Database Programming with Python and R

COSC 304 – Introduction to Database Systems

SQL Programming Overview

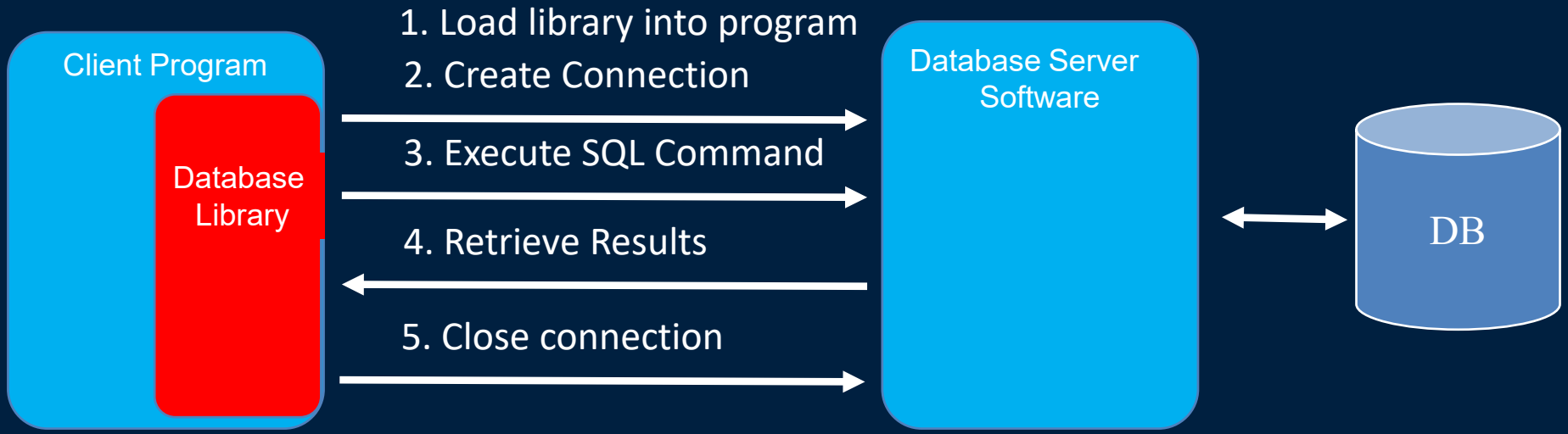
Interaction with a database is often through programs. Programming with a database requires:

- A database server and its connection information
- A programming language for writing the code to query the database
- A library or driver for connecting to the particular database system

General process for programming with a database:

1) Load the database access library
2) Create a connection to the database
3) Execute a SQL command
4) Retrieve database results produced by the command
5) Close the database connection

SQL Programming Architecture



1) Load Library Error:
- Library not found in path
- Wrong library

2) Create Connection Errors:
- Invalid server URL
- Incorrect user/password
- Network issues
- Wrong library

3) Execute SQL Errors:
- Incorrect SQL
- Wrong database/table
- Improper library use

4) Retrieve Results Errors:
- Wrong column name
- Wrong column index
- Off-by-one
- Improper library use

5) Close Connection:
- FORGET TO DO IT!

SQL Programming in Python

Python can connect to many SQL databases by loading the proper library. We will use MySQL, Microsoft SQL Server, and SQLite.

- Python has a standard API called DB-API 2.0.

Step #1 – Install Libraries:

- MySQL: `conda install mysql-connector-python`
- SQL Server:
  - https://docs.microsoft.com/en-us/sql/connect/python/python-driver-for-sql-server
  - Install ODBC driver for SQL Server then: `pip install pyodbc`
- SQLite: `pip install pysqlite3`
- Note: May have to find `conda.bat` or `conda.sh` or `pip`. Check `Anaconda3/Library/bin`.

Python and MySQL

```
import mysql.connector
try:
    cnx = mysql.connector.connect(user='testuser', password='todo',
                                  host='localhost', database='workson')
    cursor = cnx.cursor()
    query = "SELECT eno, ename, salary FROM emp WHERE salary < 50000"
    cursor.execute(query)
    for (eno, ename, salary) in cursor:
        print(eno, ename, salary)
    cursor.close()
except mysql.connector.Error as err:
    print(err)
finally:
    cnx.close()
```

Import MySQL API

DB Connection Info

Execute query and process results

Close connection

Python and MySQL Query with Parameters

```
import mysql.connector
try:
    cnx = mysql.connector.connect(user='testuser', password='todo',
                  host='localhost', database='workson')
    cursor = cnx.cursor()
    query = "SELECT pno, count(*) FROM emp NATURAL JOIN workson "
            "WHERE salary < %s and ename > %s GROUP BY pno"
    cursor.execute(query, (45000, 'L'))
    for (pno, cnt) in cursor:
        print(pno, cnt)
    cursor.close()
except mysql.connector.Error as err:
    print(err)
finally:
    cnx.close()
```

Python and MySQL INSERT Statement

```
try:
    cnx = mysql.connector.connect(user='testuser', password='todo',
                host='localhost', database='testuser')
    cursor = cnx.cursor()
    sql = "INSERT INTO emp (eno, ename, salary) VALUES (%s, %s, %s)"
    cursor.execute(sql, ('E10', 'Test Person', 100000))
    cnx.commit() # Update data by committing transaction on connection

    sql = "DELETE FROM emp WHERE eno = 'E10'"
    cursor.execute(sql) # Remove employee just added
    cursor.close()
    cnx.commit()
except mysql.connector.Error as err:
    print(err)
finally:
    cnx.close()
```

Try it: Python and MySQL

Question: Write a program that queries the workson database and returns the employees grouped by title where the employee name is after `'J'`. The output should display their title and the average salary for that title. Connection info:

- `cnx = mysql.connector.connect(user='testuser', password='todo', host='localhost', database='workson')`

Output:

EE	30000.000000
ME	40000.000000
PR	20000.000000
SA	50000.000000

ODBC Overview

ODBC (Open Database Connectivity) is a standard database ***application programming interface*** (***API***) developed by Microsoft for connecting to databases.

- Java has a similar standard API called JDBC.

The ODBC API contains methods that allow user programs to connect to a database, execute queries, and retrieve results.

For each database system, the vendor writes an **ODBC driver** that implements the API. Application programs can access different systems simply by changing the driver used in their program.

Python and Microsoft SQL Server

ODBC is the underlying technology used to connect to Microsoft data sources (Excel, Access, SQL Server, etc.) from Python.

ODBC is language independent (developed first in C/C++) and has been implemented in Python and packaged as library `pyodbc`.

Python and Microsoft SQL Server

```
import pyodbc
try:
    cnx = pyodbc.connect("""DRIVER={ODBC Driver 17 for SQL Server};
                            SERVER=localhost;
                            DATABASE=workson;UID=sa;PWD=todo""")
    cursor = cnx.cursor()
    cursor.execute("SELECT * FROM emp WHERE salary < ?", [50000])
    for row in cursor:
        print(row[0],row[1], row.salary)
except pyodbc.Error as err:
    print(err)
finally:
    cnx.close()
```

Try it: Python and Microsoft SQL Server

Question: Write a program that asks the user for a total salary value. Write a query on the workson database that returns the projects where the total salary of employees working on the project is greater than the value the user entered. Connection info:

- `cnx = pyodbc.connect("""DRIVER={ODBC Driver 17 for SQL Server}; SERVER=localhost; DATABASE=workson;UID=sa;PWD=todo""")`

Output:

```
Enter total salary:75000
('P1   ', Decimal('80000.00'))
('P2   ', Decimal('120000.00'))
('P3   ', Decimal('80000.00'))
```

Python and SQLite

SQLite is an embedded database designed to be used within other programs.

The amount of data stored in a SQLite database is often considerably smaller than other systems. However, the system is easy to use and transport data between systems.

Python and SQLite

```
import sqlite3
try:
    cnx = sqlite3.connect("test.db")
    cursor = cnx.cursor()
    cursor.execute("CREATE TABLE test (id int, name text)")
    cursor.execute("INSERT INTO test VALUES(1, 'Joe')")
    cursor.execute("INSERT INTO test VALUES(2, 'Jen')")
    cursor.execute("INSERT INTO test VALUES(3, 'Jeff')")
    cnx.commit()
    cursor.execute("SELECT * FROM test WHERE name > 'Je'")
    for row in cursor:
        print(row)
    cursor.execute("DROP TABLE test")
except sqlite3.Error as err:
    print(err)
finally:
    cnx.close()
```

SQL Programming in R

R can connect to many SQL databases by loading the proper library. We will use MySQL, Microsoft SQL Server, and SQLite.

- R has `DBI` standard API for querying databases.
  - https://www.rdocumentation.org/packages/DBI

Step #1 – Install Libraries:

- MySQL: `install.packages("RMariaDB")`
- SQL Server: `install.packages("odbc")`
- SQLite: `install.packages("RSQLite")`

R and MySQL

```
library("RMariaDB")
con <- dbConnect(RMariaDB::MariaDB(), user='testuser', password='todo',
dbname='workson', host='localhost')

# List database tables
dbListTables(con)

# Execute a query
res <- dbGetQuery(con, "SELECT * FROM emp")

# Disconnect
dbDisconnect(con)
```

R and Microsoft SQLServer

```
library("DBI")
library("odbc")

db <- dbConnect(odbc::odbc(),
                      Driver = 'ODBC Driver 17 for SQL Server',
                      Server = 'localhost', Database = "workson",
                      UID='sa', PWD='todo')

data <- dbGetQuery(db,"SELECT * FROM emp")
data

# Disconnect
dbDisconnect(db)
```

R and SQLite

```
library("RSQLite")
library("DBI")

con <- dbConnect(RSQLite::SQLite(), ":memory:")

# Write data frame to table
testData <- data.frame(id = 1:5,
                       name=c("Abe", "Ben", "Cindy", "Dana", "Emma"))

dbWriteTable(con, "test", testData)
data <- dbGetQuery(con,"SELECT * FROM test")
data

# Disconnect
dbDisconnect(con)
```

R Database Querying with dplyr

```
library(dplyr)
library(dbplyr)

con <- dbConnect(RMariaDB::MariaDB(), user='testuser', password='todo',
dbname='workson', host='localhost')

# Execute a query
data <- tbl(con, "emp") %>%
  group_by(title) %>%
  summarise(
    totalSalary = sum(salary),
    totalEmp = n())
show_query(data)
data # Executes query

# Disconnect
dbDisconnect(con)
```

SQL Programming: R and Python Question

Question: **True or False:** Although the API methods may be slightly different, the general approach to querying a database is very similar in R and Python.

A) true

B) false

R SQL Programming Question

Question: What is the API used to query Microsoft SQL Server with R?

A) JDBC

B) RMariaDB

C) pyodbc

D) ODBC

Try it: R and SQLite

Question: Write a program that creates a data frame with 5 rows consisting of id (integer) and message (string). Save the data frame to SQLite. Query SQLite to only return values with id < 12 and message has an `"e"` in it. Print the results.

Data:

```
  id message
1 10   Hello
2 11 Goodbye
3 12  Friend
4 13     Ant
5 14     Out
```

Conclusion

Querying databases can be done from any language with many different APIs. Querying has these steps:

1) Load the database access library
2) Create a connection to the database
3) Execute a SQL command
4) Retrieve database results produced by the command
5) Close the database connection

We have seen how to query MySQL, Microsoft SQL Server, and SQLite using both Python and R.

Objectives

- Explain the common steps in querying a database using a programming language.
- Draw and explain the database-program architecture and the key components.
- Write a simple program to query a database in both Python and R.
- Use a variety of databases including MySQL, Microsoft SQL Server, and SQLite.
- Explain the purpose of ODBC.
- List the components of a database connection URL.
- Use dplyr to query a database.
- Debug and resolve database connection and usage issues in Python and R code.


UBC
THE UNIVERSITY OF BRITISH COLUMBIA