A Study on Database Management Systems

Abstract

This article provides a comprehensive review of the current state-of-the-art database management systems (DBMS). The review covers the different types of DBMS, such as relational, object-oriented, NoSQL, and cloud-based systems etc. It analyzes the Merits and Demerits ,structure, Database Language and Types, Concept of ER Diagram.The article concludes with future research directions that can enhance the efficiency and reliability of Data management systems.

Keyword

Merits and Demerits, Structure, Database Architecture and Independences, Database Languages and Types, Data models and ER Diagram.

Date

Submitted on April 26, 2023

Introduction

DBMS short for database management system plays a major role in most real-world projects that require storing, retrieving, and querying digital data. For instance, dynamic websites, accounting information systems, payroll systems, stock management systems all rely on internal databases as a container to store and manage their data . Database management system (database management system-DBMS) is software that makes it easy for organizations to centralize data, manage data efficiently, and provide data access for all the application programs. DBMS acts as an interface between application programs and physical data files. When an application program calls a data file, such as gross salary, the DBMS searches this data in the database and gives it to the application program. (Susanto, JUNE 2019)

MERITS OF A DBMS

Data independence:

Application programs should be as independent as possible from details of data representation and storage. The DBMS can provide an abstract view of the data to insulate application code from such details.

Efficient data access:

A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.

Data integrity and security:

If data is always accessed through the DBMS, the DBMS can enforce integrity constraints on the data. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, the DBMS can enforce access controls that govern what data is visible to different classes of users.

Data administration:

When several users share the data, centralizing the administration of data can offer significant improvements. Experienced professionals who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation.

Concurrent access and crash recovery:

A DBMS schedules current accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures. (Singh, January – March 2015)

DEMERITS OF A DBMS

Complexity: DBMS can be complex and difficult to learn, implement, and maintain, especially for non-technical users**.**

Cost: DBMS can be expensive, requiring significant hardware and software resources, as well as skilled personnel to manage and operate**.**

Security Risks: DBMS can be vulnerable to security risks such as hacking, unauthorized access, and data breaches. Database administrators need to ensure proper security measures are in place to protect sensitive data**.**

Performance Issues: DBMS can sometimes have performance issues, such as slow response times, especially when dealing with large amounts of data.

Data Corruption: DBMS can sometimes be prone to data corruption, which can lead to data loss or errors in data processing. (Singh, January – March 2015)

Structure of DBMS

DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations (insert, delete, update and retrieval) on the database. The components of DBMS perform these requested operations on the database and provide necessary data to the users.

....

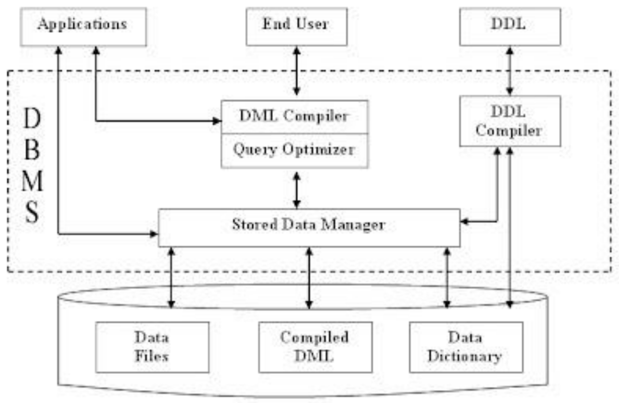


Fig:- Structure of DBMS

The structure of a Database Management System (DBMS) can be divided into three main components: the Internal Level, the Conceptual Level, and the External Level.

Internal Level:

This level represents the physical storage of data in the database. It is responsible for storing and retrieving data from the storage devices, such as hard drives or solid-state drives. It deals with low-level implementation details such as data compression, indexing, and storage allocation.

Conceptual Level:

This level represents the logical view of the database. It deals with the overall organization of data in the database and the relationships between them. It defines the data schema, which includes tables, attributes, and their relationships. The conceptual level is independent of any specific DBMS and can be implemented using different DBMSs.

External Level:

This level represents the user's view of the database. It deals with how users access the data in the database. It allows users to view data in a way that makes sense to them, without worrying about the underlying implementation details. The external level provides a set of views or interfaces to the database, which are tailored to meet the needs of specific user groups. (Sr)

DBMS Architecture and Data Independence

The Three-Schema Architecture

The goal of the three-schema architecture, illustrated in Figure 1, is to separate the user applications and the physical database. In this architecture, schemas can be defined at the following three levels:

The internal level

An internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

The conceptual level

A conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. A high-level data model or an implementation data model can be used at this level.

The external or view level

A number of external schemas or user views Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. A high-level data model or an implementation data model can be used at this level.

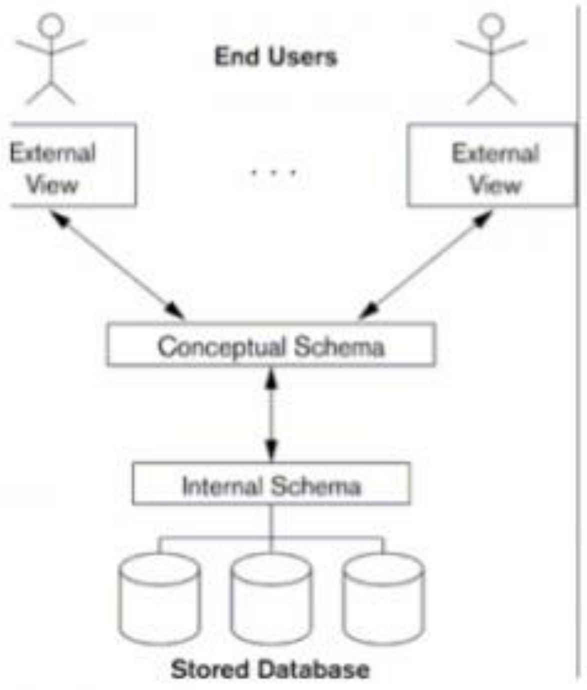


Figure 1 The Three Schema Architecture

The three-schema architecture is a convenient tool for the user to visualize the schema levels in a database system. Most DBMSs do not separate the three levels completely, but support the three-schema architecture to some extent. Some DBMSs may include physical-level details in the conceptual schema. In most DBMSs that support user views, external schemas are specified in the same data model that describes the conceptual-level information. Some DBMSs allow different data models to be used at the conceptual and external levels.

These mappings may be time-consuming, so some DBMSs—especially those that are meant to support small databases—do not support external views. Even in such systems, however, a certain amount of mapping is necessary to transform requests between the conceptual and internal levels. (krishna, 2020)

Data Independence

The three-schema architecture can be used to explain the concept of data independence, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. We can define two types of data independence:

Logical data independence

It is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), or to reduce the database (by removing a record type or data item). In the latter case, external schemas that refer only to the remaining data should not be affected. Only the view definition and the mappings need be changed in a DBMS that supports logical data independence. Application programs that reference the external schema constructs must work as before, after the conceptual schema undergoes a logical reorganization. Changes to constraints can be applied also to the conceptual schema without affecting the external schemas or application programs.

Physical data independence

It is the capacity to change the internal schema without having to change the conceptual (or external) schemas. Changes to the internal schema may be needed because some physical files had to be reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update. If the same data as before remains in the database, we should not have to change the conceptual schema.

Whenever we have a multiple-level DBMS, its catalog must be expanded to include information on how to map requests and data among the various levels. The DBMS uses additional software to accomplish these mappings by referring to the mapping information in the catalog. Data independence is accomplished because, when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed. Hence, application programs referring to the higher-level schema need not be changed.

The three-schema architecture can make it easier to achieve true data independence, both physical and logical. However, the two levels of mappings create an overhead during compilation or execution of a query or program, leading to inefficiencies in the DBMS. Because of this, few DBMSs have implemented the full three-schema architecture. (krishna, 2020)

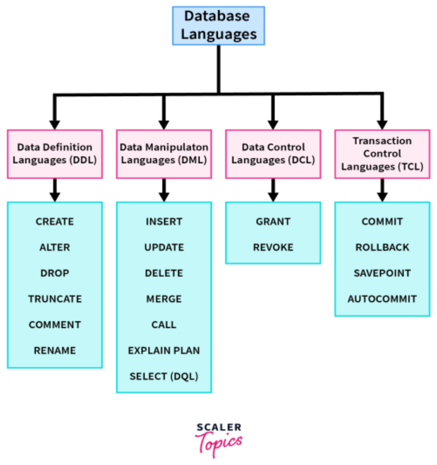
Database Language in DBMS

Database Language is a special type of programming language used to define and manipulate a database. Based on their application, database languages are classified into four different types: DDL, DML, DCL, and TCL.

In the data world, we need some special kind of programming languages to make the DBMS software understand our needs and manage the data stored in the databases accordingly. These programming languages are known as database languages or query languages. Database languages are used to perform a variety of critical tasks that help a database management system function correctly. These tasks can be certain operations such as read, update, insert, search, or delete the data stored in the database. (Gulati, 2022)

Types of database languages

The Data Languages are categorized into four different types based upon the various operations performed by the language. These include:



Data Definition Language (DDL)

Data Definition Language (DDL) is a set of special commands that allows us to define and modify the structure and the metadata of the database. These commands can be used to create, modify, and delete the database structures such as schema, tables, indexes, etc. Since DDL commands can alter the structure of the whole database and every change implemented by a DDL command is auto-committed (the change is saved permanently in the database), these commands are normally not used by an end-user (someone who is accessing the database via an application).

Data Manipulation Language (DML)

Data Manipulation Language (DML) is a set of special commands that allows us to access and manipulate data stored in existing schema objects. These commands are used to perform certain operations such as insertion, deletion, updation, and retrieval of the data from the database. These commands deal with the user requests as they are responsible for all types of data modification. The DML commands that deal with the retrieval of the data are known as Data Query language.

Data Control Language (DCL)

Data Control Language (DCL) is a set of special commands that are used to control the user privileges in the database system. The user privileges include ALL, CREATE, SELECT, INSERT, UPDATE, DELETE, EXECUTE, etc. We require data access permissions to execute any command or query in the database system. This user access is controlled using the DCL statements. These statements are used to grant and revoke user access to data or the database.

Transaction Control Language (TCL)

Transaction Control Language (TCL) is a set of special commands that deal with the transactions within the database. A transaction is a collection of related tasks that are treated as a single execution unit by the DBMS software. Hence, transactions are responsible for the execution of different tasks within a database.

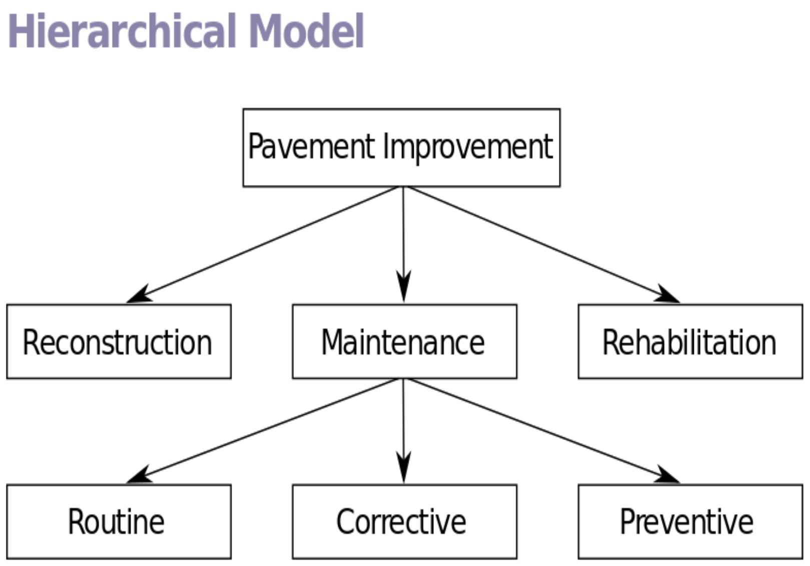
The modifications performed using the DML commands are executed or roll backed with the help of TCL commands. (Sr)

Data Models

Data Model is the modeling of the data description, data semantics, and consistency constraints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction. Therefore, there are following data models used for understanding the structure of the database:

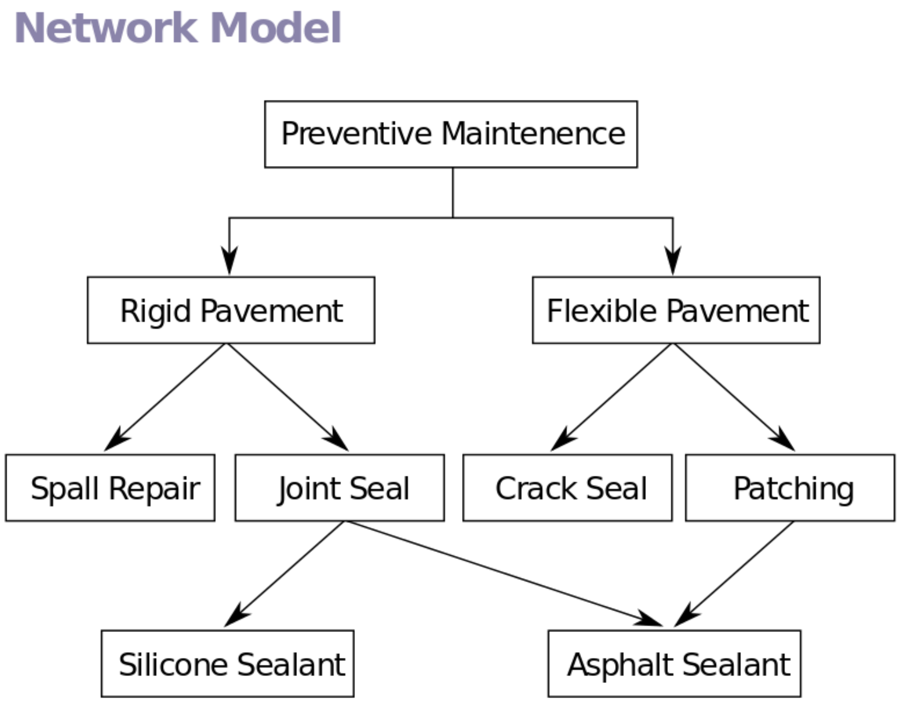
The Hierarchical Model

Hierarchical model redirects here. For the statistics usage, see hierarchical linear modeling. A hierarchical database model is a data model in which the data is organized into a tree-like structure. The structure allows representing information using parent/child relationship). All attributes of a specific record are listed under an entity type.



The Network Model

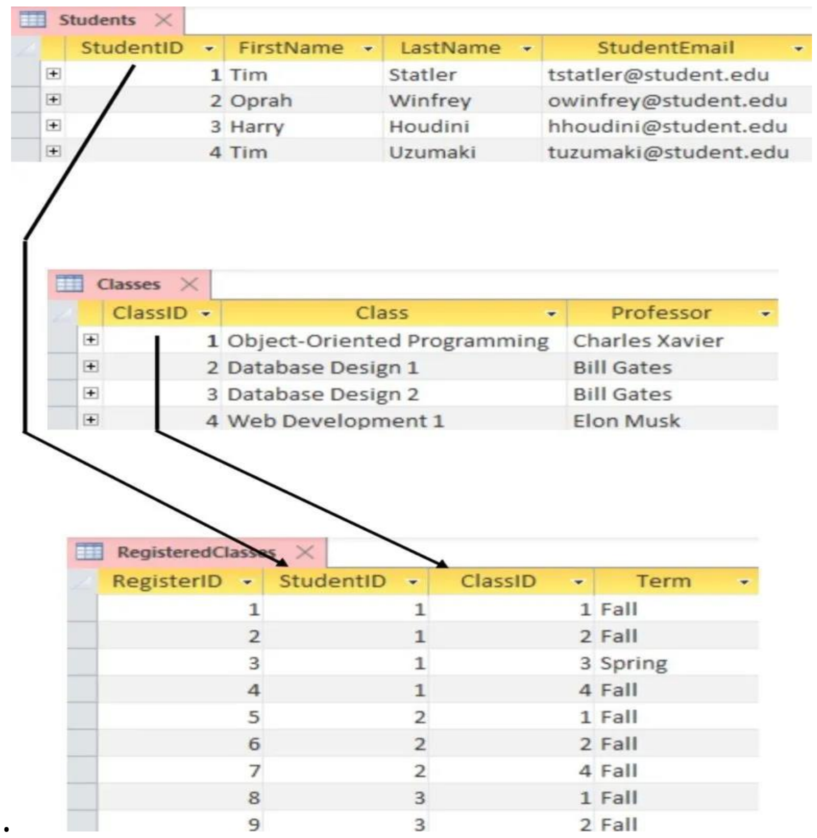
The network model's original inventor was Charles Bachman, and it was developed into a standard specification published in 1969 by the CODASYL Consortium. The network structure consists of more complex relationships. Unlike the hierarchical structure, it can relate to many records and accesses them by following one of several paths. In other words, this structure allows for many-to-many relationships.



Relational Data Model

Relational Data model is the most commonly used model when constructing or redesigning a database. The relational model consists of multiple tables that bear some relationship with each other. Each table contains attributes that are the key that forms these relationships. The relationship between data files is relational, not hierarchical.

For example, consider the tables below to understand relational model:



StudentID	FirstName	LastName	StudentEmail
1	Tim	Statler	tstatler@student.edu
2	Oprah	Winfrey	owinfrey@student.edu
3	Harry	Houdini	hhoudini@student.edu
4	Tim	Uzumaki	tuzumaki@student.edu

ClassID	Class	Professor
1	Object-Oriented Programming	Charles Xavier
2	Database Design 1	Bill Gates
3	Database Design 2	Bill Gates
4	Web Development 1	Elon Musk

RegisterID	StudentID	ClassID	Term
1	1	1	Fall
2	1	2	Fall
3	1	3	Spring
4	1	4	Fall
5	2	1	Fall
6	2	2	Fall
7	2	4	Fall
8	3	1	Fall
9	3	2	Fall

……………….

There is a table for Students that contains attributes for their student ID, their first and last name, and their email.

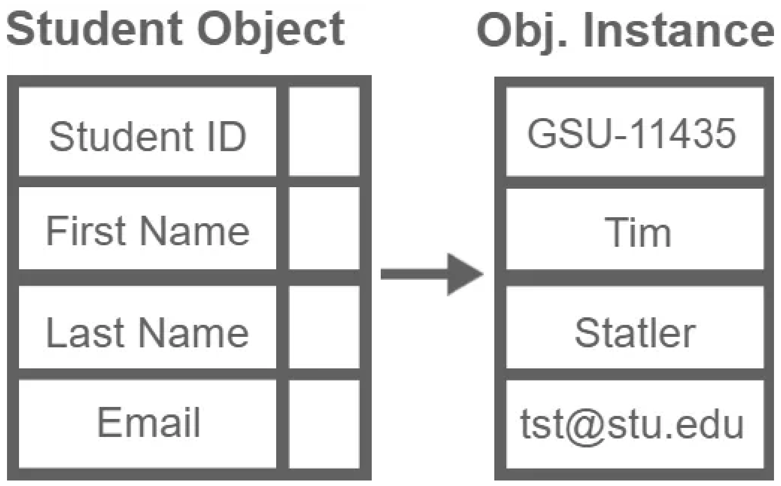
There is another table for Classes that contains attributes for the class ID, the class name, and the professor for the class.

Last, there is a table for the Registered Classes which has a register ID, term, and two attributes taken from the previous two tables: student ID, and class ID. This represents how these tables are related to each other and thus, how relational mode. (Il-Yeol Song, 1995)

Object-Oriented Database Model

The object-oriented database (OODB) model is similar to the relational model in that various tables represent real-life objects. However, similar or object-oriented programming (OOP) instances of objects can be created within the database.

Consider the following example of a student object:



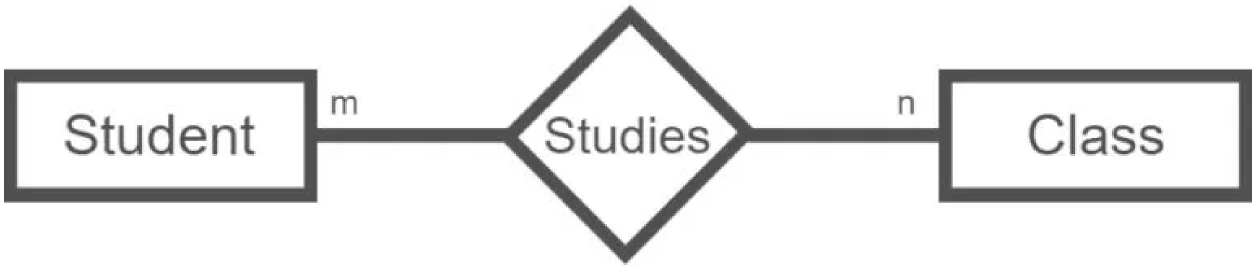
The student object has four attributes, similar to the first example: a student ID, first and last name, and a student email address. This object acts as a template when creating instances of the object. The object instances are digital representation.

Entity-Relationship Model

The entity-relationship database model is similar to the network model because it shows the relationship between two entities. However, the entity-relationship model or more detailed and allows for additional types of relationships, known as cardinality.

To be specific, these models can have one-to-one, one-to-many, or many-to-many relationship types. How the entities are related is also specified in the entity-relationship model.

Let’s look at the following entity-relationship:



This shows two entities, a student and a class. Entities are represented with a rectangle and the type of their relationship is represented with a diamond. The type of relationship will always be between the two entities.

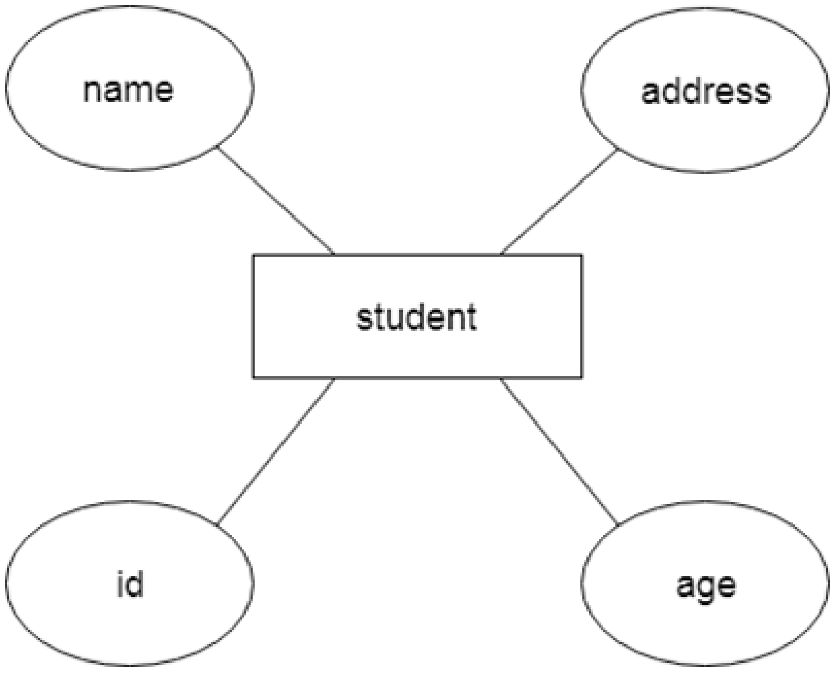
Class and student also have a many-to-many cardinality represented by the ‘m’ and ‘n’ next to the entities. This represents that many students can take many classes at once.

ER DIAGRAM

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.

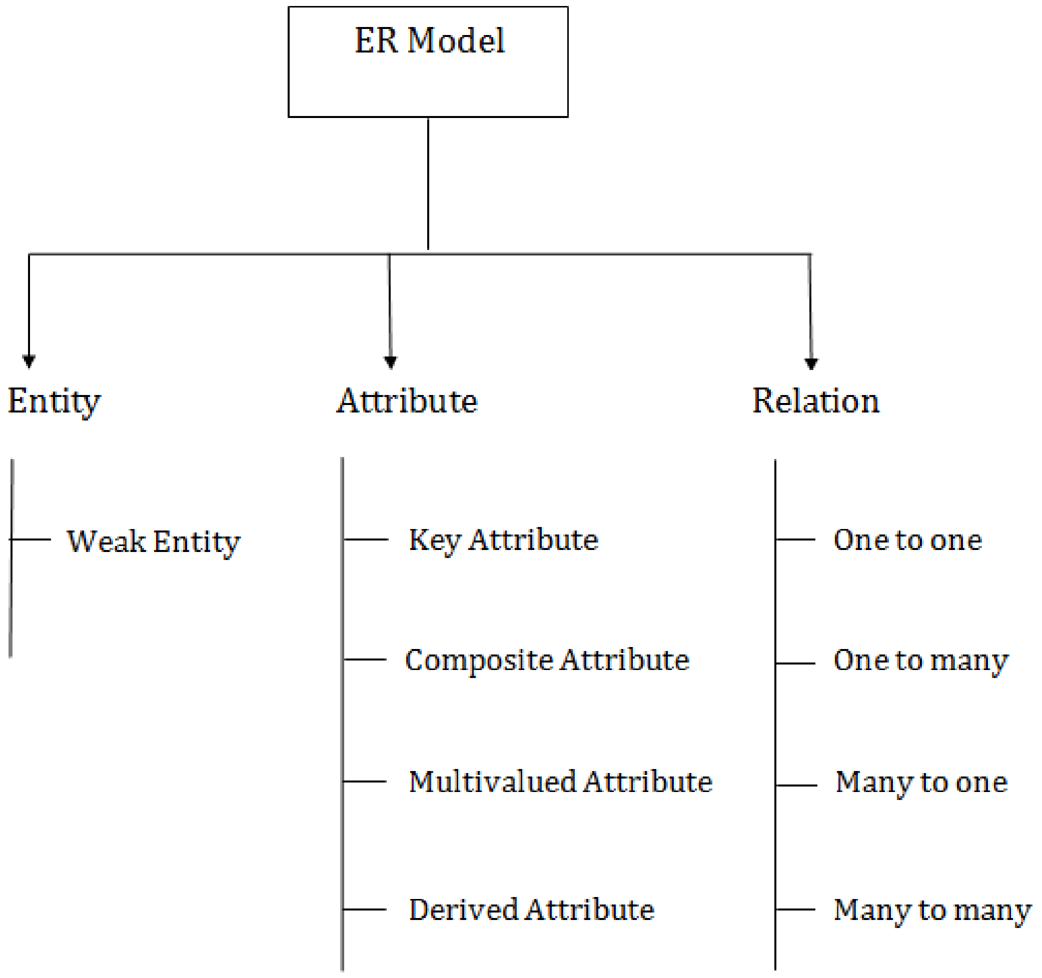
For example:

Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc. and there will be a relationship between them. (Il-Yeol Song, 1995)



....

Components of ER Diagram

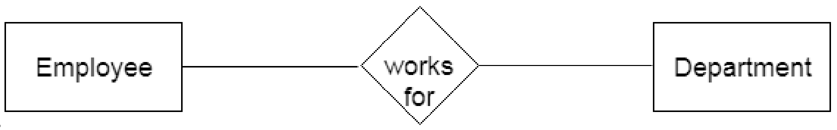


....

Entity

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

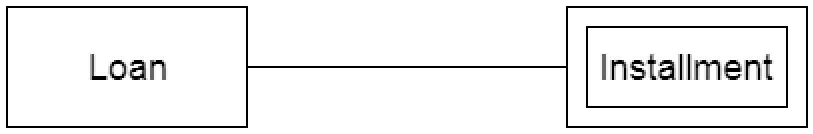
Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



....

Weak Entity

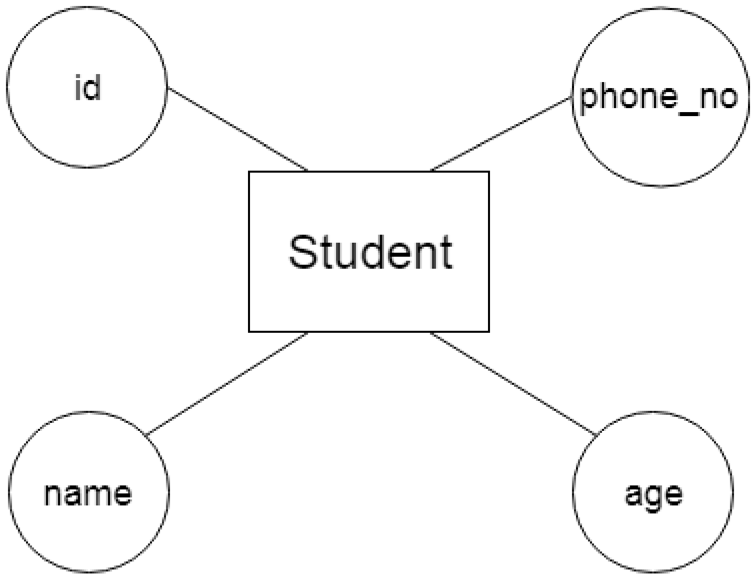
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



Attributes

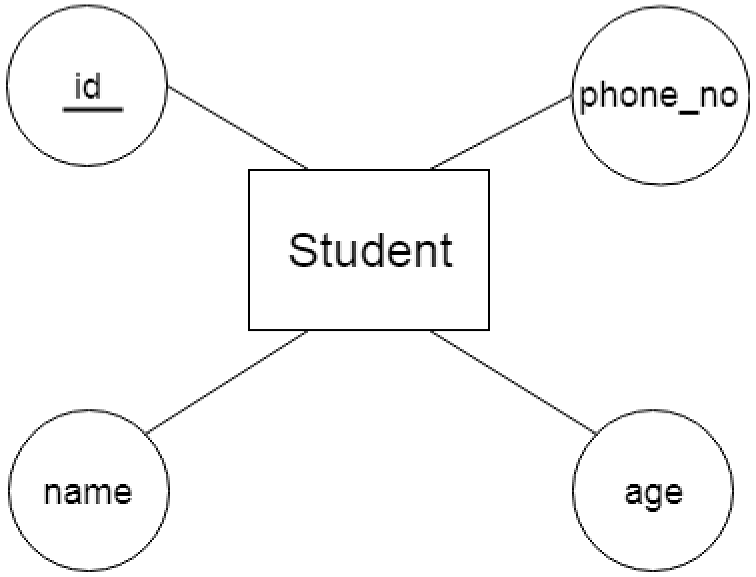
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example**,** id, age, contact number, name, etc. can be attributes of a student.



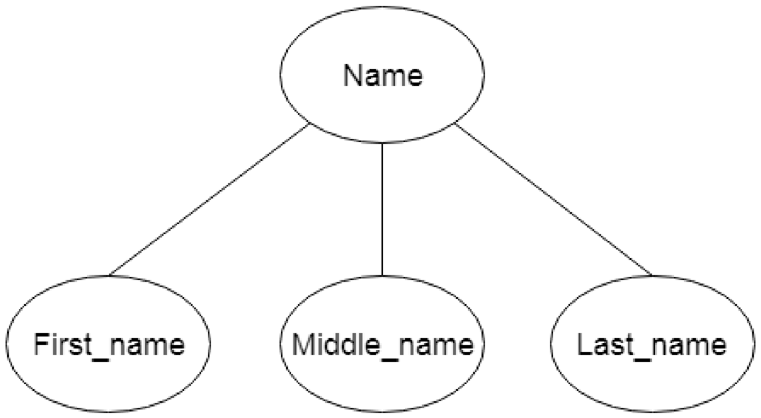
Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



Composite Attribute

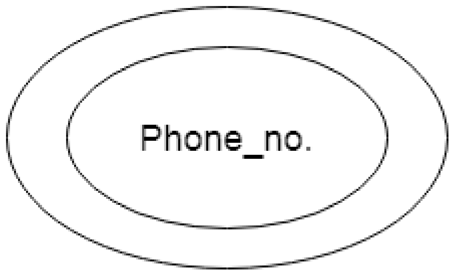
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

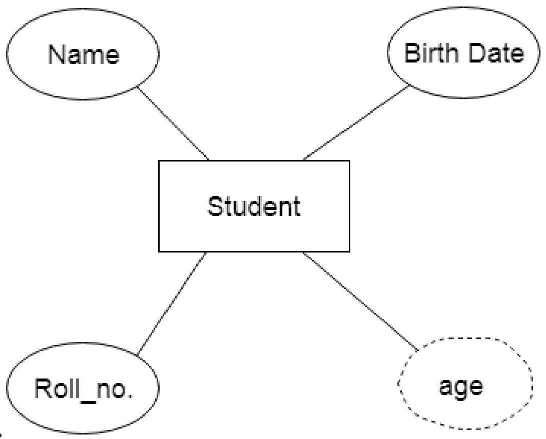
For example, a student can have more than one phone number.



Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

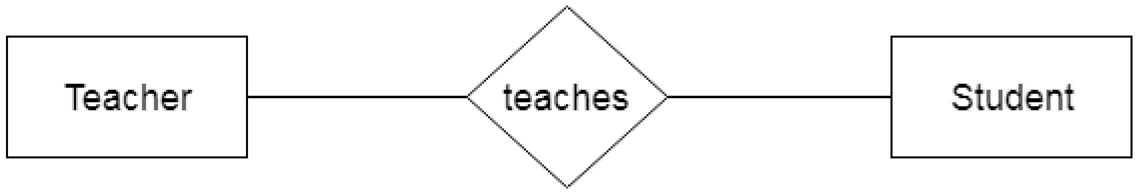
For example, A person's age changes over time and can be derived from another attribute like Date of birth.



....

Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

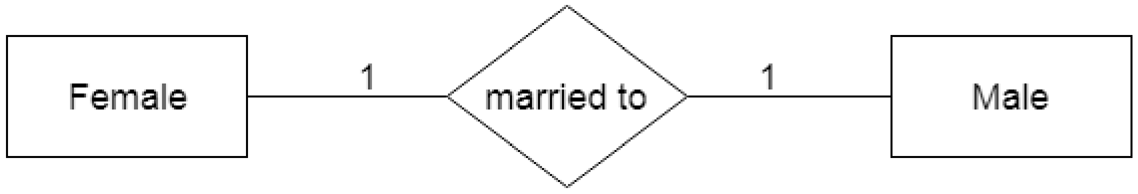


Types of relationship are as follows:

One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

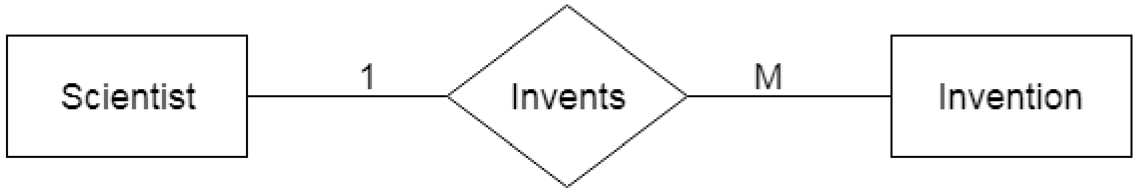
For example, A female can marry to one male, and a male can marry to one female.



One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

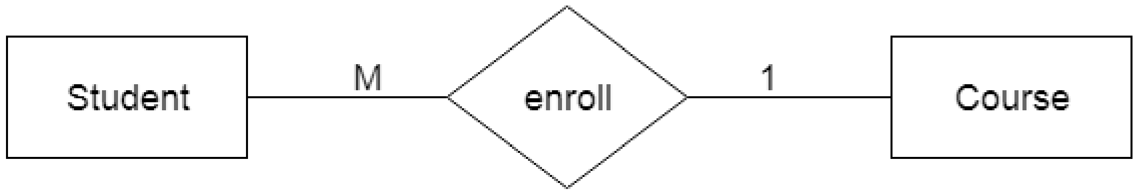
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

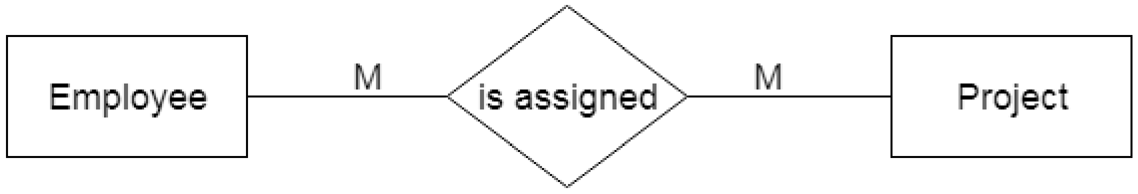
For example**,** Student enrolls for only one course, but a course can have many students.



Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example**,** Employee can assign by many projects and project can have many employees.



(Sr)

Bibliography

Bassil, Y. (February 2012). A Comparative Study on the Performance. *ournal of Computer Science & Research (JCSCR) - ISSN 2227-328X*, 20-31.

Gulati, V. (2022). *Database language in DBMS*, 20-30.

II-Yeol Song, M. E. (1995). A Comparative Analysis of Entity-Relationship Diagrams1. *Journal of Computer and Software Engineering, Vol. 3,*, 427-459.

krishna. (2020, 08 11). *edguru.* Retrieved 4 25, 2023, from edguru: https://blog.eduguru.in/rdbms/dbms/dbms-architecture-and-data-independence

Singh, S. (January – March 2015). DATABASE MANAGEMENT SYSTEM. *Journal of Management Research and Analysis*, 8-19.

Sr, R. p. (n.d.). Journal of Advanced Database Management & Systems. *Industrial Management & Data Systems*, 1-5.

Susanto, A. (JUNE 2019). Database Management System. *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 8, ISSUE 06*, 4-5.