DBMS Architecture
Chapter 6

- Client/Server Architecture
- Replication/Fragmentation
- Distributed Architecture
Client/Server Architecture
**Standalone Systems**

- The DBMS and the Application on a single micro processing Machine
- Serves only one user at a time
- Except the benefit of using the Database Principles it *lacks*:
  - Data sharing
  - Centralized data management
  - Multi user feature
  - Data administration is difficult
    - The user has to stop working on the machine and the admin will takeover
Client/Server DB

- **Centralized DB** – DB located at single site or location
- At least one server will store and process a database. This server is called DB server
- It is responsible for the DB processing and shared access functions

![Diagram showing client-server architecture with requests and responses.]
Client/Server

A distinction exists between computers that make available network resources (servers) and those computers that use the resources (clients or workstations).

Pros:
- Relatively secure
  - Centralized servers easy to manage
  - Physically centralized
  - Secure OS
- Better performance
- Centralized backups
- Reliability
  - Simple job to do plus built in redundancy

Cons:
- Require professional administration
- More hardware intensive
Client/Server Model

- Microcomputer users, or clients, share database services of a centralized computer called a server.
## Functions of client/server

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presents and manages the user interface</td>
<td>Accepts and processes database requests</td>
</tr>
<tr>
<td>Accepts and checks the syntax of user inputs</td>
<td>Checks for user authorization</td>
</tr>
<tr>
<td>Processes application</td>
<td>Ensures data integrity and security</td>
</tr>
<tr>
<td>Generate DB requests</td>
<td>Performs queries and updates</td>
</tr>
<tr>
<td>Receives results from server and formats results</td>
<td>Provides concurrent access control</td>
</tr>
</tbody>
</table>
Architecture

- Client/server is normally a 2-tier architecture
  1. Tier-1 : Client
  2. Tier-2 : DB server

- More complex client/server or web-based systems may require 3-tier architecture
  1. Tier 1 : Client
  2. Tier 2 : Application server
  3. Tier 3 : DB server
Distributed Database Systems
Distributed DB Concepts

- Distributed Computing System
  - Consists of a number of processing elements interconnected by a computer network that cooperate in processing certain tasks
- Distributed Database
  - Collection of logically interrelated databases over a computer network
- Distributed DBMS
  - Software system that manages a distributed DB
Distributed Database

- Database development facilitates the integration of data available in an organization and enforces security on data access.
- But it is not always the case that organizational data reside in one site.
- This demand databases at different sites to be integrated and synchronized with all the facilities of database approach.
- This leads to Distributed Database Systems
A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network.

In a distributed database system, the database is stored on several computers.

Data management is decentralized but act as if they are centralized.

A distributed database system consists of loosely coupled sites that share no physical component and database systems that run on each site are independent of each other.
Distributed Database Cont...

- Transactions may access data at one or more sites
- There is a need for a controller
- Computer Systems may be located at each local branch office.
Reasons for Distributed DB

- Business unit autonomy and distribution
- More Data sharing need
- Data communication reliability and costs
- Multiple application vendors with various system architectures
- Database recovery need (Security and reliability)
- Complex transaction and analytic processing
The functionalities of a DDBMS will include:
- Extended Communication Services,
- Distributed Query Processing,
- Extended Concurrency Control and
- Extended Recovery Services.
Advantage of Distributed DB

- Management of distributed data with different levels of transparency
  - **Replication transparency** - store copies at multiple sites.
    - User is unaware of the existence of multiple copies
  - **Fragmentation transparency** - stores fragment of data at different sites
    - **Horizontal fragmentation**
    - **Vertical fragmentation**
Advantage of Distributed DB (contd...)

- **Increased Reliability and Availability**
  - When the data and the DBMS software are distributed over several sites, one site may fail, but other sites continue to operate. Only the data and the software that exist at the failed site cannot be accessed. This improves both reliability and availability.

- **Improved Performance**
  - **Data Localization:** A Distributed database management system fragments the database by keeping the data closer to where it is needed.

- **Easier Expansion**
  - In a Distributed environment, expansion of the system in terms of adding more data, increasing the database sizes or adding more processors is much more easier.
Additional Functions of DDBs (contd…)

- Keeping track of data
- Distributed query processing
  - Ability to access remote sites and transmit queries
- Distributed transaction management
  - devise execution strategies for queries and transactions
  - Synchronize access and maintain integrity of data
- Replicated data management
  - decide and maintain replicated
- Distributed database recovery
  - Ability to recover from individual site crashes and failure of communication links
Additional Functions of DDBs (contd…)

- Security
  - Proper management of security of the data
  - Proper authorization/access privileges of users
- Distributed directory (catalog) management
  - Directory contains information about data in the database
  - Directory may be global for the entire DDB or local for each site
Disadvantages of DDBMS...

- Software development cost
- Greater potential for bugs (parallel processing may endanger correctness)
- Increased processing overhead (due to communication jargons)
- Communication problems
• Concepts in DDBMS
  • **Replication:** System maintains multiple copies of data, stored in different sites, for faster retrieval and fault tolerance.
  • **Fragmentation:** Relation is partitioned into several fragments stored in distinct sites
  • **Data transparency:** Degree to which system user may remain unaware of the details of how and where the data items are stored in a distributed system
Distributed database environments

Homogeneous
  - Autonomous
  - Non-Autonomous

Heterogeneous
  - Systems
    - Full DBMS functionality
    - Partial-Multidatabase
      - Federated
        - Loose integration
      - Unfederated
        - Tight integration
  - Gateways
Homogeneous vs Heterogeneous DDB

- **In a homogeneous distributed database**
  - All sites have identical software (DBMS)
  - Are aware of each other and agree to cooperate in processing user requests.
  - Each site surrenders part of its autonomy in terms of right to change schemas
  - Appears to user as a single system
  - Autonomous - Independent DBMSs
Homogeneous vs Heterogeneous DDB

- In a heterogeneous distributed database
  - Different sites may use different schemas and software
    - Difference in schema is a major problem for query processing
    - Difference in software is a major problem for transaction processing
  - Sites may not be aware of each other and may provide only limited facilities for cooperation in transaction processing
Data allocation

- Distributed DB stores logically related data at several independent sites connected via network
- **Data allocation** is the process of deciding where to allocate/store particular data
- There are 3 data allocation strategies:
  1. Centralized
  2. Partitioned/Fragmented
  3. Replicated
Fragmentation/Replication
Data Fragmentation

- Breaking up the database into logical units called fragments and assigned for storage at various sites.

- Types of Fragmentation
  - **Horizontal Fragmentation**
  - **Vertical Fragmentation**
  - **Mixed (Hybrid) Fragmentation**

- Fragmentation Schema
  - Definition of a set of fragments that include all attributes and tuples in the database
  - The whole database can be reconstructed from the fragments
Fragmentation

**Horizontal fragmentation**

- It is a horizontal subset of a relation which contain those tuples which satisfy selection conditions.

**Vertical fragmentation**

- It is a subset of a relation which is created by a subset of columns. Thus a vertical fragment of a relation will contain values of selected columns. There is no selection condition used in vertical fragmentation.

**Mixed Fragmentation**

- Intermixing the two types of fragmentation
- Take some tuples and some attributes of a relation and keep it in a different node.
Data Replication

- Process of storing the whole or some part of the data in more than one site
- Fully replicated distributed database
  - Replicating the whole database at every site
  - Improves availability
  - Improves performance of retrieval
  - Can slow down update operations drastically
  - Expensive concurrency control and recovery techniques
Issues for Data Replication

- Data timeliness – high tolerance for out-of-date data may be required
- DBMS capabilities – if DBMS cannot support multi-node queries, replication may be necessary
- Performance implications – refreshing may cause performance problems for busy nodes
- Network heterogeneity – complicates replication
- Network communication capabilities – complete refreshes place heavy demand on telecommunications
Comparison of Distributed Database Design

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Reliability</th>
<th>Expandability</th>
<th>Communications Overhead</th>
<th>Manageability</th>
<th>Data Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized</td>
<td>POOR:</td>
<td>POOR:</td>
<td>VERY HIGH:</td>
<td>VERY GOOD:</td>
<td>EXCELLENT:</td>
</tr>
<tr>
<td></td>
<td>Highly dependent on central server</td>
<td>Limitations are barriers to performance</td>
<td>High traffic to one site</td>
<td>One, monolithic site requires little coordination</td>
<td>All users always have same data</td>
</tr>
<tr>
<td>Replicated with snapshots</td>
<td>GOOD:</td>
<td>VERY GOOD:</td>
<td>LOW to MEDIUM:</td>
<td>VERY GOOD:</td>
<td>MEDIUM:</td>
</tr>
<tr>
<td></td>
<td>Redundancy and tolerated delays</td>
<td>Cost of additional copies may be less than linear</td>
<td>Not constant, but periodic snapshots can cause bursts of network traffic</td>
<td>Each copy is like every other one</td>
<td>Fine as long as delays are tolerated by business needs</td>
</tr>
<tr>
<td>Synchronized replication</td>
<td>EXCELLENT:</td>
<td>VERY GOOD:</td>
<td>MEDIUM:</td>
<td>MEDIUM:</td>
<td>MEDIUM to TO VERY GOOD:</td>
</tr>
<tr>
<td></td>
<td>Redundancy and minimal delays</td>
<td>Cost of additional copies may be low and synchronization work only linear</td>
<td>Messages are constant, but some delays are tolerated</td>
<td>Collisions add some complexity to manageability</td>
<td>Close to precise consistency</td>
</tr>
<tr>
<td>Integrated partitions</td>
<td>VERY GOOD:</td>
<td>VERY GOOD:</td>
<td>LOW to MEDIUM:</td>
<td>VERY GOOD:</td>
<td>VERY POOR:</td>
</tr>
<tr>
<td></td>
<td>Effective use of partitioning and redundancy</td>
<td>New nodes get only data they need without changes in overall database design</td>
<td>Most queries are local but queries which require data from multiple sites can cause a temporary load</td>
<td>Easy for each site, until there is a need to share data across sites</td>
<td>Considerable effort, and inconsistencies not tolerated</td>
</tr>
<tr>
<td>Decentralized with independent partitions</td>
<td>GOOD:</td>
<td>GOOD:</td>
<td>LOW:</td>
<td>VERY GOOD:</td>
<td>LOW:</td>
</tr>
<tr>
<td></td>
<td>Depends on only local database availability</td>
<td>New sites independent of existing ones</td>
<td>Little if any need to pass data or queries across the network (if one exists)</td>
<td>Easy for each site, until there is a need to share data across sites</td>
<td>No guarantees of consistency, in fact pretty sure of inconsistency</td>
</tr>
</tbody>
</table>
Database Security
Backup and Recovery
A database should be properly secured using appropriate controls.

Database security encompasses hardware, software, people and data.

Database security and integrity is about protecting the database from being inconsistent and being disrupted. We can also call it database misuse.

Database misuse could be Intentional or accidental, where accidental misuse is easier to cope with than intentional misuse.
Cont...

- Accidental inconsistency could occur due to:
  - System crash during transaction processing
  - Anomalies due to concurrent access
  - Anomalies due to redundancy
- Likewise, even though there are various threats that could be categorized in this group, intentional misuse could be:
  - Unauthorized reading of data
  - Unauthorized modification of data or
  - Unauthorized destruction of data
Most systems implement good **Database Integrity** to protect the system from accidental misuse while there are many computer based measures to protect the system from intentional misuse, which is termed as **Database Security** measures.

Database security is considered in relation to the following situations:

- *Theft and fraud*
- *Loss of confidentiality (secrecy)*
- *Loss of privacy*
- *Loss of integrity*
- *Loss of availability*
Security Mechanisms

- **Discretionary Security Mechanisms**
  - To grant privileges to users, including the capability to access specific data files, records, or fields in a specific mode (such as read, insert, delete, or update).

- **Mandatory Security Mechanisms**
  - To enforce multilevel security by classifying the data and users into various security classes (or levels) and then implementing the appropriate security policy of the organization.
• **Levels of Security Measures**
  
  • Security measures can be implemented at several levels and for different components of the system. These levels are:
  
  • **Physical Level:** concerned with securing the site containing the computer system should be physically secured.
    
    • The backup systems should also be physically protected from access except for authorized users.
  
  • **Human Level:** concerned with authorization of database users for access the content at different levels and privileges.
**Operating System:** concerned with the weakness and strength of the operating system security on data files.
- Weakness may serve as a means of unauthorized access to the database.
- This also includes protection of data in primary and secondary memory from unauthorized access.

**Database System:** concerned with data access limit enforced by the database system.
- Access limit like password, isolated transaction and etc.

Even though we can have different levels of security and authorization on data objects and users, *who access which data is a policy matter rather than technical.*
Any database access request will have the following three major components

- **Requested Operation**: what kind of operation is requested by a specific query?
- **Requested Object**: on which resource or data of the database is the operation sought to be applied?
- **Requesting User**: who is the user requesting the operation on the specified object?

The database should be able to check for all the three components before processing any request.

The checking is performed by the security subsystem of the DBMS.
There are different forms of user authorization on the resource of the database.

These forms are privileges on what operations are allowed on a specific data object.

**User authorization on the data/extension**

- **Read Authorization**: the user with this privilege is allowed only to read the content of the data object.
- **Insert Authorization**: the user with this privilege is allowed only to insert new records or items to the data object.
Cont...

- **Update Authorization**: users with this privilege are allowed to modify content of attributes but are not authorized to delete the records.

- **Delete Authorization**: users with this privilege are only allowed to delete a record and not anything else.
  - Different users, depending on the power of the user, can have one or the combination of the above forms of authorization on different data objects.
The major responsibilities of DBA in relation to authorization of users are:

- **Account Creation**: involves creating different accounts for different **USERS** as well as **USER GROUPS**.
- **Security Level Assignment**: involves in assigning different users at different categories of access levels.
- **Privilege Grant**: involves giving different levels of privileges for different users and user groups.
Cont...

- **Privilege Revocation**: involves denying or canceling previously granted privileges for users due to various reasons.

- **Account Deletion**: involves in deleting an existing account of users or user groups. Is similar with denying all privileges of users on the database.
Granting Privileges

- The owner account holder can pass privileges on any of the owned relations to other users by granting privileges to their accounts.

- In SQL the following types of privileges can be granted on each individual relation $R$:
  - SELECT
  - MODIFY (UPDATE, DELETE, INSERT)
    - INSERT and UPDATE - certain attributes
REVOKE & GRANT OPTION

- REVOKE
  - To cancel privileges

- GRANT option
  - If grant option is given then the account can also grant that privilege on the relation to other accounts.
  - Suppose that B is given the GRANT OPTION by A and that B then grants the privilege on R to a third account C, also with GRANT OPTION.
  - In this way, privileges on R can propagate to other accounts without the knowledge of the owner of R.
  - If the owner account A now revokes the privilege granted to B, all the privileges that B propagated based on that privilege should automatically be revoked by the system.