**SQL**

* 数据库:按照数据结构,组织,管理信息,存放数据的仓库.
* DBMS管理数据库的软件.
* 数据管理阶段经历的过程:手工-->文件-->数据管理阶段

**CMD操作**

**数据库**

* 查看数据库的版本:select version();
* 查看有哪些数据库:show databases;
* 切换数据库命令:use 数据库名;
* 查看数据库中有哪些数表:show tables;
* 查看数据库的表结构:desc 表名;

**存储引擎**

* 存储引擎:指的是创建表的类型
* 查看存储引擎:show engines;
* **数据库**
* 创建数据库:create database 数据库名;
* 删除数据库:drop database 数据库名
* 创建数据库表:create table 表名(字段名 数据类型 [约束], 字段名 数据类型 [约束], 字段名 数据类型 [约束]);
* Desc 表名;查看表
* Show tables;展示有哪些表

例如:create table stu(

sid varchar(20) primary key,

sname varchar(15),

sage int(4)

);

//创建动物表

Aid 编号 长度是20 是主键

Anme姓名 长度是10

Asex性别 长度是4

Acolor颜色 长度是5

Create table ani(

Aid varchar(20) primary key,

Aname varchar(10),

Asex varchar(4),

Acolor varchar(5)

);

**实体之间的关系:**一对一,一对多,多对多

**作业:**

**1.**

Create table DEPT(

DEPTNO int(2) primary key AUTO\_INCREMENT,

DNAME varchar(20),

LOC varchar(4)

);

**2.**

Create table EMP(

EMPNO INT(4) primary key AUTO\_INCREMENT ,

ENAME VARCHAR(10),

JOB VARCHAR(10),

MGR VARCHAR(50),

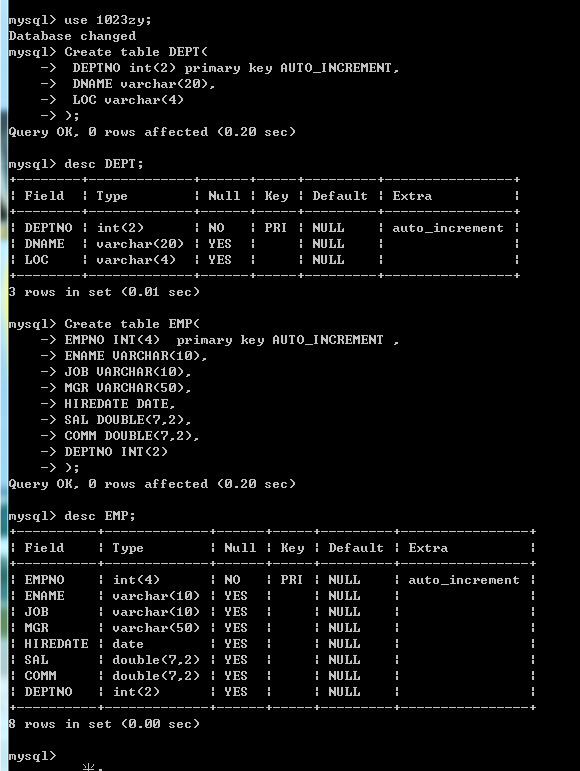
HIREDATE DATE,

SAL DOUBLE(7,2),

COMM DOUBLE(7,2),

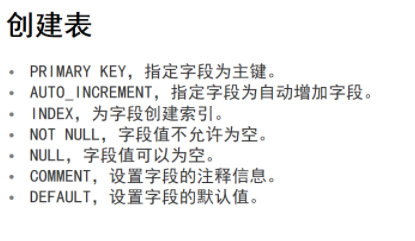
DEPTNO INT(2)

);



**1023**

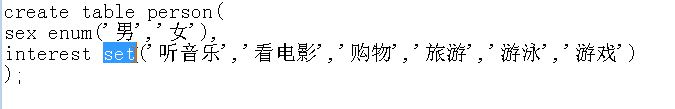
* **创建表**

****

* **五种约束 constraint**

1. NOT NULL 非空约束,指定某列的所有行数据不能包含空值.
2. UNIQUE 唯一性约束,指定列或者列的组合 的所有行数据必须唯一.
3. PRIMARY KEY 主键约束,表的每行的唯一性标识,指定列或者列的组合 的所有行数据必须唯一.
4. FOREIGN KEY 外键约束,在列及引用列上建立的一种强制依赖关系.
5. CHECK 检查性约束,在列上指定一个必须满足的条件.

* **枚举**



Create table person(

Sex enum(’男’,’女’),

Interest set (’听音乐’,’看电影’, ’购物’,’旅游’,’游泳’,’游戏’)

);

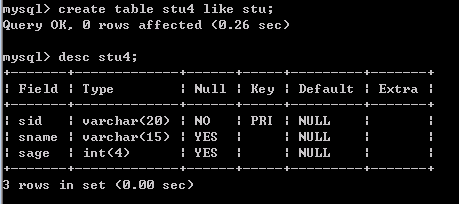
* **插入**

**C:\Users\neustu\Desktop\插入数据.PNG**

Insert into person values(‘男’,’看电影,游泳,听音乐’);

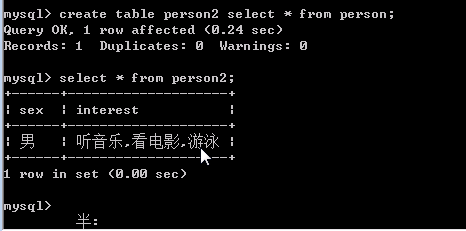
* **复制表**
* 复值表结构

Create table 表名 like 原表名



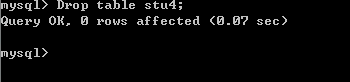
* 复制表结构和内容

Create table 表名 select\*from原表名

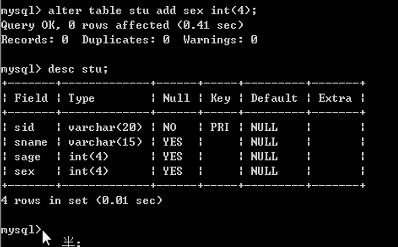


* **删除表**

Drop table 表名;



* **修改表结构**
* Alter table 表名 add列名 数据类型和长度 列属性(增加)



* Alter table 表名 modify 列名 新数据类型和长度 新列属性(修改)
* Alter table 表名 drop column 列名(删除)
* **修改表名:**

Alter table 旧表名 rename 新表名



**第三章-数据操作与事务控制(重点)**

* **DML(数据操作语言)**

增删改

* **插入语句:insert**

**Insert into 表名(字段名1,,,,,)values(值1,,,,,,);**

* **按照指定列进行插入(字段的个数和类型要保持一致)**

向部门表中插入数据 编号是1号 名称是研发部 地点北京

insert into dept(deptno,dname,loc)values(1,"研发部","北京");

* **插入空值**

--向部门表中插入2号部门,名称运营部

insert into dept(deptno,dname,loc)values(2,"运营部",**null**);

* **插入日期**

SYSDATE():获取当前时间

* **按照所有列插入数据**

insert into 表名 values(值1,值2,,,,,,,,);

insert into dept values (3,"后勤部",null);

* **批量插入数据**

insert into 表名 values(值1,值2 ,,,,,,),(值1,值2,,,,,,),(值1,值2,,,,,,);

* **修改数据:update**

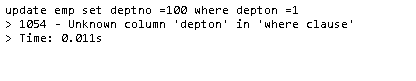
**update 表名 set 字符段=值,字符段=值 [where 条件];**

-- 把部门编号为10的员工部门编号改为20,工资+100

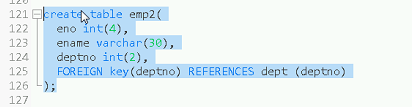
update emp set deptno =20,sal=sal+100 where deptno =10;

* **完整性约束错误**

update emp set deptno =100 where depton =1;



* **创建外键关联**



* **NULL**

条件判断如果为null是 is null;

--修改奖金为null的员工,奖金设置为0

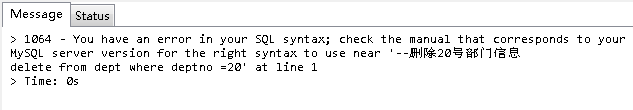
update emp set comm=0 where comm **is null**;

* **删除数据:delete**

**delete from 表 where**

* 删除记录时**完整性约束错误**

delete from dept where deptno =20;



* **截断表**

TRUNCATE table 表名;

**TRUNCATE和DELETE区别---(面试题)**

TRUNCATE是DDL, 只能删除表中所有记录,释放存储空间,使用ROLLBACK不可以回滚.

DELETE 是DML, 还可以删除指定记录,不释放储存空间,使用ROLLBACK可以回滚.

* **事务(TPL)**

**事务特性,事务处理**

* **事务概念**:是一个而工作单元由一个或多个sql语句组成,执行的时候要么都成功,要么都失败.由此保持数据的一致性.
* **事务组成:**在数据库中,事务由一组相关的DML或SELECT语句,加上一个TPL语句(COMMIT,ROLLBACK) 或一个DDL(CREAT,ALTER,DROP,TRUNCATE,RENAME)语句 或一个DCL(GRANT,REVOKE)语句.
* **事务的四个特性**(ACID):原子性 一致性 隔离性 持久性
* **事务处理**语言:简称TPL,有两种方法来实现事务处理

1. 用begin,rollback,commit来实现

Begin:开始一个事务;

COMMIT :对事务进行确认提交;

ROLLBack:对事物进行回滚;

1. 直接用set来改变MySQL的自动提交模式:

mysql 是**默认自动提交**的,也就是运行一个sql语句会自动提交执行

set autocommit=0;禁止自动提交

set autocommit=1;开启自动提交

* **隐式结束**:当下列任意一种情况发生时,会发生隐式提交

执行一个DDL(CREAT,ALTER,DROP,TRUNCATE,RENAME)语句;

执行一个DCL(GRANT,REVOKE)语句;

* **隐式回滚:**当下列任意一种情况发生时,会发生隐式回滚

客户端强行退出

客户端连接到服务器端异常中断

系统崩溃

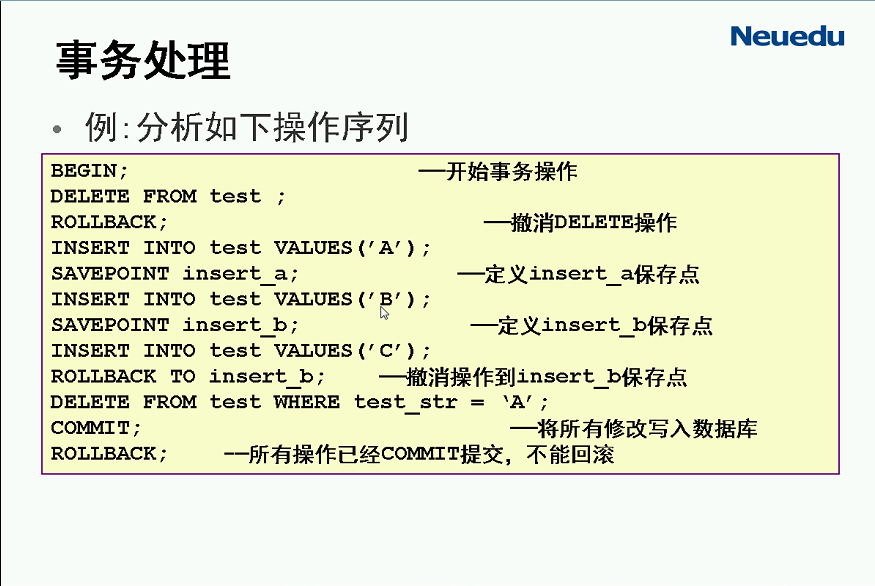
* **设置保存点:**如果在一个事物内,想要回滚到指定位置,不是回滚到事物的起始点,可以通过保存点来实现.

SAVEPOINT savepointname;--定义一个保存点语句;

ROLLBACK TO savepointname;--回滚到指定保存点

注意:如上两条语句不结束事务的执行.

* **例:**



**第四章-简单查询**

**结构化查询语言**

可分为五类:



**查询语句select:**

select 字段名1,字段名2,,,,,from表名 [where 条件]

1. **查询表中所有数据(\*表示所有列)**

select \*from 表名;

1. **算术运算符:改变输出结果**

+ - / \*

1. **空值**

空值是指一种无效的,未赋值,未知的或不可用的值,不能正常参与运算,任何包含空值的运算表达式运算后的结果都为空值null,通过ifnull可解决.

select empno,ename,hiredate,deptno,sal\*12+**IFNULL(comm,0)** as '年薪' from emp

1. **列别名**

-- **第一种**带as :**列名 as 列别名**

* select 字段 as 名字,字段 as 名字....from 表名
* select empno **as** '员工编号',ename **as** '姓名',hiredate as '日期',deptno as '部门编号',sal\*12+IFNULL(comm,0) as '年薪' from emp

-- **第二种**不带as:**列名 列别名**

* select 字段 名字,字段 名字....from 表名
* select empno '员工编号',ename '姓名',hiredate '日期',deptno '部门编号',sal\*12+IFNULL(comm,0) '年薪' from emp

--**以下三种情况列别名两侧需加双引号**

* 列别名中包含空格
* 要求区分大小写
* 包含特殊字符

1. **消除重复行distinct**

select **distinct** deptno from emp;

1. **查看表结构**

Desc 表名;或 右键,Design Table

1. **比较操作符**

**>,<,=,>=,<=,<>不等于**

1. **特殊比较运算符**

**Between…and…:判断要比较的值是否在指定范围之内,包含头和尾;**

**In(集合列表): 判断要比较的值是否和集合列表的任何一个值相等;**

**Like(模糊查询):判断要比较的值是否满足部分匹配;**

%:任意个数的任意字符,’%啊%’ / ’%啊’ / ’啊%’;

\_: 一个任意字符 ,'\_A%';

例:查询员工姓名中第二个而字母是A的员工编号,姓名,薪资,入职日期

SELECT ename,empno,sal,HIREDATE from emp where ename like '\_A%';

**Is null:判断要比较的值是否为空值;**

1. **逻辑运算符 and or !**

**And**:逻辑与,用来连接多个条件表达式.都是真则为真;

**Or**:逻辑或,用来连接多个条件表达式,满足任意一个条件即可;

**!/not**:非 用来否定

1. **优先级**

****

1. **排序 :order by**

对查询的结果进行排序显示:空值升序排在前面,降序放在后面

升序 asc(默认)

降序 desc

例: select ename,sal,sal\*12 as yearsal from emp order by 3 desc; -- 使用结果集的序列号排序

**多列参与排序:** **order by 列名 ,列名 desc/asc**

例:-- 查询员工姓名,部门编号,薪资,要求按照部门序号升序排列,薪资降序排序

select ename,deptno,sal from emp ORDER BY deptno ,sal DESC;( 先排序deptno,如果相同的情况下,再给sal排序).

1. **限制记录长度limit**

Select ,,,,from,,,limit m,n; (m表示起始位, n代表取几个)

分页操作(当前页-1)\*每页显示条数

例:查询第二页内容 每页显示五条

select \*from emp limit 5,5;

**第五章-常用函数**

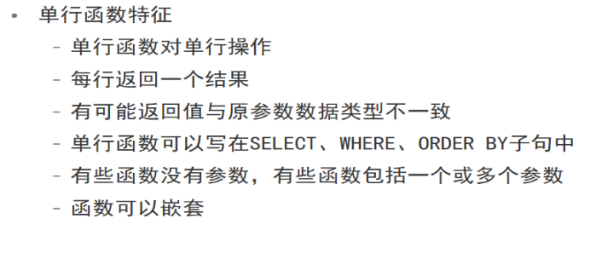
**函数分类**

**单行函数:**

**语法:**函数名[(参数1,参数2,….)]

其中的参数可以是以下之一:变量,列名,表达式

**特征:**



**多行函数:**

**常用函数分类:**

* 数学函数
* 字符串函数
* 日期和时间函数
* 流程控制函数
* 其他函数

**数学函数**

select abs(-1); -- 绝对值

select SQRT(4); -- 返回非负数的平方根

select PI();-- 返回圆周率

select MOD(5,3);-- MOD(x,y)或%:返回x被y除的余数

select CEIL(4.123);-- 返回大于或等于它的最小整数值(向上取整)

select FLOOR(4.123)-- 返回小于或等于它的最大整数值(向下取整)

select ROUND(4.125,2);-- 返回保留小数点后y位,四舍五入的整数

select TRUNCATE(4.125,2);-- 返回被舍弃的小数点后y位的数字x

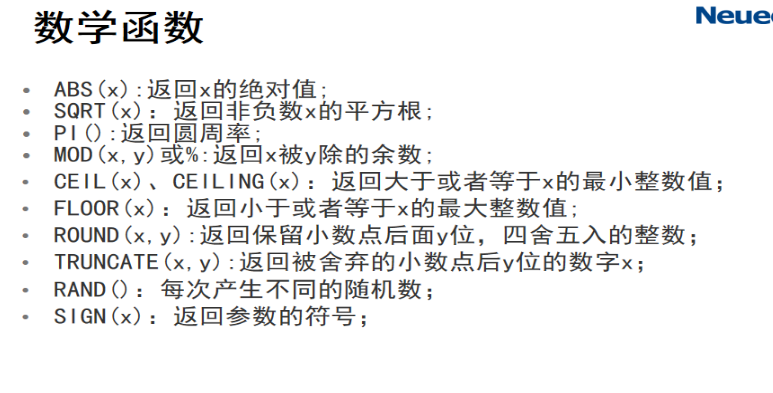
select rand();-- RAND():每次产生不同的随机数

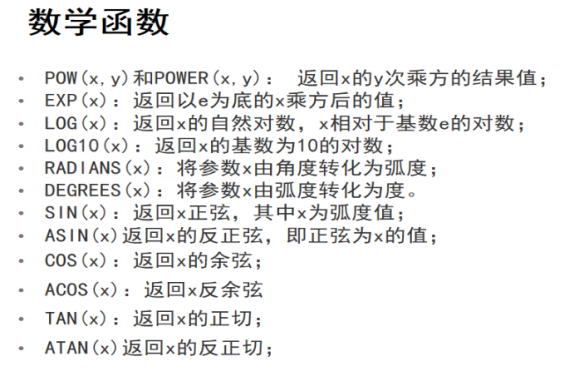
select SIGN(-4);-- SIGN(X):返回参数的符号

select pow(2,3);-- 返回x的y次乘方的结果值

select exp(2);-- 返回以e为底的x乘方后的值

select log(2);-- 返回x的自然对数,x相对于基数e的对数





**字符串函数**

select CHAR\_LENGTH("数据库");-- 返回字符串STR的字符个数

select LENGTH("MYSQL");-- 返回字符串字节的长度

select CONCAT("abc","\*\*\*",null)-- 字符串连接(如果有null结果为null)

select CONCAT(",","张三","李四")-- 用','分隔

select INSERT("mysql",3,2,"data");-- 插入字符串INSERT(s1,x,len,s2)第一数字表示位置(1开始) 第二个数字表示原字符串替换的个数

select LOWER("ABC");-- 字符串转化为小写

select UPPER("abc");-- 字符串转化为大写

select left("I like mysql",3);-- 获取指定长度的字符串函数:LEFT(s,n)返回字符串s开始的最左边n个字符

select left(ename,3) from emp ;-- 显示所有员工名字的前三个字符

select lpad("MySQL",10,"\*");-- 填充字符串函数:LPAD(s1,len,s2)返回字符串s1,其左侧由字符串s2填补到len字符长度

select LTRIM(" MYSQL ");-- 删除字符串空格函数:LTRIM(s)返回字符串s,字符串左侧空格字符被删除.RTRIM(s)返回字符串s,字符串右侧空格字符被删除

select REPEAT("abc",4)-- 重复生成字符串函数

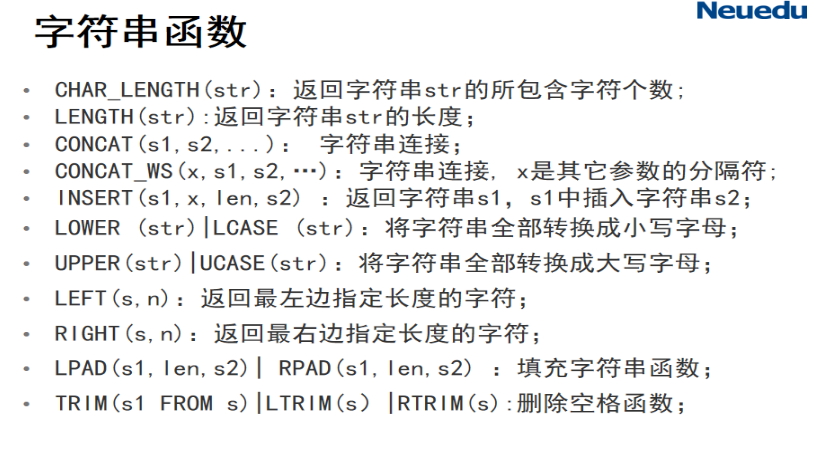
select SPACE(3);-- 空格函数SPACE(N):返回一个有n个空格组成的字符串

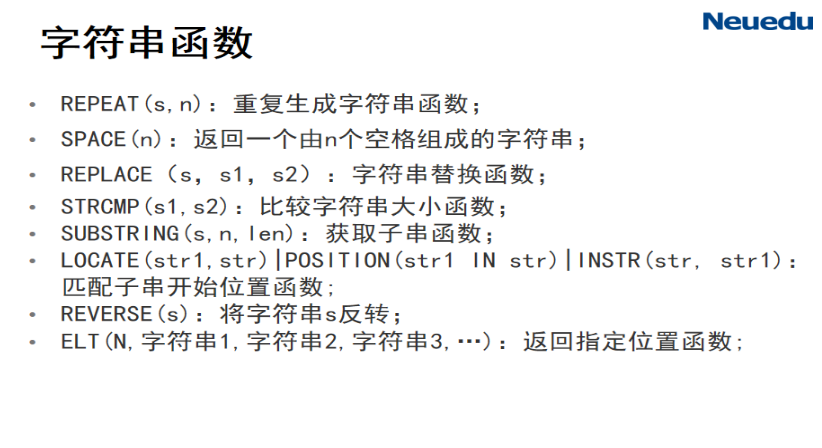
select REPLACE("I like mysql","like","喜欢");-- 字符串替换函数REPLACE(s,s1,s2):将s字符串的s1换成s2

select substring("string",2,3);-- 获取子字符串:SUBSTRING(S,N,LEN)带有len参数的格式从字符串s返回一个长度同len字符相同的字符串,起始于位置n.

select REVERSE("abc");-- 字符串逆排序REVERSE

select ELT(2,"I","like","MySQL")-- 返回指定位置的字符串





**日期函数**

select CURDATE();-- 获取当前日期函数

select now();-- 返回服务器的当前日期和时间

select CURTIME();-- CURTIME():返回当前时间,只包含时分秒

select TIMEDIFF("18:10:10","6:00:00");-- 返回两个日期相减相差的时间

select DATEDIFF("2020-10-27","1995-10-22")-- 返回两个日期相差的天数

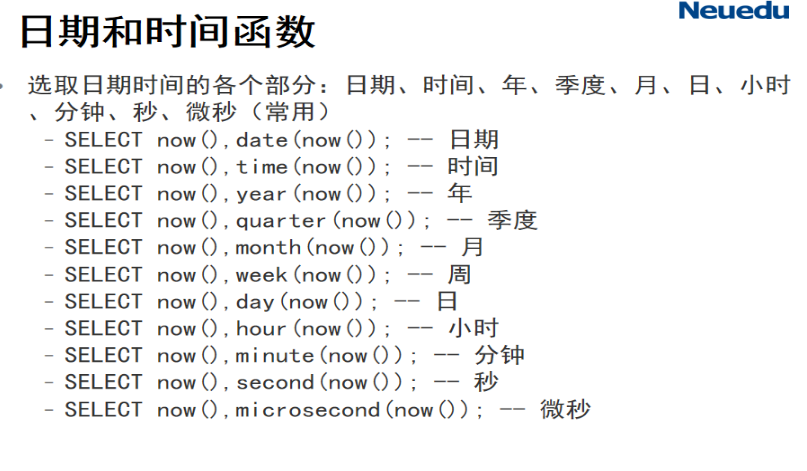
select DATE\_ADD(NOW(),INTERVAL 1 day);-- 计算日期和时间的函数,分别给定的日期date加上(add)或减去(sub)一个时间间隔值expr

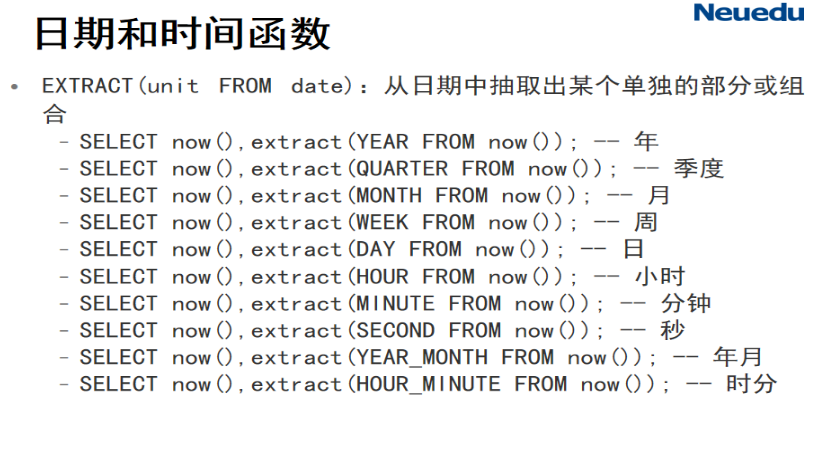
-- 选取日期的各个部分

select now(),date(now());

选取日期的各个部分



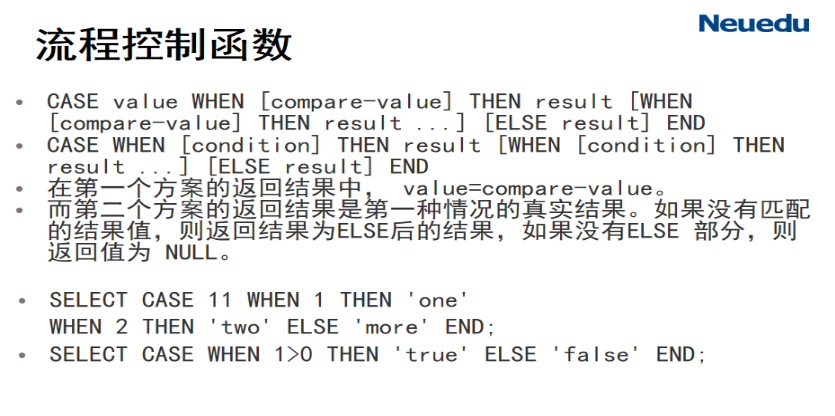




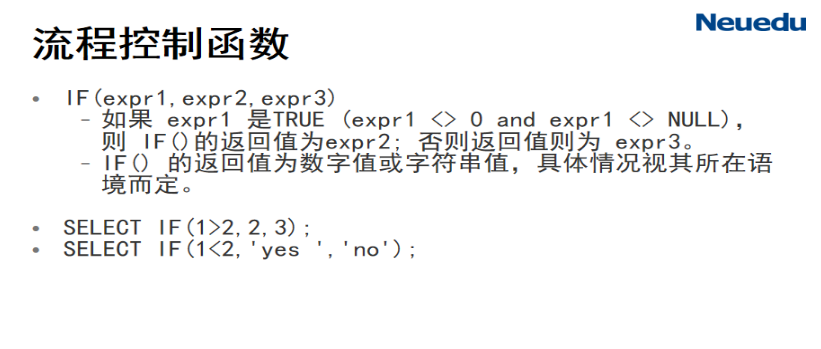
**流程控制函数**

**CASE,IF,IFNULL,NULLIF**

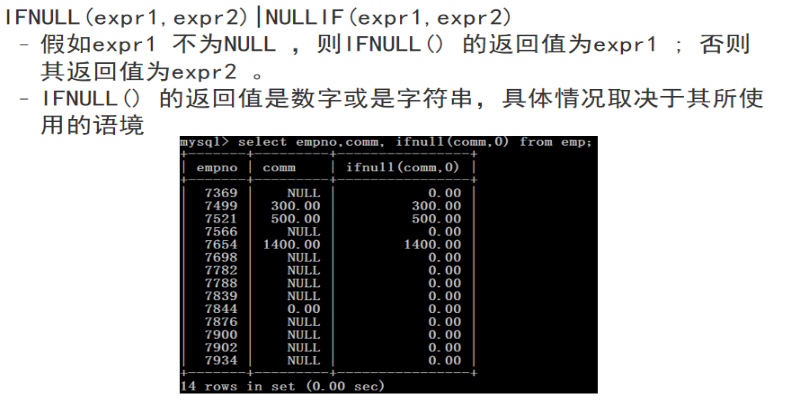
**CASE**



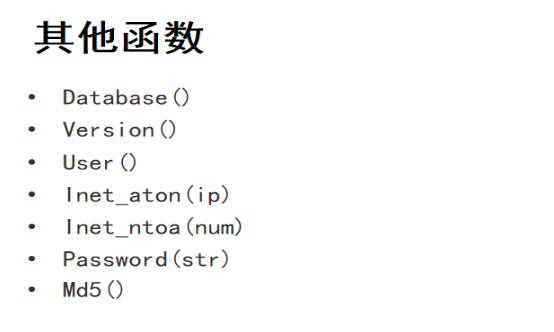
**IF**

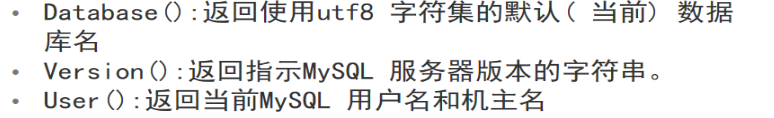


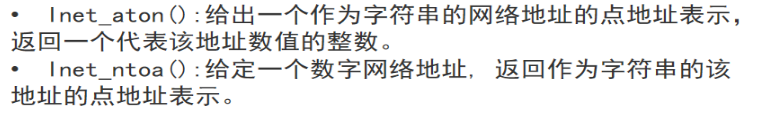
**IFNULL,NULLIF**

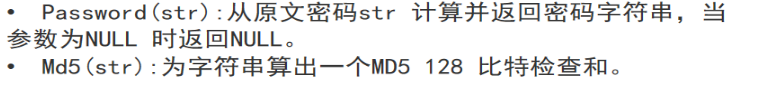


**其他函数**









**第六章-多表连接查询**

**多表连接:**

**按连接条件分:**等值连接,非等值连接

**按其他连接方法分:**外部连接,内部链接

**连接**:是在多个表之间通过一定的连接条件使表之间发生关联关系,进而获取多个表的数据

1. **笛卡尔积**:第一个表中的所有行数据和第二个表的每一行数据都发生连接(开发应避免)

-- 出现笛卡尔积现象:连接条件被省略;连接条件无效.

例:-- 查询员工姓名,部门名称,工作地点(出现笛卡尔积现象)

select ename,dname,loc from emp,dept;

例(正确):-- 查询员工表和部门表的所有信息

select \* from emp,dept where emp.DEPTNO=dept.DEPTN

1. **等值连接:**

**select 字段 from 表名 where a.字段=b.字段**

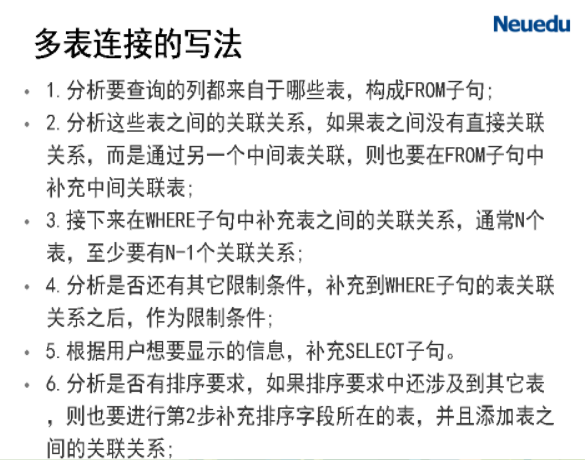
1. **表的别名(带as,不带as)**

例:-- 查询员工编号,姓名,部门编号,工作地点

select empno,ename,e.deptno,loc from emp as e,dept as d where e.DEPTNO=d.DEPTNO;

1. **多于两张表的连接**

**select 字段 from 表名 where a.字段=b.字段 and a.字段=c.字段;**



1. **非等值连接**

**select 字段 from 表名 where 字段 between 字段 and 字段;**

例:-- 查询每个员工的姓名,工资,工资等级(不等值连接)

select ename,sal,GRADE from emp e,salgrade s where sal BETWEEN LOSAL and HISAL

1. **自身连接**

例:-- 查询每个员工的姓名和直接上级的名称

select w.ename ,m.ENAME from emp w,emp m where w.MGR=m.EMPNO

1. **标准的连接语法:**

**7.1 交叉连接**:使用 CROSS JOIN 子句完成.

会产生连个表的交叉乘积,和两个表之间的笛卡尔积一样;

例:-- 查询员工表,部门表所有信息

select \* from emp cross join dept;

**7.2 自然连接**:使用NATURAL JOIN 子句来完成

是对两个表之间相同名字和数据类型的列进行的等值连接;如果两个表之间相同名称的列的数据类型不同,则会产生失误.

例:-- 查询员工表,部门表所有信息

select \* from emp NATURAL JOIN dept

**7.3 USING子句**

1. 可以指定用某个或某几个相同名字和数据类型的列作为连接条件;
2. Using字句中的用到的列不能使用表明和别名作为(前缀);
3. Natural join子句和using子句是相互排斥的,不能同时使用

例:-- 查询员工表,部门表所有信息

select \* from emp join dept using (deptno)

**7.4 ON子句**,指定任意连接条件,或指定要连接的列,可以使用ON句子(提高代码的可读性)

-- 查询员工表,部门表所有信息

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

**7.5 外部连接**

**左外连接**(以from子句中的左表为基表,该表所有行数据按照连接条件无论是否与右边表能匹配上,都会被显示出来)

例:select \* from dept left join emp on dept.DEPTNO=emp.DEPTNO

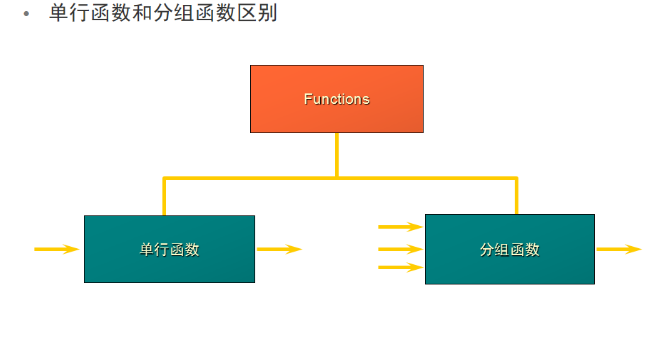
**右外连接**(以from子句中右边表为基表,该表所有行数据按照连接条件无论是否与左边表能匹配上,都会被显示出来)

例:select \* from dept right join emp on dept.DEPTNO=emp.DEPTNO

**第七章-高级查询**

**1分组函数**

**1.1单行函数,分组函数的区别:**



**1.2分组函数:是对数据行的集合进行操作并安组给出一个结果,这个结果可以输出,或做判断条件**

**常用分组函数** **sum avg min max count**

1. **sum** 求和,**avg** 求平均值**(注意null不参与运算使用ifnull解决)** 只针对**数字**

例:求查询所有员工的每个月工资总和,平均工资

select sum(sal),avg(IFNULL(comm,0)) from emp where job ='MANAGER';

1. **min** 求最小值,**max** 求最大值

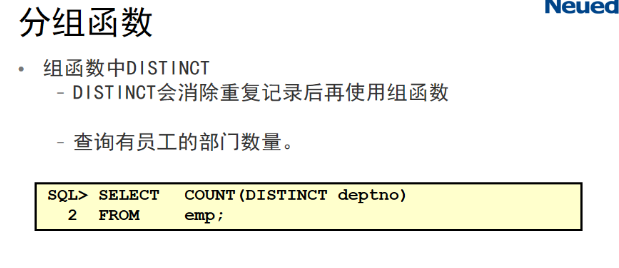
例:查询员工表中最高薪资和最低薪资 既可以是**数字**还可以是**日期**

select max(sal),min(sal) from emp

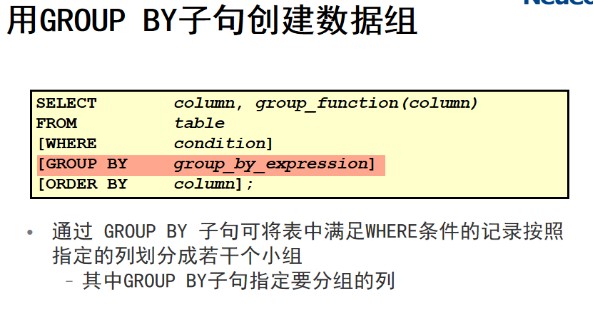
1. **count**:查人数

select count(\*) from emp

select count(empno) from emp -- 优化方案,使用主键进行查询.



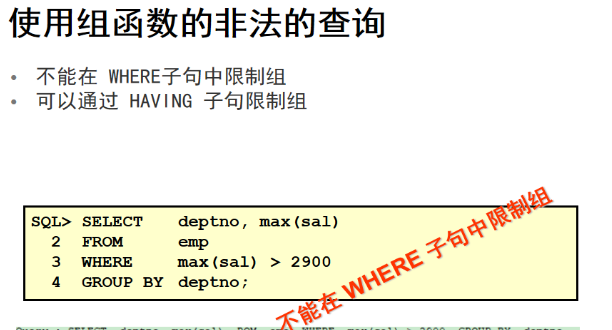
**2.创建数据组:group by**

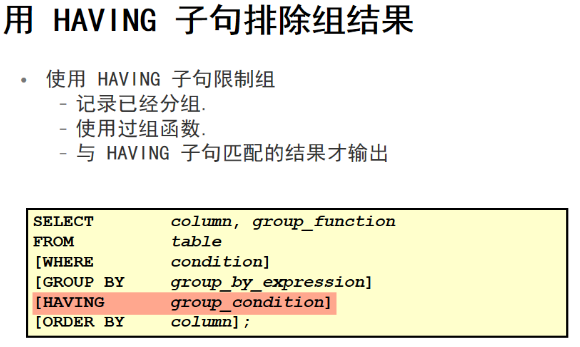


-- 查询每个经理管理多少个员工,经理的编号,名字,要求显示没有经理人的信息

select COUNT(y.empno),y.mgr,l.ENAME from emp y LEFT JOIN emp l on y.mgr=l.empno GROUP BY y.MGR

**3.排除组结果 having(分组函数不能和where连用)**

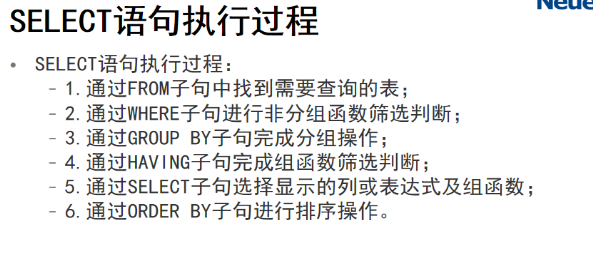




**例**-- 查询每个部门最高工资大于2900的部门编码,最高工资

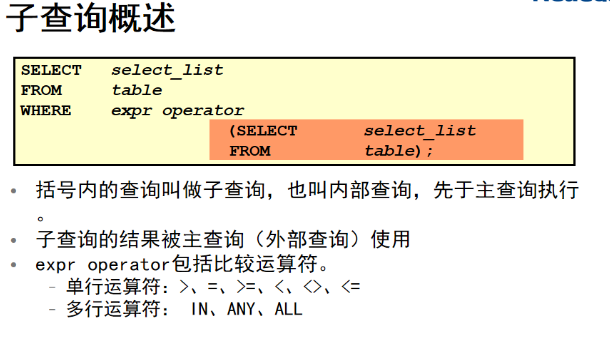
select DEPTNO,MAX(sal) from emp GROUP BY DEPTNO having max(sal)>2900

**4.SELECT语句执行过程**



**5.子查询**

**5.1 概述:**



**5.2 子查询可以嵌于以下SQL子句中:**

* WHERE子句
* HAVING子句
* FROM 子句

**5.3子查询的使用:**

* 1. 子查询要用括号括起来;
  2. 将子查询放在比较运算符的右边;
  3. 对于单行子查询要使用单行运算符;
  4. 对于多行子查询要使用多行运算符

**5.4子查询的类型:**

1. 单行子查询
2. 多行子查询
3. 多列子查询

**5.4.1 单行子查询**

**单行子查询:只能返回一行一列 <,>,>=,<=,=,!=/<>**



例:-- 显示和雇员7369从事相同工作并且工资大于雇员7876的雇员的姓名和工作

SELECT ename, JOB FROM emp WHERE job = ( SELECT job FROM emp WHERE empno = 7369 ) AND sal > ( SELECT sal FROM emp WHERE empno = 7876 );

**5.4.2 多行子查询**

**多行子查询: in,all,any(单行运算符与多行子查询不匹配)返回记录的条数,可以是一条或多条**

1. **IN:操作符和以前介绍的功能一致,判断是否与子查询的任意一个返回值相同**

**例:**-- 查询每个部门最低薪资的员工编号和姓名

select empno,ename from emp where (sal,deptno) in (select MIN(sal),deptno from emp GROUP BY DEPTNO)

1. **Any:表示和子查询的任意一行结果进行比较,有一个满足条件即可**

**< ANY** 表示小于子查询结果集中的任意一个,即小于最大值就可以.

**> ANY** 表示大于子查询结果集中的任意一个,即大于最小值就可以.

**= ANY** 表示等于子查询结果集中的任意一个,即等于任意一个就可以,相当于in.

**例:**-- 查询部门编号不为20,且工资比20部门任意一个名员工工资高的员工编号,姓名,职位,工资.

select empno,ename,job,sal,deptno from emp where sal> ANY(select sal from emp where deptno=20) and deptno!=20

1. **ALL:表示和子查询的所有行结果进行比较,每一行必须满足条件**

**< ALL**:表示小于子查询结果集中的所有行,即小于最小值;

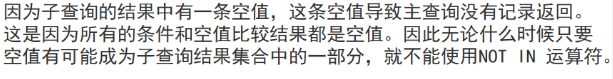
**> ALL**:表示大于子查询结果集中的所有行,即大于最大值;

**= ALL**:表示等于子查询结果集合中的所有行,即等于所有值,通常无意义.

**例**:-- 查询部门标号不为20,且工资比20部门所有员工工资高的员工编号,姓名,职位,工资

select empno,ename,job,sal from emp where deptno !=20 and sal >all(select sal from emp)

**5.4.3 子查询中的空值**



-- **注意**:只要空值有可能成为子查询结果集合中的一部分,就不能使用NOT IN运算符.因为NULL和所有条件比较都是空值.

-- 查询不是领导的员工姓名

select ename from emp where empno not in (select IFNULL(mgr,'') from emp)

* + 1. **在from子句中使用子查询**

**例:**-- 查询比自己部门平均工资高的员工姓名,工资,部门编号,部门平均工资

select a.ename,a.sal,a.deptno,b.AVG from emp a ,(select AVG(sal) avg,deptno from emp GROUP BY deptno) b where a.DEPTNO=b.deptno and a.sal>b.avg

**第八章-视图和索引**

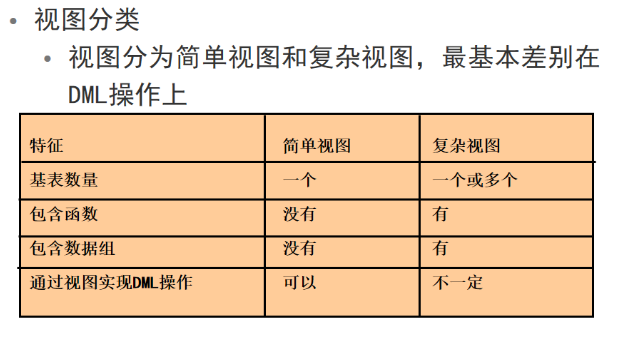
**视图:**

逻辑上来自一个或者多个表的集合(是虚拟的表).

**使用视图:**

* 1. 限制其他用户对数据库表的访问,因为视图可以有选择性的显示数据库表的一部分;
  2. 容易实现复杂的查询;
  3. 对于相同的数据可以产生不同的视图.

**视图分类:**



**创建视图:**

**Create [OR REPLACE] view 视图名**

**as select 字段 from 表 [where 条件]------ OR REPLACE修改表结构**

**例:**-- 创建一个视图查询v-emp20,通过视图只能查看20号部门的员工编号,员工姓名,职位

create view v\_emp20

as select empno,ename,job from emp where deptno=20;

**例:**-- 创建一个视图,通过视图可以查到工资在2000-5000内并且姓名中包含A的员工编号,姓名,工资

CREATE or replace view v\_empsal

as select empno,ename,sal from emp where sal BETWEEN 2000 and 5000 and ename like '%a%';

**例:**-- 在创建视图时,子查询中可以使用别名

-- 创建一个视图,通过视图可以查到员工编号,姓名,员工薪资,入职日期 要求82年之后的员工信息

create view v\_empyear

as select \* from emp where hiredate>'1982-12-31'

**创建复杂视图**

**例:**-- 例:创建一个视图,通过该视图可以查看每个部门的名称,最低工资,最高工资,平均工资

create view V\_emp

as select dname,min(sal),max(sal),avg(sal) from emp e cross join dept d on e.deptno=d.deptno GROUP BY e.deptno

**例:**-- 查询视图v\_emp中 所有信息

select \*from v\_emp;

-- 视图上执行DML操作(会影响基表数据)

-- 删除操作

-- 通过视图删除表中姓名是JONES的员工

delete from v\_emp20 where ename='JONES'

-- 创建视图通过视图可以查询每个部门的最低薪资

create view v\_minsal

as select deptno,min(sal) from emp GROUP BY deptno

-- 修改操作

-- 修改 v\_emp20视图中ADAMS的部门编号改为10

UPDATE v\_emp20 SET DEPTNO=10 WHERE ENAME='ADAMS'

-- 插入操作

-- 向 v\_emp20视图中插入编号为9999姓名为李四,入职日期为2020-1-1,部门编号为20的员工信息

insert into v\_emp20 values (9999,'李四','20')

-- 删除视图:

drop view v\_minsal

-- 创建索引

-- 创建普通索引

create table index1(

id int,

name varchar(20),

sex boolean,

index(id)

);

-- 创建唯一性索引

create table index2(

id int unique,

name varchar(20),

unique index index2\_id(id asc)

);

-- 全文索引

create table index3(

id int,

info varchar(20),

FULLTEXT index index3\_info(info)

)ENGINE=myisam;

-- 创建单列索引

create table index4(

id int,

subject varchar(20),

index index4\_st(subject(10))

);

-- 多列索引

CREATE table index5(

id int,

name varchar(20),

sex char(4),

index index5\_n(name,sex)

);

-- 空间索引

create table index6(

id int,

space geometry not null,

SPATIAL index index6\_sp(space)

)ENGINE=myisam;

-- 删除索引

drop index 索引名 on 表名;

-- 第九章 权限

-- 创建用户

create user 'ttcuser'@'localhost' IDENTIFIED by '123456'

select PASSWORD('123456')

create user 'ttcuser2'@'localhost' IDENTIFIED by '\*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9'

-- 创建用户并且分配权限

grant select,update on \*.\* to 'test1'@'localhost' IDENTIFIED by '123456';

-- 删除用户

drop user 'ttcuser2'@'localhost';

-- 修改用户名

-- rename user 'haha'@'localhost' to 'haha2'@'localhost';

-- 修改密码

SET PASSWORD FOR 'king'@'localhost' =PASSWORD('queen1')

-- 授权

-- 用户grantUser对所有的数据有查询,插入权限,并授予GRANT权限

grant select,insert om \*.\* to 'ttcuser'@'localhost' IDENTIFIED by '123456' with grant option;

-- 第十章