Batch: B2 Roll No.: 1611087

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:

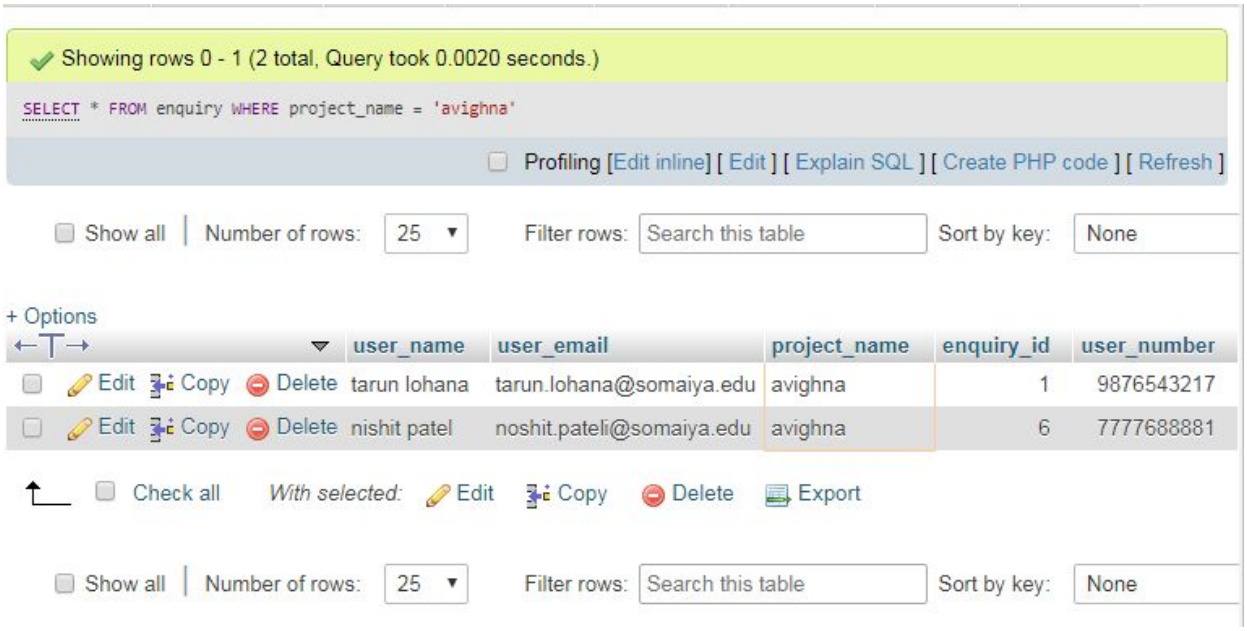
Perform following;

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

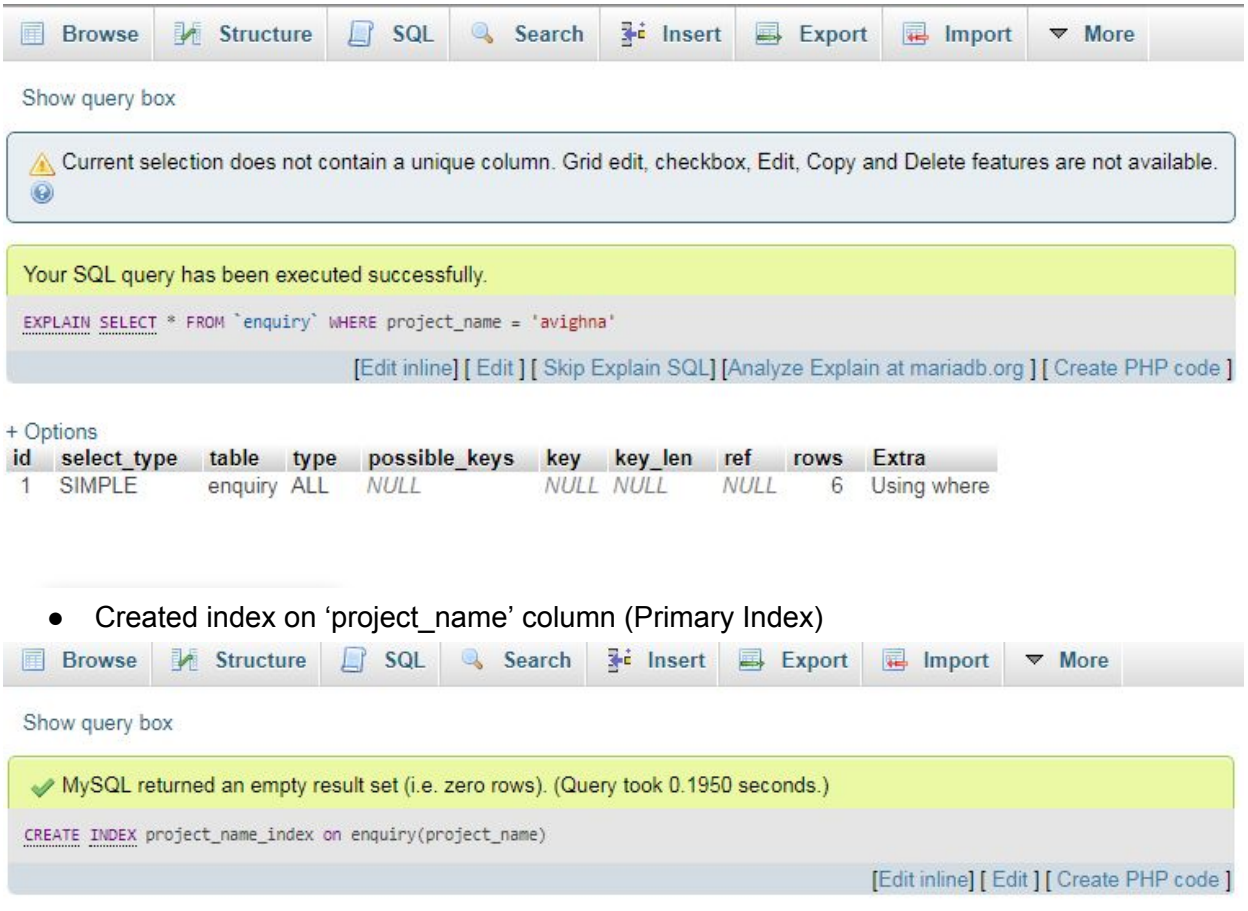
And mention why tuning is required? How it is performed and what is the effect of tuning? for all mentioned three categories of tuning.

INDEX TUNING

- Normal Select operation which accesses all records

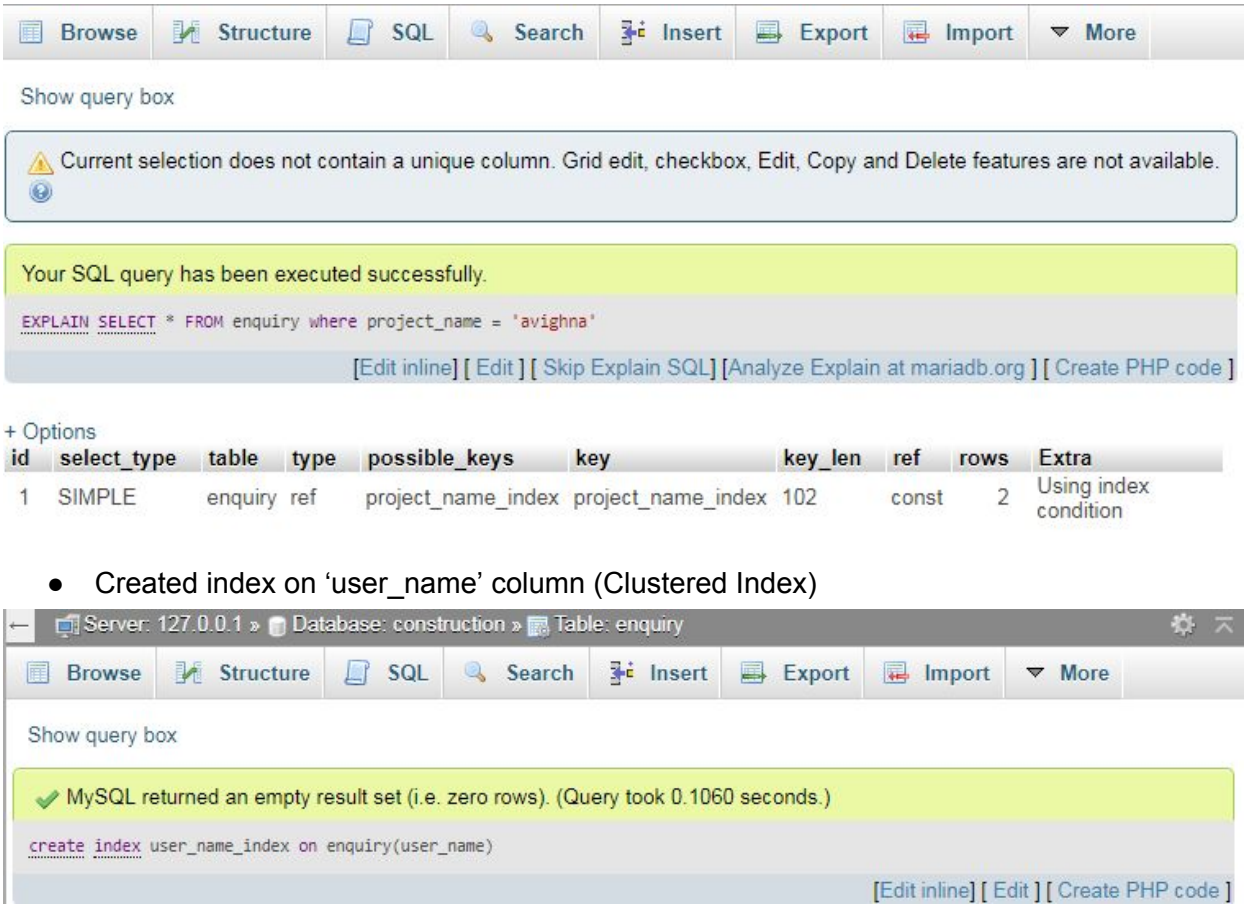


- Explain select operation thus proving that all records are accessed



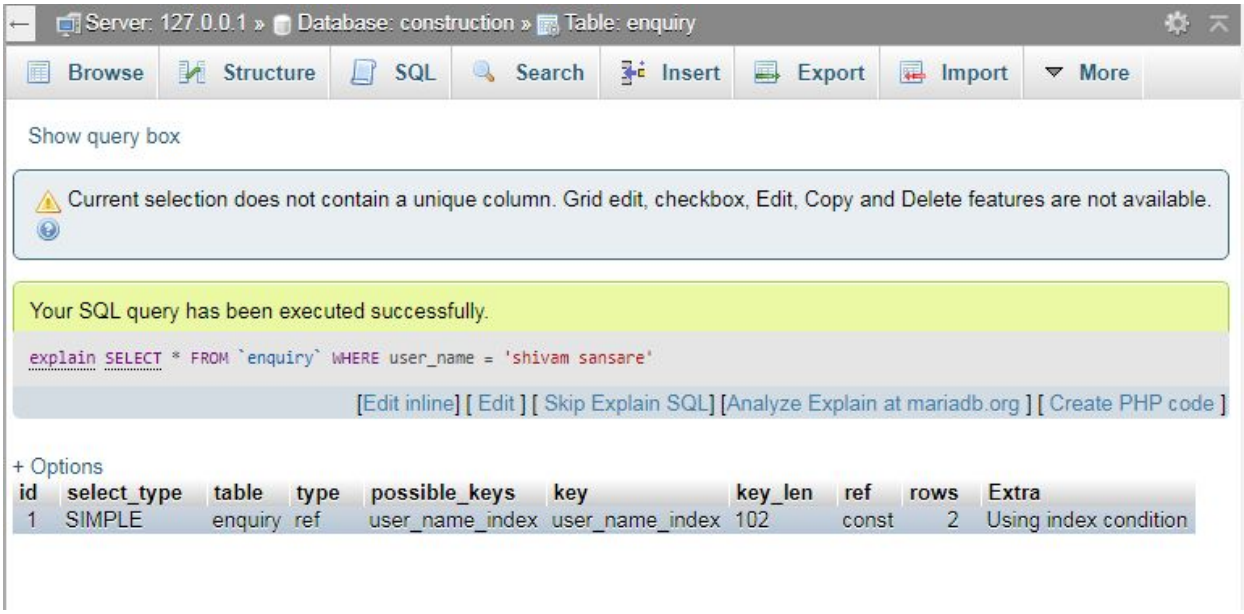
- Created index on 'project_name' column (Primary Index)

- Indexed query (We can see that less number of records are fetched thus reducing processing time)

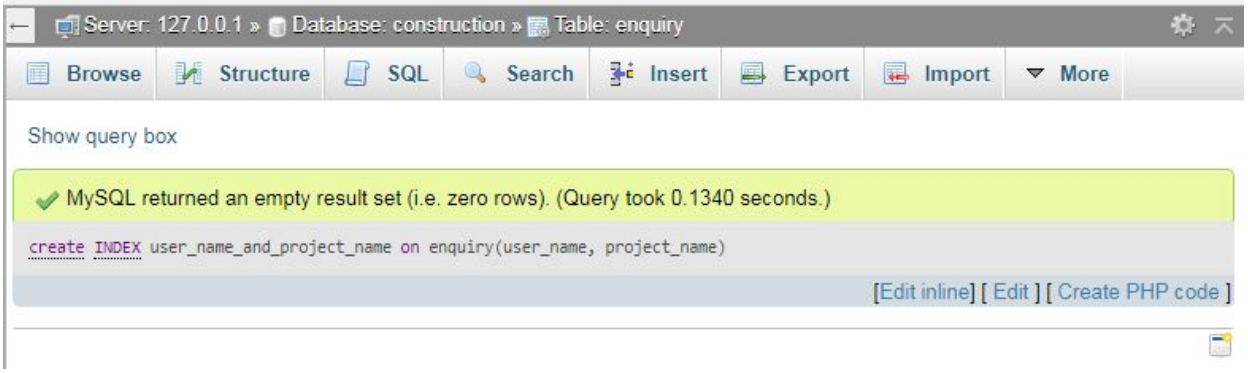


- Created index on ‘user_name’ column (Clustered Index)

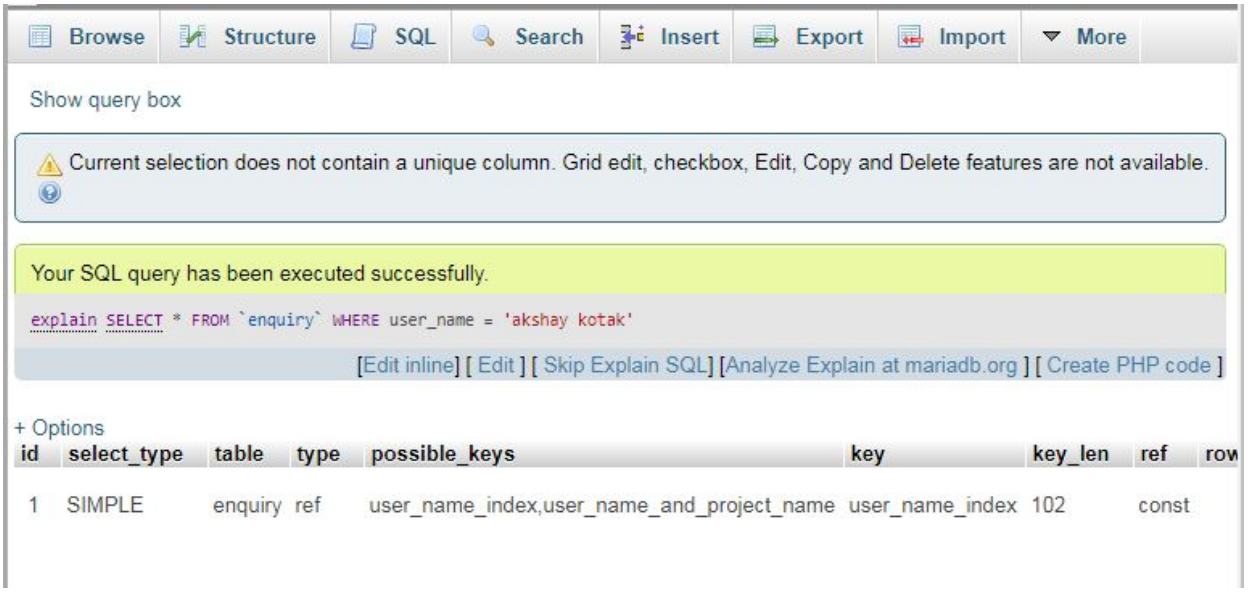
- Indexed query (Same explanation as above)



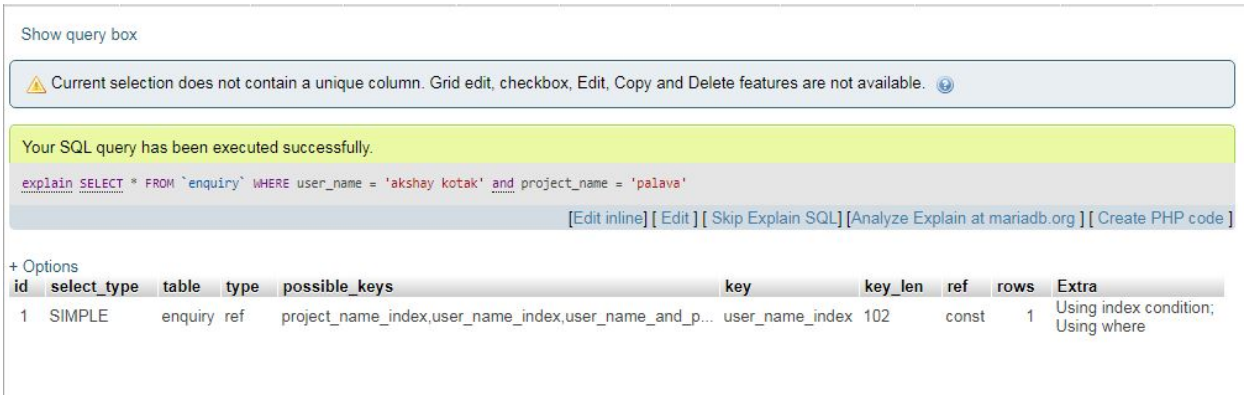
- Create index on 2 columns (Secondary Index)



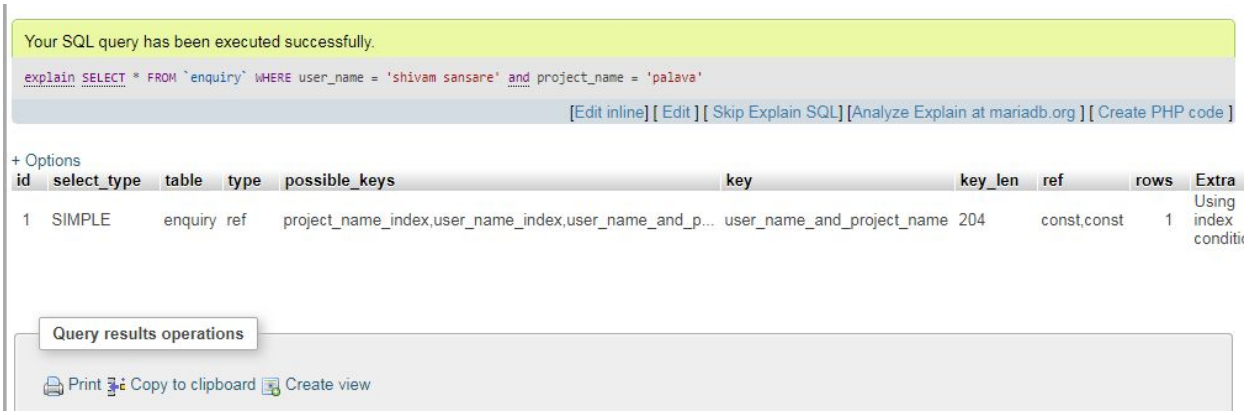
- Explain query on 2 indexes(We can see that less rows are fetched from)



- Explain query on 2 indexes



- Explain query on 2 indexes



Your SQL query has been executed successfully.

```
explain SELECT * FROM `enquiry` WHERE user_name = 'shivam sansare' and project_name = 'palava'
```

[Edit inline] [Edit] [Skip Explain SQL] [Analyze Explain at mariadb.org] [Create PHP code]

+ Options

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	enquiry	ref	project_name_index,user_name_index,user_name_and_p...	user_name_and_project_name	204	const,const	1	Using index conditi

Query results operations

Print Copy to clipboard Create view

QUERY TUNING

- SELECT (enquiry_id) FROM enquiry
  WHERE project_name IN (SELECT project_name
  FROM project
  WHERE project_name is NOT NULL)

SQL PERFORMANCE TUNING

recommendations:

SELECT (enquiry_id) FROM enquiry
WHERE project_name IN (SELECT project_name
FROM project
WHERE project_name is NOT NULL)

ERRORS SECTION

- ORA-00942: table or view does not exist

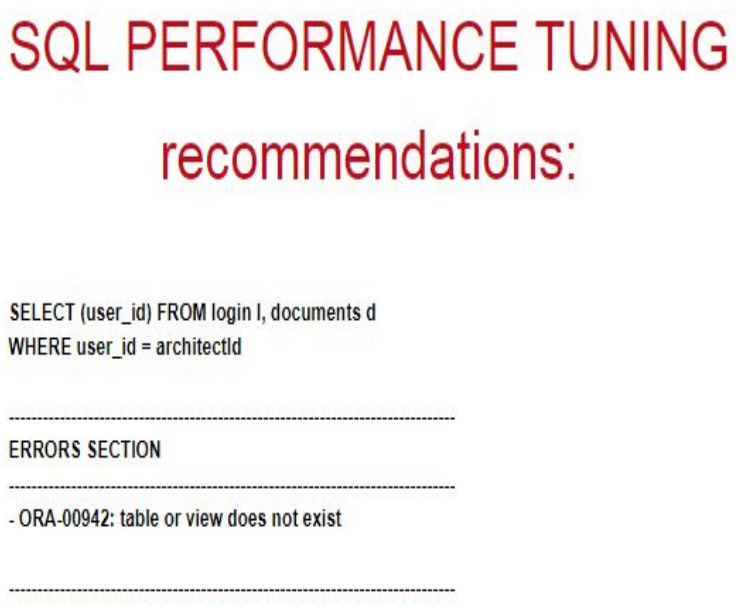
ALSO CONSIDER ABOUT

Instead of IN you should also try to use OR or EXISTS for subqueries

Why tuning is necessary?
Ans: Since we are fetching records from inner query by using IN operator which doesn't check if inner query returned rows or not which isn't a good option.

How tuning is performed?
Ans: By including EXISTS clause we tune the query for optimal performance.

- SELECT (user_id) FROM login l, documents d
  WHERE user_id = architectId



Why tuning is necessary?
Ans: No tuning is necessary for this query, it is optimal.

- SELECT (user_id) FROM login

```
WHERE user_id in ( select architectId FROM documents
            WHERE project_name exists ( select project_name
                        FROM project
                        where price_per_sqft > 10000) )
```

SQL PERFORMANCE TUNING

recommendations:

```
SELECT (user_id) FROM login
WHERE user_id in ( select architectId FROM documents
WHERE project_name exists ( select
project_name
FROM project
where price_per_sqft
> 10000) )

---------------------------------------------------------------------------
ERRORS SECTION
---------------------------------------------------------------------------
- ORA-00920: invalid relational operator

---------------------------------------------------------------------------

ALSO CONSIDER ABOUT

Instead of IN you should also try to use OR or EXISTS for subqueries
Instead of EXISTS you should also try to use IN
```

- select designation
  from login L
  where L.user_id in (select purpose
  from accounts A
)

```
select designation
from login L
where L.user_id in ( select purpose
from accounts A
)
```

ERRORS SECTION

- ORA-00942: table or view does not exist

ALSO CONSIDER ABOUT

Instead of IN you should also try to use OR or EXISTS for subqueries

Why tuning is necessary?
Ans: Instead of comparing each records from inner query with outer query, one can use JOIN condition and do the same with optimal performance.
How tuning is performed?
Ans: By using JOIN relational operator.

- SELECT document_name

FROM documents d, project p
WHERE a.project_name = p.project_name;

SQL PERFORMANCE TUNING recommendations:

```
SELECT document_name
FROM documents d, project p
WHERE a.project_name = p.project_name

--------------------------------------------------------------------------------
ERRORS SECTION
--------------------------------------------------------------------------------
- ORA-00942: table or view does not exist

--------------------------------------------------------------------------------
```

No tuning required. Query is already optimised.

- SELECT user_name, remaining_flats
  FROM enquiry e, project p
  WHERE e.project_name = p.project_name;

```
SELECT user_name, remaining_flats
FROM enquiry e, project p
WHERE e.project_name = p.project_name

--------------------------------------------------------------------------------
ERRORS SECTION
--------------------------------------------------------------------------------
- ORA-00942: table or view does not exist

--------------------------------------------------------------------------------
```

- SELECT user_name,count(1)
  FROM enquiry e
  WHERE e.project_name in (select project_name
  from project)

SQL PERFORMANCE TUNING recommendations:

```
SELECT user_name,count(1)
FROM enquiry e
WHERE e.project_name in ( select project_name
from project)

--------------------------------------------------------------------------------
ERRORS SECTION
--------------------------------------------------------------------------------
- ORA-00942: table or view does not exist

--------------------------------------------------------------------------------

ALSO CONSIDER ABOUT

Instead of IN you should also try to use OR or EXISTS for subqueries
```

Why tuning is necessary?
Ans: Since count(*) counts all the rows of the result table for every column which is redundant. We can do so by specifying count(1) too.
How tuning is performed?
By using count(1) instead of count(*) we reduced the query processing time.

DATABASE TUNING

Database tuning means tuning the database for optimal performance while querying the database.
Like making less use of number to string conversions etc.
Here,

NO DENORMALIZATION IS REQUIRED

Conclusion:

Hence the various tuning like index,query,database tuning been successfully understood and implemented

Post Lab Descriptive Questions:

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

Ans:

Physical design is an activity where the goal is not only to create the appropriate structuring of data in storage, but also to do so in a way that guarantees good performance. For a given conceptual schema, there are many physical design alternatives in a given DBMS. It is not possible to make meaningful physical design decisions and performance analyses until the database designer knows the mix of queries, transactions, and applications that are expected to run on the database. This is called the job mix for the particular set of database system applications. The database administrators/designers must analyze these applications, their expected frequencies of invocation, any timing constraints on their execution speed, the expected frequency of update operations, and any unique constraints on attributes. We discuss each of these factors next.

Design Criteria

- Response Time

  Elapsed Time between submitting a database transaction for execution and receiving a response
- Space Utilization

  Storage space used by database files and their access path structures
- Transaction throughput
  - Average number of transactions/minute
  - Must be measured under peak conditions

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:

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

For each update operation or update transaction, the following information would be needed:

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

Again, the attributes listed in item 3 are candidates for access structures on the files, because they would be used to locate the records that will be updated or deleted. On the other hand, the attributes listed in item 4 are candidates for *avoiding an access structure*, since modifying them will require updating the access structures.

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

Besides identifying the characteristics of expected retrieval queries and update transactions, we must consider their expected rates of invocation. This frequency information, along with the attribute information collected on each query and transaction, is used to compile a cumulative list of the expected fre-quency of use for all queries and transactions. This is expressed as the expected fre-quency of using each attribute in each file as a selection attribute or a join attribute, over all the queries and transactions. Generally, for large volumes of processing, the informal *80–20 rule* can be used: approximately 80 percent of the processing is accounted for by only 20 percent of the queries and transactions. Therefore, in prac-tical situations, it is rarely

necessary to collect exhaustive statistics and invocation rates on all the queries and transactions; it is sufficient to determine the 20 percent or so most important ones.

C.Analyzing the Time Constraints of Queries and Transactions.

Some queries and transactions may have stringent performance constraints. For example, a transaction may have the constraint that it should terminate within 5 seconds on 95 percent of the occasions when it is invoked, and that it should never take more than 20 seconds. Such timing 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 structures for the files, because the primary access structures are generally the most efficient for locating records in a file.

D.Analyzing the Expected Frequencies of Update Operations.

A minimum number of access paths should be specified for a file that is frequently updated, because updating the access paths themselves slows down the update operations. For example, if a file that has frequent record insertions has 10 indexes on 10 different attributes, each of these indexes must be updated whenever a new record is inserted. The overhead for updating 10 indexes can slow down the insert operations.

E.Analyzing the Uniqueness Constraints on Attributes.

Access paths should be specified on all *candidate key* attributes—or sets of attributes—that are either the primary key of a file or unique attributes. The existence of an index (or other access path) makes it sufficient to only search the index when checking this uniqueness constraint, since all values of the attribute will exist in the leaf nodes of the index. For example, when inserting a new record, if a key attribute value of the new record *already exists in the index*, the insertion of the new record should be rejected, since it would violate the uniqueness constraint on the attribute.

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

.

Date: _____