sql语句

1.操作数据库

(1)创建数据库

create database [if not exists] db_name [character set xxx] [collate xxx]

创建一个名为mydb1的数据库

```
create database mydb1;
```

创建一个使用gbk字符集的mydb2数据库

```
create database mydb2 character set gbk;
```

创建一个使用utf-8字符集，并待校对规则的mydb3数据库

```
create database mydb3 character set utf8 collate utf8_bin;
```

(2)查看数据库

显示数据库语句：查看当前数据库服务器中的所有数据库

```
show databases;
```

显示数据库创建语句：查看前面创建的数据库的定义信息

```
show create database db_name;
```

(3)修改数据库

alter database [IF NOT EXISTS] db_name [alter_specification [, alter_specification] ...]

查看服务器中的数据库，并把其中某一个库的字符集修改为utf8

```
alter database mydb2 character set utf8;
```

(4)删除数据库

drop database [IF NOT EXISTSI] db_name

删除前面创建的mydb1数据库

```
drop database mydb1;
```

(4)选择数据库

进去数据库：use db_name;

查看当前所选的数据库：select database();

2.操作表

(1).创建表

```
create table table_name
(
    field1 datatype,
    field2 datatype,
    field3 datatype
)[character set 字符集] [collate 校对规则]
```

field：指定列名 datatype：指定列类型

字符集和校对规则是可选项，不写的话默认继承当前数据库的字符集和校对规则

**MySQL常用数据类型：

字符串型：VARCHAR(可变长度)、CHAR(固定长度)

name varchar(40)---最大长度不超过40字节

gender char(1)---固定长度为1

大数据类型：BLOB(大二进制)、TEXT(大文本)

数值类型：TINYINT(1字节，相当于java中的byte)、SMALLINT(2字节，相 当于Java中的short)、INT(4字节，Java中int)、BIGINT(8字节，Java中的long)

FLOAT、DOUBLE

逻辑性：BIT

日期型：DATE、TIME、DATETIME、TIMESTAMP

约束：

primary key(auto_increment自动增长)---主键约束，不允许为空，不允许重复，唯一标识一个表

unique---唯一约束，不能重复

not null---非空约束、

外键约束

(2).查看表

show tables：查看当前数据库中的所有表

desc table_name：查看表结构

show create table table_name:查看建表语句

(3).修改表

alter table table_name add/modify/drop (column datatype[default expr] [,column datatype]...)

在上面员工表的基础上增加一个image列：

```
alter table employee add image  blob;
```

修改job列，使其长度为60

```
alter table  employee modify job varchar(60);
```

删除gender列：

```
alter table employee drop  gender;
```

修改表名为user

```
rename table employee to user;
```

修改表的字符集编码为gbk：

```
alter table  user character set gbk;
```

修改列名name为username：

```
alter table user change name username varchar(20)
```

(3).删除表

```
drop table table_name;
```

3.操作表记录(CRUD)

(1)INSERT

INSERT INTO table_name [(colume [,colume...])] VALUES (value [, value...]);

使用insert语句向表中插入三个员工信息

insert into user (id,username,gender,birthday,entry_date,job,salary,resume)
values (null,'张飞','1','1999-09-09','1999-10-01', '打手' , 998.0, '能打');
***乱码问题：show variables like 'character%';查看服务器端的各种编码字符集设置。
服务器默认连接的客户端的字符集，当前连接的字符集，最后要返回的字符集均为utf-8
set names gbk; 命令修改三个字符集
***
当插入一条包含所有列的记录时，列信息可以省略：

insert into user values (null,'关羽',1,'2000-01-01','2001-02-02','打手',1998.0,'特别能打')、

insert into user values (...),(...)插入多条语句

(2)UPDATEA

UPDATE table_name SET colume_name1=expr1 [,colume_name2=expr2...][WHERE where_defination]

将所有员工薪水修改为5000：

update user set salary=5000;

将姓名为张飞的员工薪水修改为3000,工作修改为保镖：

update user set salary=3000, job='保镖' where username='张飞';

将姓名为诸葛亮的员工薪水在原来的基础上增加1000：

update user set salary=salary+1000;

(3)DELETE

DELETE FROM table_name [Where where_definaition];

删除表中名称为张飞的记录：

delete from user where username='张飞';

删除表中所有记录：

delete from user; 删除表中所有元素，但是保留表

使用truncate删除表中记录：

TRUNCATE TABLE user; 删除表，并创建一个具有相同表结构的新表

TRUNCATE的效率比DELETE高，因为在删除时delete是逐条删除，truncate则是直接摧毁整张表，然后创建空表。

(4)SELECT

a.基本查询

SELECT[DISTINCT] *|{colume1,colume2,colume3...} FROM table_name;

```
create table exam(
 id int primary key auto_increment,
 name varchar(20) not null,
 chinese double,
 math double,
 english double
);

insert into exam values(null,'关羽',85,76,70);
insert into exam values(null,'张飞',70,75,70);
insert into exam values(null,'赵云',90,65,95);
```

查询表中所有学生信息：

select * from exam;

查询表中所有学生的姓名和英语成绩：

select name,english from exam;

过滤表中重复元素：

```
select distinct english from exam;
```

在所有学生分数上加10分特长分显示：

```
select name, math+10,english+10,chinese+10 from exam;
```

统计每个学生的总分：

```
select name ,math+english+chinese from exam;
```

使用别名显示：

```
select name as 姓名，math+english+chinese as 总成绩 from exam
```

或者简写select name 姓名，math+english+chinese 总成绩 from exam

b.使用where子句进行过滤查询

查询姓名为张飞的学生成绩

```
select * from exam where name='张飞';
```

查询英语成绩大于90分的同学：

```
select name from exam where english>90;
```

查询总成绩大于200分的所有同学：

```
select name 姓名,math+english+chinese as 总成绩 from exam where
math+english+chinese>200;
```

查询英语成绩在80-100之间的同学：

```
select * from exam where english between 80 and 100;
```

查询数学成绩为75，76,77的同学：

```
select * from exam where math in(75,76,77);
```

查询所有姓张的学生成绩：

```
select * from exam where name like '张%';
```

%代表一个或多个字符，_表示一个字符

查询数学分数>70，语文分数>80的同学：

```
select * from exam where math>70 and chinese>80;
```

c.使用order by关键字对查询结果进行排序

SELECT colume1,colume2,colume3...FROM table_name order by colume asc|desc

asc---升序 desc---降序

对语文成绩排序后输出：

```
select name chinese from exam order by chinese desc;
```

对总成绩排序后输出：

```
select name 姓名,math+english+chinese 总成绩 from exam order by 总成绩;
```

对姓张的学生成绩排序输出：

```
elect name 姓名,math+english+chinese 总成绩 from exam where name like
'张%' order by 总成绩;
```

d.使用聚合函数

(1).COUNT---用来统计符合条件的行的个数

统计一个班级共有多少学生：

```
select count(*) from exam;
```

统计数学成绩大于90的学生有多少：

```
select count(*) from exam where math>90;
```

统计总分大于250的学生人数：

```
select count(*) from exam where math+chinese+english>250;
```

(2).SUM---用来将符合条件的记录的指定列进行求和操作

统计一个班级数学总成绩：

```
select sum(math) from exam;
```

统计一个班级各科总成绩：

select sum(math),sum(english),sum(chinese) from exam;

统计一个班级各科总成绩之和：

select sum(math+english+chinese) from exam;

在执行计算时，只要有null参与计算，整个计算结果都是null，此时可以使用ifnull(math,0)

select sum(ifnull(chinese,0)+ifnull(english,0)+ifnull(math,0)) from exam;

统计一个班级语文成绩的平均分：

select sum(chinese)/count(*) 语文平均分 from exam;

(3).AVG---用来计算符合条件的记录的指定列的平均值

求一个班级数学平均分：

select avg(math) from exam;

求一个班级的总平均分：

select avg(ifnull(chinese,0)+ifnull(english,0)+ifnull(math,0)) from exam

(4)MAX/MIN---用来获取符合条件的所有记录指定列的最大值和最小值

求班级的最高分和最低分：

select max(ifnull(chinese,0)+ifnull(english,0)+ifnull(math,0)) from exam

select min(ifnull(chinese,0)+ifnull(english,0)+ifnull(math,0)) from exam;

e.分组查询

GROUP BY子句

create table orders(id int,product varchar(20),price float);

insert into orders values(1,'电视',900),(2,'洗衣机',100),(3,'洗衣粉',90),(4,'桔子', 9), (5,'洗衣粉',90)

对订单表中的商品归类后，显示每一类商品的总价：

select product,sum(price) from orders group by product

查询购买了几类商品，并且每一类总价大于100的商品：

select product,sum(price) from orders group by product having sum(price)>100

***where子句和having子句的区别：

where在分组之前过滤，having在分组之后过滤

having子句中可以使用聚合函数，where不可以

很多情况下使用where的地方可以用having代替

***

查询单价小于100而总价大于150的商品

select product from orders where price<100 group by product having sum(price)>150;

sql语句的书写顺序：select from where group by having order by

sql语句的执行顺序：from where select group by having order by

4.备份和恢复数据库

备份:在cmd窗口下 mysqldump -u root -p dbName>c:/1.sql

恢复:

方式1:在cmd窗口下 mysql -u root -p dbName<c:/1.sql

方式2:在mysql命令下, source c:/1.sql

要注意恢复数据只能恢复数据本身,数据库没法恢复,需要先自己创建出数据后才能进行恢复