بسمه تعالی

محمد نورمحمدی ۹۸۱۲۰۵۴

```python
import psycopg2
con1 = psycopg2.connect(
    host = "localhost",
    database = "bookProject",
    user = "postgres",
    password="12345678"
)
con2 = psycopg2.connect(
    host = "localhost",
    database = "second_db",
    user = "postgres",
    password="12345678"
)
con2.autocommit =True
con1.autocommit =True
cur = con1.cursor()
cur_2 = con2.cursor()
```

بنده از کتابخانه psycopg2 برای وصل شدن به دیتابیس استفاده کردم و ابتدا آن را import کردم و به دیتابیس مبدا و مقصد وصل شدم و به ترتیب برابر با con1و con2 قرار دادم.
سپس برای اینکه کوئزی های تغییرات ما در دیتابیس ثبت شود autocommit آن ها را برابر با true قرار دادم.
و از هر یک اتصال ها یک cursor ساختم.

```python
# get names of tables from first db
cur.execute("SELECT table_name FROM informat
temp = cur.fetchall()
table_name = []
for r in temp:
    table_name.append(r[0])
# get attribute for each table
attr = {}
for x in table_name:
    attr[x] = []
    cmd = "SELECT column_name FROM informati
    cur.execute(cmd.format(x))
    temp = cur.fetchall()
    for r in temp:
        attr[x].append(r[0])

# get foreign key list for tables
fk_list = {}
for x in table_name:
    fk_list[x]= []
    cmd = "SELECT distinct ccu.table_name AS
    cur.execute(cmd.format(x))
    temp = cur.fetchall()
    for r in temp:
        fk_list[x].append(r[0])
# get primary key list for tables
pk_list = {}
for x in table_name:
    pk_list[x]=[]
    cmd = "SELECT    pg_attribute.attname FR
    cur.execute(cmd.format(x))
    temp = cur.fetchall()
    for r in temp:
        pk_list[x].append(r[0])
```

در ادامه کد من با کوئری های اسم جدول های دیتابیس مبدا را در آوردم و داخل لیست table_name قرار دادم و سپس اتربیوت های هر حدول را اسخراج کردم و داخل دیکشنری attr قرار دادم که این دیکشنری اسم هر جدول را به لیستی از اتربیوت های آن جدول متصل میکند .
سپس با کوئری foreign key های هر جدول را استخراج کردم و در دیکشنری fk_list قرار دادم که این دیکشنری اسم هر جدول را به لیستی از جدول هایی که اتربیوتی دارد که به آن جدول foreign key دارد متصل میکند .
و سپس با کوئری primarykey های هر جدول را استخراج کردم و در دیکشنری pk_list قرار دادم.

```
# sort table name by dag
temp = []
while len(table_name)>0:
    root =""
    for x in table_name:
        if(len(fk_list[x]) == 0):
            root = x
            break
    temp.append(root)
    table_name.remove(root)
    for x in table_name:
        if( fk_list[x].count(root )> 0):
            fk_list[x].remove(root)
table_name = temp

```

در این بخش جدول ها را توپولوژی مرتب کردیم بر اساس foriegn key هایی که بین آن ها بود در fk_list هر جدول به لیستی از از جدول ها که به آن ها وابسته بود متصل بود و ما میدانیم که این اتصالات DAG هستند و در DAG همیشه یک راس هست که درجه ورودی آن صفر است که در اینجا میشود fk_list آن تهی باشد .
اگر ما آن جدولی که fk_list آن تهی است( root) را در ابتدا آرایه مرتب شده بگذاریم و ان را از fk_list بقیه حذف کنیم چیزی که باقی می ماند باز هم DAG خواهد ماند و استقرایی DAG حاصل را مرتب میکنیم.

```
# pk num in attribute list
pk_num = {}
for x in table_name:
    j = 0
    pk_num[x] = []
    for i in range(len(pk_list[x])) :
        while pk_list[x][i] != attr[x][j] :
            j+=1
        pk_num[x].append(j)
```

در این تیکه کد برای هر جدول این را حساب کردیم که هر primary key اتربیوت چندم از لیست اتربیوت های آن جدول قرار دارد. و این ها را لیست کردم به ترتیب و یک دیکشنری ساختم به نام pk_num و این لیست را به اسم جدول متصل کردم.

```
        pk_num[x].append(j)
for name in table_name:
    cmd = "select count (*) from {};"
    cur.execute(cmd.format(name))
    cnt = cur.fetchall()[0][0]
    for x in range(cnt) :
        cmd = "select * from {} offset {} limit 1 ;"
        cur.execute(cmd.format(name, x))
        row = cur.fetchall()
        row = row[0]
        cmd = "select * from {} where "
        for i in range(len(pk_list[name])):
            cmd += pk_list[name][i] + " = '" + str(row[pk_num[name][i]]) + "',"
        cmd = cmd[:len(cmd) - 1]
        cmd += ";"
        cur_2.execute(cmd.format(name))
        temp = cur_2.fetchall()
        # insert
        if (len(temp) ==0):
            cmd = "INSERT INTO {} VALUES ("
            for z in row :
                    cmd += "'"+str(z) +"',"
            cmd = cmd[:len(cmd)-1]
            cmd +=");"
            cur_2.execute(cmd.format(name))
        # update
        else:
            cmd = ""
            for z in range(len(row)):
                if temp[0][z] != row[z]:
                    cmd += attr[name][z] + " = '"+str (row[z])+"' ,"
            if len(cmd) > 0:
                cmd = "update {} set "+cmd
                cmd = cmd[:len(cmd)-1]
                cmd += "where "
                for i in range(len(pk_list[name])):
                    cmd += pk_list[name][i] + " = '" + str(row[pk_num[name][i]]) + "',"
                cmd = cmd[:len(cmd) - 1]
                cmd += ";"
                cur_2.execute(cmd.format(name))
```

در این بخش بر اساس توپولوژی سورتی که برای جدول ها داشتیم جدول های دیتابیس مقصد را آپدیت میکنیم. ابتدا با یک کوئری count(*) می فهمیم که جدول مبدا ما چند سطر دارد
حالا هر سطر را جداگانه با کوئری نوشته شده از جدول مبدا بازیابی میکنیم و در row قرار میدهیم .
حال با کوئری مناسب که از خط ۸۶ تا ۹۰ ساخته میشود همین سطر را در جدول مقصد بازیابی میکنیم و در temp قرار میدهیم.
اگر temp خالی بود یعنی جدول مقصد چنین سطری ندارد و ما در خط ۹۳ تا ۱۰۰ آن را به جدول مقصد اضافه میکنیم.
حال اگر وجود داشت بررسی میکنیم که فرقی با سطر ما در جدول مبدا دارد یا خیر.
هر اتریبوتی که مقدارش متفاوت باشد با جدول مبدا در set کوئری update قرار میدهیم
و در دستور cmd مینویسم اگر دستور cmd تهی نباشد یعنی حداقل یک اتریبوت مقدراش با اتریبوت سطر جدول مبدا متفاوت است
حال کوئری را کامل کرده و جدول مقصد را آپدیت میکنیم.

```
    # delete
    cmd = "select count(*) from {};"
    cur_2.execute(cmd.format(name))
    cnt = cur_2.fetchall()[0][0]
    for x in range(cnt):
        cmd = "select * from {} offset {} limit 1 ;"
        cur_2.execute(cmd.format(name,x))
        row = cur_2.fetchall()
        row = row[0]
        cmd = "select * from {} where "
        cmd_1=""
        for i in range(len(pk_list[name])):
            cmd_1 += pk_list[name][i] + " = '" + str(row[pk_num[name][i]]) + "',"
        cmd_1= cmd_1[:len(cmd_1) - 1]
        cmd_1 += ";"
        cmd += cmd_1
        cur.execute(cmd.format(name))
        temp = cur.fetchall()
        if len(temp) == 0:
            cmd = "delete from {} where " + cmd_1
            cur_2.execute(cmd.format(name))
```

در قسمت بعد مانند قسمت قبل ما ابتدا با دستور count(*) تعداد سطر جدول مقصد را میبابیم و در cnt قرار میدهیم
حال هر سطر جدول مقصد را جداگانه با کوئری مناسب بازیابی میکنیم و در row قرار میدهیم. حال یک کوئری مینویسیم که همین سطر را در جدول مبدا بازییای کند و در Temp قرار دهد اگر temp تهی بود یعنی این سطر در جدول مبدا حذف شده در نتیجه ما باید این سطر را در جدول مقصد حذف کنیم
سپس با کوئری مناسب این کار را میکنیم.