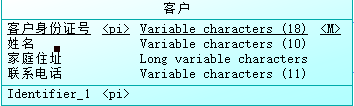
银行业务管理系统数据库设计

PB19111682 李清伟

1 概念模型设计

* 1. 实体设计

**客户**

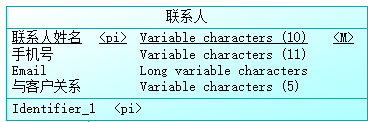


具有客户身份证号、姓名、联系电话、家庭住址属性。

其中客户身份证号是主键。

根据“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。”，客户有客户身份证号、姓名、联系电话、家庭住址等属性，且客户身份证号是唯一标识，作为主键。

**联系人**

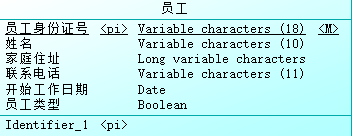


具有联系人姓名、手机号、Email、与客户关系属性。

联系人姓名作为主键。

根据“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。”，联系人具有联系人姓名、手机号、Email、与客户关系属性。因为只有一位联系人，所以可以将联系人姓名作为联系人标识。

**员工**

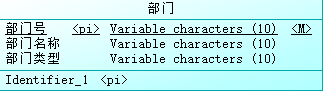


具有员工身份证号、姓名、家庭住址、联系电话、开始工作日期、员工类型属性。

其中员工身份证号作为员工的唯一标识，作为主键。

根据“银行员工也通过身份证号来标识。”以及“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”得到每个员工具有员工身份证号、姓名、电话号码、家庭住址的属性。以员工身份证号进行标识，因此作为主键。根据“银行还需知道每个员工开始工作的日期”，员工有开始工作日期属性。根据“员工分为部门经理和普通员工”，员工有两种类型，用boolean型的员工类型属性表示，0表示部门经理，1表示普通员工。

**部门**

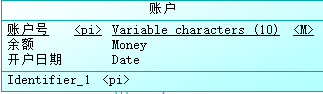


具有部门号，部门名称，部门类型属性。

其中部门号作为部门唯一标识，作为主键。

根据“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”，由于部门内部具有部门号、部门名称、部门类型属性，因此作为一个实体独立出来。

**账户**



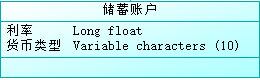
具有账户号、余额、开户日期属性。

其中账户号作为账户唯一标识，作为主键。

根据“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”，账户具有账户号、余额、开户日期属性。其中每个账户被赋以唯一的账户号，因此账户号作为主键。

由于储蓄账户和支票账户存在共同属性，因此提出账户这一抽象实体。

**储蓄账户**



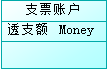
是账户的子类。

具有账户号、余额、开户日期、利率、货币类型属性

其中账户号作为唯一标识，为主键。

储蓄账户作为账户，继承账户的所有属性。根据“每个储蓄帐户有利率和货币类型”，储蓄账户还具有利率和货币类型属性。

**支票账户**



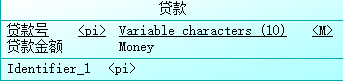
是账户的子类。

具有账户号、余额、开户日期、透支额属性

其中账户号作为唯一标识，为主键。

支票账户作为账户，继承账户所有属性。根据“每个支票帐户有透支额”，支票账户还具有透支额属性。

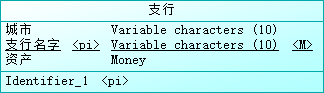
**贷款**



具有贷款号、贷款金额属性

根据“每笔贷款用唯一的贷款号标识”，贷款号作为贷款的唯一标识，根据“银行需要知道每笔贷款所贷金额”，贷款金额是贷款的一个属性。

支行

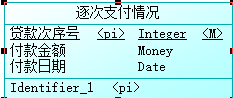


具有城市、支行名称、资产属性

其中支行名称唯一标识支行，作为主键。

根据“各个支行位于某个城市，每个支行有唯一的名字”，支行具有城市、支行名称属性。因为每个支行有唯一的名字，因此支行名称唯一标识支行，作为主键。根据“银行要监控每个支行的资产。”，支行具有资产属性。

逐次支付情况



具有贷款次序号、付款金额、付款日期属性

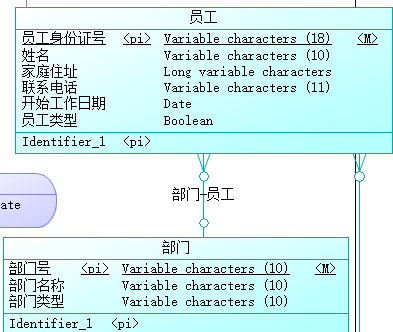
贷款次序号的含义为一笔贷款中逐次支付的次序号

该实体依赖于贷款，贷款次序号与贷款号结合起来唯一标识一次付款，此由“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。”得到。

根据“对每次的付款需要记录日期和金额”，逐次支付情况有付款日期和付款金额的属性。

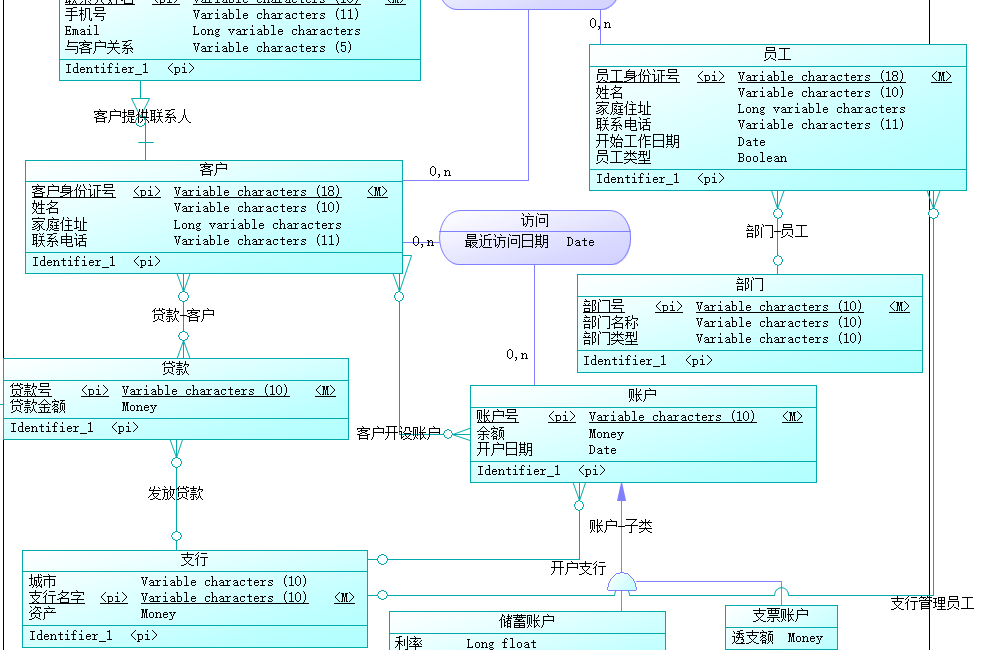
1.2 联系设计

**部门-员工**



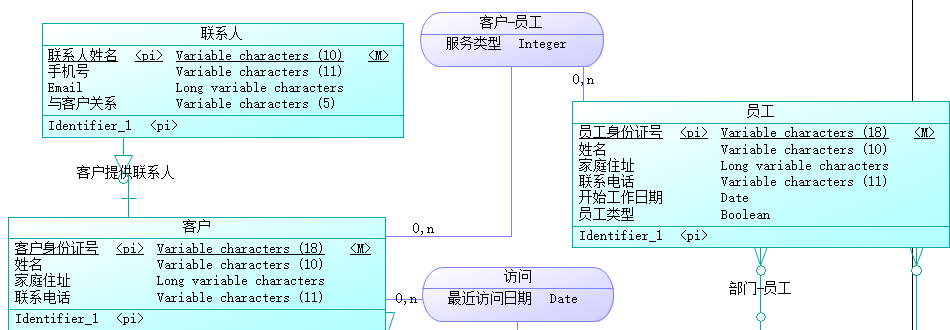
根据“并且每个员工只允许在一个部门内工作”，员工和部门是多对1的关系，根据“每个支行的管理机构存储每个员工的姓名、电话号码、家庭地址、所在的部门号、部门名称、部门类型及部门经理的身份证号。”，部门与员工有该关系，部门经理的省份证号可以通过员工类型进行判断。根据“每个部门经理都负责领导其所在部门的员工”，员工中的领导也可以通过这个关系得到其领导的员工。

**支行管理员工**



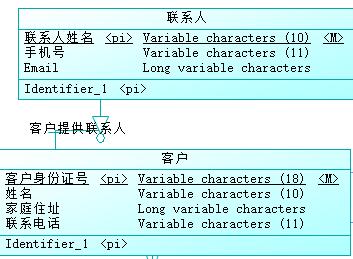
根据“每个支行的管理机构存储每个员工的姓名、电话号码、家庭地址、所在的部门号、部门名称、部门类型及部门经理的身份证号。”每个支行管理多个员工，每个员工在一个支行内部被管理，所以支行与员工是1对多的关系。

**客户-员工**



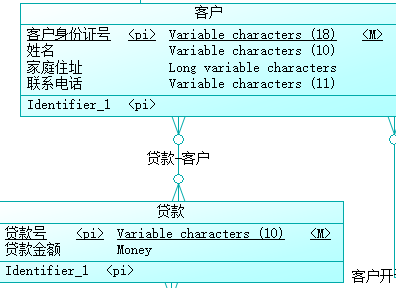
根据“客户可能和某个银行员工发生联系，该员工是此客户的贷款负责人或银行帐户负责人 ”，客户与员工之间存在联系，联系具有服务类型属性，表示贷款负责人或者银行账户负责人。在该联系中，客户可能不出现，或出现1次，员工可能不出现，或出现多次。服务类型为贷款负责人，则为1，银行账号负责人则为0。

**客户提供联系人**



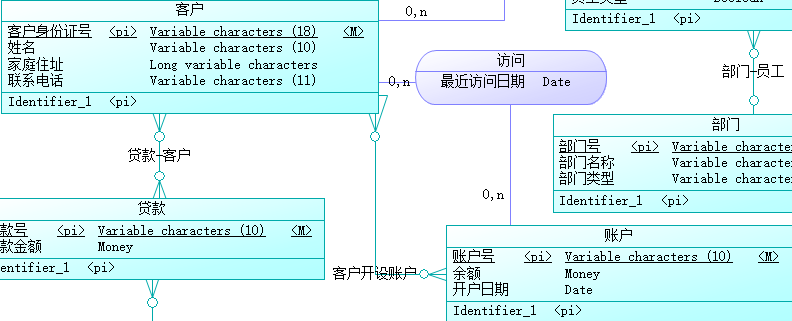
根据“银行还要求客户提供一位联系人的信息”，客户与联系人之间存在依赖关系，联系人是弱实体，依赖于客户，如果客户被删除，则联系人也相应的应该被删除。

**贷款-客户**



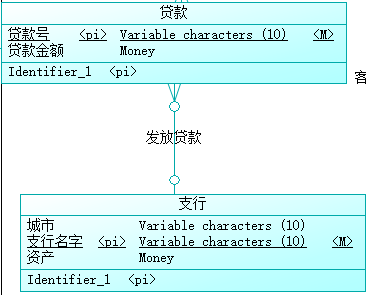
根据“每笔贷款……能被一个或多个客户所共有”，一笔贷款可以和多个客户发送关系，一个客户也可以有多笔贷款。因此是多对多的关系。

**客户开设账户**



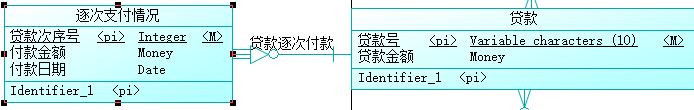
根据“帐户可以由多个客户所共有，一个客户也可开设多个账户”，客户与账户有多对多的开设关系。

**发放贷款**



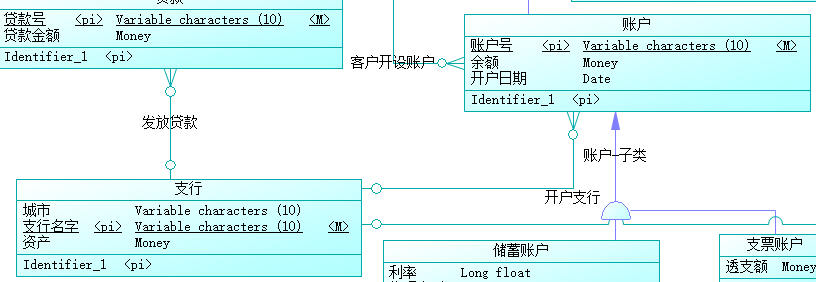
根据“每笔贷款由某个分支机构发放”，一笔贷款对应一个支行，而一个支行可以对应多笔贷款，所以是一对多的关系。

**贷款逐次付款**



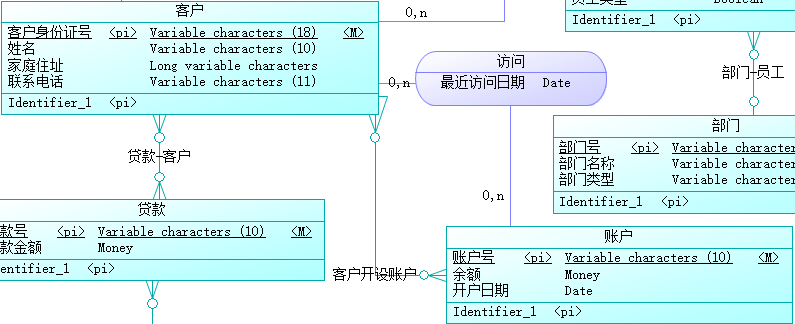
根据“银行需要知道每笔贷款所贷金额以及逐次支付的情况（银行将贷款分几次付给客户）。虽然贷款号不能唯一标识银行所有为贷款所付的款项，但可以唯一标识为某贷款所付的款项”，逐次支付情况依赖于贷款

**开户支行**



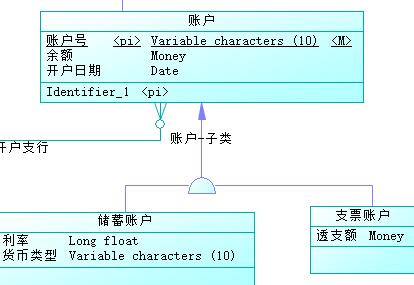
根据“银行记录每个帐户的余额、开户日期、开户的支行名”，每个账户对应一个开户支行，一个开户支行对应多个账户，因此是一对多的关系。

**访问**



根据“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”，客户与账户之间存在访问关系，具有最近访问日期属性。

**账户子类**

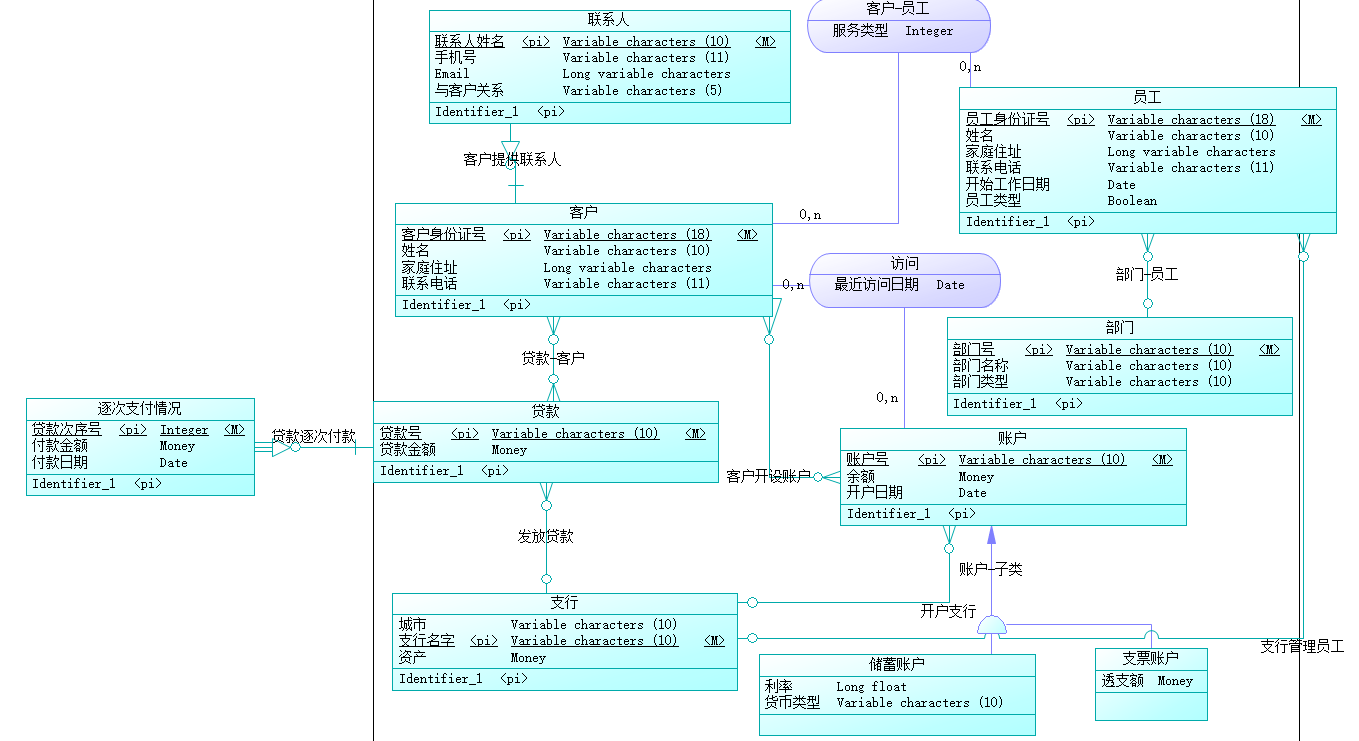


继承关系

储蓄账户和支票账户继承账户的所有属性，因为两者具有相同的账户号、余额、开户日期属性。账户不能被generate，因为是抽象实体。

**备注：**“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。”用存储过程动态实现。

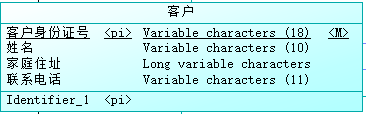
1.3 Power Designer 的 ER 图

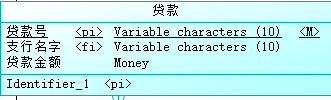


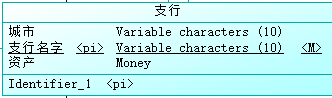
2 概念模型到逻辑模型的转换

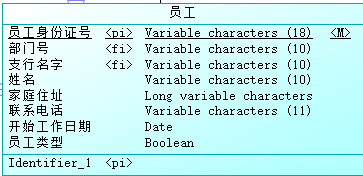
2.1 实体转换

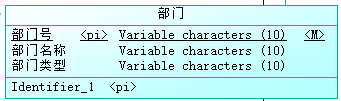
强实体转化：客户、贷款、支行、员工、部门、账户这几个强实体直接转化为为对应的关系模型，标识作为关系模式的主码

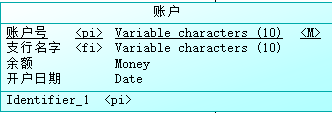




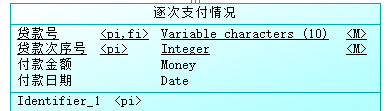


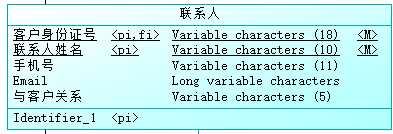




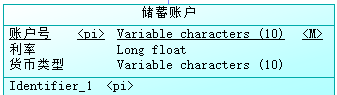


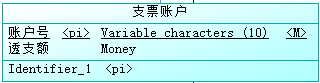
弱实体转化：在概念模型中，弱实体“逐次支付情况”依赖于强实体“贷款”，弱实体“联系人”依赖于强实体“客户”，因此在将逐次支付情况和联系人转为对应关系模式后，分别添加贷款和客户的标识，将主码设置为弱实体标识+强实体标识，如下图所示。





子类转换：储蓄账户、支票账户作为账户的子类，继承账户的主码，在生成的关系模式中中加入账户的主码，并且由于两个子类自身没有表示，所以将账户的主码作为关系模式的主码。

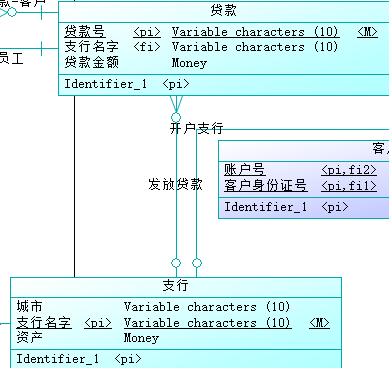




2.2 联系转换

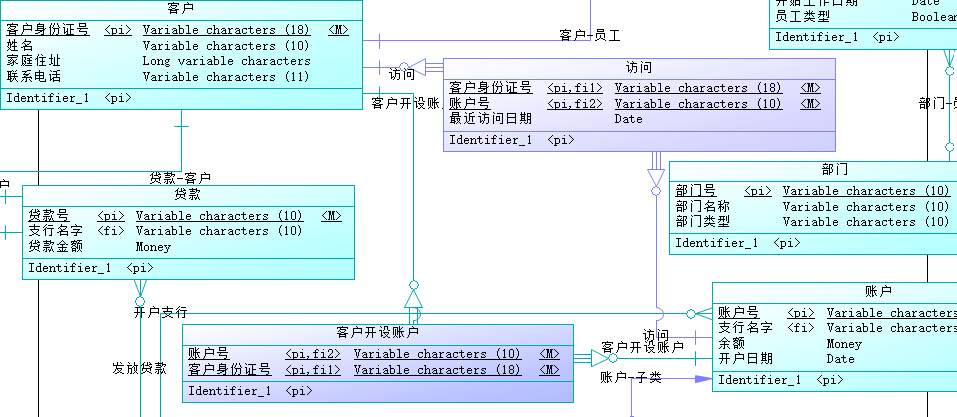
**发放贷款**

贷款与支行之间是多对一的关系，因此将支行的主码支行名称加入贷款关系模式中。如下图



**客户开设账户**

由于客户与账户是多对多关系，因此生成新关系模式“客户开设账户”，包含客户的主码客户身份证号，账户的主码账户号。主码设置为客户身份证号和账户号。如下图

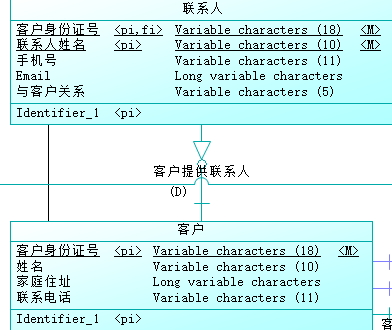


**访问**

由于客户访问账户的关系是多对多的，因此生成新关系模式“访问”（如上图所示），包含客户主码客户身份证号，账户主码账户号，最近访问日期是联系的属性。主码设置为客户身份证号和账户号。

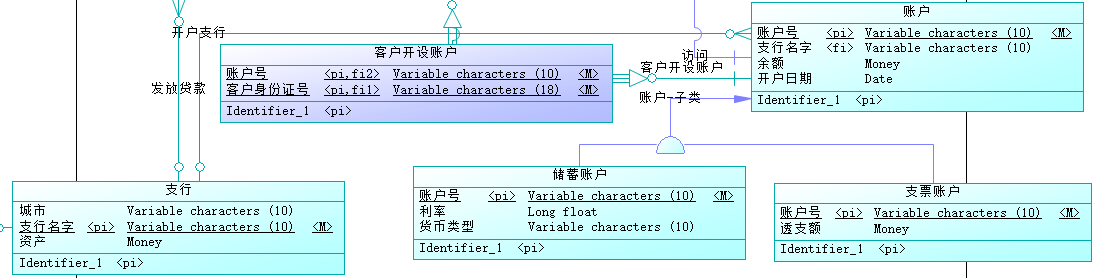
**客户提供联系人**

由于该关系是依赖关系，因此将客户的主码客户身份证号加入弱实体联系人的主码中即可。如下图



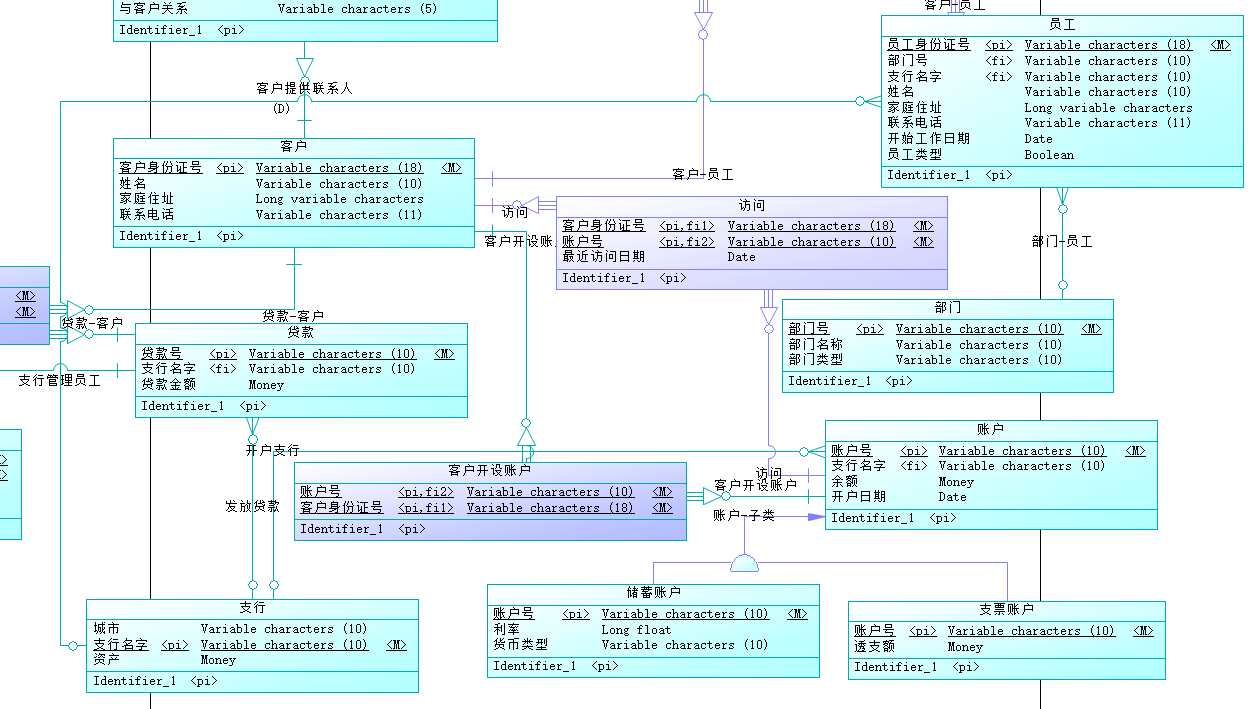
**开户支行**

由于支行与账户是一对多的关系，因此将支行的主码支行名字加入账户的关系模式中。如下图



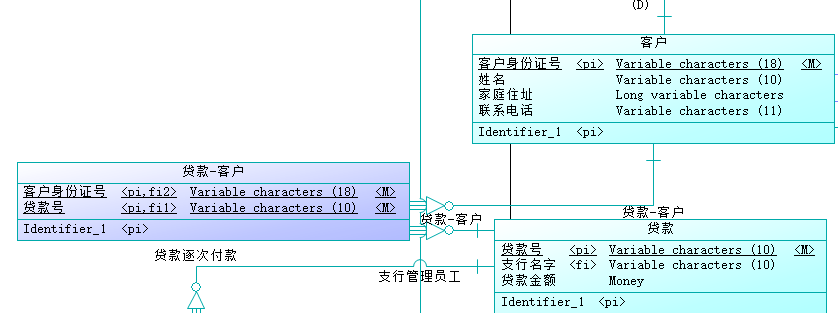
**支行管理员工**

由于支行与员工之间是一对多的关系，因此将支行的主码支行名字加入员工的关系模式中。如下图



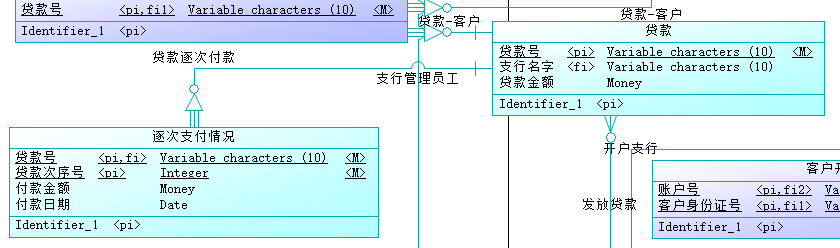
**贷款-客户**

由于贷款和客户之间是多对多的关系，因此生成新的关系模式贷款客户，包含客户的主码客户身份证号，贷款的主码贷款号。主码设置为客户身份证号和贷款号。如下图



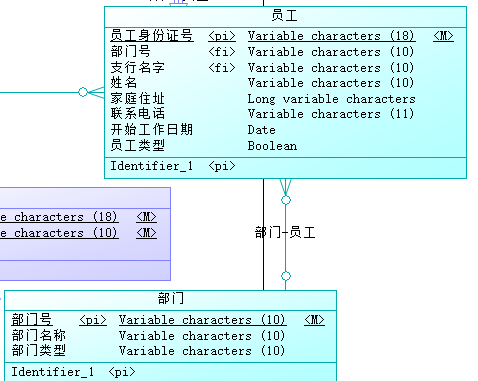
**贷款逐次付款**

该关系是依赖关系，因此将贷款的主码贷款号加入逐次支付情况的关系模式中，并将主码设置为两者标识的组合：贷款号和贷款次序号。如下图



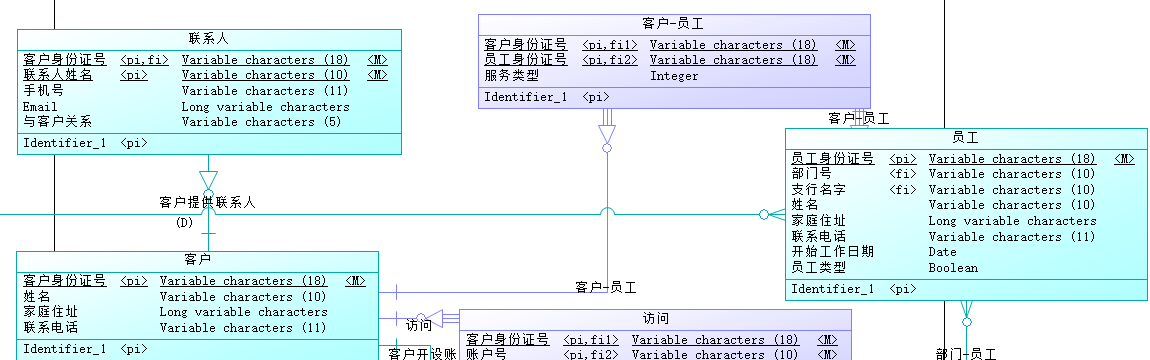
**部门-员工**

部门与员工是一对多的关系，因此将部门的主码部门号加入员工的关系模式中。如下图



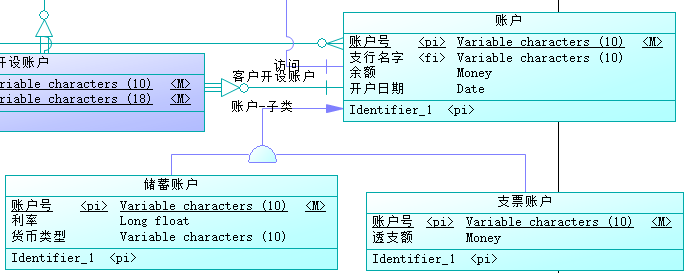
**客户-员工**

由于客户与员工是多对多的关系，因此新生成关系模式客户-员工，包含客户的主码客户身份证，员工的主码员工身份证号，以及联系属性服务类型。主码设置为两个实体的标识组合。如下图

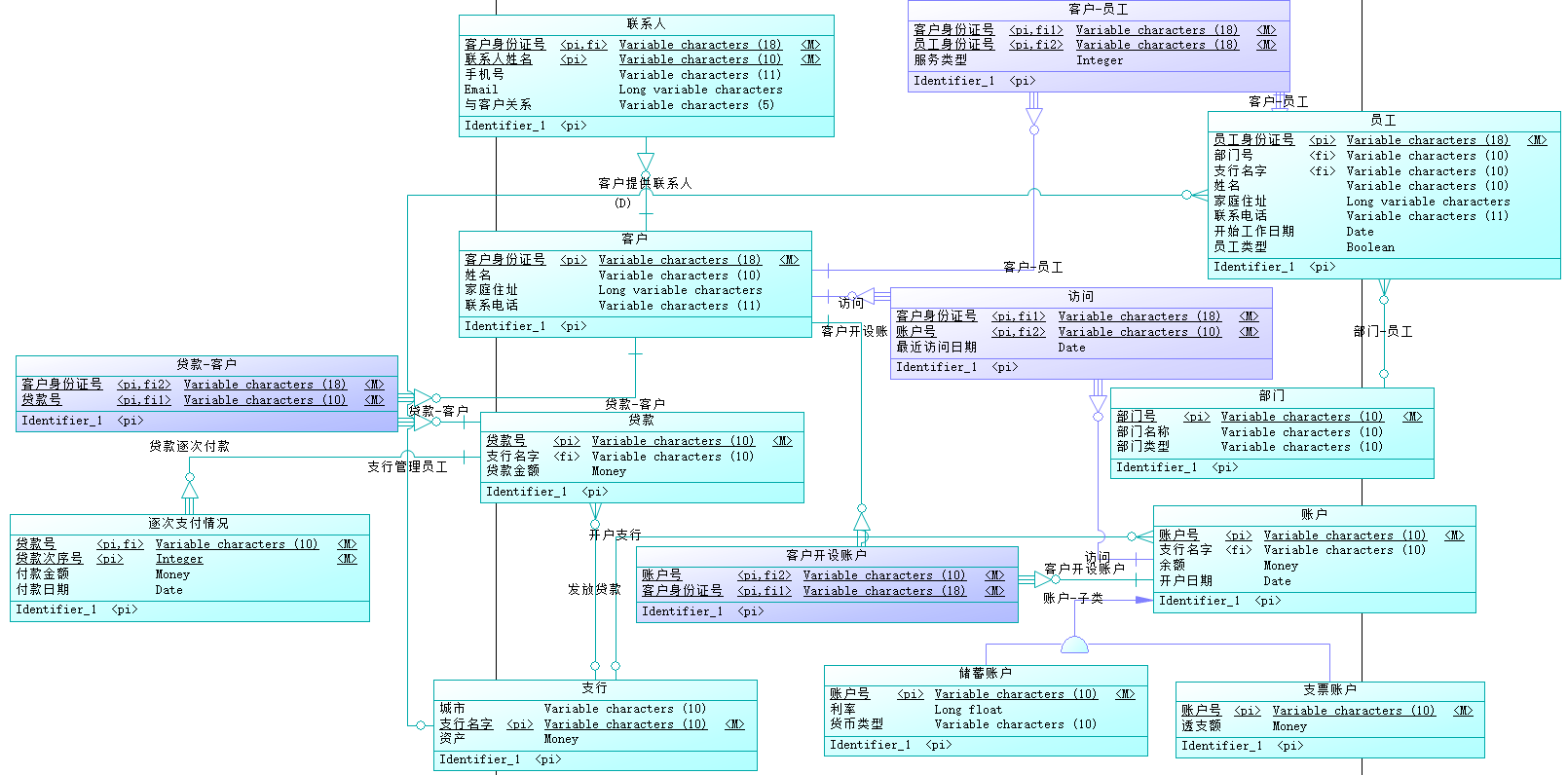


**账户-子类**

由于储蓄账户、支票账户是账户的子类，因此将账户的主码账户号分别加入两者的关系模式中，并且结合子类的标识（概念模型中两者没有标识）形成子类的主码。如下图

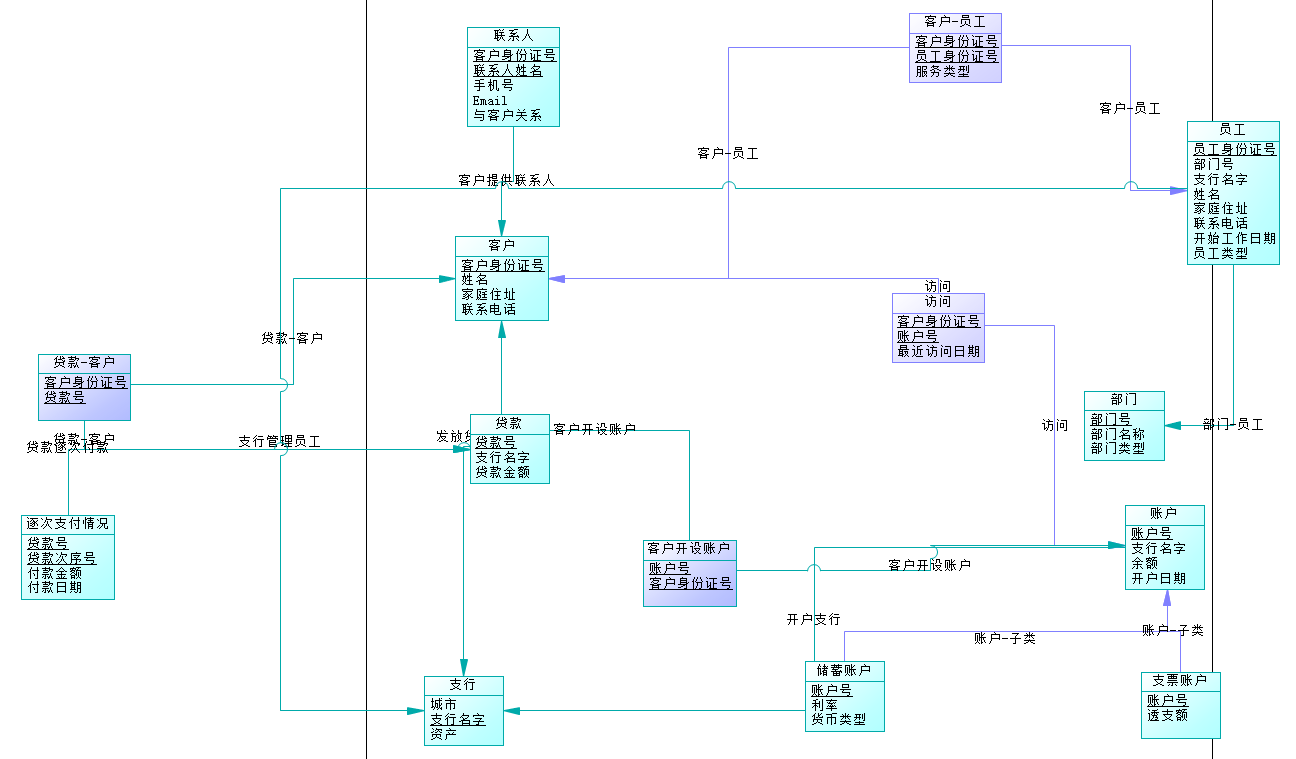


2.3 最终的关系模式



3 MySQL 数据库结构实现

3.1 Power Designer 的 PDM 图



3.2 数据库表定义

下表中T代表True、允许、是，F代表False、不允许、不是

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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(18) | F | T | F |
| name | 姓名 | varchar(10) | T | F | F |
| home\_address | 家庭住址 | long varchar | T | F | F |
| phone\_num | 联系电话 | varchar(11) | T | F | F |

表 1 客户(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(18) | F | T | client.client\_id |
| contacter\_name | 联系人姓名 | varchar(10) | F | T | F |
| contacter\_phone | 手机号 | varchar(11) | T | F | F |
| contacter\_email | Email | long varchar | T | F | F |
| contacter\_relat | 与客户关系 | varchar(5) | T | F | F |

表 2 联系人(contacter)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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) |
| staff\_id | 员工身份证号 | varchar(18) | F | T | F |
| dept\_id | 部门号 | varchar(10) | T | F | dept.dept\_id |
| branch\_name | 支行名字 | varchar(10) | T | F | branch.branch\_name |
| name | 姓名 | varchar(10) | T | F | F |
| home\_address | 家庭住址 | long varchar | T | F | F |
| phone\_num | 联系电话 | varchar(11) | T | F | F |
| start\_date | 开始工作日期 | date | T | F | F |
| staff\_type | 员工类型 | smallint | T | F | F |

表 3 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  (若是则给出引用的表和列名) |
| dept\_id | 部门号 | varchar(10) | F | T | F |
| dept\_name | 部门名称 | varchar(10) | T | F | F |
| dept\_type | 部门类型 | varchar(10) | T | F | F |

表 4 部门(dept)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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(10) | F | T | F |
| branch\_name | 支行名字 | varchar(10) | T | F | branch.branch\_name |
| loan\_money | 贷款金额 | numeric(8,2) | T | F | F |

表 5 贷款(loan)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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(10) | F | T | loan.loan\_id |
| loan\_pay\_no | 贷款次序号 | integer | F | T | F |
| loan\_pay\_money | 付款金额 | numeric(8,2) | T | F | F |
| loan\_pay\_date | 付款日期 | date | T | F | F |

表 6 逐次支付情况(times\_pay)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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) |
| city | 城市 | varchar(10) | T | F | F |
| branch\_name | 支行名字 | varchar(10) | F | T | F |
| branch\_money | 资产 | numeric(8,2) | T | F | F |

表 7 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) |
| account\_id | 账户号 | varchar(10) | F | T | account.account\_id |
| interest\_rate | 利率 | double | T | F | F |
| currency\_type | 货币类型 | varchar(10) | T | F | F |

表 8 储蓄账户(saving\_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 | varchar(10) | F | T | account.account\_id |
| overdraft | 透支额 | numeric(8,2) | T | F | F |

表 9 支票账户(checking\_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 | varchar(10) | F | T | F |
| branch\_name | 支行名字 | varchar(10) | T | F | branch.branch\_name |
| balance | 余额 | numeric(8,2) | T | F | F |
| account\_create\_date | 开户日期 | date | T | F | F |

表 10 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) |
| client\_id | 客户身份证号 | varchar(18) | F | T | client.client\_id |
| staff\_id | 员工身份证号 | varchar(18) | F | T | staff.staff\_id |
| service\_type | 服务类型 | integer | T | F | F |

表 11 客户-员工(client\_staff\_service)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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(18) | F | T | client.client\_id |
| account\_id | 账户号 | varchar(10) | F | T | account.account\_id |
| recent\_access\_date | 最近访问日期 | date | T | F | F |

表 12 访问(client\_access\_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 | varchar(10) | F | T | account.account\_id |
| client\_id | 客户身份证号 | varchar(18) | F | T | client.client\_id |

表 13 客户开设账户(client\_own\_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) |
| client\_id | 客户身份证号 | varchar(18) | F | T | client.client\_id |
| loan\_id | 贷款号 | varchar(10) | F | T | loan.loan\_id |

表 14 贷款-客户(loan\_client)

4 总结与体会

1. 仅仅通过一两次的需求理解很难做到完整的设计，需要多轮交流迭代才能做充分。
2. 有些内容即使用概念模型也很难表达清楚，需要记录并以动态方式（存储过程）实现。