**Oracle**

一、Oracle概述

**1.数据库基础**

@ 数据-----描述事物的符号，是数据库中存储的基本对象

@ 数据库-----按照一定的数据模型组织、描述和存储的数据的集合，具有较高的数据独立性，小的冗余度，可以实现数据共享

@ 数据库管理系统-----为科学组织和存储数据库中的数据，高效率的获取和维护数据，采用DBMS的系统软件来管理数据库

位于用户和操作系统之间的一层数据管理软件，通常所说的数据库产品（Oracle,DB2,Informix,Sybase）都是DBMS。

@ 数据库系统-----数据库、数据库管理系统、计算机系统共同组成了一个数据库系统，通常由数据库，操作系统，DBMS，开发工具，应用系统，DBA，用户组成

用户(另类系统) 用户(终端) 用户(PC)

|

应用系统(订票系统)

|

开发工具（PL/SQL,Dbvisual）

|

DBMS

|

操作系统

|

数据库

**2.数据库的发展**

@ 人工管理阶段，1950以前，数据不共享，不独立，冗余量大

@ 文件系统管理阶段，1950-1965，数据可以长期保存，共享性差，独立性差

@ 数据库系统管理阶段，1965以后，数据结构化存，共享性强，独立性好，冗余量低，统一数据管理和控制

**3.数据模型**

@ ER图

实体（Entity）-矩形

属性（Attribute）-椭圆

关系（Relation）-菱形

@ 四种对象模型

层次模型---树

网状模型---图

关系模型---表，Oracle,DB2等流行数据库都是关系数据库

关系：表

元祖：行

元素：列

属性：列

优点：存储路径对用户透明，具有更高的数据独立性和保密性，简化了程序员的开发工作

缺点：查询效率往往不太乐观，必须对用户的查询语句进行优化。

对象模型---对象

**4.Oracle数据库简介**

Oracle公司于1979年发布的世界上第一个关系数据库

1992发布oracle7,1997.6发布oracle8,1998.9发布oracle8i，2000发布oracle9i，2004发布oracle10g

@ 特点

支持多用户、大事物量的事物处理

提供标准操作接口

实施安全性控制和完整性控制

支持分布式数据处理

具有可移植、可兼容性和可连接性

@ 应用结构

客户机-服务器（C/S）模式--客户端需要安装客户端程序，直接访问服务器，速度快，维护升级不方便，如PL/SQL DEVELEPOR

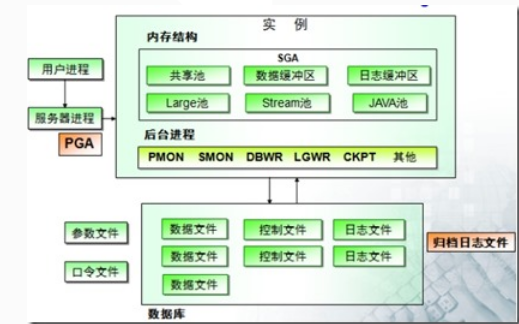
浏览器-服务器（B/S）模式--三级架构，客户端只要安装浏览器就OK了，通过访问WEB服务器对数据库进行操作和访问，方便了程序的维护和升级，但是降低了网络执行效率

分布式数据库模式--分布于计算机网络的多个逻辑相关的数据库组成，每个数据库都具有独立的处理能力

@ 创建数据库：安装时创建；DCBA创建

@ OEM和SQL\*PLUS

## 二、体系结构



**1 一般情况下是一个实例instance和一个database组成。实际生产很多实用到RAC技术，就是一个数据库对应多个实例。**

Oracle实例：实例是指操作系统中一系列的进程和进程所分配的内存块。通俗的说，实例是访问数据库文件的通道。

数据库是通过数据库实例来访问的，数据库实例由SGA（SYSTEM GLOBAL AREA）和PGA(前台进程，一个用户连接一般对应一个PGA)，8大后台进程组成。

Oracle的体系结构中有这么几个概念：实例、表空间、数据文件、用户、表。

Oracle数据库不同其他数据库，Oracle可以理解为一个大的数据库，就只有这一个。你所操作的“多个数据库”都是Oracle数据库下的不同的实例。安装Oracle的时候会默认给你创建一个实例orcl。

表空间是一个逻辑概念，一个数据库实例可以有多个表空间。可以说表空间是对这个数据库实例物理磁盘内存上的划分，也许你平时使用的时候并没有创建表空间，因为这个时候使用的是系统默认的表空间，后面详说。一个表空间会创建多个数据文件，这是个一对多的关系。

数据文件是用来存储数据的，这个顾名思义。是什么数据？平时接触最多的是我们平时创建的表，create table...但是表的概念还不能从这里引出。先说用户。

我看有的理解是一个表空间下有很多个用户，这个理解可以说是勉强正确。因为我觉得用户和表空间并不是上司与下属的关系，算是同级吧，都是隶属于数据库实例的。只不过在创建用户的时候都要给用户指定一个表空间，授权该用户可以访问的表空间，他们的关系是多对一，多个用户可以访问一个表空间，但一个用户只能访问一个表空间。（总感觉这句话逻辑不顺）如果没有指明表空间，则是默认表空间users表空间。而表只能由用户来创建，用户将这个表放入与用户绑定的表空间，但最后管理表的存放的确是这个表空间，表空间随机的将表存放到它所在的一个或多个数据文件中。

需要特别指出的是：

在创建用户时是只能指明一个默认表空间和一个默认临时表空间，你后续create table都是在这个默认表空间中，但是可以更改。

alter user xx default tablespace xxxx

**2 PGA**

用户进程对应应用服务器的连接，oracle数据库为每一个用户进程分配一个服务器进程（PGA）来处理该连接。

PGA（程序全局区）为每个服务进程私有的内存区域，程序全局区主要包含排序区、会话区、堆栈区和游标区四个部分的内容：

**（1）堆栈区**

保存绑定变量、会话变量、SQL语句运行时的内存结构等重要的信息。

**（2）游标区**

当使用游标时，会在游标区分配一块区域，当关闭游标时，会释放。 但是频繁创建和释放，需要耗费系统资源，降低语句的执行性能。

Oralce通过限制游标的数量来提高数据库的性能。数据库系统中有一个初始化参数OPEN\_CURSORS。管理员可以根据实际的需要，来设置这个参数，控制用户能够同时打开游标的数目。

ora-01000: 超出打开游标的最大数。在java代码中,执行conn.createStatement()和conn.prepareStatement()的时候，实际上都是相当与在数据库中打开了一个cursor。尤其是，如果你的createStatement和prepareStatement是在一个循环里面的话，就会非常容易出现这个问题。因为游标一直在不停的打开，而且没有关闭。

如果你不使用连接池，那么就没有什么问题，一旦Connection关闭，数据库物理连接就被释放，所有相关Java资源也可以被GC回收了。

但是如果你使用连接池，那么请注意，Connection关闭并不是物理关闭，只是归还连接池，所以PreparedStatement和ResultSet都被持有，并且实际占用相关的数据库的游标资源，在这种情况下，只要长期运行，往往就会报“游标超出数据库允许的最大值”的错误，导致程序无法正常访问数据库。

一般来说，写Java代码的时候，createStatement和prepareStatement都应该要放在循环外面，而且使用了这些Statment后，及时关闭。最好是在执行了一次executeQuery、executeUpdate等之后，如果不需要使用结果集（ResultSet）的数据，就马上将Statment关闭，调用close()方法。

**（3）排序区**

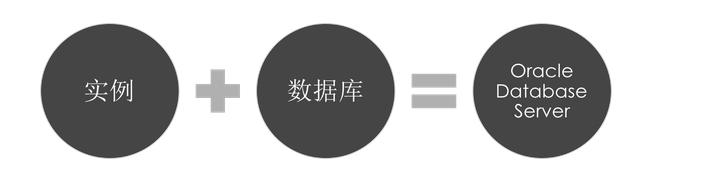
A、基于操作符的排序，group by、order by、rollup和窗口函数

B、hash散列连接

C、位图合并、位图创建

**（4）会话区**

这个会话区内保存了会话所具有的权限、角色、性能统计等信息。这个会话区一般都是由数据库进行自我维护的，系统管理员不用干预。



**3 SGA**

SGA（System Global Area）系统全局区，对系统内的所有进程都是共享的。当多个用户同时连接到一个例程时，

所有的用户进程、服务进程都可以共享使用这个SGA区。各种数据库的操作主要就是在这个SGA区内完成。

**（1）Buffer Cache**

数据库的数据缓存，oracle将数据文件的信息缓存在此区间。Buffer Cache的命中率等，直接影响数据库的性能。

**（2）Redo Log**

如果数据需要写到在线重做日志中，则在写至磁盘之前要在重做缓冲区（redo buffer）中临时缓存这些数据。由于内存到内存的传输比内存到磁盘的传输快得多，因此使用重做日志缓冲区可以加快数据库的操作。如果没有这个区，那操作一次数据库就访问一次磁盘，而如果有的话，则可以操作很多次之后才访问一次磁盘。速度当然要快很多了。

**（3）Shared Pool**

Shared Pool主要由两部分组成，一部分是库缓存（Library Cahce），另一部分是数据字典缓存（Data Dictionary Cache），在ORACLE中也叫ROW Cache。

Library Cache主要用于存储SQL语句、SQL语句相关的解析树、执行计划、PL/SQL程序块（包括匿名程序块、存储过程、包、函数等）以及它们转换后能够被Oracle执行的代码等,字典缓存的内容是按行（Row）存储的（其他数据通常按Buffer存储），所以又被称为Row Cache。

**（4）large pool**

在Oracle 8.0引入大池之前，所有内存分配都在共享池中进行。如果你使用的特性要利用“大块的”内存分配（如共享服务器UGA内存分配），倘若都在共享池中分配就不太好。

另外，与共享池管理内存的方式相比，处理（需要大量内存分配）会以不同的方式使用内存，所以这个问题变得更加复杂。共享池根据LRU来管理内存，这对于缓存和重用数

据很合适。不过，大块内存分配则是得到一块内存后加以使用，然后就到此为止，没有必要缓存这个内存。

**4 后台进程**

oracle中的进程共分为三类：用户进程、服务进程、后台进程。其中后台进程伴随实例的启动而启动，他们主要是维护数据库的稳定，相当于一个企业中的管理者及内部服务人员。他们并不会直接给用户提供服务。

**（1）DB Writer：**将Buffer Cache中的数据写入数据文件。

触发条件：（1）检查点（2）一个服务进程在设定的时间内没有找到空闲块(3)每三秒自动唤醒一次。

**（2）Log Writer：**将Log Buffer中的数据写入Redo Log中。

触发条件:（1）用户发出commit命令。（2）三秒定时唤醒。（3）log buffer超过1/3，或日志数量超过1M。（4）DBWR进程触发:DBWn视图将脏数据块写入磁盘先检测他的相关redo记录是否写入联机日志文件，如果没有就通知LGWR进程。

**（3）Check Point**维护数据库一致性状态。检测点时刻数据文件与SGA中的内容一致，这不是一个单独的进程，要和前两个进程一起工作呦。DBWR写入脏数据，同时触发LGWR进程。

**（4）Process Monitor:** 发现用户进程异常终止，并进行清理。释放占用资源。

**（5）System Monitor:**

1）负责实例恢复，前滚(Roll Forward)恢复到实例关闭的状态，使用最后一次检查点后的日志进程重做。这时包括提交和未提交的事务。打开数据库，进行回滚(Roll Back):回滚未提交的事务。(oracle承诺commit之后数据不会丢失，现在我们可以大致的了解是如何实现这个承诺，以及在数据的安全性和数据库性能之间的平衡选择。)

2）负责清理临时段，以释放空间。

## SQL分类

**1、数据定义语言 DDL**  
有 create alter drop

**2、数据操纵语言 DML**  
insert select delete update

**3、事务控制语言 TCL**  
commit savepoint rollback

**4、数据控制语言 DCL**  
grant revoke

## SQL基础

**1 谈谈对oracle的row\_ id是否理解?请简述?**

ORACLE的row\_id是一个伪列，其个是为18个字节可将这18个字节用6363来划分，分别表示段编号，数据文件编号，数据块编号和记录编号。Row\_id表示的是一个记录的物理存储地址。

**标识表中每条数据实际的物理地址。**

**可以用它来检索数据。有时繁杂的数据检索时，普通检索条件不能达到要求，可以利用rowid来精确检索的结果。一般在子查询的时候使用比较多。**

**2 DML语句，包含insert、update、delete、select等常用语句**

（1）增删改操作

新增：insert into book(id,name) values(2,"Wade");

批量新增：insert into book(id,name) select t.id,t.nameCn from book\_change;

修改：update book set name="McGrady" where id=2;

删除：delete book where id=2

全部删除(低效)：delete book

全部删除(高效)：Truncate table book

**employee雇员信息表**

emp\_id 雇员编号 NOT NULL NUMBER(4)

emp\_name 雇员姓名 VARCHAR2(10)

emp\_station 工作岗位 VARCHAR2(9)

emp\_managerId 该雇员经理人的编号 NUMBER(4)

emp\_inDate 入职时间 DATE

emp\_salary 薪水 NUMBER(7,2)

emp\_allowance 津贴 NUMBER(7,2)

emp\_depId 雇员所在部门编号 NUMBER(2)

departmentà部门信息表

dep\_id 部门编号 NOT NULL NUMBER(2)

dep\_name 部门名字 VARCHAR2(14)

dep\_loction 部门所在地 VARCHAR2(13)

salarygradeà薪水等级表

Sg\_id 薪水等级 NUMBER

Sg\_minSal 该等级的最低薪水值 NUMBER

Sg\_maxSal 该等级的最高薪水值 NUMBER

●基本查询(消除重复或字段)

select distinct \* from employee;

select emp\_id, emp\_name, emp\_salary from employee;

●查询算术表达式语句

select emp\_id, emp\_salary\*2 from employee;

select 55/2 from dual;

select sysdate from employee where emp\_id = 3;

select emp\_name,emp\_salary\*12+emp\_allowance from employee

●别名

select emp\_id, emp\_salary\*2 ‘doubleSal’from employee;

select em.emp\_id, em.emp\_salary\*2 as‘doubleSal’from employee em;

●where过滤数据

select \* from employee where emp\_depId = 3;

Select \* from employee where emp\_name = 'Wade' and emp\_depId = 3;

Select emp\_name, emp\_salary from employee where emp\_salary between 800 and 1500;

Select emp\_name, emp\_salary from employee where emp\_salary <= 800 or emp\_salary >=1500;

Select \* from employee where emp\_allowance is not null;

Select \* from employee where emp\_allowance is in(100,200,400,800,1000);

select \* from t\_application\_info where app\_id in (select t.bussines\_id from t\_profile t);

select \* from t\_application\_info app where exists (select 1 from t\_profile t where t.bussines\_id =app.app\_id );

Select \* from employee where emp\_name is not in('Wade','McGrady','James');

Select \* from employee where emp\_inDate > '20-2月-2010';

Select \* from employee where emp\_inDate between to\_date('2012-01-01','yyyy-mm-dd') and sysdate;

Select \* from employee where emp\_name like‘%Wade%’escape '\'; 包含或等于Wade,%代表多个字符

Select \* from employee where emp\_name like‘\_W%’escape '\'; 第二个字母为w，\_代表一个字符

Select \* from employee where emp\_name like '%\%%' escape '\' or emp\_name like '%\\_%' escape '\'; 查找包含%或\_的词

**in 和 exists的区别:** **如果子查询得出的结果集记录较少，主查询中的表较大且又有索引时应该用in, 反之如果外层的主查询记录较少，子查询中的表大，又有索引时使用exists**。其实我们区分in和exists主要是造成了驱动顺序的改变(这是性能变化的关键)，如果是exists，那么以外层表为驱动表，先被访问，如果是IN，那么先执行子查询，所以我们会以驱动表的快速返回为目标，那么就会考虑到索引及结果集的关系了 ，另外IN时不对NULL进行处理。in 是把外表和内表作hash 连接，而exists是对外表作loop循环，每次loop循环再对内表进行查询。一直以来认为exists比in效率高的说法是不准确的。

**not in 和not exists**

如果查询语句使用了not in 那么内外表都进行全表扫描，没有用到索引；而not extsts 的子查询依然能用到表上的索引。**所以无论那个表大，用not exists都比not in要快。**

●order by 语句

Select emp\_name, emp\_salary from employee order by emp\_name,emp\_salary,emp\_inDate;

Select emp\_name, emp\_salary from employee order by emp\_salary desc;

●group by语句

Select emp\_depId , avg(emp\_salary) as ‘平均工资’，max(sal) as ‘最大工资’ from employee group by emp\_depId having max(sal) > 2000;

Select emp\_depId ,avg(emp\_salary) from employee where emp\_salary > 1200 group by emp\_depId having avg(emp\_salary) > 1500 Order by avg(emp\_salary);

使用group by 的规律：

出现在select列表中的字段，如果没有出现在组函数中，则必须出现在group by子句中。

●表的连接

select \* from t\_application\_info ap,t\_profile t where ap.app\_id = t.bussines\_id; --自然连接

select \* from t\_application\_info ap join t\_profile t on ap.app\_id = t.bussines\_id; --等效自然连接，写法不一样而已

select \* from t\_application\_info ap left join t\_profile t on ap.app\_id = t.bussines\_id; --左连接，以左表为基准，右表为空的记录页查询出来

select \* from t\_application\_info ap right join t\_profile t on ap.app\_id = t.bussines\_id; --右连接，以右表为基准，左表为空的记录页查询出来

select \* from t\_application\_info ap full join t\_profile t on ap.app\_id = t.bussines\_id; --全连接，左表和右表为空的记录页查询出来

select \* from t\_application\_info ap where ap.app\_id in(select distinct(t.bussines\_id) from t\_profile t); --类似自然连接，但只能查询左表的属性

select \* from t\_application\_info ap left join (select \* from t\_application\_info ap,t\_profile t where ap.app\_id = t.bussines\_id) w1

on ap.app\_id = w1.bussines\_id --多表连接，原理是把一个或若干个连接的结果当做一个表再与外表进行连接

select \* from t\_application\_info ap, (select \* from t\_profile t1 where t1.bussines\_id is not null) t

where ap.app\_id = t.bussines\_id; --原理是把一个或若干个连接的结果当做一个表在与外表自然连接

select w2.\* from (select w1.\*, rownum mynum from (select \* from t\_application\_info) w1 where rownum <= 100) w2 where mynum >= 20 --分页语句

**关于多表连接**

SELECT \*

FROM ACT\_RU\_TASK art

INNER JOIN ACT\_RE\_PROCDEF arp

ON art.proc\_def\_id\_ = arp.id\_

AND arp.key\_ = 'shop\_data\_input\_flow'

LEFT JOIN ACT\_HI\_TASKINST aht ON art.ID\_ = aht.ID\_

LEFT JOIN ACT\_RU\_TASK\_EXT arte ON art.ID\_ = arte.task\_id

LEFT JOIN ACT\_RE\_ACTDEF\_EXT arae ON arp.KEY\_ = arae.process\_define\_key AND art.TASK\_DEF\_KEY\_ = arae.act\_id

LEFT JOIN erp\_shop\_data\_input esdi ON art.proc\_inst\_id\_ = esdi.proc\_ins\_id

LEFT JOIN erp\_order\_original\_info eooi

ON (esdi.order\_id = eooi.id

AND eooi.cancel = 0

AND eooi.del\_flag = 0)

LEFT JOIN sdi\_flow sf

ON (esdi.order\_id = sf.order\_id

AND sf.del\_flag = 0

AND sf.proc\_ins\_id = art.PROC\_INST\_ID\_)

LEFT JOIN erp\_shop\_info esi ON esdi.shop\_id = esi.zhangbei\_id AND esi.del\_flag = 0

WHERE art.ASSIGNEE\_ = '130419632037907607';

**分析：三者都是笛卡尔积，匹配上的数目相乘，如果要保证以一边为主，确认另一边能够匹配上的只有1 ，否则数据翻倍。**

**所谓的左连接，右连接是指另一边匹配不上的数据不查出来，这边没有关联的空显示；和数据没有关系，如果另一边不能保证1，还是会翻倍的。**

**上面的SQL ACT\_RU\_TASK为主体，后面的表 能够保证有一个能够匹配。**

1 A 1 C

2 B 3 D

left join

1 A C

2 B

right join

1 A C

3 D

inner join（full join）

1 A C

2 B

3 D

●子查询(理解子查询的关键-->把它当成一张表)

Select emp\_name,emp\_salary from employee where emp\_salary =(Select max(emp\_salary) from employee);

按照部门进行分组之后挣钱最多的那个人的名字，部门编号？

Select emp\_salary,emp\_name, emp\_deptId from employee e , (Select max(emp\_salary) as‘maxSalary’,emp\_deptId from employee group by emp\_deptId) t where e.emp\_salary = t.maxSalary and e.emp\_deptId = t.emp\_deptId;

求每个部门的平均薪水等级是多少？

Select t.avgSalary, t.emp\_deptId, s.sg\_id from salarygrade s, (Select avg(emp\_salary) as‘avgSalary’,emp\_deptId from employee group by emp\_deptId) t where t.avgSalary between s.sg\_minSal and s.sg\_maxSal group by emp\_deptId.

**（2）表的连接**

select \* from t\_application\_info ap,t\_profile t where ap.app\_id = t.bussines\_id; --自然连接

select \* from t\_application\_info ap join t\_profile t on ap.app\_id = t.bussines\_id; --等效自然连接，写法不一样而已

select \* from t\_application\_info ap left join t\_profile t on ap.app\_id = t.bussines\_id; --左连接，以左表为基准，右表为空的记录页查询出来

select \* from t\_application\_info ap right join t\_profile t on ap.app\_id = t.bussines\_id; --右连接，以右表为基准，左表为空的记录页查询出来

select \* from t\_application\_info ap full join t\_profile t on ap.app\_id = t.bussines\_id; --全连接，左表和右表为空的记录页查询出来

select \* from t\_application\_info ap where ap.app\_id in(select distinct(t.bussines\_id) from t\_profile t); --类似自然连接，但只能查询左表的属性

select \* from t\_application\_info ap left join

(select \* from t\_application\_info ap,t\_profile t where ap.app\_id = t.bussines\_id) w1

on ap.app\_id = w1.bussines\_id --多表连接，原理是把一个或若干个连接的结果当做一个表再与外表进行连接

select \*

from t\_application\_info ap,

(select \* from t\_profile t1 where t1.bussines\_id is not null) t

where ap.app\_id = t.bussines\_id; --原理是把一个或若干个连接的结果当做一个表在与外表自然连接

select w2.\*

from (select w1.\*, rownum mynum

from (select \* from t\_application\_info) w1

where rownum <= 100) w2

where mynum >= 20 --分页语句

//批量新增数据SQL语句

insert into t\_profile

select SEQ\_APPLICATION\_INFO.Nextval id,

buid.id bussines\_id,

th.profile\_ids,

th.profile\_names,

th.id,

th.profile\_name,

th.profile\_type,

buid.type,

th.language

from (select id,'catalog' type

from (select b.id

from w3cn.catalog b

start with b.id = '4517'

connect by nocycle prior b.id = b.parentid) c1

where c1.id != '4517'

union all

select d.web\_doc\_id id,'document' type

from w3cn.documentation d

where d.cid in (

select \*

from (select b.id ID

from w3cn.catalog b

start with b.id = '4517'

connect by nocycle prior b.id = b.parentid) c1

where c1.id != '4517')

and d.doc\_status = 2) buid,

(select \* from t\_profile p where p.bussines\_id = '4517') th

**（3）JDBC的批量新增逻辑**

boolean isOperateing = false;

String sqlStr = " insert into w3en.t\_profile(id,language)values (?, ?)";

if(isOperateing)

{

return false;

}

PreparedStatement sm = null;

for (Object object : items)

{

UserProfileVO item = (UserProfileVO)object;

sm.setString(1, item.getId());

sm.setString(2, null==item.getBussines\_id()?"":item.getBussines\_id());

sm.addBatch();

}

sm.executeBatch();

isOperateing = true;

**（4）找到有重复的数据**

select a.timestamp, a.userid, a.start\_datetime

from t\_huawei\_log\_events a

where rowid > (select min(rowid)

from t\_huawei\_log\_events b

where a.timestamp = b.timestamp

and a.start\_datetime = b.start\_datetime

)

**（5）删除**

delete from t\_huawei\_log\_events u where to\_number(substr(timestamp,0,8)) < 20110901

**2.DDL语句，包含create、alter、truncate、drop等常用语句**

(1)模式对象：用户数据是以对象的形式存在的，并以模式的单位进行组织

(2)表、视图、索引、索引化表、序列、同义词、PL/SQL包、存储过程、存储函数、JAVA类属于模式对象；

表空间、用户、角色、回退段、概要文件不属于模式对象；

(3)表

创建普通表：

create table book

(

id number(10) not null,

name varchar2(50) default ‘wade’ ,

shopId number(10) not null constraint book\_shopId\_fk refrences shop(id),

produceDate date default sysdate,

constraints pk1 primary key(id, produceDate)

)

利用子查询创建表：

a:可以修改列的名称，不能修改数据类型

b:子查询返回数据则新标包含查到的数据，否则为一个空表

create table t2(no,name,dept)

as

select e.empno,e.name,e.deptno from emp e;

**（4）数据类型表**

|  |  |  |
| --- | --- | --- |
| **数据类型** | **参数** | **描述** |
| char(n) | n=1 to 2000字节 | 定长字符串，n字节长，如果不指定长度，缺省为1个字节长（一个汉字为2字节） |
| varchar2(n) | n=1 to 4000字节 | 可变长的字符串，具体定义时指明最大长度n， 这种数据类型可以放数字、字母以及ASCII码字符集(或者EBCDIC等数据库系统接受的字符集标准)中的所有符号。 如果数据长度没有达到最大值n，Oracle 8i会根据数据大小自动调节字段长度， 如果你的数据前后有空格，Oracle 8i会自动将其删去。VARCHAR2是最常用的数据类型。 可做索引的最大长度3209。 |
| number(m,n) | m=1 to 38 n=-84 to 127 | 可变长的数值列，允许0、正值及负值，m是所有有效数字的位数，n是小数点以后的位数。 如：number(5,2)，则这个字段的最大值是99,999，如果数值超出了位数限制就会被截取多余的位数。 如：number(5,2)，但在一行数据中的这个字段输入575.316，则真正保存到字段中的数值是575.32。 如：number(3,0)，输入575.316，真正保存的数据是575。 |
| date | 无 | 从公元前4712年1月1日到公元4712年12月31日的所有合法日期， Oracle 8i其实在内部是按7个字节来保存日期数据，在定义中还包括小时、分、秒。 缺省格式为DD-MON-YY，如07-11月-00 表示2000年11月7日。 |
| long | 无 | 可变长字符列，最大长度限制是2GB，用于不需要作字符串搜索的长串数据，如果要进行字符搜索就要用varchar2类型。 long是一种较老的数据类型，将来会逐渐被BLOB、CLOB、NCLOB等大的对象数据类型所取代。 |
| raw(n) | n=1 to 2000 | 可变长二进制数据，在具体定义字段的时候必须指明最大长度n，Oracle 8i用这种格式来保存较小的图形文件或带格式的文本文件，如Miceosoft Word文档。 raw是一种较老的数据类型，将来会逐渐被BLOB、CLOB、NCLOB等大的对象数据类型所取代。 |
| long raw | 无 | 可变长二进制数据，最大长度是2GB。Oracle 8i用这种格式来保存较大的图形文件或带格式的文本文件，如Miceosoft Word文档，以及音频、视频等非文本文件。 在同一张表中不能同时有long类型和long raw类型，long raw也是一种较老的数据类型，将来会逐渐被BLOB、CLOB、NCLOB等大的对象数据类型所取代。 |
| blob clob nclob | 无 | 三种大型对象(LOB)，用来保存较大的图形文件或带格式的文本文件，如Miceosoft Word文档，以及音频、视频等非文本文件，最大长度是4GB。 LOB有几种类型，取决于你使用的字节的类型，Oracle 8i实实在在地将这些数据存储在数据库内部保存。 可以执行读取、存储、写入等特殊操作。 |
| bfile | 无 | 在数据库外部保存的大型二进制对象文件，最大长度是4GB。 这种外部的LOB类型，通过数据库记录变化情况，但是数据的具体保存是在数据库外部进行的。 Oracle 8i可以读取、查询BFILE，但是不能写入。 大小由操作系统决定。 |

（5）Oracle 基本知识

一个表空间只能属于一个数据库。

每个数据库最少有一个控制文件（建议3个，分别放在不同的磁盘上）。

每个数据库最少有一个表空间（SYSTEM表空间）建立SYSTEM表空间的目的是尽量将目的相同的表存放在一起，以提高使用效率，只应存放数据字典。

每个数据库最少有两个联机日志组，每组最少一个联机日志文件。

一个数据文件只能属于一个表空间。

一个数据文件一旦被加入到一个表空间中，就不能再从这个表空间中移走，也不能再加入到其他表空间中。

建立新的表空间需要建立新的数据文件。

数据文件被ORACLE格式化为ORACLE块，Oracle9i以前版本中，ORACLE块的大小是在第一次创建数据库时设定的，

并且以后不能改变，要想改变，只能重建数据库。

一个段segment只能属于一个表空间，但可以属于多个数据文件

一个区extent只能属于一个数据文件，即区间（extent）不能跨越数据文件

（6）一个块的最大长度为16KB(有2K、4K、8K、16K)

　　每个数据库最大文件数（按块大小）

　　2K块　　　　　　　　20000个文件

　　4K块　　　　　　　　40000个文件

　　8K块或以上　　　　　65536个文件

（7）oracle server可以同时启动多个数据库

　　一套操作系统上可以安装多个版本的ORACLE数据库系统（UNIX可以，NT不可以）

　　一套ORACLE数据库系统中可以有多个ORACLE数据库及其相对应的实例

　　每个ORACLE数据库拥有一个数据库实例（INSTANCE）（OPS除外）

　　所以，一套操作系统上同时可以有多个oracle数据库实例启动

如何给表、列加注释？

SQL>comment on table 表 is '表注释';

注释已创建。

SQL>comment on column 表.列 is '列注释';

注释已创建。

SQL> select \* from user\_tab\_comments where comments is not null;

SQL> select \* from user\_col\_comments where comments is not null;

**3 创建临时表**

约束的种类

表级约束--同时约束多个列名

列级约束--约束一个列名，跟随在列名之后

主键约束--唯一、非空、一个表只有一个主键，但一个主键可以包含多个字段，同时建立一个唯一索引

id number(20) primary key

constraint pk1 primary key (id)

alter table book add constraint pk1 primary key (id)

alter table drop constraint pk1

唯一性约束--建立一个唯一索引，如果未定义非空约束，则可以包含多个空值

id number(20) unique

alter table book add constraint puk unique(name)

alter table book drop unique(name)

alter table book constraint puk

外键约束--取值要么是外表对应的值，要么为空

shopId number(10) constraint fk1 refrences shop(id)

constraint fk1 foreign key (shopId) refrences shop(id)

alter table book add constraint fk1 foreign key (shopId) refrences shop(id) on delete cascade/set null/resttricted

cascade/set null/resttricted: 删除字表相关的数据/设为空值/受限删除

alter table book drop constraint fk1

非空约束--列级约束

alter table book modify name not null

alter table book modify name null

检查约束--限制列值所允许的取值范围的

age number(20) check(age between 20 and 30)

alter table book add constraint fc1 check(age between 20 and 30)

alter table book drop constraint fc1

禁用和激活约束

alter table book disable unique(name)

alter table book modify constraint puk disable

alter table book enable unique(name)

alter table book modify constraint puk enable

查询book表所有约束信息

select \* from user\_constraints where table\_name = 'book'

表的操作

alter table book add(id number(20),author varchar(20) not null)

alter table book modify author number(40)

alter table book drop column author

alter table book modify name default ‘McGrady’

alter table book rename column author to author2

alter table book rename to book2

comment on table book is '书籍信息表'

comment on column book.name is '书名 '

drop table book

truncate table book

常见的系统表

user\_constraints-----约束表

dual-----哑表

user\_indexes-----索引表

all\_users-----包含数据库所有的用户的用户名，ID，创建时间

dba\_users-----包含数据库所有的用户的详细信息

user\_users-----包含当前用户的详细信息

v$open\_cursor-----包含用户执行的SQL语句信息

dba\_tab\_privs-----包含数据库所有对象的授权信息

user\_tab\_privs-----包含当前用户的授权信息

dba\_roles-----所有角色及其描述

**4 索引**

**（1）基础知识**

@ 提高表的查询效率，建立在表列上的数据库对象，有独立的物理和逻辑结构

@ 如果没有建立索引，则进行全表扫描，否则索引记录了数据的物理地址，即rowid,根据rowid可以很快找到数据。

@ 唯一索引--索引值不重复的索引

@ 位图索引--每一个索引值建立一个位图

@ 复合索引--多个列上建立索引

@ 优点：增加查询速速，特别是连接查询

@ 缺点：作为一个独立的模式对象存在，需要占用存储空间和间期进行维护

create index index1 on book(id)

create unique index index1 on book(id)

create index index1 on book(id,name, produceDate)

drop index index1

alter index iudex1 rename to index2

select \* from user\_indexes

**（2）从数据结构角度**

1、B+树索引(O(log(n)))：关于B+树索引，可以参考 MySQL索引背后的数据结构及算法原理

2、hash索引：

a 仅仅能满足"=","IN"和"<=>"查询，不能使用范围查询

b 其检索效率非常高，索引的检索可以一次定位，不像B-Tree 索引需要从根节点到枝节点，最后才能访问到页节点这样多次的IO访问，所以 Hash 索引的查询效率要远高于 B-Tree 索引

c 只有Memory存储引擎显示支持hash索引

3、FULLTEXT索引（现在MyISAM和InnoDB引擎都支持了）

4、R-Tree索引（用于对GIS数据类型创建SPATIAL索引）

**（3）从物理存储角度**

1、聚集索引（clustered index）

2、非聚集索引（non-clustered index）

**（4）从逻辑角度**

1、主键索引：主键索引是一种特殊的唯一索引，不允许有空值

2、普通索引或者单列索引

3、多列索引（复合索引）：复合索引指多个字段上创建的索引，只有在查询条件中使用了创建索引时的第一个字段，索引才会被使用。使用复合索引时遵循最左前缀集合

4、唯一索引或者非唯一索引

5、空间索引：空间索引是对空间数据类型的字段建立的索引，MYSQL中的空间数据类型有4种，分别是GEOMETRY、POINT、LINESTRING、POLYGON。

MYSQL使用SPATIAL关键字进行扩展，使得能够用于创建正规索引类型的语法创建空间索引。创建空间索引的列，必须将其声明为NOT NULL，空间索引只能在存储引擎为MYISAM的表中创建

CREATE TABLE table\_name[col\_name data type]

[unique|fulltext|spatial][index|key][index\_name](col\_name[length])[asc|desc]

**（5）索引什么优缺点？**

（1）创建索引可以大大提高系统的性能:

第一，通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。

第二，可以大大加快数据的检索速度，这也是创建索引的最主要的原因。

第三，可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。

第四，在使用分组和排序 子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。

第五，通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

（2）增加索引也有许多不利的方面:

第一，创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。

第二，索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。

第三，当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。

**（6）建立索引规则，什么情况应该建立，什么情况不应该？**

数据库建立索引常用的规则如下：

1、表的主键、外键必须有索引；

2、数据量超过300的表应该有索引；

3、经常与其他表进行连接的表，在连接字段上应该建立索引；

4、经常出现在Where子句中的字段，特别是大表的字段，应该建立索引；

5、索引应该建在选择性高的字段上；

6、索引应该建在小字段上，对于大的文本字段甚至超长字段，不要建索引；

7、复合索引的建立需要进行仔细分析；尽量考虑用单字段索引代替：

A、正确选择复合索引中的主列字段，一般是选择性较好的字段；

B、复合索引的几个字段是否经常同时以AND方式出现在Where子句中？单字段查询是否 极少甚至没有？如果是，则可以建立复合索引；否则考虑单字段索引；

C、如果复合索引中包含的字段经常单独出现在Where子句中，则分解为多个单字段索引；

D、如果复合索引所包含的字段超过3个，那么仔细考虑其必要性，考虑减少复合的字段；

E、如果既有单字段索引，又有这几个字段上的复合索引，一般可以删除复合索引；

8、频繁进行数据操作的表，不要建立太多的索引；

9、删除无用的索引，避免对执行计划造成负面影响；

10、BLOB，CLOB字段肯定也不适合建索引。

11、索引不会包含有NULL值的列

12、般情况下不鼓励使用like操作，如果非使用不可，如何使用也是一个问题。like “a%” 不会使用索引而like “aaa%”可以使用索引。

**5 索引化表---是一种特殊的表，**将表的数据和索引的数据存储在一起，主要是和经常通过主键查询整个记录或部分记录的情况。

create table book

(

id number(10) not null,

name varchar2(50) default ‘wade’ ,

shopId number(10) not null constraint book\_shopId\_fk refrences shop(id),

produceDate date default sysdate,

constraints pk1 primary key(id, produceDate)

)

organization index;

**6 视图--一个或多个表或视图中提取出来的一种表现形式**，是一个虚表，通过视图可以提高数据的安全性，隐藏数据的复杂性，简化查询语句，保存复杂查询，分离引用程序和基础表。

create or replace view v1 as select \* from book where id <200 with read only （只能查询）

select \* from v1

drop view v1

可以对视图进行DML操作，前提是没有with read only，但是出现以下的关键字时不能操作

@集合操作符（union,union all,intersect,minus）

@聚集函数（avg,sum,count,max）

@group by,connect by或start with，distinct等语句

**7 序列--产生唯一序号的数据库对象**吗，一般为主键，长度为38位的十进制数

create sequence s1 increment by 1 start with 1 maxvalue 1000000;

select s1.currval from dual;

select s1.nextval from dual;

insert into book(id,name) values(s1.nextval,'james');

drop sequence s1

**8 分区表和分区索引**

@随着数据库技术的飞速发展，数据库数据容量越来越大，表中数据达到GB甚至是TB的十分普遍，如何对这些海量数据进行管理是数据库的难题。分区技术是ORACLE对巨型表或巨型索引进行维护和管理的重要技术。

@将一个巨型表或巨型索引分成若干个独立的组成部分进行存储和管理叫分区技术。

create table student\_rang

(

sno number(20) primary key,

name varchar2(50),

birth date

)

partition by rang/list/hash(birth)

(

partition p1 values less then to\_date('2000-01-01','yyyy-mm-dd') tablespace oracle1,

partition p2 values less then to\_date('2005-01-01','yyyy-mm-dd') tablespace oracle2,

partition p3 values less then to\_date('2010-01-01','yyyy-mm-dd') tablespace oracle3,

);

create index i1 on p1(name) local;

**9 同义词-----数据库中表、视图、索引或其他模式对象的别名**

A:可以保证对象的安全性（隐藏实际名称）

B:可以简化对象访问

C:数据对象改变时，可以修改同义词而不要修改应用程序

create public synonym book\_syn for czj.book;

update book\_syn set name = 'wade' where id = 34500 ;

drop synonym book\_syn;

**10 数据库连接**

分布式系统数据库环境一个数据库与另一个数据库之间的通信途径，将远程数据库映射到本地，所有可以访问本地数据库的应用程序都可以操作远程

create database link d\_link connect to czj identified by wade399 using 'w3en' ;

update emp@d\_link set name = 'wade' where id = 34500

**11 TCL语句，commit、rollback等常用语句**

declare

v\_idDelete number(10):=3;

v\_idUpdate number(10):=2;

v\_name varchar(20):="James";

begin

delete book where id= v\_idDelete;

update book set name=v\_name where id = v\_idUpdate;

commit;

exception

when others then

rollback;

end;

/

java代码里面使用：

Connection con = null;

try

{

con = DriverManager.getConnection(“jdbc:oracle:thin:@127.0.0.1:1521:MyOrcl”, ”czj”,”cc”);

con.setAutoCommit(false);

addBook(book);

addCostumer(cos);

updateSum(1);

con.commit();

}

catch(Exception ex)

{

con.rollback();

throw new RuntimeException(“Sorry! System error!”,ex);

}

finally

{

con.close();

}

**12 DCL语句，invoke、voke主要用于权限的分配与回收.(见第四部分)**

**13 SQL语句优化：**

1 使用索引

2 使用表的别名

3 只查询需要的列，不要select \*

4 in关键字清晰易懂，但是效率不高，建议用exists代替，not in用not exists代替

5 is null或is not null 是不会引用索引的，因为索引是不会索引空值的，建议用其他功能 相似的关键字

6 >或<操作符，>2和>=3的效果完全不一样，>2会先找到=2的记录在进行比较，而>=3的会直 接找到=3的记录

7 用union all 代替union，union不仅会把两个结果合而且会去掉重复的排序，union all仅 仅是合并

8 from多个表时候，把数据量最少的表写在最后面，首先扫描；

9 where字句采用自下而上的顺序依次过滤，所以把表的连接条件写在最前面，那些可以过滤 掉最大数量的条件写在末尾

10 用where字句代替having子句，having只会在检索出所有记录之后才会进行过滤

## SQL调优

**1 SQL语句的执行步骤**

1）语法分析，分析语句的语法是否符合规范，衡量语句中各表达式的意义。

2）语义分析，检查语句中涉及的所有数据库对象是否存在，且用户有相应的权限。

3）视图转换，将涉及视图的查询语句转换为相应的对基表查询语句。

4）表达式转换， 将复杂的 SQL 表达式转换为较简单的等效连接表达式。

5）选择优化器，不同的优化器一般产生不同的“执行计划”

6）选择连接方式， ORACLE 主要有三种连接方式，对多表连接ORACLE会选择适当的连接方式。

7）选择连接顺序， 对多表连接 ORACLE 选择哪一对表先连接，选择这两表中哪张表做为基础数据表。

8）选择数据的搜索路径，根据以上条件选择合适的数据搜索路径，比如，是选用全表搜索还是利用索引或是其他的方式。

9）运行“执行计划”

**2 我们可以通过如下语句来查询缓存中的执行计划：**

SELECT t1.\*,

't2-->',

t2.\*

FROM v$sql\_plan t1

JOIN v$sql t2

ON t1.address = t2.address

AND t1.hash\_value = t2.hash\_value

AND t1.child\_number = t2.child\_number;--缓存中的执行计划。

**3 典型SELECT语句完整的执行顺序**

1）from子句组装来自不同数据源的数据；

2）where子句基于指定的条件对记录行进行筛选；

3）group by子句将数据划分为多个分组；

4）使用聚集函数进行计算；

5）使用having子句筛选分组；

6）计算所有的表达式；

7）计算select的字段；

8）使用order by对结果集进行排序。

**4 SQL语句执行过程**

如下图所示：

说明：

\*这是一张SQL语句执行过程图

\*执行计划是SQL语句执行过程中必然用到的

\*执行计划是优化器（Optimizer）的产物

\*两种不同的方式：CBO和RBO

**5 查看优化器设置**

（1）方法一

SELECT VALUE FROM v$parameter t WHERE t.name = 'optimizer\_mode';

（2）方法二(SQLPLUS下执行)：

showparameter optimizer\_mode

\*CBO用到了字典中的Statistics,而RBO没有

分析统计信息相关SQL：

（3）

analyze table tablename compute statistics;

（4）

analyze table tablename compute statistics for all indexes

（5）

analyze table tablename delete statistics

**6 SQL优化方法论**

\*ORACLE10g以后的版本，SQL优化的本质是基于对CBO和执行计划的深刻理解，进入CBO时代，一定要理解执行计划。

\*查看执行计划有好多方式，比如使用PL/SQL Developer工具，选中select语句，按F5键就可以显示其执行计划，不过显示的不完全

\*最好使用在Oracle官方的sqlplus工具，性能最好，方便直观，下面介绍两种查看执行计划方式（也是最简单的两种方式）

**7 关于执行计划的一些知识：**

\* Full Table Scans 全表扫描

\* Rowid Scans rowid扫描

\* Index Scans 索引扫描

\* Index Unique Scans

\* Index Range Scans

\* Index Range Scans Descending

\* Index Skip Scans

\* Full Scans

\* Fast Full Index Scans（CBO）

\* Index Joins

\* Bitmap Joins

\* Cluster Scans 簇扫描

\* Hash Scans 散列扫描

\* Sample Table Scans 表取样扫描

²在RBO时代，关于access path，很简单，有index就用，而对于join方法，编程人员一般会通过调整关联表之间的先后顺序来获得比较好的运行结果。有什么缺点呢？

²有了CBO，简单就是两个字-----CBO走的是包办婚姻：你的事交给我办。

ORACLE默认情况下，周一到周五每天晚上10点到第二天早上6点以及整个周末期间会自动收集统计信息

**8 可以查看参数：**

show parameter STATISTICS\_LEVEL

²问题：CBO执行计划依赖的statistic不准确（缺失或者太旧），导致在计算执行成本时就会出现偏差，很可能会产生错误的执行计划，怎么办呢？

第一步：重新收集统计信息！

第二部：第一部解决不了的情况下，使用Hints

**9 合理应用Hints**

1. Hints

**Oracle Hints是一种机制，用来告诉优化器按照我们的告诉它的方式生成执行计划，而不是根据它自己的分析。**

慎用hint，可能会产生严重的后果，比如append会产生锁块，导致并发资源等待等

Hints的分类：

\*Hints forOptimization Approaches and Goals(4)

/\*+ ALL\_ROWS \*/

/\*+ FIRST\_ROWS ( n )\*/

/\*+ CHOOSE \*/

/\*+ RULE \*/

\*Hints for AccessPaths(12)

/\*+ FULL ( table ) \*/

/\*+ INDEX ( tableindex) \*/

/\*+ INDEX\_ASC ( tableindex) \*/

/\*+ INDEX\_COMBINE (table index) \*/

/\*+ INDEX\_JOIN (table index) \*/

/\*+ INDEX\_DESC (table index) \*/

/\*+ INDEX\_FFS ( tableindex) \*/

/\*+ NO\_INDEX ( tableindex) \*/

/\*+ AND\_EQUAL ( tableindex index) \*/

\*Hints for QueryTransformations(10)

\*Hints for JoinOrders(2)

\*Hints for JoinOperations(11)

/\*+ USE\_NL ( table )\*/

/\*+ USE\_MERGE ( table) \*/

/\*+ USE\_HASH ( table) \*/

/\*+ LEADING ( table )\*/

\*Hints for ParallelExecution(5)

\*Additional Hints(13)

（2）以下为使用Hints的例子

create table t\_1(owner varchar2(30),table\_name varchar2(30));

create table t\_2(owner varchar2(30),table\_name varchar2(30));

insert into t\_1 SELECT owner,table\_name FROM dba\_tables;

insert into t\_2 SELECT owner,view\_name FROM dba\_views t;

create index idx\_t\_1 on t\_1(table\_name);

create index idx\_t\_2 on t\_2(table\_name);

analyze table t\_1 compute statistics;

analyze table t\_2 compute statistics;

SELECT \*

FROM (SELECT \* FROM t\_1

UNION ALL

SELECT \* FROM t\_2) aa

WHERE aa.table\_name LIKE 'Z%'; ---- Full Table Scans

SELECT /\*+ index(AA.t\_1 idx\_t\_1) index(AA.t\_2 idx\_t\_2)\*/ \*

FROM (SELECT \* FROM t\_1

UNION ALL

SELECT \* FROM t\_2) AA

WHERE AA.table\_name LIKE 'Z%'; ---- Index Scans

**10 索引及应用实例**

（1）什么是索引

\*Oracle的索引是一种自平衡的B\*Tree存储结构，其基本存储单位为数据块，称之为节点，共有三种类型的节点：根（root）节点，分枝（Branch）节点，叶（leaf）节点。

\*分枝节点存储{索引值，键值对应下一级节点块地址，lmc指针}

\*叶节点存储{索引值及其rowid，当前节点的前后节点的数据块地址}

所有叶节点上的两个指针形成一个双向链表，在这个双向链表上的所有索引值，从小到大排列，而对于倒序desc索引，则是从大到小排列

（2）索引分类

@逻辑上：

Single column 单列索引

Concatenated 多列索引

Unique 唯一索引

Non-Unique 非唯一索引

Function-based函数索引

Domain 域索引

@物理上：

Partitioned 分区索引

Non-Partitioned 非分区索引

@B\*tree：

Normal 正常型B树

ReverseKey 反转型B树

Bitmap 位图索引

（3）什么时候使用索引

\*如果要检索全表，不必要建索引，因为索引会带来额外的IO操作。

\*如果检索的记录数占全部表记录的10%以下可以考虑建索引(大表)。

\*表之间的关联字段可以考虑建索引，特别是一张大表和一张小表的关联。

\*B\*Tree索引适合于大量的增、删、改（OLTP）；

不适合用包含OR操作符的查询；一般不适用NULL判断；

适合高基数的列（重复值少）

\*Bitmap索引适合于决策支持系统OLAP；

做UPDATE代价比较高；会锁块；

非常适合OR操作符的查询；

适合低基数的列（比如，只有Y和N两种值） ；

\*Reverse索引反转了b\*tree索引码中的字节，是索引条目分配更均匀，多用于并行服务器环境下，用于减少索引叶的竞争。

索引是’双刃剑’，在查询与DML之间寻求平衡

（4）改写SQL使用索引

\*普通索引列 a is not null 按逻辑改为a>0或a>''

\*like操作改写

\*能用union all绝不用union，除非要去重

\*in操作虽然简单易懂，但oracle内部会转换为表连接查询，使用in会多一步转换操作，所以建议使用表关联查询

\*not in 强烈建议使用not exists或（外连接+判断为空）

\*<>（不等于）操作不走索引，推荐a<>0改为(a>0 ora<0) a<>’’改为a>’’

\*提防隐式类型转换， oracle内部处理a=0与a=‘0’是完全不同的，甚至会导致不走索引，这个深有体会，最近一个项目就是这个隐式类型转换出了问题，导致速度

很慢

（5）索引应用

例1.用合适的索引来避免不必要的全表扫

如果要在索引列查询is not null条件，建议列加上is not null约束，默认值约束，

然而确实由于某种原因索引列设计为null，还想通过is null条件走索引，该如何是好呢？请看

drop table t\_tab1;

create table t\_tab1 as

SELECT t.owner,

t.object\_name,

t.object\_type,

t.created,

t.last\_ddl\_time

FROM dba\_objects t;

analyze table t\_tab1 compute statistics;

create index idx01\_t\_tab1 on t\_tab1(last\_ddl\_time);--普通索引

set autotrace trace;

SELECT \* FROM t\_tab1 t where t.last\_ddl\_time is null;

如上情况调整为复合索引

drop index idx01\_t\_tab1;

create index idx01\_t\_tab1 on t\_tab1(last\_ddl\_time,1);--加了个常量

set autotrace trace;

SELECT \* FROM t\_tab1 t where t.last\_ddl\_time is null;

例2：用合适的函数索引来避免看似无法避免的全表扫描

drop table t\_tab1 purge;

create table t\_tab1 as

SELECT t.owner,

t.object\_name,

t.object\_type,

t.OBJECT\_ID,

t.created,

t.last\_ddl\_time

FROM dba\_objects t;

CREATE INDEX IDX01\_T\_TAB1 ON T\_TAB1(object\_name);

analyze table t\_tab1 compute statistics;

set autot trace

SELECT \* FROM t\_tab1 t where t.object\_name like '%20121231';

改进索引，此处使用反转函数索引，此外经常用到的函数索引还有，instr(),substr()等

drop index IDX01\_T\_TAB1;

CREATE INDEX IDX02\_T\_TAB1 ON T\_TAB1(reverse(object\_name));

analyze table t\_tab1 compute statistics;

SELECT \* FROM t\_tab1 t where reverse(t.object\_name) like reverse('%20121231');

**5.其他优化技术及应用**

（1）并行技术，并行执行目标SQL语句，这实际上是以额外的资源消耗来换取执行时间的缩短，很多情况下使用并行是针对某些SQL的唯一优化手段。

（2）使用shell调度或其他调度工具。

SQL语句级别的并行：/\*+parallel\*/

/\*+ parallel(table\_name 4)\*/

（3）表压缩技术

（4）Partition技术

（5）中间表/临时表事务分解思路

**6 SQL优化总结**

1.对查询进行优化，要尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。

2.应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描，如：

select id from t where num is null

最好不要给数据库留NULL，尽可能的使用 NOT NULL填充数据库.

3.应尽量避免在 where 子句中使用 != 或 <> 操作符，否则将引擎放弃使用索引而进行全表扫描。

4.应尽量避免在 where 子句中使用 or 来连接条件，如果一个字段有索引，一个字段没有索引，将导致引擎放弃使用索引而进行全表扫描，如：

select id from t where num=10 or Name = 'admin'

可以这样查询： select id from t where num = 10 union allselect id from t where Name = 'admin'

5.in和 not in 也要慎用，否则会导致全表扫描，如：

6.下面的查询也将导致全表扫描：

select id from t where name like ‘%abc%’

若要提高效率，可以考虑全文检索。

7.如果在 where 子句中使用参数，也会导致全表扫描。因为SQL只有在运行时才会解析局部变量，但优化程序不能将访问计划的选择推迟到运行时；它必须在编译时进行选择。然 而，如果在编译时建立访问计划，变量的值还是未知的，因而无法作为索引选择的输入项。如下面语句将进行全表扫描：

select id from t where num = @num

可以改为强制查询使用索引： select id from t with(index(索引名)) where num = @num

8.应尽量避免在 where子句中对字段进行表达式操作，这将导致引擎放弃使用索引而进行全表扫描。如：

select id from t where num/2 = 100

应改为:

select id from t wherenum = 100\*2

9.应尽量避免在where子句中对字段进行函数操作，这将导致引擎放弃使用索引而进行全表扫描。如：

select id from t where substring(name,1,3) = ’abc’ -–name以abc开头的id

select id from t where datediff(day,createdate,’2005-11-30′) = 0 -–‘2005-11-30’ --生成的id

应改为:

select id from t where name like 'abc%'

select id from t where createdate >= '2005-11-30' and createdate < '2005-12-1'

10.不要在 where 子句中的“=”左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引。

11.在使用索引字段作为条件时，如果该索引是复合索引，那么必须使用到该索引中的第一个字段作为条件时才能保证系统使用该索引，否则该索引将不会被使用，并且应尽可能的让字段顺序与索引顺序相一致。

12.不要写一些没有意义的查询，如需要生成一个空表结构：

select col1,col2 into #t from t where 1=0

13.Update 语句，如果只更改1、2个字段，不要Update全部字段，否则频繁调用会引起明显的性能消耗，同时带来大量日志。

14.对于多张大数据量（这里几百条就算大了）的表JOIN，要先分页再JOIN，否则逻辑读会很高，性能很差。

15.select count(\*) from table；这样不带任何条件的count会引起全表扫描，并且没有任何业务意义，是一定要杜绝的。

16.索引并不是越多越好，索引固然可以提高相应的 select 的效率，但同时也降低了 insert 及 update 的效率，因为 insert 或 update 时有可能会重建索引，所以怎样建索引需要慎重考虑，视具体情况而定。一个表的索引数最好不要超过6个，若太多则应考虑一些不常使用到的列上建的索引是否有 必要。

17.应尽可能的避免更新 clustered 索引数据列，因为 clustered 索引数据列的顺序就是表记录的物理存储顺序，一旦该列值改变将导致整个表记录的顺序的调整，会耗费相当大的资源。若应用系统需要频繁更新 clustered 索引数据列，那么需要考虑是否应将该索引建为 clustered 索引。

18.尽量使用数字型字段，若只含数值信息的字段尽量不要设计为字符型，这会降低查询和连接的性能，并会增加存储开销。这是因为引擎在处理查询和连 接时会逐个比较字符串中每一个字符，而对于数字型而言只需要比较一次就够了。

19.尽可能的使用 varchar/nvarchar代替 char/nchar ，因为首先变长字段存储空间小，可以节省存储空间，其次对于查询来说，在一个相对较小的字段内搜索效率显然要高些。

20.任何地方都不要使用 select \* from t ，用具体的字段列表代替“\*”，不要返回用不到的任何字段。

21.尽量使用表变量来代替临时表。如果表变量包含大量数据，请注意索引非常有限（只有主键索引）。

22. 避免频繁创建和删除临时表，以减少系统表资源的消耗。临时表并不是不可使用，适当地使用它们可以使某些例程更有效，例如，当需要重复引用大型表或常用表中的某个数据集时。但是，对于一次性事件， 最好使用导出表。

23.在新建临时表时，如果一次性插入数据量很大，那么可以使用 select into 代替 create table，避免造成大量 log ，以提高速度；如果数据量不大，为了缓和系统表的资源，应先create table，然后insert。

24.如果使用到了临时表，在存储过程的最后务必将所有的临时表显式删除，先 truncate table ，然后 drop table ，这样可以避免系统表的较长时间锁定。

25.尽量避免使用游标，因为游标的效率较差，如果游标操作的数据超过1万行，那么就应该考虑改写。

26.使用基于游标的方法或临时表方法之前，应先寻找基于集的解决方案来解决问题，基于集的方法通常更有效。

27.与临时表一样，游标并不是不可使用。对小型数据集使用 FAST\_FORWARD 游标通常要优于其他逐行处理方法，尤其是在必须引用几个表才能获得所需的数据时。在结果集中包括“合计”的例程通常要比使用游标执行的速度快。如果开发时 间允许，基于游标的方法和基于集的方法都可以尝试一下，看哪一种方法的效果更好。

28.在所有的存储过程和触发器的开始处设置 SET NOCOUNT ON ，在结束时设置 SET NOCOUNT OFF 。无需在执行存储过程和触发器的每个语句后向客户端发送 DONE\_IN\_PROC 消息。

29.尽量避免大事务操作，提高系统并发能力。

30.尽量避免向客户端返回大数据量，若数据量过大，应该考虑相应需求是否合理。

## 六、 SQL函数

**1 函数说明**

ASCII（<c1>）

该函数用于返回c1第一个字母的ASCII码，其中c1是字符串，它的逆函数是CHR()

CHR(<i>)

该函数用于求i对应的ASCII字符，其中i是一个数字

CONCAT（c1,c2）

该函数将c2连接到c1的后面，如果c1为nul，将返回c2；如果c2为null，则返回c1；如果c1、c2都为null，则返回为null。其中，c1、c2均为字符串，它和操作符“||”返回结果相同。

INITCAP（c1）

该函数将c1中每个单词的第一个字母大写，其他字母小写返回。单词由空格、控制字符、标点符号限制，其中c1为字符串

INSTR（c1，[c2,<i>[,j]]）

该函数用于返回c2在c1中第j次出现的位置，搜索从c1的第i个字符开始。当没有发现需要的字符时返回0，如果i为负数，那么搜索将从从右到左进行，但是位置还是从左到右来计算，i和j的默认值为1。其中，c1、c2均为字符串，i、j为整数。

LENGTH(c1)

该函数用于返回c1的长度，如果c1为null，那么将返回null值。其中c1为字符串。

LOWER(c1)

该函数用于返回c1的小写字符，经常出现在WHERE子串中。UPPER（c1）用来返回大写字母

LTRIM(c1,c2)

该函数表示将c1中最左边的字符去掉，使其第一个字符不在c2中，如果没有c2，那么c1就不会改变

REPLACE(c1,c2[,c3])

该函数用c3代替出现在c1中的c2后返回，其中c1、c2、c3都是字符串

SUBSTR(c1,<i>[,j])

该函数表示从c1的第i位开始返回长度为j的子字符串，如果j为空，则直到串的尾部。其中，c1为一字符串，i、j为整数

ABS(n)  用于返回n的绝对值

ACOS(n) 反余弦函数

ASIN(n)  反正切函数

ATAN(n)  用于返回大于或等于n的最小整数

CEIL(n)   用于返回大于或等于n的最小整数

COS(n)   用于返回n的余弦值，n为弧度

COSH(n)  用于返回n的双曲余弦值，n为数字

EXP(n)  用于返回e的n次幂，e=2.71828183

FLOOR(n)  用于返回小于等于n的最大整数

LN(n) 用于返回n的自然对数，n必须大于0

LOG(n1,n2)  用于返回以n1为底n2的对数

MOD(n1,n2)  用于返回n1除以n2的余数

POWER(n1,n2)  用于返回n1的n2次方

ROUND(n1,n2)  传回一个数值，该数值是按照指定的小数位元数进行四舍五入运算的结果

SIGN()  若n为负数，则返回-1，若n为正数，则返回1，诺n=0，则返回0

SIN(n)   用于返回n的正弦值，n为弧度

SINH(n) 用于返回n的双曲正弦值，n为弧度

SQRT(n)  用于返回n的平方根，n为弧度

TAN(n)   用于返回n的正切值，n为弧度

TANH(n)  用于返回n的双曲正切值，n为弧度

TRUNC(n1,n2) TRUNC函数为指定元素而截去的日期值。

ADD\_MONTHS(D,<I>)  返回日期d加上i个月后的结果

LAST\_DAY(d)  返回包含日期d月份的最后一天

MONTHS\_BETWEEN(d1,d2)  返回d1和d2之间月的数目

NEW\_TIME(d1,tz1,tz2)

SYSDATE  返回当前日期和时间，该函数没有参数

CHARTORWID(c1)  该函数将c1转换成RWID数据类型，其中c1是一个字符串

CONVERT(c1,dser[,sset])   该函数将字符串c1由sset字符集转换为dset字符集，sset默认设置为数据库的字符集，其中c1为字符串，dset，sset是两个字符集

ROWIDTOCHAR()   该函数将ROWID数据类型转换为CHAR数据类型

TO\_CHAR(x[,fmt[n;sparm,]])  该函数将x转换成字符串

TO\_DATE  获取时间

TO\_MULTI\_BYTE(c1) 该函数将c的单字节字符转换成多字节字符，其中c1表示一个字符串

TO\_NUMBER(string[,format[,nlsparams]])  将CHAR或VARCHAR2类型的string转换为一个NUMBER类型的数值，如果指定了format,那么string应该遵循相应的数字格式。nlsparams的用来指定小数点和钱分位分隔符，以及货币符号。它与TO\_CHAR()互为反函数

TO\_SINGLE\_BYTE(c1)   将字符串c1中的多字节字符转换成等价的单字节字符。该函数仅当数据库字符集同时包含单字节和多字节时才使用

AVG(x[{DISTINCT|ALL}]) 用于返回数值的平均值。默认设置为ALL

COUNT(x[{DISTINCT|ALL}}])  用于返回查询中行的数目，默认设置是ALL，表示返回所有的行

MAX(x[{DISTINCT|ALL}}]) 用于返回查询列表最大值

MIN(x[{DISTINCT|ALL}}]) 用于返回选择列表项目的最小值

STDDEV(x[{DISTINCT|ALL}}])   用于返回选择列表项目的标准差，所谓的标准差是方差的平方根

SUM(x[{DISTINCT|ALL}}])   用于返回选择列表项目的数值的总和

VARIANCE(x[{DISTINCT|ALL}}])   用于返回选择列表项目的统计方差

**2 实例举例**

& 把'0755-'、'23288344'、'转23'三个字符串连接起来

select concat('0755-','23288344')||'转23' from dual;

& 将'wade'字符串的第一个字母变为大写

select initcap('wade') from dual;

& 从'oracleoracle'搜索'a'第2次出现的位置，从第4个字母开始搜索

select instr('oracleoracle','a','4','1') instring from dual;

& 返回数字5555的长度

select length(5555) from dual;

& 返回'Wade'所有的字符大写和小写的字符串

select upper('Wade'),lower('Wade') from dual;

& 先在字符串'gao'的右边贴上10个'\*'，再在左边贴上27个'\*'

select lpad(rpad('gao',10,'\*'),27,'\*')from dual;

& 字符串'wade and mcgrady'从第3个开始取，取8个

select substr('wade and mcgrady',3,8) from dual;

& 字符串'he love no you'中'he'换成'i'

select replace('he love no you','he','i') from dual;

& 返回-100的绝对值

select abs(-100) from dual;

& 返回大于或等于3.14156的最小整数

select ceil(3.14156) from dual;

& 数值55.5和-55.7的四舍五入和去掉小数部分

select round(55.5),round(-55.7),trunc(55.3),trunc(-55.7) from dual;

& 增加2月和减少2月

select add\_months(to\_date('199912','yyyymm'),2),add\_months(sysdate,-2) from dual;

& 返回日期所在月的最后一天

select last\_day(sysdate) from dual;

& 用来得到系统的当前日期

select to\_char(sysdate,'yyyy--mm--dd') from dual;

& 把日期、数字转化成字符串

select to\_char(sysdate,'yyyy--mm--dd') from dual;

select to\_char(sysdate,'yyyy/mm/dd') from dual;

select to\_char(44444) from dual;

& 把字符串转化成日期

select to\_date('2012-11-12','yyyy-mm-dd') from dual;

& 把字符串转化成数字

select to\_number('1999') from dual;

& 返回当前的用户

select user from dual;

& 求一个字段的平均值、最小值、最大值、总数量、总数

select avg(t.app\_info\_source),min(t.app\_info\_source),max(t.app\_info\_source),count(t.app\_info\_source),sum(t.app\_info\_source) from t\_application\_info t

& case when then 语句

(1)和decode用法一样

select

case t.name

when 'wade' then 'WWW'

when 'james' then 'jjj'

else 'Other'

end

from student t;

(2)特有的用法

select case when 55 <60 then 'd'

when 55 >=60 and 55 <70 then 'c'

when 55 >=70 and 55 <80 then 'b'

else 'a' end from dual;

& NVL和NVL2用法

select nvl(null,1) from dual; 1

select nvl(3,1) from dual; 3

select nvl2(null,1,2) from dual; 2

select nvl2(5,3,1) from dual; 3

&DECODE函数

select

decode(concat('wa','de'),'wade','www','james','jjj','other')

from dual;

& 结果处理函数

UNION： 将多个结果集的行合并，并移除重复的行。

UNION ALL： 将多个结果集所有的行合并，不论是否有重复的行。

EXCEPT： 提取只在 EXCEPT 左侧存在，但右侧不存在的行。

INTERSECT： 只提取两个结果集里，都存在的行。

select t1.app\_name\_en from t\_application\_info t1

union all

select t2.app\_name\_en from t\_application\_info\_change t2

select t1.app\_name\_en from t\_application\_info t1

union

select t2.app\_name\_en from t\_application\_info\_change t2

select t1.app\_name\_en from t\_application\_info t1

intersect

select t2.app\_name\_en from t\_application\_info\_change t2

SELECT t1.app\_name\_en from t\_application\_info t1

EXCEPT

select t1.app\_name\_en from t\_application\_info t1

INTERSECT

select t2.app\_name\_en from t\_application\_info\_change t2;

& 分析函数

select name,row\_number() over(partition by name order by point) from wade;

以name分区,以point排序的rownumber

select name,rank() over(partition by name order by point) from wade;

和row\_number()类似

select name,min(point) over(partition by name order by point) from wade;

以name分区,以point排序的point最小值

select name,max(point) over(partition by name order by point) from wade;

以name分区,以point排序的最大值

select name,count(point) over(partition by name order by point) from wade;

以name分区,以point排序的数量

select name,avg(point) over(partition by name order by point) from wade;

以name分区,以point排序的平均数

select name,sum(point) over(partition by name order by point) from wade;

以name分区,以point排序的总数

&其它函数

(1)排序总规则(中文-拼音-数字)

select \* from wade order by nlssort(name,'NLS\_SORT=SCHINESE\_PINYIN\_M');

(2)找出id等于3的所有子记录

select \*

from wade

start with id = 3

connect by nocycle prior id = parentId

(3)判断该字段是否是全数字组成

select replace(translate('22','1234567890',' '),' ','') from dual; null

select replace(translate('ee22','1234567890',' '),' ','') from dual; ee

select replace(translate('','1234567890',' '),' ','') from dual; null

select replace(translate(null,'1234567890',' '),' ','') from dual; null

## 七、 SQL经典

1、查看当前所有对象

SQL > select \* from tab;

2、建一个和a表结构一样的空表

SQL > create table b as select \* from a where 1=2;

SQL > create table b(b1,b2,b3) as select a1,a2,a3 from a where 1=2;

3、察看数据库的大小，和空间使用情况

SQL > col tablespace format a20SQL > select b.file\_id　　文件ID,　　b.tablespace\_name　　表空间,　　b.file\_name　　　　　物理文件名,　　b.bytes　　　　　　　总字节数,　　(b.bytes-sum(nvl(a.bytes,0)))　　　已使用,　　sum(nvl(a.bytes,0))　　　　　　　　剩余,　　sum(nvl(a.bytes,0))/(b.bytes)\*100　剩余百分比　　from dba\_free\_space a,dba\_data\_files b　　where a.file\_id=b.file\_id　　group by b.tablespace\_name,b.file\_name,b.file\_id,b.bytes　　order by b.tablespace\_name　　/　　dba\_free\_space --表空间剩余空间状况　　dba\_data\_files --数据文件空间占用情况

4、查看现有回滚段及其状态

SQL > col segment format a30SQL > SELECT SEGMENT\_NAME,OWNER,TABLESPACE\_NAME,SEGMENT\_ID,FILE\_ID,STATUS FROM DBA\_ROLLBACK\_SEGS;

5、查看数据文件放置的路径

SQL > col file\_name format a50SQL > select tablespace\_name,file\_id,bytes/1024/1024,file\_name from dba\_data\_files order by file\_id;

6、显示当前连接用户

SQL > show user

7、把SQL\*Plus当计算器

SQL > select 100\*20 from dual;

8、连接字符串

SQL > select 列1 | |列2 from 表1;SQL > select concat(列1,列2) from 表1;

9、查询当前日期

SQL > select to\_char(sysdate,'yyyy-mm-dd,hh24:mi:ss') from dual;

10、用户间复制数据

SQL > copy from user1 to user2 create table2 using select \* from table1;

11、视图中不能使用order by，但可用group by代替来达到排序目的

SQL > create view a as select b1,b2 from b group by b1,b2;

12、通过授权的方式来创建用户

SQL > grant connect,resource to test identified by test;

SQL > conn test/test

13、查出当前用户所有表名。

select unique tname from col;

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

/\* 向一个表格添加字段 \*/alter table alist\_table add address varchar2(100);

/\* 修改字段 属性 字段为空 \*/alter table alist\_table modify address varchar2(80);

/\* 修改字段名字 \*/create table alist\_table\_copy as select ID,NAME,PHONE,EMAIL,QQ as QQ2, /\*qq 改为qq2\*/ADDRESS from alist\_table;

drop table alist\_table;rename alist\_table\_copy to alist\_table/\* 修改表名 \*/

空值处理有时要求列值不能为空create table dept (deptno number(2) not null, dname char(14), loc char(13));

在基表中增加一列alter table deptadd (headcnt number(3));

修改已有列属性alter table deptmodify dname char(20);注：只有当某列所有值都为空时，才能减小其列值宽度。只有当某列所有值都为空时，才能改变其列值类型。只有当某列所有值都为不空时，才能定义该列为not null。例：alter table dept modify (loc char(12));alter table dept modify loc char(12);alter table dept modify (dname char(13),loc char(12));

查找未断连接select process,osuser,username,machine,logon\_time ,sql\_textfrom v$session a,v$sqltext b where a.sql\_address=b.address;

-----------------------------------------------------------------1.以USER\_开始的数据字典视图包含当前用户所拥有的信息, 查询当前用户所拥有的表信息:select \* from user\_tables;2.以ALL\_开始的数据字典视图包含ORACLE用户所拥有的信息,查询用户拥有或有权访问的所有表信息:select \* from all\_tables;

3.以DBA\_开始的视图一般只有ORACLE数据库管理员可以访问:select \* from dba\_tables;

4.查询ORACLE用户：conn sys/change\_on\_installselect \* from dba\_users;conn system/manager;select \* from all\_users;

5.创建数据库用户：CREATE USER user\_name IDENTIFIED BY password;GRANT CONNECT TO user\_name;GRANT RESOURCE TO user\_name;授权的格式: grant (权限) on tablename to username;删除用户(或表):drop user(table) username(tablename) (cascade);6.向建好的用户导入数据表IMP SYSTEM/MANAGER FROMUSER = FUSER\_NAME TOUSER = USER\_NAME FILE = C:\EXPDAT.DMP COMMIT = Y7.索引create index [index\_name] on [table\_name]( "column\_name ")intersect运算

返回查询结果中相同的部分

exp:各个部门中有哪些相同的工种

selectjob

fromaccount

intersect

selectjob

fromresearch

intersect

selectjob

fromsales;

minus运算

返回在第一个查询结果中与第二个查询结果不相同的那部分行记录。

有哪些工种在财会部中有，而在销售部中没有？

exp:selectjobfromaccount

minus

selectjobfromsales;

——————————————————————————————————

1. oracle安装完成后的初始口令?

　internal/oracle

　　sys/change\_on\_install

　　system/manager

　　scott/tiger

　　sysman/oem\_temp

2. oracle9ias web cache的初始默认用户和密码？

administrator/administrator

3. oracle 8.0.5怎么创建数据库?

用orainst。假如有motif界面，可以用orainst /m

4. oracle 8.1.7怎么创建数据库?

dbassist

5. oracle 9i 怎么创建数据库?

dbca

6. oracle中的裸设备指的是什么?

裸设备就是绕过文件系统直接访问的储存空间

7. oracle如何区分 64-bit/32bit 版本？？？

$ sqlplus '/ as sysdba'

sql\*plus: release 9.0.1.0.0 - production on mon jul 14 17:01:09 2003

(c) copyright 2001 oracle corporation. all rights reserved.

connected to:

oracle9i enterprise edition release 9.0.1.0.0 - production

with the partitioning option

jserver release 9.0.1.0.0 - production

sql> select \* from v$version;

banner

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

oracle9i enterprise edition release 9.0.1.0.0 - production

pl/sql release 9.0.1.0.0 - production

core 9.0.1.0.0 production

tns for solaris: version 9.0.1.0.0 - production

nlsrtl version 9.0.1.0.0 - production

sql>

8. svrmgr什么意思？

svrmgrl，server manager.

9i下没有，已经改为用sqlplus了

sqlplus /nolog

变为归档日志型的

9. 请问如何分辨某个用户是从哪台机器登陆oracle的?

select machine , terminal from v$session;

10. 用什么语句查询字段呢？

desc table\_name 可以查询表的结构

select field\_name,... from ... 可以查询字段的值

select \* from all\_tables where table\_name like '%'

select \* from all\_tab\_columns where table\_name='??'

11. 怎样得到触发器、过程、函数的创建脚本？

desc user\_source

user\_triggers

12. 怎样计算一个表占用的空间的大小？

select owner,table\_name,

num\_rows,

blocks\*aaa/1024/1024 "size m",

empty\_blocks,

last\_analyzed

from dba\_tables

where table\_name='xxx';

here: aaa is the value of db\_block\_size ;

xxx is the table name you want to check

13. 如何查看最大会话数？

select \* from v$parameter where name like 'proc%';

sql>

sql> show parameter processes

name type value

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

aq\_tm\_processes integer 1

db\_writer\_processes integer 1

job\_queue\_processes integer 4

log\_archive\_max\_processes integer 1

processes integer 200

这里为200个用户。

select \* from v$license;

其中sessions\_highwater纪录曾经到达的最大会话数

14. 如何查看系统被锁的事务时间？

select \* from v$locked\_object ;

15. 如何以archivelog的方式运行oracle。

init.ora

log\_archive\_start = true

restart database

16. 怎么获取有哪些用户在使用数据库

select username from v$session;

17. 数据表中的字段最大数是多少?

表或视图中的最大列数为 1000

18. 怎样查得数据库的sid ?

select name from v$database;

也可以直接查看 init.ora文件

19. 如何在oracle服务器上通过sqlplus查看本机ip地址 ?

select sys\_context('userenv','ip\_address') from dual;

假如是登陆本机数据库，只能返回127.0.0.1，呵呵

20. unix 下怎么调整数据库的时间？

su -root

date -u 08010000

21. 在oracle table中如何抓取memo类型字段为空的数据记录?

select remark from oms\_flowrec where trim(' ' from remark) is not null ;

22. 如何用bbb表的数据去更新aaa表的数据(有关联的字段)

up2003-10-17 aaa set bns\_snm=(select bns\_snm from bbb where aaa.dpt\_no=bbb.dpt\_no) where bbb.dpt\_no is not null;

23. p4计算机安装方法

将symcjit.dll改为sysmcjit.old

24. 何查询server是不是ops?

select \* from v$option;

假如parallel server=true则有ops能

25. 何查询每个用户的权限?

select \* from dba\_sys\_privs;

26. 如何将表移动表空间?

alter table table\_name move tablespace\_name;

27. 如何将索引移动表空间?

alter index index\_name rebuild tablespace tablespace\_name;

28. 在linux,unix下如何启动dba studio?

oemapp dbastudio

29. 查询锁的状况的对象有?

v$lock, v$locked\_object, v$session, v$sqlarea, v$process ;

查询锁的表的方法:

select s.sid session\_id, s.username, decode(lmode, 0, 'none', 1, 'null', 2, 'row-s (ss)', 3, 'row-x (sx)', 4, 'share', 5, 's/row-x (ssx)', 6, 'exclusive', to\_char(lmode)) mode\_held, decode(request, 0, 'none', 1, 'null', 2, 'row-s (ss)', 3, 'row-x (sx)', 4, 'share', 5, 's/row-x (ssx)', 6, 'exclusive', to\_char(request)) mode\_requested, o.ccbzzp||'.'||o.object\_name||' ('||o.object\_type||')', s.type lock\_type, l.id1 lock\_id1, l.id2 lock\_id2 from v$lock l, sys.dba\_objects o, v$session s where l.sid = s.sid and l.id1 = o.object\_id ;

30. 如何解锁?

alter system kill session ‘sid,serir#’;

31. sqlplus下如何修改编辑器?

define \_editor="" -- 必须加上双引号

来定义新的编辑器，也可以把这个写在$oracle\_home/sqlplus/admin/glogin.sql里面使它永久有效。

32. oracle产生随机函数是?

dbms\_random.random

33. linux下查询磁盘竞争状况命令?

sar -d

33. linux下查询cpu竞争状况命令?

sar -r

34. 查询当前用户对象?

select \* from user\_objects;

select \* from dba\_segments;

35. 如何获取错误信息?

select \* from user\_errors;

36. 如何获取链接状况?

select \* from dba\_db\_links;

37. 查看数据库字符状况?

select \* from nls\_database\_parameters;

select \* from v$nls\_parameters;

38. 查询表空间信息?

select \* from dba\_data\_files;

39. oracle的interal用户要口令?

修改 sqlnet.ora

sqlnet.authentication\_services=(nts)

40. 出现java.exe的解决办法?

一般是将oracleorahomexihttpserver改成手工启动可以的

x是8或9

41. 如何给表、列加注释？

sql>comment on table 表 is '表注释';

注释已创建。

sql>comment on column 表.列 is '列注释';

注释已创建。

sql> select \* from user\_tab\_comments where comments is not null;

42. 如何查看各个表空间占用磁盘情况？

sql> col tablespace format a20

sql> select

b.file\_id 文件id号,

b.tablespace\_name 表空间名,

b.bytes 字节数,

(b.bytes-sum(nvl(a.bytes,0))) 已使用,

sum(nvl(a.bytes,0)) 剩余空间,

sum(nvl(a.bytes,0))/(b.bytes)\*100 剩余百分比

from dba\_free\_space a,dba\_data\_files b

where a.file\_id=b.file\_id

group by b.tablespace\_name,b.file\_id,b.bytes

order by b.file\_id

43. 如把oracle设置为mts或专用模式？

#dispatchers="(protocol=tcp) (service=sidxdb)"

加上就是mts，注释就是专用模式，sid是指你的实例名。

44. 如何才能得知系统当前的scn号 ?

select max(ktuxescnw \* power(2, 32) + ktuxescnb) from x$ktuxe;

45. 请问如何在oracle中取毫秒?

9i之前不支持,9i开始有timestamp.

9i可以用select systimestamp from dual;

## 安全方面

## **1 安全性是对于一个数据库系统是至关重要的**，衡量一个数据库产品的重要指标，如果没有足够的安全性，可能导致数据的丢失、泄露、破坏，造成无法估计的损失。

**2 数据库安全性包括2个方面：一是防止非法用户对数据库的访问，未授权的用户不能登录数据库；二是每个数据库用户都有不同的权限，只能进行自己权限范围内的操作。**

**3 Oracle安全性包括2个部分：**

**@系统安全性--**-有效的用户名与口令的组合、用户是否被授权、用户的磁盘空间使用大小、用户的资源限制、是否启动了数据库审计功能。

**@数据安全性---**控制数据的存取和操作机制

**4 用户管理**

@Oracle初始化用户

sys---具有最高权限的用户，可启动修改关闭数据库，拥有数据字典

system---辅助的数据库管理员，不能启动关闭数据库，但能创建用户，删除用户等操作

scott---用于测试网络连接的用户，口令为tiger

public---实际上是一个用户组，数据库所有的用户都属于该组

@用户认证方式

数据库身份认证---用户口令以加密的方式保存在数据库内部

create user u1 identified by wade399

外部身份认证---用户的账户由ORACLE数据库管理，但口令管理和身份验证有外部服务完成，外部服务可以使操作系统和网络服务，

数据库不要求用户输入口令，但会从外部服务获取用户的登录信息，在这种方式下，oracle不会保存用户的口令信息，但需要创建响应的用户，

create user u2 identified externally

全局身份认证---使用网络中的安全管理服务器对用户进行身份认证

create user u3 identified globally as 'cn=d1,l=dalian,c=us'

@创建用户

创建用户名字为wade,口令为wade399，默认表空间名称为users,表空间配额为10M，初始状态密码过期需要修改，账户锁定

create user wade inentidied by wade399 default tablespace users quota 10M on users password expire account lock;

@修改用户

修改用户wade的默认表空间users表空间配额为50M，orcltbs1表空间为10M，账户不锁定

alter user wade default tablespace users quota 50M on users quota 10M on orcltbs1 account unlock;

@删除用户

drop user wade cascade ---用户wade被删除，该用户所拥有的对象也随之删除

drop user wade ---用户wade被删除，该用户所拥有的对象失效

**5 权限管理**

系统权限-----数据库级别执行某种操作的权限，超过100种系统权限，可将这些权限授予用户、角色、public用户组，若授予public用户组某些权限，则所有用户都具备；

对象权限-----对数据库对象执行某种操作的权限，增删改查

直接授权-----利用grant命令直接授权

间接授权-----先授权给角色，再把角色授予用户

ORACLE权限管里的过程就是权限授予与回收的过程

@授权

注意

(1)只有DBA才有alter database的权限

(2)应用程序开发者一般具有create table,create view,create index等权限

(3)普通用户一般只具有create session系统会话连接权限

(4)授权时带有with admin option时，授权传递性，用户可以把这些权限传给其他用户

为public用户组授权create session

grant create session to public

为用户wade授予create table,create view，create session权限，wade再给wade2用户授予create table权限

system: grant create table,create view，create session to wade with admin option

wade : grant create table to wade2

revoke create table,create view，create session from wade;

虽然wade的权限被收回了，但是前段时间wade传给wade2的权限不受到影响

对象权限：

将scott用户下的emp表的select、update权限授予wade，wade再给wade2用户授予emp表的select权限

system: grant select,update on scott.emp to wade with admin option

wade : grant select on scott.emp to wade2

revoke select,update on scott.emp from wade;

wade的权限被收回了，同时前段时间wade传给wade2的权限也没了

**6 角色管理**

所谓角色就是相关权限的集合，角色、用户、权限、授权的关系相当于工作流中的角色、用户、模块、权限的关系。

@预定义角色-----在数据库创建的时候系统自动创建的一些常用的角色，oracle中有30种，常见的有connect、resource、dba等

@自定义角色

create role r1 (此不需要进行验证)

create role r2 identidied by r2 (此进行验证的口令为r2)

@修改角色

alter role r1 identidied by r1

alter role r2 not identidied

@角色的生效于失效

set role none

set role r1 none

@删除角色

drop role r1

@权限授予角色

grant create session,create table,create view to r1

revoke create session,create table,create view from r1;

@角色授予角色

grant connect,dba to r1

revoke connect,dba from r1;

@角色授予用户

grant connect,dba to wade

revoke connect,dba from wade;

@权限授予用户

grant create session to wade

revoke create session from wade

@用户角色屏蔽与激活

alter user wade default role none

alter user wade default role connect,dba

**7 概要文件管理**

概要文件是数据库和系统资源限制的集合，是安全策略的组成部分，每个用户必须有一个概要文件，创建数据库的时候，会创建一个default的默认文件。

分为资源管理参数和口令管理参数。

@创建一个概要文件，用户连续4次登录失败，锁定该账户，10天后自动解锁

create profile res\_pro limit failed\_login\_attempts 4 password\_lock\_time 10;

@创建一个概要文件，用户最多创建4个并发会话，每个回话最多持续60分钟，连续回话有20分钟空闲，停止会话，每个会话私有SQL区为100K，每个SQL语句占用CPU时间总量不超过10秒

create profile res\_pro2 limit session\_per\_user 4 connect\_time 60 idle\_time 20 private\_sga 100k cpu\_per\_call 10;

@将概要文件分发给用户

alter user wade profile res\_pro

@删除概要文件

drop profile res\_pro2

## 九、备份恢复与数据导出

**数据库备份的目的是防止意外丢失或毁坏数据，如果对数据库进行了备份，如果发生了意外也可以进行恢复**

**1、物理备份与恢复**

将数据库物理文件进行备份，包括数据文件、控制文件、重做日志文件、归档重做日志文件的物理复制

@冷备份----在数据库关闭状态下进行备份，恢复就是用备份的文件替换原来的文件

@热备份----在数据库开启状态下进行备份

**2、逻辑备份与恢复**

对表结构、模式对象进行备份，是SQL语句的集合

oracle的export工具读取数据并保存在一个称为导出转储文件（dmp）的二进制文件汇总，可以选择导出整个数据库、用户、表的内容。

exp help = y 显示导出工具帮助信息

@导出数据

(1)交互式数据导出

exp

登录

导出文件：expdat> d:\test.dmp

(2)U(用户)，或(3)T表：U

。

。

。

在没有警告的条件下成功终止导出

(2)命令行模式导出

exp userid = scott/tiger file =d:/test.dmp tables=test

(3)参数模式导出

I: 创建一个参数文件test.parfile

userid=scott/tiger

file=d:\test.dmp

tables=test

II: 输入命令行：exp parfile=d:\test.parfile

(4)oOEM界面导出

orcl-数据管理-导出

@导入数据

(1)交互式数据导入

imp

登录

导入文件：expdat> d:\test.dmp

。

。

。

在没有警告的条件下成功终止导入

(2)命令行模式导入

imp userid = scott/tiger file =d:/test.dmp tables=test rows=y

(3)参数模式导入

I: 创建一个参数文件test.parfile

userid=scott/tiger

file=d:\test.dmp

tables=test

II: 输入命令行：imp parfile=d:\test.parfile

(4)oOEM界面导入

orcl-数据管理-导入

**3.监听器**

Oracle net监听器简称监听器，是位于服务器端的一个后台进程，负责对客户端传入的连接请求进行监听，接收客户端的连接请求后，交给服务器进行处理，一旦

客户端和服务器的连接建立起来，两端可以直接通信，不需要监听器的参与。

常见的网络配置工具

Oracle Net Configuration Assistant

Oracle Net Manager

## 十、PL/SQL编程

**1 PL/SQL是oracle对标准SQL语言的过程化扩展**，是Oracle专用的一种高级程序设计语言。PL/SQL块分成2类，一为匿名块，动态生成，只能执行一次，不能由其它程序调用的块；另一类为命名块，一次编译可多次执行的块，包括函数、存储过程、包、触发器等，编译后放在服务器中由应用程序或指定环境调用。

**2 特点长处**

(1)与sql语句紧密集成，所有SQL语句在PL/SQL中都得到支持；

(2)减少网络流量，提高应用程序的运行性能，一个PLSQL程序可包含多个SQL语句；

(3)模块化的程序设计功能，提高系统的可靠性；

(4)服务器端的程序设计，可移植性好，以编译的形式存储在数据库中；

**3 PL/SQL程序块的组成**

declare

申明部分，定义变量，数据类型，异常，局部子程序等【可选】

begin

执行部分，实现块的功能，可嵌套其它的子程序【必选】

exception

异常处理部分【可选】

end;

**4 定义一个包含子块的PL/SQL块**

declare

v\_name varchar2(20);

v\_sal number(5);

begin

begin

select name into v\_name from emp where empno = 1887;

end;

select avg(sal) into v\_sal from emp where name = v\_name;

dbms\_output.putline(v\_sal);

end;

注： select ... into ...语句: 只能查出一条记录信息，没数据生成no\_data\_found异常，查询到多个记录则生成too\_many\_rows异常，返回多个记录用游标处理。

**5 数据类型**

数字类型：number(2,1),binary\_integer。number(2,1)表示总共2位，小数点后1位

字符类型：varchar2(30),char,varchar。varchar2最大字节数oracle为4000，PL/SQL中为32767

日期类型：date,timestamp。

布尔类型：true,false,null。

%type和%rowtype: 定义一个变量的数据类型和表某一列的数据类型一致，表示为%type；和表结构一致，表示为%rowtype。

v\_name emp.name%type;

v\_emp emp%rowtype;

**6 变量与常量**

变量名称 [constant] 数据类型 [not null] [default|:= 值]

v1 number(3);

v2 varchar2(20) not null := '4455';

v3 constant number(3,1) default 50.5;

**7 DML操作**

declare

v\_empno emp.empno%type;

v\_sal\_after emp.sal%type;

begin

select empno+'555' into v\_empno from emp where empno = 7600;

insert into emp(empno,ename,sal,deptno) values (v\_empno,'Jon',2300,20);

update emp set sal = sal+100 where empno = v\_empno returning sal into v\_sal\_after; //returning把修改之后的薪水返回

delete emp where empno = v\_empno;

dbms\_output.putline(v\_sal\_after);

end;

**8 控制流程之【if】**

declare

v\_no number(1) :=5;

v\_result varchar2(2);

begin

if v\_no < 5 then v\_result := 'less then 5';

else if v\_no = 5 then v\_result := 'equals 5';

else if v\_no > 5 then v\_result := 'more then 5';

else v\_result := 'error';

end if;

end;

**9 控制流程之【case】**

declare

v\_no number(1) :=5;

v\_result varchar2(2);

begin

case v\_no

when 1 then v\_result := 'equals 1';

when 3 then v\_result := 'equals 3';

when 5 then v\_result := 'equals 5';

else v\_result := 'other';

end case;

case

when v\_no<5 then v\_result := 'less then 5';

when v\_no=5 then v\_result := 'equals 5';

when v\_no>5 then v\_result := 'more then 5';

else v\_result := 'error';

end case;

end;

**10 控制流程之【简单循环】**

declare

v\_no binary\_integer :=1;

begin

loop

insert into emp(empno,ename) values (v\_no,'Jon');

v\_no:=v\_no+1;

exit when v\_no = 50;

end loop;

end;

**11 控制流程之【while循环】**

declare

v\_no binary\_integer :=1;

begin

while v\_no <50 loop

insert into emp(empno,ename) values (v\_no,'Jon');

v\_no:=v\_no+1;

end loop;

end;

**12 控制流程之【for循环】**

declare

v\_no binary\_integer :=1;

begin

for v\_no in 1..50 loop

insert into emp(empno,ename) values (v\_no,'Jon');

end loop;

end;

**13 异常处理**

预定义异常--常见的异常

非预定义异常--不常见的异常

自定义异常

declare

v\_no binary\_integer :=2000;

v\_ex exception;

begin

if v\_no>1500 then raise v\_ex;

exception

when v\_ex then dbms\_output.put\_line('too high!');

when others then

v\_code:=sqlcode;

v\_text:=sqlerrm;

dbms\_output.put\_line(v\_code+': '+v\_text);

end;

**14 游标 --处理select语句返回多条数据**

显示游标：用户自定义和操作

隐式图标：系统自动进行操作，用于处理DML语句和单行数据

**（1）显示游标**

declare

cursor c1 is select id,name from book ; //定义游标

v\_id book.id%type;

v\_name book.name%type;

begin

open c1; //打开游标

loop

fetch c1 into v\_id,v\_name ; //检索游标

exit when c1%notfound;

dbms\_output.put\_line(v\_id+”;”+v\_name);

end loop;

close c1; //关闭游标

end;

/

3种方式检索游标，以上是简单loop方式

open c1;

fetch c1 into v\_id,v\_name ;

while c1%found

loop

fetch c1 into v\_id,v\_name ;

dbms\_output.put\_line(v\_id+”;”+v\_name);

end loop;

close c1;

for v\_id,v\_name in c1

loop

fetch c1 into v\_id,v\_name ;

dbms\_output.put\_line(v\_id+”;”+v\_name);

end loop;

注意：用for循环检索的时候不用打开、关闭游标

**显示游标的属性**

**%isopen 布尔型，游标是否打开**

**%found 布尔型，是否最近一次能检索到数据**

**%notfound 布尔型，与%found相反**

**%rowcount 数字型，返回到目前为止游标从缓冲区检索到的记录个数**

参数化游标

declare

cursor c\_1(e\_name emp.ename%type,e\_sal emp.sal%type) is select \* from emp where ename = e\_name and sal < e\_sal;

v\_emp emp%rowtype;

begin

open c\_1('wade',4000);

loop

fetch c\_1 into v\_emp;

dbms\_output.put\_line(v\_emp.empno);

exit when c\_1%notfound;

end loop;

end;

利用游标更新或删除数据

declare

cursor c\_1(e\_name emp.ename%type,e\_sal emp.sal%type) is select \* from emp where ename = e\_name and sal < e\_sal for update;

v\_emp emp%rowtype;

begin

open c\_1('wade',4000);

loop

fetch c\_1 into v\_emp;

update emp set sal =sal+100 where current of c\_1;

exit when c\_1%notfound;

end loop;

end;

**（2）隐式游标**

sql%isopen 布尔型，游标是否打开

sql%found 布尔型，是否最近一次能检索到数据

sql%notfound 布尔型，与%found相反

sql%rowcount 数字型，返回到目前为止游标从缓冲区检索到的记录个数

begin

update emp set sal =sal+100 where empno =1009;

if sql%notfound or sql%rowcount =0 then

insert into emp(empno,sal) values ('1009',2000);

end if;

end;

**15 存储子程序-----指被命名的PL/SQL块，以编译的形式存储在数据库服务器中，可以在应用程序中进行调用，包括存储过程、函数**

**（1）存储过程和函数区别：**

通常存储过程是一个独立的表达式被调用，没有返回值；

函数是一个独立的表达式的组成部分被调用，有返回值；

**（2）存储过程**

(1) 参数模式

in- 默认的，调用过程中实参没有被形参改变；

out - 返回调用环境时，形参的值赋予实参；

inout- 返回调用环境时，形参的值赋予实参；

create or replace proceduce p1(v\_id in number(10),v\_name in varchar(10)) as

v\_id number(10);

v\_name varchar(20);

begin

dbms\_output.put\_line(v\_id+”;”+v\_name);

end;

declear

v\_id number(10):=10;

v\_name varchar(20):=’wade’;

begin

exec p1(v\_id,v\_name);

call p1(v\_id,v\_name);

end;

**（2）触发器----是特殊的存储过程，由系统自动的调用，不能由应用程序调用。**

1 触发器的组成

(1)建立在基本表的操作(DML触发器),响应基本的增删改操作；

建立在视图上面的操作(instead of触发器)；

建立在系统或模块上面的操作(系统触发器)；

(2)触发事件可以用OR连接起来

(3)触发时间可以是before和after

(4)触发级别：

语句级（触发一次执行一次）

行级for each row（触发事件每作用于一行记录，触发一次）

(5)触发条件和触发操作

create or replace trigger t1 after insert or delete or update on book for each row;

declear

v\_id number(10):=1;

v\_name number(10):=1;

begin

loop

insert into shop(id,name) values(v\_id, v\_name);

v\_id++;

v\_name++;

exit when v\_id = 100000;

end loop;

end t1;

create or replace trigger t2 before update of sal on emp for each row when (old.sal<new.sal)

begin

raise\_application\_error(-2000,'error...');

end t2;

alter trigger t1 disable;

alter trigger t1 enable;

drop trigger t1;

**（3）函数**

create or replace function f1(e\_no emp.empno%type) return emp.sal%type as

v\_sal emp.sal%type;

begin

select sal into v\_sal from emp where empno = e\_no;

return v\_sal;

end f1;

declare

v\_sal emp.sal%type;

begin

v\_sal = f1(1996);

end;

alter function f1 compile;

drop function f1;

**（4）包------将相关的PL/SQL块或元素（过程、函数、变量、游标等）组织在一起，成为一个完整的单元，供应用程序调用。**

create or replace package body pp1 as

create or replace proceduce p1(v\_id number(10),v\_name varchar(10)) as

begin

dbms\_output.put\_line(v\_id+”;”+v\_name);

end p1;

create or replace proceduce p2() as

begin

dbms\_output.put\_line('55555');

end p1;

end pp1;

begin

pp1.p1(23,'wade');

pp1.p2();

end;

## Oracle客户端

**1 pl/sql**

**2 navagation**

**3 dbeaver-ce**

## 事务&锁&并发

**1 数据库事务说明**

一组SQL,一个逻辑工作单位，执行时整体修改或者整体回退。

数据库是一个多用户使用的共享资源。当多个用户并发地存取数据时，在数据库中就会产生多个事务同时存取同一数据的情况。若对并发操作不加控制就可能会读取和存储不正确的数据，破坏数据库的一致性。如果是单用户的系统，那完全没有必要这个锁，就是因为有多用户并发操作，我们为了确保资源的安全性（也就是Oracle的数据完整性和一致性）才引申出这个锁出来。Oracle 利用其锁机制来实现事务间的数据并发访问及数据一致性。

**排他锁（exclusive lock，即X锁）**

事务设置排它锁后，该事务单独获得此资源，另一事务不能在此事务提交之前获得相同对象的共享锁或排它锁。

**共享锁（share lock，即S锁）**

共享锁使一个事务对特定数据库资源进行共享访问——另一事务也可对此资源进行访问或获得相同共享锁。

共享锁为事务提供高并发性，但如拙劣的事务设计+共享锁容易造成死锁或数据更新丢失。

行级锁，一般是指排它锁，即被锁定行不可进行修改，删除，只可以被其他会话select。行级锁之前需要先加表结构共享锁。

表级锁，一般是指表结构共享锁，是不可对该表执行DDL操作，但对DML操作都不限制

共享锁又称为读锁，简称S锁，顾名思义，共享锁就是多个事务对于同一数据可以共享一把锁，都能访问到数据，但是只能读不能修改。

排他锁又称为写锁，简称X锁，顾名思义，排他锁就是不能与其他所并存，如一个事务获取了一个数据行的排他锁，其他事务就不能再获取该行的其他锁，包括共享锁和排他锁，但是获取排他锁的事务是可以对数据就行读取和修改

**2.事务相关概念**

　1）事务的提交和回滚：COMMIT/ROLLBACK

　2）事务的开始和结束

　开始事务：连接到数据库，执行DML、DCL、DDL语句

　结束事务：1. 执行DDL(例如CREATE TABLE),DCL(例如GRANT),系统自动执行

　　　　　　2. 执行COMMIT/ROLLBACK

　　　　　　3. 退出/断开数据库的连接自动执行COMMIT语句

　　　　　　4. 进程意外终止，事务自动rollback

　　　　　　5. 事务COMMIT时会生成一个唯一的系统变化号（SCN）保存到事务表

　　3）保存点（savepoint）： 可以在事务的任何地方设置保存点，以便ROLLBACK

　　4）事务的四个特性ACID :

　　　　1. Atomicity（原子性）: 事务中sql语句不可分割，要么都做，要么都不做

　　　　2. Consistency(一致性) ： 指事务操作前后，数据库中数据是一致的，数据满足业务规则约束（例如账户金额的转出和转入），与原子性对应。

　　　　3. Isolation（隔离性）：多个并发事务可以独立运行，而不能相互干扰，一个事务修改数据未提交前，其他事务看不到它所做的更改。

　　　　4. Durability（持久性）：事务提交后，数据的修改是永久的。

　　5） 死锁：当两个事务相互等待对方释放资源时，就会形成死锁，下面章节详细分析

**3 oracle事务隔离级别**

（1）两个事务并发访问数据库数据时可能存在的问题

**@幻想读：**事务T1读取一条指定where条件的语句，返回结果集。此时事务T2插入一行新记录并commit，恰好满足T1的where条件。然后T1使用相同的条件再次查询，结果集中可以看到T2插入的记录，这条新纪录就是幻想。

**@不可重复读取：**事务T1读取一行记录，紧接着事务T2修改了T1刚刚读取的记录并commit，然后T1再次查询，发现与第一次读取的记录不同，这称为不可重复读。

**@脏读：**事务T1更新了一行记录，还未提交所做的修改，这个T2读取了更新后的数据，然后T1执行回滚操作，取消刚才的修改，所以T2所读取的行就无效，也就是脏数据。

**4 oracle事务隔离级别**

oracle支持的隔离级别：（不支持脏读）

READ COMMITTED--不允许脏读，允许幻想读和不可重复读

SERIALIZABLE--以上三种都不允许

sql标准还支持READ UNCOMMITTED (三种都允许)和 REPEATABLE READ（不允许不可重复读和脏读，只允许幻想读）。以上区别在下面章节事务建立，隔离级别分析中说明

**5 事务相关语句**

（1）事务相关语句概括

SET TRANSACTION----设置事务属性

SET CONSTRAINT -----设置约束

SAVEPOINT ------------建立存储点

RELEASE SAVEPOINT --释放存储点

ROLLBACK---------------回滚

COMMIT------------------提交

**（2）建立事务、隔离级别分析**

1）建立事务

SET TRANSACTION READ ONLY--事务中不能有任何修改数据库中数据的操作语句，这包括 insert、update、delete、create语句

SET TRANSACTION READ WRITE--默认设置,该选项表示在事务中可以有访问语句、修改语句

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE--serialzable可以执行DML操作

注意：这些语句是互斥的，不能够同时设置两个或者两个以上的选项

2）read only

eg：

set transaction read only;

select \* from student;

结果：

执行：

update student set name='小丸子' where id='ccc';

结果：

在行 3 上开始执行命令时出错:

update student set name='小丸子' where id='ccc'

错误报告:

SQL 错误: ORA-01456: 不能在 READ ONLY 事务处理中执行插入/删除/更新操作

01456. 00000 - "may not perform insert/delete/update operation inside a READ ONLY transaction"

\*Cause: A non-DDL insert/delete/update or select for update operation

was attempted

\*Action: commit (or rollback) transaction, and re-execute

3） read write

eg：

set transaction read write;

select \* from student;

update student set name='小丸子' where id='ccc';

select \* from student;

结果：

transaction READ 成功。

1 行已更新。

结论：允许读写

4）isolation level read committed（可幻读和重复读）

a:建立两个事务如下：

事务1：

set transaction read write;

select \* from student;

事务2：

set transaction isolation level read committed;

select \* from student;

b:在事务1中修改某行数据并commit

update student set score=88 where id='fff';

commit;

c:在事务2中查询结果如下：

select \* from student;

结果：

结论：事务2的隔离级别为 isolation level read committed, 支持不可重复读

d:在事务1中插入一行数据，并提交

insert into student values('iii','小梦','1',3,4,10);

e:在事务2中查看：

select \* from student;

结论：事务2隔离级别为 isolation level read committed, 允许幻想读

5） isolation level serializable

a:建立两个事务如下：

事务1：

set transaction read write;

select \* from student;

事务2：

set transaction isolation level serializable;

select \* from student;

b:在事务1中修改某行数据并commit

update student set score=88 where id='fff';

commit;

c:在事务2中查询结果如下：

select \* from student;

结果：

结论：事务2的隔离级别为 isolation level serializable, 不支持不可重复读

d: 在事务1中插入一行数据，并提交

insert into student values('iii','小梦','1',3,4,10);

e:在事务2中查看：

select \* from student;

结论：事务2的隔离级别为 isolation level serializable, 不支持幻想读

5 ORACLE锁机制

1）说明,锁是一种机制，多个事务同时访问一个数据库对象时，该机制可以实现对并发的控制

2） oracle中锁的类别

　　　　1.DDL锁： oracle自动的施加和释放

　　　　2.DML锁：事务开始时施加，使用Commit后者Rollback被释放、

　　　　3.内部锁： 由oracle自己管理以保护内部数据库结构

3）oracle锁的粒度

　　　　1. 行级锁（TX）：阻止该行上的DML操作，直到Commit或者Rollback

　　　　2. 表级锁（TM）：

　　　　3. 数据库级锁: eg： 将数据库锁定为只读模式 alter database open read only;

　　　　　　　　　　　 eg: 将数据库设置为限制模式（导入导出数据库时使用）：alter system enable restricted session;

**6 锁的模式**

1)概括说明：1. 数字越大，级别越高

2）eg：

lock table student in row share mode;

lock table student in row exclusive mode; --用于行的修改

lock table student in share mode; --阻止其他DML操作

lock table student in share row exclusive mode; 　--阻止其他事务操作

lock table student in exclusive mode; --独立访问使用

**7 锁查看**

1）概括

SELECT \* FROM V$SESSION;--查看会话和锁的信息

SELECT \* FROM V$SESSION\_WAIT;--查看等待的会话信息

SELECT \* FROM V$LOCK;--系统中所有锁

SELECT \* FROM V$LOCKED\_OBJECT;--系统中DML锁

2)eg:

set transaction read write;

insert into student values ('jjj','小欣','1',3,4,90);

SELECT \* FROM V$LOCked\_object;

分析：1. locked\_mode:3 ,因为我们执行的是insert， 因而是行级排他锁 row exclusive mode

**8 死锁**

（1）说明

当两个用户希望持有对方的资源时就会发生死锁。即两个用户互相等待对方释放资源，oracle认定为产生了死锁，在这种情况下，将以牺牲一个用户作为代价，另一个用户继续执行，牺牲的用户的事务将回滚。lORA-00060的错误并记录在数据库的日志文件alertSID.log中。同时在user\_dump\_dest下产生了一个跟踪文件，详细描述死锁的相关信息。

（2）死锁产生条件

　　1. Mutual exclusion（互斥）：资源不能被共享，只能由一个进程使用。

　　2. Hold and wait（请求并保持）：已经得到资源的进程可以再次申请新的资源。

　　3. No pre-emption（不可剥夺）：已经分配的资源不能从相应的进程中被强制地剥夺。

　　4. Circular wait（循环等待条件）：系统中若干进程组成环路，该环路中每个进程都在等待相邻进程正占用的资源。

（3）死锁模拟

　　student table如下

1）开两个进程（此处使用sqldeveloper模拟）建立两个事务

事务1：

set transaction read write;

事务2：

set transaction read write;

2）t1 时刻 事务1和事务2中分别执行如下语句

事务1：

update student set classno=8 where id='iii';

事务2：

update student set score=score+10 where id='jjj';

结果如下：

1行已更新

1行已更新

3）t2时刻事务1和事务2中分别执行如下语句

事务1：

update student set score=score+20 where id='jjj';

事务2：

update student set classno=9 where id='iii';

错误报告:

SQL 错误: ORA-00060: 等待资源时检测到死锁

00060. 00000 - "deadlock detected while waiting for resource"

\*Cause: Transactions deadlocked one another while waiting for resources.

\*Action: Look at the trace file to see the transactions and resources

involved. Retry if necessary.

注意：ScriptRunner处红色的叉，出于等待状态

4） commit 事务2，事务1结果如下：

注意：用时38.048s， 事务2commit前等待的时间

**9 解决死锁冲突**

1）执行commit或者rollback结束事务

2）终止会话

还是借用3中死锁的例子

在等待资源时执行，查找阻塞会话

select sid,serial#,username from v$session where sid in (select blocking\_session from v$session);

执行：

alter system kill session '423,896';

结果：

在行 10 上开始执行命令时出错:

alter system kill session '423,896'

错误报告:

SQL 错误: ORA-00027: 无法终止当前会话

00027. 00000 - "cannot kill current session"

\*Cause: Attempted to use ALTER SYSTEM KILL SESSION to kill the current

session.

\*Action: None.

3）解决方法：另起一个session，关闭当前session（If it is necessary to kill the current session, do so from another session.）

另起线程执行上面的语句：

alter system kill session '423,896';

结果如下：（session kill成功，死锁解除，事务1更新成功）

**10 事务和死锁预防总结**

（1）避免应用不运行长事务。

（2）经常提交以避免长时间锁定行。

（3）避免使用LOCK命令锁定表。

（4）在非高峰期间执行DDL操作，在非高峰期间执行长时间运行的查询或事务。

**11 锁是并发访问的时候用于保护不共享资源不被同时并发修改的机制。**

（1）oracle锁分为DML锁，DDL锁，内部锁和latch

（2）DML锁确保一次只能只有一个人修改某一行（TX锁），而且正在处理一个表时别人不能删除（TM锁）。

（3）DDL锁，在DDL操作是系统会自动为对象加上DDL锁，保护这些对象不被其他会话锁修改。

（4）latch是轻量级的串行化设备，用于协调对共享数据结构、对象、文件的多用户访问，一般都是保护共享内存结构使用的锁，在此不做讨论。

**12 一般的锁管理器工作过程：**

1.找到想要锁定的那一行地址

2.在锁管理器排队

3.锁定列表

4.搜索列表，查看别人是否锁定这一行

5.在列表中创建一个新的条目，表明已经锁定这一行

6.对列表解锁

接下里修改，之后提交修改后，继续过程：

7.再次排队

8.锁住锁的列表

9.在这个列表中锁定，并释放所有的锁

10.对列表解锁

**13 oracle锁管理方式：**

（1）通常lock有三个组件：Resource Structure(资源)、Lock Structure(锁）和Enqueue(排队机制）。Resource和lock是数据结构,而Enqueue是算法。

（2）Resource Structure

每一个需要并发控制的资源都有用这个数据结构来描述，先关的成员为:owner、waiter和converter,这是三个指针，分别指向3个由Lock Structure组成的链表。

（3）Lock Structure

每当进程需要访问共享资源时，必须先“锁定”该资源，这个动作实际上是从内存中申请一个Lock Structure，

，在其中记录“锁模式、进程ID”等重要信息。然后看是否立即能够获得资源的访问权，如果不能的话将这个Lock structure挂到Resource Structure的Waiter链表中，如果能够获得，则把Lock Structure的owner链表中。

（4）最常用的锁模式

Share 拥有这对资源进行只读访问，允许其他用户并发只读访问

Exclusive 拥有者对资源进行修改访问，不允许其他用户并发访问

（5）Enqueue 算法

Lock使用的是Enqueue算法，可以理解为“先入先出队列”，如果进程的锁定请求不能满足，该进程的Lock Structure就被加到Waiter链表的末端。当占用进程释放锁时，会检查Waiter和Converter队列，把锁分配给先入对的请求者。converter和waiter两个等待队列，算法的有些区别：如果某个操作先后需要2中不同模式的锁，比如先是share mode然后是exclusive mode,则进程会先请求share mode 后获得lock structure会挂在owner队列上，当需要exclusive mode锁时，进程先释放share mode的锁，然后再次申请exclusive mode的锁，但是可能无法立即获得，这时请求会挂在converter队列下，converter队列会被优先于waiter队列处理。

**14 oracle行级锁机制**

（1）首先明白三个概念：

ITL：每个数据块的头部有一个叫做ITL的数据结构，用于记录那些事务修改了这个数据块的内容。

记录头ITL索引：每条记录的记录头部有一个字段，用于记录ITL表项号，可以看做指向ITL表的指针

TX锁，事务锁

TM锁：保护表或视图定义不被修改的锁

（2）当一个事务开始时，必须申请一个TX锁，这种锁保护资源是回滚段、回滚段数据块，因此这个这个申请意味着：用户进程必须先申请到回滚段资源后才能开始一个事务，才能执行DML语句修改数据。

申请到回滚段资源后，用户事务就可以开始修改数据了，事务信息可在v$transaction中查到，在修改数据表的记录时，需要遵守如下操作顺序：

首先获得这个表的TM锁，这个锁用于保护事务执行过程中其他用户不能修改表结构；

事务修改某个数据块记录时，首先需要在改数据块块头的ITL表中申请一个空闲表项，并在其中记录事务号，实际就是在记录这个事物要使用的回滚段地址；

事务修改该数据块的某条记录时，会设置该记录头部的ITL索引指向上一步申请到的表项，然后再修改记录内容，修改前先在回滚段对记录修改前的状态做一个拷贝，然后才能修改数据记录，这个拷贝用于以后的回滚、恢复和一致性读。当其他用户并发修改这条记录时，会根据记录头的ITL索引读取ITL表项内容，查看这个事务是否已经提交,如果没有提交，则这个用户的TX锁会等待前一个用户的TX锁的释放。

例如如下式转储的一个数据块的ITL信息：

Block header dump:  0×00411819

Object id on Block? Y

seg/obj: 0x10396  csc: 0×00.d62e7  itc: 2  flg: O  typ: 1 – DATA

fsl: 0  fnx: 0×0 ver: 0×01

Itl           Xid                  Uba         Flag  Lck        Scn/Fsc

0x01   0×0008.00b.0000029b  0x00c05271.006a.3c  —-    3  fsc 0×0000.00000000

0x02   0×0000.000.00000000  0×00000000.0000.00  —-    0  fsc 0×0000.00000000

seg/obj：seg/obj id

csc:clean scn

itc：itl slots的个数，此时多少个事务在对本data block进行操作

flg: 0=on the freelist

typ:数据块类型

fsl: itl tx freelist slot

fnx: dba of next block on freelist

Itl：interested transaction list index

Xid：transaction id

Uba：undo address

Flag：事务状态标志

Lck：事物所影响行的数量

（3）oracle 在对数据行锁定时，行指向事务ID的一个副本，事务ID存储在包含数据的块中，释放锁时，事务ID会保存下来，这个事务ID时事务特有的，表示了

回滚段号、槽和序列号，事务ID留在包含数据行的块中，可以告诉其他会话：一个会话拥有这个数据行。另一个会话会看到锁ID，由于锁ID表示一个事务，所以

可以很快的查看持有这个锁的事务是否活动的。如果锁不活动的，则允许会话访问这个数据，如果锁还是活动的，会话会要求一旦释放锁就得到通知。所以这需

要一个排队机制：请求锁的会话会排队，等待目前拥有这个锁的事务执行，然后的到这个数据。可以根据v$lock视图的lmode和request mode判断谁是owner、

waiter和converter

owner:lomode>0,request=0

waiter:lmode=0，request>0

converter:lmode>0,request>0

例如下试验可以清楚看到这些信息：

系统已更改。

SQL> create table t1 ( x int );

表已创建。

SQL> create table t2 ( x int );

表已创建。

SQL> insert into t1 values ( 1 );

已创建 1 行。

SQL> insert into t2 values ( 1 );

已创建 1 行。

SQL> select (select username

2                 from v$session

3                 where sid = v$lock.sid) username,

4         sid,

5         id1,

6         id2,

7         lmode,

8         request, block, v$lock.type

9    from v$lock

10   where sid = (select sid

11                  from v$mystat

12                 where rownum=1)

13  /

USERNAME  SID   ID1   ID2   LMODE  REQUEST    BLOCK    TYPE

——– —–  —-   —-  —–   ——-   —–    —-

SYS       13    66455   0     3        0         0      TM

SYS       13    66456   0     3        0         0      TM

SYS       13    589840  662   6        0         0      TX

SQL> select object\_name, object\_id

2    from user\_objects

3   where object\_name in (‘T1′,’T2′)

4  /

OBJECT\_NAME    OBJECT\_ID

————  ———

T1            66455

T2            66456

每个事务只能有一个TX锁，但是TM锁依照修改的对象个数而定，TM对应的ID1列就是DML锁定对象ID.

SQL> select username,

2         v$lock.sid,

3         trunc(id1/power(2,16)) rbs,

4         bitand(id1,to\_number(‘ffff’,'xxxx’))+0 slot,

5         id2 seq,

6         lmode,

7         request

8  from v$lock, v$session

9  where v$lock.type = ‘TX’

10    and v$lock.sid = v$session.sid

11    and v$session.username = USER;

USERNAME  SID    RBS    SLOT   SEQ   LMODE  REQUEST

——– —– ——  —–  —— —— ———

SYS       13     9      16      662       6     0

SQL> select XIDUSN, XIDSLOT, XIDSQN  from v$transaction;

XIDUSN    XIDSLOT     XIDSQN

—— ———- ———-

9         16        662

oracle事务不同于其他数据库之处，不需要专门语句显示开始事务，事务会在修改数据的第一条语句处开始，但是一定要用commit或rollback事务。

**15 oracle的commit做了如下操作：**

为事务生成一个SCN

LGWR将所有余下的缓存重做日志条目写至磁盘，并把SCN记录到在线重做日志文件中

v$lock中记录着会话持有的锁，这些锁将被释放，而排队等待这些锁的每一个队列都会被唤醒

如果事务处理的某些块还在缓存中，则会快速的模式访问并清除

Itl           Xid                  Uba         Flag  Lck        Scn/Fsc

0x01   0×0008.00b.0000029b  0x00c05271.006a.3c  C—    0  scn 0×0000.000d6470

0x02   0×0005.007.00000372  0x00c0dbca.006e.2f  –U-    1  fsc 0×0000.000d6584

如上的flag列，第一条ITL信息显示数据块当前事务信息已经被清除，第二个还未清除事务ITL信息标志为U；

oracle并发支持,实现了一种多版本体系，能够同时物化多个版本的数据，能够提供读一致性机制，数据读取器绝不会被写入器所阻塞，也就是写不会阻塞读。一种情况例外，那就是在分布式事务处理（2PC）期间。

另外，记住大多数DDL都带排它锁，有些DDL没有DDL锁，如create index idx on t(x) online;online关键字会改变建立索引的方法。oracle只会得到表上的TM锁，防止其他DDL发生，但是运行DML运行。oracle发生死锁的原因外键未加索引、位图索引发生更新，外键未加索引更新或删除父表都会对整个子表加锁

会话1：

create table p ( x int primary key );

create table c ( x references p );

insert into p values ( 1 );

insert into p values ( 2 );

commit;

insert into c values ( 2 );

会话2：

delete from p where x = 1;;

这个时候机会发生阻塞:

SQL> select

2        (select username from v$session where sid=a.sid) blocker,

3         a.sid,

4        ‘ is blocking ‘,

5         (select username from v$session where sid=b.sid) blockee,

6             b.sid

7    from v$lock a, v$lock b

8   where a.block = 1

9     and b.request > 0

10     and a.id1 = b.id1

11     and a.id2 = b.id2;

BLOCKER    SID ‘ISBLOCKING’  BLOCKEE     SID

——-  —–  ———   ——— ——-

SYS       142  is blocking  SYS          13

不需要对外键加索引的情况：

1、没有从父表删除行

2、没有更新父表的唯一键/主键值

3、没有从父表联结子表

## 定时任务

**1、创建定时任务**

DECLARE

JOB\_ID PLS\_INTEGER;

BEGIN

SYS.DBMS\_JOB.SUBMIT(JOB => JOB\_ID,

WHAT => 'PRO\_JOB;',

INTERVAL => 'TRUNC(SYSDATE) + 1 + 3 / (24)');

COMMIT;

END;

PRO\_JOB是你要执行的存储过程名称，多个存储过程已分号分割，注意最后面也有一个分号。

**2、查询定时任务**

SELECT \* FROM DBA\_JOBS;

SELECT \* FROM USER\_JOBS;

**3、手动执行定时任务**

BEGIN

DBMS\_JOB.RUN(643); --643为任务的ID

END;

**4 在特定时间间隔后,重复运行该任务**

SYSDATE+n’,n泛指一个以天为单位的时间间隔.eg:

**描述 Interval参数值**

每天运行一次 ‘SYSDATE+1′

每小时运行一次 ‘SYSDATE+1/24′

10分钟运行一次 ‘SYSDATE+10/(60\*24)’

每30秒运行一次 ‘SYSDATE+30/(60\*24\*60)’

每星期运行一次 ‘SYSDATE+7′

注:此任务表达式不能保证任务的下一次运行时间在特定的日期或者时间,只能够指定一个任务两次运行之间的时间间隔。

**5 在特定的日期和时间运行任务**

定时到特定日期或时间的任务,eg

**描述 Lnterval参数值**

每天午夜12点 ‘TRUNC(SYSDATE+1)’

每天早上8点30分 ‘TRUNC(SYSDATE+1)+(8\*60+30)/(24\*60)’

每星期二中午12点 ‘NEXT\_DAY(TRUNC(SYSDATE),”TUESDAY”)+12/24′

每个月第一天的午夜12点 ‘TRUNC(LAST\_DAY(SYSDATE)+1)’

每个季度最后一天的晚上11点 ‘TRUNC(ADD\_MONTHS(SYSDATE+2/24,3),’Q')-1/24′

每星期六,日早上6点10分 ‘TRUNC(LEAST(NEXT\_DAY(SYSDATE,”SATURDAY”),

NEXT\_DAY(SYSDATE,”SUNDAY”)))+(6\*60+10)/(24\*60)’

## 系统表

**1 表的说明**

-- DBA/ALL/USER/V\_$/GV\_$/SESSION/INDEX开头的绝大部分都是视图

-- DBA\_TABLES意为DBA拥有的或可以访问的所有的关系表。

-- ALL\_TABLES意为某一用户拥有的或可以访问的所有的关系表。

-- USER\_TABLES意为某一用户所拥有的所有的关系表。

-- 当某一用户本身就为数据库DBA时，DBA\_TABLES与ALL\_TABLES等价。

-- DBA\_TABLES >= ALL\_TABLES >= USER\_TABLES

-- 需要注意的是在ORACLE数据库中大小写是敏感的，而此三表中数据默认都是大写的，所以在进行查询的时候注意小写的数据可能会造成数据无法查到。

SELECT \* FROM dba\_views WHERE view\_name LIKE 'DBA%';

SELECT \* FROM dba\_views WHERE view\_name LIKE 'ALL%';

SELECT \* FROM dba\_views WHERE view\_name LIKE 'USER%';

SELECT \* FROM dba\_views WHERE view\_name LIKE 'V\_$%'; -- 针对某个实例的视图

SELECT \* FROM dba\_views WHERE view\_name LIKE 'GV\_$%'; -- 全局视图，针对多个实例环境

SELECT \* FROM dba\_views WHERE view\_name LIKE 'SESSION%';

SELECT \* FROM dba\_views WHERE view\_name LIKE 'INDEX%';

SELECT count(1) FROM dba\_tables;

SELECT count(1) FROM all\_tables;

SELECT count(1) FROM user\_tables;

**2 V$/GV$开头的绝大部分都是V\_$/GV\_$表的别名**

SELECT \* FROM dba\_synonyms WHERE synonym\_name LIKE 'V$%';

SELECT \* FROM dba\_synonyms WHERE synonym\_name LIKE 'GV$%';

**3 X$没有对应的X\_$**

SELECT \* FROM dba\_synonyms WHERE synonym\_name LIKE 'X$%';

**4 比较常用的DBA开头的视图**

**DBA\_\*意为DBA拥有的或可以访问的所有的对象。**

**ALL\_\*意为某一用户拥有的或可以访问的所有的对象。**

**USER\_\*意为某一用户所拥有的所有的对象。**

select \* from dba\_users; --数据库用户信息

select \* from dba\_roles; --角色信息

select \* from dba\_segments; --表段信息

select \* from dba\_extents; --数据区信息

select \* from dba\_objects; --数据库对象信息

select \* from dba\_lobs; --lob数据信息

select \* from dba\_tablespaces; --数据库表空间信息

select \* from dba\_data\_files; --数据文件设置信息

select \* from dba\_temp\_files; --临时数据文件信息

select \* from dba\_rollback\_segs; --回滚段信息

select \* from dba\_ts\_quotas; --用户表空间配额信息

select \* from dba\_free\_space; --数据库空闲空间信息

select \* from dba\_profiles; --数据库用户资源限制信息

select \* from dba\_sys\_privs; --用户的系统权限信息

select \* from dba\_tab\_privs; --用户具有的对象权限信息

select \* from dba\_col\_privs; --用户具有的列对象权限信息

select \* from dba\_role\_privs; --用户具有的角色信息

select \* from dba\_audit\_trail; --审计跟踪记录信息

select \* from dba\_stmt\_audit\_opts; --审计设置信息

select \* from dba\_audit\_object; --对象审计结果信息

select \* from dba\_audit\_session; --会话审计结果信息

select \* from dba\_indexes; --用户模式的索引信息

**5 比较常用的ALL开头的视图**

select \* from all\_users; --数据库所有用户的信息

select \* from all\_objects; --数据库所有的对象的信息

select \* from all\_def\_audit\_opts; --所有默认的审计设置信息

select \* from all\_tables; --所有的表对象信息

select \* from all\_indexes; --所有的数据库对象索引的信息

select \* from all\_tab\_comments; --查询所有用户的表,视图等

select \* from all\_col\_comments; --查询所有用户的表的列名和注释.

select \* from all\_tab\_columns; --查询所有用户的表的列名等信息(详细但是没有备注)

**6 比较常用的ALL开头的视图**

select \* from user\_objects; --用户对象信息

select \* from user\_source; --数据库用户的所有资源对象信息

select \* from user\_segments; --用户的表段信息

select \* from user\_tables; --用户的表对象信息

select \* from user\_tab\_columns; --用户的表列信息

select \* from user\_constraints; --用户的对象约束信息

select \* from user\_sys\_privs; --当前用户的系统权限信息

select \* from user\_tab\_privs; --当前用户的对象权限信息

select \* from user\_col\_privs; --当前用户的表列权限信息

select \* from user\_col\_comments; -- 查询本用户的表的列名和注释

select \* from user\_role\_privs; --当前用户的角色权限信息

select \* from user\_indexes; --用户的索引信息

select \* from user\_ind\_columns; --用户的索引对应的表列信息

select \* from user\_cons\_columns; --用户的约束对应的表列信息

select \* from user\_clusters; --用户的所有簇信息

select \* from user\_clu\_columns; --用户的簇所包含的内容信息

select \* from user\_cluster\_hash\_expressions; --散列簇的信息

**7 比较常用的V$开头的别名有**

select \* from v$database; --数据库信息

select \* from v$datafile; --数据文件信息

select \* from v$controlfile; --控制文件信息

select \* from v$logfile; --重做日志信息

select \* from v$instance; --数据库实例信息

select \* from v$log; --日志组信息

select \* from v$loghist; --日志历史信息

select \* from v$sga; --数据库SGA信息

select \* from v$parameter; --初始化参数信息

select \* from v$process; --数据库服务器进程信息

select \* from v$bgprocess; --数据库后台进程信息

select \* from v$controlfile\_record\_section; --控制文件记载的各部分信息

select \* from v$thread; --线程信息

select \* from v$datafile\_header; --数据文件头所记载的信息

select \* from v$archived\_log; --归档日志信息

select \* from v$archive\_dest; --归档日志的设置信息

select \* from v$logmnr\_contents; --归档日志分析的DML DDL结果信息

select \* from v$logmnr\_dictionary; --日志分析的字典文件信息

select \* from v$logmnr\_logs; --日志分析的日志列表信息

select \* from v$tablespace; --表空间信息

select \* from v$tempfile; --临时文件信息

select \* from v$filestat; --数据文件的I/O统计信息

select \* from v$undostat; --Undo数据信息

select \* from v$rollname; --在线回滚段信息

select \* from v$session; --会话信息

select \* from v$transaction; --事务信息

select \* from v$rollstat; --回滚段统计信息

select \* from v$pwfile\_users; --特权用户信息

select \* from v$sqlarea; --当前查询过的sql语句访问过的资源及相关的信息

select \* from v$sql; --与v$sqlarea基本相同的相关信息

select \* from v$sysstat; --数据库系统状态信息

**8 比较常用的SESSION开头的视图有**

select \* from session\_roles; --会话的角色信息

select \* from session\_privs; --会话的权限信息

**9 比较常用的INDEX开头的视图有**

select \* from index\_stats; --索引的设置和存储信息

**10 伪表，参考oracle 中 dual 详解：http://blog.csdn.net/ozhouhui/article/details/7935196**

**select \* from dual; --系统伪列表信息**

select sysdate from dual; --可将Sysdate视为一个其结果为当前日期和时间的函数，在任何可以使用Oracle函数的地方都可以使用Sysdate。也可以将它视为每个表的一个隐藏的列或伪列。

select current\_date from dual; --报告会话的时区中的系统日期。注：可以设置自己的时区，以区别于数据库的时区。

select SYSTIMESTAMP from dual; --报告TIMESTAMP数据类型格式的系统日期。

**11 系统权限**

-- GRANTEE 接受该权限的用户名

-- OWNER 对象的拥有者

-- GRANTOR 赋予权限的用户

SELECT \* FROM dba\_sys\_privs WHERE grantee = 'SYS';

SELECT \* FROM dba\_sys\_privs WHERE grantee = 'CONNECT';

SELECT \* FROM dba\_sys\_privs WHERE grantee = 'RESOURCE';

**12 角色权限**

-- 查看某个用户有哪些角色

select \* from dba\_role\_privs where grantee='SYS';

-- 查看某个角色被赋予了哪些用户

SELECT \* FROM dba\_role\_privs WHERE granted\_role = 'DBA';

-- 对象权限

SELECT \* FROM dba\_tab\_privs;

-- 授予某个用户某些角色

GRANT connect,resource TO 'USER';

GRANT dba to 'USER'; --给普通用户授予dba角色时，要重新连接才能生效

REVOKE dba to 'USER';

-- 直接授予某个用户某些权限

GRANT CREATE VIEW TO 'USER';

-- 查看某个系统用户是否有SYSDBA或者SYSOPER权限

-- oracle:DBA,SYSDBA,SYSOPER三者的区别：http://blog.chinaunix.net/uid-22457844-id-3045741.html

select \* from V$PWFILE\_USERS;

-- 锁定、解锁用户

SELECT \* FROM dba\_users WHERE username = 'SCOTT';

ALTER USER SCOTT account LOCK; --锁定用户

ALTER USER SCOTT account UNLOCK; --解锁用户

COMMIT;

**13 oracle10g 修改用户密码:**

http://blog.163.com/benbenfafa\_88/blog/static/64930162200972594612972/

-- User Default Password Check in Oracle 11g: http://www.dbform.com/html/2009/673.html

SELECT password FROM dba\_users WHERE username = 'SCOTT';

alter user SCOTT identified by new\_password; --修改用户密码

**14 SERVICE\_NAMES:**

**http://docs.oracle.com/database/121/REFRN/GUID-AC956707-D568-4F8A-BF2E-99BA41E0A64F.htm#REFRN10194**

SELECT \* FROM global\_name; -- 查看oracle的全局数据库名

SELECT \* FROM v$database; -- 查看数据库名 show parameter db\_name;

**15 数据库实例名对应着SID**

-- SID: http://docs.oracle.com/database/121/LADBI/glossary.htm#LADBI8021

-- linux下在配置oracle环境变量的情况可以使用 echo $ORACLE\_SID,如果没有可以使用ps -ef |grep oracle 来查询，结果中的xxxx就是对应的SID。

-- oracle 2548 1 0 Aug17 ? 00:00:00 ora\_pmon\_xxxx

-- 在windows环境下,oracle是以后台服务的方式被管理的,所以看"控制面板->管理工具->服务 里面的名称:"OracleServiceORCL",则ORCL就是sid;

SELECT \* FROM v$instance; --查看数据库实例名 show parameter instance\_name;

select instance from v$thread;

**16 show parameter是oracle的命令，不是标准SQL语句**

-- 可以在sqlplus或者pl/sql dev的命令窗口执行

-- show parameter aaaa;等价于SELECT \* FROM v$parameter WHERE name like '%aaaa%';

SELECT \* FROM v$parameter WHERE name like '%name%'; --等价于show parameter name;

select \* from v$parameter where name like '%db\_domain%'; --查询数据库域名

select username from all\_users where username like '%SCOTT%';

drop user SCOTT cascade;

commit;

-- ERROR at line 1:

-- ORA-01940: cannot drop a user that is currently connected

select 'ALTER SYSTEM KILL SESSION '||''''||SID||','||SERIAL#||''''||';' as KILLER from v$session where username='SCOTT';

-- KILLER

-- ALTER SYSTEM KILL SESSION '363,35';

-- ALTER SYSTEM KILL SESSION '364,51';

commit;

select \* from dba\_roles where role like '%CONNECT%';

drop role CONNECT;

commit;

select \* from dba\_tablespaces where tablespace\_name like 'EXAMPLE';

drop tablespace EXAMPLE including contents and datafiles cascade constraints ;

-- including contents 删除表空间中的内容，如果删除表空间之前表空间中有内容，而未加此参数，表空间删不掉，所以习惯性的加此参数。

-- including datafiles 删除表空间中的数据文件。

-- cascade constraints 同时删除 tablespace 中表的外键参照。

**17 如何创建dblink和视图**

-- http://docs.oracle.com/database/121/SQLRF/statements\_5006.htm#i2061505

-- 如果需要创建全局 DBLink，则需要先确定用户有创建 dblink 的权限：

select \* from user\_sys\_privs where privilege like upper('%DATABASE LINK%');

**18 如果没有，则需要使用 sysdba 角色给用户赋权：**

grant create public database link to dbusername;

**19 如果创建全局 dblink，必须使用 systm 或 sys 用户**，在 database 前加 public。

create /\* public \*/ database link dblink1

connect to dbusername identified by dbpassword

using '(DESCRIPTION =(ADDRESS\_LIST =(ADDRESS =(PROTOCOL = TCP)(HOST = 192.168.0.1)(PORT = 1521)))(CONNECT\_DATA =(SERVICE\_NAME = orcl)))';

**20 创建dblink后，就可以直接在dblink上创建视图**

create or replace view cptp as (select SJDH from dbusername.cptp@dblink1); drop view cptp;

**21 锁表查询SQL**

SELECT object\_name, machine, s.sid, s.serial#

FROM gv$locked\_object l, dba\_objects o, gv$session s

WHERE l.object\_id　= o.object\_id

AND l.session\_id = s.sid;

**22 解除锁表**

alter system kill session 'sid, serial#';

**23 备份某个表**

create table new\_table as select \* from old\_table;

**24 查看数据库是否在rac环境的集群中的**

show parameter cluster\_database;

select \* from v$parameter where name = 'cluster\_database';

**25 列操作**

-- 增加和修改列不需要加关键字COLUMN

-- 删除单列的话，一定要加COLUMN，删除多列的时候，不能加COLUMN关键字

-- 增加一列

alter table emp4 add test varchar2(10);

-- 修改一列

alter table emp4 modify test varchar2(20);

-- 删除一列

alter table emp4 drop column test;

-- 增加多列

alter table emp4 add (test varchar2(10),test2 number);

-- 修改多列

alter table emp4 modify (test varchar2(20),test2 varchar2(20));

-- 删除多列

alter table emp4 drop (test,test2);

**26 Windows下以管理员身份启动数据库**

net start oracleserviceorcl -- 后面的orcl是你安装的数据库实例名

net start oracleoradb11g\_home1tnslistener --非必须

-- linux下以sysdba用户登录，然后启动数据库

sqlplus / as sysdba

startup

-- sqlplus登陆方式

sqlplus / as sysdba --以操作系统权限认证的oracle sys管理员登陆

sqlplus /nolog

conn / as sysdba --以操作系统权限认证的oracle sys管理员登陆

sqlplus sys/password@orcl as sysdba --以sys用户登陆必须使用as sysdba

sqlplus /nolog --不在cmd或者teminal当中暴露密码的登陆方式

conn sys/password as sysdba

sqlplus --不显露密码的方式登陆

Enter user-name：sys

Enter password：password as sysdba --以sys用户登陆的话 必须要加上as sysdba子句

sqlplus scott/tiger@orcl --非管理员用户登陆

desc v$database; --查询v$database数据库的表结构

**27 在sqlplus中执行sql脚本，下面两种方式都可以**

START file\_name

@file\_name

**28 判断表是否存在，如果存在则删除**

declare

num number;

begin

select count(1) into num from all\_tables where TABLE\_NAME = 'EMP' and OWNER='SCOTT';

if num=1 then

execute immediate 'drop table EMP';

end if;

end;

/

--创建表

CREATE TABLE EMP

(EMPNO NUMBER(4) NOT NULL,

ENAME VARCHAR2(10),

JOB VARCHAR2(9),

MGR NUMBER(4),

HIREDATE DATE,

SAL NUMBER(7, 2),

COMM NUMBER(7, 2),

DEPTNO NUMBER(2));

可以将上述存储过程加载到每一个create table前面。

--ORACLE 判断序列是否存在,如果存在就删除

declare

V\_NUM number;

BEGIN

----多次删除时，每次都将v\_num设置成为0

V\_NUM := 0;

----判断序列 seq\_name\_1 是否存在（区分大小写）

select count(0) into V\_NUM from user\_sequences where sequence\_name = 'SEQ\_BUSINESS\_PROCESS\_INDEX\_ID';

----如果存在立即删除

if V\_NUM > 0 then

execute immediate 'DROP SEQUENCE SEQ\_BUSINESS\_PROCESS\_INDEX\_ID';

end if;

END;

-- 设置sqlplus模式显示总行数

show pagesize; --查看当前的pagesize

set pagesize 300;

-- 设置sqlplus模式显示行宽度

show linesize; --查看当前的linesize

set linesize 300;

-- 修改安装目录glogin.sql文件才能保证之前的设置永久生效

set pagesize 300;

set linesize 300;

**29 删除表对象**

select 'drop table '||segment\_name from dba\_segments where owner='VPMUSER' and segment\_type='TABLE';

-- 创建表对象

select

'create table '||segment\_name || ' as select \* from '||segment\_name ||'@DBLINK'

from dba\_segments where owner='VPMUSER' and segment\_type='TABLE';

-- 检查表是否完整导入

select segment\_name from dba\_segments@aaa where owner='VPMUSER' and segment\_type='TABLE'

and (segment\_name not like 'BIN$%'

and segment\_name not like '%201%')

minus

select segment\_name from dba\_segments where owner='VPMUSER' and segment\_type='TABLE' and segment\_name not like 'BIN$%'

**30 查询用户所有表的语句1**

select t.table\_name,t.comments from user\_tab\_comments t

--查询用户所有表的语句2:

select r1, r2, r3, r5

from (select a.table\_name r1, a.column\_name r2, a.comments r3

from user\_col\_comments a),

(select t.table\_name r4, t.comments r5 from user\_tab\_comments t)

where r4 = r1

-- 查找表的所有索引（包括索引名，类型，构成列）：

select t.\*,i.index\_type from user\_ind\_columns t,user\_indexes i where t.index\_name = i.index\_name and t.table\_name = i.table\_name and t.table\_name = 要查询的表

-- 查找表的主键（包括名称，构成列）：

select cu.\* from user\_cons\_columns cu, user\_constraints au where cu.constraint\_name = au.constraint\_name and au.constraint\_type = 'P' and au.table\_name = 要查询的表

-- 查找表的唯一性约束（包括名称，构成列）：

select column\_name from user\_cons\_columns cu, user\_constraints au where cu.constraint\_name = au.constraint\_name and au.constraint\_type = 'U' and au.table\_name = 要查询的表

-- 查找表的外键（包括名称，引用表的表名和对应的键名，下面是分成多步查询）：

select \* from user\_constraints c where c.constraint\_type = 'R' and c.table\_name = 要查询的表

-- 查询外键约束的列名：

select \* from user\_cons\_columns cl where cl.constraint\_name = 外键名称

-- 查询引用表的键的列名：

select \* from user\_cons\_columns cl where cl.constraint\_name = 外键引用表的键名

-- 查询表的所有列及其属性

select t.\*,c.COMMENTS from user\_tab\_columns t,user\_col\_comments c where t.table\_name = c.table\_name and t.column\_name = c.column\_name and t.table\_name = 要查询的表

--备份表数据

create table emp as select \* from scott.emp

--还原表数据

insert into emp select \* from scott.emp

--查看已经执行过的sql这些是存在共享池中的，用户名需要大写，必须具有DBA 的权限

select \* from v$sqlarea t where t.PARSING\_SCHEMA\_NAME in ('用户名') order by t.LAST\_ACTIVE\_TIME desc

--ORACLE11G 字符集更改（这里更改为AL32UTF8）

sqlplus sys as sysdba

--执行下面命令，有可能造成数据库中已有数据混乱的情况，所以在进行操作前，要进行数据库的备份操作

shutdown immediate;

STARTUP MOUNT;

ALTER SESSION SET SQL\_TRACE=TRUE;

ALTER SYSTEM ENABLE RESTRICTED SESSION;

ALTER SYSTEM SET JOB\_QUEUE\_PROCESSES=0;

ALTER SYSTEM SET AQ\_TM\_PROCESSES=0;

ALTER DATABASE OPEN;

ALTER DATABASE character set INTERNAL\_USE AL32UTF8;

ALTER SESSION SET SQL\_TRACE=FALSE;

shutdown immediate;

startup;

--察看 NLS\_LANG 信息：

SELECT parameter, value FROM v$nls\_parameters WHERE parameter LIKE '%CHARACTERSET';

UPDATE STAFF

SET MODIFY\_TIME = TO\_DATE('2016/04/22 00:01:00', 'yyyy/MM/dd hh24:mi:ss')

WHERE MODIFY\_TIME < TO\_DATE('2016/04/22 00:01:00', 'yyyy/MM/dd hh24:mi:ss');

UPDATE STAFF

SET MODIFY\_TIME = TO\_TIMESTAMP('19-03-2008 02:36:00.360000', 'dd-MM-yyyy hh24:mi:ss.ff')

WHERE STAFF\_ID = '01';

## 数据分区

**1 分区类别**

**（1）范围分区**

每个分区都由一个分区键值范围指定（对于一个以日期列作为分区键的表，“2005 年 1 月”分区包含分区键值为从“2005 年 1 月 1 日”到“2005 年 1 月 31 日”的行）。

**（2）列表分区**

每个分区都由一个分区键值列表指定（对于一个地区列作为分区键的表，“北美”分区可能包含值“加拿大”“美国”和“墨西哥”）。

**（3）散列分区**

将散列算法用于分区键来确定指定行所在的分区

**（4）组合范围散列分区**

范围和散列分区技术的组合，通过该组合，首先对表进行范围分区，然后针对每个单独的范**围分区再使用散列分区技术进一步细分。索引组织表只能进行范围分区。**

**（5）组合范围列表分区**

范围和列表分区技术的组合，通过该组合，首先对表进行范围分区，然后针对每个单独的范围分区再使用列表分区技术进一步细分。索引组织表可以按范围、列表或散列进行分区。

**2 范围分区**

（1）范围分区将数据基于范围映射到每一个分区，这个范围是你在创建分区时指定的分区键决定的。这种分区方式是最为常用的，并且分区键经常采用日期。举个例子：你可能会将销售数据按照月份进行分区。

（2）当使用范围分区时，请考虑以下几个规则：

@每一个分区都必须有一个VALUES LESS THEN子句，它指定了一个不包括在该分区中的上限值。分区键的任何值等于或者大于这个上限值的记录都会被加入到下一个高一些的分区中。

@所有分区，除了第一个，都会有一个隐式的下限值，这个值就是此分区的前一个分区的上限值。

@在最高的分区中，MAXVALUE被定义。MAXVALUE代表了一个不确定的值。这个值高于其它分区中的任何分区键的值，也可以理解为高于任何分区中指定的VALUE LESS THEN的值，同时包括空值。

（3）例一：

假设有一个CUSTOMER表，表中有数据200000行，我们将此表通过CUSTOMER\_ID进行分区，每个分区存储100000行，我们将每个分区保存到单独的表空间中，这样数据文件就可以跨越多个物理磁盘。下面是创建表和分区的代码，如下：

CREATE TABLE CUSTOMER

(

CUSTOMER\_ID NUMBER NOT NULL PRIMARY KEY,

FIRST\_NAME VARCHAR2(30) NOT NULL,

LAST\_NAME VARCHAR2(30) NOT NULL,

PHONE VARCHAR2(15) NOT NULL,

EMAIL VARCHAR2(80),

STATUS CHAR(1)

)

PARTITION BY RANGE (CUSTOMER\_ID)

(

PARTITION CUS\_PART1 VALUES LESS THAN (100000) TABLESPACE CUS\_TS01,

PARTITION CUS\_PART2 VALUES LESS THAN (200000) TABLESPACE CUS\_TS02

)

（4）例二：按时间划分

CREATE TABLE ORDER\_ACTIVITIES

(

ORDER\_ID NUMBER(7) NOT NULL,

ORDER\_DATE DATE,

TOTAL\_AMOUNT NUMBER,

CUSTOTMER\_ID NUMBER(7),

PAID CHAR(1)

)

PARTITION BY RANGE (ORDER\_DATE)

(

PARTITION ORD\_ACT\_PART01 VALUES LESS THAN (TO\_DATE('01- MAY -2003','DD-MON-YYYY')) TABLESPACEORD\_TS01,

PARTITION ORD\_ACT\_PART02 VALUES LESS THAN (TO\_DATE('01-JUN-2003','DD-MON-YYYY')) TABLESPACE ORD\_TS02,

PARTITION ORD\_ACT\_PART02 VALUES LESS THAN (TO\_DATE('01-JUL-2003','DD-MON-YYYY')) TABLESPACE ORD\_TS03

)

（5）例三：MAXVALUE

CREATE TABLE RangeTable

(

idd INT PRIMARY KEY ,

iNAME VARCHAR(10),

grade INT

)

PARTITION BY RANGE (grade)

(

PARTITION part1 VALUES LESS THEN (1000) TABLESPACE Part1\_tb,

PARTITION part2 VALUES LESS THEN (MAXVALUE) TABLESPACE Part2\_tb

);

**3 列表分区**

（1）该分区的特点是某列的值只有几个，基于这样的特点我们可以采用列表分区。

（2）例一

CREATE TABLE PROBLEM\_TICKETS

(

PROBLEM\_ID NUMBER(7) NOT NULL PRIMARY KEY,

DESCRIPTION VARCHAR2(2000),

CUSTOMER\_ID NUMBER(7) NOT NULL,

DATE\_ENTERED DATE NOT NULL,

STATUS VARCHAR2(20)

)

PARTITION BY LIST (STATUS)

(

PARTITION PROB\_ACTIVE VALUES ('ACTIVE') TABLESPACE PROB\_TS01,

PARTITION PROB\_INACTIVE VALUES ('INACTIVE') TABLESPACE PROB\_TS02

);

（3）例二

CREATE TABLE ListTable

(

id INT PRIMARY KEY ,

name VARCHAR (20),

area VARCHAR (10)

)

PARTITION BY LIST (area)

(

PARTITION part1 VALUES ('guangdong','beijing') TABLESPACE Part1\_tb,

PARTITION part2 VALUES ('shanghai','nanjing') TABLESPACE Part2\_tb

);

**4 散列分区**

（1）这类分区是在列值上使用散列算法，以确定将行放入哪个分区中。当列的值没有合适的条件时，建议使用散列分区。

散列分区为通过指定分区编号来均匀分布数据的一种分区类型，因为通过在I/O设备上进行散列分区，使得这些分区大小一致。

（2）例一：

CREATE TABLE HASH\_TABLE

(

COL NUMBER(8),

INF VARCHAR2(100)

)

PARTITION BY HASH (COL)

(

PARTITION PART01 TABLESPACE HASH\_TS01,

PARTITION PART02 TABLESPACE HASH\_TS02,

PARTITION PART03 TABLESPACE HASH\_TS03

)

**简写：**

CREATE TABLE emp

(

empno NUMBER (4),

ename VARCHAR2 (30),

sal NUMBER

)

PARTITION BY HASH (empno) PARTITIONS 8

STORE IN (emp1,emp2,emp3,emp4,emp5,emp6,emp7,emp8);

（3）hash分区最主要的机制是根据hash算法来计算具体某条纪录应该插入到哪个分区中,hash算法中最重要的是hash函数，Oracle中如果你要使用hash分区，只需指定分区的数量即可。建议分区的数量采用2的n次方，这样可以使得各个分区间数据分布更加均匀。

**5 组合分区**

（范围-散列分区，范围-列表分区）

**6 范围-散列分区**

create table graderecord

(

sno varchar2(10),

sname varchar2(20),

dormitory varchar2(3),

grade int

)

partition by range(grade)

subpartition by hash(sno,sname)

(

partition p1 values less than(75)

(

subpartition sp1,subpartition sp2

),

partition p2 values less than(maxvalue)

(

subpartition sp3,subpartition sp4

)

);

以grade划分范围，然后以sno和sname划分散列分区，当数据量大的时候散列分区则趋于“平均”。

**7 范围-列表分区**

create table MobileMessage

(

ACCT\_MONTH VARCHAR2(6), -- 帐期 格式：年月 YYYYMM

AREA\_NO VARCHAR2(10), -- 地域号码

DAY\_ID VARCHAR2(2), -- 本月中的第几天 格式 DD

SUBSCRBID VARCHAR2(20), -- 用户标识

SVCNUM VARCHAR2(30) -- 手机号码

)

partition by range(ACCT\_MONTH,AREA\_NO) subpartition by list(DAY\_ID)

(

partition p1 values less than('200705','012')

(

subpartition shangxun1 values('01','02','03','04','05','06','07','08','09','10'),

subpartition zhongxun1 values('11','12','13','14','15','16','17','18','19','20'),

subpartition xiaxun1 values('21','22','23','24','25','26','27','28','29','30','31')

),

partition p2 values less than('200709','014')

(

subpartition shangxun2 values('01','02','03','04','05','06','07','08','09','10'),

subpartition zhongxun2 values('11','12','13','14','15','16','17','18','19','20'),

subpartition xiaxun2 values('21','22','23','24','25','26','27','28','29','30','31')

),

partition p3 values less than('200801','016')

(

subpartition shangxun3 values('01','02','03','04','05','06','07','08','09','10'),

subpartition zhongxun3 values('11','12','13','14','15','16','17','18','19','20'),

subpartition xiaxun3 values('21','22','23','24','25','26','27','28','29','30','31')

)

)

## 十六、[Oracle数据库--读写分离架构](http://www.baidu.com/link?url=UswP4DmrSoRdGvz2G5VwA5u4M9LJN8s91uiimtAhkZy25rRruGEBI9dufBPget4L33G1S_HkZMv54V9nqDfZUa" \t "https://www.baidu.com/_blank)

**1 采用Oracle读写分离的思路**，Writer DB和Reader DB采用日志复制软件实现实时同步；

Writer DB负责交易相关的实时查询和事务处理，Reader DB负责只读接入，处理一些非实时的交易明细,报表类的汇总查询等。同时，为了满足高可用性和扩展性等要求，对读写端适当做外延，比如Writer DB采用HA或者RAC的架构模式，Reader DB可以采用多套，通过负载均衡或者业务分离的方式，有效分担读库的压力。

**2 对于Shared-nothing的数据库架构模式**，核心的一个问题就是读写库的实时同步；另外，虽然Reader DB只负责业务查询，但并不代表数据库在功能上是只读的。只读是从应用角度出发，为了保证数据一致和冲突考虑，因为查询业务模块可能需要涉及一些中间处理，如果需要在数据库里面处理(取决与应用需求和设计)，所以Reader DB在功能上仍然需要可写。

**3 下面谈一下数据同步的技术选型问题**

**（1）能实现数据实时同步的技术很多，**基于OS层(例如VERITAS VVR)，基于存储复制(中高端存储大多都支持)，基于应用分发或者基于数据库层的技术。因为数据同步可能并不是单一的DB整库同步，会涉及到业务数据选择以及多源整合等问题，因此OS复制和存储复制多数情况并不适合做读写分离的技术首选。

**（2）基于日志的Oracle复制技术，Or**acle自身组件可以实现，同时也有成熟的商业软件。选商业的独立产品还是Oracle自身的组件功能，这取决于多方面的因素。比如团队的相应技术运维能力、项目投入成本、业务系统的负载程度等。

**（3）采用Oracle自身组件功能，**无外乎Logical Standby、Stream以及11g的Physical Standby(Active Data Guard)，对比来说，Stream最灵活，但最不稳定，11g Physical Standby支持恢复与只读并行，但由于并不是日志的逻辑应用机制，在读写分离的场景中最为局限。如果技术团队对相关技术掌握足够充分，而选型方案的处理能力又能支撑数据同步的要求，采用Oracle自身的组件完全可行。

**（4）选择商业化的产品，更多出于稳定性、处理能力等考虑。**市面上成熟的Oracle复制软件也无外乎几种，无论是老牌的Shareplex，还是本土DSG公司的RealSync和九桥公司的DDS，或是Oracle新贵Goldengate，都是可供选择的目标。随着GoldenGate被Oracle收购和推广，个人认为GoldenGate在容灾、数据分发和同步方面将大行其道。当然，架构好一个可靠的分布式读写分离的系统，还需要应用上做大量设计，不在本文讨论范围内

# 十七、oracle高级复制之一—三种复制技术特点

**1 oracle提供三种高级备份功能**

（1）高级复制（Advanced Replication）

（2）流复制（Streams Replication）

（3）备库（Dataguard）

**2 备库dataguard**

（1）dataguard在高可用及容灾方面一般是dba的首选，不管是物理备用库（physical standby database）还是逻辑备用库（logical standby database），它们都具有一些共同的待征。

（2）配置和管理方面的成本：dataguard比stream replication简单方便；

（3）安全与稳定方面的成本：dataguard比stream replication稳定可靠。

**3 流复制**

（1）适用于如下情况：

@局部复制 stream可以只复制某些表或某些模式

@异构环境 充分利用现有的设备与技术

@远程容灾 stream对网络的要求较dataguard低

（2）stream replication有灵活的复制策略，不仅可以配置只复制某些表，还可以配置仅复制某些表上的ddl或dml，相比dataguard必须整个数据库复制而言，可以节省相当的存储投资。如果在异构环境，即不同的操作系统，那dataguard将会束手无策，非stream replication莫属，这样可以充分利用现有的环境，配置高用可方案，在异构环境，stream replication将会是advanced replication的强劲对手。

（3）stream replication传播的是经过logmnr挖掘并包装的逻辑改变记录（LCRs），相比dataguard传送archived redo log、advanced replication的mview log与mview刷新的方式，stream replication对网络的需求降低了很多，

**4 高级复制**

（1）advanced replication相对于dataguard，缺点是：配置与管理较复杂、安全与稳定性不够；优点：局部复制、异构环境等。advanced replication是一种相当成熟的技术，在许多关键系统中得到成功的运用，相对于9iR2推出的stream replication而言，双方适用的环境虽然相当，比如都可以进行局部复制、异构复制、远程容灾等，advanced replication目前在稳定性与安全性方面更经得起考验。

（2）对比stream replication与advanced replication底层的实现技术，stream replication在实时性、稳定性、高效率、低消耗（较少的cpu/network资源）等方面更有优势，但凡一些新推出的功能，都或多或少存在一些不确定的因素。

（3）在10gR1中，oracle针对目前stream replication存在的弱点进行了增强，不仅提供了从advanced replication迁移到stream replication的脚本，还提供了stream replication的配置与监控工具，stream replication在配置与管理方面必将智能化、简单化，担负起与shareplex争夺企业数据复制市场的重任。

**5 高级复制与流复制区别**

高级复制与Streams Replication的原理是完全不同的，Streams Replication可以到表，用户，数据库级别，但高级复制似乎只能到表一级。Streams Replication不是高级复制的升级版。异构环境下，oracle的高可用和容灾有高级复制和stream 复制两种，两种的异同点如下：

（1）高级复制是基于触发器（trigger）原理，而stream是基于日志挖掘原理，因此stream复制对源数据库的性能影响更小，但实时性不如高级复制。

（2）高级复制复制的对象是基于数据库目标（object）的，如表、索引和存储过程，而stream复制可以针对表、方案（schema）和整个数据库，因此如果出于容灾整个数据库的考虑，stream复制的配置相对简单。

（3）高级复制是一种相当成熟的技术，在许多关键系统中得到成功的运用，相对于9iR2推出的stream复制，高级复制目前在稳定性与安全性方面更经得起考验。

（4）从发展的角度看，流的应用会越来越多，从oracle10g，oracle公司提供了从高级复制向流复制移植的工具，可以看出，oracle公司会更偏重于基于流的新技术。

（5）由于高级复制是基于触发器的，因此所有的复制对象结构（ddl）的改变，都必须通过oracle提供的复制包来实施，和应用结合的比较紧，更适合于开发者使用，而流复制则更适合dba来实施。

（6）流复制支持双向数据复制，而高级复制会有冲突；

（7）流复制支持异构数据库复制，而没有资料说明高级复制也有相同功能；

（8）两种实际使用来看，streams复制需要更少的带宽，2m带宽，如果 streams复制不行，高级复制大概更没戏，但是用streams最好别网络断线时间过长，不知道是bug还是oracle没考虑这种情况，如果复制停 顿一段时间，再恢复正常，大概是队列表中消息太多了，入队出队都很慢，非线性增长啊，这样就需要不短的一段时间来同步数据，高级复制就没这种状况。

## 十八、Oracle引擎

**1 oracle中不存在引擎的概念，数据处理大致可以分成两大类**

**联机事务处理OLTP（on-line transaction processing）**

**联机分析处理OLAP（On-Line Analytical Processing）**

**2 OLTP**

传统的关系型数据库的主要应用，主要是基本的、日常的事务处理，例如银行交易。强调数据库内存效率，强调内存各种指标的命令率，强调绑定变量，强调并发操作；

**3 OLAP**

OLAP 系统则强调数据分析，强调SQL执行市场，强调磁盘I/O，强调分区等。数据仓库系统的主要应用，支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果。

## 十九、Oracle分表分库问题

**1、事务问题**

解决事务问题目前有两种可行的方案，分布式事务和通过应用程序与数据库共同控制实现事务下面对两套方案进行一个简单的对比。

**方案一：使用分布式事务**

优点：交由数据库管理，简单有效

缺点：性能代价高，特别是shard越来越多时

**方案二：由应用程序和数据库共同控制**

原理：将一个跨多个数据库的分布式事务分拆成多个仅处 于单个数据库上面的小事务，并通过应用程序来总控 各个小事务。

优点：性能上有优势

缺点：需要应用程序在事务控制上做灵活设计。如果使用 了spring的事务管理，改动起来会面临一定的困难。

**2、跨节点Join的问题**

只要是进行切分，跨节点Join的问题是不可避免的。但是良好的设计和切分却可以减少此类情况的发生。解决这一问题的普遍做法是分两次查询实现。在第一次查询的结果集中找出关联数据的id,根据这些id发起第二次请求得到关联数据。

**3、跨节点的count,order by,group by以及聚合函数问题**

这些是一类问题，因为它们都需要基于全部数据集合进行计算。多数的代理都不会自动处理合并工作。

**解决方案：**与解决跨节点join问题的类似，分别在各个节点上得到结果后在应用程序端进行合并。和join不同的是每个结点的查询可以并行执行，因此很多时候它的速度要比单一大表快很多。但如果结果集很大，对应用程序内存的消耗是一个问题。

**4、数据迁移，容量规划，扩容等问题**

来自淘宝综合业务平台团队，它利用对2的倍数取余具有向前兼容的特性（如对4取余得1的数对2取余也是1）来分配数据，避免了行级别的数据迁移，但是依然需要进行表级别的迁移，同时对扩容规模和分表数量都有限制。总得来说，这些方案都不是十分的理想，多多少少都存在一些缺点，这也从一个侧面反映出了Sharding扩容的难度。

**5、分布式事务**

**（1）两阶段提交**

**优点:**基于两阶段提交，最大限度地保证了跨数据库操作的“原子性”，是分布式系统下最严格的事务实现方式。实现简单，工作量小。由于多数应用服务器以及一些独立的分布式事务协调器做了大量的封装工作，使得项目中引入分布式事务的难度和工作量基本上可以忽略不计。

**缺点:**系统“水平”伸缩的死敌。基于两阶段提交的分布式事务在提交事务时需要在多个节点之间进行协调,最大限度地推后了提交事务的时间点，客观上延长了事务的执行时间，这会导致事务在访问共享资源时发生冲突和死锁的概率增高，随着数据库节点的增多，这种趋势会越来越严重，从而成为系统在数据库层面上水平伸缩的"枷锁"， 这是很多Sharding系统不采用分布式事务的主要原因。

**（2）基于Best Efforts 1PC模式的事务**

参考spring-data-neo4j的实现。鉴于Best Efforts 1PC模式的性能优势，以及相对简单的实现方式，它被大多数的sharding框架和项目采用

**（3）事务补偿（幂等值）**

对于那些对性能要求很高，但对一致性要求并不高的系统，往往并不苛求系统的实时一致性，只要在一个允许的时间周期内达到最终一致性即可，这使得事务补偿机制成为一种可行的方案。事务补偿机制最初被提出是在“长事务”的处理中，但是对于分布式系统确保一致性也有很好的参考意义。笼统地讲，与事务在执行中发生错误后立即回滚的方式不同，事务补偿是一种事后检查并补救的措施，它只期望在一个容许时间周期内得到最终一致的结果就可以了。事务补偿的实现与系统业务紧密相关，并没有一种标准的处理方式。一些常见的实现方式有：对数据进行对帐检查;基于日志进行比对;定期同标准数据来源进行同步，等等。

1. **ID问题**
2. **一旦数据库被切分到多个物理结点上**，我们将不能再依赖数据库自身的主键生成机制。一方面，某个分区数据库自生成的ID无法保证在全局上是唯一的；另一方面，应用程序在插入数据之前需要先获得ID,以便进行SQL路由.一些常见的主键生成策略

**（2）UUID**

使用UUID作主键是最简单的方案，但是缺点也是非常明显的。由于UUID非常的长，除占用大量存储空间外，最主要的问题是在索引上，在建立索引和基于索引进行查询时都存在性能问题。结合数据库维护一个Sequence表。此方案的思路也很简单，在数据库中建立一个Sequence表，表的结构类似于：

CREATE TABLE `SEQUENCE` (

`table\_name` varchar(18) NOT NULL,

`nextid` bigint(20) NOT NULL,

PRIMARY KEY (`table\_name`)

) ENGINE=InnoDB

每当需要为某个表的新纪录生成ID时就从Sequence表中取出对应表的nextid,并将nextid的值加1后更新到数据库中以备下次使用。此方案也较简单，但缺点同样明显：由于所有插入任何都需要访问该表，该表很容易成为系统性能瓶颈，同时它也存在单点问题，一旦该表数据库失效，整个应用程序将无法工作。有人提出使用Master-Slave进行主从同步，但这也只能解决单点问题，并不能解决读写比为1:1的访问压力问题。

**（3）Twitter的分布式自增ID算法Snowflake**

在分布式系统中，需要生成全局UID的场合还是比较多的，twitter的snowflake解决了这种需求，实现也还是很简单的，除去配置信息，核心代码就是毫秒级时间41位 机器ID 10位 毫秒内序列12位。

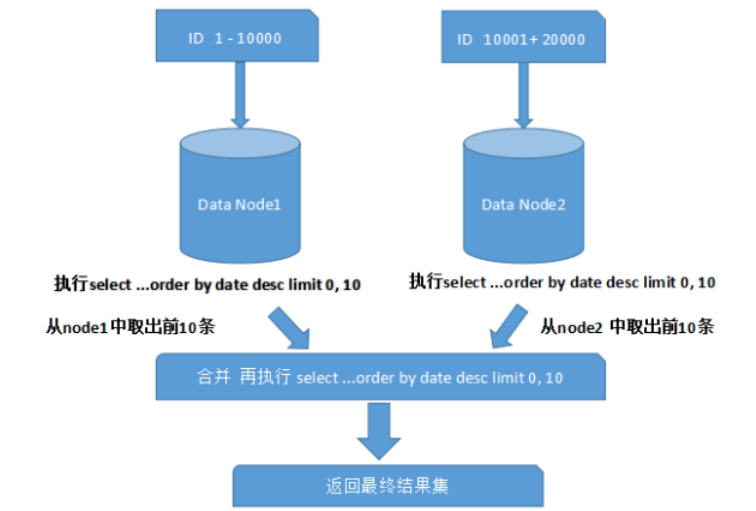
\* 10---0000000000 0000000000 0000000000 0000000000 0 --- 00000 ---00000 ---000000000000

在上面的字符串中，第一位为未使用（实际上也可作为long的符号位），接下来的41位为毫秒级时间，然后5位datacenter标识位，5位机器ID（并不算标识符，实际是为线程标识），然后12位该毫秒内的当前毫秒内的计数，加起来刚好64位，为一个Long型。

这样的好处是，整体上按照时间自增排序，并且整个分布式系统内不会产生ID碰撞（由datacenter和机器ID作区分），并且效率较高，经测试，snowflake每秒能够产生26万ID左右，完全满足需要。

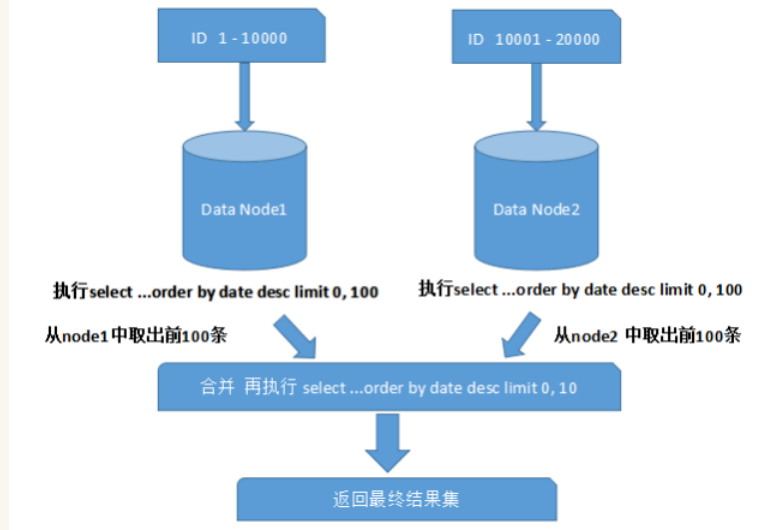
**7、跨分片的排序分页**

一般来讲，分页时需要按照指定字段进行排序。当排序字段就是分片字段的时候，我们通过分片规则可以比较容易定位到指定的分片，而当排序字段非分片字段的时候，情况就会变得比较复杂了。为了最终结果的准确性，我们需要在不同的分片节点中将数据进行排序并返回，并将不同分片返回的结果集进行汇总和再次排序，最后再返回给用户。如下图所示：上面图中所描述的只是最简单的一种情况（取第一页数据），看起来对性能的影响并不大。但是，如果想取出第10页数据，情况又将变得复杂很多，如下图所示：



有些读者可能并不太理解，为什么不能像获取第一页数据那样简单处理（排序取出前10条再合并、排序）。其实并不难理解，因为各分片节点中的数据可能是随机的，为了排序的准确性，必须把所有分片节点的前N页数据都排序好后做合并，最后再进行整体的排序。很显然，这样的操作是比较消耗资源的，用户越往后翻页，系统性能将会越差。

那如何解决分库情况下的分页问题呢？有以下几种办法：



如果是在前台应用提供分页，则限定用户只能看前面n页，这个限制在业务上也是合理的，一般看后面的分页意义不大（如果一定要看，可以要求用户缩小范围重新查询）。如果是后台批处理任务要求分批获取数据，则可以加大page size，比如每次获取5000条记录，有效减少分页数（当然离线访问一般走备库，避免冲击主库）。分库设计时，一般还有配套大数据平台汇总所有分库的记录，有些分页查询可以考虑走大数据平台。

**8、分库策略**

分库维度确定后，如何把记录分到各个库里呢?

一般有两种方式：

**根据数值范围，**比如用户Id为1-9999的记录分到第一个库，10000-20000的分到第二个库，以此类推。

**根据数值取模**，比如用户Id mod n，余数为0的记录放到第一个库，余数为1的放到第二个库，以此类推。

**优劣比较：**

（1）评价指标按照范围分库按照Mod分库

（2）库数量前期数目比较小，可以随用户/业务按需增长前期即根据mode因子确定库数量，数目一般比较大

（3）访问性能前期库数量小，全库查询消耗资源少，单库查询性能略差前期库数量大，全库查询消耗资源多，单库查询性能略好

（4）调整库数量比较容易，一般只需为新用户增加库，老库拆分也只影响单个库困难，改变mod因子导致数据在所有库之间迁移

（5）数据热点新旧用户购物频率有差异，有数据热点问题新旧用户均匀到分布到各个库，无热点

（6）实践中，为了处理简单，选择mod分库的比较多。同时二次分库时，为了数据迁移方便，一般是按倍数增加，比如初始4个库，二次分裂为8个，再16个。这样对于某个库的数据，一半数据移到新库，剩余不动，对比每次只增加一个库，所有数据都要大规模变动。

（7）补充下，mod分库一般每个库记录数比较均匀，但也有些数据库，存在超级Id，这些Id的记录远远超过其他Id，比如在广告场景下，某个大广告主的广告数可能占总体很大比例。如果按照广告主Id取模分库，某些库的记录数会特别多，对于这些超级Id，需要提供单独库来存储记录。

**9、分库数量**

分库数量首先和单库能处理的记录数有关，一般来说，Mysql 单库超过5000万条记录，Oracle单库超过1亿条记录，DB压力就很大(当然处理能力和字段数量/访问模式/记录长度有进一步关系)。在满足上述前提下，如果分库数量少，达不到分散存储和减轻DB性能压力的目的；如果分库的数量多，好处是每个库记录少，单库访问性能好，但对于跨多个库的访问，应用程序需要访问多个库，如果是并发模式，要消耗宝贵的线程资源；如果是串行模式，执行时间会急剧增加。最后分库数量还直接影响硬件的投入，一般每个分库跑在单独物理机上，多一个库意味多一台设备。所以具体分多少个库，要综合评估，一般初次分库建议分4-8个库。

**10、路由透明**

分库从某种意义上来说，意味着DB schema改变了，必然影响应用，但这种改变和业务无关，所以要尽量保证分库对应用代码透明，分库逻辑尽量在数据访问层处理。当然完全做到这一点很困难，具体哪些应该由DAL负责，哪些由应用负责，这里有一些建议：对于单库访问，比如查询条件指定用户Id，则该SQL只需访问特定库。此时应该由DAL层自动路由到特定库，当库二次分裂时，也只要修改mod 因子，应用代码不受影响。对于简单的多库查询，DAL负责汇总各个数据库返回的记录，此时仍对上层应用透明。

**11、使用框架还是自主研发**

目前市面上的分库分表中间件相对较多，其中基于代理方式的有MySQL Proxy和Amoeba，基于Hibernate框架的是Hibernate Shards，基于jdbc的有当当sharding-jdbc，基于mybatis的类似maven插件式的有蘑菇街的蘑菇街TSharding，通过重写spring的ibatis template类是Cobar Client，这些框架各有各的优势与短板，架构师可以在深入调研之后结合项目的实际情况进行选择，但是总的来说，我个人对于框架的选择是持谨慎态度的。一方面多数框架缺乏成功案例的验证，其成熟性与稳定性值得怀疑。另一方面，一些从成功商业产品开源出框架（如阿里和淘宝的一些开源项目）是否适合你的项目是需要架构师深入调研分析的。当然，最终的选择一定是基于项目特点、团队状况、技术门槛和学习成本等综合因素考量确定的。

## 二十、Oracle内核

**1 ORACLE内核参数**

服务器内存为4G的情况下

修改/etc/sysctl.conf文件 (ROOT账户)

**kernel.shmmax = 2147483648**

//公式:2G\*1024\*1024\*1024=2147483648(字节)

//表示最大共享内存，如果小的话可以按实际情况而定，一般为物理内存的一半（单位:字节）

**kernel.shmmni=4096**

//表示最小共享内存固定4096KB（由于32位操作系统默认一页为4K）

**kernel.shmall=1048576**

//公式:4G\*1024\*1024/4K = 1048576(页)

//表示所有内存大小（单位：页）

**kernel.sem=250 32000 100 128**

//4个参数依次是SEMMSL：每个用户拥有信号量最大数，SEMMNS：系统信号量最大数，SEMOPM：每次semopm系统调用操作数，SEMMNI：系统辛苦量集数最大数。这4个参数为固定内容大小

**fs.file-max=65536**

//file-max固定大小65536

**net.ipv4.ip\_local\_port\_range=1024 65000**

//ip\_local\_port\_range表示端口的范围，为指定的内容

以上步骤做完执行 /sbin/sysctl -p 使内核生效

**2 验证参数（root账户执行）：**

#/sbin/sysctl -a | grep shm

#/sbin/sysctl -a | grep sem

#/sbin/sysctl -a | grep file-max

#/sbin/sysctl -a | grep ip\_local\_port\_range

**3 与oracle相关的信号量和共享内存段参数**

一般unix系统中和信号量相关的是三个参数SEMMNI SEMMSL SEMMNS。他们相互关联决定系统可以分配的信号量。Oracle使用信号量完成内部进程之间的通信。关于共享内存段使用shmmx参数进行总体控制。它指定了系统可以分配的共享内存段最大大小，实际并没有分配那么多只是给出一个可以使用的最大限制。对于类核参数的修改必须要重新启动系统之后才会生效。

**4 出现信号量和共享内存段相关问题的情况**

oracle只有在startup nomount的时候才会请求os的这些资源，用于建立SGA和启动后台进程。有些情况下因为oracle崩溃之后os没有清除oracle分配的SGA，也可能造成共享内存段不足，需要人工清除。

**5 如何解决相关的问题**

你可以简单的修改init参数减少oracle对共享内存段和信号量的需求。对于控制信号量的三个参数SEMMNI SEMMSL SEMMNS 。最终可以使用的信号量由下面公式 提取 (semmsl \* semmni) 或者 semmns中最小的值。

**6 例如在linux下. 进入目录/proc/sys/kernel；用cat命令或more命令查看semaphore当前参数的值：**

cat sem

命令运行后将会出现如下的结果：

250 32000 32 128

其中, 250 是参数SEMMSL的值,32000是参数SEMMNS的值, 32是参数SEMOPM的值，而128则是参数SEMMNI的值。250\*128=32000

对于oracle7需要信号量的设置等于init中processes的设置。对于8i 9i需要等于processes\*2。

**7 对于信号量参数的设定一定要小心**，因为不正确的设置可能会让系统使用默认值。这个值一般比oracle系统要求的低。在HP unix上遇到过这样的问题，当时在参数配置的时候指定两个不同的sem-mni造成系统使用默认的设置。对于共享内存段，系统的设置至少要等于SGA的大小。

## 二十一、Oracle服务器的优化

**1 设置合适的SGA**

常常有人抱怨服务器硬件很好，但是Oracle就是很慢。很可能是内存分配不合理造成的。

1. **假设内存有512M**，这通常是小型应用。建议Oracle的SGA大约240M，其中：共享池（SHARED\_POOL\_SIZE）可以设置60M到80M，根据实际的用户数、查询等来定。数据块缓冲区可以大致分配120M-150M，8i下需要设置DB\_BLOCK\_BUFFERS，DB\_BLOCK\_BUFFER\*DB\_BLOCK\_SIZE等于数据块缓冲区大小。9i 下的数据缓冲区可以用db\_cache\_size来直接分配。

**(2 假设内存有1G，**Oracle 的SGA可以考虑分配500M：共享池分配100M到150M，数据缓冲区分配300M到400M。

**(3) 内存2G，SGA可以考虑分配1.2G**，共享池300M到500M，剩下的给数据块缓冲区。

**(4) 内存2G以上：**共享池300M到500M就足够啦，再多也没有太大帮助；(Biti\_rainy有专述)数据缓冲区是尽可能的大，但是一定要注意两个问题：一是要给操作系统和其他应用留够内存，二是对于32位的操作系 统，Oracle的SGA有1.75G的限制。有的32位操作系统上可以突破这个限制，方法还请看Biti的大作吧。

**2 分析表和索引，更改优化模式**

Oracle默认优化模式是CHOOSE，在这种情况下，如果表没有经过分析，经常导致查询使用全表扫描，而不使用索引。这通常导致磁盘I/O太多，而导致查询很慢。如果没有使用执行计划稳定性，则应该把表和索引都分析一下，这样可能直接会使查询速度大幅提升。分析表命令可以用ANALYZE TABLE 分析索引可以用ANALYZE INDEX命令。对于少于100万的表，可以考虑分析整个表，对于很大的表，可以按百分比来分析，但是百分比不能过低，否则生成的统计信息可能不准确。可以通过DBA\_TABLES的LAST\_ANALYZED列来查看表是否经过分析或分析时间，索引可以通过DBA\_INDEXES的LAST\_ANALYZED列。下面通过例子来说明分析前后的速度对比。（表CASE\_GA\_AJZLZ大约有35万数据，有主键）首先在SQLPLUS中打开自动查询执行计划功能。(第一次要执行\RDBMS\ADMIN\utlxplan.sql来创建PLAN\_TABLE这个表)

　SQL> SET AUTOTRACE ON

　SQL>SET TIMING ON

通过SET AUTOTRACE ON 来查看语句的执行计划，通过SET TIMING ON 来查看语句运行时间。

　SQL> select count(\*) from CASE\_GA\_AJZLZ;

　　COUNT(\*)

　　----------

　　346639

　　已用时间: 00: 00: 21.38

　　Execution Plan

　　　　0 SELECT STATEMENT Optimizer=CHOOSE

　　1 0 SORT (AGGREGATE)

　　2 1 TABLE ACCESS (FULL) OF 'CASE\_GA\_AJZLZ'

　　……………………

请注意上面分析中的TABLE ACCESS(FULL)，这说明该语句执行了全表扫描。而且查询使用了21.38秒。这时表还没有经过分析。下面我们来对该表进行分析：

SQL> analyze table CASE\_GA\_AJZLZ compute statistics;

表已分析。已用时间: 00: 05: 357.63。然后再来查询：

SQL> select count(\*) from CASE\_GA\_AJZLZ;

　　COUNT(\*)

　　----------

　　346639

　　已用时间: 00: 00: 00.71

　　Execution Plan

　　0 SELECT STATEMENT Optimizer=FIRST\_ROWS (Cost=351 Card=1)

　　1 0 SORT (AGGREGATE)

　　2 1 INDEX (FAST FULL SCAN) OF 'PK\_AJZLZ' (UNIQUE) (Cost=351

　　Card=346351)

　　…………………………

请注意，这次时间仅仅用了0.71秒！这要归功于INDEX(FAST FULL SCAN)。通过分析表，查询使用了PK\_AJZLZ索引，磁盘I/O大幅减少，速度也大幅提升！下面的实用语句可以

用来生成分析某个用户的所有表和索引，假设用户是GAXZUSR：

SQL> set pagesize 0

　　SQL> spool d:\analyze\_tables.sql;

　　SQL> select 'analyze table '||owner||'.'||table\_name||'

compute statistics;' from dba\_tables where owner='GAXZUSR';

　　SQL> spool off

　　SQL> spool spool d:\analyze\_indexes.sql;

　　SQL> select 'analyze index '||owner||'.'||index\_name||'

compute statistics;' from dba\_indexes where owner='GAXZUSR';

　　SQL> spool off

　　SQL> @d:\analyze\_tables.sql

　　SQL> @d:\analyze\_indexes.sql

**解释：**上面的语句生成了两个sql文件，分别分析全部的GAXZUSR的表和索引。如果需要按照百分比来分析表，可以修改一下脚本。通过上面的步骤，我们就完成了对表和索引的分析，可以测试一下速度的改进啦。建议定期运行上面的语句，尤其是数据经过大量更新。

当然，也可以通过dbms\_stats来分析表和索引，更方便一些。但是我仍然习惯上面的方法，因为成功与否会直接提示出来。

**另外，**我们可以将优化模式进行修改。optimizer\_mode值可以是RULE、CHOOSE、FIRST\_ROWS和ALL\_ROWS。对于OLTP系统，可以改成FIRST\_ROWS，来要求查询尽快返回结果。这样即使不用分析，在一般情况下也可以提高查询性能。但是表和索引经过分析后有助于找到最合适的执行计划。

**3 设置cursor\_sharing=FORCE 或SIMILAR**

这种方法是8i才开始有的，oracle805不支持。通过设置该参数，可以强制共享只有文字不同的语句解释计划。例如下面两条语句可以共享：

SQL> SELECT \* FROM MYTABLE WHERE NAME='tom'

SQL> SELECT \* FROM MYTABLE WHERE NAME='turner'

这个方法可以大幅降低缓冲区利用率低的问题，避免语句重新解释。通过这个功能，可以很大程度上解决硬解析带来的性能下降的问题。个人感觉可根据系统的实际情况，决定是否将该参数改成FORCE。该参数默认是exact。不过一定要注意，修改之前，必须先给ORACLE打补丁，否则改之后oracle会占用100%的CPU,无法使用。对于ORACLE9i，可以设置成SIMILAR，这个设置综合了FORCE和EXACT的优点。不过请慎用这个功能，这个参数也可能带来很大的负面影响！

**4 将常用的小表、索引钉在数据缓存KEEP池中**

内存上数据读取速度远远比硬盘中读取要快，据称，内存中数据读的速度是硬盘的14000倍！如果资源比较丰富，把常用的小的、而且经常进行全表扫描的表给钉内存中，当然是在好不过了。可以简单的通过ALTER TABLE tablename CACHE来实现，在ORACLE8i之后可以使用ALTER TABLE table STORAGE(BUFFER\_POOL KEEP)。一般来说，可以考虑把200数据块之内的表放在keep池中，当然要根据内存大小等因素来定。关于如何查出那些表或索引符合条件，可以使用本文提供的access.sql和access\_report.sql。这两个脚本是著名的Oracle专家 Burleson写的，你也可以在读懂了情况下根据实际情况调整一下脚本。对于索引，可以通过ALTER INDEX indexname STORAGE(BUFFER\_POOL KEEP)来钉在KEEP池中。

将表定在KEEP池中需要做一些准备工作。对于ORACLE9i 需要设置DB\_KEEP\_CACHE\_SIZE，对于8i，需要设置buffer\_pool\_keep。在8i中，还要修改db\_block\_lru\_latches，该参数默认是1，无法使用buffer\_pool\_keep。该参数应该比2\*3\*CPU数量少，但是要大于1，才能设置DB\_KEEP\_CACHE\_BUFFER。buffer\_pool\_keep从db\_block\_buffers中分配，因此也要小于db\_block\_buffers。设置好这些参数后，就可以把常用对象永久钉在内存里。

**5 设置optimizer\_max\_permutations**

对于多表连接查询，如果采用基于成本优化(CBO)，ORACLE会计算出很多种运行方案，

从中选择出最优方案。这个参数就是设置oracle究竟从多少种方案来选择最优。如果设置太大，那么计算最优方案过程也是时间比较长的。Oracle805和8i默认是80000，8建议改成2000。对于9i，已经默认是2000了。

**6 调整排序参数**

1. **SORT\_AREA\_SIZE:**默认的用来排序的SORT\_AREA\_SIZE大小是32K，通常显得有点小，一般可以考虑设置成1M（1048576）。这个参数不能设置过大，因为每个连接都要分配同样的排序内存。
2. **SORT\_MULTIBLOCK\_READ\_COUNT:增**大这个参数可以提高临时表空间排序性能，该参数默认是2，可以改成32来对比一下排序查询时间变化。注意，这个参数的最大值与平台有关系。

## 二十二、归档与不归档工作模式

**1 归档模式和非归档模式**

**数据库的任何改变，都会写到日志里面记录着，这个就叫做在线重做日志。  
可是当这个日志写满了怎么办？那就将这个文件归档，保存着，这就是[归档日志](https://www.baidu.com/s?wd=%E5%BD%92%E6%A1%A3%E6%97%A5%E5%BF%97&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)，然后在线重做日志就可以被复写。 这个有什么用处呢？比如，你保存了一个月的[归档日志](https://www.baidu.com/s?wd=%E5%BD%92%E6%A1%A3%E6%97%A5%E5%BF%97&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)，那就意味着你可以将数据库还原到这个月内的任意一个时间点。**

**重做日志redo log file是LGWR进程从Oracle实例中的redo log buffer写入的，是循环利用的。就是说一个redo log file(group) 写满后，才写下一个。**

**归档日志archive log是当数据库运行在归档模式下时，一个redo log file(group)写满后，由ARCn进程将重做日志的内容备份到归档日志文件下，然后这个redo log file(group)才能被下一次使用。**

在DBA部署数据库之初，必须要做出的最重要决定之一就是选择归档模式（ARCHIVELOG）或者非 归档模式（NOARCHIVELOG ）下运行数据库。我们知道，Oracle 数据库需要至少两组联机日志，每当一组 联机日志写满后会发生日志切换，继续向下一组联机日志写入。如果是归档模式，日志切换会触发归档进程 （ARCn）进行归档，生成归档日志。Oracle 保证归档完成前，联机日志不会被覆盖，如果是非归档模式， 则不会触发归档动作。

**2 归档模式的优缺点**

归档日志文件中保留了数据库的改动信息。

1. **在这种模式下可以获得如下好处:**

可以进行完全、不完全恢复：由于对数据库所做的全部改动都记录在日志文件中，如果发生硬盘故 障等导致数据文件丢失的话，则可以利用物理备份和归档日志完全恢复数据库，不会丢失任何数据。可以进行联机热备：所谓联机热备，就是在数据库运行状态下，对数据库进行备份。备份时用户对 数据库的使用不受任何影响。可以实施 Data Guard：可以部署 1 个或多个备用数据库，从而最大限度地提供灾难保护手段。可以实施 Stream：利用 Stream 技术，可以实现最简单的单向复制到复杂的双向复制、多向复制， 提供更加灵活的数据冗余方案。表空间可以脱机：可以备份部分数据库，比如重要的表空间。能够增量备份：只需做一次完全备份，以后只备份发生改变的数据，可以提高备份速度。更多的优化选项：随着 Oracle 版本升级，在联机热备方面不断有新的优化策略出现。

**（2）使用归档模式的缺点在于：**

需要更多的磁盘空间保存归档日志；

DBA 会有更多的管理工作，包括维护归档空间、备份归档日志。

**3 非归档模式的优缺点**

**（1）非归档模式不生成归档日志**，从数据安全角度来说，这种模式缺点是主要的，而优点可以忽略不计。

**（2）非归档模式的缺点包括：**

只能进行脱机备份，也就是所谓的“ 冷备份”，和联机备份的“ 热备份” 相对应，数据库必须完全 关闭后备份，在备份过程中数据库不可用；必须备份整个数据库，不能只备份部分数据库；不能增量备份，对于 TB 级数据库（VLDB） ，这是一个非常大的缺点；只能部分恢复，如果数据文件丢失需要恢复，DBA 只能恢复最后一次的完全备份，而之后的所有 数据库改变全部丢失。

**（3）非归档模式的优点包括：**

DBA 的管理工作减少，因为非归档模式不产生归档日志，因此 DBA 不用考虑对归档的管理；

**4 性能会有提升。**

非归档模式转换成归档模式，数据库创建过程中需要指定归档和非归档模式，如果选择的是非归档模式，可以在数据库创建完成后 手工改变成归档模式，具体操作步骤如下。

（1 ）关闭数据库：

shutdown immediate;

（2 ）启动数据库到 mount 状态：

startup mount;

（3 ）修改数据库归档模式：

alter database archivelog;

（4 ）启动数据库：

alter database open;

（5 ）定义归档位置，也就是归档日志保存路径：

alter syste set log\_archive\_dest\_1="location=d:\oradata\example\archive" scope=both;

（6 ）确认配置生效：

archive log list;

## 二十三、Mysql比较Oracel

**1、锁差异**

• Oracle锁加在数据块上

• InnoDB 是在索引上加锁，所以MySQL锁的粒度没有Oracle 精细。

**2、导入导出**

• Oracle采用EXP /IMP ，EXPDP/IMPDP导入导出。

• MySQL采用mysqldump导出，导入可以采用管道或source。

**3、commit**

• Oracle默认手动提交

• MySQL默认自动提交

1. **SQL 缓存**

MySQL只能缓存结果集，不能缓存SQL解析结果

**5、数据库对象**

• Oracle将数据库对象编译存储，直接执行二进制码

• MySQL只存储代码，临时解析执行，所以MySQL触发器、存储过程、函数等对象创建时仅检查语法，不检查逻辑

**6、事务**

• MySQL不是所有引擎都支持，建议优先使用InnoDB，相比其他引擎有更好的并发性

**7、空字符串处理**

• 而Oracle 只有NULL的概念

• MySQL有空字符串 '' 和NULL，空串不等于NULL

**8、结果集行数限制**

• Oracle使用rownum

• MySQL使用limit;

**9、执行SQL**

• Oracle需要select func() from dual;

• MySQL可以直接select func();

**10、修改字段类型或长度**

• Oracle：alter table tablename modify column ...;

• MySQL：alter table tablename modify (...);

**11、复制数据**

• Oracle的as是必选，create table table1 as select \* from table2;

• MySQL的as是可选，create table table1 select \* from table2;

**12、字符串处理**

• Oracle里用单引号包起字符串

• MySQL用双引号包起字符串