银行业务管理系统数据库设计

PB19030888 张舒恒

1.概念模型设计

1.1实体设计

1.1.1部门实体

用户需求："每个支行的管理机构存储每个员工的姓名、电话号码、家庭地址、所在的部门号、部门名称、部门类型及部门经理的身份证号。"

设计：部门(部门号，部门名，部门类型，经理身份证号)，主码是部门号

1.1.2员工实体

用户需求："每个支行的管理机构存储每个员工的姓名、电话号码、家庭地址、所在的部门号、部门名称、部门类型及部门经理的身份证号。银行还需知道每个员工开始工作的日期，由此日期可以推知员工的雇佣期。"

设计：员工(员工号，员工名，员工电话，员工地址，入职日期)，主码是员工号

1.1.3客户实体

用户需求："银行的客户通过其身份证号来标识。银行存储每个客户的姓名、联系电话以及家庭住址。"

设计：客户(客户号，客户名，客户电话，客户地址)，主码是客户号

1.1.4联系人实体

用户需求："为了安全起见，银行还要求客户提供一位联系人的信息，包括联系人姓名、手机号、Email 以及与客户的关系。"

设计：联系人(联系人名，联系人邮箱，联系人电话，联系人和客户的关系)，主码是联系人名

1.1.5支行实体

用户需求："银行有多个支行。各个支行位于某个城市，每个支行有唯一的名字。银行要监控每个支行的资产。"

设计：支行(支行名，城市，资产)，主码是支行名

1.1.6账户实体

用户需求："帐户可以由多个客户所共有，一个客户也可开设多个账户，但在一个支行内最多只能开设一个储蓄账户和一个支票账户。每个帐户被赋以唯一的帐户号。银行记录每个帐户的余额、开户日期、开户的支行名以及每个帐户所有者访问该帐户的最近日期。另外，每个储蓄帐户有利率和货币类型，且每个支票帐户有透支额。 "

设计：账户(账户号，余额，开户日期)，主码是账户号，根据账户类型分为子实体储蓄账户(利率，货币类型)，支票账户(透支额)

1.1.7贷款实体

用户需求："每笔贷款由某个分支机构发放，能被一个或多个客户所共有。每笔贷款用唯一的贷款号标识。银行需要知道每笔贷款所贷金额以及逐次支付的情况（银行将贷款分几次付给客户）。虽然贷款号不能唯一标识银行所有为贷款所付的款项，但可以唯一标识为某贷款所付的款项。对每次的付款需要记录日期和金额。"

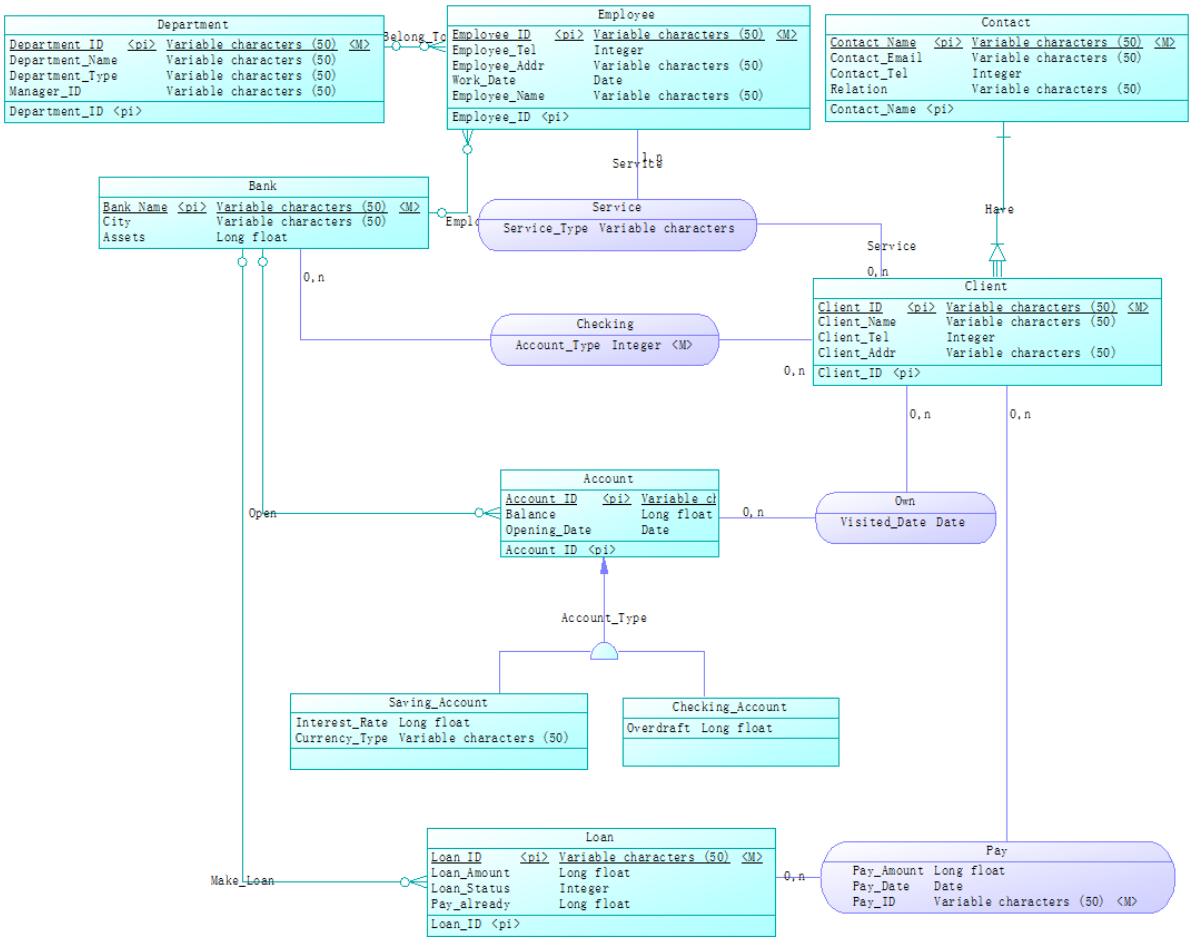
设计：贷款(贷款号，贷款总额，贷款状态，已发放金额)，主码是贷款号，其中贷款状态是记录该笔贷款是否已经全部发放给一个或多个客户

1.2联系设计

1. 联系人：客户——客户的联系人（N-1），可能有多个客户的联系人是同一个人
2. 负责服务（负责人类型）：客户——银行员工（N-N）,负责人类型是指员工是此客户的贷款负责人或银行帐户负责人，可能有多个客户由同一个员工服务，同时可能有多个员工服务同一个客户
3. 就职：银行员工——银行部门（N-1），多个员工在同一个部门工作

   银行员工——支行（N-1），多个员工在同一个支行工作
4. 开户约束(账户类型)：支行——客户（N-N），每个客户在每个支行内最多只能开一个储蓄账户和一个支票账户
5. 账户从属（最近访问日期）：客户——账户（N-N），一个帐户可以由多个客户所共有同时一个客户也能有多个账户
6. 贷款发放（贷款发放金额，贷款发放日期，贷款发放号）：客户——贷款（N-N），一个贷款可以由多个客户所共有同时一个客户也能有多个贷款

1.3ER 图



2.概念模型到逻辑模型的转换

2.1实体转换

根据1.1的实体设计导出逻辑模型：

```
Department
(
   Department_ID         varchar(50) not null,
   Department_Name       varchar(50) not null,
   Department_Type       varchar(50),
   Manager_ID   varchar(50),
   primary key (Department_ID)
);

Employee
(
   Employee_ID         varchar(50) not null,
   Employee_Name       varchar(50) not null,
   Employee_Tel        int,
   Employee_Address       varchar(50),
   Work_Date    date,
   primary key (Employee_ID)
);

Client
(
   Client_ID         varchar(50) not null,
   Client_Name       varchar(50) not null,
   Client_Tel        int,
   Client_Address       varchar(50),
   primary key (Client_ID)
);

Contact
(
   Contact_Name      varchar(50) not null,
   Contact_Email     varchar(50),
   Contact_Tel       int,
   Relation     varchar(50),
   primary key (Client_ID, Contact_Name)
);

Bank(
   Bank_Name       varchar(50) not null,
   City         varchar(50) not null,
   Assets       float(15) not null,
   primary key (Bank_Name)
);

Account
(
   Account_ID          varchar(50) not null,
   Balance       float(15),
   Opening_Date  date,
   primary key (Account_ID)
);
```

```
Checking_Account
(
   Overdraft    float(15),
   primary key (Account_ID)
);

Saving_Account
(
   Interest_Rate  float(15),
   Currency_Type  varchar(50),
   primary key (Account_ID)
);

Loan
(
   Loan_ID          varchar(50) not null,
   Loan_Amount      float(15) not null,
   Loan_Status       int default 0 not null,
   Pay_already     float(15) not null,
   primary key (Loan_ID)
);
```

2.2联系转换

对应1.2的联系设计，将联系进行转换

1. 联系人：客户——客户的联系人（N-1），考虑到客户还会和其他多种实体发生关系，所以这里选择将客户主码客户号纳入联系人模式中
2. 负责服务（负责人类型）：客户——银行员工（N-N）,新增关系模式服务（客户号，员工号，服务类型），主码是（客户号，员工号）
3. 就职：银行员工——银行部门（N-1），多个员工在同一个部门工作，银行员工——支行（N-1），将部门和支行的主码纳入员工模式中
4. 开户约束(账户类型)：支行——客户（N-N），新增关系模式开户约束（客户号，支行号，账户类型），主码是（客户号，支行号，账户类型）
5. 账户从属（最近访问日期）：客户——账户（N-N），新增关系模式从属（客户号，账户号，最近访问日期），主码是（客户号，账户号）
6. 贷款发放（贷款发放金额，贷款发放日期，贷款发放号）：客户——贷款（N-N），新增关系模式发放（贷款发放金额，贷款发放日期，贷款发放号，客户号，贷款号），主码是（贷款发放号，客户号，贷款号）
7. 将账户的主码加入到账户的子类储蓄账户和支票账户中

2.3最终的关系模式

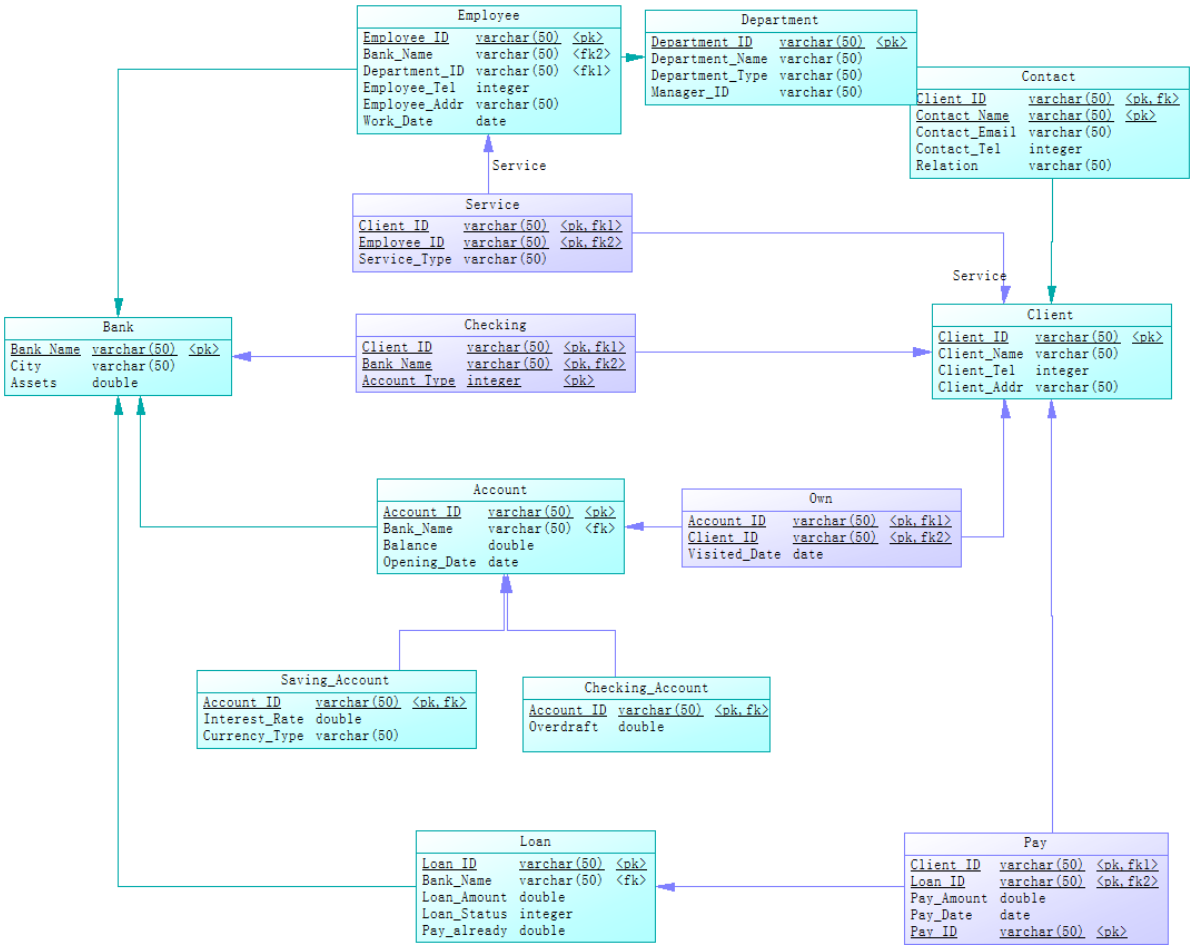
```
Department
(
   Department_ID         varchar(50) not null,
   Department_Name       varchar(50) not null,
   Department_Type       varchar(50),
   Manager_ID   varchar(50),
   primary key (Department_ID)
);

Employee
(
   Employee_ID         varchar(50) not null,
```

```
   Employee_Name       varchar(50) not null,
   Bank_Name       varchar(50) not null,
   Department_ID         varchar(50),
   Employee_Tel        int,
   Employee_Address       varchar(50),
   Work_Date    date,
   primary key (Employee_ID)
);

Client
(
   Client_ID         varchar(50) not null,
   Client_Name       varchar(50) not null,
   Client_Tel        int,
   Client_Address       varchar(50),
   primary key (Client_ID)
);

Contact
(
   Client_ID         varchar(50) not null,
   Contact_Name      varchar(50) not null,
   Contact_Email     varchar(50),
   Contact_Tel       int,
   Relation     varchar(50),
   primary key (Client_ID, Contact_Name)
);

Bank(
   Bank_Name       varchar(50) not null,
   City         varchar(50) not null,
   Assets       float(15) not null,
   primary key (Bank_Name)
);

Account
(
   Account_ID          varchar(50) not null,
   Bank_Name        varchar(50) not null,
   Balance       float(15),
   Opening_Date  date,
   primary key (Account_ID)
);

Checking_Account
(
   Account_ID         varchar(50) not null,
   Overdraft    float(15),
   primary key (Account_ID)
);

Saving_Account
(
   Account_ID           varchar(50) not null,
   Interest_Rate  float(15),
   Currency_Type  varchar(50),
   primary key (Account_ID)
);
```

```
Loan
(
   Loan_ID         varchar(50) not null,
   Bank_Name       varchar(50) not null,
   Loan_Amount     float(15) not null,
   Loan_Status      int default 0 not null,
   Pay_already    float(15) not null,
   primary key (Loan_ID)
);

# 新增的关系模式
Service
(
   Client_ID         varchar(50) not null,
   Employee_ID         varchar(50) not null,
   Service_Type       varchar(50),
   primary key (Client_ID, Employee_ID)
);

Checking
(
   Client_ID         varchar(50) not null,
   Bank_Name       varchar(50) not null,
   Account_Type       int not null,
   primary key (Client_ID, Bank_Name, Account_Type)
);

Own
(
   Client_ID         varchar(50) not null,
   Visited_Date date,
   Account_ID         varchar(50),
   primary key (Client_ID, Account_ID)
);

Pay
(
   Client_ID         varchar(50) not null,
   Loan_ID         varchar(50) not null,
   Pay_ID         varchar(50) not null,
   Pay_Amount     float(15),
   Pay_Date     date,
   primary key (Client_ID, Loan_ID, Pay_ID)
);
```

3.MySQL 数据库结构实现

3.1 Power Designer 的 PDM 图



3.2 数据库表定义

表一：部门表(Department)

列名	中文含义	类型	是否允许为空	主键	外键
Department_ID	部门号	varchar(50)		Y	
Department_Name	部门名	varchar(50)			
Department_Type	部门类型	varchar(50)	Y		
Manager_ID	经理号	varchar(50)	Y		

表二：员工表(Employee)

列名	中文含义	类型	是否允许为空	主键	外键
Employee_ID	员工号	varchar(50)		Y	
Employee_Name	员工名	varchar(50)			
Employee_Tel	员工电话	int	Y		
Employee_Address	员工地址	varchar(50)	Y		
Work_Date	入职日期	date	Y		
Bank_Name	支行名	varchar(50)			Y
Department_ID	部门名	varchar(50)			Y

表三：客户表(Client)

列名	中文含义	类型	是否允许为空	主键	外键
Client_ID	客户号	varchar(50)		Y	
Client_Name	客户名	varchar(50)			
Client_Tel	客户电话	int	Y		
Client_Address	客户地址	varchar(50)	Y		

表四：联系人表(Contact)

列名	中文含义	类型	是否允许为空	主键	外键
Client_ID	客户号	varchar(50)		Y	Y
Contact_Name	联系人名	varchar(50)		Y	
Contact_Email	联系人邮箱	varchar(50)	Y		
Contact_Tel	联系人电话	int	Y		
Relation	联系人和客户关系	varchar(50)	Y		

表五：支行表(Bank)

列名	中文含义	类型	是否允许为空	主键	外键
Bank_Name	支行名	varchar(50)		Y	
City	支行所在城市	varchar(50)			
Assets	支行资产	float(15)			

表六：账户表(Account)

列名	中文含义	类型	是否允许为空	主键	外键
Account_ID	账户名	varchar(50)		Y	
Bank_Name	支行名	varchar(50)			Y
Balance	账户余额	float(15)	Y		
Opening_Date	开户日期	date	Y		

表七：支票账户表(Checking_Account)

列名	中文含义	类型	是否允许为空	主键	外键
Account_ID	账户名	varchar(50)		Y	Y
Overdraft	透支额度	float(15)	Y		

表八：储蓄账户表(Saving_Account)

列名	中文含义	类型	是否允许为空	主键	外键
Account_ID	账户名	varchar(50)		Y	Y
Interest_Rate	利率	float(15)	Y		
Currency_Type	货币类型	varchar(50)	Y		

表九：贷款表(Loan)

列名	中文含义	类型	是否允许为空	主键	外键
Loan_ID	贷款号	varchar(50)		Y	
Bank_Name	支行名	varchar(50)			Y
Loan_Amount	贷款总额	float(15)			
Loan_Status	贷款状态	int			
Pay_already	已发放贷款金额	float(15)			

表十：服务表(Service)

列名	中文含义	类型	是否允许为空	主键	外键
Client_ID	客户号	varchar(50)		Y	Y
Employee_ID	员工号	varchar(50)		Y	Y
Service_Type	服务类型	varchar(50)	Y		

表十一：开户约束表(Checking)

列名	中文含义	类型	是否允许为空	主键	外键
Client_ID	客户号	varchar(50)		Y	Y
Bank_Name	支行名	varchar(50)		Y	Y
Account_Type	账户类型	int		Y	

表十二：账户从属表(Own)

列名	中文含义	类型	是否允许为空	主键	外键
Client_ID	客户号	varchar(50)		Y	Y
Visited_Date	最近访问日期	date	Y		
Account_ID	账户号	varchar(50)		Y	Y

表十三：贷款发放表(Pay)

列名	中文含义	类型	是否允许为空	主键	外键
Client_ID	客户号	varchar(50)		Y	Y
Loan_ID	贷款号	varchar(50)		Y	Y
Pay_ID	贷款发放号	varchar(50)		Y	
Pay_Amount	贷款发放金额	float(15)	Y		
Pay_Date	贷款发放日期	date	Y		

4.总结与体会

数据库模式的设计可以按照流程一步步完成，在设计过程中不仅要充分考虑用户需求以及设计原则，还要灵活变通以便于数据库将来的增删改查和继承复用。