一，启动  
1.#su - oracle 切换到oracle用户且切换到它的环境  
  
2.$lsnrctl status 查看监听及数据库状态  
  
3.$lsnrctl start 启动监听  
  
4.$sqlplus / as sysdba 以DBA身份进入sqlplus  
  
5.SQL>startup 启动db  
  
二，停止  
1.#su - oracle 切换到oracle用户且切换到它的环境  
  
2.$lsnrctl stop 停止监听  
  
3.$sqlplus / as sysdba 以DBA身份进入sqlplus  
  
4.SQL>SHUTDOWN IMMEDIATE 关闭db  
  
其中startup和shutdowm还有其他一些可选参数，有兴趣可以另行查阅  
  
三，查看初始化参数及修改  
1.#su - oracle 切换到oracle用户且切换到它的环境

2.$sqlplus / as sysdba 以DBA身份进入sqlplus

3.SQL>show parameter session; 查看所接受的session数量  
######################################################  
Oracle process与session2008年12月29日 星期一 15:47  
  
Connected to Oracle Database 10g Release 10.1.0.5.0   
Connected as ifsapp  
  
SQL> show parameter session  
  
NAME TYPE VALUE  
------------------------------------ ----------- ------------------------------  
java\_max\_sessionspace\_size integer 0  
java\_soft\_sessionspace\_limit integer 0  
license\_max\_sessions integer 0  
license\_sessions\_warning integer 0  
logmnr\_max\_persistent\_sessions integer 1  
session\_cached\_cursors integer 0  
session\_max\_open\_files integer 10  
sessions integer 225  
shared\_server\_sessions integer   
  
  
SQL> show parameters processes;  
  
  
NAME TYPE VALUE  
------------------------------------ ----------- ------------------------------  
aq\_tm\_processes integer 0  
db\_writer\_processes integer 1  
gcs\_server\_processes integer 0  
job\_queue\_processes integer 3  
log\_archive\_max\_processes integer 2  
processes integer 200  
  
  
SQL> alter system set processes=600 scope=both;  
SQL>   
  
  
SQL> alter system set processes=600 scope=both;  
  
  
alter system set processes=600 scope=both  
  
  
ORA-02095: specified initialization parameter cannot be modified  
  
  
SQL> alter system set sessions=600 scope=both;  
  
  
alter system set sessions=600 scope=both  
  
  
ORA-02095: specified initialization parameter cannot be modified  
  
  
SQL>   
  
  
  
  
SQL> alter system set processes=600 scope=spfile;  
  
  
System altered  
  
  
SQL> show parameters processes;  
  
  
NAME TYPE VALUE  
------------------------------------ ----------- ------------------------------  
aq\_tm\_processes integer 0  
db\_writer\_processes integer 1  
gcs\_server\_processes integer 0  
job\_queue\_processes integer 3  
log\_archive\_max\_processes integer 2  
processes integer 200  
  
  
SQL> alter system set processes=600 scope=memory;  
  
  
alter system set processes=600 scope=memory  
  
  
ORA-02095: specified initialization parameter cannot be modified  
重启数据库之后生效：  
  
  
SQL> show parameters processes;  
  
  
NAME TYPE VALUE  
------------------------------------ ----------- ------------------------------  
aq\_tm\_processes integer 0  
db\_writer\_processes integer 1  
gcs\_server\_processes integer 0  
job\_queue\_processes integer 3  
log\_archive\_max\_processes integer 2  
processes integer 600  
  
  
SQL> show parameter session  
  
  
NAME TYPE VALUE  
------------------------------------ ----------- ------------------------------  
java\_max\_sessionspace\_size integer 0  
java\_soft\_sessionspace\_limit integer 0  
license\_max\_sessions integer 0  
license\_sessions\_warning integer 0  
logmnr\_max\_persistent\_sessions integer 1  
session\_cached\_cursors integer 0  
session\_max\_open\_files integer 10  
sessions integer 665  
shared\_server\_sessions integer   
  
  
  
  
修改Oracle process 和 session 的方法  
  
  
先备份spfile  
1.通过SQLPlus修改  
Oracle的sessions和processes的关系是  
sessions=1.1\*processes + 5  
使用sys，以sysdba权限登录：  
SQL> show parameter processes;  
NAME TYPE VALUE  
------------------------------------ ----------- ---------------------------------------  
aq\_tm\_processes integer 1  
db\_writer\_processes integer 1  
job\_queue\_processes integer 10  
log\_archive\_max\_processes integer 1  
processes integer 150  
SQL> alter system set processes=400 scope = spfile;  
系统已更改。  
SQL> show parameter processes;  
NAME TYPE VALUE  
------------------------------------ ----------- -----------------------------------------  
aq\_tm\_processes integer 1  
db\_writer\_processes integer 1  
job\_queue\_processes integer 10  
log\_archive\_max\_processes integer 1  
processes integer 150  
SQL> create pfile from spfile;  
文件已创建。  
重启数据库,OK！  
【注:sessions是个派生值,由processes的值决定,公式sessions=1.1\*process + 5】  
2.通过修改oracle配置文件进行修改  
修改SPFILEORCL.ORA文件中的processes的值。8.1.5中是init.ora文件，在9i中修改init.ora文件是无效的，这个文件由于是一个二进制的文件，不能直接使用notepad此类的编辑器打开。  
否则会报错误ORA-27101 Shared memory realm does not exist。使用UltraEdit或者EditPlus之类的可以编辑二进制文件的编辑器打开此文件（直接编辑二进制文件），然后在Windows服务中重新启动Oracle服务器即可。  
3.通过Oracle Enterprise Manager Console在图形化管理器中修改  
a.以系统管理员的身份登入  
b.进入数据库的例程-配置-一般信息-所有初始化参数  
c.修改processes的值  
d.Sessions的值由公式sessions=1.1\*process + 5派生得到  
  
  
  
  
一个process可以有0个，1个或多个session  
  
  
=============================================  
Oracle中系统process与session的关系  
  
  
在实际的应用中，Shared Server中的Process与Oracle数据库中的Session并不是一一对应的，Shared Server中的Process(一个)对应着Oracle中的一个或者一个以上的Session。   
  
  
　　在dedicated server机器上的测试如下：   
  
  
　　数据库的session和操作系统process是对应的。   
  
  
　　这表示一个session对应一个process，但是一个process未必对应一个session。   
  
  
　　大家可以通过SELECT spid   
  
  
　　FROM v$process   
  
  
　　WHERE NOT EXISTS ( SELECT 1   
  
  
　　FROM v$session   
  
  
　　WHERE paddr = addr); 查看   
  
  
　　或   
  
  
　　SQL> select count(\*) from v$process;   
  
  
　　COUNT(\*)   
  
  
　　----------   
  
  
　　53   
  
  
　　SQL> select count(\*) from v$session;   
  
  
　　COUNT(\*)   
  
  
　　----------   
  
  
　　50  
  
  
  
  
oracle10g中只要指定processes，sessions就会自动计算出，所以不需要指定sessions也可，除非对sessions有特殊要求，要求其比oracle自动计算出的sessions要大，那么就需要手动指定sessions了  
   
#########################################################  
  
  
4.SQL>alter system set shared\_servers=10; 将shared\_servers的数量设置为10  
  
  
   
  
  
四．数据库连接数目  
  
  
其中一个数据库连接需要一个session,它的值由processes决定，session与processes通常有以下关系：  
  
  
session = 1.1 \* processes + 5  
  
  
不过这也不是绝对的，还要受到CPU和内存等硬件条件的限制。另外processes和session不可以通过alter system语句直接修改，只可以修改服务器参数文件来更改(Server Parameter File)。如果存在一个server parameter file，通过alter system语句所作的更改将会被持久化到文件中。  
  
  
首先以Oracle用户登录  
  
  
1、 $/oracle/oracle/app/product/101/bin/dbstart：启动数据库。  
  
  
2、 $/oracle/oracle/app/product/101/bin/dbshut：关闭数据库。  
  
  
3、 $/oracle/oracle/app/product/101/bin/dbca：数据库配置助理，进行数据库安装、删除和修改。  
  
  
4、 $/oracle/oracle/app/product/101/bin/netca：数据库网络服务配置管理。  
  
  
5、 $/oracle/oracle/app/product/101/bin/netmgr：数据库网络服务和监听配置助理。  
  
  
6、 $/oracle/oracle/app/product/101/bin/sqlplus：可运行SQL，进行数据库操作，并可起停数据库。运行sqlplus命令，输入sys用户和口令登陆以后，运行如下命名：  
  
  
〉shutdown immediate 立即关闭数据库  
  
  
〉startup 启动数据库  
  
  
7、 $/oracle/oracle/app/product/101/bin/lsnrctl start：启动数据库监听  
  
  
$/oracle/oracle/app/product/101/bin/lsnrctl stop：关闭数据库监听  
  
  
$/oracle/oracle/app/product/101/bin/lsnrctl status：察看数据库监听运行状态  
  
  
Oracle数据库命令速查  
  
  
Oracle数据库命令速查  
  
  
Oracle数据库命令速查  
  
  
1. 日志管理  
  
  
1.1. 强制日志切换（forcing log switches）  
  
  
sql> alter system switch logfile;  
  
  
1.2. 强制执行检查点（forcing checkpoints）  
  
  
sql> alter system checkpoint;  
  
  
1.3. 增加一个重做日志组（adding online redo log groups）  
  
  
sql> alter database add logfile [group 4]sql> ('/disk3/log4a.rdo','/disk4/log4b.rdo') size 1m;  
  
  
1.4. 增加一个重做日志文件（adding online redo log members）  
  
  
sql> alter database add logfile member  
  
  
sql> '/disk3/log1b.rdo' to group 1,  
  
  
sql> '/disk4/log2b.rdo' to group 2;  
  
  
1.5. 改变重做日志文件名（changes the name of the online redo logfile）  
  
  
sql> alter database rename file 'c:/oracle/oradata/oradb/redo01.log'  
  
  
sql> to 'c:/oracle/oradata/redo01.log';  
  
  
1.6. 删除重做日志组（drop online redo log groups）  
  
  
sql> alter database drop logfile group 3;  
  
  
1.7. 删除重做日志文件（drop online redo log members）  
  
  
sql> alter database drop logfile member 'c:/oracle/oradata/redo01.log';  
  
  
1.8. 清空重做日志文件（clearing online redo log files）  
  
  
sql> alter database clear [unarchived] logfile 'c:/oracle/log2a.rdo';  
  
  
1.9. 使用logminer分析重做日志文件（using logminer analyzing redo logfiles）  
  
  
a. in the init.ora specify utl\_file\_dir = ' '  
  
  
b. sql> execute dbms\_logmnr\_d.build('oradb.ora','c:\oracle\oradb\log');  
  
  
c. sql> execute dbms\_logmnr\_add\_logfile('c:\oracle\oradata\oradb\redo01.log',  
  
  
sql> dbms\_logmnr.new);  
  
  
d. sql> execute dbms\_logmnr.add\_logfile('c:\oracle\oradata\oradb\redo02.log',  
  
  
sql> dbms\_logmnr.addfile);  
  
  
e. sql> execute dbms\_logmnr.start\_logmnr(dictfilename=>'c:\oracle\oradb\log\oradb.ora');  
  
  
f. sql> select \* from v$logmnr\_contents(v$logmnr\_dictionary,v$logmnr\_parameters  
  
  
sql> v$logmnr\_logs);  
  
  
g. sql> execute dbms\_logmnr.end\_logmnr;  
  
  
2. 表空间管理  
  
  
2.1. 创造表空间（create tablespaces）  
  
  
sql> create tablespace ts\_name datafile 'c:\oracle\oradata\file1.dbf' size 100m ,  
  
  
sql>'c:\oracle\oradata\file2.dbf' size 100m minimum extent 550k [logging/nologging]sql> default storage (initial 500k next 500k maxextents 500 pctinccease 0)  
  
  
sql> [online/offline] [permanent/temporary] [extent\_management\_clause]  
  
  
2.2. 创建本地管理的表空间（locally managed tablespace）  
  
  
sql> create tablespace user\_data datafile 'c:\oracle\oradata\user\_data01.dbf'  
  
  
sql> size 500m extent management local uniform size 10m;  
  
  
2.3. 创建临时表空间（temporary tablespace）  
  
  
sql> create temporary tablespace temp tempfile 'c:\oracle\oradata\temp01.dbf'  
  
  
sql> size 500m extent management local uniform size 10m;  
  
  
2.4. 改变表空间的存储参数（change the storage setting）  
  
  
sql> alter tablespace app\_data minimum extent 2m;  
  
  
sql> alter tablespace app\_data default storage(initial 2m next 2m maxextents 999);  
  
  
2.5. 使表空间离线或连线（taking tablespace offline or online）  
  
  
sql> alter tablespace app\_data offline;  
  
  
sql> alter tablespace app\_data online;  
  
  
2.6. 设置表空间为只读/可写模式（read\_only tablespace）  
  
  
sql> alter tablespace app\_data read only|write;  
  
  
2.7. 删除表空间（droping tablespace）  
  
  
sql> drop tablespace app\_data including contents;  
  
  
2.8. 允许数据文件自动扩展（enableing automatic extension of data files）  
  
  
sql> alter tablespace app\_data add datafile 'c:\oracle\oradata\app\_data01.dbf' size 200m  
  
  
sql> autoextend on next 10m maxsize 500m;  
  
  
2.9. 手工改变数据文件的大小（change the size fo data files manually）  
  
  
sql> alter database datafile 'c:\oracle\oradata\app\_data.dbf' resize 200m;  
  
  
2.10. 改名表空间中的数据文件（Moving data files: alter tablespace）  
  
  
sql> alter tablespace app\_data rename datafile 'c:\oracle\oradata\app\_data.dbf  
  
  
sql> to 'c:\oracle\app\_data.dbf';  
  
  
2.11. 改变数据库中的数据文件（moving data files:alter database）  
  
  
sql> alter database rename file 'c:\oracle\oradata\app\_data.dbf' to 'c:\oracle\app\_data.dbf';  
  
  
3. 表  
  
  
3.1. 创建表（create a table）  
  
  
sql> create table table\_name (column datatype,column datatype]....)  
  
  
sql> tablespace tablespace\_name [pctfree integer] [pctused integer]  
  
  
sql> [initrans integer] [maxtrans integer]sql> storage(initial 200k next 200k pctincrease 0 maxextents 50)  
  
  
sql> [logging|nologging] [cache|nocache]  
  
  
3.2. 复制一个已存在的表（copy an existing table）  
  
  
sql> create table table\_name [logging|nologging] as subquery  
  
  
3.3. 创建一个临时表（create temporary table）  
  
  
sql> create global temporary table xay\_temp as select \* from xay;  
  
  
on commit preserve rows/on commit delete rows  
  
  
3.4. pctfree和pctused参数计算公式  
  
  
pctfree = (average row size - initial row size) \*100 /average row size  
  
  
pctused = 100-pctfree- (average row size\*100/available data space)  
  
  
这是标准的计算公式。  
  
  
PCTFREE：指定表内每个数据块中空间的百分比。 PCTFREE 的值必须介于 0 和 99 之间。如果值为零，表示可以通过插入新行来填充整个块。缺省值为 10。此值表示每个块中保留着 10% 的空间，用于更新现有的行以及插入新行，每个块最多可填充到 90%。   
  
  
PCTUSED：指定为表内每个数据块维护的已用空间的最小百分比。如果一个块的已用空间低于 PCTUSED，则可在该块中插入行。PCTUSED 的值为介于 0 和 99 之间的整数，缺省值为 40。   
  
  
结合 PCTFREE 和 PCTUSED 就可以确定将新行插入到现有数据块中，还是插入到新块中。这两个参数值的和必须小于或等于 100。使用这两个参数可以更有效地利用表内的空间。   
  
  
设置 PCTFREE 和 PCTUSED   
  
  
设置 PCTFREE   
  
  
PCTFREE 值越高，可为数据库块内的更新提供的空间就越大。如果表存在下面两种情况，则应设置一个更高的值：   
  
  
? 某些列最初为 NULL，后来更新为某个值   
  
  
? 某些列由于更新，大小可能增加   
  
  
PCTFREE 的值越高，块密度就越低，即每个块容纳的行数就越少。   
  
  
上面的公式确保块中有足够的空闲空间供行增长使用。   
  
  
设置 PCTUSED  
  
  
设置 PCTUSED 以确保只有在具备足够空间来容纳一个平均大小的行时才将块返回到空闲列表中。如果空闲列表中的某个块没有足够的空间来插入一行，Oracle 服务器将查找空闲列表中的下一个块。直到找到具备足够空间的块或者到达列表的末尾，这种线性扫描才会结束。使用给定的公式可以增加找到具有所需空闲空间的块的概率，从而缩短扫描空闲列表的时间。   
  
  
注：可以使用 ANALYZE TABLE 命令估算平均行大小的值。  
  
  
注：Oracle9i “自动段空间管理” 功能可替代 PCTUSED、FREELISTS 和 FREELIST GROUPS。  
  
  
3.5. 改变存储和块利用率参数（change storage and block utilization parameter）  
  
  
sql> alter table table\_name pctfree=30 pctused=50 storage(next 500k minextents 2 maxextents 100);  
  
  
3.6. 手工分配区间（extents）（manually allocating extents）  
  
  
sql> alter table table\_name allocate extent(size 500k datafile 'c:/oracle/data.dbf');  
  
  
3.7. 改变表的所属表空间（move tablespace）  
  
  
sql> alter table employee move tablespace users;  
  
  
3.8. 释放表中未用空间（deallocate of unused space）  
  
  
sql> alter table table\_name deallocate unused [keep integer]  
  
  
3.9. 截断表（truncate）（truncate a table）  
  
  
sql> truncate table table\_name;  
  
  
截断一个表将删除表中所有行，从而释放已使用的空间。   
  
  
对应的索引将被截断。  
  
  
(注：truncate table不是DML语句，是DDL语句  
  
  
另外，truncate 与delete的区别是，delelte不释放空间，truncate释放空间。)  
  
  
3.10. 删除表（drop a table）  
  
  
sql> drop table table\_name [cascade constraints];  
  
  
3.11. 删除列（drop a column）  
  
  
sql> alter table table\_name drop column comments cascade constraints checkpoint 1000;  
  
  
alter table table\_name drop columns continue;  
  
  
3.12. 标识某一列为未使用（unused）（mark a column as unused）  
  
  
sql> alter table table\_name set unused column comments cascade constraints;  
  
  
alter table table\_name drop unused columns checkpoint 1000;  
  
  
alter table orders drop columns continue checkpoint 1000  
  
  
data\_dictionary : dba\_unused\_col\_tabs  
  
  
除将列从表中删除以外，还可以先将列标记为 “未使用”，以后再删除。因为没有删除数据，所以此操作不回收磁盘空间，因而具有速度比较快的优点。被标为 “未使用” 的列可在以后系统活动较少时从表中删除。   
  
  
未使用的列就像不属于表一样。查询时看不到未使用列中的数据。此外，在执行 DESCRIBE 命令时，也不会显示这些列的名称和数据类型。用户可以添加与未使用的列同名的新列。   
  
  
如果想删除同一表中的两列，则可先将列设置为 “未使用” 然后再删除。在删除两列时，表中的所有行都会更新两次；但如果将这些列设置为 “未使用” 然后再删除，则所有的行仅更新一次。  
  
  
4. 索引  
  
  
4.1. 创建一个基于函数的索引（creating function-based indexes）  
  
  
sql> create index summit.item\_quantity on summit.item(quantity-quantity\_shipped);  
  
  
基于函数的索引(Function-based indexes)：   
  
  
如果在表中要建立索引的一列或多列上使用了函数或表达式，则创建的是基于函数的索引。基于函数的索引预先计算函数或表达式的值，并将结果存储在索引中。可以将基于函数的索引创建为 B 树或位图索引。  
  
  
4.2. 创建一个B树索引（create a B-tree index）  
  
  
sql> create [unique] index index\_name on table\_name(column,.. asc/desc) tablespace  
  
  
sql> tablespace\_name [pctfree integer] [initrans integer] [maxtrans integer]  
  
  
sql> [logging | nologging] [nosort] storage(initial 200k next 200k pctincrease 0 maxextents 50);  
  
  
B树：平衡二叉树，oracle中用的最多的索引模式，适用于取值唯一性高的情况。只有两层，非叶级、叶级（指针所在级）  
  
  
4.3. 索引中pctfree参数计算公式  
  
  
pctfree(index)=(maximum number of rows-initial number of rows)\*100/maximum number of rows  
  
  
4.4. 创建一个反向键索引（creating reverse key indexes）  
  
  
sql> create unique index xay\_id on xay(a) reverse pctfree 30 storage(initial 200k next 200k pctincrease 0 maxextents 50) tablespace indx;  
  
  
4.5. 创建位图索引（create bitmap index）  
  
  
sql> create bitmap index xay\_id on xay(a) pctfree 30 storage( initial 200k next 200k pctincrease 0 maxextents 50) tablespace indx;  
  
  
在下列情况中，位图索引比 B 树索引更有利：   
  
  
? 当表包含数百万行且键列的基数很低（即，该列中重复的值很多）时。例如，对于包含护照记录的表的性别列和婚姻状况列而言，位图索引比 B 树索引更适合   
  
  
? 当查询经常使用涉及 OR 运算符的多个 WHERE 条件组合时   
  
  
? 当键列上存在只读或很少的更新操作时   
  
  
（位图索引适用于取值的唯一性很低的情况）  
  
  
4.6. 改变索引的存储参数（change storage parameter of index）  
  
  
sql> alter index xay\_id storage (next 400k maxextents 100);  
  
  
4.7. 为索引分配空间（allocating index space）  
  
  
sql> alter index xay\_id allocate extent(size 200k datafile 'c:/oracle/index.dbf');8.alter index xay\_id deallocate unused;  
  
  
手动分配索引空间：   
  
  
在表上进行频繁的插入操作前，可能需要向索引添加区。添加区可防止索引动态扩展并导致性能降低。   
  
  
（其中指定的数据文件一定是索引所在表空间的数据文件）  
  
  
5. 约束（constraints）  
  
  
5.1. 将约束定义为立即（immediate）或延迟(deferred)（define constraints as immediate or deferred）  
  
  
sql> alter session set constraint[s] = immediate/deferred/default;  
  
  
set constraint[s] constraint\_name/all immediate/deferred;  
  
  
SET CONSTRAINTS 语句用于将特定事务处理的约束设置为 DEFERRED 或 IMMEDIATE。可以使用此语句设置约束名称列表或约束的模式。SET CONSTRAINTS 模式将一直持续到事务处理完成或者另一个 SET CONSTRAINTS 语句重置模式。SET CONSTRAINTS 语句不允许在触发器内部使用。   
  
  
ALTER SESSION 语句还包含将约束设置为 IMMEDIATE 或 DEFERRED 的子句 SET CONSTRAINTS。此命令缺省为设置所有 (ALL) 可延迟的约束（不能指定约束名称列表）。ALTER SESSION SET CONSTRAINTS 语句仅适用于当前的会话。  
  
  
5.2. 删除表或表空间时连带删除其上的外键（约束）  
  
  
sql> drop table table\_name cascade constraints  
  
  
sql> drop tablespace tablespace\_name including contents cascade constraints  
  
  
在删除父表之前，必须先删除外键。可以使用以下一条语句同时执行这两个操作：   
  
  
 DROP TABLE table CASCADE CONSTRAINTS  
  
  
在未删除或禁用外键之前无法截断(truncated)父表。  
  
  
在删除包含父表的表空间之前，必须先删除外键。可使用下列命令完成该操作：   
  
  
 DROP TABLESPACE tablespace INCLUDING CONTENTS   
  
  
 CASCADE CONSTRAINTS  
  
  
如果从父表中删除行时没有使用 DELETE CASCADE 选项，Oracle 服务器必须确保子表中的行不包含相应的外键。同样，仅当子行中不包含旧键值时，才允许更新父键。如果子表的外键上没有索引，则 Oracle 服务器锁定子表并禁止更改以确保引用完整性。如果表上有索引，则可通过锁定索引项并避免子表上有更具限制性的锁来维护引用完整性。如果必须从不同的事务处理同时更新两个表，则在外键列上创建索引。  
  
  
5.3. 在创建表时定义约束（define constraints while create a table）  
  
  
sql> create table xay(id number(7) constraint xay\_id primary key deferrable  
  
  
sql> using index storage(initial 100k next 100k) tablespace indx);  
  
  
primary key/unique/references table(column)/check  
  
  
5.4. 启用当前禁用的约束（enable constraints）  
  
  
sql> alter table xay enable novalidate constraint xay\_id;  
  
  
启用 NOVALIDATE:  
  
  
对于当前已有索引的 PRIMARY KEY 和 UNIQUE 约束，启用 NOVALIDATE 约束比启用 VALIDATE 约束要快得多，这是因为，如果约束是可延迟的，则不检查现有数据是否违反约束。如果使用该选项启用约束，则不要求锁定表。这种方法适合表上有许多 DML 活动的情况，如在 OLTP 环境中。  
  
  
但是，如果需要创建索引，使用这种启用约束的方法并不能比 ENABLE VALIDATE 带来更多的好处，因为 Oracle 服务器在建立索引时锁定表。  
  
  
5.5. 启用约束（enable constraints）  
  
  
sql> alter table xay enable validate constraint xay\_id;  
  
  
6. 加载（load）数据  
  
  
6.1. 使用insert语句从另一张表中“直接加载”数据到新表（loading data using direct\_load insert）  
  
  
sql> insert /\*+append \*/ into emp nologging  
  
  
sql> select \* from emp\_old;  
  
  
6.2. 使用insert方式的“并行直接加载”数据（parallel direct-load insert）  
  
  
sql> alter session enable parallel dml;  
  
  
sql> insert /\*+parallel(emp,2) \*/ into emp nologging  
  
  
sql> select \* from emp\_old;  
  
  
6.3. 使用sql\*loader加载数据（using sql\*loader）  
  
  
sql> sqlldr scott/tiger \  
  
  
sql> control = ulcase6.ctl \  
  
  
sql> log = ulcase6.log direct=true  
  
  
7. 重整数据（reorganizing data）  
  
  
7.1. 使用export倒出数据（using export）  
  
  
$exp scott/tiger tables(dept,emp) file=c:\emp.dmp log=exp.log compress=n direct=y  
  
  
7.2. 使用import倒入数据（using import）  
  
  
$imp scott/tiger tables(dept,emp) file=emp.dmp log=imp.log ignore=y  
  
  
7.3. 以移动表空间（transporting）方式导入导出一个表（transporting a tablespace）  
  
  
sql>alter tablespace sales\_ts read only;  
  
  
$exp sys/.. file=xay.dmp transport\_tablespace=y tablespace=sales\_ts  
  
  
triggers=n constraints=n  
  
  
$copy datafile  
  
  
$imp sys/.. file=xay.dmp transport\_tablespace=y datafiles=(/disk1/sles01.dbf,/disk2  
  
  
/sles02.dbf)  
  
  
sql> alter tablespace sales\_ts read write;  
  
  
7.4. checking transport set  
  
  
sql> DBMS\_tts.transport\_set\_check(ts\_list =>'sales\_ts' ..,incl\_constraints=>true);  
  
  
在表transport\_set\_violations 中查看  
  
  
sql> dbms\_tts.isselfcontained 为true 是，表示自包含  
  
  
8. 管理口令及相关资源（managing password security and resources）  
  
  
8.1. 修改用户帐号加/解锁及口令（controlling account lock and password）  
  
  
sql> alter user juncky identified by oracle account unlock;  
  
  
8.2. user\_provided password function  
  
  
sql> function\_name(userid in varchar2(30),password in varchar2(30),  
  
  
old\_password in varchar2(30)) return boolean  
  
  
8.3. 创建概要文件：设置口令参数（create a profile : password setting）  
  
  
sql> create profile grace\_5 limit failed\_login\_attempts 3  
  
  
sql> password\_lock\_time unlimited password\_life\_time 30  
  
  
sql>password\_reuse\_time 30 password\_verify\_function verify\_function  
  
  
sql> password\_grace\_time 5;  
  
  
8.4. 修改概要文件（altering a profile）  
  
  
sql> alter profile default limit  
  
  
sql> failed\_login\_attempts 3  
  
  
sql> password\_life\_time 60;  
  
  
8.5. 删除概要文件（drop a profile）  
  
  
sql> drop profile grace\_5 [cascade];  
  
  
8.6. 创建概要文件：设置资源限额（create a profile : resource limit）  
  
  
sql> create profile developer\_prof limit sessions\_per\_user 2  
  
  
sql> cpu\_per\_session 10000 idle\_time 60 connect\_time 480;  
  
  
8.7. view => resource\_cost : alter resource cost  
  
  
dba\_Users,dba\_profiles  
  
  
8.8. 允许资源限额（enable resource limits）  
  
  
sql> alter system set resource\_limit=true;  
  
  
9. 管理用户（Managing users）  
  
  
9.1. 创建用户（数据库认证方式）（create a user: database authentication）  
  
  
sql> create user juncky identified by oracle default tablespace users temporary tablespace temp quota 10m/unlimited on data password expire [account lock|unlock] [profile profilename|default];  
  
  
9.2. 修改用户的表空间限额（change user quota on tablespace）  
  
  
sql> alter user juncky quota 0 on users;  
  
  
9.3. 删除用户（drop a user）  
  
  
sql> drop user juncky [cascade];  
  
  
9.4. 监控用户的视图（monitor user）  
  
  
view: dba\_users , dba\_ts\_quotas  
  
  
10. 管理权限（managing privileges）  
  
  
10.1. 系统权限视图（system privileges:）  
  
  
view => system\_privilege\_map ,dba\_sys\_privs,session\_privs  
  
  
10.2. 授权系统权限（grant system privilege）  
  
  
sql> grant create session,create table to managers;  
  
  
sql> grant create session to scott with admin option;  
  
  
with admin option can grant or revoke privilege from any user or role;  
  
  
10.3. sysdba和sysoper的权限（sysdba and sysoper privileges:）  
  
  
sysoper: startup,shutdown,alter database open|mount,alter database backup controlfile,alter tablespace begin/end backup,recover database,alter database archivelog,restricted session  
  
  
sysdba:sysoper privileges with admin option,create database,recover database until  
  
  
10.4. 口令文件成员视图（password file members:）  
  
  
view:=> v$pwfile\_users  
  
  
10.5. O7\_dictionary\_accessibility =true  
  
  
 restriction access to view or tables in other schema  
  
  
10.6. 撤销系统权限（revoke system privilege）  
  
  
sql> revoke create table from karen;  
  
  
sql> revoke create session from scott;  
  
  
10.7. 授权对象权限（grant object privilege）  
  
  
sql> grant execute on dbms\_pipe to public;  
  
  
sql> grant update(first\_name,salary) on employee to karen with grant option;  
  
  
10.8. 显示对象权限的视图（display object privilege ）:   
  
  
view => dba\_tab\_privs, dba\_col\_privs  
  
  
10.9. 撤销用户的对象权限（revoke object privilege）  
  
  
sql> revoke execute on dbms\_pipe from scott [cascade constraints];  
  
  
10.10. 审计记录视图（audit record view） :  
  
  
=> sys. aud$  
  
  
10.11. 保护审计线索（protecting the audit trail）  
  
  
sql> audit delete on sys.aud$ by access;  
  
  
保护审计线索：  
  
  
应保护审计线索，以防添加、修改或删除审计信息。发出以下命令：  
  
  
 SQL> AUDIT delete ON sys.aud$ BY ACCESS;  
  
  
可防止审计线索未经授权即被删除；只有 DBA 才拥有 DELETE\_CATALOG\_ROLE 角色。  
  
  
10.12. 语句审计（statement auditing）  
  
  
sql> audit user;  
  
  
语句审计：该种审计对 SQL 语句进行选择性审计，而并不审计语句针对的特定方案对象。例如，AUDIT TABLE 跟踪多个 DDL 语句，而与这些语句针对的表无关。可以设置语句审计，以便对数据库中的所选用户或每个用户进行审计。  
  
  
10.13. 权限审计（privilege auditing）  
  
  
sql> audit select any table by summit by access;  
  
  
权限审计：  
  
  
该种审计对执行操作应具有的相应系统权限进行选择性审计，如 AUDIT CREATE ANY TRIGGER。可以设置权限审计对数据库中的所选用户或每个用户进行审计。  
  
  
10.14. 方案对象审计（schema object auditing）  
  
  
sql> audit lock on summit.employee by access whenever successful;  
  
  
方案对象审计：  
  
  
该种审计对特定方案对象上的特定语句进行选择性审计，如 AUDIT SELECT ON HR.EMPLOYEES。方案对象审计始终适用于所有数据库用户。  
  
  
11. 审计选项视图（view audit option） :   
  
  
view=> all\_def\_audit\_opts,dba\_stmt\_audit\_opts,dba\_priv\_audit\_opts,dba\_obj\_audit\_opts  
  
  
12. 审计结果视图（view audit result）:   
  
  
view=> dba\_audit\_trail,dba\_audit\_exists,dba\_audit\_object,dba\_audit\_session,dba\_audit\_statement  
  
  
13. 管理角色（manager role）  
  
  
13.1. 创建角色（create roles）  
  
  
sql> create role sales\_clerk;  
  
  
sql> create role hr\_clerk identified by bonus;  
  
  
sql> create role hr\_manager identified externally;  
  
  
13.2. 修改角色（modify role）  
  
  
sql> alter role sales\_clerk identified by commission;  
  
  
sql> alter role hr\_clerk identified externally;  
  
  
sql>alter role hr\_manager not identified;  
  
  
13.3. 分配角色给用户（assigning roles）  
  
  
sql> grant sales\_clerk to scott;  
  
  
sql> grant hr\_clerk to hr\_manager;  
  
  
sql> grant hr\_manager to scott with admin option;  
  
  
13.4. 建立缺省角色（establish default role）  
  
  
sql> alter user scott default role hr\_clerk,sales\_clerk;  
  
  
sql> alter user scott default role all;  
  
  
sql> alter user scott default role all except hr\_clerk;  
  
  
sql> alter user scott default role none;  
  
  
13.5. 允许和禁止角色（enable and disable roles）  
  
  
sql> set role hr\_clerk;  
  
  
sql> set role sales\_clerk identified by commission;  
  
  
sql> set role all except sales\_clerk;  
  
  
sql> set role none;  
  
  
13.6. 撤销用户的角色（remove role from user）  
  
  
sql> revoke sales\_clerk from scott;  
  
  
sql> revoke hr\_manager from public;  
  
  
13.7. 删除角色（remove role）  
  
  
sql> drop role hr\_manager;  
  
  
13.8. 显示角色信息的视图（display role information）  
  
  
view: =>dba\_roles,dba\_role\_privs,role\_role\_privs,dba\_sys\_privs,role\_sys\_privs,role\_tab\_privs,session\_roles  
  
  
14. 备份和恢复（BACKUP and RECOVERY）  
  
  
14.1. 备份恢复用相关视图  
  
  
v$sga,v$instance,v$process,v$bgprocess,v$database,v$datafile,v$sgastat  
  
  
14.2. 为Rman设置初始化参数  
  
  
Rman need set dbwr\_IO\_slaves or backup\_tape\_IO\_slaves and large\_pool\_size  
  
  
14.3. 监控并行回滚的视图（Monitoring Parallel Rollback）  
  
  
> v$fast\_start\_servers , v$fast\_start\_transactions  
  
  
14.4. 执行一个冷备份（perform a closed database backup (noarchivelog)）  
  
  
> shutdown immediate  
  
  
> cp files /backup/  
  
  
> startup  
  
  
14.5. 改变数据文件的位置（restore to a different location）  
  
  
> connect system/manager as sysdba  
  
  
> startup mount  
  
  
> alter database rename file '/disk1/../user.dbf' to '/disk2/../user.dbf';  
  
  
> alter database open;  
  
  
14.6. 恢复命令(recover)语法 （recover syntax）  
  
  
--recover a mounted database  
  
  
>recover database;  
  
  
>recover datafile '/disk1/data/df2.dbf';  
  
  
>alter database recover database;  
  
  
--recover an opened database  
  
  
>recover tablespace user\_data;  
  
  
>recover datafile 2;  
  
  
>alter database recover datafile 2;  
  
  
14.7. 设置自动应用重做日志文件（how to apply redo log files automatically）  
  
  
>set autorecovery on  
  
  
>recover automatic datafile 4;  
  
  
14.8. 完全恢复（complete recovery:）  
  
  
--method 1(mounted databae)  
  
  
>copy c:\backup\user.dbf c:\oradata\user.dbf  
  
  
>startup mount  
  
  
>recover datafile 'c:\oradata\user.dbf;  
  
  
>alter database open;  
  
  
--method 2(opened database,initially opened,not system or rollback datafile)  
  
  
>copy c:\backup\user.dbf c:\oradata\user.dbf (alter tablespace offline)  
  
  
>recover datafile 'c:\oradata\user.dbf' or  
  
  
>recover tablespace user\_data;  
  
  
>alter database datafile 'c:\oradata\user.dbf' online or  
  
  
>alter tablespace user\_data online;  
  
  
--method 3(opened database,initially closed not system or rollback datafile)  
  
  
>startup mount  
  
  
>alter database datafile 'c:\oradata\user.dbf' offline;  
  
  
>alter database open  
  
  
>copy c:\backup\user.dbf d:\oradata\user.dbf  
  
  
>alter database rename file 'c:\oradata\user.dbf' to 'd:\oradata\user.dbf'  
  
  
>recover datafile 'e:\oradata\user.dbf' or recover tablespace user\_data;  
  
  
>alter tablespace user\_data online;  
  
  
--method 4(loss of data file with no backup and have all archive log)  
  
  
>alter tablespace user\_data offline immediate;  
  
  
>alter database create datafile 'd:\oradata\user.dbf' as 'c:\oradata\user.dbf''  
  
  
>recover tablespace user\_data;  
  
  
>alter tablespace user\_data online  
  
  
14.9. 对一个打开的数据库进行备份（perform an open database backup）  
  
  
> alter tablespace user\_data begin backup;  
  
  
> copy files /backup/  
  
  
> alter database datafile '/c:/../data.dbf' end backup;  
  
  
> alter system switch logfile;  
  
  
14.10. 备份一个控制文件（backup a control file）  
  
  
> alter database backup controlfile to 'control1.bkp';  
  
  
> alter database backup controlfile to trace;  
  
  
14.11. 非归档模式下的恢复（recovery (noarchivelog mode)）  
  
  
> shutdown abort  
  
  
> cp files  
  
  
> startup  
  
  
14.12. 备份模式下的文件恢复（recovery of file in backup mode）  
  
  
>alter database datafile 2 end backup;  
  
  
14.13. 清空重做日志文件（clearing redo log file）  
  
  
>alter database clear unarchived logfile group 1;  
  
  
>alter database clear unarchived logfile group 1 unrecoverable datafile;  
  
  
14.14. 重做日志的恢复（删除与重建）（redo log recovery）  
  
  
>alter database add logfile group 3 'c:\oradata\redo03.log' size 1000k;  
  
  
>alter database drop logfile group 1;  
  
  
>alter database open;  
  
  
or >cp c:\oradata\redo02.log' c:\oradata\redo01.log  
  
  
>alter database clear logfile 'c:\oradata\log01.log';  
  
  
Oracle数据库手边命令  
1、得到数据库名和创建日期  
SELECT name, created, log\_mode, open\_mode FROM v$database;  
2、ORACLE数据库的计算机的主机名，ORACLE数据库的实例名及ORACLE数据库管理系统的版本信息  
SELECT host\_name, instance\_name, version FROMv$instance;  
3、为了知道oracle数据库版本的一些特殊信息  
select \* from v$version;  
4、获取控制文件名字  
select \* from v$controlfile;  
5、得到Oracle数据库的重做日志配置信息  
SELECT group#, members, bytes, status, archived FROM v$log;  
select GROUP#,MEMBER from v$logfile;  
6、获取oracle的每个重做日志（成员）文件所存放的具体位置  
select \* from v$logfile;  
7、知道ORACLE数据库的备份和恢复策略和归档文件的具体位置  
archive log list  
8、知道ORACLE数据库中到底有多少表空间以及每个表空间的状态  
select tablespace\_name, block\_size, status, contents, logging from dba\_tablespaces;  
select tablespace\_name, status from dba\_tablespaces;  
9、知道每个表空间存在哪个磁盘上以及文件的名字等信息  
SELECT file\_id, file\_name, tablespace\_name, status, bytes from dba\_data\_files;  
select file\_name, tablespace\_name from dba\_data\_files;  
10、知道Oracle数据库系统上到底有多少用户和都是什么时候创建的  
select username,created from dba\_users;  
select username, DEFAULT\_TABLESPACE from dba\_users;  
11、从控制文件中取出信息涉及到以下一些相关的命令  
   
select \* from v$archived  
select \* from v$archived\_log  
select \* from v$backup  
select \* from v$database  
select \* from v$datafile  
select \* from v$log  
select \* from v$logfile  
select \* from v$loghist  
select \* from v$tablespace  
select \* from v$tempfile  
12、控制文件由两大部份组成：可重用的部份和不可重用的部分。可重用的部分的大小可用CONTROL\_FILE\_RECORD\_KEEP\_TIME参数来控制，该参数的默认值为7天，即可重用的部份的内容保留7天，一周之后这部份的内容可能被覆盖。可重用的部份是供恢复管理器来使用的，这部份的内容可以自动扩展。Oracle数据库管理员可以使用CREAT DATABASE或CREAT CONTROLFILE语句中的下列关键字（参数）来间接影响不可重用的部份的大小：  
 MAXDATAFILES  
 MAXINSTANCES  
 MAXLOGFILES  
 MAXLOGHISTORY  
 MAXLOGMEMBERS  
13、查看控制文件的配置  
SELECT type, record\_size, records\_total, records\_used FROM v$controlfile\_record\_section;  
14、如果您的显示被分成了两部分，您需要使用类似于set pagesize 100的SQL\*Plus命令先格式化输出。有关的格式化输出命令有以下这些：  
record\_size： 为每个记录的字节数。  
records\_total：为该段所分配的记录个数。  
records\_used：为该段所使用的记录个数。  
15、知道控制文件中的所有数据文件(DATAFILE)，表空间(TABLESPACE)，和重做日志(REDO LOG)所使用的记录情况  
SELECT type, record\_size, records\_total, records\_used  
 FROM v$controlfile\_record\_section  
 WHERE type IN ( 'DATAFILE', 'TABLESPACE', 'REDO LOG');  
16、获取控制文件名字  
select value from v$parameter where name ='control\_files';  
或者：select \* from v$controlfile  
17、如何在一个已经安装的Oracle数据库中添加或移动控制文件呢?  
以下是在一个已经安装的Oracle数据库中添加或移动控制文件的具体步骤：  
a、利用数据字典v$controlfile来获取现有控制文件名字。  
b、正常关闭Oracle数据库。  
c、将新的控制文件名添加到参数文件的CONTROL\_FILES参数中。  
d、使用操作系统的复制命令将现有控制文件复制到指定位置。  
e、重新启动Oracle数据库。  
f、利用数据字典v$controlfile来验证新的控制文件名字是否正确。  
g、如果有误重做上述操作，如果无误删除无用的旧控制文件。  
注：如果您使用了服务器初始化参数文件(SPFILE),您不能关闭Oracle数据库而且应该在第3步使用alter system set control\_files的Oracle命令来改变控制文件的位置。  
SQL> alter system set control\_files =  
 'D:\Disk3\CONTROL01.CTL',  
 'D:\Disk6\CONTROL02.CTL',  
 'D:\Disk9\CONTROL03.CTL' SCOPE=SPFILE;  
18、由于控制文件是一个极其种要的文件，除了以上所说的将控制文件的多个副本存在不同的硬盘上的保护措施外，在数据库的结构变化之后，您应立即对控制文件进行备份。可以用Oracle命令来对控制文件进行备份：  
alter database backup controlfile to 'D:\backup\control.bak';  
19、您也可将备份到一个追踪文件中。该追踪文件包含有重建控制文件所需的SQL语句。可使用以下SQL语句来产生这一追踪文件：  
alter database backup controlfile to trace;  
20、正常关闭oracle命令  
shutdown immeditae  
  
  
  
  
本文来自CSDN博客，转载请标明出处：http://blog.csdn.net/foresail/archive/2007/10/26/1844695.aspx  
  
  
Oracle数据库命令集   
1、sqlplus启动方式：   
Dos下运行sqlplus 用户名/密码 as sysdba   
例如：c:>sqlplus sys/password AS sydba(系统帐户)   
或者：c:>sqlplus scott/password   
2、用户连接(切换)操作：   
sql>conn Scott/password(默认为tiger)   
注：conn与connect使用方法一样   
3、手动解锁：   
sql>ALTER USER scott Account Unlock;(将scott帐户解锁)   
4、查看用户(两种方式)：   
sql>select user form dual;   
sql>show user;   
5、sql帮助：   
sql>help index;   
6、数据库卸载和加载   
sql>shutdown [NORMAL][IMMEDLATE][ABORT][TRANSACTIONAL];   
sql>startup;   
7、查询select操作：   
sql>select \*from dept;   
sql>select dname,loc from dept;   
sql>select \*from emp where deptno=20;   
注：在sql中，字符串使用单引号。   
8、字段别名的定义：   
sql>select hiredate AS 雇用日期 from emp   
AS可以省略，但是推荐使用。   
9、字符串连接   
sql>selcet dname||loc from dept;   
10、提取表中的前N个记录：   
sql>select \*from emp where rownum<=N;   
11、null的使用：   
sql>select ename,job from emp where comm is[not] null;   
注：null参与数学表达式计算时，结果一定为null。   
12、like关键字(字符串匹配)：   
sql>select dname from dept where dname like '\_S%';   
注：\_代表单个字符，%代表多个字符。   
13、in关键字：   
sql>select dname,dpart,log from dept where deptno in(10,20);   
注：只选择输出deptno值为10或20的记录。   
类似于：where deptno=10 or deptno=20;   
14、between and关键字：   
sql>select \* from dept where deptno between 10 and 30;   
包括10和30.   
15、set关键字：   
sql>set pagesize 10;   
sql>set linesize 100;   
16、连接查询：   
求解sal大于平均值的记录   
sql>select \* from emp   
 where sal>(select avg(sal) from emp);   
17、联合查询：   
字段个数和类型必须保持一致，合并为一个查询结果。   
sql>select empno,ename from emp   
 union[ALL]   
 select deptno,dname from dept;   
ALL：代表可以有重复记录。   
18、内连接查询：   
内连接查询分为2中：一是在form后直接加表名，中间用逗号分隔。   
sql>select \* from emp e,dept d (e d为表的别名)   
 where e.deptno=d.deptno;   
二是使用join关键字，条件使用ON   
sql>select \* from emp join dept   
 ON emp.deptno=dept.deptno;   
19、自连接查询：将一个表起不同的别名，使其在逻辑上成为两张表。   
sql>select \* from emp a,emp b   
 where a.mgr=b.empno;   
20、外连接查询：   
左外连接查询：left join 左表全部显示   
右外连接查询：right join右表全部显示   
全外连接查询：full join 左右表均全部显示   
以左外连接为例：   
sql>select \* from dept   
 left join emp   
 on dept.deptno=emp.deptno;   
左表(dept)信息全部显示，右表(emp)中不符合条件的记录为null。   
21、集合函数(数学表达式)：   
sql>select AVG(sal) from emp;   
sql>select Count([distinct]sal) from emp;   
distinct表示不计算重复数据。   
22、插入记录：   
sql>Insert Into tablename (字段列表) values(属性类表)；   
sql>Insert Into emp(eno,ename,dept) values(1,'java',20);   
23、修改(更新)记录：   
sql>UpDate emp set deptno=80   
 where deptno=81;   
24、删除记录：   
sql>Delete from tablename where 条件   
sql>Delete from emp where deptno=80;   
25、创建视图：注意权限，本操作是在sys帐户下进行的。   
sql>create view view\_name(ename,dname)   
 AS   
 select emp.ename,dept.dname   
 from scott.emp,scott.dept   
 with read only;   
sql>select \* from view\_name;   
26、同义词   
 create synonym 同义词名称 for 原对象   
sql>create synonym empbak for scott.emp;   
sql>select \* from empbak;   
27、修改对象(表、视图等)结构ALTER：   
sql>alter table 表名 add("book\_id"VARCHAR2(10));//添加字段，注意双引号   
sql>alter table 表名 modify("book\_id",not null);   
28、修改表的操作   
sql>rename 原表名 to 新表名//修改表名   
sql>select \* from tab;//查找当前用户表信息   
sql>desc 表名//查看表结构   
sql>drop table 表名//删除表   
sql>truncate table 表名//删除表中数据，保留表结构   
29、事务处理   
sql>commit 成功提交事务   
sql>rollback 返回到事务未开始的状态   
sql>savepoint A 创建事务的保存点   
sql>rollback to [savepoint] A 返回保存点   
30、EXISTS关键字   
sql>select \* from dept where EXISTS (select deptno from emp);   
31、从另外一个表中拷贝记录   
sql>Insert Into emp(empno,job)   
 select dname from dept where deptno=50;   
sql>insert into (select deptno,dname from dept)   
 values (50,'Java');   
32、更新(2个表操作)表记录：   
sql>update emp set ename=(select dname from dept where deptno=50 ),   
 job=(select loc from dept where deptno=50 )   
 where ename like 'S%'   
33、删除(2个表操作)操作：   
sql>delete from emp where empno=   
 (select deptno from dept where deptno=50)   
34、合并操作MERGE INTO：   
sql>MERGE INTO 表名   
 Using 表名   
 ON(条件)   
 when Matched then (匹配)   
update set ....   
 when not Matched then   
insert into....   
34、any/some操作，代表满足any后面的任何一个条件即可：   
sql>select \* from emp   
 where sal<=any(2500,6000,10000);   
35、all操作，代表满足all后面的所有条件：   
sql>select \* from emp   
 where sal<=all(2500,6000,10000);   
37、list操作：列出缓冲区的内容，可简写l，后接整型数字或为空   
sql>list(n)   
38、/操作：执行缓冲区sql命令或PL/SQL块   
39、change操作：修改缓冲区里的信息，可简写c   
sql>c/fron/from   
将fron改写为from   
40、del操作：删除指定行缓冲区命令   
sql>del 2 3 删除第2行、第3行的信息。   
41、在缓冲区中追加信息(追加到末尾)   
sql>append where select;   
42、save命令，保存缓冲区内容到指定的文件中：   
sql>save c:\select.sql;   
43、get命名(与save对应)   
sql>get c:\select.sql;   
44、@命令(与get相似，get是加载语句，然后执行，而@是加载和执行同时进行)   
sql>@ c:\select.sql;   
45、edit命令，载入windows操作系统的可编辑窗口，简写ed：   
sql>edit   
  
  
一、SQL PLUS   
1 引言   
  
  
SQL命令   
以下17个是作为语句开头的关键字：   
alter drop revoke   
audit grant rollback\*   
commit\* insert select   
comment lock update   
create noaudit validate   
delete rename   
这些命令必须以“;”结尾   
带\*命令句尾不必加分号，并且不存入SQL缓存区。   
  
  
SQL中没有的SQL\*PLUS命令   
这些命令不存入SQL缓存区   
@ define pause   
# del quit   
$ describe remark   
/ disconnect run   
accept document save   
append edit set   
break exit show   
btitle get spool   
change help sqlplus   
clear host start   
column input timing   
compute list ttitle   
connect newpage undefine   
copy   
  
  
-------   
2 数据库查询   
  
  
数据字典   
TAB 用户创建的所有基表、视图和同义词清单   
  
  
DTAB 构成数据字典的所有表   
  
  
COL 用户创建的基表的所有列定义的清单   
  
  
CATALOG 用户可存取的所有基表清单   
  
  
select from tab;   
  
  
describe命令 描述基表的结构信息   
describe dept   
  
  
select   
from emp;   
  
  
select empno,ename,job   
from emp;   
  
  
select from dept   
order by deptno desc;   
  
  
逻辑运算符   
= !=或<> > >= < <=   
in   
between value1 and value2   
like   
%   
\_   
in null   
not   
no in,is not null   
  
  
谓词in和not in   
有哪些职员和分析员   
select ename,job   
from emp   
where job in ('clerk','analyst');   
  
  
select ename,job   
from emp   
where job not in ('clerk','analyst');   
  
  
谓词between和not between   
哪些雇员的工资在2000和3000之间   
select ename,job,sal from emp   
where sal between 2000 and 3000;   
  
  
select ename,job,sal from emp   
where sal not between 2000 and 3000;   
  
  
谓词like,not like   
select ename,deptno from emp   
where ename like 'S%';   
(以字母S开头)   
select ename,deptno from emp   
where ename like '%K';   
(以K结尾)   
select ename,deptno from emp   
where ename like 'W\_\_\_';   
(以W开头，后面仅有三个字母)   
select ename,job from emp   
where job not like 'sales%';   
(哪些雇员的工种名不以sales开头)   
  
  
谓词is null,is not null   
没有奖金的雇员（即commision为null）   
select ename,job from emp   
where comm is null;   
  
  
select ename,job from emp   
where comm is not null;   
  
  
多条件查询   
select ename,job   
from emp   
where deptno=20   
and job!='clerk';   
  
  
表达式   
/   
  
  
算术表达式   
选择奖金高于其工资的5%的雇员   
select ename,sal,comm,comm/sal from emp   
where comm>.05\*sal   
 order by comm/sal desc;   
  
  
日期型数据的运算   
add two days to 6Mar-87   
6-Mar-87 2 = 8-Mar-87   
add two hours to 6-Mar-87   
6-Mar-87 2/24 = 6-Mar-87 and 2hrs   
add 15 seconds to 6-Mar-87   
6-Mar-87 15/(24\*60\*60) = 6-Mar-87 and 15 secs   
  
  
列名的别名   
select ename employee from emp   
 where deptno=10;   
（别名：employee）   
select ename,sal,comm,comm/sal "C/S RATIO" from emp   
 where comm>.05\*sal   
 order by comm/sal desc;   
  
  
SQL命令的编辑   
list or l 显示缓冲区的内容   
list 4 显示当前SQL命令的第4行，并把第4行作为当前行，在该行号后面有个\*。   
change or c 用新的内容替换原来在一行中第一次出现内容   
SQL>c/(...)/('analyst')/   
input or i 增加一行或多行   
append or a 在一行后追加内容   
del 删除当前行 删除SQL缓冲区中的当前行   
run 显示并运行SQL缓冲区中的命令   
/ 运行SQL缓冲区中的命令   
edit 把SQL缓冲区中的命令写到操作系统下的文本文件，   
并调用操作系统提供的编辑器执行修改。   
  
  
-----------   
3 数据操纵   
数据的插入   
insert into dept   
 values (10,'accounting','new york');   
  
  
insert into dept (dname,deptno)   
 values ('accounting',10);   
  
  
从其它表中选择插入数据   
insert into emp (empno,ename,deptno)   
select id,name,department   
from old\_emp   
where department in(10,20,30,40);   
  
  
使用参数   
insert into dept   
 values(&deptno,&dname,&loc);   
执行时，SQL/PLUS对每个参数将有提示用户输入   
  
  
参数对应日期型或字符型数据时，可在参数上加引号，输入时就可不用引号   
insert into dept   
 values(&deptno,'&dname','&loc');   
  
  
插入空值（NULL）   
insert into dept   
 values(50,'education',null);   
  
  
插入日期型数据   
日期型数据缺省格式：DD-MON-YY   
insert into emp   
(empno,ename,hiredate)   
values(7963,'stone','07-APR-87');   
  
  
系统时间：SYSDATE   
insert into emp   
(empno,ename,hiredate)   
values(7600,'kohn',SYSDATE);   
  
  
数据更新   
update emp   
set job='manager'   
where ename='martin';   
  
  
update emp   
set job='market rep'   
where ename='salesman';   
  
  
update emp   
set deptno=40,job='market rep'   
where job='salesman';   
  
  
数据删除   
delete emp   
where empno=765;   
  
  
更新的提交   
commit   
  
  
自动提交方式   
set autocommit on   
如果状态设为开，则使用inesrt,update,delete会立即提交。   
  
  
更新取消   
rollback   
  
  
两次连续成功的commit之间的操作，称为一个事务   
  
  
-------------   
4 创建基表、视图   
创建基表   
create table dept   
(deptno number(2),   
dname char(14),   
loc char(13));   
  
  
数据字典会自动更新。   
一个基表最多254列。   
  
  
表名列名命名规则：   
限制   
第一个字符必须是字母，后面可任意（包括 $ # 但不能是逗号）。   
名字不得超过30个字符。   
  
  
唯一   
某一用户的基表名必须唯一，不能是ORACLE的保留字，同一基表的列名互不相同。   
  
  
使用双引号   
如果表名用双引号括起来，则可不满足上述规则；   
只有使用双引号，才能区别大、小写；   
命名时使用了双引号，在以后的操作也必须使用双引号。   
  
  
数据类型：   
char(n) （不得超过240字符）   
number(n,d)   
date   
long （最多65536字符）   
raw （二进制原始数据）   
  
  
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 dept   
add (headcnt number(3));   
  
  
修改已有列属性   
alter table dept   
modify 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));   
  
  
创建视图   
create view managers as   
select ename,job,sal   
from emp   
where job='manager';   
  
  
为视图列名取别名   
create view mydept   
(person,title,salary)   
as select ename,job,sal   
from emp   
where deptno=10;   
  
  
with check option选项   
使用with check option，保证当对视图插入或更新数据时，   
该数据必须满足视图定义中select命令所指定的条件。   
create view dept20 as   
select ename,job,sal,deptno   
from emp   
where deptno=20   
with check option;   
在做下述操作时，会发生错误   
update dept20   
set deptno=30   
where ename='ward';   
  
  
基表、视图的拷贝   
create table emp2   
as select from emp;   
  
  
基表、视图的删除   
drop table 表名   
drop view 视图名   
  
  
----------   
5 SQLPLUS报表功能   
SQL\*PLUS的一些基本格式命令   
column deptno heading department   
  
  
column ename heading name   
  
  
column sal heading salary   
  
  
column sal format $99,999.00   
  
  
ttitle sample report for|hitech corp   
  
  
btitle strictly confidential   
  
  
break on deptno   
  
  
compute sum of sal on deptno   
  
  
run   
  
  
表头和表尾   
ttitle sample report for|hitech corp   
btitle right strictly confidential   
  
  
“|”表示换行，结尾不必加分号   
选项有三种：left right center   
  
  
使用TTITLE，系统将自动地在每页的顶部显示日期和页号。   
TTITLET和BTITLE命令有效，直至重新设置表头或表尾，或退出SQL\*PLUS。   
  
  
下面命令使标题语句失效   
TTITLE OFF   
BTITLE OFF   
  
  
列名   
column命令定义用于显示列名   
若名字为一个单词，不必加引号   
column ename heading employee   
  
  
column ename heading 'employee|name'   
（|为换行）   
  
  
取消栏定义   
column ename clear   
  
  
列的格式   
column ename format A15   
  
  
column sal format $9,999.99   
  
  
column comm like sal   
  
  
like子句，使得某一列的格式参照另一列格式，它拷贝列名及其格式   
  
  
控制记录显示分组顺序   
break on deptno   
（不显示重复值）   
  
  
select deptno,ename   
from emp   
order by deptno;   
（ORDER BY子句用于控制BREAK）   
  
  
显示为   
10 clark   
 niller   
20 smith   
 scott   
30 allen   
 blake   
  
  
每次只有一个BREAK命令起作用，但一次可以在多个列上使用BREAK命令   
break on 列名1 on 列名2   
  
  
记录分组   
break on deptno skip 2   
select deptno,ename   
from emp   
order by deptno;   
  
  
每个deptno之间空两行   
clear break（取消BREAK命令）   
break on page（每次从一新页开始）   
break on report（每次从一新报表开始）   
break on page on report（联合使用）   
  
  
分组计算   
break on deptno skip 2   
compute sum of sal on deptno   
计算每个部门的工资总和   
skip子句使部门之间的信息分隔开   
  
  
其他计算命令   
compute avg of sal on deptno（平均值）   
count 非空值的总数   
MAX 最大值   
MIN 最小值   
STD 标准偏差   
VAR 协方差   
NUMBER 行数   
  
  
使compute命令失效   
一旦定义了COMPUTE，则一直有效，直到   
关闭COMPUTE（clear compute）   
  
  
SQL/PLUS环境命令   
show 选项   
（显示当前参数设置情况）   
  
  
show all（显示全部参数）   
  
  
设置参数   
set 选项 值或开关   
  
  
set autocommit on   
  
  
SET命令包括   
set autocommit {off|on|immediate}   
（自动提交，OFF缺省）   
  
  
set echo {off|on}   
（命令文件执行，是否在终端上显示命令本身，OFF缺省）   
  
  
set feedback {off|on}   
（ON：查询结束时，给出结果，记录数的信息，缺省；   
OFF：无查询结果，记录数的信息）   
  
  
set heading {off|on}   
（ON：列的头标在报表上显示，缺省；OFF：不在报表上显示）   
  
  
set linesize {n}   
一行显示的最大字符数，缺省为80   
  
  
set pagesize {n}   
每页的行数，缺省是14   
  
  
set pause {off|on|text}   
（ON：当报表超过一页时，每显示一屏就暂停显示，等待用户打回车键，再接着显示；   
OFF：页与页不停顿，缺省；text：页与页停顿，并向用户提示信息）   
  
  
SET BUFFER buffer   
设置当头的命令缓冲区，通常情况下，SQL命令缓冲区已为当前缓冲区。   
由于SQL命令缓冲区只能存放一条SQL命令，   
所以可用其它缓冲区来存放SQL命令和SQL\*PLUS命令。   
  
  
经常用到的设置可放在login.sql文件中。   
  
  
SET NULL   
set null 'no data'   
  
  
select ename,comm   
from emp   
where deptno=30;   
把部门30中无佣金雇员的佣金显示为“NO DATA”。   
  
  
set null是SQL\*PLUS命令，用它来标识空值（NULL），可以设置为任意字符串。   
  
  
存盘命令SAVE   
save 文件名   
  
  
input   
1 select empno,ename,job   
2 from emp   
3 where job='analyst'   
  
  
save research   
  
  
目录中会增加一个research.sql文件。   
  
  
编辑命令EDIT   
edit   
  
  
EDIT编辑当前缓冲区中的内容。   
  
  
编辑一个文件   
edit research   
  
  
调入命令GET   
get research   
把磁盘上的文件内容调入缓冲区，并显示在屏幕上，文件名尾不必加.sql。   
  
  
START命令   
运行指定的文件   
start research   
  
  
输出命令SPOOL   
spool tryfile   
不仅可以使查询结果在屏幕上显示，还可以使结果存入文件   
  
  
停止向文件输出   
spool off   
  
  
把查询结果在打印机上输出，先把它们存入一个文件中，   
然后不必使用SPOOL OFF，而用：   
spool out   
SPOOL OUT关闭该文件并在系统缺省的打印机上输出   
  
  
制作报表举例   
edit tryfile   
  
  
set echo off   
set autocommit on   
set pagesize 25   
insert into emp (empno,ename,hiredate)   
 values(9999,'geiger',sysdate);   
insert into emp (empno,ename,deptno)   
 values(3333,'samson',20);   
spool newemp   
select from emp   
 where deptno=20   
or deptno is null   
/   
spool off   
set autocommit off   
  
  
用start命令执行这个文件   
  
  
------   
6 函数   
字符型函数   
initcap(ename);将ename中每个词的第一个字母改为大写。   
如：jack smith--Jack Smith   
  
  
length(ename);计算字符串的长度。   
  
  
substr(job,1,4);   
  
  
其它   
lower   
upper   
least 取出字符串列表中按字母排序排在最前面的一个串   
greatest 取出字符串列表中按字母排序排在最后的一个串   
  
  
日期函数   
add\_month(hiredate,5) 在雇佣时间上加5个月   
month\_between(sysdate,hiredate) 计算雇佣时间与系统时间之间相差的月数   
next\_day(hiredate,'FRIDAY') 计算受雇日期之后的第一个星期五的日期   
  
  
例   
select ename,sal,next\_day(sysdate,'FRIDAY') as\_of   
from emp   
where deptno=20;   
（as\_of是别名）   
  
  
如果不用to\_char函数，日期在ORACLE中的缺省格式是'DD\_MON\_YY'   
to\_char(date,date picture)   
  
  
select ename,to\_char(hiredate,'Dy Mon dd,yyyy') hired   
from emp   
where deptno=10;   
  
  
to\_date(字符串,格式)   
  
  
insert into emp(empno,ename,hiredate)   
values(7999,'asms',to\_date('070387083000','MMDDYYHHMISS'));   
  
  
日期型数据的格式   
dd 12   
dy fri   
day friday   
ddspth twelfth   
  
  
mm 03   
mon mar   
month march   
  
  
yy 87   
yyyy 1987   
  
  
例   
Mar 12,1987 'Mon dd,yyyy'   
MAR 12,1987 'MON dd,yyyy'   
Thursday MARCH 12 'Day MONTH dd'   
Mar 12 11:00am 'Mon dd hh:miam'   
Thu,the twelfth 'Dy,"the"ddspth'   
  
  
算术函数   
least(v1,v2)   
  
  
select ename,empno,mgr,least(empno,mgr) lownum   
from emp   
where empno0   
  
  
trunc(sal,0)   
取sal的近似值（截断）   
  
  
空值函数   
nvl(v1,v2)   
v1为列名，如果v1不是空值，nvl返回其列值。   
v1为空值，返回v2的值。   
  
  
聚组函数   
select sum(comm)   
from emp;   
（返回一个汇总信息）   
不能把sum用在select语句里除非用group by   
  
  
字符型、日期型、数字型的聚组函数   
min max count可用于任何数据类型   
  
  
select min(ename)   
from emp;   
  
  
select min(hiredate)   
from emp;   
  
  
select min(sal)   
from emp;   
  
  
有多少人有工作？   
select count(job)   
from emp;   
  
  
有多少种不同的工种？   
select count(distinct job)   
from emp;   
  
  
count distinct 计算某一字段中不同的值的个数   
  
  
其它聚组函数（只用于数字型数据）   
avg 计算平均工资   
select avg(sal)   
from emp;   
  
  
stddev 计算工资的平均差   
select stddev(sal)   
from emp;   
  
  
sum 计算总工资   
select sum(sal)   
from emp;   
  
  
group by子句   
select deptno,sum(sal),avg(sal)   
from emp   
group by deptno;   
  
  
按多个条件分组   
每个部门的雇员数   
select deptno,count()   
from emp   
group by deptno;   
  
  
每个部门的每个工种的雇员数   
select deptno,job,count(\*)   
from emp   
group by deptno,job;   
  
  
满足条件的分组   
（where是针对select的，having是针对group by的）   
哪些部门的工资总和超过了9000   
select deptno,sum(sal)   
from emp   
group by deptno   
having sum(sal)>9000;   
  
  
select小结   
除去职员，哪些部门的工资总和超过了8000   
select deptno,sum(sal)   
from emp   
where job!='clerk'   
group by deptno   
having sum(sal)>8000   
order by sum(sal);   
  
  
-------   
7 高级查询   
等值联接   
select empno,ename,job,emp.deptno,dname   
from emp,dept   
where emp.deptno=dept.deptno;   
  
  
外联接   
select ename,dept.deptno,loc   
from emp,dept   
where emp.deptno(+)=dept.deptno;   
如果在dept.deptno中有的数值，在emp.deptno中没有（如deptno=40），   
则作外联接时，结果中会产生一个空值   
  
  
自联接：同一基表的不同行要做联接，可使用自联接   
指出每个雇员的经理名字   
select worker.ename,manager.ename manager   
from emp worker,emp manager   
where worker.mgr=manager.empno;   
  
  
非等值联接   
哪些雇员的工资属于第三级别   
select ename,sal   
from emp,salgrade   
where grade=3   
and sal between losal and hisal;   
（基表salgrade：grade losal hisal）   
  
  
集合运算   
行的连接   
集合运算把2个或多个查询结果合并为一个   
union-set union   
Rows of first query plus of second query, less duplicate rows   
  
  
intersect-set intersection   
Rows both queries have in common   
  
  
minus-set difference   
rows unique to the first query   
  
  
介绍几个视图   
account view   
ename sal job   
  
  
sales view   
ename sal job   
  
  
research view   
ename sal job   
  
  
union运算   
返回一个查询结果中有但又不重要的行，它将基表或视图中的记录合并在一起   
所有部门中有哪些雇员工资超过2000   
对应列的数据类型必须相同   
select ename,sal   
from account   
where sal>2000   
union   
select ename,sal   
from research   
where sal>2000   
union   
select ename,sal   
from sales   
where sal>2000;   
  
  
intersect运算   
返回查询结果中相同的部分   
各个部门中有哪些相同的工种   
select job   
from account   
intersect   
select job   
from research   
intersect   
select job   
from sales;   
  
  
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？   
select job from account   
minus   
select job from sales;   
  
  
子查询   
slect ename,deptno   
from emp   
where deptno=   
(select deptno   
from emp   
where ename='smith');   
  
  
多级子查询   
select ename,job,sal   
from emp   
where job=   
(select job   
from emp   
where ename='clark')   
or sal>   
(select sal   
from emp   
where ename='clark');   
  
  
多个基表与子查询   
select ename,job,sal   
from emp,dept   
where loc='new york'   
and emp.deptno=dept.deptno   
and sal>   
(select sal   
from emp   
where ename='scott');   
  
  
子查询中使用聚组函数   
select ename,hiredate   
from emp   
where hiredate=   
(select min(hiredate)   
from emp);   
  
  
8 授权   
系统权限   
DBA 所有权限   
RESOURCE 注册，创建新的基表   
CONNECT，注册，查询   
  
  
只有DBA才有权创建新的用户   
grant connect to scott   
identified by tiger;   
  
  
DBA或用户自己可以改变用户口令   
grant connect to scott   
identified by leopard;   
  
  
基表权限1   
有两种方法获得对基表操作的权限   
  
  
创建自己的基表   
获得基表创建用户的许可   
grant select,insert   
on emp   
to scott;   
  
  
这些权限有   
select insert update delete alter index   
  
  
把所有权限授于他人   
grant all on emp to scott;   
  
  
同义词   
select   
from scott.emp   
  
  
创建同义词   
为用户allen的EMP基表创建同义词employee   
create synonym employee   
for allen.emp   
  
  
基表权限2   
你可以使其他用户有这样的权力，即其他用户可以把你的基表权限授予他人   
grant all   
on emp   
to scott   
with grant option;   
  
  
收回权限   
系统权限 只有被DBA收回   
  
  
基表权限 随时都可以收回   
  
  
revoke insert   
on emp   
from scott;   
  
  
-------   
9 索引   
建立索引   
create index emp\_ename   
on emp(ename);   
  
  
删除索引   
drop index emp\_ename;   
  
  
关于索引   
只对较大的基表建立索引（至少50条记录）   
建立索引之前插入数据   
对一个基表可建立任意多个索引   
一般是在作为主键的列上建立索引   
建立索引之后，不影响SQL命令的执行   
建立索引之后，ORACLE自动维护和使用索引   
  
  
保证数据唯一性   
提高执行速度的同时，索引还可以保证每个记录中的每个列值是不重复的。   
create unique index emp\_empno   
on emp(empno);   
  
  
------   
练习和答案   
  
  
有没有工资比奖金多的雇员？如果有，按工资的降序排列。   
如果有两个以上的雇员工资相同，按他们的名字排序。   
select ename employee,sal salary,comm commision   
from emp   
where sal>comm   
order by sal desc,ename;   
  
  
列出有关雇员姓名、奖金占收百分比的信息。   
要求显示时列名意义明确，按雇员姓名排序，不包括奖金未知的雇员。   
select ename employee,(comm/(comm+sal))100 incentive   
from emp   
where comm is not null   
order by ename;   
  
  
在chicago（部门30）工作的所有雇员的工资上涨10%。   
update emp   
set sal=1.1\*sal   
where deptno=30;   
  
  
update emp   
set sal=1.1\*sal   
where deptno=(select deptno   
from dept   
where loc='chicago');   
  
  
为hitech公司新建一个部门，编号为50，其它信息均不可知。   
insert into dept(dname,deptno)   
values('faclities',50);   
  
  
创建视图，三个列名，其中不包括职员信息   
create view employee("employee name",   
"employee number",   
"employee job")   
as select ename,empno,job   
from emp   
where job!='clerk';   
  
  
制作工资报表，包括雇员姓名、受雇时间（按星期计算），工资和部门编号，   
一页显示一个部门的信息，每页尾，显示该页的工资之和以及受雇时间之和，   
报表结尾处，显示所有雇员的工资总和以及受雇时间总和，   
工资按美元计算，受雇时间按星期计算，每页的上方应有标题。   
ttitle 'service'   
break on deptno on page on report   
compute sum of sal on deptno   
compute sum of sal on report   
compute sum of service\_length on deptno   
compute sum of service\_length on report   
column sal format $99,999.00   
column service\_length format 9999   
select deptno,ename employee,(sysdate-hiredate)/7 service\_length,sal   
from emp   
order by deptno;   
  
  
制作报表，包括雇员姓名、总收入和受佣日期，   
且：姓名的第一个字母必须大写，雇佣日期格式为MM/DD/YYYY，   
总收入包括没有奖金的雇员的总收入，姓名按字母顺序排列。   
col "hire date"format A12   
col "employee" format A10   
col "compensation" format $99,999.00   
select initcap(ename) "employee",   
(sal+nvl(comm,0)) "compensation",   
to\_char(hiredate,'MM/DD/YYYY') "hire date"   
from emp   
order by ename;   
  
  
列出有超过7个周边国家的国家名字和面积。   
select nation,area   
from nation   
where code in   
(select nation\_code   
from border   
group by nation\_code   
having count(\*)>7);   
  
  
列出所有面积大于等于日本的岛国的国名和人口。   
select nation,population   
from nation,border   
where code=nation\_code(+)   
and nation\_code is null   
and area>=   
(select area   
from nation   
where upper(nation)='JAPAN');   
  
  
列出所有边界在其它国家中的国名，并且显示其边界国家名字。   
break on nation   
select nation1.nation,   
nation2.nation borderin\_country   
from nation nation1,border,nation nation2   
where nation1.code=border.nation\_code   
and border.border\_code=nation2.code   
order by nation1.nation;   
  
  
---------   
---------   
PL/SQL   
  
  
2 PL/SQL的块结构和数据类型   
  
  
块结构的特点   
嵌套   
begin   
......   
 begin   
 ......   
 exception   
 ......   
 end;   
exception   
......   
end;   
  
  
标识符：   
不能超过30个字符   
第一个字符必须为字母   
其余字符可以是字母，数字，$，\_，或#   
不区分大小写形式   
如果用双引号括起来，则字符顺序可以任意排列，并区分大小写形式   
无SQL保留字   
  
  
数据类型   
数字型：   
整数，实数，以及指数   
  
  
字符串：   
用单引号括起来   
若在字符串表示单引号，则使用两个单引号   
字符串长度为零（两个单引号之间没有字符），则表示NULL   
  
  
字符：   
长度为1的字符串   
  
  
数据定义   
语法   
标识符[常数] 数据类型[NOT NULL][:=PL/SQL表达式];   
':='表示给变量赋值   
  
  
数据类型包括   
数字型 number(7,2)   
字符型 char(120)   
日期型 date   
布尔型 boolean（取值为true,false或null，不存贮在数据库中）   
  
  
日期型   
anniversary date:='05-JUL-95';   
project\_completion date;   
  
  
布尔型   
over\_budget boolean not null:=false;   
available boolean;   
（初始值为NULL）   
  
  
%type类型匹配   
books\_printed number(6);   
books\_sold book\_printed%type;   
manager\_name emp.ename%type;   
  
  
变量赋值   
变量名:=PL/SQL表达式   
numvar:=5;   
boolvar:=true;   
datevar:='11-JUN-87';   
  
  
字符型、数字型表达式中的空值   
null+<数字>=null（空值加数字仍是空值）   
null><数字>=null（空值与数字进行比较，结果仍是空值）   
null||'字符串'='字符串'（null即''）   
（空值与字符串进行连接运算，结果为原字符串）   
  
  
变量作用范围   
标识符在宣言它的块中有效   
标识符如果不在子块中重新定义，则在PL/SQL块的所有子块中同样有效   
重新定义后的标识符，作用范围仅在本子块中有效   
  
  
例   
declare   
 e\_mess char(80);   
begin   
 /\*子块1\*/   
 declare   
 v1 number(4);   
 begin   
 select empno into v1 from emp   
 where job='president';   
 exception   
 when too\_many\_rows then   
 insert into job\_errors   
 values('more than one president');   
 end;   
 /\*子块2\*/   
 declare   
 v1 number(4);   
 begin   
 select empno into v1 from emp   
 where job='manager';   
 exception   
 when too\_many\_rows then   
 insert into job\_errors   
 values('more than one manager');   
 end;   
exception   
 when others then   
 e\_mess:=substr(sqlerrm,1,80);   
 insert into general errors values(e\_mess);   
end;   
  
  
-------   
3 SQL和PL/SQL   
  
  
插入   
declare   
 my\_sal number(7,2):=3040.55;   
 my\_ename char(25):='wanda';   
 my\_hiredate date:='08-SEP-88';   
begin   
 insert into emp   
 (empno,enmae,job,hiredate,sal,deptno)   
 values(2741,my\_ename,'cab driver',my\_hiredate,my\_sal,20);   
end;   
  
  
删除   
declare   
 bad\_child\_type char(20):='naughty';   
begin   
 delete from santas\_gift\_list where   
 kid\_rating=bad\_child\_type;   
end;   
  
  
事务处理   
commit[WORK];   
rollback[WORK];   
（关键字WORK可选，但对命令执行无任何影响）   
savepoint 标记名;（保存当前点）   
在事务中标记当前点   
rollback [WORK] to [SAVEPOINT] 标记名;（回退到当前保存点）   
取消savepoint命令之后的所有对数据库的修改   
关键字WORK和SAVEPOINT为可选项，对命令执行无任何影响   
  
  
函数   
PL/SQL块中可以使用SQL命令的所有函数   
insert into phonebook(lastname) value(upper(my\_lastname));   
select avg(sal) into avg\_sal from emp;   
  
  
对于非SQL命令，可使用大多数个体函数   
不能使用聚组函数和参数个数不定的函数，如   
x:=sqrt(y);   
lastname:=upper(lastname);   
age\_diff:=months\_between(birthday1,birthday2)/12;   
  
  
赋值时的数据类型转换   
4种赋值形式:   
变量名:=表达式   
insert into 基表名 values(表达式1,表达式2,...);   
update 基表名 set 列名=表达式;   
select 列名 into 变量名 from ...;   
  
  
数据类型间能进行转换的有:   
char转成number   
number转成char   
char转成date   
date转成char   
  
  
例   
char\_var:=nm\_var;   
数字型转换成字符型   
date\_var:='25-DEC-88';   
字符型转换成日期型   
insert into 表名(num\_col) values('604badnumber');   
错误，无法成功地转换数据类型   
  
  
-------   
4 条件控制   
例   
declare   
 num\_jobs number(4);   
begin   
 select count(\*) into num\_jobs from auditions   
 where actorid=&&actor\_id and called\_back='yes';   
 if num\_jobs>100 then   
 update actor set actor\_rating='word class'   
 where actorid=&&actor\_id;   
 elsif num\_job=75 then   
 update actor set actor\_rating='daytime soaps'   
 where actorid=&&actor\_id;   
 else   
 update actor set actor\_rating='waiter'   
 where actorid=&&actor\_id;   
 end if;   
 end if;   
 commit;   
end;   
  
  
------   
5 循环   
语法   
loop   
......   
end loop;   
exit;（退出循环）   
exit [when];（退出循环，当满足WHEN时）   
例1   
declare   
 ctr number(3):=0;   
begin   
 loop   
 insert into table1 values('tastes great');   
 insert into table2 values('less filling');   
 ctr:=ctr+1;   
 exit when ctr=100;   
 end loop;   
end;   
（注：如果ctr取为NULL，循环无法结束）   
  
  
例2   
FOR语法   
for 变量<范围> loop   
......   
end loop;   
  
  
declare   
 my\_index char(20):='fettucini alfredo';   
 bowl char(20);   
begin   
 for my\_index in reverse 21..30 loop   
 insert into temp(coll) values(my\_index);   
 /\*循环次数从30到21\*/   
 end loop;   
 bowl:=my\_index;   
end;   
跟在in reverse后面的数字必须是从小到大的顺序，必须是整数，不能是变量或表达式   
  
  
--------   
6 游标   
显式游标   
  
  
打开游标   
open <游标名>   
例   
open color\_cur;   
  
  
游标属性   
%notfound   
%found   
%rowcount   
%isopen   
例   
fetch my\_cur into my\_var;   
while my\_cur %found loop   
 (处理数据)   
 fetch my\_cur into my\_var;   
 exit when my\_cur %rowcount=10;   
end loop;   
  
  
%notfound属性   
取值情况如下：   
fetch操作没有返回记录，则取值为true   
fetch操作返回一条记录，则取值为false   
对游标无fetch操作时，取值为null   
<游标名> %notfound   
例   
if color\_cur %notfound then...   
注：如果没有fetch操作，则<游标名> %notfound将导致出错，   
因为%notfound的初始值为NULL。   
  
  
关闭游标   
close <游标名>   
例   
close color\_cur;   
  
  
游标的FOR循环   
语法   
for <记录名> in <游标名> loop   
<一组命令>   
end loop;   
其中：   
索引是建立在每条记录的值之上的   
记录名不必声明   
每个值对应的是记录名，列名   
初始化游标指打开游标   
活动集合中的记录自动完成FETCH操作   
退出循环，关闭游标   
  
  
隐式游标   
隐式游标是指SQL命令中用到的，没有明确定义的游标   
insert,update,delete,select语句中不必明确定义游标   
调用格式为SQL%   
存贮有关最新一条SQL命令的处理信息   
  
  
隐式游标的属性   
隐式游标有四个属性   
SQL%NOTFOUND   
SQL%FOUND   
SQL%ROWCOUNT：隐式游标包括的记录数   
例：   
delete from baseball\_team where batting\_avg<100;   
if sql%rowcount>5 thn   
 insert into temp   
 values('your team needs help');   
end if;   
  
  
SQL%ISOPEN：取值总为FALSE。SQL命令执行完毕，PL/SQL立即关闭隐式游标。   
  
  
  
  
7 标号   
GOTO语句   
用法：   
goto you\_are\_here;   
其中you\_are\_here是要跳转的语句标号   
标号必须在同一组命令，或是同一块中使用   
  
  
正确的使用   
<>（标号）   
x:=x+1   
if a>b then   
b:=b+c;   
goto dinner;   
end if;   
  
  
错误的使用   
goto jail;   
if a>b then   
b:=b+c;   
<>（标号）   
x:=x+1;   
end if;   
  
  
标号：解决意义模糊   
标号可用于定义列值的变量   
<>   
declare   
deptno number:=20;   
begin   
 update emp set sal=sal\*1.1   
 where deptno=sample.deptno;   
 commit;   
end sample;   
如果不用标号和标号限制符，这条命令将修改每条记录。   
  
  
--------   
8 异常处理   
预定义的异常情况   
任何ORACLE错误都将自动产生一个异常信息   
一些异常情况已命名，如：   
no\_data\_found 当SELECT语句无返回记录时产生   
too\_many\_rows 没有定义游标，而SELECT语句返回多条记录时产生   
whenever notfound 无对应的记录   
  
  
用户定义的异常情况   
由用户自己获取   
在DECLARE部分定义：   
declare   
 x number;   
 something\_isnt\_right exception;   
用户定义的异常情况遵循一般的作用范围规则   
条件满足时，获取异常情况：raise something\_isnt\_right   
注意：同样可以获取预定义的异常情况   
  
  
exception\_init语句   
允许为ORACLE错误命名   
  
  
调用格式：   
pragma exception\_init(<表达式>,);   
例   
declare   
 deadlock\_detected exception;   
 pragma exception\_init(deadlock\_detected,-60);   
  
  
raise语句   
单独使用RAISE命令，可再一次获取当前的异常情况（就象异常情况被重复处理了一样）。   
在异常处理中，此语句只能单独使用。   
  
  
异常处理标识符   
一组用于处理异常情况的语句：   
exception   
 when <表达式> or [表达式...] then   
 <一组语句>   
 ...   
 when others then--最后一个处理   
 <一组语句>   
end;既结束PL/SQL块部分，也结束异常处理部分   
  
  
------   
练习与答案   
1:   
接收contract\_no和item\_no值，在inventory表中查找，如果产品：   
已发货，在arrival\_date中赋值为今天后的7天   
已订货，在arrival\_date中赋值为今天后的一个月   
既无订货又无发货，则在arrival\_date中赋值为今天后的两个月，   
并在order表中增加一条新的订单记录。   
  
  
product\_status的列值为'shipped'和'ordered'   
  
  
inventory:   
product\_id number(6)   
product\_description char(30)   
product\_status char(20)   
std\_shipping\_qty number(3)   
  
  
contract\_item:   
contract\_no number(12)   
item\_no number(6)   
arrival\_date date   
  
  
order:   
order\_id number(6)   
product\_id number(6)   
qty number(3)   
  
  
答案：   
declare   
i\_product\_id inventory.product\_id%type;   
i\_product\_description inventory.product\_description%type;   
i\_product\_status inventory.product\_status%type;   
i\_std\_shipping\_qty inventory.std\_shipping\_qty%type;   
  
  
begin   
 select product\_id,product\_description,product\_status,std\_shipping\_qty   
 into i\_product\_id,i\_product\_description,   
 i\_product\_status,i\_std\_shipping\_qty   
 from inventory   
 where product\_id=(   
 select product\_id   
 from contract\_item   
 where contract\_no=&&contractno and item\_no=&&itemno);   
 if i\_product\_status='shipped' then   
 update contract\_item   
 set arrival\_date=sysdate+7   
 where item\_no=&&itemno and contract\_no=&&contractno;   
 elsif i\_product\_status='ordered' then   
 update contract\_item   
 set arrival\_date=add\_months(sysdate,1)   
 where item\_no=&&itemno and contract\_no=&&contractno;   
 else   
 update contract\_item   
 set arrival\_date=add\_months(sysdate,2)   
 where item\_no=&&itemno and contract\_no=&&contractno;   
 insert into orders   
 values(100,i\_product\_id,i\_std\_shipping\_qty);   
 end if;   
 end if;   
 commit;   
end;   
  
  
  
  
2:   
1.找出指定部门中的所有雇员   
2.用带'&'的变量提示用户输入部门编号   
3.把雇员姓名及工资存入prnttable表中，基结构为：   
create table prnttable   
(seq number(7),line char(80));   
4.异常情况为，部门中奖金不为空值的雇员信息才能存入prnttable表中。   
  
  
答案：   
declare   
cursor emp\_cur is   
 select ename,sal,comm   
 from emp where deptno=&dno;   
emp\_rec emp\_cur%rowtype;   
null\_commission exception;   
begin   
 open emp\_cur;   
 fetch emp\_cur into emp\_rec;   
 while (emp\_cur%found) loop   
 if emp\_rec.comm is null then   
 begin   
 close emp\_cur;   
 raise null\_commission;   
 end;   
 end if;   
 fetch emp\_cur into emp\_rec;   
 end loop;   
 close emp\_sur;   
 exception   
 when null\_commission then   
 open emp\_cur;   
 fetch emp\_cur into emp\_rec;   
 while (emp\_cur%found) loop   
 if emp\_rec.comm is not null then   
 insert into temp values(emp\_rec.sal,emp\_rec.ename);   
 end if;   
 fetch emp\_cur into emp\_rec;   
 end loop;   
 close emp\_cur;   
 commit;   
end   
  
  
第一课：客户端   
 1. Sql Plus(客户端），命令行直接输入：sqlplus，然后按提示输入用户名，密码。   
 2. 从开始程序运行:sqlplus，是图形版的sqlplus.   
 3. http://localhost:5560/isqlplus   
   
 Toad：管理， PlSql Developer:   
   
   
第二课：更改用户   
 1. sqlplus sys/bjsxt as sysdba   
 2. alter user scott account unlock;(解锁)   
   
第三课：table structure   
   
 1. 描述某一张表：desc 表名   
 2. select \* from 表名   
第四课：select 语句：   
   
 1.计算数据可以用空表：比如：.select 2\*3 from dual   
   
 2.select ename,sal\*12 annual\_sal from emp;与select ename,sal\*12 "annual sal" from emp;区别，加双引号保持原大小写。不加全变大写。   
   
 3. select ename || "abcd" 如果连接字符串中含有单引号，用两个单引号代替一个单引号。   
   
第五课：distinct   
 select deptno from emp;   
 select distinct deptno from emp;   
   
 select distinct deptno from emp;   
 select distinct deptno ,job from emp   
 去掉deptno,job两者组合的重复。更多的项，就是这么多项的组合的不重复组合。   
第六课：Where   
 select \* from emp where deptno =10;   
 select \* from emp where deptno <>10;不等于10   
 select \* from emp where ename ='bike';   
 select ename,sal from emp where sal between 800 and 1500 (>=800 and <=1500)   
 空值处理:   
 select ename,sal,comm from emp where comm is (not) null;   
 select ename,sal,comm from emp where ename ( not)in ('smith','king','abc');   
 select ename from emp where ename like '\_A%';\_代表一个字母,%代表0个或多个字母. 如果查询%   
 可用转义字符.\%. 还可以用escape '$'比如:select ename from emp where ename like '%$a%' escape '$';   
   
第七课: orderby   
   
 select \* from dept;   
 select \* from dept order by dept desc;(默认:asc)   
 select ename,sal,deptno from emp order by deptno asc,ename desc;   
   
第八课: sql function1:   
 select ename,sal\*12 annual\_sal from emp   
 where ename not like '\_A%' and sal>800   
 order by sal desc;   
   
 select lower(ename) from emp;   
   
 select ename from emp   
 where lower(ename) like '\_a%';等同于   
 select ename from emp where ename like '\_a%' or ename like '\_A%';   
   
 select substr(ename,2,3) from emp;从第二字符截,一共截三个字符.   
 select chr(65) from dual 结果为:A   
 select ascii('a') from dual 结果为:65   
 select round(23.652,1) from dual; 结果为: 23.7   
 select round(23.652,-1) from dual; 20   
   
 select to\_char(sal,'$99\_999\_999') from emp;   
 select to\_char(sal,'L99\_999\_999') from emp;人民币符号,L:代表本地符号   
   
 这个需要掌握牢:   
 select birthdate from emp;   
 显示为:   
 BIRTHDATE   
 ----------------   
 17-12月-80   
 ----------------   
   
 改为:   
 select to\_char(birthdate,'YYYY-MM-DD HH:MI:SS') from emp;   
   
 显示:   
   
 BIRTHDATE   
 -------------------   
 1980-12-17 12:00:00   
 -------------------   
   
 select to\_char(sysdate,'YYYY-MM-DD HH24:MI:SS') from dual; //也可以改为:HH12   
 TO\_CHAR(SYSDATE,'YY   
 -------------------   
 2007-02-25 14:46:14   
   
   
 to\_date函数:   
 select ename,birthdate from emp where birthdate > to\_date('1981-2-20 12:34:56','YYYY-MM-DD HH24:MI:SS');   
 如果直接写 birthdate>'1981-2-20 12:34:56'会出现格式不匹配,因为表中的格式为: DD-MM月-YY.   
   
   
 select sal from emp where sal>888.88 无错.但   
 select sal from emp where sal>$1,250,00;   
 会出现无效字符错误.   
 改为:   
 select sal from emp where sal>to\_number('$1.250.00','$9,999,99');   
   
 把空值改为0   
 select ename,sal\*12+nvl(comm,0) from emp;   
 这样可以防止comm为空时,sal\*12相加也为空的情况.   
   
   
第九课: Group function 组函数   
 max,min,avg ,count,sum函数   
   
 select to\_char(avg(sal),'99999999,99') from emp;   
   
   
 select round(avg(sal),2) from emp;   
 结果:2073.21   
   
 select count(\*) from emp where deptno=10;   
 select count(ename) from emp where deptno=10; count某个字段,如果这个字段不为空就算一个.   
 select count(distinct deptno) from emp;   
 select sum(sal) from emp;   
   
第十课: Group by语句   
   
 需求:现在想求,求每个部门的平均薪水.   
 select avg(sal) from emp group by deptno;   
 select deptno avg(sal) from emp group by deptno;   
   
 select deptno,job,max(sal) from emp group by deptno,job;   
   
 求薪水值最高的人的名字.   
 select ename,max(sal) from emp;出错,因为max只有一个值,但等于max值的人可能好几个,不能匹配.   
 应如下求:   
 select ename from emp where sal=(select max(sal) from emp);   
   
 Group by语句应注意,   
   
 出现在select中的字段,如果没出现在组函数中,必须出现在Group by语句中.   
   
   
 第十一课: Having 对分组结果筛选   
   
 Where是对单条纪录进行筛选,Having是对分组结果进行筛选.   
   
 select avg(sal),deptno from emp   
 group by deptno   
 having avg(sal)>2000;   
   
 查询工资大于1200雇员,按部门编号进行分组,分组后平均薪水大于1500,按工薪倒充排列.   
 select \* from emp   
 where sal>1200   
 group by deptno   
 having avg(sal)>1500   
 order by avg(sal) desc;   
   
 第十二课:字查询   
   
 谁挣的钱最多(谁:这个人的名字, 钱最多)   
   
 select 语句中嵌套select 语句,可以在where,from后.   
   
   
 问那些人工资,在平均工资之上.   
   
 select ename,sal from emp where sal>(select avg(sal) from emp);   
   
   
 查找每个部门挣钱最多的那个人的名字.   
 select ename ,deptno from emp where sal in(select max(sal) from ename group by deptno) 查询会多值.   
   
 应该如下:   
   
 select max(sal),deptno from emp group by deptno;当成一个表.语句如下:   
 select ename, sal from emp join(select max(sal) max\_sal,deptno from emp group   
 by deptno) t on(emp.sal=t.max\_sal and emp.deptno=t.deptno);   
   
 每个部门的平均薪水的等级.   
 分析:首先求平均薪水(当成表),把平均薪水和另外一张表连接.   
   
   
第十四课:self\_table\_connection   
   
 把某个人的名字以及他的经理人的名字求出来(经理人及这个人在表中同处一行)   
   
 分析:首先求出这个人的名字,取他的编号,然后从另一张表与其相对应编号,然后找到经理的名字.   
   
 select e1.ename ,e2.ename from emp e1,emp e2 where e1.mgr= e2.empno.   
   
 empno编号和MGR都是编号.   
   
   
第十15课: SQL1999\_table\_connections   
   
 select ename,dname,grade from emp e,dept d, sqlgrade s   
 where e.deptno = d.deptno and e.sql between s.losal and s.hisal and   
 job<>'CLERK';   
   
 有没有办法把过滤条件和连接条件分开来? 出于这样考虑,Sql1999标准推出来了.有许多人用的还是   
 旧的语法,所以得看懂这种语句.   
   
   
   
 select ename,dname from emp,dept;(旧标准).   
 select ename,dname from emp cross join dept;(1999标准)   
   
 select ename,dname from emp,dept where emp.deptno=dept.deptno (旧)   
 select ename,dname from emp join dept on(emp.deptno = dept.deptno); 1999标准.没有Where语句.   
 select ename,dname from emp join dept using(deptno);等同上句,但不推荐使用.   
   
 select ename,grade from emp e join salgrade s on(e.sal between s.losal and s.hisal);   
 join 连接语句, on过滤条件。连接，条件一眼分开。如果用Where语句较长时，连接语句和过滤语句混在一起。   
   
 三张表连接：   
 slect ename,dname, grade from   
 emp e join dept d on(e.deptno=d.deptno)   
 join salgrade s on(e.sal between s.losal and s.hisal)   
 where ename not like '\_A%';   
 把每张表连接 条件不混在一起，然后数据过滤条件全部区分开来。读起来更清晰，更容易懂一点。   
   
 select e1.ename,e2.ename from emp e1 join emp e2 on(e1.mgr = e2.emptno);   
   
 左外连接：会把左边这张表多余数据显示出来。   
 select e1.ename,e2,ename from emp e1 left join emp e2 on(e1.mgr =e2.empno);left 后可加outer   
 右外连接：   
 select ename,dname from emp e right outer join dept d on(e.deptno =d.deptno); outer可以取掉。   
   
 即把左边多余数据，也把右边多余数据拿出来，全外连接。   
 select ename,dname from emp e full join dept d on(e.deptno =d.deptno);   
   
   
16-23 课：求部门平均薪水的等级   
   
 A.求部门平均薪水的等级。   
   
 select deptno,avg\_sal,grade from   
 (select deptno,avg(sal) avg\_sal from emp group by deptno)t   
 join salgrade s on(t.avg\_sal between s.losal and s.hisal)   
   
 B.求部门平均的薪水等级   
 select deptno,avg(grade) from   
 (select deptno,ename, grade from emp join salgrade s on(emp.sal between s.losal and   
 s.hisal)) t   
 group by deptno   
   
 C.那些人是经理   
 select ename from emp where empno in(select mgr from emp);   
 select ename from emp where empno in(select distinct mgr from emp);   
   
 D.不准用组函数，求薪水的最高值（面试题）   
   
 select distinct sal from emp where sal not in(   
 select distinct e1.sal from emp e1 join emp e2 on (e1.sal<e2.sal));   
###########################################  
内连接（INNER JOIN）  
  
  
内连接（INNER JOIN）：有两种，显式的和隐式的，返回连接表中符合连接条件和查询条件的数据行。（所谓的链接表就是数据库在做查询形成的中间表）。  
  
  
例如：下面的语句3和语句4的结果是相同的。  
语句3：隐式的内连接，没有INNER JOIN，形成的中间表为两个表的笛卡尔积。  
SELECT O.ID,O.ORDER\_NUMBER,C.ID,C.NAME  
FROM CUSTOMERS C,ORDERS O  
WHERE C.ID=O.ID;  
  
  
语句4：显示的内连接，一般称为内连接，有INNER JOIN，形成的中间表为两个表经过ON条件过滤后的笛卡尔积。  
SELECT O.ID,O.ORDER\_NUMBER,C.ID,C.NAME  
FROM CUSTOMERS C INNER JOIN ORDERS O ON C.ID=O.ID;  
语句3和语句4的查询结果相同  
  
  
#####################################################  
   
 E.平均薪水最高的部门编号   
   
 select deptno,avg\_sal from   
 (select avg(sal)avg\_sal,deptno from emp group by deptno)   
 where avg\_sal=   
 (select max(avg\_sal)from   
 (select avg(sal) avg\_sal,deptno from emp group by deptno)   
 )   
   
 F.平均薪水最高的部门名称   
 select dname from dept where deptno=   
 (   
 select deptno from   
 (select avg(sal)avg\_sal,deptno from emp group by deptno)   
 where avg\_sal=   
 (select max(avg\_sal)from   
 (select avg(sal) avg\_sal,deptno from emp group by deptno)   
 )   
 )   
   
 G.求平均薪水的等级最低的部门的部门名称   
   
 组函数嵌套   
 如：平均薪水最高的部门编号，可以E.更简单的方法如下：   
 select deptno,avg\_sal from   
 (select avg(sal) avg\_sal,deptno from emp group by deptno)   
 where avg\_sal =   
 (select max(avg(sal)) from emp group by deptno)   
   
 组函数最多嵌套两层   
   
 分析：   
 首先求   
 1.平均薪水： select avg(sal) from group by deptno;   
   
 2.平均薪水等级： 把平均薪水当做一张表，需要和另外一张表连接salgrade   
 select deptno,grade avg\_sal from   
 ( select deptno,avg(sal) avg\_sal from emp group by deptno) t   
 join salgrade s on(t.avg\_sal between s.losal and s.hisal)   
   
 上面结果又可当成一张表。   
   
 DEPTNO GRADE AVG\_SAL   
 -------- ------- ----------   
 30 3 1566.66667   
 20 4 2175   
 10 4 2916.66667   
   
 3.求上表平均等级最低值   
   
 select min(grade) from   
 (   
 select deptno,grade,avg\_sal from   
 (select deptno,avg(sal) avg\_sal from emp group by deptno)t   
 join salgrade s on(t.avg\_sal between s.losal and s.hisa)   
 )   
   
 4.把最低值对应的2结果的那张表的对应那张表的deptno, 然后把2对应的表和另外一张表做连接。   
   
 select dname ,deptno,grade,avg\_sal from   
 (   
 select deptno,grade,avg\_sal from   
 (select deptno,avg(sal) avg\_sal from emp group by deptno)t   
 join salgrade s on(t.avg\_sal between s.losal and s.hisal)   
 ) t1   
 join dept on (t1.deptno = dept.deptno)   
 where t1.grade =   
 (   
 select deptno,grade,avg\_sal from   
 (select deptno,avg(sal) avg\_sal from emp group by deptno) t   
 join salgrade s on(t.avg\_sal between s.losal and s.hisal)   
 )   
 )   
 结果如下：   
   
 DNAME DEPTNO GRADE AVG\_SAL   
 -------- ------- -------- --------   
 SALES 30 3 1566.6667   
   
   
 H: 视图（视图就是一张表，一个字查询）   
   
 G中语句有重复，可以用视图来简化。   
 conn sys/bjsxt as sysdba;   
 grant create table,create view to scott;   
 conn scott/tiger   
 创建视图：   
 create view v$\_dept\_avg-sal\_info as   
 select deptno,grade,avg\_sal from   
 ( select deptno,avg(sal) avg\_sal from emp group by deptno)t   
 join salgrade s on 9t.avg\_sal between s.losal and s.hisal)   
   
 然后   
 select \* from v$\_dept\_avg-sal\_info   
   
 结果如下：   
 DEPTNO GRADE AVG\_SAL   
 -------- ------- ----------   
 30 3 1566.66667   
 20 4 2175   
 10 4 2916.66667   
   
 然后G中查询可以简化成：   
 select dname,t1.deptno,grade,avg\_sal from   
 v$\_dept\_avg-sal\_info t1   
 join dept on 9t1.deptno =dept.deptno)   
 where t1.grade=   
 (   
 select min(grade) from v$\_dept\_avg-sal\_info t1   
 )   
  
  
  
  
  
  
--desc 表名 描述表的内容   
desc emp;   
   
--加上数学表达式和列名 ""保持格式   
select ename "name space", sal\*12 year\_sal from emp;   
   
select 2\*3 from dual;   
   
select sysdate from dual;   
   
--空值的数学表达式 结果都是空值   
select ename, sal\*12 + comm from emp;   
   
--"||"字符串连接 单引号中的是字符串，字符串中的单引号，''表示   
select ename||sal from emp;   
   
--distinct 修饰两个字段   
select distinct deptno, job from emp;   
   
select ename, sal from emp where sal between 800 and 1500;   
select ename, sal from emp where sal >= 800 and sal <= 1500;   
   
select ename, sal from emp where comm is null;   
   
   
--一个事务开始于一条ddl语句，结束语rollback commit ddl数据定义语句 dcl数据控制语句语句   
--正常断开连接 提交 非正常断开连接 回滚   
   
select ename, sal from emp where sal not in (800 ,1500);   
   
-- 模糊查询   
select ename, sal from emp where ename like '\_A%';   
select ename, sal from emp where ename like '%\%%';   
   
--转义字符   
select ename, sal from emp where ename like '%$%%' escape '$';   
   
--排序   
select \* from dept order by deptno desc;   
select ename, sal from emp order by deptno asc, ename desc;   
   
--函数   
select lower(ename) from emp;   
select substr(ename,1,3) from emp;   
select round(23.652, 1) from dual;   
select to\_char(sal, '$99,999.999') from emp;   
select to\_char(sal, 'L000,000.000') from emp;   
select to\_char(hiredate, 'yyyy-mm-dd hh:mi:ss') from emp;   
select to\_char(sysdate, 'yyyy-mm-dd hh24:mi:ss') from dual;   
select ename, hiredate from emp   
 where hiredate > to\_date('1982-09-11 12:44:44', 'YYYY-MM-DD HH24:MI:SS');   
select sal from emp where sal > to\_number('$1,250.000', '$9,999.000');   
select ename,sal\*12 + nvl(comm, 0) from emp;   
   
--组函数 多行输入，只有一行输出 select ename, max(sal) from emp; 结果不能保证只有一行   
select max(sal) from emp;   
select sum(sal) from emp;   
select count(distinct deptno) from emp;   
   
--分组函数   
select deptno,avg(sal) from emp group by deptno;   
select deptno, job, max(sal) from emp group by deptno, job;   
   
--出现在select后，不是组函数的，必须出现在group by里   
--where语句对单挑记录进行过滤，分组后过滤用having   
   
--子查询 把一次查询结果当成表   
select ename from emp where sal = (select max(sal) from emp);   
select ename, sal from emp   
 join (select max(sal)max\_sal, deptno from emp group) t   
 on(t.max\_sal = emp.sal and t.deptno = emp.deptno);   
   
--自连接   
select t1.ename, t2.ename from emp t1, emp t2 where t1.mgr = t2.empno;   
   
   
select ename, dname, grade from emp e, dept d, salgrade s where e.deptno = d.deptno and   
   
e.sal between   
s.losal and s.hisal and job <> 'CLERK';   
   
--SQL 1999 链接条件不写在where中，where只负责过滤数据   
select ename, dname from emp cross join dept;   
select ename, dname from emp join dept on (emp.deptno = dept.deptno);   
select ename, dname from emp join dept using (deptno)；   
   
select ename, grade from emp e join salgrade s on (e.sal between s.losal and s.hisal);   
select e1.ename, e2.ename from emp e1 join emp e2 on (e1.mgr = e2.empno);   
   
--左外连接   
select e1.ename, e2.ename from emp e1 left join emp e2 on (e1.mgr = e2.empno);   
--右外连接   
select ename, dname from emp e right join dept d on (e.deptno = d.deptno);   
select ename, dname from emp e full outer join dept d on (e.deptno = d.deptno);   
   
--求部门中那些人薪水最高   
select ename, sal from emp   
join (select deptno, max(sal) max\_sal from emp group by deptno) t   
on (emp.sal = t.max\_sal and emp.deptno = t.deptno);   
   
--求部门平均薪水的等级   
select deptno, grade from salgrade s   
join (select avg(sal) avg\_sal, deptno from emp group by deptno) t   
on (avg\_sal between losal and hisal);   
   
--求平均薪水最高的部门的部门标号   
select deptno, avg\_sal from   
(select avg(sal) avg\_sal, deptno from emp group by deptno)   
where avg\_sal =   
(select max(avg(sal)) from emp group by deptno);   
   
--平均薪水的等级最低的部门的部门名称   
select dname, t1.deptno, grade, avg\_sal from   
( select deptno, grade, avg\_sal from   
 (select deptno, avg(sal) avg\_sal from emp group by deptno) t   
 join salgrade s on (t.avg\_sal between s.losal and s.hisal)   
) t1   
join dept on (t1.deptno = dept.deptno)   
where t1.grade =   
(   
 select min(grade) from   
 ( select deptno, grade, avg\_sal from   
 (select deptno, avg(sal) avg\_sal from emp group by deptno) t   
 join salgrade s on (t.avg\_sal between s.losal and s.hisal)   
 )   
)   
   
--建立视图   
create view v$\_dept\_avg\_sal\_info as   
select deptno, grade, avg\_sal from   
 (select deptno, avg(sal) avg\_sal from emp group by deptno) t   
join salgrade s on (t.avg\_sal between s.losal and s.hisal);   
   
--变简单了   
select dname, t1.deptno, grade, avg\_sal from   
v$\_dept\_avg\_sal\_info t1   
join dept on (t1.deptno = dept.deptno)   
where t1.grade =   
(   
 select min(grade) from v$\_dept\_avg\_sal\_info   
)   
   
--求部门平均薪水的等级   
select deptno, avg\_sal, grade from   
 (select avg(sal) avg\_sal, deptno from emp group by deptno)   
join salgrade s   
on (avg\_sal between s.losal and s.hisal);   
   
--求部门平均的薪水等级   
select avg(grade), deptno from   
(   
 select deptno, ename, grade from emp   
 join salgrade s   
 on emp.sal between s.losal and s.hisal   
)   
group by deptno   
   
--雇员中有哪些人是经理人   
select ename from emp where empno in   
(select distinct mgr from emp);   
   
--不使用组函数，求薪水的最高值   
select distinct sal from emp where sal not in   
(select distinct e1.sal from emp e1 join emp e2 on (e1.sal < e2.sal));   
   
--平均薪水最高的部门的部门名称   
select dname from dept where deptno =   
(   
 select deptno from   
 (select avg(sal) avg\_sal, deptno from emp group by deptno)   
 where avg\_sal =   
 (select max(avg\_sal) from   
 (select avg(sal) avg\_sal, deptno from emp group by deptno)   
 )   
)   
   
--求比普通员工的最高薪水还要高的经理人名称   
select ename from emp   
where empno in (select distinct mgr from emp where mgr is not null) and sal >   
(   
 select max(sal) from emp where empno not in   
 (select distinct mgr from emp where mgr is not null)   
)   
   
--rownum只能和小于等于号联合使用   
--使用rownum 要关联表   
select ename from   
(select rownum r, ename from emp) where r > 10;   
   
--薪水最高的5个人   
select ename, sal from   
(select ename, sal from emp order by sal desc)   
where rownum <= 5;   
   
--薪水最高的第六个到第十个雇员   
select ename, sal from   
 (   
 select ename, sal, rownum r from   
 (select ename, sal from emp order by sal desc)   
 )   
where r >= 6 and r <= 10   
   
   
--http://penghao122.javaeye.com/blog/54810   
--SQL面试题   
--找出没选过“黎明”老师的所有学生姓名。   
select sname from s   
join sc on (sc.sno = s.sno)   
join c on (c.cno = s.cno)   
where c.cteacher <> 'liming';   
   
--列出2门以上（含2门）不及格学生姓名及平均成绩。   
select sname where sno in   
 (   
 select sno from sc where scgrade < 60   
 group by sno having count(\*) >= 2   
 )   
   
--即学过1号课程有学过2号课所有学生的姓名   
select sno from sc where cno = 1 and sno in   
(   
 select sno from sc where cno = 2   
);   
   
--一个事务开始于一条ddl语句，结束语rollback commit ddl数据定义语句 dcl数据控制语句语句   
--正常断开连接 提交 非正常断开连接 回滚   
  
  
--desc 表名 描述表的内容   
desc emp;  
  
  
--加上数学表达式和列名 ""保持格式  
select ename "name space", sal\*12 year\_sal from emp;   
  
  
select 2\*3 from dual;  
  
  
select sysdate from dual;  
  
  
--空值的数学表达式 结果都是空值  
select ename, sal\*12 + comm from emp;   
  
  
--"||"字符串连接 单引号中的是字符串，字符串中的单引号，''表示  
select ename||sal from emp;   
  
  
--distinct 修饰两个字段  
select distinct deptno, job from emp;  
  
  
select ename, sal from emp where sal between 800 and 1500;  
select ename, sal from emp where sal >= 800 and sal <= 1500;  
  
  
select ename, sal from emp where comm is null;  
  
  
  
  
--一个事务开始于一条ddl语句，结束语rollback commit ddl数据定义语句 dcl数据控制语句语句  
--正常断开连接 提交 非正常断开连接 回滚  
  
  
select ename, sal from emp where sal not in (800 ,1500);  
  
  
-- 模糊查询  
select ename, sal from emp where ename like '\_A%';  
select ename, sal from emp where ename like '%\%%';  
  
  
--转义字符  
select ename, sal from emp where ename like '%$%%' escape '$';  
  
  
--排序  
select \* from dept order by deptno desc;  
select ename, sal from emp order by deptno asc, ename desc;  
  
  
--函数  
select lower(ename) from emp;  
select substr(ename,1,3) from emp;  
select round(23.652, 1) from dual;  
select to\_char(sal, '$99,999.999') from emp;  
select to\_char(sal, 'L000,000.000') from emp;  
select to\_char(hiredate, 'yyyy-mm-dd hh:mi:ss') from emp;  
select to\_char(sysdate, 'yyyy-mm-dd hh24:mi:ss') from dual;  
select ename, hiredate from emp  
 where hiredate > to\_date('1982-09-11 12:44:44', 'YYYY-MM-DD HH24:MI:SS');  
select sal from emp where sal > to\_number('$1,250.000', '$9,999.000');  
select ename,sal\*12 + nvl(comm, 0) from emp;  
  
  
--组函数 多行输入，只有一行输出 select ename, max(sal) from emp; 结果不能保证只有一行  
select max(sal) from emp;  
select sum(sal) from emp;  
select count(distinct deptno) from emp;  
  
  
--分组函数  
select deptno,avg(sal) from emp group by deptno;  
select deptno, job, max(sal) from emp group by deptno, job;  
  
  
--出现在select后，不是组函数的，必须出现在group by里  
--where语句对单挑记录进行过滤，分组后过滤用having  
  
  
--子查询 把一次查询结果当成表  
select ename from emp where sal = (select max(sal) from emp);  
select ename, sal from emp  
 join (select max(sal)max\_sal, deptno from emp group) t  
 on(t.max\_sal = emp.sal and t.deptno = emp.deptno);  
  
  
--自连接  
select t1.ename, t2.ename from emp t1, emp t2 where t1.mgr = t2.empno;  
  
  
  
  
select ename, dname, grade from emp e, dept d, salgrade s where e.deptno = d.deptno and   
  
  
e.sal between  
s.losal and s.hisal and job <> 'CLERK';  
  
  
--SQL 1999 链接条件不写在where中，where只负责过滤数据  
select ename, dname from emp cross join dept;  
select ename, dname from emp join dept on (emp.deptno = dept.deptno);  
select ename, dname from emp join dept using (deptno)；  
  
  
select ename, grade from emp e join salgrade s on (e.sal between s.losal and s.hisal);  
select e1.ename, e2.ename from emp e1 join emp e2 on (e1.mgr = e2.empno);  
  
  
--左外连接  
select e1.ename, e2.ename from emp e1 left join emp e2 on (e1.mgr = e2.empno);  
--右外连接  
select ename, dname from emp e right join dept d on (e.deptno = d.deptno);  
select ename, dname from emp e full outer join dept d on (e.deptno = d.deptno);  
  
  
--求部门中那些人薪水最高  
select ename, sal from emp   
join (select deptno, max(sal) max\_sal from emp group by deptno) t   
on (emp.sal = t.max\_sal and emp.deptno = t.deptno);  
  
  
--求部门平均薪水的等级  
select deptno, grade from salgrade s   
join (select avg(sal) avg\_sal, deptno from emp group by deptno) t   
on (avg\_sal between losal and hisal);  
  
  
--求平均薪水最高的部门的部门标号  
select deptno, avg\_sal from  
(select avg(sal) avg\_sal, deptno from emp group by deptno)  
where avg\_sal =  
(select max(avg(sal)) from emp group by deptno);  
  
  
--平均薪水的等级最低的部门的部门名称  
select dname, t1.deptno, grade, avg\_sal from  
( select deptno, grade, avg\_sal from  
 (select deptno, avg(sal) avg\_sal from emp group by deptno) t  
 join salgrade s on (t.avg\_sal between s.losal and s.hisal)  
) t1  
join dept on (t1.deptno = dept.deptno)  
where t1.grade =   
(  
 select min(grade) from  
 ( select deptno, grade, avg\_sal from  
 (select deptno, avg(sal) avg\_sal from emp group by deptno) t  
 join salgrade s on (t.avg\_sal between s.losal and s.hisal)  
 )  
)  
  
  
--建立视图  
create view v$\_dept\_avg\_sal\_info as  
select deptno, grade, avg\_sal from  
 (select deptno, avg(sal) avg\_sal from emp group by deptno) t  
join salgrade s on (t.avg\_sal between s.losal and s.hisal);  
  
  
--变简单了  
select dname, t1.deptno, grade, avg\_sal from  
v$\_dept\_avg\_sal\_info t1  
join dept on (t1.deptno = dept.deptno)  
where t1.grade =   
(  
 select min(grade) from v$\_dept\_avg\_sal\_info  
)  
  
  
--求部门平均薪水的等级  
select deptno, avg\_sal, grade from  
 (select avg(sal) avg\_sal, deptno from emp group by deptno)  
join salgrade s  
on (avg\_sal between s.losal and s.hisal);  
  
  
--求部门平均的薪水等级  
select avg(grade), deptno from  
(  
 select deptno, ename, grade from emp  
 join salgrade s  
 on emp.sal between s.losal and s.hisal  
)   
group by deptno  
  
  
--雇员中有哪些人是经理人  
select ename from emp where empno in  
(select distinct mgr from emp);  
  
  
--不使用组函数，求薪水的最高值  
select distinct sal from emp where sal not in  
(select distinct e1.sal from emp e1 join emp e2 on (e1.sal < e2.sal));  
  
  
--平均薪水最高的部门的部门名称  
select dname from dept where deptno =  
(  
 select deptno from  
 (select avg(sal) avg\_sal, deptno from emp group by deptno)  
 where avg\_sal =  
 (select max(avg\_sal) from  
 (select avg(sal) avg\_sal, deptno from emp group by deptno)  
 )  
)  
  
  
--求比普通员工的最高薪水还要高的经理人名称  
select ename from emp   
where empno in (select distinct mgr from emp where mgr is not null) and sal >  
(  
 select max(sal) from emp where empno not in  
 (select distinct mgr from emp where mgr is not null)  
)  
  
  
--rownum只能和小于等于号联合使用  
--使用rownum 要关联表  
select ename from  
(select rownum r, ename from emp) where r > 10;  
  
  
--薪水最高的5个人  
select ename, sal from  
(select ename, sal from emp order by sal desc)  
where rownum <= 5;  
  
  
--薪水最高的第六个到第十个雇员  
select ename, sal from  
 (  
 select ename, sal, rownum r from  
 (select ename, sal from emp order by sal desc)  
 )  
where r >= 6 and r <= 10  
  
  
  
  
--http://penghao122.javaeye.com/blog/54810  
--SQL面试题  
--找出没选过“黎明”老师的所有学生姓名。   
select sname from s   
join sc on (sc.sno = s.sno)  
join c on (c.cno = s.cno)   
where c.cteacher <> 'liming';  
  
  
--列出2门以上（含2门）不及格学生姓名及平均成绩。   
select sname where sno in   
 (  
 select sno from sc where scgrade < 60  
 group by sno having count(\*) >= 2  
 )  
  
  
--即学过1号课程有学过2号课所有学生的姓名  
select sno from sc where cno = 1 and sno in  
(  
 select sno from sc where cno = 2  
);  
  
  
--一个事务开始于一条ddl语句，结束语rollback commit ddl数据定义语句 dcl数据控制语句语句  
--正常断开连接 提交 非正常断开连接 回滚  
  
  
  
  
  
  
  
  
送上mysql中的，oracle默认表(emp,dept, salgrade)的建表语句   
  
  
Sql代码   
CREATE TABLE EMP   
 (EMPNO numeric(4) NOT NULL,   
 ENAME VARCHAR(10),   
 JOB VARCHAR(9),   
 MGR numeric(4),   
 HIREDATE DATE,   
 SAL numeric(7, 2),   
 COMM numeric(7, 2),   
 DEPTNO numeric(2));   
   
INSERT INTO EMP VALUES   
 (7369, 'SMITH', 'CLERK', 7902,   
 '1980-12-17', 800, NULL, 20);   
INSERT INTO EMP VALUES   
 (7499, 'ALLEN', 'SALESMAN', 7698,   
 '1981-11-20', 1600, 300, 30);   
INSERT INTO EMP VALUES   
 (7521, 'WARD', 'SALESMAN', 7698,   
 '1981-02-22', 1250, 500, 30);   
INSERT INTO EMP VALUES   
 (7566, 'JONES', 'MANAGER', 7839,   
 '1981-04-02', 2975, NULL, 20);   
INSERT INTO EMP VALUES   
 (7654, 'MARTIN', 'SALESMAN', 7698,   
 '1981-09-28', 1250, 1400, 30);   
INSERT INTO EMP VALUES   
 (7698, 'BLAKE', 'MANAGER', 7839,   
 '1981-05-01', 2850, NULL, 30);   
INSERT INTO EMP VALUES   
 (7782, 'CLARK', 'MANAGER', 7839,   
 '1981-01-09', 2450, NULL, 10);   
INSERT INTO EMP VALUES   
 (7788, 'SCOTT', 'ANALYST', 7566,   
 '1982-12-09', 3000, NULL, 20);   
INSERT INTO EMP VALUES   
 (7839, 'KING', 'PRESIDENT', NULL,   
 '1981-11-07', 5000, NULL, 10);   
INSERT INTO EMP VALUES   
 (7844, 'TURNER', 'SALESMAN', 7698,   
 '1981-09-08', 1500, 0, 30);   
INSERT INTO EMP VALUES   
 (7876, 'ADAMS', 'CLERK', 7788,   
 '1983-01-12', 1100, NULL, 20);   
INSERT INTO EMP VALUES   
 (7900, 'JAMES', 'CLERK', 7698,   
 '1981-11-03', 950, NULL, 30);   
INSERT INTO EMP VALUES   
 (7902, 'FORD', 'ANALYST', 7566,   
 '1981-12-03', 3000, NULL, 20);   
INSERT INTO EMP VALUES   
 (7934, 'MILLER', 'CLERK', 7782,   
 '1982-01-12', 1300, NULL, 10);   
   
   
CREATE TABLE DEPT   
 (DEPTNO numeric(2),   
 DNAME VARCHAR(14),   
 LOC VARCHAR(13) );   
   
INSERT INTO DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK');   
INSERT INTO DEPT VALUES (20, 'RESEARCH', 'DALLAS');   
INSERT INTO DEPT VALUES (30, 'SALES', 'CHICAGO');   
INSERT INTO DEPT VALUES (40, 'OPERATIONS', 'BOSTON');   
   
CREATE TABLE BONUS   
 (ENAME VARCHAR(10),   
 JOB VARCHAR(9),   
 SAL numeric,   
 COMM numeric);   
   
CREATE TABLE SALGRADE   
 (GRADE numeric,   
 LOSAL numeric,   
 HISAL numeric);   
   
INSERT INTO SALGRADE VALUES (1, 700, 1200);   
INSERT INTO SALGRADE VALUES (2, 1201, 1400);   
INSERT INTO SALGRADE VALUES (3, 1401, 2000);   
INSERT INTO SALGRADE VALUES (4, 2001, 3000);   
INSERT INTO SALGRADE VALUES (5, 3001, 9999);   
   
CREATE TABLE DUMMY   
 (DUMMY numeric);   
   
INSERT INTO DUMMY VALUES (0);   
  
  
给个通俗的解释吧.   
例表a   
aid adate   
1 a1   
2 a2   
3 a3   
表b   
bid bdate   
1 b1   
2 b2   
4 b4   
两个表a,b相连接,要取出id相同的字段   
select \* from a ,b where a.aid = b.bid这是仅取出匹配的数据.   
此时的取出的是:   
1 a1 b1   
2 a2 b2   
那么left join 指:   
select \* from a left join b on a.aid = b.bid   
首先取出a表中所有数据,然后再加上与a,b匹配的的数据   
此时的取出的是:   
1 a1 b1   
2 a2 b2   
3 a3 空字符   
同样的也有right join   
指的是首先取出b表中所有数据,然后再加上与a,b匹配的的数据   
此时的取出的是:   
1 a1 b1   
2 a2 b2   
4 空字符 b4  
inner join是内联查询语法是select T1.XX,T1.XX,T2.XX,T2.XX from T1 inner join T2 on T1.X=T2.X这是两表内联起来表连接有四种连接法内联:inner join 筛选两表都存在的数据全联:full join 在两表中最多行数的那张表为主表,筛选主表数据,从表没有数据时以NULL代替左联:left join 左表为主表,跟FULL JOIN差不多右联:right join 右表为主表交叉联连:cross join 不带条件.表1的每一条纪录都与表2的每一条记录连接产生新表如果表1有6条记录,表2有4条记录,那么不带条件的表连接装产生6\*4条记录当我没有说话,我说不清楚啦.  
sql多表连接查询inner join, left join , right join ,full join ,cross join2010-01-08 11:43inner join,full outer join,left join,right jion  
内部连接 inner join 两表都满足的组合  
full outer 全连 两表相同的组合在一起，A表有，B表没有的数据（显示为null）,同样B表有  
A表没有的显示为(null)  
A表 left join B表 左连,以A表为基础，A表的全部数据，B表有的组合。没有的为null  
A表 right join B表 右连,以B表为基础，B表的全部数据，A表的有的组合。没有的为null  
  
  
  
  
查询分析器中执行：  
--建表table1,table2：  
create table table1(id int,name varchar(10))  
create table table2(id int,score int)  
insert into table1 select 1,'lee'  
insert into table1 select 2,'zhang'  
insert into table1 select 4,'wang'  
insert into table2 select 1,90  
insert into table2 select 2,100  
insert into table2 select 3,70  
如表  
-------------------------------------------------  
table1|table2|  
-------------------------------------------------  
idname|idscore|  
1lee|190|  
2zhang|2100|  
4wang|370|  
-------------------------------------------------  
  
  
以下均在查询分析器中执行  
  
  
一、外连接  
1.概念：包括左向外联接、右向外联接或完整外部联接  
  
  
2.左连接：left join 或 left outer join  
(1)左向外联接的结果集包括 LEFT OUTER 子句中指定的左表的所有行，而不仅仅是联接列所匹配的行。如果左表的某行在右表中没有匹配行，则在相关联的结果集行中右表的所有选择列表列均为空值(null)。  
(2)sql语句  
select \* from table1 left join table2 on table1.id=table2.id  
-------------结果-------------  
idnameidscore  
------------------------------  
1lee190  
2zhang2100  
4wangNULLNULL  
------------------------------  
注释：包含table1的所有子句，根据指定条件返回table2相应的字段，不符合的以null显示  
  
  
3.右连接：right join 或 right outer join  
(1)右向外联接是左向外联接的反向联接。将返回右表的所有行。如果右表的某行在左表中没有匹配行，则将为左表返回空值。  
(2)sql语句  
select \* from table1 right join table2 on table1.id=table2.id  
-------------结果-------------  
idnameidscore  
------------------------------  
1lee190  
2zhang2100  
NULLNULL370  
------------------------------  
注释：包含table2的所有子句，根据指定条件返回table1相应的字段，不符合的以null显示  
  
  
4.完整外部联接:full join 或 full outer join   
(1)完整外部联接返回左表和右表中的所有行。当某行在另一个表中没有匹配行时，则另一个表的选择列表列包含空值。如果表之间有匹配行，则整个结果集行包含基表的数据值。  
(2)sql语句  
select \* from table1 full join table2 on table1.id=table2.id  
-------------结果-------------  
idnameidscore  
------------------------------  
1lee190  
2zhang2100  
4wangNULLNULL  
NULLNULL370  
------------------------------  
注释：返回左右连接的和（见上左、右连接）  
  
  
二、内连接  
1.概念：内联接是用比较运算符比较要联接列的值的联接  
  
  
2.内连接：join 或 inner join   
  
  
3.sql语句  
select \* from table1 join table2 on table1.id=table2.id  
-------------结果-------------  
idnameidscore  
------------------------------  
1lee190  
2zhang2100  
------------------------------  
注释：只返回符合条件的table1和table2的列  
  
  
4.等价（与下列执行效果相同）  
A:select a.\*,b.\* from table1 a,table2 b where a.id=b.id  
B:select \* from table1 cross join table2 where table1.id=table2.id (注：cross join后加条件只能用where,不能用on)  
  
  
三、交叉连接(完全)  
  
  
1.概念：没有 WHERE 子句的交叉联接将产生联接所涉及的表的笛卡尔积。第一个表的行数乘以第二个表的行数等于笛卡尔积结果集的大小。（table1和table2交叉连接产生3\*3=9条记录）  
  
  
2.交叉连接：cross join (不带条件where...)  
  
  
3.sql语句  
select \* from table1 cross join table2  
-------------结果-------------  
idnameidscore  
------------------------------  
1lee190  
2zhang190  
4wang190  
1lee2100  
2zhang2100  
4wang2100  
1lee370  
2zhang370  
4wang370  
------------------------------  
注释：返回3\*3=9条记录，即笛卡尔积  
  
  
4.等价（与下列执行效果相同）  
A:select \* from table1,table2