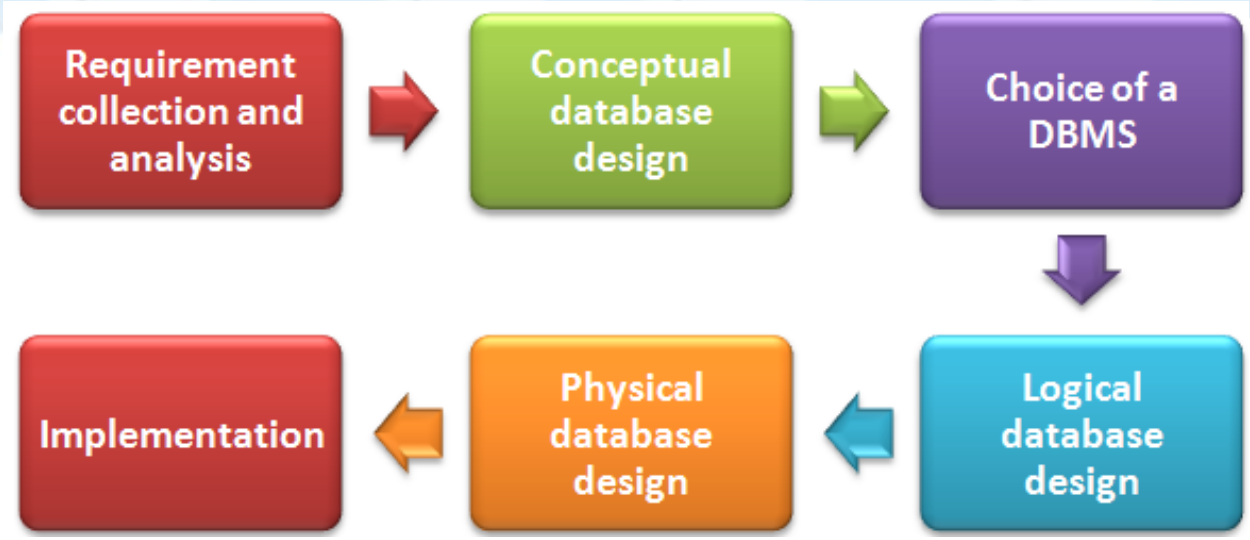
UNIT II – Database Design, Architecture & Model – 6 HRS

Overview of Database Design Process

- Database systems are designed to manage large bodies of information.
- Database design mainly involves the design of the database schema. The design of a complete database application environment that meets the needs of the enterprise being modeled requires attention to a broader set of issues.
- In this text, we focus initially on the writing of database queries and the design of database schemas.
- Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of database systems.
- It helps produce database systems
    1. That meet the requirements of the users
    2. Have high performance.

Overview of Database Design Process



- A high-level data model provides the database designer with a conceptual framework in which to specify the data requirements of the database users, and how the database will be structured to fulfill these requirements.
- The **initial phase** of database design, then, is to characterize fully the data needs of the prospective database users. The database designer needs to interact extensively with domain experts and users to carry out this task. The **outcome** of this phase is a **specification of user requirements**.

Overview of Database Design Process

- Next, the designer chooses a data model, and by applying the concepts of the chosen data model, translates these requirements into a **conceptual schema of the database**.
- The schema developed at this **conceptual-design phase** provides a detailed overview of the enterprise. The designer reviews the schema to confirm that all data requirements are indeed satisfied and are not in conflict with one another. The designer can also examine the design to remove any redundant features. The focus at this point is on describing the data and their relationships, rather than on specifying physical storage details.
- In terms of the relational model, the conceptual-design process involves decisions on what attributes we want to capture in the database and how to group these attributes to form the various tables.

Overview of Database Design Process

- A fully developed conceptual schema indicates the functional requirements of the enterprise. In a specification of functional requirements, users describe the kinds of operations (or transactions) that will be performed on the data.
- Example operations include modifying or updating data, searching for and retrieving specific data, and deleting data. At this stage of conceptual design, the designer can review the schema to ensure it meets functional requirements.
- The process of moving from an abstract data model to the implementation of the database proceeds in **two final design phases**.
- In the **logical-design phase**, the designer maps the high-level conceptual schema onto the implementation data model of the database system that will be used.
- The designer uses the resulting system-specific database schema in the subsequent **physical-design phase**, in which the physical features of the database are specified.

Database Design for a University Organization

- To illustrate the design process, let us examine how a database for a university could be designed. The initial specification of user requirements may be based on interviews with the database users, and on the designer's own analysis of the organization.
- The description that arises from this design phase serves as the basis for specifying the conceptual structure of the database. Here are the **major characteristics** of the university.

✓ The university is organized into departments. Each department is identified by a unique name (dept name), is located in a particular building, and has a budget.

✓ Each department has a list of courses it offers. Each course has associated with it a course id, title, dept name, and credits.

✓ Instructors are identified by their unique ID. Each instructor has name, associated department (dept name), and salary.

Database Design for a University Organization

- ✓ Students are identified by their unique ID. Each student has a name, an associated major department (dept name), and tot cred (total credit hours the student earned thus far).
- ✓ The university maintains a list of classrooms, specifying the name of the building, room number, and room capacity.
- ✓ The university maintains a list of all classes (sections) taught. Each section is identified by a course id, sec id, year, and semester, and has associated with it a semester, year, building, room number, and time slot id (the time slot when the class meets).
- ✓ The department has a list of teaching assignments specifying, for each instructor, the sections the instructor is teaching.
- ✓ The university has a list of all student course registrations, specifying, for each student, the courses and the associated sections that the student has taken (registered for).

View of Data – Data Abstraction

- A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.
- A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained.
- For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database.
- Since many database-system users are not computer trained, developers hide the complexity from users through **several levels of abstraction**, to simplify users' interactions with the system:

View of Data - Data Abstraction

- ✓ **Physical level.** The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.
- ✓ **Logical level.** The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.
- ✓ **View level.** The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

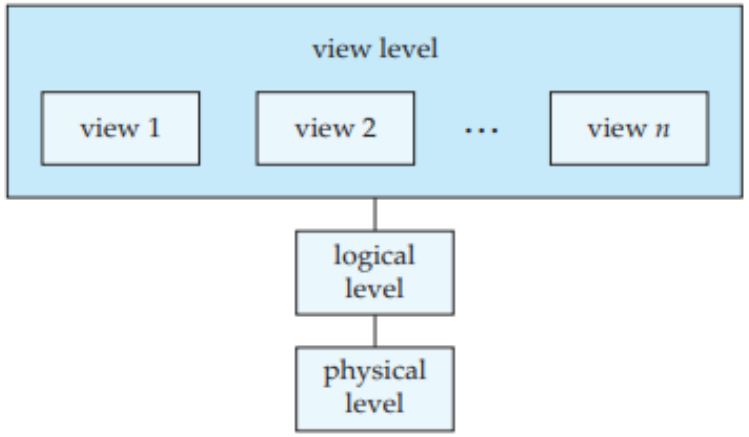


Figure 1.1 The three levels of data abstraction.

Data Models

- **Data Model** a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. A data model provides a way to describe the design of a database at the physical, logical, and view levels.
- The data models can be classified into four different categories:

✓ **Relational Model.** The relational model uses a collection of tables to represent both data and the relationships among those data.

✓ Each table has multiple columns, and each column has a unique name. Tables are also known as relations. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes.

✓ The columns of the table correspond to the attributes of the record type. The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.

✓ **Entity-Relationship Model.** The entity-relationship (E-R) data model uses a collection of basic objects, called entities, and relationships among these objects. An entity is a “thing” or “object” in the real world that is distinguishable from other objects. The entity-relationship model is widely used in database design.

Data Models

- ✓ **Object-Based Data Model.** Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology. This led to the development of an object-oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity. The object-relational data model combines features of the object-oriented data model and relational data model.

- ✓ **Semistructured Data Model.** The semistructured data model permits the specification of data where individual data items of the same type may have different sets of attributes. The **Extensible Markup Language (XML)** is widely used to represent semistructured data.

- ✓ Historically, **the network data model and the hierarchical data model** preceded the relational data model. These models were tied closely to the underlying implementation, and complicated the task of modeling data. As a result they are used little now, except in old database code that is still in service in some places.

Database Languages

- ✓ A database system provides a **data-definition language** to specify the database schema and a **data-manipulation language** to express database queries and updates.

Data-Manipulation Language(DML)

- ✓ A **data-manipulation language (DML)** is a language that enables users to access or manipulate data as organized by the appropriate data model.
- ✓ **Examples of DML:**
- **SELECT** – is used to retrieve data from the a database.
- **INSERT** – is used to insert data into a table.
- **UPDATE** – is used to update existing data within a table.
- **DELETE** – is used to delete records from a database table.
- ✓ There are basically two types:
    - **Procedural DMLs** require a user to specify what data are needed and how to get those data.
    - **Declarative DMLs (also referred to as nonprocedural DMLs)** require a user to specify what data are needed without specifying how to get those data.

Database Languages

Data-Definition Language (DDL)

- ✓ **DDL or Data Definition Language** actually consists of the SQL commands that can be used to define the database schema.
- ✓ It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in database.

- ✓ **Examples of DDL commands:**

- **CREATE** – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- **DROP** – is used to delete objects from the database.
- **ALTER**-is used to alter the structure of the database.
- **TRUNCATE**–is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT** –is used to add comments to the data dictionary.
- **RENAME** –is used to rename an object existing in the database.

Database Languages

Data Control Language (DCL)

- ✓ DCL includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.
- ✓ **Examples of DCL commands:**
- **GRANT**-gives user's access privileges to database.
- **REVOKE**-withdraw user's access privileges given by using the GRANT command.

Transaction Control Language (TCL)

- ✓ TCL commands deals with the transaction within the database.
- ✓ **Examples of TCL commands:**
- **COMMIT**– commits a Transaction.
- **ROLLBACK**– rollbacks a transaction in case of any error occurs.
- **SAVEPOINT**– sets a save point within a transaction.
- **SET TRANSACTION**– specify characteristics for the transaction.

Query By Example (QBE)

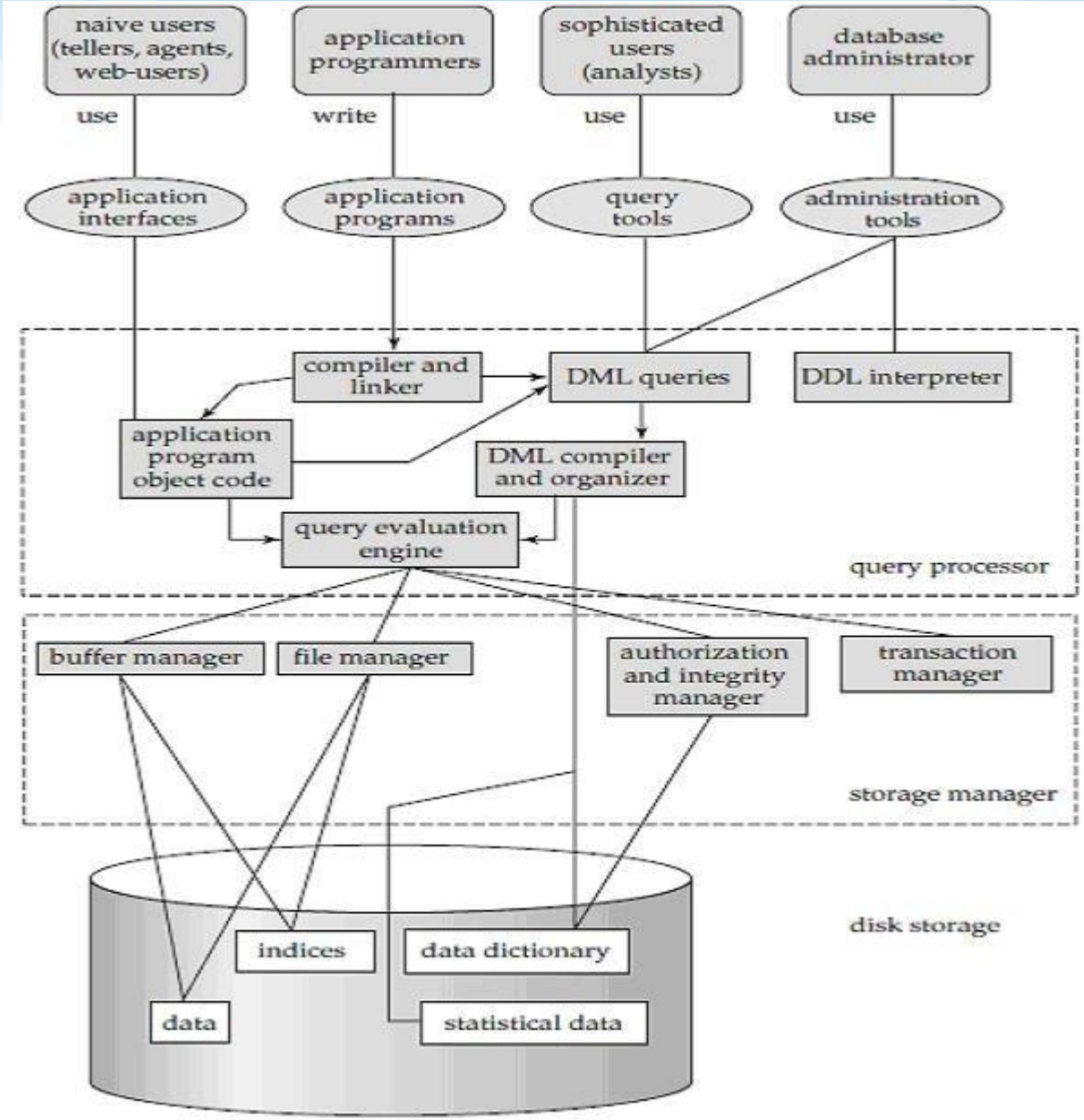
- ✓ **Query by example** is a query language used in relational databases that allows users to search for information in tables and fields by providing a simple user interface where the user will be able to input an example of the data that he or she wants to access.
- ✓ The principle of QBE is that it is merely an abstraction between the user and the real query that the database system will receive. In the background, the user's query is transformed into a database manipulation language form such as SQL, and it is this SQL statement that will be executed in the background.
- ✓ (QBE) is a method of query creation that allows the user to search for documents based on an example in the form of a selected text string or in the form of a document name or a list of documents.
- ✓ Because the QBE system formulates the actual query, QBE is easier to learn than formal query languages, such as the standard Structured Query Language (SQL), while still enabling powerful searches.
- ✓ In terms of database management system, QBE can be thought of as a "fill-in-the blanks" method of query creation. The Microsoft Access Query Design Grid is an example. To conduct a search for field data matching particular conditions, the user enters criteria into the form, creating search conditions for as many fields as desired. A query is automatically generated to search the database for matching data.

DBMS Structure

- ✓ A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the **storage manager** and the **query processor** components.

- ✓ The **storage manager** is important because databases typically require a large amount of storage space.

- ✓ The **query processor** is important because it helps the database system simplify and facilitate access to data.



DBMS Structure

Query Processor

- ✓ The query processor components include:
- **DDL interpreter**, which interprets DDL statements and records the definitions in the data dictionary.
- **DML compiler**, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
- ✓ A query can usually be translated into any of a number of alternative evaluation plans that all give the same result.
- ✓ The DML compiler also performs query optimization, that is, it picks the lowest cost evaluation plan from among the alternatives.

Storage Manager

- ✓ A storage manager is a program module that provides the interface between the low level data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for the interaction with the file manager.
- ✓ It is responsible for storing, retrieving, and updating data in the database.

DBMS Structure

Storage Manager

The storage manager components include:

- ✓ **Authorization and integrity manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.

- ✓ **Transaction manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.

- ✓ **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

- ✓ **Buffer manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

DBMS Structure

Transaction Manager

- ✓ A transaction is a collection of operations that performs a single logical function in a database application.
- ✓ Each transaction is a unit of both atomicity and consistency. Thus, we require that transactions do not violate any database-consistency constraints.
- ✓ That is, if the database was consistent when a transaction started, the database must be consistent when the transaction successfully terminates.
- ✓ Transaction - manager ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

DBMS Architecture

- ✓ **DBMS architecture** helps in design, development, implementation, and maintenance of a database. A database stores critical information for a business. Selecting the correct Database Architecture helps in quick and secure access to this data.

1 Tier Architecture

Single Tier Architecture

- ✓ The simplest of Database Architecture are 1 tier where the Client, Server, and Database all reside on the same machine.
- ✓ Anytime you install a DB in your system and access it to practise SQL queries it is 1 tier architecture. But such architecture is rarely used in production.

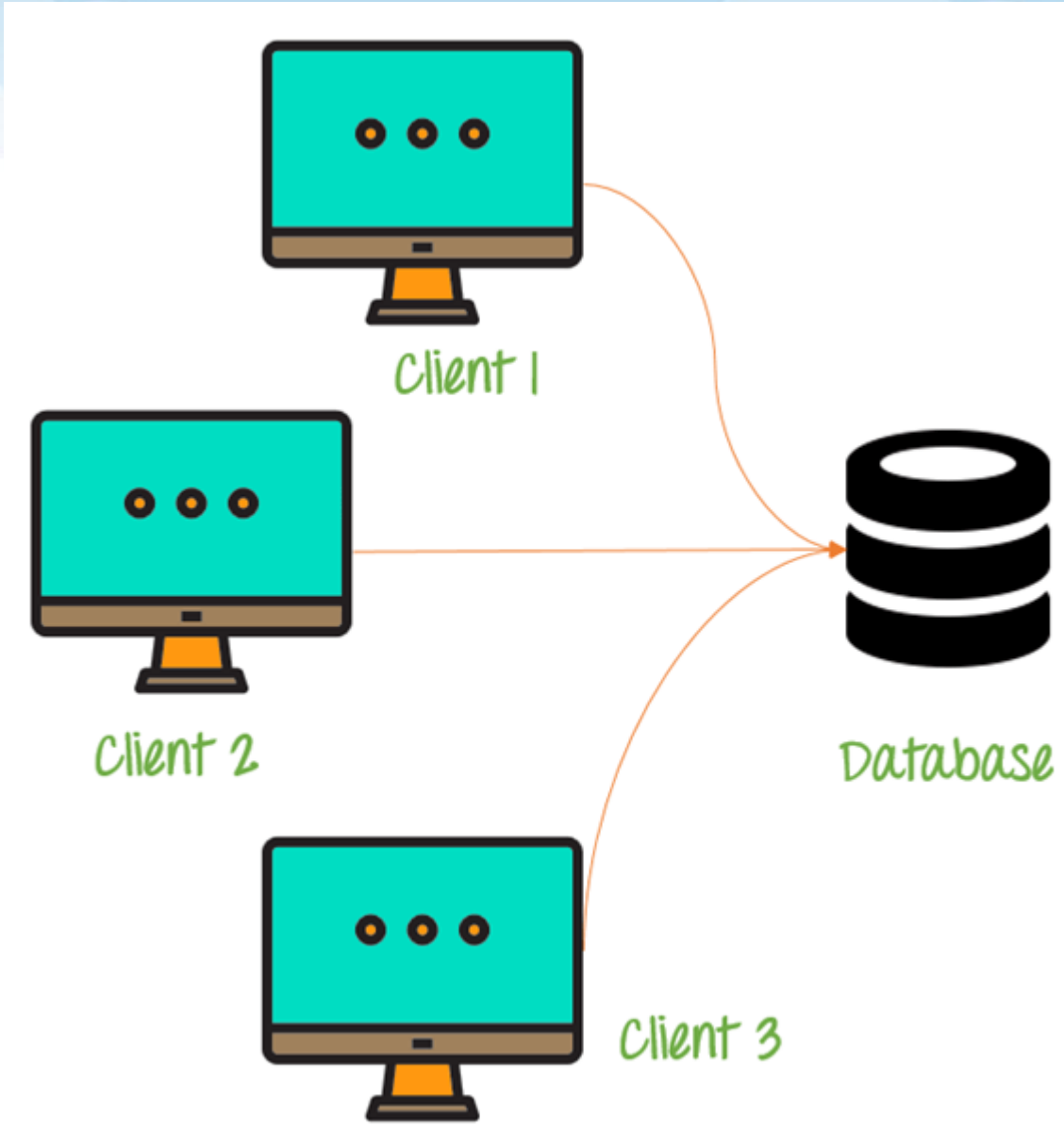
DBMS Architecture

2 Tier Architecture

- ✓ A two-tier architecture is a database architecture where:

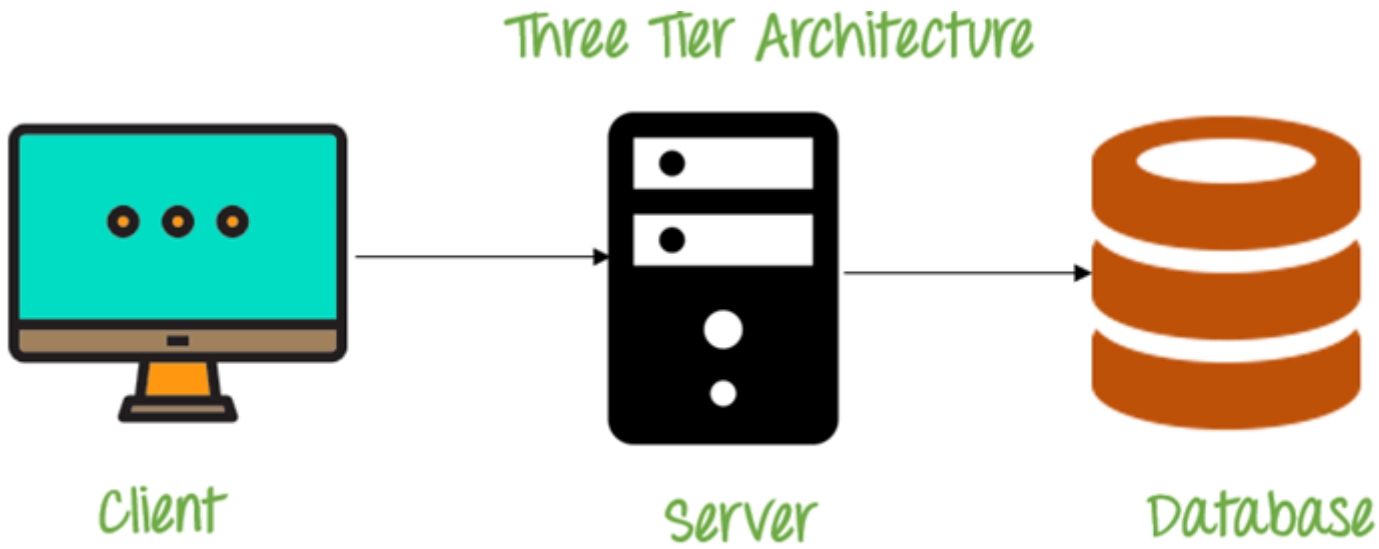
1. Presentation layer runs on a client (PC, Mobile, Tablet, etc.)
2. Data is stored on a Server.

- ✓ An application interface which is called ODBC (Open Database Connectivity) an API which allows the client-side program to call the DBMS.
- ✓ Today most of the DBMS offers ODBC drivers for their DBMS. 2 tier architecture provides added security to the DBMS as it is not exposed to the end user directly.



DBMS Architecture

3 Tier Architecture



✓3-tier schema is an extension of the 2-tier architecture. 3-tier architecture has following layers:

1.Presentation layer (your PC, Tablet, Mobile, etc.)

2.Application layer (server)

3.Database Server

DBMS Architecture

3 Tier Architecture

- ✓ This DBMS architecture contains an Application layer between the user and the DBMS, which is responsible for communicating the user's request to the DBMS system and send the response from the DBMS to the user.
- ✓ The application layer(business logic layer) also processes functional logic, constraint, and rules before passing data to the user or down to the DBMS
- ✓ Three tier architecture is the most popular DBMS architecture.

- ✓ **The goal of Three-tier architecture is:**
- To separate the user applications and physical database
- Proposed to support DBMS characteristics
- Program-data independence
- Support of multiple views of the data

- ✓ **Example of Three-tier Architecture** is any large website on the internet.

Data Independence

- ✓ The main purpose of **data abstraction** is achieving **data independence** in order to save time and cost required when the database is modified or altered.
- ✓ **Data independence** is the ability of to make changes to **data** characteristics without have to make changes to the programs that access the **data**. It's **important** because of the savings in time and potential errors caused by reducing modifications to **data** access software.

Physical level data independence :

- ✓ It refers to the characteristic of being able to modify the physical schema without any alterations to the conceptual or logical schema.
- ✓ These alterations or modifications to the physical structure may include:
- Utilising new storage devices.
- Modifying data structures used for storage.
- Altering indexes or using alternative file organisation techniques etc.

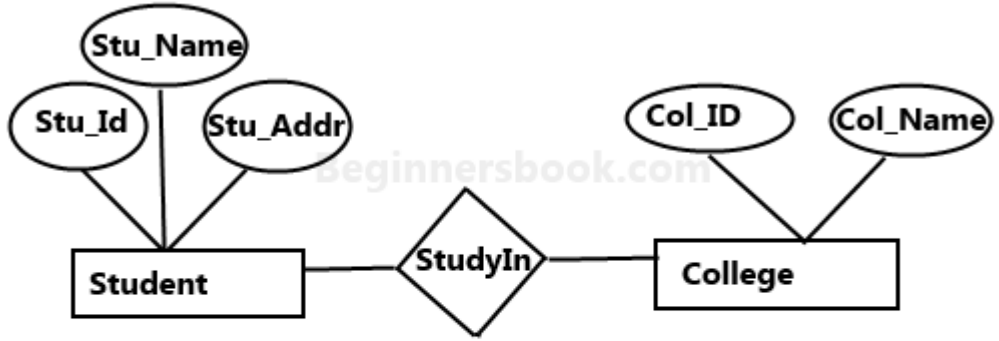
Data Independence

Logical level data independence :

- ✓ It refers characteristic of being able to modify the logical schema without affecting the external schema or application program.
- ✓ The user view of the data would not be affected by any changes to the conceptual view of the data.
- ✓ These changes may include insertion or deletion of attributes, altering table structures entities or relationships to the logical schema etc.

ER Model - Basic Concepts

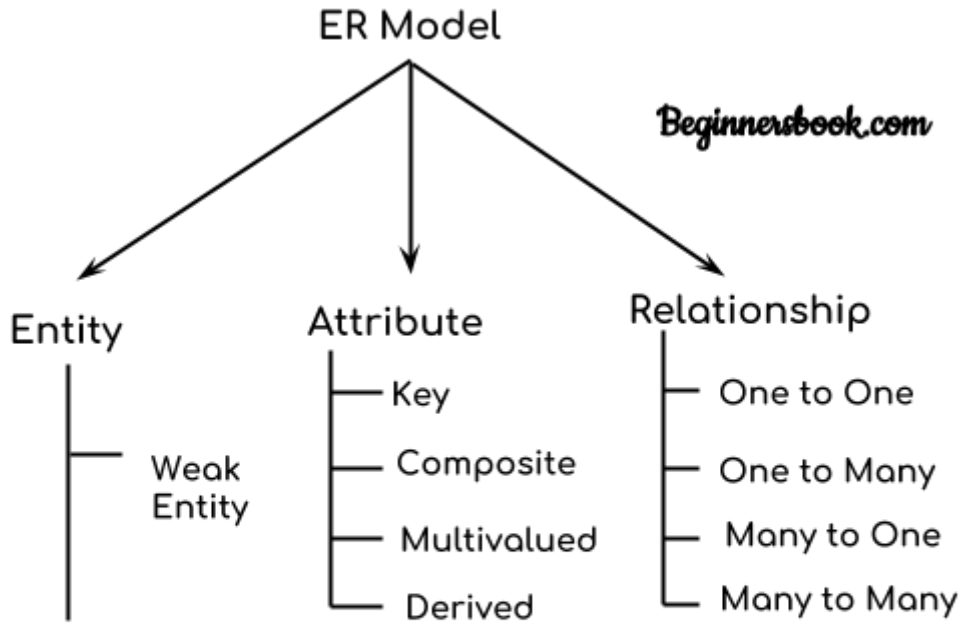
- ✓ The **ER model** defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.
- ✓ **Entity Relationship Diagram**, also known as **ERD**, **ER Diagram** or **ER** model, is a type of structural **diagram** for use in database design.
- ✓ An **ERD** contains different symbols and connectors that visualize two important information: The major entities within the system scope, and the inter-relationships among these entities.



Sample E-R Diagram

Components of a ER Diagram

- ✓ **Rectangle**: Represents Entity sets.
- ✓ **Ellipses**: Attributes
- ✓ **Diamonds**: Relationship Set
- ✓ **Lines**: They link attributes to Entity Sets and Entity sets to Relationship Set
- ✓ **Double Ellipses**: Multivalued Attributes
- ✓ **Dashed Ellipses**: Derived Attributes
- ✓ **Double Rectangles**: Weak Entity Sets
- ✓ **Double Lines**: Total participation of an entity in a relationship set

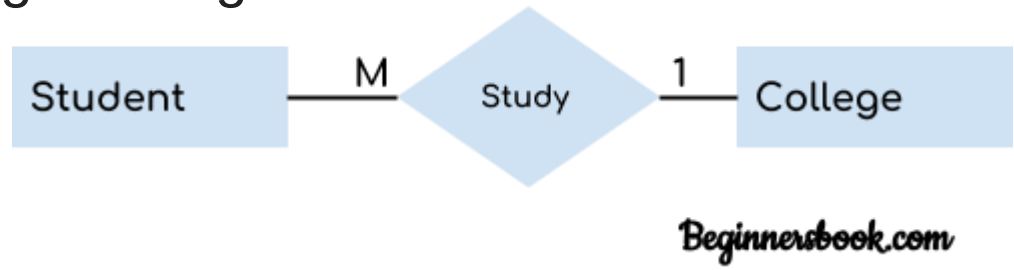


Components of ER Diagram

Components of a ER Diagram

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.



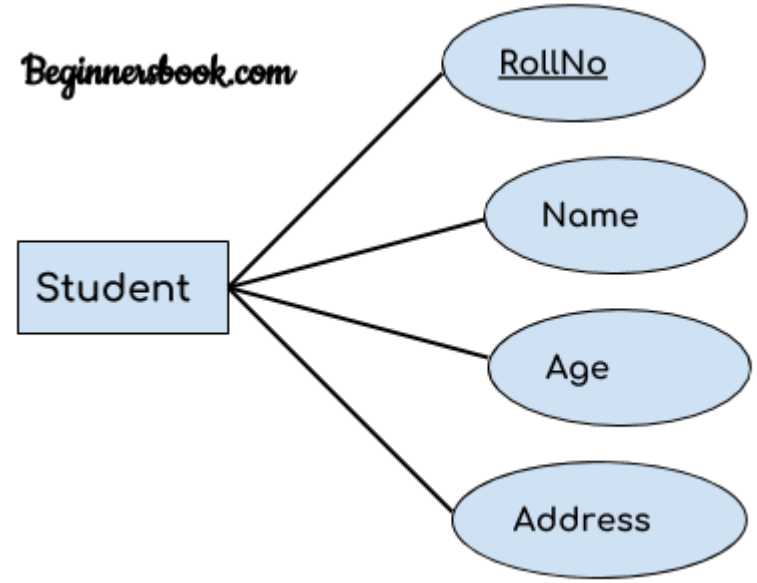
Weak Entity:

- ✓ 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.

Components of a ER Diagram

2. 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:
  - a) Key attribute
  - b) Composite attribute
  - c) Multivalued attribute
  - d) Derived attribute



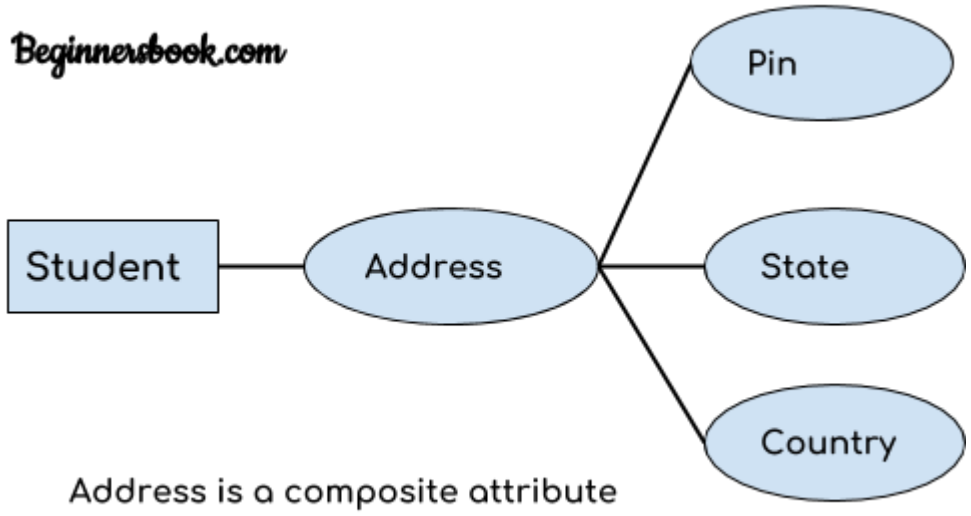
a) Key attribute

- ✓ 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**.

Components of a ER Diagram

b) 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.



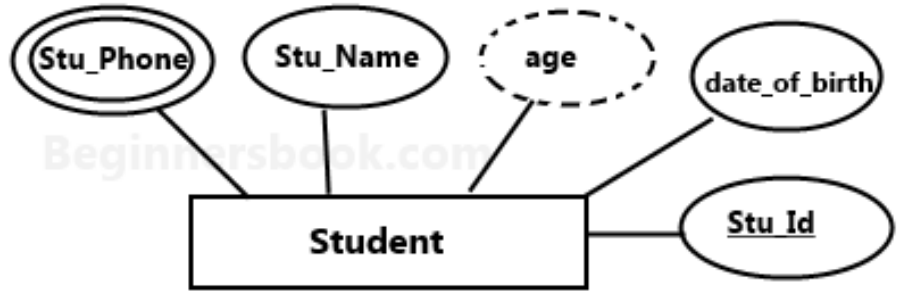
Components of a ER Diagram

c) 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.

d) 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).



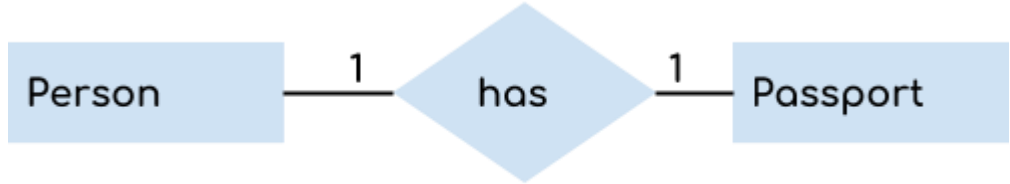
Components of a ER Diagram

3. Relationship

- ✓ A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:
  - a) One to One
  - b) One to Many
  - c) Many to One
  - d) Many to Many

a) One to One

- ✓ 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.



Components of a ER Diagram

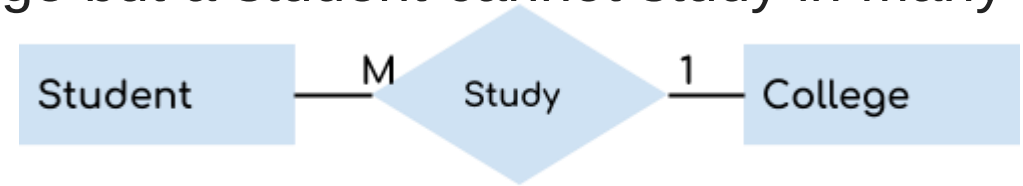
b) One to Many

✓ 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.

Beginnersbook.com

c) Many to One

✓ 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.

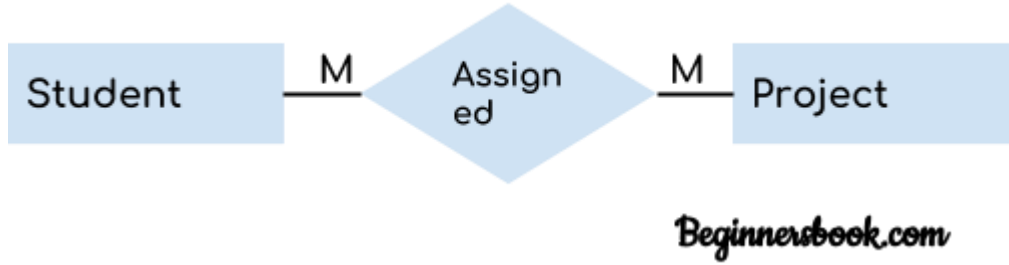


Beginnersbook.com

Components of a ER Diagram

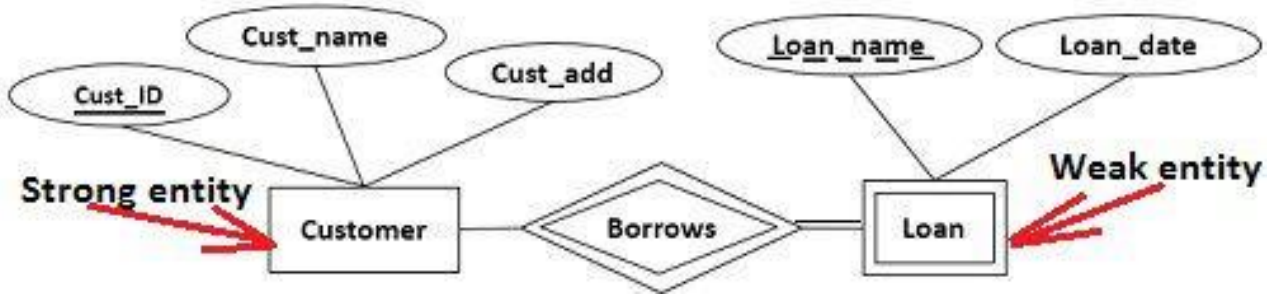
d) Many to Many

- ✓ 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.



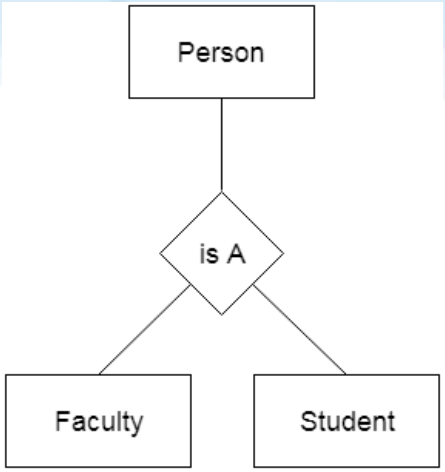
Strong Entity Set Vs Weak Entity Set

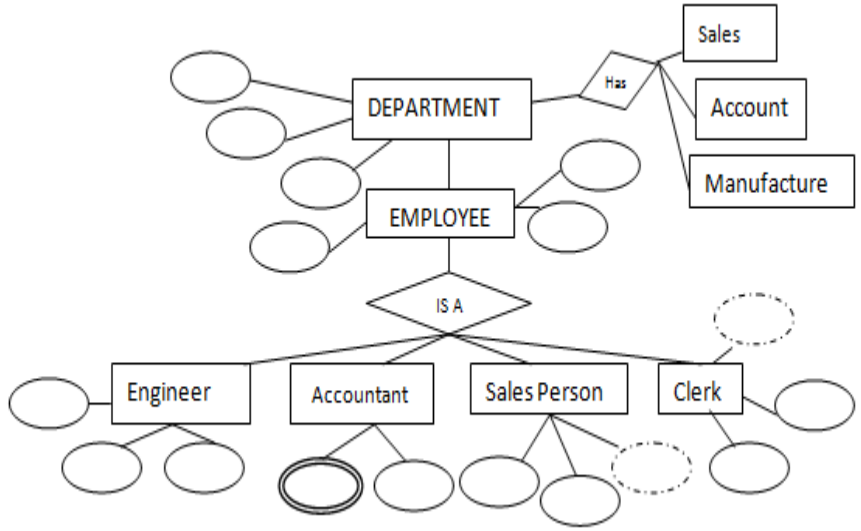
1. The basic difference between strong entity and a weak entity is that the strong entity has a **primary key** whereas, a weak entity has the **partial key** which acts as a discriminator between the entities of a weak entity set.
2. A weak entity always **depends** on the strong entity for its existence whereas, a strong entity is **independent** of any other entity's existence.
3. A strong entity is denoted with a **single rectangle** and a weak entity is denoted with a **double rectangle**.
4. The relationship between two strong entities is denoted with **single diamond** whereas, a relationship between a weak and a strong entity is denoted with double diamond called **Identifying Relationship**.



Generalization

- ✓ **Generalization** is like a bottom-up approach in which **two or more entities of lower level** combine to form **a higher level entity** if they have some attributes in common.
- ✓ In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.
- ✓ Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.
- ✓ In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.
- ✓ **For example**, **Faculty and Student** entities can be generalized and create a higher level entity **Person**.

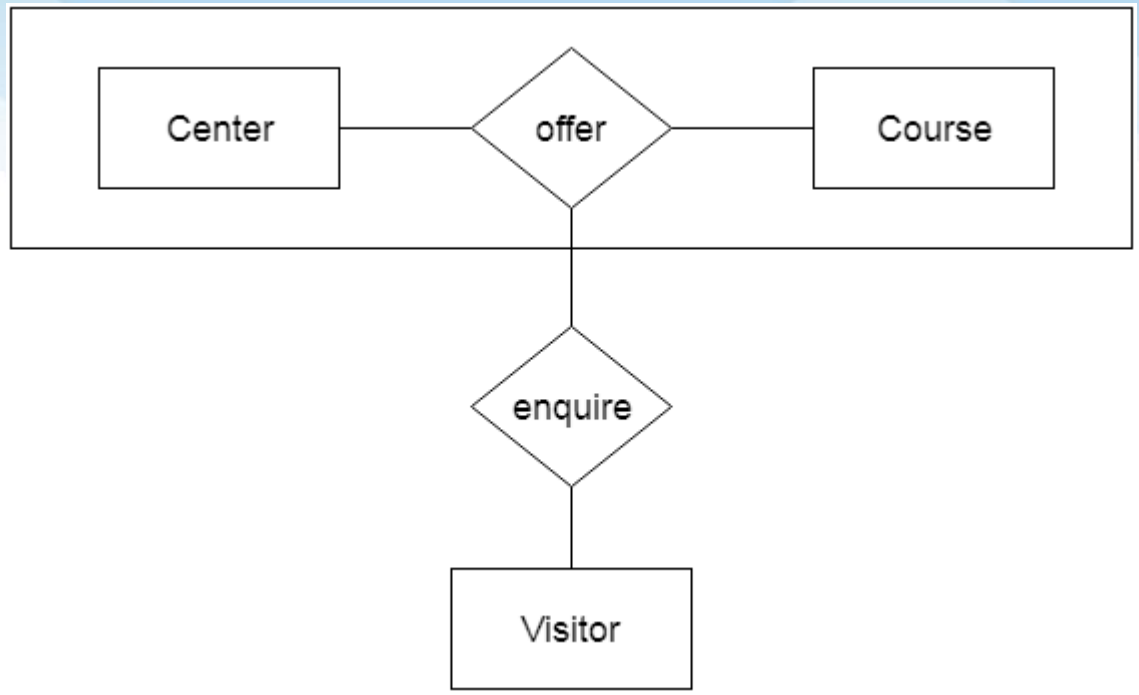


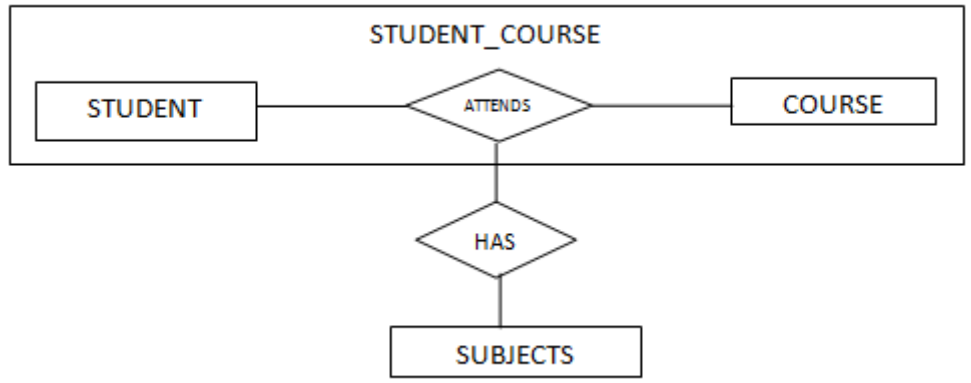


Aggregation

- ✓ In aggregation, the relation between two entities is treated as a single entity.
- ✓ In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

- ✓ **For example:** Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor.
- ✓ In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.





Converting ER-Diagram to Table

Rules:

- ✓ Entity type becomes a table.
- ✓ All single-valued attribute becomes a column for the table.
- ✓ A key attribute of the entity type represented by the primary key.
- ✓ The multivalued attribute is represented by a separate table.
- ✓ Composite attribute represented by components.
- ✓ Derived attributes are not considered in the table.