Distributed Databases

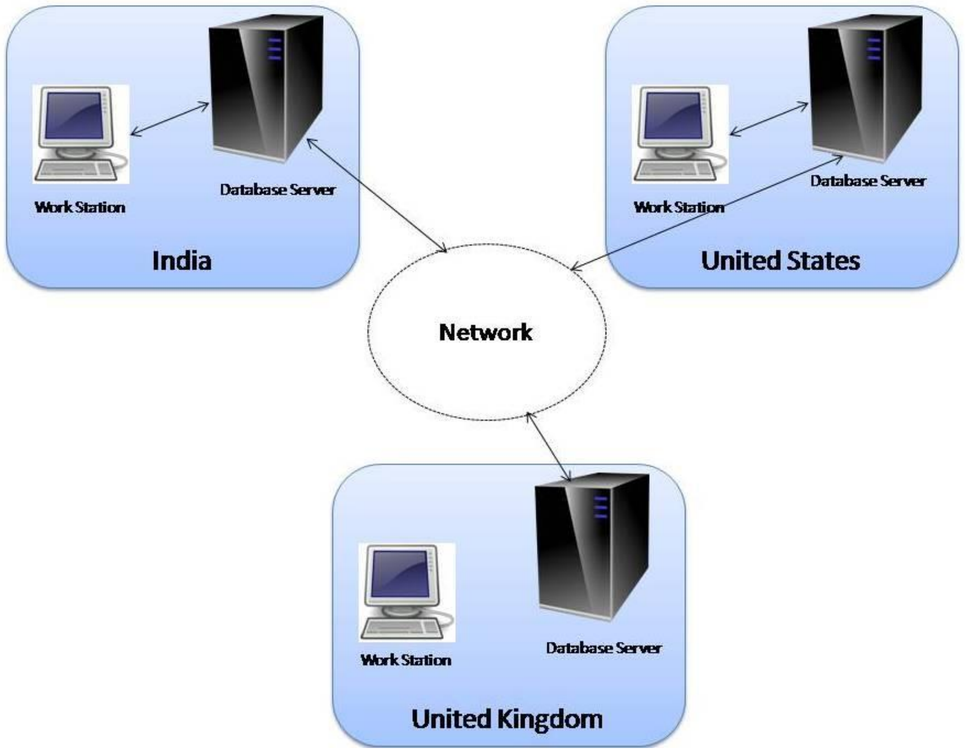
A distributed database is a database in which data is stored across different physical locations. It may be stored in multiple computers located in the same physical location (e.g. a data centre); or maybe dispersed over a network of interconnected computers.

A distributed database system consists of loosely coupled sites that share no physical components.

A distributed database is a database in which not all storage devices are attached to a common processor. It may be stored in multiple computers, located in the same physical location; or may be dispersed over a network of interconnected computers.

A distributed database is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network.

A distributed database is basically a database that is not limited to one system, it is spread over different sites, i.e, on multiple computers or over a network of computers. A distributed database system is located on various sites that don't share physical components. This may be required when a particular database needs to be accessed by various users globally. It needs to be managed such that for the users it looks like one single database.



Types:

1. <u>Homogeneous Database:</u>

In a homogeneous database, all different sites store database identically. The operating system, database management system and the data structures used – all are same at all sites. Hence, they're easy to manage.

2. <u>Heterogeneous Database:</u>

In a heterogeneous distributed database, different sites can use different schema and software that can lead to problems in query processing and transactions. Also, a particular site might be completely unaware of the other sites. Different computers may use a different operating system, different database application. They may even use different data models for the database. Hence, translations are required for different sites to communicate.

Features

Databases in the collection are logically interrelated with each other. They often represent a single logical database.

Data is physically stored across multiple sites. Data in each site can be managed by a DBMS independent of the other sites.

The processors in the sites are connected via a network. They do not have any multiprocessor configuration.

A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system.

The following factors encourage moving over to DDBMS −

Distributed Nature of Organizational Units − Most organizations in the current times are subdivided into multiple units that are physically distributed over the globe. Each unit requires its own set of local data. Thus, the overall database of the organization becomes distributed.

Need for Sharing of Data − The multiple organizational units often need to communicate with each other and share their data and resources. This demands common databases or replicated databases that should be used in a synchronized manner.

Support for Both OLTP and OLAP − Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP) work upon diversified systems which may have common data. Distributed database systems aid both these processing by providing synchronized data.

The following factors encourage moving over to DDBMS – (Contd…)

Database Recovery – One of the common techniques used in DDBMS is replication of data across different sites. Replication of data automatically helps in data recovery if database in any site is damaged. Users can access data from other sites while the damaged site is being reconstructed. Thus, database failure may become almost inconspicuous to users.

Support for Multiple Application Software – Most organizations use a variety of application software each with its specific database support. DDBMS provides a uniform functionality for using the same data among different platforms.

Advantages of Distributed database

Basically, we can define a Distributed database as a collection of multiple interrelated databases distributed over a computer network and a distributed database management system as a software system that basically manages a distributed database while making the distribution transparent to the user.

Distributed database management is basically proposed for the various reason from organizational decentralization and economical processing to greater autonomy. Some of these advantages are as follows:

1. Management of data with different level of transparency –
Ideally, a database should be distribution transparent in the sense of hiding the details of where each file is physically stored within the system. The following types of transparencies are basically possible in the distributed database system:

- **Network transparency:**
  This refers to the freedom for the user from the operational details of the network. These are of two types : Location and Naming transparency.
- **Replication transparencies:**
  It made user unaware of the existence of copies as we know that copies of data may be stored at multiple sites for better availability, performance and reliability.
- **Fragmentation transparency:**
  It made user unaware about the existence of fragments : vertical fragment or horizontal fragmentation.

Advantages of Distributed database (Contd…)

2. Increased availability –

Availability is defined as the probability that the system is continuously available during a time interval. When the data and DBMS software are distributed over several sites, one site may fail while other sites continue to operate. Users are not able to only access the data that exist at the failed site and this leads to improvement in reliability and availability.

3. Easier Expansion –

In a distributed environment, expansion of the system in terms of adding more data, increasing database sizes adding more processor is much easier.

4. Improved Performance –

We can achieve parallelism by executing multiple queries at different sites by breaking up a query into a number of sub-queries that basically executes in parallel which basically leads to improvement in performance.

Advantages of Distributed database (Contd…)

5. Modular Development − If the system needs to be expanded to new locations or new units, in centralized database systems, the action requires substantial efforts and disruption in the existing functioning. However, in distributed databases, the work simply requires adding new computers and local data to the new site and finally connecting them to the distributed system, with no interruption in current functions.

6. More Reliable − In case of database failures, the total system of centralized databases comes to a halt. However, in distributed systems, when a component fails, the functioning of the system continues, may be at a reduced performance. Hence DDBMS is more reliable.

7. Better Response − If data is distributed in an efficient manner, then user requests can be met from local data itself, thus providing faster response. On the other hand, in centralized systems, all queries have to pass through the central computer for processing, which increases the response time.

8. Lower Communication Cost − In distributed database systems, if data is located locally where it is mostly used, then the communication costs for data manipulation can be minimized. This is not feasible in centralized systems.

Following are some of the disadvanteges associated with distributed databases.

1. Need for complex and expensive software − DDBMS demands complex and often expensive software to provide data transparency and co-ordination across the several sites.

2. Processing overhead − Even simple operations may require a large number of communications and additional calculations to provide uniformity in data across the sites.

3. Data integrity − The need for updating data in multiple sites pose problems of data integrity.

4. Overheads for improper data distribution − Responsiveness of queries is largely dependent upon proper data distribution. Improper data distribution often leads to very slow response to user requests.

There are 2 ways in which data can be stored on different sites. These are:

1. Replication

2. Fragmentation

1. **Replication**
In this approach, the entire data/relation is stored redundantly at two or more sites. If the entire database is available at all sites, it is a fully redundant database. Hence, in replication, systems maintain copies of data.
This is advantageous as it increases the availability of data at different sites. Also, query requests can be processed in parallel.

However, it has certain disadvantages as well. Data needs to be constantly updated. Any change made at one site needs to be recorded at every site where that relation is stored or else it may lead to inconsistency. This is a lot of overhead. Also, concurrency control becomes way more complex as concurrent access now needs to be checked over a number of sites.

2. **Fragmentation**

In this approach, the relations are fragmented (i.e., they're divided into smaller parts) and each of the fragments is stored in different sites where they're required. It must be made sure that the fragments are such that they can be used to reconstruct the original relation (i.e, there isn't any loss of data).

Fragmentation is advantageous as it doesn't create copies of data, and hence, consistency is not a problem.

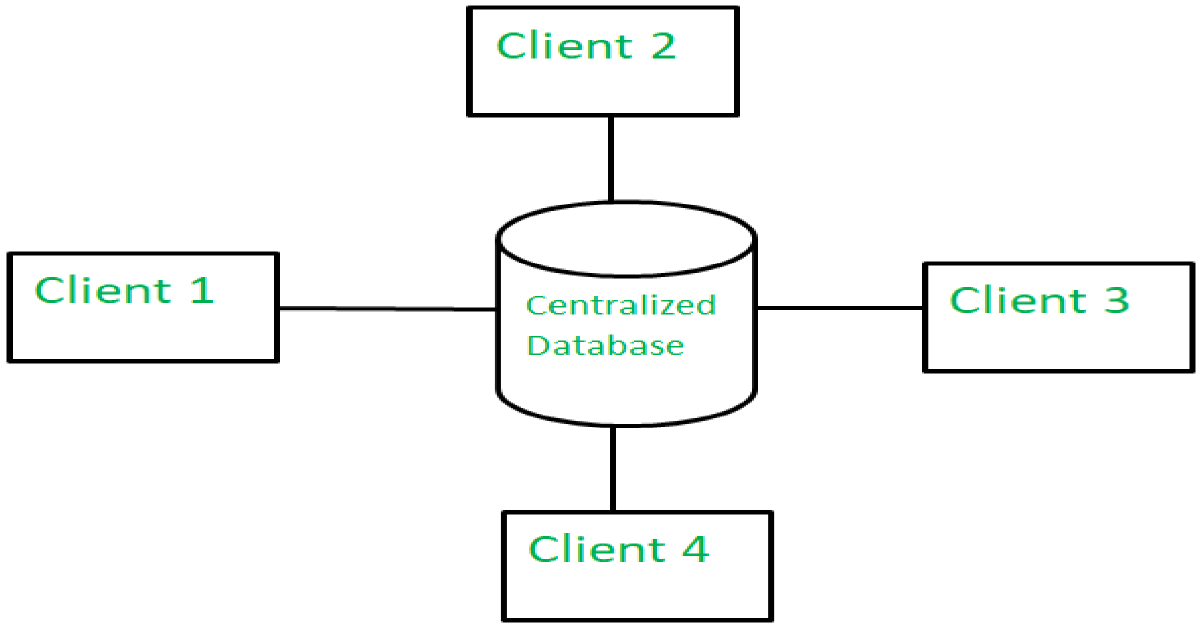
Fragmentation of relations can be done in two ways:

- Horizontal fragmentation – Splitting by rows – The relation is fragmented into groups of tuples so that each tuple is assigned to at least one fragment.
- Vertical fragmentation – Splitting by columns – The schema of the relation is divided into smaller schemas. Each fragment must contain a common candidate key so as to ensure lossless join.

In certain cases, an approach that is hybrid of fragmentation and replication is used.

1. Centralized Database :

A centralized database is basically a type of database that is stored, located as well as maintained at a single location only. This type of database is modified and managed from that location itself. This location is thus mainly any database system or a centralized computer system. The centralized location is accessed via an internet connection (LAN, WAN, etc). This centralized database is mainly used by institutions or organizations.



Centralized Database :

Advantages –

- Since all data is stored at a single location only thus it is easier to access and co-ordinate data.
- The centralized database has very minimal data redundancy since all data is stored at a single place.
- It is cheaper in comparison to all other databases available.

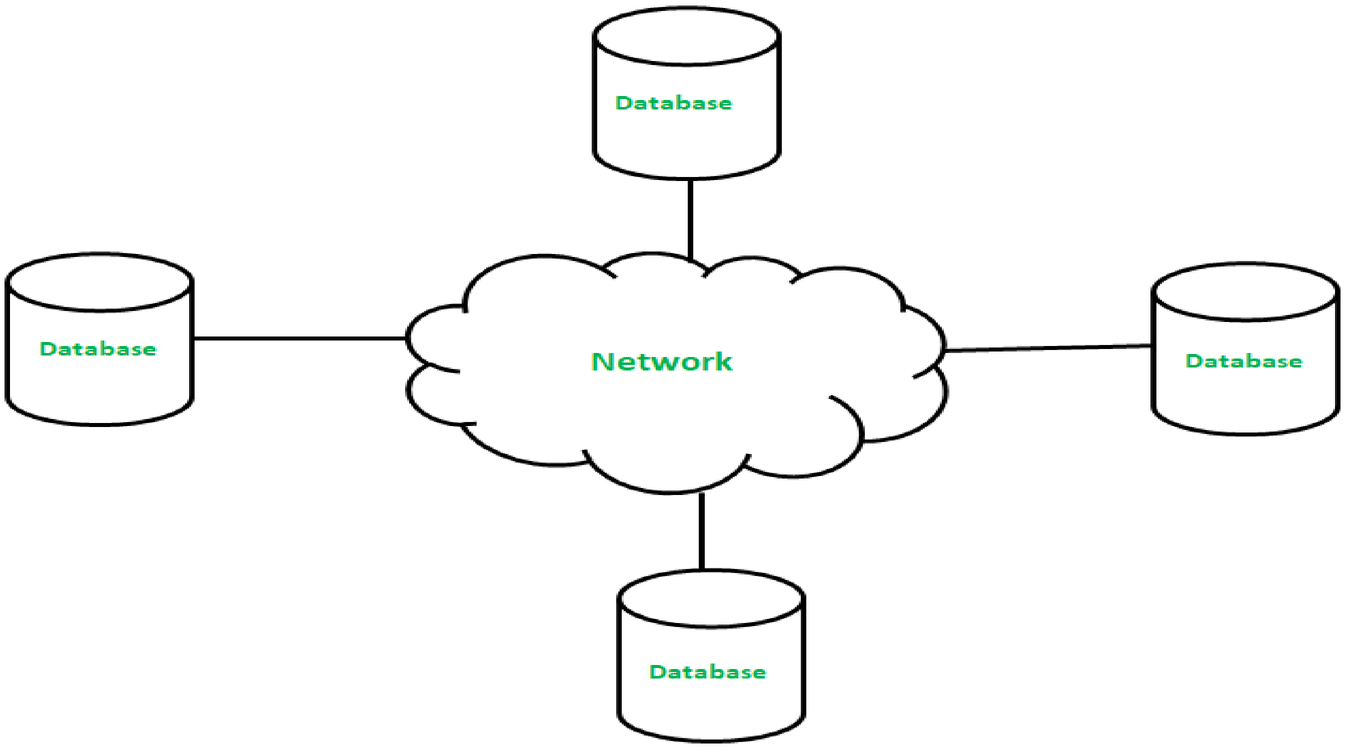
Disadvantages –

- The data traffic in case of centralized database is more.

If any kind of system failure occurs at centralized system then entire data will be destroyed.

2. Distributed Database :

A distributed database is basically a type of database which consists of multiple databases that are connected with each other and are spread across different physical locations. The data that is stored on various physical locations can thus be managed independently of other physical locations. The communication between databases at different physical locations is thus done by a computer network.



Distributed Database :

Advantages –

- This database can be easily expanded as data is already spread across different physical locations.
- The distributed database can easily be accessed from different networks.
- This database is more secure in comparison to centralized database.

Disadvantages –

- This database is very costly and it is difficult to maintain because of its complexity.
- In this database, it is difficult to provide a uniform view to user since it is spread across different physical locations.

Difference between Centralized DB and Distributed DB :

CD : It is database that is stored, located as well as maintained at a single location only.

DD :It is a database which consists of multiple databases which are connected with each other and are spread across different physical locations.

CD: The data access time in case of multiple users is more in a centralized database.

DD: The data access time in case of multiple users is less in a distributed database.

CD: The management, modification and backup of this database is easier as entire data is present at the same location.

DD: The management, modification and backup of this database is very difficult as it is spread across different physical locations.

Difference between CD and DD: (contd…)

CD: This database provides a uniform and complete view to the user.
DD: Since it is spread across different locations thus it is difficult to provide a uniform view to the user.

CD: This database has more data consistency in comparison to distributed database.
DD: This database may have some data replications thus data consistency is less.

CD: The users cannot access database in case database failure occurs.
DD: In distributed database if one database fails users have access to other databases.

CD: Centralized database is less costly.
DD: This database is very expensive.

The design of a distributed database system is a complex task. Therefore, a careful assessment of the strategies and objectives is required. Some of the strategies and objectives that are common to the most DBS design are as follows:

Data fragmentation, which are applied to relational database system to partition the relations among network sites.

Data allocation, in which each fragment is stored at the site with optimal distribution.

Data replication, which increases the availability and improves the performance of the system.

Location transparency, which enables a user to access data without knowing, or being concerned with, the site at which the data resides. The location of the data is hidden.

Data Fragmentation

Data fragmentation allows you to break a single object into two or more segments, or fragments. The object might be a user’s database, a system database, or a table. Each fragment can be stored at any site over a computer network.

Data fragmentation strategies, as discussed here, are based at the table level and consist of dividing a table into logical fragments. There are three types of data fragmentation strategies: horizontal, vertical, and mixed.

Horizontal fragmentation refers to the division of a relation into subsets (fragments) of tuples (rows). Each fragment is stored at a different node, and each fragment has unique rows. However, the unique rows all have the same attributes (columns).

Vertical fragmentation refers to the division of a relation into attribute (column) subsets. Each subset (fragment) is stored at a different node, and each fragment has unique columns—with the exception of the key column, which is common to all fragments.

Mixed fragmentation refers to a combination of horizontal and vertical strategies. In other words, a table may be divided into several horizontal subsets (rows), each one having a subset of the attributes (columns).

Data Allocation

Data Allocation is an intelligent distribution of data pieces, (data fragments) to improve database performance and Data Availability for end-users. It aims to reduce overall costs of transaction processing while also providing accurate data rapidly in DDBMS systems. Data allocation strategies are as follows:

- With centralized data allocation, the entire database is stored at one site.
- With partitioned data allocation, the database is divided into two or more disjointed parts (fragments) and stored at two or more sites.
- With replicated data allocation, copies of one or more database fragments are stored at several sites.
- Data distribution over a computer network is achieved through data fragmentation, through data replication, or through a combination of both. Data allocation is closely related to the way a database is fragmented. Data allocation focuses on one issue: which data to store/locate where.

Data allocation algorithms take into consideration a variety of factors, including:

- Performance and data availability goals.
- Size, the number of rows, and the number of relations that an entity maintains with other entities.
- Types of transactions to be applied to the database and the attributes accessed by each of those transactions.

Data Replication:

Data replication refers to the storage of data copies at multiple sites served by a computer network. Fragmented copies can be stored at several sites to serve specific information requirements. Because the existence of fragment copies can enhance data availability and response time, data copies can help to reduce communication and total query costs.

Suppose database A is divided into two fragments, A1 and A2. Within a replicated distributed database : fragment A1 can be stored at sites S1 and S2, while fragment A2 can be stored at sites S2 and S3.

Replicated data are subject to the mutual consistency rule. The mutual consistency rule requires that all copies of data fragments be identical. Therefore, to maintain data consistency among the replicas, the DDBMS must ensure that a database update is performed at all sites where replicas exist.

The different replica overheads imposed on DDBMS are as follows.

- If the database is fragmented, the DDBMS must decompose a query into sub-queries to access the appropriate fragments.
- If the database is replicated, the DDBMS must decide which copy to access. A READ operation selects the nearest copy to satisfy the transaction. A WRITE operation requires that all copies be selected and updated to satisfy the mutual consistency rule.
- The TP sends a data request to each selected DP for execution.
- The DP receives and executes each request and sends the data back to the TP.
- The TP assembles the DP responses.
- The problem becomes more complex when you consider additional factors such as network topology and communication throughputs.

Three replication scenarios exist: a database can be fully replicated, partially replicated, or unreplicated.

- A fully replicated database stores multiple copies of each database fragment at multiple sites. In this case, all database fragments are replicated. A fully replicated database can be impractical due to the amount of overhead it imposes on the system.

- A partially replicated database stores multiple copies of some database fragments at multiple sites. Most DDBMSs are able to handle the partially replicated database well.

- An unreplicated database stores each database fragment at a single site. Therefore, there are no duplicate database fragments.

Several factors influence the decision to use data replication:

Database size:
The amount of data replicated will have an impact on the storage requirements and also on the data transmission costs. Replicating large amounts of data requires a window of time and higher network bandwidth that could affect other applications.

Usage frequency:
The frequency of data usage determines how frequently the data needs to be updated. Frequently used data needs to be updated more often, for example, than large data sets that are used only every quarter.

Costs: including those for performance, software overhead, and management associated with synchronizing transactions and their components vs. fault-tolerance benefits that are associated with replicated data.

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.

Data at rest

Data at rest in information technology means inactive data that is stored physically in any digital form (e.g. databases, data warehouses, spreadsheets, archives, tapes, off-site backups, mobile devices etc.). Data at rest is subject to threats from hackers and other malicious threats. To prevent this data from being accessed, modified or stolen, organizations will often employ security protection measures such as password protection, data encryption, or a combination of both. The security options used for this type of data are commonly referred to as data at rest protection (DARP).

Data at motion

Data in motion refers to a stream of data moving through any kind of network. It is one of the two major states of data, the other being data at rest. It can be considered the opposite of data at rest as it represents data which is being transferred or moved, while data at rest is data which is static and is not moving anywhere.

What is Partitioning?

Partitioning is the database process where very large tables are divided into multiple smaller parts. By splitting a large table into smaller, individual tables, queries that access only a fraction of the data can run faster because there is less data to scan. The main goal of partitioning is to aid in maintenance of large tables and to reduce the overall response time to read and load data for particular SQL operations.

When to partition a table?

Here are some suggestions for when to partition a table:

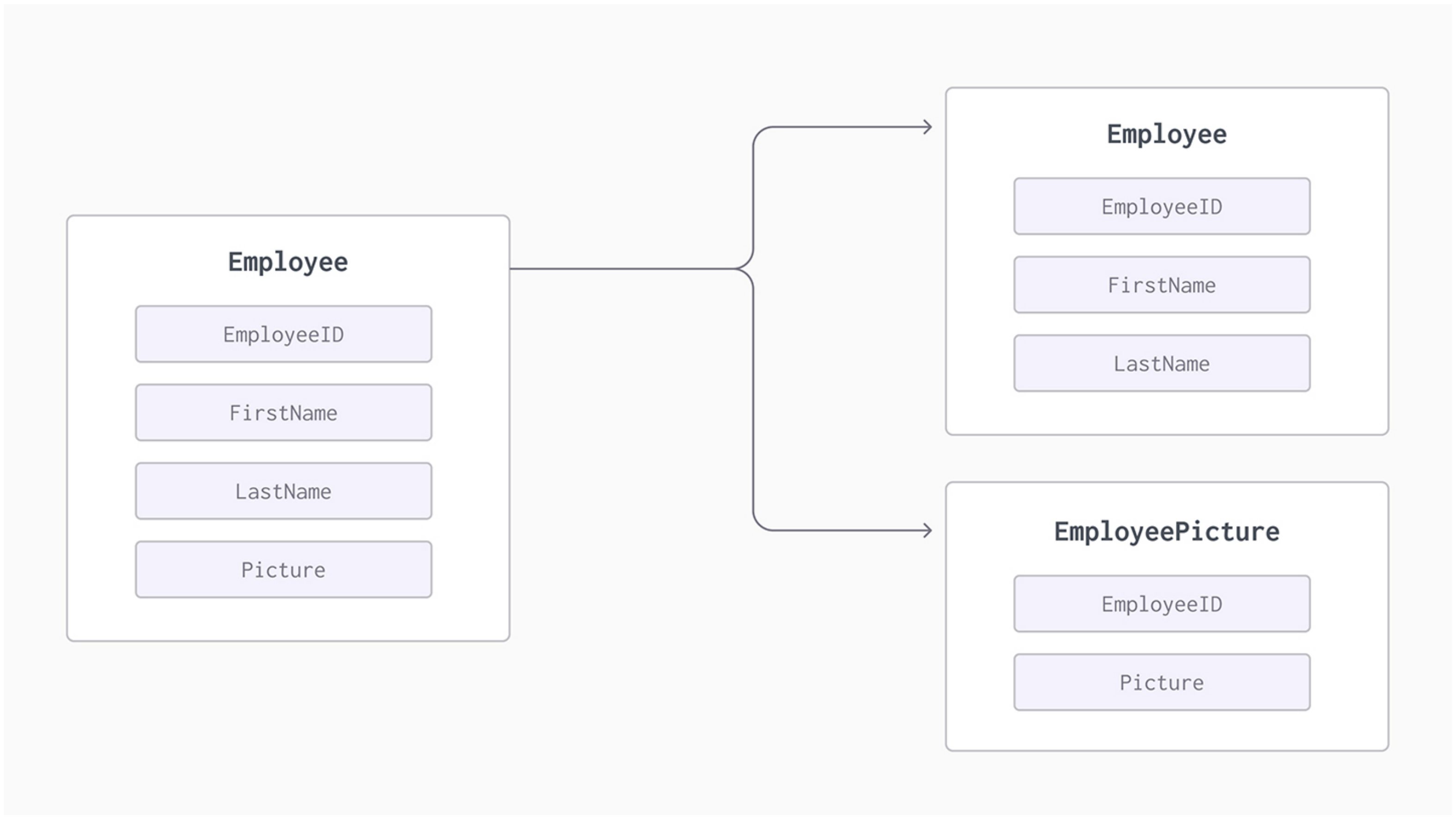
Tables greater than 2 GB should always be considered as candidates for partitioning.

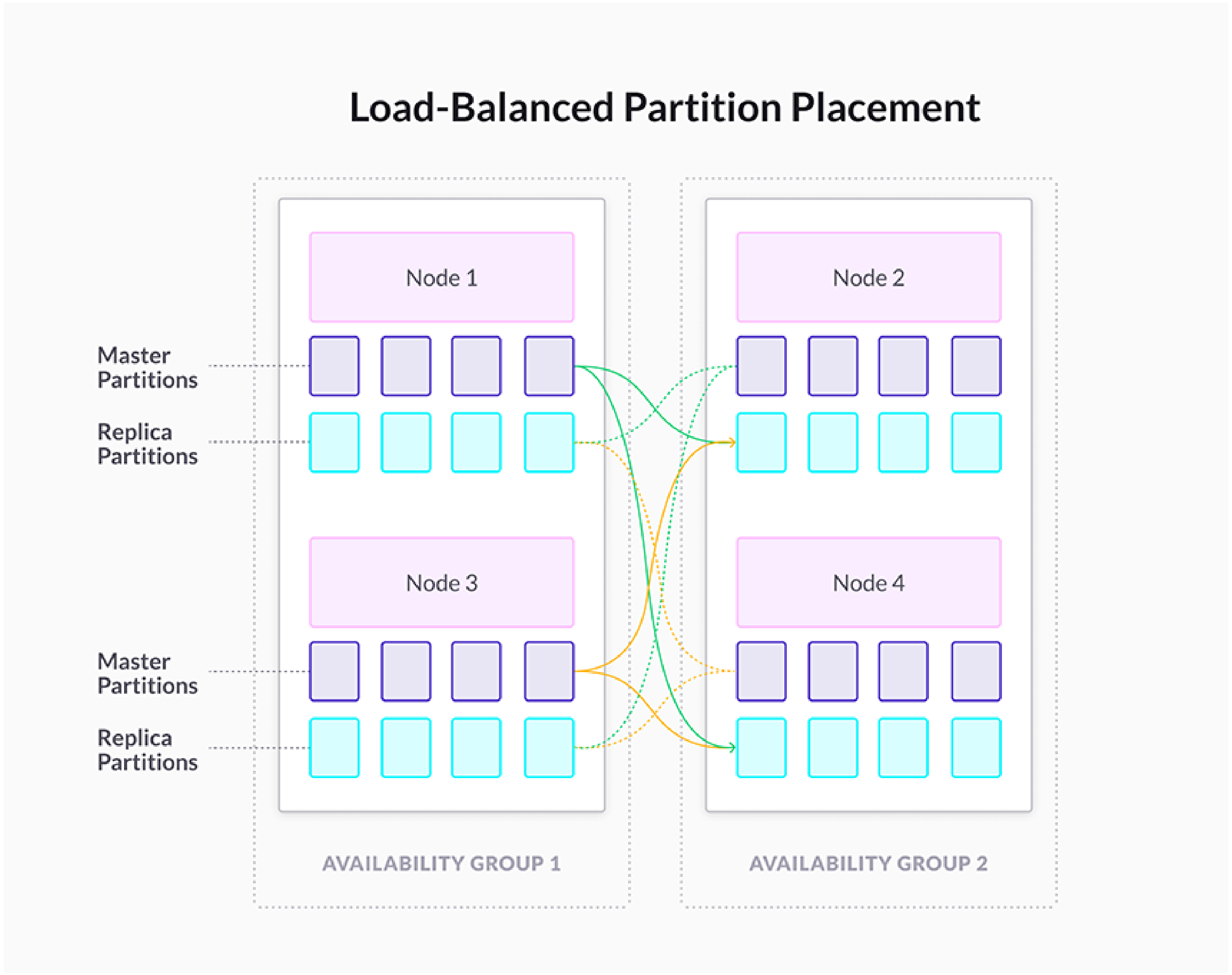
Tables containing historical data, in which new data is added into the newest partition. A typical example is a historical table where only the current month's data is updatable and the other 11 months are read only.

When the contents of a table need to be distributed across different types of storage devices.

Vertical Partitioning

Vertical table partitioning is mostly used to increase SQL Server performance especially in cases where a query retrieves all columns from a table that contains a number of very wide text or BLOB columns. In this case to reduce access times the BLOB columns can be split to its own table. Another example is to restrict access to sensitive data e.g. passwords, salary information etc. Vertical partitioning splits a table into two or more tables containing different columns:


Employee
EmployeeID
FirstName
LastName
Picture
Employee
EmployeeID
FirstName
LastName
EmployeePicture
EmployeeID
Picture


Load-Balanced Partition Placement
Master Partitions
Replica Partitions
Master Partitions
Replica Partitions
Node 1
Node 2
Node 3
Node 4
AVAILABILITY GROUP 1
AVAILABILITY GROUP 2

Sharding

A database shard is a horizontal partition of data in a database or search engine. Each individual partition is referred to as a shard or database shard. Each shard is held on a separate database server instance, to spread load. Some data within a database remains present in all shards, but some appears only in a single **shard**.

Sharding is a type of database partitioning that separates very large databases the into smaller, faster, more easily managed parts called data shards. The word shard means *a small part of a whole*.

Technically, sharding is a synonym for horizontal partitioning. In practice, the term is often used to refer to any database partitioning that is meant to make a very large database more manageable.

The governing concept behind sharding is based on the idea that as the size of a database and the number of transactions per unit of time made on the database increase linearly, the response time for querying the database increases exponentially.

Additionally, the costs of creating and maintaining a very large database in one place can increase exponentially because the database will require high-end computers. In contrast, data shards can be distributed across a number of much less expensive commodity servers. Data shards have comparatively little restriction as far as hardware and software requirements are concerned.

In some cases, database sharding can be done fairly simply. One common example is splitting a customer database geographically. Customers located on the East Coast can be placed on one server, while customers on the West Coast can be placed on a second server. Assuming there are no customers with multiple locations, the split is easy to maintain and build rules around.

Data sharding can be a more complex process in some scenarios, however. Sharding a database that holds less structured data, for example, can be very complicated, and the resulting shards may be difficult to maintain.

When to Shard a table?

Sharding your data can lead to many large performance improvements in your database. The following are some examples of how sharding can help improve performance:

Reduced index size - Since the tables are divided and distributed into multiple servers, the total number of rows in each table in each database is reduced. This reduces index size, which generally improves search performance.

Distribute database over multiple machines - A database shard can be placed on separate hardware, and multiple shards can be placed on multiple machines. This enables a distribution of the database over a large number of machines, greatly improving performance

Segment data by geography - In addition, if the database shard is based on some real-world segmentation of the data (e.g., European customers v. American customers) then it may be possible to infer the appropriate shard membership easily and automatically, and query only the relevant shard.

When NOT to shard a table?

Sharding should be used only when all other options for optimization are inadequate. The complexity of database sharding causes the following potential problems:

SQL complexity - Increased bugs because the developers have to write more complicated SQL to handle sharding logic

Additional software - that partitions, balances, coordinates, and ensures integrity can fail

Single point of failure - Corruption of one shard due to network/hardware/systems problems causes failure of the entire table.

Fail-over server complexity - Fail-over servers must have copies of the fleets of database shards.

Backups complexity - Database backups of the individual shards must be coordinated with the backups of the other shards.

Operational complexity - Adding/removing indexes, adding/deleting columns, modifying the schema becomes much more difficult.

Sharding vs. Partitioning: What’s the Difference?

Sharding and partitioning are both about breaking up a large data set into smaller subsets. The difference is that sharding implies the data is spread across multiple computers while partitioning does not. Partitioning is about grouping subsets of data within a single database instance.

Partitioning is a generic term that just means dividing your logical entities into different physical entities for performance, availability, or some other purpose. "Horizontal partitioning", or sharding, is replicating the schema, and then dividing the data based on a shard key.

On a final note, you can combine both partitioning and sharding techniques on your database. In fact, sometimes using both strategies is required for data-intensive applications.

Clustering

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including pattern recognition, image analysis, information retrieval, bioinformatics, data compression, computer graphics and machine learning.

Cluster is a group of objects that belongs to the same class. In other words, similar objects are grouped in one cluster and dissimilar objects are grouped in another cluster.

Clustering is the process of making a group of abstract objects into classes of similar objects.

Points to Remember about clustering

- A cluster of data objects can be treated as one group.
- While doing cluster analysis, we first partition the set of data into groups based on data similarity and then assign the labels to the groups.
- The main advantage of clustering over classification is that, it is adaptable to changes and helps single out useful features that distinguish different groups.