华东交通大学

# 毕业设计（论文）

题目：铁路设备信息管理系统的设计与实现

| | | | |
|---|---|---|---|
| 学　　院: | 软件学院 | | |
| 专　　业: | 软件+交通设备信息工程 | 班　　级: | 2 班 |
| 学生姓名: | 付泽奇 | 学　　号: | 20112110120217 |
| 指导教师: | 魏波 | 完成日期: | 2015 年 5 月 25 日 |

## 毕业设计（论文）诚信声明

**本人郑重声明：所呈交的毕业设计（论文）是我个人在导师指导下进行的研究工作及取得的研究成果。就我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表和撰写的研究成果，也不包含为获得华东交通大学或其他教育机构的学位或证书所使用过的材料。**

**如在文中涉及抄袭或剽窃行为，本人愿承担由此而造成的一切后果及责任。**

本人签名

导师签名

**2015 年 5 月 25 日**

# 华东交通大学毕业设计（论文）任务书

| 姓名 | 付泽奇 | 学号 | 20112110120217 | 毕业届别 | 2015 | 专业 | 软件+交通设备信息工程 |
|---|---|---|---|---|---|---|---|
| 毕业设计（论文）题目 | 铁路设备信息管理系统的设计与实现 | | | | | | |
| 指导教师 | 魏波 | 学 历 | 研究生 | 职 称 | 讲师 | | |

具体要求：

（1）基本要求：

对铁路设备管理的各项数据充分了解，对铁路信号设备的需求进行调研，依据需求设计系统的各个模块，熟悉 C#在开发中的使用，以及数据库（MS-SQL Server）的建立和使用，熟悉 Microsoft Windows Visual Studio 2013 开发环境，对三层结构体系构架的模式不断完善，绘制系统的主体功能图，E-R 分析图，设计数据库，表，数据。依据上述完成各个部分模块的开发。

（2）创新性要求：

1. 按照要求科学的建立数据库，便于对数据的全方面的管理和控制。
2. 更人性化的 UI 设计，方便简单易懂的设计理念。
3. 不断的完善系统稳定性和可利用价值，软件有更好的扩展性。

严格按照华东交通大学毕业设计（论文）管理规范，不舞弊，不抄袭，按照规定时间内独立完成。

进度安排：

第 1～2 周：资料准备阶段。按任务书要求进行资料收集、调研，功能、需求分析，文献查阅等。

第 3 周：论文前期阶段。完成开题报告和检索相关外文资料，选择正规出处的外文资料进行翻译工作。

第 4～8 周：系统分析设计、论文初期，毕业设计中期检查。按任务书要求进行系统分析设计工作，并于第 8 周内将毕业设计中期检查材料（开题报告、外文资料翻译等）提交指导教师审核。

第 9～11 周：论文初稿撰写及提交。 撰写毕业设计(论文)初稿，并于第 11 周周日前提交指导教师批改。

第 12 周：在指导教师初稿批改的基础上，完成毕业设计（论文）的第二稿，并提交指导教师进行修改审阅，直到指导教师认为可以定稿为止。

第 13 周：论文定稿提交。论文终稿必须在第 13 周周五前提交指导教师，采用活页形式装订、待查重、评阅、答辩后再正式装订。

第 14 周：查重、答辩准备。经指导教师认可签字方能参加答辩，并做好答辩准备。

第 14 周周末：毕业设计答辩。

第 15 周：公开答辩，实践管理系统中资料完善。

指导教师签字：

2014 年 12 月 29 日

| 题目发出日期 | 2014.12.30 | 设计（论文）起止时间 | 2014.2.24-2015.5.23 |
|---|---|---|---|

学院意见：

同意发布题目

毕业设计领导小组组长签章

# 华东交通大学毕业设计（论文）开题报告书

| 课题名称 | 铁路设备信息管理系统的设计与实现 | | | | | |
|---|---|---|---|---|---|---|
| 课题来源 | 自选 | | 课题类型 | CX | 导 师 | 魏波 |
| 学生姓名 | 付泽奇 | 学 号 | 20112110120217 | | 专 业 | 软件+交通设备信息工程 |

**一、开题报告内容：**

1、文献综述

“铁路设备信息管理系统”旨在铁路系统内，为铁路从业者提供更好的软件服务。软件主要针对铁路运营单位，为铁路运营单位提供即时、准确的设备信息，互联网带动了世界的信息化革命，传统的管理方式已经不复存在，数字化的管理模式日益凸显它的优势，但是，光有计算机的硬件飞跃式发展是不行的，软件也必须紧跟时代的发展，一个更加开放的，优质、稳定、高效、优美的管理软件就应运而生。不断在实践中摸索管理系统的稳定性，可扩展性，给企业客户带来最便捷的体验。

本系统主要为铁路管理客户提供注册、登录、设备管理、用户权限处理等功能，并进行系统分析、总体设计、详细设计和系统测试等。为了实现这些需求，主界面做到功能齐全，模块划分比较明确。在这个时代，管理系统已经和我们大家的工作紧紧相关了，随着计算机的到来，电脑技术把现在的生活和工作变得更加高效和方便。

信号设备的多样化、丰富化使得我们在传统的管理上比较费时费力，计算机管理已经成为我们现代必不可少的方式，通过管理系统可以轻松快捷地实现设备信息的管理，机械种类多，我们之前需要靠非常多的记录员来确认设备的状态和设备的各种情况信息。现在，有了计算机的帮助，管理系统的引进，管理上变得越来越智能和便捷。

2、参考文献


[1] 李德坤. 车站信号设备管理系统的设计与实现[D]. 大连理工大学, 2008.

[2]Shahram Gilaninia,Seyyed Javad Mousavian, Orang Taheri, Hamid Nikzad, HodaMousavi , Fatemeh Zadbagher Seighalani. Information Security Management onperformance of Information Systems Management [J], Journal of Basic and AppliedScientific Research, 2(3)2582-2588, 2012.

[3]Daniel Mellado,David G. Rosado. An Overview of Current Information Systems Security Challenges and Innovations[J]. Journal of Universal Computer Science, vol.18,no. 12 (2012),1598-1607.

[4]韦银星, 张申生, 曹健. UML 类图的形式化及分析[J]. 计算机工程与应用, 2002, 38(10):5-7. DOI:10.3321/j.issn:1002-8331.2002.10.002.

[5]郑人杰等.实用软件工程(第二版)[M].北京:清华大学出版社,2004.

[6]李平,赵丽华,马丽.管理信息系统[M].北京:清华大学出版社,北京交通大学出版社,2006.

[7] LeszekA.Maciaszek, Bruc Lee Liong.实用软件工程[M].北京:机械工业出版社,2007.

[8]薛华成.管理信息系统(第四版)[M].北京:清华大学山版社,2003.

[9]邝孔武,王晓敏?信总系统分析与设计(第 3 版)[M].北京:清华大学出版社,2006.

[10]王珊,萨师煊.数据库系统概论(第四版)[M].北京:高等教育出版社,2006.


课题类型：（1）A—工程设计；B—技术开发；C—软件工程；D—理论研究；

（2）X—真实课题；Y—模拟课题；Z—虚拟课题

（1）、（2）均要填，如 AY、BX 等。此部分可以附页

# 华东交通大学毕业设计（论文）开题报告书（续）

**二、方法及预期目的：**

1、拟采用的研究方法（手段）

使用基于 C#语言，在 Windows + Visual Studio+MS-SQL2014 开发环境下，写的软件设计。充分了解铁路设备管理的各项数据，并且要求建立数据库，便于对数据的全方面的管理和控制。以及不断的完善系统稳定性和可利用价值。

2、本课题要研究或解决的问题及预期目的

随着铁路的现代化、跨越式的发展，铁路内部的信息沟通也变得越来越重要，甚至是对设备的“沟通”，为了更方便的管理铁路内部的资产信息，以及设备的维护情况和设备的运行状态，操作维护人员能及时和方便的了解设备的运行状态以及设备的基础信息，开发一个设备资产及维护信息管理系统。更直观的展现出，内部的资产和设备的维护信息，对设备进行较为及时的监控信息，对维护员的业绩考核，产品的质量能有更加直观的了解。同时，对设备维修时间，维修员能更加清楚的知道设备性能以及运行状态，及时检修，以避免带来铁路运行上的事故发生。

信息化是各行各业都面临的一个巨大的改革，如何科学的运用新技术，也是我们面临的挑战，互联网甚至是物联网也好，给我们的设备运行，人员工作将会带来更多的好处，更少的人工定点定时操作，取而代之的是信息化给我们带来的便利和可靠。

3、进度表

第 1～2 周：资料准备阶段按任务书要求进行资料收集、调研，功能、需求分析，文献查阅等。
第 3 周：完成开题报告和检索相关外文资料，选择正规出处的外文资料进行翻译工作。
第 4～9 周：按任务书要求进行系统分析设计工作，并按时提交开题报告、外文资料翻译等。
第 10 周：论文的初稿撰写 撰写毕业设计(论文)初稿并上交指导教师修改。
第 11 周：论文修改、定稿 在指导教师初稿批改的基础上，完成毕业设计（论文）的终稿，并提交给指导教师进行修改审阅，直到指导教师认为可以定稿为止。
第 12 周：配合指导教师完成毕业论文打印、装订工作，并将打印好的论文和相关材料交指导教师评阅，经指导教师认可方可参加答辩，并做好答辩准备。
第 13 周：论文修改论文答辩
第 14 周：公开答辩

**三、指导老师意见**

同意开题

指导教师签名：　　日期：2015.3.16

# 华东交通大学毕业设计(论文)评阅书(1)

| 姓名 | 付泽奇 | 学号 | 20112110120217 | 专业 | 软件+交通设备信息工程 |
|---|---|---|---|---|---|
| 毕业设计(论文)题目 | 铁路设备信息管理系统的设计与实现 | | | | |

**指导教师评语：**

| 具 体 要 求 | 优 | 良 | 中 | 一般 | 差 |
|---|---|---|---|---|---|
| 出勤及工作态度（20%） | | | | | |
| 方法合理，设计工作量饱满（20%） | | | | | |
| 论点正确，论文内容有一定难度（20%） | | | | | |
| 结构严谨，论文有一定应用价值（20%） | | | | | |
| 对前人工作有改进或有独特见解。（10%） | | | | | |
| 论文格式正确，撰写规范（10%） | | | | | |

| 得分 | |
|---|---|

**指导教师签字：**

**2015 年 5 月 日**

**评阅人评语：**

| 具 体 要 求 | 优 | 良 | 中 | 一般 | 差 |
|---|---|---|---|---|---|
| 选题合理，设计有应用价值（20%） | | | | | |
| 方法合理，设计工作量饱满（20%） | | | | | |
| 论点正确，论文内容有一定难度（20%） | | | | | |
| 对前人工作有改进或有独特见解。（20%） | | | | | |
| 论文格式正确，撰写规范（20%） | | | | | |

| 得分 | |
|---|---|

**评阅人签字：**

**2015 年 5 月 日**

# 华东交通大学毕业设计(论文)评阅书(2)

| 姓名 | 付泽奇 | 学号 | 20112110120217 | 专业 | 软件+交通设备信息工程 |
|---|---|---|---|---|---|
| 毕业设计(论文)题目 | 铁路设备信息管理系统的设计与实现 | | | | |

**答辩小组评语：**

| 具 体 要 求 | 优 | 良 | 中 | 一般 | 差 |
|---|---|---|---|---|---|
| 符合要求 | | | | | |
| 答辩准备充分，论文题目与内容相符 | | | | | |
| 语言精练能突出重点，思路清晰能准确表达 | | | | | |
| 论点正确，论文内容有一定难度 | | | | | |
| 方法合理，论文内容工作量饱满 | | | | | |
| 结构严谨，论文有一定应用价值 | | | | | |
| 对前人工作有改进或有独特见解 | | | | | |
| 正面回答问题 | | | | | |
| 回答问题有理论依据，基本概念清楚 | | | | | |
| 主要问题回答准确，深入 | | | | | |

| 得分 | |
|---|---|

**组长签字：**

**2015 年 5 月 日**

**答辩委员会意见：**

**同意以上评定，根据前面三项得分，按 25:15:60 的比例评定总成绩为**

| 等级 | |
|---|---|

**进入二次答辩** ☐ **评优组** **最终成绩由二次答辩决定**

☐ **合格组**

**答辩委员会主任签字：**

**2015 年 5 月 日（学院公章）**

注：答辩小组根据评阅人的评阅签署意见、初步评定成绩，交答辩委员会审定，盖学院公章。

“等级”用优、良、中、及、不及五级制（可按学院制定的毕业设计(论文)成绩评定办法评定最后成绩）。

# 华东交通大学毕业设计（论文）答辩记录

| 姓名 | 付泽奇 | 学号 | 20112110120217 | 毕业届别 | 2015 | 专业 | 软件+交通设备信息工程 |
|---|---|---|---|---|---|---|---|
| 题目 | 铁路设备信息管理系统的设计与实现 | | | | 答辩时间 | | 2014 年 5 月　日 |
| 答辩组成员（签字）： | | | | | | | |
| 答辩记录：<br><br>记录人（签字）：<br>2014 年 5 月　　日<br><br>答辩小组组长（签字）：<br>2014 年 5 月　　日 | | | | | | | |

# 铁路设备信息管理系统的设计与实现

## 摘　要

中国铁路几十年的发展历程以及大量投资设备建设，铁路电子信息化部分同时也取得了很大的成就。但是因为较早前的信息化建设普遍缺乏统一的规划和部署,从而导致了各个业务体系不完善,一些路局单位上的设备信息管理工作仍然靠人工统计。因此，统一部署的铁路设备信息管理系统的设计工作十分必要。

本文简单给出了铁路设备信息管理系统的主要设计背景和其重要意义，并且根据铁路业务上的实际需求特别制定了符合要求的 C/S 构架信息管理系统。在整体的要求框架下，以及在实际需求分析过程中，重点的阐述了系统的各个功能模块和不同模块间的相互关联，相互调用关系，最后说明了系统重点功能的一些重要的技术实现过程。

通过对鹰潭铁路局的参观，切实的感受到铁路设备信息管理的重要性，以及信息化在铁路运输的重要地位和信息管理的科学合理的调配在各个领域的重要性。

**关键词：铁路设备信息；C/S 构架；数据库设计**

# Design and Realization of Railway Equipment Information Management System


## Abstract

China's railway development process for decades and a large investment in equipment construction, electronic information section of railway also made great achievements. But ,because of the earlier information technology were lack of unified planning and deployment, resulting in various business system is not perfect now, some of the equipment information management unit on the railway administration still rely on manual statistics. Therefore, the design of railway equipment information management system unified deployment is necessary.

This paper mainly described the background of information management system of railway equipment and its significance are given, according to the actual needs of the railway business especially formulated with C/S framework of information management system requirements. The requirements of the overall framework, as well as the actual demand analysis process, focusing on the relevance of the various functional modules of the system and between different modules, call each other, finally explains the implementation process of some important technical system key function.

According to visit the real the Yintan Railway Station, feel the importance of railway equipment information management really, and know the importance of information in the important position of railway transportation and information management of scientific and reasonable allocation in various fields.



**Keywords: Railway equipment information System; C/S architecture ; Design for Database**

# 目　录

# 1 绪论

## 1.1 研究的背景及意义

### 1.1.1 选题的背景

铁路现代技术近年来迅猛发展，使得铁路运营已经成为我们生活中不可或缺的部分，列车运行公里数的持续增加，铁路上的设备也随着呈几何次数增加，尤其是电子技术的推广及铁路系统上的计算机联锁的广泛使用，使高速铁路运营在远距离出行中凸显出更加特别的优点，但也因此带来了更多的问题，如设备的维护问题等等。

以一次在鹰潭机务段的实习经历为例，大量标有相应的号码的配件和电器设备正在安装，这些号码以能唯一确定设备的功能属性以及使用单位，简洁方便，可免去纸质登记，报修，纸质记账等繁琐程序。

车站信号系统是铁路系统的一个重要的组成部分,能保证铁路运输安全,提高运输效率。它主要实现车站行车与调车进路管理的自动化,保证行车与调车的安全,提高站内的通过能力。[1]

### 1.1.2 国内外研究现状

现在我国的高速铁路运输目前处于高速的发展期，自动化要求日益明显，更加要求管理系统的重要性。

在国外，Shahram Gilaninia, Seyyed Javad Mousavian, Orang Taheri, Hamid Nikzad, HodaMousavi, Fatemeh Zadbagher Seighalani,提出信息安全规划的五个阶段,并给出了相应的建议措施有很强的借鉴意义[2]。Daniel Mellado, David G. Rosado,主要是从安全组织方面入手,介绍了目前信息安全系统所面临的挑战,然后从加密技术、安全标准,隐私等出发阐述如何处理解决安全事件[3]。

### 1.1.3 研究的意义

铁路信息管理系统是更智能化、多元化的管理系统，同事让铁路运输更加高效和方便，节省更多人力、物力，给铁路运输带来更高效的管理体验。

## 1.2 系统目标

### 1.2.1 C#语言介绍

在.NET 平台上的 C#语言，是计算机语言发展的主流，是完全面向对象的编程语言，当前正式版的诞生 C#语言是 3.0。 C#语言是基于 Java 和 C ++的基础上，它的代码风格的开发编程语言和 Java 类似，ASP.NET 开发背景的代码是通过 C#语言来实现的。C#结合 VB 简单的可视化以及 C ++效率高的优势，强大的运营能力，新颖的语言特性等来支持面向组件的编程语言.NET 的开发。

C#是流行的 Java 诞生后，一种新的语言。C#是依赖于强大的.NET 开发。在 Windows 2003 和 Vista 系统中可以看出（原建.net1.1，后者建.net2.0）。C#语言目前在网络编程使用相对比较广泛，在数据库以及在 Windows 窗口程序设计中，也不断的普及中。

### 1.2.2 SQL Server 2014 数据库优点

SQL Server 为了减少黑客篡改的风险，数据库中敏感的数据部分功能将不能使用，除非管理员使用了管理员权限主动开启。和以前版本的 SQL 一样，2014 也要求数据库的密码强度也需要和 Windows 一样，将不可以使用较弱密码。在一定条件下，SQL 还会要求使用者在下次登录时候，把登录的密码进行修改，和上次不同。

微软在云计算方面的努力也是不容忽视，在 2014 上带来全新的云服务的理念，产品全面面向 Azure 服务，包过代码均可以托管在微软云当中，或者 GitHub 当中，使用微软提供的云数据库的功能，可以帮助公司建设完备的云服务体系，减少 IT 设备的使用费用等。

综合概括而言：SQL Server 2014 数据库优点包括提供关键业务的高性能支撑、从任意数据中快速获取洞察力、完整和一致的混合云平台支持。

### 1.2.3 ASP 介绍

Microsoft.net 的战略核心产品，即 ASP.net。 ASP.NET 与 ASP 有强的兼容性，代码上基本相似，ASP.NET 面向开发者推出新编程模型，对应用程序可提供较强的安全性能。在 ASP 开发的应用程序中，加入 ASP.NET 功能，可以让 ASP 应用程序具有更厉害的功能。

ASP.NET 可以用任何与 .NET 兼容的语言创作应用程序。另外，任何 ASP.NET 应用程序都可以使用整个 .NET Framework。程序开发人员在使用 ASP 中，可以很轻松的使用新技术的优点，可以在平台中轻松的编写自己的程序和网页等。

ASP.NET 的新性能，在 ASP.NET 中，微软给程序员提供了比较稳定的性能，开发也更加方便，管理更加简洁，提供更加强大的网络服务。编程中更多繁复的流程，已经不再需要，在 IDE 编程器中，微软已经提供了整套的解决方案，程序员只要处理好代码问题即可。

# 2 需求分析

在正常软件系统开发流程当中，系统的分析以及软件设计阶段，在需求分析中有举足轻重地位。做好分析工作，有助于尽快帮助和避免或及早消除误差，提高软件生产率，降低开发成本，提高软件质量。

## 2.1 系统功能概述

铁路设备信息管理系统是本系统主要完成的任务，包括用户登录，用户注册管理，设备管理，密码修改，日志查询，便民服务，关于帮助等功能，充分展现了系统的便捷和以及对系统维护的简便。同时支持多端操作和访问，给操作上带来更多便捷。Web 端服务，只要有浏览器就能操作，不再拘束在电脑前；手机客户端，更加方便的查询和了解各设备的状态和库存。

### 2.1.1 系统功能需求

1、设备信息管理，首先对设备信息管理系统进行布局上的调整，选择适应于窗口的图像作为我们的软件 Background 背景，对用户 UI 做了比较易用化的调整。功能上完全能满足大多数场景的使用，本软件主要还是针对铁路设备的信息管理进行详细的设计，主要比较详细的记录如，设备的序号，设备名称，使用单位，设备生产日期，设备质量，质量检测人员等。同时支持导入和导出记录（Excel 格式），对数据的维护备份，打印数据报表，带来不可比拟的方便。对设备信息支持精确搜索以及模糊搜索。

2、登陆日志管理，对于用户的使用情况进行详细的登记，登陆情况，以及用户的权限分配都有详细的说明，不同的账户使用权限不同，这样便于对用户的管理和信息的记录。

3、用户信息管理，对用户进行创建，权限分配，密码修改，都是不可或缺的部分，我们将用户注册权限仅分配给管理员。在信息管理系统中，系统最高管理员才有对普通用户进行权限分配的功能，以及新建新用户组的功能。这样做，可以表现出完善的管理形式。

4、对于设备采销系统进行模块化集成，对设备购置进行统一处理，统一平台进行操作，给系统带来更多的便利性。

### 2.1.2 用例分析

例图的主要功能，可以很生动形象的表现出主模块的运行流程。可以让用户选择需要的功能，并且通过类似流程图的方式展现给客户。简单的讲，它就较为详细的描述软件不同功能模块，以及模块功能。系统分析阶段开始前需要做的前期工作，我们可以通过需求分析来解决这个问题。

### 2.1.3 类图分析

类就是，具有一致属性，使用一样方法的对象的集合。类图是由解释性的模型元素组成的集合。类之间的联系表示施加在对象上的约束。类图的语义模型应该能够描述与某个类相关的对象的集合以及施加在对象上的约束。[4]

## 2.2 系统性能要求

### 2.2.1 系统时间特性要求

对资源实时搜索的速度，必须高效。系统运行速度，必须使用户可接受。但实际使用中，考虑到需要实时检测设备，以及设备相关属性情况。

### 2.2.2 系统灵活性

系统中要求较好的扩展性，才能让方便的增加设备成为可能，增加设备其他属性，满足系统以后扩容的需求。

### 2.2.3 数据管理能力要求

在这次的设计里，用的是微软 SQL Server 2014 数据库管理软件。整个系统，要有较好的数据分析和处理性能。因为不确定用户的数量，不清楚设备总量，默认设置数据存储量为 15GB，确保数据的安全存放，存储空间够用。

## 2.3 可行性分析

在系统调查的基础上,根据客户可能提供的时间和资源条件进行的专门研究,充分详实的可行性分析可以避免人力、物力和财力上的浪费,保证系统开发的成功[5]。该系统的可行性分析包括以下几个方面的内容[6]：

### 2.3.1 技术可行性

在经过前期调查后，发现结合 C#编程语言以及 Microsoft SQL Server 2014 等开发工具，能够成功地创建出基础型铁路信息管理系统来提供给客户使用。

### 2.3.2 经济可行性

在铁路设备信息管理系统中，前期的投入成本较大，主要是系统的设计费用，以及运行设备的投入，但后续的维护投入相对较小，较长时间内都不需要再次进行大额投资，管理和维护费用也很少，而系统实际提供的价值远远大于将投入的开发费用，综上这个系统是十分有开发价值的。

### 2.3.3 操作可行性

这次开发的管理系统，在界面上，直观、简单易懂、操作方便。它能够准确记录、检索和管理有关铁路各类设备信息和用户信息，帮助铁路工作人员及时掌握和分析铁路交通运行情况，及时做出正确决策，并且便于相关内部人员对整个铁路系统的管理，因而大大提高了铁路运行的管理水平与效率。

综上，我们认为从经济方面、技术方面和操作等方面来说，系统完成基本开发没有技术和经济障碍，是值得使用和推广的。同时我们对软件的 UI 进行了美化操作，让使用者能感受到便捷的操作。

# 3 总体设计

## 3.1 运行环境

在现代，计算机技术的飞速发展，不得不考虑设备升级，我们将提供最低配置供参考，开发环境应比此环境配置稍高，运行环境则足够。我们详细制定了，针对不同客户平台的方案，让客户更加方面操作。对于铁路上的使用者，考虑到数据量的巨大，建议购买性能较好的服务器来承载大量数据库。或者，将数据大规模迁移至弹性云平台，以满足数据的处理和储存能力。我们尽可能保障软件在多种环境下，安全可靠运行，由于软件系统环境的多样性，不保证在所有的环境下都能持续无故障运行。

1．PC 端软件使用环境：

操作系统：Windows XP 或 Windows 7 以上；

Web 端软件使用环境： 操作系统：Windows XP 或 Windows 2003 以上，要求 IE 7 以上，推荐使用 Chrome 内核浏览器，最佳分辨率：1680*1050；整体硬件使用环境：CPU：Intel Pentium(R) 4 以上；内存：512M；硬盘：80G；网卡：100M。

2．PC 端硬件使用配置：CPU：Intel i3 以上；内存：2G；硬盘：50G；网卡：10M；

服务器搭建最低配置：CPU：Intel 至强系列；内存：20G；硬盘：1T；网卡：1G。

3．开发使用环境配置：

平台环境：Windows 7 + Visual Studio 2013 + MS-SQL 2014 ，C#，ASP；

开发工具：Microsoft SQL Server 2014，Visual Studio 2013，Dreamweaver；

开发使用硬件配置：CPU：Intel i5 以上；内存：4G；硬盘：100G；网卡：100M。

## 3.2 基本处理流程

PC 端程序系统功能模块图的基本流程，从登陆界面开始，登录到主界面，选择主界面不同功能。详细如图 3-1 系统功能模块图所示。在统一的主界面下，相关的但是不同模块间也有相互调用，任意子模块均能跳转回主界面。

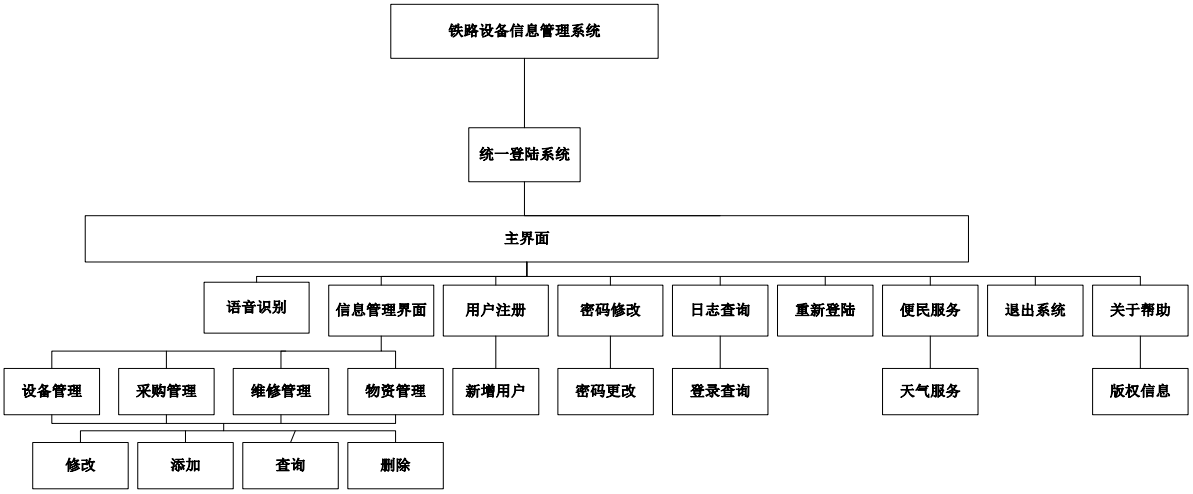


图 3-1 系统功能模块图

## 3.3 数据库设计

### 3.3.1 概念结构设计

系统需求分析是确定需求和详细说明需求的活动[7]。数据库新建以及在系统中用到的设计，我们称为数据库设计，开发和建设的主要用到了信息系统先进技术，特别是在数据库实现当中，在特别的使用环境下，使用更加优化的数据库构架。铁路信息管理系统中重要的部分就是数据库设计部分。管理系统设计的数据库使用了 Microsoft SQL Server2014，作为主要数据库。因为，用微软提供的数据库管理工具，增加、删除、修改工作相对简便。数据库的按一定序列排列以及主键索引功能，带来更多便捷。这项有点，可以对数据库内，进行迅速定位、查询等操作。将需求分析得到的用户需求抽象为信息结构即概念模型的过程就是概念结构设计。它是整个数据库设计的关键[8]。

### 3.3.2 设计思路

如下问题在系统中，需要解决。主要问题是，怎么去处理收集来的基本数据以及对数据的操作要求。

1.对数据的基本要求：怎么从数据库获取相关数据，然后合理地处理怎么存储在数据库当中。

2.对数据的操作要求：通过建立用户的关系数据，让确定数据关系更加方便。

### 3.3.3 实体图和 E-R 图

数据库设计的关键，就是结构设计。可以通过的需求，归纳和抽象合成概念模型。从而形成 DBMS，最后生产独立 E-R 图。E-R 图，即用于数据的分析。铁路信息管理系统的主要实体如下分析。

下面所示为管理员实体图，如图 3-2 所示，主要展示管理员和普通用户之间的联系和权限关系。普通用户具有用户名，用户密码，权限等属性，普通用户可以在相应的权限下进行不同的操作。

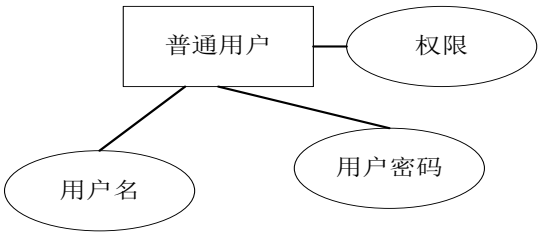


图 3-2 管理员属性实体图

用户日志实体图，如图 3-3 所示，展示了用户日志各个属性关系，对用户登录，退出操作，进行了详细的记录，方便后期查询。 记录的内容包过序号，登陆过的用户名，用户类型，登陆的状态是登陆还是退出，登陆时间。对这些内容都有详细的记录，程序获取

到这些信息之后，提交到数据库保存。

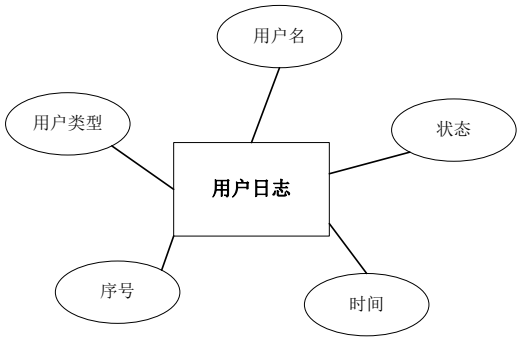


图 3-3 用户日志实体图

设备信息实体图如 3-4 所示，阐释了铁路设备信息之间的关系。设备具有的属性应该有：序号，设备编号，设备名称，设备型号，生产日期等属性。这些信息能记录一个设备的基本属性，以及这些属性在设备中的记录。不同的设备有唯一的序号，同种设备有唯一的设备编号，这样管理才不会显得比较混乱。

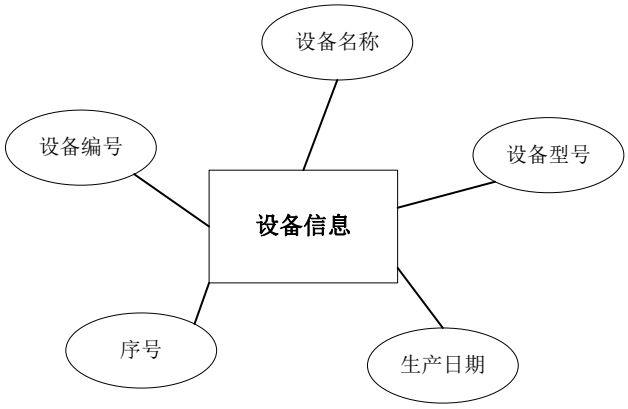


图 3-4 设备信息实体图

用户属性实体图，如图 3-5 所示，描述了用户的属性，对用户不同的属性进行分类操作。不同的用户获得的权限也不一样，所以必须对用户的权限进行细分。分为：日志权限，改密权限，注册权限，管理权限。同时用户的属性还一定包过用户名和密码，以及用户类型，每个用户都唯一对应一个序号。为了方便管理，用户注册初始时都会赋予管理员最高的权限，普通用户最低权限。

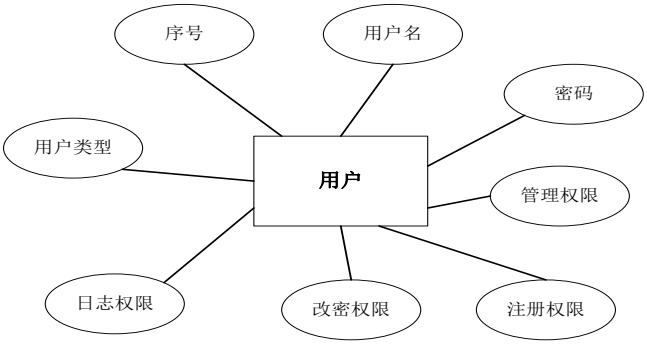

图 3-5 用户属性实体图

设备的多种属性才给了系统的多样性，设备中心信息实体图，如图 3-6 所示。设备的详细信息，例如序号，设备编号，设备名称，设备类别名称，设备编号，符合，购置价格，生产日期，可用年限，报废日期，入录时间，备注等，都是铁路设备不可缺少的详细属性，在生产生活中，经常需要的用到的设备信息。

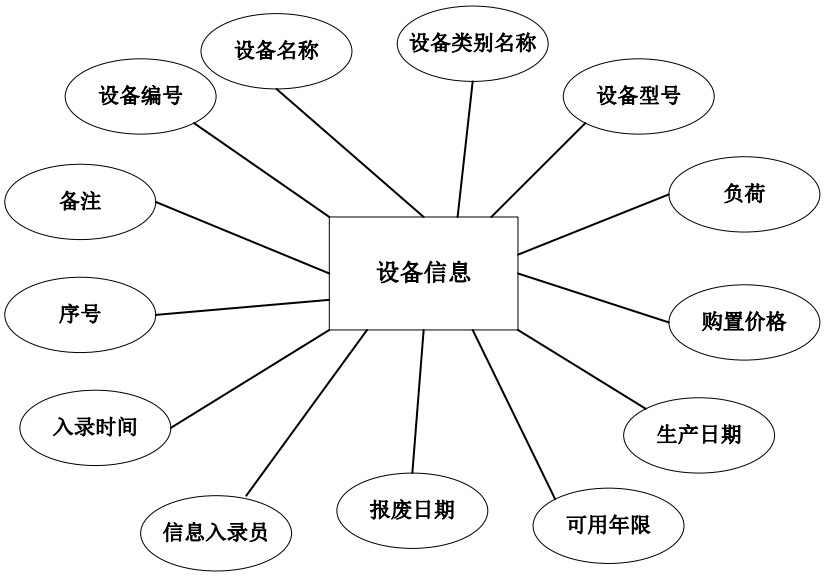


图 3-6 设备中心信息实体图

下面所示即为用户关系 E-R 图，主要展示管理员和普通用户之间的联系和权限关系。一个用户可以有多个登陆日志。比如说，管理员可以登陆到系统中很多次，但是一条用户日志，仅仅只能有一个用户登陆的记录。所以绘制出下图 3-7 所示，用户的不同属性，对应在用户日志中的关系。

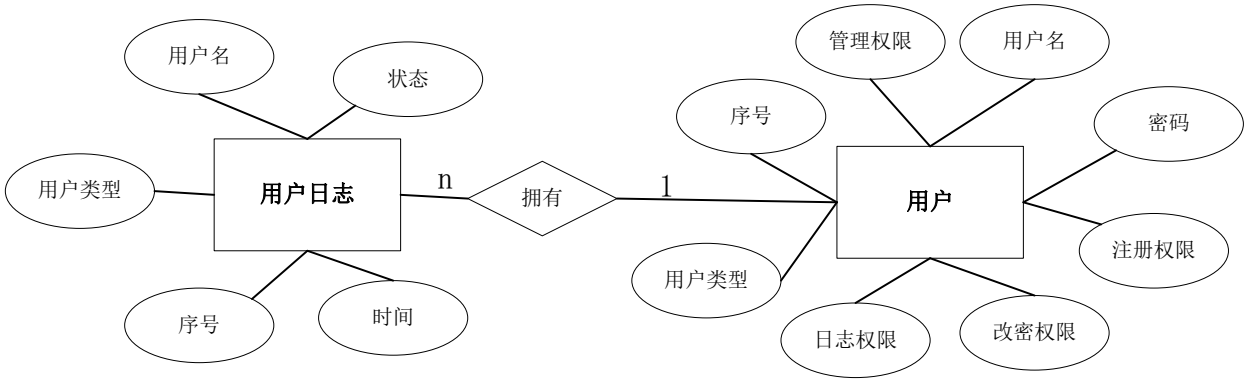


图 3-7 用户关系 E-R 图

下面所示为设备与报修 E-R 图，主要展示了设备信息与设备维修之间的关系。同一个设备可以有多个维修记录，但是每一条维修记录只能对应一个设备。所以绘制出下图 3-8 所示，在设备故障的记录中，设备序号与设备的名称等于设备信息是对应一致的。

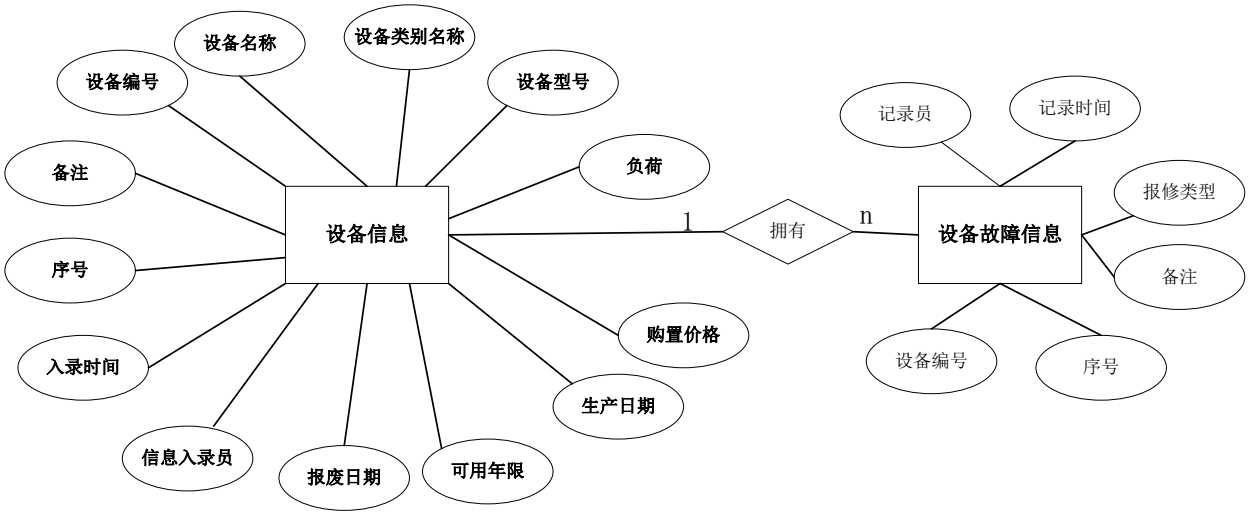


图 3-8 设备与报修关系 E-R 图

## 3.4 逻辑结构设计

### 3.4.1 设计思路

逻辑结构转化成为 DBMS 所能支持的数据库模型，并对模型进行优化。根据上面所述 E-R 图，对数据库结构进行详细设计和分析。才能得出如下主要的数据表格。

### 3.4.2 逻辑模型

通过设备信息属性之间的关系，分析出设备信息表。如表 3-1 所示。在表中详细的分析各设备信息的属性。序列号为主键，并且不可以设置为空，建立索引。序列号设定为自动更新加 1，不可以被手动更改。为了保障了数据的完整性，设置不同数据类型。

表 3-1 设备信息表

| 数据库表名称 | 数据项目（字段号） | 数据类型 | 描述 | 主键 | 是否可以为空 |
|---|---|---|---|---|---|
| dbo.Device | Id | int | 序列号 | 是 | 否 |
| dbo.Device | Dno | int | 设备编号 | | 否 |
| dbo.Device | Dmodel | nvarchar(50) | 设备型号 | | |
| dbo.Device | Dname | nvarchar(50) | 设备名称 | | |
| dbo.Device | Dprodatetime | datetime | 生产时间 | | |
| dbo.Device | DEngineLoad | numeric | 负荷 | | |
| dbo.Device | DEngineName | nvarchar(50) | 设备类别名称 | | |
| dbo.Device | DEngineNo | nvarchar(50) | 发动机编号 | | |
| dbo.Device | DEngineNote | nvarchar(100) | 备注 | | |
| dbo.Device | DEnginePower | mumeric | 功率 | | |
| dbo.Device | DEquAttribute | nvarchar(50) | 属性 | | |
| dbo.Device | DEquBuyPrice | numeric | 购置价格 | | |
| dbo.Device | DEquBuyTime | datetime | 购置时间 | | |
| dbo.Device | DEquColor | nvarchar(50) | 机车颜色 | | |
| dbo.Device | DEquDepreciation | numeric | 年折旧率 | | |
| dbo.Device | DEquDiscard | datetime | 设备报废日期 | | |

表 3-1 设备信息表(续)

| 数据库表名称 | 数据项目（字段号） | 数据类型 | 描述 | 主键 | 是否可以为空 |
|---|---|---|---|---|---|
| dbo.Device | DEquKind | nvarchar(50) | 机种 | | |
| dbo.Device | DEquLife | nvarchar(50) | 可用年限 | | |
| dbo.Device | DEquNo | nvarchar(50) | 牌照号 | | |
| dbo.Device | DEquPlace | nvarchar(50) | 产地 | | |
| dbo.Device | DEquproName | nvarchar(50) | 生产厂家 | | |
| dbo.Device | DEquRunTime | numeric | 运行台时 | | |
| dbo.Device | DEquStatus | nvarchar(50) | 设备状态 | | |
| dbo.Device | DEquType | nvarchar(50) | 类别 | | |
| dbo.Device | DEquUseTime | datetime | 购置时间 | | |
| dbo.Device | DEquValue | numeric | 设备净值 | | |
| dbo.Device | DEquWeight | numeric | 机械重量 | | |
| dbo.Device | DFuelType | nvarchar(50) | 燃油类别 | | |
| dbo.Device | DFundOri | nvarchar(50) | 资金来源 | | |
| dbo.Device | DmodelName | nvarchar(50) | 规格型号 | | |
| dbo.Device | DPassUnit | nvarchar(50) | 批准单位 | | |
| dbo.Device | DTransferDate | datetime | 设备转让日期 | | |
| dbo.Device | DTransferPrice | numeric | 转让价格 | | |
| dbo.Device | DTransferUnit | nvarchar(50) | 转让单位 | | |
| dbo.Device | DUserUnitName | nvarchar(50) | 使用单位 | | |
| dbo.Device | InputTime | datetime | 入录时间 | | |
| dbo.Device | TUserName | nvarchar(50) | 信息入录员 | | |

通过设备与报修之间的关系，分析不同的业务逻辑关系，得出设备维修登记表。如表 3-2 所示。表中包过了设备在维修中会使用的业务属性。表中以故障登记编号作为主键，并且建立好索引，在使用该表进行查询的时候，可以更方便和快捷的进行数据的搜索和分析。表中，设备编号，使用单位，故障分类，备注，故障现象，开始时间，入录员为子属性，故障开始时间，入录时间的中使用 datetime 为该属性的数据类型，文字类使用 nvarchar(50)为数据类型，数字类使用 int 作为数据类型。并且设备编号作为外键，不可以为空。其他属性根据业务需求，可以设定为空。

表 3-2 维修表登记表

| 数据库表名称 | 数据项目（字段号） | 数据类型 | 描述 | 主键 | 是否可以为空 |
|---|---|---|---|---|---|
| dbo.DeviceFaultRecord | FRNo | nvarchar(50) | 故障登记编号 | 是 | 否 |
| dbo.DeviceFaultRecord | Dno | int | 设备编号 | | 否 |
| dbo.DeviceFaultRecord | DUserUnitName | nvarchar(50) | 使用单位 | | |
| dbo.DeviceFaultRecord | Fclass | nvarchar(50) | 故障分类 | | |
| dbo.DeviceFaultRecord | Fnote | nvarchar(500) | 备注 | | |
| dbo.DeviceFaultRecord | Fphenomena | nvarchar(50) | 故障现象 | | |
| dbo.DeviceFaultRecord | FStartTime | datetime | 故障开始时间 | | |
| dbo.DeviceFaultRecord | InputTime | datetime | 入录时间 | | |
| dbo.DeviceFaultRecord | TUserName | nvarchar(50) | 信息入录员 | | |

通过设备与故障诊断之间的关系，分析故障的业务逻辑关系，通过设备和故障之间的关系，分析得出故障诊断信息表。如表 3-3 所示。表中可以看出维修表中以故障登记编号作为主键，并且建立好索引，在使用该表进行查询的时候，可以更方便和快捷的进行数据的搜索和分析。表中，故障登记编号，验收人，设备编号，备注，使用单位，故障描述，故障处理意见，实际完工时间承修单位，维修方法为子属性，实际完工时间，入录时间，实际修理时间等中使用 datetime 为该属性的数据类型，文字类使用 nvarchar(50)为数据类型，数字类使用 int 作为数据类型。并且设备编号作为外键，不可以为空。其他属性根据业务需求，可以设定为空。

表 3-3 故障诊断信息表

| 数据库表名称 | 数据项目（字段号） | 数据类型 | 描述 | 主键 | 是否可以为空 |
|---|---|---|---|---|---|
| dbo.DeviceFaultDiagnose | FDNo | nvarchar(50) | 故障诊断序号 | 是 | 否 |
| dbo.DeviceFaultDiagnose | FRNo | nvarchar(50) | 故障登记编号 | 是 | 否 |
| dbo.DeviceFaultDiagnose | AuditPerson | nvarchar(50) | 验收人 | | |
| dbo.DeviceFaultDiagnose | Dno | int | 设备编号 | | |
| dbo.DeviceFaultDiagnose | DNote | nvarchar(50) | 备注 | | |
| dbo.DeviceFaultDiagnose | DUserUnitName | nvarchar(50) | 使用单位 | | |
| dbo.DeviceFaultDiagnose | FDDescription | nvarchar(50) | 故障描述 | | |
| dbo.DeviceFaultDiagnose | FDEndTime | datetime | 实际完工时间 | | |
| dbo.DeviceFaultDiagnose | FDOpinion | nvarchar(50) | 故障处理意见 | | |
| dbo.DeviceFaultDiagnose | FDStartTime | datetime | 实际修理时间 | | |
| dbo.DeviceFaultDiagnose | InputTime | datetime | 入录时间 | | |
| dbo.DeviceFaultDiagnose | PlanFDEndTime | datetime | 计划结束维修时间 | | |
| dbo.DeviceFaultDiagnose | PlanFDStartTime | datetime | 计划开始维修时间 | | |
| dbo.DeviceFaultDiagnose | PlanRepairPerson | nvarchar(50) | 计划主修人 | | |
| dbo.DeviceFaultDiagnose | PlanRepairUnit | nvarchar(50) | 承修单位 | | |
| dbo.DeviceFaultDiagnose | RepairMeasure | nvarchar(50) | 维修方法 | | |
| dbo.DeviceFaultDiagnose | RepairTools | nvarchar(50) | 维修用工具 | | |
| dbo.DeviceFaultDiagnose | TUserName | nvarchar(50) | 信息入录员 | | |

根据实际需求，以及通过用户和登陆情况的关系，才能分析得出日志信息管理表。如表 3-4 所示。在用户关系中，人员序号为主键，建立索引。用户名和密码为非空，不然用户没有用户名，或者没有密码，不符合业务逻辑，所以设定为非空。

表 3-4 日志管理表

| 数据库表名称 | 数据项目（字段号） | 数据类型 | 描述 | 主键 | 是否可以为空 |
|---|---|---|---|---|---|
| dbo.Log | Id | int | 人员序号 | 是 | |
| dbo.Log | TUserName | nvarchar(50) | 用户名 | | 否 |
| dbo.Log | UserType | nvarchar(50) | 用户类型 | | 否 |
| dbo.Log | Situation | nvarchar(50) | 登陆状态 | | |

| | | | | | |
|---|---|---|---|---|---|
| dbo.Log | Time | datetime | 登陆时间 | | |

通过用户和相关属性情况的关系，分析得出用户信息管理表。用户信息表中包过了人员的序号，用户名，密码，用户类型，以及不同模块的权限。权限值为 0，或 1，所以数据类型设定为 int 型。人员编号为主键，用户名，密码，用户类型不可为空。详细设计如表 3-5 所示。

表 3-5 用户信息管理表

| 数据库表名称 | 数据项目（字段号） | 数据类型 | 描述 | 主键 | 是否可以为空 |
|---|---|---|---|---|---|
| dbo.TrainAccount | Tid | int | 人员序号 | 是 | |
| dbo.TrainAccount | TUserName | nvarchar(50) | 用户名 | | 否 |
| dbo.TrainAccount | TUserPwd | nvarchar(50) | 密码 | | 否 |
| dbo.TrainAccount | UserType | nvarchar(50) | 用户类型 | | 否 |
| dbo.TrainAccount | RightFLog | int | 用户日志查看权限 | | |
| dbo.TrainAccount | RightFManager | int | 用户管理权限 | | |
| dbo.TrainAccount | RightFPwdChange | int | 用户密码修改权限 | | |
| dbo.TrainAccount | RightFRegistration | int | 用户注册权限 | | |

# 4 详细设计与编码实现

## 4.1 界面设计与功能实现

### 4.1.1 PC 客户端主界面

双击编译完的 exe 程序文件，软件运行开始时，即可进入登陆界面，如图 4-1 所示。根据提示输入用户名，密码。密码设定为 password char，输入的时候密码会以*号代替，而不是直接出现密码明文，给用户带来更方便的体验。UI 设计中，背景色为灰色，显得更加的稳重，输入框颜色改成与背景色匹配的灰色（157, 158, 153），登陆和退出按钮以橙色为背景色，白色文字，青春而不失稳态。界面附有虚拟键盘（osk.exe）调用系统虚拟键盘，方面部分用户，不方便使用键盘的时候，也能通过鼠标的点击使用系统。这也是对用户体验上的一种考虑，更加人性化。

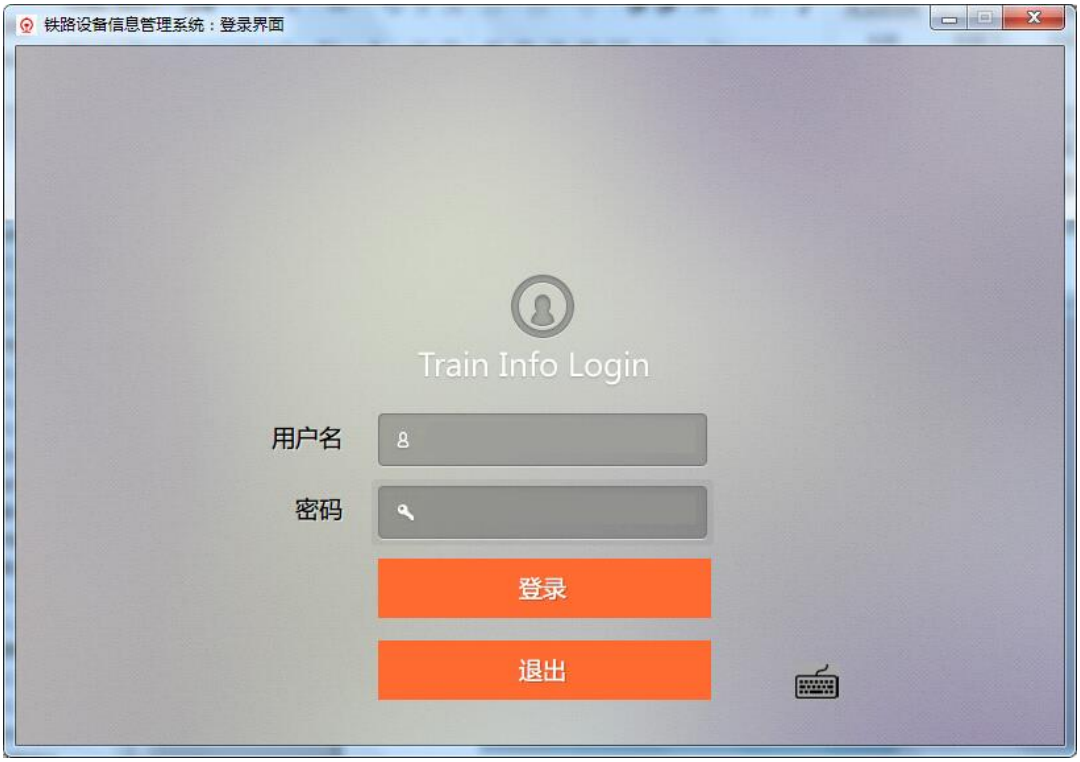


图 4-1 登陆界面

点击登陆按钮，程序将会执行如下代码，如图 4-2 所示，创建数据库的链接，将文本框里获取用户输入的字符串，与数据库里获取的数据进行对比判断，如果成功，则提示登录成功，如果密码错误，则提示密码错误，并清除用户输入的用户名和密码。

```
private void BLogin_Click(object sender, EventArgs e)
{
   //使用SqlConnection 来连接数据库
   using (SqlConnection conn = new SqlConnection(connStr))
   {
      //创建sql 查询语句
      string sql = "select TUserPwd,UserType,RightFManager,RightFRegistration,RightFPwdChange,RightFLog from TrainAccount where TUserName ='" +
         TUser.Text + "'";
      //创建 SqlCommand 执行指令
      using (SqlCommand cmd = new SqlCommand(sql, conn))
      {
         //打开数据库连接
         conn.Open();
         //使用 SqlDataReader 来 读取数据库
         using (SqlDataReader sdr = cmd.ExecuteReader())
         {
            //SqlDataReader 在数据库中为 从第1条数据开始 一条一条往下读
            if (sdr.Read())  //如果读取账户成功(文本框中的用户名在数据库中存在)
            {
               //则将第1条 密码 赋给 字符串pwd  ,并且依次往后读取 所有的密码
               //Trim()方法为移除字符串前后的空白
               //string pwd = sdr.GetString(0).Trim();
               string pwd = sdr.GetString(0).Trim();
```

图 4-2 登陆连接数据库关键代码

对用户权限进行判断，设置 4 个值，分别为用户的 4 种权限，每个权限的值为 0 或 1，当权限值为 4 个 1 的时候，即为管理员权限，获取 4 种功能的权限。当有权限值为 0 是，则该用户不能使用该功能，使用 false 对按钮控件进行屏蔽。权限判断代码如下图 4-3 所示。

```
if (UserType == "NormalUser")
{
   FRight = new int[4];
   for (int i = 0; i < FRight.Length; i++)
   {
      if (sdr.GetInt32(i + 2) == 0)     //如果数据读取器中读到数据库中的权限值为0
      {
         FRight[i] = 0;              //则赋给全局变量0  说明该功能被禁用
      }
      else if (sdr.GetInt32(i + 2) == 1) //如果 为1
      {
         FRight[i] = 1;              //赋给全局变量1   该功能可用
      }
      else
      {
         FRight[i] = 0;              //否则默认为0  该功能不可用
      }
   }
}
```

图 4-3 权限判断关键代码

登陆完成后，软件即可进入主界面，在主界面可以选择不同的功能模块，不同的模块，之间均可回到主界面，登陆后主界面如下图 4-4 所示。在主界面中，有 9 个可以选择的模块，分别是，信息管理界面，用户注册，密码修改，日志查询，重新登陆。便民服务，退出系统，关于帮助，语音助手几大模块。

在主界面，背景使用长沙铁路局的卡通动画作为背景，体现出软件的使用背景是铁路上使用，界面左上方显示当前的用户名。并且在后面有温馨提示，早上好，中午好，晚上好，等语句。给使用者带来更亲切的问候。界面底部，提示当前的用户，以及当前用户的权限，用户类型，登陆的时间。底部右侧，显示系统时间，即当前的使用者所在地区的时间。

模块选项使用的是 ToolStrip 工具条，点击添加 Button，并将 Button 属性改为 Text，这样就能实现选项条的效果。当然如果需要更好看的效果，可以设置为 Image 属性，并且将

对于类似效果的 PNG 或 ICO 图片置入，这样功能显得更加的直观，用户的感官会更加舒服些。为了完成整个软件的编写，这方便并没有详细的去精雕细琢。


图 4-4 登陆后主界面

界面中显示用户类型与登陆时间的实现，使用的是文本框。登陆时，获取系统时间，将系统时间赋值给临时变量，再将临时变量的值传到文本框的 Text 的属性当中。这样，程序在执行的时候便会出现登陆时间。同理，在数据库中，获取当前用户的类型，传到框中，即可实现上述功能。关键代码如图 4-5 所示。

```
private void First_Load(object sender, EventArgs e)
{
   //这里 新增来一个用户类型 判断  FrmMain.UserType
   //将当前登录用户名和登录时间赋给label的text属性
   //并在当前主界面窗体加载的时候 显示在label框上
   lblCurrentUser.Text = "尊敬的用户，欢迎使用铁路设备信息管理系统，当前登录的用户为: " +
      Login.Uid + " 用户类型  " + Login.UserType +" " + "登录时间为:" + Login.Time;
```

图 4-5 用户类型及登录时间显示关键代码

### 4.1.2 PC 端子界面

主界面可选择使用信息功能界面，也是软件主功能点模块，可以实现铁路设备信息的管理，实现增加，删除，修改等功能。信息管理界面如图 4-6 所示。在实现查询的功能的时候，首先点击查询按钮，然后执行实现功能关键代码如图 4-7 所示。先连接数据库，用 SQL 语句查找数据库中所需要的数据，然后将所得到的数据，传到 dvgManager 的数据表中。这样即可实现查询后的数据能够在界面中显示。


图 4-6 信息管理界面

```
private void BView_Click(object sender, EventArgs e)
{
   string sql = "select Id,Dno,Dname,Dmodel,Dprodatetime from Device";
   SqlConnection conn = new SqlConnection(connStr);
   SqlCommand cmd = new SqlCommand(sql, conn);
   System.Data.DataTable dt = new System.Data.DataTable();
   SqlDataAdapter sda = new SqlDataAdapter(cmd);
   sda.Fill(dt);
   dgvManager.DataSource = dt;
}
```

图 4-7 信息管理界面

在信息管理系统中，对于数据操作，添加必然是不可缺少的工作，添加数据模块核心代码如图 4-8 所示。首先需要判断插入的数据是否为空。即，将输入框中的文字读取出来，并且进行非空判断。如果输入的是空白的字符串，则提示输入值为空，请重新输入，同时清除输入的信息。当输入的数据符合数据类型，则打开数据库的连接，插入数据。

```
//定义一个初始值n=0, 用于判断后期是否成功插入数据
int n = 0;
string sql = "insert into Device(Dno,Dname,Dmodel,Dprodatetime) values (@Dno,@Dname,@Dmodel,@Dprodatetime)";
//判断插入的数据是否为空,如果为空,则提示重新插入!
if (txtDno.Text.Trim() == "" || txtDname.Text.Trim() == "" || txtDmodel.Text.Trim() == "" || txtDprodatetime.Text.Trim() == "")
{
   MessageBox.Show("插入数据不能为空,请按要求插入数据!");
   return;
}
//向数据库插入参数
SqlParameter[] param ={
                new SqlParameter("@Dno",txtDno.Text),
                new SqlParameter("@Dname",txtDname.Text),
                new SqlParameter("@Dmodel",txtDmodel.Text),
                new SqlParameter("@Dprodatetime",Convert.ToDateTime(txtDprodatetime.Text))

            };
SqlConnection conn = new SqlConnection(connStr);
SqlCommand cmd = new SqlCommand(sql, conn);
conn.Open();
```

图 4-8 添加数据核心代码

数据若插入成功，则显示添加成功，否则显示添加失败。然后关闭数据库连接，使用Refresh();函数对表格中的数据进行刷新。判断操作核心代码如图 4-9 所示。

```
            cmd.Parameters.AddRange(param);
            n = cmd.ExecuteNonQuery();
            if (n == 0)
            {
                MessageBox.Show("添加失败!");
                return;
            }
            else if (n > 0)
            {
                MessageBox.Show("添加成功!");
            }
            conn.Close();
            //调用refresh方法,在添加完成数据后 自动刷新 显示新数据
            Refresh();
            //this.deviceTableAdapter.Fill(this.userDataSet.Device);
        }
```

图 4-9 判断操作核心代码

删除依然是是非常重要的工作，双击选中需要删除的数据行。使用判断语句，当选中的行不为空，则使用 SQL 语句，将选中行的数据进行删除操作。然后打开数据库，对数据库内数据进行刷新。如果选中行是非正常行，则显示删除失败。删除完对内存也进行更新处理，这样新的数据会同步显示到数据表中。删除数据模块核心代码如图 4-10 所示。

```
private void BDelete_Click(object sender, EventArgs e)
{
    //使用sql删除语句
    string sql10 = "delete from Device where 1=1";
    //如果datagridview的当前行被选中
    if (dgvManager.CurrentRow.Selected)
    {
        //将sql语句 delete from Device where 1=1 + and Id = + 当前选中行的第0个单元格的号码(即Id号)
        sql10 = sql10 + "and Id=" + Convert.ToInt32(dgvManager.CurrentRow.Cells[0].Value.ToString());
    }
    int n = 0;
    SqlConnection conn10 = new SqlConnection(connStr);
    SqlCommand cmd2 = new SqlCommand(sql10, conn10);
    conn10.Open();
    n = cmd2.ExecuteNonQuery();
    if (n == 0)
    {
        MessageBox.Show("不存在的ID!");
        return;
    }
```

图 4-10 删除数据核心代码

对现有数据进行更新修改操作也是数据处理常用操作，若输入员在检查的过程中发现了数据有异常，便可以对某一项进行进行更新修改，而不用再次输入其他正确的数据，这样给用户带来更多方便。更新操作使用的的 SQL 的 update 语句，更新完成后，关闭数据库连接，以节约资源。更新数据模块核心代码如图 4-11 所示。更新成功后，有 Message 消息提示，更新失败也有 Message 消息提示更新失败。以方便告诉用户数据的状态，是否更新成功。

```
private void BSave_Click(object sender, EventArgs e)
{
   //在对数据进行修改之前 对文本框的内容做一下检查, 如果为空 则提示重新输入
   if (txtDno.Text.Trim() == "" || txtDname.Text.Trim() == "" || txtDmodel.Text.Trim() == "" || txtDprodatetime.Text.Trim() == "")
   {
      MessageBox.Show("文本框的输入不能为空!");
      return;
   }
   //使用SQL update 更新语句
   //获取文本框中输入的内容, 通过Id进行更新(Id为当前鼠标点击行的Id)
   string sqlUpdate = "update Device set Dno ='" + txtDno.Text + "',Dname ='"
    + txtDname.Text + "',Dmodel='" + txtDmodel.Text +"',Dprodatetime ='"
    + txtDprodatetime.Text + "'where Id='" + dgvManager.CurrentRow.Cells[0].Value.ToString() + "'";
   SqlConnection conn = new SqlConnection(connStr);
   SqlCommand cmdUpdate = new SqlCommand(sqlUpdate, conn);
   conn.Open();
   int n = cmdUpdate.ExecuteNonQuery();
   if (n == 0)
   {
      //提示更新失败
      MessageBox.Show("更新失败!");
      return;// 并且返回
   }
```

图 4-11 更新数据核心代码

信息安全一直是大家比较关注的方面，所以我们在设计铁路设备信息系统的时候，考虑到使用的数据量较大，为了减少黑客攻击，或者意外导致数据大面积丢失，所以，我们给系统增加了导入导出到 Excel 的功能。这个功能可以让管理者，对数据进行合理的备份。数据导出 Excel 功能演示图，如图 4-12 所示，数据导入功能演示图如图 4-13 所示。

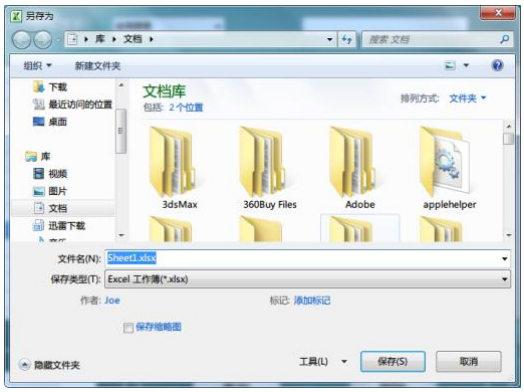


图 4-12 导出 Excel 界面

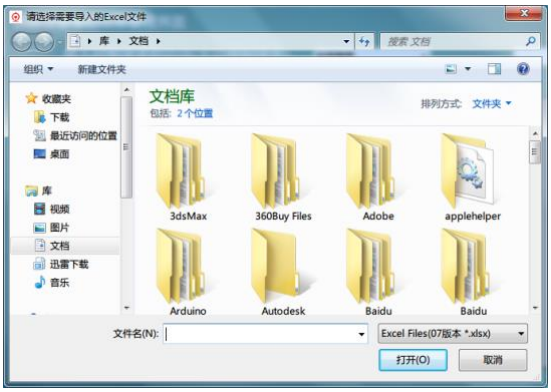


图 4-13 导入 Excel 界面

首先导入导出的功能，调用微软提供的非常强大的技术支持(即，我们可以通过使用 Microsoft.Office.Interop.Excel 程序集),使我们可以非常容易将 DataGridView 的数据输出到 Excel 表格中。主要原理：通过遍历 DataGridView 中所有的单元格，然后将里面的数据一一对应的填补到 Excel 表格中，从而完成整个数据的备份操作。操作代码如图 4-14 所示。

数据导入主要分两步：1.读取 Excel 数据：打开 Excel 文件后，调用 System.Data.OleDb 数据访问接口，读取 Excel 文件数据，然后将数据临时性存储到 DataSet 数据集中。2.将读取到的数据存入 SQLServer，遍历 DataSet 数据集 Record 表中的所有行数据。执行 SQL 数据库操作将对应表中的数据插入到 SQL Server 中;最后调用 SQL Server 查看数据的方法将导入的数据刷新并显示在 DataGridView 上。即可完成数据的导入功能。操作代码如图 4-15 所示。

```
public void ExportExcel(DataGridView dgv)
{
    try
    {
        //首先判断 DataGridView里面是否有内容 没内容则弹出提示并不再执行
        if (dgv.Rows.Count == 0)
        {
            MessageBox.Show("没数据可导出,请插入数据!");
            return;
        }
        //首先我们需要创建一个Excel对象
        Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();
        excel.Application.Workbooks.Add(true); //给Excel 对象 添加一个Excel Workbooks

        //生成Excel的 列头名称
        for (int i = 0; i < dgv.ColumnCount; i++)
        {
            excel.Cells[1, i + 1] = dgvManager.Columns[i].HeaderText.ToString();
        }
        //遍历所有行
        for (int i = 0; i < dgv.RowCount - 1; i++)
        {
            //遍历每一行的中的所有列 从而实现所有单元格的遍历
```

图 4-14 导出功能核心代码

```
//创建OleDbConn连接
OleDbConn = new OleDbConnection(connExcel);
OleDbConn.Open();
//创建 Excel 数据表
//Microsoft.Office.Interop.Excel.DataTable dtExcel = OleDbConn.GetOleDbSchemaTable(OleDbSchemaGuid.Tables, null);
System.Data.DataTable dtExcel = OleDbConn.GetOleDbSchemaTable(OleDbSchemaGuid.Tables, null);
//获取Excel表
string tableName = dtExcel.Rows[0][2].ToString().Trim();
tableName = "[" + tableName.Replace("'", "") + "]";
string queryExcel = "select Id,Dno,Dname,Dmodel,Dprodatetime  from " + tableName;
//创建数据集ds  用于后期装载 OleDbDataAdapter 中的获取的结果
DataSet ds = new DataSet();
OleDbDataAdapter oleAdapter = new OleDbDataAdapter(queryExcel, connExcel);
oleAdapter.Fill(ds, "Device");
OleDbConn.Close();
```

图 4-15 导入功能核心代码

在信息管理界面下，可以选择使用设备详细管理模块，对更多属性进行操作。由于是详细信息，所以该模块，可以对所有的属性进行修改。由于权限的设定，普通的用户是无法进入此界面。只有管理员账号才被允许使用该功能，该功能和上次一层目录下设备管理的功能基本类似，使用的技术也类似，在这里不在进行赘述。界面进行了优化，功能类的操作放在了一起，数据类的操作放在了一起，方便用户操作。在此三级目录下，可以实现跳转一级目录和二级目录，更加方便可靠。详细管理界面如图 4-16 所示。

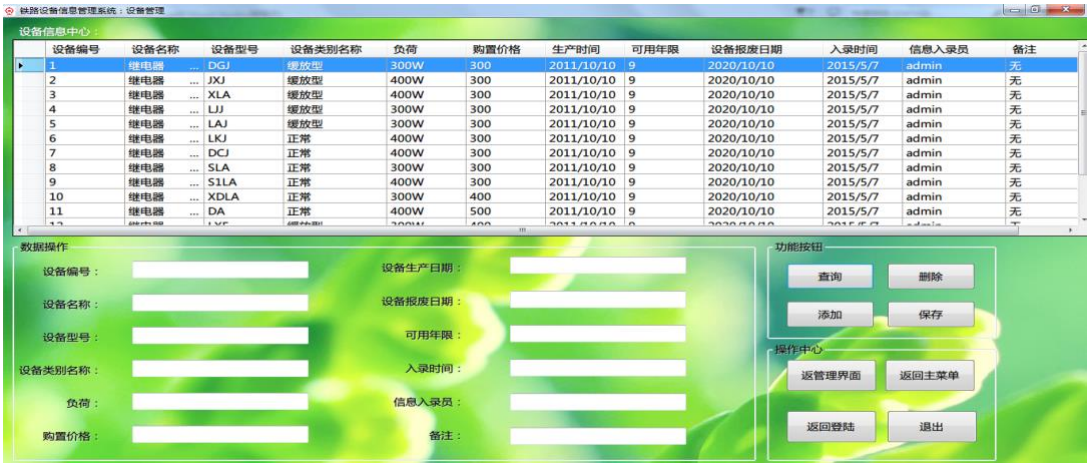


图 4-16 设备详细管理界面

用户注册模块中，可以实现用户的注册功能。首先判断用户名是否已经存在于数据库中，然后再执行 SQL select 语句，同时获取文本框中的用户名。然后通过 SQLDataReader 对数据库里的数据进行逐行读取，并判断数据库中是否已存在相同的用户名。若存在，则禁止用户注册；如果不存在，则对用户输入的密码以及密码确认分别进行判断。如果都满足密码设定条件(两次密码输入相同，并且密码长度不低于 4 位弱密码)，则执行 SQL 插入语句，将注册的用户名和密码均插入至数据库中，并提示注册成功。用户注册窗体 TRegistration 中增加 2 个选项，用于注册的时候进行用户分类注册。一个是管理员，一个是普通用户，当选择管理员是，默认赋给最高权限，普通用户默认最低权限。用户注册界面如图 4-17 所示。实现功能代码如图 4-18 所示。


图 4-17 用户注册界面

```
if (sdr.Read())
{
   //如果读到相同的用户名 则提示 用户名已存在
   lblUserName.Text = "用户名已存在,请重新输入!";
   TUserName.Text = "";
   TPwd.Text = "";
   TrePwd.Text = "";
   return;
}
//如果用户名不存在 则判断下用户名密码是否为空
else if (TUserName.Text.Trim() == "")
{
   //如果用户名 不为空, 将提示语句赋给label框 并在窗体上
   lblUserName.Text = "用户名不能为空!";
}
   //判断密码是否为空
else if (TPwd.Text.Trim() == "")
{
   //同理
   lblPwd.Text = "密码不能为空!";
   //同时清除 用户名 提示框的内容
   lblUserName.Text = "";
   lblPwd.Text = "";
   lblrePwd.Text = "";
}
//判断再次输入密码是否为空
else if (TrePwd.Text.Trim() == "")
{
   //同理
   lblrePwd.Text = "验证密码不能为空!";
   //同时清除 用户名 提示框 和 第一次密码输入提示框的内容
   lblUserName.Text = "";
   lblPwd.Text = "";
   lblrePwd.Text = "";

}
//判断2次密码输入是否相同
```

图 4-18 用户注册实现核心代码

在注册窗体中创建一个全局变 Fright，用来设定用户注册成功后的权限的值，权限值设定为 1,最高权限；权限值为 0 时，最低权限。用户类型赋值代码如图 4-19 所示。

```
if (RAdministrator.Checked)  //当管理员的radiobutton被点击后
{
   uType = "Administrator";//传给 uType 一个管理员
   Uright="1";
}
else if (RNormalUser.Checked)  //同理
{
   uType = "NormalUser";
   Uright = "0";
}
else                        //若不点击 则默认为普通用户注册
{
   uType = "NormalUser";
   Uright = "0";
}
//sql 插入语句  我们新增了 一列 UserType
//使用sql 数据插入语句
string sqlInsert = "insert into TrainAccount(TUserName,TUserPwd,
```

图 4-19 用户类型赋值代码

在登录窗体类中创建一个全局变量 FRight, 用来获取用户成功登录后的功能权限值，权限值等于 1 时，功能可用；权限值等于 0 或者其它时，功能不可用。当普通用户成功登录后，在主窗体 Main 中可以获取这个全局变量，并且同时将相应的功能按钮的 Enable 属性设置为 true 或者 false(这里管理员默认是拥有所有功能权限,系统只判断普通用户的功能权限)。

SQLHelper 是微软提供的.NET Framework 的数据库操作组件；用于帮助程序简化掉那些重复的数据库操作语句，包括 SQLConnection，SqlCommand，SqlDataReader 等等。SQLHelper 封装过后通常是只需要给数据库操作方法传入一些参数如数据库连接字符串，SQL 查询语句，SQL 参数等，便可以访问数据库。本系统采用了改功能，以方便数据库的访问操作。

这里创建了一个 TUserManager 窗体，对普通用户权限进行管理(仅限于管理员使用)，用于管理员创建普通用户以及对普通用户账户信息和分配权限等。因为业务的需求，管理员本身信息只能被查看，不能被做任何修改。这里使用了 CheckBox 控件，通过判断 CheckBox 的 Checked 属性是否被点中，来给普通用户赋予相应的功能权限。被点中则赋值 1，否则默认为 0。然后通过执行 SQL 数据操作将权限值 1 或者 0，存入数据库。不同的用户分配不同的权限，对用户的操作有相应的权限分配界面，用户权限管理界面，如图 4-20 所示。

为了方便对大量用户数据进行查找，增加了搜索的功能，更加方便的将适配用户在海量数据中查找出来。在界面设计中，对不同功能的操作进行分类，更加直观的展现出相应的功能。此功能为三级功能，仍然可以通过按钮跳转至第一级或者二级功能界面中。界面表中直观的展现了各个用户的详细信息，以及各个用户的相应的权限。此功能仅针对管理员赋予普通用户的操作。无法对管理员的权限进行操作，当选中是管理员权限时，checkbox 变成灰色，不允许更改操作。详细实现核心代码如图 4-21 所示。

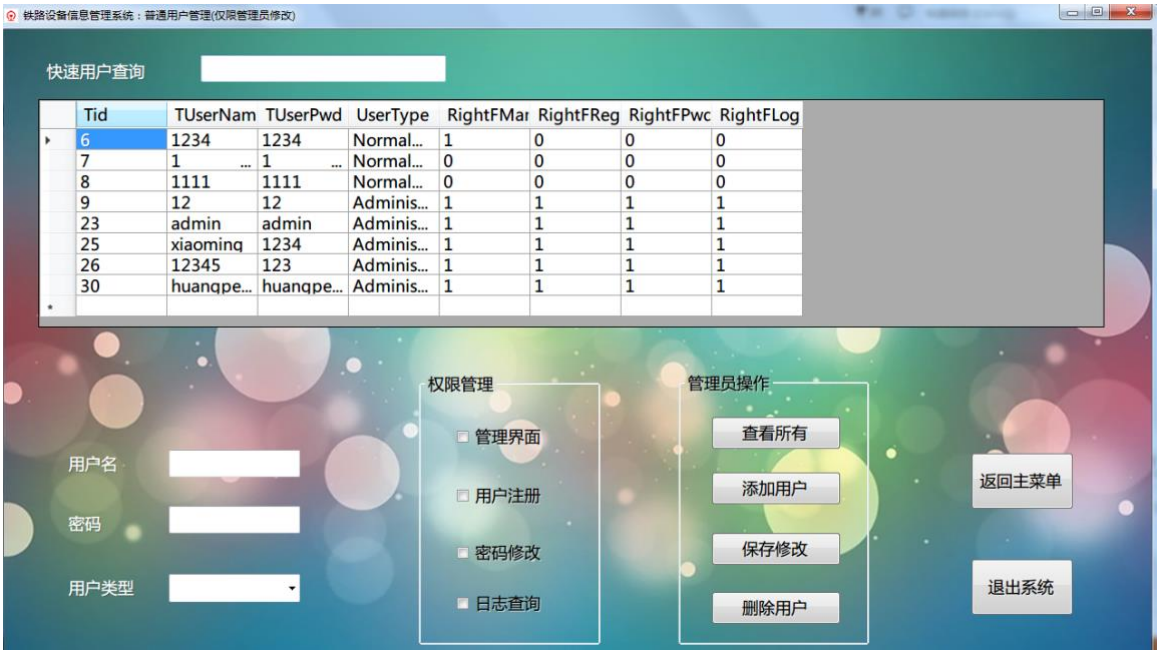


图 4-20 用户权限管理界面

```
if (strUType == "NormalUser")
{
    //如果先前点击了 DataGridView里的 管理员行内容 checkbox 会保留之前的点击状态 我们需要先前的点击状态清除
    chkManager.Checked = true;
    chkRegist.Checked = true;
    chkManager.Checked = true;
    chkLog.Checked = true;

    //获取当前DataGridView被点中行内容中的用户权限值
    rightFManager = Convert.ToInt32(dvgUserManger.Rows[e.RowIndex].Cells["RightFManager"].Value);
    rightFRegistration = Convert.ToInt32(dvgUserManger.Rows[e.RowIndex].Cells["RightFRegistration"].Value);
    rightFPwdChange = Convert.ToInt32(dvgUserManger.Rows[e.RowIndex].Cells["RightFPwdChange"].Value);
    rightFLog = Convert.ToInt32(dvgUserManger.Rows[e.RowIndex].Cells["RightFLog"].Value);

    //同时判断获取的权限值 并相应的选中Checkbox
    if (rightFManager == 1)
        chkManager.Checked = true;
    else
        chkManager.Checked = false;
    if (rightFRegistration == 1)
        chkRegist.Checked = true;
    else
        chkRegist.Checked = false;
    if (rightFPwdChange == 1)
        chkPwdChange.Checked = true;
```

图 4-21 用户权限管理实现核心代码

在用户管理中，密码修改是必不可少的部分，密码修改模块解决用户的密码修改问题，密码修改界面。原理很简单，系统获取用户输入的用户名，执行 SQL 中 select 语句从数据库中获取该用户名下的密码，再与用户输入的旧密码进行匹配。若相同则允许输入新密码，然后执行 Update 语句将新密码保存到数据库中；若不相同则提示密码错误，清空文本框数据，要求用户重新再试。密码修改界面如图 4-22 所示。界面设计上较为简洁。通过界面显示较容易判断为，用户密码的相关操作。关键代码如图 4-23 所示。


图 4-22 密码修改界面

```
string sqlUpdate = "update TrainAccount set TUserPwd ='" + TrenewPwd.Text +
"' where TUserName ='" + Tuser.Text + "'";
//new一个数据库更新指令
SqlCommand cmdUp = new SqlCommand(sqlUpdate, conn);
// 如果cmdUp获取的行数为0 则说明系更新指令出现异常
if (cmdUp.ExecuteNonQuery() == 0)
{
    //系统报错
    MessageBox.Show("未知错误!请检查系统完整性！");
    return;
}
else
{
    //否则 更新成功
    Tuser.Text = "";
    ToldPwd.Text = "";
    TnewPwd.Text = "";
    TrenewPwd.Text = "";
    MessageBox.Show("恭喜你!密码修改成功!");
```

图 4-23 密码修改核心代码

方便大家出行，出门前可以用便民服务模块查询天气情况，音乐播放的操作。此界面使用 WebBrowser 空间，访问预设好的本地网页。便民服务界面，如图 4-24 所示。调用本地网页源码，如图 4-25 所示。


图 4-24 便民服务界面

```
<html>

<h1>当地天气情况</h1><iframe src="http://cache.xixik.com.cn/2/nanchang/"
width="890" height="70" frameborder="0" marginwidth="0" marginheight="0"
scrolling="no"></iframe>
<h1>我的音乐欣赏</h1>
<iframe frameborder="no" border="0" marginwidth="0" marginheight="0" width=330
height=450 src="http://music.163.com/outchain/player?
type=0&id=75277947&auto=1&height=430"></iframe>

</html>
```

图 4-25 便民服务本地网页源码

关于软件的一些信息开发时间，开发人员以及版本信息等，关于界面设计，如图 4-26 所示。文本滚动设计，使用 Timer 定时器，每隔一定时间，文字滚动一定距离。使动态效果显示更加平稳。文字滚动设计源码如图 4-27 所示。


图 4-26 关于界面

```
private void timer1_Tick(object sender, EventArgs e)
{
    lblWord.Top = lblWord.Top - 3;
    if (lblWord.Top < 0)
    {
        lblWord.Top = this.Height;
    }
}
```

图 4-27 文字滚动效果设计代码

语音识别技术也日渐成熟，所以在系统中也引入了语音识别服务。在图 4-28 界面中使用的是微软提供的 Speech 语音技术，下载安装好微软的语音识别开发包，在开发资源中引用微软预留的 API，通过调用其中的 Speak 功能，可以实现 TTS，文本转语音的服务。在图 4-29 中，调用的是科大讯飞的在线语音识别服务，可以将语音识别为文字，希望在以后的版本中可以实现对话的功能。

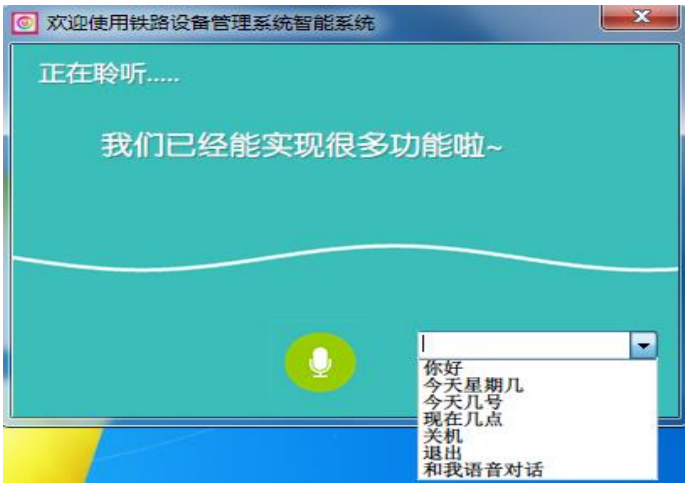


图 4-28 微软 TTS 语音服务


图 4-29 科大讯飞语音识别服务

为了更友好的提示用户退出界面，独立窗体退出提示，以免误操作退出系统。用户可以被提醒，是继续退出，还是回到主界面继续操作。退出提示界面设计醒目感叹号为主题，用户感知能更加强烈。演示如图 4-30 所示。

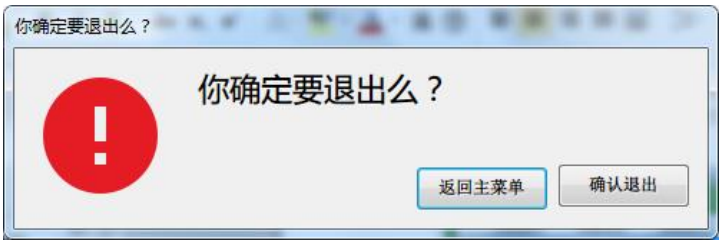


图 4-30 退出提示界面

### 4.1.3 Web 端界面

登陆 Web 端口首页时，打开浏览器，输入 localhost，跳转至登陆界面，如图 4-31 所示。登陆界面设计思路，以简洁为主，背景使用深浅色背景轮换。用户名，密码输入框均

设置透明，且有相应提升。输入用户名和密码后，使用 post 功能，将 HTML 静态网页数据 post 至 aspx.login 中的动态网页中去认证。动态网页连接数据库后，对用户进行判断，成功后，提示用户登陆成功，并在 3 秒后自动跳转至主界面。登录失败立即跳转至登陆界面。


图 4-31Web 登陆界面

登陆验证成功后，即可进入系统主界面，同样可以选择更多的功能，但是功能相比 PC 端口要简单一些，Logo 选用 PNG 图片。首页一共六个功能外链。分别是主页，管理界面，用户管理，日志查询，关于我们，退出功能。Web 端主演示界面如图 4-32 所示。


图 4-32 Web 主界面

跳转到信息管理子网页下，可以方便查询到设备详细情况，同样，也是先连接好数据库，然后对数据库进行查询，得出查询结果，使用表单显示在页面上。点击更多记录。对更多属性进行查询操作，可以点击查看更多记录，Web 信息管理界面如图 4-33 所示。用户管理界面如图 4-34 所示，可以实现用户密码修改功能。


图 4-33 Web 信息管理界面


图 4-34 Web 用户管理界面

### 4.1.3 其他细节设计

PC 端软件，还有很多细节可以介绍下。软件的 ICON 经过自己的设计，以突显软件的特性，即铁路上使用的软件。如图 4-35 所示。

图 4-35 软件图标设计

在主界面下，有一个温馨提示的功能，给使用者一个温馨的提醒。比如当时间处于早上 6 点-中午 12 点的时候，提示，早上好；中午 12 点-下午 6 点的时候，提示，下午好；中午 6 点-晚上 0 点的时候，提示晚上好；在凌晨 0-6 点的时候，提示，深夜了,该休息了！给使用者宾至如归的感觉。

## 4.2 系统主要功能模块介绍

前面已经介绍过，系统内分为十一大模块，分别是信息管理模块，用户注册模块，用户权限管理模块，密码修改模块，日志查询模块，便民服务模块，设备管理模块，采购管理模块，维修管理模块，物资管理模块，语音服务模块。不同的模块各司其职，比如用户

注册模块，针对用户的密码，权限进行管理；而设备管理模块，针对设备的不同属性进行管理。所有的模块对完善整个系统，有不可或缺的作用。语音模块可以实现基本的沟通功能，还有其他更多的功能还在不断的完善中。

## 4.3 系统主要功能模块设计

数据流程图是单从数据流动过程来考查实际业务的数据处理模式,主要包括对信息的流动、传递、处理、存储等的分析[9][10]

### 4.3.1 登录模块流程

登陆模块的功能，主要对使用者进行身份验证，系统非法用户不能登陆管理界面。用户再点击登录按钮之后，登录功能模块，调用数据库中用户表，进行比对。只有合法的用户，密码正确后，才可以正常跳转到主界面中。当用户输入用户名，或者密码有误时候，提示密码或者用户名有误，并引导用户重新再次输入，清空之前输入框中的数据。铁路设备信息系统软件中登录模块流程图，如图 4-36 所示。

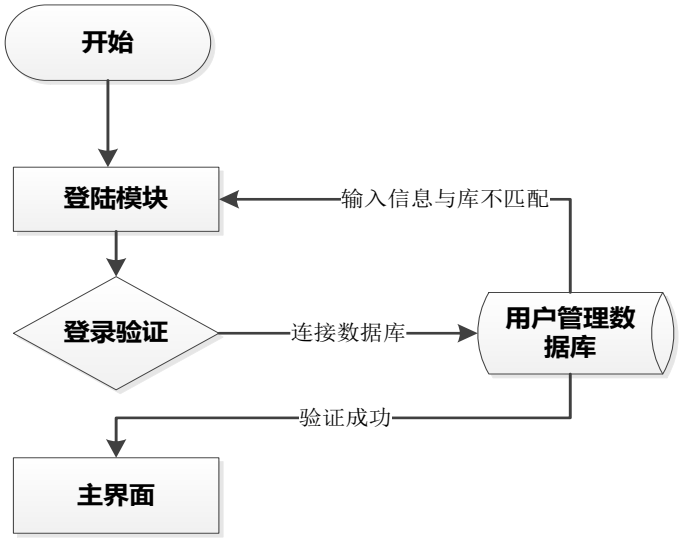


**图 4-36 登陆流程**

### 4.3.2 设备管理流程

功能：在设备信息管理模块中，对设备当中重要属性进行统计。

设备信息查询：本模块对所有的设备的属性情况进行全面细致的查询，包涵设备的各项属性，比如型号，生产日期，等等。对设备数据新增，去除，更新都可以在软件中轻松完成。设备管理流程如图 4-37 所示。

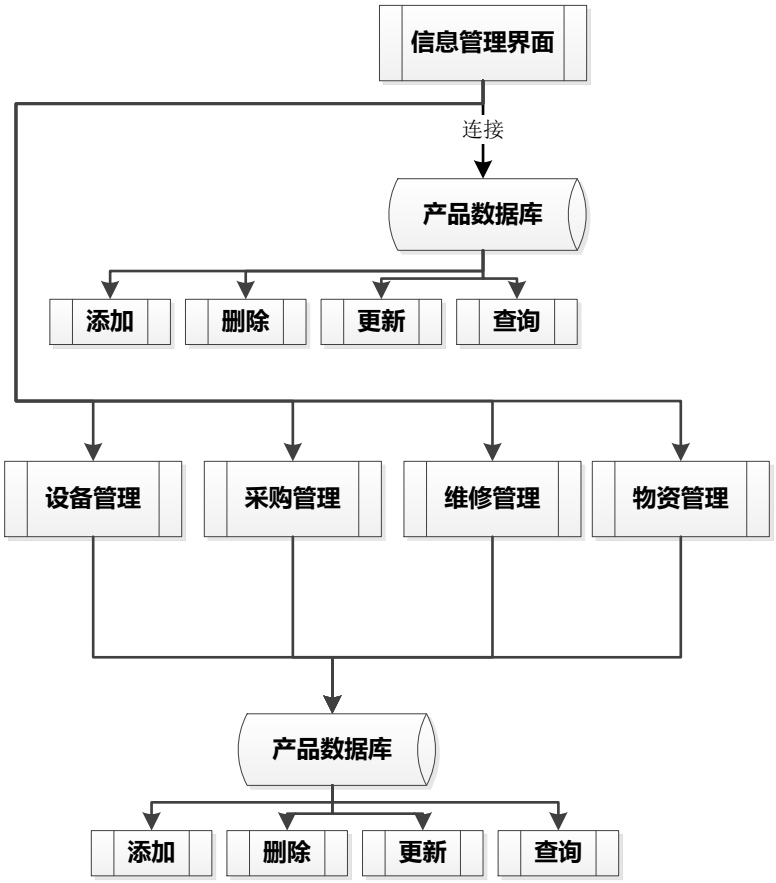


**图 4-37 设备管理系统流程**

### 4.3.3 用户管理流程

功能：模块主要针对用户方面的操作，管理用户注册，对用户所具有权限进行分配。

用户模块：本模块对所有的用户的权限有详细的记录，管理员有最高管理权限，其他普通管理员则没有。具有最高管理员，可以支配其他的管理员权限，对设备数据可以进行新增，去除，更新操作。用户管理流程如图 4-38 所示。

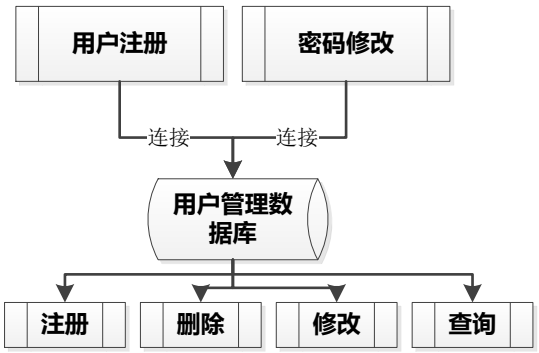


**图 4-38 用户管理流程**

### 4.3.4 用户日志查询流程

功能：模块主要实现了对用户的登陆，退出情况进行查询。

日志查询模块：本模块对所有的用户的登陆情况有详细的记录，能对登陆设备类型进

行查询，如 PC 端，Web 端，日志查询流程如图 4-39 所示。

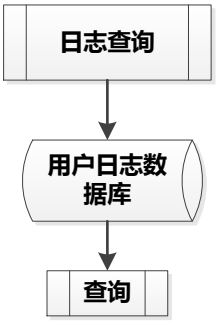


**图 4-39 用户日志查询流程**

### 4.3.5 便民服务流程

功能：模块主要实现了天气预报功能，通过调用天气通 API 实现，天气查询。

日志查询模块：本模块对所有用户开放的查询权限，方便使用者便捷查询天气，便民服务流程如图 4-40 所示。

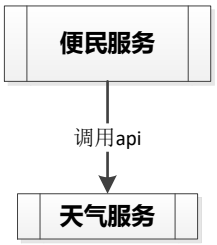


**图 4-40 便民服务流程**

### 4.3.6 语音识别流程

功能：模块主要实现了语音识别功能，通过调用微软的语音识别接口，实现 TTS 功能，调用科大讯飞的识别技术，实现语音的识别服务。

语音识别模块：本模块对所有用户开放权限，用户均可体验语音服务。功能仍需要不断完善，以提高识别效率和准确度。语音识别服务流程如图 4-41 所示。

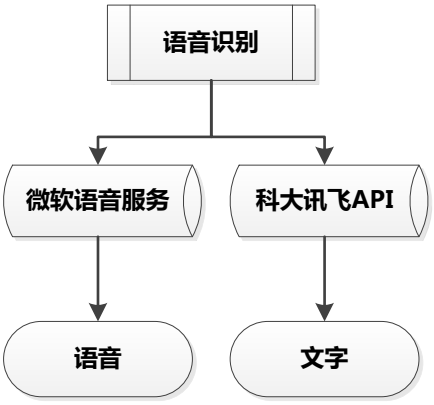


**图 4-41 语音识别服务流程**

## 4.4 模块总设计

### 4.4.1 登陆主界面模块算法描述

主界面登陆，首先要判断用户是否是合法用户，是系统用户后，获取到用户的权限，并且不同的用户能获取到的软件的权限在主界面中的提示。然后根据用户的鼠标点击，相应不同的操作。例如，用户注册操作，用户点击用户注册按钮，即可跳转至用户注册模块，程序相应当前操作，主界面即刻隐藏。操作完后，用户可以自行选择退出，或者退回到主界面，执行其他操作均可。所有界面下均可回到主界面，且退出操作均会调用日志功能，记录退出的时间。

### 4.4.2 系统整体优化解决方案

在退出模块使用的时候，编写了统一的退出模块，其他的功能模块，均可调用此退出模块，首先能解决某一模块重复使用退出语句的麻烦和不便捷性。此外，使用统一调用的退出模块，能够更加方便的进行修改，和其他的测试操作。统一性编码的使用在代码的编写当中有不可比拟的优越性，给编程工作减少重复代码的编写，增加代码的可读性。

为了减少对数据库的压力，在每次使用完数据库后，都对数据库进行关闭，以减少资源的消耗，加快软件的相应速度。网页端对显示图片进行了定向优化，图片的码率大幅降低，增加了传输的顺畅度，加载速度有一定的提升。在传输协议上，并没有使用 Https 传输方案，因为网站暂时不需要特别的加密措施保障数据传输过程中的安全性。若要使用 Https 加密传输，会降低传输效率，消耗更多资源。与获得的效益相比，并不是最大的优化效率。为了适配移动端（手机，平板）在网页的编写中，尽量减少 px 的描述，尽量使用 em，百分比的描述，这样能较为方便的适配到移动设备浏览器。否则如果要同时适配不同设备的浏览器，将会付出更大的代价，这对开发者将是致命的工作量。

系统以 PC 端为主，部分敏感功能，例如注册功能，并没有对在网页端开放，仅在 PC 端才能完成注册的功能。首先是考虑到安全的问题，第二，考虑到系统在网页端主要实现的是信息的查询功能，所以并没有开放此权限。

### 4.4.2 模块程序总流程图

系统的整体设计为了更好的展示系统的整体概况，总流程图如图 4-42 模块程序总流程所示。总流程图可以很直观的看到整个系统的设计思路，以及系统的业务逻辑。数据库的使用在本系统中尤为重要。信息管理系统在数据的处理也是非常的关键。软件的整体的开发难度并不大，但是在短时间内完成大量系统属性，以及多客户端的适配任务，重视用户的 UI 交互体验，还是有一定的难度。

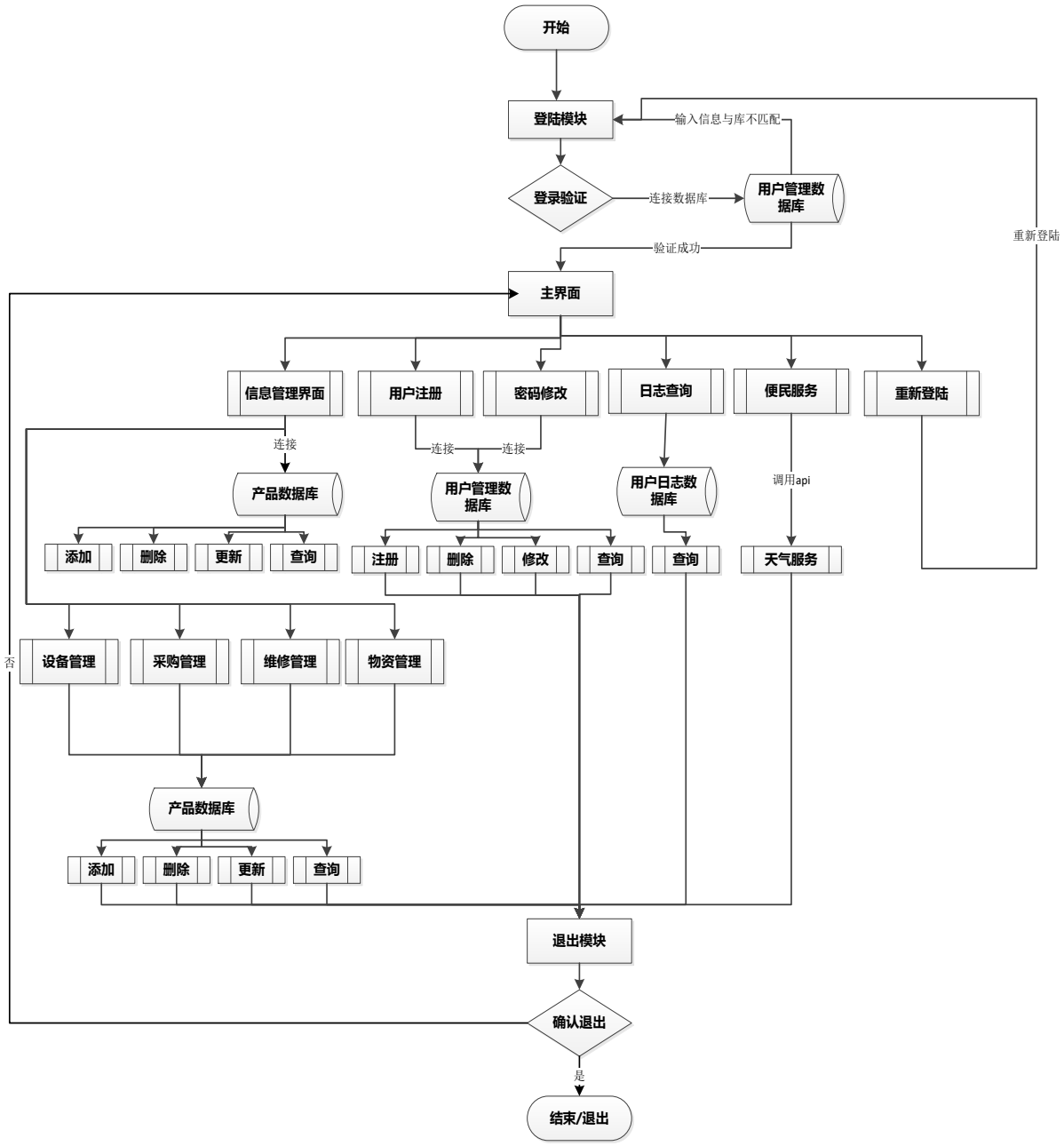


**图 4-42 模块程序总流程**

# 5 测试

## 5.1 系统测试步骤

### 5.1.1 单元测试

**PC 端登陆界面测试**

运行程序，在框内输入用户名以及密码，进行测试用户名以及密码的是否正确，输入如下测试用例，如下表 5-1 所示：

**表 5-1 登录测试用例表**

| 组别 | 用户名 | 密码 | 类型 |
|---|---|---|---|
| 1 | admin | admin | 管理员 |
| 2 | 12 | 12 | 管理员 |
| 3 | Test | Test | 管理员 |
| 4 | 1234 | （NULL） | 普通用户 |

数据库预先置入 1,2,3 组数据，程序进入登陆界面，当输入 admin，和 12 时候，验证成功，转向登陆主界面，获取管理员权限，可以进行对应权限的操作；当输入为 Test 的普通用户时，验证成功，转向登陆主界面，获取普通用户权限，但是仅仅有普通用户的查看权限。

当仅输入空白的用户名或空白的密码时，不输入具体用户名或具体密码，系统则提示用户并没有输入用户名或密码，同时清空所有的输入框，要求重新输入用户名。

**网页端登陆界面测试**

数据库预先置入相同数据，进入登陆界面，当输入 admin，和 12 时候，验证成功，转向主页面，获取管理员权限，可以进行对应权限的操作；当输入为 Test 的普通用户时，验证成功，转向主页面，获取普通用户权限，但是仅仅有普通用户的查看权限。

当在网页端，只输入空白的用户名或空白的密码时，而不输入具体用户名或具体密码时，系统则提示用户并没有输入用户名或密码，重新跳转登陆界面。

**信息管理界面测试**

登陆后选择信息管理界面，测试各个按钮的功能是否正常，对数据操作是否符合程序流程以及业务的逻辑流程，如果不符合则回到代码，使用单步调试，测试数据流程是否符合规范，最终得出测试结果：按钮测试正常，数据处理正常，搜索正常。

### 5.1.2 确认测试

系统最终完成以后，对所有的模块进行联调联试，查找出有冲突的模块，进行修改，对各个界面的 UI 进行测试，保证用户交互正常，视觉观感正常，数据库调用正常。

## 5.2 系统测试

### 5.2.1 界面测试

软件的界面测试，主要测试如下方面的功能：

窗口是否能在系统中打开，打开后是否能正常显示，有无乱码错误。背景以色调有无错位等现象。窗口中的表单的大小样式是否符合规范。在提交用户数据，例如，用户名，密码时候是否正常，密码是否以密文形式展现而不是明文。窗口的位置是不是正常，或者出现残缺情况。窗口元素，最大化，关闭按钮是否都在，测试窗口拖动是否出现异常等情况。

### 5.2.2 功能测试

软件的连接测试，主要的测试如下方面进行：

在点击提交按钮之后，数据是否能按照正常流程。提交到软件处理，连接到数据库，在比对完数据库后，对用户操作进行相应。如，跳转提示，弹出新窗口等提示。

# 6 总结与展望

## 6.1 设计工作总结

为了解决目前铁路信息化所面临的问题,铁路部门提出建设信息管理系统,然而铁路信息管理系统的建设会带来新的信息安全问题,本文着眼于铁路信息管理系统建设安全和使用安全,运用便捷、安全等角度大胆提出铁路信息管理系统的设计。以下是本文的主要工作和结论:

1. 铁路信息管理系统给用户提供便捷的操作及最大的信息量可供查询。

2. 铁路信息管理系统的安全性强，可操作性强、技术保障、经济性合理

3. 铁路信息管理系统给铁路部门的管理带来便捷，提高工作效率，真正做到全心全意为人民服务的宗旨。

4．软件的成功运行，也预示着铁路信息管理系统在市场推广性方面还是很有前景的。

## 6.2 未来工作展望

随着我国铁路的进一步深化改革以及信息技术的飞速发展,我们对未来铁路信息管理系统有以下展望：

技术飞速变革，当前技术较发达国家还较为落后，有待时时刻刻更新换代。

后续还需根据用户的反馈多次修改这一铁路信息管理系统来不断地更适应用户的使用习惯和需求。

在当下网络稳定和安全的状态下，加强信息管理系统的安全防护，最大可能地维护铁路信息不受侵犯。

随着我国铁路的进一步深化改革以及信息技术的飞速发展,我们对未来铁路信息管理系统有以下展望：

技术飞速变革，当前技术较发达国家还较为落后，有待时时刻刻更新换代。

后续还需根据用户的反馈多次修改这一铁路信息管理系统来不断地更适应用户的使用习惯和需求。

在当下网络稳定和安全的状态下，加强信息管理系统的安全防护，最大可能地维护铁路信息不受侵犯。

# 谢 辞

白驹过隙，四年的大学阶段弹指间就已飞逝，留给我无尽的回忆。在大学的尾声，我在导师魏波老师的指导和帮助下完成了我的毕业设计论文，由此，我由衷地感谢魏老师的悉心指点以及多次对我论文的雅正。本论文是基于我在鹰潭铁路实习的经历而编撰的。因此，我由衷地感谢鹰潭铁路站的我那亲切的领导和同事们。正是你们的不厌其烦的启发，使我从一个只用理论武装自己的到学生变得开始学会积累实践经验，逐步走入社会，适应社会。

接下来的大学四年，那些朝夕相处的同学和老师们，你们给我带来的帮助和快乐，我这辈子都会珍藏在我心灵最深处，永远保存起来。可怜天下父母心，儿女总归是父母的心头肉，我必须对我的父母说一声：爸妈，你们辛苦了，儿子长大了，懂事了，你们含辛茹苦的培育，我永远记在心间，我一定勇往直前，不断努力，创造佳绩！

# 参考文献

[1] 李德坤. 车站信号设备管理系统的设计与实现[D]. 大连理工大学, 2008.

[2]Shahram Gilaninia,Seyyed Javad Mousavian, Orang Taheri, Hamid Nikzad, HodaMousavi , Fatemeh Zadbagher Seighalani. Information Security Management onperformance of Information Systems Management [J], Journal of Basic and AppliedScientific Research, 2(3)2582-2588, 2012.

[3]Daniel Mellado,David G. Rosado. An Overview of Current Information Systems Security Challenges and Innovations[J]. Journal of Universal Computer Science, vol.18,no. 12 (2012),1598-1607.

[4]韦银星,张申生,曹健.UML 类图的形式化及分析[J]. 计算机工程与应用, 2002, 38(10):5-7. DOI:10.3321/j.issn:1002-8331.2002.10.002.

[5]郑人杰等.实用软件工程(第二版)[M].北京:清华大学出版社,2004.

[6]李平,赵丽华,马丽.管理信息系统[M].北京:清华大学出版社,北京交通大学出版社,2006.

[7] LeszekA.Maciaszek, Bruc Lee Liong.实用软件工程[M].北京:机械工业出版社,2007.

[8]薛华成.管理信息系统(第四版)[M].北京:清华大学山版社,2003.

[9]邝孔武,王晓敏总系统分析与设计(第 3 版)[M].北京:清华大学出版社,2006.

[10]王珊,萨师煊.数据库系统概论(第四版)[M].北京:高等教育出版社,2006.

# 附录 A 外文翻译—原文部分

From: Programmer

Data Management

Why Data Management?

We have already considered hardware and software in some detail. In this chapter, we turn our attention to a third basic computer resource, Data. Many computer applications require that data be stored for subsequent processing. Simply storing the data is not enough, however. A typical computer system, even a small one, can have dozens of disks and tapes, each holding data for dozens of different applications. For any given application, one and only one set of data will do. We must be able to store, locate, and retrieve the specific data needed by a given program. That is the concern of data management.

Accessing Data

Imagine a single diskette containing several programs. For a particular application, only one of those programs will do. How is a given program selected, loaded, and executed? In chapter 6, we learned that the operating system, responding to a user's command, reads the disk's index, searches it for the requested program name, extracts the program's track and sector address, and issues primitive commands to read it into main memory. Later, following a RUN command, the program is given control of the processor.

Accessing data presents a similar problem. A single diskette can hold data for several different applications. For a given application, one and only one set of data will do, and finding the right data is much like finding the right program. There are differences between accessing programs and accessing data, however. When a program is needed, all its instructions must be loaded into memory. Data, on the other hand, are typically processed selectively, a few elements at a time. Thus, it is not enough merely to locate the data; we must be able to distinguish the individual data elements, too.

Data Structures

The key to retrieving data is remembering where they are stored. If the data elements are stored according to a consistent and well understood structure, it is possible to retrieve them by remembering that structure. The simplest data structure is a list. For example, data for a program that computes an average might be stored as a series of numbers separated by commas . The commas distinguish the individual data elements.

Most programming languages support a more complex data structure called an array . Each array element can hold one data value. Each element is assigned a unique identifying number of numbers, and individual data elements can be inserted, extracted, or manipulated by referencing those numbers.Once an array has been filled, it can be written to disk, tape, or any other

secondary medium, and later read back into memory for processing. Consider a program that generates name and address labels. For each label, we need a name, a street address, a city, a state, and a zip code. If we needed only a few labels we might store the data in a list, but separating the elements would soon become tedious. An option is to set up an array of names and addresses, with each row holding the data for a single label. The only problem is that the entire array must be in main memory before the individual elements can be accessed, and main memory space is limited. even with an array, we could generate relatively few labels.

A better solution is to organize the data as a file . All computer data begin as patterns of bits. On a file, the bits are grouped to form characters. Groups of characters, in turn, form meaningful data elements called fields. A group of related fields is a record; the file is a set of related records. For example, in a name and address file, and individual's name is a field. Each record holds a complete set of data for a single individual (a name, a street address, and so on). The file consists of all the records.

The data in a file are processed record by record. Normally, the file is stored on a secondary medium such as disk. Programs are written to read a record, process its fields, generate the appropriate output, and then read and process another record. Because only one record is in main memory at a time, very little memory is needed. Because many records can be stored on a single disk, a great deal of data can be processed in this limited space.

Locating Files

Imagine a file stored on disk. The first step in accessing its data is finding the rile. The task is much like finding a program, but there are differences. Following a command such as LOAD or RUN, programs are loaded by the operating system. Data, on the other hand, are processed by application programs, in the context of a program's logic. Typically, just before the data are required, the program asks the operating system to open the file. Each file has a name; the open logic reads the disk index, searches it by name, and finds the address of the first record in the file.

Locating Records

Once a file has been located, the process of accessing its records can begin. When a program needs input data, it reads a record; when it is read go output results, it writes a record. Note that these instructions deal with selected records, not with the entire file. We open files. We read and write records.

Let's examine the data accessing process more closely. A programmer views data logically, requesting the next record, or the name and address for a particular customer. The data are stored on a secondary medium such as disk. To access a record physically, the disk drive must be given a set of primitive commands: seeks, reads, and writes. The programmer thinks in terms of logical I/O. The external device stores and retrieves physical sectors; it "thinks" in terms of physical I/O.

There must be a mechanism for translating the programmer's logical requests to the appropriate physical commands. On small computers, much of the logic is found in the operation system's input/output control system; on larger machines, access methods are used. Increasingly, the programmer's logical data request is translated to physical form by a database management system.

The Relative Record Concept

How does software, be it operating system, access method, or database software, find specific records in a file? The key to many storage and retrieval techniques is the relative record number. Imagine a string of 100 records. Number the first one 0, the second 1, the third 2, and so on. The numbers indicate a given record's position relative to the first record in the file. The file's first record (relative record 0) is at "start of file plus 0"; its second record is at "start of file plus 1," and do on.

Now, store the records on disk ; to keep our initial example simple, we'll store one per sector. Number the sectors relative to the start of the file-0,1,2, and so on. Note that the relative record number, a logical concept, and the relative sector number, it is possible to compute a relative sector number. Give a relative sector number it is possible to compute a physical address on disk.

Assume a file begins at track 30, sector 0, and that one logical record s stored in each sector. As Fig. 8.7 shows, relative record 0 is stored at track 30, sector0; relative record 1 is at track 30, sector1; and so on. Where is relative record 10? It must be stored at track 30, sector 10. In our example, the relative record number indicates how many sectors away from the beginning of the file the record is stored. Thus, we can compute the physical location of any record by adding its relative record number to the start-of-file address (which, remember, was extracted from the disk's index when the file was opened). The file starts at track 30, sector 0. Retative record 10 is stored 10 sectors away, at track 30, sector 10. To read record 10. We have translated a logical data request to specific physical commands.

We might complicate matters by storing two or more logical records in each sector, or by creating a file extending over two or more tracks. While we won't discuss the details, in either case it is still possible to develop a simple algorithm to compute a record's physical location, given its relative record number. Many different algorithms are used. Some allow records to be stored or retrieved sequentially. Others allow individual records to be accessed in random order. Let's examine a few common data access techniques.

Access Methods

Imagine preparing meeting announcements for a club. You need a set of mailing labels, and each member's name and address is recorded on an index card. Probably the easiest way to generate the labels is to copy the data from the first card, turn to the second card and copy it, and

so on, processing the records sequentially, from the beginning of the file to the end.

Magazine publishers face the same problem with each new issue, but need mailing labels for tens of thousands of subscribers. Rather than using index cards, they store customer data on disk or mag.NETic tape, one record per subscriber. The easiest way to ensure that all labels are generated is to process the records in the order in which they are stored, proceeding sequentially from the first record in the file to the last. To simplify handing, the records might be presorted by zip code or a mailing zone, but the basic idea of processing the data in physical order still holds.

How does this relate to the relative record number concept? A relative record number indicates a record's position on the file. With sequential access, processing begins with relative record 0, then moves to relative record 1, 2, and so on. Accessing data sequentially involves little more than counting. For example, imagine a program has just finished processing relative record 14. What is the next record? Obviously, relative record 15. We've already seen how a relative record number can be converted to a physical address; simply by counting records, it is possible to read them, or write them, in physical order.

Processing records in not always acceptable. For example, when a subscriber moves, his or her address must be changed in the file. Searching for that subscriber's record sequentially is like looking for a telephone number by starting with the first page of the telephone book and reading line by line. That's not how we use a telephone book. Instead, knowing the records are stored in alphabetical order, we quickly narrow our search to a portion of a single page and then begin reading the entries, ignoring the bulk of the data. The way we use a telephone book ins good example of direct, or random, access.

A disk drive reads or writes one record at a time. To randomly access a specific record, all the programmer must do is remember its address, and ask for it . the problem is remembering all those disk addresses. One solution is maintaining an index of the records. Again, we'll use the name and address file as an example. We want to access individual customer records by name. As the file is created, records are written, one at a time, in relative record number order. Additionally, as each record is written, the customer name and the associated relative record number are recorded in an array or index . After the last record has been written to disk and its position recorded on the index, the index is itself stored. Once the index has been created, it can be used to find individual records.

Assume, for example, that Susan Smith has changed her address. To record her new address on the file, a program could.

(1) read the file index,

(2) search the index for her name,

(3) find her relative record number,

(4) compute the disk address, and read her record,

(5) change her address,

(6) rewrite the record to the same place on disk.

Note that this specific record is accessed directly, and that no other records in the file are involved.

The basic idea of direct access is assigning each record an easy-to-remember, logical key, and then converting that key to a relative record number. Given this relative location, a physical address can be computed, and the record accessed. Using an index is one technique for converting, keys to physical addresses. An option is passing a numeric key to an algorithm and computing a relative record number. Goth techniques have the same objective: converting a programmer's logical data requests to physical form.

Earlier in the chapter we identified the gap separating logical and physical I/O. An access method is a software module that bridges this gap, converting logical keys to physical addresses, and issuing the appropriate primitive commands. There are many variations of sequential, indexed, and direct organizations, and each one has its own access rules. Using a variety of data access techniques can be confusing, and this is one reason for the growing popularity of database management systems.

Database Management

They are problems with traditional data management. Many result from viewing applications independently. For example, consider payroll. Most organizations prepare their payrolls y computer because using a machine instead of a small army of clerks saves money. Thus, the firm develops a payroll program to process a payroll file. Inventory, accounts receivable, accounts payable, and general ledger analysis are similar applications, so the firm develops an inventory program, and inventory file, an accounts receivable program, and accounts receivable file, and so on. Each program is independent, and each processes it own independent data file.

Why is this a problem? For one thing, different application often need the same data elements. For example, schools generate both bills and student grade reports. View the applications independently. The billing program reads a file of billing data, and the grade report program reads an independent file of grade data. The outputs of both programs are mailed to each student's home; thus, student names and addresses must be redundantly recorded on both files. What happens when a student moves? Unless both files are updated, one will be wrong. Redundant data are difficult to maintain.

A more subtle problem is data dependency. Each access certain "tricks of the trade" can significantly improve the efficiency of a given program. Because the motivation for using the computer is saving money, the programmer is often tempted to save even more by taking advantage of these efficiencies. Thus, the program's logic becomes dependent upon the physical

structure of the data, When a program's logic is tied to its physical data structure, changing that structure will almost certainly require changing the program. As a result, programs using traditional access methods can be difficult to maintain.

The solution to both problems is often organizing the data as a single, integrated database. The task of controlling access to all the data can then be concentrated in a centralized database management system.

How does the use of a centralized database solve the data redundancy problem? All data are collected and stored in a single place; consequently, there is one and only one copy of any given data element. When the value of an element (an address, for example) changes, the single database copy is corrected. Any program requiring access to this data element gets the same value, because there is only one value.

How does a database help to solve the data dependency problem? Since the responsibility for accessing the physical data rests with the database management system, the programmer can ignore the physical data structure. As a result, programs tend to easier to maintain. Expect the trend toward database management to continue.

# 附录 B 外文翻译—译文部分

摘自：程序员

## 数据管理

一、为什么要数据管理

我们已经在一些细节上研究了硬件和软件，本章把注意力转向计算机的第三个基本资源--数据。许多计算机应用要求数据存放起来，以备后用，然而简单地存放数据是不够的。一个普通的计算机系统，甚至一个小系统，可有几十个磁盘和磁带，每一个都为几十个不同的应用保存数据，对任意给定的应用，只用到一组数据就够了。必须能存储、定位和检索一个给定的程序所需要的特定数据，这就是为什么需要数据管理的原因。

二、存储数据

设想一个含有若干程序的软盘，对于一个特定的应用，这些程序中只有一个要工作。如何选定、装入和执行这个程序呢？在第六章中我们知道操作系统响应用户的命令，读出磁盘的索引，根据所要求的程序名字查找程序，得到了存放程序的磁道和扇区的地址，并发出原始命令，把程序从磁盘读到主存中，接着，在 RUN 命令之后，处理器就执行该程序了。

存取数据提出了类似的问题。单个软盘能为若干个不同应用程序保存数据，对于一修特定的应用，只用到一组数据，查找正确的数据很像查找正确的程序，然而存取程序和存取数据之间有着一些差别，当需要一个程序的时候，该程序的所有指令必然装入到主存中，而数据通常是被有选择地处理，一次只处理几个数据元素。因此，仅能确定这组数据的位置是不够的，还必须辨别每个单独的数据元素。

三、数据结构

检索数据的关键是记住数据存放在什么地方。如果数据元素是按着某种一致的和很清晰的结构存放的，就可能用记住结构的方法检索数据。 最简单的数据结构是列表。例如，为计算平均值程序所准备的数据，能以用逗号隔开的数值形式存放，逗号区分了每个单独的数据元素。

大多数编程语言都支持较复杂的数据结构，称为数组。每个数组元素能存放一个数据的值，分配给每个元素一个独有的编号，用来区别于其它的元素。引用这个编号，一个特定的数据元素能被插入、提取或处理。设想一个产生姓名和地址标签的程序。每个标签需要姓名、街道地址、城市名、州名和邮政编码，如果仅需要几个标签，那么可以按列表方式存放数据。但需要的标签一多，分离的数据元素很快就变得冗长了。一种可供选择的方案是建立一个姓名和地址的数组，每行保存一个标签的数据。唯一的问题是，每个单独的元素被处理之前，整个数组必须被存放在主存中，而主存的空间又是有限的，因此，即使采用了数组，相对地说也产生不了多少标签。

一个比较好的解决办法是把这些数组组成一个文件。所有的计算机数据都以数位的排列开始。在文件中把数位分组，形成字符；接着，字符组形成有意义的数据元素，称为字

段；一组相关的字段就是一个记录。文件就是一组相关的记录。例如，在姓名和地址的文件中，单独的姓名一项就是一个字段。每个记录为一个标签存放了完整的一组数据（姓名、街道地址等）。文件由所有的记录组成。

文件中的数据按记录逐个地被处理。通常，文件存放在辅助存储器上，如磁盘。为了读一个记录，处理它的字段，产生合适的输出，接着读出和处理另外的记录，要编写一些程序。因为一次只有一个记录在主存内，所以只需要很少的主存空间；又因为许多记录能存在单个磁盘上，所以大量的数据可在一个有限的空间内被处理。

四、查找文件

假设一个文件存在磁盘上，想存取它的数据，首先要找到这个文件。这项任务很像找程序，但也有些差别。依照程序的处理命令 LOAD 或 RUN，程序被操作系统装入；另一方面，从程序逻辑的功能上讲，数据由应用程序处理。一般来说，只有当需要数据时，程序才要求操作系统打开文件。每个文件都有名字，操作系统打开文件的程序读磁盘索引，用名字找文件，找到文件中第一个记录的地址。

五、查找记录

文件一被找到，存取它的记录过程就能开始了。当程序需要输入数据时，它就读记录；当它准备输出结果时，它就写记录。注意：这些命令只处理被选中的记录，而不处理整个文件。我们能打开一些文件。我们能读和写一些记录。

让我们更深地研究数据存取过程。程序员从逻辑上考虑数据，为一个特定的顾客请求下一个记录或姓名与地址。数据存放在辅助存储介质上，如磁盘。为了实际地存取一个记录，必须写入磁盘驱动器一组原始命令：查寻、读出和写入等。程序员用逻辑 I/O 考虑问题，外围设备能存放和检索物理的扇区，所以它用物理 I/O “思考”。因此，必然存在着一个简单的装置，把程序员的逻辑请求转换成相应的物理命令。在小型计算机上，在操作系统的 I/O 控制系统中可找到大量这方面的逻辑功能；在大型计算机上采用了若干存取方法。用数据库管理系统把程序员的逻辑数据请求转换成物理形式是越来越广泛采用的一种方法。

六、相对记录概念

软件，不管是操作系统、存取方法或者数据库，是如何找到文件中的特定记录呢？许多存储和检索技术的关键是相对记录号。设想一个含有一百个记录的串，第一个记录的编号为 0，第二个为 1，第三个为 2 等等。编号指明了一个给定的记录相对于文件中第一个记录的位置。文件的第一个记录（相对记录编号为 0）是在文件的起点上加 0，第二个记录则加 1，以此类推。

现在把这些记录存在磁盘上。为简化这个例子，在每个扇区上存放一个记录。相对于文件的起始点给扇区编号为 0、1、2 等等。注意，相对记录编号（即一个逻辑概念）与相对扇区编号（即一个物理位置）是完全相同的。给出一个相对记录编号，可计算出相对扇

区编号，给出相对扇区编号，就可计算出磁盘上的一个物理地址。

假设一个文件从第 30 磁道第 0 扇区开始存放，并在每个扇区上存放一个逻辑记录，如图 8.7 所示，第一号相对记录存放在第 30 磁道第一扇区上，如此等等。那么第 10 号相对记录在哪呢？它必在第 30 磁道第 10 区段上。在本例中，某个相对记录编号指明了该记录存放在从文件起点算起的第几号扇区上。因此，可以用相对记录号加上文件起始地址的方法算出任意一个记录的物理位置。（记住：文件的起始地址是当文件被打开时，从磁盘的索引中得到了）。文件从第 30 磁道第 0 扇区开始，第 10 号相对记录存放在离起点第 10 个扇区的第 30 磁道第 10 扇区上。我们已经把逻辑数据请求转换成了专用的物理命令了。

如果每个扇区上存放两个或更多的逻辑记录，或者建立一个文件，跨越两个或更多的磁道，使查寻的方法会更复杂。虽然我们不考虑这些细节，但以上两种情况给定了它的相对记录编号，开发一种计算记录物理位置的算法仍然是可能的。可使用各种不同的算法，有些允话记录按顺序存放或检索，还有些允许各个记录被随机存取。让我们研究几种通用的数据存取技术。

七、存储方法

设想为一个俱乐部准备会议通知，需要一组邮递标签，每个成员的姓名和地址都记录在索引卡上。大概产生这些标签最容易的方法是从第一个卡片上复制数据，然后第二个，以此类推。从文件的开始到结束，按顺序地处理记录。

杂志的发行人对每一新期刊的发行都面临着与上例同样的问题。只不过需要数以万计的订户邮递标签。发行人把订户的数据存在磁盘或磁带上，为每个订户准备了一个记录，而不是使用索引卡片。保证产生所有标签的最容易办法是，处理前先把记录按顺序存放好，然后按着该顺序，从文件的第一个记录到最后一个记录，逐个处理。逐个处理。为简化处理，记录必须用邮政编码或根据邮政区域事先分类。但是，按物理顺序处理数据的基本思想依然保留。

以上所述与相对记录编号的概念有何关系呢？相对记录编号指明了记录在文件上的位置。采用顺序存取，从第 0 个相对记录开始，然后移到第一个，第二个……以此类似。按顺序地存取数据仅涉及计数。假设一个程序刚好处理完第 14 号相对记录，那么接着处理哪个记录呢？显然是第 15 号。我们已经明白了，相对记录编号如何才能转换成一个物理地址。简单地计数记录即可，所以按物理顺序读写记录是可能的。

并不总是采用按顺序处理记录的方法。例如，当用户搬家了，他的地址在文件中必须改变。按顺序地查找订户的记录很像从电话簿的第一页开始逐页逐行地查找某个电话号码一样，这不是我们使用电话簿的方法。我们的方法是，知道了记录按字母顺序存放，很快地就把我们的查找范围缩小到某页中的一部分，忽略了大量数据，然后开始读那几项登记内容。用电话簿的方法是一个直接或随机存取的好例子。

磁盘驱动器一次只读或写一个记录，为了随机地存取一个特定的记录，程序员必须记

住它的地址，然后存取记录。问题是需要记住所有的磁盘地址，解决的办法是建立记录的索引。我们还用姓名和地址的文件作为例子，若用姓名存取每个顾客的记录，当建立文件时按相对记录的编号顺序写入记录，每次写入一个。除此之外，当每个记录被写入时，顾客的姓名和与姓名有关的相对记录编号记录在一个数组或索引中。在最后一个记录已被写入磁盘并把它的位置记录在索引中以后，把索引也存放起来。

索引一旦建立，就能用索引寻找每一个记录。例如，假设 Susan Smith 已经改变了她的地址，为了在文件上记录她的新地址，调用一个程序就能办到：

(1) 读文件索引；

(2) 在索引中找她的名字；

(3) 找出她的相对记录编号；

(4) 计算磁盘地址和读出她的记录；

(5) 改变她的地址；

(6) 把记录重写到磁盘的原位置上。

注意，这个特定的记录是被直接存取的，并不涉及文件中的其它记录。

直接存取的基本思想是分配给每个记录一个容易记忆的关键字，然后把那个关键字转换成一个相对记录编号，给出相对位置，就能算出一个物理地址，存取记录。为了把关键字转换成物理地址，用索引就是一种技术。还有一种选择就是把数值关键字传送给一个算法，然后算出一个相对记录编号。这两种技术有同样的目的，都是把程序员的逻辑数据请求转换成物理形式。

本章的前部分，我们已经标明了把逻辑 I/O 和物理 I/O 隔开的缝隙。存取方法是一个软件模块，它就是连通这个缝隙的桥梁，它把逻辑原始命令。有许多种顺序的、索引的和直接组织的存取方式，每一种都有它自己的存取规则。正在使用的种类繁多的数据存取技术可能会产生混乱，这正是促使数据库管理系统日益普及的一个原因。

## 八、数据库管理

传统的数据管理存在一些问题，许多问题是由于单独地考虑应用而引起的。例如，工资单问题，大多数组织机构都用计算机做职工的工资单，因为用计算机可代表一小批职员，节省了开支，所以公司开发了一个为处理工资文件的工资程序。清单、收入帐、支出帐和总帐分析都与工资单有相似的应用。于是，公司又开发了清单程序、清单文件、收入帐程序、收入帐文件、支出帐程序和支出帐文件等等。每个程序都是独立的，并且每个程序都只处理自己的独立的数据文件。

这为什么成为问题了呢？对某件事，不同的应用需要同样的数据，例如，学校要产生帐单和学生成绩两个报告，把它们看做独立的应用。帐单程序读出一个帐单数据文件，成绩报告程序读出一个成绩数据的独立文件。两个程序的输出都要邮递到每个学生家中，因此学生的姓名和地址必须重复地记录在两个文件中。当学生搬家时，会发生什么情况呢？

除非两个文件都被修改，否则，就有一个是错的。重复的数据是很难维护的。

较错综复杂的问题是依赖性。为了存放和检索数据，每种存取方法都有自己的规则，某些数据管理技巧能够极大地改善一个给定程序的效率。因为用计算机的动机是为了节省钱，所以程序员总想，利用这些效率来节省更多的钱。因此，程序的逻辑结构变得依赖于数据的物理结构。当程序的逻辑结构受到它的物理数据结构束缚的时候，改变了数据结构，肯定要改变程序，引出的结果是采用传统存取方法的程序很难维护。

解决以上两个问题的方法经常是把数据组成一体化的数据库，这样就能把控制存取所有数据的任务集中在一个中心的数据库管理系统中。

使用集中化数据数据库如何解决数据冗余的问题呢？把所有的数据都收集和存放在一个地方，因此任意给定的数据元素只有一个拷贝，要求存取这些数据元素的任何一个程序都能得到同样的值。原因是只存在一个值。

数据库又如何协助解决数据依赖性问题呢？由于存取实际数据的责任由数据库系统承担，因此程序员就可不考虑实际的数据结构，导致了程序更少地依赖于它的数据，一般是更容易维护了。可以预料，数据库管理会继续向前发展。