Module 1

Introduction and Database Modelling using ER Model

Contents

- Introduction
  - General introduction to database systems
  - Its advantages and applications
  - Database System Architecture
  - Database users and Administrators
  - Data Models
  - Database management system
  - Database Languages
  - View of Database
- ER Model
  - Entity set
  - Entity Types
  - Attributes
  - Notations
  - Relationship sets
  - Relationship types
  - Keys: Super key , candidate key, primary key, extended features of ER Model Generalization Specialization and Aggregation

General introduction to database systems

- Collection of interrelated data
- Set of programs to access the data
- DBMS contains information about a particular enterprise
- DBMS provides an environment that is both convenient and efficient to use.

Why use DBMS?

- To develop software applications In less time.
- Data independence and efficient use of data.
- For uniform data administration.
- For data integrity and security.
- For concurrent access to data, and data recovery from crashes.
- To use user-friendly declarative query language.

Applications of DBMS

- Airlines: reservations, schedules, etc
- Telecom: calls made, customer details, network usage, etc
- Universities: registration, results, grades, etc
- Sales: products, purchases, customers, etc
- Banking: all transactions etc

A DBMS manage data and has many advantages

- Data independence: Application programs should be as free or independent as possible from details of data representation and storage. DBMS can supply an abstract view of the data for insulating application code from such facts.
- Efficient data access: DBMS utilizes a mixture of sophisticated concepts and techniques for storing and retrieving data competently, and this feature becomes important in cases where the data is stored on external storage devices.
- Data integrity and security: If data is accessed through the DBMS, the DBMS can enforce integrity constraints on the data.
- Data administration: When several users share the data, integrating the administration of data can offer major improvements. Experienced professionals understand the nature of the data being managed and can be responsible for organizing the data representation to reduce redundancy and make the data to retrieve efficiently.

Purpose of Database System

- In the early days, database applications were built on top of file systems
- Drawbacks of using file systems to store data:
  - Data redundancy and inconsistency
    - Multiple file formats, duplication of information in different files
  - Difficulty in accessing data
    - Need to write a new program to carry out each new task
  - Data isolation — multiple files and formats
  - Integrity problems
    - Integrity constraints (e.g. account balance > 0) become part of program code
    - Hard to add new constraints or change existing ones

Purpose of Database Systems (Cont.)

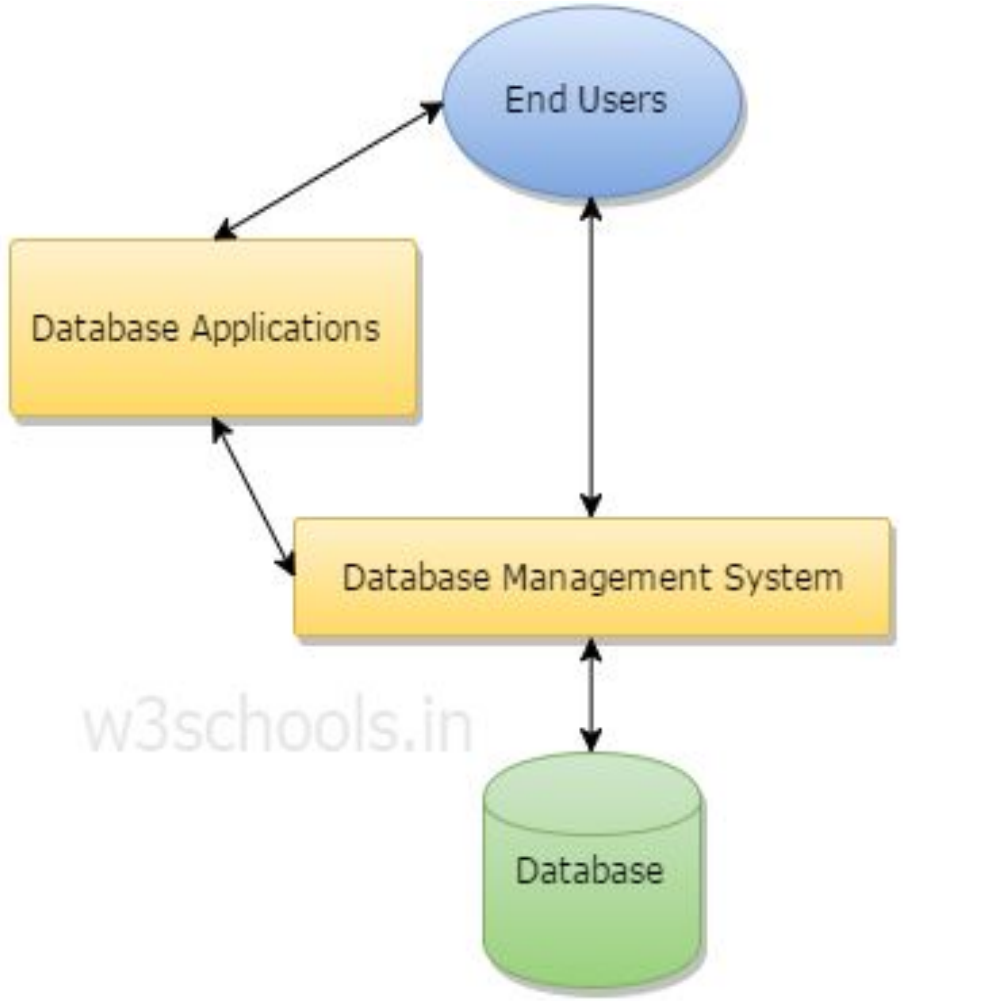
- Drawbacks of using file systems (cont.)
  - Atomicity of updates
    - Failures may leave database in an inconsistent state with partial updates carried out
    - E.g. transfer of funds from one account to another should either complete or not happen at all
  - Concurrent access by multiple users
    - Concurrent accessed needed for performance
    - Uncontrolled concurrent accesses can lead to inconsistencies
      - E.g. two people reading a balance and updating it at the same time
  - Security problems
- Database systems offer solutions to all the above problems

Difference Between File system and dbms

S.NO.	File System	DBMS
1.	File system is a software that manages and organizes the files in a storage medium within a computer.	DBMS is a software for managing the database.
2.	Redundant data can be present in a file system.	In DBMS there is no redundant data.
3.	It doesn’t provide backup and recovery of data if it is lost.	It provides backup and recovery of data even if it is lost.
4.	There is no efficient query processing in file system.	Efficient query processing is there in DBMS.
5.	There is less data consistency in file system.	There is more data consistency because of the process of normalization.
6.	It is less complex as compared to DBMS.	It has more complexity in handling as compared to file system.
7.	File systems provide less security in comparison to DBMS.	DBMS has more security mechanisms as compared to file system.
		It has a comparatively higher cost than a file

Components of DBMS

- Users: Users may be of any kind such as DB administrator, System developer or database users.
- Database application: Database application may be Departmental, Personal, organization's and / or Internal.
- DBMS: Software that allows users to create and manipulate database access,
- Database: Collection of logical data as a single unit.



Components of a Database Management System

Database Architecture

- Database management systems architecture will help us understand the components of database system and the relation among them.
- The architecture of DBMS depends on the computer system on which it runs.
- For example, in a client-server DBMS architecture, the database systems at server machine can run several requests made by client machine.

Types of DBMS Architecture

- There are three types of DBMS architecture:
- 1. Single tier architecture
  2. Two tier architecture
  3. Three tier architecture

Single tier architecture

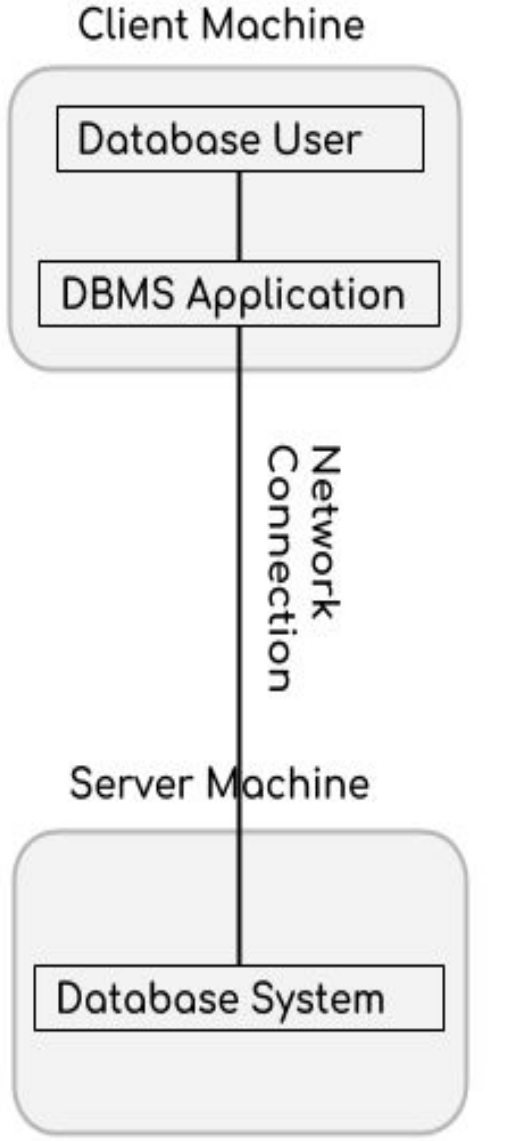
- In this type of architecture, the database is readily available on the client machine, any request made by client doesn't require a network connection to perform the action on the database.

- For example, lets say you want to fetch the records of employee from the database and the database is available on your computer system, so the request to fetch employee details will be done by your computer and the records will be fetched from the database by your computer as well. This type of system is generally referred as local database system.

Two tier architecture

- In two-tier architecture, the Database system is present at the server machine and the DBMS application is present at the client machine, these two machines are connected with each other through a reliable network as shown in the above diagram.

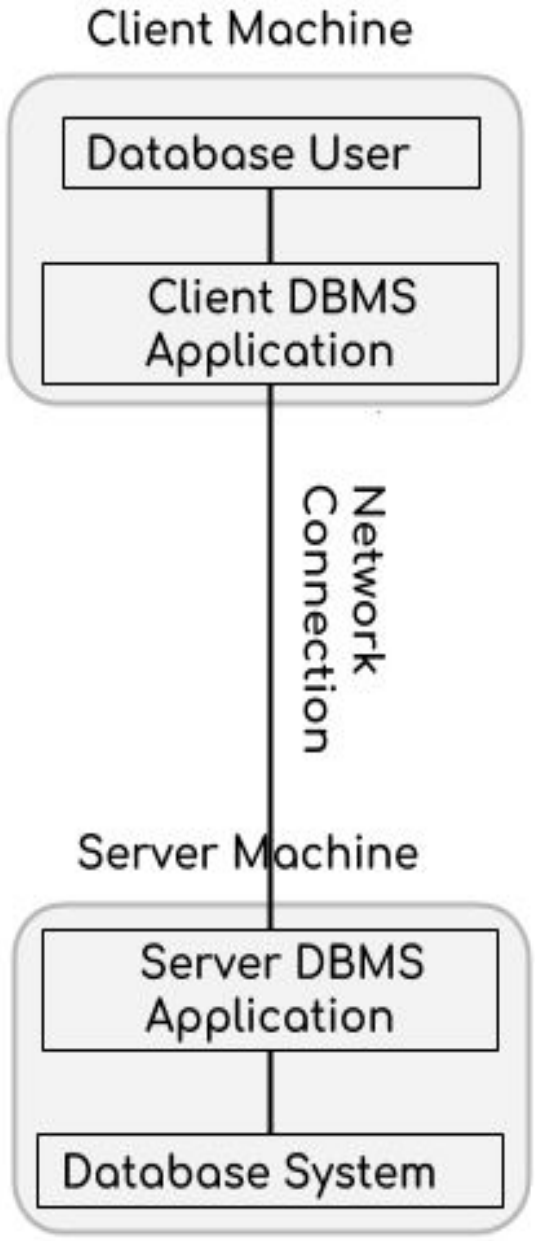
- Whenever client machine makes a request to access the database present at server using a query language like sql, the server perform the request on the database and returns the result back to the client. The application connection interface such as JDBC, ODBC are used for the interaction between server and client.



Two-Tier architecture

Three tier architecture

- In three-tier architecture, another layer is present between the client machine and server machine. In this architecture, the client application doesn't communicate directly with the database systems present at the server machine, rather the client application communicates with server application and the server application internally communicates with the database system present at the server.



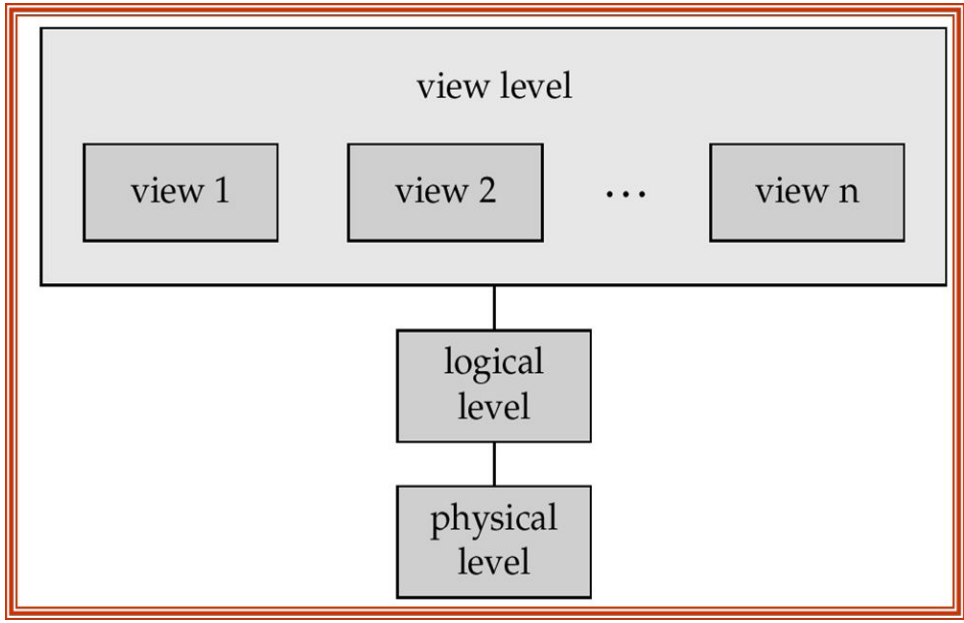
Three-Tier architecture

Data View

- Main purpose is an Abstract view of data.
- System hides certain details of how the data are stored and maintained.

Abstraction

- Abstraction is one of the main features of database systems.
- Hiding irrelevant details from user and providing abstract view of data to users, helps in easy and efficient **user-database** interaction.
- In the previous tutorial, we discussed the **three level of DBMS architecture.**
- he top level of that architecture is “view level”. The view level provides the “**view of data**” to the users and hides the irrelevant details such as data relationship, database schema, **constraints**, security etc from the user.



Abstraction continued..

- 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.
- **Physical level**: This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.
- **Logical level**: This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.
- **View level**: Highest level of data abstraction. This level describes the user interaction with database system.

Example of Abstraction

- **Example**: Let's say we are storing customer information in a customer table. At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers.
- At the **logical level** these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.
- At **view level**, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

Levels of Abstraction

- Physical level describes how a record (e.g., customer) is stored.
- Logical level: describes data stored in database, and the relationships among the data.

```
type customer = record
        name : string;
        street : string;
        city : integer;
    end;
```

- View level: application programs hide details of data types. Views can also hide information (e.g., salary) for security purposes.

Instances

- Definition of instance: The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

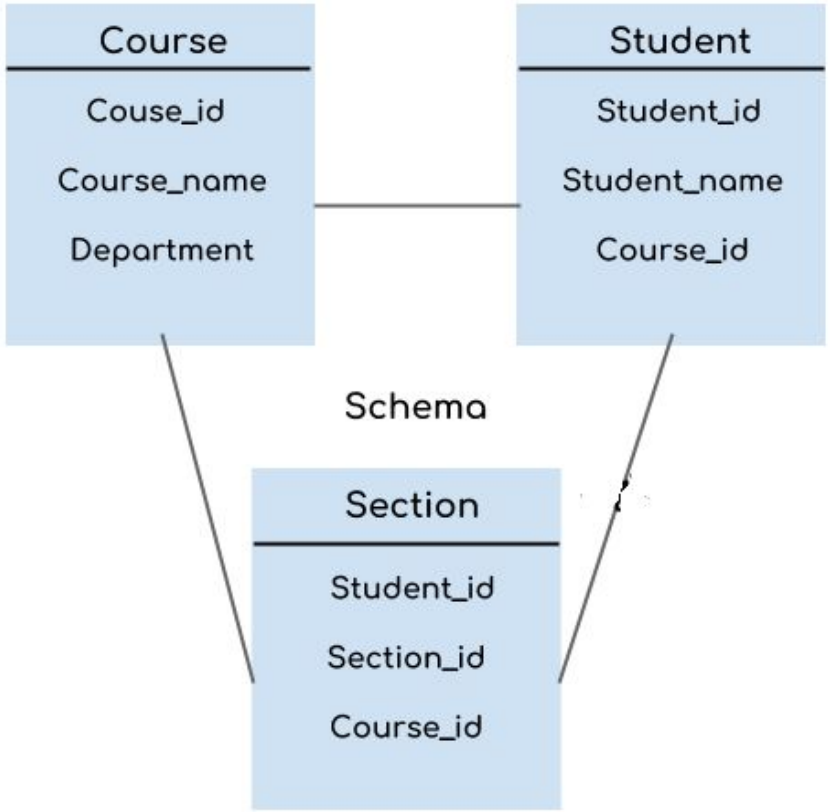
- For example, lets say we have a single table student in the database, today the table has 100 records, so today the instance of the database has 100 records. Lets say we are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table. In short, at a particular moment the data stored in database is called the instance, that changes over time when we add or delete data from the database.

schema

- Definition of schema: Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.

- Schema is only a structural view(design) of a database as shown in the diagram below.

For example: In the following diagram, we have a schema that shows the relationship between three tables: Course, Student and Section. The diagram only shows the design of the database, it doesn't show the data present in those tables.



Database System Architecture

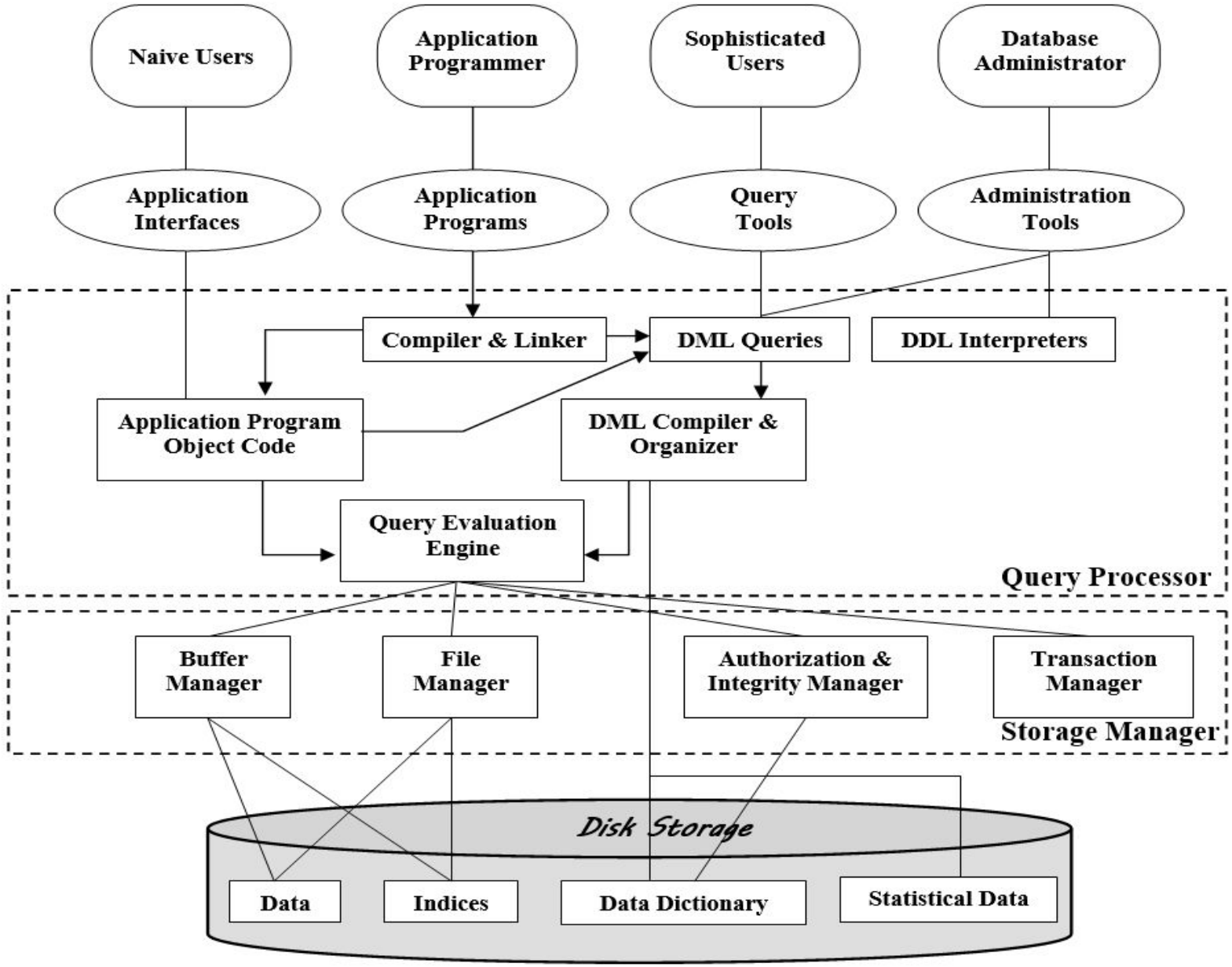


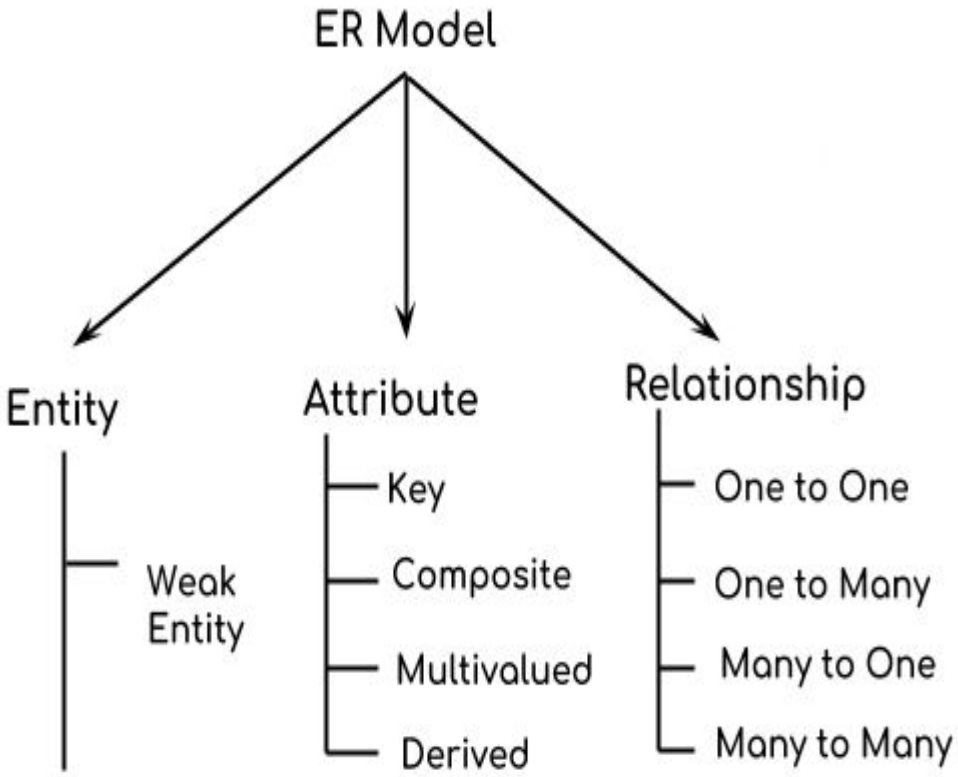
Figure: System Architecture

Data Models

- Structure of data model
- Collection of conceptual tools for describing data, data relationships, data semantics & consistency constraints.
  - ER Models
  - Relational Model
  - Object oriented data model
  - Object relational data model

ER model

- Based on perception of real world.
- Collection of Basic objects (Entities) and Relationship among them.
- An Entity is a ”thing” or “ Objects” in real world.
- Relationship is an association among them.
- ER model is created using different components.

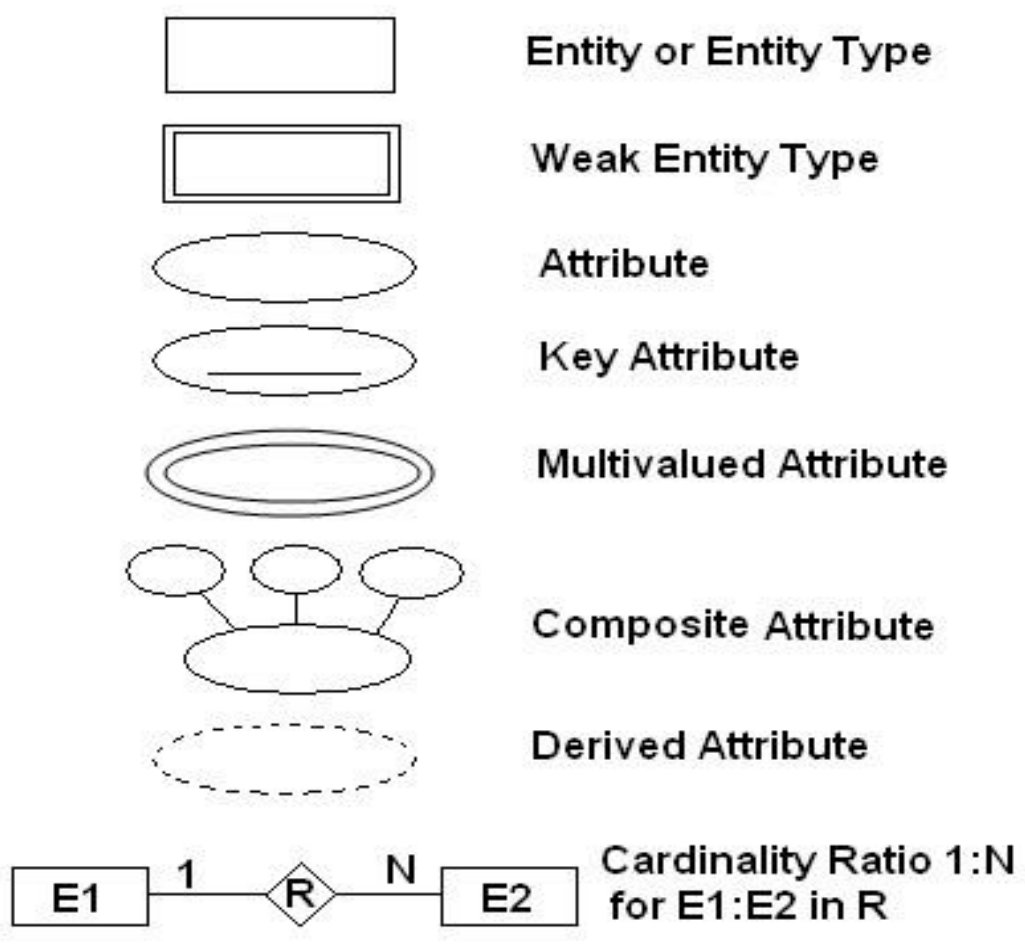


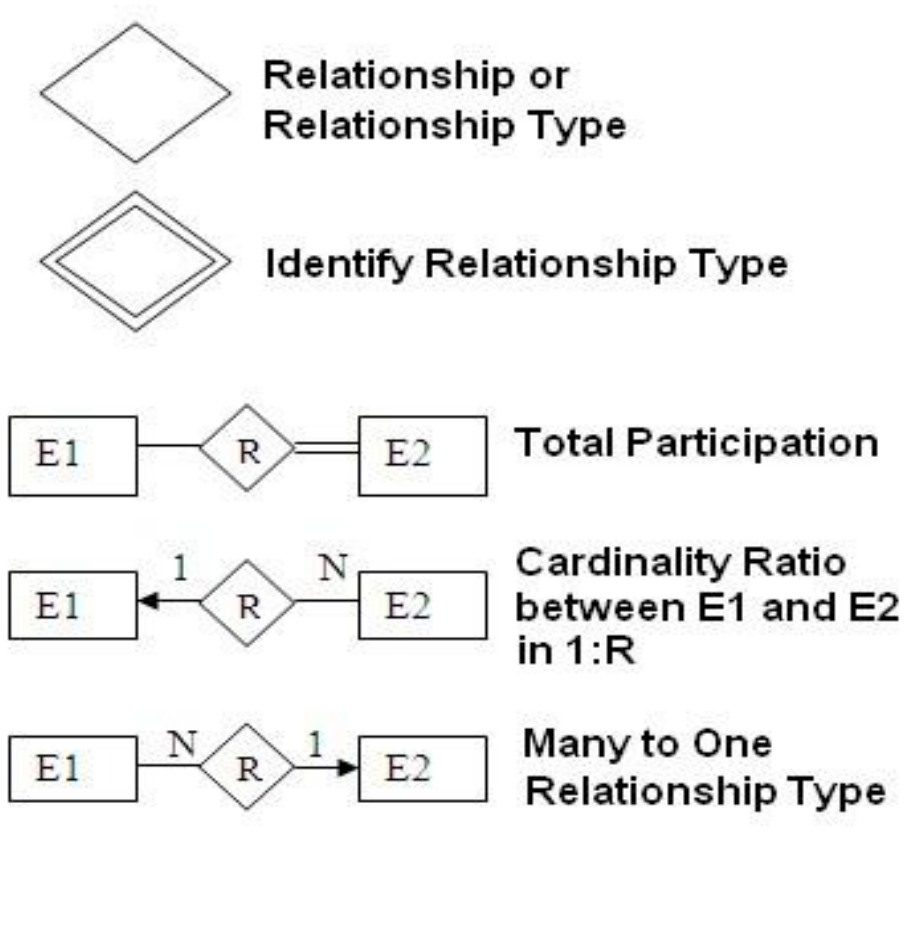
Components of ER Diagram

Purpose of ER model

- An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.
- An **entity-relationship diagram** (**ERD**) is crucial to creating a good database design. It is used as a high-level logical data model, which is **useful** in developing a conceptual design for databases. ... A relationship is the association that describes the interaction between entities.

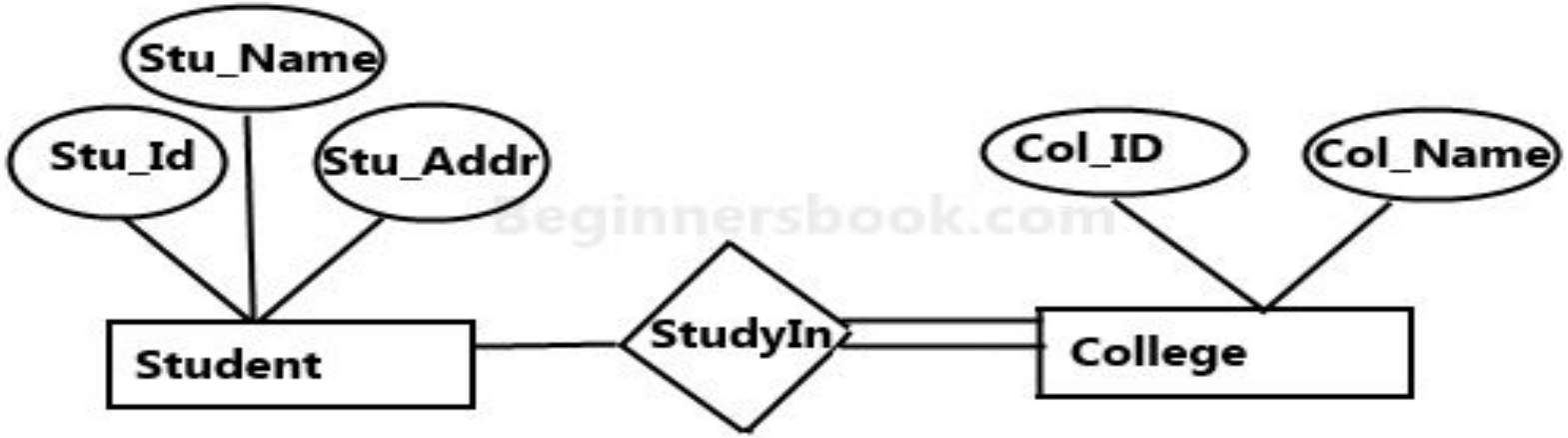
The geometric shapes and their meaning in an E-R Diagram.





Total Participation of an Entity set

- A Total participation of an entity set represents that each entity in entity set must have at least one relationship in a relationship set. For example: In the below diagram each college must have at-least one associated Student.



E-R Digram with total participation of College entity set in StudyIn relationship Set - This indicates that each college must have atleast one associated Student.

Relational Model

In relational model, the data and relationships are represented by collection of inter-related tables. Each table is a group of column and rows, where column represents attribute of an entity and rows represents records.

d	d	d
		0
		1
		42

Sample relationship Model: Student table with 3 columns and four records.

Table: Student

Stu_Id	Stu_Name	Stu_Age
111	Ashish	23
123	Saurav	22
169	Lester	24
234	Lou	26

- Table: Course

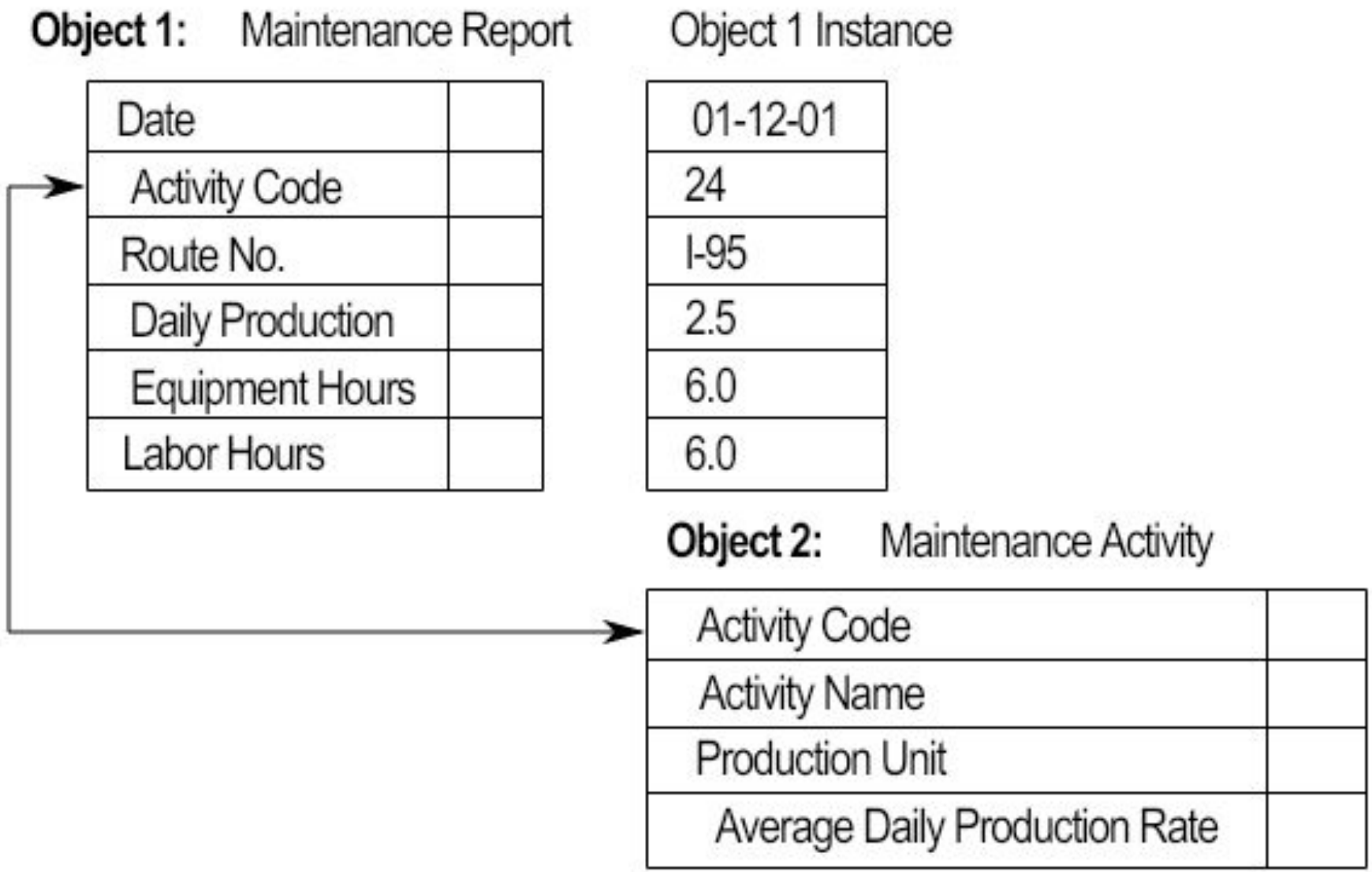
Stu_Id	Course_Id	Course_Name
111	C01	Science
111	C02	DBMS
169	C22	Java
169	C39	Computer Networks

Here Stu_Id, Stu_Name & Stu_Age are attributes of table Student and Stu_Id, Course_Id & Course_Name are attributes of table Course. The rows with values are the records (commonly known as tuples).

Object oriented data model

- Object oriented data model is based upon real world situations. These situations are represented as objects, with different attributes. All these object have multiple relationships between them.
- **Elements of Object oriented data model**
- **Objects**
- The real world entities and situations are represented as objects in the Object oriented database model.
- **Attributes and Method**
- Every object has certain characteristics. These are represented using Attributes. The behavior of the objects is represented using Methods.
- **Class**
- Similar attributes and methods are grouped together using a class. An object can be called as an instance of the class.
- **Inheritance**
- A new class can be derived from the original class. The derived class contains attributes and methods of the original class as well as its own.

Object-Oriented Model



Object relational data model

- An Object relational model is a combination of a Object oriented database model and a Relational database model. So, it supports objects, classes, inheritance etc. just like Object Oriented models and has support for data types, tabular structures etc. like Relational data model.

- One of the major goals of Object relational data model is to close the gap between relational databases and the object oriented practices frequently used in many programming languages such as C++, C#, Java etc.

Advantages of Object Relational model

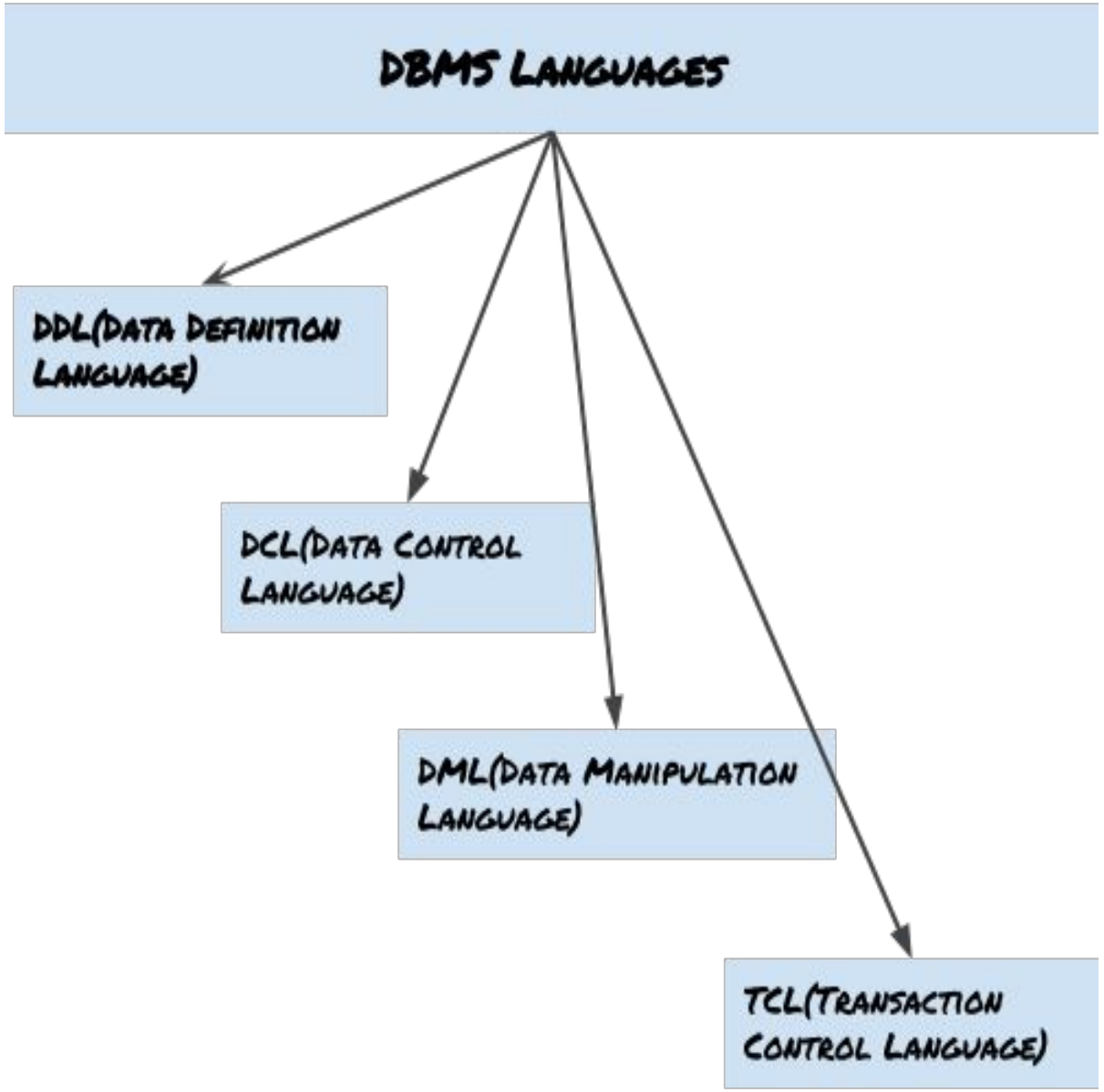
- Inheritance
  - The Object Relational data model allows its users to inherit objects, tables etc. so that they can extend their functionality. Inherited objects contains new attributes as well as the attributes that were inherited.
- Complex Data Types
  - Complex data types can be formed using existing data types. This is useful in Object relational data model as complex data types allow better manipulation of the data.
- Extensibility
  - The functionality of the system can be extended in Object relational data model. This can be achieved using complex data types as well as advanced concepts of object oriented model such as inheritance.

Disadvantages of Object Relational model

- The object relational data model can get quite complicated and difficult to handle at times as it is a combination of the Object oriented data model and Relational data model and utilizes the functionalities of both of them.

Database Languages

- Database languages are used to read, update and store data in a database.



Data Definition Language (DDL)

- DDL is used for specifying the database schema. It is used for creating tables, schema, indexes, constraints etc. in database. Lets see the operations that we can perform on database using DDL:
  - To create the database instance – **CREATE**
  - To alter the structure of database – **ALTER**
  - To drop database instances – **DROP**
  - To delete tables in a database instance – **TRUNCATE**
  - To rename database instances – **RENAME**
  - To drop objects from database such as tables – **DROP**
  - To Comment – **Comment**
- All of these commands either defines or update the database schema that's why they come under Data Definition language.

Data Manipulation Language (DML)

- DML is used for accessing and manipulating data in a database. The following operations on database comes under DML:

- To read records from table(s) – SELECT
- To insert record(s) into the table(s) – INSERT
- Update the data in table(s) – UPDATE
- Delete all the records from the table – DELETE

Data Control language (DCL)

- DCL is used for granting and revoking user access on a database –

- To grant access to user – GRANT
- To revoke access from user – REVOKE
- In practical data definition language, data manipulation language and data control languages are not separate language, rather they are the parts of a single database language such as SQL.

Transaction Control Language(TCL)

- The changes in the database that we made using DML commands are either performed or rollbacked using TCL.

- To persist the changes made by DML commands in database – COMMIT
- To rollback the changes made to the database – ROLLBACK

ER Model

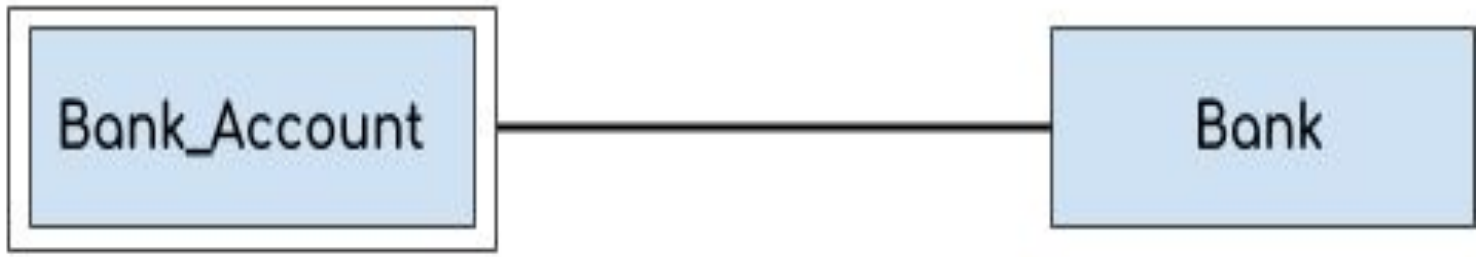
- an ER diagram has three main components:
- 1. Entity
- 2. Attribute
- 3. Relationship

1. Entity

- An entity is an object or component of data. An entity is represented as rectangle in an ER diagram.
- For example: In the following ER diagram we have two entities Student and College and these two entities have many to one relationship as many students study in a single college. We will read more about relationships later, for now focus on entities.

Weak Entity

- An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle.
- For example – a bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.



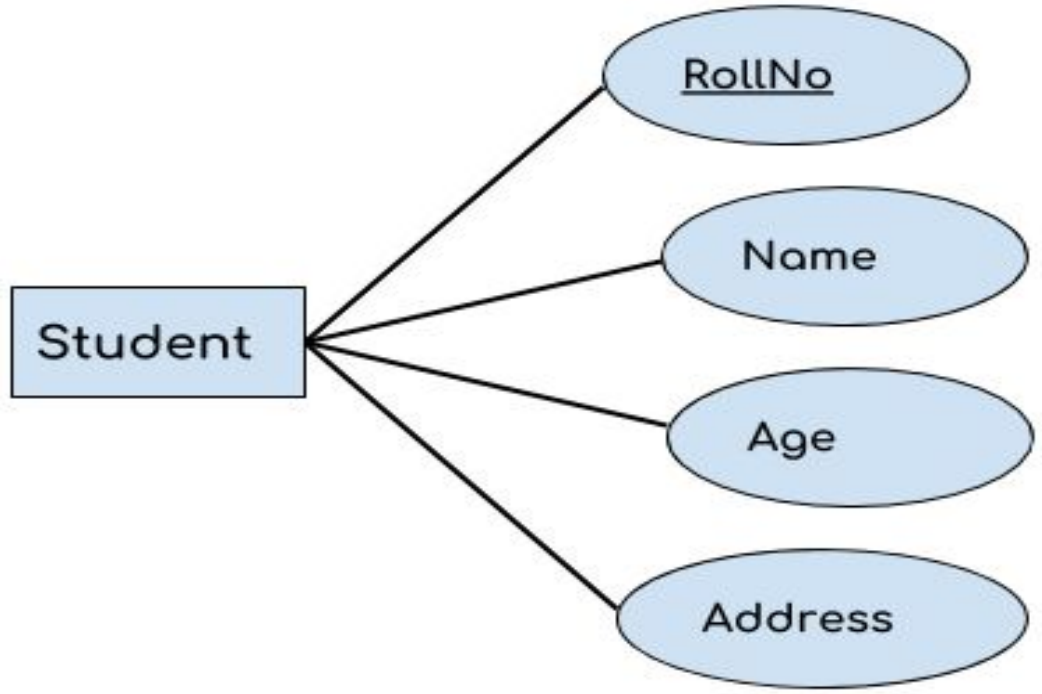
Attribute

- An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

- 1. Key attribute
- 2. Composite attribute
- 3. Multivalued attribute
- 4. Derived attribute

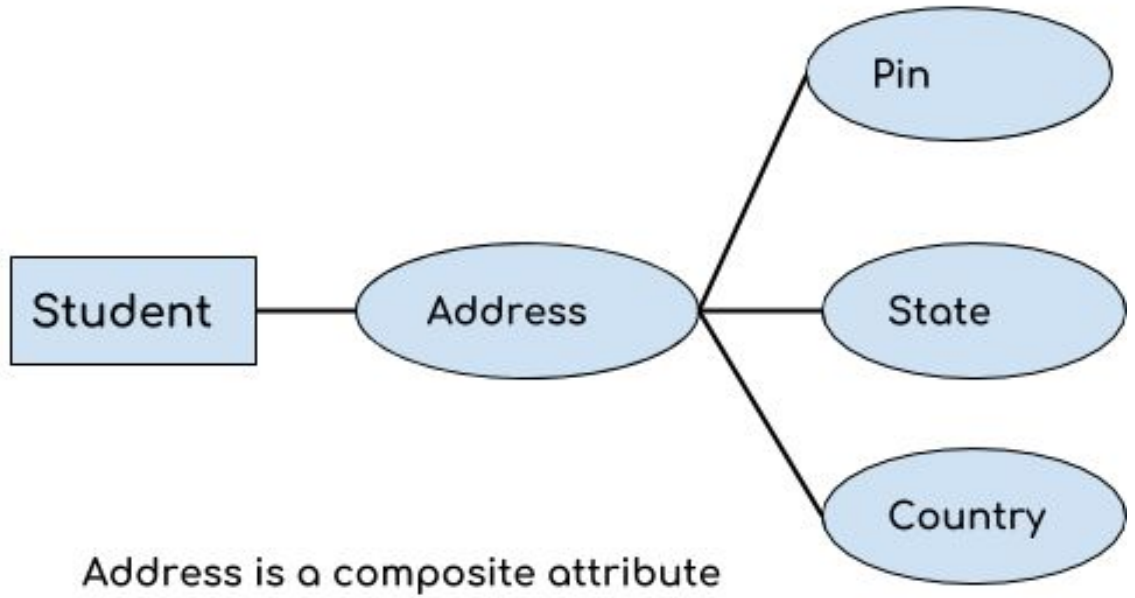
Key attribute

- A key attribute can uniquely identify an entity from an entity set. For example, student roll number can uniquely identify a student from a set of students. Key attribute is represented by oval same as other attributes however the text of key attribute is underlined.



Composite attribute:

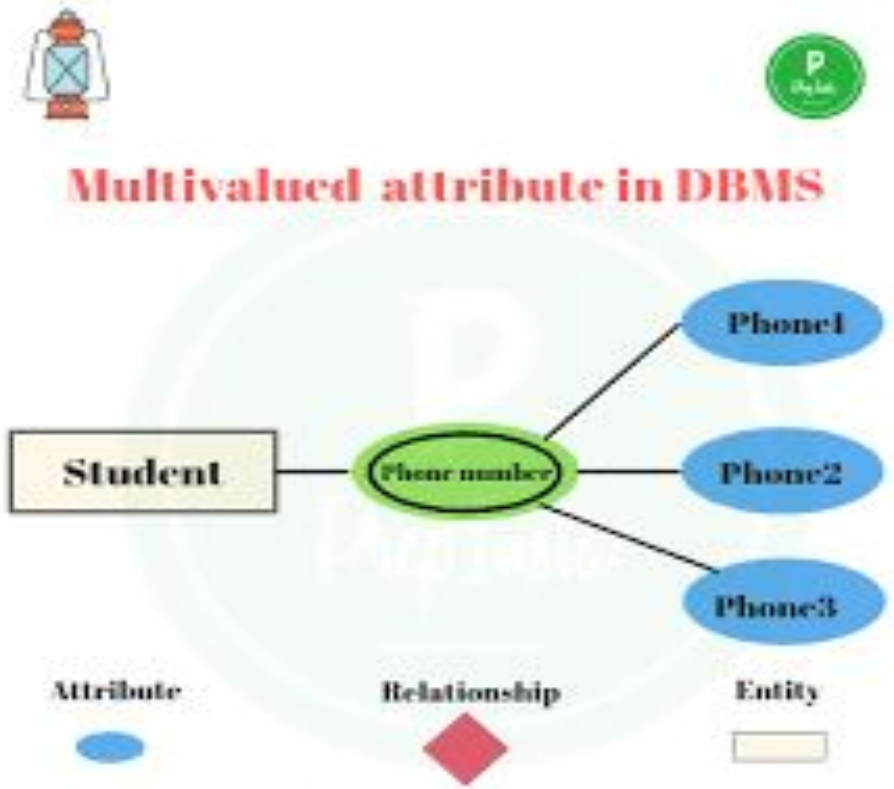
- An attribute that is a combination of other attributes is known as composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country.



Address is a composite attribute

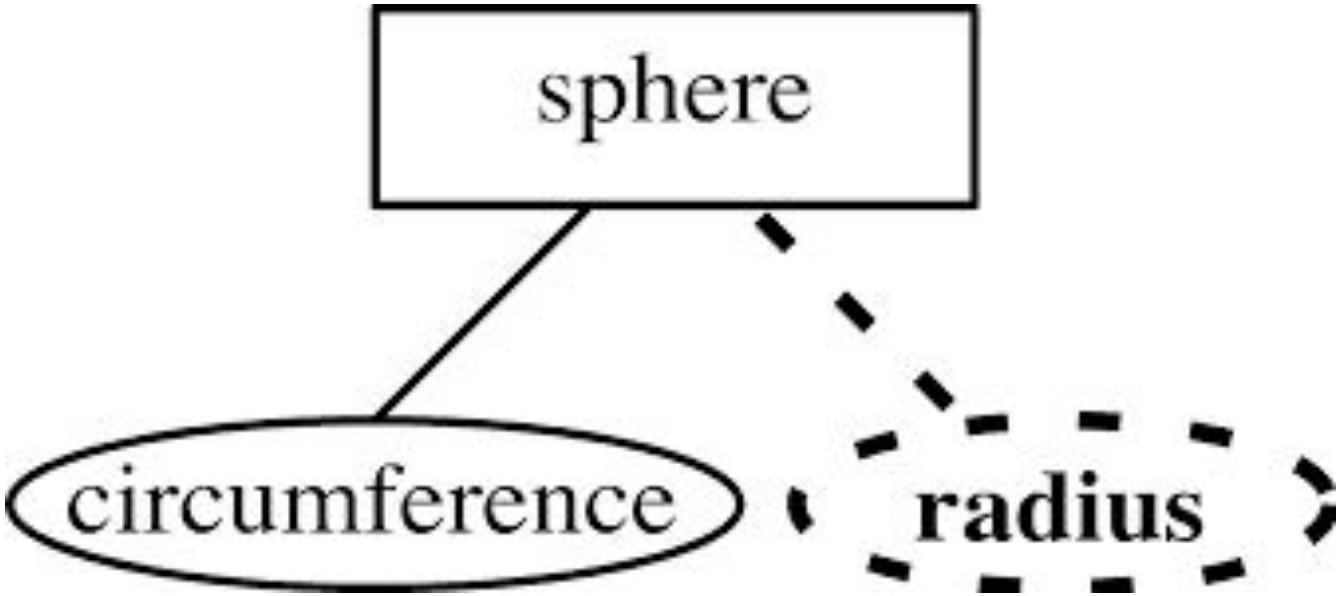
Multivalued attribute:

- An attribute that can hold multiple values is known as multivalued attribute. It is represented with double ovals in an ER Diagram. For example – A person can have more than one phone numbers so the phone number attribute is multivalued.



Derived attribute:

- A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by dashed oval in an ER Diagram. For example – Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).



Relationship

- A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:
- 1. One to One
- 2. One to Many
- 3. Many to One
- 4. Many to Many

1. One to One Relationship

- When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship. For example, a person has only one passport and a passport is given to one person.

One to Many Relationship

- When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship. For example – a customer can place many orders but a order cannot be placed by many customers.

Many to One Relationship

- When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship. For example – many students can study in a single college but a student cannot study in many colleges at the same time.

Many to Many Relationship

- When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationship. For example, a can be assigned to many projects and a project can be assigned to many students.

Key

- Key plays an important role in relational database; it is used for identifying unique rows from table. It also establishes relationship among tables.
- **Primary Key** – A primary is a column or set of columns in a table that uniquely identifies tuples (rows) in that table.
- **Super Key** – A super key is a set of one of more columns (attributes) to uniquely identify rows in a table.
- **Candidate Key** – A super key with no redundant attribute is known as candidate key
- **Alternate Key** – Out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate or secondary keys.
- **Composite Key** – A key that consists of more than one attribute to uniquely identify rows (also known as records & tuples) in a table is called composite key.
- **Foreign Key** – Foreign keys are the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.

Primary Key:A **primary key** is a minimal set of attributes (columns) in a table that uniquely identifies tuples (rows) in that table.

- Example:
  - Lets take an example to understand the concept of primary key. In the following table, there are three attributes: Stu_ID, Stu_Name & Stu_Age. Out of these three attributes, one attribute or a set of more than one attributes can be a primary key.
  - Attribute Stu_Name alone cannot be a primary key as more than one students can have same name.
  - Attribute Stu_Age alone cannot be a primary key as more than one students can have same age.
  - Attribute Stu_Id alone is a primary key as each student has a unique id that can identify the student record in the table.

Table Name: STUDENT

Stu_Id	Stu_Name	Stu_Age
101	Steve	23
102	John	24
103	Robert	28
104	Steve	29
105	Carl	29

Points to Note regarding Primary Key

- We denote usually denote it by underlining the attribute name (column name).
- The value of primary key should be unique for each row of the table. The column(s) that makes the key cannot contain duplicate values.
- The attribute(s) that is marked as primary key is not allowed to have null values.
- Primary keys are not necessarily to be a single attribute (column). It can be a set of more than one attributes (columns).

Way to define ：

- Create table STUDENT
- (
- Stu_Id int primary key,
- Stu_Name varchar(255) not null,
- Stu_Age int not null
-)

Super key: A super key is a set of one or more attributes (columns), which can uniquely identify a row in a table.

- Table: Employee
  - Emp_SSN Emp_Number Emp_Name
  - --- --- ---
  - 123456789 226 Steve
  - 999999321 227 Ajeet
  - 888997212 228 Chaitanya
  - 777778888 229 Robert
- The above table has following super keys. All of the following sets of super key are able to uniquely identify a row of the employee table.
- {Emp_SSN}
- {Emp_Number}
- {Emp_SSN, Emp_Number}
- {Emp_SSN, Emp_Name}
- {Emp_SSN, Emp_Number, Emp_Name}
- {Emp_Number, Emp_Name}

Candidate Key – A super key with no redundant attribute is known as candidate key

- How candidate key is different from super key?
  - Candidate keys are selected from the set of super keys, the only thing we take care while selecting candidate key is: It should not have any redundant attribute. That's the reason they are also termed as minimal super key.
- **Candidate Keys**: a candidate key is a minimal super key with no redundant attributes. The following two set of super keys are chosen from the above sets as there are no redundant attributes in these sets.
- {Emp_SSN}
- {Emp_Number}
- Only these two sets are candidate keys as all other sets are having redundant attributes that are not necessary for unique identification.

Super key vs Candidate Key

1. First you have to understand that all the candidate keys are super keys. This is because the candidate keys are chosen out of the super keys.

2. How we choose candidate keys from the set of super keys? We look for those keys from which we cannot remove any fields. In the above example, we have not chosen {Emp_SSN, Emp_Name} as candidate key because {Emp_SSN} alone can identify a unique row in the table and Emp_Name is redundant.

- Primary key:
- A Primary key is selected from a set of candidate keys. This is done by database admin or database designer. We can say that either {Emp_SSN} or {Emp_Number} can be chosen as a primary key for the table Employee.

Alternate key: Out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate or secondary keys.

All the keys which are not primary key are called an alternate key. It is a candidate key which is currently not the primary key. However, A table may have single or multiple choices for the primary key.
Example: In this table.
StudID, Roll No, Email are qualified to become a primary key. But since StudID is the primary key, Roll No, Email becomes the alternative key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

Composite Key – A key that consists of more than one attribute to uniquely identify rows (also known as records & tuples) in a table is called composite key.

- Any key such as **super key**, **primary key**, **candidate key** etc. can be called composite key if it has more than one attributes.
- Lets consider a table Sales. This table has four columns (attributes) – cust_Id, order_Id, product_code & product_count.
- **Table – Sales**
  - cust_Id order_Id product_code product_count
  - -------- -------- ------------ -------------
  - C01 O001 P007 23
  - C02 O123 P007 19
  - C02 O123 P230 82
  - C01 O001 P890 42

- None of these columns alone can play a role of key in this table.

- Column cust_Id alone cannot become a key as a same customer can place multiple orders, thus the same customer can have multiple entires.

- Column order_Id alone cannot be a primary key as a same order can contain the order of multiple products, thus same order_Id can be present multiple times.

- Column product_code cannot be a primary key as more than one customers can place order for the same product.

- Column product_count alone cannot be a primary key because two orders can be placed for the same product count.

- Based on this, it is safe to assume that the key should be having more than one attributes:
- Key in above table: {cust_id, product_code}

- This is a composite key as it is made up of more than one attributes.

- A foreign key is a column which is added to create a relationship with another table. Foreign keys help us to maintain data integrity and also allows navigation between two different instances of an entity. Every relationship in the model needs to be supported by a foreign key.

Example:

DeptCode	DeptName
001	Science
002	English
005	Computer

Teacher ID	Fname	Lname
B002	David	Warner
B017	Sara	Joseph
B009	Mike	Brunton

In this example, we have two table, teach and department in a school. However, there is no way to see which search work in which department.
In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

This concept is also known as Referential Integrity.

What is a Surrogate Key?

- An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key. They do not lend any meaning to the data in the table. Surrogate key is usually an integer.

Fname	Lastname	Start Time	End Time
Anne	Smith	09:00	18:00
Jack	Francis	08:00	17:00
Anna	McLean	11:00	20:00
Shown	Willam	14:00	23:00

- Above, given example, shown shift timings of the different employee. In this example, a surrogate key is needed to uniquely identify each employee.

- Surrogate keys are allowed when
  - No property has the parameter of the primary key.
  - In the table when the primary key is too big or complicated.

Why we need a Key?

- Here, are reasons for using Keys in the DBMS system.
- Keys help you to identify any row of data in a table. In a real-world application, a table could contain thousands of records. Moreover, the records could be duplicated. Keys ensure that you can uniquely identify a table record despite these challenges.
- Allows you to establish a relationship between and identify the relation between tables
- Help you to enforce identity and integrity in the relationship.

Summary

- A DBMS key is an attribute or set of an attribute which helps you to identify a row(tuple) in a relation(table)
- DBMS keys allow you to establish a relationship between and identify the relation between tables
- Seven Types of DBMS keys are Super, Primary, Candidate, Alternate, Foreign, Compound, Composite, and Surrogate Key.
- A super key is a group of single or multiple keys which identifies rows in a table.
- A column or group of columns in a table which helps us to uniquely identifies every row in that table is called a primary key
- All the keys which are not primary key are called an alternate key
- A super key with no repeated attribute is called candidate key
- A compound key is a key which has many fields which allow you to uniquely recognize a specific record
- A key which has multiple attributes to uniquely identify rows in a table is called a composite key
- An artificial key which aims to uniquely identify each record is called a surrogate key
- Primary Key never accept null values while a foreign key may accept multiple null values.

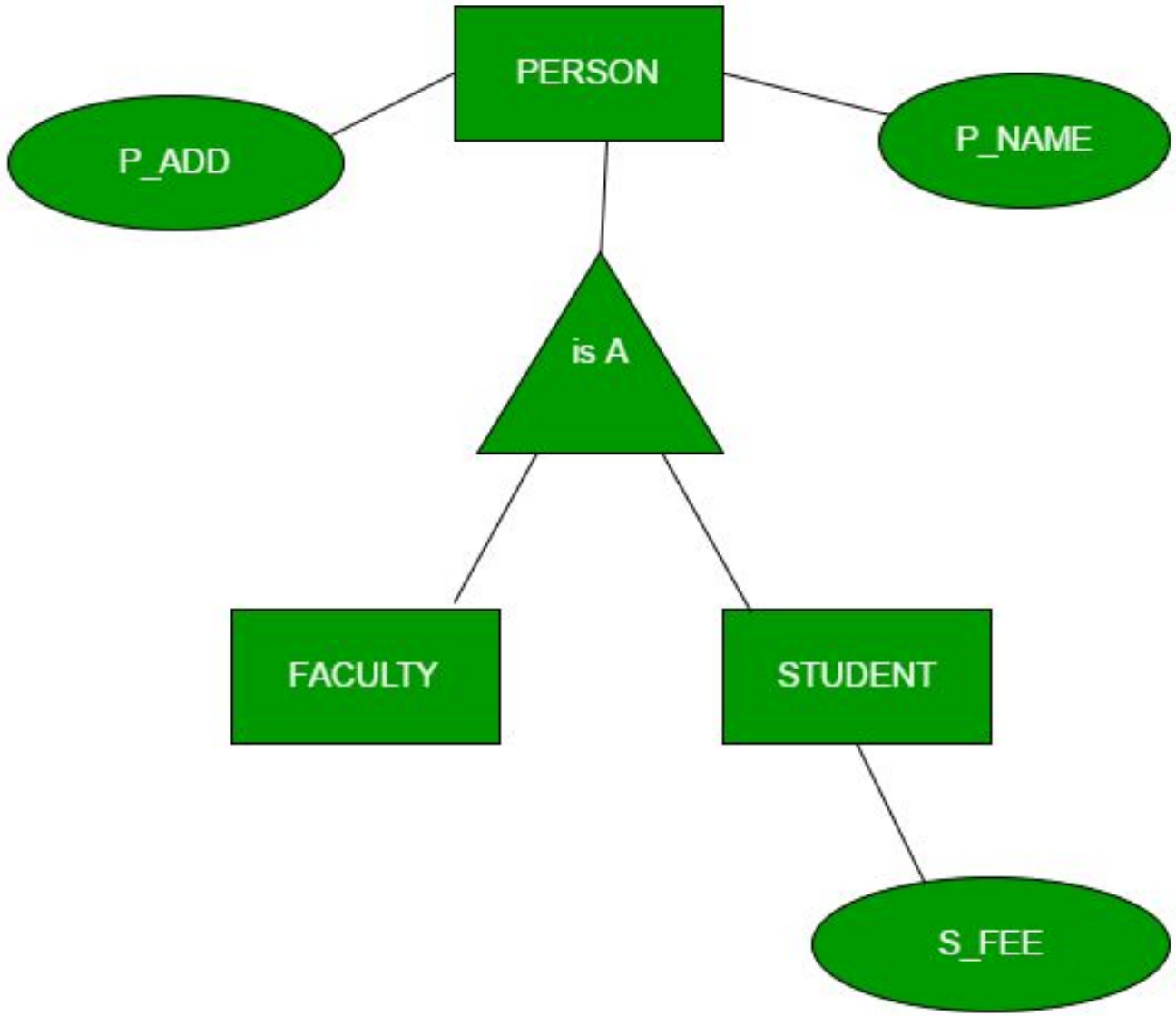
Extended Features of ER model

- Generalization
- Specialization
- Aggregation

- Generalization, Specialization and Aggregation in ER model are used for data abstraction in which abstraction mechanism is used to hide details of a set of objects.

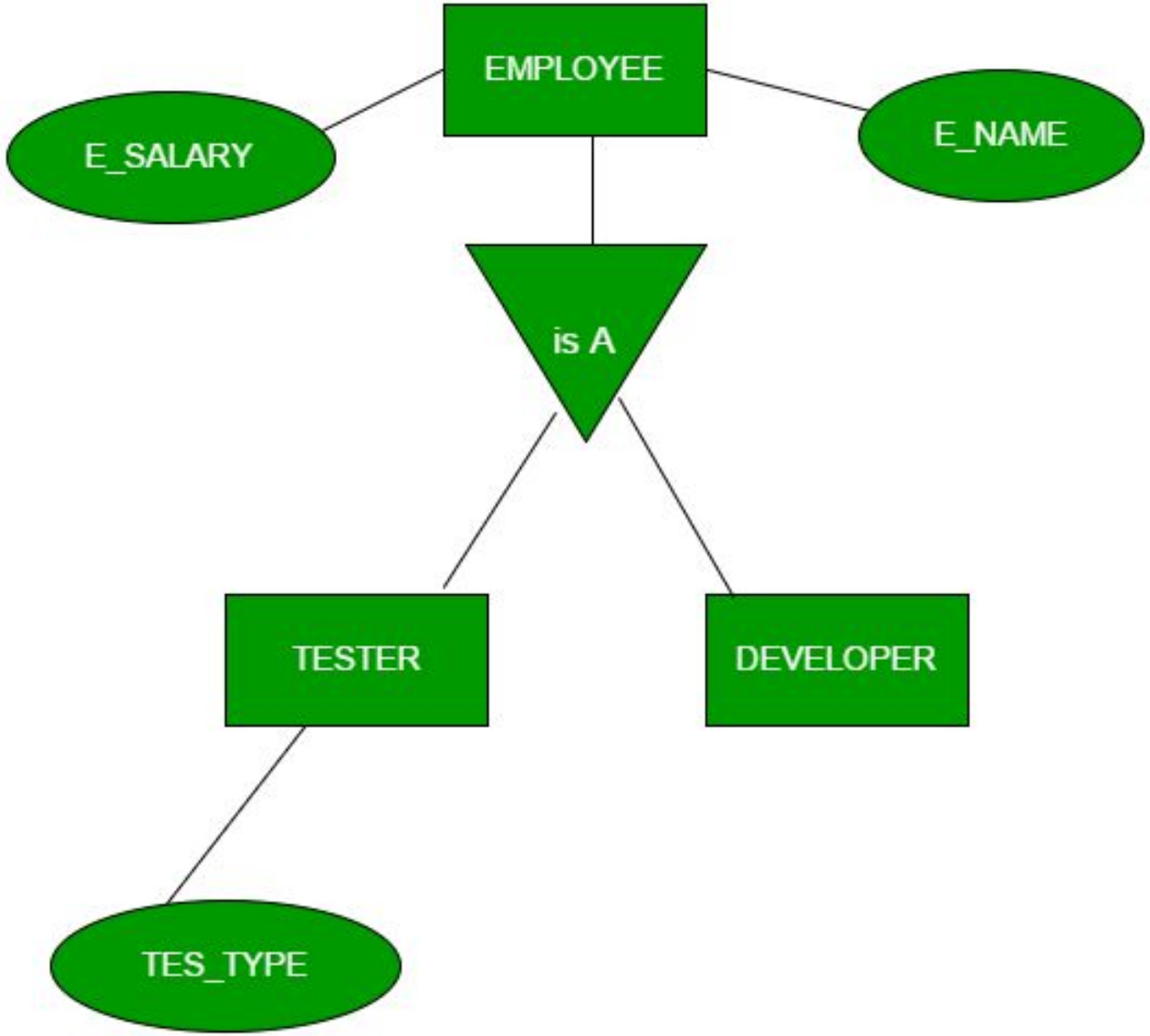
Generalization

- Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it.
- It is a bottom-up approach in which two or more entities can be generalized to a higher level entity if they have some attributes in common.
- For Example, STUDENT and FACULTY can be generalized to a higher level entity called PERSON as shown in following Figure . In this case, common attributes like P_NAME, P_ADD become part of higher entity (PERSON) and specialized attributes like S_FEE become part of specialized entity (STUDENT).


PERSON
P_ADD
P_NAME
is A
FACULTY
STUDENT
S_FEE

Specialization

- In specialization, an entity is divided into sub-entities based on their characteristics.
- It is a top-down approach where higher level entity is specialized into two or more lower level entities.
- For Example, EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER etc. as shown in following Figure . In this case, common attributes like E_NAME, E_SAL etc. become part of higher entity (EMPLOYEE) and specialized attributes like TES_TYPE become part of specialized entity (TESTER).

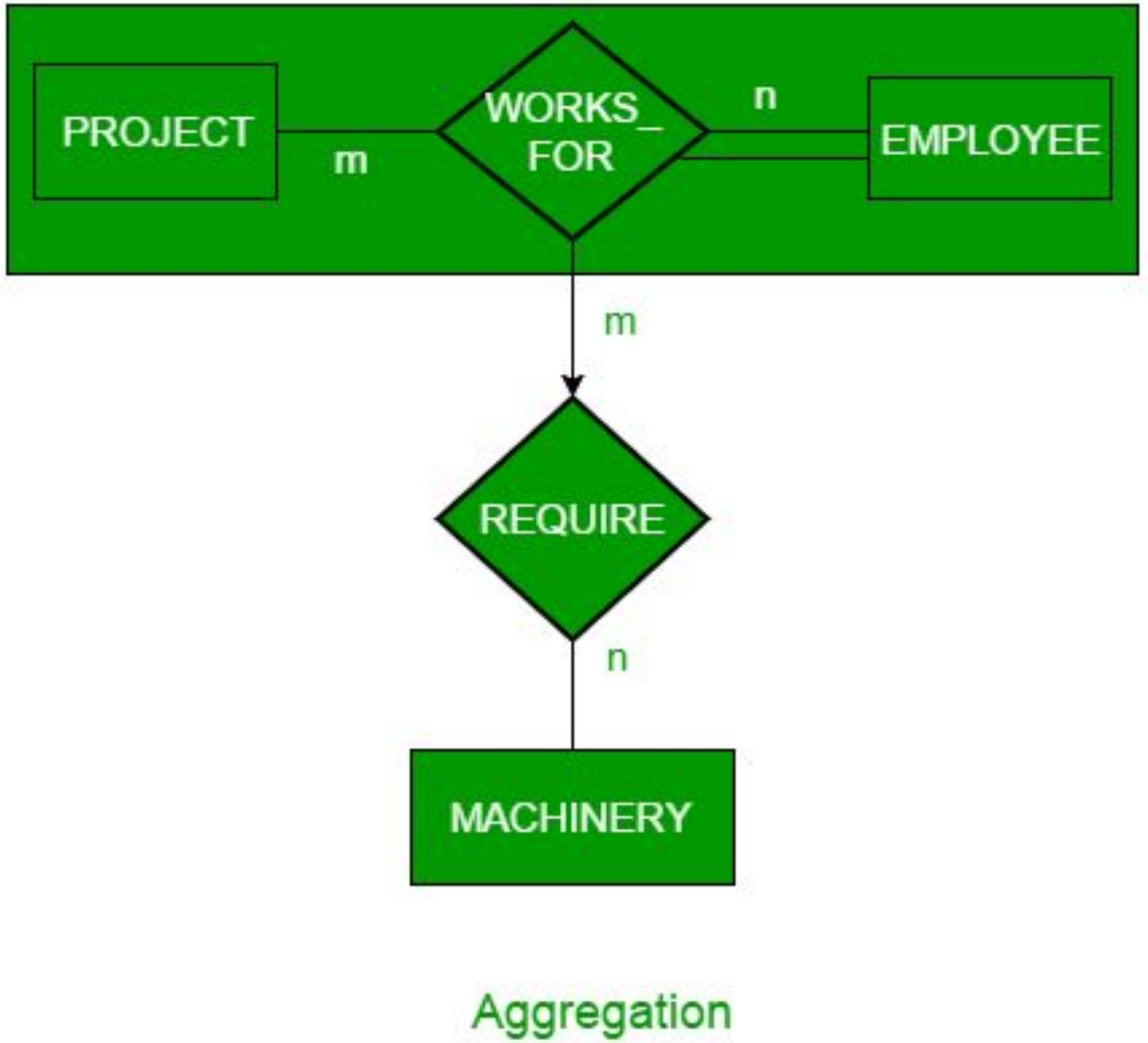


Specialization

Aggregation

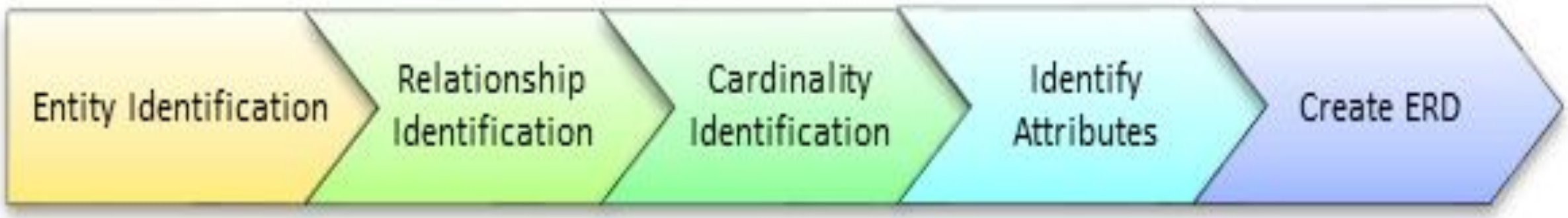
- An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios.
- In those cases, a relationship with its corresponding entities is aggregated into a higher level entity.

For Example, Employee working for a project may require some machinery. So, REQUIRE relationship is needed between relationship WORKS_FOR and entity MACHINERY. Using aggregation, WORKS_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationship REQUIRE is created between aggregated entity and MACHINERY.



Aggregation

Steps to Create an ERD

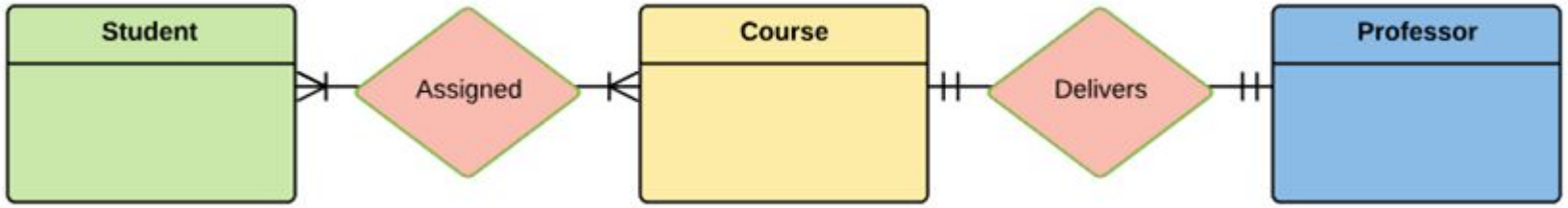


- In a university, a Student enrolls in Courses. A student must be assigned to at least one or more Courses. Each course is taught by a single Professor. To maintain instruction quality, a Professor can deliver only one course

- **Step 1) Entity Identification**
- We have three entities
- Student
- Course
- Professor
-

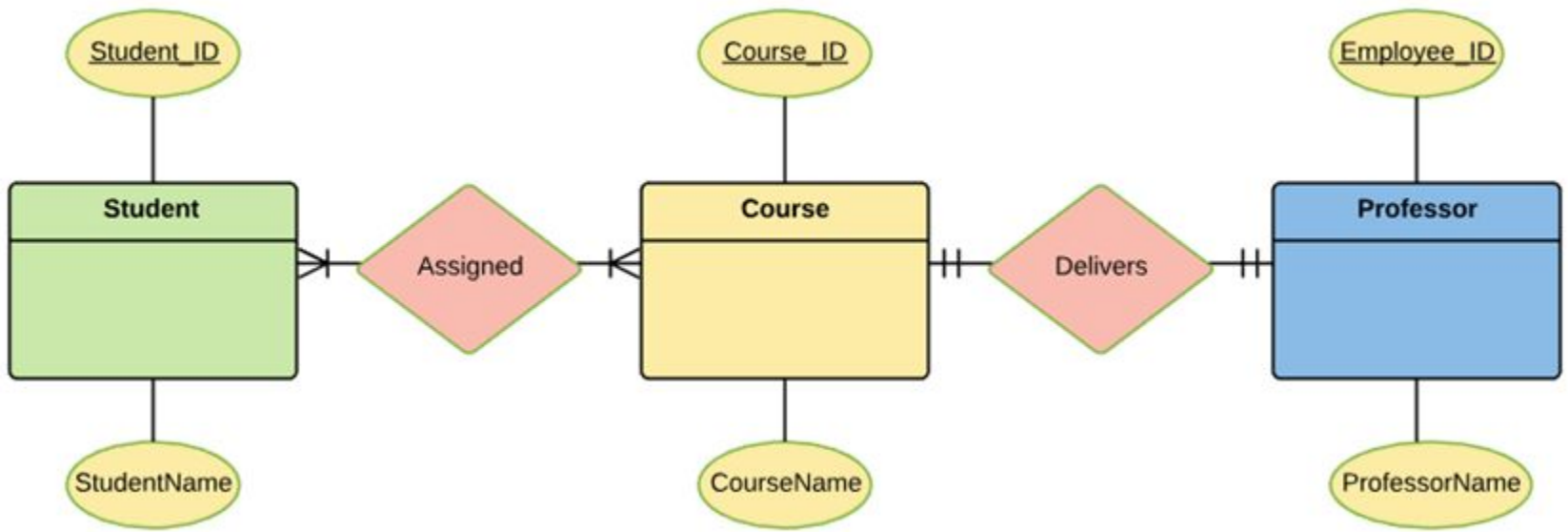
- **Step 2) Relationship Identification**
- We have the following two relationships
- The student is **assigned** a course
- Professor **delivers** a course

- **Step 3) Cardinality Identification**
- For them problem statement we know that,
- A student can be assigned **multiple** courses
- A Professor can deliver only **one** course



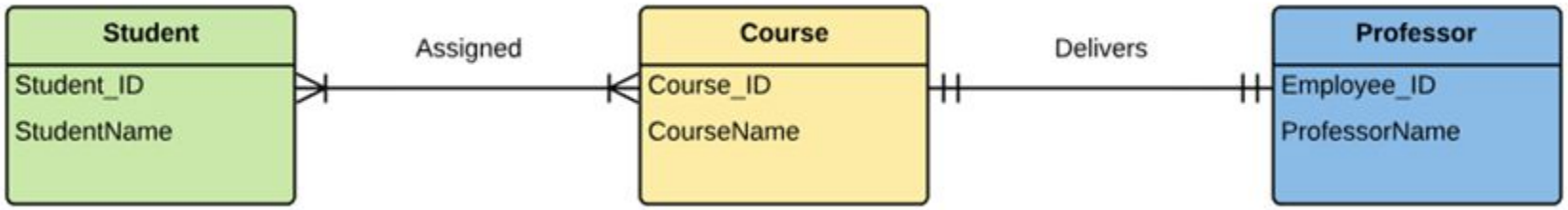
- **Step 4) Identify Attributes**
- You need to study the files, forms, reports, data currently maintained by the organization to identify attributes. You can also conduct interviews with various stakeholders to identify entities. Initially, it's important to identify the attributes without mapping them to a particular entity.
- Once, you have a list of Attributes, you need to map them to the identified entities. Ensure an attribute is to be paired with exactly one entity. If you think an attribute should belong to more than one entity, use a modifier to make it unique.
- Once the mapping is done, identify the primary Keys. If a unique key is not readily available, create one.

Entity	Primary Key	Attribute
Student	Student_ID	StudentName
Professor	Employee_ID	ProfessorName
Course	Course_ID	CourseName


Student_ID
Course_ID
Employee_ID
Student
Assigned
Course
Delivers
Professor
StudentName
CourseName
ProfessorName

Step 5) Create the ERD

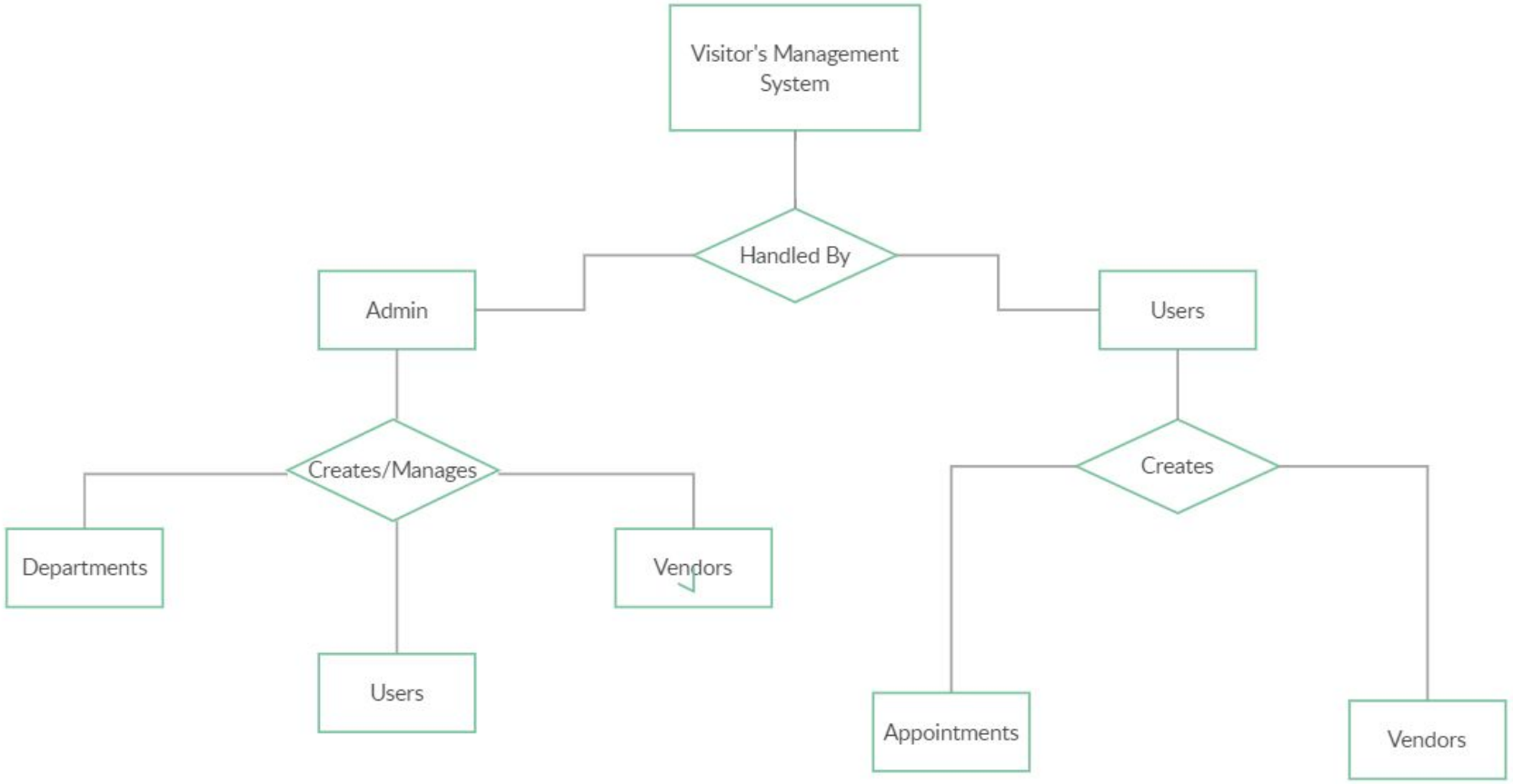
A more modern representation of ERD Diagram



Short for ER Diagram

- The ER model is a high-level data model diagram
- ER diagrams are a visual tool which is helpful to represent the ER model
- Entity relationship diagram displays the relationships of entity set stored in a database
- ER diagrams help you to define terms related to entity relationship modeling
- ER model is based on three basic concepts: Entities, Attributes & Relationships
- An entity can be place, person, object, event or a concept, which stores data in the database
- Relationship is nothing but an association among two or more entities
- A weak entity is a type of entity which doesn't have its key attribute
- It is a single-valued property of either an entity-type or a relationship-type
- It helps you to defines the numerical attributes of the relationship between two entities or entity sets
- ER- Diagram is a visual representation of data that describe how data is related to each other
- While Drawing ER diagram you need to make sure all your entities and relationships are properly labeled.

Visitors Management System



Hostel Management System