**Hexaware Technical Training Program**

**Day 4 - March 13 , 2025**

**Database Management System (DBMS)**

**DBMS :**

A **Database Management System (DBMS)** is software that allows users to **create, manage, and manipulate databases** efficiently. It acts as an **interface between the user and the database**, ensuring data is **stored, retrieved, and modified** securely and efficiently.

**Features of DBMS**

* **Data Organization** – Stores data in structured formats like tables.
* **Data Security** – Provides authentication and access control.
* **Data Integrity** – Ensures accuracy and consistency of data.
* **Concurrency Control** – Allows multiple users to access data simultaneously.
* **Backup & Recovery** – Prevents data loss in case of system failure.
* **Query Processing** – Uses SQL for data retrieval and manipulation.

**Types of DBMS**

1️. **Hierarchical DBMS** – Data is organized in a tree-like structure (e.g., IBM’s IMS).  
2️. **Network DBMS** – Uses a graph structure, allowing multiple relationships (e.g., IDMS).  
3️. **Relational DBMS (RDBMS)** – Stores data in tables with rows and columns (e.g., MySQL, PostgreSQL).  
4️. **Object-Oriented DBMS (OODBMS)** – Uses object-oriented principles (e.g., MongoDB).

**Advantages of DBMS**

✔ **Data Consistency** – Ensures uniform data across applications.  
✔ **Scalability** – Handles large amounts of data efficiently.  
✔ **Reduced Redundancy** – Avoids duplicate data storage.  
✔ **Better Performance** – Optimized data retrieval and transactions.

**Data Storage in DBMS**

Data in a **Database Management System (DBMS)** is stored in a **structured format** within tables, files, and indexes, ensuring efficient access, retrieval, and management.

**Types of Data Storage in DBMS:**

1️. **Primary Storage (Main Memory):** Stores active database transactions and buffers for quick access.  
2️. **Secondary Storage (Disk Storage):** Stores database files permanently (e.g., HDDs, SSDs).  
3️. **Tertiary Storage (Backup & Archival):** Stores backups for disaster recovery (e.g., cloud storage, tapes).  
4️. **Indexing & Caching:** Helps in **faster retrieval** by organizing data efficiently.

**Example:** In a food delivery app, customer orders, restaurant details, and payments are stored in different tables and retrieved when needed.

**Data Models in DBMS**

A **Data Model** defines how data is **structured, stored, and manipulated** in a DBMS.

**Types of Data Models:**  
1️. **Hierarchical Model:** Organizes data in a **tree-like structure** (Parent-Child relationship).  
📌 **Example:** File directories in an operating system.

2️. **Network Model:** Uses a **graph structure** to represent multiple relationships between records.  
📌 **Example:** A university system where students enroll in multiple courses.

3️. **Relational Model (RDBMS):** Data is stored in **tables (rows & columns)** using keys and relationships.  
📌 **Example:** MySQL database for an e-commerce website.

4️. **Object-Oriented Model (OODBMS):** Stores data as objects (similar to OOP concepts).  
📌 **Example:** MongoDB storing multimedia content.

5️. **Entity-Relationship (ER) Model:** Represents data with **entities, attributes, and relationships**.  
📌 **Example:** ER diagram for an online shopping platform.

**Data Retrieval in DBMS**

**Data retrieval** is the process of fetching or extracting **relevant data** from a database based on a given query or request. It is an essential function in **Database Management Systems (DBMS)** that allows users to access stored information efficiently.

**Factors Affecting Data Retrieval Speed**

✅ **Query Optimization** – Writing efficient SQL queries (e.g., avoiding SELECT \*)  
✅ **Proper Indexing** – Creating indexes on frequently searched columns  
✅ **Using Joins Wisely** – Avoiding unnecessary joins that slow down retrieval  
✅ **Database Normalization** – Organizing tables to **reduce redundancy** and improve speed  
✅ **Load Balancing** – Distributing queries across **multiple database servers**

**Data vs. Information**

Data and information are often used interchangeably, but they have distinct meanings in **computer science** and **data management**.

**What is Data?**

✅ **Definition**: Data refers to **raw, unprocessed facts** or **figures** that do not have any specific meaning on their own.  
✅ **Characteristics**:

* Unorganized, unprocessed
* Can be **numbers, text, images, or symbols**
* Has **no specific context**
* Needs **processing to become meaningful**  
  ✅ **Example**:
* 100, "John", "A", 95 → These are just numbers and text without context.
* A file containing a list of temperatures recorded every hour.

**What is Information?**

✅ **Definition**: Information is **processed, organized, and structured data** that has meaning and can be used for decision-making.  
✅ **Characteristics**:

* Processed and **meaningful**
* Organized in a **specific context**
* Helps in **decision-making**  
  ✅ **Example**:
* "John scored 95 in Math and received an A grade."
* A report showing the **average temperature** of a city in the last 7 days.

**Traditional File Storage**

In **early computing systems**, data was stored in **files** instead of databases.

**What is File Storage?**

✅ **Definition**: File storage is a system where data is stored in **individual files** and accessed using **file-handling operations**. Each file is **independent**, and the system does not enforce any relationships between files.

✅ **Characteristics**:

* Data is stored in **flat files** (e.g., .txt, .csv).
* Each file contains a **specific type of data** (e.g., customers, orders).
* Requires **manual handling** of data relationships.
* No **automatic indexing** for faster searches.

✅ **Example**:

* A company stores customer details in a **text file** (customers.txt).
* Orders are stored in another **file** (orders.txt).
* If a customer places an order, **manual linking** is needed to match the correct file entries

**Limitations of File-Based Storage**

❌ **Data Redundancy** – The same data is stored in multiple files, wasting storage space.  
❌ **Lack of Consistency** – Changes made in one file may not reflect in others.  
❌ **Difficult Data Retrieval** – Searching for specific data requires scanning the entire file.  
❌ **No Security Features** – No user access control, making data **vulnerable**.  
❌ **Data Integrity Issues** – No built-in mechanisms for maintaining **accura**

**What is a Database?**

**✅ Definition**

A **database** is an **organized collection of data** that allows users to **store, retrieve, manage, and manipulate** information efficiently. It provides a **structured way** to handle large amounts of data **securely and consistently**.

**✅ Example**

* A **bank database** stores customer details, account balances, and transaction records.
* An **e-commerce database** manages products, customers, and orders.

**Key Features of a Database**

✔ **Structured Storage** – Data is stored in **tables, rows, and columns**.  
✔ **Efficient Data Retrieval** – Uses **indexes and queries** for fast searching.  
✔ **Data Integrity & Consistency** – Prevents duplication and ensures accuracy.  
✔ **Security & Access Control** – Protects data with **user authentication**.  
✔ **Scalability** – Can handle **large amounts of data**.  
✔ **Multi-user Support** – Allows multiple users to access data **simultaneously**.

**Why Do We Use Databases?**

**🔹 To store and manage large amounts of data efficiently.  
🔹 To prevent data duplication and inconsistency.  
🔹 To speed up search and retrieval operations.  
🔹 To enforce security with access control mechanisms.  
🔹 To allow multiple users to access data simultaneously.**

**Data Model in DBMS**

A **data model** defines how data is structured, stored, and manipulated in a **Database Management System (DBMS)**. It provides an abstract representation of data, relationships, constraints, and rules.

**Types of Data Models**

Data models are broadly categorized into the following types:

**1. Hierarchical Data Model**

* Data is organized in a **tree-like structure**.
* Each parent node can have multiple child nodes, but each child has only one parent.
* Uses **pointers** to define relationships.
* Example: **IBM’s Information Management System (IMS)**.

**2. Network Data Model**

* Uses a **graph structure** instead of a tree.
* A child node can have **multiple parent nodes** (many-to-many relationships).
* Uses **pointers and links** to define relationships.

**3. Relational Data Model (RDBMS)**

* Data is stored in **tables (relations)** consisting of rows and columns.
* Uses **keys (Primary, Foreign)** to establish relationships.
* Ensures **data integrity and normalization**.
* Example: **MySQL, PostgreSQL, Oracle DB**.

**4. Object-Oriented Data Model (OODBMS)**

* Combines **object-oriented programming** and databases.
* Data is stored as **objects**, with attributes and methods.
* Supports **inheritance, encapsulation, and polymorphism**.
* Example: **db4o, ObjectDB**.

**5. Entity-Relationship (E-R) Model**

* Represents data using **entities (objects)** and **relationships**.
* Entities are connected using **lines** and symbols in an **ER Diagram**.

**6. Document-Oriented Data Model (NoSQL)**

* Data is stored in **JSON, BSON, or XML format**.
* Does not require a fixed schema (schema-less).
* Used in **MongoDB, CouchDB**.

**7. Physical Data Model**

* Defines how **data is physically stored** on hardware.
* Concerned with **disk storage, indexing, file organization, and access methods**.
* Optimized for **performance, storage efficiency, and retrieval speed**.

**Components of Physical Data Model**

🔹 **Storage Structure** – How data is physically stored (e.g., B-Trees, Hashing).  
🔹 **Access Methods** – How data is accessed (Sequential, Indexed, Hashed).  
🔹 **Indexing** – Speeds up searches (e.g., B+ Trees, Bitmap Indexing).

🔹 **Partitioning** – Divides large tables into smaller pieces

**Advantages**

✔ Improves database performance.  
✔ Optimizes query execution.  
✔ Manages disk space efficiently.

**Disadvantages**

❌ Complex to design.  
❌ Requires database tuning.

**Database Keys in DBMS**

Database **keys** are attributes or sets of attributes that help uniquely identify a row (record) in a table. They play a crucial role in maintaining **data integrity and relationships** between tables.

**Types of Database Keys**

**1. Primary Key**

* Uniquely identifies each record in a table.
* **Cannot have NULL values.**
* **Each table can have only one primary key.**
* Example:

🔹 **Employees Table**

| **emp\_id (PK)** | **emp\_name** | **dept\_id** |
| --- | --- | --- |
| 101 | Alice | 1 |
| 102 | Bob | 2 |

**2. Candidate Key**

* A set of **unique** attributes that can be a **primary key**.
* A table can have **multiple candidate keys**, but only one is chosen as the **primary key**.

🔹 **Example:**

| **emp\_id (Candidate Key)** | **emp\_email (Candidate Key)** | **emp\_name** |
| --- | --- | --- |
| 101 | alice@gmail.com | Alice |
| 102 | bob@gmail.com | Bob |

🔹 **Possible Candidate Keys**: emp\_id, emp\_email  
🔹 **Chosen Primary Key**: emp\_id

**3. Super Key**

* A **set of attributes** that can uniquely identify a row.
* **Includes the primary key and additional attributes.**
* **Every primary key is a super key, but not every super key is a primary key.**

🔹 **Example:** Super Keys in Employees Table:

1. {emp\_id}
2. {emp\_email}
3. {emp\_id, emp\_email}
4. {emp\_id, emp\_name}

**4. Foreign Key**

* Used to establish **relationships** between two tables.
* A **foreign key in one table** refers to the **primary key in another table**.
* Allows **referential integrity** between tables.

🔹 **Example:** **Employees Table**

| **emp\_id (PK)** | **emp\_name** | **dept\_id (FK)** |
| --- | --- | --- |
| 101 | Alice | 1 |
| 102 | Bob | 2 |

**Departments Table**

| **dept\_id (PK)** | **dept\_name** |
| --- | --- |
| 1 | HR |
| 2 | IT |

**5. Composite Key**

* A **key consisting of multiple attributes** that uniquely identify a record.
* Used when **a single attribute cannot uniquely identify a record**.

🔹 **Example:**

| **order\_id** | **product\_id** | **quantity** |
| --- | --- | --- |
| 1 | 101 | 2 |
| 1 | 102 | 1 |
| 2 | 101 | 3 |

🔹 **Composite Key:** {order\_id, product\_id} (since neither order\_id nor product\_id alone is unique).

**6. Alternate Key**

* A **candidate key that is NOT chosen as the primary key.**

🔹 **Example:**

| **emp\_id (PK)** | **emp\_email (Alternate Key)** | **emp\_name** |
| --- | --- | --- |
| 101 | alice@gmail.com | Alice |
| 102 | bob@gmail.com | Bob |

🔹 emp\_email is an **alternate key** since emp\_id is chosen as the **primary key**.

**7. Unique Key**

* Ensures all values in a column are **unique**, like a primary key.
* **Allows NULL values**, unlike a primary key.
* Example:

**Comparison of Keys**

| **Key Type** | **Unique?** | **NULL Allowed?** | **Number Per**  **Table** | **Purpose** |
| --- | --- | --- | --- | --- |
| **Primary Key** | ✅ Yes | ❌ No | 1 | Uniquely identifies a record |
| **Candidate Key** | ✅ Yes | ❌ No | Multiple | Potential primary key |
| **Super Key** | ✅ Yes | ❌ No | Many | Uniquely identifies a record |
| **Foreign Key** | ❌ No | ✅ Yes | Multiple | Links to another table’s primary key |
| **Composite Key** | ✅ Yes | ❌ No | 1 | Uses multiple columns as a key |
| **Alternate Key** | ✅ Yes | ❌ No | Multiple | Non-primary candidate key |
| **Unique Key** | ✅ Yes | ✅ Yes | Multiple | Ensures unique values |

**ER Model in DBMS**

The **Entity-Relationship (ER) Model** is a conceptual framework used to design and represent a **database structure** visually. It helps in understanding how **entities**, **attributes**, and **relationships** interact within a system.

**Key Components of ER Model**

**1️. Entities**

* An **entity** is a real-world object or concept that can have data stored about it.
* Each entity has **attributes** (characteristics).
* Represented as **rectangles** in an ER diagram.

🔹 **Example**:

* Student, Course, and Professor are entities.

**2️. Attributes**

* Properties or characteristics of an entity.
* Represented as **ellipses** in an ER diagram.

**Types of Attributes**

1. **Simple (Atomic) Attributes** – Cannot be divided further.  
   🔹 Example: student\_name, age
2. **Composite Attributes** – Can be divided into sub-parts.  
   🔹 Example: Full Name → (First Name, Last Name)
3. **Derived Attributes** – Can be derived from other attributes.  
   🔹 Example: Age (derived from Date of Birth)
4. **Multivalued Attributes** – Can have multiple values.  
   🔹 Example: Phone Numbers
5. **Key Attributes** – Uniquely identify an entity.  
   🔹 Example: Student\_ID

**3️. Relationships**

* Defines how two or more entities are related.
* Represented as **diamonds** in an ER diagram.

**Types of Relationships**

1. **One-to-One (1:1)** – Each entity is related to only one other entity.
   * Example: **One student** has **one library card**.
2. **One-to-Many (1:M)** – One entity is related to many entities.
   * Example: **One professor** teaches **many courses**.
3. **Many-to-Many (M:N)** – Many entities relate to many others.
   * Example: **Many students** enroll in **many courses**.

**4️, Keys in ER Model**

* **Primary Key**: Uniquely identifies an entity.
* **Foreign Key**: Helps link tables by referring to a primary key in another table.

**Example**

**Student-Course Database**:

**Entities:**

* **Student** (Student\_ID, Name, Age, Phone)
* **Course** (Course\_ID, Course\_Name, Credits)
* **Professor** (Prof\_ID, Prof\_Name, Department)

**Relationships:**

* **Enrolls In** (Student ↔ Course) → **Many-to-Many**
* **Teaches** (Professor ↔ Course) → **One-to-Many**

**Notation**

| **Symbol** | **Meaning** |
| --- | --- |
| 🟦 Rectangle | Entity |
| 🟠 Ellipse | Attribute |
| 🔷 Diamond | Relationship |
| 🔽 Line | Connection |
| ⭑ Underlined | Primary Key |

**Advantages of ER Model**

✅ Simple and easy to understand  
✅ Graphical representation for clarity  
✅ Helps in database design before implementation  
✅ Identifies relationships between entities

**Entity Relationship Model**

The **ER model** provides a **conceptual framework** for designing databases before implementation. It consists of:

* **Entities** (objects)
* **Attributes** (properties of entities)
* **Relationships** (associations between entities)

Entities are represented as **rectangles**, attributes as **ellipses**, and relationships as **diamonds** in ER diagrams.

**Types of Entities**

Entities are classified into **three main types**:

**1. Strong Entity**

* **Has a primary key** to uniquely identify records.
* Does not depend on any other entity.
* Represented by **a single rectangle** in an ER diagram.

🔹 **Example**:

* Student (Student\_ID, Name, Age, Department)
* Car (Car\_ID, Model, Price)

**2. Weak Entity**

* **Does not** have a primary key.
* Depends on a **strong entity** for identification.
* Always has a **foreign key** referencing a strong entity.
* Represented by **a double rectangle** in an ER diagram.

🔹 **Example**:

* Dependent (Dependent\_Name, Relationship) (depends on Employee)
* Order\_Item (Order\_ID, Item\_Name, Quantity) (depends on Order)

🛑 **Key Concept:** A **weak entity** must have a **discriminator** (partial key) to distinguish its instances.

**3. Associative (Bridge) Entity**

* Used to represent a **many-to-many (M:N) relationship**.
* Converts an M:N relationship into **two 1:M relationships**.

🔹 **Example**:

* Enrollment (Student\_ID, Course\_ID, Enrollment\_Date), which links Student and Course.
* Order\_Product (Order\_ID, Product\_ID, Quantity), linking Order and Product.

**Entity vs. Entity Set**

* **Entity** → A **single instance** of an object.  
  🔹 *Example*: A specific student **(John, 101, CS Department)**.
* **Entity Set** → A **collection of similar entities**.  
  🔹 *Example*: The entire **Student database** containing multiple students.

**Entity Representation in ER Diagram**

* **Strong Entity**: 🔳 **Single Rectangle**
* **Weak Entity**: 🔲 **Double Rectangle**
* **Attributes**: ⭕ **Ellipses (Ovals)**
* **Primary Key**: Underlined attribute
* **Relationships**: 🔷 **Diamonds**

**Types of Attributes**

An **attribute** is a characteristic or property of an entity that stores **data values**. Each entity consists of **one or more attributes** that define its properties.

**Types of Attributes**

**1. Single-Valued Attribute**

* Stores **only one value** for each entity instance.  
  🔹 *Example:*
  + Student → Roll Number (Single Value per Student)
  + Car → Engine Number (Each Car has one Engine Number)

**2. Multi-Valued Attribute**

* Stores **multiple values** for each entity instance.
* Represented by **a double oval** in ER diagrams.  
  🔹 *Example:*
  + Student → Phone Numbers (Can have multiple numbers)
  + Employee → Skills (Can have multiple skills)

**3. Composite Attribute**

* Can be **divided into sub-parts** (sub-attributes).  
  🔹 *Example:*
  + Full Name → (First Name, Last Name)
  + Address → (Street, City, State, ZIP Code)

**4. Derived Attribute**

* **Calculated from other attributes** rather than stored.
* Represented by a **dashed oval** in ER diagrams.  
  🔹 *Example:*
  + Age (Derived from Date of Birth)
  + Total Salary (Derived from Base Salary + Bonus)

**5. Key Attribute**

* **Uniquely identifies** an entity instance.
* **Primary keys** are key attributes.  
  🔹 *Example:*
  + Student → Roll Number
  + Car → Vehicle Registration Number

**6. Null Attribute**

* Can have a **NULL (empty) value**.  
  🔹 *Example:*
  + Employee → Spouse Name (If unmarried, it’s NULL)

| **Attribute Type** | **Description** | **Example** |
| --- | --- | --- |
| **Single-Valued** | One value per entity | Student → Age |
| **Multi-Valued** | Multiple values per entity | Student → Phone Numbers |
| **Composite** | Can be divided into sub-parts | Address → (Street, City, State) |
| **Derived** | Computed from other attributes | Age (from DOB) |
| **Key** | Unique identifier | Employee → Employee\_ID |
| **Null** | No value assigned | Employee → Spouse Name (NULL) |

**Relationships**

A **relationship** in a database defines how two or more entities are related to each other. It represents **associations** between entities and is depicted using **diamonds (♦) in ER diagrams**.

**Types of Relationships**

**1. One-to-One (1:1) Relationship**

* Each entity in **Set A** is related to at most **one** entity in **Set B** and vice versa.
* Rare in databases but used for special cases.

🔹 **Example:**

* **Person & Passport** → Each person has **one passport**, and each passport belongs to **one person**.

💡 **ER Diagram Representation:**  
Person (1) ⇔ (1) Passport

**2. One-to-Many (1:M) Relationship**

* One entity in **Set A** is related to **many** entities in **Set B**, but each entity in **Set B** is related to only **one** entity in **Set A**.

🔹 **Example:**

* **Customer & Orders** → A **customer** can place **many orders**, but each **order** belongs to **only one customer**.

💡 **ER Diagram Representation:**  
Customer (1) ⇔ (M) Order

**3. Many-to-Many (M:N) Relationship**

* One entity in **Set A** can be related to **many** entities in **Set B**, and vice versa.

🔹 **Example:**

* **Students & Courses** → A **student** can enroll in **multiple courses**, and a **course** can have **multiple students**.

💡 **ER Diagram Representation:**  
Student (M) ⇔ (N) Course

🛑 **To implement this in relational databases, an intermediate (junction) table is used.**  
For example:  
Enrollment (Student\_ID, Course\_ID)

**Degree of Relationships**

The **degree** of a relationship refers to the number of entities involved:

| **Degree** | **Description** | **Example** |
| --- | --- | --- |
| **Unary (1 entity)** | Relationship within the same entity | Employee → Manager (Employee reports to another Employee) |
| **Binary (2 entities)** | Relationship between two entities | Student → Course (Student enrolls in Course) |
| **Ternary (3 entities)** | Relationship between three entities | Doctor → Patient → Hospital |

**Cardinality of Relationships**

Cardinality defines the **number of instances** of one entity that can be associated with instances of another entity.

🔹 **Common Types of Cardinality:**

* **1:1** → One **CEO** manages **one company**.
* **1:M** → One **teacher** teaches **many students**.
* **M:N** → Many **students** enroll in many **courses**.

**ER Diagram Representation**

In an **ER diagram**, relationships are represented as **diamonds (♦)** connecting entities.