# 上课笔记

数据库服务器：安装了数据库管理系统的服务器；

数据库管理系统：DBMS，管理和组织数据的软件；

数据库：存放数据的仓库；

表：真正放数据的地方；

字段：标记数据的属性；

记录：每条数据成为一条记录；

数据：字段和记录交叉的地方；

PHPStudy：集成环境

类似office：word，excel，power；

包含：

MySQL数据库管理系统，就是数据库服务器；

Apache：web服务器软件；

PHP：负责解析和执行PHP代码；

一个数据库管理系统（MySQL）里面，我们可以创建多个数据库；

MySQL:

最高级别账号：root， 默认密码：root

默认端口：3306；

## 创建数据库：



数据库命名：

见名知意：你的名字应该代表一定的含义；

## SQL概述

结构化查询语言(Structured Query Language)简称SQL，是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统；同时也是数据库脚本文件的扩展名。

## 数据操作：

增删改查：SQL语句，结构化查询语句，用来操作数据库；

### 增加：

Document.createElement

**INSERT INTO**  tablename(f1, f2, f3…) VALUES (v1, v2, v3…)

INSERT INTO students(sname, tel, snum, status) VALUES (‘吴正鹏’, ‘13258336666’, ‘SH1807006’, 1);

**INSERT** **INTO** students(sname, snum, tel, **status**) **VALUES** ('吴正鹏', 'SH1807006', '16989898989', 1);

### 删除：

DELETE FROM tablename [WHERE 判断条件]；

= ：字段等于指定的值，不是==；

Js里面所有的比较运算符都可以在SQL里面使用；

**DELETE** **FROM** students **WHERE** 5=5

**DELETE** **FROM** students **WHERE** 1

**DELETE** **FROM** students **WHERE** sid=5

**清空表：TRUNCATE** `students`;

但是很少使用：危险操作，真正删除的时候应该是修改操作

删除业务：

删除的时候干了什么事情？应该是状态修改，而不是直接从表里面删除数据；

### 修改：

UPDATE tablename SET f1 = nv1, f2=nv2, f3=nv3 …. [WHERE 判断条件];

**UPDATE** students **SET** sname = '张计荣', snum='SH1807009', tel='18989898765' **WHERE** sid = 6

### 查询：

SELECT f1, f2, f3…. FROM tablename [WHERE 判断条件];

字段可以使用\*代替，表示查询所有的字段，一般情况下不要使用,性能低；

select \* **from** students **WHERE** sid=7

SELECT后面支持表达式：

**SELECT** \*, (age-5) **AS** na, **NOW**() **FROM** students **ORDER** **BY** **RAND**() **DESC** **LIMIT** 1

判断条件支持逻辑运算符：

与： AND 不是 &&

或： OR 不是 ||

非： NOT 不是！

**select** \* **from** students **WHERE** sid > 7 **AND** sid < 9

指定查询范围：

SELECT \* FROM tablename WHERE 判断条件 LIMIT start, num(不是end)

Start表示开始位置，最小值是0 ，当为0的时候，可以省略；

Num：表示要查询的记录数量；

**SELECT** \* **FROM** top20 **LIMIT** 1, 10

分页的实现：

Pagenum = 20;

Page = 1:

**SELECT** \* **FROM** top20 **LIMIT** 0, 20

Page = 2:

**SELECT** \* **FROM** top20 **LIMIT** 20, 20

Page = 3:

**SELECT** \* **FROM** top20 **LIMIT** 40, 20

Page = n:

**SELECT** \* **FROM** top20 **LIMIT** (n-1)\* Pagenum, Pagenum

模糊查询：看起来像 like

SELECT \* FROM tablename WHERE kw LIKE ‘%羽绒服%’

**SELECT** \* **FROM** top20 **WHERE** kw **LIKE** '%羽绒服%' //包含该关键词

**SELECT** \* **FROM** top20 **WHERE** kw **LIKE** '羽绒服%' //关键词开头的

**SELECT** \* **FROM** top20 **WHERE** kw **LIKE** '%羽绒服' //关键词结束的

对查询结果进行排序：order BY

SELECT \* FROM tablename WHERE 判断条件 ORDER BY 字段|表达式 ASC|DESC

UTF8编码下的中文排序问题：

**SELECT** \* **FROM** top20 **WHERE** id < 50 **ORDER** **BY** **CONVERT**(kw **USING** **GBK**) **ASC** **limit** 0, 50

可以指定多个排序条件：

**SELECT** \* **FROM** e **ORDER** **BY** deptid **ASC**, salary **DESC**;

更多的判断条件：

IN：指定多个查询值；

**SELECT** \* **FROM** top20 **WHERE** id **IN**(10,15,45,78)

Between n AND m：

指定查询范围；

**SELECT** \* **FROM** top20 **WHERE** id **BETWEEN** 101 **AND** 199

**SELECT** \* **FROM** top20 **WHERE** id **NOT** **BETWEEN** 1 **AND** 100 **LIMIT** 50

**SELECT** \* **FROM** top20 **WHERE** kw **NOT** **LIKE** '%羽绒服%' **LIMIT** 50

创建表原则：

一个表做一件事情；

一个事情创建一张表；

多表查询：

左连接：

SELECT table1.\*, table2.\* FROM table1

LEFT JOIN table2

ON table1.pid = table2.pid

WHERE table1.sid = 2 ORDER BY table1.sid DESC

**SELECT** students.\*, classlist.cname **FROM** students **LEFT** **JOIN** classlist **ON** students.cid = classlist.cid

更好的方式是给表取个缩写的名称：

**SELECT** s.\*, c.cname **FROM** students **AS** s **LEFT** **JOIN** classlist **AS** c **ON** s.cid = c.cid **WHERE** s.sid = 2

多个表左连接的时候：

**SELECT**

s.sname, s.sid, s.tel, s.snum, c.cname, p.pname, c1.cityname, a.aname

**FROM** students **AS** s

**LEFT** **JOIN** classlist **AS** c **ON** s.cid = c.cid

**LEFT** **JOIN** province **AS** p **ON** s.pid = p.pid

**LEFT** **JOIN** city **AS** c1 **ON** s.cityid = c1.cityid

**LEFT** **JOIN** **area** **AS** a **ON** s.areaid = a.areaid

**WHERE** s.sid = 2

ORDER BY s.sid;

左连接的时候，允许右边的表的记录为null，主表不能是空记录；

右连接：

和左连接相反：不允许右边的表有空记录，左边的表可以有空记录；

**SELECT** s.sname, s.tel, s.snum, c.cname **FROM** students **AS** s

**RIGHT** **JOIN** classlist **AS** c **ON** s.cid = c.cid

内连接：

**SELECT** s.sname, s.sid, s.tel, s.snum, c.cname **FROM** students **AS** s

**INNER**  **JOIN** classlist **AS** c **ON** s.cid = c.cid

所有的表都不能有空记录

多表连接：

**SELECT** s.sname, s.sid, s.tel, s.snum, c.cname **FROM** students **AS** s, classlist **AS** c WHERE s.cid = c.cid

查询满足条件的记录数：

Count(sid)

SELECT COUNT(sid) AS nums FROM students WHERE 1

**SELECT** \* **FROM** students **WHERE** 1;

**SELECT** **COUNT**(sid) **FROM** students **WHERE** 1;

**SELECT** **COUNT**(sid) **AS** nums **FROM** students **WHERE** 1;

分组查询：

对查询结果进行分组

GROUP BY cid

统计每个班级学生数：

**SELECT** s.cid, **COUNT**(s.sid) **AS** nums, c.cname **FROM** students **AS** s **LEFT** **JOIN** classlist **AS** c **ON** s.cid = c.cid **WHERE** 1 **GROUP** **BY** s.cid

根据性别统计学生：

**SELECT** s.gender, **COUNT**(s.sid) **AS** nums **FROM** students **AS** s **WHERE** 1 **GROUP** **BY** s.gender

*/\*查询部门里面 高于部门平均工资 的员工\*/*

**SELECT** e.\*, a.avgs **FROM** e

**LEFT** **JOIN** (**SELECT** deptid, **AVG**(salary) **AS** avgs **FROM** e **GROUP** **BY** deptid) **AS** a

**ON** e.deptid = a.deptid

**WHERE** e.salary > a.avgs

**ORDER** **BY** e.deptid **ASC**, e.salary **DESC**;

**SELECT** e.\*, a.avgs **FROM** e, (**SELECT** deptid, **AVG**(salary) **AS** avgs **FROM** e **GROUP** **BY** deptid) **AS** a

**WHERE** e.salary > a.avgs **AND** e.deptid = a.deptid

**ORDER** **BY** e.deptid **ASC**, e.salary **DESC**;

*/\*列出各个部门中工资高于本部门的平均工资的员工数和部门号，并按部门号排序\*/*

**SELECT**

**COUNT**(t.id) **AS** 人数, t.deptid **AS** 部门序号

**FROM**

(**SELECT** e.\*, a.avgs **FROM** e, (**SELECT** deptid, **AVG**(salary) **AS** avgs **FROM** e **GROUP** **BY** deptid) **AS** a

**WHERE** e.salary > a.avgs **AND** e.deptid = a.deptid

**ORDER** **BY** e.deptid **ASC**, e.salary **DESC**) **AS** t

**GROUP** **BY** t.deptid

**ORDER** **BY** t.deptid

* MyISAM存储引擎
  + 由ISAM(Indexed Sequential Access Method:有索引的顺序访问方法)升级而来，适用于有大量SELECT操作的场景；
  + 每个表对应三个文件，文件名都是以表名开始，其中：
    - .frm存储表定义；
    - .MYD (MYData)存储表的数据；
    - .MYI (MYIndex)存储表的索引；
  + 数据以文件的形式存储，在数据的备份、恢复或转移时很方便；
  + 自动增长列必须是索引；
  + 允许没有任何索引和主键的表存在，索引都是保存行的地址；
  + 保存有表的总行数，但是加了WHERE条件后还是要遍历获取满足条件的记录数；
  + 不支持外键；
  + 不支持事务；
* InnoDB存储引擎
  + 适用于有大量更新、删除和插入操作的场景；
  + 所有表的数据存储在同一个文件里，或者配置innodb\_file\_per\_table=1把每个表的数据存储在独立的文件里；
  + 没有保存表的总行数，select count(\*) from table时会遍历整个表；
  + DELETE 性能优，但DELETE FROM table时是一行一行的删除，如果要清空保存有大量数据的表，建议使用truncate table命令；
  + 支持外键；
  + 支持事务；
  + 支持行级锁；
  + 如果没有特殊要求，创建表时一般选择InnoDB引擎；
* Memory存储引擎
  + 基于memory存储引擎的表对应一个”表名.frm”文件，该文件存储的是表结构；
  + 数据存储在内存中，有利于对数据进行快速处理，提高整个表的处理能力；
  + 如果mysqld进程发生异常，服务器重启或关闭都会导致数据的丢失；