**Query Processing in Distributed Databases**

Query Processing Strategies:

1. Transfer data from S2 to S1 and process the query.
2. Transfer data from S1 to S2 and process the query.
3. Transfer data from S1 and S2 to S3 and process the query.

Choice depends on factors like relation sizes, comm costs, and location where result is needed. Cost of transferring data over network is usually high so optimization is necessary.

*Data transfer cost = C \* Size*

C: Cost per byte of data transfer Size: Number of bytes transmitted

Example relations: *Employee at Site 1* and *Department at Site 2*

Employee at site 1: 10,000 rows. Row size = 100 bytes. Table size = 106 bytes.



Department at Site 2. 100 rows. Row size = 35 bytes. Table size = 3,500 bytes.



The result of this query will have 10,000 tuples, assuming that every employee is related to a department. Suppose each result tuple is 40 bytes long. The query is submitted at site 3 and the result is sent to this site. Employee & Department relations are absent at site 3.

→Employee at Site 1: 10,000 rows, each row is 100 bytes, total size is 1,000,000 bytes.

Department at Site 2: 100 rows, each row is 35 bytes, total size is 3,500 bytes.

Result tuple size: 40 bytes.

Q: For each employee, retrieve employee name & department name where he works.

→*Strategy 1:* Transfer both Employee and Department relations to Site 3.

Total transfer bytes = 10,000 rows \* 100 bytes/row + 100 \* 35 = 1,003,500 bytes.

*Strategy 2:* Transfer Employee to Site 2, perform join, and send result to Site 3.

Query result size = 40 \* 10,000 = 400,000 bytes.   
 Total transfer size = 400,000 + 1,000,000 = 1,400,000 bytes.

*Strategy 3:* Transfer Department to Site 1, perform join, and send result to Site 3.

Total bytes transferred = 400,000 (result) + 3,500 (Dept) = 403,500 bytes.

*Preferred strategy: Strategy 3* (minimizes data transfer).

Q’: Π Fname, Lname, Dname (Employee Dno = Dnumber Department)

For each department, retrieve the department name and the name of the department manager. Relational Algebra expression:

ΠFname,Lname,Dname (Employee Mgrssn = SSN Department)

→ The result of this query will have 100 tuples, assuming that every department has a manager, the execution strategies are:

*Strategy 1:* Transfer Employee and Dept to the result site & perform the join at site 3.

Total bytes transferred = 1,000,000 + 3500 = 1,003,500 bytes.

*Strategy 2:* Transfer Employee to site 2, execute join at site 2 & send result to site 3.

Query result size = 40 \* 100 = 4000 bytes.

Total transfer size = 4000 + 1,000,000 = 1,004,000 bytes.

*Strategy 3:* Transfer Dept reln to site 1, execute join at site 1 & send result to site 3.

Total transfer size = 4000 + 3500 = 7500 bytes.

*Preferred strategy:*  Choose strategy 3.

Q?: Executing queries Q and Q' when the result site is Site 2.  
→ *Strategy 1:* Transfer Employee Relation to Site 2

Employee relation is transferred to Site 2, where the query is executed.

The total transfer size for both queries Q and Q' is 1,000,000 bytes. This is because the Employee relation, which is common to both queries, is transferred to Site 2.

*Strategy 2:* Transfer Department Relation to Site 1

In this strategy, the Dept relation is transferred to Site 1, where join operation is executed.

Q: total transfer size = 400,000 (query result) + 3,500 (Dept relation) = 403,500 bytes.

Q': total transfer size = 4,000 (query result) + 3,500 (Dept relation) = 7,500 bytes.

**Semijoin** is used to reduce the amount of data transferred between distributed sites by performing a preliminary join operation at one site and transferring only the necessary attributes to another site for further processing. This helps optimize query processing in distributed database systems and minimizes network overhead.

In the example execution, the semijoin technique is applied to reduce the amount of data transferred between distributed sites.

1. Projection of Join Attributes at Site 2 and transferred to Site 1.

Q: 4 \* 100 = 400 bytes are transferred

Q': 9 \* 100 = 900 bytes are transferred.

1. The transferred attributes from the Dept reln are joined with Employee reln at Site 1.

Q: 34 \* 10,000 = 340,000 bytes transferred

Q': 39 \* 100 = 3900 bytes transferred.

1. The joined result is presented to the user at Site 2.

**Query optimization:** Enhancing overall performance & efficiency of distributed database systems. DDBMS deals with db distributed across multiple nodes or sites, & optimizing queries in such a distributed environment involves addressing unique challenges.

1. *Distribution Strategy:*

- DBs are distributed across multiple sites to improve data availability and performance.

- Consider the distribution strategy, which can be horizontal, vertical, or a combo of both.

1. *Global Query Optimization:*

- Unlike centralized dbs, DDBMS must optimize queries globally, considering multiple sites.

- Global query optimization aims to minimize the total cost of executing a query across all distributed sites, including data transfer and communication costs.

1. *Cost Model:*

- DDBMS uses a cost model to estimate execution cost of a query on each distributed site.

- It includes: communication costs, processing costs, & data transfer costs b/wn sites.

1. *Query Decomposition:*

- The query may be decomposed into subqueries that can be executed on individual sites.

- Decomposition strategies aim to minimize data transfer and optimize local processing.

1. *Data Localization:*

- Minimizing data movement across sites is a key consideration.

- It involves moving computation to the data rather than bringing data to the computation.

1. *Concurrency Control:*

- Query optimization must consider concurrency control mechanisms in a distributed environment to ensure consistency and isolation of transactions.

1. *Adaptability to Network Conditions:*

- Query optimization in DDBMS should be adaptable to varying network conditions, considering factors like latency and bandwidth.

**Example:-**

Site1: EMPLOYEE (1000 records, record size: 60 bytes)

Site2: DEPARTMENT (50 records, record size: 30 bytes)

Find employee names and department names; executed at Site 3.

→ *Strategies and Costs:*

1. Transfer both tables to Site 3 for joining: 1000 \* 60 + 50 \* 30 = 61,500 bytes.
2. Transfer EMPLOYEE to Site 2, join, and then transfer to Site 3: 60 \* 1000 + 60 \* 1000 = 120,000 bytes.
3. Transfer DEPARTMENT to Site 1, join, and then transfer to Site 3: 30 \* 50 + 60 \* 1000 = 61,500 bytes.

*→ Optimization Criteria:* Strategies 1 and 3 minimize data transfer.

→ Semi Join:-

1. Project attributes of Employee at Site 1 & transfer to S3: 30 \* 1000 = 30,000 bytes.
2. Transfer Dept to Site 3 and join with EMPLOYEE: 30 \* 50 = 1,500 bytes.

Total data transferred: 30,000 + 1,500 = 31,500 bytes.