Python and Postgres: ODBC

Elly Fernández

Dynamic SQL

- Done in a general-purpose program
- Enables a program to construct and submit SQL queries at runtime and feed the results into a program variables as tuples
- There are a variety of standards used to connected to database servers from specific general-purpose languages
- JDBC is the standard for Java
- ODBC is another that works with a variety of languages
- Python has a database API (built on top of ODBC)

Accessing databases using Python

- Psycopg2 is the most popular PostgresSQL adapter for Python
- Need to import it; pip install if not already installed
- General process:
  - Establish a connection to the database
  - Call the cursor class that allows Python code to execute PostgreSQL; bound to the connection
  - Cursors created with the same connection are aware of the changes made to other cursors in the connection, but if there are multiple connections, the cursors may or may not see the changes–dependent on isolation levels
  - Commit the data; PYTHON DOES NOT AUTO-COMMIT BY DEFAULT, so this must be called after each transaction that alters data
  - Close the connection to the database

Psycopg connection class functions

- `connect(host="127.0.0.1", port: 5432, database="flight", user="efernandez_123", password="**********")`
  - Initializes a connection, note that the port will default to 5432
- `cursor(name=None, cursor_factory=None, scrollable=None, withhold=False)`
  - Returns a new cursor object using the connection created above
- `commit()`
  - Without calling commit all pending transaction to the database will be lost; there is a way to set up a connection with an "autocommit" mode
- `rollback()`
  - Roll back to the start of any pending transaction; also can do by closing a connection without committing
- `close()`
  - Close the connection and an InterfaceError will be raised if the connection is invoked via a cursor or another operation

Psycopg cursor class functions

- `execute(query, vars=None)`
  - Execute a database operation
  - Variables must be specified either with positional or named placeholders (example to come)
  - Returns None, but if query was executed, the return values can be retrieved via fetch*() where * could be (one, many, all)
- `executemany(query, vars_list)`
  - Executes a database operation against all parameter tuples or mappings found in the sequence vars_list
- `fetchone()`
  - Returns a single tuple of the next row of a query result set
- `fetchall()`
  - Fetches all the remaining rows of a query result as a list of tuples (an empty list if there is nothing left)

Open Database Connectivity (ODBC)

- It is a standard that defines a database application program interface (API)
- It is independent of all database management systems and operating systems, and is language independent
- This means that ODBC can be used with C, C++, C#, Go, and Ruby to name a few languages
- Allows general-purpose programs to connect and communicate with database servers

Sources

https://docs.microsoft.com/en-us/sql/odbc/reference/what-is-odbc?view=sql-server-ver15

https://www.geeksforgeeks.org/python-psycopg-connection-class/

https://www.psycopg.org/

https://docs.oracle.com/cd/A87860_01/doc/appdev.817/a76939/adg09dyn.htm

https://www.psycopg.org/docs/usage.html#transactions-control

https://www.programcreek.com/python/example/2032/psycopg2.connect

Database System Concepts by Abraham Silberschatz, Henry F. Korth, S. Sudarshan