5 SEMANTIC DATA CONTROL

5.1 : Introduction of Semantic Data Control

Q.1 Explain view management in centralized as well as distributed DBMS. ☞ [GTU : Winter-16, Summer-18 Marks 7]

OR Explain views in DDBMS. ☞ [GTU : Winter-17, Marks 3]

Ans. : • Views are virtual relations that are defined as the result of a query on base relation. When a view is used in a query, view composition must take place in order to derive an execution strategy for the query. If views are not objects of authorization, this composition can take place at any site.

- If views are objects of authorization, site autonomy considerations require that the view definition site maintain control over the materialization of the view.
- Views are defined in terms of queries which may reference local and non-local tables and views. Views enable full logical data independence.
- In a relational system, a view is a virtual relation, defined as the result of a query on base relations, but not materialized like a base relation, which is stored in the database.
- A view is a dynamic window in the sense that it reflects all updates to the database.
- Two types of views are recognized. Shorthand views provide the data hiding, data conversion, typing elimination and renaming functions associated with views, but are not objects of authorization.
- The user posing queries against a shorthand view must be authorized to access the objects referenced by the view.

- Protection views provide the same semantics as shorthand views and in addition are objects of authorization. Authorization to access the objects referenced by the protection view belongs to the view and the user of the view only needs the privilege to use the protection view.
- Queries referencing remotely defined shorthand views will in general execute more efficiently than identical queries with shorthand views replaced by protection views.

Views in Centralized DBMS

- View is a relation that is derived from a base relation via a query. It can involve selection, projection, aggregate functions, etc.

For example : The view of project manager derived from relation PROJECT (PNO, PTYPE, PCORDINATOR), can be defined by the following SQL query :

PDATA

PNO	PTYPE
P1	DATABASE APPLICATION
P12	BANK PROJECT
P17	MEDICAL PROJECT
P21	MECHANICAL

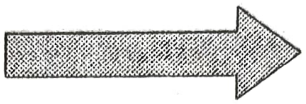
Sol. :

```
CREATE VIEW     PDATA (PNO, PCORDINATOR)
AS    SELECT PNO,  PCORDINATOR
            FROM   PROJECT
WHERE  PCORD       INATOR = "RUPALI"
```

- The result of the query defining the view is not produced.

For example : "Find the names of all the system analysts with their project number and responsibility ?"

Sol. : This involves the view SYSAN and the relation ASG(ENO, PNO, RESP, DUR)

SELECT ENAME, PNO, RESP FROM SYSAN, ASG WHERE SYSN.ENO = ASG.ENO	**is translated into** 	ELECT ENAME, PNO, RESP FROM EMP, ASG WHERE EMP.ENO = ASG.ENO AND TITLE = "Syst. Anal."

ENAME	PNO	RESP
RAKSHITA DHOTRE	P1	Manager
MAHESH AWATI	P2	Analyst
SANTOSH PATIL	P3	Analyst
RAJAN MODI	P4	Manager
RUPALI DHOTRE	P5	Analyst

- Automatic query modification is required, i.e., ANDing query qualification with view qualification.

View Updates

- Updatable view

```
CREATE VIEW   SYSAN(ENO,ENAME)
AS    SELECT  ENO,ENAME
      FROM    EMP
      WHERE   TITLE="Syst. Anal."
```

- Non-updatable view

```
CREATE VIEW   EG(ENAME,RESP)
```

```
AS    SELECT    ENAME,RESP
      FROM      EMP, ASG
      WHERE     EMP.ENO=ASG.ENO
```

View Management in DDBMS

- Views might be derived from fragments.
- View definition storage should be treated as database storage.
- Query modification results in a distributed query.
- View evaluations might be costly if base relations are distributed.
- A **materialized view** stores the tuples of a view in a database relation. Access to a materialized view is much faster than deriving the view in a distributed DBMS where base relations can be remote.
- Materialized views are actual structures stored within the database and written to disk. They are updated based on the parameters defined when they are created.
- Materialized view selection is a critical problem in many applications such as query processing, data warehousing, distributed and semantic web databases, etc.

Maintenance of Materialized Views

- **View Maintenance :** Whenever a base relation is changed, the materialized views built on it have to be updated in order to compute up-to-date query results. The process of updating a materialized view is known as view maintenance.
- Materializing a view causes it to be refreshed every time a change is made to the base tables that it references. It can be costly to rematerialize the view each time a change is made to the base tables that might affect it.

5.2 : Authentication

Q.2 Explain discretionary access control. ☞ [GTU : Summer-18, Marks 7]

OR What is authorization control ? How to imply authorization control in centralized and distributed environment.

☞ [GTU : Summer-17, Marks 7]

Ans. : • Authorization control is also called as Discretionary Access Control (DAC). It defines access rights based on the users, the type of access (e.g., SELECT, UPDATE) and the objects to be accessed.

- DAC models enforce access control based on user identities, object ownership and permission delegation. The owner of an object may delegate the permission of the object to another user.
- DAC is based on the concept of access rights or privileges for objects (tables and views), and mechanisms for giving users privileges (and revoking privileges).
- DMBS keeps track of who subsequently gains and loses privileges, and ensures that only requests from users who have the necessary privileges are allowed.
- The security levels are formed by coupling privileges with database objects. Database objects of interest are tables and views. Security at the column level is not always available. This feature would be desirable to have in an RDBMS system.
- The privileges of interest that can be granted on database objects are listed below and are commonly referred to as the CRUD privileges.

a. CREATE : The right to insert rows
b. READ : The right to select rows and read data
c. UPDATE : The right to update the contents of a row
d. DELETE : The right to delete rows

- Other privileges such as creating, dropping and altering tables are usually given to the system administrator or the user who owns the schema.
- Privileges are assigned to individual users or to a group of users. The group of users represents divisions in the real world, where people having the same role or job within an organization perform similar job functions.
- DBMSs allow discretionary access control which means that users with privileges on objects may pass on these privileges to other users.
- The privileges of the subjects over objects are recorded in the catalog (directory) as authorization rules. There are several ways to store the authorizations.
- The most convenient approach is to consider all the privileges as an authorization matrix, in which a row defines a subject, a column an object, and a matrix entry , the authorized operations.
- The authorized operations are specified by their operation type (e.g., SELECT, UPDATE).
- The authorization matrix can be stored in three ways : by row, by column, or by element.
- When the matrix is stored by row, each subject is associated with the list of objects that may be accessed together with the related access rights. This approach makes the enforcement of authorizations efficient, since all the rights of the logged-on user are together.
- The manipulation of access rights per object is not efficient since all subject profiles must be accessed.
- When the matrix is stored by column, each object is associated with the list of subjects who may access it with the corresponding access rights.

- Discretionary access control is less secure than mandatory access control. It assigns security classes to database objects and clearances to users. However, commercial DBMSs do not support mandatory access control.
- Distributed authorization rules are expressed in the same way as centralized ones. Like view definitions, they must be stored in the catalog. They can be either fully replicated at each site or stored at the sites of the referenced objects.

Q.3 List and explain requirement of database security.

Ans. : Requirements of Database Security

1. **Physical database integrity :** The data of a database should be immune to physical problems : Loss of power and Loss of machine. Journals and transaction logging can be used to restore incomplete actions.
2. **Logical database integrity :** Structure of the database should be preserved. This is usually enforced through a standardized language and the implementation of the RDBMS.
3. **Element integrity :** Data contained in each element or cell should be accurate and remain so.
4. **Auditability :** It should be possible to track who or what has access the database. This can be perform in different way.
   a. Log files generated by RDBMS.
   b. Data collected by access applications.
   c. Data collected and stored as part of the schema.
5. **Access control :** Users should be allowed to access only authorized data. Different users can be restricted to different modes of access. MySQL uses the concept of privileges.
6. **Availability :** Users can access data when they need it without deadlock, starvation or other DoS type issues.

Q.4 How to achieve reliability and integrity in database security ?

Ans. : • Integrity requires that data is protected from improper modification and integrity is lost if unauthorized changes are made by intent or by accident. Database integrity problems can have many sources. A problem may be caused by a hardware malfunction, a software bug, an attack on a system or a user error.

- The undesirable changes to a database may also be classified broadly as malicious and non-malicious.
- There are many strategies to try to avoid, detect and correct problems. Avoidance strategies include encrypting data, journaling or using read-only storage in appropriate situations. Errors may be detected by replicating or mirroring data, parity checking and the use of checksums generated by several kinds of hash functions.
- If loss of integrity is not corrected, the continued use of corrupted data could result in further damage, inaccuracy or erroneous decisions.

Q.5 How to prevent unauthorized remote access in distributed authorization control. ☞ **[GTU : Winter-17, Marks 3]**

Ans. : • To prevent remote access by unauthorized users or applications, users must also be identified and authenticated at the accessed site.

- Instead of using passwords that could be obtained from sniffing messages, encrypted certificates could be used.
- Three solutions are possible for managing authentication :
  1. Authentication information is maintained at a central site for global users which can then be authenticated only once and then accessed from multiple sites.
  2. The information for authenticating users is replicated at all sites in the catalog. Local programs, initiated at a remote site, must also indicate the user name and password.

3. All sites of the distributed DBMS identify and authenticate themselves similar to the way users do.

5.3 : Semantic Integrity Control

Q.6 What is semantic integrity control ? Which are the constraints enforce semantic integrity ?

Ans. : Semantic Integrity Control

- A database is said to be consistent if it satisfies a set of constraints, called semantic integrity constraints. In a database system, a semantic integrity subsystem (SIS) is responsible for managing and enforcing integrity constraints to ensure that these rules are not violated by the database and the database is in a consistent state.
- Semantic integrity ensures that data entered into a row reflects an allowable value for that row. The value must be within the domain or allowable set of values, for that column.
- For example, the quantity column of the items table permits only numbers. If a value outside the domain can be entered into a column, the semantic integrity of the data is violated.
- The following constraints enforce semantic integrity :
  1. Data type : The data type defines the types of values that user can store in a column. For example, the data type SMALLINT allows you to enter values from –32,767 to 32,767 into a column.
  2. Default value : The default value is the value inserted into the column when an explicit value is not specified. For example, the user_id column of the cust_calls table defaults to the login name of the user if no name is entered.
  3. Check constraint : The check constraint specifies conditions on data inserted into a column. Each row inserted into a table must meet these conditions. For example, the quantity column of the items table might check for quantities greater than or equal to one.

- DBMS must have the ability to specify and enforce correctness assertions in terms a set of semantic integrity rules.
- Integrity constraints are examined periodically, i.e., on a specific date or it is examined with every relevant update request. It provides the essential source about regularities, restrictions, rules and laws for a database extracted from the database requirements.
- When an integrity constraint is violated, several actions can be undertaken, for example raise an exception, print a message for the user of the system, dump a program trace or even stop the application. These actions leave the database in an inconsistent state. Therefore, the programmer can specify some actions to enforce the constraint.

Semantic Integrity Constraints

- A relational DBE supports four basic types of semantic integrity constraints or rules :
  1. Data type constraints
  2. Relation constraints
  3. Referential constraints
  4. Explicit constraints
- **Data type constraints** are semantic integrity rules that specify the data type for columns of relational tables. Some relational database systems may allow specification of user-defined data types.
- **Relation constraints** are the methods used to define a relation or a table.
- Referential integrity constraints restrict the values stored in the database.
- Explicit constraints are not inherited from the ERM like the other three constraints.

Q.7 What do you mean by distributed semantic integrity control ? Explain with example. ☞ [GTU : Summer-17, Marks 7]

Ans. : Distributed Semantic Integrity Control

- Distributed databases exacerbate the problem of constraint maintenance. Owing to horizontal and vertical fragmentation of relations with the fragments stored at different locations, the integrity constraints must be translated into constraints on the fragments.
- Thus there are a lot more constraints to maintain when databases get distributed. In addition, replication of data imposes a constraint that the replicas have equal values at all times.
- Challenges in distributed semantic integrity control :
  1. Creation of multisite semantic integrity rules
  2. Enforcement of multisite semantic integrity rules
  3. Maintaining mutual consistency of local semantic integrity rules for each copy of data
  4. How to maintain consistency of local and global semantic integrity rules

Compile Time Validation

- Transaction compile time enforcement, meaning that the execution of a transaction is not started before all constraints are evaluated to hold; thus, the correctness of the transaction is established before it is actually executed;
- In this validation method, transaction is given permission to run only after all the semantic integrity constraints have been validated. Semantic integrity data is locked because they are not changed during transaction execution.
- Compile time validation is simple to implement.
- There is no cost for abort operations.
- A main constraint is after a transaction has been validated, it starts its execution.

Run Time Validation

- Transaction run time enforcement, meaning that the constraints are enforced during transaction execution while the actual updates have not yet been carried out
- There is no need to hold transactions back until they are validated. Transactions are validated during execution.
- The transaction is rolled back when semantic integrity rules are violated.
- Invalid transactions need to be rolled back.

Post execution Time Validation

- Post transaction execution enforcement, in which the constraints are evaluated after all updates have been performed against the database
- Validation is performed after the execution of the transaction but just before the transaction is committed.
- If validation fails, transaction will be aborted.

Q.8 What do you mean by set-oriented assertions ?

☞ [GTU : Winter-17, Marks 3]

Ans. : • Set-oriented constraint are multivariable; that is, they involve join predicates. Although the assertion predicate may be multirelation, a pretest is associated with a single relation.

- The constraint definition can be sent to all the sites that store a fragment referenced by these variables.
- Compatibility checking also involves fragments of the relation used in the join predicate.
- Predicate compatibility is useless here, because it is impossible to infer that a fragment predicate p is false if the constraint C is true. Therefore C must be checked for compatibility against the data.
- This compatibility check basically requires joining each fragment of the relation (R), with all fragments of the other relation (S), involved in the constraint predicate.
- This operation may be expensive and, as any join, should be optimized by the distributed query processor.

5.4 : Cost of Enforcing Semantic Integrity

Q.9 Describe cost of enforcing semantic integrity.

Ans. : In a distributed system, the validation cost is dominated by the communication cost between the sites involved in a transaction. The costs of the above three methods are measured only in I/O costs.

- Run time constraint enforcement is superior to any of the other methods examined for realistic database operations (i.e. with most transactions committing). In some situations, compile time enforcement yields a better performance

Semantic Integrity Enforcement Cost in Distributed System

- In calculating the cost of semantic integrity enforcement in distributed systems, the cost of the CPU is negligible as compared to the communication cost.
- Communication cost is directly related to the number of messages required for each approach. This assumption ignores the amount of data transferred by.

Constraint optimization

- The execution cost of constraint enforcement is one of the major problems in the field of constraint handling. Therefore, constraint optimization is of great importance. Various types of optimization can be applied to integrity constraints :
  1. The amount of data to be checked in constraint enforcement can be reduced by evaluating the constraints only on the relevant parts of relations.
  2. The constraint rules can be manipulated algebraically to obtain equivalent rules that can be evaluated cheaper.
  3. In a distributed system with fragmented relations, fragmentation knowledge can be used to reduce the number of fragments used in a constraint.
  4. The constraint rules can be transformed using semantic knowledge about the application being modelled by the database and about the database itself.

END...

6 QUERY PROCESSING

6.1 : Query Processing Problem

Q.1 What is query processing ? Explain in detail

Ans. : • Queries are one of the things that make databases so powerful. A "query" refers to the action of retrieving data from your database. Query is expressed by using high level language (Structured Query Language).

- A query is language expressions that describe data to be retrieved from a database.
- Query processing is an important concern in the field of distributed databases. The main problem is : if a query can be decomposed into sub-queries that require operations at geographically separated databases, determine the sequence and the sites for performing this set of operations such that the communication cost and processing cost for processing this query is minimized.
- The problem is complicated by the fact that query processing not only depends on the operations of the query, but also on the parameter values associated with the query. Distributed query processing is an important factor in the overall performance of a distributed database system
- Query processing in a distributed system requires the transmission of data between computers in a network. The arrangement of data transmissions and local data processing is known as a distribution strategy for a query.
- The retrieval of data from different sites in a network is known as distributed query processing.

- The difference between query processing in a centralized database and a distributed database is the potential for decomposing a query into sub-queries which can be processed in parallel, and their intermediate results can be sent in parallel to the required computers.

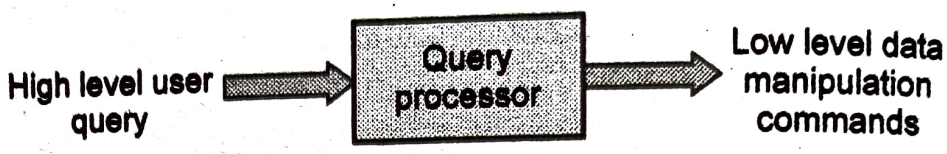


Fig. Q.1.1 Query processing

- The role of a distributed query processor is to map a high level query on a distributed database (a set of global relations) into a sequence of database operations (of relational algebra) on relational fragments
- A distributed database system is characterized by the distribution of the system components of hardware, control and data.
- Complex query can be executed in many different ways but main objective of query processing is to determine which one is the most cost effective. This complicated task is performed by query processor.
- A query processor is a module in the DBMS that performs the tasks to process, to optimize, and to generate execution strategy for a high-level query.

Q.2 Explain characterization of query processors.

☞ [GTU : Summer-18, Marks 4]

Ans. : Characterization of Query Processors

1. The input language to the query processor can be based on relational calculus or relational algebra.
2. Query optimization is to select a best point of solution space that leads to the minimum cost.
3. Optimization can be done statically before executing the query or dynamically as the query is executed.

4. Dynamic query optimization requires statistics in order to choose the operation that has to be done first.
5. Static query optimization requires statistics to estimate the size of intermediate relations
6. The semi-join operation reduces the size of the data that are exchanged between the sites so that the communication cost can be reduced.
7. Distributed query processor exploits the network topology.

Q.3 Explain the objectives of query processing in detail.

☞ **[GTU : Winter-16, 17, Marks 7]**

Ans. : • The objective of query processing in a distributed context is to transform a high-level query on a distributed database, which is seen as a single database by the users, into an efficient execution strategy expressed in a low-level language on local databases.

- An important aspect of query processing is query optimization.
- A good measure of resource consumption is the total cost that will be incurred in processing the query. Total cost is the sum of all times incurred in processing the operators of the query at various sites and in inter-site communication.
- Another good measure is the response time of the query , which is the time elapsed for executing the query.
- Since operators can be executed in parallel at different sites, the response time of a query may be significantly less than its total cost.
- In a distributed database system, the total cost to be minimized includes CPU, I/O, and communication costs. The CPU cost is incurred when performing operators on data in main memory.
- The I/O cost is the time necessary for disk accesses. This cost can be minimized by reducing the number of disk accesses through fast access methods to the data and efficient use of main memory.

- The communication cost is the time needed for exchanging data between sites participating in the execution of the query. This cost is incurred in processing the messages and in transmitting the data on the communication network.
- The aim of distributed query optimization reduces to the problem of minimizing communication costs generally at the expense of local processing.
- The advantage is that local optimization can be done independently using the known methods for centralized systems.

Q.4 Explain Layers of Query Processing.

☞ [GTU : Winter-17, Marks 7, Summer-18, Marks 4]

Ans. : Layers of Query Processing

- Fig. Q.4.1 shows phases of query processing.

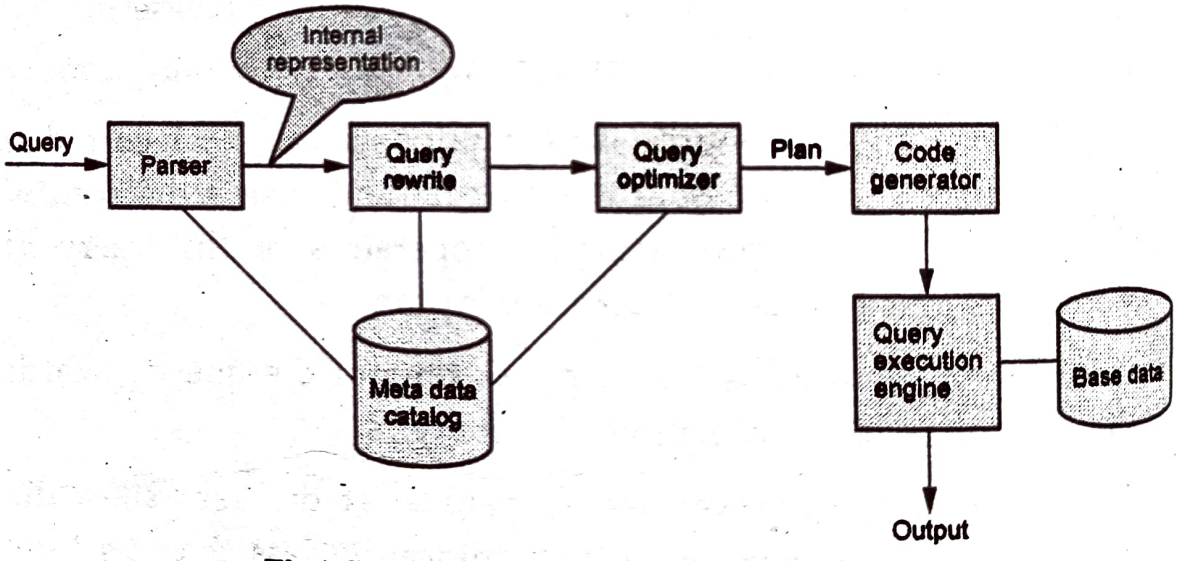


Fig. Q.4.1 Phases of query processing

1. **Parser :** In the first phase, the query is parsed and translated into an internal representation that can be easily processed by the later phases. The parsing and translation will first translate the query into its internal form, then translate the query into relational algebra and verifies relations.
2. **Query rewrite :** It transforms a query in order to carry out optimizations that are good regardless of the physical state of the system. It contains the size of tables, presence of indices, locations of copies of tables, speed of machines, etc.

3. **Query optimizer :** This component carries out optimizations that depend on the physical state of the system. The optimizer decides which indices to use to execute a query, which methods (e.g., hashing or sorting) to use to execute the operations of a query (e.g., joins and group-bys), and in which order to execute the operations of a query. It also decides how much main memory to allocate for the execution of each operation.
4. **Plan :** A plan specifies precisely how the query is to be executed. Probably every database system represents plans in the same way as trees. The nodes of a plan are operators, and every operator carries out one particular operation.
5. **Code generation :** Code generation is also called plan refinement. This component transforms the plan produced by the optimizer into an executable plan. In some systems, plan refinement also involves carrying out simple optimizations which are not carried out by the query optimizer in order to simplify the implementation of the query optimizer.
6. **Query execution engine :** This component provides generic implementations for every operator.
7. **Catalog :** The catalog stores all the information needed in order to parse, rewrite, and optimize a query. It maintains the schema of the database.

- The query execution engine is like a virtual machine that runs physical query plans.

6.2 : Query Processing in Centralized Systems

Q.5 Draw and explain query processing in centralized system.

☞ [GTU : Summer-17, Marks 7]

Ans. : Query Processing in Centralized Systems

- Goals of query processor in centralized system is as follows :
  1. Minimize the query response time.
  2. Maximize the parallelism in the system.
  3. Maximize the system throughput.
  4. Minimize the total resources used

- In centralized DBMS, query processing consists of four steps :
  1. Query decomposition
  2. Query optimization
  3. Code generation
  4. Query execution
- Fig. Q.5.1 shows steps for query processing in centralized DBMS.

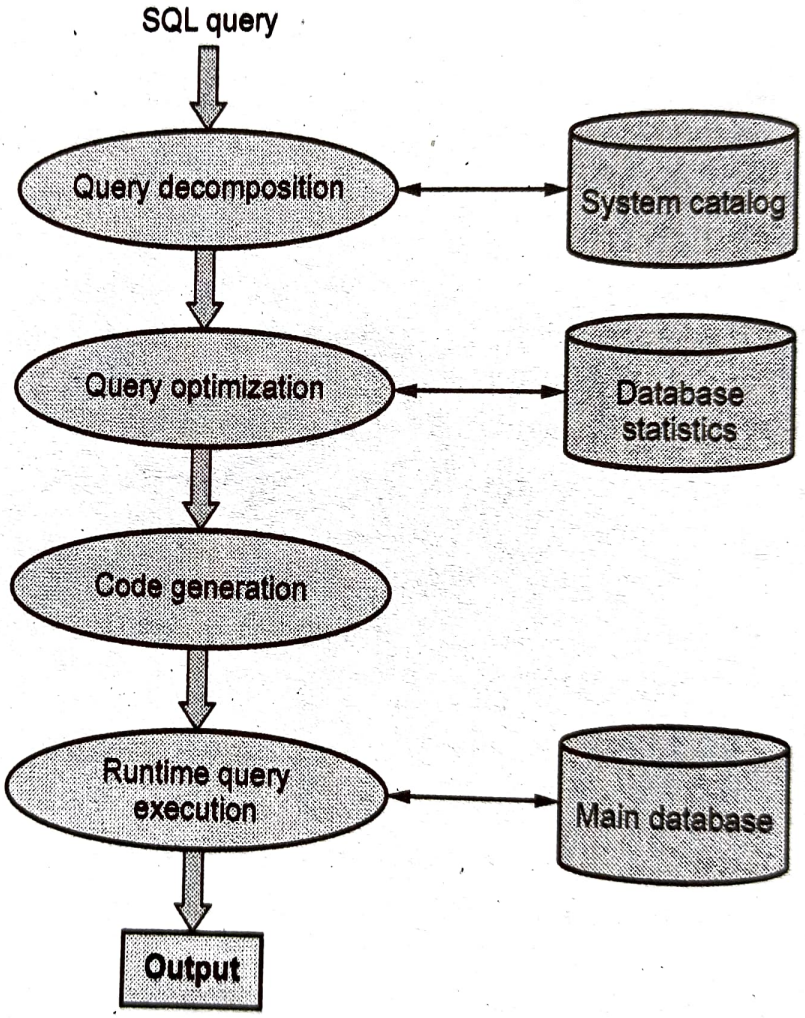


Fig. Q.5.1 Steps for query processing in centralized DBMS

- **Query decomposition** : It consists of scanning, parsing and validation steps. The query parser checks the validity of the query and then translates it into an internal form. Translate the query into an equivalent relational algebra expression. Validation is done to make sure that all relations and their attributes are valid and meaningful

- **Query optimization :** Generate an optimal evaluation plan for the query plan. There are generally many different methods that can be used to process a query and compute the result. Query optimization is the process of selecting the most efficient method for computing the result.
- **Code generation :** Once an optimal execution plan is produced by the query optimizer, it is the job of the code generator to generate the code for executing the plan.
- **Query execution :** Runtime database processor is responsible for executing the code, whether in compiled mode or interpreted mode to produce the response to the query.
- In a centralized system, the catalog contains dictionary information about tables, indexes, views, and columns associated with each table or index. The catalog also contains statistics about the structures in the database.
- A system may store the number of pages used by each relation and indexes, the number of rows per page for a given relation, the number of unique values in the key columns of a given relation, the types of keys, the number of leaf index pages, and so on.

6.3 : Example Query Processing in Distributed Systems

Q.6 Write a short note on : phases of distributed query processing.
☞ [GTU : Summer-17, Marks 7]

Ans. : Example Query Processing in Distributed Systems

- In a **distributed DBMS** the catalog has to store additional information including the location of relations and their replicas. The catalog must also include system wide information such as the number of sites in the system along with their identifiers.

- Fig. Q.6.1 shows distributed query processing architecture.

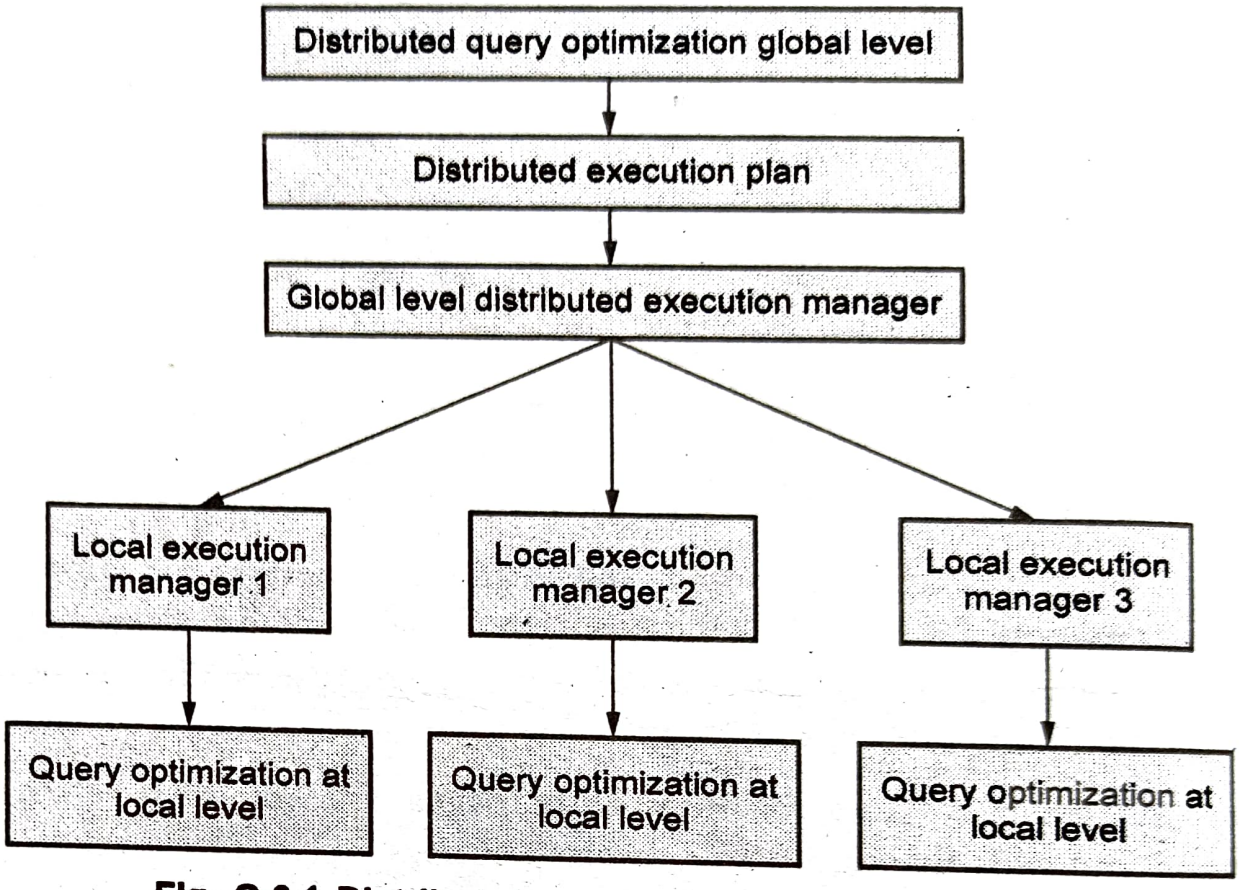


Fig. Q.6.1 Distributed query processing architecture

- Client site : Query enters into the system. It is also called controlling site. This site validates user or application.
- Client site checks syntax of query and take decision to forward or reject. If forward the correct query and discard the incorrect query.
- Forwarded query is translated to relational algebra; and to globally optimize the query.

Q.7 How to map global query to local query ?

Ans. : Mapping Global Query to Local

- Local query optimization is for single database commands. Problem of this type of optimization is how to translate a "what representation" into an efficient "how representation". There are several methods for this type of optimization :

1. Cost evaluation method
2. Heuristic rule method
3. Hybrid method

- In a distributed database system, processing a query comprises of optimization at both the global and the local level. The query enters the database system at the client or controlling site. Here, the user is validated, the query is checked, translated, and optimized at a global level.

- The process of mapping global queries to local is as follows :

1. The tables required in a global query have fragments distributed across multiple sites. The local databases have information only about local data. The controlling site uses the global data dictionary to gather information about the distribution and reconstructs the global view from the fragments.
2. If there is no replication, the global optimizer runs local queries at the sites where the fragments are stored. If there is replication, the global optimizer selects the site based upon communication cost, workload, and server speed.
3. The global optimizer generates a distributed execution plan.
4. The local queries are optimized by the local database servers.

END...