**Project Description**

1. Description of database table structure

In this PJ, our team's idea is to create table structures for 8 entity sets (four different identities, patients, locations, daily information and nucleic acid test sheets) respectively. In addition, no related entity set has a table structure. The specific table structure is as follows:

Attending doctor, head nurse\_ Nurse, ward\_ Nurse and emergency\_ The table structure is similar, and its primary keys are d\_ id, hn\_ id, wn\_ id, en\_ ID, name, contact\_ Info, password attribute, where the primary key is the login user name by default and password is the login password

The primary key of patient table is p\_ The other attributes are name and contact\_ Info, age, severity, life\_ State, EN\_ Because of the many to one relationship between the patient and the emergency nurse, En will be used\_ The ID is placed in the patient entity set as a foreign key

Location entity set is the core of our PJ, and the primary key is (area, room)\_ no, bed\_ The area represents the area information (0: isolation area, 1: mild disease, 2: severe disease, 3: critical disease), and the remaining two represent the room number and bed number of the area respectively, so the three can act as the primary key to determine the only location at the same time, because the attending doctor, head nurse and ward nurse all have a one-to-many relationship with them, while the patient and location have a one-to-one relationship, so LOC Each of the four entity sets has its ID as foreign Key, note that when a bed has no patients, the information of the attending doctor and head nurse of the location still exists, because they are in charge of all the locations of the corresponding area of the bed (by determining a location, the four groups of people responsible for the location can be determined, but when the location has no patients, the ID of the patient and ward nurse is null)

Daily\_ The info table contains the daily information of the ward nurses to the patients, and its primary key is info\_ ID, and the rest\_ Date, temperature, symptom, result, life\_ State, and has P\_ ID and WN\_ ID as foreign key

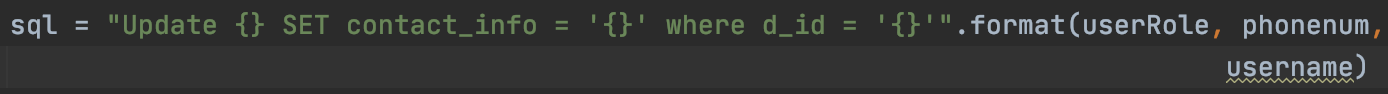
The last table structure is covid\_ Test, the primary key is t\_ ID, and the\_ Date, result, severity and P\_ ID and D\_ The ID is foreign key

2. Key SQL statements:

**2. Key SQL statements:**

**Most of our group's queries, additions, deletions and modifications depend on the ID (d) entered by different users when they log in\_ id, hn\_ id, wn\_ id, en\_ ID) to connect to the location table.**

**The key operations are as follows:**

**Changing personal information / password is the same**

Doctors check information of head nurse in their own area

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

Select a patient's nucleic acid test sheet / daily information sheet in descending order of date

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

Change patients‘ life state

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

Find out the information of patients with mild / severe / critical symptoms in isolation area

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

Clear the original location information before transferring patients

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

Replace it again

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

Add nucleic acid test sheet / daily information

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

Check the ID of the doctor who performed the rehabilitation operation, and compare the D of the mild area in the location\_ ID to see if he is a regional doctor with mild disease

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

Check the list of nurses who can accept the patients in the new mild ward

"(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)"

Note: this sentence is more complex. Our group has a special operation for nurses. Instead of setting up a nurse database for each area (there is no regional ID as the foreign key in the nurse table), we regard all nurses as a general database. Only when there is a patient in a certain location can there be a nurse at the same time. When the patient is transferred, the nurse ID will be removed from the location together. This SQL statement uses two unions to select the list of nurses who are in charge of patients in the three hospital areas, and then subtracts the list of nurses who are in charge of patients from the total list (through not Finally, we unionized this area (in this case, mild area) to form a general table of nurses that can be selected. If the table is empty, you cannot add new patients. If it is not empty, you can select a nurse from the table and update it to the location information together with the new patient ID

3. Operation and other instructions

Python is the language we use

Please run MySQL - U root - P in terminal first< init.sql

And then python DB.py Run (please change first) DB.py Connect port password in the program)

Different user identities can be selected to login in the login interface. For example, 165690143690109823 is the user name of the attending doctor in the mild, severe and critical areas, and the initial password of all users is 000000

We mainly use the default primary indexing, Because each ID is unique and the amount of data is low, it can also be queried quickly through the primary key index. At the same time, all our automatic operations (for example, when the doctor changes the patient's status to rehabilitation, discharge or death in the original full area, there will be an isolation area or other areas automatically, and the qualified patients will be transferred to the previous patient's location and matched with the corresponding nurses For example, when considering new patients, we will consider whether there are spare beds and nurses. They are all written in Python statements in the program, and there is no trigger operation involved.