

UNIT – I

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

1. INTRODUCTION

Data: Data is a piece of information. Data can exist in a variety of forms:

- As numbers or text on pieces of paper
- As bits and bytes stored in computer memory
- As facts stored in a person's mind.

Data is raw information and it does not give correct meaning. The processed **data** becomes **information** and it gives correct meaning.

Database: Database is a collection of inter-related data which is used to retrieve, insert, delete and manipulate the data efficiently.

Database Management System (DBMS): The software which is used to manage database is called Database Management System (DBMS).

Examples of popular DBMS:

- MySql
- Microsoft Access
- Oracle
- IBM DB2

2. A HISTORICAL PERSPECTIVE

From the earliest days of computers, storing and manipulating data have been a major application focus. The first general-purpose DBMS called the Integrated Data Store (IDS) was designed by Charles Bachman in the early 1960s. It formed the basis for the *network data model*.

In the late 1960s, IBM developed the Information Management System (IMS). This formed the basis for an alternative data representation framework called the *hierarchical data model*.

In 1970, **Edgar Codd**, proposed a new data representation framework called the ***relational data model***. In a relational data model, the data is stored in the form of table containing rows and columns. This became very famous database model. The SQL (Structured Query Language) is the standard query language used to access relational databases.

Several vendors (e.g., IBM's DB2, Oracle 8, etc) developed *data warehouses*. A Data warehouse collects data from several databases and this data is used for carrying out specialized analysis.

In mid 90s, DBMSs have entered the Internet Age. All the database vendors are added features to their DBMS aimed at making it more suitable for deployment over the Internet. Database management continues to gain more popularity and more data is brought online to access through computer networking.

Today the field is being driven by exciting visions such streaming data (youtube, vimeo, etc) as interactive video (flash, wirewax etc), multimedia databases (facebook, instagram, gaana etc), digital libraries (DELNET, Shodh ganga, etc). Thus the study of database systems could prove to be richly rewarding in more.

3. DATABASE APPLICATIONS

We use Database Management Systems in almost all application sectors. They are:

1. **Telecom:** A database is required to keep track of the information regarding calls history, network usage, customer details, generating monthly bills, maintaining balances on prepaid calling cards etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond. Ex: Airtel, IDEA, Jio, etc
2. **Banking System:** A database stores bank customer's information, maintain day to day credit and debit transactions, generate bank statements etc. Ex: SBI, HDFC, etc
3. **Online shopping:** The online shopping websites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. Ex: Amazon, Flipkart etc.
4. **Airlines:** Passenger details, reservation information along with flight schedule is stored in database. Eg: Air India, Indigo, etc

5. **Education sector:** Database systems are used in schools, colleges and universities to store and retrieve the data regarding student details, staff details, course details, exam details, attendance details, fees details etc. Ex: JNTUH, IITB, etc
6. **Sales:** To store customer information, stock details and invoice details a database is needed. Ex: Reliance Fresh, D-Mart etc.
7. **Human resources:** For information about employees, salaries, payroll taxes, and benefits and for generation of paychecks a database is required.
8. **Credit card transactions:** For purchases on credit cards and generation of monthly statements.
9. **Stock market:** For storing information about holdings, sales, and purchases of stocks; also for storing real-time market data to enable online trading.

4. DIFFERENCE BETWEEN FILE SYSTEM AND DBMS

	File System	DBMS
Definition	A file system is a software that manages the data files in a computer	DBMS is a software used to create and manage databases.
Operations	Operations such as storing, retrieving and searching are done manually in a file system. Therefore, it is difficult to manage data.	Operations such as storing, retrieving and searching data is easier in DBMS because it allows using SQL query language.
Data Consistency	Data Inconsistency is more in file system.	Data Inconsistency is less in DBMS.
Data Redundancy	Data Redundancy is more in file system.	Data Redundancy is less in a DBMS.
Backup and Recovery Process	Backup and recovery process is not efficient in files system.	DBMS has a sophisticated backup and recovery techniques.
Concurrent access	Concurrent access to the data in the file system has many problems	DBMS takes care of Concurrent access using some form of locking.

Physical address	User can locate the physical address of the files to access data in File System.	In DBMS, user is unaware of physical address where data is stored.
Security	File system provides less security to the data as compared to DBMS.	DBMS provides more security to the data.
Example	FAT, NTFS and Ext are some examples of file systems.	MySQL, MS-Access, Oracle, and DB2 are some examples of DBMS.

5. DBMS DATABASE MODELS

A Database model defines the logical design and structure of a database. It explains how data will be stored, accessed and updated in a DBMS. The different DBMS data models are:

- Network Model
- Hierarchical Model
- Entity-relationship Model
- Relational Model
- Object oriented data model

Network Data Model

Network model has the entities which are organized in a graphical representation and some entities in the graph can be accessed through several paths. The data in this model is represented as collection of records and the relationship among data are represented by links.

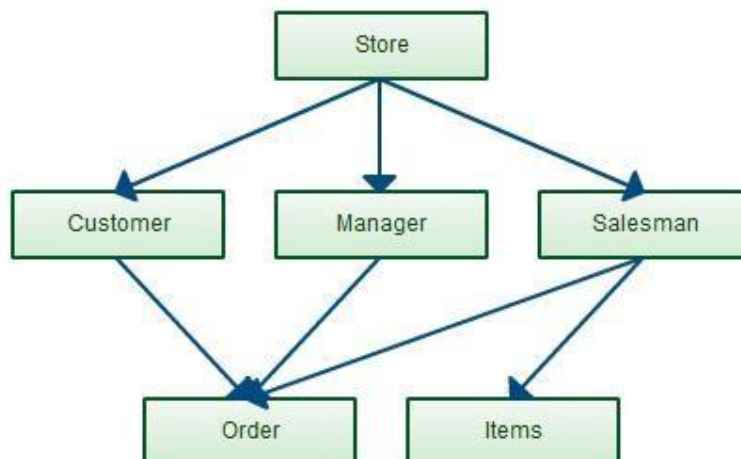


Figure: Network Model

Hierarchical Model

Hierarchical database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked. In this model, a child node will only have a single parent node.

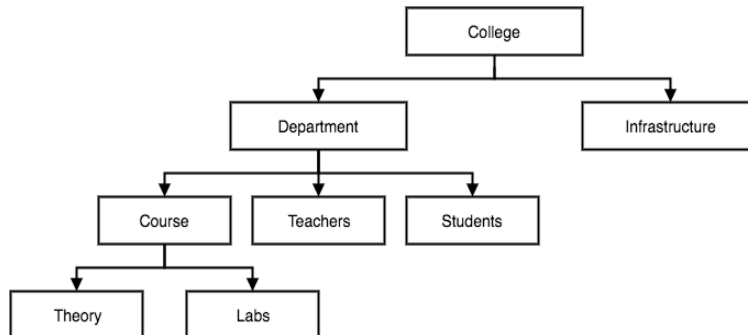
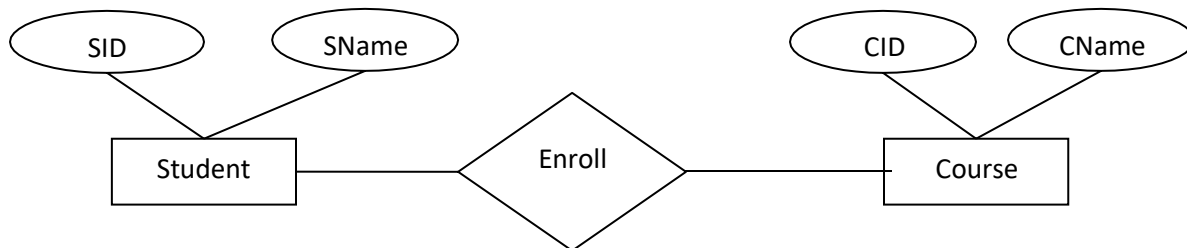


Figure: Hierarchical Data Model

Entity-Relationship Model

Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. ER Model is best used for the conceptual design of a database. While formulating real-world scenario into the database model, it depends on two important things. They are:

- Entity and their attributes
- Relationships among entities



Relational Model

The most popular data model in DBMS is the Relational Model. The relational model contains a set of tables (relations). Each table has a specified number of columns but can have any number of rows.

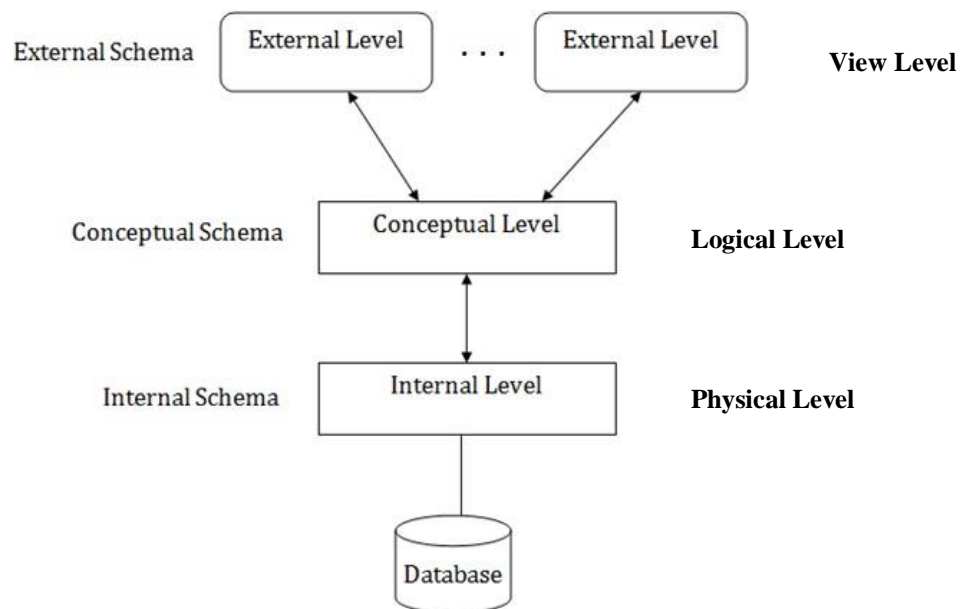
AdmissionNo	Name	Age	Class
1001	Ram	15	9
1002	Ajay	14	9
1003	Jhon	14	9
1004	Akbar	15	10

Object oriented Data Model

Object oriented data model defines a database as a collection of objects with associated attributes and methods. This model can incorporate multimedia, such as images, audio, video. The object-oriented database model is the best known post-relational database model, since it incorporates tables, but isn't limited to tables. Such models are also known as hybrid database models.

6. LEVELS OF ABSTRACTION IN A DBMS

Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction. We have three levels of abstraction.



10. Physical level: This is the lowest level of data abstraction. It describes **how** data is actually stored in database. It deals with physical memory storage details of records. These details are often hidden from the programmers.

11. Logical level: This is the middle level of 3-level data abstraction architecture. It describes **what** data is stored in database. This level gives details about each attribute data type and size, the relationship among attributes and defined constraints (Primary key, foreign key etc) on the table. The programmers generally work at this level because they are aware of such things about database systems.

12.View level: This is the highest level of data abstraction. This level describes the user interaction with database system. At this level, user enters the query to get the answer. Many users may require different sets of fields from a table. Therefore there exist many view levels.

7. DATA INDEPENDENCE

Data Independence is defined as a property of DBMS that helps you to change the Database schema at one level without requiring changing the schema at the next higher level. Data independence helps you to keep data separated from all programs that make use of it.

In DBMS there are two types of data independence

1. Physical data independence
2. Logical data independence.

Physical Data Independence: Physical data independence is the ability to change the internal schema without having to change the conceptual schema. That is, if we do any changes in the storage side of the database system server, then the Conceptual structure of the database will not be affected. For example, in case we want to change or upgrade the storage system itself – suppose we want to replace hard-disks with SSD – it should not have any impact on the logical data or schemas.

Logical Data Independence: Logical data independence is the ability to change the conceptual schema without having to change the external schema. That is, if we do any changes in the logical view of the data, then the user view/ external view of the data should not be affected.

8. STRUCTURE OF A DBMS

The DBMS accepts SQL commands generated from a variety of user interfaces such as web forms, applications, SQL interface and etc. When a user issues a query, the parsed query is presented to a query optimizer, which uses information about how the data is stored to produce an efficient execution plan for evaluating the query. An execution plan is a blueprint for evaluating a query. It executes these plans against the database, and returns the answers to the user.

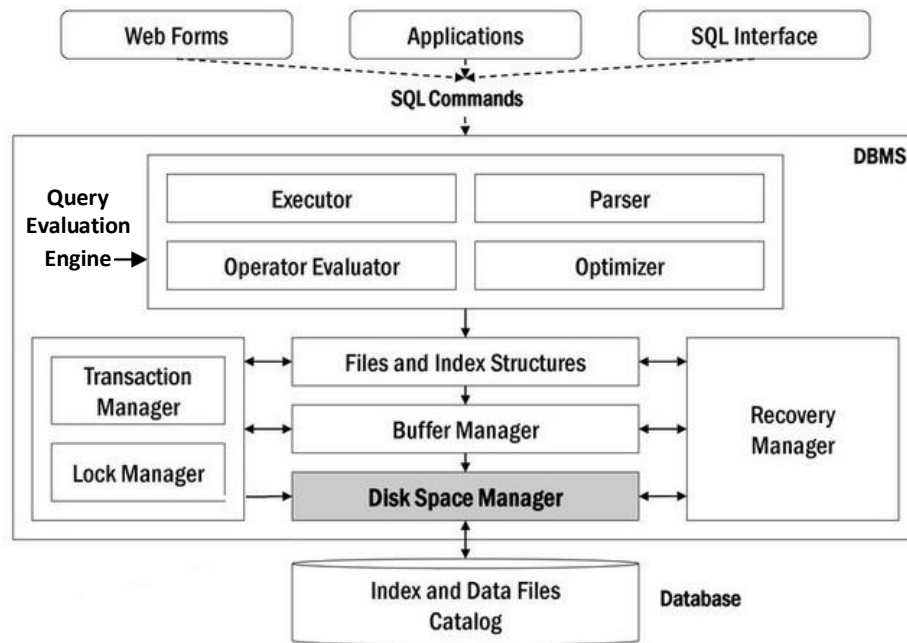


Figure: DBMS Structure

- DBMS consists of a **Query Evaluation engine** which accepts commands from the front end applications like web forms, SQL interfaces and evaluates the query to retrieve the requested data.
- **Query Evaluation engine** consists of the following components
 - **Parser:** It parses the received SQL commands.
 - **Operator evaluator:** It evaluates the operators used in the query.
 - **Plan executor:** It designs a plan to obtain the result.
 - **Optimizer:** It optimizes the query to improve the process of retrieving the resultant data.
- **File and access methods:** It is responsible for the abstraction of file structures stored and for creating indexes on the files for faster access.
- **Buffer Manager:** The purpose of buffer manager is to move pages in and out from a disk to main memory.
- **Disk Space Manager:** It manages space on the disk by providing empty space for new requests, deleting space allocated for existing files which are deleted by the user.
- **Transaction Manager and lock manager:** It is responsible for maintaining concurrency of the data, when accessed by multiple users.

- **Recovery manager:** It is responsible for maintaining log files and supports crash recovery. When a system crashes recovery manager is responsible for bringing the system to a safe state.

9. DATABASE DESIGN

The database design process can be divided into six steps.

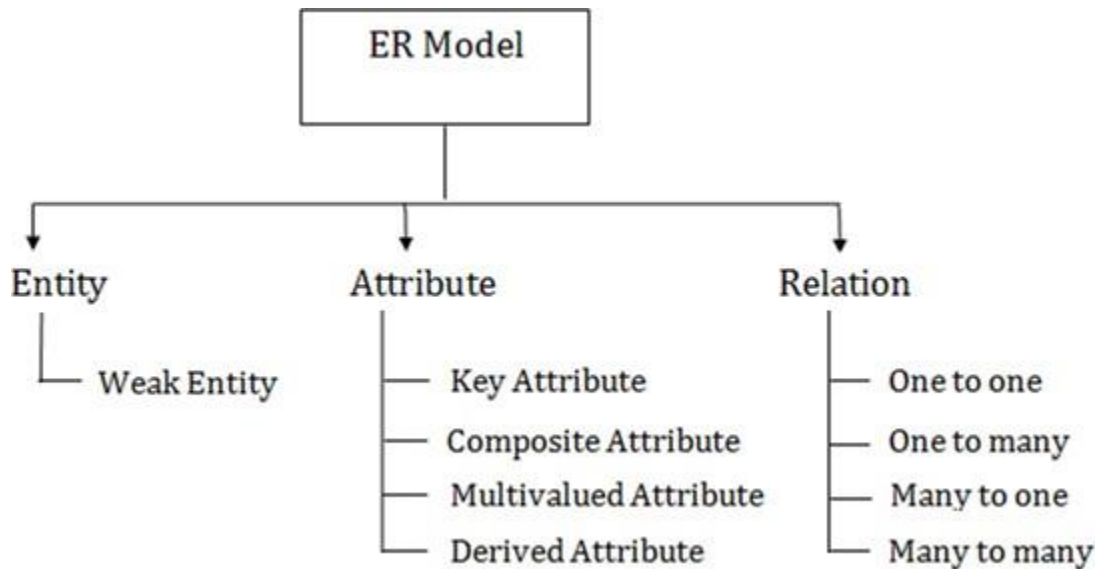
- Requirements Analysis:** The very first step in designing a database application is to gather information from different stake holders such as management, employees and end users. The development team conducts discussions with different user groups, study the current operating environment, analyze any available documentation on existing applications and gather all of the types of information that to be recorded in the database. The gathered information is documented properly.
- Conceptual Database Design:** The information gathered in the requirements analysis step is used to develop Entity Relationship (ER) model. The ER model facilitates discussion among all the people involved in the design process, even those who have no technical background.
- Logical Database Design:** The task in this stage is to convert the ER model into relational schemas. Each entity and each relationship is converted into a relation or a table.
- Schema Refinement:** The fourth step in database design is to analyze the collection of relations in our relational database schema to identify potential problems, and to refine it. This process is called normalization.
- Physical Database Design:** In this step, the database design is refined to ensure that it meets desired performance criteria and satisfies the expected workload. This step may simply involve building indexes on some tables and clustering some tables, or it may involve a substantial redesign of parts of the database schema obtained from the earlier design steps.
- Application and Security Design:** For each role (manager / accountant / clerk), some part of the database is accessible and other part of the database is not accessible. The software developer should enforce these accessing rules while developing the applications (using application languages like java) to access data using DBMS.

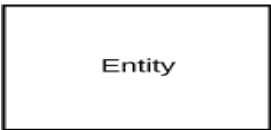

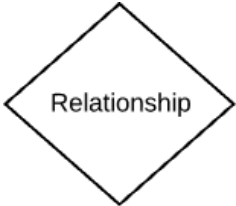
Realistically, all above six design steps are repeated until the design is satisfactory to complete database design.






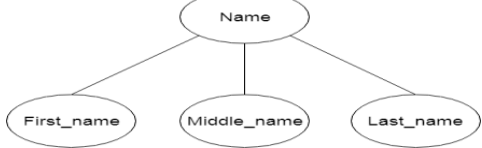
10. ER DIAGRAMS

An *entity-relationship (ER) diagram* is a graphical representation of entities and their relationships to each other, typically used to the organization of data within databases. An *entity-relationship (ER) diagram* is also called as an *entity relationship model*.

Component of ER Diagram



Symbol	Name	Description
	Entity / Strong entity	An entity may be any object, class, person or place.
	Weak entity	Weak entities depend on some other entity type. They don't have primary keys, and have no meaning in the diagram without their parent entity.
	Relationship	Relationships are associations between or among entities.

Symbol	Name	Description
	Weak relationship	Weak Relationships are connections between a weak entity and its owner.
	Attribute	Attributes are characteristics of an entity. The attribute is used to describe the property of an entity.
	Key Attribute	A <i>key attribute</i> is the unique characteristic of the entity. It represents a primary key.
	Multivalued attribute	Multivalued attributes are those that are can take on more than one value.
	Derived attribute	Derived attributes are attributes whose value can be calculated from other attribute values.
	Composite attribute	An attribute that composed of many other attributes is known as a composite attribute.

Types of relationship are as follows:

The **cardinality** of a relationship is the number of instances of entity B that can be associated with entity A. Based on the cardinality; the relationships are classified into four types. They are:

a. 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.

Example: A female can marry to one male, and a male can marry to one female.



b. 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.

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



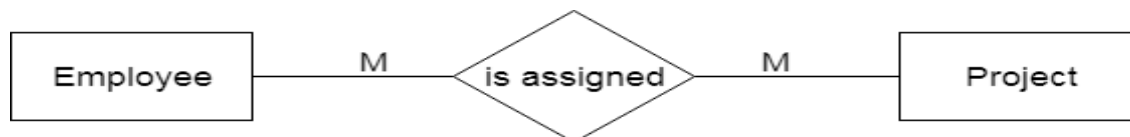
c. Many-to-one relationship: When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

Example: Student enrolls for only one course, but a course can have many students.

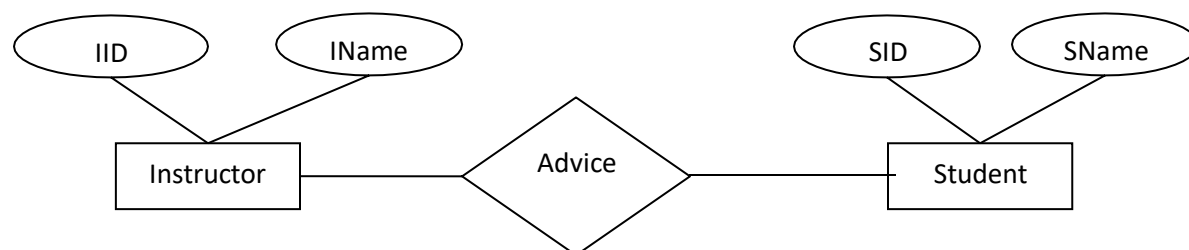


d. Many-to-many relationship: When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

Example: Employee can assign by many projects and project can have many employees.



Entity Set and Relationship Set



Entity Set: An *Entity set* is a *set of entities* of the same type that share the same properties.

The above diagram contains two entities; *Instructor* and *Student*. In the below figure, *Instructor* entity contains six different instructor values (rows) called as *Instructor entity set* and *Student* entity contain seven different student values (rows) called as *Student entity set*.

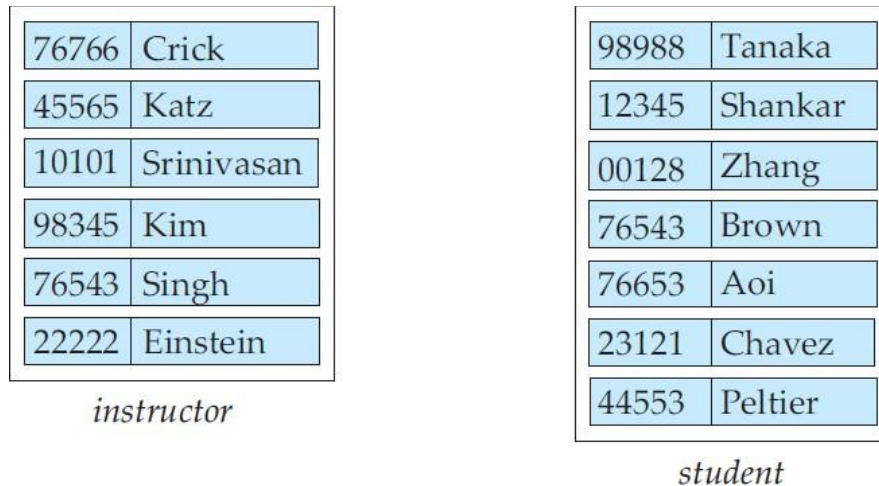


Figure: Entity set Instructor and Student

Relationship Set: A *relationship set* is a *set of relationships* of the same type. In the below figure one instructor can advise many students but every student is advised by only one instructor. This relationship is called as many-to-one relationship.

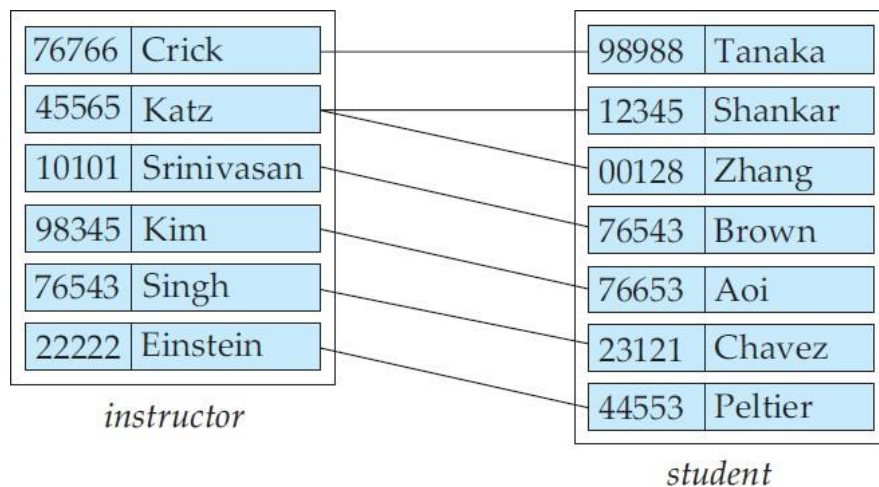


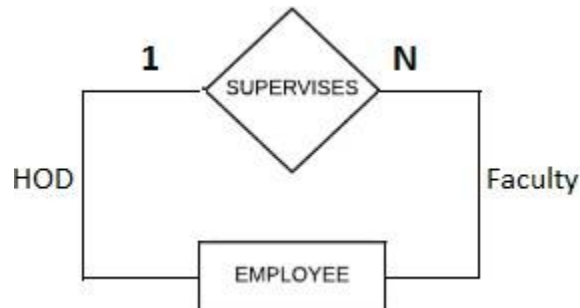
Figure: Relationship set advisor

11. ADDITIONAL FEATURES OF THE ER MODEL

N-ary relationship

In an n-ary relationship, the n shows the number of entities in the relationship. It can be anything but the most popular relationships are unary, binary and ternary relationship.

Unary Relationship: When there is a relationship between two entities of the same type, it is known as a unary or recursive relationship. This means that the relationship is between different instances of the same entity type.



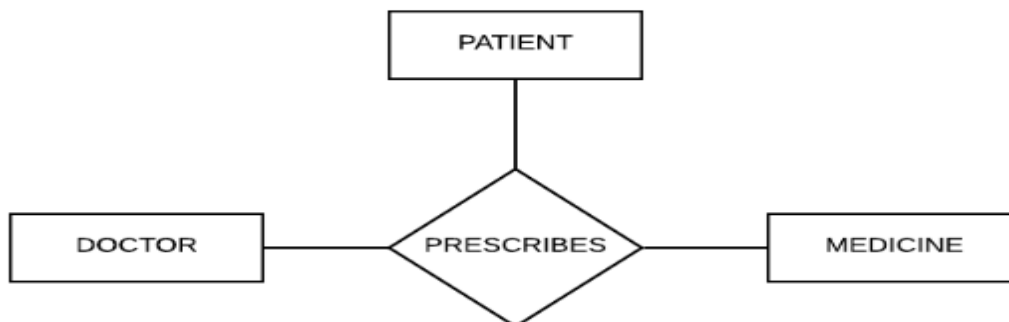
For example, an employee can supervise multiple employees. The role of one employee is HOD and the role of other employees is faculty. That is, one HOD supervises many faculties.

Binary Relationship: When there is a relationship between two different entities, it is known as a binary relationship.



Each employee only has a single ID card. Hence this is a one to one binary relationship where 1 employee has 1 ID card.

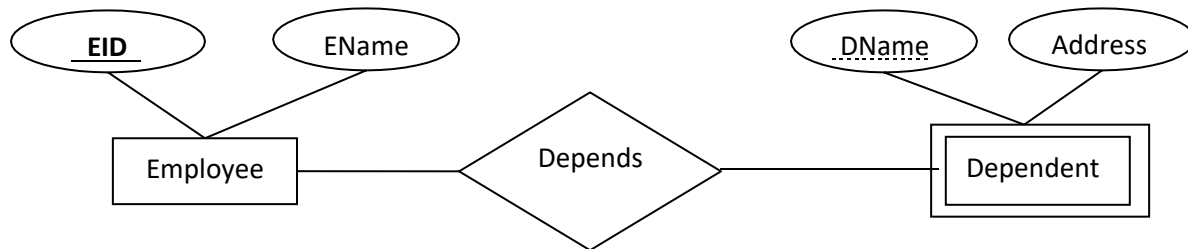
Ternary Relationship: When there is a relationship between three different entities, it is known as a ternary relationship. An example of a ternary relationship can be shown as follows:



In this example, there is a ternary relationship between Doctor, Patient and Medicine.

Weak Entity

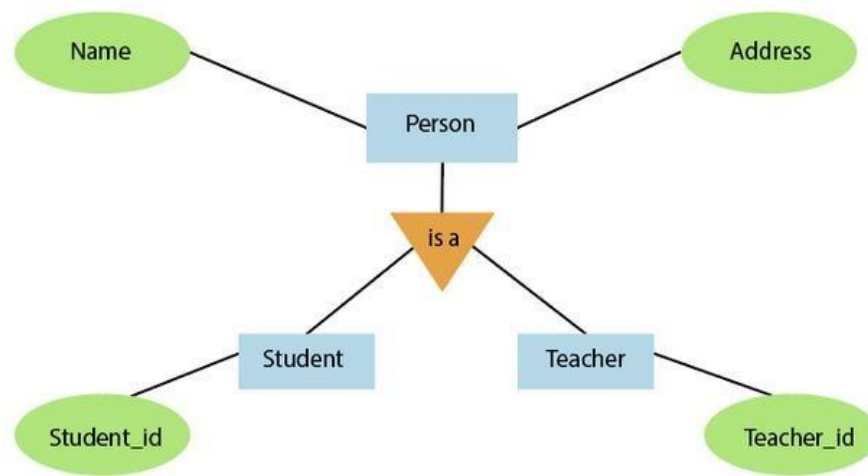
A **Weak entity** is the one that depends on its owner entity for its existence. A weak entity is denoted by the **double rectangle**. Weak entity does **not** have the **primary key**. The **primary key of a weak entity** is a composite key formed from the **primary key of the strong entity** and **partial key of the weak entity**.



There can be an employee without a dependent in the Company but there will be no record of the Dependent in the company systems without any association with an Employee.

Generalization

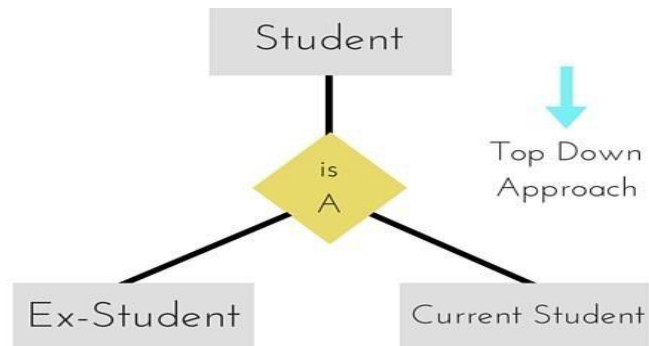
Generalization is a bottom-up approach in which two or more lower-level entities combine to form a new higher-level entity. In generalization, the generalized entity of higher level can also combine with entities of the lower-level to make further higher-level entity. It is like a superclass and subclass system, but the only difference is that it uses the bottom-up approach. In this process, the common attributes of two or more lower level entities are given to higher level entity



For example, **Student** and **Teacher** entities can be generalized and **Person** entity is created.

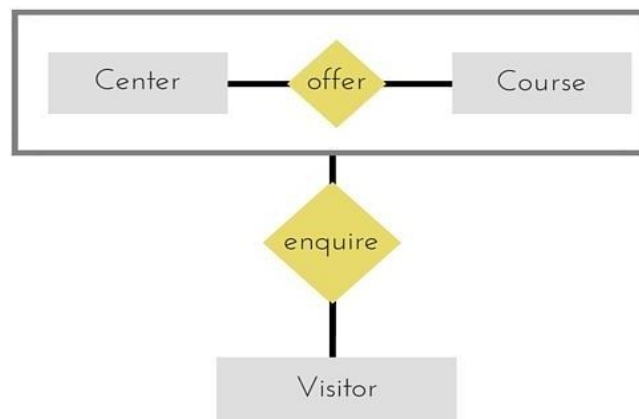
Specialization

Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two or more lower level entity.



Aggregation

Aggregation is a process when relation between two entities is treated as a **single entity**.



In the diagram above, the relationship between **Center** and **Course** together, is acting as an Entity, which is in relationship with another entity **Visitor**. Now in real world, if a Visitor or a Student visits a Coaching Center, he/she will never enquire about the center only or just about the course, rather he/she will ask enquire about both.

12. CONCEPTUAL DESIGN WITH THE ER MODEL

The document prepared in the requirement analysis phase is used to generate ER Model by following below six steps:

- **Find the entities:** Look for general nouns in requirement specification document which are of business interest to business users.
- **Identify relevant attributes:** Identify all attributes related to each entity.
- **Find the key attributes for every entity:** Identify the attribute or set of attributes which can identify each entity instance uniquely.
- **Find the relationships:** Identify the natural relationship and their cardinalities between all possible combinations of the entities.
- **Complete E-R diagram:** Draw E-R diagram along with all attributes and entities.
- **Review your results with your business users:** Show the completed ER diagram to your business user and make necessary changes.

PROBLEM: UNIVERSITY CASE STUDY

A University has many departments. Each department has a name and location. Each department has multiple instructors; one among them is the head of the department. Every instructor has a name, mobile number and room number. An instructor belongs to only one department. Each department offers multiple courses, each of which is taught by a single instructor. Each course has unique course number, name, duration and pre-requisite course. A student may enroll for many courses offered by different departments. Every student has a ID, name and date of birth.

SOLUTION

Step 1: Identify the Entities

1. DEPARTMENT
2. COURSE
3. INSTRUCTOR
4. STUDENT

Step 2: Identify all relevant attributes

1. For the "Department" entity, the relevant attribute are "Department Name" is "Location".
2. For the "Course" entity, the relevant attributes are "Course Number" are "Course Name", "Duration" and "Pre Requisite".
3. For the "Instructor" entity, the relevant attributes are "Instructor Name" are "Room Number" and "Telephone Number".
4. For the "Student" entity, the relevant attributes are "Student Number" are "Student Name" and "Date of Birth".

Step 3: Identify the key attributes

1. DName (Department Name) which identifies the department uniquely will be the key attribute for "DEPARTMENT" entity.
2. STUDENT# (Student Number) which identifies the student entity uniquely Will be the key attribute for "STUDENT" entity.
3. IName (Instructor Name) is the key attribute for "INSTRUCTOR" entity.
4. COURSE# (Course Number) is the key attribute for COURSE entity.

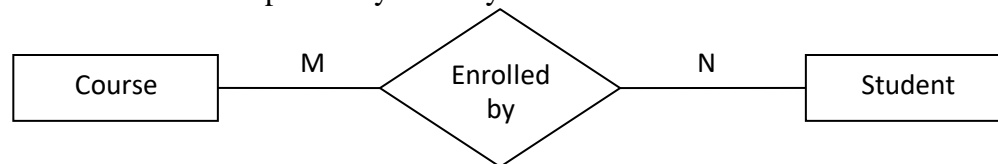
STEP 4: Find relationships.

We can derive the following relationships:

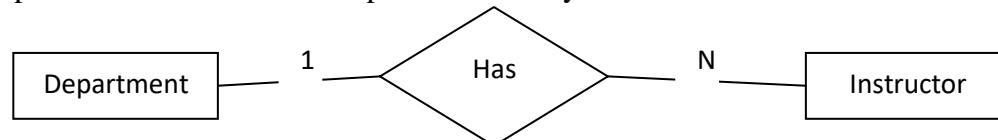
1. The department offers multiple courses and each course belongs to only one department. So the cardinality between department and course is one to many.



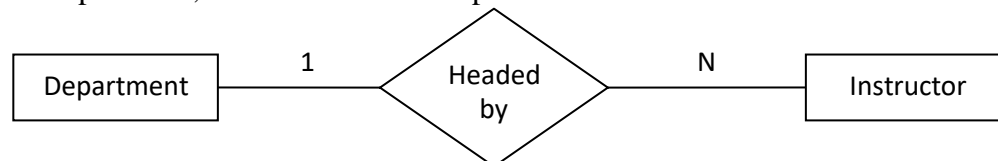
2. One course is enrolled by multiple students and also one student enrolls for multiple courses. So the relationship is many to many.



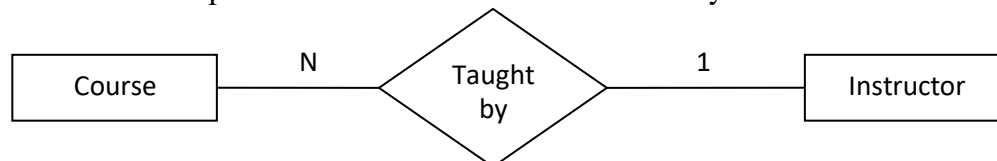
3. One department has multiple instructors and also one instructor belongs to one and only one department. So the relationship is one to many.



4. Each department has one "Head of Department" and one Instructor is Department" for only one department, hence the relationship is one to one.



5. One course is taught by only one instructor but one instructor teaches many courses, hence the relationship between course and instructor is many to one.

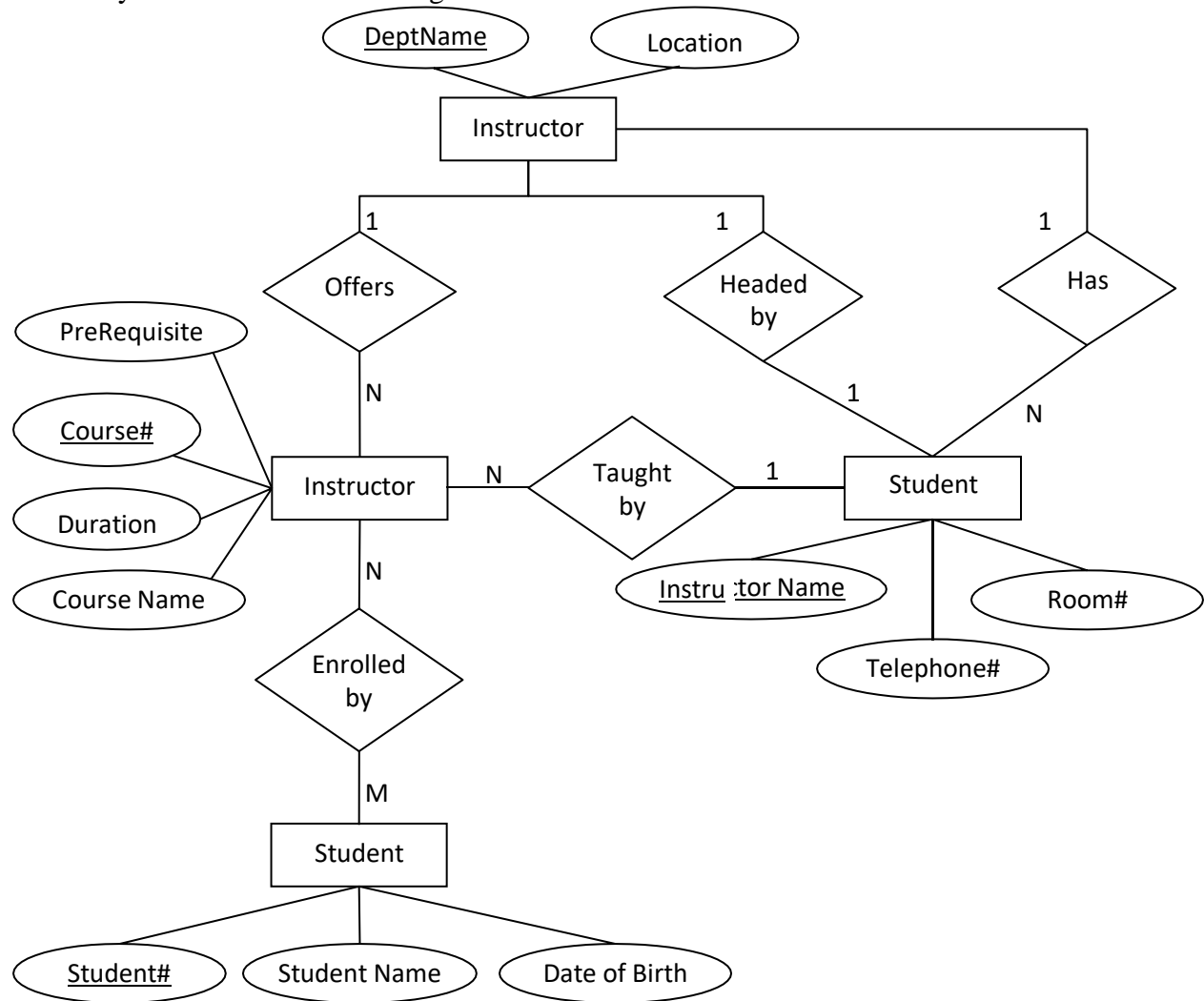


The relationship between instructor and student need NOT be defined in the diagram. The reasons are as follows:

1. There is no business significance of this relationship.
2. We can always derive this relationship indirectly through course and instructor, and course and students.

Step 5: Complete E-R diagram

After considering all the above mentioned guidelines one can generate the E-R Model for the university database as shown in Figure.



13. DESIGN CHOICES IN CONCEPTUAL DESIGN

- Should a concept be modeled as an entity or an attribute?
- Should a concept be modeled as an entity or a relationship?
- Identifying relationships: Binary or ternary? Aggregation?

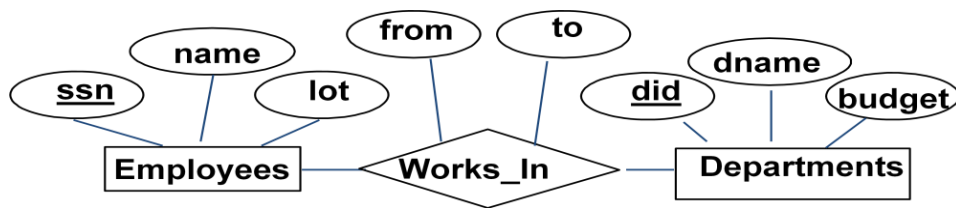
Entity vs. Attribute

- Should *address* be an attribute of Employees or an entity (related to Employees)?
- Depends upon how we want to use address information, and the semantics of the data:

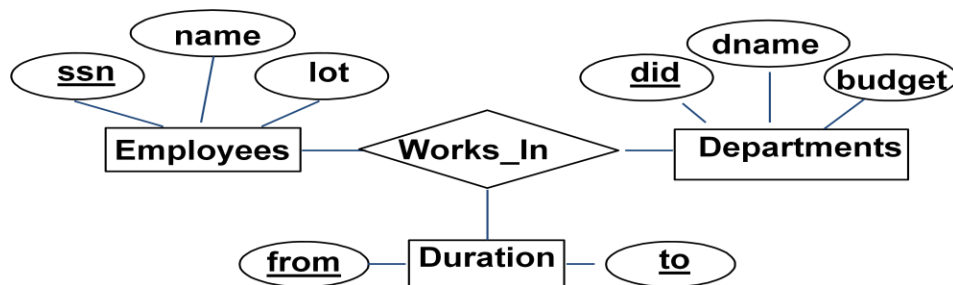
- If we have several addresses per employee, *address* must be an entity
- If the structure of address is important (plotNo, street, city, state, country and pinCode values are compulsory for each address) then, *address* must be modeled as an entity.
- Otherwise address can be modeled as an attribute.

Binary vs. Ternary Relationship

- A relationship can also have attributes.
- If an employee work in a department for a period time, then it can be modeled as given below.
- This is a binary relationship diagram



- If an employee work in a department for two or more periods, then it should be remodeled as given below.
- Then it becomes as a ternary relationship diagram



When to use aggregation?

When an entity maintains a common relationship with two or more entities, not individually then aggregation need to be used.

