بسمه تعالی

پایتون و استخراج، تبدیل و بارگذاری داده‌ها

اغلب کاربران نیاز به «استخراج» (Extract) ، «تبدیل» (Transform) و «بارگذاری» (Load) داده‌ها از «پایگاه‌داده‌های» (databases)گوناگون به «انبارهای داده» (data warehouse) به منظور تجمیع داده‌ها جهت تحلیل‌های «هوش تجاری»(BI | Business Intelligence) و «تحلیل‌های داده» (Data Analysis) دارند. نرم‌افزارهای گوناگونی برای انجام فرآیند ETL تولید شده‌اند، ولی استفاده از آن‌ها برای کاربردهای ساده اغلب زیاده‌روی به نظر می‌رسد. در این مطلب چگونگی استخراج داده‌ها از SQL-Server،MySQL و Firebird، تبدیل داده‌ها و بارگذاری آن‌ها در SQL-Server (به عنوان انبار داده) با استفاده از پایتون ۳.۶ آموزش داده شده است. اکنون، کار با راه‌اندازی دایرکتوری پروژه آغاز می‌شود:

```
python_etl
    |__main.py
    |__db_credentials.py
    |__variables.py
    |__sql_queries.py
    |__etl.py
```

علاوه بر نصب SQL-Server ،MySQL و Firebird ، نیاز به سه ماژول پایتونی که در زیر بیان شده و اجرای pip install [module_name] وجود دارد.

- mysql-connector-python : اتصال به MySQL
- pyodbc : اتصال به sql-server
- fdb : اتصال به Firebird

راه‌اندازی مجوزها و متغیرهای پایگاه داده

راه‌اندازی یک «متغیر» برای ذخیره‌سازی نام پایگاه‌داده انبار داده در variables.py با دستور زیر انجام می‌شود.

```
datawarehouse_name = 'your_datawarehouse_name'
```

نصب همه پایگاه‌داده‌های منبع و هدف‌گیری رشته‌های اتصال پایگاه داده و مجوزها در db_credentials.py به صورتی که در زیر نمایش داده شده قابل انجام است. سپس، کانفیگ باید به صورت list ذخیره شود تا بتوان بعدا آن را در پایگاه داده‌های زیادی تکرار کرد.

```python
from variables import datawarehouse_name

# sql-server (target db, datawarehouse)
datawarehouse_db_config = {
  'Trusted_Connection': 'yes',
  'driver': '{SQL Server}',
  'server': 'datawarehouse_sql_server',
  'database': '{}'.format(datawarehouse_name)
  'user': 'your_db_username',
  'password': 'your_db_password',
  'autocommit': True,
}

# sql-server (source db)
sqlserver_db_config = [
  {
    'Trusted_Connection': 'yes',
    'driver': '{SQL Server}',
    'server': 'your_sql_server',
    'database': 'db1'
    'user': 'your_db_username',
    'password': 'your_db_password',
    'autocommit': True,
  }
]

# mysql
mysql_db_config = [
  {
    'user': 'your_user_1',
    'password': 'your_password_1',
    'host': 'db_connection_string_1',
    'database': 'db_1',
  },
  {
    'user': 'your_user_2',
    'password': 'your_password_2',
    'host': 'db_connection_string_2',
    'database': 'db_2',
  },
]

# firebird
fdb_db_config = [
  {
    'dsn': "/your/path/to/source.db",
    'user': "your_username",
    'password': "your_password",
  }
]
```

کوئری‌های SQL

sql_queries.py جایی است که همه کوئری‌های SQL برای استخراج از پایگاه‌داده‌های منبع و بارگذاری در پایگاه‌داده هدف (انبار داده) ذخیره می‌شوند. با توجه به اینکه در این فرآیند با پلتفرم‌های گوناگونی کار می‌شود، می‌توان از «نحو» (syntax) گوناگونی برای هر سکوی داده با جداسازی کوئری‌ها مطابق با نوع پایگاه داده استفاده کرد.

```
# example queries, will be different across different db platform
firebird_extract = ('''
  SELECT fbd_column_1, fbd_column_2, fbd_column_3
  FROM fbd_table;
''')

firebird_insert = ('''
  INSERT INTO table (column_1, column_2, column_3)
  VALUES (?, ?, ?)
''')

firebird_extract_2 = ('''
  SELECT fbd_column_1, fbd_column_2, fbd_column_3
  FROM fbd_table_2;
''')

firebird_insert_2 = ('''
  INSERT INTO table_2 (column_1, column_2, column_3)
  VALUES (?, ?, ?)
''')

sqlserver_extract = ('''
  SELECT sqlserver_column_1, sqlserver_column_2, sqlserver_column_3
  FROM sqlserver_table
''')

sqlserver_insert = ('''
  INSERT INTO table (column_1, column_2, column_3)
  VALUES (?, ?, ?)
''')

mysql_extract = ('''
  SELECT mysql_column_1, mysql_column_2, mysql_column_3
  FROM mysql_table
''')

mysql_insert = ('''
  INSERT INTO table (column_1, column_2, column_3)
  VALUES (?, ?, ?)
''')

# exporting queries
class SqlQuery:
  def __init__(self, extract_query, load_query):
    self.extract_query = extract_query
    self.load_query = load_query

# create instances for SqlQuery class
fbd_query = SqlQuery(firebird_extract, firebird_insert)
fbd_query_2 = SqlQuery(firebird_extract_2, firebird_insert_2)
sqlserver_query = SqlQuery(sqlserver_extract, sqlserver_insert)
mysql_query = SqlQuery(mysql_extract, mysql_insert)

# store as list for iteration
fbd_queries = [fbdquery, fbd_query_2]
sqlserver_queries = [sqlserver_query]
mysql_queries = [mysql_query]
```

استخراج، تبدیل، بارگذاری

در etl.py، ماژول‌ها و متغیرهای پایتون زیر باید بارگذاری شوند تا کار آغاز شود.

```
# python modules
import mysql.connector
import pyodbc
import fdb
# variables
from variables import datawarehouse_name
```

در اینجا دو متد etl() و etl_process() وجود دارند. etl_process() متدی برای راه‌اندازی اتصال منبع پایگاه داده مطابق با پلتفرم پایگاه‌داه و فراخوانی متد etl() است. در متد etl()، ابتدا کوئری استخراج اجرا، داده‌های SQL در متغیر Data ذخیره و این متغیر در پایگاه‌داده هدف که در واقع انبار داده کاربر است «درج» می‌شود. تبدیل داده‌ها با دستکاری متغیر data که از نوع «تاپل» است صورت می‌پذیرد.

```
def etl(query, source_cnx, target_cnx):
  # extract data from source db
  source_cursor = source_cnx.cursor()
  source_cursor.execute(query.extract_query)
  data = source_cursor.fetchall()
  source_cursor.close()

  # load data into warehouse db
  if data:
    target_cursor = target_cnx.cursor()
    target_cursor.execute("USE {}".format(datawarehouse_name))
    target_cursor.executemany(query.load_query, data)
    print('data loaded to warehouse db')
    target_cursor.close()
  else:
    print('data is empty')

def etl_process(queries, target_cnx, source_db_config, db_platform):
  # establish source db connection
  if db_platform == 'mysql':
    source_cnx = mysql.connector.connect(**source_db_config)
  elif db_platform == 'sqlserver':
    source_cnx = pyodbc.connect(**source_db_config)
  elif db_platform == 'firebird':
    source_cnx = fdb.connect(**source_db_config)
  else:
    return 'Error! unrecognised db platform'

  # loop through sql queries
  for query in queries:
    etl(query, source_cnx, target_cnx)

  # close the source db connection
  source_cnx.close()
```

قرار دادن کلیه موارد در کنار هم

در main.py می‌توان از طریق مجوزها و انجام ETL برای همه پایگاه‌های داده حلقه زد. کد زیر همه متغیرها و متدهای مرتبط را ایمپورت می‌کند.

```
# variables
from db_credentials import datawarehouse_db_config, sqlserver_db_config, mysql_db_config, fbd_db_config
from sql_queries import fbd_queries, sqlserver_queries, mysql_queries
from variables import *
# methods
from etl import etl_process
```

```
def main():
  print('starting etl')

  # establish connection for target database (sql-server)
  target_cnx = pyodbc.connect(**datawarehouse_db_config)

  # loop through credentials

  # mysql
  for config in mysql_db_config:
    try:
      print("loading db: " + config['database'])
      etl_process(mysql_queries, target_cnx, config, 'mysql')
    except Exception as error:
      print("etl for {} has error".format(config['database']))
      print('error message: {}'.format(error))
      continue

  # sql-server
  for config in sqlserver_db_config:
    try:
      print("loading db: " + config['database'])
      etl_process(sqlserver_queries, target_cnx, config, 'sqlserver')
    except Exception as error:
      print("etl for {} has error".format(config['database']))
      print('error message: {}'.format(error))
      continue

  # firebird
  for config in fbd_db_config:
    try:
      print("loading db: " + config['database'])
      etl_process(fbd_queries, target_cnx, config, 'firebird')
    except Exception as error:
      print("etl for {} has error".format(config['database']))
      print('error message: {}'.format(error))
      continue

  target_cnx.close()

if __name__ == "__main__":
  main()
```

اکنون کافیست کاربر در ترمینال python main.py را وارد کند و از اسکریپت پایتونی که برای انجام فرآیند ETL نوشته شده استفاده کند.

منبع: https://codeburst.io/using-python-script-for-data-etl-53138c567906