SQL语句分类：

数据定义语言 DDL　创建数据库及表　常用语句：create

数据控制语言 DCL

数据操纵语言 DML　主要进行 插入，删除，增（改及更新）insert delete update

数据查询语言 DQL　主要进行 查询 select

启动MySQL的服务

mysql --skip-grant-tables

登录数据库：

mysql -uroot -p123456　　mysql 登录的用户名和密码

mysql -u root -p　输入密码

退出mysql

quit;

查看数据库中已有的库：

show databases;

进入库：

use 库名　　如：use mydb

创建库：

create database 库名;　如：create database mydb11;

创建规定字符集为UTF8的库：

create database IF NOT EXISTS 库名 DEFAULT CHARSET utf8 COLLATE utf8_general_ci;

删除库：

drop database 库名;　　如：drop database mydb11;

库中的表操作：

数据类型

数值型：

TINYINT	SMALLINT	INT	BIGINT	FLOAT	DOUBLE	SQL数据类型
byte	short	int	long	float	doduble	JAVA数据类型

TINYINT SMALLINT INT BIGINT FLOAT DOUBLE

byte short int long float double

逻辑型：

BIT占1位 类似于 JAVA里的 boolean

日期型：

DATE 用于表示日期

TIME 用于表示时间

DATETIME 手动添加时间到表里面

TIMESTAMP 自动添加时间到表里面

大数据类型：

BLOB类型 图片，视频，音频等文件 TEXT类型 文字内容

字符串：

varchar(长度) 可变长度字符串类型 char(长度) 固定长度

查看表：

show tables;

创建表：

create table 表名(字段名 数据类型，.....);

如：create table user22(id int,name varchar(64),sex varchar(5),age int);

创建表带约束：

非空：not null

主键：primary key 一行数据的唯一识别标识（不可重复且不能为空）

自增长：auto_increment　多用于ID

默认值：default

创建带有约束的表：

create table 表名(字段名 数据类型 约束条件，.....);

如：create table user22(id int primary key auto_increment,name varchar(64) not null,sex varchar(5),age int);

创建一个ID为主键且自动增长的，姓名必须有不能为空的表

查看表结构：

desc 表名;　如：desc user;

删除表：

drop table 表名;　如：drop table userss;

删除有关联关系的表：

注：**MySQL删除表：Cannot delete or update a parent row: a foreign key constraint fails。**

SET foreign_key_checks = 0;　// 先设置外键约束检查关闭

drop table 表名; // 删除表，如果要删除视图，也是如此

SET foreign_key_checks = 1; // 开启外键约束检查，以保持表结构完整性

MySQL的环境变量中存在一个foreign_key_checks，这是默认检查外键的配置项，如果将其设置为0，则表示不检查外键约束。查看foreign_key_checks的值： show VARIABLES like "foreign%";

注意：在删除完表之后，最好是**重新打开检查**(SET foreign_key_checks = 1) 以保持表格结构的完整性。

修改表名：

alter table 原表名 rename 新表名;　如：alter table user rename newuser;

修改表的某一列名

alter table 表名 change 原列名 新列名 类型;

如：alter table user change id u_id int;

删除表的某一列

alter table 表名 drop 列名;　如：alter table user drop uid;

添加某一列

alter table 表名 add 列名 数据类型;　　如：alter table user add uuid int;

数据库操作流程：

1、登录数据库　mysql -uroot -p123456　mysql 登录的用户名和密码

2、查看已存在的库

（如有则进入库，没有则创建库　create database 库名;　如：create database mydb11;）

show databases;

3、进入库　use 库名　　如：use mydb

4、查看库中已有的表　shou tables;

（如有则对表进行增删改查操作，如没有则创建表）

5、创建表　create table user(id int,name varchar(18),sex varchar(5),age int);

6、查看表结构 desc 表名;　desc user;

对表中的数据进行增删改查：

1、插入数据(增数据)

insert into 表名 values(对应数据结构的数据，.....);

如：insert into user values(125425,"xm","man",25);

只插入部分数据（根据需要插入数据）

insert into 表名(字段名，.....) values(对应数据结构的数据，.....);

如：insert into user(id,name,sex) values(125425,"xm","man");
注意：如果插入数据时某一字段没有规定值就自动存为NULL

值 为 NULL 为条件的查改删方法：
select * from 表名 where 字段 is null;
查询user表中age为null的所有数据
select * from user where age is null;

2、查询数据

查询所有数据
select * from 表名；从某个表中查询所有的数据
如： select * from user;

根据表中的字段查询数据
select 字段 from 表名； 从某个表中根据字段查询数据
如： select id,name from user;

根据条件查询数据
select * from 表名 where 字段=值； 从某个表中根据字段查询数据
如： select * from user where name="xiaoming";

3、改表中的数据(where为修改的条件)

update 表名 set 字段=新的值（要改成的新数据） where 字段=原来的字段数据;
如：update user set name="xiaoyu" where name="xm";
把姓名是 xm 的名字改成 xiaoyu

4、删除表中的数据

delete from 表名；删除表中的所有数据
delete from 表名 where 字段=值；根据条件删除对应的数据
如：delete from user where name="xm"; 删除姓名是xm的数据

高级查询：

1、去重查询 加关键字 distinct 去重是不显示表中多条所有字段值完全相同的数据

select distinct * from 表名; 如：select distinct * from user22;

2、条件查询 加关键字 where 根据字段是什么的条件查询
可以用值等于，大于，小于，大于等于，小于等于（=，>，<，>=，<=）
select * from 表名 where 字段=值;
如：select * from user22 where id=2;（或 where age>20; where age<20;）

3、多条件查询 关键字 and（同时）or（或者）
select * from 表名 where 字段=值 and 字段>值;
如：select * from user22 where name="xm" and age>20;
查询name是xm并且age大于20的数据
注意在：java代码中 如果SQL数据类型是字符串（varchar类型）那么JAVA中一定要用 ‘’ 单引号 阔起来
如：String sqls ="select * from user where id=2018001 and name='"+"liurong'";

select * from 表名 where 字段=值 or 字段>值;
如：select * from user22 where id<5 or age>20;
查询id小于5或者age大于20的数据

4、多条件子句 in :在指定范围内选择 但不是范围内所有数据
select * from 表名 where 字段=值 and 字段>值;
如：select * from user22 where age in (18,33);
查询age为18和33的所有数据，但不包含大于18小于33的其它数据

5、多条件子句 between 加 and 表示查询的范围（只能是从小到大）
select * from 表名 where 字段 between 值 and 值;
如：select * from user22 where age between 18 and 33;

查询age为18和33的所有数据

6、模糊查询：字符数据类型中，找出相似的数据 like

name	age
xiao	20
lx	21
ssx	26
wdsxid	28
sx	99
xoo	29

通配符：

_ 代表任意的一个字符

表达方式："_x"　数据中有x这个字符且x前面只有一个字符的所有数据

"_x_"　数据中有x这个字符且x前面后面只有一个字符的所有数据% 代表任意的多个字符

表达方式："%x"　数据中有x这个字符且x前面有多个字符的所有数据

"%x%"　数据中有x这个字符且x前面后面有多个字符的所有数据

select * from 表名 where 字段 like '通配符';

SELECT * FROM ajaxs WHERE username LIKE '%c%';

如：select * from user22 where name like '_x';

用法：'_x%'　　'_x_'　　'%x%'

表示查询name中有x这个字符且x前面只有一个字符的所有数据

获取的结果：

name	age
lx	21
sx	99

7、查询排序 按条件升序、降序显示 order by

select * from 表名 order by 字段; 默认升序

select * from 表名 order by 字段 asc; 指定顺序

select * from 表名 order by 字段 desc; （倒序—从最后向最前面）降序

查询MYSQl数据表中的最后一条记录（表里有ID且ID有顺序的情况）

方法一： SELECT * FROM 表名 ORDER BY 自增id字段 DESC LIMIT [要取的数据（如果是 1 就是最后一条，要是3就是最后3条）]；

如：SELECT * FROM product ORDER BY id DESC LIMIT 1 ;

方法二：SELECT * FROM 表名 WHERE id=(SELECT MAX(id字段) FROM 表名);

SELECT * FROM my2 WHERE id=(SELECT MAX(id) FROM my2);

方法三：SELECT * FROM 表名 WHERE id=(SELECT LAST_INSERT_ID());

SELECT * FROM product WHERE id=(SELECT LAST_INSERT_ID());

表里没有ID且无顺序时：

//1、先查到表里的总条数

SELECT COUNT(*) FROM 表名;

//2、先查到表里的总条数减去1的值

SELECT * FROM 表名 LIMIT 总条数减去1的值,1;

SELECT * FROM product LIMIT 5,1;

查第一条记录： SELECT * FROM 表名 LIMIT 1;

8、聚集函数：对表中某字段的数据进行聚集操作。如： 求和 平均值等

select 字段，.... 聚集函数(字段名) as 函数别名 from 表名;

字段可有可无，如找最大最小可以有用来显示最大的是什么，如最大年龄 字段就可以有name《as 函数别名》可以不要

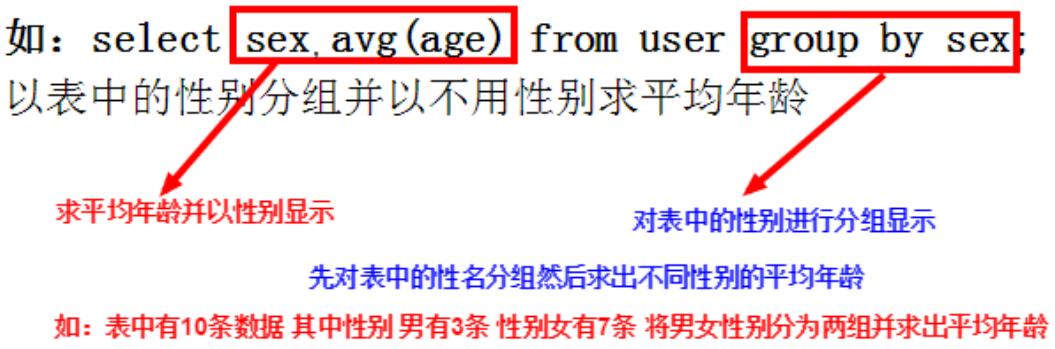
如：select sum(age) as sum_age from user; 求age字段总合，并显示结果名是sum_age

函数：sum 求和 avg 求平均值 max 求最大值 min 求最小值 count 统计函数

9、分组查询：通常结合聚集函数一起使用 group by

select 聚集函数 from 表名 group by bp 字段;

如：select sex,avg(age) from user group by sex;
以表中的性别分组并以不用性别求平均年龄



10、分页操作

11、SQL注入：当用户登录时需要输入用户名和密码，SQL语句是接收了用户输入的用户名和密码，使用了and多条件查询，如：

String name = ″liurong″;

String spwd = ″123456″;

String sqls ="select * from user where name='"+name+"' and "+"pwd="+"'"+spwd+"'";

这个语句的意思是　查询表里 用户输入的 用户名和密码 都同时存在数据库中的一条数据

注入是这样的：用户输入用户名通过组拼SQL语句传入SQL关键字参数， String name = ″liurong or '1'='1'″;

如此形成的新SQL就是：

select * from user where name=liurong or 1=1 and pwd=123456;

SQL语句执行这条新SQL时　是先判断 and 再判断 or　and时必须左右同时符合条件才成立，但是 or 是只要一边为符合 就可以 条件就成立。

所以此时 实际就是 一个单条件的查询语句，绕开了密码的比对。