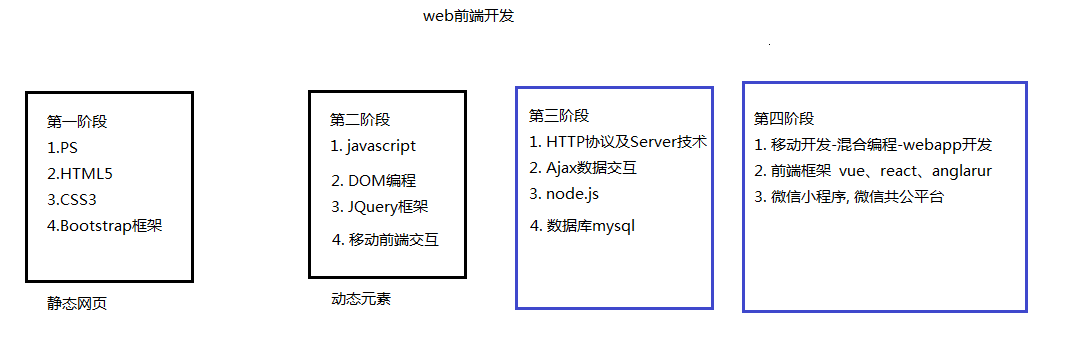
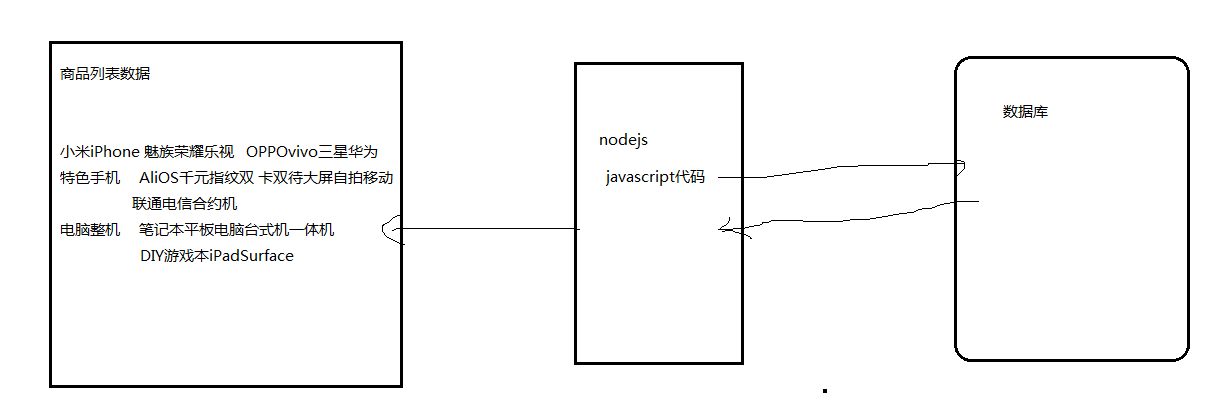
# 学习笔记

# web前端技术学习内容介绍

## 1.web前端技术



## 2. web应用程序三大部份, 界面-业务-数据存储(数据库)



## 二、 数据库mysql学习

### 1、Mysql数据库安装

1. 安装，卸载Mysql数据库服务

1. 那么如何启动和关闭MySQL服务呢？

　　方法一：右击”我的电脑“-->”管理“-->"服务和应用程序"-->"服务"。

　　方法二：通过 DOS 窗口启动和关闭MySQL服务。

　　在 DOS 窗口中，如果想查看 Windows 系统已经启动的服务，使用命令：net start

　　关闭MySQL服务：net stop MySQL

开启MySQL服务：net Start MySQL

注：net命令找不到，设置path环境变量值为: C:\Windows\System32

2. 移除原来的mysql服务

mysqld -remove MySQL

3. 下载mysql

ZIP Archive版是免安装的。只要解压就行了,不需要安装。解压到D盘mysql目录下D:\mysql\mysql-5.7.21-win32

4. 配置

新建my.ini文件

(下面的内容即为所新建的my.ini文件的内容。记得修改里面的路径。)

[mysql]

# 设置mysql客户端默认字符集

default-character-set=utf8

[mysqld]

#设置3306端口

port = 3306

# 设置mysql的安装目录

basedir=D:\mysql\mysql-5.7.16-win32

# 设置mysql数据库的数据的存放目录

datadir=D:\mysql\mysql-5.7.16-win32\data

# 允许最大连接数

max\_connections=200

# 服务端使用的字符集默认为8比特编码的latin1字符集

character-set-server=utf8

# 创建新表时将使用的默认存储引擎

default-storage-engine=INNODB

1. 安装

1.安装

D:\mysql\mysql-5.7.16-win32\bin>**mysqld install**  
 Service successfully installed.

1. 初始化

D:\mysql\mysql-5.7.16-win32\bin>mysqld --initialize //–initialize前面是两横’–‘

1. 启动服务

D:\mysql\mysql-5.7.16-win32\bin>net start mysql

MySQL 服务正在启动 .

MySQL 服务已经启动成功。

1. 登录mysql服务

**mysql –uroot –p root@%**

其中，默认的初始化密码在用“mysqld –initialize”初始化后的data文件夹的\*.err文件中。用记事本打开该文件找到类似下图的地方

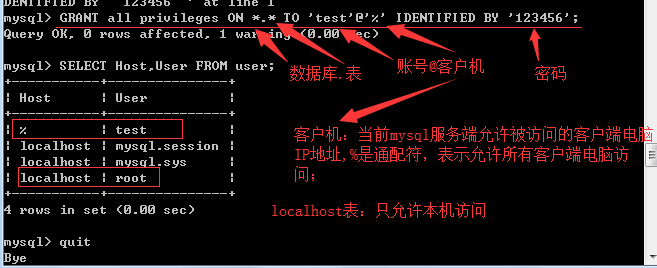
Note] A temporary password is generated for root@localhost: dcvz&(Rel1J2

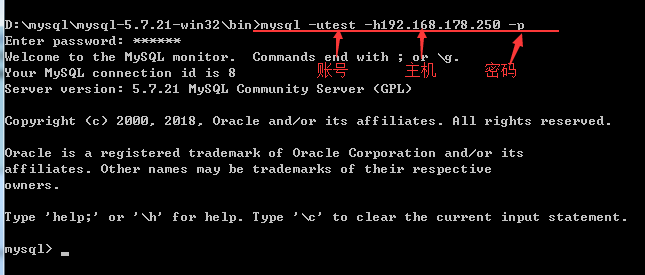
其中反白部分的”dcvz&(Rel1J2“就为初始化的密码。

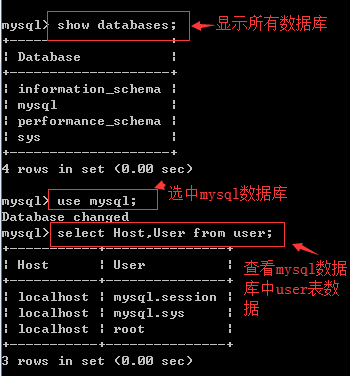
修改默认ROOT密码

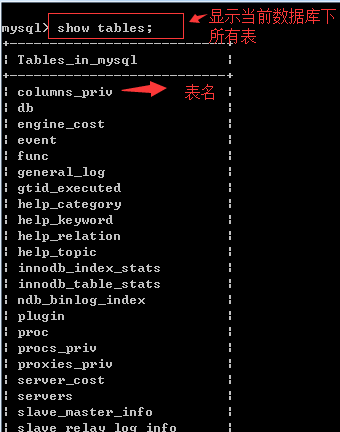
首先使用默认密码登陆数据库，然后使用 ” set password = password(‘你自己的密码’); “ 语句即可。

1. 创建mysql账号









## 2. 数据库入门

**数据库表(关系),字段,记录**

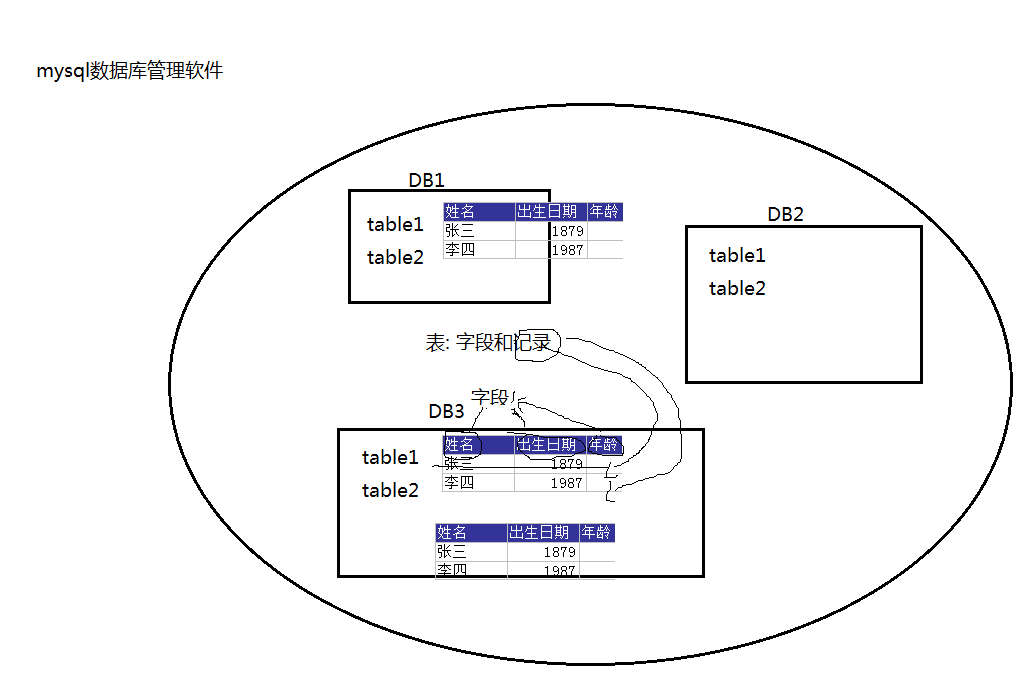
**表**: 一个关系数据库就是由二维表和他们之间的关系组成. 二维表在数据库中就叫表.

**字段（Field）**: 对应于属性的数据称为字段，也称为数据项。字段的命名往往和属性名相同。

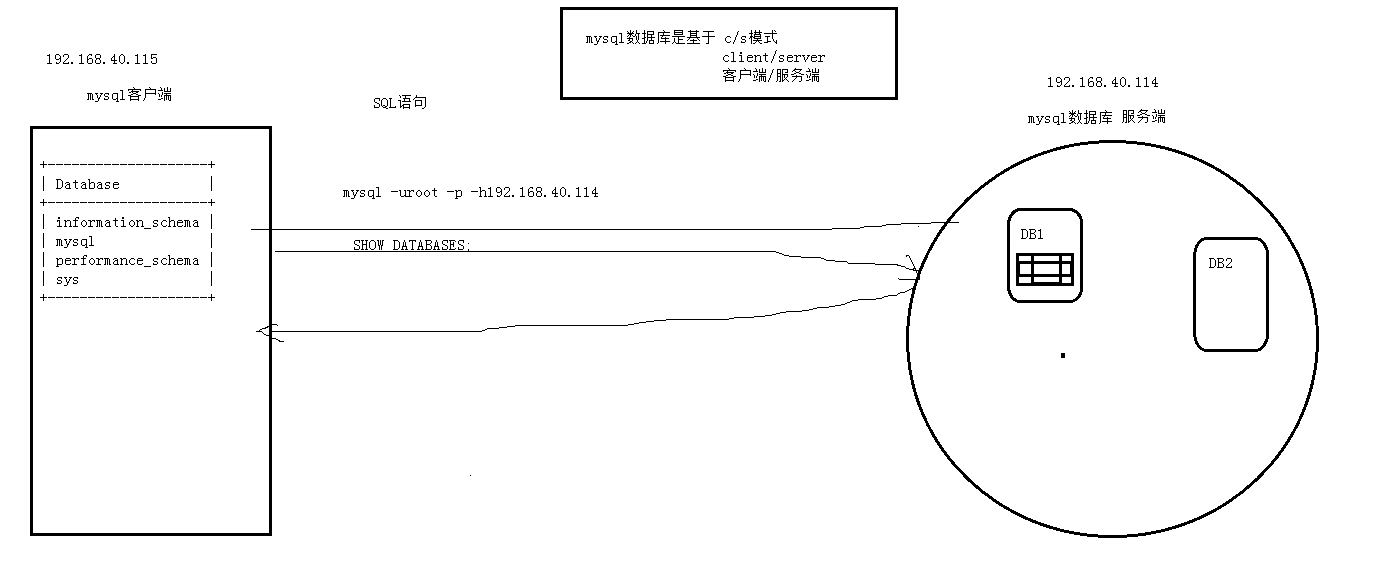
如学生有学号、姓名、年龄、性别、系等字段。

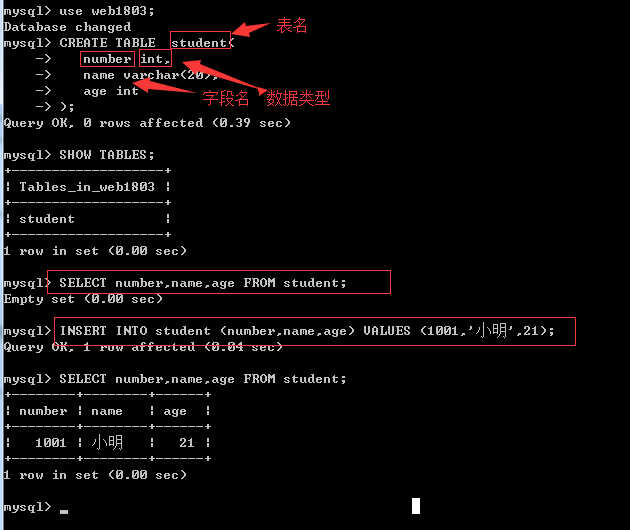
**记录（Record）**: 对应于每个实体的数据称为记录。

如一个学生（990001，张立，20，男，计算机）为一个记录。

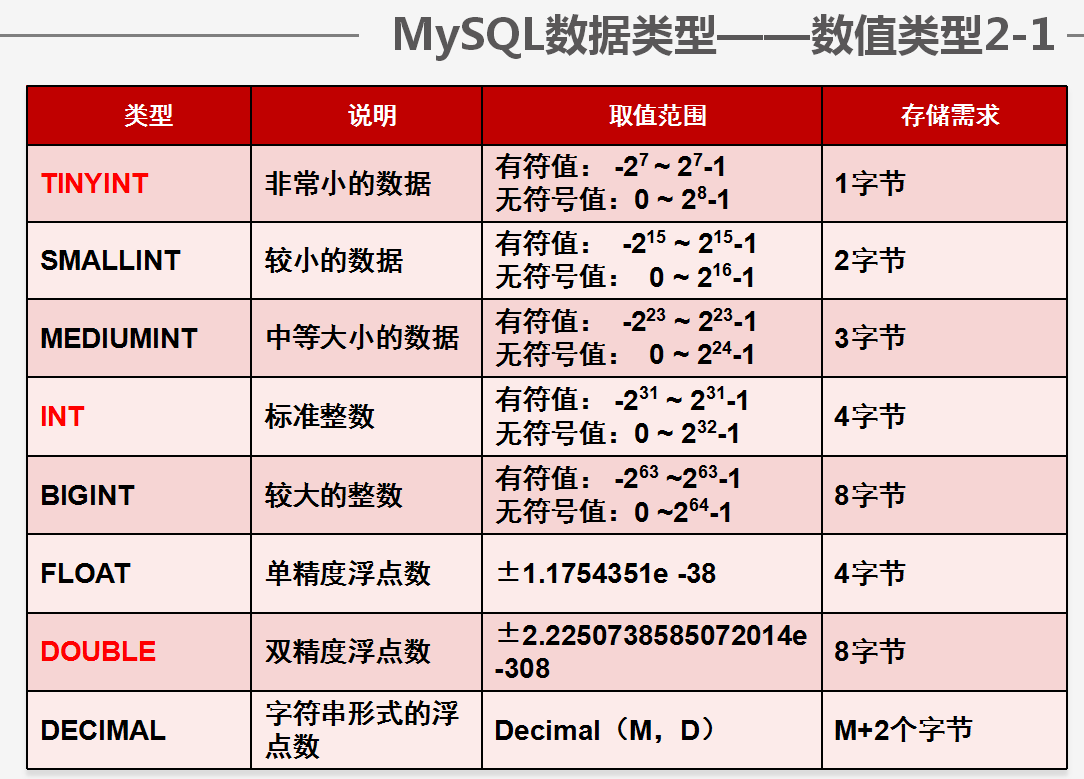


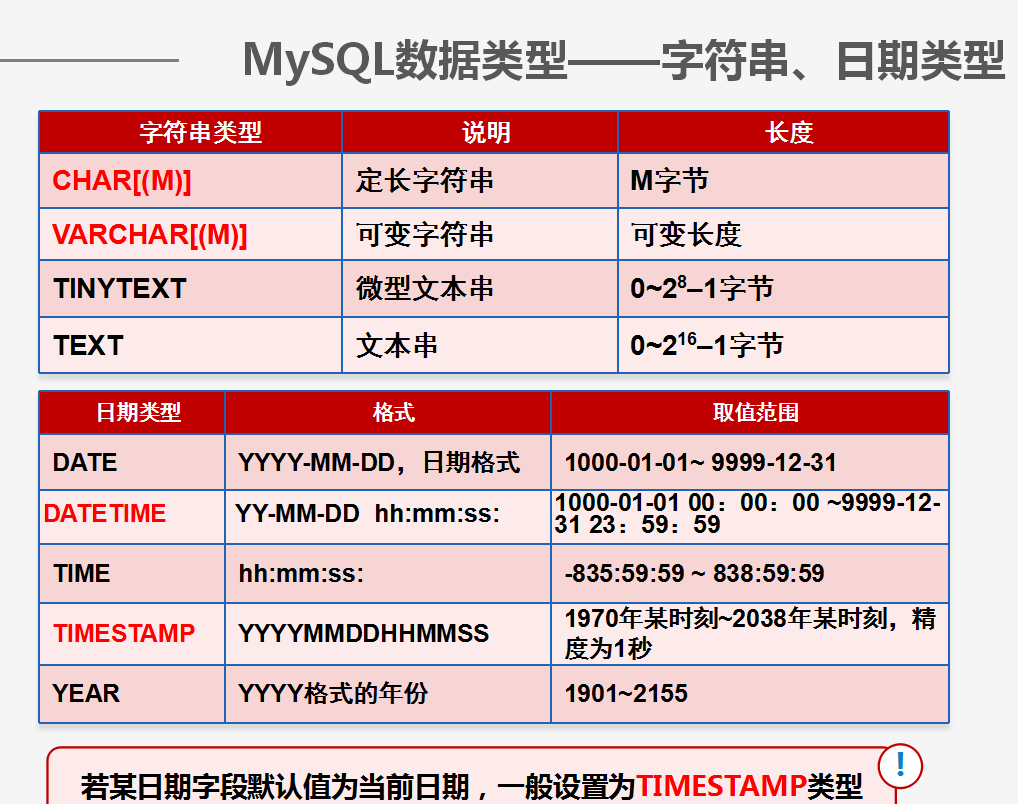
## .MYSQL客户端/服务端模式



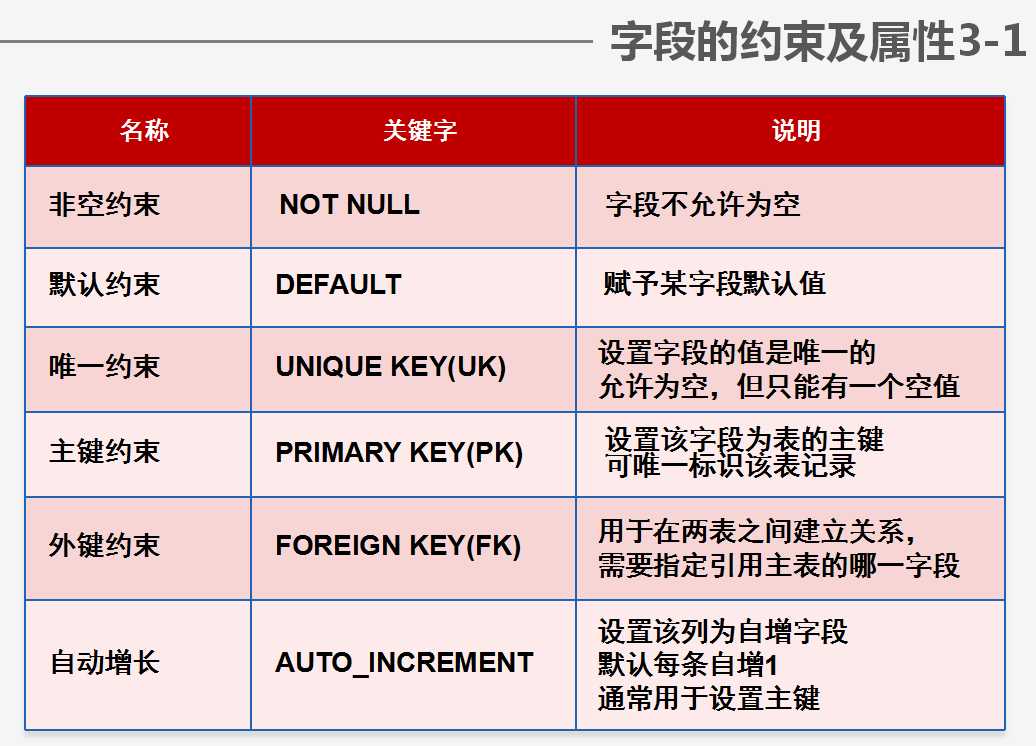


1. 数据类型



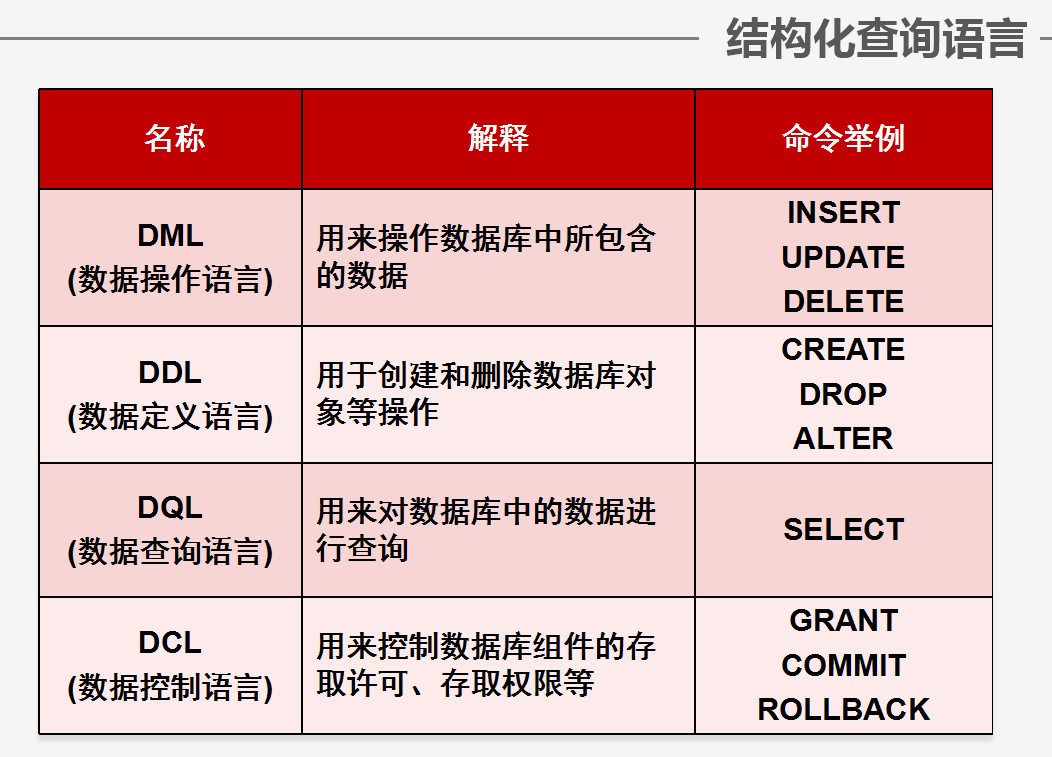


三 、约束



|  |
| --- |
| CREATE TABLE `student`(  `number` int(4) AUTO\_INCREMENT COMMENT '学号',  `name` varchar(50) NOT NULL DEFAULT '小明' COMMENT '姓名',  `age` int(3) NOT NULL DEFAULT 10 COMMENT '年龄',  PRIMARY key (number)  )  DROP TABLE 表名; |

1. 结构化查询语言SQL



1. DDL数据定义语言

|  |
| --- |
| CREATE DATABASE 数据库名;  DROP DATABASE 数据库名;  CREATE TABLE 表名; //创建表  DROP TABLE 表名; //删除表  ALTER TABLE 表名 DROP 字段名; //删除字段  ALTER TABLE 表名 ADD 字段名 字段类型; //添加字段  ALTER TABLE student ADD PRIMARY KEY(number); //给无主键的表结构添加主键  ALTER TABLE student ALTER score SET DEFAULT 0; //修改字段默认值  ALTER TABLE 表名 ALTER 字段名 SET NOT NULL; //设置非空约束  ALTER TABLE student CHANGE COLUMN number number int(4) NOT NULL AUTO\_INCREMENT PRIMARY KEY; |
|  |

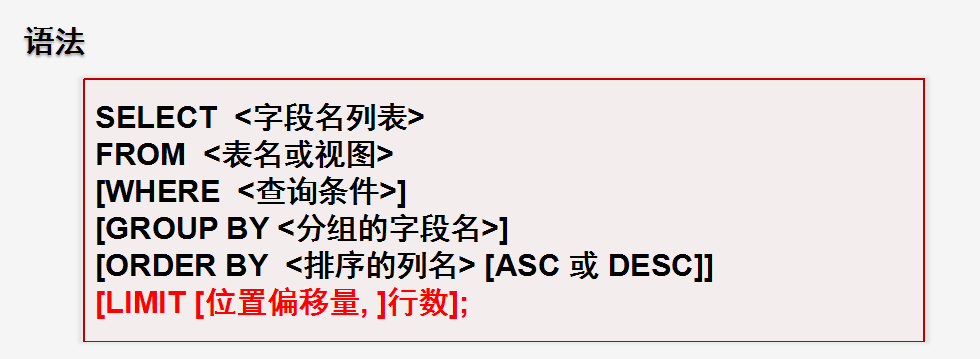
1. DML数据操作语言

|  |
| --- |
| INSERT INTO 表名 字段列表 VALUES 值列表  INSERT INTO 表名 (字段1，字段2...) VALUES (值1，值2...)  UPDATE 表名 SET 字段1=值1,字段2=值2 ...  [WHERE 条件] ;  DELETE FROM 表名  [WHERE 条件]； |

1. DQL数据查询语言

|  |
| --- |
| SELECT 字段列表 FROM 表名  [WHERE 条件] |

# 五、Where子句





# 聚合函数

sum(column\_name) 求和

avg(column\_name) 求平均值

max(column\_name) 求最大值

min(column\_name) 求最小值

round(值,小数个数) 去小数

count(column\_name) 求记录条数

# 六、分组Group By

**示例:分组Group by子句**

**创建订单表**

|  |  |  |  |
| --- | --- | --- | --- |
| **O\_Id** | **OrderDate** | **OrderPrice** | **Customer** |
| 1 | 2008/12/29 | 1000 | Bush |
| 2 | 2008/11/23 | 1600 | Carter |
| 3 | 2008/10/05 | 700 | Bush |
| 4 | 2008/09/28 | 300 | Bush |
| 5 | 2008/08/06 | 2000 | Adams |
| 6 | 2008/07/21 | 100 | Carter |

**CREATE TABLE `Orders`(**

**`O\_id` int(3) PRIMARY KEY COMMENT '订单ID',**

**`orderDate` date COMMENT '订单日期',**

**`OrderPrice` int COMMENT '订单价格',**

**`Customer` varchar(20) COMMENT'客户名'**

**)**

**INSERT INTO Orders VALUES(1,'2008/12/29',1000,'张三');**

**INSERT INTO Orders VALUES(2,'2008/11/23',1600,'李四');**

**INSERT INTO Orders VALUES(3,'2008/10/05',700,'张三');**

**INSERT INTO Orders VALUES(4,'2008/09/28',300,'张三');**

**INSERT INTO Orders VALUES(5,'2008/08/06',2000,'王二');**

**INSERT INTO Orders VALUES(6,'2008/07/21',100,'李四');**

查找每个客户的总金额（总订单）

**SELECT SUM(OrderPrice)**

**FROM Orders**

**GROUP BY Customer;**

**mysql> SELECT SUM(OrderPrice) AS 总金额, Customer**

**-> FROM Orders**

**-> GROUP BY Customer;**

**+-----------+----------+**

**| 总金额 | Customer |**

**+-----------+----------+**

**| 2000 | 张三 |**

**| 1700 | 李四 |**

**| 2000 | 王二 |**

**+-----------+----------+**

查找总订单金额大于1800的客户?

**mysql> SELECT SUM(OrderPrice) AS 总金额, Customer**

**FROM Orders**

**GROUP BY Customer**

**HAVING SUM(OrderPrice) > 1800;**

**HAVING条件可以与函数一起使用，where不能**

# 七、Order by 排序

mysql> SELECT \* FROM Orders;

+------+------------+------------+----------+

| O\_id | orderDate | OrderPrice | Customer |

+------+------------+------------+----------+

| 1 | 2008-12-29 | 1000 | 张三 |

| 2 | 2008-11-23 | 1600 | 李四 |

| 3 | 2008-10-05 | 700 | 张三 |

| 4 | 2008-09-28 | 300 | 张三 |

| 5 | 2008-08-06 | 2000 | 王二 |

| 6 | 2008-07-21 | 100 | 李四 |

+------+------------+------------+----------+

6 rows in set (0.00 sec)

**mysql> SELECT \* FROM Orders**

**-> ORDER BY OrderPrice ASC; // ASC升序， DESC 降序**

+------+------------+------------+----------+

| O\_id | orderDate | OrderPrice | Customer |

+------+------------+------------+----------+

| 6 | 2008-07-21 | 100 | 李四 |

| 4 | 2008-09-28 | 300 | 张三 |

| 3 | 2008-10-05 | 700 | 张三 |

| 1 | 2008-12-29 | 1000 | 张三 |

| 2 | 2008-11-23 | 1600 | 李四 |

| 5 | 2008-08-06 | 2000 | 王二 |

+------+------------+------------+----------+

# 八、 Limit分页子句

mysql> SELECT \* FROM Orders

-> ;

+------+------------+------------+----------+

| O\_id | orderDate | OrderPrice | Customer |

+------+------------+------------+----------+

| 1 | 2008-12-29 | 1000 | 张三 |

| 2 | 2008-11-23 | 1600 | 李四 |

| 3 | 2008-10-05 | 700 | 张三 |

| 4 | 2008-09-28 | 300 | 张三 |

| 5 | 2008-08-06 | 2000 | 王二 |

| 6 | 2008-07-21 | 100 | 李四 |

+------+------------+------------+----------+

6 rows in set (0.00 sec)

mysql> SELECT \* FROM Orders

-> LIMIT 3;

+------+------------+------------+----------+

| O\_id | orderDate | OrderPrice | Customer |

+------+------------+------------+----------+

| 1 | 2008-12-29 | 1000 | 张三 |

| 2 | 2008-11-23 | 1600 | 李四 |

| 3 | 2008-10-05 | 700 | 张三 |

+------+------------+------------+----------+

**mysql> SELECT \* FROM Orders**

**-> LIMIT 1,4;**

**+------+------------+------------+----------+**

**| O\_id | orderDate | OrderPrice | Customer |**

**+------+------------+------------+----------+**

**| 2 | 2008-11-23 | 1600 | 李四 |**

**| 3 | 2008-10-05 | 700 | 张三 |**

**| 4 | 2008-09-28 | 300 | 张三 |**

**| 5 | 2008-08-06 | 2000 | 王二 |**

**+------+------------+------------+----------+**

**语法:**

**Limit [参数1，] 参数2**

**第一个参数: 可选 表示： 偏移量, 序号，第一条记录序号从0开始， 参数1，表示从第二条记录开始**

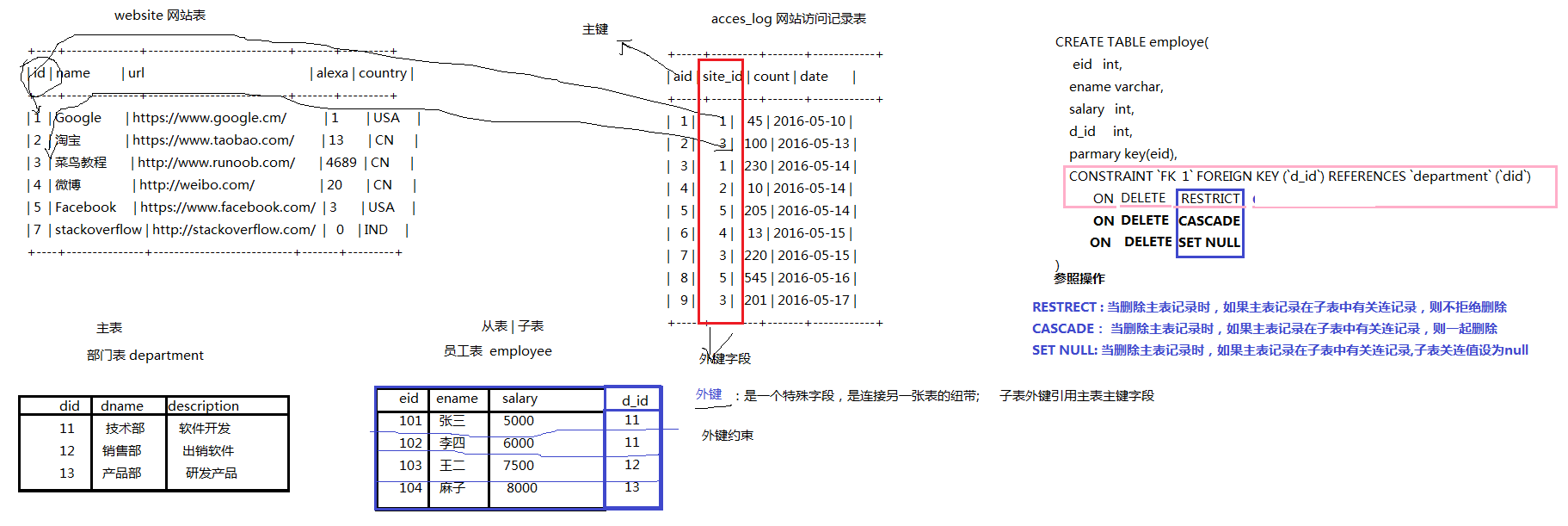
**第二个参数：必选 表示: 查询记录条数**

****

**SELECT \* FROM 表名**

**LIMIT 1,5**

# 九、外键和外键约束



# 十、表间关系

**表关系**

一对一联系（1:1）

实体集A中的一个实体至多与实体集B中的一个实体相对应，反之亦然。

如：班级与班长，观众与座位，病人与床位

一对多联系（1:n）

实体集A中的一个实体与实体集B中的多个实体相对应，而B中的一个实体至多与A中的一个实体相对应.

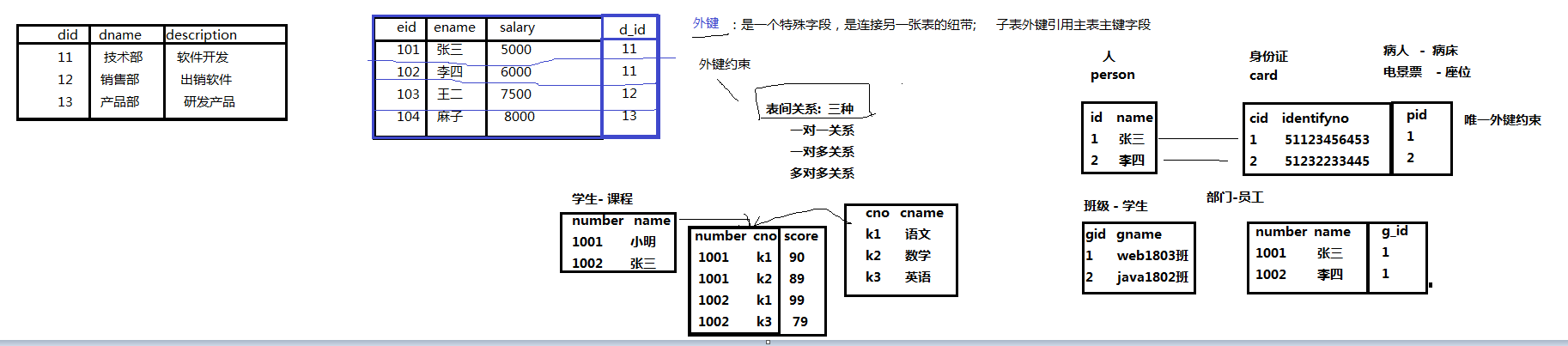
如：班级与学生、公司与职员、省与市



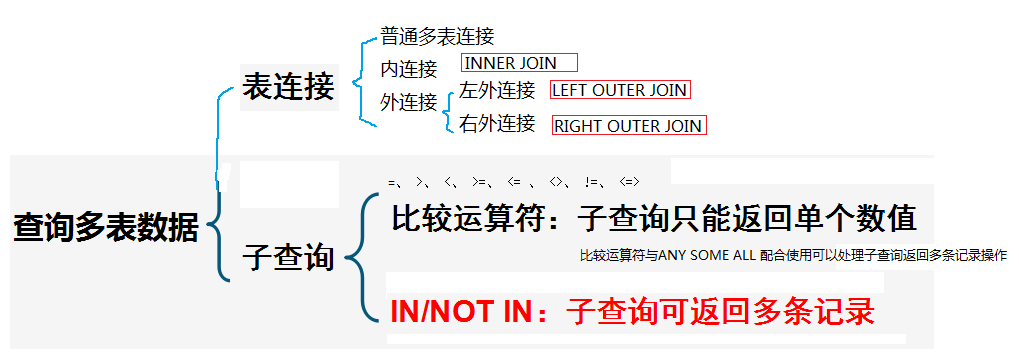
多对多（m:n）

实体集A中的一个实体与实体集B中的多个实体相对应，反之亦然.

如：教师与学生,学生与课程，工厂与产品



# 十一、查询多表数据



## 连接查询

多表查询就是在多个有逻辑联系的表之间进行的查询,逻辑关系主要是指主外键之间的联系;

要实现多表之间的查询就要依靠表连接或者是子查询的方式实现：

如果两张表没加关连条件进行查询得到的是两张表的笛卡尔集:

什么是笛卡尔集:

{a,b} {1,2}

笛卡尔集: {a,1} { a,2} {b,1} {b,2}

下述查询得到 deparment表和employee表的笛卡尔集

mysql> SELECT \*

-> FROM department, employee;

+-----+-----------+------+--------+--------+------+

| id | name | eid | name | salary | did |

+-----+-----------+------+--------+--------+------+

| 102 | 市场部 | 2001 | 张三 | 4000 | 102 |

| 103 | 销售部 | 2001 | 张三 | 4000 | 102 |

| 102 | 市场部 | 2005 | 小明 | 3000 | 102 |

| 103 | 销售部 | 2005 | 小明 | 3000 | 102 |

| 102 | 市场部 | 2006 | 小李 | 3500 | 103 |

| 103 | 销售部 | 2006 | 小李 | 3500 | 103 |

| 102 | 市场部 | 2007 | 小丽 | 4500 | 103 |

| 103 | 销售部 | 2007 | 小丽 | 4500 | 103 |

+-----+-----------+------+--------+--------+------+

加入关连条件 d.id = e.did后查询得到 两表交集

mysql> SELECT e.name,d.name, e.salary

FROM employee as e,department as d

**WHERE d.id = e.did;**

+--------+-----------+--------+

| name | name | salary |

+--------+-----------+--------+

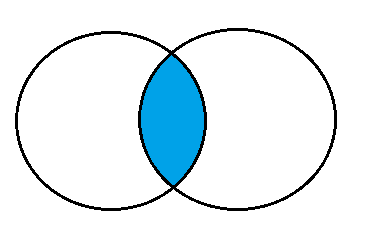
| 张三 | 市场部 | 4000 |

| 小明 | 市场部 | 3000 |

| 小李 | 销售部 | 3500 |

| 小丽 | 销售部 | 4500 |

+--------+-----------+--------+



### 1普通多表连接查询

语法:

SELECT 字段1…..

FROM 表1，表2

WHERE 主键=外键 // 连接条件

示例: 张三所在的部门?

分析: 查询结果和查询条件

SELECT d.name

FROM department AS d,employee AS e

WHERE d.id = e.did

AND e.name = '张三';

### 连接查询

#### 1>内连接

##### 语法:

SELECT 字段1 ….

FROM 表1 INNER JOIN 表2

ON 连接条件(主键=外键)

WHERE 其它条件

mysql> SELECT d.name,e.name,e.salary

-> FROM department AS d INNER JOIN employee AS e

-> ON d.id = e.did;

+-----------+--------+--------+

| name | name | salary |

+-----------+--------+--------+

| 市场部 | 张三 | 4000 |

| 市场部 | 小明 | 3000 |

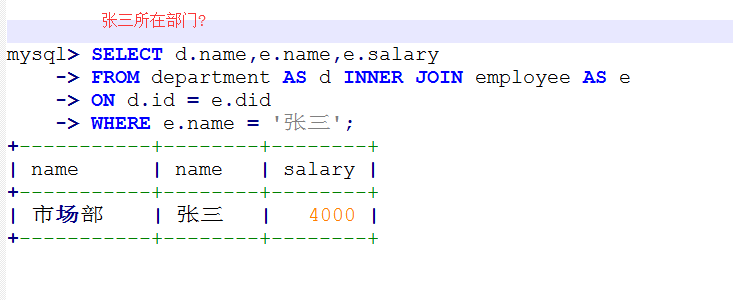
| 销售部 | 小李 | 3500 |

| 销售部 | 小丽 | 4500 |

+-----------+--------+--------+

4 rows in set (0.01 sec)

##### 示例: 张三所在的部门?



mysql> SELECT d.name,e.name,e.salary

-> FROM department AS d INNER JOIN employee AS e

-> ON d.id = e.did

-> WHERE e.name = '张三';

+-----------+--------+--------+

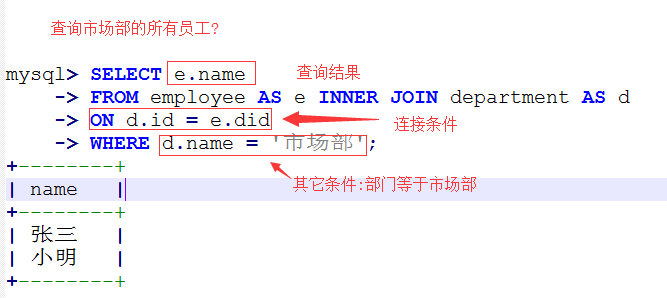
| name | name | salary |

+-----------+--------+--------+

| 市场部 | 张三 | 4000 |

+-----------+--------+--------+

##### 示例: 市场部下所有员工?



mysql> SELECT e.name

-> FROM employee AS e INNER JOIN department AS d

-> ON d.id = e.did

-> WHERE d.name = '市场部';

+--------+

| name |

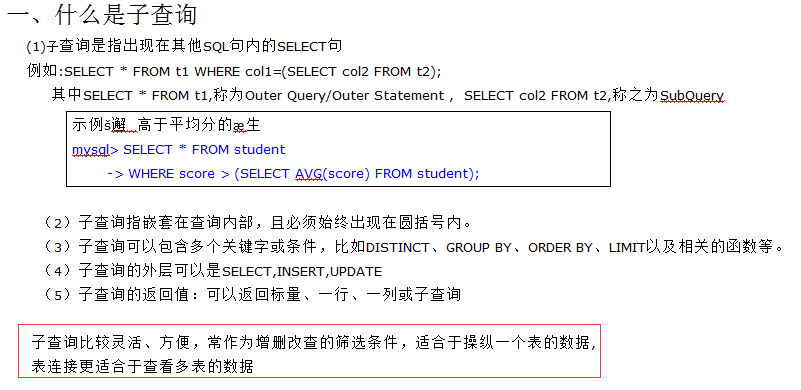
+--------+

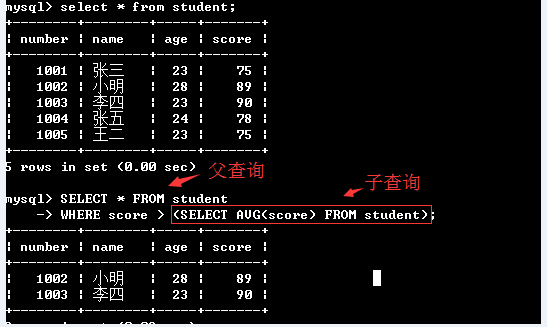
| 张三 |

| 小明 |

+--------+

### 3．子查询





若子查询结果有多个，在进行条件判断时，要加修饰符: ANY 或ALL

graphic

>ANY ： 比子查询结果任一结果大则满足条件

如: WHERE sno >ANY (2,5)

sno 只要比2大或者5大都可以

大于最小值

>ALL : 比子查询结果所有结果大则满足条件

如: WHERE sno >ALL (2,5)

sno 大于最大值5

<ANY : 比子查询结果任一结果小则满足条件

小于最大值

<ALL 小于最小值

检索学号比李同学大，而年龄比他小的学生姓名。

mysql> SELECT SNAME

-> FROM student

-> WHERE sno > ALL (SELECT SNO from student WHERE sname LIKE '李%')

-> AND age < ALL (SELECT AGE FROM student WHERE sname LIKE '李%');

+--------+

| SNAME |

+--------+

| 张友 |

+--------+