**说明书**

**项目软件系统设计**

目录

[1.软件系统结构 1](#_Toc74064499)

[1.1总体架构 1](#_Toc74064500)

[1.2子模块的设计 2](#_Toc74064501)

[1.2.1模块功能 2](#_Toc74064502)

[1.2.2模块结构 3](#_Toc74064503)

[1.2.3处理流程 3](#_Toc74064504)

[1.2.4界面原型 4](#_Toc74064505)

[1.3 接口设计 6](#_Toc74064506)

[1.3.1外部接口 7](#_Toc74064507)

[1.3.2内部接口 7](#_Toc74064508)

[2.数据库设计 7](#_Toc74064509)

[2.1数据库说明 8](#_Toc74064510)

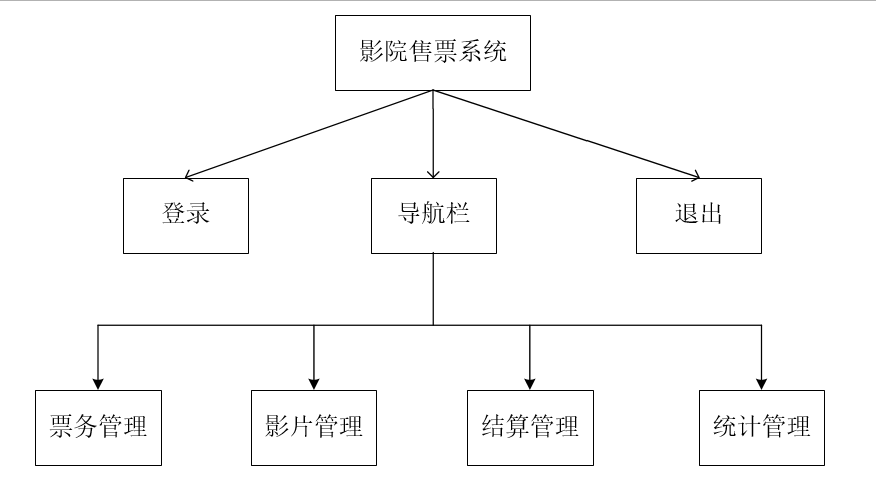
[2.2数据库逻辑结构 10](#_Toc74064511)

[3.软件重用方案 12](#_Toc74064512)

[4.设计关键类的重点服务 13](#_Toc74064513)

# 1.软件系统结构

## 1.1总体架构

****

**图1-1总体框架**

新视界电影院售票管理系统采用B/S结构的软件设计模式，运用Tomcat服务器，基于JSP技术，Servlet技术，MYSQL数据库来实现高校科研管理系统的科研项目管理的主要功能。

B/S整体的结构如下:

第一层是浏览器，即客户端，只有简单的输入输出功能，处理极少部分的事务逻辑。由于客户不需要安装客户端，只要有浏览器就能上网浏览，所以它面向的是大范围的用户，所以界面设计得比较简单通用。

第二层是WEB服务器，扮演着信息传送的角色。当用户想要访问数据库时，就会首先向WEB服务器发送请求，WEB服务器统一请求后会向数据库服务器发送访问数据库的请求，这个请求是以SQL语句实现的。

第三层是数据库服务器，他扮演着重要的角色，因为它存放着大量的数据。当数据库服务器收到了WEB服务器的请求后，会对SQL语句进行处理，并将返回的结果发送给WEB服务器，接下来，WEB服务器将收到的数据结果转换为HTML文本形式发送给浏览器，也就是我们打开浏览器看到的界面**。**

## 1.2子模块的设计

实质上，电影票销售系统系统的综合性相对较强，复杂程度相对较高，可对现有软件进行充分利用，进行系统设计与规划。编写一个完整的电影销售系统，需要涉及网页界面、处理程序、MySQL 后台数据库系统等，在网站页面上显示电影简介、电影海报、电影评分等诸多内容。数据库方面需要存储用户信息，电影信息以及排片信息等。

### 1.2.1模块功能

系统按照功能主要分为注册用户、管理员两个模块，用户具有用户注册、用户登录、查看电影、付款、退订的功能，管理员具有退订、编辑电影简介内容、后台电影添加删除、查看用户信息的功能。如图1-2所示



**图1-2系统模块功能图**

### 1.2.2模块结构

系统总体结构可分为系统前台和系统后台两个功能模块。

前台功能实现以下功能，用户注册、用户登录、电影信息、电影搜索、付款、个人设置。

系统后台功能实现以下功能，用户管理、排片管理和系统设置。

系统总体功能结构如图1-3所示：

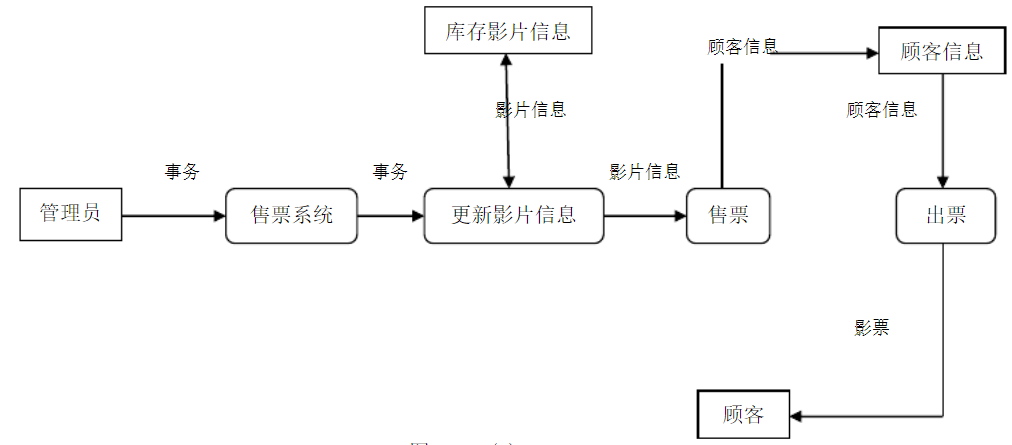
****

**图1-3系统总体功能结构图**

### 1.2.3处理流程

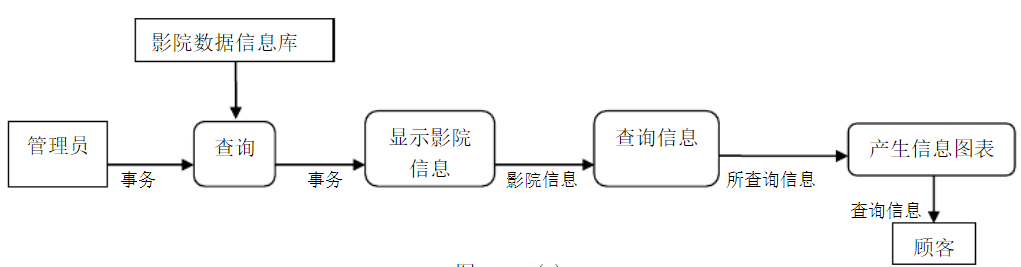
（1）前台模块

为用户提供影票销售功能；后台判断库存票是否充足，然后将电影票信息反馈给用户。具体电影销售模块流程如图1-4所示：



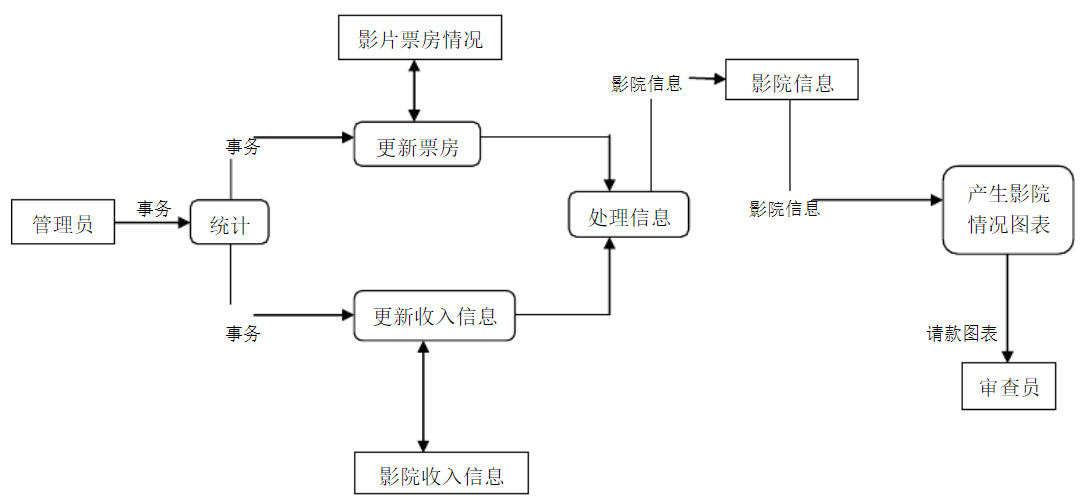
**图1-4电影销售处理流程图**

用户可以查询拍片信息以及电影简介，拍片等信息，具体流程如图1-5所示：



（2）统计系统数据模块

为用户提供影片的票房，评分与口碑，为管理员反馈影院收益情况。具体流程如图1-6所示：



**图1-6商品交易处理流程图**

### 1.2.4界面原型

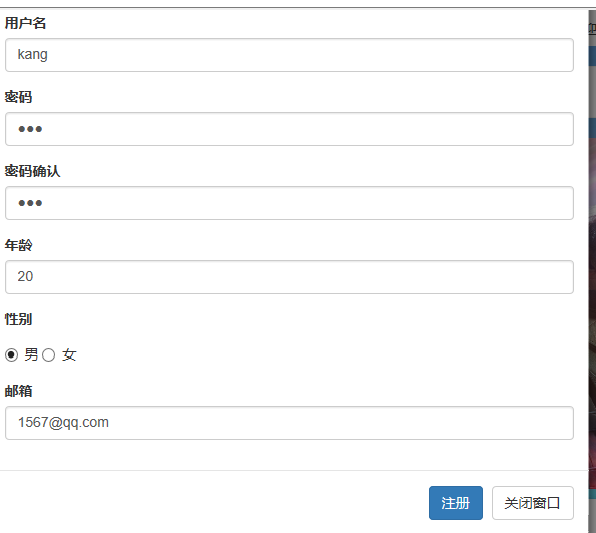
本系统界面原型使用墨刀原型开发工具设计实现，

（1）首页界面：



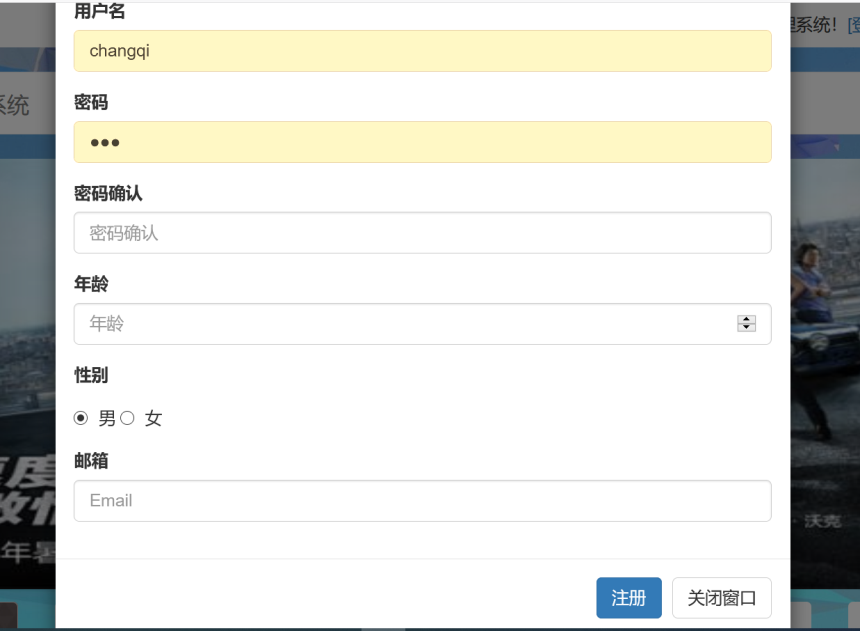
**图1-7 首页界面原型图**

（2）登录界面



**图1-8 登录界面原型图**

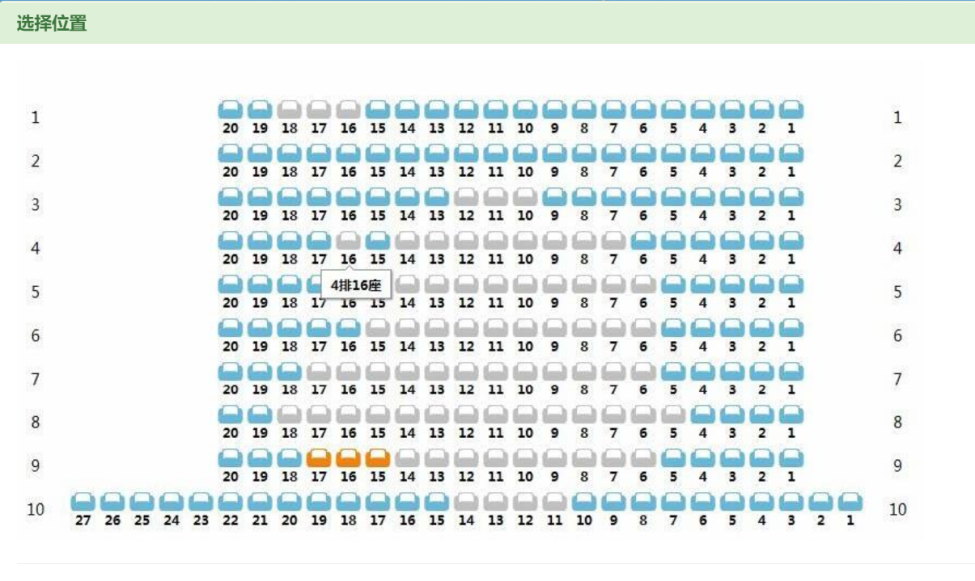
（3）注册界面



**图1-9注册界面原型图**

（4）部分物品类别展示界面

➀选座



**图1-10 选座界面原型图**

➁付款界面



**图1-11 付款界面原型图**

## 1.3 接口设计

本系统软件的接口设计遵循六大基本原则：分别是：单一职责原则、依赖倒置原则、接口隔离原则、迪米特法则、里氏替换原则、开闭原则。这6个原则是建立稳定的，灵活和健壮设计的基础，但设计的时候还要根据实际情况考虑。

（1）单一职责原则(SRP)：该原则要求我们在设计类或者接口的时候。尤其在设计接口的时候把职责分清楚，通常一个职责不是单一的方法，是一类方法的组合。

（2）依赖倒置原则：为高层模块不应该依赖底层模块，他们都应该依赖抽象。抽象不能依赖细节。细节应该依赖抽象，这个原则要求设计的时候尽量用抽象（抽象类或者接口）把各个模块独立开来，实现解耦，使各模块相对独立。简单来说就是要用面向接口设计。

（3）接口隔离原则(ISP)：接口分两种，分别为类实例接口和类接口。这个原则依赖建立在最小的接口之上，依赖自己需要的接口。

（4）迪米特法则(LoD)：也称为最少知识原则。就是一个对象应该对其他对象有最少的了解。这个法则的本质就是解耦，解耦是有限度的不能为了解耦而解耦。

（5）里氏替换原则：（所有引用基类的地方，都能透明地使用其子类的对象。）简单来说就是父类出现的地方，替换为子类不会产出异常。里氏替换原则的好处就是增强程序的健壮性，保持程序的兼容性。

（6）开闭原则：软件的实体如类，模块和函数应该对扩展开发，对修改关闭。这个原则要求我们设计的系统扩展性好，因为需要不会一直不变的，我们需要应对的永远是变化。

### 1.3.1外部接口

（1）用户界面：在界面设计上，应做到简单明了，易于操作，并且要注意到界面的布局，应突出的显示重要以及出错信息。外观上也要做到合理化，考虑到用户多对Windows风格较熟悉，所以该系统尽量向这一方向靠拢。

（2）软件与硬件接口：本系统设有人机操作界面，考虑到操作简单，易于管理方面，主要硬件与接口设备为pc、鼠标、键盘。而软件接口主要以Windows平台为基本平台。

### 1.3.2内部接口

们用到了数据库这种的软件接口。从计算机系统来讲，计算机，数据库资源、第三方主体客户、开发商共用一套数据模式，开发商利用系统完成授权，借助规范化的计算机运行原理，进一步完成数据的交换工作，数据库模式属于软件开发商定制的数据模式，包括Oracle以及sqslrver，在文件交换、程序函数数据库开发的难度较低，灵活性较高，但是在特定应用的过程中，数据库应用的空间较为狭窄，开发商对其进行个性化的定制使用。

数据库应用的流程为系统一、数据库、系统二、数据库、系统一。

# 2.数据库设计

数据库的设计关系到整个应用系统的运行效率，数据库设计得好，不仅有利于日常数据的维护更新，而且可以提高系统的运行效率，缩短数据查询响应周期，增加网站的流量。合理的数据库设计可以使围绕它支持的Web页面的Java代码简单化，易于实现，并且可以提高数据存储的效率，保证数据的完整一致。电影销售系统采用MySQL作为后台数据库开发工具。

## 2.1数据库说明

概念模型用于信息世界的建模，与具体的DBMS无关。为了把现实世界中的具体事物抽象、组织为某一DBMS支持的数据模型。人们常常首先将现实世界抽象为信息世界，然后再将信息世界转换为机器世界。也就是说，首先把现实世界中的客观对象抽象为某一种信息结构，这种信息结构并不依赖于具体的计算机系统和具体的DBMS，而是概念级的模型，然后再把模型转换为计算机上某一个DBMS支持的数据模型。实际上，概念模型是现实世界到机器世界的一个中间层次。

信息世界中包含的基本概念有实体和联系。

（1）实体

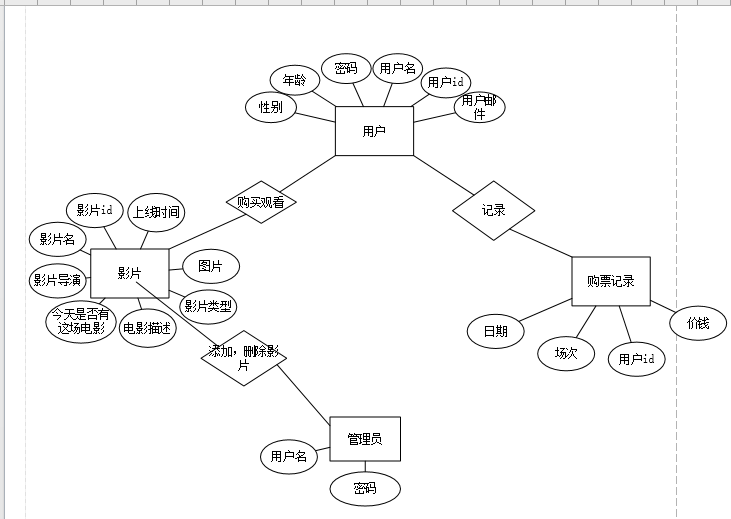
客观存在并可相互区别的事物称为实体。实体可以是具体的人、事、物，也可以是抽象的概念或联系。例如，一个学生、一门课、一个供应商、一个部门、一本书、一位读者等都是实体。

（2）联系

在现实世界中，事物内部以及事物之间是有联系的，这些联系在信息世界中反映为实体内部的联系和实体之间的联系。实体内部的联系通常是组成实体的各属性之间的联系。两个实体型之间的联系可分为3类，一对一联系，(1:1)；一对多联系(1:n)；多对多联系(m:n)。

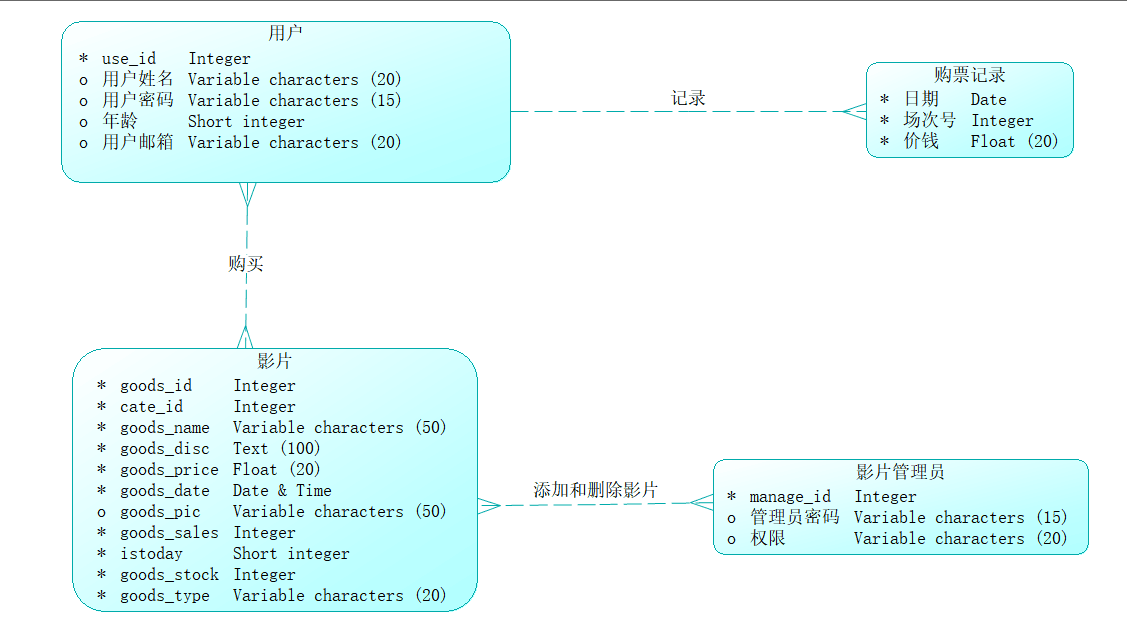
概念模型是对信息世界建模，所以概念模型应该能够方便、准确地表示信息世界中的常用概念。概念模型的表示方法很多，其中最为常用的是P.P.S.Chen于1976年提出的实体，联系方法(Entity-Relationship Approach)简记为E-R表示法)。该方法用E-R图来描述现实世界的概念模型，称为实体-联系模型，简称E-R模型。

根据数据流程分析，绘制校园二手商品交易系统的全局E-R模型如图2-1所示。



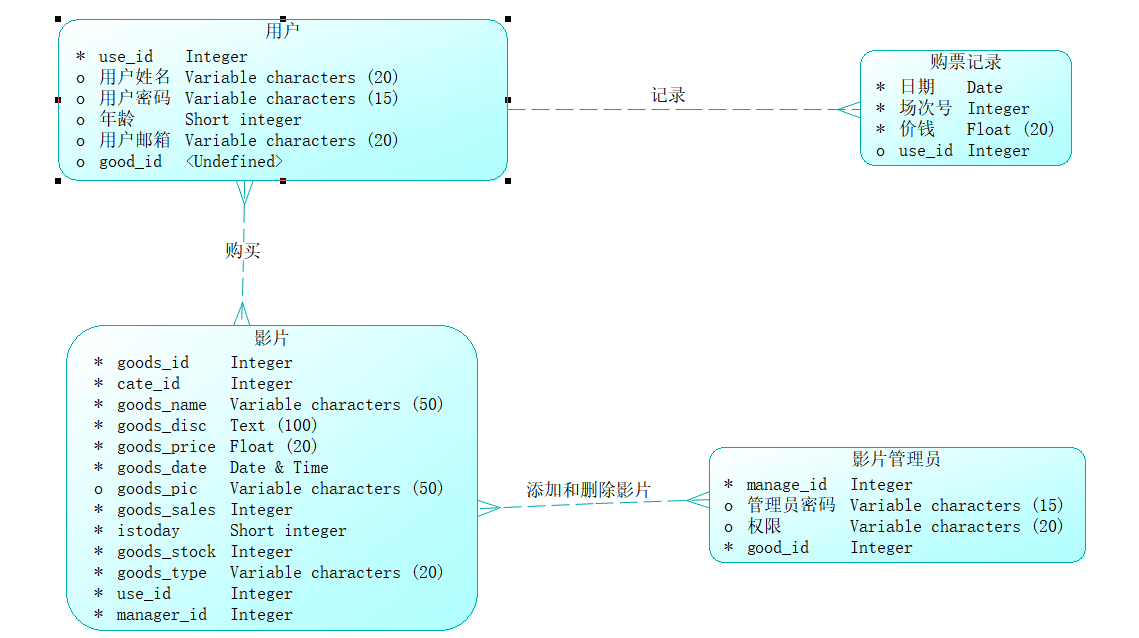
**图2-1系统全局E-R图**

1.概念模型



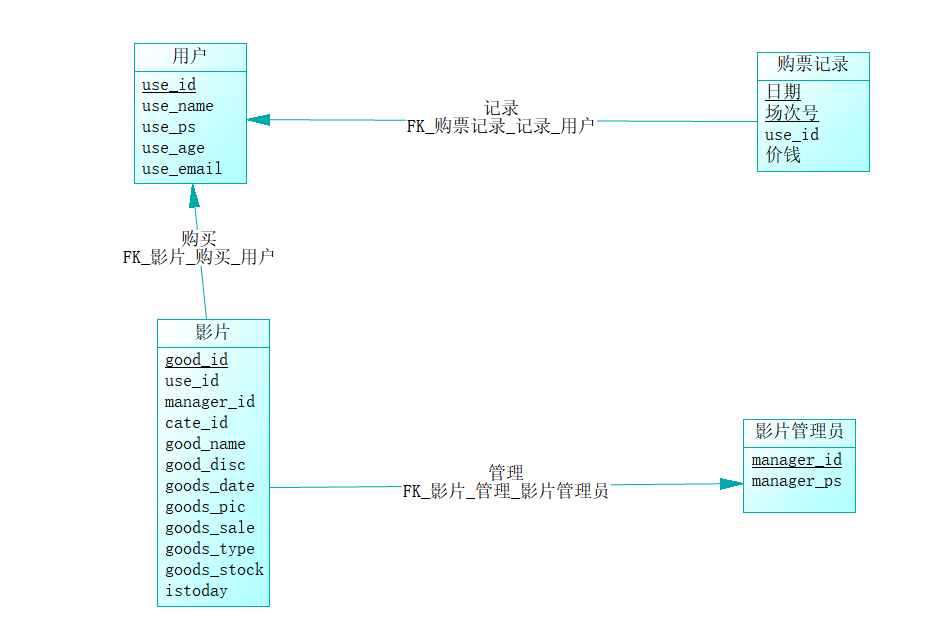
**图 2-2 概念模型图**

2.逻辑模型



**图 2-3 逻辑模型图**

3.物理模型



**图 2-4 物理模型图**

## 2.2数据库逻辑结构

数据库逻辑设计主要是把数据库概念设计时设计好的基本E-R图转换为与选用DBMS产品所支持的数据模型相符合的逻辑结构。它包括数据项、记录及记录间的联系、安全性和一致性约束等等。导出的逻辑结构是否与概念模式一致，从功能和性能上是否满足用户的要求，要进行模式评价。

本系统数据库名称为db\_secondhandmarket，数据库中包括：

1）种类表（category）

2）影片表（goods）

3）影片内容表（goodsdetail）

4）影片种类表（goodsdetailtype）

5）图片表（pic）

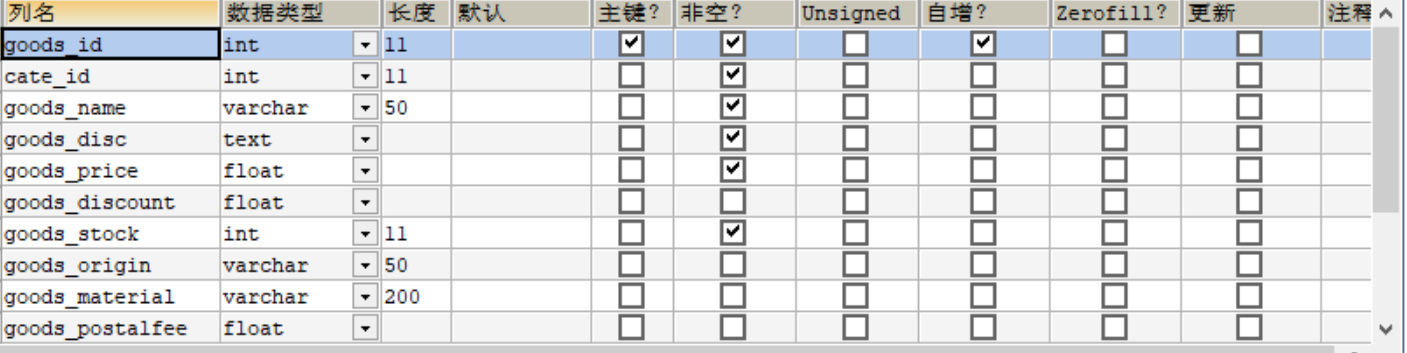
6）用户表（user）

各表数据结构如下：

（1）种类表（category）用来记录是管理员还是用户



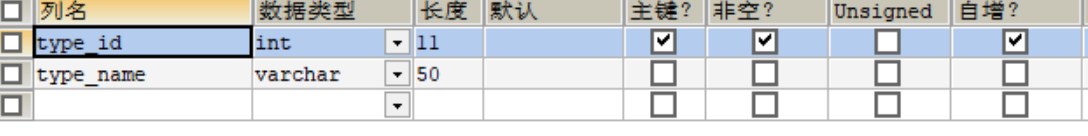
（2）影片表（goods）用于记录影片



（3）影片内容表（goodsdetail）用于记录影片细节



（4）影片种类表（goodsdetailtype）用于记录影片种类



（5）图片表（pic）用于记录图片



（6）用户表（user）



# 3.软件重用方案

从广义的角度来说，本系统在以下三个层次进行了重用，知识的重用，方法和标准的重用以及软件成分的重用。

知识的重用是如软件工程等相关知识的重用，应用相同的知识设计规划、建设不同的软件产品。方法和标准的重用是比如本产品是利用面向对象方法设计又或者是采用国家标准局制定的软件开发文档规范。

软件成分的重用进一步我们划分为三个级别，即代码重用，设计结果重用和分析结果重用。而本软件产品的重用主要在源代码级，这与我们的开发工具有很大的关系，我们选用的开发工具是eclipse和IDE，采用spring MVC框架。利用该框架可以快速开发出一个程序的框架，是一个灵活性较强的框架。

我们的重用设计是建立在MVC框架之上。我们分析可能重用部分的实例有两种方式，一种是继承类库中的构件要用到的基本功能的类。主要是一些界面元素，如菜单活框、列表框，这些构件在很多模块中都要用到，且处理逻辑大致相同，并进行扩充其处理逻辑，如增加输入合法性检测等，这样我们在使用这些经过扩充的构件时不必每个都去重复那些处理逻辑，大大提高了效率，而且，这些构件都是经过测试或其他人使用过的，质量也有保证。

另一种方式是我们的构件在类库找不到相似的类，我们将从头创建自己的类，但为了将自己创建的类，如用户权限处理排序打印预览等，也纳入MVC的框架，我们自己创建的类都继承框架的一个抽象类，这样做的好处是把自己创建的类纳入Controller类的层次化管理，且这也是为实现虚拟函数和动态联编方便。我们也为每个件的功能和接口建立了文档，供应用开发中使用。这样我们就在MVC的框架下，建立了我们自己的一个重用构件库。最后是在应用中重用构件库中的构件。当用户需求变化时，我们能用构件库快速重构我们的应用。

使用重用明显会将开发效率提高一个层次，它不仅提高了开发效率，还保证了应用的风格和质量。特别是对我们这种新手较多的团队开发，采用重用的方法，让有经验的成员负责整个应用的框架，让新手使用重用来创建应用，这非常有利于提高效率，在保证质量的同时也为新手提供一个循序渐进的学习机会，有利于新手的成长。当然使用重用容易，但自己建立重用构件库也是需要代价的，是一个需时间积累的过程，也有一定风险。我们在重用中也遇到了一些问题，一个比较突出的问题是我们的重用主要是在源代码级，因此我们的开发必须在MVC的框架下。

我们的重用主要是在源代码级，通过类的继承来实现。其实可重用的范围是很大的，如设计的重用，测试用列的重用，可运行的代码的重用等。我们想将来扩大重用的粒度，在框架基础上，进一步根据我们单位的软硬件环境定制出一个适用于我们系统的框架，如加进注册功能，数据库连接的功能等。这样，我们可直接重用这个框架，这可以极大地提高软件的开发和维护的效率。最后，我们还设想，将来应有专门的人员管理重用，把重用的维护和应用开发分开，责任明确这样可以更进一步地做好重用工作。

# 4.设计关键类的重点服务

本系统主要设计了三个大的关键类:servers , user, goods,即管理员类，用户类，商品类。基于这三个父类的基础上，继承更行，查询，修改等子类。

GoodsServers类主要负责对电影的上架，下架以及排片的工作；

HomeServers类只要负责用户的注册信息入库以及增删改查用户的个人信息。

Category类负责定义电影的分类；

Goods类负责定义电影的分类、ID、名字、图片、评分等信息；

Goodsdetail类时Goods的子类，负责记录电影的内容和名字；

Goodsdetailtype是Goodsdetail的子类，负责电影具体分类的工作；

Pic类用于记录图片信息；

User类负责记录一些用户的个人信息，对用户的属性信息操作服务。

以上为本系统设计关键类的对应重点服务，遵循软件工程中软件设计的模块化原则，力求尽量达到“高内聚，低耦合”的程度，降低类的复杂度，且使方法属性单一，逻辑关系简单。这样可以提高类的可读性，降低功能模块之间的关联性。