MySQL基础

【学习目标】

【学习目标】

- SQL结构化查询语法
- 内置数据库
- 内置函数
- 远程连接

1. 数据库的概念

什么是数据库？为什么要学数据库?

数据库： 存放数据的仓库

学习数据库的作用：

1. **学习SQL注入漏洞(漏洞之王)**
2. 所有的web系统==》 数据==》 存放起来===》数据库里面

前端 --------->后端 --------->数据库

编程相关的所有东西，都是来源于生活

现实生活种的仓库：

京东自建物流， 全国各地有很多很多仓库

京东买东西之后，就近原则在仓库里面直接出货

仓库里面要存放很多的东西， 这些东西必定是需要分类的， 分区域存放

理一个关系：

```
有一个大的京东仓库----------------------------------------------数据库管理系统( DBMS: database management system)

仓库里面有不同的分区(A区，B区, C区)-------------------------具体的数据库(database)

每个区有很多货架-----------------------------------------------------表 (table)

每个货架又有很多层---------------------------------------------------行 (row)

每一层里面才是物品---------------------------------------------------列--物品(column)
```

我们学习的东西就是DBMS

2. 数据库的分类

数据库里面我们大概可以分为两种：

2.1.1. 关系型数据库：

```
1    表格的形式
```

- Oracle数据库： 全球排名第一的数据库，性能最牛逼(以前在**银行**里面用的很多)，不开源，收费
- MySQL数据库: 占有率相当高，Oracle青春版, 开源，免费, 性能也行----(DBA)
- MSSQL数据库： 微软数据库， 也很优秀，很多政府项目用， ---》 .NET平台 (编程语言用C#, 操作系统也是微软的， 数据库也是微软)，微软-->巨硬， 微软现在云服务做的很好
- PGSQL数据库: 和MySQL感觉是孪生姐妹， MSF 里面
- SQLITE数据库: 嵌入式数据库， 做工具， 做爬虫， 做POC校验， 做EXP验证等等， 需要存放很多===> 文件/SQLITE数据库里面

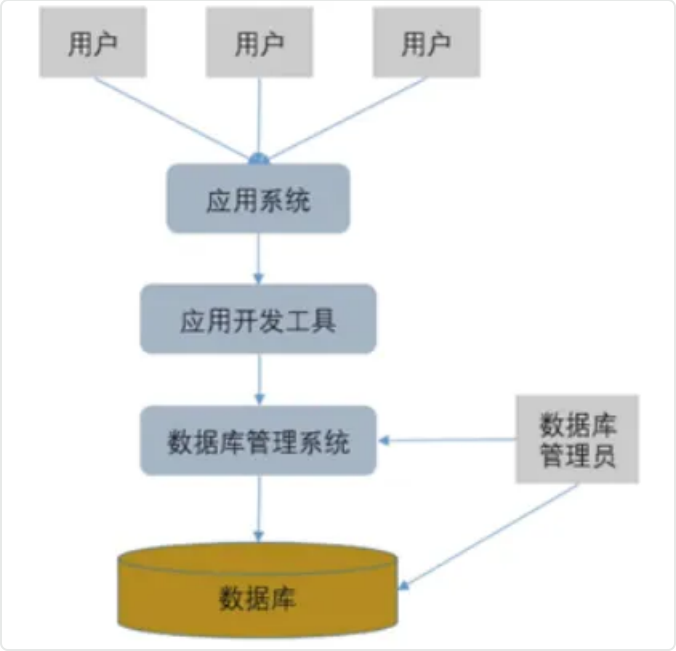
2.1.2. 非关系型数据库：

```
1    图片、文件夹、视频、文档
```

- Redis数据库: 缓存---> 数据库存在内存里面的， 减少数据库IO，增加系统的吞吐量， 后面会和SSRF漏洞进行结合
- memecahce: 缓存
- MongoDB: 真的很牛皮， 大数据库刚刚出来的时候，(一行数据，就可以把一个人的所有数据全部包含)

3. 数据库系统

数据库系统是指在计算机系统中引入数据库后的系统。完整的数据库系统结构关系如图所示：



角色分析：

用户： 普通人/ 会员

应用系统： 访问的京东/淘宝/天猫.....

应用开发工具： webStrom

数据库管理系统: DBMS => DataBase Manager System 每一种数据库都有对应的DBMS

通常我们口中所说的MYSQL都指的是MYSQL的DBMS

数据库：真正存放数据的系统（通常和硬盘交互）

我们学的是： 数据库管理系统+数据库

4. 安装数据库

数据库安装方式有三种：

1. 官网下载安装包，直接安装即可。
2. 官网下载免安装包，绿色版，解压打开即可。
3. 集成环境（WAMP） Windows Apache MySQL PHP

   集成环境有很多种: XAMPP / WAMPSERVER / **PHPstudy**

   PHPStudy =》 切换环境超级方便

安装基本就是傻瓜式安装，安装需要注意的东西：

```
1. 注意安装的盘符，尽量不要在C盘
2. 安装之前，将自己电脑上的MySQL数据库停止掉且删除服务
win+R
输入：services.msc
需要找到是否有MySQL的服务，有就停掉，且删除服务： sc delete 服务名字(进入cmd命令行删除)
```

配置数据库的环境变量

```
win+R--->CMD
```

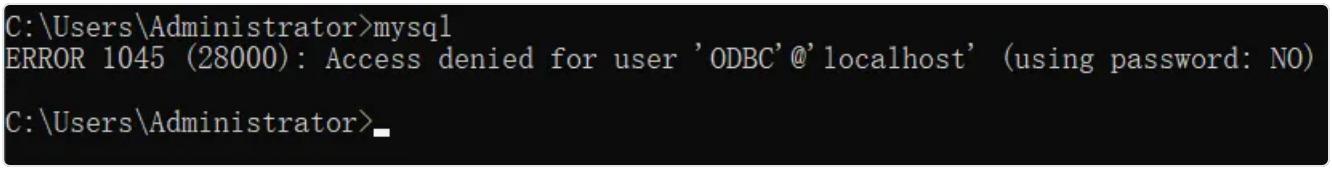
这个时候是需要配置数据库的环境变量。

什么是环境变量： 方便操作系统快速的去找到对应软件的可执行文件。

配置的方式：

1. 我的电脑-->右键-->属性
2. 高级系统设置--->环境变量
3. 系统变量-->双击Path
4. 进入环境变量编辑页面--->新建-->输入mysql可执行文件路径(D:\phpstudy_pro\Extensions\MySQL5.7.26\bin)
5. 后续就是全部确定即可

在任何路径下面输入mysql出现如下界面说明配置成功：



5. MySQL的登录命令

PHPstudy安装的MySQL默认的用户名和密码

用户名： root

密码： root

```
mysql -uroot -p 回车 =》 输入密码
mysql: mysql命令
-u : user 登录用户
-p : 指定密码
MySQL的默认端口是： 3306
mysql -uroot -h localhost -P 3306 -p
-u : 用户名
-h : 主机地址-->MySQL数据库安装的机器的ip地址(127.0.0.1: 回环ip, localhost: 本地主机)
-P : 端口号 默认是 3306 端口即服务
-p : 密码
```

```
C:\Users\Administrator>mysql -uroot -p
Enter password: ****
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)

C:\Users\Administrator>mysql -uroot -p
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.12 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

SQL

SQL是什么?

结构化查询语言

SQL 是所有数据库查询语言的统称

后续操作数据库的主要工作内容： **就是写SQL**

数据库就是根据所写的SQL来执行相关的命令(**增删改查**)

记住： 操作数据库的核心点=》 **建库，建表， 数据的增删改查**

DDL

数据定义语言： Data Define Language

这种语言主要用于： **建库，建表，修改表结构， 删除表，删除库**

DML

数据管理语言: Data Manager Language

这种语言主要用于： 数据的**增删改**

DQL

数据查询语言: Data Query Language

这种语言主要用于： **数据查询**

DCL

数据控制语言： Data Control Language

这种语言主要用于： 数据库的**权限管理**

6. 创建数据库

SQL

```
# 展示所有的数据库
show databases;
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| admin              |
| connect            |
| db1                |
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| security           |
| sys                |
| test               |
| web2401            |
| zhoujielunyyds     |
+--------------------+
12 rows in set (0.01 sec)
# 创建数据库
# 关于数据库名字： 统一小写
create database 数据库的名字;
# 标准的创建数据库的语句
create database if not exists 数据库的名字;
# MySQL数据存放在什么地方的?
# 默认情况是存放在： MySQL安装目录/data目录下面
```

```sql
# 通过观察，MySQL的数据库，在文件系统中就是一个 ：  文件夹而已
# 删除数据库
drop database 数据库名字;
# 标准的删除语句
drop database if exists 数据库名字
```

7. 增删改查

7.1. 增加数据

```sql
# 语法
insert into 表名(字段名1， 字段名2， 字段名3....字段名n) values (值1，值2，值3....值n);

# 省略的字段中约束必须是null的，如果是not null 那就不能省略
mysql> insert into user(user_name,user_age) values('jack',18);
Query OK, 1 row affected (0.01 sec)

mysql> insert into user(user_name,user_age) values('rose',17);
Query OK, 1 row affected (0.00 sec)

mysql> select * from user;
+----+-----------+----------+-------------+--------------+-----------+
| id | user_name | user_age | user_gender | user_address | user_like |
+----+-----------+----------+-------------+--------------+-----------+
| 1 | jack | 18 | 1 | NULL | NULL |
| 2 | rose | 17 | 1 | NULL | NULL |
+----+-----------+----------+-------------+--------------+-----------+
2 rows in set (0.00 sec)
```

```
# 简写 => 全插入
# 必须写所有的字段值
# 自增ID可以写成null => MySQL的特性
insert into 表名 values(所有字段的值依次填写，用逗号隔开);
mysql> insert into user values(null,'lucy',14,2,'Beijing','rap');
Query OK, 1 row affected (0.00 sec)

mysql> select * from user;
+----+-----------+----------+-------------+--------------+-----------+
| id | user_name | user_age | user_gender | user_address | user_like |
+----+-----------+----------+-------------+--------------+-----------+
| 1 | jack | 18 | 1 | NULL | NULL |
| 2 | rose | 17 | 1 | NULL | NULL |
| 3 | lucy | 14 | 2 | Beijing | rap |
+----+-----------+----------+-------------+--------------+-----------+
3 rows in set (0.00 sec)

mysql> insert into user values(null,'lucy',14,2,'Beijing');
ERROR 1136 (21S01): Column count doesn't match value count at row 1

# 批量插入
insert into user(user_name, user_age) values('lucy', 20),('tom',19),('Davi
d',
22), ('Allen', 23);
```

7.2. 删除数据

```
# 语法
delete from 表名 [where 条件]
mysql> delete from user where id=3;
Query OK, 1 row affected (0.01 sec)
mysql> select * from user;
+----+-----------+----------+-------------+--------------+-----------+
| id | user_name | user_age | user_gender | user_address | user_like |
+----+-----------+----------+-------------+--------------+-----------+
| 1 | jack | 18 | 1 | NULL | NULL |
| 2 | rose | 17 | 1 | NULL | NULL |
+----+-----------+----------+-------------+--------------+-----------+
```

7.2.1. 逻辑删除

数据库里面的删除： delete => 直接将数据删除掉在目前这种互联网时代 =》 数据是最重要的很多时候在业务里面，我们是不会去直接将数据删除的， 而是要将数据保留下来，以备后续的数据分析=》 溯源这种情况就不能直接删除，而是逻辑删除我们需要将这条数据加一个状态字段 =》 is_deleted 默认值 0 ， 如果删除 将这个字段值设置 1正常情况查询数据：select * from student where is_deleted = 0;所以所谓删除 =》 实际在数据库里面就是updateupdate student set is_deleted=1 where stu_code=?

7.3. 修改数据

```
# 语法
update 表名 set 字段名1=字段值1,字段名2=字段值2....字段名n=字段值n [where 条件]
# 将jack的地址修改被成都
mysql> update user set user_address='成都' where id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
mysql> select * from user;
+----+-----------+----------+-------------+--------------+-----------+
| id | user_name | user_age | user_gender | user_address | user_like |
+----+-----------+----------+-------------+--------------+-----------+
| 1 | jack | 18 | 1 | 成都 | NULL |
| 2 | rose | 17 | 1 | NULL | NULL |
+----+-----------+----------+-------------+--------------+-----------+
```

7.4. 查询（重点）

7.4.1. 投影查询

```
# 语法：* 代表所有字段(列)
select * from 表名;
## 按照字段进行查询
select user_name,user_gender from user;
```

7.4.2. 限制查询（重中之重）

- 关键字：limit
- 方法一：限制条数 limit n

```
#只查2条数据
select * from user limit 2;
```

- 方法2：分段查询 limit index，length

```
# index : 下标 从0开始
# length : 长度
# 下标从0开始 ，查询2条
select * from user limit 0,2;
# 定一个基调： 3条一页
# 第一页
select * from user limit 0,3;
# 第二页
select * from user limit 3,3;
# 第三页
select * from user limit 6,3;
# 分页有两个参数
# page 当前的页数
# pageSize 每页的条数
# 第一页： page=1&pageSize=60 ===> limit 0, 60
# 第二页: page=2&pageSize=60 ====> limit 60, 60
# 第三页: page=3&pageSize=60 ====> limit 120, 60
# 总结一下规律
# 页码： page
# 页容量: pageSize
limit (page-1)*pageSize, pageSize;
# 这个规律记下来，后面要用

如果有SQL注入漏洞： 我们的核心是啥？==》 查数据
通过注入漏洞去查询： 数据库，数据库中的表，表里面的字段，数据
```

7.4.3. 条件查询

- 关键字：where
- 单条件：

```
select * from user where user_name="JACK";
```

- 多条件：

```
#多条件 =》在JS中 多条件用的逻辑运算符： && || ！
# 在数据库中对应： and or not
select * from user where user_name="ROSE" and user_age >20;
select * from user where user_age > 21 or user_gender="男";

# not 有几个组合
# 以某个字段值是否为空作为条件: is null/ is not null
select * from user where user_address is not null;
select * from user where user_address is null;

# not 还有一种情况 ： 判断某个值是否在某个集合中
# in / not in
select * from user where user_age=20 or user_age=22 or user_age=23;
select * from user where user_name in("jack","rose",23);
select * from user where user_age between 20 and 30;
```

- 比较

```
# >,>=,<,<= , != , between ...and..
# between ...and.. 等价于 >= and <=
```

- 将条件，限制，投影 结合起来

  限制查询一定是放在SQL语句的最后

```
  select user_name,user_age from user where user_age >20 limit 1;
```

7.4.4. 模糊查询

- 关键字：like % _
- 全模糊查询

语法：select * from 表名 where 字段名 like "%条件值%";

```
#查询名字中间有k的同学的所有信息
select * from user where user_name like "%k%";
```

- 半模糊

%放在开头或结尾

- 占位符_表示一个字符

```
#查询名字由三个字符组成且中间字母是c的学生信息
select * from user where user_name like "_o_";
```

拓展：正则表达式查询，关键字：REGEXP

```
select * from user where user_name REGEXP 'ja.k';
```

7.4.5. 排序查询

排序在实际的业务相当多

销量排行， 热歌榜， 成绩同级排名，热点话题排名。。。。。。

关键字 ： order by

语法： order by 字段名 asc/desc

asc 升序 =》 默认值

desc 降序 =》 用的多

```
#通过年龄升序
select * from user order by user_age asc;
#通过年龄降序
select * from user order by user_age desc;
#通过年龄降序后，通过id降序
update user set user_age=22 where user_name="lucy";
select * from user order by user_age desc, id desc;

# 一定记住一点：
# 程序是从左到右，从上到下 依次扫描
# 和安全相关的排序：
select * from table order by 数字
数字：代表的是第几列，就是第几个字段
```

7.4.6. 分组查询

分组的作用就是用来做统计

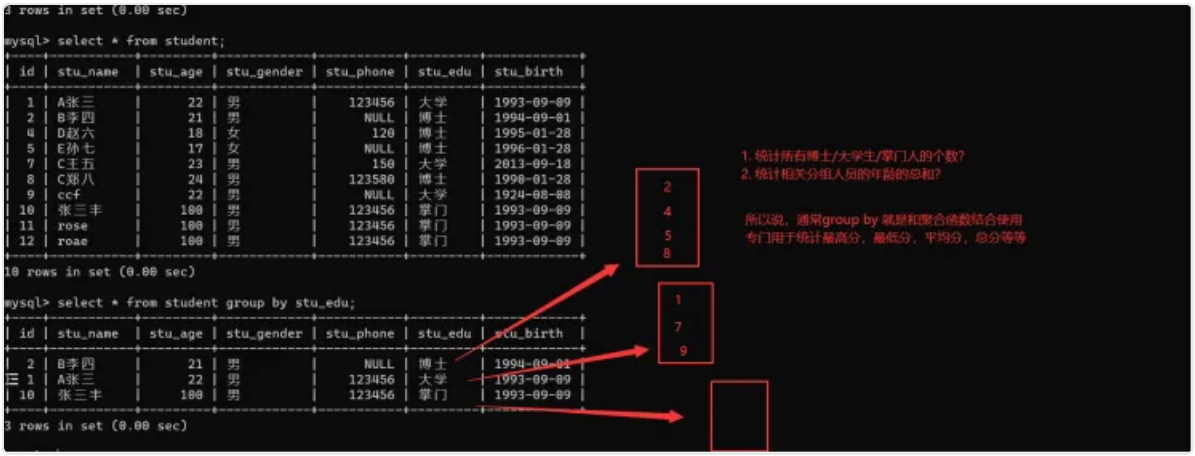
```sql
select * from 表名 [where ...] group by 字段名 [having....]
# group by 通常是和聚合函数一起使用
每个分组是一个虚拟表
所以分组之后只能看到虚拟表中的第一条数据
报错：set sql_mode ='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';


select * from web2401.goods as g ,web2401.category as c where g.cat_id=c.cat_id;

#group分类
select * from web2401.goods as g ,web2401.category as c where g.cat_id=c.cat_id group by cat_name;

#group分类后计数
select cat_name,count(goods_id) as "数量" from web2401.goods as g ,web2401.category as c where g.cat_id=c.cat_id group by cat_name;

#分类后再筛选
select * from web2401.goods as g ,web2401.category as c where g.cat_id=c.cat_id group by cat_name having goods_price>90;
```



```
# 相关的聚合函数
# count(字段名): 求个数
# sum(字段名): 求和
# avg(字段名): 求平均值
# max(字段名): 最大值
# min(字段名): 最小值
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# where 和 having的区别?
# where 比 having 更先执行
# where 是从硬盘上去筛选数据， having 是从where 筛选出来的数据(临时表)里面再去筛选数据
# where 之后不能用聚合函数， having 之后是可以用聚合函数
# where 和 having 都是用来过滤筛选数据
# 相同点： 都是加条件，都是用来过滤
# 不同点:
# where只能用于原来表里面有的字段
# having只能使用聚合函数算出来的中间数据
# SQL语句的位置：where在group之前， having在group之后

# 补充一个 group_concat
# 行转列
```

7.4.7. 连接查询

把多张表的数据进行结合(垂直)

```
# 第一种
# 语法: 笛卡尔乘积
select * from t_student as s, t_class as c where s.c_id=c.c_id;

# 第二种：左连接，右连接， 内连接
# 左连接： 以左表的数据为基准，会将左表有的数据全部查询出来
select * from t_student s
left join t_class c on s.c_id=c.c_id;
# 右连接: 以右表的为基准， 右表所有的数据都可以查询出来
select * from t_student s
right join t_class c on s.c_id=c.c_id;
# 内连接：以两张表的数据为基准，只会查询出两张表有关系的数据
select * from t_student s
inner join t_class c on s.c_id=c.c_id;
```

7.4.8. 子查询

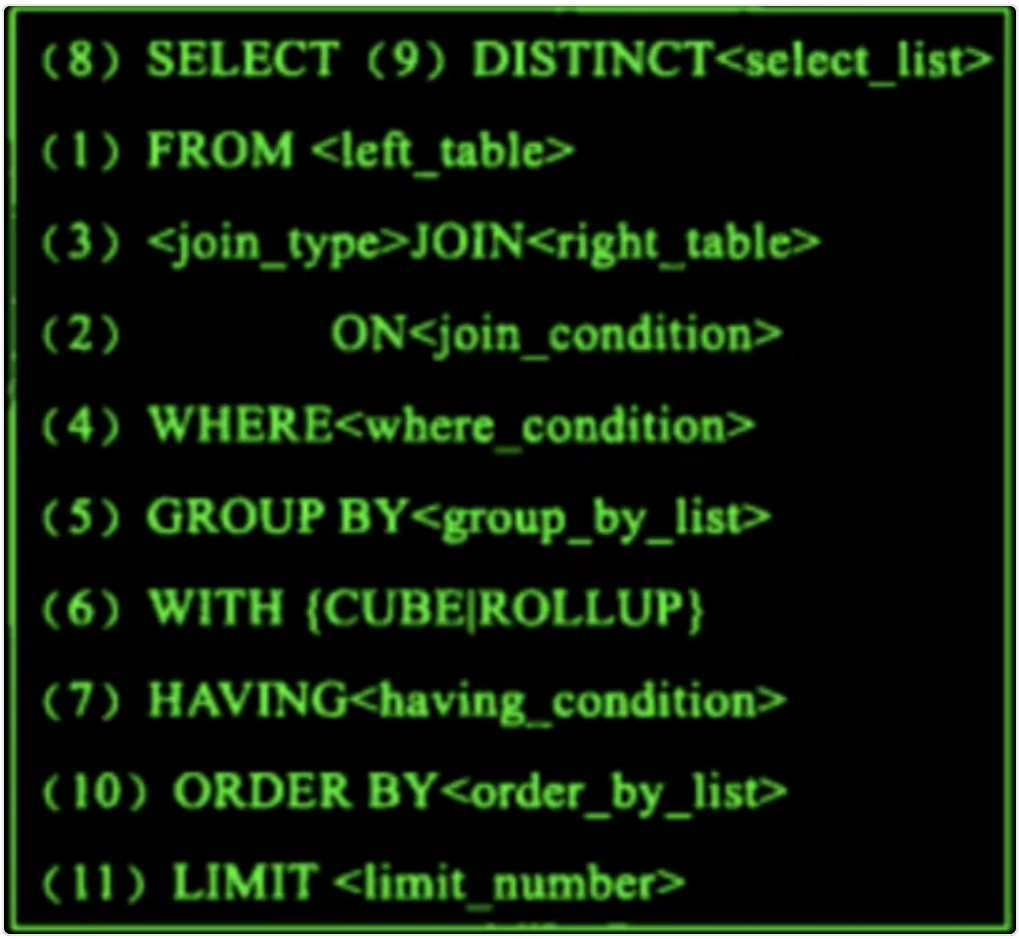
```
# 子查询总的来说可以分成两大类：
# 1. where子查询
# where 字段 in | not in (子查询)
# where 字段= | > | >= | < | <= | !=(子查询)
select cat_id from goods order by shop_price desc limit 1;
select * from category where cat_id=(select cat_id from goods order by
                                     shop_price desc limit 1);
# where exists(子查询)
select * from category where cat_id=1;
select * from goods where goods_id=1 and exists(select * from category whe
re
                                                cat_id=2);

# 2. from子查询
select * from goods order by goods_price desc limit 5;
select * from (select * from goods order by goods_price desc limit 5) as t
est;
```

7.4.9. 联合查询

```sql
# union
# 他的作用：将两个查询的结果集 组合成一个结果集
# 语法： 查询语句 union 查询语句;
select goods_id,goods_name from goods
UNION
SELECT cat_id,cat_name from category;
# 需要注意： union前后的查询的字段数，必须是相等的(和内容无关)，才能结合起来。
select goods_id,goods_name from goods
union
SELECT 1,2,3;
# union的结果集的结合的条件
# 条件就是：两次查询的结果的字段数必须一致
select * from t_class
union
select stu_id, stu_name from t_student;
# sql注入的地方
select * from t_class where class_id=-1
union
select 1,2;
```

8. SQL语句的执行顺序



下面我们来具体分析一下查询处理的每一个阶段：

- FORM: 对FROM的左边的表和右边的表计算笛卡尔积。产生虚表VT1
- ON: 对虚表VT1进行ON筛选，只有那些符合的行才会被记录在虚表VT2中。
- JOIN： 如果指定了OUTER JOIN（比如left join、 right join），那么保留表中未匹配的行就会作为外部
- 行添加到虚拟表VT2中，产生虚拟表VT3, rug from子句中包含两个以上的表的话，那么就会对上一个
- join连接产生的结果VT3和下一个表重复执行步骤1~3这三个步骤，一直到处理完所有的表为止。
- WHERE： 对虚拟表VT3进行WHERE条件过滤。只有符合的记录才会被插入到虚拟表VT4中。
- GROUP BY: 根据group by子句中的列，对VT4中的记录进行分组操作，产生VT5.
- CUBE | ROLLUP: 对表VT5进行cube或者rollup操作，产生表VT6.
- HAVING： 对虚拟表VT6应用having过滤，只有符合的记录才会被 插入到虚拟表VT7中。
- SELECT： 执行select操作，选择指定的列，插入到虚拟表VT8中。
- DISTINCT： 对VT8中的记录进行去重。产生虚拟表VT9.

- ORDER BY: 将虚拟表VT9中的记录按照<order_by_list>进行排序操作，产生虚拟表VT10.
- LIMIT：取出指定行的记录，产生虚拟表VT11, 并将结果返回。

9. 内置数据库

- MySQL 默认存在 4 个内置数据库
- **performance_schema**
  - 主要用于收集数据库服务器性能参数。
- **sys**
  - Sys 库所有的数据源来自：performance_schema。
  - 目标是把 performance_schema 的把复杂度降低，让 DBA 能更好的阅读这个库里的内容。
- **mysql**
  - mysql 的核心数据库，类似于 sql server 中的 master 表，主要负责存储数据库的用户、权限设置、关键字等 mysql 自己需要使用的控制和管理信息。

```
1    select host,user,authentication_string from user;
```

- **information_schema**（重点掌握）
  - information_schema 提供了访问数据库元数据的方式。
  - 元数据是关于数据的数据，如数据库名或表名，列的数据类型，或访问权限等。
  - 换句换说，information_schema 是一个信息数据库，它保存着关于 MySQL 服务器所维护的所有其他数据库的信息。
  - information_schema 有三张关键的表：**schemata**、**tables**、**columns**。
  - **schemata**
    - 提供了关于数据库中的库的信息。
    - 详细表述了某个库的名称，默认编码，排序规则。
    - 各字段说明如下：

字段	含义
schema_name	数据库名称
default_character_set_name	数据库编码

default_collation_name	数据库排序规则

```
1       select schema_name from information_schema.schemata;
```

- tables
  - 提供了关于数据库中的表的信息（包括视图）。
  - 详细表述了某个表属于哪个schema，表类型，表引擎，创建时间等信息。
  - 各字段说明如下：

字段	含义
table_catalog	数据表登记目录
table_schema	数据表所属的数据库名
table_name	表名称
.....	.....

```PL/SQL
-- 所有数据库的表名
mysql> select table_name from information_schema.tables;

-- 指定数据库的表名
mysql> select table_name from information_schema.tables where table_schema
= 'student';
```

```PL/SQL
select column_name from information_schema.columns where table_schema = 'st
udent' and table_name = 'users';
```

- columns
  - 提供了关于表中的列的信息。
  - 详细表述了某个列属于哪个表。
  - 各字段说明如下：

字段	含义

table_schema	表所有者（对于schema的名称）
table_name	表名
column_name	列名
.....	.....

10. 内置函数

10.1. 查看数据库版本

```
select version();
select @@version;
```

10.2. 获取数据库路径

```
#获取数据库数据存放的路径
select @@datadir;
#获取数据库路径
select @@basedir;
```

10.3. 获取数据库名

```
select database();
```

10.4. 获取用户名

```
select user();
```

10.5. 获取组合数据

- concat()：用于将多个字符串连接成一个字符串，形成单独一列，可能有多行。

- concat_ws()：用于将多个字符串连接成一个字符串，形成单独一列，可能有多行。
- group_concat()：返回一个字符串结果，该结果由分组中的值连接组合而成，形成单独一列，只有一行。

```
mysql> select concat(user_name,'-',user_age) from user;
+------------------------------+
| concat(user_name,'-',user_age) |
+------------------------------+
| jack-18                      |
| rose-17                      |
| lucy-22                      |
| tom-19                       |
| David-22                     |
| Allen-23                     |
+------------------------------+
6 rows in set (0.00 sec)

mysql> select concat_ws('-',user_name,user_age) from user;
+---------------------------------+
| concat_ws('-',user_name,user_age) |
+---------------------------------+
| jack-18                         |
| rose-17                         |
| lucy-22                         |
| tom-19                          |
| David-22                        |
| Allen-23                        |
+---------------------------------+
6 rows in set (0.00 sec)

mysql> select group_concat(user_name,'-',user_age) from user;
+-------------------------------------------------+
| group_concat(user_name,'-',user_age)            |
+-------------------------------------------------+
| jack-18,rose-17,lucy-22,tom-19,David-22,Allen-23 |
+-------------------------------------------------+
1 row in set (0.00 sec)
```

10.6. 获取字节长度

```
mysql> select length('123456');
+------------------+
| length('123456') |
+------------------+
|                6 |
+------------------+
1 row in set (0.00 sec)

#获取字符的长度
mysql> select char_length('国科');
+----------------------+
| char_length('国科')   |
+----------------------+
|                    2 |
+----------------------+
1 row in set (0.00 sec)
```

10.7. 编码转换

- ord()：字符串转 ASCII 编码
- char(97)：ASCII 编码转字符串
- hex()：转 十六 进制
- bin()：转 二 进制

```
mysql> select ord('a');
+----------+
| ord('a') |
+----------+
|       97 |
+----------+
1 row in set (0.00 sec)

mysql> select char(98);
+----------+
| char(98) |
+----------+
| b        |
+----------+
1 row in set (0.00 sec)

mysql> select hex('c');
+----------+
| hex('c') |
+----------+
| 63       |
+----------+
1 row in set (0.00 sec)

mysql> select bin(10);
+---------+
| bin(10) |
+---------+
| 1010    |
+---------+
1 row in set (0.00 sec)
```

10.8. 字符截取

- **substr()：字符截取（重点）**
- left()：从左开始字符截取
- right()：从右开始字符截取

```
mysql> select substr('yongz',1,1);
+--------------------+
| substr('yongz',1,1) |
+--------------------+
| y                  |
+--------------------+
1 row in set (0.00 sec)

mysql> select left('yongz',1);
+-----------------+
| left('yongz',1) |
+-----------------+
| y               |
+-----------------+
1 row in set (0.00 sec)

mysql> select right('yongz',1);
+------------------+
| right('yongz',1) |
+------------------+
| z                |
+------------------+
1 row in set (0.00 sec)
```

10.9. 睡眠/延时函数

```
mysql> select sleep(3);
+----------+
| sleep(3) |
+----------+
|        0 |
+----------+
1 row in set (3.01 sec)
```

10.10. 判断

```
mysql> select if(substr(database(),1,1)='s','是','不是');
+-----------------------------------------------+
| if(substr(database(),1,1)='s','是','不是')    |
+-----------------------------------------------+
| 是                                            |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

11. 开启远程权限

- 查看当前 MySQL 用户：

```
mysql> select Host,User from mysql.user;
+-----------+---------------+
| Host      | User          |
+-----------+---------------+
| localhost | mysql.session |
| localhost | mysql.sys     |
| localhost | root          |
+-----------+---------------+
3 rows in set (0.00 sec)
```

- 对 root 添加一个新的授权：

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'root' WITH GRANT OPTION;
```

- 刷新MySQL权限表以使更改生效：

```
FLUSH PRIVILEGES;
```

- 当然也可以创建一个新用户并给予授权：

```
CREATE USER 'winner'@'%' IDENTIFIED BY '123456';
GRANT ALL PRIVILEGES ON *.* TO 'winner'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

11.1.1. 案例

```sql
-- 商品类别表 --
CREATE TABLE category(
  cat_id INT PRIMARY KEY AUTO_INCREMENT,#类别编号
  cat_name VARCHAR(30) NOT NULL#类别名称
);
-- 商品表 --
CREATE TABLE goods(
  goods_id INT PRIMARY KEY AUTO_INCREMENT,#商品编号
  goods_name VARCHAR(30) NOT NULL,#商品名称
  goods_price DOUBLE,#商品进价
  shop_price DOUBLE,#商品卖价
  market_price DOUBLE,#市场价
  cat_id INT,#商品类别
  goods_number INT,#商品数量
  FOREIGN KEY(cat_id) REFERENCES category(cat_id)
);

INSERT INTO category(cat_name) VALUES('航模'),('车模'),('船模');
INSERT INTO category(cat_name) VALUES('动物模型');
INSERT INTO goods (goods_name,goods_price,shop_price,market_price,cat_id,goods_number)
VALUES('F16战斗机',300,1000,900,1,120);
INSERT INTO goods
(goods_name,goods_price,shop_price,market_price,cat_id,goods_number)
VALUES('F35战斗机',400,1200,1000,1,210);
INSERT INTO goods
(goods_name,goods_price,shop_price,market_price,cat_id,goods_number)
VALUES('F117隐形轰炸机',290,800,600,1,99);
INSERT INTO goods
(goods_name,goods_price,shop_price,market_price,cat_id,goods_number)
VALUES('牧马人',120,600,500,2,1200);
INSERT INTO goods
(goods_name,goods_price,shop_price,market_price,cat_id,goods_number)
VALUES('宝马Z4',130,560,510,2,231);
INSERT INTO goods
(goods_name,goods_price,shop_price,market_price,cat_id,goods_number)
VALUES('地中海帆船',90,300,180,3,68);
INSERT INTO goods
(goods_name,goods_price,shop_price,market_price,cat_id,goods_number)
VALUES('密西西比号蒸汽明轮',100,560,520,3,114);
INSERT INTO goods
(goods_name,goods_price,shop_price,market_price,cat_id,goods_number)
VALUES('德鲁伊号16门炮护卫舰',1322,2322,2600,3,100);
INSERT INTO goods
(goods_name,goods_price,shop_price,market_price,cat_id,goods_number)
VALUES('皇家理查德号 74门炮战舰',350,800,769,3,312);

```

```
1.求每个类别下商品种类数
select cat_id, count(*) as count from goods group by cat_id;

2.查询本店每个商品价格比市场价低多少;
select market_price-shop_price as cha_price from goods;

3.查询每个类别下面积压的货款
货款:进价*数量
select cat_id, sum(goods_price*goods_number) as total_price from goods group by cat_id;

4.查询本店商品价格比市场价低多少钱,输出低200元以上的商品
select market_price-shop_price as cha_price  from goods having cha_price > 200;

5.查询积压货款超过2万元的栏目,以及该栏目积压的货款
select goods_name, goods_price*goods_number as total_price from goods having total_price > 20000;

6.按类别号升序排列,每个类别下的商品进价降序排列
select * from goods order by cat_id asc, goods_price desc;

7.取价格第1-6高的商品
select * from goods order by shop_price desc limit 6;

8.查询每个类别下进价最高的商品
select cat_id, max(goods_price) as max_goods_price from goods group by cat_id;
```

12. 作业

12.1. 作业1

```
1) 创建一张学生表，包含以下信息，学号，姓名，年龄，性别，联系电话,学历,出生日期
table_name: student
学号: id int primary key auto_increment
姓名: stu_name varchar(10) not null
年龄: stu_age int not null
性别: stu_gender char(1) not null
联系电话: stu_phone bigint not null
学历: stu_edu varchar(5)
出生年月: stu_birth date
engine=innodb default charset=utf8;

2) 向学生表添加如下信息:
学号 姓名 年龄 性别 联系电话 学历 出生日期
1 A张三 22 男 123456 小学 1993-09-09
2 B李四 21 男 119 中学 1994-09-01
3 C王五 23 男 150 高中 1992-04-22
4 D赵六 18 女 120 大学 1995-01-28
5 E孙七 17 女 911 大专 1996-01-28
6 C郑八 24 男 12580 中专 1990-01-28

3) 修改学生表的数据，将电话号码以11开头的学员的学历改为“大专”
修改: update
以什么开头:  需要用到模糊查询 like like "11%"

4) 删除学生表的数据，姓名以C开头，性别为‘男'的记录删除

5) 将所有年龄小于22岁的，学历为“大专”的学生的电话删除

6) 修改C开头，并且学历为高中的学生出生日期为2013-09-18

7) 删除出生日期在（1990年-1992年，包括1990以及1992年）的学生信息范围:

8) 添加一名未知电话的同学“ccf”

9) 修改ccf同学的出生年月为1924-08-08
```

12.2. 作业2

```
班级表：t_class
编号:c_id 整型 主键 自增
名称：c_name 字符串 不能为空 不允许重复
学生表：t_student

编号s_id 整形 主键 自增
姓名：s_name 字符串 不允许重复
性别：s_sex 字符串 默认值 男
年龄：s_age 整型
班级编号：s_class_id 整形 外键 指向班级表的班级编号

-- 建表 并添加数据，并完成以下题目
-- 向t_class中添加数据
INSERT INTO t_class VALUES
(NULL,1),
(NULL,2),
(NULL,3),
(NULL,4);
-- 向t_student中添加数据
INSERT INTO t_student VALUES
(1,'刘基','男',22,4),
(2,'吴丽','女',18,1),
(3,'马伯伯','男',55,2),
(4,'祝枝山','男',60,2),
(5,'马莹','女',9,1),
(6,'李蕾蕾','女',20,3),
(7,'王二娃','男',33,3),
(8,'李晓','男',28,4),
(9,'马强','男',30,4),
(10,'韩璐','女',26,4)

1.把刘基的名字修改为刘伯温
2.唐伯虎年龄20，性别男，班级3，添加到表中
3.查询出所有姓名包括伯的所有的人员的信息
4，查询年龄在10-20之间的所有人员的信息
5，查询年龄在10-20之间的所有人员前5条的信息 并且将查询出的每个人的年龄加10
# 要对列进行运算： MySQL是支持列运算的
# as 是别名 将 s_age+10 别名为 age
# as 是可以省略的，通常我们尽量不要去省略，这样可读性更高

6,查询年龄在10岁以下或20岁以上的所有人员的姓名和年龄前5条的记录，
并且将查询出的每个人的年龄加10，并且给每个字段起一个别名
```

12.3. 作业3

```
一.商品销售记录表
id 商品编号 销售日期 销售数量 商品单价 销售总金额 销售员工
1 xsl001 2013/12/2 124 134.5 16678 张三
2 xsl002 2013/12/2 50 80 4000 李四
3 xsl003 2013/12/5 66 55 3630 张三
4 xsl001 2013/11/20 10 134.5 1345 张三
5 xsl001 2013/11/2 20 134.5 2690 王五
6 xsl002 2013/11/5 30 80 2400 张三
7 xsl002 2013/11/9 23 80 1840 王五
8 xsl003 2013/12/11 10 55 550 李四
9 xsl003 2013/12/12 50 55 2750 王五
10 xsl004 2013/11/30 45 100 4500 张三

 (1) 创建商品销售记录表
CREATE TABLE t_shop (
  id INT PRIMARY KEY AUTO_INCREMENT,
  goods_no VARCHAR(10),
  sale_date DATE,
  sale_count INT,
  goods_price DECIMAL(10,2),
  sale_money DECIMAL(10,2),
  sale_empl VARCHAR(10)
);

 (2) 插入数据
insert into t_shop values(null,"xsl001","2013/12/2",124,134.5,16678,"张三"
);
INSERT INTO t_shop VALUES(NULL,"xsl002","2013/12/2",50,80,4000,"李四");
INSERT INTO t_shop VALUES(NULL,"xsl003","2013/12/5",66,55,3630,"张三");
INSERT INTO t_shop VALUES(NULL,"xsl001","2013/11/20",10,134.5,1345,"张三")
;
INSERT INTO t_shop VALUES(NULL,"xsl001","2013/11/2",20,134.5,2690,"王五");
INSERT INTO t_shop VALUES(NULL,"xsl002","2013/11/5",30,80,2400,"张三");
INSERT INTO t_shop VALUES(NULL,"xsl002","2013/11/9",23,80,1840,"王五");
INSERT INTO t_shop VALUES(NULL,"xsl003","2013/12/11",10,55,550,"李四");
INSERT INTO t_shop VALUES(NULL,"xsl003","2013/12/12",50,55,2750,"王五");
INSERT INTO t_shop VALUES(NULL,"xsl004","2013/11/30",45,100,4500,"张三");
 (3) 完成下列sql语句
1.查询张三的所有销售记录
2.查询张三12月份的销售记录
3.查询销售总金额大于2000 的12月份销售记录
4.查询前10条销售记录
5.查询前10条销售记录中商品编号xsl001的记录
6.查询销售数量大于20 销售人员为李四的记录
7.查询前5条 销售人员为王五的记录，只显示 商品编号 销售总金额 销售人员 这些列（要求列名
用中文别名显出）
8.查询2013年11月20日之后 2013年12月10日之前的记录
```

```
45    9.查询从第三条数据开始，到第10条数据结束的记录，要求商品单价大于100 或则销售数量大于50
```

12.4. 作业4

```
姓名 年龄 性别 学号 成绩 班级
Jacky 20 男 xh1001 90 T01
Simth 30 男 xh1002 75 T02
Jay 18 男 xh1003 80 T01
Helen 19 女 xh1004 75 T02
lily 22 女 xh1005 90 T03
Green 23 男 xh1006 85 T02
RedChar 18 男 xh1007 60 T01
Kevin 17 女 xh1008 45 T03

表名: student2
id int primary key auto_increment
stu_name varchar(10) not null
stu_age int not null
stu_gender char(1) not null
stu_num varchar(10) not null
stu_code int not null
stu_class varchar(5) not null
插入数据:
INSERT INTO student2 VALUES (NULL,"Jacky",20,"男","xh1001",90,"T01");
INSERT INTO student2 VALUES (NULL,"Simth",30,"男","xh1002",75,"T02");
INSERT INTO student2 VALUES (NULL,"Jay",18,"男","xh1003",80,"T01");
INSERT INTO student2 VALUES (NULL,"Helen",19,"女","xh1004",75,"T02");
INSERT INTO student2 VALUES (NULL,"lily",22,"女","xh1005",90,"T03");
INSERT INTO student2 VALUES (NULL,"Green",23,"男","xh1006",85,"T02");
INSERT INTO student2 VALUES (NULL,"RedChar",18,"男","xh1007",60,"T01");
INSERT INTO student2 VALUES (NULL,"Kevin",17,"女","xh1008",45,"T03");

1.统计每个班的学员的数量

2.统计每个班的总分

3.统计每个班的平均分

4.统计每个班的最高分

5.统计每个班的最低分

6.统计每个班学员的数量,总分,平均分,最高分,最低分

8.统计班级ID为T01的学员的数量,总分,平均分,最高分,最低分

9.查询平均分上85的班级有哪些?

10.查询有女生的班级是哪些?
```

12.5. 作业5

```
/*
商品表
create table t_shop(
s_id int primary key auto_increment,
s_shopcode varchar(30), -- 商品编号
s_name varchar(40), -- 商品名称
s_price int , -- 商品价格
s_class varchar(50) -- 商品类别
);
insert into t_shop(s_shopcode,s_name,s_price,s_class) values
('n11','橙子',9,'水果'),
('x330','血橙',11,'水果'),
('yx673','柚子',7,'水果'),
('n12','白菜',2,'蔬菜'),
('a13','冬瓜',3,'蔬菜'),
('n14','西瓜',4,'水果'),
('n15','丝瓜',5,'蔬菜'),
('c16','苦瓜',6,'蔬菜'),
('m17','南瓜',5,'蔬菜'),
('d18','茄子',6,'蔬菜');
*/
-- 1 查询所有包含瓜的商品名称信息

-- 2 查询价格在1 到8 的所有商品信息

-- 3 查询商品的最高价格的值是多少

-- 4 查询商品价格最高的前三个商品的信息

-- 5 查询所有商品的平均价格

-- 6 查询所有包含瓜的商品的平均价格

-- 7 查询最高商品的价格是最低商品的价格的倍数是多少

-- 8 查询商品名称中包含橙字的有多少个商品

-- 9 修改 西瓜的价格为2块

-- 10 删除id 为，4,9,1 的商品信息

-- 11 查询蔬菜类别中最高的价格是多少?
```

MYSQL作业练习，提权码：GOKT

百度网盘为您提供文件的网络备份、同步和分享服务。空间大、速度快、安全稳固，支持教育网加速，…

https://pan.baidu.com/s/1wjkKsVnH5CbU_HBHe5zMgw