Tore Risch

Dept. of Information Technology

Uppsala University

Sweden

(Tore.Risch@it.uu.se)

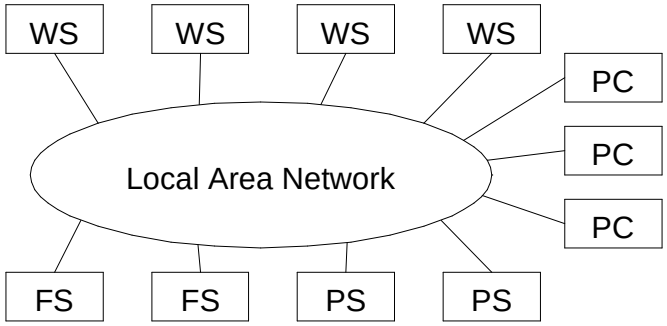
Background

What is a Distributed System?

A Distributed System is a number of autonomous computers communicating over a network with software for integrated tasks.

Examples of Distributed Systems:

- SUN's Network File System (NFS), distributed file system



Tore Risch

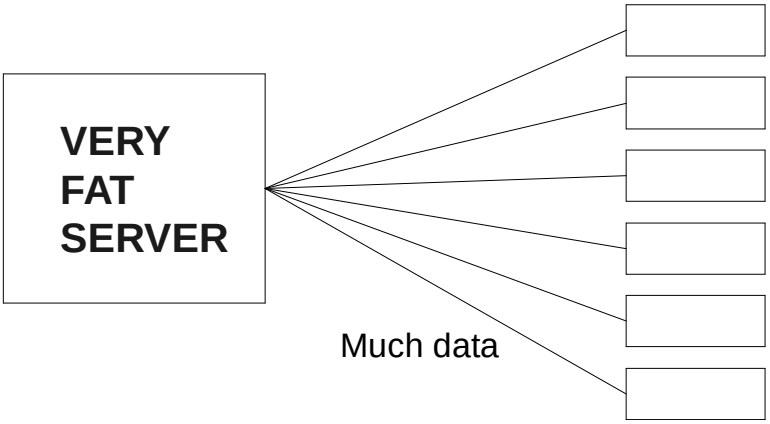
Dept. of Information Technology

Uppsala University

Sweden

(Tore.Risch@it.uu.se)

Centralized Database Server



Many clients

- Stream (row-by-row) based client-server interfaces
- DBMS specific interfaces
- Compiler integrated interfaces (embedded SQL)
- ODBC: SQL-based standardized subroutine call library (MicroSoft)
- JDBC: ODBC for Java (not MicroSoft)

Tore Risch

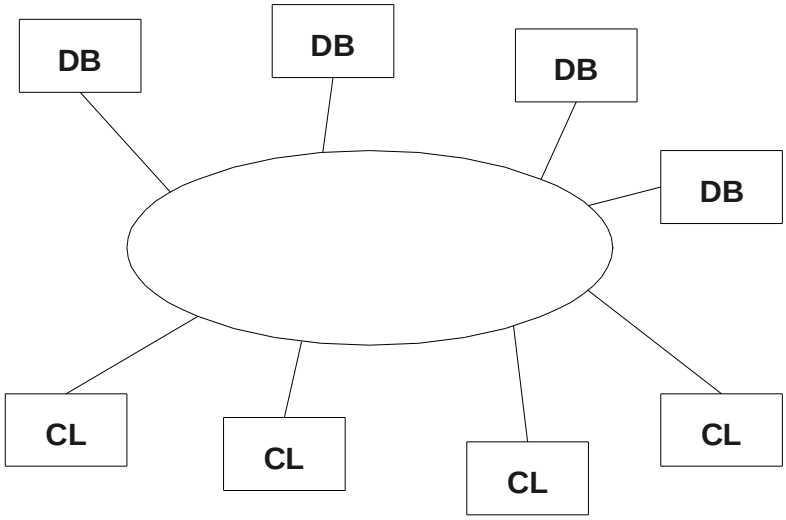
Dept. of Information Technology

Uppsala University

Sweden

(Tore.Risch@it.uu.se)

Distributed Databases



- Database seen as one unit; queries and updates to ONE database.
- Data in database *transparently* distributed over many DB nodes.
- Manual partitioning or *fragmentation* of data tables.
- DBMS automatically optimizes queries and updates to distributed database.

Tore Risch

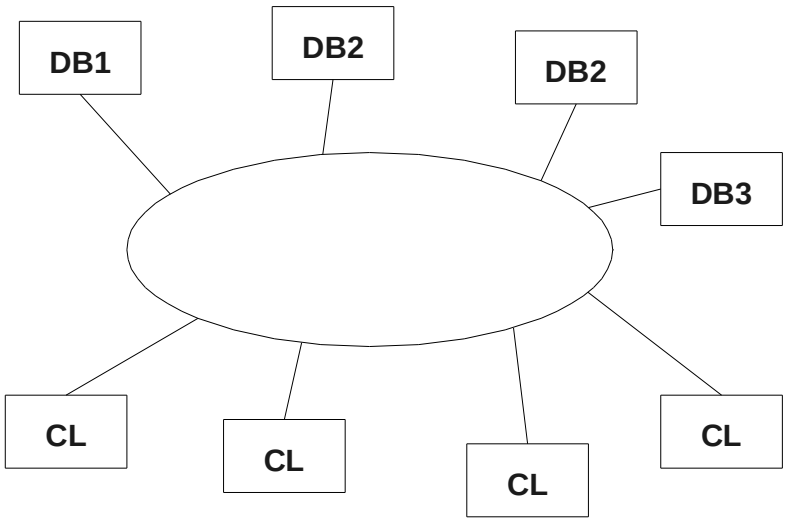
Dept. of Information Technology

Uppsala University

Sweden

(Tore.Risch@it.uu.se)

Multi Databases



- Database seen as several heterogeneous units
- Multi-database query language needed to combine data from the databases.
- Primitives needed to integrate (combine, fuse) data from the databases.
- Special query optimization techniques to deal with heterogneity and dynamism.

Tore Risch

Dept. of Information Technology

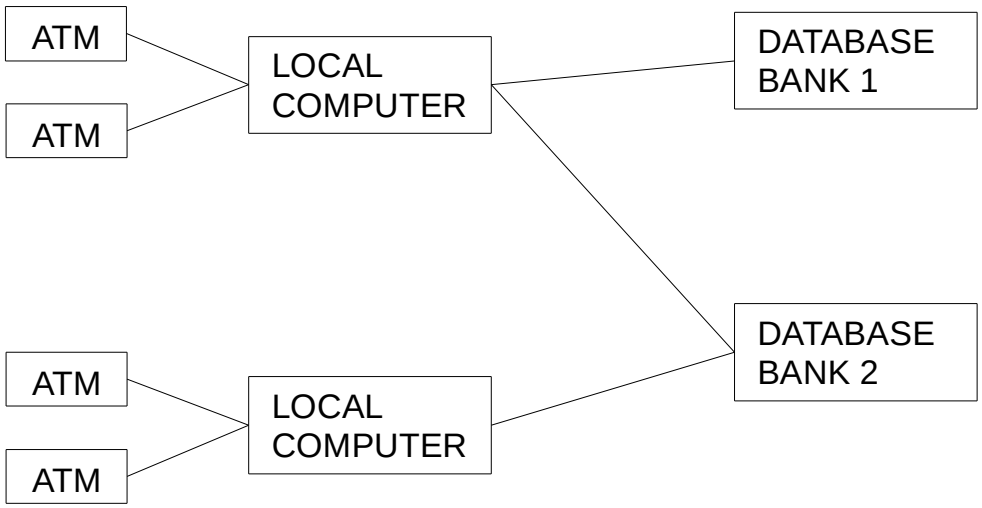
Uppsala University

Sweden

(Tore.Risch@it.uu.se)

Example of Multi-Database

Automatic Teller Machines, ATMs



Tore Risch

Dept. of Information Technology

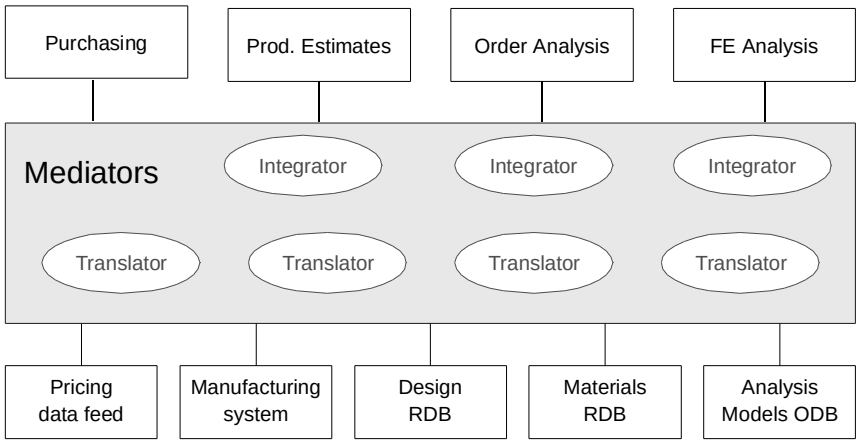
Uppsala University

Sweden

(Tore.Risch@it.uu.se)

Amos2 meditator system

- Purpose: Software to facilitate queries to databases and other data sources.



Tore Risch

Dept. of Information Technology

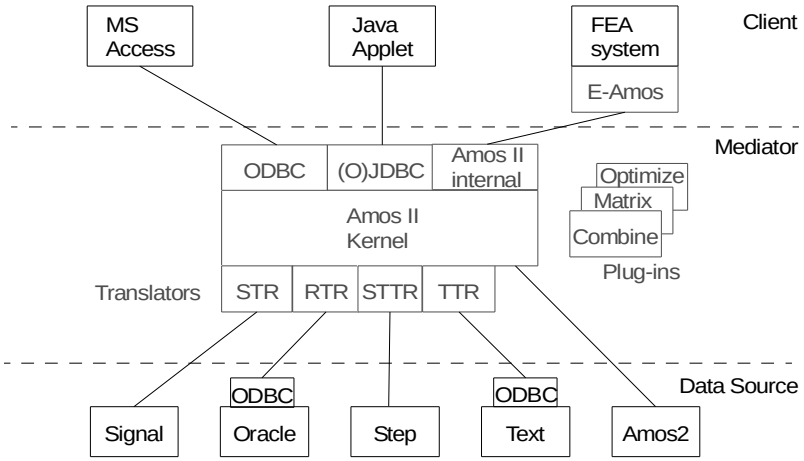
Uppsala University

Sweden

(Tore.Risch@it.uu.se)

Amos2 Architecture

- 3-level multi-database architecture.



Tore Risch

Dept. of Information Technology

Uppsala University

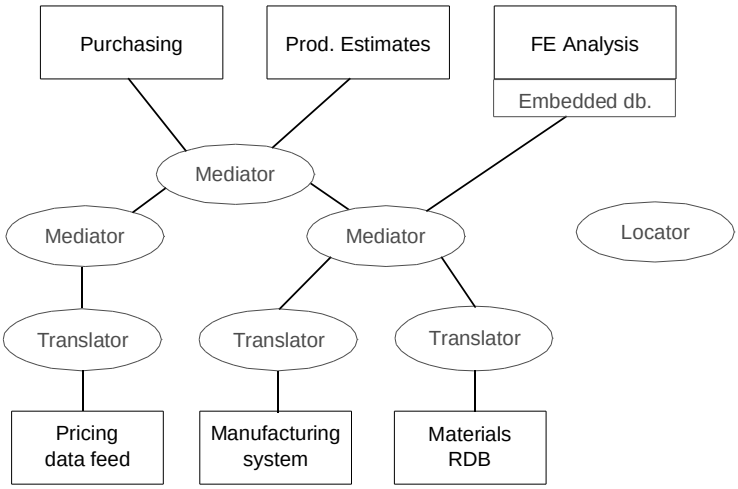
Sweden

(Tore.Risch@it.uu.se)

Amos II Distribution Architecture

- Amos II is a distributed system:
  Mediator data server may access other mediator data servers.

  => many layers of mediator servers possible.



=> Special optimizations to push computations close to sources when needed.

Tore Risch

Dept. of Information Technology

Uppsala University

Sweden

(Tore.Risch@it.uu.se)

What is a Distributed Database?

"A distributed database (DDB) is a collection of multiple, *logically interrelated* databases *distributed over a computer network*. A DDBMS is the software system that permits the management of DDBs and makes the distribution transparent to the user."

- A DDB is NOT:
  - A collection of files (need structure and DB manager)
  - A client-server interface to a database

    Data on one node, clients on other nodes in network

    (Almost) every centralized DBMS has client-server interface

Tore Risch

Dept. of Information Technology

Uppsala University

Sweden

(Tore.Risch@it.uu.se)

Example of DDB

- Multi-national company with distributed departments
- Distributed data management:
  - Each location keep local records of local employees
  - R & D keeps track of what is going on at its facility.
  - Manufacturing plants keep data related to their engineering operations and access to R & D
  - Manufacturing keeps track of local inventory. Access to warehouse also possible.
  - Warehouses keeps local inventory. Manufacturing can access inventory levels.
  - Headquarters keep marketing and sales records per region. Share with other headquarters and can access inventory data at plants and warehouses.

Tore Risch
Dept. of Information Technology
Uppsala University
Sweden
(Tore.Risch@it.uu.se)

Advantages with DDBs

- Local autonomy
  - Local control
  - Local policies
- Improved performance
  - Avoid data shipping
- Improved Reliability
  - Crashes less severe (if appl. no dep. on non-local data)
- Expandability
  - Easy to add new nodes (not always linear scale-up because of central directory)
- Sharability
  - Uniform interface and sharing through DDBMS

Tore Risch
Dept. of Information Technology
Uppsala University
Sweden
(Tore.Risch@it.uu.se)

Replication and fragmentation

- Data replication
  Same data on several nodes
  - For reliability and access performance
  - Not necessary to replicate all tables
    Fully replicated vs *partially replicated*
    Full replication often not realistic!
  - Updates must be propagated to each replica!
  - Special procedures after failures to restore consistency
  - More problematic transaction synchronization!
- Data Fragmentation (= data partitioning)
  Tables transparently split over several nodes
  - For access performance
  - Good when nodes far apart

Tore Risch
Dept. of Information Technology
Uppsala University
Sweden
(Tore.Risch@it.uu.se)

Problems with DDBs

- Complexity
  Database administration may be complex
  E.g. recovery after crashes complex
- Distribution of Control
  More meetings
- Security
- Networking a known problem

Tore Risch
Dept. of Information Technology
Uppsala University
Sweden
(Tore.Risch@it.uu.se)

Problems with DDBs

- Distributed schema management
  Schema is accessed whenever SQL query issued!
  - Global directory => Central Database becomes *hot spot*
  - Local directories => Data replication
  => Since schema is not updated often but need to be accessed very often it is normally *fully replicated* by the DDBMS.
- Distributed concurrency control
- Reliability of DDBMS
  - Consistency of replicas
  - Bring up (fragmented) database at failed sites
- OS Support
  - Multiple layers of network software

Transparency

- Replication Transparency
  - User unaware of data replicas
  - Automatic replica propagation at update
  - Special problems when nodes are down
- Fragmentation Transparency
  - E.g. a logical relation is horizontally fragmented into local physical tables
  - Translation from *global queries* to *fragmented queries*
- Network Transparency
  - Protect user from operational details of network
  - Hide existence of network
  - No machine names in database table references
    - Location transparency
    - Naming transparency

Distributed database design

- Where to put data and applications?
- Partitioned data:
  - Split data into distributed partitions
- Replicated data:
  - Several sites have data copies
- Goal: Minimize combined cost of storing data, communication, transactions.
- NP complete optimization problem.

- Distributed Query Processing
  - Automatically done by distributed query processor of DDBMS
  - Analyze query --> distributed execution plan
  - Factors:
    - Data replication
    - Data availability
    - Communication costs

Tore Risch
Dept. of Information Technology
Uppsala University
Sweden
(Tore.Risch@it.uu.se)

Parallel Databases

Parallel data servers

- Use parallel processing in cluster of computer nodes for data servers
- Automatic *fragmentation and replication* of data over the different nodes by the Parallel DBMS (PDBMS)

  To achieve maximal throughput and availablility
- Utilization of modern hardware
  - Multiple independent hardware components interconnected through fast communication medium (e.g. a bus)
  - Modern multi-processor architectures natural to support interquery, intraquery, and intraoperation parallelism
  - Connect disk per processor unit

Tore Risch
Dept. of Information Technology
Uppsala University
Sweden
(Tore.Risch@it.uu.se)

DDBMS Reliability

Two-Phase Commit Protocol (2PC)

- The most well known protocol to ensure atomic commitment of distributed transactions.
- Extend local site commit protocols by ‘contract’ (synchronization) that they all agree on committing (or aborting) in a distributed transaction.

Tore Risch

Dept. of Information Technology

Uppsala University

Sweden

(Tore.Risch@it.uu.se)