小型人事管理系统

维护手册

项目组长：叶娅娟

项目成员：郭孟鸽（完成人）

胡晨馨

## 1引言

## 1.1编写目的

软件维护是软件生命周期的最后一个阶段，它处于系统投入生产性运行以后的时期中。软件维护需要的工作量非常大，虽然在不同应用领域维护成本差别很大，但是，平均说来，大型软件的维护成本高达开发成本的四倍左右。目前国外许多软件开发组织把60%以上的人力用于维护已有的软件，而且随着软件数量增多和使用寿命延长，这个百分比还在持续上升。

软件维护就是在软件已经交付使用之后，为了改正错误或者满足新的需要而修改软件的过程。它有如下几种性质的维护：

* 改正性维护

因为软件测试不可能暴露出一个大型软件系统中所有潜藏的错误，所以在使用期间，用户必然会发现程序性错误，并且把他们遇到的问题报告给维护人员。我们把诊断和改正错误的过程称为改正性维护。

* 适应性维护

计算机科学技术领域的各方面都在迅速进步，需要经常地修改版本。为了和变化了的环境适当地配合而进行的修改软件地活动称为适应性维护。

* 完善性维护

在软件编写完成之后，投入实践，在使用软件的过程中，用户往往提出增加新功能或修改已有的功能的建议，这就需要进行完善性维护。

* 预防性维护

为了改进未来的可维护性或可靠性，或为了给未来的改进奠定更好的基础而修改软件时，就需要进行预防性维护。

维护的过程本质上是修改和压缩了的软件定义和开发过程，而且事实上远在提出一项维护要求之前，与软件维护有关的工作已经开始了。

鉴于以上各点，编写维护软件的文档十分重要。它给软件维护人员提供了一份完整、清晰的说明文档，便于其快速有效地进行维护工作。

## 1.2项目背景

对于一些小型企业来说，如何管理好企业内部员工的信息，成为企业管理中的一个大问题。在这种情况下，一个可以规范化，自动化的企业人事管理系统就显得非常必要，人事管理系统就是把分散的企事业单位的职工信息实行统一，本系统的主要功能是实现管理员对员工基本信息和工作信息的查询、添加、修改和删除，方便对人事信息的管理。

## 1.3定义

MySQL : 系统服务器所使用的数据库管理系统。

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。

## 1.4参考资料

《软件工程》 钱秋乐，清华大学出版社,2007

# 2系统说明

## 2.1系统用途

输入：管理员登录

输出：系统操作界面

功能：实现管理员对员工基本信息和工作信息的查询、添加、修改和删除，方便对人事信息的管理。

## 2.2安全保密

系统提供一定的方式让用户表示自己的身份，系统进行核实，通过鉴定后才提供系统使用权。常用的方法是用一个用户名或用户标识号来标识用户身份。

## 2.3总体说明

系统的总体功能：管理员通过浏览员工信息，进行网页下拉、鼠标滑轮滚动、输入特定的部门，或者员工职工号进行员工信息搜索。管理员点击注册或登录按钮的操作管理员输入相应信息点击添加按钮。管理员点击查看员工信息，删除等按钮并实现相应操作。

系统的具体功能：

管理员：管理员必须完成登录后，才能对员工基本信息和工作信息的查询、添加、修改和删除。

## 2.4程序说明

小型人事管理系统主要有注册界面、登录界面、主界面、查找浏览界面、添加员工信息界面、修改员工信息界面、删除员工信息界面。

### 2.4.1程序登录的说明

● 功能：管理员登录。

● 方法：管理员通过输入数据库已有的信息进行登录。

● 输入：管理员输入用户名及其匹配的密码。

● 处理：登录系统将用户名和密码与后台进行匹配，如果正确则进入系统主界面。

● 输出：系统主界面。

● 表格：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **字段描述** | **字段类型** | **允许空** | **主键** |
| managerID | 管理员ID | int | 否 | 是 |
| password | 密码 | varchar | 否 |  |
| role | 角色 | varchar | 否 |  |

# 3操作环境

## 3.1设备

一台电脑，CPU为Pentium4以上，内存为256MB以上。

## 3.2支持平台

Windows 7及以上

## 3.3数据库

标识符：职工号，姓名，性别，年龄，住址，部门，缺勤一季度，缺勤二季度，缺勤三季度，缺勤四季度，工资一季度，工资二季度，工资三季度，工资四季度。

静态数据：保存在硬盘上的数据。

动态数据：正处于处理过程中的数据。

数据库的存储媒体：硬盘。

# 4维护过程

## 4.1约定

1. 设计原则

1. 密切结合数据设计和行为设计。

2. 有机结合硬件、软件、技术和管理的界面

3. 具体程序实现过程中，对记录、字段的引用参照PersInfo类。

4. 存储区的标识符也参照PersInfo类。

5. 在设计过程中参照瀑布模型、ER模型、层次图、Jackson程序设计方法。

2. 设计程序变更的准则

1. 检查可供选择的设计方案，寻找一种与程序的原始设计原理相容的变更设计。

2. 努力使设计简化。

3. 能满足可变性要求的设计。

4. 不降低程序质量。

5. 用可测试的并具备测试方法的术语描述设计。

6. 考虑处理时间、存储量和操作过程方面的变化。

7. 考虑标更对用户服务的干扰以及实施变更的代价与时间。

3. 修改程序代码的准则

1. 必须要先熟悉整个程序的控制流程。

2. 不要做不必要的修改。

3. 不影响原始程序的风格和相容性。

4. 记录所作过的修改。

5. 审查软件质量是否符合标准。

6. 更新程序文档以反映修改并保留修改前的程序代码版本。

4. 重新验证程序的准则

1. 首先测试程序故障，然后测试程序的未改动部分，最后测试程序的修改部分。

2. 不允许做修改的维护程序员成为唯一的重新验证程序的人。

3. 鼓励终端用户参与到重新测试进程中来。

4. 在重新验证进程中，记录出错的次数与类型，并把结果同所提供的测试功能进行比较，以便估量出程序是否退化。

## 4.2验证过程

每当软件被修改后，都要校验其正确性。维护员应该有选择地做些重新测试工作，不仅要证实新的逻辑的正确性，而且要校验程序的无修改部分是否无损害，并且整个程序运行正确。若发现错误，则要马上进行修正。

## 4.3专门维护过程

系统运行一段时间后，由于记录的不断增加、删除和修改，会使数据库的物理存储变坏。例如，逻辑上属于同一记录型或同一关系的数据被分散到了不同的文件或文件的多个碎片上。这样就会降低数据库存储空间的利用率和数据的访存效率，使数据库的性能下降。这时就要进行数据库的重组织。在重组过程中，按原设计要求重新安排记录的存储位置，调整数据区和溢出区，回收“垃圾“，减少指针链等。

## 4.4程序清单和流程图

详见概要设计和详细设计文档。