UNIT-6

Query Processing

Outlines…

- Introduction of Query Processing
- Query Processing Problem
- Layer of Query Processing
- Query Processing in Centralized Systems
- Query Processing in Distributed Systems

Introduction of Query Processing

- 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.

- The main function of a relational query processor is to transform a high-level query (typically, in relational calculus) into an equivalent lower-level query (typically, in some variation of relational algebra).

Query Processing Problem

- The main difficulty is to select the execution strategy that minimizes resource consumption.
- The low-level query actually implements the execution strategy for the query. The transformation must achieve both correctness and efficiency.
- It is correct if the low-level query has the same semantics as the original query, that is, if both queries produce the same result.
- The well-defined mapping from **relational calculus** to **relational algebra** makes the correctness issue easy.

Query Processing Example

- **Example:** Transformation of an SQL-query into an RA-query.
  Relations: EMP(ENO, ENAME, TITLE), ASG(ENO,PNO,RESP,DUR)
  Query: *Find the names of employees who are managing a project?*

– **High level query**

```
SELECT ENAME
FROM EMP,ASG
WHERE EMP.ENO = ASG.ENO AND DUR > 37
```

– **Two possible transformations of the query are:**

- Expression 1: $\text{ENAME}(\text{DUR}>37 \wedge \text{EMP.ENO}=\text{ASG.ENO}(\text{EMP} \times \text{ASG}))$
- Expression 2: $\text{ENAME}(\text{EMP} \rtimes_{\text{ENO}} (\text{DUR}>37(\text{ASG})))$

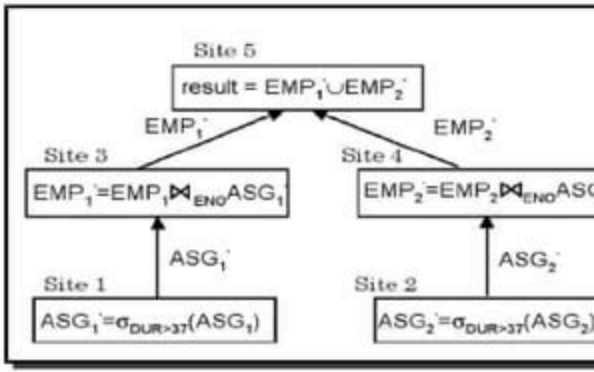
– Expression 2 avoids the expensive and large intermediate Cartesian product, and therefore typically is better.

Query Processing Example

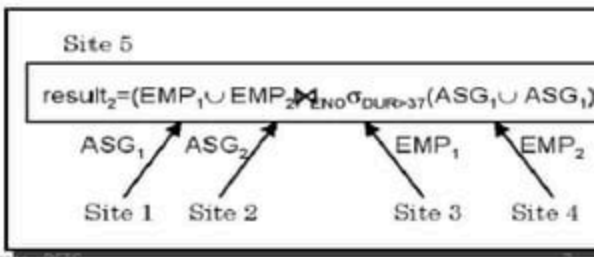
- We make the following assumptions about the data fragmentation
  - **Data is (horizontally) fragmented:**
- Site1: ASG1 = ENO≤"E3"(ASG)
- Site2: ASG2 = ENO>"E3"(ASG)
- Site3: EMP1 = ENO≤"E3"(EMP)
- Site4: EMP2 = ENO>"E3"(EMP)
- Site5: Result
- Relations ASG and EMP are fragmented in the same way
- Relations ASG and EMP are locally clustered on attributes RESP and ENO,
- respectively

Query Processing Example . . .

- Now consider the expression $\Pi_{ENAME}(EMP \bowtie_{ENO} (\sigma_{DUR>37}(ASG)))$
- Strategy 1 (partially parallel execution):
  - Produce ASG'_1 and move to Site 3
  - Produce ASG'_2 and move to Site 4
  - Join ASG'_1 with EMP_1 at Site 3 and move the result to Site 5
  - Join ASG'_2 with EMP_2 at Site 4 and move the result to Site 5
  - Union the result in Site 5



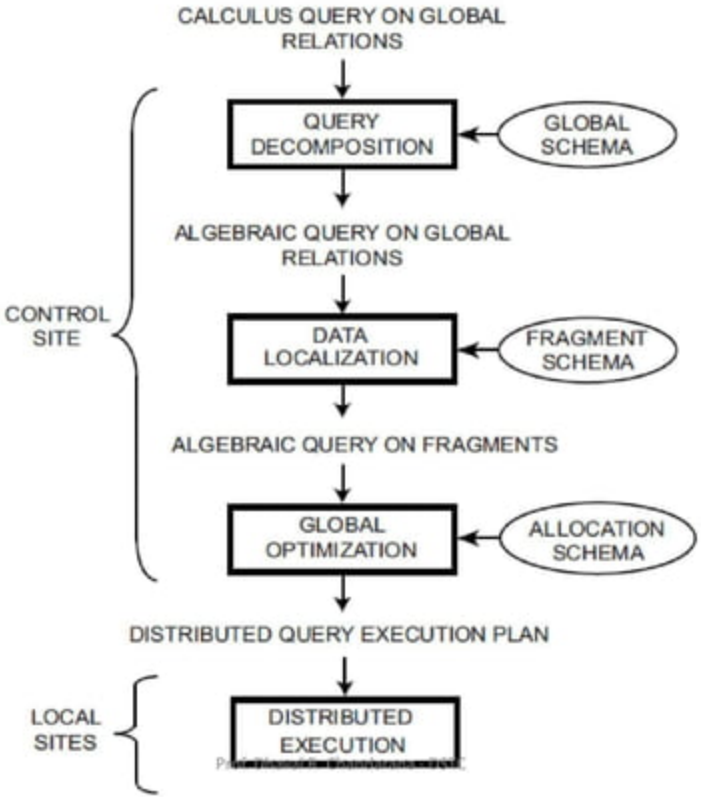
- Strategy 2:
  - Move ASG_1 and ASG_2 to Site 5
  - Move EMP_1 and EMP_2 to Site 5
  - Select and join at Site 5



- For simplicity, the final projection is omitted.

Query Processing Example . . .

- Calculate the cost of the two strategies under the following assumptions:
  - Tuples are uniformly distributed to the fragments; 20 tuples satisfy DUR>37
  - size(EMP) = 400, size(ASG) = 1000
  - tuple access cost = 1 unit; tuple transfer cost = 10 units
  - ASG and EMP have a local index on DUR and ENO
- Strategy 1
  - Produce ASG's: (10+10) * tuple access cost 2
  - Transfer ASG's to the sites of EMP's: (10+10) * tuple transfer cost 20
  - Produce EMP's: (10+10) * tuple access cost * 2 4
  - Transfer EMP's to result site: (10+10) * tuple transfer cost 20
  - Total cost 46
- Strategy 2
  - Transfer EMP_1, EMP_2 to site 5: 400 * tuple transfer cost 4,00
  - Transfer ASG_1, ASG_2 to site 5: 1000 * tuple transfer cost 10,00
  - Select tuples from $ASG_1 \cup ASG_2$: 1000 * tuple access cost 1,00
  - Join EMP and ASG': 400 * 20 * tuple access cost 8,00
  - Total cost 23,00


CALCULUS QUERY ON GLOBAL RELATIONS
QUERY DECOMPOSITION
GLOBAL SCHEMA
ALGEBRAIC QUERY ON GLOBAL RELATIONS
CONTROL SITE
DATA LOCALIZATION
FRAGMENT SCHEMA
ALGEBRAIC QUERY ON FRAGMENTS
GLOBAL OPTIMIZATION
ALLOCATION SCHEMA
DISTRIBUTED QUERY EXECUTION PLAN
LOCAL SITES
DISTRIBUTED EXECUTION

Query Decomposition

- The first layer decomposes the calculus query into an algebraic query on global relations. The information needed for this transformation is found in the global conceptual schema describing the global relations.
- Query decomposition can be viewed as four successive steps.
- **First**, the calculus query is rewritten in a normalized form that is suitable for subsequent manipulation. Normalization of a query generally involves the manipulation of the query quantifiers and of the query qualification by applying logical operator priority.
- **Second**, the normalized query is analyzed semantically so that incorrect queries are detected and rejected as early as possible. Techniques to detect incorrect queries exist only for a subset of relational calculus. Typically, they use some sort of graph that captures the semantics of the query.

Query Decomposition

- **Third**, the correct query (still expressed in relational calculus) is simplified. One way to simplify a query is to eliminate redundant predicates. Note that redundant queries are likely to arise when a query is the result of system transformations applied to the user query. such transformations are used for performing semantic data control (views, protection, and semantic integrity control).
- **Fourth**, the calculus query is restructured as an algebraic query. The traditional way to do this transformation toward a "better" algebraic specification is to start with an initial algebraic query and transform it in order to find a "go
- The algebraic query generated by this layer is good in the sense that the worse executions are typically avoided.

Data Localization

- The input to the second layer is an algebraic query on global relations. The main role of the second layer is to localize the query's data using data distribution information in the fragment schema.
- This layer determines which fragments are involved in the query and transforms the distributed query into a query on fragments.
- A global relation can be reconstructed by applying the fragmentation rules, and then deriving a program, called a localization program, of relational algebra operators, which then act on fragments.
- Generating a query on fragments is done in two steps
  - **First**, the query is mapped into a fragment query by substituting each relation by its reconstruction program (also called materialization program).
  - **Second**, the fragment query is simplified and restructured to produce another "good" query.

Global Query Optimization

- The input to the third layer is an algebraic query on fragments. The goal of query optimization is to find an execution strategy for the query which is close to optimal.
- The previous layers have already optimized the query, for example, by eliminating redundant expressions. However, this optimization is independent of fragment characteristics such as fragment allocation and cardinalities.
- **Query optimization consists of finding the "best" ordering of operators in the query, including communication operators that minimize a cost function.**
- The output of the query optimization layer is a optimized algebraic query with communication operators included on fragments. It is typically represented and saved (for future executions) as a distributed query execution plan .

Distributed Query Execution

- The last layer is performed by all the sites having fragments involved in the query.
- Each sub query executing at one site, called a local query, is then optimized using the local schema of the site and executed.

Characterization of Query Processors

- The input language to the query processor can be based on relational calculus or relational algebra.
- Query optimization is to select a best point of solution space that leads to the minimum cost.
- Optimization can be done statically before executing the query or dynamically as the query is executed.
- Dynamic query optimization requires statistics in order to choose the operation that has to be done first.
- Static query optimization requires statistics to estimate the size of intermediate relations.
- Distributed query processor exploits the network topology.

Query Processing in Centralized Systems

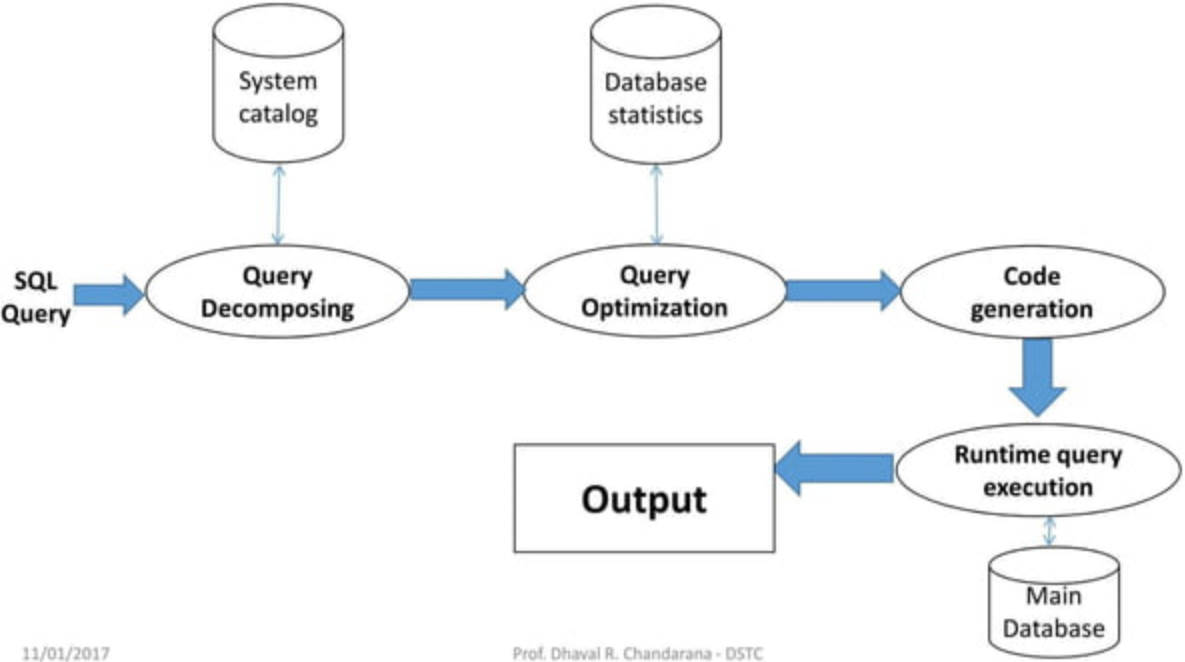
- **Goal of query processor in centralized system is:**

1. Minimize the query response time.
2. Maximize the parallelism in the system
3. Maximize the system throughput.

- In centralized DBMS query processing consists of four steps:

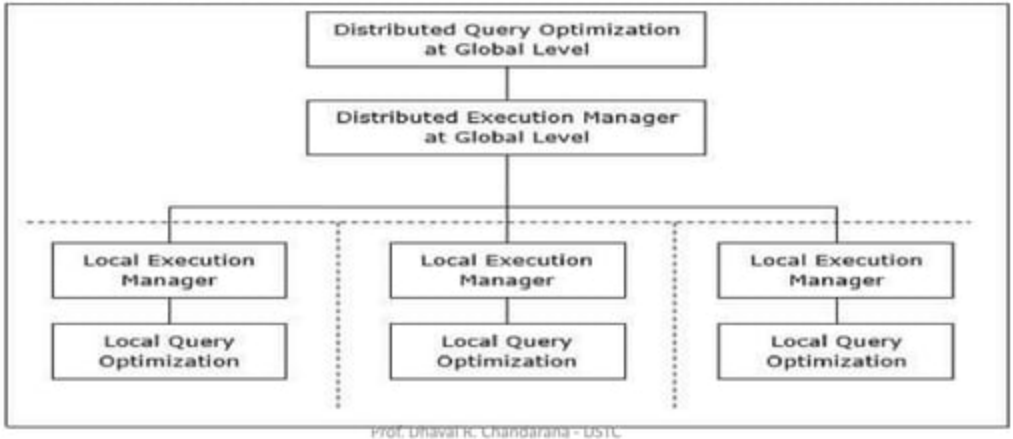
1. Query decomposition
2. Query optimization
3. Code generation
4. Query execution

Query Processing in Centralized Systems



Query Processing in Distributed Systems

In a distributed DBMS the catalog has to store additional informatio including the location of relations and their replicas. The catalog must als include system wise information such as the number of site in the syster along with their identifiers'.



Mapping Global Query to Local

The tables required in a global query have fragments distributed acros multiple sites. The local databases have information only about local data The controlling site uses the global data dictionary to gather informatio about the distribution and reconstructs the global view from th fragments.

If there is no replication, the global optimizer runs local queries at the site where the fragments are stored. If there is replication, the global optimize selects the site based upon communication cost, workload, and serve speed.

Mapping Global Query to Local

The global optimizer generates a distributed execution plan so that leas amount of data transfer occurs across the sites. The plan states th location of the fragments, order in which query steps needs to b executed and the processes involved in transferring intermediate results.

The local queries are optimized by the local database servers. Finally, th local query results are merged together through union operation in case o horizontal fragments and join operation for vertical fragments.

Example

For example, let us consider that the following Project schema is horizontally fragmented according to City, the cities being New Delhi, Kolkata and Hyderabad.

PROJECT

Pid	City	Department	Status

Suppose there is a query to retrieve details of all projects whose status is "Ongoing".

The global query will be

$$\sigma_{status=\text{"ongoing"}}(PROJECT)$$

Example

Query in New Delhi's server will be

$$\sigma_{status="ongoing"}(NewD-PROJECT)$$

Query in Kolkata's server will be

$$\sigma_{status="ongoing"}(Kol-PROJECT)$$

Query in Hyderabad's server will be

$$\sigma_{status="ongoing"}(Hyd-PROJECT)$$

In order to get the overall result, we need to union the results of the three queries as follows

$$\sigma_{status="ongoing"}(NewD-PROJECT) \cup \sigma_{status="ongoing"}(kol-PROJECT) \cup \sigma_{status="ongoing"}(Hyd-PROJECT)$$

Important Question

- Q:1 Explain Layer of Query Processing.
- Q:2 Explain Query Processing in Centralized System.
- Q:3 Explain Query Processing in Distributed System.
- Q:4 Explain Query Processing Problem.