Seminary 6

# Practical Exam

- Solution -

# Practical Exam

Create a database for a MiniFacebook system.

- ✓ The entities of interest to the problem domain are: *Users, Pages, Likes, Categories, Posts, and Comments.*
- ✓ *Each user has a name, a current city and date of birth.*
- ✓ *A user can like multiple pages.*
- ✓ *The system stores the date of each like.*
- ✓ *A page has a name and a category, e.g., sports, movies, music, etc.*
- ✓ *A category also has a category description.*
- ✓ *Users write posts and comment on existing posts.*
- ✓ *A user's post has a date, text, and number of shares.*
- ✓ *A comment is anonymous, has a text, a date, and a flag indicating whether it's a top comment for the corresponding post.*

# Practical Exam

- 1) Write an SQL script that creates the corresponding relational data model. (3p)
- 2) Create a master – detail windows form that allows searching the posts for a given user. The form should have at least the following components: a grid (combo box or list box) to display the users, a grid for all the posts of the selected user, and a button for saving added / deleted / modified posts. It's up to you how to add / delete / change a post. (4p)
- 3) Create a scenario that reproduces the non-repeatable reads concurrency issue, through stored procedures or standalone queries. Find a solution to solve / workaround this concurrency issue. (2p)

(1p of)

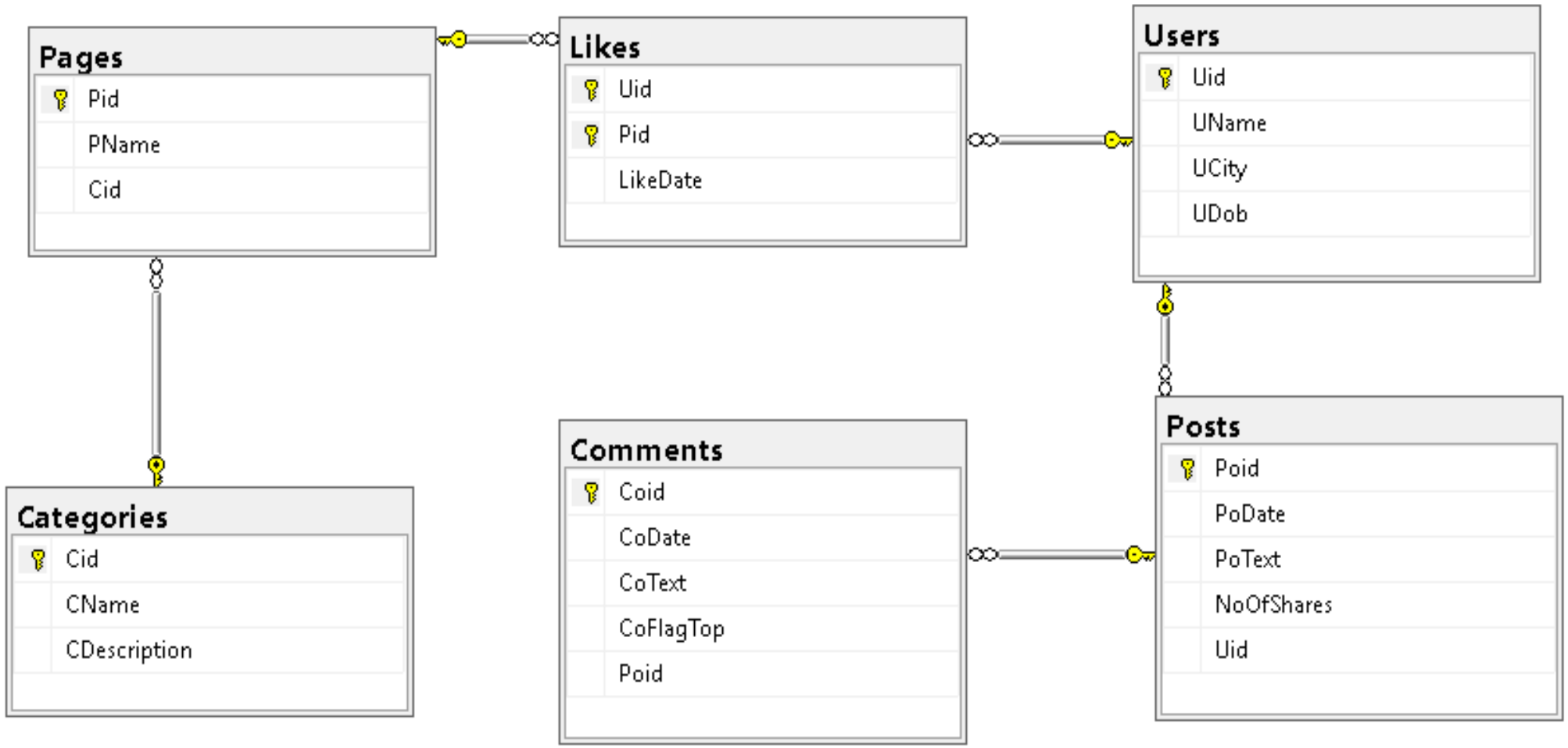
Solution :

# Practical Exam

```
create database MiniFacebook_IE
go
use MiniFacebook_IE
go

create table Users(
Uid int primary key identity(1,1),
UName VARCHAR(50),
UCity VARCHAR(50),
UDob Date
)

create table Categories(
Cid int primary key identity(1,1),
CName VARCHAR(50),
CDescription varchar(50)
)

create table Pages(
Pid int primary key identity(1,1),
PName VARCHAR(50),
Cid int FOREIGN KEY REFERENCES Categories(Cid)
)
```

```
create table Likes(
Uid int references Users(Uid),
Pid int references Pages(Pid),
LikeDate date,
CONSTRAINT pk_Like PRIMARY KEY(Uid, Pid)
)

create table Posts(
Poid int primary key identity(1,1),
PoDate date,
PoText VARCHAR(50),
NoOfShares int,
Uid int FOREIGN KEY REFERENCES Users(Uid)
)

create table Comments(
Coid int primary key identity(1,1),
CoDate date,
CoText varchar(50),
CoFlagTop bit,
Poid int FOREIGN KEY REFERENCES Posts(Poid)
)
```

# Practical Exam



# Practical Exam

- INSERT Users VALUES('User 1', 'Cluj-Napoca', '10/10/2000'), ('User 2', 'Bucuresti', '2/2/1998')
- INSERT Categories VALUES('Category 1', 'very interesting'), ('Category 2', 'interesting')
- INSERT Pages VALUES('Page 1', 1), ('Page 2', 1)
- INSERT Likes VALUES(1,1,'2/21/2018'), (1,2,'6/7/2018'), (2,1, '2/3/2017')
- INSERT Posts VALUES ('1/1/2018', 'Post 1', 3, 1), ('3/23/2018', 'Post 2',5, 2)
- insert Comments values ('6/8/2019','comment 1', 1,1), ('6/9/2017', 'commmmmmm', 0,2)
- GO

- select * from Users
- select * from Categories
- select * from Pages
- select * from Likes
- select * from Posts
- select * from Comments

# Practical Exam

Results | Messages

| | Uid | UName | UCity | UDob |
|---|---|---|---|---|
| 1 | 1 | User 1 | Cluj-Napoca | 2000-10-10 |
| 2 | 2 | User 2 | Bucuresti | 1998-02-02 |

| | Cid | CName | CDescription |
|---|---|---|---|
| 1 | 1 | Category 1 | very interesting |
| 2 | 2 | Category 2 | interesting |

| | Pid | PName | Cid |
|---|---|---|---|
| 1 | 1 | Page 1 | 1 |
| 2 | 2 | Page 2 | 1 |

| | Uid | Pid | LikeDate |
|---|---|---|---|
| 1 | 1 | 1 | 2018-02-21 |
| 2 | 1 | 2 | 2018-06-07 |
| 3 | 2 | 1 | 2017-02-03 |

| | Poid | PoDate | PoText | NoOfShares | Uid |
|---|---|---|---|---|---|
| 1 | 1 | 2018-01-01 | Post 1 | 3 | 1 |
| 2 | 2 | 2018-03-23 | Post 2 | 5 | 2 |

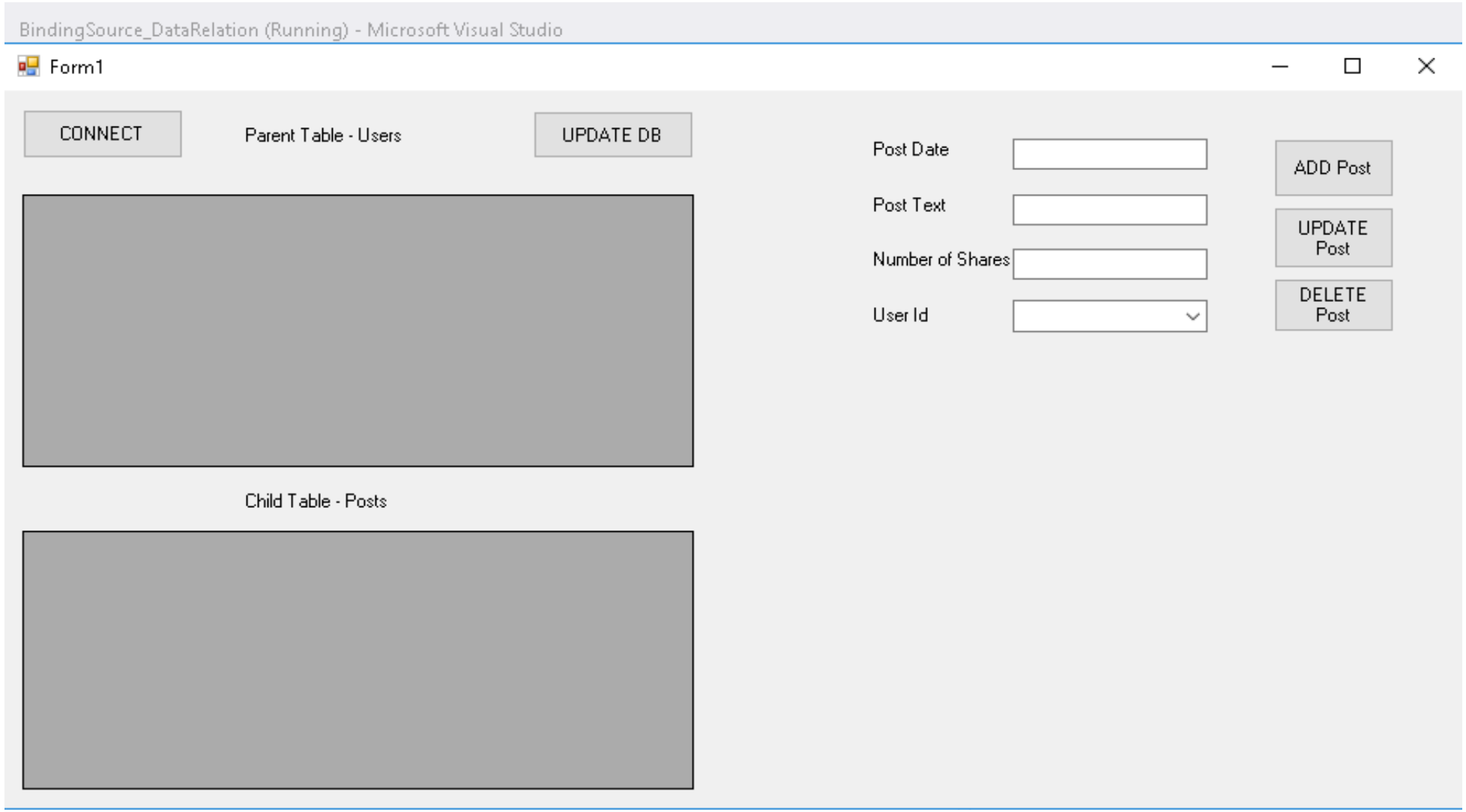
| | Coid | CoDate | CoText | CoFlagTop | Poid |
|---|---|---|---|---|---|
| 1 | 1 | 2019-06-08 | comment 1 | 1 | 1 |
| 2 | 2 | 2017-06-09 | commmmmmm | 0 | 2 |



Query executed successfully. | DESKTOP-ATJN5FL\SQLEXPRESS ... | DESKTOP-ATJN5FL\Emi (54) | MiniFacebook_IE | 00:00:00 | 13 rows

# Practical Exam

2. Aplicatie Windows Forms Application



# Practical Exam

***Connect to Database – Parent-Child relationship (the tables Users-Posts)***

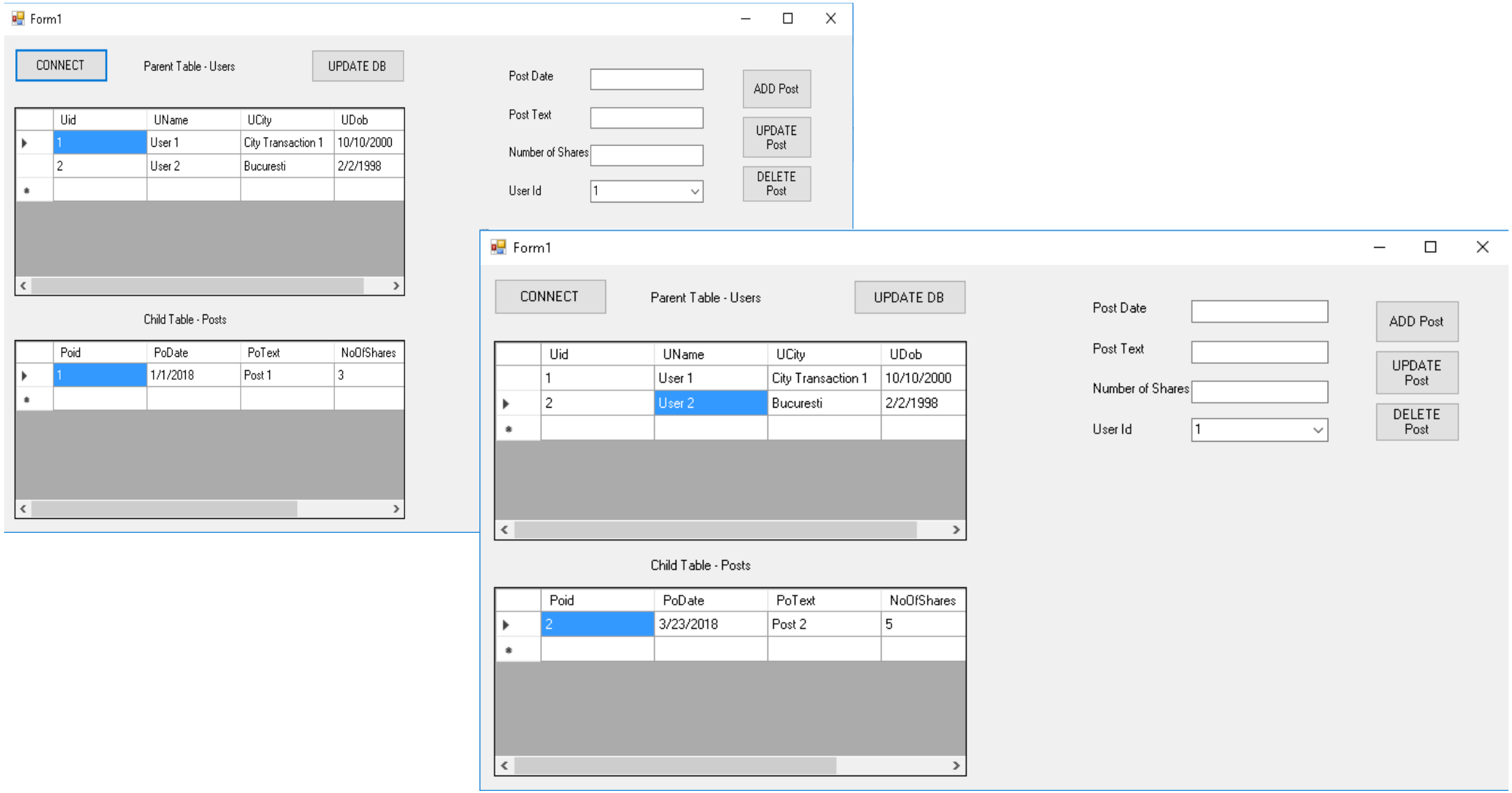
```
connection = new SqlConnection(@"Data Source = DESKTOP-ATJN5FL\SQLEXPRESS; Initial Catalog=MiniFacebook_IE;
    Integrated Security = True");
        ds = new DataSet();
        daChild = new SqlDataAdapter("select * from Posts", connection);
        daParent = new SqlDataAdapter("select * from Users", connection);
        cb = new SqlCommandBuilder(daChild);

        daParent.Fill(ds, "Users");
        daChild.Fill(ds, "Posts");

        DataRelation dr = new DataRelation("FK_Users_Posts", ds.Tables["Users"].Columns["Uid"],
    ds.Tables["Posts"].Columns["Uid"]);
        ds.Relations.Add(dr);
```

# Practical Exam

```
bsParent = new BindingSource();
bsChild = new BindingSource();

bsParent.DataSource = ds;
bsParent.DataMember = "Users";

bsChild.DataSource = bsParent;
bsChild.DataMember = "FK_Users_Posts";

GridChild.DataSource = bsChild;
GridParent.DataSource = bsParent;
```

Update Button:  daChild.Update(ds, "Posts");

# Practical Exam

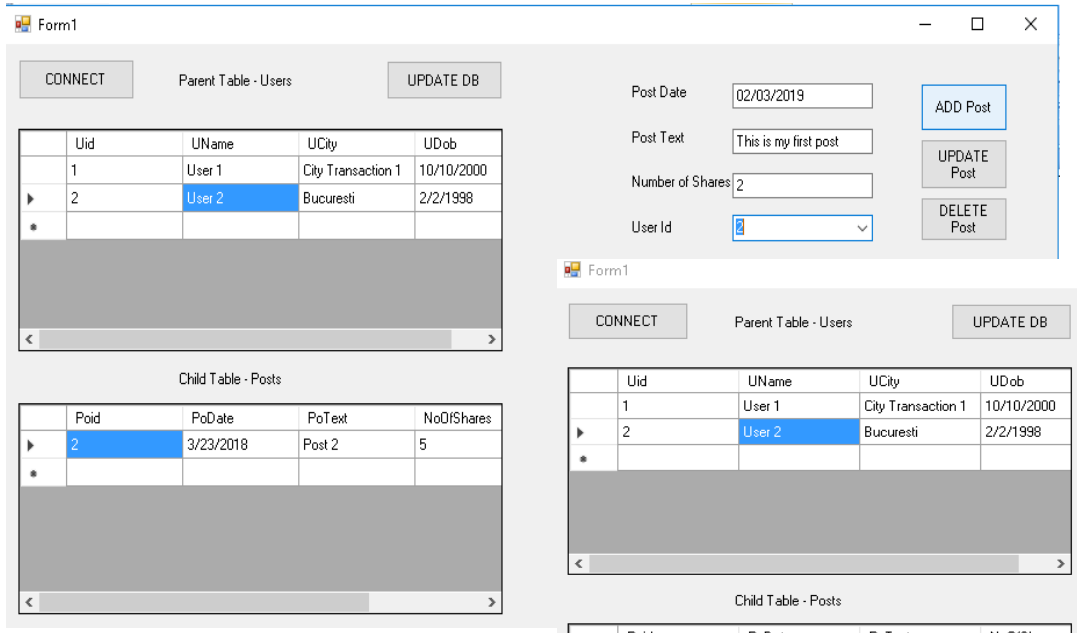


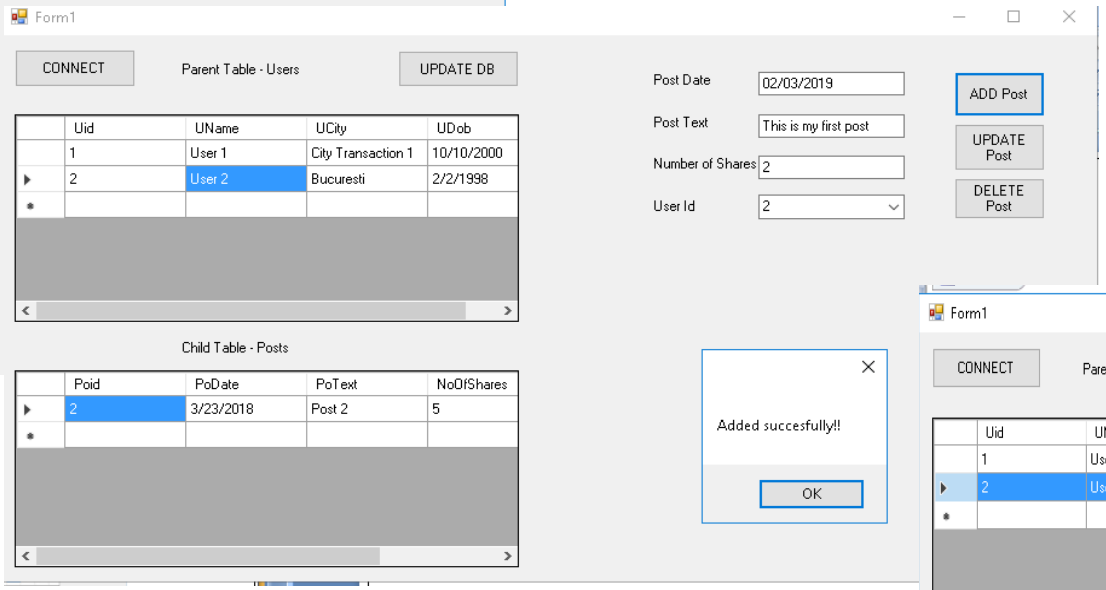
# Practical Exam

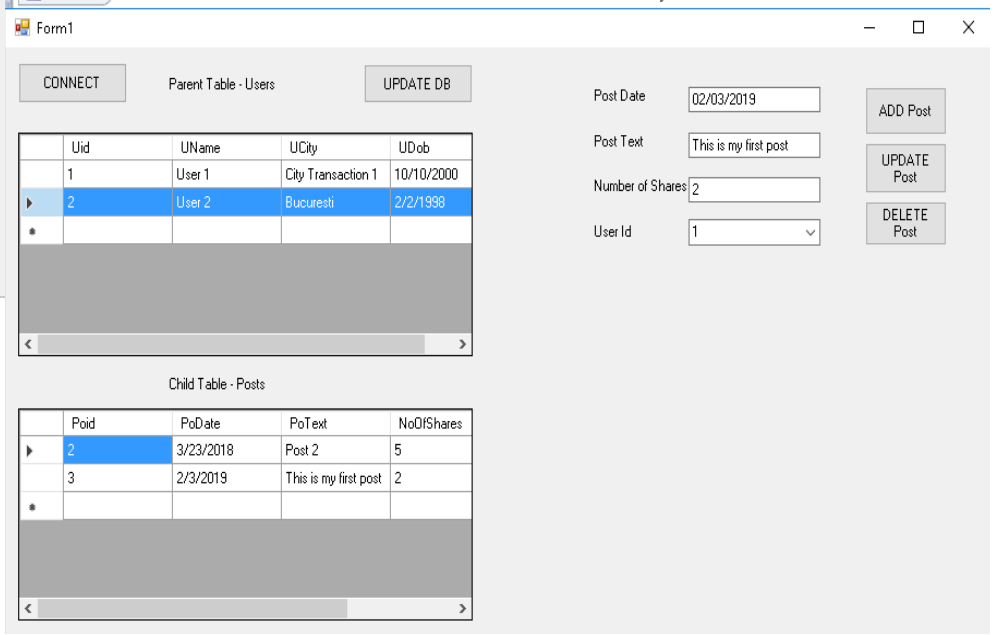
***INSERT – in the child table (Posts)***

```
daChild.InsertCommand = new SqlCommand("INSERT INTO Posts(PoDate, PoText, NoOfShares, Uid) VALUES (@d, @t, @n, @c)", connection);
daChild.InsertCommand.Parameters.Add("@d", SqlDbType.Date).Value = DateTime.Parse(textBox1.Text);
daChild.InsertCommand.Parameters.Add("@t", SqlDbType.VarChar).Value = textBox2.Text;
daChild.InsertCommand.Parameters.Add("@n", SqlDbType.Int).Value = Int32.Parse(textBox3.Text);
daChild.InsertCommand.Parameters.Add("@c", SqlDbType.Int).Value = Int32.Parse(comboBox1.Text);

connection.Open();
daChild.InsertCommand.ExecuteNonQuery();
connection.Close();

MessageBox.Show("Added succesfully!!");
```

# Practical Exam



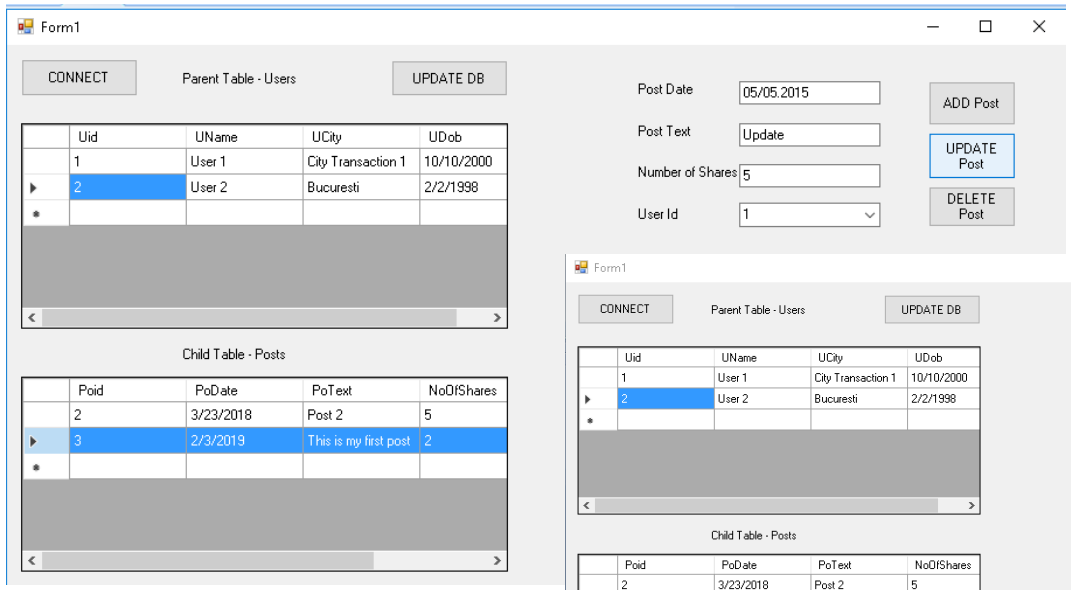



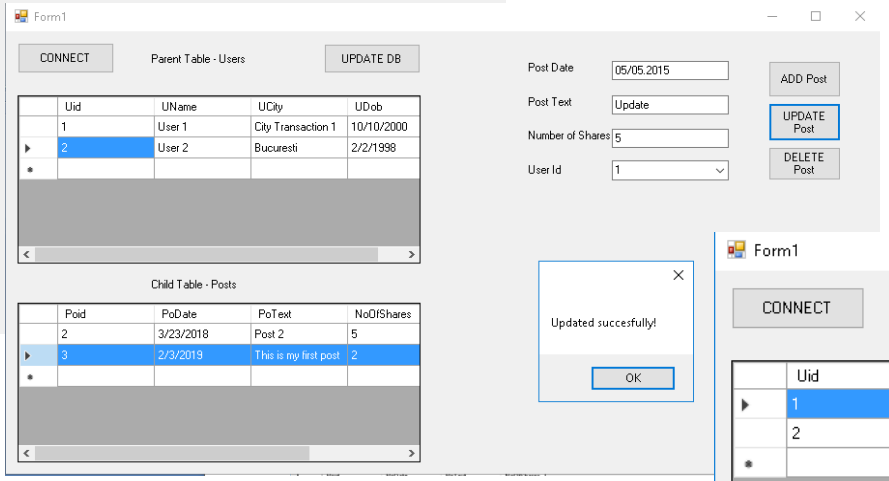
# Practical Exam

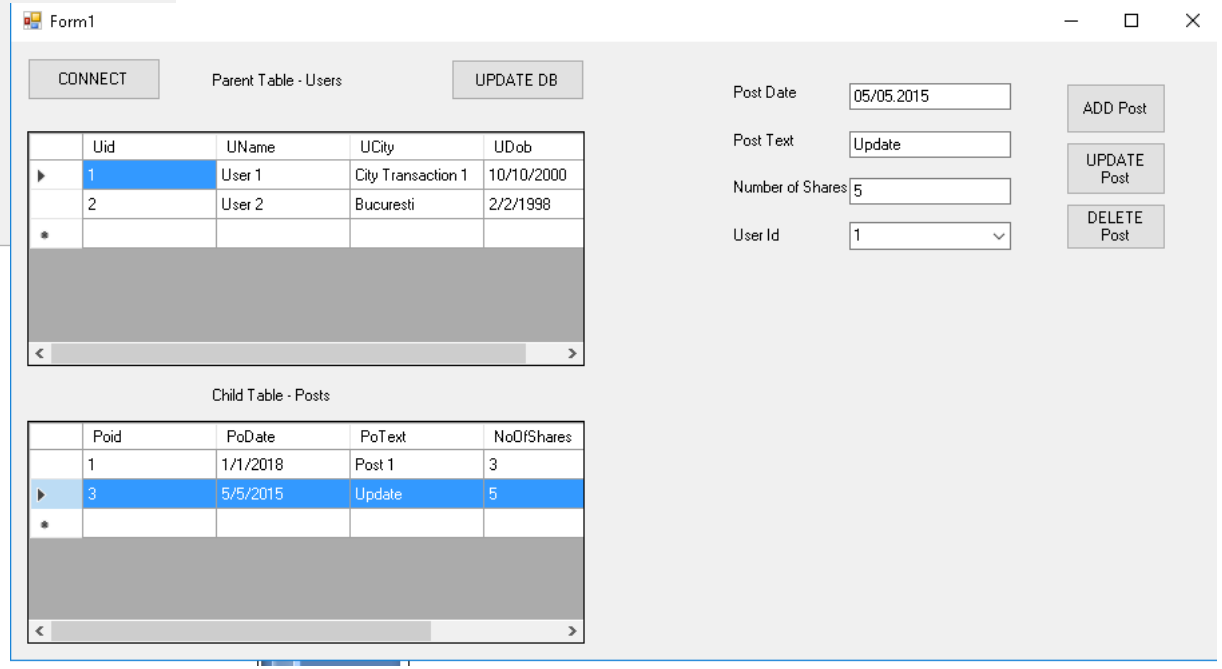
***UPDATE – for a selected child (a row from table Posts)***

```
var selectedRowP = Int32.Parse(GridChild.SelectedRows[0].Cells[0].Value.ToString());
daChild.UpdateCommand = new SqlCommand("UPDATE Posts SET PoDate=@d, PoText=@t,
NoOfShares=@n, Uid=@c WHERE Poid=@pid", connection);
daChild.UpdateCommand.Parameters.Add("@d", SqlDbType.Date).Value = DateTime.Parse(textBox1.Text);
daChild.UpdateCommand.Parameters.Add("@t", SqlDbType.VarChar).Value = textBox2.Text;
daChild.UpdateCommand.Parameters.Add("@n", SqlDbType.Int).Value = Int32.Parse(textBox3.Text);
daChild.UpdateCommand.Parameters.Add("@c", SqlDbType.Int).Value = Int32.Parse(comboBox1.Text);
daChild.UpdateCommand.Parameters.Add("@pid", SqlDbType.Int).Value = selectedRowP;

connection.Open();
daChild.UpdateCommand.ExecuteNonQuery();
connection.Close();

MessageBox.Show("Updated succesfully!");
```

# Practical Exam



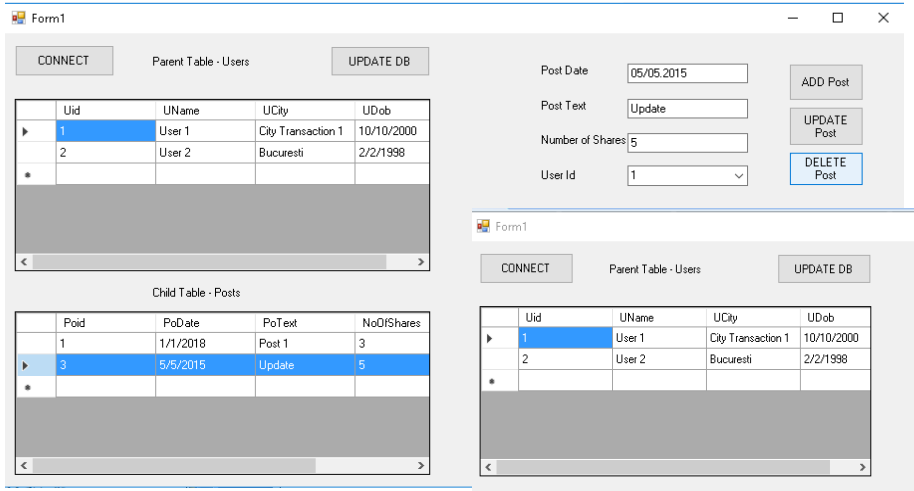



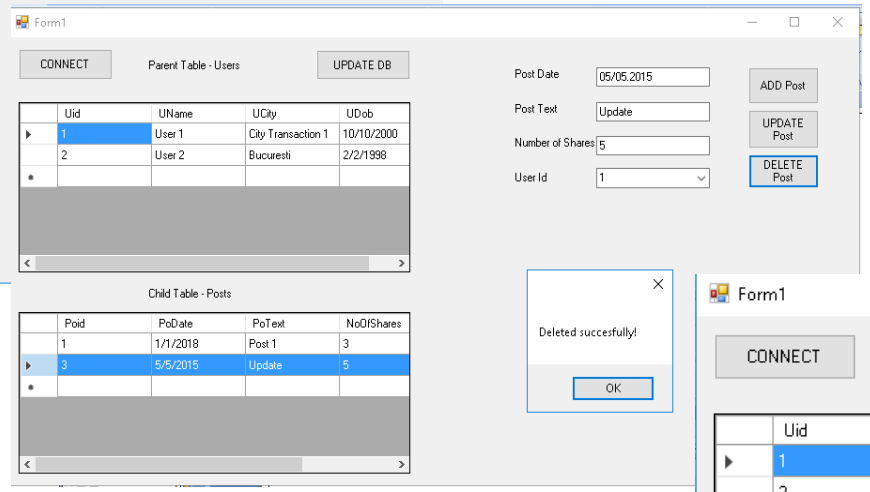
# Practical Exam

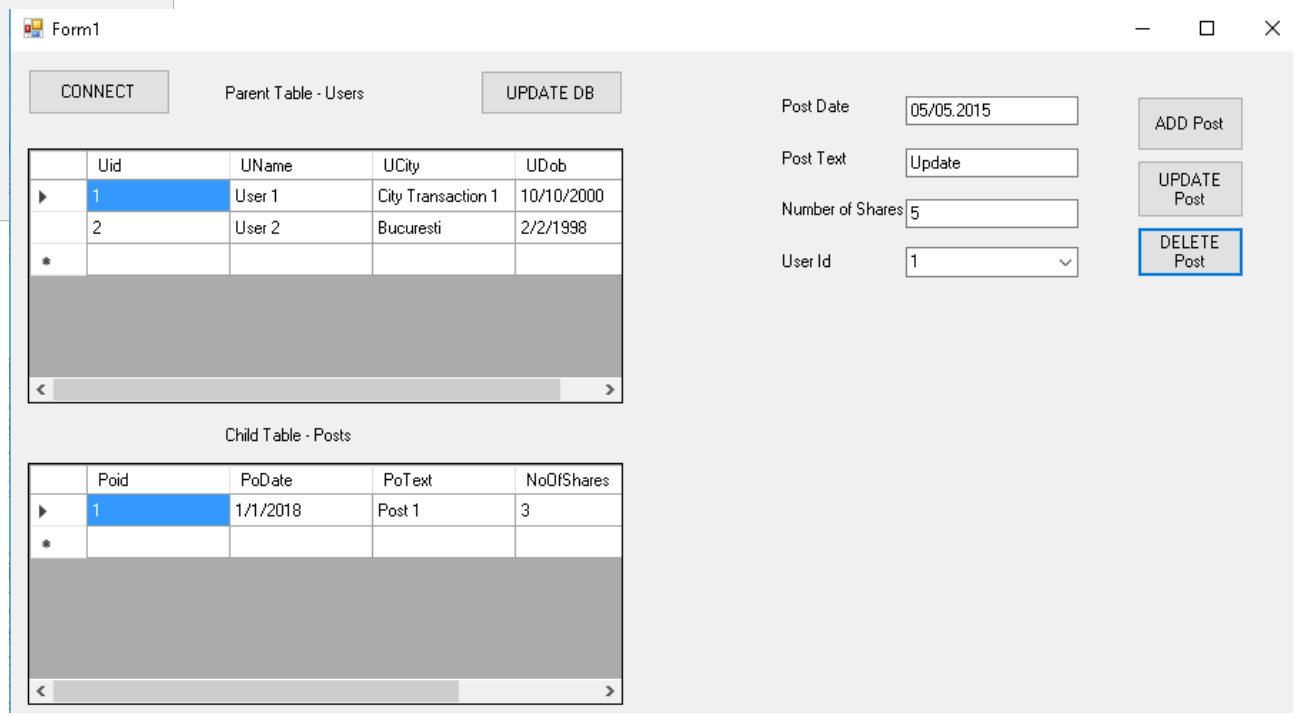
***DELETE – for a selected child (from the table Posts)***

```
var selectedRowP = Int32.Parse(GridChild.SelectedRows[0].Cells[0].Value.ToString());
daChild.DeleteCommand = new SqlCommand("DELETE FROM Posts WHERE Poid=@pid", connection);
daChild.DeleteCommand.Parameters.Add("@pid", SqlDbType.Int).Value = selectedRowP;

connection.Open();
daChild.DeleteCommand.ExecuteNonQuery();
connection.Close();

MessageBox.Show("Deleted succesfully!");
```

# Practical Exam

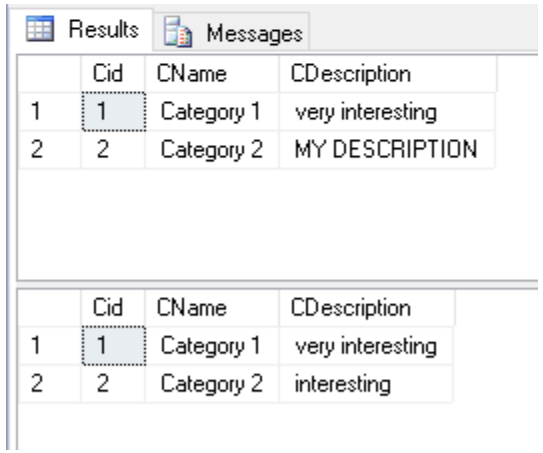


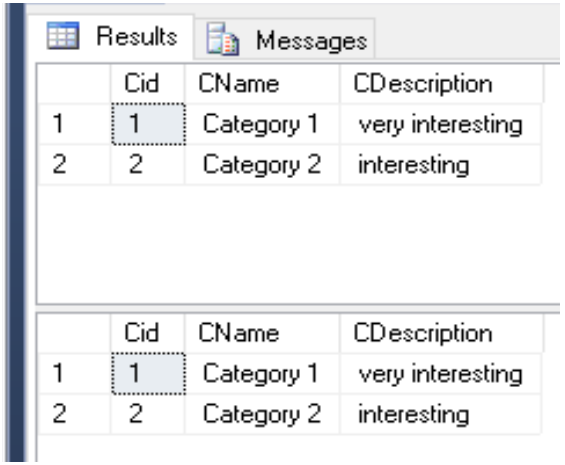
# Practical Exam

3. Dirty reads

| *Transaction 1* | *Transaction 2* |
| --- | --- |
| use MiniFacebook_IE<br>Go<br>BEGIN TRANSACTION<br>UPDATE Categories SET CDescription='MY DESCRIPTION'<br>WHERE Cid = 2<br>WAITFOR DELAY '00:00:10'<br>ROLLBACK TRANSACTION | use MiniFacebook_IE<br>Go<br>SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED<br>BEGIN TRAN<br>select * from Categories<br>WAITFOR DELAY '00:00:15'<br>select * from Categories<br>COMMIT TRAN |
| (1 row(s) affected) |  |

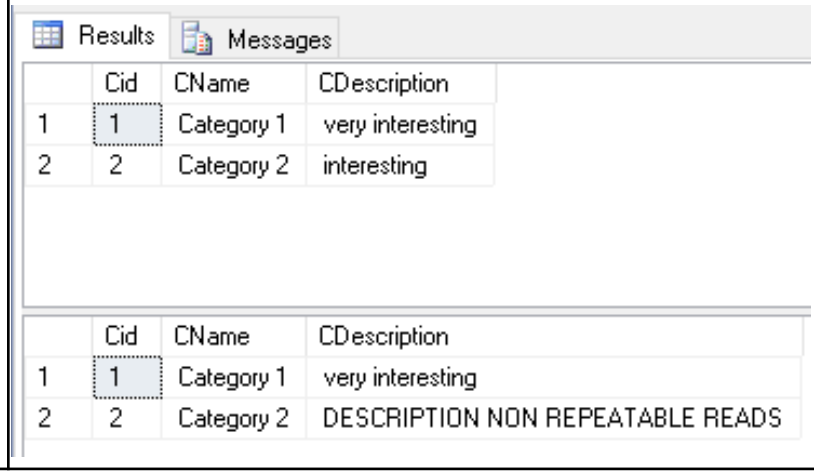
# Practical Exam

Dirty reads - Solution

| *Transaction 1* | *Transaction 2* |
|---|---|
| use MiniFacebook_IE<br>Go<br>BEGIN TRANSACTION<br>UPDATE Categories SET CDescription='MY DESCRIPTION' WHERE Cid = 2<br>WAITFOR DELAY '00:00:10'<br>ROLLBACK TRANSACTION | use MiniFacebook_IE<br>Go<br>--Solution: SET TRANSACTION ISOLATION LEVEL TO READ COMMITTED<br>SET TRANSACTION ISOLATION LEVEL READ COMMITTED<br>BEGIN TRAN<br>select * from Categories<br>WAITFOR DELAY '00:00:15'<br>select * from Categories<br>COMMIT TRAN |
| (1 row(s) affected) |  |

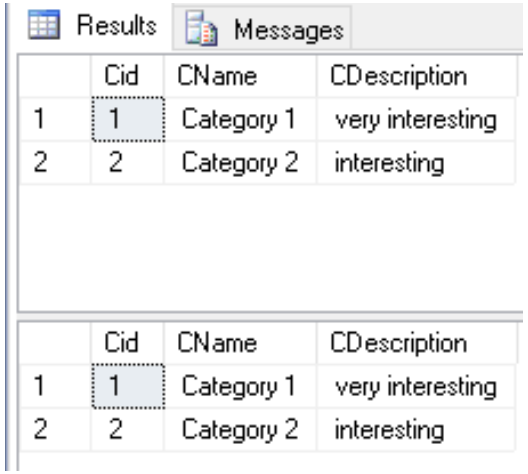
# Practical Exam

Non-repeatable reads

| *Transaction 1* | *Transaction 2* |
|---|---|
| use MiniFacebook_IE<br>Go<br>BEGIN TRAN<br>WAITFOR DELAY '00:00:05'<br>UPDATE Categories SET CDescription='DESCRIPTION NON REPEATABLE READS' WHERE Cid = 2<br>COMMIT TRAN | use MiniFacebook_IE<br>Go<br>SET TRANSACTION ISOLATION LEVEL READ COMMITTED<br>BEGIN TRAN<br>SELECT * FROM Categories<br>WAITFOR DELAY '00:00:05'<br>SELECT * FROM Categories<br>COMMIT TRAN |
| (1 row(s) affected) |  |

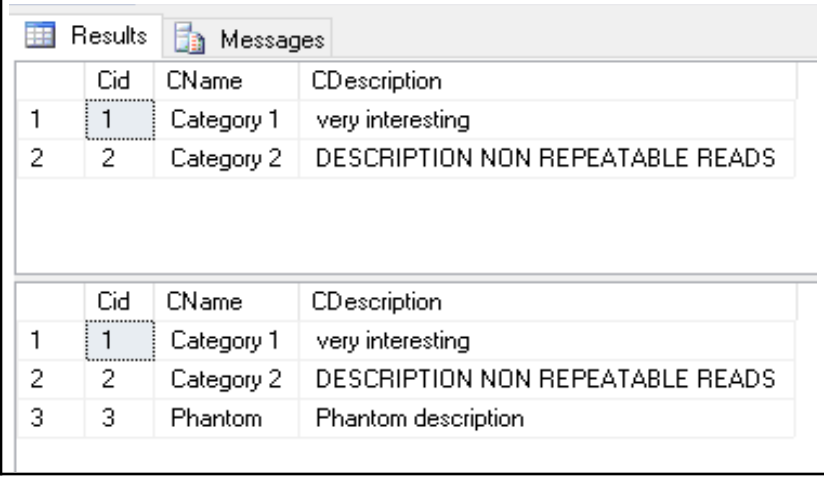
# Practical Exam

Non-repeatable reads - Solution

| *Transaction 1* | *Transaction 2* |
|---|---|
| use MiniFacebook_IE<br>Go<br>BEGIN TRAN<br>WAITFOR DELAY '00:00:05'<br>UPDATE Categories SET<br>CDescription='DESCRIPTION NON REPEATABLE READS' WHERE Cid = 2<br>COMMIT TRAN | use MiniFacebook_IE<br>Go<br>--Solution: SET TRANSACTION ISOLATION LEVEL TO REPEATABLE READ<br>SET TRANSACTION ISOLATION LEVEL REPEATABLE READ<br>BEGIN TRAN<br>SELECT * FROM Categories<br>WAITFOR DELAY '00:00:05'<br>SELECT * FROM Categories<br>COMMIT TRAN |
| (1 row(s) affected) |  |

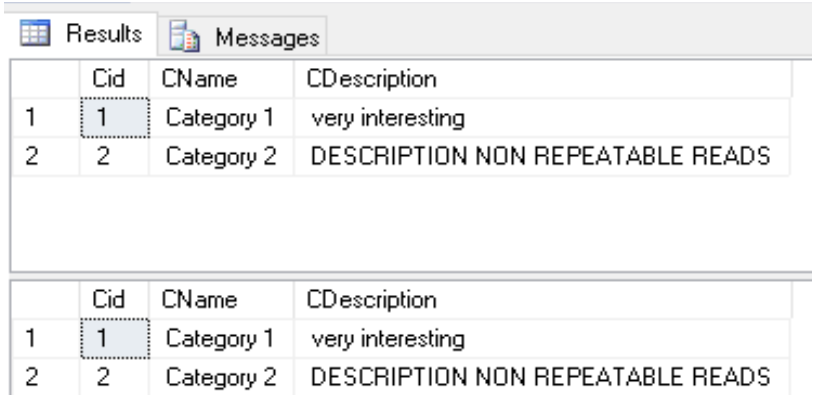
# Practical Exam

Phantom reads

| *Transaction 1* | *Transaction 2* |
|---|---|
| use MiniFacebook_IE<br>Go<br>BEGIN TRAN<br>WAITFOR DELAY '00:00:05'<br>INSERT INTO Categories(CName,CDescription)<br>VALUES ('Phantom','Phantom description')<br>COMMIT TRAN | use MiniFacebook_IE<br>go<br>SET TRANSACTION ISOLATION LEVEL REPEATABLE READ<br>BEGIN TRAN<br>SELECT * FROM Categories<br>WAITFOR DELAY '00:00:05'<br>SELECT * FROM Categories<br>COMMIT TRAN |
| (1 row(s) affected) |  |

# Practical Exam

Phantom reads - Solution

| *Transaction 1* | *Transaction 2* |
|---|---|
| use MiniFacebook_IE<br>Go<br>BEGIN TRAN<br>WAITFOR DELAY '00:00:05'<br>INSERT INTO Categories(CName,CDescription)<br>VALUES ('Phantom','Phantom description')<br>COMMIT TRAN | use MiniFacebook_IE<br>Go<br>--Solution: Set transaction isolation level to SERIALIZABLE<br>SET TRANSACTION ISOLATION LEVEL SERIALIZABLE<br>BEGIN TRAN<br>SELECT * FROM Categories<br>WAITFOR DELAY '00:00:05'<br>SELECT * FROM Categories<br>COMMIT TRAN |
| (1 row(s) affected) |  |

Results | Messages

| | Cid | CName | CDescription |
|---|---|---|---|
| 1 | 1 | Category 1 | very interesting |
| 2 | 2 | Category 2 | DESCRIPTION NON REPEATABLE READS |

| | Cid | CName | CDescription |
|---|---|---|---|
| 1 | 1 | Category 1 | very interesting |
| 2 | 2 | Category 2 | DESCRIPTION NON REPEATABLE READS |

# Practical Exam

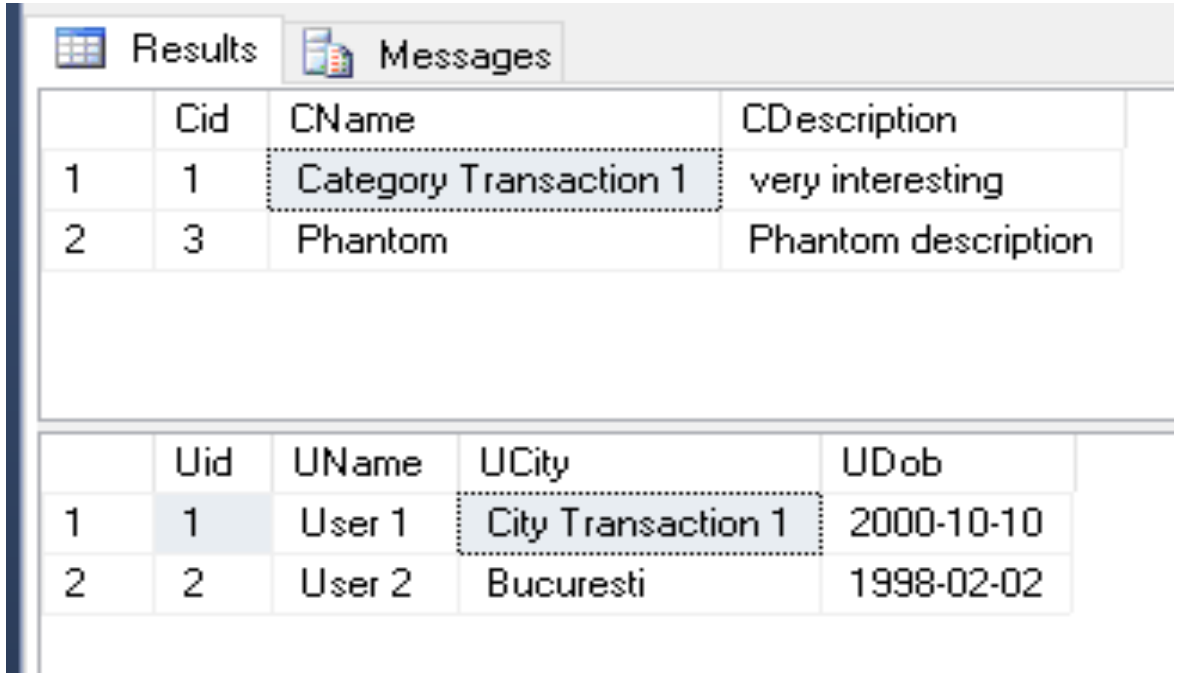
Deadlock

| *Transaction 1* | *Transaction 2* |
| --- | --- |
| use MiniFacebook_IE<br>Go<br>begin tran<br>update Users set UCity='City Transaction 1' where Uid=1<br>-- this transaction has exclusively lock on table Users<br>waitfor delay '00:00:10‘<br>update Categories set CName='Category Transaction 1' where Cid=1<br>commit tran | use MiniFacebook_IE<br>Go<br>begin tran<br>update Categories set CName='Category Transaction 2' where Cid=1<br>-- this transaction has exclusively lock on table Categories<br>waitfor delay '00:00:10‘<br>update Users set UCity='City Transaction 2' where Uid=1<br>commit tran |
| (1 row(s) affected)<br><br>(1 row(s) affected) | (1 row(s) affected)<br>Msg 1205, Level 13, State 51, Line 12<br>Transaction (Process ID 56) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction. |

# Practical Exam

Deadlock

- select * from Categories
- select * from Users




| | Cid | CName | CDescription |
|---|---|---|---|
| 1 | 1 | Category Transaction 1 | very interesting |
| 2 | 3 | Phantom | Phantom description |

| | Uid | UName | UCity | UDob |
|---|---|---|---|---|
| 1 | 1 | User 1 | City Transaction 1 | 2000-10-10 |
| 2 | 2 | User 2 | Bucuresti | 1998-02-02 |

# Practical Exam

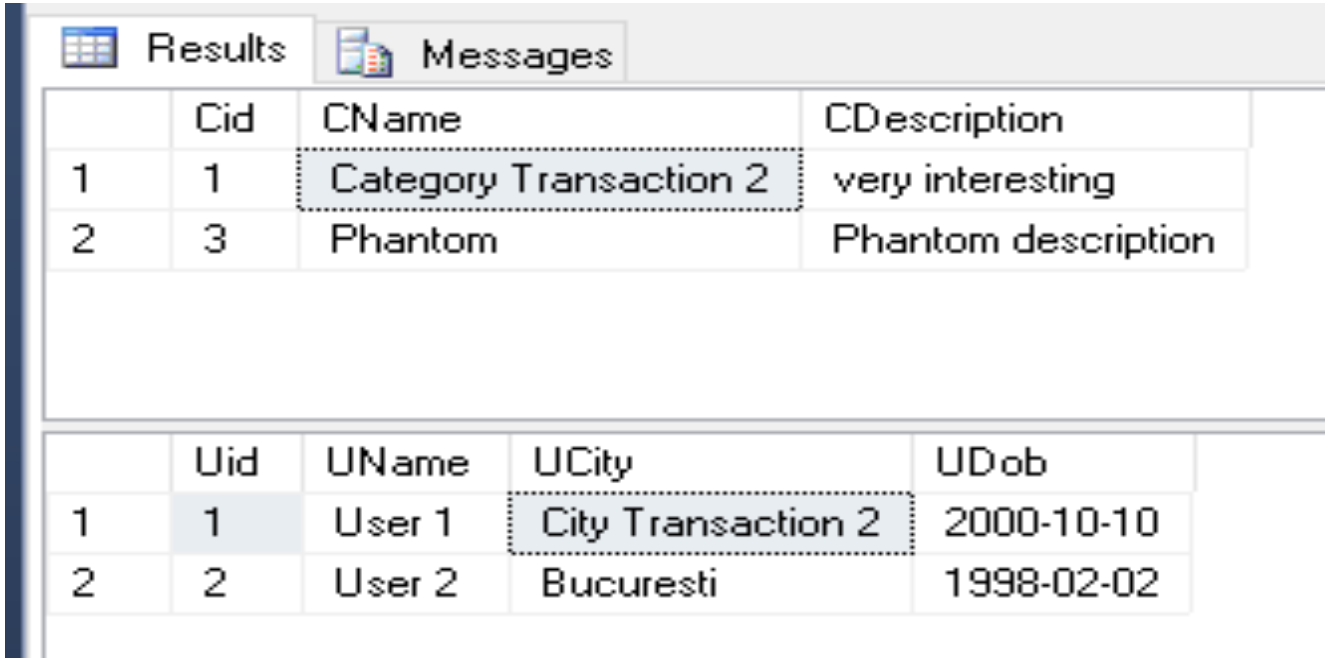
Deadlock - Solution

| *Transaction 1* | *Transaction 2* |
| --- | --- |
| use MiniFacebook_IE<br>Go<br>begin tran<br>update Users set UCity='City Transaction 1' where Uid=1<br>-- this transaction has exclusively lock on table Users<br>waitfor delay '00:00:10‘<br>update Categories set CName='Category Transaction 1' where Cid=1<br>commit tran | use MiniFacebook_IE<br>Go<br>SET DEADLOCK_PRIORITY HIGH<br>begin tran<br>update Categories set CName='Category Transaction 2' where Cid=1<br>-- this transaction has exclusively lock on table Categories<br>waitfor delay '00:00:10‘<br>update Users set UCity='City Transaction 2' where Uid=1<br>commit tran |
| (1 row(s) affected)<br>Msg 1205, Level 13, State 51, Line 12<br>Transaction (Process ID 56) was deadlocked on lock resources with another process and has been chosen as the deadlock victim. Rerun the transaction. | (1 row(s) affected)<br><br>(1 row(s) affected) |

# Practical Exam

Deadlock – Solution

- select * from Categories
- select * from Users




| | Cid | CName | CDescription |
|---|---|---|---|
| 1 | 1 | Category Transaction 2 | very interesting |
| 2 | 3 | Phantom | Phantom description |

| | Uid | UName | UCity | UDob |
|---|---|---|---|---|
| 1 | 1 | User 1 | City Transaction 2 | 2000-10-10 |
| 2 | 2 | User 2 | Bucuresti | 1998-02-02 |