**Entity-Relationship (ER) Model in DBMS**

The **ER Model** is a diagrammatic way to design a database by representing real-world objects (entities) and their relationships. It helps in structuring data before creating a database.

**Components of ER Model**

1. **Entity** – A real-world object with unique properties. Example: *Student, Course*.
2. **Attributes** – Characteristics of an entity. Example: *Student has Student\_ID, Name, Age*.
3. **Relationship** – Defines how two entities are connected. Example: *A Student enrolls in a Course*.

**Example of ER Model**

Consider a **School Database**:

* **Entities**:
  + *Student* (Student\_ID, Name, Age)
  + *Course* (Course\_ID, Course\_Name)
* **Relationship**: A Student **enrolls** in a Course.

**ER Diagram for School Database**

Here is the ER diagram representation:

pgsql

CopyEdit

+------------+ +-------------+

| Student | | Course |

|------------| |-------------|

| Student\_ID |◄──────────► | Course\_ID |

| Name | Enrolls | Course\_Name |

| Age | | |

+------------+ +-------------+

**Explanation**

* The **Student** and **Course** are entities.
* **Enrolls** is the relationship between them.
* **Attributes** define the properties of each entity.

**Conclusion**

The **ER Model** is useful for designing databases in a structured way. It ensures clear relationships between data and helps in database creation.

**Levels of Abstraction in DBMS**

In **Database Management Systems (DBMS)**, **levels of abstraction** help in managing and organizing data efficiently. These levels hide unnecessary details from users and provide a structured way to view the database.

**Three Levels of Abstraction in DBMS**

There are **three main levels** of abstraction in DBMS:

1. **Internal Level (Physical Level)**
   * It describes **how** data is stored in the database.
   * It includes **file structures, indexing, and data storage methods**.
   * Example: A database stores data as **tables in hard disks or memory**.
2. **Conceptual Level (Logical Level)**
   * It defines **what** data is stored and the **relationships** between data.
   * It hides physical storage details and focuses on **data organization**.
   * Example: A **Student table** with attributes *Student\_ID, Name, Age*.
3. **External Level (View Level)**
   * It defines how users view the database.
   * Different users have different views of data.
   * Example: A **Student** sees only their marks, while an **Admin** sees all student details.

**Diagram of Levels of Abstraction in DBMS**

pgsql

CopyEdit

+-------------------------+

| External Level (Views) |

| User's View of Data |

+-------------------------+

│

▼

+-------------------------+

| Conceptual Level |

| Logical Structure of DB|

+-------------------------+

│

▼

+-------------------------+

| Internal Level |

| Physical Storage of DB |

+-------------------------+

**Explanation**

* The **External Level** allows users to see only relevant data.
* The **Conceptual Level** manages database structure and relationships.
* The **Internal Level** deals with actual data storage in memory.

**Conclusion**

The **three levels of abstraction** ensure **security, efficiency, and ease of database management**. This structure helps different users access data without knowing storage details.

**Data Model in DBMS**

A **Data Model** is a way to organize and define how data is stored, connected, and manipulated in a database. It provides a **framework** for designing databases and ensures consistency in data representation.

**Types of Data Models in DBMS**

There are **four main types** of data models:

**1. Hierarchical Data Model**

* Data is organized in a **tree-like structure** (parent-child relationship).
* One parent can have multiple children, but each child has **only one parent**.
* **Example:** An organization structure where a **Manager** has multiple **Employees**.

markdown

CopyEdit

Manager

/ \

Employee1 Employee2

**2. Network Data Model**

* Similar to the hierarchical model but allows **many-to-many relationships**.
* A child can have **multiple parents**.
* **Example:** A **Student** enrolls in multiple **Courses**, and each course has multiple students.

nginx

CopyEdit

Student ◄──► Course

**3. Relational Data Model *(Most widely used)***

* Data is stored in **tables (relations)** with rows and columns.
* **Primary keys** uniquely identify records, and **foreign keys** create relationships.
* **Example:** A **Student Table** with Student\_ID, Name, and Age, related to a **Course Table**.

| **Student\_ID** | **Name** | **Age** |
| --- | --- | --- |
| 101 | Alex | 20 |
| 102 | John | 21 |

**4. Entity-Relationship (ER) Model**

* Uses **entities (objects)** and **relationships** to design databases.
* Entities have **attributes**, and relationships define connections.
* **Example:** A **Student** enrolls in a **Course**.

nginx

CopyEdit

Student ◄──Enrolls──► Course

**Conclusion**

* **Data models** help in structuring databases.
* The **Relational Model** is the most commonly used.
* Each model has its own way of storing and managing data.
* --------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Integrity Constraints** are rules applied to a database to ensure **accuracy, consistency, and reliability** of data. These constraints prevent invalid data entry and maintain correctness in the database.

**Types of Integrity Constraints with Examples**

**1. Primary Key Constraint**

* Ensures that **each row in a table is unique**.
* The **primary key** column cannot have **duplicate or NULL** values.
* **Example:**

**Student Table**

| **Student\_ID (Primary Key)** | **Name** | **Age** |
| --- | --- | --- |
| 101 | Alex | 20 |
| 102 | John | 21 |

* + Here, **Student\_ID** is the **Primary Key**, ensuring each student has a unique ID.

**2. Foreign Key Constraint**

* Ensures **referential integrity** by linking two tables.
* The foreign key in one table must match a **primary key in another table**.
* **Example:**

**Student Table**

| **Student\_ID** | **Name** | **Age** |
| --- | --- | --- |
| 101 | Alex | 20 |
| 102 | John | 21 |

**Course Table**

| **Course\_ID** | **Course\_Name** | **Student\_ID (Foreign Key)** |
| --- | --- | --- |
| C101 | DBMS | 101 |
| C102 | Java | 102 |

* + Here, **Student\_ID** in the **Course Table** is a **Foreign Key**, ensuring that only existing students can enroll in a course.

**3. Not Null Constraint**

* Ensures that a column **cannot have NULL values**.
* **Example:**  
  If we define **Name** as **NOT NULL** in the **Student Table**, every student **must have a name**.

**4. Unique Constraint**

* Ensures that all values in a column are **different** (except NULL values).
* **Example:**  
  A table with a **Phone Number** column should not allow **duplicate numbers**.

**5. Check Constraint**

* Ensures that values meet a **specific condition**.
* **Example:**
  + A **Check Constraint** on the **Age** column to allow only students above **18 years old**.

**Conclusion**

* **Integrity Constraints** maintain **data correctness and consistency**.
* **Primary Key, Foreign Key, Not Null, Unique, and Check** constraints prevent **invalid data entry**.
* They help in **better database management and security**.