## 1.2 DISTRIBUTED DATA STORAGE

Consider a relation r that is to be stored in the database. There are two approaches to storing this relation in the distributed database.

**Replication:**

The system maintains several identical replicas of the relation, and stores each replica at different site. The alternative to replication is to store only one copy of relation r.

**Fragmentation:**

The system partitions the relation into several fragments, and stores each fragment at a different site.

**Data Replication**

If relation r is replicated, a copy of relation r is stored in two or more sites. In the most extreme case, we have full replication, in which a copy is stored in every site in the system.

There are a number of advantages and disadvantages to replication. **Availability:** If one of the sites containing relation r fails, then the relation r can be found in another site. Thus, the system can continue to process queries involving r, despite the failure of one site.

**Increased parallelism**. In the case where the majority of accesses to the relation r result in only the reading of the relation, then several sites can process queries involving r in parallel.

The more replicas of r there are, the greater the chance that the needed data will be found in the site where the transaction is executing. Hence, data replication minimizes movement of data between sites.

**Increased overhead on update**.

The system must ensure that all replicas of a relation r are consistent; otherwise, erroneous computations may result. Thus, whenever is updated, the update must be propagated to all sites containing replicas.

The result is increased overhead. For example, in a banking system, where account information is replicated in various sites, it is necessary to ensure that the balance in a particular account agrees in all sites.

**Data Fragmentation** : If relation r is fragmented, r is divided into a number of fragments r1, r2,...,rn. These fragments contain sufficient information to allow reconstruction of the original relation r.

There are two different schemes for fragmenting a relation: horizontal fragmentation and vertical fragmentation.

Horizontal fragmentation splits the relation by assigning each tuple of r to one or more fragments.

Vertical fragmentation splits the relation by decomposing the scheme R of relation r.

In horizontal fragmentation, a relation r is partitioned into a number of subsets, r1, r2,...,rn. Each tuple of relation r must belong to at least one of the fragments, so that the original relation can be reconstructed, if needed.

account1 = branch name = Hillside (account)

account2 = branch name = Valleyview (account)

Horizontal fragmentation is usually used to keep tuples at the sites where they are used the most, to minimize data transfer.

In general, a horizontal fragment can be defined as a selection on the global relation r. That is, we use a predicate Pi to construct fragment ri:

$$r_i = \sigma_{p_i}(r)$$

We reconstruct the relation r by taking the union of all fragments; that is: r = r1 r2 ⋯ rn

**Transperency** :

The user of a distributed database system should not be required to know where the data are physically located nor how the data can be accessed at the specific local site. This characteristic, called data transparency, can take several forms:

**Fragmentation transparency**. Users are not required to know how a relation has been fragmented.

**Replication transparency**. Users view each data object as logically unique. The distributed system may replicate an object to increase either system performance or data availability. Users do not have to be concerned with what data objects have been replicated, or where replicas have been placed.

**Location transparency**. Users are not required to know the physical location of the data. The distributed database system should be able to find any data as long as the data identifier is supplied by the user transaction.

## 1.3 DISTRIBUTED TRANSACTIONS

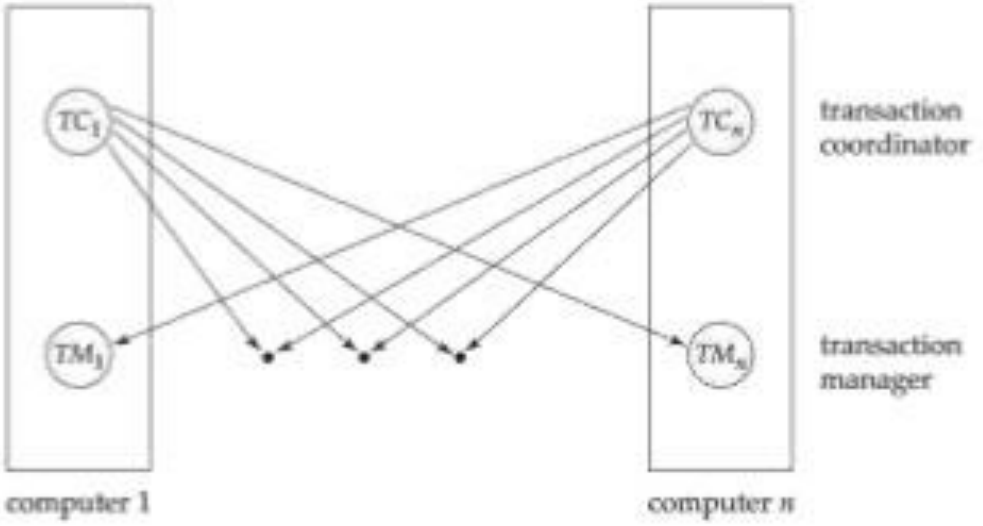
There are two types of transaction that we need to consider.

- Local transactions are those that access and update data in only one local database;
- Global transactions are those that access and update data in several local databases.

**System Structure**

Each site has its own local transaction manager, whose function is to ensure the ACID properties of those transactions that execute at that site. The various transaction managers cooperate to execute global transactions. To understand how such a manager can be implemented, consider an abstract model of a transaction system, in which each site contains two subsystems:

- The **transaction manager** manages the execution of those transactions (or sub transactions) that access data stored in a local site.
- The **transaction coordinator** coordinates the execution of the various transactions (both local and global) initiated at that site.
- Maintaining a log for recovery purposes.
- Participating in an appropriate concurrency-control scheme to coordinate the concurrent execution of the transactions executing at that site.




The transaction coordinator subsystem is not needed in the centralized environment, since a transaction accesses data at only a single site. A transaction coordinator, as its name implies, is responsible for coordinating the execution of all the transactions initiated at that site. For each such transaction, the coordinator is responsible for:

• Starting the execution of the transaction.

• Breaking the transaction into a number of sub transactions and distributing these sub transactions to the appropriate sites for execution.

• Coordinating the termination of the transaction, which may result in the transaction being committed at all sites or aborted at all sites.

**System Failure Modes**

- Failure of a site.
- Loss of messages.
- Failure of a communication link.
- Network partition.