Sample Questions:

1. **Explain Query processing in DBMS using a neat diagram.**

Refers to the range of activities involved in extracting data from a database.

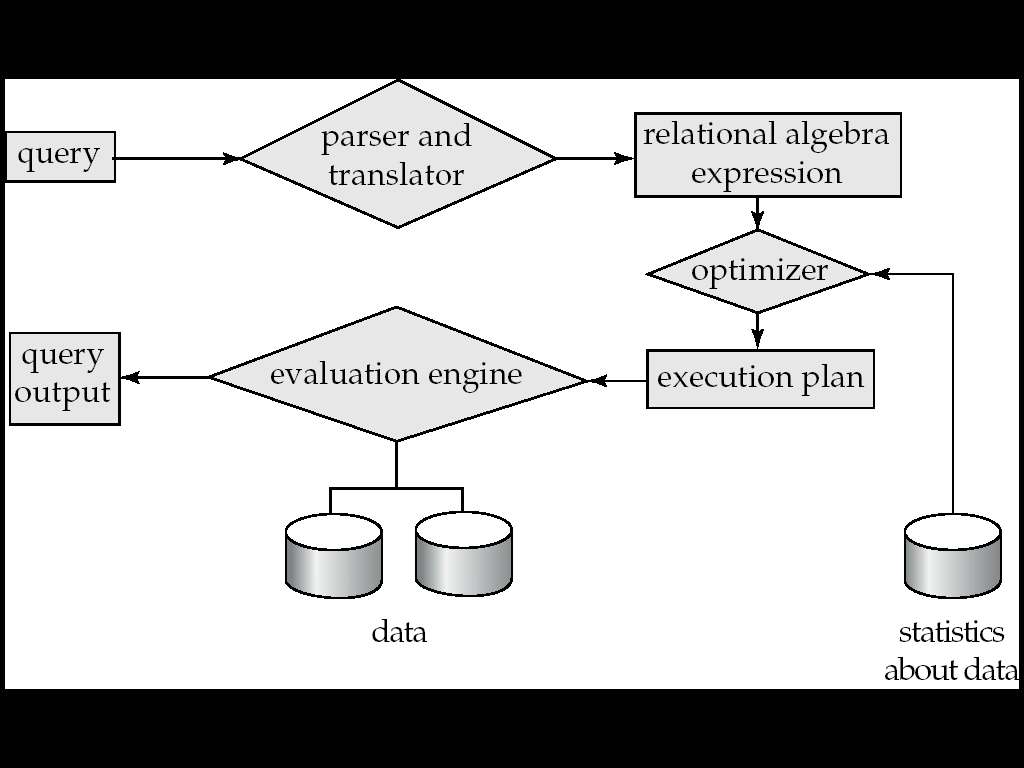
The activities includes translation of queries in high level database language into expression that can be used at the physical level of the file system

Variety of query optimizing transformations are used.

Actual evaluation plan is created.

Query Processing would mean the entire process or activity which involves **query translation into low level instructions, query optimization to save resources, cost estimation or evaluation of query, and extraction of data from the database.**

**Goal:** To find an efficient Query Execution Plan for a given SQL query which would minimize the cost considerably, especially time.



1. **Functions of every component in query processing**

**Query tree**

Query Processing is the activity performed in extracting data from the database.

In query processing, it takes various steps for fetching the data from the database. The steps involved are:

Steps involved in processing a query are:

1.Parsing and translation

2. Optimization

3. Evaluation

**Parsing**

This is the first step of any query processing.

The user typically writes request in SQL language.

In order to process and execute this request, DBMS has to convert it into low level – machine understandable language.

Any query issued to the database is first picked by the query processor.

It scans and parses the query into individual tokens and examines for the correctness of the query.

It checks for the validity of tables / views used and the syntax of the query.

**Translation**

If we have written a valid query, then it is converted from high level language SQL to low level instruction in Relational Algebra.

Once it is passed, then it converts each token into relational algebra expressions , trees and graphs.

These are easily processed by the other parsers in the DBMS.

Initially, the SQL query is scanned.

Then it is parsed to look for syntactical errors and correctness of data types.

If the query passes this step, the query is decomposed into **smaller query blocks**.

Each block is then translated to an equivalent relational algebra expression.

**Query Tree Generation:**

A query tree is a tree data structure representing a relational algebra expression.

The tables of the query are represented as leaf nodes.

The relational algebra operations are represented as the internal nodes.

The root represents the query as a whole.

During execution, an internal node is executed whenever its operand tables are available.

The node is then replaced by the result table.

This process continues for all internal nodes until the root node is executed and replaced by the result table.

**Optimization:**

Optimization is the process of selecting the most efficient query evaluation plan from among multiple strategies possible for processing a given query.

Optimizer uses the statistical data stored as part of a data dictionary.

The statistical data are information about the size of the table, the length of records, the indexes created on the table, etc.

Optimizer also checks for the conditions and conditional attributes which are parts of the query.

Query Optimization: Amongst all equivalent evaluation plans choose the one with lowest cost.

The goal of the query optimizer is to find a *reasonably efficient* strategy for executing the query using the access routines.

**Query evaluation plan:**

Annotated expression specifying a detailed evaluation strategy is called an evaluation-plan.

After the query tree is generated, a query plan is made. A query plan is an extended query tree that includes access paths for all operations in the query tree.

Access paths specify how the relational operations in the tree should be performed.

For example, a selection operation can have an access path that gives details about the use of B+ tree index for selection.

Besides, a query plan also states how the intermediate tables should be passed from one operator to the next, how temporary tables should be used and how operations should be pipelined/combined.

Sequence of operations that can be used to evaluate a query is called a Query Evaluation Plan.

E.g., can use an index on *balance* to find accounts with balance < 2500,

or can perform complete relation scan and discard accounts with balance ≥ 2500

**Query evaluation plan are represented as tree of relational operators, along with labels that can be performed at every node**

1. **Query Evaluation Plan**

Annotated expression specifying a detailed evaluation strategy is called an evaluation-plan.

After the query tree is generated, a query plan is made. A query plan is an extended query tree that includes access paths for all operations in the query tree.

Access paths specify how the relational operations in the tree should be performed.

For example, a selection operation can have an access path that gives details about the use of B+ tree index for selection.

Besides, a query plan also states how the intermediate tables should be passed from one operator to the next, how temporary tables should be used and how operations should be pipelined/combined.

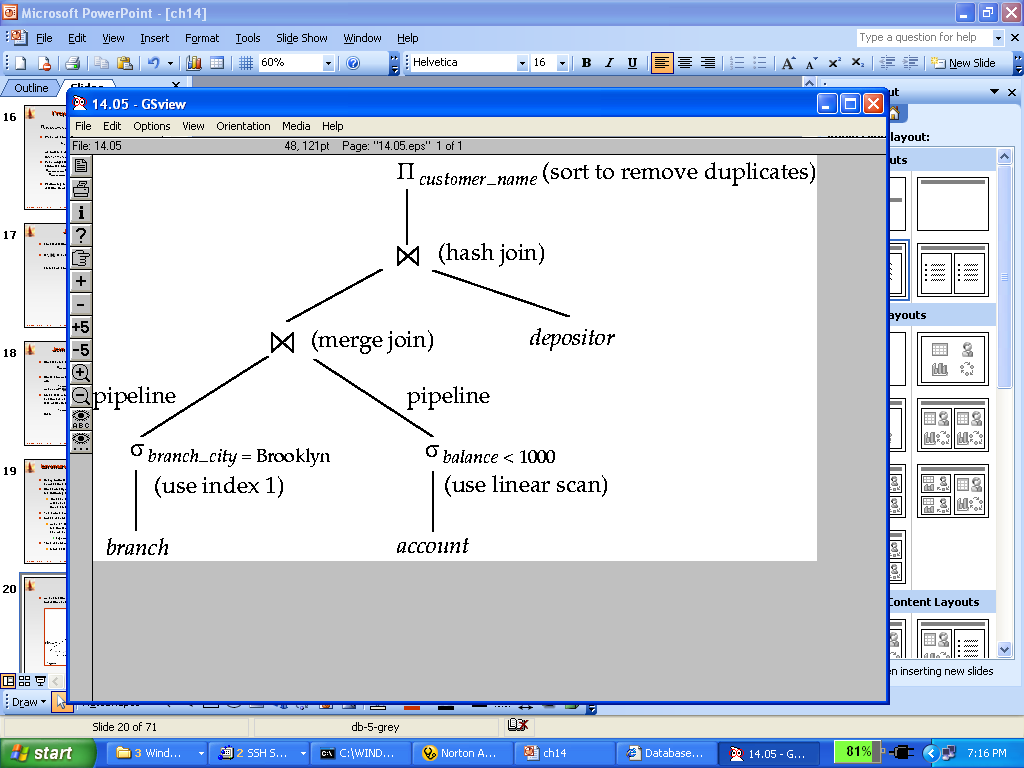
Sequence of operations that can be used to evaluate a query is called a Query Evaluation Plan.

E.g., can use an index on *balance* to find accounts with balance < 2500,

or can perform complete relation scan and discard accounts with balance ≥ 2500

**Query evaluation plan are represented as tree of relational operators, along with labels that can be performed at every node**

**An evaluation plan defines exactly what algorithm is used for each operation, and how the execution of the operations is coordinated.**

****

1. **Query optimization and different types of Query Optimization**

**Query optimization** is the process to find a *reasonably efficient* strategy for executing the query using the access routines.

Amongst all equivalent evaluation plans choose the one with lowest cost.

**Importance of Query Optimization**

Query optimization provides faster query processing.

It requires less cost per query.

It gives less stress to the database.

It provides high performance of the system.

It consumes less memory.

Types of Query optimization :

We divide the query optimization into two types:  
**Heuristic** (sometimes called Rule based) and **Systematic** (Cost based).

In Heuristic Optimization, the query execution is refined based on *heuristic rules* for reordering the individual operations.

With Cost Based Optimization, the overall cost of executing the query is systematically reduced by estimating the costs of executing several different execution plans.

1. Differentiate Query tree and Query Evaluation engine
2. **Heuristic and Cost-based Query Optimization**

**Heuristic Optimization**

Is also called Rule Based optimization.

A query can be represented as a tree data structure. Operations are at the interior nodes and data items (tables, columns) are at the leaves.

The query is evaluated in a *depth-first* pattern.

The main heuristic is to apply first the operations that reduce the size of intermediate results.

Perform as early as possible SELECT operations to reduce the number of tuples and PROJECT to reduce the number of attributes.

This is done by moving SELECT and PROJECT operation as far down the tree possible.

SELECT-JOIN operation which is most restrictive should be executed first by reordering the leaf nodes and adjusting the rest of the tree appropriately.

**Steps in Heuristic Optimization:**

Step 1: Design the initial canonical tree of the query

(Canonical tree

• tree where any subtree can be evaluated in any order (informal definition)

Step 2.Deconstruct the conjunctive selections into a sequence of single selection operations.

Step 3: Move the SELECT operation down the query tree Step 4: Apply more restrictive SELECT operation first.

Step 5: Replace CARTESIAN PRODUCT and SELECT with JOIN operation Step

Step 6: Move PROJECT operations down the query tree

Step 7.Apply equivalence rules for further optimization.

* **Cost Based Optimization:**

A query optimizer does not depend solely on heuristic rules.

It also estimates and compares the costs of executing a query using different execution strategies and algorithms, and it then chooses the strategy with the lowest cost estimate.

For this approach to work, accurate cost estimates are required so that different strategies can be compared fairly and realistically.

In addition, the optimizer must limit the number of execution strategies to be considered .

**Cost Components of Query Execution**

The cost of executing the query includes the following components:

* Access cost to secondary storage.
* Storage cost.
* Computation cost.
* Memory uses cost.
* Communication cost.

The cost of executing a query includes the following components:

**Access cost to secondary storage.**

This is the cost of transferring (reading and writing) data blocks between secondary disk storage and main memory buffers. This is also known as disk I/O (input/output) cost.

The cost of searching for records in a disk file depends on the type of access structures on that file, such as ordering, hashing, and primary or secondary indexes.

In addition, factors such as whether the file blocks are allocated contiguously on the same disk cylinder or scattered on the disk affect the access cost.

**Disk storage cost**.

This is the cost of storing on disk any intermediate files that are generated

by an execution strategy for the query.

**Computation cost.** This is the cost of performing in-memory operations on the records within the data buffers during query execution. Such operations include searching for and sorting records, merging records for a join or a sort operation, and performing computations on field values. This is also known as CPU (central processing unit) cost.

**Memory usage cost.** This is the cost pertaining to the number of main memory buffers needed during query execution.

**Communication cost**. This is the cost of shipping the query and its results from the database site to the site or terminal where the query originated.

1. **Draw a query tree for a given query.**
2. **Convert Query tree into optimized Query tree using heuristic optimization.**
3. **Heuristics Rules for heuristic query optimization Equivalence Rules**

* Query processing page 67

1. **Parameters used for cost optimization**
2. **Differentiate heuristic and cost based optimization**



**Distributed Database:**

1. **Explain different types of transparency in distributed database management systems with examples.**

**Management of distributed data with different levels of transparency**

The user of a distributed system should not be required

To know where the data are physically located or how the data can be accessed at the specific local site. This characteristic is called data transparency. (This refers to the physical placement of data (files, relations, etc.) which is not known to the user (distribution transparency).

**Types of transparency:**

**Distribution or network transparency-** Users do not have to worry about operational details of the network.

**Location transparency :**

Users are not required to know the physical location of the data. The DDBMS should be able to find any data as long as the data identifier is supplied by the user transaction.

(refers to freedom of issuing commands from any location without knowing its working).

Location transparency ensures that the user can query on any table(s) or fragment(s) of a table as if they were stored locally in the user’s site.

The fact that the table or its fragments are stored at a remote site in the distributed database system, should be completely oblivious to the end user.

The address of the remote site(s) and the access mechanisms are completely hidden.

**Naming transparency :**

DDBMS creates a set of aliases for data items. Users may refer to this data item by a simple name that is translated by the system to the complete name.

Mapping of aliases to the real names can be stored on each site. (allows access to any named object (files, relations, etc.) from any location).

**Replication transparency-** allows to store copies of data at multiple sites. This is done to minimize access time to the required data.

User is unaware of the existence of multiple copies

Users view each data object as logically unique.

The distributed system may replicate an object to increase either system performance or data availability.

System should determine which replica to reference on a read request and should update all replicas on a write request.

**Fragmentation transparency**-Allows to fragment a relation horizontally (create a subset of tuples of a relation) or vertically (create a subset of columns of a relation).

Fragmentation transparency enables users to query upon any table as if it were unfragmented.

Thus, it hides the fact that the table the user is querying on is actually a fragment or union of some fragments.

It also conceals the fact that the fragments are located at diverse sites.

This is somewhat similar to users of SQL views, where the user may not know that they are using a view of a table instead of the table itself.

Users are not required to know how a relation has been fragmented.

Horizontal fragmentation

Vertical fragmentation

1. **Explain advantages and disadvantages of distributed DBMS.**

**Advantages**

* **Reduced Communication Overhead**

Most data access is local, less expensive and performs

better.

* **Improved Processing Power**

Instead of one server handling the full database, we now

have a collection of machines handling the same database.

* **Removal of Reliance on a Central Site**

If a server fails, then the only part of the system that is

affected is the relevant local site. The rest of the system

remains functional and available.

* **Expandability**

It is easier to accommodate increasing the size of the

global (logical) database.

* **Local autonomy**

The database is brought nearer to its users. This can effect

a cultural change as it allows potentially greater control

over local data

**Robust**–A problem in one part of the organization will not stop other branches working.

**Security-** Staff access can be restricted to only their portion of databases.

**Network traffic** is reduced, thus reducing the bandwidth cost.

**High Performance**–Queries and updates are largely local so that there is no network bottleneck.

**Removal of Reliance on a Central Site** If a server fails, then the only part of the system that is affected is the relevant local site. The rest of the system remains functional and available.

**Expandability**

It is easier to accommodate increasing the size of the

global (logical) database. **Increased Reliability and Availability**

**Reliability** – Probability that a system is running at a given time

**Availability** – Probability that a system is continuously available during a time interval

When the data and the DBMS software are distributed Over several sites ,one site may fail other sites continue to Operate. Only the data and the software that exist at the failed site cannot be accessed. This improves both reliability and availability

**Improved Performance**

Data Localization – A Distributed database management system fragments the database by keeping the data closer to where it is needed. Data Localization reduces the contention for CPU and I/O services and simultaneously reduces access delays involved in wide area networks.

**Easier Expansion-** In a Distributed environment , expansion of the system in terms of adding more data, increasing the database sizes or adding more processors is much easier

**Disadvantages.**

Not Cost effective.

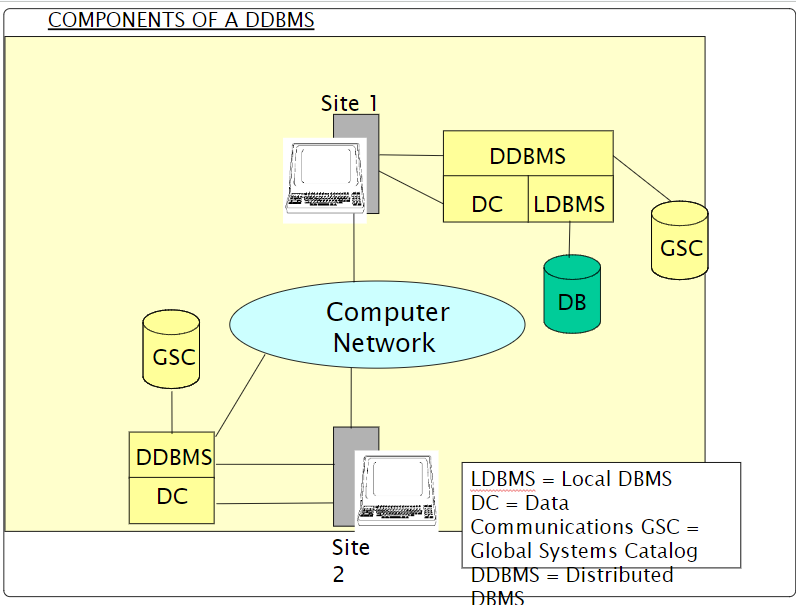
Integrity control more difficult

Lack of standards

Lack of experience

Database design more complex

1. **Explain different components of distributed DBMS.**

****

* DATA CATALOG:- Describes all data objects stored and managed by the system.
* Global catalog: which represents the sum of all data dictionaries within the system.
* Approaches:
* Completely dispersed: each site has a copy of the local catalog.
* Completely replicated : each site has a complete copy of the entire system catalog.
* Centralized Global catalog.

1. **Explain different types of fragmentations in distributed databases.**
2. **Fragmentation examples and correctness criteria.**
3. **Explain different design issues in distributed databases.**

Data Replication

Data Allocation

Concurrency Control

1. **What is replication? Explain advantages and disadvantages of using replication.**
2. **Explain different allocation techniques in distributed databases.**
3. **Design fragmentation schema with guard conditions for given case study like hospital database etc.**
4. **Fragmentation examples**

**Advanced Data Management Model:**

Need and measures of Data security

Explain DAC,MAC and Role based access Control.

Give examples of Role based access control and MAC