CSE 544
Principles of Database Management Systems

Magdalena Balazinska

Fall 2007

Lecture 12 - Distribution:

query optimization

References

- **R* Optimizer Validation and Performance Evaluation for Distributed Queries.**

  L. F. Mackert and G. M. Lohman. VLDB'86.

- **Database management systems.**

  Ramakrishnan and Gehrke.

  Third Ed. **Chapter 22**

Where We Are

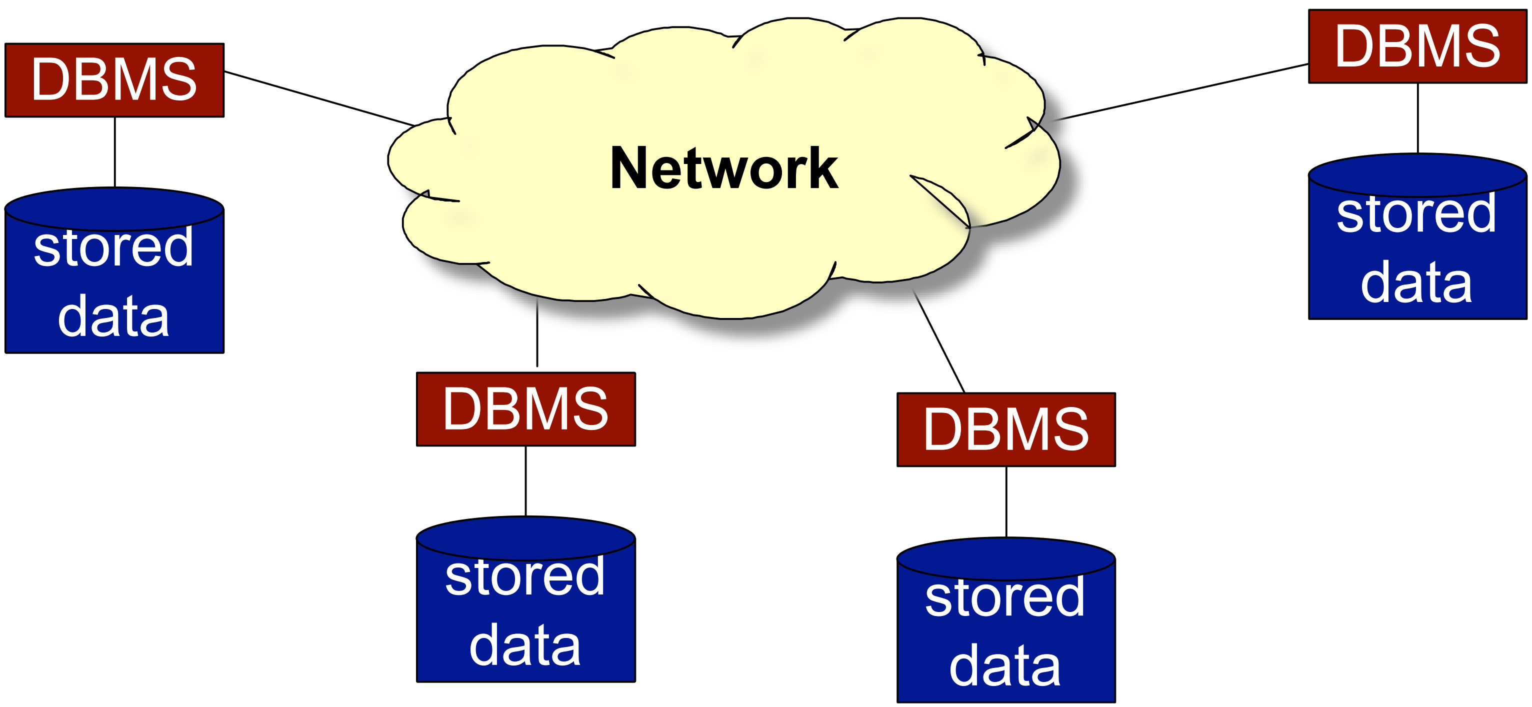
- We covered the fundamental topics
  - Relational model & schema normalization
  - DBMS Architecture
  - Storage and indexing
  - Query execution
  - Query optimization
  - Transactions
- Starting advanced topics: **distribution**

Outline

- Distributed DBMS motivation
- Distributed query optimization
- Distributed DBMS limitations and challenges

Distributed DBMS

- Important: many forms and definitions
- Our definition: shared nothing infrastructure
  - Multiple machines connected with a network



Reasons for a Distributed DBMS

- Scalability (ex: Amazon, eBay, Google)
  - Many small servers cheaper than large mainframe
  - Need to scale incrementally

- Inherent distribution
  - Large organizations have data at multiple locations (different offices) -> original motivation
  - Different types of data in different DBMSs
  - Web-based and Internet-based applications

Goals of a Distributed DBMS

- Shield users from distribution details
- Distribution transparency
  - Naming transparency
  - Location transparency
  - Fragmentation transparency
  - Performance transparency
    - Distributed query optimizer ensures similar performance no matter where query is submitted
  - Schema change and transaction transparency
- Replication transparency (next week)
- and more...

Distributed DBMS Features

- 70's and 80's, three main prototypes:
  - SDD-1, distributed INGRES, and R*
- Main components of a distributed DBMS
  - Defining data placement and fragmentation
  - Distributed catalog
  - Distributed query optimization (today)
  - Distributed transactions (next lecture)
  - Managing replicas (next week)

Outline

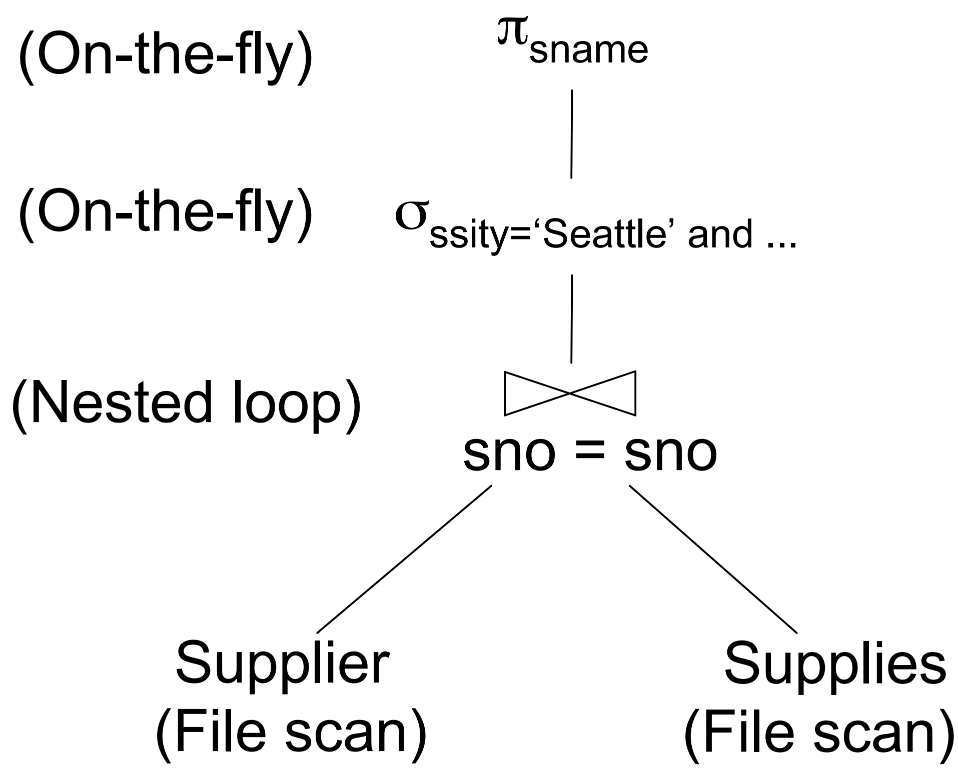
- Distributed DBMS motivation
- Distributed query optimization
- Distributed DBMS limitations and challenges

Review: Query Evaluation

Steps of query evaluation

- Query Parser
- Query Rewrite
- Optimizer
- Executor

Query plan



Review: Query Optimization

- Enumerate alternative plans
  - Many possible equivalent trees: e.g., join order
  - Many implementations for each operator
- Compute estimated cost of each plan
  - Compute number of I/Os and CPU utilization
  - Based on statistics
- Choose plan with lowest cost

Distributed Query Optimization

- Search space is larger
  - Must select sites for joining relations
  - Must select method for shipping inner relation: whole or matches
- Minimize resource utilization
  - I/O, CPU, & ***communication costs***
  - Example cost function used in R*
  - $W_{CPU} Nb_{inst} + W_{I/O} Nb_{I/O} + W_{msg} Nb_{msg} + W_{byte} Nb_{bytes}$
- Could also try to minimize response time
  - Least cost plan != Fastest plan

Inner Table Transfer Strategy

- Ship whole
  - Read inner relation at its home site (either using an index or not)
  - Project inner relation to remove attributes that are not needed
  - Apply any single-table predicates
  - Ship results to site of outer relation and store in temporary file
    - Note: we lose any indexes on the inner relation
- Fetch matches
  - For each tuple of outer relation, project tuple on join column
  - Send value to site of inner relation
  - Find matching tuples from inner relation
  - Ship projected, matching tuples back

Why is fetch matches so inefficient?

Distributed vs Local Joins

Why can distributed joins be faster than local ones?

- More resources are available to the join
  - Ex: Distributed query can use twice the buffer pool (useful when accessing relations through unclustered indexes)
- Different parts of the join can proceed in parallel
  - Ex: Join tuples from page 1 while shipping page 2
  - Ex: Can sort the two relations in parallel

Additional Join Strategies

- Dynamically-created temporary index on inner
  - Ship whole inner relation, store in temp table, index temp table
- Semijoin
  - Project outer relation on join column (eliminate duplicates)
  - Ship projected column to site with inner relation
  - Compute a natural join and ship matching tuples back
  - Join the two relations
- Bloomjoin
  - Same idea as semijoin, but use Bloom filter instead of sending all values in the join column
  - Bloom filter creates some false positives through collisions

Outline

- Distributed DBMS motivation
- Distributed query optimization
- Distributed DBMS limitations and challenges

Distributed DBMS Limitations

- Top-down
  - Global, a priori data placement
  - Global query optimization, one query at a time; no notion of load balance
  - Distributed transactions, tight coupling
- Assumes full cooperation of all sites
- Assumes uniform sites
- Assumes short-duration operations
- Limited scalability

Distributed DBMS Challenges

- Autonomy: different administrative domains
  - Cannot always assume full cooperation
  - Do not want distributed transactions
- Heterogeneity
  - Different capabilities at different locations
  - Different data types, different semantics -> data integration pb
- Large-scale
  - Internet-scale query processor