|  |  |  |
| --- | --- | --- |
| 2020春《数据库系统》实验报告 | | |
| 实验1：数据库设计与应用开发 | | |
| 姓名：陶飞宇  姓名：王鹏 | 学号：1170300720  学号：1170200227 | 班级：1737301  班级：1737301 |

1. 需求分析

描述该应用中数据的结构和约束。

**设计：**

概述：经过网络上的调查，发现了一个以“物业管理”为基础的数据库设计模型，将其提炼后将其实现。

**需求分析：**

对于物业管理，“业主”为主要的管理对象，下面以业主为中心进行分析：

①每一个业主都有一个独特的编号，同时应记录业主的姓名、性别和年龄，还有业主所在的楼宇号，房间号。

②每一个业主都可能拥有家庭成员，应为每一个家庭成员赋予一个独特的编号，同样的记录其姓名性别和年龄，同时标记其所属业主的ID。

③每一栋楼宇也应该拥有独立的编号，并记录楼宇的名称，竣工年份和建筑面积。

④每一个房间也应拥有独立的编号，同时也记录其房间名，房型，建筑面积和住户的入住时间，并记录住户的业主的ID和房间所属楼宇的编号。

⑤每一个业主都应缴纳物业费，为每一次缴费设置独特的编号，物业费有不同的类型，应记录缴费人的ID（业主ID），缴费时间和实际付款金额。

⑥每一种收费类型都应有自己的独特的编号，同时应记录该种收费的名称和应付的金额。

联系的需求分析：

①一个业主可能有0个或多个家庭成员

②一个栋楼宇可以没有业主（无人居住），或者有多个业主（多户）

③一个房间可以没有业主（无人居住），或者有一个业主（假设一个业主只拥有一个房间，两者一一对应）

④一个房间必须隶属于一栋楼宇，一栋楼宇必须有多个房间

⑤一个业主必须至少缴纳一项或多项物业费，一项物业费记录必须有他的唯一的缴纳人

⑥一种收费类型可能有0条或多条缴费记录，每一项缴费记录必须有唯一的缴费类型

**约束条件：**

主键约束：

所有的主键的属性都不可能为空。在建表的时候直接设置成NOT NULL即可。

外键约束：

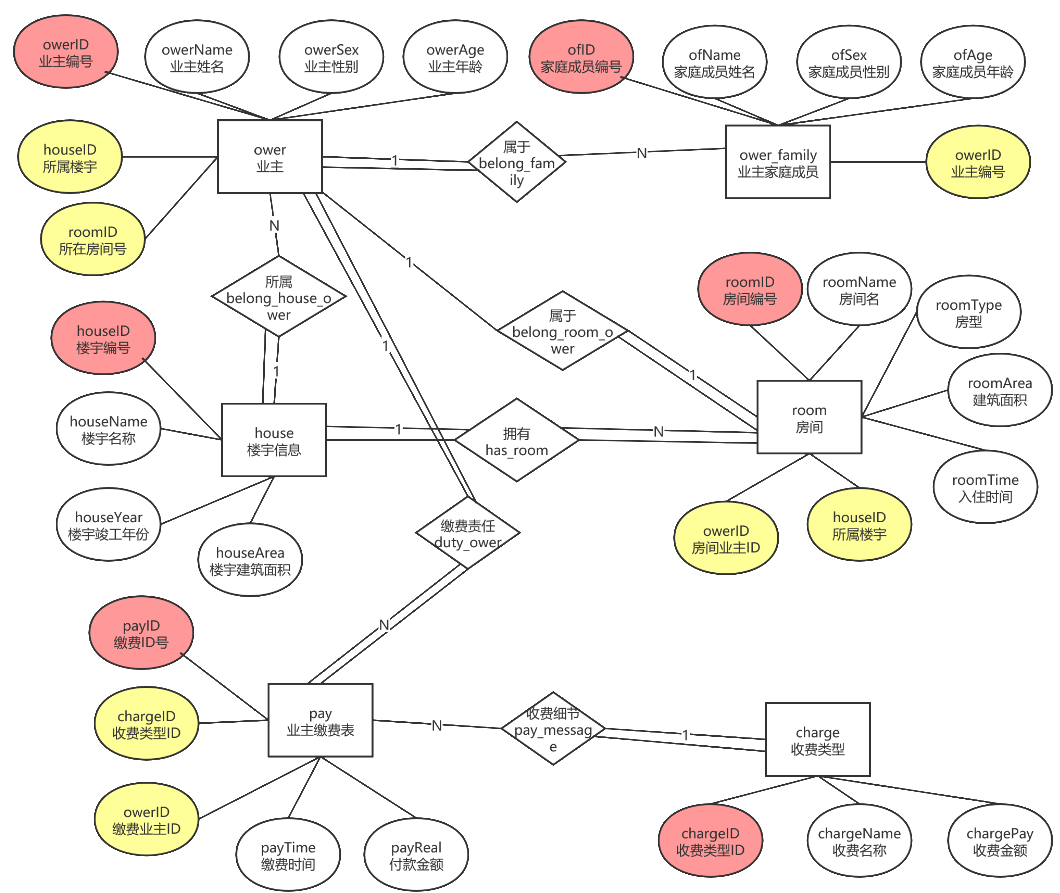
当主键表发生改变，外键所在的表随之做出某些改变。这个MYSQL自己可以做到，不需要用户考虑。

用户自定义约束：

所有主键都是独特的，不依赖其他关系，只根据本关系的主键就能独立进行查询，这要求我们为每一种主键制定合适的内容，比如家庭成员ID中应隐含业主ID，房间ID应隐含楼宇ID。

2. 概念数据库设计

根据上述的需求分析，作出ER关系模式图（红色为主键，黄色为外键）：



3. 逻辑数据库设计

将ER模型转换为关系数据库模式（主键加粗，外键加斜加下划线）：

ower(**owerID**,owerName,owerSex,owerAge,*houseID,roomID*)

ower\_family(**ofID**,ofName,ofSex,ofAge,*owerID*)

house(**houseID**,houseName,houseYear,huoseArea)

room(**roomID**,roomName,roomType,roomArea,roomTime,*houseID,owerID*)

pay(**payID**,payTime,payReal,*chargeID,owerID*)

charge(**chargeID**,chargeName,chargePay)

值得注意的是，在设计时，每一个外键都对应了一种关系，故而在进行转换的时候，对于联系性的转换，不再需要为关系添加额外的属性。

运用关系数据库规范化理论，对数据库模式进行规范化。

逐步进行规范化：

①1NF

显而易见的，所有关系里不存在多值属性和符合属性，故而所有关系都是1NF

②2NF

**ower：**业主姓名并不能直接决定任何其他属性（因为重名现象），而roomID和owerID一一对应，故而候选键为{owerID,roomID}，ower是2NF的。

**ower\_family:**同ower，家庭成员姓名也不能决定，故而候选键为{ofID}，ower\_family是2NF的

**house**：每栋楼宇既有唯一的ID也应有唯一的名字，故而候选键为{houseID,houseName}，两者都能决定其他所有属性，故而将house分为：

house\_ID(**houseID**,houseYear,huoseArea)

house\_Name(**houseName**,houseYear,huoseArea)

**room**：room情况并不完全相同，roomName可以决定roomType、roomArea，（房间名称决定房间户型和建筑面积）但无法决定其他的非主属性，而roomID和owerID一一对应，是2NF的。

pay：一条缴费记录仅由payID来决定，chargeID和owerID不决定其他任何属性，故而pay是2NF的。

chargeID：chargeID和chargeName都决定了其他属性，将其分解：

charge\_ID(**chargeID**,chargePay)

charge\_name(**chargeName**,chargePay)

更新后的关系为：

ower(**owerID**,owerName,owerSex,owerAge,*houseID,roomID*)

ower\_family(**ofID**,ofName,ofSex,ofAge,*owerID*)

house\_ID(**houseID**,houseYear,huoseArea)

house\_Name(**houseName**,houseYear,huoseArea)

room(**roomID**,roomName,roomType,roomArea,roomTime,*houseID,owerID*)

pay(**payID**,payTime,payReal,*chargeID,owerID*)

charge\_ID(**chargeID**,chargePay)

charge\_name(**chargeName**,chargePay)

③3NF

ower已经在2NF叙述，只有owerID决定其他所有关系，其他任何关系不决定自身之外的关系，故而ower是3NF的

ower\_family,同ower，只有ofID决定着其他关系，故而是3NF的

house\_ID，同ower，只有houseID决定着其他关系，是3NF的

house\_name，同上，3NF的（两者来源一致）

room，因为roomID决定所有属性，而roomName可以决定roomType、roomArea，存在了明显的传递依赖，将其分解：

room\_ID(**roomID**,roomName,roomTime,*houseID,owerID*)

room\_name(roomName,roomType,roomArea)

pay：只有payID决定着其他关系，故而是3NF的

charge\_ID和charge\_name同house分析，是3NF的

修改后的关系组：

ower(**owerID**,owerName,owerSex,owerAge,*houseID,roomID*)

ower\_family(**ofID**,ofName,ofSex,ofAge,*owerID*)

house\_ID(**houseID**,houseYear,huoseArea)

house\_Name(**houseName**,houseYear,huoseArea)

room\_ID(**roomID**,roomName,roomTime,*houseID,owerID*)

room\_name(roomName,roomType,roomArea)

pay(**payID**,payTime,payReal,*chargeID,owerID*)

charge\_ID(**chargeID**,chargePay)

charge\_name(**chargeName**,chargePay)

④BCNF

根据3NF中的阐述，可知所有在函数依赖左侧的项都为候选键当中的项。故而所有的项都是BCNF。不进行更改。

⑤4NF

存在一些多值依赖（左侧非候选键）如：

owerID→→ofID、owerID→→payID

但是将其消除后影响数据库完整性，保留第④步的结果

4. 物理数据库设计

根据数据库工作负载，运用反规范化和分表等方法，调整数据库模式。

在规范化的过程中可以发现，对room，house，chargeID的分解，都是因为一些重复的，都具有全部决定能力的属性，导致一张表被分为了即为相似的两张表，这样的做法是浪费空间的。故而删除不是主键的重复属性，减少空间开销。其结果为：

ower(**owerID**,owerName,owerSex,owerAge,*houseID,roomID*)

ower\_family(**ofID**,ofName,ofSex,ofAge,*owerID*)

house(**houseID**,houseYear,houseArea)

room(**roomID**, roomType,roomArea,roomTime,*houseID,owerID*)

pay(**payID**,payTime,payReal,*chargeID,owerID*)

charge(**chargeID**,chargePay)

根据数据库工作负载，设计数据库索引。

索引会降低增删改的速度，并占用磁盘空间，故而我们只在常用的几个地方使用它，以下是几种可能的常用的索引：  
 ①owerID，业主是物业管理的核心，其唯一标识owerID必然是经常查询的部分。

②ofID+owerID，用于查找小区中任一成员，以户主+家属的方式

③roomArea+roomType，同种户型+同种面积的房间为统一构造，可以统一管理相同构造的房间。

④chargeID+owerID 用来查询用户一种费用的缴费情况

5. 数据库建立

声明概念模式的同时建立索引：

CREATE TABLE ower

(owerID INT ,

owerName CHAR(12),

owerSex CHAR(1),

owerAge INT,

houseID INT,

roomID INT,

PRIMARY KEY(owerID ASC),

FOREIGN KEY(houseID) REFERENCES house(houseID),

FOREIGN KEY(roomID) REFERENCES room(roomID));

CREATE TABLE ower\_family

(ofID INT,

ofName CHAR(12),

ofSex CHAR(1),

ofAge INT,

owerID INT,

PRIMARY KEY(ofID ASC),

FOREIGN KEY(owerID) REFERENCES ower(owerID),

KEY (ofID,owerID) key\_ower\_family);

CREATE TABLE house

(houseID INT,

houseYear INT,

houseArea FLOAT,

PRIMARY KEY(houseID ASC));

CREATE TABLE room

(roomID INT,

roomType CHAR(24),

roomArea FLOAT,

roomTime DATE,

houseID INT,

owerID INT,

PRIMARY KEY(roomID ASC),

FOREIGN KEY(houseID) REFERENCES house(houseID),

KEY (roomArea,roomType) key\_room\_str

);

CREATE TABLE pay

(payID INT,

payTime DATE,

payReal FLOAT,

chargeID INT,

owerID INT,

PRIMARY KEY(payID ASC),

FOREIGN KEY(chargeID) REFERENCES charge(chargeID),

FOREIGN KEY(owerID) REFERENCES ower(owerID)

KEY (chargeID,owerID) key\_charge

);

CREATE TABLE charge

(chargeID INT,

chargePay INT,

PRIMARY KEY(chargeID ASC)

);

根据应用需求，设计数据库的外模式，并使用SQL定义数据库视图。

①业主的家庭成员视图 (ID为要查询的业主)

CREATE VIEW family AS

SELECT ofName,ofSex,ofAge,owerID

FROM ower\_family

WHERE owerID = ‘ID’

②业主的住宅明细(ID为要查询的业主)

CREATE VIEW building AS

SELECT owerName,houseID,roomID,roomType,roomArea,roomTime

FROM ower JOIN room

WHERE owerName = ‘Name’

③一个业主的缴费明细视图（ID为要查询的业主）

CREATE VIEW payment AS

SELECT payID,chargeID,payTime,chargePay,payReal,owerID

FROM pay JOIN charge

WHERE owerID = ‘ID’

6. 总结与体会

总结你的数据库设计方案，突出设计方案的亮点。

对现实的物业管理和缴费进行了关系数据的构造，通过了层层规范化将规范等级提高到了BCNF级别，并进行反规范化去除冗余的表，并根据一些常用的查询建立了索引。

本方案的亮点在于每个实体之间都有（至少间接有）关系，整体性强。

说明你在数据库设计过程中的心得体会。

掌握了设计数据库的一些细节，明确了ER关系图的绘制方式，特别地亲身体会到各级范式的标准以及其不断规划的过程，对数据库建立的建立能力有较大的提升。