**银行业务管理系统数据库设计**

学号SA24225464 姓名 赵建桥

1. **概念模型设计**
   1. 实体设计

//阐述每一个实体的设计结果和理由

//文字解释即可。

1. 支行：
   1. 属性：支行名称（主键）、城市、资产。
   2. 设计理由：其中由于支行的名称为唯一的，因此采用支行名称是唯一的，因此采用支行名称作为主键。支行名称、城市都采用子串作为数据类型，资产采用小数。
2. 客户：
   1. 属性：客户身份证号（主键）、客户姓名、街道、城市。
   2. 设计理由：由于每个人都身份证号都不一样，因此采用身份证号作为主键。客户身份证号、姓名、居住街道和城市都采用字串作为进行存储的数据类型。
3. 员工：
   1. 属性：员工身份证号（主键）、员工姓名、手机号、家庭地址、入职时间
   2. 设计理由： 采用能唯一标识员工的身份证号用做主键，
4. 部门经理：
   1. 属性：部门
   2. 设计理由：部门经理继承自员工，采用属性部门可以表示出部门经理工作的部门。
5. 普通员工：
   1. 属性：部门经理身份证号
   2. 设计理由： 普通员工继承自员工。由于要求必须保存员工部门经理的身份证号，因此记录部门经理身份证号作为属性，通过这条属性也可以推出员工工作的部门。
6. 账户：
   1. 属性： 账户号码（主键），余额，最近访问日期
   2. 设计理由：由于每个账户的账户号码唯一，用于作为主键可以唯一的标识某个账户。账户号码用字符串数据类型，余额使用金钱类型，最近访问日期使用日期类型。
7. 储蓄账户
   1. 属性： 利率
   2. 设计理由：储蓄账户继承自账户，存储了用浮点数标识的账户利率。
8. 支票账户
   1. 属性：透支额度
   2. 设计理由：支票账户继承自账户，存储了用金钱类型标识的透支额度
9. 贷款
   1. 属性：贷款号（主键），金额，逐次支付
   2. 设计理由：用唯一标识每一笔贷款的贷款号作为贷款的主键。其中贷款号使用字符串表示，金融采用金钱来表示，逐次支付记录贷款付清的次数，采用整数类型进行表示。
10. 支付
    1. 属性：支付号（主键），支付金额，交易日期
    2. 设计理由：支付是依赖于贷款的弱实体，记录了对于贷款的支付情况。支付号用于唯一的标识每一笔支付，作为主键。其中，支付号是字符串，支付金融为金钱类型，交易日期采用日期类型。
    3. 联系设计
11. 拥有账户：
    1. 涉及双方：客户，账户
    2. 关系：多对多
    3. 设计理由：一个客户可以有多个帐户，多个客户可以共用一个账户
12. 客户贷款：
    1. 涉及双方：客户，贷款
    2. 关系：多对多
    3. 设计理由：一个客户可以有多个贷款，一个贷款可以被多个客户共享
13. 发放贷款：
    1. 涉及双方：支行，贷款
    2. 关系：一对多
    3. 设计理由：每笔贷款只能由特定的某个支行发放，但是每个支行会发放多笔贷款。
14. 负责：
    1. 涉及双方：员工，客户
    2. 属性：负责类型，用于区分员工为客户的贷款负责人还是银行账户负责人。
    3. 关系：多对多
    4. 设计理由：每个员工可能有多个员工作为负责人，每个员工也会负责多个客户。
15. 开户：
    1. 涉及双方：支行，账户
    2. 关系：一对多
    3. 设计理由：每个账户只能有一个支行作为开户行，但是每个支行可以为多个账户提供开户服务。
16. 贷款支付：
    1. 涉及双方：贷款，支付
    2. 关系：一对多
    3. 设计理由：每笔贷款由多笔支付完成，但是每笔支付只能对应于一个贷款。
17. 雇佣：
    1. 涉及双方：支行，员工
    2. 关系：一对多
    3. 设计理由：每个员工只能由一个支行雇佣，每个支行可以雇佣多个员工。
    4. 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。

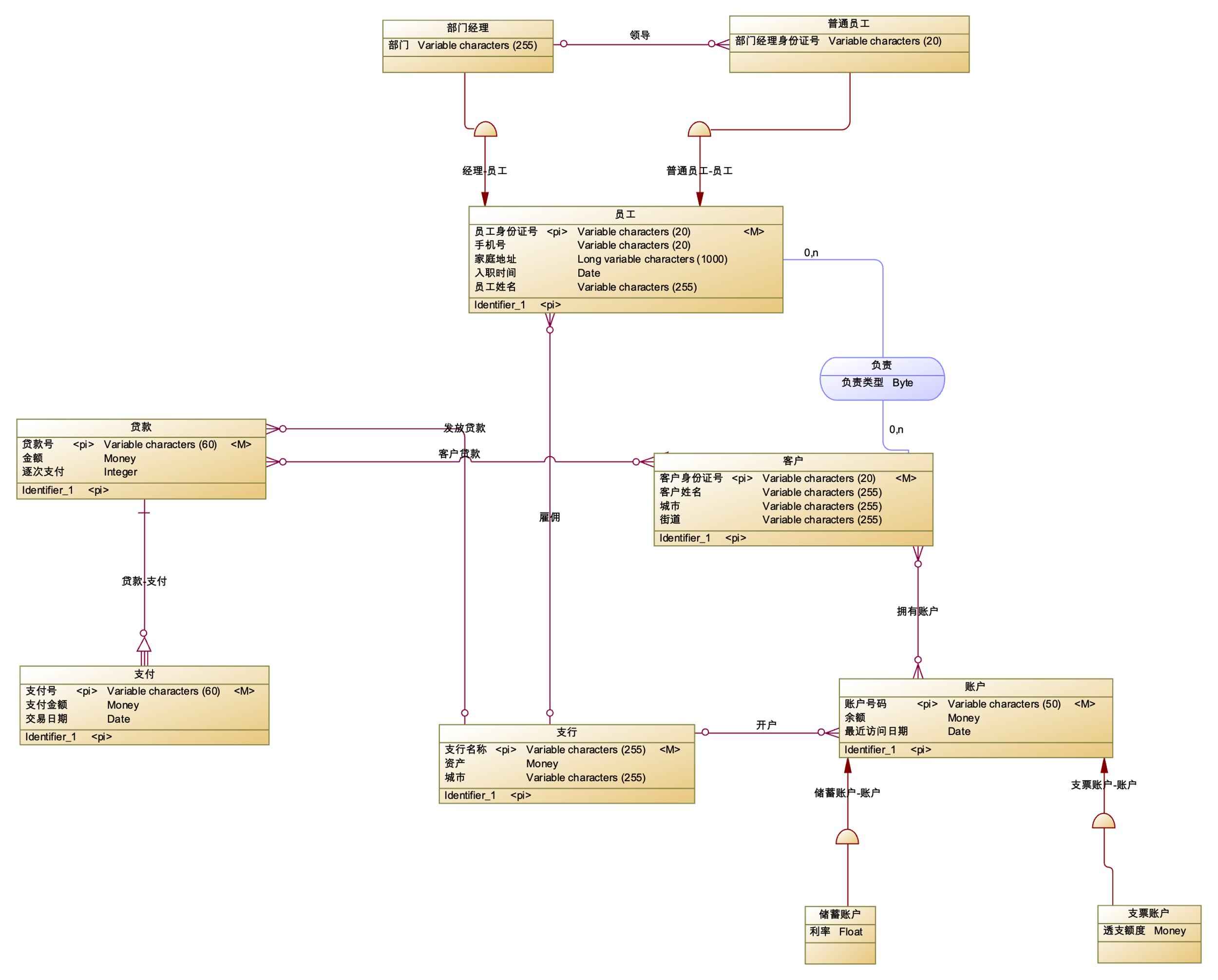


图1. 银行业务管理系统的数据库概念模型

1. **概念模型到逻辑模型的转换**
   1. 实体转换

//文字解释即可。给出每个实体的转换结果，建议用列表方式给出

|  |  |
| --- | --- |
| **实体名称** | **属性（其中主键标有下划线）** |
| 员工 | 员工身份证号，手机号，家庭地址，入职时间，员工姓名 |
| 普通员工 | 员工身份证号，部门经理身份证号 |
| 部门经理 | 员工身份证号，部门 |
| 客户 | 客户身份证号，客户姓名，城市，街道 |
| 账户 | 账户号码，余额，最近访问日期 |
| 储蓄账户 | 账户号码，利率 |
| 支票账户 | 账户号码，透支额度 |
| 支行 | 支行名称，资产，城市 |
| 贷款 | 贷款号，金额，逐次支付 |
| 支付 | 支付号，交易金额，交易日期 |

* 1. 联系转换

//文字解释即可。给出每个联系的转换结果，建议用列表方式给出

|  |  |
| --- | --- |
| **关系** | **转换结果** |
| 拥有账户 | 拥有账户表（客户身份证号码，账户号码） |
| 雇佣 | 将支行名称作为员工的一项属性 |
| 客户贷款 | 客户贷款表（客户身份证号码，贷款号） |
| 发放贷款 | 将支行名称作为贷款的一项属性 |
| 负责 | 负责（客户身份证号，员工身份证号，负责类型） |
| 贷款支付 | 将贷款号作为支付的一项属性 |
| 开户 | 将支行名称作为账户的一项属性 |

* 1. 最终的关系模式

//文字解释即可。建议用列表方式给出最终得到的关系模式

|  |  |
| --- | --- |
| **表** | **属性（其中主键标有下划线）** |
| 员工 | 员工身份证号，手机号，家庭地址，入职时间，员工姓名，支行名称 |
| 普通员工 | 员工身份证号，部门经理身份证号 |
| 部门经理 | 员工身份证号，部门 |
| 客户 | 客户身份证号，客户姓名，城市，街道 |
| 账户 | 账户号码，余额，最近访问日期，支行名称 |
| 储蓄账户 | 账户号码，利率 |
| 支票账户 | 账户号码，透支额度 |
| 支行 | 支行名称，资产，城市 |
| 贷款 | 贷款号，金额，逐次支付，支行名称 |
| 支付 | 支付号，交易金额，交易日期，贷款号 |
| 拥有账户 | 客户身份证号码，账户号码 |
| 客户贷款 | 客户身份证号码，贷款号 |
| 负责 | 客户身份证号，员工身份证号，负责类型 |

1. **MySQL数据库结构实现**
   1. Power Designer的PDM设计

//把CDM转换成面向MySQL的物理数据模型（PDM）

//得到的物理数据模型图放到这里

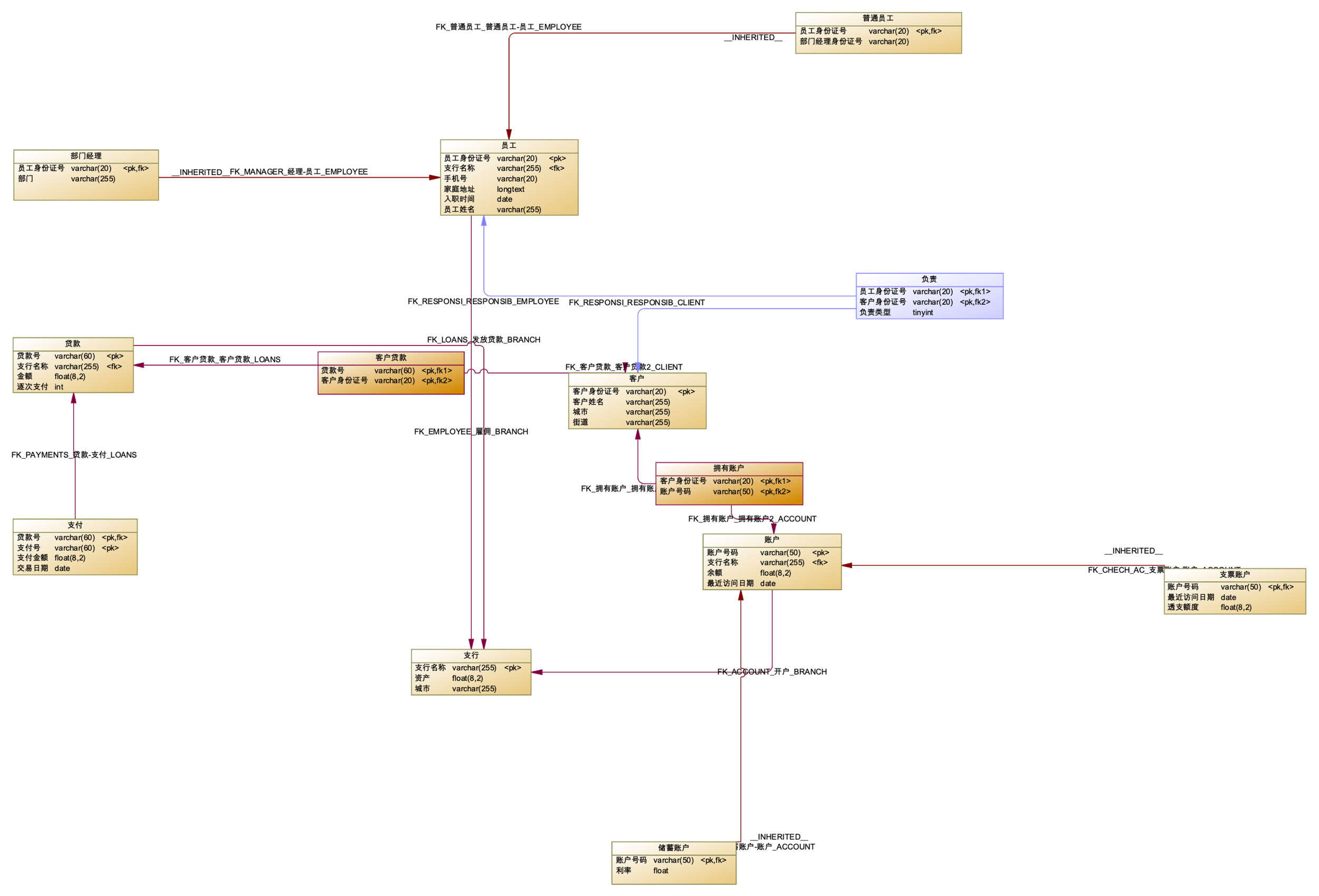


图2 银行业务管理系统的PDM设计结果

* 1. 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X。

表1. 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  (若是则给出引用的表和列名) |
| account\_id | 账户号码 | varchar(50) | 否 | 是 | 否 |
| branch\_name | 支行名称 | varchar(255) | 否 | 否 | 是 |
| balance | 余额 | float(8,2) | 否 | 否 | 否 |
| latest\_visited | 最后访问 | date | 是 | 否 | 否 |

表2. 支行(Branch)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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  (若是则给出引用的表和列名) |
| branch\_name | 支行名 | varchar(255) | 否 | 是 | 否 |
| property | 资产 | float(8,2) | 否 | 否 | 否 |
| city | 城市 | varchar(255) | 是 | 否 | 否 |

表3.支票账户 (Chech\_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  (若是则给出引用的表和列名) |
| account\_id | 账户号码 | varchar(50) | 否 | 是 | 是 |
| overdraft\_limit | 透支额度 | float(8,2) | 是 | 否 | 否 |

表4.顾客 (Client)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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  (若是则给出引用的表和列名) |
| client\_id | 客户身份证号 | varchar(20) | 否 | 是 | 否 |
| client\_name | 客户姓名 | varchar(255) | 否 | 否 | 否 |
| city | 城市 | varchar(255) | 是 | 否 | 否 |
| street | 街道 | varchar(255) | 是 | 否 | 否 |

表5.客户贷款 (ClientLoans)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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  (若是则给出引用的表和列名) |
| loan\_id | 贷款号 | varchar(60) | 否 | 是 | 是 |
| client\_id | 客户身份证号 | varchar(20) | 否 | 是 | 是 |

表6.普通员工 (Common\_employee)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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  (若是则给出引用的表和列名) |
| loan\_id | 员工身份账号 | varchar(20) | 否 | 是 | 是 |
| Leader\_id | 部门经理身份证号 | varchar(20) | 是 | 否 | 是 |

表7.储蓄账户 (Deposit\_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  (若是则给出引用的表和列名) |
| account\_id | 账户号码 | varchar(50) | 否 | 是 | 是 |
| rate | 利率 | float | 是 | 否 | 否 |

表8.员工 (Employee)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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  (若是则给出引用的表和列名) |
| employee\_id | 员工身份证号 | varchar(20) | 否 | 是 | 否 |
| employee\_name | 员工姓名 | varchar(20) | 否 | 否 | 否 |
| branch\_name | 支行名称 | varchar(255) | 是 | 否 | 是 |
| address | 家庭地址 | longtext | 是 | 否 | 否 |
| date\_be\_employ | 雇佣时间 | Date | 否 | 否 | 否 |

表9.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  (若是则给出引用的表和列名) |
| loan\_id | 贷款号 | varchar(60) | 否 | 是 | 否 |
| branch\_name | 支行名称 | varchar(255) | 是 | 否 | 是 |
| amount | 金额 | Float(8,2) | 是 | 否 | 否 |
| times | 逐次支付 | Int | 是 | 否 | 否 |

表10. 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  (若是则给出引用的表和列名) |
| loan\_id | 员工身份账号 | varchar(20) | 否 | 是 | 是 |
| department | 部门 | varchar(255) | 否 | 否 | 否 |

表11.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  (若是则给出引用的表和列名) |
| client\_id | 客户身份证号 | varchar(20) | 否 | 是 | 是 |
| account\_id | 账户号 | varchar(50) | 否 | 是 | 是 |

表12.支付 (Payments)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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  (若是则给出引用的表和列名) |
| loan\_id | 贷款号 | varchar(60) | 否 | 否 | 是 |
| Payment\_id | 支付号 | varchar(60) | 否 | 是 | 否 |
| payment\_amount | 金额 | Float(8,2) | 是 | 否 | 否 |
| Payment\_date | 支付时间 | date | 是 | 否 | 否 |

表13.负责 (Responsible)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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  (若是则给出引用的表和列名) |
| client\_id | 客户身份证号 | varchar(20) | 否 | 是 | 是 |
| employee\_id | 员工身份证号 | varchar(20) | 否 | 是 | 是 |
| Res\_type | 负责类型 | tinyint | 是 | 否 | 否 |

1. **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：从现实世界问题中抽象出概念模型，是数据库设计的关键环节。通过这一步，可以理清楚业务需求，确保整个系统的稳定性和扩展性。

（2）工具的使用：Power Designer 等专业工具在设计过程中十分高效，可以帮助快速生成概念模型和逻辑模型，减少人工出错的概率。此外，这些工具的可视化功能让复杂的模型变得直观易懂。

（3）概念模型到逻辑模型的转换：这一过程需要深入理解数据库的结构和业务逻辑。通过在概念模型基础上进行细化和优化，可以合理地安排表与表之间的关系，确保数据一致性和完整性。

（4）外键约束的设计：在逻辑模型中，正确设置外键约束不仅可以维持数据库的参照完整性，还能够预防数据异常，是数据库设计中不可缺少的一环。

（5）性能考虑：在设计数据库结构时，需要考虑查询性能和数据操作的效率。通过合理的索引设计和规范化的表设计，保证系统在处理大量数据时具备良好的性能。

（6）实际数据库实现：由逻辑模型到物理数据库的转换是最后一步。在这个过程中，要结合数据库系统的实际情况进行调整，确保数据库在实际运行中的稳定性和高效性。