~ ~

A Jasim Azizi ~ UTCS 327E - Databases

Professor Shirley Cohen

TA:- Sameer, Karan

Final Project
Milestone One
MySQL

SECTION ONE: OVERVIEW
Background Information on the Project

I assessed the write capability for a database management system. The evaluation was around MySQL's compute engine, in which case I used an instance to launch it and make the proper commands to achieve my results. There were a total of 1 million records inserted into a table called Person which is in a database called load_testing.

The results were astounding. The average time and the standard deviation were found as a result. Batch insert techniques were used in order to rapid the efficiency at which the data is being loaded into the load_testing database. System variable turning as lll as hardware resources Are used in order to fill the Person table.

My goal was to increase the total capability threshold levels for the central processing unit (CPU) and the Random Access Memory (RAM) in order to assess the interval change of time as opposed to the same 1 million data records being loaded. As a result I found that this project helped me learn quite a lot about how to evaluate a database system. The goal and hope is that this can help me expand my database management skills into other systems.

SECTION TWO: Baseline Run (Milestone 1)

Short Description
Time Results
Observations/Challenges

I used Jupyter Notebook along with Python to load the data into my milestone1.ipynb project. A total of 8 million records were loaded into the Person table. The records were inserted separately from each other. The commits were done at 5000 record points. There are more comments and explanations in the .ipynb file on how insertions were nested using for loops. Below is the code which I used to implement this logic.

```
%%timeit
import mysql.connector

# connecting to mysql DB
connection = mysql.connector.connect(
  host="10.128.0.3",
  user="root",
  password="data",
  database="load_testing",
  autocommit=False
)

print("Connection to MySQL \t ", connection)
print("\n")

sqlData = "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)"

# to keep track of the cursor
counterCursor = 0

# Per question, as commented above
# TRY
try:
    for n, row in df_us_1000000.iterrows():

        record = (row['first_name'], row['last_name'], row['company_name'], row['address'], row['city'],\
               row['county'], row['state'], row['zip'], row['phone1'], row['phone2'], row['email'], row['web'])
        cursor = connection.cursor()
        cursor.execute(sqlData, record)
        counterCursor += cursor.rowcount

        if (n + 1) % 5000 == 0:
            connection.commit()
            print(counterCursor, "records inserted successfully into Person table")
            cursor.close()
# EXCEPT
except mysql.connector.Error as error:
    print("Unsuccessful to insert record(s) into Person table {}".format(error))

# FINALLY
finally:
    if connection.is_connected():
        connection.close()
        print("MySQL connection is closed")
```

```
Connection to MySQL       <mysql.connector.connection_cext.CMySQLConnection object at 0x7f0d22c33d50>
```

SECTION THREE: Batch Runs (Milestone 1)

Upon attaining my average time as lll as standard deviation of the baseline run, I implemented a method called *executemany()* where instead of inserting 1 record at a time, I inserted batches of 5,000, 50,000, and 500,000 records at a time. This improved the efficiency of the record inserts and I found the following average time and standard deviation for each of the batch runs below.

The results I observed are below:

- 500,000 batch 30CPUs server:

1min 50s ± 951 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

- **500,000 batch 30CPUs server (*Higher Bandwidth*):**

1min 52s ± 735 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

SECTION FOUR: Hardware upgrade runs (Milestone)

Upon testing the batch runs method, I closed the instance and upgraded the database hardware capabilities. I updated our current server in which the CPU limit was 8 and I upgraded it to 30 CPUs. For our second Hardware upgrade, I created a new server that has high bandwidth capabilities. I ran different servers using the 500,000 record batch insert method and recorded the average time and standard deviation. The hardware upgrade from 8 CPUs to 30 CPUs changed the overall speed of inserting records but it was not sufficient enough in my opinion. However, after adding more higher bandwidth, the insertions to the table *Person* was noticeably faster.

SECTION FIVE: Potential future improvements

It’s possible to expand the project into newer horizons. By inserting more data into the table, I can time more of our efficiency and support for the team and customers. Evaluating a MySQL database and also comparing average time and standard deviation for the inserts to another database management system. More time would be necessary but also better hardware is important for much better efficiency. I could continue evaluating different database systems but with more data records. Bigger sample sizes of multiple millions would be ready for testing.

Thank you!