## （p.94）子查询小练习

### 一、查询与ZlotKey相同部门的人的信息

① 查询ZlotKey的部门名

SELECT department\_id

FROM employees

WHERE last\_name='Zlotkey'

②查询部门号=①的姓名

SELECT last\_name ,salary

FROM employees

WHERE department\_id IN

(SELECT department\_id

FROM employees

WHERE last\_name='Zlotkey')

### 二、查询工资比公司平均工资高的员工的员工号、姓名与工资

①查询平均工资

SELECT AVG(salary)

FROM employees

②查询公司>标量的

SELECT last\_name,employee\_id,salary

FROM employees

WHERE salary>(SELECT AVG(salary)

FROM employees )

### 三、查询各部门中工资比本部门平均工资高的员工的员工号、姓名和工资

①查询各部门的平均工资

SELECT department\_id,AVG(salary) AS avgS

FROM employees

GROUP BY department\_id

②将①的结果与employees表进行连接后筛选

SELECT last\_name,employee\_id,salary,e.department\_id

FROM employees e

JOIN (SELECT department\_id,AVG(salary) AS avgS

FROM employees

GROUP BY department\_id) AS AVG

ON e.department\_id=AVG.department\_id

WHERE e.salary>avg.avgS

### 四、查询和姓名中包含字母u的员工在相同部门的员工的员工号和姓名

①查询姓名中包含字母u的员工的部门

SELECT DISTINCT department\_id

FROM employees

WHERE last\_name LIKE '%u%'

②查询部门号=①中任意一个的员工号和姓名

SELECT employee\_id,last\_name

FROM employees

WHERE department\_id IN

(SELECT DISTINCT department\_id

FROM employees

WHERE last\_name LIKE '%u%')

### 五、查询在部门的location\_id为1700的部门工作的员工的员工号

①查询location\_id为1700的部门

SELECT DISTINCT department\_id

FROM departments

WHERE location\_id=1700

②查询部门号=①中的任意一个的员工号

SELECT employee\_id

FROM employees

WHERE employees.department\_id =ANY(SELECT department\_id

FROM departments

WHERE location\_id=1700)

### 六、查询管理者是K\_ing的员工姓名和工资

①查询姓名为K\_ing的员工姓名和工资

SELECT employee\_id

FROM employees

WHERE last\_name='K\_ing'

②查询哪个员工的manage\_id=①

SELECT last\_name,salary

FROM employees

WHERE manager\_id IN(SELECT employee\_id

FROM employees

WHERE last\_name='K\_ing')

### 七、查询工资最高的员工的姓名，要求first\_name和last\_name显示为一列，列名为姓.名

SELECT CONCAT(first\_name,last\_name) AS '姓.名'

FROM employees

WHERE salary=(SELECT MAX(salary)

FROM employees)

## （p.95）分页查询

应用场景：当要显示的数据，一页显示不全，需要分页提交sql请求

语法：在所有查询语句的最后加上LIMIT offset,size

offset为起始索引0开始，size为显示条数；若offset为0，可以省略offset

## （p.96）DQL总结（执行顺序）

select 查询列表 7

from 表 1

【join type join表二 2 产生笛卡尔积

on 连接条件 3 对笛卡尔积进行筛选

where 筛选条件 4 再进行筛选

group by 分组字段 5

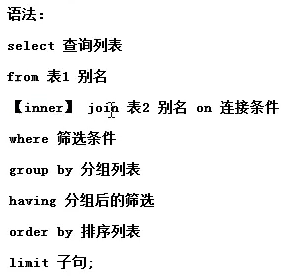
having 分组后的筛选 6

order by 排序后的字段】 8

limit 【offset,】size 9

## （p.97）SQL99总结

### 一、内连接



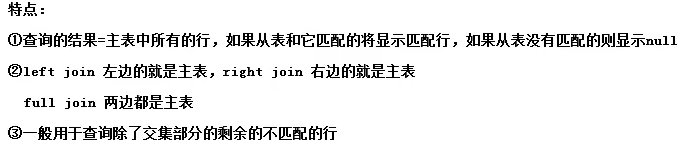
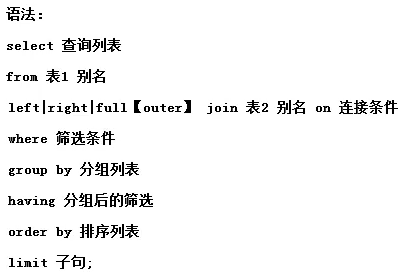
特点：①表的顺序可以调换

②内连接的结果=多表的交集

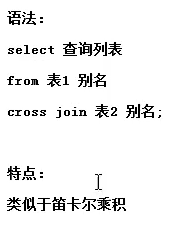
③n表连接至少需要n-1个连接条件

分类：等值、非等值、自连接

### 二、外连接

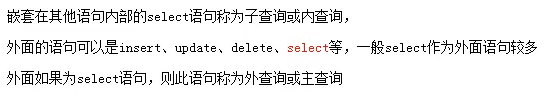


### 三、交叉连接



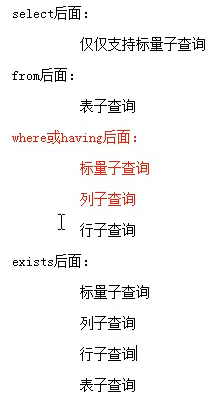
### 四、子查询

1.含义

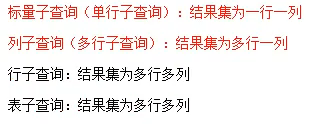


2.分类

①按出现位置

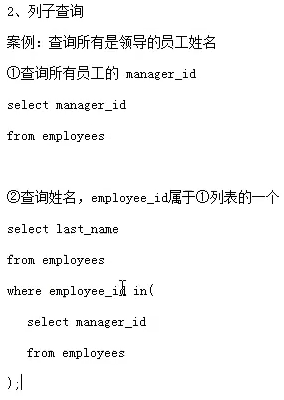


②按结果集的行列



3.示例





## （p.98）练习

### 二、查询平均工资最低的部门信息

#### 方法一：

①各部门的平均工资

SELECT AVG(salary) AS avgs,department\_id

FROM employees

GROUP BY department\_id

②各部门平均工资的最低值

SELECT MIN(avgs)

FROM (SELECT AVG(salary) AS avgs,department\_id

FROM employees

GROUP BY department\_id)AS a

③平均工资等于**最低平均工资**的部门编号

SELECT department\_id

FROM employees

GROUP BY department\_id

HAVING AVG(salary) IN (SELECT MIN(avgs)

FROM (SELECT AVG(salary) AS avgs,department\_id

FROM employees

GROUP BY department\_id)AS a)

④查询出③编号对应的部门信息

SELECT \*

FROM departments

WHERE department\_id=(

SELECT department\_id

FROM employees

GROUP BY department\_id

HAVING AVG(salary) IN (SELECT MIN(avgs)

FROM (SELECT AVG(salary) AS avgs,department\_id

FROM employees

GROUP BY department\_id)AS a)

)

#### 方法二：

①在方法一的②步使用**排序**加limit第一条记录即可得到最低工资

SELECT department\_id

FROM employees

GROUP BY department\_id

ORDER BY AVG(salary)

LIMIT 1

### 三、查询平均工资最低的部门信息及其平均工资

①查询最低平均工资及其部门编号（排序加limit）

SELECT AVG(salary) avgg,department\_id

FROM employees

GROUP BY department\_id

ORDER BY AVG(salary)

LIMIT 1

②使用连接两张表将其并为一行

SELECT d.\*,dd.avgg

FROM (SELECT AVG(salary) avgg,department\_id

FROM employees

GROUP BY department\_id

ORDER BY AVG(salary)

LIMIT 1) dd

INNER JOIN departments d

ON d.department\_id=dd.department\_id

### 五、查询低于公司平均工资的部门编号

SELECT department\_id

FROM employees

GROUP BY department\_id

HAVING AVG(salary)>(SELECT AVG(salary)

FROM employees)

### 六、查询所有manager的详细信息

①查询公司所有manager的员工编号

SELECT DISTINCT manager\_id

FROM employees

②查询对应信息

SELECT \*

FROM employees

WHERE employee\_id IN(SELECT DISTINCT manager\_id

FROM employees)

### 七、查询各部门中最高工资中最低的那个部门的最低工资是多少

①查询各部门最高工资中最低的那个部门

SELECT department\_id

FROM employees

GROUP BY department\_id

ORDER BY MAX(salary)

LIMIT 1

②查询对应部门的最低工资

SELECT MIN(salary)

FROM employees

WHERE department\_id=(SELECT department\_id

FROM employees

GROUP BY department\_id

ORDER BY MAX(salary)

LIMIT 1)

### 八、查询平均工资最高的部门的manager的详细信息

①查询平均工资最高的部门编号

SELECT department\_id

FROM employees

GROUP BY department\_id

ORDER BY MAX(salary) DESC

LIMIT 1

②将departments表和employees表连接

SELECT e.\*

FROM employees e

INNER JOIN departments d

ON e.employee\_id=d.manager\_id

WHERE e.department\_id=(SELECT department\_id

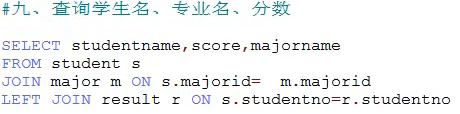
FROM employees

GROUP BY department\_id

ORDER BY MAX(salary) DESC

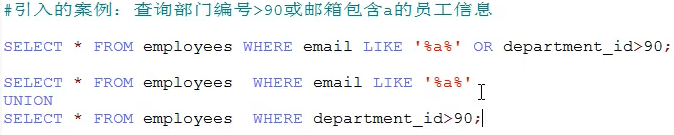
LIMIT 1)

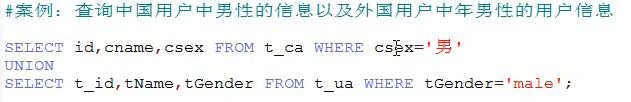
## （p.99）作业

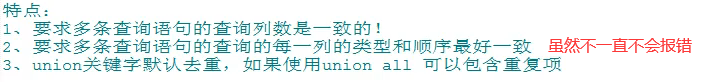


## （p.100-101）联合查询

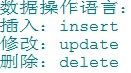








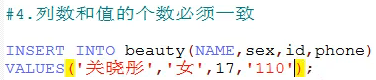
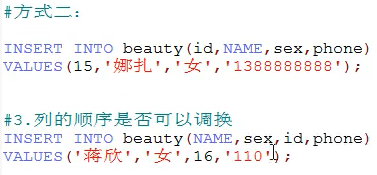
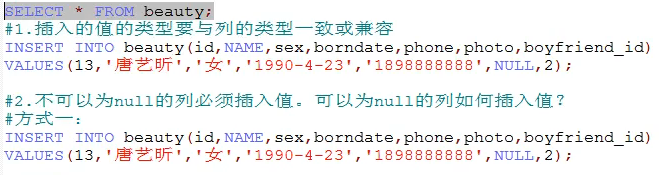
## （p.102）DML语言

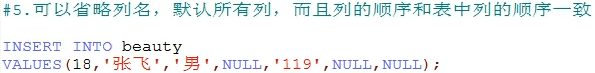


### 插入语句（p.103-104）

#### ①语法一



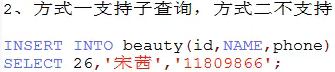
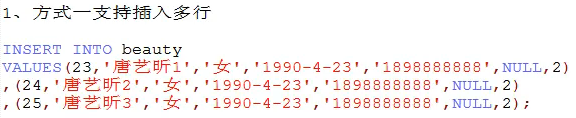




#### ②语法二

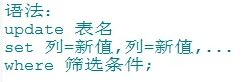


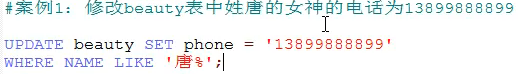
#### ③两种语法对比

\*

### 修改语句（p.105-106）

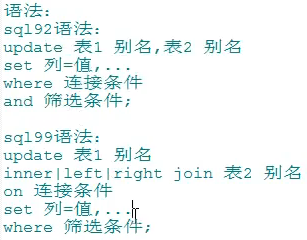
#### ①修改单表的记录

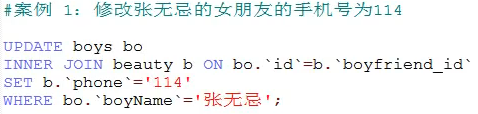


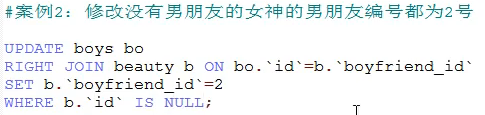




#### ②修改多表的记录







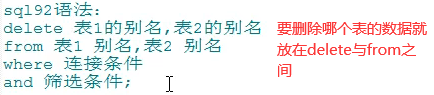
### 删除语句（P.107-110）

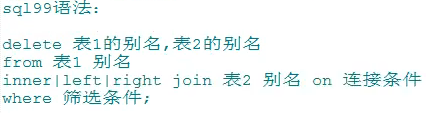
#### ①方式一

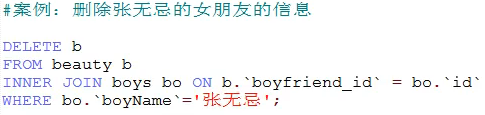


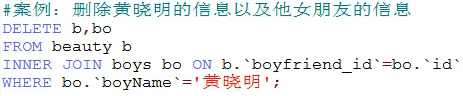


（级联删除）





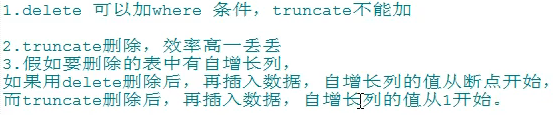




#### ②方式二

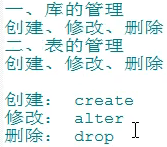
***truncate语句不许加where，也称清空数据***

#### ③两个方式的区别





## （p.111）DDL语言



### 库的管理

#### 库的创建



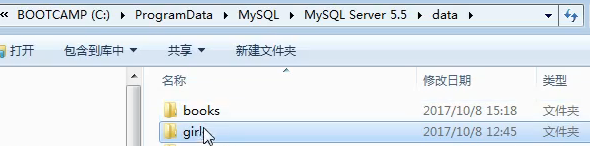


（创建前判断存在与否）

#### 库的修改

①库名修改

一般不改，若一定要改，去mysql的安装目录中停止服务后改变表明对应的文件夹名称



②更改库的字符集（创库时默认utf8）

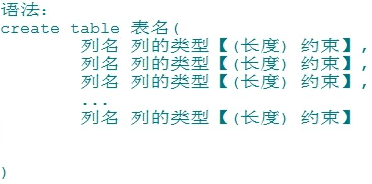


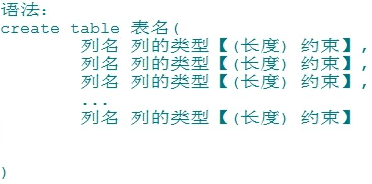
#### 库的删除



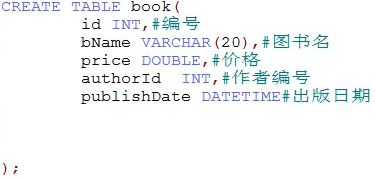
### 表的管理

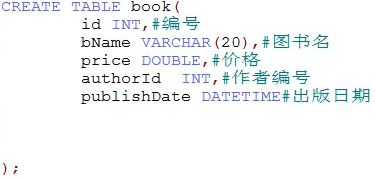
#### 表的创建





（可加if exists）





#### 表的修改

语法：



①修改列名



②修改列的类型或约束



③添加新列



④删除列



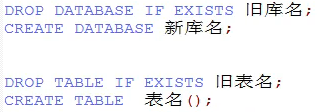
⑤修改表名



#### 表的删除



#### 表创建时通用的写法



#### 表的复制（结构/数据）

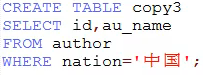
①仅复制表的结构



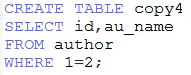
②复制表的结构+数据



③只复制部分数据



④仅复制表的部分结构（不含数据）

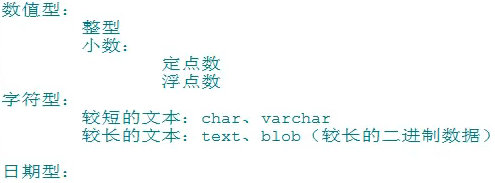


⑤创建一个库，插入别的库的表的数据（跨库）



#### 数据类型

常见的数据类型



选择原则：

所选择的类型越简单越好，能保存数值的类型越小越好

**（1）整型**

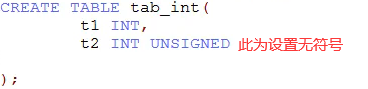


特点：

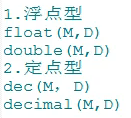
①如果不设置无符号还是有符号，默认为有符号，如果想设置无符号，需要加上unsigned关键字

②如果插入的数值超出了整型的范围，会报out of range异常，并且插入相应的临界值

③如果不设置长度，会有默认的长度；长度代表了显示的最大宽度，使用zerofill关键字在位数不够是前面会用零填充，且zerofill关键字不支持负数（即自带了unsigned）



**（2）小数**



特点：

① M：整数部位与小数部位长度之和；

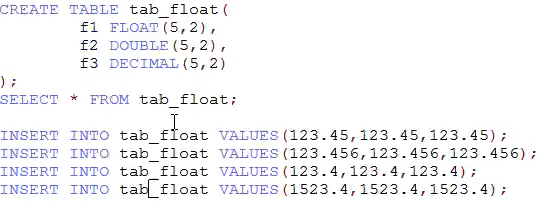
D：小数部位长度

若整数部位超出长度，则显示临界值，若小数部位超出长度，四舍五入至小数长度

② M和D可以省略；若为decimal，M默认为10，D为0；若为float和double，则会根据插入的数值的精度来决定精度

③定点型的精确度较高，如果要求插入数值的精度较高（如货币运算）

***案例一：测试M和D***





**（3）字符型**

①较短的文本：char、varchar

*特点*：



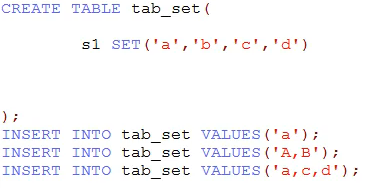
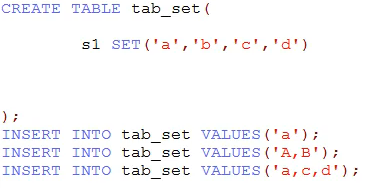
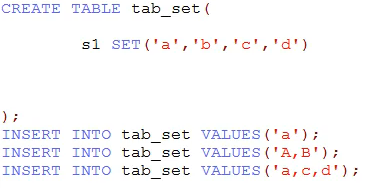
char可以省略M，varchar不可以省略

②较长的文本：text、blob（较大的二进制）

③保存二进制文件：binary、varbinary（类似于①，但只能包含二进制字符串）

④枚举：ENUM（只能插入规定的字符，对大小写无要求；若超出，插入无数据）

⑤SET类型：保存集合，类似于ENUM，内可以保存0~64个成员，SET一次可以选取多个成员



**（4）日期型**

①date(4字节)：只保存日期

②datetime(8字节)：保存日期+时间

③timestamp(4字节): 保存日期+时间，与datetime的区别为，与实际的时区有关系，更能反映实际的日期，datetime只能反映插入时当地的时区；timestamp比datetime的范围小



④time(3字节)：只保存时间

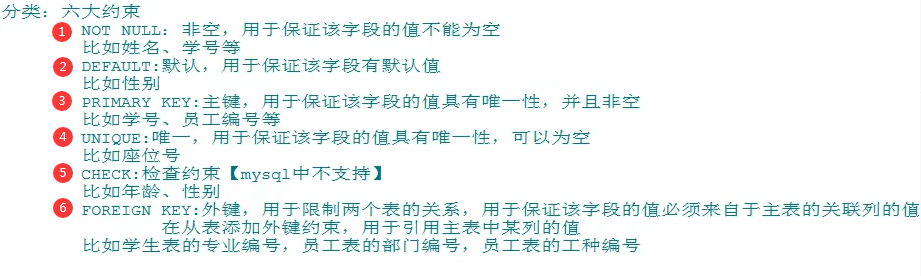
⑤year(1字节)：只保存年

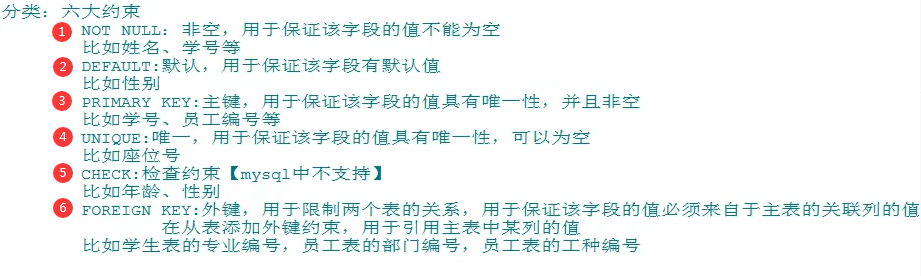
### 约束（P.125）

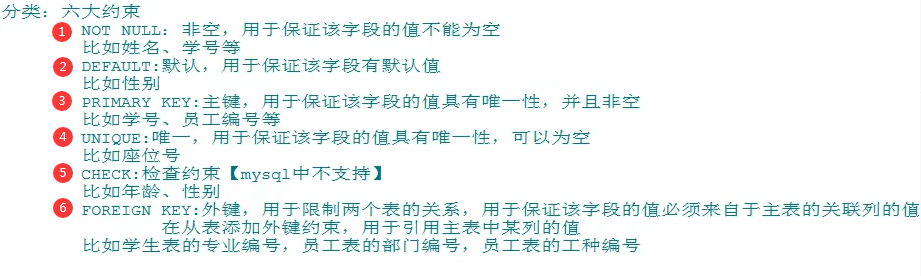
①含义



②分类





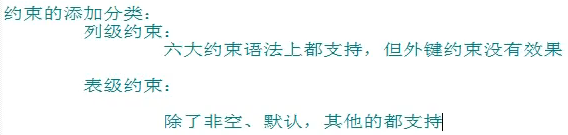


***主键、外键、唯一键会自动生成索引***

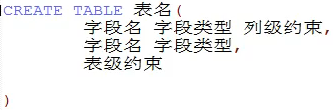
③添加约束的时机



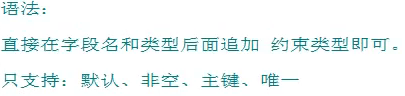
④约束的分类



⑤语法：



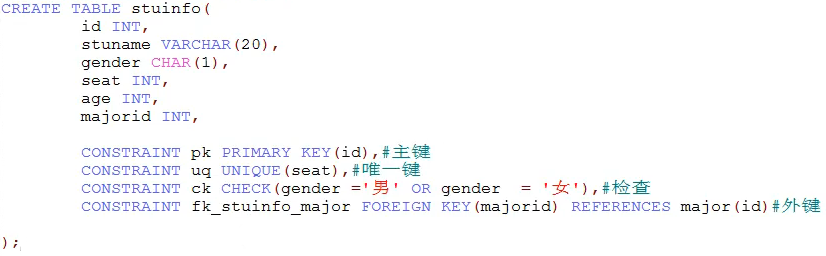
#### 添加列级约束（列级约束不支持外键，但语法不会报错）



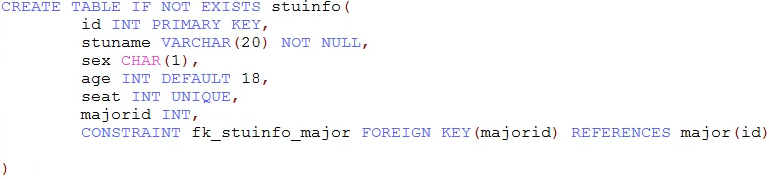


#### 添加表级约束（不支持默认与非空）

***（前面【】内的为可省略）***



#### 通用写法（除了外键为表级约束，其余皆为列级约束）



#### 各约束的不同与细节

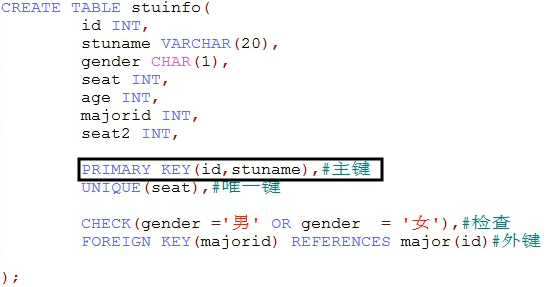
①主键与唯一的对比(***p.128***)



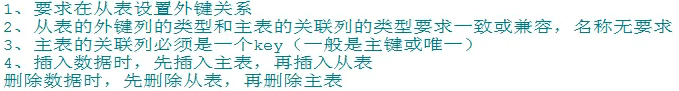
***唯一也只允许当前列中存在一个null***

***可以在表级约束中定义联合主键（即多个列组成一个主键）*如下图**

**若两列为组合主键，那么只有两行数据对应两列皆相同时才算主键重复**



②***外键细节（P.129）***



#### 修改表时添加约束



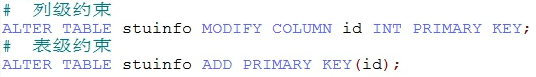
①添加非空约束



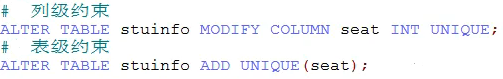
②添加默认约束



③添加主键



④添加唯一键



⑤添加外键



#### 修改表时删除约束P.131

①删除非空约束



②删除默认约束



③删除主键



④删除唯一（删除其对应索引即可）



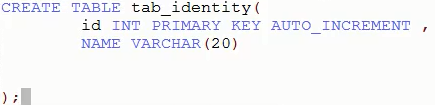
⑤删除外键

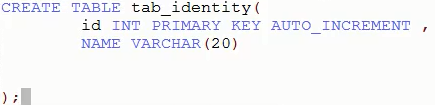




#### 标识列（自增长）P.133

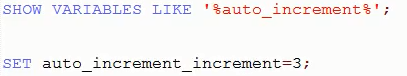
①创建表时设置标识列，可以不用手动插入值



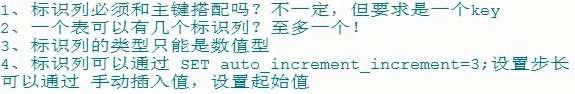


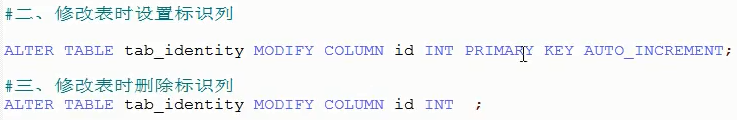


②更改自增长的步长



③特点





## （P.134）TCL语言（事务控制语言）

Transaction Control Language

### 事务的ACID属性

1. 原子性（Atomicity）

事务是一个不可分割的工作单位，要不全部执行，要不全部不执行

1. 一致性（Consistency）

使数据库从一个一致性状态变换到另外一个一致性状态（例如转账，转账前后两人的余额之和必相同）

1. 隔离性（Isolation，第一个字母是i）

一个事务的执行不能被其他事务干扰，即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰

1. 持久性（Durability）

持久性指一个事务一旦被提交，它对数据库中数据的改变是永久性的，接下来的其他操作和数据库故障不应该对其有任何影响。

### 事务的创建

#### 事务的分类：

①隐式事务：事务无明显的开启和结束的标记

例如insert、update、delete都是隐式事务

查看隐式事务是否开启：SHOW VARIABLES LIKE 'autocommit'

②显式事务：事务具有明显的开启和结束的标记

***前提：禁用事务自动提交功能***

Set autocommit=0

此条语句只针对当前的会话有效，即手动事务必每次都运行上条语句

#### 事务的使用：

步骤一：开启事务

Set autocommot=0

Start transaction（此条语句不写也没关系）

步骤二：编写事务中的sql语句（select、insert、update、delete）

语句一；

语句二；

……

步骤三：结束事务（以下二选一）

Commit；提交事务

Rollback；回滚事务

结束事务的意义是：是否将此回话中的任务撤销（Rollback）还是将结果保存至磁盘文件（Commit）

### 数据库的隔离级别

对于同时运行多个事务，这些事务访问数据库中相同的数据时，如果没有采用必要的隔离机制，会导致各种并发问题

#### 事务并发问题

**脏读**：若两个事务T1，T2，T1读取了**已经被T2更新但尚未提交**的数据。若T2回滚，T1读取的数据就是临时且无效的。

**不可重复读**：若T1读取了一个字段，然后T2提交**更新**了该字段。T2再次读取同一个字段，值就不同了。

**幻读**：若T1在插入主键为1的数据前先查询是否主键为1的数据是否存在；在T1查询结束发现主键为1的数据不存在后，T2插入一条主键为1的数据，此时T1再**插入**主键为1的数据就会发生主键冲突问题。（即莫名其妙出现了多的数据）

#### 数据库的事务隔离级别

①oracle支持两种：READ COMMITTED（默认）、SERIALIZABLE

②mysql支持四种：

1. READ UNCOMMITTED(读未提交的数据)

允许事务读取未被其他事务提交的变更，三种问题皆可能出现

2）READ COMMITTED （读已提交的数据）

只允许事务读取已经被其他事务提交的变更，可以避免脏读，但是不可重复读与幻读不可避免

3）REPEATABLE READ（可重复读）

确保事务可以多次从一个字段读取相同的值。在这个事务持续期间，禁止其他事务对这个字段进行更新，可以避免脏读和不可重复读，但幻读存在。（***当一个事务修改表中某行数据时，另一个事务想修改同行数据会阻塞）行锁***

4）SERIALIZABLE（串行化）

确保事务从一个表中读取相同的行。在这个事务持续的期间，禁止其他事务对该表进行插入、更新和删除操作，避免了所有并发问题，但性能低下。***（当一个事务在修改一张表中数据时，另一张表要修改此表数据会阻塞）表锁***

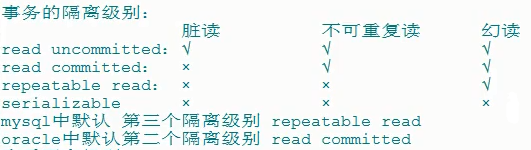
### 事务的演示（p.137）

#### 查看/修改事务隔离级别（事务隔离级别基于会话，即每次创建一个会话都需要改成自己需要的隔离级别）

①查看：SELECT @@tx\_isolation

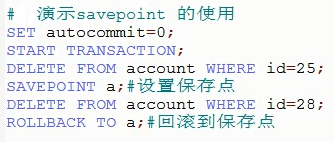
②修改：Set global/session transaction isolation level read uncommitted

加上global是设置整个数据库全局的连接，session只是针对当前会话



### 事务的保存点（savepoint）（p.138）

搭配rollback使用



### delete与truncate在事务使用时的区别

delete可以被rollback；在一个事务中若使用truncate，就算回滚，数据也没了。

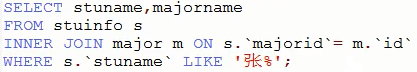
## （P.139）视图（虚拟表）

Mysql5.1后出现的新特性，是通过表动态生成的数据。视图只保存sql逻辑，不保存查询结果。

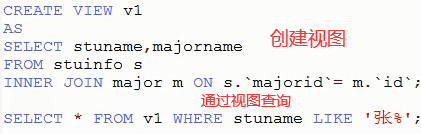
### 介绍

案例：查询姓张的学生名和专业名

1. 使用联合查询



1. 使用视图



### 视图的创建（p.140）

#### 语法：

Create view 视图名

As

查询语句

#### 案例：

查询有相中包含a字符的员工名、部门名和工种信息

**①创建视图**

CREATE VIEW myv1

AS

SELECT last\_name,department\_name,job\_title

FROM employees e

JOIN departments d ON e.department\_id=d.department\_id

JOIN jobs j ON e.job\_id=j.job\_id

**②从条件查询结果**

SELECT \* FROM myv1 WHERE last\_name LIKE 'a%'

#### 视图的好处：

①重用sql语句

②简化了复杂的sql操作，不必知道细节

③保护数据，提高安全性

### 视图的修改（p.141）

#### 方式一：

语法：create or replace view 视图名

as

查询语句；

#### 方式二：

语法：alter view 视图名

as

查询语句；

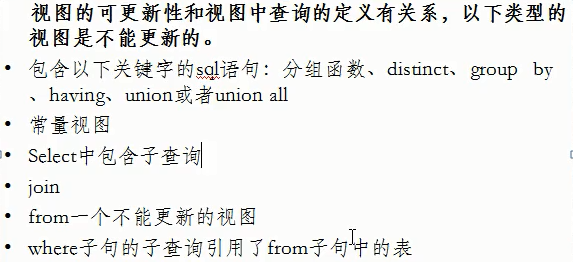
### 删除视图（p.142）

语法：drop view 视图名,视图名,…；（可批量删除）

### 查看视图

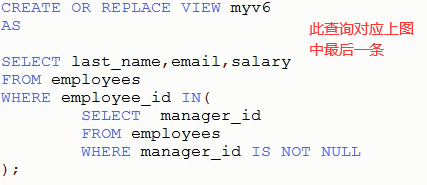
①Desc 视图名 ②show create view myv3

### 视图的更新



①上述条件以外的视图可以进行增删改数据，且同时改变原表中的数据。

②一般视图都会设置权限不允许修改。



### 视图与表的区别



## （P.149）变量

系统变量：

全局变量

会话变量

自定义变量：

用户变量

局部变量

### 系统变量

变量由系统提供，不是用户定义，属于服务器层面。

#### 全局变量与会话变量的区别

全局变量：

作用域：服务器每次启动将为所有的全局变量赋初始值，针对于所有的会话有效，但重启后恢复默认（想要永久修改全局变量，需要修改配置文件）

会话变量：

作用域：仅仅针对于当前会话（连接）有效

#### 语法：

①查看所有的**全局/会话**的系统变量

Show global |【session】 Variables

②查看满足条件的部分系统变量

Show global |【session】 Variables like ‘%char%’

③查看指定的某个系统变量的值

Select @@global |【session】.系统变量名

***例子*：select @@global.autocommit**

④为某个系统变量赋值

方式一：

Set global |【session】 系统变量名=值

***例子*：set session tx\_isolation =’read-uncommitted’**

方式二：

Set @@global |【session】.系统变量名=值

***例子*：set @@session.tx\_isolation =’read-committed’**

**tip**: 加上global代表改变全局变量，不写或加上session表示会话变量

### 自定义变量

说明：变量是用户自定义的，不是有系统的

使用步骤：

声明、赋值、使用（查看、比较、运算等）

#### 用户变量

作用域：针对当前会话（连接）有效，等同于会话变量的作用域

**①声明并初始化**

方式一：set @用户变量名=值

方式二：set @用户变量名:=值

方式三：select @用户变量名:=值

**tip**：set支持=与:=，select仅支持:=

**②赋值（更新）**

**方式一**：通过set或select

Set @用户变量名=值

set @用户变量名:=值

select @用户变量名:=值

***例子***： set @name=’john’ #初始化；Set @name=100 #重新赋值

**方式二**：通过select into

Select 字段 into @变量名 from 表；（必须是一个值）

***例子***： select count(\*) into @count from employees;（将employees表的行数放入变量）

**③使用（查看用户变量的值）**

Select @用户变量名;

#### 局部变量

**作用域**：仅仅在定义它的begin end中有效（存储过程）且必须为begin end中第一句。

**①声明：**

Declare 变量名 类型；（仅定义类型与变量）

Declare 变量名 类型 default 值；（定义后初始化）

**②赋值**

**方式一**：通过set或select

Set局部变量名=值

set 局部变量名:=值

select @局部变量名:=值

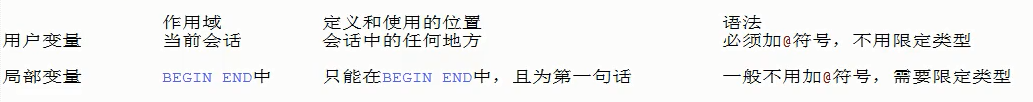
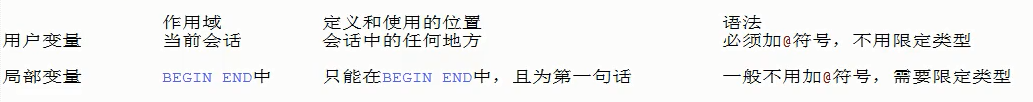
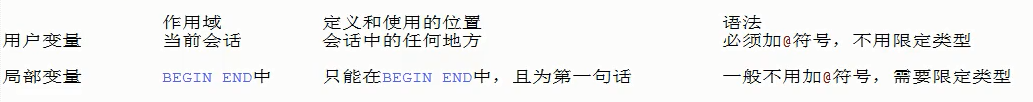
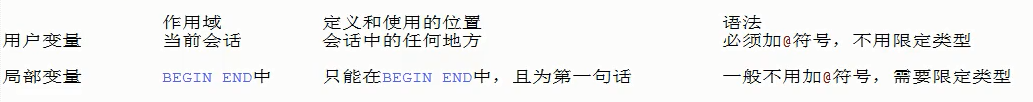
**方式二**：通过select into

Select 字段 into 局部变量名 from 表；（必须是一个值）

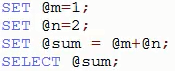
**③使用**

Select 局部变量名;

#### 用户变量与局部变量的对比（p.154）



例子（用户变量）：



## （p.155）存储过程

存储过程与函数类似于java中的方法

**好处**：提高代码的重用性；简化操作；减少了编译次数与减少连接数据库服务器的次数

存储过程：

一组预先编译好的sql语句的集合，理解成批处理语句

### 语法

#### 创建语法

Create procedure 存储过程名(参数列表)

Begin

存储过程体（一组合法有效的sql语句）

End

**tip**：① 参数列表包含三部分

参数模式 参数名 参数类型

***例子***：in stuname varchar(20)

参数模式：in（该参数可以作为输入，需要调用方传入值）、out（该参数作为输出，该参数可以作为返回值）、inout（该参数既可以作为输入又可以作为输出，也就是该参数既需要传入值，也可以返回值）

② 若存储过程仅仅只有一句话，begin与end可以省略；存储过程体中每条sql语句的结尾必须要加分号。存储过程的结尾可以使用delimiter重新设置

结束标记：Delimiter $

#### 调用语法

call存储过程名(实参列表);

#### 删除语法

drop procedure 存储过程名;（不支持删除多条）

#### 查看存储过程的信息

show create procedure myp2;

### 二、案例

#### 1. 空参列表的：

插入到admin表中五条记录（p.157）

***#建立存储过程***

DELIMITER $

CREATE PROCEDURE myp1()

BEGIN

INSERT INTO admin(username,PASSWORD)

VALUES('xx','0000'),('qq','0000'),('ww','0000'),('ee','0000'),('xx','0000');

END $

***#运行存储过程***

CALL myp1() $

#### 2.带in模式参数的

***例子1***：根据女神名，查询对应的男神信息

***#建立存储过程***

DELIMITER $

CREATE PROCEDURE myp2(IN beautyName VARCHAR(20))

BEGIN

SELECT bo.boyName

FROM boys bo

RIGHT JOIN beauty b ON b.boyfriend\_id=bo.id

WHERE b.name=beautyName;

END $

***#运行存储过程***

CALL myp2('热巴')

***例子2***：判断用户登录是否成功

***#建立存储过程***

CREATE PROCEDURE myp3(IN username VARCHAR(20),IN PASSWORD VARCHAR(20))

BEGIN

DECLARE result INT DEFAULT 0;

SELECT COUNT(\*)INTO result

FROM admin

WHERE admin.username=username AND admin.password=PASSWORD;

SELECT IF(result>0,'成功','错误');

END $

***#运行存储过程***

Call myp3(‘qq’,’0000’)$

#### 带out模式参数的（p.159）

***例子1***：根据女神名，返回对应的男神名（一个out）

***#建立存储过程***

CREATE PROCEDURE myp4(IN beautyName VARCHAR(20),OUT boyName VARCHAR(20))

BEGIN

SELECT bo.boyName INTO boyName

FROM boys bo

INNER JOIN beauty b ON bo.id=b.boyfriend\_id

WHERE b.name=beautyName;

END

***#运行存储过程***

set @bname$

call myp4('热巴',@bname)$

select @bname$

***例子2***：根据女神名，返回对应的男神名和男神魅力值（多个out）

***#建立存储过程***

CREATE PROCEDURE myp5(IN beautyName VARCHAR(20),OUT boyName VARCHAR(20),OUT userCP VARCHAR(20))

BEGIN

SELECT bo.boyName,bo.userCP INTO boyName,userCP

FROM boys bo

INNER JOIN beauty b ON bo.id=b.boyfriend\_id

WHERE b.name=beautyName;

END

***#运行存储过程***

call myp5('热巴',@bName,@userCP)$

select @bName,@userCP$

#### 带inout模式参数的（p.160）

***例子1***：传入a和b两个值，最终a和b都翻倍返回

***#建立存储过程***

CREATE PROCEDURE myp6(INOUT a INT,INOUT b INT)

BEGIN

SET a=a\*2;

SET b=b\*2;

END

***#运行存储过程***

set @a=5$

set @b=10$

call myp6(@a,@b)$

select @a,@b$

#### 小练习

①将日期转换成指定格式：

CREATE PROCEDURE myp9(IN birth1 DATETIME,IN birth2 DATETIME,OUT result INT)

BEGIN

SELECT DATEDIFF(birth1,birth2)INTO result;

END$

②输入女神名，返回女神名 and 男神名

***#建立存储过程***

CREATE PROCEDURE test(INOUT myname VARCHAR(50))

BEGIN

SELECT CONCAT(b.name,' and ',IFNULL(boyName,'null')) INTO myname

FROM beauty b

JOIN boys bo ON bo.id=b.boyfriend\_id

WHERE b.name=myname;

END$

***#运行存储过程***

set @name='热巴'$

call test(@name);

select @name;

## （p.165）函数

存储过程：可以有0个返回，也可以有多个返回，适合做批量插入与批量更新

函数：有且仅有一个返回，适合做处理数据后返回一个结果

### 一、创建语法

Create function 函数名（参数列表） returns 返回类型

Begin

函数体

End

***注意***：1.参数列表包含两个部分：参数名 参数类型

2.函数体必须要有return语句，没有会报错；return不放在函数体最后也不会报错但是不建议此做法。

3.函数体只有一句话，可以省略begin end

4.需要delimiter语句设置结束标记

### 二、调用语法

select 函数名（参数列表）

### 查看与删除语法（p.169）

①查看：Show create function myf3;

②删除：DROP FUNCTION myf2

### 案例演示

#### 无参有返回：

返回公司员工个数

CREATE FUNCTION myf1() RETURNS INT

BEGIN

DECLARE c INT DEFAULT 0;

SELECT COUNT(\*) INTO c

FROM employees;

RETURN c;

END$

#### 有参有返回

①根据员工名，返回他的工资

CREATE FUNCTION myf2(NAME VARCHAR(20)) RETURNS DOUBLE

BEGIN

DECLARE result DOUBLE DEFAULT 0;

SELECT salary INTO result

FROM employees

WHERE last\_name=NAME;

RETURN result;

END$

②传入两个float，返回二者之和

create function myf3(num1 float,num2 float)returns float

begin

declare result float default 0;

set result=num1+num2;

return result;

end$

## （p.170）控制流程结构

顺序结构：程序从上往下依次执行

分支结构：程序从两条或多条路径中选择一条去执行

循环结构：程序在满足一定条件的基础上，重复执行一段代码

### 一、if函数与case结构

1. if函数

if(表达式1，表达式2，表达式3)若表达式1为ture，返回2，不成立返回3

2.case结构

情况1：类似于switch（若为语句要加分号）

case 变量|表达式|字段

when 要判断的值 then 返回的值1或语句1;

when 要判断的值 then 返回的值2或语句2;

…

else 要返回的值n或语句n;

end case;

情况2：多重区间判断

case

when 要判断的条件1 then 返回的值1或语句1;

when 要判断的条件2 then 返回的值2或语句2;

…

else 要返回的值n或语句n;

end case;

特点：

①若作为表达式，嵌套在其他语句中使用，可以放在任何地方

若作为独立的语句，只能放在begin end中

②如果when中的值满足或条件成立，则执行对应的then后面的语句，并且结束case

如果都不满足，则执行else中的语句或值

③else可以省略，如果else省略了，并且所有when条件都不满足，返回null

***例子***：创建存储过程，根据传入的成绩显示等级，传入的成绩：90-100显示A，80-90显示B，60-80显示C，否则显示D

CREATE PROCEDURE myp1(IN score INT)

BEGIN

CASE

WHEN score>=90 AND score<=100 THEN SELECT 'A';

WHEN score>=80 THEN SELECT 'B';

WHEN score>=60 THEN SELECT 'C';

ELSE SELECT 'D';

END CASE;

END$

### 二、if结构

实现多重分支

#### 语法

if 条件1 then 语句1；

elseif 条件2 then 语句2；

…

【else 语句n；】此结构只能放在begin end中

end if；

#### 案例

根据传入的成绩显示等级，传入的成绩：90-100返回A，80-90返回B，60-80返回C，否则返回D

CREATE FUNCTION myf2(score INT) RETURNS CHAR

BEGIN

IF score>=90 AND score<=100 THEN RETURN 'A';

ELSEIF score>=80 THEN RETURN 'B';

ELSEIF score>=60 THEN RETURN 'C';

ELSE RETURN 'D';

END IF;

END$

### 三、循环结构（必须放在begin end中）

分类：while、loop、repeat

循环控制：iterate：类似于java中continue，结束本次循环

leave：类似于break，结束当前所在循环

#### while

语法：【标签：】while 循环条件 do

循环体

end while【标签】；加了标签才可以使用循环控制关键字

#### loop

语法：【标签：】loop

循环体

end loop【标签】；无循环条件、要终止必须使用循环控制关键字，可以模拟死循环

#### repeat（类似于do…while，但是条件是终止条件）

语法：【标签：】repeat

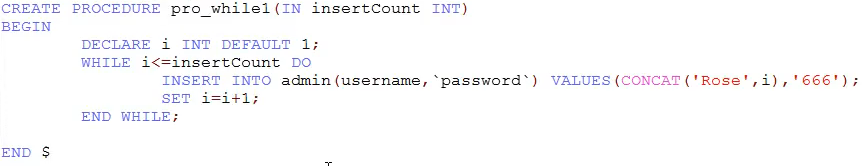
循环体

until 结束循环的条件

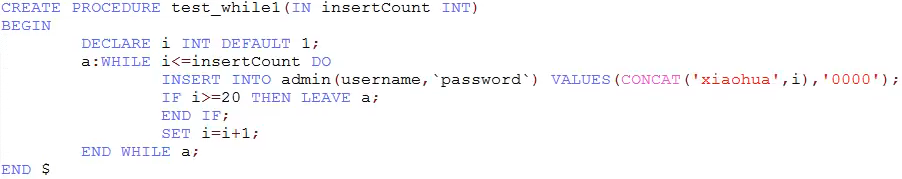
end repeat 【标签】；

#### 案例

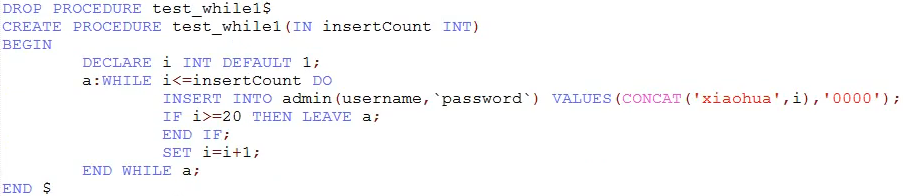
①批量插入，根据次数插入到admin表中多条记录



②添加leave：批量插入，根据次数插入到admin表中多条记录（若大于20则停止）



③添加iterate语句：批量插入，根据次数插入到admin表中多条记录（仅插入偶数次）



④生成指定个数的随机字符串

