**Assignment -2**

Name: Srinath Kallu

ID: 900741822

**DIFFERENT TYPE OF DATABASES**

A database model is a type of data model that determines the logical structure of a database and fundamentally determines in which manner data  can be stored, organized, and manipulated. The most popular example of a database model is the relational model (or the SQL approximation of relational), which uses a table-based format.

**Now we go in detail about few concepts like:**

Why do we have different type of databases?

How many types of databases

How and why are they used?

Some of the types of databases are:

* Relational database
* Flat-file database
* Hierarchical  database
* Object-oriented database
* Object-relational database
* Navigational database
* Entity-relationship database

**Relational database:**

A relational database is a collective set of multiple data sets organized by tables, records and columns

It is called relational database because it is based on the “relational data model”. Each table contains one or more categories in columns. Each row contains a unique instance of data for the categories defined by the columns. This interconnection between the different tables and rows by using some unique keys makes a general database to relational database.

The relational database model exhibits the following structural characteristics:

* A relational database contains multiple tables.
* Each table stores data one specific subject.
* Fields contain data describing the subject of a table.
* Records are instances of the subject of a table.
* A special primary key field uniquely identifies each record in a table.

Relational databases scale well, but usually only when that scaling happens on a single server (“scale-up”). When the capacity of that single server is reached, you need to “scale-out” and distribute that load across multiple servers, moving into so-called distributed computing. This is when the complexity of relational databases starts to cause problems with their potential to scale. If you try to scale to hundreds or thousands of servers, the complexities become overwhelming. The characteristics that make relational databases so appealing are the very same that also drastically reduce their viability as platforms for large distributed systems.

In order to reduce this complexity data table is divided into different tables with one field in common with which we can access the data from different tables.

**Flat-file database:**

A flat database is a simple database system in which each database is represented as a single table in which all of the records are stored as single rows of data, which are separated by delimiters such as tabs or commas. The table is usually stored and physically represented as a simple text file.

Because of the limitations of flat databases, they are not unsuitable for most software applications in which there is a need to represent and store complex business relationships. However, some application developers still use flat files in order to reduce the cost and complexity of integrating a relational database.

Flat databases are also sometimes referred to as flat-file databases.

Unlike relational databases, flat databases cannot represent complex relationships between entities. They also have no way of enforcing constraints between data. For instance in an application used by a commercial bank, it is a good idea to ensure that, at the time of creation, a new account must be linked to an existing customer. In a relational database this is easily enforced using the concept of foreign keys to ensure that customer IDs are filled in while creating an account, and also that said customer IDs already exist in another table. This is not possible with flat databases, which means that such a constraint has to be enforced by other means, such a through application code logic.  
  
Another limitation of flat databases vis-a-vis relational databases is the former's lack of query and indexing capability. SQL queries cannot be written in flat databases because the data is not relational, and indexes cannot be created because the data is all lumped together in one table. Data in a flat database is typically only readable by and useful to the software application associated with the database.

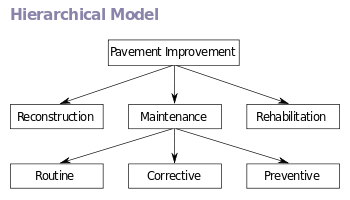
Flat databases are, or should only be, created for small, simple databases that will never grow large enough for the limitations outlined above to really become a problem. Some real-life examples of flat databases are contact lists in a mobile phone and the storage of a high-scores list in a simple video game. In such cases, there would be little point and no justifiable expense in integrating a complex relational database engine into the computing platform because a simple flat database will do nicely.

**Hierarchical database:**

A hierarchical database is a design that uses a one-to-many relationship for data elements. Hierarchical database models use a tree structure that links a number of disparate elements to one "owner," or "parent," primary record.

The idea behind hierarchical database models is useful for a certain type of data storage, but it is not extremely versatile. Its limitations mean that it is confined to some very specific uses. For example, where each individual person in a company may report to a given department, the department can be used as a parent record and the individual employees will represent secondary records, each of which links back to that one parent record in a hierarchical structure.

Hierarchical databases were popular in early database design, in the era of mainframe computers. While some IBM and Microsoft models are still in use, many other types of business databases use more flexible models to accommodate more sophisticated types of data management. Hierarchical models make the most sense where the primary focus of information gathering is on a concrete hierarchy such as a list of business departments, assets or people that will all be associated with specific higher-level primary data elements.



**Object-oriented database:**

An object-oriented database is a database that subscribes to a model with information represented by objects. Object-oriented databases are a niche offering in the relational database management system (RDBMS) field and are not as successful or well-known as mainstream database engines.

As the name implies, the main feature of object-oriented databases is allowing the definition of objects, which are different from normal database objects. Objects, in an object-oriented database, reference the ability to develop a product, then define and name it. The object can then be referenced, or called later, as a unit without having to go into its complexities. This is very similar to objects used in object-oriented programming.  
  
A real-life parallel to objects is a car engine. It is composed of several parts: the main cylinder block, the exhaust system, intake manifold and so on. Each of these is a standalone component; but when machined and bolted into one object, they are now collectively referred to as an engine. Similarly, when programming one can define several components, such as a vertical line intersecting a perpendicular horizontal line while both lines have a graded measurement. This object can then be collectively labeled a graph. When utilizing the ability to plot components, there is no need to first define a graph; but rather the instance of the created graph can be called.  
  
Examples of object-oriented database engines include db4o, Smalltalk and Cache.

**Object-relational database:**

An object-relational database (ORD) is a database management system (DBMS) that's composed of both a relational database (RDBMS) and an object-oriented database (OODBMS). ORD supports the basic components of any object-oriented database model in its schemas and the query language used, such as objects, classes and inheritance.

An object-relational database may also be known as object relational database management systems (ORDBMS).

ORD is said to be the middleman between relational and object-oriented databases because it contains aspects and characteristics from both models. In ORD, the basic approach is based on RDB, since the data is stored in a traditional database and manipulated and accessed using queries written in a query language like SQL. However, ORD also showcases an object-oriented characteristic in that the database is considered an object store, usually for software that is written in an object-oriented programming language. Here, APIs are used to store and access the data as objects. 

One of ORD’s aims is to bridge the gap between conceptual data modeling techniques for relational and object-oriented databases like the entity-relationship diagram (ERD) and object-relational mapping (ORM). It also aims to connect the divide between relational databases and the object-oriented modeling techniques that are usually used in programming languages like Java, C# and C++.  
  
Traditional RDBMS products concentrate on the efficient organization of data that is derived from a limited set of data-types. On the other hand, an ORDBMS has a feature that allows developers to build and innovate their own data types and methods, which can be applied to the DBMS. With this, ORDBMS intends to allow developers to increase the abstraction with which they view the problem area.

**Entity-relationship database:**

An entity-relationship model (ERM) is a theoretical and conceptual way of showing data relationships in software development. ERM is a database modeling technique that generates an abstract diagram or visual representation of a system’s data that can be helpful in designing a relational database. These diagrams are known as entity-relationship diagrams, ER diagrams or ERDs.

The first step in information system design dictates that the requirements analysis models illustrate the type of data or information that needs to be collected. The data modeling method may be used to illustrate a specific interest area’s ontology. Like the relational model, abstract data is converted to a logical data model when the design of an information system is built on a database. Likewise, this is converted to a physical model when it is physically designed.

The building blocks of an ERD are entities, relationships and attributes. Entities have entity types, which are known as instances of the corresponding entities. Each entity type can exist independently of another; for example, the entity "vehicle" can have the entity types "car" and "bus." Relationship is the property that links the entity types together. For example, the entity type husband is related to the entity type wife by a relationship known as "is-married-to." Attributes are properties that belong to the entity types as well as to the relationships.

There is a number of ER diagramming tools available on the market. The most common ones are MySQL Workbench and OpenModelSphere.