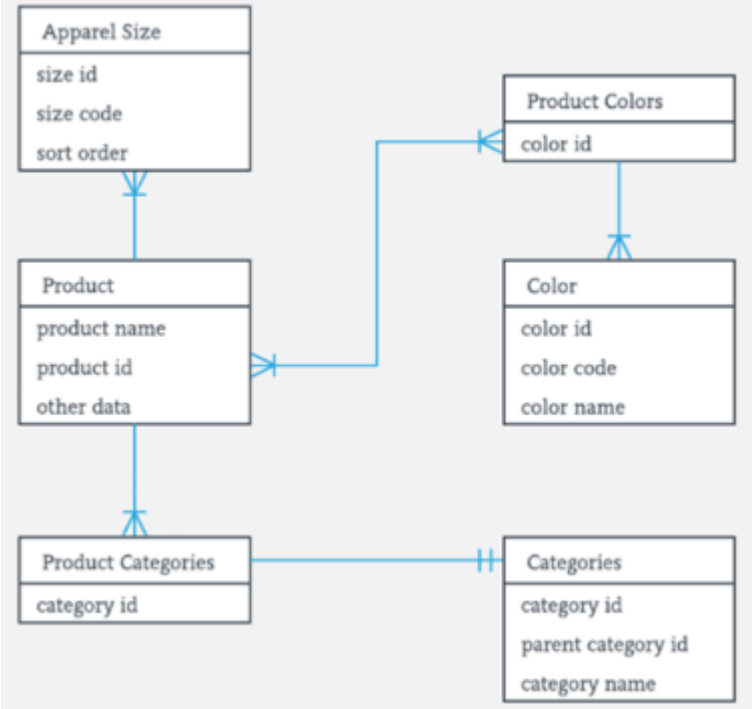
ER Diagram

What is ER diagram?

- ER Diagram stands for Entity Relationship Diagram, that displays the relationship of entity sets stored in a database.
- In other words, ERD help to explain the logical structure of databases.
- ERD are created based on `three basic concepts`:
  - `ENTITIES`
  - `ATTRIBUTES`
  - `RELATIONSHIPS`

- ERD contain different sysmbols that use
  - `Rectangle` to represent entities
  - `Oval` to define attributes
  - `Diamond` shapes to represent relationships.



Here all are entities as they are in rectangle shape. Apparel Size, Product colors etc.

What is ER Model?

- **ER Model** stands for Entity Relationship Model is a high-level conceptual data model diagram. ER model helps to systematically analyze data requirements to produce a well-designed database.
- The ER Model represents real-world entities and the relationships between them. Creating an ER Model in DBMS is considered as a best practice before implementing your database.
- ER Modeling helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.

Why use ER Diagrams?

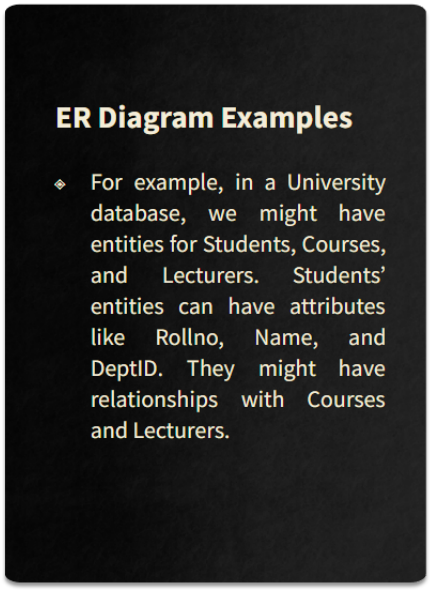
- Helps you to define terms related to entity relationship modeling.

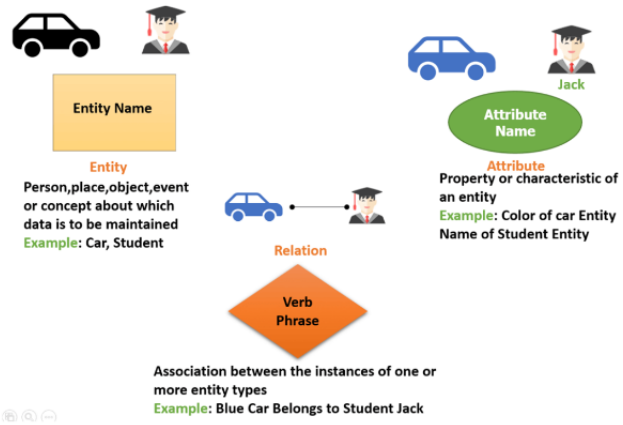
- Provide a preview of how all your tables should connect, and what fields are going to be on each table.
- Helps to describe entities, attributes, and relationships.
- ER diagrams are translatable into relational tables which allows you to build databases quickly.
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications.
- The database designer gains a better understanding of the information to be contained in the database with the help of an ERP diagram.
- ERD Diagram allows you to communicate the logical structure of the database to users.

Facts about ER Diagram Model

- ER model allows you to draw Database Design.
- It is an easy-to-use graphical tool for modeling data.
- Widely used in Database Design.
- It is a GUI representation of the logical structure of a Database.
- It helps you to identify the entities which exist in a system and the relationships between those entities.

Example





Entity

- A real-world thing either living or non-living that is easily recognizable and nonrecognizable. It is anything in the enterprise that is to be represented in our database.
- It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.
- An entity can be a place, person, object, event or concept, that stores data in the database

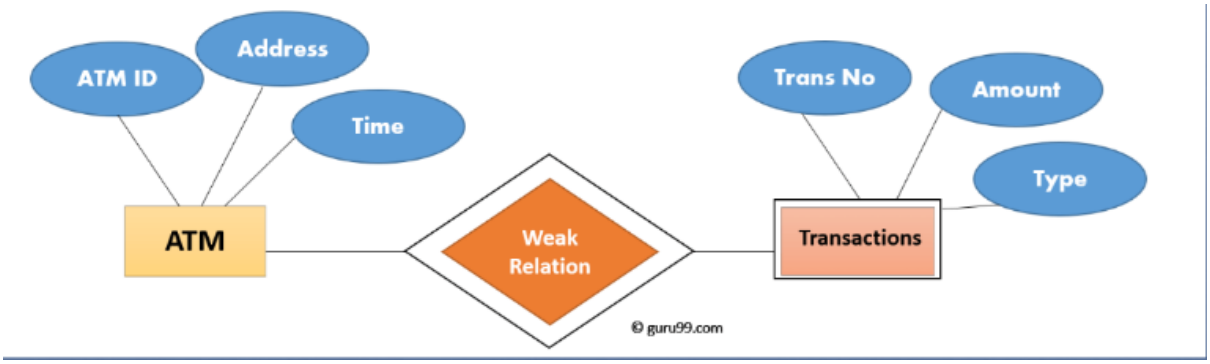
Relationship

- Relationship is nothing but an association among two or more entities. E.g., Tom works in the Chemistry department.

Weak Entites

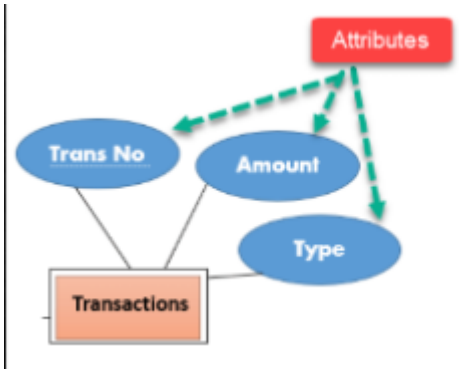
A weak entity is an entity that depends on another entity, known as its parent entity, for its existence. A weak entity cannot be uniquely identified without considering the primary key of its parent entity as part of its own primary key. In other words, a weak entity's primary key is formed by a combination of its attributes and the primary key attributes of its parent entity.

weak entities are essential in database modeling to represent entities with a strong dependency on their parent entities and to ensure proper data organization and integrity in the database design.



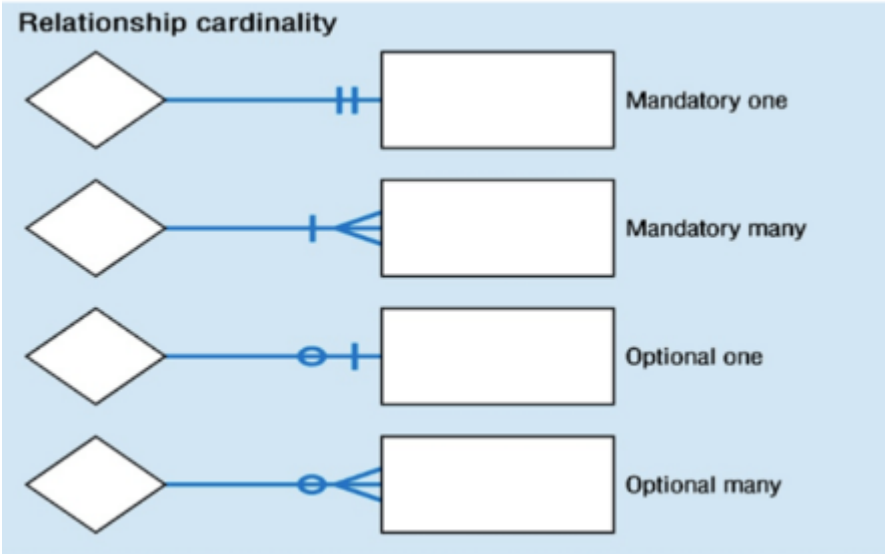
Attributes

- It is a single-valued property of either an entity type or a relationship type.
- For example, a lecture might have attributes: time, date, duration, place, etc.
- An attribute in ER Diagram examples, is represented by an Ellipse.



CARDINALITY

- Defines the numerical attributes of the relationship between two entities or entity sets.
- Different types of cardinal relationships are:
  - `One-to-One Relationships`
  - `One-to-Many Relationships`
  - `Many to One Relationship`
  - `Many-to-Many Relationships`



- When a relationship is designated as `mandatory one` , it means that for every instance of the entity on the "one" side of the relationship, there must be at least one related instance of the entity on the "many" side of the relationship. This ensures that every occurrence of the "one" entity is associated with at least one occurrence of the "many" entity.
- Vertical bar indicates the mandatory aspect of the relationship.
    - Here's an example to illustrate a "mandatory one" relationship:

      Consider two entities, "Department" and "Employee," in an organization database. Each department can have many employees, but each department must have at least one employee (i.e., a "mandatory one" relationship). In this scenario, if a department is created, it must have at least one employee associated with it; otherwise, it would violate the "mandatory one" constraint.

- When a relationship is designated as `mandatory many` it means that for every instance of the entity on the "many" side of the relationship, there must be at least one related instance of the entity on the "one" side of the relationship. This ensures that every occurrence of the "many" entity is associated with at least one occurrence of the "one" entity, and it allows for multiple associations between the two entities.
    - Here's an example to illustrate a "mandatory many" relationship:

      Consider two entities, "Course" and "Student," in an educational institution database. Each course can have many students, but each course must have at least one student enrolled (i.e., a "mandatory many" relationship). In this scenario, if a course is offered, it must have at least one student enrolled in it; otherwise, it would violate the "mandatory many" constraint. However, a course can have multiple students enrolled.

- when a relationship is designated as `optional one` , it means that an instance of the entity on the "one" side of the relationship may or may not be associated with an instance of the entity on the "many" side of the relationship. However, if there is an association, it can only be with a single instance of the entity on the "many" side.
- This circle indicates the optional aspect of the relationship

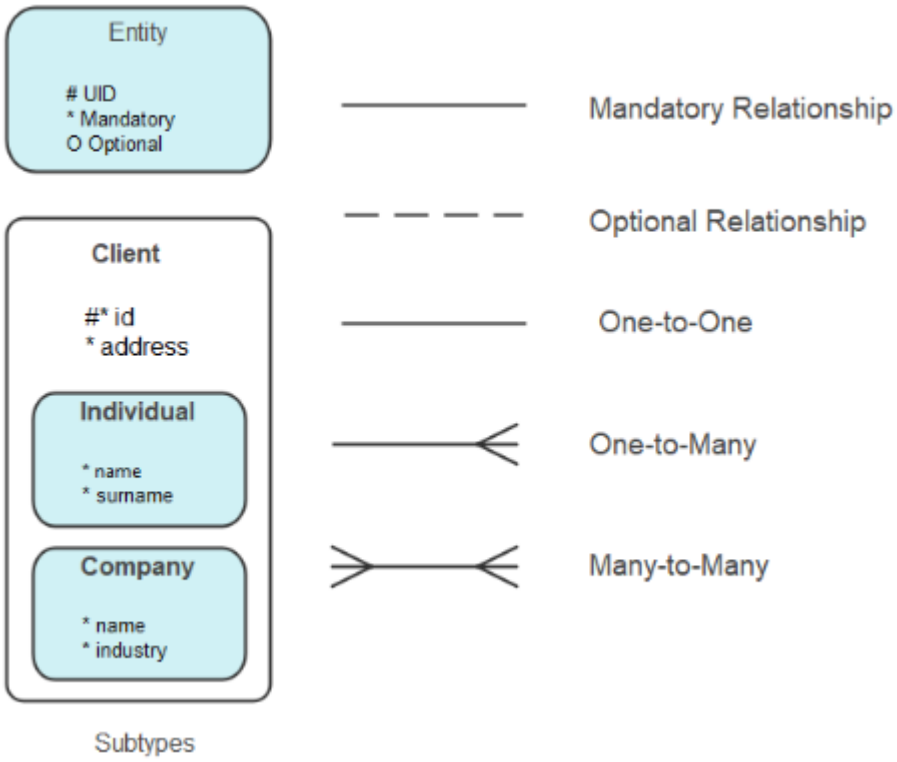
    - Here's an example to illustrate an "optional one" relationship:

      Consider two entities, "Department" and "Manager," in an organization database. Each department may or may not have a manager, but if it has a manager, it can have only one manager (i.e., an "optional one" relationship). In this scenario, some departments might not have a manager assigned, which is allowed due to the "optional one" constraint. However, if a department has a manager, it can have only one manager associated with it.

- when a relationship is designated as `optional many` it means that an instance of the entity on the "many" side of the relationship may or may not be associated with an instance of the entity on the "one" side of the relationship. Additionally, if there is an association, the "many" side can have multiple instances associated with the "one" side.
    - Here's an example to illustrate an "optional many" relationship:

      Consider two entities, "Project" and "Team Member," in a project management database. Each project may or may not have team members assigned, and if there are team members assigned, there can be multiple team members working on the project (i.e., an "optional many" relationship). In this scenario, some projects might not have any team members associated, which is allowed due to the "optional many" constraint. If a project has team members, it can have multiple team members working on it.

Relationship



One to one

- One entity from entity set X can be associated with at most one entity of entity set Y and vice versa.
- Example: One student can register for numerous courses. However, all those courses have a single line back to that one student.

One to many

- One entity from entity set X can be associated with multiple entities of entity set Y, but an entity from entity set Y can be associated with at least one entity.

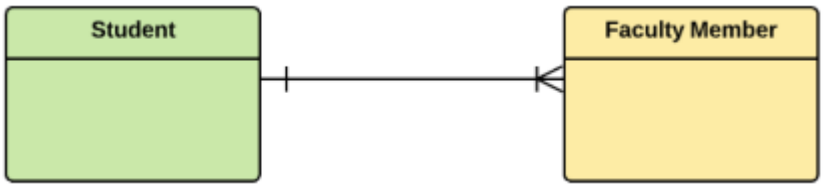
- For example, one class is consisting of multiple students.

Many to one

- More than one entity from entity set X can be associated with at most one entity of entity set Y. However, an entity from entity set Y may or may not be associated with more than one entity from entity set X.
- For example, many students belong to the same class.

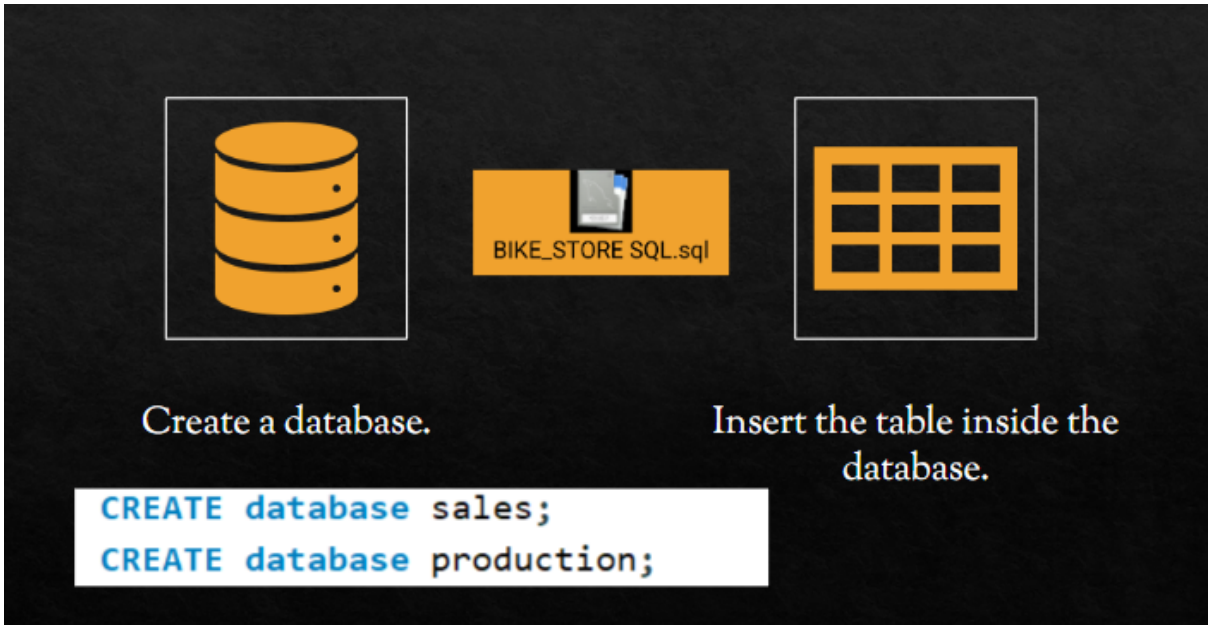
Many to many

- One entity from X can be associated with more than one entity from Y and vice versa.
- For example, Students as a group are associated with multiple faculty members, and faculty members can be associated with multiple students.

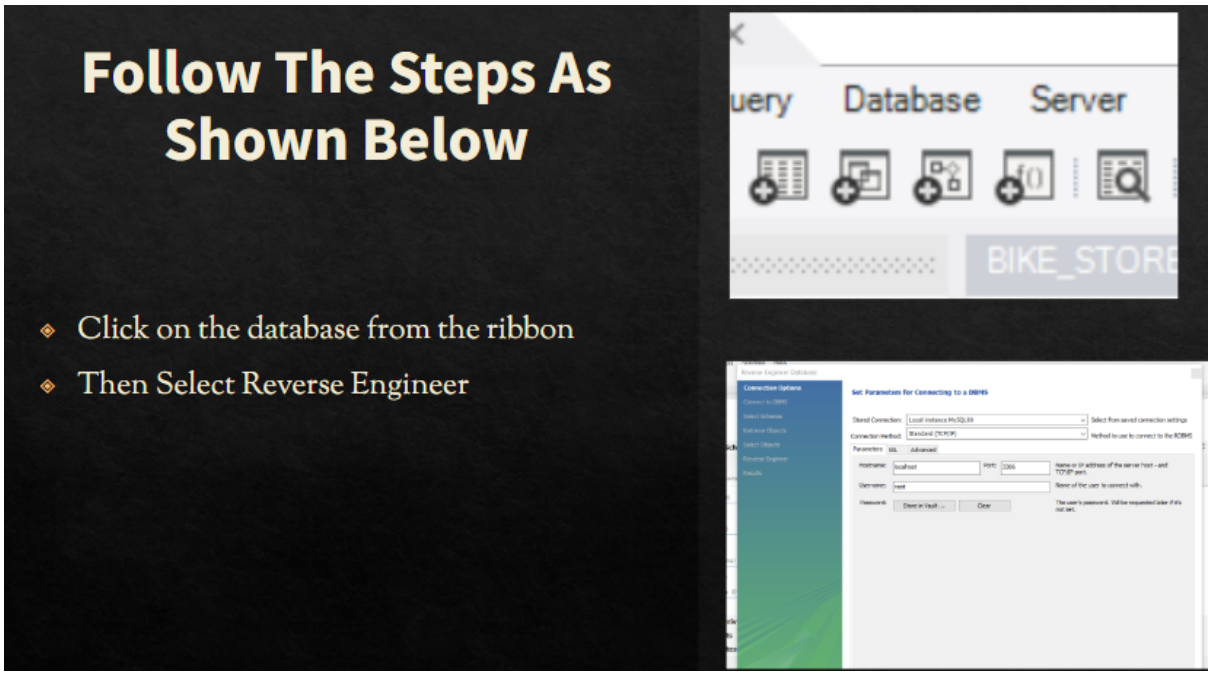


How to create an ERD in MySQL workbench

First

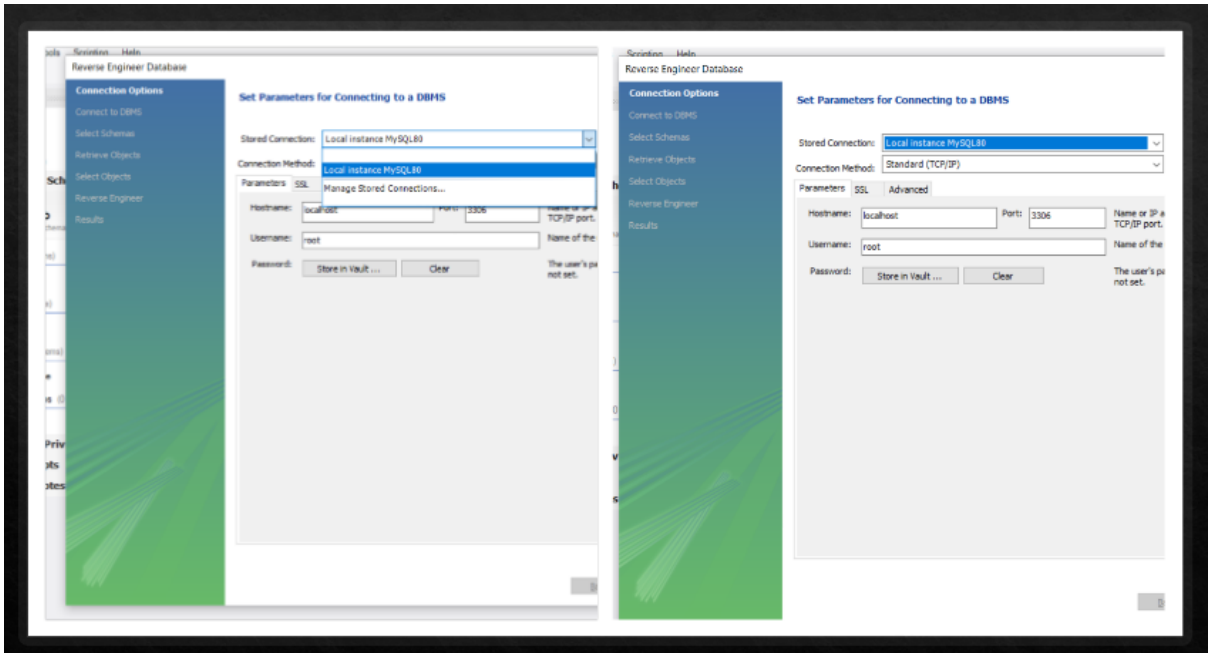


Second

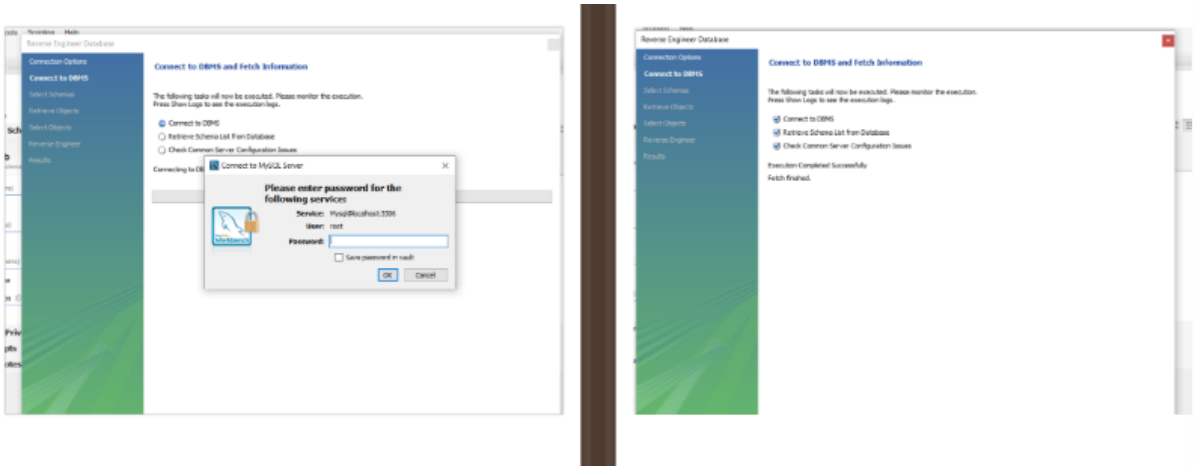


- Here Ribbon means Menu bar

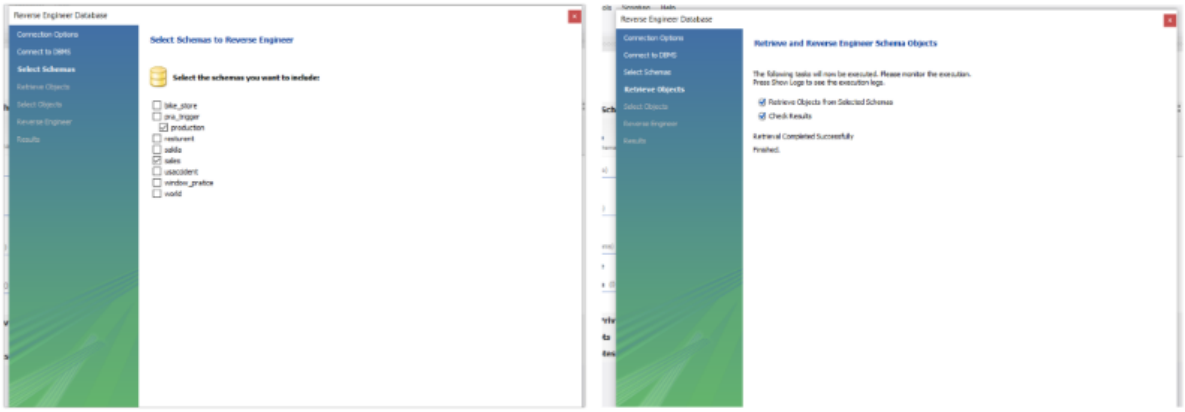
Third


Reverse Engineer Database
Connection Options
Connect to DBMS
Select Schemas
Retrieve Objects
Select Objects
Reverse Engineer
Results
Set Parameters for Connecting to a DBMS
Stored Connection:
Local instance MySQL80
Manage Stored Connections...
Connection Method:
Standard (TCP/IP)
Parameters
SSL
Advanced
Hostname:
localhost
Port:
3306
Username:
root
Password:
Store in Vault ...
Clear

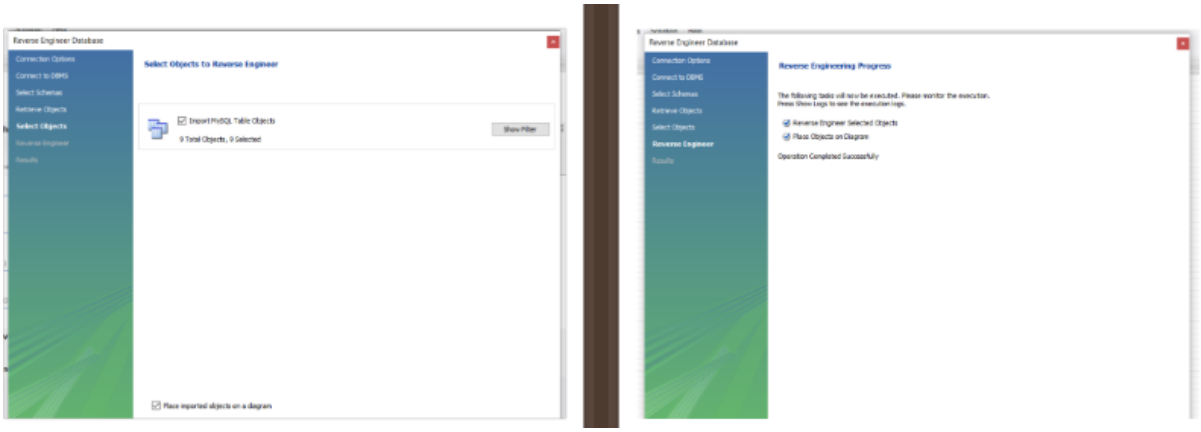
Forth


Connect to DBMS and Fetch Information
Connect to MySQL Server
Please enter password for the following services
OK
Cancel

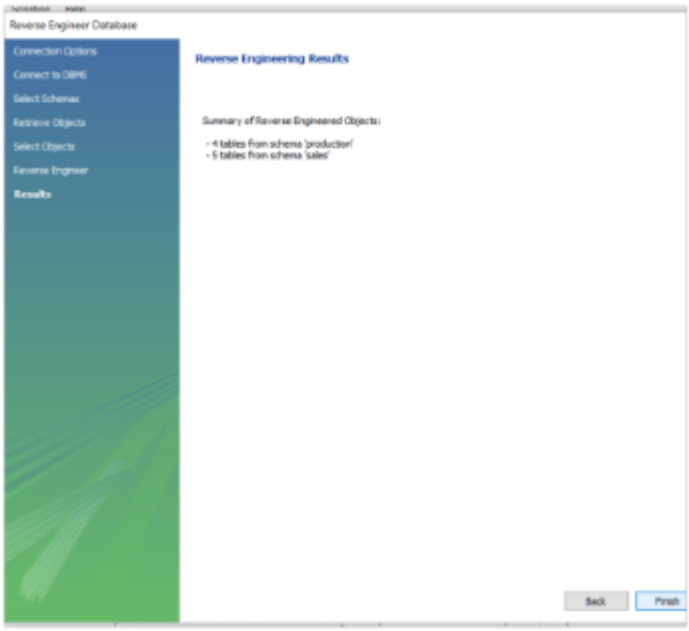
Fifth



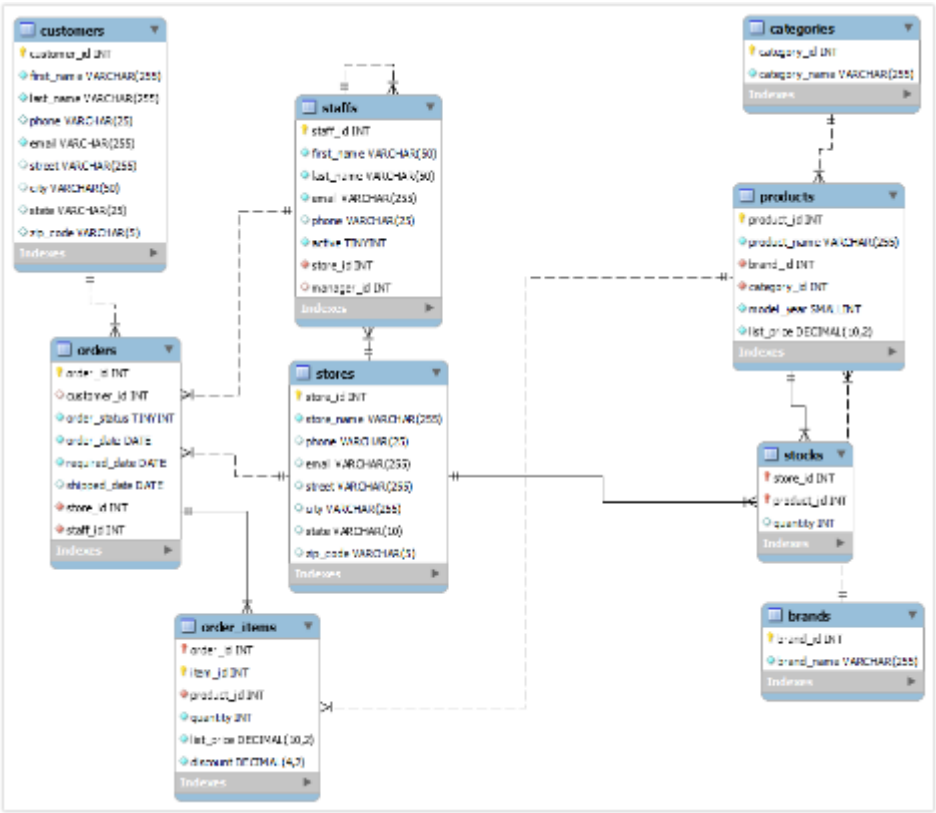
Sixth



Seventh


Reverse Engineer Database
Connection Options
Connect to DBMS
Select Schemas
Retrieve Objects
Select Objects
Reverse Engineer
Results
Reverse Engineering Results
Summary of Reverse Engineered Objects:
- 4 tables from schema 'production'
- 5 tables from schema 'sales'
Back
Finish

Result


customers
staffs
categories
products
orders
stores
stocks
order_items
brands