

**票务系统开发文档**



**目 录**

[一 项目开发计划 6](#_Toc422368052)

[1.1 简介 6](#_Toc422368053)

[1.1.1 编写目的 6](#_Toc422368054)

[1.1.2 项目背景 6](#_Toc422368055)

[1.1.3 定义 6](#_Toc422368056)

[1.2 项目概述 6](#_Toc422368057)

[1.2.1 工作内容 6](#_Toc422368058)

[1.2.2 条件与限制 6](#_Toc422368059)

[1.2.3 运行环境 7](#_Toc422368060)

[1.2.4 验收标准 7](#_Toc422368061)

[1.2.5 实施计划 7](#_Toc422368062)

[1.2.6 人员及分工 7](#_Toc422368063)

[二 可行性研究报告 8](#_Toc422368064)

[2.1 引言 8](#_Toc422368065)

[2.1.1 编写目的 8](#_Toc422368066)

[2.1.2 背景 8](#_Toc422368067)

[2.1.3 定义 8](#_Toc422368068)

[2.1.4 参考资料 8](#_Toc422368069)

[2.2 可行性研究的前提 8](#_Toc422368070)

[2.2.1 要求 8](#_Toc422368071)

[2.2.2 目标 9](#_Toc422368072)

[2.2.3 条件、假定和限制 9](#_Toc422368073)

[2.2.4 进行可行性研究的方法 9](#_Toc422368074)

[2.3 所建议的系统 9](#_Toc422368075)

[2.3.1 对所建议系统的说明 9](#_Toc422368076)

[2.3.2 处理流程和数据流程 9](#_Toc422368077)

[2.4 投资及效益分析 10](#_Toc422368078)

[2.4.1 支出 10](#_Toc422368079)

[2.4.1.1 基本建设投资 10](#_Toc422368080)

[2.4.1.2 其他一次性支出 10](#_Toc422368081)

[2.4.1.3 非一次性支出 10](#_Toc422368082)

[2.4.2 收益 10](#_Toc422368083)

[2.4.2.1 一次性收益 10](#_Toc422368084)

[2.4.2.2 非一次性收益 11](#_Toc422368085)

[2.4.2.3 不可定量的收益 11](#_Toc422368086)

[2.4.3 收益/投资比 11](#_Toc422368087)

[2.4.4 投资回收周期 11](#_Toc422368088)

[2.5 社会因素方面的可能性 11](#_Toc422368089)

[2.5.1 法律方面的可行性 11](#_Toc422368090)

[2.5.2 使用方面的可行性 11](#_Toc422368091)

[2.5.3 结论 11](#_Toc422368092)

[三 需求分析报告 12](#_Toc422368093)

[3.1 引言 12](#_Toc422368094)

[3.1.1 编写的目的 12](#_Toc422368095)

[3.1.2 背景 12](#_Toc422368096)

[3.1.3 定义 12](#_Toc422368097)

[3.1.4 参考文献 12](#_Toc422368098)

[3.2 任务概述 12](#_Toc422368099)

[3.2.1 目标 12](#_Toc422368100)

[3.2.2 用户特点 12](#_Toc422368101)

[3.2.3 假定和约束 13](#_Toc422368102)

[3.3 需求规定 13](#_Toc422368103)

[3.3.1 系统功能需求 13](#_Toc422368104)

[3.3.2 客户端系统功能 13](#_Toc422368105)

[3.3.2.1 普通用户 13](#_Toc422368106)

[3.3.2.2 机场人员 13](#_Toc422368107)

[3.3.3 服务器端系统功能 14](#_Toc422368108)

[3.4 数据字典 15](#_Toc422368109)

[3.4.1 系统中的数据元素： 15](#_Toc422368110)

[3.5 项目模块划分 16](#_Toc422368111)

[3.6 运行环境规定 17](#_Toc422368112)

[3.6.1 硬件要求： 17](#_Toc422368113)

[3.6.2 软件要求 17](#_Toc422368114)

[3.7 测试 17](#_Toc422368115)

[3.8 需求小结 17](#_Toc422368116)

[四 概要设计说明书 18](#_Toc422368117)

[4.1 引言 18](#_Toc422368118)

[4.1.1 编写目的 18](#_Toc422368119)

[4.1.2 项目背景 18](#_Toc422368120)

[4.1.3 定义 18](#_Toc422368121)

[4.1.3.1 专门术语 18](#_Toc422368122)

[4.1.3.2 缩写 19](#_Toc422368123)

[4.1.4 参考资料 19](#_Toc422368124)

[4.2 任务概述 19](#_Toc422368125)

[4.2.1 目标 19](#_Toc422368126)

[4.2.1.1 运行环境 19](#_Toc422368127)

[4.2.1.2 需求概述 19](#_Toc422368128)

[4.3 总体设计 20](#_Toc422368129)

[4.3.1 处理流程 20](#_Toc422368130)

[4.3.1.1 客户机程序流程 20](#_Toc422368131)

[4.3.1.2 服务器程序流程 21](#_Toc422368132)

[4.3.2 总体结构和模块外部设计 22](#_Toc422368133)

[4.3.2.1 客户机部分 22](#_Toc422368134)

[4.3.2.2 服务器部分 23](#_Toc422368135)

[4.3.3 功能分配 24](#_Toc422368136)

[4.4 接口设计 24](#_Toc422368137)

[4.4.1 外部接口 24](#_Toc422368138)

[4.4.1.1 用户界面 24](#_Toc422368139)

[4.4.1.2 软件接口 24](#_Toc422368140)

[4.4.1.3 硬件接口 24](#_Toc422368141)

[4.4.2 内部接口 24](#_Toc422368142)

[4.5 数据库设计 24](#_Toc422368143)

[4.5.1 数据表的设计 24](#_Toc422368144)

[4.5.1.1 1.用户表（users） 24](#_Toc422368145)

[4.5.1.2 登录表（login） 25](#_Toc422368146)

[4.5.1.3 订单表（order） 25](#_Toc422368147)

[4.5.1.4 票务信息（tickets） 25](#_Toc422368148)

[4.5.1.5 信息表（flight\_info） 25](#_Toc422368149)

[4.5.1.6 7.座位表(seat) 26](#_Toc422368150)

[4.5.1.7 8.用户操作记录表（deal\_log） 26](#_Toc422368151)

[4.5.1.8 9.动态表（state） 26](#_Toc422368152)

[4.5.2 物理数据结构设计 26](#_Toc422368153)

[4.5.3 数据结构与程序的关系 29](#_Toc422368154)

[4.6 运行设计 29](#_Toc422368155)

[4.6.1 运行模块的组合 29](#_Toc422368156)

[4.6.2 运行控制 29](#_Toc422368157)

[4.6.3 运行时间 29](#_Toc422368158)

[4.7 出错处理设计 29](#_Toc422368159)

[4.7.1 出错输出信息 29](#_Toc422368160)

[4.7.2 出错处理对策 30](#_Toc422368161)

[4.8 安全保密设计 30](#_Toc422368162)

[4.9 维护设计 30](#_Toc422368163)

[五 详细设计说明书 31](#_Toc422368164)

[5.1 引言 31](#_Toc422368165)

[5.1.1 编写目的 31](#_Toc422368166)

[5.1.2 背景 31](#_Toc422368167)

[5.1.3 定义 31](#_Toc422368168)

[5.1.4 参考资料 31](#_Toc422368169)

[5.2 程序系统的结构 31](#_Toc422368170)

[5.2.1 Server类图： 31](#_Toc422368171)

[5.2.2 Cilent类图： 32](#_Toc422368172)

[5.3 Server端程序说明 33](#_Toc422368173)

[5.3.1 ServerStart类设计说明 33](#_Toc422368174)

[5.3.1.1 ServerStatrt()方法 33](#_Toc422368175)

[5.3.1.2 ServerLogon()方法 34](#_Toc422368176)

[5.3.1.3 ServerGUI()方法 34](#_Toc422368177)

[5.3.2 database类设计说明 34](#_Toc422368178)

[5.3.2.1 database()方法 34](#_Toc422368179)

[5.3.3 Userdb类设计说明 35](#_Toc422368180)

[5.3.3.1 Userdb()方法 35](#_Toc422368181)

[5.3.3.2 isExit()方法 35](#_Toc422368182)

[5.3.3.3 updatedata()方法 35](#_Toc422368183)

[5.3.3.4 InsertNew(User user) 35](#_Toc422368184)

[5.3.3.5 QueryData() 35](#_Toc422368185)

[5.3.3.6 CloseAll() 36](#_Toc422368186)

[5.3.4 Flightdb类设计说明 36](#_Toc422368187)

[5.3.4.1 Flightdb()方法 36](#_Toc422368188)

[5.3.4.2 isExit()方法 36](#_Toc422368189)

[5.3.4.3 updatedata()方法 36](#_Toc422368190)

[5.3.4.4 InsertNew(Flight flight) 37](#_Toc422368191)

[5.3.4.5 QueryData() 37](#_Toc422368192)

[5.3.4.6 CloseAll() 37](#_Toc422368193)

[5.3.5 UserServer类设计说明 37](#_Toc422368194)

[5.3.5.1 UserServer ()方法 37](#_Toc422368195)

[5.3.5.2 Query()方法 38](#_Toc422368196)

[5.3.6 FlightServer类设计说明 38](#_Toc422368197)

[5.3.6.1 FlightServer ()方法 38](#_Toc422368198)

[5.3.6.2 Query()方法 38](#_Toc422368199)

[5.3.7 User类设计说明 38](#_Toc422368200)

[5.3.7.1 User()方法 38](#_Toc422368201)

[5.3.7.2 getID()方法 38](#_Toc422368202)

[5.3.7.3 setID()方法 39](#_Toc422368203)

[5.3.7.4 其他类似方法 39](#_Toc422368204)

[5.3.8 Flight类设计说明 39](#_Toc422368205)

[5.3.8.1 Flight()方法 39](#_Toc422368206)

[5.3.8.2 getnumofF()方法 39](#_Toc422368207)

[5.3.8.3 setnumofF()方法 39](#_Toc422368208)

[5.3.8.4 其他类似方法 39](#_Toc422368209)

[5.3.9 Administrator类设计说明 40](#_Toc422368210)

[5.3.9.1 addUserData()方法 40](#_Toc422368211)

[5.3.9.2 delateUserData()方法 40](#_Toc422368212)

[5.3.9.3 addFlightLine ()方法 40](#_Toc422368213)

[5.3.9.4 delateFlightLine ()方法 41](#_Toc422368214)

[5.3.10 serverThread类设计说明 41](#_Toc422368215)

[5.3.10.1 serverThread ()方法 41](#_Toc422368216)

[5.3.10.2 run()方法 41](#_Toc422368217)

[5.3.10.3 WriteControl()方法 41](#_Toc422368218)

[5.3.10.4 DealRequest()方法 42](#_Toc422368219)

[5.3.11 getAllMessage类设计说明 42](#_Toc422368220)

[5.3.11.1 getInformation ()方法 42](#_Toc422368221)

[5.4 client端程序说明： 42](#_Toc422368222)

[5.4.1 类Customer设计说明 42](#_Toc422368223)

[5.4.1.1 login()方法 42](#_Toc422368224)

[5.4.1.2 register()方法 43](#_Toc422368225)

[5.4.1.3 exit()方法 43](#_Toc422368226)

[5.4.1.4 queryFlights()方法 43](#_Toc422368227)

[5.4.1.5 orderFlight()方法 44](#_Toc422368228)

[5.4.1.6 queryOrders()方法 45](#_Toc422368229)

[5.4.2 类order设计说明 46](#_Toc422368230)

[5.4.2.1 clearOrder()方法 46](#_Toc422368231)

[5.4.2.2 deleteFlight()方法 46](#_Toc422368232)

[5.4.2.3 addFlight()方法 47](#_Toc422368233)

[5.4.3 类account设计说明 47](#_Toc422368234)

[5.4.3.1 decreaseMoney()方法 47](#_Toc422368235)

[5.4.3.2 addMoney()方法 47](#_Toc422368236)

[5.4.3.3 getID()方法 47](#_Toc422368237)

[5.5 注释设计 47](#_Toc422368238)

[5.6 限制条件 48](#_Toc422368239)

[5.7 测试计划 48](#_Toc422368240)

[5.8 尚未解决的问题 48](#_Toc422368241)

[5.8.1 客户端尚未解决的问题 48](#_Toc422368242)

[5.8.2 服务器端尚未解决的问题 48](#_Toc422368243)

[5.8.3 难点问题 48](#_Toc422368244)

1. 项目开发计划
   1. 简介
      1. 编写目的

编写本文档的目的主要是为了给剧院订票系统做主要的规划和整合，使软件项目能够在预定成本、进度、质量的前提下顺利完成。明确各时间节点的任务分配各模块人员分配，确保项目开发有序进行，避免重复开发造成损失。此文档供设计人员、开发人员、测试人员阅读。

* + 1. 项目背景

本项目为剧院票务系统，通过建立该系统，扩展了顾客的订票渠道，可以使订票过程变得更加便捷安全。

* + 1. 定义

MYSQL: 系统服务器所使用的数据库软件。

SQL: 结构化查询语言（Structured Query Language）。

事务流：数据进入模块后可能有多种路径进行处理。

主键：又称主码。数据库表中对储存数据对象予以唯一和完整标识的数据列或属性的组合。

外部主键：数据库表中与其他表主键关联的域。

JAVA：主要运用Wform设计和与数据库的数据传递。

系统：若未特别指出，统指本机票预定系统。

SQL: Structured Query Language（结构化查询语言）。

* 1. 项目概述
     1. 工作内容

小组根据项目进度安排，分别对项目进行开发。

需求分析小组通过讨论与调研其他机票订票系统来完成需求分析文档以及数据说明文档。

设计小组首先解决实现该系统需求的程序模块设计问题，完成概要设计文档（包括如何把该系统划分成若干个模块、决定各个模块之间的接口、模块之间传递的信息）。数据结构（数据库设计文档）、模块结构的设计等）然后，根据概要设计文档进行详细设计，完成详细设计文档。

开发小组对软件进行集中开发，完成登录，用户查询，订票，退票等功能。

测试小组对软件进行单元测试，集成测试，审核，评定。

维护小组在软件交付后对使用者进行培训并承担软件后期维护工作。

* + 1. 条件与限制

完成项目应具备的条件：

调研环境

开发平台

组织开发人员（3人）

开发基础设施（JAVA+数据库）

测试维护人员（2人）

* + 1. 运行环境

运行软件环境：

操作系统：Windows

数据库：MYSQL

开发环境：

ECLIPSE、MYSQL

* + 1. 验收标准

系统稳定，运行流畅，业务流程内无严重安全隐患。

出错率：0~1次/ 年

* + 1. 实施计划

|  |  |
| --- | --- |
| 工作内容 | 时间分配 |
| 进行需求分析与调研 | 3天 |
| 概要设计与数据库设计 | 1天 |
| 详细设计 | 1天 |
| 测试与调试 | 1天 |
| 维护 | 未定 |

* + 1. 人员及分工

1. 可行性研究报告
   1. 引言
      1. 编写目的

可行性研究的目的是为了对问题进行研究，以最小的代价在最短的时间内确定问题是否可解。经过对此项目进行详细调查研究，初拟系统实现报告，对软件开发中将要面临的问题及其解决方案进行初步设计及合理安排。明确开发风险及其所带来的经济效益。本报告经审核后，交予审查。

* + 1. 背景

开发软件名称：剧院票务系统。

项目与其他软件，系统的关系：

本项目采用客户机服务器原理，客户端的程序是建立在Windows NT系统上以JAVA为开发软件的应用程序，服务器端采用Linux为操作系统的工作站，是采用MYSQL的为开发软件的数据库服务程序。

* + 1. 定义

Oracle 8: 系统服务器所使用的数据库软件。

SQL: 结构化查询语言（Structured Query Language）。

事务流：数据进入模块后可能有多种路径进行处理。

主键：又称主码。数据库表中对储存数据对象予以唯一和完整标识的数据列或属性的组合。

外部主键：数据库表中与其他表主键关联的域。

系统：若未特别指出，统指本机票预定系统。

SQL: Structured Query Language（结构化查询语言）。

* + 1. 参考资料
  1. 可行性研究的前提
     1. 要求

主要功能：为顾客提供票务预定服务，方便剧院的售票工作，提高剧院的服务质量和服务效率。

性能要求：剧院提供的信息必须及时的反映在平台上。售票系统的定单必须无差错的存储在剧院的主服务器上。对服务器上的数据必须进行及时正确的刷新。

输入要求：数据完整，详实。

输出要求：简捷，快速，实时。

安全与保密要求：服务器的管理员享有对剧场演出信息库及票务信息库和定票信息库的管理与修改。售票员只享有对订票信息库的部分修改（写入与读出）。

* + 1. 目标

系统实现后，大大提高剧院的戏票预定服务效率。降低售票服务中的错误发生率，减少信息 交流的烦琐过程及其带来的开销。

* + 1. 条件、假定和限制

建议软件寿命：5年。

经费来源：

硬件条件：服务器sun工作站，终端为pc机。

运行环境：Linux

数据库：MY SQL

投入运行最迟时间：

* + 1. 进行可行性研究的方法

同时满足以下三个方面的条件：成本/效益分析结果，效益 > 成本。 技术可行，现有技术可完全承担开发任务。操作可行，软件能被原有工作人员快速接受。

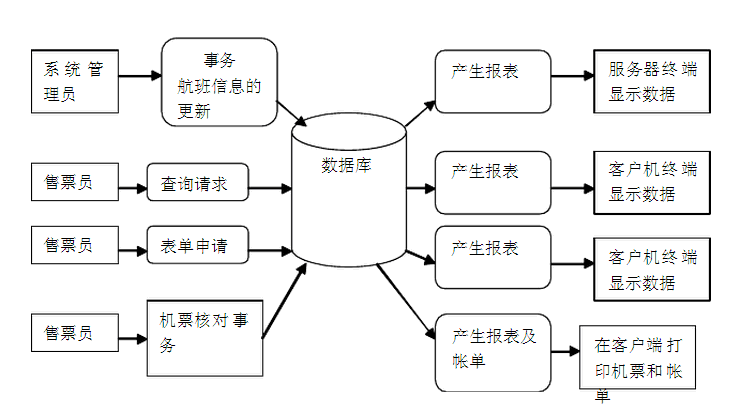
* 1. 所建议的系统

* + 1. 对所建议系统的说明

在剧院的终端是安装了WindowsNT的PC机，主要目的是向订票平台的服务器传递数据。当顾客在旅游局进行咨询时，终端向服务器发出查询请求，服务器根据剧院信息库的实时数据，向终端发送数据，显示在终端的屏幕上。当顾客向售票员定票时，终端向服务器发出详尽的一份定单，服务器核对后，存入定票信息库，并修改票务信息库。当顾客再次来取票时，终端向服务器发出查询定票请求，服务器接收后，查询定票信息库，核对后，传送戏票确认表单，终端打印出戏票。

* + 1. 处理流程和数据流程

（见下一页）



在客户端打印戏票和账单

戏票核对信息

事务演出信息的更新

* 1. 投资及效益分析
     1. 支出
        1. 基本建设投资
        2. 其他一次性支出
        3. 非一次性支出

人工费用：6(月) \* 20(人) \* 5000(元) = 60万

其他不可知额外支出：20万

共计：80万

* + 1. 收益
       1. 一次性收益

0元。

* + - 1. 非一次性收益

按银行利率1%计算：

减少员工20人 (1000圆/人)

五年收益：1000 \* (1.1 + 1.1\*2 + 1.1\*3 + 1.1\*4 + 1.1\*5) \* 20 \* 12 \* 5 = 120万

工作效率提高收益（工作效率提高30%）: 30 \* (1.1 + 1.1\*2 + 1.1\*3 + 1.1\*4 + 1.1\*5) \* (30%) \* 5 = 45万

经常性收益共计:160万

* + - 1. 不可定量的收益

因服务质量提高增加客量10%

1000万 \* 10% \* (90% + (90%) \* 2 + (90%) \* 3 + (90%) \* 4 + (90%) \* 5) = 360万

* + 1. 收益/投资比

520万 / 151万 = 344%

* + 1. 投资回收周期

2.3年．

* 1. 社会因素方面的可能性
     1. 法律方面的可行性

所有软件都选用正版；所有技术资料都由提出方保管；合同制定确定违约责任。

* + 1. 使用方面的可行性

使用本软件人员要求有一定计算机基础的人员，系统管理员要求由计算机的专业知识，所有人员都要经过本公司培训。管理人员也需经一般培训，经过培训人员将会熟练使用本软件。两名系统管理员，一名审计员将进行专业培训，他们将熟练管理本系统。

* + 1. 结论

由于投资效益比远大于100%,技术、经济、操作都有可行性，可以进行开发。

1. 需求分析报告
   1. 引言
      1. 编写的目的

为了巩固数据库第一阶段的学习，模拟一个剧院订票系统来实现剧院的戏票销售的自动化的计算机系统，为企业的决策层提供准确、精细、迅速的票务销售信息。

* + 1. 背景

开发软件名称：剧院预订系统。

项目组长：张力天

用户：所有客户

运行环境：win8/win7/xp/vista

开发平台：Eclipse、 MYSQL数据库。

系统架构：该系统采用java完成，用AWT开发系统界面设计。

* + 1. 定义

MYSQL: 系统服务器所使用的数据库管理系统（DBMS）

SQL: 一种用于访问查询数据库的语言

事务流：数据进入模块后可能有多种路径进行处理

* + 1. 参考文献
  1. 任务概述
     1. 目标

《网上戏票预订系统》本是在Internet环境下运行的，但根据课程需求在此我们先将它做为一个c/s程序。该项软件开发的意图是为了方便剧院进行顾客预定票的管理，减少管理中出现的麻烦，它主要在某一航空公司内部进行使用，再加之这是一项独立的软件，全部内容自含，所以不会涉及到与其它系统、产品的联系和接口问题。

* + 1. 用户特点

管理员(维护人员)：熟练掌握Java语言。熟悉掌握 sql语句的使用。

普通用户：能够熟练地使用桌面程序，有一定的电脑基础。

* + 1. 假定和约束

普通管理员，只能对库（演出库和客户库）中的信息进行查询操作；超级管理员，可以根据具体需要进行适当的数据管理（增、删、改、更）。

客户只能对航班信息库中的内容进行查询操作，客户进入到页面之后在不进行登录的情况下只能进行演出信息查询操作，要预订戏票就必须要先注册登录提交自己的基本信息；系统会根据管理员和客户的各种操作做出相应的返回信息进行提示。

* 1. 需求规定
     1. 系统功能需求

本系统用于远程戏票预订，包括远程演出信息查询、戏票预订与确认等；主要分为四大功能：查询、订票、退票和管理。

管理员登录、注销 到系统并进行插入、删除、更新以及查看戏票后台数据库操作

插入：戏票的插入可以按照演出号、演出时间、座位号等等插入数据库。

删除：戏票可以按照演出号、演出时间进行删除

* + 1. 客户端系统功能
       1. 普通用户

查询：根据演出号、剧院号查询出票类信息

订票： 根据观看日期和演出名预订，一份订单可订多张票 。

退票：要填写订单号、身份证、客户姓名以及戏票信息等等。

后台管理系统：

演出动态：可以根据演出、剧院人员查看这个演出的最新动态

* + - 1. 剧院人员

查询：根据演出号、剧院号查询出票类信息

订票： 根据观看日期和演出名预订票，一份订单可订多张票 。

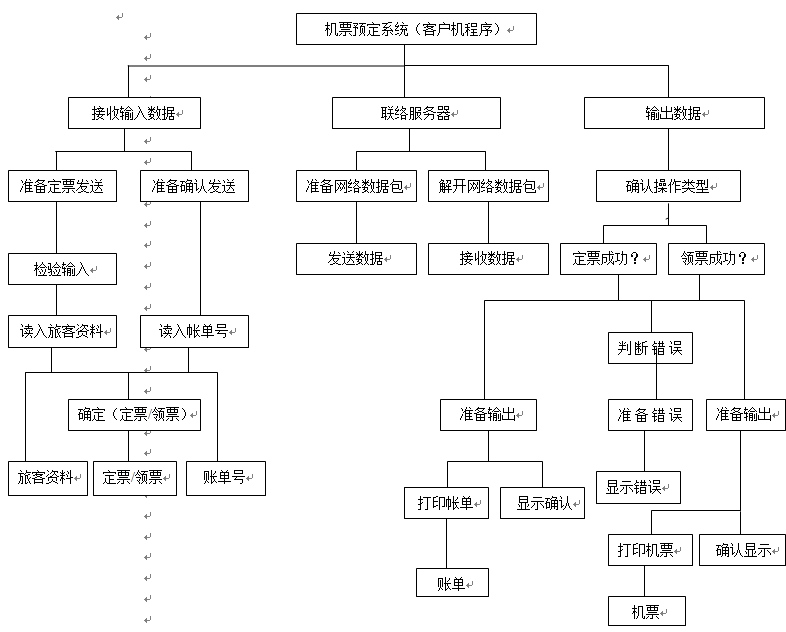
退票：要填写订单号、身份证、客户姓名以及戏票信息等等。

后台管理系统：

销售统计：根据年份、月份统计总共买出的票数

后台管理：管理员可以根据总部要求去创建演出、取消演出、演出查询

下面以结构图来描述戏票预定系统的软件总体结构（见下一页）



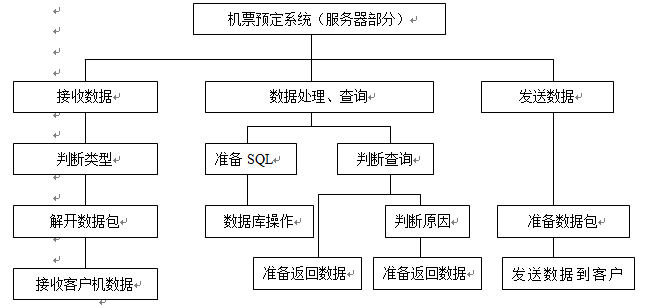
戏票预订系统

* + 1. 服务器端系统功能

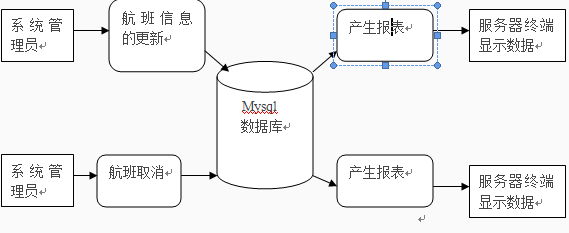
查询：当有顾客查询机票时，接收其数据信息，并能返回查询结果

统计：对售出的戏票作出统计

更新信息：当客户端发来请求信息，服务器进行接收。



**戏票预订系统（服务器）**



演出取消

演出信息的更新

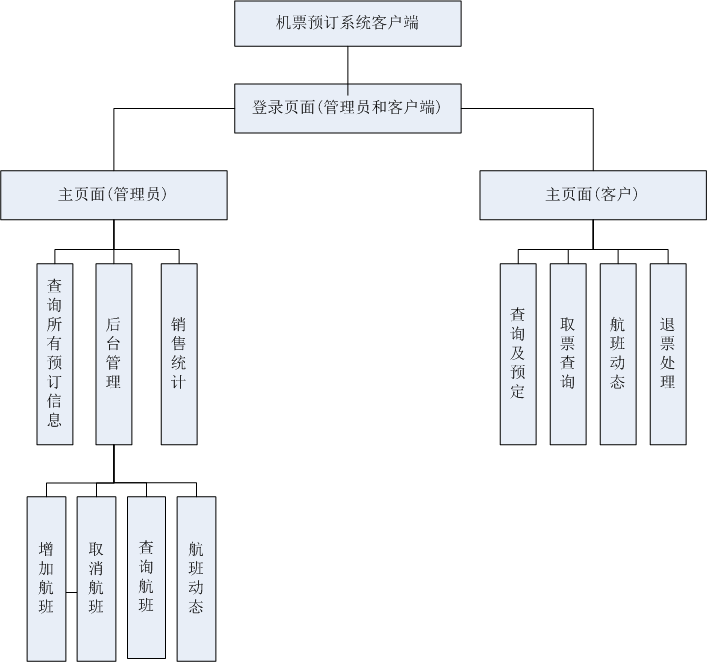
信息更新：系统的管理员通过对演出信息的修改，使数据库中的信息发生变化。

演出取消：当由于某种原因某演出被取消了，系统的管理人员要第一时间的更改数据库中的信息，使顾客能及时了解最新信息。

* 1. 数据字典
     1. 系统中的数据元素：

|  |
| --- |
| 名字：顾客信息  别名：  描述：顾客的个人信息，用于对顾客的确认  定义：顾客信息=姓名+性别+工作单位+身份证号码+演出时间+座位号 |
| 名字：演出信息  别名：  描述：演出的地点及起止时间等，让顾客能够查看和确认  定义：演出信息=编号+地点+起止时间 |
| 名字：戏票信息  别名：  描述：顾客所观看的演出时间，用于确定顾客的座位  定义：订票信息=顾客观看时间+顾客座位 |
| 名字：戏票订单  别名：  描述：顾客交付费用的凭证  定义：戏票账单=顾客姓名+交付费+时间 |
| 名字：取票通知  别名：  描述：顾客领取票的凭证  定义：取票通知=顾客姓名+领票时间 |
| 名字：打印票信息  别名：  描述：顾客的演出戏票的信息  定义：打印机票信息=顾客的演出戏票 |
| 名字：剧院数据库信息  别名：  描述：已订票的顾客在剧院的记录和演出信息等等  定义：剧院数据库信息=演出戏票信息+订票顾客的信息等等 |

* 1. 项目模块划分



演出动态

演出动态

增加演出

取消演出

查询演出

戏票预定系统客户端

* 1. 运行环境规定
     1. 硬件要求：

服务器：CPU133M，64M或以上内存，300M硬盘空间；VGA显示器或更高

客户机：最低CUP133M，16MB内存；最少8MB的RAM存储器；VGA显示器或更高

* + 1. 软件要求

服务器：Windows 2000 Server中文版或更高；oracle 8i或更高

客户机：Windows 95以上或以上；该软件运行于Windows系统，使用java语言开发

* 1. 测试

在测试过程中，首先需要对各子单元过程进行测试。在各子单元过程测试完毕后，再对各模块（包括各子单元过程之间的接口）进行测试，处理好各模块之间的接口，最后对系统进行测试和维护。

主要通过测试部分，发现开发人员忽视的部分，通过测试人员和编码人员的沟通和协作，可以不断的改进系统，不断提高系统的性能以及安全性。

* 1. 需求小结

我们小组分工合作、齐心协力，一起完成了课程设计前的准备工作（阅读课程设计相关文档）、小组讨论分工、完成系统开发的各个文档、课程设计总结报告、，个人小结的任务。在课程设计中我们便对这次任务进行了规划和分工。我们组的成员一起努力，查阅资料、小组讨论、对资料进行分析，并在这段时间里完成了整个设计

从这次的课程设计中，通过对需求分析说明书的报告很好的学习了软件工程的知识，对以后书写和学习课程设计留下了很好的经验。需求分析的书写要很正式，要根据实验指导书的模式来写，从中要写出对目标概述，对功能的规定，对性能的规定，运行环境的规定等等。

1. 概要设计说明书
   1. 引言
      1. 编写目的

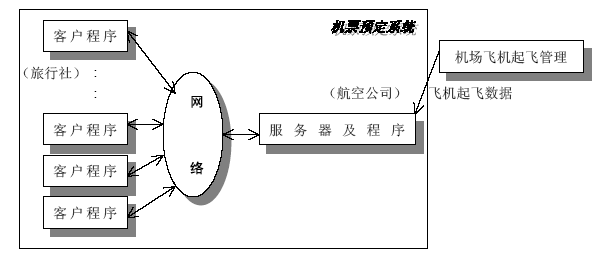
在本票务预定系统项目的前一阶段，也就是需求分析阶段中，已经将系统用户对本系统的需求做了详细的阐述，这些用户需求已经在上一阶段中对演出信息、剧场的实地调研中获得，并在需求规格说明书中得到详尽得叙述及阐明。

本阶段已在系统的需求分析的基础上，对戏票预定系统做概要设计。主要解决了实现该系统需求的程序模块设计问题。包括如何把该系统划分成若干个模块、决定各个模块之间的接口、模块之间传递的信息，以及数据结构、模块结构的设计等。在以下的概要设计报告中将对在本阶段中对系统所做的所有概要设计进行详细的说明。

在下一阶段的详细设计中，程序设计员可参考此概要设计报告，在概要设计对戏票预定系统所做的模块结构设计的基础上，对系统进行详细设计。在以后的软件测试以及软件维护阶段也可参考此说明书，以便于了解在概要设计过程中所完成的各模块设计结构，或在修改时找出在本阶段设计的不足或错误。

* + 1. 项目背景

机票预定系统将由两部分组成：置于个定票点的前台客户程序，以及置于剧院的数据库服务器。本系统与其他系统的关系如下：



预订系统

演出数据

剧院演出管理

* + 1. 定义
       1. 专门术语

MYSQL: 系统服务器所使用的数据库管理系统（DBMS）。

SQL: 一种用于访问查询数据库的语言

事务流：数据进入模块后可能有多种路径进行处理。

主键：数据库表中的关键域。值互不相同。

外部主键：数据库表中与其他表主键关联的域。

ROLLBACK: 数据库的错误恢复机制。

* + - 1. 缩写

系统：若未特别指出，统指本机票预定系统。

SQL: Structured Query Language（结构化查询语言）。

ATM: Asynchronous Transfer Mode (异步传输模式)。

* + 1. 参考资料
  1. 任务概述
     1. 目标
        1. 运行环境

系统将由两部分程序组成，安装在各客户机上的客户程序及剧场内的数据服务器程序。

根据调研得知所有旅行社的计算机配置均在Pentium 133级别以上，客户程序应能够在Pentium 133级别以上, Win NT环境下运行。

* + - 1. 需求概述

顾客把预定戏票的顾客信息，包括姓名、性别、工作单位、身份证号码、演出时间、地点，输入票预定系统的客户端程序，系统经过查询剧院内的航班数据服务器后，为顾客安排座位，印出取票通知。顾客在飞机起飞前一天凭取票通知和帐单交款后取票，系统校对无误后即印出票给顾客。

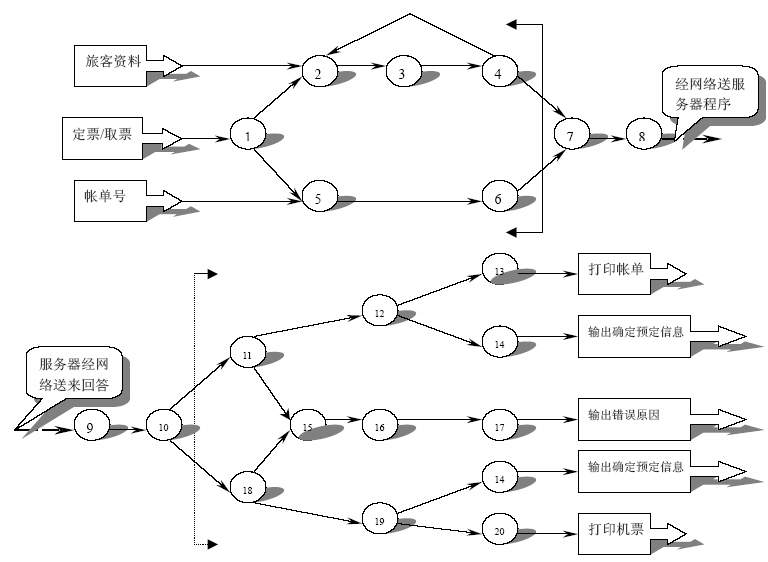
要求系统能有效、快速、安全、可靠和无误的完成上述操作。并要求客户机的界面要简单明了，易于操作，服务器程序利于维护。

* 1. 总体设计
     1. 处理流程

下面将使用（结构化设计）面向数据流的方法对戏票预定系统的处理流程进行分析。系统可分为两大部分：一、客户机上的程序，二、服务器上的程序。以下将分别对系统的这两大部分进行流程分析：

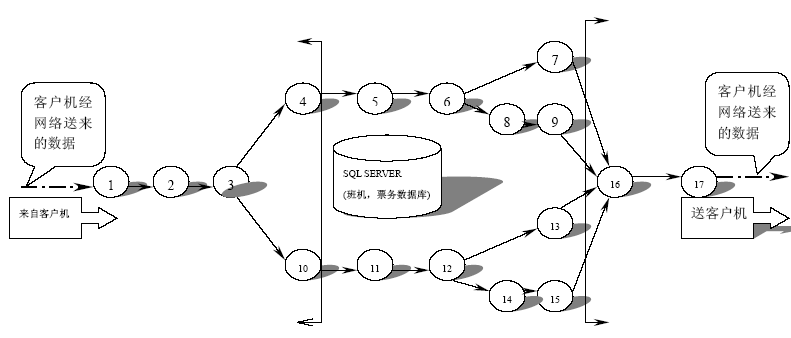
* + - 1. 客户机程序流程

客户机上的输入信息为顾客资料或帐单号，还包括一个定票/领票选项。输出为打印帐单或戏票，和确认或出错信息。其内部处理流程如下：



下面对各模块（以数字表示）进行功能说明：

1. 读入并确定（定票/领票） 它是一个事务处理中心，若输入是定票时转入2 ，输出则转入5。
2. 读入旅客资料 读取所输入的顾客资料，并将其格式化，将格式化数据送3。
3. 检验输入资料 检验格式化输入资料，初步检查输入是否有误，它是一个事务处理中心，有误则转2再次读入，检查无误则将数据送4。
4. 准备服务器数据 由格式化旅客资料准备需发送到服务器的数据，将准备好的数据送7。
5. 读入帐单号 读取所输入的帐单号，并将其格式化，将格式化数据送6。
6. 准备服务器数据 由格式化帐单号准备需发送到服务器的数据，不同于4，它将准备的是帐单号，将准备好的数据送7。
7. 准备网络数据包 将数据准备/加密成网络传输数据包。
8. 网络数据发送 将数据包经网络发送到服务器程序， 它是程序与网络的接口。
9. 网络数据接受 接受由服务器程序经网络传送到客户机的数据包， 它是程序与网络的接口。
10. 解开/判断 解开9 所接受的数据包，并判断数据是定票数据或是领票数据，它是一事务处理中心，是定票则将数据送11，是领票则将数据送18。
11. 定票成功？ 由数据判断定票是否成功，它是一事务处理中心，若定票成功则将数据送11，不成功则将数据送18。
12. 准备输出数据 由接收的服务器数据准备输出数据，其中关于帐单的数据将送入13，其他包括顾客资料等送入14。
13. 打印帐单 由帐单信息处理帐单格式并打印出帐单。
14. 输出定位确定 在屏幕上显示确定定位及其他信息。
15. 判断错误 由数据判断出错误类型，将类型送到16。
16. 准备错误语句 由具体错误类型准备错误语句，送17。
17. 输出错误语句 在屏幕上显示错误语句。
18. 领票核对成功？ 由数据判断领票核对是否成功，它是一事务处理中心，若成功则将数据送19，不成功则将数据送15。
19. 准备票输出数据 由接收的服务器数据准备将打印的票数据数，据其中关于帐单的票的数据将送入20，其他包括顾客资料等送入14。
20. 打印机票 由票信息处理机票格式并打印出票。
    * + 1. 服务器程序流程



下面对各模块（以数字表示）进行功能说明：

1. 网络数据接收 接受由客户机程序经网络传送到客户机的数据包，它是程序与网络的接口。
2. 解开数据 解开/解密打包的数据，并将恢复的数据格式化送入3。
3. 判断操作 由数据判断需要做什么数据库操作，它是一事务处理中心，若为定票操作则将数据送4，若为领票操作则将数据送15。
4. 准备数据库查询语句 由定票数据准备访问MYSQL数据库所需的SQL 语句，送入5。
5. 数据库操作 运行SQL 语句，查询并存入数据库，获得相应的定票数据，将数据送入6。它需包括初步对查询的判断。
6. 判断查询是否成功 利用查询所得的数据判断定票是否成功，它是一事务处理中心，若为定票操作成功则将数据送7，若不成功则将数据送8。
7. 准备回送数据 准备定票成功后需回送客户机程序的数据，送16。
8. 确定原因 由查询数据判断定票不成功的原因，送9。
9. 准备回送数据 根据不成功的原因，准备需送回客户机程序的数据，送入16。
10. 准备数据库查询语句 由领票数据准备访问MYSQL数据库所需的SQL语句，送入11。
11. 数据库操作 运行SQL 语句，查询并存入数据库获得相应的领票数据，将数据送入6。它需包括初步对查询的判断。
12. 判断查询是否成功 利用查询所得的数据判断领票是否成功，它是一事务处理中心，若为领票操作得到确认则将数据送11，若不成功则将数据送8。
13. 准备回送数据 准备领票确认后需回送客户机程序的数据，送16。
14. 确定原因 由查询数据判断领票不能确认的原因，送15。
15. 准备回送数据 根据不成功的原因，准备需送回客户机程序的数据，送入16。
16. 准备网络数据包 将数据准备/加密成网络传输数据包。
17. 网络数据发送 将数据包经网络发送到服务器程序，它是程序与网络的接口。

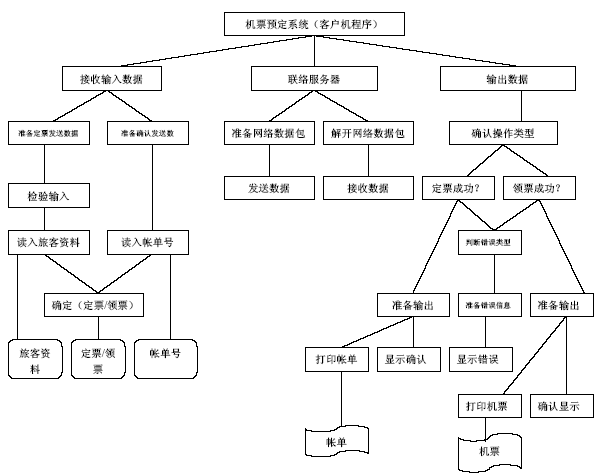
以上显示的各模块数字相同的为同一模块；由于事务流处理上的便利，将有些功能模块，拆分到各事务流中。

* + 1. 总体结构和模块外部设计

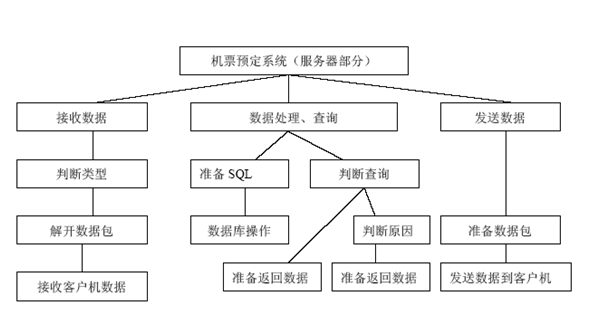
下面以结构图来描述票预定系统的软件总体结构。框内注明了模块的名字；方框之间的直线表示模块的调用关系。

* + - 1. 客户机部分

（见下一页）



* + - 1. 服务器部分



其中在数据处理、查询下作了省略…只给出了一部分模块，实际上是有两部分，分别对应于定位及确认。

* + 1. 功能分配

各项模块的功能可参照3.1 中的说明。客户机程序主要有三大块：接收数据、网络通信及输出部分。服务器程序主要也是由三大功能：接收网络数据、数据库操作及发送网络数据部分。服务器程序需与已建立的MYSQL 数据库互连，其接口将于下面部分阐述。

* 1. 接口设计
     1. 外部接口
        1. 用户界面

在用户界面部分，根据需求分析的结果，用户需要一个用户友善界面。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WINDOW 风格较熟悉，应尽量向这一方向靠拢。在设计语言上，已决定使用JAVA进行编程，在界面上可使用SWING所提供的可视化组件，向WINDOWS风格靠近。 其中服务器程序界面要做到操作简单，易于管理。在设计上采用下拉式菜单方式，在出错显示上可调用VISUAL C++ 库中的错误提示函数。

总的来说，系统的用户界面应作到可靠性、简单性、易学习和使用。

* + - 1. 软件接口

服务器程序可使用JAVA提供的对MYSQL的接口，进行对数据库的所有访问。

服务器程序上可使用MYSQL的对数据库的备分命令，以做到对数据的保存。

在网络软件接口方面，使用一种无差错的传输协议，采用滑动窗口方式对数据进行网络传输及接收。

* + - 1. 硬件接口

在输入方面，对于键盘、鼠标的输入，可用JAVA的标准输入/输出，对输入进行处理。

在输出方面，打印机的连接及使用，也可用JAVA的标准输入/输出对其进行处理。在网络传输部分，在网络硬件部分，为了实现高速传输，将使用高速ATM。

* + 1. 内部接口

内部接口方面，各模块之间采用函数调用、参数传递、返回值的方式进行信息传递。具体参数的结构将在下面数据结构设计的内容中说明。接口传递的信息将是以数据结构封装了的数据，以参数传递或返回值的形式在各模块间传输。

* 1. 数据库设计
     1. 数据表的设计
        1. 1.用户表（users）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 记录内容 | 是否为空 | 备注 |
| u\_id | Varchar2(20) | 用户名 | 0 | 主键 |
| u\_password | Varchar2(20) | 密码 | 0 |  |
| u\_name | Varchar2(20) | 用户姓名 | 1 |  |
| u\_telephone | Varchar2(20) | 手机号码 | 1 |  |
| u\_address | Varchar2(100) | 用户地址 | 1 |  |
| u\_email | Varchar2(20) | Email | 1 |  |
| u\_idcard | Varchar2(20) | 身份证 | 1 |  |
| u\_power | number | 权限 | 1 | 剧院人员：1顾客：0 |

* + - 1. 登录表（login）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 记录内容 | 是否为空 | 备注 |
| l\_uid | Varchar2(20) | 用户名 | 0 |  |
| l\_password | Varchar2(20) | 密码 | 0 |  |

* + - 1. 订单表（order）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 记录内容 | 是否为空 | 备注 |
| o\_id | Vachar2（100） | 订单编号 | 0 | 自增长 |
| o\_uid | Varchar2(20) | 用户名 | 0 |  |
| o\_tid | Varchar2(40) | 票编号 | 0 |  |
| o\_time | Varchar2(40) | 下单时间 | 0 |  |

* + - 1. 票务信息（tickets）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 记录内容 | 是否为空 | 备注 |
| t\_id | Vachar2（100） | 票编号 | 0 | 主键 |
| t\_f\_id | number | 演出id 号 | 0 |  |
| t\_uid | Varchar2(20) | 用户Id | 0 |  |
| t\_seatId | Number | 座位号 | 0 |  |
| t\_date | Varchar2(20) | 该票的日期 | 0 | 这是一个出发的日期 |
| t\_type | Varchar2（30） | 座位类型 | 1 | 这是一个座位的类型 |
| t\_price | Number | 票价 | 1 | 这个舱位的价格 |

* + - 1. 演出信息表（flight\_info）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 记录内容 | 是否为空 | 备注 |
| f\_id | number | 演出id 号 | 0 | 航班起飞的ID号(主键) |
| f\_number | Vachar2(30) | 剧院号 | 0 |  |
| f\_start\_time | Varchar2(40) | 开始时间 | 0 |  |
| f\_end\_time | Varchar2(40) | 结束时间 | 0 |  |
| f\_departure | Varchar2(40) | 地点 | 0 |  |
|  |  |  |  |  |
|  |  |  |  |  |

* + - 1. 7.航空仓类型——座位表(seat)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 记录内容 | 是否为空 | 备注 |
| s\_id | Number | 座位唯一Id | 0 | 标识 能够以下得到信息 |
| f\_id | number | 演出id | 0 | 演出id(外键) |
| s\_type | Varchar2（30） | 座位类型 | 0 | 座位类型 |
| s\_number | Number | 座位数量 | 0 | 座位数量 |
| s\_price | Number | 座位价格 | 0 | 座位价格 |

* + - 1. 8.用户操作记录表（deal\_log）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 记录内容 | 是否为空 | 备注 |
| d\_number | number | 用户编号 | 0 | 主键，自增长 |
| d\_uid | Varchar2(30) | 用户名 | 0 |  |
| d\_opertype | Varchar2(40) | 操作类型 | 0 | 订票，退票，改签 |
| d\_tid | Varchar2(40) | 票编号 | 0 |  |
| d\_time | Varchar2(10) | 时间 | 0 | 操作的时间 |

* + - 1. 9.航班动态表（state）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 记录内容 | 是否为空 | 备注 |
| fs\_number | Varchar2(30) | 演出号 | 0 | 演出号 |
| fs\_date | Varchar2(10) | 演出日期 | 0 | 演出 |
| fs\_state | Varchar2(300) | 演出动态 | 0 | 演出动态 |

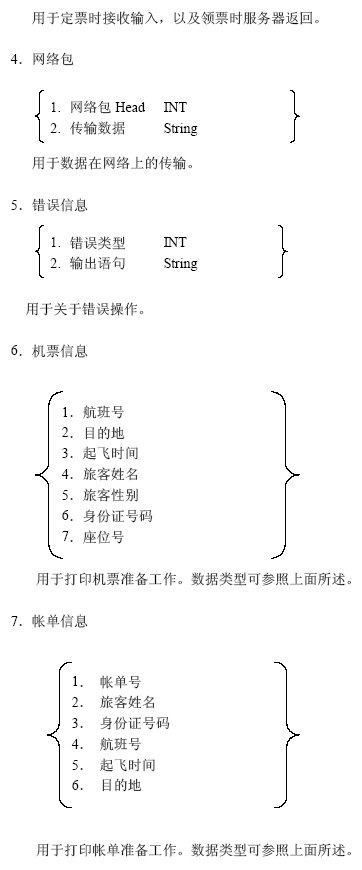
* + 1. 物理数据结构设计

物理数据结构设计主要是设计数据在模块中的表示形式。数据在模块中都是以结构的方式表示。

（见下一页）



（见下一页）



* + 1. 数据结构与程序的关系

服务器程序在对定票/领票进行操作时需对数据库数据库数据结构，也就是数据表进行查询和修改：在定票/领票过程中都需要对数据库中的所有表，进行联合查询、修改。

物理数据结构主要用于各模块之间函数的信息传递。接口传递的信息将是以数据结构封装了的数据，以参数传递或返回值的形式在各模块间传输。出错信息将送入显示模块中，机票结构，帐单结构，送入打印准备模块中准备打印格式。

* 1. 运行设计
     1. 运行模块的组合

客户机程序在有输入时启动接收数据模块，通过各模块之间的调用，读入并对输入进行格式化。在接收数据模块得到充分的数据时，将调用网络传输模块，将数据通过网络送到服务器，并等待接收服务器返回的信息。接收到返回信息后随即调用数据输出模块，对信息进行处理，产生相应的输出。

服务器程序的接收网络数据模块必须始终处于活动状态。接收到数据后，调用数据处理/查询模块对数据库进行访问，完成后调用网络发送模块，将信息返回客户机。

* + 1. 运行控制

运行控制将严格按照各模块间函数调用关系来实现。在各事务中心模块中，需对运行控制进行正确的判断，选择正确的运行控制路径。

在网络传方面，客户机在发送数据后，将等待服务器的确认收到信号，收到后，再次等待服务器发送回答数据，然后对数据进行确认。服务器在接到数据后发送确认信号，在对数据处理、访问数据库后，将返回信息送回客户机，并等待确认。

* + 1. 运行时间

在软体的需求分析中，对运行时间的要求为必须对作出的操作有较快的反应。网络硬件对运行时间有最大的影响，当网络负载量大时，对操作反应将受到很大的影响。所以将采用高速ATM 网络，实现客户机与服务器之间的连接，以减少网络传输上的开销。其次是服务器的性能，这将影响对数据库访问时间即操作时间的长短，影响加大客户机操作的等待时间，所以必须使用高性能的服务器，建议使用 Pentium III 处理器。硬件对本系统的速度影响将会大于软件的影响。

* 1. 出错处理设计
     1. 出错输出信息

程序在运行时主要会出现两种错误：

1. 由于输入信息，或无法满足要求时产生的错误，称为软错误。
2. 由于其他问题，如网络传输超时等，产生的问题，称为硬错误。

对于软错误，须在定票/领票操作成功判断及输入数据验证模块由数据进行数据分析，判断错误类型，再生成相应的错误提示语句，送到输出模块中。对与硬错误，可在出错的相应模块中输出简单的出错语句，并将程序重置。返回输入阶段。

出错信息必须给出相应的出错原因，例：

“全部座位已满！” 、“查无此演出信息！”等。

* + 1. 出错处理对策

所有的客户机及服务器都必须安装不间断电源以防止停电或电压不稳造成的数据丢失的损失。若真断电时，客户机上将不会有太大的影响，主要是服务器上：在断电后恢复过程可采用 MYSQL 的日志文件，对其进行ROLLBACK 处理，对数据进行恢复。

在网络传输方面，可考虑建立一条成本较低的后备网络，以保证当主网络断路时数据的通信。

在硬件方面要选择较可靠、稳定的服务器机种，保证系统运行时的可靠性。

* 1. 安全保密设计

由于数据的传输上需要通过网络传输，为了客户资料进行保密，需要在网络的传输过程中对数据进行加密。这个工作主要是在准备网络包，及解开网络包这两个模块完成，它们各对数据进行加密及解密还原工作。在加密算法选择上将使用RSA 加密算法。具体算法可参照参考资料中《Computer Network》p.598。

* 1. 维护设计

维护方面主要为对服务器上的数据库数据进行维护。可使用 MYSQL的数据库维护功能机制。例如，定期为数据库进行Backup，维护管理数据库死锁问题和维护数据库内数据的一致性等。

1. 详细设计说明书
   1. 引言
      1. 编写目的

在前一阶段（概要设计说明书）中，已解决了实现该系统需求的程序类设计问题。包括如何把该系统划分成若干个类、决定各类之间的接口、类之间传递的信息，以及数据结构、类结构的设计等。在以下的详细设计报告中将对在本阶段中对系统所做的所有详细设计进行说明。

在本阶段中，确定应该如何具体地实现所要求的系统，从而在编码阶段可以

这个描述直接翻译成用具体的java程序语言书写的程序。主要的工作有：根据在《需求分析说明书》中所描述的数据、功能、运行、性能需求，并依照《概要设计说明书》所确定的处理流程、总体结构、数据结构设计和类设计，逐个类的进行尽可能详尽的程序描述（包括各方法的功能、性能、输入、输出、算法、程序逻辑、接口等等），尽可能的减少编码时间。

* + 1. 背景

软件系统的名称:戏票预订系统

项目分两个部分：

面向客户的客户终端程序

部署在服务器上的Java服务程序

* + 1. 定义

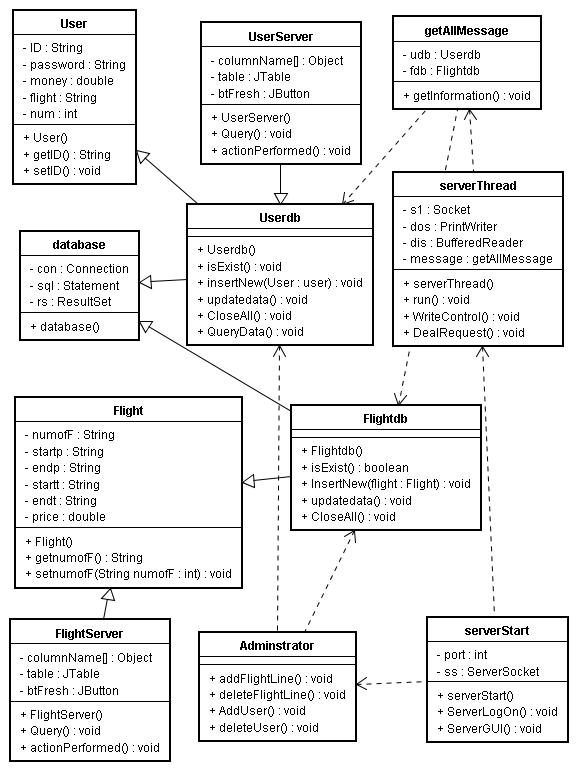
Socket：网络通信模式。

Server：机票预订系统的服务器端。

Client：机票预定系统的客户端

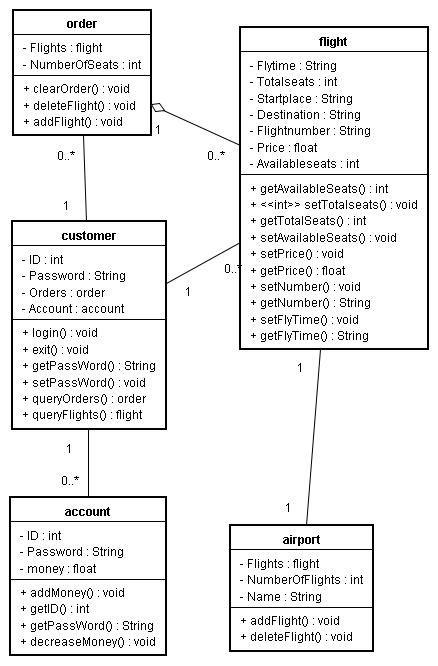
* + 1. 参考资料
  1. 程序系统的结构
     1. Server类图：

（见下一页）



* + 1. Cilent类图：

（见下一页）



* 1. Server端程序说明
     1. ServerStart类设计说明
        1. ServerStatrt()方法

调用窗口显示方法ServerGUI()，显示服务器主界面，Server be ready。

**功能：**

启动服务器。

**测试计划：**

能顺利启动服务器

* + - 1. ServerLogon()方法

用于服务器管理员登录。

**功能：**

直接访问数据库，验证输入的用户名密码，如正确则进入服务器管理界面。

**输入项：**

用户名 String

密码 String

**输出项：**

是否正确 bool

**算法：**

1. 用getText()从管理员登陆界面读取用户名与密码。
2. 将信息传递到方法中到数据库中查找。
3. 返回查找情况，找到true，未找到false。

**测试计划：**

能正确识别输入的用户名与密码。输入正确的用户名与密码可跳转至服务器管理员界面，输入错误的则发出提示，不允许进入。

* + - 1. ServerGUI()方法

用于编辑管理员界面并显示。

**功能：**

编辑管理员界面，传递参数给ServerLongON()，并显示界面。

**算法：**

使用控件的监听事件达到上述效果

**测试计划：**

界面能否正确显示，SeverLogON()方法能否正常运行。

* + 1. database类设计说明
       1. database()方法

用于建立与数据库的链接，可被继承。

**功能：**

建立与数据库的链接，可被继承，使其他类在访问数据库时不必再次建立连接。

**性能：**

为提升数据传输的效率，访问数据库时间应尽量短。

* + 1. Userdb类设计说明
       1. Userdb()方法

Userdb的构造方法。

**功能：**

继承父类database()的构造方法，从而建立起与数据库的链接。

* + - 1. isExit()方法

判断数据是否出现，可被其他方法调用。

**功能：**

在查询函数中可被调用，用于判断所查询的数据是否出现。

**输出项：**

输出bool变量，若存在，则为true，or 则为false

* + - 1. updatedata()方法

更新当前行数据

**功能：**

更新当前行数据为修改过的数据，修改数据库内内容。

**输入项：**

要修改数据所在的列名 String

修改后的数据 Object

**输出项：**

数据内的内容被修改。

* + - 1. InsertNew(User user)

向数据库加入一行数据

**功能：**

读入User数据，将这行数据加入数据库中

**输入项：**

用户信息 User

**输出项：**

用户信息加入数据库

* + - 1. QueryData()

查询数据是否存在。

**功能：**

接收数据，到数据库中查询数据数据是否存在。

**输入项：**

用户名String

密码String

余额 double

航班号String

票数int

查询种类int

查询种类必须输入，其他可根据需要选择性的输入。

**输出项：**

* + - 1. CloseAll()

关闭链接。

**功能：**

关闭服务器与数据库之间的链接

* + 1. Flightdb类设计说明
       1. Flightdb()方法

Flightdb的构造方法。

**功能：**

继承父类database()的构造方法，从而建立起与数据库的链接。

* + - 1. isExit()方法

判断数据是否出现，可被其他方法调用。

**功能：**

在查询函数中可被调用，用于判断所查询的数据是否出现。

**输出项：**

输出bool变量，若存在，则为true，or 则为false

* + - 1. updatedata()方法

更新当前行数据

**功能：**

更新当前行数据为修改过的数据，修改数据库内内容。

**输入项：**

要修改数据所在的列名 String

修改后的数据 Object

**输出项：**

数据内的内容被修改。

* + - 1. InsertNew(Flight flight)

向数据库加入一行数据

**功能：**

读入flight数据，将这行数据加入数据库中

**输入项：**

航班信息 Flight

**输出项：**

航班信息加入数据库

* + - 1. QueryData()

查询数据是否存在。

**功能：**

接收数据，到数据库中查询数据数据是否存在。

**输入项：**

航班号 String

起飞地点String

到达地点String

起飞时间String

到达时间String

票价 double

查询类型int

查询种类必须输入，其他可根据需要选择性的输入。

**输出项：**

查询得到的信息加入游标数组。

* + - 1. CloseAll()

关闭链接。

**功能：**

关闭服务器与数据库之间的链接

* + 1. UserServer类设计说明
       1. UserServer ()方法

UserServer的构造方法。

功能：

继承JFrame，建立界面用于把存放用户信息的数据库中的信息显示出来，便于管理员查看管理

* + - 1. Query()方法

遍历数据库中信息。

功能：

点击刷新时，遍历User数据库，将数据信息重新显示。

算法：

1. 建立足够大的存储空间
2. 与User数据库建立连接
3. 遍历数据库，把数据加入存储空间
4. 数据加入显示界面，显示出来。
   * 1. FlightServer类设计说明
        1. FlightServer ()方法

FlightServer的构造方法。

功能：

继承JFrame，建立界面用于把存放航班信息的数据库中的信息显示出来，便于管理员查看管理。

* + - 1. Query()方法

遍历数据库中信息。

功能：

点击刷新时，遍历Flight数据库，将数据信息重新显示。

算法：

1. 建立足够大的存储空间
2. 与Flight数据库建立连接
3. 遍历数据库，把数据加入存储空间
4. 数据加入显示界面，显示出来。
   * 1. User类设计说明
        1. User()方法

User的构造方法，建立方便调用用户数据的数据结构。

* + - 1. getID()方法

得到用户名。

功能：

调用时，可返回用户名

输出项：

用户名 String

* + - 1. setID()方法

修改用户名。

功能:

调用时，可修改用户名为指定值

输入项:

用户名 String

* + - 1. 其他类似方法

getpassword(),setpassword(),getmoney(),setmoney,getflight(),setflight,getnum(),setflight()

* + 1. Flight类设计说明
       1. Flight()方法

Flight类的构造方法，建立方便调用航班数据的数据结构。

* + - 1. getnumofF()方法

得到航班号。

功能：

调用时，可返回航班号

输出项：

航班号 String

* + - 1. setnumofF()方法

修改航班号。

功能：

调用时，可修改航班号为指定值

输入项：

航班号 String

* + - 1. 其他类似方法

getnumofF(),getstartp(),setstartp(),getendp(),setendp(),getstartt(),setstartt(),getendt(),setendt(),getprice(),setprice(),getnum(),setnum()

* + 1. Administrator类设计说明
       1. addUserData()方法

添加用户信息。

功能：

管理员可以通过这个方法跳过注册直接添加用户。

输入项：

用户名String

密码String

余额 double

输出项：

用户信息被加到User数据库中

测试计划：

建立服务器与数据库的链接，使需要添加的用户信息如预料添加。

* + - 1. delateUserData()方法

删除用户信息。

功能：

管理员可以通过这个方法删除用户。

输出项：

用户信息被从User数据库中删除

测试计划：

建立服务器与数据库的链接，使需要删除的用户信息如预料删除。

* + - 1. addFlightLine ()方法

添加新航线。

功能：

管理员可以通过这个方法添加新航线。

输入项：

航班号 String

起飞地点String

到达地点String

起飞时间String

到达时间String

票价 double

输出项：

新航线被加到Flight数据库中

测试计划：

建立服务器与数据库的链接，使需要添加的航线如预料添加。

* + - 1. delateFlightLine ()方法

删除废弃航线。

功能：

管理员可以通过这个方法删除废弃航线。

输出项：

航线被从Flight数据库中删除

测试计划：

建立服务器与数据库的链接，使需要删除的航线如预料删除。

* + 1. serverThread类设计说明
       1. serverThread ()方法

serverThread类的构造方法

功能：

建立Socket与端口，使客户端可以与服务器连接。

* + - 1. run()方法

启动多线程

功能：

启动多线程，调用DealRequest()处理客户端发送的各种请求。

* + - 1. WriteControl()方法

写控制台。

功能：

把各个客户端的请求处理过程写进控制台，方便管理员查看。

输出项：

客户请求信息写入控制台。

测试计划：

建立服务器与客户端的链接，客户端发送请求，确保服务器可以在控制台上写出正确的处理过程。

* + - 1. DealRequest()方法

处理客户端请求。

功能：

多线程接受客户端发送的请求，调用DealRequest()方法处理。

DealRequest()根据具体情况调用getAllMessage类中getInformation()方法处理，把结果发送至客户端。

测试计划：

建立服务器与客户端的链接，客户端发送请求，确保服务器可以调用正确的getInformation()方法

* + 1. getAllMessage类设计说明
       1. getInformation ()方法

服务器端最终要的方法，客户端与服务器交互的核心。

功能：

处理客户端传递的一切请求，并得出相应的处理结果传递给客户端。

输入项：

请求类型 int

输出项：

处理结果 String

算法：

1. 客户端发出请求。
2. 多线程接受请求，调用DealRequest()处理请求。
3. DealRequest()调用getinformation()处理具体请求，并将结果发送至客户端。

测试计划：

建立服务器与客户端的链接，客户端发送请求，确保服务器可以返回正确的处理结果至客户端。

* 1. client端程序说明：
     1. 类Customer设计说明
        1. login()方法

用于用户登陆系统。

功能：

用于用户登陆系统。

测试计划：

能顺利登录系统

* + - 1. register()方法

用于用户注册系统。

功能：

用于用户注册系统。

输入项：

用户名 String

密码 String

确认密码 String

输出项：

提示：是否注册成功

测试计划：

能正确识别输入的用户名与密码。如果该用户名已被使用，那么不能注册，如果输入的密码和确认的密码不统一，则输出提示。

* + - 1. exit()方法

用于用户退出系统。

功能：

用于用户退出系统。

测试计划：

能正确辨认用户是否已经登录，若已经登录，则可以退出系统。

* + - 1. queryFlights()方法

用于用户查询演出。

功能：

用于用户查询演出。

输入项：

起始地 String

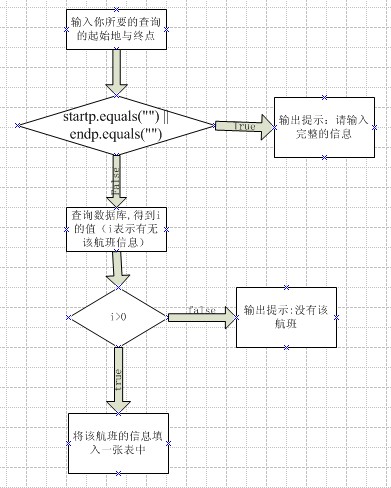
目的地 String

输出项：

符合条件的各演出的信息。

算法：

（见下一页）



测试计划：

能正确识别输入的开始时间和地点，根据这两个数据进行查询。

* + - 1. orderFlight()方法

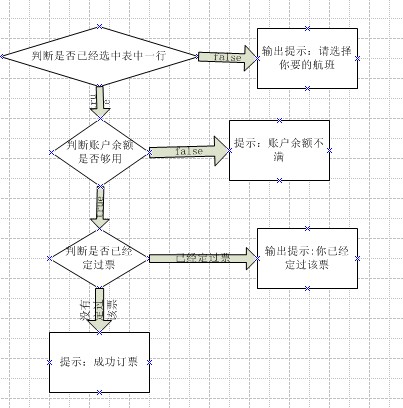
用于用户预定票。

功能：

用于用户预订票。

算法：

（见下一页）



测试计划：

能正确识别用户是否选择相应的航班，若用户没选择，则会提示用户请选择你所需要的演出。

* + - 1. queryOrders()方法

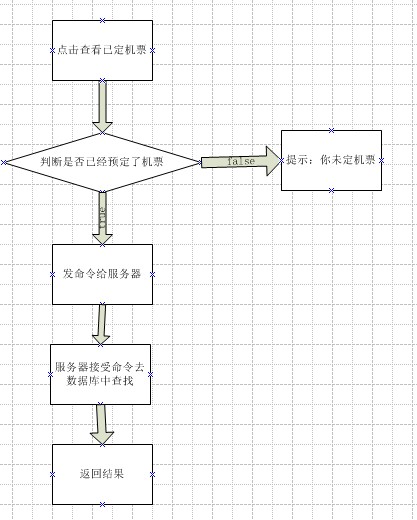
用于用户查询已定票。

功能：

用于用户查询已定票。

算法：

（见下一页）



* + 1. 类order设计说明
       1. clearOrder()方法

用于清除某个用户的订单。

功能：

用于清除某个用户的订单。

测试计划：

能顺利清除订单

* + - 1. deleteFlight()方法

用于用户取消某一座位。

功能：

用于用户取消某一座位。

测试计划：

能按要求正确删除

* + - 1. addFlight()方法

用于用户预定演出后，加入到用户的订单中。

功能：

用于用户预定演出后，加入到用户的订单中。

测试计划：

能正确添加演出。

* + 1. 类account设计说明
       1. decreaseMoney()方法

用于用户账户的扣钱。

功能：

用于用户账户的扣钱。

测试计划：

用户订票后，能根据所定的票，扣除相应的钱。

* + - 1. addMoney()方法

用于用户往账户里充钱。

功能：

用于用户往账户里充钱。

测试计划：

能正确往账户里充钱。

* + - 1. getID()方法

得到账户的用户名

功能：

得到账户的用户名

测试计划：

能得到正确的用户名

* 1. 注释设计

尽可能的将软件中插入注释语句，使语句功能明了。制作客户端的ASP网页的时候应该制作两分，一份是标有注释语句的网页，用来给维护人员、测试人员和开发人员了解开发过程所用，另一份是不带有注释语句的网页，用来最后实际应用当中，这样可以充分的利用有限的带宽，降低客户的计算机打开网页的时间，提高客户端的浏览速度。

* 1. 限制条件

当系统第一次使用时，具有统一的用户ID和密码，超级用户在3次验证错误后，系统自动关闭。限制必要的条件，以排除由于用户的误操作造成不必要的错误。

* 1. 测试计划

数据库连接正常并且传入数据库的数据匹配，则合法；若不匹配则非法，数据库连接异常，而且传入数据库连接正常，成功修改出库表库存表，若传入数据库数据异常，给出错误类型和相应的提示。

在软件编辑的工作进行当中，测试人员便要开始制定测试计划，其中要包括白盒和黑盒的具体测试项目，及其必要的测试数据和出错的信息。每次测试的结果要写报告，并就发现和怀疑的问题与编辑人员联系。测试的结果要让编辑人员明白。

* 1. 尚未解决的问题
     1. 客户端尚未解决的问题

客户端尚未解决的问题是，如何与订票人的银行账户连接。

* + 1. 服务器端尚未解决的问题

服务器端尚未解决的问题是，系统自动检索到过期的航班后自动将其信息删除并且要保所有与该航班关联的订票人的信息和乘客的信息都删除，难点在于，如何保证数据库中的表间的一致性。

* + 1. 难点问题

难点问题是服务器端的程序的删除部分，因为订票人的信息关系到部分客人的信息，必须保持数据的完整性。其次是演出信息，它关系着部分订票人的信息和顾客信息，其中还嵌套着这部分订票人当中有部分关系的顾客信息，并且同一定票人可以订购多个演出的票和所订购的票的数量也不仅仅是1张，要将信息删除，也要保数据的完整性。