适当的措施。比如，发现有很高的锁存自由等待。那么就要再看V$LATCH以查出问题。但是，光从V$LATCH视图也不能反映竞争的问题所在。

对于过分的缓冲区忙等待，要查看高速缓冲等待 V$SESSION_WAIT，使你能了解到那种类型块等待的次数最多和等待的时间最长。用户可以对文件进行解码得到对应的块号。

§21.2.1 从 V$LATCH 中查询锁存器的竞争

查询V$LATCHE视图，可得到有关锁存器竞争的信息，在Oracle9i系统里，V$LATCH的结构为：

```
SQL> desc v$latch
 名称                                      是否为空? 类型
 ----------------------------------------- -------- ----------------
 ADDR                                      RAW(4)
 LATCH#                                    NUMBER
 LEVEL#                                    NUMBER
 NAME                                      VARCHAR2(64)
 GETS                                      NUMBER
 MISSES                                    NUMBER
 SLEEPS                                    NUMBER
 IMMEDIATE_GETS                            NUMBER
 IMMEDIATE_MISSES                          NUMBER
 WAITERS_WOKEN                             NUMBER
 WAITS_HOLDING_LATCH                       NUMBER
 SPIN_GETS                                 NUMBER
 SLEEP1                                    NUMBER
 SLEEP2                                    NUMBER
 SLEEP3                                    NUMBER
 SLEEP4                                    NUMBER
 SLEEP5                                    NUMBER
 SLEEP6                                    NUMBER
 SLEEP7                                    NUMBER
 SLEEP8                                    NUMBER
 SLEEP9                                    NUMBER
 SLEEP10                                   NUMBER
 SLEEP11                                   NUMBER
 WAIT_TIME                                 NUMBER
```

可查询等待类型，如下面的列：

从 V$LATCH 视图中列值可以得到是否存在竞争：

列名	说明
NAME	锁存器的名字
GETS	请求类型为" WILLING-TO_WAIT",并最终获得锁存器，则该值表示请求的数目
MISSES	请求类型为" WILLING-TO_WAIT",并最终未得锁存器，则该值表示失败的数目
SLEEP	请求类型为" WILLING-TO_WAIT",该值表示等待的次数
Wait_time	等待愿意请求的毫秒（millisecond）数
Cwait_time	等待的积累数（包括轮换和休眠）
IMMEDIATE_GETS	不需要等待或得的次数(反映立即请求)
IMMEDIATE_MISSES	不需要等待失败的次数(反映立即请求)
WAITERS_WOKEN	等待被唤醒的次数
SLEEP1	睡眠 1 次的等待
SLEEP2	睡眠 2 次的等待

§21.2.2 用 STATSPACK 报告锁存器的竞争

可用STATSPACK工具来报告竞争的信息，如：

Top 5 Wait Events

Event	Waits	Wait Time (cs)	% Total Wt Time
latch free	217,224	65,056	63.55
db file sequential read	39,836	31,844	31.11
db file scattered read	3,679	2,846	2.78
SQL*Net message from dblink	1,186	870	.85
log file sync	830	775	.76

在本例子中，latch free 值很高。

§21.3 减少锁存器竞争

对于一般的 DBA，在 Oracle9i 系统里，不要试图去调整锁存器，但是，可进行下面的工作。

- 研究未来整锁存器相关的竞争问题；
- 对共享池和库缓存来说，如果竞争成为主要问题的话，可调整应用；
- 进一步研究后，可建议改变共享池和库缓存的大小。

§21.3.1 减少锁存器竞争的主要点

一般，对于 DBA 来说，主要是：

- 共享池(Shared pool)：保护内存分配；
- 库缓存（Library cache）：用于分配匹配的 SQL 语句；
- 数据高速缓存的 LRU(Cache buffer LRU chain)链：保护数据缓存的 LRU 列表；
- 高速缓存链(Cache buffer chain)：在 SGA 中搜寻数据块时需要；
- 重做日志分配（Redo allocation）：在日志缓冲区中管理重做日志空间的分配；
- 重做日志拷贝（Redo copy）：用于写重做日志记录到重做日志文件。

1.共享池锁存器(SHARED POOL LATCH)和库高速缓存锁存器(LIRARY CACHE LATCH)

这两个锁存器的冲突主要是 SQL 和 PL/SQL 语句不能被重用。可能原由是语句没有绑定变量；或光标高速缓冲区不够。解决问题是：调整共享池或调整应用。

Oracle 服务器如果每一个大的共享池(Large Pool)，可能导致锁存器竞争。这种情况就要建立一个大的共享池或者回复直接连结。

2. 库高速缓存锁存器(Library Cache LATCH)

与 SHARED POOL LATCH 类似，Library Cache 锁存器频繁地等待也预示 SHARED POOL 性能不好。见“SGA 调整”章节。

3. 高速缓存 LRU 链锁存器(Cache Buffer LRU Chain LATCH)

当脏的块(Dirty block)被写到磁盘或一个服务器进程正搜寻要写的块时，这种冲突提示需要过多的缓冲区要求，比如：需要更多的高速缓冲区排序，低效率的 SQL 语句（如搜寻索引范围很大）；或全表扫描等。

要解决这些问题，考虑调整数据缓中区或数据库写入器操作。

4. 高速缓存区链锁存器(Cache Buffer Chain LATCH)

用户进程试图在数据缓冲区中定位一个数据块时需要这种锁存器。这种冲突表示有些指定的块（hot block-热块）被重复访问。
在 Oracle8i 中，可在 INIT.ORA 中设置 DB_BLOCK_LRU_LATCHES 参数来避免，但是在 Oracle9i 里，不再使用此参数。在 Oracle8i 版本，一个缓冲区池被等价地分给工作的 LRU 锁存器集，所以每个缓冲区都被每一个 LRU 锁存器所保护着。正常情况下，你指定更多的锁存器，则锁存器很少存在竞争；然而，在小的 LRU 列表中有过多的锁存器，也潜在地减少数据库块的**缓存生命周期**。

最大的锁存器数(CPU_COUNT*2*3)不能超过 CPU 个数的两倍和共享池的数。一般，你可设置此参数达 CPU 的数或几倍于 CPU 的数。
如果有多个 DBWn 在运行，LRU 锁存器的数就要大于或等于 DBWn 进程数。为了平衡工作，LRU 锁存器应该是 DBWn 进程的几倍。

如果不设置此参数，Oracle 服务器采用 CPU_COUNT/2 来替代该参数。

5. 重做日志分配锁存器(Redo Allocation LATCH)

Redo Allocation 锁存器 用来管理 Redo Log Buffer 中的空间分配。如果用户试图同时将重做日志信息放入 Redo Log Buffer 中，就会发生对该锁存器的争用。

6. 重做日志拷贝锁存器(Redo Copy LATCH)

Redo Copy 锁存器由服务器进程在复制它们的信息到 Redo Log Buffer 时访问。

§21.3.2 共享池与库高速缓存锁存器

共享池和库缓存冲突的主要原因是不必要的 SQL 语句分析。解决这个问题的方法在“调整共享池”中给出。下面介绍比必要的分析的内容：

- 非共享 SQL:

类似的语句可用绑定变量来实现共享，你可预见一个类似的语句不止一次地被执行。可 从 V$SQLAREA 视图中查询出类似的语句：

```
SELECT sql_text
FROM V$SQLAREA
WHERE EXECUTIONS = 1
ORDER BY UPPER(sql_text);
```

- 找出重分析的共享 SQL:

```
SELECT SQL_TEXT,PARSE_CALLS,EXECUTIONS
FROM V$SQLAREA
ORDER BY PARSE_CALLS;
```

§21.3.3 减少重做日志缓冲区锁存器竞争

Oracle 使用重做日志将内存的日志信息写到日志文件中。但是即使重做日志缓冲区出现了竞争，也很少对数据库的性能产生严重影响。由于 Oracle 有很多锁存器，不过可以用下面语句查询出：

```
1* select level#,name,gets,misses from v$latch order by name
SQL> set linesize 100
```

```
SQL> /

  LEVEL# NAME                                     GETS     MISSES
--------- ---------------------------------------- ---------- ----------
        8 AQ Propagation Scheduling Proc Table          0          0
        8 AQ Propagation Scheduling System Load         0          0
        5 Active checkpoint queue latch             23149          0
        7 Checkpoint queue latch                    140908          3
        6 Direct I/O Adaptor                             2          0
        0 GDS latch                                      0          0
        0 JOX SGA heap latch                          2981          0
        8 KCL freelist latch                             0          0
        5 KCL instance latch                             0          0
        4 KCL lock element parent latch                  0          0
        3 KCL name table latch                           0          0
        6 KSFQ                                           0          0
        1 NLS data objects                               4          1
        6 Token Manager                                999          0
        6 X$KSFQP                                        0          0
        8 address list                                   1          0
        4 archive control                                0          0
        4 archive process latch                          0          0
        8 batching SCNs                                  0          0
        1 begin backup scn array                         0          0
        3 cache buffer handles                          26          0
        1 cache buffers chains                     2422102        486
        3 cache buffers lru chain                    42103          0
        4 cache protection latch                         0          0
        0 cached attr list                               0          0
       13 cas latch                                      0          0
        5 channel handle pool latch                     32          0
        5 channel operations parent latch              130          0
        3 comparison bit cache                           0          0
        8 constraint object allocation                   0          0
        8 cost function                                  0          0
        5 device information                             0          0
        4 dictionary lookup                              0          0
        1 direct msg latch                               0          0
        8 dispatcher configuration                       0          0
        5 dlm cr info freelist latch                     0          0
        5 dlm cr request queue latch                     0          0
        3 dlm deadlock list                              0          0
        3 dlm domain lock latch                          0          0
        3 dlm domain lock table latch                    0          0
```

1	dlm domain table latch	0	0
5	dlm group lock latch	0	0
5	dlm group lock table latch	0	0
0	dlm group table freelist	0	0
5	dlm lock table freelist	0	0
3	dlm process hash list	0	0
4	dlm process table freelist	0	0
4	dlm recovery domain record latch	0	0
1	dlm recovery domain table latch	0	0
1	dlm resource hash list	0	0
3	dlm resource scan list	0	0
3	dlm resource table freelist	0	0
5	dlm shared communication latch	0	0
1	dlm statistic table latch	0	0
3	dlm synchronous data latch	0	0
3	dlm timeout list	0	0
3	dml lock allocation	121730	1
3	done queue latch	0	0
4	dropped object history latch	0	0
8	end-point list	0	0
4	enqueue hash chains	351005	69
5	enqueues	576152	24
5	error message lists	0	0
0	event group latch	21	0
7	file number translation table	11	0
0	first spare latch	0	0
0	fixed table rows for x$hs_session	0	0
1	global transaction	0	0
2	global tx free list	0	0
1	global tx hash mapping	0	0
1	hash table modification latch	0	0
2	i/o slave adaptor	0	0
3	instance information	0	0
5	intra txn parallel recovery	0	0
0	job_queue_processes parameter latch	858	0
0	ksfv messages	0	0
2	ksfv subheap	0	0
5	ktm global data	180	0
8	kwqit: protect wakeup time	0	0
7	large memory latch	0	0
9	latch wait list	521	0
5	library cache	2742065	468
5	library cache load lock	1134	0
3	list of block allocation	79518	2

3	loader state object freelist	94	0
8	longop free list	0	0
6	message pool operations parent latch	3	0
8	messages	281069	17
8	mostly latch-free SCN	0	0
2	msg queue latch	0	0
3	multiblock read objects	1358	0
1	multiple dbwriter suspend	0	0
0	name-service entry	0	0
3	name-service memory objects	0	0
2	name-service namespace bucket	0	0
1	name-service pending queue	0	0
0	name-service request	0	0
0	name-service request queue	0	0
8	ncodef allocation latch	858	0
6	parallel query alloc buffer	0	0
8	parallel query stats	0	0
5	parallel txn reco latch	0	0
8	parameter list	0	0
3	policy information	0	0
8	presentation list	0	0
0	process allocation	21	0
0	process group creation	29	0
2	process parent latch	0	0
5	process queue	0	0
4	process queue reference	0	0
6	query server freelists	0	0
2	query server process	0	0
6	**redo allocation**	**210295**	**11**
5	**redo copy**	**54**	**0**
5	redo writing	210733	20
0	resmgr group change latch	0	0
6	resmgr:actses active list	0	0
8	resmgr:actses change group	0	0
11	resmgr:actses change state	0	0
8	resmgr:gang list	0	0
6	resmgr:method mem alloc latch	0	0
7	resmgr:plan CPU method	0	0
8	resmgr:queued list	0	0
12	resmgr:resource group CPU method	0	0
8	resmgr:runnable lists	0	0
8	resmgr:running actses count	0	0
5	resmgr:schema config	0	0
2	resmgr:session queuing	0	0

```
4 row cache objects                          227717         3
0 second spare latch                              0         0
8 sequence cache                                 45         0
5 session allocation                         405604        66
1 session idle bit                           259052         2
1 session queue latch                             0         0
0 session switching                             867         0
8 session timer                               17787         0
8 shared java pool                              673         0
7 shared pool                                311338        46
3 sort extent pool                              212         0
3 temp lob duration state obj allocation          0         0
1 temp table ageout allocation latch              0         0
8 temporary table state object allocation         0         0
4 trace latch                                     0         0
8 transaction allocation                     122303         8
8 transaction branch allocation                 858         0
5 undo global data                           147017        10
3 user lock                                      26         0
2 vecio buf des                                   0         0
0 virtual circuit buffers                         0         0
8 virtual circuit queues                       2671         0
8 virtual circuits                                0         0

151 rows selected.
```

我们只介绍用于控制重做日志缓冲区活动的两个锁存器：**重做分配**（redo allocation）锁存器和**重复复制**（redo copy）锁存器。

§21.3.3.1 检查重做日志竞争

重做分配锁存器：

通过重做分配锁存器，用户可以控制在重做分配锁存器中的主要条目进行空间分配。为了获得在缓冲区内分配一定的空间，Oracle 用户进程必须获得重做分配锁存器，因为整个系统只有一个重做分配锁存器，所以每次只有一个用户进程能在缓冲区分配空间。

重做复制锁存器：

用户进程首先获得允许进行**复制锁存器**，然后有获得**分配锁存器**。在执行分配工作后在释放分配锁存器。在上述操作进行之后，进程就在复制锁存器上进行复制工作。而该用户也就保持有分配锁存器一段较短的时间，而用户进程保持有锁存器期间，用户进程不得复制锁存器。

§21.3.3.2 分析重做日志竞争

Oracle 的锁存器(Latch)是用来保护内存结构的一种技术。每个锁保护的是不同的结构或机制。在同一时刻只有一个进程能访问一个锁存器。如果当一个进程需要的锁是在忙时，就产生等待。Oracle 有两种类型的锁存器：

- 如果锁存器是 ***Willing-to-wait*** 愿意等待锁存器，则请求进程将等待一段时间，并再次对锁存器提出请求。
- 如果锁存器是 ***IMMEDIATE*** 即刻锁存器，则请求进程将不会等待。

我们可以从 V$LATCH 动态视图中查询出数据，这些数据反映出各种锁存器的请求活动。在不同类型的锁存器的请求之间存在明显的差别，主要有：

WILLING-TO_WAIT　如果某个“愿意等待”的请求所请求的锁存器不可用，那么请求进程将等待一段时间，并再次对给锁存器提出请求，总之，该进程将不断进行等待与请求，直到能够获得相应的锁存器为止。

IMMEDIATE　　如果某个“即刻”请求所请求的锁存器不可用，那么该请求进程不会等待，而继续处理过程。

对视图中信息进行分析：

- 请求获得成功，gets 值加 1；
- 因为请求导致等待（失败），则 misses 加 1；
- 由于对锁存器进行过两次请求，一次在初始请求之后，另一次在第 2 次请求之后，所以 sleeps 的值加 2。

下面查询用户在一段时间内对重复锁存器和复制锁存器的活动情况：

```
SELECT ln.name, gets, misses, immediate_gets, immediate_misses
FROM v$latch l, v$latchname ln
WHERE ln.name IN ('redo allocation', 'redo copy')
AND ln.latch# = l.latch#;
```

```
NAME            GETS               MISSES    IMMEDIATE_GETS  IMMEDIATE_MISSES
--------------- ------------------ ---------- ---------------- ----------------
redo allocation      252867              83               0                0
redo copy                 0               0           22830                0
```

- 如果 MISSES 与 GETS 之比超过 1%
- 如果 IMMEDIATE_MISSES 与(IMMEDIATE_GETS+IMMEDIATE_MISSES)之比超过 1%。

如果上面任何一种成立都表明**有锁存器争用**。减少竞争的办法如下：

§21.3.3.3 减少锁存器的竞争

从上面可以看到，由于两个Oracle进程都同时试图获得锁存器的时，就会出现锁存器的竞争。在单个CPU时，由于只有单个进程在活动，所以很少出现竞争。

降低对重做分配锁存器的争用：

缩短单个进程保持有锁存器的时间，**减少** LOG_SMALL_ENTRY_MAX_SIZE的值来实现。

降低对重做复制锁存器的争用：

多CPU，多个重做日志锁存器 可以允许多个进程并发地将条目复制到重做日志缓冲区内。参数LOG_SIMULTANEOUS_COPIES的默认为实例可使用的CPU数。
如果**存在竞争**，则**增加LOG_SIMULTANEOUS_COPIES的值**。最好是CPU数的两倍。

§21.3.4 减少对 LRU 锁存器的竞争

LRU 是用来控制缓冲区的数据淘汰的操作。一般在 SMP 环境下，LRU 锁存器的数目可以设置为 CPU 的一半。对于非 SMP 来说，只有 1 个就够了。

- 对于具有多个 CPU 的 SMP 环境，出现 LRU 锁存器的竞争会影响系统的性能;
- 可以查询 V$LATCH、V$SESSION 和 V$SYSTEM_EVENT 来检测 LRU 的竞争情况；
- 通过初始化文件的 DB_LOCK_LRU_LATCHES 参数来设置 LRU 锁存器的数目。

在设置 LRU 锁存器的数目时，主要考虑下面原则：

- LRU 锁存器的最大数目（DB_LOCK_LRU_LATCHES 参数来设置）为 CPU 的两倍；
- 用户运行在单进程环境下，不要设置 LRU 锁存器数目；
- 如果实例工作负荷较大，可以考虑设置 LRU 锁存器的较大数目。

§21.4 减少自由列表的竞争

考虑应用系统的性能，在声明段(table, index,cluster)结构时，一般建议声明storage中的pctfree、pctused等。以在数据插入和修改及删除时，系统按照所设置参数来优化块的使用。当段的块还未被装满到 pctfree值所指定时，free list 表示块中的自由空间的值。当用户频繁向块中插入数据时，服务器进程就要访问该表的free list 时，这时可能出现等待。这些等待实际就是free list 的争用。调整free list 目标是：保证进程不用等待就能访问段的free list ，从而最大限度地减少free list的争用。

§21.4.1 检测 free list 是否存在竞争

自由列表就是在数据块内使用还未达到 PCTFREE 限制的块自由空间。自由列表的多少由 PCTFREE 和 PCTUSED 控制。通过查询 V$WAITSTAT 视图可以确定自由列表是否存在竞争。通过下面的过程，用户可以确定发生竞争的段名（SEGMENT_NAME）和自由列表：

1.使用V$SYSTEM_EVENT检测

```
SQL>select  event, total_waits   from v$system_event
Where event = 'buffer busy waits' ;

EVENT                              TOTAL_WAITS
---------------------------------- ------------------------
buffer busy waits                        1033
```

如果查询到 buffer busy waits 有值，还不能确定真的存在free list 问题，还要看下面的情况。

2.使用V$WAITSTAT检测

该视图的列如下：

CLASS	VARCHAR2(18)	块所属的段类型（如table,index等）
COUNT	NUMBER	访问一个块的等待次数
TIME	NUMBER	等待访问块所花的总时间

查询free list 和segment header 的count，如果count>0 ，则存在块竞争：

```
SQL>SELECT class, count  from  v$waitstat
Where  class in ('free list','segment header') ;

CLASS                                                COUNT
---------------------------------------------------- ----------------
segment header                                         12
free list                                               4
```

从结果看出，free list 存在竞争。

3.查询V$SESSION_WAITHE 和DBA_SEGMENTS 检测

V$SESSION_WAIT视图存放会话等待统计；而DBA_SEGMENTS存放每个段的相关信息。

SID	NUMBER	会话的SID号
SEQ#	NUMBER	等待顺序号
EVENT	VARCHAR2(64)	事件

P1TEXT	VARCHAR2(64)	第1次附加参数说明
P1	NUMBER	第1次附加参数
P1RAW	RAW(4)	第1次附加参数
P2TEXT	VARCHAR2(64)	第2次附加参数说明
P2	NUMBER	第2次附加参数
P2RAW	RAW(4)	第3次附加参数
P3TEXT	VARCHAR2(64)	第3次附加参数说明
P3	NUMBER	第3次附加参数
P3RAW	RAW(4)	第3次附加参数
WAIT_TIME	NUMBER	已等待时间数，0为正在等
SECONDS_IN_WAIT	NUMBER	已等待秒数
STATE	VARCHAR2(19)	状态

```
SQL>SELECT s.segment_name, s.segment_type, s.freelists
FROM dba_segments s , v$session_wait w
WHERE  w.p1 = s.header_file
AND w.p2=s.header_block
AND w.event = 'buffer busy waits' ;

SEGMENT_NAME                   SEGMENT_TYPE         FREELISTS
------------------------------ ------------------------------ ----------
EMP                            TABLE                         1
```

显然，emp表只有一个free list 是不够的。

§21.4.2 调整 free list 以减少竞争

调整free list 有两种方法：

1．增加 free list 值：

在默认下，只有一个free list 分配给段。可用下面语句来增加free list 值：

```
ALTER TABLE SCOTT.EMP  STORAGE( FREELISTS 5 );
```

2．将段移到本地自动管理的表空间上：

```
CREATE TABLESPACE APPL_DATA
DATAFILE '/u01/oradata/prod/app1_data.dbf' SIZE 300M
EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;

ALTER TABLE scott.emp move TABLESPACE appl_data;
```

§21.5 减少对并行服务器的竞争

在使用并行服务器时，也存在竞争，所以也要进行适当的调整。

§21.5.1 确定并行服务器存在竞争

可以从V$PQ_SYSSTAT视图中查看统计信息。要查看SERVERS BUSY、SERVERS IDLESERVERS STARTED及SERVERS SHUTDOWN等列的值。

- 一般并行服务器数与CPU个数和I/O带宽有关；
- 不要轻易修改并行服务器的最大数；
- 如果并行服务器经常处于停止的切换状态，则将PARALLEL_MIN_SERVERS参数加大。

1)先确定并行服务器的最大数，比如为100（PARALLEL_MAX_SERVERS=100）；
2)在确定每个操作需要多少个并行服务器及多少个并行操作同时进行；
3)比如每个用户有 2 个并行操作，平均并行度为20，则可以实例需要80个并行服务器；
4)这样就设置PARALLEL_MIN_SERVERS参数为80。

例。下面例子查询80个并行服务器　是否都处于“忙”状态：

```
SELECT * FROM V$PQ_SYSSTAT
WHERE STATISTIC = "SERVERS BUSY";
The result of this query might look like this:
STATISTIC                      VALUE
------------------------- ----------
SERVERS BUSY                       70
```

§21.5.2 降低并行服务器竞争

- 如果处于“忙”的并行服务器个数比PARALLEL_MIN_SERVERS参数的值少，则表示有空闲的并行服务器，就应该减少PARALLEL_MIN_SERVERS参数的值;
- 如果发现有多于PARALLEL_MIN_SERVERS参数的值的并行服务器在“忙”，则要增加PARALLEL_MIN_SERVERS参数的值。

第 22 章　各种锁和完整性

除前面介绍的锁存器（LATCH）的锁外，在 Oracle 系统运行中还有各种锁。这些有些是 Oracle 系统运行时所必需的，则有些是编程不严谨引起的。不管理是哪一种类型的锁，弄清它们的概念对于优化都有好处。

§22.1 锁的概念

由于系统在处理时需要锁。比如：

- 数据读的一致性

Oracle 利用回滚段的机制和锁保证各个用户对数据的查询一致性，即让用户看到数据基本上是“静态”的。

- 数据的并行性

锁可防止对同一数据进行存取的用户之间破坏性的相互影响。Oracle 系统的锁可使多个用户同时对数据进行安全的存取。

- 数据的完整性

Oracle 利用提交（COMMIT）或回滚（ROLLBACK）以前的保留锁，提供数据的完整性。

- 表级锁

某个用户正在修改表中的数据；而另一用户则试图删除整个表的数据。Oracle 为第 1 个用户取得的表级锁来锁定该表，第 2 个想要删除表的用户只能等第 1 个用户释放表级锁后才能删除该表。

- 事务级锁

用户 A 和用户 B 都在对同一个表进行修改并提交，但是用户间只能看到他本人修改的数据，而看不到对方修改并提交的数据。这是 Oracle 为了保证在一个事务中读取的数据与事务开始时保持一致。Oracle 系统使用了系统改变号(SCN)来实现事务级的读取一致性。

****
系统改变号 SCN:
系统改变号是一个数据结构，它定义了一个给定时间点提交的数据的版本（可看成 Oracle 系统的逻辑时钟）。每次提交的修改都给出一个唯一的号。比如某个事务执行了一个 UPDATE 并

提交，则系统给出一个 SCN 为 100；而下一次的另外修改或删除后的提交的 SCN 号为 101。
****

- 共享锁(SHARED LOCK)

在并行处理中，假如获得了一个共享锁，其它用户就可以一起使用该共享锁的资源。所以，多个用户可获得相同资源上的共享锁。

- 专用锁(EXCLUSIVE LOCK)

专用锁可防止多个用户共享相同的资源。一般，一个用户获得某一资源上的一个专用锁，则一直到该用户释放为止前，其它的用户是不能使用该资源。

- 数据锁(DML lock)

为了保护数据被多个用户并行存取的数据的完整性，系统采用 DML 锁来避免同时的 DML 操作的破坏性干扰。

- 数据定义锁(DDL lock)

DDL 锁也叫字典锁，当模式下的对象被一个正在运行的事务引用时，DDL 锁能够保护该模式对象的定义。例如，用户对某个对象创建一个存储过程时，它就会在过程编译期间获得该对象的 DDL 锁，也就是说，在该过程编译完成前，不能对该对象进行改变或删除。

- 系统的锁存器锁(latch)

为了保护内部的数据结构，系统需要锁存器锁，见前面章节。

- 并行高速缓存锁(PCM)

见前面介绍。

- 人工锁

在一些事务处理中，要求对表中的所有列进行全局更新，以便能在该事务不必等待其它事务完成就能运行。可用 LOCK TABLE 语句来实现。它的语法如下：

LOCK TABLE <table_name> IN <mode>

其中：
<mode>为锁的方式，可选择的项如下：

ROW SHARE (行共享)
ROW SHARE 运行并发访问所锁的表，但禁止用户为了独占而锁整个表。ROW SHARE与SHARE UPDATE同步。
ROW EXCLUSIVE (行独占)
ROW EXCLUSIVE 与 ROW SHARE类似, 但是禁止以共享锁来锁。行独占锁是在UPDATE、INSERT或DELETE时自动获得。
SHARE UPDATE （共享更新）
与 ROW SHARE类似。
SHARE (共享)
SHARE 允许并发查询但禁止更新的锁表。
SHARE ROW EXCLUSIVE （共享行独占）
SHARE ROW EXCLUSIVE(**共享行独占)用于查找整个表和允许其它用户在表中查找行，但禁止其它用户以**SHARE方式获得锁。
EXCLUSIVE (独占)
EXCLUSIVE 允许对所锁的表进行查询，但禁止其它用户在表上活动。
NOWAIT（非等待）
指定NOWAIT，如果希望Oracle将被其它用户所控制表的控制返回给你，则Oracle返回一个包括该表在内的信息给你。

锁表例子：

```
LOCK TABLE employees
IN EXCLUSIVE MODE
NOWAIT;
```

下面语句通过数据库链来锁一个远程的表：

```
LOCK TABLE accounts@boston
IN SHARE MODE;
```

- 数据库死锁

1)程序死锁或访问死锁
死锁(deadlock)是指两个或更多用户都在等待彼此之间相互锁定的同一数据而形成一种局面。

2）Oracle 的机制产生的锁
另外就是 Oracle 的共享锁，共享锁是为了保护数据库的完整性和保护原来的数据不被更新而采用的机制。在 Oracle 系统中，每一个锁都要占用 SGA 中共享池的 4 个字节，这样就可能在 SELECT 语句处理时出现 RAM 短缺。这也会损坏整个数据库的性能。比如，一条 SELECT 语句可能将 1000 行的数据读到缓冲区中，则这时需要 4000 字节的锁空间，这就容易引发数据库死锁。

****

- **单用户数据库不需要锁；**
- **Sybase 的锁：**

Sybase 只提供页级锁，在 Insert 时，所有数据插入到非 Cluster 的表的最后的页面中；
Update 时，Sybase 实际是 Delete 后再 Insert ，这样就使得 sybase 每运行一条语句后立即提交；这样在进行 Update 时，许多的 Select 语句就被阻塞了。如果你正要查询一个被修改锁定的表，就需要等待。
Informix 系统提供行级的锁定，但是行级锁很贵，因为或得和释放都要花时间。即消耗内存，必须在系统启动前指定锁的数量，在使用时超过该数则不能进行锁。所以许多情况只能是表以页面锁。所以行和页面锁都停留在跟踪的查询中。

****

§22.2 分析 v$lock

在 Oracle 系统运行中，所产生的死锁都动态地记录在 V$LOCK 视图中。视图的结构如：

```
SQL> desc v$lock
 名称                                      是否为空? 类型
 ----------------------------------------- -------- ------------
 ADDR                                      RAW(4)
 KADDR                                     RAW(4)
 SID                                       NUMBER
 TYPE                                      VARCHAR2(2)
 ID1                                       NUMBER
 ID2                                       NUMBER
 LMODE                                     NUMBER
 REQUEST                                   NUMBER
 CTIME                                     NUMBER
 BLOCK                                     NUMBER
```

其中：
SID 会话的标识号，在连接成功后由 Oracle 系统给出。
TYPE 会话中等待的锁的类型。可能值为：

MR 界质恢复
RT
XR
TS 磁盘空间事务

LMODE / REQUEST 包含锁的模式。可能的值为：

0 无
1 空
2 行共享(RS)
3 行专用(RX)

4 共享(R)
5 共享行专用锁(SRX)
6 专用(X)

ID1 为对象的标识，根据 ID1 可从 DBA_OBJECTS 查出对象的名称，（ID1=DBA_OBJECTS.Object_id）。

如果 lmode 列包含的值不是 0,也不是 1，则说明该进程获得了一个锁。
如果 request 列包含的值不是 0，也不是 1，表明该进程正在等待一个锁。
如果 request 列包含的值是 0，表明该进程正在等待获得一个锁。

§22.2.1 用 lock table 锁定表

可用 lock table 语句对需要锁定的表进行锁定，如：

LOCK TABLE EMP IN EXCLUSIVE MODE ;

此语句将在专用模式下锁定 emp 表，此时，另外的用户只能进行 select ；不能进行 update,delete,insert 及 DDL 语句操作。
可从 V$LOCK 视图中查到该会话锁的信息，如(假设发出 lock table 的会话标识为 30)：

select sid, type, lmode, request, id1, id2 from v$lock where sid =30 ;

```
sid      ty       lmode      request     id1     id2
-------- ---- ------------ -------------- ------ -------
30       TM       6                0       7500     0
```

所获得的锁为 TM 表锁;
lmode 为 6,表示获得表级专用锁 ;

id1 为 7500 ，这是 emp 表的对象标识。可从 sys.obj$字典中查到，也可从 DBA_OBJECTS 中查到。如 select obj#, name,常 owner#, TYPE# from sys.obj$ where obj# = 7500 ;

id2 为 0

select name from sys.obj$ where obj# = 7500 ;

```
NAME
-----------------
EMP
```

§22.2.2 会话更新专用锁定表的行-

update emp set name ='MANISH' where empno='1086' ;
假设发出 lock table 的会话标识为 31，可从 V$LOCK 视图中查到该会话锁的信息，如：

select sid, type, lmode, request, id1, id2 from v$lock where sid in (30,31) ;

sid	ty	lmode	request	id1	id2
30	TM	6	0	7500	0
31	TM	0	3	7500	0

31 号会话的请求(request)为 3,表示正等待获得表中的一个锁。31 号会话的事务类型仍为 TM，当 30 号会话释放锁（如 rollback 或 commit）后，在查询 V$LOCK 变为：

sid	ty	lmode	request	id1	id2
30	TM	3	0	7500	0
31	TX	6	0	327680	10834

31 号会话获得事务锁为 TX;对应的 lmode 为 6（即事务锁）。
Id1 的值 327680 为回滚段的号；可查询：
Select name from v$rollname where usn = trunc(327680/65536)
NAME

RB03

31 号会话在 RB03 回滚段上写恢复数据。

§22.2.3 一个会话试图更新另一个会话更新过的行

两个用户在**同时间**都更新同一行数据，系统只能保留最后更新的行，所以叫**“丢失的更新”**。
看下面的解释：
1) user1 查询：数据行 1；
2) user2 查询：数据行 2;
3) User1 修改：数据行 1 并提交;
4) User2 修改：数据行 1 并提交;

Oracle 的 Forms 可避免这样的事件发生。它采用在查询时不可更新，而在修改前对记录进行锁定来实现；而 VB 和 JAVA 就不能做到锁定。

下面是程序员要注意的编程习惯：

SQL>select * from emp where deptno=20 ;
(此时系统不会对数据进行行锁)。

为了要获得要更新的行处于锁，比如要修改 empno=7369 和 Ename=' SMITH'的行，则要：

SQL>select * from emp where empno=7369 and Ename=' SMITH' FOR UPDATE NOWAIT;

发出带 FOR UPDATE 语句有下面两种可能的结果：

1)由于 empno 为主键，上面的语句也只对一个数据行进行锁定。如果此时另外的用户也对该数据行进行修改，则会不能进行并收到"ORA-00054 Resource Busy."

2)如果我们发出 select * for Update 语句时，该行被另外的用户锁定，则我们会获得 0 行。当然这样只能等待该用户释放该行才能进行修改。

一旦你获得某行的锁定，就应该立即发出 UPDATE 语句和 COMMIT 语句：

SQL>UPDATE emp set sal=sal*1.5 where empno=7369 and Ename=' SMITH' ;
SQL>COMMIT;

为了确保更新的数据行是有效（不是丢失的更新），应该对 select . . . for update 语句进行确认。只有不是 0 行时才能发出 Update 语句。

另外的情况是：如果你发出了 select . . . for Update 语句时，另外的用户已经先使用了 select . . . for Update 语句；这时可能该用户离开了工作岗位，你在等待也不可能立即得到释放的数据行。这时，可从数据库中的用户管理的 RESOURCE PROFILE 来限制用户的空闲时间。

§22.2.4 显式锁下的并发例子-

下面例子说明 lock table 和 select … for update 上完整性锁的处理情况。

见 Oracle9i Database Concepts p**20-34 (p556)**

Transaction 1	**Time Point**	**Transaction 2**
LOCK TABLE scott.dept IN ROW SHARE MODE;		
Statement processed		
1		
2 **DROP TABLE scott.dept;**		
DROP TABLE scott.dept		

```
*
ORA-00054
(exclusive DDL lock not possible
because of T1's table lock)
3 LOCK TABLE scott.dept
IN EXCLUSIVE MODE NOWAIT;
ORA-00054
4 SELECT LOC
FROM scott.dept
WHERE deptno = 20
FOR UPDATE OF loc;
LOC
- - - - - - -
DALLAS
1 row selected
UPDATE scott.dept
SET loc = 'NEW YORK'
WHERE deptno = 20;
(waits because T2 has locked same
rows)
5
6 ROLLBACK;
(releases row locks)
1 row processed.
ROLLBACK;
7
LOCK TABLE scott.dept
IN ROW EXCLUSIVE MODE;
Statement processed.
8
9 LOCK TABLE scott.dept
IN EXCLUSIVE MODE
NOWAIT;
ORA-00054
```

§22.3 监控系统中的锁

当系统将锁加到 Oracle 的默认机制中时，可能就出现锁的争用问题。我们可从 V$LOCK 和 V$LOCKED_OBJECT 动态视图查到。也可从 DBA_WAITERS 与 DBA_LOCKS 字典中查到。

§22.3.1 查询 V$LOCK 是否存在争用

例子1.找出(ID1,ID2类型)会话等待一个锁(lomde=0)；找出会话持有锁（request=0）：

查询是否存在正在等待获得表锁的进程、再找该进程执行的语句。步骤如下：

步骤 1：查询是否存在正在等待获得表锁的进程：

SELECT COUNT（*） FROM V$LOCK WHERE LMODE=0;

如果返回的值大于 0，则表示系统当前有会话正等待获得锁。

ID1 与锁的类型有关，
ID2 与锁的类型有关，

步骤 2：如果存在等待锁，则再查询等待获得表锁的所有进程信息：

```
SELECT lpad(' ',DECODE(request,0,0,1))||sid, id1,id2,lmode
,request, type  FROM V$LOCK
WHERE id1 IN (SELECT id1 FROM V$LOCK WHERE lmode = 0)
ORDER BY id1,request;
```

```
SID      ID1      ID2     LMODE      REQUEST   TY
------ ---------- ---------- ---------- ---------- --
1237    196705    200493     6        0       TX <- Lock Holder
1256    196705    200493     0        6       TX <- Lock Waiter
1176    196705    200493     0        6       TX <- Lock Waiter
938     589854    201352     6        0       TX <- Lock Holder
1634    589854    201352     0        6       TX <- Lock Waiter
```

1237 控制

步骤 3：查询进程执行的语句对应的 Hash 值：

```
SELECT sid, sql_hash_value
FROM  V$SESSION
WHERE SID IN (1237,1256,1176,938,1634);
```

SID SQL_HASH_VALUE
----- --------------
938 2078523611 <-Holder
1176 1646972797 <-Waiter
1237 3735785744 <-Holder
1256 1141994875 <-Waiter
1634 2417993520 <-Waiter

步骤 4：根据语句对应的 Hash(hash_value)**值查询具体的语句：**

```
select hash_value,SQL_TEXT from v$sqlarea ;

HASH_VALUE SQL_TEXT
---------- ------------------------------------------------------------------------
1141994875 SELECT TO_CHAR(CURRENT_MAX_UNIQUE_IDENTIFIER + 1 ) FROM PO_UNI
QUE_IDENTIFIER_CONTROL WHERE TABLE_NAME = DECODE(:b1,'RFQ','PO_
HEADERS_RFQ','QUOTATION','PO_HEADERS_QUOTE','PO_HEADERS') FOR UP
DATE OF CURRENT_MAX_UNIQUE_IDENTIFIER
1646972797 SELECT TO_CHAR(CURRENT_MAX_UNIQUE_IDENTIFIER + 1 ) FROM PO_UNI
QUE_IDENTIFIER_CONTROL WHERE TABLE_NAME = 'PO_HEADERS' FOR UPD
ATE OF CURRENT_MAX_UNIQUE_IDENTIFIER
2078523611 select CODE_COMBINATION_ID, enabled_flag, nvl(to_char(start_da
te_active, 'J'), -1), nvl(to_char(end_date_active, 'J'), -1), S
EGMENT2||'.'||SEGMENT1||'.'||||SEGMENT6,detail_posting_allowed_f
lag,summary_flag from GL_CODE_COMBINATIONS where CHART_OF_ACCO
UNTS_ID = 101 and SEGMENT2 in ('000','341','367','388','389','4
52','476','593','729','N38','N40','Q21','Q31','U21') order by S
EGMENT2, SEGMENT1, SEGMENT6
2417993520 select 0 into :b0 from pa_projects where project_id=:b1 for upd
ate
3735785744 begin :X0 := FND_ATTACHMENT_UTIL_PKG.GET_ATCHMT_EXISTS(:L_ENTITY
_NAME, :L_PKEY1, :L_PKEY2, :L_PKEY3, :L_PKEY4, :L_PKEY5, :L_FUNC
TION_NAME, :L_FUNCTION_TYPE); end;
```

会话1176和1256 在等待由会话1237在PO_UNIQUE_IDENTIFIER_CONTROL中所控制的锁。
而会话1634 在等待由会话938在PA_PROJECTS中所控制的锁。
可再查询V$SESSION_WAIT, V$SESSION和V$SESSION_EVENT 得到更详细的会话与用户的信息。

§22.3.2 查询 V$LOCKED_OBJECT 是否存在争用

```
SQL> desc v$locked_object
 名称                                      是否为空? 类型
 ----------------------------------------- -------- ------------
 XIDUSN                                    NUMBER
 XIDSLOT                                   NUMBER
 XIDSQN                                    NUMBER
 OBJECT_ID                                 NUMBER
 SESSION_ID                                NUMBER
 ORACLE_USERNAME                           VARCHAR2(30)
 OS_USER_NAME                              VARCHAR2(30)
 PROCESS                                   VARCHAR2(12)
 LOCKED_MODE                               NUMBER
```

```
SELECT LPAD(' ',DECODE(1,XIDUSN,0,3,0))||L.ORACLE_USERNAME
"USERNAME" ,
O.OWNER, O.OBJECT_NAME, O.OBJECT_TYPE
FROM V$LOCKED_OBJECT L,DBA_OBJECTS O
WHERE L.OBJECT_ID = O.OBJECT_ID
ORDER BY O.OBJECT_ID,1 DESC;
```

```
USERNAME    OWNER       OBJECT_NAME    OBJECT_TYPE
----------- ----------- -------------- -----------
SCOTT          JOE              EMP              TABLE
   BRENDA      JOE              EMP              TABLE
    JOE        JOE              EMP              TABLE
ROBIN          APPS             PAYROLL          TABLE
  REGI         APPS             PAYROLL          TABLE
```

从结果可看出，scott,joe,brenda 都在给 joe 的 EMP 表中的行加锁；
robin,regi 也在给 payroll 表中的行加锁。

§22.3.3 查询 DBA_WAITERS 是否存在争用

DBA_WAITERS 字典的各列如下：

WAITING_SESSION	NUMBER	用户会话的 ID，正等待获得一个锁的会话
HOLDING_SESSION	NUMBER	用户会话的 ID，正持有 WAITING_SESSION 所希望的一个锁
LOCK_TYPE	VARCHAR2(26)	HOLDING_SESSION 对应用户所持有的锁类型
MODE_HELD	VARCHAR2(40)	HOLDING_SESSION 对应用户所持有的锁模式
MODE_REQUESTED	VARCHAR2(40)	HOLDING_SESSION 对应用户所请求的锁模式
LOCK_ID1	NUMBER	内部锁标识，代表 HOLDING_SESSION 用户持有

		的第 1 个锁（共享锁）
LOCK_ID2	NUMBER	内部锁标识，代表 HOLDING_SESSION 用户持有的第 2 个锁（独占锁）

§22.3.4 查询 DBA_BLOCKERS 是否存在争用

DBA_BLOCKERs 字典只有一个列：

```
名称                                      是否为空? 类型
----------------------------------------- -------- --------
HOLDING_SESSION                                    NUMBER
```

§22.4 管理锁的竞争

除了上面介绍的锁的竞争外，在日常工作中也存在锁的管理问题。Oracle 的锁是用在当有多个用户同时，维护数据的一致性与完整性。如果有两个以上用户试图竞争一个对象的锁时，锁就成为坏消息。因此，DBA 监视和管理锁也就是一项基本的任务。主要有：

- 管理锁竞争首先识别等待获得锁的会话；
- 第 2 步是哪个会话有阻塞锁；
- 强制持有锁的会话继续或终止。

方法 1：
Oracle 提供了两个监视和管理锁的脚本：CATBLOCK.SQL 和 UTLLOCKT.SQL。这两个脚本都可以在$ORACLE_HOME/rdbms/admin 目录找到。CATBLOCK.SQL 脚本创建许多从 V$LOCK 等视图搜集数据的视图；而 UTLLOCKT.SQL 则将查询由 CATBLOCK.SQL 创建的视图，并报告等待锁的会话及锁的有关信息。

方法 2: 直接查询数据字典视图：

```
SQL> connect as sysdba
请输入用户名:  sys
请输入口令:  ***
已连接。
SQL>
 select a.sid,serial#,process,program,terminal from v$lock a,v$session b where a.lmode=6 and a.sid > 10 and a.sid=b.sid;
```

```
SID      SERIAL#    PROCESS   PROGRAM                          TERMINAL
---------- ---------  ---------  -------------------------------- ----------------
      28      3966   429467232   F45DBG32.EXE                     Windows
      46      1438   429481395   PLUS33W.EXE                      Windows
      50      2502   429461059   F45DBG32.EXE                     Windows

3 rows selected.

SVRMGR> alter system kill session '46,1438';
Statement processed.
```

方法 3：使用 Oracle 的企业管理器。

§22.4.1 查询产生锁的 SQL 语句

可以从 V$LOCK 和 V$SESSION 进行连接查询就得到需要的信息。如：

```
set linesize 130
select s.username  username,
       a.sid  sid ,
       a.owner||'.'||a.object  object,
       s.lockwait lockwait,
       t.sql_text SQL
       FROM V$SQLTEXT t, V$SESSION s, V$ACCESS a
       WHERE t.address=s.sql_address
       AND t.hash_value=s.sql_hash_value
       AND s.sid = a.sid
       AND a.owner != 'SYS'
       AND upper(substr(a.object,1,2)) != 'V$';
```

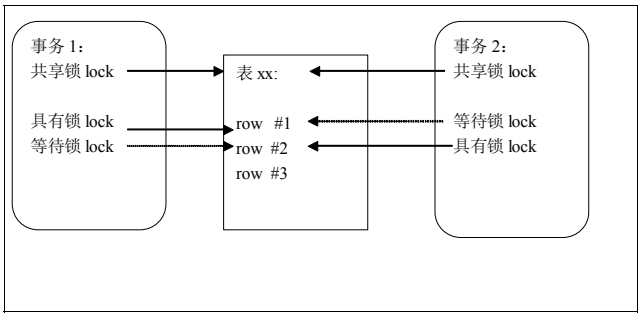
§22.4.2 怎样释放锁

可以有几种办法：

- 等待直到持有锁的会话完成他自己的事务为止；
- 通过让该用户直接提交该锁的事务来释放，比如用 commit;
- 通过让该用户直接回滚该锁的事务来释放，比如用 rollback;
- 删除持有锁的会话来终止。

§22.5 程序死锁的产生与避免

§22.5.1 程序死锁原因



处于死锁状态的两个事务（dead_lock 图）

- 事务 1 在 Row#1 有排它锁；并等待事务 2 释放 row#2;
- 事务 2 在 Row#2 有排它锁；并等待事务 1 释放 row#1;
- 两个事务互相等待对方，因而死锁。

死锁的原因是编程不科学所引起，许多情况下是由于不认真的编程引起。看下面例子。

事务 1	事务 2
UPDATE sales.parts Set onhand=onhand-10 where id=1;	UPDATE sales.parts Set onhand=onhand-10 where id=2;
UPDATE sales.parts Set onhand=onhand-10 where id=2;	UPDATE sales.parts Set onhand=onhand-10 where id=1;
等待事务 2 释放 id=2 的行	等待事务 1 释放 id=1 的行
ORA-00060: deadlock detected while waiting for resource.	

§22.5.2 避免程序死锁方法

要避免死锁也很简单，只要在使用完该行后即时的释放即可。比如在更新语句后加判断后提交或回滚。可以简单将上面例子改为：

事务 1	事务 2
UPDATE sales.parts Set onhand=onhand-10 where id=1;	UPDATE sales.parts Set onhand=onhand-10 where id=2;
COMMIT;	COMMIT;
UPDATE sales.parts Set onhand=onhand-10 where id=2;	UPDATE sales.parts Set onhand=onhand-10 where id=1;
COMMIT;	COMMIT;

§22.5.3 程序死锁例子

要编写两个互相产生死锁的存储过程，然后在 SCOTT 用户下运行。

```
/*
select * from emp;
EMPNO ENAME      JOB              MGR HIREDATE         SAL
----- ---------- --------- ---------- ---------- ----------
 7844 TURNER     SALESMAN       7698 08-9 月 -81       1500
 7876 ADAMS      CLERK          7788 23-5 月 -87       1100
 7900 JAMES      CLERK          7698 03-12 月-81        950
 7902 FORD       ANALYST        7566 03-12 月-81       3000
 7934 MILLER     CLERK          7782 23-1 月 -82       1300
*/
```

1.产生死锁的存储过程 1 deadlock1 代码如下：

```
create or replace procedure deadlock1(empno1 in number ,
 empno2 in number ) is
/* 本例子是创建两个互相产生死锁的存储过程，使 DBA 能从数据字典中查询到并解锁 */
/* 在 SQL>下运行本存储过程前登录到 SCOTT 用户，先设置：
SET SERVEROUTPUT ON SIZE 1000000; 在用 SQL>EXEC deadlock_tst(7902,7934);
这里的 7902 和 7934 是 EMP 表的雇员代码。
DBMS_OUTPUT.ENABLE(1000000);
DBMS_OUTPUT.PUT_LINE('开始运行...');
*/
begin

UPDATE emp Set  sal = 8888  where empno = empno1;
UPDATE emp Set  sal = 9999  where empno = empno2;
/*
```

```
UPDATE emp Set  sal = 6666  where empno = empno2;
UPDATE emp Set  sal = 7777  where empno = empno1;
*/
end;
```

2.产生死锁的存储过程 2 deadlock2 代码如下：

```
create or replace procedure deadlock2(empno1 in number ,
 empno2 in number ) is
begin
/* UPDATE emp Set  sal = 8888  where empno = empno1;
UPDATE emp Set  sal = 9999  where empno = empno2;
*/
UPDATE emp Set  sal = 6666  where empno = empno2;
UPDATE emp Set  sal = 7777  where empno = empno1;
end;
```

3.在 SCOTT 下分别创建上面的存储过程;

4.以 SCOTT 登录到第一个窗口，并运行存储过程 1：

```
SQL> exec deadlock1(7902,7934);

PL/SQL 过程已成功完成。

SQL>
```

不要退出。

5.以 SCOTT 再登录到第二个窗口，并运行存储过程 2：

```
SQL> show user
USER 为"SCOTT"
SQL> exec deadlock2(7902,7934);
BEGIN deadlock2(7902,7934); END;

*
ERROR 位于第 1 行:
ORA-00028: 您的会话已被删去
ORA-06512: 在"SCOTT.DEADLOCK2", line 20
ORA-06512: 在 line 1
```

实际上，当第二个窗口开始运行 EXEC DEADLOCK2 时，该窗口已经没有任何反应。只有当在另外窗口用 ALTER SYSTEM KILL SESSION 来杀掉该会话时，才出现 BEGIN …后面的提

示。

6.以 SYSTEM 登录查询 V$SESSION 视图得到死锁的会话：

SQL> select username,sid,serial#,lockwait from v$session;

USERNAME	SID	SERIAL#	LOCKWAIT
	1	1	
	2	1	
	3	1	
	4	1	
	5	1	
	6	1	
	7	2305	
	8	2305	
	9	2301	
	10	2301	
SCOTT	11	1083	
SCOTT	12	3289	**02909684**
DBSNMP	14	12	
SYSTEM	17	91	

已选择 14 行。

凡是 lockwait 列有值都表示该用户会话已经产生死锁。

这里的 02909684 就是死锁的会话，可用下面命令来解锁：

SQL> alter system kill session '12,3289';

系统已更改。

SQL>

§22.5.4 避免程序死锁例子

在上面的 UPDATE 语句后加 COMMIT 和例外处理即可：

```
create or replace procedure nodeadlock3(empno1 in number ,
 empno2 in number ) is
```

```
begin
UPDATE emp Set  sal = 8888  where empno = empno1;
Commit;
UPDATE emp Set  sal = 9999  where empno = empno2;
Commit;
/*
UPDATE emp Set  sal = 6666  where empno = empno2;
UPDATE emp Set  sal = 7777
where empno = empno1;
*/
  EXCEPTION
    WHEN others  THEN  ROLLBACK;
end;

create or replace procedure nodeadlock4(empno1 in number ,
 empno2 in number ) is
begin
/*
UPDATE emp Set  sal = 8888  where empno = empno1;
Commit;
UPDATE emp Set  sal = 9999  where empno = empno2;
Commit;
*/
UPDATE emp Set  sal = 6666  where empno = empno2;
Commit;
UPDATE emp Set  sal = 7777  where empno = empno1;
Commit;

  EXCEPTION
    WHEN others  THEN ROLLBACK;
end;
```

****

有关事务提交的建议：

- 事务是数据库的全部工作；
- 只要必要，要尽量推迟提交以减轻系统负担，有的事务要根据商业逻辑来确定；
- 如果必要，要对数据进行锁定；
- 对于锁定，不要锁定面过大。

****

第 23 章　调整回滚段竞争

回滚段是 Oracle 的一个很重要的内容。回滚段的争用也会产生性能问题。Oracle 在运行时会将一个以上用户的事务分配给单个回滚段。当两个或两个以上的用户事务在处理时都访问该回滚段的标题时，就可能发生争用。一个或多个事务必须等待该回滚段的标题再次变为可访问。所以这种等待可能使系统严重恶化。为了减少回滚段的争用，首先要确定是否存在争用。

§23.1 回滚段的用途

Oracle 的回滚段(Rollback Segment)是用于撤消操作和读的一致性及事务恢复。

事务回滚（Transaction Rollback）：
比如一个事务对表的一个行进行修改时，系统先将原来的数据（old image）存放早回滚段中。这些旧的映象叫撤消段(undo segment)。如果该事务回滚时，该撤消段数据被写回原来表的行中。

事务恢复（Transaction recovery）：
如果一个实例在运行中出现失败，则在数据库再次打开时，数据库就要恢复那些还未提交的改变事务。这种回滚就叫事务恢复。这种改变尽可能通过重做日志文件来保护。

读一致性（Read Consistency）：
读一致性包括下面条件：

- 当用户的事务在进行过程中，数据库中的其它会话看不到任何未提交的改变；
- 当语句开始执行时，语句看不到已提交的改变。

在回滚段中存放旧数据，也叫撤消信息（undo information）就是用于读的一致性。

§23.2 监测回滚段的竞争

1．查询 V$WAITSTAT 视图
可以从V$WAITSTAT视图来确定是否有回滚段的冲突。V$WAITSTAT包括一些块竞争。默认下，该表（视图）只对SYS用户有效，其它的用户如果有 SELECT ANY TABLE 系统权限的话（如SYSTEM），也能查到该表的不同类型块的竞争信息。

SYSTEM UNDO HEADER SYSTEM回滚段头块缓冲区等待数
SYSTEM UNDO BLOCK SYSTEM回滚段除头块外的缓冲区等待数
UNDO HEADER 除SYSTEM回滚段外的回滚段的块缓冲区等待数
UNDO BLOCK 除SYSTEM回滚段和除头块外的缓冲区等待数

例1．使用下面命令来查询系统运行时某个期间的统计数据(等待数)：

```
SELECT CLASS, COUNT
FROM  V$WAITSTAT
WHERE CLASS IN ('SYSTEM UNDO HEADER', 'SYSTEM UNDO BLOCK',
'UNDO HEADER', 'UNDO BLOCK');

CLASS                COUNT
------------------ ----------
SYSTEM UNDO HEADER   2089
SYSTEM UNDO BLOCK    633
UNDO HEADER          1235
UNDO BLOCK           942
```

从结果可以看出各个类型块的**等待数**。

例2.下面是整个请求数查询：

```
SELECT SUM(VALUE) FROM V$SYSSTAT
WHERE  NAME  IN ('DB BLOCK GETS', 'CONSISTENT GETS');

SUM(VALUE)
----------
929530
```

如果任何类型的等待数超过请求数的1%，则考虑建立更多的回滚段以减少竞争。

2.检测回滚段头的竞争：

```
select class, count  from  v$waitstat
where class like 'undo%';
```

3. V$rollstat, V$rollname

数据字典 V$rollstat, V$rollname 存放有回滚段使用的信息.
V$rollname：存放回滚段的编号和名字；
V$rollstat：存放回滚段的使用统计。

例子：查询数据字典 V$rollstat, V$rollname 信息：

```
Select  Rn.name, Rs.gets, Rs.waits,(Rs.waits/Rs.gets)*100 RATIO
From V$Rollstat rs, V$Rollname Rn
Where Rs.usn=m.usn;
```

这里 gets 为回滚段被访问 的 次数。
Waits 为进程等待回滚段块的次数。
比例百分比＞2% 为存在竞争，这种情况应用 Create Rollback 建立更多的回滚段. 一般情况为 4 个并发任务需用一个回滚段。

```
SQL>select name, gets , waits , xacts “Active Transactions”
From V$ROLLNAME a , V$ROLLSTAT b
Where a.usn = b usn ;
```

NAME	GETS	WAITS	Active Transactions
SYSTEM	270	0	0
RB01	22766	6	4
RB02	19197	3	4
RB03	26346	8	4
RB05	22154	5	4
RB06	15565	2	5
RB07	20310	4	4
RB08	26592	3	4

WAITS 在理想下，等待值应该是 0；
GETS 是对回滚段访问次数；

如果等待值大于 0，则肯定存在争用，可以在创建一些新的回滚段。

不同规模的系统建立的回滚段的个数。

事务个数 n	建议的回滚段个数
N < 16	4
16 <= N < 32	8
32 < = n	N / 4

§23.3 调整回滚段的竞争

§23.3.1 最小化回滚段的扩展

回滚段虽然可以扩展，但最好少扩展。我们可以查询 V$ROLLSTAT 中的动态值来确定回滚的使用情况。如果 EXTENDS(扩展次数)过大，我们也重建回滚段并将 INITIAL 值设的大些。如果扩展过后没有被退回到合适的大小，也可以将回滚段缩回（SHRINKS）到一个合适的大小中。

§23.3.2 分布回滚段的 I/O

由于回滚段存放的回滚信息，它也遇到许多读/写操作。这样的操作在 OLTP 中很常见。如果回滚段都建立在一个单一的表空间里（通常是 RBS 表空间），则容易导致 I/O 瓶颈。因此建议在多个磁盘中创建多个回滚表空间以达到将回滚分布的目的。如：

回滚段名	回滚段表空间	包含数据文件的磁盘
R01	Rbs1	Disk1
R02	Rbs2	Disk2
R03	Rbs3	Disk3
R04	Rbs4	Disk4
R05	Rbs5	Disk5
R06	Rbs6	Disk6

§23.3.3 调整回滚段的存储参数

- INITIAL参数

可从8k开始；多达10m左右。如果事数据较大，则设置大些。

- NEXT参数

一般可设置NEXT与INITIAL一样大。也可小些。一般INITIAL都大于NEXT参数的值。

- 创建一些相同大小的回滚段，并使它们联机。
- 创建一个大的回滚段，不用时，可将其设置为脱机。
- 如果 INITIAL 设置较小，可设置 minextents 为 20 。
- 设置 OPTIMAL 参数确保表空间在事务完成后能释放一定的空间。
- 将不需要恢复的全表 DELETE 删除语句成 TRUNCATE 语句。
- 可从 V$TRANSACTION 和 V$SESSION 中查询到事务中回滚段所使的大小（块数），如：

```
select  s.username,  t.used_ublk,  t.start_time
from  v$transaction t , v$session s
where  t.addr = s.taddr ;
```

§23.3.4 使用较小的回滚段

- **应用要有规律的提交**

用户应该有规律地提交事务以便释放被锁住的回滚段：

- **不要开发运行时间长的应用**

开发人员不要编写太长的应用。

- IMP 中加 COMMIT=y 确保每个插入组都被提交
- BUFFER_SIZE 指定数组大小
- 在 EXP 中加 CONSISTENT=N，从只读中保护数据。
- 在 SQL*LOADER 中设置 rows 参数以尽快提交。

§23.4 Oracle9i 的自动撤消管理

§23.4.1 自动管理撤消

Oracle9i 提供一个自动撤消管理功能。用于在 Oracle9i 的数据库中管理撤消表空间的新方法。在 Oracle8i 以前的版本，只能用回滚段来撤消操作。DBA 要管理回滚段的大小、数量及位置等。现在，这些工作叫回滚段撤消(Rollback Segment Undo=RBU)。

你可通过设置 UNDO_MANAGEMENT=AUTO 或 MANUAL 来使回滚段为自动或手工管理。

- UNDO_MANAGEMENT=AUTO

使实例管理回滚段自动(Rollback Segment Automatically=AUM)。

- UNDO_MANAGEMENT=MANUAL

手工建立和管理回滚段。

- UNDO_RETENTION

指定事务提交后撤消信息能维持多久（秒数），默认为 900 秒。可避免事务急需空间而覆盖掉原来的数据，从而避免“Snapshot Too Old”错误。

- UNDO_SUPPRESS_ERRORS

当设置为 TRUE 时，用于抑制在 AUM 模式下所引发的错误。比如：alter rollback segment online 等。

§23.4.2 自动撤消管理表空间的建立

- 在创建数据库创建撤消管理表空间
- 用 CREATE UNDO TABLESPACE 创建撤消表空间
- 不能在撤消表空间上建立任何对象
- 你可指定数据文件和 extent_management 选项
- MINIMUM EXTENT 和 DEFAULT STORAGE 由系统生成。

在创建数据库时创建UNDO TABLESPACE

```
CREATE DATABASE mynewdb
USER SYS IDENTIFIED BY pz6r58
USER SYSTEM IDENTIFIED BY y1tz5p
LOGFILE GROUP 1 ('/vobs/oracle/oradata/mynewdb/redo01.log') SIZE 100M,
GROUP 2 ('/vobs/oracle/oradata/mynewdb/redo02.log') SIZE 100M,
GROUP 3 ('/vobs/oracle/oradata/mynewdb/redo03.log') SIZE 100M
MAXLOGFILES 5
MAXLOGMEMBERS 5
MAXLOGHISTORY 1
MAXDATAFILES 100
MAXINSTANCES 1
CHARACTER SET US7ASCII
```

```
NATIONAL CHARACTER SET AL16UTF16
DATAFILE '/vobs/oracle/oradata/mynewdb/system01.dbf' SIZE 325M REUSE
EXTENT MANAGEMENT LOCAL
DEFAULT TEMPORARY TABLESPACE tempts1
DATAFILE '/vobs/oracle/oradata/mynewdb/temp01.dbf'
SIZE 20M REUSE
UNDO TABLESPACE undotbs
DATAFILE '/vobs/oracle/oradata/mynewdb/undotbs01.dbf'
SIZE 200M REUSE AUTOEXTEND ON NEXT 5120K MAXSIZE UNLIMITED;
```

用CREATE UNDO TABLESPACE命令创建撤消表空间

```
CREATE UNDO TABLESPACE undotbs_02
DATAFILE '/u01/oracle/rbdb1/undo0201.dbf' SIZE 2M REUSE AUTOEXTEND ON;
```

§23.4.3 检测自动撤消管理表空间的使用

撤消空间视图：

V$UNDOSTAT
包含有撤消空间的监视和调整信息。使用本视图可得到当前撤消空间的工作量。Oracle也用本信息帮助调整系统的撤消空间。本视图在自动撤消管理和手工撤消管理都有效。

V$ROLLSTAT
在自动撤消管理模式下，视图信息反映撤消表空间的撤消段的行为。

V$TRANSACTION
包含撤消段信息。

DBA_UNDO_EXTENTS
显示撤消表空间中每次扩展提交次数。

监视UNDO TABLESPACE:

V$UNDOSTAT视图监视当前实例在撤消空间中事务执行情况。统计撤消空间消费、事务并发、查询长度的有效信息。视图中每行包含每10秒钟所搜集的统计数据。行是随着BEGIN_TIME列而递减。每个行与时间间隔(BEGIN_TIME, END_TIME)有关。每个列反映字时间间隔内特殊统计搜集。第1行视图包含当前时间内的统计。视图包含1008行，是7天一个周期生成的数据。

例1. 查询V$UNDOSTAT部分列:

```
SELECT BEGIN_TIME, END_TIME, UNDOTSN, UNDOBLKS, TXNCOUNT,
MAXCONCURRENCY AS "MAXCON"  FROM V$UNDOSTAT;

BEGIN_TIME END_TIME UNDOTSN UNDOBLKS TXNCOUNT MAXCON
------------------- ------------------- ------- -------- -------- ------
07/28/2000 18:26:28 07/28/2000 18:32:13 2 709 55 2
07/28/2000 18:16:28 07/28/2000 18:26:28 2 448 12 2
07/28/2000 14:36:28 07/28/2000 18:16:28 1 0 0 0
07/28/2000 14:26:28 07/28/2000 14:36:28 1 1 1 1
07/28/2000 14:16:28 07/28/2000 14:26:28 1 10 1 1
...
```

本例子是从18:32:13 开始的24小时的撤消空间的消费列表。

例2. 查询V$UNDOSTAT全部列:
下面例子显示撤消空间从16:07开始的24小时前的消费情况。

```
SELECT * FROM V$UNDOSTAT;
End-Time UndoBlocks TxnConcrcy TxnTotal QueryLen ExtentsStolen SSTooOldError
-------- ---------- ---------- -------- -------- ------------ -------------
16:07       252     15     1511     25       2              0
16:00       752     16     1467     150      0              0
15:50       873     21     1954     45       4              0
15:40       1187    45     3210     633      20             1
15:30       1120    28     2498     1202     5              0
15:20       882     22     2002     55       0              0
```

在统计搜集中，你可看出撤消消费的高峰是在15:30 至 15:40 之间。 1187撤消块在10分钟内消费（每秒两块）。最高的事务高峰是45个。最长的查询(花1202秒)在15:20至15:30之间执行。实际查询是间隔进行（15:00→15:10）并继续到15:20为止。

第 24 章 调整共享服务器

Oracle9i 版本的共享服务器(Shared Server)就是 Oracle7 和 Oracle8/8i 版本的多线程服务器(MTS)。它允许多个用户进程同时共享几个有限的服务进程。

在使用共享服务器时，某些数据库功能，比如BFILE、并行执行等，可能影响系统的一些性

能。原因是：当会话处于激活时，上述的功能可能会防碍会话从其它共享服务器中移植。如果出现竞争，可以考虑用下面的方法解决：

- 使用特定的调度程序视图来鉴别竞争存在；
- 降低对调度程序的竞争；
- 降低对共享服务器的竞争；
- 确定最佳的调度程序和共享服务器数目。

§24.1 查询调度程序视图监测竞争

可用写视图来获得竞争信息：

- V$MTS
- V$DISPATCHER
- V$DISPATCHER_RATE

1. V$DISPATCHER

V$DISPATCHER视图提供调度程序的通用信息。

```
SQL>
select network “protocol”, status “status”,
Sum(owned) “clients” ,
Sum(busy)*100/(sum(busy)+sum(idle)) “busy rate”
From v$dispatcher  group by network, status ;
```

```
protocol                                                          status            clients  busy rate
----------------------------------------------------------------- ---------------- ---------- ----------
(ADDRESS=(PROTOCOL=tcp)(HOST=D8MYY61X)(PORT=1030)) WAIT                    0         0
```

2. V$DISPATCHER_RATE

CUR_ 当前会话的统计信息；
AVG_ 自搜集以来的统计信息平均值；
MAX_ 自搜集以来的统计信息最大值；

- 当前值与最大值比较；
- 当前值接近 平均值，且低于最大值， 则MTS优化比较合适；
- 如果当前的速率和平均速率 低于最大值，则需要减少调度程序的数目；
- 如果当前的速率和平均速率 接近最大值，则需要增加调度程序的数目；

§24.2 降低调度程序竞争

1.如何鉴别调度程序的竞争

如果存在下面的情况，就说明存在调度程序的竞争：

- 对于现在正在调度的进程，存在过多的等待；
- 对于调度进程队列的响应，等待时间在增加。

2.检查调度进程的“忙”速率

查看V$DISPATCHER中的下面列的值：

IDLE 调度进程的闲置时间（0.01秒）；
BUSY 调度进程的忙时间（0.01秒）。

要得到反映实际的值，要考虑高峰和低峰的情况，如果一天内有50%的时间在忙，则需要增加调度进程数了。

3.检查调度进程响应队列的等待时间

从V$QUEUE视图中查询：

WAIT 所有曾经位于队列中的进程的响应总时间（0.01秒）
TOTALQ 所有曾经位于队列中的响应数目

```
SELECT CONF_INDX "INDEX",
DECODE( SUM(TOTALQ), 0, 'NO RESPONSES',
SUM(WAIT)/SUM(TOTALQ) || ' HUNDREDTHS OF SECONDS')
"AVERAGE WAIT TIME PER RESPONSE"
FROM V$QUEUE Q, V$DISPATCHER D
WHERE Q.TYPE = 'DISPATCHER'
AND Q.PADDR = D.PADDR
GROUP BY CONF_INDX;

INDEX    AVERAGE WAIT TIME PER RESPONSE
-------- ------------------------------
0        .1739130 HUNDREDTHS OF SECONDS
1        NO RESPONSES
```

从结果看，第1个MTS_DISPATCHER在队列中平均响应时间0.17百分之一秒。第2个MTS_DISPATCHER在队列中没有得到响应。

4.增加调度进程数

- 可以用ALTER SYSTEM SET MTS_DISPATCHER=number来增加调度进行数；
- 调度进程数受 初始化文件的MTS_MAX_DISPATCHER的限制；

- MTS_MAX_DISPATCHER的默认值为 5 ；
- MTS_MAX_DISPATCHER 的最大值与操作系统有关。

5.启用连接池机制

当系统的负载增加时，调度程序的吞吐量已经达最大化。简单增加调度进程数可能不是好办法。另外的办法是设置MTA_DISPATCHER参数值来解决，如：

MTS_DISPATCHERS = "(PROTOCOL=TCP)(POOL=ON)(TICK=1)"

这里POOL是连接池，即启用NET8连接池。TICK是网络的TICK时间（秒），默认为１ 5秒。

6.启用连接集中制

可以采用多路复用技术，因为多路复用技术能够最大化用户与调度进程之间的连接。如：

MTS_DISPATCHER=”(PROTOCOL=TCP)(MULTIPLEX=ON)”

关于MTS_DISPATCHER参数的设置详细见“NET8管理员指南”

§24.3 降低对共享池的竞争

1.如何识别共享池存在竞争

如果请求队列中等待事件增加，则存在共享池竞争。
另外可以查询V$QUEUE视图来确定是否存在竞争：

```
SELECT DECODE(TOTALQ, 0, 'No Requests',
WAIT/TOTALQ || ' HUNDREDTHS OF SECONDS')
"AVERAGE WAIT TIME PER REQUESTS"
FROM V$QUEUE
WHERE TYPE = 'COMMON';
```

显示结果如下：

```
AVERAGE WAIT TIME PER REQUEST
------------------------------
.090909 HUNDREDTHS OF SECONDS
```

从结果看出，平均等待0.09百分之一秒。

另外还可以确定当前并发的共享服务器的数目：

```
SELECT COUNT(*) "Shared Server Processes"
FROM   V$SHARED_SERVER
WHERE STATUS != 'QUIT';
```

The result of this query might look like this:

```
SHARED SERVER PROCESSES
----------------------
10
```

如果检测出是MTS竞争所引起，则分析共享池。要降低对共享池的竞争：

1)设置和调整MTS

设置下面参数：

- MTS_MAX_DISPATCHERS 调度程序的最大数
- MTS_MAX_SERVERS 服务器的最大数
- MTS_DISPATCHERS 调度程序的数目
- MTS_SERVERS 服务器的数目

一般MTS_MAX_SERVERS 的值是 MTS_SERVERS的 2 倍。

2)自我调整MTS体系结构

- 数据库启动后，将根据MTS_SERVERS创建由该参数所要求的数目的共享服务器；
- Oracle不允许少于MTS_SERVERS所要求的数目；
- 在数据库活动中，Oracle可以根据情况增加附加的共享服务器，因为Oracle就自动加共享服务器的数目；
- 所附加的共享服务器数目最多达 MTS_MAX_SERVERS所要求的数；
- 根据上面的特点，用户所能修改的就是MTS_MAX_SERVERS和MTS_SERVERS；
- 用户可以不断地调整这两个参数值以达到理想的性能。

3)设置MTS的上限与MTS_MAX_SERVERS相等

如果是共享服务器的数目不够，性能就会下降。可以设置 MTS_SERVERS 的值与 MTS_MAX_SERVERS的值一样。

- 看共享服务器的最大数目：

SHOW parameter MTS_MAX_SERVERS

* 检查共享服务器的最高上限：

```
SELECT maximum_connections "MAXIMUM_CONNECTIONS",
servers_started "SERVERS_STARTED",
servers_terminated "SERVERS_TERMINATED",
servers_highwater "SERVERS_HIGHWATER"
FROM V$MTS;
```

结果显示为：

```
MAXIMUM_CONNECTIONS SERVERS_STARTED SERVERS_TERMINATED SERVERS_HIGHWATER
------------------- --------------- ------------------ -----------------
                 60              30                 30                50
```

这里的 HIGHWATER 应当与MTS_MAX_SERVERS相等。
MAXIMUM_CONNECTIONS 单个调度进程所能处理的最大连接数；
SERVERS_SHARED 已经启动的共享服务器的总数；
SERVERS_TERMINATED 已经终止的共享服务器的总数。

2.如何增加共享服务器的数目

共享服务器是一种进程，它们执行数据库访问并将信息返回给调度程序。调度程序再将 数据传给客户进程。如果没有 足够的共享服务器来处理所有的请求，就会出现队列的进行备份（V$QUEUE）而处理时间延长。

- 检查V$QUEUE 中统计信息，是否用完目前的共享服务器的数；
- 如果RAM存在空闲，可以增加共享服务器的数；
- 可以不但增加 MTS_MAX_SERVERS的值，直到出现内存交换(swapping)为止，交换是由于共享服务器引起的。如果出现交换就要减少共享服务器的数；
- 用户也可以增加RAM来确保不出现交换；
- 修改MTS_MAX_SERVERS值，建议对初始化参数文件进行编辑；
- 如果确定是机器引起时间的延长，则将MTS_SERVERS设置与MTS_MAX_SERVERS一样。

§24.4 确定最佳的调度程序和共享服务器数

MTS_SERVERS参数可以激活共享服务器的数量，在指定了MTS_DISPATCHERS情况，MTS_SERVERS参数的默认值为1 。

1.确定调度进程的最大数

调度进程最大数MTS_MAX_DISPATCHERS和MTS_DISPATCHERS的取值应当等于最大并发会话数 / 每个调度程序所允许连接数。大多数系统每个调度程序支持1,000个连接时性能比较

好。

2.禁止用户以并发来使用MTS

如果希望不让其他用户来访问共享服务器，可以用 ALTER SYSTEM SET MTS_SERVERS=0 来实行用户不能访问共享服务器。如果再将参数设置为一个正的值，则当前的用户又能可以访问共享服务器。

第 25 章 操作系统与网络调整

本章主要 讨论下面三个题目：

- 理解操作系统性能问题
- 检测操作系统性能问题
- 解决操作系统问题

本章内容可参考 Oracle 公司《Oracle8i Designing and Tuning for Performance》中的
chapter 23.Tuning the Operating System 或
Oracle9i Database Performance Tuning Guide and Reference
Release 2 (9.2) Part No. A96533-01

§25.1 理解操作系统性能问题

操作系统问题通常涉及到进程管理、内存管理及时序安排等。如果你调整Oracle实例后仍没有得到好的性能的话。你应该检查一下你的工作或减少系统时间。确保有足够的I/O带宽、CPU能力及交换区，不要作任何的猜测。接下来就是调整操作系统使应用性能有效。改变配置或应用。

§25.1.1 操作系统与硬件高速缓存

操作系统和设备控制器提供的高速缓存一般不会直接与Oracle自己的高速缓存管理有直接的竞争，但是，当所系统提供的很小高速缓存时，这种结构仍是消耗资源的。在UNIX下，有数据库文件存储在目录下。所有的I/O都可以是通过文件系统来进行访问。有些操作系统可直接对文件存储进行I/O操作。这样就要求就允许访问UNIX文件系统、迂回(bypassing)的文件系统。当然这样的情况在NT不存在。

§25.1.2 原始设备系统

评估你的原始设备(或裸设备)的使用，使用原始设备涉及到一项大量的工作。但它会带来性能上的好处。原始设备也以全表扫描换来代价。如果执行不支持“全部写”高速缓存，则在UNIX上执行全表扫描就是必须的。当服务器启动读连续的数据块时，可通过读文件头来加速全表扫描的速度。这时系统也缓存表扫描的数据。如果 UNIX 系统在写文件系统时不支持“全部写”选择，那么使用原始设备就是必须的。另外的情况是，如果 Oracle 是并行服务器的话，则只能选择原始设备系统。使用原始设备有下面优点：

- 原始设备消除了与使用 UNIX I/O 缓冲区相关的性能开销；
- 由于不使用缓冲区 I/O，可为系统节省大量内存。

§25.1.3 进程调度程序

许多进程或NT上的“线程”都与Oracle的操作有关。它们都访问在SGA上共享内存资源。要确保Oracle的所有进程（包括Oracle的后台进程和用户进程）都有相同的优先级。当Oracle安装完成后，所有的后台进程都根据操作系统给出默认的优先级。这些后台进程的优先级是不能改变的。只能让用户的进程也有同样的优先级。
假定不同的优先级确实给Oracle进程带来不好的效果。则如果某个高级别的进程也请求处理时间时的话。你可以不授权级别低的进程请求处理时间。如果级别高进程需要访问由级别低的进程所控制的内存的话，在高级别的进程可以不确定地等待级别低的进程释放资源而得到该CPU。另外，不要将Oracle的后台进程与CPU进行绑定。这容易引起进程CPU缺乏。尤其是交叉绑定更是如此。这种情况容易使父进程和所有的线程都绑定到专门的CPU上。

§25.1.4 操作系统资源管理器

有些平台提供了操作系统资源管理器，这主要是在按级别访问系统资源时，为了减少负荷加载竞争而设计的。这些工具通常执行用户访问资源的管理和各个用户的资源的限制等。
操作系统资源管理器不同于域管理或其他类似的工具。域管理器是在一个系统盘、CPU、内存里提供一个或多个完成分开的环境。并且所有其他的资源归到每个域中，并且不允许其他的域来访问。而其他的类似工具恰好是将系统的资源分成不同的域。通常分为CPU/或内存区。与域类似，分开的资源区也只能分配给一个进程；进程不能跨越区域进行分配。与域不同的是，所有的资源（通常是磁盘）在系统中有所有的分区来访问。假定分区内存资源的分配是固定（非动态）的话，在域中运行的Oracle也很少建立分区。去掉RAM分配使得内存板（条）复位是动态改变内存的一个例子。这也是Oracle不支持的环境变量的例子。
操作系统资源管理器按照资源的全局区（通常是域或整个系统）来等级分配资源。进程被分配到组，这些组在资源的全局区被轮换地分配资源。

§25.2 检测操作系统性能问题

检查操作系统的性能问题主要有：

- CPU 负荷
- 设备队列
- 网络活动（队列）
- 内存管理(内存分页/交换)

检查CPU的使用情况以确定在***应用方式下***运行时间消耗与在***操作系统方式***下运行时间的消耗的比率。通过查找运行队列可找出有多少进程在运行和有多少系统调用被执行。查看有无页分配和页交换产生以及检查I/O 执行的的数量及扫描的比例。

§25.3 解决操作系统性能问题-

§25.3.1 调整服务器内存相关的参数

在调整SGA中，如果你的Oracle版本是Oracle8i及以前版本时，则DBA可在INIT.ORA参数以满足数据库实例的需要；如果你的Oracle版本是Oracle9i，则可在INIT.ORA中设置参数；也可能在会话中设置相关的参数。但是，这些SGA参数与操作系统的核心参数有关，主要有：

参数	描述
SHMMAX	一个共享内存段（如SGA）的最大长度（字节）
SHMMIN	一个共享内存段（如SGA）的最小长度（字节）
MAXFILES	单个进程可以利用的文件（如数据文件）数量
NPROC	服务器上可同时运行的进程最大数

注：本部分内容可参考Oracle产品的安装手册。

为了使Oracle系统运行比较好话。要注意下面的方面：

- UNIX和WINDOWS 2000服务

每个操作系统都有OS 级的服务程序。这些服务程序一般都在服务器引导时自动被启动。如UNIX上的打印机服务、卷管理器、X-Windows服务；TELNET、cron, ftp 等。这些服务都消耗部分内存。

- 应用程序和WEB服务器

有的数据库服务器还提供带有应用程序的业务逻辑或表示层的WEB服务器。例如，Oracle的电子商务应用程序包等。这些软件如果在一台机器上运行，也会与Oracle系统竞争内存。

- 软件代理

与Oracle自己的Intelligent Agent 、OEM 类似，第三方的备份与监测软件，如：Simple Network Management Protocol(SNMP)等都会占用内存。

- 应用软件

一些服务器可允许将应用软件包安装在上面。这些软件包括WEB浏览器、Perl应用程序等。一般不建议将这些程序放在服务器上运行。因为它们会占大量的内存。

- 页面管理

一般内存分为OS 内核组件 、Oracle 内存结构与后台进程。这些内存都是具体的物理内存。现在服务器的物理内存(RAM)可大2GB以上。物理内存可再分成更小的片段，这些片段叫页面。操作系统用一种叫页面管理机制**将页面分配给 进程和数据** 。如果物理内存被装满了数据，操作系统必需将页面或进程从内存中移出到虚拟内存（即磁盘）上。以便释放出空间为

其它请求页面或进程使用。这一过程叫“调度”或“交换”。
页面交换所用方法有：

页面调度	页面交换
单个内存页面临时地从内存中被移走，并写到磁盘上。以便为另一个请求**页面**腾出空间。	整个进程从内存中移出，并写到磁盘上。以便为另一个请求**进程**腾出空间。

当页面发生页面调度或页面交换时，虚拟内存（即磁盘）就用来存放从内存移走的页面回进程的副本。与Oracle的SGA类似，OS调整的目标是：在内存中存放最大化被请求页面和进程，以防止从磁盘中读取副本。

- UNIX上监视页面调度和进程交换

命令	描述
ipcs	显示服务器上所有共享段的虚拟内存大小
ps	带 –l -u 显示单个进程所使用的虚拟内存数
vmstat	显示服务器上调度页面的系统级信息
sar	带 -r –g –p，显示系统页面调度信息

§25.3.2 基于 UNIX 系统的执行

在UNIX系统中，可以考虑建立操作系统和应用系统的运行时间的比例，比如目标是应用方式的时间是60%到70%；操作系统的运行时间是25%到40%，而已经找出每次操作系统都用去50%时间，则说明不合理。这些时间的花费的比例仅是一种征兆，因为他们还要涉及到：

- 交换
- 执行过多的操作系统调用
- 运行过多的进程等。

如果以上这些情况都存在，则要减少应用的运行时间，另外，你可以从操作系统中来释放资源，使得可以运行较多的应用。

§25.3.3 基于 NT 系统的执行

在NT中，可以用性能监视器来监视系统的性能，如：CPU、网络、I/O及内存。从而避免瓶颈的发生。

§25.3.4 基于大型系统的执行

在大型系统中，要考虑页交换的问题，并且记住，Oracle 可以设置一个很大的参数工作集。
在VAX/VMS环境中，自由内存就是实际内存，所有该实际内存就不能映射到任何操作系统进程中。在一个很忙的系统中，自由内存包括的一个页是属于一个或多个实际的进程的。当访问出现时，软件页故障回发生，并且这些页包括有工作集的进程。如果进程不能扩展它的工作集的话，则进程所映射（占用）的某个页就必须被移到自由集中去。
在工作集中的任何进程都可以有共享内存页。工作集的大小总和可以超出有效的内存大小。
当Oracle系统正在运行时，SGA区、Oracle核心代码及Oracle Forms可执行程序都是共享的。而且可能有80%到90% 的页是可访问的。增加更多缓冲区可能是不必要的，因为每个应用在命中率的都有一个停止上升的极限值。所以简单地设置较高的值只能增加Oracle和操作系统的管理负荷。

§25.4 网络系统性能问题

对于只要不是单机的Oracle系统，都有网络性能问题。也就是说，在调整Oracle系统参数及应用程序后，有必要对网络进行一些检查和调整。

§25.4.1 网络的连接模式

可以有三种的连接方式：

- 多线程服务器配置（MTS）
- 专用服务器配置
- 预产生专用服务器配置

§25.4.1.1 多线程服务配置

1.注册调度程序

用lsnrctl services 来列出所注册调度的程序。如：

```
lsnrctl services:
LSNRCTL for Solaris: Version 8.1.6.0.0 - Production on 27-MAY-99 13:38:02
(c) Copyright 1999 Oracle Corporation. All rights reserved.
Connecting to (ADDRESS=(PROTOCOL=TCP)(Host=ecdc2)(Port=1521))
Services Summary...
ORCL has 2 service handler(s)
DEDICATED SERVER established:0 refused:0
LOCAL SERVER
DISPATCHER established:0 refused:0 current:0 max:1 state:ready
```

D000 <machine: ecdc2, pid: 16011>
(ADDRESS=(PROTOCOL=tcp)(DEV=20)(HOST=144.25.216.223)(PORT=55304))
The command completed successfully.

2.配置初始化参数文件

要使用MTS，就要在INIT.ORA文件设置下面的参数：

1)加MTS_DISPATCHER参数：

MTS_DISPATCHERS =
"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=
hostname)(PORT=1492)(queuesize=32
)))
(DISPATCHERS= 1)
(LISTENER = alias)
(SERVICE = servicename)
(SESSIONS = 1000)
(CONNECTIONS = 1000)
(MULTIPLEX = ON)
(POOL = ON)
(TICK = 5)"

2)设置MTS_MAX_DISPATCHER参数：

此参数表示调度进程的总数。如：

MTS_MAX_DISPATCHERS = 4

3)设置MTS_MAX_SERVERS参数：

要设置共享服务器的最大个数。如：

MTS_MAX_SERVERS = 5

可以从V$MTS数据字典中查出目前最大的连接数。如：

SELECT maximum_connections "MAX CONN", servers_started "STARTED",
servers_terminated "TERMINATED", servers_highwater "HIGHWATER"
FROM V$MTS;

4)设置MTS_SERVERS参数：

要设置共享服务器的个数。如：

MTS_SERVERS = 5

§25.4.1.2 注册检查

1.连接情况检查：

要用 lsnrctl 实用工具对连接情况进行检查。比如是否存在“连接拒绝（refused:x）”。正常情况下，连接拒绝的数应该是0，即有 refused: 0 提示 。看下面的信息显示：

```
LSNRCTL> set displaymode normal
Service display mode is NORMAL
LSNRCTL> services
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=net)(QUEUESIZE=32)))
Services Summary...
Service "net.regress.rdbms.dev.us.oracle.com" has 1 instances.
Instance "net"
Status: READY Total handlers: 3 Relevant handlers: 3
DEDICATED established:0 refused:0 current:0 max:0 state:ready
Session: NS
D001 established:0 refused:0 current:0 max:16383 state:ready
(ADDRESS=(PROTOCOL=tcp)(HOST=dlsun1012.us.oracle.com)(PORT=52217))
Session: NS
D000 established:0 refused:0 current:0 max:16383 state:ready
(ADDRESS=(PROTOCOL=tcp)(HOST=dlsun1012.us.oracle.com)(PORT=52216))
Session: NS
```

可以采用重启lsnrctl 来得到最新的连接信息。

2.检查连接速率：

可以从监听连接日志文件中查看连接情况。

- 如果请求超过监听器所设的限制，则要进行请求排队。因而用户可以用 QUEUESIZE=number 参数对排队的数进行设置。
- 如果存在多个监听器，则排队就少些。因而也就提高处理速度。
- 可以设置多个监听器，每个监听器都设置排队数。

例。在监听器中加排队数：

```
listener =
(address =
(protocol = tcp)
(host = sales-pc)
```

(port = 1521)
(queuesize = 20)
)

§25.4.1.3 预产生专用服务配置

预产生(Pre_Spawned)专用服务器可以在没有MTS下改善与专用服务器的连接时间。用户可以通过启用预产生进程，以使监听器能将连接重定向到其它的进程中，在连接出现时，这些进程不需要等待。所以连接请求也就不必等待新的进程启动。

1.检查预产生专用服务器数：

要配置预产生专用服务器，要在监听器上加PRESPAWN参数，如：

LISTENER =
(ADDRESS_LIST =(ADDRESS= (PROTOCOL= TCP)(Host= ecdc2)(Port= 1521)))
SID_LIST_LISTENER =
(SID_LIST =
(SID_DESC =
(ORACLE_HOME = /u01/oracle/product/oracle/8.1.6)
(SID_NAME = ORCL)
(PRESPAWN_MAX = 12)
(PRESPAWN_LIST =
(PRESPAWN_DESC =
(PROTOCOL = TCP)
(POOL_SIZE = 1)
(TIMEOUT = 1)))))

这样配置后，有下面的结果：

- 当客户发出连接时，在系统内创建影子进程，以缩短连接时间；
- 连接完成后，监听器再为下一个连接创建影子进程；
- 影子进程的数量可以控制，当达到影子进程的最大数时，新的连接就用专用连接；
- 没有用户连接，预产生服务器的数＝POOL_SIZE的数；
- 如果有连接，则连接数在POOL_SIZE和PRESPAWN之间。

上面例子表示在没有数据库活动时，可只有 1 个预产生服务器；当数据库活动频繁（多个用户连接）时，预产生服务器最大可大 1 2 个。

用户可以用 lsnrctl services 来检查预产生服务器的数目。如：

lsnrctl services:

LSNRCTL for Solaris: Version 8.1.6.0.0 - Production on 26-MAY-99 18:22:49
(c) Copyright 1999 Oracle Corporation. All rights reserved.
Connecting to (ADDRESS=(PROTOCOL=TCP)(Host=ecdc2)(Port=1521))
Services Summary...
ORCL has 2 service handler(s)
DEDICATED SERVER established:0 refused:0
LOCAL SERVER
PRESPAWNED SERVER established:0 refused:0 current:0 max:1 state:ready
PID:15587
(ADDRESS=(PROTOCOL=tcp)(DEV=8)(HOST=144.25.216.223)(PORT=55221))
The command completed successfully

在UNIX下，可以用下面命令查看：

ps -ef | grep oracle
oracle 15587 1 0 17:54:21 ? 0:00 oracleORCL /
(DESCRIPTION=(COMMAND=prespawn)(PROTOCOL=TCP)(SERVICE_ID=2)(HANDLER_

2.设置检查故障参数：

在监听器参数中加 SERVER = DEDICATED 可以实现故障检测。

3.确定有足够的内存：

如果要启用预产生专用服务器，需要配置合理的SGA区。主要考虑因素是：

- 并发连接数；
- 预期的连接速率。

§25.4.2 检测网络故障

§25.4.2.1 使用动态数据字典检测

可以查询下面的动态数据字典来查看网络信息。主要是：

- V$SESSION_EVENT 主要看sqlnet message from client
- V$SESSION_WAIT 等待的对象
- V$SESSTAT看发送到客户端的字节、客户发出的次数。

§25.4.2.2 了解网络环境的速率

主要包括：

- 以太网

- 快速以太网
- 千兆以太网
- 令牌环网
- FDDI
- ATM

广域网要考虑：

- DSL
- ISDN
- 帧中继
- T-1, T-3, E-2, E-3
- ATM
- SONAT

拓普结构与载体	带宽
以太网	10 M / 每秒
快速以太网	10 0 M / 每秒
千兆以太网	1GB/每秒
令牌环网	16M/每秒
FDDI	100M/每秒
ATM	155M/每秒(OC3)；622M/每秒(OC12)
T-1(美国)	1.544M/每秒
T-3(美国)	44.736M/每秒
E-1	2.048M/每秒
E-3	34.368M/每秒
帧中继	可大载波速率，一般不用
DSL	可大载波速率
ISDN	可大载波速率
拨号解调器	56K/每秒

§25.4.3 解决网络故障

§25.4.3.1 分析瓶颈所在

1.跟踪客户与服务器之间的路由。如：

tracert usmail05
Tracing route to usmail05.us.oracle.com [144.25.88.200]over a maximum of 30
hops:
1 <10 ms <10 ms 10 ms whq1davis-rtr-749-f1-0-a.us.oracle.com
[144.25.216.1]
2 <10 ms <10 ms <10 ms whq4op3-rtr-723-f0-0.us.oracle.com
[144.25.252.23]
3 220 ms 210 ms 231 ms usmail05.us.oracle.com [144.25.88.200]

Trace complete.

2.使用ping 来每个设备进行测试。如：

ping -l 1472 -n 1 -f 144.25.216.1
Pinging 144.25.216.1 with 1472 bytes of data:
Reply from 144.25.216.1: bytes=1472 time<10ms TTL=255
ping -l 1472 -n 1 -f 144.25.252.23
Pinging 144.25.252.23 with 1472 bytes of data:
Reply from 144.25.252.23: bytes=1472 time=10ms TTL=254
ping -l 1472 -n 1 -f 144.25.88.200
Pinging 144.25.88.200 with 1472 bytes of data:
Reply from 144.25.88.200: bytes=1472 time=271ms TTL=253

§25.4.3.2 分析瓶颈信息

1.确定是否是NET8的问题：

可以通过在SQLNET.ORA参数文件加相应参数来对服务器端、客户端进行跟踪。

- 跟踪服务器端：在SQLNET.ORA加：

TRACE_LEVEL_SERVER=16
TRACE_UNIQUE_SERVER=ON

- 跟踪客户端，在SQLNET.ORA加：

TRACE_LEVEL_CLIENT=16
TRACE_UNIQUE_CLIENT=ON

- 在 Listener.ora 文件中加：

TRACE_LEVEL_listener_name = 16

这里的16 是跟踪的级别。

2.确定是客户端或服务器端：

- 在客户端收到 ORA-03113，则问题在服务器端；
- 在服务器和监听器收到ORA-03113 ，则问题在服务器端；
- 如果客户端、服务器和监听器收到ORA-03113 ，则问题客户端。

3 .检查服务器是否配置为MTS：
ps –ef |grep ora_d
要看到类似 ora_d000, ora_d001之类的提示，表示有调度进程。

Ps –ef | grep ora_s

要看有共享进程。

§25.4.3.3 调整阵列接口

可以通过调整阵列接口来改善性能，见“Oracle调用接口程序员指南”

§25.4.3.4 调整数据单元缓冲区大小

可以通过NET Assistant 来调整 SDU(SESSION DATA UNIT)的大小，以加快检索速度。详细见“NET8管理员指南”。

§25.4.3.5 在 protocol.ora 中加 TCP.NODELAY

可以在protocol.ora参数文件中加TCP.NODELAY 参数。此参数与平台有关。

§25.4.3.6 使用连接管理器

可以用连接管理器来实现多路复用。见“NET8管理员指南”。

第 27 章　数据库关闭/启动工作

当在 ORACLE 关闭/启动期间，由于没有用户使用数据库，所以 DBA 可以在关机/启动的周期内执行以下各个操作：

- 当数据库关闭时，删除或归档旧的跟踪文件
- 当数据库关闭时，重新命名报警日志
- 通过 alter database backup controlfile to trace 来产生 Create controlfile 命令
- 把使用的最多的程序包驻留在 SGA 区中
- 创建 Owner_object 位置图，显示某人在数据库的某个地方拥有什么。
- 重新计算访问最多的数据表和索引的统计资料
- 缩小扩展超过最优的回滚段

§27.1 删除或归档旧的跟踪文件和跟踪日志

在使用 Oracle 实例时，Oracle 把有关实例的诊断信息写进许多跟踪文件和跟踪日志。每一个后台进程产生一个跟踪文件，这个跟踪文件存储在由 init.ora 或 config.ora 文件中的 back_ground_dest 参数标识的目录中。用户创建跟踪文件由 init.ora 或 config.ora 文件中的 USER_DUMP_DEST 参数标识的目录中。

在 UNIX 中，实例产生的跟踪文件大都为 n_nnnn.trc，这些跟踪文件数量在不断的增长，同样，写到 USER_DUMP_DEST 参数标识的目录中的文件也在增长。为了避免这些跟踪文件占用盘空间。应该在数据库关机时对跟踪文件进行备份和删除工作。

如对 listener.log 日志文件进行维护：

- 确定 listener.log 所在位置
- $lsnrctl status
- 先停止 监听进程： $lsnrctl stop
- 重新命名 listener.log 文件(相当于删除)： $mv listener.log old_list.log、
- 重新启动监听进程： $lsnrctl start

§27.2 重新命名警报日志

当使用 Oracle 实例时，Oracle 把实例中诊断信息变化写到警报日志中，警报日志由 init.ora 或 config.ora 文件中的 BACKGROUND_DUMP_DEST 参数标识的目录中。警告日志通常以实例命名，对于名为 PROD 的实例，警告日志可能命名为 alter_prod.log。警告日志主要包括：启动，关闭，重写日志转换，表空间变化，文件变化，内部错误及表空间备份状况等。

由于警告日志文件不断增长，可能增长到不可管理和不可读的程度。但可采用定期重新

命名来解决。警告日志可以在任何时侯进行命名（在打开也可以），但建议定期在数据库关闭/启动周期内进行重新命名为好。如：

$mv alert_prod.log old_alert.log

现在你可以将 old_alert.log 的有关部分打印出进行分析。

§27.3 产生创建控制文件命令*

控制文件的创建可以用下面命令完成，控制文件的管理详细另见其它章节。

ALTER DATABASE BACKUP CONTROLFILE TO TRACE;

§27.4 驻留程序包

当使用 PL/SQL 对象时，oracle 将其存储在 SGA 区的共享 SQL 区的库缓存内。如果用户已经把程序加载到内存，那么其它用户在执行这个程序时就感到快多了。因此将程序驻留在内存将减少在执行过程中用户的响应时间。

要想将大的 PL/SQL 对象驻留在内存中，就应该在数据块刚一开始(系统一启动后)将它们加载到 SGA 中。使用 DBMS_SHARED_POOL 程序包可以将 PL/SQL 对象驻留在内存中。步骤如下：

- 对 PL/SQL 程序包重新编译；
- 使用 DBMS_SHARED_POOL.KEEP 来将程序驻留在内存中。如：

sql>alter package APPOWNER.ADD_CLIENT complie;

- sql>execute DBMS_SHARED_POOL.KEEP('APPOWNER.ADD_CLIENT','P');

这里'P'为 程序包，如果是'C' 则表示指示器。

为了通过高速缓存管理标准的 Least recently used 算法把装驻留的对象从 SGA 中移走，使用 DBMS_SHARED_POOL.UNKEEP 程序来完成，如：

$sql>execute DBMS_SHARED_POOL.UNKEEP('APPOWNER.ADD_CLIENT');

选择要驻留的程序包

每个数据库有两个程序包：１）数据库核心用的程序包（sys 拥有）；２）特殊应用程序的专用程序包。

下面给出ＳＹＳ拥有的程序包，要驻留的程序包不同包列表如下：

普通的应用程序	高级复制附加的程序包

DIUTIL	DBMS_DEFER
STANDARD	DBMS_REPUTIL
DIANA	DBMS_SNAPSHOT
DBMS_SYS_SQL	DBMS_REFRESH
DBMS_SQL	DBMS_DEFER_INTERNAL_SYS
DBMS_UTILITY	REP$WHAT_AM_I
DBMS_DESCRIBE	
DBMS_JOB	
DBMS_STANDARD	
DBMS_OUTPUT	
PIDL	

应首先驻留最大的程序包，为了确定次序，可从 DBA_OBJECT_SIZE 字典中查出各个程序包的大小：

```
select  owner, name, type
source_size + code_size + parsed_size + error_size  total_bytes
from dba_object_size
where  type='PACKAGE_BODY' order by  total_bytes desc;
```

§27.5 创建拥有者-对象的位图

在恢复，维护和调整的操作中，需要 DBA 知道用户的对象及其所在的表空间是很有用的。如：

```
breae on Tablespace_name skip 1 on Owner
select tablespace_name,owner,segment_name,segment_type
from  dba_segments
order by tablespace_name,owner,segment_name;
```

```
tablespace_name   owner      segment_name     segment_type
----------------- ----------- ------------------- ----------------------
APP_DATA      APPOWNER         LINE_ITEMS       TABLE
                  ORDERS             TABLE
                  SHOSHEE           CONTACTS          TABLE
APP_INDEXES  APPOWNER       LINE_ITEMS_PK    INDEX
                  ORDERS_PK        INDEX
                  ORDERS-TEMP    TABLE
```

从显示来看，可以了解拥有者与对象的位置图，从而在恢复等过程中知道哪个表放在哪个表空间上，比如丢了一个索引，也可以知道重新建立时应放在哪个表空间上。另一方面可以知道应用存放位置的不规范性。如
APPOWNER 用户有一个对象 ORDERS_TEMP 表放在索引表空间中，这样在数据输入不能做到完全输入。

§27.6 重新计算统计资料

为了给基于优化提供基本的资料，应该用 analyze 对表和索引进行分析，如：

```
analyze table Orders compute statistics; /* 需较大的临时段 */
```

为了简化对大量对象的分析，可以使用 DBMS_UTILITY 程序包的 ANALYZE-SCHEMA 程序来完成。该程序有两个输入参数：拥有者 和 分析方法，如：

```
SQL>execute DBMS_UTILITY.ANALYZE_SCHEMA('APPOWNER','COMPUTE');
```

§27.7 缩小扩展超过最佳值的回滚段*

可以使用 alter rollback segment 命令的 shrink 选项来缩小回滚段，如果未指出要缩小回多少字节，则它将缩回到 optimal 所指示的大小。

```
Alter rollback segment r1 shrink to 15m; /* 缩小到 15mb */
Alter rollback segment r1 shrink; /* 缩小回到 optimal 所指示的大小 */
```

第 48 章 调整实例恢复性能*(tuning 24)

§48.1 理解实例恢复*

§48.2 调整实例延迟及碰撞恢复*

§48.3 监示实例恢复*

§48.4 实例恢复阶段调整*

第 49 章　应用程序性能调整*

在 Oracle 应用中，除了影响性能的大都是系统的配置参数等原因外，影响应用系统性能的另一个原因是应用程序的编写不当所致。因而，调整应用程序也是改善性能的一个方面。

§49.1 在基表上创建索引

为基表建立索引是提高查询速度的方法之一。为了提高基表的某个查询的性能时，一般可以选择在该表上建立一个合适的索引。关于建立索引的介绍见《Oracle8i/9I 数据库基础》。

§49.1.1 何时创建 B*树索引

- 经常在 where 后首先用 column_name=xx;

select * from oreders where order#= <order_no>;

就要为 order# 字段建立索引。

- 经常在两个以上表进行连接条件的，如：

```
select a.field1, b.field2, … from tab1 a, tab2 b
where a.file_x=b.field_x;
```

就要为 两个表的 fieldx 字段建立索引。

- 在几个表中选择合适的字段作为主键和外部键。

§49.1.2 何时创建位图树索引

位图索引是在 Oracle7.3 之后才提供的技术。主要用于数据仓库中。可以在下面情况建立位图索引：

- 一个列或几个列经常出现在 where 子句中。
- 相对于表中的记录数而言，列的值是可数的（没有多少中可能的值）。
- 经常用于多个表连接的列。
- 不常更新的列。

§49.1.3 何时创建逆关键字索引

有时应用需要对最新的一组记录进行操作比对旧的记录进行操作更频繁。也就是说，用户经常操作的记录中，对最近的记录操作次数更多。这样的应用应该建立逆关键字索引。
建议在下面情况建立逆关键字索引：

- 被索引的列顺序是逆序递减的数值。
- 表中存在并发 INSERT 和 DELETE。
- 应用程序用于 Oracle 并行服务器环境。
- 查询中很少用索引范围，如在。。。之间等。

§49.1.4 何时不要建索引

建立索引是为了加快查询，如果所选择的索引列不合适，在可能带来负面影响。下面情况谨慎考虑：

- 索引列的宽度过宽；
- 在应用中根本就很少用作为查询条件；
- 该列如果成为索引可能消耗很大的空间。

§49.2 在 select 语句上使用索引

建立索引的目的除了主键和外部键是为了保证数据库记录的完整性和一致性外。更主要的目的是为了加快数据库表的查询。如果编程人员没有了解这一点，不在 select 语句的 where 后用到索引的列。那么就等于没有使用索引。

§49.2.1 评估索引的使用情况

如何知道程序没有引用索引呢？，直接的方法是判断在 select 语句后是否在 where 子句后引用索引列的名字。另外一个评估方法是查询数据字典。我们可以从动态视图 V$SYSSTAT 来查询到相应的数据。一般可以用下面语句进行直接的查询：

```
SQL> select * from v$sysstat where name like 'table%';

STATISTIC# NAME                                 CLASS     VALUE
---------- ---------------------------------- ---------- ----------
       150 table scans (short tables)             64        40
       151 table scans (long tables)              64        41
       152 table scans (rowid ranges)             64         0
       153 table scans (cache partitions)         64         0
       154 table scans (direct read)              64         0
       155 table scan rows gotten                 64    163862
       156 table scan blocks gotten               64      1469
       157 table fetch by rowid                   64     10545
       158 table fetch continued row              64       226

已选择 9 行。
```

而索引的使用率可以用公式：
index_use_ratio=table fetch by rowid/(table fetch by rowid+table scan rows getten)

如果结果大于 80%以上，则索引的使用率还可以，如果结果很低，则说明索引很少被 select 语句引用。

§49.2.2 避免不使用索引的 SQL 语句

并不是说只要在 where 子句后加上索引列的名字就能使用索引了。下面情况就不使用索引：

- 在 索引列上加了函数，如：

```
select name,address  from  persons where  upper(name)='JOHE';
```

- 在 where 子句中用了 NOT 的 SQL 语句，如：

```
select name, address,city from perons
where  city not in('BOSTON','NEWYORK');
```

- 使用 IS NULL 或 IS NOT NULL 的 SQL 语句。
- 对索引列进行内部转换的 SQL 语句。

§49.3 把具有大量数据 I/O 的处理放在服务器上

影响性能的另外因素可能是网络的通信问题，比如商场的日报，月报、银行的结息等都需要对表进行大量的 I/O。这类应用一般不需要实时的交互，一般只是发出执行命令后就等带结果就可以了。这样的应用如果在客户端来操作则需要客户机有大的内存，还需要网络的快速才行。这样的应用可以考虑使用 PL/SQL 工具来编写。并将程序放在服务上。

§49.4 在内存中保持 PL/SQL 代码

将 PL/SQL 包和触发器或 Oracle 序列（Sequences）在内存中永久缓存可以改善系统的的性能，即将那些频繁使用的 PL/SQL 包驻留在共享池(Shared Pool)中使它们被多个用户使用，以提高命中率直到实例关闭为止。完成这样的操作是 oracle 的著名的围巾（pinning ）。它由 Oracle 支持的 DBMS_SHARED_POOL 包来完成。围巾包是由在 INITsid.ora 文件中的 SHARED_POOL_RESERVED_SIZE 设置而被存储。

1. 建立 DBMS_SHARED_POOL

DBMS_SGARED_POOL 包在运行 CATPROC.SQL 建立数据库时并不建立。因而需要运行 DBMSPOOL.SQL 脚本才能建立。该脚本在 UNIX 操作系统下是在 $ORACLE_HOME/rdbms/admin 目录下；在 NT 下是在 %ORACLE_HOME%\rdbms\admin 目录下。用下面命令来建立：

```
SQL>connect sys/password
SQL>@ORACLE_HOME/rdms/admin/dbmspool.sql
```

2. 使用 DBMS_SHARED_POOL

DBMS_SHARED_POOL 包包括两个存储过程：KEEP 和 UNKEEP 包。她们分别用于在内存中**驻留（pin）**或**不驻留(unpin)**一个包、触发器或序列。

下面例子表示把 APPROVE_PO 包驻留在内存：

SQL>EXECUTE DBMS_SHARED_POOL.KEEP('APPROVE_PO');

3.识别所被驻留在内存的对象

可以查询 V$DB_OBJECT_CACHE 字典的内容得到某个对象是否被驻留在内存中：

SQL>select owner , name , type from v$db_object_cache
Where kept='YES';

§49.5 鼓励代码重用

要想 SQL 和 PL/SQL 语句达到命中率高，就要养成代码编写的规范。只有这样才能达到代码重用。请看下面例子：

SQL>select last_name, first_name from app.customer where customer_id='2204';

SQL>select last_name, first_name from app.customer where customer_id='2203';

两个语句分别查询 顾客代码为 2204 和 2203 ，除此以外，语句的其它部分是相同的，如果我们将这个使用频率很高的语句改为：

SQL>select last_name, first_name from app.customer where customer_id= :customer_id;

这样就可以达到代码重用的目的。

§49.6 将表缓存在内存中

对于在应用中被频繁访问的表，应该将它们设置成缓存方式，这样在数据库启动完成后被说明为缓存的表的数据就被系统读到 SGA 区中了。我们可以用 Alter table 或 Create table 命令上加 cache 子句来使表成为**缓存表**(一直放在 SGA 区内不被清除)。如：

```
alter table ACCOUNT_CODE cache;
```

如果不需要缓存而取消缓存设置可用 nocache 来完成。如：

```
alter ACCTOUNT_CODE nocache;
```

例：在建表时将表设置成缓存的方式：

```
SQL> CREATE table phone_lst
(employee_Id number,
phone_no varchar2(15),
extension varchar2(5)
) tablespace users
storage(INITIAL 1m next 500k pctincrease 0 )
CACHE;
```

对于已经将表设置成 Cache 的信息，可以从 DBA_TABLES 字典中查到，如：

```
SQL>select owner, table_name from dba_tables
Where LTRIM(cache) ='Y';
```

§49.7 了解 SQL 语句的速度差别

在编写应用时，应该了解哪些语句尽量少用，那些语句速度快。下面是各种语句的速度顺序：

ORACLE 的访问路径排序如下（快至慢 ）：

顺序	访问路径
1	通过 rowid 单行访问
2	通过 cluster 连接的单行访问
3	通过 散列键或主键的单行访问
4	通过主键字的单行访问

5	cluster 连接
6	散列簇键
7	索引簇键
8	复合键
9	单列索引
10	在索引列上的有界搜索
11	在索引列上的无界搜索
12	排序（order by ）,合并（union）连接
13	索引列的最大(max)到最小(min)
14	通过索引列排序
15	全表扫描

我们了解 SQL 语句这些特性后，就应该少用 order by 等之类的语句。

§49.8 在表结构设计和 select 语句中用 decode

在 Oracle 系统里，提供了功能特别的 decode 函数。使用它可以在性能上得到特别的效果。这样的使用前提的是，先在建立表结构时就需要将该列描述为较短的宽度。然后在 SQL 语句中来使用它。如：

例：银行的活期存款表结构，可以对借贷字段（存入或取出）：

```
create table per_detail
(
hf_seq       char(11) NOT NULL ,  -- 序号,格式为:NNNNNNNNNC--注1
bank_code    varchar2(6) NOT NULL,  -- 经办行代码
Acc_no       varchar2(15) NOT NULL,  -- 公积金代号
emp_acc_no  varchar2(20) NOT NULL,  -- 职工帐号
Tran_date   date not null,          -- 交易日期 , 汇、补缴、支取日期
Tran_val    number(13,2) ,  --  交易金额
tran_code   char(1) ,      -- 交易代码(参照 tran_cont 表)
balance     number(13,2),  -- 余额
db_cr_flag  char(1) ,    -- 借贷类型   0—贷 ,1—借
crea_date       date not null, -- 起息日
start_month     varchar2(7), -- 本次汇补缴起始年月,yyyymm
end_month       varchar2(7), -- 本次汇补缴终止年月,yyyymm
status          char(1),     -- 状态
cash_check      char(1),     --现金,支票
oper_no         varchar2(10),   -- 操作员号
sys_date            date
)pctfree 5 pctused 40
```

```
tablespace hf_data
storage ( initial 50m next 5m maxextents 121 pctincrease 0 )
/
```

上面表结构中，对于 db_cr_flag char(1) , 是借贷类型代码。 0－贷（取出） ,1－借(存入) 。可以在计算个人的余额时使用 decode ， 如：

```
select emp_acc_no ,
sum(decode(to_number(db_cr_flag),0,-1,to_number(db_cr_flag) )*tran_val)
from per_detail group by emp_acc_no;
```

§49.9 将 LOB 类型单独存放

Oracle 支持在表结构中描述 LOB 类型数据，这并不是说明希望把 LOB 类型与其它的类型放在一起。相反，为了性能，建议将 LOB 类型存在另外的表中。比如某企业有 10000 名职工，建立职工信息表时需要将彩色照片存到数据库中。可能按照习惯我们可能就在建立职工基本信息表时，顺便就建立了一个 long raw 字段以存照片。这样设计是非常不好的，原因是 long raw 类型是一个没有明确宽度的字段。当我们在查询职工的一般信息时，就会感到非常的慢。所以我们可以将该 long raw 存放在另一个表中。在查询中，当明确要阅读某个职工的信息时，在点击该记录，然后使用主键外部键来读出相关的 long raw 记录。所有职工信息表可以设计为：

例：建立带有照片的职工信息表：

```
CREATE TABLE PERSONS(
Per_id  varchar2(20) primary  key,-- 职工身份证号
Name  varchar2(20),
Birdate  date,
Address  varchar2(40),
Tel_no  varchar2(16),
…
) tablespace users
storage(initial 2m next 1m, pctincrease 0 );

CREATE TABLE PERSON_PICS(
Per_id  varchar2(20) constraint  fk_id References persons(per_id),-- 职工身份证号
Picture long raw
) tablespace users
storage(initial 2m next 1m, pctincrease 0 );
```

§49.10 分而治之

对于那些在处理中需要连接多个表的应用，如果每个表的记录数都相当大时，不要想一次就完成所有的操作。这样会消耗大量的内存及回滚段等。从而影响速度。建议你采用分段处理的方法尽量将处理分为不同阶段来处理。
另外，下面的因素也建议采用分而治之的方法。
.过于集中对某个盘的 I/O,该盘成为瓶颈，可以考虑逻辑结构，物理结构分开。
.需要连接的多个表在一个盘内，但不是以 CLUSTER 构造，可带来性能下降，原因是先找表 1 数据块，然后磁盘旋转再找表 2 的数块，查找操作占时间。改进方法：经常连接的表放在不同的盘上，RAID 盘类似：连接的表存放于同一卷上也有瓶颈问题，建议将连接的表村放于不同的 RAID 卷上。

第 50 章　内存和 CPU 的优化

主要考虑命中率，命中率是指直接从内存中取得数据而不从硬盘中取得数据的比率。

由于 ORACLE 中 I/O 处理和 CPU 处理是相互联系的，所以内存和 CPU 的调整一起考虑。

§50.1 应用类型

从内存管理角度看，ORACLE 数据库应用可以分为两种：

。OLTP（Online Transaction Processing ）应用，多个用户执行的小型事务。

。批（Batch）应用，少数用户执行大型事务。

§50.1.1 oracle 如何响应 OLTP 数据访问请求

热点（Hot Spot ）：指在磁盘上频繁地访问的一个数据区。如大家都为帐目输入数据，这些数据都通过表的完整性检查，结果表通过完整性建查成为热点。

ORACLE 使用下面三个功能来提高 OLTP 的性能：

1) SGA （系统全局区）：存放着从数据文件中读取的数据块；

2) SGA 共享 SQL 区：存放着被解释的 SQL 命令；

3) 索引可加快对数据的访问。

在数据块缓存中，ORACLE 使用最近很少用(Least Recently Used)来管理空间，当需要从硬盘读入一个新块到缓存中而缓存又无空间时，ORACLE 会将那些最近很少用的块请出来释放出空间。

可以在 Alter table 或 Create table 后加 cache 子句来使表成为**缓存表**(不被请出)。如： alter table ACCOUNT_CODE cache;

取消缓存设置为 nocache ,如：alter ACCTOUNT_CODE nocache;

§50.1.2 oracle 如何响应批数据访问请求

大多数的批数据处理方案就优化调整全表扫描，ORACLE 使用三种技术来实现全表扫描的最优化。

。多块读，在一个读请求里从表中读多个块；

。缓存老化，通过全表扫描读的块很快地从数据块缓存中移除；

。每块有一个一致性获取，ORACLE 在全表扫描过程中检查时间标记（Timestamp），而不检查每一记录的时间标记。

1.多块读

两种方法实现：

1)创建数据库时，声明 DB_BLOCK_SIZE 合适的参数；

2)修改当前的参数文件 INIsid.ora 中的 DB_FILE_MULTIBLOCK_READ_COUNT 参数来实现一次读多块。

HP_UX 操作系统中，一次只允许读 64KB 的数据，如果 DB_BLOCK_SIZE=4KB,则 DB_FILE_MULTIBLOCK_READ_COUNT<=16，

2.缓存老化

当一个用户通过全表扫描正在读取数据时，其他用户要读取该扫描块是不大可能的，因此数据缓存的一部分留给通过全表扫描的块使用（该区的大小为每次读数据的大小）。

3.每块有一个一致性获取（按块进行一致性获取—Consistent gets）

当一行内容被改变时，它的 SCN(system change number)也随之改变，在全表扫描中要检查块的 SCN(不要建行的 SCN) 一提高速度，如果块从开上始到后来都没有改变，则块内行的扫描无需从另外的一致行获得。

§50.2 如何计算命中率

可以用确定在一个时间间隔内逻辑读和物理读的次数计算命中率。数据库记录了自数据库启动开始的逻辑读和物理读。ORACLE 在 /RDBMS/ADMIN/子目录下给出 Utlbstat.sql 和 Utlestat.sql 来确定给定时期内逻辑读和物理读。

逻辑读和物理读的统计可以从动态视图 V$SYSSTAT中查到。逻辑读＝consistent gets + db block gets，当块被更新读（Read for Update）和段头块读访问时， db block gets 统计值在增加。物理读＝physical reads。下面查询显示命中率：

```
SQL> desc v$sysstat
 名称                                      空值?    类型
 ----------------------------------------- -------- ------------
 STATISTIC#                                NUMBER
 NAME                                      VARCHAR2(64)
 CLASS                                     NUMBER
 VALUE                                     NUMBER

select
sum(decode(name,'consistent gets',value,0)) consistent,
sum(decode(name,'db block gets',value,0)) dbblockgets,
sum(decode(name,'physical read',value,0)) physical,
round(((sum(decode(name,'consistent gets',value,0)) +
      sum(decode(name,'db block gets',value,0)) –
      sum(decode(name,'physical reads',value,0))/
       sum(decode(name,'consistent gets',value,0)) +
       sum(decode(name,'db block gets',value,0 ))))
      *100,2 ) hitratio
from v$sysstat;

   CONSISTENT    DBBLOCKGETS    PHYSRDS   HITRATIO
--------------------  --------------------  ---------------  ----------------
```

81538694 40086468 8793798 89.72

分别表示：一致性获取= 81,538,694 次，数据库块获取＝4,008,468 次，
物理读 ＝ 8,538,798 次，
逻辑读 ＝ 一致性获取 + 数据库块获取 = 85,547,158 次
逻辑读 ＝ 一致性获取 + 数据库块获取
= sum(decode(name,'consistent gets',value,0)) +
sum(decode(name,'db block gets',value,0))
命中率 ＝ (逻辑读 - 物理读) / 逻辑读

§50.3 影响命中率的因素

为了了解命中率，必须考虑在数据库块缓存中包含在数据处理中的段。即必须考虑字典表的影响,临时段的影响，回滚段影响，索引扫描的影响以及表扫描的影响。

§50.3.1 字典表活动

当一个 SQL 语句首次到达 Oracle 内核，则它被分析并放在 SGA 中，当该 SQL 语句所需的对象（表或视图）不在 SGA 中时，Oracle 就到数据字典中去查（称递归查询）。将数据块从数据字典中读到 SGA 的数据缓存中。由于数据字典的信息量不大，把它们存放在缓存以提高命中率（被多个用户所用）是个好办法。

注：如果有大块的存储过程代码从数据字典中读取，过多的使用存储过程会明地增加缓存的工作量，因为存储过程使通过数据块缓存被读入共享 SQL 区的。

§50.3.2 临时段的活动

当用户执行一个需要排序的查询时，Oracle 设法对内存中排序区内的所有行进行排序。排序区的大小由 INITsid.ora 文件的 SORT_AREA_SIZE 参数确定。如果内存不够用则 Oracle 分配一块临时段用于存放排序数据。然后通过临时段对数据作写入和写出。并且 Oracle 不使用一致读取机制来读取数据。Oracle 将数据移入和移出临时段执行的是物理读和写（不是逻辑读）。结果有可能出现负的命中率。

例如，临时段有 1 个逻辑读和 100 个物理读，则命中率=(1-100)/1=-99=-9900%。 由于对临时段的访问增加了物理读，而没有逻辑读，所以临时段的活动就人为地降低了 OLTP 应用的命中率。在 OLTP 应用中应避免临时段（临时表空间）的访问。***要确定 SORT_AREA_SIZE 参数的设置要足够大，以避免对临时段的需要。***

§50.3.3 回滚段的活动

回滚段的活动有：回滚活动 和 回滚段头活动。回滚段头是一个数据块，它控制数据块向

回滚段的写。由于回滚段头经常被改变，对于回滚段头块的访问会增加应用的命中率。回滚段头的活动对 OLTP 命中率影响最大。

即要访问数据段又要访问回滚段的查询 比 只访问数据段的查询要慢。

§50.3.4 索引活动

对 数据字典 的访问包括对索引的访问 一取得 ROWID 和 TABLE ACCESS BY ROWID 操作以取得数据块。由于索引和表的操作是独立的，一致获取被用来既在索引中读也从表中读。

如果在查询中不使用唯一索引或只使用部分关键字，则执行的扫描（逻辑读）是索引高度与索引范围内行数的 2 倍。

§50.3.5 表扫描

由于通过表扫描读得的块在数据块缓存中不会保存很长时间，因而表扫描会降低 OLTP 应用和批应用 全部的命中率。

要提高表扫描和排序操作的性能，可以使用 Oracle 的并行选项。并行选项将查询所需的处理分配给多个过程，因此查询可以并发地使用更多的系统资源。

§50.3.6 OLTP 和批应用类型

1. OLTP应用

计算命中率需考虑不同 **数据库段** 的影响。由下表显示不同的数据段对命中率的影响：

表 6-1 OLTP 应用的命中率影响

活动类型	命中率	发生频率
数据字典段	大量增加	高
回滚段	小量增加	低
索引扫描	中等增加	高
临时段	大量减少	低
表扫描	中等减少	低

由表可看出，影响命中率的 临时段访问 和 表扫描 在 OLTP 应用中并不经常发生；而 提高命中率的 数据字典访问 和 索引扫描 在 OLTP 应用中经常发生。

2.批处理

与上面的情形不一样，看下表：

表 6-2 批应用的命中率影响

活动类型	命中率	发生频率
数据字典段	大量增加	低
回滚段	小量增加	变化
索引扫描	中等增加	中

临时段 表扫描	大量减少 中等减少	中 高

由表可看出，影响命中率的 临时段访问 和 表扫描 在**批处理应用中经常**发生；而 提高命中率的 数据字典访问 和 索引扫描 在**批处理应用中经常中不经常**发生。

§50.4 内存和 CPU 的优化调整问题

当只要访问内存中的数据就能完成数据访问请求就称为命中，如果数据不在内存则称请求就称为未命中。

由于 CPU 执行**物理读** 比 **缓存数据直接访问** 要慢 8 倍以上，并且 I/O 操作需要 CPU 资源来管理 SGA 的内容。

正因为物理 I/O 操作是与 CPU 相关的，所以减少 I/O 的操作次数 就 等于减少应用对 CPU 的要求。**减少物理 I/O** 的最好方法是**少用全表扫描**。

§50.5 为应用选择目标命中率

即使对于批应用，也应该努力使命中率超过 89%，应用的命中率依赖于数据库中的 OLTP 用户和批用户的混合，一般来说，可以用动态性能视图 V$SESS_IO 来显示数据库中每个用户的命中率。

可用下列查询来获得批过程的命中率：

```
col hitratio for 999.99
select  username,  consistent_gets , block_gets, physicaL_reads,
100*(consistent_gets + block_gets – physical_reads) /
  ( consistent_gets + block_gets) hiratio
from  v$session, V$sess_io
where  v$session.sid = v$sess_io.sid
and ( consistent_gets + block_gets ) > 0
and username is not null ;
```

用户	一致获取	块获取	物理读	命中率
SYSTEM	298	9	25	91.86
GLORIA	112	4	23	80.17
MARY	190	6	48	75.51

PEACH	145	10	22	85.81
BETTY	2065	128	381	82.63
STACI	47825	40424	4312	95.11
KELLI	532	10	46	91.51
FOR A	112350	25	4711	95.81
POLLY	23891	1061	337	98.65

§50.6 内存和 CPU 的要求

数据库应用使用三个内存区域，即 1）应用的内存区域；2）用户 PGA(会话和排序)；3）SGA 区。

共享 SQL 区比数据块缓存还消耗大的内存。在多线程服务器配置中，共享 SQL 区用于存放特定的会话信息（如文本区域，排序）。另外，当共享 SQL 区存在使用竞争时，多线程服务器可能引起内存繁忙，有时增加共享 SQL 区不会有好结果，而只能增加响应时间。

注：为了避免性能的降低，不要在对存储过程依赖程度很大的应用中使用多线程服务器。

调整共享池(shared_pool)的原则

5) 查询共享池的大小

select value from V$PARAMETER where name='shared_pool_size';

value

30000000

共享池为 30,000,000 字节

如在 NT 环境上：

```
1* select value from v$parameter where name='shared_pool_size'
SQL>/

VALUE
----------
15728640
```

6) 查询 V$SGASTAT 来确定分配给共享池的值

SQL> select name ,bytes from v$sgasta order by name;

或 用下面命令查询：

SQL> select * from v$sgastat order by name ;

POOL	NAME	BYTES
shared pool	DML locks	89760
shared pool	KGFF heap	6552
shared pool	KGK heap	17520
shared pool	KQLS heap	138740
shared pool	PL/SQL DIANA	463972
shared pool	PL/SQL MPCODE	93960
shared pool	PLS non-lib hp	2096
shared pool	State objects	247360
shared pool	VIRTUAL CIRCUITS	275752
shared pool	character set memory	136812
	db_block_buffers	**16777216**
shared pool	db_block_buffers	278528
shared pool	db_block_hash_buckets	98328
shared pool	db_files	370988
shared pool	dictionary cache	379784
shared pool	distributed_transactions-	180152
shared pool	enqueue_resources	129024
shared pool	event statistics per sess	584800
shared pool	fixed allocation callback	320
	fixed_sga	75804
shared pool	free memory	48401048
large pool	free memory	614400
java pool	free memory	15704064
shared pool	joxlod: in ehe	108448
shared pool	joxlod: in phe	114516
shared pool	joxs heap init	4248
shared pool	ktlbk state objects	80036
shared pool	library cache	1426308
	log_buffer	**98304**
java pool	memory in use	5267456
shared pool	message pool freequeue	124552
shared pool	miscellaneous	934308
shared pool	processes	121200
shared pool	sessions	366520
shared pool	sql area	409124
shared pool	table columns	16972
shared pool	transaction_branches	368000
shared pool	transactions	166804
shared pool	trigger defini	1580
shared pool	trigger inform	500
shared pool	trigger source	716

已选择 41 行。

仅管 db_block_buffers 和 log_buffer 不是共享池的一部分，但显示在其中。

空闲内存（Free Memory）是共享池中可以使用的字节数，空闲百分比=（空闲内存/共享池字节数）*100%，如果空闲百分比小于 20%，则应增加大共享池的字节数；否则应该减少共享池的字节数。

字典缓存（shared pool dictionary cache 194804 ）存放数据字典表中的行。
库缓存（shared pool library cache 434264）存放用来确定 SQL 缓存，存储过程和数据字典对象的对象信息。
光标缓存（shared pool sql area 352460 ）反映光标缓存使用空间情况，光标缓存存放着 SQL 语句和 SQL 执行的统计值。

1．如何衡量 SQL 区域缓存的效力
没有简单的办法去检查 SQL 区域缓存的效力，原因是：

- 除非使用第三方的产品（如 SQL Impact,在附加的软盘找），否则无法知道光标缓存和库缓存要支持多少个语句。
- 有些应用动态地建立 SQL 语句，动态地增加共享池中 SQL 语句的个数。如果应用不使用约束变量（Bind variables）SQL 语句的支持数会更加恶化。

可以用查询数据字典来确定使用 SQL 区域缓存的效率：

```
select decode(executions,0,'0',1,'01-02',2,'01-02',
      decode(sign(executions-6),-1,'03-05',
      decode(sign(executions-11),-1,'06-10',
      decode(sign(executions-21),-1,'11-20',
      decode(sign(executions-41),-1,'21-40',41+'))))) executions
 count(*) from v$sqlarea
 where parsing_user_id  != 0 /* execute sys statements  */
 group by  decode(executions,0,'0',1,'01-02',2,'01-02',
      decode(sign(executions-6),-1,'03-05',
      decode(sign(executions-11),-1,'06-10',
      decode(sign(executions-21),-1'11-20',
      decode(sign(executions-41),-1,'21-40',41+'))))；
```

比如得到如下结果：

```
EXEC          COUNT(*)
------------ -----------------
0                          3
01-20                    701
03-05          51
06-10          37
11-29          30
```

```
21-40          41
40+            83
```

从第 2 行中看到，光标缓存中有多达 702 个 SQL 语句只执行了一次或两次。说明共享池定义得太小或者应用没有正确地使用约束变量。如果应用没有正确地使用约束变量，那么应用将会产生许多不同的 SQL 语句，并且每个语句只执行很少的次数。在这种情况下增加共享池不会明显地提高 SQL 区域缓存的效率。

为了确定应用是否正确地使用变量，可从数据字典 V$SQLAREA 中查询 SQL 语句的内容。如查询将显示当前 100 个 SQL 语句，则：

```
select SQL_TEXT from v$SQLAREA
 where  rownum < 101 and SQL_Text not  like '%:%' and Parsing_user_ID !=0;
```

如果发现许多 SQL 语句中包含实际的数据而不是约束变量，则应该对应用进行修改以更好地使用约束变量。

计算 SQL 区域缓存效率的另一方法是 每个语句被执行的次数，下面查询每个语句被执行的次数：

```
    select Min_cached, count(*),  round(avg(executions),2)
    from (
      select decode(Min_cached,
          0,'1) 00-01min',
          1,'1) 01-01min',
          2,'1) 01-01min',
          decode( sign(min_cached-6),-1,'3) 03-05min',
          decode( sign(min_cached-16),-1,'4) 06-15min',
          decode( sign(min_cached-6),-1,'5) 16-30min',
          decode( sign(min_cached-6),-1,'6) 31-60min',
          decode( sign(min_cached-6),-1,'7) 1-2hour',
          8,) 2hr + ' ))) ))) Min_cached,
        Executions
      From (
        Select round( ( sysdate-to_date (first_load_time,
          'yyyy-mm-dd/hh24: mi: ss' )) *24*60)min_cached,
        Executions from V$sqlarea where parsing_user_id != 0
        )
      )
  group by Min_cached;
```

假如输出的结果如下：

```
MIN_CACHED    COUNT(*)   ROUND(AVG(EXECUTIONS),2)
------------------- ---------------- ----------------------------------------
1) 00-01min           16                        1
```

2) 01-02min	128	1.4
3) 03-05min	160	1.11
4) 06-15min	423	2.26
5) 16-30min	39	10.05
6) 31-60min	44	15.48
7) 1-2hour	81	163.09
8) 2hour+	59	238.07

其中：
MIN_CACHED 为缓存时间（分），
COUNT(*) 为分组后的 SQL 语句数量
ROUND(AVG(EXECUTIONS),2) 为每组 SQL 语句在**光标缓存**中执行的平均时间数
由此可见，执行的次数越多，使用光标缓存的效率就越高。如果多数 SQL 语句只在缓存中存在几分钟，则缓存没有被高效地使用。语句在 SQL 缓存区域里的时间越长，执行的将越多。
从上例中，大部分 SQL 语句在共享池里村在的时 间小于 15min,并且每个 SQL 语 句被执行的次数小于 3 次。

可以通过查询动态性能视图来确定内缓存的 SQL 语句的数目，它们使用了多少内存和每个 SQL 语句平均消耗的内存。

```
Select bytes, sql_count ,bytes / sql_count
From ( select count(*) sql_count from v$sqlarea ), V$SGASTAT
Where name='sql area';
```

输出的结果为：

```
BYTES    SQL_COUNT    BYTES/SQL_COUNT
---------- ----------  ------------------------------
12166816       1207          10080.2121
```

结果是有 1207 个 SQL 被缓存，它们共用了共享池 12,166,816 字节。每个语句占用约 10080 字节。要是想增加 2400 个 SQL 语句，则需对共享池再加 12,000,000 字节。

一旦知道了有多少 SQL 语句被缓存，每个 SQL 语句平均空间，就可估算出支持更多 SQL 语句要分配的空间。
下面是查询当前对象在**库缓存**里存放的每种对象的数目：

```
select object_type, dictionary_count, entry_count
trunc(decode(dictionary_count,0,0,
   entry_count/dictionary_count*100),2) pct_cached
from ( select count(*) dictionary_count,object_type from DBA_objects
     group by object_type ) data_dictionary,
   (select count(*) entry_count,type
      from v$db_object_cache  group by type ( libaray_cached
```

where type(+)=object_type;

第五部分　ORACLE 系统高级用法介绍

第 52 章　安全管理+

第 53 章 分布式数据库管理

Oracle8i/9I 企业版可支持分布环境，而一般的标准版不支持分布环境。但是分布选件一般需要

单独购买。如果你的环境具有分布结构并且你也购买了 Oracle 的分布选件，则可以阅读本章的内容来帮助了解一般的分布数据库的管理。

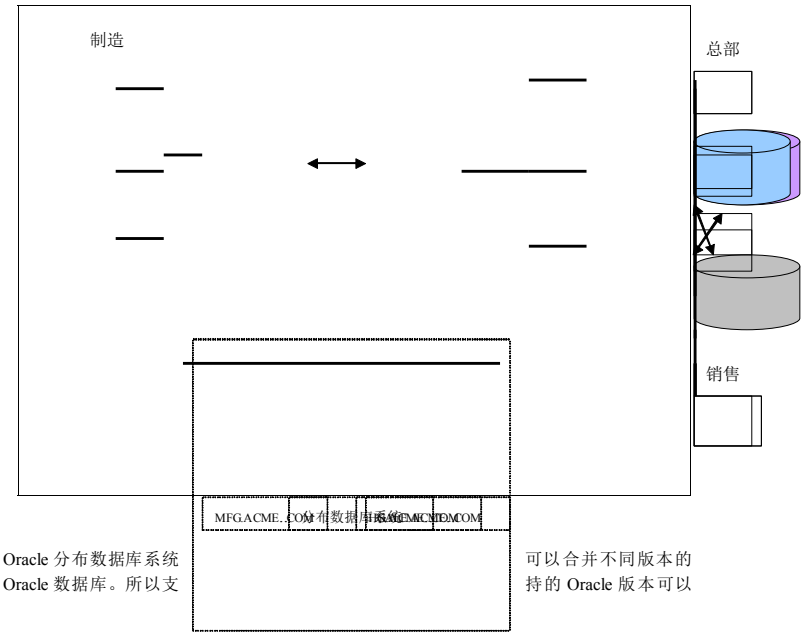
本章参考《Oracle8i Distributed Database System》-- a76960

§53.1 分布式数据库概念

分布式数据库系统(*distributed database system)*是一个单独的自成体系的数据库。它位于多个 Oracle 数据库系统的一个不同的位置上。它可以从本地或远程的数据库来进行访问。在同类分布数据库系统中，每个数据库都是 Oracle 数据库；在异类数据库系统中，至少有一个数据库不是 Oracle 数据库系统。Oracle8i/9i 为分布数据库系统提供直观的图形管理界面。创建分布数据库环境可以使用户的数据存放在不同的地方。使各个服务器的负荷达到平衡；此外也为安全可靠等提供更灵活的空间。

§53.1.1 同质分布数据库

同质数据库系统（*homogenous distributed database system*）是两个或多个 Oracle 数据库的网络，这些数据库驻留在一个或多个机器上。如下图所示：



Oracle 分布数据库系统 可以合并不同版本的 Oracle 数据库。所以支 持的 Oracle 版本可以

分享分布数据库系统。在分布数据库中的应用必须知道至少一个有效的节点。

§53.1.2 异类分布数据库系统

在异类分布数据库系统中至少有一个数据库是非Oracle系统的数据库。对于应用而言，异类分布数据库系统的Oracle数据库是以单个的、本地的方式出现。本地的Oracle数据库服务器隐藏在分布和异类数据中。
Oracle数据库服务器可以通过Oracle8i的异类服务代理连接来访问非Oracle的数据库。如果你使用Oracle透明网关来访问非Oracle数据库，那么，该代理就是一个系统明细应用。例如可以在一Oracle分布系统中包括一个Sybase 数据库。然后得到一个Sybase 明细透明网关来使系统中的Oracle数据库可以与其进行通信。

§53.1.3 客户/服务器数据库结构

Oracle数据库服务器是管理一个数据库的Oracle软件，并且客户是从服务器端请求信息的一个应用。网络中的每个计算机都是一个节点。在分布数据库中的每个节点都可以扮演一个客户端或服务端或两者均有。

§53.2 数据库连接

数据库连接（database link）是一个指针。它定义从一个 Oracle 数据库服务器到另外一个数据库服务器的唯一通信路径。在数据字典表中，该连接指针实际是以一个入口来定义的。要访问一个指针就要连接到包含有数据字典入口的本地数据库中。

§53.2.1 为什么使用数据库连接

使用数据库连接的最大好处就是可以允许用户在所允许的权限内从远程能访问到另外的用户的对象。换句话说，本地用户可以访问一个连接来访问一个远程的数据库而不需指出用户是一个远程的数据库。

§53.2.2 数据库连接中的全局名称

在分布数据库中每个数据库都是以全局数据库名字来进行唯一的标识。Oracle窗口中的全局数据库名是在创建数据库时通过DB_DOMAIN初始参数命名的，单独的数据库名以DB_NAME参数来指定。

§53.2.3 数据库连接中的命名

具有代表性是，一个数据库连接与远程数据库的全局数据库名有相同的名字。例如，一个数据库的全局数据库名是 SALES.US.ORACLE.COM，那么数据库连接（的名字）也是 SALES.US.ORACLE.COM。
当你设置初始化参数 GLOBAL_NAMES = TRUE 时。Oracle 就确保数据库连接名与远程数据库的全局数据库名字有相同的名字。例如，HQ 的全局数据库名字是 HQ.ACME.COM，并且 GLOBAL_NAMES = TRUE ，那么连接名字必须是 HQ.ACME.COM 。
注：Oracle 系统检查全局数据库名字的范围部分（domain part）是存放在数据字典中，而不是在 DB_DOMAIN 中设置。
如果你设置GLOBAL_NAMES = FALSE，则不能使用全局命名。

§53.2.4 数据库连接中的类型

有三种类型的数据库连接：私有的、公共的和全局的。

私有的(Private)：用户创建的带有模式的连接就是私有的连接， 通过DBA_DB_LINKS或ALL_DB_LINKS来访问。

公共的（Public）：用户调用 PUBLIC来创建的连接就是公共连接。

全局的(Global)：当使用Oracle Names，管理员可以管理的全局数据库连接就是全局数据库连接。

SQL语句	连接到数据库	以…连接	连接类型
CREATE DATABASE LINK sales.us.americas.acme_ auto.com USING 'sales_us';	SALES 使用 net 服务名称 SALES_US	连接用户	私有连接用户
CREATE DATABASE LINK foo CONNECT TO CURRENT_USER USING 'am_sls';	SALES 使用 服务名称 AM_SLS	当前全局私有用户	当前用户
CREATE DATABASE LINK sales.us.americas.acme_ auto.com CONNECT TO scott IDENTIFIED BY tiger USING 'sales_us';	SALES 使用 net 服务名称 SALES_US	SCOTT 使用口令 TIGER	私有固定用户
CREATE PUBLIC DATABASE LINK sales CONNECT TO scott IDENTIFIED BY tiger USING 'rev';	SALES 使用 net 服务名称 REV	SCOTT 使用口令 TIGER	公共固定用户
CREATE SHARED PUBLIC DATABASE LINK sales.us.americas.acme_ auto.com CONNECT TO scott IDENTIFIED BY tiger AUTHENTICATED BY anupam IDENTIFIED BY bhide USING 'sales';	SALES 使用 net 服务名称 SALES	SCOTT 使用口令 TIGER, 以 ANUPAM 使用 BHIDE	共享公共固定用户

§53.3 分布数据库管理

下面给出分布数据库管理的简单描述。

§53.3.1 站点自治

站点自治（*Site autonomy*）意思是在分布环境中每个参与服务器都是可从所有的数据库中来独立的管理。虽然几个数据库可以在一起工作，但是每个数据库都是一个分开数据知识库，这些知识库是可以分开管理的。站点自治的优点包括：

- 系统的每个节点可以镜像为一个逻辑的公司或集团的组织。而这些组织是独立管理的；
- 本地管理员控制相应的本地数据，因此每个数据库管理员的知识库范围较小和可管理；
- 独立的失败对于其他分布的数据库的站点来说损失更小。单个数据库停机其它的可以继续运行或减少瓶颈；
- 管理员可以从单独的系统恢复失败；
- 一个数据字典作为本地数据库而存在-全局目录对于本地来说必要；
- 节点可以独立地更新软件。

虽然Oracle允许你在分布数据库系统下来管理每个数据库，但是你不能忽略系统全局的需要，如：

- 在每个接点上创建另外的用户以支持服务器到服务器的连接；
- 设置附加的初始化参数，如：DISTRIBUTED_LOCK_TIMEOUT, DISTRIBUTED_TRANSACTIONS, COMMIT_POINT_STRENGTH等。

§53.3.2 分布数据库安全

在分布环境下，Oracle支持所有在非分布的安全功能。包括：

- 用户和角色口令认证；
- 用户和角色的外部认证的某些类型，包含：

– Kerberos 版本 5 用户连接；
– DCE 用户连接

- 客户端到服务器端和服务器到服务器的登录信息包加密。

下面是分布环境下考虑的一些标题：

通过数据库连接认证

数据库连接可以是 私有的（*private*）或公共的（*public*）, 认证的（*authenticatedI*）或非认证的（*non-authenticate*d）的任何一种。如：

```
CREATE PUBLIC DATABASE LINK foo USING 'sales';

CREATE DATABASE LINK sales CONNECT TO scott IDENTIFIED BY tiger USING 'sales';

CREATE SHARED PUBLIC DATABASE LINK sales CONNECT TO mick IDENTIFIED BY jagger
AUTHENTICATED BY david IDENTIFIED BY bowie USING 'sales';
```

下表说明用户如何通过连接来访问远程数据库：

连接类型	认证否	安全访问
私有	no	连接到远程数据库时，Oracle用户安全信息（用户名/口令）可从本地会话得到，因此，该连接是一个用户数据库连接。口令在两个数据库之间要同步。
私有	yes	用户名/口令是从定义的连接中得到，而不是从本地会话得到，因此，该连接是一个固定的用户数据库连接。这种配置可允许两个数据库的口令不一样。但本地数据库连接口令必须与远程数据库口令匹配。这个口令存放在本地系统目录的文本中。
公共	No	除了所有用户都可以引用指向远程数据库的指针外，与非认证的私有一样。
公共	yes	本地数据库的所有用户可以访问远程数据库，并且所有用户使用相同的用户名/口令进行连接。而且这些口令以明文存放在本地目录中。所以如果你在本地数据库中有足够的权限的话可以看到口令。

§53.3.3 审计数据库连接

对于Oracle来说，总需要执行本地的审计。即，一个用户在本地数据库活动并且通过数据库连接来访问远程数据库，则本地数据库的行动就是进行本地审计；远程数据库的行动就是在远程审计。远程数据库不能决定请求连接是否成功并且后来的SQL语句可以来自另外的数据库。例如：

- 固定用户连接HR.ACME.COM以远程用户SCOTT将本地用户JANE连接到远程的HR数据库。
- SCOTT在远程进行审计。

在远程数据库会话中的行动以SCOTT来审计，如果SCOTT连接到本地的HR话，并且执行同样的行动。这种情况，需要你在远程数据库设置选件以扑捉用户名-在HR数据库中的SCOTT。

在远程对象中你不能设置本地的审计选件。因此，你不能审计数据库连接的使用。虽然访问远程对象可以在远程来审计。

§53.3.4 管理工具

管理分布数据库系统有几个数据库管理工具可选择：

- 企业管理器

- 第三方管理工具
- SNMP支持

企业管理器（Enterprise Manager）
企业管理器是Oracle数据库管理工具。允许你使用图形化来管理数据库任务。你可以用它管理分布数据库：
- 可以管理多个数据库；
- 集中数据库管理任务，可以管理位于世界个地的本地数据库和远程数据库；
- 动态执行SQL、PL/SQL 和企业管理器命令，可以用企业管理器输入、编辑和执行语句；
- 管理安全功能，比如全局用户、角色及企业目录服务等。

第三方工具
多于60种公司的150多种产品可以帮助你管理Oracle数据库和网络。

SNMP 支持
除了网络管理管理能力外，Oracle的SNMP(*Simple Network ManagementProtocol=SNMP*) 支持任何基于SNMP网络管理系统对数据库服务器查找和询问。包括：
- HP的开放视窗（ HP's OpenView）
- Digital的 POLYCENTER Manager on NetView
- IBM的 NetView/6000
- Novell的 NetWare Management System
- SunSoft的 SunNet Manager

§53.4 分布数据库中事务处理

事务是用户执行的一个或多个SQL语句的处理单位，事务由用户发出的SQL语句开并以该用户发出的提交或回滚语句结束。
远程事务只包括一个访问远程节点的语句。一个分布事务包含访问多于一个节点的语句。

§53.4.1 远程 SQL 语句

远程查询语句是一个查询，该查询从驻留在远程节点上的一个或多个远程表中选择数据。例如，下面查询从远程数据库SALES中的SCOTT模式的DEPT上访问数据。

```
SELECT * FROM scott.dept@sales.us.americas.acme_auto.com;
```

远程修改可以修改所有相同节点上的一个或多个表。例如：

```
UPDATE scott.dept@mktng.us.americas.acme_auto.com
SET loc = 'NEW YORK'
WHERE deptno = 10;
```

§53.4.2 分布 SQL 语句

分布SQL语句查询语句从两个或多个节点返回信息。例如，下面查询从本地数据库访问数据又从远程的SALES数据库来访问数据。

```
SELECT ename, dname
FROM scott.emp e, scott.dept@sales.us.americas.acme_auto.com d
WHERE e.deptno = d.deptno;
```

分布更新语句在两个或多个节点来修改数据。分布更新在PL/SQL存储过程或触发器中是可以的。如：

```
BEGIN
UPDATE scott.dept@sales.us.americas.acme_auto.com
SET loc = 'NEW YORK'
WHERE deptno = 10;
UPDATE scott.emp
SET deptno = 11
WHERE deptno = 10;
END;
COMMIT;
```

Oracle在程序中发该语句给远程节点，可能成功地执行，也可能失败。

§53.4.3 远程共享 SQL 语句和分布语句

远程和分布的SQL语句的机制与本地的一样，也是SQL文本必须匹配，并且引用对象也必须匹配有效。共享SQL区可以作为本地和远程处理任何语句或已分解的查询来使用。

§53.4.4 远程事务

远程事务包含一个或多个远程语句，这些语句引用同一个远程节点。例如，下面事务包含两

个语句，每个访问远程SALES数据库：

```
UPDATE scott.dept@sales.us.americas.acme_auto.com
SET loc = 'NEW YORK'
WHERE deptno = 10;
UPDATE scott.emp@sales.us.americas.acme_auto.com
SET deptno = 11
WHERE deptno = 10;
COMMIT;
```

§53.4.5 分布事务

分布事务是一个包含一个或多个独立的语句或一组的事务，在分布数据库的两个或多节点上更新数据。例如，下面事务更新本地数据库和远程SALES数据库：

```
UPDATE scott.dept@sales.us.americas.acme_auto.com
SET loc = 'NEW YORK'
WHERE deptno = 10;
UPDATE scott.emp
SET deptno = 11
WHERE deptno = 10;
COMMIT;
```

§53.4.6 两阶段提交机制

一个数据库必须保证所有事务中的分布或非分布的语句能提交或回滚。正在进行的事务的效果对于其它节点的所有事务来说应该是不可见的。这样的透明对于任何类型的操作，包括查询、更新或远程调用都应该是正确的。
在分布数据库中，Oracle必须网络上使用相同特性来调整事务控制并管理数据的一致性，即使是在网络或系统出现失败时也是一样。
Oracle的两阶段提交机制保证分布事务下无论是提交或回滚所有的数据库服务器分享事务。同时保护隐含的DML操作的完整性、远程过程调用及触发器的执行。

§53.4.7 数据库连接名称的解析

全局数据库名是一个详细说明的对象，全局对象的基本部件有：

- 对象名（Object name）
- 数据库名（dabase name）
- 域名（Domain）

下面语句显示明确的说明：

```
SELECT * FROM scott.emp@orders.us.acme.com;
```

§53.5 分布数据库应用开发

在分布系统中的应用不能在非分布的系统上发行。

§53.5.1 分布数据库系统的透明性

你可以很小的努力，就开发出在分布系统下透明的应用。透明的目标就是使分布数据库系统以单个Oracle数据库面目出现。

位置的透明性

Oracle分布数据库系统有一个特性，它允许应用开发人员和管理员可以隐藏数据库的物理位置。包括：

- 远程数据访问简单，因为数据库用户不需要知道对象的物理位置；
- 管理员可以在最终用户中或现有的数据库应用中使用隐含移动对象。

特别是管理员和开发人员可以使用同义词来为表和支持的对象建立位置的透明。例如：

```
CREATE PUBLIC SYNONYM emp
FOR scott.emp@sales.us.americas.acme_auto.com
CREATE PUBLIC SYNONYM dept
FOR scott.dept@sales.us.americas.acme_auto.com
```

现在的就可以访问远程表：

```
SELECT ename, dname
FROM scott.emp@sales.us.americas.acme_auto.com e,
scott.dept@sales.us.americas.acme_auto.com d
WHERE e.deptno = d.deptno;
```

可以不用用户名和位置的查询：

```
SELECT ename, dname
FROM emp e, dept d
WHERE e.deptno = d.deptno;
```

SQL 和 COMMIT 透明性

Oracle的分布数据库结构也允许提供查询、更新和事务的透明性。例如，标准的SQL命令（SELECT,INSERT,UPDATE,DELETE）可以与非分布一样工作。同样，可以使用标准的命令 COMMIT, SAVEPOINT, 和 ROLLBACK。

- 在单事务中的语句可以引用任何本地或远程数量；
- Oracle保证分布事务的所有节点采取一致的行动：它们要么提交要不就回滚；
- 如果在分布事务提交期间出现网络或系统失败，该事务会自动并且断然地解决，即当网络或系统恢复时，任何一个都提交或回滚该事务。

§53.5.2 分布查询的优化

分布查询优化是Oracle8i的默认的功能。分布查询优化是基于代价的优化或从远程表中产生必要的SQL表达式，可以在远程站点或本地站点来处理数据，并将最终结果送到本地站点上。这样的操作减少需求数据转换的数量。
可以使用不同的基于代价的优化器，如 DRIVING_SITE, NO_MERGE和INDEX等。

§53.6 民族语言的支持

Oracle支持可以在客户端、服务端和非服务器端使用不同的字符集。在Oracle8i，NCHAR支持异类的字符。你可以设置NLS和HS参数来控制不同字符集间的数据转换。

参数	环境	定义
NLS_LANG	客户－服务器	客户端
NLS_LANGUAGE NLS_CHARACTERSET NLS_TERRITORY	客户－服务器 非异类分布 异类分布	Oracle 数据库服务器
HS_LANGUAGE	异类分布	非Oracle 数据库服务器 透明网关
NLS_NCHAR HS_NLS_NCHAR	异类分布	Oracle 数据库服务器 透明网关

§53.7 管理分布系统全局名称

在分布数据库系统中，每个数据库应该有一个唯一的全局数据库名字（global database name.），全局数据库名字在系统中唯一地标识一个数据库。在分布系统中一个主要的管理任务就是建立和改变全局数据库名字。

§53.7.1 理解全局数据库名字版式

全局数据库名字是由两个部分构成：数据库名字和域名。数据库名字和域名由下面初始参数确定。

部件	参数	要求	例子

数据库名字	DB_NAME	少于8个字符	SALES
包含数据库的域名	DB_DOMAIN	由点分开的，域名在前面	US.ACME.COM

下表是有效的一些全局名字：

DB_NAME	DB_DOMAIN	Global Database Name
SALES	AU.ORACLE.COM	SALES.AU.ORACLE.COM
SALES	US.ORACLE.COM	SALES.US.ORACLE.COM
MKTG	US.ORACLE.COM	MKTG.US.ORACLE.COM
PAYROLL	NONPROFIT.ORG	PAYROLL.NONPROFIT.ORG

DB_DOMAIN 初始化参数仅在数据库建立时是很重要的，它与DB_NAME初始化参数一起形成数据库的全局名字。在这一点上，数据库的全局名字存储在数据字典上。你必须使用ALTER DATABASE语句改变数据库的全局名字，而不能在初始化参数中改变DB_DOMAIN参数。然而，一个好的习惯，可以在下次数据库启动前改变DB_DOMAIN 来实现域名的改变。

§53.7.2 确定全局数据库名字执行

本地的连接名与远程数据库全局名字有关，如果远程数据库执行了全局名字，则你必须使用远程数据库的全局数据库名作为连接的名字。例如，你可以连接到HQ服务器和为远程MNFG数据库建立一个连接，这样，MNFG就执行了全局命名，因此你必须使用MNFG的全局数据库名字作为连接名字。
你也可以用服务名作为数据库连接名的一部分。例如，如果你用SN1和SN2 连接到数据库HQ.ACME.COM，则HQ执行了全局命名。你可以为HQ建立下面连接：

```
HQ.ACME.COM@SN1
HQ.ACME.COM@SN2
```

确定一个数据库中的全局命名是否被执行，检查数据库的初始化文件或者查询V$PARAMETER都可以得到结果，如：

```
COL name FORMAT a12
COL value FORMAT a6
SELECT name, value FROM v$parameter
WHERE name = 'global_names'
/


SQL> @globalnames
NAME VALUE
------------ ------
global_names FALSE
```

§53.7.3 浏览全局数据库名

可以查询GLOBAL_NAME 视图来查看数据库的全局名字，例如：

```
SELECT * FROM global_name;
GLOBAL_NAME
--------------------------------------------------------------------------
SALES.AU.ORACLE.COM
```

§53.7.4 在全局数据库名中改变域名

使用ALTER DATABASE 语句改变一个数据库的全局名的域名。
注意，在数据库创建完成后，改变初始化DB_DOMAIN或改变数据库连接名是没有效果的。

下面例子说明重新命名的语句，这里，*database*是一个数据库名，而*domain* 是网络域名：

```
ALTER DATABASE RENAME GLOBAL_NAME database.domain;
```

在全局数据库名中改变域名：

1. 确定当前全局数据库的名字：

```
SELECT * FROM global_name;
GLOBAL_NAME
--------------------------------------------------------------------------
SALES.AU.ORACLE.COM
```

2. 使用ALTER DATABASE 语句重新命名数据库名字：

```
ALTER DATABASE RENAME GLOBAL_NAME sales.us.oracle.com;
```

3. 查询 GLOBAL_NAME 表来检查新名字：

```
SELECT * FROM global_name;
GLOBAL_NAME
------------------------------------------------------------
SALES.US.ORACLE.COM
```

§53.7.5 改变全局数据库名

在下面的假定中，你可以改变本地数据库的全局数据库名字的一部分，也可以使用部分指定全局名建立数据库连接以测试Oracle如何解决名字问题。你回发现Oracle使用本地数据库的全局数据库名字的网络域名部分解决了部分名字，而不是初始化参数DB_DOMAIN的值。

1. 连接到 SALES.US.ACME.COM ， 查询 GLOBAL_NAME 数据字典确定当前全局名字：

```
CONNECT sys/sys_pwd@sales.us.acme.com
SELECT * FROM global_name;
GLOBAL_NAME
------------------------------------------------------------
SALES.US.ACME.COM
```

2. 查询 V$PARAMETER 确定当前DB_DOMAIN 初始化参数的设置：

```
SELECT name, value FROM v$parameter WHERE name = 'db_domain';
NAME VALUE
--------- ----------
db_domain US.ACME.COM
```

3. 为数据库HQ建立数据库连接，使用部分指定：

```
CREATE DATABASE LINK hq USING 'sales';
```

4. 查询 USER_DB_LINKS 以确定哪个域名用于指定全局名字：

```
SELECT db_link FROM user_db_links;

DB_LINK
-----------------
HQ.US.ACME.COM
```

这样的结果表明，本地书记的全局数据库名字的域名部分是US.ACME.COM，Oracle在解决部分数据库连接名字中（当数据库连接被建立后）使用这个名字。

5. 假定你收到 SALES 数据库已经移到了日本，所以你必须命名 SALES 数据库为 SALES.JP.ACME.COM：

```
ALTER DATABASE RENAME GLOBAL_NAME TO sales.jp.acme.com;

SELECT * FROM global_name;
GLOBAL_NAME
------------------------------------------------------------
SALES.JP.ACME.COM
```

6. 你再查询V$PARAMETER会发现：尽管你已经命名了全局数据库的域名，但DB_DOMAIN 的值没有改变：

```
SELECT name, value FROM v$parameter
```

```
WHERE name = 'db_domain';
NAME VALUE
--------- -----------
db_domain US.ACME.COM
```

这样就表明，DB_DOMAIN参数的值是不依赖ALTER DATABASE RENAME GLOBAL_NAME语句的。ALTER DATABASE 语句确定全局数据库名字的域名，但DB_DOMAIN初始化参数不是。

1. 你可以 为SUPPLY数据库连接另外的数据库连接，然后查询USER_DB_LINKS 看到Oracle解决SUPPLY的全局数据库名字的域名的情况：

```
CREATE DATABASE LINK supply USING 'supply';
SELECT db_link FROM user_db_links;
DB_LINK
------------------
HQ.US.ACME.COM
SUPPLY.JP.ACME.COM
```

这个结果表明，Oracle使用域名JP.ACME.COM来处理连接名字的指定。当连接创建是这个域名被使用，因为它是本地数据库的全局数据库的名字的域名部分。Oracle在处理局部连接名字是不使用DB_DOMAIN参数来进行设置。

8. 你收到信息： SALES 将变为ASIA.JP.ACME.COM 域，不是JP.ACME.COM 域。因此要重新命名全局数据库名字：

```
ALTER DATABASE RENAME GLOBAL_NAME TO sales.asia.jp.acme.com;
SELECT * FROM global_name;
GLOBAL_NAME
--------------------------------------------------------------------------
SALES.ASIA.JP.ACME.COM
```

9. 你查询 V$PARAMETER 再检查DB_DOMAIN参数的设置：

```
SELECT name, value FROM v$parameter
WHERE name = 'db_domain';
NAME VALUE
---------- -----------
db_domain US.ACME.COM
```

结果表明，在参数文件中的域名的设置是与发出ALTER DATABASE RENAME语句前是一样的。

10. 最后， 你为WAREHOUSE数据库连接一个连接，然后再查询USER_DB_LINKS确定Oracle是如何处理部分指定关键名字的：

```
CREATE DATABASE LINK warehouse USING 'warehouse';
SELECT db_link FROM user_db_links;
DB_LINK
----------
HQ.US.ACME.COM
SUPPLY.JP.ACME.COM
WAREHOUSE.ASIA.JP.ACME.COM
```

§53.8 建立数据库连接

要想在分布数据库系统中使应用能访问数据和模式对象，就必须建立数据库连接。

§53.8.1 建立数据库连接的权限

在本地数据库中，数据库连接是一个指针。要建立数据库连接，需要必要的权限。
可以查询ROLE_SYS_PRIVS 视图了解你当前的权限。

```
CREATE DATABASE LINK  建立本地的数据库连接
CREATE PUBLIC DATABASE LINK  建立本地的公共的数据库连接
CREATE SESSION  建立任何远程的数据库连接

SELECT DISTINCT privilege AS "Database Link Privileges"
FROM role_sys_privs
WHERE privilege IN ( 'CREATE SESSION','CREATE DATABASE LINK',
'CREATE PUBLIC DATABASE LINK')
/

SQL> @privs
Database Link Privileges
----------------------------------------
CREATE DATABASE LINK
CREATE PUBLIC DATABASE LINK
CREATE SESSION
```

§53.8.2 指定数据库连接类型

在建立数据库连接时，需要确定谁是访问该连接的用户。要关心的有：

- 建立私有的数据库连接
- 建立公共的数据库连接

- 建立全局的数据库连接

建立私有的数据库连接

CREATE DATABASE LINK link_name ...;

例子：

语句	建立
CREATE DATABASE LINK supply.us.acme.com;	这个私有连接使用全局数据库名字SUPPLY.
CREATE DATABASE LINK link_2 CONNECT TO jane IDENTIFIED BY doe USING 'us_supply';	连接名link_2；私有固定用户JANE/DOE；服务名US_SUPPLY
CREATE DATABASE LINK link_1 CONNECT TO CURRENT_USER USING 'us_supply';	私有连接LINK_1：服务名US_SUPPLY；这个连接使用当前用户的用户名/口令 登录到远程数据库中。

建立公共的数据库连接

在语句中加PUBLIC就可以建立公共数据库连接：

CREATE PUBLIC DATABASE LINK link_name ...;

Following are examples of public database links:

SQL语句	建立…
CREATE PUBLIC DATABASE LINK supply.us.acme.com;	为远程数据库SUPPLY建立连接，这个连接 使用用户名/口令可进行连接。
CREATE PUBLIC DATABASE LINK pu_link CONNECT TO CURRENT_USER USING 'supply';	公共的连接PU_LINK；服务名SUPPLY；这个连接 使用 用户名/口令可连接到远程数据库上。
CREATE PUBLIC DATABASE LINK sales.us.acme.com CONNECT TO jane IDENTIFIED BY doe;	远程数据库SALES的公共的固定用户连接，这个连接可以使用JANE/DOE连接到远程数据库上。

§53.8.3 指定连接用户

数据库连接定义一个从一个数据库到另外一个数据库的通信路径。当一个应用使用一个数据库连接访问一个远程数据库时，Oracle使用数据库连接为远程数据库建立一个数据库会话，Oracle在远程数据库建立一个数据库会话。当你建立了私有或公共的数据库连接时，你就可以确定远程数据库的那些模式与固定用户建立连接。

建立固定用户数据库连接

CREATE DATABASE LINK ... CONNECT TO username IDENTIFIED BY password ...;

下面连接用SCOTT/TIGER 可连接到远程数据库上：
CREATE PUBLIC DATABASE LINK
supply.us.acme.com CONNECT
TO scott AS tiger;

下面连接用JANE/DOE 可连接到远程数据库上(服务名为FINANCE)：

CREATE DATABASE LINK foo
CONNECT TO jane IDENTIFIED
BY doe USING 'finance';

§53.9 建立共享数据库连接

每个使用标准数据库连接来引用远程服务器的应用都建立一个本地数据库与远程数据库之间的连接。许多同时运行的应用可引起本地数据库和远程数据库很高的连接数。公享数据库连接可使你限制数据库的连接数。

§53.9.1 建立共享数据库连接

可以在CREATE DATABASE LINK 语句后加 建立SHARED 就实现建立公享数据库连接：

CREATE SHARED DATABASE LINK dblink_name
[CONNECT TO username IDENTIFIED BY password]|[CONNECT TO CURRENT_USER]
AUTHENTICATED BY schema_name IDENTIFIED BY password
[USING 'service_name'];

例：建立固定用户，为SALES数据库建立link2sales：

CREATE SHARED DATABASE LINK link2sales
CONNECT TO scott IDENTIFIED BY tiger
AUTHENTICATED BY keith IDENTIFIED BY richards
USING 'sales';

§53.9.2 管理数据库连接

关闭数据库连接

如果你访问一个数据库连接，则该数据库 保留开发直到关闭该会话为止。

- 如果20个用户打开会话并在本地数据库访问同一个公共连接，则该20个数据库连接是打开的；
- 如果20个用户打开会话并每个用户访问一个私有连接，则打开了20个数据库连接;
- 如果一个用户启动一个会话并访问了20个不同的连接，则20个数据库连接是打开的；当你关闭了一个会话后，与该会话相关的连接自动关闭。你也可以手工关闭某个连接。

```
ALTER SESSION CLOSE DATABASE LINK linkname;
```

删除数据库连接

可以象删除表和视图一样删除连接，只要有 DROP PUBLIC DATABASE LINK 权限。

```
DROP [PUBLIC] DATABASE LINK dblink;
```

删除数据库连接的步骤：

1. 用SQL*Plus登录到Oracle：
CONNECT scott/tiger@local_db

2. 查询USER_DB_LINKS视图：

```
SELECT db_link FROM user_db_links;
DB_LINK
-----------------------------------
SALES.US.ORACLE.COM
MKTG.US.ORACLE.COM
2 rows selected.
```

3. 用DROP DATABASE LINK语句删除不需要的连接：

```
DROP DATABASE LINK sales.us.oracle.com;
```

删除公共数据库连接的步骤：

1. 登录到SQL*PLUS，具有DROP PUBLIC DATABASE LINK 权限即可：

```
CONNECT sys/change_on_install@local_db AS SYSDBA
```

2. 查询 DBA_DB_LINKS：

```
SELECT db_link FROM user_db_links
WHERE owner = 'PUBLIC';

DB_LINK
-----------------------------------
DBL1.US.ORACLE.COM
SALES.US.ORACLE.COM
INST2.US.ORACLE.COM
RMAN2.US.ORACLE.COM
4 rows selected.
```

3. 用DROP PUBLIC DATABASE LINK 语句删除：

```
DROP PUBLIC DATABASE LINK sales.us.oracle.com;
```

限制数据库连接数

可以初始化参数OPEN_LINKS 来显著连接的数目。 但要考虑：

- 该值要大于或等于数据库参考的数目；
- 如果过去有几个分布数据库经常被访问，则增加该值；比如有规律访问3个数据库，则OPEN_LINKS 设置为3 或更大；
- 一般OPEN_LINKS 的默认值是4 ，如果设置为0 则表示非分布方式。

§53.10 浏览数据库连接

可以从下面的视图中查询到有关数据库连接的信息：

```
DBA_DB_LINKS 数据库中所有连接信息
ALL_DB_LINKS 列出可访问的数据库连接信息
USER_DB_LINKS 列出用户的所有数据库连接信息

COL owner FORMAT a10
COL username FORMAT a8 HEADING "USER"
COL db_link FORMAT a30
COL host FORMAT a7 HEADING "SERVICE"
SELECT * FROM dba_db_links
/
SQL>@link_script
OWNER DB_LINK USER SERVICE CREATED
```

---------- -------------------------------- -------- ------- ----------
SYS TARGET.US.ACME.COM SYS inst1 23-JUN-99
PUBLIC DBL1.UK.ACME.COM BLAKE ora51 23-JUN-99
PUBLIC RMAN2.US.ACME.COM inst2 23-JUN-99
PUBLIC DEPT.US.ACME.COM inst2 23-JUN-99
JANE DBL.UK.ACME.COM BLAKE ora51 23-JUN-99
SCOTT EMP.US.ACME.COM SCOTT inst2 23-JUN-99
6 rows selected.

查看口令信息：

```
col userid format a10
col password format a10
SELECT userid,password
FROM sys.link$
WHERE password IS NOT NULL
/
SQL>@linkpwd

USERID PASSWORD
---------- ----------
SYS ORACLE
BLAKE TYGER
SCOTT TIGER
3 rows selected.
```

查看认证口令：

如果有DBA角色，则可以看所有连接的AUTHENTICATED BY ... IDENTIFIED BY ...的用户名和口令；如果你没有DBA角色，但有SELECT ANY TABLE权限，并且O7_DICTIONARY_ACCESSIBILITY 设置为TRUE(默认)。则仍然可以看这些信息：

```
col authusr format a10
col authpwd format a10
SELECT authusr as userid, authpwd as password
FROM sys.link$
WHERE password IS NOT NULL
/
SQL> @authpwd
USERID PASSWORD
---------- ----------
ELLIE MAY
```

1 row selected.

例2：

```
COL owner FORMAT a8
COL db_link FORMAT a15
COL username FORMAT a8 HEADING "CON_USER"
COL password FORMAT a8 HEADING "CON_PWD"
COL authusr FORMAT a8 HEADING "AUTH_USER"
COL authpwd format a8 HEADING "AUTH_PWD"
COL host FORMAT a7 HEADING "SERVICE"
COL created FORMAT a10
SELECT DISTINCT d.owner,d.db_link,d.username,l.password,
l.authusr,l.authpwd,d.host,d.created
FROM dba_db_links d, sys.link$ l
WHERE password IS NOT NULL
AND d.username = l.userid
/
SQL> @user_and_pwd
OWNER DB_LINK CON_USER CON_PWD AUTH_USE AUTH_PWD SERVICE CREATED
-------- ---------- -------- -------- -------- -------- ------- ----------
JANE DBL.ACME.COM BLAKE TYGER ELLIE MAY ora51 23-JUN-99
PUBLIC DBL1.ACME.COM SCOTT TIGER ora51 23-JUN-99
SYS TARGET.ACME.COM SYS ORACLE inst1 23-JUN-99
```

```
COL db_link FORMAT a25
COL owner_id FORMAT 99999 HEADING "OWNID"
COL logged_on FORMAT a5 HEADING "LOGON"
COL heterogeneous FORMAT a5 HEADING "HETER"
COL protocol FORMAT a8
COL open_cursors FORMAT 999 HEADING "OPN_CUR"
COL in_transaction FORMAT a3 HEADING "TXN"
COL update_sent FORMAT a6 HEADING "UPDATE"
COL commit_point_strength FORMAT 99999 HEADING "C_P_S"
SELECT * FROM v$dblink
/
SQL> @dblink

DB_LINK OWNID LOGON HETER PROTOCOL OPN_CUR TXN UPDATE C_P_S
--------------------------- ------ ----- ----- --------- ------- --- ------ ------
```

INST2.ACME.COM 0 YES YES UNKN 0 YES YES 255

§53.11 建立位置透明性

当你配置了必要的数据库连接后，就可以使用各种工具来隐藏数据库系统的分布面目。

```
CREATE VIEW company
AS
SELECT a.empno, a.ename, b.dname
FROM emp a, dept@hq.acme.com b
WHERE a.deptno = b.deptno;
```

```
CREATE PROCEDURE fire_emp (enum NUMBER) AS
BEGIN
DELETE FROM emp WHERE empno = enum;
END;
```

§53.12 分布事务

分布事务是一个包含一个或多个独立的语句或一组的事务，在分布数据库的两个或多节点上更新数据。例如，下面事务更新本地数据库SALES和远程HQ、MAINT数据库：

```
UPDATE scott.dept@hq.us.acme.com
SET loc = 'REDWOOD SHORES'
WHERE deptno = 10;
UPDATE scott.emp
SET deptno = 11
WHERE deptno = 10;
UPDATE scott.bldg@maint.us.acme.com
SET room = 1225
WHERE room = 1163;
COMMIT;
```

§53.13 设置分布事务初始化参数

可以设置初始化参数来选种事务的处理。

DISTRIBUTED_TRANSACTIONS 指定分布事务处理的最大值。

DISTRIBUTED_LOCK_TIMEOUT 指定分布是事务等待的秒数
DISTRIBUTED_RECOVERY_CONNECTION_HOLD_TIME 一个分布事务失败后，Oracle 支持打开远程连接的秒数
COMMIT_POINT_STRENGTH 在分布事务中提交点的值

§53.14 查看分布事务信息

可以从DBA_2PC_PENDING视图中列出怀疑的分布事务的信息，如：

```
COL local_tran_id FORMAT a13
COL global_tran_id FORMAT a30
COL state FORMAT a8
COL mixed FORMAT a3
COL host FORMAT a10
COL commit# FORMAT a10
SELECT local_tran_id, global_tran_id, state, mixed, host, commit#
FROM dba_2pc_pending
/
SQL> @pending_txn_script
LOCAL_TRAN_ID GLOBAL_TRAN_ID STATE MIX HOST COMMIT#
------------- ------------------------------ -------- --- ---------- ----------
1.15.870 HQ.ACME.COM.ef192da4.1.15.870 commit no dlsun183 115499
```

可以从DBA_2PC_NEIGHBORS视图中列出所有的怀疑会话：

```
COL local_tran_id FORMAT a13
COL in_out FORMAT a6
COL database FORMAT a25
COL dbuser_owner FORMAT a15
COL interface FORMAT a3
SELECT local_tran_id, in_out, database, dbuser_owner, interface
FROM dba_2pc_neighbors
/
SQL> CONNECT sys/sys_pwd@hq.acme.com
SQL> @neighbors_script
```

```
LOCAL_TRAN_ID IN_OUT DATABASE DBUSER_OWNER INT
------------- ------ -------------------------- --------------- ---
1.15.870 out SALES.ACME.COM SYS C
```

§53.15 处理怀疑的事务

如果用户的应用提交出现下面信息，则有两阶段提交的问题：

```
ORA-02050: transaction ID rolled back,
some remote dbs may be in-doubt
ORA-02051: transaction ID committed,
some remote dbs may be in-doubt
ORA-02054: transaction ID in-doubt
```

- 当怀疑事务的数据被另外的事务所锁时就出现ORA-01591
- 如果是回滚段ID冲突，则查询DBA_2PC_PENDING和DBA_ROLLBACK_SEGS；
- 两阶段提交在可接受的时间内不能完成；

查看事务通知

```
ALTER SESSION ADVISE COMMIT;
INSERT INTO emp@hq ... ; /*advice to commit at HQ */
ALTER SESSION ADVISE ROLLBACK;
DELETE FROM emp@sales ... ; /*advice to roll back at SALES*/
ALTER SESSION ADVISE NOTHING;
```

§53.16 手工处理处理怀疑的事务

可以使用COMMIT或 ROLLBACK加FORCE选件来使怀疑事务提交，如：

```
COMMIT FORCE ’ transaction_id’;
```

变量 *transaction_id* 是事务的标识。这些事务标识可以从DBA_2PC_PENDING 的LOCAL_TRAN_ID或GLOBAL_TRAN_ID列来查到。

例1，当前的事务ID号是LOCAL_TRAN_ID 1.45.13 ，则可以用下面语句强行提交：

COMMIT FORCE ’1.45.13’;

例2 用SCN号提交，例如希望手工提交全局ID号为

SALES.ACME.COM.55d1c563.1.93.29

则首先要查询远程数据库的DBA_2PC_PENDING 视图的有关事务，然后在COMMIT语句后加SCN号（如果SCN 为829381993）：

COMMIT FORCE ’SALES.ACME.COM.55d1c563.1.93.29’, 829381993;

例3：手工回滚怀疑事务

ROLLBACK FORCE ’transaction_id’;

如果当前的事务ID是2.9.4，则：

ROLLBACK FORCE ’2.9.4’;

第 54 章　复制管理-

本章详细见《Oracle8i Replication》a76959

§54.1 复制概念

复制(Replication)是在分布数据库环境下拷贝一个表的副本的过程。插入、更新、删除存储在本地的应用于分布环境的表。复制对象包括在一台以上机器上存在的表、索引、数据库触发器、包与视图。Oracle 支持复制对象的完全拷贝和复制对象的部分拷贝。

§54.1.1 复制对象、组及站点

Oracle 的复制可以对对象、组及站点进行复制。下面是简单的描述。

复制对象（Replication Objects）

Oracle可以复制下列对象：

- Tables
- Indexes
- Views
- Packages and Package Bodies
- Procedures and Functions
- Triggers
- Sequences
- Synonyms

复制组（Replication Groups）

在复制环境下，Oracle使用复制组（*replication groups*）管理复制对象。复制组是一组相关的复制对象的集合。这些相关的对象可以在一起来管理。

复制站点（Replication Sites）

复制组可以存在于多个复制站点中。复制环境支持两个基本类型的站点：主站点（*master sites*）和快照站点（*snapshot sites*）。在同一时间内可以有一个主站点和一个快照站点。

§54.1.2 复制环境类型

Oracle 支持下面几种复制类型：

多主机复制（Multimaster Replication）

多主机复制也叫对等或多路复制（peer-to-peer or n-way replication)。它允许多个站点同时活动。在每个复制环境的站点也是一个主机站点。在多主机环境下应用可以更新任何站点的任何表。在多主机复制环境中经常采用异步复制。

快照复制（Snapshot Replication）

快照包含一个从单点上的完全或部分的目标表的拷贝。快照可以是只读的或可更新的表。它的优点有：

- 使本地可访问，改善响应时间和可用性
- 从主机节点上卸下查询，因为用户可以查询本地的快照
- 增加数据的安全性。

§54.1.3 复制环境管理工具

Oracle8i可以使用它所提供的"Oracle复制管理（Oracle's Replication Manager）"工具等来对复制进行管理。这个 工具可以以图形的界面为用户提供方便的管理。下面是各个工具的简单描述：

Oracle 复制管理

Oracle复制管理提供的图形界面如下图：

复制管理 API

复制管理API是一组PL/SQL压缩包和函数。你可以用这些工具进行Oracle环境的配置。复制管理API是一些通用的管理复制的命令行。比如你可以用：
DBMS_REPCAT.CREATE_MASTER_REPGROUP 包创建一新的主机组。

复制目录（Replication Catalog）

在复制环境中的每个主机和快照都是复制目录。一个站点中的复制目录是一组不同的数据字典表和视图。这些数据字典表和视图用于维护复制对象和复制组的信息。在复制环境下，每台服务器都使用这些复制目录来 自动对象复制。

分布模式管理（Distributed Schema Management）

在复制环境下，所有的DDL语句必须用DBMS_REPCAT包或复制管理器来发送。当你使用这两重之一时，复制环境的所有的DDL被复制给所有的站点。

§54.2 主机概念与结构

要理解主机复制的详细结构，就要理解主机复制的概念。

§54.2.1 什么是主机复制

Oracle有两种主机复制的窗体：单主机复制和多主机复制。
单主机复制包括一个主机站点，支持一个或多个快照站点。所有的连接快照更新它们的主机的站点，所有的数据冲突检查和解决是主机站点上执行。
一个主机可以是一个多主机复制环境的一个节点，或者是一个或多个快照的主机。

Oracle的多主机复制

Oracle's *multimaster replicatio*n, also known as peer-to-peer or n-way replication, allows multiple sites, acting as equal peers, to manage groups of replicated database objects. Applications can update any replicated table at any site in a multimaster configuration. Figure 2–1 illustrates a multimaster replication system.
Oracle database servers operating as master sites in a multimaster replication environment automatically work to converge the data of all table replicas, and ensure global transaction consistency and data integrity.

Master Sites

A master site can be both a node in a multimaster replication environment and/or the master for one or more snapshot sites in a single master replication environment. The replicated objects are stored at the master site and are available for user access.

Master Definition Site In a multimaster replication environment, a single master site operates as the master definition site for a master group. This particular site performs many of the administrative and maintenance tasks for the multimaster replication environment.
There can be only one master definition site for a master group, though the master definition site can be any of the master sites in the multimaster environment. Additionally, the master definition site can be changed to a different master site if necessary.

See Also: Chapter 3, "Snapshot Concepts & Architecture" for more information about snapshots.

§54.2.1 主群组

§54.2.2 快照组

§54.3 传播的类型

§54.3.1 异步传播

§54.3.2 同步传播

§54.4 复制的类型

§54.4.1 行级复制

§54.4.2 串行传播

§54.4.3 并行传播

§54.4.4 过程化复制

§54.5 冲突解决

§54.5.1 冲突的类型

§54.5.2 避免冲突

§54.5.3 鉴别冲突

§54.5.4 解决冲突

§54.6 快照

第 56 章　Oracle 并行服务器安装与配置

本章讲述并行服务器的下面内容：

- 硬件并行体系结构概述
- Oracle并行体系结构概述
- Oracle 内部实例
- 并行缓存的管理
- Oracle 并行服务器组件
- Oracle 并行服务器存储考虑
- 可伸缩与并行服务器
- 高可用性与并行服务器

§56.1 硬件环境概述

Oracle 提供了可伸缩性体系结构，可以在对称多处理器（SMP= Symmetric Multi-Processing system）及大规模并行系统（MPP=Massively Parallel Processing）上运行。

对称多处理器（SMP）：

优点：应用开发简单、管理容易。
缺点：容错差，如果有一个元件损坏，则导致系统瘫痪。

一致的内存访问（UMA）

一致的内存访问（UMA= uniform memory access）可以使所有的处理器在相同的速度下访问内存。在这样的配置下，内存访问是一致的，有时也叫对称多处理系统（SMP）。

非一致性内存访问
非一致性访问所有的处理器可以访问所有的内存结构。但是每个处理器的内存是可变的。

CLUSTER 和大规模并行处理器（MPP）：

Cluster 和 MPP 都消除了 SMP 在系统容错上的不足。这种系统由多个节点组成。每个节点都包括一个简单的处理器（CPU 或 SMP 单元）以及各自复杂的系统内存。

§56.2 Oracle 并行服务器

在 Oracle8i 系统中。只有 Oracle Enterprise Edition 才支持并行处理平台，Oracle Workgroup Server 不支持并行处理平台。

§56.2.1 Oracle 并行服务器组成

在 Oracle 8i 企业版系统中，需要下面的部件才能实现并行处理：

- Parallel Query 选项
- Parallel Server 选项
- Distributed Database 选项
- Data Replication 选项
- 64 位选项

1．Parallel Query 选项通过将查询工作分配到多个处理之间执行来完成复杂的任务，从而提高速度。象全表扫描这样的操作可以被分配到各个 CPU 上执行。

2．Parallel Server 选项在一个或多个服务器上或在网络的一个节点上，它为 Oracle 实例和数据库创建了一个实时的拷贝。它的主要工作是：
1)在同一个网络的各个独立文件服务器间协调用户的 负载平衡。
2)对一个节点进行热备份。创建一个 Oracle RDBMS 的实时更新备份。
3)根据数据值或一个哈希算法划分表和索引，从而允许不同的 CPU 在独立分散的磁盘驱动器上来分割大的数据查询。

3．Distribute Database 选项将两个以上的数据库连接成一个数据库。远程的数据库地理位置和服务名对用户来说是透明的。

4．Data Replication 选项主要完成从主数据库拷贝或更新到当前的本地机器上。

5．64 位选项可以在许多操作系统下运行，有了 Oracle 64-bit 选项，可使速度比 32 位提高 100 多倍。此外还有下面的优点：
1）为 Oracle 实例提供几十 GB 的内存；
2）可以在内存缓存多达几十 GB 的数据量；
3）实现更快的数据库读写；
4）64 位可执行代码；
5）Oracle Pro*C 和 Pro*COBOL 的 64 位可执行代码。

§56.2.2 Oracle 动态并行执行

Oracle 的并行部件可以在具有并行的硬件上运行，也就是说，Oracle Parallel Server 可以在SMP、Cluster 及 MPP环境上一样的运行。可以在数据流操作运算符进行并行执行任务的有：

- 表扫描
- 分类(order by)
- 嵌套循环，分类－哈希及哈希连接
- 求和（SUM, AVERAGE 等）
- 分组（GROUP BY ）
- 联合操作（UNION,UNION ALL）
- 排除重复（DISTINCT）
- 复制表（CREATE TABLE AS SELECT …）
- 数据下载
- 使用嵌套查询的(UPDATE,INSERT 和 DELETE)
- 建立索引

§56.3 Oracle 并行服务器

关于并行服务器的安装与配置请参阅：
《Oracle8 i Parallel Server Setup and Configuration Guide》**Part No. A76934-01**

1.要购买 Oracle 的 Parallel Server 产品；
2.Oracle Parallel Server Software Components
3.Operating System Dependent Layer

了解操作系统供应商的支持，Oracle所认可的操作系统等。

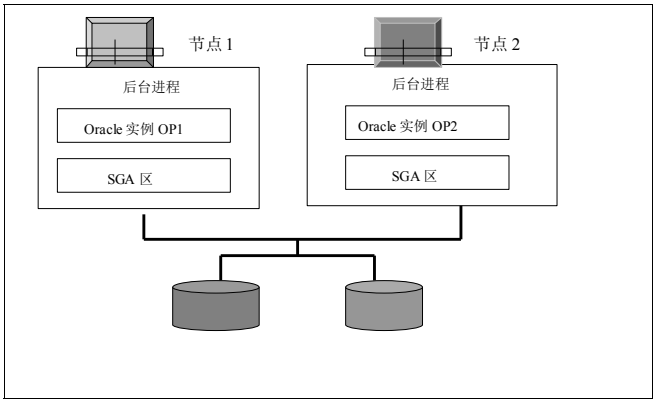
§56.3.1 Oracle 并行服务器概述

Oracle 并行服务器是一组使多个实例能访问一个共享数据库（文件）的结构。它提供：
- 通过节点间工作量加载实现可伸缩性；
- 通过多节点访问数据库实现高可用性。

它可实现所有的实例访问：
- 同一组数据文件；
- 同一组控制文件。

当一个节点失败后，客户应用可以重新连接到另外的节点上。

下图是两个实例共享一组数据库文件的例子：



两个实例共享一组数据库文件 (pallel 图)

§56.3.2 Oracle 并行服务器软件组成

在安装Oracle的OPS(Oracle Parallel Server)软件前，一定要安装供应商提供的、Oracle认可的有关产品。包括两个主要部分：

- 集群管理器(CM=Cluster Manager)：发现集群及实例成员的状态。
- 进程间通信（IPC=Inter-Process Communication）：提供节点间实例的可靠通信。

§56.3.3 Oracle 并行服务器安装概述

Oracle并行服务器安装需要下面软件：

- 操作系统有关的软件层
- Oracle8i 企业版（Oracle8i Enterprise Edition）
- Oracle并行服务器（Oracle Parallel Server）

类似，建立块，必须按照下面顺序安装各部件：
1. 由供应商提示安装操作系统层软件；
2. 从企业版光盘中安装Oracle8i 企业版和Oracle并行服务器(Oracle Parallel Server)

§56.3.3 数据库配置概述

要在并行环境下安装Oracle并行服务器，首先要建立数据库。数据库建立涉及：

1. 建立原始设备：

- 共享数据库文件
- 共享控制文件
- 各个节点重做日志文件

2. 使用Oracle数据库配置助理或手工建立数据库。

§56.4 Oracle 并行服务器安装前任务

§56.4.1 并行服务器节点硬件或软件环境需求

硬件：

- 操作要求的特殊硬件；
- 外部共享硬盘

软件：

每个节点都需要下面软件：

- 安装手册中指定的特殊软件；
- 操作系统层的软件；
- Oracle8i 企业版；
- Net8服务器（NET8 Server）；
- Oracle并行服务器（Oracle Parallel Server）；
- Oracle 智能代理（Oracle Intelligent Agent）；
- 安装下面软件之一：

– Netscape Navigator Version 4.0 或更高版本（http://www.netscape.com）
– Microsoft Internet Explorer Version 4.0 或更高版本（http://www.microsoft.com）

§56.4.2 企业管理器的硬件或软件环境需求

可以用Oracle 企业管理器来管理环境变量等。比如：

- 控制台及DBA工作室

控制台通过Oracle智能代理来执行单个数据库的管理。如果DBA工作室是带单节点或没有与管理服务器连接，则需要配置tnsnames.ora 文件。

- 管理服务器

管理服务器执行控制台需要的功能。如果管理服务器是在一台分开的机器上，还需要安装和配置NET8。

- 知识数据库

知识数据库是一组管理所需要的Oracle的表。

- 通过网络管理有关代理目标

一个管理目标是一个可以被Oracle企业管理器管理的任意的服务或实体，如节点。数据库，Web服务，监听器及页服务等。每个Oracle 并行服务器必须有 V8.1.6 或更高版本的Oracle 智能代理（Oracle Intelligent Agent）。

操作系统：

- Windows NT version 4.0
- Solaris 2.6

企业管理器软件（Enterprise Manager Software）

- Oracle企业管理器(Oracle Enterprise Manager version 2)
- Oracle诊断包（Oracle Diagnostics Pack version 2)
- Net客户端（ Net8 Client on the machine where the Diagnostics Pack is installed. ）

§56.4.3 共享磁盘子系统

Oracle并行服务器需要一个共享磁盘子系统来存放原始设备的共享分区。所有的Oracle8i 数据、日志及控制文件都放在共享原始设备分区里。

注: 每个Oracle并行服务器有它自己的日志文件，但控制文件和数据文件由集群的实例来共享。然而，这些日志文件必须是可访问的和对其它的实例是可读的。

§56.5 设置原始设备

集群不提供对集群的所有节点访问共享文件系统的方法。因此，数据文件、重做日志文件及控制文件都要存放在原始设备里。使所有的实例都可以访问这些数据文件和控制文件。然而，每个实例有它自己的重做日志文件。但所有的实例必须访问所有恢复的日志文件。所以在安装前要确认设置的原始设备是否正确。如果你运行Oracle数据库配置助理（**Oracle Database Configuration Assistant**）时没有设置原始设备。则数据库就不能建立。

为了用Oracle数据库配置助理来建立数据库文件，在数据库建立前原始设备的准确数字必须指定。包括：

- 数据文件6个

- Oracle8i *inter*Media 之一
- 两个控制文件
- 每个节点两个重做日志文件

另外建立原始设备要求如下：

要建立的原始设备	文件大小
SYSTEM 表空间	200 MB
USERS表空间	108 MB
TEMP 表空间	72 MB 多用途或数据仓库联机事务处理需要520MB
RBS 表空间	520 MB 多用途或数据联机事务处理
	1032 MB 数据仓库
INDX 表空间	58 MB
TOOLS 表空间	12 MB
DRYSYS 表空间	80 MB
第一个控制文件	100 MB
第二个控制文件	100 MB
每个节点的重做日志文件	1 MB

注: 在建立完成可以修改其大小，但在NT上的原始设备一旦建立就不能扩展或缩小。

下面给出UNIX和NT的建立原始设备的说明：

UNIX

1. 建立原始设备
2. 建立数据文件、控制文件及重做日志文件。
如：

文件名	原始设备（Raw Device）
db_name_system01.dbf	SYSTEM tablespace raw device
db_name_user01.dbf	USERS tablespace raw device
db_name_temp01.dbf	TEMP tablespace raw device
db_name_rbs01.dbf	RBS tablespace raw device
db_name_indx01.dbf	INDX tablespace raw device
db_name_tools01.dbf	TOOLS tablespace raw device
db_name_drysys01.dbf	DRYSYS raw device
db_name_control01.clt	First control file raw device
db_name_control02.clt	Second control file raw device
db_name_redo	thread_number.log

3. 如果使用特殊安装，还需要建立各个数据库对象及数据文件。

数据库对象　　用　　于：
system1　　SYSTEM tablespace data file
users1　　USERS tablespace data file
temp1　　TEMP tablespace data file
rbs1　　RBS tablespace data file
indx1　　INDX tablespace data file
tools1　　TOOLS tablespace data file
drysys1　　DRYSYS tablespace data file
control1　　First control file
control2　　Second control file
redo　　thread_number

具体如：
system1 device/path/op_system1.dbf
users1 device/path/op_user1.dbf
temp1 device/path/op_temp1.dbf
rbs1 device/path/op_rbs1.dbf
indx1 device/path/op_indx1.dbf
tools1 device/path/op_tools1.dbf
drsys1 device/path/op_drsys1.dbf
control1 device/path/op_control1.clt
control2 device/path/op_control2.clt
redo1_1 device/path/op_redo1_1.log
redo1_2 device/path/op_redo1_2.log
redo2_1 device/path/op_redo2_1.log
redo2_2 device/path/op_redo2_2.log

b. 设置环境变量 DBCA_RAW_CONFIG 使它指向这些数据文件。

当Oracle数据库配置助理建立数据库时，它要查找这些环境变量，读ASCII文件，并在建立表空间时用到文件。

Windows NT

由于NT不支持真的分布系统，因此，数据文件，控制文件及日志文件都驻留在未格式化的原始设备里。一个扩展分区指到原始磁盘以便为数据文件分配多个逻辑分区。
因为在NT上原始设备没有文件名或驱动符相关联。所以需要建立多个逻辑分区，并且用下面的格式分配*符号名字*:
\\.\ symbolic_link_name.
符号与逻辑分区连接相当简单。如SYSTEM表空间的 \\.\op_system1。当建立SYSTEM表空间时，数据文件的拷贝就是\\.\op_system1。它与一个逻辑分区连接。

符号连接名字　　用于：

\\.\db_name_system1 SYSTEM tablespace data file
\\.\db_name_users1 USERS tablespace data file
\\.\db_name_temp1 TEMP tablespace data file
\\.\db_name_rbs1 RBS tablespace data file
\\.\db_name_indx1 INDX tablespace data file
\\.\db_name_tools1 TOOLS tablespace data file
\\.\db_name_drsys1 DRSYS tablespace data file
\\.\db_name_control1 First control file
\\.\db_name_control2 Second control file
\\.\db_name_redo thread_

§56.6 安装前步骤

1. 安装供应商给出的操作系统层软件；
2. 按照供应商提供的文档执行集群诊断；
3. 建立原始设备；
4. 如果是UNIX，则以root来执行下面的步骤：
a. 确认在所有的集群节点上的/etc/group 文件中有OSDBA组和OSOPER组，
b. 在每个节点上建立Oracle帐户，使：
- 该帐号是 OSDBA 组的一个成员。
- 该帐号用于安装和更新Oracle软件
- 该帐号在远程有写的权限

c. 建立一个安装点目录，使：
- 每个节点的安装点名字是一样的。
- Oracle帐户有读、写和执行的权限。

d. 在节点上运行Oracle通用安装程序。并以Oracle帐户运行：
/etc/hosts.equiv 文件

e. 当做完后，退出root 帐户。

5. 对于UNIX集群，以Oracle帐户，检查每个节点的Oracle用户的远程登录(rlogin)。 如果提示口令，那么说明在所有节点上没有给出相同的名字。所以Oracle通用安装程序就不能使用rcp命令拷贝Oracle产品到远程的目录上。

2.数据库配置

1). 建立原始设备:
- 共享数据文件
- 共享控制文件
- 各个节点的重做日志文件

2). 用 Oracle 数据库配置助理建立数据库

3.安装并行软件服务器

4.配置服务器

§56.7 安装过程

在安装过程中，要在节点上安装Oracle软件部件。下面是安装的步骤：
1. 确认安装前的任务完成并正确。
2. 从节点启动安装程序前，要确保节点有足够的权限。
3. 在集群的节点上安装Oracle8i 企业版。
4. 要注意下面的任务的安装：

a. 选择 Oracle8i Enterpise Edition 上有效的产品
b. 选择特殊安装或客户安装
d. 如果是客户安装，要确认并行服务器在Oracle选件文件夹内有效。除非在集群上进行过特别的配置，一般Oracle安装程序是看不到Oracle并行服务器的。
e. 从屏幕上选择节点。
如果列出的节点不包括所希望的节点，则可能是供应商的集群件没有安装。没有运行或没有配置正确。如果集群件没有安装，点击“Previous”。安装Cluster Manager (CM),然后点击“Next”。这样该节点就会看到。
f. 数据库的标识，要输入全局名（**global database name**），即数据库名和数据库域名。

§56.8 Oracle 的安装配置

Oracle在安装中，默认一般是按照OFA**(Optimal Flexible Architecture)**目录结构来安装产品。在相应的目录里，存放有关键的文件：
UNIX中的 oratab 文件：
Oracle并行服务器数据库也在oratab文件上建立，语法：

```
db_name:$ORACLE_HOME:N
```

N 表示在重新启动期间不共享。见下面例子(OP是数据库名字)：

OP:/private/system/op:N

UNIX 上的db_name.conf 文件：
该文件在$ORACLE_HOME/ops上找到。 Oracle企业管理器使用该文件以便知道哪个实例在节点上运行。

初始化参数文件（init sid.ora 和init db_name.ora）：

Oracle使用参数文件来启动实例。每个节点都有一个实例使用initsid.ora 参数文件和并行服务器使用的db_name.ora参数文件。

文件位置：
对于特殊安装， Oracle配置助理在UNIX的$ORACLE_BASE/admin/ db_name/pfile建立init sid.ora和db_name.ora文件。 在NT的$ORACLE_BASE\admin\db_name\pfile建立init sid.ora和db_name.ora文件。

多线程的配置：
特殊安装，需要在init db_name.ora文件配置MTS:

- 为OLTP配置至少20个用户；
- 共享服务器模式配置：

mts_dispatchers="(protocol=tcp)(listener=listeners_db_name)"

非默认监听器：
如果配置一个非默认的监听器，就要在init sid.ora文件上指定配置为：

local_listener = listener_sid

§56.9 监听器(listener.ora)

在配置listener.ora文件中，服务段和客户段用相同的配置以使客户能与服务器连接。默认安装，Net8配置助理建立的是默认的监听器叫LISTENER。Oracle 企业管理器通过listener.ora来发现数据库。数据库信息包括在global database name中。监听器建立后由Net8配置助理来启动。

例如：

```
listener=
(description=
(address=(protocol=ipc)(key=extproc)))
(address=(protocol=tcp)(host=op1-sun1)(port=1521)))
```

sid_list_listener=
(sid_list=
(sid_desc=
(sid_name=plsextproc)
(oracle_home=/orahome81)
(program=extproc)
(sid_desc=
(oracle_home=/orahome81)
(sid_name=op1)))

这里的第2个SID_DESC是实例的入口（不用GLOBAL_DBNAME参数）。这个入口是对单实例而言。
(sid_desc=
(global_dbname=sales.us.acme.com)
(sid_name=sales)
(oracle_home=/u01/app/oracle/8.1.6)))

注意：在Oracle 并行服务器中，Oracle公司要求不要用GLOBAL_DBNAME参数，因为此参数不激活连接次数或透明应用。

§56.10 目录服务访问(ldap.ora)

如果在 Net8配置助理中选择 LDAP 目录服务的话，则建立了 ldap.ora 文件，内容包括：

- 目录类型
- 目录位置
- 可管理的文本

§56.11 Net 服务名(tnsnames.ora)

数据库连接（database connect）：

Oracle 企业管理器搜寻数据库入口名，这项使得 Oracle 企业管理器能发现 Oracle 并行服务器，并确定可以用哪一个进行连接。为每个实例配置协议地址。还要加上 LOAD_BALANCE 和 FAILOVER 选项。

例子：op.us.acme.com 目标数据库：
op.us.acme.com=
(description=

(load_balance=on)
(address=(protocol=tcp)(host=op1-sun)(port=1521)
(address=(protocol=tcp)(host=op2-sun)(port=1521)
(connect_data=
(service_name=op.us.acme.com)))

注：FAILOVER 默认就是 FAILOVER = ON ，所以不用再设置。

实例连接入口（instance connect）：

例2：用于Oracle企业管理器在op1_server上连接到op1实例：

op1.us.acme.com=
(description=
(address=(protocol=tcp)(host=op1-server)(port=1521))
(connect_data=
(service_name=op.us.acme.com)
(instance_name=op1)))

NT实例连接入口（instance connect）：

在NT中，服务名由sid_startup而（不是数据库服务名）来识别。远程连接可以从一个其它节点来执行。如果数据库关闭后远程连接也接受SERVICE_NAME和INSTANCE_NAME参数，因此远程连接必须来在专门服务器模式启动SID。下面例子是Oracle企业管理器启动Opl实例的代表：

op1_startup.us.acme.com=
(description=
(address=(protocol=tcp)(host=op1-pc)(port= 1521))
(connect_data=
(sid=op1)
(server=dedicated))

当Oracle企业管理器在UNIX上启动一个实例时，它将命令传给一个节点进程以执行当地的连接，因此说，这些入口并没有建立。

远程监听器：

作为MTS配置，需要在init sid.ora文件中配置MTS_DISPATCHERS参数。如：

mts_dispatchers="(protocol=tcp)(listener=listeners_ db_name)"

这样可使实例知道有关其他节点上的远程监听器。

下面是特殊安装所建立的文件：

```
op.us.acme.com=
(description=
(load_balance=on)
(failover=on)
(address_list=
(address=(protocol=tcp)(host=op1-sun)(port=1521))
(address=(protocol=tcp)(host=op2-sun)(port=1521)))
(connect_data=
(service_name=op.us.acme.com)))
op1.us.acme.com=
(description=
(address=(protocol=tcp)(host=op1-sun)(port=1521))
(connect_data=
(service_name=op.us.acme.com)
(instance_name=op1)))
op2.us.acme.com=
(description=
(address=(protocol=tcp)(host=op2-sun)(port=1521))
(connect_data=
(service_name=op.us.acme.com)
(instance_name=op2)))
listeners_op.us.acme.com=
(address=(protocol=tcp)(host=op1-sun)(port= 1521))
(address=(protocol=tcp)(host=op2-sun)(port=1521))
extproc_connection_data.us.acme.com=
(description=
(address_list=
(address=(protocol=ipc)(key=extproc))
(connect_data=
(sid=plsextproc)
(presentation=RO)))
```

§56.12 PROFILE 文件(SQLNET.ORA)

在安装完成后，就有了 SQLNET.ORA 文件。它包含计算机的域名和连接描述。它的命名顺序是：目录名、tnsnames.ora 文件、Oracle 命名服务器（Oracle Names Server）及主机名。典型的

SQLNET.ORA 如下：

names.default_domain=us.acme.com
names.directory_path=(**tnsnames, onames,hostname**)

§56.13 安装之后建立数据库

如果在安装时没有建立数据库，则可以用Oracle数据库配置助理或手工来建立数据库。

§56.13.1 用 Oracle 数据库配置助理建立数据库

准备：
为了用Oracle数据库管理器建立数据库，必须建立表空间的原始设备。在数据库建立期间，Oracle数据库管理器要确认每个表空间的原始设备已经建立才行。如果原始设备设置不合适，则Oracel数据库管理器建立数据库就失败。

建立数据库：
用Oracle数据库配置助理建立数据库：
1. 在节点上启动Oracle数据库配置助理

- 在 NT： 开 始 > 程 序 > Oracle for Windows NT -[*HOME_NAME*] > Oracle Database Configuration Assistant.
- UNIX： 在 $ORACLE_HOME/bin 目录上运行 **dbassist** 进入交互界面 。
- 在UNIX上：检测锁管理器软件或节点列表。
- 在NT上：加载客户管理器软件（见供应商提供的文档），然后启动Oracle 数据库配置助理。

2. 选择Oracle Parallel Server 配置，然后点击“Next”

3. 点击“Create a database”，然后点击“Next”

4. 选择希望建立数据库的节点

5. 选择建立数据库的特殊或客户安装

6. 回答Oracle 数据库配置助理页的提问

- 输入适当的 **global database name**, 数据库名及数据库域名，如： op.us.acme.com.
- 在SID字段内接受或改变 **Oracle System Identifier (SID)**

7. 当提示最后的屏幕时，点击“Finish”来建立Oracle的并行服务器数据库。

为使数据库建立成功，Oracle 数据库配置助理加了一些必要信息到网络配置中。如果目录访问由NET8配置助理完成，数据库服务被加到该目录中。则客户也可以用该网络信息来访问数据库服务而且不要tnsnames.ora文件也能连接到数据库中。

§56.13.2 用手工建立数据库

你可以使用现成的脚本来手工创建数据库。手工主要涉及下面任务：

任务1: 指定数据库和实例设置
任务2: 备份已有的数据库
任务3: 设置操作系统配置
任务 4: 设置各个节点的ORACLE_SID
任务 5: 建立初始参数文件.
任务 6: 建立口令文件.
任务 7: 准备 CREATE DATABASE OPS 脚本.
任务 8: 建立数据库.
任务 9: 备份数据库.
任务10: 配置Net8节点。

§56.14 以并行模式启动立数据库

1. 在NT，在各个节点上启动 OracleService*esid*实例
- 从MS-DOS 用命令：

C:\> net start OracleService sid
- 从控制面板上， 选择 OracleService*esid*，再点击 Start.

2. 如果监听器没有启动，则用下面命令启动：

```
LSNRCTL
LSNRCTL> start [ listener_name]
```

3. 在某个节点启动数据库：

```
SQL> CONNECT internal/ password
SQL> STARTUP;
```

4. 在其余的节点上启动数据库：

```
SQL> CONNECT internal/ password
SQL> STARTUP;
```

§56.15 确认实例在运行

1. 在任何节点上输入：

```
SQL> CONNECT internal/ password
SQL> SELECT * from v$active_instances;
Output similar to the following is returned.
INST_NUMBER INST_NAME
----------- ----------------
1 op1-sun:op1
2 op2-sun:op2
3 op3-sun:op3
```

2. 连接到所有节点上，查询emp表：

```
SQL> CONNECT scott/tiger
SQL> SELECT * from emp;
```

能查出结果，就表明正确。

§56.16 Oracle 并行服务器客户端配置

要配置连接多个监听地址，要用FAILOVER=ON 和 LOAD_BALANCE=ON 来实现。比如在tnsnames.ora文件中配置成：

```
op.us.acme.com=
(description=
(load_balance=on)
(failover=on)
(address=(protocol=tcp)(host=idops1)(port=1521))
(address=(protocol=tcp)(host=idops2)(port=1521))
(connect_data=
(service_name=op.us.acme.com)))
```

配置完成后，可用下面步骤进行测试：

1. 在某个节点或客户端用下面命令连接带实例：

SQL> CONNECT internal/ password@net_service_name

2. 进行更新测试：
SQL> UPDATE emp
set sal = sal + 1000
where ename = 'miller';
commit;

3. 在其他节点上查看结果：
SQL> SELECT * from emp;

§56.17 Oracle 并行服务器参数文件

由Oracle 数据库配置助理建立的1号节点和2号节点的两个参数文件，如：

例1：initop1.ora文件：

ifile='C:\OracleSW\admin\op\pfile\initop.ora'
rollback_segments=(rbs1_1,rbs1_2)
thread=1
parallel_server=true
instance_name=op1
remote_login_passwordfile=exclusive

例2：initop2.ora文件：

ifile='C:\OracleSW\admin\op\pfile\initop.ora'
rollback_segments=(rbs2_1,rbs2_2)
thread=2
parallel_server=true
instance_name=op2
remote_login_passwordfile=exclusive

其中：
IFILE 参数文件的路径和文件名
ROLLBACK_SEGMENTS 指定一个或多个回滚段
THREAD 指定重做线程号，每个实例用不同的线程号。
INSTANCE_NAME 唯一的实例名
INSTANCE_NUMBER 影象到每个数据库的自由列表的号
REMOTE_LOGIN_PASSWORDFILE 口令文件，设置为EXCLUSIVE. EXCLUSIVE，表示可以

允许多个用户（不是以sys和internal）来启动数据库。

例3：initop.ora文件：

```
db_name="op"
db_domain=us.acme.com
service_names=op.us.acme.com
db_files=1024 # INITIAL
control_files=("\\.\op_control1", "\\.\op_control2")
open_cursors=100
db_file_multiblock_read_count=8 # INITIAL
db_block_buffers=13816 # INITIAL
shared_pool_size=19125248 # INITIAL
large_pool_size=18087936
java_pool_size=2097152
log_checkpoint_interval=10000
log_checkpoint_timeout=1800
processes=50 # INITIAL
parallel_max_servers=5 # SMALL
log_buffer=32768 # INITIAL
max_dump_file_size=10240 # limit trace file size to 5M each
global_names=true
oracle_trace_collection_name=""
background_dump_dest=C:\OracleSW\admin\op\bdump
user_dump_dest=C:\OracleSW\admin\op\udump
db_block_size=4096
remote_login_passwordfile=exclusive
os_authent_prefix=""
distributed_transactions=10
mts_dispatchers="(protocol=TCP)(lis=listeners_op)"
compatible=8.1.0
sort_area_size=65536
sort_area_retained_size=65536
```

BACKGROUND_DUMP_DEST 跟踪文具的路径
CONTROL_FILES 指定控制文件名
DB_NAME 指定数据库名 op
DB_DOMAIN 指定数据库域名
MTS_DISPATCHERS 为数据库使能**multi-threaded server (MTS)**
MTS_DISPATCHERS 还可以包括许多属性，如：

- PROTOCOL (PRO or PROT) 发布生成器的网络协议
- LISTENER (LIS or LIST) 监听器的别名

§56.18 配置恢复管理器

下面给出RMAN在并行服务器环境下的配置：
1. 配置目录使所有的归档日志文件可被所有的节点在备份和恢复时以共享来访问。
2. 配置归档日志使得归档日志可写到多个目标数据库上。在网络中，SERVICE_NAMES 指定数据库服务名，如： op.us.acme.com 。默认下，配置助理建立成全局名字，即包含数据库名(DB_NAME)和域名(DB_DOMAIN)。
如果可能可以提供多个服务名（由独立 SERVICE_NAMES入口）。这样实例可以以不同的用法来区别。
USER_DUMP_DEST 指定跟踪文件的目录

§56.19 为 RMAN 配置目录

UNIX 目录配置：

使用网络系统NFS(Network File Server) 配置共享归档日志目的文件。即为每个实例建立相同的归档日志目录结构。比如对于一个有三节点的集群。其中一个入口之一就是本地的归档日志。比如每个节点都有下面的文件：

$ORACLE_HOME/admin/ db_name/arch1
$ORACLE_HOME/admin/ db_name/arch2
$ORACLE_HOME/admin/ db_name/arch3

每个实例写归档日志到它所在的本地目录上，同时也写到远程的目录上。

NFS考虑事项：
Oracle公司建议使用NFS来达到高实用性或软件安装NFS目录。

1)NFS高实用性:
最佳的方案就是使用NFS来执行高实用性。这种方案在共享磁盘上输出NFS目录存储。有一个节点是用于访问文件的主要节点。如果该节点失败，失败进程（failover process）改变该访问路径到一个备份节点也能访问到共享磁盘。如果采用高实用性，还要考虑供应商的配置特点。

2)软件启动NFS目录:
软件启动指一个进程试图访问一个失败后（但未被阻塞的启动）的目录（ mounted directory）直到该目录变为有效为止。

例如在SUN的Solaris 操作系统下，可用下面命令进行启动：

mount -F NFS -o soft,rw,retry=10,timeo=30 node1:
/ORACLE_HOME/admin/ db_name/arch1
/ORACLE_HOME/admin/ db_name/arch1

要确保每个节点在其本地分区上都有（产生）归档日志，就要设置LOG_ARCHIVE_DEST 参数为本地的归档指定路径。比如，以前的例子可以修改为：

In initop1.ora enter:
log_archive_dest_1="location=/ ORACLE_HOME/admin/ db_name/arch1"

In initop2.ora enter:
log_archive_dest_1="location=/ ORACLE_HOME/admin/ db_name/arch2"

In initop3.ora enter:
log_archive_dest_1="location=/ ORACLE_HOME/admin/ db_name/arch3"

Windows NT:

1. 在集群（cluster）上分配一个未使用过的驱动器给每个节点。比如有节点idops1, idops2和idops3，驱动器J, K, L 。则可以分配为：

节点名　　驱动器
idops1　J:
idops2　K:
idops3　L:

2. 使用NT磁盘管理器建立新的包括NTFS的逻辑分区。每个分区是实例所运行节点的回档日志目的目录。进行这样的配置，要分配节点的所有者给新的分区。看下面例子：
在建立新分区时也就建立了目录层次，如\archivelogs ，节点和分区的分配如下：

节点名　　命令
idops1　mkdir J:\archivelogs
idops2　mkdir K:\archivelogs
idops3　mkdir L:\archivelogs

3.在每个节点上，用下面命令来共享新的NTFS分区:
net share < db_name>_logs=< drive_letter>:\

节点名 命令
idops1 net share op_logs=J:\

idops2 net share op_logs=K:\
idops3 net share op_logs=L:\

4. 使用NT 的Explorer 设置共享驱动器的许可权
5. 从远程来映射共享驱动器，如：
net use \\< node_name>\< db_name>_logs < drive_letter>:

具体例子如下：
在 idops1,有本地 drive J:, 则输入：
net use \\node2\OP_logs K:
net use \\node3\OP_logs L:

在 idops2,有本地 drive K:, 则输入：
net use \\node1\OP_logs J:
net use \\node3\OP_logs L:

在 idops3,有本地 drive L:, 则输入：
net use \\node1\OP_logs J:
net use \\node2\OP_logs K:

6. 在本地分区里，确保每个节点有相应的归档日志文件，就要在每个实例中设置 LOG_ARCHIVE_DEST_*n* 参数。如：在 initop1.ora 上输入：
log_archive_dest_1="location=J:\archivelogs"
In initop2.ora enter:
log_archive_dest_1="location=K:\archivelogs"
In initop3.ora enter:
log_archive_dest_1="location=L:\archivelogs"

§56.20 并行服务器管理结构

你可以控制 Oracle 并行服务器的行为和使用 Oracle 并行管理。Oracle 并行管理器是一个综合的、集成的 Oracle 并行服务器管理解决方案。Oracle 并行管理可以在异种环境下通过客户-服务结构来使多个实例运行。此外，还可以管理并行数据库、性能管理、监视性能及获得调整数据库所需的统计数据等。

Oracle 企业管理器（Oracle Enterprise Manager）为 DBA 提供一个强有力的管理工具：监视、从单工作站上管理复杂的数据库网络。因此也称企业管理器控制台（Enterprise Manager Console）。

Oracle并行服务器可以管理的部件和内容如下：

部件	描述
控制台（Console）	控制台（Console）是一个图形界面，它可以在客户端上或浏览器上运行。 ● 导航窗口（Navigator window）--包括对象浏览及数据库对象的观察等。 ● 组窗口（Group window）--提供客户化的管理员窗口。 ● 事件管理窗口（Event Management window）--提供远程管理窗口。 ● 作业窗口（Job window）--提供管理员可以自动重复操作的能力。
管理服务器（Management Server）	管理服务器（Management Server）执行控制台请求。它处理所有系统的任务并管理不同节点上的Oracle 智能代理（Oracle Intelligent Agents）。
知识库（Repository Database）	所有管理员共享的数据库表视图，包括每个管理员的帐号。管理服务器利用这个知识库存储所有系统的数据、应用数据及管理节点的数据。 这些知识库必须分开存放在各个节点上。
Oracle智能代理（Oracle Intelligent Agents）	Oracle智能代理（Oracle Intelligent Agents）控制和完成由控制台发来的任务。一旦安装完成，Oracle智能代理可以进行： ● 监听和回答由控制台发出的作业请求； ● 进度作业请求，包括检测和校正异常条件，运行标准的数据库管理存储过程及监视事件的情况等。 此外，Oracle智能代理在节点上需要安装。
Oracle性能管理器（Oracle Performance Manager）	Oracle性能管理器（Oracle Performance Manager）可以使从从多的图形性能统计中选择Oracle并行服务器的性能。这些统计代表所有并行服务器的所有性能的总计。统计结果可以按单独的图形显示，包括数据块的ping的信息、锁的行为、文件的I/O及会话与用户信息等。
Oracle数据搜集器(Oracle Data Gatherer)	Oracle数据搜集器 搜集Oracle的性能统计信息。它在安装智能代理时被安装。

§56.21 并行实例的启动

用 SQL*Plus启动实例：

1. 在参数文件中设置 PARALLEL_SERVER 为TRUE；

2. 确认Cluster Manager 软件在运行。如果提示下面信息，就是不能运行：

ORA-29701: "Unable to connect to Cluster Manager".

3. 启动操作系统必需的进程。见与Oracle有关的操作系统文档。

使用RETRY 来启动数据库为共享模式

如果当前的数据库被另外的实例正在进行恢复操作，而你希望重启动该实例和以共享模式安装数据库。那么你的新实例就不能安装(启动)数据库直到恢复完成为止。如果不想等到恢复完成就启动该实例。就要用STARTUP RETRY语句来实现。在这样的情况下，新实例每5秒就试图安装该数据库直到安装成功为止。语法为：

STARTUP OPEN database_name RETRY

设置和连接到实例：

默认下，一般就连接到当前的实例中。要从当前的实例连到远程的实例，就要进行：

- 重新执行 CONNECT 命令使它指到远程实例的net 服务名上：

CONNECT SYSTEM/MANAGER@ net_service_name

- 可用 SET INSTANCE 命令来去掉连接，从而再连接到其它的实例上。

SET INSTANCE net_service_name

用SET INSTANCE 和 SHOW INSTANCE 命令：

用 SET INSTANCE 可以指定到远程节点。
用 SHOW INSTANCE 可显示当前实例的net 服务名。

§56.22 并行实例的关闭

关闭掉一个并行服务器实例与关闭掉一个单独的实例一样。

- 在并行服务器下，关闭掉一个实例不会影响正在运行的实例。
- 在共享模式下关闭掉一个数据库，就等于关闭了该集群的每个实例。
- 当正常关闭掉一个实例，Oracle强行将所有的在该实例上正在运行的用户进程退出数据库。如果用户正在访问数据库，Oracle就终止该访问并提示"ORA-1092: Oracle instance terminated. Disconnection forced"。一般的用户得到提示是："ORA-1012: Not logged on"。
- 用NORMAL或IMMEDIATE关闭数据库，不需要实例恢复。但如果以SHUTDOWN ABORT关闭数据库，则SMON进程还得进行实例恢复。
- 如果多个SQL*PLUS 会话连接到同一个实例上，则所有的会话都必须断开才能进行正常关

闭。你可以用SHUTDOWN IMMEDIATE 或SHUTDOWN ABORT 来关闭掉实例。

第 57 章　并行环境应用设计

当管理员将并行环境搭建好后，要使应用能利用并行环境来提高性能的话，还要考虑应用的设计问题。

本章参考《Oracle8 i Parallel Server Administration, Deployment, and Performance》Release 2 (8.1.6) Part No. A76970-01

§57.1 分析应用

需要分析人员了解Oracle并行服务器如何处理基表里各种事务。下面是几种类型的应用：

§57.1.1 只读的表

可以将表空间设置为只读方式，使得该表空间下的变为只读的表，对于只读的表，Oracle并行服务器快速初始化PCM锁(Parallel Cache Management Lock)为共享模式，并很少出现锁。理想情况下，每个只读的表及其索引结构只需要一个PCM锁。这就是只读表有高性能的原因。另外还要考虑只读表放到只读表空间的问题，它的优点是：

- 它能迅速的恢复
- 不需要PCM锁
- 表空间在只读后只需要进行一次备份。

§57.1.2 随机的查询与修改

对于随机的查询与修改类型应用，它需要多个锁。首先,实例在一个或数据块内执行事务需要一个共享的PCM（共享高速缓存管理）锁。这些锁的请求引起其它节点的性能下降。当实际执行UPDATE时，实例必需得到一个可执行的PCM锁。如果用户在不同的节点上修改同一个数据块，这样的副面影响是很明显的。

§57.1.3 插入、修改与删除

随机进行INSERT,UPDATE 和 DELETE 的表需要读一些数据块和修改这些数据块。这些数据

快需要转换成PCM（共享高速缓存管理）锁共享模式再将修改块转换成独占方式。这样的过程也存在性能问题。
当索引需要更新和管理是，如果随机修改表可能引发性能问题。当Oracle正在为表和索引的存放而寻找自由空间时尤其如此。

当进行Insert、Delete和Update时就是对索引进行键的修改，所以你需要对表的索引进行管理。这个过程需要访问多个索引块（从根搜索到分支、页），因此，潜在的锁就会增加。这些索引的分支或块的页可能分离着的，这样就增加实例冲突的可能性。由此看来，键值的分布很重要，为了自动增加索引键，一个办法就是建立正确的索引键。
如果INSERT 和 DELETE服从于长时间运行的大事务，则对于另外的实例来说就有机会进行连续的读完整个信息。这样的类型需要处理读缓冲区的问题。
当从多个并行服务器节点中使用序列产生器来产生唯一键时，索引块竞争可能是个问题。即当产生唯一键时，就使得产生主键号的实例去**寻找**执行INSERT到不同的索引中，所以分散的插入加载可以克服多个实例和单个实例执行所带来的竞争。

§57.1.4 建立逆序键索引

在Oracle并行服务器环境内建立逆序索引可以改善性能。一个逆序索引在保持列的顺序时候颠倒各个所有列的顺序。由于索引的逆序关键字，在插入时在索引中就变为分布的交叉关键字。例如：

```
CREATE INDEX i ON t (a,b) REVERSE;
```

这里的“a”是实例的号，“b”是唯一键。

在插入操作中，一个扩展中分配的自由列表可以引起很高的锁转换率。这是因为有多个实例都希望插入新的行到同一个数据块内或插入到相邻的数据块中。如果这些数据块是由PCM锁引起的话，要避免它，只能采用对表和索引进行分区来实现。

§57.1.5 应用分区技术

将应用分成子部件以分散到集群的各个节点上。下面是分区的操作步骤：

步骤1: 定义系统主功能区
识别应用的主要功能，如旅馆的管理，考虑有下面的功能：
- 预约
- 财产管理与维修
- 营销
- 前台，房门及餐厅管理

步骤 2: 识别表的访问需求和交叉定义

确定功能内表的访问及交叉，从下面看出有3个表出现交叉：

旅馆预约操作	前台操作
Table 1	Table 12
Table 7	Table 14
Table 15	**Table 15**
Table 11	Table 16
Table 19	**Table 19**
Table 20	**Table 20**

步骤 3: 定义各个交叉的访问

旅馆预约操作	预约交叉 访问类型	交 叉	前台交叉 访问类型	前台操作
Table1	S(Select)	Table15	S(Select)	Table12
Table7	I(Insert)	Table19	I(Insert)	Table14
Table11	U(Update)	Table20	U(Update)	Table16

在例子中，功能访问两次的有：

Table 15

Table 19

Table 20

步骤 4: 识别事务卷

估计希望交叉产生事务的数量

旅馆预约操作	预约交叉 访问类型	交 叉	前台交叉 访问类型	前台操作
Table1	S(10 per second)	Table15	S(50 per	Table12
Table7	I(100 per second)	Table19	second)	Table14
Table11	U(50 per second)	Table20	I(10 per second) U(90 per ecnd)	Table16

步骤 5: 交叉分类

使用下面标准确定如何处理这些表：

- 省去非交叉的表，只选有交叉的表并频率低的表；
- 对表及其索引加以分类；
- 对表及其索引加以混合访问

索引的表及索引：

对于这些表，可以由实例来分配扩展来获得一个独占的锁。这些表的冲突及其索引对于Oracle的并行服务器来说较容易解决。

§57.2 并行环境的数据库设计技术

在共享的服务器数据库环境下来设计数据库，需要考虑的是全局共享数据库要被多个节点的事务来访问。即多个节点要比单节点事务会带来更长的等待时间和消耗更多的 CPU。所以要考虑：

- 分配事务到指定节点；
- 使用参数来建立数据对象，使得在全局共享下更有效。

主要考虑共享数据由集群（各个节点）来交叉共享和同步。采用缓存（Cache）会减少由于全局访问数据库带来的代价。这对于集群的所有节点来说也提高了可用性。当某些事务在并行环境下执行时数据库资源变得很危急。多实例交替对同一个表的块进行访问能引起I/O、消息、文本切换的增长并产生进程空闲。如果一个表有多个索引，这写代价还要高。

§57.2.1 数据库操作、块类型及访问控制

所有的事务都访问数据块，数据块可以归类为：

- 数据块
- 索引块（根，枝，页）
- 段的头块
- 回滚段头块
- 回滚段块

对数据块的同时访问由访问或锁模式来控制。在高速缓冲区中，一个块可以被独占当前读（XCUR= exclusive current read）、共享当前读（SCUR= shared current read）或一致性读（CR= consistent read）模式来访问。为确保全局共享一致，访问模式会映射到全局锁模式上。

§57.2.2 全局缓冲区的一致性工作和块种类

一般数据快，所有块，回滚段头块，自由列表组块，回滚段块及段的头块都被确定为不同类型的块。所有这些块都表现为共享数据库的结构而服从于发布(pinging)
对于回滚段头和回滚段块，由于在Oracle并行服务器环境下，实例有私有的回滚段，所以全局缓冲区同步的效果无太多意义。这些限制要写到回滚段中。由于高速缓冲区能缓解一致性读。所以回滚高速缓存操作的效果受到限制。多数一致性读信息是本地建立会话的实例来产生。
对于表和索引来说，没有自由列表组和不理想的存储参数，则段的头块可能被频繁的被发布（pinged）。配置自由列表组可以减轻这样的影响。一般来说，段的头发布的总和应该小于总发布（ping）5% 。
V$CLASS_PING视图列出不同块种类的特殊类型的转换数量。用该视图可以监视不同块种类

发布的比例。

§57.2.3 产生数据库对象参数建议

对于数据库对象而言，下面的描述具有一般性：

- 对表或索引，在OLTP下，预先估计其增长或缩小，或频繁的插入或从多个节点删除。这样就用FREELISTS 和 FREELIST GROUPS来建立对象。
- 在定义GC_FILES_TO_LOCKS时，预分配（Preallocate）对象的FREELIST GROUP的扩展和（或）使用块因子。这样可确保块是由一个实例来扩展和由相同锁进行转换。
- 由于本地和远程访问的随机性，对于对象和文件来说，要在查询块中限制行的数量或者将PCTFREE设置得很高来得到更多自由空间是个问题。

§57.2.4 索引问题

一般，在建立索引时要避免下面问题：

- 叶和分枝块冲突
- 根块冲突
- 索引段头的冲突

避免叶和分枝块冲突（最小化叶/分枝块冲突）

可以使用各种策略类隔离或对索引的不同的部分作分布访问，从而改善性能。
使用逆序键字索引可以避免索引树的高增长。用了逆序键字索引，可以实现将索引分布到索引块的叶上。这样从多个实例来访问时就降低了访问的概率。当然，逆序键字索引不允许范围扫描。所以使用时要慎重。

对于基于序号的索引，对各个实例分配不同的子序号。在这种类型下，应该采用索引分区方法也能分布访问样本。

§57.2.5 使用序列号

在 Oracle 的并行环境下设计应用，要用到序列号，有时需要修改序列号的缓存数，如：
ALTER SEQUENCE SEQUENCE_NAME CACHE 200;

计算序列号的缓存数

估计序列号的缓存数要考虑三点：

- 事务比率
- 集群中节点数
- 集群的稳定性

§57.2.6 逻辑和物理数据库布局

物理分区策略是：

- 对象的事务概要
- 对象经常如何被访问
- 对象如何被访问
- 与对象有关的功能
- 商务功能
- 数据

将对象分组到表空间再到文件。

物理布局的一般建议：

在将对象分组到表空间前，先要建立一个原始设备卷，以使你后来能分配它给表空间。因为Oracle并行服务器要访问共享原始卷或共享设备分区（对物理存储数据而言）。有些的磁盘可以被分成不同大小的卷。建议如下：

1. 建立一个大的共享卷以便以后分配给逻辑卷
2. 为原始设备定义标准大小的卷，如10M, 100M, 500M, 及 1G 等。
3. 在初始化卷时以多个磁盘来分片或条形化卷。考虑将读写的15%给全局高速缓存同步I/O。

表空间设计

表空间的设计的目标是将数据库对象与它们的访问分布分组在一起。要考虑数据库对象的从属关系和事务概要，就要考虑将表空间分为下面几类：

- 频繁和随机访问的表较少发生
- 表和索引几乎是大部分读或只读和频繁修改
- 表和索引的数据可以再细分为大的不连接的和自治的数据集

将数据库对象分解到表空间，要考虑：

- 表应该从索引上被分解
- 分配为真，则只读的表分配到READ-ONLY 表空间
- 每个节点应该有TEMPORARY表空间
- 将较小的表分组到小的表空间
- 将只插入的表分配到另外的表空间。

§57.2.6 全局缓存锁分配

在完成表空间的设计后，就要估计访问频率及事务和节点的关系。包括：
- 频繁和随机访问的文件是很少或与事务或节点相关。
- 只读或大部分读的文件
- 频繁访问的文件由多个节点来访问，但这些数据被分成大的自治的数据集。
- 文件包含有回滚 段和临时段。

基于这样的细分，分配锁如下：
- 分配可释放的锁给频繁的和多节点随机访问的文件
- 不分配锁给指定的表空间
- 仅分配固定锁给大部分读的数据
- 不分配锁给回滚段和临时表空间对应的文件

不然，由GC_ROLLBACK_LOCKS给少数固定回滚段分配锁。

§57.3 并行缓存设置

下面给出在并行服务器中数据文件的并行缓存管理（PCM=Parallel Cache Management) 锁与非锁的设置。

本章节参考《Oracle8iParallel server Setup and Configuration Guide》中 的 7 Planning the Use of PCM and Non-PCM

§57.3.1 并行缓存管理锁的设置

计划和维护实例锁

分布锁管理允许分配有限数量的锁。所以在应用中分析需求锁的数量和锁所需的内存多少。主要考虑：
- 如果使用比DLM配置时的锁还多。Oracle就显示错误信息和关闭掉失例。
- 改变GC_* 或 LM_* 参数以设置更大的锁和有些的SGA内存量。
- 实例的数量也影响到内存的需求和系统所需要锁的数量。

分配PCM锁

分配PCM锁的关键是分析数据是如何被INSERT, UPDATE和DELETE语句来改变。然后才确定如何将对象聚合到一个文件中。最后在基于该组来分配锁。一般有下面规则：

- 分配几个锁给只读文件；
- 分配较多的锁给读 / 写频繁的文件；
- 如果整个表空间是只读，可以分配一个锁。如果没有分配锁给表空间，则系统使用剩余的锁。这样可能发生冲突，因为其它表空间也可能正在使用剩余锁。这种结果就产生不必要的强制读 / 写。

检查数据文件和数据块

１．确定文件号、表空间名及块号

```
SELECT FILE_NAME, FILE_ID, TABLESPACE_NAME, BLOCKS
FROM DBA_DATA_FILES;

FILE_NAME FILE_ID TABLESPACE_NAME BLOCKS
---------------------------------------------------------------
/v7/data/data01.dbf 1 SYSTEM 200
/v7/data/data02.dbf 2 ROLLBACK 1600
. . .
```

２．确定要求锁的号

使用下面方法估计锁的数量：

- 考虑数据的自然量和应用
- 分配多个锁给文件使多个实例可同时修改。
- 分配少量锁给文件使多个实例可同时访问，可避免不必要的锁空闲。
- 检查文件的对象再考虑在文件的操作。
- 分组只读的对象到只读表空间。

§57.3.2 Oracle 如何给块分配锁

文件到锁的映射：

1.从FILE_LOCK数据字典中查询锁的数量：

```
SELECT * FROM FILE_LOCK ORDER BY FILE_ID;
```

如果设置GC_FILES_TO_LOCKS="1=500:5=200"，则查询的结果如下：

```
FILE_ID FILE_NAME          TS_NAME           START_LK    NLOCKS     BLOCKING
------- ---------- ---------- ---------- ---------- ----------
1          \\.\OPS_SYS01    SYSTEM                100       1500          1
2          \\.\OPS_USR01 USER_DATA            1600          3000          1
3          \\.\OPS_RBS01    ROLLBACK_DATA         0           100          1
4          \\.\OPS_TMP01    TEMPORARY_DATA        0           100          1
```

5	\\.\OPS_USR03	TRAVEL_DEMO	4600	4000	1
6	\\.\PROBLEM_REP	PROBLEM_REP	0	100	1

6 rows selected.

2. 计算表空间中锁的数量（Nlocks 列是锁的数量）：

在例子中，file1和file5有不同的值，所以它们的总和为700个锁。
如果设置GC_FILES_TO_LOCKS="1-2=500:5=200"，则查询的结果如下：

FILE_ID	FILE_NAME	TABLESPACE_NAME	START_LK	NLOCKS	BLOCKING
1	file1	system	1	500	1
1	file2	system	1	500	1
1	file3	system	0		
1	file4	system	0		
1	file5	system	501	200	1

这里，file1和file2有相同的START_LCK，即共享同样的锁500，而File 5有不同的锁200。所以总共锁是700个。

每个块的锁数量：

下面查询每个类分配的锁的数量：

```
SELECT CLASS, COUNT(*)
FROM V$LOCK_ELEMENT
GROUP BY CLASS
ORDER BY CLASS;
```

下面查询固定 PCM 锁的分配数量：

```
SELECT COUNT(*)
FROM V$LOCK_ELEMENT
WHERE bitand(flag, 4)!=0;
```

下面查询 PCM 锁释放的锁的数量：

```
SELECT COUNT(*)
FROM V$LOCK_ELEMENT
WHERE bitand(flag, 4)=0;
```

§57.4 并行服务器应用的调整-

下面给出在并行服务器环境下应用的调整介绍，主要内容包括：

§57.4.1 调整并行服务器概述

在单实例环境下，可以调整高速缓冲区、共享池及并行服务器的磁盘子系统来得到较好的性能，但是Oracle 并行服务器所介绍的参数不能用于单实例环境。这些参数大部分可能提高性能，但如果没有认真分析的话也可能引起别的问题。所以调整Oracle并行服务器环境参数需要监视有关视图和收集统计资料。

在DSS(Decision Support System)系统中，并行可以减少数据密集型操作的响应时间；在一般的OLTP类型的系统中，并行执行可以改进下面的处理过程：

- 需要扫描大型表的查询、连接、和/或分区索引扫描；
- 大型索引的创建；
- 大型表的创建（包括实体化视图的创建）；
- 批量插入、更新和删除。

下面类型的系统特征适合于使用并行：

- 对称多处理器（Symmetric multi-Processor-SMP）、集群或大型并行系统；
- 足够的I/O带宽；
- 低占用率或中等占用率的CPU，比如CPU的使用率小于30%；
- 足够的内存来 支持额外的内存密集型操作，比如排序、散列和I/O缓冲。

并不是任何时候都可以使用并行执行，下面情况要注意：

- 在DSS系统中，可以考虑使用并行；
- 在OLTP系统中，一般可以不使用并行，只有秩序大量的批处理的时候才适合并行，比如银行系统只有进行年终结息时，可以通过并行批处理程序高速地处理利息的更新。

§57.4.2 统计并行服务器性能

1．统计视图 V$SYSSTAT

视图V$SYSSTAT包括下面统计数据：
一致性获取、数据块获取、数据块改变、物理读、物理写等内容。

2．统计视图 V$SYSTEM_EVENT

视图V$SYSTEM_EVENT 包括下面统计数据：
数据文件连续读、数据文件分散读、数据文件并行写等内容。

3．统计视图 V$CACHE

视图V$CACHE 包括有并行服务器的统计数据，主要包括：
数据文件号(FILE#)、块号(BLOCK#)、块的状态（STATUS）等。

4．统计视图V$LOCK_ACTIVITY

视图V$LOCK_ACTIVITY包括有并行服务器的当前实例的DLM锁的情况数据，主要包括：
PCM 锁初始说明（FROM_VAL）、PCM 锁下一个初始说明（TO_VAL）等。

5．统计视图V$CLASS_PING
视图V$ CLASS_PING每个块类ping的块数。主要包括：
块类别编号（CLASS）、独占到空闲（exclusive_to_NULL）的块转换数等。

6．统计视图V$FILE_PING
视图V$ FILE_PING显示每个数据文件ping的块数。主要包括：
数据文件编号（FILE_NUMBER）、频率（FREQUENCY）、文件中所有块从排斥到空闲的锁转换数（X_2_NULL）等。

7．统计视图V$DLM_MISC
视图V$ DLM_MISC显示DLM的统计数据。包括：

统计数据号（STATISTIC#）、统计数据名(NAME)及统计数据有关值（VALUE）。

§57.4.3 确定同步代价

计算CPU数据库需求：

V$SYSSTAT视图存放有 CPU 的用户和事务情况信息。如：
SQL> select name,value from v$sysstat;

NAME VALUE
---------- ----------
• • • • • •

user commits 0
user rollbacks 0
• • • • • •
CPU used when call started 0
CPU used by this session 0
session connect time 0
• • • • • •

估计I/O同步的代价

查询V$SYSSTAT视图的下面字段：
- DBWR cross-instance writes
- physical writes
- physical reads

再查询V$SYSTEM_EVENT视图的下面字段：

- db file parallel write
- db file sequential read
- db file scattered read

§57.4.4 初始化并行执行参数

最基本的并行参数是PARALLEL_AUTOMATIC_TUNING参数，它影响到并行处理方面的几个参数。也就是说此参数是一个**自动调整的参数**，当将它设置为真时，系统会自动进行另外参数的设置。

- 在多数的OLTP系统下，可以设置PARALLEL_AUTOMATIC_TUNING=FALSE
- 在INITsid.ORA参数文件中设置PARALLEL_AUTOMATIC_TUNING=TRUE

当我们启动了 PARALLEL_AUTOMATIC_TUNING=TRUE；则 Oracle 将根据环境的情况，比如并行度(degree of parallelism)情况，自动将 PARALLEL_ADAPTIVE_MULTI_USER=TRUE 设置真。自动根据用户的负载来调整并行度。

****

并行度(degree of parallelism)

当数据库并行执行一条 SQL 语句时，要在大量的并行查询服务器进程上进行，一个操作所使用的并行查询服务器进程的数目就所谓的并行度。

****

1.在表和索引中启用并行度

如果不希望系统使用默认值，可以在创建表中指定并行度，如：

```
CREATE TABLE individual(
Firstname  varchar2(20) not null,
Lastname varchar2(20) not null,
Birthdate   date not null,
Gender  varchar2(1) not null,
Initial_a  varchar2(1),
Favorite_beatle  varchar2(6))
PARALLEL (DEGREE 4);
```

2.强行在会话中使用并行执行

如果用户需要并行处理而又想避免设置表的并行度或者修改相关表的查询时，可以采用在会话状态来强行并行：

```
SQL> ALTER SESSION FORCE PARALLEL QUERY;
```

3.用 PARALLEL_THREADS_PER_CPU 来控制并行性能

PARALLEL_THREADS_PER_CPU 影响并行度和自适应的算法。Oracle 用此参数值乘以 CPU 的数目来得到在并行操作中可使用的线程的数目。该参数的默认值为 2。最高可设置为 8 。

§57.4.5 普通参数的调整

可以进行下面几种类型的调整：

- 并行操作中资源限制的参数；
- 影响资源消耗的参数；
- 与 I/O 有关的参数。

§57.4.5.1 并行操作中资源限制的参数

建立资源限制的参数有：

- PARALLEL_MAX_SERVERS
- PARALLEL_MIN_SERVERS
- LARGE_POOL_SIZE / SHARED_POOL_SIZE
- SHARED_POOL_SIZE
- PARALLEL_MIN_PERCENT
- PARALLEL_SERVER_INSTANCES

1.PARALLEL_MAX_SERVERS

- 如果用户设置有 PARALLEL_AUTOMATIC_TUNING=FALSE ，则需要手工对 PARALLEL_MAX_SERVERS 进行设置。
- 设置该参数的建议值为：

PARALLEL_MAX_SERVERS= 2* DOP(并行度)*并发用户数;

- 如果用户设置有 PARALLEL_AUTOMATIC_TUNING=FALSE，则默认值为 5 。这对于一般的操作是可以的，但对于并行是不够的。
- 简单的方法是该值=CPU 个数*10
- 要设置合适的值，要进行：

select * from GV$SYSSTAT where name like ‘parallel operation%’;

2. PARALLEL_MIN_SERVERS

此参数建议设置为 0 。它的语法是：
PARALLEL_MIN_SERVERS=N

N 为希望保留的并行操作启用的进程数目。

3. LARGE_POOL_SIZE

本参数是共镶池的调整。这个参数没有推荐值，实际上，只要 设置：
PARALLEL_AUTOMATIC_TUNING=TURE 后，由 Oracle 自己来调整该参数值。

如果要设置 LARGE_POOL_SIZE 值，则需要查询 V$SGASTAT 视图，并根据需要来加大或减少此参数的值。

但是如果出现：

ORA-27102:out of memory
SVR4 Error:12:Not enough space

则要减少 **LARGE_POOL_SIZE** 的值。

4. SHARED_POOL_SIZE

如果 PARALLEL_AUTOMATIC_TUNING=FALSE，可由自己来调整该参数值：

- 已经共享池程序（如共享光标和存储程序），就调本参数；
- 较大的值可以改善多用户系统性能；但占用较多内存；
- 要产生更多的光标，建议查询 V$SQLAREA 视图来确定。

5. PARALLEL_MIN_PERCENT

本参数可使应用程序来等待一个可以接收的并行。比如设置 50 ，则表示为 50%,这对于自适应算法或者系统资源的限制就减少 50% 或更多。

- 这个参数建议值为 0 ;
- 如果设置了非 0 的 值，而系统无法满足该值的要求时会出现错误。

6. PARALLEL_SERVER_INSTANCES

- 本参数的推荐值=并行服务器环境的实例数；
- 如果 PARALLEL_AUTOMATIC_TUNING=TURE，则用这个参数来计算 LARGE_POOL_SIZE 的值。

§57.4.5.2 影响资源消耗的参数

影响资源消耗的参数有：

- HASH_AREA_SIZE
- SORT_AREA_SIZE
- PARALLEL_EXECUTION_MESSAGE_SIZE
- OPTIMIZER_PERCENT_PARALLEL
- PARALLEL_BROADCAST_ENABLE

1. HASH_AREA_SIZE

- HASH_AREA_SIZE 确定每个进程可以使用的工作内存的数量；
- 观察交换内存和自由内存，发生交换则减少本参数值，如果出现自由内存，则增加本参数；
- 本参数的计算公式为：

HASH_AREA_SIZE=SQRT(S)/2

这里 SQRT 是平方根， S 是连接的操作数（MB）。

比如 S = 16MB，则 HASH_AREA_SIZE = 2MB 。

2. SORT_AREA_SIZE

- 为每个排序操作的查询服务器进程分配的内存数量；
- 本参数推荐值为 256K 到 4MB 之间；
- 如果用户的内存比较富裕，可设置大些；
- 如果此排序区过小，则进行大量排序时就转移到磁盘进行；
- 本参数与并行操作、DML、DDL 等有关。

3.PARALLEL_EXECUTION_MESSAGE_SIZE

- 此参数指定并行执行的上限，默认与操作系统有关；
- 此参数的推荐值在 4KB 左右；
- 更大的 **PARALLEL_EXECUTION_MESSAGE_SIZE** 需要更大的 LARGE_POOL_SIZE 或更大的 SHARED_POOL_SIZE；

4.OPTIMIZER_PERCENT_PARALLEL

- 此参数确定优化器在多大的程度上执行计划；
- 此参数的推荐值为 100/number_of_concurrent_users；
- 此参数的默认值为 0 。
- 如果设置了 OPTIMIZER_MODE=FIRST_ROWS，则此参数的非 0 值就被忽略。

注意：如果给定合适的索引，Oracle 可以快速从表中取出一记录；这时并不需要并行操作。只有查找某行的全表扫描才用到并行执行。一般是，每个并行进程都检索许多行，这样自然就占用系统的时间和资源。所以一定要了解何时需要并行。

5.PARALLEL_BROADCAST_ENAABLE

- PARALLEL_BROADCAST_ENAABLE 的默认值为 FALSE;
- 当要一个很大的结果集和一个很小的结果集进行连接时，需要设置此参数为 TRUE；
- 当设置为 TURE 时，优化器可以将较小的集合的行向处理较大集合的查询服务器进行广播。

§57.4.5.3 与 I/O 相关的参数

影响 I/O 的参数有：

- DB_BLOCK_BUFFERS
- DB_BLOCK_SIZE
- DB_FILE_MULTIBLOCK_READ_COUNT
- HASH_MULTIBLOCK_IO_COUNT
- SORT_MULTIBLOCK_READ_COUNT
- DISK_ASYNCH_IO / TAPE_ASYNCH_IO

1. DB_BLOCK_BUFFERS

数据缓冲区的块数，根据物理内存来设置，参考 SGA 的设置。

2.DB_BLOCK_SIZE

- 数据库块的大小，推荐值是 8KB 至 16KB；
- 数据库系统安装完成后，此参数就不能修改；
- 要修改此参数，只能创建另外的实例才能指定此参数。

3.DB_FILE_MULTIBLOCK_READ_COUN

- 指定操作系统一次可读多少个数据库块；
- 此参数与操作系统有关；
- 对于数据库块为 8KB,推荐此参数为 8; 数据库块为 16KB，则推荐为 4.

4.HASH_MULTIBLOCK_IO_COUNT

- 此参数指定一个散列连接一次可以读取和写入的块数；
- 增加此参数可以减少散列表单元的数量；
- 默认值为 4;

5.SORT_MULTIBLOCK_READ_COUNT

- 推荐使用默认值；
- 此参数指定一个排序每次读取临时段时可以读取的数据库块数目；
- 当数据在内存不能进行排序时就用临时段；
- 执行过多的 I/O，内存较空闲，则加大此参数的值。

6. DISK_ASYNCH_IO / TAPE_ASYNCH_IO

- 此参数启用或禁止操作系统的异步 I/O 功能；
- 允许服务器进程在执行表扫描时可进行重叠 I/O 请求；
- 推荐值为 TURE.

第 58 章　并行查询管理

在 Oracle 7.1 就已经引入了并行查询选项，在 Oracle8i/9i 又对并行选项进行了增强。用于在多个 CPU 的环境来运行应用。以减少资源密集的 SQL 操作的运行时间。
并行查询选项使用多个服务器进程来执行一定的操作，即将一条语句的执行被分配到多个服务器中执行，协同所有服务器的结果。并将结果返回给用户。

§58.1 Oracle 并行选项、管理及调整

如果有多个CPU和足够的I/O容量，可考虑采用并行选项。并行处理就是把任务分配到多个CPU上，它实际并没有减少处理量。在非并行环境中，一个任务是完全被串行执行的。即只有一个处理器来执行任务。即使有多个处理器，单个语句也不能利用空闲的CPU和I/O的功能。采用并行后，数据密集的SQL语句、数据库恢复和数据加载就由多个进程来处理。

§58.1.1 Oracle 并行选项

可能使用并行处理的任务有：

- 加排序的SELECT语句，包括 UPDATE和DELETE语句中的子查询；
- CREATE TABLE AS SELECT 语句；
- CREATE INDEX 语句；
- SQL*LOADER Direct Path；
- RECOVERY;
- 索引扫描。

在DML并行会话时，要由Alter session parallel 命令来设置并行模式。
要使用Oracle的并行选项，就要设置查询协调进程和多个查询服务器进程。

§58.1.2 Oracle 并行选项设置

§58.1.2.1 用户的限制设置

要确认每个并行查询选项（PQO=Parallel Query Option）的用户都能用由**查询服务器**进程创建的进程，执行查询的用户必须能够创建查询服务器进程。许多操作系统都限制了每个帐户可以建立的进程最大数。在Oracle系统里，还有一个限制用户资源数额的配置文件（PROFILE）。可以采用下面办法来改变这些不合适的参数：
比如要修改操作系统的限制，在Solaris下，可以修改/etc目录下的system文件中的参数，即修改：
set shmsys:shminfo_shmseg=10
这里：

SHMSEG=The maxmum number of shared memory segment that can be attached
by a process. 进程能连接的段的最大数目。

比如要修改Oracle的配置参数，可以用下面命令来完成：
Alter profile default limit session_per_user unlimited;

§58.1.2.2 分配查询服务器进程

要使用查询服务器，需要在initsid.ora文件中设置下面参数(详细参考前面章节)：
1） PARALLEL_MIN_SERVERS
实例启动时需要启动的查询服务器最小进程数。
2)**PARALLEL_MAX_SERVERS**
实例启动时需要启动的查询服务器最大进程数。Oracle向查询服务器池增加查询服务器进程达到PARALLEL_MAX_SERVERS所设置的数时，Oracle就停止增加新的查询服务器进程。
3)**PARALLEL_SERVER_IDLE_TIME**
查询进程不使用时的闲置时间。如果一个查询进程空闲时间比PARALLEL_SERVER_IDLE_TIME设置的时间还长（以分钟），这个查询进程就被终止。
4)PARALLEL_MIN_PERCENT
这个参数允许在得不到要求的并行度的情况下就阻止查询运行。如果得不到所要求的查询服务器进程数，而且设置了PARALLEL_MIN_PERCENT参数，则Oracle只按照进程最小数允许进程数。
PARALLEL_MIN_SERVERS、PARALLEL_MAX_SERVERS和PARALLEL_SERVER_IDLE_TIME三个参数确定：无论何时有多少查询服务器进程在查询共享池中，可以得到的查询服务器多由PARALLEL_MIN_SERVERS来限制最小值；由PARALLEL_MAX_SERVERS来限制最大值。而PARALLEL_SERVER_IDLE_TIME用来确保查询服务器的进程尽量减少。

§58.1.2.3 并行度及其设置

1.并行度定义：
一个查询语句可以使用的并行服务器的数量就是并行度(degree of parallelism)。决定并行度主要由用户的输入、实例的初始化参数及表所数据文件的数量、CPU的数量等。
2.并行度设置：
可以在数据表层和查询层来给出并行度。可以在CREATE TABLE ,ALTER TABLE,CREATE CLUSTER 及ALTER CLUSTER 命令上规定并行度。可以在数据表层的并行度设置来取代任何的实例的设置。在数据表层设置并行度，就是在ALTER TABLE、CREATE TABLE 命令后加PARALLEL子句来完成。如设置emp表的并行度为4，则：
ALTER TABLE EMP PARALLEL（degree 4）;
可以在子句加上instance 关键字来说明在处理中要求的实例个数。比如要求用5个实例，每个实例用4个进程（实际是20路并行），则：
ALTER TABLE EMP PARALLEL(degree 4 instance 5);
3.去掉并行度设置：
可以在ALTER TABLE 、CREATE TABLE 命令后加 noparallel 子句来去掉并行的设置，从而变为串行处理。如：

```
ALTER TABLE EMP NOPARALLEL;
```

§58.1.2.4 关于提示和查询提示强行并行

其实，只要用PARALLEL和NOPARALLEL就可以控制查询的并行度，在查询层上定义的并行度将取代在任何数据层的并行度设置。

1．关于Oracle的提示语法：
可以利用提示改变由基于开销的优化器选择的执行路径。提示就插入在SQL语句里。这个提示只改变该语句的执行路径。提示的语法是：在 SQL 语句后加 /*+ 符号表示提示开始；以*/表示提示结束。它与注释几乎一样。注意 + 号前不能有空格。

2．查询提示强行并行：
要使用并行，需要PARALLEL参数，它有三个参数：数据表名、并行度及实例名。但在提示中使用PARALLEL就不需要关键字，如：
例1．当查询emp 表时，使用并行度为5：
```
select  /*+ PARALLEL(emp, 5)  */  *  from  EMP;
```

如果使用的环境是并行服务器，可以使用提示来规定在查询中涉及的实例的个数。如：
例2：设置并行度为5，用4个实例来处理该查询（实际是20路）：
```
select  /*+ PARALLEL(emp, 5, 4)  */  *  from emp;
```

如果想禁止并行查询，则用NOPARALLEL参数，如：
```
select   /*+  NOPARALLEL(emp)  */  * from emp;
```

§58.1.2.5 监视并行查询情况

1．在操作系统层：
在UNIX下，可以用(Oracle进程以ora开头)：
```
$ps  -ef|grep ora_p0
```

2．在Oracle系统内：
在Oracle数据库系统内，主要是几个以 V$PQ开头的数据字典，它们多是动态视图。并行信息主要是下面的四个视图：

- V$PQ_SYSSTAT 系统层对并行查询的统计资料
- V$PQ_SESSTAT 会话层对并行查询的统计资料
- V$PQ_SLAVE 实例中每一个活动的并行查询服务器的统计资料
- V$PQ_TQSTAT 会话期间所有并行查询和并行查询操作的统计资料。

§58.2 并行数据加载（SQL*Loader）

在 SQL*Loader 中可以使用并行操作来加快数据的加载操作。在 SQL*LOAD 中对并行的

使用主要是在 SQLLOAD 命令上进行说明。如：

SQLLOAD sqledba/sqledba control=sale1.ctl **parallel=TRUE direct=TRUE**
SQLLOAD sqledba/sqledba control=sale2.ctl **parallel=TRUE direct=TRUE**
SQLLOAD sqledba/sqledba control=sale3.ctl **parallel=TRUE direct=TRUE**

在这里用了 parallel=TRUE 和 direct=TRUE 来实现并行操作。

并行数据加载有下面特性：

- 每个 SQL*Loader 分配一个新的区间并将数据加载到该区间内。
- 并行数据加载要求不能有本地或全局索引。否则错误。
- 每个数据加载会话需要表的共享锁。
- 每个数据加载会话是独立的，在加载期间没有联系。
- 当一个加载完成后，Oracle 将该区间与现有的区间连在一起。

§58.3 并行恢复

Oracle 的基本操作单位是数据库块（数据库块的大小在安装时就被确定）。每当对数据库块进行修改时，Oracle 将这些改变以重做日志的形式记录下来，如果在需要时，可以使用日志来重建这些块。比如由于介质失败或任何其他的原因造成当前数据文件的内容丢失。则可以从适当的备份中重建，然后进行恢复。步骤如下：

1）读日志文件并获得数据块的修改序列号；
2）决定哪个数据块需要改变；
3）在高速缓冲区中读煊赫些数据块；
4）从重做日志文件中将相应的改变应用到这些数据块中；
5）将修改完的数据块写回到硬盘中。

为了在并行环境下进行恢复，需要设置参数 recovery_parallelism 参数为真。也可以在 RECOVERY 中带 PARALLEL 子句来达到。见《Oracle8i 数据库管理员实用教程》“数据库的启动”章节。

§58.4 并行 SQL 执行

Oracle 数据库是由不同进程维护的物理数据文件的集合及共享内存组成，称作实例（Instance）。这些后台进程可以使并发用户与数据库进行数据交换。当一个用户需要进行 select,update,delete,insert 时，他必须与数据库系统连接。而当用户连接成功时，系统派生一个附加的进程（叫影子进程或前台）。

*注意：在多线程（MTS）环境下，当用户注册到数据库时，一个共享服务器进程用于**代替**影子进程。而在并行查询操作中，这个共享服务器进程只用作**协调器**.*

由于 Oracle 处理是以块为单位，当用户需要数据时，影子进程就读取内存的高速缓存的数据块，然后由用户的影子进程来对该块进行修改，被修改过的块（脏缓存器）由 DBWR 进程写回硬盘。这样的操作在 OLTP 系统运行得可以，但在 DSS 系统中，由于用户需要处理大量的数据，影子进程完成的必须工作需要大量的时间。这时系统很可能有空闲的 CPU 与内存资源，这就是 Oracle 可以利用的并行操作的地方。通过设置选项，Oracle 可以将一个用户的大量数据处理分解成多个相对小的工作单元，这些单元可以由不同进程来并发执行。Oracle 使用一套专有后台进程（即并行查询服务器）来完成这些 工作。现在影子进程被称作查询协调器。它的职责如下：

1） 将功能分解为小的片。
2） 确包可以得到足够数量的并行查询伺服器。
3） 为给定的工作初始化伺服器，并并行伺服器间分派工作。
4） 完成预期的工作后，查询伺服器内释放。

§58.5 可以并行的 SQL 操作

当一条语句被执行时，Oracle 首先分析它然后才执行。在分析完和执行前期间，优化器建立了执行计划。查询协调器进程检查执行计划中的操作以决定单独的操作是否可以被优化。现在的 Oracle8 可以对下面的操作引入并行：

- 表扫描。
- 嵌套循环连接。
- 排序合并连接。
- 哈西连接。
- NOT IN
- Group By
- Select distinct
- UNION 与 UNION ALL
- 集合
- 从 SQL 中调用 PL/SQL 函数
- Order By
- Create table as select
- Create index
- Rebuild index
- Move index
- Split partition
- Update
- Delete
- Insert into .. select
- Enable 约束等。

§58.6 理解并行 DML

Oracle 在并行环境下可以进行并行的 Insert、Update、Delete。但要在会话中使用并行就要进行设置：

1.设置 DML 并行：

SQL> alter session enable parallel dml;

在并行 select 、Insert、Update、Delete 时，需要注意下面问题：

- PARALLEL

可以在 SQL 语句中使用 PARALLEL 来指定并行，Oracle8 用 PARALLEL 来为 select 、Insert、Update、Delete 指定并行度。语法如下：

PARALLEL (<table-name>, m, n)

M=SQL 操作的并行服务器的数目；n=执行 SQL 操作的实例数目

- NO PARALLEL

不进行并行操作，只需要顺序操作。

- APPEND

需要在 INSERT 操作进行并行操作，就要设置该参数（默认就是 APPEND）。它对 INSERT 的数据使用新的块，不使用现有空闲块。

- NOAPPEND

它对 INSERT 的数据使用现有空闲块。

- PARAELLE_INDEX

允许对分区索引进行并行扫描执行。

2.设置 DML 为非并行：

不需要将 DML 设置为并行，就要用在 ALTER SESSION 语句后加 DISABLE PARALLEL 子句。如：

ALTER SESSION disable parallel dml;

§58.7 并行创建数据库表和索引

如果在创建表和创建索引时使用并行也能加快处理速度。

§58.7.1 并行创建数据库表

在 CREATE TABLE xx AS SELECT 命令中可以用并行来加快处理速度。在默认下，这个命令是不使用并行的。

- 在串行（非并行）：

上面命令的 storage 子句是根据数据库表的 minextents 和 initial 来分配空间。而在并行下，每个查询服务器都分配并填充 minextents 区间。每个区间为 initial 指定的值。例如 initial 10m , 而 minextents 2 ，则：串行：创建两个区，每个为 10M

- 并行：

假设并行度为 4（即有 4 个查询服务器），则每个创建 2 个区间（因为 minextents 2），每个区间 10M。总共 80MB 。当查询协调器联合所有的这样创建和填满的区间时，它可能回收一些空间。从而在并行特性效益 和 附加的空间代价 之间进行平衡。

另外，由于 CREATE TABLE AS SELECT 命令要求是复制一个已存在的表，所以一般都不要恢复（不要回滚）。在 Oracle8i 以后的版本提供一个 nologing 子句来阻止写入重做日志文件中，从而提高速度。这样可以提高 30%至 50%左右。

例 1：一般的并行创建表

```
CREATE TABLE EMP2  PARALLEL(DEGREE 4)
AS  SELECT /*+ PARALLEL(EMP,4)  */
*  FROM  EMP;
```

例 2：不需要写日志的并行创建表

```
CREATE TABLE stock_xactions
(stock_symbol CHAR(5),
stock_series CHAR(1),
num_shares NUMBER(10),
price NUMBER(5,2),
trade_date DATE)
STORAGE (INITIAL 100K NEXT 50K) LOGGING
PARTITION BY RANGE (trade_date)
(PARTITION sx1992 VALUES LESS THAN (TO_DATE('01-JAN-1993','DD-MON-YYYY'))
TABLESPACE ts0 NOLOGGING,
PARTITION sx1993 VALUES LESS THAN (TO_DATE('01-JAN-1994','DD-MON-YYYY'))
TABLESPACE ts1,
PARTITION sx1994 VALUES LESS THAN (TO_DATE('01-JAN-1995','DD-MON-YYYY'))
TABLESPACE ts2);
```

§58.7.2 并行创建索引

与创建数据库表类似，也可以在创建索引时使用并行方式，即使用多个查询服务器一起工作，在处理中，由一组查询服务器同时扫描数据库表的各个部分，得到列值和该行的 ROWID 值。另外一组查询服务器进程根据列值对性进行排序，再由查询协调来组合结果，构成完整的索引结果。

例 1：建立一般的并行索引

```
CREATE INDEX I_LOC ON big_table ( okey )
NOSORT
NOLOGGING
PARALLEL;
```

例 2：建立带并行度的并行索引

```
CREATE INDEX inx_sale  on sales(product_id)
Nosort
Nologging
Parallel ( degree 5 ) ;
```

例 3：重新创建一个并行索引

```
ALTER INDEX idx_sale  REBUILD  nologging  parallel ( degree 5 );
```

§58.8 并行性能有关的视图

在并行环境中，由于所面对的多个数据库实例，所以要关心的视图是以“G”开头的。它与原来的名称类似，例如 V$FILESTAT 视图用 GV$FILESTAT 来代替。下面给出一般的视图名称及功能说明：

V$PX_SESSION

V$PX_SESSION 视图 显示关于查询服务器的会话、分组、序列和服务器的数量等。

V$PX_SESSTAT

V$PX_SESSTAT 视图显示会话统计信息。

V$PX_PROCESS

V$PX_PROCESS 视图包括并行进程的信息。包括状态、会话 ID、进程 ID 及其他信息等。

V$PX_PROCESS_SYSSTAT

V$PX_PROCESS_SYSSTAT 包括服务器状态，缓冲区统计等。

V$PQ_SESSTAT

V$PQ_SESSTAT 视图包括所有运行的服务器组的状态，如查询如何分配进程数据、多用户载入算法法等。

V$FILESTAT

V$FILESTAT视图总结了读取和写入的请求、块的数量以及每个表空间的数据文件的服务时间等。使用本视图可以查看 I/O 和工作的分布等。

V$PARAMETER

本参数是 Oracle 系统的一个综合参数，它包括系统的所有参数。

V$PQ_TQSTAT

V$PQ_TQSTAT 视图提供在表排队级的消息的详细流量。

第 60 章　高级安全管理

§120.1 Oracle 的高级安全

附录 A　动态性能视图(V$)

本附录介绍动态性能视图。这些视图一般作为 V$视图引用。本附录包括下列内容：

- 动态性能视图。
- 视图说明。

A.1 动态性能视图

Oracle 服务器包括一组基础视图，这些视图由服务器维护，系统管理员用户 SYS 可以访

问它们。这些视图被称为动态性能视图，因为它们在数据库打开和使用时不断进行更新，而且它们的内容主要与性能有关。

虽然这些视图很像普通的数据库表，但它们不允许用户直接进行修改。这些视图提供内部磁盘结构和内存结构方面的数据。用户可以对这些视图进行查询，以便对系统进行管理与优化。

文件 CATALOG.SQL 包含这些视图的定义以及公用同义词。必须运行 CATALOG.SQL 创建这些视图及同义词。

A.1.1 V$ 视图

动态性能视图由前缀 V_$标识。这些视图的公用同义词具有前缀 V$。数据库管理员或用户应该只访问 V$对象，而不是访问 V_$对象。

动态性能视图由企业管理器和 Oracle Trace 使用，Oracle Trace 是访问系统性能信息的主要界面。

建议： 一旦实例启动，从内存读取数据的 V$视图就可以访问了。从磁盘读取数据的视图要求数据库已经安装好了。

警告：给出动态性能视图的有关信息只是为了系统的完整性和对系统进行管理。公司并不承诺以后也支持这些视图。

A.1.2 GV$ 视图

在 Oracle 中，还有一种补充类型的固定视图。即 GV$（Global V$，全局 V$）固定视图。对于本章介绍的每种 V$ 视图（除 V$CACHE_LOCK、V$LOCK_ACTIVITY、V$LOCKS_WITH_COLLISIONS 和 V$ROLLNAME 外），都存在一个 GV$视图。在并行服务器环境下，可查询 GV$视图从所有限定实例中检索 V$视图的信息。除 V$信息外，每个 GV$视图拥有一个附加的名为 INST_ID 的整型列。INST_ID 列显示从其获得相关的 V$视图信息的实例号。INST_ID 列可用作一个从可得到的实例集检索 V$信息的过滤器。例如，下列查询：

SELECT * FROM GV$LOCK WHERE INST_ID＝2 OR INST_ID＝5

表示从实例 2 和 5 上的 V$ 视图中检索信息。

GV$视图可用来返回用 OPS_ADMIN_GROUP 参数定义的实例组上的信息。

GV$视图具有下列限制：

- 在安装数据库的所有实例上，PARALLEL_MAX_SERVERS 参数的值必须大于零。
- 为了成功完成查询，必须至少用一个成员来定义 OPS_ADMIN_GROUP 参数。

A.1.2 访问动态性能表

在安装之后，仅有用户名为 SYS 或具有 SYSDBA ROLE 的用户能够访问动态性能表。

A.2 视图说明

本节列出动态性能视图的列和公用同义词。

1．V$ACCESS

此视图显示数据库中当前锁定的对象及访问它们的会话。

列	数据类型	说明
SID	NUMBER	访问一个对象的会话号

OWNER	VARCHAR2(64)	对象的拥有者
OBJECT	VARCHAR2(1000)	对象号
TYPE	VARCHAR2(12)	对象的类型标识符

2．V$ACTIVE_INSTANCES

对所有当前使数据库安装的实例，此视图将实例名映射到实例号。

列	数据类型	说明
INST_NUMBER	NUMBER	实例号
INST_NAME	VARCHAR2(60)	实例名

3．V$AQ

此视图描述数据库中队列的统计数据。

列	数据类型	说明
QID	NUMBER	唯一的队列标识符
WAITING	NUMBER	队列中处于“WAITING”状态的消息号
READY	NUMBER	队列中处于“READY”状态的消息号
EXPIRED	NUMBER	队列中处于“EXPIRED”状态的消息号
TOTAL_WAIT	NUMBER	队列中处于“READY”消息的总等待时间
AVERAGE_WAIT	NUMBER	队列中处于“READY”消息的平均等待时间

4．V$ARCHIVE

此视图包含需要归档的重做日志文件的信息。每行提供一个线程的信息。这些信息在V$LOG 中也可得到。Oracle 建议使用 V$LOG。更多信息，请参阅“V$LOG”。

列	数据类型	说明
GROUP#	NUMBER	日志文件组号
THREAD#	NUMBER	日志文件线程号
SEQUENCE#	NUMBER	日志文件序列号
CURRENT#	VARCHAR2(3)	当前在使用的归档日志
FIRST_CHANGE#	NUMBER	存储在当前日志中的第一个 SCN

5． V$ARCHIVE_DEST

对于当前实例,此视图描述所有归档日志目标、它们的当前值、模式以及状态。

列	数据类型	说明
DEST_ID	NUMBER	ID(1-5)
STATUS	VARCHAR2(9)	状态：VAILID：初始化并可得到； INACTIVE：无目标信息； DEFERRED：用户手工禁用； ERROR：打开或拷贝中出错； DISABLED：出错后禁用； BAD PARAM：参数有错；
BINDING	VARCHAR2(9)	成功请求：MANDATORY-必须成功，否则 OPTIONAL-不需要成功（依赖于 LOG_ARCHIVE_MIS_SUCCEED_DEST）

NAME_SPACE	VARCHAR2(7)	定义范围：SYSTEM-系统定义或 SESSION-会话定义
TARGET	VARCHAR2(7)	目标：PRIMARY-拷贝到主目标或 STANDBY-拷贝到备用目标
REOPEN_SECS	VARCHAR2(7)	按秒计算的重试时间（出错之后）
DESTINATION	VARCHAR2(256)	目标文本串（转换为主位置或备用服务器名）
FAIL_DATE	DATE	最后出错的日期和时间
FAIL_SEQUENCE	NUMBER	最后出错的日志序列号
FAIL _BLOCK	NUMBER	最后出错的块号
ERROR	VARCHAR2(256)	最后出错的文本

关于归档日志目标,请参阅 LOG_ARCHIVE_DEST、LOG_ARCHIVE_DUPLEX_DEST、LOG_ARCHIVE_DEST_n、LOG_ARCHIVE_DEST_STATE_n、STANDBY_ARCHIVE_DEST 和 LOG_ARCHIVE_MIN_SUCCEED_DEST 。

6. V$ARCHIVE_LOG

此视图从包含归档日志名的控制文件中显示归档日志信息。在联机重做日志成功地归档或清除后插入的（如果日志被清除，则名称列为 NULL）后，插入一个归档日志记录。如果该日志归档两次，将有两个具有 THREAD#、SEQUENCE#、FIRST_CHANGE#但名称不同的归档日志记录。在从备份集或拷贝重新存储归档日志时，也插入一个归档日志记录。

列	数据类型	说明
RECID	NUMBER	归档日志记录 ID
STAMP	NUMBER	归档日志记录时间戳
NAME	VARCHAR2(512)	归档日志文件名。如果设置为 NULL，则日志文件在被归档前清除
THREAD#	NUMBER	重做线程号
SEQUENCE#	NUMBER	重做日志序列
RESETLOGS_ CHANGE#	NUMBER	在写入此日志时重置数据库的日志更改号
RESETLOGS_TIME	DATE	在写入此日志时重置数据库的日志时间
FIRST_CHANGE#	NUMBER	归档日志中的第一个更改号
FIRST_TIME	NUMBER	第一个更改的时间戳
NEXT_CHANGE#	NUMBER	下一日志中的第一个更改
NEXT_TIME	NUMBER	下一个更改的时间戳
BLOCKS	NUMBER	以块表示的归档日志大小
BLOCK_SIZE	NUMBER	重做日志块的尺寸
ARCHIVED	VARCHAR2(3)	YES/NO
DELETED	VARCHAR2(3)	YES/NO
COMPLETION_TIME	DATE	归档完成时间

7. V$ARCHIVE_PROCESSES

此视图提供实例的各种 ARCH 进程的状态信息。

列	数据类型	说明
PROCESS	NUMBER	实例的 ARCH 进程标识符，编号从 0 至 9
STATUS	VARCHAR2(10)	ARCH 进程的状态，显示为一个关键字。

		可能的值为：STOPPED、SCHEDULED、STARTING、ACTIVE、STOPPING 和 TERMINATED
LOG_SEQUENCE	NUMBER	如果 STATE="BUSY"，则这是当前归档的联机重做日志序号
STATE	VARCHAR2(4)	这是 ARCH 进程的当前状态，显示为一个关键字。可能的关键字为：IDLE 和 BUSY

8. V$BACKUP

此视图显示所有联机数据文件的备份状态。

列	数据类型	说明
FILE#	NUMBER	文件标识符
STATUS	VARCHAR2(18)	文件状态：NOT ACTIVE、ACTIVE(正在进行备份)、OFFINE NORMAL 或一个错误说明
CHANGE#	NUMBER	备份开始时的系统更改号
TIME	DATE	备份开始时间

9. V$BACKUP_ASYNC_IO

此视图显示控制文件中的备份集信息。在成功地完成备份集时，插入一个备份集记录。

列	数据类型	说明
SID	NUMBER	进行备份或恢复的会话的 Oracle SID
SERIAL	NUMBER	进行备份或恢复的 SID 的使用计数
USE_COUNT	NUMBER	可用来标识来自不同备份集的行计数器
DEVICE_TYPE	VARCHAR2(17)	放置文件的设备类型
TYPE	VARCHAR2(9)	INPUT、OUTPUT、AGGREGATE
STATUS	VARCHAR2(11)	NOT STARTED；IN PROGRESS；FINISHED
FILENAME	VARCHAR2(513)	被读取或写入的备份文件名
SET_COUNT	NUMBER	被读取或写入的备份集的集计数
SET_STAMP	NUMBER	被读取或写入的备份集的集时间戳
BUFFER_SIZE	NUMBER	用来读写这个文件的缓冲区的尺寸
BUFFER_COUNT	NUMBER	用来读写这个文件的缓冲区的数量
TOTAL_BYTES	NUMBER	如果知道，为将对这个文件进行读写的字节总数。如果不知道，此列为空
OPEN_TIME	DATE	文件打开的时间。如果 TYPE='AGGREGATE'，则这是聚集中第一个文件打开的时间
CLOSE_TIME	DATE	文件关闭的时间。如果 TYPE=AGGREGATE'，则这是聚集中第一个文件关闭的时间
ELAPSED_TIME	NUMBER	文件打开的时间，以百分之一秒计
MAXOPENFILES	NUMBER	同时打开的磁盘文件数。这个值仅在 TYPE='AGGREGATE'的行中给出
BYTES	NUMBER	迄今为止读写的字节数
BFFECTIVE_BYTES_PER_	NUMBER	在这个备份中用这个设备归档的 I/O 率

SECOND		
IO_COUNT	NUMBER	对这个文件执行的 I/O 数
READY	NUMBER	缓冲区立即作好使用准备的异步请求数
SHORT_WAITS	NUMBER	缓冲区不立即可用，但缓冲区在进行 I/O 完成的非阻塞轮询后的可用次数
SHORT_WAIT_TIME_TOTAL	NUMBER	I/O 完成的非阻塞轮询所用的时间总数，以百分之一秒计
SHORT_WAIT_TIME_MAX	NUMBER	I/O 完成的非阻塞轮询所用的最大时间数，以百分之一秒计
LOG_WAITS	NUMBER	缓冲区不立即可用，缓冲区仅在进行 I/O 完成的非阻塞轮询后可用的次数
LOG_WAITS	NUMBER	缓冲区不立即可用，缓冲区仅在进行 I/O 完成的非阻塞轮询后可用的次数
LOG_WAITS_TIME_TOTAL	NUMBER	I/O 完成的阻塞等待所有的时间总数，以百分之一秒计
LOG_WAITS_TIME_MAX	NUMBER	I/O 完成的阻塞等待所有的最大时间数，以百分之一秒计

10. V$BACKUP_CORRUPTION

此视图显示来自控制文件的数据文件备份中出错的相关信息。注意，在控制文件和归档日志备份中是不容许出错的。

列	数据类型	说明
RECID	NUMBER	备份出错的记录 ID
STAMP	NUMBER	备份出错的记录时间戳
SET_STAMP	NUMBER	备份集时间戳
SET_COUNT	NUMBER	备份集计数
PIECE#	NUMBER	备份的片号
FILE#	NUMBER	数据文件号
BLOCK#	NUMBER	出错范围中的第一块
BLOCKS	NUMBER	出错范围中的邻接块数
CORRUPTION_CHANGE#	NUMBER	检查到逻辑错的更改号。设置为 0 表示介质错
MARKED_ CURRUPT	VARCHAR2(3)	YES/NO。如果设置为 YES，则在数据文件中不标记块出错，而在进行数据文件备份时检查并标记

11. V$BACKUP_DATAFILE

此视图显示来自控制文件的备份数据文件和备份控制文件。

列	数据类型	说明
RECID	NUMBER	备份数据文件记录 ID
STAMP	NUMBER	备份数据文件记录时间戳
SET_STAMP	NUMBER	备份集时间戳
SET_COUNT	NUMBER	备份集计数
FILE#	NUMBER	数据文件号
CREATION_CHANGE#	NUMBER	数据文件的创建更改
CREATE_TIME	DATE	数据文件的创建时间戳
RESETLOGS_CHANGE#	NUMBER	数据文件备份时的重置日志更改号

RESETLOGS_TIME#	DATE	数据文件备份时的重置日志时间戳
INCREMENTAL_LEVEL	NUMBER	(0~4) 个增量备份级
INCREMENTAL_ CHANGE#	NUMBER	增量更改号包含在这个备份中后更改的所有块。全备份设置为 0
CHECKPOINT_ CHANGE#	NUMBER	直到检查点更改号的所有更改都包含在此备份中
CHECKPOINT_TIME	DATE	检查点时间戳
ABSOLUTE_FUZZY_ CHANGE#	NUMBER	此备份中的最高更改号
MARKED_CORRUPT	NUMBER	标记出错的块数
MEDIA_CORRUPT	NUMBER	介质出错的块数
LOGICALLY_CORRUPT	NUMBER	逻辑出错的块数
DATAFILE_BLOCKS	NUMBER	备份时按块计的数据文件的尺寸。这个值也是从这个备份重新开始时数据文件占用的块数
BLOCKS	NUMBER	数据文件以块计的尺寸。未用块不拷贝到备份
BLOCKS	NUMBER	块尺寸
OLDEST_OFFLINE_ RANGE	NUMBER	此备份控制文件中最旧的脱机范围记录的 RECID。对于数据文件为 0
COMPLETION_TIME	DATE	完成时间

12. V$BACKUP_DEVICE

这个视图显示支持设备的设备的有关信息。如果某种设备类型不支持指定的设备，则返回该设备的一个带设备类型和空设备的行。如果某种设备类型支持指定的设备，则为该类型的每个可用设备返回一行。这个视图不返回特殊的设备类型 DISK，因为它总是可用的。

列	数据类型	说明
DEVICE_TYPE	VARCHAR2(17)	备份设备的类型
DEVICE_NAME	VARCHAR2(512)	备份设备的名称

13. V$BACKUP_PIECE

这个视图显示来自控制文件的备份片的相关信息。每个备份集由一个或多个备份片组成。

列	数据类型	说明
RECID	NUMBER	备份片记录 ID
STAMP	NUMBER	备份片记录时间戳
SET_STAMP	NUMBER	备份集时间戳
SET_COUNT	NUMBER	备份集计数
PIECE#	NUMBER	备份片号（1~n）
COPY#	NUMBER	确定用允许双工创建的备份片的拷贝号。如果备份片不是双工的，为 1
DEVICE_TYPE	VARCHAR2(17)	备份片驻留的设备类型。备份组在磁盘上设置为 DISK。请参 V$BACKUP_DEVICE
HANDLE	VARCHAR2(513)	备份片句柄确定正在恢复的备份片
COMMENTS	VARCHAR2(81)	操作系统或存储子系统返回的注

		释。对于磁盘上的备份片设置为NULL。这个信息只是提示性的；恢复时不需要
MEDIA	VARCHAR2(65)	备份片驻留其上的介质数。这个值是提示性的；恢复时不需要
MEDIA_POOL	NUMBER	备份片驻留在其中的介质池。这是一个与 Recovery Manager 的 BACKUP 命令的 POOL 操作数中输入的值相同的值
CONCUR	VARCHAR2(3)	YES/NO，确定介质上的片是否可并发地访问
TAG	VARCHAR2(32)	备份片标记。在备份集级指定这个标记，但在片级存储
DELETED	VARCHAR2(3)	如果设置为 YES，表示片被删除，否则设置为 NO
START_TIME	DATE	开始时间
COMPLETION_TIME	DATE	完成时间
ELAPSED_SECONDS	NUMBER	占用的秒数

14. V$BACKUP_REDOLOG

此视图显示来自控制文件的备份集中归档日志的信息。注意，联机重做日志不能直接备份；它们必须首先归档到磁盘，然后再备份。一个归档日志备份集可包含一个或多个归档日志。

列	数据类型	说明
RECID	NUMBER	此行的记录 ID。它是一个标识此行的整数
STAMP	NUMBER	RECID 唯一地标识此行所用的时间戳
SET_STAMP	NUMBER	备份集时间戳
SET_COUNT	NUMBER	标识这个备份集的 V$BACKUP_SET 表的行外部键之一
THREAD#	NUMBER	日志的线程号
SEQUENCE#	NUMBER	日志序列号
RESETLOGS_CHANGE#	NUMBER	在写入前的最后重置日志的更改号
RESETLOGS_TIME	DATE	在日志写入前的最后重置日志的更改时间。同一备份集中所有日志的这个值都是相同的
FIRST_CHANGE#	NUMBER	在将日志切换入时的 SCN。日志中的重做是在此 SCN 或更大进行的
FIRST_TIME	DATE	切换入日志时所分配的时间
NEXT_CHANGE#	NUMBER	切换出日志时的 SCN。日志中的重做低于此 SCN
NEXT_TIME	DATE	切换出日志时分配的时间
BLOCKS	NUMBER	逻辑块中的日志尺寸，包括标题块
BLOCK_SIZE	NUMBER	以字节表示的日志块尺寸

15. V$BACKUP_SET

此视图显示来自控制文件的备份集信息。在成功完成备份集后，插入一个备份集记录。

列	数据类型	说明
RECID	NUMBER	备份集记录 ID。
STAMP	NUMBER	备份集记录的时间戳
ET_STAMP	NUMBER	备份集时间戳。备份集时间戳和计数唯一标识备份集 V$BACKUP_PIECE； V$BACKUP_DATAFILE； V$BACKUP_REDOLOG； V$BACKUP_CORRUPTION；
SET_COUNT	NUMBER	备份集计数器。备份集计数每当开始一个新备份集时加 1（如果备份集永不结束则此数会丢失）。如果重新创建控制文件，则此计数重置为 1。因此此计数必须与唯一标识一个备份集的时间戳一起使用 V$BACKUP_PIECE 表的主键，以及下列表的外部键： V$BACKUP_PIECE； V$BACKUP_DATAFILE； V$BACKUP_REDOLOG； V$BACKUP_CORRUPTION
BACKUP_TYPE	VARCHAR2(1)	此备份中的文件类型。如果此备份含有归档重做日志，则为‘L’。如果这是一个数据文件完全备份，则值为‘D’。如果这是一个增量备份，则值为‘I’。
CONTROLFILE_INCLUDED	VARCHAR2(3)	如果此备份集中含有一个控制文件，则设置为 YES，否则设置为 NO
INCREMENTAL_LEVEL	NUMBER	此备份集适合于数据库备份策略的位置。对完全的数据文件备份设置为零，对增量数据文件备份设置为非零，而对归档日志备份设置为 NULL
PIECES	NUMBER	备份集中不同备份片的数目
START_TIME	DATE	开始时间
COMPLETION_TIME	DATE	在成功完成备份时，这是备份集的完成时间。这也是由 backupEnd 返回的相同时间。如果备份正在进行中或已经失败，则设置为 NULL
ELAPSED_SECONDS	NUMBER	所占用的秒数
BLOCK_SIZE	NUMBER	备份集的块尺寸

16. V$BACKUP_SYNC_IO

此视图显示来自控制文件的备份集的信息。在备份集成功完成之后，插入一个备份集记录。

列	数据类型	说明

SID	NUMBER	进行备份或恢复的会话的 Oracle SID
SERIAL	NUMBER	进行备份或恢复的 SID 的使用计数
USE_COUNT	NUMBER	可用来标识来自不同备份集的行计数器
DEVICE_TYPE	VARCHAR2(17)	放置文件的设备类型
TYPE	VARCHAR2(9)	INPUT、OUTPUT、AGGREGATE
STATUS	VARCHAR2(11)	NOT STARTED、IN PROGRESS; FINISHED
FILENAME	VARCHAR2(512)	被读取或写入的备份文件名
SET_COUNT	NUMBER	被读取或写入的备份集的集计数
SET_STAMP	NUMBER	被读取或写入的备份集的集时间戳
BUFFER_SIZE	NUMBER	用来读写这个文件缓冲区的尺寸
BUFFEER_COUNT	NUMBER	用来读写这个文件缓冲区的数量
TOTAL_BYTES	NUMBER	如果知道，为将对这个文件进行读写的字节总数。如果不知道，此列为空
OPEN_TIME	DATE	文件打开的时间。如果 TYPE='AGGREGATE'，则这是聚集中第一个文件打开的时间
CLOSE_TIME	DATE	文件关闭的时间。如果 TYPE='AGGREGATE'，则这是聚集中第一个文件打开的时间
ELAPSED_TIME	DATE	文件打开时间，以百分之一秒计
MAXOPENFILES	NUMVER	同时打开的磁盘文件数。这个值仅在 TYPE='AGGREGATE'的行中同时给出
BYTES	NUMBER	迄今为止读写的字节数
BFFECTIVE_BYTES_PER_SECOND	NUMBER	在这个备份中用这个设备归档的 I/O 率
IO_COUNT	NUMBER	对这个文件执行的 I/O 数
I/O_TIME_TOTAL	NUMBER	进行此文件的 I/O 所占的时间总数，以百分之一秒计
I/O_TIME_MAX	NUMBER	单个 I/O 请求所用的最大时间
DISCRETE_BYTES PER_SECOND	NUMBER	这个文件的平均传输率

17. V$BGPROCESS

此视图描述后台进程。

列	数据类型	说明
PADDR	RAW(4)	进程状态对象的地址
NAME	VARCHAR2	后台进程的名称
DESCRIPTION	VARCHAR2	后台进程的说明
ERROR	NUMBER	所遇到的错误

18. V$BH

这是一个并行服务器视图。这个视图给出 SGA 中每个缓冲区的 ping 状态和数目。

列	数据类型	说明

FILE#	NUMBER	数据文件标识号（为找到文件名，可查询 DB_DATA_FILES 或 V$DBFILES）
BOLCK#	NUMBER	块号
CLASS#	NUMBER	类号
STATUS	VARCHAR2(1)	FREE=当前不用 XCUR=互斥的 SCUR=当前共享 CR=一致的读取 READ=从磁盘读取 MREC=处于介质恢复模式 IREC=处于实例恢复模式
XNC	NUMBER	由于与其他实例争用导致的空锁变换的 PCMx 数。此列已作废，但为了历史兼容性仍然保留
LOCK_ELEMENT_ADDR	RAW(4)	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT_NAME	NUMBER	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT_CLASS	NUMBER	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
FORCED_READS	NUMBER	锁必须从磁盘上读取的次数，重新读取是由于其他实例通过在锁模式中请求此锁上的 PCM 锁，强迫它退出了此实例的高速缓存
FORCED_WRITES	NUMBER	由于这个实例已经搞坏了这个块并且其他实例已经以冲突的模式请求了这个锁上的 PCM 锁，而导致 DBWR 必须将这个块写入磁盘的次数
DIRTY	VARCHAR(1)	Y=修改过的块
DIRTY	VARCHAR(1)	Y=临时块
DIRTY	VARCHAR(1)	Y=ping 过的块
DIRTY	VARCHAR(1)	Y=块是陈旧的
DIRTY	VARCHAR(1)	Y=直接块
DIRTY	VARCHAR(1)	总是设置为 N。此列已废弃，但为了历史兼容而保留
OBJD	NUMBER	缓冲区代表的块的数据库对象数
TS#	NUMBER	块的表空间数

19. V$BUFFER_POOL

此视图显示实例可用的所有缓冲池的相关信息。这个“设施”适合于 LRU 栓锁组的数目。更多的信息，请参阅“DB_BLOCK_LRU_LATCHES”。

列	数据类型	说明

ID	NUMBER	缓冲池 ID 号
NAME	VARCHAR2(20)	缓冲池名称
LO_SETID	NUMBER	低设置 ID 号
HI_SETID	NUMBER	高设置 ID 号
SET_COUNT	NUMBER	这个缓冲池中的设置数，为 HI_SETID_LO_SETID+1
BUFFERS	NUMBER	分配给缓冲池的缓冲区数
LO_BNUM	NUMBER	本缓冲池的低缓冲区号
HI_BNUM	NUMBER	本缓冲池的高缓冲区号

20．V$BUFFER_POOL_STATISTICS

此视图显示事例可用的所有缓冲池的相关信息。这个“设施”适合于 LRU 栓锁组的数目。更多的信息，请参阅“DB_BLOCK_LRU_LATCHES”。

列	数据类型	说明
ID	NUMBER	缓冲池 ID 号
NAME	VARCHAR2(20)	缓冲池名称
SET_MSIZE	NUMBER	缓冲池最大设置尺寸
CNUM_REPL	NUMBER	替换列表中的缓冲区数
CNUM_WRITE	NUMBER	写入列表中的缓冲区数
CNUM_SET	NUMBER	设置中的缓冲区数
BUF_GOT	NUMBER	设置获得的缓冲区数
SUM_WRITE	NUMBER	设置写入的缓冲区数
SUM_SCAN	NUMBER	设置扫描的缓冲区数
FREE_BUFFER_WAIT	NUMBER	可用缓冲区等待统计数据
WRITE_COMPLETE_WAIT	NUMBER	写完成等待统计数据
BUFFER_BUSY_WAIT	NUMBER	缓冲区忙等待统计数据
RFEE_BUFFER_INSPECTED	NUMBER	可用缓冲区检查统计数据
DIRTY_BUFFER_INSPECTED	NUMBER	灰缓冲区检查统计数据
DB_BLOCK_CHANGE	NUMBER	数据块更改统计数据
DB_BLOCK_GETS	NUMBER	取得数据库块统计数据
CONSISENT_GETS	NUMBER	一致取统计数据
PHYSICAL_READS	NUMBER	物理读统计数据
PHYSICAL_WRITES	NUMBER	物理写统计数据

21．V$CACHE

这是一个并行服务器视图。此视图包含来自当前实例的 SGA 中每个块的块标题的信息，这些信息涉及特定的数据库对象。

列	数据类型	说明
FILE#	NUMBER	数据文件标识号（为找到文件名，可查询 DB_DATA_FILES 或 V$DBFILES）
BOLCK#	NUMBER	块号
CLASS#	NUMBER	类号
STATUS	VARCHAR2(1)	FREE=当前不用

		XCUR=互斥的 SCUR=当前共享 CR=一致的读取 READ=从磁盘读取 MREC=处于介质恢复模式 IREC=处于实例恢复模式
XNC	NUMBER	由于与其他实例争用导致的空锁变换的 PCMx 数。此列已作废，但为了历史兼容性仍然保留
FORCED_READS	NUMBER	强制读取
FORCED_WRITES	NUMBER	强制写入
NANE	VARCHAR2(30)	包含该块的数据库对象名
PARTITION_NAME	VARCHAR2(30)	分区名；非分区对象为 NULL
KIND	VARCHAR2930)	数据库对象名。请参阅表 B-1
OWNER#	NUMBER	拥有者
LOCK_ELEMENT_ADDR	RAW(4)	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT	NUMBER	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区@@@@@

表 B-1 KIND 列的值

类型号	KIND 值	类型号	KIND 值
1	index	11	PACKAGE BODY
2	TABLE	12	TRIGGER
3	CLUSTER	13	TYPE
4	VIEW	14	TYPE BODY
5	SYNONYM	19	TABLE PARTITION
6	SEQUENCE	20	INDEX PARTITION
7	PROCEDURE	21	LOB
8	FUNCTION	22	LIBRARY
9	PACKAGE	NULL	UNKNOWN
10	NON_EXISTENT	……	……

22．V$CACHE_LOCK

这是一个并行服务器视图

列	数据类型	说明
FILE#	NUMBER	数据文件标识号（为找到文件名，可查询 DB_DATA_FILES 或 V$DBFILES）
BOLCK#	NUMBER	块号

STATUS	VARCHAR2(1)	FREE=当前不用 XCUR=互斥的 SCUR=当前共享 CR= 一致的读取 READ=从磁盘读取 MREC=处于介质恢复模式 IREC=处于实例恢复模式
XNC	NUMBER	由于与其他实例争用导致的并行高速缓存管理锁变换的数目
NANE	VARCHAR2(30)	包含该块的数据库对象名
PARTITION_NAME	VARCHAR2(30)	分区名；非分区对象为 NULL
KIND	VARCHAR2(30)	数据库对象名。请参阅表 B-1
OWNER#	NUMBER	拥有者号
LOCK_ELEMENT_ADDR	RAW(4)	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT_NAME	NUMBER	包含覆盖缓冲区的 PCM 锁的锁元素的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
FORCED_READS	NUMBER	锁必须从磁盘读取次数，重新读取是由于其他实例通过在锁模式中请求此锁上的 PCM 锁，强迫它退出了此实例的高速缓存
FORCED_WRITES	NUMBER	由于这个实例已经搞坏了这个块并且其他实例已经以冲突的模式请求了这个锁上的 PCM 锁，而导致 DBWR 必须将这个块写入磁盘的次数
INDX	NUMBER	平台专用的锁管理程序标识符
CLASS	NUMBER	平台专用的锁管理程序标识符

除了平台专用的锁管理标识符外，V$CASHE_LOCK 与 V$CACHE 类似。如果平台专用的锁管理程序提供了监控正在进行的 PCM 锁操作的工具，则这个信息是很有用的。例如，前一个查询利用 INDX 和 CLASS 找到了锁元素的地址，然后查询 V$BH 被锁覆盖的缓冲区。请参阅“V$CACHE”。

23．V$CIRCUIT

该视图包含虚电路的有关信息，虚电路是通过调度程序和服务器对数据库的用户连接。

列	数据类型	说明
CIRCUIT	RAW(4)	虚电路地址
DISPATCHER	RAW(4)	虚电路调度程序进程地址
SERVER	RAW(4)	虚电路服务器进程地址
WAITER	RAW(4)	等待（当前忙）虚电路可用的服务器进程地址
SADDR	RAW(4)	绑定到虚电路的会话地址
STATUS	RAW(4)	虚电路的状态：BREAK(当前中断)，EOF(将要被删除)，

		OUTBOUND(向外连接到远程数据库)，NORMAL(正常进入本地数据库的虚电路)
QUEUE	VARCHAR2(16)	虚电路当前所在的队列：COMMON(在公共队列上，等待被某个服务器进程选取)、DISPATCHER(等待调度程序)、SERVER(当前接受服务)、NONE(空闲虚电路)
MESSAGE0	NUMBER	以字节表示的第一个消息缓冲区中消息的大小
MESSAGE1	NUMBER	以字节表示的第二个消息缓冲区中消息的大小
MESSAGE2	NUMBER	以字节表示的第三个消息缓冲区中消息的大小
MESSAGE3	NUMBER	以字节表示的第四个消息缓冲区中消息的大小
MESSAGES	NUMBER	已通过此虚电路的消息总数
BYTES	NUMBER	已通过此虚电路的字节总数
PRESENTATION	NUMBER	此虚电路的断开（中断）数

24．V$CLASS_PING

V$CLASS_PING 显示每个块类 ping 的块数。可使用此视图在不同类中比较块的争用情况。

列	数据类型	说明
CLASS	NUMBER	表示块类别的编号
X_2_NULL	NUMBER	对指定 CLASS 的所有块，Exclusive_to_NULL 的块转换数
X_2_NULL_ FORCED_WRITE	NUMBER	对指定 CLASS 的所有块，由于 Exclusive_to_NULL 转换进行强制写的数目
X_2_NULL_ FORCED_STATE	NUMBER	CLASS 中的块，由于 Exclusive_to_NULL 转换而变陈旧的次数
X_2_S	NUMBER	指定 CLASS 的所有块的 Exclusive_to_Shared 锁转换数目
X_2_S_ FORCED_WRITE	NUMBER	对指定 CLASS 的块，由于 Exclusive_to_Shared 转换而出现的强制写的数目
X_2_SSX	NUMBER	指定 CLASS 的所有块，Exclusive_to_Sub Shared Exclusive 锁转换的数目
X_2_SSX_ FORCED _WRITE	NUMBER	对指定 CLASS 的块，由于 Exclusive_to_Sub Shared Exclusive 转换而出现的强制写的数目
S_2_NULL	NUMBER	对指定 CLASS 的所有块，Share_to_NULL 锁转换的数目
S_2_NULL_ FORCED_STATE	NUMBER	CLASS 中的块，由于 Share_to_NULL 转换而陈旧的数

		目
SS_2_NULL	NUMBER	对于指定 CLASS 的所有块 Sub Shared_to_NULL 锁转换的数目
S_2_X	NUMBER	对于指定 CLASS 的所有块，Shared_to_NULL 锁转换的数目
SSX_2_X	NUMBER	对于指定 CLASS 的所有块，Sub SharedExclusive_to_Exclusive 锁转换的数目
NULL_2_S	NUMBER	对于指定 CLASS 的所有块，NULL_to_Shared 锁转换的数目
NULL_2_SS	NUMBER	对于指定 CLASS 的所有块，NULL_to_Sub Shared 锁转换的数目

25．V$COMPATIBILITY

此视图显示数据库实例正在使用的特性，以防止性能降为以前的版本。这是此信息的动态（SGA）板，并不反映其他实例已经使用的特性，并有可能包含临时的不兼容（如 UNDO 段），这种不兼容在数据库完全关闭后就不存在了。

列	数据类型	说明
TYPE_ID	VARCHAR2(8)	内部特性标识符
RELEASE	VARCHAR2(60)	发布该特性的版本
DESCRIPTION	VARCHAR2(64)	特性描述

26．V$COMPATSEG

此视图列出数据库实例正在使用的永久特性，以防止返回到早期版本。

列	数据类型	说明
TYPE_ID	VARCHAR2(8)	内部特性标识符
RELEASE	VARCHAR2(60)	发布该特性的版本。
UPDAED	VARCHAR2(60)	首先使用此特性的版本

27．V$CONTEXT

此视图列出当前会话中设置的属性

列	数据类型	说明
NAMESPACE	VARCHAR2(30)	名称空间名
ATTRIBUTE	VARCHAR2(30)	属性名
VALUE	VARCHAR2(64)	属性值

28．V$CONTROLFILE

这个视图列出控制文件名

列	数据类型	说明
STATUS	VARCHAR2(7)	如果不能确定名称（这是不应该发生的），则为

		INVALID。如果可以确定名称，为 NULL
NAME	VARCHAR2(257)	控制文件名

29．V$CONTROLFILE_RECORD_SECTION

这个视图显示控制文件记录部分的相关信息

列	数据类型	说明
TYPE	VARCHAR2(7)	DATABASE/CKPT_PROGRESS/REDO THREAD/REDO LOG/DATAFILE/FILENAME/TABLESPACE/LOG HISTORY/OFFLINE RANGE/ARCHIVED LOG/BACKUP SET/BACKUP PIECE/BACKUP DATAFILE/BACKUP REDOLOG/DATAFILE COPY/BACKUP CORRUPTION/COPY CORRUPTION/DELETED OBJECT
RECORD_SIZ	NUMBER	以字节表示的记录尺寸
RECORD_TOTAL	NUMBER	为该部分分配的记录数
RECORD_USED	NUMBER	该部分中使用的记录数
FIRST_INDEX	NUMBER	第一个记录的索引（位置）
LAST_INDEX	NUMBER	最后一个记录的索引
LAST_RECID	NUMBER	最后一个记录的记录 ID

30．V$COPY_CORRUPTION

这个视图显示来自控制文件的数据文件拷贝出错的相关信息。

列	数据类型	说明
RECID	NUMBER	拷贝出错的记录 ID
STAMP	NUMBER	拷贝出错的记录时间戳
COPY_ RECID	NUMBER	数据文件拷贝记录 ID
COPY_STAMP	NUMBER	数据文件拷贝记录时间戳
FILE#	NUMBER	数据文件号
BLOCK#	NUMBER	出错范围的每一个块
BLOCKS	NUMBER	出错范围中的邻接块数
CORRUPTION_CHANGE#	NUMBER	检测到逻辑错的更改号。设置为 0 表示介质出错
MARKED_CORRUPT#	VARCHAR2(3)	YES/NO。如果设置为 YES，则数据文件中不标记出错块，但在进行数据文件拷贝时进行检测和标记

31．V$DATABASE

这个视图包含来自控制文件的数据库信息

列	数据类型	说明
DBID	NUMBER	数据库 ID
NAME	VARCHAR2	数据库名
CREATED	DATE	创建日期
LOG_MODE	VARCHAR2	归档日志模式：NOARCHIVELOG 或 ARCHIVELOG
CHECKPOINT_CHANGE#	NUMBER	最后一个 SCN 检查点
ARCHIVE_CHANGE#	NUMBER	归档的最后一个 SCN
DBID	NUMBER	在所有文件标题中创建和存取数据库时，计算的数据库 ID
RESETLOGS_CHANGE#	NUMBER	打开重置日志时的更改号
RESETLOGS_TIME	DATE	打开重置日志的时间戳
PRIOR_RESETLOGS_CHANGE#	NUMBER	以前重置日志时的更改号
PRIOR_RESETLOGS_TIME	DATE	以前重置日志的时间戳
CONTROLFILE_TYPE	VARCHAR2(9)	CURRENT/STANDBY/CLONE/BACKUP/CREATED。STANDBY 表示数据库处于备用状态。CLONE 表示一个克隆数据库 BUCKUP/CREATED 表示正利用备份或创建的控制文件恢复数据库。在恢复更改类型为 CURRENT 后备用数据库启动或数据库打开
CONTROLFILE_CREATED	DATE	控制文件创建时间戳
CONTROLFILE_SEQUENCE#	NUMBER	由控制文件事务处理增加的控制文件序列号
CONTROLFILE_CHANGE#	NUMBER	备份控制文件事务中的最后更改号。如果控制文件未备份，则设置为 NULL
CONTROLFILE_TIME	DATE	备份控制文件事务中的最后时间戳。如果控制文件未备份，则设置为 NULL
OPEN_RESETLOGS	VARCHAR2(11)	NOT ALLOWED/ALLOWED/REQUIRED 指出下一次数据库打开是否允许或需要重置日志选项
VERSION_TIME	DATE	版本时间
OPEN_MODE	VARCHAR2(10)	打开模式的信息

32．V$DATAFILE

此视图含有来自控制文件的数据文件信息。还可以参阅“V$DATAFILE_HEADER”视图，该视图显示来自数据文件标题的信息。

列	数据类型	说明
FILE#	NUMBER	文件标识号

STATUS	VARCHAR2	文件（系统或用户）类型极其状态。值：OFFLINE、ONLINE、SYSTEM、RECOVER。SYSOFF（来自系统表空间的脱机文件）
ENABLED	VARCHAR2(10)	描述怎样从 SQL 访问文件。取值为表 B-1 中所示的任一值
CHECKPOINT_CHANGE#	NUMBER	最后一个检查点的 SCN
CHECKPOINT_TIME	DATE	检查点时间戳
UNRECOVERABLE_CHANGE#	NUMBER	对这个文件所做的最后一个不可恢复的更改号。此列总是在一个不可恢复的操作完成后更新
UNRECOVERABLE_TIME	DATE	最后一个不可恢复更改的时间戳
BYTES	NUMBER	以字节计的当前尺寸；如果不可访，则为 0
CREATE_BYTES	NUMBER	创建的尺寸，以字节计
NAME	VARCHAR2	文件名
CREATION_CHANGE#	NUMBER	创建数据文件的更改号
CREATION_TIME	DATE	创建数据文件的时间戳
TS#	NUMBER	表空间号
RFLE#	NUMBER	表空间的相关数据文件号
LAST_CHANGE#	NUMBER	对此数据文件所做的最后一次更改号。如果数据文件正在更改，设置为 NULL
LAST_TIME	DATE	最后一次更改的时间戳
OFFLINE_CHANGE#	NUMBER	最后脱机文件的更改号。此列仅在数据文件进入联机时更新
ONLINE_CHANGE#	NUMBER	最后脱机范围的联机更改号
ONLINE_TIME	DATE	最后脱机范围的联机时间戳
BLOCKS	NUMBER	以块计的当前数据文件尺寸；如果不可访问，为 0
BLOCK_SIZE	NUMBER	数据文件的块尺寸
NAME	VARCHAR2(512)	数据文件名
PLUGGED_IN	NUMBER	描述是否插入表空间。如果插入表空间且未进行读写，设置为 1，否则设置为 0

表 B-2ENABLED 列的值

ENABLED 列值	说明
DISABLED	不允许 SQL 访问
READ ONLY	不允许 SQL 更新
READ WRITE	允许完全访问
UNKNOWN	除非控制文件出错，否则不应该出现

33．V$DATAFILE_COPY

这个视图显示来自控制文件的数据文件拷贝信息。

列	数据类型	说明

RECID	NUMBER	数据文件拷贝记录 ID
STAMP	NUMBER	数据文件拷贝记录时间戳
NAME	VARCHAR2(512)	数据文件拷贝文件名。此名字的最大长度与 OS 有关
TAG	VARCHAR2(32)	数据文件拷贝标记
FILE#	NUMBER	绝对数据文件号
RFILE#	NUMBER	表空间的相关数据文件号
CREATION_CHANGE#	NUMBER	数据文件的创建更改号
CREATE_TIME	DATE	数据文件的创建时间戳
RESETLOGS_CHANGE#	NUMBER	数据文件拷贝时的重置日志更改号
RESETLOGS_TIME	DATE	数据文件拷贝时的重置日志时间戳
CHANGE#		
INCREMENTAL_LEVEL	NUMBER	增量级
CHECKPOINT_ CHANGE#	NUMBER	进行数据文件拷贝时的检查点更改号
CHECKPOINT_TIME	DATE	进行数据文件拷贝时的检查点时间戳
ABSOLUTE_FUZZY_ CHANGE#	NUMBER	数据文件拷贝时看到的最高更改
RECOVERY_FUZZY_ CHANGE#	NUMBER	介质恢复写到此文件的最高更改
RECOVERY_FUZZY_ TIME	DATE	介质恢复写到此文件的最高更改时间戳
ONLINE_FUZZY	VARCHAR2(3)	YES/NO。如果设置为 YES，这是一个在崩溃或立即脱机后利用一个操作系统实用程序进行的拷贝。（或者是一个在数据库联机或打开时的无效拷贝）。恢复将需要应用所有重做直到下一个崩溃恢复标记使该文件一致
BACKUP_FUZZY	VARCHAR2(3)	YES/NO。如果设置为 YES，这是一个利用 BEGIN BACKUP/END BACKUP 技术进行的拷贝。恢复将需要应用所有重做直到下一个结束标记使这个拷贝一致
MARKED_CORRUPT	NUMBER	这个拷贝操作标记出错的块数。即在原数据文件中未标记出错，但在拷贝操作中检测到并标记为出错的块数
MEDIA_CORRUPT	NUMBER	介质出错块的总数。例如，校验和错的块被标记为介质错
LOGICALLY_CORRUPT	NUMBER	逻辑出错块的总数。例如，对不可恢复的操作应用重做将标记涉及的块为逻辑错
BLOCKS	NUMBER	以块计的数据文件拷贝的尺寸（也是进行拷贝时的数据文件尺寸）
BLOCK_SIZE	NUMBER	
OLDEST_OFFLINE_ RANGE	NUMBER	在这个控制文件拷贝中最旧的脱机范围记录的 RECID。数据文件拷

		贝为0
COMPLETION_TIME	DATE	拷贝完成时间
DALETED	VARCHA2(3)	YES/NO。如果设置为 YES，则该数据文件拷贝已经删除或重写

34．V$DATAFILE_HEADER

这个视图显示来自数据文件标题的数据文件信息

列	数据类型	说明
FILE#	NUMBER	数据文件号（来自控制文件）
STATUS	VARCHAR2(7)	ONLINE/OFFLINE（来自控制文件）
ERROR	VARCHAR2(18)	如果数据文件标题读取和验证成功，为 NULL 如果读失败，则其余的列为 NULL。如果验证失败，则其余的列可能显示无效数据。如果存在错误，则在数据文件可以恢复或使用前，必须利用备份进行修复
FORMAT	NUMBER	指出标题块的格式。可能的值为 6、7、8、0 6 表示 Oracle 版本 6 7 表示 Oracle 版本 7 8 表示 Oracle 版本 8 0 表示此格式不能确定（例如，标题不能读出）
RECOVER	VARCHAR2(3)	文件需要介质恢复
FUZZY	VARCHAR2(3)	文件是模糊的 YES/NO
CREATION_CHANGE#	NUMBER	数据文件创建的更改号
CREATION_TIME	DATE	数据文件创建的时间戳
TABLESPACE_NAME	VARCHAR2(30)	表空间名
TS#	NUMBER	表空间号
RFILE#	NUMBER	表空间的相关数据文件号
RESETLOGS_CHANGE#	NUMBER	重置日志更改号
CHECKPOINT_CHANGE#	NUMBER	数据文件检查点更改号
CHECKPOINT_ TIME	DATE	数据文件检查点时间戳
CHECKPOINT_COUNT	NUMBER	数据文件检查点计数
BYTES	NUMBER	以字节计的当前数据文件尺寸
BLOCKS	NUMBER	以块计的当前数据文件的文件尺寸
NAME	VARCHAR2(512)	数据文件名

35．V$DBFILE

此视图列出构成数据库的所有数据文件。这个视图是为了历史兼容而保存的。建议使用

V$DATAFILE。更多的信息，请参阅 V$DATAFILE。

列	数据类型	说明
FILE#	NUMBER	文件标识符
NAME	VARCHAR2	文件名

36．V$DBLINK

这个视图描述在 V$DBLINK 上发布查询的会话打开的所有数据库连接（带 IN_TRANSACTION=YES 的连接）。这些数据库连接在关闭前必须提交或退回。

列	数据类型	说明
DBLINK	VARCHAR2(128)	数据连接名
OWNER_ID	NUMBER	数据库连接 UID 的拥有者
LOGED_ON	VARCHAR2(3)	当前数据库连接是否登录
HETEROGENEOUS	VARCHAR2(3)	数据库连接是否异构
PROTOCOL	VARCHAR2(6)	数据库连接的通讯协议
OPEN_CURSORS	NUMBER	此数据库连接是否存在打开的游标
IN_TRANSACTION	NUMBER	此数据库连接当前是否处于事务处理中
UPDATE_SENT	VARCHAR2(3)	数据库连接上是否曾经有过更新
COMMIT_STRENGTH	NUMBER	数据库连接上的事务处理的提交点强度

37．V$DB_OBJECT_CACHE

这个视图显示高速缓存在数据库高速缓存中的数据库对象。这些对象包括表、索引、簇、同义词定义、PL/SQL 过程和程序包、触发器。

列	数据类型	说明
OWNER	VARCHAR2	对象的拥有者
NAME	VARCHAR2	对象名
DBLINK	VARCHAR2	数据库连接名
NAMESPACE	VARCHAR2	对象的库高速缓存名称空间：TABLE/PROCEDURE、BODY、TIGGER、INDEX、CLUSTER、OBJECT
TYPE	VARCHAR2	对象类型：INDEX、TABLE、CLUSTER、VIEW、SET、SYNONYM、SEQUENCE、PROCEDURE、FUNCTION、PACKAGE、PACKAGE BODY、TIGGER、CLASS、OBJECT、USER、DBLINK
HARARLE_MEM	NUMBER	对象消耗的共享缓冲池中共享内存量

LOADS	NUMBER	对象被装载的次数。在使某个对象无效时也增加这个计数
EXECUTIONS	NUMBER	不使用。为了看到实际的执行计数，请参阅 V$SQLAREA
LOCKS	NUMBER	当前锁住这个对象的用户数
PINS	NUMBER	当前固定这个对象的用户数
KEPT	VARCHAR2	YES 或者 NO，有赖于这个对象是否已经利用 PL/SQL 过程 DBMS_SHARED_POOL.KEEP“保持”（永久固定在内存中）

38．V$DB_PIPES

这个视图显示当前位于数据库中的管道

列	数据类型	说明
OWNER_ID	NUMBER	如果这是一个私有管理，则为拥有者 ID；否则为 NULL
NAME	VARCHAR2(1000)	管道名；例如，scott.pipe
TYPE	VARCHAR2(2)	PUBLIC 或 PRIVATE
PIPE_SIZE	NUMBER	管道占用的内存量

39．V$DELETED_OBJECT

这个视图显示来自控制文件的删除归档日志、数据文件拷贝和备份片的相关信息。这个视图的唯一用途是优化恢复目录的重新同步操作。在删除归档日志、数据文件拷贝或备份片时，相应的记录标记为删除。

列	数据类型	说明
RECID	NUMBER	删除对象记录的 ID
STAMP	NUMBER	删除对象的记录时间戳
TYPE	VARCHAR2(13)	ARCHIVED LOG/DATAFILE COPY/ BACKUPPIECE。删除对象的类型
OBJECT_RECID	NUMBER	删除对象的记录 ID
OBJECT_STAMP	NUMBER	删除对象的记录时间戳

40．V$DISPATCHER

这个视图提供有关调度程序进程的信息

列	数据类型	说明
NAME	VARCHAR2	调度进程名
NETWORK	VARCHAR2	调度程序的网络地址
PADDR	RAW(4)	进程地址
STATUS	VARCHAR2	调度程序状态：WAIT(空闲)、SEND(发送一个连接消息)、RECEIVE(接收一个消息)、CONNECT(建立一个连接)、DISCONNECT(处理断开请求)、BREAK (处理断开) 、

		OUTBOUND(建立出站连接)
ACCEPT	VARCHAR2	这个调度程序是否接受新的连接：YES、NO
MESSAGES	NUMBER	此调度程序处理的消息数目
BYTES	NUMBER	此调度程序处理的消息大小、以字节计
BREAKS	NUMBER	这个连接中发生的断开数目
OWNED	NUMBER	这个调度程序拥有的虚电路数目
CREATED	NUMBER	这个调度程序创建的虚电路数目
IDLE	NUMBER	这个调度程序的总空闲时间，以百分之一秒计
BUSY	NUMBER	这个调度程序的总繁忙时间，以百分之一秒计
LISTENER	NUMBER	此调度程序从监听程序收到的最近Oracle 错误数目
CONF_INDX	NUMBER	这个调度程序使用的MTS_DISPATCHERS 配置的基于零的索引

41．V$DISPATCHER_RATE

这个视图提供调度程序进程的速率统计数据

列	数据类型	说明
NAME	VARCHAR2	调度进程名
PADDR	RAW(4)	进程地址
CUR_LOOP_RATE	NUMBER	循环事件的当前速率
CUR_EVENT_RATE	NUMBER	事件的当前速率
CUR_EVENTS._PER_LOOP	NUMBER	每个循环的当前事件
CUR_MSG_RATE	NUMBER	消息的当前速率
CUR_SVR_BYTE_RATE	NUMBER	服务器缓冲区当前速率
CUR_SVR_BYTE_PER_BUF	NUMBER	服务器当前字节速率
CUR_CLT_BUF_RATE	NUMBER	客户机缓冲区当前速率
CUR_CLT_BYTE_RATE	NUMBER	客户机缓冲区当前字节速率
CUR_CLT_BYTE_PER_BUF	NUMBER	客户机每个缓冲区当前字节
CUR_BUF_RATE	NUMBER	缓冲区当前速率
CUR_BYTE_RATE	NUMBER	当前字节速率
CUR_BYTE_PER_BUF	NUMBER	每个缓冲区当前字节
CUR_IN_CONNECT_RATE	NUMBER	当前入站连接
CUR_OUT_CONNECT_RATE	NUMBER	当前出站连接
CUR_RECONNECT_RATE	NUMBER	连接迟和复用的当前连接
MAX_LOOP_RATE	NUMBER	循环事件的最大速率
MAX_EVENT_RATE	NUMBER	事件的最大速率
MAX_EVENTS_PER_LOOP	NUMBER	每个循环的最大事件数
MAX_MSG_RATE	NUMBER	消息的最大速率
MAX_SRV_BUF_RATE	NUMBER	服务器缓冲区最大速率
MAX_SRV_BYTE_RATE	NUMBER	服务器最大字节速率
MAX_SRV_BYTE_PER_BUF	NUMBER	服务器的每个缓冲区最大字节数
MAX_CLT_BUF_RATE	NUMBER	客户机缓冲区最大速率
MAX_CLT_BYTE_RATE	NUMBER	客户机最大字节速率

MAX_CLT_BYTE_PER_BUF	NUMBER	客户机的每个缓冲区最大字节数
MAX_BUF_RATE	NUMBER	缓冲区最大速率
MAX_BYTE_RATE	NUMBER	最大字节速率
MAX_BYTE_PER_BUF	NUMBER	每个缓冲区最大字节数
MAX_IN_CONNECT_RATE	NUMBER	入站连接的最大数
MAX_OUT_CONNECT_RATE	NUMBER	出站连接的最大数
MAX_RECONNECT_RATE	NUMBER	连接池和复用的最大连接数
AVG_LOOP_RATE	NUMBER	循环事件的平均速率
AVG_EVENT_RATE	NUMBER	循环事件的平均速率
AVG_EVENTS_PER_LOOP	NUMBER	每个循环的平均事件数
AVG_MSG_RATE	NUMBER	消息的平均速率
AVG_SVR_BUF_RATE	NUMBER	服务器缓冲区平均速率
AVG_SVR_BYTE_RATE	NUMBER	服务器的平均字节速率
AVG_SVR_BYTE_PER_BUF	NUMBER	服务器每个缓冲区的平均字节数
AVG_CLT_BUF_RATE	NUMBER	客户机缓冲区平均速率
AVG_CLT_BYTE_RATE	NUMBER	客户机平均字节速率
AVG_CLT_BYTE_PER_BUF	NUMBER	客户机每个缓冲区平均字节数
AVG_BUF_RATE	NUMBER	缓冲区平均速率
AVG_BYTE_RATE	NUMBER	平均速率字节速率
AVG_BYTE_PER_BUF	NUMBER	每个缓冲区平均字节数
AVG_IN_CONNECT_RATE	NUMBER	平均入站连接数
AVG_OUT_CONNECT_RATE	NUMBER	平均出站连接数
AVG_RECONNECT_RATE	NUMBER	连接池和复用的平均重连接数
NUM_LOOPS_TRACKED	NUMBER	跟踪循环的数目
NUM_MSG_TRACKED	NUMBER	跟踪消息的数目
NUM_SRV_BUF_TRACKED	NUMBER	跟踪服务器的缓冲区数目
NUM_CLT_BUF_TRACKED	NUMBER	跟踪客户机的缓冲区数目
NUM_BUF_TRACKED	NUMBER	跟踪缓冲区的数目
NUM_IN_CONNECT_RATE	NUMBER	跟踪入站连接的数目
NUM_OUT_CONNECT_RATE	NUMBER	跟踪出站连接的数目
NUM_RECONNECT_RATE	NUMBER	跟踪重连接的数目
SCALE_LOOPS	NUMBER	循环的规模
SCALE_MSG	NUMBER	消息的规模
SCALE_SRV_BUF	NUMBER	服务器缓冲区的规模
SCALE_CLT_BUF	NUMBER	客户机缓冲区的规模
SCALE_BUF	NUMBER	缓冲区的规模
SCALE_IN_CONNECT	NUMBER	入站连接的规模
SCALE_OUT_CONNECT	NUMBER	出站连接的规模
SCALE_RECONNECT	NUMBER	重连接的规模

42．V$DLM_ALL_LOCKS

这是一个并行服务器视图。此视图列出锁管理程序当前已知的被其他锁阻塞或阻塞其他锁的所有锁的信息。

列	数据类型	说明
LOCKP	RAW(4)	锁指针
GRANT_LEVEL	VARCHAR2(9)	锁的授权级别
REQUEST_LEVEL	VARCHAR2(9)	锁的请求级别
RESOURCE_NAME1	VARCHAR2(30)	锁的资源名

RESOURCE_NAME2	VARCHAR2(30)	锁的资源名
PID	NUMBER	拥有锁的进程标识
TRANSACTION_ID0	NUMBER	锁所属的事务处理标识符的低 4 个字节
TRANSACTION_ID1	NUMBER	锁所属的事务处理标识符的高 4 个字节
GROUP_ID	NUMBER	锁的组标识符
OPEN_OPT_ DEADLOCK	NUMBER	如果设置 DAADLOCK 开放选项，为 1，否则为 0
OPEN_OPT_ PERSISTENT	NUMBER	如果设置 PERSISTENT 开放选项，为 1，否则为 0
OPEN_OPT_ PROCESS_OWNED	NUMBER	如果设置 PROCESS_OWNED 开放选项, 为 1，否则为 0
OPEN_OPT_NO_XID	NUMBER	如果设置 NO_XID 开放选项，为 1，否则为 0
CONVERT_OPT_ GETVALUE	NUMBER	如果设置 GETVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_ PUTVALUE	NUMBER	如果设置 PUTVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_ NOVALUE	NUMBER	如果设置 NOVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_ DUBVALUE	NUMBER	如果设置 DUBVALUE 转换选项，为否则为 0
CONVERT_OPT_ NOQUEUE	NUMBER	如果设置 NOQUEUE 转换选项，为 1，否则为 0
CONVERT_OPT_ EXPRESS	NUMBER	如果设置 EXPRESS 转换选项，为 1，否则为 0
CONVERT_OPT_ NODEADLOCKWAIT	NUMBER	如果设置 NODEADLOCKWAIT 转换选项，为 1，否则为 0
WHICH_QUEUE	NUMBER	锁当前所在的队列。NULL 队列为 0；GRANTED 队列为 1；CONVER 队列为 2
LOCKSTATE	NUMBER	拥有者所看到的锁的状态
AST_EVENT0	NUMBER	最后一个 AST 事件
OWNER_NODE	NUMBER	如果这个锁请求被其他锁请求阻塞，为否则为 0
BLOCKER	NUMBER	如果这个锁正阻塞其他锁，为 1，否则为 0

43．V$DLM_CONVERT_LOCAL

这个视图显示本地锁转换操作所用的时间。

列	数据类型	说明
INST_ID	NUMBER	实例的 ID
CONVERT_TYPE	VARCHAR2(64)	列在表 B-3 中的转换类型
AVERAGE_CONVERT _TIME	NUMBER	每种锁操作类型的平均转换时间，以百分之一秒计

CONVERT_COUNT	NUMBER	锁转换操作的数目

44．V$DLM_CONVERT_REMOTE

V$DLM_CONVERT_REMOTE 显示远程锁转换操作的时间（见表 B-3）。

列	数据类型	说明
INST_ID	NUMBER	实例的 ID
CONVERT_TYPE	VARCHAR2(64)	列在表 B-3 中的转换类型
AVERAGE_CONVERT_TIME	NUMBER	每种锁操作类型的平均转换时间，以百分之一秒计
CONVERT_COUNT	NUMBER	操作的次数

表 B-3 CONVERT_TYPE 列的值

转换类型	说 明
NULL->SS	NULL 模式到子共享模式
NULL->SX	NULL 模式到共享互斥模式
NULL->S	NULL 模式到共享模式
NULL->SSX	NULL 模式到子共享互斥模式
SS->SX	NULL 模式到互斥模式
SS->S	子共享模式到共享互斥模式
SS->SSX	子共享模式到子共享互斥模式
SS->X	子共享模式到互斥模式
SX->S	共享互斥模式到共享模式
SX-SSX	共享模式到子共享互斥模式
SX->X	共享互斥模式到互斥模式
S->SX	共享模式到共享互斥模式
S-SSX	共享模式到子共享互斥模式
S->X	共享模式到互斥模式
SSX->X	共享互斥模式到互斥模式

45．V$DLM_LATCH

这个视图已经废弃。关于 DCM 栓锁的性能统计数据请参阅 V$LATCH。

46．V$DLM_LOCKS

这是一个并行服务器视图。此视图列出锁管理程序当前已知的被其他锁阻塞或阻塞其他锁的信息。

列	数据类型	说明
LOCKP	RAW(4)	锁指针
GRANT_LEVEL	VARCHAR2(9)	锁的授权级别
REQUEST_LEVEL	VARCHAR2(9)	锁的请求级别
RESOURCE_NAME1	VARCHAR2(30)	锁的资源名
RESOURCE_NAME2	VARCHAR2(30)	锁的资源名
PID	NUMBER	拥有锁的进程标识符
TRANSACTION_ID0	NUMBER	锁所属的事务处理标识符的低 4 个字节
TRANSACTION_ID1	NUMBER	锁所属的事务处理标识符的高 4 个字节

GROUP_ID	NUMBER	锁的组标识符
OPEN_OPT_ DEADLOCK	NUMBER	如果设置 DAADLOCK 开放选项，为 1，否则为 0
OPEN_OPT_ PERSISTENT	NUMBER	如果设置 PERSISTENT 开放选项，为 1，否则为 0
OPEN_OPT_ PROCESS_OWNED	NUMBER	如果设置 PROCESS_OWNED 开放选项，为 1，否则为 0
OPEN_OPT_NO_XID	NUMBER	如果设置 NO_XID 开放选项，为 1，否则为 0
CONVERT_OPT_ GETVALUE	NUMBER	如果设置 GETVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_ PUTVALUE	NUMBER	如果设置 PUTVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_ NOVALUE	NUMBER	如果设置 NOVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_ DUBVALUE	NUMBER	如果设置 DUBVALUE 转换选项，为 1，否则为 0
CONVERT_OPT_ NOQUEUE	NUMBER	如果设置 NOQUEUE 转换选项，为 1，否则为 0
CONVERT_OPT_ EXPRESS	NUMBER	如果设置 EXPRESS 转换选项，为 1，否则为 0
NODEADLOCKWAIT	NUMBER	如果设置 NODEADLOCKWAIT 转换选项，为 1，否则为 0
NODEADLOCKBLOCK	NUMBER	如果设置 NODEADLOCKBLOCK 转换选项，为 1，否则为 0
WHICH_QUEUE	NUMBER	锁当前所在的队列。NULL 队列为 0；GRANTED 队列为 1；CONVER 队列为 2
LOCKSTATE	VARCHAR2(64)	拥有者所看到的锁的状态
AST_EVENT0	NUMBER	最后一个 AST 事件
OWNER_NODE	NUMBER	节点标识符
BLOCKED	NUMBER	如果这个锁请求被其他锁请求阻塞，为 1，否则为 0
BLOCKER	NUMBER	如果这个锁正阻塞其他锁，为 1，否则为 0

47．V$DLM_MISC

这个视图显示其他 DLM 统计数据

列	数据类型	说明
STATISTIC#	NUMBER	数据统计号
NAME	VARCHAR2(64)	统计数据名
VALUE	NUMBER	与统计数据有关的值

48．V$DLM_RESS

这是一个并行服务器视图。它显示锁管理器当前所知的所有资源的信息。

列	数据类型	说明
RESP	RAW(4)	资源指针
RESOURCE_NAME	VARCHAR2(30)	锁的十六进制表示的资源名
ON_CONVERT_Q	NUMBER	如果在转换队列上，为 1，否则为 0
ON_GRANT_Q	NUMBER	如果在授权队列上，为 1，否则为 0
PERSISTENT_RES	NUMBER	如果是一个永久资源，为 1，否则为 0
RDOMAIN_NAME	VARCHAR2(25)	恢复域名
RDOMAINP	RAW(4)	恢复指针域名
MASTER_NODE	NUMBER	主机节点 ID
NEXT_CVT_LEVEL	VARCHAR2(9)	全局转换队列上转换的下一个锁的级别
VALUE_BLK_STATE	VARCHAR2(32)	值块的状态
VALUE_BLK	VARCHAR2(64)	值块的前 64 字节

49．V$ENABLEDPRIVS

此视图显示启用的权限。这些权限可在表 SYS.SYSTEM_PRIVIELEGES_MAP 中找到。

列	数据类型	说明
PRIV_NUMBER	NUMBER	启用权限的数字标识符

50．V$ENQUEUE_LOCK

这个视图显示排队状态对象拥有的所有锁。这个视图中的列等同于 V$LOCK 视图中的列。更多的信息，请参阅 V$LOCK。

列	数据类型	说明
ADDR	RAW(4)	锁状态对象的地址
KADDR	RAW(4)	锁地址
SID	NUMBER	拥有或获得此锁的会话的标识符
TYPE	VARCHAR2(2)	锁的类型。可能具有锁的用户和系统类型列表
ID1	NUMBER	锁标识符#1（依赖于类型）
ID2	NUMBER	锁标识符#2（依赖于类型）
LMODE	NUMBER	会话拥有此锁的锁模式： 0，没有 1，空（NULL） 2，行子共享模式（SS） 3，行共享互斥模式（SX） 4，共享模式（S） 5，行子共享互斥模式 6，互斥模式（X）
REQUEST	NUMBER	进程请求锁的模式： 0，没有 1，空（NULL）

		2，行子共享模式（SS） 3，行共享互斥模式（SX） 4，共享模式（S） 5，行子共享互斥模式 6，互斥模式（X）
CTIME	NUMBER	授予当前模式以来的时间
BLOCK	NUMBER	此锁正阻塞其他锁

51．V$EVENT_NAME

这个视图包含等待事件的有关信息

列	数据类型	说明
EVENT#	NUMBER	等待事件号
NAME	VARCHAR2(64)	等待事件名
PARAMETER1	VARCHAR2(64)	等待事件的第一个参数的说明
PARAMETER2	VARCHAR2(64)	等待事件的第二个参数的说明
PARAMETER3	VARCHAR2(64)	等待事件的第三个参数的说明

52．V$EXECUTE

这个视图并行执行的有关信息。

列	数据类型	说明
PID	NUMBER	会话 ID
DEPTH	NUMBER	深度
FUCTION	VARCHAR2(10)	会话系列号
TYPE	VARCHAR2(7)	计划表中的 OBJECT_NODE 的名称
NVALS	NUMBER	OBJECT_NODE 占用的时间
VAL1	NUMBER	编号 1 的值
VAL2	NUMBER	编号 2 的值
SEQH	NUMBER	一个序列
SEQL	NUMBER	一个序列

53．V$FALSE_PING

这个视图是一个并行服务器视图。这个视图显示可能正处于假 ping 的缓冲区。即，那些由相同的锁随其他 ping 了 10 次以上的缓冲区保护的 ping 了 10 次以上的缓冲区。标识为假 ping 的缓冲区可重新映射到 GC-FILE_TO_LOCKS 以减少锁冲突。

列	数据类型	说明
FILE#	NUMBER	文件标识号（为找到文件名，可查询 DB_DATA_FILES 或 V$DBFILES）
BLOCKS#	NUMBER	块号
STATUS	VARCHAR2(1)	块状态： FREE=当前不用 XCUR=互斥的 CR=一致的读取 READ=从磁盘读

		MREC=处于介质恢复模式 IREC=处于实例恢复模式
XNC	NUMBER	由于与其他实例争用导致的 PCM 锁变换数。此例已作废，但为了历史兼容性仍然保留
FORCED_READS	NUMBER	锁必须从磁盘重新读取的次数，重新读取是由于其他实例通过在锁模式中请求此锁上的 PCM 锁，强迫它退出了此实例的高速缓存
FORCED_WRITES	NUMBER	由于这个实例已经搞坏了这个块并且其他实例已经以冲突的模式请求了这个锁上的 PCM 锁，而导致 DBWR 必须将这个块写入磁盘的次数
NAME	VARCHAR2(30)	包含该块的数据库对象名
PARTITION_NAME	VARCHAR2	非分区对象为 NULL
KIND	VARCHAR2(12)	数据库对象名。请参阅表 B-1
OWNER#	NUMBER	拥有者号
LOCK_ELEMENT_ADDR	RAW(4)	包含覆盖缓冲区的 PCM 锁的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT_NAME	NUMBER	包含覆盖缓冲区的 PCM 锁的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT_CLASS	NUMBER	锁元素类

54．V$FAST_START_SERVERS

此视图提供执行并行事务处理恢复的恢复从服务器的有关信息

列	数据类型	说明
STATE	VARCHAR2(11)	服务器状态：IDLE 或 RECOVERING
UNDOBLOCKSDONE	NUMBER	至今所做的分配工作的百分比
PID	NUMBER	进程 ID

55．V$FAST_START_TRANSACTIONS

这个视图包含与 Oracle 正在恢复的事务处理的进展有关的信息。

列	数据类型	说明
USN	NUMBER	事务处理的撤消段号
SLT	NUMBER	在回退段内的位置
SEQ	NUMBER	位置的具体编号
STATE	VARCHAR2(16)	事务处理的状态可能是
UNDOBLOCKSDONE	NUMBER	这个事务处理中完成的撤消块数目 TO BE RECOVERED、RECOVERING

UNDOBLOCKSTOTAL	NUMBER	需要恢复的撤消块总数
PID	NUMBER	被分配给当前服务器的 ID
CPUTIME	NUMBER	进行恢复的已用时间，以秒计
PARENTUSN	NUMBER	PDML 中父级事务处理的撤消段号
PARENTSLT	NUMBER	PDML 中父级事务处理的位置
PARENTSEQ	NUMBER	PDML 中父级事务处理的序列号

56．V$FILE_PING

这个视图显示每个数据文件 ping 的块号。这个信息又可以用来确定对现有数据文件的访问模式和确定从数据文件块到 PCM 锁的新映射。

列	数据类型	说明
FILE_NUMBER	NUMBER	数据文件号
FREQUENCY	NUMBER	频率
X_2_NULL	NUMBER	文件中所有块从互斥到空的锁转换数
X_2_NULL_FORCED^|_WRITE	NUMBER	指定文件中的块由于互斥到空的转换而强制写的数目
X_2_NULL_FORCED^|_STALE	NUMBER	文件中的块由于互斥到空的转换而使其 STATE 的次数
X_2_S	NUMBER	文件中所有块从互斥到共享的锁转换数
X_2_S_FORCED_	NUMBER	指定文件中的块由于互斥到共享的转换而强制写的数目
X_2_SSX	NUMBER	文件中所有块从互斥到共享互斥的锁转换数
X_2_SSX_FORCED^|_WRITE	NUMBER	指定文件中的块由于互斥到子共享互斥的转换而强制写的数目
S_2_NULL	NUMBER	文件中所有块从共享到空的锁转换数
S_2_NULL_FORCED^|_STALE	NUMBER	文件中的块由于共享到空的转换而使其 STATE 的次数
SS_2_NULL	NUMBER	文件中所有块从子共享到空的锁转换数
SS_2_RLS	NUMBER	释放的 PCM 锁子共享锁的数目。在 Oracle 8.1 中为 0
WRB	NUMBER	实例接收到一个跨实例调用这个文件的写入单个缓冲区的次数
WRB_FORECD_ WRITE	NUMBER	由于跨实例调用这个文件而写入单个缓冲区导致写入的块数
RBR	NUMBER	实例接收到一个跨实例调用这个文件的重用块范围的次数
RBR_FORECD_ WRITE	NUMBER	由于跨实例调用这个文件而重用块范围的块数
RBR_FORECD_ STATE	NUMBER	由于跨实例调用而重用块范围而使得这个文件中的块 STATE 的数目
CBR	NUMBER	实例接收到一个跨实例调用这个文件的检查点块范围的次数
CBR_FORECD_ WRITE	NUMBER	由于跨实例调用这个文件的检查点跨范围所导致的写入这个文件中块的数目
NULL_2_X	NUMBER	指定文件中所有块从空到互斥的锁转换数
S_2_X	NUMBER	指定文件中所有块从共享到互斥的锁转

		换数
SSX_2_X	NUMBER	指定文件中所有块从子共享互斥到的互斥的锁转换数
NULL_2_S	NUMBER	指定文件中所有块从空到共享的锁转换数转换数
NULL_2_SS	NUMBER	指定文件中所有块从空到子共享的锁
OP_2_SS	NUMBER	打开的 PCM 锁子共享锁的数目。Oracle 8.1 中为 0

57．V$FILESTAT

这个视图包含文件读写统计数据的有关信息

列	数据类型	说明
FILE#	NUMBER	文件号
PHYRDS	NUMBER	所在行的物理读的次数
PHYWRTS	NUMBER	DBWR 要求写入的次数
PHYBLKRD	NUMBER	物理块读取的次数
PHYBLKWRT	NUMBER	写入磁盘的块数目；如果所有写入都是单块，这个数与 PHYWRTS 相同
READTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为完成读取所用的时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
WRITETIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为完成写入所用的时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
AVGIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为 I/O 所用的时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
LSTIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE ,则为完成最后 I/O 所用的时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
LSTIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE ,则为完成最后 I/O 所用的时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
MINIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为单个 I/O 所用的最小时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
MAXIOWTM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为完成单个写入所用的最大时间（以百分之一秒计）；如果该参数为 FALSE，则为 0
MAXIORTM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为完成单个读取所用的最大时间（以百分之一秒计）；如果该参数为 FALSE，则为 0

58．V$FIXED_TABLE

这个视图显示数据库中所有动态性能表、视图和导出表。某些 V$表(如 V$ROLLNAME)涉及实际的表，因此没有列出。

列	数据类型	说明
NAME	VARCHAR2(30)	对象名
OBJECT_ID	NUMBER	固定对象的标识符
TYPE	VARCHAR2(5)	对象类型：TABLE、VIEW
TABLE_NUM	NUMBER	如果动态性能表为TABLE 类型，则为标识它的号码

59．V$FIXED_VIEW_DEFINITION

这个视图包含所有固定视图（以 V$起头的视图）的定义。应该仔细使用这些表。Oracle 试图保持这些固定视图在各个版本中都相同，但还是可能会更改某些固定视图的定义而不作通知。可通过利用动态性能表的索引列来使用这些定义以优化查询。

列	数据类型	说明
VIEW_NAME	VARCHAR2(30)	固定视图名
VIEW_DEFINTION	VARCHAR2(2000)	固定视图定义

60．V$GLOBAL_BLOCKED_LOCKS

这个视图显示全局阻塞的锁。

列	数据类型	说明
ADDR	RAW(4)	锁状态对象的地址（raw）
KADDR	RAW(4)	锁地址（raw）
SID	NUMBER	拥有或获得此锁的会话的标识符（number）
TYPE	VARCHAR2(2)	资源类型（number）
ID1	NUMBER	资源标识符#1（number）
ID2	NUMBER	资源标识符#2（number）
LMODE	NUMBER	拥有的锁模式（number）
REQUEST	NUMBER	请求的锁模式（number）
CTIME	NUMBER	授予当前模式以来的时间

61．V$GLOBAL_TRANSACTION

这个视图显示当前归档的全局事务处理的有关信息

列	数据类型	说明
FORMATID	NUMBER	全局事务处理的格式标识符
GLOBALID	NUMBER	全局事务处理的标识符
BRANCHID	NUMBER	全局事务处理的分支标识符
BRANCHES	NUMBER	全局事务处理的分支总数
REFCOUNT	NUMBER	这个全局事务处理的同级数，必须与分支数相同
PREPARECOUNT	NUMBER	已经预备的全局事务处理的分支数

STATE	VARCHAR2(18)	全局事务处理的分支状态
FLAGS	NUMBER	状态的数字表示
COUPLING	VARCHAR2(15)	分支是松散耦合还是紧密耦合

62．V$HS_AGENT

这个视图确定当前运行在某个主机上的 HS 代理，每个代理进程采用一行。

列	数据类型	说明
AGENT_ID	NUMBER	连接到代理使用的 Net8 会话标识符（listernet.ora SID）
MACHINE	NUMBER	操作系统机器名
PROCESS	NUMBER	代理的操作系统进程标识符
PROGRAM	NUMBER	代理的程序名
OSUSER	NUMBER	操作系统用户
STARTTIME	DATE	启动时间
AGENT_TYPE	NUMBER	代理的类型
FDS_CLASS_ID	NUMBER	外部数据存储类的 ID
FDS_INST_ID	NUMBER	外部数据存储的实例名

63．V$HS_SESSION

这个视图确定当前为 Oracle 服务器打开的 HS 会话组。

列	数据类型	说明
HS_SESSION_ID	NUMBER	唯一的 HS 会话标识符
AGENT_ID	NUMBER	V$HS_AGENT 的外部键
SID	NUMBER	用户会话标识符（V$SESSION 的外键）
DB_LINK	VARCHAR2(128)	用来访问代理的服务器数据库连接名，NULL 表示没有使用数据库连接（即，在使用外部过程时）
DB_LINK_OWNER	NUMBER	DB_LINK 中的数据库连接的拥有者
STARTTIME	DATE	连接的启动时间

64．V$INDEXED_FIXED_COLUMN

这个视图显示索引的动态性能表（X$表）中的列。X$表可以更改而不用通告。仅将此视图用于编写针对固定视图（V$视图）的查询更为有效。

列	数据类型	说明
TABLE_NAME	VARCHAR2(30)	索引的动态性能表的名称
INDEX_NUMBER	NUMBER	区分某个列属于哪个索引的编号
COLUMN_NAME	VARCHAR2(30)	被索引的列号
COLUMN_POSITION	NUMBER	索引键中列的位置（这主要与多列索引有关）

65．V$INSTANCE

这个视图显示当前实例的状态。这个版本的 V$INSTANCE 与前面版本的的 V$INSTANCE 不兼容。

列	数据类型	说明
INSTANCE_NUMBER	NUMBER	实例注册所用的实例号。对应与 INSTANCE _NUMBER 初始化参数。可参阅 INSTANCE_NUMBER
INSTANCE_NAME	VARCHAR2(16)	实例名
HOST_NAME	VARCHAR2(64)	主机名
VERSION	VARCHAR2(17)	RDBMS 版本
STARTUP_TIME	DATE	实例启动的时间
STATUS	VARCHAR2(7)	STARTED/MOUNTED/OPEN STARTED：启动安装后或数据库关闭后，OPEN：启动后或数据库打开后
PARALLEL	VARCHAR2(3)	YES/NO：是否并行服务器模式
THREAD#	NUMBER	实例打开的重做线程
ARCHIVER	VARCHAR2(7)	STOPPED/STARTED/FAILED；FAILED 表示归档程序最后一次归档某个日志失败，但在 5 分钟内将重试
LOG_SWITCH_WAIT	VARCHAR2(11)	正在等待 ARCHIVELOG/CLEAR LOG/ CHECKPOINT 事件日志切换。注意：如果 ALTER SYSTEM SWITCH LOGFILE 挂起，但在当前联机重做日志中还有空间则这个值为 NULL
LOGINS	VARCHAR2(10)	ALLOWED/RESTRICTED
SHUTDOWN_PENDING	VARCHAR2(3)	YES/NO
DATABASE_STATUS	VARCHAR2(17)	数据库的状态

66．V$INSTANCE_RECOVERY

这个视图用来监控实现恢复读取的用户特定限制的机制。

列	数据类型	说明
RECOVERY_ EXTIMATED_IOS	NUMBER	根据快速启动检查点在内存中的值，估计在恢复中将处理的块数
ACTUAL_REDO _BLOCKS	NUMBER	恢复所需的重做块的当前实际数目
TARGET_REDO _BLOCKS	NUMBER	恢复必须处理的重做块的当前目标数目
LOG_FILE_SIZE _REDO_BLKS	NUMBER	保证在检查点完成前不进行日志切换所需重做块的最大数目
LOG_CHKPT_TIME OUT_READ_BLKS	NUMBER	恢复中为满足 LOG_CHECKPOINT_TIMEOUT 需要处理的重做块数
LOG_CHKPT_ INTERVAL_READ_	NUMBER	恢复中为满足 LOG_CHECKPOINT_INTERVAL 需要

BLKS		处理的重做块数
FAST_START_IO_	NUMBER	恢复中为满足 FAST_START_IO_
TARGET_REDO_		TARGET 需要处理的重做块数
BLKS		

67．V$LATCH

这个视图列出非父级栓锁的统计数据和父级栓锁的汇总统计数据。即，父级栓锁的统计数据包括从其每个子级栓锁开始的数据。

说明：列 SLEEP5、SLEEP6、…SLEEP11 是为了与以前的 Oracle 班本兼容而给出的。不累加这些列的数据。

列	数据类型	说明
ADDR	RAW(4)	栓锁对象的地址
LATCH#	NUMBER	栓锁编号
LEVEL#	NUMBER	栓锁级别
NEME	VARCHAR2(64)	栓锁名
GETS	NUMBER	等待获得的次数
MISSES	NUMBER	等待获得但第一次尝试失败的次数
SLEEPS	NUMBER	在需要等待时睡眠的次数
IMMEDIATE_GETS	NUMBER	不用等待获得的次数
IMMEDIATE_MISSES	NUMBER	不等待获得失败的次数
WAITERS_WOKEN	NUMBER	等待被唤醒多少次
WAITS_HOLDING_LATCH	NMBER	拥有一个不同的栓锁时等待的次数
SPIN_GETS	NUMBER	第一次尝试失败，但在以后的轮次中成功
SLEEP1	NUMBER	睡眠 1 次的等待
SLEEP2	NUMBER	睡眠 2 次的等待
SLEEP3	NUMBER	睡眠 3 次的等待
SLEEP4	NUMBER	睡眠 4 次的等待
SLEEP5	NUMBER	睡眠 5 次的等待
SLEEP6	NUMBER	睡眠 6 次的等待
SLEEP7	NUMBER	睡眠 7 次的等待
SLEEP8	NUMBER	睡眠 8 次的等待
SLEEP9	NUMBER	睡眠 9 次的等待
SLEEP10	NUMBER	睡眠 10 次的等待
SLEEP11	NUMBER	睡眠 11 次的等待

68．V$LATCHHOLDER

这个视图包含当前栓锁拥有者的相关信息。

列	数据类型	说明
PID	NUMBER	拥有栓锁的进程标识符
SID	NUMBER	拥有栓锁的会话标识符
LADDR	LAW(4)	栓锁地址
NAME	VARCHAR2	被拥有的栓锁名称

69． V$LATCHNAME

这个视图包含确定 V$LATCH 视图中给出栓锁的栓锁名的有关信息。V$LATCHNAME 的

行与 V$LATCH 的行具有一一对应的关系。更多的信息，可参阅 V$LATCH。

列	数据类型	说明
LATCH#	NUMBER	栓锁号
NAME	VARCHAR2(64)	栓锁名

70．V$LATCH_CHILDREN

这个视图包含子级栓锁的有关统计数据。这个视图包含了 V$LATCH 的所有列再加上 CHILD# 列。注意，如果子级栓锁的 LATCH# 列互相配合，则子级栓锁具有相同的父级栓锁。更多信息，可参阅 V$LATCH。

列	数据类型	说明
ADDR	RAW(4)	栓锁对象的地址
LATCH#	NUMBER	父级栓锁的栓锁号
CHILD#	NUMBER	LATCH#中给出的父级栓锁的子级栓锁编号
LEVEL#	NUMBER	栓锁级别
NAME	VARCHAR2(64)	栓锁名
GETS	NUMBER	等待获得的次数
MISSES	NUMBER	等待获得但第一次尝试失败的次数
SLEEPS	NUMBER	在需要等待时睡眠的次数
IMMEDIATE_GETS	NUMBER	不用等待获得的次数
IMMEDIATE_MISSES	NUMBER	不等待获得失败的次数
WAITERS_WORKEN	NUMBER	等待被唤醒多少次
WAITS_HOLDING_LATCH	NUMBER	拥有一个不同的栓锁时等待的次数
SPIN_GETS	NUMBER	第一次尝试失败，但在以后的轮次中成功
SLEEPn	NUMBER	睡眠 n 次的等待

71．V$LATCH_MISSES

这个视图包含试图获得一个栓锁但失败的有关信息

列	数据类型	说明
PARENT_NAME	VARCHAR2	父级栓锁名
WHERE	VARCHAR2	试图获得此栓锁的位置
NWFAIL_COUNT	NUMBER	不等待获取栓锁失败的次数
SLEEP_COUNT	NUMBER	导致睡眠的获取尝试的次数
WTR_SLP_COUNT	NUMBER	
LONGHOLD_COUNT	NUMBER	

72．V$LATCH_PARENT

这个视图包含父级栓锁的有关统计数据。此视图的列与 V$LATCH 的列相同。更多的信息请参阅 V$LATCH。

73．V$LIBRARYCACHE

这个视图包含库高速缓存性能与活动的有关统计数据。

列	数据类型	说明

NAMESPACE	VARCHAR2(15)	名称空间
GETS	NUMBER	为这个名称空间中的对象请求某个锁的次数
GETHITS	NUMBER	在内存中找到某个对象的句柄的次数
GETHITRATIO	NUMBER	GETHITS 与 GETS 的比例
PINS	NUMBER	为这个名称空间中的对象请求 PIN 的次数
PINHITS	NUMBER	在内存中找到相应库对象的所有元数据片的次数
PINHITRATIO	NUMBER	PINHITS 与 PINS 的比例
RELOADS	NUMBER	自某个对象句柄建立以来，进行该对象的非第一次 PIN 的任意 PIN，这样需要将对象从磁盘装入
INVALIDATIONS	NUMBER	此名称空间中的对象由于相关的对象被修改而无效的总次数
DLM_LOCK_ REQUESTS	NUMBER	GET 请求锁定实例锁的次数
DLM_PIN_REQUESTS	NUMBER	PIN 请求锁定实例锁的次数
DLM_PIN_RELEASES	NUMBER	发布请求 PIN 实例锁的次数
DLM_INVALIDATION _REQUESTS	NUMBER	GET 请求无效实例锁的次数
DLM_INVALIDATIONS	NUMBER	从其他实例接收到的无效 ping 的次数

74．V$LICENSE

这个视图包含有关认证限制的信息

列	数据类型	说明
SESSIONS_MAX	NUMBER	实例允许的并发用户会话的最大数目
SESSIONS_WARNING	NUMBER	实例的并发用户会话的警告数目
SESSIONS_CURRENT	NUMBER	并发用户会话的最大数目
SESSIONS_HIGHWATER	NUMBER	实例启动以来的并发用户会话的最大数目
USERS_MAX	NUMBER	数据库允许的指定用户最大数目

75．V$LOADCSTAT

这个视图包含执行一个直接装载中搜集的 SQL*Loader 的统计数据。这些统计数据应用到整个装载过程。对于这个表的任何 SELECT 都会产生“无行返回”，因为不能同时装载数据又同时进行查询。

列	数据类型	说明
READ	NUMBER	读取记录数
REJECTED	NUMBER	拒绝的记录数
TDISCARD	NUMBER	装载中废弃的记录数
NDISCARD	NUMBER	从当前文件中废弃的记录数

76．V$LOADTSTAT

列	数据类型	说明
LOADED	NUMBER	装载的记录数
REJECTED	NUMBER	拒绝的记录数
FAILWHEN	NUMBER	不能满足任意 WHEN 子句的记录数
ALLNULL	NUMBER	全空从而不装载的记录数
LEFT2SKIP	NUMBER	在连续装载中要跳过的记录数
PTNLOADED	NUMBER	装载 PTN 的记录数

77．V$LOCK

这个视图列出 Oracle 服务器当前拥有的锁以及未完成的锁或栓锁请求。

列	数据类型	说明
ADDR	RAW(4)	锁状态对象的地址
KADDR	RAW(4)	锁地址
SID	NUMBER	拥有或获得此锁的会话的标识符
TYPE	VARCHAR2(2)	锁的类型。可能具有锁的用户和系统类型列表，请参阅表 B-4 和 B-5
ID1	NUMBER	锁标识符#1（依赖于类型）
ID2	NUMBER	锁标识符#2（依赖于类型）
LMODE	NUMBER	会话拥有此锁的锁模式： 0，没有 1，空（NULL） 2，行子共享模式（SS） 3，行共享互斥模式（SX） 4，共享模式（S） 5，行子共享互斥模式 6，互斥模式（X）
REQUEST	NUMBER	进程请求锁的锁模式： 0，没有 1，空（NULL） 2，行子共享模式（SS） 3，行共享互斥模式（SX） 4，共享模式（S） 5，行子共享互斥模式 6，互斥模式（X）
CTME	NUMBER	授予当前模式以来的时间
BLOCK	NUMBER	此锁正阻塞其他锁

表 B-4 中用户类型上的锁可由用户应用程序获得。任何阻塞其他进程的进程都可能拥有这些锁中的某一个。

表 B-4 用户类型

用户类型	说明
TM	DML 排队

TX	事务处理排队
UL	用户提供

表 B-5 中系统类型上的锁被拥有的时间计短。

表 B-5 类型列的值：系统类型

用户类型	说明
BL	缓冲区散列表实例
CF	控制文件模式全局排队
CI	交叉实例功能调用实例
CU	游标绑定
DF	数据文件实例
DL	直接装载程序并行索引创建
DM	安装/启动数据库主/副实例
DR	分布式恢复进程
DX	分布式事务处理项
FS	文件集
HW	特定段上的空间管理
IN	实例号
IR	实例恢复串行全局队列
IS	实例状态
IV	库高速缓存无效实例
JQ	作业队列
KK	线程突跳
LA…LP	库高速缓存锁实例锁（A…P=名称空间）
MM	安装定义全局队列
MR	介质恢复
NA…NZ	库高速缓存固定实例（A…Z=名称空间）
PF	口令文件
PI，PS	并行操作
PR	进程启动
QA…QZ	行高速缓存实例（A…Z=高速缓存）
RT	重做线程全局队列
SC	系统提交编号实例
SM	SMON
SN	序列号实例
SQ	序列号队列
SS	排序段
ST	空间事务处理队列
SV	序列号值
TA	一般队列
TS	临时段队列（ID2=0）
TS	新块分配队列（ID2=1）
TT	临时表队列
UN	用户名
US	撤消段 DDL
WL	开始写重做日时局实例

表 B-5 中系统类型上的锁被拥有的时间计短。

78．V$LOCK_ACTIVITY

这是一个并行服务器视图。这个视图显示当前实例的 DLM 锁的操作。每行对应锁操作的一种类型。

列	数据类型	说明
FROM_VAL	VARCHAR2(4)	PCM 锁初始状态：NUL；S；X；SSX
TO_VAL	VARCHAR2(4)	PCM 锁初始状态：NULL；S；X；SSX
ACTION_VAL	VARCHAR2(51)	锁转换说明 读取的锁缓冲区 写入的锁缓冲区 使缓冲区 CR(非写入)读锁为写 使缓冲区 CR(写灰缓冲区) 将写锁降为读(写灰缓冲区) 写事务处理表/撤消块 事务处理表/撤消块（写灰缓区） 使事务处理表/撤消块可共享 重新配备事务处理表写机制
COUNTER	NUMBER	执行的锁操作次数

79．V$LOCK_ELEMENT

这是一个并行服务器视图。每个 PCM 锁在此视图中有一项，这项由缓冲区高速缓存使用。对应一个锁元素的 PCM 锁的名称为{‘BL’，索引，类}。

列	数据类型	说明
LOCK_ELEMENT_ADDR	RAW(4)	包含覆盖缓冲区的 PCM 锁的锁元素地址。如果不止一个缓冲区具有相同的地址，则这些缓冲区由相同的 PCM 覆盖。
LOCK_ELEMENT_NAME	NUMBER	包含覆盖缓冲区的 PCM 锁的锁的名称。
INDX	NUMBER	平台专用的锁管理程序标识符
CLASS	NUMBER	平台专用的锁管理程序标识符
MODE_HELD	NUMBER	与平台相关的拥有锁模式的值；一般：3=共享；5=互斥
BLOCK_COUNT	NUMBER	PCM 锁覆盖的块数
RELEASING	NUMBER	如果 PCM 降级，则非零
ACQUIRING	NUMBER	如果 PCM 升级，则非零
INVALID	NUMBER	如果 PCM 锁无效，则非零（系统失败后，锁可能变得无效）
FLAGS	NUMBER	LE 的进程级标志

80．V$LOCKED_OBJECT

这个视图列出系统上的每个事务处理所获得的所有锁。

列	数据类型	说明

XIDUSN	NUMBER	撤消的段号
XIDSLOT	NUMBER	位置号
XIDSQN	NUMBER	序列号
OBJECT_ID	NUMBER	被锁定的对象 ID
SESSION_ID	NUMBER	会话 ID
ORACLE_USERNAME	VARCHAR2(30)	Oracle 用户名
OS_USER_NAME	VARCHAR2(15)	OS 用户名
PROCESS	VARCHAR2(9)	OS 进程名
LOCKED_MODE	NUMBER	锁模式

81．V$LOCKS_WITH_COLLISIONS

这是一个并行服务器视图。利用这个视图来查找保护多个缓冲区的锁，其中每个曾经被强制写或强制读至少 10 次。很可能这些缓冲区由于被映射到相同的锁而经历过假的固定。

列	数据类型	说明
LOCK_ELEMENT_ADDR	NUMBER	撤消的段号

82．V$LOG

此视图包含来自控制文件的日志文件信息

列	数据类型	说明
GROUP#	NUMBER	日志组号
THREAD#	NUMBER	日志线程号
SEQUENCE#	NUMBER	日志序列号
BYTES	NUMBER	以字节表示的日志大小
MEMBERS	NUMBER	日志组中成员数
ARCHIVED	VARCHAR2	归档状态：YES、NO
STATUS	VARCHAR2(16)	日志状态。STATUS 列可具有表-6 中的值
FIRST_CHANGE#	NUMBER	日志中的最低 SCN
FIRST_TIME	DATE	日志中的第一个 SCN 的时间

表 B-6 给出日志 STATUS 列中的值。

表 B-6 STATUS 列的值

STATUS	含 义
UNUSED	表示联机重做日志文件从来没有写入过。如果不是当前重做日志，这是在刚增加的或刚好在 RESETLOGS 后的重做日志的状态
CURRENT	指出这是当前重做日志。这表示此重做日志是活动的。这个重做日志打开或关闭的
ACTIVE	表示此日志是活动的，但不是当前日志。需要进行崩溃恢复。有可能正用于块恢复。它能够或不能够归档
CLEARING	表示此日志在 ALTER DATABASE CLEAR LOGFILE 命令后正在作为空日志重建。在清除此日志后，这个状态变为 UNUSED
CLEARINGCURRENT	表示当前日志正由于关闭线程而被清除。如

INACTIVE	果在切换中存在某种故障（如写新日志标题的 I/O 错误），此日志可保持这种状态 表示实例恢复不再需要这个日志。它可能在用于介质恢复。它有可能归档，也有可能不归档

83．V$LOGFILE

这个视图包含重做日志文件的有关信息

列	数据类型	说明
GROUP#	NUMBER	重做日志组标识符号
STATUS	VARCHAR2	这个日志成员的状态：INVALID(文件不可访问)、STATE(文件的内容不完整)、DELETED (文件不再使用)、或空（文件正在使用）
MEMBER	VARCHAR2	重做日志成员名

84．V$LOGHIST

这个视图包含控制文件中的日志历史信息。此视图是为历史兼容而保留的。建议使用 C$LOG_HISTORY。更多的信息，请参阅 V$LOG_HISTORY。

列	数据类型	说明
THREAD#	NUMBER	日志线程号
SEQUENCE#	NUMBER	日志序列号
FIRST_CHANGE#	NUMBER	日志中的最低 SCN
FIRST_TIME	DATE	日志中的第一个 SCN 时间
SWITCH_CHANGE#	NUMBER	发生日志切换的 SCN；比日志中最高 SCN 更高的那个 SCN

85．V$LOGMNR_CONTENTS

这个视图包含日志历史信息。

列	数据类型	说明
SCN	NUMBER(15)	系统更改号
TIMESTAMP	DATE	时间戳
THREAD#	NUMBER	线程号
LOG_ID	NUMBER	日志 ID
XIDUSN	NUMBER	事务处理 ID 撤消段号
XIDSLOT	NUMBER	事务处理 ID 位置号
XIDSQN	NUMBER	事务处理 ID 日志序列号
RBASQN	NUMBER	RBA 日志序列号
RBABLK	NUMBER	RBA 块号
RBABYTE	NUMBER	RBA 字节偏移量
UBAFIL	NUMBER	UBA 文件号

UBABLK	NUMBER	UBA 块号
UBAREC	NUMBER	UBA 记录索引
UNASQN	NUMBER	UBA 撤消块序列号
ABS_FILE#	NUMBER	数据块绝对文件号
REL_FILE#	NUMBER	数据块相对文件号
DATD_BLK#	NUMBER	数据块号
DATA_OBJ#	NUMBER	数据块对象号
DATA_DOBJ#	NUMBER	数据块数据对象号
SEG_OWENR	VARCHAR2(30)	段拥有者
SEG_NAME	VARCHAR2(81)	段名
SEG_TYPE	NUMBER	段类型
TABLE_SPACE_NAME	VARCHAR2(30)	段的表空间名
ROW_ID	VARCHAR2(18)	行 ID
SESSION#	NUMBER	会话号
SERIAL#	NUMBER	系列号
USER_NAME	VARCHAR2(30)	用户名
SESSION_INFO	VARCHAR2(4000)	会话信息
ROLLBACK	NUMBER	回退请求
OPERATION	VARCHAR2(30)	操作
SQL_REDO	VARCHAR2(4000)	SQL 重做
SQL_UNDO	VARCHAR2(4000)	SQL 撤消
RS_ID	VARCHAR2(30)	记录集 ID
SSN	NUMBER	SQL 序列号
CSF	NUMBER	连续 SQL 标志
INFO	VARCHAR2(32)	通知信息
STATUS	VARCHAR2(16)	状态

86．V$LOGMNR_DICTIONARY

这个视图包含日志历史信息

列	数据类型	说明
TIMESTAMP	DATE	建立字典的时间
DB_ID	NUMBER	数据库 ID
DB_NAME	VARCHAR2(8)	数据库名
FILENAME	VARCHAR2(513)	字典文件名
DICTIONARY_SCN	NUMBER	建立字典时的系统更改号
RESET_SCN	NUMBER	建立字典时重置日志 SCN
RESET_SCN_TIME	NUMBER	获得重置日志 SCN 建立字典时的时间
ENABLED_THREAD_MAP	RAW(16)	建立字典时当前启用线程的位图
INFO	VARCHAR2(32)	信息/状态消息 BAD_DATE 表示字典文件的 SCN 与日志文件的 SCN 范围不匹配
STATUS	NUMBER	NULL 表示日志文件列表的有效字典文件。非 NULL 值表示进一步的信息作为文本串包含在 INFO 列中

87．V$LOGMNR_LOGS

此视图包含日志信息。

列	数据类型	说明
LOG_ID	NUMBER	指出日志文件。这个字段的值也在V$LOG 的 LOG_ID 列中给出
FILENAME	VARCHAR2(512)	文件名
LOW_TIME	DATE	文件中任意记录的最早日期
HIGL_TIME	DATE	文件中任意记录的最近日期
DB_ID	NUMBER	数据库 ID
DB_NAME	VARCHAR2(8)	数据库名
RESET_SCN	NUMBER	建立日志时重置日志 SCN
RESET_SCN_TIME	NUMBER	获得重置日志 SCN 建立字典时的时间
THREAD_ID	NUMBER	线程号
THREAD_SQN	NUMBER	线程序列号
LOW_SCN	NUMBER	切换入日志时分配的 SCN
NEXT_SCN	NUMBER	此日志后的 SCN。下一日志的低 SCN
INFO	VARCHAR2(32)	通知性的消息。将 MISSING_LOGFILE 值分配给其中所需日志文件从日志文件列表中遗漏的行项
STATUS	NUMBER	表示日志文件的状态。NULL 值表示一个有效的日志文件；非 NULL 值表示进一步的信息为一个文本串包含在 INFO 列中。如果所有日志文件成功地加到文件列表，状态值为 NULL

88．V$LOGMNR_PARAMETERS

此视图包含日志信息。

列	数据类型	说明
START_DATE	DATE	开始搜索的日期
END_DATE	DATE	开始搜索的日期
START_SCN	NUMBER	开始搜索的系统更改号
END_SCN	NUMBER	结束搜索的系统更改号
INFO	VARCHAR2(32)	通知性的消息。
STATUS	NUMBER	状态。NULL 值表示一个参数有效。非 NULL 值表示进一步的信息为一个文本串包含在 INFO 列中。

89．V$LOG_HISTORY

这个视图包含来自控制文件的历史信息。

列	数据类型	说明

THREAD#	NUMBER	归档日志线程号
SEQUENCE#	NUMBER	归档日志序列号
FIRST_TIME	NUMBER	归档日志中的第一项的时间（最低SCN）。此列以前名为 TIME
FIRST_CHANGE#	NUMBER	日志中最低 SCN。这个列以前名为 FIRST_CHANGE#
NEXT_CHANGE#	NUMBER	日志中最高 SCN。这个列以前名为 HOGH_CHANGE#
RECID	NUMBER	控制文件记录 ID
STAMP	NUMBER	控制文件记录时间戳

90．V$MLS_PARAMETERS

这是一个 Trusted Oracle 服务器视图，它列出 Trusted Oracle 服务器专用的安装参数。

91．V$MTS

这个视图包含优化多线程服务器的信息。

列	数据类型	说明
MAXIMUN_CONNECTIONS	NUMBER	每个调度程序可支持的连接的最大数目。这个值在启动时用 Net8 常量和其他端口专用的信息确定，或者可利用 MTS_DISPATCHERS 参数降低
SERVERS_STARTED	NUMBER	自实例启动以来启动的多线程服务器总数（但不包括启动中的启动的那些多线程服务器）
SERVERS_TERMINATED	NUMBER	自实例启动以来一次运行的服务器最大数目。如果这个值达到了 MTS_MAX_SERVERS 初始化参数设置的值，则应考虑提高 MTS_SERVERS 的值。更详细的信息，请参阅 MTS_SERVERS

92．V$MYSTAT

这个视图包含当前会话的统计数据。

列	数据类型	说明
SID	NUMBER	当前会话的 ID
STATISTIC	NUMBER	统计数据号
VALUE	NUMBER	统计数据值

93．V$NLS_PARAMETERS

这个视图包含 NLS 参数的当前值。

列	数据类型	说明
PARAMETERS	VARCHAR2	参数名： NLS_CALENDAR

	NLS_CHARACTERSET	
	NLS_CURRECY	
	NLS_DATE_LANGUAGE	
	NLS_ISO_CURRENCY	
	NLS_NUMEC_CHARACTERS	
	NLS_SORT	
	NLS_TERRITORY	
	NLS_UNION_CURRENCY	
	NLS_NCHAR_CHARACTERSET	
	NLS_COMP	
VALUE	VARCHAR2	NLS 参数值

94．V$NLS_VALID_VALUES

此视图列出 NLS 参数的所有有效值。

列	数据类型	说明
PARAMETERS	VARCHAR2（64）	参数名： LANGUATE；SORT； TERRITORY CHARACTERSETNLS 参数值
VALUE	VARCHAR2（64）	

95．V$OBJECT_DEPENDENCY

这个视图可用来确定当前共享池中装入的程序包、过程或游标依赖于什么样的对象。例如，它可与 V$SESSION 和 V$SQL 一道用来确定某个用户当前正在执行的 SQL 语句中所用的表。更详细的信息，请参阅 V$SESSION 和 V$SQL。

列	数据类型	说明
FROM_ADDRESS	RAW(4）	当前装入共享池中的过程、程序包或游标的地址
FROM_HASH	NUMBER	当前装入共享池中的过程、程序包或游标的散列值
TO_OWNER	VARCHAR2(64)	所依赖对象的拥有者
TO_NAME	VARCHAR2(1000)	所依赖的对象名
TO_ADDRESS	RAW(4)	所依赖对象的地址。这些地址可用来在 V$DB_OBJECT_CACHE 中查找关于对象的更多信息
TO_HASH	NUMBER	所依赖对象的散列值。这些值可用来在 V$DB_OBJECT_CACHE 中查找关于对象的更多信息
TO_TYPE	NUMBER	所依赖对象的类型

96．V$OBSOLETE_PARAMETER

此视图列出作废的参数。如果任何值为真，则应该检查为什么会这样。

列	数据类型	说明

NAME	VARCHAR2（64）	参数名
ISSPECIFIED	VARCHAR2（5）	在配置文件中是否给出此参数

97．V$OFFLINE_RANGE

这个视图显示控制文件中的数据文件中脱机信息。注意，每个数据文件的最后脱机范围保存在 DATAFILE 记录中。详细内容，请参阅 V$DATAFILE。

脱机范围是在数据文件的表空间 ALTER 为 OFFLINE NORMAL 或 READ ONLY，然后再 ALTER 为 ONLINE 或读写时建立的。注意，如果数据文件自身 ALTER 为 OFFLINE 或如果相应表空间 ALTER 为 OFFLINE IMMEDIATE 时，不建立脱机范围。

列	数据类型	说明
RECID	NUMBER	记录 ID
STAMP	NUMBER	记录时间戳
FILE#	NUMBER	数据文件号
OFFLINE_CHANGE#	NUMBER	脱机时的 SCN
ONLINE_CHANGE#	NUMBER	联机时的 SCN
ONLINE_TIME	NUMBER	脱机 SCN 的时间

98．V$OPEN_CURSOR

这个视图列出每个用户会话当前已经打开和分析的游标。

列	数据类型	说明
SADDR	RAW	会话地址
SID	NUMBER	会话标识符
USER_NAME	VARCHAR2(30)	登录到会话的用户
ADDRESS	RAW	与 HASH--VALUE 一道用来唯一地标识会话中执行的 SQL 语句
HASH_VALUES	NUMBER	与 ADDRESS 一道用来唯一地标识会话中执行的 SQL 语句
SQL_TEXT	VARCHAR2(60)	解析进入打开的游标的 SQL 语句的前 60 个字符

99．V$OPTION

这个视图列出与 Oracle 一道安装的选项。

列	数据类型	说明
PARAMETERS	VARCHAR2（64）	选项名
VALUE	VARCHAR2（64）	LANGUATE；SORT；TERRITORY CHARACTERSETNLS 参数值

100. V$PARALLEL_DEGREE_LIMIT_MTH

这个视图显示所有可用的并行度限额资源分配方法。

列	数据类型	说明

NAME	VARCHAR2（40）	并行度限客资源分配方法的名称

101. V$PARAMETER

这个视图列出初始化参数的有关信息。

列	数据类型	说明
NUM	NUMBER	参数名
NAME	VARCHAR2(64)	参数名 参数类型；1=布尔；2=串；3=整数
VALUE	VARCHAR2(512)	参数值
ISDEFAULT	VARCHAR2(9)	参数是否为缺省值
ISSES_MODIFIABLE	VARCHAR2(5)	TRUE=参数可用 ALTER SEEEION 更改。FALSE=参数不能用 ALTER SESSION 更改
ISSYS_MODIFIABLE	VARCHAR2(9)	IMMEDIATE=参数可用 ALTER SYSTEM 更改。DEFERRED=参数不能用 ALTER SYSTEM 更改
ISMODEIFIED	VARCHAR2(10)	指出参数怎样更改。如果执行一条 ALTER SESSION，则值将被 MODIFIED。如果执行一条 ALTER SYSTEM (它将导致所有当前登录会话的值被修改)，则参数值将为 SYS_ MODEIFIED
ISADJUSTED	VARCHAR2(5)	指出关系型数据库系统调整输入值即，参数值应该为素数，但用户输入一个非素数，因此关系型数据库系统将该值调整为下一个素数）
DESCRIPTION	VARCHAR2(64)	有关此参数的一个描述性的注释

102. V$PING

这是一个并行服务器视图。此视图等同于 V$CACHE 视图，但只显示至少 ping 了一次的块。这个视图包含来自当前实例的 SGA 中每个块的块标题的与特定数据库对象有关的信息。更多的信息，请参阅 V$CACHE。

列	数据类型	说明
FILE#	NUMBER	数据文件标识号（为找到文件名，可查询 DB_DATA_FILES 或 V$DBFILES）
BLOCKS#	NUMBER	块号
CLASS#	NUMBER	类号
STATUS	VARCHAR2(1)	块状态： FREE=当前不用 XCUR=互斥 SCUR=当前共享 READ=从磁盘读 MREC=处于介质恢复模式 IREC=处于实例恢复模式

XNC	NUMBER	由于与其他实例争用导致的空锁变换的 PCM 锁转换数。此例已作废，但为了历史兼容性仍然保留
FORCED_READS	NUMBER	锁必须从磁盘重新读取的次数，重新读取是由于其他实例通过在锁模式中请求此锁上的 PCM 锁，强迫它退出了此实例的高速缓存
FORCED_WRITES	NUMBER	由于这个实例已经搞坏了这个块并且其他实例已经以冲突的模式请求了这个锁上的 PCM 锁，而导致 DBWR 必须将这个块写入磁盘的次数
NAME	VARCHAR2(30)	包含该块的数据库对象名
PARTITION_NAME	VARCHAR2	非分区对象为 NULL
KIND	VARCHAR2(12)	数据库对象名。请参阅表 B-1
OWNER#	NUMBER	拥有者号
LOCK_ELEMENT_ADDR	RAW(4)	包含覆盖缓冲区的 PCM 锁的地址。如果不止一个缓冲区具有相同的地址，则相同的 PCM 锁也覆盖这些缓冲区
LOCK_ELEMENT_NAME	NUMBER	包含覆盖缓冲区的 PCM 锁的锁名。

103. V$PQ_SESSTAT

这个视图列出并行查询的统计数据。

注意：这个视图在以后的版本中将作废。

列	数据类型	说明
STATISTIC	VARCHAR2（30）	统计数据名，请参阅表 B-7
LAST_QUERY	NUMBER	最后处理的统计数据的值
SESSION_TOTAL	NUMBER	整个会话按时到达此点的统计值

已经为这个视图定义了表 B-7 中的统计数据（固定行）。在执行了查询或 DML 操作后，可利用从此视图得出的信息查看所利用的从进程，以及会话与系统的其他信息。

表 B-7 STATISTIC 列中的统计数据名

统计数据（固定行）	说明
Queries Parallelized	并行运行的查询数
DML Parallelized	并行运行的 DML 操作数
DFO Trees	执行 DFO 树的数目
Server Threads	使用的并行服务器的总数
Allocation Height	每个实例服务器请求数
Allocation Width	实例的请求数
Local Msgs Sent	发送的本地（实例内）消息数
Distr Msgs Sent	发送的远程（实例间）消息数

Local Msgs Recv'd	接收到的本地（实例内）消息数
Distr Msgs Recv'd	接收到的远程（实例间）消息数

104. V$PQ_SLAVE

这个视图列出实例上每个活动并行执行服务器的统计数据。

说明 此视图在未来的版本中将被一个称为 V$PX_PROCESS 的视图替代或废弃。

列	数据类型	说明
SLAVE_NAME	VARCHAR2（4）	并行执行的服务器名
STATUS	VARCHAR2（4）	并行执行服务器的当前状态（BUSY 或 IDLE）
SESSIONS	NUMBER	使用过这个并行执行服务器的会话数目
IDLE_TIME_CUR	NUMBER	当前会话中处理语句时空闲所占的时间总数
BUSY_TIME_CUR	NUMBER	当前会话中处理语句时忙所占的时间总数
CPU_SECS_CUR	NUMBER	用在当前会话的 CPU 时间总数
MSGS_SENT_CUR	NUMBER	处理当前会话的语句时发送的消息数
MSGS_RCVD_CUR	NUMBER	处理当前会话的语句时接收到的消息数
IDLE_TIME_TOTAL	NUMBER	此查询服务器空闲的时间总数
BUSY_TIME_TOTAL	NUMBER	这个查询服务器激活的时间总数
CPU_SECS_TOTAL	NUMBER	这个查询服务器用来处理语句的 CPU 时间总数
MSG_SENT_TOTAL	NUMBER	这个查询服务器发送的消息总数
MSG_RCVD_TOTAL	NUMBER	这个查询服务器接收到的消息总数

105. V$PQ_SYSSTAT

这个查询列出并行查询的系统统计数据。

说明 此视图在未来的版本中将被一个称为 V$PX_PROCESS_SYSSTAT 的视图替代或废弃。

列	数据类型	说明
STATISTIC	VARCHAR2(30)	统计数据名，请参阅表 B-8
VALUE	NUMBER	统计数据的值

为这个视图定义了表 B-8 中的统计数据（固定行）。在执行一个查询或 DML 操作后，可利用从 V$PQ_SYSSTAT得出的信息查看所利用的从进程数，以及其他系统信息。

表 B-8 STATISTIC 列中的统计数据名

统计数据（固定行）	说明
Server Busy	这个实例上当前忙的服务器数
Server Idle	这个实例上当前空闲的服务器数
Server Highwater	这个实例上参与>=1 等操作的活动服务器的

	数目
Server Sessions	这个实例上所有服务器中执行的操作总数
Server Started	这个实例上启动的服务器总数
Server Shutdown	这个实例上关闭的服务器总数
Server Cleaned Up	这个实例上由于进程死亡而清除的服务器总数
Queries Initiated	这个实例上启动的并行查询总数
DML Initiated	启动的并行 DML 操作的总数
DFO Trees	这个实例上执行的 DFO 树的总数
Local Msgs Sent	这个实例上发送的本地（实例内）消息总数
Distr Msgs Sent	这个实例上发送的远程（实例间）消息总数

106. V$PQ_TOSTAT

这个视图包含并行执行操作的统计数据。这些统计数据在查询完成后搜索，并且只在会话期间保留。此视图显示利用每个并行执行服务器在执行数的每个步骤处理的行数。它可以帮助确定查询执行中的扭曲问题。

说明 此视图在以后的版本中将被重新命名为 V$PX_TQSTAT。

列	数据类型	说明
DFO_NUMBER	NUMBER	区分查询的数据流运算符（DFO）树号
TQ_ID	NUMBER	查询内的表队列 ID，它表示查询执行树中两个 DFO 节点间的连接
SERVER_TYPE	ARCHAR2(10)	表队列中的角色：生成者/使用者/管理员
NUM_ROWS	NUMBER	生产/使用的行数
BYTES	NUMBER	生产/使用的字节数
OPEN_TIME	NUMBER	表队列保持打开的时间（秒）
AVG_LATENCY	NUMBER	表队列保持打开的时间（毫秒）
WAITS	NUMBER	退出队列时遇到的等待次数
TIMEOUTS	NUMBER	等待一个消息时超时的次数
PROCESS	VARCHAR2(10)	进程 ID
INSTANCE	NUMBER	实例 ID

107. V$PQ_PROCESS

这个视图包含当前活动进程的有关信息。在 LATCHWAIT 列表示进程等待某个栓锁时，LATCHSPIN 表示进程正在某个栓锁上循环，Oracle 进程在某个栓锁上等待前将在其上循环。

列	数据类型	说明
ADDR	RAW(4)	进程状态对象的地址
PID	NUMBER	Oracle 进程标识符
SPID	VARCHAR2	操作系统进程标识符
USERNAME	VARCHAR2	操作系统进程用户名。来自网络上的任意双任务，用户在用户名上都附加有“-T”
SERIAL#	NUMBER	进程系列号
TERMINAL	VARCHAR2	操作系统终端标识符

PROGRAM	VARCHAR2	正在进行中的程序
BACKGROUND	VARCHAR2	后台进程为 1；普通进程为 NULL
LATCHWAIT	VARCHAR2	进程正在等待的栓锁地址；如果没有，为 NULL
LATCHSPIN	VARCHAR2	进程正在其上循环的栓锁地址；如果没有，为 NULL

108. V$PROXY_ARCHIVEDLOG

这个视图包含归档日志备份的说明，这是与一个称为 Prpxy Copy 的新功能一道采用的。每行表示一个归档日志的备份。

列	数据类型	说明
RECID	NUMBER	代理拷贝记录 ID
STAMP	NUMBER	代理拷贝记录时间戳
DEVICE_TYPE	VARCHAR2(17)	拷贝驻留的设置类型
HANDLE	VARCHAR2(513)	代理拷贝句柄标识存储的拷贝
COMMENTS	VARCHAR2(81)	操作系统或存储子系统返回的注释。这个值只是通知性质的；不需要存储
MEDIA	VARCHAR2(65)	拷贝驻留的介质名。这个值只是通知性质的；不需要存储
DELETE	VARCHAR2(3)	如果设置为 YES，指出拷贝被删除，否则设置为 NO
THREAD#	NUMBER	重做线程号
SEQUENCE#	NUMBER	重做日志序列号
RESETLOGS_ CHANGE	NUMBER	写入这个日志时的数据库重置日志更改号
RESETLOGS_ TIME	DATE	写入这个日志时的数据库重置日志时间
FIRST_CHANGE#	NUMBER	归档日志中的第一个更改号
FIRST_TIME	DATE	第一个更改的时间
NEXT_CHANGE#	NUMBER	下一个日志中的第一个更改号
NEXT_TIME	DATE	下一个更改的时间戳
BLOCKS	NUMBER	以块表示的归档日志尺寸
BLOCK_SIZE	NUMBER	重做日志块尺寸
COMPLETION_ TIME	DATE	完成时间
ELAPSED_ SECONDS	NUMBER	占用的秒数

109. V$PROXY_ DATAFILE

这个视图包含数据文件和控制文件的描述，这是与称为 Proxy Copy 的新功能一道采用的。每行表示一个数据库文件的备份。

列	数据类型	说明
RECID	NUMBER	代理拷贝记录 ID
STAMP	NUMBER	代理拷贝记录时间戳

DEVICE_TYPE	VARCHAR2(17)	拷贝驻留的设置类型
HANDLE	VARCHAR2(513)	代理拷贝句柄标识存储的拷贝
COMMENTS	VARCHAR2(81)	操作系统或存储子系统返回的注释。这个值只是通知性质的；不需要存储
MEDIA	VARCHAR2(65)	拷贝驻留的介质名。这个值只是通知性质的；不需要存储
MEDIA_POOL	NUMBER	拷贝驻留的介质池。它与RecoveryManager；不需要存储
TAG	VARCHAR2(32)	代理拷贝标志
DELETE	VARCHAR2(3)	如果设置为 YES，指出拷贝被删除，否则设置为 NO
FILE#	NUMBER	绝对数据文件号，如果这是一个控制文件备份，则设置为 0
CREATION_CHANGE#	NUMBER	数据文件创建更改号
CREATION_TIME	DATE	数据文件创建时间戳
RESETLOGS_CHANGE#	NUMBER	进行这个拷贝时的数据文件的重置日志更改号
RESETLOGS_TIME	DATE	进行这个拷贝时的数据文件的重置日志时间戳
CHECKPOINT_CHANGE#	NUMBER	进行拷贝时的数据文件检查点更改号
CHECKPOINT_TIME#	DATE	进行拷贝时的数据文件检查点时间戳
ABSOLUTE_FUZZY_CHANGE	NUMBER	如果知道的话，为这个文件的任意块中最高更改
RECOVERY_FUZZY_CHANGE	NUMBER	介质恢复写到这个文件中的最高更改
RECOVERY_FUZZY_CHANGE	DATE	介质恢复写到这个文件中的最高更改时间戳
INCREMENTAL_LEVEL	NUMBER	如果这个备份为增量备份策略的组成部分，则为 0，否则为 NULL
ONLINE_FUZZY	VARCHAR2(3)	YES/NO。如果设置为 YES，则这个拷贝是在崩溃或脱机后立即做的（或者是在数据库打开后不正确地进行的拷贝的拷贝）。恢复需要应用下一个恢复标记前的所有重做以便使文件一致
BACKUP_FUZZY	VARCHAR2(3)	YES/NO。如果设置为 YES，则这是一个利用 BEGIN BACKUP/END BACKUP 技术进行的拷贝。注意：BEGIN BACKUP/END BACKUP 技术是在创建开放文件的代理备份时内部使用的。恢复需要应用 下一个恢复标记前的所有重做以便使这个备份文件一致
BLOCKS	NUMBER	以块表示的拷贝尺寸（也是进行拷贝时的数据文件尺寸）
BLOCK_SIZE	NUMBER	数据文件块尺寸

OLDEST_OFFLINE_RANGE	NUMBER	如果文件号为0（即，这是一个控制文件备份），则最旧的的脱机范围的RECID记录在这个控制文件拷贝中。数据文件拷贝为0
START_TIME	DATE	开始时间
COMPLETION_TIME	DATE	完成时间
ELAPSED_SECONDS	NUMBER	占用的秒数

110. V$PWFILE_USERS

这个视图列出从口令文件中导出的授予SYSDBA和SYSOPER权限的用户。

列	数据类型	说明
USERNAME	VARCHAR2(30)	包含在口令文件中的用户名
SYSDBA	VARCHAR2(5)	如果此列的值为TURE，则该用户可利用SYSDBA权限进行连接
SYSOPER	VARCHAR2(5)	如果此列的值为TURE，则该用户可利用SYSOPER权限进行连接

111. V$PX_PROCESS

此视图包含进行并行执行的会话的相关信息。

列	数据类型	说明
SERVER_NAME	VARCHAR2(4)	并行服务器名（P000、P001等）
STATUS	VARCHAR2(9)	并行服务器的状态。或者为In Use或者为Available
PID	NUMBER	进程标识符
SPID	VARCHAR2(9)	操作系统进程ID
SID	NUMBER	如果使用，为从服务器的会话ID
SERIAL#	NUMBER	如果使用，为从服务器的会话系列号

112. V$PX_PROCESS_SYSSTAT

这个视图包含进行并行执行的会话的有关信息。

列	数据类型	说明
STATISTIC	VARCHAR2(30)	统计数据名
VALUE	NUMBER	统计数据的值

STATISTIC列的值列在表B-9中。

表B-9 STATISTIC列中的统计数据名

统计数据（固定行）	说明
Servers In Use	当前执行并行操作的PX服务器号
Servers Available	执行并行操作可用的PX服务器号
Servers Started	系统必须创建PX服务器进程的次数

Server Shutdown	PX 服务器进程被关闭的次数。如果 PX 服务器进程最近未使用，将被关闭。它保持 Available 的时间长度由初始化参数 PARALLEL_SERVER_IDLE_TIME 控制。如果这个值较大，应该考虑增加该参数。由于免除了 PX 服务器进程的创建等待时间，所以将会提高性能
Server HWM	并发 PX 服务器进程的最大数目。如果这个数等于初始化参数 PARALLEL_MAX_SERVERS，应该考虑增加该参数。这样能够增加吞吐量，特别是如果系统利用不充分，并且 V$SYSSTAT 统计数据“Parallel operations downgraded to serial”较大时更是如此
Servers Cleaned Up	PMON 必须清除某个 PX 服务器的次数。它只应该在并行操作不正常结束时进行。如果这个数较大，建议查处其原因
Sessions	所有 PX 服务器建立的会话总数
Memory Chunks Allocs	PX 服务器分配的大内存块数
Memory Chunks Freed	空闲的大内存块数
Memory Chunks Current	当前使用的大内存块数
Memory Chunks HWM	当前分配内存块的最大数目
Buffers Allocated	某个消息缓冲区被分配的次数
Buffers Freed	某个消息缓冲区被释放的次数
Buffers Current	当前使用的消息缓冲区的数目
Buffers HWM	当前分配的消息缓冲区的最大数目

113. V$PX_SESSION

这个视图包含进行并行执行的会话的相关信息。

列	数据类型	说明
SADDR	NUMBER	会话地址
SID	NUMBER	会话标识符
SERIAL#	NUMBER	会话序列号
QCSID	NUMBER	并行协调程序的会话标识符
QCSERIAL#	NUMBER	并行协调程序的会话序列号
QCINST_ID	NUMBER	并行协调程序在其上运行的实例号
SERVER_GROUP	NUMBER	此并行服务器进程所属的服务器的逻辑组
SERVER_SET	NUMBER	此并行服务器进程所属的服务器逻辑组合。单个服务器组至少有两个服务器组合
SERVER#	NUMBER	某个并行服务器进程在一个服务器集合内的逻辑号
DEGREE	NUMBER	服务器集合所用的并行度
REQ_DEGREE	NUMBER	在发布语句且优先于任意资源、多用户或负载平衡降低时，用户所请求的并行度

114. V$PX_SESSTAT

这个视图包含进行并行执行的会话的有关信息。

列	数据类型	说明
SADDR	RAW(4)	会话地址
SID	NUMBER	会话标识符
SERIAL#	NUMBER	会话序列号
QCSID	NUMBER	并行协调程序的会话标识符
QCSERIAL#	NUMBER	并行协调程序的会话序列号
QCINST_ID	NUMBER	并行协调程序在其上运行的实例号
SERVER_GROUP	NUMBER	此并行服务器进程所属的服务器的逻辑组
SERVER_SET	NUMBER	此并行服务器进程所属的服务器逻辑组合。单个服务器组至少有两个服务器组合
SERVER#	NUMBER	某个并行服务器进程在一个服务器集合内的逻辑号
DEGREE	NUMBER	服务器集合所用的并行度
REQ_DEGREE	NUMBER	在发布语句且优先于任意资源、多用户或负载平衡降低时，用户所请求的并行度
STATISTIC#	NUMBER	统计数据号（标识符）
VALUE	NUMBER	统计数据值

115. V$QUEUE

这个视图包含多线程消息队列的相关信息。

列	数据类型	说明
PADDR	RAW(4)	拥有对列的进程地址
TYPE	VARCHAR2	队列的类型：COMMON(由服务器处理)、DISPATCHER
QUEUED	NUMBER	队列中的项数
WAIT	NUMBER	此队列中所有项等待的总时间。除以 TOTAL 得每项的平均等待时间
TOTALQ	NUMBER	曾经在队列中的项数

116. V$RECOVER_FILE

这个视图显示需要介质恢复的文件状态。

列	数据类型	说明
FILE#	NUMBER	文件标识号
ONLINE	VARCHAR2	联机状态：ONLINE、OFFLINE
ERROR	VARCHAR2	为什么此文件需要恢复：如果不知道原因，则为NULL，或者如果不需要恢复，为OFFLINE

		NORMAL
CHANGE#	NUMBER	恢复必须开始的 SCN
TIME	DATE	恢复必须开始的 SCN 时间

117. V$RECOVER_FILE_STATUS

这个视图对每个 RECOVER 命令的每个数据文件包含一行。此视图仅在 Oracle 进程进行恢复时包含有用信息。在 Recovery Manager 引导服务器进程完成恢复时，仅 Recovery Manager 能够查看这个视图中的相关信息。此视图对所有其他 Oracle 用户将是空的。

列	数据类型	说明
FILENUM	NUMBER	被恢复的文件数
FILENAME	VARCHAR2(257)	被恢复的数据文件名
STATUS	VARCHAR2(13)	恢复的状态。包含下列值： IN RECOVERY；CURRENT； NOT RECOVERED

118. V$RECOVERY_LOG

这个视图列出关于需要完成介质恢复的归档日志的信息。这些信息是从日志历史视图 V$LOG_HOSTORY 中导出的。更多的信息，请参阅“V$LOG_HISTORY”。

此视图仅在 Oracle 进程进行恢复时含有有用的信息。在 Recover Manager 引导服务器进程完成恢复时，仅 Recovery Manager 能够查看这个视图中的相关信息。此视图对所有其他 Oracle 用户将是空的。

列	数据类型	说明
THREAD	NUMBER	归档日志的线程号
SEQUENCE#	NUMBER	归档日志的序列号
TIME	VARCHAR2	日志中的第一项（最低 SCN）时间
ARCHIVE_NAME	VARCHAR2	在归档时，使用由 LOG_ARCHIVE_ FORMAT 指定的命名约定的文件名

119. V$RECOVERY_PROGRESS

此视图可用来跟踪数据库恢复操作以保证它们不停止，并估计完成正在进行中的操作所需的时间。

此视图是 V$SESSION_LONGOPS 的子视图。

列	数据类型	说明
TYPE	VARCHAR2(64)	正执行的恢复操作的类型
ITEM	VARCHAR2(32)	正被估计的项
SOFAR	NUMBER	迄今为止所完成的工作量
TOTAL	NUMBER	预期的总工作量

120. V$RECOVERY_STATUS

这个视图包含当前恢复进程的统计数据。此视图仅在 Oracle 进程恢复时含有有用信息。在 Recovery Manager 引导服务进程完成恢复时，仅 Recovery Manager 能够查看这个视图中的

相关信息。此视图对所有其他 Oracle 用户将是空的。

列	数据类型	说明
RECOVERY_ CHECKPOINT	DATE	恢复发生的时间点。如果没有应用日志，这是恢复开始的时间点
THREAD	NUMBER	当前正处理的重做日志线程号
SEQUENCE_ NEEDED	NUMBER	恢复进程所需的日志的序列号。如果不需要日志，这个值为 0
SCN_NEEDED	VARCHAR2(16)	恢复所需的日志的低 SCN。如果不知道或不需要日志，这个值为 0
TIME_EEEDED	DATE	创建日志时间。如果不知道时间或不需要日志，这个值为 88 年 1 月 1 日午夜
PREVIOUS_LOG_ NAME	VARCHAR2(257)	日志的文件名
PREVIOUS_LOG_ STATUS	VARCHAR2(13)	以前日志的状态。包含下列某个值：RELEASY；WRONE NAME；MISSING NAME；UNNEEDED NAME；NONE
REASON	VARCHAR(13)	合理的恢复将控制返回到用户。包含下列某个值：NEED LOG；LOG REUSED；THREAD DISABLED

121. V$REQDIST

这个视图列出 MTS 调度程序请求时间除以 12 个时间块或时间范围的柱状图统计数据。时间范围作为时间块的函数按指数增长。

列	数据类型	说明
BUCKET	NUMBER	时间块数：0-11；每个块的最大时间为（4*2^N）100 秒
COUNT	NUMBER	其完成总时间（包括等待时间）落入这个范围的请求计数

122. V$RESERVED_WORDS

这个视图给出 PL/SQL 编译程序使用的所有关键字列表。这个视图帮助开发者判断某个词是否已被用作语言中的关键字。

列	数据类型	说明
KEYWORD	VARCHAR2(64)	关键字名
LENGTH	NUMBER	关键字长度

123. V$RESOURCE

此视图包含资源的名称和地址的信息。

列	数据类型	说明
ADDR	RAW(4)	资源对象的地址

TYPE	NUMBER	资源类型。资源类型列在表 B-3 中
ID1	NUMBER	资源标识符#1
ID2	NUMBER	资源标识符#2

124. V$RESOURCE_LIMIT

这个视图显示某些系统资源的全局资源应用信息。可利用这个视图监控资源的使用情况，以便有必要时能够采取正确的措施。许多资源对应于表 B-10 所列出的初始化参数。

表 B-10 RESOURCE_LIMIT 列的值

资源名	相应的初始化参数
DISTRIBUTED_TRANSACTIONS	DISTRIBUTED_TRANSACTIONS：关于这个参数的更多信息，请参阅 DISTRIBUTED_ TRANSACTIONS
DML_LOCKS	DML_LOCKS：关于这个参数的更多信息，请参阅 DML_LOCKS
ENQUEUE_LOCKS	这个值是 Oracle 计算出来的。可使用 V$ENQUEUE_LOCK 视图获得关于排队锁的更多信息
ENQUEUE_RESOURCES	ENQUEUE_RESOURCES：关于这个参数的更多信息，请参阅 ENQUEUE_RESOURCES
LM_PROCESSES	LM_PROCS：关于这个参数的更多信息，请参阅 LM_PROCS
LM_RESOURCES	LM_RESS：关于这个参数的更多信息，请参阅 LM_RESS
LM_LOCKS	LM_LOCKS：关于这个参数的更多信息，请参阅 LM_LOCKS
MTS_MAX_SERVERS	MTS_MAX_SERVERS：关于这个参数的更多信息，请参阅 MTS_MAX_SERVERS
PARALLEL_SLAVES	PARALLEL_SLAVES：关于这个参数的更多信息，请参阅 PARALLEL_SLAVES
PROCESSES	PROCESSES：关于这个参数的更多信息，请参阅 PROCESSES
ROLLBACK_SEGMENTS	MAX_ROLLBACK_SEGMENTS：关于这个参数的更多信息，请参阅 MAX_ ROLLBACK_ SEGMENTS
SESSIONS	SESSIONS：关于这个参数的更多信息，请参阅 SESSIONTS
SORT_SEGMENT_LOCKS	这个值由 Oracle 计算
TEMPORARY_LOCKS	这个值由 Oracle 计算
TRANSACTIONS	TRANSACTIONS：关于这个参数的更多信息，请参阅 TRANSACTIONS

某些资源（如，那些 DML 所用的）有一个初始分配（软限制）和一个硬限制，从理论上说它是无限的（但实际上要受 SGA 尺寸的限制）。在 SGA 预定/初始化中，在 SGA 中为资源的 INITIAL_ALLOCATION 保留一个位置，但如果超出这个分配，可分配额外的资源，最多可达 LIMIT_VALUE 指定的值。CURRENT_UTILIZATION 列指出的是否已经超过初始分配。

在超出初始分配值时，从共享池中分配额外需要的资源，在那里它们必须与其他资源竞争空间。

恰当选择 INITIAL_ALLOCATION 的值将避免空间争用。对于大多数资源，INITIAL_ALLOCATION 和 LIMIT_VALUE 的值相同。超过 LIMIT_VALUE 的值将出现错误。

列	数据类型	说明
RESOURCE_NAME	VARCHAR2(30)	资源名（见表 B-10）
CURRENT_@@@@@	NUMBER	当前在用的数目（资源、锁或进程）
UTILIZATION	NUMBER	资源类型。资源类型列在表 B-3 中
MAX_UTILIZATION	NUMBER	自最后一个实例启动以来这个资源的最大消耗
INITIAL_ ALLOCATION	VARCHAR2(10)	初始分配。这个值将等于初始化参数文件中为此资源指定的值。（无限制分配为 UNLIMITED）
LIMIT_VALUE	VARCHAR2(10)	对资源和锁无限制。它可大于初始分配值。（无限制为 UNLIMITED）

125. V$ROLLNAME

这个视图列出所有联机回退段的名称。它只有在数据库打开时访问。

列	数据类型	说明
USN	NUMBER	回退（撤消）段号
NAME	VARCHAR2	回退段名

126. V$ROLLSTAT

这个视图包含回退段统计数据。

列	数据类型	说明
USN	NUMBER	回退段号
EXTENTS	NUMBER	回退段中的区数
RSSIZE	NUMBER	回退段以字节极的尺寸
WRITES	NUMBER	写到回退段的字节数
XACTS	NUMBER	活动的事务处理数
GETS	NUMBER	标题获得的数目
WAITS	NUMBER	标题等待的数目
OPTSIZE	NUMBER	回退段的最佳尺寸
HWMSIZE	NUMBER	回退段尺寸的高水位标记
SHRINKS	NUMBER	回退段尺寸减少的倍数
WRAPS	NUMBER	回退段缠绕的倍数
EXTENDS	NUMBER	回退段段尺寸扩展的倍数
AVESHRINK	NUMBER	平均收缩尺寸
AVEACTIVE	NUMBER	活动区随时间平均的当前尺寸
STATUS	VARCHAR2(15)	回退段状态
CUREXT	NUMBER	当前区
CURBLK	NUMBER	当前块

127. V$ROWCACHE

这个视图显示数据字典活动的统计数据。每行包含一个数据字典高速缓存的统计数据。

列	数据类型	说明
CACHE#	NUMBER	行高速缓存 ID 号
TYPE	VARCHAR2	父级或子级行高速缓存类型
SUBORDINATE#	NUMBER	子级集合号
PARAMETER	NUMBER	确定数据字典高速缓存中项数的初始化参数名
COUNT	NUMBER	高速缓存中项的总数
USAGE	NUMBER	包含有效数据的高速缓存项数
FIXED	NUMBER	高速缓存中的固定项数
GETS	NUMBER	请示数据对象信息的总数
GETMISSES	NUMBER	导致高速缓存未中的数据请求数
SCANS	NUMBER	扫描请求数
SCANMISSES	NUMBER	查找高速缓存中的数据扫描失败的次数
SCANCOMPLETES	NUMBER	对于子级项的列表，完全扫描列表的次数
MODIFICATIONS	NUMBER	插入、更新与删除的次数
FLUSHES	NUMBER	对磁盘进行刷新的次数
DLM_REQUESTS	NUMBER	DLM 请求的次数
DLM_CONFILICTS	NUMBER	DLM 冲突的次数
DLM_RELEASES	NUMBER	DLM 释放的次数

128. V$ROWCACHE_PARENT

这个视图显示数据字典中父级对象的信息。每个锁拥有一行，每个对象的等待者有一行。这个行显示拥有或请求的模式。对于没有拥有者的或等待者的对象，只显示单行。

列	数据类型	说明
INDX	NUMBER	行索引
HASH	NUMBER	散列值
ADDRESS	RAW(4)	父级对象的地址
CACHE#	NUMBER	父级高速缓存 ID
CACHE_NAME	VARCHAR2(64)	父级高速缓存名
EXISTENT	VARCHAR2(1)	此对象是否一个现存的对象
LOCK_MODE	NUMBER	锁被占有的模式
LOCK_REQUEST	NUMBER	锁被请求的模式
TXN	RAW(4)	当前缩定对象的事物处理
SADDR	RAW(4)	会话的地址
INST_LOCK_ REQUEST	NUMBER	只与并行服务器有关。实例锁被请求模式
INST_LOCK_ RELEASE	NUMBER	只与并行服务器有关。是否需要释放实例锁
INST_LOCK_TYPE	VARCHAR2	只与并行服务器有关。实例锁的类型
INST_LOCK_ID1	RAW(4)	只与并行服务器有关。与实例有关的 ID

INST_LOCK_ID2	RAW(4)	只与并行服务器有关。与实例有关的 ID
KEY	RAW(100)	只与并行服务器有关。键的内容

129. V$ROWCACHE_SUBORDINATE

这个视图显示数据字典中子级对象的信息。

列	数据类型	说明
INDX	NUMBER	索引
HASH	NUMBER	散列值
ADDRESS	RAW(4)	子级对象的地址
CACHE#	NUMBER	父级高速缓存 ID
SUBCACHE_NAME	VARCHAR2(64)	子级高速缓存名
EXISTENT	VARCHAR2(1)	此对象是否一个现存的对象
PARENT	RAW(4)	父级对象的地址
KEY	RAW(100)	只与并行服务器有关。键的内容

130. V$RSRC_CONSUMER_GROUP

这个视图显示与当前活动资源消耗者组有关的数据。

列	数据类型	说明
NAME	VARCHAR2(32)	使用者组名
ACTIVE_SESSIONS	NUMBER	这个使用者组中当前活动会话数
EXECUTION_ WAITERS	NUMBER	等待执行限额的当前活动会话数
REQUESTS	NUMBER	这个使用者组中执行的请求总数
CUP_WAIT_TIME	NUMBER	会话等待 CPU 的时间总数
CPU_WAITS	NUMBER	这个使用者组中所有会话必须等待 CPU 的时间数
CONSUMED_CPU_ TIME	NUMBER	这个使用者组中所有会话使用的 CPU 的时间总数
YIELDS	NUMBER	这个使用者组中所有会话必须的 CPU 时间总数
SESSIONS_QUEUED	NUMBER	等待变成活动的当前排队的会话计数

131. V$RSRC_CONSUMER_GROUP_CPU_MTH

这个视图显示资源使用者组的所有可用资源分配方法。

列	数据类型	说明
NAME	VARCHAR2(40)	CPU 资源分配方法的名称

132. V$RSRC_PLAN

这个视图显示所有当前活动资源计划的名称。

列	数据类型	说明

NAME	VARCHAR2(32)	资源计划的名称

133. V$RSRC_PLAN_CPU_MTH

这个视图显示资源计划的所有可用 CPU 资源分配方法。

列	数据类型	说明
NAME	VARCHAR2(32)	资源分配方法的名称

134. V$SESSION

这个视图列出每个当前会话的会话信息

列	数据类型	说明
SADDR	RAW(4)	会话地址
SID	NUMBER	会话标识符
SERIAL#	NUMBER	会话序列号。用来唯一地标识绘画对象。如果该会话结束且其他会话以相同的会话 ID 开始，则保证会话级的命令被应用到正确会话对象
AUDSID	NUMBER	审计会话 ID
PADDR	RAW(4)	拥有这个会话的进程地址
USER#	NUMBER	Oracle 用户标识符
USERNAME	VARCHAR(30)	Oracle 用户名
COMMAND	NUMBER	正进行的命令（分析的最后一个语句），关于值的列表，请参阅表 B-11
OWNERID	NUMBER	如果值为 2147483644，则此列的内容无效。否则此列包含拥有可移植会话的用户标符。对于利用并行从服务器的操作，将这个值解释为一个 48 字节的值。其低位两字节表示会话号，而高位字节表示查询协调程序的实例 ID
TADDR	VARCHAR2(8)	事务处理状态对象的地址
LOCKWAIT	VARCHAR2(8)	等待锁的地址；如果没有，为 NULL
STATUS	VARCHAR2(8)	会话的状态：ACTIVE (当前执行的 SQL)、INACTIVE、KILLED(标记为终止)、CACHED（为 Oracle*XA 使用而临时高速缓存）、SNIPED（会话不活动，在客户机上等待）
SERVER	VARCHAR2(9)	服务器类型：DEDICATED、SHARED、PSEUDO、NONE
SCHEMA#	NUMBER	模式用户标识符
SCHEMANANME	VARCHAR2(30)	模式用户名
OSUSER	VARCHAR(15)	操作系统客户机用户名
PROCESS	VARCHAR2(9)	操作系统客户机进程 ID
MACHINE	VARCHAR2(64)	操作系统机器名
TERMINAL	VARCHAR2(10)	操作系统终端名

PROGRAM	VARCHAR(48)	操作系统程序名
TYPE	VARCHAR2(10)	会话类型
SQL_ADDRESS	RAW(4)	与 SQL_HASH_VALUE 一道使用标识当前正在执行的 SQL 语句
SQL_HASH_VALUE	NUMBER	与 SQL_ADDRESS 一道使用标识当前正在执行的 SQL 语句
MODULE	VARCHAR2(48)	包含当前正在执行的模块名，正如由调用 DBMS_APPLICATION_INFO.SET_MODU LE 过程所设置
MODULE_HASH	NUMBER	上面 MODULE 的散列值
ACTION	VARCHAR2(32)	包含当前执行活动的名称，正如由调用 DBMS_APPLICATION_INFO.SET_ACTI ON 过程所设置
ACTION_HASH	NUMBER	上列活动名称的散列值
CLIENT_INFO	VARCHAR2(64)	由 DBMS_APPLICATION_INFO.SET_CLIENT_INFO 过程设置的信息
FIXED_TABLE_ SEQUENCE	NUMBER	此列包含一个数，每当会话完成一个数据库调用并且存在来自动态性能表的介入选择，它个数就增加。这个列可被性能监控程序用来监控数据库中的统计数据。每当性能监控程序查看数据库时，只需要查看当前活动的会话或在这个列中具有比上次性能监控程序所看到的最大值更大的值的会话即可。所有其他会话自上次性能监控程序查看数据库以来都是空闲的
ROW_WAIT_OBJ#	NUMBER	包含 ROW_WAIT_ROW#中指定的 ROW#的表的对象 ID
ROW_WAIT_FILE#	NUMBER	包含 ROW_WAIT_ROW#中指定的 ROWID 的数据文件的标识符。此列仅在会话当前正在等待其他事务处理提交并且 ROW_ WAIT_OBJ#不为-1 时有效
ROW_WAIT_BLOCK #	NUMBER	包含 ROW_WAIT_ROW#中指定的 ROWID 的数据文件的标识符。此列仅在会话当前正在等待其他事务处理提交并且 ROW_正在等待其他事务处理提交并且 ROW_
ROW_WAIT_ROW#	NUMBER	被锁定的当前 ROWID。此列仅在会话当前正在等待其他事务处理提交并且 ROW_ WAIT_OBJ#不为-1 时有效
LOGON_TIME	DATE	登录时间
LAST_CALL_ET	NUMBER	最后一次调用
PDML_STATUS	VARCHAR2(8)	如果 ENABLED，则会话正处于 PARALLEL DML 启用方式。如果 DISABLED，则此会话不支持

		PARALLEL DML 启用方式。如果 FORCED，则会话已经更改为强制 PARALLEL DLL
PDML_ENABLED	VARCHAR2(3)	此列已被 PDML_ENABLED 和 PDML_ STATUS 所替代。请看上列内容
FAILOVER_TYPE	VARCHAR2(10)	如果这个会话禁止失败切换，则为 NONE ，如果客户机能够在断开之后失败切换其会话，则为 SESSION，如果客户机还能失败切换正在进行的选择，则为 SELECT
FAILOVER_METHOD	VARXHAR2(3)	如果这个会话禁止失败切换，则为 NONE，如果客户机能够在断开之后重新连接，为 BASIC，如果备份实例能够支持它所支持的每个实例的所有连接，则为 PRECONNECT
FAILED_OVER	VARCHAR2(13)	如果运行在失败切换方式并进行过失败切换，为 TRUE，否则为 FALSE
RESOURCE_CON SU_MER_GROUP	VARCHAR2(32)	会话的当前资源使用者组的名称

表 B-11 列出对应于会话中正在执行的命令的数字值。这些值可出现在 V$AESSION COMMAND 列中。它们还出现在数据字典视图 SYS.AUDIT_ACTIONS 中。

表 B-11 COMMAND 列的命令数字

命令数字	命令
0	正在执行的命令。在进程处于过渡状态时出现
1	CREATE TABLE
2	INSERT
3	SELECT
4	CREATE CLUSTER
5	ALTER CLUSTER
6	UPDATE
7	DELETE
8	DROP CLUSTER
9	CREATE INDEX
10	DROP INDEX
11	ALTER INDEX
12	DROP TABLE
13	CREATE SEQUENCE
14	ALTER SEQUENCE
15	ALTER TABLE
16	DROP SEQUENCE

17	GRANT
18	REVOKE
19	CREATE SYNONYM
20	DROP SYNONYM
21	CREATE VIEW
22	DROP VIEW
23	VALIDATE INDEX
24	CREATE PROCEDURE
25	ALTER PROCEDURE
26	LOCK TABLE
27	NO OPERATION
28	RENAME
29	COMMIT
30	AUDIT
31	NOAUDIT
32	CREATE DATABASE LINK
33	DROP DATABASE LINK
34	CREATE DATABASE
35	ALTER DATABASE
36	CREATE ROLLBACK SEGMENT
37	ALTER ROLLBACK SEGMENT
38	DROP ROLLBACK SEGMENT
39	CREATE TABLESPACE
40	ALTER TABLESPACE
41	DROP TABLESPACE
42	ALTER SESSION
43	ALTER USER
44	COMMIT
45	ROLLBACK
46	SAVEPOINT
47	PL/SQL EXECUTE
48	SET TRANSACTION
49	ALTER SYSTEM SWITCH LOG
50	EXPLAIN
51	CREATE USER
52	CREATE ROLE
53	DROP USER
54	DROP ROLE
55	SET ROLE
56	CREATE SCHEMA
57	CREATE CONTROL FILE
58	ALTER TRACTING
59	CREATE TRIGGER
60	ALTER TRIGGER

61	DROP TRIGGER
62	ANALYZE TABLE
63	ANALYZE INDEX
64	ANALYZE CLUSTER
65	CREATE PROFILE
66	DROP PROFILE
67	ALTER PROFILE
68	DROP PROCEDURE
69	DROP PROCRDURE
70	ALTER RESOURCE COST
71	CREATE SNAPSHOT LOG
72	ALTER SNAPSHOT
73	DROP SNAPHOST
74	CREATE SNAPHOST
75	ALTER SNAPSHOT
76	DROP SNAPSHOT
79	ALTER ROLE
85	TRUNCATE TABLE
86	TRUNCATE CLUSTE
88	ALTER VIEW
91	CREATE FUNCTION
92	ALTER FUNCTION
93	DROP FUNCTION
94	CREATE PACKAGE
95	ALTER PACKAGE
96	DROP PACKAGE
97	CREATE PACKAGE BODY
98	ALTER PACKAGE BODY
99	DROP PACKAGE BODY

135. V$SESSION_CONNECT_INFO

这个视图显示当前会话的网络连接的有关信息。

列	数据类型	说明
SID	NUMBER	会话标识符（可用来将这个视图与 V$SESSON 视图进行连接）
AUTHENTICATION_TYPE	VARCHAR2(15)	怎样验证：OS、PROTOCOL 或 NETWORK
OSUSER	VARCHAR2(30)	这个数据库用户的外部用户名
NETWORK_SERVICE_BANNER	VARCHAR2(2000)	用于这个连接的每个 Net8 服务的产品标识（每个标识一行）

136. V$SESSION_CURSOR_CACHE

这个视图显示关于当前会话的游标使用信息。

说明 V$SESSION_CURSOR_CACHE 视图不是 SESSION_CACHED_CURSORS 初始化参

数的有效性度量。

列	数据类型	说明
MAXIMUM	NUMBER	高速缓存的游标最大数。一旦达到这个数，则为了打开更多的游标，将需要关闭某些游标。这个列中的值是从初始化参数 OPEN_CURSORS 导出的
COUNT	NUMBER	游标的当前数目（不管是否在用）
OPENED_ONCE	NUMBER	至少打开过一次游标
OPEN	NUMBER	打开游标的当前数目
OPENS	NUMBER	游标打开的累积总数减一。这是因为本查询 当前打开且正在使用的游标并为计入 OPENS 统计数据中
HITS	NUMBER	游标打开命中的累计总数
HIT_RATIO	NUMBER	找到打开游标的次数除以查找游标的次数所得到的比例

137. V$SESSION_EVENT

此视图列出会话等待某个事件的信息。注意，TIME_WAITED 和 AVERAGE_WAIT 列在不支持快速时间机制的平台。如果在这些平台上运行，并且希望此列反映真正的等待时间，则必须设置参数文件中的 TIMED_STATISTICS 为真。请注意，这样做对系统性能有轻微的负面影响。更多的信息，请参阅 TIMED_STATISTICS。

列	数据类型	说明
SID	NUMBER	会话的 ID
EVENT	VARCHAR2(64)	等待事件的名称
TOTAL_WAITS	NUMBER	此会话对这个事件的总等待次数
TOTAL_TIMEOUTS	NUMBER	此会话对这个事件的总等待超时次数
TIME_WAITED	NUMBER	此会话对这个事件的总等待时间数，以百分之一秒计
AVERAGE_WAIT	NUMBER	此会话对这个事件的平均等待时间数，以百分之一秒计
MAX_WAIT	NUMBER	此会话对这个事件的最大等待时间数，以百分之一秒计

138. V$SESSION_LONGOPS

这个视图显示某些长运操作的状态。它利用列 SOFAR 和 TOTALWORK 提供操作的进展报告。可监控下列操作状态：

- 散列值的创建
- 备份操作
- 恢复操作

列	数据类型	说明

SID	NUMBER	会话标识符
SERIAL#	NUMBER	会话系列号
OPENNAME	VARCHAR2(64)	操作名
TARGET	VARCHAR2(64)	在其上进行操作的对象
TARGET_DESC	VARCHAR2(32)	目标描述
SOFAR	NUMBER	至今为止完成的工作计数
TOTALWORK	NUMBER	总的工作量
UNITS	VARCHAR2(32)	工作度量单位
START_TIME	DATE	操作开始的时间
LAST_UPDATE_TIME	DATE	统计数据最后一次更新的时间
ELAPSED_SECOUNDS	NUMBER	操作开始以来占用的秒数
CONTEXT	NUMBER	前后关系
MESSAGE	VARCHAR2(512)	统计数据摘要消息

139. V$SESSION_OBJECT_CACHE

这个视图显示本地服务器（实例）上当前用户会话的对象高速缓存统计数据。

列	数据类型	说明
PINS	NUMBER	高速缓存中固定或查找出的对象数
HITS	NUMBER	发现对象已经在高速缓存中的对象固定的数目
TRUE_HITS	NUMBER	发现对象已经在高速缓存中并处于所需状态（从而不需要从数据库调入）的对象的数目
HIT_RATIO	NUMBER	HITS/PINS 的比例
TRUE_HIT_RATIO	NUMBER	TRUS_HITS/PINS 的比例
OBJECT_REFRESHES	NUMBER	高速缓存中利用来自数据库的新值进行刷新的对象数
CACHE_REFRESHES	NUMBER	整个高速缓存（所有对象）进行刷新的次数
OBJECT_FLUSHES	NUMBER	高速缓存中对数据库进行刷新的对象数
CACHE_FLUSHES	NUMBER	整个高速缓存（所有对象）对数据库进行刷新的次数
CACHE_SHRINKS	NUMBER	高速缓存收缩到最佳尺寸的次数
CACHE_OBJECTS	NUMBER	当前高速缓存的对象数
PINNED_OBJECTS	NUMBER	当前固定的对象数
CACHE_SIZE	NUMBER	以字节表示的高速缓存当前尺寸
OPTIMAL_SIZE	NUMBER	以字节表示的高速缓存最佳尺寸
MAXIMUM_SIZE	NUMBER	以字节表示的高速缓存最大尺寸

140. V$SESSION_WAIT

这个视图列出活动的会话等待的资源或事件。下面是优化时的考虑：

- 除显示的数值为十六进制以外，PIRAW，P2RAW，P3RAW 显示的值与 P1、P2、P3 列的值相同。
- WAIT_TIME 列在不支持快速时间机制的平台上包含值-2。如果在这些平台上运行，并且希望此列反映真正的等待时间，则必须设置参数文件中的 他—STATISTICS 为

真。请注意，这样做对系统性能有轻微的负面影响。更过的信息，请参阅TIMED_STATISTICS.

在以前的版本中，WAIT_TIME 列包含一个任意大的值（而不是一个负值）以表示此平台不具备快速时间机制。

- STATE 列解释 WAIT_TIME 的值并描述当前或最近的等待状态。

列	数据类型	说明
SID	NUMBER	会话标识符
SEQ#	NUMBER	唯一标识这个等待的序列号。每次等待都增加
EVENT	VARCHAR2(64)	会话等待的资源或会话
P1TEXT	VARCHAR2	第一个附加参数的描述
P1	NUMBER	第一个附加参数
P1RAW	RAW(4)	第一个附加参数
P2TEXT	VARCHAR2	第二个附加参数的描述
P2	NUMBER	第二个附加参数
P2RAW	RAW(4)	第二个附加参数
P3TEXT	VARCHAR2	第三个附加参数的描述
P3	NUMBER	第三个附加参数
P3RAW	RAW(4)	第三个附加参数
WAIT_TIME	NUMBER	非零值为会话的上一次等待时间。零值表示会话当前正在等待
SECONDS_IN_WAIT	NUMBER	等待的秒数
STATE	VARCHAR2	等待的状态

表 B-12 定义了 V$SESSION_WAIT 的 STATE 列的值。

STATE	值	说明
WAITING	0	当前正在等待的会话
WAITED UNKNOWN TIME	-2	未知的最后一次等待持续时间
WATIED SHORT TIME	-1	最后一次等待<百分之一秒
WAITED KNOWN TIME	>0	WAIT_TIME=最后一次等待持续时间

141. V$SESSTAT

这个视图给出用户会话的统计数据。为了找到与每个统计数据号（STATISTIC#）有关的统计数据名称，请参阅 V$STATNAME。

列	数据类型	说明
SID	NUMBER	会话标识符
STATISTIC#	NUMBER	统计数据名（标识符）
VALUE	NUMBER	统计数据值

142. V$SESS_IO

这个视图列出每个用户会话的 I/O 统计数据

列	数据类型	说明

SID	NUMBER	会话标识符
BLOCK_GETS	NUMBER	这个会话的块存取
CONSISTENT_GETS	NUMBER	此会话的一致性读取
PHYSICAL_READS	NUMBER	此会话的物理读取
BLOCK_CHANGES	NUMBER	此会话的块更改
CONSISTENT_ CHANGES	NUMBER	此会话的一致性更改

143. V$SGA

这个视图包含系统全局区的摘要信息。

列	数据类型	说明
NAME	VARCHAR2	SGA 组件名
VALUE	NUMBER	以字节表示的内存尺寸

144. V$SGASTAT

这个视图包含系统全局区的详细信息

列	数据类型	说明
NAME	VARCHAR2	SGA 组件名
BYTES	NUMBER	以字节表示的内存尺寸
POOL	VARCHAR2	指出 NAME 中内存驻留的池子。其值可以是：LARGE POO——从大型池中分配的内存 SHARED POOL—从共享池中分配的内存

145. V$SHARED_POOL_RESERVED

这个固定视图列出有助于优化共享存储池中保留池和空间的统计数据。

V$SHARED_POOL_RESERVED 视图的以下列仅在初始化参数 SHARED_POOL_ RESERVED_SIZE 设置为有效值时有效。更多的信息，请参阅 SHARED_POOL_ RESERVED _SIZE。

列	数据类型	说明
FREE_SPACE	NUMBER	保留列表中可用的空间总数
AVG_FREE_SIZE	NUMBER	保留列表上可用内存的平均尺寸
FREE_COUNT	NUMBER	保留列表上可用的内存片数量
MAX_FREE_SIZE	NUMBER	保留列表上最大可用内存片的尺寸
USED_SPACE	NUMBER	保留列表上使用的内存的总数
AVG_USED_SIZE	NUMBER	保留列表上使用的内存的平均尺寸
USED_COUNT	NUMBER	保留列表上使用的内存片的数量
MAX_USED_SIZE	NUMBER	保留列表上最大使用内存片的尺寸
REQUESTS	NUMBER	搜索保留列表查找可用内存片的次数
REQUESTS_MISSED	NUMBER	保留列表没有满足请求的可用内

		存片，从而开始利用 LRU 列表刷新对象的次数
LAST_MISS_SIZE	NUMBER	在保留列表没有满足请求的可用内存片，从而开始利用 LRU 列表刷新对象的次数
MAX_MISS_SIZE	NUMBER	在保留列表没有满足请求的可用内存片，从而开始利用 LRU 列表刷新对象时的最大请求未命中的请求尺寸

V$SHARED_POOL_RESERVED 视图的以下列包含的值在即使不设置初始化参数 SHARED_POOL_ RESERVED_SIZE 时也有效。

列	数据类型	说明
REQUEST_FAILURES	NUMBER	未找到满足请求的内存次数（即，出现 ORA_4031 错误的次数）
LAST_FAILURE_SIZE	NUMBER	最后失败请求的请求尺寸（即，出现 ORA_4031 错误的请求尺寸）
ABORTED_REQUEST _THRESHOLD	NUMBER	通知出现 ORA-4031 错误的的请求的最大尺寸
ABORTED_REQUEST	NUMBER	通知出现 ORA-4031 错误而不刷新对象的请求的数目
LAST_ABORTED_SIZE	NUMBER	返回一个 ORA-4031 错误而不利用 LRU 列表刷新对象的最后请求尺寸

146. V$SHARED_SERVER

这个视图包含共享服务器进程的有关信息。

列	数据类型	说明
NAME	VARCHAR2	服务器名
PADDR	RAW(4)	服务器的进程地址
STATUS	VARCHAR2	服务器状态： EXEC(执行 SQL) WAIT (ENQ)(等待一个锁) WAIT(SEND)(等待发送数据到用户) WAIT(COMMON)(空闲；等待用户请求) WAIT(RESET)(等待中断后重新设置的虚电路) QUIT(终止)
MESSAGES	NUMBER	处理的消息数
BYTES	NUMBER	所有消息中的总字节数
BREAKS	NUMBER	中断数
CIRCUIT	RAW(4)	当前正服务的虚电路地址
IDLE	NUMBER	以百分之一秒表示的空闲总时间

BUSY	NUMBER	以百分之一秒表示的繁忙总时间
REQUESTS	NUMBER	在此服务器的生存时间内从公用队列中取现的请求总数@@@@@

147. V$SORT_SEGMENT

这个视图包含一个给定实例中每个排序段的有关信息。此视图仅在表空间为TEMPORARY 类型时更新。

列	数据类型	说明
TABLESPACE_NAME	VARCHAR2(31)	表空间名
SEGMENT_FILE	NUMBER	第一个区的文件号
SEGMENT_BLOCK	NUMBER	第一个区的块号
EXTENT_SIZE	NUMBER	区尺寸
CURRENT_USERS	NUMBER	段的活动用户号
TOTAL_EXTENTS	NUMBER	段中区的总数
TOTAL_BLOCKS	NUMBER	段中块的总数
RELATIVE_FNO	NUMBER	排序段标题的相对文件号
USED_EXTENTS	NUMBER	分配给活动排序的区
USED_BLOCKS	NUMBER	分配给活动排序的块
FREE_EXTENTS	NUMBER	未分配给任意排序的区
FREE_BLOCKS	NUMBER	未分配给任意排序的块
ADDED_EXTENTS	NUMBER	区分配的数目
EXTENT_HITS	NUMBER	在池中找到未用区的次数
FREED_EXTENTS	NUMBER	解除分配的区的数目
FREE_REQUESTS	NUMBER	请求解除分配的数目
MAX_SIZE	NUMBER	曾经使用过的区的最大数目
MAX_BLOCKS	NUMBER	曾经使用过的块的最大数目
MAX_USED_SIZE	NUMBER	所有排序使用过的区的最大数目
MAX_USED_BLOCKS	NUMBER	所有排序使用过的块的最大数目
MAX_SORT_SIZE	NUMBER	单个排序使用过的区的最大数目
MAX_SORT_BLOCKS	NUMBER	单个排序使用过的块的最大数目

148. V$SORT_USAGE

这个视图描述排序用法。

列	数据类型	说明
USER	VARCHAR2(30)	请求临时空间的用户
SESSION_ADDR	RAW(4)	共享 SQL 游标的地址
SESSION_NUM	NUMBER	会话的系列号
SQLADDR	RAW(4)	SQL 语句的地址
SQLHASH	NUMBER	SQL 语句的散列值
TABLESPACE	VARCHAR2(31)	在其中分配空间的表空间
CONTENTS	VARCHAR2(9)	指出表空间是否是 TEMPORARY/ PERMANENT

SEGFILE#	NUMBER	初始区的文件号
SEGBLK#	NUMBER	初始区的块号
EXTENTS	NUMBER	分配给排序的区
BLOCKS	NUMBER	分配给排序的以块表示的区
SEGFNO	NUMBER	初始区的相对文件号

149. V$SQL

这个视图列出没有 GROUP BY 子句的共享 SQL 区的有关统计数据，而且对录入的原始 SQL 文本的每个孩子包含一行。

列	数据类型	说明
SQL_TEXT	VARCHAR2(1000)	当前游标的 SQL 文本的前 8 位字符
SHARABLE_MEM	NUMBER	这个子级游标使用的以字节表示的共享内存量
PERSISTENT_MEM	NUMBER	这个子级游标使用的以字节表示的持久内存量
RUNTIME_MEM	NUMBER	这个子级游标使用的临时结构尺寸
SORTS	NUMBER	为这个子级游标完成的排序数
LOADED_VERSIONS	NUMBER	如果装载了上下文堆栈，为 1，否则为 0
OPEN_VERSIONS	NUMBER	如果锁定了子级游标，为 1，否则为 0
USERS_OPENING	NUMBER	执行相应语句的用户数目
EXECUTIONS	NUMBER	自这个对象装入库高速缓存以来，在这个对象上的执行数目
USERS_EXECUTING	NUMBER	执行这个语句的用户数目
LOADS	NUMBER	对象被装入或重新装入的数目
FIRST_LOAD_TIME	VARCHAR2(19)	父级创建时间的时间戳
INVALIDATIONS	NUMBER	使子级游标无效的次数
PARSE_CALLS	NUMBER	这个子级游标的分析调用数目
DISK_READS	NUMBER	这个子级游标的磁盘读取数目
BUFFER_GETS	NUMBER	这个子级游标的缓冲区获取数目
ROWS_PROCESSED	NUMBER	分析 SQL 语句返回的总行数
COMMAND_TYPE	NUMBER	Oracle 命令类型定义
OPTIMIZER_MODE	VARCHAR2(10)	SQL 语句在其下执行的模式
OPTIMIZER_COST	NUMBER	优化程序给出这个查询的代价
PARSING_USER_ID	NUMBER	最初建立这个子游标的用户的用户 ID
PARSING_SCHMA_ID	NUMBER	最初用来建立这个子级游标的模式 ID
KEPT_VERSIONS	NUMBER	指出这个子级游标是否已经利用 DBMS_SHARED_POOL 程序包标记为固定在高速缓存中
ADDRESS	RAW(4)	这个子级游标的双亲的句柄地址
TYPE_CHK_HEAP	RAW(4)	这个子级游标的类型描述符的检查堆栈
HASH_VALUE	NUMBER	库高速缓存中的父级语句的散列值
CHILD_NUMBER	NUMBER	这个子级游标的编号
MODULE	VARCHAR2(64)	包含第一次分析 SQL 语句执行时的

		模块名，正如调用 DBMS_APPLICATION_INFO.SET_MODEL 所设置的那样
MODEL_HASH	NUMBER	在 MODULE 列中指定的模块的散列值
ACTION	VARCHAR2(64)	包含第一次分析 SQL 语句时执行的动作名，正如调用 DBMS_APPLICATION_INFO.SET_MODEL 所设置的那样
ACTION_HASH	NUMBER	在 ACTION 列中指定的动作的散列值
SERIALIZABLE_ABORT	NUMBER	每个游标的串行化事务处理失败，产生 ORA-8177 错误的次数

150. V$SQL_BIND_DATA

这个视图显示客户机为每个游标中每个明确绑定变量发送的绑定数据，前提是服务器中可得到这个数据，其中游标为查询这个视图的会话所有。

列	数据类型	说明
CURSOR_NUM	NUMBER	这个绑定的游标数
POSITION	NUMBER	绑定位置
POSITION	NUMBER	绑定位置
DATATYPE	NUMBER	绑定数据类型
SHARED_MAX_LEN	NUMBER	来自与这个绑定有关的共享游标对象的共享最大长度
PRIVATE_MAX_LEN	NUMBER	从客户机发送的这个绑定的私有最大长度
ARRAY_SIZE	NUMBER	数组元素的最大数目（仅对数组绑定）
PRECISION	NUMBER	精度（对于数值绑定）
SCALE	NUMBER	小数点位置（对于数值绑定）
SHARED_FLAG	NUMBER	共享绑定数据标志
SHARED_FLAG2	NUMBER	共享绑定数据标志(续)
BUF_ADDRESS	RAW(4)	绑定缓冲区内存地址
BUF_LENGTH	NUMBER	绑定缓冲区长度
VAL_LENGTH	NUMBER	实际的绑定值长度
BUF_FLAG	NUMBER	绑定缓冲区标志
INDICATOR	NUMBER	绑定指示器
VALUE	VARCHAR2(4000)	绑定缓冲区的内容

151. V$SQL_BIND_METADATA

这个视图显示客户机为每个游标中每个明确绑定变量提供的绑定元数据，其中的游标为查询这个视图的会话所拥有。

列	数据类型	说明
ADDRESS	RAW(4)	拥有这个绑定变量的子级游标的内存地址
POSITION	NUMBER	绑定位置
DATATYPE	NUMBER	绑定数据类型

MAX_LENGTH	NUMBER	绑定值的最大长度
ARRAY_LEN	NUMBER	数组元素的最大数目（仅对数组绑定）
BIND_NAME	VARCHAR2(30)	绑定变量名（如果使用）

152. V$SQL_CURSOR

这个视图显示与查询这个视图的会话有关的每个游标的调试信息。

列	数据类型	说明
CURNO	NUMBER	游标号
FLAG	NUMBER	游标中的标志设置
STATUS	VARCHAR2(9)	游标的状态：即，游标处于什么状态
PARENT_HANDLE	RAW(4)	指向父级游标句柄的指针
PARENT_LOCK	RAW(4)	指向父级游标锁的指针
CHILD_LOCK	RAW(4)	指向子级游标锁的指针
CHILD_PIN	RAW(4)	指向子级游标固定的指针
PERS_HEAP_MEM	NUMBER	从这个游标的永久堆栈中分配的内存总量
WORK_HEAP_MEM	NUMBER	从这个游标的工作堆栈中分配的内存总量
BIND_VARS	NUMBER	进入此游标的当前分析的查询中绑定位置的总数
DEFINE_VARS	NUMBER	进入此游标的当前分析的查询中绑定位置的总数
BIND_MEM_LOC	VARCHAR2(64)	绑定变量存入的内存堆栈；或者是 UGA 或者是 CGA
INST_FLAG	VARCHAR2(64)	安装对象标志
INST_FLAG2	VARCHAR2(64)	安装对象标志（续）

153. V$SQL_SHARED_MEMORY

该使徒显示有关内存快照共享的游标信息。在共享池中共享的每个 SQL 语句都有一个或多个与之相关的子对象。每个子对象包括几部分，其中一个是上下文堆栈，她拥有查询计划。

列	数据类型	说明
SQL_TEXT	VARCHAR2(1000)	此行正显示信息的共享游标子对象的 SQL 文本
HASH_VALUE	NUMBER	共享池的 SQL 文本上的散列值
HEAP_DESC	RAW(4)	本行所描述的子级游标的上下文堆栈的描述符地址
STRUCTURE	VARCHAR2(16)	如果本行描述的内存组块使用格式为“X:Y”的说明进行分配，则这是该说明的“X”部分
FUNCTION	VARCHAR2(16)	类似于 STRUCTURE 列，这是该说明的“Y”域
COMMENT	VARCHAR2(16)	这是在分配内存组块时提供的整个说明域

CHUNK_PTR	RAW(4)	这是分配内存组块的起始地址
CHUNK_SIZE	NUMBER	该组块所分配的内存数量
ALLOC_CLASS	VARCHAR2(8)	内存组块所属的内存类。它通常为 FREEABLE 或 PERMANENT
CHUNK_TYPE	NUMBER	进入回收函数表的一个索引，这些函数告诉服务器如何重建内存组块
CHUNK_TYPE	NUMBER	进入回收函数表的一个索引，这些函数告诉服务器如何重建内存组块
SUBHEAP_DESC	RAW(4)	如果上下文堆栈的父堆栈本身是一个子堆栈，则这是父堆栈描述符的地址

154. V$SQLAREA

本视图列出共享 SQL 区域中的统计数据，并且每个 SQL 串包含一行。它提供在内存中的、经过语法分析的、准备执行的 SQL 语句的统计数据。

列	数据类型	说明
SQL_TEXT	VARCHAR2(1000)	当前游标的 SQL 文本的前 80 个字符
SHARABLE_MEM	NUMBER	在父级游标中的所有子级游标的以字节为单位的所有共享内存的总和
PERSISTENT_MEM	NUMBER	在父级游标中的所有子级游标的以字节为单位的所有永久内存的总和
RUNTIME_MEM	NUMBER	所有子级游标的临时帧尺寸的总和
SORTS	NUMBER	所有子级游标完成的排序数量的总和
VERSION_COUNT	NUMBER	出现在父级游标高速缓存中的子级游标的数量
LOADED_VERSIONS	NUMBER	出现在高速缓存中并使用上下文堆栈（KGL 堆栈 6）被加载的子级游标的数量
OPEN_VERSIONS	NUMBER	在当前父级游标中打开的子级游标的数量
USERS_OPENING	NUMBER	所有打开的子级游标的用户数量
EXECUTIONS	NUMBER	在所有子级游标上的执行总数
USERS_EXECUTING	NUMBER	在所有子级游标上执行语句的用户总数
LOADS	NUMBER	对象被加载或重新加载的次数
FIRST_LOAD_TIME	VARCHAR2(19)	父级游标创建时间的时间戳
INVALIDATIONS	NUMBER	所有子级游标上无效的总数
PARSE_CALLS	NUMBER	对父级游标中子级游标分析调用的总数
DISK_READS	NUMBER	在所有子级游标上的磁盘读取总数
BUFFER_GETS	NUMBER	在所有子级游标上的缓冲区的总数
ROWS_PROCESSED	NUMBER	代表 SQL 语句处理的总的行数
COMMAND_TYPE	NUMBER	Oracle 命令类型定义
OPTIMIZER_MODE	VARCHAR2(10)	SQL 语句在其下执行的模式
PARSING_USER_ID	NUMBER	分析了这个父级游标下的每个第一游标的用户的用户 ID
PARSING_SCHMA_ID	NUMBER	用来分析这个子级游标的模式 ID
KEPT_VERSIONS	NUMBER	已经利用 DBMS_SHARED_POOL 程

		序包标记为保持的子级游标的数目
ADDRESS	RAW(4)	这个游标的父级游标的句柄地址
HASH_VALUE	NUMBER	库高速缓存中的父级语句的散列值
MODULE	VARCHAR2(64)	包含第一次分析 SQL 语句执行时的模块名，正如调用 DBMS_APPLICATION_INFO.SET_MODEL 所设置的那样
MODEL_HASH	NUMBER	在 MODULE 列中指定的模块的散列值
ACTION	VARCHAR2(64)	包含第一次分析 SQL 语句时执行的动作名，正如调用 DBMS_APPLICATION_INFO.SET_MODEL 所设置的那样
ACTION_HASH	NUMBER	在 ACTION 列中指定的动作的散列值
SERIALIZABLE_ABORT	NUMBER	串行化事务处理失败，产生 ORA-8177 错误的次数，所有子级游标上的总计

155. V$SQLTEXT

这个视图包含属于 SGA 共享 SQL 游标的 SQL 语句文本。

列	数据类型	说明
ADDRESS	RAW(4)	与 HASH_VALUE 一道用来唯一标识一个高速缓存游标
HASH_VALUE	NUMBER	与 ADDRESS 一道用来唯一标识一个高速缓存游标
PIECE	NUMBER	用来排序 SQL 文本片段的编号
SQL_TEXT	VARCHAR2	一列包含 SQL 文本的一个片段
COMMAND_TYPE	NUMBER	SQL 语句（SELECT、INSERT）等的类型代码

156. V$SQLTEXT_WITH_NEWLINES

这个视图除为了增加可读性不用空格替换 SQL 语句中的新行和表外，与 V$SQLTEXT 视图相同。更多的信息，请参阅 V$SQLTEXT。

列	数据类型	说明
ADDRESS	RAW(4)	与 HASH_VALUE 一道用来唯一标识一个高速缓存游标
HASH_VALUE	NUMBER	与 ADDRESS 一道用来唯一标识一个高速缓存游标
PIECE	NUMBER	用来排序 SQL 文本片段的编号
SQL_TEXT	VARCHAR2	一列包含 SQL 文本的一个片段
COMMAND_TYPE	NUMBER	SQL 语句（SELECT、INSERT）等的类型代码

157. V$STATNAME

这个视图显示列在 V$SESSTAT 和 V$SYSSTAT 表中的统计数据的解码统计数据名。详细信息，请参阅 V$SESSTAT 和 SYSSTAT。

列	数据类型	说明
STATISTIC#	NUMBER	统计数据号
NAME	VARCHAR2	统计数据名。参见表 B-13
CLASS	NUMBER	1（用户）；2（重做）；4（排队）；8（高速缓存）；16（操作系统）；32（并行服务器）；128（调试）

表 B-13 列出了 V$ATATNAME 返回的普通 Oracle 统计数据。

表 B-13 V$SESSETAT 和 V$SYSSTAT 的统计数据名

此会话使用的 CPU	调用开始时使用的 CPU
建立的 CR 块	引用的高速缓存提交 SCN
引用的提交 SCN	为 CR 转换的当前块
扫描的 DBWR 缓冲区	DBWR 检查点缓冲区写入
DBWR 检查点	DBWR 强制写入
找到的 DBWR 可用缓冲区	DBWRLRL 扫描
DBWR 构造可用空间请求	访问被写入缓冲区的 DBWR
DBWR 事务处理表写入	DBWR 累加扫描深度
并行化 DDL 语句	DBWR 撤消块写入
并行化 DML 语句	并行化 DFO 数
读写 OShars	OS 所有其他睡眠时间
OS Input 块	OS 数据页故障睡眠时间
OS Karnel 页故障睡眠时间	OS 强制上下文切换
接收到的 OS Messages	OS 消息发送
OS Minor 页故障	OS 其他系统陷阱 CPU 时间
OS Output 块	OS 进程堆尺寸
OS Process 堆栈尺寸	OS 信号接收
OS 交换	OS 系统调用 CPU 时间
OS 系统调用	OS 文本叶故障睡眠时间
OS 用户级 CPU 时间	OS 用户锁等待睡眠时间
OS 主动环境切换	OS 等待 CPU（延迟时间)
接收到的 PX 本地消息	PX 本地消息发送
接收到的 PX 远程消息	PX 远程消息发送
串行卸载的并行操作	SQL*Net 从客户机往返
SQL*Net 往返数据库连接	SCN 批处理的不必要的进程清除
完成的后台检查点	后台检查点开始
后台超时	未固定缓冲区计数
固定缓冲区计数	通过 SQL*Net 从客户机接收的字节
通过 SQL*Net 从数据库连接接收到的字节	通过 SQL*Net 发送到客户机的字节
通过 SQL*Net 发送到数据库连接的字节	取得快照的 SCN 调用：kcmgss
对 kcmgas 的调用	调用 kcmgss
对 kcmgrs 的调用	更改写入时间
清除和回退一致性读获取	只清除一致性读获取
簇键扫描块获取	簇键扫描
提交清除失败；块丢失	提交清除故障：缓冲区被写入
提交清除失败；回叫失败	提交清除故障：不能固定

提交清除失败：正在进行热备份
提交清除故障：写禁止
提交清除
一致性获取
游标验证
数据库块更改
数据库块获取
延迟（当前）块清除应用程序
查到灰缓冲区
排队转换
排队死锁
排队释放
排队请求
排队超时
排队等待
交死锁
执行计数
查到空闲缓冲区
请求的空闲缓冲区
全局高速缓存转换时间
全局高速缓存转换超时
全局高速缓存转换
全局高速缓存 cr 块接收时间
全局高速缓存 cr 块接收
全局高速缓存从磁盘读取
全局高速缓存 cr 超时
全局高速缓存延迟
全局高速缓存顺利转换
全局高速缓存空闲列表等待
全局高速缓存获取时间
全局高速缓存转换时间
全局高速缓存排队转换
全局锁同义词转换@@@@@
全局锁异步获取
全局锁转换时间
全局锁转换（异步）
全局锁转换（非异步）
全局锁获取时间
全局高速缓冲散栓锁等待
全局锁获取（异步）
全局锁获取（非异步）
全局锁释放
全局锁同义词转换@@@@@
全局锁同步获取
热缓冲区移动到 LRU 的标题
立即（CR）清除应用程序
立即（CURRENT）块清除应用程序
实例恢复数据库冻结计数
kcmccs 调用取得当前 scn
kcmccs 读取 scn 不转到 DCM
kcmccs 等待批处理
登录累计
当前登录
收到消息
消息发送
本地散列算法执行
本地散列算术失败
不用转到 DLM 取得的下一 scm
无缓冲区固定计数
非工作一致性读获取
累计打开的游标
当前打开的游标
打开替换文件
打开请求高速缓存替换
分析计数（硬）
分析计数（总计）
分析 CPU 时间
占用的分析时间
物理读取
物理写
直接物理读取
直接物理写
非检查点物理写
非检查点物理写
查到固定换冲区
处理最后非空闲时间
并行化查询
恢复数组读取时间
恢复数组读取
恢复块读取
递归调用
递归 cpu 用法
写入重做块
重做缓冲区分配项
重做项
重做日志空间请求
重做日志空间等待时间
重做日志切换中断
重做排序标记
重做尺寸
重估同步时间
重做同步写
重做消耗
重做写时间
重做写入程序闭锁时间
重做写
远程实例撤消块写入
远程实例撤消标题写入
应用回退更改撤消记录
回退段仅一致性读取获取

可串行终止	会话连接时间
会话游标高速缓存计数	会话游标高速缓存命中
会话逻辑读取	会话 pga 内存
最大会话 pga 内存	会话存储过程空间
会话 uga 内存	最大会话 uga 内存
排序（磁盘）	排序（内存
排序（行）	总计灰队列长度
按行标识符的表取数	表取数据连续行
获取的表扫描块	获得表扫描行
表扫描（高速缓存分区）	表扫描（直接读取）
表扫描（长表）	表扫描（行标识范围）
表扫描（短表）	总的文件打开
事务处理锁后台获取时间	事务处理锁后台获取
事务处理锁前台请求	事务处理锁前台等待时间
事务处理回退	事务处理表一致性读取回退
应用事务处理表一致性读取撤消记录	用户调用
用户提交	用户回退

其他信息：在某些平台上，NAME 和 CLASS 列将包含其他操作系统的专门数据。

158. V$SUBCACHE

这个视图显示当前装入库高速缓存内存的下级高速缓存的相关信息。此视图预排库高速缓存，每个库高速缓存对象的每个装入下级高速缓存印出一行。

列	数据类型	说明
OWNER_NAME	VARCHAR2(64)	包含这些高速缓存项的对象的拥有者
NAME	VARCHAR2(1000)	对象名
TYPE	NUMBER	对象类型
HEAP_NUM	NUMBER	包含这个下级高速缓存的堆栈号
CACHE_ID	NUMBER	下级高速缓存 ID
CACHE_CNT	NUMBER	这个对象中这个高速缓存的项数
HEAP_SZ	NUMBER	分配给这个堆栈的区空间量
HEAP_ALOC	NUMBER	从这个堆栈中分配的区空间量
HEAP_USED	NUMBER	从这个堆栈中利用的空间量

159. V$SYSSTAT

这个使徒列出系统统计数据。为找到与每个统计数据号（STATISTIC#）关联的统计数据名称，请参阅 V$STATNAME。

列	数据类型	说明
STATISTIC#	NUMBER	统计数据号
NAME	VARCHAR2	统计数据名。参见表 B-13
CLASS	NUMBER	统计数据类别：1（用户）；2（重做）；4（排队）；8（高速缓存）；16（操作系统）；32（并行服务器）；64（SQL）；128（调试）
VALUE	NUMBER	统计数据值
CLASS	NUMBER	统计数据类别：

160. V$SYSTEM_CURSOR_CACHE

除这个视图的信息是系统范围的外，其显示的信息与 V$SESSION_CURSOR_CACHE 的类似。更详细的信息，请参阅 V$SESSION_CURSOR_CACHE。

列	数据类型	说明
OPENS	NUMBER	游标打开的累计总数
HITS	NUMBER	游标打开命中的累计总数
HIT_RATIO	NUMBER	找到打开游标的次数除以查找游标的次数所得到的比例

161. V$SYSTEM_EVENT

这个视图包含所有等待某个事件的相关信息。注意，TIME_WAITED 和 AVERAGE_WAIT 列在那些不支持快速时间机制的平台上将包含零值。如果在这些平台上运行，并且希望此列反映真正的等待时间，则必须设置参数文件中的 TIMED_STATISTICS 为 TRUE。请注意，这样做对系统性能有轻微的负面影响。更多的信息，请参阅“TIMED_STATISTICS”。

列	数据类型	说明
EVENT	VARCHAR2(64)	等待事件的名称
TOTAL_WAITS	NUMBER	这个事件的总等待次数
TOTAL_TIMEOUTS	NUMBER	这个事件的总等待超时次数
TIME_WAITED	NUMBER	这个事件的总等待时间数，以百分之一秒计
AVERAGE_WAIT	NUMBER	这个事件的平均等待时间，以百分之一秒计

162. V$SYSTEM_PARAMETER

这个视图包含关于系统参数的信息。

列	数据类型	说明
NUM	NUMBER	参数号
NAME	VARCHAR2(64)	参数名
TYPE	NUMBER	参数类型：1=布尔；2=串；3=整数
VALUE	VARCHAR2(512)	赋给此参数的值
ISDEFAULT	VARCHAR2(9)	赋给此参数的值是否为缺省值
ISSES_MODIFIABLE	VARCHAR2(5)	此参数是否可用 ALTER SESSION 更改
ISSYS_MODIFIABLE	VARCHAR2(9)	此参数是否可用 ALTER SYSTEM 更改
ISMODIFIED	VARCHAR2(8)	指出参数怎样更改。如果执行一条 ALTER SESSION，则值将被 MODIFIED。如果执行一条 ALTER SYSTEM(它将导致所有当前登录会话的值被修改)，则参数值将为 SYS_MODIFIED
ISADJUSTED	VARCHAR2(5)	指出关系型数据库系统调整输入

DESCRIPTION	VARCHAR2(64)	值为一个更合适的值（即，参数值应该为素数，但用户输入一个非素数，因此关系型数据库系统将该值调整为下一个素数） 有关此参数的描述性文本

163. V$TABLESPACE

这个视图来自控制文件的信息。

列	数据类型	说明
TS#	NUMBER	表空间数
NAME	VARCHAR2(30)	表空间名

164. V$TEMPFILE

这个视图显示临时文件信息。

列	数据类型	说明
FILE#	NUMBER	数据文件号
CREATION_CHANGE#	NUMBER	创建系统更改号
CREATION_TIME	DATE	创建时间
TS#	NUMBER	表空间号
RFILE	NUMBER	表空间中的相对文件号
STATUS	VARCHAR2(7)	文件状态（脱机/联机）
ENABLED	VARCHAR2(10)	读和/或写的启用
BYTES	NUMBER	以字节表示的文件尺寸（来自文件标题）
BLOCKS	NUMBER	块中的文件尺寸（来自文件标题）
CREATE_BYTES	NUMBER	文件的创建尺寸（以字节表示）
BLOCK_SIZE	NUMBER	文件的块尺寸
NAME	VARCHAR2(513)	文件名

165. V$TEMPORARY_LOBS

这个视图显示临时 lobs。

列	数据类型	说明
SID	NUMBER	会话 ID
CACHE_LOBS	NUMBER	编号高速缓存临时 lobs
NOCACHE_LOBS	NUMBER	非高速缓存临时 lobs 的数目

166. V$TEMP_EXTENT_MAP

这个视图显示所有临时表空间的每个单元的状态。

列	数据类型	说明
TABLESPACE_NAME	NUMBER	这个单元所属的表空间名

FILE_ID	NUMBER	绝对文件号
BLOCK_ID	NUMBER	这个单元的起始块号
BYTES	NUMBER	区中的字节数
BLOCKS	NUMBER	区中的块数
OWNER	NUMBER	拥有这个单元（串）的实例
RELATIVE	FNO	相对文件号

167. V$TEMP_EXTENT_POOL

这个视图显示为某个给定实例高速缓存和使用的临时表空间的状态。注意，临时表空间高速缓存的装载是很迟钝的，该实例可以是静止的。关于所有实例的信息可使用GV$TEMP_EXTENT_POOL。

列	数据类型	说明
TABLESPACE_NAME	NUMBER	这个单元所属的表空间名
FILE_ID	NUMBER	绝对文件号
EXENTS_CACHED	NUMBER	已经高速缓存了多少个区
EXENTS_USED	NUMBER	实际正在使用的区有多少个
BLOCKS_CACHED	NUMBER	高速缓存的块数
BLOCKS_USED	NUMBER	使用的块数
BYTES_CACHED	NUMBER	高速缓存字节数
BYTES_USED	NUMBER	使用的字节数
RELATIVE_FNO	NUMBER	相对文件号

168. V$TEMP_PING

这个视图显示每个数据文件 ping 的块数。这个信息又可用来确定现有数据文件的访问模式，决定从数据文件块到 PCM 锁的新映射。

列	数据类型	说明
FILE_NUMBER	NUMBER	数据文件号
FREQUENCY	NUMBER	频率
X_2_NULL	NUMBER	文件中所有块从互斥到空的锁转换数
X_2_NULL_FORCED_WRITE	NUMBER	指定文件中的块由于互斥到空的转换而强制写的数目
X_2_NULL_FORCED_STALE	NUMBER	文件中的块由于互斥到空的转换而使其 STATE 的次数
X_2_S	NUMBER	文件中所有块从互斥到共享的锁转换数
X_2_S_FORCED_WRITE	NUMBER	指定文件中的块由于互斥到共享的转换而强制写的数目
X_2_SSX	NUMBER	文件中所有块从互斥到子共享互斥的锁转换数
X_2_SSX_FORCED_WRITE	NUMBER	指定文件中的块由于互斥到子共享互斥的转换而强制写的数目
S_2_NULL	NUMBER	文件中所有块从共享到空的锁转换数
S_2_NULL_FORCED_STALE	NUMBER	文件中的块由于共享到空的转换而使其 STATE 的次数

SS_2_NULL	NUMBER	文件中所有块从子共享到空的锁转换数
WRB	NUMBER	实例接收到一个跨实例调用这个文件的写入单个缓冲区的次数
WRB_FORECD _ WRITE	NUMBER	由于跨实例调用这个文件而写入单个缓冲区导致写入的块数
RBR	NUMBER	实例接收到一个跨实例调用这个文件的重用块范围的次数
RBR_FORECD_ WRITE	NUMBER	由于跨实例调用这个文件而重用块范围导致写入的块数
RBR_FORECD_ TATE	NUMBER	由于跨实例调用而重用块范围而使得这个文件中的块的 STATE 的数目
CBR	NUMBER	实例接收到一个跨实例调用这个文件的检查点块范围的次数
CBR_FORECD_ WRITE	NUMBER	由于跨实例调用这个文件的检查点跨范围所导致的写入这个文件中块的数目
NULL_2_X	NUMBER	指定文件中所有块从空到互斥的锁转换数
S_2_X	NUMBER	指定文件中所有块从共享到互斥的锁转换数
SSX_2_X	NUMBER	指定文件中所有块从子共享互斥到的互斥的锁转换数
NULL_2_S	NUMBER	指定文件中所有块从空到共享的锁转换数
NULL_2_SS	NUMBER	指定文件中所有块从空到子共享的锁转换数
OP_2_SS	NUMBER	打开的 PCM 锁子共享锁的数目。Oracle 8.1 中为 0

169. V$TEMP_SPACE_HEADER

这个视图显示每个临时表空间的每个文件的聚集信息，即当前已使用的内存量和每个空间标题的可用空间量的信息。

列	数据类型	说明
TABLESPACE_NAME	VARCHAR2(30)	临时表空间名
FILE_ID	NUMBER	绝对文件号
BYTES_USED	NUMBER	已使用的字节数
BLOCK_USED	NUMBER	已使用的块数
BYTES_FREE	NUMBER	可用字节数
BLOCKS_FREE	NUMBER	可用块数
RELATIVE_FNO	NUMBER	文件的相对文件号

170. V$TEMPSTAT

该视图包含有关文件读/写统计的信息。

列	数据类型	说明
FILE#	NUMBER	文件号

PHYRDS	NUMBER	完成物理读取的数量
PHYWRTS	NUMBER	DBWR 需要写操作的次数
PHYBLKRD	NUMBER	读取物理块的数量
PHYBLKWRT	NUMBER	写入磁盘的块数：如果所有的写操作是针对单个块的，则 PHYBLKWRT 与 PHYWRTS 相同
READTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为进行读操作所花费的时间（以百分之一秒计）；如果为 FALSE，则为 0
WRITETIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为进行写操作所花费的时间（以百分之一秒计）；如果为 FALSE，则为 0
AVGIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为 I/O 所花费的时间（以百分之一秒计）；如果为 FALSE，则为 0
LSTIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为进行最近的 I/O 所花费的时间（以百分之一秒
LSTIOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为进行最近的 I/O 所花费的时间（以百分之一秒计）；如果为 FALSE，则为 0
MINOTIM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为单个 I/O 所花费的最小时间（以百分之一秒计）；如果为 FALSE，则为 0
MAXIOWTM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为单个写操作所花费的最大时间（以百分之一秒计）；如果为 FALSE，则为 0
MAXIORTM	NUMBER	如果 TIMED_STATISTICS 参数为 TRUE，则为单个读操作所花费的最大时间（以百分之一秒计）；如果为 FALSE，则为 0

171. V$THREAD

该视图包含控制文件中的线程信息。

列	数据类型	说明
THREAD#	NUMBER	线程号
STATUS	VARCHAR2	线程状态：OPEN、CLOSE
ENABLE	VARCHAR2	启用状态：DISABLED，（启用）PRIVATE，或（启用）PUBLIC
ENABLE_CHANGE#	NUMBER	启用线程的 SCN
ENABLE_TIME	DATE	启用 SCN 的时间
DISABLE_CHANGE#	NUMBER	禁用线程的 SCN

DISABLE_TIME	DATE	禁用 SCN 的时间
GROUPS	NUMBER	分配给该线程的日志组的数量
INSTANCE	VARCHAR2	实例名
OPEN_TIME	DATE	打开线程的最近时间
CURRENT_GROUP#	NUMBER	当前日志组
SEQUENCE#	NUMBER	当前日志序列号
CHECKPOINT_CHANGE#	NUMBER	最近的检查点的 SCN
CHECKPOINT_TIME	DATE	最近的检查点的时间

172. V$TIMER

此视图列出以百分之一秒为单位的消耗时间。时间自启动以来进行度量，它是操作系统专有的，当该值溢出 4 字节时（大约为 497 字节），再次返回到 0。

列	数据类型	说明
SHECS	NUMBER	占用的时间（以百分之一秒计）

173. V$TRANSACTION

该视图列出系统的活动事务处理。

列	数据类型	说明
ADDR	RAW(4)	事务处理状态对象的地址
XIDUSN	NUMBER	撤消段的号
XIDSLOT	NUMBER	插曹号
XIDSQN	NUMBER	序列号
UBAFIL	NUMBER	撤消块地址（UBA）的文件号
UBABLK	NUMBER	UBA 块号
UBASQN	NUMBER	UBA 序列号
UBAREC	NUMBER	UBA 记录号
STATUS	VARCHAR2(16)	状态号
START_TIME	VARCHAR2(20)	起始时间（挂钟）
START_SCNB	NUMBER	起始系统更改号（SCN）的基点
START_SCNW	NUMBER	起始 SCN 包
START_UEXT	NUMBER	起始区号
START_UBAFIL	NUMBER	起始 UBA 文件号
START_UBABLK	NUMBER	起始 UBA 块号
START_UBASQN	NUMBER	起始 UBA 序列号
START_UBAREC	NUMBER	起始记录号
SER_ADDR	RAW(4)	用户会话对象地址
FLAG	NUMBER	标志
SPACE	VARCHAR2(3)	如果为空间事务处理，则为 Yes
RECURSIVE	VARCHAR2(3)	如果为递归事务处理，则为 Yes
NOUNDO	VARCHAR2(3)	如果为撤消事务处理，则为 Yes
PTX	VARCHAR2(3)	如果为并行事务处理，则为

		Yes，否则设为 No
PRV_XIDUSN	NUMBER	上一个事务处理的撤消段的号
PRV_XIDSLT	NUMBER	上一个事务处理的插槽号
PRV_XIDSQN	NUMBER	上一个事务处理的序列号
PTX_XIDUSN	NUMBER	父级 XID 的回退段号
PTX_XIDSLT	NUMBER	父级 XID 的插曹号
PTX_XIDSQN	NUMBER	父级 XID 的序列号
DSCN_B	NUMBER	独立的 SCN 基点
DSCN_W	NUMBER	独立的 SCN 包
USED_UBLK	NUMBER	已用的撤消块数量
USED_UREC	NUMBER	已用的撤消记录数量
LOG_IO	NUMBER	逻辑 I/O
PHY_IO	NUMBER	物理 I/O
CR_GET	NUMBER	一致性获取
CR_CHANGE	NUMBER	一致性更改

174. V$TRANSACTION_ENQUEUE

显示由事务处理状态对象拥有的锁。

列	数据类型	说明
ADDR	RAW(4)	锁状态对象的地址
KADDR	RAW(4)	锁地址
SID	NUMBER	会话拥有或获取锁的标识符
TYPE	VARCHAR2(2)	锁类型。TX=事务处理队列
ID1	NUMBER	锁标识符#1（取决于类型）
ID2	NUMBER	锁标识符#2（取决于类型）
LMODE	NUMBER	会话拥有锁的方式：0，无；1，空（NULL）；2，行-S（SS）；3，行-X（SX）；4，共享（S）；5，S/行-X（SSX）；6，独占（X）
REQUEST	NUMBER	会话请求锁的方式：0，无；1，空（NULL）；2，行-S（SS）；3，行-X（SX）；4，共享（S）；5，S/行-X（SSX）；6，独占（X）
CTIME	NUMBER	自当前方式授权以来的时间
BLOCK	NUMBER	该锁正在阻塞另一个锁

175. V$TYPE_SIZE

该视图列出各种数据库组件的尺寸以估计数据块的容量。

列	数据类型	说明
COMPONENT	VARCHAR2	组件名，如段或缓冲区标题
TYPE	VARCHAR2	组件类型
DESCRIPTION	VARCHAR2	组件说明
TYPE_SIZE	NUMBER	组件大小

176. V$VERSION

Oracle服务器的核心库组件的版本号。每个组件一行。

列	数据类型	说明
BANNER	VARCHAR2	组件名和版本号

177. V$WAITSTAT

此视图列出块争用统计信息。此表只能在启用计时统计信息时更新。

列	数据类型	说明
CLASS	VARCHAR2	块类
COUNT	NUMBER	针对块类的操作的等待次数
TIME	NUMBER	针对块类的操作的所有等待次数的总和

附录B 初始化参数

附录 C　基本概念解释

C.1 分页是什么？，如何避免分页？*

见《Oracle8/8I 开发使用手册》 P291

C.2 检查点是什么？*

C.3 什么是模式？

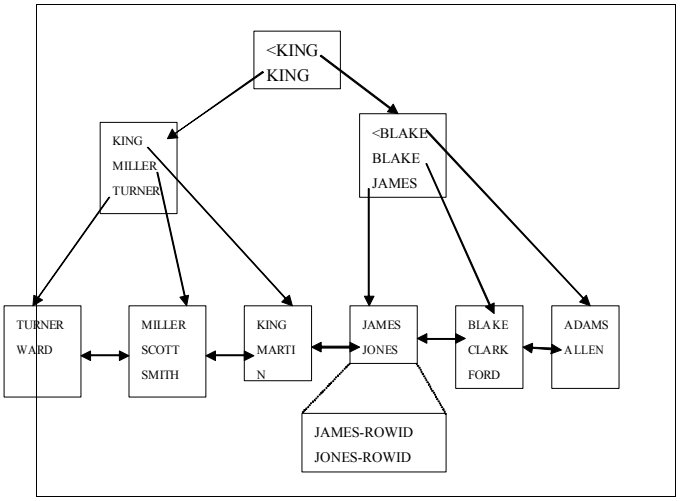
1.简单定义：
简单说，模式在 Oracle 系统里就是有组织的数据库对象。也就是说，模式关系数据库的对象逻辑地组织在一起，与数据库对象所存放的物理位置无关。
可以通过模式来访问到对象，如使用标准点表示法，有两个模式 S1 和 S2，它们都有表 t1；但可以通过 S1.t1 和 S2.t1 来区分。
2.模式与用户帐号的区别：
在 Oracle 系统中，数据库模式与用户名是一一对应。所以用户和模式有相同的名字。但是用户**拥有**某个对象；而模式**包含**某个对象。用户可以与其他的系统进行连接。

3.数据字典中的 SYS 模式：
Oracle 系统中的数据字典都存放在 SYS 模式中。

C.4 B 树索引

日常的声明都是 B 树索引。B 树索引是索引节点的有序树，每个节点有一个或多个索引记录，每个索引项对应表中的一行，包括：

- 该记录的索引列值；



B 树索引（B-tree）

- 该记录的ROWID值。

C.5 位图索引*

C.6 Oracle 的锁机制

C.6.1 自动锁和显式锁

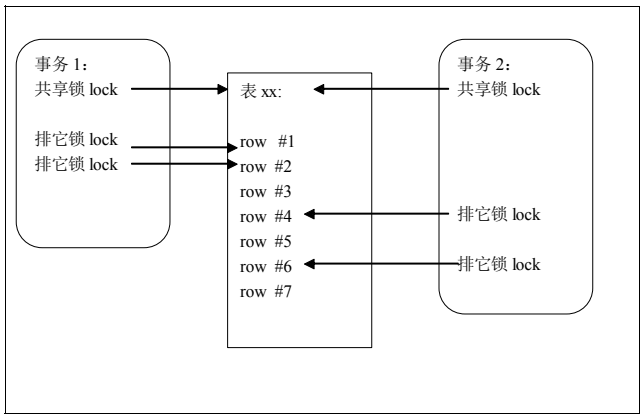
默认下，当进行数据库操作时，一个事务自动获得数据库资源所需的锁—自动锁。例如在更新(UPDATE)记录前，Oracle 要进行检查以确定没有其它打开的事务锁定该记录行：

- 如果有其它事务锁定当前的记录行，等待其它的事务结束并释放该行。
- 如果没有其它事务锁定该记录行，你的事务自动获得该记录的锁。你持有这个锁直到提交或回滚。

默认的锁（自动锁）对于大多数的应用是不够的。所以，Oracle 提供了另外的锁叫显式锁。比如对一个大表在操作前，对该表进行上锁比起每一条记录自动的默认锁效果更好。

C.6.2 锁的级别

Oracle 有两种锁：共享锁和排它锁。如下图：



共享锁和排它锁 允许对记录的操作示意图（lock 图）

1. 共享锁

数据库资源的共享锁给予一个事务对特定资源的共享访问。这样做的目的是允许事务的高度并发。

2. 排它锁

排它锁可以事务单独获得资源。图中的事务 1 对 row#1 和 row#2 获得排它锁。另外的事务只能在事务 1 提交或回滚后才获得排它锁。

C.6.3 DML 锁

当数据库在进行 INSERT、UPDATE 和 DELETE 时，自动获得表和索引的锁。为了保证数据库的高度并发并阻止破坏性交互，Oracle 可以提供行级锁和表级锁。

1.行级锁：
当进行 DML 操作时，该事务自动获得该表中有关行的排它锁。Oracle 通过行级锁来锁定数据，多个并发的事务可以更新相同的表而互不影响，除非他们都更新同一行。

例：

UPDATE emp set ename='Smith' where ename='smith' and deptno=10;

另外的方法是，可以用 SELECT … FOR UPDATE 来实现对特定行进行排它锁。

2.表级锁：

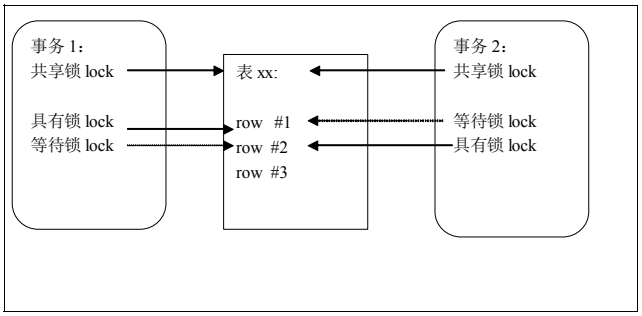
在事务中，可以允许在表级上获得共享锁和排它锁，当事务实现的 INSERT、UPDATE、DELETE 语句时，事务总是获得表上的共享锁。只有使用 LOCK TABLE 语句才能实现对表进行排它锁—表级锁。

例。使事务获得 customer 表的排它锁：

LOCK TABLE customer IN EXECLUSIVE MODE NOWAIT;

- 如果得不到排它锁，就会收到"NOWAIT"；
- 如果在声明时没有加"NOWAIT"，则要求得到排它锁的事务只能等待（干等）。

3.死锁：



处于死锁状态的两个事务（dead_lock 图）

- 事务 1 在 Row#1 有排它锁；并等待事务 2 释放 row#2;
- 事务 2 在 Row#2 有排它锁；并等待事务 1 释放 row#1;
- 两个事务互相等待对方，因而死锁。

死锁的原因是编程不科学所引起，许多情况下是由于不认真的编程引起。看下面例子。

事务 1	事务 2
UPDATE sales.parts Set onhand=onhand-10 where id=1;	UPDATE sales.parts Set onhand=onhand-10 where id=2;
UPDATE sales.parts Set onhand=onhand-10 where id=2;	UPDATE sales.parts Set onhand=onhand-10 where id=1;
等待事务 2 释放 id=2 的行	等待事务 1 释放 id=1 的行
ORA-00060: deadlock detected while waiting for resource.	

要避免死锁也很简单，只要在使用完该行后即时的释放即可。比如在更新语句后加判断后提交或回滚。可以简单将上面例子改为：

事务 1	事务 2
UPDATE sales.parts Set onhand=onhand-10 where id=1;	UPDATE sales.parts Set onhand=onhand-10 where id=2;
COMMIT;	COMMIT;
UPDATE sales.parts Set onhand=onhand-10 where id=2;	UPDATE sales.parts Set onhand=onhand-10 where id=1;
COMMIT;	COMMIT;

C.6.4 DDL 锁

1.排它的 DDL 锁：

CREATE、ALTER、DROP 一个对象时，Oracle 都为该对象产生排它锁。比如用 ALTER TABLE 语句加一个约束时，Oracle 就对该表获得排它锁。

2.共享的 DDL 锁：

数据库对象之间存在依赖关系的 DDL 语句需要共享的 DDL 锁。比如创建一个包，该包使用了许多数据库表；在创建该包时，你的事务获得该包上的排它 DDL 锁和引用表的共享 DDL 锁。

3.容易损坏分析锁(Breakable Parse Locks):

Oracle在共享池中的SQL语句（或PL/SQL程序）为每个对象持有分析锁。这个分析锁的获得是为了在某些对象**被删除或改变**时**使SQL区变为无效**。一个分析锁不接受任何DDL操作也不能取消竞争的DDL操作，因而叫 易碎的分析锁(*breakable parse lock*)。

C.6.5 锁存器与内部锁(Latches and Internal Locks)

锁存器和内部锁保护数据库与内存结构。这两个锁对于用户来说是不可访问的，因为用户不用去控制它们。下面内容将帮助你解释Oracle企业管理器或SQL*PLUS的LOCKS及LATCHES监视程序。

锁存器(Latches)

锁存器是保护SGA中共享数据结构的简单低级机制。例如，锁存器保护当前用户访问列表及高速缓冲区的数据结构。一个服务器或后台进程在查找这些数据结构或操作时需要一个短时间的锁存器。锁存器的执行与操作系统有关，特别是一个进程等待锁存器多长时间等。

内部锁（Internal Locks）

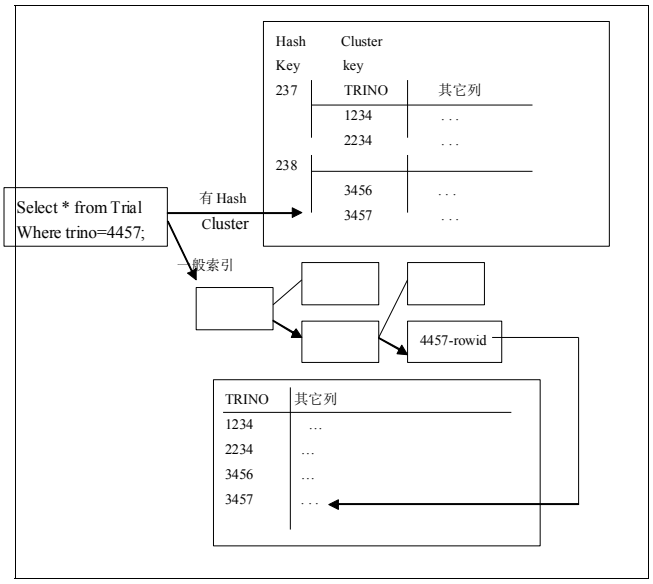
内部锁是比锁存器更复杂的和服务各种目的高级机制。

C.7 Oracle 的哈希簇(HASH CLUSTER)

哈希聚合是一种改善数据的查询性能的存储表的方法。在创建表时使用hash cluster并且将数据加载到相应的表，则Oracle以Hash 聚合来存储（物理上）表的行，并且以一个Hash函数的结果来取到这些行。
Oracle使用Hash函数来产生一个分类的数字值。这个数据值叫Hash值。它可以指定聚合的键值。这些Hash聚合键（如索引聚合键）可以是一个单列或复合列。Oracle对行的聚合键值使用这些Hash函数来查找或存储。

C.7.1 在 Hash Cluster 中数据是如何存放的

哈希聚合将相关的行（根据它们的Hash值）存储在一起。当创建一个哈希聚合时，Oracle为聚合的数据段（data segment）分配初始空间并且预报哈希键的平均大小。



具有 HASH CLUSTER和一般索引的I/O方法（Hash_cluster图）

C.7.2 Hash 键值

在一个Hash 簇中查找或存储一个行，Oracle 为该行的簇键值申请Hash函数。因而发生Hash值与簇中的一个数据块对应。每个Hsah簇的Hash数在创建时就是固定的并且由CREATE CLUSTER 语句的HASHKEYS参数确定。
HASHKEYS的值限制了Hash值的唯一性，改Hash值由Hash函数产生。Oracle为HASHKEYS的数进行四舍5入以接近素数（prime number.）。例如，设置HASHKEYS为100意思是对任何簇键值，Hash函数产生值在0 到100之间（有101个Hash值）。因此，在一个Hash 簇中行的分布是直接由所设置的HASHKEYS 参数所控制。为行的数给出一个大的Hash值数可以减少冲突（有两个簇键值一样就是冲突）。

C.7.3 Hash 函数

Hash 函数是一个用于申请一个簇键值的函数，它可以返回一Hash值。Oracle用这个Hash值分配Hash簇中的块的行。Hash函数的工作是改善簇键值当中有效行的最大分布。要达到这个目的，Hash函数必须最小化冲突的数目。

使用Oracle的内部 Hash 函数

当你建立一个簇时，你可以使用Oracle内部的Hash函数或旁边使用这些函数。内部Hash函数允许簇键值为单列或复合键。而且，簇键值可以包括任何数据类型（除LONG和LONG RAW外）。内部的Hash函数提供足够的簇键值分布。可减少冲突的数量。

以Hash函数指定簇键值

In cases where the cluster key is already a unique identifier that is uniformly distributed over its range, you might want to bypass the internal hash function and simply specify the column on which to hash.
Instead of using the internal hash function to generate a hash value, Oracle checks the cluster key value. If the cluster key value is less than HASHKEYS, then the hash value is the cluster key value. However, if the cluster key value is equal to or greater than HASHKEYS, Oracle divides the cluster key value by the number specified for HASHKEYS, and the remainder is the hash value. That is, the hash value is the cluster key value mod the number of hash keys.
Use the HASH IS parameter of the CREATE CLUSTER statement to specify the cluster key column if cluster key values are distributed evenly throughout the cluster. The cluster key must be comprised of a single column that contains only zero scale numbers (integers). If the internal hash function is bypassed and a non-integer cluster key value is supplied, then the operation (INSERT or UPDATE statement) is rolled back and an error is returned.

Specifying a User-Defined Hash Function

You can also specify any SQL expression as the hash function for a hash cluster. If your cluster key values are not evenly distributed among the cluster, then you should consider creating your own hash function that more efficiently distributes cluster rows more efficiently among the hash values.

For example, if you have a hash cluster containing employee information and the cluster key is the employee's home area code, then it is likely that many employees will hash to the same hash value. To alleviate this problem, you can place the following expression in the HASH IS clause of the CREATE CLUSTER statement:

```
MOD((emp.home_area_code + emp.home_prefix + emp.home_suffix), 101)
```

The expression takes the area code column and adds the phone prefix and suffix columns, divides by the number of hash values (in this case 101), and then uses the remainder as the hash value. The result is cluster rows more evenly distributed among the various hash values.

Allocation of Space for a Hash Cluster

As with other types of segments, the allocation of extents during the creation of a hash cluster is controlled by the INITIAL, NEXT, and MINEXTENTS parameters of the STORAGE clause. However, with hash clusters, an initial portion of space, called the *hash table*, is allocated at creation so that all hash keys of the cluster can be mapped, with the total space equal to SIZE * HASHKEYS. Therefore, initial allocation of space for a hash cluster also depends on the values of SIZE and HASHKEYS. The larger of (SIZE*HASHKEYS) and that specified by the STORAGE clause (INITIAL, NEXT, and so on) is used.

Space subsequently allocated to a hash cluster is used to hold the overflow of rows from data blocks that are already full. For example, assume the original data block for a given hash key is full. A user inserts a row into a clustered table such that the row's cluster key hashes to the hash value that is stored in a full data block. Therefore, the row cannot be inserted into the *root block* (original block) allocated for the hash key. Instead, the row is inserted into an overflow block that is chained to the root block of the hash key.

Frequent collisions can result in a larger number of overflow blocks within a hash cluster, thus reducing data retrieval performance. If a collision occurs and there is no space in the original block allocated for the hash key, then an overflow block must be allocated to hold the new row. The likelihood of this happening depends on the average size of each hash key value and corresponding data, specified when the hash cluster is created, as illustrated in Figure 10–11.(图 p10-60)

Figure 10–11 Collisions and Overflow Blocks in a Hash Cluster

If the average size is small and each row has a unique hash key value, then many hash key values can be assigned per data block. In this case, a small colliding row can likely fit into the space of the root block for the hash key. However, if the average hash key value size is large or each hash key value corresponds to multiple rows, then only a few hash key values can be assigned per data block. In this case, it

is likely that the large row will not fit in the root block allocated for the hash key value and an overflow block is allocated.

Single Table Hash Clusters

A single-table hash cluster can provide fast access to rows in a table. In an ordinary hash cluster, Oracle scans all the rows for a given table in the block, even if there actually happens to be just one row with the matching key. In a single-table hash cluster, if there is a one-to-one mapping between hash keys and data row, then Oracle can locate a row without scanning all the rows in the data block.
Oracle preallocates space for each hash key value when the single-table hash cluster is created. There cannot be more than one row per hash value (not the underlying cluster key value), and there cannot be any row chaining in the block. Otherwise Oracle scans all rows in that block to determine which rows match the cluster key.

See Also: *Oracle8i SQL Reference* for details about the SINGLE TABLE HASHKEYS clause of the CREATE CLUSTER statement

C.8 Oracle 的嵌套表(Nested table)

Oracle 的嵌套表(Nested table)

- 嵌套表是另一个表中的表；
- 一个嵌套表是某些行的集合；
- 嵌套表在主表中表示为一个列；
- 对主表中的每一条记录，嵌套表可以包括多个行；
- 嵌套表类似在一个表中存储一对多关系的一种方法；
- 简单说就是：**一个表在另外一个表中；**
- **对于主表中的每个行，在嵌套表中可以拥有多个行。**

典型例子：
- 一个部门信息表(department)，每个部门在任何时间内有多个项目在实施；
- 一个专家，可以在一年内发表多篇文章；可以参加几个项目。
- 动物饲养员可以养多个动物，每个动物都需要记录信息；
- 可以在 department 表中存放有关项目的信息；
- 不需要进行关联就可直接通过 department 表查询项目表中的记录；
- 在一般的严格关系中(所谓的范式)，department 和 project 两个表需要通过外部键才能实现；

创建嵌套表的步骤：

1.建立有关动物信息的**类型**：

CREATE or REPLACE TYPE **ANIMAL_TY** as object
(breed varchar2(25), --动物
name varchar2(25),-- 动物名字
birthdate date -- 出生日期
);

2.说明上面类型为嵌套表基础

要使用 **ANIMAL_TY** 类型作为一个嵌套表的基础，要创建一个新的类型，且该类型是与上面申明过的类型有关：

CREATE type **ANIMALS_NT** of ANIMAL_TY ;

基础类型的名称为 **ANIMALS_NT**。

3.用**基础类型**来创建嵌套表：

CREATE table BREEDER (
Breedername varchar2(25),--饲养员名字
Animals ANIMALS_NT
)
nested table ANIMALS store as ANIMALS_NT_TAB;

上面命令结果是：
- 第 1 列是 Breedername，它为普通的列；
- 第 2 列是 animals，它被定义为一个嵌套表 ANIMALS_NT，即：

CREATE table BREEDER
(breedername VARCHAR2(25),
animals ANIMALS_NT
);

嵌套表与主表分开存放，即：列 animals 中的数据存放在一个表中，同时列 name 中的数据存放在另外的独立表中。Oracle 将这两个表间保持多个指针。本例中，嵌套表的数据存放在 ANIMALS_NT_TAB 表中。

主表存放在 BREEDER；
ANIMALS 列组成的嵌套表存放在 ANIMALS_NT_TAB，因为说明语句是：
nested table ANIMALS store as ANIMALS_NT_TAB；

向嵌套表插入数据：

例如饲养员 Jan Janmes 只养 3 个动物，则：

```
INSERT into BREEDER values
(
'JANE JAMES', ANIMALS_NT(
                    ANIMAL_TY('DOG','BUTCH','31-MAR-97'),
                    ANIMAL_TY('DOG','ROVER','05-JUN-97'),
                    ANIMAL_TY('DOG','JULIO','10-JUN-97')
                    )
);
```

在数据字典中的情况：

- 表的列名存放在：User_tab_columns；
- 存放定义的类型在： User_type；
- 数据类型作为一个嵌套表的基础：User_coll_types

查询嵌套表：

要对嵌套表进行查询，要遵循：

- 不能直接对嵌套表进行查询，如 select birthdate from ANIMALS_NT where name='JULIO'; 语句是不正确的。
- 要同时对嵌套表的列和主表的列指定限定条件；
- 要用 THE 关键字(实际当成函数)来描述所查询的主表，不需指定嵌套表：

```
select NT.birthdate
    from  THE ( select animals from BREEDER where breedername='JANE JAMES' ) NT
where NT.name='JULIO' ;
```

这里就是对嵌套表中 ANIMALS 的列和主表的饲养者名字(breedername)进行限制。

即先将饲养者为 JANE JAMES 作为结果集，再将其当成条件来限制和动物的名字=JULIO 来查询。

更新嵌套表：

要对嵌套表进行更新，而不是对主表的值进行更新。比如饲养员 JANE JAMES 又新养了一个名字叫 MARCUS 的狗，则使用 insert 语句：

```
Insert into
THE ( select animals from BREEDER where Breedername='JANE JAMES' )
Values ( ANIMAL_TY('DOG','MARCUS','01-AUG-98')
      ) ;
```

对主表 BREEDER 的嵌套表 ANIMALS_NT 进行增加新记录，同时使这些记录建立关系。

嵌套表有关的字典：

DBA_NEST_TABLE

```
SQL> select table_name from dict where table_name like '%NEST%';

TABLE_NAME
------------------------------
ALL_NESTED_TABLES
DBA_NESTED_TABLES
USER_NESTED_TABLES

SQL> desc DBA_NESTED_TABLES
 名称                                      空?    类型
 ----------------------------------------- -------- ------------
 OWNER                                     VARCHAR2(30)
 TABLE_NAME                                VARCHAR2(30)
 TABLE_TYPE_OWNER                          VARCHAR2(30)
 TABLE_TYPE_NAME                           VARCHAR2(30)
 PARENT_TABLE_NAME                         VARCHAR2(30)
 PARENT_TABLE_COLUMN                       VARCHAR2(4000)
 STORAGE_SPEC                              VARCHAR2(30)
 RETURN_TYPE                               VARCHAR2(20)
```

Oracle9i 电子文档目录

文档编号	文档名称
A42523	Pro*FORTRAN_ Supplement to the Oracle_ Precompilers Guide Release 1.8

A42525	Programmer's Guide to the Oracle_Precompilers Release 1.8
A75154	Oracle Internet File system
A75172	Oracle Internet File System Developer's Guide
A77218	Legato Storage Manager Administrator's Guide 8.1.6
A81197	Oracle Internet File System Setup And Admin. Guide
A86647	Oracle Enterprise Manager database Tuning with the Oracle Tuning Pack 9.0.1
A86720	Oracle9i OLAP Services Developer's Guide to the OLAP DML 9.0.1
A87499	Oracle9i Replication 9.0.1
A87502	Oracle9i Replication Management API Reference 9.0.1
A87503	**Oracle9i Database Performance Guide and reference 9.0.1**
A87504	**Oracle9i Database Performance Methods 9.0.1**
A88717	Oracle Enterprise ManagerGetting Stared with Oracle Change Management pack 9.0.1
A88720	Oracle Enterprise Manager Getting Started with Oracle Management Pack for Oracle Application 9.0.1
A88749	**Oracle Enterprise Manager Getting Started with** Oracle Standard Management Pack 9.0.1
A88755	Oracle9i OLAP Services Concept and Administration Guide9.0.1
A88756	Oracle9i OLAP Services Developer's Guide to the Oracle OLAP API 9.0.1
A88758	Oracle Enterprise Manager Messages Manual 9.0.1
A88767	Oracle Enterprise Manager Administrator's Guide 9.0.1
A88768	Oracle SNMP Support Reference Guide 9.0.1
A88769	Oracle Enterprise Manager configuration Guide 9.0.1
A88770	**Oracle Enterprise Manager Concept guide 9.0.1**
A88771	Oracle Intelligent Agent User's Guide 9.0.1
A88783	Oracle Dynamic services User's and Administrator's Guide9.0.1
A88784	Oracle InterMedia Annotator user's Guide 9.0.1
A88785	Oracle InterMedia Java Classes User's Guide and Reference 9.0.1
A88786	Oracle InterMedia User's guide and Reference 9.0.1
A88787	Oracle Syndication Server User's Reference 9.0.1
A88805	Oracle Spatial User's Guide and Reference 9.0.1

A88806	**Oracle9i Application Developer's Guide 9.0.1**
A88807	Oracle9i Data Guard Broker 9.0.1
A88808	Oracle9i Data Guard Concepts and Administration 9.0.1
A88810	?
A88812	Oracle Universal installer Concept Guide 2.0.1
A88826	ISQL*Plus User's Guide and Reference 9.0.1
A88827	SQL*Plus User's Guide and Reference 9.0.1
A88828	SQL*Plus Quick reference 9.0.1
A88876	**Oracle9i Application Developer's Guide-Fundamentals 9.0.1**
A88890	**Oracle9i Application Developer's Guide-Advanced Queuing 9.0.1**
A88894	Oracle9i Application Developer's Guide-XML 9.0.1
A88895	Oracle9i Case Studies –XML 9.0.1
A88896	?
A88898	Oracle9i Supplied Java Packsges Reference 9.0.1
A88899	?
A89852	**Oracle9i Supplied PL/SQL Package and Type Reference 9.0.1**
A89856	**PL/SQL User's Guide and reference 9.0.1**
A89857	Oracle Call Interface Programmer's Guide 9.0.1
A89860	Oracle C++ Call Interface Programmer's Guide 9.0.1
A89861	Pro*C/C++ precompiler Programmer's Guide 9.0.1
A89865	Pro*COBOL precompiler Programmer's Guide 9.0.1
A89867	Oracle9i Real Application Cluster 9.0.1
A89868	Oracle9i Real Application Cluster Installation and configuration 9.0.1
A89869	Oracle9i Real Application Cluster Administration 9.0.1
A89870	Oracle9i Real Application Cluster Deployment and Performance 9.0.1
A89872	Oracle Enterprise Manager Event Test Reference Manual 9.0.1
A90120	Oracle9i Database New Features 9.0.1
A90121	Oracle Text Reference 9.0.1
A90122	Oracle Text Application Developer's Guide 9.0.1

A90125	**Oracle9i SQL Reference 9.0.1**
A90129	Oracle9i Sample Schemas 9.0.1
A90123	**Oracle9i Backup and Recovery Concept 9.0.1**
A90134	Oracle9i User-Manual Backup and Recovery User's Guide 9.0.1
A90135	**Oracle9i Recovery Manager User's Guide 9.0.1**
A90148	Oracle9i Security Overview 9.0.1
A90149	Oracle Label Security Administrator's Guide
A90151	Oracle Internet Directory Administrator's Guide 3.0.1
A90152	Oracle Internet Directory Application Developer's Guide 3.0.1
A90153	Oracle9i Directory Service Integration and Deploment Guide 9.0.1
A90154	**Oracle9i Net Service Administrator's Guide 9.0.1**
A90155	**Oracle9i Net Service Reference Guide 9.0.1**
A90164	**Oracle9i database Administrator Guide for Windows 9.0.1**
A90165	Oracle9i Network,Directory and security Guide 9.0.1
A90166	Oracle Call Interface Getting Started for windows 9.0.1
A90167	Pro*C/C++ Procompiler Getting Started for windows 9.0.1
A90168	Pro*COBOL Procompiler Getting Started for windows 9.0.1
A90169	Oracle COM automation Feature Developer's Guidefor Window 9.0.1
A90170	Oracle Developer's Guide for Microsoft Transaction Server for Windows 9.0.1
A90174	Legato Storage Manager Administrtor Guide for Windows 9.0.1
A90175	Legato Storage Manager Command Reference Guide
A90181	Oracle Workflow Guide Reference Guide
A90185	Oracle Workflow Server Installation Netes
A90187	Oracle9i CORBA Developer's Guide and reference 9.0.1
A90188	Oracle9i Enterprise Java Beans Developer's Guide and Reference 9.0.1
A90190	**Oracle9i database Reference 9.0.1**
A90191	Oracle9i Database Migration 9.0.1
A90192	**Oracle9i Database Utilities 9.0.1**
A90207	Oracle9i Java Tools Reference 9.0.1

A90208	Oracle JSP Support Java Server Pages Developer's Guide and Reference 1.1.2
A90209	Oracle9i Java developer's Guide 9.0.1
A90210	Oracle9i Java Storage Procedure Developer's Guide 9.0.1
A90211	Oracle9i JDBC Developer's Guide and Reference 9.0.1
A90212	Oracle9i SQLJ Developer's Guide and Reference 9.0.
A90213	Oracle9i servlet Engine Developer's Guide 9.0.1
A90214	Oracle9i Jpublisher User's Guide 9.0.1
A90235	Oracle Workflow Client Installation 2.6.1
A90236	Oracle9i Globalization Support Guide 9.0.1
A90237	Oracle9i Data Warehousing Guide 9.0.1
A90371	Oracle9i OLAP Services Concept and Administrator Guide for Windows 9.0.1
A95491	Oracle9i database Adminstrator's Guide for Windows 9.2
A96521	**Oracle9i database Adminstrator's Guide 9.2**
A96524	**Oracle9i Database Concept 9.2**
A96525	**Oracle9i Database Error Message 9.2**
A96533	**Oracle9i Database Performance Tuning Guide and Reference 9.2**
A96567	Oracle9i Advanced Replication 9.2
A90573	Oracle Advanced Security Administrator Guide 9.2
A96580	**Oracle9i Net Services Administrator Guide 9.2**
A96581	**Oracle9i Net Services Reference Guide 9.2**
A96591	Oracle9i Application Developer's Guide-Large Objects(LOBS)9.2
A96594	Oracle9i Application Developer's Guide-Object Relational Features 9.2
A97297	**Oracle9i Administrator's Reference(9.2)for UNIX、AIX、Tru64、HP-UX、Solaris**