

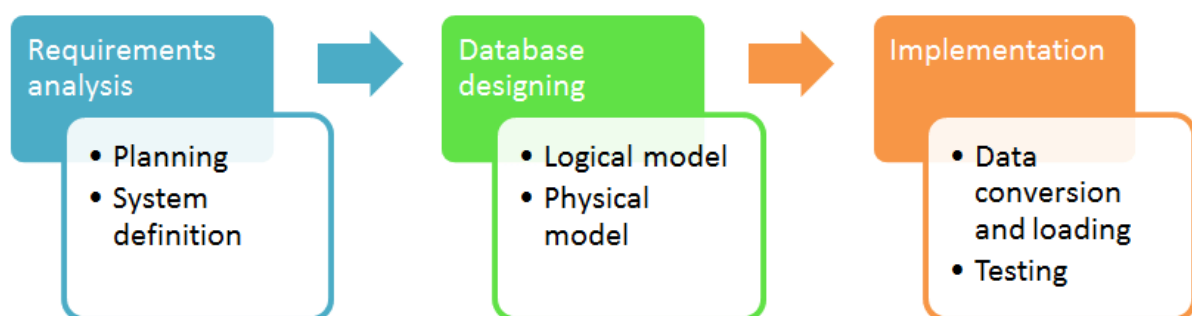
## Database Design in DBMS:

- It is a collection of processes that facilitate the designing, development, and implementation.
- Properly designed database are easy to maintain, and are cost effective in terms of disk storage space.
- The database designer decides how the data elements correlate and what data must be stored.
- Objective: to produce logical and physical designs models of the proposed database system.
- The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.
- The physical data design model involves translating the logical DB design of the database into physical media using hardware resources and software systems such as database management systems (DBMS).

## Importance of Database Design:

- It helps to produce database systems:
  1. That meet the requirements of the users
  2. Have high performance.

## Database development life cycle:



### 1. Requirements analysis

- Planning – This stage of database design concepts are concerned with planning of entire Database Development Life Cycle. It takes into consideration the Information Systems strategy of the organization.

- System definition – This stage defines the scope and boundaries of the proposed database system.

## **2. Database designing:**

- 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.

## **3. Implementation:**

- Data conversion and loading – this stage of relational databases design is concerned with importing and converting data from the old system into the new database.
- Testing – this stage is concerned with the identification of errors in the newly implemented system. It checks the database against requirement specifications.

## **Two Types of Database Design Techniques:**

1. Normalization
2. ER Modelling

### **Normalization:**

- It reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization rules divides larger tables into smaller tables and links them using relationships.
- The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

### **KEY in SQL:**

- It is a value used to identify records in a table uniquely.
- It is a single column or combination of multiple columns used to uniquely identify rows in the table.
- SQL Key is used to identify duplicate information, and it also helps establish a relationship between multiple tables in the database.
- Columns in a table that are NOT used to identify a record uniquely are called non-key columns.

### **Primary Key:**

- It is a single column value used to identify a database record uniquely.
- It has following attributes:
  1. It cannot be NULL
  2. Its value must be unique
  3. Its values should rarely be changed
- A composite key is a primary key composed of multiple columns used to identify a record uniquely

### Database Normal Forms:

List of Normal Forms in SQL:

- 1NF (First Normal Form)
- 2NF (Second Normal Form)
- 3NF (Third Normal Form)
- BCNF (Boyce-Codd Normal Form)
- 4NF (Fourth Normal Form)
- 5NF (Fifth Normal Form)
- 6NF (Sixth Normal Form)

### Database Normalization with Examples:

- Assume, a video library maintains a database of movies rented out. Without any normalization in database, all information is stored in one table as shown below:

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3 <sup>rd</sup> Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5 <sup>th</sup> Avenue	Clash of the Titans	Mr.

### 1NF (First Normal Form):

- **Rules:**

1. Each table cell should contain a single value.
  2. Each record needs to be unique.
- Consider previous table in 1NF:

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3 <sup>rd</sup> Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3 <sup>rd</sup> Street 34	Daddy's Little Girls	Mr.
Robert Phil	5 <sup>th</sup> Avenue	Clash of the Titans	Mr.

## 2NF (Second Normal Form):

- **Rules:**

1. Be in 1NF
  2. Single Column Primary Key that does not functionally dependent on any subset of candidate key relation
- It is clear that we can't move forward to make our simple database in 2<sup>nd</sup> Normalization form unless we partition the table above.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
<b>1</b>	Janet Jones	First Street Plot No 4	Ms.
<b>2</b>	Robert Phil	3 <sup>rd</sup> Street 34	Mr.
<b>3</b>	Robert Phil	5 <sup>th</sup> Avenue	Mr.

MEMBERSHIP ID	MOVIES RENTED
<b>1</b>	Pirates of the Caribbean
<b>1</b>	Clash of the Titans
<b>2</b>	Forgetting Sarah Marshal
<b>2</b>	Daddy's Little Girls
<b>3</b>	Clash of the Titans

- In Table 2, Membership ID is the Foreign Key.

## Database – Foreign Key:

- A foreign key can have a different name from its primary key.
- It ensures rows in one table have corresponding rows in another.
- Unlike the Primary key, they do not have to be unique. Most often they aren't.
- Foreign keys can be null even though primary keys cannot.

### Transitive functional dependencies:

- It is when changing a non-key column, might cause any of the other non-key columns to change
- Consider the table 1, changing the non-key column Full Name may change Salutation.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 <sup>rd</sup> Street 34	Mr.
3	Robert Phil	5 <sup>th</sup> Avenue	Mr.

*Change in Name* (circled around 'Robert Phil' in row 3) → *May Change Salutation* (arrow pointing to 'Mr.' in row 3)

### 3NF (Third Normal Form):

- Rules:
  1. Be in 2NF
  2. Has no transitive functional dependencies
- To move our 2NF table into 3NF, we again need to again divide our table.
- Below is a 3NF example in SQL database:

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION ID
1	Janet Jones	First Street Plot No 4	2
2	Robert Phil	3 <sup>rd</sup> Street 34	1
3	Robert Phil	5 <sup>th</sup> Avenue	1

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

SALUTATION ID	SALUTATION
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

### BCNF (Boyce-Codd Normal Form):

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency  $X \rightarrow Y$ , X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Example: Let's assume there is a company where employees work in more than one department.

EMPLOYEE table:

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

In the above table Functional dependencies are as follows:

1.  $EMP\_ID \rightarrow EMP\_COUNTRY$
2.  $EMP\_DEPT \rightarrow \{DEPT\_TYPE, EMP\_DEPT\_NO\}$

Candidate key: {EMP-ID, EMP-DEPT}

- The table is not in BCNF because neither EMP\_DEPT nor EMP\_ID alone are keys.
- To convert the given table into BCNF, we decompose it into three tables:

**EMP\_COUNTRY table:**

EMP_ID	EMP_COUNTRY
--------	-------------

264	India
264	India

EMP\_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP\_DEPT\_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

- Functional dependencies:

- EMP\_ID  $\rightarrow$  EMP\_COUNTRY
- EMP\_DEPT  $\rightarrow$  {DEPT\_TYPE, EMP\_DEPT\_NO}

- Candidate keys:

For the first table: EMP\_ID

For the second table: EMP\_DEPT

For the third table: {EMP\_ID, EMP\_DEPT}

- Now, this is in BCNF because left side part of both the functional dependencies is a key.

#### 4NF (Fourth Normal Form) Rules:

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency  $A \twoheadrightarrow B$ , if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.
- **Example:**

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

Table: Student

- The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.
- In the STUDENT relation, a student with STU\_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU\_ID, which leads to unnecessary repetition of data.
- So to make the above table into 4NF, we can decompose it into two tables:

#### STUDENT\_COURSE:

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

#### STUDENT\_HOBBY:



STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

### 5NF (Fifth Normal Form) Rules:

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).
- Example:

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

- In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.
- Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

- So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1:

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2:

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3:

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

## **6NF (Sixth Normal Form) Proposed**

- 6<sup>th</sup> Normal Form is not standardized, yet however, it is being discussed by database experts for some time. Hopefully, we would have a clear & standardized definition for 6<sup>th</sup> Normal Form in the near future...

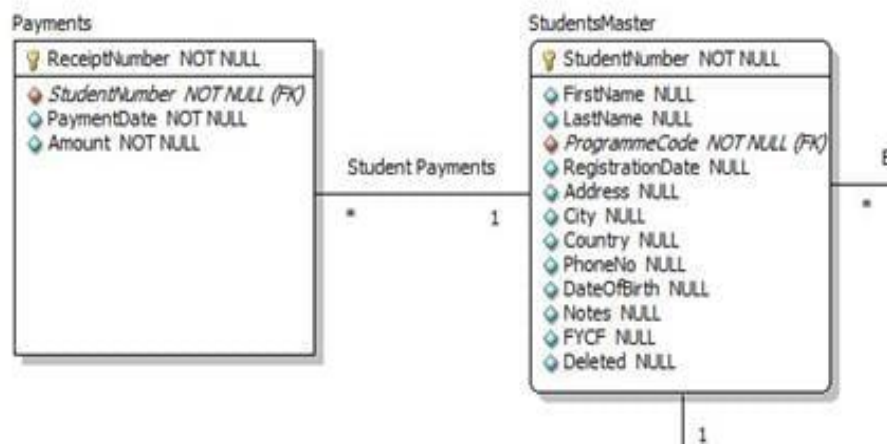
## **Entity Relationship Model (ER Modelling):**

- It is a graphical approach to database design.
- It is a high-level data model that defines data elements and their relationship for a specified software system.
- It is used to represent real-world objects.
- An **Entity** is a thing or object in real world that is distinguishable from surrounding environment.
- For example, each employee of an organization is a separate entity.
- Following are some of major characteristics of entities.
  1. An entity has a set of properties.
  2. Entity properties can have values.
- E.g.: An employee of an organization is an entity. If “Peter” is a programmer (an employee) at Microsoft, he can have attributes (properties) like name, age, weight, height, etc. It is obvious that those do hold values relevant to him.
- Each attribute can have Values. In most cases single attribute have one value. But it is possible for attributes have multiple values also.
- For example Peter’s age has a single value. But his “phone numbers” property can have multiple values.
- Entities can have relationships with each other.
- Assume that each Microsoft Programmer is given a Computer. It is clear that that Peter’s Computer is also an entity. Peter is using that computer, and the same computer is used by Peter. In other words, there is a mutual relationship between Peter and his computer.
- In Entity Relationship Modelling, we model entities, their attributes and relationships among entities.

## **Enhanced Entity Relationship (EER) Model:**

- It is a high-level data model which provides extensions to original Entity Relationship (ER) model.
- It supports more details design.

- EER Modelling emerged as a solution for modelling highly complex databases.
- EER uses UML notation.
- UML is the acronym for Unified Modelling Language, it is a general-purpose modelling language used when designing object-oriented systems.
- Entities are represented as class diagrams.
- Relationships are represented as associations between entities.
- The diagram shown below illustrates an ER diagram using the UML notation.



### Importance of ER Model:

- One of the challenges faced when designing a database is the fact that designers, developers, and end-users tend to view data and its usage differently. If this situation is left unchecked, we can end up producing a database system that does not meet the requirements of the users.
  - Communication tools understood by all stake holders (technical as well as non-technical users) are critical in producing database systems that meet the requirements of the users. ER models are examples of such tools.
  - ER diagrams also increase user productivity as they can be easily translated into relational tables.
- Case Study: ER diagram for “MyFlix” Video Library:**
- Let’s now work with the MyFlix Video Library database system to help understand the concept of ER diagrams.
  - MyFlix is a business entity that rents out movies to its members. MyFlix has been storing its records manually. The management now wants to move to a DBMS

- Let's look at the steps to develop EER diagram for this database:
1. Identify the entities and determine the relationships that exist among them.
  2. Each entity, attribute, and relationship, should have appropriate names that can be easily understood by the non-technical people as well.
  3. Relationships should not be connected directly to each other. Relationships should connect entities.
  4. Each attribute in a given entity should have a unique name.

- **Entities in the “MyFlix” library:**

- The entities to be included in our ER diagram are;
1. Members – this entity will hold member information.
  2. Movies – this entity will hold information regarding movies
  3. Categories – this entity will hold information that places movies into different categories such as “Drama”, “Action”, and “Epic” etc.
  4. Movie Rentals – this entity will hold information that about movies rented out to members.
  5. Payments – this entity will hold information about the payments made by members.

- **Defining the Relationships Among Entities:**

1. Members and movies

- The following holds true regarding the interactions between the two entities.
- A member can rent more than one movie in a given period.
- A movie can be rented by more than one member in a given period.
- From the above scenario, we can see that the nature of the relationship is many-to-many. Relational databases do not support many-to-many relationships. We need to introduce a junction entity. This is the role that the MovieRentals entity plays. It has a one-to-many relationship with the members table and another one-to-many relationship with movies table.

2. Movies and categories entities:

- The following holds true about movies and categories.
- A movie can only belong to one category but a category can have more than one movie.
- We can deduce from this that the nature of the relation between categories and movies table is one-to-many.

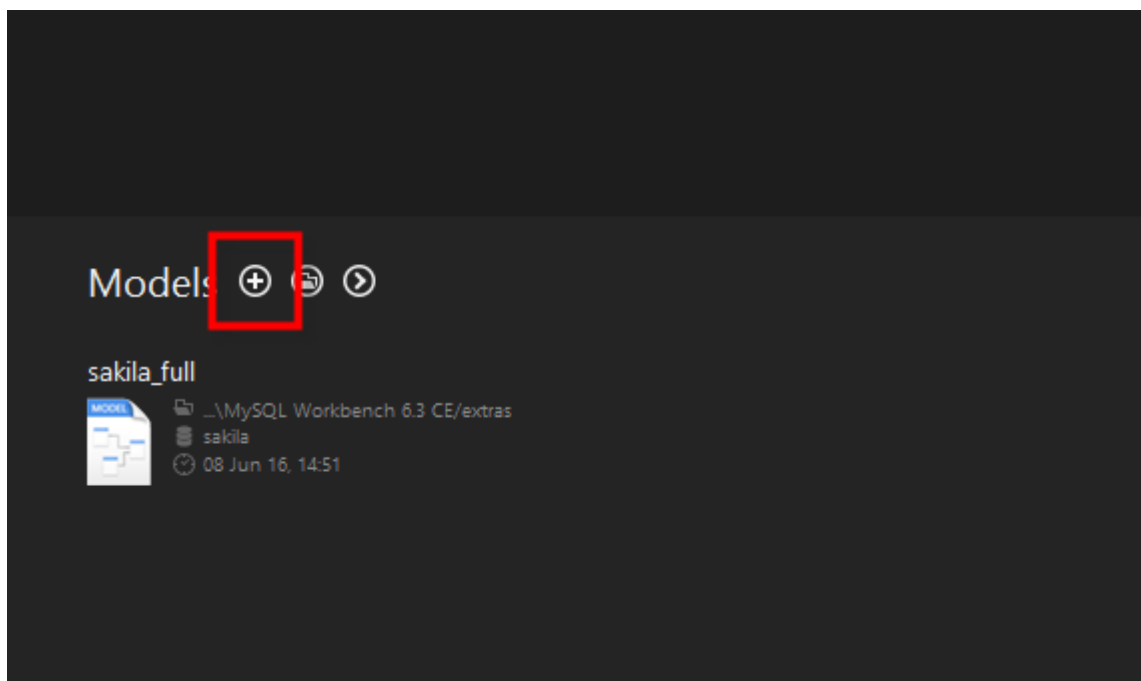
3. Members and payments entities:

- The following holds true about members and payments

- A member can only have one account but can make a number of payments.
- We can deduce from this that the nature of the relationship between members and payments entities is one-to-many.

Now let's create EER model using MySQL Workbench

In the MySQL work bench, Click – “+” Button



Double click on Add Diagram button to open the workspace for ER diagrams.

MySQL Workbench interface showing the "MySQL Model" tab.

The interface includes a menu bar (File, Edit, View, Arrange, Model, Database, Tools, Scripting, Help) and a toolbar with icons for various database operations.

The left sidebar contains the "Description Editor" (No Selection) and the "User Types List" table.

The right sidebar shows the "Model Overview" section, which is highlighted with a red box. It contains an "Add Diagram" button (labeled PHYSICAL) and a "Physical Schemas" section.

The "Physical Schemas" section lists the "mydb" MySQL Schema. Below this, there are sections for Tables (0 items), Views (0 items), Routines (0 items), and Routine Groups (0 items), each with an "Add" button.

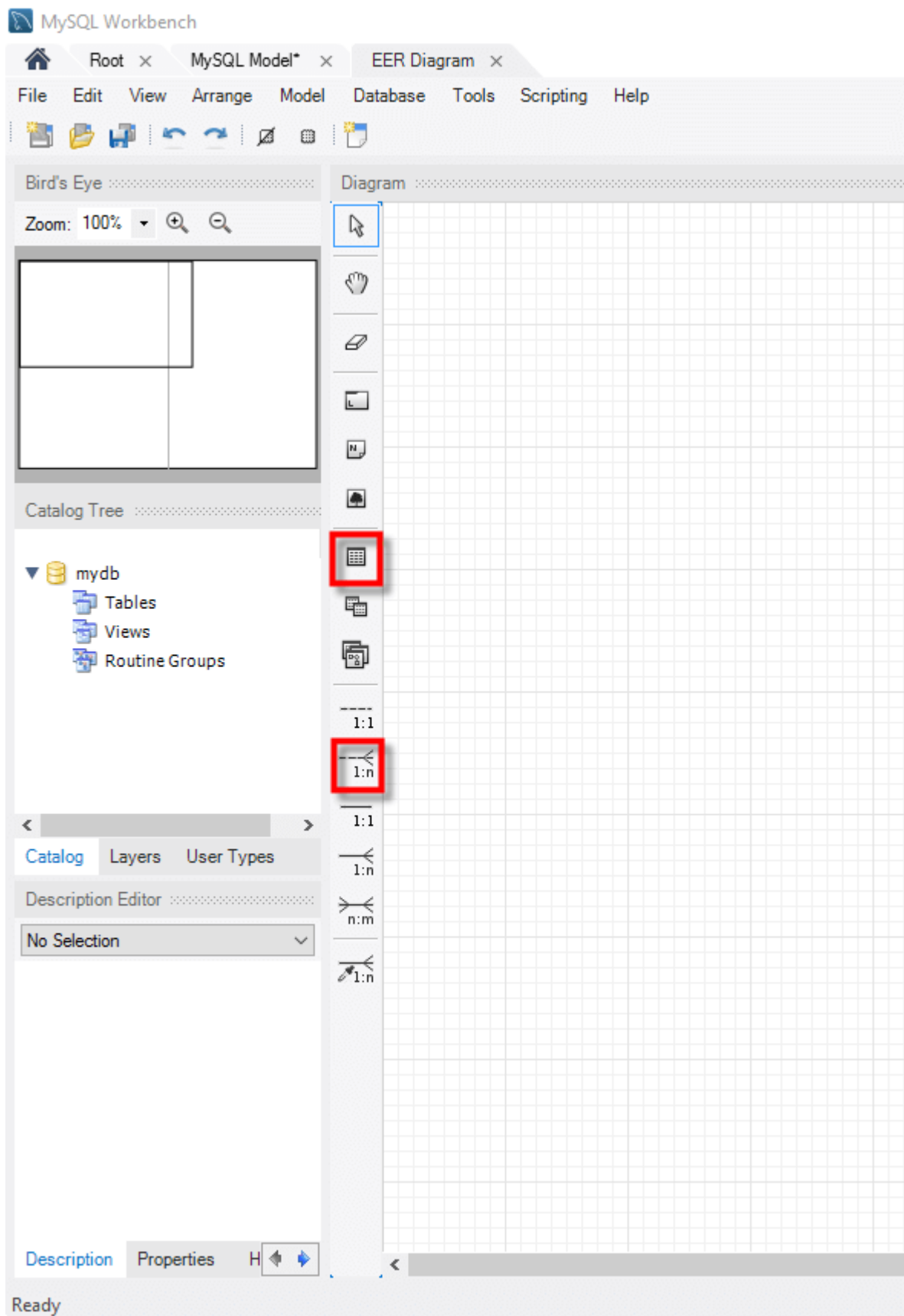
At the bottom of the right sidebar, there are expandable sections for "Schema Privileges", "SQL Scripts", and "Model Notes".

The "User Types List" table is visible in the bottom left:

Type	Definition
BOOL	TINYINT(1)
BOOLEAN	TINYINT(1)
FIXED	DECIMAL(10...
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT(4)
INT2	SMALLINT(6)
INT3	MEDIUMINT...
INT4	INT(11)
INT8	BIGINT(20)
INTEGER	INT(11)
LONG VAR...	MEDIUMBLOB
LONG VAR...	MEDIUMTEXT
LONG	MEDIUMTEXT
MIDDLEINT	MEDIUMINT...
NUMERIC	DECIMAL(10...
DEC	DECIMAL(10...
CHARACTER	CHAR(1)


The status bar at the bottom indicates "Ready".

Following window appears:





Let's look at the two objects that we will work with.

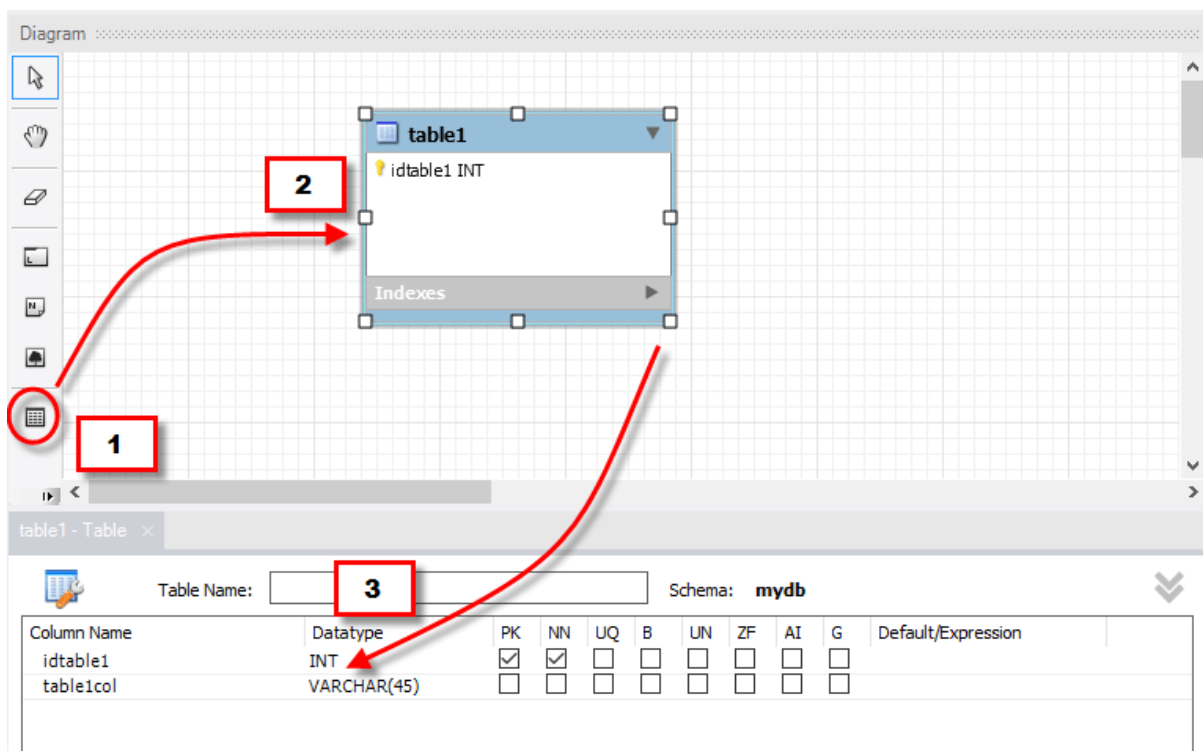
- The table object allows us to create entities and define the attributes associated with the particular entity.
-  The place relationship button allows us to define relationships between entities.

The members' entity will have the following attributes:

- Membership number
- Full names
- Gender
- Date of birth
- Physical address
- Postal address

Let's now create the members table

1. Drag the table object from the tools panel
2. Drop it in the workspace area. An entity named table 1 appears
3. Double click on it. The properties window shown below appears



Next ,

1. Change table 1 to Members
2. Edit the default idtable1 to membership\_number

3. Click on the next line to add the next field
4. Do the same for all the attributes identified in members' entity.

Your properties window should now look like this.

Members - Table

Table Name:  Schema:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AT	G	Default/Expression
membership_number	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
full_names	VARCHAR(150)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
gender	VARCHAR(6)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
date_of_birth	DATE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
physical_address	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
postal_address	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
contact_number	VARCHAR(75)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
email	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name:

Collation:

Comments:

Data Type:

Expression:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☐ Unique

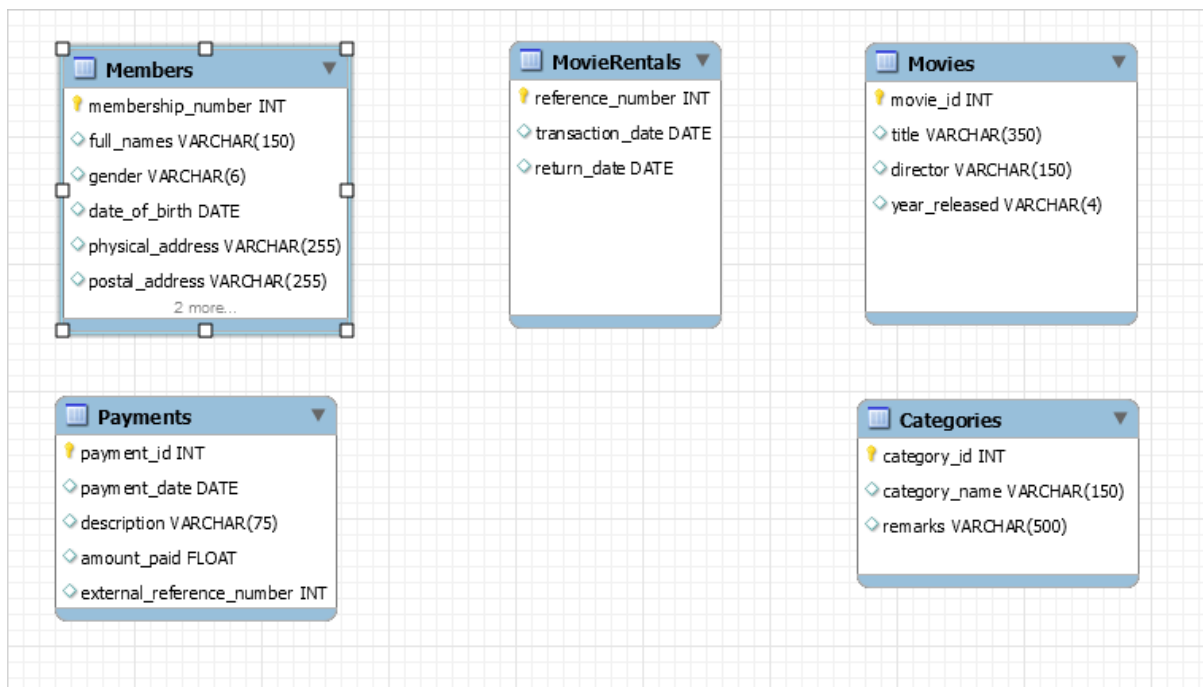
☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☒ Generated

Columns | Indexes | Foreign Keys | Triggers | Partitioning | Options | Inserts | Privileges

Repeat the above steps for all the identified entities.

Your diagram workspace should now look like the one shown below.



Let's create relationship between Members and Movie Rentals

Select the place relationship using existing columns too

Click on membership\_number in the Members table

Click on reference\_number in the MovieRentals table