

DBMS Unit 1

Database System Applications

A Historical Perspective

- **Early 1960s:** Hierarchical and navigational data models used, inefficient and difficult to manage.
- **Late 1960s:** Charles Bachman invents the first general-purpose DBMS (Integrated Data Store) using the network model.
- **Late 1970s:** Edgar Codd proposes the relational model, using tables for easier data storage and retrieval.
- **Late 1980s:** IBM develops SQL for relational databases, allowing management of large datasets.
- **1980s:** Introduction of user-friendly DBMS products like Oracle, Sybase, and Microsoft SQL Server.
- **1990s:** Object-oriented DBMS (OODBMS) emerge for complex data structures, but adoption is limited commercially.
- **1991:** Microsoft releases MS Access, dominating the personal DBMS market.
- **1997:** XML integration into DBMS products begins.
- **2000s:** Rise of web applications and cloud computing leads to new DBMS systems like NoSQL databases.
- **Today:** DBMS continues to evolve, focusing on scalability, performance, and cloud support. Popular options include Oracle, SQL Server, MySQL, PostgreSQL, and MongoDB.

Database System Applications

- A Historical Perspective
- File Systems versus a DBMS
- The Data Model
- Levels of Abstraction in a DBMS
- Data Independence
- Structure of a DBMS

Introduction to Database Design

- Database Design and ER Diagrams
- Entities, Attributes, and Entity Sets
- Relationships and Relationship Sets

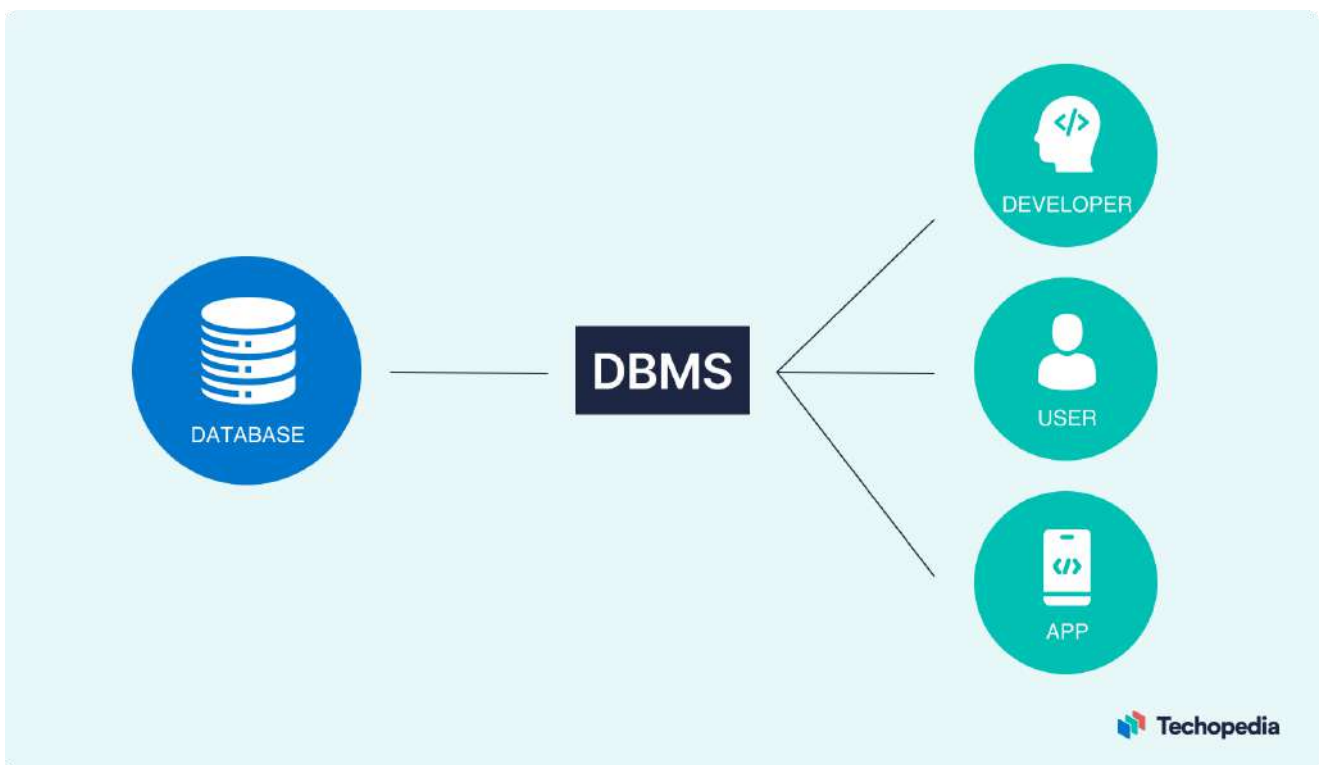
- Additional Features of the ER Model
- Conceptual Design With the ER Model

A Historical Perspective

- Data collection and storage challenges
- Evolution from hierarchical to relational models
- Contributions of Charles Bachman and Edgar Codd
- Emergence of SQL and relational databases
- Evolution into object-oriented and NoSQL databases
- Contemporary trends towards cloud-based applications

Introduction to Database Management System (DBMS)

- Definition of data, database, and DBMS
- Role of DBMS in managing data
- Importance in various sectors like banking, education, etc.
- Distinction between data and information
- Significance of metadata in database management



Database System Applications

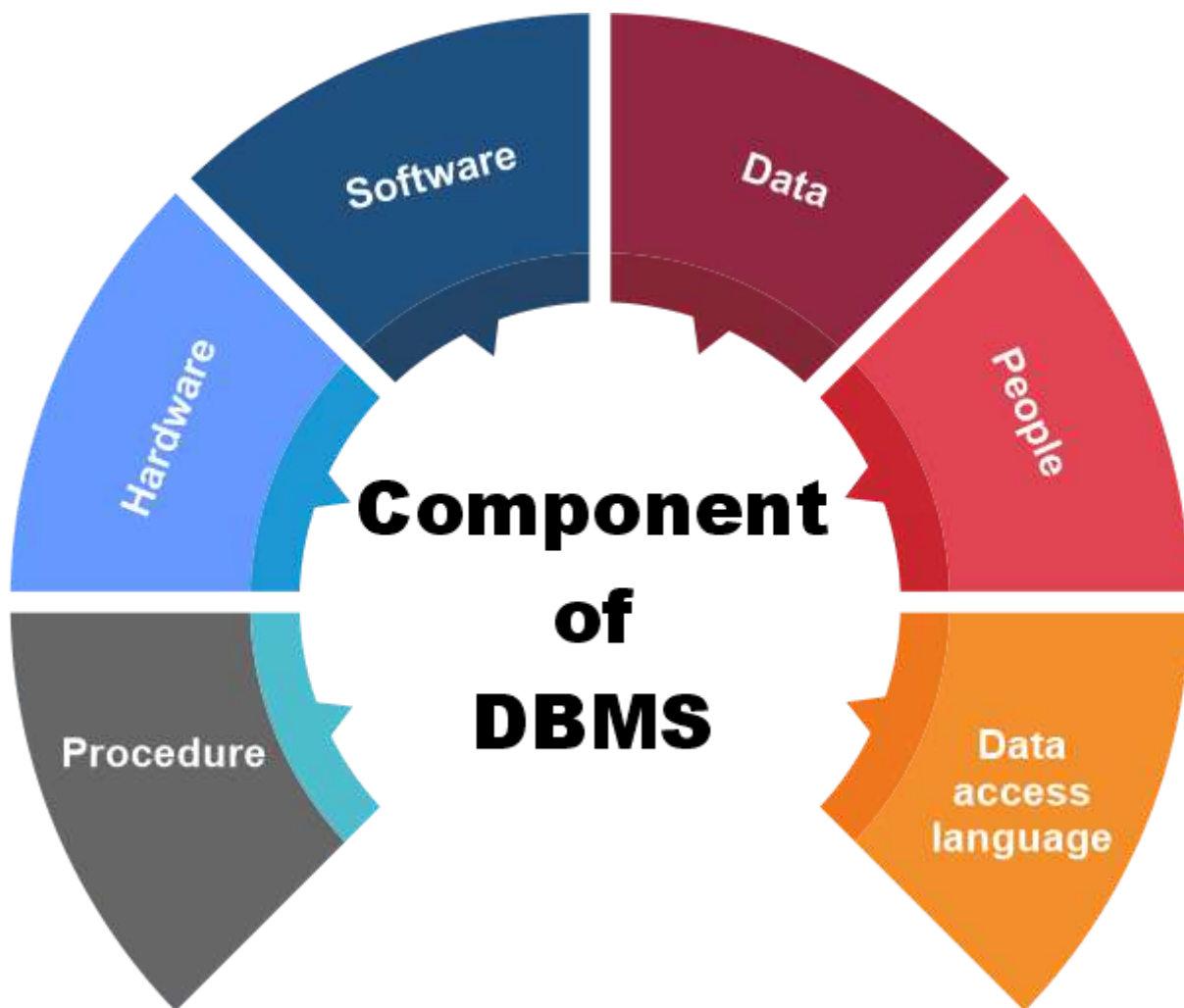
1. Banking
2. Universities
3. Airlines
4. Railways

5. Sales
6. Manufacturing
7. Telecommunication
8. Human Resources

Advantages of Database Management System

1. Efficient data storage and retrieval
2. Data independence
3. Backup and recovery facilities
4. Consistent and updated information
5. Enhanced data security
6. Centralized data administration
7. Representation of complex data relationships

Components of DBMS



1. Hardware
 - Importance of hardware in DBMS
 - Examples of hardware components
2. Software

- Role and types of software in DBMS
- Examples of DBMS software

3. Data

- Nature and importance of data in DBMS
- Types of data and metadata

4. People

- Roles of database administrator, software developer, and end user
- Classification of database users

File Systems vs. DBMS

What is a File System?

- Technique for arranging files in storage mediums like hard disks, pen drives, DVDs, etc.
- Organizes data for easy retrieval.
- Consists of various file types (e.g., mp3, mp4, txt, doc) grouped into directories.
- Installed directly into the computer's operating system (e.g., Windows, Linux).

What is DBMS?

- Software for storing and retrieving user data with security measures.
- Consists of programs manipulating the database.
- Accepts data requests from applications and provides specific data via the DBMS engine.
- Facilitates data storage and retrieval for users and third-party software in large systems.

Difference between File System and DBMS:

Feature	File System	DBMS
Purpose	Used to manage and organise the files stored in the hard disk of the computer	A software to store and retrieve the user's data
Redundant data	Present	No presence of redundant data
Query processing efficiency	Not so efficient	Efficient
Data consistency	Low	High (due to normalization)

Feature	File System	DBMS
Complexity	Less complex, does not support complicated transactions	More complexity in managing data, supports complicated transactions easier
Security	Less	Supports more security mechanisms
Cost	Less expensive compared to DBMS	Higher cost than file system
Crash recovery	Does not support crash recovery	Highly supported crash recovery mechanism

Features of File System

- Stores data in groups of files.
- Data dependency among files.
- Designed using languages like C/C++ and COBOL.
- Supports shared file systems.
- Enables fast file system recovery.

Features of DBMS

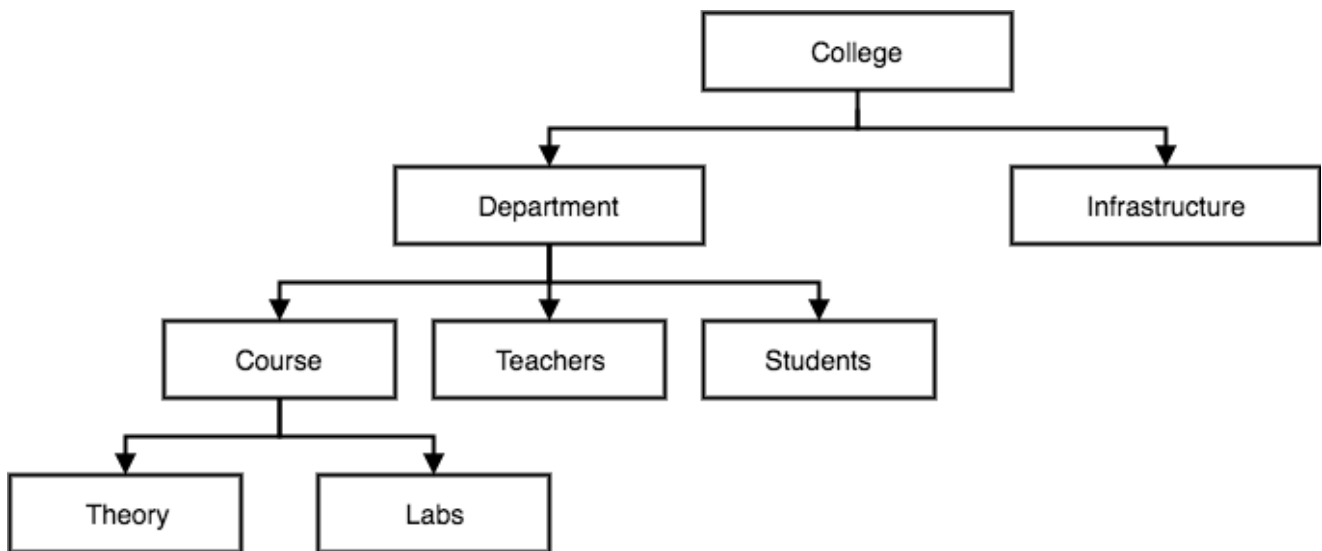
- User-accessible catalog of data.
- Transaction support.
- Concurrency control with recovery services.
- Authorization services.
- Ensures consistent data value across all locations.
- Offers data communication support.
- Provides independent utility services.
- Allows simultaneous file sharing among multiple users.

The Data Model

Data models in DBMS help in understanding database design at conceptual, physical, and logical levels, providing clarity for developers in creating physical databases.

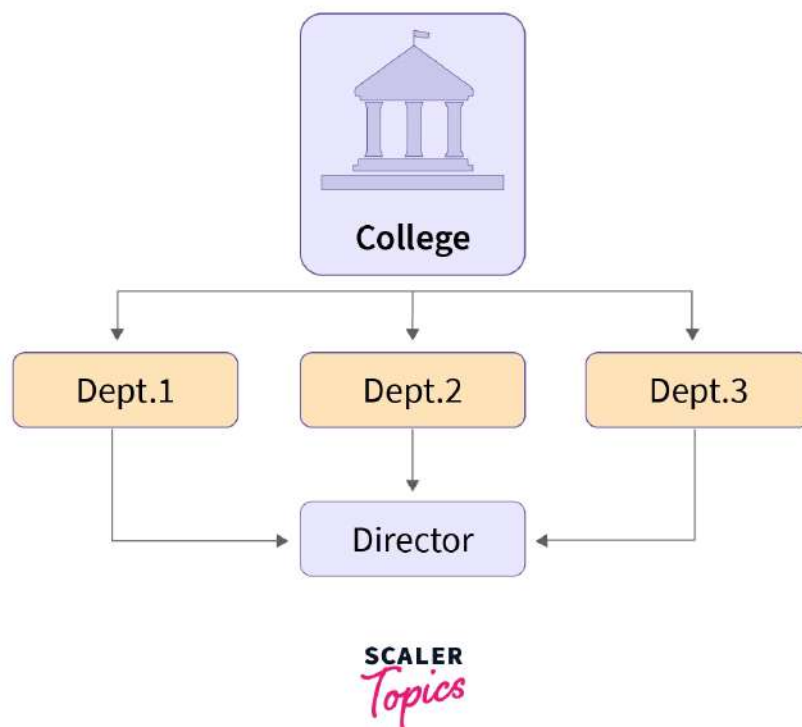
Hierarchical Model

- Organizes data into a tree-like structure with a single root.
- Each child node has only one parent node.
- Efficiently represents one-to-many relationships (e.g., index of a book, recipes).



Network Model

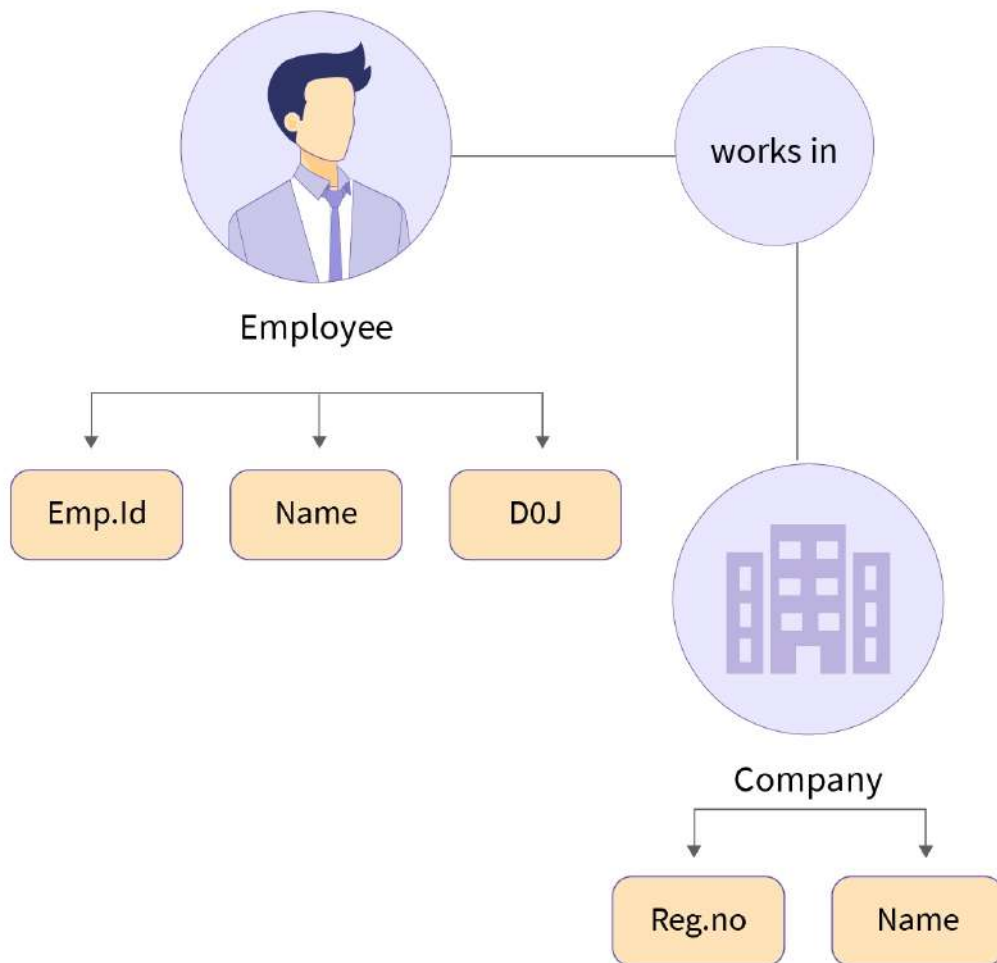
- An extension of the Hierarchical model.
- Organizes data more like a graph, allowing nodes to have more than one parent.
- Establishes multiple relationships, facilitating easier and faster data access.



Entity-relationship Model (E-R Model)

- Divides objects of interest into entities and attributes.
- Establishes relationships between entities using relationships.
- Represents relationships in pictorial form for better understanding.

- Example: A School Database with entities like Student and Address, and their attributes.

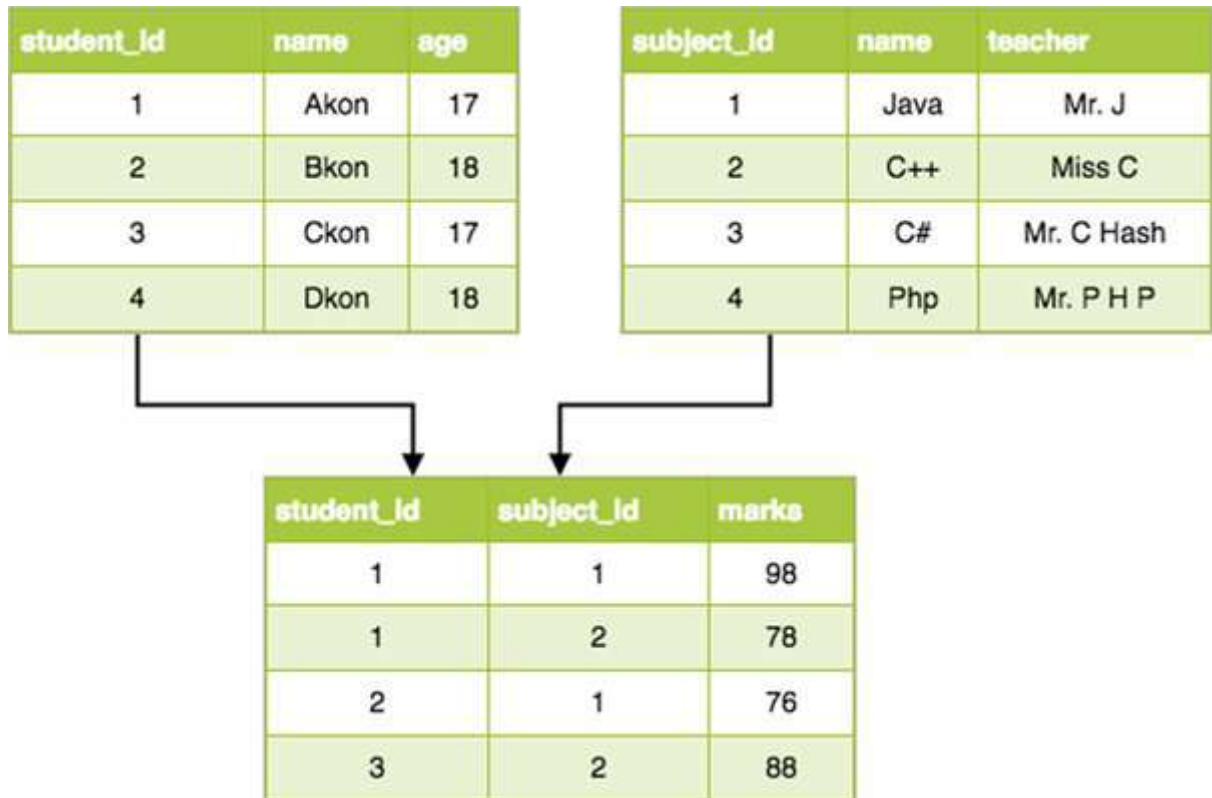


SCALER
Topics

Relational Model

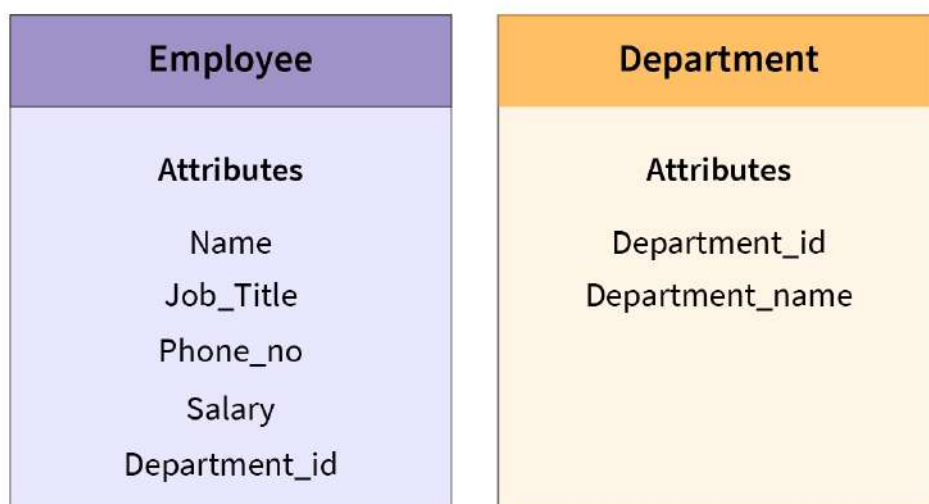
- Organizes data in two-dimensional tables.
- Maintains relationships by storing a common field.
- Introduced by E.F. Codd in 1970, it's the most widely used model globally.

- Tables are also known as relations in this model.



Object-Oriented Data Model

- Combines object-oriented programming and relational data model.
- Represents data and relationships in a single structure called an object.
- Facilitates storage of complex data types like audio, video, images.



Levels of Abstraction in a DBMS

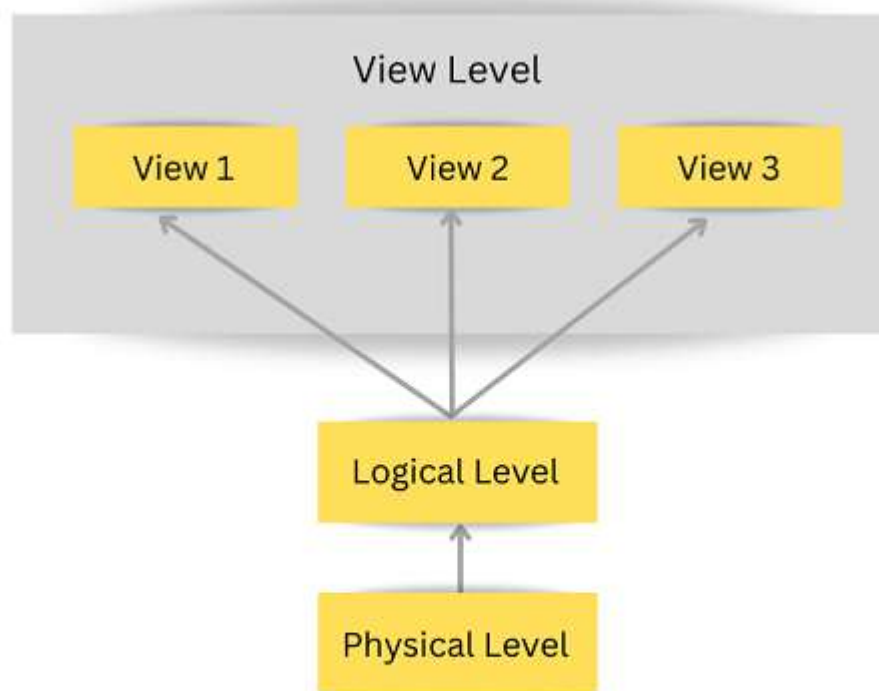
Data abstraction in DBMS involves hiding unimportant database details from end users to facilitate easy and secure data access.

Purpose

- Hide irrelevant data from users and provide a focused view of relevant data.
- Simplify data access for users, improving system efficiency.

Example

- Analogy: Turning on a fan without needing to understand electrical connections.
- Data abstraction hides background details, allowing users to perform tasks without complexities.



Levels of Data Abstraction in DBMS

Levels of Data Abstractions

1. Physical or Internal Level:

- Defines how data is stored in the database.
- Complex data structures are detailed at this level.
- Managed by Data Administrators (DBAs) who handle data storage and organization.

2. Logical or Conceptual Level:

- Describes stored data and their relationships.
- Represents data structure using tables.
- Less complex than the physical level.
- Helps DBAs abstract data from the physical level.

3. **View or External Level:**

- Defines different views of the overall database data.
- Allows end-user interaction.
- Users access data based on queries.

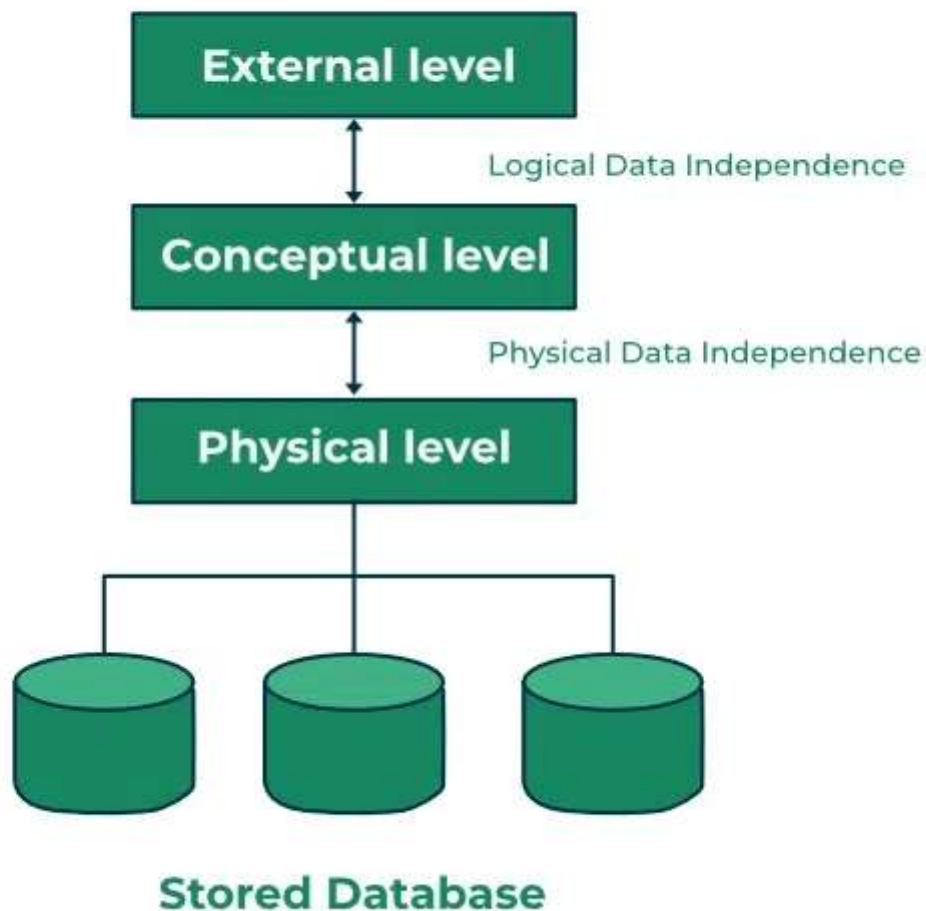
Advantages

- Easy data access based on user queries.
 - Enhanced data security.
 - Improved database system efficiency.
-

Data Independence

Data independence in DBMS allows altering the schema of a database at one level without affecting the schema definition at the next higher level, improving data quality and minimizing the need for modifications in application programs.

Data Independence in DBMS



Types of Data Independence

1. Logical Data Independence

- Alters the schema at the conceptual level without affecting the external or user view.
- Changes in the logical structure of the database necessitate modifications at the logical level.
- Examples:
 - Breaking existing records into multiple records.
 - Combining multiple records into one.
 - Modifying, deleting, or inserting attributes.

2. Physical Data Independence

- Alters the schema at the internal or physical level without affecting the conceptual level.
- Changes in storage size or physical structure do not affect the conceptual view.

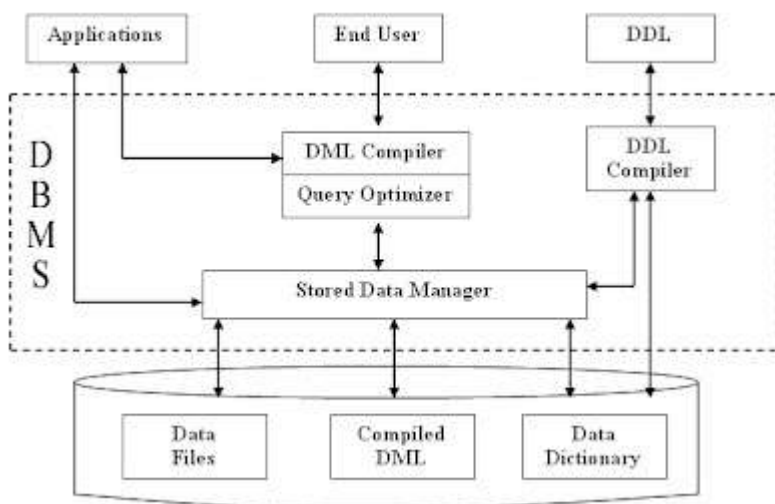
- Achievable more easily compared to logical data independence.
- Examples:
 - Modifying indexes in the database system.
 - Modifying the data structure used for storage.

Data independence enhances flexibility and maintenance of database systems by allowing changes at different levels without disrupting higher-level schemas or application programs.

Structure of a DBMS

DBMS (Database Management System) serves as an interface between users and databases, facilitating various operations like insertion, deletion, updating, and retrieval of data. It comprises three essential components: Query Processor, Storage Manager, and Disk Storage, each playing a crucial role in managing databases effectively.

Components of DBMS Structure



1. Query Processor

Responsible for executing user queries, the query processor translates application program-provided requests into machine-readable instructions. Its components include:

- **DDL Interpreter:** Interprets Data Definition Language (DDL) statements for schema definitions.
- **DML Compiler:** Converts Data Manipulation Language (DML) statements into executable code and optimizes queries.
- **Embedded DML Pre-compiler:** Converts embedded DML commands in application programs into standard procedural calls.
- **Query Optimizer:** Analyzes and executes query evaluation plans efficiently.

2. Storage Manager

Acts as a bridge between queries and database data, maintaining consistency and integrity. Its components include:

- **Integrity Manager:** Manages integrity constraints in the database.
- **Authorization Manager:** Verifies user authentication for specific queries or requests.
- **File Manager:** Manages database files and data structures.
- **Transaction Manager:** Ensures database consistency before and after transactions and controls concurrent processes.
- **Buffer Manager:** Transfers data between primary and main memory and manages cache memory.

3. Disk Storage

Utilizes various data structures for physical system implementation. Its components include:

- **Data Dictionary:** Stores metadata about database objects' structures.
- **Data Files:** Stores data in files.
- **Indices:** Enables fast and efficient data access and retrieval.

The structured organization of DBMS components ensures efficient data management and access, providing users with seamless interaction with databases.

Characteristics of DBMS:

1. Support for Multiple Views of Data:

- A database supports multiple views of data, allowing different users or groups of users to access subsets of the database tailored to their needs.

2. Sharing of Data and Multiuser System:

- Modern database systems support multiple users accessing the same database simultaneously, facilitated by concurrency control strategies to maintain data integrity.

3. Control of Data Redundancy:

- DBMS aims to minimize data redundancy by storing each data item in only one place, although some redundancy may be introduced for performance reasons, controlled by the application programming.

4. Data Sharing:

- Integration of data within a database system allows for easier data sharing among employees and users, enabling the generation of more information from the available data.

5. Enforcement of Integrity Constraints:

- DBMS must enforce integrity constraints to ensure data validity and integrity, including constraints related to data type, uniqueness, and other rules defined by

the user.

6. **Restriction of Unauthorized Access:**

- Security subsystems in a DBMS restrict unauthorized access by defining different user accounts with varying levels of access privileges, such as read-only or read-write access.

7. **Data Independence:**

- DBMS allows for data independence by separating data descriptions (metadata) from application programs, enabling changes to the data structure without impacting the application programs.

8. **Transaction Processing:**

- DBMS includes concurrency control subsystems to ensure data consistency and validity during transaction processing, even when multiple users are updating the same information simultaneously.

9. **Backup and Recovery Facilities:**

- Database systems provide backup and recovery facilities to protect data from loss due to hardware failures or other issues, ensuring data integrity and availability even in the event of system failures.

10. **Provision for Multiple Views of Data:**

- DBMS inherently permits multiple users to access its database simultaneously, with users being unaware of the underlying storage and organization of data.

Introduction to Database Design

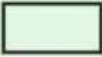





Database Design and ER Diagrams

Database Design

Database Design in DBMS involves a series of processes aimed at creating efficient and effective systems for managing enterprise data. The main objectives of database design are to produce logical and physical models of the proposed database system, focusing on data requirements, storage, and organization. One common tool used in database design is the Entity-Relationship (ER) Diagram, which visually represents the relationships between entities stored in the database. This diagram helps to explain the logical structure of the database and is based on the concepts of entities, attributes, and relationships.

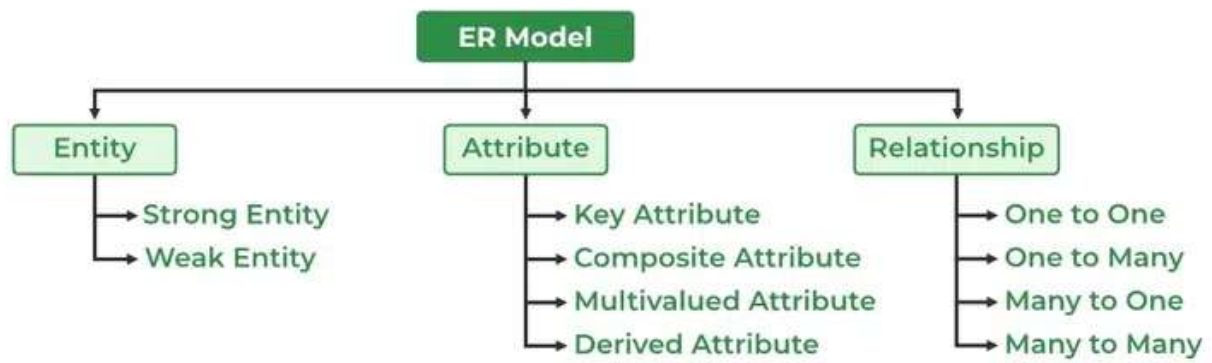
Entity-Relationship (ER) Diagram:

- **Entities:** Represented by rectangles, entities are the objects or concepts about which data is stored in the database. Examples of entities include customers, products, employees, etc.
- **Attributes:** Represented by ellipses, attributes are the characteristics or properties of entities. Each entity has a set of attributes that describe it. For example, a customer entity may have attributes such as name, address, and phone number.
- **Relationships:** Represented by diamond shapes, relationships define how entities are related to each other. They indicate how data from one entity is linked to data in another entity. Relationships can be one-to-one, one-to-many, or many-to-many.
- **Lines:** Lines connecting entities and attributes represent the connections or relationships between them. These lines indicate how data flows between different parts of the database.
- **Double Ellipses:** Used to represent multi-valued attributes, which are attributes that can have multiple values for a single entity.
- **Double Rectangles:** Represent weak entities, which are entities that cannot be uniquely identified by their attributes alone and depend on a related entity for identification.

Figures	Symbols	Represents
Rectangle		Entities in ER Model
Ellipse		Attributes in ER Model
Diamond		Relationships among Entities
Line		Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse		Multi-Valued Attributes
Double Rectangle		Weak Entity

Entities, Attributes, and Entity Sets

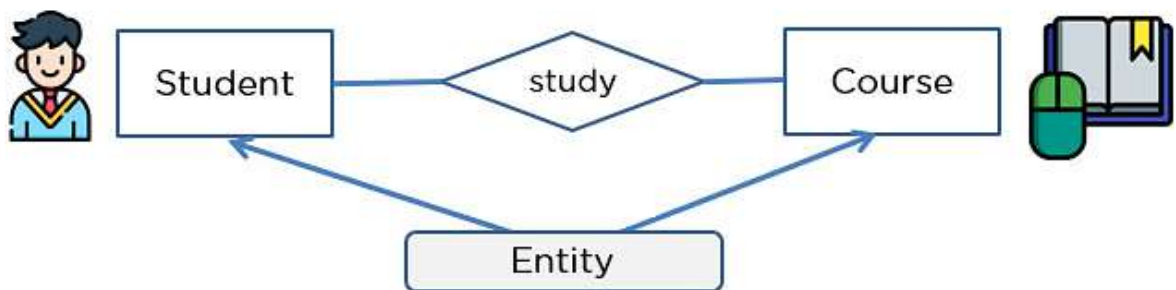
Components of an ER Diagram:



<https://www.simplilearn.com/tutorials/sql-tutorial/er-diagram-in-dbms>

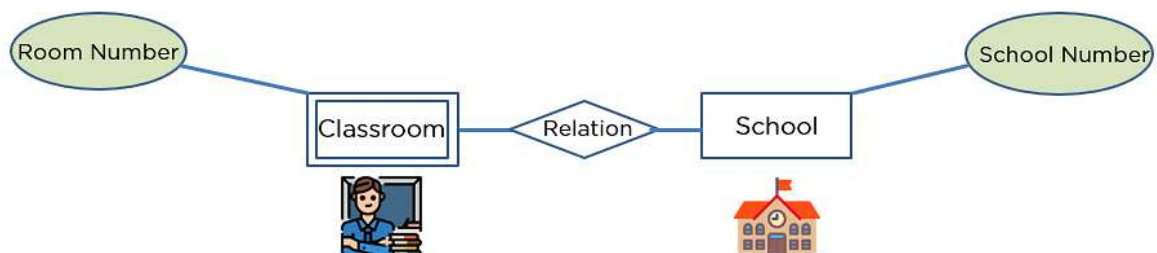
1. Entities:

- An entity represents a real-world object or concept that can be identified and distinguished from other objects. It is depicted as a rectangle in an ER diagram.
- Example: In a university database, entities could include Student, Course, Professor, and Department.



2. Weak Entity:

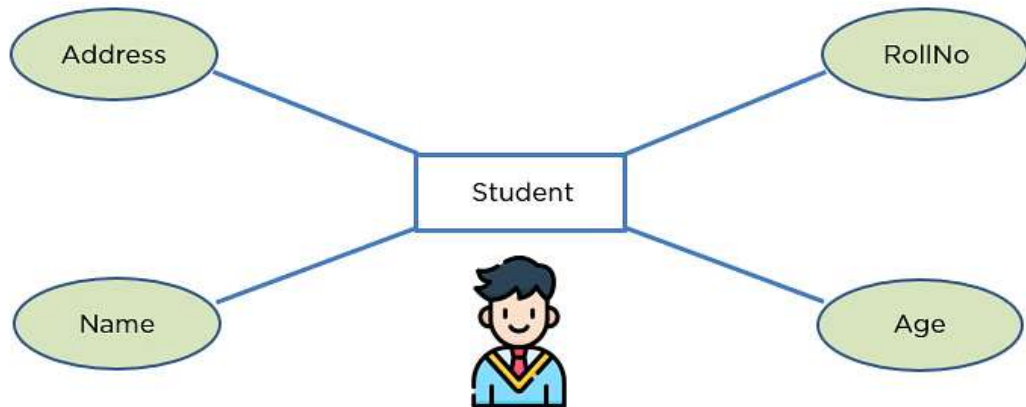
- A weak entity is an entity that cannot be uniquely identified by its attributes alone and relies on a related entity for identification. It is represented as a double rectangle in an ER diagram.
- Example: In a library database, a BookCopy entity might be weak because it depends on the Book entity for identification.



3. Attributes:

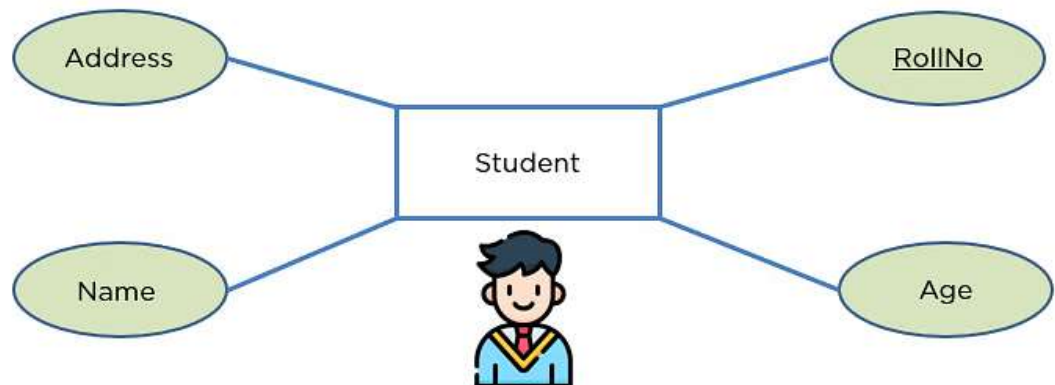
- Attributes are the properties or characteristics of entities. They describe the entity's properties.
- Attributes are depicted as ovals connected to their respective entity rectangles.

- Example: Attributes of a Student entity might include StudentID, Name, DateOfBirth, etc.



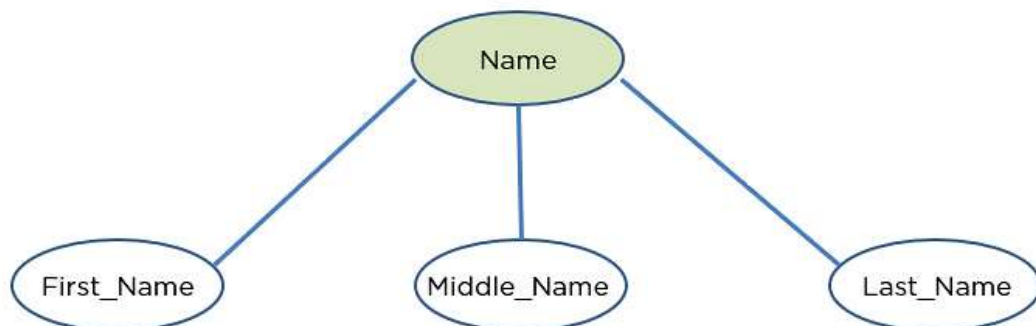
4. Key Attribute:

- A key attribute uniquely identifies an entity within an entity set. It is underlined in an ER diagram.
- Example: In a Customer entity, the CustomerID might be a key attribute.



5. Composite Attribute:

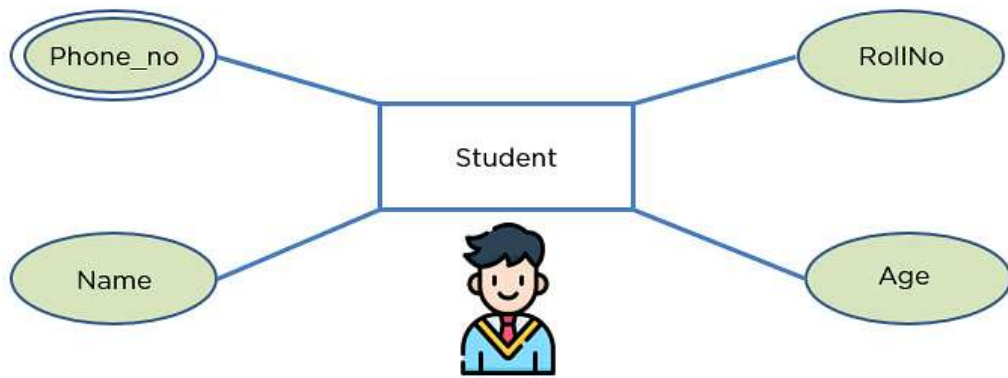
- A composite attribute is an attribute composed of multiple simpler attributes. It is represented as an oval connected to other ovals.
- Example: An Address attribute might be composed of Street, City, State, and Zip Code.



6. Multivalued Attribute:

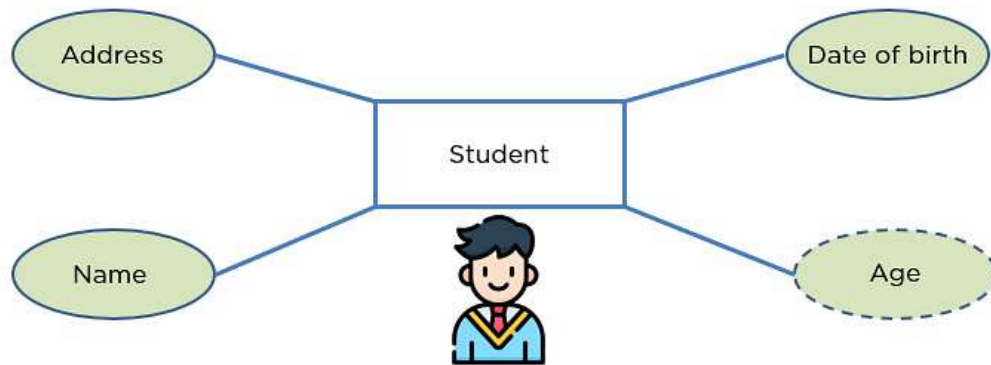
- A multivalued attribute can have multiple values for a single entity. It is represented as a double oval.

- Example: In a Product entity, the Color attribute might be multivalued if a product can come in multiple colors.



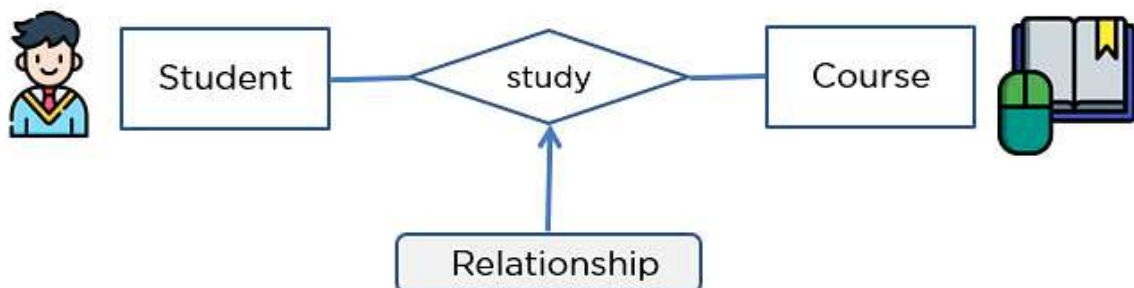
7. Derived Attribute:

- A derived attribute is an attribute whose value can be derived from other attributes of the entity. It is represented by a dashed oval.
- Example: Age can be a derived attribute if it is calculated based on the DateOfBirth attribute.



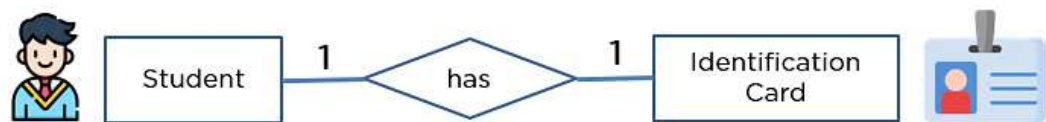
Relationships and Relationship Sets

Relationships in an ER Diagram:



1. One-to-One Relationship:

- In a one-to-one relationship, each record in the first entity (A) is associated with exactly one record in the second entity (B), and vice versa.
- Represented by a line connecting the two entities, with the cardinality notation "1" on one side of the line and "1" on the other side.
- Example: A Student entity may be associated with exactly one Address entity, and vice versa.



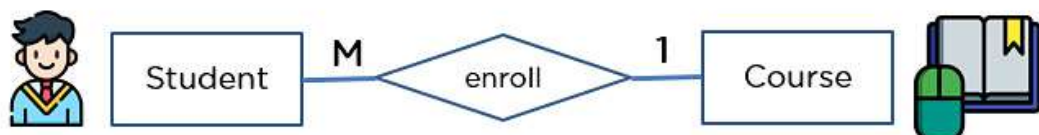
2. One-to-Many Relationship:

- In a one-to-many relationship, each record in the first entity (A) can be associated with many records in the second entity (B), but each record in the second entity (B) is associated with at most one record in the first entity (A).
- Represented by a line connecting the two entities, with the cardinality notation "1" on one side of the line and "N" (or "M") on the other side.
- Example: A Department entity may have many Employees, but each Employee is associated with only one Department.



3. Many-to-One Relationship:

- In a many-to-one relationship, many records in the first entity (A) can be associated with one record in the second entity (B), but each record in the second entity (B) is associated with at most one record in the first entity (A).
- Represented by a line connecting the two entities, with the cardinality notation "N" (or "M") on one side of the line and "1" on the other side.
- Example: Many Orders can be placed by one Customer, but each Order is associated with only one Customer.



4. Many-to-Many Relationship:

- In a many-to-many relationship, each record in the first entity (A) can be associated with many records in the second entity (B), and vice versa.
- Represented by a line connecting the two entities, with the cardinality notation "N" (or "M") on both sides of the line.

- Example: Many Students can enroll in many Courses, and each Course can have many Students.



Additional Features of the ER Model

Features of ER Model:

1. ER Diagram:

- ER diagrams illustrate the logical structure of databases by depicting entities, attributes, and relationships between them.

2. Database Design:

- The Entity-Relationship model facilitates the design of databases in a simple and conceptual manner, aiding database designers in building effective database structures.

3. Graphical Representation for Better Understanding:

- ER diagrams provide a graphical representation that is easy to understand, enabling developers to communicate database designs effectively with stakeholders.

4. Easy to Build:

- ER models are straightforward to construct, making them accessible for database designers of varying levels of expertise.

5. Extended E-R Features:

- The ER model includes additional features such as specialization, inheritance, aggregation, and generalization, which enhance its expressive power and flexibility.

6. Integration:

- The ER model can be seamlessly integrated into a common relational model, serving as a valuable tool for database designers in communicating and implementing database structures.

7. Simplicity and Versatility:

- ER diagrams provide a blueprint for implementing data in specific software applications, offering a preview of table connections and fields, which contributes to their simplicity and versatility in various applications.
-

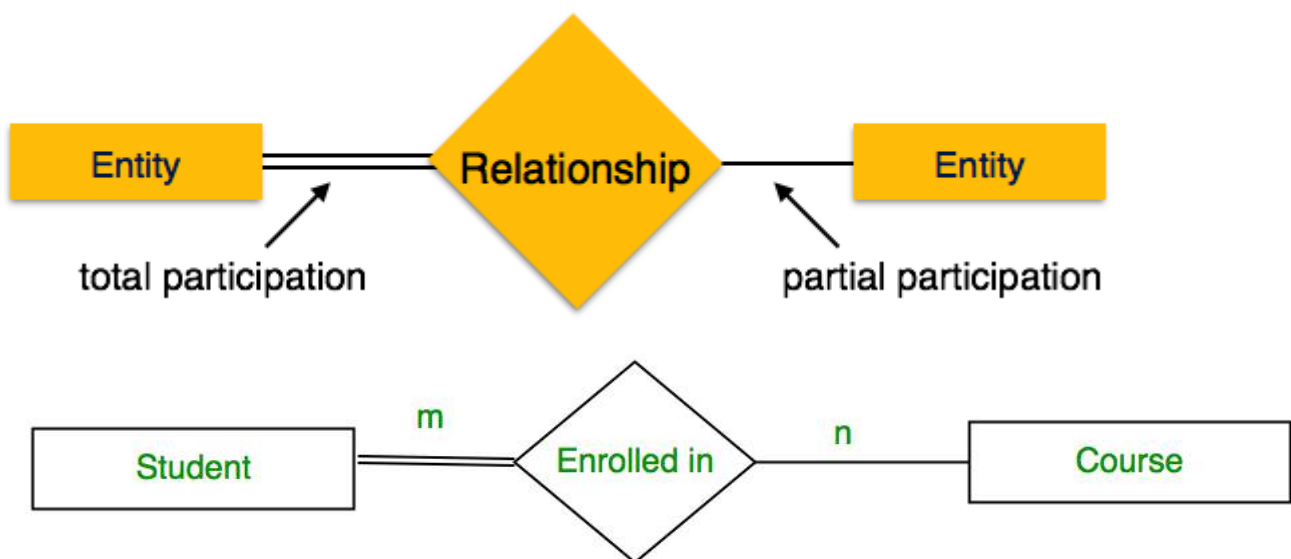
Participation Constraint in ER Diagrams:

1. Total Participation:

- In total participation, each entity in the entity set must participate in the relationship. This means that every instance of the entity set must be associated with at least one instance of the related entity set.
- Represented by a double line connecting the entity to the relationship in the ER diagram.
- Example: If every student must be enrolled in at least one course, then the participation of students in the "Enrolled in" relationship is total.

2. Partial Participation:

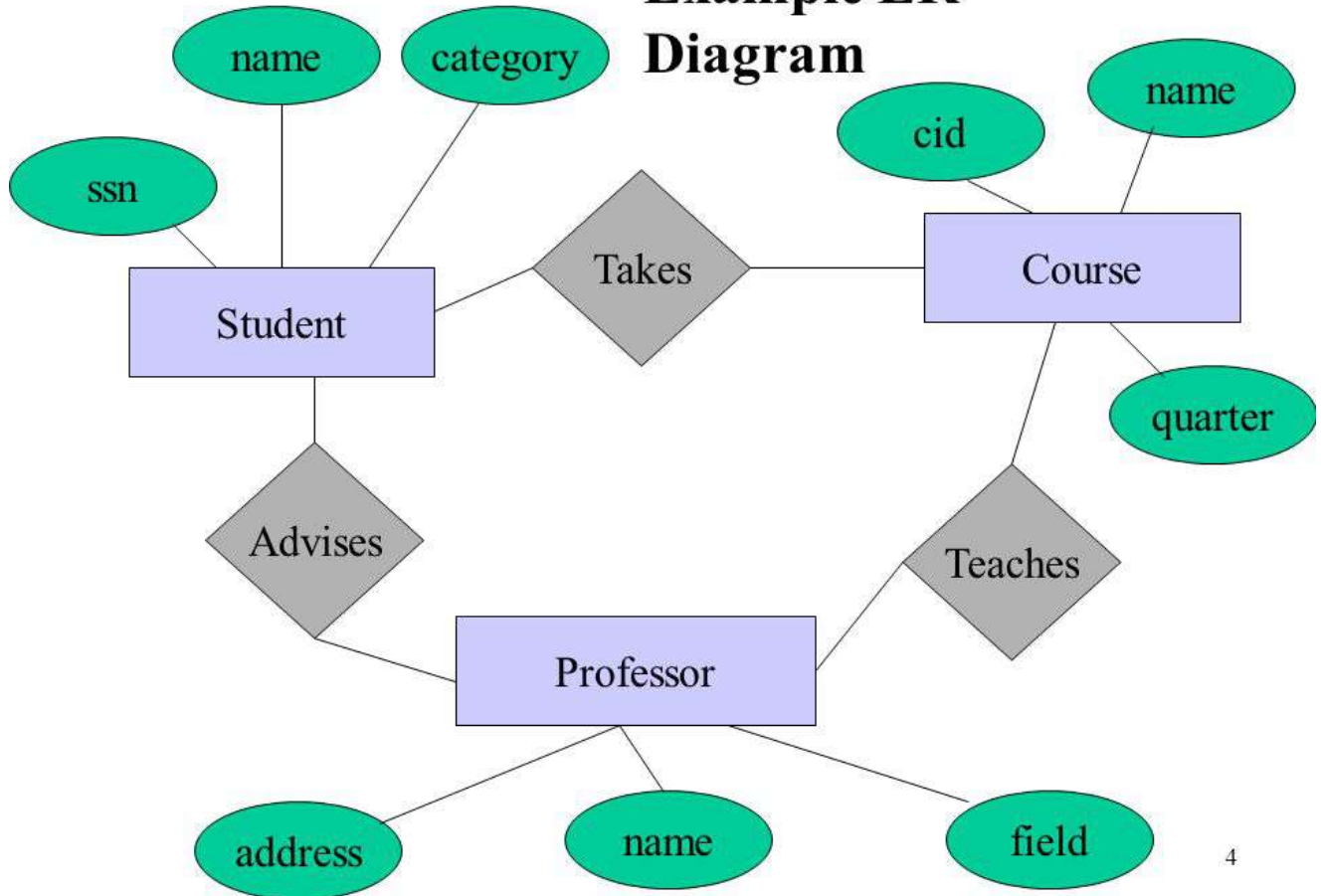
- In partial participation, entities in the entity set may or may not participate in the relationship. This means that some instances of the entity set may not be associated with any instances of the related entity set.
- Represented by a single line connecting the entity to the relationship in the ER diagram.
- Example: If some courses may not have any students enrolled, then the participation of courses in the "Enrolled in" relationship is partial.



Conceptual Design With the ER Model

Conceptual design using the Entity-Relationship (ER) model involves creating a high-level representation of the data requirements and relationships in the system.

Example ER Diagram



4

1. **Identify Entities:** Determine main objects or concepts to be stored as entities (e.g., Student, Course).
 2. **Define Attributes:** Specify properties for each entity (e.g., StudentID, Name for Student).
 3. **Establish Relationships:** Identify connections between entities (e.g., Enrollment between Student and Course).
 4. **Refine Cardinality and Participation:** Determine how many instances of one entity relate to another and if participation is mandatory or optional.
 5. **Validate the Model:** Review to ensure accuracy and alignment with requirements.
 6. **Iterate and Refine:** Make adjustments as needed based on feedback and changes.
 7. **Document the ER Model:** Create diagrams and descriptions to communicate the database structure.
-