**1.对mysql 进行操作：**

进入：mysql -h host\_name -u user username -p passwd

例：mysql -u root -p密码

host\_name:对方服务器主机名或ip地址

显示所有的数据库：show databases; (sql语句)

选定要操作的数据库：use dbname;

显示指定数据库中所有的数据表：show tables;

退出：mysql:\q

结束当前语句的输入：\c

**2. 对库进行操作：**

新建数据库：create database 数据库名；

例：create database students;

删除数据库：drop database 数据库名;

例：drop database students; 注：一次只能删一个。

。。。建一个库students并进入该库.........

**3.对表进行操作：**

新建数据表：

create table 表名(

列名1 列类型 [列的完整性约束],

列名2 列类型 [列的完整性约束],

... ...)default charset=utf8;

例：

create table student(

id int,

name varchar(20),

sex varchar(10))default charset=utf8;

显示表的创建结构：desc 表名 例：desc student；

删除数据表：drop table 表名： 例：drop table student;

(删除当前数据库内)， drop table 库名.表名(删除其他数据库内的表)。

**插：**向数据表中插入记录（行数据）

a.一次插入一行数据

insert [into] 表名([列名]) values(值列表);

注：如果表名后面没有写列名，则默认向所有的列添加值，另外字符串的值要用" "或' '括起来

赋值的数据类型必须要与表中相匹配。

例：insert into student(id,sex) values(1,"男");

insert into student values(7,"徐桂双",'男');

查看整张表的所有信息：select \* from 表名；

b.一次插入多行数据

insert [into] 表名([列名]) values(值列表),(值列表),...

例：insert into student values(3,'dd','女'),(4,'vv','男');

c.从其它表中引入要插入的数据

insert [into] 表名([列名]) select <列名> form 源表名[where ...];

create table student1(id int,name varchar(20),sex varchar(10),age int)default charset=utf8;

insert into student1 values(1,"张无忌",'男',26),(2,'段誉','男',27);

insert into student select id,name,sex from student1 where id=2;

select \* from student;

**改**：更改数据表中的记录

update 表名 set <列名=更新值> [where <更新条件>];

注：where子句是用来限定更新条件的，不带的话则会更新整列。

例：

update student1 set sex='女';

update student1 set sex='男' where id=1;

update student1 set sex='男',age=22 where id=2;

**删除**：删除数据表中的记录：

delete from 表名[where <删除条件>]

例：delete from student1 where age<26;

注：此语句为删除表中的行数据，如果不带where子句则删除整个表中的记录，但 表不会被删除。

**查**：查询表中的信息：

select <列名> from 表名 [where <查询条件表达式>] [order by <排序的列名>] [asc或者desc]

注：asc升序，desc降序，不写就是默认，默认就是升序。

例：

select \* from student;

select id,name from student;

select \* from student where id<=3 order by id;

select \* from student where id<=7 order by id desc;

更改表的结构(列)

**1.为表添加一列，如果没有指定添加的位置，则默认在列尾添加**

alter table 表名 add 列名 建表语句[after 列名]；

建表语句=列类型+列的完整性约束

例：

alter table student add age int;

alter table student add xxx int after id;

**2.删除一列：**

alter table 表名 drop 列名；例：alter table student drop xxx

**3.更改指定列的默认值**

alter table 表名 alter 列名 set default 默认值;

例：alter table student alter age set default 10;

**4.更改列名和它的类型。**

alter table 表名 change 旧列名 新列名 <建表语句> [after 列名]

注：当旧列名和新列名相同时，可以改变该列的类型，不同，在改变旧列名的同时改变它的类型

例：alter table student change id num int not null;

**5.为表增加一个主键**

主键：即把某一列设置为主键，主键列里的每一个成员互不相同，它存在的理由是能够唯一的标识一行。

alter table 表名 add primary key(列名);

注：一个表里只有一个主键列，通过会选标识符/顺序值为主键，选为主键的列必须是非空(not null).

例：alter table student add primary key(num);

把顺序值列设置为主键列的，默认升序排列

已经存在的值能不能被插入，特别是默认值0.

**6.把主键列设置为自增长列**

例：alter table student change num num int auto\_increment;

注：有0不行 设置为自增长的列要是整型，非空，且是主键列。在此例中int不可以省略不写

**7.从当前位置重新设置自增长的初值，否则删掉一个记录之后再添加一个记录会出现隔断的情况。**

alter table 表名 auto\_increment=初值；

**8.删除主键：**

alter table 表名 drop primary key;

注：删除主键之前要删除自增长才能成功。

例：alter table student change num num int ;//删除自增长

例：alter table student drop primary key; //删除主键

**9.更改表名：**

alter table 表名 rename as 新表名;

例:alter table student rename as student2;

例（**主要**）：

创建一个数据表：

create table student3(

code int not null auto\_increment,

name varchar(20) not null,

age int,

sex varchar(10),

score int,

grade int,

address varchar(20) default '未知',

major varchar(20),

primary key(code))default charset=utf8;

向表中插入数据：

insert into student3(name,age,sex,score,grade,address,major)

values('xxx',13,'m',78,1,'asdfgh','sss'),

('xcc',15,'w',88,2,'afdfdh','nnn'),

('cxx',9,'m',99,3,'sdsvfbf','vvv'),

('ccc',45,'m',56,1,'cdfsff','sss'),

('xbb',34,'w',66,1,'cdfgdv','sss');

查看全体学生的详细信息：

select \* from student3;

查看全体学生的学号和姓名：

select code,name from student3;

查看全体学生的学号和姓名，用中文显示列名

select code as ‘学号’,name as ‘姓名’ from student3;

//只是在此次查询时以中文显示，在表中没有改变。

给表设置别名：

select s.code,s.name from student3 as s;

查询所有年级号(去掉重复的年级号)

select distinct grade from student3;

查询年龄在20及以下的学生的姓名;

select name from student3 where age<=20;

查询全体学生的所有信息并按年龄的降序排列：

select \* from student3 order by age desc;//升序的话，删除desc

查询年龄最大的前3个学生的所有信息：

select \* from student3 order by age desc limit 3;

查询年龄最大的第4,5个学生的所有信息

select \* from student3 order by age desc limit 3,2;

计算学生的总数：

select count(\*) from student3; //count(列名)：该列的元素的总个数

计算1班的平均成绩：

select avg(score) from student3 where grade=1;//avg(列名):对某一列的值计算平均值

查找sss专业的最高分和最低分:

select max(score) as ‘最高分’,min(score) as '最低分' from student3 where major='sss';

查询每个专业的平均成绩：

select major,avg(score) as '平均成绩' from student3 group by major;

查询专业的平均成绩在70以上的专业；

select major,avg(score) as '平均成绩' from student3 group by major having avg(score) >70;

求总分：

select sum(score) from student3;//sum(列名)：对某一整型列的值求和

查询年龄[不]在20~50之间的学生的个人信息：

select \* from student3 where age [not] between 20 and 50;

查询'nnn','vvv'专业的学生的所有信息；

select \* from student3 where major in('nnn','vvv');

<=>select \* from student3 where major='nnn' or major='vvv';

查询学号为1的学生的所有信息；

select \* from student3 where code=1;

<=>select \* from student3 where code like 1;

查询名字以x开头的所有学生的所有信息；

select \* from student3 where name like 'x%'；

查询名字以x结尾的所有学生的所有信息；

select \* from student3 where name like '%x';

查询名字里含有x的所有学生的所有信息：

select \* from student3 where name like '%x%';

查询当前数据库里以stu开头的所有的数据表

show tables like 'stu%';

查询性别为'm',并且分数小于70分的学生姓名;

select name from student3 where sex='m' and score<70;

查询性别为m的学生的学号，名字和年龄，查询结果按年龄的降序排列

select name,age,code from student3 where sex='m' order by age desc;

查询全体学生的信息，查询结果按年级的升序排列，同一年级的学生按分数的降序排列：

select \* from student3 order by grade,score desc;

注：select \* from 表名 order by m[desc],n[desc],p[desc],...

查询表的所有信息，以m排序，在m相同的记录里以n排序，在n相同的记录里以p排序...

-----------先建2个数据表-------------------

create table stu(

id int,

name varchar(20),

age int,

sex varchar(6))default charset=utf8;

create table cour(

num int,

course varchar(20),

score int)default charset=utf8;

--------向2个表中插入数据.......

insert into stu values(1,'张无忌',26,'男'),(2,'乔峰',33,'男'),(3,'徐大侠',18,'男'),(5,'风清扬',22,'男');

insert into cour values(1,'语文',78),(1,'数学',88),(2,'语文',90),(3,'数学',99),(3,'英语',88),(4,'美术',90);

多表查询(2表)

1.内连接(inner join):最后的表为2张表连接条件的交集的合集。

1)

select s.id,s.name,s.age,c.course,c.score

from stu as s

inner join cour as c

on s.id=c.num;

2) select stu.id,stu.name,stu.age,cour.couse,cour.score

from stu,cour

where stu.id=cour.num;

...当有3张表时....

1) select A.a,B.b,C.c

from A,B,C

where A.x=B.y=C.z3r;

2.左外连接(left join):以左表为主

select s.id,s.name,s.age,c.course,c.score

from stu as s

left join cour as c

on s.id=c.num;

3.右外连接(right join):以右表为主

select s.id,s.name,s.age,c.course,c.score

from stu as s

right join cour as c

on s.id=c.num;

子查询：将一个查询块嵌套在另一个查询块的where子句或having短语中的查询称为子查询

注：一个select-from-where 语句称为一个查询块

例：select name,age from stu where id in(select num from cour where course='语文');

复制表：

1）复制一张已知的数据表：

create table 新表名 select \* from 已知表;

例：create table x1 select \* from stu;

2）在 复制表时限制复制表中的内容

create table 新表名 select [列名...] from 已知表；

例：create table x2 select name,sex from stu;

3)创建一个已存在的表的空表

create table 新表名 select \* from 已知表 where 0=1;

create table x3 select \* from stu where 0=1;

|  |  |
| --- | --- |
| 函数名 | 描述 |
| 字符串UPPER（列 | 字符串） |  |
| 字符串 LOWER（列 | 字符串） |  |
| 字符串 INITCAP（列 | 字符串） |  |
| 数字 LENGTH（列 | 字符串） |  |
| 字符串 SUBSTR（列|字符串，开始索引，[长度]） |  |
| 字符串 REPLACE（列|字符串，旧内容，新内容） |  |
|  |  |
|  |  |
|  |  |

批量插入：

sql = "INSERT INTO `testTable`(name,sex) VALUES(%s,%s);"

Datas = (shj, man)

self.cur.executemany(sql, datas)

self.coon.commit()

批量更新：

## **前言**

批量插入由于mysql的VALUES原生支持，使用较为便利。

批量更新的写法一般有三种，在更新数量较少的情况下，前两种性能不相上下。但是在更新字段增加，更新条数较多(500以上)建议使用第三种写法。

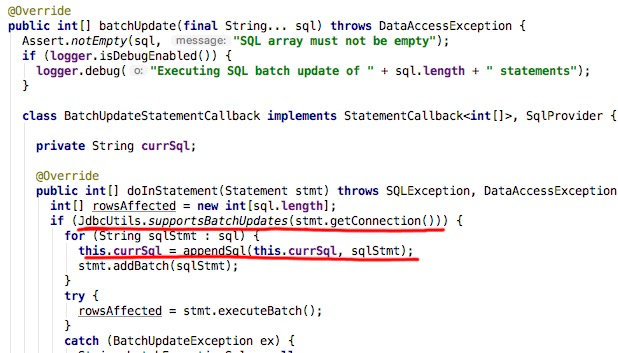
* 常规写法，拼接多个单条更新语句。
* CASE...WHEN... 写法
* JOIN 写法

## **Batch Update**

spring/mybatis/JDBI都支持这种批量更新方式。  
这种更新方式需要设置jdbc连接的参数：

allowMultiQueries=true# 完整url举例

jdbc.url=jdbc:mysql://localhost:3306/db\_name？useUnicode=true&characterEncoding=utf8&allowMultiQueries=true

具体实现以Spring的JdbcTemplate为例。  
batchUpdate的主要代码如下图：  


首先检查了jdbc连接是否支持批量更新操作，如果allowMultiQueries值为false，将被拦截。  
然后拼接了SQL语句，拼接代码如下：

IMG_257

直接用分号拼接。

## **CASE WHEN**

示例:

UPDATE testSET code = (CASE WHEN id = 1 THEN 11WHEN id = 2 THEN 22WHEN id = 3 THEN 33END

) WHERE id IN (1,2,3);

注意：CASE WHEN一定要和WHERE语句一起使用，否则UPDATE会遍历和更新数据库中所有的行。会把未出现在WHEN中的数据都更新成null，如果code列设置为NOT NULL则会报错，否则会置为NULL或者默认值。

## **JOIN**

这种写法不太常见。

UPDATE `test` a JOIN

(SELECT 1 AS id, 11 AS code, 'holy' AS nameUNION SELECT 2 AS id, 22 AS code, 'shit' AS name

) b USING(id, code)SET a.name=b.name;

上述SQL要表达的更新语义是：将id=1且code=11的name更新为'holy'，将id=2且code=22的name更新为'shit'。  
注意，条件字段必须放在USING中

## **性能对比**

#### **RC隔离级别**

更新条数小(一般小于500条)，CASE WHEN和JOIN优于UDPATE。  
更新条数较大(千级别)，CASE WHEN效率迅速下降，UPDATE居中，推荐使用JOIN写法

#### **RR隔离级别**

JOIN性能优于CASE WHEN，CASE WHEN优于UPDATE。

#### **RC级别下的测试数据**

数据库实例规格：8核，20G内存，100G硬盘  
隔离级别：READ-COMMITTED  
连接方式：JDBC

| **更新方式** | **更新300条记录平均耗时** | **更新3000条记录平均耗时** |
| --- | --- | --- |
| UPDATE | 230ms | 560ms |
| CASE WHEN | 110ms | 1170ms |
| JOIN | 100ms | 320ms |

# 删除重复的数据，保留一条id最小的

delete from `tableName` where id in

(

select id from (

select \* from `tableName`

where `fieldName` in

(

select `fieldName` from `tableName` group by `fieldName` having count(`fieldName`) >1

)

and id not in (select min(id) from `tableName` group by `fieldName` having count(`fieldName`)>1)

) a

);