TABLE OF CONTENTS

regent@111

[Data vs. Information 2](#_Toc436596601)

[Comparison chart 2](#_Toc436596602)

[Data 2](#_Toc436596603)

[Information 2](#_Toc436596604)

[Database 2](#_Toc436596605)

[Database Model 2](#_Toc436596606)

[Flat File Model 2](#_Toc436596607)

[Hierarchical Model 4](#_Toc436596608)

[Network Model 6](#_Toc436596609)

[Relational Model 7](#_Toc436596610)

[Object Relational Model 9](#_Toc436596611)

[Codd Rules 10](#_Toc436596612)

# Data vs. Information

Data are the facts or details from which information is derived. Individual pieces of data are rarely useful alone. For data to become information, data needs to be put into context.

## Comparison chart

|  | Data | Information |
| --- | --- | --- |
| Meaning | Data is raw, unorganized facts that need to be processed. Data can be something simple and seemingly random and useless until it is organized. | When data is processed, organized, structured or presented in a given context so as to make it useful, it is called information. |
| Example | Each student's test score is one piece of data. | The average score of a class or of the entire school is information that can be derived from the given data. |

# Database

Database is a collection of inter-related information, which is stored in DBMS.

A database management system (DBMS) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS’s designed to allow the definition, creation, querying, update, and administration of databases

# Database Model

A Database model defines the logical design of data. It also defines how the data is structured. The model describes the relationships between different parts of the data. In history of database design, five models have been in use.

* Flat files Model
* Hierarchical Model
* Network Model
* Relational Model
* Object relational Model

## 

## Flat File Model

A flat file database uses files to store simple information on a computer. A UNIX or Linux operating system runs on a series of flat file databases. A Windows computer also uses flat file databases to store information which is used every day. A Macintosh computer also does this. But a flat file database is not relational, and cannot satisfy Edgar F. Codd's 12 rules that define the relational nature of databases.

There are distinct advantages and also disadvantages to the flat file database.

Flat Files can be used for Storing Configuration Simply .The Windows Registry, however complex, is not relational. It is a flat file which expands and contracts as software and hardware get added and removed. It keeps track of user data, stores logs and helps find information very quickly. Passwords are stored by UNIX and Linux in a flat file database along with configuration information which is easily editable, and which makes the system function. Any changes to these UNIX configuration files cause instant effects which can be dramatic. A lot of this is what makes UNIX so popular because of its ease of configuration.

Flat File Database Advantages

A flat file database is a one record per line text file, a binary file (or a combination of both text and binary) file which separates columns by using "delimiters" which separate the fields in a standard recognizable way.

This helps in searching through records for information. Records can also be delimited by fixed length. If a record is too short, some form of field-padding can be used so that the length of the record is the same as the next. This allows for a uniform byte-length from one field to the next.

Configuration files are often stored in XML (Extended Markup Language), which is a flat file database with some quite specific delimiters, such as "greater than" and "less than" (<>) symbols. These XML files are often used to build specific modules on a website, when the standard build follows a very specific pattern that is configured by a single value stored in an XML file, or a series of related values.

Flat File Database Disadvantages

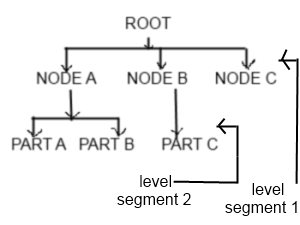
Databases accessed on a network share are useful for access by many people who are looking for information.

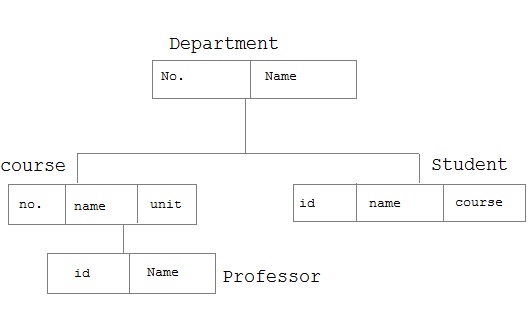
Flat file databases are not usually accessed like this since they belong with offline entities and form the machinery of operating systems and local devices. Also, there are no transactions in a flat file database, so it is limited in what it can actually do as a database entity. So a flat file database is disadvantageous to a network user, who is accessing a multi-access, multi-tasking relational online database which can be viewed from many different aspects. Duplication of data is very high.

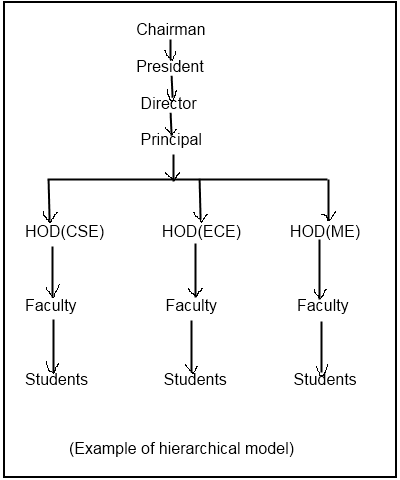
## Hierarchical Model

In this model each entity has only one parent but can have several children. At the top of hierarchy there is only one entity which is called Root.

This structure contains levels or segments. The top layer is known as the parent of the segment .Root segment is the parent of level 1 segment. This model defines a set of one to many relationships between parent and children segments







ADVANTAGES OF HIERARCHICAL MODEL:-

1) It provides simplicity and it is easy to design.

2) It also provides data integrity. Since, it is based on parent-child relationship. So, there is always a link between these segments.

DRAWBACKS OF HIERARCHICAL MODEL:-

1) There exist operational anomalies in this model.

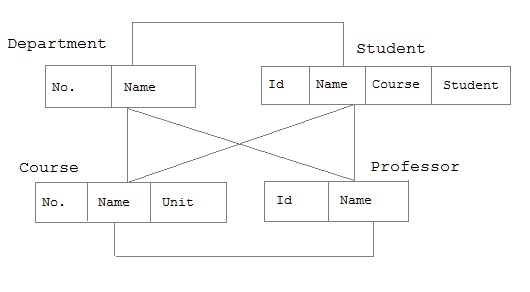
2) Implementation Complexity

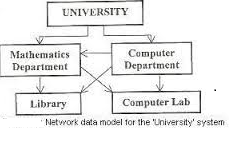
3) It is easy to design but very difficult to implement.

4) Database Management System-If you make any change in the database structure then you need to make all the necessary changes in all the application program that access the database.

## Network Model

In the network model, entities are organized in a graph, in which some entities can be accessed through several paths. In network model, the relationship can be defined using sets. A set consists of record types or entities. It has an owner record type and child record type. An owner record type can be part of only one set. But a child can have owners from multiple sets. It allows us to define one to many and many to many relationships.





ADVANTAGES OF NETWORK MODEL-

The major advantage of network model are-

1.) Conceptual simplicity-Just like the hierarchical model, the network model is also conceptually simple and easy to design.

2.) Capability to handle more relationship types-The network model can handle the one to many and many to many relationships which is real help in modeling the real life situations.

3.) Ease of data access-The data access is easier and flexible than the hierarchical model.

4.) Data integrity- The network model does not allow a member to exist without an owner.

5.) Data independence- The network model is better than the hierarchical model in isolating the programs from the complex physical storage details.

6.) Database standards

DIS-ADVANTAGE OF NETWORK MODEL-

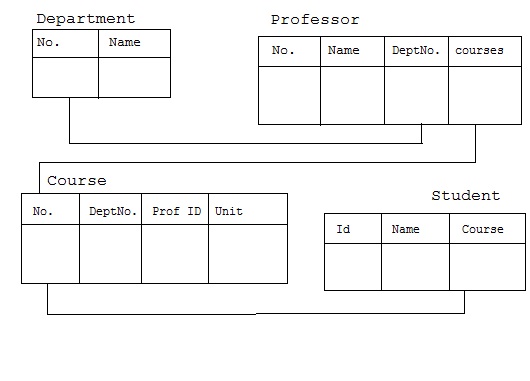
1.) System complexity- All the records are maintained using pointers and hence the whole database structure becomes very complex.

2.) Operational Anomalies- The insertion, deletion and updating operations of any record require large number of pointers adjustments.

3.) Absence of structural independence-structural changes to the database is very difficult.

## Relational Model

In this model, data is organized in two-dimensional tables called **relations**. The tables or relation are related to each other.



Relational model stores the data in the form of table. It consists of three major components.

1.) The set of relations and set of domains that defines the way data can be represented.

2.) Integrity rules that defines the procedure to protect the data.

3.) The operations that can be performed on the table.

CHARACTERISTICS-

Characteristics of relational database are:-

1.) All data is conceptually represented as an orderly arrangement of data into rows and column called a relation or table.

2.) All the values are scalar i.e. at any given row or column position there is one and only one value.

3.) All operations are performed on the entire relation and the result is an entire relation.

BASIC TERMINOLOGY OF RELATIONAL DATABASE-

1.) TUPLE OF A RELATION-Each row in a table is known as tuple.

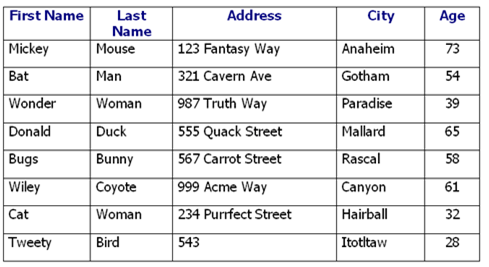
2.) CARDINALITY OF A RELATION-  It is the no of tuples in the relation.

3.) DEGREE OF A RELATION-No of columns is known as degree of a relation.

4.) DOMAIN OF A RELATION-It defines the kind of data represented by the attribute. It is the set of all possible values that an attribute may contain.

5.) BODY OF THE RELATION-It consists of an unordered set of 0 or more tuples.

EXAMPLE:- Consider a table



1.) TUPLE OF A RELATION- 8

2.) CARDINALITY OF A RELATION- 8

3.) DEGREE OF A RELATION- 5

4.) DOMAIN OF A RELATION- All the data entered in the table is the domain of a relation.

KEYS OF A RELATION-

1.) PRIMARY KEY- It is the key that uniquely identifies a record. It doesn't have null values.

2.) FOREIGN KEY- It refers to the primary key of some other table. It permits only those values which appear in the primary key of the table to which it refers.

3) Super Key – A key used to identify uniquely the records in the table is called super key.

4) Candidate Keys – one or more keys which can uniquely identify records in the table is called Candidate keys. One or more Candidate keys will act as Primary key.

5) Composite Key - One or more keys which act as Primary key is called as Composite Key.

6) Secondary Key - A key used to retrieve the values of rows as an additional condition in combination with Primary key.

## Object Relational Model

An ORDBMS supports an extended form of SQL called SQL3 that is still in the development stages. The extensions are needed because ORDBMSs have to support ADT's.

The ORDBMS has the relational model in it because the data is stored in the form of tables having rows and columns and SQL is used as the query language and the result of a query is also table or tuples (rows).

Characteristics of an ORDBMS:

1. Base data type extension,
2. Support complex objects,
3. Inheritance, and
4. Rule Systems.

Users define data types:

1. Object-Relational Database Management Systems (ORDBMSs) allow users to define data types, functions and operators. As a result, the functionality of the ORDBMSs increases along with their performance.

An example schema of a student relation which ORDBMS supports

1. STUDENT(fname,lname,ID,sex,major,address,dname,location,picture)
2. Notice: extra attributes "location" and "picture" which are not present in the traditional EMPLOYEE relation of RDBMS. The data type of "location" is "geographic point" and "picture" is "image".

**The differences between the two approaches**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **RDBMS** | **ORDBMS** |
| Defining standard | SQL2 | SQL3 (in process) |
| Support for object-oriented features | Does not support;  It is difficult to map program object to the database | Limited support; mostly to new data type |
| Usage | Easy to use | Easy to use except for some extensions |
| Support for complex relationships | Does not support abstract data types | Supports Abstract data types and complex relationships |
| Performance | Very good performance | Expected to perform very well |

# Codd Rules

Dr. Edgar Frank Codd (August 19, 1923 – April 18, 2003) was an computer scientist, while working for IBM he invented the relational model for database management ( theoretical basis for relational databases). Codd proposed thirteen rules (numbered zero to twelve) and said that if a Database Management System meets these rules, it can be called as a Relational Database Management System. These rules are called as Codd's12 rules. Hardly any commercial product follows all.

**Rule 1:** The information rule: All information in the database is to be represented in one and only one way, namely by values in column positions within rows of tables.

**Rule 2:** The guaranteed access rule: All data must be accessible. This rule is essentially a restatement of the fundamental requirement for primary keys. It says that every individual scalar value in the database must be logically addressable by specifying the name of the containing table, the name of the containing column and the primary key value of the containing row.

**Rule 3:** Systematic treatment of null values: The DBMS must allow each field to remain null (or empty). Specifically, it must support a representation of "missing information and inapplicable information" that is systematic, distinct from all regular values (for example, "distinct from zero or any other number", in the case of numeric values), and independent of data type. It is also implied that such representations must be manipulated by the DBMS in a systematic way.

**Rule 4:** Active online catalog based on the relational model: The system must support an online, inline, relational catalog that is accessible to authorized users by means of their regular query language. That is, users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data.

**Rule 5:** The comprehensive data sub language rule: The system must support at least one relational language that

1. Has a linear syntax

2. Can be used both interactively and within application programs,

3. Supports data definition operations (including view definitions), data manipulation operations (update as well as retrieval), security and integrity constraints, and transaction management operations (begin, commit, and rollback).

**Rule 6:** The view updating rule: All views those can be updated theoretically, must be updated by the system.

**Rule 7:** High-level insert, update, and delete: The system must support set-at-a-time insert, update, and delete operators. This means that data can be retrieved from a relational database in sets constructed of data from multiple rows and/or multiple tables. This rule states that insert, update, and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

**Rule 8:** Physical data independence: Changes to the physical level (how the data is stored, whether in arrays or linked lists etc.) must not require a change to an application based on the structure.

**Rule 9:** Logical data independence: Changes to the logical level (tables, columns, rows, and so on) must not require a change to an application based on the structure. Logical data independence is more difficult to achieve than physical data independence.

**Rule 10:** Integrity independence: Integrity constraints must be specified separately from application programs and stored in the catalog. It must be possible to change such constraints as and when appropriate without unnecessarily affecting existing applications.

**Rule 11:** Distribution independence: The distribution of portions of the database to various locations should be invisible to users of the database. Existing applications should continue to operate successfully:

1. When a distributed version of the DBMS is first introduced; and

2. When existing distributed data are redistributed around the system.

**Rule 12**: The non subversion rule: If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert the system, for example, bypassing a relational security or integrity constraint

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*