Name: Kate Ashriah Dianne F. Hayao

Year/Course/Section: 3 BSCS-B

**TYPES OF DATABASE**

**Relational Database**

A relational database is a type of database that stores and provides access to data points that are related to one another. Relational [databases](https://www.oracle.com/ph/database/what-is-database/) 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**

The relational data model provided a standard way of representing and querying data that could be used by any application. From the beginning, developers recognized that the chief strength of the relational database model was in its use of tables, which were an intuitive, efficient, and flexible way to store and access structured information.

Over time, another strength of the relational model emerged as developers began to use structured query language (SQL) to write and query data in a database. For many years, SQL has been widely used as the language for database queries. Based on relational algebra, SQL provides an internally consistent mathematical language that makes it easier to improve the performance of all database queries. In comparison, other approaches must define individual queries.

**Rational Model Concepts**

1. **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student\_Rollno, NAME,etc.
2. **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:**Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
9. **Relation key** – Every row has one, two or multiple attributes, which is called relation key.
10. **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain.

**Analytical Database**

An [analytic database](http://exasol.com/) is a database management system that is optimized for business analytics applications and services. It is specifically designed to support [business intelligence](https://www.exasol.com/en/insights/big-data-and-analytics-glossary/operational-bi/) (BI) and analytic applications, typically as part of a [data warehouse](http://www.exasol.com/i-need-to/use-case/data-warehousing/) or data mart.

**Example of Analytical**

* Market data - Historical price and volume data for financial markets for testing trading strategies.
* Transactional data - Historical transactions that can include purchasing patterns for improved marketing.
* Sensor data - Historical data from sensors that monitor situations like the weather.
* Natural language data - Study of social media posts for research purposes.
* Process data - Study of processes to better understand logistics and find bottlenecks.
* Machine data - Software and hardware-generated data from products to improve efficiency.

**Key-value Database**

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.

**Column Family Database**

A **column family** is a database object that contains columns of related data. It is a [tuple](https://en.wikipedia.org/wiki/Tuple) (pair) that consists of a [key–value pair](https://en.wikipedia.org/wiki/Attribute%E2%80%93value_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](https://en.wikipedia.org/wiki/Tuple) ([triplet](https://en.wikipedia.org/wiki/Triplet_(disambiguation))) consisting of a column name, a value, and a [timestamp](https://en.wikipedia.org/wiki/Timestamp). In a [relational](https://en.wikipedia.org/wiki/Relational_database) [database table](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Standard_column_family): contains only columns
* [Super column family](https://en.wikipedia.org/wiki/Super_column_family): contains a map of [super columns](https://en.wikipedia.org/wiki/Super_column)

**Graph Databases**

A graph database is one that stores data in terms of entities and the relationships between entities. A variant on this theme are RDF (resource description framework) databases which store data in the format subject-predicate-object, which is known as a triple.

There are three types of graph database: true graph databases, triple stores and conventional databases that provide some graphical capabilities. Triple stores are often referred to as RDF databases. The difference between a true graph product and a triple store is that the former supports index free adjacency (which means you can traverse a graph without needing an index) and the latter doesn’t. The former are designed to support property graphs (graphs where properties may be assigned to either entities or their relationships, or both) but recently some triple stores have added this capability.

Both graph and RDF databases may be native products or they may be built on top of other database types. Most commonly, other database types are forms of [No SQL](https://www.bloorresearch.com/technology/nosql-databases/) database though there are some relational implementations.

## **Document-Oriented Database**

The central concept of a document-oriented database is the notion of a *document*. While each document-oriented database implementation differs on the details of this definition, in general, they all assume documents encapsulate and encode data (or information) in some standard format or encoding. Encodings in use include [XML](https://en.wikipedia.org/wiki/XML), [YAML](https://en.wikipedia.org/wiki/YAML), [JSON](https://en.wikipedia.org/wiki/JSON), as well as binary forms like [BSON](https://en.wikipedia.org/wiki/BSON).

Documents in a document store are roughly equivalent to the programming concept of an object. They are not required to adhere to a standard schema, nor will they have all the same sections, slots, parts or keys. Generally, programs using objects have many different types of objects, and those objects often have many optional fields. Every object, even those of the same class, can look very different. Document stores are similar in that they allow different types of documents in a single store, allow the fields within them to be optional, and often allow them to be encoded using different encoding systems.