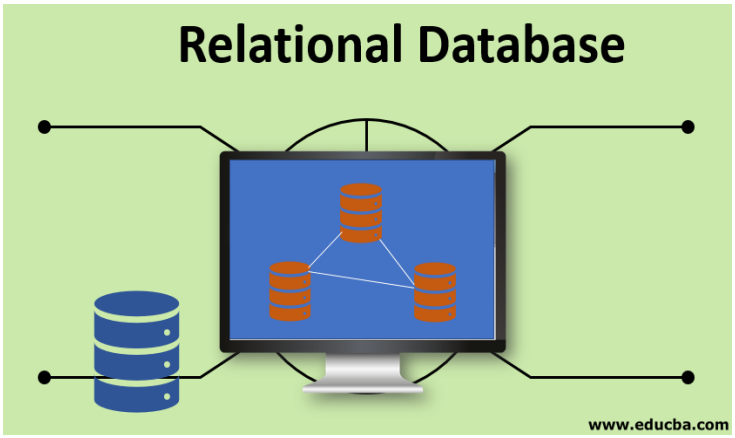
Name: Khalid L. Anta **Date:** 11/24/21

Year/Section: 3/BSCS/B

Discuss the following type of Databases

Relational

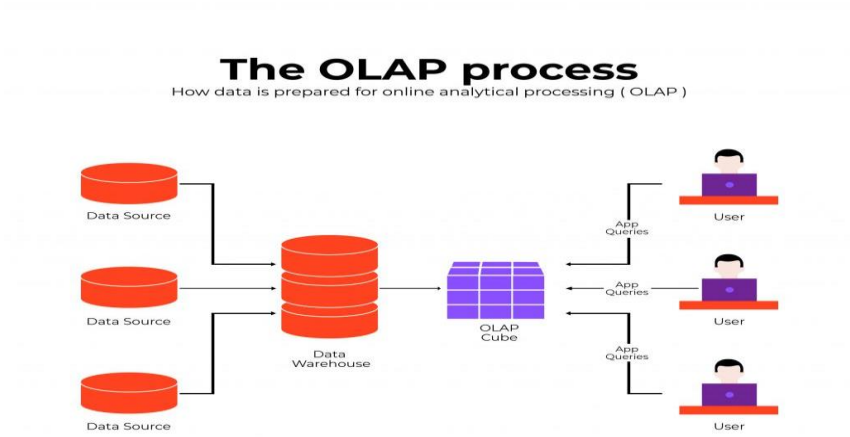


The relational model means that the logical data structures the data tables, views, and indexes are separate from the physical storage structures. This separation means that database administrators can manage physical data storage without affecting access to that data as a logical structure. For example, renaming a database file does not rename the tables stored within it.

The distinction between logical and physical also applies to database operations, which are clearly defined actions that enable applications to manipulate the data and structures of the database. Logical operations allow an application to specify the content it needs, and physical operations determine how that data should be accessed and then carries out the task.

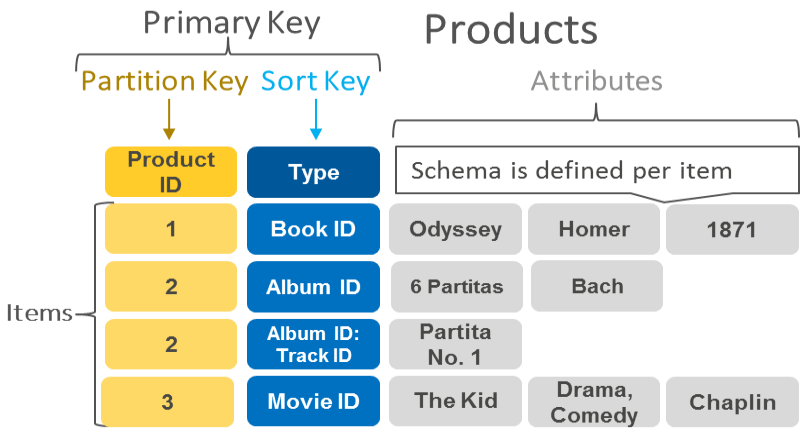
To ensure that data is always accurate and accessible, relational databases follow certain integrity rules. For example, an integrity rule can specify that duplicate rows are not allowed in a table in order to eliminate the potential for erroneous information entering the database.

Analytical (OLAP)



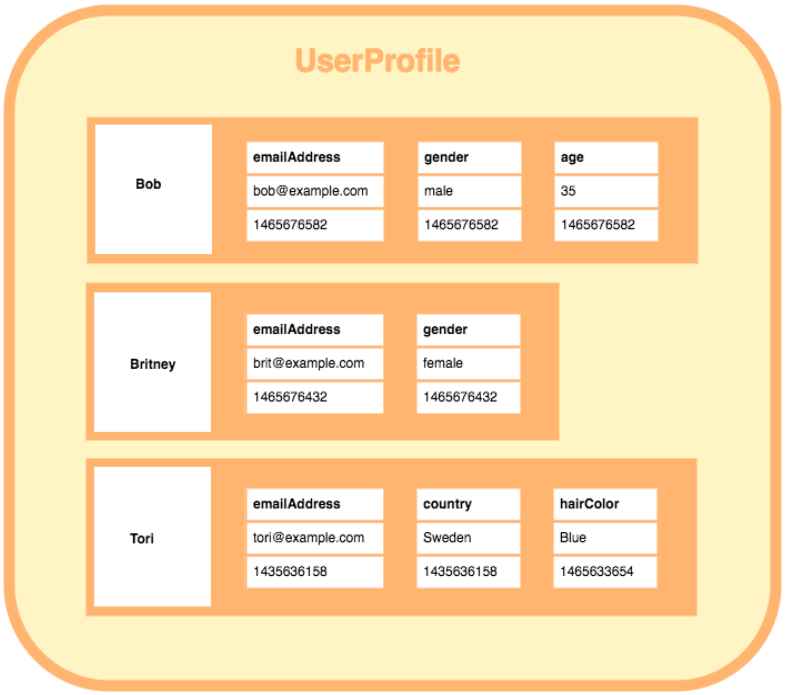
To facilitate this kind of analysis, data is collected from multiple data sources and stored in data warehouses then cleansed and organized into data cubes. Each OLAP cube contains data categorized by dimensions (such as customers, geographic sales region and time period) derived by dimensional tables in the data warehouses. Dimensions are then populated by members (such as customer names, countries and months) that are organized hierarchically. OLAP cubes are often pre-summarized across dimensions to drastically improve query time over relational databases.

Key-Value



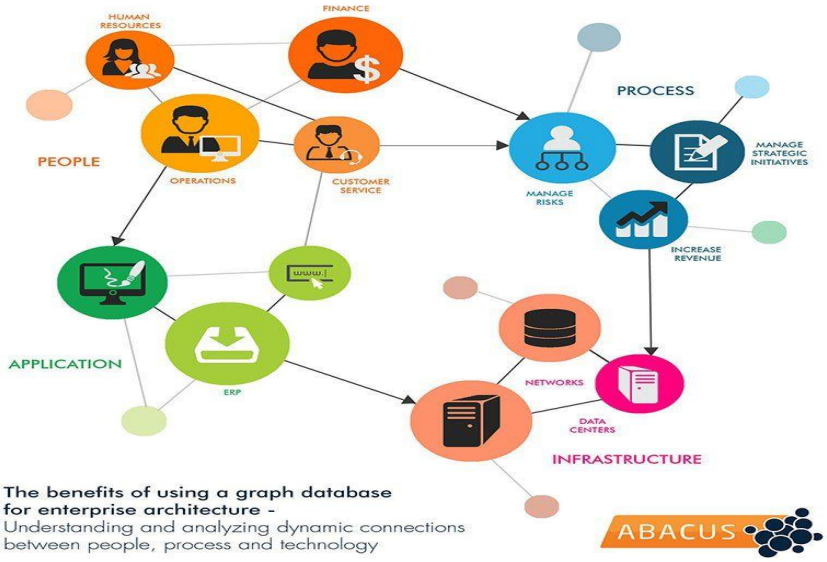
A key-value database is **a type of nonrelational database that uses** a simple key-value method to store data. A key-value database stores data as a collection of key-value pairs in which a key serves as a unique identifier. Both keys and values can be anything, ranging from simple objects to complex compound objects.

Colum-Family



Column family database is another aggregate oriented database. In **NoSQL** column family database we have a single key which is also known as row key and within that, we can store multiple column families where each column family is a combination of columns that fit together.
Column family as a whole is effectively your aggregate. We use row key and column family name to address a column family.
It is, however, one of the most complicated aggregate databases but the gain we have in terms of retrieval time of aggregate rows. When we are taking these aggregates into the memory, instead of spreading across a lot of individual records we store the whole thing in one database in one go.
The database is designed in such a way that it clearly knows what the aggregate boundaries are. This is very useful when we run this database on the cluster.
As we know that aggregate binds the data together, hence different aggregates are spread across different nodes in the cluster.

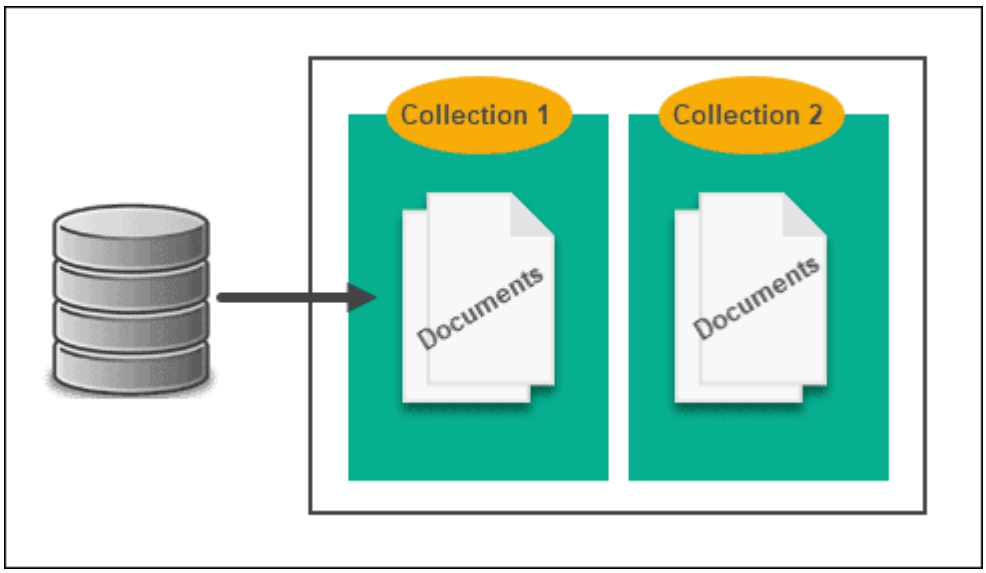
Graph



A graph database is a database designed to treat the relationships between data as equally important to the data itself. It is intended to hold data without constricting it to a pre-defined model. Instead, the data is stored like we first draw it out - showing how each individual entity connects with or is related to others.

There are no isolated pieces of information, but rich, connected domains all around us. Only a database that natively embraces relationships is able to store, process, and query connections efficiently. While other databases compute relationships at query time through expensive JOIN operations, a graph database stores connections alongside the data in the model.

Document



A document database is a type of no relational 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 catalogues, 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.