# Oracle 11g R2 DBA

shutdown immediate关闭数据库

startup启动数据库

删除从多余的监听器

G:\oracleDB\product\11.2.0\dbhome\_1\BIN/netca.bat

删除注册表regedit

HKEY\_LOCAL\_MACHINE/SYSTEM/ControlSet001/Services 删除相应的Oracle开头如OracleOraHomeTNS...的文件夹。  
HKEY\_LOCAL\_MACHINE/SYSTEM/ControlSet003/Services 删除相应的Oracle开头如OracleOraHomeTNS...的文件夹。  
  
HKEY\_LOCAL\_MACHINE/SYSTEM/CurrentControlSet/Services 删除相应的Oracle开头如OracleOraHomeTNS...的文件夹。

## 数据库连接方式

connect as sysdba ：用户名，密码可以随便写，连接oracle数据库

$sqlplus sys/oracle as sysdba

$sqlplus / as sys(用户名)dba(角色)

Conn / as sysdba

select name,created,log mode,open mode from v$database;查看数据库的当前信息，数据库名，创建时间，非归档模式，打开模式为读写，

## 数据库的启动

数据库启动

启动实例

打开控制文件

打开数据库文件，日志文件。。。。

startup：启动数据库(超管才能启动)

数据库启动涉及到3个状态(3个文件)

nomount状态：打开数据库实例，读取参数文件

mount状态：ORACLE根据参数文件中控制文件的位置找到并打开控制文件

open状态：打开数据文件，日志文件，数据恢复

### 数据启动到nomount状态

一个参数文件就可以启动到nomout状态

$sqlplus /nolog

connect / as sysdba --连接到空闲例程,这是一种操作系统认证方式

startup

数据库的启动过程记录在告警追踪文件中，该追踪文件包括数据库启动的信息，存放在BACKGROUND\_DUMP\_DEST目录下，alert\_orcl.log

show parameter background\_dump\_dest;--查询告警日志的存储目录

select \* from v$controlfile;--测试启动到nomount状态时控制文件是否打开，没打开，因为此刻状态仅仅加载参数文件，没有打开控制文件

show parameter control\_files;--获得控制文件的位置

ALTER SYSTEM SET CONTROL\_FILES=' G:\oracleDB\oradata\orcl\ CONTROL01.CTL'   
SCOPE=SPFILE

### 数据库启动到mount状态

2种方式启动：

1：直接启动数据库到mount状态

2：启动到nomount状态，使用指令到mount状态

Alter database mount;-- 切换到mount状态

此时可以通过v$controlfile动态数据字典视图，获得控制文件的存储目录,因为此状态打开了控制文件，仅仅读取了数据库文件的和日志文件的位置，控制文件各种参数配置，但是数据库文件不能读取

### 数据库启动到open状态

启动方式：

1：startup open/startup（默认open）--直接启动到open状态

2:alter database open;--数据库处于nomount/mount状态可使用此命令

此时可以开启查询数据库表数据

## 数据库的关闭

步骤：关闭各种数据库文件(CLOSE)，关闭打开的控制文件(DISMOUNT)，关闭实例(SHUTDOWN)

Alter database close;--关闭数据库,可以通过告警文件查看

Alter database dismount;---dismount状态

Shutdown;--关闭数据库实例，会进行一些额外的操作，譬如断开连接，回滚数据….

数据库关闭的几个参数及其含义

Shutdown normal:默认方式，关闭数据库时，不允许新的数据库连接，只有当前所有的连接都退出时才会关闭数据库，

优点：安全关闭数据库

缺点：当有大量的连接用户时，关闭数据库时间较长

Shutdown immeidate：较快且安全的关闭数据库(DBA常用)，会做一些操作，中断当前事务，回滚未提交的事务，强制断开所有用户连接，执行检查点，吧脏数据写到数据文件中

Shutdown transactional:数据库当前连接继续执行，但不允许新的连接，一旦当前所有事务执行完毕，则关闭数据库。

Shutdown abort:不安全的关闭数据的方式，会造成数据丢失。断开当前得所有用户的连接，拒绝新的连接，断开当前的所有执行事务，立即关闭数据库。当使用这种方式关闭时，数据重启时需要进行数据库恢复。

## Oracle数据库体系结构

### Oracle实例

内存区和后台进程组成

访问数据库必须启动实例，

启动实例时-先分配内存区-启动后台进程-后台进程执行库数据输入，输出以及监控其他oracle进程

数据库启动进程中有5个进程必须启动。监控进程(SMON)，进程监控(PMON)，数据库编写进程(DBWR)，日志写进程(LGWR)，检查点进程(CKPT)，否则实例无法创建，数据库启动过程可以在告警日志查看。

### Oracle服务器

由据库实例和数据库文件组成，也就是数据库管理系统(DBMS)

数据库服务作用：维护实例，数据库文件，在用户建立与服务器的连接时启动服务器进程并分配pga

### 数据库文件

数据文件(data files):数据库信息最终存储位置

控制文件(control files):包含维护数据库和验证数据库完整性的信息，2进制文件

重做日志文件(redo log files)：包含数据库发生变化的记录，发生故障时用于数据恢复

### 参数文件，密码文件，归档日志文件

参数文件（paramter file）：定义数据库实例的特性，包含sga中内存结构分配空间的参数，如分配数据库告诉缓冲区的大小…

密码文件(password file):密码文件授予用户启动和关闭数据库实例，在刚安装数据库时，oracle默认用户名和密码就在密码文件中，oracle可以依此判断用户操做权限

归档日志文件(archive log files)：归档日志文件是日志文件的脱机备份，发生故障后进行数据恢复可能使用该文件

### 数据库连接与会话

#### 数据库连接

连接值用户进程与数据库服务器之间的通信途径，一个连接可以有多个会话

Oracle3种连接方式

1：基于主机的方式(Host-Based)：此方式种，服务器和客户端运行在同一台计算机上，用户可以直接连接数据库服务器

2：基于客户机-服务器的方式(Client-Server):该方式中数据库服务器和客户端运行在不同的计算机上，客户通过网络连接数据库服务器，

3：客户-应用服务器-数据库服务器方式(client-application server-server)：这种方式称为3层访问方式，用户首先访问应用服务器，然后由应用服务器连接数据库服务器，应用服务器如中介，完成客户和数据库的交互。

#### 会话

值一个明确的数据库连接，一旦采用上述3种其中之一连接数据库服务器方式，就把这样的连接称之为会话。

专有连接：用户通过工具在专用连接的情况下访问数据库，账户密码服务器校验通过后，服务器就会自动创建一个与该用户进程相对应的服务器进程。2者 1对1关系。

共享连接：在共享服务器配置的情况下，多个用户进程可以同时共享服务器进程。多对1关系

select serial#,username,status,server,process,program,logon\_time from v$session;

### oracle数据库内存结构

内存结构由2部分组成

SGA(系统全局区)：数据库实例的一部分，当数据库实例启动时，会首先分配系统全局区，在系统全局区中包含几个重要的内存区，即数据库高速缓存(Database buffer cache)，重做日志缓存(Redo log buffer cache)，共享池(Shared pool)，大池(Large Pool)和java池(java pool)

PGA(程序全局区):程序全局区不是实例一部分，当服务器进程启动时，才分配PGA

#### SGA

##### 共享池(Shared pool)

作用：共享sql/plsql代码，即把解析的sql代码的结果在这里缓存。Plsql不仅在这里缓存还在这里共享

由2部分组成，库高速缓存(library cache)和数据字典高速缓存(data dict cache)。

##### 库高速缓存

存储最近使用的sql/plsql语句，

容量有限，不能直接设置容量，但是可以通过设置共享池的大小间接的修改。而共享池是sga的一部分，所以共享池不能超过sga。

内部采用lru(least recently used)算法管理库高速缓存，基本思想，把一段时间内没有被使用的语句清除，一旦缓冲区填满，把最近最少使用的执行计划和解析树从库高速缓存中清除。

Alter system set shared\_pool\_size=16M;--设置共享池大小

show parameter shared\_pool\_size;--查看共享池大小

##### 数据字段高速缓存

称为：字典缓存，行缓存

与数据字典相关的一段缓冲区，存储了数据文件，表，索引，列，用户，权限信息，和其他一些数据库对象的定义。

大小取决于共享池的大小。

##### 数据库高速缓冲区(DATA BUFFER CACHE)

存储了最近从数据文件读入的数据块信息，或用户更改后需要写回数据库的数据信息。

此时没有提交给数据库的更改后的数据称为脏数据

当用户执行select时，会先在这里读取。

DB\_BLOCK\_SIZE：设置oracle数据块的大小

DB\_BLOCK)BUFFERS：数据库的个数

2者乘积就是库高速缓存的大小。

Show parameter db\_block\_size;--查询数据库块的大小

Show parameter db\_cache\_size;--查询数据库高速缓存的大小

因为在oracle11g中，sga为数据库服务器自动管理，所以该db\_cache\_size为0，当然在运行时oracle数据库时，数据库高速缓存内存已分配好。

Show sag;--查看数据库高速缓存的大小

Alter system set db\_cache\_size=200m;--动态设置数据库高速缓冲区的大小

Buffer cache advisory parameter：让oracle对于数据库缓冲区的内存分配提供一些建议，用于启动或关闭统计信息，这些信息用于预测不同缓冲区大小导致的不同行为特性。Dba可以参考这些统计信息，基于当前的数据库工作负载设置优化的数据库高速缓存。

顾问通过初始化参数DB\_CACHE\_ADVICE 启动或关闭。

该参数有3状态

Off：关闭缓存顾问，不分配缓存顾问的工作内存

On：打开缓存顾问，分配工作内存

Ready：打开缓存顾问，不分配缓存顾问内存

Show parameter db\_cache\_advice;--查看数据库高速缓存顾问状态，默认开启

Alter system set db\_cache\_advice=off;--关闭数据库高速缓存顾问

设置开启后。可以通过动态性能视图，v$db\_cache\_advice 查看缓冲区的建议信息

Col id for 99;

Select id ,name,block\_size,size\_for\_estimate,buffers\_for\_estimate 2 from v$db\_cache\_advice;

##### 重做日志高速缓冲区(redo buffer cache)

当用户执行insert，update，delete，create，alter或drop操作后，数据发生了变化，这些变化的数据在写入数据库高速缓存之前会先写入重做日志缓冲区，同时变化之前的数据也放入重做日志高速缓存。方便oracle最后执行回滚。

Show parameter log\_buffer;--查看重做日志缓存区

##### 大池(Large pool)和java池(java pool)

###### 大池

是sga一段可选内存区,只在共享服务器环境配置

作用，用户进程区域uga(pga在共享服务器中的另一种称呼)的大部分都在大池分配

减轻共享池的负担，在大规模输入，输出及备份过程中也需要大池作为缓存空间

Show parameter large\_pool size;--查看大池大小,参数为0说明自动管理

alter system set large\_pool\_size=48M;--修改大池大小

###### java池

一段可选的内存区域

安装完java或者设置使用java程序时必须设置java池，用于编译java语言编写的指令

Java与plsql在数据库中有相同的存储方式，

Show parameter java\_pool\_size;--查看java池的大小,参数为0说明自动管理

alter system set java\_pool\_size=48M;修改参数值

##### 流池

也称流内存，oracle流专用的内存流，流是oracle数据库中的一个数据共享，

大小可通过stream\_pool\_size设置

#### PGA(进程全局区)和UGA（用户全局区）

PGA是服务器进程专用的一块内存，他是操作系统进程专用的内存，系统的其他进程无法访问。Pag独立sga，不会再sga出现，由操作系统在本地分配

PGA内存管理

Show parameter sort\_area\_size;--查询PGA中排序区的大小

在服务器进程最初查询时，会使用512KB内存实现数据排序。Oracle排序处理完后，数据排序区的大小就由sort\_AREA\_SIZE决定

##### PGA

作用：存储服务器进程或单独的后台进程的数据信息和控制信息。

生命周期：随服务器进程的创建而被分配内存。进程结束他结束。

在转有服务器中，pag存储

1：排序区，对某些sql语句执行结果进行排序

2：会话信息：包含本次会话的用户权限和性能统计信息

3：游标状态，包含其他的会话变量

共享服务器配置中，多个用户进程共享一个服务器进程，如果创建了大池，内存结构存在大池中，否则存在共享池中。

在共享服务器结构中，会话信息是存储在SGA中的，2种模式下堆栈存在pga中

##### UGA

存储用户的会话状态，每个共享服务器都可以访问它，在PGA中分配

手动管理PGA：必须告诉oracle一个特定的进程需要的排序区，运行使用多少内存。

自动管理PGA:要求告诉oracle在系统范围内可以为PGA中的特定功能(如：排序区分配多少内存)

#### 如何获得内存缓冲区的信息

Show sga;--查看sga中内存分配情况

固定SGA(和fixed size相关):存储一组指向sga中其他组件的变量，他的大小用户无法控制，oracle使用这个内存区来寻找其他sga区。可理解为数据库的自举区。

和variable size相关的内存区，该部分内存区包括共享池，java池和大池，其中variable size大小高于sga其他内存结构之和。因为total sga中除去db\_cache\_size部分也包括在variable size中

Show parameter sga\_max\_size;--查看sga大小

### 服务器进程和用户进程

服务器进程:犹如中介，完成用户的各种数据库服务请求，而把数据库服务器返回的数据和结果发给客户端，在专有连接中，一个服务器进程对应一个用户进程，2者1对1关系。当用户连接中断，服务器退出。在共享连接中，一个服务器进程对应多个用户进程，此时服务器通过opi(oracle program interface)与数据库服务器通信。

用户进程：当用户使用连接数据库时，启动一个进程。

如：conn scott/tiger@orcl

### Oracle数据库后台进程

后台进程是在实例启动时，在数据库服务端启动的管理程序，它使数据库的内存结构和数据库物理结构之间协调工作，

Sga

后台进程

Oracle数据库文件

数据库后台进程5个必须启动，否则数据库实例无法启动成功。Dbwr/lgwr/pmon/smon/ckpt

#### 系统监控进程 (Smon)

1：数据库实例恢复。

除此之外，系统监控进程执行某些空间维护的作用

2：Combine，coalesces，adjacent数据文件中的自由空间

3：回收数据文件中的临时段

当数据库发生故障，数据库重新启动后，系统监控进程自动恢复实例，实例恢复3步骤

1：前滚所有没有写入数据文件而记录在重做日志文件中的数据，此时，系统监控进程读取重做日志文件，把用户更改的数据重新写入数据块。

2：打开数据库，此时或许系统监控进程的前滚操作还没有完成，oracle这样做的目的就是方便用户及时登录，以免前滚时间太长，影响用户的行为，这样用户就可以操作那些没有被事务恢复锁住的数据

3：回滚未提交的事务

#### 进程监控进程(PMON)

负责服务器进程的管理和维护工作，在进程失败或连接异常发生时该进程负责一些清理工作

1：回滚没有提交的事务

2：释放所持有的当前的表或行锁

3：释放进程占用的sga资源

4：监视其他oracle的后台进程，有必要时重启这些后台进程

5：向oracleTNS监听器注册刚启动的实例，如果监听器在运行，就与这个监听器通信并传递如服务器名和实例的负载等参数，如果监听器没有启动，进程监控会定期地尝试连接监听器来注册实例

#### 数据库写进程(DBWR)

脏数据就是用户更改了但没有提交的数据库中的书库。导致数据库的数据文件和数据库高速缓存中的数据不一致，称为脏数据。

这种脏数据在特定的条件下写到数据文件中，这就是数据库写进程的作用

这样做的好处是减少i/o次数，不需要频繁提交。

如下事件发生会触发数据库写进程吧脏数据写入数据库的数据文件中

1:发生检查点事件

2:脏数据达到门限值

3：数据库缓冲区没有足够的缓存为其他事务提供足够的空间

4：表空间处于，热备份状态/离线状态/只读状态

5：删除表/截断表

6：超时

在数据中，一个数据库实例可以启动多个数据库写进程，来分担单个写进程的工作负载

#### 重做日志写进程(LGWR)

将重做日志缓冲区中的数据写入重做日志文件，此时重做日志缓冲区中的内容是恢复事务所需要的信息。回滚。

满足以下条件，会触发

1：事务提交

2:重做日志缓冲区1/3被占用

3：重做日志缓冲区中有1mb数据

4：数据库写进程把脏数据写到数据文件之前

日志写进程会通知数据库写进程将脏数据写到数据文件，但是数据库写进程不会把脏数据写到在线重做日志，也不会通知日志写进程做任何事情

数据写进程跟重做日志写进程的对比

数据库写进程是离散写到不同数据库文件上的，在指向一个更新时，数据库写进程会修改不同空间中存储的数据块和索引块，所以数据库写进程的离散写的速度很慢。

重做日志写进程是顺序写，把每个事务的重做信息全部放在重做日志中，通过在数据库高速缓存中缓存脏数据块，而重做日志写进程完成大规模顺序写，整体上提高系统性能

#### 归档日志进程(ARCH)

可选进程，并不在实例启动时自动启动，

作用:把写满的重做日志文件的数据写到一个归档日志中，用作介质故障时的数据库恢复。磁盘损坏恢复。.

Archive log list;--查看系统的归档模式，默认关闭

设置数据库为归档模式的过程

1：关闭数据库，

2：以mount参数启动数据库，更改后在启动到open状态

Shutdown immediate;

Connect /as sysdba;

Startup mount;

Alter database archivelog;--重点是这里

Alter database open;

数据库日志模式改为存档模式。同时启动自动存档。归档日志的存储目录即存档终点是数据库恢复文件目录

USE\_DB\_RECOVERY\_FILE\_DEST：归档文件存在在此参数指定的目录下

查看数据库恢复目录的位置

Show parameter db\_recovery;

#### 校验点

校验点就是一个事件，当数据库写进程把sga中所有被修改的数据库高速缓冲中的数据写到数据文件上时产生，这些被修改的数据包括提交和未提交的数据，由于引入了校验点，使得所有的校验点的所有变化了的数据都写到数据文件中，在实例恢复时，就不必恢复校验点之前的重做日志中的数据，加快系统恢复的效率

校验点进程并不用于建立校验点，而是在校验点发生时，会触发这个进程一系列工作

执行流程

1：校验点进程把校验点号码写入相关的数据文件的文件头中

2：校验点进程吧校验点号码,SCN号，重做日志序列号，归档日志名字等都写入控制文件中

Alter system checkpoint;--强制执行校验点

## SQL语言概述

数据查询语句:SELECT

数据操纵语句(DML)：delete/insert/update

数据定义语句(DDL)：create/alter/drop/rename/truncate

事务控制语句:commit/rollback

数据控制语句:grant(授权)/revoke(回收权)

## 数据字典(基于基表)

它在数据库创建时由数据库服务器创建与修改。包含如下信息

1：所有的模式(用户)对象的定义，这些对象包括表，视图，索引，族，同义词，序列号，存储过程，函数触发器等

2：数据库的逻辑结构和物理结构，如数据文件和重做日志文件的信息等。

3：所有模式对象被分配多少存储空间，以及当前使用的空间

4：默认列的值

5：对象完整性约束信息

6：用户信息

7：用户或角色的特权信息

8：审计信息，如那个用户具有访问或者修改某些模式对象的权利

使用场景

1：oracle数据库服务器用来寻找用户信息，模式对象信息(如表，索引，触发器..)和存储结构

2：当使用ddl语句时，会触发oracle服务器修改数据字典

3：普通用户或者dba用户使用数据字典获得关于数据库的信息，如数据文件的存储位置，数据库实例名，参数文件中的参数值，控制文件的信息等

### 数据字典视图分类

分成3类，都是静态视图(静态表示这些视图在数据库运行期间不会发生变化，除非执行analyze指令)，这3类数据字典视图以不同前缀区分。

DBA\_\*\*\*：包含数据库中整个对象的信息，只能由管理员查询，不要在这些视图上创建同义词

ALL\_\*\*\*：这些视图包含某个用户所能看到的全部数据库信息，包括当前用户所拥有的模式对象和用户可以访问的其他公共对象，还有通过授权或授权角色可以访问的模式对象

USER\_\*\*\*：包含当前用户访问的数据库对象信息，反映数据库中某个用户的全部信息，隐含owner信息，全部内容为以ALL为前缀的视图的子集

\*\*\*:表示数据库模式对象，如table，索引，视图等

DESC dba\_objects;:查看dba\_objects视图结构

--查看dba\_objects中scott用户的数据库对象信息

Col owner for a20

Col object\_name for a40

Select owner,object\_name,created From dba\_objects Where owner=’scott’;

Desc all\_objects;--查看all\_objects结构信息

Desc user\_objects;--查看用户对象表，没有owner列。

Select \* from user\_objects;

### 使用数据字典视图(静态视图)

User\_tables：查看当前用户所有拥有的表

User\_indexs：查看当前用户创建的索引

User\_views:查看当前用户拥有的视图

User\_catalog：查看当前用户所有表的名字和类型

Dba\_users:查看数据库系统上何时创建了多少个用户

### 动态性能视图及使用

只存在于运行的数据库中，一组虚表，也称为动态性能表

只有管理员可以查看，管理员可以创建，其他成员要查看，需要管理员授权

固定以v$为前缀

Select \* from v$fixed\_table where name like ‘V$LOG%’;--查询日志文件相关的信息

*Select group#,members,archived,status from v$log;* --查看当前正在使用的重做日志组，current说明日志组使用中，inactive说明数据库系统没有使用该重做日志组

Select \* from v$logfile;--查看当前数据库系统的重做日志组的日志成员的存储目录，文件名和状态。

查询正在使用的重做日志文件的信息

Select l.group#,l.archived,l.status,lf.type,lf.member from v$log l,v$logfile lf where

l.group#=lf.group# and l.status=’current’;

select instance\_name,host\_name,version,startup\_time,logins from v$instance;--查看实例信息(实例名，主机名，版本号，实例启动时间)

select name,created,log\_mode from v$database;查看当前数据库的信息

v$controlfile:包含控制文件存储目录和文件名信息

v$datafile:包含数据库文件信息

## 网络配置管理

### Oralce的网络连接

Oracle提供了oracle net service组件：方便地配置和管理网络连接，由以下组成

1:oracle net

2:oracle net listeners

3:oracle connection listeners

4: oracle configuration assistant

5: oracle net manager

安装数据库服务器或者客户端时自动安装好了

其中oracle net补习服务器和客户端都安装上，他负责完成客户机与服务器之间连接的初始化，建立以及维护工作,

由2部分组成

1:oracle network foundatioin leayr:建立和维护客户端与数据库服务器之间的连接和信息交互

2:Oracle protocol support ：映射transparent network substrate(TNS)到业内标准协议

Oracle网络连接流程描述

1:客户端发起连接，确定服务器，监听端口，协议，数据库服务器名

2：客户端一旦与监听器建立连接，会在客户端生成用户进程，同时监听器会判断客户端所请求的服务名是否是自己所管理的服务名

如果客户端传过来的连接字符串不包含服务名，报错，

如果请求的服务名字不是自己管理的，报错并中断，

如果请求的服务名是自己管理的，监听器就在数据库服务器上创建服务器进程。

3：监听器在创建服务器进程后，会将用户进程与服务器进程建立连接，之后监听器退出与客户端的连接。

4:服务器进程根据用户进程提提供的用户名和密码到数据字典里判断是否正确

5:不匹配报错，匹配则分配pga，并生成session

### 服务器监听器配置

无论是专有服务器连接还是共享服务器连接，数据库服务器必须启动监听程序

监听程序(oracle net service服务)：由oralce文件管理，文件名为listener.ora，

在linux默认位于$oracle\_home/network/admin目录下，只运行在数据库服务器上，完成监听客户连接请求的作用。Oralce使用lsnrctl实用程序完成监听程序的配置和管理

Listener.ora目录: G:\oracleDB\product\11.2.0\dbhome\_1\NETWORK\ADMIN\ listener.ora

不指定监听器，默认listen监听

1：新建监听

LISTENER1 =

(DESCRIPTION\_LIST =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1522))

)

)

2:启动监听$lsnrctl start listener1;

$ lsnrctl status listener1;--查看监听状态

没有任何服务注册在该服务器上，客户端无法连接到该服务器，因为涉及到服务器名的问题，只有客户端将《服务器名》传给监听器之后，监听器服务经过处理才能建立客户端到数据库服务器的连接。

#### 动态注册

修改完毕，立即生效，无需重启。

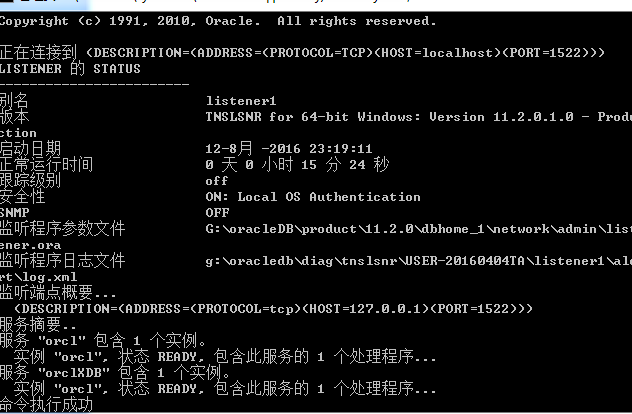
PMON进程将当前实例的服务名注册到同一台服务器上，默认在1521端口上监听的监听器就是名为listener的监听器

Show parameter service\_name;--查看当前数据库的服务名

Show parameter local\_listener;--local\_listener控制动态注册到的监听器也就是将数据库动态注册到这个监听器上

修改参数local\_listener动态注册orcl数据库(上述查询的当前数据库服务名)

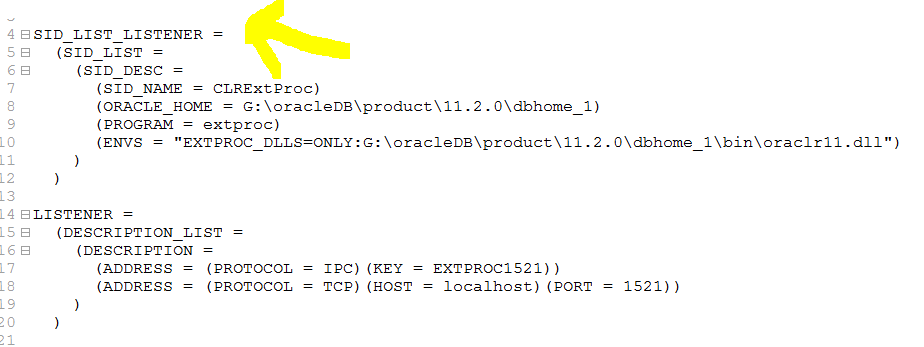
Alter system set local\_listener='(address\_list=(address=(protocol=tcp)(host =localhost)(port=1521)))';



#### 静态注册

将数据库的信息直接注册到监听器配置文件中，只要监听启动就会静态注册该服务，必须重启监听器才能生效。

$ lsnrctl stop listener;--重启监听器

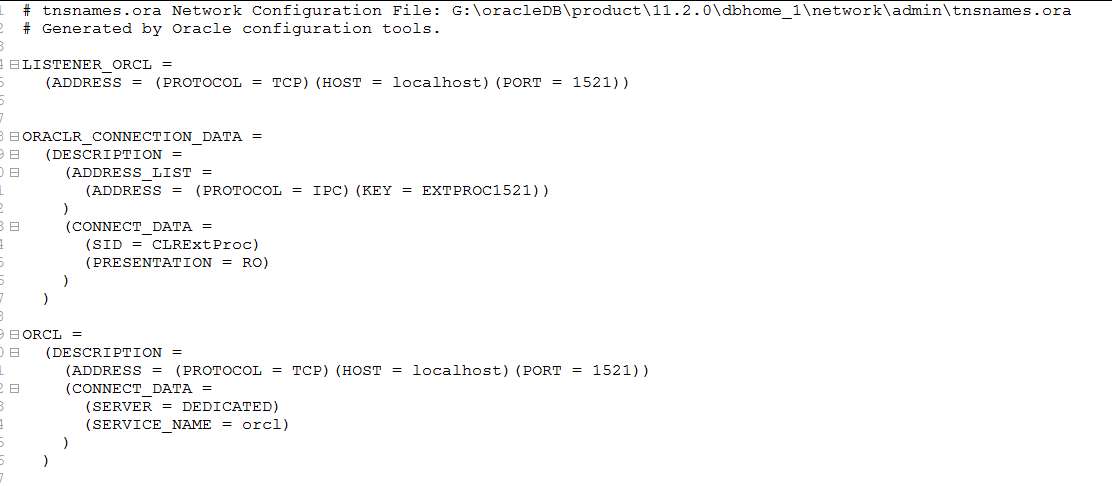


Oracle默认添加了静态。修改3个参数SID\_NAME/ORACLE\_HOME（不需要修改，默认路径是对的）/GLOBAL\_DANAME

#### 连接测试

作为客户端需要有自己的配置，客户端需要修改tnsnames.ora将客户端的服务名与监听器服务名之间建立映射关系，并告诉客户端软件要连接到数据库服务器的实体信息，这些信息包括通信协议，数据库服务器主机地址以及使用连接端口等。

该文件与listener.ora同目录。



$tnsping localhost;--测试客户端到监听器是否畅通

select sid,serial#,username,program,machine,status,port from v$session where username='SYSTEM';--确认该连接的存在和相关进程信息

$lsnrctl help;--查看相关命令信息

Start：启动监听程序，默认启动名为listener的监听程序，不是默认需要指定监听器名

Stop:关闭监听程序，默认启动名为listener，不是默认监听程序则需要指定

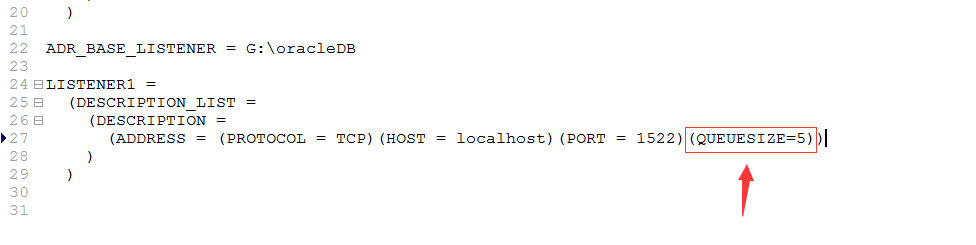
Service：查看监听为连接请求监控的服务内容

Reload:允许重载监听程序，如监听文件更改，可以使用该指令使得修改生效，在重载过程中已经建立连接的客户机继续保持连接

Set:设置监听管理密码，防止未授权而操作监听器

1：多监听设置，在一个服务器上可以运行多个监听程序，需要设置CONNECT\_TIME\_FAILOVER参数:当客户机使用新的监听程序建立连接前需要等待通过当前监听程序连接的时间长度。

2：设置队列长度，如果有大量客户请求连接导致监听程序无法提供足够的资源，此时发生的监听程序失败难以避免，控制客户连接到监听器的数量，告诉监听器可以为多少个连接提供并发服务。



3：设置监听程序密码

Set password/change\_password/save\_config;

### 客户端配置

客户端连接到服务器需要知道数据库服务器的ip，端口，协议等等，这些都在数据库客户端设置。信息存储在G:\oracleDB\product\11.2.0\dbhome\_1\NETWORK\ADMIN\ tnsnames.ora文件中。该文件主要设置连接到每一个数据库服务器的实体信息。

执行流程，

当使用sqlplus或其他方式初始化一个客户端连接时，oracle net组件会提供数据库详细信息（这些信息存储在tnsnames.ora文件中，信息包括数据库服务器的网络地址，通信协议以及连接端口。）

通过这些信息与监听程序建立通信，而后监听程序会将该连接交给SERVICE\_NAME指定的数据库，

一旦数据库服务器确定了用户名和密码匹配则建立客户端的连接。

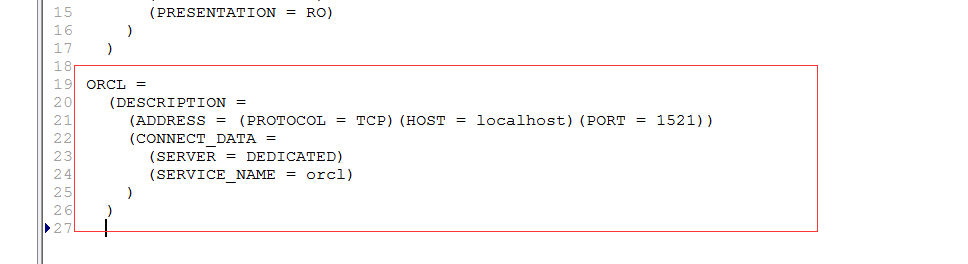
此时tnsnames.ora起到服务名解析作用，将一个服务名解析为一个网络连接实体，确定要连接的具体的数据库服务。

#### 本地命名

在tnsnames.ora中添加

Address:监听器实体信息

Connect\_date使用的服务器体系架构和对应的数据库服务名

SERVICE\_NAME:建立连接后，监听器根据此值找到注册的数据库

测试连接

Connect username/password@net\_service\_name

Sqlplus scott/tiger@orcl

#### 简单命名

不需要配置

Connect username/password@[//]host[:port][/service\_name]

//:可选

Host：必须，可以主机名也可以是ip

Port：可选，是默认端口可不写

Service-name:指定数据库服务器的服务名，默认该主机名不选。如果主机名和数据库服务器名相同，可省略。不同，必须提供数据库服务名的服务名

Sqlplus scott/tiger@localhost:1521/orcl

### ORACLE数据库服务器支持的2种连接方式

专有连接方式，一个用户连接对应一个数据库服务器进程。耗费内存资源(PGA)，减少竞争，长事务有用

共享连接方式，多个用户使用一个数据库服务器(分时复用)，对于事务执行时间短且服务器资源受限的系统是有利的。

#### 服务器进程

无论哪种连接方式连接数据库服务器都需要进程参与

专有服务器：采用专有服务器时，数据库服务器会针对每一个连接产生一个服务器进程，这个进程就是专属这个连接，数据库连接于服务器上的一个进程或线程是一一对应关系

共享服务器：采用共享服务器时，数据库服务器的一个服务器进程服务于多个会话，即使用服务器进程池为多个会话服务，在建立连接时，客户端会首先连接到调度器，由调度器协调数据库服务器的资源。

#### 共享连接

用户进程与服务器进程是多对多关系，多个服务器进程会处理多个用户进程。

在使用共享服务器连接时，必须使用oracle NET协议软件。不使用则无法使用共享服务器

客户端🡪调度器🡪SGA(请求队列)🡪共享服务器

客户端🡨调度器🡨SGA(响应队列)🡨共享服务器

##### 共享连接涉及初始化参数

调度器(DISPATCHERS):负责将用户请求传递放入SGA，并负责监控SGA中的响应队列，如果有返回数据则将数据返回给用户。

共享服务器进程(shared\_servers):设置共享服务器进程的数量

最大共享服务器进程(Max\_shard\_servers):该参数设置数据库服务器支持的最多的服务器进程数量

共享服务器会话数(shared\_server\_session):该参数设置当前数据库服务器在共享模式下的最大会话数，而专用会话数=session-共享会话数

##### 共享连接的工作过程

PMON定期将每个Dispatcher地址，以及工作负载注册到这个监听器里面。

1. 用户进程连接监听器
2. 监听器根据注册各个dispatcher的负载情况，选择一个负载最低的dispatcher，并将其地址返回给用户进程
3. 用户进程根据监听器返回的dispatcher地址连接到该dispatcher
4. Dispatcher接受到用户进程发出的请求以后，会将该请求放入请求队列(SGA)，请求队列被所有的dispatcher共有
5. 在服务器进程中，最空闲的服务器进程会从请求队列中按照先进先出的原则，挑选一个进行处理
6. 服务器进程处理请求后，得到的结果放入响应队列。Oracle为每一个dispatcher分配一个对应的响应队列
7. Dispatcher到相应的队列中取出结果，返回给用户进程

##### 共享连接的配置

##### 1:配置dispatcher进程的数量

Alter system set dispatchers=’(protocol=tcp)(dispatchers=3)’

属性

Address:dispatchers监听所在网络地址

Description：disptachers监听所在网络地址描述，包括网络协议地址

Protocol:指明disptachers使用的网络协议

指定在哪个地址上监听，dispatchers数量为2

Dispatchers=’(address=(protocol=tcp)(host=127.0.0.1))(dispatchers=2)’

设置数量时需要设置max\_dispatchers，限制最大数量

Alter system set max\_dispatcheres=5;

##### 2:配置共享服务器进程的数量

Alter system set shared\_servers=5;

Alter system set max\_shared\_servers=20

##### 3:配置共享会话数

Alter system set shared\_servers\_session=100

show parameter session;--查看最大会话数

最大会话数(session)-共享会话数(shared\_servers\_session)=专有连接会话个数

##### 共享连接常见问题

有些操作不能使用共享连接

1. 启动关闭数据库实例
2. 创建表空间和数据文件
3. 维护表和索引等数据库的管理工作

有些操作不适合共享服务器连接

共享服务器适合oltp，对于运行时间较长的操作，不适合采用(如备份)

#### 专有连接

专有连接中，用户进程没有发出命令，服务器进程处于空闲状态，资源一直占用，共享模式中，只要是服务器进程空闲，就可以处理其他用户发出的命令，因此服务器进程的数量减少，对资源的利用更加高效。占用的pga减少，可以支持更多的用户。

这是共享服务器的优势，相反就是专有服务器的劣势。

执行流程，客户应用程序通过api连接到数据库，而这些api知道如何将sql查询传递给数据库，并处理返回的数据。这些api知道如何将数据库请求打包成网络调用。而专有服务器进程则知道如何解包这些网络应用。整个过程由网络协议软件或oracle net支持

客户端🡪oracle net🡪专有服务器进程🡪SGA

使用本地命名连接数据时，使用默认专有连接。

#### 数据库驻留连接池(DRCP)

减少内存使用，同样的内存资源支持更多的用户

工作原理：DROP使用一个名为BROKER的中介，当用户连接时，它负责分配一个可用的池服务器，如果池中没有空闲池服务器，只要没有超过池服务器的最大值，BROKER就可以创建新的池服务器，如果达到最大池服务器数目，则BROKER将用户连接放入请求队列，直到有空闲池服务器为止。

DROP使用的内存量只与活跃的池服务器数量有关。这是他与上述2中连接方式相比的优势

Select connection\_pool,status,maxsize from dba\_cpool\_info;--查看连接池信息

Status默认禁用。

Exec dbms\_connection\_pool.start\_pool();--开启DRCP

Exec dbms\_connection\_pool.stop\_pool();--关闭DRCP

##### 配置DRCP

DBA\_CONNECTION\_POOL:实现修改或设置配置

CONFIGURE\_POOL:设置参数值

ALTER\_PARAM:修改参数

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

INACTIVITY\_TIMEOUT:池服务器禁止服务前的最大空闲时间

MAX\_LIFETIME\_SESSION:每个池会话的存活时间

MAX\_USE\_SESSION:一个池服务器放到连接池的最大次数

MAXSIZE/MINSIZE:池服务器最大和最小的个数

MAX\_THINK\_TIME:用户获得池服务器后保持不活动的最大时间

上述参数可以通过dbms\_connection\_pool包中的过程来设置

查询默认连接池的状态以及相关参数

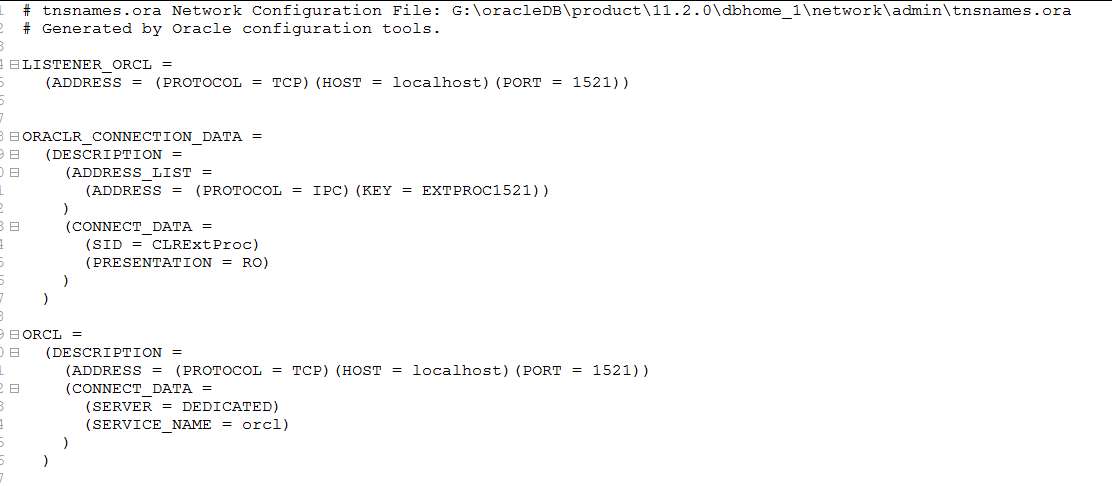
Col connection\_pool for a30

Set line 120;

Select connection\_pool,status,maxsize,minsize,inactivity\_timeout from dba\_cpool\_info

exec dbms\_connection\_pool.alter\_param('','maxsize','70');--设置池服务器最大数

配置TNSNAMES.ora,使该客户端的连接使用DRCP连接池功能



图中SERVER设置为POOLED

select pool\_name,num\_open\_servers,num\_requests,num\_hit,num\_misses from v$cpool\_stats;--查看池服务器状态信息

num\_open\_servers:服务器池中空闲和被占用的服务器数量

num\_requests:客户端请求服务器池中的服务器的次数

num\_hit:客户端请求池中的服务器且获得服务器的次数

num\_misses:客户端请求池中的服务器而没有获得该服务器的次数

查询DRCP下用户和进程信息(使用drcp连接的用户),其中包含会话id，进程id信息

Select session\_addr,username,process\_id,connection\_status from v$pool\_conn\_info;

## 内存管理

### 内存架构

数据库启动时会首先分配sga(数据库实例一部分)，pag可以成为私有变量区，是某个用户独有。专有连接模式下，每个用户会话都会分配一个pga，用户保存会话信息。Sga与pga之间是服务器进程，最终是服务器进程访问sga来满足用户的数据访问的

数据库实例启动时，oracle数据库会分配内存区并启动后台进程，内存区存储了如下的信息

1. 程序代码
2. 建立过连接的会话信息(即使该会话当前不是活跃的inactive)
3. 程序执行期间所需信息，如sql查询的当前状态
4. 在进程间共享并且进行通信的信息如锁数据
5. 缓存数据，如数据库快，重做快。缓存这些在磁盘上的数据

Oracle数据库基本的内存结构包括

系统全局区(SGA)：一组共享的内存结构，包括数据库实例所需的数据和控制信息，所有的服务器进程和后台进程共享

程序全局区(PGA):是非共享的内存区域，包含oracle进程所独自使用的数据和控制信息，pga与服务器进程和后台进程共存。Pga是为服务器进程和后台进程服务的内存区，为这些进程独有。为整个实例拥有

UGA用户全局区：和用户会话相关联

软件代码区：用于存储正在运行的程序代码

#### PGA概述

存储专有或共享服务器进程所需的会话相关变量。服务器进程负责在pga中分配它所需要的内存结构

PGA组件(可手动调整各个组件的内存分配)：

私有sql区：包含解析后的sql语句和处理所需要的会话信息，当服务器进程运行sql/plsql代码时，该进程使用私有sql区存储绑定变量的值，查询执行状态信息以及查询执行工作区。在同一个或不同会话中的多个私有sql区可以执行sga中的一个执行计划，此时对于每一个执行计划的私有sql去不共享，各自包括不同的值或者数据

私有区域包含2部分

* 运行时区：该区域包含sql查询执行状态信息，如运行时区跟踪全表扫描中到目前为止返回的数据行数，在执行请求的第一步oracle数据库就创建运行时区，对于dml语句运行时区在sql语句关闭后自动释放内存
* 永久数据区：这部分存储绑定变量，绑定变量的值在sql语句运行时赋予该语句，永久数据库只有游标关闭后才释放，如果增加了session\_cache\_cursors的值，此时在私有SQL区就存储了较多的执行计划，此时与该执行计划相关的绑定变量占用空间不会释放，造成占用过的pga资源。所以在pga优化时要主语是否增加pga大小

客户端进程负责管理私有SQL区，私有sql区的分配和回收主要依赖于应用。客户端进程能够分配的私有sql区的数量收到初始化参数open\_cursors的限制。

SQL工作区:是一个私有内存区域，在pga中分配，用于内存密集型的操作，如排序操作符使用排序区来排序行集合。hash-join操作符使用哈希区从他的左输入构建哈希表，然而bitmap merge使用bitmap merge区来合并从多位图索引扫描获得的数据。

如果操作符处理的数据总量过大，不能放入工作区，此时oracle数据库将输入分割成更小的碎片，用这种方式，数据库在内存中处理数据片，同时将其余无法处理的数据放入临时表空间待以后处理。

#### SGA概述

是可读写的内存区，SGA和后台进程组成了oracle数据库实例，代替用户工作的服务器进程可以读sga中的数据，服务器和后台进程不驻留sga中，而是存在于独立的内存空间中

每一个数据库都有自己的sga，数据库实例启动时候自动分配sga，关闭时自动关闭sga

SGA由几个内存组件组成，称为内存池，这些内存池的空间分配以granules为单位

即粒度为单位。粒度是一个连续的内存空间。粒度的大小与操作平台有关，由整个sga的大小决定。

* Database buffer cache:数据库高速缓存
* Redo log buffer:重做日志缓存
* Shared pool:共享池
* Large pool:大池
* Java pool：java池
* Streams pool:流池
* Fixed sga：固有区域

数据库高速缓存是一个共享的内存结构，这意味着发生内存争用的事件都发生在共享的内存结构中。数据库高速缓存用来存储从数据文件读取的数据块，临时存储当前或最近读取的数据块，提高oracle数据库读写数据的效率，

#### UGA概述

会话内存，为某个用户会话服务，给会话分配内存存储会话变量，这些变量包括登录信息以及会话所需的其他信息

如果会话加载了plsql包，此时该包的状态就存在UGA中，这个状态是指在指定时刻包变量的值得集合，默认包变量在会话周期内是唯一并且持久存在的。

Olap页池：该池管理OLAP数据页，他与数据块等同。随着OLAP会话的启动和结束而分配和释放内存。

共享服务器模式下，UGA在SGA中分配，这样任何共享服务器进程都可以访问它。

专有服务器模式下，UGA在PGA中分配，因为PGA只为单个进程服务。不需要共享

### 内存管理

Alter system set memory\_target=1000M scope=both;--设置后Sga和PGA自动调整，根据负载状态协调内存的使用。因为是动态参数，所以无需重启数据库

Memory\_max\_target设置memory\_targer的上限

使用DBCA建库的话，默认启动自动内存管理

#### 配置内存组件

如果没有启动自动内存管理（如使用DBCA建库时设置了SGA和PGA相关参数的值，或者使用了CREATE DATABASE 手工建库时没有设置MEMORY\_TARGET参数），此时就需要启动自动内存管理特性

设置MEMORY\_TARGET值，这个值通过SGA\_TARGET+PGA\_AGGREGATE\_TARGET计算

Memory\_target=sga\_target+max(pga\_aggregate\_target,maximum PGA allocated)

Sga\_target和pga\_aggregate\_target可通过show parameter target获得

Maximum pga allocated通过select value from v$pgastat where name='maximum PGA allocated';获得

考虑到SGA和PGA的将来预期，所以MEMORY\_MAX\_TARGET和MEMORY\_TARGET值相同,他是一个静态参数。需要如下修改

ALTER SYSTEM SET MEMORY\_MAX\_TARGET=Nm Scope=SPFILE;

如果使用PFILE启动数据库，也可以在初始化参数文件中写入如下参数值

Memory max target=nM

Memory\_target=Mm

SELECT \* FROM V$MEMORY\_TARGET\_ADVICE ORDER BY MEMORY\_SIZE; --查看对于MEMORY\_SIZE的大小设置建议，

MEMORY\_SIZE\_FACTOR=1(内存尺寸因子)的MEMORY\_SIZE为当前的内存大小，也就是参数memory\_target的大小

ESTD\_DB\_TIME：完成当前负载预计的DB时间

SHOW PARAMETER MEMORY TARGET;--查看memory\_target的大小,可以发现与动态视图V$MEMORY\_TARGET内存因子为1的memory\_size大小相等

因为设置了参数memory\_target所以sga\_target和pga\_ aggregate \_target根据负载和事务情况自动调节,对应的这2个参数的值为0

如果手动设置sga\_target和pga\_aggregate\_TARGET的大小，则oracle认为这个大小值为对应参数的最小值

Show parameter sga/pga;--查看pga和sga的大小，因为开启了自动管理所以都为0

#### SGA和PGA的自动调整

推荐使用自动内存管理

Show parameter area\_size;--查询pag相关组件的大小

Workarea\_size\_policy：决定PGA自动调整还是手工调整。如果手动调整必须要设置为Manual

DBCA建库



#### 配置数据库smart flash缓存

Smart flash cache：11g版本提供，默认此内存组件没有启动

作用：缓存更多的数据，提高读数据的效率，减少cpu的负担以及减少数据库高速缓存的压力

要求

1. 必须是solaris/oracle Enterprise Linux系统
2. 通过AWR/statspck报告提示加倍数据库高速缓存的大小
3. CPU资源短缺

启动smart flash cache后，从数据高速缓存移动到flash cache中的数据块而言，将有部分数据块的元数据存储在数据库高速缓存中.

手工调整内存，需要增加数据库高速缓存的大小，其中值为进入smart flash cache的数据块数量\*100。如果采用自动内存调整则需要调整MEMORY\_TARGET，大小参考手工调整内存的大小。

自动共享内存调整(sga组件自动调整)：需要增加sga\_target的值

设置smart flash cache 需要设置2参数,db\_flash\_cache\_file/db\_flash\_cache\_size

指定存储smart flash cache数据块的磁盘目录以及名称，要求必须在flash盘上存储，否则影响性能。

Show parameter flash\_cache;--查询smartFlashCache涉及到的2个参数

设置2个参数值,静态参数需要重启

Alter system set db\_flash\_cache\_file=’文件目录’ scope=spfile;

Alter system set db\_flash\_cache\_size=1g scope=spfile;

重启

Shutdown immediate;

Startup;

## 用户管理和资源文件

### 创建用户

Create user jane 创建用户jane

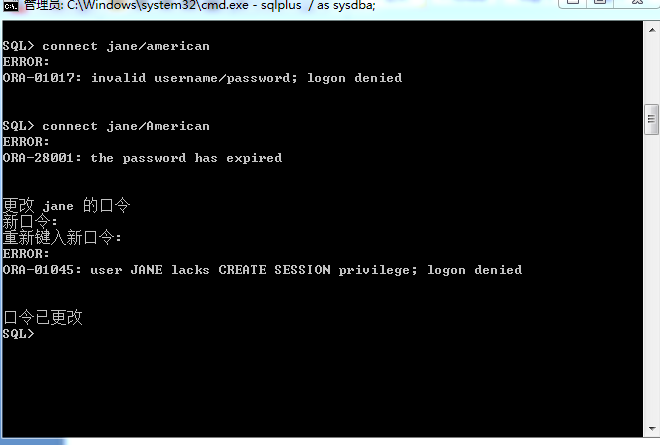
identified by American 设置用户密码

default tablespace users 设置默认表空间为users表空间，该表存储用户数据

temporary tablespace temp 创建临时表空间temp，该表空间用户诸如排序等操作的数据空间

quota 10m on users 设置该用户对于表空间users的配额为10M

password expire; 说明用户jane登录数据库时，密码立即失效，oracle会提示重新输入密码



登录后密码立即失效，更改后无法登录，因为没有授予该用户create session的权限。

所以无法建立与数据库的会话连接。

Grant create session,resource to jane;为用户jane授予create session权限

Select username ,password,expiry\_date,default\_tablespace,temporary\_tablespace, created from dba\_users where username='JANE';使用数据字典查看用户jane的信息

Select tablespace\_name,username,max\_bytes from dba\_ts\_quotas where username ='JANE';查看jane的表空间配额信息,就是创建用户时quota属性的值

### 创建用户语法及参数含义

CREATE USER user:创建用户user

IDENTIFIED {BY password | EXTERNALLY}:设置用户密码，EXTERNALLY说明该用户由操作系统授权，参数不能省略

[DERAULT TABLESPACE tablespace]:设置用户默认表空间

[TEMPORARY TABLESPACE tablespace]:设置用户的临时表空间

[QUOTA {integer [K|M]} ON tablespace ]:设置该用户对于表空间的配额，即表空间多大给该用户使用，参数UNLIMITED说明没有限制。K|M配额单位

[QUOTA {integer [K|M]} ON tablespace ]….

[PASSWORD EXPIRE]：设置用户密码第一次使用时作废，需要重新设置该用户密码

[ACCOUNT {LOCK|UNLOCK} ]:选择是否锁定该用户，默认值unlock

[PROFILE {profile|default}]:使用指定的概要文件，profile为概要文件名。不指定默认使用default的概要文件，默认的概要文件对所有限制的初始值 都没有限制

### 改变用户参数

用户创建数据库对象如表，索引时，没有指定存储表空间就存放在创建用户时的默认表空间，会造成内存不足。

解决方案1：修改默认表空间的quota参数，2：增加默认表空间

修改用户的默认表空间配额

Alter user jane quota 20m on users;

为用户jane增加一个默认表空间(newtbs，要事先创建)，配额unlimited无限制。

Alter user jane default tablespace newtbs quota unlimited on system;

回收用户jane在表空间users的使用权用户jane在表空间users的使用权

Alter user jane quota 0 on users;通过users配额设置0对其 回收,-1表示无限制

注意：如果在回收users表空间之前，已经在该表空间上使用了5M空间，则不能给用户jane分配使用空间。

### 删除用户

Drop user jane;删除用户，如果在删除时，该用户已经连接到了数据库服务器，则无法删除。可以断开该用户的连接在删除用户

select username from dba\_users where username='JANE'; 未选定行,表示已经删除

查看当前系统上的所有用户信息

select username ,account\_status,created from dba\_users;

account\_status:表示用户的状态，open可用。Expired过期，locked锁定

解锁用户:alter user scott account unlock

设置密码:alter user scott identified by tiger;

解锁用户并设置密码:alter user scott identified by tiger account unlock;

### 用户和数据模式

创建用户后，需要授予该用户权限使得它可以创建数据库对象如表，索引，触发器等。

模式与用户一一对应，且名字相同。用户创建成功时，也对应的创建了一个模式。

模式是命名的数据库对象的集合。

模式对象

* 表
* 触发器
* 约束
* 索引
* 视图
* 序列号
* 存储过程
* 同义词
* 用户定义的数据类型
* 函数
* 软件包

查看用户scott所拥有的模式对象

Select distinct(object\_type) from user\_objects;

### 用户管理中的重要文件-概要文件

创建用户后需要给该用户各种系统资源，如cpu，并行会话数，空闲时间限制等资源限制。同时需要对口令做出更详细的管理方案，如尝试登陆指定的次数后账户锁定，口令过期之后的处理等。如果对每个用户都进行资源限制或口令管理，要输入大量的指令。当用户过多效率过低，所以引入概要文件进行管理用户。

概要文件：一组指令的集合，这些指令限制用户资源的使用或口令的管理。

2种指令赋予用户指定概要文件alter user指令和CREATE USER时指定的profile参数，没有指定概要文件，自动使用默认概要文件

使用资源管理和口令管理的概要文件步骤

1. 使用create profile 指令创建一个概要文件
2. 使用alter user(已有用户)或create user(新用户)将概要文件赋予用户
3. 对于资源管理而言，启动资源限制，修改动态参数resource\_limit为true，此时可以通过修改参数文件或者alter system来修改。

### 使用概要文件管理会话资源

格式如下

CREATE PROFILE profile\_name LIMIT

[SESSIONS\_PER\_USER n]:表示每个用户的最大会话数

[CPU\_PER\_SESSION n]:每个会话占用的cpu时间，单位0.001秒

[CPU\_PER\_CALL n]:每个调用占用的CPU时间，单位0.001秒

[CONNECT\_TIME n]:每个连接支持连接的时间

[IDLE\_TIME n]：每个会话的空闲时间

[LOGICAL\_READS\_PER\_SESSION n]:每个会话的物理和逻辑读数据块数

[LOGICAL\_READS\_PER\_CALL n

创建资源限制概要文件

Create profile scott\_prof limit

Sessions\_per\_user 10

Cpu\_per\_session 10000

Idle\_time 40

Connect\_time 120;

概要文件名为scott\_prof，每个用户的并行会话数为10，每个会话的cpu时间为1000秒，连接空闲时间为40分，保持连接的时间为120分

查看概要文件scott\_prof

Select \* from dba\_profiles where profile=’SCOTT\_PROF’ order by limit;

Resource:kernel表示一个资源参数，password表示一个安全限制。

除了创建概要文件时指定的值，其余都默认值

### 口令管理参数以及含义

创建口令管理的概要文件与创建资源限制的概要文件一样，都是create user/alter user指令将概要文件赋予用户。口令文件一旦赋予用户立即生效。

查看scott\_prof概要文件的口令管理参数

Select resource\_name from dba\_profiles where profile='SCOTT\_PROF' and resource\_type='PASSWORD';

FAILED\_LOGIN\_ATTEMPTS:尝试登录失败的次数，超过则锁定

PASSWORD\_LIFE\_TIME:口令有效实现，超过指定的天数则口令无效

PASSWORD\_REUSE\_TIME:口令在能够重用之前的天数

PASSWORD\_REUSE\_MAX:口令能够重用之前的最大变化数

PASSWORD\_VERIFY\_FUNCTION：在为一个新用户赋予口令之前要检验口令的复杂性是否满足要求的一个函数。该函数使用plsql编写，名字为verify\_function

PASSWORD\_LOCK\_TIME：登录失败后，锁定的天使

PASSWORD\_GRACE\_TIME：口令过期之后还可以继续使用的天数

口令要求

最小长度4字符，不能与用户名相同，至少包含一个字符一个数字一个特殊字符。

新口令至少有3个字母与旧口令不同

要使用oracle提供的口令验证函数，需要先运行一个名为utlpwdmg.sql的脚本文件

执行创建口令复杂性验证函数,创建VERIFY\_FUNCTION函数,更改默认概要文件

@ G:\oracleDB\product\11.2.0\dbhome\_1\RDBMS\ADMIN\utlpwdmg.sql

验证口令函数VERIFY\_FUNCTION函数是否创建

Select owner,object\_name,object\_type,created from dba\_objects where object\_type='FUNCTION' and object\_name='VERIFY\_FUNCTION';

尝试修改scott密码，修改失败，因为oracle不符合函数VERIFY\_FUNCTION中定义的规则之一，说明口令管理的概要文件即时生效

alter user scott identified by oracle;

### 创建口令管理的概要文件

CREATE PROFILE profile\_name LIMIT

[parameter1 para\_valuel]

[parameter2 para\_valuel]

…

parameter1，parameter2为上述7个参数

创建口令管理的概要文件

create profile password\_prof limit

failed\_login\_attempts 5 :登录失败5次，锁定

password\_life\_time 90:密码90天内有效

password\_reuse\_time 30:失效后30天后才能使用

password\_lock\_time 15:登录失败后，锁定15天

password\_grace\_time 3;口令过期后，4天内可以使用旧口令(过期)登录数据库

查看密码概要文件password\_prof的口令参数

Select \* from dba\_profiles where profile='PASSWORD\_PROF' and resource\_type= 'PASSWORD';

### 修改和删除概要文件

修改口令管理概要文件的参数

Alter profile password\_prof limit

failed\_login\_attempts 3

Password\_life\_time 60

Password\_grace\_time 7;

删除概要文件,如果删除的文件关联了用户，需要使用cascade

Drop profile password\_prof;

drop profile scott\_prof;

## 控制文件和数据库启动

### 控制文件和数据库启动概述

2进制文件

记录数据库的状态信息，如重做日志文件与数据文件的名字和位置，归档重做日志的历史等。

<=64M

Mount阶段被读取，

一个控制文件只能与一个数据库相关联。

只有数据库服务器可以修改控制文件中信息

数据库启动与控制文件的关系

启动数据库(通过默认搜索规则找到参数文件)🡪打开参数文件(在参数文件中找到控制文件的位置)🡪打开控制文件

### 如何获得控制文件信息

通常一个数据库文件至少需要3个控制文件，控制文件最好不要放在一起，可以防止磁盘故障造成数据库无法启动。

可以通过参数文件获得控制文件位置，但是这样容易修改。Oracle提供了视图

使用视图v$paramete查看控制文件位置

select value from v$parameter where name='control\_files';

使用show parameter查看当前控制文件的位置

Show parameter control\_files;不添加S也可以

通过数据字典v$controlfile查看控制文件的名字和存储目录

Select status ,name from v$controlfile;

### 控制文件的内容

#### 内容

* 数据库名：在初始化DB\_NAME获得，或者CREATE DATABASE语句执行时用的名字
* 数据库标识符：数据库创建时oracle记录得标示符
* 数据库创建时间：创建数据库时由oracle自动记录
* 表空间信息：当表增加或者删除表空间时记录该信息
* 重做日志文件历史：在日志切换时记录
* 归档日志文件的位置和状态信息：在归档进程发生时完成
* 备份的状态信息和位置：由恢复管理器记录
* 当前日志序列号：日志切换时记录
* 校验点信息：当校验点事件发生时记录

#### 查看所存内容的记录信息

查询控制文件中所存内容的记录信息

select type , record\_size, records\_total,records\_used from v$controlfile\_record\_section;

很多动态数据字典视图就是从控制文件中获得数据。

* V$backup
* V$database
* V$tempfile
* V$tablespace
* V$archive
* V$log
* V$logfile
* V$loghist
* V$archived\_log
* V$database

查看数据库id，创建时间，数据库是否处于归档模式等

Select name,created,log\_mode from v$database;

### 存储多重控制文件

Oracle数据库会同时维护多个完全相同的控制文件，称为多重控制文件

在不同磁盘上存储多重控制文件可以避免控制文件的单点失效问题。

如果一个磁盘的控制文件失效，oracle会自动使用参数文件中记录的其他控制文件启动数据库。

控制文件目录

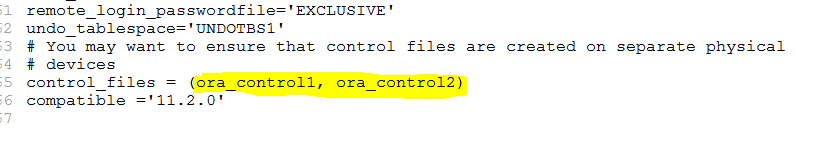
G:\ORACLEDB\ORADATA\ORCL\CONTROL01.CTL, G:\ORACLEDB\FLASH\_RECOVERY\_AREA\ORCL\CONTROL02.CTL

参数文件目录

G:\oracleDB\product\11.2.0\dbhome\_1\dbs\init.ora

#### 使用pfile(init.ora)文件时移动控制文件

可以直接打开修改。黄色区域更换为控制文件地址即可。



关闭并重启数据库，

验证控制文件的修改结果

Select \* from v$controlfile;

#### 使用spfile文件时移动控制文件

修改spfile中的控制文件名

Alter system set control\_files=

'G:\ORACLEDB\ORADATA\ORCL\CONTROL01.CTL',

' G:\ORACLEDB\FLASH\_RECOVERY\_AREA\ORCL\CONTROL02.CTL'

SCOPE=SPFILE;

#### 添加控制文件

CONTROL01.CTL复制到另外一个目录，多添加一个文件的目录即可。

Alter system set control\_files=

'G:\ORACLEDB\ORADATA\ORCL\CONTROL01.CTL',

' G:\ORACLEDB\FLASH\_RECOVERY\_AREA\ORCL\CONTROL02.CTL'

SCOPE=SPFILE;

### 备份和恢复控制文件

#### 备份

使用database backup controlfile备份控制文件

alter database backup controlfile to 'd:\ss.ora';

备份控制文件，将控制文件备份到追踪文件中，使用该文件可以恢复控制文件。

1：设置参数sql\_trace=true,默认false

alter session set sql\_trace=true;

2:备份控制文件到追踪文件中

Alter database backup controlfile to trace;

查看跟踪文件的存储目录

Show parameter user\_dump\_dest;

#### 恢复

1：部分控制文件损坏

关闭数据库，将完好的控制文件复制到已经丢失或者出错的控制文件的位置。更改文件名字为该丢失或者出错的控制文件的名字。

如果目录也被损坏，重建新目录放控制文件。需要修改数据库初始参数文件中控制文件的位置信息。

2:控制文件全部损坏或丢失

关闭数据库，将备份的控制文件复制到先前控制文件所在位置，

并更改备份控制文件名为先前控制文件的文件名

打开数据库到mount状态

Startup mount

打开数据库

Alter datebase open using backup controlfile;

3:通过跟踪文件重建控制文件

跟踪文件记录了用于建立控制文件的sql语句，适当的编辑下。然后使用slq命令即可

G:\oracleDB\diag\rdbms\orcl\orcl\trace\alert\_orcl.log

4：手工重建控制文件

Create controlfile 指令逐步创建控制文件

## 重做日志管理

作用数据恢复。

用户更改的信息会暂时存放在数据库高速缓冲区中，是为了提高读写的速度。当达到一定的容量，DBWR就把变化的数据写到数据库中。中间假如出现了down就会数据无法恢复。所以引入重做日志管理，LGWR进程负责把用户更改的数据优先写到重做日志中。称为日志写优先。Down机后，重启时，数据库系统会从重做日志文件中读取变化的数据。

为了提高磁盘效率，并为了防止重做日志文件的损坏。Oracle引入了一种重做日志结构

重做日志文件结构由3个重做日志组组成，每个日志组有2个重做日志成员(文件)，可以设置更多。数据库优先使用日志组1，该组写满后，切换到日志组2并为该日志组设置一个日志序列号，在写满后切换到日志组3，然后循环使用日志组1。

Oracle规定每个数据库实例最少要有2个重做日志组，每个重做日志组有一个重做日志成员。每个重做日志成员内容相同。

没有启动归档日志，当一个循环结束，在次使用之前的重做日志组。覆盖形式写入数据。

非归档模式下，重新使用心得联机重做日志前，dbwr进程会将所有的数据写入数据文件中。称为dbwr归档。

归档模式下，日志写满后，oracle会关闭当前的日志文件，ARCH进程把旧的重做日志文件中的数据移动到归档重做日志文件中。寻找下一个可用重做日志组。进行写操作。归档进程并不是一直存在。

再归档模式下， oracle无法使用一个已经关闭的重做日志，所以需要等待arch释放了联机重做日志后，数据库才能工作。

### 读取重做日志文件信息

#### V$log视图

记录当前数据库的日志组号，日志序列号，每个日志文件的大小(字节单位)，每个日志组的成员数量，以及日志组的当前状态。

Select group#,sequence#,bytes,members,archived,status from v$log;

输出说明，

当前有3个日志组，每个日志文件对应的日志序列号(全局唯一，同一个日志组中的日志序列号相同)，每个日志组成员数量，状态(CURRENT表示当前正在使用),ARChive表示是否归档模式

#### V$logfile视图

记录当前日志组号，状态，类型和成员信息

Col mumber for a10

Select group#,status,type,member from v$logfile;

Status参数的含义

* Stale:该文件内容不完整
* 空白:该日志组正在使用
* Invalid：该文件不能被访问
* Deleted：该文件不在使用

Type：online表示联机状态

#### 归档模式

查看当前数据库是否处于归档模式

Archive log list;

设置数据库为归档模式

1:关闭数据库，到mount状态

2：修改

Shutdown immediate

Startup nomount;

Alter database mount;

Alter database archivelog;

上述可知，存档终点：USE\_DB\_RECOVERY\_FILE\_DEST

查看USE\_DB\_RECOVERY\_FILE\_DEST指定的文件目录，与大小

Show parameter db\_recovery\_file\_dest;

Oracle9i设置归档模式

设置相同的步骤，另外修改参数log\_archive\_start=true

可以修改init.ora，开启log\_archive\_start=true。

第二个是使用spfile.

### 重做日志组及其管理

添加重做日志组格式

ALTER DATABASE [database\_name]

ADD LOGFILE [GROUP number] filename SIZE n

[,ADD LOGFILE [GROUP number] filename SIZE n…..]

ALTER DATABASE ADD LOGFILE GROUP 6

('G:\oracleDB\temp\redo.log' , 'G:\oracleDB\temp\redo2.log')

size 11m;

不选择group参数，默认在原有日志组号的基础上自动增长。

删除重做日志组

ALTER DATABASE [database\_name]

DROP LOGFILE{GROUP n|(‘filename’[,’filename’]…)}

{GROUP n|(‘filename’[,’filename’]…)}…

Alter database drop logfile group 4,group 5,group 6;

注意：删除重做日志组后，作为重做日志组成员的操作系统文件还在。

### 重做日志成员及维护

#### 添加重做日志成员

格式：

ALTER DATABASE [databasename]

ADD LOGFILE MEMBER

[‘filename’ [REUSE] [, ‘filename’ [REUSE]] …

TO {GROUP n | (‘filename’ [ , ‘filename’]…)}…

]…

Alter database add logfile member

'G:\oracleDB\temp\a1.log' to group 1,

'G:\oracleDB\temp\a2.log' to group 2,

'G:\oracleDB\temp\a3.log' to group 3;

查看成员个数

Select group#,sequence#,bytes,members,archived,status from v$log;

查看重做日志组以及对应成员信息

Select group#,status,type,member from v$logfile;

注意：添加日志成员文件已存在，则需要使用reuse参数。日志成员要使用绝对路径，否则oracle数据库会在默认路径下建立该重做日志文件

#### 删除联机重做日志成员

格式如下

ALTER DATABASE [database\_name]

DROP LOGFILE MEMBER ‘filename’ [,’filename’]…

只需要知道重做日志成员的目录和文件名，但是操作系统中该成员文件没删除

Alter database drop logfile member 'A2.LOG';

删除限制

* 删除的日志成员是日志组最后一个有效成员，不能删除
* 该日志组正在使用。
* 数据库运行在ARCHIVELOG模式，并且要删除的日志成员所属的日志组没有归档

#### 重设联机重做日志的大小

Oracle没有提供直接修改联机重做日志大小的方法。

可以使用删除日志文件所在日志组，新建一个日志文件设置大小，来间接的达到目的

切换日志并删除当前没有使用的日志组

alter system switch logfile;

如果该日志组处于inactive，可以直接删除。如果处于active，继续使用日志切换指令加速重做日志组的归档工作。或者此时强制加入一个检查点。Alter database checkpoint。使得要删除的重做日志组处于inactive状态。因为强制检查点使DBWR将保存在联机重做日志中已经变化的内容写到了数据文件中。

### 清除联机重做日志

在数据库处于归档模式(ARCHIVELOG)时，如果当前正在使用的重做日志组中的重做日志文件损坏，则该重做日志不能完成归档，导致数据库因无法归档而挂起。

解决方案:通过清除联系机重做日志来重新初始化联机重做日志文件。

ALTER DATABASE CLEAR LOGFILE GROUP n;

注意：使用指令后，已删除的重做日志组中重做日志的序列号变为0，所以此时需要做数据库的全备份。因为oracle在进行数据库恢复时，需要连续的序列号。

### 日志切换和检查点事件

当一组重做日志组写满或者用户发出alter database switch logfile时，就会触发日志切换。此时oralce寻找下一个可用的重做日志组，如果数据库处于归档模式，则在将当前写满的日志组归档完成前不会使用新的重做日志组。

检查点事件是oracle减少数据库实例恢复时间而设置的一个事件。当该事件发生时LGWR进程将重做日志缓冲区中的数据写入重做日志文件中。而同时通知DBWR进程将数据库高速缓存中的已经提交的数据写到数据文件。所以检查点时间越频繁则用于数据库恢复的重做数据就越少。检查点事件也会修改数据文件头信息和控制文件信息。以记录检查点的SCN

强制启动检查点事件

ALTER DATABASE checkpoint;

如果不是强制性的产生检验点事件，检验点事件由DBWR数据库写进程触发。

### 归档重做日志

是联机重做日志的脱机备份。当数据库处于归档模式，发生日志切换或者日志写满，就会触发归档进程ARCH把重做日志文件中的数据移动到归档重做日志中。归档完毕，归档进程自动关闭。

归档日志文件存储在参数文件SPFILE或init.ora(log\_archive\_dest\_n)文件参数指定的位置。

## 管理归档日志

作用：保证数据库发生介质故障时，可以完全恢复数据库

Log buffer 🡪lgwr进程🡪redo log🡪arch进程🡪archived log

查看数据库的归档模式

Archive log list;

查询默认归档目录

Show parameter db\_recovery\_file\_dest;

启动归档模式,此时做一次数据库的全备份，包括数据文件和控制文件。Oracle11G自动启动归档进程，之前的版本需要手动启动

Alter database archivelog;

### 设置归档进程与归档目录

归档进程把重做日志文件读取到写入归档日志文件，这个过程相对LGWR工作慢。

如果当前的生产数据库的DML操作很频繁，产生重做数据的效率很高，有可能出现归档进程无法匹配LGWR进程的工作。即归档进程慢，造成写重做日志的等待。

解决方案，启动更多的后台归档进程。

查看最大的日志归档进程。

show parameter log\_archive\_max\_processes;

修改最大日志归档进程

alter system set log\_archive\_max\_processes=4;

归档日志是实现数据库完全恢复的关键数据，必须做好归档日志的保护工作。Oracle提供了30个归档目录，用来冗余归档日志备份。只要我们实现了归档日志的物理冗余备份，就极大保证了归档日志的高复用性，防止发生数据库介质故障造成的数据丢失。

查询归档日志相关参数

show parameter log\_archive\_Dest;

其中log\_archive\_dest\_n参数指定归档目录的绝对路径，

log\_archive\_dest\_state\_n参数指定这些归档目录的状态。可以用来归档目录的维护。

Log\_archive\_dest\_n设置相关的关键字

Location 本地文件系统或ASM磁盘组log\_archive\_dest\_n=’location=[/disk1/arc]|[+DGP]’

Location使用快速恢复区FastRecovery Area log

Log\_archive\_dest\_n=’LOCATION=USE\_DB\_RECOVERY\_FILE\_DEST’

Service通过OracleNetServiceName的远程归档

Log\_archive\_dest\_N=’service=sd1’

设置参数log\_archive\_dest\_1

Alter system set log\_archive\_dest\_1=’location=f:\temp\mandatory’

Location ：归档日志在本地磁盘，设置目录不存在，默认去上一层目录存，不存在或者损坏则数据库挂起。

Mandatory:强制该目录下的归档日志写成功，否则未成功的归档的重做日志组不能重用，数据库会挂起。也可以设置optional

Optional:即使该归档目录对应的重做日志没有向该目录写成功，所对应的重做日志文件依然可以使用。默认optional。

查询归档目录

Show parameter log\_arrchive\_dest\_1;

验证归档终点设置

Select destination ,binding,target,status from v$archive\_dest where destination is not null;

为了保证归档文件高可用性，依然需要使用冗余方法实现归档数据的保护，同时oracle提供了参数log\_archive\_min\_succed\_dest，以保证最少的成功归档终点。

即如果设置了3个归档目录，而参数值为3，则至少保证其中3个归档终点必须归档成功。

查询默认值

Show parameter log\_archive\_min\_succeed\_dest;

修改参数值，强制保证至少2个归档目录归档成功

Alter system set log\_archive\_min\_succeed\_dest=2;

如果只需要设置归档目录log\_archive\_dest即可，如果设置主备2个归档目录，可设置Log\_archive\_dest和log\_archive\_duplex\_dest;

查询归档文件的格式

Show parameter log\_archive\_format;

ARC%S\_%R.%T：%S表示日志序列号，%R表示resetlogs ID,%T表示线程号(单实例数据库，此线程号不会改变)。

### 维护归档目录

归档目录状态，即log\_archive\_dest\_state\_n参数的值

ENABLE:磁盘目录或者服务名换已经指定，并且是有效的

DEFER:目录临时禁止，维护归档目录时，会经常使用

ALTERNATIVE:该目录为备用目录，当与其对应的主目录发生故障，自动启动。

查看归档目录的状态

Select dest\_name,status,binding,process,error from v$archive\_dest where status not like 'INA%';

修改归档目录状态为defer

Alter system set log\_archive\_dest\_state\_1=defer;

此时如果发生归档行为，则归档日志无法使用该目录。手工归档时可以归档成功，因为设置了log\_archive\_min\_succeed\_dest参数为2，并且设置了一个MANDATORY归档目录和2个optional归档目录

设置备用归档日志目录

1:设置一个归档终点log\_archive\_dest\_4

Altere system set log\_archive\_dest\_4=’location=f:\app\archive4\’；

2：给主归档目录终点增加alternate参数

Alter system set log\_archive\_dest\_1=’location=f\app\archive1\ mandatory alternamte=LOG\_ARCHIVE\_DEST\_4’;

3:将log\_archive\_dest\_4状态修改为alternate

Alter system set log\_archive\_dest\_4=alternate;

## 表空间与数据文件管理

ORACLE数据库应用系统🡨🡪ORACLE中间件🡨🡪DBMS🡨🡪数据文件

逻辑结构：一个数据库有多个表空间，一个表空间有多个段，一个段有多个区段，一个区段有多个数据块，一个数据块有多个操作系统数据库块

物理结构：一个表空间有多个数据库文件，一个数据文件物理上由操作系统块组成

查看表空间的数据库块大小

Select tablespace\_name,block\_size,contents from dba\_tablespaces;

数据文件物理结构

数据文件:oracle格式的操作系统文件，如dbf。数据文件大小决定表空间大小，当表空间不足，需要增加新的数据文件或者重新设置数据文件的大小

操作系统块：操作系统操作数据文件的最小单位。一个或多个构成数据库快

### 表空间分类以及创建表空间

数据库必备的表空间,【System,临时，还原，默认】表空间。Oracle11g还有sysaux表空间，它是system表空间的扩展，包含各种oracle产品和功能部件使用的数据。

表空间分为2类，

系统表空间：数据库系统创建时需要的表空间，

非系统表空间，还原表空间和临时表空间(默认系统表空间),数据表空间，索引表空间

语法

CREATE TABLESPACE tablespace\_name

[DATAFILE clause]:数据文件名字，给出完整目录和文件名，目录必须存在，否则无法创建

[MINIMUM EXTENT integer[K|M]]：定义该表空间最小区段的大小，这样该表空间中的区段大小为该最小值的整数倍

[BLOCKSIZE integer[K]]:指出该表空间使用的非标准块尺寸的大小，单位K。设置前需要设置DB\_CACHE\_SIZE 和 DB\_Nk\_CACHE\_SIZE，且要与BLOCKSIZE相同。使用默认的话，DB\_Nk\_CACHE\_SIZE=0；

[LOGGING[NOLOGGING]]：是否把该表空间中数据的变化记录在重做日志文件中，LOGGING记录，NOLOGGING不记录

[DEFAULT storage\_clause]

[ONLINE|OFFLINE]

[PERMANENT|TEMPORARY]:前者表示只能存储永久对象，后者说明为临时表，不需要将变化记录到重做日志文件，它只包含用户会话期间的数据，如排序中间结果。不使用参数默认永久表空间

创建表空间

Create tablespace user\_data Datafile 'G:\oracleDB\temp\userdata1.dbf' size 100M ;

查看表空间是否创建

Select tablespace\_name,logging,status from dba\_tablespaces;

查看表空间对应的数据文件

Select file\_name ,tablespace\_name,status from dba\_data\_files where tablespace\_name='USER\_DATA';

### 表空间磁盘管理的2种方案

数据字典管理，本地管理

管理表空间的实质是为用户分配可用的区段以及回收空闲区段的过程。

Oracle9I以上默认本地管理

#### 数据字典管理的表空间磁盘管理

将每个数据字典管理得表空间的使用情况记录在数据字典表中。当分配或者撤销表空间区段的分配时，则隐含使用sql语句对表操作以记录当前表空间区段的使用情况。并且在还原段中记录了当前变换前的区段使用情况。这种方式增加了数据字典的频繁操作。引入本地管理

#### 本地管理的表空间磁盘管理

让每一个表空间自己管理表空间区段的分配，记录区段的使用情况

使用位图在数据文件头中记录数据文件的可用和已用的空间信息。将表空间中数据文件的可用和已用信息记录起来。位图使用一个数据为表示一个数据库块或者一组数据库块的使用情况，而表空间分配的最小单位是EXTENT，而一个区段由多个数据库块组成。当需要在表空间中增加新对象时，需要查找位图，看能否有一段连续的oracle数据库块空闲。

局限:使用本地管理的表空间不能随意更改默认的存储参数。如初始区段的大小，最大区段数..

### 创建表空间

#### 创建数据字典管理的表空间

9i的第一个版本默认system表空间是字典管理的，9.2.0.1就默认是本地管理了。要改只能安装的时候选择

该表空间有3文件，分别放在不同目录下，是为了保证io平衡，每个数据文件大小为100M，最小区段为20K，

默认存储参数为，初始区段大小，和在次分配区段时，分配的区段大小为20k，所分配的最大磁盘空间为500区段。

CREATE TABLESPACE Tianjin\_data

Datafile

'G:\oracleDB\temp\01.dbf' size 100M,

'G:\oracleDB\temp\02.dbf' size 100M,

'G:\oracleDB\temp\03.dbf' size 100M

minimum extent 20k

extent management dictionary

default storage(initial 20k next 20k maxextents 500 pctincrease 0);

#### 创建本地管理的表空间

创建表空间

CREATE TABLESPACE BEIJING\_DATA

Datafile

'G:\oracleDB\temp\01.dbf' size 100M

EXTENT management local

Uniform size 1M; 设置统一的区段尺寸

验证表空间

Select tablespace\_name,block\_size,extent\_management,status,initial\_extent ,next\_extent,max\_extents,pct\_increase from dba\_tablespaces WHERE tablespace\_name='BEIJING\_DATA';

Select file\_name ,tablespace\_name,status from dba\_data\_files where tablespace\_name='BEIJING\_DATA';

#### 创建还原表空间

存放还原段，用户1修改了数据，其他用户查看此数据，此时显示的应该不是用户1更改的数据，因为用户1还未提交。为了保证读一致性，oarcle设计了还原段。在还原段中存放更改前的数据。

不能存放其他任何对象，创建还原表空间时，只能使用datafile,extent management子句。

创建还原表空间

CREATE undo tablespace user\_undo

Datafile 'G:\oracleDB\temp\user\_undo.dbf'

size 30M;

SELECT tablespace\_name,status,contents,logging,extent\_management from dba\_tablespaces;

Contents为undo表示为还原表空间，logging说明该表空间的变化受重做日志的保护，区段管理为本地

#### 创建临时表空间

用于用户特定会话状态，如用户会话中的排序操作，排序中间结果需要存储在某个区域，这个区域就是临时表空间，临时表空间的排序段是在实例启动后有第一个排序操作时间时创建的。没有创建临时表空间，则默认system表空间。在oracle9I中会自动创建temp的临时表空间，oracle11g临时表空间会默认创建，名字为TEMP，文件名为$ORACLE\_HOME/ORADATA/TEMP01.DBF

供多个用户使用，临时表空间中的区段在需要时按照创建临时表空间时的参数或管理方式进行扩展

创建临时表空间

Create temporary tablespace user\_temp

Tempfile 'G:\oracleDB\temp\user\_temp.dbf' size 20m

extent management local uniform size 1m;

select tablespace\_name,status,contents,logging from dba\_tablespaces where tablespace\_name like

CONTENTS为TEMPORARY，contents为 NOLOGGING，即不需要将临时表空间的变化记录到重做日志文件中。

在初始化参数文件中，有sort\_area\_size参数，表示排序区尺寸大小。为了优化临时表空间中排序操作的性能，最好将Uniform size设置为ssort\_area\_size整数倍

#### 默认临时表空间

可以在创建数据库时创建，此时使用DEFAULT TEMPORARY TABLESPACE,

也可以在创建数据库之后创建，此时需要建立临时表空间，在使用

Alter database default temporary tablespace指令更改临时表

如果在数据库创建时，没有建立表空间，数据库创建后也没有创建临时表空间且更改默认设置。Oarcle将system表空间设置为默认临时表空间，并且给出告警在alter.log文件中。

查看当前数据库默认的临时表空间

Select \* from database\_properties where property\_name like 'DEFAULT';

切换临时表空间，当设置临时表空间为USER\_TEMP时出现临时表空间占用，不满足条件，使用下面即可

Alter database default temporary tablespace user\_temp;

可以发现当前数据库的临时表空间为USER\_TEMP。

select \* from database\_properties where property\_name like 'DEFAULT%';

删除当前使用的临时表空间。不能删除默认的临时表空间

drop tablespace user\_temp;

可以创建另外一个新的临时表空间，切换到新建立的表空间，然后删除

默认临时表空间的类型不能改为PERMANENT永久表空间，不能置为脱机状态

将默认临时表空间脱机

Alter tablespace user\_temp offline ;

#### 创建大文件表空间

大文件表空间(big File tablespace)简称BFT，不要求段空间管理的类型可以使用，如回滚表空间和临时表空间

优势，大大减少数据文件数量，容量比普通表空间大，一个普通文件的表空间最多1024个数据文件，而大文件表空间只包含一个文件，但是容量可以达到1024倍数据文件的容量。

创建大文件表空间3种方式

创建数据库时，定义大文件表空间并把它作为默认表空间,

一旦创建默认表空间为大文件表空间类型，以后创建的表空间都为大文件表空间。

Create database set default bigfile tablespace tbs\_name datafile ‘d:\temp\tbs01.dbf’

Size 2G;

使用create tablespace bigfile创建大文件表空间

Create bigfile tablespace bigfiletbs datafile 'G:\oracleDB\temp\temp1.dbf' size 2g;

查询表空间的数据文件信息

Select tablespace\_name ,file\_name,bytes/(1024\*1024\*1024) G from Dba\_data\_files;

查询表空间的区段管理方式和段空间管理方式

Select tablespace\_name ,initial\_extent,contents,extent\_management,segment\_s

pace\_management from dba\_tablespaces where tablespace\_name like 'BIG%';

初始字段(initial\_extent)：大小为64KB，区段管理方式(extent\_management)为local本地管理方式，段空间管理方式(EXTENT\_MANAGEMENT)为自动管理(AUTO)

通过改变默认表空间为大文件表空间，使得后面创建的表空间都为大文件表空间。

Alter tablespace set default bigfile tablespace;

2种方式实现大文件表空间容量的修改

1:直接更改

Alter tablespace 表空间名 resize 4G;

2:修改表空间为自动扩展

Alter tablespace表空间名autoextend on next 1G;

Select tablespace\_name ,file\_name,bytes ,autoextensible from dba\_data\_files;

Autoextensible:表示是否为自动扩展

查看当前数据库的默认表空间的类型

Select \* from database\_properties where property\_name=’DEFAULT\_TBS\_TYPE’;

SMALLFILE小文件表空间类型,也就是普通表空间

### 表空间管理

#### 脱机管理，脱机管理的表空间无法实现数据访问，

有些情况需要将表处于脱机状态，情况如下

1：允许用户访问数据库的一部分，而某些空间不允许用户访问

2：执行脱机的表空间备份

3：在数据库打开时，恢复表空间或表空间中的数据文件

4：在数据库打开时，移动表空间中数据文件

设置为脱机状态

Alter tablespace USER\_UNDO offline;

查看是否脱机，status

Select status,contents,logging from dba\_tablespaces where tablespace\_name = 'USER\_UNDO';

查看表空间中的数据状态

Select file#,name,status from v$datafile ;

此时无法读取表信息.

#### 只读管理

只读的表空间不被重做日志保护，减少重做日志的大小。

Alter tablespace bigfiletbs read only;

查看表空间状态

Select tablespace\_name,status from dba\_tablespaces;

设置为正常状态

Alter tablespace bigfiletbs read write;

### 表空间和数据文件管理

#### 修改表空间大小

修改表空间4种方法

* 创建表空间时，使用autoextend on子句使得表空间在需要时可以自动扩展
* 创建表空间后使用alter database datafile file\_name autoextend on 修改。
* 表空间增加数据文件
* 修改数据文件大小，即重新设置表空间某个数据文件的大小
* 创建数据文件自动扩展的表空间,每次扩展1m

Create tablespace manager\_tbs1 datafile 'G:\oracleDB\temp\TBS1.DBF' SIZE 100

Uniform size 1M

M AUTOEXTEND ON;

查看表空间的数据扩展方式

SELECT FILE\_NAME,TABLESPACE\_NAME,BLOCKS,STATUS,AUTOEXTENSIBLE FROM DBA\_DATA\_FILES WHERE TABLESPACE\_NAME LIKE 'MAN%';

* 使用alter database命令修改表空间中的数据文件为自动扩展

Create tablespace manage\_tbs datafile

查看表空间的是否为自动扩展

select tablespace\_name ,file\_name ,autoextensible from dba\_data\_files;

alter database datafile

使用alter database 修改表空间数据文件为自动扩展,当该文件空间不足时可以自动扩展，每次扩展空间为1m;

alter database datafile 'G:\oracleDB\temp\TEMP1.DBF' autoextend on next 1m;

* 向表空间增加数据文件以达到间接扩展表空间大小

Alter tablespace manager\_tbs1 add datafile 'G:\oracleDB\temp\tbs02.DBF' size 50M;

查看表空间中的数据文件信息

Select tablespace\_name,file\_name,status,autoextensible from dba\_data\_files

where tablespace\_name='MANAGER\_TBS1';

#### 修改表空间的存储参数

9i支持，11g不支持

只对数据字典管理的表空间有效

修改表空间最小extent尺寸

Alter tablespace Beijin\_data minimum extent 2m;

修改表空间默认存储子句

Alter tablespace BEIJIN\_DATA default storage(initial 2m next 2m maxextents 50);

查看修改后的表空间的存储参数

Select \* from dba\_tablespaces where tablespace\_name=’BEIJIN\_DATA’;

#### 删除表空间

Drop tablespace tablespace\_name

[INCLUDING CONTENTS[ADD DATAFILES][CASCADE CONSTRAINTS]]

INCLUDING CONTENTS:删除表空间中的所有区段(EXTENTS)

ADD DATAFILES:删除表空间中的数据文件，文件格式为oracle格式

CASCADE CONSTRAINTS:删除和该表空间中的表相关的引用完整性约束。

删除表空间并且删除该表空间下的数据文件，不添加数据文件，默认删除该表文件指针，实际数据文件不会被删除

Drop tablespace MANAGER\_TBS1 including contents and datafiles;

#### 迁移数据文件

把当前表空间中的数据文件迁移到其他磁盘空间。迁移数据文件分为2种

* 迁移系统表空间中的数据文件

1:查看系统表数据信息

Select tablespace\_name,file\_name,bytes,status from dba\_data\_files where tablespace\_name= 'SYSTEM';

2:根据系统表路径复制一份到其他地方

Host copy sysytem表路径 新路径

3:打开到mount，然后迁移

Startup mount;

Alter database rename ‘system表路径’ to ‘新路径‘ ;

Recover database;--介质回复(oracle11G需要) Alter database open;

4:验证

Select tablespace\_name,file\_name,bytes,status from dba\_data\_files where tablespace\_name='SYSTEM';

* 迁移非系统表

要求没有活跃的还原段，临时段，排序段等，才可以迁移

1:设置表空间脱机

Alter tablespace USER\_UNDO offline;

2:复制数据文件到新磁盘下

host copy G:\ORACLEDB\TEMP\USER\_UNDO.DBF G:\ORACLEDB\TEMP\USER\_T.DBF

3:迁移数据文件

Alter tablespace USER\_UNDO rename datafile 'G:\ORACLEDB\TEMP\USER\_UNDO.DBF' to 'G:\ORACLEDB\TEMP\USER\_T.DBF';

4:联机

Alter tablespace USER\_UNDO online;

5:查看

Select tablespace\_name,file\_name,bytes,status from dba\_data\_files where ta

blespace\_name='USER\_UNDO';

#### 数据字典和本地管理的表空间

ORACLE9I中切换为本地管理

EXECUTE DBMS\_SPACE\_ADMIN.TABLESPACE\_MIGRATE\_TO\_LOCAL(tbs\_name);

切换到数据字典管理

EXECUTE DBMS\_SPACE\_ADMIN.TABLESPACE\_MIGRATE\_FROM\_LOCAL(tbs\_name)

## UNDO表空间管理

还原数据是为了实现数据更改的同时，其他用户或进程可以并发访问正在更新而没有提交的数据。

### 引入还原段的作用

解决修改数据时并行读数据的问题，即事务恢复，事务回滚，读一致性

当用户修改数据时，该数据首先复制到还原段上，一个事务将他需要修改的全部数据放在同一个还原段中。

事务恢复

读一致性

还原段

事务回滚

事务恢复：将还原段上数据的变化记录在重做日志文件中。事务执行期间数据库实例崩溃，重启还原没提交的数据。

事务回滚：用户使用rollback，oracle服务器数据库就使用还原段中的数据完成数据的回滚

读一致性：用户修改数据，未提交，其他用户访问读取时，看到没被修改提交的数据，这些数据放在还原段中

### 还原段如何完成读一致性

表中有1W条数据，9点读取，9点15结束。按理说应该有1W条数据，可是中途有用户删除了一条数据，查询的是9999则为脏读。1W条才是正常。而还原段就是为了保证读取的是1W条。此为读一致性。

用户在9点发出select时，服务器会记录9点那个时候的SCN号(SCN号是以时间戳作为参数的一个函数返回值。Scn可以和时间戳相互转换),该SCN号应该大于T表中等于所有记录在数据块头部的ITL槽中的SCN号(多个ITL，SCN为最大那个)。服务器进程扫描T表，会把头部的ITL中的SCN和当前SCN比较，如果比他小，说明没修改，比他大说明修改，为了保证读一致性，会去根据ITL槽中记录的UNDO块的地址。进行读取。

若ITL槽中的SCN>9，接着找ITL槽中的UNDO地址，接着在找。直到找到比9点小的SCN，读取。没有找到ORA-1555 snapshoot too old

事务提交以后，undo回滚段就可以被覆盖，出现1555不可避免。

### 还原段的实例恢复与实务回滚

实例恢复时，oracle会读取回滚段的头部记录的事务表，每一个事务是否提交等信息都存在里面，对于未提交的事务完成回滚

回滚时(错误或者rollback都会产生)，根据ITL槽中记录的undo数据块的地址找到undo数据块，从而实现恢复数据，即回滚事务。

### Undo segment选择算法

将每一个undo segment绑定一个事务，

没空闲的undo segment，系统尝试将脱机的undo segment联机

没有可用的，创建新的，不能创建新的，寻找最早使用的undo segment。这种情况下

一个undo segment可能绑定多个事务同时进行。每隔12小时会收缩一次，删除idle状态的extent。

Dml操作需要undo时，发现空间不够，会唤醒SMON进行一次收缩将undo segment里面暂时没有使用的extent使用

### 讨论undo\_retention参数

时间值：说明当还原段中的数据在事务提交后继续保留的时间。默认900秒。超过时间该undo块可以被其他事务覆盖.

使用aum，设置undo\_retention后，undo存在4状态

Active：正在使用事务，还没有提交|回滚

Inactive:没有活动事务，可以被其他事务覆盖

Expired：inactive时间超过undo\_retention指定时间

Freed：数据块是空的，从未被使用

Undo表存在多个undo segment，undosegment存在多个extents。

Oracle11g中，undo表空间足够，oracle会将undo信息保留的时间与当前运行时间最长的查询所需要的时间相同。

默认情况下，oracle每隔30秒收集统计信息来自动调整undo retention。收集信息包括运行时间最长的查询和产生undo的速度。设置undo retention为0，使用自动调整功能。并且900秒最低限。不为0则不支持自动调整。

设置undo\_retention的值

Alter system set undo\_retention=1200;

校验值

Show parameter undo;

方式2

Select name,value from v$parameter where name='undo\_retention';

Undo表空间数据文件可以设置自动扩展。

### 还原段分类

分为2类，系统还原段和非系统还原段。

系统还原段：系统表空间使用，当系统表空间中的对象发生变化，这些对象的原始值保存在此。可工作自动模式和手动模式。

非系统还原段：非非系统表空间使用。如用户表空间。

当一个数据库具有非系统表空间时，至少要一个非系统还原段|自动管理的还原表空间。

自动管理模式由数据库服务器自动维护，但至少需要一个还原表空间。

手动管理模式需要管理员创建非系统还原段。

手动的非系统还原段包括2类型，公有还原段(RAC多实例)和私有还原段。

Oracle9i以上都实现还原度胺的自动管理

使用自动管理需要首先创建一个还原表空间。并经还原表空间告诉数据库服务器，之后的维护工作由数据库服务器自动完成。

### ORACLE自动还原段管理

设置还原段自动管理，设置2参数

1：undo\_management，说明还原段的管理方式。不是动态参数，需要在参数文件中修改，然后重启服务器生效

2：undo\_tablespace：说明还原表空间名字，动态参数

查看和还原段相关参数,也可以在init.ora配置文件看

Show parameter undo;

AUTO自动管理，还原表空间UNDOTBS1；

修改还原表空间

Alter database set undo\_tablespace undo\_tablespace\_name;

### 创建还原表空间

创建还原表空间,自动扩展

Create undo tablespace my1\_undo

Datafile

'G:\oracleDB\temp\my1\_undo.dbf'

size 100m

autoextend on;

查看还原表空间信息

select tablespace\_name,extent\_management,contents,logging,status from

dba\_tablespaces where contents='UNDO';

本地管理。LOGGING说明受到重做日志保护。在线，说明随时可以切换指令到此还原表空间。

查看还原表空间数据信息

Col tablespace\_name for a15

Col file\_name for a20

Select tablespace\_name,file\_name,bytes/1024/1024 MB,autoextensible from dba\_data\_files Where tablespace\_name= 'MY\_UNDO';

### 维护还原表空间

重命名还原表空间

Alter tablespace my\_undo rename to lin\_undo;

向还原表空间增加数据文件，达到增加还原表空间的容量

Alter tablespace lin\_undo add datafile 'G:\oracleDB\temp\MY\_UNDO1.DBF'

size 100m;

查看添加成员信息

Select tablespace\_name,file\_name,bytes/1024/1024 MB,autoextensible from

dba\_data\_files Where tablespace\_name='LIN\_UNDO';

设置数据文件为自动扩展

Alter database datafile 'G:\ORACLEDB\TEMP\MY\_UNDO1.DBF' autoextend on;

### 切换还原表空间

Oracle只允许有一个活跃的还原表空间

切换还原表空间

Alter system set undo\_tablespace='LIN\_UNDO';

验证是否成功undo\_tablespace为LIN\_UNDO说明成功

Show parameter undo;

### 删除undo表空间实例

1：创建undo

Create undo tablespace my2\_undo

Datafile

'G:\oracleDB\temp\my2\_undo.dbf'

size 100m

autoextend on

retention guarantee;保证给定时间内还原记录不会飞覆盖。

2：创建表，并插入数据，模拟事务

Create table t(id number,name varchar2(20));

Insert into t values(1,'hello');

3:切换undo表空间

Alter system set undo\_tablespace= my2\_undo;

确认修改成功

Show parameter undo;

4:删除旧的表空间undotbs，此时报错，因为有活动事务。

Drop tablespace LIN\_UNDO ;= Drop tablespace LIN\_UNDO including contents;

提交后

Commit;

继续删除表，失败，因为该undo 的参数undo\_retention,在该参数范围内。Oracle依然会保留就得undo记录。所以不允许删除

设置retention guarantee属性，保证足够的时间内undo信息不会被覆盖掉

Alter tablespace undotbs1 retention guarantee;

### Dba\_undo\_extents数据字典

插入数据后，查看回滚段的名字

Select a.username,b.name,c.used\_ublk from v$session a ,v$rollname b ,v$transaction c

Where a.saddr=c.ses\_addr and b.usn=c.xidusn;

使用回滚字段名为:\_SYSSMU28\_3237543520$

查看该回滚段所在的表空间以及区段信息

Select segment\_name,tablespace\_name,extent\_id from dba\_undo\_extents where segment\_name='\_SYSSMU28\_3237543520$';

## 事务

DBMS使用事务协调用户的并发行为，减少用户访问资源的冲突

### 提交4方式

Commit显示提交

Ddl语句隐式提交。

正常结束程序，oracle数据库应用程序正常结束，如sqlplus更改数据正常退出，提交

Set autocommit on/off;执行DML操作，update，delete，insert自动提交

### 回滚2方式

Rollback回滚

非正常结束，回滚,隐式回滚

### 事务4特性(ACID)

原子性:事务要么执行成功，要么什么也不执行。如果执行了一部分出现系统崩溃或者异常，则回滚。此时oracle使用还原段管理更改数据的原始值用户事务回滚

一致性：数据库必须保证。如用户1删除了一条部门，而用户表中有绑定了要删除的部门的职员。就拒绝这样请求

隔离性：隔离性使得多个用户隔离执行实现数据库的并发访问，这样的隔离性要求一个事务修改的数据在未提交前，其他事务看不到他所做的更改。Oracle使用并发控制机制实现事务的隔离性。

持久性：保证提交的事务永久保存数据库中。Oracle中并不是立即写入数据文件中，而是先保存到数据库高速缓存中，为了防止实例崩溃，使用日志优先的方法，首先将提交的数据更改到重做日志文件中。即使实例崩溃也可以在实例恢复。保证事务持久性。

### 事务控制

用户控制事务的行为，有显示和隐式

Commit提交前，数据库内部发生

1:非系统还原段中生成要更改的数据的备份

2：重做日志缓冲区创建重做日志选项

3：数据库高速缓冲区修改数据(删除|更新)

Commit提交后

1. 重做记录的事务表中标记已提交事务的SCN，说明该事务以提交
2. LGWR将事务的重做日志信息和已提交事务的SCN号写入重做日志文件，此时认为提交完成
3. 释放oracle持有对象的对更改的数据对象的锁，标记事务完成。

## 角色管理

角色可以理解为权限的集合。

给10个用户授予10个相同的权限，需要100次操作，使用角色授予10个权限，然后角色授予10个用户，只需要11次操作。减少数据库操作，以及提高DBA管理数据库效率，减少出错机会

角色特点

Grant授权，revoke回收

可以授权任何用户或角色，但是不能赋予角色自己或循环授予

角色包含系统权限和对象权限

允许启动或关闭赋予用户的角色

允许使用密码启动一个角色

角色名唯一

不被任何人拥有，不属于任何模式

查看角色描述

Select \* from dba\_roles;

### 创建角色

CREATE ROLE role\_name [NOT IDENTIFIED|IDENTIFIED{ BY password |EXTERNALLY |GLOBALLY |USING package}]

role\_name:角色名

NOT IDENTIFIED:激活角色时不需要密码验证

IDENTIFIED:激活角色时需要密码验证

BY password:密码

EXTERNALLY：激活前，必须通过外部服务如操作系统或第三方服务授权

GLOBALLY:使用set role 激活角色时，用户必须通过企业路径服务授权来使用角色

USING package:创建应用角色，该角色只能由应用通过授权的package激活

创建角色，不需要任何口令

Create role mk\_clerk;

创建角色，需要激活时使用口令

Create role at\_clerk identified by rmb;

创建角色，使用外部服务来标识

Create role manager identified by externally;

查看角色信息

select \* from dba\_roles where role in('MK\_CLERK','AT\_CLERK','MANAGER');

### 修改角色

只能修改验证方法

ALTER ROLE role { NOT IDENTIFIED | IDENTIFIED {BY password |USING package |EXTERNALLY |GLOBALLY}}

验证方法设置为外部标识

Alter role mk\_clerk identified by externally;

不需要标识

Alter role at\_clerk not identified;

需要密码标识

Alter role manager identified by rmb;

### 赋予角色权限

GRANT 权限|角色 TO 角色名

授权

Grant create session,select any table,create view to at\_clerk

验证角色的权限信息

Select \* from role\_sys\_privs where role ='AT\_CLERK';

ADMIN\_：no表示该角色不能再将拥有的权限赋予给其他按用户|角色

将权限和角色授予角色

Grant create any table,at\_clerk to manager;

验证，无法查看授予角色权限

Select \* from role\_sys\_privs where role ='MANAGER';

查看角色授予给了那些角色, ADMIN\_表示不能授权角色给其他角色

Select \* from dba\_role\_privs where granted\_role='AT\_CLERK';

### 授予用户角色

GRANT role [, role] ……

TO {user | role | public } | [,{yser | role | public }]…..

[WITH ADMIN OPTION]

Role：赋予的角色名，多个角色逗号隔开

User：被赋予的用户，多个用户逗号隔开

Pubic：角色赋予给所有用户

WITH ADMIN OPTION:被授予该角色的用户或角色可以继续将该角色赋予其他用户或角色

创建用户

Create user clerk identified by my123456;

Create user a\_clerk identified by my123456;

尝试登陆，失败缺少创建session权限

connect clerk/my123456;

将角色赋予给用户,并with admin option可以继续将该角色赋予给其他用户

Grant at\_clerk to clerk with admin option;

验证授权用户信息

Select \* from dba\_role\_privs where granted\_role='AT\_CLERK';

可以看出，该用户具有继续将角色MANAGER继续授权的能力，因为ADM为yes，角色manager为clerk的默认角色，因为def为yes；

测试连接成功

connect clerk/my123456;

查看用户的会话级权限

select \* from session\_privs;

在clerk用户模式再将角色at\_clerk赋予给用户a\_clerk;

Grant at\_clerk to a\_clerk;

使用a\_clerk角色登录

connect a\_clerk/my123456;

查看当前用户角色信息

select \* from user\_role\_privs;

该操作不是操作系统验证，因为os\_为no。

查看权限

select \* from session\_privs;

### 默认角色

查看用户角色，default选项为yes，说明都为默认角色

Select \* from dba\_role\_privs where grantee='CLERK';

更改为非默认角色

alter user clerk default role all except manager;

将用户的所有角色设置为非默认

Alter user clerk default role none;

为用户设置默认角色

Alter user clerk default role at\_clerk;

将赋予的所有角色设置为默认角色

Alter user clerk default role all;

再将某个角色设置为默认时，如果该角色时通过其他角色授予该角色的，则角色不能再default子句中使用。

### 禁止和激活角色

查看用户权限

Select \* from session\_privs;

禁止所有角色

Set role none;

激活用户角色

Set role AT\_CLERK;

激活设置密码验证的角色

Set role manager identified by rmb;

### 回收和删除角色

查看角色赋予用户的信息

Select \* from dba\_role\_privs where granted\_role in ('AT\_CLERK','MANAGER');

因为角色是在system下授权的，所以该用户自动授予该角色

回收角色时可以同时回收多个，多个角色名逗号隔开。用户也同理。

Revoke at\_clerk from clerk;

删除角色

Drop role at\_clerk;

将角色赋予给所有用户,public表示所有用户

Grant manager to public;

查看赋予情况

Select \* from dba\_role\_privs where granted\_role in ('MANAGER');

回收授予的public的manager角色

revoke manager from public;

### ORACKE预定义的角色

* AQ\_ADMINISTRATOR\_ROLE:管理queue权限的角色
* Connect：连接数据库权限
* Dba：数据库管理员权限
* EXP\_PULL\_DATABASE：导出数据库
* IMP\_FULL\_DATABASE：导入数据库
* JAVADEBUGPRIV：调试java程序
* MGMT\_USER:创建会话和创建触发器权限
* OEM\_ADVISOR:执行OEM顾问权限
* OEM\_MONITOR：执行OEM监视权限
* OLAP\_USER:执行联机事务处理时的USER权限
* OLAP\_DBA：执行联机事务处理时的DBA权限
* RECOVERY\_CATALOG\_OWNER：恢复数据库字典
* RESOURCE:创建一系列数据库对象权限
* SCHEDULER\_ADMIN：管理各种调度的权限，如创建任务，执行程序等

查看预定义角色，以及权限

Select \* from role\_sys\_privs;

## 管理和维护表

### 数据存储类型

标量数据类型:其他

集合数据类型:VARRAY,TABLE

关系数据类型:REF

标量数据类型

Varchar2(size)|nvarchar2(size)：变长字符型数据1~1700大小。Nvarchar2支持全球化数据类型，支持定长和变长。每次使用需要不断计算存储的数据长度，再分配数据块

Char(SIZE)NCHAR(SIZE)定长字符型数据1~2000大小, NCHAR支持全球化数据类型，支持定长和变长

Date：日期，使用7个定长存储区存储日期型数据，月，年，日，世纪，时,分,秒

177712年1月1日~公元9999年12月31日

Number(P,S)：p总长度个数，S小数个数。P范围1~38，s范围-817~1217

CLOB|LONG：存储大数据对象，定长的字符串数据。如学术论文，个人简历。CLOB列不能直接使用ORACLE数据库指令，需要一个DBMS\_LOB的plsql软件包维护

BLOB|LONG RAW:存储无结构的大对象，如照片，PPT，二进制图像等。需要一个DBMS\_LOB的plsql软件包维护

BFILE：在操作系统文件中存储无结构的二进制对象。他是oracle的外部数据类型。Oracle无法直接维护这些数据类型。必须由操作系统维护

RAW:该数据类型使得数据库可以直接存储2进制数据。在计算机之间传输该类型数据库不做任何转换。所以存储和传输效率较高，最长2000字节

LONG/LONG RAW数据类型主要用在ORACLE8版本，LONG可以用8以上版本CLOB/BLOB代替

集合数据类型

VARRY数据类型：有数量限制

嵌套表:列值包含表，元素数量无限制，不能超过表空间的大小

关系数据类型

关系类型REF指向一个对象，如游标cursor

### 创建表

创建表的规则

1. 命令简单
2. 每个表一个表空间(较少维护数据字典的负担)，易于管理和维护。表空间是本地管理，对表空间的维护不影响其他的表
3. 使用标准extent尺寸减少表空间碎片
4. 表名长度最大为30字符

查看scott下的所有表

Select owner,table\_name,tablespace\_name from dba\_tables where owner='SCOTT';

创建表时，不指定用户名字直接写表名，默认当前用户创建的表。不指定表空间，使用默认表空间创建该表

创建表空间，然后在该表空间下创建一个表

Create tablespace lin datafile 'G:\oracleDataBase\TEMP\lin.dbf' size 30m

extent management local uniform size 1m;

create table scott.employees(ecode number(17)) storage (initial 100k next 1

00 k pctincrease 0 minextents 1 maxextents 8) tablespace lin;

Storage:initial针对该表，当表的数据量增加时，需要自动分配磁盘空间时第一次分配100K，第二次也是100K，所分配的最大磁盘为8个extents。最小为1个extents。Pctincrease是一个权值参数表示当第三次为该表增加磁盘空间时，需要按规矩计算。

Next\*(1+pctincrease/100)^(n-2);当n>=3时，如果第三次需要增加磁盘空间时

分配100\*(1+0/100)^(3-2)=100k。第四次需要增加时分配100\*（1+0/100）^(17-2)=100k。当pctincrease=0时，每次分配的磁盘空间和next参数相同。100K

查看表信息

Select owner,table\_name,tablespace\_name,initial\_extent,next\_extent from dba\_tables where owner='SCOTT' AND table\_name='EMPLOYEES';

### 创建临时表

只对当前用户的当前会话有效。当前会话只操作自己的数据，没有数据锁的争用。

临时表默认存储在系统的临时段中，如果临时表空间为空，无法创建成功。

创建临时表空间

Create temporary tablespace user\_temp

Tempfile 'G:\oracleDataBase\temps\user\_temp.dbf' size 20m

创建临时表

Create global temporary table scott.testTemp on commit preserve rows as se

lect \* from scott.emp;

查看表是否存在, TABLESPACE\_NAME为空说明临时表不存在默认表空间和默认临时表空间中，而是存储在临时段，临时段是一个磁盘区，当用户使用sql语句执行查询时如果需要对返回的数据进行排序时，oracle首先需要在内存中完成排序工作，如果内存容量不够就把计算的结果放在临时段中

Select owner,table\_name,tablespace\_name from dba\_tables where table\_name='TESTTEMP';

查看表是否为临时表

Select table\_name ,tablespace\_name,temporary from dba\_tables where owner='SCOTT' and table\_name ='TESTTEMP';

DROP TABLE scott.TESTTEMP;

### 段空间管理

段空间管理：是表段的空间管理，其他段数据都需要自己的段管理方式。段是oracle一个逻辑结构，对数据进行操作都会造成段空间的变化。

版本9i之后使用自动段空间管理(ASSM)，使用位图管理段空间的使用状况。表空间是assm，段空间则也是assm

### 高水位线(HWM)

表段使用高水位线来标记，表示使用过的数据块的边界

|1|2|3|4|5|…|1000|未用

此时1000个数据块就是这个表段 的高水位线所在

新建的表，高水位线在第一块。插入数据，HWM会随之移动。如果删除表中的数据，依旧保留HWM的高度(此时查询数据会扫描删除前的所有数据块，花费时间)。在进行truncate而不是delete时，会调整HWM到第一个位置

ASSM(Automatic segment space managerment,自动断空间管理)：维护一个低HWM，在低HWM之前的所有数据都是有数据的，可以直接读取。而低HWM🡪HWM之间的所有数据不能保证有。此时需要参考ASSM位图查看数据块使用情况

### 行迁移

读取一行数据，数据过于庞大，导致该行无法存储在创建这一行的数据块中，此时oracle会把该行离开原来的块，存储到另一个数据块中，使用rowid关联。

避免方式，设置合理数据库块大小等

这样方式访问增加了io，其次还有io相关的latch(缓存多个数据块)，访问速度下降。

### 创建索引组织表(IOT)

数据与索引存储在一起，按照索引的结构来组织和存储表中的数据。索引组织表的存储结构不是堆组织表(无序)。表中数据按照某个主键排序后存储，然后在以B树的组织结构存储在数据段中。

场景：

* 经常使用主键字段来实现查询的事务
* 完全由主键组成的表，如果采用堆组织表，因为所有的数据全部同样存储在索引中，所以表本身就是多余开销。
* 只会通过一个主键访问表。
* 数据以某种特定的顺序物理存储。

限制：主键不能被删除，延期，禁止。

优点：节约磁盘空间占用，降低IO，减少访问缓冲区缓存。

创建IOT表

CREATE TABLE tiot(x int primary key,y number,z varchar2(20)) organization index;

提取表定义元数据

SELECT DBMS\_METADATA.GET\_DDL('TABLE','TIOT') FROM DUAL;

**索引组织表属性**

* Organization index:表明为索引组织表
* Overflow：允许创建一个新段，如果IOT的行记录太大，则可以储存在这个新段上，什么时候使用这个新段，涉及2参数。PCTTHRESHOLD/including
* INCLUDING:行中从第一列直到INCLUDING所指定的列的所有列数据都储存在索引表上，其余列存储在溢出段上。
* PCTTHRESHOLD：如果行中的数据量超过数据块大小这个百分比，行中其余的数据放入溢出段上
* COMPRESS(键压缩):此子句可以进行键压缩以消除重复值，普通索引也可以。具体操作为在创建索引表后面添加organization index之后添加COMPRESS n子句。N表示指定压缩的列数，默认无穷大。

列Y的数据溢出到overflow参数指定的表空间USERS。违反PCTTHRESHOLD规则的行记录溢出到表空间USERS

CREATE table iot\_test

(x int ,

Y number,

Z varchar2(20)

Constraint iot\_test\_pk primary key(X)

)

Organization index

Pctthreshold 10

Including y

Overflow tablespace users;

### 表参数以及参数维护

**HEAP TABLE**:普通表，获取表中的数据是按照命中率来得到的，没有明确先后之分。

**IOT**:类似全是索引的表，表中的所有字段都放在索引上，所以等于是约定了数据存放的时候是按照严格规定来的，在数据插入以前就已经确定了其位置。不管插入先后顺序，他在物理上的位置与插入的先后顺序无关。查询时可以少访问很多block。插入比一般表慢。

**Intrans**：控制对数据块的并行操作的参数

事务槽：在数据块头中，存储事务的控制信息。每行数据有一个锁位，锁位跟事务槽号相同。数据库服务器就是通过每行的锁位找到数据库头中的事务槽，然后通过事务槽中的事务信息完成对该行数据的操作。每个事务只能使用一个事务槽。

Intrans定义创建数据块时事务槽的初始值，默认1。说明数据库服务器实现在一个数据块中可以有最多2个并行的事务。每个事务独立并行并且通过自己的事务槽实现对该数据的事务操作

**Maxtran**s：定义创建数据块时事务槽最大值。默认255。

查看2参数

select ini\_trans,max\_trans from dba\_tables where table\_name='EMPLOYEE';

pctfree:设置每个数据块中预留空间的百分比数。默认10%

数据块结构，从上到下3部分，1：数据块头，2：空闲区，3：数据区

数据块头从上往下增长，数据区从下往上增长。2者碰头则空闲区被占满。

一旦空闲区满，后续操作需要占用空间，数据块无法满足，就占用其他数据块的空间。这种空间置换会带来磁盘io效率低下。

Oracle针对此情况设置预留空间，满足在数据操作时对数据块空间的要求

如果修改数据行需要额外的空间，使用此参数指定预留空间大小

**FREELISTS**：空闲数据块队列的列表，当用户向表中插入数据时，需要数据块作为存储空间，那么该参数中的数据块就作为候选的数据块。

**PCTUSED**：定义数据块中已经使用的空间的百分比数，数据块已经使用的空间低于该参数值时，数据块才放入段中的空闲数据块列表。默认170%。

Select table\_name,tablespace\_name,pct\_free,pct\_used from dba\_tables where table\_name='EMPLOYEE';

修改2参数

Alter table employee pctused 50 pctfree 30;

## 索引

**全表扫描**：通过多块读操作读取，大大减少io次数和提高全表扫描速度，只有在全表扫描的时候才能使用多块读。

通过Explain plan解释执行计划

Explain plan for select \* from dual;

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

**通过行id(ROWID)读取**

采用单快读方式。

通过执行计划确定单块读

Explain plan for select \* from dept where rowed=’AAA’;

Select \* from table(dbms\_xplan.display);

**索引读取**

扫描分为2步

扫描索引得到相应的rowid，逻辑读

通过rowid从表中读取相应的数据，物理读

单块io读，因为索引小且经常使用，通常被cache。

因为涉及到逻辑读和物理读，所以性能较低

改进性能取决2参数

1：数据的选择性

2：表数据在数据块上的分布

选择性高，查询的结果数据越少，选择性地，则相反。

相关的行在表中的存储位置不靠拢，会减少索引的益处。也就是看数据块的分散程度

增加索引会导致Insert，delete性能下降，update性能提升。

通过执行计划，判断索引作用

Explain plan from select \* from emp where empno=10;

Select \* from table(dbms\_xplan.display);

### 索引扫描类型

#### 索引唯一扫描

通过唯一键，主键，oracle通常返回一个数据行

Explain plan for select \* from emp where empno=10;

Select \* from table(dbms\_xplan.display);

#### 索引范围扫描

如下情况oracle会使用范围扫描

* 唯一键上使用range操作符(>,<,>=,<=,between)
* 在组合索引上，只使用部分列进行查询，导致查询很多行
* 对非唯一索引列上进行查询

Explain plan for select \* from emp where empno>10;

Select \* from table(dbms\_xplan.display);

#### 索引全扫描

当使用索引全扫描时，查询出的数据必须全部从索引中得到

Explain plan for select \* from emp;

Select \* from table(dbms\_xplan.display);

#### 索引快速扫描

扫描索引块中的所有数据块，不进行数据排序，此种方式下，可以使用多块读功能，并行读功能，最大化数据的吞吐量

BE\_IX索引是一个多列索引

Explain plan for select name from emp;

Select \* from table(dbms\_xplan.display);

### 限制索引使用的情况

#### 使用<>,!=会导致索引失效

Explain plan for select \* from emp where empno<>10;

Select \* from table(dbms\_xplan.display);

#### 使用is null或者is not null

创建索引

Create index idx\_sal\_emp on emp(sal);

根据表名查询表中所有索引

Select column\_name from user\_ind\_columns where table\_name=’EMP’;

Explain plan for select empno,ename,deptno from emp where sal is not null;

Select \* from table(dbms\_xplan.display);

建议要是想用索引的话，对列添加not noll或者default方式。

#### 使用函数

不使用基于函数的索引，在where子句中对存在索引的列使用函数，会导致优化器忽略索引。常见的函数如trunc，substr，to\_date，to\_char

Explain plan for select \* from emp where trunc(hiredate)=’03-12月-81’；

Select \* from table(dbms\_xplan.display);

避免方案

1：使用基于函数的索引

2：避免索引列上使用sql函数

如：

Explain plan for select \* from emp where hiredate>’03-12月-81’ and hiredate <(todate(‘03-12月-81’)+0.99999);

#### 比较不匹配的数据类型

Explain plan for select name from user where name =123;

这里自动加函数to\_number(name)，对于不匹配的数据类型，oracle隐式转换。之所以不加到123上，是因为123是常量，常量不能改变。

解决方案

name =’123’;保证类型一致即可

### 集群因子

优化器决定是否使用索引取决2因素

* 选择性：选择性高意味着不同的值越多，使用索引返回的值越少
* 集群因子：是索引与他所基于的表相比得出的有序性度量，用于检查在索引访问之后执行的表查找的成本，使用索引访问数据之后的表查找成本计算依据。

集群因子含义：如果通过一个索引扫描一张表，需要访问表的数据块的数量

计算方法如下

1. 扫描一个索引
2. 比较某行的rowid和前一行的rowid，如果这2个rowid不属于同一个数据块，集群因子+1
3. 整个索引扫面完毕后，得到该索引的集群因子

如果集群因子接近表存储的块数，则说明这张表是按照索引字段的顺序存储的。如果集群因子接近行的数量，说明这张表不是按索引字段顺序存储的。

集群因子\*选择性参数=访问索引开销。

如果一个表大多数访问时按照某个索引做索引扫描，将该表的数据按照索引字段的顺序重新组织，可以提高该表的访问性能。

### 二元高度

索引查找2过程

1. 根据树定位，找出rowid(索引查找)
2. 根据rowid找出表中的数据行(表数据查找)

进行索引查找的时，首先从树根开始读数据，通过中间节点，最后定位到叶节点，整个过程只能进行单数据块的读取。

每定位一次就读取一个数据块，定位次数即是二元高度。

查看索引2元高度

Select blevel ,index\_name from user\_indexes where index\_name=’emp\_depa’;

影响2元高度因素：表中索引列的非null值/索引列中值的宽度

删除行，索引2元高度不影响。重建会降低2元高度。

### 直方图

作用：在分析表和索引时，记录数据分布。通过该信息，基于成本的优化器可以决定使用将返回少量行的索引，而避免使用基于限制条件返回许多行的索引。

使用范围：表的任何列上构建。通常索引列上构建。

原因：帮助优化器在表中数据严重倾斜时做出更好的规划。如果一个表中的列(通常索引列)数据发生严重的倾斜，在这个列上构建直方图，能够给予优化器更好的参考意见。

在sys用户下面的temp\_dept表的name列上建立直方图，这个直方图有10个桶

Exec dbms\_stats.gather\_table\_stats('SYS','TEMP\_DEPT',METHOD\_OPT=>'FOR COLUMNS SIZE 10 name');

将整个列上所有行数据分成10个存储桶。

默认直方图会产生75个存储桶，范围1~254。

### 创建索引

CREATE [UNIQUE|BITMAP] INDEX [schema.]index\_name on [schema.]table\_name

(column\_name[DESC|ASC][,column\_name[DESC|ASC]]…..)

[REVERSE]

[TABLESPACE tablespace\_name]

[PCTFREE n]

[INITRANS n]

[MAXTRANS n]

[instorage state]

[LOGGING|NOLOGGING]

[NOSORT]

* Unique：索引唯一
* Bitmap：创建位图索引
* Desc|asc:索引降序or升序
* Reverse：创建反向键索引
* Tablespace：指定表空间
* Pctfree：索引块中预先保留的空间比例
* Initrans：每一个索引块中分配的事务数
* Maxtrans：每个索引块中分配的最大事务数
* Instorage state:索引中区段extent如何分配
* Logging|nologging：说明”要记录|不记录”索引相关的操作。并保存在联机重做日志中。
* Nosrt：不需要在创建索引时在按键值进行排序

Create index emp\_ename\_idx on emp(ename);

使用数据字典查看索引信息

Select index\_name,index\_type,table\_name,tablespace\_name from user\_indexes;

查看索引所对应的表空间信息

Select a.index\_name "索引名" ,a.tablespace\_name "索引对应的表空间名" ,

b.file\_name "索引对应的磁盘文件" from dba\_indexes a, dba\_data\_files b where a.tablespace\_name=b.tablespace\_name ;

创建索引空间

Create tablespace index\_tbs datafile ‘路径’ size 100m aotoextend on;

创建表多列索引并指定索引空间

Create index emp\_ename\_sal\_idx on emp(ename,sal) tablespace index\_tbs;

查看多列索引的信息

Select index\_name,table\_name,tablespace\_name from user\_indexes where index\_name like ‘emp’;

查看索引是建立在表的哪几列,使用数据字典USER\_IND\_COLUMNS

### 查看索引

查询与索引列相关的信息,索引对应的表名，索引基于那些列创建

Select index\_name,table\_name ,column\_name from user\_ind\_columns where index\_name like ‘emp%’;

查看索引信息

Select index\_name,table\_name,table\_owner,dropped,tablespace\_name from user\_indexes where index\_name like ’表名%’;

Dropped:是否删除。

### B树索引

Oracle默认索引类型,可以单列也可以是组合索引(max 32col)。保存在索引列上有值的每个数据行的rowid。

oracle不会对索引列上包含null值的行进行索引，如果索引是一个组合索引，而其中列上包含null，这一行会包含索引列中

B树索引结构，根节点，分支节点，叶子节点。

叶子节点：存储实际的索引列的值和该列所对应的记录的行id(rowid)。Row是oracle唯一的指针，指向该行的物理位置。双向链表。

分支节点：包含其他分支节点。

### 位图索引

Oracle11g enterprise edition支持

使用位图标识被索引的列值

适用：没有大量更新任务的数据仓库，因为使用位图索引时，每个位图索引项与表中大量的行有关联。当表有大量数据更新操作时，位图索引相应的需要做大量修改。而且索引所占用的磁盘空间也会增加。并且索引在更新时受影响的索引需要锁定，所以位图索引不适合有大量更新操作的oltp系统。虽然可以通过重建索引类位图索引。

场合：列值的数量要求较少|中等(索引列基数较小)，就是表行记录少。

创建位图

Create bitmap index emp\_job\_bitmap\_idx on emp(job);

使用位图索引插入数据时注意：

1. 位图索引在批处理(单用户)操作中加载表(插入操作)方面比b树好
2. 当有多个会话同时向表中插入数据行时不应该用位图索引
3. 每条记录都增加新值时，b树索引比位图快3倍。

位图中因为对rowid进行压缩存放(一个rowid范围+位图)，所以每次锁定的都是整个rowid位图，因此对表的位图索引进行更新的时，并发性差容易死锁。

位图索引限制：

1. 基于规则的优化器不会考虑位图索引
2. Alter table语句并修改包含位图索引的列时，位图索引失效
3. 位图索引不包含任何列数据，不能用于任何类型的完整性检查，如主键，唯一键约束。
4. 不能被声明为唯一索引
5. 位图索引最大长度30

### HASH索引

Hash索引必须要使用hash cluster。

非常适合做数据仓库(相对静态值)；

在存储数据时，所有与这个集群键相关的行都存储在一个数据块上，如果数据都存储在同一个数据块上，并且将hash索引作为where子句的确切匹配条件，oracle就可以通过执行一个hash函数和一个io来访问数据。

Select \* from table\_name where col=？ 🡪散列函数🡪[hash键|集群键|数据]

查询过程变换为等价查询，最后匹配hash列和确切值，最后基于hash函数确定行的物理位置。

创建cluster表

Create cluster credit\_cluster(

Card\_no varchar2(16),

Transdate date sort

)hashkeys 100000 hash is ora\_hash(card\_no) size 2186;

Create cluster credit\_orders(

Card\_no varchar2(16),

Transdate date sort,

Amount number

)cluster credit\_cluster (Card\_no, Transdate);

Hash列为表1的card\_no，集群键列为表2的Card, Transdate

Hash索引可能是最快的访问数据库的方法，但是其本身也有缺点

* 集群键上不同值数目必须在创建hash集群之前先确定，需要在创建hash集群时候指定这个值，低估集群键不同值的数值可能会导致集群的冲突(2个集群键有相同的hash值)
* 一旦值设置过低，需要从重建cluster。
* Hash集群可能浪费空间，无法确定需要多少空间来维护某个集群键上所有的行，会造成空间浪费
* 不能为集群的未来增长分配好附加空间，hash集群不是最好的选择
* 应用程序经常进行全表扫描，hash集群不是最好的选择。由于需要为未来分配好集群的剩余的空间，全表扫描非常浪费资源。

### 反向键索引

作用：指在创建索引过程中对索引列创建的索引键值的字节反向

好处：将值连续插入到索引中时反向键能避免争用

场景：如果一个索引值是按照序列值递增，那么连续插入大量数据时，所有的记录都将插入B树索引结构中的最右侧的叶子节点，并且会写入同一叶子节点中。难以避免产生争用问题而影响索引性能。引入反向键索引使得每个键值被颠倒顺序。将序列性的键值分散开，使得键值平衡的保存在叶子节点中

键值颠倒如下

Larry🡪yrraL

2468🡪8642

创建反向键索引

Create index emp\_sal\_reverse\_idx on emp(sal) reverse;

通过数据字典视图查看反向键索引信息

Select index\_name,index\_type,table\_name from user\_indexes where index\_name like ’EMP%’;

### 基于函数的索引

在用户查询数据时，如果查询语句的where子句中有函数存在，oracle使用函数索引将加快查询速度。基于函数的索引使用表的列的函数值作为键值建立索引结构。

创建基于upper函数的函数索引

Create index dept\_dname\_idx on dept(UPPER(dname));

创建一个基于dept中列dname的函数索引，创建该索引时首先将该列dname中的值转换成大写，然后对大写的dname创建索引，放入索引表。

此时查询时，oracle就不必对where子句的条件做转换并逐条检索，对于选择的结果也不必使用upper函数在做转换的计算。显然此时使用基于函数的索引会极大提高查询速度。

Select upper(dname) from dept where upper(dname) =’NEW YORK’;

查看索引信息，column\_name为系统赋予的一个值，因为这个列不是oracle可以使用明确的列名标识的，他是某个列的函数。

SELECT INDEX\_NAME,TABLE\_NAME,COLUMN\_NAME FROM USER\_IND\_COLUMNS;

注意事项

* 基于函数的索引只针对一种函数，对于其余函数不起作用
* 控制函数的数量，因为会影响DML

### 监控索引的使用

创建索引需要监控索引使用情况，这样就能判断创建的索引的有效性。对于无效的索引可以删除，以释放磁盘空间并减少dml操作带来的修改索引的各种开销。

启动对索引的监控

Alter index emp\_ename\_bitmap\_idx monitoring usage;

等待用户对表emp的各种操作，对于OLTP(联机事务处理)系统这个周期可能很短，对数据仓库则需要更多时间来监控。

终止监控索引

Alter index emp\_ename\_bitmap\_idx nomonitoring usage;

查看索引使用情况

Select index\_name,table\_name,monitoring,used,start\_monitoring,end\_monitoring from v$object\_usage;

Mon:是否监控索引

Used：是否使用索引

Start/end\_monitoring:监控开始/结束时间

### 重建索引

在删除操作后，删除值所占用的索引空间不能被索引自动重新使用，而插入操作后会不断使得索引变大。

重建索引并迁移表空间，使得索引空间可以重用删除值所占用的空间，使索引更加紧凑,不影响用户使用索引，但是不能使用ddl操作和dml。

Alter index emp\_ename\_bitmap\_idx

rebuild //此处添加 online 的话说明联机重建索引，并且可以使用dml。

tablespace index\_tbs1

pctfree 30

storage (next 100k)

;

查看索引名，表名，表空间名，状态。Valid表示有效

Select index\_name,table\_name,tablespace\_name,status from user\_indexes;

### 维护索引

查看当前索引的参数设置

Select index\_name,pct\_free,pct\_increase,initial\_extent,next\_extent from user\_indexes;

修改pctfree, storage 参数

Alter index emp\_ename\_bitmap\_idx

pctfree 30 单位%。

storage (next 100k) next\_extents参数修改无效，因为索引存储的表空间是本地管理。

;

手工增加索引磁盘空间

Alter index emp\_job\_bitmap\_idx allocate extent;

Oracle对于每个索引默认的extent区段数为1，此时为他增加一个区段。

Select segment\_name,segment\_type,tablespace\_name,extents from user\_segments

Where segment\_type=’INDEX’ and segment\_name like ‘表名%’;

合并索引碎片,释放磁盘空间

Alter index emp\_job\_bitmap\_idx coalesce;

删除索引,并查看是否删除

Drop index 索引名;

Select \* from user\_indexes where index\_name like ‘索引名%’;

## 系统和对象权限管理

权限分类

* 系统权限：允许用户执行一个或一类特殊的数据库操作，如创建数据库，创建用户，创建于维护表空间以及管理会话等
* 对象权限：对象权限是用户维护数据库对象的权利，如维护表，视图，序列号，存储过程，函数等

### 系统权限

**与索引相关**

Create any index

Alter any index

Drop any index

Create table权限中包含了Create index权限

**与表相关**

* Create table
* Create any table
* Alter any table
* Drop any table
* Select any table
* Update any table
* Delte any table

创建表时必须为表分配表空间配额，或者对表空间具有无限制使用权利。

如：unlimited tablespace;

**会话相关**

Create session

Alter session

一个新用户创建后，需要授予session权限才能访问数据库

**与表空间相关**

* Create tablespace
* Alter tablespace
* Drop tablespace
* Unlimited tablespace:允许使用所有表空间的权限

### 授予用户权限

GRANT {system\_privilege|role}

[,{system\_privilege|role}]……

To {user|role|PUBLIC}

[,{user|role|PUBLIC}]…..

[WITH ADMIN OPTION]

创建用户并设置密码

Create user jane identified by abc123#;

赋予用户权限

Grant create session,create table,select any table to jane;

Select \* from dba\_sys\_privs where grantee=’JANE’;

回收用户权限

Revoke create session,select any table,create table from jane;

部分权限授予所有用户

Grant create session,select any table to public;

查看用户系统权限

Select \* from dba\_sys\_privs where frantee=’用户’;

### SYSDBA和SYSOPER系统权限

**SYSDBA权限**

SYSOPER PRIVILEGES WITH ADMIN OPTION：具有sysoper所具有的操作，并且可以将这些操作授予其他用户

* CREATE DATABASE :创建数据库
* ALTER DATABASE BEGIN/END BACKUP:将数据库置于备份状态
* RESTRICTED SESSION:设置会话限制
* RECOVER DATABASE UNTIL:介质恢复数据库到UNTIL指定的状态

**SYSOPER权限**

* STARUP:启动数据库
* SHUTDOWN:关闭数据库
* ALTER DATABASE OPEN |MOUNT:数据库切换到打开|挂起状态
* ALTER DATABASE BACKUP CONTROLFILE TO:备份控制文件
* RECOVER DATABASE:介质恢复数据库
* ALTER DATABASE ARCHIVELOG:数据库设置为归档模式

### 回收用户权限

REVOKE {system\_privilege|role}

[,{system\_privilege|role}]……

FROM {user |role|PUBLIC}

[,{user|role|PUBLIC}]……

查看用户系统权限

Select \* from dba\_sys\_privs where grantee in (‘JANE’,’LARRY’) ORDER BY grantee;

回收用户权限

Revoke create session from larry;

用户1授权给用户2，用户2授权给用户3，用户1回收用户2的权限后，用户3还依然有用户2之前的权限，说明权限不具备级联特性。

回收所有用户的系统权限

Revoke create session from PUBLIC;

### 授予对象权限

和系统权限相对应的是对象权限，对象包括表，视图(物化视图),序列号，存储过程等。

在这些数据库对象上实现某种特殊的行为的权限称为对象权限。如删除表

对象权限包括：alter,delete ,execute,index,insert,references,select,update

**数据库对象的权限与对应的数据库对象关系的列表**

表：具备出execute的所有权限

视图：具备除alter，execute，references所有权限

序列号：具备alter，select

过程:具备alter，execute

GRANT {object\_privilege[(column\_list)] }

[,{object \_privilege[(column\_list)]]……

| ALL [PRIVILEGE]

ON [schema.]object

To {user|role|PUBLIC}

[,{user|role|PUBLIC}]…..

[WITH GRANT OPTION]

* GRANT：授权关键字
* OBJECT\_PRIVILEGE:对象权限
* COLUMN\_LIST：对象权限操作的列的列表
* ALL：当前用户的某个数据库对象的所有权限赋予给用户
* ON OBJECT：说明具体的数据库对象，如表|存储过程
* WITH GRANT OPTION：新用户可以继续授权

将scott用户的表的update权限赋予给用户

Grant update on 表 to 用户 with grant option;

查看scott用户中表对象的授权信息

Select \* from user\_tab\_privs\_made;

把对表dept的列的update对象权限授予给用户

Grant update (列1,列2) on 表 to 用户;

查看用户对列的权限授予信息

Select \* from user\_col\_privs\_made;

### 回收对象权限

REVOKE {object\_privilege[,object\_privilege]….

|ALL[PRIVILEGE]}

ON [schema.] object

From {user|role|PUBLIC}

[,{user|role|PUBLIC}]…..

[CASCADE CONSTRAINTS]

回收用户对表的update权限

Revoke update on emp from larry;

验证是否回收

Select \* from user\_tab\_privs\_made where grantee=’LARRY’;

回收用户对dept表的所有权限，只能从整个表回收对象权限，不能按照列回收。此种方式间接回收了列的权限

Revoke all on dept from 用户名

Select \* from user\_col\_privs\_made where grantee=’LARRY’;

**对象权限回收是级联的。如用户1授权给用户2，用户2授权给用户3，用户1回收用户2的权限，间接的用户3也丧失权限。**

## 视图

CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view\_name [别名[,别名]…]

AS

[WITH CHECK OPTION [ CONSTRAINT 约束名]]

[WITH READ ONLY]

FORCE/NOFORCE：创建视图时，不论基表是否存在都创建该视图。NOFORCE相反。

AS:说明下面是查询子句。用户定义视图

WITH CHECK OPTION：当更新某一数据行时，必须满足where子句的条件

WITH READ ONLY：设置该视图只读。说明无法进行DML操作

创建视图二种方式

Create view accounting\_view as select \* from temp;

Create or replace view sales\_view (“col1”,”col2”) as select name,id from temp;

查询数据字典中记录的视图定义

Select view\_name from user\_views;

查询视图account\_view定义的语句

Select text from user\_views where view\_name=’ACCOUNTING\_VIEW’;

查询ACCOUNTING\_VIEW视图的所有信息

Select \* from accounting\_view;

### 使用视图的WITH子句

Create view emp\_view as select \* from emp where job in (‘SALARY’,’MANAGER’);

Where子句限制只选择表emp中的job为salary和manager的所有员工数据才可以插入，否则不允许插入。

### 视图的修改

可以通过replace替换修改的视图

### Oracle视图管理

视图查询的内部过程

* 读取数据字典，获得该视图的定义，查找到该视图所引用的表
* 从数据字典中查询当前用户对于该视图所引用的表的权限。
* 执行定义该视图的sql语句，实现执行视图查询。

### 视图DML操作的限制

#### 简单视图

简单视图从一个表读取数据，不包括函数和分组数据。简单视图可以进行DML操作，简单视图的dml操作直接转换成对定义它的表的dml操作

#### 复杂视图

从多个表提取数据，包括函数和分组数据，复杂视图不一定能进行DML操作。

ORACLE对复杂视图上进行DML操作加了限制：

* 如果复杂视图包含分组函数，group by|distinct关键字，就不能使用delete/UPDATE/INSERT。
* 如果复杂视图中的列包含表达式，或者有伪列ROWNUM，则不能使用复杂视图进行DML操作。

### 物化视图

普通视图会重新执行创建视图的表的sql语句，当表多时，相当耗时。引入物化视图

物化视图：具有物理存储的特殊视图，基于表，视图或者其他物化视图创建的。创建一个物化视图时，oracle会自动创建一个内部表来存放物化视图的数据。占用存储空间，可以进行分区和创建索引等操作。

#### 查询重写

对sql查询语句进行重写，当用户使用sql语句对基表进行查询时，如果已经创建了基于这些基表的物化视图，oracle将自动计算和使用物化视图来完成查询。此称为查询重写。

查看query\_rewrite\_enabled的值,是否允许查询重写功能。

Show parameter query\_rewrite\_enabled;

#### 物化视图的同步

物化视图基于基表创建，当基表变化时，需要同步数据以更新物化视图中的数据，保持他两数据一致。

**2种物化视图刷新方式**

* on commit:基表变化时，物化视图自动刷新。完成与基表的同步
* on demand：需要手动同步物化视图和基表数据，此时必须DBMS\_MVIEW.REFRESH过程来同步物化视图

选择一种刷新方式，可以选择1种刷新类型完成数据同步。

刷新类型：指刷新数据时如何实现基表与物化视图的同步，从而将基表的变化反映在物化视图中。

**4种刷新类型**

* COMPLETE：重新执行创建物化视图的sql查询语句，无论基表中修改的数据量多少，都需要完成一次对物化视图的重新计算
* FAST：使用每个基表的物化视图日志只同步变化的数据，节约查询时间成本
* FORCE：显示fast类型更新数据，如果失败在使用complete类型刷新整个物化视图
* NEVER：从不更新。对基表数据不变的物化视图有效。

不指定刷新类型，默认force。

注意:启动查询重写机制，oracle查询重写不一定实现。可以使用DBMS\_MVIEW程序包中的过程分析。

#### 创建物化视图

**1:授权**

授予创建物化视图的权限，query rewrite权限,对创建物化视图所涉及的表的访问权限和创建表的权限。

Grant create materialized view to scott;

GRANT QUERY REWRITE TO SCOTT;

GRANT CREATE ANY TABLE TO SCOTT;

GRANT SELECT ANY TABLE TO SCOTT;

**2：创建物化视图日志**

物化视图日志是用户选择刷新类型为fast时要使用的。以同步基表的变化。

Create materialized view log on dept;

**3:创建物化视图的语句**

CREATE materialized view mtrlview\_test

Build immediate

Refresh fast on commit

Enable query rewrite

As

Select \* from temp;

Build immediate :立即创建物化视图。Bulid deffered，使用该视图时创建

Refresh fast on commit：刷新的数据类型为fast，刷新类型要求使用物化视图日志实现与基表数据的同步。on commit说明基表数据提交后立即更新物化视图

Enable query rewrite：启动查询重写功能，

**4：删除物化视图**

Drop materialized view mtrlview\_test;

#### 物化视图使用场景

总结：物化视图是一种用于汇总，计算，复制以及发布数据的方案。

物化视图适用于，数据仓库，决策支持，分布式计算，以及移动计算等环境

* 数据仓库：在数据仓库中往往需要存储对于基表的汇总或平均数据等。物化视图用与进行类似的计算来存储聚合后的数据。因为在数据仓库中的物化视图通常汇总数据，所以数据仓库中的物化视图也称为概要
* 分布式环境：在分布式环境中可以通过物化视图实现不同节点间的数据同步，使得同样的数据分布在不同的物理空间，更好的响应用户的查询，减少中心数据库服务器的负担。物化视图的复制功能使得用户可以把数据复制到远程节点，而数据同步能力有可以同步各个节点间的数据。从而实现数据的本地访问。

用户🡪物化视图🡨(视图更新)🡪中心服务

* 移动计算环境:利用物化视图的优化查询功能来节约查询时间，同时移动客户端使用物化视图下载一个数据子集，也可以定期地从中央服务器获得新数据，在客户端修改后发到中央服务器。

## 序列号和同义词

### 序列号

主要完成在多用户环境下产生唯一的数字序列，但不会造成额外的磁盘io或事务锁。

在不使用序列号时，多个用户同时向表插入一条记录，用户必须等待以得到下一个可用的员工号，而一旦使用序列号则用户无需相互等待就可以得到下一个可用的员工号。序列生成器会自动为每个用户创建正确的员工编号。

创建序列号

Create sequence emp\_seq

Start with 1000

Increment by 1

Nomaxvalue

Nocycle;

使用数据字典查看序列号信息

Select sequence\_name,min\_value,increment\_by,cycle\_flag,cache\_size from user\_sequences

Where sequence\_name like 'EMP%';

**创建序列**

create sequence 序列名

start with 1//1开始

increment by 1 //步长

maxvalue 1|nomaxvalue //设置最大值|不设置最大值，升序时为10^27，降序-1

minvalue 1|nominvalue//设置最小值|不设置最小值，最小值为1，降序为-10^26

cycle|nocycle //假设1开始，到100结束。开启则重新由1在循环到100

cache|nocache 20 //缓存

当大量语句发生请求，申请序列时，为了避免序列在运用层实现序列而引起的性能瓶

Oracle序列允许将序列提前生成 cache x个先存入内存，在发生大量申请序列语句时，

可直接到运行最快的内存中去得到序列。但cache个数也不能设置太大，因为在数据库重启时，

会清空内存信息，预存在内存中的序列会丢失，当数据库再次启动后，

序列从上次内存中最大的序列号+1 开始存入cache x个

查看序列属性信息

Desc 序列名

修改序列属性值,不能修改start with，alter对之前的无影响，影响之后的序列号

Alter sequence 序列名

Increment by n

Maxvalue n|nomaxvalue

Minvalue n|nominvalue

Cache n|nocache

Cycle n|nocycle

### 同义词

同义词是数据库对象的别名，如表，视图，物化视图，触发器，序列号，java等。

方便用户访问属于其他用户的数据库对象，或出于安全目的，因为同义词名字具有随机性，没有限制，这样就隐藏创建同义词的原始对象的值。

同义词分为私有和公有，私有的只能指定用户使用，没有授权其他用户无法访问某一用户的私有同义词。公有的所有用户都可以访问

Select \* from user;表不存在

Select \* from scott.user;暴露了user表的用户信息，使用别名可以避免

#### 公有同义词

创建公有同义词的用户必须具有 create public synonym权利。进行操作时需要具有delete，update等权限。

创建公有同义词

Create public synonym department for scott.dept;

使用公有同义词访问

Select \* from department;

删除公有同义词

Drop public synonym department;

#### 私有同义词

Create synonym department for scott.dept;

删除私有同义词

Drop synonym department;

#### 切换用户模式

Alter session set current\_schema=scott;

## RMAN备份与恢复数据库

Rman在数据库服务器的帮助下实现数据文件，控制文件以及数据库文件和控制文件的映像副本，归档日志文件，数据库服务器参数文件的备份。RMAN也也允许使用脚本文件实现数据的备份与恢复，而且这写脚本存储在数据库内，文件的名称也有RMAN自己维护。

### RMAN独特之处

优点如下

* 支持增量备份：在传统的备份工具(EXP|EXPDP)中，只能实现一个完整备份而不能增量备份，RMAN采用备份级别实现增量备份，在一个完整备份的基础上，采用增量备份，和传统备份方式比，这样减少备份的数据量。
* 自动管理备份文件：RMAN备份的数据时RMAN自动管理的，包括文件名字，备份文件，备份存储目录，以及识别最近的备份文件，搜索恢复时需要的表空间，模式或数据文件等备份文件。
* 自动化备份与恢复：在备份和恢复操作时，使用简单的指令就可以实现备份与恢复。且执行过程完全由RMAN自己维护
* 不产生重做信息：与用户管理的联机备份不同，使用RMAN的联机备份不产生重做信息。
* 恢复目录：RMAN的自动化备份与恢复功能归功于恢复目录的使用，RMAN直接在其中保存了备份和恢复脚本
* 支持映像复制：映像是以操作系统上的文件格式存在，此种复制方式类似于用户管理的脱机备份方式。
* 新块的比较特性：这是RMAN支持增量备份的基础，此特性使得在备份时，跳过文件中从未使用过的数据块的备份。备份数据量的减少直接导致备份存储空间需求和备份时间的减少。
* 备份的数据文件压缩处理：RMAN提供一个参数，说明是否对备份文件进行2进制压缩。可以减少备份文件的存储空间
* 备份文件有效性检查功能：验证备份的文件是否可用，再恢复前需要验证备份文件的有效性。

### RMAN系统架构

Oracle的RMAN工具使用会话建立客户端到数据库服务器的连接。

用户🡪启动RMAN可执行程序🡪建立客户端与服务端的会话连接🡪通过RMAN的客户端进行RMAN操作🡪执行备份与恢复指令（在服务器端的服务器进程中执行，完成实际的磁盘读写操作）

**RMAN系统结构**

* RMAN可执行程序：客户端工具，启动与数据库服务器的连接，实现备份与恢复
* RMAN客户端：一旦建立与数据库服务器的会话连接，RMAN可执行程序就创建一个客户端，通过客户端完成与数据库服务器之间通信。客户端可以连接通过ORACLE Net连接到可访问的任何主机上
* 服务器进程：在RMAN建立与数据库服务器的会话连接后，在数据库服务器端启动一个后台进程，执行RMAN客户端发出的各种数据恢复与备份指令，并完成实际的磁盘或磁带设备的读写任务
* RMAN信息库：记录RMAN的信息，如备份的数据文件及副本的目录，归档的重做日志备份文件和副本，表空间和数据文件，以及备份或恢复的脚本和RMAN的配置信息。默认使用数据库服务器的控制文件记录此信息。
* 恢复目录：记录RMAN信息库的信息，但是恢复目录需要先配置，信息库可以存储在数据库控制文件or恢复目录中，在oracle默认将rman信息库写入控制文件，如果存在恢复目录则需要继续写到恢复目录。使用控制文件记录rman信息库的空间有限，空间不足时可能被覆盖掉。

发送到RMAN恢复目录的元数据是从控制文件同步来的

RMAN客户端🡪服务器进程(操作DB)🡪RMAN信息库(可以保存到控制文件或控制目录)

### 快闪恢复区(flash recovery area)

是存储与备份和恢复数据文件以及相关信息的存储区。快闪恢复区保存了每个数据文件的备份，增量备份，控制文件备份以及归档重做日志备份，Oracle也允许在快闪恢复区中保存联机重做日志的冗余副本以及当前控制文件的冗余副本，闪回特性中的闪回日志也保存在此。

使用RMAN实现数据库的备份与恢复时，配置的快闪恢复区就是RMAN存储所有与备份相关的文件存储区。自动管理

优点：实现备份文件的自动管理，使得备份与恢复数据库更简洁，并且可以集中管理磁盘空间。

修改快闪恢复区大小

3方式，init.ora/指令动态修改/OEM工具

查看恢复区参数，包括目录和恢复区的空间大小。

Show parameter db\_recovery\_file\_dest;

修改大小，不需要快闪恢复区，将db\_recorery\_file\_dest设置空格。

Alter system set db\_recovery\_file\_dest\_size=2g;

查看空间使用情况以及文件数量

Select name,space\_limit,space\_used,number\_of\_files from v$recovery\_file\_dest;

#### 解决快闪恢复区空间不足问题

1. 增加恢复区磁盘空间，但是受磁盘空间的限制

Alter system set db\_recovery\_file\_dest\_size=4g;

1. 删除没用的备份文件或将备份文件复制到磁带设备

使用crosscheck、delete obsolete删除不需要的文件。或者使用expired指令删除不需要的备份文件，或者使用rman的backup recovery area指令将快速恢复区中的文件复制到磁带中。

1. 删除当前的恢复区，重新设置

Alter system set db\_recovery\_file\_dest=’目录’;

当向恢复区添加新文件时，oracle自动更新文件列表。删除符合条件的备份文件，复制到磁带的过渡文件，不符合保留策略的文件。而重做日志文件和控制文件任何时候不会被删除。

查询闪回恢复区的空间使用情况

Select file\_type,percent\_space\_used,percent\_space\_reclaimable,number\_of\_files from v$flash\_recovery\_area\_usage;

### 建立RMAN到数据库的连接

创建用户，并授予dba权限。只有dba权限才能操作RMAN。

create user rman identified by oracle;

grant resource,connect,dba to rman;

连接rman，使用rman用户连接，也可以使用操作系统认证连接

$rman target rman/oracle

$rman target /

### RMAN的相关概念与配置参数

**概念**

* 备份集：一个逻辑数据集合，由一个或多个RMAN的备份片组成。

备份片是RMAN格式的操作系统文件，包括数据文件，控制文件或者归档日志文件。

默认情况下执行RMAN的备份时，将产生备份文件的备份集，备份集只有RMAN可以识别，所以在恢复的时候必须使用RMAN来访问备份集来恢复。

备份集是RMAN默认备份文件类型，备份集是备份片的逻辑结构。

一般一个通道产生一个备份集，控制文件的备份集以操作系统块作为最小单位，数据文件备份集以数据库块作为最小单位，所以他们不能放在同一个备份集中。

* 通道：RMAN是通过与数据库服务器的会话建立连接，通道代表这个连接。他指定了备份或回复数据库的备份集所在的设备，如磁盘。通道数量越多，恢复or备份任务执行越短。
* 映像复制：是数据库文件的操作系统文件的一个备份。类似操作系统copy指令备份文件一样。一个数据文件生成一个映像文件副本，整个复制过程是RMAN进行数据块的复制过程，RMAN会检测每一个数据块是否出现损坏，不需要将表空间设置为begin backup，镜像副本中包括使用过的数据块，从未使用过的数据块。

好处在于：恢复速度相对备份集快

默认创建备份集。设置备份类型为copy

Configure device type disk backup type to copy;

映像复制整个数据库/单个表空间/整个数据库中的一个数据文件,3表示数据文件id，它与数据字典DBA\_DATA\_FILES中的file\_id一致。

Backup as copy database/tablespace users/datafile 3;

**查看当前所有的RMAN参数**

Show all;

* CONFIGURE RETENTION POLICY TO REDUNDANCY 1:保留备份的副本数量，如果每天都备份一个数据文件，1说明只保留一个该数据文件的副本，并且保留最新备份副本
* CONFIGURE DEFAULT DEVICE TYPE TO DISK：说明备份的数据文件默认备份到数据库服务器的磁盘上，也可以更改备份到磁带上

更改RMAN的备份设置类型为磁带/磁盘

CONFIGURE DEFAULT device type to sbt/disk;

* CONFIGURE BACKUP OPTIMIZATION OFF/ON:配置备份优化，模式不使用备份优化，使用备份优化的所用是如果已经备份了某文件的相同版本，则不会再备份该文件。即只保留一份备份文件。On开启
* CONFIGURE CONTROLFILE AUTOBACKUP OFF/ON：配置模式下不启动控制文件的自动备份
* CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET：rman在备份和恢复中的并行度，BACKUPSET备份集

修改并行数以及备份文件类型

CONFIGURE DEVICE TYPE DISK PARALLELISM 3 BACKUP TYPE TO COPY;

### RMAN备份控制文件

没有启动快闪恢复区：使用FORMAT参数指定控制文件的备份目录

启动的话：自动将控制文件复制到快闪恢复区的备份集中(BACKUPSET目录下)

没有启动快闪恢复区时备份文件

Backup current controlfile format;

启动快闪恢复区备份控制文件

Backup current controlfile;

**备份例子中出现的%u，作用是产生唯一的备份文件名。**

配置控制文件备份的磁盘类型和备份目录

CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO ‘/a/backup/%F’;

配置控制文件自动备份,**数据库结构发生变化，自动更新备份文件,使用BACKUP备份数据库时也会备份**

CONFIGURE CONTROLFILE AUTOBACKUP ON;

查看目录/a/backup/%F下是否有备份的控制文件

! ls –al /a/backup

**备份文件名格式:c-dbid-控制文件备份日期-序列号。另外备份数量每天最多256**

### RMAN实现脱机备份

**需要设置为归档模式等**

1：启动RMAN登录数据库服务器

2：启动数据库到mount状态，backup database指令备份整个数据库。

查看设备类型是磁带还是磁盘

$Backup as compressed backupset database;

配置控制文件和spfile文件自动备份

CONFIGURE CONTROLFILE AUTOBACKUP ON；

打开数据库

sql 'alter database open';

### RMAN联机备份

准备工作

* 开启归档模式,9i需要设置alter system set log\_archive\_start=true scope=spfile;

因为处于联机备份的数据库中要备份的所有数据文件头中的SCN被锁定，此时依旧可以对数据库表进行操作,修改的数据不能写入数据文件，而是被重做日志进程写入🡪重做日志文件。当备份时间过长而且在此期间产生了大量的变化数据，重做日志会切换将变化的数据写到归档日志文件中。

* 开启快闪恢复区，设置归档重做日志备份保存位置，容量大小。

设置归档模式

Shutdown immediate;

Startup;alter database archivelog;

查看归档模式,使用快闪恢复区作为备份文件，归档日志存储区。FILE格式为OMF

Archive log list;

#### 联机备份整个数据库

备份整个数据库（包括数据文件和归档日志文件）,备份归档日志文件，备份完成后删除归档日志目录下的归档文件。数据文件备份完，oracle自动发生一次日志切换，继续备份剩余的归档日志，因为只有一个归档日志文件。

备份整个数据库时，备份数据文件，其中包括当前控制文件和参数文件。重做日志文件或归档日志文件不是联机状态数据库全备份的内容，所以使用联机热备份的数据库在数据恢复时要recover数据库，即将联机备份开始到故障点之间的所有提交的数据重新写入数据文件。

R>Backup as compressed backupset database plus archivelog delete all input;

手工指定多个通道并将数据文件分布到不同通道上,format在指定通道时指定

$Ran {

Allocate channel ch1 device type disk format ‘/database/backup/ch1\_%U’,

Allocate channel ch2 device type disk format ‘/database/backup/ch2\_%U’;

Backup as backupset

(datafile 1 ,4 channel ch1)

(datafile 2,3,4 channel ch2);

Sql ‘alter system archive log current’;

}

Format在backup命令中指定,**format决定备份集的目录和名称**

$Run {

Allocate channel ch1 device type disk;

Allocate channel ch2 device type disk;

Backup as backupset format ‘/database/backup/%U ’

(datafile 1 ,4 channel ch1)

(datafile 2,3,4 channel ch2);

Sql ‘alter system archive log current’;

}

备份流程图如下：

Datafile1

Datafile2

Datafile3

Datafile4

Datafile5

Datafile6

通道1

RMAN

Channel2

Disk

Channel3

不指定datafile分配，rman会根据文件的大小大致分配，尽量保证通道分配平均。手工分配后会将默认的通道覆盖掉，默认通道的分配取决于PARALLELISM参数

#### 联机备份一个表空间

备份表空间

$Backup tablespace users;

压缩备份,压缩比大概5：1

$Backup as compressed backupset tablespace users;

#### 备份一个数据文件

通道生成的备份文件的名称和格式如下:

* %c:备份片的副本数
* %d：数据库名称
* %D：位于该月第几天
* %M：位于该年的第一个月
* %n：数据库名称，向右填补到最大19字符
* %u：一个19个字符的名称，表示备份集和创建时间
* %p：该备份集的备份片号，从1开始到创建的文件数
* %U:一个唯一的名字%u\_%p\_%c
* %s:备份集的编号
* %t:备份集的时间戳
* %T:年月日格式YYYY-MM-DD

备份数据文件

Backup as backupset datafile 1 format ‘/backup/datafile\_1%U’;

备份控制文件

$Backup current controlfile;

默认情况下RMAN在备份时检查数据块是否发生物理损坏，为加快速度关闭物理检查

$Backup nochecksum tablespace users tag=’weekly\_backup’;

默认情况，RMAN不会检查数据块是否发生逻辑损坏，启动逻辑损坏检查

$Backup check logical tablespace users;

RMAN备份时发现新的坏块，就立即停止备份。如果发现的坏块是上次已经发现的继续备份。设置最大坏块数达到指定数量停止备份

$Ran {

Set maxcorrupt for datafile 2,4 to 10;

Backup database;

}

数据文件2,4出现新块的数目超过10停止备份。解决方案，使用以前的RMAN备份恢复，然后在备份数据文件

### RMAN增量备份

BACKUP DATABASE是全库备份，增量备份相对而言，只备份自上次备份以来变化的数据。优点：效率比全库备份块，备份数据量减小。

2个级别的增量备份，级别0的增量备份与全库备份相同。级别1的增量备份执行的是差异备份，即对级别0的备份后变化的数据做备份。另外一种级别1的增量备份为累积备份，每次实现增量备份时，备份自级别0备份以来所有变化的数据。累积备份相对差异备份，他只有一个备份文件，而且可以减少数据库恢复时间。

使用RMAN实现增量备份的级别0备份

$Backup incremental level 0 database;

实现增量备份的级别1备份

$Backup incremental level 1 database;

### 快速增量备份

使用增量备份减少全库备份的时间，节约空间。无论上次增量备份数据库是否发生变化，增量备份必须要扫面整个数据文件，以确定是否有变化的数据。

避免上述情况，引入快速增量备份。原理是将数据库中发生变化的数据块位置记录在一个更改跟踪文件中。下次实现增量备份时通过此文件来备份变化的数据。

启动块跟踪特性后，会启动一个后台进程CTWR负责将变化的数据块的位置写入定义的块跟踪文件。

启动快跟踪特性

>Alter database enable block change tracking Using file ‘目录/chtrack.log’;

log文件丢失或者损坏都会造成数据库无法启动，禁用此跟踪即可

Alter database disable block change tracking;

查询是否启动or关闭快跟踪特性

>Select filename,status,bytes from v$block\_change\_tracking;

修改跟踪文件的存放位置or重命名，需要数据库在mount状态

Alter database rename ‘目录/chtrack.log’ to ‘目录/chtrack.log’;

### 在映像副本上应用增量备份

Oracle11g能够将增量备份应用到某个镜像副本上，镜像部分是对数据库的一个映像备份。

作用如下

* 第一天生成一个镜像副本
* 第二天进行增量备份，然后将产生的增量备份应用到第一天所做的镜像副本上，这时第一天备份的镜像副本就是最新数据。

以此类推。对镜像副本应用增量备份最大好处在于加快恢复速度。

对镜像副本应用增量备份

$Run{

Backup incremental level 1 for recover of copy with tag ‘incr\_copy\_backup’ database

Recover copy of database with tag ‘incr\_copy\_backup’

}

第一条语句表示，生成级别为1的tag值为’incr\_copt\_backup’，针对整个数据库并且应用于增量备份的镜像副本。

第一次执行程序，

执行第一条语句时由于没有0级备份，所以会生成整个数据库的镜像副本(该副本可以在其上应用增量文件)

执行第二条时，因为没有增量备份，所以不会执行。

第二次执行时，执行第二条语句时，由于有了第一次的0级备份，所以会生成一个一级增量备份，执行第二条语句时，会将第一条语句生成的增量备份应用于第一次所生成的镜像副本上。以后的每一次都会生成一个增量备份，并将该生成的增量备份应用到镜像副本上。

如果需要恢复，先恢复镜像副本🡪应用最近一次增量备份以来的所有归档日志🡪恢复

每天执行run，能达到如下效果

* 每天将增量备份添加到镜像副本上
* 恢复时使用最近的镜像副本+少量归档日志，恢复速度会很快。

查询当前的映像副本

$list copy;

### 创建和维护恢复目录

1:创建恢复目录表空间,使用prod库的一个特定表空间存储RMAN备份元数据。

Create tablespace recat\_tbs datafile ‘/app/oracle/oradata/PROD/rcat\_tbs01.dbf’ size 100m;

2:创建恢复目录用户,为该用户创建默认表空间用来存恢复目录。

Create user recat\_owner identified by oracle default tablespace rcat\_tbs temporary tablespace temp;

Alter user rcat\_owner quota unlimited on rcat\_tbs;配额限制

3:为该用户授权，使得该用户为恢复目录拥有者

Grant recovery\_catalog\_owner to rcat\_owner;

Grant connect,resource to rcat\_owner;

4:连接到恢复目录数据库，才能使用恢复目录

$Rman catalog rman/oracle@orcl

如果恢复目录数据库和目标数据库在同一个数据库上，上述方法同时也连接到目标数据库。

5:如果不在的话，需要如下连接，才能连接上目标数据库

$connect target system/oracle@orcl

一次连接到恢复目录和目标数据库

$Connect catalog rman/oracle@orcl target system/orcl@orcl;

6:连接到恢复目录数据库创建恢复目录,该表空间在创建recat\_owner用户时指定的默认表空间

$Create catalog tablespace rcat\_tbs;

删除目录

$drop catalog rcat\_tbs;

7:注册目标数据库。使得恢复目录知道目标数据库的信息，并且自动与目标数据库通信获得相关元数据。要注册目标数据库，需要连接目标数据库

Rman target rman/oracle catalog rcat\_owner/oracle@prod

Register database;

8:同步恢复目录,第一次注册恢复目录没有目标数据库注册信息，同步即可获得相关信息。

Resync catalog;

### RMAN的脚本管理

**创建RMAN备份脚本,要求必须连接到恢复目录和目的数据库**

Create script rman\_backup{

Sql ‘alter system checkpoint’;

Backup database format

‘\backup\offline\_backup\back\_%u.dbf’;

Backup current controlfile format

‘\backup\offline\_backup\back\_ctl\_%u.dbf’;

}

执行脚本

Run {execute script rman\_backup;}

**使用操作系统文件存储RMAN指令，并在RMAN中调用该文件执行RMAN命令**

**1:创建rman\_backup.rcv文件,内容如下:**

Sql ‘alter system checkpoint’;

Backup database format

‘f:\backup\offline\_backup\back\_%u.dbf’;

Backup current controlfile format

‘\backup\offline\_backup\back\_ctl\_%u.dbf’;

**2:调用操作系统执行RMAN指令**

在rman\_backup.rcv文件当前目录>

* rman catalog rman/oracle@orcl target system/oracle@orcl cmdfile ‘rman\_backup.rcv’;
* $sql ‘alter system checkpoint’;

Backup database format

‘f:\backup\offline\_backup\back\_%u.dbf’;

Backup current controlfile format

‘\backup\offline\_backup\back\_ctl\_%u.dbf’;

将脚本文件转换为操作系统文件,

$Print script rman\_backup to file ‘rman\_backup.txt’;

### 使用RMAN非归档模式下的完全恢复

模拟整个过程

1：启动数据库到mount状态

Startup mount;

2:备份(使用快闪恢复区作为备份文件的存储目录)

RMAN TARGET RMAN/ORACLE;

Backup as compressed backupset database;

3:创建表

Crate table test as select \* from t;

4：删除控制文件，数据文件，以及重做日志文件。模拟丢失

5:提示无法重新启动数据。

6: 恢复控制文件

$restore controlfile from ‘控制文件目录中保存的备份文件地址’;

7:启动数据库到mount

Alter database mount;

8:恢复所有数据

$resore database;

9:恢复数据库并使用noredo选项(因为已经丢失了所有重做日志文件，所以无法使用此选项)

Recorer database noredo;

10:打开数据库(因为没有重做日志文件，所以使用resetlogs打开数据库，会创建新的重做日志文件，日志序列号从1开始)

Alter database open resetlogs;

11:查看当前重做日志序列号如何记数

Select group#,sequence#,status from v$log;

12:发现创建的表丢失，说明只能恢复上次备份的文件。

#### 只有数据文件丢失的恢复

要保证重做日志文件没有被覆盖。

1：删除表空间的物理文件

2: start；无法启动数据库，提示无法识别or锁定数据文件,对应信息下面有文件编号

3:使用rman恢复数据文件

$resore datafile 文件编号

4:查看文件是否存在。

5：使用recover恢复数据文件，过程是使用重做日志文件记录的过程，包括回滚和后滚2步骤

$recover datafile 文件编号;

6:恢复完成，因为没有覆盖重做日志文件。如果失败，使用如下实现不完全恢复

Recover datafile 文件编号 until calcel;输入cancel

$alter database open;打开数据库

#### 联机重做日志文件和数据文件损坏的恢复

模拟控制文件没有丢失，所有的重做日志文件和数据文件损坏

1:删除所有重做日志和数据文件

2：start;重新启动数据库，由于控制文件没有丢失，所以数据库知道数据文件位置，试图打开数据文件，发现文件不存在，数据库无法启动。如果是非当前的默认永久表空间，或者不是system表空间，undo空间，可以使用 alter database datafile …offline的方式，使得数据库离线，然后打开数据库。

3:使用RMAN复原数据库

$Restore database;

4:恢复数据库

Recover database until cance;

5:重做日志文件没有恢复

Alter database open;失败

6：使用resetlogs创建依据控制文件中记录的重做日志组信息重建日志文件

Alter database open resetlogs;

#### 将数据文件恢复到其他磁盘目录

数据文件损坏后or磁盘损坏无法在使用，更新磁盘时间上不现实，所以使用恢复到指定目录。

1:数据启动到mount状态

2：恢复

Run {

Set newname for datafile

‘目录’ to ‘目录’;

Set newname for datafile

‘目录’ to ‘目录’;

Restore database from tag=TAG2011…;

Switch datafile all;

}

3:recover database until cancel;

4:alter database open resetlogs;

### 使用RMAN归档模式下的完全恢复

归档模式下，使用RMAN的备份和所有归档重做日志以及当前的重做日志文件可以实现数据库完全恢复。

要求RMAN备份以来数据库都运行在归档模式，且归档文件和重做日志文件没有损坏。

优点：可在联机下恢复数据库，不需要关闭数据库。从而不影响其他业务。

#### 非系统表空间损坏的恢复

数据库运行在归档模式，表空间USERS数据文件损坏，有完整的备份以及当前归档的日志文件。

为保证其他业务不中断，将该表空间offline，然后使用备份复原数据文件，在使用归档日志，当前重做日志恢复数据文件。

解决步骤

1：根据路径删除表空间user的数据文件

2:重启数据库报错，将数据文件离线，在打开数据库,错误信息有文件编号

Alter database datafile 文件编号 offline;

Alter database open;

3:启动RMAN还原数据文件

$restore datafile 文件编号;

4:应用归档日志或者重做日志数据恢复数据文件

Recover datafile 文件编号;

5：数据文件online

Alter database datafile 编号 online;

6:验证

Select \* from 表

**也可以使用如下方式完成上述操作,run内的操作都作为事务处理**

$Run {

Sql ‘alter database datafile 文件编号 offline’;

Restore tablespace users;

Recover tablespace users;

Sql ‘alter database datafile 文件编号 online’;

}

#### 系统表空间损坏恢复

System表空间损坏，而控制文件，重做日志文件完好。此时需要把数据库启动到mount状态，使用RMAN恢复表空间。因为在system表空间损坏数据库无法启动，所以只能在mount恢复

* 启动到mount状态
* 恢复system表空间

$run{

Slq ‘alter database datafile 1 offline’;

Restore datafile 文件编号;

Recover datafile 文件编号 ;

Sql ‘alter database datafile 编号 online’;

}

* 打开数据库

alter database open;

#### 所有数据文件丢失的恢复

Run{

Restore database;

Recover database;

Sql ‘alter database open’;

}

### RMAN实现数据块恢复

某个数据文件的数据块损坏，就造成整个数据文件无法使用，必须通过备份恢复整个文件，修复时间长。所以引入块级恢复。通过数据文件的完成备份恢复

* 打开rman

Rman target /;

* 备份整个数据库

Backup database plus archivelog;

* 关闭数据库修改数据文件

Shundown immediate;

* 损坏后，无法启动数据库

Startup;

* 检验数据文件

$backup validate datafile 文件编号;

* 查看数据文件中损坏的数据块，也可以使用告警日志文件

Select \* from v$database\_block\_corruption;

* 数据块恢复

$blockrecover datafile 文件编号 block 数据块号 from backupset;

* 提示有块未恢复，可以查看告警日志文件，所以在打开数据库前必须recover

Recover datafile 文件编号;

Alter database open;

**完整恢复完成**

### RMAN的备份维护指令

#### VALIDATE BACKUPSET指令

启动数据库文件备份集验证，成功的话说明此备份集是有效的，可用于恢复操作

Validate backupset (rman的所有备份集中代表某个备份集的关键字)

查看备份集汇总信息

List backup summary;

#### RESTORE VALIDATE指令

验证数据库对象是否在当前的备份集中,用户恢复文件时，可以确认该对象备份信息是否存在

验证表空间是否在备份集中

Restore tablespace users validate;

验证数据文件是否在备份集中

Restore datafile ‘文件目的’ validate;

也可以使用数据文件编号

Restore datafile 4 validate;

#### RESTORE PREVIEW指令

查看执行恢复的所有文件是否存在，如恢复表空间时，查看该表空间中的所有数据文件是否在备份集中。

Restore database preview;

Restore tablespace 表空间 preview;

Restore datafile 5 preview;

#### LIST指令

查看当前的 备份信息，某个表空间所在的备份集，某个数据文件所在的备份集信息，查看备份是否存在，存在几个备份。

查看备份集信息

List backupset;

通过备份集标识来查看备份集

List backupset 5;

列出备份集

List backup of tablespace users;

查询某个数据文件在备份集中信息

List backup of datafile 1;

查看归档日志文件备份信息

List backup of archivelog all;

List backup of archivelog from time=’sysdate-2’;

查看控制文件以及参数文件备份

List backup of controlfile;

List backup of spfile;

List copy of controlfile;

查看备份汇总情况,key表示备份集标识。B表示备份集。LV里面的A表示是归档文件，F表示数据文件的完全备份。S的A表示可用。

List backup summary;

#### REPORT指令

显示数据库的结构,列出永久数据文件和临时文件。

Report schema;

列出需要备份的数据文件,此指令会与RMAN备份指令策略关联

Report need backup;

## ORACLE闪回技术

一种数据库恢复技术，具有恢复时间快，不需要备份文件的特点，他使得数据库可以回到过去的某个状态，可以满足用户的逻辑错误的快速恢复。

### 闪回级别

闪回级别(闪回粒度)

* 数据库级闪回：允许将整个数据库恢复到过去的某个时间点。如误删用户，误截断表，可以使用此恢复。
* 表级闪回：将表闪回到过去的某个时间点，或恢复到过去的某个SCN，闪回删除通过DROP指令删除的表

### 闪回数据库

使用闪回日志来恢复用户的逻辑错误，值针对用户逻辑错误的恢复。不涉及到整个数据库，恢复更具有针对性而且恢复时间大大减少。

闪回日志由oracle自动创建，并存储在闪回恢复区中，由闪回恢复区管理。快闪恢复区空间不足时会自动删除旧的闪回日志文件以腾出空间，所以不能保证闪回日志被保存的数据的可靠性。快闪恢复区中备份文件的存储优先。由RVWR进程负责。

闪回数据库，闪回日志不会被归档。

#### 启动闪回数据库

默认不启动，启动需要将数据库设置为归档模式，并启用闪回恢复区，因为闪回日志文件存储在此。如果在RAC环境下，必须将闪回恢复区储在集群文件或ASM文件中。

查看数据库归档状态

Archive log list;

Archive destination:快速恢复区目录。

确定归档的操作系统存储位置

Show parameter db\_recovery\_file\_dest;

启动归档模式

Startup mount;

Alter database archivelog;

查看将数据闪回到过去的最长时间

Show parameter db\_flashretention;

修改数据闪回到过去的最长时间, 60\*24\*2=2880保留2天

Alter system set db\_flashback\_retention\_target=2880 scope=both;

为避免由于备份数据库空间不足，删除较早的闪回日志，导致不能闪回到某个时间段的某个点。所以通过数据字典，评估需要的快闪恢复空间

Select estimated\_flashback\_size,retention\_target,flashback\_size from v$flashback\_database\_log;

estimated\_flashback\_size:系统估计的快闪恢复区大小

flashback\_size：当前闪回数据大小

启动闪回数据库，需要启动数据库到mount状态

Startup mount;

Alter database flashback on;

查看启动状态

Select dbid,name,flashback\_on from v$database;

启动后，数据库永久表空间都会受到闪回数据库保护。

#### 关闭闪回数据库

禁用对某个表表空间的闪回特性

Alter tablespace users flashback off;

查看该表空间是否不被闪回保护

Select name,flashback\_on from v$tablespace;

该表空间启用闪回数据库

Startup mount;

Alter tablespace user flash on;

关闭数据库级别的闪回数据库特性,关闭后闪回日志自动删除

Startup mount;

Alter database flashback off;

#### 闪回数据方法

* 使用RMAN闪回

闪回到过去某个时间点

$FLASHBACK DATABASE TO TIME=TO\_DATE(‘2010-5-25 10:01:11’,’yyyy-MM-dd HH23:MI:SS’);

闪回到过去某个系统SCN

$Flashback database to scn=638832;

闪回到特定日志序列号之前的状态，不包括序列号345；

$Flashback databse to sequence=345 thread=1;

* 使用SQL闪回

需要切换到mount状态

Flashback database to timestamp(sysdate-1/24);

Flash back database to scn 678854;

闪回后需要重新设置重做日志打开数据库，使得重做日志序列号重新计数

Alter database open resetlogs;

#### 闪回数据库

**模拟**

**创建用户🡪创建表并插入数据🡪删除该用户和数据🡪闪回数据库恢复**

Create user vfast identified by vfast account unlock;

Create table vtest(id number,name varchar2(20));

Insert into vtest vales(1,’20’);

Select to\_char(sysdate,’yyyy-MM-dd hh24:mi:ss’) from duan;

Drop user vfast cascade;

删除用户后，闪回日志中记录了相关数据,假设SCN为1111

Select \* from v$flashback\_database\_log;

记录闪回日志区域可以闪回的最早SCN以及最早时间，并且评估所需要闪回恢复区大小。

查询可以闪回到的最早时间,假设为2015-10-30

Select to\_char(oldest\_flashback\_time,’yyyy-MM-dd hh24:mi:ss’) from v$flashback\_database\_log;

闪回数据库恢复误删除用户vfast

Startup mount;

Flashback database to timespace to\_date(‘2015-10-30’,’yyyy-MM-dd’);

使用read only模式打开数据库，验证是否恢复。如果没恢复，可以继续使用闪回数据库恢复。

Alter database open read only;

Select username,account\_status from dba\_users where username like ‘VF%’;

使用resetlogs打开数据库,闪回日志依然有效，可以继续山回到以前某时间点OR SCN

Startup mount;

Alter database open resetlogs;

闪回数据库到某个scn

Flashback database to scn 1111;

使用RMAN也可以闪回，但是不能使用to\_char and to\_date;因此需要提前设置

NLS\_DATE\_FORMAT OR NLS\_LANG

如果闪回没有达到要求，撤销闪回操作

Recovery database ;

闪回太多，可以将数据库恢复到以前的某个时间

Recover database until;

#### 监控闪回数据库

查看可以闪回到的最小SCN号以及可以闪回到的时间点

Select oldest\_flashback\_scn,oldest\_flashback\_time from v$flashback\_database\_log;

查看系统当前scn

SELECT CURRENT\_SCN FROM V$DATABASE;

查看闪回日志开销,监控闪回日志写入闪回数据的各种开销。每一行记录一小时间隔状态

Select \* from v$FLASHBACK\_DATABASE\_STAT;

Begin\_time：当前闪回记录开始时间

End\_time: 当前闪回记录结束时间

Flashback\_data:在时间间隔内记录的多少闪回数据(字节)

Db\_data:记录时间间隔内有多少数据块读写(单位数据块)

Redo\_data:时间间隔内记录了多少重做数据(字节)

监控快闪恢复区变化

Select name,space\_limit,space\_used,space\_reclaimable,number\_of\_files from v$recovery\_file\_dest;

Name:快闪恢复区目录

Space\_limit：空间最大使用上限

Space\_used:已经使用空间

Space\_reclaimable:可以回收空间

#### 使用闪回数据库限制

**以下几个只能使用不完全恢复将数据库恢复到过去某个时段**

* 数据文件被删除or缩短
* 闪回时间范围内复原or重建一个控制文件
* 在resetlogs操作之前
* 表空间被删除

### 闪回删除

启动闪回后，删除表，物理没被删除，只是占用的空间回到空闲列表，也就是这段空间在某种条件下是可以被占用的

#### 闪回删除原理

使用drop table删除表时，该表不会被删除，而是保持原表位置，但是将删除的表重新命名，并将删除的表信息存储在回收站中，回收站记录了被删除表的新名字和原名字。此时被删除的表占有的空间没有被立即释放，变成数据库可以使用的潜在空间。记录在回收站中的信息会保留一段时间，直到回收站不足or使用purge指令删除回收站记录。

查看oracle是否启用闪回删除

Show parameter recyclebin;

开启闪回删除

Alter system set recyclebin=on scope=both;

#### 回收站使用

查看数据字典dba\_recyclebin结构

Desc recyclebin;

* OBJECT\_NAME:删除的表重命名
* ORIGINAL\_NAME：删除表原始名
* OPERATION：对表的操作，如删除表，所以显示为drop
* TYPE：删除的数据库对象类型，如表，索引
* TS\_NAME：删除的数据库对象对应的表空间
* CREATETIME：被删除对象的创建时间
* DROPTIME：删除时间
* CAN\_UNDROP：是否能闪回删除
* CAN\_PURGE：是否可以永久删除

select \* from Dba\_recyclebin; 🡪select \* from user\_recyclebin;🡺show recyclebin;(是user\_recyclebin的同义词)

#### 恢复删除表

create table tempS(name number(10));

drop table tempS;

Flashback table tempS to before drop;

Select table\_name from user\_tables;

Flashback table tempS to before drop rename to new\_temp;

#### 永久删除表

Drop table test purge;

永久删除表空间，同时会触发将回收站中和表空间相关联的全部删除

Drop tablespace test including contents;

永久删除回收站表

Purge table test1;

永久删除回收站表空间，同时删除回收站中与该表空间联系的所有表

Purge tablespace users;

永久删除回收站中和某个表空间相关的某个用户的表

Purge tablespace test user scott;

### 闪回表(回到历史某一刻时间)

闪回表，将表里的数据回退到历史上某个时间点。闪回表利用undo记录的数据就映像。

指定undo表空间retention guarantee保证闪回成功

Show parameter undo;

Alter system set undo\_retention=86400 scope=spfile;

Alter tablespace undotbs1 retention guarantee;

闪回表的操作会修改表里的数据，从而可能引起数据行的移动。比如某一行数据在当前A数据块里面，而在表闪回到以前的某个时间点上，在哪个时间点上该行数据位于B数据块里面，针对这种情况，需要启动数据行的移动特性

Alter table 表名 enable row movement;

Flashback table 表名 to timestamp to\_date (‘’,’’);

局限性：假设当前时间为B，需要将表回到历史上的A点，如果A🡪B这段时间内对表执行了DDL操作，则闪回表失败。

### 闪回版本查询

版本：每次事务引起的数据行的变化情况，每一次变化就是一个版本。

闪回版本查询使用的是undo表空间里记录的undo数据

Select 伪列1，伪列2，伪列3…from version scn between … where 伪列1;

伪列：

* Versions\_starttime:事务开始时间
* Versions\_startscm:事务开始的SCN号
* Versions\_endtime:事务结束时间
* Versions\_endscn:事务结束的SCN
* Versions\_xid:事务的ID号
* Versions\_operation:事务所进行的操作类型，包括输入(显示为1)，删除(显示0)，更新(显示U)

Versions between scn minvalue and maxvalue：显示数据行所有变化

Select versions\_starttime,versions\_startscn,versions\_endtime,versions\_endscn,versions\_xid,versions\_operation,id,name from 表名 versions between scn minvalue and maxvalue;

### 闪回事务查询

Flashback\_transaction\_query显示那些事务引起数据变化，并提供撤销事务的sql语句。闪回事务查询利用的是undo表空间undo数据

Select xid, operation,undo\_sql from flashback\_transaction\_query where table\_name=’表名’ order by xid;

### 闪回查询

查询该表过去某个时刻的数据情况。

Select count(\*) from 表名 as of timestamp to\_date(‘’,’’);

### 复原点技术

将数据恢复到该点时状态,复原点是SCN别名

创建一个复原点

Create restore point rpl;

查看创建复原点

Select name,scn,storage\_size,guarantee\_flashback\_database from v$restore\_point;

此还原点无法保证闪回数据成功，因为这依赖于需要的闪回日志数据文件是否存在，而闪回日志依赖于闪回恢复区。

如下还原点，使得在快闪恢复区空间保证的情况下总可以闪回到该还原点，如果快闪恢复区空间不足则数据库关闭。与还原日志无关，还原日志还是会保存自复原点之后的闪回日志不删除。

创建一个有保证的还原点

Create restore point 还原点名 guarantee flashback database;

查看是否创建,

Select name,scn,storage\_size,guarantee\_flashback\_database from v$restore\_point;

guarantee\_flashback\_database表示是否是保证的还原点

storage\_size:有值，说明为保证的复原点指定的存储空间

删除还原点

Drop restore point 还原点名;

闪回删除不是直接将数据从数据库中删除放入回收站，而是将数据库对象的定义从数据字典中删除，数据存储在原处，在回收站中记录这个被删除的数据库对象。一旦需要闪回到删除前的状态，使用回收站中记录的对象信息进行闪回

## 手工管理的备份恢复

### 备份

#### 物理备份

将数据库文件(数据文件，控制文件，重做日志文件)复制到指定目录作为数据文件备份的方式。2种实现方式

使用操作系统工具CP and dd来本分和管理数据文件

使用ORACLE实用工具RMAN来备份和管理数据文件。可以使用命令行，也可以使用GC or EM图像化工具。

#### 逻辑备份

指使用oracle体用的数据迁移工具，如EXPDP,EXP等。导出数据库对象的逻辑结构以及数据。恢复时需要数据库创建所有的数据库对象并插入相应的数据。

#### 冷备份和热备份

冷：数据库关闭的情况下的备份如果数据库是正常关闭的如shutdown immediate。此时冷备份是一致的数据库备份、

热：在数据库打开的状态下的备份，此时用户可以继续访问数据库，执行DML操作。但是要求必须归档模式。归档模式下可以对整个数据库，单独的表以及数据文件进行备份。

#### 数据库恢复

##### 实例恢复

实例突然失败造成的数据库故障，在启用数据库会实例失败，实例恢复使用当前数据文件以及当前的重做日志文件实现数据恢复。由oracle自动完成。

实例恢复2步骤，

前滚：将重做日志文件中记录的用户提交和未提交的事务涉及的数据写入联机数据文件。

RoolForward🡪commit records(uncommit records)🡪datafiles

回滚：将未提交的事务涉及的修改通过UNDO中的记录回退回去，

重做日志文件很大，恢复时间会很长，因此引入fast\_start\_mttr\_target使得oracle定时将脏数据写入数据文件，来调整检查点。检查点之前的提交事务不需要回滚，间接提交速度。

##### 介质恢复

由于磁盘损坏或者数据文件损坏而需要的数据库恢复方式。oracle不自动完成，需要用户恢复。介质恢复需要具有备份的数据文件，如果完全恢复还需要自备份依赖的归档日志文件，当前日志文件。

介质恢复2步骤

复原：将备份的数据文件复制到相应目录，如果不是该文件以前的目录需要使用alter database rename datafile指令告诉数据库新的数据文件位置。

恢复：使用归档重做日志以及当前的日志文件恢复数据库到最新状态，使用前滚后在使用后滚恢复。

##### 完全和不完全恢复

完全恢复：当前数据发生故障后，通过备份将数据库恢复到最新状况，不丢失任何数据，如果数据库有数据文件备份和自备份以来全部的归档日志文件和当前的日志文件，就不会丢失数据。可以不同粒度实现恢复，如数据库级，表空间级，数据文件级。

不完全恢复：不能把数据库恢复到最新状况，会丢失数据。可以将数据库恢复到指定的时间点，通常这种事是由于丢失归档日志or当前日志造成。只能在数据库级实现不完全恢复，不能再表空间级or数据文件级实现不完全恢复。

### 非归档模式下的冷备与恢复

因为数据完全关闭，所以冷备是一致的数据库备份。

缺点：备份全部数据文件，数据文件过大，则需要更多的存储空间。变化过小，则不划算。

#### 准备步骤

查看数据库文件的存储目录

Select file\_name from dba\_data\_files;

查看控制文件

Select name from v$controlfile;

查看重做日志文件

Select group#,number,status from v$logfile;

关闭数据库

Shutdown immediate;

创建备份目录，并把所有数据文件，控制文件，重做日志文件复制到该目录

打开数据库

Startup;

**非归档模式备份的数据库会造成从备份开始到故障发生之间的所有用户提交的数据丢失，所以备份只能恢复到冷备开始时。**

#### 冷备下恢复

创建表并插入数据

Create table temps(name number(10));

Insert into temps values (123);

删除文件模拟故障

重启数据库报错,关闭数据库

Startup;shutdown immediate;

复制回相关文件

打开数据库

Startup;

查看数据文件是否存在

Select file\_name from dba\_data\_files;

创建的表丢失。

Desc temps;

#### 模拟缺少重做日志文件的恢复方法

* 将备份的数据文件和控制文件复制到相应目录
* 删除数据库文件
* Startup;启动数据库，提示没有控制文件错误
* 使用备份控制文件恢复数据库,需要在mount状态

Recover database until cancel using backup controlfile;

Cancel:让oracle到归档日志中寻找需要的日志数据，满足条件自动停止

* 打开数据库

Alter database open;提示错误

Alter database open resetlog;

* 查看当前重做日志信息

Select group#,status,member from v$logfile;

Oracle通过控制文件中记录的重做日志文件的信息重建了重做日志文件组以及重做日志成员。

* 查看重做日志状态信息

Select group#,sequence#,bytes,status from v$log;

**自备份以来所有提交数据丢失**

### 归档模式与非归档模式

Oracle重做日志文件是循环使用的。再切换日志时，如果启动了归档模式，后台归档进程ARCH会启动，当前重做日志中的记录会存储到归档目录下，形成归档文件。

#### 设置数据库的归档模式

Alter database open;

* 查看是否是归档模式

Archive log list;

* 查看PROD库进程信息

Ps –ef|grep arc;

* 手工归档当前的重做日志

Alter system archive log current;

* 数据库运行时使用指令设置数据库自动归档模式

Alter system archive log start;

* Oracle 10g之前，重启数据库后归档进程不自动启动，设置静态参数，重启数据生效，该参数使得每次启动数据库归档进程自动启动.**10g以上版本不需要设置。**

Alter system set log\_archive\_start=true scope=spfile;

#### 设置归档进程相关参数

**LGWR进程将内存中的数据写入重做日志文件，是内存读磁盘写。而ARC进程是将重做日志文件写入归档文件，是磁盘读磁盘写。效率比前者慢。当大量DML操作时，频繁写入归档文件，归档慢重做日志写入快，会导致挂起。此时数据库什么不做等待归档完成。针对这情况多开归档进程解决。**

查看当前归档进程数量

Show parameter log\_archive\_max\_processes;

修改归档进程数量

Alter system set log\_archive\_max\_processes=3;

查看归档进程

Ps –ef|grep arc

**备份多份归档日志目录，解决一旦一份文件出错无法恢复数据库情况**

查看归档日志目录or状态

Show parameter log\_archive\_dest;

创建多个归档目录

指定多个归档目录

Alter system set log\_archive\_dest\_n=’目录名 mandatory’;

Alter system set log\_archive\_dest\_n=’目录名 optional;

Mandatory：要求改归档日志必须写成功,即oracle要求日志数据文件必须归档到该目录成功后才可以切换。

Optional：没有归档成功也可以切换日志。

查看当前目录设置状态

Show parameter log\_archive;

执行日志切换启动归档

Alter system switch logfile;

去相应的目录下是否存在归档文件

设置必须要归档成功的最小数量

Alter system set log\_archive\_min\_successed\_dest=2;

### 手工热备份数据库步骤

热备：数据库运行过程中的数据库备份。可备份某表空间一个或者多个数据文件。备份过程中表空间依然可用，支持DML操作。

热备过程中数据文件头SCN锁定，不会有任何数据写入数据文件。之所以还支持DML操作是因为此时数据的变化都写入了重做日志文件，当表空间or数据文件结束备份模式时，会触发恢复过程，将涉及到该表空间的所有变化数据写入该表空间，提升数据文件头的SCN，使得控制文件和数据文件中记录的SCN一致。

步骤：

* 表空间处于备份模式
* 使用操作系统工具将表空间对应的数据文件备份到指定目录
* 结束备份模式
* 归档当前日志，备份归档日志。

查看表空间对应的数据文件

Select file\_id,file\_name,tablespace\_name from dba\_data\_files;

将表空间置于备份模式

Alter tablespace 表空间名 begin backup;

查看数据文件是否处于备份状态

Select \* from v$backup;

复制数据文件到其他目录

结束表空间的备份模式

Alter tablespace 表空间名 end backup;

将当前重做日志文件信息写入归档日志文件

Alter system archive log current;

查看当前数据文件检查点信息

Select checkpoint\_change#,file# from v$datafile;

### 热备过程中的对数据库崩溃的处理方法

模拟过程，步骤如下

启动表空间备份模式

Alter tablespace users begin backup;

查看表空间是否开启

Select \* from v$backup;

Change#表示scn号

查看数据文件起始校验点(SCN号)

Select checkpoint\_change#,file# from v$datafile;

重启数据库，因为数据文件头与控制文件scn不一致，无法启动,此时数据库处于mount

Shutdown abort;

Startup;

查看那些文件处于备份状态，记录编号

Select \* from v$backup;

使用reconver指令恢复数据文件

Recover datafile 编号;

打开数据库

Alter database open;

查看那些文件处于备份状态，记录编号

Select \* from v$backup;

**方法2：结束表空间备份模式，解锁数据文件头SCN，oralce会自动恢复**

Alter tablespace users end backup;

### 备份控制文件

**1：二进制备份**

* 备份到指定目录

Alter database backup controlfile to ‘目录/文件名.bak’;

查看目录下文件

Host ls 目录;

* 备份到trace文件(user\_dump\_dest参数指定的目录下)

Alter database backup controlfile to tracel;

确认用户的dump目录

Show parameter user\_dump\_dest;

查找trace文件

Select spid from v$process where addr=(Select paddr from v$session where sid=(Select sid from v$mystat where rownum=1));

**使用控制文件脚本即可生成控制文件;**

**2：使用Rman自动备份**

备份的触发条件是控制文件的内容发生变化，如创建了新表空间。

设置Rman的控制文件自动备份

Configure controlfile autobackup on;

设置控制文件的备份目录

Configure controlfile autobackup format for device type disk to ‘目录/%F’;

创建表空间

Create tablespace test datafile ‘目录/文件.dbf’ size 100m;

再去控制文件备份目录查看备份文件是否存在。

### 介质恢复原理

数据库文件损坏or数据文件所在磁盘故障，需要数据库恢复称为介质恢复。

步骤如下

1：还原，使用备份的文件替换丢失or损坏的文件。将备份的文件复制到坏文件目录，或者复制到不同目录，在修改控制文件配置

2:恢复，将归档的日志文件以及联机重做日志文件里的重做记录应用到还原出来的文件上，使用recover命令完成。

打开数据库时，oracle读取控制文件，对控制文件以及数据文件中的SCN比较，不一致则恢复。数据文件，控制文件，联机重做日志文件不同步，恢复。SCN基于系统时间戳，所以不会跟本机时间不一致。SCN是唯一的。

控制文件记录了3种SCN,如下:

1：系统检查点SCN

检查点进程启动时，将检查点结束时间转换为SCN号记录在控制文件中。发生文件级别检查点时，不会更新该系统检查点SCN。因为他是全局范围的。

查看系统检查点，该视图就是从控制文件中读取信息

Select name,checkpoint\_change# from v$database;

2：数据文件检查点

检查点进程启动时，包括全局范围以及文件级别的检查点都会在控制文件里面记录每个数据文件上所发生的检查点SCN。

将表空间设置只读，begin backup，或某个数据文件设置为offline会触发文件级别的检查点并将该检查点更新到控制文件和数据文件头部，并以后不再变化。

查看当前的文件检查点

Select file#,checkpoint\_change#,last\_change# from v$datafile;

last\_change:表示数据文件结束的SCN

将表空间offline，测试检查点是否修改

Alter tablespace uers offline;

Select file#,checkpoint\_change#,last\_change# from v$datafile;

3:结束SCN

每个数据文件都会有结束SCN。在数据库正常运行期间，在线的，可读写数据文件，终止SCN都为空。而对非正常关闭数据库时，或者将数据文件正常离线，或只读时，都会触发检查点进程，从而在控制文件里记录每个数据文件上的结束时的SCN号。

将表空间users设置为ONLINE;

ALTER TABLESPACE USERS online;

查看文件检查点SCN信息

Select file#,checkpoint\_change#,last\_change# from v$datafile;

表空间设置为只读状态

Alter tablespace audit\_tbs read only;

查看检查点,修改的表空间起始和结束scn相同

Select file#,checkpoint\_change#,last\_change# from v$datafile;

表空间修改为读写模式，结束SCN为空

Alter tablespace tudit\_tbs read write;

Select file#,checkpoint\_change#,last\_change# from v$datafile;

查看数据文件头部的SCN信息

Select file#,checkpoint\_change# from v$datafile\_header;

发现上述检查点SCN不一致，触发系统检查点，保证一致

ALTER SYSTEM CHECKPOINT;

Select file#,checkpoint\_change# from v$datafile;

Select checkpoint\_change# from v$database;

### 归档模式下的完全恢复

设置归档目录

Alter system set log\_archive\_dest\_1=’location=目录 mandatory’;

Alter system set log\_archive\_dest\_2=’location=目录 mandatory’;

查询归档模式

Archive log list;

#### 数据文件在有备份情况下恢复

备份表空间users的数据文件

Alter tablespace users begin backup;

创建表t1，插入数据并提交

Create table t1(idnumber )tablespace users;

删除users表空间文件,模拟文件损坏

重启数据库，故障

Startup;

将数据文件or数据文件对应表空间离线

Alter database datafile 4 offline;

将备份文件复制到损坏的数据文件目录，如果数据文件所在磁盘损坏可以复制到其他目录，使用alter Database rename datafile修改控制文件中对该损坏数据文件的记录

查看哪些数据文件需要恢复

Select \* from v$recover\_file;

Change#:SCN号

查看数据文件信息，这里的4对应的是v$recover\_file的file#

Select file#,checkpoint\_change#,last\_change# from v%datafile where file#=4;

将4对应的表空间users设置为online，提示需要介质恢复

Alter tablespace users online;

Recover datafile 4;

查看表失败，因为该表空间设置的是离线状态

Select \* from t1;

Alter database datafile 4 online;

查看控制文件与数据文件中的SCN,一致

SELECT FILE#,CHECKPOINT\_CHANGE#,LAST\_CHANGE# FROM V$DATABASE WHERE FILE#=4;

Select file#,checkpoint\_change# from v$datafile\_header where file#=4;

#### 数据文件在无备份情况下的恢复

创建表空间，创建一个表并插入数据

Create tablespace test datafile ‘目录/test.dbf’ size50m;

Create table tsb(id number)tablespace test;

Insert into tsb values(1);

删除该目录下文件

修改系统刷新缓存，并查看表数据，显示错误

Alter system flush buffer\_cache;

Select \* from tsb;

将无法打开的文件离线

Alter database datafile 文件编号 offline;

创建新的数据文件

Alter database create datafile ‘目录/test.dbf’;

将文件在线，失败

Alter database datafile 文件编号 online;

恢复数据文件并将数据文件online

Recover datafile 文件编号;

Alter database datafile 文件编号 online;

查看表信息，完全恢复，因为启动了数据库归档模式，并且自该表空间创建后，所有插入的数据都被重做日志或者归档日志保护。

Select \* from tsb;

#### 系统表空间数据文件损坏的完全恢复

备份系统表空间，复制一份文件到其他目录,然后删除系统表空间文件

启动数据库，失败

Startup;

将数据文件offline,再打开数据库失败，

Alter database datafile 文件编号 offline;

Alter database open;

此时数据库处于mount状态，复制备份文件到系统表空间目录

将数据文件online，打开数据库失败

Alter database datafile 文件编号 online;

Alter database open;

介质恢复，并打开

Recover datafile 文件编号

Alter database open;

#### 当前undo表空间损坏的完全恢复

Undo表空间损坏，无法回滚。

查询当前undo信息

Show parameter undo;

查看相关数据文件信息

Select file\_name,tablespace\_name,online\_status,status from dba\_data\_files where tablespace\_name like ‘UNDO%’;

热备2数据文件

Alter tablespace 文件名 begin backup;

Host 手动复制Exit;

Alter tablespace 文件名 end backup;

删除还原表空间文件

创建用户，授权并创建表

Create user udo identified by oracle;

Grant resource ,connect to udo;

Create table ut(age number);

失败，因为非sys用户无法使用系统还原段

启动数据库到mount状态

Startup mount;

将备份文件cp到原来目录

恢复数据文件

Recover datafile 文件编号；

打开数据库，并查看还原表空间以及数据文件状态信息

Alter database open;

Select file\_name,tablespace\_name,online\_status,status from dba\_data\_files where tablespace\_name like ‘UNDO%’;

#### 非当前undo表空间损坏的完全恢复

创建还原表空间，查看表信息, 并查看还原表空间以及数据文件状态信息

Create undo tablespace undotbs2 datafile ‘目录/文件名.dbf’ size 100m extent management local;

Show parameter undo;

Select file\_name,tablespace\_name,online\_status,status from dba\_data\_files where tablespace\_name like ‘UNDO%’;

关闭数据库，删除表空间对应的数据文件，

重启数据库，报错。

Startup;

将数据文件离线，然后打开数据库

Alter database datafile 文件编号 offline;

Alter database open;

查看还原表空间以及数据文件状态信息

Select file\_name,tablespace\_name,online\_status,status from dba\_data\_files where tablespace\_name like ‘UNDO%’;

表空间数据文件依然记录在数据字典汇总，重建该表空间时会由于重名报错，删除

Drop tablespace 表空间。

通过重建方式恢复非当前的还原表空间

Create undo tablespace undotbs2 datafile ‘目录/undotbs2.dbf’ size 100m extent management local;

### 不完全恢复

场合：

* 应用归档日志过程中，丢失多个归档日志文件，只能将数据库恢复到最后丢失的日志文件为止
* 丢失当前正在使用的日志文件or丢失了没有归档的非当前日志文件
* 使用备份的控制文件完成恢复
* 误操作，如删除表，可以使用不完全恢复将数据恢复到误操作之前的时间点。

#### 不完全恢复类型

* 基于时间点的不完全恢复

Recover database until time ‘yyyy-mm-dd hh24:mi:ss’;

* 基于撤销的不完全恢复，恢复到输入cancel命令时，用于丢失联机重做日志or归档日志文件。

Recover database until cancel;

* 基于SCN号的不完全恢复。恢复到指定，历史上的SCN

Recover database until cancel scn scn号;

**建议使用不完全恢复时，对数据库进行冷备。恢复失败可以还原。**

### 所有控制文件丢失的恢复方法

模拟所有控制文件丢失，使用备份的控制文件实现完全恢复

创建表

Create table t1 tablespace users as select \* from scott.emp;

备份控制文件

Alter database backup controlfile to ‘目录/control.ctl.bak’;

创建表2

Create table t2 tablespace users as select \* from scott.dept;

将所有控制文件删除,启动数据库，报错。此时数据库处于nomount状态

Startup;

将备份的控制文件复制到指定目录

启动数据库到mount状态,查看控制文件和数据文件中的SCN，是否需要恢复，以提高控制文件中的SCN

Startup mount;

Select file#,checkpoint\_change# from v$datafile;

Select file#,checkpoint\_change# from v$datafile\_header;

不一致，则通过归档or当前日志文件记录来提升SCN恢复

Recover database using backup controlfile;

使用当前重做日志恢复,如果不是当前重做日志组，则需要切换

Recover database using backup controlfile;

打开数据库

Alter database open resetlogs;

Select \* from t1;

Select \* from t2;

完全恢复。

### 重建控制文件方式2

在没有备份也没有冗余情况下，使用。但是需要一些数据库信息，这些信息可以存放在trace文件中。

获取重建需要的数据库信息，并保存到指定trace文件,文件在user\_dump\_file目录下

Alter database backup contrilfile to trace;

创建表

Create table t2 tablespace users as select \* from scott.emp;

关闭数据库，删除所有控制文件，重启数据库报错。

使用trace文件的内容运行，作用重建控制文件

获取当前控制文件SCN.

Select checkpoint\_change# from v$database;

从当前日志文件的first\_change#获取,当前控制文件SCN.

Select group#,first\_change#,status from v$log;

从数据文件获取scn

Select filr#,checkpoint\_change# from v$datafile;

从数据文件头部获取scn

Select file#,checkpoint\_change# from v$datafile\_header;

恢复数据库

Recover database using backup controlfile;

此时使用完了归档日志，使用当前重做日志文件

Recover database using backup controlfile;

打开数据库

Alter database open resetlogs;

查看表是否存在

Desc t2;

## OEM管理与使用

## ORACLE实例优化

### SGA与实例优化

查询重做日志缓冲区大小

Show parameter log\_buffer;

查看sga\_max\_size

Show parameter sga\_max\_size;

查看SGA信息,total system global area与sga\_max\_size值一致

Show sga;

修改SGA最大大小,静态参数，设置当前内存一半即可

Alter system set sga\_max\_size=700m scope=spfile;

查看和sga相关的静态参数

Show parameter sga;

* Lock\_sga:将sga锁定在物理内存内，这样sga不会使用虚拟内存，可提高数据读取速度

Alter system set lock\_sga=true scope=spfile;

* Pre\_page\_sga:启动数据库实例时，将整个sga读入物理内存，提要效率

Alter system set pre\_page\_sga=true scope=spfile;

* Sga\_target:决定是否使用sga自动管理，默认值与系统的sga\_max\_size一样大，可动态修改

Alter system set sga\_target=700M;

Sga可以自动调整的内存，共享池，java池，大池，数据库缓冲区，流池

查看自动调整的内存组件信息,开启自动后，值自动设为0

Select name,value,isdefault from v$parameter where name in(‘shared\_pool\_size’,’large\_pool\_size’,’java\_pool\_size’);

也可以手动修改大小,不影响oracle自动调整

Alter system set java\_pool\_size=10M;

Show parameter java\_pool\_size;

### 将程序常驻内存

* 创建软件包DBMS\_SHARED\_POOL，不是默认安装，需要执行Sql脚本。

KEEP：实现将程序常驻内存（共享池）

UNKEEP：将程序清除出内存。

执行失败，因为没有安装包

Execute dbms\_shared\_pool.keep(‘HR.SECURE\_DML’);

* 安装包,非dba创建会失败,没权限

Sql>@G:\oracleDataBase\app\Administrator\product\11.2.0\dbhome\_1\RDBMS\ADMIN\dbmspool.sql;

* 查看用户拥有的存储过程

Select object\_name,object\_type from dba\_objects where object\_type=’PROCEDURE’ and owner =’HR’;

* 查看存储过程内容

Select line,text from dba\_source where name=’存储过程名’;

* 将过程常驻内存

Execute dbms\_shared\_pool.keep(‘用户名.存储过程名’);

* 查看数据库对象在缓存中的信息

Select name,namespace,sharable\_mem,executions,kept from v$db\_object\_cache where owner =’用户名’;

KEP:yes说明用户的数据库对象(存储过程)保存在共享池中。

* 将存储过程清除出常驻内存,kept值为no。如果没有将程序存入常驻内存则不会显示。

Execute dbms\_shared\_pool.unkeep(‘用户名.过程名’);

#### 从DBMSPOOL脚本理解软件包DBMS\_SHARED\_POOL

KEEP过程方法如下

Procedure keep(name varchar2,flag char default ‘P’);

参数1：用来说明数据库对象的名字，可以是过程，触发器，序列号，以及java对象。

参数2：说明要常驻的数据库对象的类型，默认类型为包，过程or函数。其他需要指定一个字符变量说明对象类型。

### 将数据常驻内存

SGA中段的已缓存块可以放在3个缓冲池中

* 默认池：默认区域，未设置的话所有的段都放在此(缓冲区池)
* 保持池：对于用户频繁访问的数据(表，索引or数据库对象的数据块等)可以放置在此候选的缓冲区池中。放在默认池中的数据块，也可以频繁访问，但是段数据会老化而退出默认池。推荐保持池，可以长期保存
* 回收池：对于随机访问的大段可以放此，因为大的数据段会很快老化退出缓冲池，导致缓冲区的频繁刷新输出。

保持池&回收池用户管理。

查看保持池大小信息

Show parameter keep;

查看当前数据库的数据块的默认缓冲池

Select id,name,block\_size,buffers from v$buffer\_pool;

#### 将数据常驻内存

查看表和索引的块大小

Select segment\_name,segment\_type,blocks from dba\_segments where owner =’Scott’ and segment\_name in (‘表名’,’索引名’);

由于dba\_segments是静态数据字典，使用如下获得最新段统计信息

Analyze index scott.索引名 compute statistics;

Analyze table scott.表名 compute statistics;

查询数据库块大小

Show parameter db\_block\_size;

计算要保存的表和索引的大小

Select (表块+索引块)\*数据库块大小 Kbytes from dual;

设置保持池的大小

Alter system set db\_keep\_cache\_size=10MB;

查看当前库的数据块的缓冲池信息

Select id,name,block\_size,buffers from v$buffer\_pool;

查看表&索引当前存放在什么缓冲池中

Select table\_name,tablespace\_name,buffer\_pool from user\_tables where table\_name=’表名’;

Select index\_name,table\_name,buffer\_pool from user\_indexes where index\_name=’索引名’;

将表设置为常驻内存,并查看是否修改成功，值为keep说明成功

Alter table 表名 storage (buffer\_pool keep);

Select table\_name,tablespace\_name,buffer\_pool from user\_tables where table\_name=’表名’;

将索引设置为常驻内存,并查看是否修改成功，值为keep说明成功

Alter index 索引 storage (buffer\_pool keep);

Select index\_name,table\_name,buffer\_pool from user\_indexes where index \_name=’ 索引’;

#### 将常驻内存的程序恢复为默认缓冲池

将表恢复为默认缓冲池，并查看是否更改成功

Alter table 表名 storage(buffer\_pool default);

Select table\_name,tablespace\_name,buffer\_pool from user\_tables where table\_name=’表名’;

将索引恢复为默认缓冲池

Alter index 索引 storage (buffer\_pool default);

只是修改了索引的缓冲池设置，但是保持池依然占用内存，但是其中已经没有数据，需要释放保持池中的内存。

Alter system set db\_keep\_cache\_size=0;

查看与数据库香瓜你的缓冲池信息

Select id,name,block\_size,buffers from v$buffer\_pool;

### 优化重做日志缓冲区

查看重做日志缓冲区大小

Show parameter log\_buffer;

优化重做日志缓冲区，需要确认发生了与重做日志缓冲区相关的等待时间。否则不应该随便修改大小,可通过WAIT视图和EVENT事件视图，确认等待时间以及该事件涉及的文件和会话

查看会话等待时间。

Select sw.sid,s.username,sw.event,sw.wait\_time from v$session s, v$session\_wait sw where sw.event not like ‘rdbms%’ and sw.sid=s.sid order by sw.wait\_time,sw.event;

查询与闩锁latch相关的信息

Select latch#,name,gets,misses,1-(missses/(gets+misses)) “gets rate” from v$latch where misses>1;

#### 设置重做日志缓冲区大小

查看重做日志缓存区大小

Show parameter log\_buffer;

Select name,value/(1024\*1024) “MB” from v$parameter where name=’log\_buffer’;

设置重做日志缓冲区大小，静态参数，需要重启

Alter system set log\_buffer=10485760 scope=spfile;

查询重做日志缓冲区大小

Show parameter log\_buffer;

### 优化共享池

共享池由2部分组成

**库高速缓存**：存放SQL正文，编译后的代码以及最终执行计划。

硬解析：在sql语句处理步骤中，解析需要经过sql语句的语法语义分析，基于优化模式选择优化方案，以及执行最终计划。

如果有相同的sql语句的执行计划已经缓存在库高速缓存中，此时就执行一个软分析。

**数据字典高速缓存**:存放于sql语句操作相关的数据库对象，如表，索引，以及其他对象和权限信息

库高速缓存，重用sql语句可以减少硬解析的时间，从而减少sql语句的响应时间。

数据字典高速缓存减少了对sql语句涉及的数据库对象和权限定义的磁盘访问，也减少了sql语句的响应时间

对共享池的优化目的就是不影响性能的情况下提高sql代码以及数据库字典的使用率

#### 使用绑定变量

2个sql语句不一样就分别执行硬解析，减少硬解析使用绑定变量，动态传入参数

Select \* from tablename where clo >&变量；

通过sql trace跟踪一个sql查询，查看是否执行了硬解析

清空共享池，目的方便对sql语句的分析

Alter system flush shared\_pool;

启动会话级的sql追踪功能

Alter session set sql\_trace=true;

执行sql插叙你语句并通过TKPROF解释该追踪文件

Select \* from tablename where id>4000;

文件中mis=1说明硬解析，0说明没硬解析

#### 调整CURSOR\_SHARING参数

查看cursor\_sharing默认值

Show parameter cursot\_sharing;

Exact:(默认值)只有正文完全相同的sql语句才可以重用。

Force:强制共享使用只有字面值不同的sql语句。

更改参数为force

Alter system set cursor\_sharing=force scope=both;

#### 设置共享池大小

查询共享池大小,0说明自动管理

Show parameter shared pool size;

可以安装数据库时设置，如下使用动态设置

Alter system set shared\_pool\_size=200M scope=both;

### 优化数据库高速缓存

查询相关缓冲区缓存数据

Select name,value from v$sysstat where name in (‘db block gets from cache‘,

‘consistent gets from cache’,’physical reads cache’);

db block gets :数据库高速缓冲区中存在被更改的数据，而此数据其他用户访问时已经提交。用户访问的数据在数据库缓冲区中是最新的版本

consistent gets：一致获取，数据库高速缓冲区中存在被更改的数据，而此数据其他用户访问时还未提交。也就是用户访问的是脏数据，这样的数据不会被访问，此时用户只能使用回滚段中的记录。

physical reads cache：物理读，在数据库高速缓冲区中没有用户要访问的数据，需要从磁盘中读取该数据块。

数据库高速缓冲区命中率=1-物理读/逻辑读

修改高速缓冲区大小

Alter system set db\_cache\_size=192M;

#### 缓冲池

用户访问多个表，且表相当大，此时表只有部分保存在库高速缓存区中，会降低数据库缓冲区的命中率，如果大表不是用户频繁访问的对象，可以使用缓冲池来存放。这样的数据库对象会被覆盖

用户访问多个小表，且频繁发生，可以保存到保留池中，这样对象不会被覆盖掉

其他的对象会默认存储在数据库高速缓冲区中的默认池中

缓存表的大小不要超过缓冲区的10%

CREATE INDEX 索引名 storage (buffer\_poo keep);

alter table 表名storage (buffer\_poo recycle);

alter INDEX 索引名 storage (buffer\_poo keep);

创建索引并缓存在保留池

Create index scott\_emp\_ename on scott.emp(ename) storage (buffer\_pool keep);

查看是否缓存在保留池

Run select index\_name,table\_name,buffer\_pool from user\_indexes where index\_name=’SCOTT\_EMP\_ENAME’;

将表放入回收池

Alter table 表名 storage (buffer\_pool recycle);

查看表缓存信息

Select table\_name,cache,buffer\_pool from dba\_tables where table\_name=’EMP’;

CACHE:Y，说明表已经缓存在缓冲区中，

使用cache关键字将表缓存在默认池中

Create table table2 as select \* from table1 cache;

查看表的缓存信息，默认使用default缓冲池，且没有被缓存

Select table\_name,cache,buffer\_pool from dba\_tables where table\_name=’TABLE2’;

开启缓存

Alter table table2 cache;

### 优化PGA内存

PAG主要作用大规模数据排序，如group by order by等操作。

PAG优化就是将大规模数据排序放在PGA中运行，而不是使用虚拟内存而占用操作系统的SWAP AREA。

查看PGA排序区大小

SORT\_AREA SIZE

PGA排序区设置自动管理，需要设置排序区的值(PAG\_AGGREGATE\_TARGET)和设置为自动管理(WORKAREA\_SIZE\_POLICY 为AUTO)

查看PGA排序区是否自动管理

Select name,value,isdefault from v$parameter where name in (‘pga\_aggregate\_target’,’workarea\_size\_policy’);

查看PAG状态信息

Select \* from v$pgastat;

Aggregate PGA target parameter :设置当前的PGA内存总和

Aggregate PGA auto target:为pga的排序区分配的内存大小。

Cache hit percentage:说明排序区在pga的排序区完成的比例。100说明全部的排序都在此进行。

查看PGA的排序区进行排序的百分比，如果不为100，则需要适当增加pag\_aggregate\_target的值

Select \* from v$pgastat where name like ‘cache%’;

调整pga内存大小

Alter system set pga\_aggregate\_target=76M;

查看pga内存大小

Show parameter pga;

查看pga大小以及pga中排序区大小

Select \* from v$pgastat where name in (‘aggregate PGA target parameter’ ,’aggregate PGA auto target’);

查看PGA的排序区排序的详细信息。

Select pga\_target\_for\_estimate/(1024\*1024)as estd\_target,estd\_pga\_cache\_hit\_percentage,estd\_overalloc\_count from v$pga\_target\_advice;