22IT305 – DATABASE MANAGEMENT SYSTEMS

Unit I & LP 1 – DATA MODELS & CODD'S RULE

1. DATA MODEL

Data models in DBMS help to understand the design at the conceptual, physical, and logical levels as it provides a clear picture of the data making it easier for developers to create a physical database.

Data models are used to describe how the data is stored, accessed, and updated in a DBMS. A set of symbols and text is used to represent them so that all the members of an organization can understand how the data is organized. It provides a set of conceptual tools that are vastly used to represent the description of data.

1.1 Type of Database Models

There are several different Database model types. Here is a list of the 4 popular Database models:

1. Relational Model
2. Entity-relationship Model
3. Object based Data Model
4. Semi Structured Data Model

1.1.1 Relational Model

- In this model, data is organized in two-dimensional **tables** and the relationship is maintained by storing a common field.
- This model was introduced by **E.F Codd** in 1970, and since then it has been the most widely used database model.
- The basic structure of data in the relational model is **tables**. All the information related to a particular type is stored in **rows** of that table.
- Hence, tables are also known as **relations** in the relational model.

- You can design tables, normalize them to reduce data redundancy, and use Structured Query language or SQL to access data from the tables.
- Some of the most popular databases are based on this database model. For example, **Oracle**, **MySQL**, etc.

student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	19

subject_id	name	teacher
1	Java	Mr. J
2	Python	Miss Py
3	JavaScript	Mrs. JS
4	C#	Mr. C

student_id	subject_id	marks
1	1	98
1	2	77
2	1	76
3	2	88

Advantages of the Relational Model

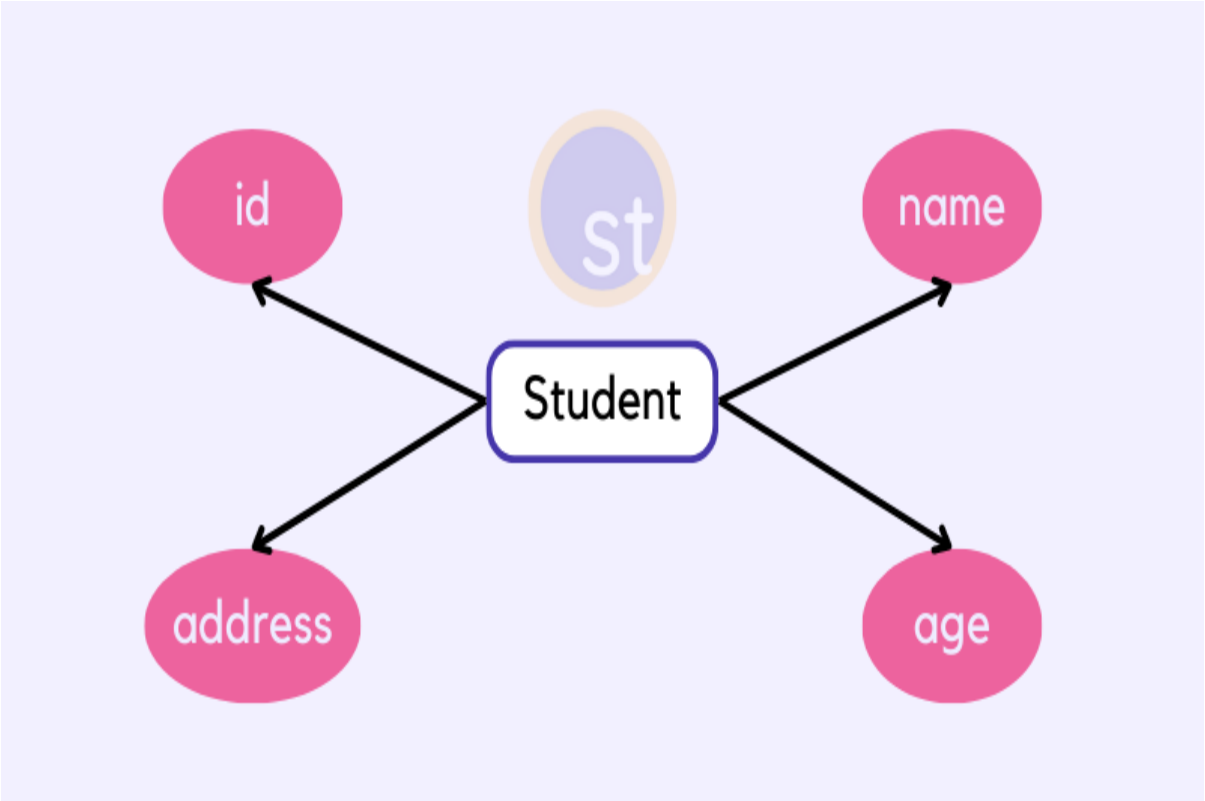
1. It's simple and easy to implement.
2. Poplar database software is available for this database model.
3. It supports SQL using which you can easily query the data.

1.1.2 Entity-relationship Model

- In this database model, relationships are created by dividing objects of interest into entities and their characteristics into attributes.
- Different entities are related using **relationships**.
- ER Models are defined to represent the relationships in pictorial form to make it easier for different stakeholders to understand.
- This model is good to design a database, which can then be turned into tables in a relational model (explained below).
- Let's take an **example**, If we have to design a School Database, then the **Student** will be an **entity** with **attributes** *name, age, address,* etc. As an **Address** is generally complex, it can be

another **entity** with **attributes** *street*, *pincode*, *city*, etc, and there will be a relationship between them.

- Relationships can also be of different types. You can learn about ER Diagrams in detail if you want to learn about entities and relationships.

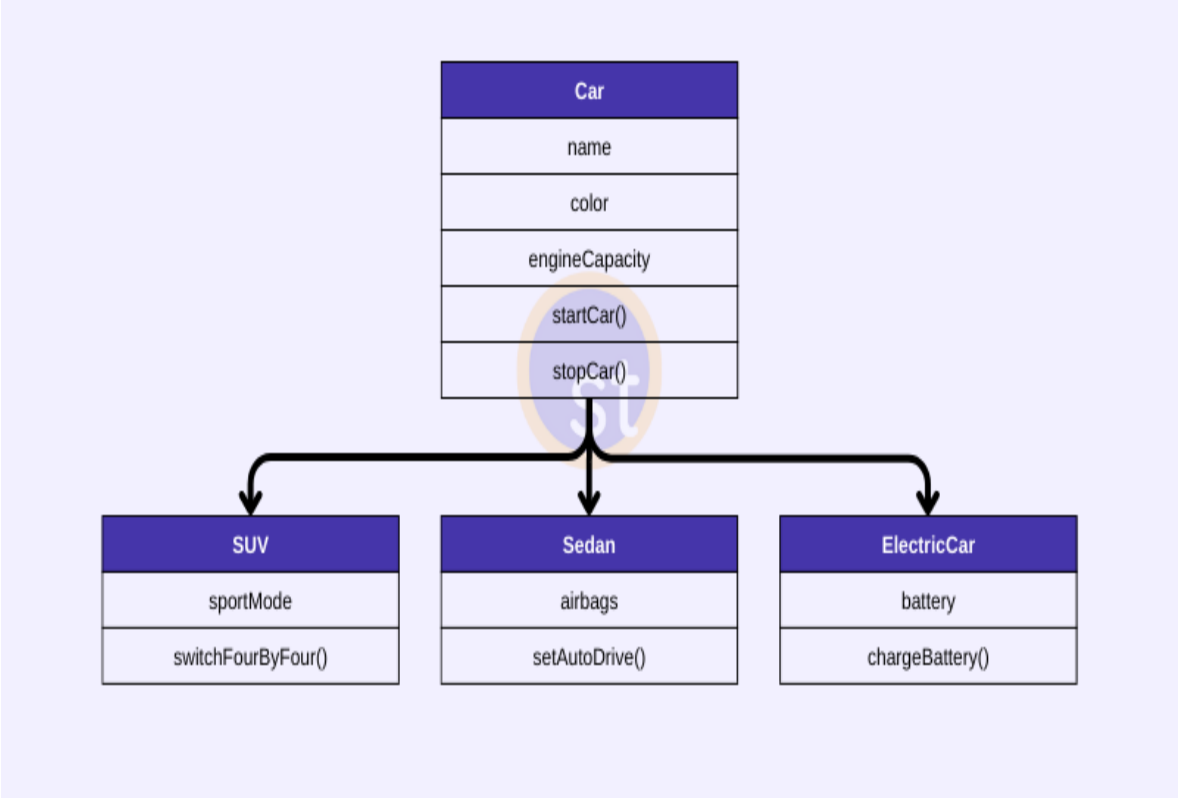


Advantages of the ER Model

1. It is easy to understand and design.
2. Using the ER model we can represent data structures easily.
3. As the ER model cannot be directly implemented into a database model, it is just a step toward designing the relational database model.

1.1.3 Object-oriented Model

- In this model, data is stored in the form of objects.
- The behavior of the object-oriented database model is just like object-oriented programming.
- A very popular example of an Object Database management system or **ODBMS** is **MongoDB** which is also a NoSQL database.
- This database model is not mature enough as compared to the relational database model.



Advantages of the Object-oriented Model

1. It can easily support complex data structures, with relationships.
2. It also supports features like Inheritance, Encapsulation, etc.

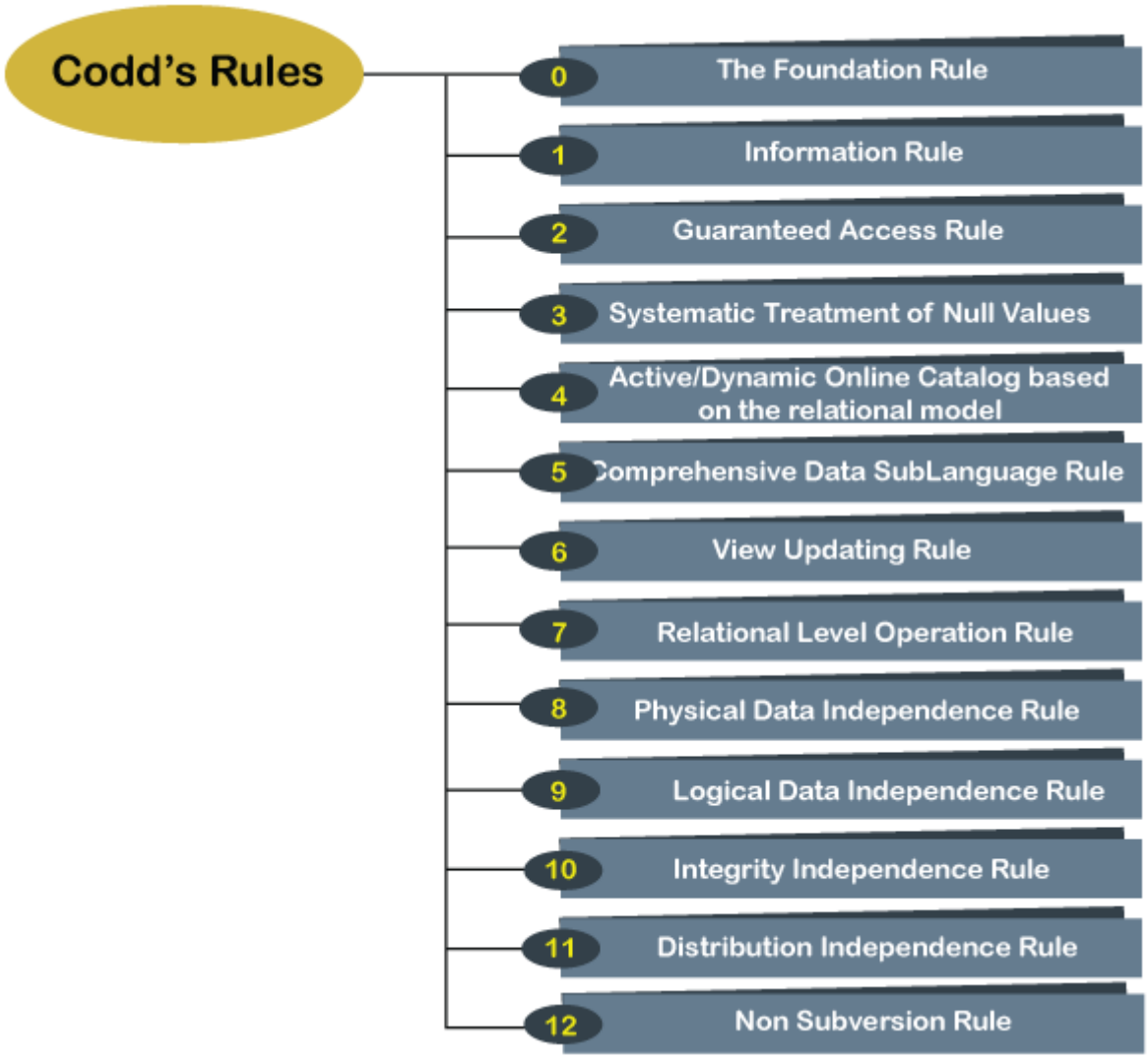
1.1.4 Semi-Structured Data Model

A semi-structured data model is a generalized form of the relational model, which allows representing data in a flexible way, hence we cannot differentiate between data and schema in this model because, in this model, some entities have a missing attribute(s) and on the other hand, some entities might have some extra attribute(s) which in turn makes it easy to update the schema of the database.

For example - We can say a data model is semi-structured if in some attributes we are storing both atomic values (values that can't be divided further, for example, Roll_No) as well as a collection of values.

2. CODD'S 12 RULES

Every database has tables, and constraints cannot be referred to as a rational database system. And if any database has only relational data model, it cannot be a **Relational Database System (RDBMS)**. So, some rules define a database to be the correct RDBMS. These rules were developed by **Dr. Edgar F. Codd (E.F. Codd)** in **1985**, who has vast research knowledge on the Relational Model of database Systems. Codd presents his 13 rules for a database to test the concept of DBMS against his relational model, and if a database follows the rule, it is called a **true relational database (RDBMS)**. These 13 rules are popular in RDBMS, known as **Codd's 12 rules**.



Rule 0: The Foundation Rule

The database must be in relational form. So that the system can handle the database through its relational capabilities.

Rule 1: Information Rule

A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns.

Rule 2: Guaranteed Access Rule

Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name, and column name.

Rule 3: Systematic Treatment of Null Values

This rule defines the systematic treatment of Null values in database records. The null value has various meanings in the database, like missing the data, no value in a cell, inappropriate information, unknown data and the primary key should not be null.

Rule 4: Active/Dynamic Online Catalog based on the relational model

It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary. It authorizes users to access the database and implement a similar query language to access the database.

Rule 5: Comprehensive Data SubLanguage Rule

The relational database supports various languages, and if we want to access the database, the language must be the explicit, linear or well-defined syntax, character strings and supports the comprehensive: data definition, view definition, data manipulation, integrity constraints, and limit transaction management operations. If the database allows access to the data without any language, it is considered a violation of the database.

Rule 6: View Updating Rule

All views table can be theoretically updated and must be practically updated by the database systems.

Rule 7: Relational Level Operation (High-Level Insert, Update and delete) Rule

A database system should follow high-level relational operations such as insert, update, and delete in each level or a single row. It also supports union, intersection and minus operation in the database system.

Rule 8: Physical Data Independence Rule

All stored data in a database or an application must be physically independent to access the database. Each data should not depend on other data or an application. If data is

updated or the physical structure of the database is changed, it will not show any effect on external applications that are accessing the data from the database.

Rule 9: Logical Data Independence Rule

It is similar to physical data independence. It means, if any changes occurred to the logical level (table structures), it should not affect the user's view (application). For example, suppose a table either split into two tables, or two table joins to create a single table, these changes should not be impacted on the user view application.

Rule 10: Integrity Independence Rule

A database must maintain integrity independence when inserting data into table's cells using the SQL query language. All entered values should not be changed or rely on any external factor or application to maintain integrity. It is also helpful in making the database-independent for each front-end application.

Rule 11: Distribution Independence Rule

The distribution independence rule represents a database that must work properly, even if it is stored in different locations and used by different end-users. Suppose a user accesses the database through an application; in that case, they should not be aware that another user uses particular data, and the data they always get is only located on one site. The end users can access the database, and these access data should be independent for every user to perform the SQL queries.

Rule 12: Non Subversion Rule

The non-submersion rule defines RDBMS as a SQL language to store and manipulate the data in the database. If a system has a low-level or separate language other than SQL to access the database system, it should not subvert or bypass integrity to transform data.