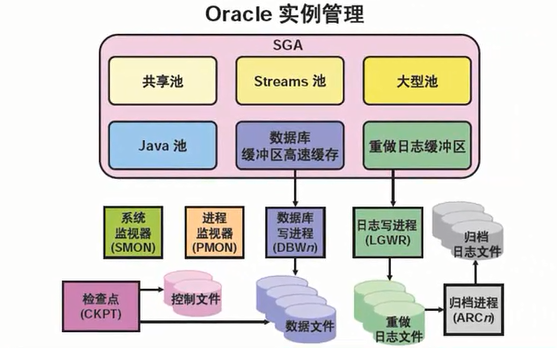
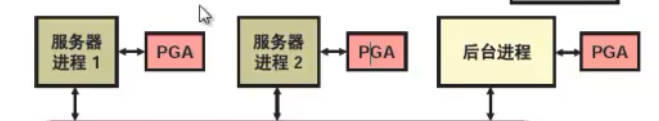
体系结构

# 概论



## 资源

CPU、IO、内存

## server process

sql通过传递过来，server process会找执行计划、没有的话，解析。

首先读buffer cache的数据，看是否有没有，如果没有，会去磁盘读（物理I/O）数据到buffer cache。

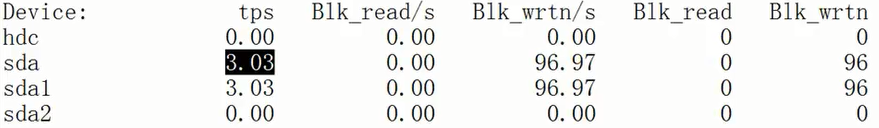
## 命中率

L/(L+P) L逻辑读 ，P 物理读

命中率低，肯定有问题；命中率高，不一定没有问题。

例如：L 10W次，P 1W次，虽然命中率不低，但是P I/O不低，服务器性能还是不高。

iostat 1 10



tps 每秒传输次数

blk\_read 每秒物理读的块数

## shared pool

用户提交的命令：解析、执行

缓冲执行计划

访问shared pool和执行，是server process做。

## buffer cache

缓冲dbf的数据

## 实例

实例是访问数据库的方法，实例有内存和后台进程组成；

一个实例只能连接一个数据库。数据库可以连接多个实例（RAC）

数据库不对外开放，只对实例开放。我们通过实例访问数据库

select \* from v$sgainfo; --sga详细概要

select \* from v$sgastat; --sga 统计

select \* from v$sgastat where name like '%SQL%';

c/stat/info

log buffer

运行sql过程

分析执行计划，按照CBO，选择最优计划。

LRU（list recent use）队列 最近最少使用

### 内存历史

8i前（含8i）

每块内存独立的设置，修改内存大小需要重新启动实例。

9i

1. 整合客户端内存——pga
2. 可以动态修改内存的大小

10g

1. 整合sga
2. 自动的调节每个内存的大小

11g

1. 整理所有的内存

查看后台进程

desc v$bgprocess

select PADDR,NAME,DESCRIPTION ,ERROR from v$bgprocess;

10g 158个

11g 295个

PADDR NAME DESCRIPTION ERROR

---------------- ----- ------------------------------ ----------

00 --00是预留的

select PADDR,NAME,DESCRIPTION ,ERROR from v$bgprocess where NAME like 'DBW%';

11g 36个 1-10+a-z=36

## 后台进程

### dbwn

dbwn，数据库写进程，将脏数据写到数据文件。

此进程最少一个，最多每个版本都有限制（所有版本都不能超过CPU个数）

### lgwr

没有下标，只有一个日志进程在写。lgwr将日志缓冲区日志写到日志文件，顺序写，循环使用。

commit时，写的是日志文件，没有写dbwn。在dbrn前，先写lgwr。

### smon 系统监控进程

主内，sga内部的维护

1. 例程恢复
   1. 前滚重做日志中的改变
   2. 打开数据库供用户访问
   3. 回退未提交的事务处理
2. 合并空闲空间（使用时间长了，碎片整理）
3. 回收临时段

rollback回退一分钟，在回退时，select查不了，会提示忙。

查某表：

select table\_name from dba\_tables where table\_name like '%SCN%';





### pmon 进程检测进程，负责和进程有关

主外，对sga外部的维护

进程失败后，后台进程PMON 通过下面的方法进行清理：

1. 回退用户的当前事务处理
2. 释放当前保留的所有表锁或行锁
3. 释放用户当前保留的其他资源(关闭server process，回收资源)
4. 重启已失效的调度程序

### CKPT

1. 在检查点发信号给DBWn
2. 使用检查点信息更新数据文件的标头
3. 使用检查点信息更新控制文件

启动检查点的原因如下：

1. 确保定期向磁盘写入内存中发生修改的数据块，一边在系统或数据库失败时不会丢失数据
2. 缩短例程恢复所需的时间。只需处理最后一个检查点后面的重做日志条目一启动恢复操作。
3. 确保提交的所有数据在关闭期间均已写入数据文件

有CKPT写入的检查点信息，包括检查点位置、系统更改号、重做日志中恢复操作的起始位置以及有关日志的信息等。

注：CKPT并不将数据块写入磁盘，或将重做块写入联机重做日志。

DBWn写，不一定有检查点；检查点发生，肯定有写操作。

select name ,checkpoint\_change# from v$datafile

进程名很多0，都是备用进程。

### ARCn

#### 缓冲区数据的状态

已连接、干净、空闲或未使用、脏

1. 已连接

server process对数据块正在读或正在写的瞬间，数据块被pin住，此时状态是已连接

1. 空闲或未使用

新数据需要放入buffer cache，优先使用空闲或未使用

1. 干净

在物理内存中，实际有对应的一个未使用的内存块

# shared pool

SQL语句执行：解析、执行、获取数据。

解析：

解析的过程是一个相当复杂的过程，它要考虑各种可能的异常情况，比如SQL语句涉及到的对象不存在、提交的用户没有权限等等。而且，还需要考虑如何执行SQL语句，采用什么方式去获取数据等。解析的最终结果是要产生oracle自己内部的执行计划，从而指导SQL的执行过程。可以看到，解析的过程是一个非常消耗资源的过程。因此，oracle在解析用户提交的SQL语句的过程中，如果对每次出现的新的SQL语句，都按照标准过程完整的从头到尾解析一遍的话，效率太低，尤其随着并发用户数量的增加、数据量的增加，数据库的整体性能将直线下降。

共享池

oracle会将用户提交来的SQL语句都缓存在内存中。每次处理新的一条SQL语句时，都会先在内存中查看是否有相同的SQL语句。如果相同则可以减少最重要的解析工作（也就是生成执行计划），从而节省了大量的资源；反之，如果没有找到相同的SQL语句，则必须重新从头到尾进行完整的解析过程。这部分存放SQL语句的内存就叫做共享池（shared pool）。当然，shared pool里不仅仅是SQL语句，还包括管理shared pool的内存结构以及执行计划、控制信息等等内存结构。

共享池：库缓存+字典缓存

shared pool的大小由初始化参数shared\_pool\_size决定。10g以后可以不用设定该参数，而只需要指定sga\_target，从而oracle将自动决定shared pool的大小尺寸。在一个很高的层次上来看，shared pool可以分为库缓存（library cache）和数据字典缓存（dictionary cache）。Library cache存放了最近执行的SQL语句、存储过程、函数、解析树以及执行计划等。而dictionary cache则存放了在执行SQL语句过程中，所参照的数据字典的信息，包括SQL语句所涉及的表名、表的列、权限信息等。dictionary cache也叫做row cache，因为这里面的信息都是以数据行的形式存放的，而不是以数据块的形式存放的。对于dictionary cache来说，oracle倾向于将它们一直缓存在shared pool里，不会将它们交换出内存，因此我们不用对它们进行过多的关注。而library cache则是shared pool里最重要的部分，也是在shared pool中进进出出最活跃的部分，需要我们仔细研究。所以，我们在说到shared pool实际上就可以认为是在指library cache。

## shared pool的组成

**shared pool 有三块区域——free （空闲空间）、library cache（缓冲sql语句和sql语句的执行计划）、row cache（数据字典信息，即dictionary cache）。**

select \* from v$sgastat a where a.NAME = 'library cache';

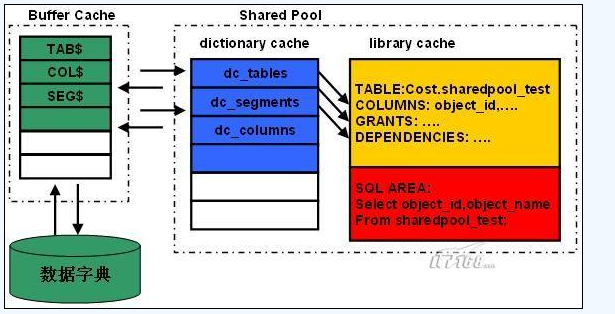
select \* from v$sgastat a where a.pool = 'shared pool' and a.NAME = 'free memory';

select \* from v$sgastat a where a.NAME = 'row cache';

简述数据字典

## shared pool的逻辑内存结构

从一个逻辑层面来看，shared pool由library cache和dictionary cache组成。shared pool中组件之间的关系可以用下图一来表示。从下面这个图中可以看到，当SQL语句（select object\_id,object\_name from sharedpool\_test）进入library cache时，oracle会到dictionary cache中去找与sharedpool\_test表有关的数据字典信息，比如表名、表的列等，以及用户权限等信息。如果发现dictionary cache中没有这些信息，则会将system表空间里的数据字典信息调入buffer cache内存，读取内存数据块里的数据字典内容，然后将这些读取出来的数据字典内容按照行的形式放入dictionary cache里，从而构造出dc\_tables之类的对象。然后，再从dictionary cache中的行数据中取出有关的列信息放入library cache中。



## shared pool的物理内存结构

从一个物理的层面来看，shared pool是由许多内存块组成，这些内存块通常称为chunk。Chunk是shared pool中内存分配的最小单位，一个chunk中的所有内存都是连续的。这些chunk可以分为六类，这六类可以从x$ksmsp（该视图中的每个行都表示shared pool里的一个chunk）的ksmchcls字段看到：

1) free：这种类型的chunk不包含有效的对象，可以不受限制的被分配。   
2) recr：意味着recreatable，这种类型的chunks里包含的对象可以在需要的时候被临时移走，并且在需要的时候重新创建。比如对于很多有关共享SQL语句的chunks就是recreatable的。  
  
3) freeabl：这种类型的chunks包含的对象都是曾经被session使用过的，并且随后会被完全或部分释放的。这种类型的chunks不能临时从内存移走，因为它们是在处理过程中间产生的，如果移走的话就无法被重建。   
  
4) perm：意味着permanent，这种类型的chunks包含永久的对象，大型的permanent类型的chunks也可能含有可用空间，这部分可用空间可以在需要的时候释放回shared pool里。

5) R-free

6) R-freea

当chunk属于free类型的时候，它既不属于library cache，也不属于dictionary cache。如果该chunk被用于存放SQL游标时，则该chunk进入library cache；同样，如果该chunk被用于存放数据字典的信息时，则该chunk进入dictionary cache。

### free 空间

free空间是由N个小的内存块用链（chain）的方式组成的。

很多链，每个链挂着可用的free内存块，每个链挂的free块大小是不一样的。

链可以将特性类似的内存块串起来，并且可以遍历（找到所有内存块），对chain的保护——latch。

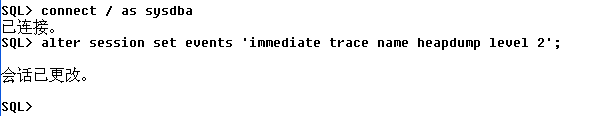
### free空间放入内容后，如何分配到chain中

sql语句 =>所有字母转换成 ASCII => 运算后得到一个数字 => 再次运算后，得到chain编号 => 就把这个sql挂到这个编号的chain上。

再次执行这个sql语句后，两次运算得到chain编号，这时，oracle会锁住这个chain，这个锁就叫latch，锁住后，会遍历这个chain，来找这个sql

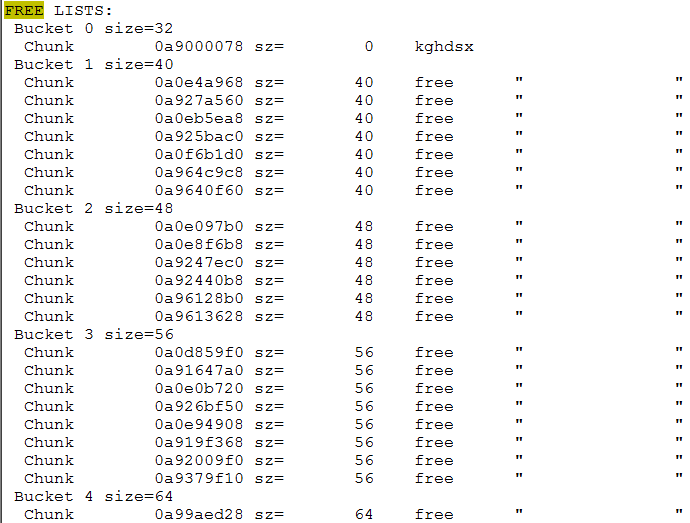
在shared pool里，可用的chunk（free类型）会被串起来成为可用链表（free lists）或者也可以叫做buckets（一个可用链表也就是一个bucket）。

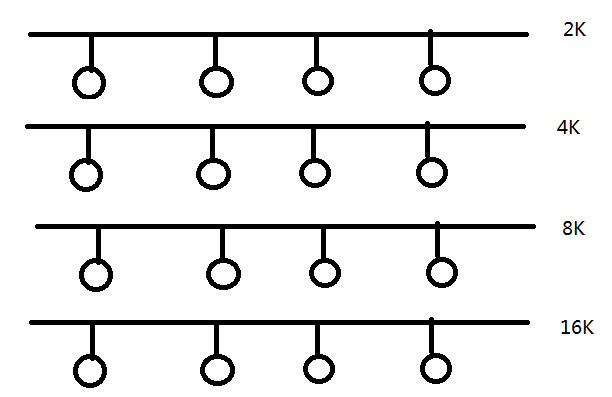
我们可以使用下面的命令将shared pool的内容转储出来看看这些bucket。   
alter session set events 'immediate trace name heapdump level 2';   
然后打开产生的转储文件，找到“FREE LISTS”部分，可以发现类似如下图所示的内容。



在11g中的diag文件夹里，vi最新产生的dump文件







每个bucket上挂的chunk的尺寸是不一样的，有一个递增的趋势。我们可以看到，每个bucket都有一个size字段，这个size就说明了该bucket上所能链接的可用chunk的大小尺寸。

如果一个12K的sql需要放入shared pool ，会找到16k的free chain，然后将一个16k中的12k分配给它，剩下的4k再放入4k的chain上继续使用。

如果硬解析找不到合适的chunk，会报4031错误。

当一个进程需要shared pool里的一个chunk时，假设当前需要21个单位的空间，则该进程首先到符合所需空间大小的bucket（这里就是bucket 2）上去扫描，以找到一个尺寸最合适的chunk，扫描持续到bucket的最末端，直到找到完全符合尺寸的chunk为止。如果找到的chunk的尺寸比需要的尺寸要大，则该chunk就会被拆分成两个chunk，一个chunk被用来存放数据，而另外一个则成为free类型的chunk，并被挂到当前该bucket上，也就是bucket 2上。然而，如果该bucket上不含有任何需要尺寸的chunk，那么就从下一个非空的bucket上（这里就是bucket 3）获得一个最小的chunk。如果在剩下的所有bucket上都找不到可用的chunk，则需要扫描已经使用的recreatable类型的chunk链表，从该链表上释放一部分的chunk出来，因为只有recreatable类型的chunk才是可以被临时移出内存的。

### 为何会产生4013错误

当某个chunk正在被使用时（可能是用户正在使用，也可能是使用了dbms\_shared\_pool包将对象钉在shared pool里），该chunk是不能被移出内存的。比如某个SQL语句正在执行，那么该SQL语句所对应的游标对象是不能被移出内存的，该SQL语句所引用的表、索引等对象所占用的chunk也是不能被移出内存的。当shared pool中无法找到足够大小的所需内存时，报ORA-4031错。当出现4031错的时候，你查询v$sgastat里可用的shared pool空间时，可能会发现name为“free memory”的可用内存还足够大，但是为何还是会报4031错呢？事实上，在oracle发出4031错之前，已经释放了不少recreatable类型的chunk了，因此会产生不少可用内存。但是这些可用chunk中，没有一个chunk是能够以连续的物理内存提供所需要的内存空间的，从而才会发出4031的错。

### latch对shared pool的保护

对bucket的扫描、管理、分配chunk等这些操作都是在shared pool latch的保护下进行的。如果shared pool含有数量巨大的非常小的free类型的chunk的话，则扫描bucket时，shared pool latch会被锁定很长的时间，这也是8i以前的shared pool latch争用的主要原因。而如果增加shared pool尺寸的话，仅仅是延缓shared pool latch的争用，而到最后，就会因为小的free chunks的数量越来越多，争用也会越来越严重。而到了9i以后，由于大大增加了可用chunk链表（也就是bucket）的数量，同时，每个bucket所管理的可用chunk的尺寸递增的幅度非常小，于是就可以有效的将可用的chunk都均匀的分布在所有的bucket上。这样的结果就是每个bucket上所挂的free类型的chunk都不多，所以在查找可用chunk而持有shared pool latch的时间也可以缩短很多。

### 处理大对象

对于非常大的对象，oracle会为它们单独从保留区域里分配空间，而不是从这个可用chunk链表中来分配空间。这部分空间的大小尺寸由初始化参数shared\_pool\_reserved\_size决定，缺省为shared\_pool\_size的5%，这块保留区域与正常的chunk的管理是完全分开的，小的chunk不会进入这块保留区域，而这块保留区域的可用chunk也不会挂在bucket上。这块保留区域的使用情况可以从视图v$shared\_pool\_reserved中看到，通常来说，该视图的request\_misses字段显示了需要从保留区域的可用链表上上获得大的chunk而不能获得的次数，该字段应该尽量为0。

## 硬解析、软解析

硬解析步骤（需从N多执行计划中挑出CBO最少的计划）、软解析步骤

shared pool内存块组成结构

ora-4031错误

select count(\*) from x$ksmsp; ---------得到shared pool里chunk的总数，每个chunk都有一行信息。

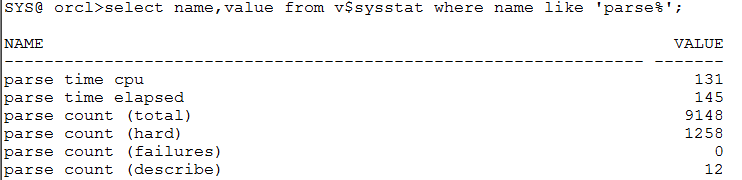
select count(\*) from dba\_objects;

select count(\*) from x$ksmsp;

**alter system flush shared\_pool;** ---将library cache和dictionary cache中所有的内容清空，**危险！**执行完毕，4031马上就不会出现了，但是**马上会出现大量硬解析。**

## v$sysstat 软硬解析的具体情况

select name,value from v$sysstat where name like 'parse%'; --判断软硬解析的数量



parse count (hard)、parse count (failures)这两个量大有问题。

parse time cpu 解析花费的cpu时间

parse time elapsed 解析花费的时间

## shared pool内存块组成结构

shared pool 有三块区域——free 、library cache（缓冲sql语句和sql语句的执行计划）、row cache（数据字典信息）。

最容易出问题的是free和library cache。我们只能设置shared pool大小，里面三块区域的大小我们不能设置。

两个概念：chain（链）、chunk（硬解析需要产生执行计划并读数据，所以需要chunk，但会产生碎片）

## SQL共享，绑定变量

SQL语句组成，动态部分、静态部分

cursor\_sharing

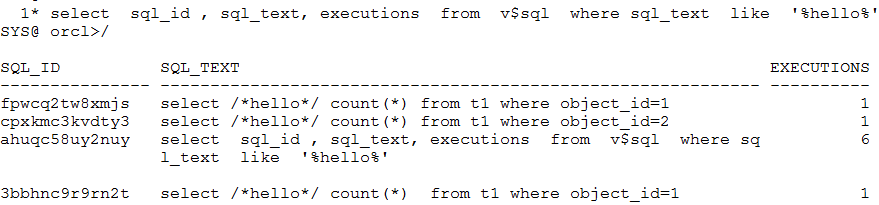
select /\*hello\*/ count(\*) from t1 where object\_id=1;

select /\*hello\*/ count(\*) from t1 where object\_id=2;

select /\*hello\*/ count(\*) from t1 where object\_id=1;

select sql\_id , sql\_text, executions from v$sql where sql\_text like '%hello%'; --查询注释

一个sql进入解析前，会分配一个sql\_id



declare

v\_sql varchar2(50);

begin

for i in 1..10000 loop

v\_sql := 'insert /\*hello\*/ into test values(:1)';

execute immediate v\_sql using i;

end loop;

commit;

end;

:1 是占位符，如果换成 :i，是直接读取的i值，导致大量硬解析

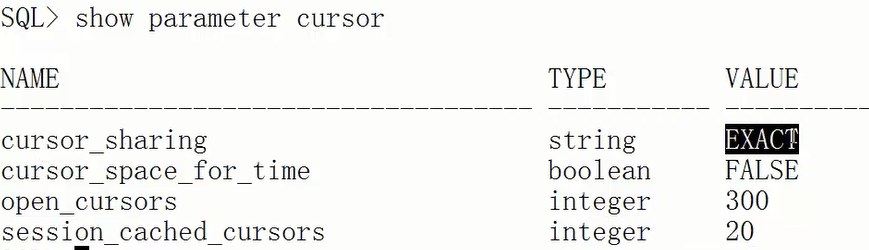
select SQL\_ID,sql\_text,EXECUTIONS from v$sql where SQL\_TEXT like '%hello%';

SQL\_ID SQL\_TEXT EXECUTIONS

------------- -------------------------------------------- --------

czhw79694nmz5 insert /\*hello\*/ into test values(:1) 10000

### cursor 修改和建议修改值



## 找出没有共享的SQL语句

如何找出不能共享cursor的sql

在v$sql查找执行次数较小的sql语句，观察这些sql语句是否是经常执行的。

select SQL\_FULLTEXT from v$sql where EXECUTIONS=1 and sql\_text like '%from t%';

select SQL\_FULLTEXT from v$sql where EXECUTIONS=1 order by sql\_text;

【实验】将上例的替换变量变成i，看是否是硬解析

## 解析命中率

select sum(pinhits)/sum(pins)\*100 from v$librarycache;

select sum(gets),sum(getmisses),100\*sum(gets-getmisses)/sum(gets) from v$rowcache where gets>0;

rowcache 一般不会出问题，生产中百分比都是99%以上；

library cache容易出问题

## 解决4031错误的方法

1、alter system flush shared\_pool; --临时性解决，但会出现大量硬解析

2、共享SQL

找出需要设置id共享sql，告知开发人员，如果开发人员也不能解决，只能改cursor\_sharing

alter system set cursor\_sharing=’force’;

没有使用共享sql原因：

1. 字面值没有使用绑定变量
2. 书写不规范情况

force解决 大量字面值没有使用绑定变量，不能解决书写不规范情况。

3、select \* from v$db\_object\_cache where sharable\_mem > 10000

and (type = 'PACKAGE' or type='PACKAGE BODY' or type = 'FUNCTION' or type='PROCEDURE')

and kept = 'NO';

执行dbms\_shared\_pool.keep('对象名');

DBMS\_SHARED\_POOL

@?/rdbms/admin/dbmspool.sql

4、保留区

select REQUEST\_MISSES from v$shared\_pool\_reserved;

5、增加shared pool空间

select COMPONENT,CURRENT\_SIZE from V$SGA\_DYNAMIC\_COMPONENTS;--------查看当前shared pool的实际大小

show parameter sga\_target

show parameter sga\_max\_size

alter system set shared\_pool\_size=150M scope=both;

## 查看执行计划

select \* from table(dbms\_xplan.display\_cursor('g4pkmrqrgxg3b'));

## 在Oracle10g中允许有多个sub shared pool,可以设置大于1G的shared pool

设置shared pool的大小

SELECT

shared\_pool\_size\_for\_estimate "SP",

estd\_lc\_size "EL",

estd\_lc\_memory\_objects "ELM",

estd\_lc\_time\_saved "ELT", estd\_lc\_time\_saved\_factor "ELTS",

estd\_lc\_memory\_object\_hits "ELMO"

FROM v$shared\_pool\_advice;

SELECT 'Shared Pool' component,shared\_pool\_size\_for\_estimate estd\_sp\_size,estd\_lc\_time\_saved\_factor parse\_time\_factor,

CASE

WHEN current\_parse\_time\_elapsed\_s + adjustment\_s < 0

THEN 0

ELSE

current\_parse\_time\_elapsed\_s + adjustment\_s

END response\_time

FROM (SELECT shared\_pool\_size\_for\_estimate,shared\_pool\_size\_factor,estd\_lc\_time\_saved\_factor,a.estd\_lc\_time\_saved,e.VALUE/100 current\_parse\_time\_elapsed\_s,c.estd\_lc\_time\_saved - a.estd\_lc\_time\_saved adjustment\_s FROM v$shared\_pool\_advice a,(SELECT \* FROM v$sysstat WHERE NAME = 'parse time elapsed') e,(SELECT estd\_lc\_time\_saved FROM v$shared\_pool\_advice WHERE shared\_pool\_size\_factor = 1) c);

# 执行计划

# Oracle登录与启动

## 认证模式

### 操作系统认证模式

操作系统认证模式，不需要用户名和密码，直接进入到最高用户（sys）：

1. dba组 (系统登录用户所属的组)
2. 连接是安全的

sqlplus /nolog

#### 【实验】 dba组才能sqlplus登陆实例

su - oracle

再 su root

sqlplus /nolog，再sqlplus / as sysdba 提示权限不足： ORA-01031: insufficient privileges

sqlnet.ora里，有下面属性，连接是安全的

SQLNET.AUTHENTICATION\_SERVICES = (NTS)

### 密码文件认证模式

密码文件认证模式——远程，或操作系统认证不具备的时候，跨系统的连接

密码文件的位置：

linux $ORACLE\_HOME/dbs

windows $ORACLE\_HOME/database

#### v$pwfile\_users

查询v$pwfile\_users视图可查看SYSOPER/SYSDBA 权限的用户信息

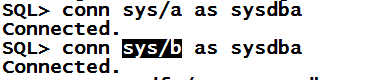
select \* from v$pwfile\_users; --查询后，应该会有记录条，记录密码文件里的相关用户信息

USERNAME SYSDB SYSOP SYSAS

------------------------------ ----- ----- -----

SYS TRUE TRUE FALSE

操作系统认证优先级高于密码文件认证 **验证：**conn sys/a as sysoper --用sysoper验证



#### 验证密码文件是否可用

select \* from v$pwfile\_users; --查询密码文件是否有效

USERNAME SYSDB SYSOP SYSAS

------------------------------ ----- ----- -----

SYS TRUE TRUE FALSE

su - oracle

su root

sqlplus /nolog

@ >conn sys/aa as sysoper

ERROR:

ORA-01031: insufficient privileges

@ >conn sys/oracle as sysoper

Connected.

## pfile 和spfile区别

|  |  |
| --- | --- |
| 初始化参数文件 | |
| 文本参数文件pfile | 二进制参数文件SPFILE |
|  | 1. 动态修改 2. 类型可以不是文件 3. spfile 可以用RMAN自动备份，pfile不能 |

#### 【实验】删除pfile和spfile，如何恢复

spfile和pfile删除后，在bdump的alert.log文件中，查找开库的语句，复制下来，然后在dbs下，vi一个pfile，粘贴进去。保存，起库即可。

11g直接粘贴即可

10g需要修改两行，将其后面加上双引号，如下图：



## 启动数据库三阶段

### nomount

读取密码文件、初始化文件

初始化数据字典

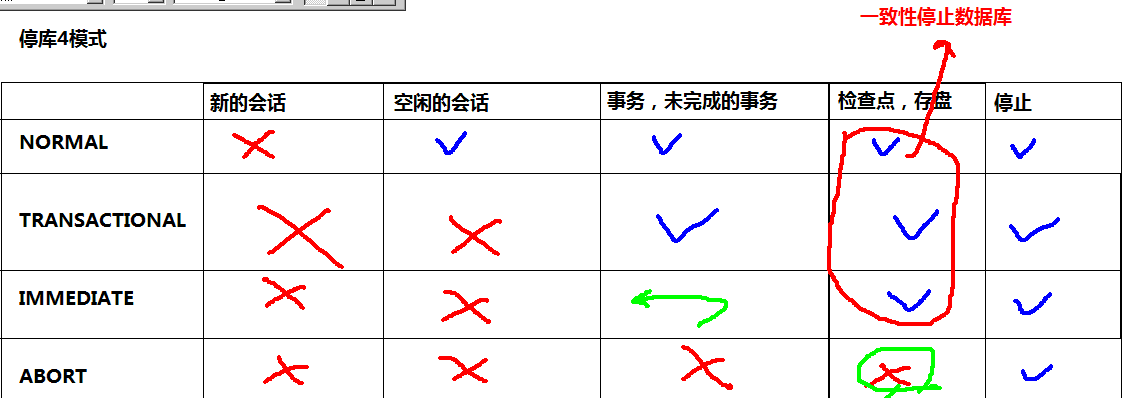
加载后台进程

### mount

### open

open库 --检查数据库一致性（库、数据文件等），如不一致，用日志恢复。

## 停库4模式



abort相当于kill进程

## 控制文件

1. 描述了数据库的结构和行为
2. 在参数文件的control\_files中描述

select name from v$controlfile; --当前实例正在使用的控制文件

select name, VALUE from v$spparameter where name ='control\_files'; --查看spfile中的参数

1. 控制文件最少1个，最多8个
2. 控制文件预先分配空间，可以自动增长，也可以建立新的、更大的控制文件

控制文件预分配空间后，一段时间后，此文件大小不会变化

1. 所有控制文件都是镜像关系

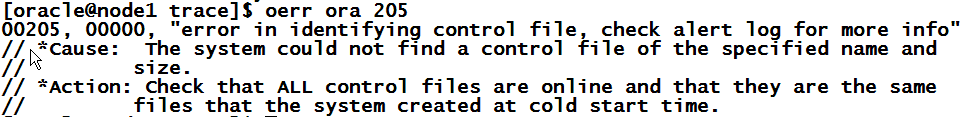
总共有三个控制文件，这三个文件是如何同步的？

第一个写完毕后，镜像到第二个，叫一个控制文件的交易

1. 在mount时，读取

#### oerr 命令

看告警日志 下图是看ora-000205信息



#### 【实验】

将已有2个控制文件增到3个

1. 关库
2. startup nomount
3. 修改参数
4. 起库，会有何变化？

### 创建控制文件

alter database backup controlfile to trace as '/home/oracle/control.txt'; --提示如果已有文件，用reuse

alter database backup controlfile to trace as '/home/oracle/control.txt' reuse; --reuse 覆盖

alter database backup controlfile to trace; --备份控制文件到udump文件中

--创建控制文件

CREATE CONTROLFILE REUSE DATABASE "ORCL" NORESETLOGS NOARCHIVELOG

MAXLOGFILES 16

MAXLOGMEMBERS 3

MAXDATAFILES 100

MAXINSTANCES 8

MAXLOGHISTORY 292

LOGFILE

GROUP 1 '/oracle/oradata/orcl/redo01.log' SIZE 50M BLOCKSIZE 512,

GROUP 2 '/oracle/oradata/orcl/redo02.log' SIZE 50M BLOCKSIZE 512,

GROUP 3 '/oracle/oradata/orcl/redo03.log' SIZE 50M BLOCKSIZE 512

-- STANDBY LOGFILE

DATAFILE

'/oracle/oradata/orcl/system01.dbf',

'/oracle/oradata/orcl/undotbs01.dbf',

'/oracle/oradata/orcl/sysaux01.dbf',

'/oracle/oradata/orcl/users01.dbf',

'/oracle/oradata/orcl/example01.dbf',

'/oracle/oradata/orcl/system02.dbf',

'/oracle/oradata/orcl/example02.dbf',

'/oracle/oradata/orcl/undotbs02.dbf',

'/oracle/oradata/orcl/sysaux02.dbf'

CHARACTER SET ZHS16GBK

;

--加临时表空间（创建没有）

ALTER TABLESPACE TEMP ADD TEMPFILE '/oracle/oradata/orcl/temp01.dbf'

SIZE 20971520 REUSE AUTOEXTEND ON NEXT 655360 MAXSIZE 32767M;

#### 【实验】控制文件重建后，可以观察到，文件大小变化了

如何判断是备份到哪个文件？

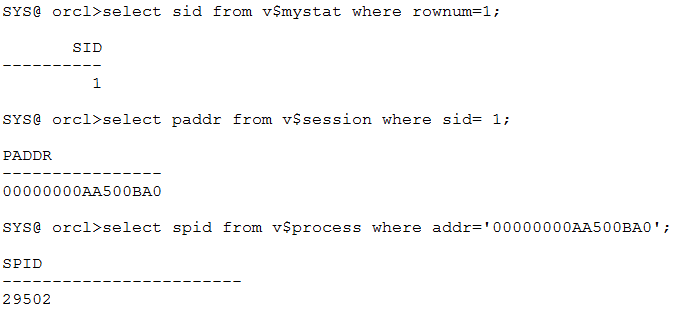
1. 查看udump文件夹，ls -trl 查找最新的dump文件（或去bdump下看alert.log，有记录trace到哪个文件）
2. 查看当前会话进程ID、地址

--v$mystat 官档解释：This view contains statistics on the current session.

select sid from v$mystat where rownum=1; --查当前会话的sid

select paddr from v$session where sid= 1;

select spid from v$process where addr= '00000000AA500BA0';



ps -ef 找到29502，是当前会话



### 查看当前会话进程ID、地址(上例)

## 日志文件

生产中，日志文件最少500m，最大500G都有。20分钟切一次，三组切完一个小时最好。

1. 记录数据库的变化，顺序循环写
2. 按组的方式（逻辑），可查v$log
3. 组内文件为成员，成员是物理文件；同组内成员是镜像关系
4. 一个日志组写完，切换到另一个组，也可手工切换：alter system switch logfile;
5. 手工存盘 alter system checkpoint;
6. 日志切换，触发检查
7. SCN= system change number：数据库做任何操作，都会有一个唯一的SCN号。SCN号是oracle自动产生的唯一、顺序、连续的号。
8. 需要归档的SCN号（SCN号上限是12个16进制的f，等于10进制的2.8147E+14）

select to\_number('ffffffffffff','xxxxxxxxxxxxxxxxxxxxx') from dual;

TO\_NUMBER('FFFFFFFFFFFF','XXXXXXXXXXXXXXXXXXXXX')

-------------------------------------------------

2.8147E+14

select current\_scn, CHECKPOINT\_CHANGE#, current\_scn - CHECKPOINT\_CHANGE# need\_CKPT from v$database;

CURRENT\_SCN CHECKPOINT\_CHANGE# NEED\_CKPT

----------- ----------------- ----------

2511401 2506921 4480

select GROUP# , THREAD#, SEQUENCE#, FIRST\_CHANGE# from v$log;

GROUP# THREAD# SEQUENCE# FIRST\_CHANGE#

---------- ---------- ---------- -------------

1 1 118 2506920

2 1 116 2468899

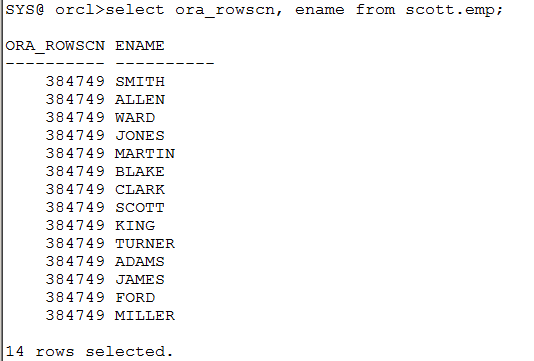
3 1 117 2503224

分布式数据库，能爆SCN号

修改某行的数据，此行的ora\_rowscn会变。

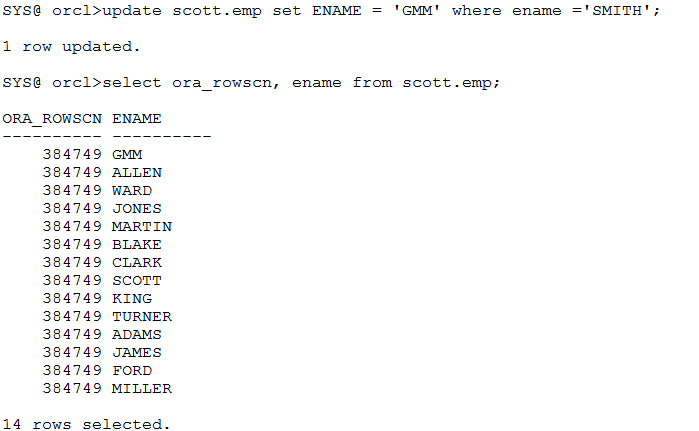
select ora\_rowscn, ename from scott.emp;

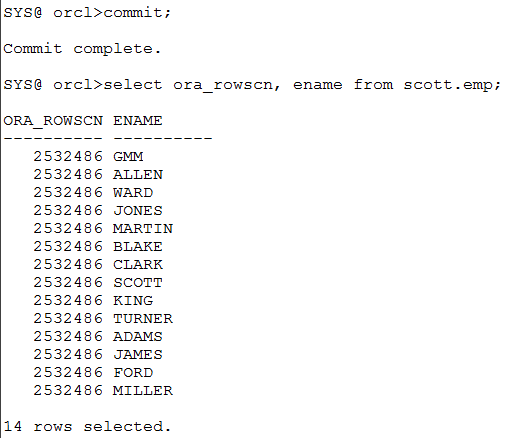




update emp set \*\*\*;

commit --只要一提交，表中所有的ora\_rowscn都会变。





为什么一提交，表中所有的ora\_rowscn都会变？

因为oracle只记到这个数据块级别的SCN。

### 检查点

切换日志 alter system switch logfile;

手工存盘 alter system checkpoint;

#### 完全检查点

1. 一致性停止数据库
2. alter system checkpoint;

#### 增量检查点

alter system switch logfile; 写部分数据

自动产生检查点，由参数log\_checkpoint\_timeout控制，生产中默认即可。

### 维护日志文件

##### 查看组和成员

select group#, archived, status from v$log;

select \* from v$logfile;

##### 增加组

alter database add logfile group 8 '/oracle/oradata/orcl/redo8a.hhh' size 50m;

##### 增加成员

alter database add logfile member '/oracle/oradata/orcl/redo8b.hhh' to group 8;

##### 删除成员

alter database drop logfile member '/oracle/oradata/orcl/redo8b.hhh';

##### 成员改名

shutdown immediate;

startup mount

--操作系统下，mv 需要修改的redo日志

mv /oracle/oradata/orcl/redo8a.hhh /oracle/oradata/orcl/redo08a.log

mv /oracle/oradata/orcl/redo8b.hhh /oracle/oradata/orcl/redo08b.log

--oracle改名

alter database rename file

'/oracle/oradata/orcl/redo8a.hhh', '/oracle/oradata/orcl/redo8b.hhh'

to '/oracle/oradata/orcl/redo08a.log', '/oracle/oradata/orcl/redo08b.log';

alter database open;

select \* from v$log;

select \* from v$logfile;

##### 清除组

先查看日志组是否在用，再清除：

select group#, archived, status from v$log;

alter database clear logfile group 8;

注意：清除组8完毕后，如果删除8组的redo文件，再重新清除，这个删除的redo文件还会恢复。

证明一个问题：清除组，就是删除此组，再重新建一个空的组和组成员。

##### 删除组

非当前、非活动的组能删除；

只有两个日志组的情况下，不能删除；

select group#, archived, status from v$log;

alter database drop logfile group 8;

##### 【实验】日志文件组和成员的增、删、改相关操作

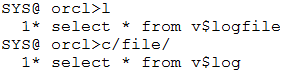
1. 增加组，增加2个组成员，此时成员数为3个。
2. 删除一个组成员，修改剩下的2个组成员数据文件名
3. 清除组

rm 一个非当前非活动的成员文件，然后清除此成员所在的组，再看此数据文件是否存在。

1. 删除组

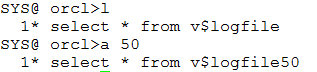
### help index

--查看替换等所有的帮助信息



c /oldfile/newfile --将sql中的oldfile改成newfile

/ --执行新的语句



a file --将sql最末位后加上file，不包括分号

### 修改表空间的位置

1. offline表空间
2. cp数据文件到新的位置
3. rename表空间数据文件为新位置

alter tablespace TBNAME rename datafile to '/新位置/\*\*\*.dbf' ;

1. online

### 【问题】如何看等待事件