**CHAPTER- I**

**INTRODUCTION**

***OBJECTIVE***:

To create a GUI of Database with for ***IMPORT AND EXPORT MANAGEMENT SYSTEM*** using MySQL and VB.Net.

**ABOUT:**

As your business takes on new staff, implements new processes and expands its operations, it’s important to have a robust business system that supports growth. If your goal is for your business to grow into a large enterprise, the best thing you can do is to start running it like one. This will help you improve productivity, cut costs, and keep your business running smoothly.

Thus this project will help an entrepreneur to develop their business at a vast and efficient scale.

The main aim of this project is to combat daily import and export database management.

The project is solely based on form application that is robust and mesmerizing in its own way which will help the user to effectively store information into the database.

This project has been developed with hard work and dedication so that its GUI shall be easier to understand even without the knowledge of its internal coding.

**CHAPTER-II**

**BACKGROUND**

***PURPOSE:***

The purpose of this project is to provide a GUI that is easy to interact with the user and the database.

***SCOPE:***

Scope of this project is to make import and export data retrieval easier and faster.

***CASE STUDY OF EXISTING SYSTEM:***

<http://www.indiantradeportal.in/> is an import and export website that deals with the imports and exports that occur in our nation India.



Fig. 2.1 case study

***Problems in Existing System:***

The GUI of this site is not understandable by a newbie using websites.

***Solution proposed by this project:*** The GUI of this project is simple to use and is easily understandable by any person even if he/she is not familiar with use of similar applications.

**CHAPTER-III**

**SYSTEM ANALYSIS**

***Here are the Projects system requirements (minimum)***

* CPU: Intel Core 2 Quad CPU Q6600 @ 2.40GHz (4 CPUs) / AMD Phenom 9850 Quad-Core Processor (4 CPUs) @ 2.5GHz
* CPU SPEED: Info
* RAM: 512 MB
* OS: Windows 10 64 Bit, Windows 8.1 64 Bit, Windows 8 64 Bit, Windows 7 64 Bit Service Pack 1, Windows Vista 64 Bit Service Pack 2\*, 32-bit also Supported.
* VIDEO CARD: NVIDIA 9800 GT 1GB / AMD HD 4870 1GB
* FREE DISK SPACE: 6.21 MB

***Projects Recommended Requirements***

* CPU: Intel Core i5 3470 @ 3.2GHz (4 CPUs) / AMD X8 FX-8350 @ 4GHz (8 CPUs)
* CPU SPEED: Info
* RAM: 2 GB
* OS: Windows 10 64 Bit, Windows 8.1 64 Bit, Windows 8 64 Bit, Windows 7 64 Bit Service Pack 1, 32-bit also Supported.
* VIDEO CARD: NVIDIA GTX 660 2GB / AMD HD 7870 2GB
* FREE DISK SPACE: 7.00 MB

**CHAPTER-IV**

**DBMS ARCHITECTURE**

A Database Management system is not always directly available for users and applications to access and store data in it. A Database Management system can be **centralised**(all the data stored at one location), **decentralised**(multiple copies of database at different locations) or **hierarchical**, depending upon its architecture.

**1-tier DBMS** architecture also exist, this is when the database is directly available to the user for using it to store data. Generally such a setup is used for local application development, where programmers communicate directly with the database for quick response.

Database Architecture is logically of two types:

1. 2-tier DBMS architecture
2. 3-tier DBMS architecture

## ***2-tier DBMS Architecture***

2-tier DBMS architecture includes an **Application layer** between the user and the DBMS, which is responsible to communicate the user's request to the database management system and then send the response from the DBMS to the user.

An application interface known as **ODBC**(Open Database Connectivity) provides an API that allow client side program to call the DBMS. Most DBMS vendors provide ODBC drivers for their DBMS.

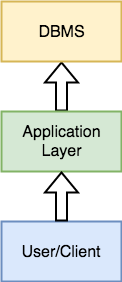


Fig. 4.1 2-tier architecture

Such an architecture provides the DBMS extra security as it is not exposed to the End User directly. Also, security can be improved by adding security and authentication checks in the Application layer too.

## ***3-tier DBMS Architecture***

3-tier DBMS architecture is the most commonly used architecture for web applications.

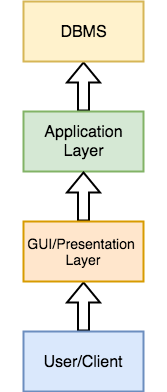


Fig. 4.2 3-tier architecture

It is an extension of the 2-tier architecture. In the 2-tier architecture, we have an application layer which can be accessed programatically to perform various operations on the DBMS. The application generally understands the Database Access Language and processes end users requests to the DBMS.

In 3-tier architecture, an additional Presentation or GUI Layer is added, which provides a graphical user interface for the End user to interact with the DBMS.

For the end user, the GUI layer is the Database System, and the end user has no idea about the application layer and the DBMS system.

If you have used **MySQL**, then you must have seen **PHPMyAdmin**, it is the best example of a 3-tier DBMS architecture.

CHOICE OF ARCHITECTURE FOR THIS PROJECT

***The Choice of Architecture for this project is a 2-tier Architecture.***

The GUI acts as the Presentation layer and utilizes the MySQL Database for the data to be shown to the user.

2-tier DBMS architecture includes an **Application layer** between the user and the DBMS, which is responsible to communicate the user's request to the database management system and then send the response from the DBMS to the user.

An application interface known as **ODBC**(Open Database Connectivity) provides an API that allow client side program to call the DBMS. Most DBMS vendors provide ODBC drivers for their DBMS.

Such an architecture provides the DBMS extra security as it is not exposed to the End User directly. Also, security can be improved by adding security and authentication checks in the Application layer too.

**Almost all Desktop applications are 2-tier applications**

**CHAPTER -V**

# **DATA MODELING**

Data modeling is the process of documenting a complex software system design as an easily understood diagram, using text and symbols to represent the way [data](https://searchdatamanagement.techtarget.com/definition/data) needs to flow. The diagram can be used as a blueprint for the construction of new software or for re-engineering a legacy application.

Traditionally, data models have been built during the analysis and design phases of a project to ensure that the requirements for a new application are fully understood. A data model can be thought of as a [flowchart](https://whatis.techtarget.com/definition/flowchart) that illustrates the relationships between data. Although capturing all the possible relationships in a data model can be very time-intensive, it's an important step that shouldn't be rushed. Well-documented conceptual, logical and physical data models allow stake-holders to identify errors and make changes before any programming [code](https://whatis.techtarget.com/definition/code) has been written.

Data modelers often use multiple models to view the same data and ensure that all processes, entities, relationships and data flows have been identified. There are several different approaches to data modeling, including:

**Conceptual Data Modeling** - identifies the highest-level relationships between different entities.

**Enterprise Data Modeling** - similar to conceptual data modeling, but addresses the unique requirements of a specific business.

**Logical Data Modeling** - illustrates the specific entities, attributes and relationships involved in a business function. Serves as the basis for the creation of the physical data model.

**Physical Data Modeling** - represents an application and database-specific implementation of a logical data model.

***DATABASE SCHEMA***

The **database schema** of a [database system](https://en.wikipedia.org/wiki/Database_system) is its structure described in a [formal language](https://en.wikipedia.org/wiki/Formal_language) supported by the [database management system](https://en.wikipedia.org/wiki/Database_management_system) (DBMS). The term "[schema](https://en.wiktionary.org/wiki/schema)" refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of [relational databases](https://en.wikipedia.org/wiki/Relational_databases)). The formal definition of a [database](https://en.wikipedia.org/wiki/Database) schema is a set of formulas (sentences) called [integrity constraints](https://en.wikipedia.org/wiki/Integrity_constraints) imposed on a database.[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the [database language](https://en.wikipedia.org/wiki/Database_language).[[1]](https://en.wikipedia.org/wiki/Database_schema#cite_note-source1-1)The states of a created [conceptual schema](https://en.wikipedia.org/wiki/Conceptual_schema) are transformed into an [explicit mapping](https://en.wikipedia.org/wiki/Explicit_and_implicit_methods), the database schema. This describes how real-world entities are [modeled](https://en.wikipedia.org/wiki/Data_modeling) in the database.

"A database schema specifies, based on the [database administrator](https://en.wikipedia.org/wiki/Database_administrator)'s knowledge of possible applications, the facts that can enter the database, or those of interest to the possible [end-users](https://en.wikipedia.org/wiki/End-user)."[[2]](https://en.wikipedia.org/wiki/Database_schema#cite_note-source3-2) The notion of a database schema plays the same role as the notion of theory in [predicate calculus](https://en.wikipedia.org/wiki/Predicate_calculus). A model of this "theory" closely corresponds to a database, which can be seen at any instant of time as a [mathematical object](https://en.wikipedia.org/wiki/Mathematical_object). Thus a schema can contain formulas representing [integrity constraints](https://en.wikipedia.org/wiki/Data_integrity#Types_of_integrity_constraints) specifically for an application and the constraints specifically for a type of database, all expressed in the same database language.[[1]](https://en.wikipedia.org/wiki/Database_schema#cite_note-source1-1) In a [relational database](https://en.wikipedia.org/wiki/Relational_database), the schema defines the [tables](https://en.wikipedia.org/wiki/Table_(database)), [fields](https://en.wikipedia.org/wiki/Field_(computer_science)), [relationships](https://en.wikipedia.org/wiki/Relational_model), [views](https://en.wikipedia.org/wiki/View_(database)), [indexes](https://en.wikipedia.org/wiki/Index_(database)), [packages](https://en.wikipedia.org/wiki/Software_package_(installation)), [procedures](https://en.wikipedia.org/wiki/Stored_procedure), [functions](https://en.wikipedia.org/wiki/Subroutine), [queues](https://en.wikipedia.org/wiki/Queue_(data_structure)), [triggers](https://en.wikipedia.org/wiki/Database_trigger), [types](https://en.wikipedia.org/wiki/Data_type), [sequences](https://en.wikipedia.org/wiki/Sequence), [materialized views](https://en.wikipedia.org/wiki/Materialized_view), [synonyms](https://en.wikipedia.org/wiki/Synonym_(database)), database links, [directories](https://en.wikipedia.org/wiki/Directory_(file_systems)), [XML schemas](https://en.wikipedia.org/wiki/XML_schema), and other elements.

A database generally stores its schema in a [data dictionary](https://en.wikipedia.org/wiki/Data_dictionary). Although a schema is defined in text database language, the term is often used to refer to a graphical depiction of the database structure. In other words, schema is the structure of the database that defines the objects in the database.

In an [Oracle Database](https://en.wikipedia.org/wiki/Oracle_Database) system, the term "schema" has a slightly different connotation.

***ER MODEL***

ER Model is used to model the logical view of the system from data perspective which consists of these components:

**Entity, Entity Type, Entity Set –**

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

An Entity is an object of Entity Type and set of all entities is called as entity set. e.g.; E1 is an entity having Entity Type Student and set of all students is called Entity Set. In ER diagram, Entity Type is represented as:

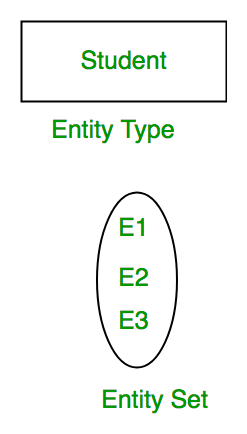


Fig 5.1 Entity

**Attribute(s):**  
Attributes are the **properties which define the entity type**. For example, Roll\_No, Name, DOB, Age, Address, Mobile\_No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval.

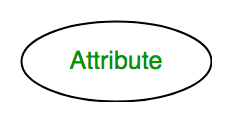


Fig 5.2 Attribute

1. **Key Attribute –**  
   The attribute which **uniquely identifies each entity** in the entity set is called key attribute.For example, Roll\_No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.
2. **Composite Attribute –**  
   An attribute **composed of many other attribute** is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals.



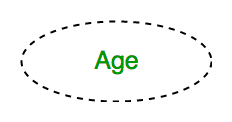
Fig 5.3 composite attribute

1. **Multivalued Attribute –**  
   An attribute consisting **more than one value** for a given entity. For example, Phone\_No (can be more than one for a given student). In ER diagram, multivalued attribute is represented by double oval.



Fig 5.4 multivalued attribute

1. **Derived Attribute –**  
   An attribute which can be **derived from other attributes** of the entity type is known as derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, derived attribute is represented by dashed oval. Represented as,



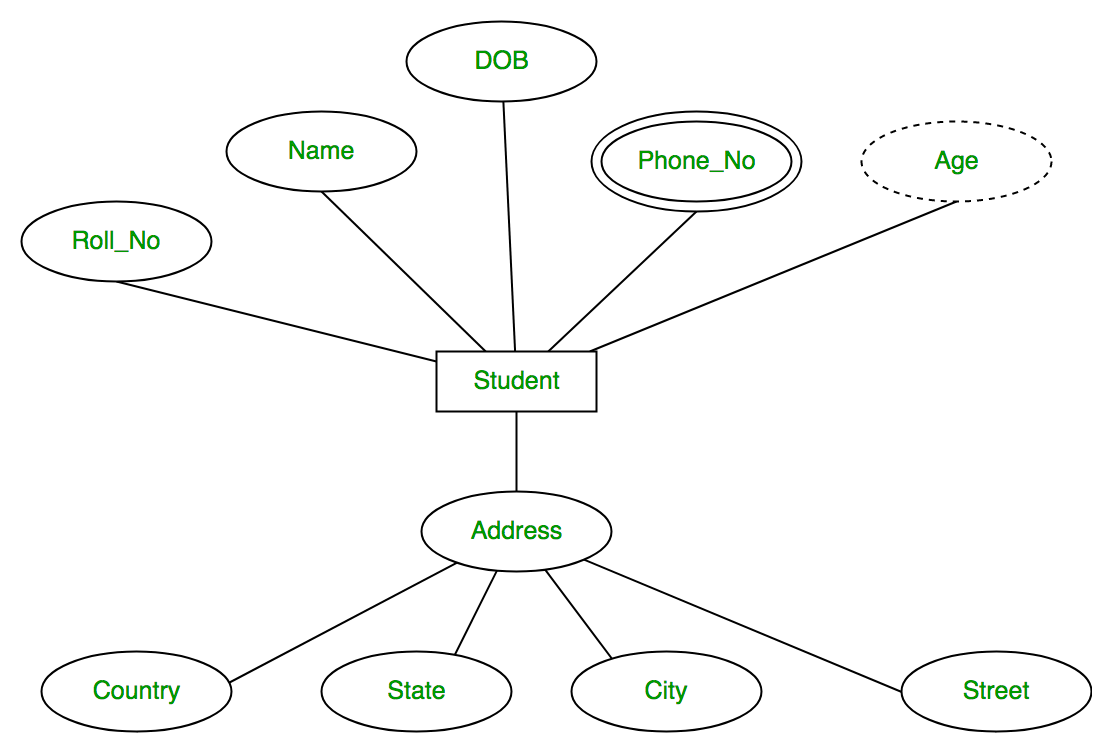


Fig 5.5 Derived attribute

The complete entity type**Student** with its attributes

**Relationship Type and Relationship Set:**  
A relationship type represents the **association between entity types**. For example,‘Enrolled in’ is a relationship type that exists between entity type Student and Course. In ER diagram, relationship type is represented by a diamond and connecting the entities with lines.

A set of relationships of same type is known as relationship set. The following relationship set depicts S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled in C3.



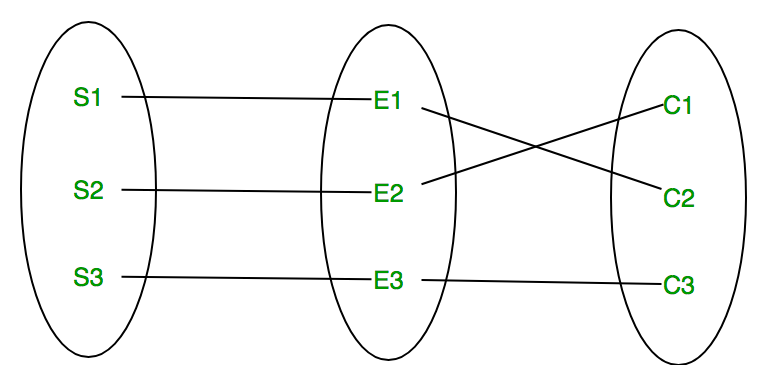


Fig 5.6 Relationship type

**Degree of a relationship set:**  
The number of different entity sets **participating in a relationship** set is called as degree of a relationship set.

1. **Unary Relationship –**  
   When there is **only ONE entity set participating in a relation**, the relationship is called as unary relationship. For example, one person is married to only one person.
2. **Binary Relationship –**  
   When there are **TWO entities set participating in a relation**, the relationship is called as binary relationship.For example, Student is enrolled in Course.
3. **n-ary Relationship –**  
   When there are n entities set participating in a relation, the relationship is called as n-ary relationship.

**Cardinality:**  
The **number of times an entity of an entity set participates in a relationship** set is known as cardinality. Cardinality can be of different types:

1. **One to one –** When each entity in each entity set can take part **only once in the relationship**, the cardinality is one to one. Let us assume that a male can marry to one female and a female can marry to one male. So the relationship will be one to one.

Using Sets, it can be represented as:

1. **Many to one –** When entities in one entity set **can take part only once in the relationship set and entities in other entity set can take part more than once in the relationship set,**cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1. It means that for one course there can be n students but for one student, there will be only one course.

Using Sets, it can be represented as:

In this case, each student is taking only 1 course but 1 course has been taken by many students.

1. **Many to many –** When entities in all entity sets can **take part more than once in the relationship** cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.

Using sets, it can be represented as:

In this example, student S1 is enrolled in C1 and C3 and Course C3 is enrolled by S1, S3 and S4. So it is many to many relationships.

**Participation Constraint:**  
Participation Constraint is applied on the entity participating in the relationship set.

1. **Total Participation –** Each entity in the entity set**must participate** in the relationship. If each student must enroll in a course, the participation of student will be total. Total participation is shown by double line in ER diagram.
2. **Partial Participation –** The entity in the entity set **may or may NOT participat**e in the relationship. If some courses are not enrolled by any of the student, the participation of course will be partial.

The diagram depicts the ‘Enrolled in’ relationship set with Student Entity set having total participation and Course Entity set having partial participation.

Every student in Student Entity set is participating in relationship but there exists a course C4 which is not taking part in the relationship.

**Weak Entity Type and Identifying Relationship:**  
As discussed before, an entity type has a key attribute which uniquely identifies each entity in the entity set. But there exists **some entity type for which key attribute can’t be defined**. These are called Weak Entity type.

For example, A company may store the information of dependants (Parents, Children, Spouse) of an Employee. But the dependents don’t have existence without the employee. So Dependent will be weak entity type and Employee will be Identifying Entity type for Dependant.

A weak entity type is represented by a double rectangle. The participation of weak entity type is always total. The relationship between weak entity type and its identifying strong entity type is called identifying relationship and it is represented by double diamond.

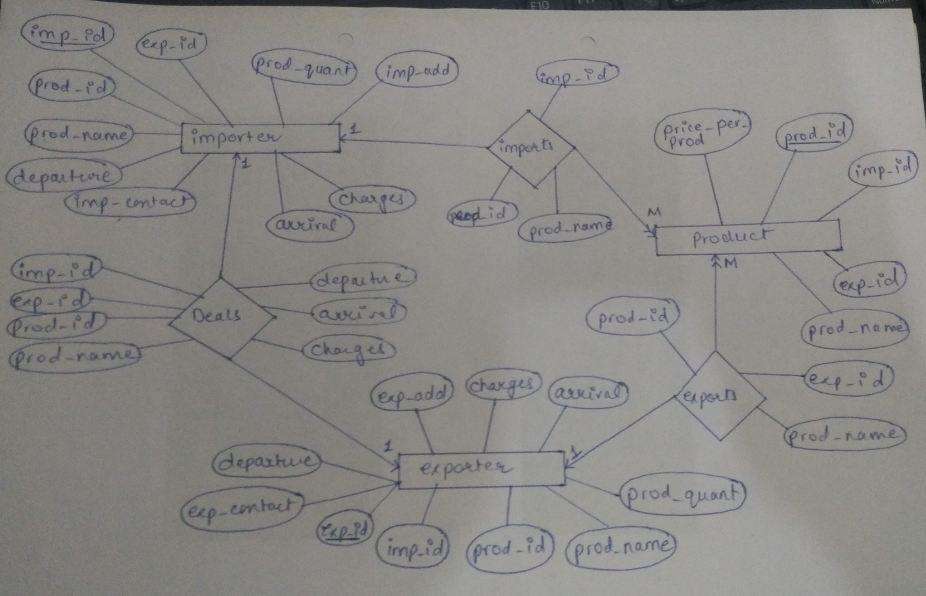


Fig 5.7 ER diagram

**CHAPTER -VI**

**DATABASE DESIGN**

# ***Normalization in DBMS: 1NF, 2NF, 3NF and BCNF in Database***

***Normalization*** is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. Let’s discuss about anomalies first then we will discuss normal forms with examples.

## **Anomalies in DBMS**

There are three types of anomalies that occur when the database is not normalized. These are – Insertion, update and deletion anomaly. Let’s take an example to understand this.

**Example**: Suppose a manufacturing company stores the employee details in a table named employee that has four attributes: emp\_id for storing employee’s id, emp\_name for storing employee’s name, emp\_address for storing employee’s address and emp\_dept for storing the department details in which the employee works. At some point of time the table looks like this:

|  |  |  |  |
| --- | --- | --- | --- |
| emp\_id | emp\_name | emp\_address | emp\_dept |
| 101 | Rick | Delhi | D001 |
| 101 | Rick | Delhi | D002 |
| 123 | Maggie | Agra | D890 |
| 166 | Glenn | Chennai | D900 |
| 166 | Glenn | Chennai | D004 |

Table 6.1 Anomalies

The above table is not normalized. We will see the problems that we face when a table is not normalized.

**Update anomaly**: In the above table we have two rows for employee Rick as he belongs to two departments of the company. If we want to update the address of Rick then we have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.

**Insert anomaly**: Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp\_dept field doesn’t allow nulls.

**Delete anomaly**: Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having emp\_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department.

To overcome these anomalies we need to normalize the data. In the next section we will discuss about normalization.

## ***Normalization***

Here are the most commonly used normal forms:

* First normal form(1NF)
* Second normal form(2NF)
* Third normal form(3NF)
* Boyce & Codd normal form (BCNF)

## **First normal form (1NF)**

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

**Example**: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

|  |  |  |  |
| --- | --- | --- | --- |
| emp\_id | emp\_name | emp\_address | emp\_mobile |
| 101 | Herschel | New Delhi | 8912312390 |
| 102 | Jon | Kanpur | 8812121212  9900012222 |
| 103 | Ron | Chennai | 7778881212 |
| 104 | Lester | Bangalore | 9990000123  8123450987 |

Table 6.2 1NF (a)

Two employees (Jon & Lester) are having two mobile numbers so the company stored them in the same field as you can see in the table above.

This table is **not in 1NF**as the rule says “each attribute of a table must have atomic (single) values”, the emp\_mobile values for employees Jon & Lester violates that rule.

To make the table complies with 1NF we should have the data like this:

|  |  |  |  |
| --- | --- | --- | --- |
| emp\_id | emp\_name | emp\_address | emp\_mobile |
| 101 | Herschel | New Delhi | 8912312390 |
| 102 | Jon | Kanpur | 8812121212 |
| 102 | Jon | Kanpur | 9900012222 |
| 103 | Ron | Chennai | 7778881212 |
| 104 | Lester | Bangalore | 9990000123 |
| 104 | Lester | Bangalore | 8123450987 |

Table 6.3 1NF (b)

## **Second normal form (2NF)**

A table is said to be in 2NF if both the following conditions hold:

* Table is in 1NF (First normal form)
* No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

**Example**: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

|  |  |  |
| --- | --- | --- |
| teacher\_id | subject | teacher\_age |
| 111 | Maths | 38 |
| 111 | Physics | 38 |
| 222 | Biology | 38 |
| 333 | Physics | 40 |
| 333 | Chemistry | 40 |

Table 6.5 2NF(a)

**Candidate Keys**: {teacher\_id, subject}  
**Non prime attribute**: teacher\_age

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher\_age is dependent on teacher\_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says “**no** non-prime attribute is dependent on the proper subset of any candidate key of the table”.

To make the table complies with 2NF we can break it in two tables like this:  
**teacher\_details table:**

|  |  |
| --- | --- |
| teacher\_id | teacher\_age |
| 111 | 38 |
| 222 | 38 |
| 333 | 40 |

Table 6.4 2NF (b)

**teacher\_subject table:**

|  |  |
| --- | --- |
| teacher\_id | subject |
| 111 | Maths |
| 111 | Physics |
| 222 | Biology |
| 333 | Physics |
| 333 | Chemistry |

Table 6.5 2NF(c)

Now the tables comply with Second normal form (2NF).

## **Third Normal form (3NF)**

A table design is said to be in 3NF if both the following conditions hold:

* Table must be in 2NF
* [Transitive functional dependency](https://beginnersbook.com/2015/04/transitive-dependency-in-dbms/) of non-prime attribute on any super key should be removed.

An attribute that is not part of any [candidate key](https://beginnersbook.com/2015/04/candidate-key-in-dbms/) is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

* X is a [super key](https://beginnersbook.com/2015/04/super-key-in-dbms/) of table
* Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

**Example**: Suppose a company wants to store the complete address of each employee, they create a table named employee\_details that looks like this:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| emp\_id | emp\_name | emp\_zip | emp\_state | emp\_city | emp\_district |
| 1001 | John | 282005 | UP | Agra | Dayal Bagh |
| 1002 | Ajeet | 222008 | TN | Chennai | M-City |
| 1006 | Lora | 282007 | TN | Chennai | Urrapakkam |
| 1101 | Lilly | 292008 | UK | Pauri | Bhagwan |
| 1201 | Steve | 222999 | MP | Gwalior | Ratan |

  Table 6.6 3NF (a)

**Super keys**: {emp\_id}, {emp\_id, emp\_name}, {emp\_id, emp\_name, emp\_zip}…so on  
**Candidate Keys**: {emp\_id}  
**Non-prime attributes**: all attributes except emp\_id are non-prime as they are not part of any candidate keys.

Here, emp\_state, emp\_city & emp\_district dependent on emp\_zip. And, emp\_zip is dependent on emp\_id that makes non-prime attributes (emp\_state, emp\_city & emp\_district) transitively dependent on super key (emp\_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

**employee table:**

|  |  |  |
| --- | --- | --- |
| emp\_id | emp\_name | emp\_zip |
| 1001 | John | 282005 |
| 1002 | Ajeet | 222008 |
| 1006 | Lora | 282007 |
| 1101 | Lilly | 292008 |
| 1201 | Steve | 222999 |

Table 6.7 3NF (b)

**employee\_zip table:**

|  |  |  |  |
| --- | --- | --- | --- |
| emp\_zip | emp\_state | emp\_city | emp\_district |
| 282005 | UP | Agra | Dayal Bagh |
| 222008 | TN | Chennai | M-City |
| 282007 | TN | Chennai | Urrapakkam |
| 292008 | UK | Pauri | Bhagwan |
| 222999 | MP | Gwalior | Ratan |

Table 6.8 3NF (c)

## **Boyce Codd normal form (BCNF)**

It is an advance version of 3NF that’s why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every [functional dependency](https://beginnersbook.com/2015/04/functional-dependency-in-dbms/) X->Y, X should be the super key of the table.

**Example**: Suppose there is a company wherein employees work in **more than one department**. They store the data like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| emp\_id | emp\_nationality | emp\_dept | dept\_type | dept\_no\_of\_emp |
| 1001 | Austrian | Production and planning | D001 | 200 |
| 1001 | Austrian | stores | D001 | 250 |
| 1002 | American | design and technical support | D134 | 100 |
| 1002 | American | Purchasing department | D134 | 600 |

Table 6.9 BCNF (a)

**Functional dependencies in the table above**:  
emp\_id -> emp\_nationality  
emp\_dept -> {dept\_type, dept\_no\_of\_emp}

**Candidate key**: {emp\_id, emp\_dept}

The table is not in BCNF as neither emp\_id nor emp\_dept alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:  
**emp\_nationality table:**

|  |  |
| --- | --- |
| emp\_id | emp\_nationality |
| 1001 | Austrian |
| 1002 | American |

**emp\_dept table:**

|  |  |  |
| --- | --- | --- |
| emp\_dept | dept\_type | dept\_no\_of\_emp |
| Production and planning | D001 | 200 |
| stores | D001 | 250 |
| design and technical support | D134 | 100 |
| Purchasing department | D134 | 600 |

Table 6.10 BCNF (b)

**emp\_dept\_mapping table:**

|  |  |
| --- | --- |
| emp\_id | emp\_dept |
| 1001 | Production and planning |
| 1001 | stores |
| 1002 | design and technical support |
| 1002 | Purchasing department |

**Functional dependencies**:  
emp\_id -> emp\_nationality  
emp\_dept -> {dept\_type, dept\_no\_of\_emp}

**Candidate keys**:  
For first table: emp\_id  
For second table: emp\_dept  
For third table: {emp\_id, emp\_dept}

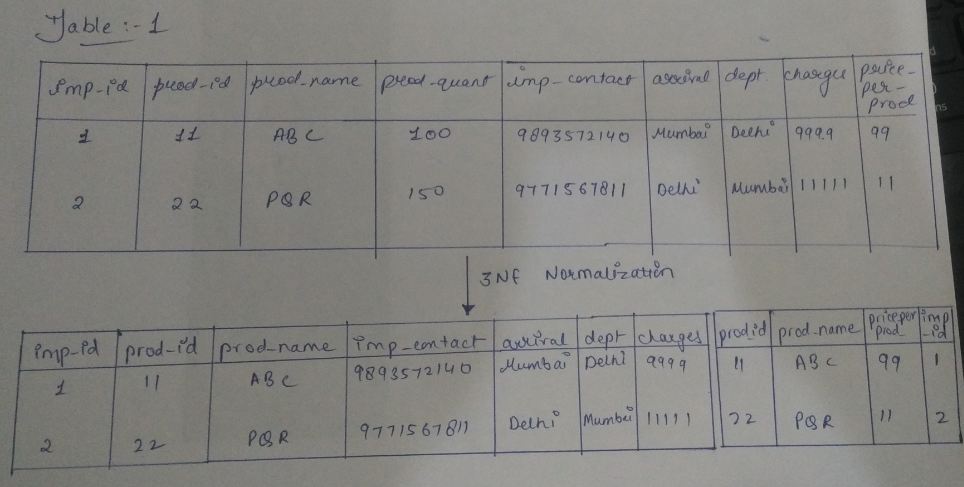
This is now in BCNF as in both the functional dependencies left side part is a key.

Table 6.11 Normalization (a)

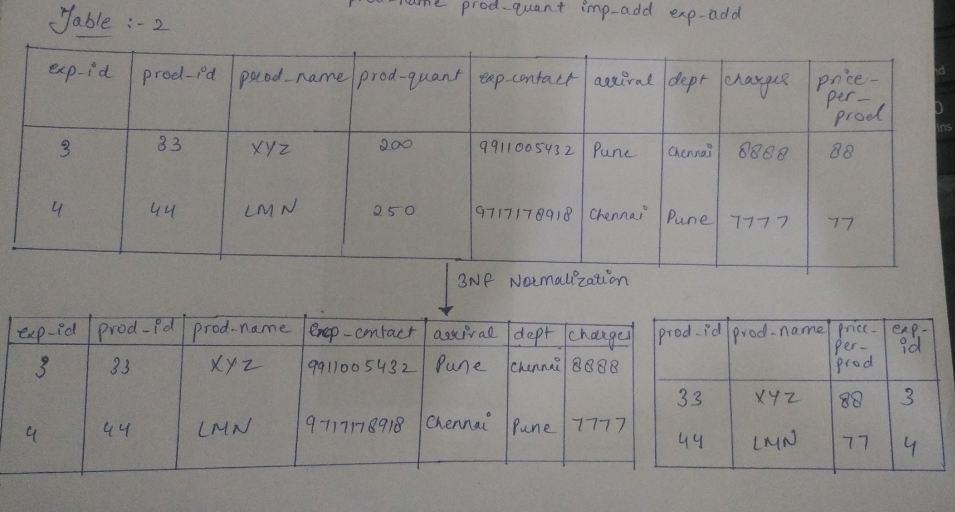


Table 6.12 Normalization (b)

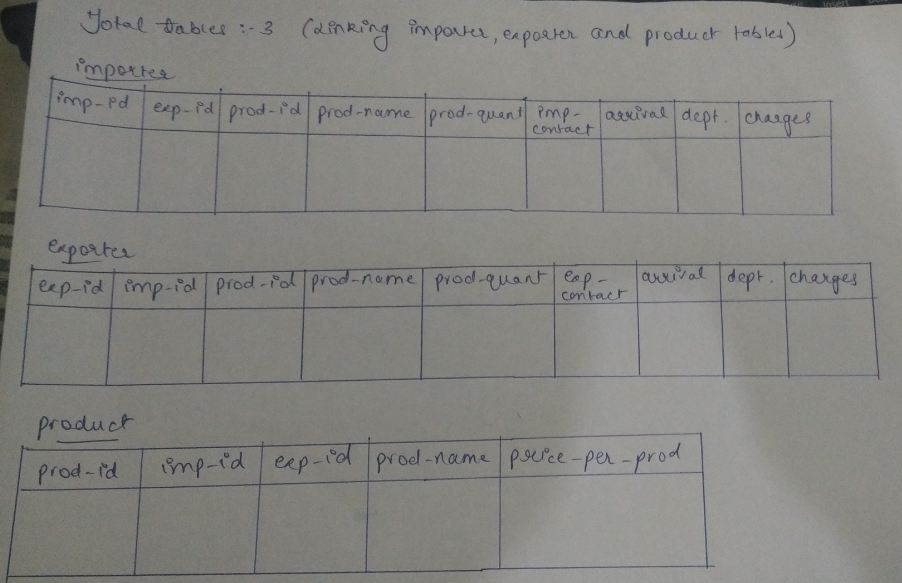


Table 6.13 Normalization (c)

**CHAPTER -VII**

**MODEL MAPPING**

ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand. ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram. We cannot import all the ER constraints into relational model, but an approximate schema can be generated.

There are several processes and algorithms available to convert ER Diagrams into Relational Schema. Some of them are automated and some of them are manual. We may focus here on the mapping diagram contents to relational basics.

ER diagrams mainly comprise of −

* Entity and its attributes
* Relationship, which is association among entities.

## **Mapping Entity**

An entity is a real-world object with some attributes.

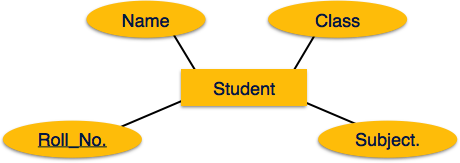


Fig 7.1 Mapping Entity

### Mapping Process (Algorithm)

* Create table for each entity.
* Entity's attributes should become fields of tables with their respective data types.
* Declare primary key.

## **Mapping Relationship**

A relationship is an association among entities.

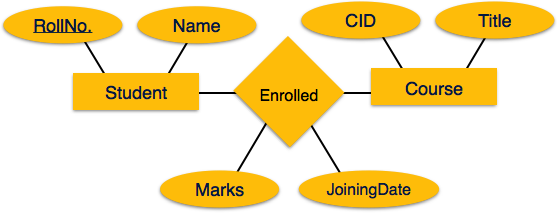


Fig 7.2 Mapping Relationship

### Mapping Process

* Create table for a relationship.
* Add the primary keys of all participating Entities as fields of table with their respective data types.
* If relationship has any attribute, add each attribute as field of table.
* Declare a primary key composing all the primary keys of participating entities.
* Declare all foreign key constraints.

## **Mapping Weak Entity Sets**

A weak entity set is one which does not have any primary key associated with it.

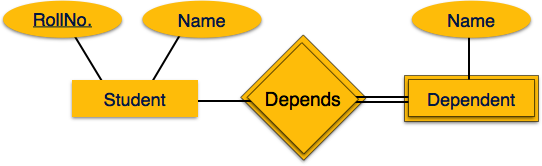


Fig 7.3 Mapping Weak Entity

### Mapping Process

* Create table for weak entity set.
* Add all its attributes to table as field.
* Add the primary key of identifying entity set.
* Declare all foreign key constraints.

## **Mapping Hierarchical Entities**

ER specialization or generalization comes in the form of hierarchical entity sets.

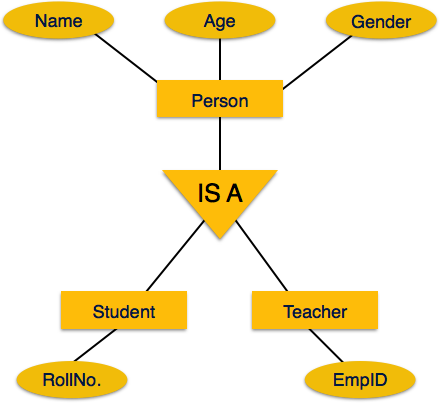


Fig 7.4 Mapping Hierarchical Entities

### Mapping Process

* Create tables for all higher-level entities.
* Create tables for lower-level entities.
* Add primary keys of higher-level entities in the table of lower-level entities.
* In lower-level tables, add all other attributes of lower-level entities.
* Declare primary key of higher-level table and the primary key for lower-level table.
* Declare foreign key constraints.

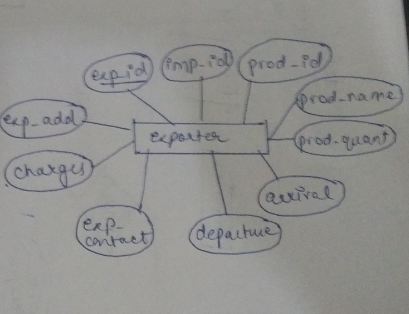


Fig 7.5 ER model for Exporter

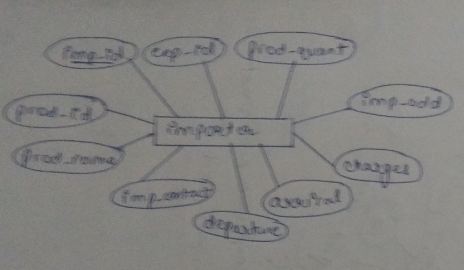


Fig 7.5 ER model for Importer

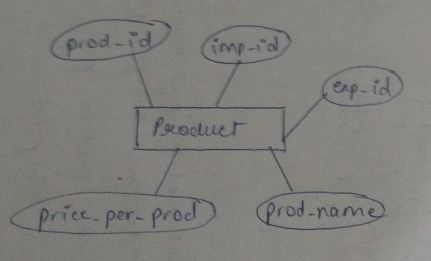


Fig 7.5 ER model for Product

**CHAPTER -VIII**

**USER INTERFACE DESIGN**

The integrated development environment used to build the GUI of this project is VB.Net.

**Visual Basic .NET** (**VB.NET**) is a [multi-paradigm](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) [programming language](https://en.wikipedia.org/wiki/Programming_language), implemented on the [.NET Framework](https://en.wikipedia.org/wiki/.NET_Framework). Microsoft launched VB.NET in 2002 as the successor to its original [Visual Basic](https://en.wikipedia.org/wiki/Visual_Basic) language. Although the ".NET" portion of the name was dropped in 2005, this article uses "Visual Basic [.NET]" to refer to all Visual Basic languages releases since 2002, in order to distinguish between them and the [classic Visual Basic](https://en.wikipedia.org/wiki/Visual_Basic). Along with [Visual C#](https://en.wikipedia.org/wiki/Microsoft_Visual_C_Sharp), it is one of the two main languages targeting the .NET framework.

Microsoft's [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for developing in Visual Basic .NET language is [Visual Studio](https://en.wikipedia.org/wiki/Visual_Studio). Most Visual Studio editions are [commercial](https://en.wikipedia.org/wiki/Commercial_software); the only exceptions are [Visual Studio Express](https://en.wikipedia.org/wiki/Visual_Studio_Express) and [Visual Studio Community](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio#Visual_Studio_Community), which are [freeware](https://en.wikipedia.org/wiki/Freeware). In addition, the [.NET Framework SDK](https://en.wikipedia.org/wiki/.NET_Framework_SDK) includes a freeware [command-line](https://en.wikipedia.org/wiki/Command-line)[compiler](https://en.wikipedia.org/wiki/Compiler) called vbc.exe. [Mono](https://en.wikipedia.org/wiki/Mono_(software)) also includes a command-line VB.NET compiler.

Let's start with creating a Window Forms Application by following the following steps in Microsoft Visual Studio - **File → New Project → Windows Forms Applications**

Finally, select OK, Microsoft Visual Studio creates your project and displays following window Form with a name **Form1**.

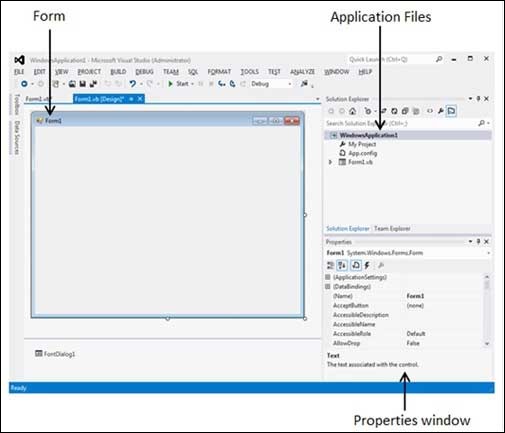


Fig 8.1 Windows Form1

Visual Basic Form is the container for all the controls that make up the user interface. Every window you see in a running visual basic application is a form, thus the terms form and window describe the same entity. Visual Studio creates a default form for you when you create a **Windows Forms Application**.

Every form will have title bar on which the form's caption is displayed and there will be buttons to close, maximize and minimize the form shown below −

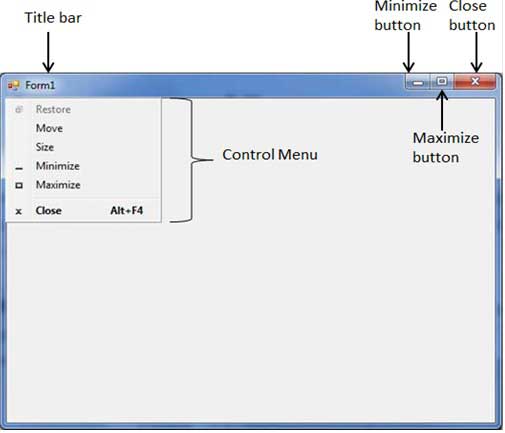


Fig 8.2 Windows Form2

If you click the icon on the top left corner, it opens the control menu, which contains the various commands to control the form like to move control from one place to another place, to maximize or minimize the form or to close the form.

# **VB.Net - Basic Controls**

An object is a type of user interface element you create on a Visual Basic form by using a toolbox control. In fact, in Visual Basic, the form itself is an object. Every Visual Basic control consists of three important elements −

* **Properties** which describe the object,
* **Methods** cause an object to do something and
* **Events** are what happens when an object does something.

## **Control Properties**

All the Visual Basic Objects can be moved, resized or customized by setting their properties. A property is a value or characteristic held by a Visual Basic object, such as Caption or Fore Color.

Properties can be set at design time by using the Properties window or at run time by using statements in the program code.

Object. Property = Value

Where

* **Object** is the name of the object you're customizing.
* **Property** is the characteristic you want to change.
* **Value** is the new property setting.

For example,

Form1.Caption = "Hello"

You can set any of the form properties using Properties Window. Most of the properties can be set or read during application execution. You can refer to Microsoft documentation for a complete list of properties associated with different controls and restrictions applied to them.

## **Control Methods**

A method is a procedure created as a member of a class and they cause an object to do something. Methods are used to access or manipulate the characteristics of an object or a variable. There are mainly two categories of methods you will use in your classes −

* If you are using a control such as one of those provided by the Toolbox, you can call any of its public methods. The requirements of such a method depend on the class being used.
* If none of the existing methods can perform your desired task, you can add a method to a class.

For example, the *MessageBox* control has a method named *Show, which is called in the code snippet below −*

Public Class Form1

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)

Handles Button1.Click

MessageBox.Show("Hello, World")

End Sub

End Class

## **Control Events**

An event is a signal that informs an application that something important has occurred. For example, when a user clicks a control on a form, the form can raise a **Click** event and call a procedure that handles the event. There are various types of events associated with a Form like click, double click, close, load, resize, etc.

Following is the default structure of a form **Load** event handler subroutine. You can see this code by double clicking the code which will give you a complete list of the all events associated with Form control −

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'event handler code goes here

End Sub

Here, **Handles MyBase.Load** indicates that **Form1\_Load()** subroutine handles **Load** event. Similar way, you can check stub code for click, double click. If you want to initialize some variables like properties, etc., then you will keep such code inside Form1\_Load() subroutine. Here, important point to note is the name of the event handler, which is by default Form1\_Load, but you can change this name based on your naming convention you use in your application programming.

## **Basic Controls**

VB.Net provides a huge variety of controls that help you to create rich user interface. Functionalities of all these controls are defined in the respective control classes. The control classes are defined in the **System.Windows.Forms** namespace.

The following table lists some of the commonly used controls −

|  |  |
| --- | --- |
| **Sr.No.** | **Widget & Description** |
| 1 | [**Forms**](https://www.tutorialspoint.com/vb.net/vb.net_forms.htm)  The container for all the controls that make up the user interface. |
| 2 | [**TextBox**](https://www.tutorialspoint.com/vb.net/vb.net_textbox.htm)  It represents a Windows text box control. |
| 3 | [**Label**](https://www.tutorialspoint.com/vb.net/vb.net_label.htm)  It represents a standard Windows label. |
| 4 | [**Button**](https://www.tutorialspoint.com/vb.net/vb.net_button.htm)  It represents a Windows button control. |
| 5 | [**ListBox**](https://www.tutorialspoint.com/vb.net/vb.net_listbox.htm)  It represents a Windows control to display a list of items. |
| 6 | [**ComboBox**](https://www.tutorialspoint.com/vb.net/vb.net_combobox.htm)  It represents a Windows combo box control. |
| 7 | [**RadioButton**](https://www.tutorialspoint.com/vb.net/vb.net_radio_button.htm)  It enables the user to select a single option from a group of choices when paired with other RadioButton controls. |
| 8 | [**CheckBox**](https://www.tutorialspoint.com/vb.net/vb.net_checkbox.htm)  It represents a Windows CheckBox. |
| 9 | [**PictureBox**](https://www.tutorialspoint.com/vb.net/vb.net_picturebox.htm)  It represents a Windows picture box control for displaying an image. |
| 10 | [**ProgressBar**](https://www.tutorialspoint.com/vb.net/vb.net_progress_bar.htm)  It represents a Windows progress bar control. |
| 11 | [**ScrollBar**](https://www.tutorialspoint.com/vb.net/vb.net_scrollbar.htm)  It Implements the basic functionality of a scroll bar control. |
| 12 | [**DateTimePicker**](https://www.tutorialspoint.com/vb.net/vb.net_date_time_picker.htm)  It represents a Windows control that allows the user to select a date and a time and to display the date and time with a specified format. |
| 13 | [**TreeView**](https://www.tutorialspoint.com/vb.net/vb.net_treeview.htm)  It displays a hierarchical collection of labeled items, each represented by a TreeNode. |
| 14 | [**ListView**](https://www.tutorialspoint.com/vb.net/vb.net_listview.htm)  It represents a Windows list view control, which displays a collection of items that can be displayed using one of four different views. |

Table 8.1 Control Events

**CHAPTER –IX**

**DATABASE CONNECTIVITY**

**MySQL has been used to provide a database to this project.**

**MySQL** is an [open-source](https://en.wikipedia.org/wiki/Open-source) [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS). Its name is a combination of "My", the name of co-founder [Michael Widenius](https://en.wikipedia.org/wiki/Michael_Widenius)'s daughter, and "[SQL](https://en.wikipedia.org/wiki/SQL)", the abbreviation for [Structured Query Language](https://en.wikipedia.org/wiki/Structured_Query_Language). The MySQL development project has made its [source code](https://en.wikipedia.org/wiki/Source_code)available under the terms of the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License), as well as under a variety of [proprietary](https://en.wikipedia.org/wiki/Proprietary_software) agreements. MySQL was owned and sponsored by a single [for-profit](https://en.wikipedia.org/wiki/Business) firm, the [Swedish](https://en.wikipedia.org/wiki/Sweden) company [MySQL AB](https://en.wikipedia.org/wiki/MySQL_AB), now owned by [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation).[[8]](https://en.wikipedia.org/wiki/MySQL#cite_note-sunacquire-8)For proprietary use, several paid editions are available, and offer additional functionality.

MySQL is a central component of the [LAMP](https://en.wikipedia.org/wiki/LAMP_(software_bundle)) open-source web application software stack (and other "[AMP](https://en.wikipedia.org/wiki/List_of_AMP_packages)" stacks). LAMP is an acronym for "[Linux](https://en.wikipedia.org/wiki/Linux), [Apache](https://en.wikipedia.org/wiki/Apache_HTTP_Server), MySQL, [Perl](https://en.wikipedia.org/wiki/Perl)/[PHP](https://en.wikipedia.org/wiki/PHP)/[Python](https://en.wikipedia.org/wiki/Python_(programming_language))". Applications that use the MySQL database include: [TYPO3](https://en.wikipedia.org/wiki/TYPO3), [MODx](https://en.wikipedia.org/wiki/MODx), [Joomla](https://en.wikipedia.org/wiki/Joomla), [WordPress](https://en.wikipedia.org/wiki/WordPress), [Simple Machines Forum](https://en.wikipedia.org/wiki/Simple_Machines_Forum), [phpBB](https://en.wikipedia.org/wiki/PhpBB), [MyBB](https://en.wikipedia.org/wiki/MyBB), and [Drupal](https://en.wikipedia.org/wiki/Drupal). MySQL is also used in many high-profile, large-scale [websites](https://en.wikipedia.org/wiki/Website), including [Google](https://en.wikipedia.org/wiki/Google) (though not for searches), [Facebook](https://en.wikipedia.org/wiki/Facebook), [Twitter](https://en.wikipedia.org/wiki/Twitter), [Flickr](https://en.wikipedia.org/wiki/Flickr), and [YouTube](https://en.wikipedia.org/wiki/YouTube).

CREATE AND CONNECT MySQL DATABASE TO VB.Net.

Startup Queries:

create schema importexport;

create table importexport.signin

(

username varchar(30) unique,

password varchar(30) unique,

primary key(username)

);

INSERT INTO importexport.signin (username, password) VALUES (‘aamirhafiez’, ‘hafiezaamir’);

INSERT INTO importexport.signin (username, password) VALUES (‘amarnathpandey’, ‘pandeynathamar’);

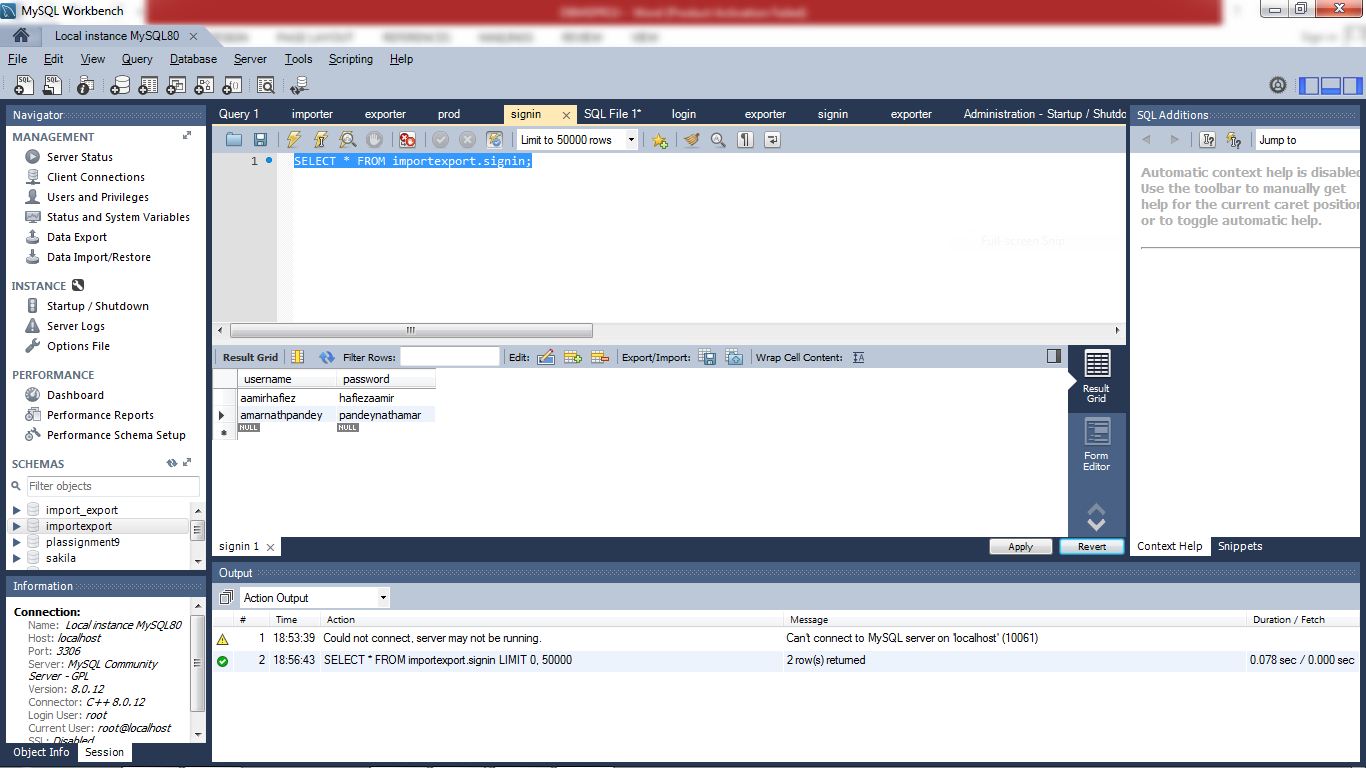


Fig 9.1 MySQL Database1

For Importer table:

create table importexport.importer

(

imp\_id int unique,

exp\_id int unique,

prod\_id int unique,

prod\_name varchar(50) not null,

prod\_quant int default 0,

imp\_add varchar(50),

charges int default 0,

arrival date,

departure date,

imp\_contact bigint,

primary key(imp\_id)

);

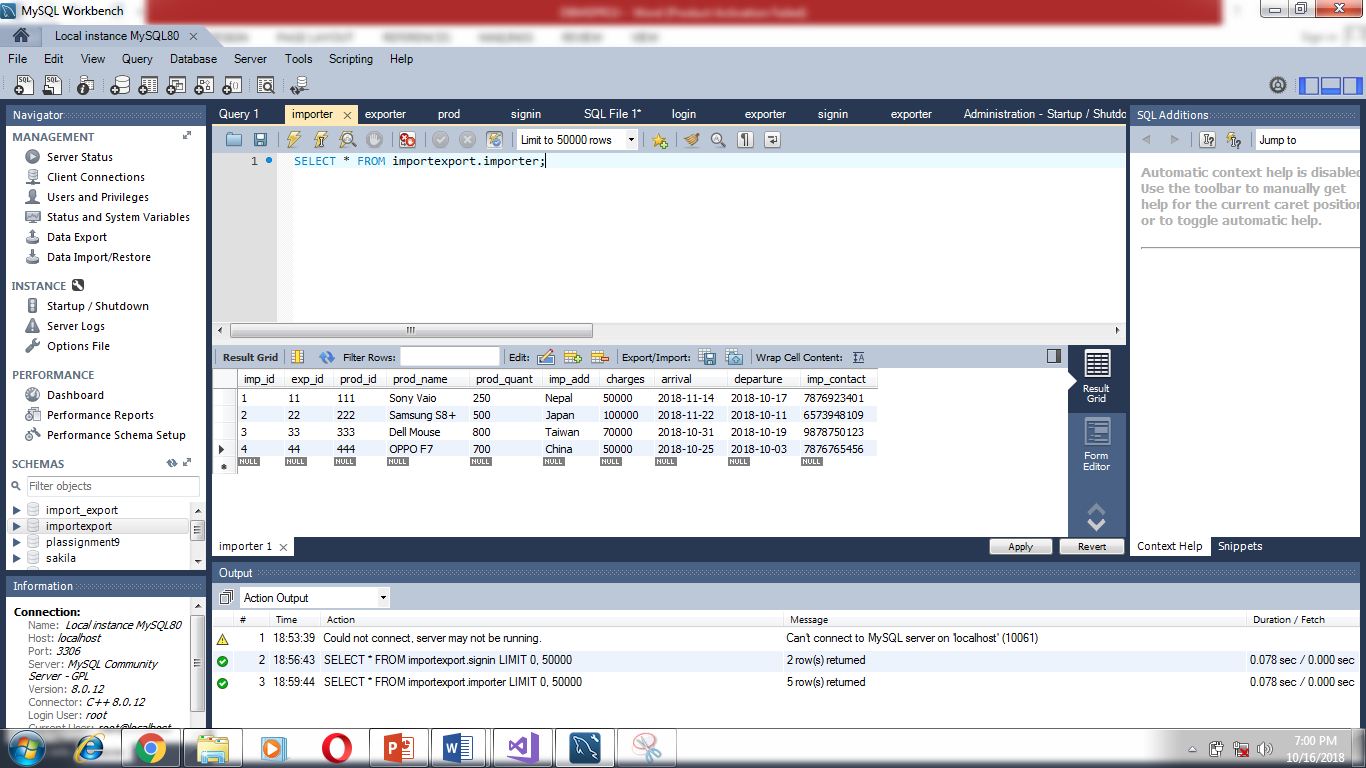


Fig 9.2 MySQL Database2

For Exporter table:

create table importexport.exporter

(

exp\_id int unique,

imp\_id int unique,

prod\_id int unique,

prod\_name varchar(50) not null,

prod\_quant int default 0,

exp\_add varchar(50),

charges int default 0,

arrival date,

departure date,

exp\_contact bigint,

primary key(exp\_id)

);

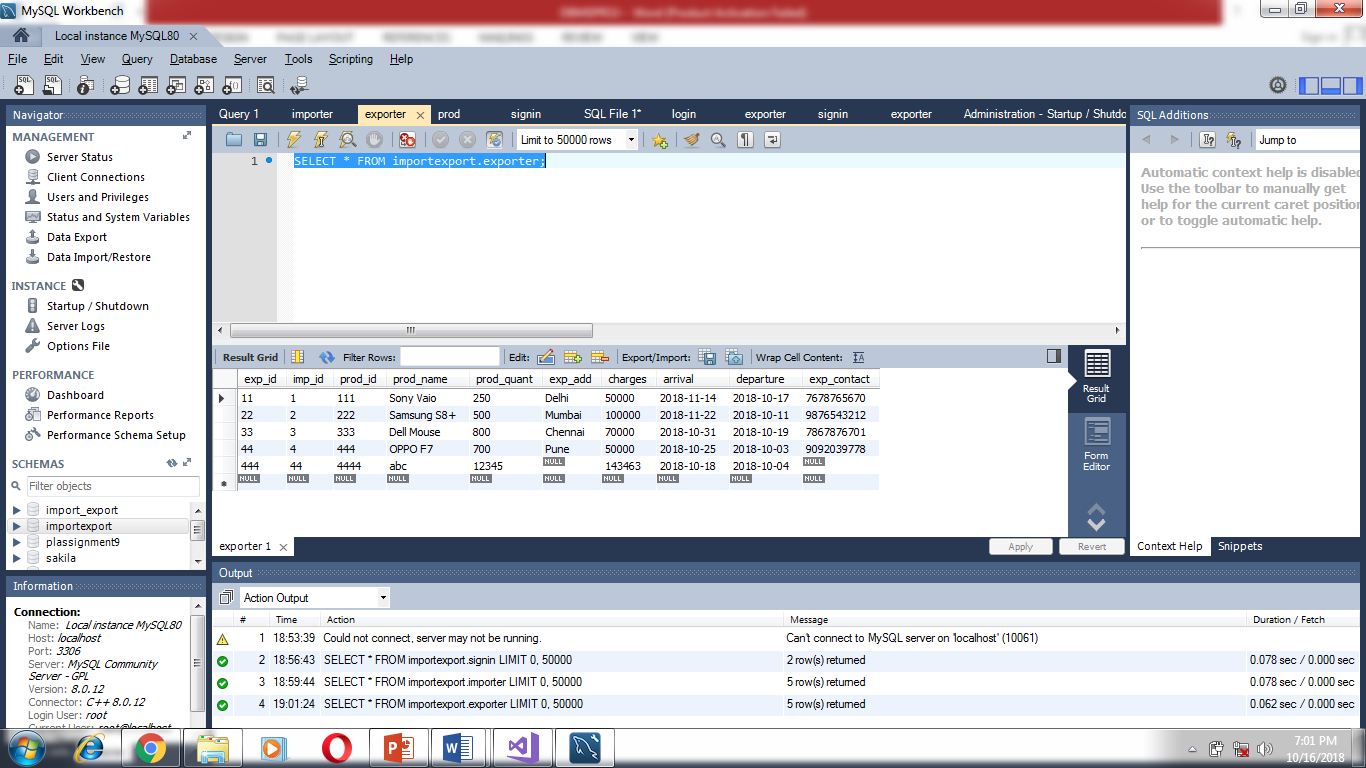


Fig 9.3 MySQL Database3

For Product table:

create table importexport.prod

(

prod\_id int unique,

imp\_id int unique,

exp\_id int unique,

prod\_name varchar(50) not null,

price\_per\_prod int default 0,

primary key(prod\_id)

);

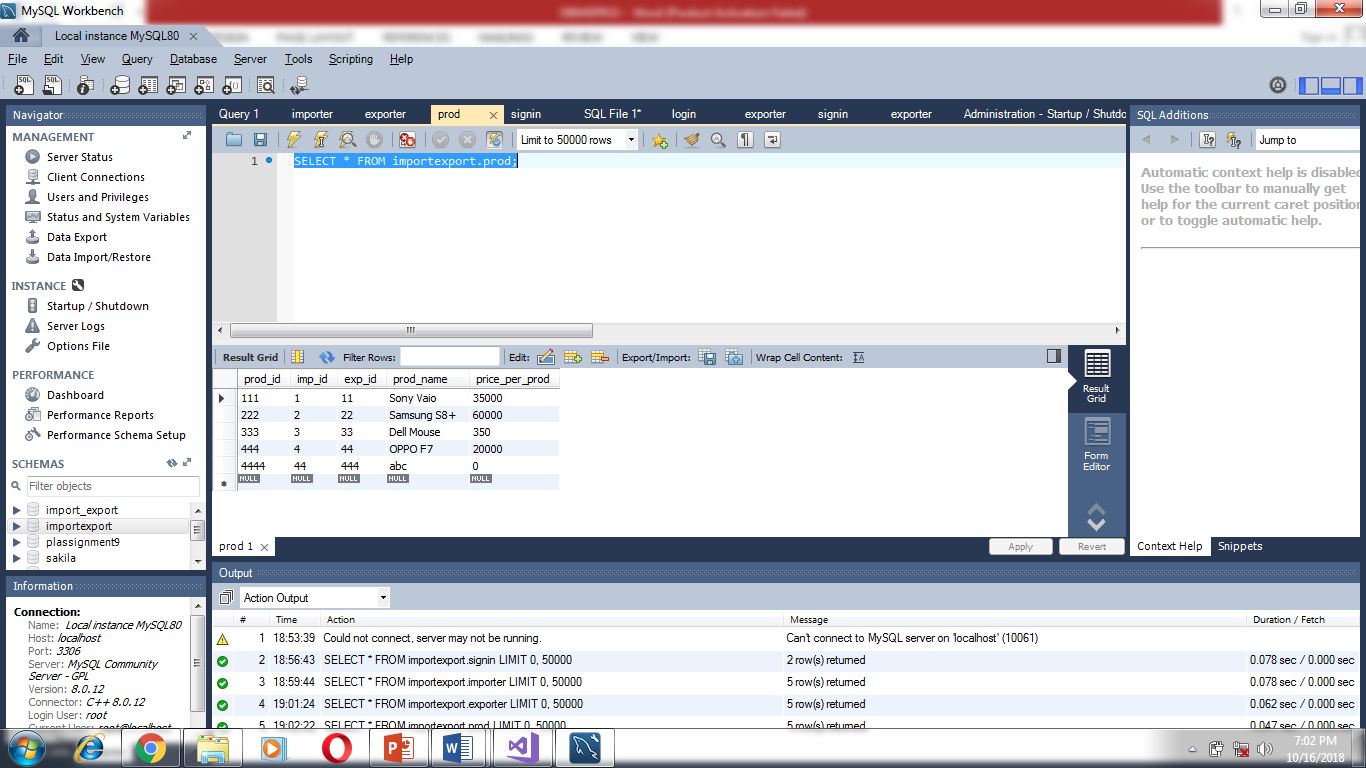


Fig 9.4 MySQL Database4

Connection of Database to VB.Net

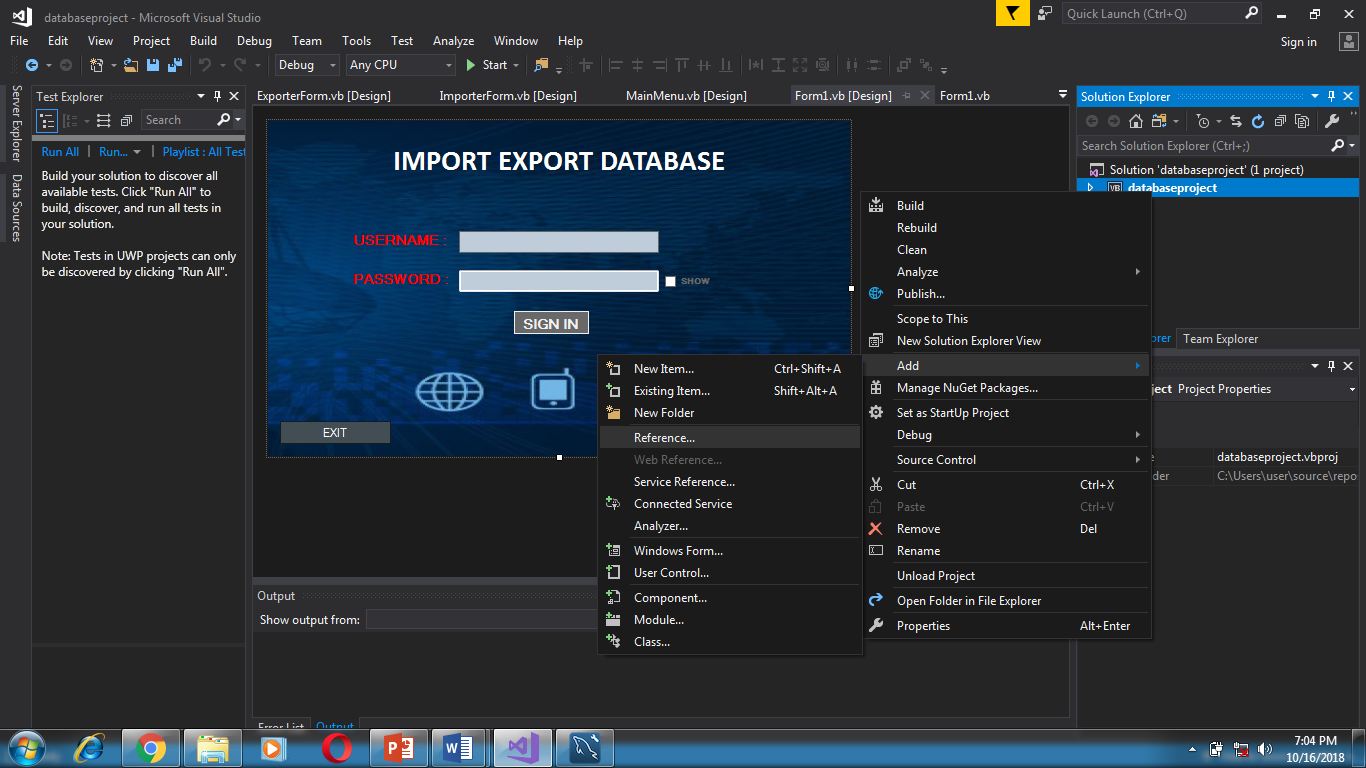


Fig 9.5 VB.Net1

Right Click on your project name in Server Explorer.

Select Add, click on Reference.

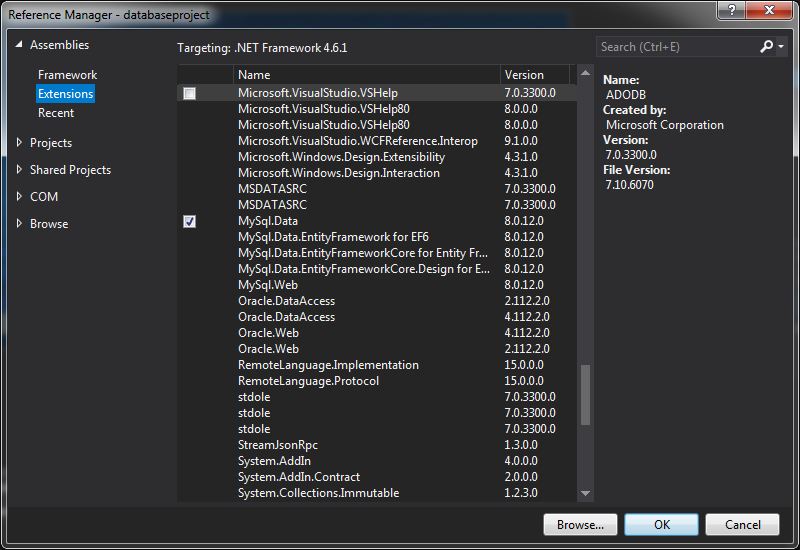


Fig 9.6 VB.Net2

Select Extensions and Check MySql.Data, Click OK.

MySql.Data references will be added to your project.

A basic function to establish a connection of MySQL database with VB.Net is:

MysqlConn = New MySqlConnection

MysqlConn.ConnectionString = "server=localhost;userid=root;password=aamir786;database=importexport"

Try

MysqlConn.Open()

MessageBox.Show("Connection Successfull")

MysqlConn.Close()

Catch ex As MySqlException

MessageBox.Show(ex.Message)

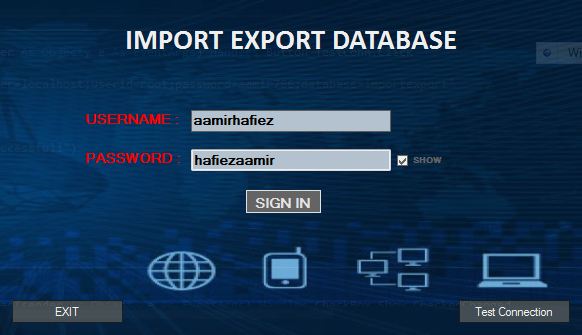
Finally

MysqlConn.Dispose()

End Try

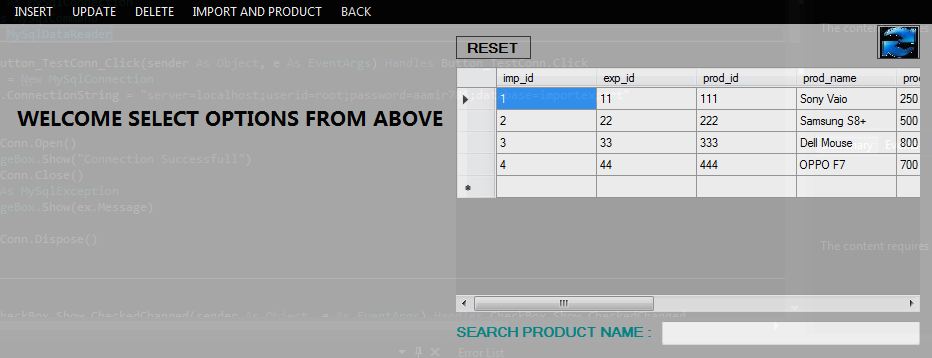
**CHAPTER-X**

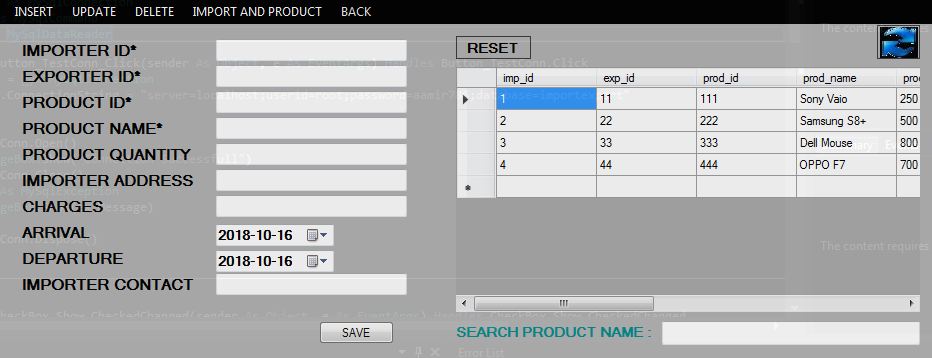
**SYSTEM IMPLEMENTATION**

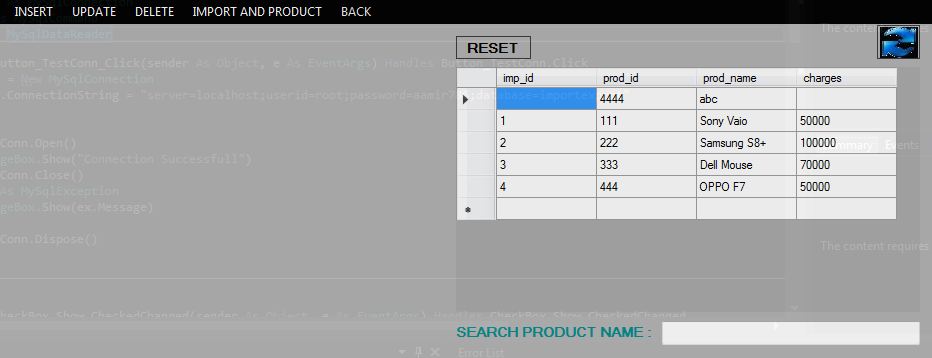


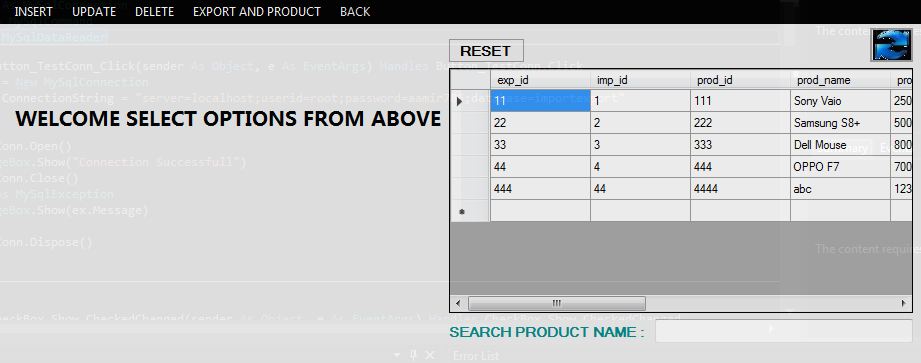


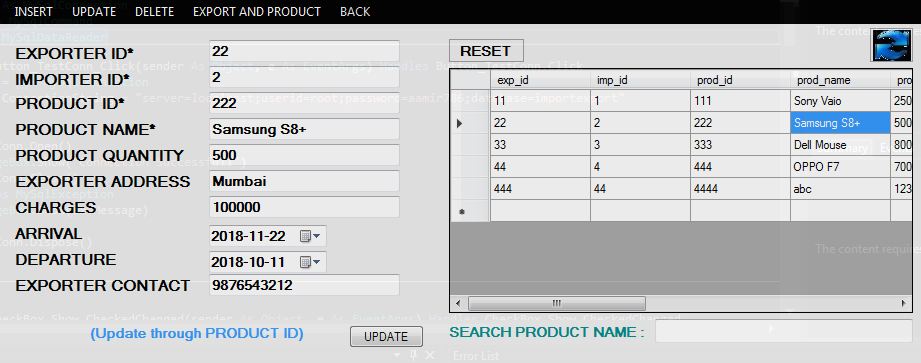


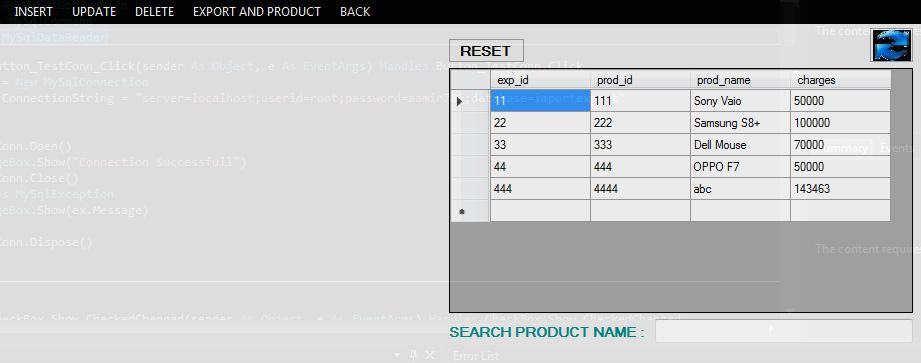


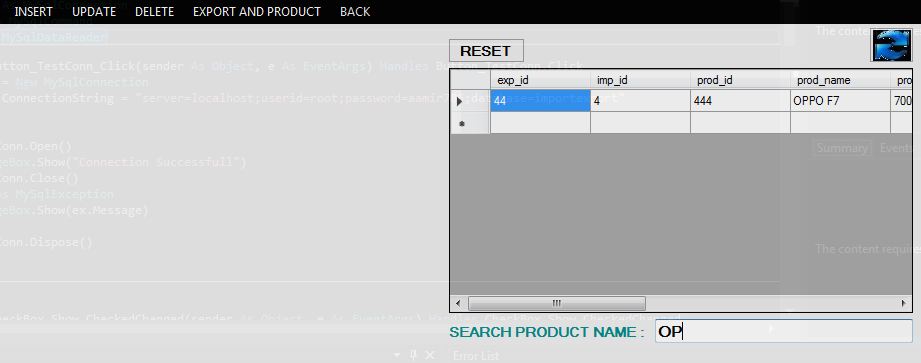


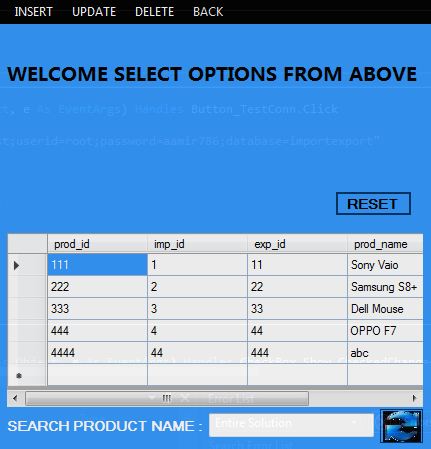


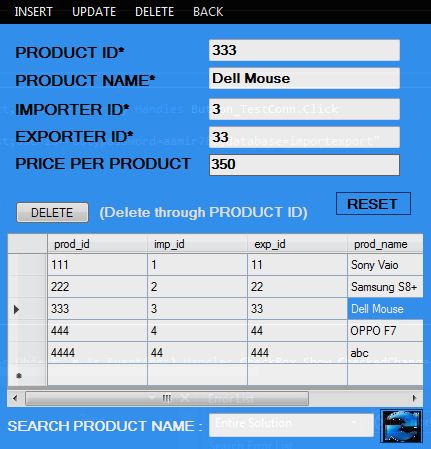


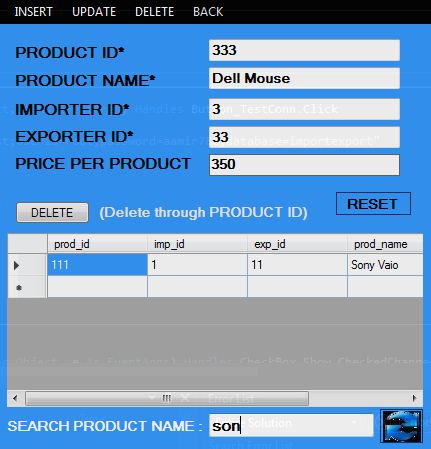


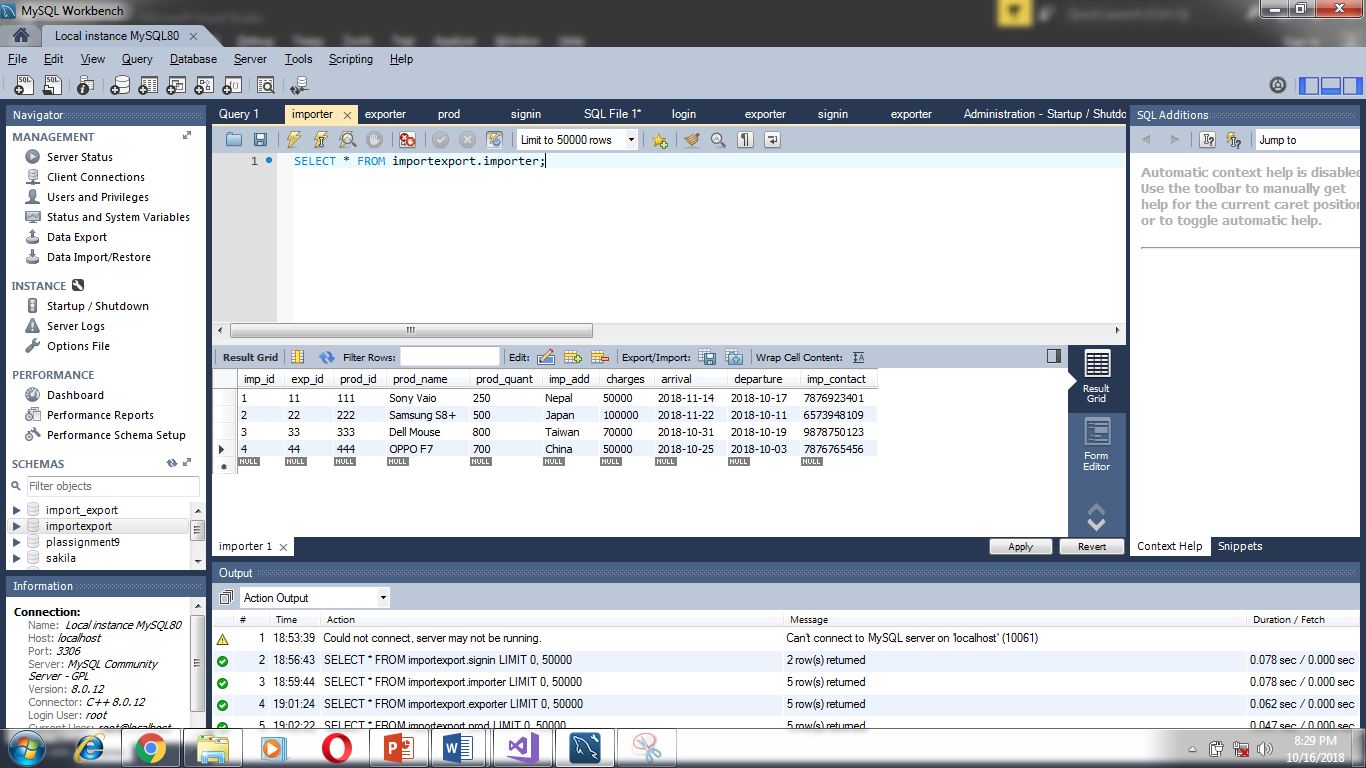


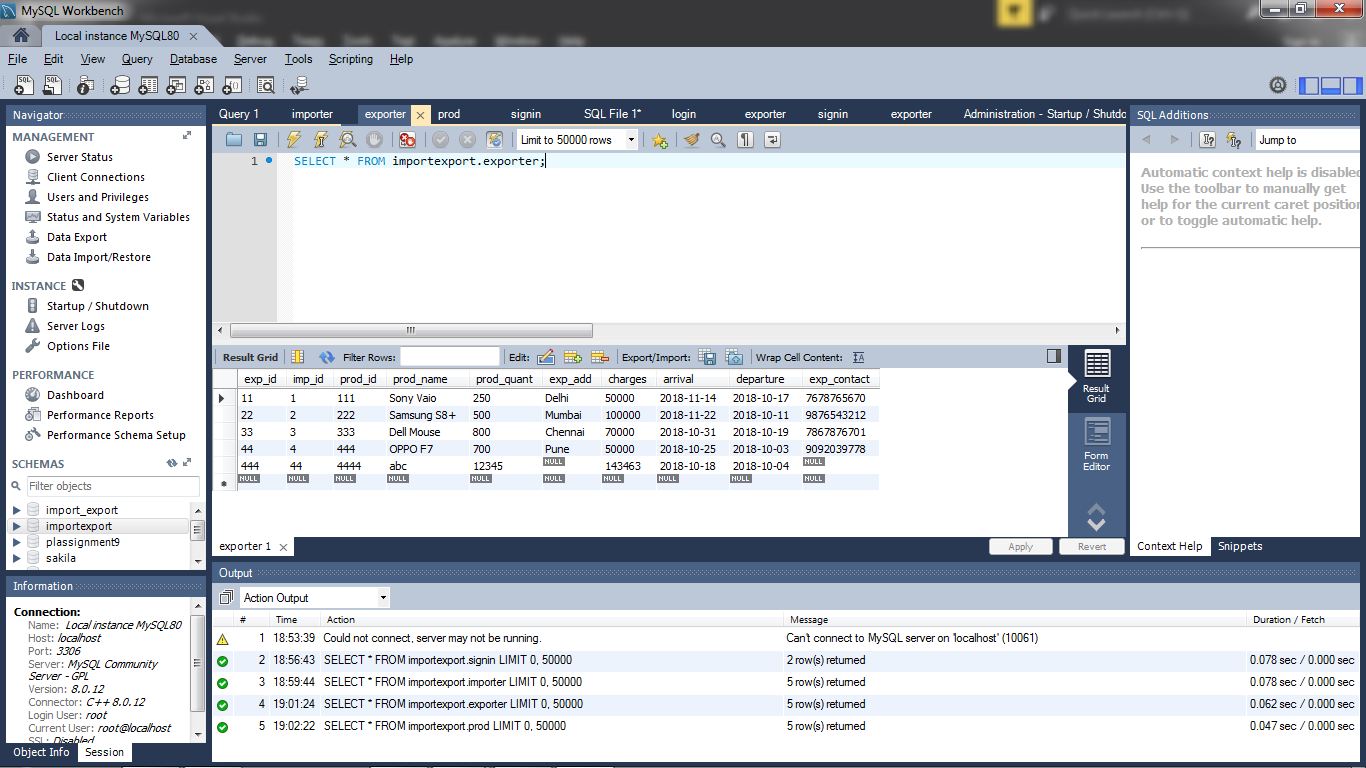


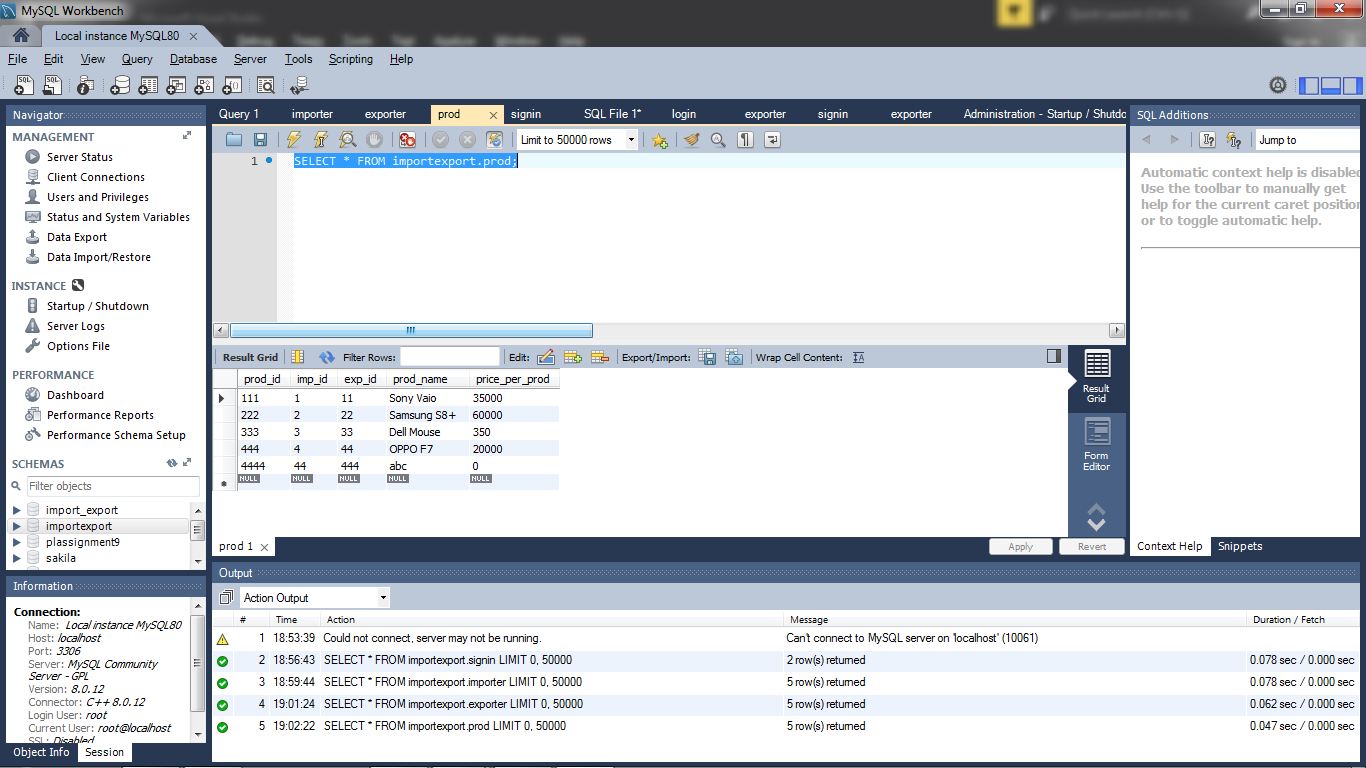


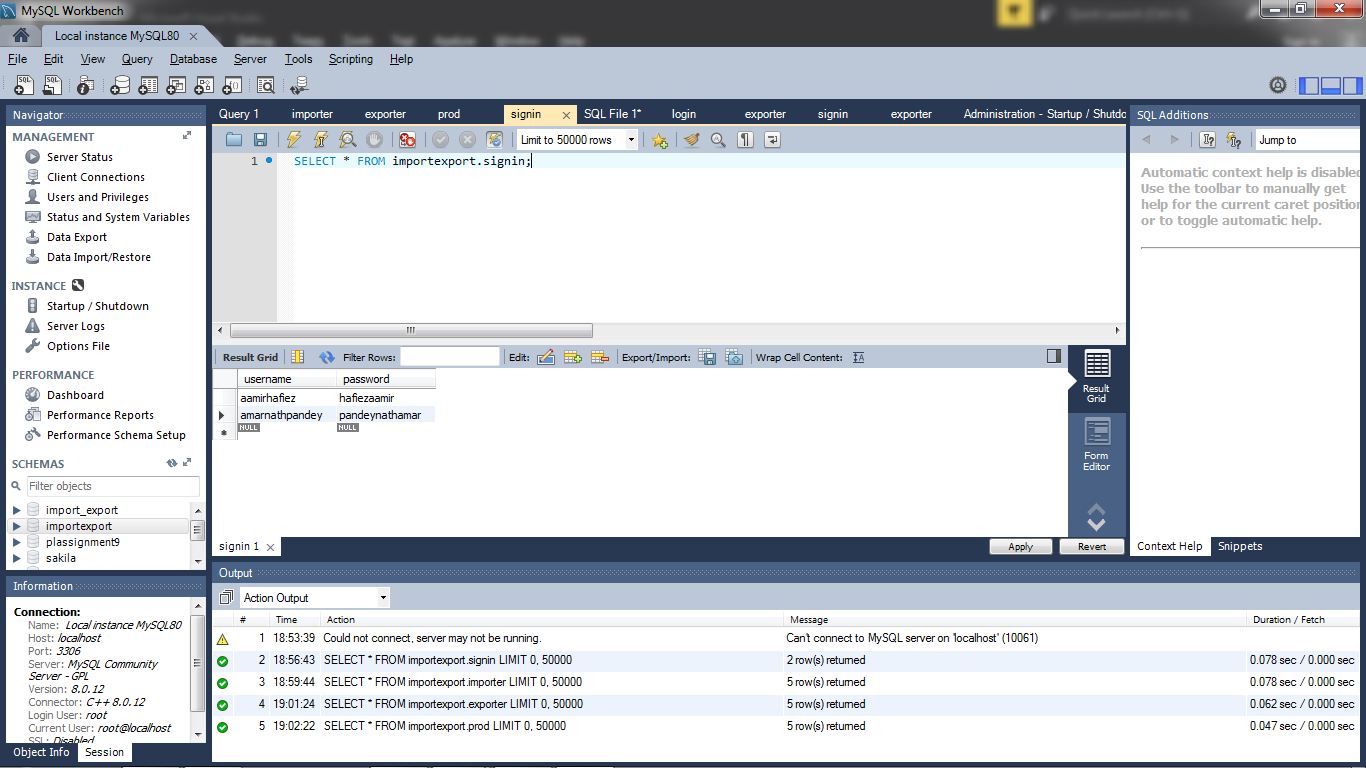












**CHAPTER-XI**

**REFRENCES**

<http://www.indiantradeportal.in/>

<https://www.w3schools.com/sql/>

<https://www.tutorialspoint.com/>

<https://www.wikipedia.org/>

<https://stackoverflow.com/>

https://www.youtube.com/