



NAME : ARKAPRATIM GHOSH

ROLL No. : 13000121058

REG. No. : 211300100110045

TOPIC: SDD-1 Algorithm



Paper Name: Distributed Systems

Paper Code: PEC-IT601B

CSE, 6th Sem (2021-2025), CA-1

CONTENT

- Introduction
 - ◆ Distributed Query Optimisation
 - ◆ Semijoin Based Approach
- Hill Climbing Algorithm
- SDD-1 Algorithm
- Conclusion
- References

Introduction

Distributed Query Optimization:

- **Definition:** Distributed query optimization involves enhancing the performance of queries in a distributed database system by minimizing the overall execution time and resource usage.
- **Challenges:** Distributed environments pose challenges like network latency, data distribution, and heterogeneous hardware, making optimization crucial for efficient query processing.
- **Goals:** The primary goals include reducing data transfer across the network, minimizing query execution time, and optimizing resource utilization across distributed nodes.
- **Techniques:** Various techniques are employed, such as parallel query processing, data partitioning, and optimization algorithms to enhance the overall efficiency of distributed queries.

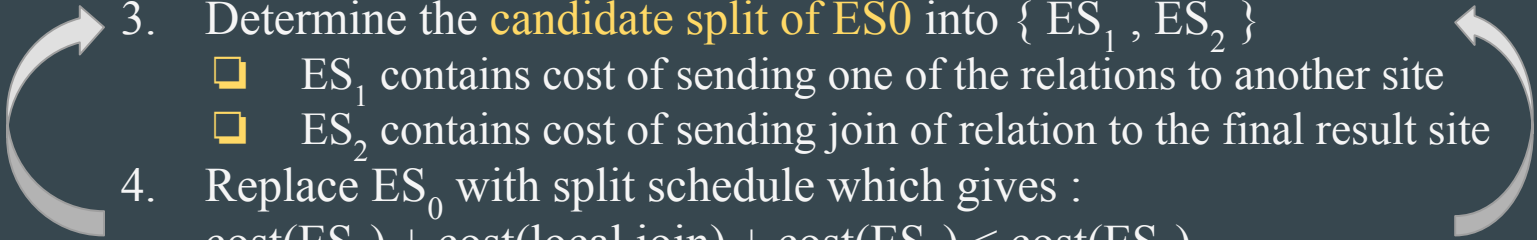
Introduction (Continued)

Semijoin Based Approach:

- **Semijoin Definition:** Semijoin is an operation that computes the intersection of two relations, retaining only the common tuples, which helps reduce data transfer in distributed query processing.
- **Advantages:** Semijoin-based approaches are efficient in distributed systems as they help minimize the amount of data exchanged between nodes, reducing network overhead.
- **Process:** In a semijoin-based approach, only the necessary information to satisfy the query conditions is sent between nodes, optimizing resource usage and query performance.
- **Applications:** Commonly used in distributed databases, semijoin operations are employed to enhance the efficiency of query processing in scenarios where minimizing data transfer is critical.
- **Example:** If two nodes need to perform a join operation, instead of sending the entire relations to each other, they exchange only the semijoin results, reducing the amount of data transmitted over the network.

Hill Climbing Algorithm

The hill-climbing algorithm is in the class of greedy algorithms, which start with an initial feasible solution and iteratively improve it.

1. Do initial Processing
 2. Select the **initial feasible solution** (ES_0)
 - ❑ Determine the candidate result site
 - ❑ Compute the cost of transferring all the referenced sites to each candidate site
 - ❑ ES_0 = Candidate site with the minimum cost
 3. Determine the **candidate split of ES_0** into $\{ ES_1, ES_2 \}$
 - ❑ ES_1 contains cost of sending one of the relations to another site
 - ❑ ES_2 contains cost of sending join of relation to the final result site
 4. Replace ES_0 with split schedule which gives :
$$\text{cost}(ES_1) + \text{cost}(\text{local join}) + \text{cost}(ES_2) < \text{cost}(ES_0)$$
- 

Applying Steps 3-4 recursively on ES_1 and ES_2 until no more plans exist and then removing the redundant transmissions

SDD-1 Algorithm

- It improves the Hill Climbing Algorithm by using semi joins.
- Objective function is expressed in terms of total communication time.
- Database statistics is used
- The main step of SDD-1 ALgorithm consists of determining and ordering beneficial semi joins i.e. **semijoin whose cost is less than their benefit.**
 - ❑ $\text{Cost}(R \text{ SJ } S) = T_{\text{MSG}} + T_{\text{TR}} * \text{size}(\Pi_A(S))$
 - ❑ $\text{Benefit}(R \text{ SJ } S) = (1 - \text{SF}_{\text{SJ}}(S.A)) * \text{size}(R) * T_{\text{TR}}$

*The semijoin-based algorithm proceeds in four phases: **initialization, selection of beneficial semi joins, assembly site selection, and post optimization.** The output of the algorithm is a global strategy for executing the query*

SDD-1 Algorithm (Continued)

Initialization

1. In the execution strategy include all the local processing.
2. Reflect the effects of local processing on the database profile
3. Construct a set of beneficial semi join operations (BS) as follows :

$$BS = \Phi$$

For each semi join SJ_i

$$BS \leftarrow BS \cup SJ_i \text{ if } \text{cost}(SJ_i) < \text{Benefit}(SJ_i)$$

Let us consider the following query:

```
SELECT R3.C
FROM R1,R2,R3
WHERE R1.A = R2.A
AND R2.B = R3.B
```



SDD-1 Algorithm (Continued)

relation	card	tuple size	relation size
R_1	30	50	1500
R_2	100	30	3000
R_3	50	40	2000

attribute	SF_{SJ}	$size(\Pi_{attribute})$
$R_1.A$	0.3	36
$R_2.A$	0.8	320
$R_2.B$	1.0	400
$R_3.B$	0.4	80

Beneficial semi joins

SJ1: $R_2 \bowtie_{SJ} R_1$, whose benefit is $2100 = (1-0.3) * 3000$ and cost is 36

SJ2: $R_2 \bowtie_{SJ} R_3$, whose benefit is $1800 = (1-0.4) * 3000$ and cost is 80

Non-beneficial semi joins

SJ3: $R_1 \bowtie_{SJ} R_2$, whose benefit is $300 = (1-0.8) * 1500$ and cost is 320

SJ4: $R_3 \bowtie_{SJ} R_2$, whose benefit is 0 and cost is 400.

SDD-1 Algorithm (Continued)

First Iteration:

- $SJ_1 (R_2)$ appended to ES.
- Changes in statistics: Size of R_2 becomes 900 ($3000 * 0.3$).
- Semijoin selectivity factor for R_2 .A reduced to 0.24.
- Size of ΠR_2 .A reduced to 96 ($320 * 0.3$).

Second Iteration:

- Beneficial semi joins: $SJ_2 (R_2 \text{ SJ } R_3)$ and $SJ_3 (R_1 \text{ SJ } R_2)$.
- SJ_3 selected due to higher benefit (1140) and lower cost (96).
- Effects on statistics: Size of R_1 becomes 360 ($1500 * 0.24$).

Third Iteration:

- Remaining beneficial semijoin $SJ_2 (R_2 \text{ SJ } R_3)$ appended to ES.
- Reduces the size of R_2 to 360 ($900 * 0.4$).

SDD-1 Algorithm (Continued)

After Reduction:

- Data stored: 360 at site 1, 360 at site 2, and 2000 at site 3.
- Site 3 chosen as the assembly site for final result computation.
- Post-optimization retains all semi joins as they remain beneficial.

Selected Strategy:

- Send $(R_2 \text{ SJ } R_1) \text{ SJ } R_3$ and $R_1 \text{ SJ } R_2$ to site 3 for final result computation.

Conclusion

Like its predecessor hill-climbing algorithm, the semijoin-based algorithm selects locally optimal strategies. Therefore, it ignores the higher-cost semi joins which would result in increasing the benefits and decreasing the costs of other semi joins. Thus this algorithm may not be able to select the global minimum cost solution.

References

Principles of Distributed Database Systems by M. Tamer Özsu



THANK YOU