**NAME**: VIÑAS, JUDAH PAULO LL.

**COURSE & SECTION**: BSIT-2F

**SUBJECT**: INFORMATION MANAGEMENT 1

1. Differentiate data and information.

* **DATA** it is a quantities, characters, or symbols on which operations are performed by a computer, being stored and transmitted in the form of electrical signals and recorded on magnetic, optical, or mechanical recording media; while **INFORMATION** is data that has been processed, organized, and given a context to make it meaningful and useful. It is the result of analyzing, interpreting, and presenting data in a structured manner.

1. What is database?

* A **DATABASE** is a collection of data that is organized, which is also called structured data. It can be accessed or stored in a computer system.

1. Types of database (explain each)

* In many cases, individuals find that they need different kinds of databases for different tasks. Below are some common types of databases:
* **Centralized Database**

A centralized database is one that operates entirely within a single location. Centralized databases are typically used by bigger organizations, such as a business or university. The database itself is located on a central computer or database system. Users can access the database through a computer network, but it is the central computer that runs and maintains the database.

* **Cloud Database**

A cloud database is one that runs over the Internet. The data is stored on a local hard drive or server, but the information is available online. This makes it easy to access your files from anywhere, as long as you have an Internet connection. To use a cloud database, users can either build one themselves or pay for a service to store their data for them. Encryption is an essential part of any cloud database, as all information needs to be protected as it is transmitted online.

* **Commercial Database**

A commercial database is any that is designed by a commercial business. Businesses develop feature-rich databases, which they then sell to their customers. Commercial databases can vary in terms of composition or the technology they use. The defining trait of commercial databases is having users pay to use them, unlike open source databases.

* **Distributed Database**

A distributed database is one that is spread out over multiple devices. Rather than having all information stored on a single device, like other databases on this list, distributed databases will operate across multiple machines, such as different computers within the same location or across a network. The benefits of a distributed database include increased speed, better reliability and ease of expansion.

* **End-user Database**

End-user is a term used in product development that refers to the person who uses the product. An end-user database is, therefore, a database that is primarily used by a single person. A good example of this type of database is a spreadsheet stored on your local computer.

* **Graph Database**

Graph databases are databases that focus equally on the data and the connections between them. In this database, data is not constricted to predefined models. Most other databases can find connections between data when you run a search. With a graph database, these connections are stored inside the database right alongside the original data. This makes for a more efficient and faster database when your primary goal is to manage the connections between your data.

* **NoSQL Database**

A [NoSQL database](https://www.indeed.com/career-advice/career-development/nosql-vs-sql) has a hierarchy similar to a file folder system and the data within it is unstructured, or non-relational. This lack of structure allows them to process larger amounts of data at speed and makes it easier to expand in the future. Cloud computing regularly makes use of NoSQL databases.

* **Object-oriented Database**

Object-oriented databases are ones in which data is represented as objects and classes. An object is an item, such as a name or phone number, while a class is a group of objects. Object-oriented databases are a type of relational database. Consider using an object-oriented database when you have a large amount of complex data that you want to process quickly.

* **Open-source Database**

An open-source database is designed for the public to use for free. Unlike commercial databases, users can download or sign up for open source databases without paying a fee. The term "open source" refers to a program in which users can see how it was written and constructed and are free to make their own changes to the program. Open-source databases are typically much cheaper than commercial databases, but they can also lack some of the more advanced features found in commercial databases.

* **Operational Database**

The purpose of an operational database is to allow users to modify data in real time. Operational databases are critical in business analytics and data warehousing. They can be set up either as relational databases or NoSQL, depending on needs. Conventional databases rely on batch processing, where commands are carried out in groups. Operational databases, on the other hand, allow you to add, edit and remove data at any moment.

* **Personal Database**

A personal database is one that is designed for a single person. It is typically stored on a personal computer and has a very simple design, consisting of only a few tables. Personal databases are not typically suitable for complex operations, large amounts of data or business operations.

* **Relational Database**

[Relational databases](https://www.indeed.com/career-advice/career-development/relational-database-schema) are the other major type of database, opposite of NoSQL. With a relational database, information is stored structured about other data. A good representation of a relational database would be the connection between a person shopping online and their shopping cart. Relational databases are often preferred when you are concerned about the integrity of your data, or when you're not particularly focused on scalability.

**Reference**: <https://www.indeed.com/career-advice/career-development/types-of-databases>

1. Evolution of file system data processing

* Before computers were developed to function on disk operating systems, each computer was built to run a single, proprietary application, which had complete and exclusive control of the entire machine. The application would write its persistent data directly to a disk, or drum, by sending commands directly to the disk controller. The application was responsible for managing the absolute locations of data on the disk, making sure that it was not overwriting already-existing data. Since only one application was running on the computer at any time, this task was not too difficult.

The advent of computer systems that could run more than one application required a mechanism to ensure that applications did not write over each other's data. Application developers addressed this problem by adopting a single standard for distinguishing disk sectors in use from those that were free by marking them accordingly. In time, these standards coalesced to become a disk operating system, which provided various services to the applications, including a file system for managing persistent storage. With the advent of a file system, applications no longer had to deal directly with the physical storage medium. Instead, they simply told the file system to write blocks of data to the disk and let the file system worry about how to do it. In addition, the file system allowed applications to create data hierarchies through an abstraction known as a directory. A directory could contain not only files but other directories, which in turn could contain their own files and directories, and so on.

The file system provided a single level of indirection between applications and the disk, and the result was that every application saw a file as a single contiguous stream of bytes on the disk even though the file system was actually storing the file in discontinuous sectors. The indirection freed the applications from having to track the absolute position of data on a storage device.

Today, virtually all system APIs for file input and output provide applications for writing information into a flat file. Applications see this file as a single stream of bytes that can grow as large as necessary until the disk is full. For a long time these APIs have been sufficient for applications to store their persistent information. Applications have made significant innovations in how they deal with a single stream of information to provide features like incremental "fast" saves.

However, in a world of component objects, storing data in a single flat file is no longer efficient. Just as file systems arose out of the need for multiple applications to share the same storage medium, so, now, do component objects require a system that allows them to share storage within the conceptual framework of a single file. Even though it is possible to store the separate objects using conventional flat-file storage, if one of the objects increases in size, or if you simply add another object, it becomes necessary to load the entire file into memory, insert the new object, and then save the whole file. This process can be extremely time-consuming.

The solution provided by COM is to implement a second level of indirection: a file system within a file. Flat-file storage requires that a large contiguous sequence of bytes on the disk be manipulated through a single file handle with a single seek pointer. By contrast, COM structured storage defines how to treat a single file system entity as a structured collection of two types of objects — storages and streams — that act like directories and files.

**Reference:** <https://learn.microsoft.com/en-us/windows/win32/stg/the-evolution-of-file-systems>