**数据库笔记**

# 基本概念

关系数据库

行称为记录（或元组），列称为字段

表对应一个类，记录对应一个对象，字段对应属性

OracleServiceMYORACLE 主服务，需启动

OracleOraDb11g\_home1TNSListener 网络监听器

OracleOraDb11g\_home1ClrAgent

OracleMTSRecoveryService

OracleJobSchedulerMYORACLE

OracleDBConsolemyoracle

OracleVssWriterMYORACLE

信息改变了需要改配置文件

E:\Oracle\product\11.2.0\dbhome\_1\install\portlist.ini

E:\Oracle\product\11.2.0\dbhome\_1\NETWORK\ADMIN\listener.ora

E:\Oracle\product\11.2.0\dbhome\_1\NETWORK\ADMIN\tnsnames.ora

访问数据库：

应用程序开发—>sqlplus.exe

运行—>sqlplusw set linesize=200 set pagesize=20

用户名：scott 密码：tiger

用第三方的：new(file 下面) 🡪sqlwindow

select \* from tab; 列出当前用户的数据库表

desc emp; 查看表的组成

select \* from emp; 查看表内容

set linesize 200 set pagesize 20

# SQL 语言

查询语言 即select语句

DML语句 数据库操作语言　包含insert update delete

DDL语句　数据库定义语言　create　alter drop

事务控制语句 包含commit Rollback等

DCL语句 用于权限的分配与回收

Number（7，2）表示小数，整数部分长度为 7-2=5 小数部分长度为2

练习：

1. 员工的所有信息

select \* from emp;

1. 每个人的部门编号，姓名，薪水

select empno,emame,job from emp;

1. 每个人的年薪

select empno,ename,sal\*12 from emp;

1. 当null值参与运算，则结果也为null

select empno,ename,sal\*12+comm from emp;

1. 计算2\*3的值

select 2\*3 from emp(会出现多条重复记录)

1. dual 万能表，只一条记录

计算2\*3的值

select 2\*3 from dual;

1. 显示当前时间

select sysdate from dual;

1. 给列起别名，比如求每个人的年薪

select sal\*12 year\_sal from emp;

1. 如果别名中有空格，需要加双引号

select sal\*12 “年 薪” from emp;

1. 拼接字符串 用||

select ‘a’||‘bc’ from dual;

select ename||’-sal-’||sal description from emp;

1. 如果字符串中有单引号，需要用另外一个单引号转义

select ename||’-sa’’l-’||sal description from emp;

1. 去重用distinct （按行去重）

select distinct deptno from emp;

1. 可以是数值类型的等值判断

select \* from emp where deptno=10;

1. 可以是字符串类型的等值判断

select \* from emp where ename=’KING’ 全字段匹配，区分大小写

1. 字符作为不等值判断

select \* from emp where ename > ‘CBA’

1. 求部门不是10的员工信息 ( <>表示！=)

select \* from emp where deptno <> 10

1. 求薪水在800到1500之间的员工信息

select \* from emp where sal>=800 and sal<=1500

select \* from emp where sal between 800 and 1500

1. where……in 的用法

select \* from emp where sal=800 or sal=1500 or sal=2000

select \* from emp where sal in (800,1500,2000) 在集合内

select \* from emp where sal not in (800,1500,2000)不在某个集合内

1. 求入职时间在20-2月-81之后的员工信息

select \* from emp where hiredate>’20-2月-81’;

select \* from emp where sal>1000 or deptno<>10;

1. 求名字中包含ALL这三个字符的员工信息(%表示通配多个字符，\_表示通配单个字符)

select \* from emp where ename like ‘%ALL%’

1. 求名字中的第二个字符为A的员工

select \* from emp where ename like ‘\_A%’

1. 求员工名字中包含特殊字符%的员工信息

select \* from emp where ename like ‘%\%%’ escape ‘\’

escape ‘\’定义\为转义字符，转移字符可以任意定义

1. 求没有年终奖的员工

select \* from emp where comm is null;

1. 求有年终奖的员工

select \* from emp where comm is not null;

not in 后面不能有null值，否则结果什么也没有

1. 员工信息按照正序排列(正序是asc，默认是正序)

select \* from emp order by ename;

1. 员工信息按照倒序排列(倒序是desc)

select \* from emp order by deptno,ename desc;

# 函数

1. Lower函数 将字符串全部转化为小写

select lower(ename) from emp;

1. upper函数 将字符串全部转化为大写

select upper(ename) from emp;

1. substr 求一个字符串的子串

select substr (‘Hello’,1,3) from dual;第一个参数表示起始位置，下标从1开始，第二个参数表示取得的子串的长度

1. chr 求某一个ASCII码值所对应的字符

select chr(65) from dual;

1. Ascii 求一个字符的ASCII码数值

select ascii(‘A’) from dual;

1. 四舍五入

select round(123.8764,2) from dual; 2 表示保留小数点后两位

select round(123.8764,0) from dual; 表示保留到整数，是默认值

select round(123.8764,-1) from dual; -1 表示对小数点前面一位进行四舍五入

1. to\_char 把数字转化成字符串

select to\_char(167.89,’999.999’) from dual;

9表示一个占位符，对小数点后面的空位也强制显示，小数点前面的不强制显示

，表示千位分隔符

$表示美元符号

L表示人民币符号

0表示一个占位符，对小数点前、后的空位都强制显示

1. to\_char 把数字转化成日期格式

select to\_char(sysdate,’YYYY/MM/DD HH24:MI:SS’) from dual;

1. to\_date将字符串转化成日期

select to\_date(‘1993-10-10’,’YYYY-MM-DD’) from dual;

不显示时分秒的原因是数据库默认的时间格式是：日-月-年

用to\_date 把字符串转化成时间格式，直接输出date类型会默认舍去 时分秒，即输出日-月-年格式，要想查看完整的结果需要把时间转化成字符串输出：

select to\_char( to\_date(‘1992-02-13 12:10:18’,

’YYYY/MM/DD HH24:MI:SS’),’YYYY/MM/DD HH24:MI:SS’)from dual;

1. to\_number把字符串转化成数字

select to\_number(‘Y1,222.333’,’L9,999.999’) from dual

出现输出格式与模版不对应的问题与上一个时间处理办法一样

1. nvl用来处理空值，这个函数有两个参数，第一个参数为字段名或表达式，如果这个参数的值为null，那么返回第二个参数的值，否则返回第一个参数的值

select sal\*12+nvl(comm,0) from emp;

1. 组函数，就是处理一组数据的函数，不能与普通字段 (group by 后出现的字段除外) 一块使用

avg 求平均值

max 求最大值

min 求最小值

sum 求总和

count 求总记录数

1. 求所有人的薪水总和，平均值，最大值，最小值，总记录数

select sum(sal),avg(sal),max(sal),min(sal),count(\*) from emp;

1. 求总行数(可以指定具体的字段)但是字段有null值的时候要小心使用

select count(\*) from emp;

select count(empno) from emp;

select count(nvl(comm,1)) from emp;

也可以过滤掉重复的行之后统计行数

select count(distinct deptno) from emp;

select max(empno) from emp where empno>9000;会产生一行，值为null

1. 可以按照某个字段进行分组，分部门统计最高薪水

有Group by 时只能用分组函数和分组条件

select deptno,max(sal) from emp group by deptno;

1. 也可以按照多个字段来分组统计，比如：分部门和岗位，统计最高薪水和行数

select deptno,job,max(sal),count(\*) from emp group by deptno,job;

1. having 仅用于group by 的过滤

select deptno,max(sal) from emp group by deptno having max(sal) >=3000;

1. 求薪水最高的员工姓名

select ename from emp where sal=(select max(sal) from emp);

1. 下面语句执行顺序：from emp 🡪group by deptno 🡪having avg(sal)>2000(至此形成虚表)🡪 deptno,avg(sal) avg\_sal(拿出字段，并赋予别名)🡪 order by avg\_sal desc(最后是按降序排序)

select deptno,avg(sal) avg\_sal from emp group by detpno having avg(sal)>2000 order by avg\_sal desc;

执行顺序：

首先执行where语句将原有记录过滤；

第二执行group by 进行分组；

第三执行having过滤分组；

第四将select中的字段值选出来；

最后执行order by 进行排序；

1. 练习：

按照部门分组统计，求最高薪水，平均薪水，只有薪水是1200以上的才参与统计，并且分组结果中只包含平均薪水在1500以上的部门，而且按照平均薪水倒叙排列

select deptno,max(sal),avg(sal) avg\_sal from emp where

sal>=1200 group by deptno having avg(sal)>=1500

order by avg(sal) desc;

把雇员按照部门分组，求最高薪水、部门号，过滤掉名字中第二个字母是‘A’的，要求分组后的平均薪水大于1500，按照部门编号倒叙排序

select deptno,max(sal) from emp where ename not like ‘\_A%’

group by deptno having avg(sal)>1500 order by deptno desc;

分组统计每个年度入职的人数

select to\_char(hiredate,’YYYY’),count(\*) from emp

group by to\_char(hiredate,’YYYY’);

如何确定出表中完全重复的行

select empno,ename,job,mgr,hiredate,sal,comm,deptno from emp2 group by empno,ename,job,mgr,hiredate,sal,comm,deptno having count(\*)>1;

create table emp2 as select \* from emp;

insert into emp2 select \* from emp where deptno=10;

# 多表连接

92语法

1. 等值连接，求员工姓名及员工所在部门的名字同时显示出来

select ename,dname from emp,dept

where emp.deptno=dept.deptno;

1. 非等值连接:要求每位雇员的薪水等级

起了别名只能用别名，原来的名字就不能用了

select a.empno,a.ename,a.sal,b.grade from emp a,salgrade b where a.sal between b.losal and b.hisal;

跨3个表：求工作职位是president 的雇员的姓名，部门名称和薪水等级

select \* from emp a ,dept b ,salgrade c where a.deptno=b.deptno and a.sal between c.losal and c.hisal and job=’PRESIDENT’;

表与表自身关联，求每个员工的姓名及其上级经理的姓名

select a.empno,a.ename,b.empno topno,b.ename topname from emp a,emp b where a.mgr=b.empno;

99语法

select \* from emp join dept on emp.deptno=dept.deptno;

92 和 99的区别：92语法把表连接的纽带和过滤条件混在一起了，而99语法把连接纽带和过滤条件分开了

select \* from emp,dept where emp.deptno=dept.deptno

and ……;

select \* from emp join dept on emp.deptno=dept.deptno

where ……;

1. 外连接——取出表中连接不到一起的多余数据

左外连接：(能取出左表中关联不到的多余数据)

(92语法) select e1.ename,e2.ename from emp e1,emp e2 where e1.mgr=e2.empno(+);

(99语法)select e1.ename,e2.ename from emp e1 left join emp e2 on e1.mgr=e2.empno;

右外连接：(能取出右表中关联不到的多余数据)

(92语法) select e1.ename,e2.ename from emp e1,emp e2 where e1.mgr(+)=e2.empno;

(99语法)select e1.ename,e2.ename from emp e1 right join emp e2 on e1.mgr=e2.empno;

全外连接：

92里没有全外连接

select e1.ename,e2.ename from emp e1 full join emp e2 on e1.mgr=e2.empno;

1. 左外连接和右外连接的区别

两个表：

A(id,name)

数据：(1,张三)(2,李四)(3,王五)

B(id,name)

数据：(1,学生)(2,老师)(4,校长)

左连接结果：

select A.\*,B.\* from A left join B on A.id=B.id;

1 张三 1 学生

2 李四 2 老师

3 王五 NULL NULL

右链接结果：

select A.\*,B.\* from A right join B on A.id=B.id;

1 张三 1 学生

2 李四 2 老师

NULL NULL 4 校长

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

补充：下面这种情况就会用到外连接

比如有两个表一个是用户表，一个是交易记录表，如果我要查询每个用户的交易记录就要用到左外外连接，因为不是每个用户都有交易记录。

用到左外连接后，有交易记录的信息就会显示，没有的就显示NULL，就像上面我举得例子一样。

如果不用外连接的话，比如【王五】没有交易记录的话，那么用户表里的【王五】的信息就不会显示，就失去了查询所有用户交易记录的意义了。

# 子查询

把子查询当作一张表来看待

1. 子查询结果作为查询条件

=只能建立在单行子查询结果连接

select \* from emp where sal =(select max(sal) from emp);

in可以建立在多行子查询结果连接

select \* from emp where sal in (select sal from emp);

1. 子查询结果作为查询表

in🡨🡪not in like🡨🡪not like is null🡨🡪is not null

in 和 between 的区别：in 相当于列举，字段与in列举的一一匹配，而between是表示范围，大于什么小于什么

求部门平均薪水等级

select t1.deptno,t1.avg\_sal,t2.grade from

(select deptno,avg(sal) avg\_sal from emp group by deptno)t1

join salgrade t2 on t1.avg\_sal between t2.local and t2.hisal;

求每个部门最高薪水的人员名称

select t.deptno,t.max\_sal,emp.ename,emp.empno from emp

join (select deptno,max(sal) max\_sal from emp group by deptno)t

on emp.deptno=t.deptno and emp.sal=t.max\_sal;

求哪些人的薪水在部门的平均薪水之上

select emp.deptno,emp.ename,emp.sal,t.avg\_sal from emp

join (select deptno,avg(sal) avg\_sal from emp group by deptno)t

on emp. deptno =t. deptno and emp.sal>avg\_sal;

# 伪字段rownum

用来标识每条记录的行号，行号从1开始，每次递增1，比较的时候从行号为1开始，如果第一条都没比较成功那么之后的都不会成功

<=0,=0,<0,<1,=2,=3……,>1,>2……,>=2…….都不可以

select emp.\*,rownum from emp where rownum<=5 ;

1. 求薪水最高的前五个人

select \* from (select emp.\* from emp order by sal desc) where rownum<=5;

rownum是整个查询结果形成时才形成的，在形成虚表期间不能用rownum;

1. 取薪水排名的6-10名

select \* from (select t.\*,rownum r from (select \* from emp order by sal desc )t where rownum <=10) where r>=6;

1. 求平均薪水最高的部门的部门编号

第一种解决方案：

select deptno,avg\_sal from (select deptno,avg(sal) avg\_sal from emp group by deptno) where avg\_sal=(select max(avg\_sal) from (select avg(sal) avg\_sal from emp group by deptno))

第二种解决方案：

select deptno,avg(sal) from emp group by deptno

having avg(sal)=(select max(avg(sal)) from emp group by deptno);

1. 求部门经理人中平均薪水最低的部门名称

select t1.\*,d.dname from (select deptno,avg(sal) from emp where empo in (select mgr from emp) group by deptno having avg(sal) = (select min(avg\_sal)) from emp where empno in (select mgr from emp) group by deptno)) t1,dept d where t1.deptno=d.deptno;

1. 求比普通员工的最高薪水还高的经理人名称

select sal from emp where empno in (select distinct mgr from emp) and sal>=( select max(sal) from (select \* from emp where empno not in (select distinct mgr from emp where mgr is not null)));

select ename,hiredate from (select \* from emp order by hiredate desc) where rownum<=5;

SQL从后往前执行，但是Oracle自身优化，所以下面两句效率一样

select \* from emp where deptno =10 and ename like “%A%”;

select \* from emp where ename like “%A%” and deptno =10;

# union 和minus

可以用来实现结果集的合并和去除（可以理解为加和减）操作的两个结果集字段结构应该相同，会自动去重，相同的结果会去重，避免去重可以增加区别字段

minus分页性能比rownum 差

select \* from emp where deptno=20

union select \* from emp where deptno=30;

# Insert语句

insert into emp2

(empno,ename,job,deptno)values(9990,’test’,’clerk’,20);

没有指定的字段是空值，插入空值时，可以显示的用NULL表示

当字段比较少时，可以用省略字段列表的插入方式，

insert into emp2 values(9990,’test’,’clerk’,20);此时字段必须一一对应

应用子查询

insert into emp2 select \* from emp where empo=7369;

insert into emp2 select 9990,’test’,’clerk’,20 from dual;

# update

update emp2 set sal=sal\*1.1;

update emp2 set sal=sal+50 where empno not in(select distinct mgr from emp where mgr is not null);

update emp2 set ename=’ff’,job=’e’ where empno=7777;

表间更新

update emp2 set sal=(select emp.sal from emp where

emp.empno=emp2.empno)

# delete

delete from emp2 where empno=7777;

truncate table emp2;物理上删除，而delete是在逻辑上删除，可以恢复

# rowid

数据库自动维护的每行一个标识，与rownum不同，rownum是一个伪列，而rowid 是真是存在的

利用rowid删除重复的行

delete from emp2 where rowid in (select min(rowid) from emp2 group by empno,ename,job,mgr,hiredate,sal,comm,deptno having count(\*)>1);

# transaction

把多个数据库操作封装到一个事务中，要么全部成功，要么全部失败

原子性：一个transaction是一个程序片段，在程序中如果出任何差错，则全部回退

一致性:回退前后要一致

隔离性：B事务一定要等A事务提交后才能进行

持久性：一旦数据库发生变化，则变化保持持久

transaction起始于DML（update delete insert）语句结束于：

commit 或rollback

执行DDL或DCL语句时事务自动提交

用户正常断开连接，自动提交

系统崩溃或断电事务自动回退

避免死锁，加版本字段

# 表

创建表：create table table\_name(字段名，类型……)

常见数据类型：匹配效率number>char>varchar

varchar2(n) 变长字符串，存储空间等于实际空间的数据大小，最大为4K

char(n)定长字符串，存储空间大小固定

number(p,s) 整数或小数

date 年月日时分秒

long 变长字符串，最大字节数达到2G

clob 大文本字段，最大存储4G

blob 存储图像视频等二进制数据，最大存储4G

设置默认值grade number(2)default 1

# 约束

1. not null 非空

id char(10) not null,

1. unique 唯一

id char(10) unique

唯一约束检查的非null的值是否唯一，对于null的值不检查

1. 主键：非空、唯一

id char(10) primary key

1. 联合主键：

create table t\_student\_info (

id char(10),

name varchar2(30) not null,

sex char(2),

//定义联合主键

constraint pk\_id\_name primary key (id,name)

)

定义在字段后面的约束是字段约束，不用自己写名字，定义在表最后的是表级约束，需要定义名字

1. 外键约束：子表外键字段的值必须参照父表的主键，父表的字段被参照时，该记录不能删除

定义外键约束，

字段约束：class\_id char(2) references t\_class(id)

表级约束：constraint fk\_class\_id foreign key (class\_id) references t\_class(id)

用create table emp2 as (select \* from emp);只复制表结构和数据，约束关系会丢失

1. check约束用于检验字段的值是否符合某个条件表达式

age number(2) check(age>0 and age<80)

constraint age\_min check(age>0)

# 修改表结构及约束

1. 增加字段

alter table stu add(addr varchar2(100))

1. 删除字段

alter table stu drop(addr)

1. 修改字段

alter table stu modify(addr varchar2(20))所修改的类型必须能够容纳原有的数据

1. 删除表

drop table table\_name

1. 添加删除外键：

alter table t\_stu add constraint stu\_class\_fk foreign key (class\_id) references t\_class(id);

alter table t\_stu drop constraint stu\_class\_fk;

1. 非空约束不能用类似的语法，只能通过修改字段来同时修改约束

将字段上的非空约束去除

alter table t\_stu modify (name varchar2(20) null)

在字段上添加非空约束

alter table t\_stu modify (name varchar2(20) not null)

# 索引

索引是为了加快对数据搜索速度，增加对表中某个字段或某几个字段的查询速度，相当于按照某种顺序排序，所以会增加查询速度，但是会增加添加，删除操作的开销

create index inx\_stu\_email on t\_stu(email)

对两个字段建立组合索引，只有组合查询的时候才提高效率，单独查询不会提高效率

create index idx\_stu\_email\_class on stu(email,class)

主键和唯一也会自动创建索引

删除索引

drop index index\_name

# 视图

create view v\_emp as select emp.\*,dname,loc from emp join dept on emp.deptno=dept.deptno

把以上的查询语句形成一个对象，每次执行select \* from v\_emp，直接从虚表拿到查询结果，并不是真正形成一个表，每次用才形成虚表

# 序列

序列对象自动维护着一组序列，可以用这个序列对象作为表的唯一标识

创建一个序列对象

create sequence seq\_id start with 1 increment by 1

应用序列对象：

insert into t\_topic(id,title) values (seq\_id.nextval , ’a’);

drop sequence seq\_id

# 数据库设计三范式

第一范式：

每个表要有主键，列不可分

第二范式：

多对多映射，根据实体对象拆分成多个表，每个实体对象一个表，然后在把关系单独形成一张表。即建立三张表，两个实体各对应一张表，二者关系形成一张表

第三范式：

把存在传递依赖关系的拆分成多个表，在维护一对多的关系时把关系放在多的一端

iteye

# PL/SQL程序开发

是Oracle下的脚本开发语言，用于解决一些复杂的问题，如循环，分支等，把多条SQL查询语句封装成一条PL/SQL语句

1. 匿名块：匿名块的组成部分

declare——可选：声明各种变量、游标

begin——必要：开始实行语句

--表示单行注释

/\* \*/表示多行注释

exception 出错后的处理

end；——必要：结束（后面有分号）

黑窗口执行：

set serveroutput on;

begin

dbms\_output.put\_line(‘HelloWorld’);//输出

end;

/

在PL/SQL developer中执行不需要

set serveroutput on;

/

例子：

declare

v\_ename varchar2(5);

v\_sal

begin

select ename，sal into v\_ename from emp where empno=&aa;

dbms\_output.put\_line(v\_ename);

exception

when no\_data\_found then

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

end;

1. 变量声明语法：

identifier [constant] datatype [not null] [:= default expr];

规则：

变量名不能是保留字

第一个字符必须是字母

变量名最多包含30个字符

不能与数据库的表或者列同名 v\_name

每一行只能声明一个变量

变量类型还有：

binary\_integer:整数，用来计数

boolean:可以取true false null

举例：

v\_count binary\_integer :=0

v\_pi constant number(3,2) := 3.14

v\_name varchar2(20) not null :=’zhangsan’

boolean 类型的不能输出

例子：

declare

v\_name varchar2(20) not null :=’zhangsan’;

begin

dbms\_output.put\_line(‘v\_temp value:’ || v\_name);

end;

1. 变量声明使用%type属性

declare

v\_empno2 emp.empno%type //定义变量与emp.empno类型相同

v\_empno3 v\_empno2%type //好处是表定义变了，这里跟着变

v\_valid :=(v\_sal = v\_sal2) //v\_sal = v\_sal2结果为boolean类型（只有一个等号）

1. table 变量类型，相当于Java里的数组

declare

type type\_table\_emp\_empno is table of emp.empno%type index by binary\_integer;

v\_empnos type\_table\_emp\_empno; //定义数组类型的变量

begin

v\_empnos(0) :=352;

v\_empnos(2) :=343;

v\_empnos(-1) :=367; //下标可以去负值

end;

1. record变量类型，相当于Java里的类

declare

type type\_record\_dept is record (

deptno dept.deptno%type

);

v\_temp type\_record\_dept;

begin

v\_temp.deptno :=50;

dbms\_output.put\_line(v\_temp.deptno);

end;

1. 使用%rowtype声明record变量，不用自己定义了，直接引用已存在的表结构

表结构变了程序不用变

declare

v\_temp dept%rowtype;

begin

v\_temp.deptno :=50;

dbms\_output.put\_line(v\_temp.deptno);

end;

1. 使用select查询给定义的变量赋值

declare

v\_ename emp.ename%type;

v\_sal emp.sal%type;

begin

select ename,sal into v\_ename,v\_sal from emp where empno =7369; //查询结果有且仅有一条

dbms\_output.put\_line(v\_ename || v\_sal);

end;

1. v\_temp只能存储一条记录

declare

v\_temp emp%rowtype;

begin

select \* into v\_emp from emp where empno = 7369;

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

end;

1. 使用insert语句

create table dept2 as select \* from dept;

declare

v\_deptno dept.deptno%type :=50;

v\_dname dept.dname%type :=’aaa’;

v\_loc dept.loc%type :=’bj’;

begin

insert into dept2 values (v\_deptno,v\_name,v\_loc);

commit;

end;

1. 用sql%rowcount记录SQL语句影响的记录数

declare

v\_deptno emp2.deptno%type :=30;

begin

update emp2 set sal=sal/2 where deptno = v\_deptno;

dbms\_output.put\_line(sql%rowcount);

end;

1. 使用DDL语句，必须写excute immediate

begin

excute immediate ‘create table T (name varchar2(20) default ‘’zhangsan’’ ) ’;

end;

1. if语句 if elsif后面then else后面有个end if; 有分号

declare

v\_sal emp.sal%type;

begin

select sal into v\_sal from emp where empno=7902;

if(v\_sal< 1200) then dbms\_output.put\_line(‘low’);

elsif (v\_sal <2000) then dbms\_output.put\_line(‘middle’);

else dbms\_output.put\_line(‘high’);

end if;

end;

1. 循环

第一种：与do ……while类似

declare

i binary\_integer :=1;

begin

loop

dbms\_output.put\_line(i);

i := i+1;

exit when (i>=11); //跳出循环的条件

end loop;

end;

第二种：类似于while循环

declare

i binary\_integer :=1;

begin

while(j< 11) loop

dbms\_output.put\_line(i);

i := i+1;

end loop;

end;

第三种:类似于增强的for循环

begin

for k in 1..10 loop //k不需要定义，正序

dbms\_output.put\_line(k);

end loop;

for k in reverse 1..10 loop //倒序

dbms\_output.put\_line(k);

end loop;

1. 异常处理

declare

v\_temp number(4);

begin

select empno into v\_temp from emp where deptno = 10;

exception

when too\_many\_rows then

dbms\_output.put\_line(‘记录太多’);

when others then

dbms\_output.put\_line(‘其他错误’);

end;

1. 游标

游标就是在某个结果集上的指针，通过这个指针的移动，我们可以遍历整个结果集

1. 使用步骤

声明游标

打开游标

处理游标中的数据

关闭游标

1. 例1：

declare

cursor c is select \* from emp;

v\_emp c%rowtype;

begin

open c;

fetch c into v\_emp;

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

fetch c into v\_emp;

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

close c;

end;

1. 游标的属性：

%isopen,boolean类型变量，判断游标是否打开

%notfound,boolean类型，如果最近没有fetch到记录取true

%found,boolean类型，如果最近fetch到记录则为true

%rowcount, number类型,代表fetch到的记录总行数

1. 例2：

declare

cursor c is select \* from emp;

v\_emp c%rowtype;

begin

open c;

loop

fetch c into v\_emp; --如果抓取不到则v\_emp保存的还是上一条记录

exit when (c%notfound);

dbms\_output.put\_line(v\_emp.ename); --如果把这行挪到上边，最后一条记录打印两边

end loop;

close c;

end;

1. 例3：

declare

cursor c is select \* from emp;

v\_emp c%rowtype;

begin

open c;

fetch c into v\_emp

while(c%found) loop

fetch c into v\_emp; --如果抓取不到则v\_emp保存的是上一条记录

dbms\_output.put\_line(v\_emp.ename); --如果把这行挪到上边，第一条不打印，最后一条记录打印两边

end loop;

close c;

end;

1. 例4： for循环

declare

cursor c is select \* from emp;

begin

for v\_emp in c loop

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

end loop;

end;

1. 例5：带参数的游标

declare

cursor c(v\_deptno emp.deptno%type,v\_job emp.job%type)

is select ename,sal from emp where deptno = v\_deptno and job =v\_job;

begin

for v\_temp in c(10,’CLERK’) loop

dbms\_output.put\_line(v\_temp.ename);

end loop;

end;

1. 例6：可更新的游标

declare

cursor c is select \* from emp2 for update;

begin

for v\_emp in c loop

if(v\_emp.sal<2000) then

update emp2 set sal = sal\*2 where current of c;

elsif(v\_emp.sal=5000) then

delete from emp2 where current of c;

end if;

end loop;

commit;

end;

1. 存储过程：

PL/SQL 语句起个名字作为数据库对象保存起来

语法：

create [or replace] procedure procedure\_name

is –用is声明变量

声明变量

begin

end;

调用方式：

使用命令调用:exec p;

写一个匿名块，在块中进行调用 begin p end;

1. 带参数的存储过程

in 传进来的参数 只读，不能修改，不指明的时候默认为in

out 传出的参数

形参number varchar2 不允许制定长度

create or replace procedure p

(v\_a in number, v\_b number, v\_ret out number, v\_temp in out number )

is

begin

if(v\_a>v\_b) then v\_ret :=v\_a;

else v\_ret := v\_b;

end if;

v\_temp := v\_temp +1；

end;

declare

v\_a number :=3;

v\_b number :=4;

v\_ret number;

v\_temp number :=5;

begin

p(v\_a,v\_b,v\_ret,v\_temp);

……

end;

树形结构：

create table article (

id number primary key,

cont varchar2(4000),

pid number);

insert into article values (1,’蚂蚁大战大象’,0);

insert into article values (2,’大象被打趴下了’,1);

insert into article values (3,’蚂蚁也不好过’,2);

insert into article values (4,’瞎说’,2);

insert into article values (5,’没有瞎说’,4);

insert into article values (6,’怎么可能’,1);

insert into article values (7,’怎么么有可能’,6);

insert into article values (8,’可能性大了’,6);

insert into article values (9,’大象进医院了’,2);

insert into article values (10,’护士是蚂蚁’,9);

create or replace procedure p (v\_id article.id%type, v\_grade binary\_integer) is

cursor c is select \* from article where pid = v\_id;

v\_preStr varchar2(1024);

begin

for v\_i in 1..v\_grade loop

v\_preStr :=v\_preStr || ‘-----’;

end loop;

for v\_a in c loop

dbms\_output.put\_line(v\_preStr || v\_a.cont);

p( v\_a.id,v\_grade +1);

end loop;

end;

begin

p(0,0)

end;

1. 自定义函数

可以在语句里调用，存储过程和匿名块都有参数声明、begin 和end 存储过程是把PL/SQL 语句封装成对象，存储过程可以用匿名块调用也可以直接执行，而函数可以在查询语句中调用，函数中不能出现 DDL DML 语句

create[or replace ] function function\_name

[(parameter,…)]

return datatype(不需要写长度)

is

variable declaration;

begin

statement;

end;

例1：

create or replace function covert\_date

(v\_date date)

return varchar2

is

begin

return to\_char(v\_date,’YYYY-MM-DD HH24:MI:ss’);

end;

调用方法：

select empno,ename,covert\_date(hiredate) from emp;

1. 触发器：

是数据库发生某种操作时自动运行的一类程序，不能被直接调用，只能在发生某事件时被触发，也是系统自动进行调用

create [or replace] trigger trigger\_name

before/after –定义触发时机

event1 [or event2 or event3…] on table\_name [for each row]

begin

end;

event 通常是DML语句 即：update insert delete

利用tigger 解决有外键约束的父项不能被修改的问题

create or replace trigger trig

after update on dept

for each row

begin

update emp set deptno = :NEW.deptno where deptno = :OLD.deptno;

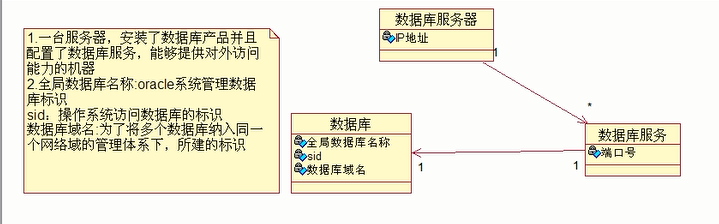
end;

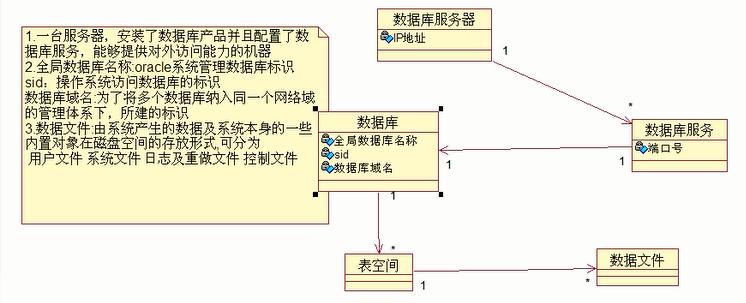
:NEW.deptno对应于deptno=99 ，:OLD.deptno 对应于deptno=10

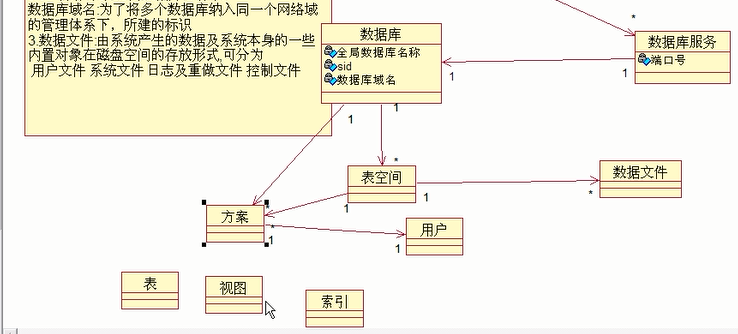
update dept set deptno = 99 where deptno=10

triger 执行完才会检查数据完整性

# Oracle相关概念







一个数据库服务器对应多个数据库服务

一个数据库服务对应一个数据库

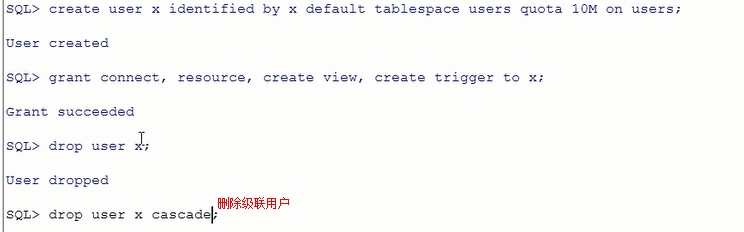
一个数据库对应多个表空间（表空间是一个逻辑概念）

一个表空间对应多个数据文件

一个表空间对应多个方案

一个方案对应一个用户

用户授权语句：



数据字典

# 数据库的导出导入

在cmd命令行

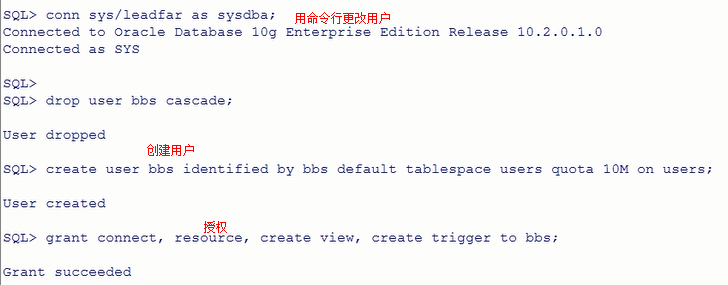
exp

用户名：scott

口令：tiger

缓冲区大小4096 （直接回车）

导出文件： c:\bbs.dmp



imp

用户名：scott

口令：tiger

导入文件： c:\bbs.dmp

缓冲区大小（直接回车）

之后直接回车

# 远程访问数据库

配置和移植工具—>NetConfiguration Assistant

# JDBC

java database connectivity

是一套允许Java与SQL数据库对话的程序设计接口

需要开启

OracleServiceMYORACLE 主服务，需启动

OracleOraDb11g\_home1TNSListener 网络监听器

PreparedStatement 简化了拼串，防止sql注入，预编译提高性能

调用存储过程

//发送SQL语句

CallableStatment cstmt = null;

String sql = “{call 存储过程名(?,?,?,?)}”;

cstmt = conn.prepareCall(sql);

//注册传出参数

cstmt.registerOutParameter(3,Types.INTEGER);

cstmt.registerOutParameter(4,Types.INTEGER);

cstmt.setInt(1,3);

cstmt.setInt(2,4);

cstmt.setInt(4,10);

cstmt.execute();

System.out.println(cstmt.getInt(3)+”,”+cstmt.getInt(4));

默认的是事务自动提交，即无法实现两条SQL语句在同一个事务中（只要有一条失败都回滚事务），所以设置事务不自动提交，

conn.setAutoCommit(false);

提交事务时需调用conn.commit();

设置结果集可滚动，默认是不可滚动

ResultSet.TYPE\_FORWARD\_ONLY

//不可滚动的结果集，只能向前移动

ResultSet.TYPE\_SCROLL\_SENSITIVE

//可滚动的结果集，向前后移动，但对外部更新敏感

ResultSet.TYPE\_SCROLL\_INSENSITIVE

//可滚动的结果集，向前后移动，对外部更新不敏感

ResultSet.CONCUR\_READ\_ONLY

//只读结果集

ResultSet.CONCUR\_UPDATABLE

//可更新的结果集

stmt = conn.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE,

ResultSet.CONCUR\_READ\_ONLY);

rs.next() 向下移动一条

rs.last() 游标移动到最后一条

rs.previous() 向前移动一条

rs.absolute(7) 定位到第七行

Oracle不支持敏感，不支持可更新游标

1. 练习

//只存年月日这种时间格式

pstmt.setDate(5,new java.sql.Date(s.getBirthday().getTime()));

//存完整的时间格式

pstmt.setTimestamp(7,

new java.sql.Timestamp(s.getRegTime().getTime()));

//存字符流，s.getPhtoto()是图片路径,class12不支持class16支持

pstmt.setBlob(7, new FileInputStream(s.getPhtoto()));

批处理

String sql=”delete from t\_student where id = ?”

pstmt = conn.prepareStatement(sql);

for(int id:ids){

pstmt.setInt(1,id);

pstmt.addBatch();

}

pstmt.executeBatch(); //只建立一次连接，把多条SQL语句一次提交到数据库

# 补充知识

1. 用户相关

conn sys/password as sysdba 以sys身份登录

conn system/manager 以system身份登录

show user; 列出当前用户

conn scott/tiger 以普通用户scott身份登录

passw改密码

exit 退出

disc 与数据库断开连接

1. 找回密码

cmd

sqlplus/nolog 去掉日志

conn /as sysdba

alter user 用户名 identified by 新密码

1. 权限相关

系统权限：用于控制用户可以执行的一个或是一组数据库的操作

create session 连接数据库

create table\view\trigger\pcocedure

grant …… to username with admin option 允许继续传播系统权限（system对A授权，A对B授权，撤销A的授权，B能使用）

用户管理常用命令

select \* from scott.emp;

create user mike identified by m;创建用户

drop user mike;

drop role rolename;

grant create session to mike; 给用户授权

grant conn to mike;

对象权限：用户访问其他用户(方案)对象的权限

scott.emp表 select\update\delete\insert

grant …on… to username with grant option允许继续传播对象权限（scott对A授权，A对B授权，撤销A的授权，B也不能使用）

属主可以把自己表的操作权限可以给其他用户

Conn scott;

Grant select on emp to newuser1 with grant option;

Conn newuser1;

Grant select on emp to newuser2;

Revoke select on emp from newuser 回收授权

1. 角色相关:

自定义角色：dba ,create role

Create role 角色名 not identified

Create role 角色名 identified by 密码

Grant … to 角色名 with …

例如：grant create session to role1 with admin option

Grant select,update on scott.emp to role1

Grant role1 to wy with admin option把角色分配给用户

预定义角色（安装时已配置）

connect 一般应用开发人员所需要的大部分权限(create session，create table/view/…)

dba(数据库管理员权限)

resource一般应用开发人员所需要其他权限(create procedure,trigger)

显示所有角色：select \* from dba\_roles

显示角色具有的系统权限：

Select privilege,admin\_option from role\_sys\_privs where role=’角色名’

显示角色具有的对象权限：Dba\_sys\_privs

显示用户具有的角色，及默认的角色：Dba\_role\_privs

1. 关系数据库理论

单一关系模式：Student<U、F>

数据依赖-函数依赖

非平凡的函数依赖

平凡的函数依赖

完全函数依赖

部分函数依赖

传递函数依赖

直接函数依赖

范式

一范式：字段不可分

二范式：非主属性对码都是完全函数依赖

三范式：非主属性对码没有传递函数依赖

包含在任何一个候选码中的属性，称为主属性，不包含在任何码中的属性称为非主属性，没有非主属性至少满足第三范式

Bc范式：消除主属性对码（除自身以外的码）的部分函数依赖和传递函数依赖

1. 数据库设计

数据库设计六个阶段

1. 1：1关系的转化

1：1转换成独立关系模式

1：1和任意一端关系模式合并

1. 1：n关系的转化

1：n转换成独立的关系模式

1：n和n端关系模式合并

1. m：n关系的转化

m：n转换成独立的关系模式