Workbook

Distributed Computing

(SE – 406)

Name ______________________________

Roll No ______________________________

Batch ______________________________

Discipline ______________________________

Semester / Term ______________________________

Department of Software Engineering

NED University of Engineering & Technology

Distributed Computing

(SE – 406)

Prepared by

Engr. Dr. Muhammad Kashif Mehboob Khan

Approved by

Chairperson

Department of Software Engineering

Department of Software Engineering

NED University of Engineering & Technology

S.NO	DATE	TOPICS	INITIALS
01.		***To Study and Implement Basic Object Oriented Programming Concepts***	
02.		***To Study and Implement DML OPERATIONS USING DATASET.***	
03.		***To Explore the functions of SQL Server utilities, creating stored procedures in SQL server.***	
04.		***To implement linked server using SQL Server and MS Access via GUI and accessing data using Distributed queries.***	
05.		***To implement linked server using SQL Server with Oracle***	
06.		***To Develop a Distributed Data structure of An Online Real Estate.***	
07		***To study and generate mail using SMTP Server.***	
08.		***To Develop a client server programming application using TCP/IP***	
09.		***To simulate the functioning of Lamport's Logical clock***	
10.		***To simulate the functioning of Lamport's Vector Clock Algorithm***	
11.		***To simulate the functioning of Lamport's Algorithm for Mutual Exclusion***	
12.		***To implement Edge chasing distributed deadlock detection algorithm.***	
13.		***SQL Injection Prevention Techniques***	
14.		***Open Ended Lab***	

LIST OF CONTENTS
LAB 1
OBJECT ORIENTED CONCEPTS

OBJECTIVE

To implement a program of insert update and delete operations using object oriented concepts.

THEORY

OBJECT ORIENTED CONCEPTS:

Object-**O**riented **P**rogramming (*OOP*) uses a different set of programming languages than old procedural programming languages (*C*, *Pascal*, etc.). Everything in *OOP* is grouped as self sustainable "*objects*". Hence, you gain re-usability by means of four main object-oriented programming concepts.

OBJECT

An object can be considered a "thing" that can perform a set of related activities. The set of activities that the object performs defines the object's behavior.

In pure OOP terms an object is an instance of a class. There can be any number of objects of a given class in memory at any one time.

CLASS

A class is simply a representation of a type of object. It is a plan that describe the details of an object.

A class is a structure that defines the data and the methods to work on that data.

Example

```
public class Student
{
}
```

ENCAPSULATION

In *OOP* the encapsulation is mainly achieved by creating classes, the classes expose public methods and properties. The class is kind of a container or capsule or a cell, which encapsulate the set of methods, attribute and properties to provide its indented functionalities to other classes. In that sense, encapsulation also allows a class to change its internal implementation without hurting the overall functioning of the system. That idea of encapsulation is to hide how a class does it but to allow requesting what to do.

Example

There are several ways that an encapsulation can be used, as an example we can take the usage of an interface. The interface can be used to hide the information of an implemented class.

```
IStudent myStudent = new LocalStudent();
IStudent myStudent = new ForeignStudent();
```

INHERITANCE:

Ability of a new class to be created, from an existing class by extending it, is called *inheritance*.

Example

```
public class Exception
{
}
public class IOException : Exception
```

```
{}
```

According to the above example the new class (*IOException*), which is called the derived class or subclass, inherits the members of an existing class (*Exception*), which is called the base class or super-class. The class *IOException* can extend the functionality of the class Exception by adding new types and methods and by overriding existing ones.

POLYMORPHISM

Polymorphism is a generic term that means 'many shapes'. More precisely *Polymorphism* means the ability to request that the same operations be performed by a wide range of different types of things.
In *OOP* the *polymorphisms* is achieved by using many different techniques named method overloading, operator overloading and method overriding.

METHOD OVERLOADING

The method overloading is the ability to define several methods all with the same name.

Example

```
public class MyClass
{
   public void MyMethod(int a)
   {
      // Implementation goes here
   }

   public bool MyMethod (int a, string message)
   {
      // Implementation goes here
   }
}
```

OPERATOR OVERLOADING:

The operator overloading (less commonly known as ad-hoc *polymorphisms*) is a specific case of *polymorphisms* in which some or all of operators like +, - or == are treated as polymorphic functions and as such have different behaviors depending on the types of its arguments.

Example:

BINARY OPERATORS:

```
+
-
*
/
%
&
|
^
<<
>>
==
!=
>
<
>=
<=
```

METHOD OVERRIDING

Providing a declaration which matches another declaration of the same name, thereby hiding the existing declaration.

In terms of object-oriented programming, overriding is the ability of a subclass to "override" and replace the functionality of a method.

Example:

```
class A {
f(){
print "A"
}
}
class B extends A {
// Function f is overridden.
// When B.f() is called, it will call this function instead of A.f()
f() {
print "B"
}
}
```

EXERCISE:

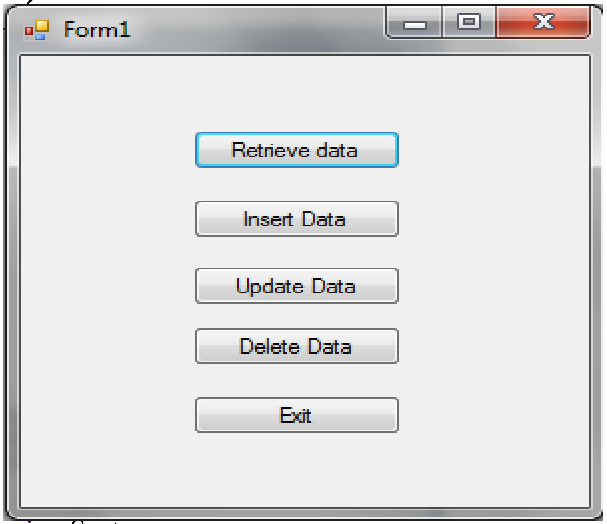
Write a program to insert update and delete a record

STEPS

1-) CONNECTION CLASS CODING

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace WindowsFormsApplication2
{
    class Class1
    {
    SqlConnection conObj = new SqlConnection("Data Source=NEDUET-7014AD82\\SQLEXPRESS;Initial
Catalog=Manipulate;Integrated Security=True");
        SqlCommand cmdObj;
        SqlDataAdapter adpObj;
        DataTable dt = new DataTable();

        public DataTable selectData(string sql)
        {
            conObj.Open();
           cmdObj = new SqlCommand(sql, conObj);
           adpObj = new SqlDataAdapter(sql,conObj);
           adpObj.Fill(dt);
           conObj.Close();
           return dt;
        }
        public bool manipulateData(string sql)
        {
           conObj.Open();
           try
           {  cmdObj = new SqlCommand(sql, conObj);
              cmdObj.ExecuteNonQuery();}
           catch
           {
              conObj.Close();
              return false;
           }
           conObj.Close();
           return true;
        }
       public int scalar(string sql)
        {
           conObj.Open();
           cmdObj = new SqlCommand(sql, conObj);
           int i = Convert.ToInt16(cmdObj.ExecuteScalar());
           conObj.Close();
           return i;
       }}}
```

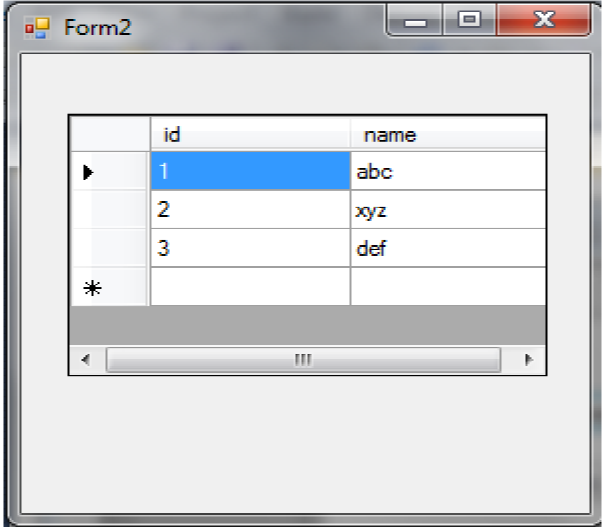
2-) MENU FORM CODING



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication2
{
```

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        Retrieve rt = new Retrieve();
        rt.Show();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        Insert it = new Insert();
        it.Show();
    }
    private void button3_Click(object sender, EventArgs e)
    {
        Update up = new Update();
        up.Show();
    }
    private void button4_Click(object sender, EventArgs e)
    {
        Delete dl = new Delete();
        dl.Show();
    }
    private void button5_Click(object sender, EventArgs e)
    {
        this.Close();
    }}}
```

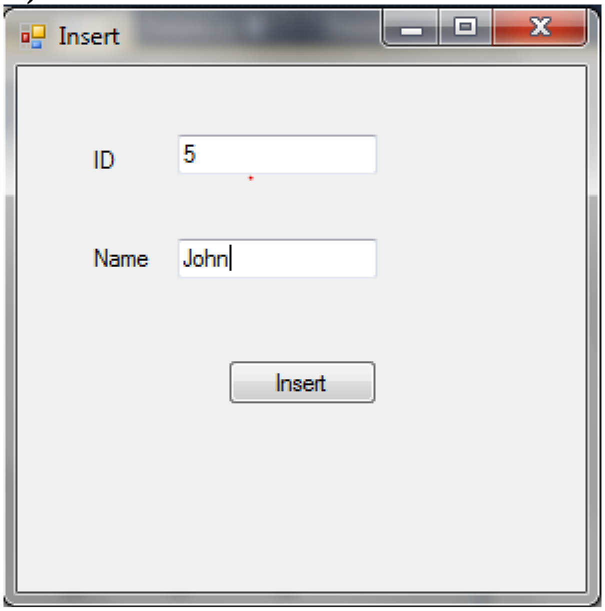
3-) RETRIEVE FORM CODING



id	name
1	abc
2	xyz
3	def

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication2
{
   public partial class Retrieve : Form
   {
      DataTable dt = new DataTable();
      public Retrieve()
      {
         InitializeComponent();
      }
      private void Retrieve_Load(object sender, EventArgs e)
      {
         this.login_tabTableAdapter.Fill(this.loginDataSet.login_tab);
         string sql = "select * from login_tab";
         Class1 db = new Class1();
         dt = db.selectData(sql);
         dataGridView1.DataSource = dt;
      }}}
```

3-) INSERT FORM CODING

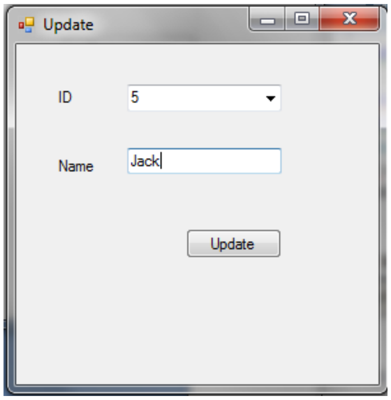


ID 5

Name John

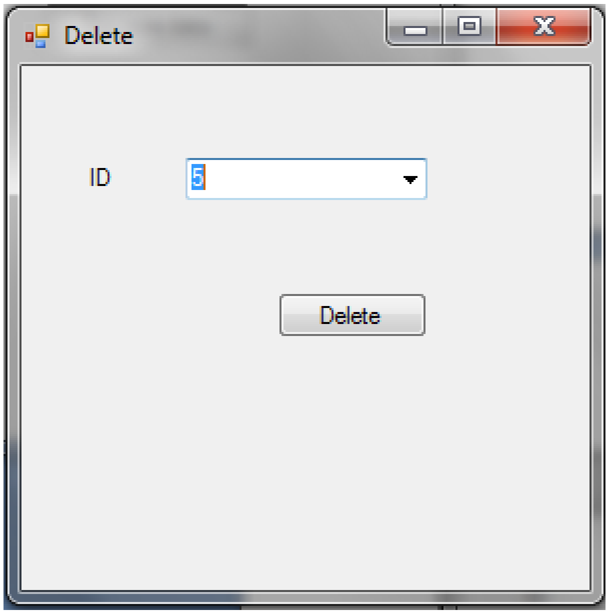
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication2
{
    public partial class Insert : Form
    {
        public Insert()
        {
            InitializeComponent();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string sql = "insert into login_tab values('" + textBox1.Text + "','" + textBox2.Text + "')";

            Class1 db = new Class1();

            bool verify = db.manipulateData(sql);
            if (verify == false)
            {
                MessageBox.Show("Some ERROR occured!");
            }
            else
            {
                MessageBox.Show("Record has sucessfully added!");
            }
            this.Close();
        }}}
```

4-) UPDATE FORM CODING



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication2
{
    public partial class Update : Form
    {
        public Update()
        {
            InitializeComponent();
        }
        private void button5_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void Update_Load(object sender, EventArgs e)
        {
            string sql = "select * from login_tab";
            Class1 db = new Class1();
            DataTable dt = new DataTable();
            dt = db.selectData(sql);
            comboBox1.DataSource = dt;
            comboBox1.DisplayMember = "ID";
            textBox1.DataBindings.Add("Text", dt, "Name");
        }
        private void button1_Click(object sender, EventArgs e)
        {
            string sql = "update login_tab SET Name = '" + textBox1.Text + "' where ID = " + 
Convert.ToInt16(comboBox1.Text.ToString()) + "";
            Class1 db = new Class1();
            bool verify = db.manipulateData(sql);
            if (verify == false)
            {
                MessageBox.Show("Some ERROR occured!");
            }
            else
            {
                MessageBox.Show("Record has sucessfully updated!");
            }
            this.Close();
}
}
}
```

5-) DELETE FORM CODING



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication2
{
    public partial class Delete : Form
    {
        public Delete()
        {
            InitializeComponent();
        }
        private void button5_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void Delete_Load(object sender, EventArgs e)
        {
            string sql = "select * from login_tab";
            Class1 db = new Class1();
            DataTable dt = new DataTable();
            dt = db.selectData(sql);
            comboBox1.DataSource = dt;
            comboBox1.DisplayMember = "ID";
        }
        private void button1_Click(object sender, EventArgs e)
        {
```

```
        string sql = "delete from login_tab where ID = " + Convert.ToInt16(comboBox1.Text.ToString()) +
"";
         Class1 db = new Class1();
           bool verify = db.manipulateData(sql);
           if (verify == false)
           {
               MessageBox.Show("Some ERROR occured!");
           }
           else
           {
               MessageBox.Show("Record has sucessfully updated!");
           }
           this.Close();

        }
    }
}
```

LAB 2
DML OPERATIONS USING DATASETS

OBJECTIVE

To implement DML operations using dataset auto generated methods

THEORY

DATA MANIPUATION LANGUAGE

The Data Manipulation Language (DML) is used to retrieve, insert and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic DML commands:

INSERT

The INSERT command in SQL is used to add records to an existing table.

Example

```
INSERT INTO personal_info
values('bart','simpson',12345,$45000)
```

SELECT

The SELECT command is the most commonly used command in SQL. It allows database users to retrieve the specific information they desire from an operational database.

Example

```
SELECT *
FROM personal_info
```

UPDATE

The UPDATE command can be used to modify information contained within a table, either in bulk or individually.

Example

```
UPDATE personal_info
SET salary = salary * 1.03
```

DELETE

The DELETE command with a WHERE clause can be used to remove his record from the personal_info table.

Example

```
DELETE FROM personal_info
WHERE employee_id = 12345
```

DATASET

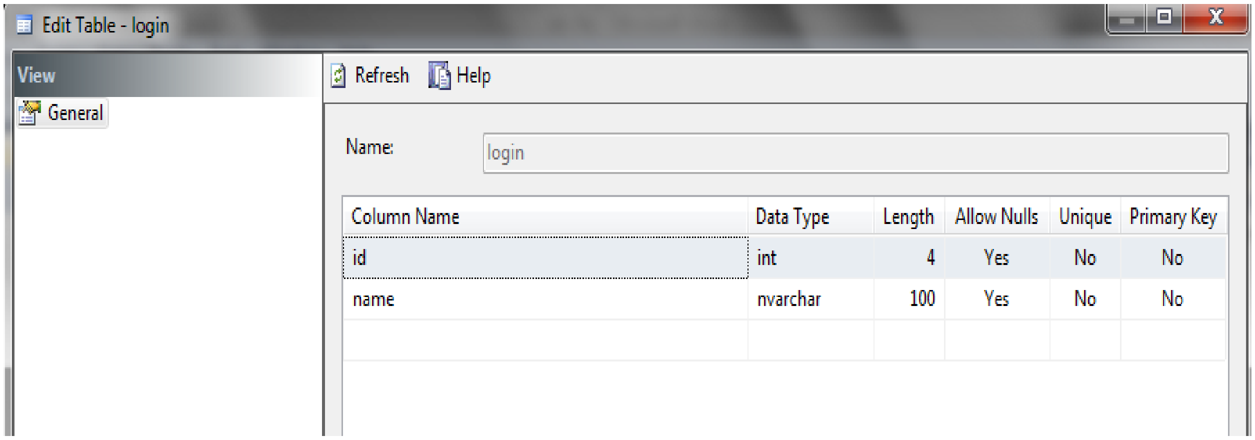
The DataSet contains the copy of the data we requested through the SQL statement. We can use Dataset in combination with SqlDataAdapter class . The SqlDataAdapter object allows us to populate Data Tables in a DataSet. We can use Fill method in the SqlDataAdapter for populating data in a Dataset.

EXERCISE

Using dataset auto generated methods implement DML operations.

TABLES

- Create a table
- Right Click **data connection** → Add Connection → **MS SQL Server compact edition 3.5 (select)**
- Enter **Database Name** → Create database→ Database created with **.sdf extension**

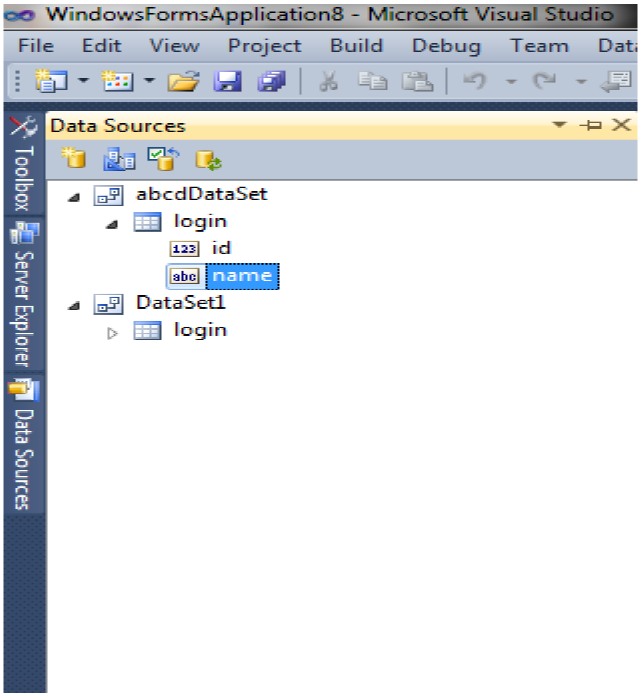


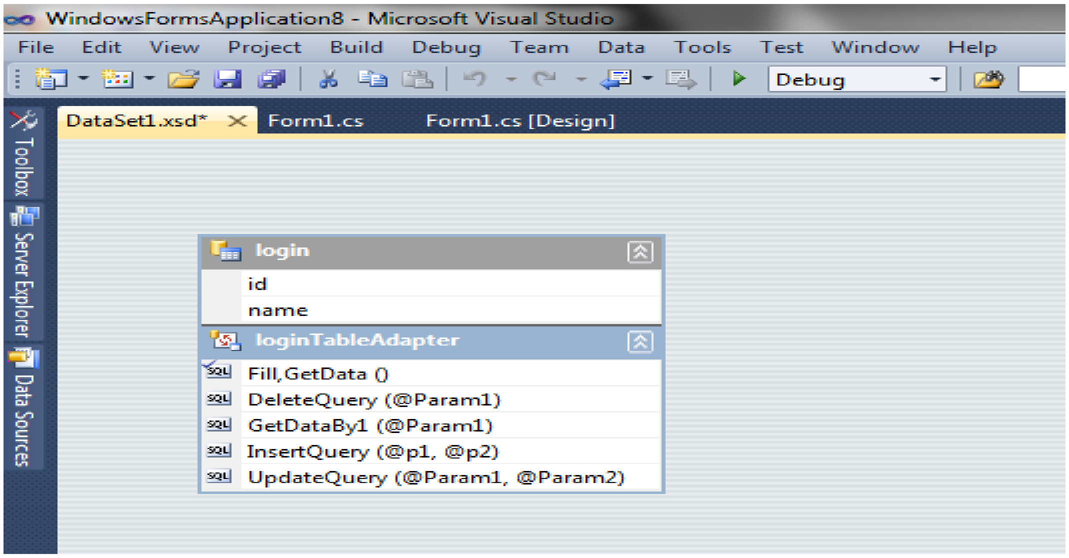
Name: login

Column Name	Data Type	Length	Allow Nulls	Unique	Primary Key
id	int	4	Yes	No	No
name	nvarchar	100	Yes	No	No

DATASET

- Solution Explorer → Windows Form→ Right Click **Add Datasets** → Drag and Drop table
- Right click **table Adapter** → Add query → Run the **Wizard of query generation**

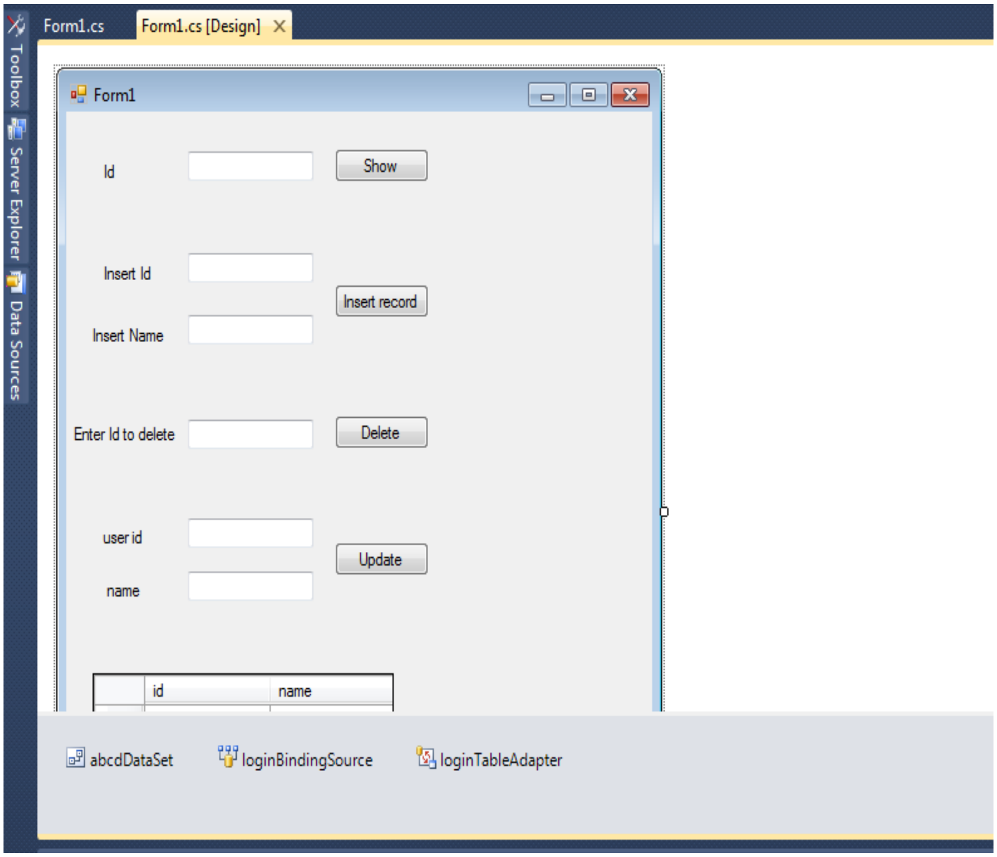



WindowsFormsApplication8 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Test Window Help
Debug
Toolbox
Server Explorer
Data Sources
DataSet1.xsd*
Form1.cs
Form1.cs [Design]

login
id
name

loginTableAdapter
Fill,GetData ()
DeleteQuery (@Param1)
GetDataBy1 (@Param1)
InsertQuery (@p1, @p2)
UpdateQuery (@Param1, @Param2)

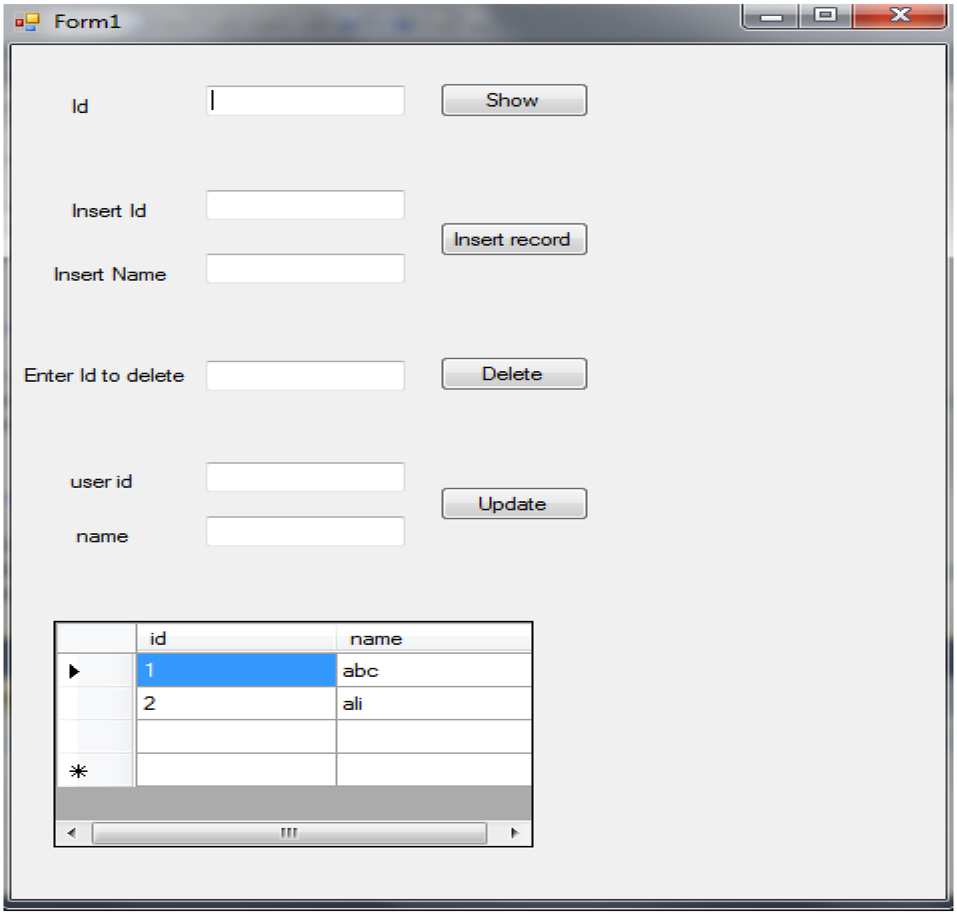
FORM DESIGN:



CODING:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using WindowsFormsApplication8.DataSet1TableAdapters;


namespace WindowsFormsApplication8
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int a = int.Parse(textBox1.Text);
            loginTableAdapter d = new loginTableAdapter();
            dataGridView1.DataSource = d.GetDataBy(a);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            int b= int.Parse(textBox2.Text);
            DataSet1TableAdapters.loginTableAdapter ob = new
DataSet1TableAdapters.loginTableAdapter();
            ob.InsertQuery(b, textBox3.Text);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            int c = int.Parse(textBox4.Text);
            DataSet1TableAdapters.loginTableAdapter ab = new
DataSet1TableAdapters.loginTableAdapter();
            ab.DeleteQuery(c);
        }

        private void button4_Click(object sender, EventArgs e)
        {
            int d = int.Parse(textBox5.Text);
            DataSet1TableAdapters.loginTableAdapter n= new DataSet1TableAdapters.loginTableAdapter();
            n.UpdateQuery(textBox6.Text, d);

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.loginTableAdapter.Fill(this.abcdDataSet.login);

        }
    }}
```

OUTPUT



	id	name
▶	1	abc
	2	ali
✳		

LAB 3
STORED PROCEDURES

OBJECTIVE

To explore functions of stored procedures in Sql server.

THEORY

STORED PROCEDURES

Introduction

- Stored procedures are special objects available in sql server. It is a precompiled statements where all the preliminary parsing operations are performed and the statements are ready for execution.
- It is very fast when compared to ordinary sql statements where the sql statements will undergone a sequence of steps to fetch the data.
- Stored procedure involves various syntax based on the parameters passed.

Syntax

Code:

```
CREATE PROCEDURE procName
AS
BEGIN
-----Query goes here
END
```

To execute the stored procedure we have to use exec command,

Code:

```
EXEC procName
```

Applications do not need to transmit all of the SQL statements in the procedure: they have to transmit only an EXECUTE or CALL statement containing the name of the procedure and the values of the parameters.

BACKUP

Every recovery model lets you back up a whole or partial SQL Server database or individual files or filegroups of the database. Table-level backups cannot be created.

DATABASE BACKUPS

Database backups are easy to use and are recommended whenever database size allows. SQL Server supports the following types of database backups.

Backup type	Description
Database backup	A full backup of the whole database. Database backups represent the whole database at the time the backup finished.
Differential database backups	A backup of all files in the database. This backup contains only the data extents that were modified since the most recent database backup of each file.

SCRIPTS

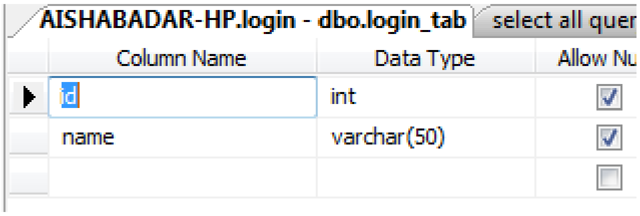
The Database Scripts project is a series of command line scripts which will dump, erase, restore and merge databases. They are specifically set up to work the best when developing within a version control environment. The primary goals are to:

- keep the database **in sync with code**
- preserve ability to **use the web GUI**
- enable a **rapid** development workflow
- **Merge** development and production databases

EXERCISE

Explore functions of stored procedures and take backups and generate scripts of database.

TABLE CREATION



Column Name	Data Type	Allow Nu
id	int	☑
name	varchar(50)	☑
		□

CREATING STORED PROCEDURE

Expand Programmability in object explorer.
Programmability → Right Click on Stored Procedure → New Stored procedure

SELECT ALL QUERY STORED PROCEDURE

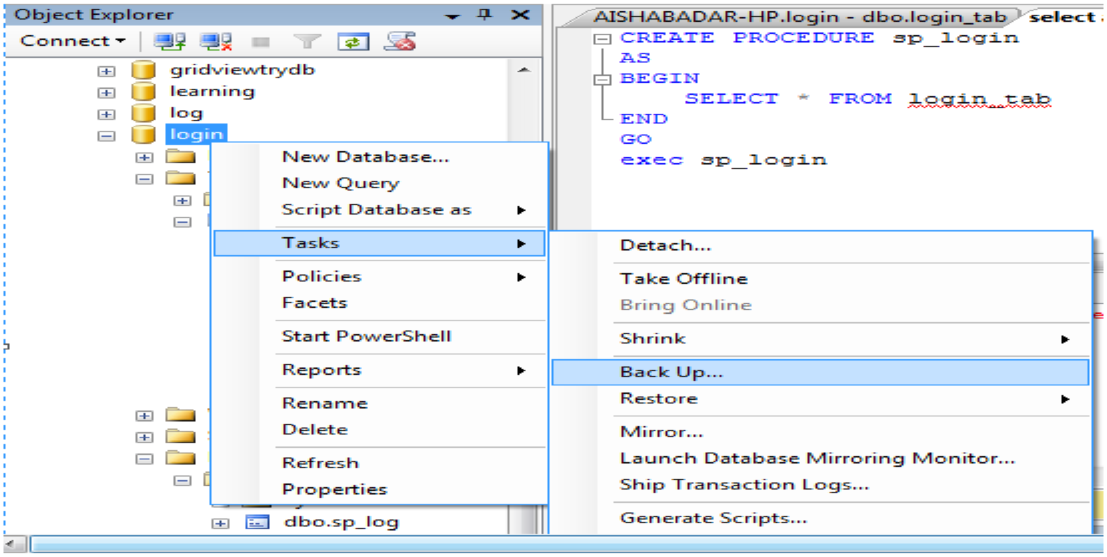
```
CREATE PROCEDURE sp_login
AS
BEGIN
        SELECT * FROM login_tab
END
GO
exec sp_login
```

PARAMETERIZED QUERY

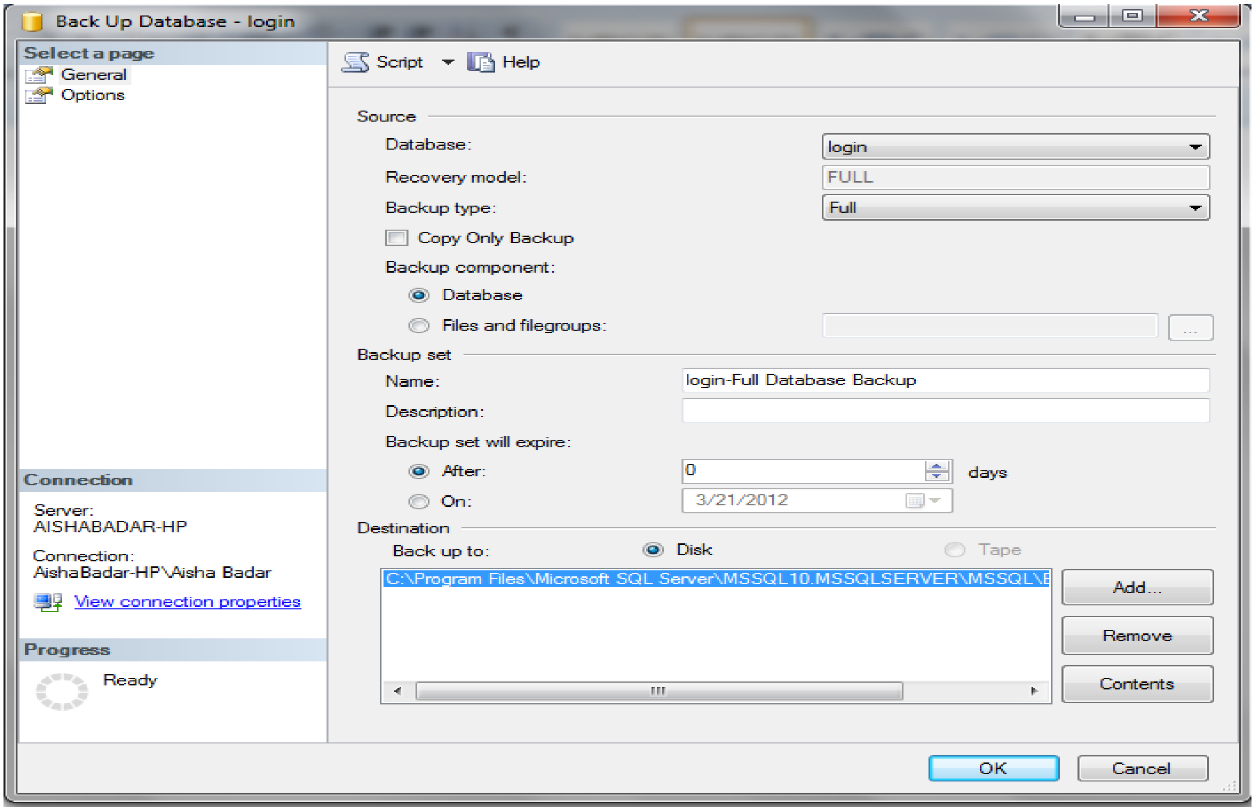
```
CREATE PROCEDURE sp_log
@id int
AS
BEGIN
        SELECT name from login_tab
        where id=@id
END
GO
exec sp_log 1
```

CREATING BACKUP

Right Click on the data base→ Tasks → Backup



```
CREATE PROCEDURE sp_login
AS
BEGIN
        SELECT * FROM login_tab
END
GO
exec sp_login
```



Server:
AISHABADAR-HP

Connection:
AishaBadar-HP\Aisha Badar

Ready

Source

Database: login

Recovery model: FULL

Backup type: Full

□ Copy Only Backup

Backup component:

● Database

○ Files and filegroups:

Backup set

Name: login-Full Database Backup

Description:

Backup set will expire:

● After: 0 days

○ On: 3/21/2012

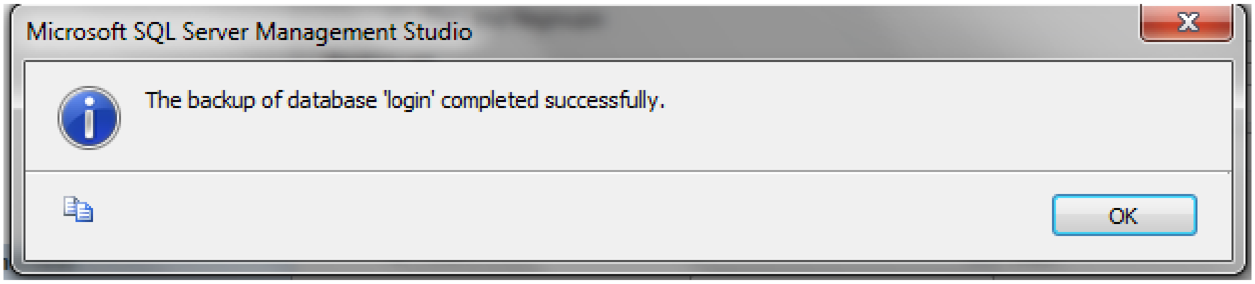
Destination

Back up to: ● Disk ○ Tape

C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\E

Press Ok

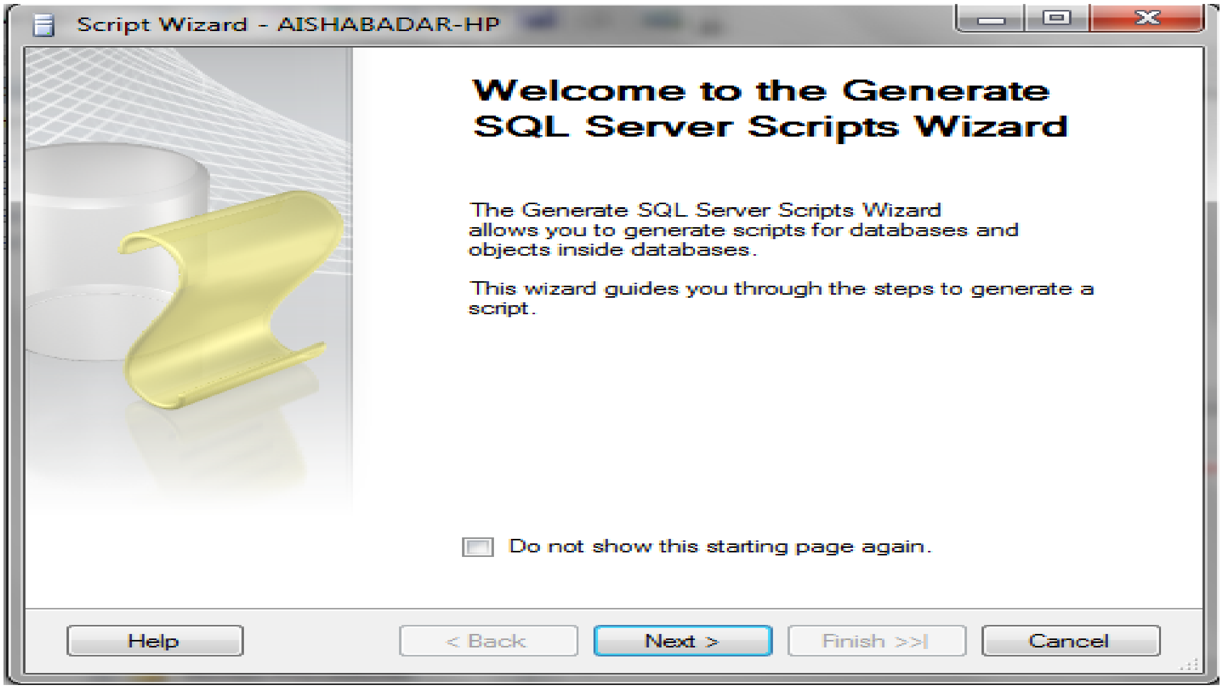
This message will appear on the screen


Microsoft SQL Server Management Studio

The backup of database 'login' completed successfully.

CREATING SCRIPTS

- Right Click on the **data base**→ **Tasks** → **Generate Scripts**


Script Wizard - AISHABADAR-HP

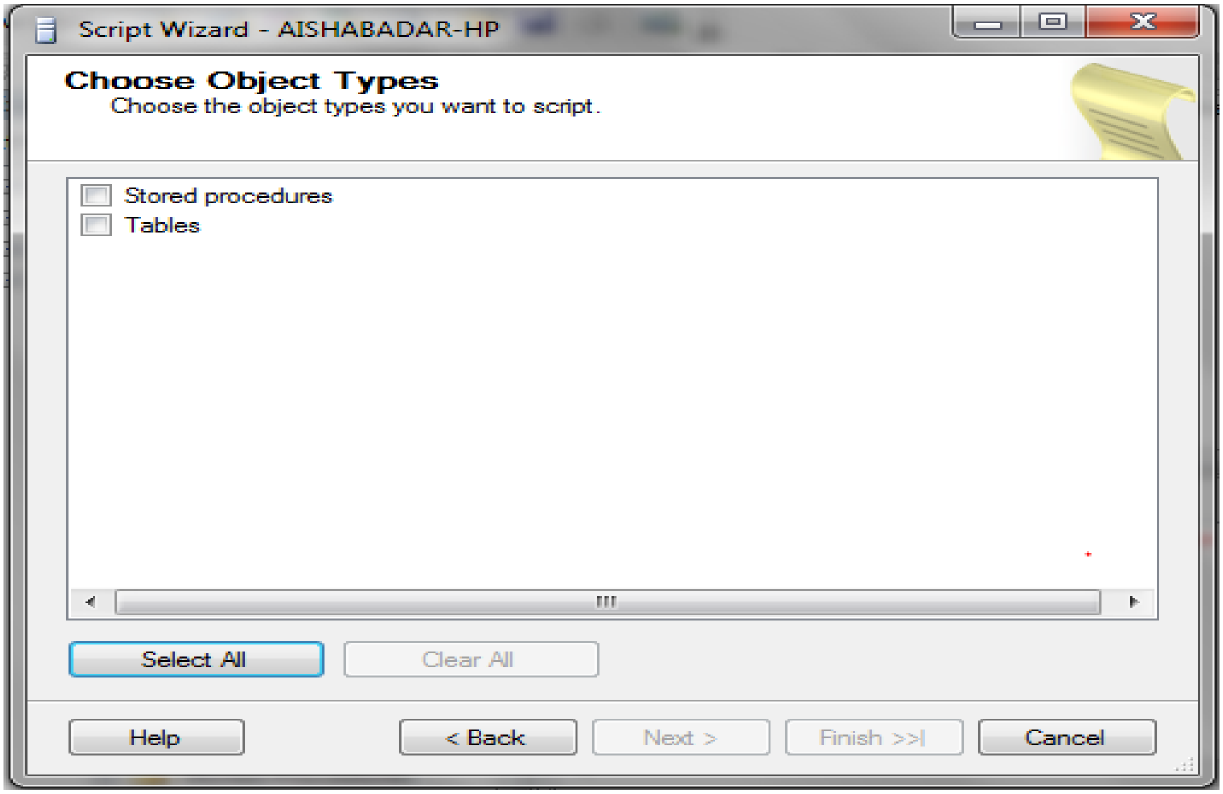
Welcome to the Generate SQL Server Scripts Wizard

The Generate SQL Server Scripts Wizard allows you to generate scripts for databases and objects inside databases.

This wizard guides you through the steps to generate a script.

□ Do not show this starting page again.

- Click Next→ Select your Database Name → Click **Next**
- Select Stored Procedures and Tables if wants to generate their scripts



Script Wizard - AISHABADAR-HP

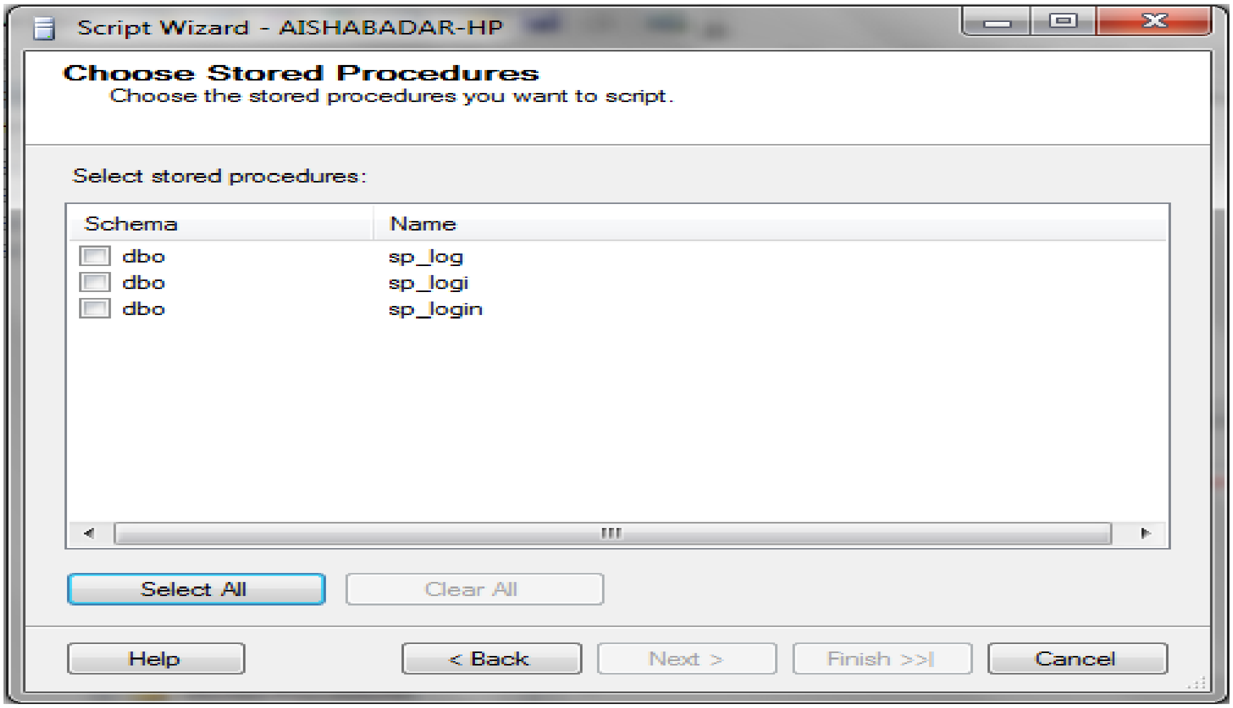
Choose Object Types

Choose the object types you want to script.

□ Stored procedures

□ Tables

- Select those Stored Procedures whose scripts are required.



Script Wizard - AISHABADAR-HP

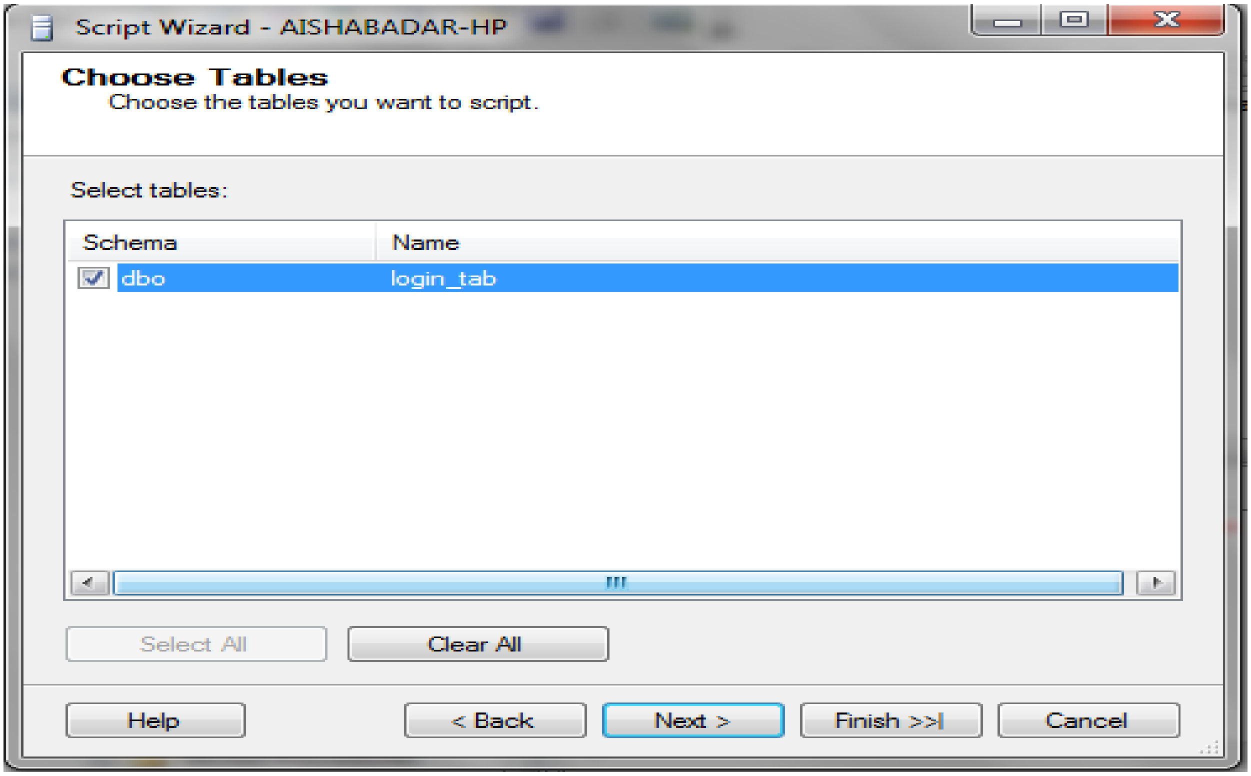
Choose Stored Procedures

Choose the stored procedures you want to script.

Select stored procedures:

Schema	Name
□ dbo	sp_log
□ dbo	sp_logi
□ dbo	sp_login

- Select those tables in the database whose scripts are required.



Script Wizard - AISHABADAR-HP

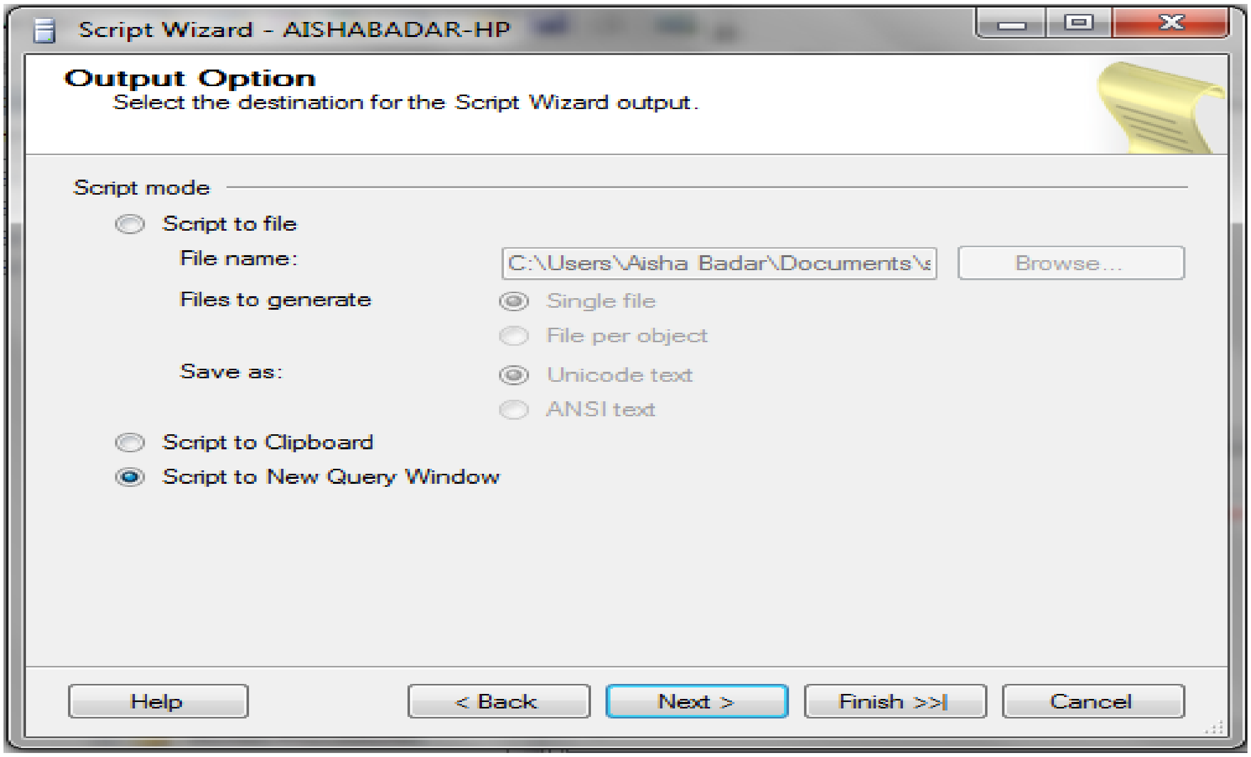
Choose Tables

Choose the tables you want to script.

Select tables:

Schema	Name
☑ dbo	login_tab

- Select the destination of the script



Script Wizard - AISHABADAR-HP

Output Option

Select the destination for the Script Wizard output.

Script mode

○ Script to file

File name: C:\Users\Aisha Badar\Documents\

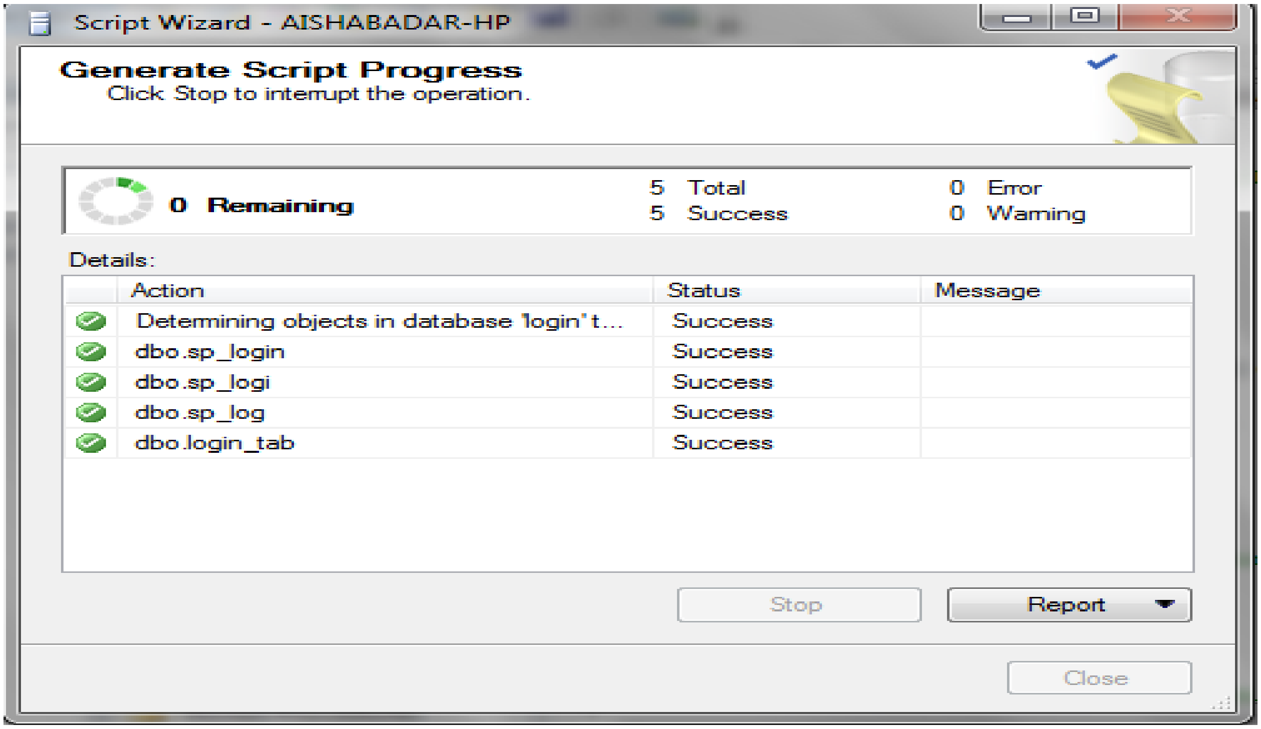
Files to generate: ● Single file ○ File per object

Save as: ● Unicode text ○ ANSI text

○ Script to Clipboard

● Script to New Query Window

- Click Finish



Generate Script Progress

Click Stop to interrupt the operation.

0 Remaining

5	Total	0	Error
5	Success	0	Warning

Details:

	Action	Status	Message
✓	Determining objects in database 'login' t...	Success	
✓	dbo.sp_login	Success	
✓	dbo.sp_logi	Success	
✓	dbo.sp_log	Success	
✓	dbo.login_tab	Success	

SCRIPT

```
USE [login]
GO
/****** Object:  Table [dbo].[login_tab]    Script Date: 03/21/2012 23:04:31 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[login_tab](
	[id] [int] NULL,
	[name] [varchar](50) NULL
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  StoredProcedure [dbo].[sp_login]    Script Date: 03/21/2012 23:04:30 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- Batch submitted through debugger: SQLQuery2.sql|0|0|C:\Users\Aisha
Badar\AppData\Local\Temp\~vsC8DD.sql
CREATE PROCEDURE [dbo].[sp_login]
```

```
AS
BEGIN
	SELECT * FROM login_tab
END
GO
/****** Object:  StoredProcedure [dbo].[sp_logi]    Script Date: 03/21/2012 23:04:30 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[sp_logi]
@id int
AS
BEGIN
	SELECT name from login_tab
	where id=@id
END
GO
/****** Object:  StoredProcedure [dbo].[sp_log]    Script Date: 03/21/2012 23:04:30 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[sp_log]
@id int
AS
BEGIN
	SELECT name from login_tab
	where id=@id
END
GO
```

LAB 4
INTRODUCTION AND IMPLEMENTATION OF LINKED SERVER

OBJECTIVE

To implement linked server using SQL Server and MS Access

THEORY

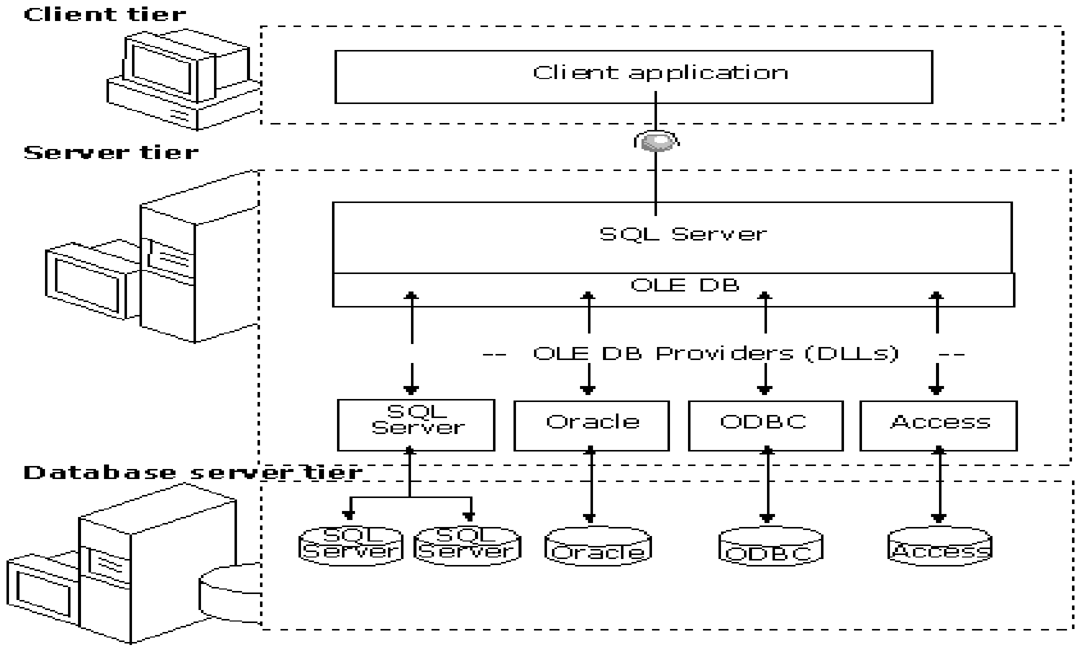
LINKED SERVER:

A linked server configuration enables SQL Server to execute commands against OLE DB data sources on remote servers. Linked servers offer the following advantages:

- Remote server access.
- The ability to issue distributed queries, updates, commands, and transactions on heterogeneous data sources across the enterprise.
- The ability to address diverse data sources similarly.

LINKED SERVER COMPONENTS

A linked server definition specifies the following objects:
An OLE DB provider
An OLE DB data source
An ***OLE DB provider*** is a DLL that manages and interacts with a specific data source.
An ***OLE DB data source*** identifies the specific database that can be accessed through OLE DB. Although data sources queried through linked server definitions are ordinarily databases, OLE DB providers exist for a variety of files and file formats. These include text files, spreadsheet data, and the results of full-text content searches.



Typically linked servers are used to handle distributed queried when a client application executes a distributed query through a linked server, sql server passes the command and sends requests to OLEDB.

EXERCISE

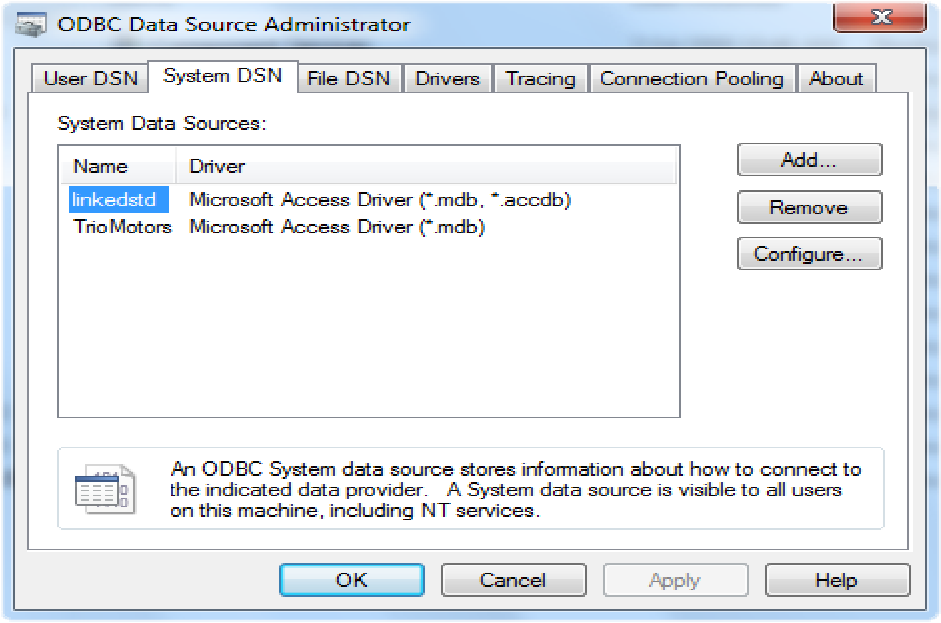
Implement Linked server using Sql Server Management Studio and Stored Procedure

USING SQL SERVER MANAGEMENT STUDIO

For data source:

Follow these steps.

- First Create a simple **Database in MS access**, add a table and save it in any drive.
- Go to **Control Pannel**, click **system security**, then click **admin tools**.
- Select **data source**, then **system DNS**,

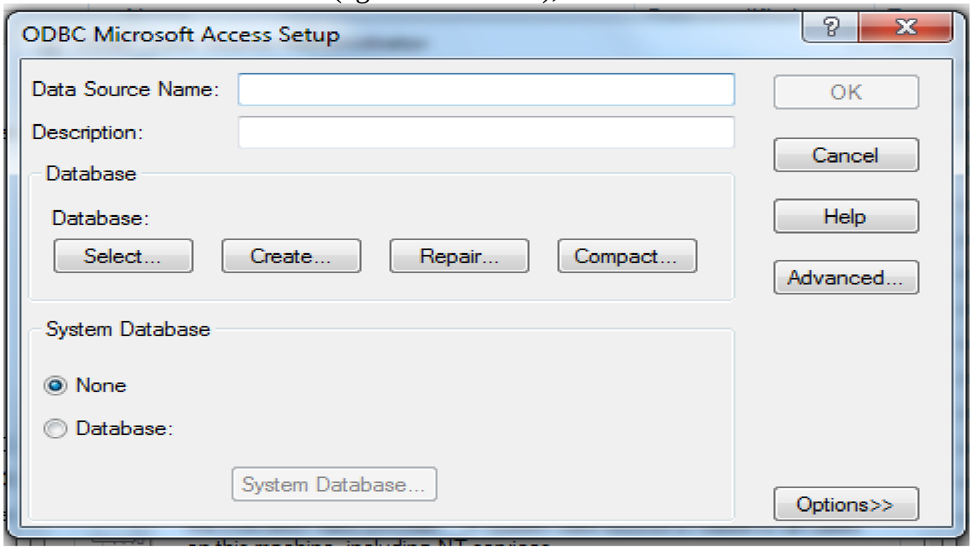


System Data Sources:

Name	Driver
linkedstd	Microsoft Access Driver (*.mdb, *.accdb)
TrioMotors	Microsoft Access Driver (*.mdb)

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

- Write data source name (eg LINKEDSTD), Then click **Select**.



Data Source Name:

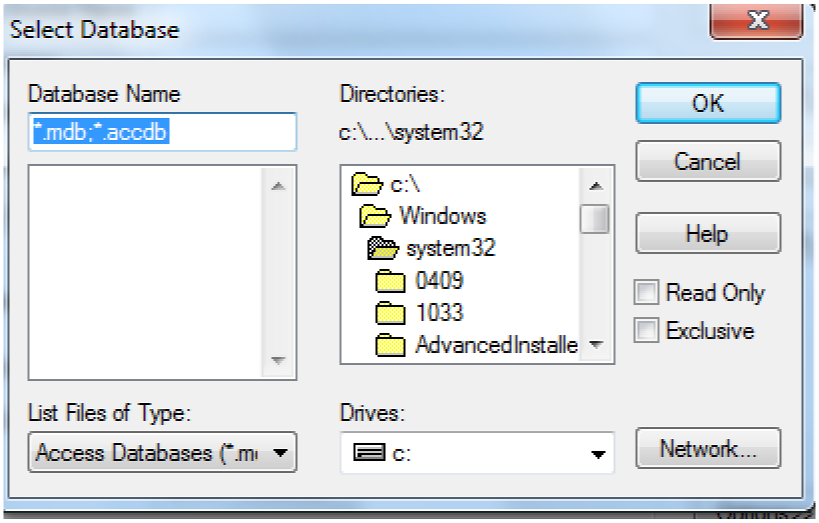
Description:

Database:

● None

○ Database:

- Browse the database you created in MS Access, and press OK.


Select Database

Database Name: *.mdb;*.accdb

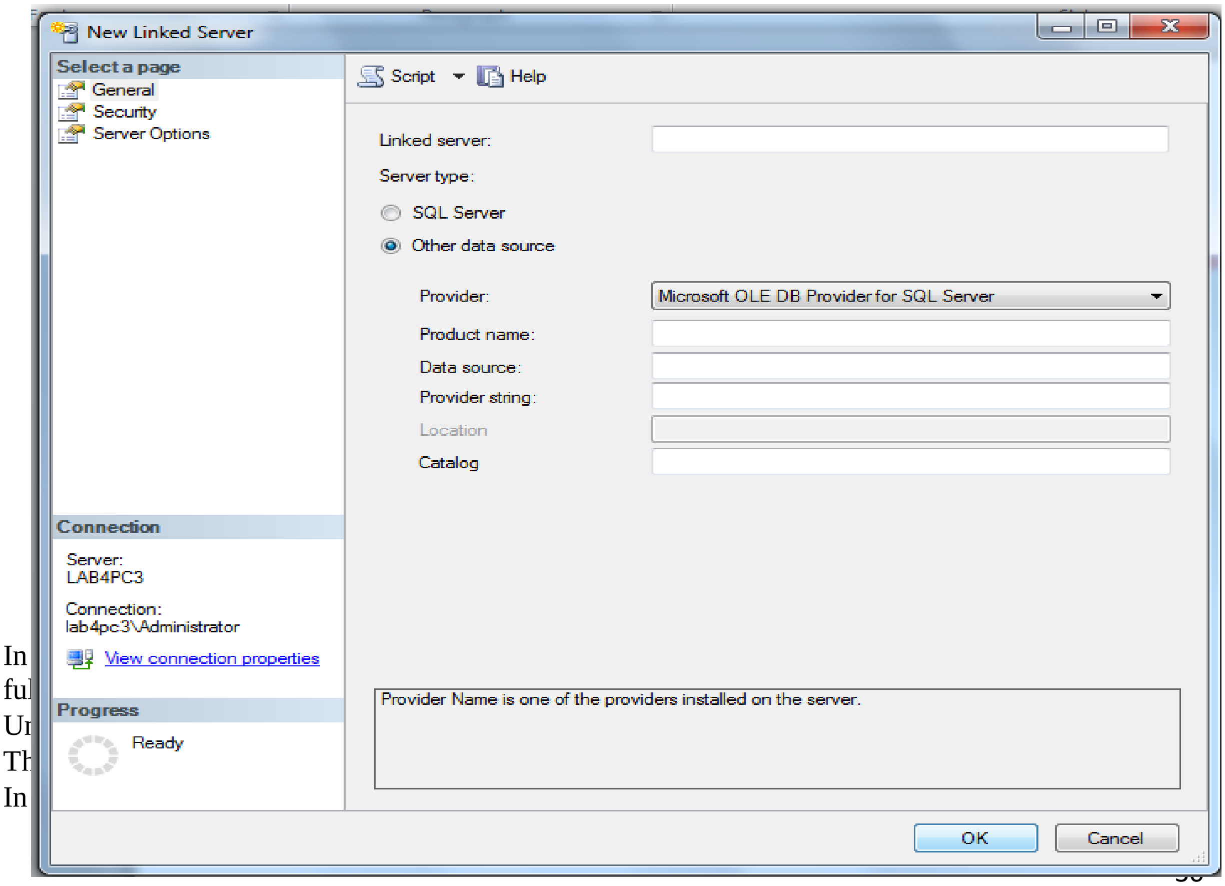
Directories: c:\...\system32

c:\
Windows
system32
0409
1033
AdvancedInstalle

List Files of Type: Access Databases (*.m

Drives: c:

- Now click **Start**, click **All Programs**, click **Microsoft SQL Server**, and then click **SQL Server Management Studio**.
- In the **Connect to Server** dialog box, specify the name of the appropriate SQL Server, and then click **Connect**.
- In **SQL Server Management Studio**, double-click **Server Objects**, right-click **Linked Servers**, and then click **New Linked Server**.


New Linked Server

Linked server:

Server type:

○ SQL Server

● Other data source

Provider: Microsoft OLE DB Provider for SQL Server

Product name:

Data source:

Provider string:

Location

Catalog

Connection

Server: LAB4PC3

Connection: lab4pc3\Administrator

Progress

Ready

Provider Name is one of the providers installed on the server.

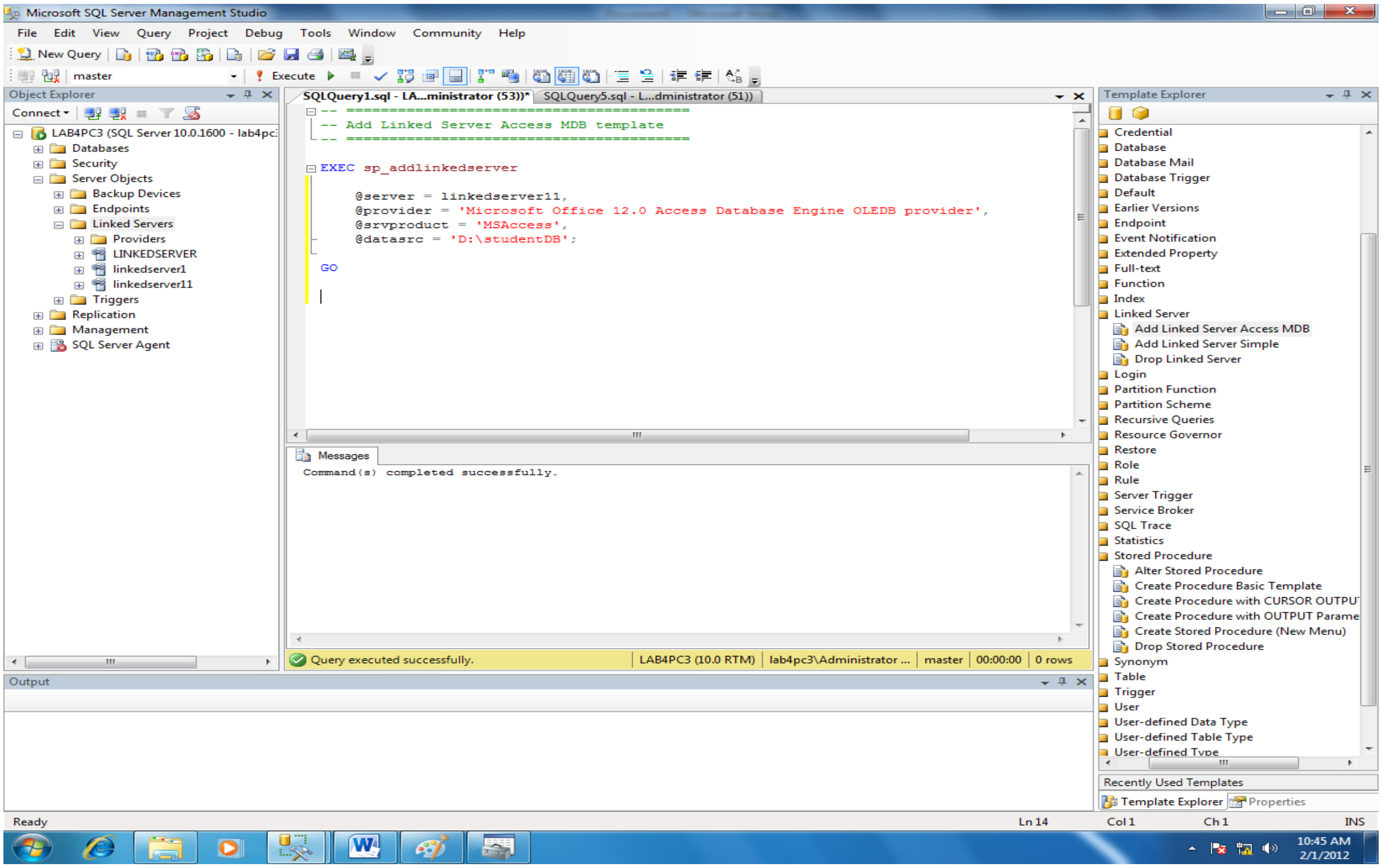
- In
ful
- Ur
- Th
- In

- Now write the **data source** name LINKEDSTD that created earlier.

USING STORED PROCEDURE

- Now click **Start**, click **All Programs**, click **Microsoft SQL Server**, and then click **SQL Server Management Studio**.
- Then from menu, select View, then Template Explorer.
- A side window will open, look for linked server and expand it,
- Choose first template for a stored procedure and make changes as follows.

```
EXECsp_addlinkedserver

        @server = linkedserver11,
        @provider ='Microsoft Office 12.0 Access Database Engine OLEDB provider',
        @srvproduct ='MSAccess',
        @datasrc ='D:\studentDB';
GO
```



```
-- ========================================
-- Add Linked Server Access MDB template
-- ========================================

EXEC sp_addlinkedserver

    @server = linkedserver11,
    @provider = 'Microsoft Office 12.0 Access Database Engine OLEDB provider',
    @srvproduct = 'MSAccess',
    @datasrc = 'D:\studentDB';

GO
```

```
Command(s) completed successfully.
```

Query executed successfully. | LAB4PC3 (10.0 RTM) | lab4pc3\Administrator ... | master | 00:00:00 | 0 rows

LAB 5
IMPLEMENTATION OF LINKED SERVER WITH ORACLE

OBJECTIVE

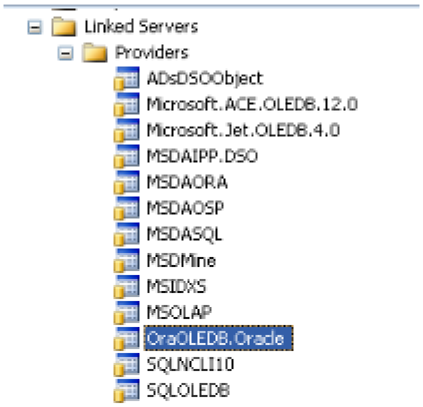
To implement linked server using Oracle

EXERCISE

Implement Linked Server using Oracle

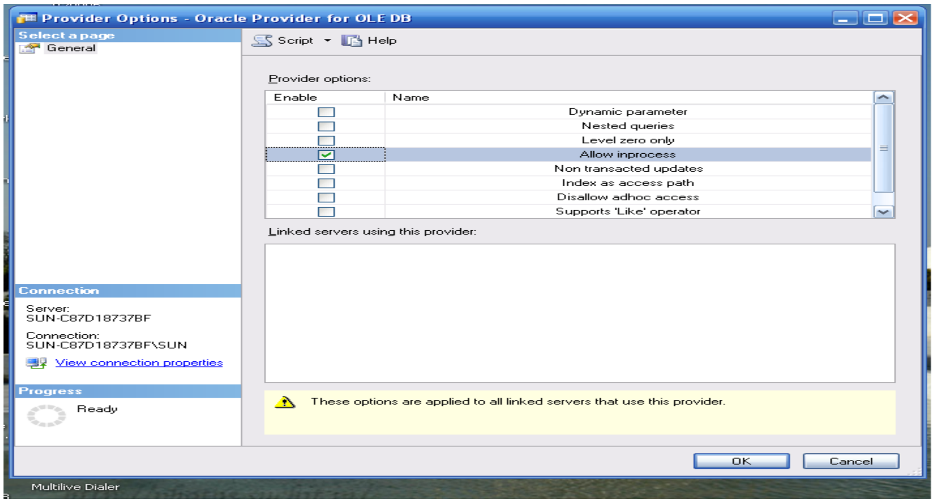
STEPS

1-) Creating a Linked Server to an Oracle System

- Go to Sql Server Management Studio→ Object Explorer → Server Objects → Linked Server → Providers →OraOLEDB.Oracle.
- **NOTE:** If this provider is present then continue your work. But this must be checked before proceeding otherwise linked server will not get configured.
- 

  Linked Servers
  Providers
  ADsDSOObject
  Microsoft.ACE.OLEDB.12.0
  Microsoft.Jet.OLEDB.4.0
  MSDAIPP.DSO
  MSDAORA
  MSDAOSP
  MSDASQL
  MSDMine
  MSIDXS
  MSOLAP
  OraOLEDB.Oracle
  SQLNCLI10
  SQLOLEDB

- Right Click on OraOLEDB.Oracle→ Properties → Check 'Allow in Process' →Press OK.



Provider Options - Oracle Provider for OLE DB

Select a page
General

Provider options:

Enable	Name
□	Dynamic parameter
□	Nested queries
□	Level zero only
☑	Allow inprocess
□	Non transacted updates
□	Index as access path
□	Disallow adhoc access
□	Supports 'Like' operator

Linked servers using this provider:

These options are applied to all linked servers that use this provider.

Connection

Server:
SUN-C87D18737BF

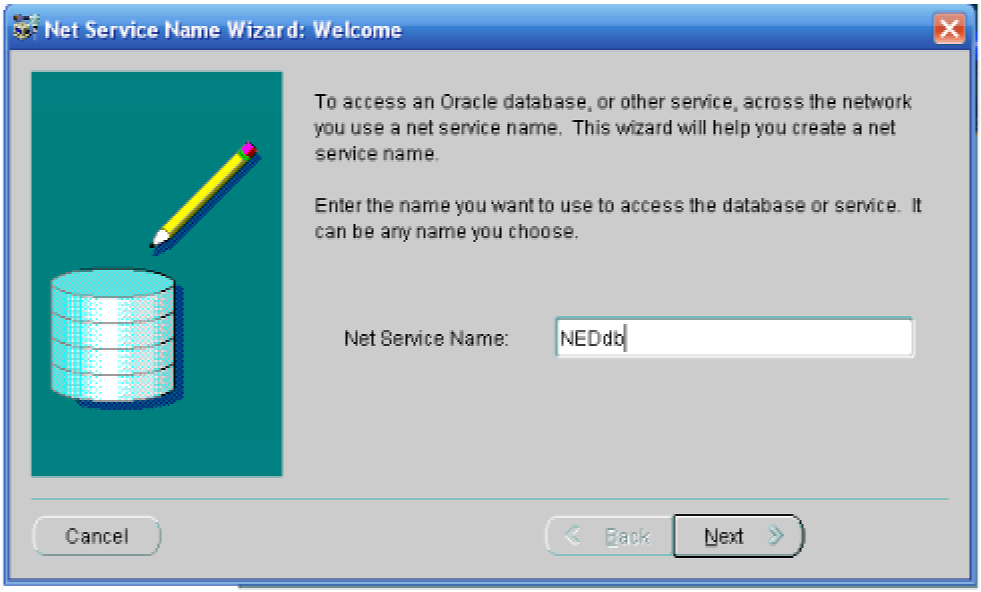
Connection:
SUN-C87D18737BF\SUN

View connection properties

Progress

Ready

- Fill Net Service Name(any name)→ Press Next



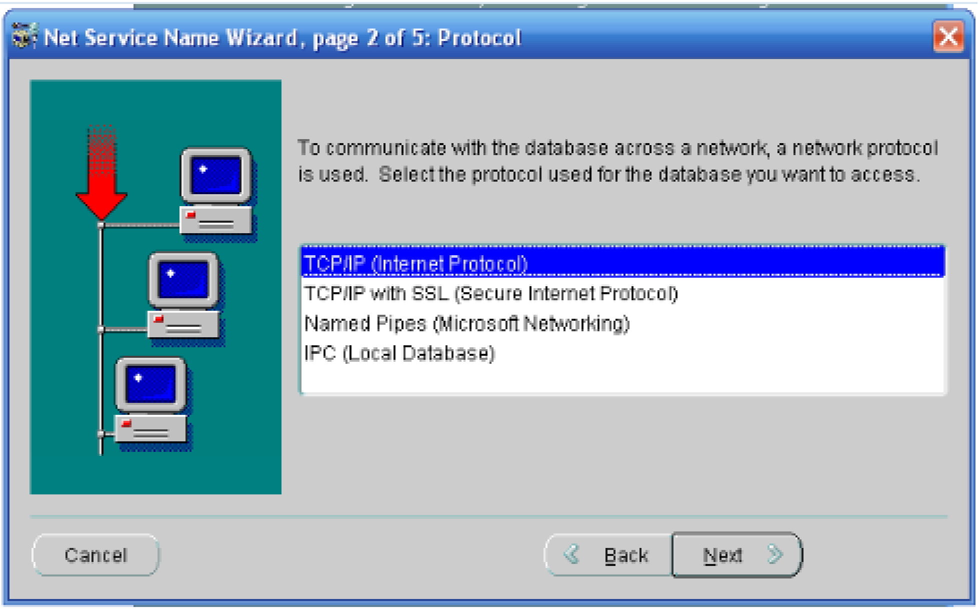
Net Service Name Wizard: Welcome

To access an Oracle database, or other service, across the network you use a net service name. This wizard will help you create a net service name.

Enter the name you want to use to access the database or service. It can be any name you choose.

Net Service Name: NEDdb

- Select TCP/IP (Internet protocol)→ Press Next

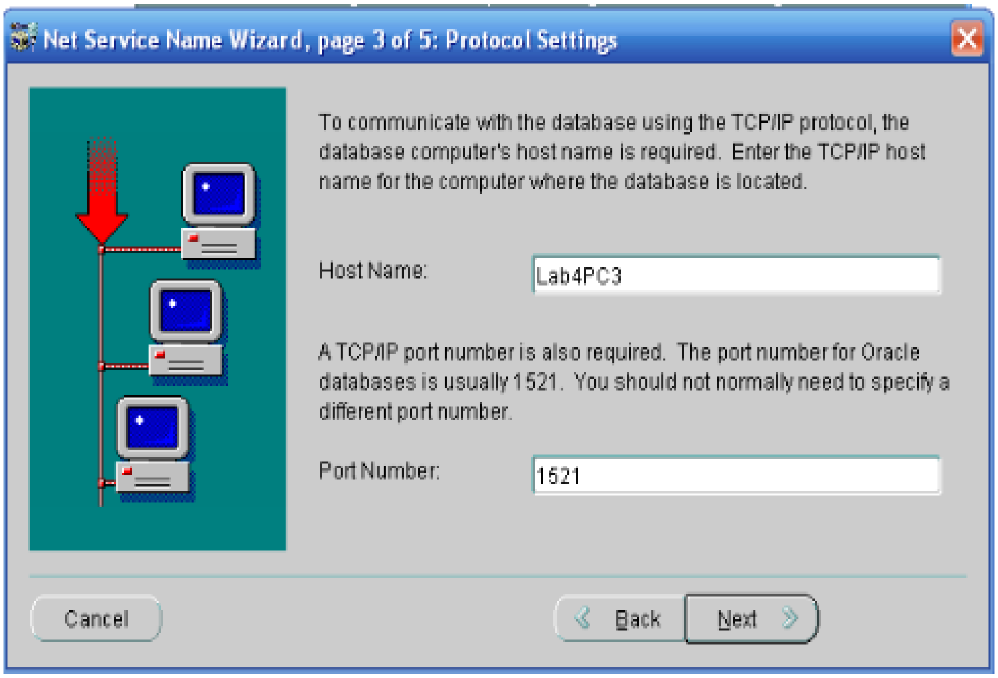


Net Service Name Wizard, page 2 of 5: Protocol

To communicate with the database across a network, a network protocol is used. Select the protocol used for the database you want to access.

TCP/IP (Internet Protocol)
TCP/IP with SSL (Secure Internet Protocol)
Named Pipes (Microsoft Networking)
IPC (Local Database)

- Enter Host Name (your PC Name) and port number 1521 → Press OK



Net Service Name Wizard, page 3 of 5: Protocol Settings

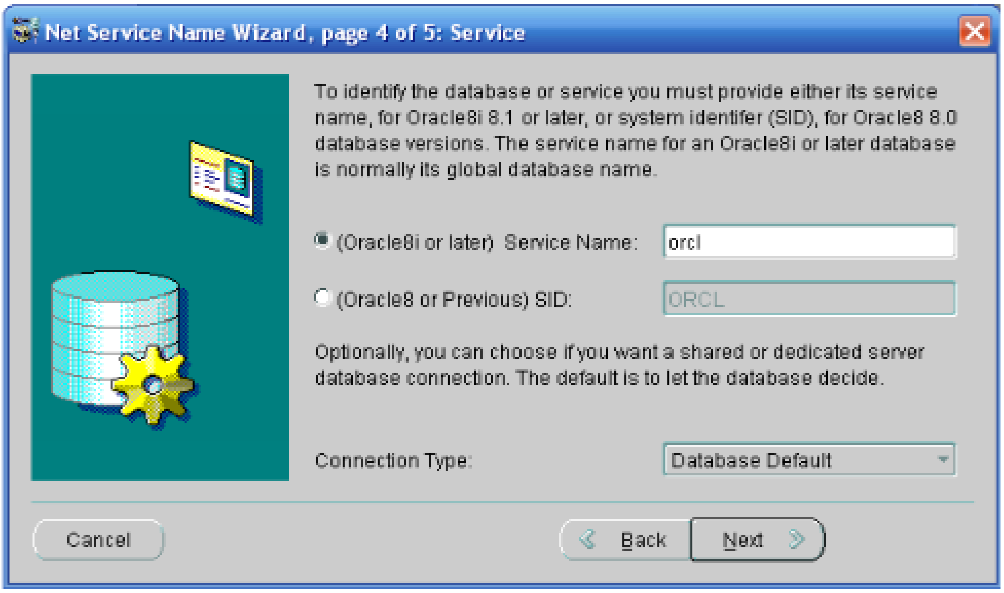
To communicate with the database using the TCP/IP protocol, the database computer's host name is required. Enter the TCP/IP host name for the computer where the database is located.

Host Name: Lab4PC3

A TCP/IP port number is also required. The port number for Oracle databases is usually 1521. You should not normally need to specify a different port number.

Port Number: 1521

- Enter Service Name 'orcl' (this name is present TNSNAMES.ORA file which is located in oracle's folder) → Press Next



Net Service Name Wizard, page 4 of 5: Service

To identify the database or service you must provide either its service name, for Oracle8i 8.1 or later, or system identifer (SID), for Oracle8 8.0 database versions. The service name for an Oracle8i or later database is normally its global database name.

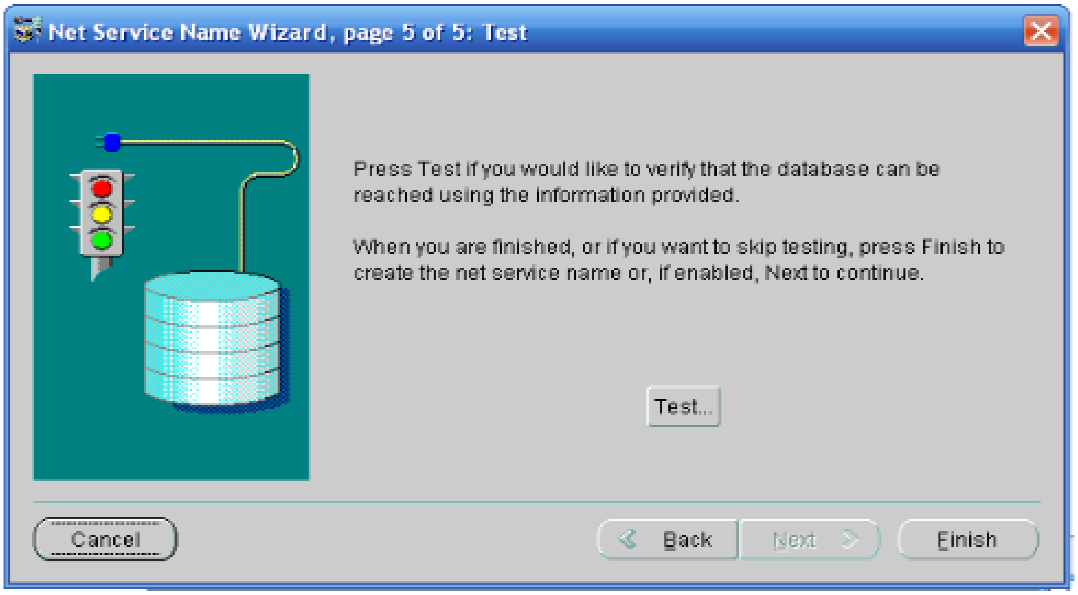
● (Oracle8i or later) Service Name: orcl

○ (Oracle8 or Previous) SID: ORCL

Optionally, you can choose if you want a shared or dedicated server database connection. The default is to let the database decide.

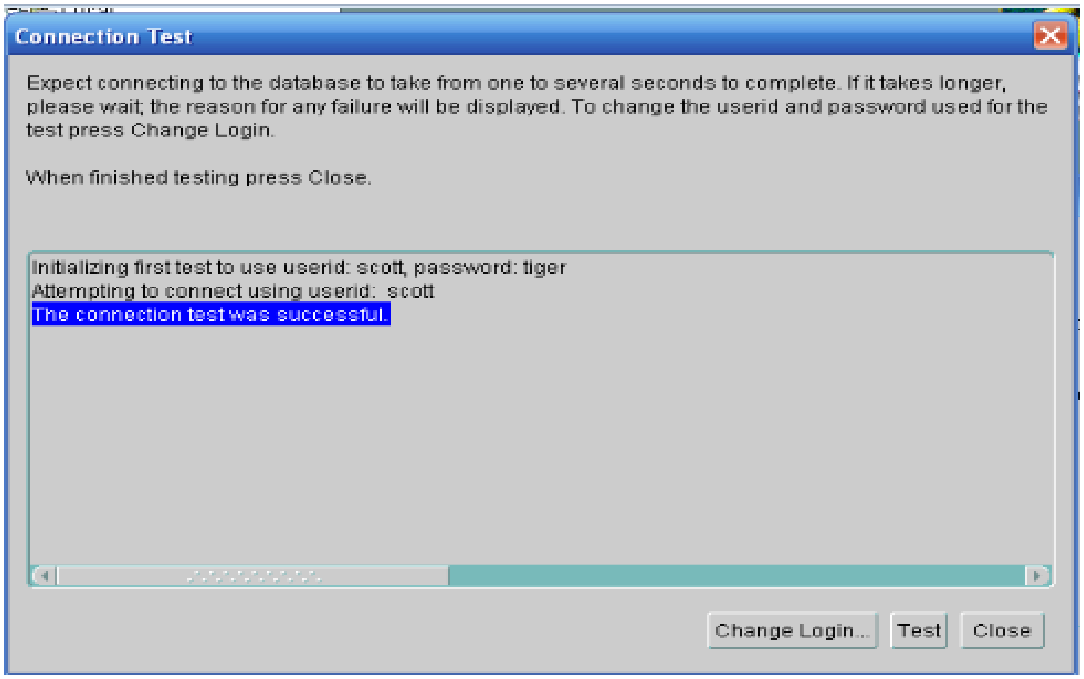
Connection Type: Database Default

- Test the Connection→ 'Connection Successful' message will appear on the screen → Close the window



Press Test if you would like to verify that the database can be reached using the information provided.

When you are finished, or if you want to skip testing, press Finish to create the net service name or, if enabled, Next to continue.

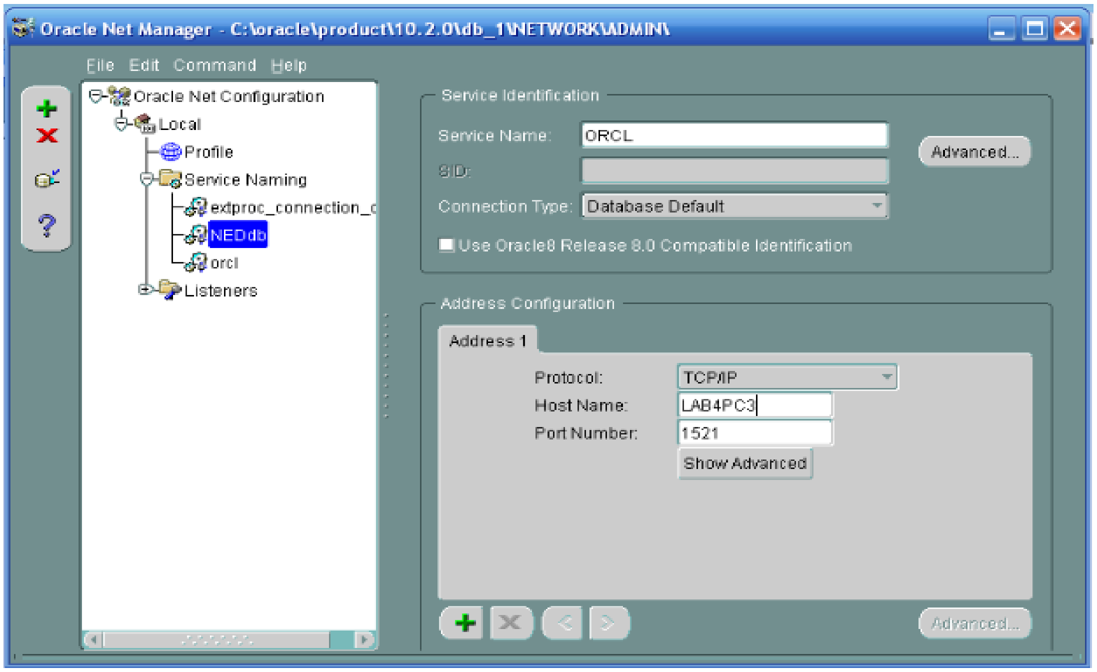


Expect connecting to the database to take from one to several seconds to complete. If it takes longer, please wait; the reason for any failure will be displayed. To change the userid and password used for the test press Change Login.

When finished testing press Close.

```
Initializing first test to use userid: scott, password: tiger
Attempting to connect using userid:  scott
The connection test was successful.
```

- Enter service name, host name and port number →Menu bar → Click File →Save Network Configuration



Service Name: ORCL

SID:

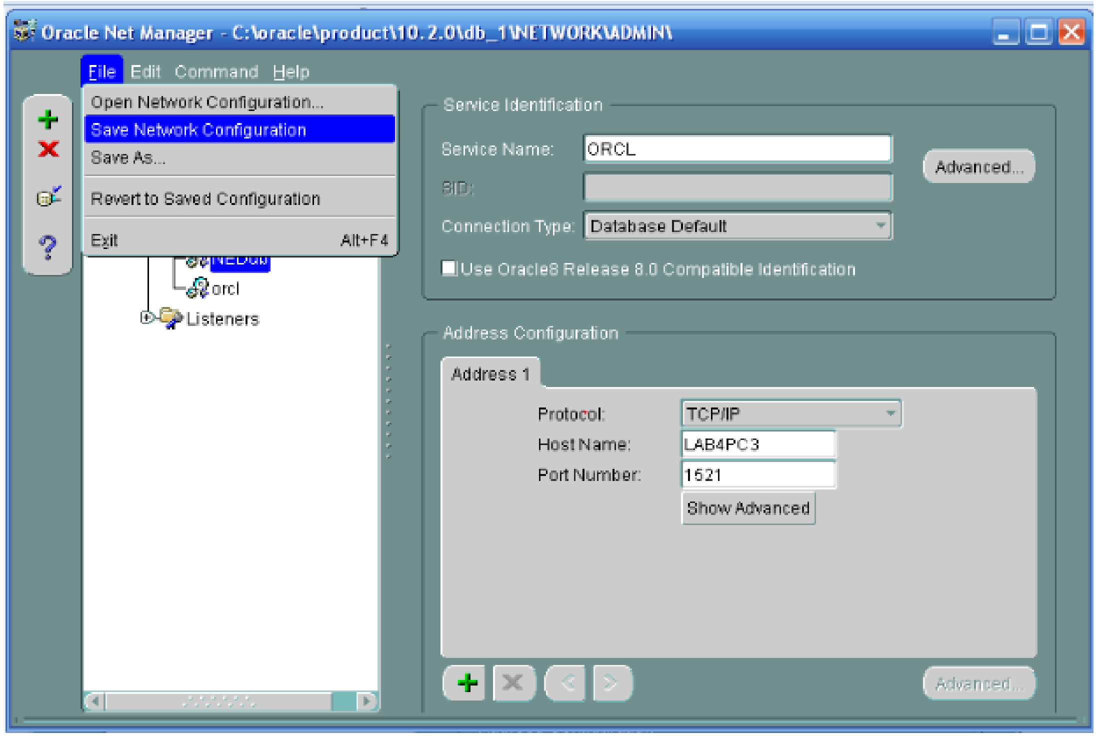
Connection Type: Database Default

□ Use Oracle8 Release 8.0 Compatible Identification

Protocol: TCP/IP

Host Name: LAB4PC3

Port Number: 1521



Oracle Net Manager - C:\oracle\product\10.2.0\db_1\NETWORK\ADMIN\
File
Edit
Command
Help
Open Network Configuration...
Save Network Configuration
Save As...
Revert to Saved Configuration
Exit
Alt+F4
NEDdb
orcl
Listeners
Service Identification
Advanced...
Address Configuration
Address 1
Show Advanced
Advanced...

Service Name: ORCL

SID:

Connection Type: Database Default

□ Use Oracle8 Release 8.0 Compatible Identification

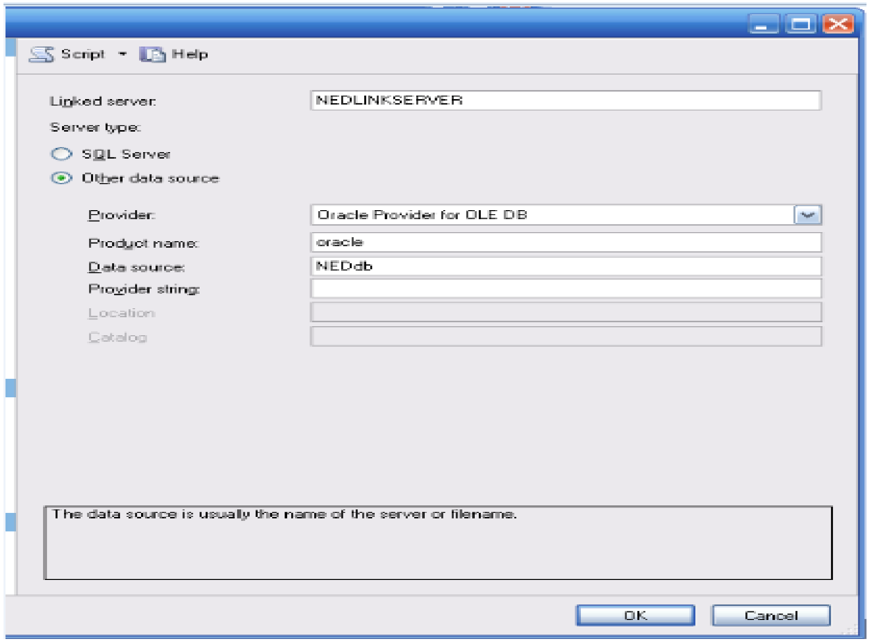
Protocol: TCP/IP

Host Name: LAB4PC3

Port Number: 1521

2-) Creating a Linked Server to an Oracle System using SQL Server Management Studio

- Go to SSMS → Object Explorer → Server Objects → Right Click Linked Server → New Linked Server



Linked server: NEDLINKSERVER

Server type:

○ SQL Server

● Other data source

Provider: Oracle Provider for OLE DB

Product name: oracle

Data source: NEDdb

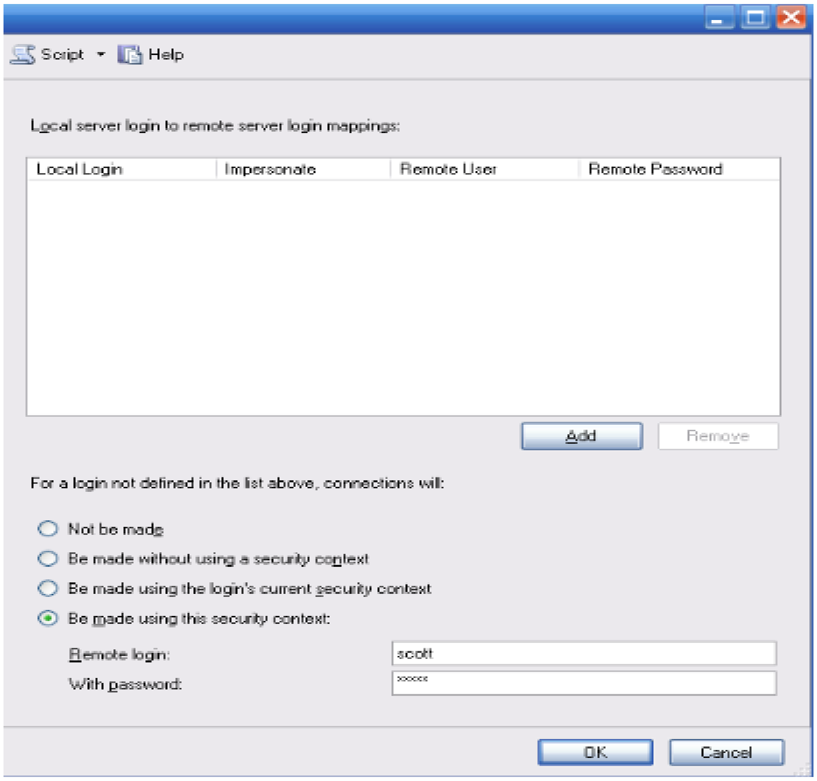
Provider string:

Location:

Catalog:

The data source is usually the name of the server or filename.

- Enter Linked Server Name (any name), Select Oracle provider for OLEDB, enter product name and data source → Press OK → Select Be made using Security Context → enter username and password.



Local server login to remote server login mappings:

Local Login	Impersonate	Remote User	Remote Password

For a login not defined in the list above, connections will:

○ Not be made

○ Be made without using a security context

○ Be made using the login's current security context

● Be made using this security context:

Remote login: scott

With password: ******

3-) Creating a Linked Server via T-SQL

A linked server can also be created via T-SQL using the sp_addlinkedserver stored procedure. When a distributed query is executed that accesses a table on the linked server, the local server must log on to the linked server on behalf of the user to access that table. The sp_addlinkedsrvlogin stored procedure creates a mapping between a login on the local instance of SQL Server and a security account on a remote server.

Executing sp_addlinkedserver automatically creates a default mapping between all logins on the local server and remote logins on the linked server. The default mapping states that SQL Server uses the user credentials of the local login when connecting to the linked server on behalf of the login. This is equivalent to executing sp_addlinkedsrvlogin with @useself set to true for the linked server, without specifying a local user name.

4-) Using OPENQUERY

The following is an example of using the T-SQL OPENQUERY construct to perform a distributed query against the linked server. This query retrieves records from the table called prod_table that resides on the remote server to which the linked server called PRODMASTER is mapped.

```
select * from openquery(PRODMASTER, 'select * from prod_table')
```

LAB 6
DISTRIBUTED DATABASE STRUCTURE

OBJECTIVE

To develop a distributed database structure against a given scenario of an online real estate

SCENARIO

Online Real Estate System

Real estate is a legal term that deals land site that are fixed in location -- immovable. Real estate is often considered synonymous with real property.
It is an integrated property platform offering a wide array of quality property investments ranging from residential and luxurious to commercial options which include homes, villas, apartments, flats, farm houses, residential lands/plots, and commercial lands/plots, shops in markets and plazas in different cities throughout Pakistan. These services are not only restricted to buying and selling in real estate accommodation and property but also encompasses home, villa, shops and other kinds of property rentals and real estate leasing. These systems should handle all these rent also.

Main Features:

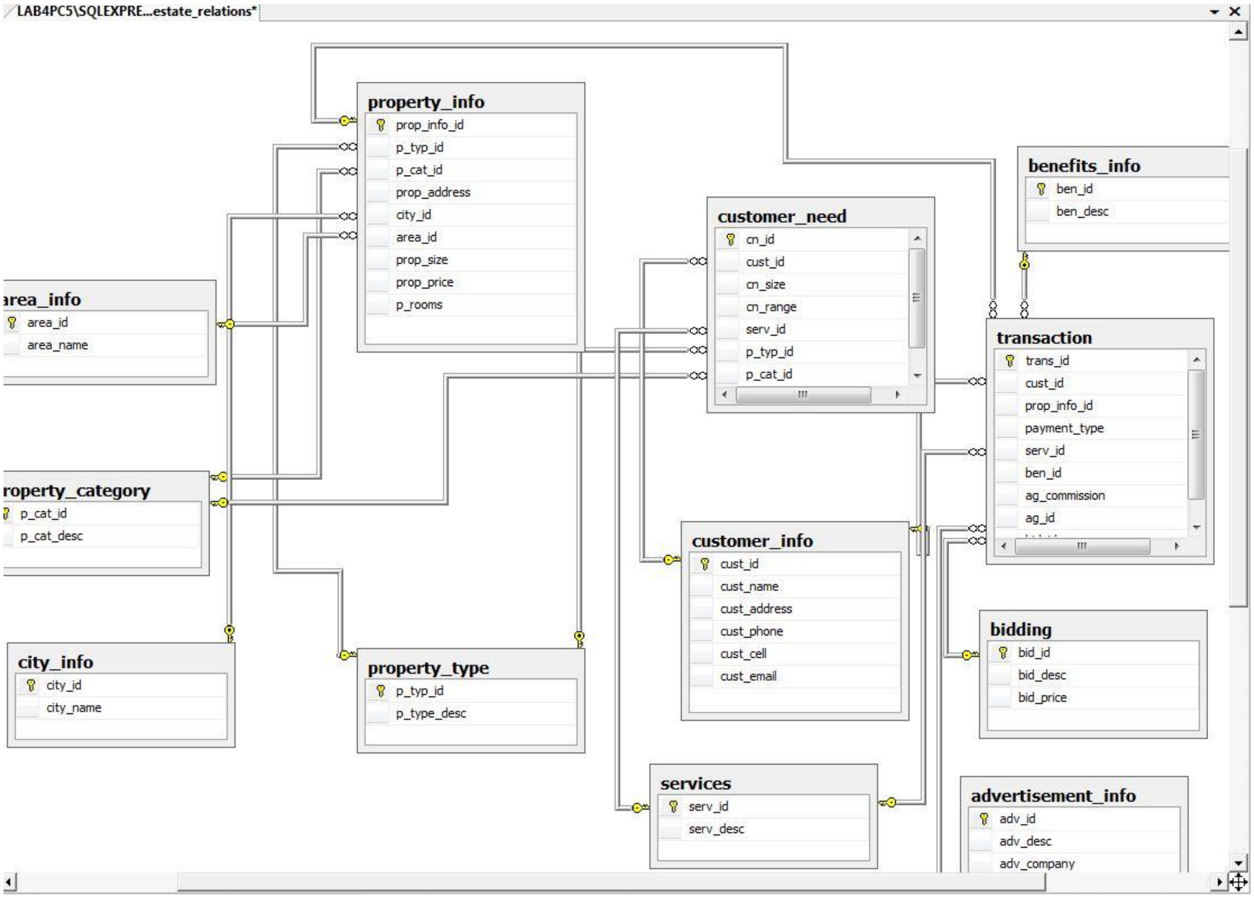
- Registration of client / companies
- Client’s requests and needs
- Agents Directory Management
- Customers to search for properties to buy/rent

- Buy Property, Sell Property or to Rent a Property
- Property Price
- Property consulting services and hold
- Lettings property management
- Maintain records of customers
- Benefits if you decide to buy sell or rent any real estate
- Biding Handling
- Handling advertisements
- Inform to different clients through automatic Email and auto generated SMS.
- Categories
- Construction and development features including wages and expenditures.
- Reports of all the things.

EXERCISE

Develop a database of a given scenario in SQL Server

DATABASE DIAGRAM



SCRIPT

```
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[benefits_info](
	[ben_id] [varchar](10) NOT NULL,
	[ben_desc] [varchar](50) NOT NULL,
 CONSTRAINT [PK_benefits_info] PRIMARY KEY CLUSTERED
(
	[ben_id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[area_info]    Script Date: 03/08/2012 13:25:10 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[area_info](
	[area_id] [varchar](10) NOT NULL,
	[area_name] [varchar](50) NOT NULL,
 CONSTRAINT [PK_area_info] PRIMARY KEY CLUSTERED
(
	[area_id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[agent_info]    Script Date: 03/08/2012 13:25:10 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[agent_info](
	[ag_id] [varchar](10) NOT NULL,
	[ag_name] [varchar](50) NOT NULL,
	[ag_address] [varchar](50) NOT NULL,
	[ag_phone] [varchar](20) NULL,
	[ag_cell] [varchar](20) NOT NULL,
	[city_id] [varchar](10) NOT NULL,
	[area_id] [varchar](10) NOT NULL,
 CONSTRAINT [PK_agent_info] PRIMARY KEY CLUSTERED
(
	[ag_id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[advertisement_info]    Script Date: 03/08/2012 13:25:10 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[advertisement_info](
	[adv_id] [varchar](10) NOT NULL,
	[adv_desc] [varchar](150) NOT NULL,
	[adv_company] [varchar](50) NOT NULL,
	[date_of_publish] [date] NOT NULL,
	[date_of_expiry] [date] NOT NULL,
	[adv_cost] [varchar](30) NOT NULL,
 CONSTRAINT [PK_advertisement_info] PRIMARY KEY CLUSTERED
(
	[adv_id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[services]    Script Date: 03/08/2012 13:25:10 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[services](
	[serv_id] [varchar](10) NOT NULL,
	[serv_desc] [varchar](30) NOT NULL,
 CONSTRAINT [PK_services] PRIMARY KEY CLUSTERED
(
	[serv_id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[property_type]    Script Date: 03/08/2012 13:25:10 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[property_type](
	[p_typ_id] [varchar](15) NOT NULL,
	[p_type_desc] [varchar](30) NOT NULL,
 CONSTRAINT [PK_property_type] PRIMARY KEY CLUSTERED
(
	[p_typ_id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[property_info]    Script Date: 03/08/2012 13:25:10 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[property_info](
	[prop_info_id] [varchar](10) NOT NULL,
	[p_typ_id] [varchar](15) NOT NULL,
	[p_cat_id] [varchar](15) NOT NULL,
	[prop_address] [varchar](50) NOT NULL,
	[city_id] [varchar](10) NOT NULL,
	[area_id] [varchar](10) NOT NULL,
	[prop_size] [varchar](20) NOT NULL,
	[prop_price] [varchar](25) NOT NULL,
	[p_rooms] [varchar](3) NOT NULL,
 CONSTRAINT [PK_property_info] PRIMARY KEY CLUSTERED
(
	[prop_info_id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[customer_need]    Script Date: 03/08/2012 13:25:10 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[customer_need](
	[cn_id] [varchar](20) NOT NULL,
	[cust_id] [varchar](20) NOT NULL,
	[cn_size] [varchar](20) NOT NULL,
	[cn_range] [varchar](25) NOT NULL,
	[serv_id] [varchar](10) NOT NULL,
	[p_typ_id] [varchar](15) NOT NULL,
	[p_cat_id] [varchar](15) NOT NULL,
 CONSTRAINT [PK_customer_need] PRIMARY KEY CLUSTERED
(
	[cn_id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  Table [dbo].[transaction]    Script Date: 03/08/2012 13:25:10 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[transaction](
	[trans_id] [varchar](10) NOT NULL,
	[cust_id] [varchar](20) NOT NULL,
	[prop_info_id] [varchar](10) NOT NULL,
	[payment_type] [varchar](30) NOT NULL,
	[serv_id] [varchar](10) NOT NULL,
	[ben_id] [varchar](10) NOT NULL,
	[ag_commission] [varchar](20) NOT NULL,
	[ag_id] [varchar](10) NOT NULL,
	[bid_id] [varchar](10) NOT NULL,
 CONSTRAINT [PK_transaction] PRIMARY KEY CLUSTERED
(
	[trans_id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
```

```
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  ForeignKey [FK_customer_need_customer_info]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[customer_need]  WITH CHECK ADD  CONSTRAINT
[FK_customer_need_customer_info] FOREIGN KEY([cust_id])
REFERENCES [dbo].[customer_info] ([cust_id])
GO
ALTER TABLE [dbo].[customer_need] CHECK CONSTRAINT [FK_customer_need_customer_info]
GO
/****** Object:  ForeignKey [FK_customer_need_property_category]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[customer_need]  WITH CHECK ADD  CONSTRAINT
[FK_customer_need_property_category] FOREIGN KEY([p_cat_id])
REFERENCES [dbo].[property_category] ([p_cat_id])
GO
ALTER TABLE [dbo].[customer_need] CHECK CONSTRAINT [FK_customer_need_property_category]
GO
/****** Object:  ForeignKey [FK_customer_need_property_type]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[customer_need]  WITH CHECK ADD  CONSTRAINT
[FK_customer_need_property_type] FOREIGN KEY([p_typ_id])
REFERENCES [dbo].[property_type] ([p_typ_id])
GO
ALTER TABLE [dbo].[customer_need] CHECK CONSTRAINT [FK_customer_need_property_type]
GO
/****** Object:  ForeignKey [FK_customer_need_services]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[customer_need]  WITH CHECK ADD  CONSTRAINT [FK_customer_need_services]
FOREIGN KEY([serv_id])
REFERENCES [dbo].[services] ([serv_id])
GO
ALTER TABLE [dbo].[customer_need] CHECK CONSTRAINT [FK_customer_need_services]
GO
/****** Object:  ForeignKey [FK_property_info_area_info]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[property_info]  WITH CHECK ADD  CONSTRAINT [FK_property_info_area_info]
FOREIGN KEY([area_id])
REFERENCES [dbo].[area_info] ([area_id])
GO
ALTER TABLE [dbo].[property_info] CHECK CONSTRAINT [FK_property_info_area_info]
GO
/****** Object:  ForeignKey [FK_property_info_city_info]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[property_info]  WITH CHECK ADD  CONSTRAINT [FK_property_info_city_info]
FOREIGN KEY([city_id])
REFERENCES [dbo].[city_info] ([city_id])
GO
ALTER TABLE [dbo].[property_info] CHECK CONSTRAINT [FK_property_info_city_info]
GO
/****** Object:  ForeignKey [FK_property_info_property_category]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[property_info]  WITH CHECK ADD  CONSTRAINT
[FK_property_info_property_category] FOREIGN KEY([p_cat_id])
REFERENCES [dbo].[property_category] ([p_cat_id])
GO
ALTER TABLE [dbo].[property_info] CHECK CONSTRAINT [FK_property_info_property_category]
GO
/****** Object:  ForeignKey [FK_property_info_property_type]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[property_info]  WITH CHECK ADD  CONSTRAINT [FK_property_info_property_type]
FOREIGN KEY([p_typ_id])
```

```
REFERENCES [dbo].[property_type] ([p_typ_id])
GO
ALTER TABLE [dbo].[property_info] CHECK CONSTRAINT [FK_property_info_property_type]
GO
/****** Object:  ForeignKey [FK_transaction_agent_info]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[transaction]  WITH CHECK ADD  CONSTRAINT [FK_transaction_agent_info]
FOREIGN KEY([ag_id])
REFERENCES [dbo].[agent_info] ([ag_id])
GO
ALTER TABLE [dbo].[transaction] CHECK CONSTRAINT [FK_transaction_agent_info]
GO
/****** Object:  ForeignKey [FK_transaction_benefits_info]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[transaction]  WITH CHECK ADD  CONSTRAINT [FK_transaction_benefits_info]
FOREIGN KEY([ben_id])
REFERENCES [dbo].[benefits_info] ([ben_id])
GO
ALTER TABLE [dbo].[transaction] CHECK CONSTRAINT [FK_transaction_benefits_info]
GO
/****** Object:  ForeignKey [FK_transaction_bidding]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[transaction]  WITH CHECK ADD  CONSTRAINT [FK_transaction_bidding] FOREIGN
KEY([bid_id])
REFERENCES [dbo].[bidding] ([bid_id])
GO
ALTER TABLE [dbo].[transaction] CHECK CONSTRAINT [FK_transaction_bidding]
GO
/****** Object:  ForeignKey [FK_transaction_customer_info]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[transaction]  WITH CHECK ADD  CONSTRAINT [FK_transaction_customer_info]
FOREIGN KEY([cust_id])
REFERENCES [dbo].[customer_info] ([cust_id])
GO
ALTER TABLE [dbo].[transaction] CHECK CONSTRAINT [FK_transaction_customer_info]
GO
/****** Object:  ForeignKey [FK_transaction_property_info]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[transaction]  WITH CHECK ADD  CONSTRAINT [FK_transaction_property_info]
FOREIGN KEY([prop_info_id])
REFERENCES [dbo].[property_info] ([prop_info_id])
GO
ALTER TABLE [dbo].[transaction] CHECK CONSTRAINT [FK_transaction_property_info]
GO
/****** Object:  ForeignKey [FK_transaction_services]    Script Date: 03/08/2012 13:25:10 ******/
ALTER TABLE [dbo].[transaction]  WITH CHECK ADD  CONSTRAINT [FK_transaction_services] FOREIGN
KEY([serv_id])
REFERENCES [dbo].[services] ([serv_id])
GO
ALTER TABLE [dbo].[transaction] CHECK CONSTRAINT [FK_transaction_services]
GO
```

LAB 7
SIMPLE MAIL TRANSFER PROTOCOL

OBJECTIVE

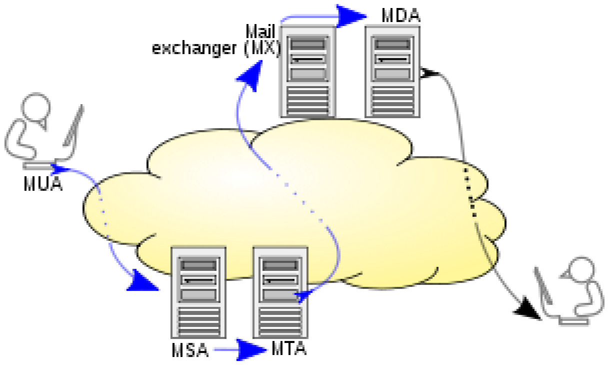
Implementation of Simple Mail Transfer protocol

THEORY

SIMPLE MAIL TRANSFER PROTOCOL (SMTP)

- Simple Mail Transfer Protocol (SMTP) is an Internet standard for electronic mail (e-mail) transmission across Internet Protocol (IP) networks.
- SMTP is specified for outgoing mail transport and uses TCP port 25.
- The protocol for new submissions is effectively the same as SMTP, but it uses port 587 instead.
- SMTP connections secured by SSL are known by the shorthand SMTPS, though SMTPS is not a protocol in its own right.
- While electronic mail servers and other mail transfer agents use SMTP to send and receive mail messages, user-level client mail applications typically only use SMTP for sending messages to a mail server for relaying.
- For receiving messages, client applications usually use either the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP) or a proprietary system (such as Microsoft Exchange or Lotus Notes/Domino) to access their mail box accounts on a mail server.

MAIL PROCESSING MODEL



Blue arrows can be implemented using SMTP variations.

Email is submitted by a mail client (MUA, mail user agent) to a mail server (MSA, mail submission agent) using SMTP on TCP port 587. Most mailbox providers still allow submission on traditional port 25. From there, the MSA delivers the mail to its mail transfer agent (MTA, mail transfer agent). Often, these two agents are just different instances of the same software launched with different options on the same machine. Local processing can be done either on a single machine, or split among various appliances; in the former case, involved processes can share files; in the latter case, SMTP is used to transfer the message internally, with each host configured to use the next appliance as a smart host. Each process is an MTA in its own right; that is, an SMTP server.

The boundary MTA has to locate the target host. It uses the Domain name system (DNS) to look up the mail exchanger record (MX record) for the recipient's domain (the part of the address on the right of @). The returned MX record contains the name of the target host. The MTA next connects to the exchange server as an SMTP client. (The article on MX record discusses many factors in determining which server the sending MTA connects to.)

Once the MX target accepts the incoming message, it hands it to a mail delivery agent (MDA) for local mail delivery. An MDA is able to save messages in the relevant mailbox format. Again, mail reception can be done using many computers or just one —the picture displays two nearby boxes in either case. An

MDA may deliver messages directly to storage, or forward them over a network using SMTP, or any other means, including the Local Mail Transfer Protocol (LMTP), a derivative of SMTP designed for this purpose.

Once delivered to the local mail server, the mail is stored for batch retrieval by authenticated mail clients (MUAs). Mail is retrieved by end-user applications, called email clients, using Internet Message Access Protocol (IMAP), a protocol that both facilitates access to mail and manages stored mail, or the Post Office Protocol (POP) which typically uses the traditional mbox mail file format or a proprietary system such as Microsoft Exchange/Outlook or Lotus Notes/Domino. Webmail clients may use either method, but the retrieval protocol is often not a formal standard.

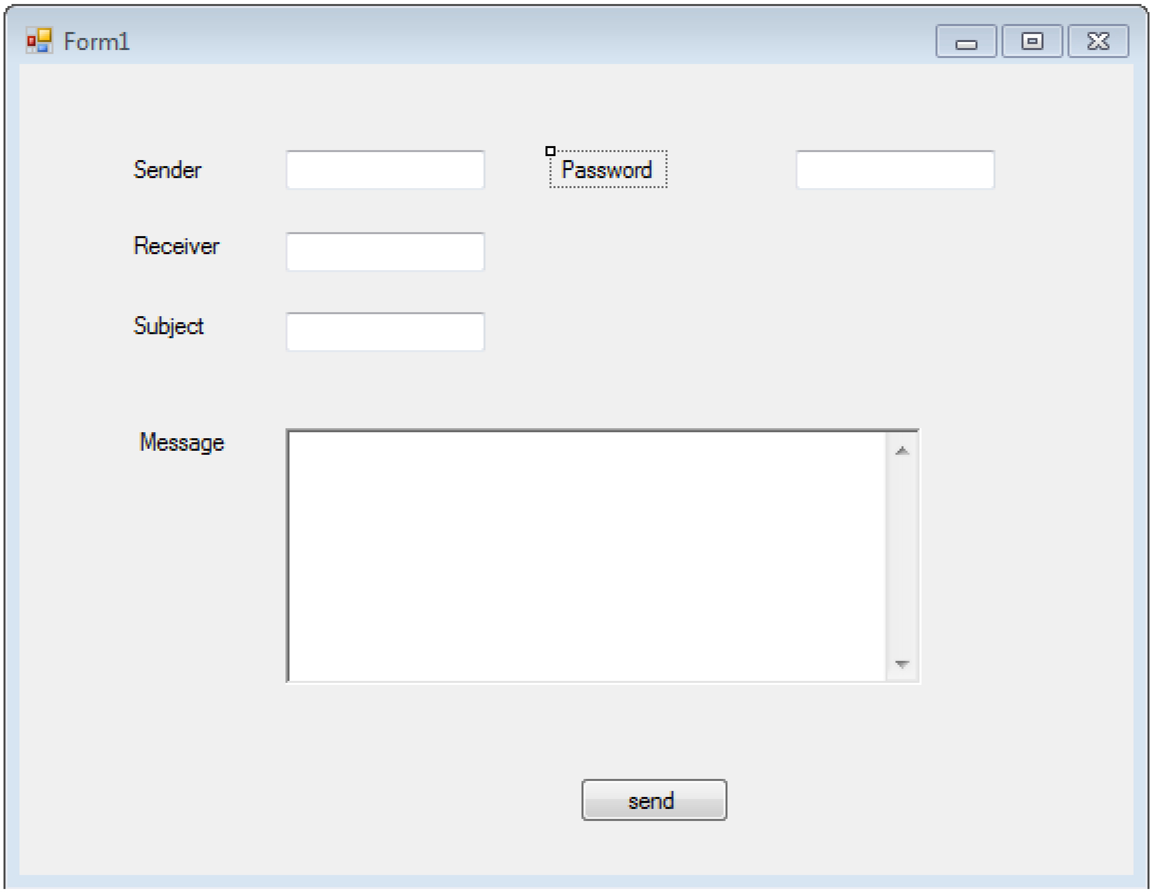
EXERCISE

Use gmail/ hotmail/yahoo as smtp server

STEPS

Open window form application in visual studio and make a form as shown in the figure below

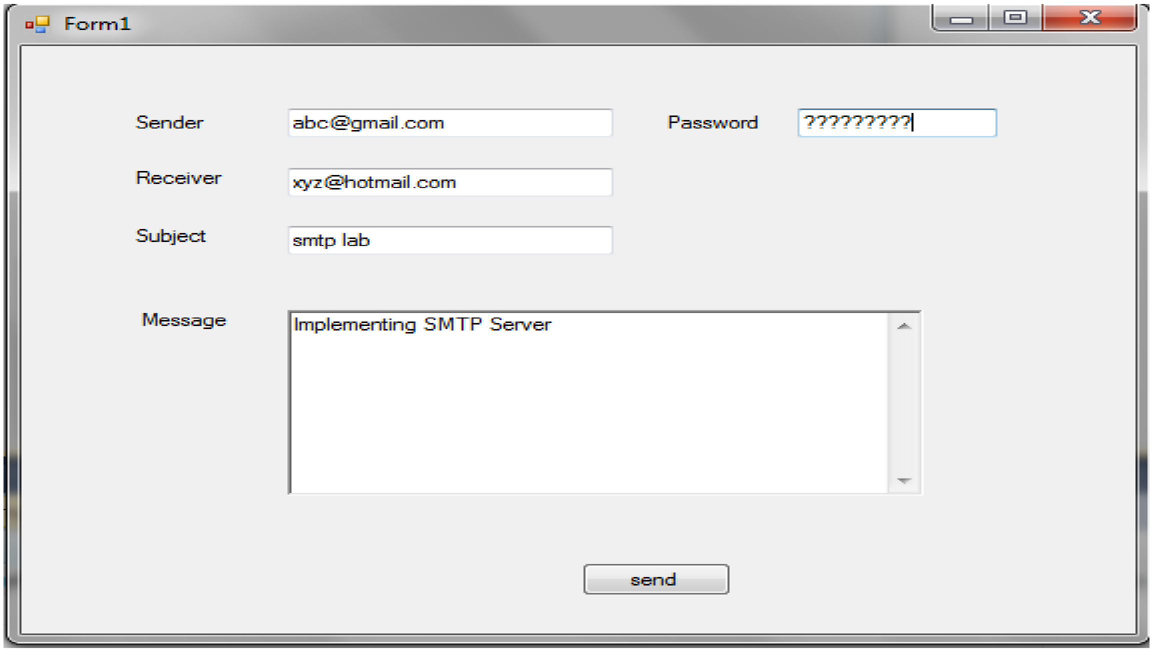
1-) FORM DESIGN



2-) CODING BEHIND SEND BUTTON

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
**using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net.Mail;
namespace smtp_lab
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            string Password = textBox4.Text;
            string MessageBody = "";
            MailMessage mail = new MailMessage();
            SmtpClient Smtp_Client = new SmtpClient("smtp.gmail.com");
            mail.From = new MailAddress(this.textBox1.Text.Trim());
            mail.To.Add(this.textBox2.Text.Trim());
            mail.Subject = this.textBox3.Text;
            mail.Body = this.richTextBox1.Text;
            mail.IsBodyHtml = false;
            Smtp_Client.Port = 587;
            Smtp_Client.Credentials = new System.Net.NetworkCredential(this.textBox1.Text,Password);
            Smtp_Client.EnableSsl = true;
            try
            {
                Smtp_Client.Send(mail);
            }
            catch (Exception Ex)
        {  MessageBox.Show(Ex.Message);   }}
}
```

3-) EXECUTE



Sender abc@gmail.com

Password ?????????

Receiver xyz@hotmail.com

Subject smtp lab

Message Implementing SMTP Server

LAB 8
CLIENT SERVER PROGRAMMING WITH TCP/IP

OBJECTIVE

Implementation of Client Server Programming with TCP/IP

THEORY

Inter-Process Communication i.e. the capability of two or more physically connected machines to exchange data, plays a very important role in enterprise software development. TCP/IP is the most common standard adopted for such communication. Under TCP/IP each machine is identified by a unique 4 byte integer referred to as its IP address (usually formatted as 192.168.0.101). For easy remembrance, this IP address is mostly bound to a user-friendly host name. The program below (*showip.cs*) uses the System.Net.Dns class to display the IP address of the machine whose name is passed in the first command-line argument. In the absence of command-line arguments, it displays the name and IP address of the local machine.

```
using System;
using System.Net;
class ShowIP{
   public static void Main(string[] args){
      string name = (args.Length < 1) ? Dns.GetHostName() : args[0];
      try{
         IPAddress[] addrs = Dns.Resolve(name).AddressList;
         foreach(IPAddress addr in addrs)
            Console.WriteLine("{0}/{1}",name,addr);
      }catch(Exception e){
         Console.WriteLine(e.Message);
      }
   }
}
```

Dns.GetHostName() returns the name of the local machine and Dns.Resolve() returns IPHostEntry for a machine with a given name, the AddressList property of which returns the IPAdresses of the machine. The Resolve method will cause an exception if the mentioned host is not found.

Though IPAddress allows to identify machines in the network, each machine may host multiple applications which use network for data exchange. Under TCP/IP, each network oriented application binds itself to a unique 2 byte integer referred to as its port-number which identifies this application on the machine it is executing. The data transfer takes place in the form of byte bundles called *IP Packets* or *Datagrams*. The size of each datagram is 64 KByte and it contains the data to be transferred, the actual size of the data, IP addresses and port-numbers of sender and the prospective receiver. Once a datagram is placed on a network by a machine, it will be received physically by all the other machines but will be accepted only by that machine whose IP address matches with the receiver's IP address in the packet. Later on, this machine will transfer the packet to an application running on it which is bound to the receiver's port-number present in the packet.

TCP/IP suite actually offers two different protocols for data exchange. The *Transmission Control Protocol* (TCP) is a reliable connection oriented protocol while the *User Datagram Protocol* (UDP) is not very reliable (but fast) connectionless protocol.

EXERCISE

Implementation of Client-Server programming with TCP/IP:

Under TCP there is a clear distinction between the *server process* and the *client process*. The server process starts on a well known port (which the clients are aware of) and listens for incoming connection requests. The client process starts on any port and issues a connection request.

The basic steps to create a TCP/IP server are as follows:

1. Create a System.Net.Sockets.TcpListener with a given local port and start it:

2.
   ```
   TcpListener listener = new TcpListener(local_port);
   listener.Start();
   ```

3. Wait for the incoming connection request and accept a System.Net.Sockets.Socket object from the listener whenever the request appears:

   ```
   Socket soc = listener.AcceptSocket(); // blocks
   ```

4. Create a System.Net.Sockets.NetworkStream from the above Socket:

   ```
   Stream s = new NetworkStream(soc);
   ```

5. Communicate with the client using the predefined protocol (*well established rules for data exchange*):
6. Close the Stream:

   ```
   s.Close();
   ```

7. Close the Socket:

   ```
   s.Close();
   ```

8. Go to **Step 2**.

Note when one request is accepted through step 2 no other request will be accepted until the code reaches step 7. (Requests will be placed in a queue or *backlog.*) In order to accept and service more than one client concurrently, steps 2 – 7 must be executed in multiple threads. Program below (*emptcpserver.cs*) is a multithreaded TCP/IP server which accepts employee name from its client and sends back the job of the employee. The client terminates the session by sending a blank line for the employee's name. The employee data is retrieved from the application's configuration file (an XML file in the directory of the application and whose name is the name of the application with a *.config* extension).

```
using System;
using System.Threading;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Configuration;

class EmployeeTCPServer{
   static TcpListener listener;
   const int LIMIT = 5; //5 concurrent clients

   public static void Main(){
      listener = new TcpListener(2055);
      listener.Start();
      #if LOG
         Console.WriteLine("Server mounted,
                     listening to port 2055");
      #endif
      for(int i = 0;i < LIMIT;i++){
         Thread t = new Thread(new ThreadStart(Service));
         t.Start();
      }
   }
   public static void Service(){
      while(true){
         Socket soc = listener.AcceptSocket();
         //soc.SetSocketOption(SocketOptionLevel.Socket,
         //      SocketOptionName.ReceiveTimeout,10000);
         #if LOG
            Console.WriteLine("Connected: {0}",
                              soc.RemoteEndPoint);
         #endif
         try{
            Stream s = new NetworkStream(soc);
            StreamReader sr = new StreamReader(s);
            StreamWriter sw = new StreamWriter(s);
            sw.AutoFlush = true; // enable automatic flushing
            sw.WriteLine("{0} Employees available",
                 ConfigurationSettings.AppSettings.Count);
            while(true){
               string name = sr.ReadLine();
               if(name == "" || name == null) break;
               string job =
                  ConfigurationSettings.AppSettings[name];
               if(job == null) job = "No such employee";
               sw.WriteLine(job);
            }
            s.Close();
         }catch(Exception e){
            #if LOG
               Console.WriteLine(e.Message);
            #endif
         }
         #if LOG
            Console.WriteLine("Disconnected: {0}",
                              soc.RemoteEndPoint);
         #endif
         soc.Close();
      }
   }
}
```

Here is the content of the configuration file (*emptcpserver.exe.config*) for the above application:

```
<configuration>
   <appSettings>
      <add key = "john" value="manager"/>
      <add key = "jane" value="steno"/>
      <add key = "jim" value="clerk"/>
      <add key = "jack" value="salesman"/>
   </appSettings>
</configuration>
```

The code between #if LOG and #endif will be added by the compiler only if the symbol LOG is defined during compilation (conditional compilation). You can compile the above program either by defining the LOG symbol (information is logged on the screen):

- *csc /D:LOG emptcpserver.cs*

or without the LOG symbol (silent mode):

- *csc emptcpserver.cs*

Mount the server using the command start emptcpserver.

To test the server you can use: telnet localhost 2055.

Or, we can create a client program. Basic steps for creating a TCP/IP client are as follows:

1. Create a System.Net.Sockets.TcpClient using the server's host name and port:

   ```
   TcpClient client = new TcpClient(host, port);
   ```

2. Obtain the stream from the above TCPClient.

   ```
   Stream s = client.GetStream()
   ```

3. Communicate with the server using the predefined protocol.
4. Close the Stream:

   ```
   s.Close();
   ```

5. Close the connection:

   ```
   client.Close();
   ```

The program below (*emptcpclient.cs*) communicates with EmployeeTCPServer:

```
using System;
using System.IO;
using System.Net.Sockets;

class EmployeeTCPClient{
    public static void Main(string[] args){
        TcpClient client = new TcpClient(args[0],2055);
        try{
            Stream s = client.GetStream();
            StreamReader sr = new StreamReader(s);
            StreamWriter sw = new StreamWriter(s);
            sw.AutoFlush = true;
            Console.WriteLine(sr.ReadLine());
            while(true){
                Console.Write("Name: ");
                string name = Console.ReadLine();
                sw.WriteLine(name);
                if(name == "") break;
                Console.WriteLine(sr.ReadLine());
            }
            s.Close();
        }finally{
            // code in finally block is guranteed
```

```
            // to execute irrespective of
            // whether any exception occurs or does
            // not occur in the try block
            client.Close();
        }
    }
}
```

LAB 9

LAMPORT’S LOGICAL CLOCK

Experiment:

To simulate the functioning of Lamport’s Logical clock

Assumption:

- The execution of a process is characterized by a sequence of events. An event can be the execution of one instruction or of one procedure.
- Sending a message is one event, receiving a message is one event.
- The events in a distributed system are not total chaos. Under some conditions, it is possible to ascertain the order of the events. Lamport's logical clocks try to catch this.

Lamport's `happened before'' relation : The ``happened before'' relation (®) is defined as follows:

A ® B if A and B are within the same process (same sequential thread of control) and A occurred before B.

A ® B if A is the event of sending a message M in one process and B is the event of receiving M by another process if A ® B and B ® C then A ® C

Event A causally affects event B iff A ® B.

Distinct events A and B are concurrent (A | | B) if we do not have A ® B or B ® A.

Exercise:

To simulate the functioning of Lamport’s Logical clock in ‘C’

Code:

```
#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

void main()

{

int i,j,k; int x=0;
```

```
char a[10][10];

int n,num[10],b[10][10]; clrscr();

printf("Enter the no. of physical clocks: "); scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("\nNo. of nodes for physical clock %d",i+1); scanf("%d",&num[i]); x=0;

for(j=0;j<num[i];j++)

{

printf("\nEnter the name of process: "); scanf("%s",&a[i][j]);

b[i][j]=x + rand() % 10; x=b[i][j]+1;

}

}

printf("\nPress a key for watching timestamp of physical clocks"); getch(); clrscr();

for(i=0;i<n;i++)

{

printf("Physical Clock %d",i+1); for(j=0;j<num[i];j++)

{

printf("\nProcess %c",a[i][j]); printf(" has P.T. :%d ",b[i][j]);

printf("\n");

}

}

printf("Press a key for watching timestamp of logical clocks"); getch(); clrscr();

x=0;
```

```
for(i=0;i<10;i++)
for(j=0;j<n;j++)
for(k=0;k<num[j];k++)
if(b[j][k]==i)
{
x = rand() % 10 + x;
printf("Logical Clock Timestamp for process %c",a[j][k]); printf(":%d ",x); printf("\n");
}
getch();
return;
}
```

Expected Output:

```
Enter the no. of physical clocks: 2
No. of nodes for physical clock 1: 2
Enter the name of process: a
Enter the name of process: b
No. of nodes for physical clock 2: 2
Enter the name of process: c
Enter the name of process: d
Press a key for watching timestamp of physical clocks
Physical Clock 1
Process a has P.T.: 6
Process b has P.T.: 7
```

```
Physical Clock 2

Process c has P.T.: 2

Process d has P.T.: 3

Press a key for watching timestamp of logical clocks

Logical Clock Timestamp for process a: 6

Logical Clock Timestamp for process b: 13

Logical Clock Timestamp for process c: 18

Logical Clock Timestamp for process d: 23
```

Applications: With a limitation of using for transitive dependency, it can be used to decide causal ordering for any two events in distributed systems.

LAB 10

LAMPORT’S VECTOR CLOCK

OBJECTIVE:
To simulate the functioning of Lamport’s Vector Clock Algorithm

OUTCOME: A vector clock is an algorithm for generating a partial ordering of events in a distributed system and detecting causality violations. Just as in Lamport timestamps, interprocess messages contain the state of the sending process's logical clock. A vector clock of a system of N processes is an array/vector of N logical clocks, one clock per process; a local "smallest possible values" copy of the global clock-array is kept in each process.

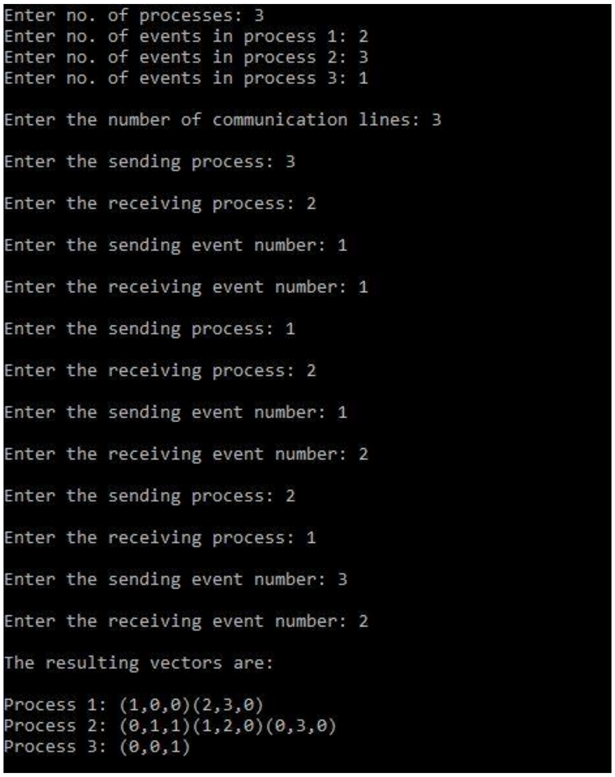
Code:

```
#include<iostream>
                    #include<conio.h>
                    #define SIZE 10
                    using namespace std;
                    class node {
                    public:
                    int data[SIZE];
                    node *next;
                    node() {
                    for(int p=0; p<SIZE; p++) {
                    data[p] = 0;
                    }
                    next = NULL;
                    }
                    node(int v[], int n1) {
                    for(int s = 0; s < n1; s++) {
                    data[s] = v[s];
                    }
                    next = NULL;
                    }
                    friend class process;
                    }*start=NULL;
                    int main() {
                    int n, events, sent, receive, sentE, recE, commLines = 0;
                    node *temp;
                    node *proc[SIZE]; //array of processes
                    cout<<"Enter no. of processes: ";
                    cin>>n;
                    int vector[n] = {0}; //representation of data
                    /*----------------INITIALIZATION LOOP-------------------------*/
                    for(int i = 0; i < n; i++) { //number of processes
                    for(int v = 0; v < n; v++) {
```

```
vector[v] = 0;
}
cout<<"Enter no. of events in process "<<i+1<<": ";
cin>>events;
for(int j = 1; j <= events; j++) {
vector[i] = j;
node *newnode = new node(vector,n);
if(start == NULL) {
start = newnode;
temp = start;
} else {
temp->next = newnode;
temp = temp->next;
}
}
proc[i] = start;
start = NULL;
}
/*-------------------DATA GATHERING--------------------*/
cout<<"\nEnter the number of communication lines: ";
cin>>commLines;
node *tempS, *tempR;
for(int i = 0; i < commLines; i++) {
cout<<"\nEnter the sending process: ";
cin>>sent;
cout<<"\nEnter the receiving process: ";
cin>>receive;
cout<<"\nEnter the sending event number: ";
cin>>sentE;
cout<<"\nEnter the receiving event number: ";
cin>>recE;
tempS = proc[sent - 1];
tempR = proc[receive - 1];
for(int j = 1; j < sentE; j++)
tempS = tempS->next;
for(int j = 1; j < recE; j++)
tempR = tempR->next;
for(int j = 0; j < n; j++) {
tempR->data[j] = (tempR->data[j] < tempS->data[j]) ? tempS->data[j]
: tempR->data[j];
}
}
/*-------------------DISPLAYING------------------------*/
cout<<"\nThe resulting vectors are:\n\n";
for(int k = 0; k < n; k++) {
cout<<"Process "<<k + 1<<": ";
```

```
node *temp1 = proc[k];
while(temp1) {
cout<<"(";
for(int f = 0; f < n - 1; f++)
cout<<temp1->data[f]<<",";
cout<<temp1->data[n-1];
cout<<")";
temp1 = temp1->next;
}
cout<<endl;
}
return 0;
}
```

OUTPUT:



```
Enter no. of processes: 3
Enter no. of events in process 1: 2
Enter no. of events in process 2: 3
Enter no. of events in process 3: 1

Enter the number of communication lines: 3

Enter the sending process: 3

Enter the receiving process: 2

Enter the sending event number: 1

Enter the receiving event number: 1

Enter the sending process: 1

Enter the receiving process: 2

Enter the sending event number: 1

Enter the receiving event number: 2

Enter the sending process: 2

Enter the receiving process: 1

Enter the sending event number: 3

Enter the receiving event number: 2

The resulting vectors are:

Process 1: (1,0,0)(2,3,0)
Process 2: (0,1,1)(1,2,0)(0,3,0)
Process 3: (0,0,1)
```

LAB 11

LAMPORT’S ALGORITHM FOR MUTUAL EXCLUSION

OBJECTIVE:

To simulate the functioning of Lamport's Algorithm for Mutual Exclusion

Theory:

Lamport's Distributed Mutual Exclusion Algorithm

Lamport's Distributed Mutual Exclusion Algorithm is a contention-based algorithm for mutual exclusion on a distributed system.

Nodal properties

1. Every process maintains a queue of pending requests for entering critical section order. The queues are ordered by virtual time stamps derived from Lamport timestamps.

Algorithm

Requesting process

1. Enters its request in its own queue (ordered by time stamps)

2. Sends a request to every node.

3. Wait for replies from all other nodes.

4. If own request is at the head of the queue and all replies have been received, enter critical section.

5. Upon exiting the critical section, send a release message to every process.

Other processes

1. After receiving a request, send a reply and enter the request in the request queue (ordered by time stamps)

2. After receiving release message, remove the corresponding request from the request queue.

3. If own request is at the head of the queue and all replies have been received, enter critical section.

Message complexity

This algorithm creates 3(N − 1) messages per request, or (N − 1) messages and 2 broadcasts.

Drawbacks

1. There exist multiple points of failure

Code:

```
#include<stdio.h>

#include<conio.h>

int main()

{

int i,d,p,a,c=0,aa[10],j,n;

char ch='y';

printf("enter no of processes");

scanf("%d",&n);

i=0;

do

{

printf("enter the process no which want to execute critical section");

scanf("%d",&a);

aa[i]=a;

i++;

c=c+1;

d=i;

printf("some other process want to execute cs? then press y");

fflush(0);

scanf("%c",&ch);

}

while(ch=='y');

for(j=1;j<=c;j++)

{

printf("\ncritical section is executing for process %d in queue......",j);
printf("\ncritical section is finished for process %d",j);

printf("\nrelease msg has sent by process%d",j);

return 0;

}

getch();

}
```

LAB 12
EDGE CHASING ALGORITHM

OBJECTIVE:

To implement edge chasing distributed deadlock detection algorithm.

EDGE CHASING ALGORITHM:

In edge chasing algorithm, a special message called probe is used in deadlock detection. A probe is a triplet (i, j, k) which denotes that process P_i has initiated the deadlock detection and the message is being sent by the home site of process P_j to the home site of process P_k

EXERCISE:

To implement edge chasing distributed deadlock detection algorithm.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int temp,process[10][15], site_count=0, process_count=0,i,j,k,waiting[15];
int p1,p2,p3;
clrscr();
printf("\n Enter the no. of sites (max 3) \n");
scanf("%d",&site_count);
for(i=1;i<=site_count;i++)
{
printf("\n Enter the no. of processes in %d site ( max 4)\n",i);
scanf("%d",&process_count);
for(j=0;j<process_count;j++)
{
process[i][j]=i+(i*j);
}
}
printf("\n Enter the blocked process \n");
scanf("%d",&k);
```

```
for(i=1;i<=3;i++)
{
for(j=0;j<=3;j++)
{
if(k==process[i][j])
{
printf("Process %d is at site %d ",k,i);
temp=i;
}
if(k==process[i][j])
printf("It is a deadlock \n");
if(k==(process[temp][j])&&((process[temp][j])==waiting[process[i][j]]) && (temp!=i))
{
//probe(temp,j,process[i][j]);
if(process[i][j]==waiting[process[temp][j]]);
printf("It is a deadlock\n");
}
}
}getch();
```

LAB 13
SQL INJECTION

OBJECTIVE:
To study SQL Injection and prevention techniques.

SQL INJECTION:

An SQL injection, which is also referred to as a **“failure to preserve SQL Query Structure”,** is a common and dangerous security issue. SQL injections are dangerous because they allow hackers to compromise your system through your web interface, letting them wreak havoc at will – i.e., modify databases, delete tables, and control your corporate network.

The factors of SQL Injection includes,

- Weak input validation.
- Dynamic construction of SQL statements without the use of type-safe parameters.
- Use of over-privileged database logins.

A set of simple techniques for preventing SQL Injection vulnerabilities by avoiding these problems. These techniques can be used with practically any kind of programming language with any type of database. There are other types of databases, like XML databases, which can have similar problems (e.g., XPath and XQuery injection) and these techniques can be used to protect them as well.

➢ *Primary Defenses:*

- Use of Prepared Statements (Parameterized Queries)
- Use of Stored Procedures
- Escaping all User Supplied Input

➢ *Additional Defenses:*

- Also Enforce: Least Privilege
- Also Perform: White List Input Validation

Unsafe Example

The following (Java) example is UNSAFE, and would allow an attacker to inject code into the query that would be executed by the database. The unvalidated “customerName” parameter that is simply appended to the query allows an attacker to inject any SQL code they want. Unfortunately, this method for accessing databases is all too common.

```
String query = "SELECT account_balance FROM user_data WHERE user_name = "
   + request.getParameter("customerName");

 try {
        Statement statement = connection.createStatement( … );
        ResultSet results = statement.executeQuery( query );
}
```

Primary Defenses

Prepared Statements (Parameterized Queries)

Parameterized queries force the developer to first define all the SQL code, and then pass in each parameter to the query later. This coding style allows the database to distinguish between code and data, regardless of what user input is supplied.

Prepared statements ensure that an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker. In the safe example below, if an attacker were to enter the userID of tom' or '1'='1, the parameterized query would not be vulnerable and would instead look for a username which literally matched the entire string tom' or '1'='1.

Language specific recommendations:

- ✓ Java EE – use PreparedStatement() with bind variables
- ✓ .NET – use parameterized queries like SqlCommand() or OleDbCommand() with bind variables
- ✓ PHP – use PDO with strongly typed parameterized queries (using bindParam())
- ✓ Hibernate - use createQuery() with bind variables (called named parameters in Hibernate)
- ✓ SQLite - use sqlite3_prepare() to create a statement object

In rare circumstances, prepared statements can harm performance. When confronted with this situation, it is best to escape all user supplied input using an escaping routine specific to your database vendor, rather than using a prepared statement. Another option which might solve your performance issue is used a stored procedure instead.

Safe Java Prepared Statement Example

The following code example uses a PreparedStatement, Java's implementation of a parameterized query, to execute the same database query.

```
String custname = request.getParameter("customerName"); // This should REALLY be validated too
 // perform input validation to detect attacks
 String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";

 PreparedStatement pstmt = connection.prepareStatement( query );
 pstmt.setString( 1, custname);
 ResultSet results = pstmt.executeQuery( );
```

Stored Procedures

Stored procedures have the same effect as the use of prepared statements when implemented safely. They require the developer to define the SQL code first, and then pass in the parameters after.

The difference between prepared statements and stored procedures is that the SQL code for a stored procedure is defined and stored in the database itself, and then called from the application. Both of these techniques have the same effectiveness in preventing SQL injection.

There are also several cases where stored procedures can increase risk. For example, on MS SQL server, you have 3 main default roles: db_datareader, db_datawriter and db_owner. Before stored procedures came into use, DBA's would give db_datareader or db_datawriter rights to the webservice's user, depending on the requirements. However, stored procedures require execute rights, a role that is not available by default. Some setups where the user management has been centralized, but is limited to those 3 roles, cause all web apps to run under db_owner rights so stored procedures can work. Naturally, that means that if a server is breached the attacker has full rights to the database, where previously they might only have had read-access.

Safe Java Stored Procedure Example

The following code example uses a CallableStatement, Java's implementation of the stored procedure interface, to execute the same database query. The "sp_getAccountBalance" stored procedure would have to be predefined in the database and implement the same functionality as the query defined above.

```
String custname = request.getParameter("customerName"); // This should REALLY be validated
 try {
        CallableStatement cs = connection.prepareCall("{call sp_getAccountBalance(?)}");
        cs.setString(1, custname);
        ResultSet results = cs.executeQuery();
        // … result set handling
 } catch (SQLException se) {
        // … logging and error handling}
```

Escaping All User Supplied Input

This third technique is to escape user input before putting it in a query. If you are concerned that rewriting your dynamic queries as prepared statements or stored procedures might break your application or adversely affect performance, then this might be the best approach for you. However, this methodology is frail compared to using parameterized queries.

This technique works like this. Each DBMS supports one or more character escaping schemes specific to certain kinds of queries. If you then escape all user supplied input using the proper escaping scheme for the database you are using, the DBMS will not confuse that input with SQL code written by the developer, thus avoiding any possible SQL injection vulnerabilities.

Escaping Dynamic Queries

An Oracle example

```
ESAPI.encoder().encodeForSQL( new OracleCodec(), queryparam );
```

So, existing Dynamic query being generated in your code that was going to Oracle

```
String query = "SELECT user_id FROM user_data WHERE user_name = '" + req.getParameter("userID")
 + "' and user_password = '" + req.getParameter("pwd") +"'";
 try {
     Statement statement = connection.createStatement( … );
     ResultSet results = statement.executeQuery( query );
 }
```

Turn off character replacement

Use SET DEFINE OFF or SET SCAN OFF to ensure that automatic character replacement is turned off. If this character replacement is turned on, the & character will be treated like a SQLPlus variable prefix that could allow an attacker to retrieve private data.

Escaping Wildcard characters in Like Clauses

The LIKE keyword allows for text scanning searches. In Oracle, the underscore '_' character matches only one character, while the ampersand '%' is used to match zero or more occurrences of any characters. These characters must be escaped in LIKE clause criteria. For example:

```
SELECT name FROM emp
WHERE id LIKE '%/_%' ESCAPE '/';
```

MySQL Escaping

MySQL supports two escaping modes:

1. ANSI_QUOTES SQL mode, and a mode with this off, which we call
2. MySQL mode.

Additional Defenses

Beyond adopting one of the three primary defenses, additional defenses are:

- **Least Privilege**
- **White List Input Validation**

Least Privilege

To minimize the potential damage of a successful SQL injection attack, you should minimize the privileges assigned to every database account in your environment. Do not assign DBA or admin type access rights to your application accounts.

SQL injection is not the only threat to your database data. Attackers can simply change the parameter values from one of the legal values they are presented with, to a value that is unauthorized for them, but the application itself might be authorized to access. As such, minimizing the privileges granted to your application will reduce the likelihood of such unauthorized access attempts, even when an attacker is not trying to use SQL injection as part of their exploit.

While you are at it, you should minimize the privileges of the operating system account that the DBMS runs under. Don't run your DBMS as root or system! Most DBMSs run out of the box with a very powerful system account. **For example**, MySQL runs as system on Windows by default! Change the DBMS's OS account to something more appropriate, with restricted privileges.

White List Input Validation

Input validation can be used to detect unauthorized input before it is passed to the SQL query.

"You can never stop hackers to hack something, you can just make his task harder by putting some extra security"

LAB 14
OPEN ENDED LAB

PROBLEM DEFINITION:

The students are required to build a distributed data structure where 2-3 databases are merged together using linked server.

The linked server technique should be different from the ones already discussed in class.

Students are required to develop and entire ER-Diagram and design the workflow of their distributed database using an appropriate example.

INSTRUCTIONS:

1. Students will lose marks if they opt for similar tools as discussed in laboratory sessions.

2. Students must highlight the linked server technique being used.

3. Students are required to explain the distributed database workflow with an appropriate example.

4. Students will lose marks if they come up with clichéd examples.

5. The distributed database workflow, example and all connections must be provided in a report format

6. The OEL will be assessed based on the following rubrics.