-- 结构

select distinct 字段1,字段2 from 表名 where 条件 group by 字段 having 条件 order by 字段;

-- 含义

select -> 查询|显示的内容

from ->数据来源

where -> 条件， 筛选每行的条件

group by -> 按照什么标准分组

having -> 条件，筛选每组的条件

order by -> 对要显示的结果排序

-- 执行顺序

1) from -> 所有的数据

2) where -> 留下符合条件的单行记录

3) group by -> 具有共同特性一组一组的记录

4) having -> 留下的符合条件的组

5) select -> 显示记录

6) order by ->对要显示的记录进行排序

DDL数据定义语言: create:创建；drop:删除；alter:修改；rename:重命名;truncate:截断

DML数据管理(操作)语言: insert:插入；delete:删除；update:更新；select:查询

DCL 数据库控制语言: grant:授权；revoke:回收权利；commit:提交事务；rollback:回滚事务

查询行(记录): where 过滤行记录条件 ,条件有

a)、= 、 >、 =、 <=、 !=、 <>、 between and

b)、and 、or、 not、 union、 union all、 intersect 、minus

c)、null :is null、 is not null、 not is null

d)、like :模糊查询 % \_ escape('单个字符')

e)、in 、 exists(难点) 及子查询

集合操作

Union、Union All、Intersect、Minus

λ Union，并集(去重) 对两个结果集进行并集操作，不包括重复行同时进行默认规则的排序； λ Union All，全集(不去重) 对两个结果集进行并集操作，包括重复行，不进行排序 ；

λ Intersect，交集(找出重复) 对两个结果集进行交集操作，不包括重复行，同时进行默认规 则的排序；

λ Minus，差集(减去重复) 对两个结果集进行差操作，不包括重复行，同时进行默认规则的排 序

like :模糊查询

模糊查询，使用通配符:

λ %:零个及以上(任意个数的)的字符

λ \_:一个字符

λ 遇到内容中包含 % \_ 使用escape('单个字符')指定转义符

排序

使用 ORDER BY 排序，排序不是真实改变存储结构的顺序，而是获取的集合的 顺序。

λ 顺序 :asc（默认） desc

λ 多字段: 在前面字段相等时，使用后面的字段排序

λ 空排序: 降序为 desc，注意 null 为最后nulls first, nulls last

# 函数—单行函数

## 日期函数

λ sysdate/current\_date 以date类型返回当前的日期

λ add\_months(d,x) 返回加上x月后的日期d的值

λ LAST\_DAY(d) 返回的所在月份的最后一天

λ months\_between(date1,date2) 返回date1和date2之间月的数目

λ next\_day(sysdate,'星期一') 下周星期一

## 转换函数（重点\*\*\*）

to\_date(c,m) ◊ 字符串以指定格式转换为日期

select \* from emp where hiredate

between to\_date('1982-01-01', 'yyyy-mm-dd') and

to\_date('1982-12-31', 'yyyy-mm-dd');

to\_char(d,m) ◊ 日期以指定格式转换为字符串

select to\_char(sysdate,'yyyy"年"mm"月"dd"日"') from dual;

## 其他函数 (保证类型兼容)

--1)、nvl nvl(string1,string2) ◊ 如果string1为null,则结果为string2的值

select ename, nvl(null,0) from emp; select ename, nvl(to\_char(comm),'hello') from emp;

--2)、decode decode(condition,case1,express1,case2 , express2,….casen , expressn, expressionm) select ename,decode(deptno, 10,'十',20,'二十') from emp;

--3)、case when then else end

--给所有的员工 涨薪,10-->10% 20-->8% 30 -->15% 其他-->20%

--decode

select ename, sal, deptno, decode(deptno, 10, sal \* 1.1, 20, sal \* 1.08,

30, sal \* 1.15, sal \* 1.2) raisesal from emp;

--case when then else end

select ename, sal, deptno, (case deptno when 10 then sal \* 1.1

when 20 then sal \* 1.08

when 30 then sal \* 1.15 else sal \* 1.2 end)

raisesal from emp;

# 组函数

组函数|多行函数|聚合函数 即多条记录 返回一个结果。我们需要掌握如下几个组函数： avg 、sum、 min、 max、 count

1)、count :统计记录数 count() -->\* 或一个列名

2)、max min: 最大值 最小值

3)、sum：求和

4)、avg:平均值

注意: 1、组函数仅在选择列表和Having子句中有效

2、出现组函数，select 只能有组函数或分组字段

说明：

λ 组信息 与单条记录不能同时查询

--查询 最高薪水的员工名称 及薪水

~~select max(sal), ename, sal from emp;~~ ◊ 错误

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

λ 组函数 不能用在 where中，能使用的地方 select having

λ null 不参与运算

# 分组

分组: group by , 将符合条件的记录 进一步的分组

过滤组:having , 过滤组信息 ，表达式 同 where 一致

## group by :分组

1)、select 出现分组函数,就不能使用 非分组信息，可以使用 group by 字段

2)、group by字段 可以不出现 select 中 ，反之select 除组函数外的，其他字段必 须出现在group by 中

## 过滤组 having

where :过滤行记录，不能使用组函数， having:过滤组 可以使用组函数

# rowid 和 rownum

## rowed

实现重复记录的删除

## rownum

1、必须排序

2、不能直接取大于 1 的数

# 表连接（92）

当我们获取的数据不是来自于同一张表而是来自于多张表时就需要使用到表连接

select \* from emp;

select \* from dept;

注意:同名列 非\* 必须区分 数据源 、关系列、 过滤条件、字段

## 笛卡尔积

--非\* 必须区分 使用表名 或别名.区分

select \* from emp , dept;

select ename , dname from emp , dept;

select ename, dname, e.deptno from emp e, dept d;

## 等值连接(在笛卡尔积基础上 取条件列相同的值)

--找出30部门的员工名称及部门名称:先关联后过滤

select ename, dname, e.deptno from emp e, dept d where e.deptno=d.deptno and e.deptno=30;

--记录很多时 :先过滤后关联

-- 数据来源： emp (select \* from emp where deptno=30) e , dept(select \* from dept where deptno=30) d

select \* from emp where deptno=30;

select \* from dept where deptno=30;

-- 查询的字段：ename, dname, e.deptno

-- 条件：e.deptno=d.deptno , deptno=30

select ename, dname, e.deptno from (select \* from emp where deptno=30) e ,(select \* from dept where deptno=30) d where e.deptno=d.deptno;

## 非等值连接 > < != <>between and

--查询员工姓名，工资及等级

-- 数据源： emp e, salgrade s

-- 字段： ename, grade, sal -- sal between losal and hisal

select ename, grade, sal from salgrade s, emp e where sal between losal and hisal;

## 自连接: 特殊的等值连接 (来自于同一张表)

--找出 存在上级的员工姓名 及上级名称

-- 数据来源： emp e, emp m

-- 字段： e.ename, m.ename

-- 条件：e.mgr=m.empno

select e.ename, m.ename from emp e, emp m where e.mgr=m.empno;

## 外连接

--找出 所有的员工姓名 及上级名称 --找出 所有部门的员工数 及部门名称

select dname, nu from dept d, (select count(1) nu, deptno from emp group by deptno) e where d.deptno(+)=e.deptno;

看+和, 主表在,的左边就叫左外连接 主表在,的右边叫右连接

# 99 连接

λ 交叉连接 cross join --->笛卡尔积

select \* from emp cross join dept;

λ 自然连接(主外键、同名列) natural join -->等值连接

select \* from emp natural join dept;

--在指定列过程中同名列归共同所有(\*除外)

select deptno,e.ename,d.dname from emp e natural join dept d;

λ join using连接(同名列) -->等值连接

select deptno,e.ename,d.dname from emp e join dept d using(deptno);

λ [inner]join on 连接 -->等值连接 非等值 自连接 (解决一切) 关系列必须区分

--on select ename, dname from emp join dept on emp.deptno = dept.deptno

where emp.deptno = 30;

on 非等值连接 、自连接 --部门编号为30的员工名称 工资等级

select ename, grade from emp e join salgrade s on e.sal between s.losal and s.hisal where deptno=30;

--部门编号为30的员工名称 上级名称

select e.ename,m.ename mname from emp e join emp m on e.mgr =m.empno where e.deptno =30;

--部门编号为30的员工 员工名称 部门名称 工资等级 上级名称

select e.ename, dname, grade, m.ename mname from emp e

join dept d on e.deptno = d.deptno

join salgrade s on e.sal between s.losal and s.hisal

join emp m on e.mgr = m.empno

where e.deptno = 30;

λ left|right [outer] join on|using -->外连接

--所有部门的 部门名称，员工数 --左外 select dname, n from dept d left outer join (select deptno, count(1) n from emp group by deptno) i on d.deptno = i.deptno;

select dname, n from dept d left outer join (select deptno, count(1) n from emp group by deptno) i using (deptno); --右外

select dname, n from (select deptno, count(1) n from emp group by deptno) i right outer join dept d on d.deptno = i.deptno;

λ full join on|using -->全连接 满足直接匹配，不满足 相互补充null ，确保 所有表的记录 都至少 出现一次

select \* from (select 1 no, 'a' "name" from dual union select 2 no, 'b' "name" from dual) a full join (select 1 no, 'c' "name" from dual union select 3 no, 'd' "name" from dual) b on a.no=b.no;

select \* from (select 1 no, 'a' "name" from dual union select 2 no, 'b' "name" from dual) a full join (select 1 no, 'c' "name" from dual union select 3 no, 'd' "name" from dual) b using(no);

# 视图

视图:建立在表|结果集|视图上的虚拟表，有以下作用

1、简化:select 查询语句

2、重用:封装select语句 命名

3、隐藏:内部细节

4、区分:相同数据不同查询

不是所有的用户都有创建视图的权限

1、前提： create view -->组 connect resource dba

2、授权: -->sqlplus /nolog

a)、sys登录 conn sys/123456@orcl as sysdba

b)、授权: grant dba to scott;

回收: revoke dba from scott;

c)、重新登录

create or replace view 视图名 as select语句 [with read only];

要求:所有列必须存在名称。

对视图的删除不会删除原有表的数据

drop view 视图名;

# 设计表

设计表首先应该按需遵循三范式

1、确定表名 2、确定字段名 类型 +约束(主键 外键 非空 默 检查认 唯一)

主键： 唯一标识一条记录（唯一并且非空）

唯一： 唯一

非空：不能为空

默认： 当没给值时使用给定一个默认值

外键：参考其他表（自己）的某个（某些）字段

检查：自定义的规则

## 1.创建表（同时创建约束+指定名称）

创建表的同时创建约束并指定约束的名称，后期方便排错，推荐使用

--删除 （先删除从表 再删除主表 ；同时删除约束）

drop table tb\_txt cascade constraints;

drop table tb\_user cascade constraints;

表名 tb\_user

主键 userid

字段名 中文 类型 为空 默认值 其他说明

userid 流水号 number(5) 否 主键

username 用户名 varchar2(30) 否 长度在4-20

userpwd 密码 varchar2(20) 否 长度在4-18

age 年龄 number(3) 18 大于>=18 gender 性别 char(2) 男 男or 女

email 邮箱 varchar2(30) 唯一

regtime 注册日期 date sysdate

create table tb\_user(

userid number(5),

username varchar2(30) constraint nn\_user\_name not null ,

userpwd varchar2(20) constraint nn\_user\_pwd not null ,

age number(3) default(18) ,

gender char(2) default('男'),

email varchar2(30),

regtime date default(sysdate),

constraint pk\_user\_id primary key (userid),

constraint ck\_user\_name check(length(username)between 4 and 20) ,

constraint ck\_user\_pwd check(length(userpwd) between 4 and 18),

constraint ck\_user\_age check(age>=18),

constraint ck\_user\_gender check(gender in('男','女')),

constraint uq\_user\_email unique(email)

);

--加入注释

comment on table tb\_user is '用户表';

comment on column tb\_user.userid is '流水号,主键';

comment on column tb\_user.username is '用户名';

comment on column tb\_user.userpwd is '密码';

comment on column tb\_user.age is '年龄';

comment on column tb\_user.gender is '性别';

comment on column tb\_user.email is '邮箱';

comment on column tb\_user.regtime is '注册日期';

## 2.创建表（追加创建约束+指定名称）

推荐， 便于后期排错

--删除 （先删除从表 再删除主表 ；同时删除约束）

drop table tb\_txt cascade constraints;

drop table tb\_user cascade constraints;

表名 tb\_user

主键 userid

字段名 中文 类型 为空 默认值 其他说明

userid 流水号 number(5) 否 主键

username 用户名 varchar2(30) 否 长度在4-20

userpwd 密码 varchar2(20) 否 长度在4-18

age 年龄 number(3) 18 大于>=18

gender 性别 char(2) 男 男or 女

email 邮箱 varchar2(30) 唯一

regtime 注册日期 date sysdate

create table tb\_user(

userid number(5),

username varchar2(30) ,

userpwd varchar2(20) ,

age number(3) ,

gender char(2) ,

email varchar2(30),

regtime date default(sysdate)

);

--追加约束

alter table tb\_user add constraint pk\_user\_id primary key (userid);

alter table tb\_user add constraint ck\_user\_name check(length(username)between 4 and 20) ; alter table tb\_user add constraint ck\_user\_pwd check(length(userpwd) between 4 and 18);

alter table tb\_user add constraint ck\_user\_age check(age>=18);

alter table tb\_user add constraint ck\_user\_gender check(gender in('男','女'));

alter table tb\_user add constraint uq\_user\_email unique(email);

--非空与默认

alter table tb\_user modify (username constraint nn\_user\_name not null);

alter table tb\_user modify (userpwd constraint nn\_user\_pwd not null);

alter table tb\_user modify (age default(18));

alter table tb\_user modify (gender default('男'));

--加入注释

comment on table tb\_user is '用户表';

comment on column tb\_user.userid is '流水号,主键';

comment on column tb\_user.username is '用户名';

comment on column tb\_user.userpwd is '密码';

comment on column tb\_user.age is '年龄';

comment on column tb\_user.gender is '性别';

comment on column tb\_user.email is '邮箱';

comment on column tb\_user.regtime is '注册日期';

## 3.修改表结构

1、修改表名 :rename to

rename tb\_txt to tb\_txt\_new;

2、修改列名: alter table 表名 rename column to

alter table tb\_txt\_new rename column txtid to tid;

3、修改类型: alter table 表名 modify(字段 类型)

alter table tb\_txt\_new modify(tid varchar2(20));

4、修改约束: 先删除 后添加

alter table tb\_txt\_new add col varchar2(30);

5、添加列: alter table 表名 add 字段 类型

6、删除列:alter table 表名 drop column 字段

alter table tb\_txt\_new drop column col;

select \* from tb\_txt\_new;

## 4.删除表

1、先删除从表 再删除主表

2、删除表的同时删除约束

drop table tb\_txt\_new cascade constraints;

drop table tb\_user cascade constraints;

## 5.截断数据

truncate: 截断所有的数据 ,如果截断的是主表，结构不能存在 外键关联

截断数据同时 从结构上 检查

create table emp\_his as select \* from emp where 1=1;

select \* from emp\_his;

--截断所有的数据

truncate table emp\_his;

--不能截断: truncate table dept;

# 序列

## 创建

create sequence 序列名 start with 起始值 increment by 步进

## 使用

在操作数据 添加 更新 -->主键

1)、currval :当前值

2)、nextval:下个值

create sequence seq\_tb\_user start with 2 increment by 2;

drop sequence seq\_tb\_user;

select seq\_tb\_user.nextval from dual;

select seq\_tb\_user.currval from dual

## 删除

drop sequence 序列名

# 事务

事务有一下四个特点：（ACID）

1、原子性（Atomic）：事务中所有数据的修改，要么全部执行，要么全部不执行。

2、一致性（Consistence）：事务完成时，要使所有所有的数据都保持一致的状态， 换言之：通过事务进行的所有数据修改，必须在所有相关的表中得到反映。

3、隔离性（Isolation）：事务应该在另一个事务对数据的修改前或者修改后进行访问。

4、持久性（Durability）：保证事务对数据库的修改是持久有效的，即使发生系统故障，也不应该丢失。

## 1.事务的开启

自动开启于 DML 之 insert delete update

## 2.事务的结束

### 1.成功

λ 正常执行完成的 DDL 语句：create、alter、drop

λ 正常执行完 DCL 语句 GRANT、REVOKE

λ 正常退出的 SQLPlus 或者 SQL Developer 等客户端

λ 如果人工要使用隐式事务，SET AUTOCOMMIT ON (只针对一个连接)

λ 手动提交 ：使用 commit

### 2.失败

λ rollback ，手动回滚

λ 非法退出 意外的断电

注意：rollback 只能对未提交的数据撤销，已经 Commit 的数据是无法撤销的，因为 commit 之后已经持久化到数据库中。

## 3.DML

### 1.insert

insert into 表名 [(字段列表)] values(值列表);

添加记录添加记录时需要满足以下条件

λ 类型 长度 兼容: 字段 兼容值

λ 值满足约束 :主键 (唯一+非空) 非空(必填) 唯一(不重复 ) 默认(没有填写使用默认值) 检查(满足条 件) 外键(参考主表主键列的值)

λ 个数必须相同: 指定列，个数顺序与列相同；没有指定，个数与表结构的列个数和顺序相同 (null也得占位，没有默认值)

添加数据

insert into 表名 values(和表结构顺序和个数和类型一致的数据->可以手写也可以从别的表中获取的);

insert into 表名(指定列名) values(和指定的列个数、顺序、类型一致的列数据)

eg:

insert into 表(指定列) select 查询列 from 源表 where 过滤数据;

insert into 表(指定列) values(值列表);

insert into 表名 select 查询列 from 源表 where 过滤数据;

insert into 表名 values(值列表 );

### 2.update

update 表名 set 字段=值 [,....] where 过滤行记录;

更改数据 从已有表中查询数据更改字段值：

update 表名 set(字段列表)=(select 字段列表 from 源表 where 过滤源表记录 ) where 更新记录的条件 手动更改字段值：

update 表名 set 字段=值 [,....] where 过滤行记录;

### 3.delete

delete [from] 表名 where 过滤行记录

说明：

1、delete 可以删除指定部分记录，删除全部记录

2、记录上存在主外键关联时， 删除存在关联的主表的记录时,注意 参考外键约束， 约束强制不让删除 先删除从表 再删除主表

截断数据与删除数据区别 truncate 与delete 区别

1、truncate -->ddl ，不涉及事务，就不能回滚 delete -->dml ,涉及事务，可以回滚

2、truncate 截断所有的数据 delete 可以删除全部 或者部分记录

3、truncate从结构上检查是否存在主外键，如果存在，不让删除 delete 从记录上检查是否存在主外键，如果存在，按参考外键约束进行删除。

ddl: 异常则失败,没有异常就是成功

dml: insert update delete -->异常则失败，记录数<=0则失败; 记录数>0就是成功

select -->异常则失败,返回结果集 存在结果集就是成功