Databases

CONTENTS:

Centralized Database:

The information(data) is stored at a centralized location and the users from different locations can access this data. This type of database contains application procedures that help the users to access the data even from a remote location. The database is located in a single location, which is generally a CPU server or a Pc or a mainframe computer accessed by many computers through a network.

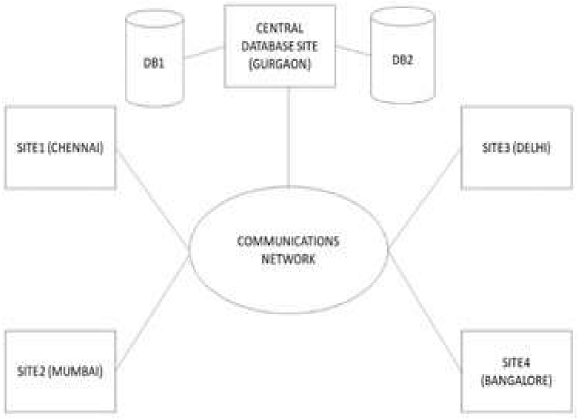


Diagram centralized database.

Shown alongside is a pictorial representation of centralized database we can see Central Database Site is connected to two other databases and a network that then connects local sites at various locations, data is collected from these sites and sent to main database and requested data is supplied by main database to the sites.

Uses of Centralized Database:

Organization needs this in order to maintain integrity of data throughout all the departments.

For example, if an organization have different database for every different department still they can use a centralized database to maintain a record of all the databases within it. Then in case administration requires data of certain employee by just his name and designation rather than searching for all databases they can just check the central database and get the output.

Advantages:

1) There is no redundant data as if any data is ever repeated, we can verify it as all data is at one place.
2) It maintains high security as there is only location for the database and hence only a particular team handles database.
3) Verification, deletion and updating of data is much simpler as the data can be updated at only one place.
4) It requires less power to run and hence it requires low maintenance and low cost.

Disadvantages:

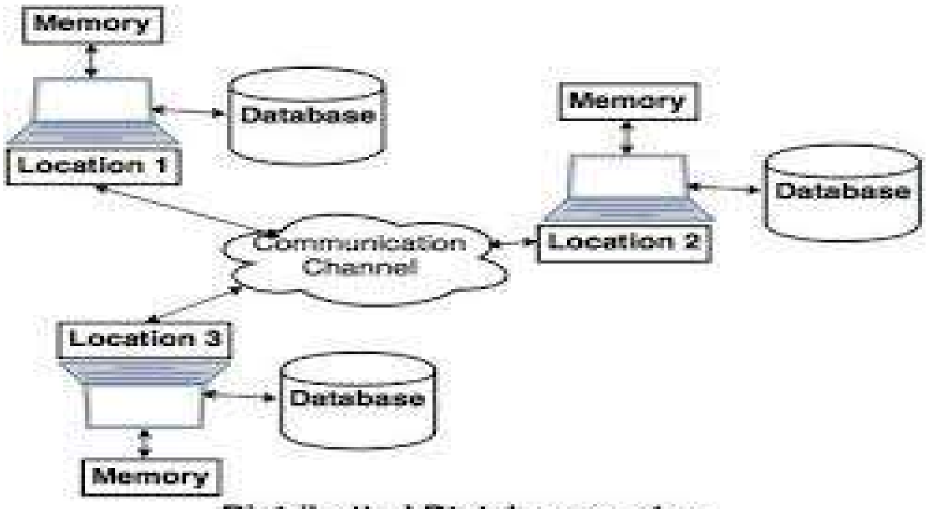
1) Efficiency and speed are compromised when number of users increases.
2) If the data is lost at the central database there can be no retrieval of data.
3) If data at central database is incorrect same incorrect information is sent to all child databases.

Distributed Database

Just opposite of the centralized database concept, the distributed database has contributions from the common database as well as the information captured by local computers also. The data is not at one place and is distributed at various sites of an organization. These sites are connected to each other with the help of communication links which helps them to access the distributed data easily.

You can imagine a distributed database as a one in which various portions of a database are stored in multiple different locations(physical) along with the application procedures which are replicated and distributed among various points in a network.

There are two kinds of distributed database, viz. homogenous and heterogeneous. The databases which have same underlying hardware and run over same operating systems and application procedures are known as homogeneous DDB, for eg. All physical locations in a DDB. Whereas, the operating systems, underlying hardware as well as application procedures can be different at various sites of a DDB which is known as heterogeneous DDB.



Distributted Dtatabase system

Shown above is a pictorial representation of a distributed database system.

We can see that the same database is stored at various locations and a communication channel keeps location and user in touch.

Uses Of Distributed Database:

1) The retrieval of data is simple as the data remains local.
2) In heterogenous system we can also use different OS, different DBMS and even use different schema.
3) Speed is never compromised.
4) It is possible to interconnect two databases at different location.
5) We can say that a DDS is platform independent, Hardware independent.

Advantages:

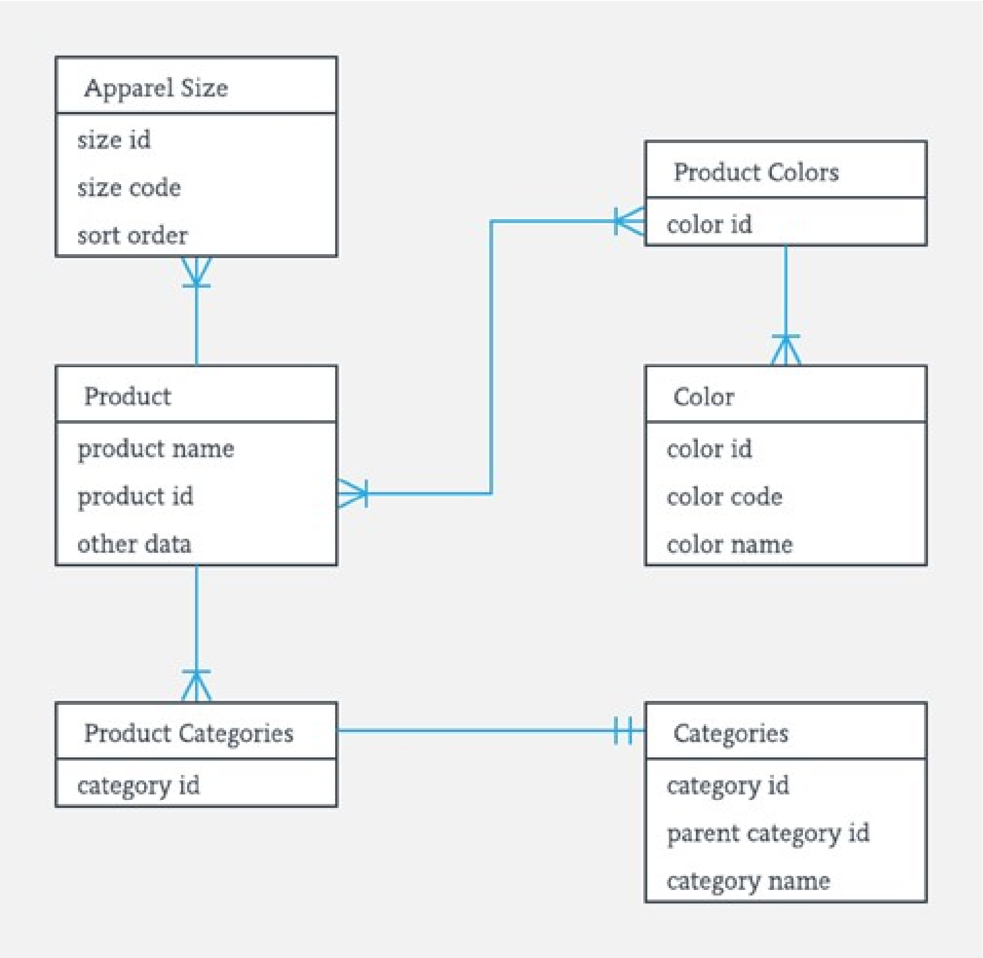
1) New computers can be added to the network and local data can be added without interruption to ongoing operations.
2) If any database fail user can just switch to database at other with identical or compatible schema.
3) Admins can achieve lower communication costs for distributed database systems if the data is located close to where it is used the most. This is not possible in centralized systems.

Disadvantages:

1) static SQL cannot be used.
2) Network traffic is increased.
3) Different data formats maybe used in different databases.
4) Managing distributed deadlock is difficult task.

Relational Databases:

A relational database is a collection of data items with pre-defined relationships between them. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database. Each column in a table holds a certain kind of data and a field stores the actual value of an attribute. The rows in the table represent a collection of related values of one object or entity. Each row in a table could be marked with a unique identifier called a primary key, and rows among multiple tables can be made related using foreign keys. This data can be accessed in many different ways without reorganizing the database tables themselves.



Above is representation of relational database as stated earlier combinations of tables, rows, and columns are shown.

Usually in relational database we use models to represent them one such is called as ER model (entity relationship model).

Uses of Relational Databases:

As it is easy and quick response database because of use of tables for every different entity and its values these type of databases are generally used by ecommerce websites.

Also, data of universities and small offices can be stored in these databases.

Advantages:

1) Very simple model.
2) Data accuracy, use of primary and foreign key it becomes easy to avoid data replication or loose integrity.
3) Data access is easy as a hierarchical tree is always followed.
4) Data integrity is maintained as if table is changed it is followed for every user.
5) **NORMALIZATION** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

Disadvantages:

1) Advances in the complexity of information cause another drawback to relational databases. Relational databases are made for organizing data by common characteristics. Complex images, numbers, designs and multimedia products defy easy categorization leading the way for a new type of database called object-relational database management systems. These systems are designed to handle the more complex applications and have the ability to be scalable.
2) Some relational databases have limits on field lengths. When you design the database, you have to specify the amount of data you can fit into a field. Some names or search queries are shorter than the actual, and this can lead to data loss.
3) Complex relational database systems can lead to these databases becoming "islands of information" where the information cannot be shared easily from one large system to another. Often, with big firms or institutions, you find relational databases grew in separate divisions differently. For example,

maybe the hospital billing department used one database while the hospital personnel department used a different database. Getting those databases to "talk" to each other can be a large, and expensive, undertaking, yet in a complex hospital system, all the databases need to be involved for good patient and employee care.

Key Value Database:

A key-value database, aka *key-value store*, associates a value (which can be anything from a number or simple string, to a complex object) with a key, which is used to keep track of the object. In its simplest form, a key-value store is like a dictionary/array/map object as it exists in most programming paradigms, but which is stored in a persistent way and managed by a Database Management System (DBMS). Key-value databases use compact, efficient index structures to be able to quickly and reliably locate a value by its key, making them ideal for systems that need to be able to find and retrieve data in constant time.

Uses of Key Value Database:

A key-value database is defined by the fact that it allows programs or users of programs to retrieve data by keys, which are essentially names, or identifiers, that point to some stored value. Because key-value databases are defined so simply, but can be extended and optimized in numerous ways, there is no global list of features, but there are a few common ones:

- Retrieving a value (if there is one) stored and associated with a given key
- Deleting the value (if there is one) stored and associated with a given key
- Setting, updating, and replacing the value (if there is one) associated with a given key
- Web applications may store user session details and preference in a key-value store. All the information is accessible via user key, and key-value stores lend themselves to fast reads and writes.
- Real-time recommendations and advertising are often powered by key-value stores because the stores can quickly access and present new recommendations or ads as a web visitor moves throughout a site.

- On the technical side, key-value stores are commonly used for in-memory data caching to speed up applications by minimizing reads and writes to slower disk-based systems. Hazel cast is an example of a technology that provides an in-memory key-value store for fast data retrieval.

Key-value databases can also have numerous other features, but at the very least support a system of operating on data in the above ways.

Advantages of key value database:

There are a few advantages that a key-value store provides over traditional row-column-based databases. Thanks to the simple data format that gives it its name, a key-value store can be very fast for read and write operations. And key-value stores are very flexible, a valued asset in modern programming as we generate more data without traditional structures.

Also, key-value stores do not require placeholders such as “null” for optional values, so they may have smaller storage requirements, and they often scale almost linearly with the number of nodes.

Document Databases:

A document database is a type of nonrelational database that is designed to store and query data as JSON-like documents. Document databases make it easier for developers to store and query data in a database by using the same document-model format they use in their application code. The flexible, semi structured, and hierarchical nature of documents and document databases allows them to evolve with applications’ needs. The document model works well with use cases such as catalogs, user profiles, and content management systems where each document is unique and evolves over time. Document databases enable flexible indexing, powerful ad hoc queries, and analytics over collections of documents.

Uses of Document Database:

A document database is a great choice for content management applications such as blogs and video platforms. With a document database, each entity that the application tracks can be stored as a single document. The document database is more intuitive for a developer to update an application as the requirements evolve.

Document databases are efficient and effective for storing catalog information. For example, in an e-commerce application, different products usually have different numbers of attributes. Managing thousands of attributes in relational databases is inefficient, and the reading performance is affected. Using a document database, each product’s attributes can be described in a single document for easy management and faster reading speed. Changing the attributes of one product won’t affect others.

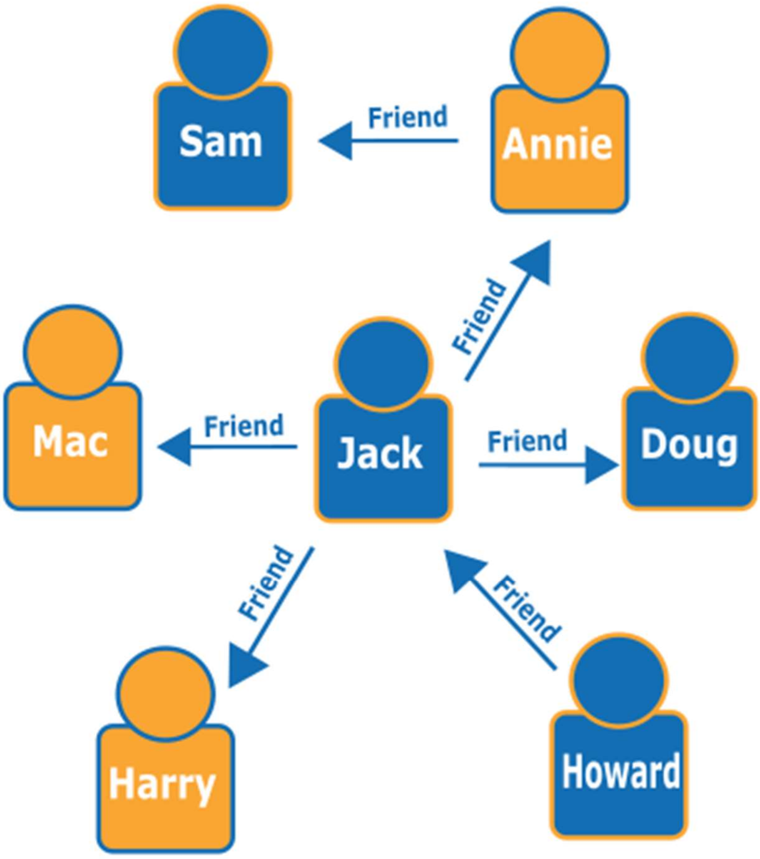
Advantages:

1) Ability to store dynamic data in any formats.
2) Ability to create persisted views from a base document and store the same for analysis.
3) Ability to store and process large data sets.
4) Schema free.
5) Ease of creation and maintenance.

Graph Databases

Graph databases are purpose-built to store and navigate relationships. Relationships are first-class citizens in graph databases, and most of the value of graph databases is derived from these relationships. Graph databases use nodes to store data entities, and edges to store relationships between entities. An edge always has a start node, end node, type, and direction, and an edge can describe parent-child relationships, actions, ownership, and the like. There is no limit to the number and kind of relationships a node can have.
A graph in a graph database can be traversed along specific edge types or across the entire graph. In graph databases, traversing the joins or relationships is very fast because the relationships between nodes are not calculated at query times but are persisted in the database. Graph databases have advantages for use cases such as social networking, recommendation engines, and fraud detection, when you need to create relationships between data and quickly query these relationships.

Given below is an example of graph databases that shows “Jack” and his friends and a friend of his friends related to each other.



Uses of Graph Database:

1) Graph databases are capable of sophisticated fraud prevention. With graph databases, you can use relationships to process financial and purchase transactions in near-real time.
2) Graph databases can also help you easily detect relationship patterns such as multiple people associated with a personal email address, or multiple people sharing the same IP address but residing in different physical addresses.
3) Graph databases are a good choice for recommendation applications. With graph databases, you can store in a graph relationship between information categories such as customer interests, friends, and purchase history.

Its advantages include Performance, better problem solving, update data in real-time use of recursive paths, AI infrastructure.