

# Data Models in DBMS

A Data Model in Database Management System (DBMS) is the concept of tools that are developed to summarize the description of the database. Data Models provide us with a transparent picture of data which helps us in creating an actual database. It shows us from the design of the data to its proper implementation of data.

## Types of Relational Models

- Conceptual Data Model
- Representational Data Model
- Physical Data Model

### 1. Conceptual Data Model

The conceptual data model describes the database at a very high level and is useful to understand the needs or requirements of the database. It is this model, that is used in the requirement-gathering process i.e. before the Database Designers start making a particular database. One such popular model is the entity/relationship model (ER model). The E/R model specializes in entities, relationships, and even attributes that are used by database designers. In terms of this concept, a discussion can be made even with non-computer science(non-technical) users and stakeholders, and their requirements can be understood.

#### Characteristics of a conceptual data model

- Offers Organization-wide coverage of the business concepts.
- This type of Data Models are designed and developed for a business audience.
- The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and technology. The focus is to represent data as a user will see it in the “real world.”
- Conceptual data models known as Domain models create a common vocabulary for all stakeholders by establishing basic concepts and scope

### 2. Representational / Logical Data Model

This type of data model is used to represent only the logical part of the database and does not represent the physical structure of the database. The representational data model allows us to focus primarily, on the design part of the database. A popular representational model is a

Relational model. The relational Model consists of Relational Algebra and Relational Calculus. In the Relational Model, we basically use tables to represent our data and the relationships between them. It is a theoretical concept whose practical implementation is done in Physical Data Model.

The advantage of using a Representational data model is to provide a foundation to form the base for the Physical model

### **3. Physical Data Model**

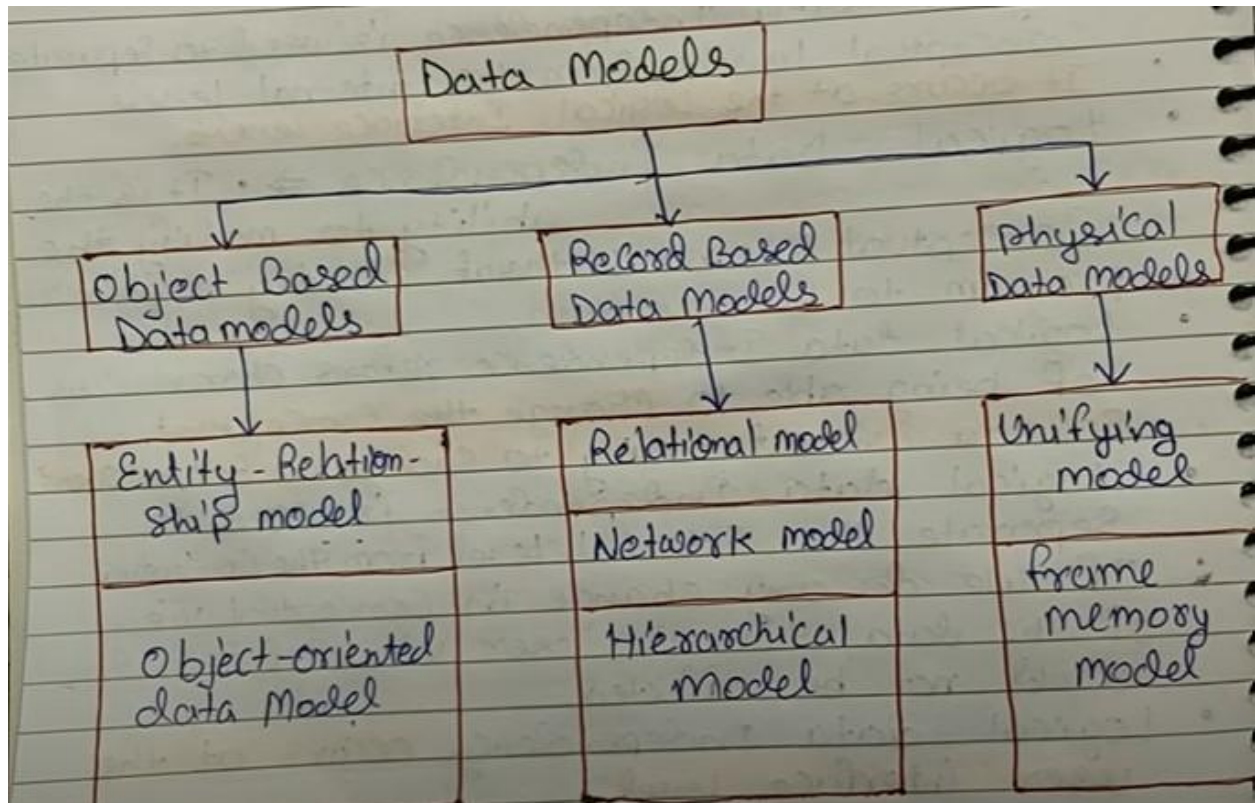
The physical Data Model is used to practically implement Relational Data Model. Ultimately, all data in a database is stored physically on a secondary storage device such as discs and tapes. This is stored in the form of files, records, and certain other data structures. It has all the information on the format in which the files are present and the structure of the databases, the presence of external data structures, and their relation to each other. Here, we basically save tables in memory so they can be accessed efficiently. In order to come up with a good physical model, we have to work on the relational model in a better way. Structured Query Language (SQL) is used to practically implement Relational Algebra.

This Data Model describes HOW the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.

#### **Characteristics of a physical data model:**

- The physical data model describes data need for a single project or application though it maybe integrated with other physical data models based on project scope.
- Data Model contains relationships between tables that which addresses cardinality and nullability of the relationships.
- Developed for a specific version of a DBMS, location, data storage or technology to be used in the project.
- Columns should have exact datatypes, lengths assigned and default values.
- Primary and Foreign keys, views, indexes, access profiles, and authorizations, etc. are defined

## Types of Data Models:



### Object-Based Data Model:

- An object-based data model represents data using object-oriented principles. In this model, data is organized into objects, which can have attributes (data fields) and methods (functions or procedures that operate on the data).
- Object-based models are used to map real-world entities and their relationships into a data structure. They are closely related to object-oriented programming concepts and are often used in object-oriented database systems.
- Examples of object-based data models include the Entity-Relationship (ER) model and the Unified Modeling Language (UML) class diagrams.

## Record-Based Data Model:

- A record-based data model organizes data into records, where each record represents a single entity or object. Records consist of fields or attributes that store specific pieces of information.
- Record-based models are commonly associated with relational database management systems (RDBMS), where data is stored in tables, and each row in a table corresponds to a record.
- The relational model is a well-known example of a record-based data model.

## Physical-Based Data Model:

- A physical-based data model, as the name suggests, focuses on the physical implementation details of data storage, access, and optimization within a specific database management system.
- It includes details like storage structures (e.g., tables, indexes, files), access paths, file organization, and data storage optimization techniques.
- Physical data models are essential for database administrators and developers to fine-tune the performance and manageability of a database system.
- An example of a physical-based data model is the way data is physically stored in a specific RDBMS, like the internal file structures used by MySQL, Oracle, or SQL Server.

## Some other types of Data model include :

### 1. Hierarchical Model

The hierarchical Model is one of the oldest models in the data model which was developed by IBM, in the 1950s. In a hierarchical model, data are viewed as a collection of tables, or we can say segments that form a hierarchical relation. In this, the data is organized into a tree-like structure where each record consists of one parent record and many children. Even if the segments are connected as a chain-like structure by logical associations, then the instant structure can be a fan structure with multiple branches. We call the illogical associations as directional associations.

## **2. Network Model**

The Network Model was formalized by the Database Task group in the 1960s. This model is the generalization of the hierarchical model. This model can consist of multiple parent segments and these segments are grouped as levels but there exists a logical association between the segments belonging to any level. Mostly, there exists a many-to-many logical association between any of the two segments.

## **3. Object-Oriented Data Model**

In the Object-Oriented Data Model, data and their relationships are contained in a single structure which is referred to as an object in this data model. In this, real-world problems are represented as objects with different attributes. All objects have multiple relationships between them. Basically, it is a combination of Object Oriented programming and a Relational Database Model.

## **4. Float Data Model**

The float data model basically consists of a two-dimensional array of data models that do not contain any duplicate elements in the array. This data model has one drawback it cannot store a large amount of data that is the tables can not be of large size.

## **5. Context Data Model**

The Context data model is simply a data model which consists of more than one data model. For example, the Context data model consists of ER Model, Object-Oriented Data Model, etc. This model allows users to do more than one thing which each individual data model can do.

## **6. Semi-Structured Data Model**

Semi-Structured data models deal with the data in a flexible way. Some entities may have extra attributes and some entities may have some missing attributes. Basically, you can represent data here in a flexible way.

## **Entity-Relationship (ER) Data Model:**

The Entity-Relationship (ER) data model is a conceptual data model used in database design to represent the structure and relationships within a database. It is a graphical representation that helps define and understand how data is organized in a relational database.

Or ( 2nd Way to define is )

ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.

- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

## **Key Concepts of ER Data model**

### **Entity:**

An entity represents a real-world object, concept, or thing that can be uniquely identified.

Examples of entities might include "Customer," "Product," or "Employee."

Each entity is typically associated with attributes that describe the properties of the entity. For instance, a "Customer" entity might have attributes like "CustomerID," "Name," and "Email."

### **Relationship:**

A relationship represents an association or connection between two or more entities.

Relationships illustrate how entities are related to each other in the database.

Relationships are usually classified as one-to-one, one-to-many, or many-to-many, depending on how instances of entities are related. For example, a "Customer" may have a one-to-many relationship with "Orders."

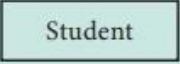
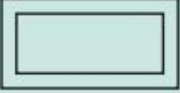


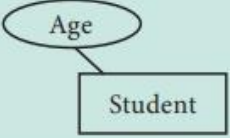
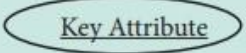
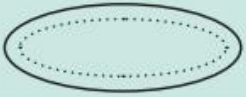

### **Attributes:**

Attributes are properties or characteristics that describe entities. Each entity can have multiple attributes that capture specific pieces of information.

Attributes are classified as simple (atomic) or composite (composed of sub-attributes). For example, an "Address" attribute can be composed of sub-attributes like "Street," "City," and "Postal Code."

### Key Attributes:

Key attributes are attributes that uniquely identify instances of an entity. They are crucial for ensuring data integrity and are often used as primary keys in relational databases. For instance, the "CustomerID" attribute in the "Customer" entity can serve as a key attribute.

Table 3.6 ER diagram Notations		
ER Component	Description (how it is represented)	Notation
Entity - Strong	Simple rectangular box	
Entity – Weak	Double rectangular boxes	
Relationships	Rhombus symbol - Strong	
between Entities	Rhombus within rhombus – Weak	
Attributes	Ellipse Symbol connected to the entity	
Key Attribute for Entity	Underline the attribute name inside Ellipse	
Derived Attribute for Entity	Dotted ellipse inside main ellipse	
Multivalued Attribute for Entity	Double Ellipse	

## Advantages of Data Models

- Data Models help us in representing data accurately.
- It helps us in finding the missing data and also in minimizing Data Redundancy.
- Data Model provides data security in a better way.
- The data model should be detailed enough to be used for building the physical database.
- The information in the data model can be used for defining the relationship between tables, primary and foreign keys, and stored procedures.
- Disadvantages of Data Models
- In the case of a vast database, sometimes it becomes difficult to understand the data model.
- You must have the proper knowledge of SQL to use physical models.
- Even smaller change made in structure require modification in the entire application.
- There is no set data manipulation language in DBMS.
- To develop Data model one should know physical data stored characteristics.

## Types of Relationship in ER- Diagram

### One-to-One (1:1) Relationship:

Definition: In a one-to-one relationship, one entity instance from the first entity (Entity A) is associated with only one entity instance from the second entity (Entity B), and vice versa. It means that for every record in Entity A, there is exactly one corresponding record in Entity B, and no more.

Example: In a healthcare database, a one-to-one relationship could exist between the "Patient" entity and the "MedicalRecord" entity. Each patient has exactly one medical record, and each medical record belongs to only one patient.

### One-to-Many (1:N) Relationship:

Definition: In a one-to-many relationship, one entity instance from the first entity (Entity A) can be associated with multiple entity instances from the second entity (Entity B), but each entity instance from Entity B is associated with only one entity instance from Entity A. This is also known as a "parent-to-child" relationship.



Example: In a customer relationship management (CRM) system, a one-to-many relationship could exist between the "Company" entity and the "Contact" entity. Each company can have multiple contacts (employees), but each contact is associated with only one company.

### **Many-to-Many (M:N) Relationship:**

Definition: In a many-to-many relationship, multiple entity instances from the first entity (Entity A) can be associated with multiple entity instances from the second entity (Entity B), and vice versa. This type of relationship often requires the use of a junction table or an associative entity to resolve it.

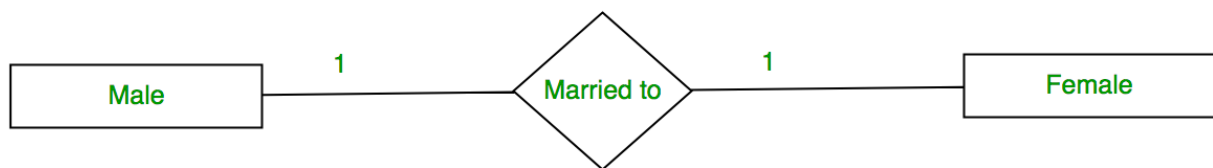
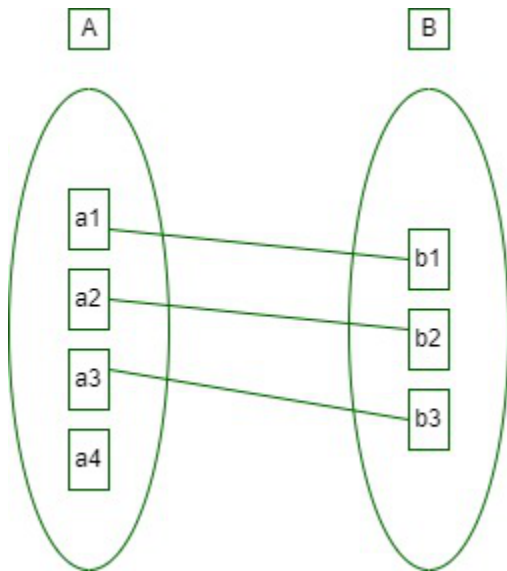
Example: In a university database, a many-to-many relationship could exist between the "Student" entity and the "Course" entity. Multiple students can enroll in multiple courses, and multiple courses can have multiple students enrolled.

### **Many-To-One (N:M) Relationship:**

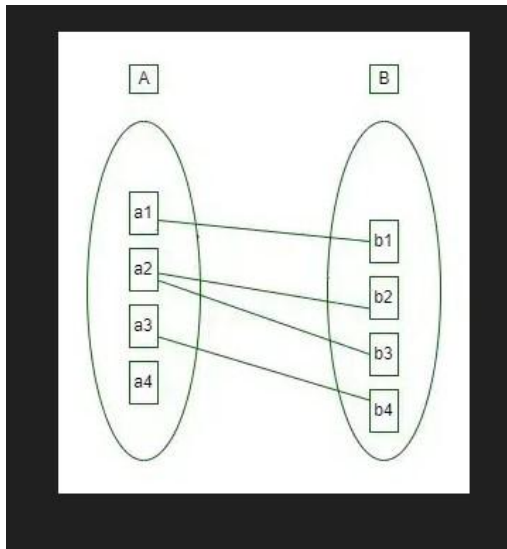
Definition: In a many-to-one relationship, multiple records or instances from Entity A can be related to a single record or instance in Entity B. This implies that for each entity instance in Entity B, there can be multiple related instances in Entity A, but for each instance in Entity A, there is only one corresponding instance in Entity B.

Example: Consider a database for a library. There could be a many-to-one relationship between the "Books" entity and the "Authors" entity. Many books can be written by a single author (one author can have multiple books), but each book is written by one author.

## One To One 👍



## One to Many



## Many To Many

