Summary report of Homework 1

WUT, MiNI, 4th semester, Databases

NIKITA AMOL, ID:309034, 16.05.2021

The solution was completed using C# ASP.NET Web Application (.NET Framework) technology in Visual Studio 2019 IDE. First of all I created the database using queries attached to this zip archive. Then, I connect my "CinemaDatabaseHomework" database to my project: 1. In "Web.config" file, inside <configuration> tag, I add a new connection string:

```
<connectionStrings>
  <add name="CinemaDB"
       connectionString="Data Source=(local);Integrated Security=True;Initial Catalog=CinemaDatabaseHomework"
       providerName="System.Data.SqlClient"/>
</connectionStrings>
```

I indicate as server "(local)", and the name of my database.

Then, I create 2 files, representing my tables: Movie and Order classes (inside Data folder), containing equivalent fields in corresponding tables. Besides fields, in Movie class I declare GetAllMovies() method, which takes connection string as parameter, and returns the List<Movie> - the whole database represented in form of C# List, and selectFiltered() method, taking connection string, start and end date, keyword to search in Movie's GENRE and minimum RATING – this method returns the List<Movie> filtered by various criteria being got from user input; in Order class – only GetAllOrders(), which does the same thing with Orders DB (SELECT * FROM ORDERS).

Now, I create the main web page "Movies.aspx" – this is the page that user sees after he enters the website. It contains MOVIES and ORDERS database at its current state, below them, there are 2 buttons: "Open filtering" and "Update or Delete orders". The first one redirects user to the interface allowing to look for filtered data and make an order for chosen film ("Filtering.aspx"); the second one redirects us to UPDATE/DELETE functionality ("UpdateDelete.aspx"). Here is how it looks like:

MOVIES

MovieID	MovieTitle	ReleaseDate	Price	Rating	Genre
1	Titanic	18.06.2012 00:00:00	200	7,81	Drama
2	Inception	19.06.2013 00:00:00	150,49	8,62	Thriller
3	Calvary	18.06.2012 00:00:00	99,99	7,73	Drama
4	The Guard	29.05.2011 00:00:00	199,99	7,64	Comedy
5	The Place Beyond the Pines	06.06.2012 00:00:00	185,9	7,55	Crime
6	The Shawshank Redemption	14.02.1994 00:00:00	1000	9,36	Drama
7	Pulp Fiction	29.09.1994 00:00:00	1500	8,97	Crime
8	Fight Club	13.01.1999 00:00:00	350,99	8,88	Drama
9	Forrest Gump	06.07.1994 00:00:00	88,95	8,89	Romance
10	Se7en	22.09.1995 00:00:00	99,49	8,61	Mystery
11	The Prestige	18.01.2007 00:00:00	239,49	8,5	Sci-Fi
12	Joker	03.10.2019 00:00:00	250	8,4	Thriller
13	Soul	21.01.2020 00:00:00	150	8,13	Adventure
14	1917	30.01.2019 00:00:00	113,49	8,3	War

ORDERS

OrderID	RentalDate	ReturnDate	MovieID	NetAmount	Discount	GrossAmount
1	18.06.2020 00:00:00	25.06.2020 00:00:00	6	1000	0,05	1197
2	01.07.2020 00:00:00	08.07.2020 00:00:00	3	99,99	0,1	113,39
3	08.08.2020 00:00:00	09.09.2020 00:00:00	7	1500	0,1	1701
4	25.03.2021 00:00:00	24.04.2021 00:00:00	13	150	0,15	160,65
5	18.04.2021 00:00:00	19.04.2012 00:00:00	6	1000	0,12	1108,8
6	01.05.2021 00:00:00	05.05.2021 00:00:00	11	239,49	0,05	286,65
7	01.09.2020 00:00:00	03.09.2020 00:00:00	2	150,49	0	189,62
8	25.11.2020 00:00:00	02.12.2020 00:00:00	4	199,99	0	251,99
9	13.01.2021 00:00:00	27.01.2021 00:00:00	10	99,49	0	125,36
10	05.03.2021 00:00:00	25.03.2021 00:00:00	9	88,95	0	112,08
1030	16.05.2021 00:00:00	19.05.2021 00:00:00	8	999	0,25	400
1031	16.05.2021 00:00:00	19.05.2021 00:00:00	9	88,95	0,25	112,07
1032	16.05.2021 00:00:00	19.05.2021 00:00:00	10	99,49	0,25	125,99
1036	16.05.2021 00:00:00	19.05.2021 00:00:00	8	350,99	0,25	400
1037	16.05.2021 00:00:00	19.05.2021 00:00:00	9	88,95	0,25	112,07
1038	16.05.2021 00:00:00	19.05.2021 00:00:00	10	99,49	0,25	125,99

Open filtering

Update or Delete orders

Then, I wrote logic for “Filtering.aspx”. (NOTE: The logic for *.aspx files are written in *.aspx.cs files) The page contains a button “Home page” to be able to return to “Movies.aspx”, 4 textboxes for start and end date, GENRE keyword and minimum rating, “Submit” button. When “Submit” is pressed the program analyses the correctness of given data and if it does not fail, under these textboxes appears a table filtered by entered restrctions. Under this table, there is a DropDownList, which contains only IDs of Movies inside this table (by default there is no values there). After user chooses the ID of the film, he/she wants to order, there is a button “Order film” which UPDATEs the ORDERS table. If order was successful, user can turn back to Home Page to see that his/her order was added successfully. The example is presented below:

Filtering by different conditions

Home Page

Release date filtering (Note: It will work only in case both dates are specified)

Start date(in format YYYY-MM-DD):

1990-01-01

End date (in format YYYY-MM-DD):

2020-01-01

Only movies with specific keyword in genre (e.g Drama):

Drama

Minimum rating for the movie (Rank from 0.00 up to 9.99):

7

Submit

Last filtered table:

MovieID	MovieTitle	ReleaseDate	Price	Rating	Genre
1	Titanic	18.06.2012 00:00:00	200	7,81	Drama
3	Calvary	18.06.2012 00:00:00	99,99	7,73	Drama
6	The Shawshank Redemption	14.02.1994 00:00:00	1000	9,36	Drama
8	Fight Club	13.01.1999 00:00:00	350,99	8,88	Drama

Please choose id of the film you want to order, then press "Order film".

3

Order film Order added. Data inserted succesfully!

And the last element is ORDERS table now is:

1039	16.05.2021 00:00:00	19.05.2021 00:00:00	3	99,99	0	125,9874

After implementing filtering, the last thing to do was to write logic for "UpdateDelere.aspx". This page contains, from top to bottom: "Home page button", current ORDERS table (to see changes immediately), DropDownList containing current all ORDER_IDs; 6 textboxes for user to enter RENTAL_DATE, RETURN_DATE, MOVIE_ID, NET_AMOUNT, DISCOUNT and GROSS_AMOUNT (Note: it is possible to update data without filling all textboxes); "Save changes" button to save the data; 1 textbox for user to write ORDER_IDs of rows in table he/she wants to delete, separated by blank spaces.

Updating and Deleting specific orders

Home Page

Current Orders table

OrderID	RentalDate	ReturnDate	MovieID	NetAmount	Discount	GrossAmount
1	18.06.2020 00:00:00	25.06.2020 00:00:00	6	1000	0,05	1197
2	01.07.2020 00:00:00	08.07.2020 00:00:00	3	99,99	0,1	113,39
3	08.08.2020 00:00:00	09.09.2020 00:00:00	7	1500	0,1	1701
4	25.03.2021 00:00:00	24.04.2021 00:00:00	13	150	0,15	160,65
5	18.04.2021 00:00:00	19.04.2012 00:00:00	6	1000	0,12	1108,8
6	01.05.2021 00:00:00	05.05.2021 00:00:00	11	239,49	0,05	286,65
7	01.09.2020 00:00:00	03.09.2020 00:00:00	2	150,49	0	189,62
8	25.11.2020 00:00:00	02.12.2020 00:00:00	4	199,99	0	251,99
9	13.01.2021 00:00:00	27.01.2021 00:00:00	10	99,49	0	125,36
10	05.03.2021 00:00:00	25.03.2021 00:00:00	9	88,95	0	112,08
1030	16.05.2021 00:00:00	19.05.2021 00:00:00	8	999	0,25	400
1031	16.05.2021 00:00:00	19.05.2021 00:00:00	9	88,95	0,25	112,07
1032	16.05.2021 00:00:00	19.05.2021 00:00:00	10	99,49	0,25	125,99
1036	16.05.2021 00:00:00	19.05.2021 00:00:00	8	350,99	0,25	400
1037	16.05.2021 00:00:00	19.05.2021 00:00:00	9	88,95	0,25	112,07
1038	16.05.2021 00:00:00	19.05.2021 00:00:00	10	99,49	0,25	125,99
1039	16.05.2021 00:00:00	19.05.2021 00:00:00	3	99,99	0	125,9874

Choose Order ID you want to UPDATE:

1

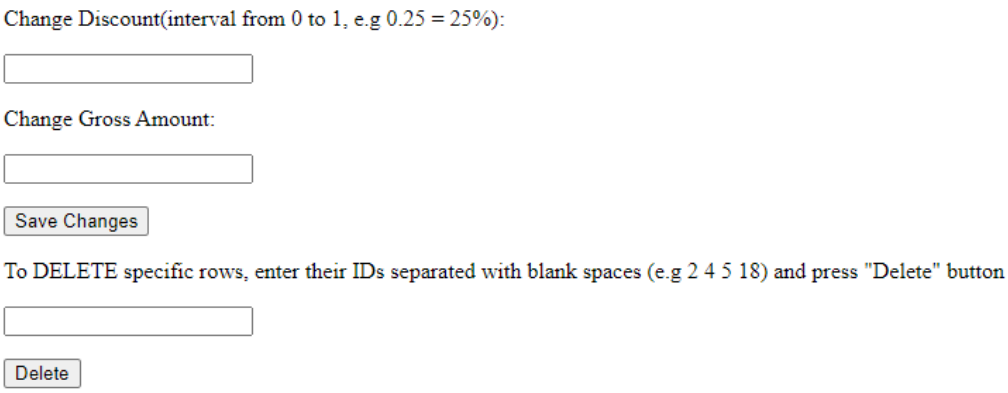
Be careful! If you enter something in incorrect format nothing will happen

Change Rental Date(in format YYYY-MM-DD):

Change Return Date(in format YYYY-MM-DD):

Change movie ID:

Change Net Amount(e.g 999.999):



To simulate the concurrent data flow, I used the following syntax (eventually I changed ROLLBACK to COMMIT to observe changes)

INSERT:

```
USE CinemaDatabaseHomework;
GO

SET TRANSACTION ISOLATION LEVEL
READ COMMITTED

BEGIN TRANSACTION

INSERT INTO ORDERS
(RENTAL_DATE,RETURN_DATE, MOVIE_ID,NET_AMOUNT,DISCOUNT,GROSS_AMOUNT)
VALUES
('2021-05-16','2021-05-19', 8, 350.99, 0.25, 400);

INSERT INTO ORDERS
(RENTAL_DATE,RETURN_DATE, MOVIE_ID,NET_AMOUNT,DISCOUNT,GROSS_AMOUNT)
VALUES
('2021-05-16','2021-05-19', 9, 88.95, 0.25, 112.07);

INSERT INTO ORDERS
(RENTAL_DATE,RETURN_DATE, MOVIE_ID,NET_AMOUNT,DISCOUNT,GROSS_AMOUNT)
VALUES
('2021-05-16','2021-05-19', 10, 99.49, 0.25, 125.99);

ROLLBACK
```

UPDATE:

```
USE CinemaDatabaseHomework;
GO

SET TRANSACTION
ISOLATION LEVEL
READ COMMITTED

BEGIN TRANSACTION

UPDATE ORDERS SET NET_AMOUNT=999 WHERE MOVIE_ID>13;

UPDATE ORDERS SET GROSS_AMOUNT=111 WHERE MOVIE_ID<16;

UPDATE ORDERS SET DISCOUNT=0.5 WHERE NET_AMOUNT>5000;


ROLLBACK
```

DELETE:

```
USE CinemaDatabaseHomework;
GO


SET TRANSACTION
ISOLATION LEVEL
READ COMMITTED

BEGIN TRANSACTION

DELETE FROM ORDERS WHERE ORDER_ID=14;

DELETE FROM ORDERS WHERE ORDER_ID=1004;

DELETE FROM ORDERS WHERE ORDER_ID=888;

ROLLBACK
```

As it is visible on screenshots, I used READ COMMITED isolation level since it is enough for our situation to preserve safe simultaneous transactions. Firstly, I executed without committing INSERT, then UPDATE and DELETE, and committed them in the same order. Before changes the ORDERS table had the following view:

	ORDER_ID	RENTAL_DATE	RETURN_DATE	MOVIE_ID	NET_AMOUNT	DISCOUNT	GROSS_AMOUNT
1	1	2020-06-18	2020-06-25	6	1000,00	0.05	1197,00
2	2	2020-07-01	2020-07-08	3	99,99	0.10	113,39
3	3	2020-08-08	2020-09-09	7	1500,00	0.10	1701,00
4	4	2021-03-25	2021-04-24	13	150,00	0.15	160,65
5	5	2021-04-18	2012-04-19	6	1000,00	0.12	1108,80
6	6	2021-05-01	2021-05-05	11	239,49	0.05	286,65
7	7	2020-09-01	2020-09-03	2	150,49	NULL	189,62
8	8	2020-11-25	2020-12-02	4	199,99	NULL	251,99
9	9	2021-01-13	2021-01-27	10	99,49	NULL	125,36
10	10	2021-03-05	2021-03-25	9	88,95	NULL	112,08
11	1030	2021-05-16	2021-05-19	8	350,99	0.25	400,00
12	1031	2021-05-16	2021-05-19	9	88,95	0.25	112,07
13	1032	2021-05-16	2021-05-19	10	99,49	0.25	125,99
14	1033	2021-05-16	2021-05-19	8	350,99	0.25	400,00
15	1034	2021-05-16	2021-05-19	9	88,95	0.25	112,07
16	1035	2021-05-16	2021-05-19	10	99,49	0.25	125,99

After committing all transactions, it looks like this:

Результаты Сообщения

	ORDER_ID	RENTAL_DATE	RETURN_DATE	MOVIE_ID	NET_AMOUNT	DISCOUNT	GROSS_AMOUNT
1	1	2020-06-18	2020-06-25	6	1000,00	0.05	1197,00
2	2	2020-07-01	2020-07-08	3	99,99	0.10	113,39
3	3	2020-08-08	2020-09-09	7	1500,00	0.10	1701,00
4	4	2021-03-25	2021-04-24	13	150,00	0.15	160,65
5	5	2021-04-18	2012-04-19	6	1000,00	0.12	1108,80
6	6	2021-05-01	2021-05-05	11	239,49	0.05	286,65
7	7	2020-09-01	2020-09-03	2	150,49	NULL	189,62
8	8	2020-11-25	2020-12-02	4	199,99	NULL	251,99
9	9	2021-01-13	2021-01-27	10	99,49	NULL	125,36
10	10	2021-03-05	2021-03-25	9	88,95	NULL	112,08
11	1030	2021-05-16	2021-05-19	8	999,00	0.25	400,00
12	1031	2021-05-16	2021-05-19	9	88,95	0.25	112,07
13	1032	2021-05-16	2021-05-19	10	99,49	0.25	125,99
14	1036	2021-05-16	2021-05-19	8	350,99	0.25	400,00
15	1037	2021-05-16	2021-05-19	9	88,95	0.25	112,07
16	1038	2021-05-16	2021-05-19	10	99,49	0.25	125,99

As we see, everything was done without any conflict, therefore READ COMMITED level of isolation is enough to satisfy us.

I certify that this assignment is entirely my own work, performed independently and without any help from the sources which are not allowed.

Nikita Amol.