

day18-MySQL基础

今日内容

- 数据库安装
- DDL-----对数据库的增删查改,对表的增删查改
- DML----对记录的增删改 重点
- DQL----对记录的查询 重点

学习目标

- ☐ 能够理解数据库的概念
- ☐ 能够安装MySQL数据库
- ☐ 能够使用SQL语句操作数据库
- ☐ 能够使用SQL语句操作表结构
- ☐ 能够使用SQL语句进行数据的添加修改和删除的操作
- ☐ 能够使用SQL语句进行条件查询数据
- ☐ 能够使用SQL语句进行排序
- ☐ 能够使用聚合函数
- ☐ 能够使用SQL语句进行分组查询
- ☐ 能够使用SQL语句进行分页查询

第一章-数据库概述

知识点-数据库的介绍

1.目标

- ☐ 知道什么是数据库

2.分析

目前来说如果我们要进行数据存储，有几种方式：

1. 我们可以使用集合等方式将数据保存在内存中，但是数据不能持久化保存，断电/程序退出，数据就清除了
2. 我们还可以将数据保存在普通文件中，可以持久化保存，但是查找，增加，修改，删除数据比较麻烦，效率低

所以我们需要一个既可以持久化保存数据又可以方便操作的地方来存储数据，这就是我们接下来要给大家介绍的数据库

3.讲解

3.1什么是数据库

数据库 (DataBase, DB) : 指长期保存在计算机的存储设备(硬盘)上, 按照一定规则组织起来, 可以被各种用户或应用共享的数据集合. 还是以文件的方式存在服务器的电脑上的。

说白了就是数据的仓库, 用来持久化保存数据的。

3.2常见的关系型数据库

- MySQL : **开源免费**的数据库, 中小型数据库, 已经被Oracle收购了。MySQL6.x版本也开始收费。后来Sun公司收购了MySQL, 而Sun公司又被Oracle收购
- Oracle: 收费的大型数据库.Oracle公司的产品.Oracle收购SUN公司, 收购MySQL.
- DB2: IBM公司的数据库产品,收费的.银行系统中.
- SQLServer: MS公司.收费的中型的数据库.
- SyBase: 已经淡出历史舞台.提供了一个非常专业数据建模的工具PowerDesigner.
- SQLite: 嵌入式的小型数据库,应用在手机端.



4.小结

1. 为什么要学习数据库?

把数据保存到文件, 内存里面,都要一些缺点. 保存到数据库里面解决这些缺点.

2. 什么数据库?

数据的仓库, 用来保存数据的

3. 常见的关系型数据库?

MySQL

Oracle

知识点-数据库结构【重点】

1.目标

- ☐ 掌握数据库结构

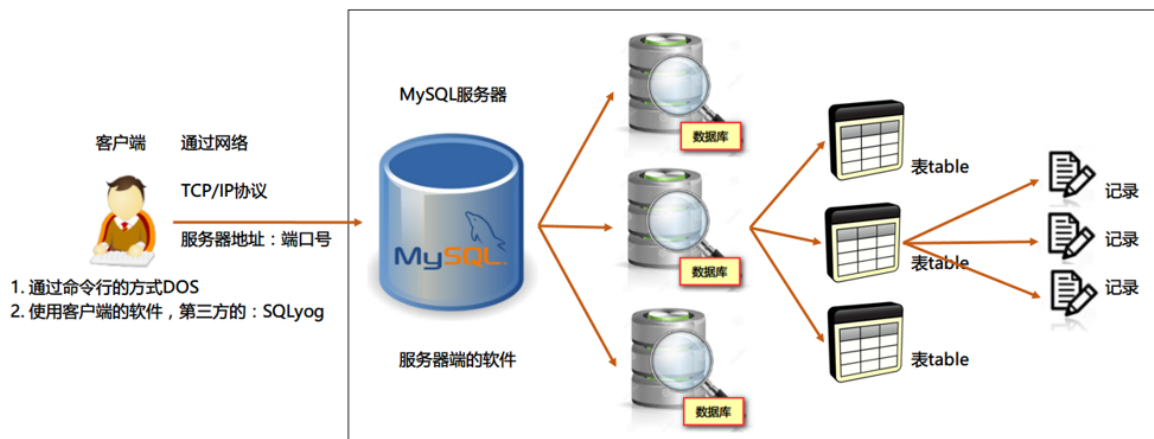
2.分析

数据库是用来存储数据的,那么到底通过什么样的方式来存的. 结构是怎么样的呢?

3.讲解

数据库管理程序(DBMS)可以管理多个数据库,一般开发人员会针对每一个应用创建一个数据库。为保存应用中实体的数据,一般会在数据库创建多个表,以保存程序中实体的数据。

数据库管理系统、数据库和表的关系如图所示:



4.小结

- 1.一般情况下,一个系统(软件,项目) 就设计一个数据库;
- 2.一个数据库里面有多(≥ 1)张表. 一个实体(java类)对应一张表
- 3.一张表里面有多条(≥ 1)记录, 一个对象对应一条记录

第二章-数据库的安装,卸载,启动,登录

实操-MySQL的安装,卸载

1.目标

- ☐ 掌握MySQL的安装和卸载

2.路径

我们要使用MySQL数据库,就需要先安装. 如果第一次安装失败了,就需要卸载,再安装.

1. MySQL的安装
2. MySQL的卸载


3.讲解

3.1MySQL的安装

具体参考文档: [资料\01_MySql安装与卸载\MySQL安装图解.docx](#)

3.2 MySQL的卸载

具体参考文档: [资料\01_MySql安装与卸载\MySQL卸载手册.doc](#)

 MySQL卸载手册.doc

4.小结

- 安装需要注意的地方: 安装路径不要有==空格和中文==,对着文档装
- 卸载需要注意的地方
 - 去360/软件管家或者控制面板卸载(删除之前先找到两个文件夹)
 - 一定要删除两个文件夹 (数据库安装路径和数据存放路径, 这两个文件夹在配置文件里面my.ini)

```
74 basedir="E:/worksoft/mysql/"
75
76 #Path to the database root
77 datadir="C:/ProgramData/MySQL/MySQL Server 5.5/Data/"
78
```

实操-数据库服务的启动和登录,退出

1.目标

我们刚刚把数据库安装成功了,下面就需要进行数据库的启动,登录和退出

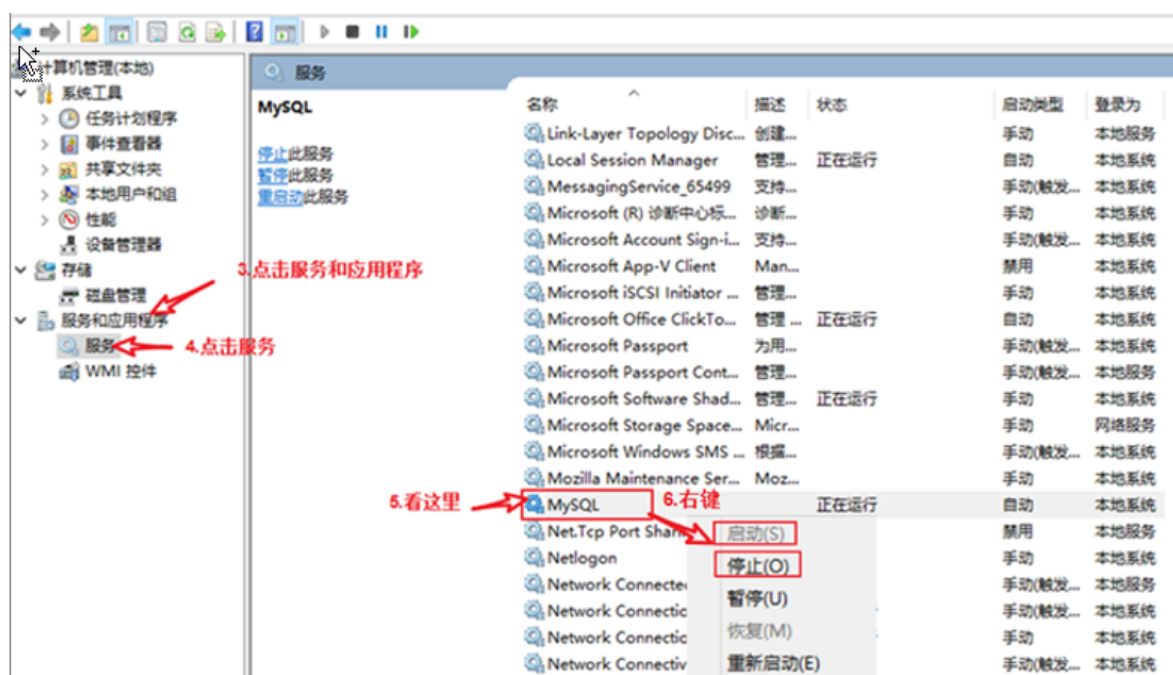
2.步骤

1. 数据库服务的启动
2. MySQL的登录(命令行和客户端方式)
3. MySQL的退出

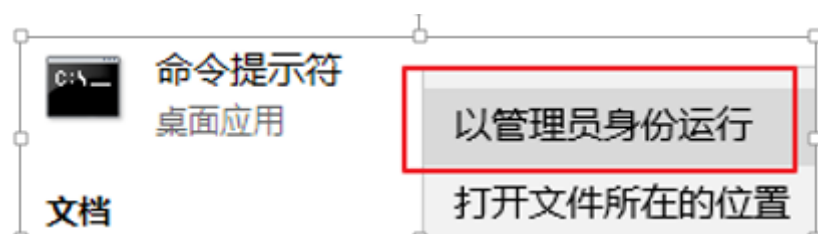
3.讲解

3.1数据库服务的启动

3.1.1 方式一通过界面启动



3.1.2方式二通过DOS命令方式启动



```
C:\windows\system32>net start mysql
MySQL 服务正在启动 .
MySQL 服务已经启动成功。

C:\windows\system32>net stop mysql
MySQL 服务正在停止 .
MySQL 服务已成功停止。
```

MySQL是一个需要账户名密码登录的数据库，登陆后使用，它提供了一个默认root账号，使用安装时设置的密码即可登录。

3.2 登录

3.2.1 命令行

```
mysql -u 用户名 -p          #然后再输入密码。
mysql -u用户名 -p密码
```

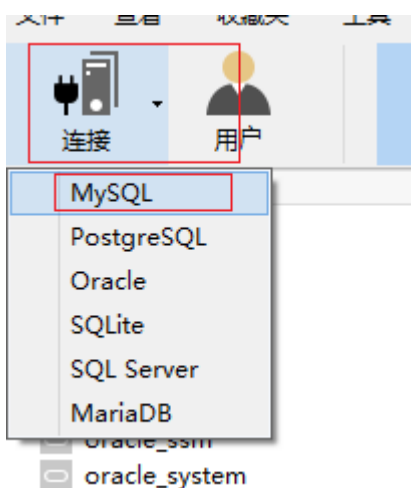
3.2.2 图形化工具

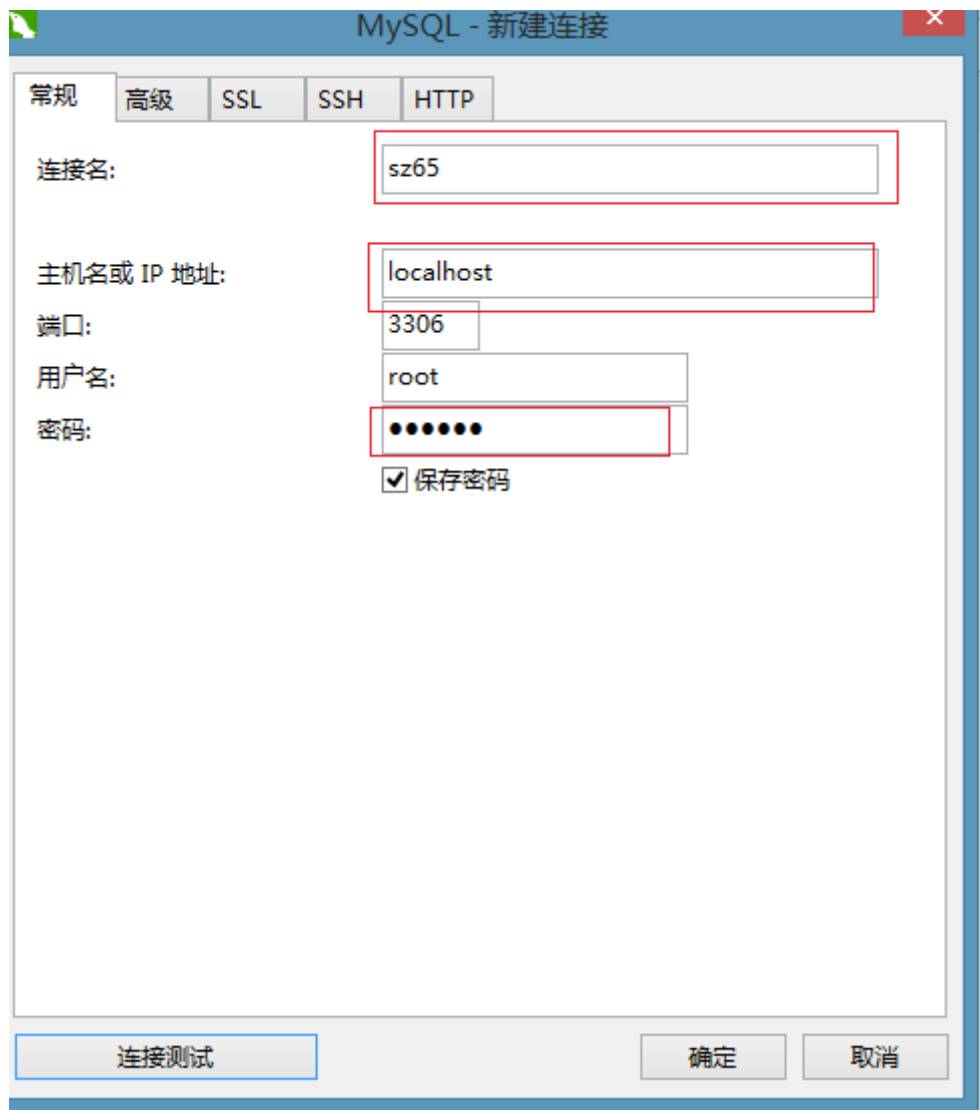
- 安装

navicat111_premium_cs_x64.exe

Navicat Premium 11.1.\$破解注册机

- 连接





2.3退出

2.3.1命令行

输入: quit或exit

2.3.2图形化工具

- 右键关闭

4.小结

1. MySql服务启动: ==建议大家开机就启动==
2. dos连接

```
mysql -u root -p
再输入密码
-----
mysql -uroot -p密码
```

3. navicat图形化工具, 会连接就OK

第三章-SQL概述

知识点-Sql介绍

1.目标

- ☐ 掌握什么是SQL

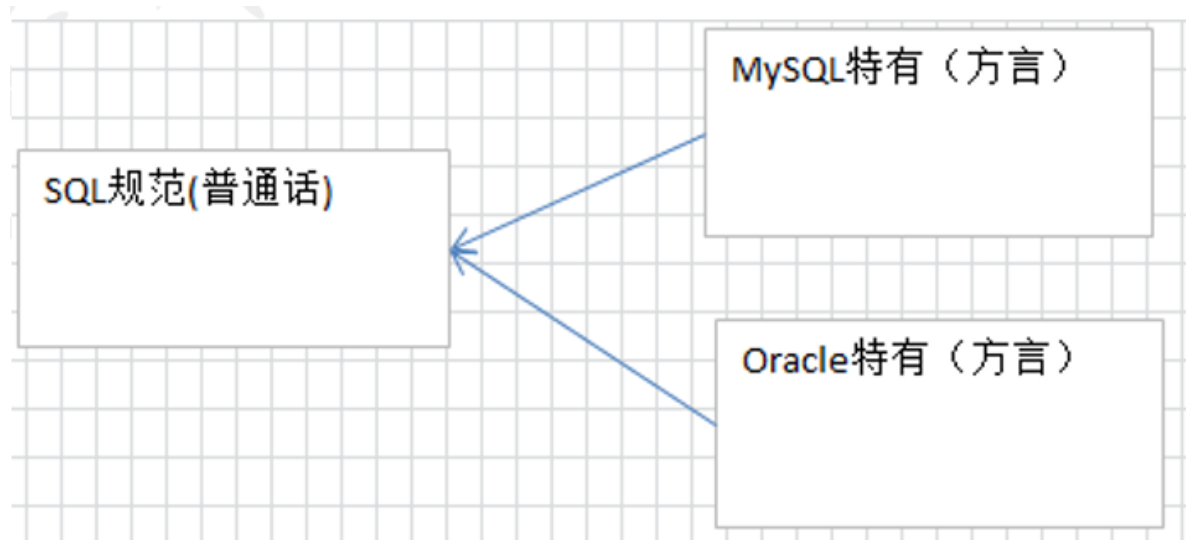
2.路径

1. 介绍Sql的概念
2. sql的语法的注意事项
3. 介绍sql的类别

3.讲解

3.1.什么是sql?

- SQL: Structure Query Language。 (结构化查询语言) ,通过sql操作数据库(操作数据库,操作表,操作数据)
- SQL被美国国家标准局 (ANSI) 确定为关系型数据库语言的美国标准, 后来被国际化标准组织 (ISO) 采纳为关系数据库语言的国际标准
- 各数据库厂商(MySql,oracle,sql server)都支持ISO的SQL标准。
- 各数据库厂商在标准的基础上做了自己的扩展。 各个数据库自己特定的语法



3.2sql的语法

- 每条语句以分号结尾(命令行里面需要), 如果在navicat,java代码中不是必须加的。
- SQL在window中不区分大小写, 关键字中认为大写和小写是一样的

3.3sql的分类

- Data Definition Language (DDL数据定义语言) 如: 操作数据库, 操作表
- Data Manipulation Language(DML数据操纵语言), 如: 对表中的记录操作增删改
- Data Query Language(DQL 数据查询语言), 如: 对表中的记录查询操作
- Data Control Language(DCL 数据控制语言), 如: 对用户权限的设置

4.小结

1. 我们工作里面用的最多的是 DML和DQL , 对数据的增删改查
2. SQL 结构化查询语言, 用SQL来操作数据库

第四章-DDL操作数据库

知识点-DDL操作数据库

1.目标

我们把Sql介绍完成了, 那下面就通过DDL操作数据库

2.步骤

- 创建数据库(掌握)
- 查看数据库
- 删除数据库(掌握)
- 修改数据库
- 数据库的其它操作(掌握)

3.讲解

3.1创建数据库

- 语法

```
create database 数据库名 [character set 字符集][collate 校对规则] 注: []意思是可选的意思
```

字符集(charset): 是一套符号和编码。

- 练习

创建一个day16的数据库 (默认字符集)

```
create database web14_1;
```

创建一个day16_2的数据库,指定字符集为gbk(了解)

```
create database web14_2 character set gbk;
```

3.2查看所有的数据库

3.2.1查看所有的数据库

- 语法

```
show databases;
```

3.2.2查看数据库的定义结构【了解】

- 语法

```
show create database 数据库名;
```

- 查看web14_1这个数据库的定义

```
show create database web14_1;
```

3.3删除数据库

- 语法

```
drop database 数据库名;
```

- 删除web14_2数据库

```
drop database web14_2;
```

3.4修改数据库【了解】

- 语法

```
alter database 数据库名 character set 字符集;
```

- 修改web14_1这个数据库的字符集(gbk)

```
alter database web14_1 character set gbk;
```

注意:

- 是utf8, 不是utf-8
- 不是修改数据库名

3.5其他操作

- 切换数据库, 选定哪一个数据库

```
use 数据库名;           //注意: 在创建表之前一定要指定数据库. use 数据库名
```

- 练习: 使用web14_1

```
use web14_1;
```

- 查看正在使用的数据库

```
select database();
```

4.小结

1. 创建数据库

```
create database 数据库的名字;
```

2. 删除数据库

```
drop database 数据库的名字;
```

3. 其它操作

```
use 数据库的名字;    -- 切换到某个数据库
select database();    -- 查看正在使用哪个数据库
```

第五章-DDL操作表

知识点-创建表【重点】

1.目标

我们第四章已经把数据库的CRUD讲解完了,下面我们就学习创建表

2.步骤

- 创建表的语法介绍
- MySql常见的类型
- MySql约束

3.讲解

3.1语法

```
create table 表名(
    字段名 字段类型 [约束],
    字段名 字段类型 [约束],
    .....
    字段名 字段类型 [约束]
);
```

3.2 类型

分类	数据类型	说明
数值类型	TINYINT [UNSIGNED] [ZEROFILL] BOOL, BOOLEAN SMALLINT [UNSIGNED] [ZEROFILL] INT [UNSIGNED] [ZEROFILL] BIGINT [UNSIGNED] [ZEROFILL] FLOAT[(M,D)] [UNSIGNED] [ZEROFILL] DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]	带符号的范围是-128到127。无符号0到255。 使用0或1表示真或假 2的16次方 2的32次方 2的64次方 M指定显示长度, d指定小数位数 表示比float精度更大的小数
文本、二进制类型	CHAR(size) char(20) VARCHAR(size) varchar(20) BLOB LONGBLOB TEXT(clob) LONGTEXT(longclob)	固定长度字符串 可变长度字符串 二进制数据 大文本
时间日期	DATE/DATETIME/TimeStamp	日期类型(YYYY-MM-DD) (YYYY-MM-DD HH:MM:SS), TimeStamp表示时间戳, 它可用于自动记录insert、update操作的时间

1. 整型 一般使用int 或者bigint

2. 浮点/双精度型

- 默认的范围 float或者double
- 指定范围 float(M,D) eg: float(4,2) 表达的范围: -99.99~99.99

3. 字符串

- 固定长度 char(n) eg: char(20), 最大能存放20个字符. 'aaa', 还是占20个字符的空间
- 可变长度 varchar(n) eg:varchar(20), 最大能存放20个字符. 'aaa', 占3个字符的空间

一般使用varchar(n) 节省空间; 如果长度(eg:身份证)是固定的话 可以使用char(n) 性能高一点

4. 关于大文件

- 一般在数据库里面很少存文件的内容, 一般存文件的路径
- 一般不使用二进制存, 使用varchar(n)存文件的路径

5. 日期

- DATE 只有日期
- DATETIME 日期和时间

3.3 约束

- 即规则,规矩 限制;
- 作用: 保证用户插入的数据保存到数据库中是符合规范的

约束	约束关键字
主键	primary key
唯一	unique
非空	not null

约束种类:

- not null: 非空 ; eg: username varchar(40) not null username这个字段不能为空,必须要有数据
- unique:唯一约束, 后面的数据不能和前面重复; eg: cardNo char(18) unique; cardNo字段不能出现重复的数据
- primary key; 主键约束(非空+唯一); 一般用在表的id列上面. 一张表基本上都有id列的, id列作为唯一标识的
 - auto_increment: ==自动增长,必须是设置了primary key之后,才可以使用auto_increment==

- id int primary key auto_increment; id不需要我们自己维护了, 插入数据的时候直接插入null, 自动的增长进行填充进去, 避免重复了.

注意:

1. 先设置了primary key 再能设置auto_increment
2. 只有当设置了auto_increment 才可以插入null 自己维护 否则插入null会报错

id列:

1. 给id设置为int类型, 添加主键约束, 自动增长
2. 或者给id设置为字符串类型, 添加主键约束, 不能设置自动增长

3.4练习

- 创建一张学生表(含有id字段,姓名字段,性别字段. id为主键自动增长)

```
create table student(  
    id int primary key auto_increment,  
    name varchar(40),  
    sex int  
);
```

4.小结

1. 语法

```
create table 表名(  
    列 类型 【约束】,  
    列 类型 【约束】  
);
```

2. 类型

- char(n) 固定长度
- varchar(n) 可变长度

3. 约束

- not null 非空
- unique 唯一
- primary key 主键(非空+唯一)
 - auto_increment 自动增长

4. id

- 可以设置成int类型, 设置成primary key, 添加auto_increment
- 作为记录唯一标识

知识点-查看表【了解】

1.目标

我们把表创建好了, 下面就来介绍查看表

2.步骤

- 查看当前数据库所有的表
- 查看表的定义结构

3.讲解

3.1查看所有的表

```
show tables;
```

3.2查看表的定义结构

- 语法
desc 表名;
- 练习: 查看student表的定义结构

```
desc student;
```

小结

知识点-修改表【掌握】

1.目标

我们表创建好了, 如果要增加一列,要删除一列呢? 那下面就来讲解修改表

2.步骤

- 增加列
- 修改列的类型约束
- 修改列的名称, 类型, 约束
- 删除列
- 修改表名

3.讲解

3.1语法

- 增加一列; alter table 表 add 字段 类型 约束;
- 修改列的类型约束; alter table 表 modify 字段 类型 约束;
- 修改列的名称, 类型, 约束; alter table 表 change 旧列 新列 类型 约束;
- 删除一列; alter table 表名 drop 列名;
- 修改表名; rename table 旧表名 to 新表名;

3.2练习

- 给学生表增加一个grade字段

```
alter table student add grade varchar(20) not null;
```

- 给学生表的sex字段改成字符串类型

```
alter table student modify sex varchar(10);
```

- 给学生表的grade字段修改成class字段

```
alter table student change grade class varchar(20);
```

- 把class字段删除

```
alter table student drop class;
```

- 把学生表修改成老师表(了解)

```
rename table student to teacher
```

4.小结

1. 都是以==alter table 表名==打头
 - 增加列 add
 - 修改列的类型 约束 modify
 - 修改列的名字 类型 约束 change
 - 删除列 drop
2. 敲一遍就行了

知识点-删除表【掌握】

1.目标

表创建好了,我们还可以删除。掌握表的删除

2.步骤

- 删除表

3.讲解

- 语法
drop table 表名;
- 把teacher表删除

```
drop table teacher;
```

4.小结

1. 删除表语法

```
drop table 表名;
```

第六章-DML操作表记录-增删改【重点】

- 准备工作: 创建一张商品表(商品id,商品名称,商品价格,商品数量.)

```
create table product(  
    pid int primary key auto_increment,  //只有设置了auto_increment id列才可以赋值为  
    null  
    pname varchar(40),  
    price double,  
    num int  
);
```

知识点-插入记录

1.目标

- 掌握插入记录

2.步骤

- 两种方式插入数据
- 使用命令行操作时候乱码的解决

3.讲解

- 方式一: 插入指定列, ==如果没有把这个列进行列出来, 以null进行自动赋值了==.
eg: 只想插入pname, price , insert into t_product(pname, price) values('mac',18000);

```
insert into 表(列, 列..) values(值, 值..);
```

注意: 如果没有插入了列设置了非空约束, 会报错的

- 方式二: 插入所有的列

```
insert into 表 values(值, 值....);
```

eg:

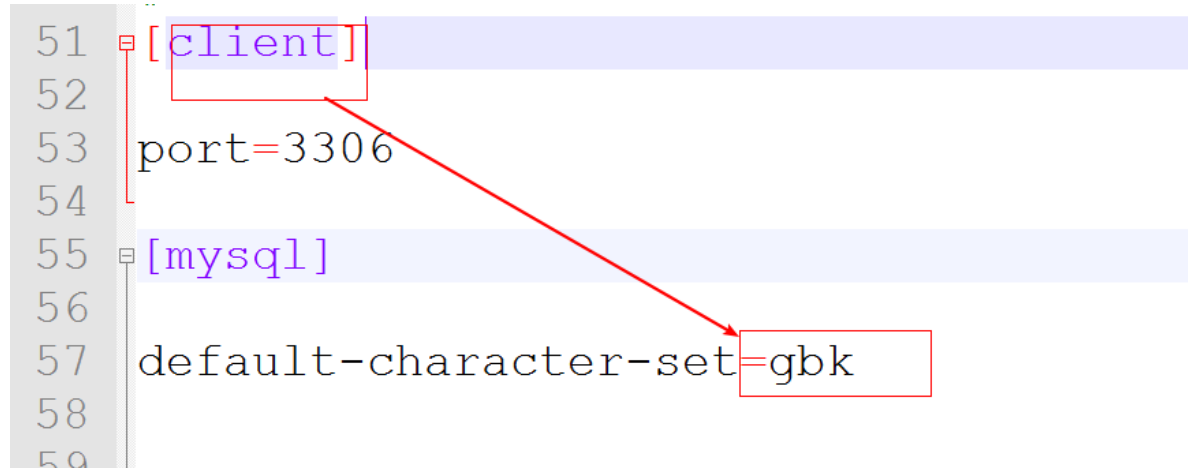
```
insert into product values(null, '苹果电脑', 18000.0, 10);  
insert into product values(null, '华为5G手机', 30000, 20);  
insert into product values(null, '小米手机', 1800, 30);  
insert into product values(null, 'iPhonex', 8000, 10);  
insert into product values(null, '苹果电脑', 8000, 100);  
insert into product values(null, 'iPhone7', 6000, 200);  
insert into product values(null, 'iPhone6s', 4000, 1000);  
insert into product values(null, 'iPhone6', 3500, 100);  
insert into product values(null, 'iPhone5s', 3000, 100);
```

```
insert into product values(null, '方便面', 4.5, 1000);  
insert into product values(null, '咖啡', 11, 200);  
insert into product values(null, '矿泉水', 3, 500);
```


命令行插入中文数据报错:

```
mysql> use day01;  
Database changed  
mysql> insert into student values(2,'张三');  
ERROR 1366 (HY000): Incorrect string value: '\xD5\xC5\xC8\xFD' for column 'name'  
at row 1  
mysql>
```

- 关闭服务, net stop MySql
- 在数据库软件的安装目录下面, 修改配置文件 my.ini中客户端的编码为gbk



```
51 [client]  
52  
53 port=3306  
54  
55 [mysql]  
56  
57 default-character-set=gbk  
58  
59
```

- 重新打开命令行,开启服务, net start MySql

4.小结

1. 语法

- 插入特定的列

```
insert into 表名(列,列) values(值,值)
```

- 插入所有的列

```
insert into 表名 values(值,值,值....)
```

2. 注意

- 插入特定的列:没有赋值的列,系统自动赋为null(前提是当前列没有设置not null 约束)
- 列名与列值的类型、个数、顺序要一一对应。
- 值不要超出列定义的长度。
- 插入的日期和字符串, 使用引号括起来。

知识点-更新记录

1.目标

我们数据插入成功了, 还可以对已有的数据进行更新。

2.步骤

- 更新数据

3.讲解

3.1语法

```
update 表 set 列 =值, 列 =值 [where 条件]
```

3.2练习

- 将所有商品的价格修改为5000元
- 将商品名是Mac的价格修改为18000元
- 将商品名是Mac的价格修改为17000,数量修改为5
- 将商品名是方便面的商品的价格在原有基础上增加2元

```
update product set price = 5000;  
UPDATE product set price = 18000 WHERE name = 'Mac';  
UPDATE product set price = 17000,num = 5 WHERE name = 'Mac';  
UPDATE product set price = price+2 WHERE name = '方便面';
```

4.小结

1. 语法

```
update 表 set 列 = 值, 列=值, 列=值 [where 条件]
```

2. 注意

- 如果没有加where 更新整个的
- 工作里面一般是加where

知识点-删除记录

1.目标

- 掌握记录的删除

2.步骤

- 使用delete删除
- 使用truncate删除

3.讲解

3.1delete

- 语法

```
delete from 表 [where 条件]      注意：删除数据用delete,不用truncate
```

- 练习
 - 删除表中名称为'Mac'的记录
 - 删除价格小于5001的商品记录
 - 删除表中的所有记录

```
delete from product where pname = 'Mac';
delete from product where price < 5001;
delete from product;
```

3.2truncate

```
truncate table 表;
```

4.小结

1. 删除记录

```
delete from 表 【where 条件】
truncate table 表;
```

2. delete 和truncate区别【面试题】

- DELETE 删除表中的数据, 表结构还在; 删除的记录可以找回
- TRUNCATE 删除是把表直接DROP掉, 然后再创建一个同样的新表(空)。删除的记录不可以找回

3. 工作里面的删除

- 物理删除: 真正的删除了, 数据不在, 使用delete就属于物理删除
- 逻辑删除: 没有真正的删除, 数据还在. 搞一个标记, 其实逻辑删除是更新 eg: state字段 1 启用 0禁用

工作里面一般使用逻辑删除用的多

第七章-DQL操作表记录-查询【重点】

知识点-单表查询

1.目标

我们上面讲解了对数据的增删改, 下面就来重点讲解数据的简单查询.

2.路径

- 基本查询语法
- 查询所有的列
- 查询某张表特定列
- 去重查询
- 别名查询
- 运算查询(+,-,*,/等)

- 基本条件查询

3.讲解

3.1基本查询语法

```
select [*][列名 ,列名][列名 as 别名 ...] [distinct 字段] from 表名 [where 条件]
```

3.2简单查询

3.2.1 查询所有的列的记录

- 语法

```
select * form 表
```

- 查询商品表里面的所有的列

```
select * from product;
```

3.2.2查询某张表特定列的记录

- 语法

```
select 列名,列名,列名... from 表
```

- 查询商品名字和价格

```
select pname, price from product;
```

3.2.3 去重查询

- 语法

```
SELECT DISTINCT 字段名 FROM 表名; //要数据一模一样才能去重
```

- 去重查询商品的价格

```
select distinct price from product;
```

注意: 去重针对某列, distinct前面不能先出现列名

3.2.4 别名查询

- 语法

```
select 列名 as 别名 ,列名 from 表 //列别名 as可以不写  
select 别名.* from 表 as 别名 //表别名(多表查询, 明天会具体讲)
```

- 查询商品名称和商品价格, 商品价格通过别名'价格'来显示

```
select pname , price as 价格 from product;
```

3.2.5运算查询(+,-,*,/等)

- 把商品名, 和商品价格+10查询出来

```
select pname ,price+10 from product;
```

注意

- 运算查询字段,字段之间是可以的
- 字符串等类型可以做运算查询, 但结果没有意义

3.3条件查询

3.3.1语法

```
select ... from 表 where 条件
```

//取出表中的每条数据, 满足条件的记录就返回, 不满足条件的记录不返回

比较运算符	> < <= >= = <>	大于、小于、大于(小于)等于、不等于
	between .. and ..	显示在某一区间的值
	in(set)	显示在in列表中的值, 例: in(100,200)
	like '张pattern'	模糊查询
	Is null	判断是否为空
逻辑运算符	and	多个条件同时成立
	or	多个条件任一成立
	not	不成立, 例: where not(salary>100);

1. between...and... 区间查询

eg: where price between 1000 and 3000 相当于 1000<=price<=3000

2. in(值,值..)

```
-- 查询id为1,3,5,7的
select * from t_product where id = 1
select * from t_product where id = 3
select * from t_product where id = 5
select * from t_product where id = 7

select * from t_product where id in(1,3,5,7)
```

3. like 模糊查询 一般和_或者%一起使用

- _ 占一位
- % 占0或者n位

```
name like '张%' --查询姓张的用户, 名字的数字没有限制
name like '张_' --查询姓张的用户 并且名字是两个的字的
```

4. and 多条件同时满足

```
where 条件1 and 条件2 and 条件3
```

5. or 任意条件满足

```
where 条件1 or 条件2 or 条件3
```

3.3.2练习

- 查询商品价格>3000的商品
- 查询id=1的商品
- 查询id<>1的商品
- 查询价格在3000到6000之间的商品
- 查询id在1, 5, 7, 15范围内的商品
- 查询商品名以iPho开头的商品(iPhone系列)
- 查询商品价格大于3000并且数量大于20的商品 (条件 and 条件 and...)
- 查询id=1或者价格小于3000的商品

```
select * from product where price > 3000;
select * from product where pid = 1;
select * from product where pid <> 1;
select * from product where price between 3000 and 6000;
select * from product where id in (1,5,7,15);
select * from product where pname like 'iPho%';
select * from product where price > 3000 and num > 20;
select * from product where pid = 1 or price < 3000;
```

4.小结

- 语法

```
select [*, [列名, 列名], [列名 as 别名], [distinct 列名], [列(+,-..)] from 表名 [where 条件]
```

- 条件
 - where price > 2000
 - where id = 1
 - where id <> 1
 - where name like '张%'
 - where price between 3000 and 5000
 - where id in(1,3,5,7,10)
 - where name like '张%' and age > 10
 - where name like '张%' or age <60
 - 逻辑运算符

知识点-排序查询

1.目标

- 能够使用SQL语句进行排序

2.分析

有时候我们需要对查询出来的结果排序显示，那么就可以通过 `ORDER BY` 子句将查询出的结果进行排序。排序可以根据一个字段排，也可以根据多个字段排序，排序只是对查询的结果集排序，并不会影响表中数据的顺序。

3.讲解

3.0环境的准备

```
# 创建学生表(有sid,学生姓名,学生性别,学生年龄,分数列,其中sid为主键自动增长)
CREATE TABLE student(
    sid INT PRIMARY KEY auto_increment,
    sname VARCHAR(40),
    sex VARCHAR(10),
    age INT,
    score DOUBLE
);

INSERT INTO student VALUES(null,'zs','男',18,98.5);
INSERT INTO student VALUES(null,'ls','女',18,96.5);
INSERT INTO student VALUES(null,'ww','男',15,50.5);
INSERT INTO student VALUES(null,'zl','女',20,98.5);
INSERT INTO student VALUES(null,'tq','男',18,60.5);
INSERT INTO student VALUES(null,'wb','男',38,98.5);
INSERT INTO student VALUES(null,'小丽','男',18,100);
INSERT INTO student VALUES(null,'小红','女',28,28);
INSERT INTO student VALUES(null,'小强','男',21,95);
```

3.1单列排序

1. 语法: 只按某一个字段进行排序，单列排序

```
SELECT 字段名 FROM 表名 [WHERE 条件] ORDER BY 字段名 [ASC|DESC]; //ASC: 升序，默认值；DESC: 降序
```

2. 练习: 以分数降序查询所有的学生

```
SELECT * FROM student ORDER BY score DESC
```

3.2组合排序

1. 语法: 同时对多个字段进行排序，如果第1个字段相等，则按第2个字段排序，依次类推

```
SELECT 字段名 FROM 表名 WHERE 字段=值 ORDER BY 字段名1 [ASC|DESC], 字段名2 [ASC|DESC];
```

2. 练习: 以分数降序查询所有的学生, 如果分数一致,再以age降序

```
SELECT * FROM student ORDER BY score DESC, age DESC
```

4.小结

1. 排序的语法

```
order by 列 asc/desc, 列 asc/desc;  
asc: 升序【默认值】  
desc: 降序
```

2. 应用场景

商城里面 根据价格, 销量, 上架时间, 评论数...

社交里面 根据距离排序

知识点-聚合函数

1.目标

- 能够使用聚合函数

2.分析

之前我们做的查询都是横向查询，它们都是根据条件一行一行的进行判断，而使用聚合函数查询是==纵向查询==，它是对一系列的值进行计算，然后返回==一个结果值==。**聚合函数会忽略空值NULL**

3.讲解

聚合函数	作用
max(列名)	求这一列的最大值
min(列名)	求这一列的最小值
avg(列名)	求这一列的平均值
count(列名)	统计这一列有多少条记录
sum(列名)	对这一列求总和

1. 语法

```
SELECT 聚合函数(列名) FROM 表名 [where 条件];
```

2. 练习


```
-- 求出学生表里面的最高分数
-- 求出学生表里面的最低分数
-- 求出学生表里面的分数的总和(忽略null值)
-- 求出学生表里面的平均分
-- 统计学生的总人数 (忽略null)
SELECT MAX(score) FROM student
SELECT MIN(score) FROM student
SELECT SUM(score) FROM student
SELECT AVG(score) FROM student
SELECT COUNT(sid) FROM student
SELECT COUNT(*) FROM student
```

注意: 聚合函数会忽略空值NULL

我们发现对于NULL的记录不会统计, 建议如果统计个数则不要使用有可能为null的列, 但如果需要把NULL也统计进去呢? 我们可以通过 IFNULL(列名, 默认值) 函数来解决这个问题. 如果列不为空, 返回这列的值。如果为NULL, 则返回默认值。

4.小结

1. 语法

```
select 聚合函数(列) from 表名;
```

2. 聚合函数

- max() 最大值
- min() 最小值
- sum() 求和
- avg() 平均值
- count() 统计数量

3. 注意事项

- 聚合函数会忽略null值的,
- 可以通过 IFNULL(列名, 默认值) 函数来解决这个问题. 如果列不为空, 返回这列的值。如果为NULL, 则返回默认值。

知识点-分组查询

1.目标

- 能够使用SQL语句进行分组查询

2.分析

分组查询是指使用 GROUP BY语句对查询信息进行分组

GROUP BY怎么分组的? 将分组字段结果中相同内容作为一组, 如按性别将学生分成两组

==GROUP BY将分组字段结果中相同内容作为一组, 并且返回每组的第一条数据, 所以单独分组没什么用处。分组的目的是为了统计, 一般分组会跟聚合函数一起使用==

3.讲解

3.1分组

1. 语法

```
SELECT 字段1,字段2... FROM 表名 [where 条件] GROUP BY 列 [HAVING 条件];
```

2. 练习:根据性别分组, 统计每一组学生的总人数

```
-- 根据性别分组, 统计每一组学生的总人数  
SELECT sex, count(*) FROM student GROUP BY sex
```

3.2 分组后筛选 having

- 练习根据性别分组, 统计每一组学生的总人数> 5的(分组后筛选)

```
SELECT sex, count(*) FROM student GROUP BY sex HAVING count(*) > 5
```

4.小结

1. 分组语法

```
group by 列 [having 条件]
```

2. 注意事项

单独分组 没有意义, 返回每一组的第一条记录

分组的目的一般为了做统计使用, 所以经常和聚合函数一起使用

分组查询如果不查询出分组字段的值,就无法得知结果属于那组

在分组里面, 如果select后面的列没有出现在group by后面,展示这个组的这个列的第一个数据

3. where和having的区别【面试】

子名	作用
where 子句	1) 对查询结果进行分组前, 将不符合where条件的行去掉, 即在分组之前过滤数据, 即 先过滤再分组 。2) where后面不可以使用聚合函数
having 字句	1) having 子句的作用是筛选满足条件的组, 即在分组之后过滤数据, 即 先分组再过滤 。2) having后面可以使用聚合函数

知识点-分页查询

1.目标

- 掌握分页查询

2.分析

LIMIT是限制的意思, 所以LIMIT的作用就是限制查询记录的条数. 经常用来做分页查询

3.讲解

1. 语法

`select ... from limit` 起始行数, 查询的记录条数.

LIMIT a,b;

a起始行数, 从0开始计数, 如果省略, 默认就是0; a=(当前页码-1)*b;

b: 返回的行数

2. 练习

eg: 分页查询学生, 每一页查询4条
b=4; a=(当前页码-1)*b;

第一页: a=0, b=4;

第二页: a=4, b=4;

第三页: a=8, b=4;

4.小结

1. 语法

limit a,b;

a: 从哪里开始查询, 从0开始计数 【a=(当前页码-1)*b】

b: 一页查询的数量【固定的, 自定义的】

2. 应用场景

如果数据库里面的数据量特别大, 我们不建议一次查询出来. 为了提升性能和用户体验, 使用分页

查询的语法小结

`select...from...[where...][group by...having...][order by...][limit...]`

`select...from...`

`select...from...where...`

`select...from...order by...`

`select...from...group by...having...`

`select...from...limit...`

总结

必须练习:

对记录的增删查改-----必须练习

- 能够理解数据库的概念

其实就是存储数据的仓库, 可以持久化保存数据, 并可以通过sql语句快速对数据进行增删查改

- 能够安装MySQL数据库

查看资料中的文档

- 能够使用SQL语句操作数据库

对数据库的增

```
create database 数据库名;
```

对数据库的删

```
drop database 数据库名;
```

对数据库的查

```
show databases;
```

```
show create database 数据库名;
```

对数据库的改

```
alter database 数据库名 character set 字符集;
```

- 能够使用SQL语句操作表结构

对表的增

```
create table 表名(  
    字段名 字段类型 [字段约束],  
    字段名 字段类型 [字段约束],  
    ...  
    字段名 字段类型 [字段约束]  
);  
字段类型:
```

数值类型: `int`, `bigint`, `float`, `double`

字符串: `varchar(size)`, `char(size)`

时间日期: `date`, `datetime`

字段约束:

`primary key` 主键约束(唯一非空)

`auto_increment` 自增(`int`)

`unique` 唯一

`not null` 非空

对表的删

```
drop table 表名;
```

对表的查

```
show tables;
```

```
desc 表名;
```

对表的改

```
alter table 表名 add 字段名 字段类型 字段约束;
```

```
alter table 表名 modify 字段名 字段类型 字段约束;
```

```
alter table 表名 change 旧字段名 新字段名 字段类型 字段约束;
```

```
alter table 表名 drop 字段名 ;
```

```
rename table 旧表名 to 新表名;
```

- 能够使用SQL语句进行数据的添加修改和删除的操作

指定列添加:

```
insert into 表名(字段名,字段名,...) values(值,值,...);
```

添加所有列:

```
insert into 表名 values(值,值,...);
```

修改列:

```
update 表名 set 列名 = 值,列名=值 [where 条件];
```

删除:

```
delete from 表名 [where 条件];
```

```
truncate table 表名;
```

- 能够使用SQL语句进行条件查询数据

```
select...from 表 [where 条件][group by 列名][having 条件][order by 列名 asc\desc]  
[limit a,b]  []:可选
```

- 能够使用SQL语句进行排序

参考上面

- 能够使用聚合函数

avg(列名)

sum(列名)

max(列名)

min(列名)

count(列名)

- 能够使用SQL语句进行分组查询

参考上面

- 能够使用SQL语句进行分页查询

参考上面