Database Management System

Objectives

At the end of this module, you will be able to

- DBMS Fundamental
- DBMS Concepts
- DBMS Architecture
- Abstraction & Three-schema architecture
- Basic modules of DBMS
- DBMS Users
- Data Modeling

Data Base Management System

- **Database** is a collection of related data and data is a collection of facts and figures that can be processed to produce information.
- Mostly data represents recordable facts. Data aids in producing information, which is based on facts.
- For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks.
- **A database management system** stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.
- Collection of interrelated data
- Set of programs to access the data
- DBMS contains information about a particular enterprise
- DBMS provides an environment that is both convenient and efficient to use.

- **Database Applications:**
  Banking: all transactions , **Airlines:** reservations, schedules, **Universities:** registration, grades, **Sales:** customers, products, purchases, Manufacturing: production, inventory, orders, supply chain, **Human resources:** employee records, salaries, tax deductions
- Databases touch all aspects of our lives

DBMS - Characteristics

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics:

- **Real-world entity:** A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.

- **Relation-based tables:** DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.

- **Isolation of data and application:** A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.

DBMS - Characteristics

- **Less redundancy**: DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.

- **Consistency:** Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.

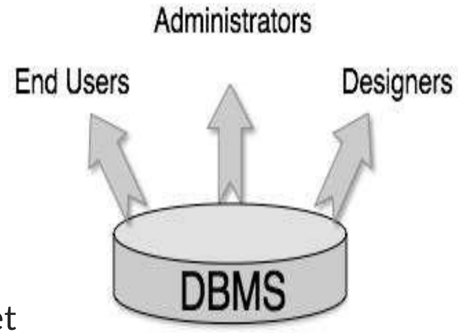
- **Query Language:** DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.

DBMS - Characteristics

- **ACID Properties:** DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.
- **Multiuser and Concurrent Access:** DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.
- **Multiple views:** DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.
- **Security:** Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

Users

- A typical DBMS has users with different rights and permissions who use it for different purposes.
- **Some users retrieve data and some back it up. The users of a DBMS can be broadly categorized as follows:**
- **Administrators:** Administrators maintain the DBMS and are responsible for administrating the database. They are responsible to look after its usage and by whom it should be used. They create access profiles for users and apply limitations to maintain isolation and force security. Administrators also look after DBMS resources like system license, required tools, and other software and hardware related maintenance.
- **Designers:** Designers are the group of people who actually work on the designing part of the database. They keep a close watch on what data should be kept and in what format. They identify and design the whole set of entities, relations, constraints, and views.
- **End Users:** End users are those who actually reap the benefits of having a DBMS. End users can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts.



ARCHITECTURE

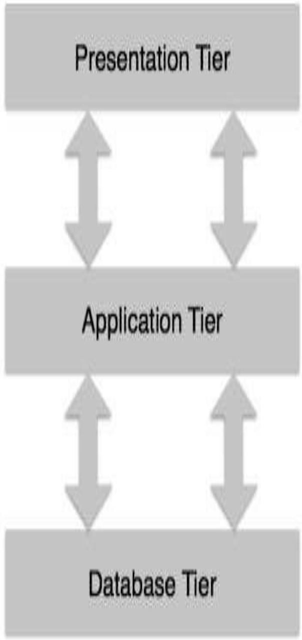
- The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed, or replaced.

- In **1-tier architecture**, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

- If the architecture of DBMS **is 2-tier**, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

3-tier Architecture

- A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.
- **Database (Data) Tier:** At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
- **Application (Middle) Tier:** At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
- **User (Presentation) Tier:** End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.
- Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

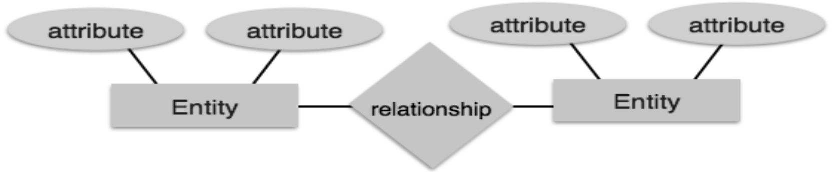


DATA MODELS - Entity-Relationship Model

- Data models define how the logical structure of a database is modeled.
- Data Models are fundamental entities to introduce abstraction in a DBMS.
- Data models define how data is connected to each other and how they are processed and stored inside the system.
- The very first data model could be flat data-models, where all the data used are to be kept in the same plane. Earlier data models were not so scientific, hence they were prone to introduce lots of duplication and update anomalies.
- **Entity-Relationship (ER)** Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship attributes, and constraints.
- **ER Model** is best used for the conceptual design of a database. ER Model is based on:
- **Entities and their attributes.**
- **Relationships among entities.** These concepts are explained below.

DATA MODELS - Entity-Relationship Model

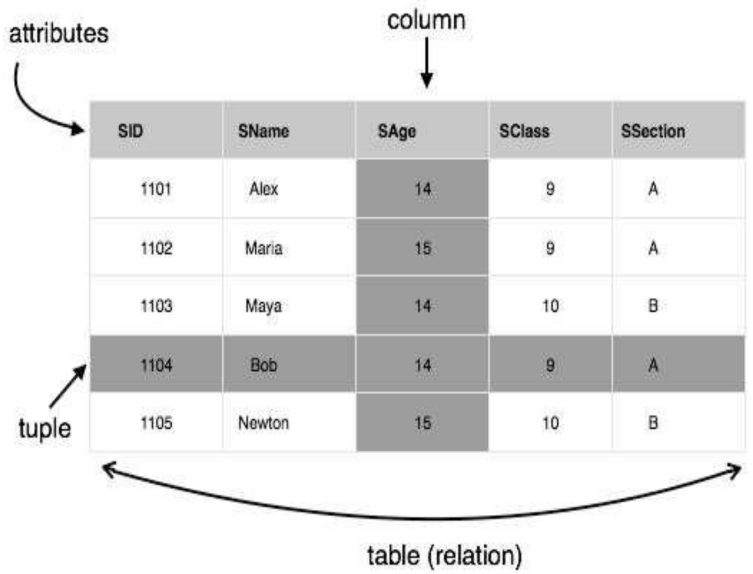
- **ER Model** is best used for the conceptual design of a database. ER Model is based on:
  - **Entities and their attributes.**
  - **Relationships among entities.** These concepts are explained below.



- **Entity :** An entity in an ER Model is a real-world entity having properties called
- **Attributes :** Every attribute is defined by its set of values called domain.
- For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.
- **Relationship**
- The logical association among entities is called relationship. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.
- **Mapping cardinalities:** one to one, one to many, many to one , many to many

Relational Model

- The most popular data model in DBMS is the Relational Model. It is more scientific a model than others.
- This model is based on first-order predicate logic and defines a table as an n-ary relation.
- The main highlights of this model are:
- Data is stored in tables called relations.
- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in a relation contains a unique value.
- Each column in a relation contains values from a same domain.



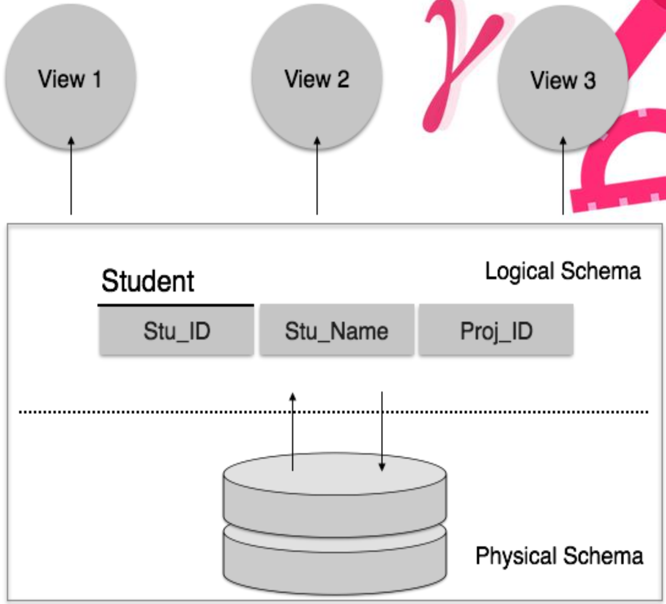
SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

DATA SCHEMAS

- A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.
- A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.

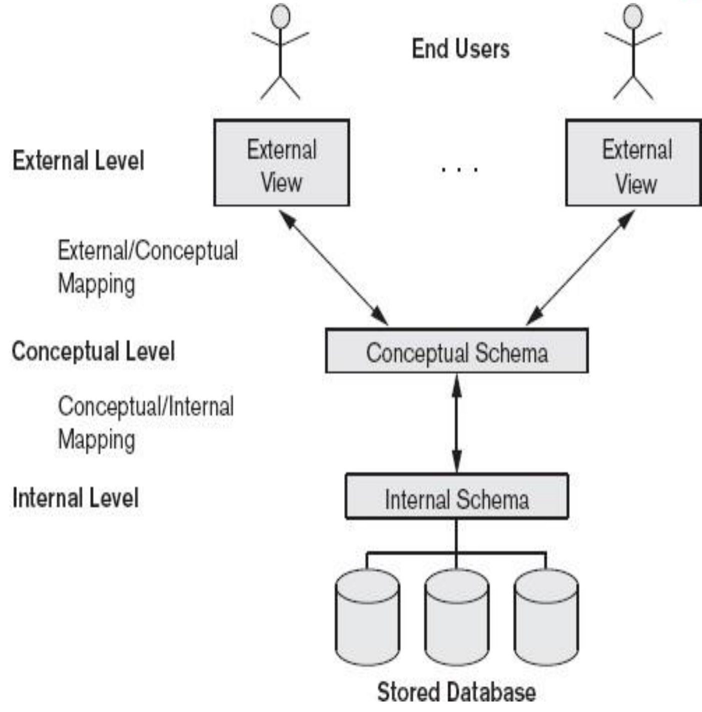
A database schema can be divided broadly into two categories:

- **Physical Database Schema:** This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- **Logical Database Schema:** This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.



THREE-SCHEMA ARCHITECTURE

- **External** or **view** level
  - Describes part of the database of interest to a particular user group
- **Conceptual** level
  - Describes structure of the whole database for the complete community of users
- **Internal** level
  - Describes physical storage structure of the database



ER MODEL – BASIC CONCEPTS

- The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

Entity

- An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.
- An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Attributes

- Entities are represented by means of their properties called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.
- There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

ER MODEL – BASIC CONCEPTS

Types of Attributes

- **Simple attribute:** Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
- **Composite attribute:** Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.
- **Derived attribute:** Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.
- **Single-value attribute:** Single-value attributes contain single value.

For example: Social_Security_Number.

- **Multi-value attribute:** Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

These attribute types can come together in a way like:

1. simple single-valued attributes
2. composite single-valued attributes
3. simple multi-valued attributes
4. composite multi-valued attributes

ER MODEL – BASIC CONCEPTS

Entity-Set and Keys

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, the roll_number of a student makes him/her identifiable among students.

- **Super Key:** A set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key:** A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- **Primary Key:** A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

ER MODEL – Relationship

- The association among entities is called a relationship. For example, an employee **works_at** a department, a student **enrolls** in a course. Here, Works_at and Enrolls are called relationships.

Relationship Set

- A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes.**

Degree of Relationship

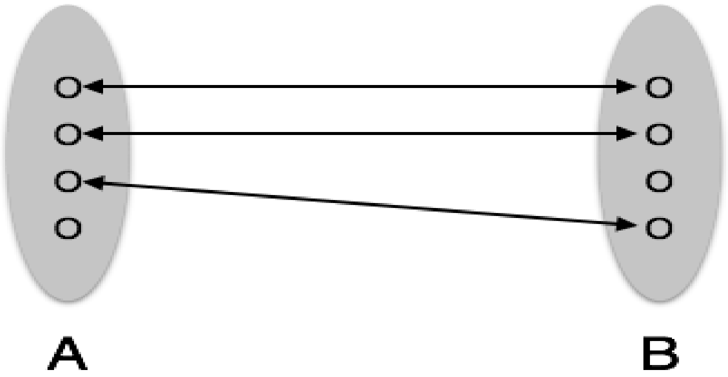
- The number of participating entities in a relationship defines the degree of the relationship.
  - Binary = degree 2
  - Ternary = degree 3
  - n-ary = degree

ER MODEL – Relationship

Mapping Cardinalities

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

- **One-to-one:** One entity from entity set A can be associated with at most one entity of entity set B and vice versa.

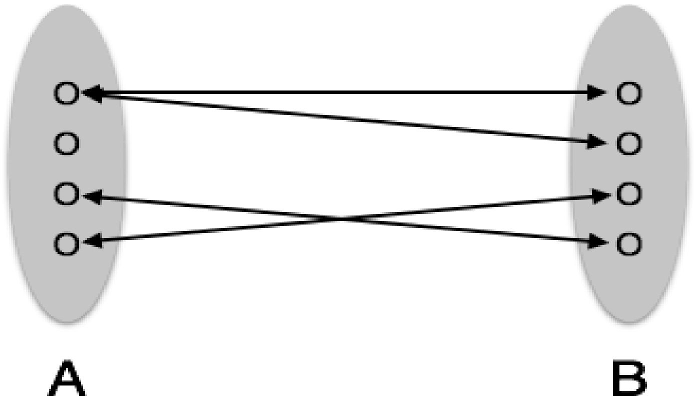


ER MODEL – Relationship

Mapping Cardinalities

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

- **One-to-many:** One entity from entity set A can be associated with more than one entities of entity set B, however an entity from entity set B can be associated with at most one entity.

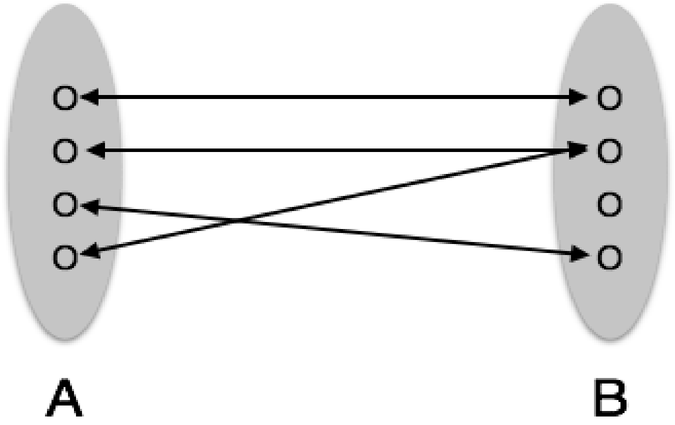


ER MODEL – Relationship

Mapping Cardinalities

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

- **Many-to-one:** More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.

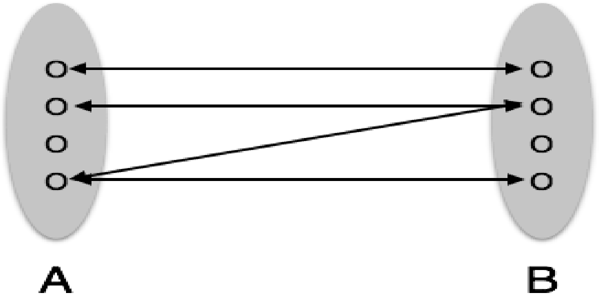


ER MODEL – Relationship

Mapping Cardinalities

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

- **Many-to-many:** One entity from A can be associated with more than one entity from B and vice versa.

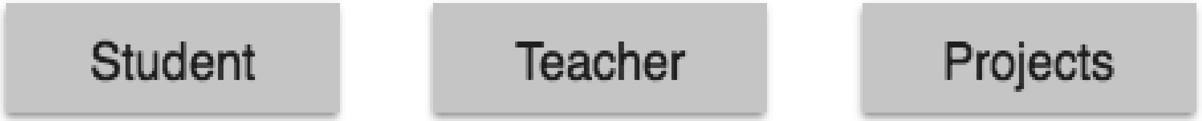


ER DIAGRAM REPRESENTATION

- Let us now learn how the ER Model is represented by means of an ER diagram.
- Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an ER diagram.

Entity

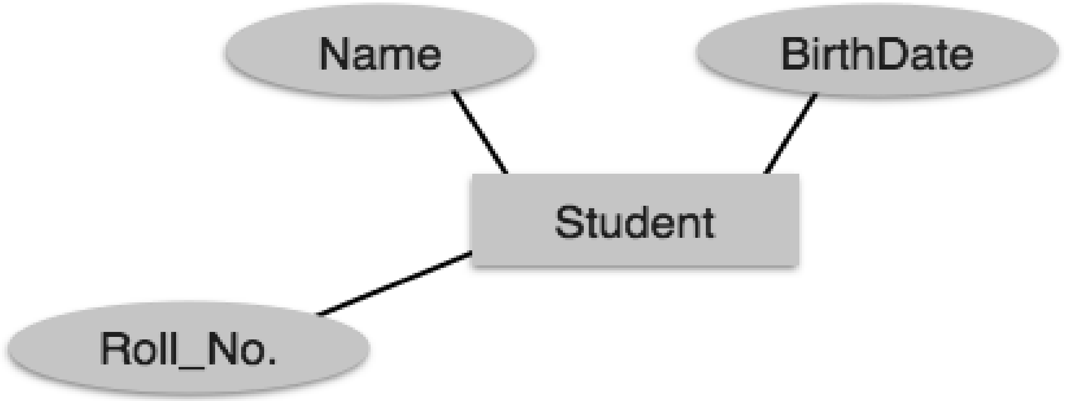
- Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.



ER DIAGRAM REPRESENTATION

Attributes

- Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).

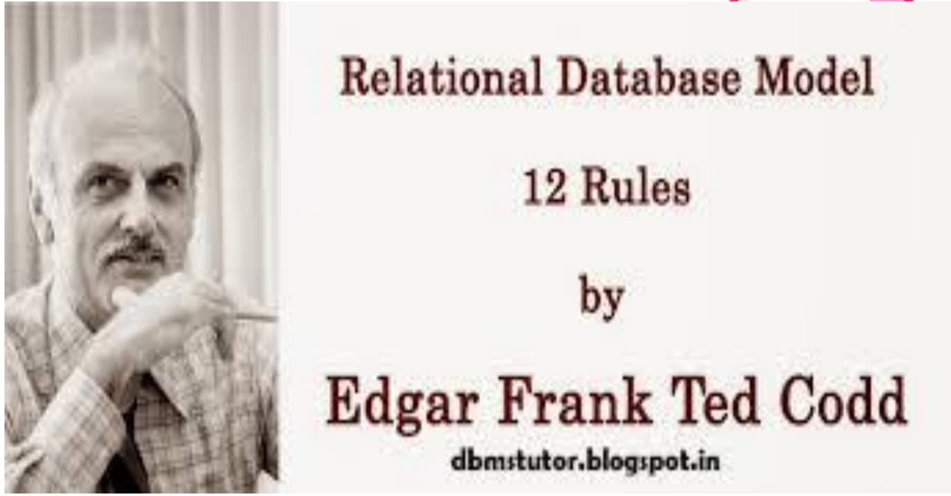


CODD'S 12 RULES

- **Dr Edgar F. Codd**, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.

- These rules can be applied on any database system that manages stored data using only its relational capabilities.

- This is a foundation rule, which acts as a base for all the other rules.



CODD'S 12 RULES

Rule 1 - Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2 - Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3 - Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following: data is missing, data is not known, or data is not applicable.

Rule 4 - Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as **data dictionary**, which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5 - Comprehensive Data Sub- Language Rule

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

Rule 6 - View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the system.

CODD'S 12 RULES

Rule 7 – High -Level Insert, Update & Delete Rule

A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, it must also support union, intersection and minus operations to yield sets of data records.

Rule 8 - Physical Data Independence

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

Rule 9 – Logical Data Independence

The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rule to apply.

Rule 10 - Integrity Independence Rule

A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11 - Distribution Independence Rule

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

Rule 12 - Non-Subversion Rule

If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

Questions

Survey

Your feedback is important to us, be it a compliment, a suggestion or a complaint. It helps us to make the course better!

Please spare few minutes to take the survey after the webinar

Thank You!