1. MySQL

查看MySQL版本信息：

$ mysql --version

mysql Ver 14.14 Distrib 5.7.25, for Win64 (x86\_64)

进入MySQL shell查看mysql状态：

$ mysql -u root -p

$ 123456

mysql>

查看数据库列表：

show databases;

information\_schema/default/mysql/performance\_schema/sys

PS: MySQL命令必须以分号结尾;

使用database\_name数据库

use database\_name;

查看数据库表格列表：

show tables;

2. 安装Python第三方库

pip install pymysql

pip install torndb

3. 创建数据库

CREATE DATABASE CDATS;

4. 授权数据库用户

GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER ON CDATS.\* TO CDATS @localhost IDENTIFIED BY '123456';

5. 修改数据库用户密码

SET PASSWORD FOR CDATS@'localhost' = PASSWORD('123456');

6. pymysql

import pymysql

config = {

'host': 'localhost',

'port': 3306,

'user': 'root',

'password': '123456',

'db': 'CDATS',

'charset': 'gbk',

'cursorclass': pymysql.cursors.DictCursor,

}

str = {

'patient': [{

'patient\_ID': '3768831',

'patient\_last\_name': '许',

'patient\_first\_name': '冬芳',

'patient\_birthday': '1977-11-12',

'patient\_sex': '女',

'patient\_picture\_url': '1768831.jpg',

'patient\_illustration': '无'

}],

'diagnose': [{

'patient\_ID': '3768831',

'doctor\_ID': '12580',

'doctor\_name': '马主任',

'patient\_disease': '细菌性角膜炎'

}, {

'patient\_ID': '3768831',

'doctor\_ID': '12180',

'doctor\_name': 'Q主任',

'patient\_disease': '细菌性角膜炎'

}, {

'patient\_ID': '3768831',

'doctor\_ID': '12080',

'doctor\_name': 'A主任',

'patient\_disease': '细菌性角膜炎'

}, {

'patient\_ID': '3768831',

'doctor\_ID': '12480',

'doctor\_name': '胡主任',

'patient\_disease': '细菌性角膜炎'

}],

'registration': [{

'registration\_ID': '2019102401',

'patient\_ID': '3768831',

'registration\_data': '2016-05-02',

'registration\_department': '眼科',

'registration\_job': '主任'

}],

'custom': [{

'patient\_ID': '3768831',

'patient\_custom': '眼睛很痛'

}, {

'patient\_ID': '3768831',

'patient\_custom': '眼睛炸了'

}, {

'patient\_ID': '3768831',

'patient\_custom': '眼睛很胀'

}]

}

class CreateDB(object):

def \_\_init\_\_(self, config):

self.config = config

self.conn = pymysql.connect(\*\*self.config)

self.cursor = self.conn.cursor()

self.sql = "set autocommit = 0;"

self.cursor.execute(self.sql)

def close\_mysql(self):

self.conn.commit()

self.cursor.close()

self.conn.close()

def ExecuteDB(self, sql):

effect\_row = None

result = None

try:

effect\_row = self.cursor.execute(sql)

result = self.cursor.fetchall()

except:

self.conn.rollback()

finally:

self.conn.commit()

self.sql = sql

self.effect\_row = effect\_row

self.result = result

return effect\_row, result

def table\_sql(self):

table\_dict = {}

foreign\_dict = {}

sql = """

create table if not exists patient(

patient\_ID varchar(20) PRIMARY KEY,

patient\_last\_name varchar(20),

patient\_first\_name varchar(20),

patient\_birthday varchar(10),

patient\_sex varchar(10),

patient\_picture\_url varchar(100),

patient\_illustration varchar(100)

)character set gbk;

"""

self.ExecuteDB(sql)

table\_dict.update(patient=sql)

sql = """

create table if not exists diagnose(

patient\_ID varchar(20),

doctor\_ID varchar(20),

doctor\_name varchar(20),

patient\_disease varchar(20),

primary key(patient\_ID,doctor\_ID)

)character set gbk;

"""

self.ExecuteDB(sql)

table\_dict.update(diagnose=sql)

sql = """

create table if not exists registration(

registration\_ID varchar(20) PRIMARY KEY,

patient\_ID varchar(20),

registration\_data varchar(20),

registration\_department varchar(20),

registration\_job varchar(20)

)character set gbk;

"""

self.ExecuteDB(sql)

table\_dict.update(registration=sql)

sql = """

create table if not exists custom(

patient\_ID varchar(20),

patient\_custom varchar(20),

primary key(patient\_ID,patient\_custom)

)character set gbk;

"""

self.ExecuteDB(sql)

table\_dict.update(custom=sql)

sql = """

alter table registration

add constraint registration\_patient

foreign key (patient\_ID)

references patient(patient\_ID) on delete cascade on update cascade;

"""

self.ExecuteDB(sql)

foreign\_dict.update(registration\_patient=sql)

sql = """

alter table diagnose

add constraint diagnose\_patient

foreign key (patient\_ID)

references patient(patient\_ID) on delete cascade on update cascade;

"""

self.ExecuteDB(sql)

foreign\_dict.update(diagnose\_patient=sql)

sql = """

alter table custom

add constraint custom\_patient

foreign key (patient\_ID)

references patient(patient\_ID) on delete cascade on update cascade;

"""

self.ExecuteDB(sql)

foreign\_dict.update(custom\_patient=sql)

self.table\_dict = table\_dict

self.foreign\_dict = foreign\_dict

return table\_dict, foreign\_dict

def drop\_table(self):

self.ExecuteDB("show tables;")

for table in self.result:

self.ExecuteDB("drop table if exists " + table['Tables\_in\_cdats'] + ";")

self.ExecuteDB("drop table if exists patient;")

def insert\_update(self,str):

for ss in str['patient']:

try:

sql = "insert into patient values('%s', '%s', '%s', '%s', '%s', '%s', '%s');" % \

(ss['patient\_ID'], ss['patient\_last\_name'], ss['patient\_first\_name'], ss['patient\_birthday'], \

ss['patient\_sex'], ss['patient\_picture\_url'], ss['patient\_illustration'])

# print(sql)

self.cursor.execute(sql)

except:

sql = "update patient set patient\_last\_name = '%s', patient\_first\_name = '%s', patient\_birthday = '%s', patient\_sex = '%s', patient\_picture\_url = '%s', patient\_illustration = '%s' where patient\_ID = '%s';" % \

(ss['patient\_last\_name'], ss['patient\_first\_name'], ss['patient\_birthday'], ss['patient\_sex'], \

ss['patient\_picture\_url'], ss['patient\_illustration'], ss['patient\_ID'])

# print(sql)

self.cursor.execute(sql)

finally:

self.conn.commit()

for ss in str['registration']:

try:

sql = "insert into registration values('%s', '%s', '%s', '%s', '%s');" % \

(ss['registration\_ID'], ss['patient\_ID'], ss['registration\_data'], ss['registration\_department'], ss['registration\_job'])

# print(sql)

self.cursor.execute(sql)

except:

sql = "update registration set registration\_ID = '%s', registration\_data = '%s', registration\_department = '%s', registration\_job = '%s' where patient\_ID = '%s'" % \

(ss['registration\_ID'], ss['registration\_data'], ss['registration\_department'], ss['registration\_job'], ss['patient\_ID'])

# print(sql)

self.cursor.execute(sql)

finally:

self.conn.commit()

for ss in str['diagnose']:

try:

sql = "insert into diagnose values('%s', '%s', '%s', '%s');" % \

(ss['patient\_ID'], ss['doctor\_ID'], ss['doctor\_name'], ss['patient\_disease'])

# print(sql)

self.cursor.execute(sql)

except:

sql = "update diagnose set patient\_disease = '%s', doctor\_name = '%s' where patient\_ID = '%s' and doctor\_ID = '%s';" % \

(ss['patient\_disease'], ss['doctor\_name'], ss['patient\_ID'], ss['doctor\_ID'])

# print(sql)

self.cursor.execute(sql)

finally:

self.conn.commit()

for ss in str['custom']:

sql = "insert into custom values('%s', '%s');" % (ss['patient\_ID'], ss['patient\_custom'])

# print(sql)

self.ExecuteDB(sql)

def insert(Database):

Database.ExecuteDB("insert into patient values('3768831','许','冬芳','1977-11-12','女','1768831.jpg','无');")

Database.ExecuteDB("select \* from patient;")

print(Database.result)

Database.ExecuteDB("insert into diagnose values('3768831','12580','马主任','细菌性角膜炎');")

Database.ExecuteDB("insert into diagnose values('3768831','12480','胡主任','细菌性角膜炎');")

Database.ExecuteDB("select \* from diagnose;")

print(Database.result)

Database.ExecuteDB("insert into registration values('2019102401','3768831','2016-05-02','眼科','主任');")

Database.ExecuteDB("select \* from registration;")

print(Database.result)

Database.ExecuteDB("insert into custom values('%s', '%s');" % ('3768831','眼睛痛啊'))

Database.ExecuteDB("insert into custom values('3768831','眼睛很胀');")

Database.ExecuteDB("select \* from custom;")

print(Database.result)

def run():

Database = CreateDB(config)

Database.table\_sql()

# Database.drop\_table()

# insert(Database)

Database.insert\_update(str)

Database.close\_mysql()

if \_\_name\_\_ == "\_\_main\_\_":

run()