Syllabus: 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, Conceptual Design for Large Enterprises .

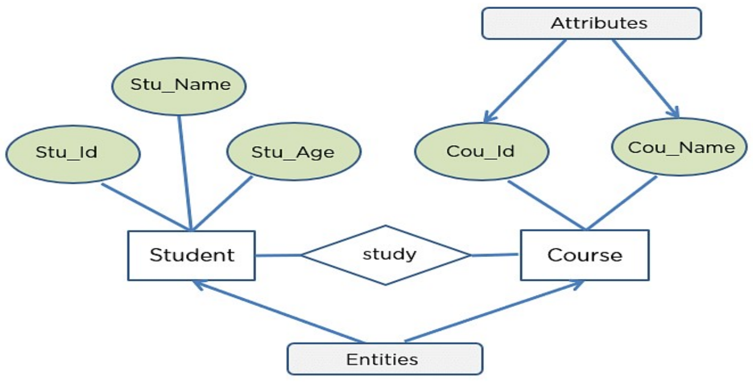
Relational Model: Introduction to the Relational Model, Integrity Constraints over Relations, Enforcing Integrity Constraints, Querying Relational Data, Logical Database Design: ER to Relational, Introduction to Views, Destroying/Altering Tables and View

Introduction to Database Design:

- Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system.
- Database designs provide the blueprints of how the data is going to be stored in a system. A proper design of a database highly affects the overall performance of any application.
- The designing principles defined for a database give a clear idea of the behavior of any application and how the requests are processed. There are two types database design:
- **Logical model** – This stage is concerned with developing a database model based on requirements. The entire design is on paper without any physical implementations or specific DBMS considerations.
- **Physical model** – This stage implements the logical model of the database taking into account the DBMS and physical implementation factors.

ER Diagrams:

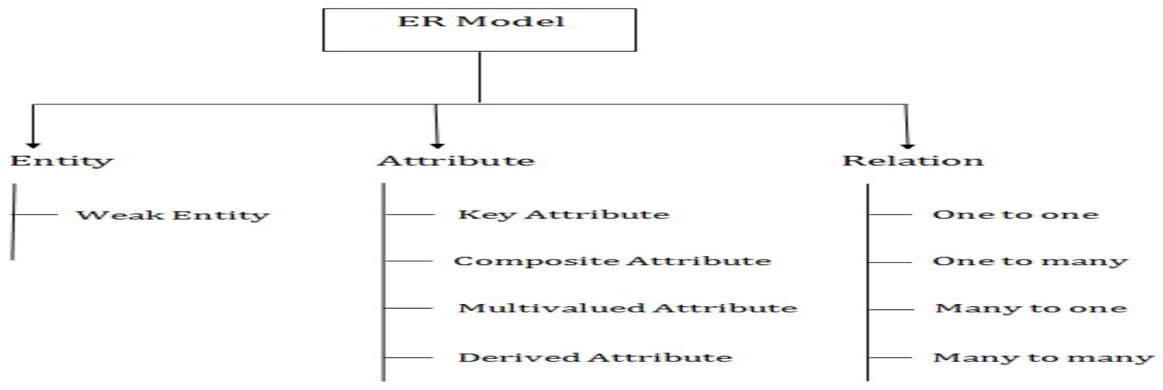
An Entity Relationship Diagram is a diagram that represents relationships among entities in a database. It is commonly known as an ER Diagram. An ER Diagram in DBMS plays a crucial role in designing the database. An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.



ER diagram has three main components:

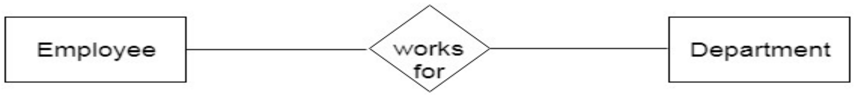
1. Entity
2. Attribute
3. Relationship

- **Component of ER Diagram**



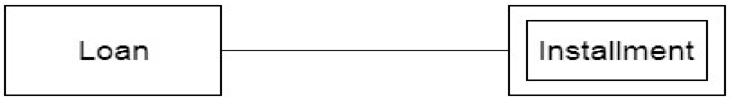
Entity:

An entity is an object or component of data. An entity is represented as rectangle in an ER diagram. Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



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.



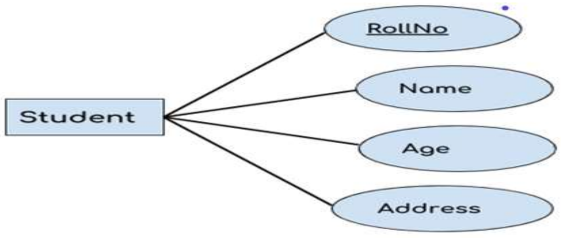
Attribute:

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. Multi valued attribute
4. Derived attribute

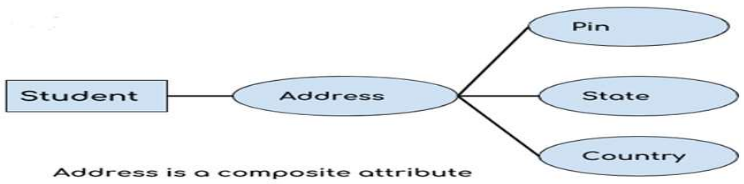
1. Key attribute:

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**.



2. 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.

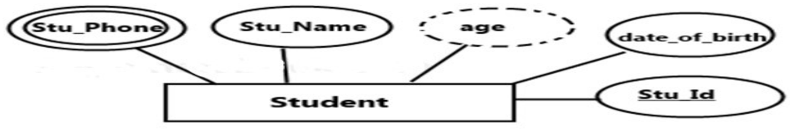


3. Multi valued attribute:

An attribute that can hold multiple values is known as multi valued 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 multi valued.

4. 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).



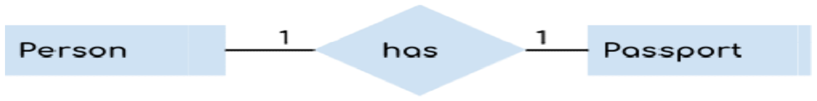
Relationship:

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

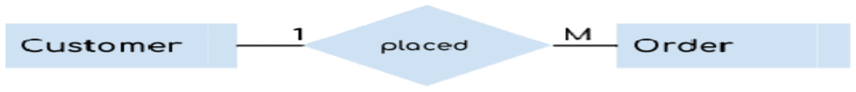
1. **One to One Relationship:**

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.



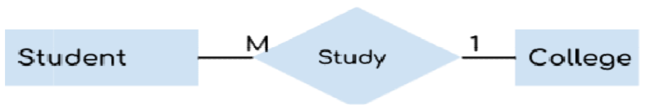
2. **One to Many Relationship:**

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.



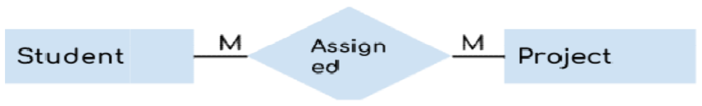
3. **Many to One Relationship:**

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.



4. **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.



****

Entity Set:

An entity set is a set of same type of entities.
An entity refers to any object having-

- Either a physical existence such as a particular person, office, house or car.
- Or a conceptual existence such as a school, a university, a company or a job.

In ER diagram,

- Attributes are associated with an entity set.
- Attributes describe the properties of entities in the entity set.
- Based on the values of certain attributes, an entity can be identified uniquely.

Relationship Sets:

A relationship set is a set of relationships of same type

Degree of a Relationship Set-

The number of entity sets that participate in a relationship set is termed as the degree of that relationship set. Thus,

Degree of a relationship set = Number of entity sets participating in a relationship set

Types of Relationship Sets-

On the basis of degree of a relationship set, a relationship set can be classified into the following types-

1. Unary relationship set
2. Binary relationship set
3. Ternary relationship set
4. N-ary relationship set

1. Unary Relationship Set-

Unary relationship set is a relationship set where only one entity set participates in a relationship set.

Example-One person is married to only one person

2. Binary Relationship Set-

Binary relationship set is a relationship set where two entity sets participate in a relationship set.

Example-Student is enrolled in a Course

3. Ternary Relationship Set-

Ternary relationship set is a relationship set where three entity sets participate in a relationship set.

4. N-ary Relationship Set-

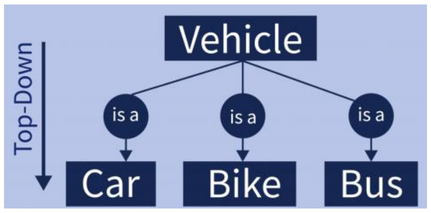
N-ary relationship set is a relationship set where 'n' entity sets participate in a relationship set.

****

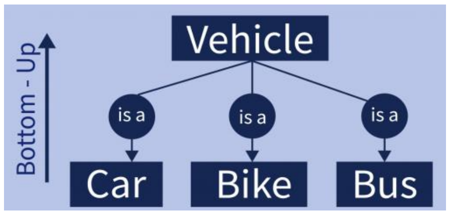
Additional Features of the ER Model:

The basic E-R concepts can model most database features, some aspects of a database may be more aptly expressed by certain extensions to the basic E-R model. The extended E-R features are specialization, generalization, higher- and lower-level entity sets, attribute inheritance, and aggregation.

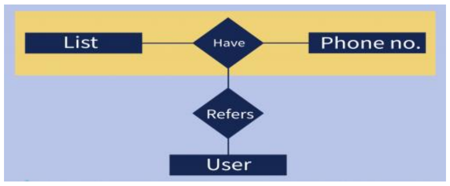
❖ **Specialization –** An entity set broken down sub-entities that are distinct in some way from other entities in the set. For instance, a subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set. The E-R model provides a means for representing these distinctive entity groupings.
Specialization is an "aTop-down approach" where a high-level entity is specialized into two or more level entities.
Example – Consider an entity set vehicle, with attributes color and no. of tires. A vehicle may be further classified as one of the following:
- Car
- Bike
- Bus



❖ **Generalization –** It is a process of extracting common properties from a set of entities and creating a generalized entity from it. Generalization is a "Bottom-up approach". In which two or more entities can be combined to form a higher-level entity if they have some attributes in common.
In generalization, Subclasses are combined to make a super class
Example: There are three entities given, car, bus, and bike. They all have some common attributes like all cars, buses, and bikes they all have no. of tires and have some colors. So they all can be grouped and make a superclass named a vehicle.



- **Inheritance –** An entity that is a member of a subclass inherits all the attributes of the entity as the member of the superclass, the entity also inherits all the relationships that the superclass participates in. Inheritance is an important feature of Generalization and Specialization. It allows lower-level entities to inherit the attributes of higher-level entities.
  Example – Car, bikes, and buses inherit the attributes of a vehicle. Thus, a car is described by its color and no. of tires, and additionally a gear attribute; a bike is described by its color and no. of tires attributes, and additionally automatic break attribute.
- **Aggregation –** In aggregation, the relation between two entities is treated as a single entity. In aggregation, the relationship with its corresponding entities is aggregated into a higher-level entity.
  Example:- phone numbers on your mobile phone. You can refer to them individually – your mother's number, your best friend's number, etc. But it's easier to think of them collectively, as your phone number list. It is also important to realize that each member of the aggregation still has the properties of the whole. In other words, each phone number in the list remains a phone number. The process of combining them has not altered them in any way.



****

<u>Conceptual Design with the ER Model:</u>

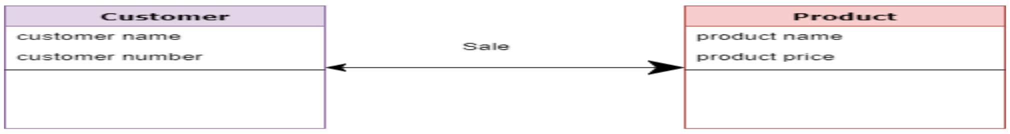
A **Conceptual Data Model** is an organized view of database concepts and their relationships. The purpose of creating a conceptual data model is to establish entities, their attributes, and relationships. In this data modeling level, there is hardly any detail available on the actual database structure. Business stakeholders and data architects typically create a conceptual data model.

The 3 basic tenants of Conceptual Data Model are

- **Entity**: A real-world thing
- **Attribute**: Characteristics or properties of an entity
- **Relationship**: Dependency or association between two entities

Data model example:

- Customer and Product are two entities. Customer number and name are attributes of the Customer entity
- Product name and price are attributes of product entity
- Sale is the relationship between the customer and product



Characteristics of a conceptual data model

- Offers Organization-wide coverage of the business concepts.
- This type of Data Models are designed and developed for a business audience.
- The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and technology. The focus is to represent data as a user will see it in the "real world."

****

Conceptual Design for Large Enterprises :

Follow the steps given below to draw an Entity Relationship (ER) diagram for a University database application −

Step 1 − Identifying the entity sets.

The entity set has multiple instances in a given business scenario.

As per the given constraints the entity sets are as follows −

- Department
- Course
- Student
- Instructor

Head of the Department (HOD) is not an entity set. It is a relationship between the instructor and department entities.

Step 2 − Identifying the attributes for the given entities

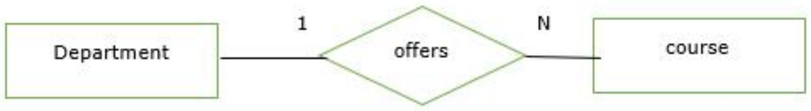
- Department − the relevant attributes are department Name and location.
- Course − The relevant attributes are courseNo, course Name, Duration, and prerequisite.
- Instructor − The relevant attributes are Instructor Name, Room No, and telephone number.
- Student − The relevant attributes are Student No, Student Name, and date of birth.

Step 3 − Identifying the Key attributes

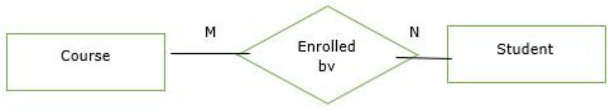
- Department Name is the key attribute for Department.
- CourseNo is the key attribute for Course entity.
- Instructor Name is the key attribute for the Instructor entity.
- StudentNo is the key attribute for Student entities.

Step 4 − Identifying the relationship between entity sets

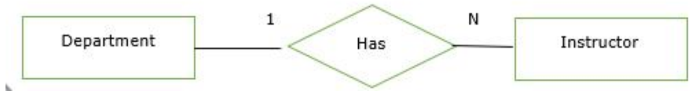
- The department offers multiple courses and each course belongs to only one department, hence cardinality between department and course if one to many.



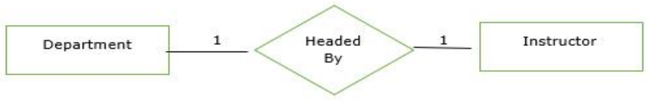
- One course is enrolled by multiple students and one student for multiple courses. Hence, relationships are many to many.



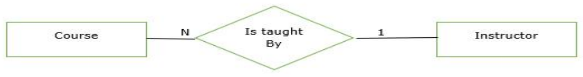
- One department has multiple instructors and one instructor belongs to one and only one department, hence the relationship is one to many.



- Each department has one “HOD” and one instructor is “HOD” for only one department, hence the relationship is one to one. Here, HOD refers to the head of the department.



- 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.

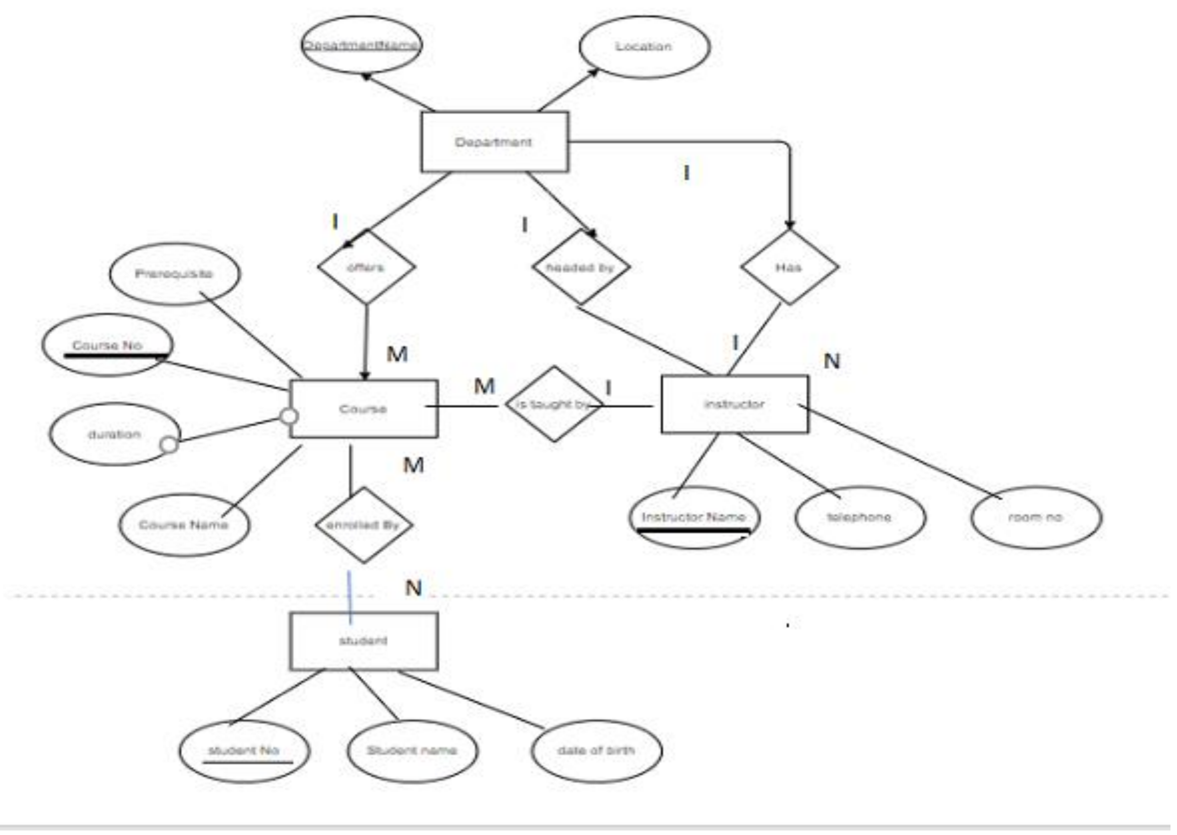


The relationship between instructor and student is not defined because of the following reasons −

- There is no significance in the relationship.
- We can always derive this relationship indirectly through course and instructors, and course and student.

Step 5 − Complete ER model

The complete ER Model is as follows −



****

<u>**Introduction to the Relational Model**</u>:

In the concept of a relational database management system, data is organized into tables. Tables are similar to folders in a file system, where each table stores a collection of information. Tables are further divided into columns and rows. Columns represent the attributes of an entity, while rows represent the entities themselves.

Relational Model Concepts:

A relational database is based on the relational model. This database consists of various components based on the relational model. These include:

- **Relation :** Two-dimensional table used to store a collection of data elements.
- **Tuple :** Row of the relation, depicting a real-world entity.
- **Attribute/Field :** Column of the relation, depicting properties that define the relation.
- **Attribute Domain :** Set of pre-defined atomic values that an attribute can take i.e., it describes the legal values that an attribute can take.
- **Degree :** It is the total number of attributes present in the relation.
- **Cardinality:** It specifies the number of entities involved in the relation i.e., it is the total number of rows present in the relation. Read more about Cardinality in DBMS.
- **Relational Schema :** It is the logical blueprint of the relation i.e., it describes the design and the structure of the relation. It contains the table name, its attributes, and their types:

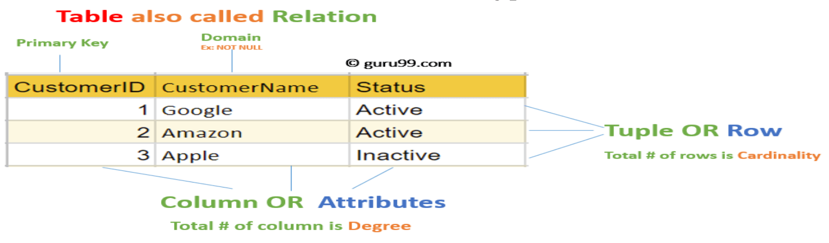


Table also called Relation
Primary Key
Domain
Ex: NOT NULL
© guru99.com
Tuple OR Row
Total # of rows is Cardinality
Column OR Attributes
Total # of column is Degree

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

****

Integrity Constraints over Relations:

In Database Management Systems, integrity constraints are pre-defined set of rules that are applied on the table fields(columns) or relations to ensure that the overall validity, integrity, and consistency of the data present in the database table is maintained. Evaluation of all the conditions or rules mentioned in the integrity constraint is done every time a table insert, update, delete, or alter operation is performed.

Types of Integrity Constraints:

There are four types of integrity constraints in DBMS:

1. Domain Constraint
2. Entity Constraint
3. Referential Integrity Constraint
4. Key Constraint

❖ **Domain Constraint:**

Domain integrity constraint contains a certain set of rules or conditions to restrict the kind of attributes or values a column can hold in the database table. The data type of a domain can be string, integer, character, DateTime, currency, etc.

Example:

Consider a Student's table having Roll No, Name, Age, Class of students.

Roll No	**Name**	**Age**	**Class**
101	rekha	14	6
102	geetha	16	8
103	Durga	8	4
104	kumar	18	12
105	naidu	6	A

In the above student's table, the value A in the last row last column violates the domain integrity constraint because the Class attribute contains only integer values while A is a character.

❖ **Entity Integrity Constraint:**

Entity Integrity Constraint is used to ensure that the primary key cannot be null. A primary key is used to identify individual records in a table and if the primary key has a null value, then we can't identify those records. There can be null values anywhere in the table except the primary key column.

Example: Consider Employees table having Id, Name, and salary of employees

ID	**Name**	**Salary**
101	rekha	40000
102	geetha	60000
103	Durga	80000
104	kumar	1800000
105	naidu	36000

In the above employee's table, we can see that the ID column is the primary key and contains a null value in the last row which violates the entity integrity constraint.

❖ **Referential Integrity Constraint**

Referential Integrity Constraint ensures that there must always exist a valid relationship between two relational database tables. This valid relationship between the two tables confirms that a foreign key exists in a table. It should always reference a corresponding value or attribute in the other table or be null.

Example: Consider an Employee and a Department table where Dept_ID acts as a foreign key between the two tables

Employees Table:

ID	**Name**	**Salary**	**Dept_ID**
101	rekha	40000	3
102	geetha	60000	2
103	Durga	80000	4
104	kumar	1800000	3
105	naidu	36000	**1**

Dept_ID	**Dept_Name**

Department Table:

1	Sales
2	HR
3	Technical

- **Key constraint:**

Keys are the set of entities that are used to identify an entity within its entity set uniquely. There could be multiple keys in a single entity set, but out of these multiple keys, only one key will be the primary key. A primary key can only contain unique and not null values in the relational database table.

Example:

Roll No	Name	Age	Class
101	rekha	14	6
102	geetha	16	8
103	Durga	8	4
104	kumar	18	12
102	naidu	6	7

The last row of the student's table violates the key integrity constraint since Roll No 102 is repeated twice in the primary key column. A primary key must be unique and not null therefore duplicate values are not allowed in the Roll No column of the above student's table.

Types of key constraints are:

- **Primary Key:**

The PRIMARY KEY constraint uniquely identifies each record in a table. Primary keys must contain UNIQUE values, and cannot contain NULL values.

Example:

```
create table emp
 ( id   number
 name varchar (20)
 age  number
 course varchar(10)
 primary key (id) );
```

- **Foreign Key:**

The foreign key a constraint is a column or list of columns that points to the primary key column of another table The main purpose of the foreign key is only those values are allowed in the present table that will match the primary key column of another table.

Example: create a foreign key

1. **Reference table/primary table:**
```
create table customers1(
  id   int ,
 name varchar (20) ,
 course varchar(10) ,
 primary key (id));
```
2. **Child table:**
```
create table customers2(
  id   int ,
marks int,
references customers1(i));
```

- **Not Null :**

Null represents a record where data may be missing data or data for that record may be optional Once **not null is applied to a particular column, you cannot enter null values to that column** and restricted to maintain only some proper value other than null

Example:

```
create table orders (
orderid number not null,
personname varchar(10));
```

- **Unique :**

  Sometimes we need to maintain only unique data in the column of a database table, this is possible by using a unique constraint Unique constraint ensures that all values in a column are unique.

  Example:

  ```
  create table persons (
   id number unique,
  lastname varchar(255),
  firstname varchar(255),
  age number );
  ```

- **Default:**

  Default clause in SQL is used to add default data to the columns. When a column is specified as default with some value then all the rows will use the same value i.e each and every time while entering the data we need not enter that value.

  Example:

  ```
   create table emp (
   id number not null,
   lastname varchar(255) not null,
   firstname varchar(255),
   age number,
   city varchar(255) default 'hyderabad');
  ```

- **Check:**

  Check constraint ensures that the data entered by the user for that column is within the range of values or possible values specified.

  Example:

  ```
     create table student (
      id int ,
      name varchar(255) ,
      age int,
      check (age>=18));
  ```

- **Candidate Key:**

  A candidate key is a set of attributes (or attributes) that uniquely identify the tuples in relation to or table. As we know the Primary key is a minimal super key.

- **Super Key**:

  The role of the super key is simply to identify the tuples of the specified table in the database. It is the superset where the candidate key is a part of the super key only.

****

<u>Enforcing Integrity Constraints:</u>

Data integrity refers to the correctness and completeness of data within a database. To enforce data integrity, you can constrain or restrict the data values that users can insert, delete, or update in the database

- Transact-SQL provides several mechanisms for integrity enforcement in a database such as rules, defaults, indexes, and triggers. These mechanisms allow you to maintain these types of data integrity:
- Requirement – requires that a table column must contain a valid value in every row; it cannot allow null values. The create table statement allows you to restrict null values for a column.
- Check or validity – limits or restricts the data values inserted into a table column. You can use triggers or rules to enforce this type of integrity.
- Uniqueness – no two table rows can have the same non-null values for one or more table columns. You can use indexes to enforce this integrity.
- Referential – data inserted into a table column must already have matching data in another table column or another column in the same table. A single table can have up to 192 references.

****

<u>Querying Relational Data:</u>

- ❖ A relational database query (query, for short) is a question about the data, and the answer consists of a new relation containing the result. For example ,we might want to find all students younger than 18 or all students enrolled inReggae203. A query language is a specialized language for writing queries.SQL is the most popular commercial query language for a relational DBMS.
- ❖ We can retrieve rows corresponding to students who are younger than 18 with the following SQL query:

```
SELECT *
FROM Students S
WHERE S.age < 18
```

Here,..The symbol ,*, means that we retain all fields of selected tuples in the result.

- ❖ In addition to selecting a subset of tuples, a query can extract a subset of the fields of each selected tuple. vVe can compute the names and logins of students who are younger than 18 with the following query:

```
SELECT S.name, S.login
FROM Students S
WHERE S.age < 18
```

- ❖ We can also combine information in the Students and Enrolled relations. If we want to obtain the names of all students who obtained an A and the id of the course in which they got an A, we could write the following query:

```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid = E.studid AND E.grade = 'A'
```

****

Logical Database Design: ER to Relational:

The ER model is convenient for representing an initial, high-level database design. Given an ER diagram describing a database.

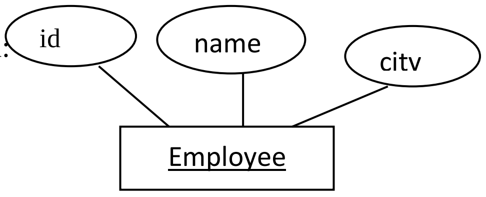
- ❖ **Entity Sets to Tables:**

An entity set is mapped to a relation in a straightforward way: Each attribute of the entity set becomes an attribute of the table. Note that we know both the domain of each attribute and the (primary) key of an entity set.

Consider the Employees entity set with attributes id, name, and city shown below.

Example:

Entity relational model:



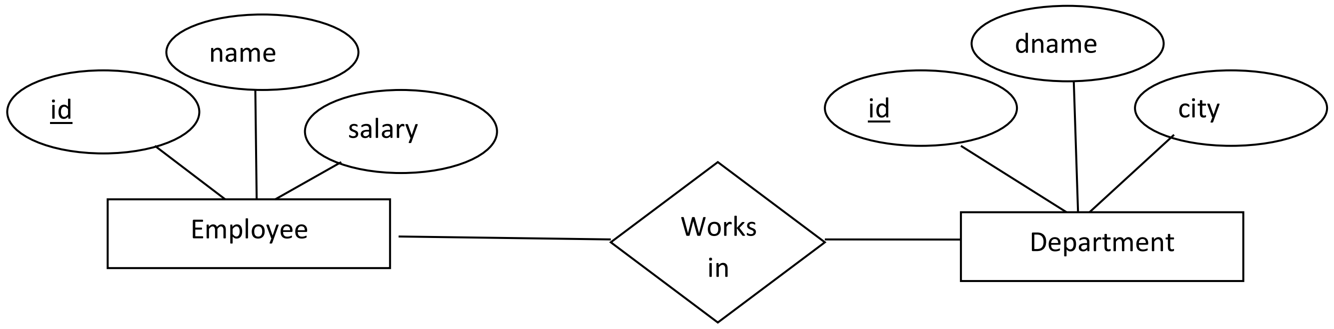
Example: Relation model:

```
Create table Employee
(id number,
name varchar2(10),
city varchar2(10));
```

id	name	city

Relationship Sets (without Constraints) to Tables;

A relationship set, like an entity set, is mapped to a relation in the relational model. we begin by considering relationship sets without key and participation constraints,



Example: Relation model:

Create table Employee
(id number,
name varchar2(10),
salary number(10)
primary key(id));

id	name	salary

Create table Department
(id number references on employee
name varchar2(10),
city number(10)
primary key(id));

id	dname	city

****

<u>Introduction to Views</u>:

1. **<u>Views:</u>**

Views in SQL are considered as a virtual table. A view also contains rows and columns. To create the view, we can select the fields from one or more tables present in the database. View is logical copy of table data.

Advantages of views:

- Provide Security.
- Increase performance.
- Update dynamically

Types of view:

- Simple view
- Composite view
- Inline view
- Forced view
- Materialized view

- **Simple view:** A view is created on single table is called simple view
  Example: create view empp
  as
  select id, name, salary from emp;
- **Composite view**: A view is created on multiple tables are called composite view.
  Example: create view empdept
  as
  select id, name, salary, dname, location from emp, dept where emp.did=dept.did;
- **Inline view:** A view based on a subquery in FROM Clause, that subquery creates a temporary table and simplifies the complex query.
  Example: select id, name,salary from (select * from emp)t;
- **Forced view**: The Force view is used to create a view when there is no table that exists in the database
  Example: create or replace force view view_name
  as
  select id,name from product;

- **Materialized view:** A materialized view in Oracle is a database object that contains the results of a query. They are local copies of data located remotely, or are used to create summary tables based on aggregations of a table's data. Materialized views, which store data based on remote tables are also, know as snapshots.

  Example:

  ```
  create materialized view new
   as
  select *from empp;
  ```

****

<u>Destroying/Altering Tables and Views</u>:

- **Dropping Views:**

  Obviously, where you have a view, you need a way to drop the view if it is no longer needed. The syntax is very simple and is given below −

  Example: DROP VIEW view_name;

- **Alter View:**
  - ALTER TABLE modifies the structure of an existing table. To add a column called maiden-name to Students, for example, we would use the following command:
    Example: alter table students
    ```
    add column maiden-name char(10)
    ```
  - ALTER TABLE can also be used to delete columns and add or drop integrity constraints on a table; we do notd iscuss these aspects of the command beyond remarking that dropping column sis treated very similarly to dropping tables or views.
  - ALTER VIEW command modifies views created using the create view command or a view projected from persistent class. The altered view replaces the existing view ,so you cannot modify specific column in a view

    Example:
    ```
    Create or replace view [brazil customers] as
    select customername, contactname, city
    from customers
    where country = "brazil";
    ```

Example:

```
Alter view newemployees as
select name,office,startdate
from sample.employees
where name='rekha';
```

****