HOMEWORK 3 -- Question 3

Author:

- Fang Han Cabrera (fh643@nyu.edu)

Folder Structure

- **driver.py** contains python script for the experiment
- **/timings** contains all the records of timings for each tables
- **/data** contains data input file *emp*
- **/image** contains screenshots

Setup

libraries used

- mysql.connector
- numpy

environment

All the python script included in *driver.py* is executed in **ipython**

mysql setup

Before establishing connection to *mysql* server using **mysql.connector**, we need to create databases and their corresponding indices in *mysql*.

I created 4 databases, described below:

1. Vanilla table **without** index, as required by 3(a). `create table Employee(ID int, Name varchar(20), Salary int, Manager int, Department varchar(20));`
2. Table with **unique** index on *ID* and a **secondary** index on *department*, as required by 3(b).

```
create table Emp_test(ID int, Name varchar(20), Salary int, Manager int, Department v
ALTER TABLE Emp_test ADD UNIQUE (ID);
ALTER TABLE Emp_test ADD INDEX (Department);
```

The resulting table is checked as shown as below:

```
mysql> describe Emp_test
    -> ;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| ID         | int(11)     | YES  | UNI | NULL    |       |
| Name       | varchar(20) | YES  |     | NULL    |       |
| Salary     | int(11)     | YES  |     | NULL    |       |
| Manager    | int(11)     | YES  |     | NULL    |       |
| Department | varchar(20) | YES  | MUL | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

3. An extra table with **BTREE** index on *ID* for comparison.

```
create table Emp_btree(ID int, Name varchar(20), Salary int, Manager int, Department
CREATE INDEX emp_btree_ind ON Emp_btree (ID) USING BTREE;
```

4. Another extra table with **HASH** index on *ID* for comparison.

```
create table Emp_hash(ID int, Name varchar(20), Salary int, Manager int, Department v
CREATE INDEX emp_hash_ind ON Emp_hash (ID) USING HASH;
```

Experiment

code explanation

global

- establish connection to mysql with `cnx = mysql.connector.connect(user, password, host, database)` command.
- create cursor with `cursor = cnx.cursor()` .
- lists to record all timings in lists, e.g. `timings1 = []`
- IO setup, e.g. `f = open("timings_no_index.txt","w+")`

per table

For each of the tables created above, I did the following:

- specify command to be passed onto mysql with table name and schema, e.g.

```
command = ("INSERT INTO <table name> "
           "(ID, Name, Salary, Manager, Department) "
           "VALUES (%s, %s, %s, %s, %s)")
```

- read from **emp** line by line into `data` while passing the resulting command to mysql:

```
cursor.execute(command, data)
```

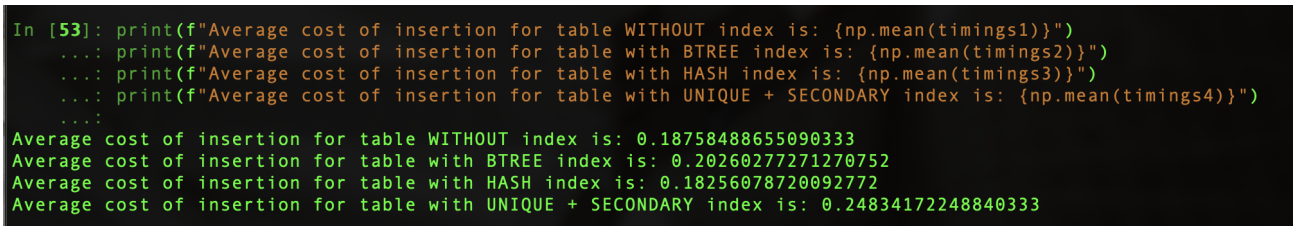
- record time elapsed with python `time.time()` .
- write time to file using `f.write()`

Result

To compare time cost of insertion into each of the tables, I took the mean of all their respective timings recorded, e.g. for the table *without* index: `print(f"Average cost of insertion for table WITHOUT index is: {np.mean(timings1)}")`

numerical

I conducted two experiments whose results are shown below:

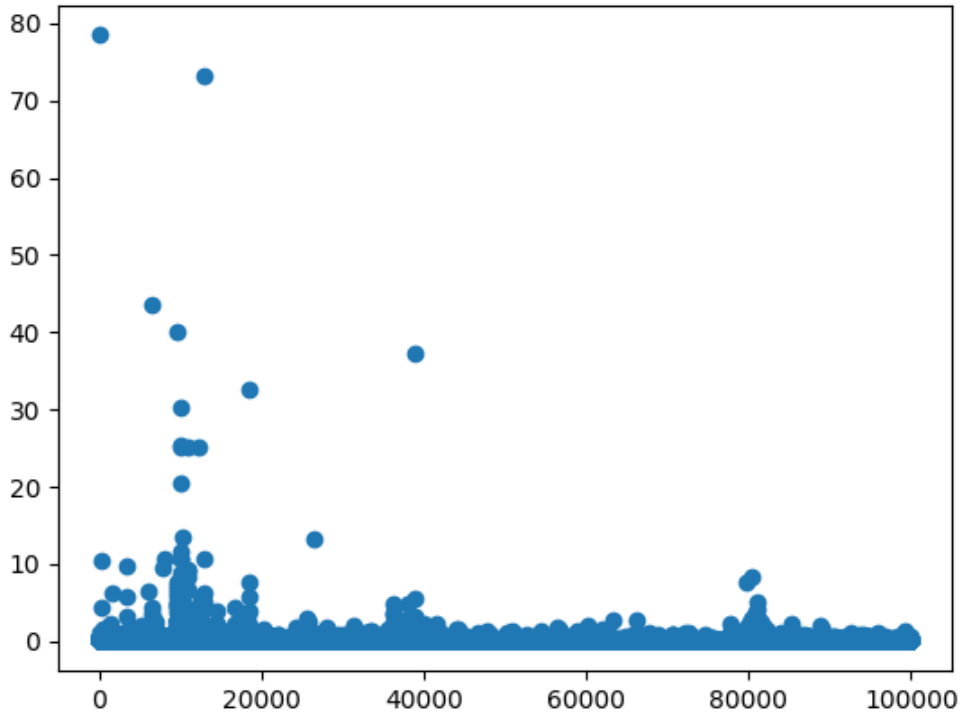
1. 
```
In [53]: print(f"Average cost of insertion for table WITHOUT index is: {np.mean(timings1)}")
    ...: print(f"Average cost of insertion for table with BTREE index is: {np.mean(timings2)}")
    ...: print(f"Average cost of insertion for table with HASH index is: {np.mean(timings3)}")
    ...: print(f"Average cost of insertion for table with UNIQUE + SECONDARY index is: {np.mean(timings4)}")
    ...:
Average cost of insertion for table WITHOUT index is: 0.18758488655090333
Average cost of insertion for table with BTREE index is: 0.20260277271270752
Average cost of insertion for table with HASH index is: 0.18256078720092772
Average cost of insertion for table with UNIQUE + SECONDARY index is: 0.24834172248840333
```
2. 
```
    ...:
Average cost of insertion for table WITHOUT index is: 0.16650681018829347
Average cost of insertion for table with BTREE index is: 0.18191001892089845
Average cost of insertion for table with HASH index is: 0.25071014881134035
Average cost of insertion for table with UNIQUE + SECONDARY index is: 0.17405854940414428
```

It can be seen that insertion into a table with **UNIQUE + SECONDARY** table costs on average more than a table without **INDEX**.

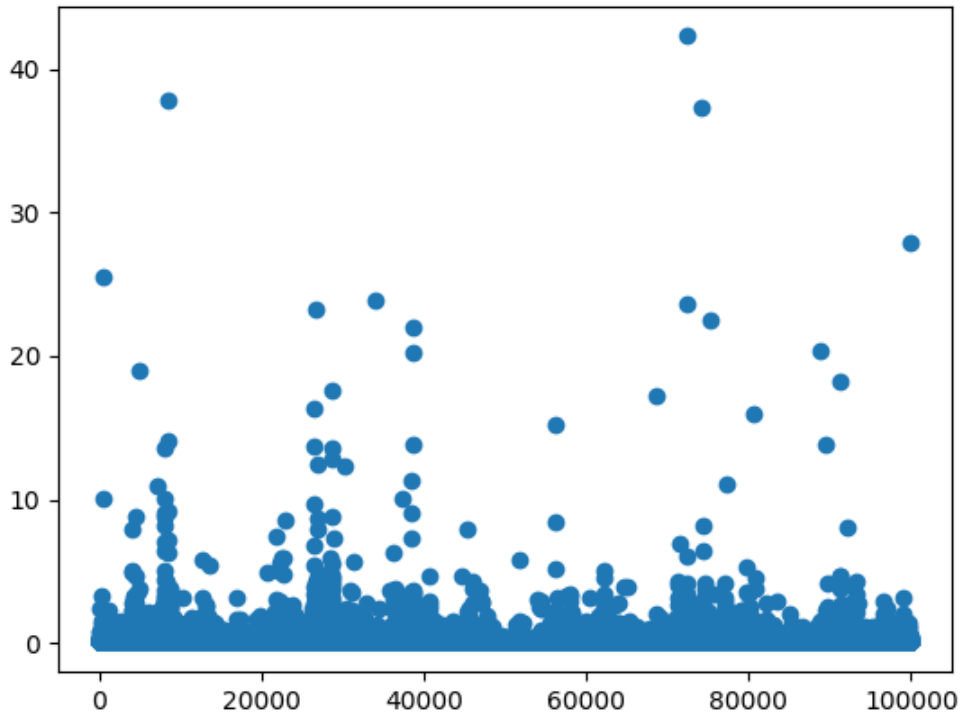
plot

If we plot the timings of inserting into these two tables, some patterns can be observed:

1. table **WITHOUT** index:
    - the spikes are clustered in the front.



1. table with **UNIQUE + SECONDARY**
    - the spikes are more scattered in a random fashion.



explanation

- With clustered (**unique**) index, the primary key and the record itself are “clustered” together, and the records are all stored in primary-key order.
- But under some circumstances, the clustered index can actually degrade performance. When you use one together with a **secondary** index, the secondary indices point to the primary key rather than the row. Therefore, you will end up with duplicate copies of that primary index, first as the clustered index stored alongside the records themselves, but then again for as many times as you have secondary indexes pointing to those clustered indexes.
- This repeated storage of the primary key plus the overhead of sorting the primary key may have contributed to the longer insertion time.