第四章 数据库设计与实现

复杂工程问题——数据库建模设计实践

一、数据库建模设计工程问题探讨

1、数据库建模设计如何确定合理的实体数据类型？

答：根据所要存储的数据属性、遵循一定的数据库设计原则，合理选择实体数据类型，改善数据库系统的资源利用率、吞吐率，及减少事务执行的平均等待时间。确定数据类型的原则如：

(1) 使用可存下数据的最小的数据类型；
(2) 使用简单地数据类型，如 int 要比 varchar 类型在数据处理上更简单；
(3) 尽可能使用 Not Null 定义字段，这是由数据库的特性决定的，因为非 Not Null 的数据可能需要一些额外的字段进行存储，这样就会增加一些 IO。

2、数据库建模设计如何确定合理的实体标识符？

答：应从实体的各个属性中选出一个代表性的属性作为实体标识符。该标识符类似主键，其值要求唯一，且不允许空值。如果实体中找不到任何单个属性可以做标识符，就必须选取多个属性的组合作为标识符。

3、数据库建模设计中如何体现业务规则约束？

答：概念数据模型中，标识实体联系符号，并给出联系名称；逻辑数据模型中，标识主键、外键并进行规范化处理；物理数据模型设计中，为每个列定义特性（数据类型、空值状态、默认值及取值约束）。

4、针对大型复杂信息系统，如何进行数据库建模设计？

答：采用混合设计策略。混合设计策略即融合以上设计策略对系统数据库进行建模设计，同时应用多种设计策略进行系统数据建模，可避免单一设计策略导致的数据库建模设计局限。例如，在针对大型组织机构的复杂数据建模设计中，首先可采用自顶向下策略分割业务数据范围，每个业务建立一个数据模型，然后在每个业务数据模型设计中采用自底向上策略，最后解决各业务数据模型之间的实体冲突、实体共享、实体冗余等问题。

5、数据库建模设计如何解决复杂系统的数据一致性、数据共享问题？

答：进行数据库规范化设计可以解决数据冗余、数据完整性与一致性问题。进行数据库并发控制与数据库锁机制可以解决数据共享问题。

6、数据库建模设计如何解决数据冗余、数据依赖问题？

答：进行数据库规范化设计使冗余数据被减少到最低程度、设计合理的表间数据依赖关系。规范化设计第一范式解决重复或可细分属性（冗余）、第二范式消除关系中的属性部分函数依赖、第三范式解决非主键属性传递函数依赖、BCNF 范式解决引起数据冗余的数据依赖、第四范式消除多值依赖、第五范式消除连接依赖。

7、数据库建模设计如何平衡规范化与性能优化？

答：以规范化设计手段对具体功能模块数据库关系表结构进行调整，以达到局部优化效

果；再通过逆规范化有效地减少数据访问的复杂度及关系表的连接操作次数。在大多数联机事务处理类型的数据库应用场景下，第三范式或 BCNF 范式就足以达到规范化设计需求。

二、数据库工程实践案例问题探讨

针对一个图书借阅管理系统数据库建模设计，探讨该数据库建模设计实践。

1、在图书借阅管理系统概念数据模型设计中，如何定义业务约束规则？

答：通过 E-R 模型分析出实体间联系的多重性、参与性、继承性等约束关系。如：“图书”实体与“借阅者”实体，一本图书可能被多个借阅者借阅或没有被借阅过，同时一个借阅者可能借阅多本图书或一本都没借阅过。所以“图书”与“借阅者”之间是“多对多”联系。进行属性定义时，还应该给出属性的数据类型及取值约束。如：图书 ISBN 号应该使用 Varchar 数据类型、借阅者 ID 取值不为空等。

2、在图书借阅管理系统概念数据模型设计中，如何解决不同子模型实体之间的冲突？

答：修改、增加、删除引起冲突的父模型或子模型属性。

3、在图书借阅管理系统概念数据模型设计中，如何实现不同子模型实体共享？

答：在父模型中建立相应的实体，子模型继承该父模型即可实现实体共享。共享实体在父模型中更改时，在子模型中也随之更改。

4、在图书借阅管理系统逻辑数据模型设计中，如何完善实体-联系模型？

答：设计人员要分析实体的数据特征，并将它们在模型图中定义出来。在进行属性定义时，设计人员需要给出属性名称、数据类型、取值约束等说明；同时，设计人员还需要从实体的各个属性中，选出一个代表性的属性作为实体的标识符。该标识符类似主键，其值要求唯一，且不允许空值。针对一些属性的取值业务规则，设计人员还需要定义属性值域，以限定属性的取值范围。在系统概念数据模型检查中，我们通常需要检查模型中是否存在实体、属性及联系的冲突问题、冗余问题、遗漏问题、错误关联问题等。同时，我们也需要从业务数据处理角度，检验模型是否支持业务处理。

5、在图书借阅管理系统物理数据模型设计中，如何设计索引？

答：可根据数据库系统在实际使用过程中的查询需要来设计索引。如，针对用户表可使用用户 ID 创建索引；针对图书表可使用图书 ID 或图书借阅日期创建索引；针对借阅表可使用用户 ID 及图书 ID 复合创建索引。

6、在图书借阅管理系统物理数据模型设计中，如何设计视图？

答：可根据数据库系统在实际使用过程中的查询需要来设计视图，将常用的连接查询保存为视图可提高查询的效率。比如将借阅表和用户表及图书表连接后保存为新视图，方便用户或管理员查询借阅图书或用户信息。

7、在图书借阅管理系统物理数据模型设计中，如何设计存储过程？

答：将常用的数据库操作封装为存储过程，可分为带参数型与不带参数型，提高数据库管理的效率。

8、在图书借阅管理系统物理数据模型设计中，如何设计参照约束级联操作？

答：将 E-R 模型转换设计为关系模型时，按以下方法转换为表之间的参照约束：（1）1:1 实体联系将一个表的主键放入另一个表中作为外键；（2）1:N 实体联系将 1 端关系表的主键放入 N 端关系表中作为外键；（3）M:N 实体联系需要增加一个关系表，并与两个表均建立参照关系。

9、在图书借阅管理系统数据库建模设计中，如何解决模型错误问题？

答：先保存模型中实体的数据，分析模型错误是由函数依赖，数据异常还是关系冲突引起的。将模型中关系库修改优化进行规范化设计，使其满足六大范式并在事务管理控制的过程中合理运用共享锁与排他锁使其正常运行。如果错误仍未解决，则删除模型的物理数据实体，重建概念数据模型设计。

三、 根据下面的需求，设计银行数据库.

银行数据库的数据需求：

1）银行有多个支行。每个支行位于某个城市，有唯一的名字标示，银行监控每个支行的资产。

2）银行客户通过其 customer_id 值来标识，银行存储每个客户的姓名及其居住地的街道和城市。客户可以有有账号，并且可以贷款。客户可能同某个银行员工发生关系，该员工作为此客户的贷款负责人或私人助理。

3）银行员工通过其 employee_id 值来标识，银行的管理机构存储每个员工的姓名、电话、亲属等信息。

4）客户可以在银行中有一个或多个存款账号，每个银行账号有唯一的账户号码和余额信息。

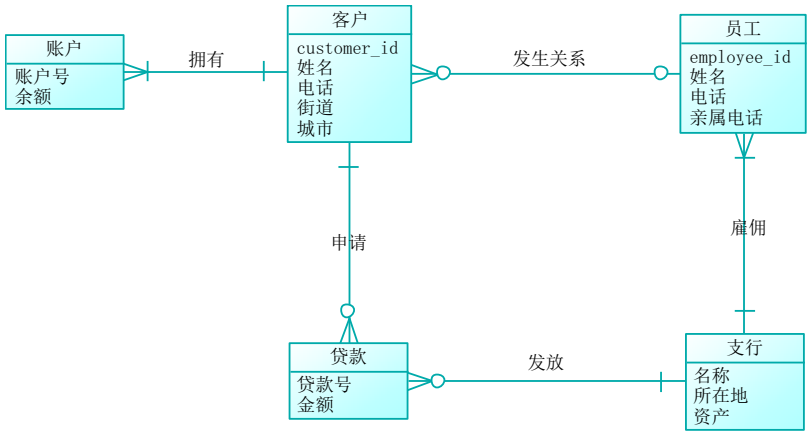
5）每笔贷款有支行发放，能被一个或多个客户共有，一笔贷款用一个唯一的贷款号标识。

根据以上的需求信息：

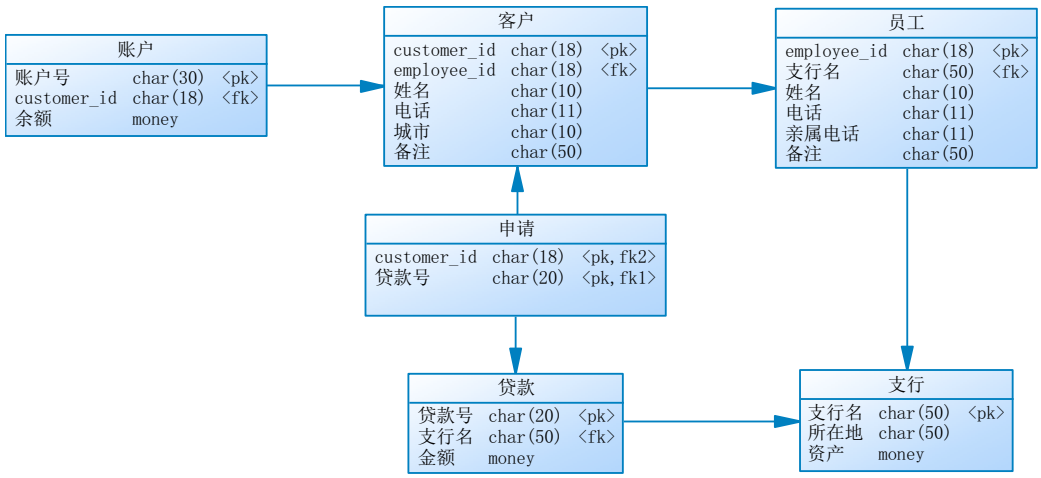
1. 给出银行数据库的实体集和每个实体的属性
2. 给出银行数据库的联系集，ER 图。
3. 给出银行数据库的表格， 包括属性及其主键和外码键

（提示：实体集、属性集、关系集、ER 图、关系模式、关系表、创建表的语句）

E-R 图：



PDM：



数据表定义：

表 1 账户表

字段名称	字段编码	数据类型	字段大小	必填字段	备注
账户号	账户号	Char	30	是	主键
customer_id	customer_id	Char	18	是	外键
余额	余额	Money	/	是	

表 2 客户表

字段名称	字段编码	数据类型	字段大小	必填字段	备注
customer_id	customer_id	Char	18	是	主键
employee_id	customer_id	Char	50	是	外键
姓名	余额	Money	10	是	
电话	电话	Char	11	是	
城市	城市	Char	11	是	
备注	备注	Char	50	否	

表 3 员工表

字段名称	字段编码	数据类型	字段大小	必填字段	备注
customer_id	customer_id	Char	18	是	主键
贷款号	贷款号	Char	20	是	外键

表 4 申请表

字段名称	字段编码	数据类型	字段大小	必填字段	备注
账户号	账户号	Char	30	是	主键，外键
customer_id	customer_id	Char	18	是	外键，外键
余额	余额	Money	/	是	

表 5 贷款表

字段名称	字段编码	数据类型	字段大小	必填字段	备注
贷款名	贷款名	Char	20	是	主键
支行号	支行号	Char	50	是	外键
金额	金额	Money	/	是	

表 6 支行表

字段名称	字段编码	数据类型	字段大小	必填字段	备注
支行名	贷款名	Char	50	是	主键
所在地	支行号	Char	50	是	
资产	金额	Money	/	是	

SQL 脚本：

```
/*==============================================================*/
/* DBMS name:      PostgreSQL 9.x                               */
/* Created on:     2020/4/21 19:43:52                           */
/*==================================== ====================*/

/*==============================================================*/
/* Table: 员工                                                  */
/*==============================================================*/
create table 员工 (
   employee_id         CHAR(18)           not null,
   支行名                CHAR(50)           null,
   姓名                  CHAR(10)           null,
   电话                  CHAR(11)           null,
   亲属电话              CHAR(11)           null,
   备注                  CHAR(50)           null,
   constraint PK_员工 primary key (employee_id)
);

/*==============================================================*/
/* Table: 客户                                                  */
/*==============================================================*/
create table 客户 (
   customer_id         CHAR(18)           not null,
   employee_id         CHAR(18)           null,
   姓名                  CHAR(10)           null,
   电话                  CHAR(11)           null,
   城市                  CHAR(10)           null,
   备注                  CHAR(50)           null,
   constraint PK_客户 primary key (customer_id)
);

/*==============================================================*/
/* Table: 支行                                                  */
/*==============================================================*/
create table 支行 (
   支行名                 CHAR(50)           not null,
   所在地                 CHAR(50)           null,
   资产                   MONEY              null,
   constraint PK_支行 primary key (支行名)
);

/*==============================================================*/
/* Table: 申请                                                  */
/*==============================================================*/
create table 申请 (
   customer_id         CHAR(18)           not null,
   贷款号                CHAR(20)           not null,
   constraint PK_申请 primary key (customer_id, 贷款号)
);
```

```
/*==============================================================*/
/* Table: 账户                                                  */
/*==============================================================*/
create table 账户 (
   账户号                CHAR(30)           not null,
   customer_id          CHAR(18)           null,
   余额                  MONEY              null,
   constraint PK_账户 primary key (账户号)
);

/*==============================================================*/
/* Table: 贷款                                                  */
/*==============================================================*/
create table 贷款 (
   贷款号                CHAR(20)           not null,
   支行名                CHAR(50)           null,
   金额                  MONEY              null,
   constraint PK_贷款 primary key (贷款号)
);

alter table 员工
   add constraint FK_员工_REFERENCE_支行 foreign key (支行名)
      references 支行 (支行名)
      on delete restrict on update restrict;

alter table 客户
   add constraint FK_客户_REFERENCE_员工 foreign key (employee_id)
      references 员工 (employee_id)
      on delete restrict on update restrict;

alter table 申请
   add constraint FK_申请_REFERENCE_贷款 foreign key (贷款号)
      references 贷款 (贷款号)
      on delete restrict on update restrict;

alter table 申请
   add constraint FK_申请_REFERENCE_客户 foreign key (customer_id)
      references 客户 (customer_id)
      on delete restrict on update restrict;

alter table 账户
   add constraint FK_账户_REFERENCE_客户 foreign key (customer_id)
      references 客户 (customer_id)
      on delete restrict on update restrict;

alter table 贷款
   add constraint FK_贷款_REFERENCE_支行 foreign key (支行名)
      references 支行 (支行名)
      on delete restrict on update restrict;
```