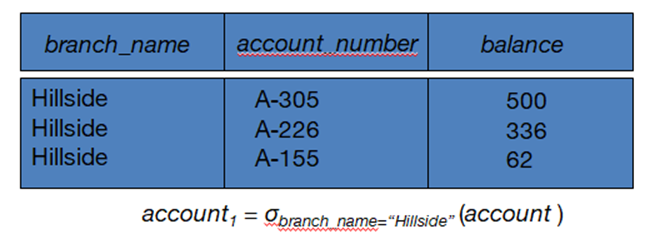
**A distributed database (DDB)** system is comprised of multiple sites that are loosely coupled, they do not share any physical components. Each site operates its own independent database system, and transactions can access data from one or more sites. These databases are managed by a distributed database management system (DDBMS), which ensures that data distribution across sites remains transparent to users. The DDBMS coordinates transactions and data access across the distributed environment, enabling efficient and reliable data management across various locations.

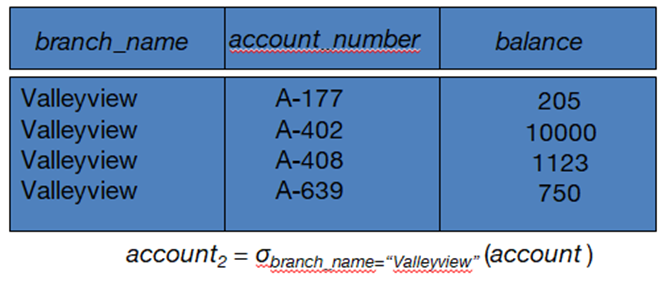
**Advantages:-**

* *Distribution Transparency:* Users are unaware of the physical placement of data, such as files or relations, as the system manages data distribution internally.
* *Replication Transparency:* Data can be replicated across multiple sites transparently to users, which helps minimize access time and enhances fault tolerance.
* *Fragmentation Transparency:* Users can fragment data without impacting application's functionality, allowing for efficient data storage & retrieval.
* *Location Transparency:* Users interact with the DDB system as if it were a centralized system, regardless of the complexities of the underlying network.
* *Naming Transparency:* Users can access named objects (files, relations, etc.) from any location without being aware of the specific location of the data.
* *Reliability*: Distributed database systems improve system uptime by distributing data across multiple nodes, reducing the likelihood of system failure or downtime.
* *Availability:* The system maintains continuous accessibility and usability, even if individual nodes fail, ensuring uninterrupted service to users.
* *Fragmentation:* The system fragments the database to store data closer to where it is needed most.
* *Easier Expansion (Scalability):* Distributed database systems allow for seamless addition of new nodes without requiring a complete reconfiguration of the system.

**Data fragmentation** in a DB involves dividing a relation (table) into smaller, more manageable pieces (fragments). These fragments contain subsets of the original relation's data, they contain all the information needed to reconstruct the original relation. Improve efficiency, performance, & manageability of the DB system.

1. **Horizontal fragmentation** partitions a relation into subsets of rows that satisfy specific selection conditions.

*Example*, in an Employee relation, if the selection condition is (DNO = 5), the horizontal fragment would contain all tuples where the department number (DNO) equals 5.

*Derived horizontal fragmentation:* partitioning a primary relation into secondary related to the primary relation through foreign key constraints.

Each horizontal fragment of a relation can be specified using the *selection operation (σ)* in relational algebra. Example, σCi (R) selects tuples from relation R that satisfy condition Ci.

*Complete horizontal fragmentation* involves each partition containing a subset of rows from the original relation. However, the condition for each fragment does not necessarily cover all tuples in the original relation.

*Disjoint complete horizontal fragmentation* ensures that tuples in different fragments do not overlap, preventing redundancy and ensuring data integrity.

The UNION operation merges the rows from different fragments, reconstructing the original relation while maintaining data consistency and integrity.

1. **Vertical fragmentation** creates subsets of a relation by selecting specific columns.

*Example*, in an Employee relation, a vertical fragment may include columns.

Each fragment must include the *primary key attribute(s)* of the parent relation. This ensures that all vertical fragments of table remain connected and allow for proper reconstruction.

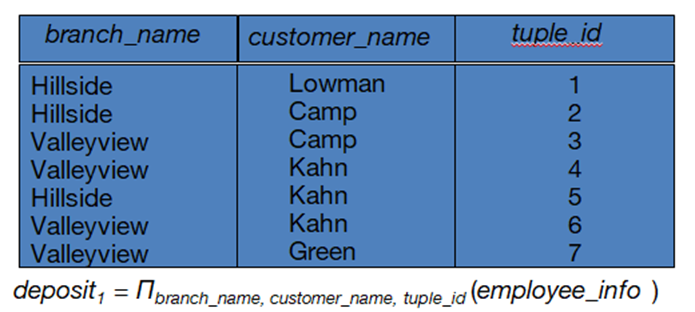
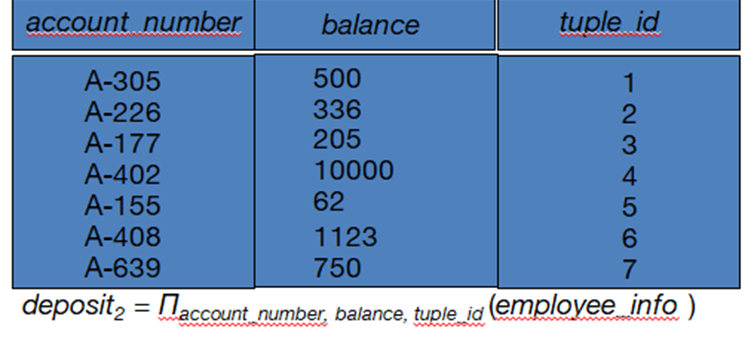
Fragment of table can be specified using *projection operation (Π)* in relational algebra. *Example,* ΠLi(R) selects the columns specified by the list Li from relation R.

*Complete vertical fragmentation*: each fragment's projection list includes all attributes of the original relation. However, the fragments share only the primary key attribute(s).

The *union* of all projection lists covers all attributes of the original relation.

The *intersection* of any pair of projection lists results in the primary key attribute(s).

*OUTER UNION* operation merges the columns from different fragments, reconstructing the original relation while maintaining data integrity and coherence.



1. **Mixed (Hybrid) fragmentation** is a combination of both vertical and horizontal fragmentation in a distributed database. It involves using SELECT-PROJECT operations, typically represented as ΠLi(σCi (R)). The choice of selection conditions and projection lists determines the characteristics of the resulting fragments, offering a balance b/w optimizing storage & retrieval efficiency.

**Advantages**:-

1. *Horizontal Fragmentation:*

Parallel Processing: distributing workload across multiple nodes or processors, improving performance. Optimized Access: It allows a relation to be split so that tuples are located where they are most frequently accessed, reducing access time and enhancing efficiency.

1. *Vertical Fragmentation:*

Optimized Storage and Access: Allows tuples to be split so that each part of the tuple is stored where it is most frequently accessed, optimizing storage and access patterns.

Efficient Joining: Efficient joining of vertical fragments, improving query performance.

Parallel Processing: Further enhancing performance.

**Fragmentation Schema:** Defines a set of fragments that covers all attributes and tuples in the database. The fragmentation schema ensures that the entire database can be reconstructed from the fragments using a sequence of UNION or OUTER JOIN operations, maintaining data integrity and completeness.

**Allocation Schema:** Describes the Fragment distribution to sites within the DDB system. Fragments can be fully or partially replicated, allowing for redundancy and fault tolerance, also be partitioned across different sites, optimizing data distribution based on factors such as access patterns, workload, and network topology.

**Replication:**

Storing redundant copies of a relation or fragment of a relation across multiple sites in a DDB system. Full replication means that the entire database or relation is replicated at all sites, while partial replication involves replicating only selected parts to specific sites.

Replication schemas are used to define how data replication is managed across the distributed system.

***Advantages***:

* *Availability:* If a site containing a relation fails, the data remains available if replicas exist at other sites, ensuring uninterrupted access to data.
* *Parallelism:* Queries can be processed concurrently by multiple nodes in parallel, enhancing query performance and system throughput.
* *Reduced Data Transfer:* Replicated data is available locally at each site containing a replica, reducing the need for data transfer across network for queries & operations.

***Disadvantages***:

* *Increased Cost of Updates:* Updating replicated data requires modifications to be propagated to each replica, increasing overhead & complexity of update operations.
* *Complexity of Concurrency Control:* Concurrent updates to different replicas may lead to inconsistent data if proper concurrency control mechanisms are not in place.

**Data Transparency:** Extent to which system users can remain unaware of the intricacies regarding how and where data items are stored within a distributed system. It ensures that users can interact with the system without needing to concern themselves with the underlying data storage and management details.

1. *Distribution Transparency:*

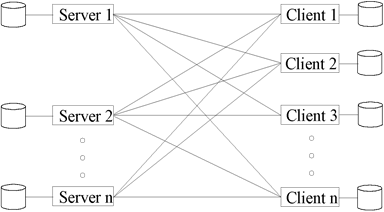
It shields users from the complexities of data distribution across multiple nodes or sites in a distributed system. Users are unaware of which data items are stored at which sites and how system manages data distribution. It ensures that users can access and manipulate data without needing to consider the physical location or distribution of data items.

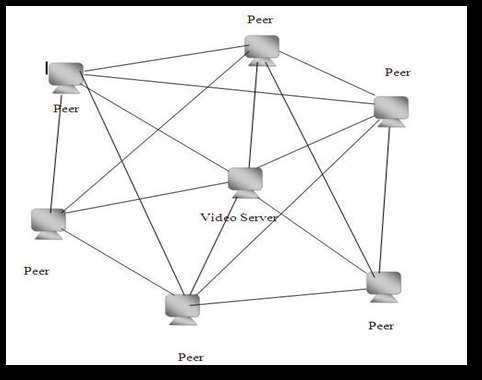
1. *Location Transparency:*

It allows users to access data without needing to know the specific physical or network location of the data. Users can issue commands or queries to retrieve data without needing to specify the exact location where the data is stored. The system handles the details of locating and retrieving data transparently to the user.

1. Replication Transparency:

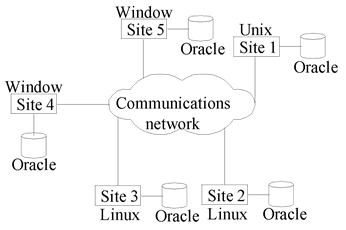
It shields users from the complexities of data replication across multiple sites in a distributed system. Users are unaware of which data items are replicated and how replication is managed within the system. It ensures that users can access and manipulate data without needing to consider the presence or management of data replicas.

**Client-Server Database Architecture:**

* *Clients:* Clients run client software and initiate requests for database services or data retrieval. They interact with the system through user interfaces or applications.
* *Servers:* Servers provide database functionalities and manage data storage, retrieval, and processing. They respond to client requests & execute DB operations.
* *Communication Infrastructure:* A reliable communication infrastructure for comms b/w clients & servers, ensuring seamless data exchange & interaction.
* Clients request services from servers but don’t provide services to other clients or servers. Servers manage local data at their sites, similar to centralized DBMS.
* The client parses user queries and decomposes them into independent sub-queries. Each *sub-query* is sent to the appropriate server for execution based on data distribution or query optimization strategies. Servers process sub-queries and send the results back to the client. The client combines the results of sub-queries and produces the final result for the user.

**Peer-to-Peer (P2P) Database Architecture:**

* *Equal Nodes (Peers):* In a P2P architecture, all nodeshave equal status, acting both as clients and servers. There is *no central server*; instead, each peer contributes resources and participates in data sharing.
* Data is distributed across multiple peers in a decentralized manner. Each peer holds a portion of the data and has autonomous control over its data, and there is no single point of control.
* Queries can be distributed across multiple peers, and each peer processes its part of the query locally. Results are then combined or shared among peers as needed. Peers collaborate in processing queries and sharing resources.
* Easily scalable by adding more peers to the network.No single point of failure, promoting fault tolerance. Peers can share computational and storage resources.

**Homogeneous Distributed Database:**

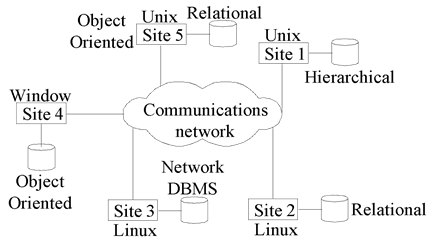
* All sites have identical software setups, meaning they use same DB system software.
* While the underlying operating systems may vary (Linux, Windows, Unix, etc.), the DB system software (e.g., Oracle, DB2, Sybase) remains consistent across all sites.
* Sites are aware of each other and cooperate in processing user requests.
* To the user, the distributed database system appears as a single, integrated system.

*Advantages:*

1. Identical software ensures consistent (non changing) operations.
2. Sites collaborate efficiently in processing requests.
3. Uniform setups simplify system management.
4. Users see it as a single, integrated system.

*Disadvantages:*

1. Sites must conform to the same schema and software.
2. Changes require coordination across all sites.
3. Scaling is constrained by uniform architecture.

**Heterogeneous Distributed Database:**

* Different sites may use different schemas and database system software.
* Differences in schema and software pose significant challenges for query processing and transaction management.
* Sites may not be fully aware of each other and may only provide limited cooperation in transaction processing.
* Data models, transaction commit protocols, concurrency control mechanisms, & system-level details may vary significantly across sites. Many database applications require data from various databases located on diverse hardware and software platforms.

*Advantages:*

1. Allows diverse schemas and software usage.
2. More adaptable configurations for diverse needs.
3. Access to various data sources and formats.
4. Different systems may offer unique features.

Disadvantages:

1. Managing diverse systems involves complexity.
2. Integration may require complex solutions.
3. Differences may lead to compatibility challenges.
4. Supporting diversity may require extra effort.

**Multidatabase System:**

* Each database can use different system to organize its data. Example, one might use Oracle, another might use MySQL, etc. The key point is that they're all different.
* It creates illusion of logical database integration without physically integrating DBs.
* Each site may run different db systems, but data access is managed through a single conceptual schema, implying minimum local autonomy.
* Federated systems typically have a global schema and centralized access policies.