# COURSE NAME: DBMS
# COURSE CODE:23AD2102A

## TOPIC:

# QUERY PROCESSING AND OPTIMIZATION

# AIM OF THE SESSION

To familiarize students with the basic concept of Query Processing and Optimization

# INSTRUCTIONAL OBJECTIVES

This Session is designed to: understand how Query processed and

Optimized to produces the output of the query

# LEARNING OUTCOMES

At the end of this session, you should be able to: Identify the concept of Query Processing and

Optimization

# What is a query?

- Query display a subset of data contained in various tables of the database
- Query is used to retrieve records from the table

# How Does Query Work?

- Query can either be a select, an action, or a combination of both. Select queries can retrieve information from data sources, and action queries work for data manipulation

  Example, to add, change or delete data

- Advanced users can also use query commands to perform various programming tasks and granting permissions

# What are the benefits of using a query?

- Review data from multiple tables simultaneously
- Filter records containing only certain fields and of certain criteria
- Automate data management tasks and perform calculations

# What is query optimization?

- Query optimization is the process of improving the performance of a database query

- Performance improving the query itself, improving the indexing of the database, or improving the database design

## What are some typical ways that the optimizer decides which query plan it should use?

- Optimizer typically looks at the cost of each query plan and chooses the one that is cheapest

- Consider other factors such as the number of rows that need to be retrieved, the order in which the rows need to be retrieved, and the availability of indexes

# What is query optimization?

- Query processing is the process of translating a high-level query, such as SQL, into a low-level query that can be executed by the database system

- Involves parsing, validating, and optimizing the query, as well as generating a query execution plan

# QUERY PROCESSING IN DBMS

- Query Processing is the activity performed in extracting data from the database
- Query processing includes certain activities for data retrieval
- Most important tasks of a DBMS is to figure out an efficient evaluation strategy (access plan) for high level statements

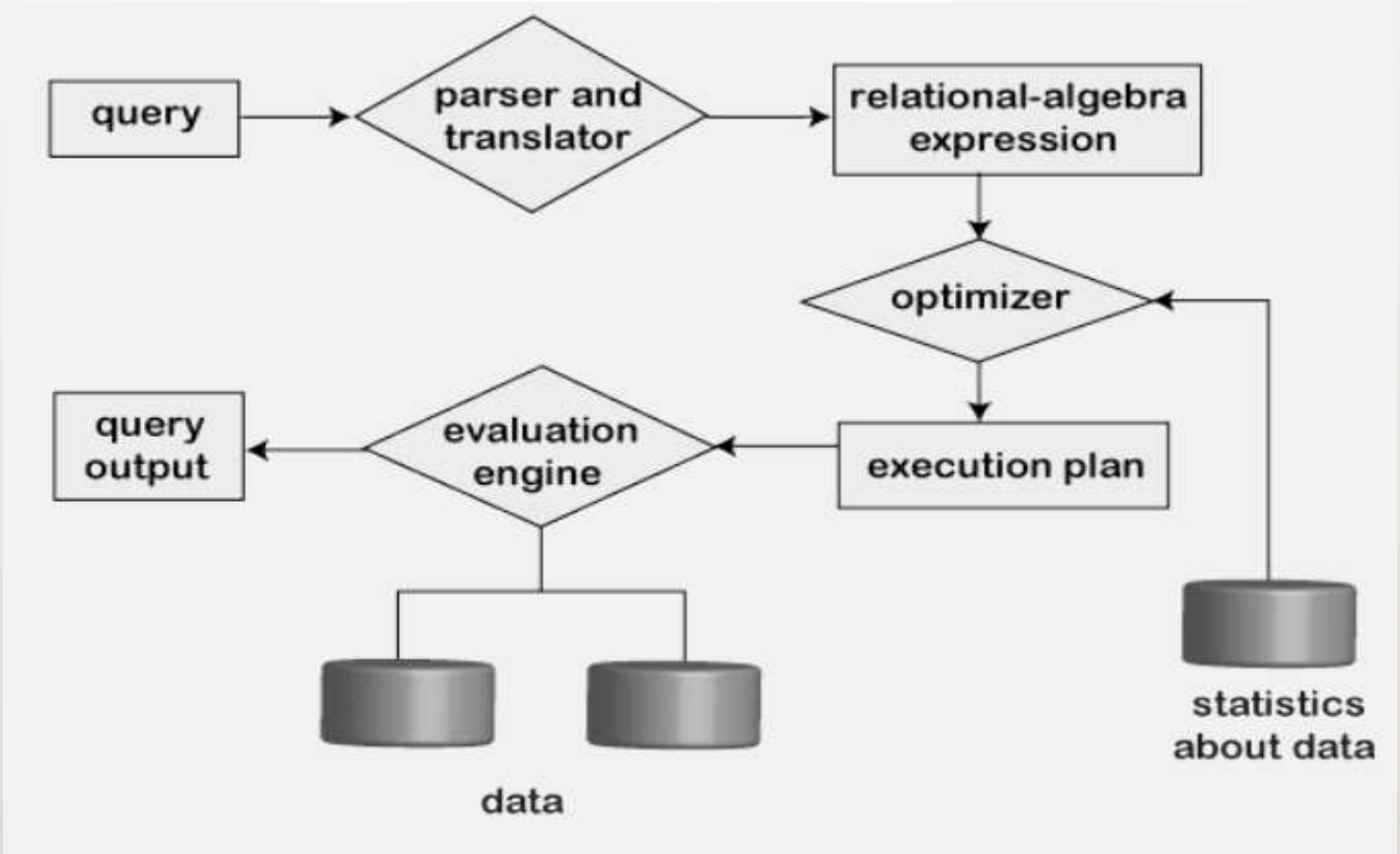
# QUERY PROCESSING IN DBMS

Query processing is a 3-step process for fetching the data from the database

***STEPS involved are***:

- Parsing and translation
- Optimization
- Evaluation

# STEPS IN QUERY PROCESSING



# STEPS IN QUERY PROCESSING - PARSING & TRANSLATION

- Computer system needs to translate the query into a human-readable and understandable language
- User queries get translated in high-level database languages such as SQL
- Relational algebra is well suited for the internal representation of a query

# STEPS IN QUERY PROCESSING - PARSING & TRANSLATION

- First step in query processing is **Parsing and Translation**
- Queries undergo lexical, syntactic, and semantic analysis
- Essentially, the query gets broken down into different tokens and white spaces are removed along with the comments (**Lexical Analysis**)
- In the next step, the query gets checked for the correctness, both syntax and semantic wise
  Query processor first checks the query if the rules of SQL have been correctly followed or not (**Syntactic Analysis**)
- Finally, the query processor checks if the meaning of the query is right or not.
  Things like if the table(s) mentioned in the query are present in the DB or not? if the column(s) referred from all the table(s) are actually present in them or not? (**Semantic Analysis**)
- Once the above mentioned checks pass, the flow moves to convert all the tokens into relational expressions, graphs, and trees
- Makes the processing of the query easier for the other parsers

# PARSING

- When a user executes any query  for generating the internal form of the query
- Parser in the system checks the syntax of the query, verifies the name of the relation in the database, the tuple, and finally the required attribute value
- Once syntax and semantic checks pass, the flow moves to convert all the tokens into relational expressions, graphs, and trees
- Makes the processing of the query easier for the other parsers
- Parser creates a tree of the query, known as 'parse-tree'

# QUERY EXAMPLE

**User wants to fetch the records of the employees whose salary is greater than or equal to 20000**

# PARSING

**QUERY:**

**User wants to fetch the records of the employees whose salary is greater than or equal to 20000**

- **Query undertaken**

**select *emp_name* from Employee where *salary* > 20000;**

# QUERY WOULD BE DIVIDED INTO TOKENS…

**TOKENS ARE**

- SELECT
- emp_name
- FROM
- employee
- WHERE
- Salary
- >
- 10000

# TOKENS(QUERY) GET VALIDATES FOR....

- Name of the queried table is looked into the data dictionary table
- Name of the columns mentioned (emp_name and salary) in the tokens are validated for existence
- Type of column(s) being compared have to be of the same type

  (salary and the value 20000 should have the same data type)

# TRANSLATION

- Next step is to translate the generated set of tokens into a relational algebra query
- These are easy to handle for the optimizer in further processes
- Gets translated into expressions that can be further used at the physical level of the file system

# TRANSLATION

**QUERY: user wants to fetch the records of the employees whose salary is greater than or equal to 20000**

- To make the system understand the user query, it needs to be translated in the form of relational algebra

**Query in the relational algebra form as:**

- $\sigma_{salary>20000}\ (\pi_{salary}\ (Employee))$

- $\pi_{salary}\ (\sigma_{salary>20000}\ (Employee))$

After translating the given query, we can execute each relational algebra operation by using different algorithms in this way, a query processing begins its working

# STEPS IN QUERY PROCESSING - QUERY EVALUATION

- After translating the user query, the system executes a query evaluation plan
- Once the query processor has the relational forms with it, the next step is to apply certain rules and algorithms to generate a few other powerful and efficient data structures
- Data structures help in constructing the query evaluation plans
- Addition to the relational algebra translation, it is required to annotate the translated relational algebra expression with the instructions used for specifying and evaluating each operation

For example: if the relational graph was constructed, there could be multiple paths from source to destination
A query execution plan will be generated for each of the paths

# QUERY EVALUATION PLAN

- In order to fully evaluate a query, the system needs to construct a query evaluation plan
- Annotations in the evaluation plan may refer to the algorithms to be used for the particular index or the specific operations
- Such relational algebra with annotations is referred to as **Evaluation Primitives**
- Evaluation primitives carry the instructions needed for the evaluation of the operation
- Query evaluation plan defines a sequence of primitive operations used for evaluating a query
- Query evaluation plan is also referred to as **the query execution plan**
- A **query execution engine** is responsible for generating the output of the given query
- It takes the query execution plan, executes it, and finally makes the output for the user query

# STEPS IN QUERY PROCESSING - QUERY OPTIMIZATION

- Cost of the query evaluation can vary for different types of queries
- DMBS picks up the most efficient evaluation plan based on the cost each plan has
- Database system generates an efficient query evaluation plan, which minimizes its cost

  Type of task performed by the database system and is known as Query Optimization
- Aim is to minimize the query evaluation time

# STEPS IN QUERY PROCESSING - QUERY OPTIMIZATION

- Optimizer also evaluates the usage of index present in the table and the columns being used. It also finds out the best order of subqueries to be executed so as to ensure only the best of the plans gets executed

- For any query there are multiple evaluation plans to execute it. Choosing the one which costs the least is called Query Optimization in DBMS

- Although the system is responsible for constructing the evaluation plan, the user does need not to write their query efficiently

# FACTORS WEIGHED IN BY THE OPTIMIZER

Factors weighted in by the optimizer to calculate the cost of a query evaluation plan

- CPU time
- Number of tuples to be scanned
- Disk access time
- Number of operations

# STEPS IN QUERY PROCESSING - QUERY OPTIMIZATION

- For optimizing a query, the query optimizer should have an estimated cost analysis of each operation.
- Overall operation cost depends on the memory allocations to several operations, execution costs and so on
- Finally, after selecting an evaluation plan, the system evaluates the query and produces the output of the query

# SUMMARY

Hashing is a DBMS technique for searching for needed data on the disc without utilising an index structure. The hashing method is basically used to index items and retrieve them in a DB since searching for a specific item using a shorter hashed key rather than the original value is faster.

# SELF-ASSESSMENT QUESTIONS

1. What is hashing?

(a) A data structure for storing key-value pairs

(b) b) A technique for converting data of arbitrary size to a fixed size

(c) c) A process of compressing data to save space

(d) d) A method for encrypting data

2. Which of the following is not a suitable use case for hashing?

a) Password storage
b) Data validation
c) Data encryption
d) Sorting large datasets

# TERMINAL QUESTIONS

1. Can you explain the difference between static and dynamic hashing, and when each is appropriate?
2. What is collision handling, and how is it handled in hashing-based index structures?
3. How does the choice of hash function affect the performance of a hashing-based index?
4. What is a primary index, and how is it implemented using hashing in DBMS?
5. How does extendible hashing differ from linear and quadratic probing?

# REFERENCES FOR FURTHER LEARNING OF THE SESSION

**Reference Books:**


1. "Database Management Systems" by Raghu Ramakrishnan and Johannes Gehrke - This book covers the basics of database management systems, including the concept of index structures.
2. "Database Systems: Design, Implementation, and Management" by Carlos Coronel, Steven Morris, and Peter Rob - This book provides a comprehensive introduction to database systems, including index structures and their importance in optimizing database performance.
3. "Database Indexing: A Practical Guide for Developers" by Will Iverson - This book focuses specifically on the concept of indexing in database management systems, providing practical advice and examples for developers.


**Sites and Web links:**


1. "Hash-Based Indexes" by Raghu Ramakrishnan and Johannes Gehrke - https://www-db.cs.wisc.edu/cidr/cidr2003/papers/cidr03p50.pdf
2. "Indexing and Hashing" by S. Sudarshan - http://www.cse.iitb.ac.in/~sudarsha/dbbook/dbchapter7.pdf
3. "Extendible Hashing" by Jerome Martin - http://pages.di.unipi.it/martin/Papers/ExtendibleHashing/ExtendibleHashing.pdf
4. "Concurrency Control in Hash-Based Database Systems" by Christoph G. Schuetz and Michael H. Böhlen - https://link.springer.com/chapter/10.1007/3-540-45481-7_2

# THANK YOU


Team – DBMS