ETL Design - Day 2 Assignment

Prepared by Nischal badal

**Extraction of Different Types of Data into the database.**

There are several steps incurred while extracting different types of data files as .csv, .json, .xml into the database. They are briefly explained below:

We initially create a table to store the data. We analyze the data and create the columns. All of the columns are initially set to VARCHAR data type which accepts any kind of data as numbers, dates, etc.

This is done because of the mismatch in date format. In the data we are extracting, the date format might not match. for e.g. 1-2-2000 is accepted date format in the database as it can be mm-dd-yyyy or dd-mm-yyyy but sometimes the data can arrive in 1-2000-2 I.e. dd-yyyy-mm. This throws an error while inserting to the database. Hence, we change the datatype to text for all the records.

The two DDL queries in our assignment are:

create table raw\_employee(

    employee\_id VARCHAR(500),

    first\_name VARCHAR(500),

    last\_name VARCHAR(500),

    department\_id VARCHAR(500),

    department\_name VARCHAR(500),

    manager\_employee\_id VARCHAR(500),

    employee\_role VARCHAR(500),

    salary VARCHAR(500),

    hire\_date VARCHAR(500),

    terminated\_date VARCHAR(500),

    terminated\_reason VARCHAR(500),

    dob VARCHAR(500),

    fte VARCHAR(500),

    location VARCHAR(500)

);

create table raw\_timesheet(

    employee\_id VARCHAR(500),

    cost\_center VARCHAR(500),

    punch\_in\_time VARCHAR(500),

    punch\_out\_time VARCHAR(500),

    punch\_apply\_date VARCHAR(500),

    hours\_worked VARCHAR(500),

    paycode VARCHAR(500)

);

Now in src/pipeline folder, we create a pipeline to extract data to the database. The connection with the postgresql database is maintained using psycopg2 driver. There is a cursor that helps to run the sql queries into the database.

The following .connect() method helps to connect the script to the database using psycopg2.

psycopg2.connect(

        host = 'localhost',

        database = 'data-internship',

        user='postgres',

        password='nischal541',

        port = 5432

    )

After we establish the connection with the database, we have all our queries in the sql/queries folder.

we first truncate the database table for clearing all the values so that the values do not duplicate. This is done by this method.

 def truncate\_timesheet\_data():

            with open("../sql/queries/truncate\_timesheet\_data.sql") as f:

                sql = ' '.join(map(str, f.readlines()))

                cursor.execute(sql)

                connection.commit()

While importing csv data to the database, we have to remove the first line as it contains the header of the attributes. This is done by placing a flag I which in value 0 i.e. first value stops the insert query.

def extract\_timesheet\_data(filename):

            with open(filename, 'r') as f:

                i = 0

                for line in f:

                    if i==0:

                        i+=1

                        continue

                    row = line[:-1].split(",")

                    with open("../sql/queries/extract\_timesheet\_data.sql") as f:

                        insert\_query = ' '.join(map(str, f.readlines()))

                        cursor.execute(insert\_query, row)

                        connection.commit()

This method extracts the sql query from the given file and converts it into string as it is read in list from the file.

It then executes the query into the database using .execute() method. In this way, csv file is imported.

Similarly with the JSON file, we load the json file using json.load() method of json library.

Now, we iterate through the values and create two different lists of values and columns. we not format the values by converting them to match INSERT INTO table\_name VALUES() syntax.

After it is done, we place an insert query and pass the columns and values into it. In this way we then execute the query and commit the connection so that the connection applies operations to the database.

def extract\_employee\_data\_json(fileName):

            with open(fileName, 'r') as f:

                record\_list = json.load(f)

                values = [list(x.values()) for x in record\_list]

                columns = [list(x.keys()) for x in record\_list][0]

                values\_str = ""

                for i, record in enumerate(values):

                    val\_list = []

                    for v, val in enumerate(record):

                        if type(val) == str:

                            val = str(Json(val)).replace('"', '')

                        val\_list += [ str(val) ]

                    values\_str += "(" + ', '.join( val\_list ) + "),\n"

                values\_str = values\_str[:-2] + ";"

                insert\_query = '''

                            INSERT INTO raw\_employee (%s)

                            VALUES %s

                            ''' %(','.join(columns),values\_str)

                cursor.execute(insert\_query)

                connection.commit()

Similarly, with the xml case, we parse the xml file using etree.parse() method of etree and convert all the xml tags into values. A list is received and then we convert the list into insert equivalent query. After this, we insert the value into database using execute() method and database is committed.

  def extract\_employee\_data\_xml(fileName):

        root = etree.parse(fileName)

        for i in root.findall("Employee"):

            values = [i.find(n).text for n in ("employee\_id", "first\_name", "last\_name", "department\_id", "department\_name","manager\_employee\_id","employee\_role","salary","hire\_date","terminated\_date","terminated\_reason","dob","fte","location")]

            value\_str = "(" + str(values)[1:-1] + ");"

            insert\_query = '''

                            INSERT INTO raw\_employee ("employee\_id", "first\_name", "last\_name", "department\_id", "department\_name","manager\_employee\_id","employee\_role","salary","hire\_date","terminated\_date","terminated\_reason","dob","fte","location")

                            VALUES %s

                            ''' %(value\_str)

            cursor.execute(insert\_query)

            connection.commit()

In this way, csv, xml and json files are extracted to a database.