**项目说明**

1. **数据库表结构说明:**

本次PJ我们小组的思路是分别给8个实体集（四种不同身份，病人，位置，日常信息以及核酸检测单）分别创建表结构，此外无任何关系实体集拥有表结构。具体表结构如下：

主治医生（doctor），护士长（head\_nurse），病房护士（ward\_nurse）与急诊护士(emergency\_nurse)表结构相似，其主键分别为d\_id, hn\_id, wn\_id, en\_id其余还有name, contact\_info, password属性，其中主键默认为登陆用户名，password为登陆密码

Patient 表主键为p\_id, 其余属性有name, contact\_info, age, severity（病情）, life\_state（生命状态）, en\_id, 其中因为病人与急诊护士为多对一的关系，所以将en\_id放在病人实体集中作为foreign key

Location实体集为我们此次PJ的核心，主键为（area, room\_no, bed\_no），其中area代表区域信息（0:隔离区，1:轻症，2:重症，3:危重症），剩余两个分别代表该区域房间号和床号，故三者同时就可以担任主键确定唯一的一个位置，因为主治医生，护士长，病房护士都与其有着一对多的关系，而病人和location拥有一对一的关系，所以location分别拥有他们四个实体集的id作为foreign key，注意当某床位无病人时，该location的主治医生和护士长信息依旧存在因为他们管辖这个床位对应的区域的所有location（通过确定一个location可以确定负责这个location的这四类人群，但当此location无病人时，病人和病房护士的id为NULL）

Daily\_info表包含病房护士每日给患者检测的日常信息，其主键为info\_id，其余还有the\_date, temperature, symptom, result（最近一次核酸检测结果）,life\_state, 并拥有p\_id 和wn\_id作为foreign key

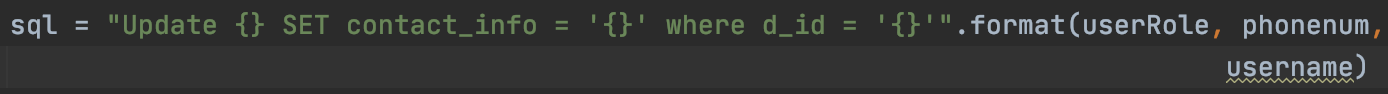
最后一个表结构为covid\_test, 主键是t\_id, 另有the\_date, result, severity并且p\_id 和d\_id 为foreign key.

**2. 关键sql语句：**

我们小组大多查询和增删改操作依靠于不同用户凭借登录时输入的id（d\_id, hn\_id, wn\_id, en\_id）来和location表进行连接。

其中重点操作有：

更改个人信息/密码同理



医生查看自己区域护士长信息/护士同理

sql = "SELECT \* FROM head\_nurse as hn WHERE hn.hn\_id = (SELECT distinct hn\_id FROM location as l WHERE (l.d\_id = '%s'))" % username

日期降序选择某一病人的核酸检测单/日常信息单同理

sql1 = "SELECT temperature FROM daily\_info WHERE daily\_info.p\_id = %s ORDER BY the\_date DESC" % p\_id[0]

更改病人状态

sql = "UPDATE patient as p SET p.life\_state = '康复出院' WHERE p\_id = '%s'" % (p\_id)

查找身在隔离区，症状为轻症/重症/危重症病人信息

sql3 = "SELECT \* FROM patient WHERE patient.p\_id = some(SELECT p\_id FROM location WHERE location.area = 0) and patient.severity = '轻症'"

转移病人时先清空原来location信息

sql5 = "UPDATE location as l SET l.p\_id = NULL WHERE l.p\_id = '%s'" % i[0]

再进行更换操作

sql4 = "UPDATE location as l SET l.p\_id = '%s' WHERE l.p\_id = '%s'" % (i[0], p\_id)

增加核酸检测单/日常信息同理

sql = "INSERT into covid\_test values ('{}', '{}',{}, '{}','{}', '{}')".format(tid, date, result, severity, p\_id,  
 username)

检验执行康复操作的医生id，对比location里轻症区域的d\_id，看是否为轻症区域医生

sql3 = "SELECT distinct area FROM location WHERE location.d\_id = '%s'" % username

检查能有条件接收新 轻症病房病人的护士名单

"(SELECT wn\_id FROM ward\_nurse WHERE wn\_id NOT IN ((SELECT wn\_id FROM location l WHERE l.area = 1 and wn\_id is not null) UNION (SELECT wn\_id FROM location l WHERE l.area = 2 and wn\_id is not null) UNION (SELECT wn\_id FROM location l WHERE l.area = 3 and wn\_id is not null) )) UNION (SELECT h.wn\_id FROM (SELECT l.wn\_id, COUNT(\*) as job FROM location as l WHERE l.area = 1 GROUP BY l.wn\_id) as h WHERE h.job < 4)"

注：此语句较为复杂，我们组对于护士的操作较为特殊，我们没有为每个区域单独设立一个护士库（在护士的表格中没有区域的id作为foreign key），而是将所有护士视作一个总库，只有某个location有病人的时候才同时有护士，病人转走时护士id一并从location里移走。此sql语句用了两个union来选出了目前在三个医院区域已有负责病人的护士名单，接着用总名单减掉这些已有负责病人的护士名单（通过NOT In），最后我们又union了此区域（本例子为轻症区域）虽然有负责的病人，但未超过上限的护士，形成一张可以被选择的护士总表，如果表为空，则不能进行加入新病人操作，如果不为空则可以从表中选出一名护士与新到患者id一起更新到location信息里

3. 运行及其他说明

Python为我们使用的语言

请先在terminal运行mysql -u root -p < init.sql

然后 python DB.py运行 （请先更改DB.py程序中的connect端口密码）

登陆界面可选择不同用户身份登陆例如165690，143690，109823分别为轻症，重症，危重症区域负责主治医生的用户名，所有用户初始密码皆为000000

我们主要用的是默认primary indexing,因为每个id具有唯一性，且数据量低，可通过主键索引同样快速查询，同时我们所有自动操作（例如在原本位置全满的区域中，医生将病人状态更改成康复出院或者死亡时，会自动有隔离区或者其他区域符合条件病人转移到之前病人的location并且匹配相应的护士，又比如在考虑新病人时会考虑有无空余床位与护士）都在程序里用python语句写成，无涉及到触发器操作。