MANAGING & PROGRAMMING DATABASE

Trishla Shah

ASSIGNMENT 03

11/11/2018

Group Members:

Mohd Nawaz Hussain (A00428036)

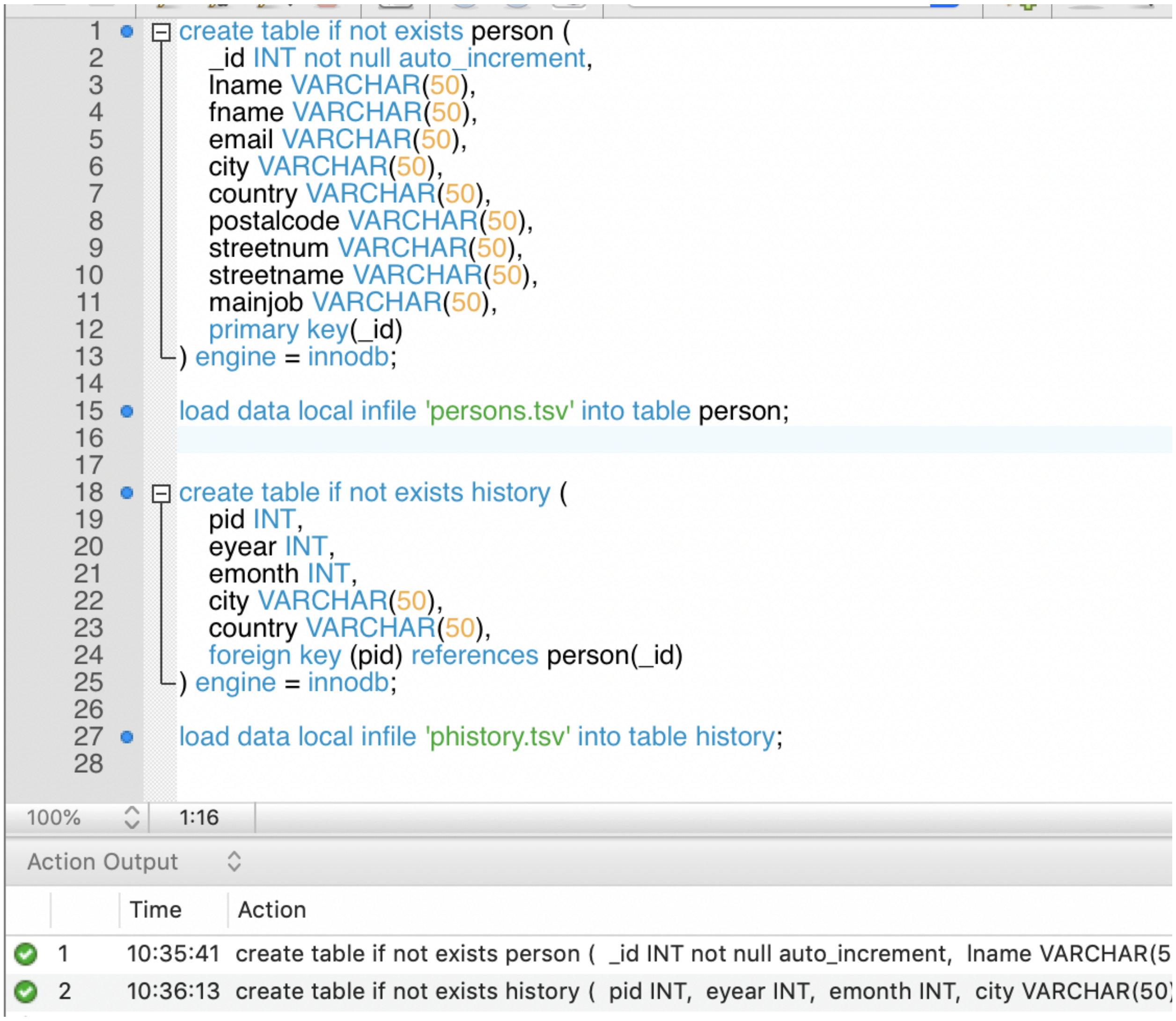
Bhagya Sharma (A00431152)

Rishab Gupta (A00429019)

Madeleine Leong (A00430926)

Table of Contents

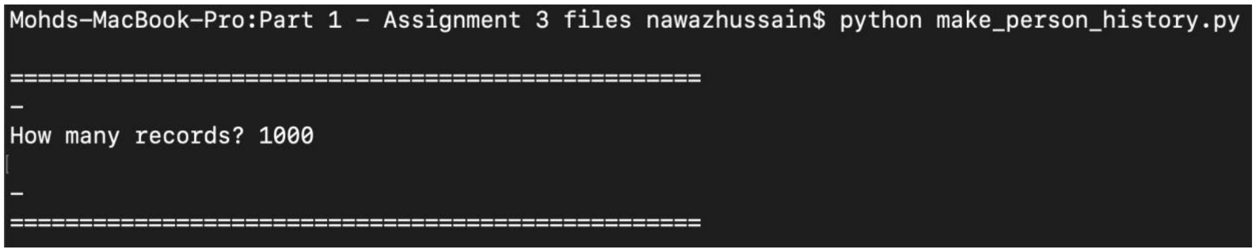
Creating the Tables



```
create table if not exists person (
    _id INT not null auto_increment,
    lname VARCHAR(50),
    fname VARCHAR(50),
    email VARCHAR(50),
    city VARCHAR(50),
    country VARCHAR(50),
    postalcode VARCHAR(50),
    streetnum VARCHAR(50),
    streetname VARCHAR(50),
    mainjob VARCHAR(50),
    primary key(_id)
) engine = innodb;

load data local infile 'persons.tsv' into table person;


create table if not exists history (
    pid INT,
    eyear INT,
    emonth INT,
    city VARCHAR(50),
    country VARCHAR(50),
    foreign key (pid) references person(_id)
) engine = innodb;

load data local infile 'phistory.tsv' into table history;
```

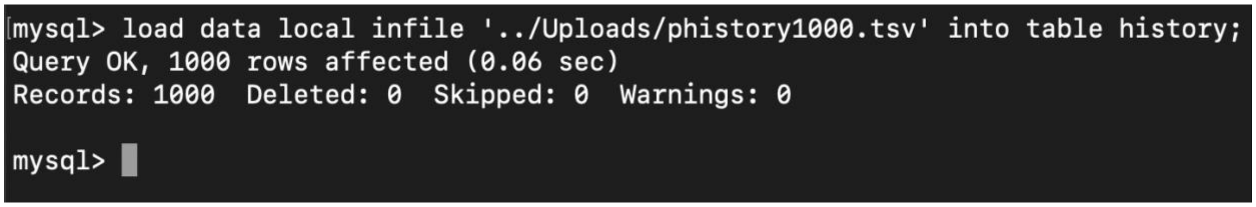
```
     Time      Action
1    10:35:41  create table if not exists person (  _id INT not null auto_increment,  lname VARCHAR(5
2    10:36:13  create table if not exists history (  pid INT,  eyear INT,  emonth INT,  city VARCHAR(50)
```

Make Records



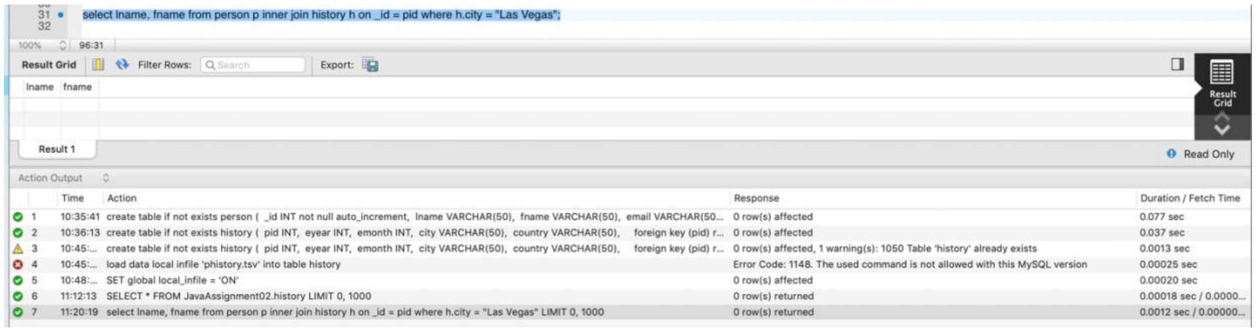
```
Mohds-MacBook-Pro:Part 1 - Assignment 3 files nawazhussain$ python make_person_history.py

==================================================
-
How many records? 1000

-
==================================================
```

Load Records Without Index (1000 Records)



```
mysql> load data local infile '../Uploads/phistory1000.tsv' into table history;
Query OK, 1000 rows affected (0.06 sec)
Records: 1000  Deleted: 0  Skipped: 0  Warnings: 0

mysql> 
```

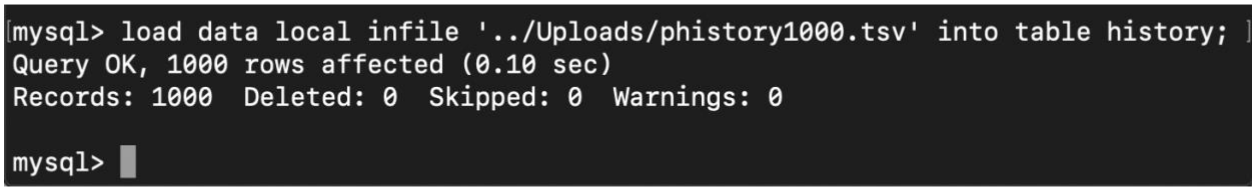
Run On 1000 Records Without Index



```
select lname, fname from person p inner join history h on _id = pid where h.city = "Las Vegas";
```

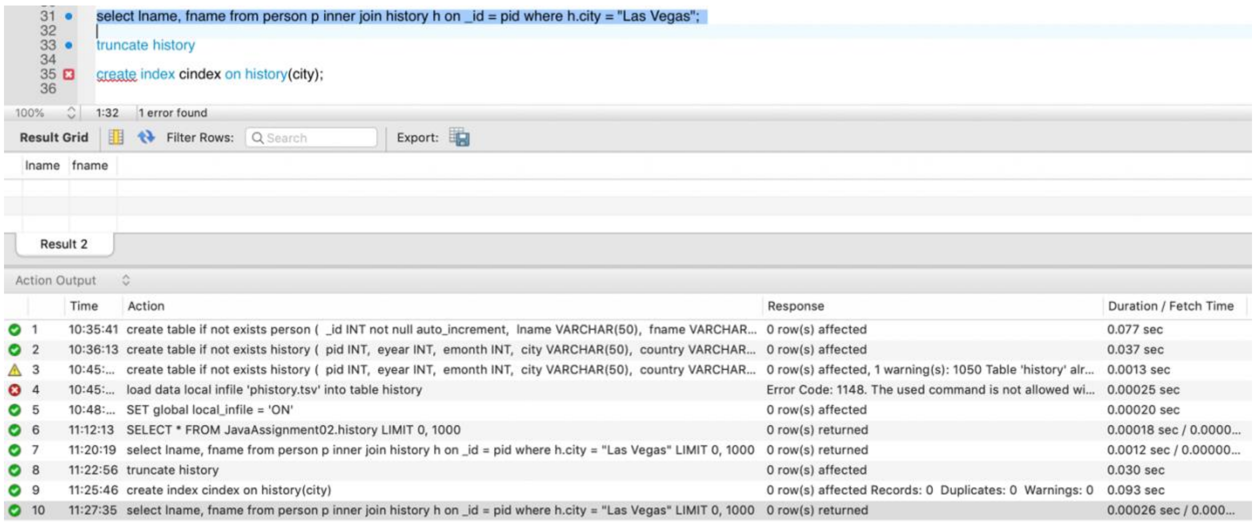
		Time	Action	Response	Duration / Fetch Time
✓	1	10:35:41	create table if not exists person (_id INT not null auto_increment, lname VARCHAR(50), fname VARCHAR(50), email VARCHAR(50...	0 row(s) affected	0.077 sec
✓	2	10:36:13	create table if not exists history (pid INT, eyear INT, emonth INT, city VARCHAR(50), country VARCHAR(50), foreign key (pid) r...	0 row(s) affected	0.037 sec
⚠	3	10:45:...	create table if not exists history (pid INT, eyear INT, emonth INT, city VARCHAR(50), country VARCHAR(50), foreign key (pid) r...	0 row(s) affected, 1 warning(s): 1050 Table 'history' already exists	0.0013 sec
✗	4	10:45:...	load data local infile 'phistory.tsv' into table history	Error Code: 1148. The used command is not allowed with this MySQL version	0.00025 sec
✓	5	10:48:...	SET global local_infile = 'ON'	0 row(s) affected	0.00020 sec
✓	6	11:12:13	SELECT * FROM JavaAssignment02.history LIMIT 0, 1000	0 row(s) returned	0.00018 sec / 0.0000...
✓	7	11:20:19	select lname, fname from person p inner join history h on _id = pid where h.city = "Las Vegas" LIMIT 0, 1000	0 row(s) returned	0.0012 sec / 0.00000...

Load Records with Index (1000 Records)



```
mysql> load data local infile '../Uploads/phistory1000.tsv' into table history;
Query OK, 1000 rows affected (0.10 sec)
Records: 1000  Deleted: 0  Skipped: 0  Warnings: 0

mysql> 
```

Run On 1000 Records with Index



```
select lname, fname from person p inner join history h on _id = pid where h.city = "Las Vegas";

truncate history

create index cindex on history(city);
```

```
100%   1:32   1 error found
```

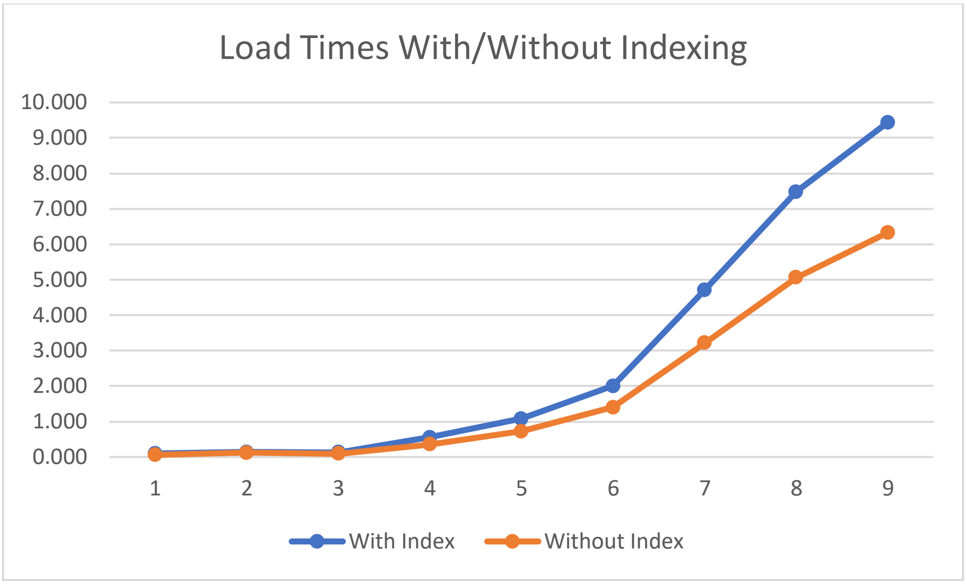
```
    Time      Action                                                                                        Response                                                          Duration / Fetch Time
1   10:35:41  create table if not exists person (  _id INT not null auto_increment,  lname VARCHAR(50),  fname VARCHAR...  0 row(s) affected                                     0.077 sec
2   10:36:13  create table if not exists history (  pid INT,  eyear INT,  emonth INT,  city VARCHAR(50),  country VARCHAR...  0 row(s) affected                                     0.037 sec
3   10:45:...  create table if not exists history (  pid INT,  eyear INT,  emonth INT,  city VARCHAR(50),  country VARCHAR...  0 row(s) affected, 1 warning(s): 1050 Table 'history' alr...  0.0013 sec
4   10:45:...  load data local infile 'phistory.tsv' into table history                                          Error Code: 1148. The used command is not allowed wi...  0.00025 sec
5   10:48:...  SET global local_infile = 'ON'                                                                   0 row(s) affected                                     0.00020 sec
6   11:12:13  SELECT * FROM JavaAssignment02.history LIMIT 0, 1000                                              0 row(s) returned                                     0.00018 sec / 0.0000...
7   11:20:19  select lname, fname from person p inner join history h on _id = pid where h.city = "Las Vegas" LIMIT 0, 1000  0 row(s) returned               0.0012 sec / 0.00000...
8   11:22:56  truncate history                                                                                  0 row(s) affected                                     0.030 sec
9   11:25:46  create index cindex on history(city)                                                              0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  0.093 sec
10  11:27:35  select lname, fname from person p inner join history h on _id = pid where h.city = "Las Vegas" LIMIT 0, 1000  0 row(s) returned               0.00026 sec / 0.000...
```

Note: Done the same steps for the next iterations all the way up to 1000000 records. Did not put screenshots as the results are summarized in the tables below.

Load Records in Table with Increasing No. of Records

No of Records	**With Index**	**Without Index**
1000	0.100	0.060
5000	0.140	0.120
10000	0.130	0.090
50000	0.550	0.360
100000	1.080	0.720
200000	2.000	1.400
500000	4.710	3.220
800000	7.480	5.060
1000000	9.440	6.330

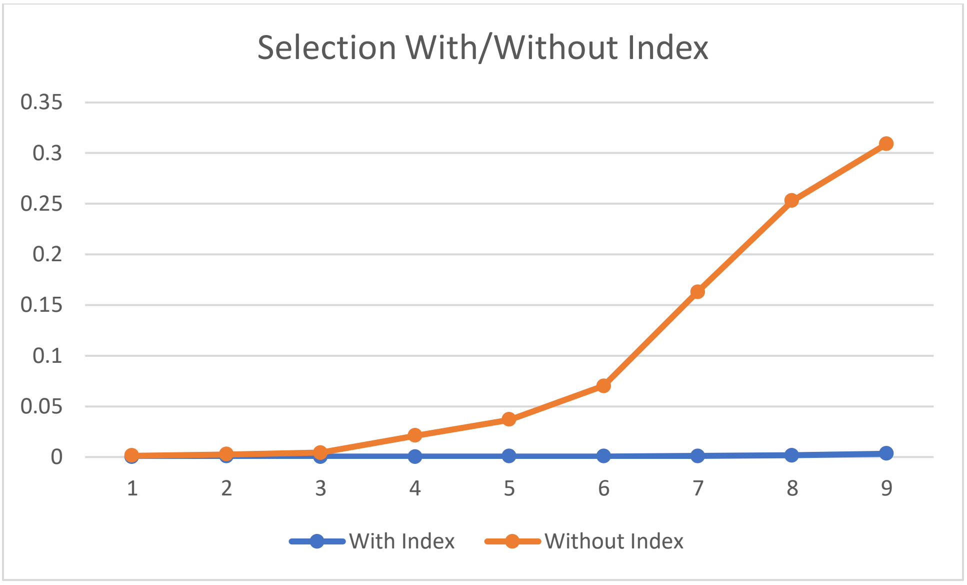
Time is in seconds.



Select Records in Table with Increasing No. Of Records

No of Records	With Index	Without Index
1000	0.00026	0.0012
5000	0.00084	0.0027
10000	0.00038	0.0041
50000	0.00039	0.021
100000	0.00068	0.037
200000	0.00059	0.07
500000	0.00096	0.163
800000	0.0016	0.253
1000000	0.0032	0.309

Time is in seconds.



Conclusions

From the experiment, we can understand the following two points about insertion and selection of data with respect to indexing

- With indexing, the time to insert data into a table increases versus a non-indexed table
- With indexing, the time to select or retrieve data becomes faster than a non-index table

The reason behind this is because in a non-indexed table, the database stores the row in a regular heap. That is, no particular row order and the database stores it wherever there is free space. However, in indexing, the database technology has to ensure the data is entered via the indexes that already exist. Adding an entry into an index is much more resource consuming and takes more time than inserting into a heap structure. This is because the database has to keep the index order and tree balance. The new data cannot just be written to any block but has to belong to a specific leaf node. Once the correct node is identified, the database checks to see if there is enough space on the leaf. If not, it splits the lead node and inserts the entries between the new and old node. Therefore, the insertion process takes more time in an indexed table. According to Markus Winand of 'Use the Index Luke', adding an additional index is enough to increase the execution time by a factor of hundred for insertion (Winand, 2017). In selection however, Indexing drastically improves the performance over a non-indexed table. As seen above in the graph, as number of data has increased, the performance of the index was dampened slightly versus that of a non-indexed table. This is because since an index exists, the selection is quicker as it knows where to find the data versus going through each record and selecting those that meet the criteria. Indexes in database e contains only sorted values, due to which the selection by searching becomes faster. The binary search algorithm is used to search from index which makes the searching even faster

References

- M. Winand. 2017. Insert. [ONLINE] Available at: https://use-the-index-luke.com/sql/dml/insert. [Accessed 11 November 2018].
- Slashroot.in. 2018. How does indexing makes database faster. [ONLINE] Available at: https://www.slashroot.in/how-does-indexing-makes-database-faster. [Accessed 12 November 2018].