Introduction to Distributed Databases

I. Basic Concepts and Notions

II. Distributed DB Design

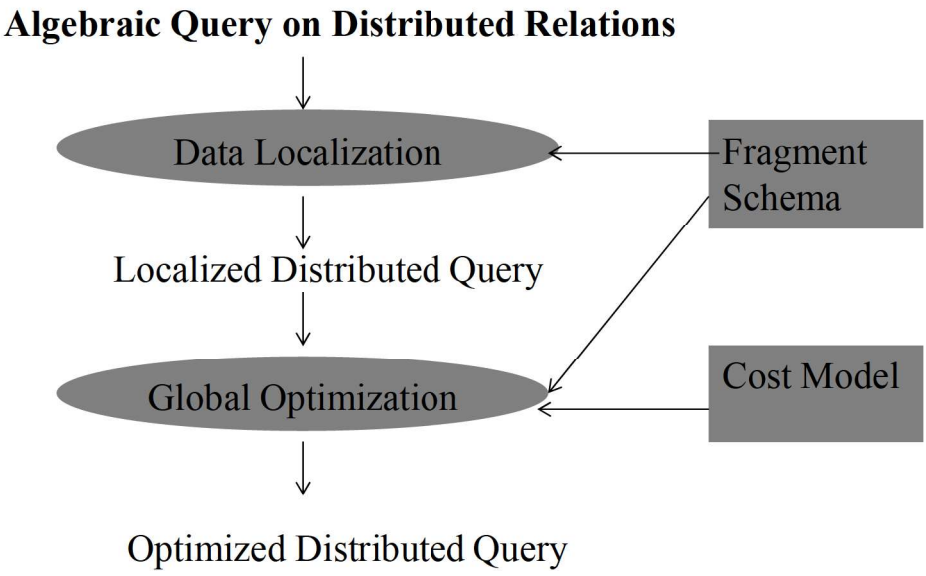
III. ***Distributed Query Processing***

PSU - Master - Hameurlain@irit.fr

III. Distributed Query Processing

- III.1 Query Processing Steps in Centralized Systems
  - Decomposition (Syntax, Semantic & Control)
  - Optimization
  - Execution
- III.2 Query Processing Steps in Distributed Systems
  - Decomposition (Syntax, Semantic & Control)
  - Localization
  - Optimization: Global & Local
  - Execution
- III.3 Sources of the Problem Complexity
  - Fragmentation and Replication Possibilities
  - Communication and Parallelism
  - Choice of Execution Sites: Availability, Load Balancing

Overview of Distributed Query Processing



III.4 Localization

- Definition
  - Transformation of an algebraic query on global relations into an equivalent distributed query expressed on fragments, using the fragment schema.
- 2 Steps
  - Generation of an equivalent canonical query
    - Substitute each relation by its reconstruction program
  - Simplification/Reduction
    a) Restructuring the algebraic tree to push the unary operators down to the fragments
    b) Application of the simplification rules to eliminate unnecessary operations

Rewrite and algebraic fragments

- Wines1 = $\pi_{\text{wineId, vineyard, area, vintage}}$(Wines)
- Wines2 = $\pi_{\text{wineId, degree, price}}$ (Wines)
- Drink1 = $\sigma_{\text{place}=\text{Paris}}$(Drink)
- Drink2 = $\sigma_{\text{place}\neq\text{Paris}}$(Drink)
- Query: vineyard of the wines consumed in Paris on March 12, 2016?

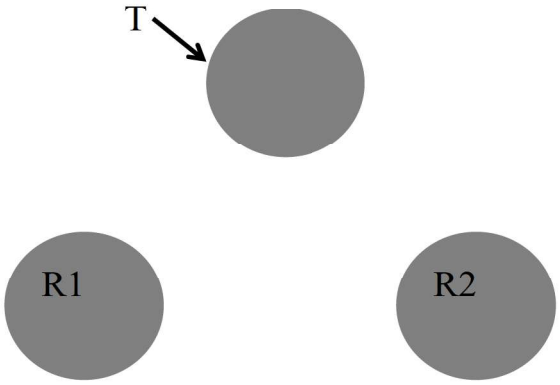
III.5 Distributed Join Algorithms

Hypothesis: Relations reside in different sites S1, S2, … Sn

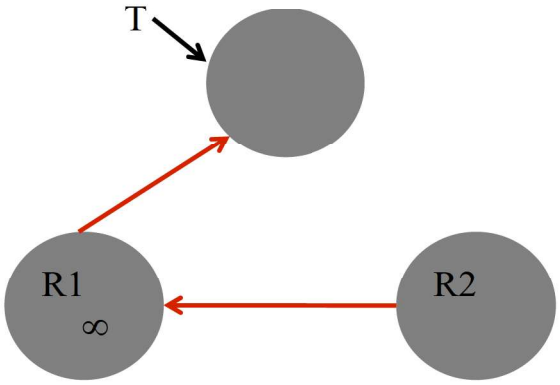
- Direct Join DJ: R1(S1) ∞ R2(S2)
  - Choice of the execution site for DJ
- Semi-join-based join SJ: Π. SJ. DJ
  - Definition: a Semi-Join (R, S, Q): {tuples of R participating in the join of R and S according to the qualification Q}
  - Objective: the semi-join consists in reducing the size of the operand relation in order to reduce the communication costs

    ➔ Minimize the Communications

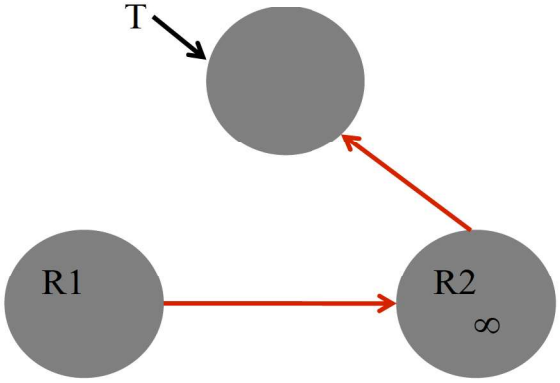
Direct join



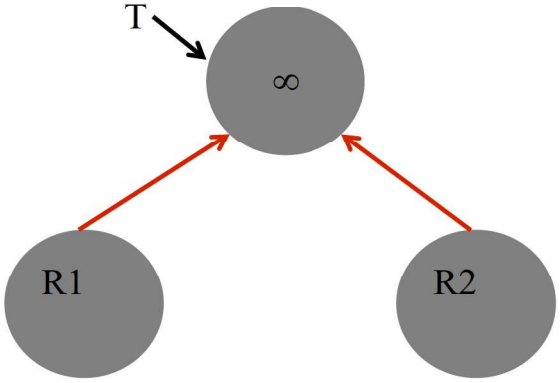
Direct join



Direct join



Direct join



Ex: hash join

- Algorithm

```
if (not (local (R1))) then receive (R1);
Build (R1, HT);
if (not (local (R2))) then receive (R2);
Probe(HT, R2, T);
if (not (local (T)) then receive (T);
    else (write(T));
```

Semi-join-based join

- R1 $\propto$ R2 = tuples in R1 that meet the join criteria
- Propose a distributed join algorithm using the semi-join operator
  - Join (R1, R2, R1.Ai=R2.Aj);

III.6 Optimization of Distributed Queries

- Simple Strategy (without optimization)
  - Execute first the unary operators on the fragments as local queries
  - Transfer the results to a unique site to assemble the results
- Strategy with Optimization (minimize a cost function) : 2 steps
  - S1: Global Optimization
  - S2: Local Optimization (optimization on each site)

S1: Global Optimization

- S11: Choice of the best sites = best localization of the fragment copies:
  - Selection of the fragment copies to access
  - Selection of the execution sites

- S12: Scheduling of the local queries = ordering the sites in order to minimize a cost function.

S2: Local optimization

- S21: Physical optimization
  - Choice of the join algorithm/select the most proper one
  - Scheduling of the joins: Enumerative/Randomize Strategies

- S22: Parallelization (with a parallel server): intra-operation and inter-operation (pipeline and independent) parallelism.