数据库系统原理学期实验项目：A 市便民防疫系统

软件学院 2011277 胡旖雯

一、基本业务需求

该系统为 A 市市民在疫情期间提供疫情相关的各项服务，为 A 市疫情防控部门提供市民健康情况的统计信息，包括以下服务：

市民端（客户端）：

1. 查询健康码情况（根据 A 市政策，新冠患者及封控区市民赋红码；位于 A 市高风险地区或途径 A 市排查范围地区未按防疫政策进行三天一检的人员赋黄码；其他情况是绿码）。
2. 预约核酸检测。
3. 查询核酸检测结果（阳性、24h 阴性、48h 阴性、72h 阴性、>72h 阴性）。
4. 预约疫苗接种。
5. 查询疫苗接种情况（未接种、已接种一针、已接种两针、已接种三针、疫苗接种的公司）。
6. 查询各地风险地区、查询本市管控范围。
7. 返市市民查询 14 天内是否到过本市管控范围地区。

防疫工作管理端：

1. 更新、管理数据；（服务器端）
2. 统计本市各区市民核酸检测情况；
3. 统计本市途径风险地区的人员信息；
4. 统计本市疫情风险区居民信息。

二、概念结构设计

1. 实体集及属性

（1）A 市市民：身份证号、姓名、手机号码；

（2）A 市各街道：区域（复合属性，包括区、街）、风险等级、管控等级；

（3）核酸检测业务：检测类型、委托机构、费用；

（4）疫苗接种业务：疫苗来源公司、疫苗种类（第几针）；

（5）A 市外全国各地：区域（复合属性，包括省、市、区、街）、

风险等级、管控等级、是否在 A 市排查范围内。

注：带下划实线属性表示主码，虚线表示弱实体集的分辨符，把健康码作为派生属性，不纳入实体集，是因为其计算规则较复杂，考虑用储存函数或 view 来实现计算；为简化模型，在市民实体中省略年龄、性别等与业务无直接关系的属性，实际情况中视情况斟酌加入性别、年龄等属性，对整个系统的结构不会造成太大影响。

2. 联系

说明：理想情况下假设 A 市市民为目前在 A 市且在常居住址的人，预约了核酸或疫苗的市民会在规定的时间内完成检测或接种。

（1）市民-街道：关联市民和 A 市各街道；

（2）预约核酸检测、查看检测报告：关联市民和核酸检测业务，包含描述性属性时间和结果；（联系集中一次检验会覆盖上一次的结果）

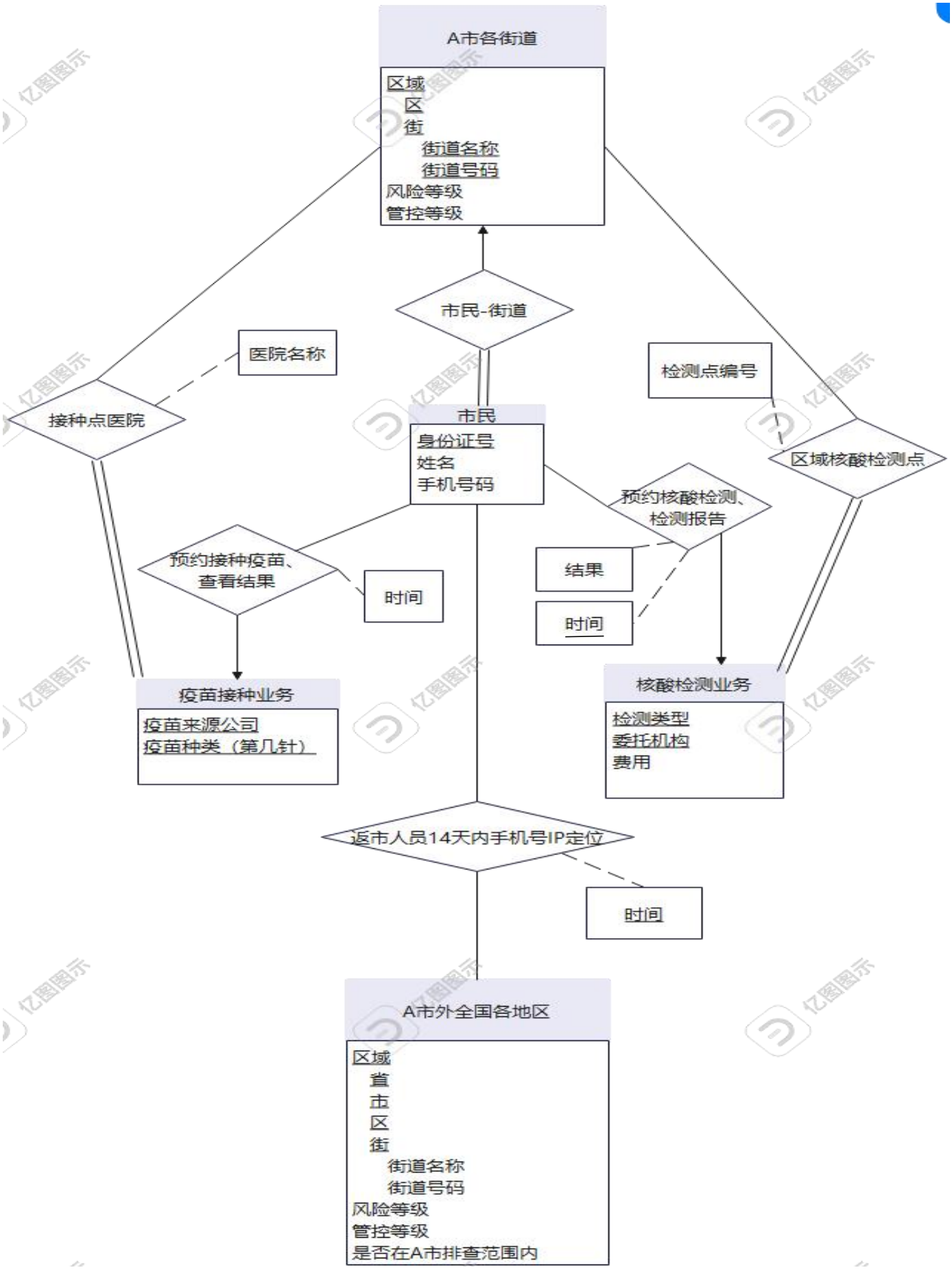
（3）区域核酸检测点：关联核酸检测业务和 A 市各街道，包含描述性属性检测点编号；

（4）预约接种疫苗、查看结果：关联市民和疫苗接种业务，包含描述性属性时间；

（5）接种点医院：关联疫苗接种业务和 A 市各街道，包含描述性属性医院名称；

（6）返市人员 14 天内手机号 IP 定位：关联市民与 A 市外全国各地区，包含描述性属性时间。

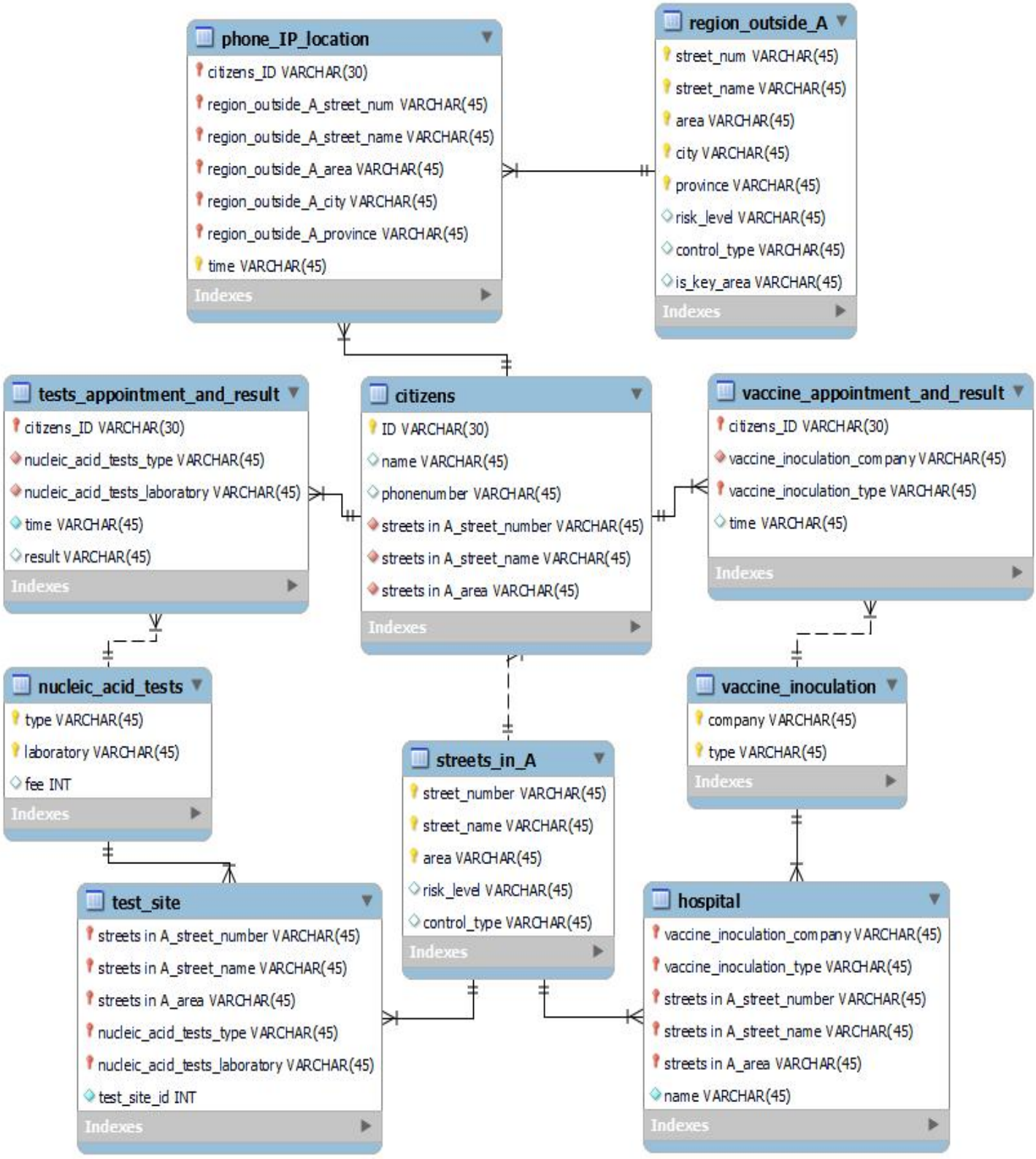
3. 实体-联系 E-R 图


A市各街道
区域
区
街
街道名称
街道号码
风险等级
管控等级
市民-街道
医院名称
接种点医院
检测点编号
区域核酸检测点
市民
身份证号
姓名
手机号码
预约核酸检测、
检测报告
预约接种疫苗、
查看结果
时间
结果
时间
疫苗接种业务
疫苗来源公司
疫苗种类（第几针）
核酸检测业务
检测类型
委托机构
费用
返市人员14天内手机号IP定位
时间
A市外全国各地区
区域
省
市
区
街
街道名称
街道号码
风险等级
管控等级
是否在A市排查范围内

三、逻辑结构设计

1. 关系数据库的 E-R 模型：

将 E-R 图实体集、联系集转换为数据库关系模式，并删除模式冗余、进行模式合并，简化得：



2. 根据范式理论分解优化后的设计

（1）分解优化

背景说明：根据 A 市政策，风险等级划分以区为单位，防控等级划分以街道为单位，外市中高风险区和管控、封控区划入 A 市排查范围，因此以下的关系模式存在冗余：

关系模式 streets_in_A 存在函数依赖：

area -> risk_level,

area,street_name->control_type;

上述两个函数依赖皆非平凡的函数依赖，且 area，(area,street_name)都不是超码，因此关系模式 street in A

(area,street_name,street_number,risk_level,control_type)

应该分解为：

(area,risk_level),(area,street_name,control_type),(area,street_name,street_number)

关系模式 region_outside_A 存在函数依赖：

risk_level,control_type->is_key_area;

注：其他各市政策与 A 市有区别，无法确定函数 risk_level,control_type 的除主键外的依赖

(street_number,street_name,area,city,province,risk_level,control_type,is_key_area)

应该分解为：

(risk_level,control_type,is_key_area),

(street_number,street_name,area,city,province,risk_level,control_type)

（2） 最终设计方案：

citizens(ID,name,phonenumber,street_number,street_name,area);

streets_in_A(street_number,street_name,area);

risk_level in area(area,risk_level);

control_type_in_street(street_name,area,control_type);

regions_outside_A(street_number,street_name,area,city,province,risk_level,control_type);

key_area_standard(risk_level,control_type,is_key_area);

phone_IP_location(ID,street_number,street_name,area,city,province,time);
vaccine_inoculation(company,type);
vaccine_appointment_and_result(ID,company,type,time);
hospital(street_number,street_name,area,name,company,type);
nucleic_acid_tests(type,laboratory,fee);
test_appointment_and_result(ID,type,laboratory,time,result);
test_sites(street_number,street_name,area,type,laboratory,test_site_id)
.

四、物理结构设计

1. SQL DDL 语句：

```
create database if not exists epidemic_system;
use epidemic_system;

drop table    if exists `vaccine_appointment_and_result`,`hospital`,
`vaccine_inoculation`,`test_appointment_and_result`,
`test_sites`,`risk_level_in_area`,`phone_ip_location`,
`control_type_in_a`,`regions_outside_a`,`key_area_standard`,
`citizens`,`streets_in_a`,`nucleic_acid_tests` ;

-- 生成数据库表：
CREATE TABLE `streets_in_a` (
   `street_number` VARCHAR(45) NOT NULL,
   `street_name` VARCHAR(45) NOT NULL,
   `area` VARCHAR(45) NOT NULL,
   PRIMARY KEY (`street_number`, `street_name`, `area`));

CREATE TABLE `citizens` (
   `ID` VARCHAR(45) NOT NULL,
   `name` VARCHAR(45) NULL,
```

```
    `phone_number` VARCHAR(45) NULL,
    `street_number` VARCHAR(45) NULL,
    `street_name` VARCHAR(45) NULL,
    `area` VARCHAR(45) NULL,
    PRIMARY KEY (`ID`),
    INDEX `citizens_fk_idx` (`street_number` ASC, `street_name` ASC,
`area` ASC) VISIBLE,
    CONSTRAINT `citizens_fk`
      FOREIGN KEY (`street_number` , `street_name` , `area`)
      REFERENCES `streets_in_a` (`street_number` , `street_name` ,
`area`)
      ON DELETE SET NULL
  ON UPDATE RESTRICT);

  CREATE TABLE `risk_level_in_area` (
    `area` VARCHAR(45) NOT NULL,
    `risk_level` VARCHAR(45) NULL,
    PRIMARY KEY (`area`));

  CREATE TABLE `control_type_in_a` (
    `street_name` VARCHAR(45) NOT NULL,
   `area` VARCHAR(45) NOT NULL,
    `control_type` VARCHAR(45) NULL,
    PRIMARY KEY (`street_name`, `area`));

  CREATE TABLE `regions_outside_a` (
    `street_number` VARCHAR(45) NOT NULL,
    `street_name` VARCHAR(45) NOT NULL,
    `area` VARCHAR(45) NOT NULL,
    `city` VARCHAR(45) NOT NULL,
    `province` VARCHAR(45) NOT NULL,
```

```
    `risk_level` VARCHAR(45) NULL,
    `control_type` VARCHAR(45) NULL,
    PRIMARY KEY (`street_number`, `street_name`, `area`, `city`,
`province`));


  CREATE TABLE `key_area_standard` (
    `risk_level` VARCHAR(45) NOT NULL,
    `control_type` VARCHAR(45) NOT NULL,
    `is_key_area` VARCHAR(45) NULL,
    PRIMARY KEY (`risk_level`, `control_type`));

  CREATE TABLE `phone_ip_location` (
    `street_number` VARCHAR(45) NOT NULL,
    `street_name` VARCHAR(45) NOT NULL,
    `area` VARCHAR(45) NOT NULL,
    `city` VARCHAR(45) NOT NULL,
    `province` VARCHAR(45) NOT NULL,
    `time` datetime NOT NULL,
    `ID` VARCHAR(45) NOT NULL,
    PRIMARY KEY (`street_number`, `street_name`, `area`, `city`,
`province`, `time`, `ID`),
    INDEX `ip_citizens_fk_idx` (`ID` ASC) VISIBLE,
    CONSTRAINT `ip_citizens_fk`
      FOREIGN KEY (`ID`)
      REFERENCES `citizens` (`ID`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION);
  ALTER TABLE `phone_ip_location`
  ADD CONSTRAINT `ip_regions_fk`
    FOREIGN KEY (`street_number` , `street_name` , `area` , `city` ,
```

```
`province`)
    REFERENCES
  `regions_outside_a` (`street_number` , `street_name` , `area` ,
`city` , `province`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION;

  CREATE TABLE `vaccine_inoculation` (
    `company` VARCHAR(45) NOT NULL,
    `type` VARCHAR(45) NOT NULL,
    PRIMARY KEY (`company`, `type`));

  CREATE TABLE `vaccine_appointment_and_result` (
    `ID` VARCHAR(45) NOT NULL,
    `type` VARCHAR(45) NOT NULL,
    `company` VARCHAR(45) NULL,
    `time` datetime NULL,
    PRIMARY KEY (`ID`, `type`),
    CONSTRAINT `appointment_citizens_fk`
      FOREIGN KEY (`ID`)
      REFERENCES `citizens` (`ID`)
      ON DELETE NO ACTION
  ON UPDATE NO ACTION);
  ALTER TABLE
  `vaccine_appointment_and_result`
  ADD INDEX `appiointment_vaccine_fk_idx` (`company` ASC, `type`
ASC) VISIBLE;

  ALTER TABLE
  `vaccine_appointment_and_result`
  ADD CONSTRAINT `appiointment_vaccine_fk`
```

```
    FOREIGN KEY (`company` , `type`)
    REFERENCES `vaccine_inoculation`
(`company` , `type`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION;

CREATE TABLE `hospital` (
    `street_number` VARCHAR(45) NOT NULL,
    `street_name` VARCHAR(45) NOT NULL,
    `area` VARCHAR(45) NOT NULL,
    `company` VARCHAR(45)    NOT NULL,
    `type` VARCHAR(45)    NOT NULL,
    `name` VARCHAR(45) NULL,
    PRIMARY KEY (`street_number`, `street_name`, `area`,`company` ,
`type`),
    INDEX  `hospital_vaccine_fk_idx`  (`company`  ASC,  `type`  ASC)
VISIBLE,
    CONSTRAINT `hospital_street_fk`
      FOREIGN KEY (`street_number` , `street_name` , `area`)
      REFERENCES `streets_in_a`
(`street_number` , `street_name` , `area`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
    CONSTRAINT `hospital_vaccine_fk`
      FOREIGN KEY (`company` , `type`)
      REFERENCES `vaccine_inoculation`
(`company` , `type`)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION);

CREATE TABLE `nucleic_acid_tests` (
```

```
    `type` VARCHAR(45) NOT NULL,
    `laboratory` VARCHAR(45) NOT NULL,
    `fee` VARCHAR(45) NULL,
    PRIMARY KEY (`type`, `laboratory`));

CREATE TABLE `test_appointment_and_result` (
    `ID` VARCHAR(45) NOT NULL,
    `time` datetime NOT NULL,
      `type` VARCHAR(45) NULL,
    `laboratory` VARCHAR(45) NULL,
    `result` VARCHAR(45) NULL,
    PRIMARY KEY (`ID`,`time`),
    INDEX `tests_fk_idx` (`type` ASC, `laboratory` ASC) VISIBLE,
    CONSTRAINT `id_fk`
        FOREIGN KEY (`ID`)
        REFERENCES `citizens` (`ID`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT `tests_fk`
        FOREIGN KEY (`type` , `laboratory`)
        REFERENCES `nucleic_acid_tests` (`type` , `laboratory`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION);
CREATE TABLE `test_sites` (
    `street_number` VARCHAR(45) NOT NULL,
    `street_name` VARCHAR(45) NOT NULL,
    `area` VARCHAR(45) NOT NULL,
    `type` VARCHAR(45) NOT NULL,
    `laboratory` VARCHAR(45) NOT NULL,
    `site_id` VARCHAR(45) NULL,
    PRIMARY  KEY  (`street_number`,  `street_name`,  `area`,  `type`,
```

```
`laboratory`),
        INDEX `test_fk_idx` (`type` ASC, `laboratory` ASC) VISIBLE,
        CONSTRAINT `street_fk`
          FOREIGN KEY (`street_number` , `street_name` , `area`)
          REFERENCES `streets_in_a` (`street_number` , `street_name` ,
`area`)
          ON DELETE NO ACTION
          ON UPDATE NO ACTION,
        CONSTRAINT `test_fk`
          FOREIGN KEY (`type` , `laboratory`)
          REFERENCES `nucleic_acid_tests` (`type` , `laboratory`)
          ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

2. 使用视图的业务案例

（1） 管理端统计本市各区市民核酸检测情况

例如，统计 72h 内核酸检测为阴性的市民需要视图便于重复查询，

```
drop view if exists negitive_72h_citizens;
create view negitive_72h_citizens as
select id,type,time
from test_appointment_and_result
where
result='negative' and TIMESTAMPDIFF(DAY,time,now())<3;
select*from negitive_72h_citizens ;
```

（2） 管理端人员统计本市途径风险地区的人员信息，需要视图便于重复查询：

```
drop view if exists riskzone_outside_A;
create view   riskzone_outside_A   as
select distinct street_name,area,risk_level,control_type from
regions_outside_a
where risk_level!='low' or control_type!='free'
group by risk_level,control_type;
```

```
select * from riskzone_outside_A;

drop view if exists key_regions;
create view key_regions as
select street_name,area,city,province
from regions_outside_a natural join key_area_standard
where
is_key_area='yes';
select* from    key_regions;

drop view if exists key_citizens;
create view key_citizens as
select id as
key_id,name,phone_number,citizens.area,citizens.street_name,
phone_ip_location.street_name as route_street,phone_ip_location.area
as route_area,phone_ip_location.city,phone_ip_location.province,time
from phone_ip_location join key_regions
using(street_name,area,city,province) join citizens using(id)
where TIMESTAMPDIFF(DAY,time,now())<14;
select* from key_citizens;
```

（3） 统计本市疫情风险区居民信息

例如统计封控区居民信息，需要创建视图便于重复查询：

```
create view locked_down_zone_citizens as
select id, name,phone_number,street_number,street_name,area
from citizens natural join control_type_in_a
where control_type='locked down'
group by street_name,area;
```

3. 使用触发器的业务案例

（1） 如在预约核酸检测前，要检查疫苗接种时间是否已超过 24h，若未超过 24h，不能进行核酸检测。可以设计一个 trigger before update 和 trigger before insert(如果从未有过核酸检测记录)。

```
drop trigger if exists test_trg;
DELIMITER $$
create trigger test_trg after insert
on test_appointment_and_result for each row
begin
if(select ID from vaccine_appointment_and_result
where ID=NEW.ID and TIMESTAMPDIFF(DAY,time,now())<1)is not null
-- then insert into test_appointment_and_result values (NEW.ID,NEW.time,NEW.type,NEW.laboratory,NEW.result);
then delete from test_appointment_and_result where ID=NEW.ID and time=NEW.time;
end if;
end$$
DELIMITER ;
```

（2） 如在预约接种疫苗时，选择类型，要检查上一针是否已经接种且该针是否未接种，如果未接种或改针已接种不能预约这一针疫苗。

```
-- 接种疫苗触发器
drop trigger if exists vaccine_trg;
DELIMITER $$
create trigger vaccine_trg after insert
on vaccine_appointment_and_result for each row
begin
select time into @temp1 from vaccine_appointment_and_result
where id=NEW.id and type=NEW.type;
select time into @temp2 from vaccine_appointment_and_results
where id=NEW.id and type='1';
select time into @temp3 from vaccine_appointment_and_result
where id=NEW.id and type='2';
```

```
IF (@temp1 is null and NEW.type='2'and @temp2 is not null)
or (@temp1 is null and NEW.type='3'and @temp3 is not null )
or( @temp1 is null and NEW.type='1')
THEN delete from vaccine_appointment_and_result where ID=NEW.ID
and type=NEW.type;
END IF;
end$$
DELIMITER ;
```

4. 使用储存函数、储存过程的业务案例：

居民端查询个人健康信息业务、预约核酸检测、疫苗接种业务都需要用到储存函数或储存过程，函数传入参数、过程输入为身份证号。

以查询健康码需要的函数为例的 sql 语句：

```
-- 储存函数
-- 健康码
set global log_bin_trust_function_creators=TRUE;
drop function if exists health_code;
DELIMITER $$
create function health_code (in_id varchar(45))
returns varchar(45)
begin
declare color varchar(45);
declare id_result varchar(45);
declare id_test_time varchar(45);
declare id_control_type varchar(45);
declare id_risk_level varchar(45);
declare temp varchar(45);
set color='green';
-- 核酸检测结果
select result into id_result
from test_appointment_and_result
where test_appointment_and_result.id=in_id and result is not null;
```

```
select time into id_test_time
from test_appointment_and_result
where test_appointment_and_result.id=in_id and result is not null;
-- 防控类型
select control_type into id_control_type
from citizens natural join control_type_in_a
where citizens.id=in_id;
-- 风险类型
select risk_level into id_risk_level
from citizens natural join risk_level_in_area
where citizens.id=in_id;
-- 14d 内是否途径排查地区 如果途径,count(*)>0
select count(*) into temp
from key_citizens;
-- 黄码
if id_risk_level='high' or (temp>0 and (id_result is null or
TIMESTAMPDIFF(DAY, id_test_time,now())>3))then set color='yellow';
end if;
if id_result='positive' or id_control_type='locked_down' then set
color='red';
end if;
return color;
end$$
DELIMITER ;
select health_code('12341');
select count(*) from key_citizens where key_id='12341';
```

5. 需要事务处理的业务案例

假设市民选定医院进行疫苗接种，向预约疫苗系统界面提交的信息也要同步到对应医院的系统，两个系统中有一个系统插入数据失败，事务回滚；市民核酸检测结果查询界面显示的应该是最新的出了结果

的检测信息和最新未出结果检测信息，（如果没有未出结果的，就显示一条已出结果的）如果出了新结果，原来那条已出结果的就要被删除，此时成功更新新结果和删除原来的结果构成一个事务，如果删除失败或更新失败，都要回滚至原状态。

6. 需要统计分析的业务案例

（1） 防疫管理端统计收集某个条件下的市民总数，比如某天核酸检测为阳性的市民人数，作为 A 市新增阳性案例；统计 14 天内从某重点排查地区返市的市民人数；统计市民疫苗三针接种人数等，都可以用 count(*)聚合函数结合存储函数实现。

示例：当天新增阳性案例数的 sql 语句实现

```
select count(*) from test_appointment_and_result
where result='positive' and TIMESTAMPDIFF(DAY,time,now())=0;
```

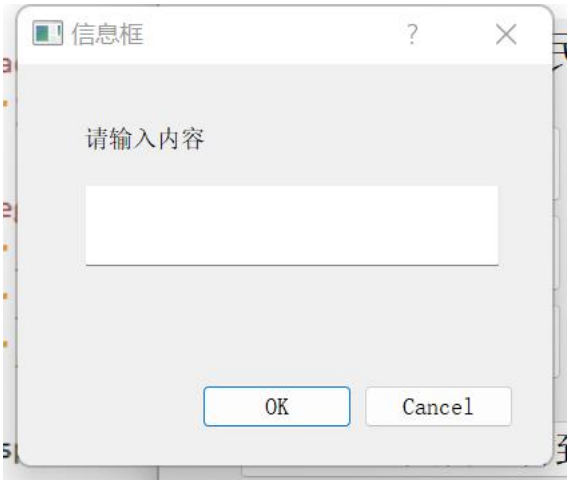
（2） 如防疫管理人员统计封控区居民信息或居民总数，用 group by area, street_name 分组统计各区居民信息。前面已有案例包含 group by 的应用。

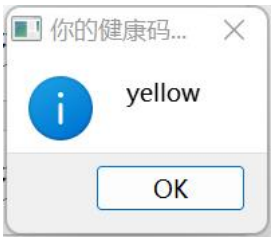
五、完整应用系统

开发软件：QT 5.9.2 qtcreator,MySQL workbench 开发语言：c++,sql。

1. GUI 界面

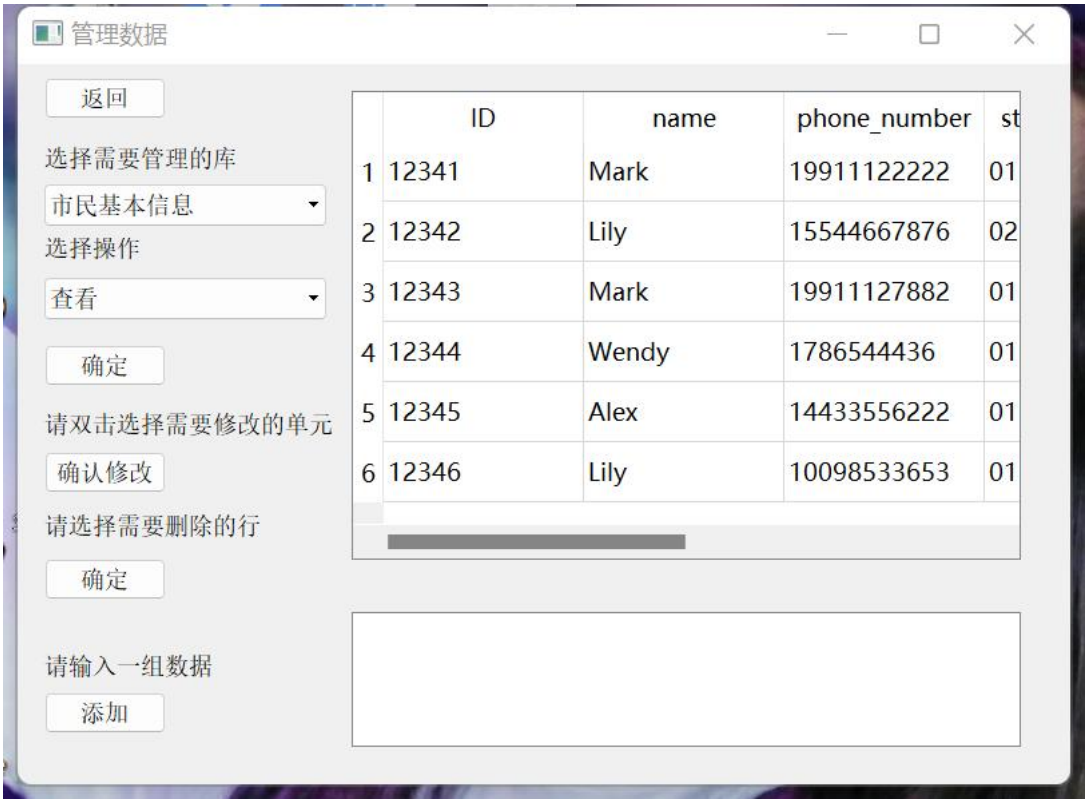

市民服务
返回
市民
输入身份证号
健康码
预约疫苗接种
查看核酸检测结果
市内风险区
预约核酸检测
查看疫苗接种结果
各地风险区
14天内是否到过风险区


信息框
请输入内容
OK
Cancel


你的健康码...
yellow
OK


疾控管理端
返回
管理端
管理、更新数据
统计数据


疫苗接种预约
疫苗接种预约
当前，你已接种
点击初始化
提供接种服务的医院及疫苗类型
type
company
街道号
街道
区
疫苗公司
第几针
医院名字
请在右表中选择服务
请选择日期
六月, 2022
周一
周二
周三
周四
周五
周六
周日
时间:
08
00
确定预约
返回


管理数据
返回
选择需要管理的库
市民基本信息
选择操作
查看
确定
请双击选择需要修改的单元
确认修改
请选择需要删除的行
确定
请输入一组数据
添加
ID
name
phone_number
st
12341
Mark
19911122222
01
12342
Lily
15544667876
02
12343
Mark
19911127882
01
12344
Wendy
1786544436
01
12345
Alex
14433556222
01
12346
Lily
10098533653
01

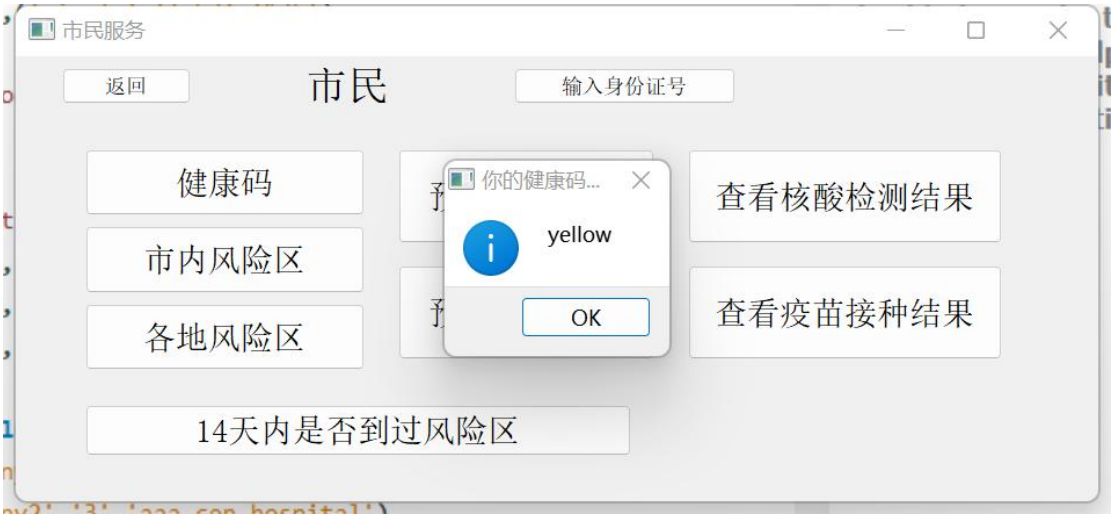
2. 功能简介

先在 MySQL 中运行 final_hw_data_2.sql 文件导入案例数据，final_hw_function&view&trigger.sql 导入存储触发器，视图和函数。

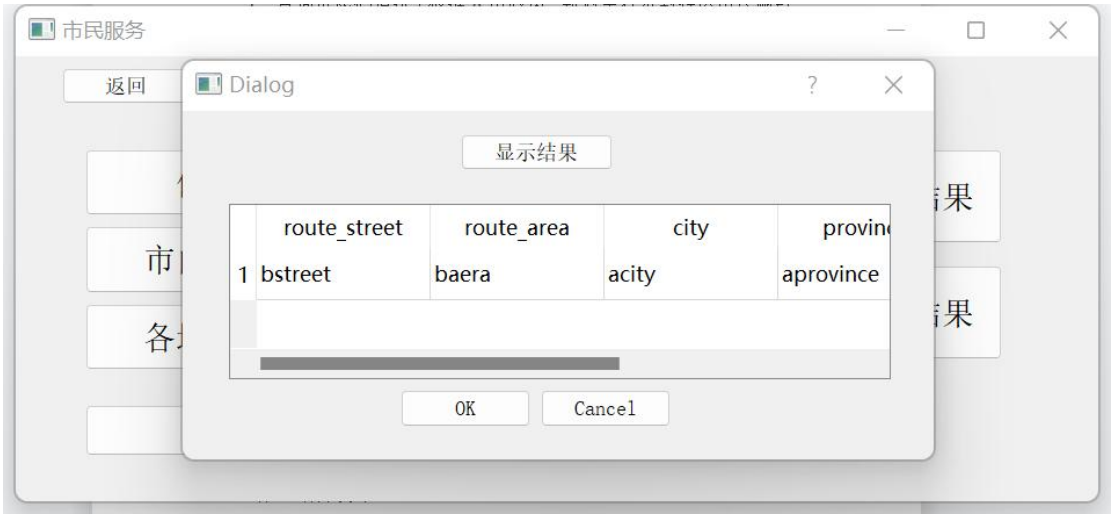
（1） 市民业务：

点击输入身份证号初始化信息。

弹出信息框，输入案例 ID:12341。



点击查看健康码，为黄码。

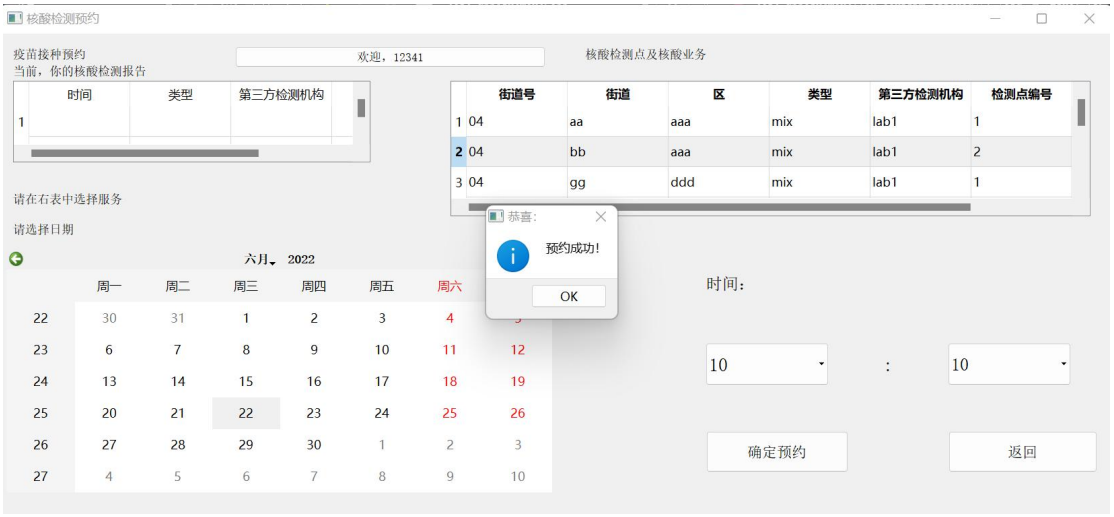


点击查看 14 天内是否到过风险区，弹出对话框后点击显示结果，发现到过。

点击核酸检测预约，进入界面：

点击初始化按钮：

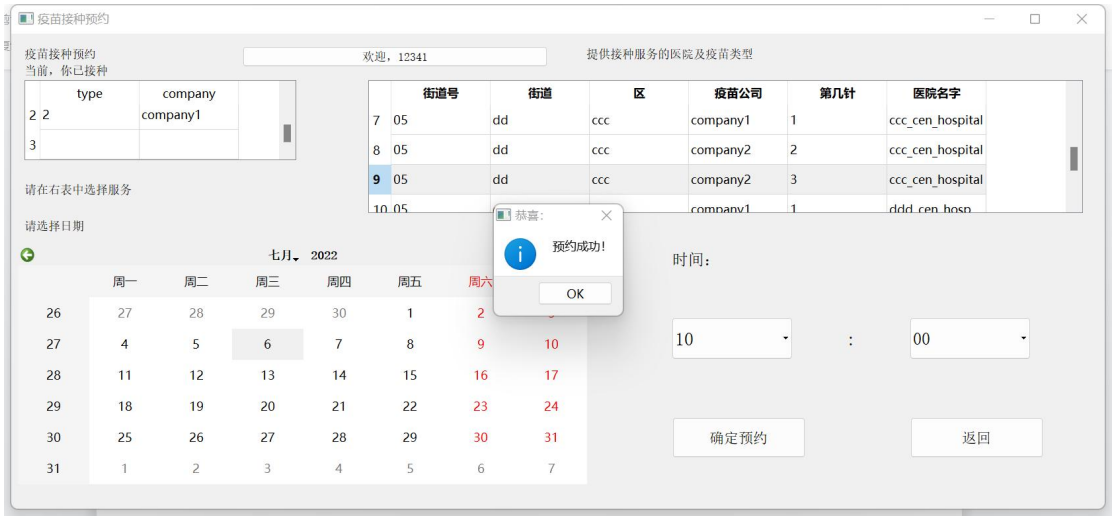
选择时间，业务，点击“确定预约”：



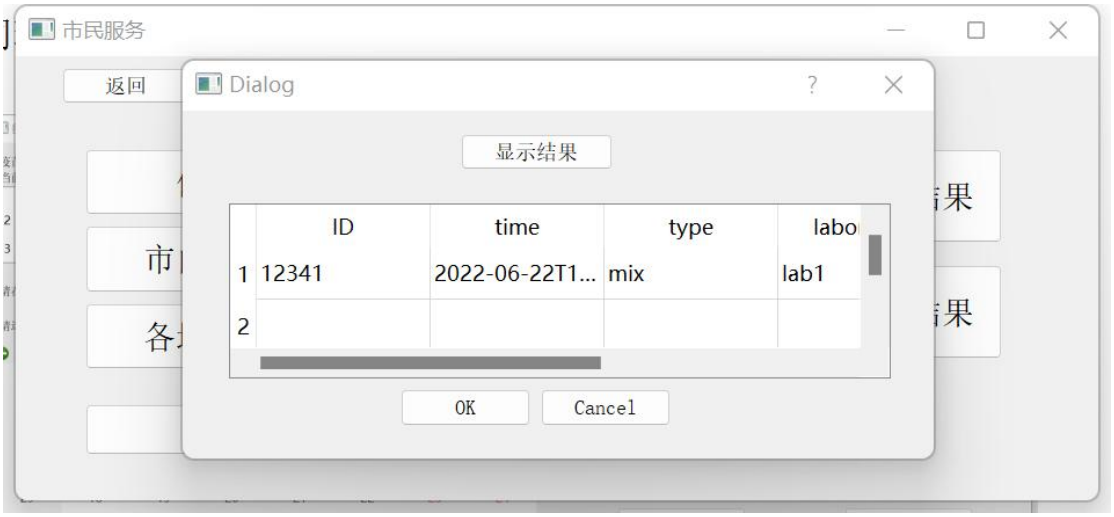
成功后再按一次“欢迎，12341”，可见更新。

结果还未出，为空。

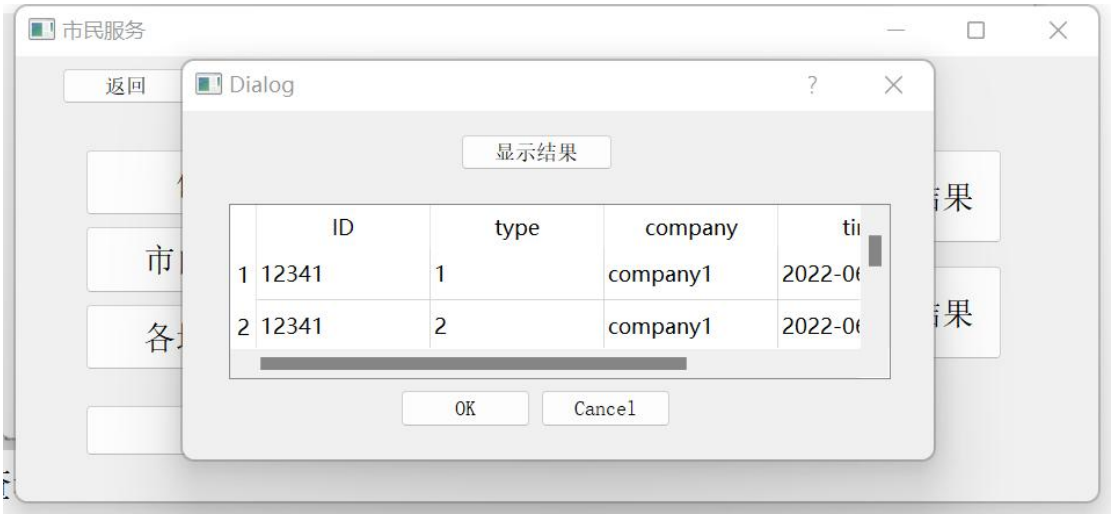
同理，预约疫苗业务：



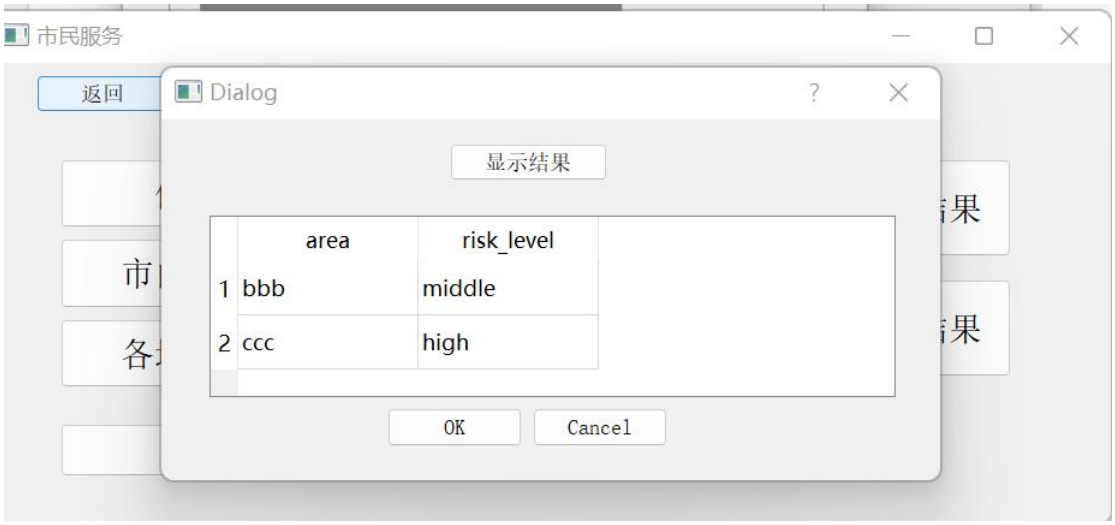
查看核酸检测结果：



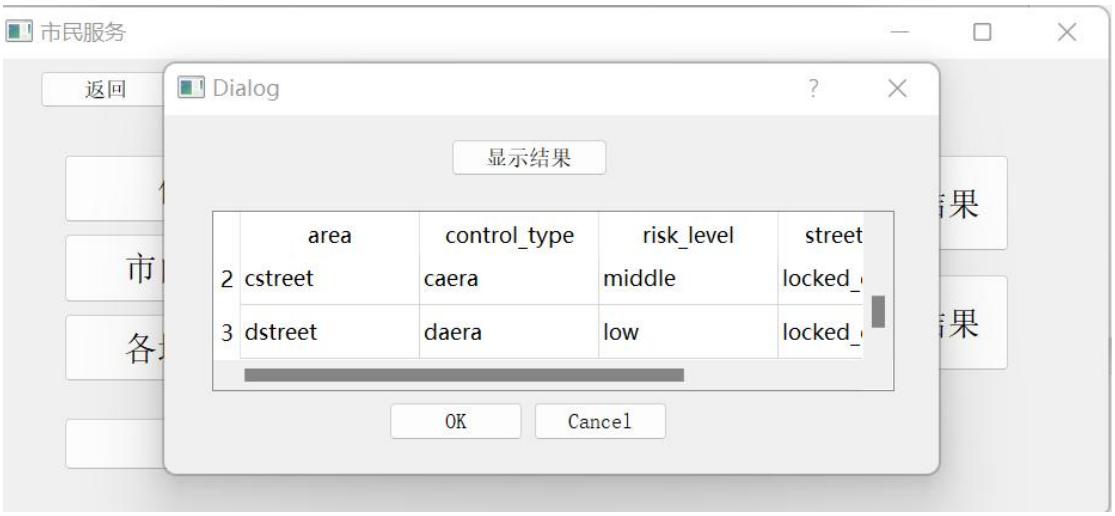
查看疫苗结果：



市内风险区：



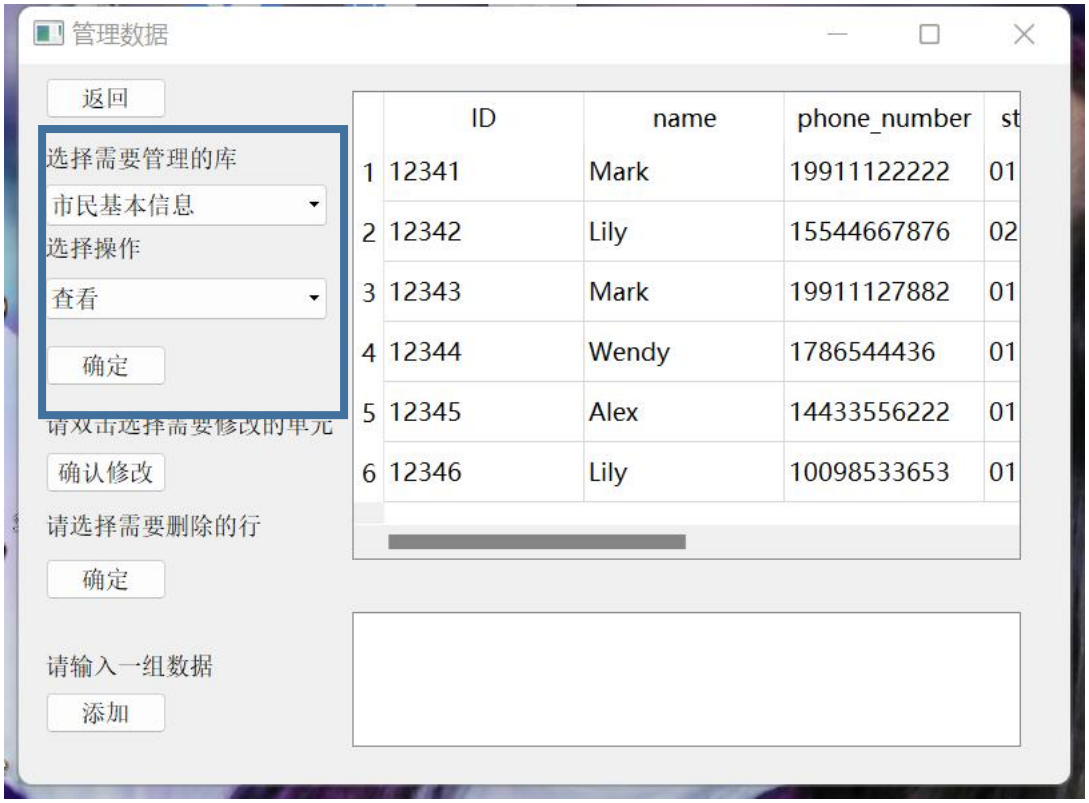
各地风险区：



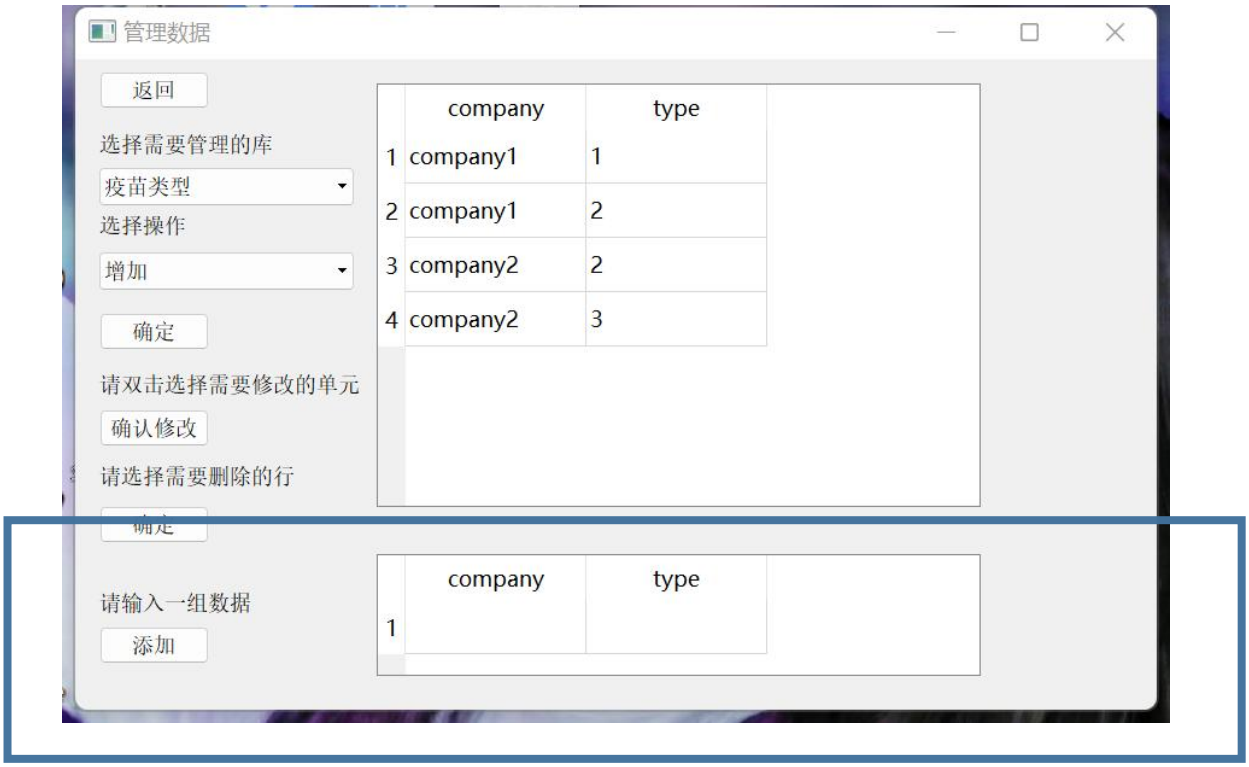
（2） 管理人员业务

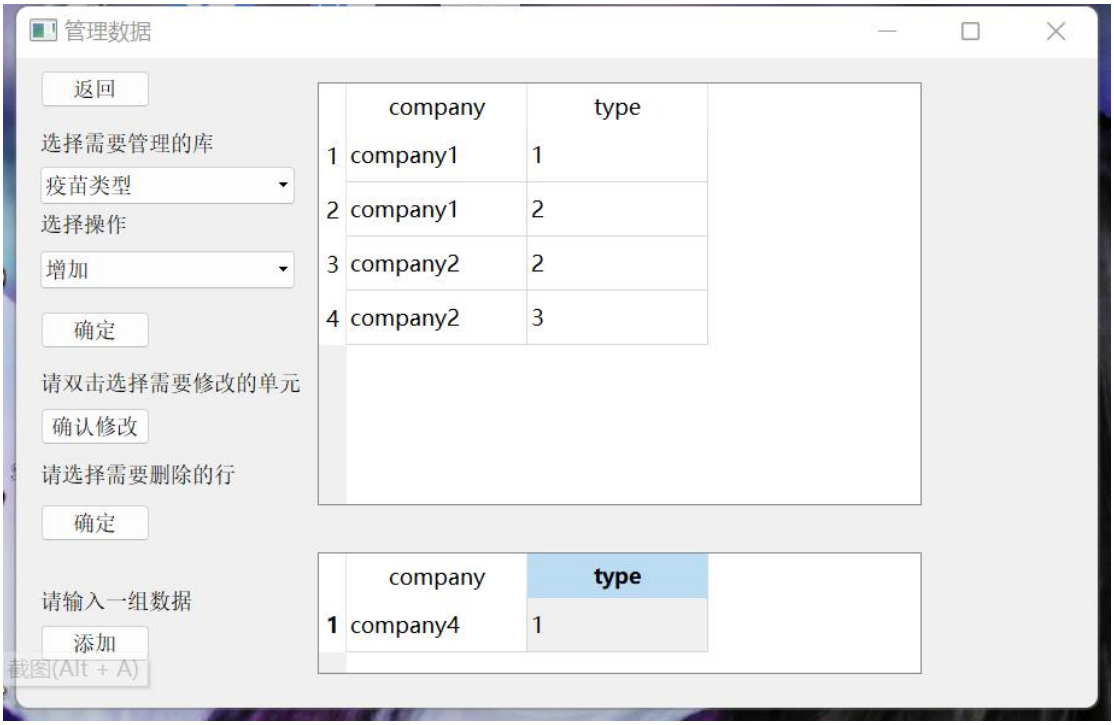
点击“管理数据”进入页面：

图标注框中选择管理的库和查看功能，点“确定”，右表能查看信息。

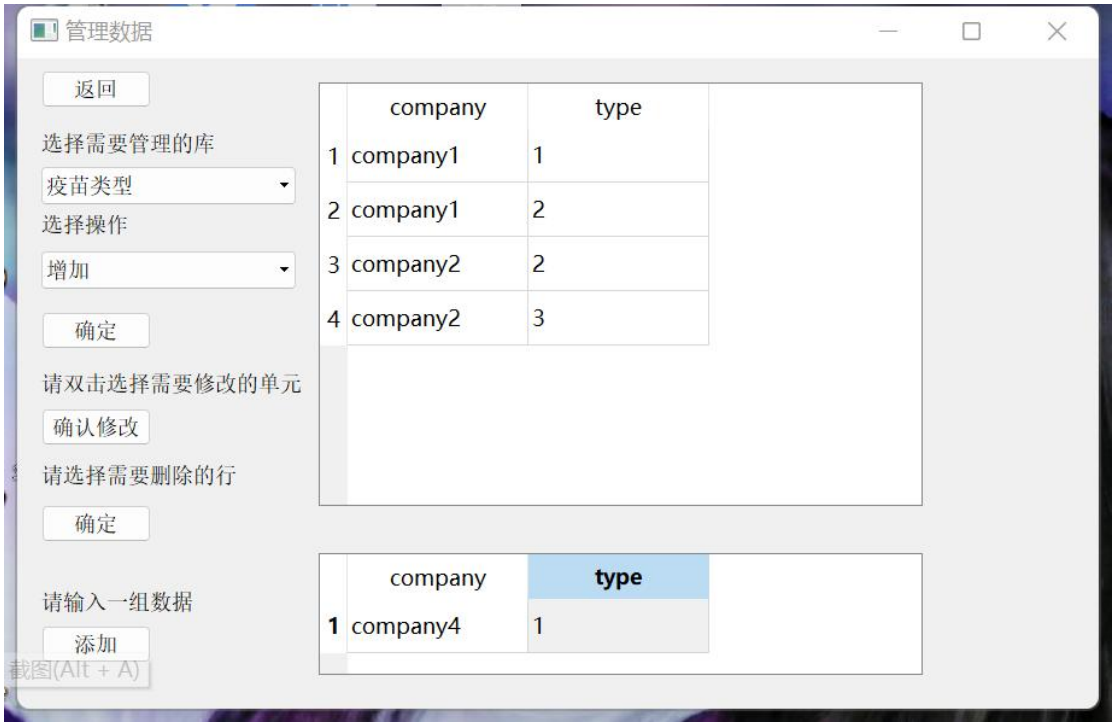


操作转换为增加，点击确定，图标注框显示输入编辑框。

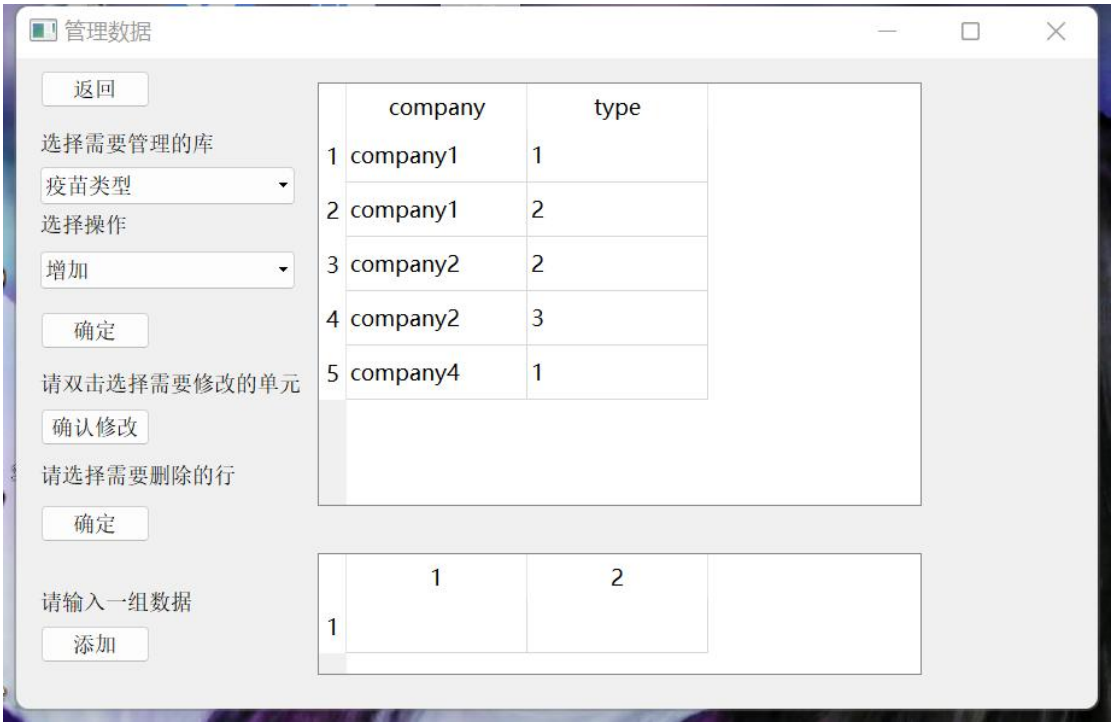




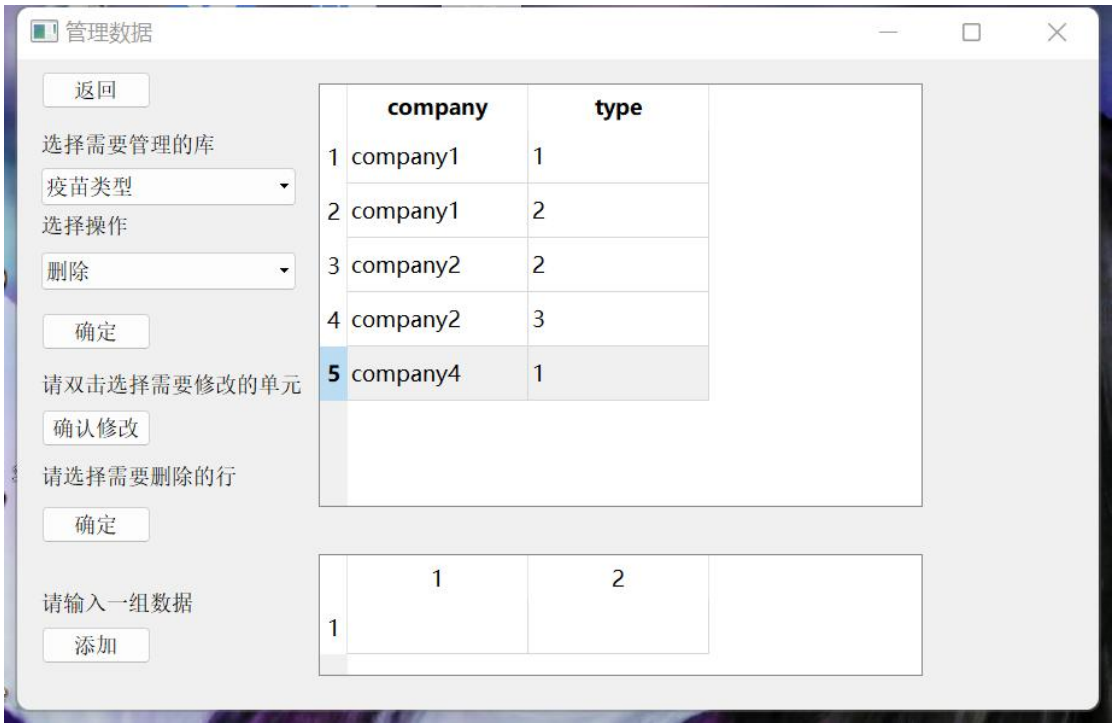
输入数据，按“添加”


管理数据
返回
选择需要管理的库
疫苗类型
选择操作
增加
确定
请双击选择需要修改的单元
确认修改
请选择需要删除的行
确定
请输入一组数据
添加
company
type
1 company1 1
2 company1 2
3 company2 2
4 company2 3
company
type
1 company4 1

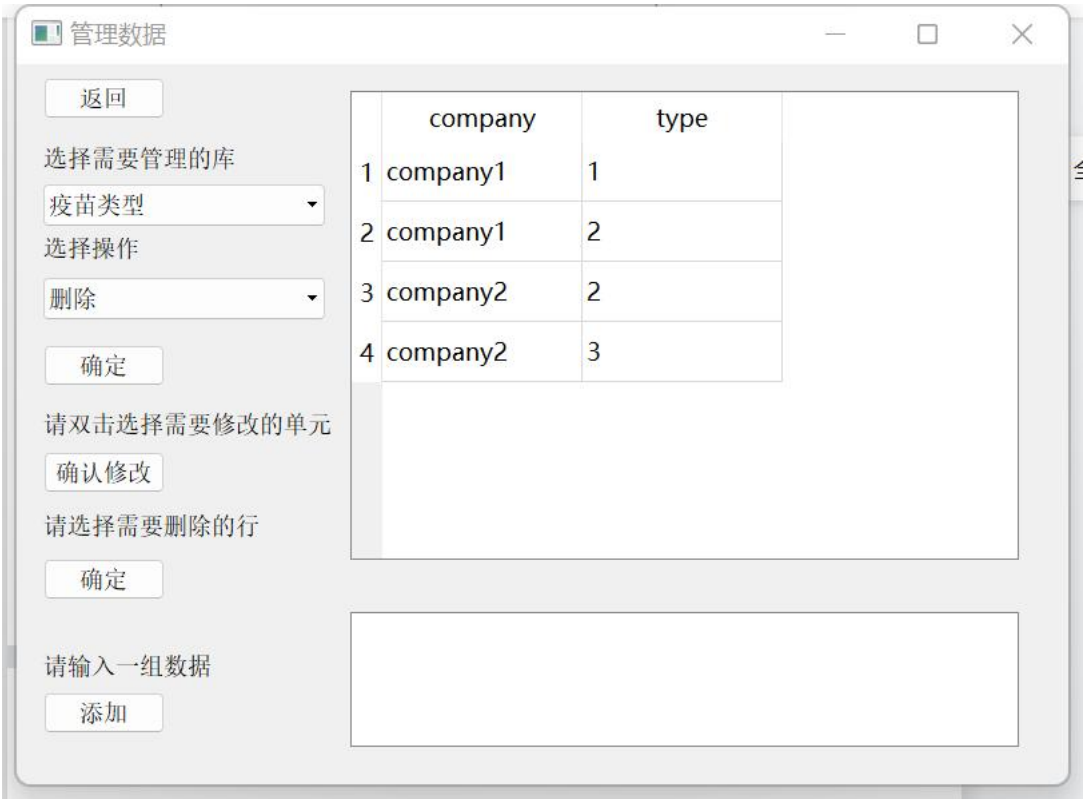
添加成功：


管理数据
返回
选择需要管理的库
疫苗类型
选择操作
增加
确定
请双击选择需要修改的单元
确认修改
请选择需要删除的行
确定
请输入一组数据
添加
company
type
1 company1 1
2 company1 2
3 company2 2
4 company2 3
5 company4 1
1
2
1

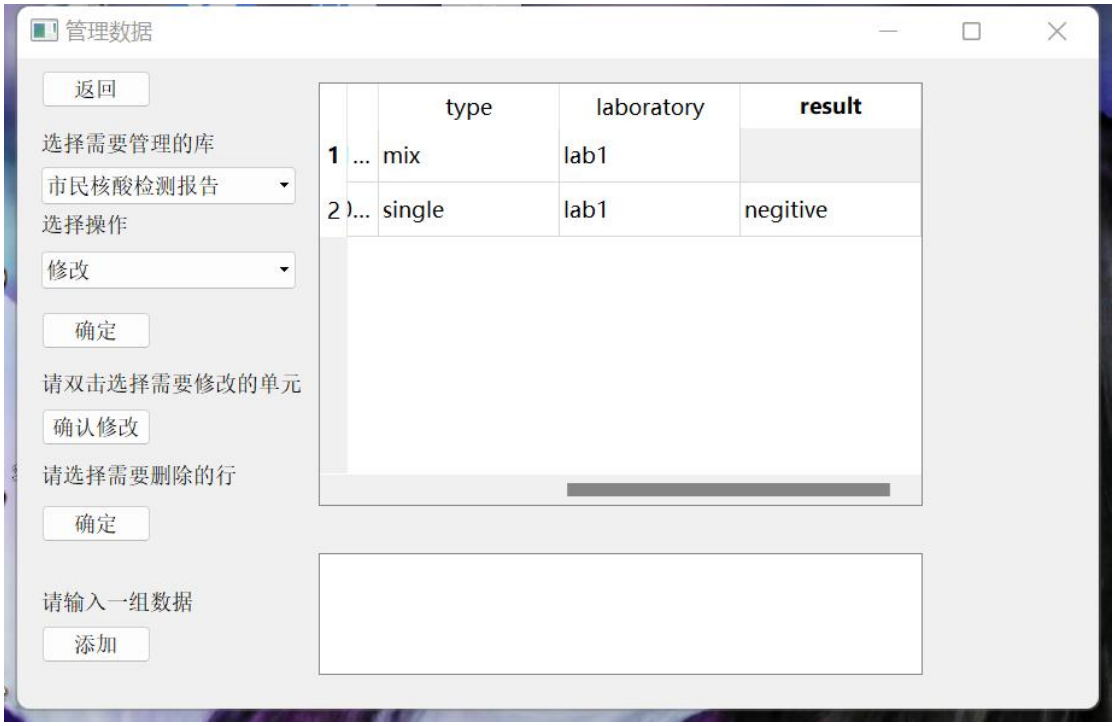
转为删除操作，按确定，选中一行：

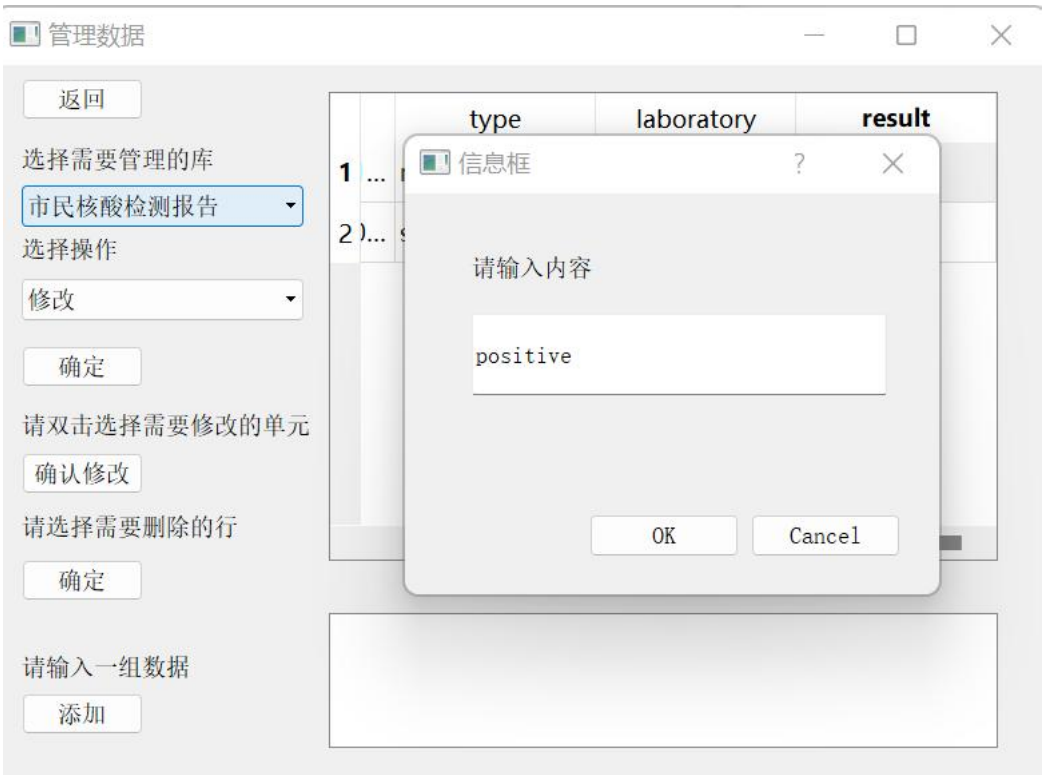


点击“需要删除的行”下面的确定按钮，删除结果：

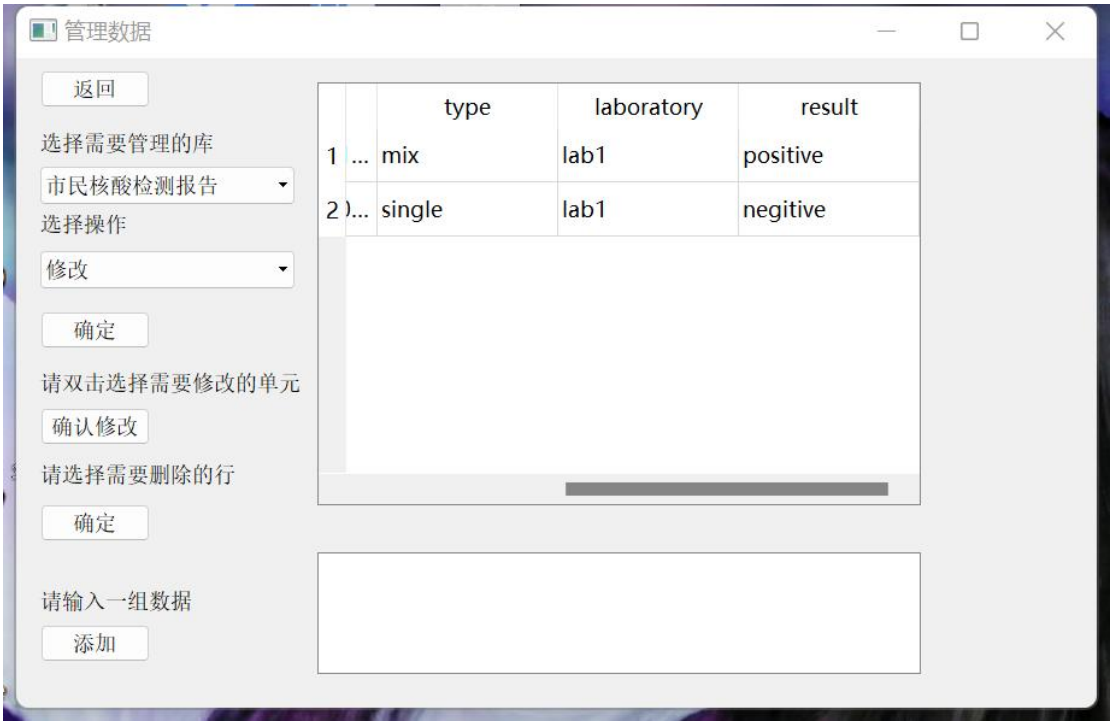


转为“修改”，“市民核酸检测报告”，双击要修改的单元格：





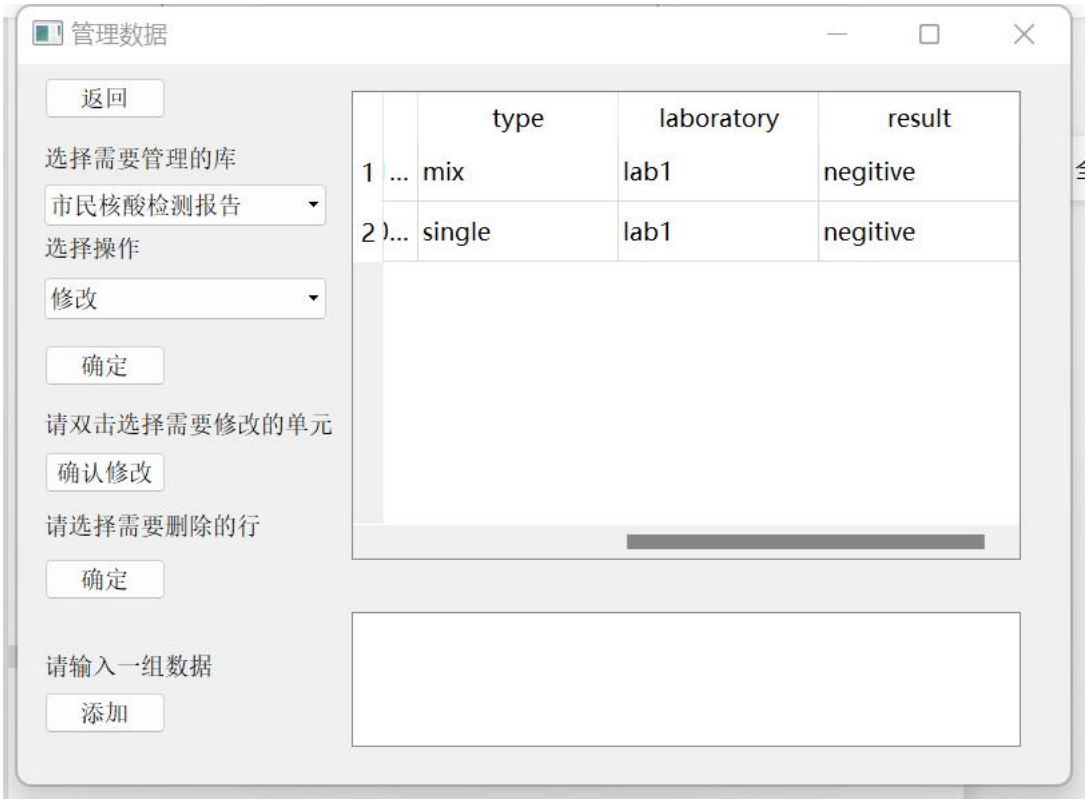
点击“确定修改”：

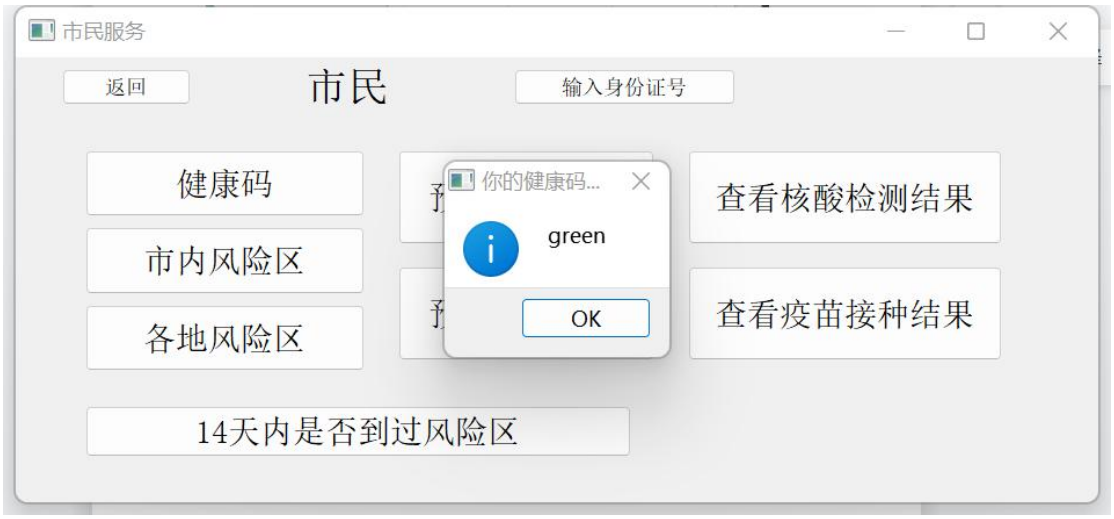


修改操作的实际应用为，把市民核酸检测结果录入。

此时回到市民端，可见健康码变红：

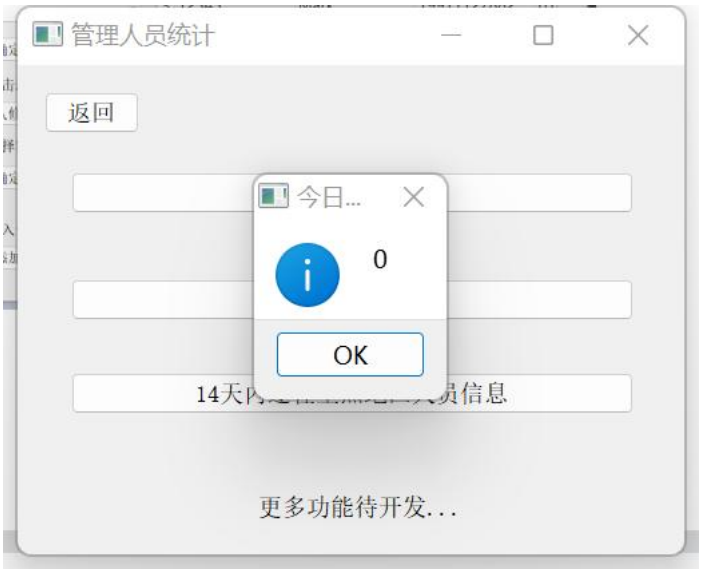
再在管理端修改成“negative”，健康码变绿：



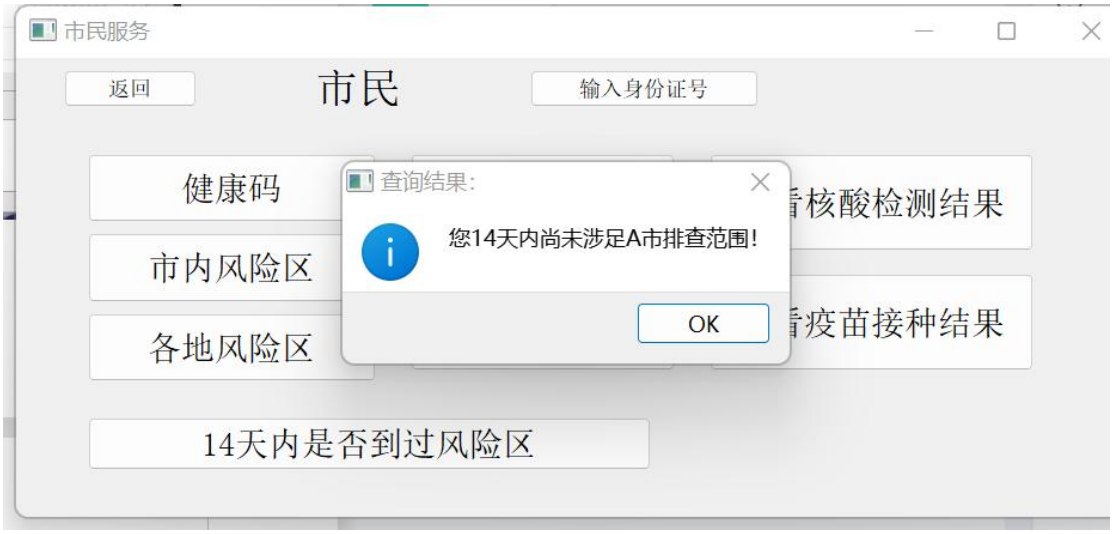


管理统计数据端：（功能较简单，待更进一步的开发）

点击“新增阳性案例”：



市民端补充：输入 ID：12344，未去过管控范围，则点击“14 天内是否去过管控范围区域”，会弹出消息框：



3. 系统的问题反思

管理人员统计端功能待开发完善；系统异常处理机制待开发完善，在数据库中有外键约束和触发器条件约束，但在应用系统中如果执行 sql 语句失败会引发异常直接退出，系统不稳定，应开发与用户端交互、进行约束条件提示的功能；缺乏数据库安全、隐私保护，应在应用系统中设置用户名密码验证，才能进入管理系统或查询对应身份证的信息；应用系统的实现语句，用的是 QsqlQuery 类，提交至数据库

执行的都是比较简单的查询语句，而复杂功能在 QT 端用实现，这样可能导致代码冗余，构建、编译的时间长，性能较差；应用系统没有使用线程类使客户端，服务端分离，实时同步。