居民小区数据库系统

Residential community database system

HAN DONGHUN

目录

1. 需求分析

1.1 系统流程图

居民小区数据库系统的用户包括物业管理部门和居民。

物业管理部门是居民小区物业服务的提供者，是居民小区数据库系统的管理员，需要掌握小区楼房、居民和车位的信息，提供的居民服务包括缴费服务管理、居民意见管理等，公共服务包括公共设备维护管理、日常清洁管理等。

居民是居民小区物业服务的享受者，需要把个人信息提供给物业管理部门，享受物业管理部门提供的服务。

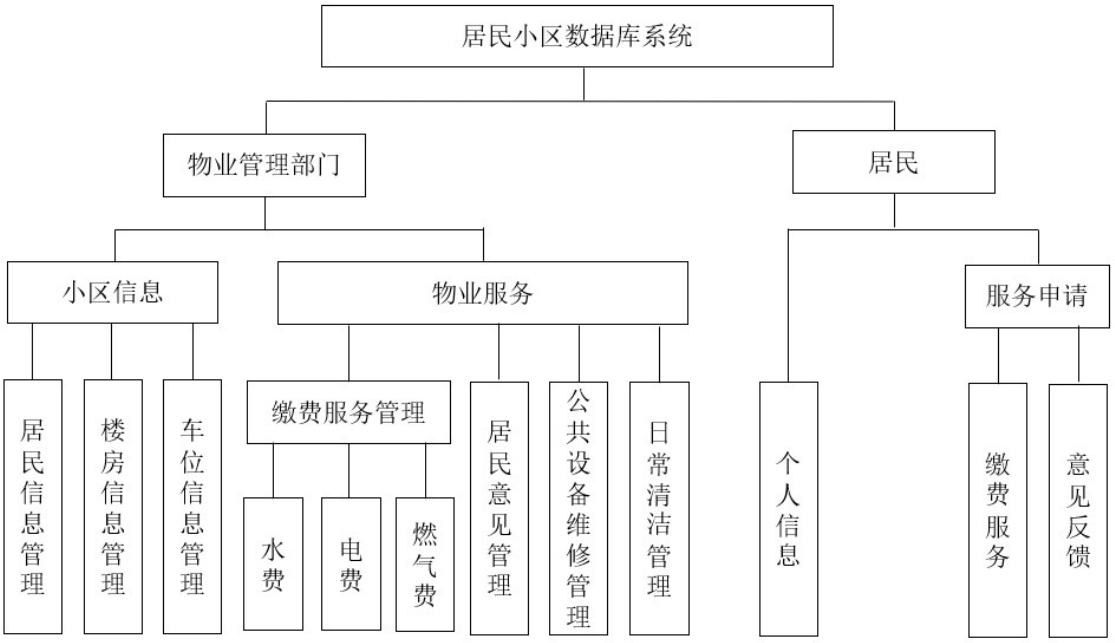
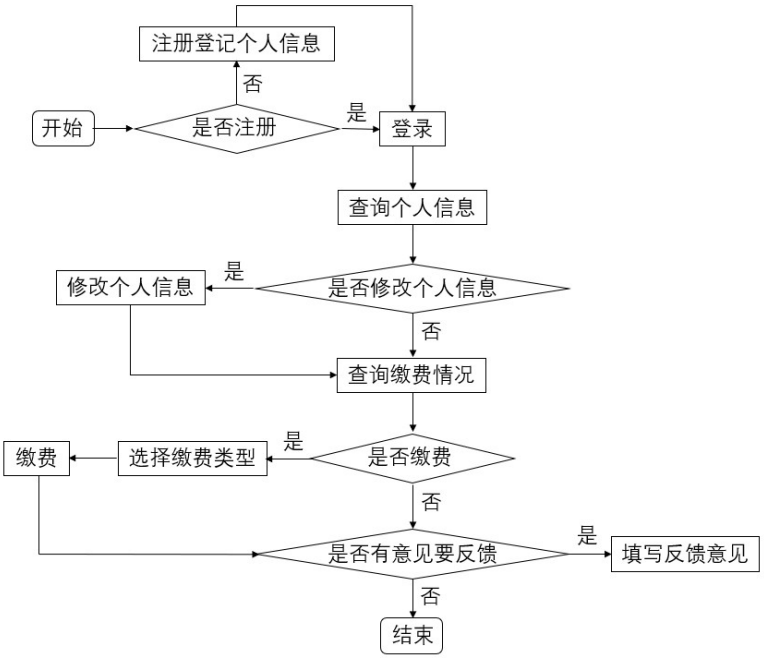


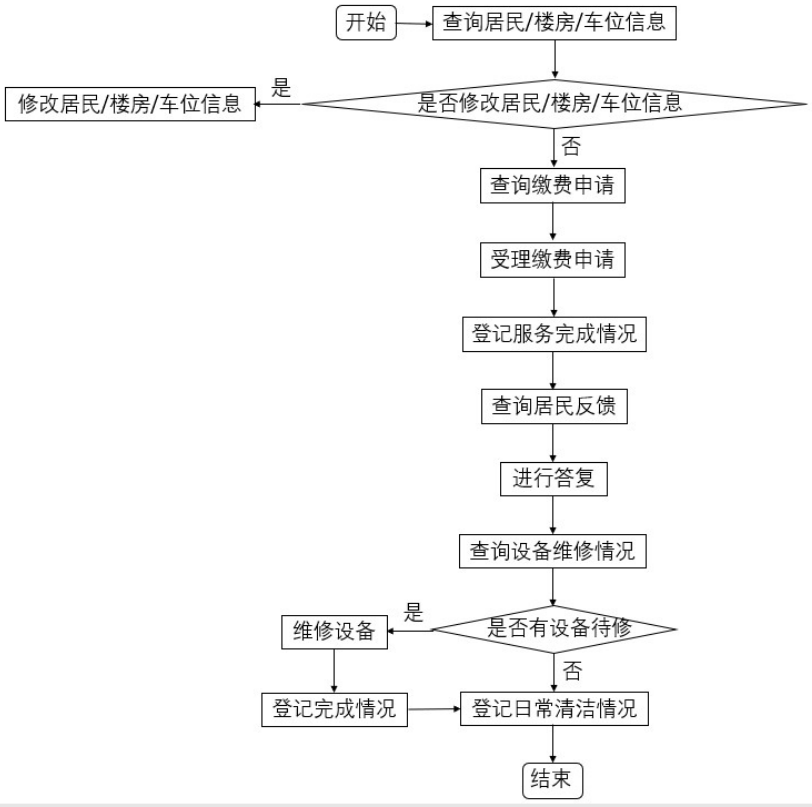
图 1.1 系统流程图

1.2 业务流程图

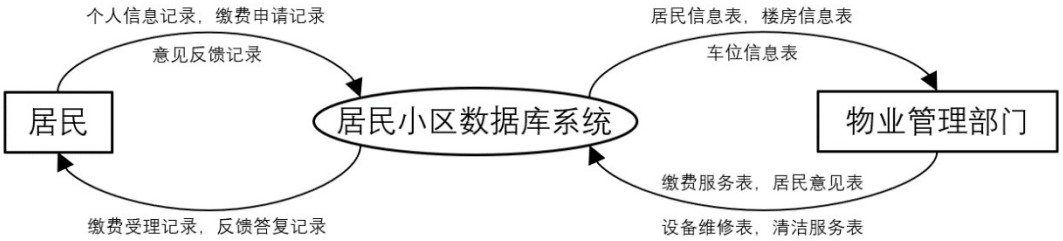
1.2.1 居民业务流程图



1.2.2 物业管理部门业务流程图



1.3 数据流程图



1.4 数据字典

1.居民信息

字段	字段类型	说明
person_id	Varchar(20)	居民编号
name	Nvarchar(20)	姓名
id_card	Varchar(20)	身份证号
sex	Nvarchar(5)	性别
tele	Varchar(20)	联系电话
password	Nvarchar(50)	登录密码
room_id	Varchar(20)	房间号
parking_id	Varchar(20)	车位号
remark	Nvarchar(255)	备注

2.楼房信息

字段	字段类型	说明
building_id	Varchar(20)	楼房号
floor	Varchar(5)	层数
remark	Nvarchar(255)	备注

3.房间信息

字段	字段类型	说明
room_id	Varchar(20)	房间号

building_id	Varchar(20)	楼房号
floor_id	Varchar(5)	层号
remark	Nvarchar(255)	备注

4.车位信息

字段	字段类型	说明
parking_id	Varchar(20)	车位号
car_id	Varchar(20)	车牌号
car_type	Varchar(20)	车型
car_host	Varchar(20)	车主
square	Varchar(20)	面积
remark	Nvarchar(255)	备注

5.缴费服务

字段	字段类型	说明
item_id	Varchar(20)	款项编号
room_id	Varchar(20)	房间号/引用房间信息表的 room_id
item	Nvarchar(10	服务事项
person_id	Varchar(20)	付款人/引用居民信息表的 person_id
date	Date	收款时间
money	Float	金额
condition	Nvarchar(20)	服务完成情况

6.居民意见

字段	字段类型	说明
advice_id	Varchar(20)	意见编号
person_id	Varchar(20)	反馈人/引用居民信息表的 person_id
ad_time	Datetime	反馈时间
content	Nvarchar(255)	投诉内容

answer	Nvarchar(255)	答复
an_time	Datetime	答复时间

7.设备维修

字段	字段类型	说明
item_id	Varchar(20)	事项编号
equ_name	Nvarchar(20)	设备名
area	Nvarchar(255)	所在区域
staff_id	Nvarchar(20)	负责人员
condition	Nvarchar(20)	完成情况
finish_time	Datetime	完成时间

8.清洁服务

字段	字段类型	说明
item_id	Varchar(20)	事项编号/主键
area	Nvarchar(255)	所在区域
staff_id	Nvarchar(20)	负责人员
condition	Nvarchar(20)	完成情况
finish_time	Datetime	完成时间

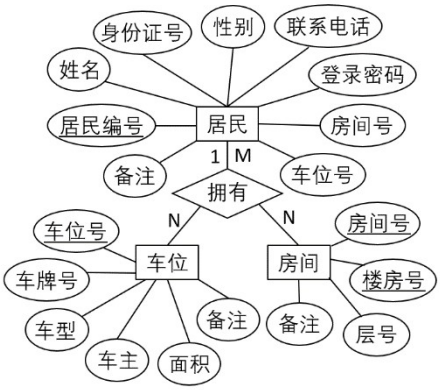
9.工作人员信息

字段	字段类型	说明
staff_id	Varchar(20)	居民编号/主键
name	Nvarchar(20)	姓名
id_card	Varchar(20)	身份证号
sex	Nvarchar(5)	性别
tele	Varchar(20)	联系电话
remark	Nvarchar(255)	备注

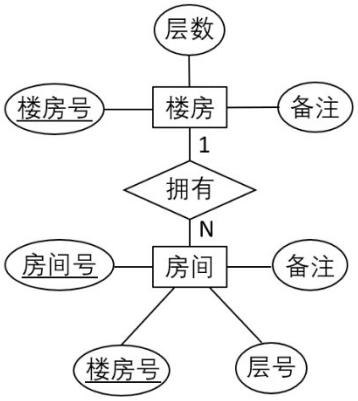
2. 概念结构设计

2.1 局部 E-R 图

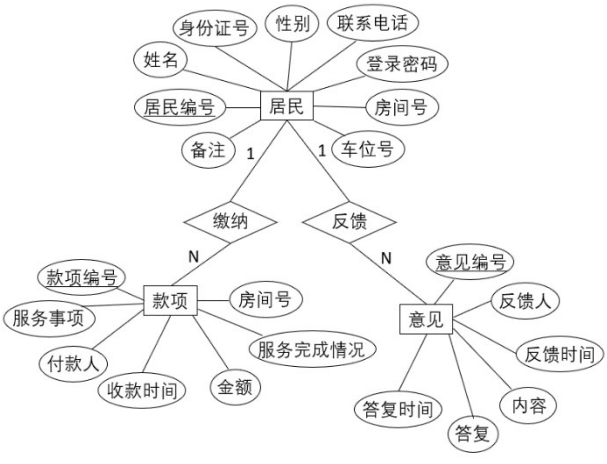
2.1.1 居民信息



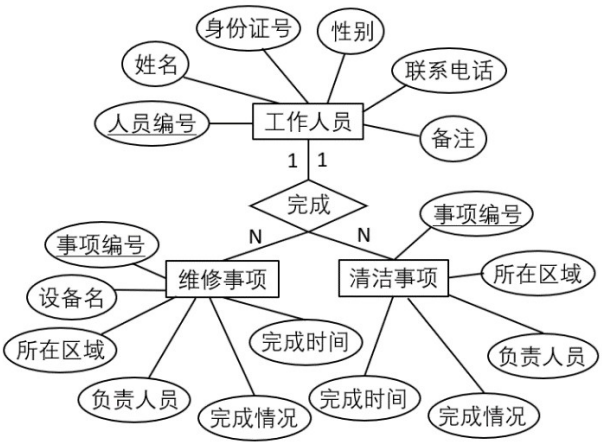
2.1.2 楼房信息



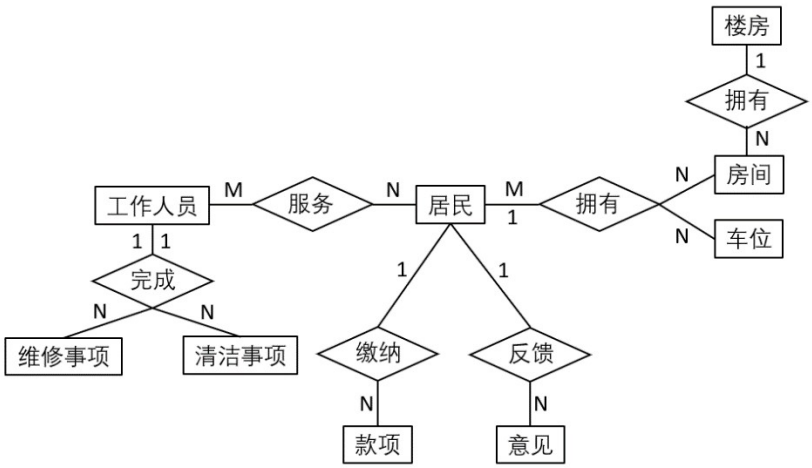
2.1.3 居民服务



2.1.4 公共服务



2.2 全局 E-R 图



3. 逻辑结构设计

3.1 将 E-R 图转换为关系模式转换后的关系模式：

居民：Person（person_id，name，id_card，sex，tele，password，room_id，parking_id， remark）楼房：Building（building_id，floor，remark）房间：Room（room_id，building_id，floor_id，remark）车位：Parking（parking_id，car_id，car_type，car_host，square，remark）缴费服务：Charge（item_id，room_id，item，person_id，date，money，condition）居民意见：Advice（advice_id，person_id，ad_time，content，answer，an_time） 设备维修：Fix（item_id，equ_name，area，staff_id，condition，finish_time）日常清洁：Clean（item_id，area，staff_id，condition，finish_time）工作人员：Staff（staff_id，name，id_card，sex，tele，remark）

（注：标有直线下划线的为主属性，标有波浪线下划线的是外键属性）

3.2 规范化

3.2.1 第二范式

关系模式“房间”中，（room_id，building_id）是候选码，但是room_id→floor_id，出现了对候选码的部分函数依赖，所以将其分解为第二范式：房间：Room（room_id，building_id，remark）楼层：Floor（room_id，floor_id）

3.2.2 第三范式

（1） 关系模式“居民”中，person_id→id_card，而其他非主属性都依赖于id_card，形成传递函数依赖，故分解为第三范式：

居民：Person（person_id，name，sex，tele，password，room_id，building_id, parking_id， remark）

居民身份证：Person_ID_card（person_id，id_card）

（2） 关系模式“车位”中，parking_id→car_id，而其他非主属性都依赖于 car_id，形成传递函数依赖，故分解为第三范式：

车位：Parking（parking_id，car_id）车辆：Car（car_id，car_type，car_host，square，remark）

(3)

关系模式“工作人员”中，person_id→id_card，而其他非主属性都依赖于 id_card，形成传递函数依赖，故分解为第三范式：

工作人员：Staff（person_id，name，sex，tele，remark）工作人员身份证：Staff_ID_card（person_id，id_card）

3.2.3 BC 范式

至此，以上关系模式全部符合 BCNF.

3.2.4 第四范式

至此，以上关系模式全部符合第四范式。

于是，规范化之后的关系模式如下：

居民：Person（person_id，name，sex，tele，password，room_id, parking_id，remark）居民身份证：Person_ID_card（person_id，id_card）楼房：Building（building_id，floor，remark）房间：Room（room_id，building_id，remark）楼层：Floor（room_id，floor_id）车位：Parking（parking_id，car_id）车辆：Car（car_id，car_type，car_host，square，remark）缴费服务：Charge（item_id，room_id，item，person_id，date，money，condition）居民意见：Advice（advice_id，person_id，ad_time，content，answer，an_time） 设备维修：Fix（item_id，equ_name，area，staff，condition，finish_time）日常清洁：Clean（item_id，area，person_id，condition，finish_time）工作人员：Staff（person_id，

name，sex，tele，remark）工作人员身份证：Staff_ID_card（person_id，id_card）

4. 数据库实施

4.1 创建数据库

```
CREATE DATABASE Neighborhood; -- 社区居民管理系统
```

4.2 创建关系

①居民

```
CREATE TABLE Person
(
person_id Varchar(20) PRIMARY KEY,
name Nvarchar(20) NOT NULL, sex
Nvarchar(5) NOT NULL, tele
Varchar(20) NOT NULL, password
Nvarchar(50) NOT NULL,
room_id Varchar(20) NOT NULL FOREIGN KEY REFERENCES Room(room_id),
parking_id Varchar(20) FOREIGN KEY REFERENCES Parking(parking_id), remark
Nvarchar(255)
);
```

②居民身份证

```
CREATE TABLE Person_ID_card
(
person_id Varchar(20) PRIMARY KEY FOREIGN KEY REFERENCES
Person(person_id),          id_card
Varchar(20) NOT NULL
);
```

③楼房

```
CREATE TABLE Building
(
building_id Varchar(20) PRIMARY KEY,
floor Varchar(5) NOT NULL, remark
Nvarchar(255)
);
```

④房间

```
CREATE TABLE Room
(
room_id Varchar(20) PRIMARY KEY,
building_id Varchar(20) NOT NULL FOREIGN KEY REFERENCES
Building(building_id), remark
Nvarchar(255)
);
```

⑤楼层

```
CREATE TABLE Floor
(
room_id Varchar(20) PRIMARY KEY FOREIGN KEY
REFERENCES Room(room_id), floor_id Varchar(5) NOT NULL
);
```

⑥车位

```
CREATE TABLE Parking
(
parking_id Varchar(20) PRIMARY KEY,
car_id Varchar(20) FOREIGN KEY REFERENCES Car(car_id)
);
```

⑦车辆

```
CREATE TABLE Car
(
car_id  Varchar(20)  PRIMARY  KEY,
car_type   Varchar(20)   NOT   NULL,
car_host Varchar(20) NOT NULL,
);
```

⑧缴费服务

```
CREATE TABLE Charge
(
item_id Varchar(20) PRIMARY KEY,
room_id Varchar(20) FOREIGN KEY REFERENCES Room(room_id),
item Nvarchar(10) NOT NULL,
person_id Varchar(20) FOREIGN KEY REFERENCES Person(person_id),
date Date NOT NULL, money Float NOT NULL, condition Nvarchar(20)
NOT NULL
);
```

⑨居民意见

```
CREATE TABLE Advice
(
advice_id Varchar(20) PRIMARY KEY,
person_id Varchar(20) FOREIGN KEY REFERENCES Person(person_id),
ad_time Datetime, content Nvarchar(255), answer Nvarchar(255), an_time
Datetime
);
```

⑩设备维修

```
CREATE TABLE Fix
```

```
(
item_id Varchar(20) PRIMARY KEY,
equ_name Nvarchar(20) NOT NULL,
area Nvarchar(255) NOT NULL,
staff Varchar(20) FOREIGN KEY REFERENCES Staff(staff_id),
condition Nvarchar(20), finish_time Datetime
);
```

⑪日常清洁

```
CREATE TABLE Clean
(
item_id Varchar(20) PRIMARY KEY, area
Nvarchar(255) NOT NULL,
staff Varchar(20) FOREIGN KEY REFERENCES Staff(staff_id),
condition Nvarchar(20), finish_time Datetime
);
```

⑫工作人员

```
CREATE TABLE Staff
(
staff_id Varchar(20) PRIMARY KEY,
name Nvarchar(20) NOT NULL, sex
Nvarchar(5) NOT NULL, tele
Varchar(20) NOT NULL, remark
Nvarchar(255)
);
```

⑬工作人员身份证

```
CREATE TABLE Staff_ID_card
(
```

```
staff_id Varchar(20) PRIMARY KEY FOREIGN KEY REFERENCES
Staff(Staff_id), id_card
Varchar(20)
);
```

4.3 添加约束实体完整性约束和参照完整性约束已经在定义时候添加，这里补充域完整性约束。

（1） 居民和工作人员性别限制显然，居民和工作人员的性别只能是男/女，所以添加 CHECK 约束：

```
ALTER TABLE Person
ADD CONSTRAINT 居民性别限制
CHECK(sex IN('男','女'));
ALTER TABLE Staff
ADD CONSTRAINT 工作人员性别限制
CHECK(sex IN('男','女'));
```

（2） 完成状态限制完成状态只能是完成/未完成，所以添加 CHECK 约束：

```
ALTER TABLE Charge
ADD CONSTRAINT 缴费服务完成状态限制
CHECK(condition IN('完成','未完成'));

ALTER TABLE Fix
ADD CONSTRAINT 设备维修完成状态限制
CHECK(condition IN('完成','未完成'));

ALTER TABLE Clean
```

```
ADD CONSTRAINT 清洁服务完成状态限制
CHECK(condition IN('完成','未完成'));
```

（3） 默认完成状态

默认完成状态为未完成，添加 DEFAULT 约束：

```
ALTER TABLE Charge
ADD CONSTRAINT 缴费服务默认完成状态
DEFAULT '未完成' for condition;
ALTER TABLE Fix
ADD CONSTRAINT 维修服务默认完成状态
DEFAULT '未完成' for condition;
ALTER TABLE Clean
ADD CONSTRAINT 清洁服务完成状态
DEFAULT '未完成' for condition;
```

4.4 建立触发器

当删除表 Building 中某一栋楼宇编号时，在 Room 中也删除此楼宇：

```
Create Trigger Del_Bd on Building
After Delete
As
    Delete From Room
    Where room_id
    In (Select room_id From Deleted)
```

5. 系统实现

注：由于操作的同质性，报告里仅举典例进行说明。

5.1 插入数据以居民和居民身份证为例，演示插入数据的过程，其他的略去：

```
Insert into Person
```

```
Values ('01102001','Mike','男','13555555555','bilibili','203','2001','新住户'),
('01102003','Lucy','女','13555555666','cilicili','205','2022','孕妇');
```

```
(2 行受影响)
```

尝试插入一组性别不符合域约束的数据：

```
Insert into Person
Values ('01102008','ChenJie','老人','13555544455','191919','303','2011','老师');
```

```
消息 547，级别 16，状态 0，第 1 行
INSERT 语句与 CHECK 约束"居民性别限制"冲突。该冲突发生于数据库"Neighborhood"，表"dbo.Person"，column 'sex'。
语句已终止。
```

```
Insert into Person_ID_card
Values ('01102001','330222200001010000'),      ('01102003','330222200001010022');
```

```
(2 行受影响)
```

5.2 查询信息

①条件查询示例查询

Mike 的车位号：

```
SELECT parking_id
FROM Person
WHERE name='Mike';
```

	parking_id
1	2001

②子查询示例

查询 advice_id 为“001”的居民的房间号：

```
SELECT room_id
FROM Person
WHERE person_id=(SELECT person_id FROM Advice WHERE advice_id='001');
```

结果 消息

	room_id
1	205

③多表连接查询示例

查询工作人员“Lily”的身份证号：

```
SELECT id_card
FROM Staff s JOIN Staff_ID_card si ON s.staff_id=si.staff_id
WHERE name='Lily';
```

结果 消息

	id_card
1	330222200205000000

5.3 更新信息把事项编号为 0101 的清洁事项的完成状

态改为“已完成”：

```
UPDATE Clean
SET condition='完成'
WHERE item_id='0101';
```

更新前：

	item id	area	staff	condition	finish time
▸	0101	北边	001	未完成	2002-02-0...

更新后：

	item id	area	staff	condition	finish time
▸	0101	北边	001	完成	2002-02-0...

5.4 创建视图创建一个视图给编号‘01102001’的居民使用，让他能够看到他的水

费缴纳情

况：

```
CREATE VIEW water(item_id,date,money,condition)
AS SELECT item_id,date,money,condition
  FROM Charge
  WHERE item='水费' AND person_id='01102001';
```

	item id	date	money	condition
▶	0101	2002-02-02	50	完成
*	NULL	NULL	NULL	NULL

5.5 授权管理

Mike 的个人水费记录视图，仅供 Mike 查询：

```
GRANT SELECT
ON water
TO Mike;
```

6. 课程设计总结

6.1 项目总结与补充说明

本次小组作业做的是居民小区数据库系统的题目，为期三周的数据库设计实践引导我们在实际情境中运用所学的理论和代码知识，推动我们在实践中温故知新、查漏补缺。我们对数据库设计的步骤以及框架有了基本的认识，从需求分析到概念结构设计、逻辑结构设计，再到数据库实施、系统实现，我们进一步理解如何根据用户的需求来设计一个合理、完整、高效的数据库。

由于知识和时间的限制，我们对本系统的复杂度做了一些限制，该系统主要考虑到了现代居民小区管理与服务的主要共性需求，并且在实现上做了一些简化处理：比如缴费服务这一块没有细化为水电燃气等更具体的模块；以及设备维修仅考虑公共设备的维护，不考虑居民个人报修的情况。这样的处理是为了在已有的认识水平和时间限制下确保最高的系统完成度和逻辑合理性。我们认为，有必要在项目总结中对系统复杂度限制的说明是必要的。