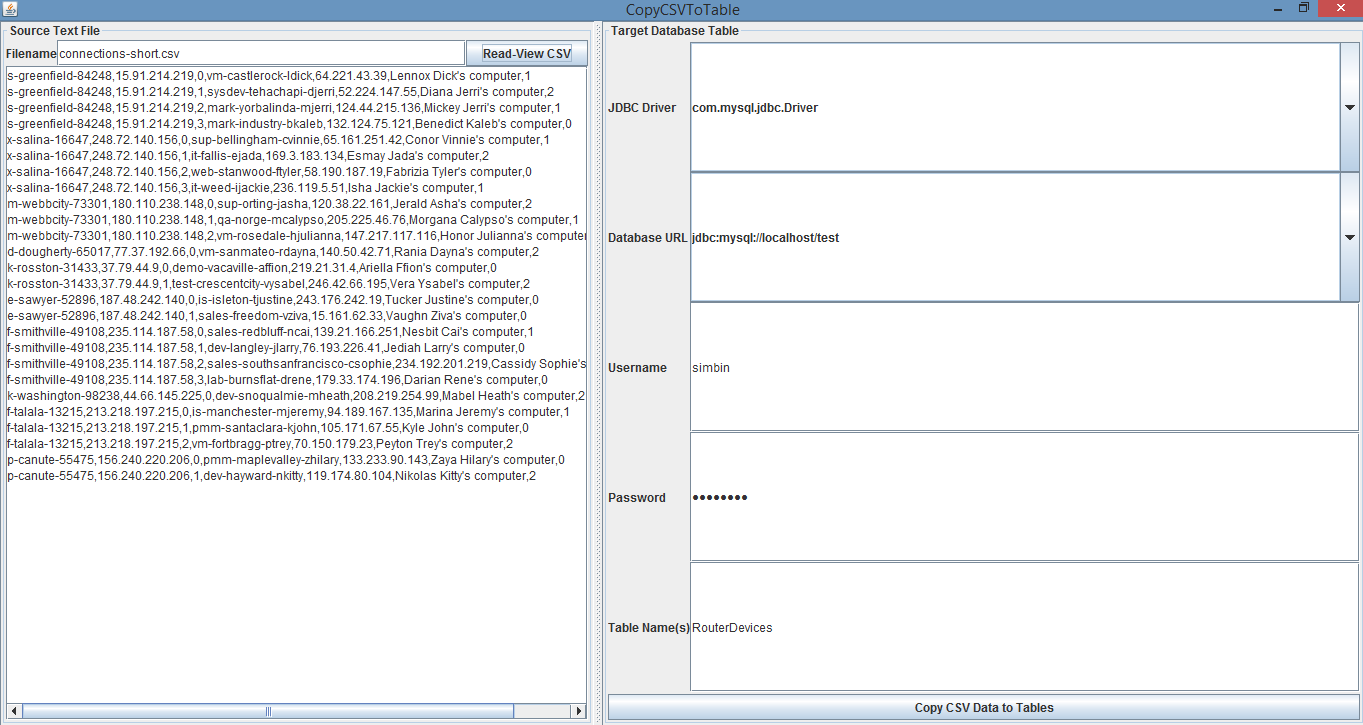
21-04-2017

**SIMEON BINIATIDIS**

**SolarWinds Coding Test**

**Copy CSV File To Tables**

****

**Figure 1**

First you Press the Read-View Button. The CSV shows up in the TextArea.

Then you Press the ‘Copy CSV Data to Tables’ and the data get distributed between the ROUTER, DEVICE, CONNECTIONS Tables. I use a ‘RouterDevices’ helper Table to Bulk Copy all data to 1 ONE Table

With Columns {RouterName, RouterIP, RouterPort, DeviceName, DeviceIP, DeviceDescription, DevicePort}. I use the ‘getGeneratedKeys’ statement API to get the auto-Increment indices of ROUTER, DEVICE Tables so that I can Insert them to CONNECTIONS Table.

I also have code to do Batch Inserts. This code is well tested in other projects in the past. Here for economy of time (non-existent) and space I disabled the code through an appropriate flag. The code is there for anybody to see. The performance gains are tremendous compared to individual Inserts done one-by-one, especially for very large CSV files.

I accommodate only single-quotes on the DeviceDescription Field, and I escape the single-quotes by doubling them, that’s how SQL wants it. I also assume the strings are naked (no double or single quotes) and I single-quote them when I read them in memory, on the RouterDevice Object. In fact, I am using an ArrayList of RouterDevice Objects to keep all CSV Lines, properly pre-processed, in memory.

When you press the “Copy CSV” button I iterate through the ArrayList, I pick the values up, build the INSERT’S accordingly and accurately and I .. “executeUpdate”. In the case of ROUTER and DEVICE I need the auto-increment value to use for the CONNECTIONS Table.

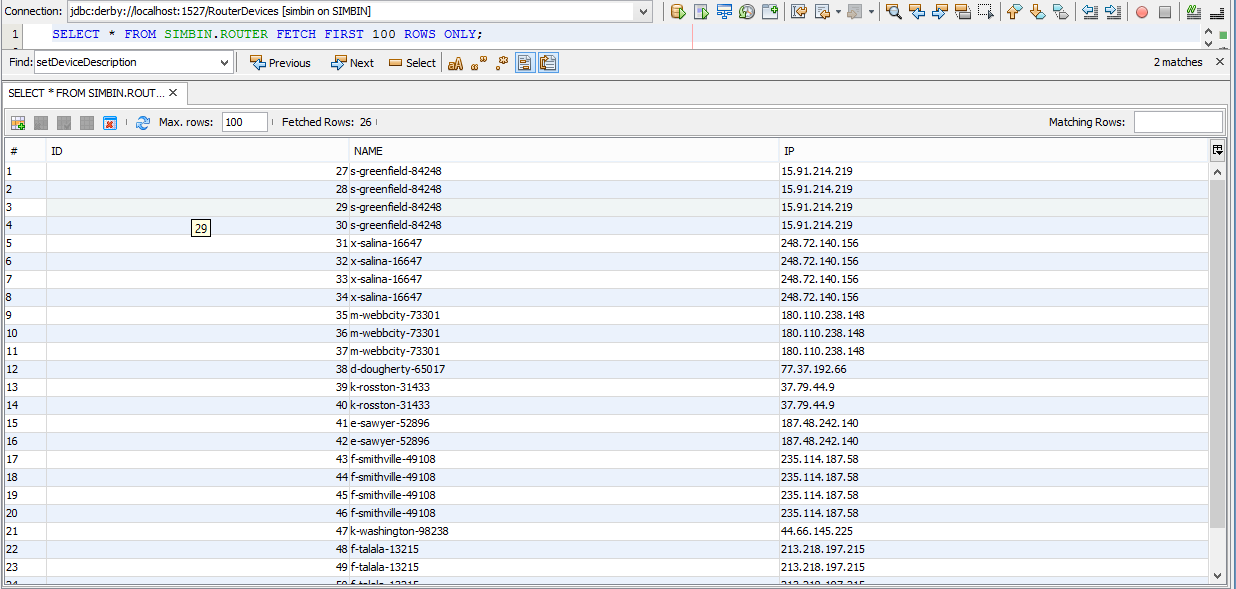


Figure 2

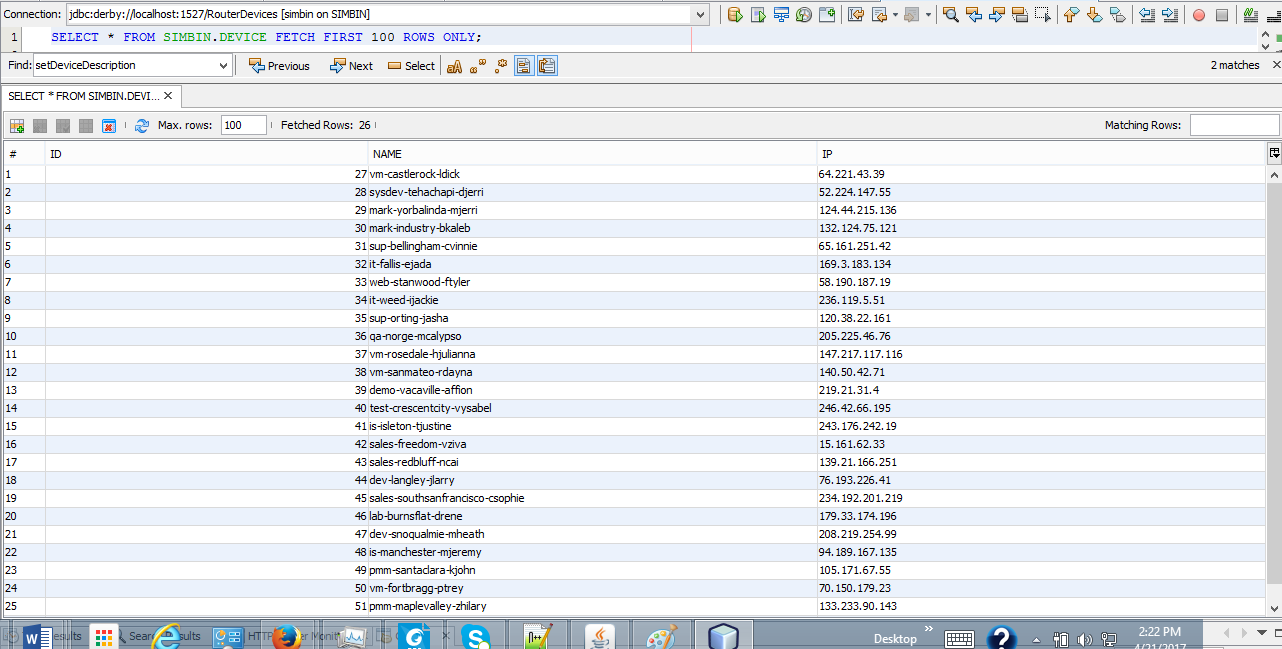


Figure 3

In the Swing Window, Figure 1, I have (2) combo’s (DB DriverName, DB URL) and a (3) three of TextField’s, for Username, Password, and Table Names, although I only insert RouterDevice there the Helper Table I’m using for debugging purposes.

I did not see enormous need for multi-threading, even though for large CSV files again it would sound

beneficial and valuable. For the purposes of this exercise I have used the Apache Derby DB. I’m working with Netbeans and IntelliJ Ultimate. If I had the Re-sharper I could work with Visual Studio but I don’t.

In Figure 4 underneath I include a pretty good view of the Apache Derby Database with the Tables, FK’s

PK’s and the rest of paraphernalia.

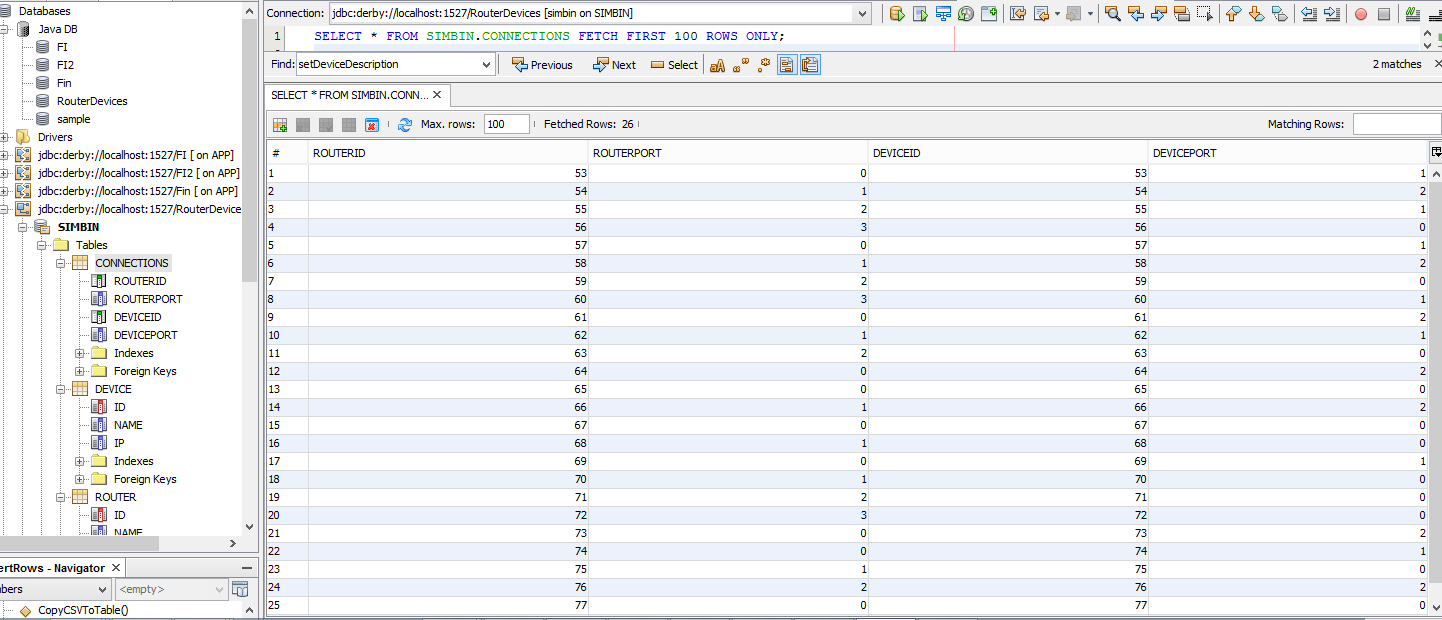


Figure 4

The following SQL DDL are needed if one decides to not use the DB Utility to build it. In the case of Apache Derby DB, an Embedded DB in Java, Netbeans, IntelliJ, might be necessary to do the DDL commands through SQL, the other DB Manipulation through the IDE leaves a lot to be desired

CREATE TABLE **ROUTER**

(

id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),

name VARCHAR(40) NOT NULL,

ip VARCHAR(1024),

CONSTRAINT primary\_key PRIMARY KEY (id)

) ;

CREATE TABLE **DEVICE**

(

id INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),

name VARCHAR(40) NOT NULL,

ip VARCHAR(1024),

CONSTRAINT primary\_key2 PRIMARY KEY (id)

) ;

CREATE TABLE **CONNECTIONS**

(

RouterId INTEGER NOT NULL CONSTRAINT FK\_RouterId REFERENCES ROUTER(Id)

ON DELETE CASCADE ON UPDATE RESTRICT,

RouterPort INTEGER,

DeviceId INTEGER NOT NULL CONSTRAINT FK\_DeviceId REFERENCES DEVICE(Id)

ON DELETE CASCADE ON UPDATE RESTRICT,

DevicePort INTEGER

) ;

The only other approach that I considered was to use Batch Updates but finally was carried away.

There are a myriad of Test Cases to consider, most of them have to do with special characters that you would wish to entertain in the content of the CSV File. You would have to take care of business and qualify the fields so that you build accurate SQL to Insert the Data to Tables.

The code Flow goes as follows. When the ‘Read-Show CSV’ is pressed, method ‘readShowFile()’ is called.

The method employs Scanner class and relevant Java I/O API’s to Read the CSV, append it to the Swing TextArea. Then it saves the same info to the RDList which is an ArrayList<RouterDevice>.

When the ‘Copy CSV to Tables’ is pressed then the CopyFile() is called, which is misnomer. Data are about to be copied to the 3 three Tables plus the Helper RouterDevice table with the 7 seven fields as those map the 7 columns of the CSV. The ‘CopyFile’ installs the Derby Driver, gets a connection object and calls ‘insertRows’. InsertRows() starts building the SQL, the Statements and inquires about Batch Update mode. If it’s supported by the specific driver, then it engages it through ‘addBatch’ and ‘executeBatch’. Otherwise it exploits the power of the executeUpdate(), getGeneratedKeys() to settle matters in the 3 Tables + the helper.

As mentioned before I utilize another class, the ‘RouterDevice’ which is the prototype of the RouterDevice Object. This Object, along with its setters and getters and the subsequent ArrayList<RouterDevice> are the basis of saving the CSV in Memory, if there is a need to.

Before the save to Memory, I pre-process the CSV, to escape the single-quote and dress-up the strings with single-quotes(all except columns 3 and 7 of CSV, are strings.3 and 7 I have them as Integers).