一， DDL

查看当前使用的数据库：select database（）；
切换数据库：use 切换数据库名；
查看数据库中的所有表：show tables；
查看表的字段信息：desc 表名；
添加列：alter table 表名 add 列名 数据类型；
修改列：alter table 表名 modify 列名 数据类型；
删除列：alter table 表名 drop 列名；
修改表名：rename table 表名 to 新表名；
修改表的字符集：alter table 表名 character set 字符集；
修改列名：alter table 表名 change列名 新列名 数据类型；
删除数据库：drop database 数据库名；
删除表：drop table 表名；
查看数据库建立的详细信息：show create database 数据库名；
创建指定字符集的数据库：create database 数据库名 character set 字符集；
创建默认字符集的数据库：create database 数据库名；

二，DML
查看指定：select *from 表名；
在mysql中，字符串类型和日期类型都需要使用单引号括起来
插入数据：insert into 表名（列名1，列名2....） values（列值1，列值2.....）
查看数据库编码的具体信息：show variable like ’ character% ‘；
临时更改客户端结果集的编码：set character_set_client = 字符集；
临时更改服务器结果集的编码：set character_set_results = 字符集；
修改表中的数据：update 表名 set 列名1 = 列值1，列名2 = 列值2，.....where 修改条件；
删除表中的数据：delete from 表名 where 删除条件；
注意：①delete删除表中的数据，表的结构还在，删除后的数据可以找回
②truncate 删除是把表直接drop掉，然后在创建一张同样的新表，删除的数据不能找回，执行
速度比delete快

三，DQL
查询返回的结果是一张虚表

查询数据：select 列名1，列名2.....from 表名 where 查询条件 group by 分组列名 having 分组后条件 order by排序列；

查询所有列：select *from 表名；

查询指定列：select 列名1，列名2.....from 表名；

查询条件：=，！=，<>，>，>=， <， <=

between a and b（含a含b）

in（set）：在集合set中

not in（set）：不再集合中

is null：是否非空

is not null：是否非空

and ：多个条件同时满足

or：多个条件只要满足一个就可以

not：不满足条件

模糊查询：select 列名1，列名2....from 表名 where 列名 like ’ _(表示一个不确定的字符)%(表示0个或多个不确定字符)’

去除指定列名的重复记录：select distinct 列名 from 表名；

select *，列名1+ifnull（列名2,0） from 表名；

排序：select *from 表名 order by 列名 desc/asc；

聚合函数：

count（列名）：统计指定列不为null的记录行数

max（列名）计算指定列的最大值，如果指定列是字符串类型，那么使用字典排序运算

min（列名）计算指定列的最小值，如果指定列是字符串类型，那么使用字典排序运算

sum（列名）计算指定列的数值和，如果指定列不是数值类型，那么计算结果是0

avg（列名）计算指定列的平均值，如果指定列不是数值类型，那么计算结果是0

分组查询：

凡是和聚合函数同时出现的列名，则一定要写在group by之后

select 列名1，sum（列名2）from 表名 group by 列名1；

select 列名1，sum（列名2）from 表名 group by 列名1 having 分组后的查询条件

注：having与where的区别:

1.having是在分组后对数据进行过滤.

where是在分组前对数据进行过滤

2.having后面可以使用分组函数(统计函数)

where后面不可以使用分组函数。

WHERE是对分组前记录的条件，如果某行记录没有满足WHERE子句的条件，那么这行记录不会参加分组；而HAVING是对分组后数据的约束。

limit：select *from 表名 limit 2,10；表示从第二行开始进行查询，每一次查询10条数据；

在多表查询中，若使用union关键字，则要求被合并的俩张表的列数和列的数据类型必须相同
使用主外键关系作为条件去除无用信息

子查询出现在where后作为被查询的条件的一部分，出现在from后作为表
all和any

数据库备份：
无需登录mysql：mysqldump -uroot -p密码 备份数据库名>备份路径：\\备份文件名.sql
数据库恢复：
方式1：登录mysql：source 文件路径\\文件名称.sql
方式2：不需要登录：mysql -uroot -p密码 数据库<文件路径:\\文件名.sql
在dos下，命令不能加；

alt+shift+z快速实现try-catch包裹

在mysql中关闭资源的时候，需要判断 if conn！=null才进行关闭操作

连接数据库的操作：

在src下创建一个file中，用于写properties的配置文件

如：dbinfo.properties

driverclass=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost:3306/anqili
root=root
password=AQL271422

创建连接以及关闭资源：

```
package CRUD;

import java.sql.*;
import java.util.ResourceBundle;

public class test {
	private static String driverclass;
	private static String url;
	private static String root;
	private static String password;

	static {
		//此对象是用于properties文件数据的
		ResourceBundle rBundle = ResourceBundle.getBundle("dbinfo");
		driverclass = rBundle.getString("driverclass");
		url = rBundle.getString("url");
		root  = rBundle.getString("root");
		password = rBundle.getString("password");
		try {
			Class.forName(driverclass);
		} catch (ClassNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}

//得到连接的方法
	public static Connection getConnection()throws Exception{

		Connection connection =
DriverManager.getConnection(url,root,password);
		return connection;
	}
```

```
        public static void closeAll(ResultSet rSet,Statement
statement,Connection connection)
        {
                if (rSet!=null) {

                        try {
                                rSet.close();
                        } catch (SQLException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
                if (statement!=null) {

                        try {
                                statement.close();
                        } catch (SQLException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
                if (connection!=null) {

                        try {
                                connection.close();
                        } catch (SQLException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
        }
}
```

mysql中的crud操作：

```
package CRUD;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

import org.junit.Test;

public class testcrud {
	//查询数据
@Test
public  void testSelect(){
	Connection connection = null;
	PreparedStatement statement = null;
	ResultSet rSet = null;
	try {
		connection = test.getConnection();
		String sql = "select *from emp2";
		statement = connection.prepareStatement(sql);
		rSet = statement.executeQuery("select *from emp2");
		while(rSet.next())
		{
			System.out.print(rSet.getObject(1)+"   ");
			System.out.print(rSet.getObject(2)+"   ");
			System.out.print(rSet.getObject(3)+"   ");
			System.out.print(rSet.getObject(4)+"   ");
			System.out.print(rSet.getObject(5)+"   ");
			System.out.print(rSet.getObject(6)+"   ");
			System.out.print(rSet.getObject(7)+"   ");
			System.out.print(rSet.getObject(8)+"   ");
			System.out.println();
```

```
                }
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }
        finally {
                test.closeAll(rSet, statement, connection);
        }
}
//增加数据
@Test
public void testCreate()
{
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet rSet = null;
        try {
                connection = test.getConnection();
                String sql = "insert into emp2 values(?,?,?,?,?,?,?,?)";
                statement = connection.prepareStatement(sql);
                statement.setInt(1, 7905);
                statement.setString(2, "anqili");
                statement.setString(3, "clerk");
                statement.setInt(4,7963);
                statement.setString(5, "2018-9-20");
                statement.setDouble(6, 10000.00);
                statement.setDouble(7, 500.00);
                statement.setInt(8, 30);

                int i  = statement.executeUpdate();
        if (i>0) {
                System.out.println("插入成功！");
        }
        else {
                System.out.println("插入失败！");
```

```
        }
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }
        finally {
                test.closeAll(rSet, statement, connection);
        }
}
//删除数据
@Test
public void testDelete()
{
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet rSet = null;
        try {
                connection = test.getConnection();
                String sql = "delete from emp2 where ename = ?";
                statement = connection.prepareStatement(sql);
                statement.setString(1,"linmingjun");
                int i  =statement.executeUpdate();
        if (i>0) {
                System.out.println("删除成功！");
        }
        else {
                System.out.println("删除失败！");
        }
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }
        finally {
                test.closeAll(rSet, statement, connection);
        }
```

```
}
//修改数据
@Test
public void testUpdate()
{
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet rSet = null;
        try {
                connection = test.getConnection();
                String sql = "update emp2 set ename = ? where ename = ?";
                statement = connection.prepareStatement(sql);
                statement.setString(1, "linmingjun");
                statement.setString(2, "anqili");
                int i  =statement.executeUpdate();
        if (i>0) {
                System.out.println("修改成功！");
        }
        else {
                System.out.println("修改失败！");
        }
        } catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
        }
        finally {
                test.closeAll(rSet, statement, connection);
        }
}

}
```