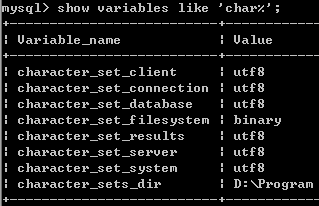
## 【学习目标】

1. 能够操纵数据库表中数据
2. 能够灵活查询数据库表中数据
3. 能够授予权限给予特定用户
4. 能够修改MySQL用户密码
5. 能够阐述什么是数据约束
6. 能够给表添加常见约束
7. 能够解释什么是外键
8. 能够运用外键

# 、一、编码

###### 1　查看MySQL编码

**SHOW VARIABLES LIKE 'char%';**



因为当初安装时指定了字符集为UTF8，所以所有的编码都是UTF8。

* character\_set\_client：你发送的数据必须与client指定的编码一致！！！服务器会使用该编码来解读客户端发送过来的数据；
* character\_set\_connection：通过该编码与client一致！该编码不会导致乱码！当执行的是查询语句时，客户端发送过来的数据会先转换成connection指定的编码。但只要客户端发送过来的数据与client指定的编码一致，那么转换就不会出现问题；
* character\_set\_results：响应的编码，即查询结果返回给客户端的编码。这说明客户端必须使用result指定的编码来解码；

###### 2　控制台编码

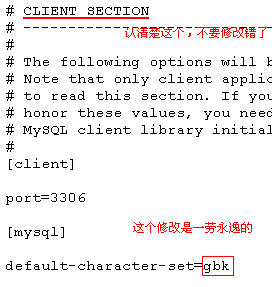
修改character\_set\_client、character\_set\_results、character\_set\_connection为GBK，就不会出现乱码了。但其实只需要修改character\_set\_client和character\_set\_results。

控制台的编码只能是GBK，而不能修改为UTF8，这就出现一个问题。客户端发送的数据是GBK，而character\_set\_client为UTF8，这就说明客户端数据到了服务器端后一定会出现乱码。既然不能修改控制台的编码，那么只能修改character\_set\_client为GBK了。

服务器发送给客户端的数据编码为character\_set\_result，它如果是UTF8，那么控制台使用GBK解码也一定会出现乱码。因为无法修改控制台编码，所以只能把character\_set\_result修改为GBK。

* 修改character\_set\_client变量：**set character\_set\_client=gbk;**
* 修改character\_set\_results变量：**set character\_set\_results=gbk;**

设置编码只对当前连接有效，这说明每次登录MySQL提示符后都要去修改这两个编码，但可以通过修改配置文件来处理这一问题：配置文件路径：D:\Program Files\MySQL\MySQL Server 5.1\ my.ini



###### 3　MySQL工具

使用MySQL工具是不会出现乱码的，因为它们会每次连接时都修改character\_set\_client、character\_set\_results、character\_set\_connection的编码。这样对my.ini上的配置覆盖了，也就不会出现乱码了。

# 二、MySQL数据库备份与还原

###### 1　生成SQL脚本

在控制台使用mysqldump命令可以用来生成指定数据库的脚本文本，但要注意，脚本文本中只包含数据库的内容，而不会存在创建数据库的语句！所以在恢复数据时，还需要自已手动创建一个数据库之后再去恢复数据。

|  |
| --- |
| mysqldump –u用户名 –p密码 数据库名>生成的脚本文件路径 |



现在可以在C盘下找到mydb1.sql文件了！

注意，mysqldump命令是在Windows控制台下执行，无需登录mysql！！！

###### 2　执行SQL脚本

执行SQL脚本需要登录mysql，然后进入指定数据库，才可以执行SQL脚本！！！

执行SQL脚本不只是用来恢复数据库，也可以在平时编写SQL脚本，然后使用执行SQL 脚本来操作数据库！大家都知道，在黑屏下编写SQL语句时，就算发现了错误，可能也不能修改了。所以我建议大家使用脚本文件来编写SQL代码，然后执行之！

|  |
| --- |
| SOURCE C:\mydb1.sql |



　　注意，在执行脚本时需要先行核查当前数据库中的表是否与脚本文件中的语句有冲突！例如在脚本文件中存在create table a的语句，而当前数据库中已经存在了a表，那么就会出错！

还可以通过下面的方式来执行脚本文件：

mysql -uroot -p123 mydb1<c:\mydb1.sql

|  |
| --- |
| mysql –u用户名 –p密码 数据库<要执行脚本文件路径 |



这种方式无需登录mysql！

# 二、数据约束

## 2.1 概念

可以给表添加约束条件，为了保证数据的完整性！！！

### 1）主键

|  |
| --- |
| -- 3.4 主键（非空+唯一）(PRIMARY KEY)  主键具备的特点：  1. 被引用  2. 唯一  3. 非空。  -- 主键设置规则：  -- 1）以后在设计表的时候，尽量给每张表设置一个主键字段，这个主键字段用于标记表数据的唯一性！  -- 2）不建议使用表的业务字段作为主键字段，而是独立添加一个叫id字段作为主键字段！  CREATE TABLE staff(  id INT PRIMARY KEY,  NAME VARCHAR(20),  gender CHAR(1)  );  INSERT INTO staff(id,NAME,gender) VALUES(1,'狗娃','男');  INSERT INTO staff(id,NAME,gender) VALUES(1,'狗蛋','男'); -- Duplicate entry '1' for key 'PRIMARY'  INSERT INTO staff(id,NAME,gender) VALUES(2,'狗蛋','男');  INSERT INTO staff(id,NAME,gender) VALUES(NULL,'狗剩','男'); -- Column 'id' cannot be null |

**添加主键：**

ALTER TABLE person ADD PRIMARY KEY(id);

**删除主键:**

ALTER TABLE person DROP PRIMARY KEY;

### 2）自增长约束

|  |
| --- |
| -- 3.5 自增长( AUTO\_INCREMENT )  -- 自增值是为了维护主键值（数据库生成主键值）  -- 效果：从1开始增值，每次+1  CREATE TABLE staff(  id INT PRIMARY KEY AUTO\_INCREMENT,  NAME VARCHAR(20),  gender CHAR(1)  );  INSERT INTO staff(NAME,gender) VALUES('狗娃','男');  INSERT INTO staff(NAME,gender) VALUES('狗蛋','男');  INSERT INTO staff(id,NAME,gender) VALUES(2,'狗蛋','男');  DELETE \* FROM staff WHERE id=2;  SELECT \* FROM staff;  -- DELETE FROM: 这种删除不会影响自增长约束 (用的多)  -- truncate table: 这种删除会影响自增长约束(必须是全表删除)（用的少）  TRUNCATE TABLE staff; |

修改：

ALTER TABLE person MODIFY id INT AUTO\_INCREMENT;

### 3）默认值

|  |
| --- |
| -- 3.1 默认值（default）  -- 需求：给gender添加默认值  -- 生效： 当不赋值的时候生效！  -- 注意：默认值可以给null  CREATE TABLE staff(  id INT,  NAME VARCHAR(20),  gender CHAR(1) DEFAULT '男'  );  INSERT INTO staff(id,NAME) VALUES(1,'狗娃');  INSERT INTO staff(id,NAME,gender) VALUES(2,'狗剩','女');  INSERT INTO staff(id,NAME,gender) VALUES(3,'狗蛋',NULL); |

### 4）非空

|  |
| --- |
| -- 3.2 非空（not null）  -- 需求：name不能出现null  -- 生效：  -- 1）当不赋值值的时候  -- 2）给null  CREATE TABLE staff(  id INT,  NAME VARCHAR(20) NOT NULL,  gender CHAR(1)  );  INSERT INTO staff(id,gender) VALUES(1,'男');  INSERT INTO staff(id,NAME,gender) VALUES(2,NULL,'女'); -- Column 'NAME' cannot be null |

### 5）唯一

|  |
| --- |
| -- 3.3 唯一(unique)  -- 需求： id值唯一  -- 生效： unique  -- 1）当数据重复  -- 注意：  -- 唯一约束不能限制null  CREATE TABLE staff(  id INT UNIQUE,  NAME VARCHAR(20),  gender CHAR(1)  );  INSERT INTO staff(id,NAME,gender) VALUES(1,'狗娃','男');  INSERT INTO staff(id,NAME,gender) VALUES(1,'狗蛋','男'); -- Duplicate entry '1' for key 'id'  INSERT INTO staff(id,NAME,gender) VALUES(2,'狗蛋','男');  INSERT INTO staff(id,NAME,gender) VALUES(NULL,'狗剩','男');  INSERT INTO staff(id,NAME,gender) VALUES(NULL,'铁蛋','男'); |

### 6）外键

作用：约束两种表的数据的完整性

问题1： 员工表的部门名称出现数据冗余！！！

解决问题1： 独立设计一张部门表，把部门名称放到部门表，员工表设置一个部门id关联部门。

|  |
| --- |
| -- 员工表  CREATE TABLE employee(  id INT PRIMARY KEY AUTO\_INCREMENT,  NAME VARCHAR(20) NOT NULL,  deptId INT -- dpetId:部门表的id  );  -- 部门表  CREATE TABLE dept(  id INT PRIMARY KEY AUTO\_INCREMENT,  deptName VARCHAR(20)  ); |

问题2：添加员工表的部门id时，可能会添加非法数据（不存在部门id）

解决问题2：给员工表的部门id设置外键

|  |
| --- |
| -- 员工表  CREATE TABLE employee(  id INT PRIMARY KEY AUTO\_INCREMENT,  NAME VARCHAR(20) NOT NULL,  deptId INT,  CONSTRAINT employee\_dept\_fk FOREIGN KEY(deptId) REFERENCES dept(id)  -- 外键名称 外键字段 参考表（主键字段）  -- 设置deptId为外键，deptId的值参考dept的id值  );  -- 部门表  CREATE TABLE dept(  id INT PRIMARY KEY AUTO\_INCREMENT,  deptName VARCHAR(20)  ); |

注意：

|  |
| --- |
| -- 外键注意事项：  -- 当设置了外键后  -- 1）添加数据时，先添加主表数据，再添加副表数据  -- 2）修改数据时，先修改副表数据，再修改主表数据(先让副表没有主表的引用数据)  -- 3）删除数据时，先删除副表数据，再删除主表数据(先让副表没有主表的引用数据) |

### 7）级联操作

级联操作是外键的一个特征。

可以给外键添加级联操作

作用：通常情况下，设置了外键，操作顺序，先修改或删除副表数据，再修改或删除主表数据，但是开发者希望直接修改或删除主表数据，从而影响副表数据，这时就可以使用级联操作。

|  |
| --- |
| -- 员工表（副表/从表）  CREATE TABLE employee(  id INT PRIMARY KEY AUTO\_INCREMENT,  NAME VARCHAR(20) NOT NULL,  deptId INT,  CONSTRAINT employee\_dept\_fk FOREIGN KEY(deptId) REFERENCES dept(id) ON UPDATE CASCADE ON DELETE CASCADE  -- 给外键添加级联修改和级联删除 on update cascade  );  -- 部门表（主表）  CREATE TABLE dept(  id INT PRIMARY KEY AUTO\_INCREMENT,  deptName VARCHAR(20)  ); |

# 2、表关系

表之间是什么关系，取决于业务！！！！

## 2.1 一对多（频率最高）

用户

商品

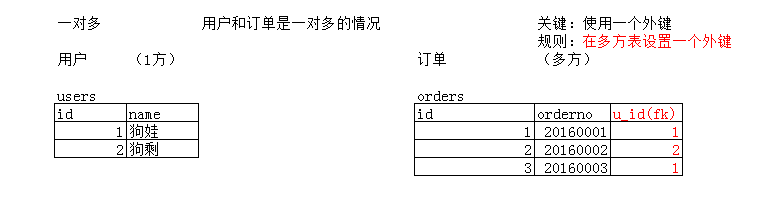
订单

商品分类

业务： 一个订单必须由一个用户下的，一个用户可以下多个订单

用户（1方） 订单（多方） （一对多）

商品分类（1方） 商品（多方）



CREATE TABLE dept(

id INT PRIMARY KEY,

NAME VARCHAR(6)

);

INSERT INTO dept(id,NAME) VALUES(1,"教学部");

INSERT INTO dept(id,NAME) VALUES(2,"学工部");

CREATE TABLE emp(

id INT PRIMARY KEY,

NAME VARCHAR(6),

deptNo INT,

CONSTRAINT dept\_emp\_fk FOREIGN KEY(deptNo) REFERENCES dept(id)

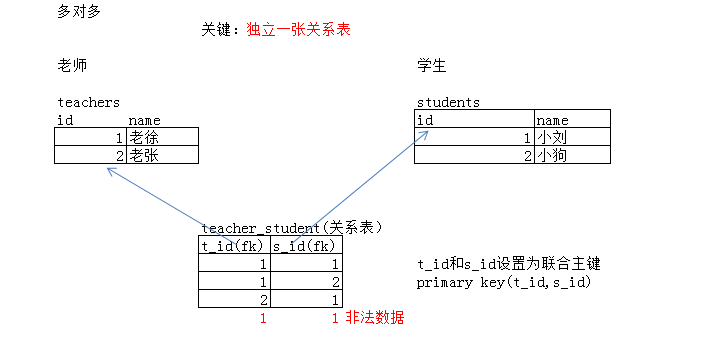
)

## 2.2 多对多（频率也高）

业务：一个学生可以多门课程，一门课程可以被多个学生选择！！

学生(多方) 课程（多方）

老师（多方） 学生（多方）



CREATE TABLE student(

id INT PRIMARY KEY ,

NAME VARCHAR(6)

);

INSERT INTO student VALUES(1,"张三 ");

INSERT INTO student VALUES(2,"李四 ");

CREATE TABLE teacher(

id INT PRIMARY KEY ,

NAME VARCHAR(6)

);

INSERT INTO teacher VALUES(1,"老一");

INSERT INTO teacher VALUES(2,"老二");

CREATE TABLE stu\_tea(

student\_id INT ,

teacher\_id INT ,

PRIMARY KEY(student\_id,teacher\_id)

);

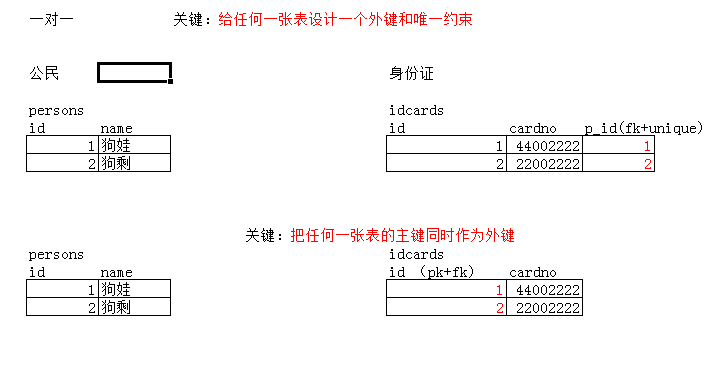
INSERT INTO stu\_tea VALUES(1,1);

INSERT INTO stu\_tea VALUES(1,1);

## 2.3 一对一（最少见）

业务：一个公民只有一张身份证，一个身份证属于一个公民

公民（１方） 身份证（１方）



CREATE TABLE person(

id INT PRIMARY KEY ,

NAME VARCHAR(6)

);

INSERT INTO person VALUES(1,'狗娃');

INSERT INTO person VALUES(2,'狗剩');

CREATE TABLE itcard(

id INT PRIMARY KEY,

idCard INT,

personId INT UNIQUE ,

CONSTRAINT id\_person\_fk FOREIGN KEY(personid) REFERENCES person(id)

);

# 5、数据库设计

数据库设计： 如何把表设计好！！！！！

数据库设计尽量遵守数据库三大范式！！！！

## 5.1 第一范式

每个表的每个字段都必须是(从业务角度)不可分割的独立单元！！！

student: id name -- 违反第一范式

1 张三|狗娃

2 张锋|小三

查询不方便： 查询现有名包含“三”的学生

select \* from student where name like '%三%';

student: id name oldname -- 符合第一范式

1 张三 狗娃

2 张锋 小三

## 5.2 第二范式

在第一范式的基础上，不同业务含义的字段应该划分到不同表中。

（一张表表达一个意思）

student: id name id name -- 违反第二范式

1 张三

2 李四

1 老张

2 老刘

student: id name -- 符合第二范式

1 张三

2 李四

teacher: id name

1 老张

2 老刘

## 5.3 第三范式

在第二范式的基础上，一张表的主键除外的其他字段都应该跟主键字段是直接决定关系！

简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。例如，存在一个部门信息表，其中每个部门有部门编号（dept\_id）、部门名称、部门简介等信息, 那么员工表除了部门表的主键之外的其他字段都不应该出现在员工表中。

员工表：

编号 姓名 薪资

部门表

编号 名称 简介

# 3、多表查询（有关系的表）

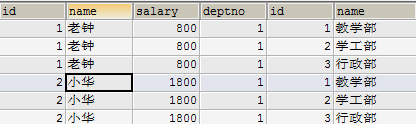
## 部门表



**员工表：**



## 内连接会产生笛卡尔积（一个表的一条数据与另外一个表的所有数据进行拼接）



我们可以通过主表的主键与从表的外键进行过滤。

## 3.1 内连接（最频繁）

|  |
| --- |
| -- 1.1 内连接查询（方言）  SELECT e.name , d.deptname  from employee e ,dept d  WHERE e.id = d.id;  -- 另一种语法(标准)  SELECT e.name,d.deptname  from employee e INNER JOIN dept d  on e.deptid = d.id; |

## 3.2 左外连接

|  |
| --- |
| -- 1.2 左外连接(LEFT OUTER JOIN )  -- 左外连接效果：首先全部显示左表数据，然后右表数据逐个匹配左表，如果满足连接条件的右表数据就显示出来，但不满足则显示null  SELECT e.name,d.deptname  from employee e left JOIN dept d  on e.deptid = d.id; |

## 3.3 右外连接

|  |
| --- |
| -- 1.3 右外连接(RIGHT OUTER JOIN)  -- 右外连接效果：首先全部显示右表数据，然后左表数据逐个匹配左表，如果满足连接条件的左表数据就显示出来，但不满足则显示null  SELECT e.name,d.deptname  from employee e right JOIN dept d  on e.deptid = d.id; |