T.C.

MARMARA UNIVERSITY

FACULTY of ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE 3055 – Database Systems Homework VII Report

Cem GÜLEÇ - 150117828

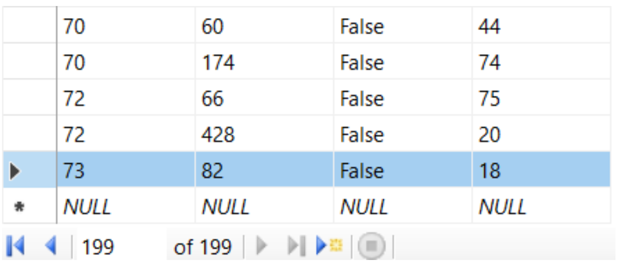
10 January 2021

Insertion Operation

```
else
begin
    -- insert case
    set @MIB  = ' '
    set @MIA  = (select i.MatchID from inserted i)
    set @PIB  = ' '
    set @PIA  = (select i.PlayerID from inserted i)
    set @IOGB = ' '
    set @IOGA = (select i.IsOwnGoal from inserted i)

    set @LogTime = SYSDATETIME()
    set @LogType = 'I'
    set @BeforeState = NULL
    set @AfterState  = convert(varchar(6), @MIA)  + ';' + convert(varchar(6), @PIA) + ';' +
                       convert(varchar(6), @IOGA) + ';' + convert(varchar(6), @Min)
end
```

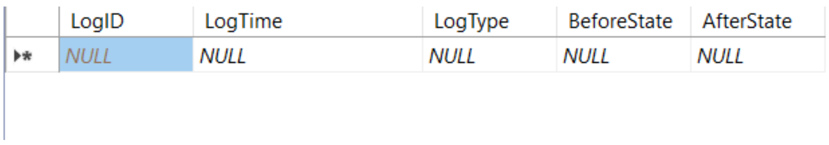
Here, only after state variable types are recorded and assigned null to @BeforeState. LogType is taken as ‘I’.

Beforehand*:*



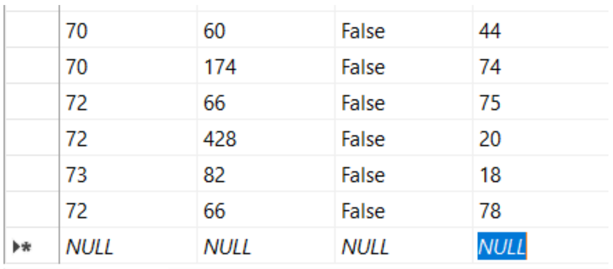
	70	60	False	44
	70	174	False	74
	72	66	False	75
	72	428	False	20
▶	73	82	False	18
*	NULL	NULL	NULL	NULL

199 of 199

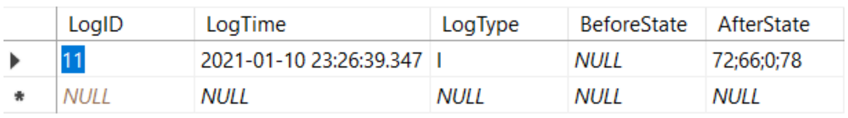


	LogID	LogTime	LogType	BeforeState	AfterState
▶*	NULL	NULL	NULL	NULL	NULL

Afterhand*:*



	70	60	False	44
	70	174	False	74
	72	66	False	75
	72	428	False	20
	73	82	False	18
	72	66	False	78
▶*	NULL	NULL	NULL	NULL



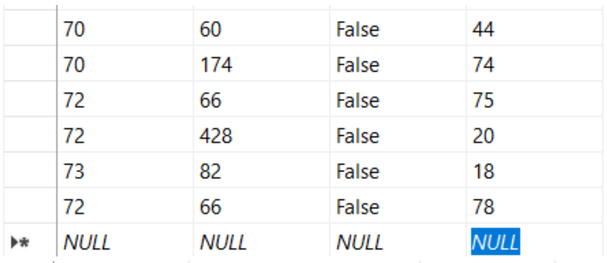
	LogID	LogTime	LogType	BeforeState	AfterState
▶	11	2021-01-10 23:26:39.347	I	NULL	72;66;0;78
*	NULL	NULL	NULL	NULL	NULL

Update Operation

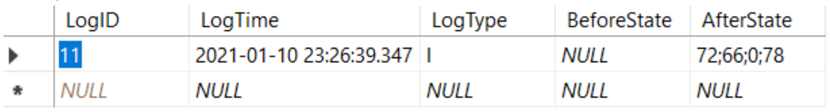
```
if exists (select * from inserted)
begin
    if exists (select * from deleted)
    begin
        -- update case
        set @MIB  = (select d.MatchID from deleted d)
        set @MIA  = (select i.MatchID from inserted i)
        set @PIB  = (select d.PlayerID from deleted d)
        set @PIA  = (select i.PlayerID from inserted i)
        set @IOGB = (select d.IsOwnGoal from deleted d)
        set @IOGA = (select i.IsOwnGoal from inserted i)

        set @LogTime = SYSDATETIME()
        set @LogType = 'U'
        set @BeforeState = convert(varchar(6), @MIB)  + ';' + convert(varchar(6), @PIB) + ';' +
                           convert(varchar(6), @IOGB) + ';' + convert(varchar(6), @Min)
        set @AfterState  = convert(varchar(6), @MIA)  + ';' + convert(varchar(6), @PIA) + ';' +
                           convert(varchar(6), @IOGA) + ';' + convert(varchar(6), @Min)
    end
```

Here, before and after state variable types are recorded and LogType is assigned as ‘U’.

Beforehand:

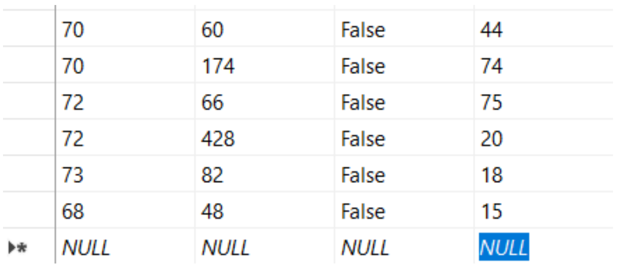


	70	60	False	44
	70	174	False	74
	72	66	False	75
	72	428	False	20
	73	82	False	18
	72	66	False	78
▶*	NULL	NULL	NULL	NULL

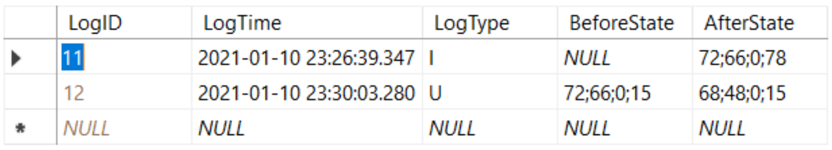


	LogID	LogTime	LogType	BeforeState	AfterState
▶	11	2021-01-10 23:26:39.347	I	NULL	72;66;0;78
*	NULL	NULL	NULL	NULL	NULL

Afterhand:



	70	60	False	44
	70	174	False	74
	72	66	False	75
	72	428	False	20
	73	82	False	18
	68	48	False	15
▶*	NULL	NULL	NULL	NULL



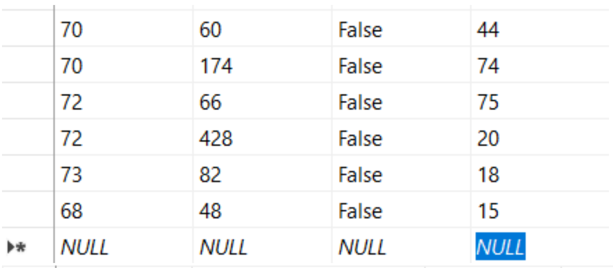
	LogID	LogTime	LogType	BeforeState	AfterState
▶	11	2021-01-10 23:26:39.347	I	NULL	72;66;0;78
	12	2021-01-10 23:30:03.280	U	72;66;0;15	68;48;0;15
*	NULL	NULL	NULL	NULL	NULL

Deletion Operation

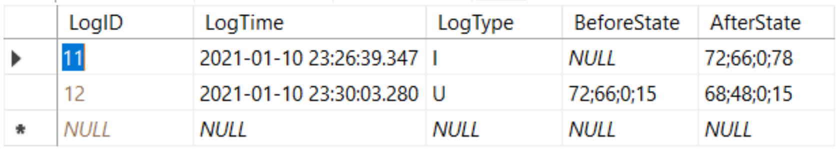
```
else
begin
    -- delete case
    set @MIB  = (select d.MatchID from deleted d)
    set @MIA  = ' '
    set @PIB  = (select d.PlayerID from deleted d)
    set @PIA  = ' '
    set @IOGB = (select d.IsOwnGoal from deleted d)
    set @IOGA = ' '

    set @LogTime = SYSDATETIME()
    set @LogType = 'D'
    set @BeforeState = convert(varchar(6), @MIB)  + ';' + convert(varchar(6), @PIB) + ';' +
                       convert(varchar(6), @IOGB) + ';' + convert(varchar(6), @Min)
    set @AfterState  = NULL
end
```

Here, only before state variable types are recorded and assigned null to @AfterState. LogType is taken as 'D'.

Beforehand:

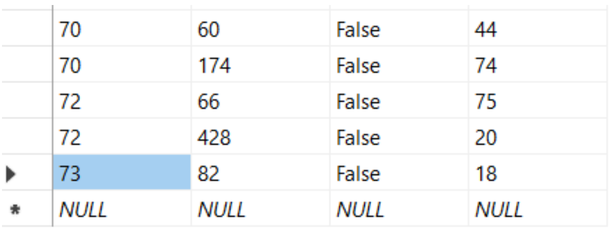


	70	60	False	44
	70	174	False	74
	72	66	False	75
	72	428	False	20
	73	82	False	18
	68	48	False	15
▶*	NULL	NULL	NULL	NULL

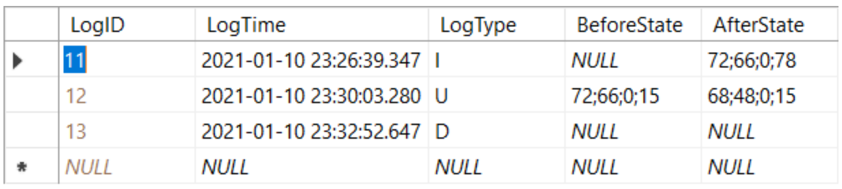


	LogID	LogTime	LogType	BeforeState	AfterState
▶	11	2021-01-10 23:26:39.347	I	NULL	72;66;0;78
	12	2021-01-10 23:30:03.280	U	72;66;0;15	68;48;0;15
*	NULL	NULL	NULL	NULL	NULL

Afterhand:



	70	60	False	44
	70	174	False	74
	72	66	False	75
	72	428	False	20
▶	73	82	False	18
*	NULL	NULL	NULL	NULL

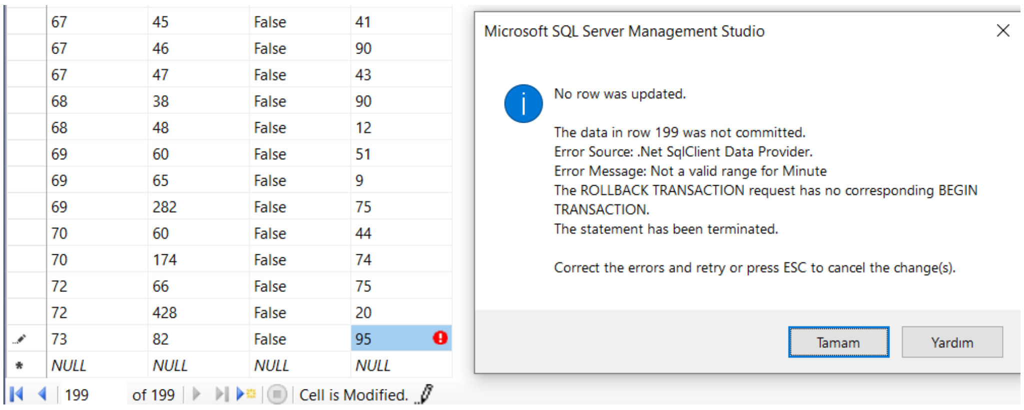


	LogID	LogTime	LogType	BeforeState	AfterState
▶	11	2021-01-10 23:26:39.347	I	NULL	72;66;0;78
	12	2021-01-10 23:30:03.280	U	72;66;0;15	68;48;0;15
	13	2021-01-10 23:32:52.647	D	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL

Error cases

- In case where Goals's minute values is not between 1 and 90 it gives an error and rollsback transaction.

```
-- in case a goal is entered explicitly check
-- whether its minute information between 1 and 90
declare @Min tinyint set @Min = (select i.Minute from inserted i)
if @Min<1 or @Min>90
begin
    rollback transaction
    raiserror('Not a valid range for Minute', 16, 1)
end
```

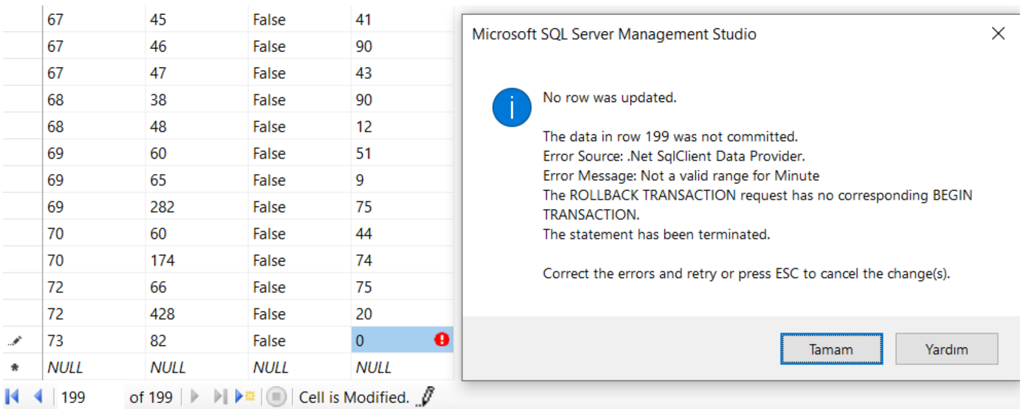


	67	45	False	41
	67	46	False	90
	67	47	False	43
	68	38	False	90
	68	48	False	12
	69	60	False	51
	69	65	False	9
	69	282	False	75
	70	60	False	44
	70	174	False	74
	72	66	False	75
	72	428	False	20
✎	73	82	False	95
*	NULL	NULL	NULL	NULL

199 of 199 Cell is Modified.

Microsoft SQL Server Management Studio

```
No row was updated.

The data in row 199 was not committed.
Error Source: .Net SqlClient Data Provider.
Error Message: Not a valid range for Minute
The ROLLBACK TRANSACTION request has no corresponding BEGIN
TRANSACTION.
The statement has been terminated.

Correct the errors and retry or press ESC to cancel the change(s).
```



	67	45	False	41
	67	46	False	90
	67	47	False	43
	68	38	False	90
	68	48	False	12
	69	60	False	51
	69	65	False	9
	69	282	False	75
	70	60	False	44
	70	174	False	74
	72	66	False	75
	72	428	False	20
✎	73	82	False	0
*	NULL	NULL	NULL	NULL

199 of 199 Cell is Modified.

Microsoft SQL Server Management Studio

```
No row was updated.

The data in row 199 was not committed.
Error Source: .Net SqlClient Data Provider.
Error Message: Not a valid range for Minute
The ROLLBACK TRANSACTION request has no corresponding BEGIN
TRANSACTION.
The statement has been terminated.

Correct the errors and retry or press ESC to cancel the change(s).
```

- Also, in case PlayerID is not in the specified rules of not in the home or visiting TeamID or in season '13-14' and error is fired and transaction is rolled back.

```sql
-- first checking whether PlayerID in table Goals is either in the home team
-- or the visiting team for that match in season 13-14.
-- if deleted virtual table is not existing, both update and delete table will not be existing
if not exists (select * from deleted)
begin
    if (select i.PlayerID from inserted i) not in (select pt.PlayerID
                                                   from inserted i, Match m, PlayerTeam pt
                                                   where (m.HomeTeamID=pt.TeamID OR m.VisitingTeamID=pt.TeamID) and pt.Season='13-14'
                                                          and m.MatchID=i.MatchID
                                                          and i.PlayerID=pt.PlayerID)
    -- In inner if statement if specified playerID is not inside the inserted table
    -- directly it will not match with our rules and transaction will be rolled back
    -- Else: transactions continue on executing and will eventually come across other
    -- if statements such as inserted, delete and update
    begin
        rollback transaction
        raiserror('Operation stopped execution due to operation out of scope.', 16, 1)
    end
end
```