Team 4 - codes-n-roses
Santhosh Saravanan
Matthew Galvez
12 May 2022

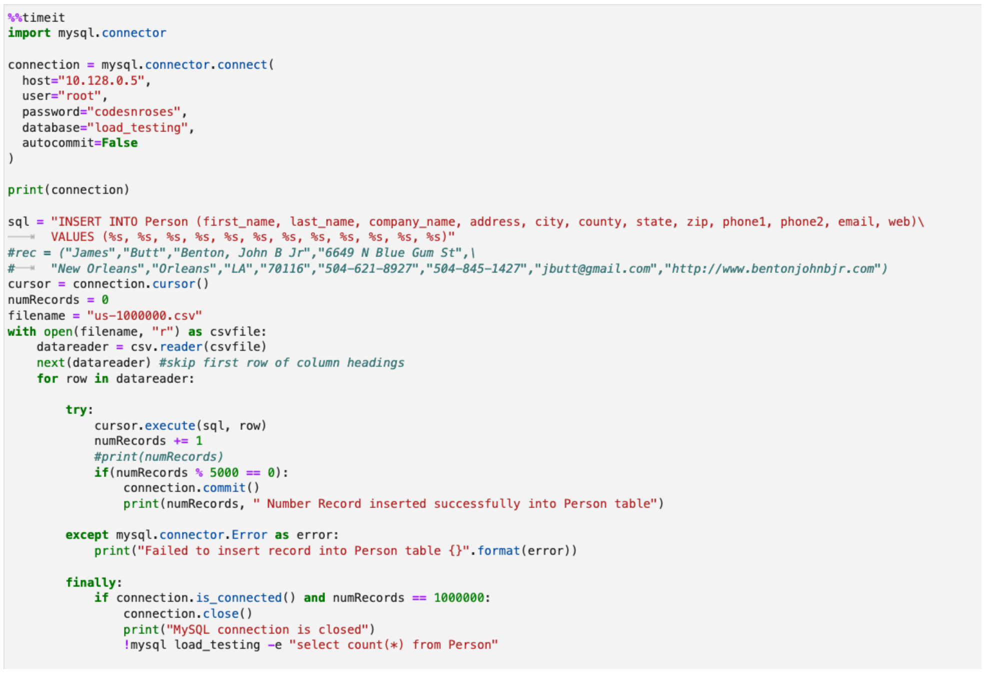
Final Project Report

Section 1: Overview:

For our final project we were to test a regional database management system and we decided to use MySQL as our regional database management system. We tested this by measuring the latency of writing 8 million records. We first recorded a baseline time by inputting a single record at a time. Following this we used batch inserts and then batch inserts with hardware upgrades.

Section 2: Baseline Run (Milestone 1):

In order to record a baseline time we first imported a mysql connector and set up a connection to our database. Following this we open the file holding our data we want to insert into the "Person" table. While going through the file after row we insert the data into the table and commit the change. When five thousand records have been inserted we then display the number of records that have been inserted and after 1 million records we close the connection and display the amount of records in the "Person" table; this was made as an effort to debug our single batch inserts statements and debug thes status of the mysql server incrementally. This continues until we hit 8 million records and get our baseline time of 10min 7s (+- 33.8s) per loop.



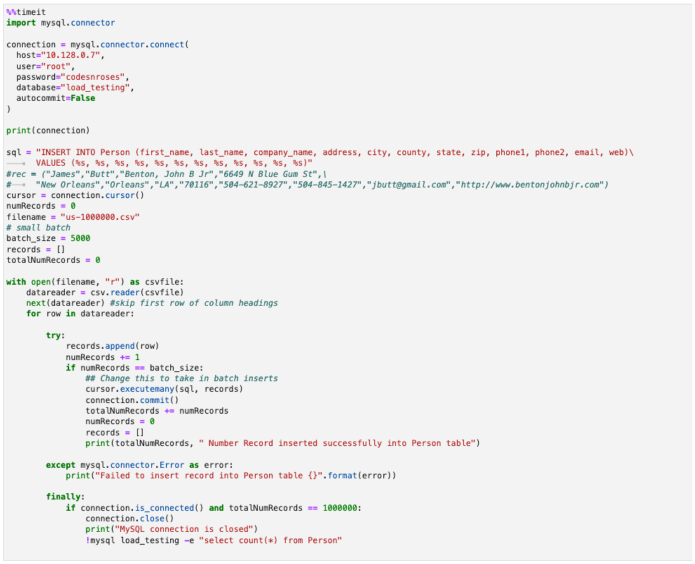
```
%%timeit
import mysql.connector

connection = mysql.connector.connect(
  host="10.128.0.5",
  user="root",
  password="codesnroses",
  database="load_testing",
  autocommit=False
)

print(connection)

sql = "INSERT INTO Person (first_name, last_name, company_name, address, city, county, state, zip, phone1, phone2, email, web)\
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
#rec = ("James","Butt","Benton, John B Jr","6649 N Blue Gum St",\
#       "New Orleans","Orleans","LA","70116","504-621-8927","504-845-1427","jbutt@gmail.com","http://www.bentonjohnbjr.com")
cursor = connection.cursor()
numRecords = 0
filename = "us-1000000.csv"
with open(filename, "r") as csvfile:
    datareader = csv.reader(csvfile)
    next(datareader) #skip first row of column headings
    for row in datareader:

        try:
            cursor.execute(sql, row)
            numRecords += 1
            #print(numRecords)
            if(numRecords % 5000 == 0):
                connection.commit()
                print(numRecords, " Number Record inserted successfully into Person table")

        except mysql.connector.Error as error:
            print("Failed to insert record into Person table {}".format(error))

        finally:
            if connection.is_connected() and numRecords == 1000000:
                connection.close()
                print("MySQL connection is closed")
                !mysql load_testing -e "select count(*) from Person"
```

Figure 1: Code Block for Inserting 1 entry at a time

Section 3: Batch Runs (Milestone 2):

Team 4 - codes-n-roses
Santhosh Saravanan
Matthew Galvez
12 May 2022

We used the same process as the step above except we replaced “execute(sql, row)” with “executemany(sql, records)”, where records is a python list that gets updated with the data from the csv file. When “records” reach the size of the batch size we then do a batch insert. The display messages are the same as above. Our small batch size (5000) had a time of 2min 41s +- 6.46s per loop, our medium batch size (50000) had a time of 2min 35s +- 6.12s per loop, and our large batch size (500000) had a time of 2min 34s +- 5.99s per loop. Problems we faced during this step and below were using outdated sudo commands and debugging the insert batch statements.



```
%%timeit
import mysql.connector

connection = mysql.connector.connect(
  host="10.128.0.7",
  user="root",
  password="codesnroses",
  database="load_testing",
  autocommit=False
)

print(connection)

sql = "INSERT INTO Person (first_name, last_name, company_name, address, city, county, state, zip, phone1, phone2, email, web)\
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
#rec = ("James","Butt","Benton, John B Jr","6649 N Blue Gum St",\
#       "New Orleans","Orleans","LA","70116","504-621-8927","504-845-1427","jbutt@gmail.com","http://www.bentonjohnbjr.com")
cursor = connection.cursor()
numRecords = 0
filename = "us-1000000.csv"
# small batch
batch_size = 5000
records = []
totalNumRecords = 0

with open(filename, "r") as csvfile:
    datareader = csv.reader(csvfile)
    next(datareader) #skip first row of column headings
    for row in datareader:

        try:
            records.append(row)
            numRecords += 1
            if numRecords == batch_size:
                ## Change this to take in batch inserts
                cursor.executemany(sql, records)
                connection.commit()
                totalNumRecords += numRecords
                numRecords = 0
                records = []
                print(totalNumRecords, " Number Record inserted successfully into Person table")

        except mysql.connector.Error as error:
            print("Failed to insert record into Person table {}".format(error))

        finally:
            if connection.is_connected() and totalNumRecords == 1000000:
                connection.close()
                print("MySQL connection is closed")
                !mysql load_testing -e "select count(*) from Person"
```

<u>Figure 2: Code block including batch insert method</u>

Section 4: Hardware Upgrade Runs (Milestone 2):

We upgraded our hardware from 8 CPUs to 30 CPUs, changed our N1 series to a C2 series, and increased our RAM to 120GB. We used the same code block as seen in Figure 2, and had a recorde time of 2min 9s +- 5.8s per loop. After this we upgraded to a high-network bandwidth and reran the code block and got a time of 2min 8s +- 3.33s per loop. There was not too much of a difference between the individual hardware upgrades (higher network bandwidth id 1 sec faster), but it was reasonably faster than the normal hardware associated with multiple batch inserts.

Section 5: Potential Future Improvements:

If the timeline allowed, we would test the latency, throughput, and cost with a read + write workload with a MySQL database system and a NoSQL database system. Following that a presentation could be made to compare and contrast the results.