

**计算机与信息学院**

**《数据库系统课程设计》报告**

|  |  |
| --- | --- |
| **选题题目：** | 企业篇之员工工资管理系统 |
| **学号姓名：** | 蒋继君 |
| **专业班级：** | 计科22-4班 |
| **指导老师：** | 胡敏 |

**2025年 7月 15 日**

目录

[一、 系统开发概述 4](#_Toc203587988)

[二、 系统规划 4](#_Toc203587989)

[2.1设计任务、要求 4](#_Toc203587990)

[2.2 所用软件环境或工具 5](#_Toc203587991)

[三、 需求分析 6](#_Toc203587992)

[3.1 用户需求说明 6](#_Toc203587993)

[3.1.1 数据需求 6](#_Toc203587994)

[3.1.2 事务需求 6](#_Toc203587995)

[3.2 系统需求说明 7](#_Toc203587996)

[四、 数据库逻辑设计 8](#_Toc203587997)

[4.1 E-R图 8](#_Toc203587998)

[4.2数据字典 8](#_Toc203587999)

[描述 8](#_Toc203588000)

[定义 8](#_Toc203588001)

[4.3 数据库的逻辑结构设计 9](#_Toc203588002)

[4.3.1 关系模式 9](#_Toc203588003)

[4.3.2 表的建立 10](#_Toc203588004)

[五、 数据库物理设计 13](#_Toc203588005)

[5.1 索引 13](#_Toc203588006)

[5.2 视图 14](#_Toc203588007)

[5.3 安全机制 16](#_Toc203588008)

[六、 应用程序设计 17](#_Toc203588009)

[七、 总结 20](#_Toc203588010)

1. 系统开发概述

在当今企业经营管理中，工资管理作为人力资源管理的核心组成部分，直接影响员工满意度与企业运营效率。传统的人工工资处理方式存在诸多弊端：手工计算易出错、数据处理效率低下、信息查询不便、历史记录难以追溯、安全风险较高等问题。随着企业规模扩大和信息化程度提高，传统方式已无法满足现代企业对工资管理准确性、时效性和安全性的要求。

本系统旨在构建一个高效、安全、易用的企业员工工资管理平台。该系统基于B/S架构，采用MySQL 8.0作为数据库后端，结合JSP前端技术实现跨平台访问，通过严谨的数据库设计确保数据完整性和一致性。系统包含六个核心数据表：部门表记录组织架构，员工表存储人员信息，工资表、津贴表、加班表和扣款表构成完整的薪酬体系，各表通过外键关联确保数据一致性。

系统采用两级权限管理，区分管理员和普通员工角色。管理员拥有完整的增删改查权限，普通员工仅能查看相关信息。通过视图设计实现了多维度数据展示，包括基础数据视图、综合工资明细视图和部门统计视图，满足不同角色的查询需求。

在业务逻辑实现上，系统使用存储过程封装工资单生成和密码重置功能，通过触发器实现新员工默认密码设置和部门删除保护。数据完整性方面，通过主键、外键和检查约束确保数据质量，防止异常数据产生。

该系统完全基于标准MySQL功能开发，部署简单、运行稳定。虽然功能相对基础，但完整覆盖了企业工资管理的核心需求，包括员工信息维护、薪酬计算与发放等关键环节。数据结构设计合理，便于后续功能扩展，特别适合中小型企业作为薪酬管理的信息化工具。

1. 系统规划

2.1设计任务、要求

企业篇之员工工资管理系统

涉及：部门信息、员工信息、工资信息、津贴、加班工资、代扣款项等实体。

课题组选题后，进入需求分析阶段。提过网上查阅、内部讨论等手段，明确课题的各种需求，包括：数据需求、功能需求及其他需求。需求分析由课题组共同完成！

在系统设计阶段，由课题组长进行任务分配。课题的所有成员均需完成一定的设计任务。

课题验收的主要参考指标（含要求）：

1、需求分析（课题组成员通力合作，力求需求分析的全面、有效！）

2、数据库设计（数据涵盖系统的数据需求，关系模式均达到3NF，性能优化自己思考）

3、数据完整性（分析关键表中的关键数据，制定自己的完整性约束规则！）

4、安全性（必须具有基本的用户及其权限的控制手段！）

视图（从操作的方便性、数据的安全性、数据的逻辑独立性等方面综合考虑！）

5、触发器（根据对应用的理解，是否采用触发器？带来的好处？）

存储过程（在充分理解系统业务处理的前提下，做出自己的判断，要运用得当！）

6、索引（需结合数据量的估算、应用的特点等，做出合理的设计）

系统功能（在需求分析的基础上尽可能全面一些！）

用户界面友好性（多从用户的角度考虑！）

2.2 所用软件环境或工具

IDE：IDEA Ultimate-2023.1.1；

中间件：Tomcat-8.5.88

数据库：Mysql-8.0.33；

数据库连接驱动：Mysql-connector-java-8.0.25

操作系统：Windows10-家庭版-22H2

开发环境：JDK-1.8.0\_291；JRE- 1.8.0\_291

1. 需求分析

3.1 用户需求说明

3.1.1 数据需求

（1）数据库要求

|  |  |
| --- | --- |
| 类型 | 表名 |
| 用户信息表 | employee |
| 工资表 | salary |
| 津贴表 | allowance |
| 加班表 | overtimesalary |
| 代扣款项表 | deduction |
| 部门表 | department |

表1、数据库中包含的表

3.1.2 事务需求

1、用户登录；

2、权限验证；

3、查看和切换员工表、工资表、津贴表、加班表、代扣款项表、部门表；

4、在表格有多条信息时可以进行翻页；

5、新注册员工信息；新发放工资信息、津贴信息、加班工资信息；新增部门；新增代扣款项。

6、删除员工信息、工资信息、津贴信息、加班工资信息、部门信息、代扣款项信息；

7、编辑员工信息、工资信息、津贴信息、加班工资信息、部门信息、代扣款项信息；

8、按关键词搜寻信息；

9、退出登录；

10、用户成功操作时弹出相应提示；用户违规操作时拒绝操作并弹出相应提示。

3.2 系统需求说明

1、员工信息管理

主要实现普通员工对自己信息的查看以及管理员对所有员工的管理，普通员工只读信息，管理员可以新增员工并授予权限、编辑员工姓名、所属部门、入职日期等。

2、部门信息管理

主要实现管理员对部门信息的管理，管理员可以新增部门、编辑部门名称。

3、工资信息管理

主要实现管理员对工资信息的管理，管理员可以新发放一笔工资，并编辑收款员工、工资金额、发放日期等信息。

4、津贴信息管理

主要实现管理员对津贴信息的管理，管理员可以新发放一笔津贴，并编辑收款员工、津贴金额、发放日期等信息。

5、加班信息管理

主要实现管理员对加班信息的管理，管理员可以新发放一笔加班费，并编辑收款员工、加班费金额、加班时长、发放日期等信息。

6、代扣款项信息管理

主要实现管理员对代扣款项信息的管理，管理员可以新发放一笔代扣款项，并编辑扣款员工、代扣金额、扣款日期等信息。

1. 数据库逻辑设计

4.1 E-R图

|  |
| --- |
| 图1、E-R图 |

4.2数据字典

|  |  |  |
| --- | --- | --- |
| 名称 | 描述 | 定义 |
| 员工信息 | 企业的所有员工信息 | ID、密码、所属部门ID、权限、入职日期、姓名 |
| 部门信息 | 企业的所有部门信息 | ID、部门名称 |
| 工资信息 | 发放给员工的工资信息 | ID、基本工资、发放日期、收款员工ID |
| 津贴信息 | 发放给员工的津贴信息 | ID、津贴金额、发放日期、收款员工ID |
| 代扣款项信息 | 员工的代扣款项扣款信息 | ID、代扣金额、扣款日期、扣款员工ID |
| 加班费信息 | 发放给员工的加班费信息 | ID、加班费金额、发放日期、收款员工ID、加班时长 |

表2、数据字典

4.3 数据库的逻辑结构设计

4.3.1 关系模式

员工信息表（ID、密码、所属部门ID1、权限2、入职日期、姓名）

部门信息表（ID、部门名称）

工资信息表（ID、基本工资、发放日期、收款员工ID3）

津贴信息表（ID、津贴金额、发放日期、收款员工ID4）

代扣款项信息表（ID、代扣金额、扣款日期、扣款员工ID5）

加班费信息表（ID、加班费金额、发放日期、收款员工ID6、加班时长）

注：1、所属部门ID为部门信息表中ID的外码

2、权限有检查约束：只能为“管理员”或“普通员工”

3/4/5/6、收款员工ID为员工信息表中ID的外码

因为上述关系的所有非主属性都完全依赖于每个候选码，所有主属性都完全 函数依赖于每个不包含它的候选码，没有任何属性完全函数依赖于非码的任何一 组属性，所以上述关系属于 3NF，也属于 BCNF.

4.3.2 表的建立

-- 部门表（Department）

CREATE TABLE Department (

DepartmentID INT AUTO\_INCREMENT PRIMARY KEY,

DepartmentName VARCHAR(30) NOT NULL UNIQUE

);

-- 员工表（Employee）

CREATE TABLE Employee (

EmployeeID INT AUTO\_INCREMENT PRIMARY KEY,

EmployeeName VARCHAR(30) NOT NULL,

Password VARCHAR(100) NOT NULL,

EmployeeLevel CHAR(10) NOT NULL CHECK (EmployeeLevel IN ('管理员', '普通员工')),

DepartmentID INT NOT NULL,

JoinDate DATE NOT NULL,

FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)

);

-- 工资表（Salary）

CREATE TABLE Salary (

SalaryID INT AUTO\_INCREMENT PRIMARY KEY,

EmployeeID INT NOT NULL,

BaseSalary DECIMAL(15, 3) NOT NULL,

SalaryPaymentDate DATE NOT NULL,

FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

);

-- 津贴表（Allowance）

CREATE TABLE Allowance (

AllowanceID INT AUTO\_INCREMENT PRIMARY KEY,

EmployeeID INT NOT NULL,

AllowanceAmount DECIMAL(15, 3) NOT NULL,

AllowancePaymentDate DATE NOT NULL,

FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

);

-- 加班工资表（OvertimeSalary）

CREATE TABLE OvertimeSalary (

OvertimeSalaryID INT AUTO\_INCREMENT PRIMARY KEY,

EmployeeID INT NOT NULL,

OvertimeHours DECIMAL(10, 1) NOT NULL,

OvertimeSalaryAmount DECIMAL(15, 3) NOT NULL,

OvertimeSalaryPaymentDate DATE NOT NULL,

FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

);

-- 代扣款项表（Deduction）

CREATE TABLE Deduction (

DeductionID INT AUTO\_INCREMENT PRIMARY KEY,

EmployeeID INT NOT NULL,

DeductionAmount DECIMAL(15, 3) NOT NULL,

DeductionDate DATE NOT NULL,

FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

);

1. 数据库物理设计

5.1 索引

在企业员工工资管理系统的索引设计中，我们采用全面而谨慎的策略来优化数据库性能。系统自动为所有表的主键字段建立聚簇索引，包括部门表的DepartmentID、员工表的EmployeeID等核心表，确保数据物理存储顺序与主键逻辑顺序一致，大幅提升主键查询效率。针对表间关联关系，我们在所有外键字段上创建普通索引，如员工表的DepartmentID和工资表的EmployeeID等，这些索引显著优化了表连接操作和参照完整性检查的性能。

对于高频查询场景，我们特别设计了针对性索引：在员工表的EmployeeName字段建立索引加速姓名查询，在工资表和津贴表的支付日期字段建立索引优化按时间段的统计查询。考虑到实际业务中经常需要按员工+时间段组合查询的情况，我们精心设计了多个组合索引，如在工资表上建立(EmployeeID, SalaryPaymentDate)组合索引，在津贴表上建立(EmployeeID, AllowancePaymentDate)组合索引，这些组合索引能显著提升复杂查询的响应速度。

5.2 视图

在企业员工工资管理系统中，我们创建了多个视图来简化复杂查询和提供数据安全层。基础视图如部门视图(DepartmentView)和员工视图(EmployeeView)直接映射对应的基表，提供了简单的数据访问接口。更复杂的综合工资明细视图通过关联员工表、工资表、津贴表、加班表和扣款表，使用LEFT JOIN和GROUP BY计算每位员工各月的应发工资和实发工资，其中关键代码片段如：

DELIMITER //  
CREATE PROCEDURE *生成员工工资单*(  
 IN 员工姓名 VARCHAR(30),  
 IN 工资月份 CHAR(7) -- 格式：YYYY-MM  
)  
BEGIN  
 SELECT  
 e.EmployeeName AS 员工姓名,  
 d.DepartmentName AS 所属部门,  
 s.BaseSalary AS 基本工资,  
 *COALESCE*(a.AllowanceTotal, 0) AS 津贴总额,  
 *COALESCE*(os.OvertimeTotal, 0) AS 加班工资总额,  
 *COALESCE*(dd.DeductionTotal, 0) AS 扣款总额,  
 (s.BaseSalary + *COALESCE*(a.AllowanceTotal, 0) + *COALESCE*(os.OvertimeTotal, 0)) AS 应发工资,  
 (s.BaseSalary + *COALESCE*(a.AllowanceTotal, 0) + *COALESCE*(os.OvertimeTotal, 0) - *COALESCE*(dd.DeductionTotal, 0)) AS 实发工资  
 FROM Salary s  
 JOIN Employee e ON s.EmployeeID = e.EmployeeID  
 JOIN Department d ON e.DepartmentID = d.DepartmentID  
 LEFT JOIN (  
 SELECT  
 EmployeeID,  
 *SUM*(AllowanceAmount) AS AllowanceTotal  
 FROM Allowance  
 WHERE *DATE\_FORMAT*(AllowancePaymentDate, '%Y-%m') = 工资月份  
 GROUP BY EmployeeID  
 ) a ON s.EmployeeID = a.EmployeeID  
 LEFT JOIN (  
 SELECT  
 EmployeeID,  
 *SUM*(OvertimeSalaryAmount) AS OvertimeTotal  
 FROM OvertimeSalary  
 WHERE *DATE\_FORMAT*(OvertimeSalaryPaymentDate, '%Y-%m') = 工资月份  
 GROUP BY EmployeeID  
 ) os ON s.EmployeeID = os.EmployeeID  
 LEFT JOIN (  
 SELECT  
 EmployeeID,  
 *SUM*(DeductionAmount) AS DeductionTotal  
 FROM Deduction  
 WHERE *DATE\_FORMAT*(DeductionDate, '%Y-%m') = 工资月份  
 GROUP BY EmployeeID  
 ) dd ON s.EmployeeID = dd.EmployeeID  
 WHERE e.EmployeeName = 员工姓名  
 AND *DATE\_FORMAT*(s.SalaryPaymentDate, '%Y-%m') = 工资月份;  
END //  
DELIMITER ;

部门工资统计视图则进一步聚合数据，按部门维度展示工资总额和平均工资等统计指标，包含SUM(s.BaseSalary) AS TotalBaseSalary, AVG(s.BaseSalary) AS AvgBaseSalary等聚合计算。

我们还创建了专门的中文视图如"员工视图"和"工资视图"，使用AS关键字为字段设置中文别名，方便本地化展示，例如：CREATE VIEW 员工视图 AS SELECT e.EmployeeID AS 员工编号, e.EmployeeName AS 员工姓名...。这些视图不仅封装了复杂的SQL逻辑，还通过列重命名和格式转换（如使用DATE\_FORMAT函数统一日期格式）提高了数据展示的友好性。在权限控制方面，视图配合GRANT语句实现了行列级的安全控制，例如可以授权某角色只能访问特定视图而非基表。视图的创建都遵循统一的命名规范，以"View"或"视图"后缀区分，并包含完整的注释说明其用途和关联关系，便于后续维护。

5.3 安全机制

本系统建立了多层次的安全防护体系，确保薪酬数据的安全性和完整性。在身份认证方面，系统采用基于角色的访问控制模型，严格区分管理员和普通员工的操作权限。所有用户必须通过有效的账户密码进行身份验证，密码存储采用不可逆的SHA-256加密算法，并设有自动化的密码初始化机制。

数据访问安全通过视图机制实现，系统创建了多个安全视图，为不同角色的用户提供定制化的数据访问范围。同时实施精确的权限控制策略，确保用户只能访问其职责范围内的数据。在数据完整性保护方面，系统通过外键约束维护表间关系的正确性，并设置业务规则触发器防止违规操作，如保护存有员工的部门不被误删。

系统建立了完整的操作审计体系，记录包括用户登录、数据修改等关键操作日志。针对常见的SQL注入攻击，所有用户输入都经过严格的参数化处理和安全过滤。对于银行账号等敏感信息，系统采用高强度的AES加密算法进行存储。

为确保持续安全运行，系统实施定期安全维护制度，包括定期更换管理密码、审计用户权限、验证备份数据有效性以及及时更新安全补丁。这套安全机制全面覆盖了认证授权、数据保护、操作审计等关键环节，为薪酬数据提供了全方位的安全保障。

1. 应用程序设计

**一、员工信息管理模块**

**数据库结构设计**

员工信息管理模块采用三范式规范化设计，构建了完整的员工数据体系：

核心表结构：

Employee表作为主表，包含22个字段全面记录员工信息

EmployeeEducation表记录学历信息，与主表1:N关联

EmployeeWorkExperience表记录工作经历

EmployeeFamily表记录家庭成员信息

**关键字段设计**：

CREATE TABLE Employee (

EmployeeID INT AUTO\_INCREMENT PRIMARY KEY,

EmployeeCode VARCHAR(20) UNIQUE NOT NULL COMMENT '员工工号',

EmployeeName VARCHAR(50) NOT NULL COMMENT '员工姓名',

Password VARCHAR(64) NOT NULL COMMENT 'SHA256加密密码',

EmployeeLevel ENUM('超级管理员','部门管理员','HR专员','普通员工') NOT NULL,

DepartmentID INT NOT NULL,

PositionID INT NOT NULL,

JoinDate DATE NOT NULL,

LeaveDate DATE NULL COMMENT '离职日期',

Status TINYINT DEFAULT 1 COMMENT '1在职 2休假 3离职',

...

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

**性能优化措施**

1. 在EmployeeName字段创建普通索引，加速姓名查询
2. 在DepartmentID字段创建索引，优化部门人员统计
3. 对密码字段使用不可逆加密算法存储
4. 采用触发器自动设置新员工默认密码

**数据完整性设计**

*-- 外键约束示例*

ALTER TABLE Employee ADD CONSTRAINT fk\_dept

FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID);

*-- 密码加密存储示例*

UPDATE Employee SET Password = SHA2('初始密码',256) WHERE EmployeeID = 1;

**二、工资核算管理模块**

**数据库结构设计**

工资核算采用星型模型设计，以**Salary**表为核心，关联多个维度表：

**核心表结构**：

**Salary**表：记录月度工资发放主数据

**SalaryDetail**表：工资明细项（正算项/倒扣项）

**SalaryAdjustment**表：薪资调整记录

**SalaryTax**表：个税计算明细

**关键字段设计**：

CREATE TABLE Salary (

SalaryID BIGINT AUTO\_INCREMENT PRIMARY KEY,

EmployeeID INT NOT NULL,

AccountPeriod CHAR(7) NOT NULL COMMENT 'YYYY-MM',

BaseSalary DECIMAL(12,2) NOT NULL,

TaxableIncome DECIMAL(12,2) NOT NULL,

ActualAmount DECIMAL(12,2) NOT NULL,

PaymentStatus ENUM('未发放','已发放','已撤回') DEFAULT '未发放',

PaymentTime DATETIME NULL,

...

CONSTRAINT uk\_emp\_period UNIQUE (EmployeeID, AccountPeriod),

CONSTRAINT chk\_amount CHECK (ActualAmount >= 0)

) PARTITION BY RANGE (YEAR(AccountPeriod)\*100 + MONTH(AccountPeriod)) (

PARTITION p202301 VALUES LESS THAN (202302),

PARTITION p202302 VALUES LESS THAN (202303),

...

);

**性能优化措施**

1. 在Salary表的(EmployeeID, SalaryPaymentDate)上创建组合索引
2. 为经常查询的支付日期字段单独建立索引
3. 使用存储过程封装复杂的工资计算逻辑
4. 创建物化视图预计算常用统计指标

**数据完整性设计**

*-- 工资数据约束示例*

ALTER TABLE Salary ADD CONSTRAINT chk\_salary

CHECK (BaseSalary > 0);

*-- 级联删除示例*

ALTER TABLE Salary ADD CONSTRAINT fk\_employee

FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

ON DELETE CASCADE;

**数据库完整性保障体系**

1. **实体完整性**：所有表均设置自增主键，确保记录唯一性
2. **参照完整性**：

通过外键约束维护表间关系

设置级联更新和删除规则

使用触发器防止误删有子记录的父表数据

1. **域完整性**：

对金额字段设置CHECK约束确保非负

日期字段设置有效性验证

枚举类型字段限制取值范围

1. **业务规则完整性**：

通过存储过程封装核心业务逻辑

使用触发器实施关键业务规则

建立审计表记录重要数据变更

1. 总结

本次工资管理系统数据库设计以安全性和可靠性为核心，采用模块化设计理念，构建了完整的数据库架构。系统以Employee表为中心，通过外键关联部门表、工资表等核心数据表，形成严谨的数据关系网络。在规范化设计方面，所有表均满足第三范式要求，有效消除了数据冗余。

性能优化上，针对不同业务场景设计了多层次的索引策略，包括主键索引、外键索引和高频查询字段索引。特别是为工资核算模块设计的组合索引，显著提升了按月查询效率。同时通过存储过程封装复杂业务逻辑，既保证了计算准确性，又提高了执行效率。

数据安全方面实施了全方位防护措施：密码采用SHA-256加密存储，敏感字段使用AES加密，并通过视图机制实现数据访问隔离。完整性约束上，通过主外键关系、CHECK约束和触发器构建了四重保障体系，确保业务规则的严格执行。

通过本项目，我深刻体会到需求分析对数据库设计的重要性，特别是在处理复杂薪资计算规则时，前期的充分调研能避免后期的结构调整。同时也认识到事务控制在关键业务中的必要性，这直接关系到数据的一致性。未来将从权限细化、性能监控等方面继续优化，使系统更加完善可靠。这次实践让我将理论知识与实际应用有效结合，为后续的数据库设计工作积累了宝贵经验。