4. Database Design

4.1 Introduction

Database is a collection of related data. Relational database stores data in a table or relations. The data stored in a relation are arranged in records. Each record consists of set of attributes. Fields can be referred to characteristics of records. This document describes the table that is used to design software, its attributes, data types, constraints and relationship among those tables.

The design process consists of the following steps:

* Determine the purpose of your database. Find and organize the information required.
* Divide the information into tables. Turn information items into columns...
* Specify primary keys.
* Set up the table relationships.
* Refine your design.
* Apply the normalization rules.

4.2 Purpose and Scope

Purpose

* Avoid Redundant Data

The table in the database should be constructed following standards and with utmost dedication. It should have different fields and minimize redundant data. The table should always have a Primary Key that would be a unique id.

* Faultless Information

The database should follow the standards and conventions and provide meaningful information useful to the organization. (Constraint)

* Data Integrity

Integrity assists in guaranteeing that the values are valid and faultless. Data Integrity is set to tables, relationships, etc.

* Modify

The database developed should be worked upon with the conventions and standards, so that it can be easily modified whenever the need arises.

Scope

* Normalization of Database.
* Imposing Integrity Constraint.
* Establishing the Relation between the tables.
* Accessing the data from multiple tables. (Usage of join and sub query….)

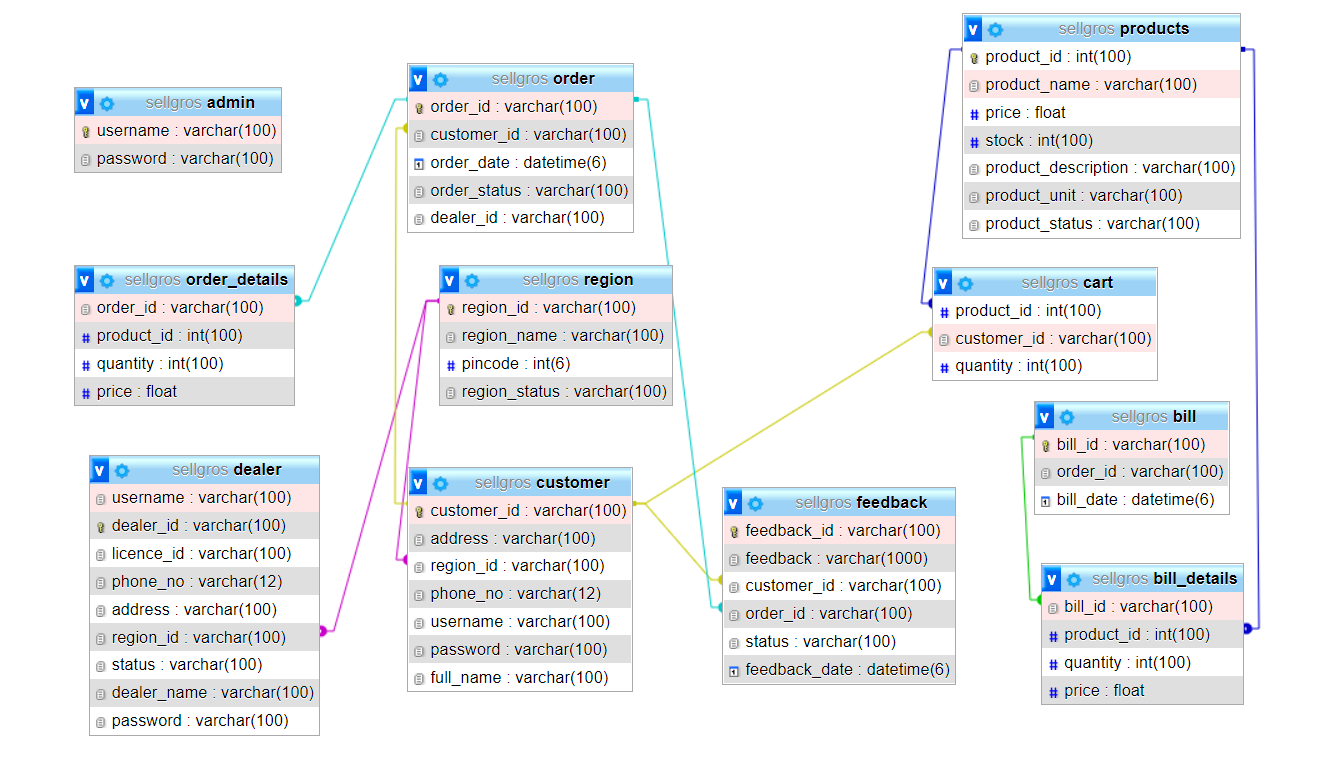
4.3 Database Identification

The identification of database by unique name given to the various database objects. The identifier is the name of database object. The following are the various database objects.

4.4 Schema Information

Database schema its structure described in a formal language supported by the Database Management System (DBMS). The term “schema” refers to the organization of the data as a blueprint of how database is constructed (divide into database tables in the case of relational databases)

In relational database, the schema defines the tables, fields, relationships, views, indexes, packages, procedures, functions, queues, triggers, types, sequences, materialized views.



4.5 Table Definition

4.5.1 Dealer table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data type | Size | Constraint | Description |
| username | varchar | 100 | Not null | User name of the dealer |
| dealer\_id | varchar | 100 | Not null (primary key) | Id of the dealer |
| licence\_id | varchar | 100 | Not null | License id of dealer |
| phone\_no | varchar | 12 | Not null | Phone no. of the dealer |
| address | varchar | 100 | Not null | Address of the dealer |
| region\_id | varchar | 100 | Not null (foreign key) | Region\_id of the dealer |
| status | varchar | 100 | Not null | Status of the dealer |
| dealer\_name | varchar | 100 | Not null | Name of the dealer |
| Password | varchar | 100 | Not null | Password of the dealer |

4.5.2 Customer table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data type | Size | Constraint | Description |
| customer\_id | Varchar | 100 | Not null (primary key) | Id of the customer |
| address | Varchar | 100 | Not null | Address of the customer |
| region\_id | Varchar | 100 | Not null (foreign key) | Region\_id of the customer |
| phone\_no | Varchar | 12 | Not null | Phone no. of the customer |
| username | Varchar | 100 | Not null | Username of the customer |
| password | Varchar | 100 | Not null | Password of the customer |
| full\_name | Varchar | 100 | Not null | Fullname of the customer |

4.5.3 Admin table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data type | Size | Constraint | Description |
| username | Varchar | 100 | Not null | Username of the admin |
| password | Varchar | 100 | Not null | Password of the admin |

4.5.4 Region table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data type | Size | Constraint | Description |
| region\_id | varchar (primary key) | 100 | Not null | Id of the region |
| region\_name | Varchar | 100 | Not null | Name of the region |
| pincode | Int | 6 | Not null | Pin code for the region |
| Region\_status | varchar | 100 | Not null | Status of the Region |

4.5.5 product table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data type | Size | Constraint | Description |
| product\_id | Int | 100 | Not null (primary key) | Id of the product |
| product\_name | Varchar | 100 | Not null | Name of the product |
| Price | Float |  | Not null | Price of the product |
| Stock | Int | 100 | Not null | Stock of the project |
| product\_description | Varchar | 100 | Not null | Description of the product |
| product\_unit | Varchar | 100 | Not null | Unit of the product |
| Product \_status | varchar | 100 | Not null | Status of the product |

4.5.6 order table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data type | Size | Constraint | Description |
| order\_id | Varchar | 100 | Not null (primary key) | Id of the order |
| customer\_id | Varchar | 100 | Not null (foreign key) | Id of the customer |
| order\_date | Datetime | 6 | Not null | Date of the order |
| order\_status | Varchar | 100 | Not null | Status of the order |
| dealer\_id | Varchar | 100 | null | Id of the dealer |

4.5.7 order\_details table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data type | Size | Constraint | Description |
| order\_id | Varchar | 100 | Not null (foreign key) | Id of the order |
| product\_id | Int | 100 | Not null (foreign key) | Id of the products |
| quantity | Int | 100 | Not null | Quantity of the product |
| price | float |  | Not null | Price of the product |

4.5.8 cart table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data type | Size | Constraint | Description |
| product\_id | Int | 100 | Not null (foreign key) | Id of the products |
| customer\_id | Varchar | 100 | Not null (foreign key) | Id of the customer |
| quantity | Int | 100 | Not null | Quantity of the product |

4.5.9 bill table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data type | Size | Constraint | Description |
| Bill\_id | Varchar | 100 | Not null(primary key) | Id of the bill |
| Order\_id | Varchar | 100 | Not null | Id of the order |
| Bill\_date | Datetime | 6 | Not null | Date of the bill |

4.5.10 bill\_details table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data type | Size | Constraint | Description |
| Bill\_id | Varchar | 100 | Not null (foreign key) | Id of the bill |
| Product\_id | Varchar | 100 | Not null (foreign key) | Id of the products |
| quantity | Int | 100 | Not null | Quantity of the products |
| price | Float |  | Not null | Price of the products |

4.5.11 Feedback table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column name | Data type | Size | Constraint | Description |
| feedback\_id | Varchar | 100 | Not null (primary key) | Id of the feedback |
| feedback | Varchar | 1000 | Not null | Feedback about order and dealer |
| customer\_id | Varchar | 100 | Not null (foreign key) | Id of the customer |
| order\_id | Varchar | 100 | Not null (foreign key) | Id of the order |
| status | Varchar | 100 | Not null | Status of the feedback |
| feedback\_date | Datetime | 6 | Not null | Date of the feedback |

4.6 Physical Design

Physical design is where you translate schemas into actual database structures.

* Entity to table
* Tuples to rows
* Attribute to columns
* Primary Key and Alternate Key to Unique Index
* Domain into Constraints

4.7 Data Dictionary

A data dictionary is a file or set of files that include metadata. The data dictionary holds records about other objects in the database, such as data ownership, data relationships to other objects, and other data.

The data dictionary, in general, includes information about the following:

* Name of the data item
* Aliases
* Description/purpose
* Related data items
* Range of values
* Data structure definition

4.8 ER Diagram

ER-modeling is a data modeling method used in software engineering to produce a conceptual data model of an information system. Diagram created using ER-modeling method are called Entry-Relationship diagram or ER-diagram or ERDs.

|  |  |
| --- | --- |
| Symbol | Conversion |
|  | Entity |
|  | Weak entity |
|  | Relationship |
|  | Identity relation |
|  | Attribute |
|  | Derived attribute |
| E  R  E  1  R | Cardinality ratio 1: N to E1:E2 to R |

Components of an ER-diagram

1. Entity

An entity can be a real word object, either animate or inanimate, that can be merely identifiable.

An entity is denoted as a rectangle in an ER diagram. For example, in a school database, students, teachers, classes, and courses offered can be treated as entities. All these entities have some attributes or properties that give them their identity.

Entity set

An entity set is a collection of related types of entities.

Strong entity

An entity with uniquely identified by its attribute.

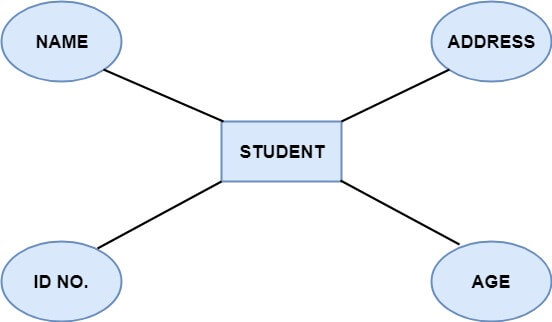
Weak Entity

In a relational database, a week entity is an entity that cannot be uniquely identified by its attributes alone.

1. Attributes

Entities are denoted utilizing their properties, known as attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

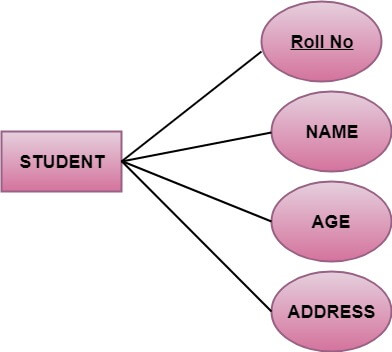
There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.



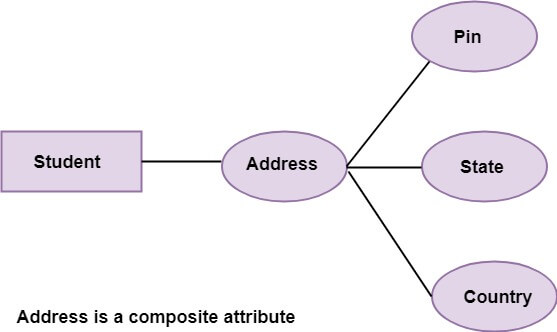
There are four types of Attributes:

* + - 1. Key attribute
      2. Composite attribute
      3. Single-valued attribute
      4. Multi-valued attribute
      5. Derived attribute

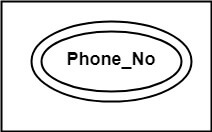
1. Key attribute: Key is an attribute or collection of attributes that uniquely identifies an entity among the entity set. For example, the roll number of a student makes him identifiable among students.



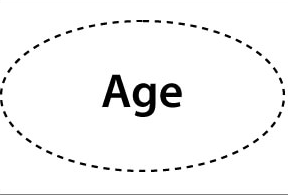
1. ****Composite attribute**:** An attribute that is a combination of other attributes is called a composite attribute. For example, in student entity, the student address is a composite attribute as an address is composed of other characteristics such as pin code, state, country.

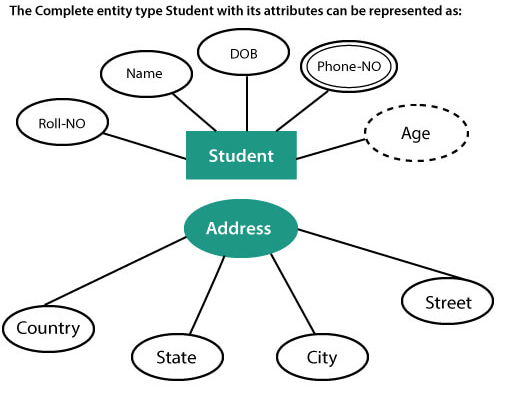


1. Single-valued attribute: Single-valued attribute contain a single value. For example, Social\_Security\_Number.
2. Multi-valued Attribute: If an attribute can have more than one value, it is known as a multi-valued attribute. Multi-valued attributes are depicted by the double ellipse. For example, a person can have more than one phone number, email-address, etc.



1. ****Derived attribute:**** Derived attributes are the attribute that does not exist in the physical database, but their values are derived from other attributes present in the database. For example, age can be derived from date\_of\_birth. In the ER diagram, Derived attributes are depicted by the dashed ellipse.





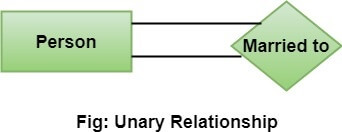
1. Relationships

The association among entities is known as relationship. Relationships are represented by the diamond-shaped box. For example, an employee works at a department, a student enrolls in a course. Here, Works at and enrolls are called relationships.

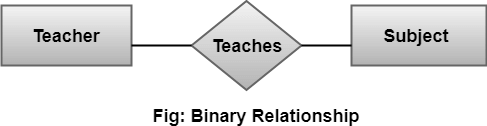
Degree of a relationship set

The number of participating entities in a relationship describes the degree of the relationship. The three most common relationships in E-R models are:

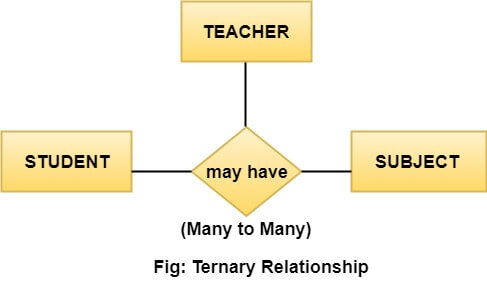
1. Unary (degree1)
2. Binary (degree2)
3. Ternary (degree3)
   1. Unary relationship: This is also called recursive relationships. It is a relationship between the instances of one entity type. For example, one person is married to only one person.



* 1. ****Binary relationship:**** It is a relationship between the instances of two entity types. For example, the Teacher teaches the subject.



* 1. ****Ternary relationship:**** It is a relationship amongst instances of three entity types. In fig, the relationships "****may have****" provide the association of three entities, i.e., TEACHER, STUDENT, and SUBJECT. All three entities are many-to-many participants. There may be one or many participants in a ternary relationship.



1. Cardinality Ratio

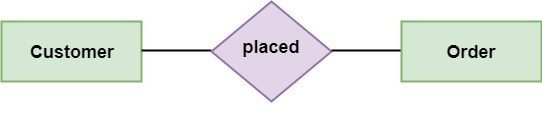
Cardinality describes the number of entities in one entity set, which can be associated with the number of entities of other sets via relationship set.

Types of Cardinalities

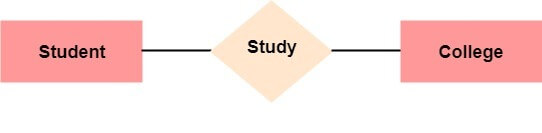
* 1. One to One: One entity from entity set A can be contained with at most one entity of entity set B and vice versa. Let us assume that each student has only one student ID, and each student ID is assigned to only one person. So, the relationship will be one to one.



* 1. ****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 relationships. For example, a client can place many orders; a order cannot be placed by many customers.



* 1. ****Many to One:**** More than one entity from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A. For example - many students can study in a single college, but a student cannot study in many colleges at the same time.

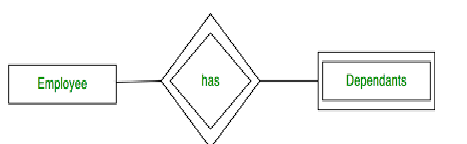


* 1. ****Many to Many:**** One entity from A can be associated with more than one entity from B and vice-versa. For example, the student can be assigned to many projects, and a project can be assigned to many students.



1. Identifying relationship

An identiffying relationship is a relationship between two entities in which an instance of a child entity is identified through its association with a parent entity, which means the child entity is dependent on the parent entity for its identity and cannot exits without it.

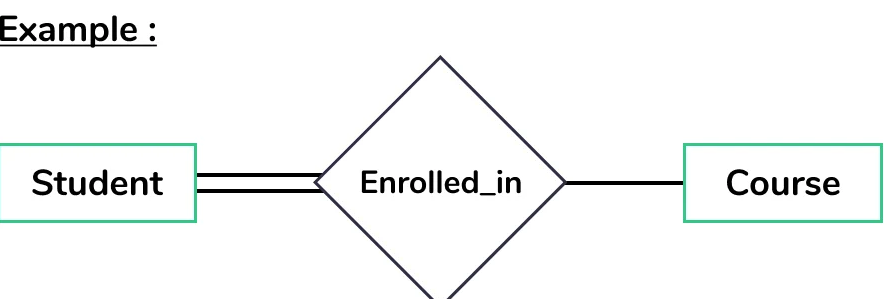


1. Participation Contraints

The participation constraint specifies the number of instance of an entity can participate in a relationship set.

Total participation – Each entity is involved in the relatioship. Total participation is represented by double lines.

Partial participation – Not all entities are involved in the relatioship. Partial participation is represented by single lines.



1

has

Region

Customer

Dealer

has

place

n

1

n

1

n

give

1

n

feedback

1

n

Order

has

Cart

has

products

1

n

n

1

1

has

n

Order\_details

n

has

1

have

1

1

bill

has

n

Bill\_details

1

adds

4.9 Database Administration

4.9.1 DBMS system Information

A database is an organized collection of structured information, or data typically stored electronically in a computer system. A data base is usually controlled by a database management system (DBMS).

In our project we are using MySQL database.

* + 1. DBMS Configuration

Steps for configure Apache and MySQL in XAMPP

* 1. In phpMyAdmin, click the Users tab at the top.
  2. Find the row that has User root and Host 127.0.0.1.
  3. Click Edit Privileges.
  4. Click Change password.
  5. Enter the password twice (write it down somewhere if you're sure, you can remember it)
  6. Click the Go button

4.9.3 Software Support Required

MySQL Required XAMPP

## Software Requirements

The following operating systems are officially supported:

* Windows xp or higher version (64-bit, Professional level or higher)
* Mac OS X 10.6.1 or higher
* Ubuntu 9.10 (64bit) or higher version

**4.9.4 Hardware (storage) requirements**

* Hard Disk:1 TB Required 500GB(Recommended)
  + - CPU: Intel Core 3GHz (or Dual Core 2GHz) or equal AMD CPU
* Cores: Dual (Quad Core is recommended)
* RAM: 4 GB (6 GB recommended)
  + - Graphic Accelerators: NVidia or ATI with support of OpenGL 1.5 or higher
    - Display Resolution: 1280×1024 is recommended, 1024×768 is minimum.

**4.9.5 Backup and Recover**

Recovery is the process of restoring a database to the correct state in the event of a failure

Database backup is a way to protect and restore a database. It is performed through database replication and can be done for a database or a database server.

Using phpMyAdmin to Back Up or Restore MySQL

If you’re running phpMyAdmin backing up and restoring your MySQL database is simple.

The export function is used as a backup, and the import function is used to restore.

Step 1: Create a MySQL Database Backup

1. Open phpMyAdmin. On the directory tree on the left, click the database you want to back up.

This should open the directory structure in the right-hand window. You’ll also notice that, in the directory tree on the left, all the assets under the main database are highlighted.

2. Click Export on the menu across the top of the display.

You’ll see a section called “Export Method.” Use Quick to save a copy of the whole database. Choose Custom to select individual tables or other special options.

Leave the Format field set to SQL, unless you have a good reason to change it.

3. Click Go. If you select Quick, your web browser will download a copy of the database into your specified downloads folder. You can copy that to a safe location.

Step 2: Clear the Old Database Information

It’s important to clear out old data before restoring a backup. If there’s any old data, it isn’t overwritten when you restore. This can create duplicate tables, causing errors and conflicts.

1. Open phpMyAdmin, on the navigation pane on the left, choose the database you want to restore.

2. Click the check all box near the bottom. Then, use the drop-down menu labeled with selected to select Drop.

3. The tool should prompt you to confirm that you want to go forward. Click yes.

This will get rid of all the existing data, clearing the way for your restoration.

Step 3: Restore Your Backed up MySQL Database

In phpMyAdmin, the Import tool is used to restore a database.

1. On the menu across the top, click Import.

2. The first section is labeled File to import. A couple of lines down, there’s a line that starts with “Browse your computer,” with a button labeled Choose File. Click that button.

3. Use the dialog box to navigate to the location where you’ve saved the export file that you want to restore. Leave all the options set to default. (If you created your backup with different options, you can select those here.)

4. Click Go.