

# *Distributed DBMS Architectures*

RAMNA SATTAR



# Distributed DBMS Architectures

## Distributed DBMS Architectures

DDBMS architectures are generally developed depending on three parameters :

### **Distribution :**

It states the physical distribution of data across the different sites.

### **Autonomy:**

It indicates the distribution of control of the database system and the degree to which each constituent DBMS can operate independently.

### **Heterogeneity :**

It refers to the uniformity or dissimilarity of the data models, system components and databases.



# Architectural Models

## Architectural Models:

Some of the common architectural models are :

- **Client - Server Architecture for DDBMS**
- **Peer - to - Peer Architecture for DDBMS**
- **Multi - DBMS Architecture**



# Architectural Models

## **Client - Server Architecture for DDBMS:**

This is a two-level architecture where the functionality is divided into servers and clients. The server functions primarily encompass data management, query processing, optimization and transaction management. Client functions include mainly user interface. However, they have some functions like consistency checking and transaction management.

The two different client - server architecture are:

- **Single Server Multiple Client**
- **Multiple Server Multiple Client**



# Architectural Models

## Peer- to-Peer Architecture for DDBMS:

In these systems, each peer acts both as a client and a server for imparting database services. The peers share their resource with other peers and co-ordinate their activities.

This architecture generally has four levels of schemas:

- **Global Conceptual Schema:** Depicts the global logical view of data.
- **Local Conceptual Schema:** Depicts logical data organization at each site.
- **Local Internal Schema:** Depicts physical data organization at each site.
- **External Schema:** Depicts user view of data.



# Architectural Models

## Multi - DBMS Architecture:

This is an integrated database system formed by a collection of two or more autonomous database systems.

Multi-DBMS can be expressed through six levels of schemas:

**Multi-database View Level:** Depicts multiple user views comprising of subsets of the integrated distributed database.

**Multi-database Conceptual Level:** Depicts integrated multi-database that comprises of global logical multi-database structure definitions.

**Multi-database Internal Level:** Depicts the data distribution across different sites and multi-database to local data mapping.

**Local database View Level:** Depicts public view of local data.

**Local database Conceptual Level:** Depicts local data organization at each site.

**Local database Internal Level:** Depicts physical data organization at each site.



# *Distribution Transparency*

RAMNA SATTAR



# Distribution Transparency

## Distribution Transparency:

Distribution transparency is the property of distributed databases by the virtue of which the internal details of the distribution are hidden from the users. The DDBMS designer may choose to fragment tables, replicate the fragments and store them at different sites. However, since users are oblivious of these details, they find the distributed database easy to use like any centralized database.

The three dimensions of distribution transparency are:

- **Location transparency**
- **Fragmentation transparency**
- **Replication transparency**





# Distribution Transparency

## Location Transparency

Location transparency ensures that the user can query on any table(s) or fragment(s) of a table as if they were stored locally in the user's site. The fact that the table or its fragments are stored at remote site in the distributed database system, should be completely oblivious to the end user. The address of the remote site(s) and the access mechanisms are completely hidden.

In order to incorporate location transparency, DDBMS should have access to updated and accurate data dictionary and DDBMS directory which contains the details of locations of data.



# Distribution Transparency

## Fragmentation Transparency

Fragmentation transparency enables users to query upon any table as if it were unfragmented. Thus, it hides the fact that the table the user is querying on is actually a fragment or union of some fragments. It also conceals the fact that the fragments are located at diverse sites.

This is somewhat similar to users of SQL views, where the user may not know that they are using a view of a table instead of the table itself.



# Distribution Transparency

## Replication Transparency

Replication transparency ensures that replication of databases are hidden from the users. It enables users to query upon a table as if only a single copy of the table exists.

Replication transparency is associated with concurrency transparency and failure transparency. Whenever a user updates a data item, the update is reflected in all the copies of the table. However, this operation should not be known to the user. This is concurrency transparency. Also, in case of failure of a site, the user can still proceed with his queries using replicated copies without any knowledge of failure. This is failure transparency.



# Distribution Transparency

## Combination of Transparencies

In any distributed database system, the designer should ensure that all the stated transparencies are maintained to a considerable extent. The designer may choose to fragment tables, replicate them and store them at different sites; all oblivious to the end user. However, complete distribution transparency is a tough task and requires considerable design efforts.



*THANK YOU*

ANY QUERY???

