# 1.课程任务概述

简要陈述介绍本实践课程的各项任务要求。

## 1.1 软件功能学习

1. 练习SQL Server或其他某个主流关系数据库管理系统软件的备份方式：数据和日志文件的脱机备份、系统的备份功能。
2. 练习在新增的数据库上增加用户并配置权限的操作，通过用创建的用户登录数据库并且执行未经授权的SQL语句验证自己的权限配置是否成功。

## 1.2 SQL练习

### 1.2.1 创建数据库和表

**第1关 创建用于新冠疫情常态防控的数据库**

1. 任务描述：创建数据库。
2. 数据库信息说明：创建用于管理新冠疫情防控的数据库:covid19mon。

**第2关 创建表和主外码约束**

1. 任务描述：建立新冠常态防控信息系统所用到的表，为表设置主码、外码，并定义主外码之间的参照完整性。
2. 表信息说明：

表 1-1 人员表(person)



表 1-2 地点表(location)



表 1-3 行程表(itinerary)



表 1-4 诊断表(diagnose\_record)



表 1-5 密切接触者表(close\_contact)



表 1-6 隔离表(isolation\_record)



表 1-7 隔离地点表(isolation\_location)



### 1.2.2 表的更新

**第1关 数据插入**

1. 任务描述：练习insert语句,向人员表person插入数据。
2. 插入信息：

表 1-8 插入人员数据



**第2关 数据删除**

任务描述：删除前一关中插入的人员编号为2，姓名为’李大锤’的这条数据。

**第3关 修改数据**

任务描述：将姓名为“张小敏”,人员编号为1的电话号码改为13607176668。

### 1.2.3 数据查询

**第1关 人流量大于30的地点**

1. 任务描述：查询累计人流量大于30的地点名称和累计人流量。
2. 编程要求：查询累计人流量大于30的地点名称和累计人流量，请用visitors作累积人流量的标题名称。查询结果按照人流量从高到低排序,人流量相同时，依地点名称顺序排序。

**第2关 每个隔离点正在进行隔离的人数**

1. 任务描述：查询每个隔离地中正在进行隔离的人数，并按数量由多到少排序。
2. 编程要求：查询每个隔离地及该地正在进行隔离的人数，以number为隔离人数的标题.查询结果依隔离人数由多到少排序，隔离人数相同时，再依隔离点名称顺序排序。

**第3关 接续行程**

1. 任务描述：查询行程表中同一个人接续行程的时间和地点信息。所谓接续行程指同一个人第一段行程的结束时间与第二段行程的开始时间重合。
2. 编程要求：查询人员编号大于30的接续行程，输出信息包括： 人员编号,姓名,重合时间,起始地点id,起始地点,结束地点id,结束地点。查询结果依人员编号排序。

**第4关 充珉瑶和贾涵山的行程情况**

1. 任务描述：查询充珉瑶和贾涵山的行程情况。
2. 编程要求：查询充珉瑶和贾涵山的行程情况。查询结果包括：姓名、电话、到过什么地方（地名），何时到达，何时离开。列名原样列出即可，不必用别名。查询结果依人员编号降序排序。同一人员行程依行程开始时间顺序排列。没有任何出行记录的，在出行有关的栏目内均填写NULL即可。

**第5关 地名中带有‘店’字的地点名称**

1. 任务描述：查询地名中带有‘店’字的地点编号和名称。查询结果按地点编号排序。
2. 编程要求：查询地名中带有‘店’字的地点编号和名称。查询结果按地点编号排序。

**第6关 确诊者的接触者**

1. 任务描述：查询确诊者的接触者。
2. 编程要求：新发现一位确诊者，已知他在2021.2.2日20:05:40到21:25:40之间在“活动中心”逗留，凡在此间在同一地点逗留过的，视为接触者，请查询接触者的姓名和电话。查询结果按姓名排序。

**第7关 仍在使用的隔离点**

1. 任务描述：查询仍正在使用的隔离点名称。
2. 编程要求：查询仍在使用的隔离点名称。注意，隔离记录里如果只有隔离结束或确诊转院的记录，表明该隔离点已暂时停用，只要还有一个人在此处隔离，表明该隔离点仍在使用。查询结果按隔离点编号排序。

**第8关 查询有出行记录的人员**

1. 任务描述：用带EXISTS关键字的子查询,查询有有出行记录的人员及其联系电话。
2. 编程要求：查询前30位有出行记录的人员姓名和电话。查询结果按照人员编号排序。

**第9关 没有去过“Today便利店“的人数**

1. 任务描述：统计查询人员表中没有去过地点“Today便利店”的人数（使用NOT EXISTS 关键字）。
2. 编程要求：查询人员表中没有去过地点“Today便利店”的人数。请给统计出的人数命名为number。

**第10关 去过所有地点的人员**

1. 任务描述：查询人员表去过所有地点的人员姓名。查询结果依人员姓名的字典顺序排序。
2. 编程要求：查询人员表去过所有地点的人员姓名。查询结果依人员姓名顺序排序。

**第11关 隔离点的现状视图**

1. 任务描述：建立隔离点现状的视图，视图命名为isolation\_location\_status，内容包括：地点编号，隔离地点名，房间容量，已占用量。请保持原列名不变，已占用量由统计函数计算得出，该计算列命名为occupied。
2. 编程要求：

视图名称：isolation\_location\_status

内容包括：隔离地点编号，隔离地点名，房间容量，已占用量。

请保持原列名不变，已占用量由统计函数计算得出，该计算列命名为occupied。只有正在该隔离点隔离的人才占用隔离点的位置。隔离结束或已转院的人表明已经腾出了原有位置，不再占用资源。

**第12关 各隔离点的剩余房间数**

1. 任务描述：从视图isolation\_location\_status中查询各隔离点的剩余房间数。
2. 编程要求：从视图isolation\_location\_status中查询各隔离点的剩余空房间的数目。需要列出的数据项为：隔离点名称，剩余房间数。其中剩余房间数为计算得出，请给该列命名为available\_rooms。查询结果依隔离点编号排序。

**第13关 与无症状感染者靳宛儿有过接触的人**

1. 任务描述：筛查发现，靳宛儿为无症状感染者。现需查询其接触 者姓名和电话，以便通知并安排隔离。凡行程表中，在同一地点逗留时间与靳宛儿有交集的，均视为接触者。
2. 编程要求：查询靳宛儿接触者的姓名和电话。与靳宛儿在同一地点逗留时间有交集的均为其接触者。查询结果按照人员姓名排序。

**第14关 每个地点发生的密切接触者人数**

1. 任务描述：查询每个地点的密切接触者的数量，列出内容包括：地点名称，密接者人数。
2. 编程要求：依据密切接触表的内容查询每个地点的密切接触者的数量，列出内容包括：地点名称，密接者人数。人数由统计获得，列名命名为close\_contact\_number.查询结果依密接者人数降序排列。密接者人数相同时，依地点名称排序。

**第15关 感染人数最多的人**

1. 任务描述：查询感染人数最多的人。
2. 编程要求：查询感染人数最多的人员编号，姓名，和被其感染的人数。感染人数由统计所得，命名为infected\_number。

**第16关 行程记录最频繁的3个人**

1. 任务描述：查询2021-02-02 10:00:00到14:00:00期间，行程记录最频繁的三个人及行程记录的条数。
2. 编程要求：查询2021-02-02 10:00:00到14:00:00期间，行程记录最频繁的三个人的姓名及行程记录条数。记录条数命名为record\_number。如果出现记录数并列情况，再按姓名排序。

**第17关 房间数第2多的隔离点**

1. 任务描述：查询隔离点中，房间数(capacity)居第二多的隔离点名称及其房间数。
2. 编程要求：从隔离点中，查询房间数(capacity)居第二多的隔离点名称及其房间数。
3. 注意：如果房间数从多到少依次有100,100,80,80,70,…，那么，两个80都是第2多。而两个100都是第1多。

### 1.2.4 创建触发器

**第1关 隔离点的人员“确诊新冠”后，自动转院**

1. 任务描述：

隔离点的人员一旦确诊“新冠”后，将被转入医院，请编写一个触发器，用于实现以下完整性控制规则：

当隔离表（isolation\_record）中的某位隔离人员在诊断表（diagnose\_record）中的诊断结果（result）为1(新冠确诊)”时，自动将隔离表中的隔离状态（state)改成3（转入医院）。

1. 编程要求：

用create trigger语句创建符合任务要求的触发器（触发器名称自已命名）：

当隔离表（isolation\_record）中的某位隔离人员在诊断表（diagnose\_record）中的诊断结果（result）为1(新冠确诊)”时，自动将隔离表中的隔离状态（state)改成3（转入医院）。

### 1.2.5 创建并使用自己创建的函数

**第1关 创建函数并在语句中使用它**

1. 任务描述：
2. 编写一个依据人员编号计算其到达所有地点的次数(即行程表中的记录数)的自定义函数，同一人员到达同一地点多次，去几次算几次。
3. 并利用其查询至少有3条行程记录的人员。
4. 编程要求：
5. 用create function语句创建符合以下要求的函数：

·依据人员编号计算其到达所有地点的次数(即行程表中的记录数)。

·函数名为：Count\_Records。函数的参数名可以自己命名

1. 利用创建的函数，仅用一条SQL语句查询在行程表中至少有3条行程记录的人员信息，查询结果依人员编号排序。

# 2.软件功能学习

**实验环境**

操作系统：Windows 10 家庭中文版

数据库：MySQL

交互图形界面：Navicat for MySQL 15.0.23

## 2.1 备份

### 2.1.1 数据导入

按照实验要求建立相关的数据库和表。

下载群文件夹中的 MySQLl实训数据批量插入.sql文件，添加到Navicat的查询列表中，执行。

得到如图 2-1 插入数据结果：

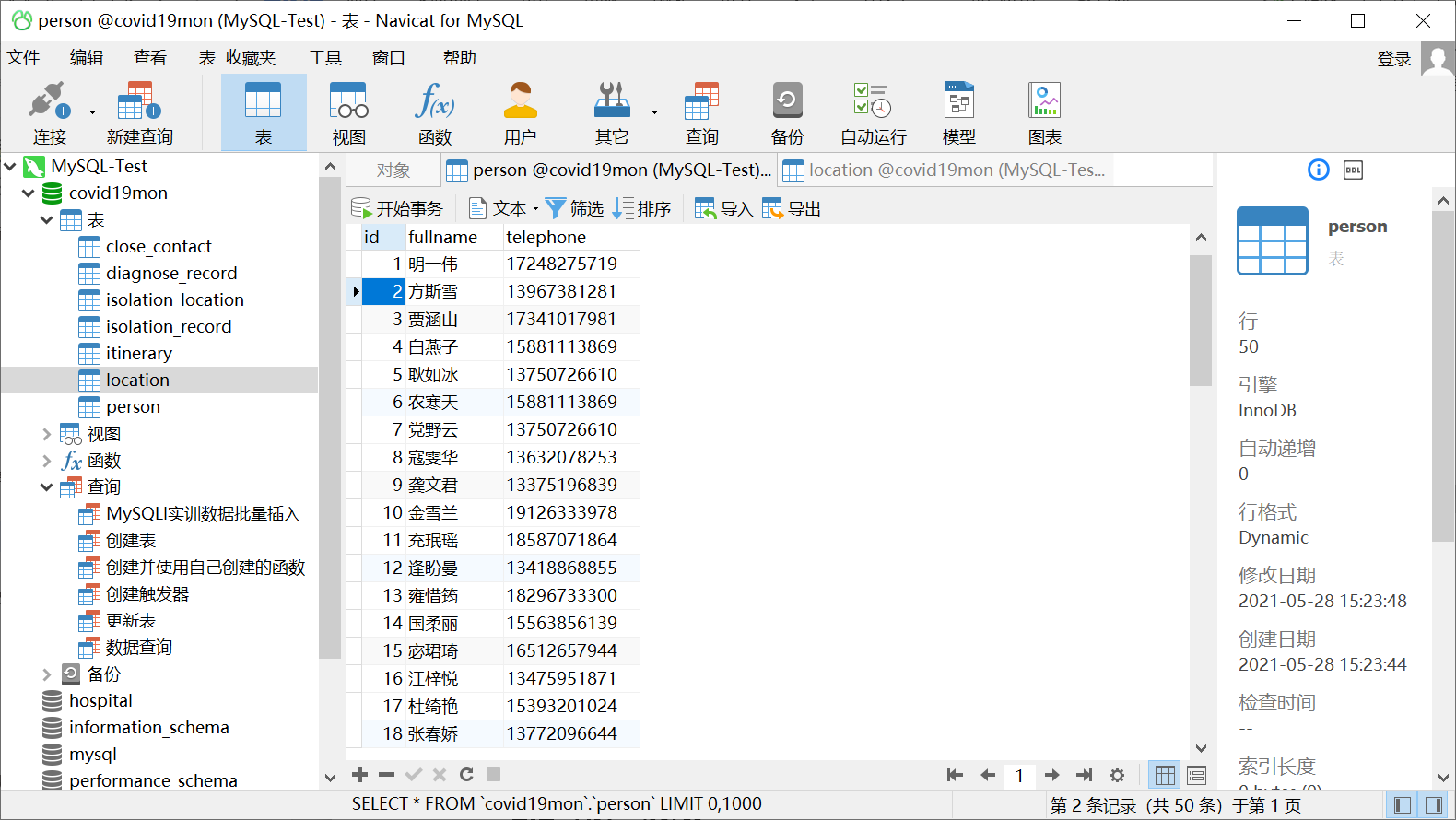


图 2-1 插入数据结果

### 2.1.2 热备份

在Navicat界面中新建备份，并选中需要备份的表、视图、函数和事件，我选择了备份所有的文件。然后会得到一个.nb3文件。

备份后得到如图 2-2 热备份结果：

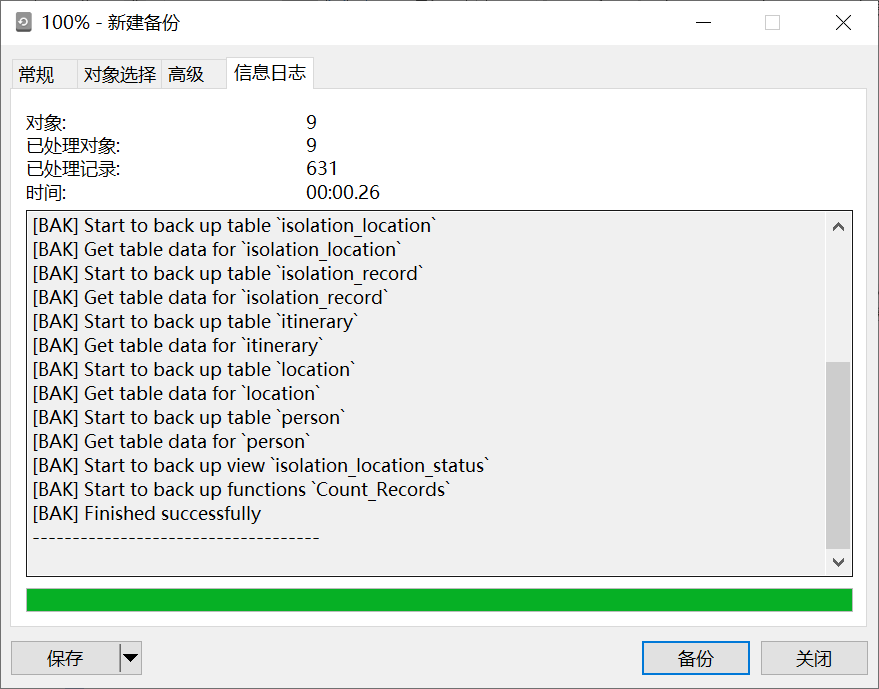


图 2-2 热备份结果

### 2.1.3 冷备份

直接从MySQL目录下复制所要备份的文件，这种备份方式不需要登陆数据库。

图 2-3 MySQL数据文件路径：

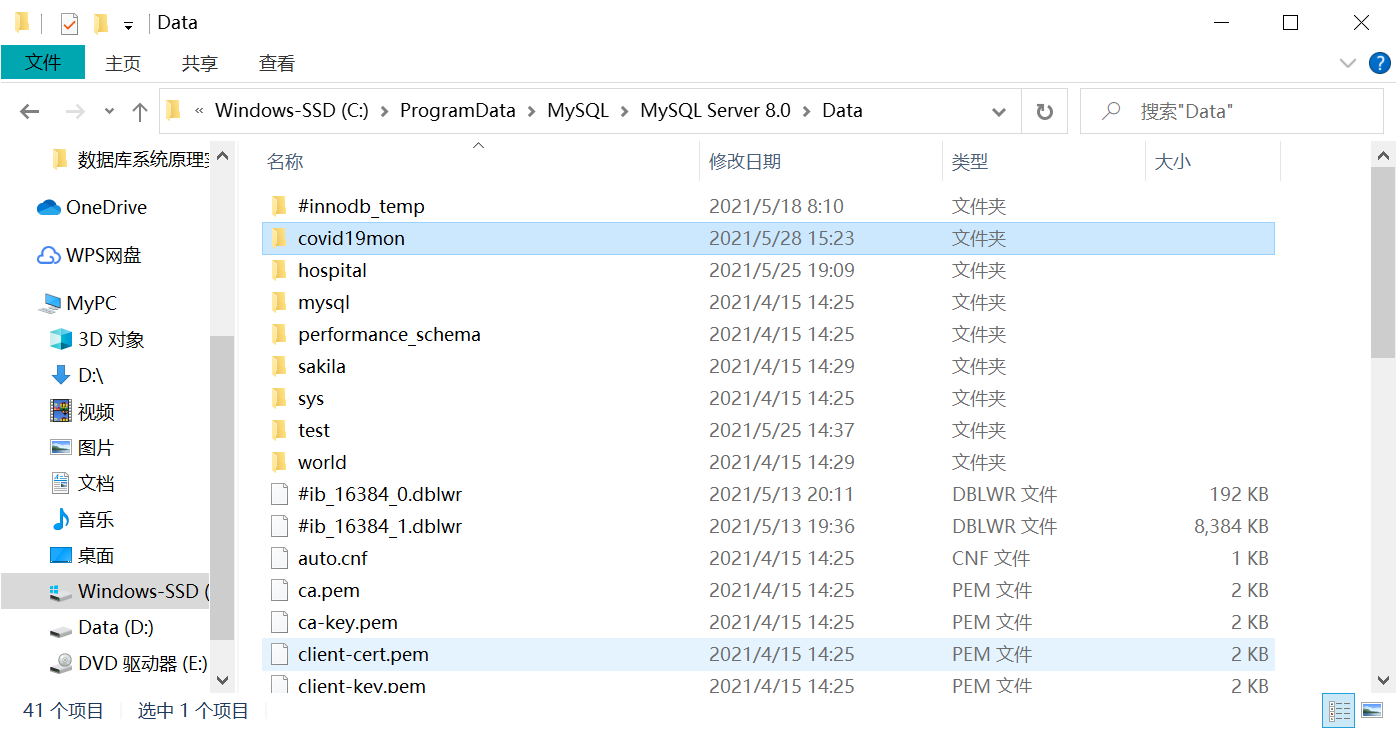


图 2-3 MySQL数据文件路径

## 2.2 用户权限

在Navicat界面中新建用户，并选择用户的操作权限。

得到如图 2-4 新建用户信息和图 2-5 新建用户权限。

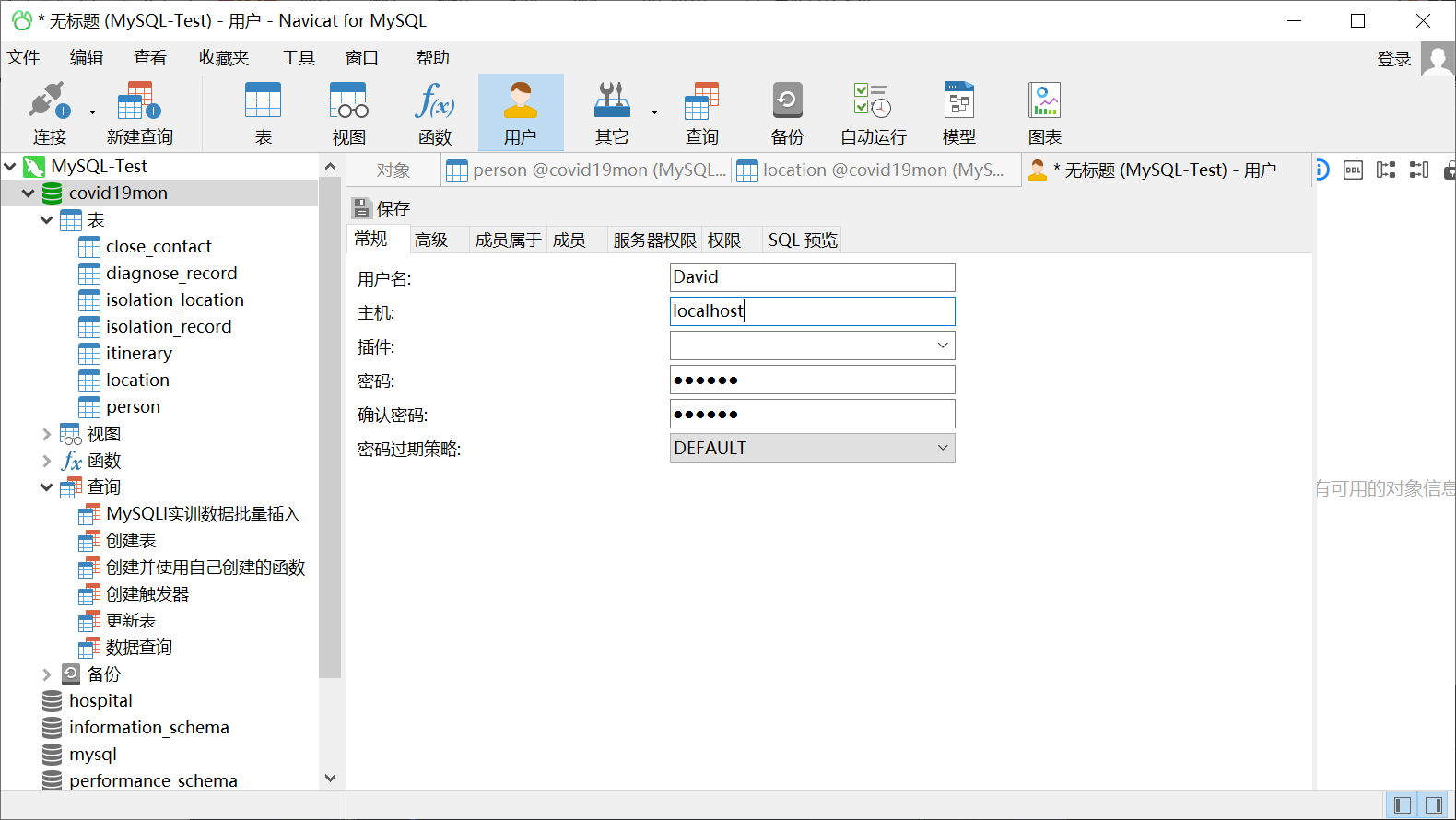


图 2-4 新建用户信息

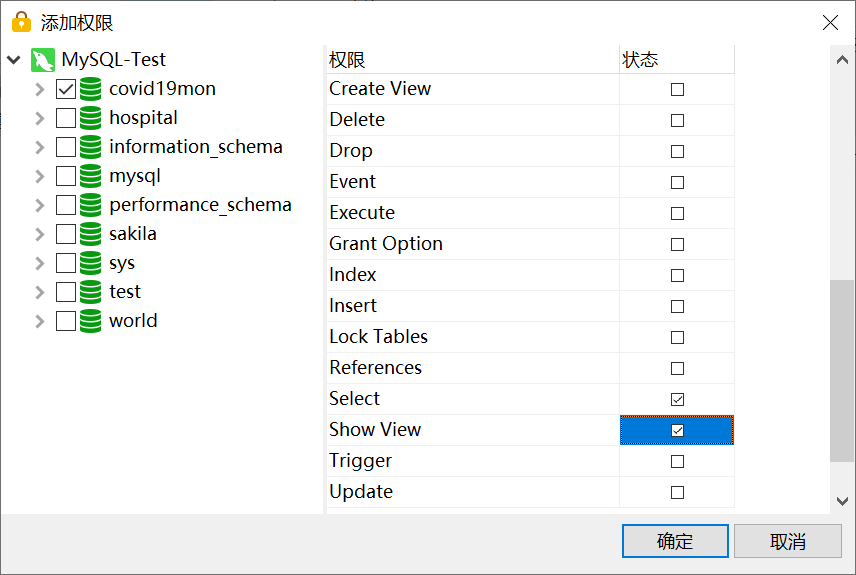


图 2-5 新建用户权限

由以上可以看出，所新建用户的权限只限于查询操作，通过用查询、插入、删除指令，验证权限的正确性。

如图 2-6 删除表操作所示，用户David没有删除表的权限：

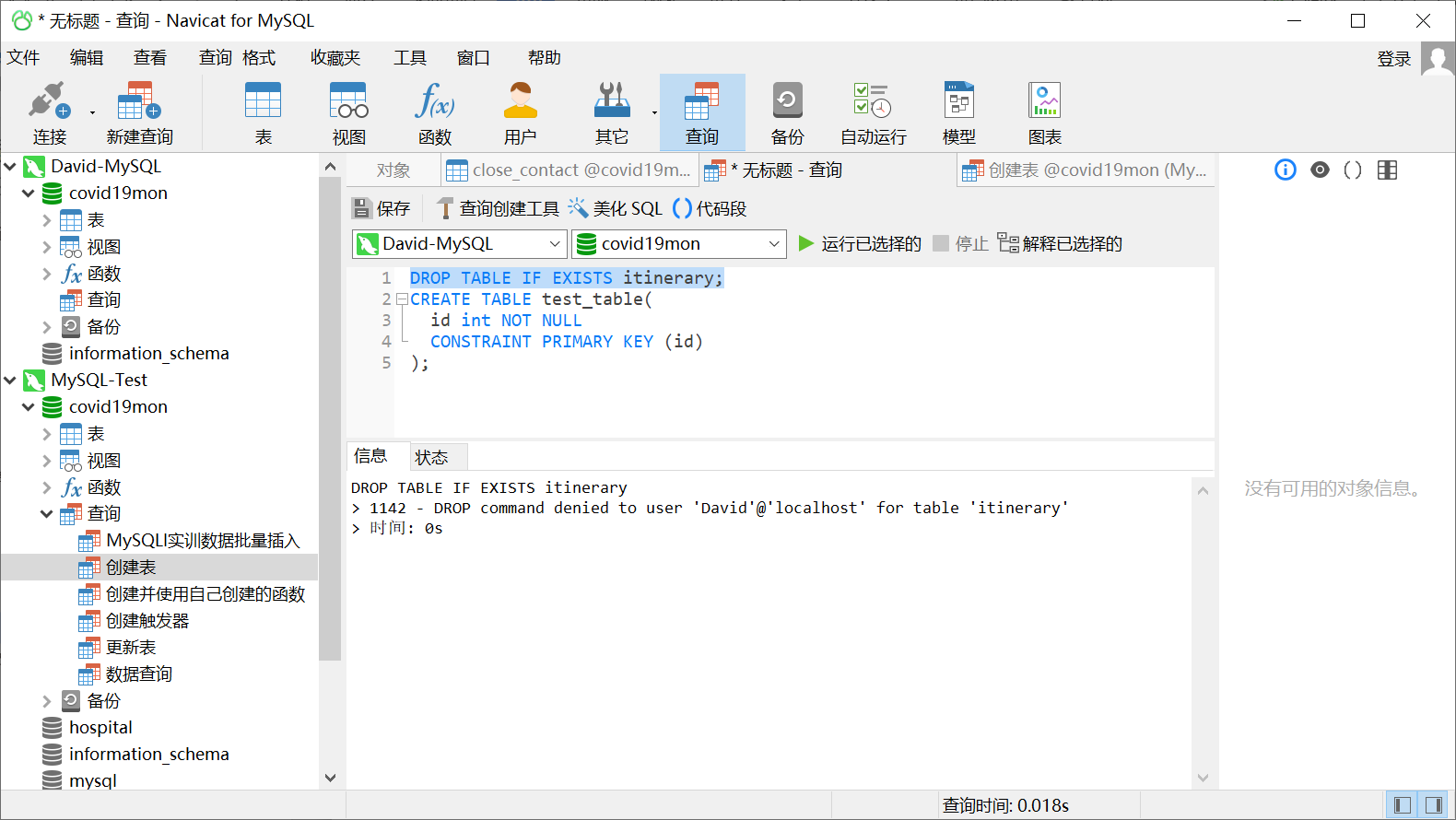


图 2-6 删除表操作

如图 2-7 新建表操作所示：用户David没有新建表的权限：

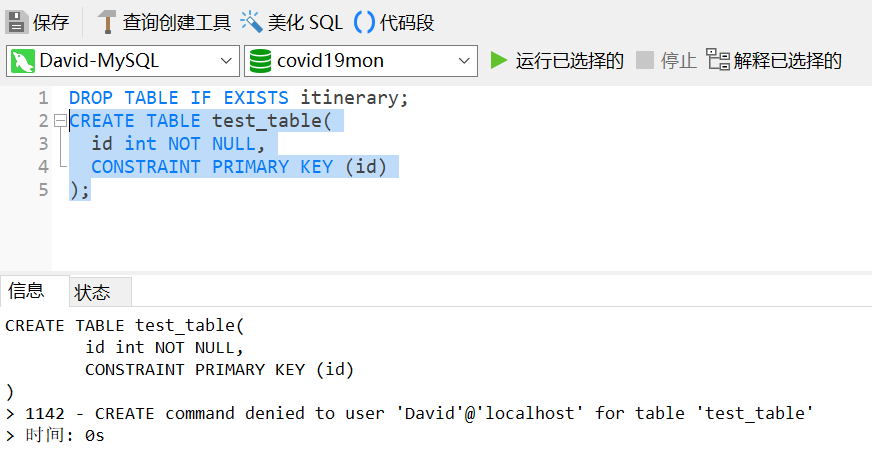


图 2-7 新建表操作

如图 2-8 查询操作所示，用户David可以对表进行查询操作：

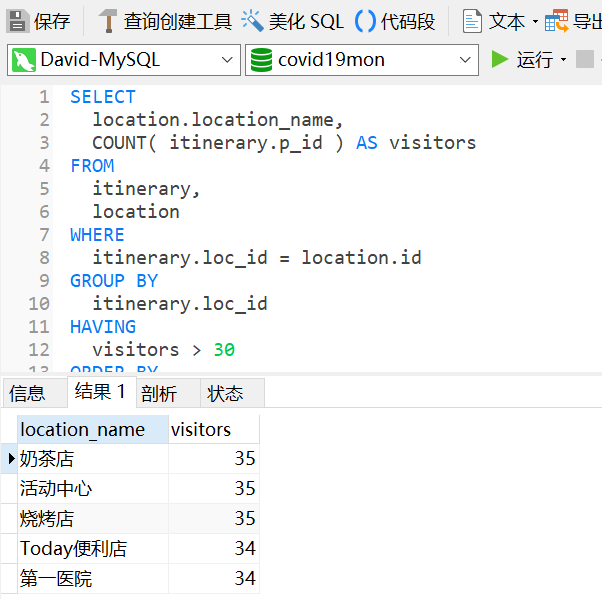


图 2-8 查询操作

# 3.SQL练习

## 3.1 创建数据库和表

### **第1关 创建用于新冠疫情常态防控的数据库**

create database covid19mon;

### 第2关 创建表和主外码约束

-- 表1 人员表(person)

CREATE TABLE person(

  id int NOT NULL,

  fullname char(20) NOT NULL,

  telephone char(11) NOT NULL,

  CONSTRAINT pk\_person PRIMARY KEY (id)

);

-- 表2 地点表(location)

CREATE TABLE location(

    id int NOT NULL,

    location\_name char(20) NOT NULL,

    CONSTRAINT pk\_location PRIMARY KEY (id)

);

-- 表3 行程表（itinerary）

CREATE TABLE itinerary(

    id int NOT NULL,

    p\_id int NULL,

    loc\_id int NULL,

    s\_time datetime NULL,

    e\_time datetime NULL,

    CONSTRAINT pk\_itinerary PRIMARY KEY (id),

    CONSTRAINT fk\_itinerary\_pid FOREIGN KEY (p\_id) REFERENCES person (id),

    CONSTRAINT fk\_itinerary\_lid FOREIGN KEY (loc\_id) REFERENCES location (id)

);

-- 表4 诊断表（diagnose\_record）

CREATE TABLE diagnose\_record(

    id int NOT NULL,

    p\_id int NULL,

    diagnose\_date datetime NULL,

    result int NULL,

    CONSTRAINT pk\_diagnose\_record PRIMARY KEY (id),

    CONSTRAINT fk\_diagnose\_pid FOREIGN KEY (p\_id) REFERENCES person (id)

);

-- 表5 密切接触者表（close\_contact）

CREATE TABLE close\_contact(

    id int NOT NULL,

    p\_id int NULL,

    contact\_date datetime NULL,

    loc\_id int NULL,

    case\_p\_id int NULL,

    CONSTRAINT pk\_close\_contact PRIMARY KEY (id),

    CONSTRAINT fk\_contact\_pid FOREIGN KEY (p\_id) REFERENCES person (id),

    CONSTRAINT fk\_contact\_lid FOREIGN KEY (loc\_id) REFERENCES location (id),

    CONSTRAINT fk\_contact\_caseid FOREIGN KEY (case\_p\_id) REFERENCES person (id)

);

-- 表6 隔离地点表（isolation\_location）

CREATE TABLE isolation\_location(

    id int NOT NULL,

    location\_name char(20) NULL,

    capacity int NULL,

    CONSTRAINT pk\_isolation\_loc PRIMARY KEY (id)

);

-- 表7 隔离表（isolation\_record）

CREATE TABLE isolation\_record(

    id int NOT NULL,

    p\_id int NULL,

    s\_date datetime NULL,

    e\_date datetime NULL,

    isol\_loc\_id int NULL,

    state int NULL,

    CONSTRAINT pk\_isolation PRIMARY KEY (id),

    CONSTRAINT fk\_isolation\_pid FOREIGN KEY (p\_id) REFERENCES person (id),

    CONSTRAINT fk\_isolation\_lid FOREIGN KEY (isol\_loc\_id) REFERENCES isolation\_location (id)

);

## 3.2 表的更新

### 第1关 数据插入

INSERT INTO person (id, fullname, telephone) VALUES (1, '张小敏', '13907110001');

INSERT INTO person (id, fullname, telephone) VALUES (2, '李大锤', '18907110002');

INSERT INTO person (id, fullname, telephone) VALUES (3, '孙二娘', '13307100003');

### 第2关 数据删除

DELETE FROM person WHERE id=2;

### 第3关 修改数据

UPDATE person SET telephone='13607176668' WHERE id=1;

## 3.3 数据查询

### 第1关 人流量大于30的地点

SELECT

    location.location\_name,

    COUNT( itinerary.p\_id ) AS visitors

FROM

    itinerary,

    location

WHERE

    itinerary.loc\_id = location.id

GROUP BY

    itinerary.loc\_id

HAVING

    visitors > 30

ORDER BY

    visitors DESC,

    location.location\_name ASC;

### 第2关 每个隔离点正在进行隔离的人数

SELECT

    isolation\_location.location\_name,

    COUNT( isolation\_record.p\_id ) AS number

FROM

    isolation\_record,

    isolation\_location

WHERE

    isolation\_record.isol\_loc\_id = isolation\_location.id

    AND isolation\_record.state = 1

GROUP BY

    isolation\_location.id

ORDER BY

    number DESC,

    isolation\_location.location\_name ASC;

### 第3关 接续行程

SELECT

    person.id,

    person.fullname,

    person.telephone,

    start\_itinerary.e\_time AS reclosing\_time,

    start\_location.id AS loc1,

    start\_location.location\_name AS address1,

    end\_location.id AS loc2,

    end\_location.location\_name AS address2

FROM

    person,

    itinerary AS start\_itinerary,

    itinerary AS end\_itinerary,

    location AS start\_location,

    location AS end\_location

WHERE

    start\_itinerary.p\_id > 30

    AND start\_itinerary.p\_id = person.id

    AND end\_itinerary.p\_id = person.id

    AND start\_itinerary.loc\_id = start\_location.id

    AND end\_itinerary.loc\_id = end\_location.id

    AND start\_itinerary.e\_time = end\_itinerary.s\_time

ORDER BY

    person.id,

    reclosing\_time;

### 第4关 充珉瑶和贾涵山的行程情况

SELECT

    person.fullname,

    person.telephone,

    location.location\_name,

    itinerary.s\_time,

    itinerary.e\_time

FROM

    person

    LEFT JOIN itinerary ON itinerary.p\_id = person.id

    LEFT JOIN location ON itinerary.loc\_id = location.id

WHERE

    person.fullname = '充珉瑶'

    OR person.fullname = '贾涵山'

ORDER BY

    person.id DESC,

    itinerary.s\_time;

### 第5关 地名中带有‘店’字的地点名称

SELECT

    location.id,

    location.location\_name

FROM

    location

WHERE

    location.location\_name LIKE '%店%';

### 第6关 确诊者的接触者

SELECT

    person.fullname,

    person.telephone

FROM

    person

    JOIN itinerary ON person.id = itinerary.p\_id

    JOIN location ON itinerary.loc\_id = location.id

WHERE

    location.location\_name = '活动中心'

    AND itinerary.e\_time > '2021-02-02 20:05:40'

    AND itinerary.s\_time < '2021-02-02 21:25:40'

ORDER BY

    person.fullname;

### 第7关 仍在使用的隔离点

SELECT DISTINCT

    isolation\_location.location\_name

FROM

    isolation\_location

WHERE

    isolation\_location.id IN ( SELECT isolation\_record.isol\_loc\_id FROM isolation\_record WHERE isolation\_record.state = 1 );

### 第8关 查询有出行记录的人员

SELECT

    person.fullname,

    person.telephone

FROM

    person

WHERE

    EXISTS ( SELECT DISTINCT itinerary.p\_id FROM itinerary WHERE person.id = itinerary.p\_id ORDER BY itinerary.p\_id )

    LIMIT 0,

    30;

### 第9关 没有去过“Today便利店“的人数

SELECT

    COUNT( person.id ) AS number

FROM

    person

WHERE

    NOT EXISTS (

    SELECT DISTINCT

        itinerary.p\_id

    FROM

        itinerary

        JOIN location ON itinerary.loc\_id = location.id

    WHERE

        itinerary.p\_id = person.id

        AND location.location\_name = 'Today便利店'

    );

### 第10关 去过所有地点的人员

SELECT

    person.fullname

FROM

    person

WHERE

    NOT EXISTS (

    SELECT

        \*

    FROM

        location

    WHERE

    NOT EXISTS ( SELECT \* FROM itinerary WHERE itinerary.p\_id = person.id AND itinerary.loc\_id = location.id ))

ORDER BY

    person.fullname;

### 第11关 隔离点的现状视图

CREATE VIEW isolation\_location\_status AS SELECT

isolation\_location.id,

isolation\_location.location\_name,

isolation\_location.capacity,

COUNT( isolation\_record.state = 1 OR NULL ) AS occupied

FROM

    isolation\_location

    INNER JOIN isolation\_record ON isolation\_location.id = isolation\_record.isol\_loc\_id

GROUP BY

    isolation\_location.id;

### 第12关 各隔离点的剩余房间数

SELECT

    isolation\_location\_status.location\_name,

    ( isolation\_location\_status.capacity - isolation\_location\_status.occupied ) AS available\_rooms

FROM

    isolation\_location\_status

ORDER BY

    isolation\_location\_status.id;

### 第13关 与无症状感染者靳宛儿有过接触的人

SELECT

    person.fullname,

    person.telephone

FROM

    person

    JOIN itinerary ON person.id = itinerary.p\_id,

    (

    SELECT

        itinerary.loc\_id,

        itinerary.s\_time,

        itinerary.e\_time

    FROM

        person

        JOIN itinerary ON person.id = itinerary.p\_id

    WHERE

        person.fullname = '靳宛儿'

    ) AS temp\_table

WHERE

    person.fullname != '靳宛儿'

    AND itinerary.loc\_id = temp\_table.loc\_id

    AND itinerary.s\_time <= temp\_table.e\_time AND itinerary.e\_time >= temp\_table.s\_time

ORDER BY

    person.fullname;

### 第14关 每个地点发生的密切接触者人数

SELECT

    location.location\_name,

    COUNT( close\_contact.p\_id ) AS close\_contact\_number

FROM

    close\_contact

    JOIN location ON close\_contact.loc\_id = location.id

GROUP BY

    close\_contact.loc\_id

ORDER BY

    close\_contact\_number DESC,

    location.location\_name;

### 第15关 感染人数最多的人

SELECT

    close\_contact.case\_p\_id,

    person.fullname,

    COUNT( close\_contact.p\_id ) AS infected\_number

FROM

    close\_contact

    JOIN person ON close\_contact.case\_p\_id = person.id

GROUP BY

    close\_contact.case\_p\_id

ORDER BY

    infected\_number DESC

    LIMIT 0,

    1;

### 第16关 行程记录最频繁的3个人

SELECT

    person.fullname,

    COUNT( itinerary.id ) AS record\_number

FROM

    person

    JOIN itinerary ON person.id = itinerary.p\_id

WHERE

    itinerary.e\_time >= '2021-02-02 10:00:00'

    AND itinerary.s\_time <= '2021-02-02 14:00:00'

GROUP BY

    person.id

ORDER BY

    record\_number DESC,

    person.fullname

    LIMIT 0,

    3;

### 第17关 房间数第2多的隔离点

SELECT

    isolation\_location.location\_name,

    isolation\_location.capacity

FROM

    isolation\_location

WHERE

    isolation\_location.capacity <( SELECT MAX( isolation\_location.capacity ) FROM isolation\_location )

LIMIT 0,1;

## 3.4 创建触发器

### 第1关 隔离点的人员“确诊新冠”后，自动转院

DROP TRIGGER IF EXISTS state\_change1;

DROP TRIGGER IF EXISTS state\_change2;

DELIMITER ||

CREATE TRIGGER state\_change1 AFTER UPDATE ON diagnose\_record FOR EACH ROW

BEGIN

    IF

        NEW.result = 1

        THEN

            UPDATE isolation\_record

            SET isolation\_record.state = 3

            WHERE

                isolation\_record.p\_id = NEW.p\_id;

    END IF;

END||

CREATE TRIGGER state\_change2 AFTER INSERT ON diagnose\_record FOR EACH ROW

BEGIN

    IF

        NEW.result = 1

        THEN

            UPDATE isolation\_record

            SET isolation\_record.state = 3

            WHERE

                isolation\_record.p\_id = NEW.p\_id;

    END IF;

END||

DELIMITER;

## 3.5 创建并使用自己创建的函数

### 第1关 创建函数并在语句中使用它

set global log\_bin\_trust\_function\_creators=1;

DELIMITER ||

DROP FUNCTION IF EXISTS Count\_Records;

CREATE FUNCTION Count\_Records(person\_id int)

    RETURNS int

BEGIN

    #Routine body goes here...

    RETURN(

        SELECT

            COUNT(itinerary.id)

        FROM

            itinerary

        WHERE

            itinerary.p\_id=person\_id);

END||

DELIMITER ;

SELECT

    \*

FROM

    person

WHERE

    Count\_Records ( person.id )>= 3

ORDER BY

    person.id;

# 4.应用系统设计

## 4.1系统设计目标

## 4.2 需求分析

## 4.3 总体设计

## 4.4 数据库设计

## 4.5 详细设计与实现

## 4.6 系统测试

## 4.7 系统设计与实现总结

阐述第3部分的完成过程。

# 5.课程总结

逐条概括、总结此次课程实践的主要工作，阐述此次课程实践的心得体会，展望此次课程实践的有待改进和完善的工作。