EPL446 – Advanced Database Systems

Lecture 19

Introduction to Distributed Databases

Chapter 25.1-25.4: Elmasri & Navathe, 5ED
Chapter 22.6-22-10: Ramakrishnan & Gehrke, 3ED

Demetris Zeinalipour
http://www.cs.ucy.ac.cy/~dzeina/courses/epl446
Lecture Outline
(Introduction to Distributed Databases)

• **Introduction** to Distributed Databases
• **Types** of Distributed Databases
  – Homogeneous, Heterogeneous (Federated, MultiDBs)
• Distributed Databases **Architectures**
  – Client Server, Collaboration Server, Middleware
• **Data Fragmentation & Replication**
  – Horizontal, Vertical and Mixed Fragmentation.
  – Synchronous vs. Asynchronous
• Distributed **Catalog Management** (next lecture)
• Distributed **Query Processing** (next lecture)
  – Centralized, Ship-to-one-site, Semi-join, Bloom-join & Bloom Filters
Introduction to Distributed Databases
(Εισαγωγή σε Κατανεμημένες Βάσεις)

• **Distributed Database (DDB)**
  – a collection of multiple **logically related** (λογικά συσχετιζόμενες) **databases** distributed over a **computer network**.

• **Distributed Database Management System (DDBMS)**
  – a **generic software system** that manages a **distributed database** while making the **distribution transparent** (διαφανής) to the user.

• Distributed Databases
  – Reality (e.g., WWW, Grids, Cloud, Sensors, Mobiles, …)

• Distributed Database Management Systems
  – A myth? … see next slide for details
Introduction to DDBMS
(Εισαγωγή σε Κατανεμημένα Συσ. Διαχ. Δεδομένων)

• The Problem
  – There is no real consensus (πλειοψηφία γνώμης) on what the design objectives of DDBMS should be.
  – The field is evolving (εξελίσσεται) mostly in response to user needs rather than generic principles.

• Some Facts
  – DDBMS come at a significant cost in terms of performance (επίδοση), software complexity (Πολυπλοκότητα λογισμικού) and administration difficulty (δυσκολία διαχείρισης).
  – A full scale comprehensive DDBMS that implements an open standard for distributed databases never emerged as a commercially viable product;
  • All DBMS vendors (e.g. SQL Server, Oracle, DB2) provide means to distribute data but there is a lack of an open architecture.
Introduction to Distributed Databases (Εισαγωγή σε Κατανεμημένες Βάσεις)

- A **Centralized database** with distributed clients

- A **Truly Distributed Database**

![Centralized Database Diagram](image1)

![Truly Distributed Database Diagram](image2)
The following properties are desirable:

- **Distributed Data Independence** (**Κατανεμημένη Ανεξαρτησία Δεδομένων**): Users should not have to know where data is located.
  - Extends **Physical** and **Logical Data Independence** principles.
  - In particular, the user is shielded from the details of how data is “stored” (e.g., sorted, not sorted) and “logically organized” (e.g., in one or more relations)

- **Distributed Transaction Atomicity** (**Κατανεμημένη Ατομικότητα Δοσοληψιών**): Users should be able to **write Transctions** accessing **multiple sites** just like **local Transactions**.
Types of Distributed Databases
(Τύποι Κατανεμημένων Βάσεων)

Homogeneous (Ομογενής): Every site runs same type of DBMS

- All sites of the database system have **identical setup**, i.e., same database system software.
- The underlying **operating system** might be **different**.
  - For example, **ALL** sites run **Oracle** or **DB2**, or **Sybase** or some other database system.
- The underlying operating systems **CAN be a mixture** of **Linux**, **Window**, **Unix**, etc.
Types of Distributed Databases
(Τύποι Κατανεμημένων Βάσεων)

**Heterogeneous (Ετερογενής):** Different sites run different DBMSs (even non-relational DBMSs).

- **Types of Heterogeneous Databases**
  - **Federated (Single Schema):** Each site may run different database system but the data access is managed through a single conceptual schema.
    - This implies that the degree of local autonomy is minimum.
    - Each site must adhere to a centralized access policy.
  - **Multidatabase (No Schema):** There is no one conceptual global schema. For data access a schema is constructed dynamically as needed by the application software.
A. Client-Server (Πελάτη-Εξυπηρετητή):

- Query can span one or more sites.
- All query processing at server.
  - Clients are “thin” (i.e., application logic implemented at the server)
  - Set-oriented communication (tuple-at-a-time communication is expensive).
  - Clients perform caching of results to minimize communication
- Drawback: Not Scalable (to combine data from multiple sources requires that the client implements all the application logic locally)
B. Collaborating-Server (Συνεργαζόμενου Εξυπηρετητή):

- Client ships query to a collaborating server which takes care of:
  - Optimizing the query and sending it to N sites
  - Collecting/Caching the results.
  - Returning the results to the user

- If the collaborating server is implemented as part of the client software stack then it is also called “Middleware”
Data Fragmentation
(Κατάτμηση Δεδομένων)

• **Horizontal Fragmentation (Οριζόντια Κατάτμηση)**
  - Create tuple subsets of relations and **assign each subset** to a distributed site (or relation might physically be fragmented).
  - **Fragments** are usually required to be **disjoint** ($S_1 \cap S_2 = \emptyset$)
  - **Union** of fragments must be **equal** to the initial relation.

• **Vertical Fragmentation (Κάθετη Κατάτμηση)**
  - create subset of **columns** of a relation and assign each subset to a distributed site (or relation might physically be fragmented).
  - **Collection** of fragments should be a **lossless-join decomposition** of the original relation (in other words we can recover the initial relation)

• **Mixed Fragmentation (Μεικτή Κατάτμηση)**
  - A **combination** of Vertical and Horizontal fragmentation.
  - This is achieved by **SELECT-PROJECT** operations which is represented by $\Pi_{L_i}(\sigma_{C_i}(R))$. 
Data Fragmentation
(Κατάτμηση Δεδομένων)

• **Horizontal fragmentation** (Οριζόντια Κατάτμηση)
  – It is a horizontal subset of a relation which contain those of tuples which satisfy selection conditions.
  – Consider the Employee relation with selection condition (DNO = 5).
    - All tuples satisfy this condition will create a subset which will be a horizontal fragment of Employee relation.
  – A selection condition may be composed of several conditions connected by AND or OR.
Data Fragmentation
(Κατάτμηση Δεδομένων)

• **Vertical fragmentation** (Κάθετη Κατάτμηση)
  – It is a **subset of a relation** which is created by a subset of columns.
    • A vertical fragment of a relation will contain **values of selected columns**.
    • There is no **selection** condition used in vertical fragmentation.
  – Consider the Employee relation.
    • A **vertical fragment** of can be created by keeping the values of Name, Bdate, Sex, and Address.
  – Because there is no condition for creating a vertical fragment, **each fragment must include the primary key** attribute of the parent relation Employee.
    • In this way all **vertical fragments** of a relation are connected.
Data Replication
(Αντίγραφα Δεδομένων)

• **Data Replication (Αντίγραφή Δεδομένων)**
  
  – Store *copies* of a data at **multiple sites** to minimize *access time* (χρόνος προσπέλασης) and increase *availability of data* (διαθεσιμότητα)

  – **Full Replication**: the entire database is replicated

  – **Partial Replication**: some selected part is replicated to some of the sites.

  – Data replication is achieved through a *replication schema*, a definition of:
    
    • all *attributes* and *tuples* in the *DB*; and

    • a *sequence of UNION operations* from which the initial database can be reconstructed.

  – Replication Strategies: **Synchronous (online) vs. Asynchronous (offline)**
Data Replication
(Αντίγραφα Δεδομένων)

- **Asynchronous Replication:** Copies of a modified relation are only *periodically updated*;
  - different copies may get out of synch in the meantime.
  - Current products follow this approach, e.g., Oracle Streams built-in feature which enables the propagation of data (*DML and DDL updates*), transactions and events in a data stream either within a database, or from one database to another.

- **Synchronous Replication:** *All copies* of a modified relation (fragment) must be updated before the modifying Xact commits.
  - Oracle Streams supports this feature but it is efficient only when changes affect a *small number of tables*.
The EMPLOYEE, PROJECT, and WORKS_ON tables may be fragmented horizontally and stored with possible replication as shown below.

Fragmentation and Replication (Κατάτμηση και Αντίγραφα)