Demo-Notes

Friday, September 17, 2021 7:35 PM

To learn some basic SQL, I decided to create a local test environment to demonstrate a Java application reading data from a Microsoft SQL server, transforming it, and then writing it to a Postgres Database.

This is a reasonable simulation, but I realize that it is missing many of the aspects needed for actual production use. Examples of a few areas where I simplified for the demo that would need to be addressed include:

1) The transformations are simple and would be much more complex in production.
2) Better security would be needed such as stronger passwords and keeping passwords in an external vault and not in SQL files and in Java code files
3) Use connection pooling to the databases

Technologies used to create the demo:

- VMware Virtual Machine
- Ubuntu 20.04 running in a VM
- Docker Engine for Linux
  - Postgres v12.8 image pulled from DockerHub and run in a Docker Container
  - Microsoft SQL 2019 image pulled from DockerHub and run in a Docker Container
- psql CLI
- sqlcmd CLI
- Dev stack:
  - Java - openJDK 11
  - java.sql - Java client library that supports ODBC connections to Postgres and to Microsoft SQL Server
  - Intellij IDE
- Version Control System: git
- Build System: Gradle
  - Maven Repository - For finding and pulling Java Library dependencies in Gradle - https://mvnrepository.com/
- Java source files
  - Main.java
  - Pg.java - contains methods for inserting and selecting from the Postgres database
  - MsSql.java - contains methods for inserting and selecting from the Microsoft SQL Server database
- T-SQL - Microsoft Transact SQL
- Postgres SQL

Detailed description of how I built the demo environment

1) Create a Virtual Machine on VMware
   a. Install Linux (Ubuntu 20.04) Operating system in the VM
2) Setup the version control environment (git)
   a. Add a repo on Github and git clone it to a local repo
   b. Set up SSH key on Linux to allow git to authenticate with an SSH public key installed on Github

      ```
      eval "$(ssh-agent -s)"
      ssh-add ~/.ssh/id_ed25519
      ```

   c. Set files to not push to the remote git repo by editing .gitignore

      ```
      # Intellii J IDE files
      **/.idea/

      # Common Ignore Folders
      **/build/
      **/.gradle/
      ```

3) Install Docker Engine on Linux - https://docs.docker.com/engine/install/ubuntu/

```
sudo apt-get install \apt-transport-https \ca-certificates \curl \gnupg \lsb-release

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor-
o/usr/share/keyrings/docker-archive-keyring.gpg

echo\"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu \
 $(lsb_release -cs) stable"| sudo tee /etc/apt/sources.list.d/docker.list >/dev/null

 sudo apt-get install docker-ce docker-ce-cli containerd.io
```

4) Download the Postgres Container image from Docker Hub
   a. Postgres Version 12.8
      sudo docker pull postgres:12.8
   b. To view local docker images: docker image ls
5) Run the Postgres Docker Container
   a. To run a docker image to start the docker container (version 12.8 Postgres Docker image name is postgres:12.8):
      sudo docker run --name pg -p 5432:5432 -e POSTGRES_PASSWORD=pw -d postgres:12.8
      a. Docker command line: https://docs.docker.com/engine/reference/commandline/cli/
         1) Quick reference: https://www.docker.com/sites/default/files/d8/2019-09/docker-cheat-sheet.pdf
         2) To stop a container:
            sudo docker container stop pg
         3) To restart a stopped container:
            sudo docker container restart pg
         4) To delete a container
            sudo docker container rm pg
         5) To view the logs for a container
            sudo docker logs pg
6) Install the Postgres command-line interface (CLI) - psql

```
https://stackoverflow.com/questions/28290488/get-error-you-must-install-at-least-one-postgresql-client-version-package-whe
```

7) Setup a database and a table in the Postgres server running in the Docker Container
   1) Connect to the Postgres SQL Server running in the docker container using an interactive psql session
      psql -h localhost -p 5432 -U postgres
      a. User is postgres
      b. PASSWORD=pw
   2) Show the postgres version: psql --version
   3) Edit a pg_demo.sql file that will create a database and a table in that database in the Postgres server
      a. Create a Postgres database - create database pg_testdb
         https://www.tutorialspoint.com/postgresql/postgresql_create_database.htm
         1) Verify doing an interactive psql session: \l
      b. To prevent an error, you can use the following to check for the database existence before creating it

```
SELECT 'CREATE DATABASE <your db name>' WHERE NOT EXISTS (SELECT FROM
pg_database WHERE datname = '<your db name>')\gexec
```

      c. Create the airplane table in Postgres -
         https://www.tutorialspoint.com/postgresql/postgresql_create_table.htm
         1) To skip creating the table if it already exists, add "IF NOT EXISTS" as in the command below

```
CREATE TABLE IF NOT EXISTS table_name (
```

      d. To show the tables in a database

```
\c   database_name
```

```
\d
```

```
SELECT * FROM pg_testdb.*;
```

e. To show the rows in a table

```
SELECT * FROM table_name;
```

f. To insert a row into a table in Postgres

```
INSERT INTO table_name(column1, column2, …) VALUES (value1, value2, …);
```

a. Execute the pg_demo.sql file using psql to Create the database and add the airplane table
   psql `-h localhost -p 5432 -U postgres` -f pg_demo.sql

b. Postgres SQL quick reference - https://www.postgresqltutorial.com/postgresql-cheat-sheet/

8) Write a Java application to read (select) and write (insert) from Postgress

a. Install Intellij IDE on the Ubuntu Linux VM

b. Install openjdk 11 on Ubuntu VM: https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt-on-ubuntu-20-04

c. Install Build tool, Gradle, on Ubuntu VM using snap: https://snapcraft.io/install/gradle/ubuntu

d. Gradle Configuration - build.gradle

   a. Add the following to the top of the build.gradle file to configure plugins, and add a dependency to have Gradle install the Postgres Java client library

```
plugins{
   //Apply the java plugin to add support for Java
   id 'java'


   //Apply the application plugin to add support for building a CLI application.
   id 'application'


   //Apply the Intellij IDEA gradle plugin
   id 'idea'
}

// In this section you declare where to find the dependencies of your project
repositories {
   // Use jcenter for resolving your dependencies.
   // You can declare any Maven/Ivy/file repository here.
   jcenter()
}

dependencies {
        // https://mvnrepository.com/artifact/org.postgresql/postgresql
        // Installs the Java JDBC client for Postgress
        // Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java
        implementation group: 'org.postgresql', name: 'postgresql', version: '42.2.23'
}
```

   b. Add the following to the bottom of the build.gradle file to be able to run the Java application

```
application {
   //Define the main class for the application.
   mainClassName ='com.suncountry.sqldemo.Main'
}
```

e. Write Java Class (Pg.java) to connect to the Postgres Database, insert rows into the airplane table, and then perform a select to retrieve rows from the airplane table.

   a. Postgres Java Client Library documentation: https://www.tutorialspoint.com/postgresql/postgresql_java.htm

   b. Setup the driver for Postgres and connect to the Postgress database (pg_testdb)

```
        Class.forName("org.postgresql.Driver");
```

```
        connection =DriverManager.getConnection("jdbc:postgresql://localhost:5432/pg_testdb",
"postgres", "pw");
        connection.setAutoCommit(false);
        System.out.println("Opened postgres database successfully");
```

c. Select rows from Postgress - Code snippet for selecting rows into a result set "rs" from the airplane table

```
        ResultSetrs =statement.executeQuery( "SELECT * FROM airplane;");
        while( rs.next() ) {
           int airplaneid =rs.getInt("airplaneid");
           String callsign =rs.getString("callsign");
           String aircraft_type  =rs.getString("aircraft_type");
           int passenger_capacity =rs.getInt("passenger_capacity");
           int max_altitude =rs.getInt("max_altitude");
           int max_speed =rs.getInt("max_speed");
           System.out.println( "airplaneid = "+airplaneid );
           System.out.println( "callsign = "+callsign );
           System.out.println( "aircraft_type = "+aircraft_type );
           System.out.println( "passenger_capacity = "+passenger_capacity );
           System.out.println( "max_altitude = "+max_altitude );
           System.out.println( "max_speed = "+max_speed );
           System.out.println();
        }
        rs.close();
```

d. Main to invoke the insert and select methods

9) Install a Microsoft SQL Server Docker Container, run it, and connect to it with sqlcmd CLI
   a. Get the MS SQL Server image from Docker Hub
      https://hub.docker.com/_/microsoft-mssql-server
      sudo docker pull mcr.microsoft.com/mssql/server:2019-CU12-ubuntu-20.04
   b. Start the MS SQL Server Container
      sudo docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=Pw123456*" --name mssql -p 1433:1433 -d mcr.microsoft.com/mssql/server:2019-CU12-ubuntu-20.04
   c. Connect to the MS SQL Server running in the docker container using CLI tools
      a. Install sqlcmd - Command-line query tool for MS SQL Server
         Install SQL Server command-line tools on Linux - SQL Server | Microsoft Docs
         1) sudo apt install msodbcsql1
         2) sudo apt install libodbc1:amd64
      b. sqlcmd help
         Working with the SQL Server command line (sqlcmd) (sqlshack.com)
      c. Connect to a MS SQL Server interactive session
         sqlcmd -S localhost -U SA -P Pw123456*
      d. Use sqlcmd to run a script in the mssql_demo.sql file to create the sql_demo database and airplane table
         sqlcmd -S localhost -U SA -P Pw123456* -i mssql_demo.sql
   d. Create the database using T-SQL (Transact SQL) --- This is what Microsoft calls SQL
      a. Create a database

```
USE master;
GO
IF DB_ID ( N'Music' ) IS NOT NULL
DROP DATABASE sql_demo;
GO
CREATE DATABASE sql_demo;
GO
```

         1) List databases:
            sp_databases
            go

b. Create the airplane table in MS SQL Server - Edit a file named mssql_demo.sql. This will create a table called airplane and insert 2 rows

```
USE sql_demo
IF OBJECT_ID ('dbo.airplane', 'U') IS NOT NULL
  DROP TABLE airplane;
GO
CREATE TABLE airplane
(
  airplaneid  int,
  callsign  varchar (50),
  aircraft_type varchar(50),
  passenger_capacity int,
  max_altitude int,
  max_speed int
);
GO

INSERT airplane
  (airplaneid, callsign, aircraft_type,  passenger_capacity, max_altitude, max_speed)
VALUES
  (12345678, 'SC737', '737', 354, 40000, 540);

INSERT airplane
  (airplaneid, callsign, aircraft_type,  passenger_capacity, max_altitude, max_speed)
VALUES
  (12345679, 'SC747', '747', 600, 47000, 640);
GO
```

1) Run the SQL file
   sqlcmd -S localhost -U SA -P Pw123456* -i mssql_demo.sql
2) View the table
   use sql_demo
   select * from airplane
   go

10) Write a Java Class (MsSQL.java) that contains methods to read (select) and write (insert) from MS SQL Server
    a. Java Ubuntu (microsoft.com)
    b. Configure the Java client for MS SQL by setting the Class.forName and then set the connection to the database (sql_demo).

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
connection =DriverManager
    .getConnection("jdbc:sqlserver://localhost;database=sql_demo;user=sa;password=Pw123456*");
connection.setAutoCommit(false);
```

    c. Add dependency for the Microsoft JDBC driver to Gradle to fix the error: java.lang.ClassNotFoundException: com.microsoft.sqlserver.jdbc.SQLServerDriver. Add to the dependencies section in build.gradle

```
// https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-jdbc
implementation group: 'com.microsoft.sqlserver', name: 'mssql-jdbc', version: '9.4.0.jre11'
```

11) Run the Java Application that reads rows from Microsoft SQL Server, transforms the data, and then inserts the rows into the Postgres database.
    a. From a terminal in the project directory
       a. ./gradlew run
    b. Or run it from within Intellij IDEA