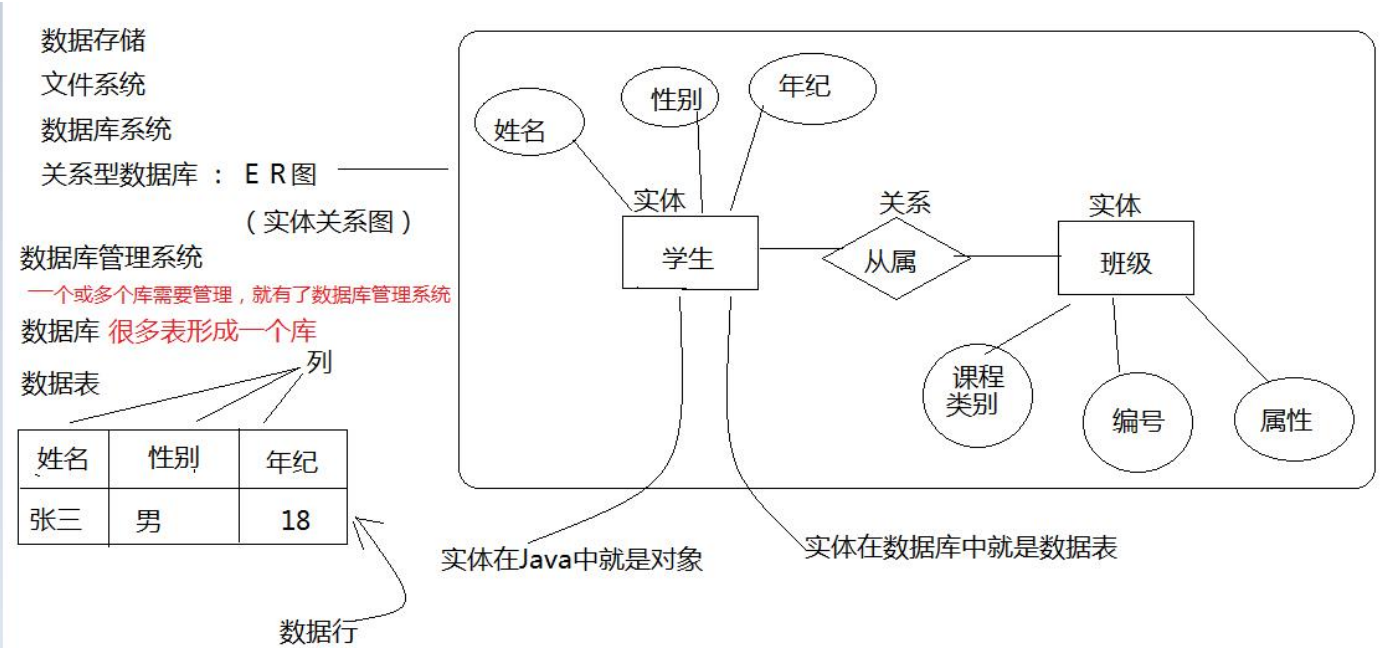
MySQL数据库基础

相关概念 ：存储数据的仓库

关系型数据库：关系型数据库：依赖于数据之间的关系，对数据进行描述与管理。

常见关系型数据库：**SQLite**（Android内置数据库系统） 、Access、SQLServer、**MySQL**、Oracle、DB2

ER模型图：



安装好的MySql，可以通过命令提示符输入：mysql -uroot -p123 进入数据库

数据存在数据表，表存在于库，库受数据库管理系统管理。一个系统可能有多个库。

库操作命令：

show databases; //查看所有的库

use 数据库名; //打开要操作的库

create database 数据库名;//创建库

drop database 数据库名;//删除库

SQL是结构化查询语言。遇到分号就是一个完整的操作命令。

SQL语言的分类

DDL：数据定义语言
DCL：数据控制语言
DML：数据操作语言： insert , delete , update
DQL:数据查询语言：select

SQL操作数据库表

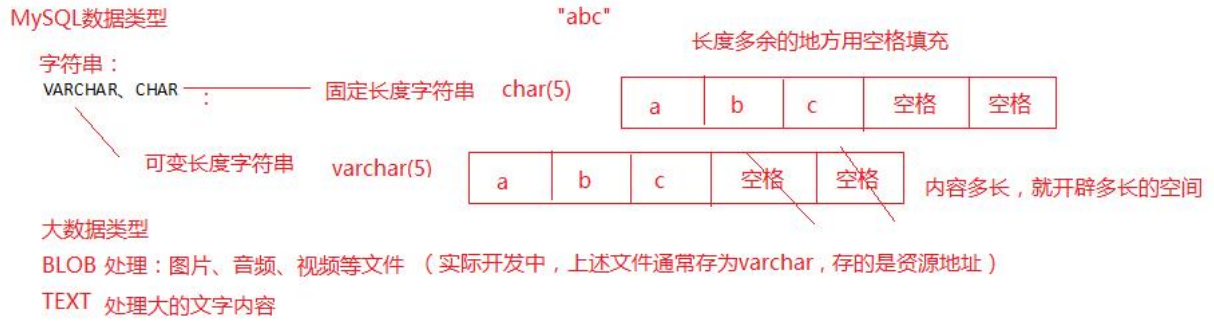
创建表：create table 表名称（name char(5)，age int，....等列）；

查看表：show tables 查看所有的表；

查看表结构：desc 表名称；

删除表：drop table 表名称；

MySQL数据类型:



数据类型：INTEGER 整数， CHAR 固定字符串、 VARCHAR 长度可变字符串

带约束的数据库表，对数据的操作提出要求。

非空：not null 对应的列必须要有数据

主键：primary key 对应列的数据为唯一标识，不可为空也不可重复

自增：auto_increment　对应列的值自动增加
用法：create　table　表名称（id　integer　primary　key　auto_increment，name　char(5)，age　int，....等列）；

操作数据库表中的数据：
增加：　insert　into　表名(列名,列名1.....)　valuse(对应表中列的数据类型值)；
删除：　delete　from　表名　where　表中列名=值；
修改：　update　表名　set　列名=值　where　列名=值2；
查询：　select　列名　form　表名；　select　*　form　表名；

查询的条件：

查非重复的所有数据：distinct　如：select　distinct　form　表名；
按某种条件范围查：where　列名>值　and　列名< 值；　列名>值　or　列名< 值　；　between　值　and　值；
按某两种条件查询：　where　in(值1，值2)；
按值 的相似性查询：　where　列名　like　"_ 值 %";
按排序查询：　表名　order　by　列名；　表名　order　by　列名　asc；　表名　order　by　列名　desc；
接聚集函数就是将数据按照列进行聚集操作：
求和：　sum(列名) from　表名；
按求和分组：　sum(列名) from　表名　group　by　列名；
平均值：　avg(列名)　from　表名；
最大值：　max()；
最小值：　min();

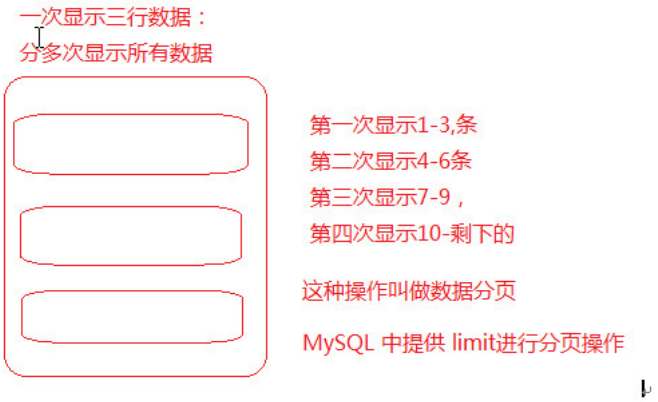
按列名统计个数：　count(列名)　from　表名　group　by　列名；　按某列名统计在表中的个数并按列名分组。

SQL中limit关键字

主要用于分条查询，一次显示多少条信息出来。 查询表中的所有数据并每次显示n条到设备。

limit关键字不是标准sql的关键字，只能在mysql数据库里面使用，实现分页的功能。

用法： select　*　form　表名　**limit　n,m;**　从指定位置第m行开始，取n条数据显示出来。　m是从0开始



取 3 条数据：

```
mysql> select * from user limit 3;
```

从指定位置第 4 行开始，取 3 条数据

```
mysql> select * from user limit 3,3;
```

JDBC操作数据库

jdbc用于连接应用程序与数据库服务器实现数据交互

原理：java定义一套规范，数据厂商提供驱动，程序员使用驱动连接到对应的数据库服务器进行数据操作。

JDBC中常用的API：

DriverManager ：驱动管理　　import　java.sql.DriverManager;

Connection ： 数据库连接　　import　java.sql.Connection;

Statement ： SQL语句的操作对象，执行SQL，返回结果　　import　java.sql.Statement;

ResultSet ： 表示数据库结果集的数据表，通常通过执行查询数据库的语句生成　　import　java.sql.ResultSet;

1、导入驱动，数据库的驱动，不同的数据库提供驱动使用jar的形式提供的，需要把jar包放到项目里面，相当于安装了数据库的驱动

2、加载驱动　　DriverManager.registerDriver(new　Driver());

推荐加载驱动的方法：　Class.forName("com.mysql.jdbc.Driver");

3、连接数对象据库对象

```
Connection　connection　=　DriverManager.getConnection("jdbc:mysql://localhost:3306/myd005",　"root",　"123");
```

3、获取SQL语句的操作对象

```
Statement　statement　=　connection.createStatement();
```

4、写执行SQL命令的语句

```
String　str　=　"insert　into　user(id,name,sex,age)　values(9572,'小明','男',22)";
```

5、执行SQL语句向数据库指定的表里增加，删除，修改，查询

增加，删除，修改用：statement.executeUpdate(str);

查询用：　ResultSet　query　=　statement.executeQuery(str);

```
while(query.next()){
                        query.getInt("列名");
                        query.getString("列名");
```

6、关闭SQL语句执行集合对象　　，关闭SQL语句操作对象，关闭连接数对象据库对象

```
query.close()　　statement.close();　　connection.close();
```

最后可以封装一个工具类：在封装的时候需要导入第三方jar包，必须使用工具完成

SQL注入

网络入侵最常用的一种手法：SQL注入（注入一段包含SQL语句命令的代码，破坏你的查询逻辑，绕过后台的检测判断），从而获得黑客想要的信息。 比如：在查询语句中输入：　and　or　组合

使用PreparedStatement预编译对象防止sql注入,创建PreparedStatement对象 prepareStatement(String sql),PreparedStatement接口的父接口Statement，预编译就是将查询逻辑预先编译好，后面输入的内容，永远被视为数据，而非逻辑代码。

```
conn = MyJdbcUtils.getConnection();
// 编写sql
String sql = "select * from user where username=? and password=?";
// 对sql进行预编译
psmt = conn.prepareStatement(sql);
// 设置参数
psmt.setString(1, username);
psmt.setString(2, password);
// 执行sql
rs = psmt.executeQuery();
```

```java
public static void login1() {
    String name = "张三";
    String pwd = "111111";
    Connection connection = DBUtil.getConnection();
    // 解决方案：使用预编译对象
    String sql = "select count(*) from userinfo where name= ? and pwd= ? ";
    ResultSet rs = null;
    try {
        PreparedStatement preparedStatement = connection
                .prepareStatement(sql);
        preparedStatement.setString(1, name);
        preparedStatement.setString(2, pwd);
        //rs = preparedStatement.executeQuery(sql); //注意不要再次传入SQL语句
        rs = preparedStatement.executeQuery();
        if (rs.next()) {
            if (rs.getInt(1) > 0) {
                System.out.println("登录成功");
            } else {
                System.out.println("登录失败");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
```

查询的条件变成参数进行传递