**Batch: A1 Roll No.: 1611019**

**Experiment No. 03**

**Grade: AA / AB / BB / BC / CC / CD /DD**

|  |
| --- |
| **Title: Database Tuning** |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** Tuning the database to improve system performance

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO1 :** Design and tune database.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. *Elmasri & Navathe “ fundamentals of Database Systems” V edition. PEARSON Education.*
2. *Korth, Silberschatzsu darshan “Database systems, concepts” 5th edition McGraw Hill.*
3. *Raghu Ramkrishnan & Johannes Gehrke “Database Management System” Tata McGraw Hill. III edition.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**Pre Lab/ Prior Concepts:** Database, ER diagram, Relation mapping, SQL

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Implementation Details:**

Perform following;

1. Index tuning
2. Query Tuning
3. Database Tuning

And mention why tuning is required? How it is performed and what is the effect of tuning?

for all mentioned three categories of tuning.

**1. Index Tuning: What:**

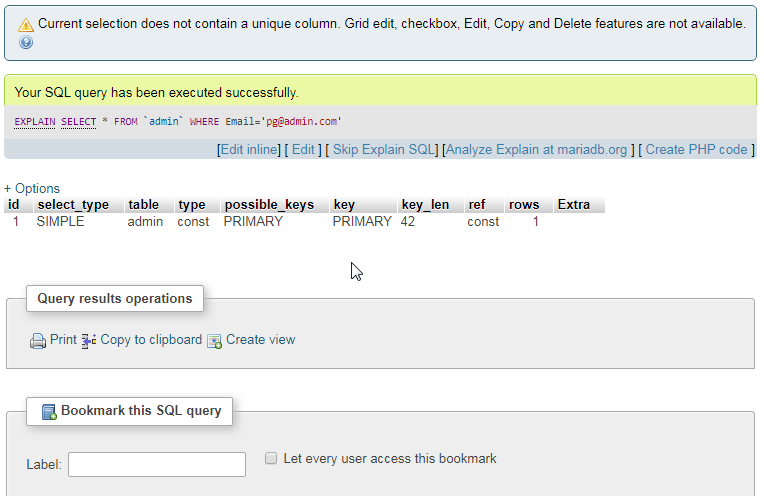
* Query performance as well as speed improvement of a database can be done using Indexes.
* The process of enhancing the selection of indexes is called Index Tuning.
* Index tuning is part of database tuning for selecting and creating indexes. The index tuning goal is to reduce the query processing time.

**Why is index tuning done and effects of index tuning:**

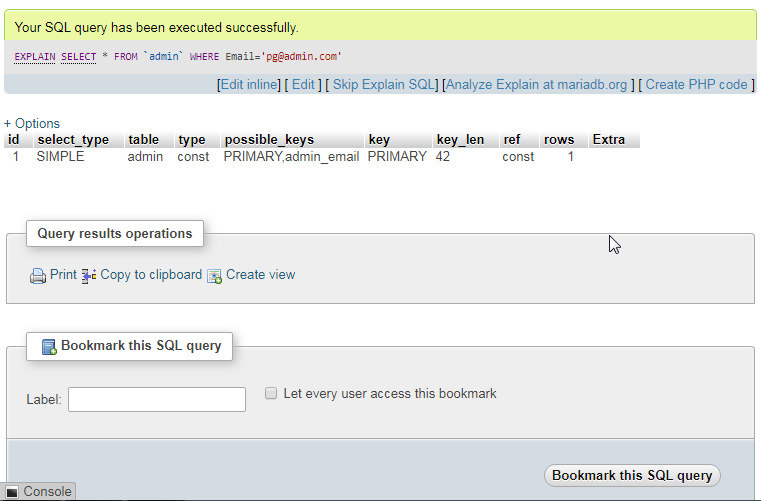
* Effective indexes are one of the best ways to improve performance in a database application.
* A table scan happens when there is no index available to help a query. In a table scan SQL Server examines every row in the table to satisfy the query results.
* Table scans are sometimes unavoidable, but on large tables, scans have a terrific impact on performance.
* One of the most important jobs for the database is finding the best index to use when generating an execution plan.

Example:

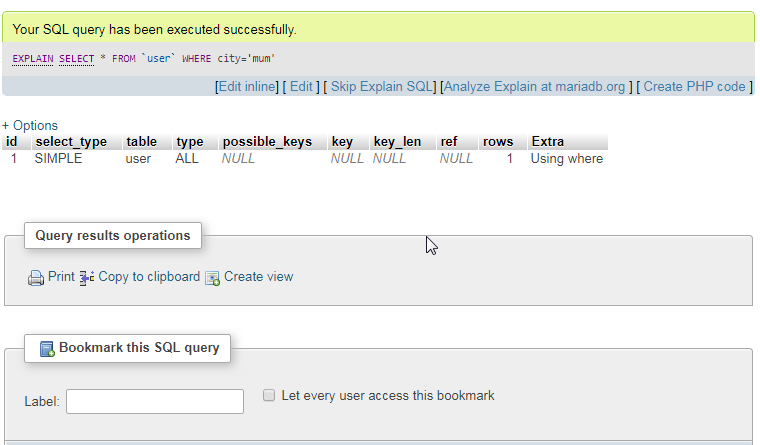
On admin table of our database Email is the primary key

Before indexing: Index is not used for getting the result of a query

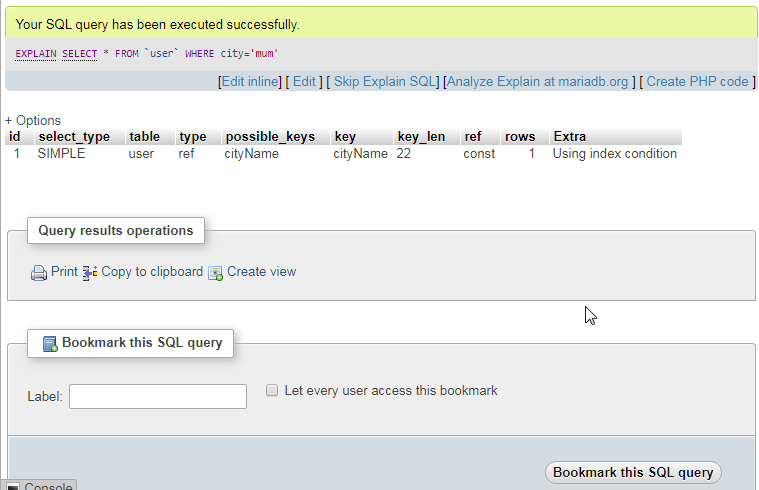
After indexing: Indexing on admin\_email



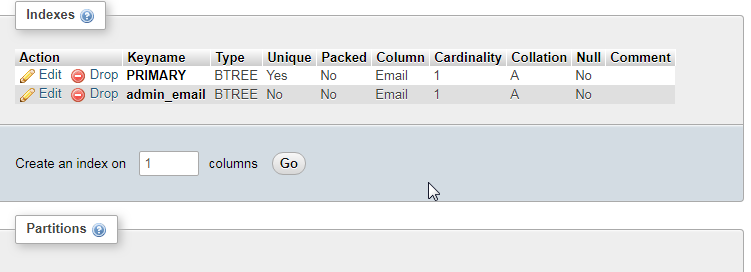
Before indexing: Query result obtained using where condition

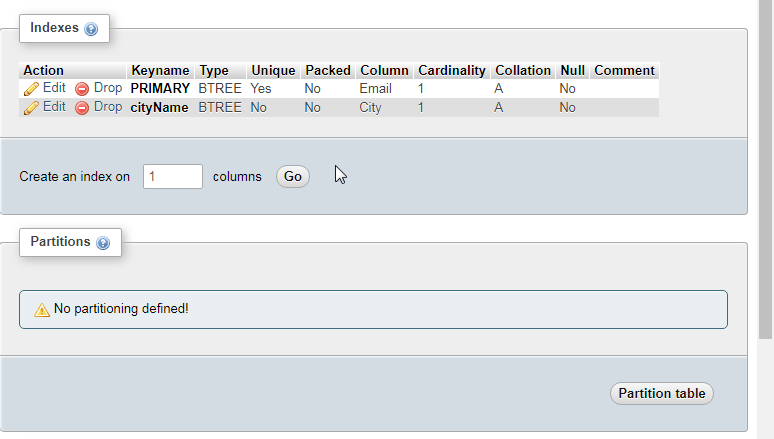


After indexing: Query result obtained using index condition

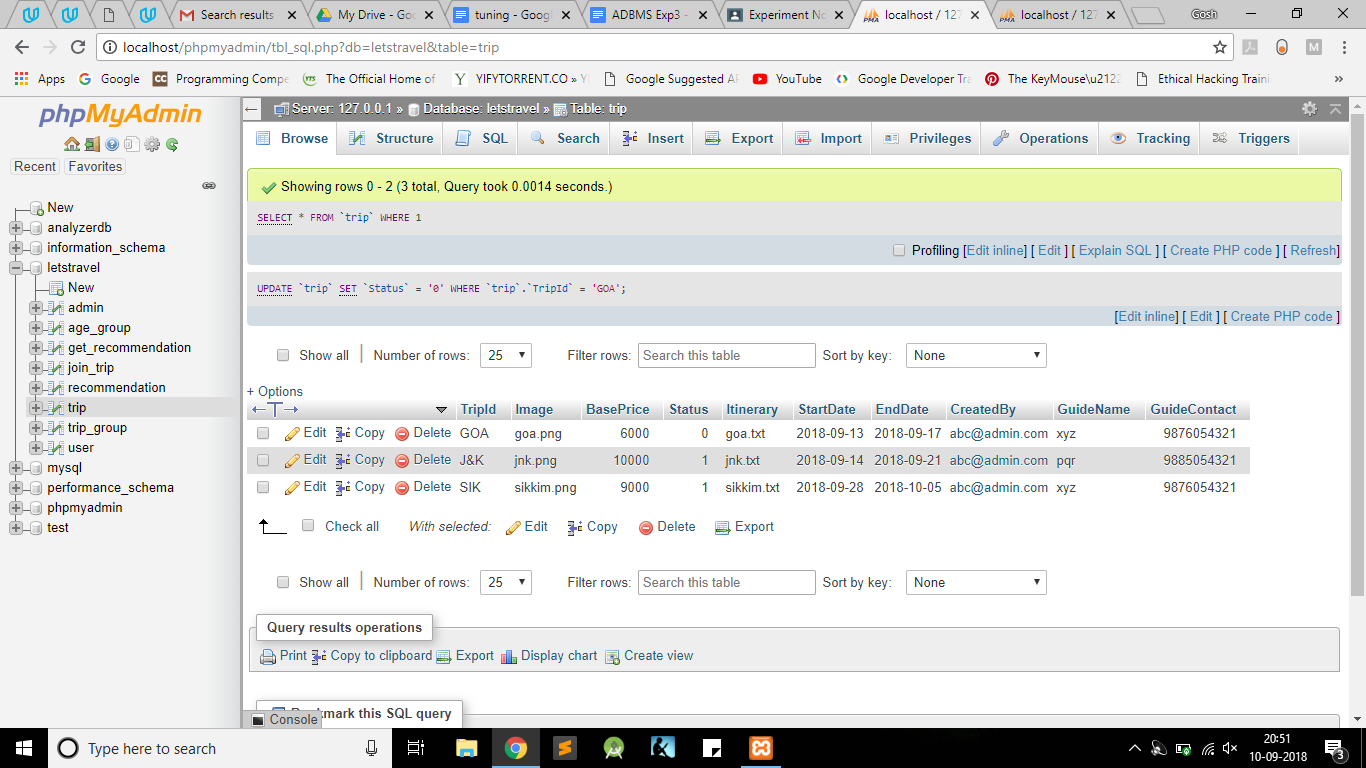
Create index cityName on user(city);

Indexes created:

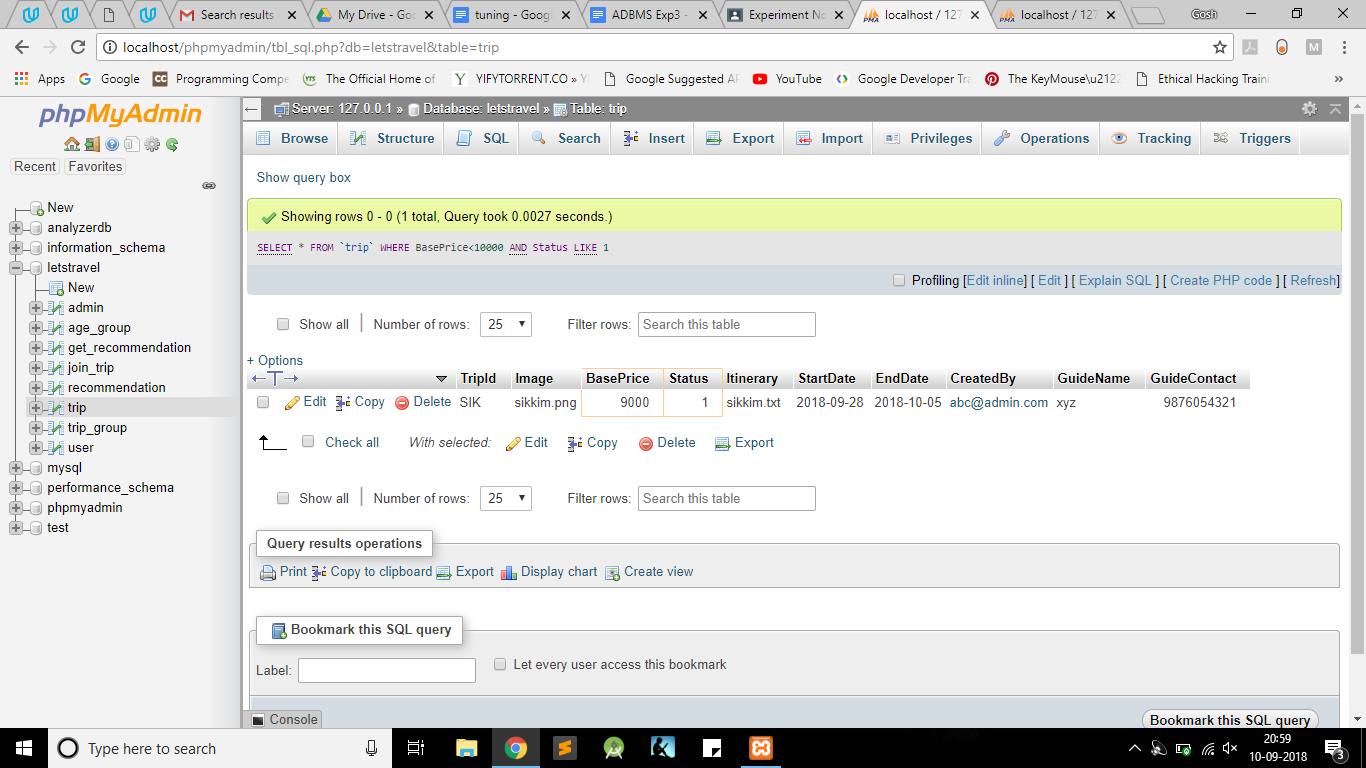


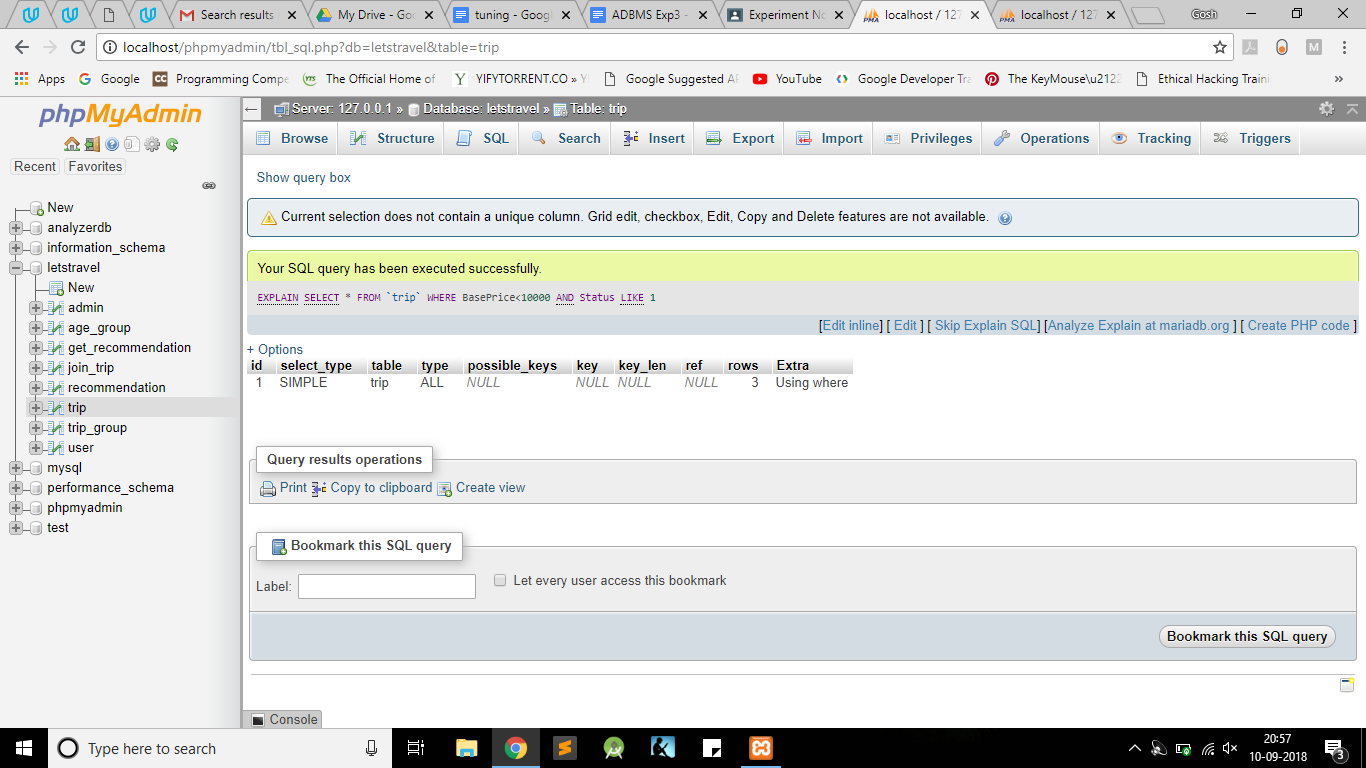


Trip table:



Query: Select \* from trip where BasePrice<10000 And Status Like 1

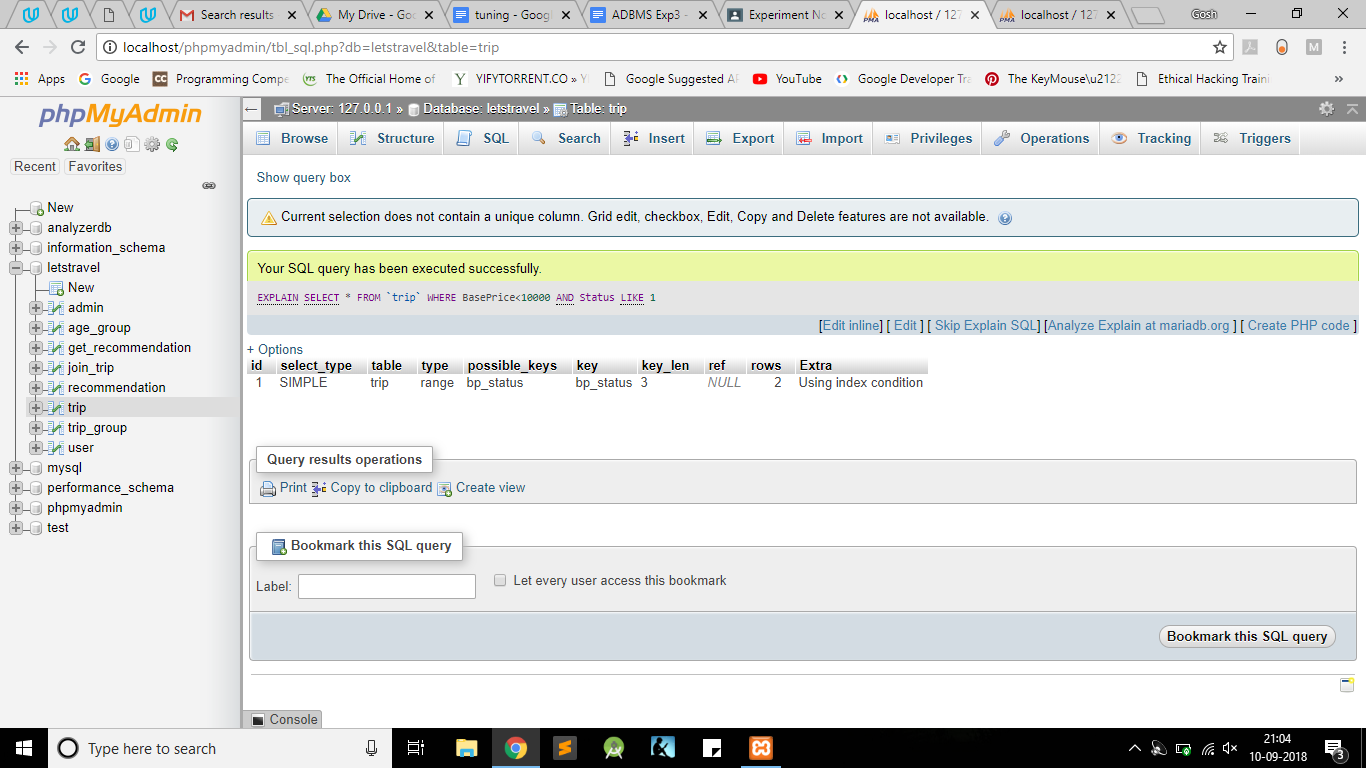
Before indexing Query result obtained using where condition

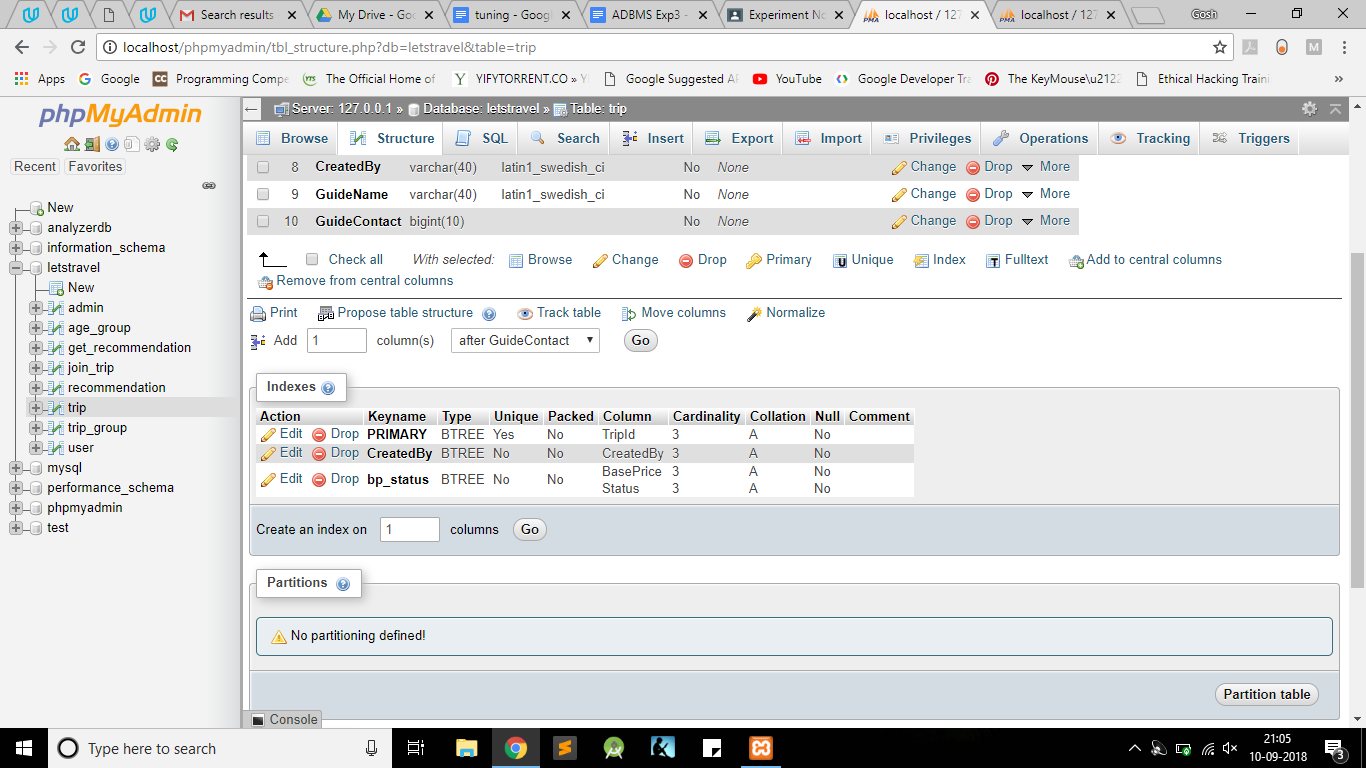


After indexing: Query result obtained using indexing condition

CREATE INDEX bp\_status on trip(BasePrice,Status)

EXPLAIN SELECT \* FROM `trip` WHERE BasePrice<10000 AND Status LIKE 1



Index in the trip table:

**2. Query Tuning**

**What is query tuning?**

Sql Statements are used to retrieve data from the database. We can get same results by writing different sql queries.

But, different queries require different time to retrieve results, some more than other. Query tuning involves optimizing a query in order to reduce the number of intermediate steps results and reduce access time.

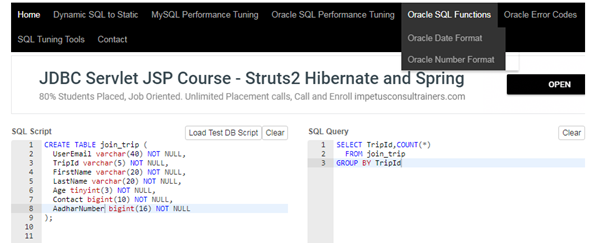
**Why is query tuning used?**

Query tuning can be used for the following:

* To reduce the response time for end users of the system.
* To reduce the resources used to process the same work.

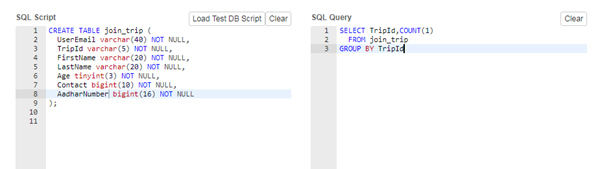
**How is it performed?**

**1. Before tuning:**

****

****

**After tuning:**

****

**Reason:**

Count(\*) is used when there is no primary key in the table. So it traces all the column of a table and records to fetch the column count.

Where as Count(1) is used when table has a primary key in the table. So it traverses only that column while computing the count.

So in this manner the count(1) is faster than the count(\*) .

**2. Before tuning:**

SELECT \* FROM USER

**After tuning:**

SELECT FirstName, LastName, Email, Password, Mobile, City, Rate, Review FROM USER

**Reason:**

When nested queries are used only specific columns are required. Hence select column\_name works better than select \*.

**3. Before tuning:**

SELECT \*

FROM USER, JOIN\_TRIP

WHERE USER.EMAIL = JOIN\_TRIP.EMAIL

**After tuning:**

SELECT \*

FROM USER u, JOIN\_TRIP j

WHERE u.EMAIL = j.EMAIL

**Reason:**

Alias can be used to access same column names from different tables.

**4. Before tuning:**

SELECT LOCATION

FROM AGE\_GROUP

WHERE AgeGroup = ’10-20’ OR AgeGroup = ’20-30’ OR AgeGroup = ’50-60’

**After tuning:**

SELECT LOCATION

FROM AGE\_GROUP

WHERE AgeGroup IN (’10-20’,’20-30’,’50-60’)

**Reason:**

if the values are constant IN sorts the list and then uses a binary search. OR evaluates them one by one in no particular order. So IN is faster in some circumstances.

**5. Before tuning:**

SELECT LOCATION

FROM AGE\_GROUP a

WHERE EXISTS (

SELECT AgeGroup FROM GET\_RECOMMENDATION g WHERE UserEmail =’parth.s@gmail.com’ AND a.AgeGroup = g.AgeGroup

)

**After tuning:**

SELECT LOCATION

FROM AGE\_GROUP

WHERE AgeGroup in (

SELECT AgeGroup FROM GET\_RECOMMENDATION WHERE UserEmail =’parth.s@gmail.com’

)

**Reason:**

The EXISTS clause is much faster than IN when the subquery results is very large. Conversely, the IN clause is faster than EXISTS when the subquery results is very small.

**6. Before tuning:**

SELECT \*

FROM USER, JOIN\_TRIP

WHERE CONCAT(‘(’, USER.EMAIL,’)’) = CONCAT(‘(’, JOIN\_TRIP.EMAIL,’)’)

**After tuning:**

SELECT \*

FROM USER, JOIN\_TRIP

WHERE USER.EMAIL = JOIN\_TRIP.EMAIL

**Reason:**

Concatenation will lead to overhead and wastage of time, instead using equals

directly will be faster.

**7. Before tuning**

SELECT COUNT (1)

FROM TRIP

WHERE BasePrice<= 15000;

SELECT COUNT (1)

FROM TRIP

WHERE BasePrice BETWEEN 15000 AND 30000;

**After tuning:**

SELECT COUNT (CASE WHEN BasePrice <= 15000

THEN 1 ELSE null END) count\_1,

COUNT (CASE WHEN BasePrice BETWEEN 15000 AND 30000

THEN 1 ELSE null END) count\_2

FROM TRIP;

**Reason:**

SQL Performance Tuning recommends using CASE statements. It is more efficient to run a single MySQL statement, rather than two or more separate MySQL statements.

**8. Before tuning:**

SELECT \*

FROM TRIP

WHERE substr(StartDate,6,7) = 09

**After tuning:**

SELECT \*

FROM TRIP

WHERE StartDate LIKE ‘%09%’

**Reason:**

Like is faster than substring.

**9. Before tuning:**

SELECT u.Email,j.TripId

FROM USER u, JOIN\_TRIP j

WHERE u.EMAIL = j.EMAIL

**After tuning:**

SELECT u.Email,j.TripId

FROM USER u INNER JOIN JOIN\_TRIP j

WHERE u.EMAIL = j.EMAIL

**Reason:**

Cartesian product generates more records than inner join and hence is inefficient to use.

**Effect:** Queries are executed at a faster rate with less overhead for accessing the queries.

**3. Database Tuning**

**What is Database Tuning and why is it used?**

Database Tuning means monitoring and revising the physical database design constantly

which helps in:

■ Applications running faster.

■ Lowering the response time of queries and transactions.

■ Improving the overall throughput of transactions.

Database Tuning can be achieved by:

* Denormalization
* Vertical Partitioning
* Horizontal Partitioning
* Repeating attributes in multiple tables

For recommendation of locations, user will enter category, budget, age group and group of people.

Originally, the tables were as follows:

recommendation

|  |  |  |  |
| --- | --- | --- | --- |
| Location | Category | Link | Budget |

age\_group

|  |  |
| --- | --- |
| Location | AgeGroup |

group

|  |  |
| --- | --- |
| Location | Group |

Whenever we have to find a suitable trip recommendation for the user based on his choices, we’ll have to join the above three tables in order to ensure that all his requirements are satisfied.

Performing join operation every time increases overhead. So, we’ll combine the three tables into one, for faster access.

After database tuning, the new table is:

recommendation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Location | Category | Link | Budget | AgeGroup | Group |

**Effect:**

Now, whenever we have to search any location based on user’s requirements, we can get the results comparatively faster as all the required data is in one table, thus, eliminating the need of intermediate join operation.

**Conclusion:** Database tuning has been done using index tuning and query tuning.

**Post Lab Descriptive Questions:**

1. **What are the factors that influence Physical Database Design ?**

**Ans:** Factors influencing Physical Database Design are:

**A. Analyzing the Database Queries and Transactions**

Before undertaking the physical database design, we must have a good idea of the intended use of the database by defining in a high-level form the queries and transactions that are expected to run on the database. For each retrieval query, the following information about the query would be needed:

1. The files that will be accessed by the query.
2. The attributes on which any selection conditions for the query are specified.
3. Whether the selection condition is an equality, inequality, or a range condition.
4. The attributes on which any join conditions or conditions to link multiple tables or objects for the query are specified.
5. The attributes whose values will be retrieved by the query.

The attributes listed in items 2 and 4 above are candidates for the definition of

access structures, such as indexes, hash keys, or sorting of the file.

For each update operation or update transaction, the following information

would be needed:

1. The files that will be updated.
2. The type of operation on each file (insert, update, or delete).
3. The attributes on which selection conditions for a delete or update are specified.
4. The attributes whose values will be changed by an update operation.

**B. Analyzing the Expected Frequency of Invocation of Queries and Transactions**

* The expected frequency information, along with the attribute information collected on each query and transaction, is used to compile a cumulative list of expected frequency of use for all the queries and transactions.
* It is expressed as the expected frequency of using each attribute in each file as a selection attribute or join attribute, over all the queries and transactions.
* 80-20 rule

20% of the data is accessed 80% of the time

**C. Analyzing the time constraints of queries and transactions**

* Performance constraints place further priorities on the attributes that are candidates for access paths.
* The selection attributes used by queries and transactions with time constraints become higher-priority candidates for primary access structure.

**D. Analyzing the expected frequencies of update operations**

* A minimum number of access paths should be specified for a file that is updated frequently.

**E. Analyzing the uniqueness constraints on attributes**

* Access paths should be specified on all candidate key attributes — or set of attributes — that are either the primary key or constrained to be unique.

**Date: \_\_11/09/2018\_\_**