# 银行业务管理系统数据库设计

学号 SA24225465 姓名 赵乐君

## 1 概念模型设计

### 1.1 实体设计

实体 1 客户(Customer)

用来存储每个客户的信息，以客户 ID(CustomerID)为主键，存储了客户姓名(CumtomerName)、对应员工 ID(StaffID)、居住街道(CustomerStreet)、居住城市(CustomerCity)信息。

实体 2 员工(Staff)

用来存储每位员工的信息，以员工 ID(StaffID)为主键，存储了员工姓名(StaffName)、员工电话(StaffTel)、员工居住地址(StaffAddress)、员工入职日期(StaffEnrollDate)、所处支行名称(BranchName)信息

实体 3 普通员工(CommonStaff)

继承自员工(Staff)，存储了每位普通员工的对应的经理 ID(ManagerID)

实体 4 经理(Manager)

继承自员工(Staff)，存储了每位经理所处部门(Department)

实体 5 支行(BankBranch)

用来存储每个支行的信息，以支行名称(BranchName)为主键，存储了每个支行所在城市(BranchCity)、支行资本(Capacity)信息

实体 6 账户(Account)

用来存储每个账户的信息，以账户 ID(AccountID)为主键，存储了户主 ID(CustomerID)、支行名称(BranchName)、账户余额(AccountBalance)、最近登录日期(LatestAccess)信息

实体 7 支票账户(CheckAccount)

继承自账户(Account)，存储了每个支票账户的额度(Overdraft)

实体 8 储蓄账户(DepositAccount)

继承自账户(Account)，存储了每个储蓄账户的利率(Interest)

实体 9 贷款(Loan)

用来存储每笔贷款信息，以贷款 ID(LoanID)为主键，存储了贷款用户 ID(CustomerID)、批准贷款支行名称(BranchName)、贷款金额(Amount)、贷款支付次数(Times)

实体 10 还款支付(Payment)

用来存储每笔还款信息，以贷款 ID(LoanID)和还款 ID(PayID)为主键，存储了还款金额(PayAmount)和还款日期(PayDate)

## 1.2 联系设计

联系 1 员工-客户负责关系

通过员工 ID 和客户 ID 可以查询员工与客户之间的 N-N 负责关系，并标明关系

联系 2 客户-账户归属关系

通过客户 ID 和账户 ID 可以查询到客户和账户之间的 1-N 归属关系

联系 3 支行-员工雇佣关系

通过支行名称和员工 ID 可以查询到支行与员工之间的 1-N 雇佣关系

联系 4 支行-账户归属关系

通过支行名称和账户 ID 可以查询到支行与账户之间的 1-N 归属关系

联系 5 支行-贷款批准关系

通过支行名称和贷款 ID 可以查询到支行与贷款之间的 1-N 批准关系

联系 6 客户-贷款借贷关系

通过客户 ID 和贷款 ID 可以查询到客户与贷款之间的 N-N 借贷关系

联系 7 还款-贷款支付关系

通过还款 ID 和贷款 ID 可以查询到还款与贷款之间的支付关系

### 1.3 Power Designer 的 ER 图

基于前述分析，利用 Power Designer 设计了银行业务管理系统的数据库概念模型，结果如图 1 所示。

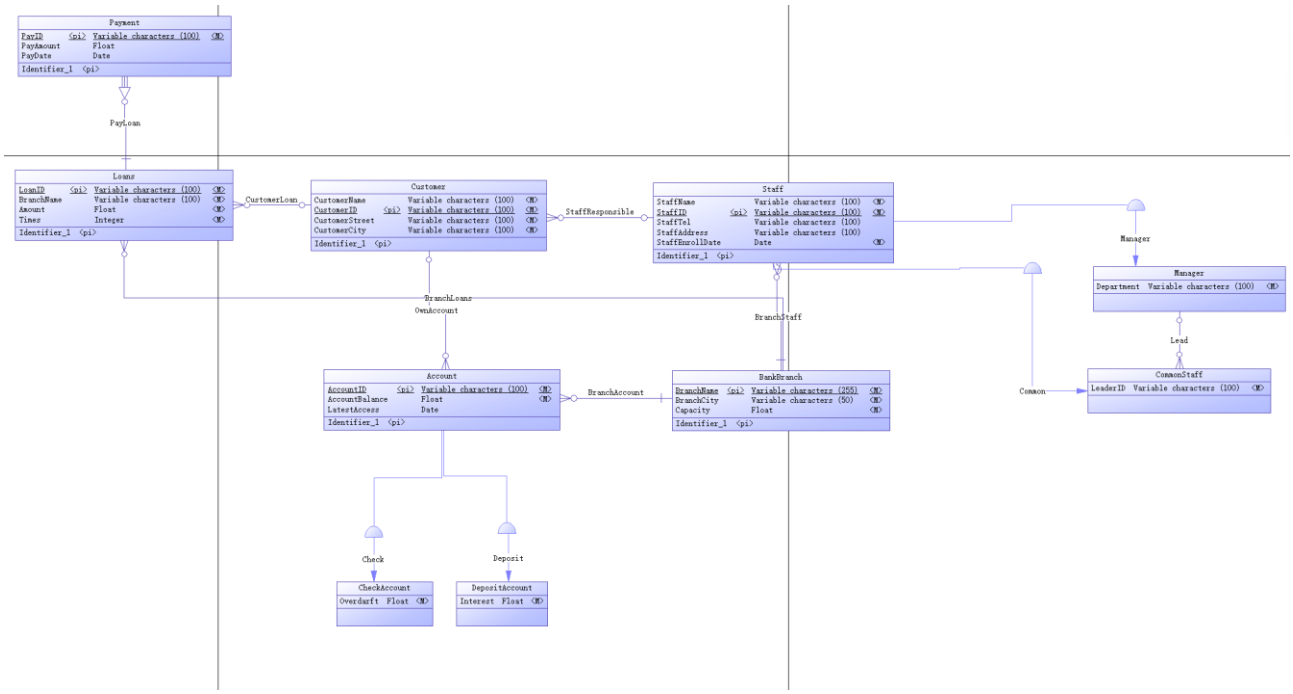


图 1 银行业务管理系统的数据库概念模型

## 2 概念模型到逻辑模型的转换

### 2.1 实体转换

| | |
|---|---|
| 客户 | CustomerID<fk>、CustomerName、StaffID<fk>、CustomerStreet、CustomerCity |
| 员工 | StaffID<fk>、StaffName、StaffTel、StaffAddress、StaffEnrollDate、BranchName<fk> |
| 普通员工 | ManagerID |
| 经理 | Department |
| 支行 | BranchName、BranchCity、Capacity |
| 账户 | AccountID<fk>、CustomerID(fk)、AccountBalance、LatestAccess、BranchName<fk> |
| 支票账户 | Overdraft |
| 储蓄账户 | Interest |
| 贷款 | LoanID<fk>、Times、Amount、CustomerID<fk>、BranchName<fk> |
| 还款支付 | LoanID、PayID、PayAmount、PayDate |

## 2.2 联系转换

| | |
|---|---|
| 员工-客户 | 创建实体 StaffResponsible 来建立 Customer 中 CustomerID 与 Staff 中 StaffID 的联系 |
| 客户-账户 | 创建实体 OwnAccount 来建立 Customer 中 CustomerID 与 Account 中 AccountID 的联系 |
| 支行-员工 | 在实体 Staff 中可以查询到每个员工对应的 BranchName<fk> |
| 支行-账户 | 在实体 Account 中可以查询到每个账户对应的 BranchName<fk> |
| 支行-贷款 | 在实体 Loans 中可以查询到每笔贷款对应的 BranchName<fk> |
| 客户-贷款 | 创建实体 CustomerLoan 来建立 Customer 中的 CustomerID 与 Loan 中的 LoanID 的联系 |
| 还款-贷款 | 在实体 Payment 中联系 PayID<pk>与 LoanID<pk, fk> |

## 2.3 最终的关系模式

| | |
|---|---|
| 客户 | CustomerID<fk>、CustomerName、StaffID<fk>、CustomerStreet、CustomerCity |
| 员工 | StaffID<fk>、StaffName、StaffTel、StaffAddress、StaffEnrollDate、BranchName<fk> |
| 普通员工 | ManagerID |
| 经理 | Department |
| 支行 | BranchName、BranchCity、Capacity |
| 账户 | AccountID<fk>、CustomerID(fk)、AccountBalance、LatestAccess、BranchName<fk> |
| 支票账户 | Overdraft |
| 储蓄账户 | Interest |
| 贷款 | LoanID<fk>、Times、Amount、CustomerID<fk>、BranchName<fk> |
| 还款支付 | LoanID、PayID、PayAmount、PayDate |
| 员工-客户 | 创建实体 StaffResponsible 来建立 Customer 中 CustomerID 与 Staff 中 StaffID 的联系 |
| 客户-账户 | 创建实体 OwnAccount 来建立 Customer 中 CustomerID 与 Account 中 AccountID 的联系 |
| 支行-员工 | 在实体 Staff 中可以查询到每个员工对应的 BranchName<fk> |
| 支行-账户 | 在实体 Account 中可以查询到每个账户对应的 BranchName<fk> |
| 支行-贷款 | 在实体 Loans 中可以查询到每笔贷款对应的 BranchName<fk> |
| 客户-贷款 | 创建实体 CustomerLoan 来建立 Customer 中的 CustomerID 与 Loan 中的 LoanID 的联系 |
| 还款-贷款 | 在实体 Payment 中联系 PayID<pk>与 LoanID<pk, fk> |

# 3 MySQL 数据库结构实现

## 3.1 Power Designer 的 PDM 设计

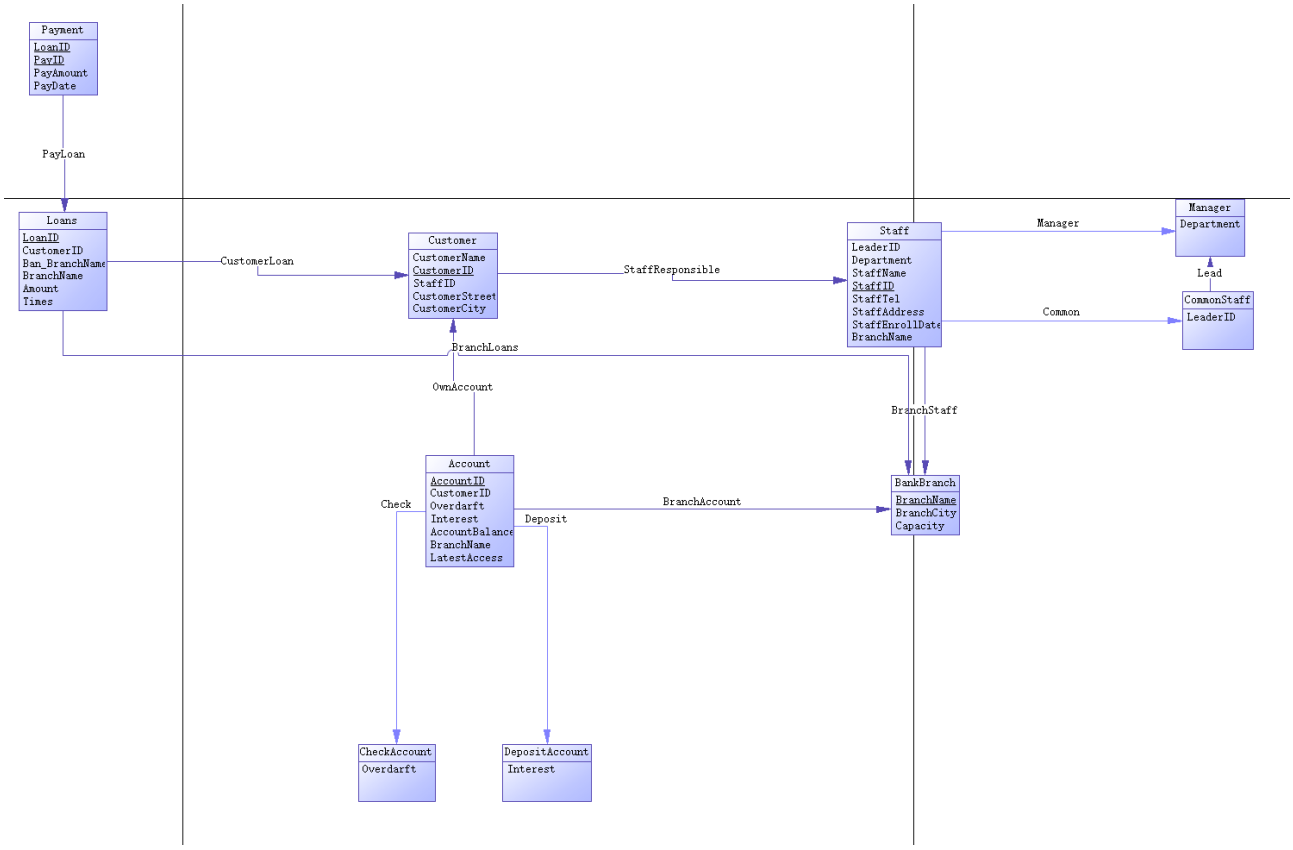


图 2 银行业务管理系统的 PDM 设计结果

## 3.2 数据库表定义

Power Designer 的 PDM 可以直接转换为 MySQL 中的基本表。下面给出了基于 PDM 构建的 MySQL 基本表设计结果，见表 1~表 13。

表 1. 客户表(Customer)

| 列名 | 中文含义 | 类型(长度) | 允许为空(Null) | 是否主键(Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| CustomerName | 客户姓名 | Char(100) | 否 | 否 | 否 |
| CustomerID | 客户编号 | Char(100) | 否 | 是 | CustomerLoan CustomerID StaffResponsible CustomerID |
| CustomerStreet | 客户街道 | Char(50) | 是 | 否 | 否 |
| CustomerCity | 客户城市 | Char(50) | 是 | 否 | 否 |

表 2. 员工(Staff)

| 列名 | 中文含义 | 类型(长度) | 允许为空 (Null) | 是否主键 (Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| StaffName | 姓名 | Char(100) | 否 | 否 | 否 |
| StaffID | 编号 | Char(100) | 是 | 是 | CommonStaff StaffID<br>Manager StaffID<br>StaffResponsible StaffID |
| StaffTel | 电话 | Char(20) | 否 | 否 | 否 |
| StaffAddress | 居住地址 | Char(100) | 否 | 否 | 否 |
| StaffEnrollDate | 入职日期 | Date | 否 | 否 | 否 |
| BranchName | 支行名称 | Char(100) | 否 | 否 | BankBranch BranchName |

表 3. 普通员工(CommonStaff)

| 列名 | 中文含义 | 类型(长度) | 允许为空 (Null) | 是否主键 (Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| StaffID | 编号 | Char(100) | 否 | 是 | Staff StaffID |
| LeaderID | 经理 ID | Char(100) | 否 | 否 | Manager StaffID |

表 4. 经理(Manager)

| 列名 | 中文含义 | 类型(长度) | 允许为空 (Null) | 是否主键 (Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| StaffID | 编号 | Char(100) | 否 | 是 | Staff StaffID |
| Department | 所属部门 | Char(100) | 是 | 否 | 否 |

表 5. 员工负责(StaffResponsible)

| 列名 | 中文含义 | 类型(长度) | 允许为空 (Null) | 是否主键 (Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| StaffID | 姓名 | Char(100) | 否 | 是 | Staff StaffID |
| CustomerID | 编号 | Char(100) | 否 | 是 | Customer CustomerID |
| ResType | 负责类型 | Int | 否 | 否 | 否 |

表 6. 支行(BankBranch)

| 列名 | 中文含义 | 类型(长度) | 允许为空 (Null) | 是否主键 (Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| BranchName | 姓名 | Char(100) | 否 | 否 | 否 |
| BranchCity | 编号 | Char(100) | 是 | 否 | 否 |
| Capacity | 资本 | Float | 否 | 否 | 否 |

表 7. 账户(Account)

| 列名 | 中文含义 | 类型(长度) | 允许为空 (Null) | 是否主键 (Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| AccountID | 编号 | Char(100) | 否 | 是 | OwnAccount AccountID<br>CheckAccount AccountID<br>DepositAccount AccountID |
| AccountBalance | 余额 | Float | 否 | 否 | 否 |
| BranchName | 支行名称 | Char(255) | 否 | 否 | BankBranch BranchName |
| LatestAccess | 最近登录 | Date | 是 | 否 | 否 |

表 8. 拥有账户(OwnAccount)

| 列名 | 中文含义 | 类型(长度) | 允许为空 (Null) | 是否主键 (Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| CustomerID | 客户编号 | Char(100) | 否 | 是 | Customer CustomerID |
| AccountID | 账户编号 | Char(100) | 否 | 是 | Account AccountID |

表 9. 支票账户(CheckAccount)

| 列名 | 中文含义 | 类型(长度) | 允许为空 (Null) | 是否主键 (Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| AccountID | 编号 | Char(100) | 否 | 是 | Account AccountID |
| Overdraft | 额度 | Float | 否 | 否 | 否 |

表 10. 储蓄账户(DepositAccount)

| 列名 | 中文含义 | 类型(长度) | 允许为空 (Null) | 是否主键 (Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| AccountID | 姓名 | Char(100) | 否 | 是 | Account AccountID |
| Interest | 利率 | Float | 否 | 否 | 否 |

表 11. 贷款(Loans)

| 列名 | 中文含义 | 类型(长度) | 允许为空 (Null) | 是否主键 (Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| LoanID | 编号 | Char(100) | 否 | 是 | CustomerLoan LoanID<br>Payment LoanID |
| BranchName | 支行名称 | Char(255) | 是 | 否 | BankBranch BranchName |
| Amount | 贷款金额 | Float | 否 | 否 | 否 |
| Times | 支付次数 | Int | 否 | 否 | 否 |

表 12. 还款支付(Payment)

| 列名 | 中文含义 | 类型(长度) | 允许为空 (Null) | 是否主键 (Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| LoanID | 贷款 ID | Char(100) | 否 | 是 | Loans LoanID |
| PayID | 支付 ID | Char(100) | 否 | 是 | 否 |
| PayAmount | 支付金额 | Float | 否 | 否 | 否 |
| PayDate | 支付日期 | Date | 否 | 否 | 否 |

表 13. 用户贷款(CustomerLoan)

| 列名 | 中文含义 | 类型(长度) | 允许为空 (Null) | 是否主键 (Primary Key) | 是否外键 |
|---|---|---|---|---|---|
| LoanID | 贷款 ID | Char(100) | 否 | 是 | Loans LoanID |
| CustomerID | 用户 ID | Char(100) | 否 | 是 | Customer CustomerID |

# 4 总结与体会

本报告给出了利用 Power Designer 进行一个银行业务管理系统数据库的基本过程，包括概念模型设计、概念模型到逻辑模型的转换以及最终的 MySQL 数据库结构实现。

设计过程中的一些个人体会如下：

1. **概念模型设计的重要性**

在数据库设计的初期，概念模型设计起到了至关重要的作用。通过明确业务需求和实体之间的关系，能有效避免后期开发过程中的结构性问题。

2. **工具使用的便利性与限制**

Power Designer 提供了直观的界面和强大的功能，可以高效完成从概念模型到逻辑模型的转换。然而，在使用过程中也发现了一些限制，例如对于复杂业务逻辑的表达，需要手动添加额外的说明或约束。

3. **实现过程中对业务规则的严格把控**

将逻辑模型实现为物理数据库结构的过程中，确保业务规则的完整性和约束条件的正确性至关重要。在 MySQL 中，通过外键约束等实现复杂的业务逻辑，进一步强化了数据的一致性和完整性。

4. **个人成长与经验总结**

通过此次设计，我对数据库建模有了更深的理解，同时也体会到了与业务团队沟通的

重要性。数据库设计不仅仅是技术实现的过程，更需要综合考虑实际业务需求、未来扩展性以及维护成本。