Name: Saguira T. Ukoman

Year/Course/Section; 2/BSCS/B

## Types of databases:

A **relational database** is a type of database that stores and provides access to data points that are related to one another. Relational databases are based on the relational model, an intuitive, straightforward way of representing data in tables. In a relational database, each row in the table is a record with a unique ID called the key. The columns of the table hold attributes of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.

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 simple yet powerful relational model is used by organizations of all types and sizes for a broad variety of information needs. Relational databases are used to track inventories, process ecommerce transactions, manage huge amounts of mission-critical customer information, and much more. A relational database can be considered for any information need in which data points relate to each other and must be managed in a secure, rules-based, consistent way.

**Analytical database** software specializes in big data management for business applications and services. Analytical databases are optimized to provide quick query response times and advanced analytics. They are also more scalable than traditional databases and often times are columnar databases that can efficiently write and read data to and from hard disk storage in order to speed up the time it takes to return a query. Analytical database features include column-based storage, in-memory loading of compressed data and the ability to search data through multiple attributes.

A **key–value database**, or **key–value store**, is a data storage paradigm designed for storing, retrieving, and managing associative arrays, and a data structure more commonly known today as a *dictionary* or *hash table*. Dictionaries contain a collection of *objects*, or *records*, which in turn have many different *fields* within them, each containing data. These records are stored and retrieved using a *key* that uniquely identifies the record, and is used to find the data within the database.

Key–value databases work in a very different fashion from the better known relational databases (RDB). RDBs predefine the data structure in the database as a series of tables containing fields with well defined data types. Exposing the data types to the database program allows it to apply a number of optimizations. In contrast, key–value systems treat the data as a single opaque collection, which may have different fields for every record. This offers considerable flexibility and more closely follows modern concepts like object-oriented programming. Because optional values are not represented by placeholders or input parameters, as in most RDBs, key–value databases often use far less memory to store the same database, which can lead to large performance gains in certain workloads.

A **column family** is a database object that contains columns of related data. It is a tuple (pair) that consists of a key–value pair, where the key is mapped to a value that is a set of columns. In analogy with relational databases, a column family is as a "table", each key-value pair being a "row". Each column is a tuple (triplet) consisting of a column name, a value, and a timestamp. In a relational database table, this data would be grouped together within a table with other non-related data.

Two types of column families exist:

- Standard column family: contains only columns
- Super column family: contains a map of super columns

In computing, a **graph database** (**GDB**) is a database that uses graph structures for semantic queries with nodes, edges, and properties to represent and store data.[1] A key concept of the system is the *graph* (or *edge* or *relationship*). The graph relates the data items in the store to a collection of nodes and edges, the edges representing the relationships between the nodes. The relationships allow data in the store to be linked together directly and, in many cases, retrieved with one operation. Graph databases hold the relationships between data as a priority. Querying relationships is fast because they are perpetually stored in the database. Relationships can be intuitively visualized using graph databases, making them useful for heavily inter-connected data.[2]

**Graph databases** are commonly referred to as a NoSQL database - implying that the approach to storing, querying and describing these data structures differs significantly from a traditional relational database. While the graph model explicitly lays out the dependencies between nodes of data, the relational model and other NoSQL database models link the data by implicit connections. In other words, relationships are a first-class citizen in a graph database and can be labelled, directed, and given properties. This is compared to relational approaches where these relationships are implied and must be reified at run-time. Graph databases are similar to 1970s network model databases in that both represent general graphs, but network-model databases operate at a lower level of abstraction[3] and lack easy traversal over a chain of edges.[4]

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, semistructured, 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.