BITS Pilani

BITS Pilani
Hyderabad Campus

Dr. Manik Gupta
Associate Professor
Department of CSIS

innovate achieve lead

BITS Pilani
Hyderabad Campus

Distributed Data Systems (CS G544)
Lecture 13-14

Thursday, 12th Sept 2024

Lecture Recap

- Distributed Database Design
  - Fragmentation
  - Allocation

Introduction

- What is the reason for success of RDBMS?
  - The success of relational database technology is due in part to **availability of non-procedural query languages** like SQL, which can significantly improve application development and end-user productivity.
  - User does not have to define the procedure to extract the answer.
  - A specific module of DBMS, called as **Query Processor** will do it.
  - This relieves the user from query-optimization, which is a time-consuming and complex task.
  - Query processor will do this optimization in an efficient way since it exploits a large amount of information about the data.

Introduction

- Query-processing is much more complicated in distributed environments.
  - This is because a large number of parameters affect the performance of distributed queries.
  - Important factors are
    - Fragmentation
    - Replication
  - Further, with many sites to access, **query response time** may become very **high**.

Distributed query processing

What is the role of a distributed query processor?

- The role of a distributed query processor is to **map a high-level query** (assumed to be expressed in relational calculus) on a distributed database (i.e., a set of global relations) into a **sequence of database operators (of relational algebra)** on relation fragments.

Distributed query processing

- Several important functions characterize this mapping.
  - First the calculus query must be **decomposed** into a sequence of relational operations called an algebraic query.
  - Second, the data accessed by the query must be **localized** so that the operations on relations are translated to bear on local data (fragments).
  - Finally, the algebraic query on fragments must be extended with **communication operations** and **optimized** with respect to a **cost function** to be minimized. This cost function typically refers to computing resources such as disk I/Os, CPUs, and communication NWs.

Query Processing Problem

- The main function of relational query processor is to transform a high-level query into an equivalent lower-level query.
- The low-level query actually implements the execution strategy for the query.
  - The above transformation must achieve both correctness and efficiency.
  - A relational calculus query may have **many equivalent and correct transformations** into relational algebra.
  - Since each equivalent execution strategy can lead to very **different consumptions of computer resources**, the main difficulty is to **select the execution strategy that minimizes resources consumption**.

Example

EMP(ENO, ENAME, TITLE)
ASG(ENO, PNO, RESP, DUR)

- "Find the names of employees who are managing a project"
- The expression of the query in relational calculus using the SQL syntax is

```
SELECT ENAME
FROM EMP,ASG
WHERE EMP.ENO = ASG.ENO AND RESP = "Manager"
```

- Two equivalent relational algebra queries that are correct transformations of the query above are as follows

$$\Pi_{\text{ENAME}}(\sigma_{\text{RESP="Manager"} \wedge \text{EMP.ENO=ASG.ENO}}(\text{EMP} \times \text{ASG}))$$

and

$$\Pi_{\text{ENAME}}(\text{EMP} \bowtie_{\text{ENO}} (\sigma_{\text{RESP="Manager"}}(\text{ASG})))$$

- It is intuitively obvious that the second query, which avoids the Cartesian product of EMP and ASG, consumes much less computing resources than the first, and thus should be retained.

Centralized context

- In a centralized context, query execution strategies can be well expressed in an extension of relational algebra.
- The main role of a centralized query processor is to choose, for a given query, the **best relational algebra query** among all equivalent ones.

Challenge in distributed system

- In a distributed system, relational algebra is not enough to express execution strategies.
- It must be **supplemented with operators for exchanging data between sites**.
- Besides the **choice of ordering relational algebra operators**, the distributed query processor must also select the **best sites to process data**, and possibly the way data should be transformed.
- This **increases the solution space** from which to choose the distributed execution strategy, making distributed query processing significantly more difficult.

Example

- Lets consider the following query

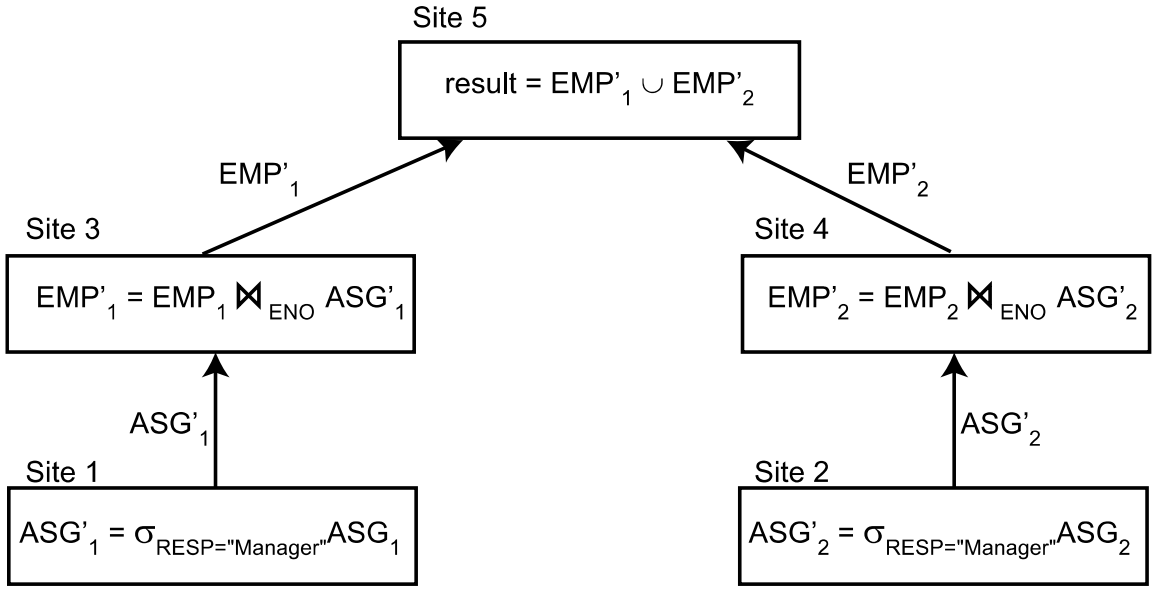
$$\Pi_{\text{ENAME}}\,(\text{EMP} \bowtie_{\text{ENO}} (\sigma_{\text{RESP}=\text{“Manager”}}\,(\text{ASG})))$$

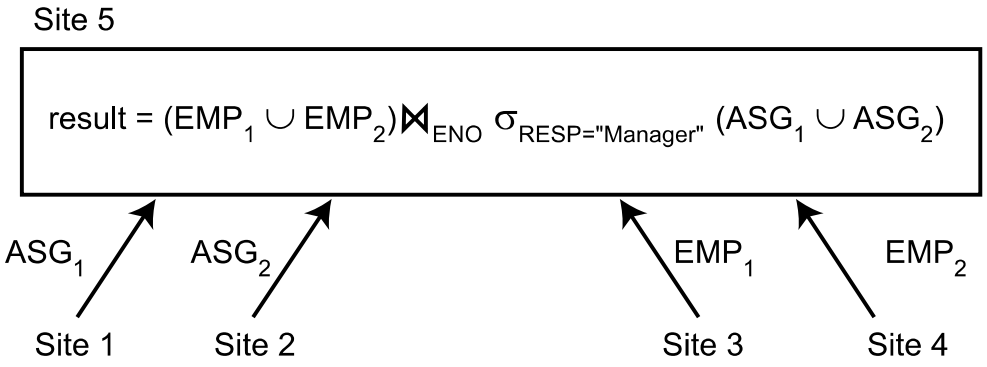
- Assume that relations EMP and ASG are horizontally fragmented as follows:

$$\text{EMP}_1 = \sigma_{\text{ENO} \leq \text{“E3”}}\,(\text{EMP})$$
$$\text{EMP}_2 = \sigma_{\text{ENO} > \text{“E3”}}(\text{EMP})$$
$$\text{ASG}_1 = \sigma_{\text{ENO} \leq \text{“E3”}}(\text{ASG})$$
$$\text{ASG}_2 = \sigma_{\text{ENO} > \text{“E3”}}(\text{ASG})$$

- Fragments ASG1, ASG2, EMP1, and EMP2 are stored at sites 1, 2, 3, and 4, respectively, and the result is expected at site 5.

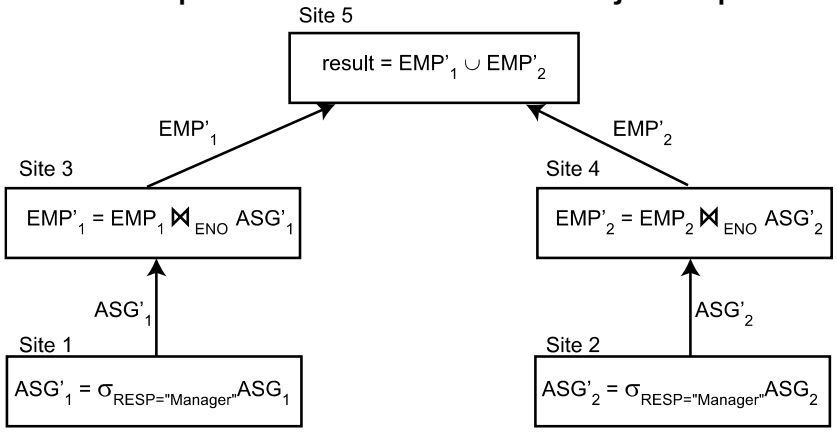
Alternate strategies





Strategy A

- An arrow from site i to site j labeled with R indicates that relation R is transferred from site i to site j. Strategy A exploits the fact that relations EMP and ASG are fragmented the same way in order to perform the select and join operator in parallel.



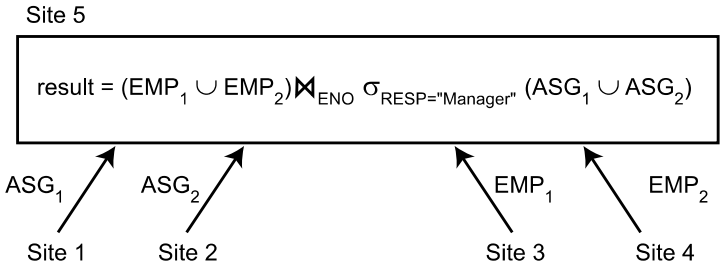
- We use simple cost model to evaluate resource consumption: We assume that a tuple access, denoted by *tupacc*, is 1 unit (which we leave unspecified) and a tuple transfer, denoted *tuptrans*, is 10 units. We assume that relations EMP and ASG have 400 and 1000 tuples, respectively, and that there are 20 managers in relation ASG.

1. Produce ASG′ by selecting ASG requires $(10+10)*tupacc$ = 20
2. Transfer ASG′ to the sites of EMP requires $(10+10)*tuptrans$ = 200
3. Produce EMP′ by joining ASG′ and EMP requires $(10+10)*tupacc*2$ = 40
4. Transfer EMP′ to result site requires $(10+10)*tuptrans$ = 200

The total cost is 460

Strategy B

- Strategy B centralizes all the operand data at the result site before processing the query.



The cost of strategy B can be derived as follows:

1. Transfer EMP to site 5 requires $400 * tuptrans$	$= 4,000$
2. Transfer ASG to site 5 requires $1000 * tuptrans$	$= 10,000$
3. Produce ASG′ by selecting ASG requires $1000 * tupacc$	$= 1,000$
4. Join EMP and ASG′ requires $400 * 20 * tupacc$	$= 8,000$
The total cost is	$23,000$

Objectives of query processing

- 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**. Because many execution strategies are correct transformations of the same high-level query, the one that optimizes (minimizes) resource consumption should be retained.

Objectives of query processing

- 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.
- In a distributed database system, the total cost to be minimized includes CPU, I/O, and communication costs.

Complexity of Relational Algebra Operations

- The complexity of relational algebra operators, which directly affects their execution time, is important to a query processor. These principles can help in choosing the final execution strategy.
- The simplest way of defining complexity is in terms of relation cardinalities independent of physical implementation details such as fragmentation and storage structures.
  - Complexity is ***O*(*n*) for unary operators**, where *n* denotes the relation cardinality, if the resulting tuples may be obtained independently of each other.
  - Complexity is ***O*(*n* * log*n*) for binary operators** if each tuple of one relation must be compared with each tuple of the other on the basis of the equality of selected attributes. This complexity assumes that tuples of each relation must be sorted on the comparison attributes.
  - Complexity is ***O*(n^2) for the Cartesian product** of two relations because each tuple of one relation must be combined with each tuple of the other.

Complexity of Relational Algebra Operations

Operation	Complexity
Select Project (without duplicate elimination)	$O(n)$
Project (with duplicate elimination) Group by	$O(n*\log n)$
Join Semijoin Division Set Operators	$O(n*\log n)$
Cartesian Product	$O(n^2)$

- This simple look at operator complexity suggests two principles.
  - First, because complexity is relative to relation cardinalities, the most selective operators that reduce cardinalities (e.g., selection) should be performed first.
  - Second, operators should be ordered by increasing complexity so that Cartesian products can be avoided or delayed.

Characterization of Query Processors

- There are some important characteristics of query processors that can be used as a basis for comparison in centralized and distributed systems.

Characterization of Query Processors

Languages

- High-level languages of Relational model give the system many opportunities for query optimization. Input language can be RC or RA.
- In a distributed context, the output language is generally some internal form of relational algebra augmented with communication primitives.

Characterization of Query Processors

innovate achieve lead

Types of optimization

- Conceptually, query optimization aims at choosing the best point in the solution space of all possible execution strategies.
  - **Exhaustive** search: to predict the cost, and select a strategy with minimum cost. This is very costly.
  - **Randomized** approach: Try to find very good solution, not necessarily the best, but avoid high cost of optimization.
  - **Heuristic** approach: This is to restrict the solution space so that only few strategies are considered. A common heuristic is to **minimize the size of intermediate relations**.
    - **perform unary operations first** and ordering the binary operators by the increasing sizes of their intermediate relations
    - **perform semijoin instead of join operations** to minimize data communication

Characterization of Query Processors

Optimization Timing

- A query may be optimized at different timings relative to the actual time of query execution. It can be done either statically before execution, or can be done dynamically during the execution.
  - **Static query optimization** is done at query compilation time. Thus the cost of optimization may be amortized over multiple query executions and best suited for exhaustive search.
  - In **dynamic query optimization**, the next operation can be based on the accurate knowledge about the result of the previous operation, hence database statistics are not needed. As the actual size of intermediate results is available, we can minimize bad choices.

Characterization of Query Processors

Statistics

•The effectiveness of the optimization relies on the statistics of the database.

•In distributed databases, statistics for query optimization typically bear on the fragments, cardinality, distinct values for a given attribute etc.

Decision sites

•When static optimization is used, single site or several sites may participate in deciding a strategy.

•Most systems use centralized decision making approach. It is simpler but requires knowledge about entire distributed database. While the distributed approach requires only local information. Hybrid approach where one site makes major decision and others make local decisions, may be used in some cases.

Characterization of Query Processors

Exploitation of the network topology

- The cost function to be minimized may be restricted to the network communication, which is a dominating factor.
- Now the query processing can be divided into two problems:
    - •Selection of global execution strategy, based on intersite communication.
    - •Selection of local execution strategy based on centralized algorithms.

Characterization of Query Processors

Exploitation of replicated fragments

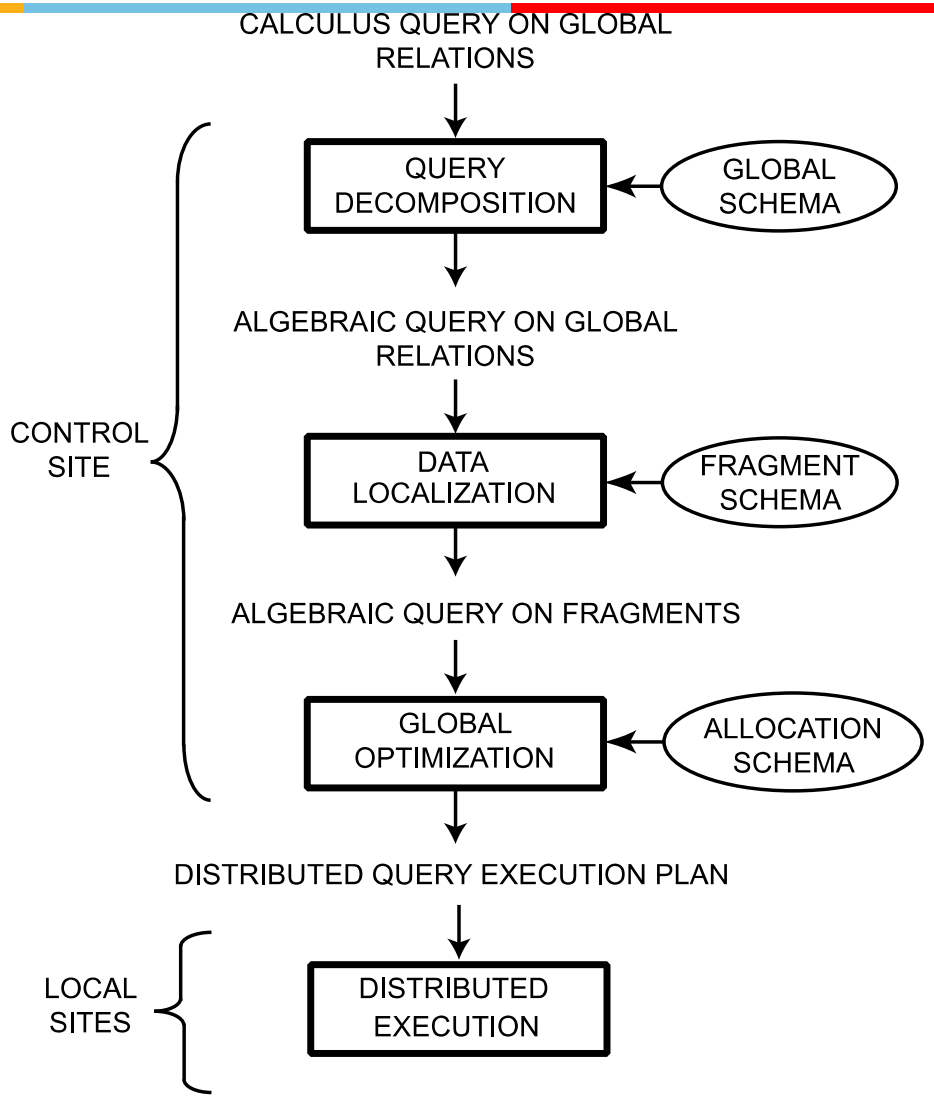
- Distributed queries expressed on global relations are mapped into queries on physical fragments of relations by translating relations into fragments. This process is known as **localization**.
- Most algorithms consider the localization process independent of the optimization.
- Some algorithms exploit existence of replicated fragments at run time in order to minimize communication cost.
- In such cases optimization algorithm becomes more complex.

Characterization of Query Processors

Use of semijoins

- The semijoin operation has an important property of reducing the size of the operand relation.
- When the main cost is due to communication, the semijoin is particularly useful as it reduces the size of data exchange between sites.

Generic Layering Scheme for Distributed Query Processing



Layering scheme for distributed query processing

- The input is a query on global data expressed in relational calculus. This query is posed on global (distributed) relations, meaning that data distribution is hidden.
- The first three layers map the input query into an optimized distributed query execution plan. They perform the functions of **query decomposition, data localization, and global query optimization**. Query decomposition and data localization correspond to query rewriting.
- The first three layers are performed by a central control site and use schema information stored in the global directory.
- The fourth layer performs **distributed query execution** by executing the plan and returns the answer to the query. It is done by the local sites.

Query decomposition

- The relational calculus query on global schema is decomposed into algebraic query. 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
  - The calculus query is rewritten in **normalized** form.
  - **Analysed** for correctness (detect errors in queries and reject them as early as possible)
  - The correct query is **simplified** by removing redundant predicates
  - Finally the query is **restructured** as an algebraic query

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 fragment query.
- 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.

Global query optimization

- The global query optimization is to find an **execution strategy for the** query which is close to optimal. An execution strategy for a distributed query can be described with relational algebra operations and communication primitives (send/receive operations).
- The query optimization consists of finding the **best ordering of operations** in the fragment query, including communication operation which minimize a cost function.
    - It is necessary to predict execution costs of alternative candidate orderings based on **fragment statistics** and the formulas for **estimating the cardinalities of results of relational operators**.
    - Optimization decisions depend on the **allocation of fragments** and available statistics on fragments which are recorded in the allocation schema.
    - Important aspect of query optimization is **join ordering**, since permutations of the joins within the query may lead to improvements of orders of magnitude.
- The output of the query optimization layer is an **optimized algebraic query with communication operators included on fragments**. It is typically represented and saved (for future executions) as a **distributed query execution plan** .

Local query optimization

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

Lecture Summary

- Overview of Query Processing (Chapter 7)

Thanks…

innovate achieve lead

- Next Lecture
  - Query Decomposition
  - Data Localization

- Questions??