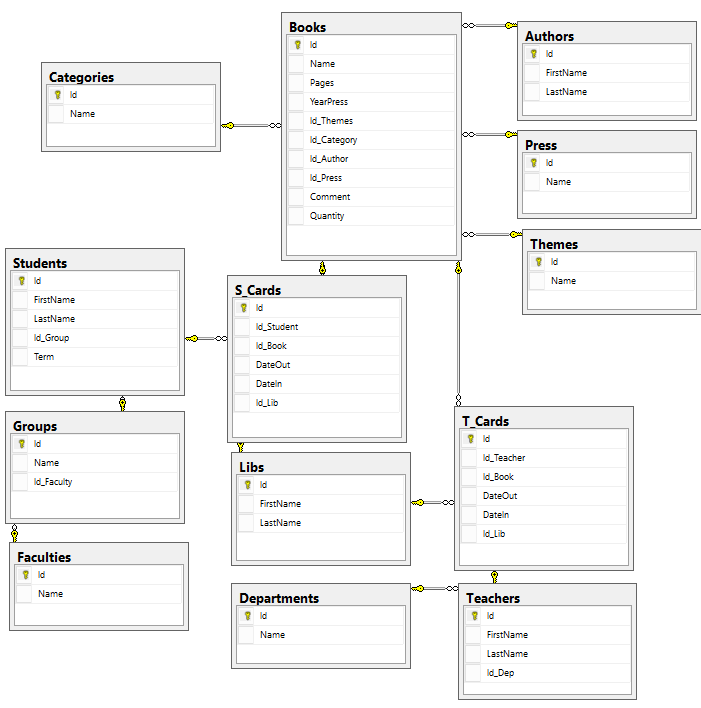
***Группа ИТ-2707. Леженко С.А.***

use library;



--Написать такие триггера:

--Чтобы при взятии определенной книги, ее кол-во уменьшалось на 1.

CREATE TRIGGER onStudentTakeBook

ON S\_Cards

AFTER INSERT

AS

BEGIN

DECLARE @dOut smalldatetime;

SELECT @dOut = DateOut

FROM inserted;

DECLARE @dIn smalldatetime;

SELECT @dIn = DateIn

FROM inserted;

DECLARE @IdBook int;

SELECT @IdBook = Id\_Book

FROM inserted;

DECLARE @Quantity int;

SELECT @Quantity = Quantity

FROM Books

WHERE Id = @IdBook;

IF(@dOut IS NOT NULL

AND @dIn IS NULL

AND @Quantity>0)

UPDATE Books

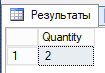
SET Quantity = @Quantity-1

WHERE Id=@IdBook;

END;

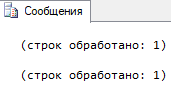
GO

SELECT Quantity FROM Books WHERE Id=1;

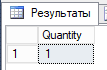


INSERT INTO S\_Cards

VALUES(11,1,'16/01/2017',NULL,1);



SELECT Quantity FROM Books WHERE Id=1;



GO

--Триггер на карточки преподавателей

CREATE TRIGGER onTeacherTakeBook

ON T\_Cards

FOR INSERT

AS

BEGIN

DECLARE @dOut smalldatetime;

SELECT @dOut = DateOut

FROM inserted;

DECLARE @dIn smalldatetime;

SELECT @dIn = DateIn

FROM inserted;

DECLARE @IdBook int;

SELECT @IdBook = Id\_Book

FROM inserted;

DECLARE @Quantity int;

SELECT @Quantity = Quantity

FROM Books

WHERE Id = @IdBook;

IF(@dOut IS NOT NULL

AND @dIn IS NULL

AND @Quantity>0)

UPDATE Books

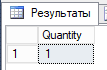
SET Quantity = @Quantity-1

WHERE Id=@IdBook;

END;

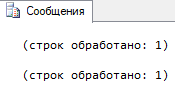
GO

SELECT Quantity FROM Books WHERE Id=1;

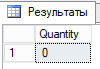


INSERT INTO T\_Cards

VALUES(4,1,'17/01/2017',NULL,1);



SELECT Quantity FROM Books WHERE Id=1;



--Чтобы при возврате определенной книги, ее кол-во увеличивалось на 1

--При возврате книги преподавателем

CREATE TRIGGER onTeacherReturnBook

ON T\_Cards

FOR UPDATE

AS

BEGIN

DECLARE @dIn smalldatetime;

SELECT @dIn = DateIn

FROM inserted;

DECLARE @IdBook int;

SELECT @IdBook = Id\_Book

FROM inserted;

DECLARE @Quantity int;

SELECT @Quantity = Quantity

FROM Books

WHERE Id = @IdBook;

IF(@dIn IS NOT NULL)

UPDATE Books

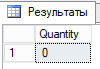
SET Quantity = @Quantity+1

WHERE Id=@IdBook;

END;

GO

SELECT Quantity FROM Books WHERE Id=1;



UPDATE T\_Cards

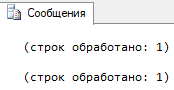
SET DateIn='05/05/2017'

WHERE Id\_Book=1

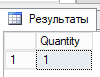
AND Id\_Teacher=4

AND Id\_Lib=1

AND DateOut='17/01/2017';



SELECT Quantity FROM Books WHERE Id=1;



GO

--При возврате книги студентом

CREATE TRIGGER onStudentReturnBook

ON S\_Cards

FOR UPDATE

AS

BEGIN

DECLARE @dIn smalldatetime;

SELECT @dIn = DateIn

FROM inserted;

DECLARE @IdBook int;

SELECT @IdBook = Id\_Book

FROM inserted;

DECLARE @Quantity int;

SELECT @Quantity = Quantity

FROM Books

WHERE Id = @IdBook;

IF(@dIn IS NOT NULL)

UPDATE Books

SET Quantity = @Quantity+1

WHERE Id=@IdBook;

END;

GO

SELECT Quantity FROM Books WHERE Id=1;



UPDATE S\_Cards

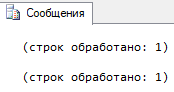
SET DateIn='06/05/2017'

WHERE Id\_Book=1

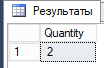
AND Id\_Student=11

AND Id\_Lib=1

AND DateOut='16/01/2017';



SELECT Quantity FROM Books WHERE Id=1;



--Чтобы нельзя было выдать книгу, которой уже нет в библиотеке (по кол-ву)

GO

--Триггер на карточки студентов

CREATE TRIGGER onStudentTakeAbsentBook

ON S\_Cards

FOR INSERT

AS

BEGIN

DECLARE @IdBook int;

SELECT @IdBook = Id\_Book

FROM inserted;

DECLARE @Quantity int;

SELECT @Quantity = Quantity

FROM Books

WHERE Id = @IdBook;

IF(@Quantity=0)

BEGIN

PRINT('Book is absent');

ROLLBACK TRANSACTION;

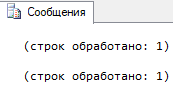
END;

END;

GO

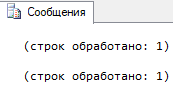
INSERT INTO S\_Cards

VALUES(4,1,'17/01/2017',NULL,1);



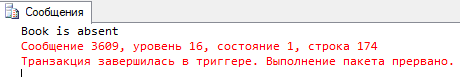
INSERT INTO T\_Cards

VALUES(4,1,'17/01/2017',NULL,1);



INSERT INTO T\_Cards

VALUES(4,1,'17/01/2017',NULL,1);



GO

--Чтобы нельзя было выдать более трех книг одному студенту

CREATE TRIGGER onStudentTakeMoreThen3Books

ON S\_Cards

FOR INSERT

AS

BEGIN

DECLARE @CountBooks int;

DECLARE @IdStudent int;

SELECT @CountBooks = COUNT(Id)

FROM S\_Cards

WHERE Id\_Student =

(SELECT Id\_Student

FROM inserted)

AND DateIn IS NULL;

IF(@CountBooks=3)

BEGIN

PRINT('You can not issue more than 3 books');

ROLLBACK TRANSACTION;

END

END;

GO

INSERT INTO S\_Cards

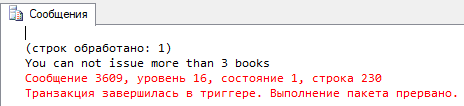
VALUES(11,18,'14/11/2016',NULL,1);

INSERT INTO S\_Cards

VALUES(11,18,'14/11/2016',NULL,1);

INSERT INTO S\_Cards

VALUES(11,18,'14/11/2016',NULL,1);



--Чтобы при удалении книги, данные о ней копировались в таблицу Удаленные

CREATE TABLE DeletedBooks(

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [varchar](100) NOT NULL,

[Pages] [int] NOT NULL,

[YearPress] [int] NOT NULL,

[Id\_Themes] [int] NOT NULL,

[Id\_Category] [int] NOT NULL,

[Id\_Author] [int] NOT NULL,

[Id\_Press] [int] NOT NULL,

[Comment] [varchar](50) NULL,

[Quantity] [int] NOT NULL);

GO

CREATE TRIGGER onDeleteBook

ON Books

AFTER DELETE

AS

BEGIN

INSERT INTO DeletedBooks

SELECT Name,Pages,YearPress,

Id\_Themes,Id\_Category,

Id\_Author,Id\_Press,

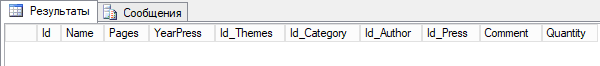
Comment, Quantity

FROM deleted;

END;

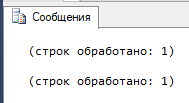
GO

SELECT \* FROM DeletedBooks;



DELETE FROM Books

WHERE Id=5;



SELECT \* FROM DeletedBooks;



--Если книга добавляется в базу, она должна быть удалена из таблицы Удаленные

GO

CREATE TRIGGER onInsertBookFromDeleted

ON Books

AFTER INSERT

AS

BEGIN

DELETE FROM DeletedBooks

WHERE EXISTS

(SELECT \*

FROM DeletedBooks

WHERE Name = (SELECT Name FROM inserted)

AND Pages = (SELECT Pages FROM inserted)

AND YearPress = (SELECT YearPress FROM inserted)

AND Id\_Themes = (SELECT Id\_Themes FROM inserted)

AND Id\_Category = (SELECT Id\_Category FROM inserted)

AND Id\_Author = (SELECT Id\_Author FROM inserted)

AND Id\_Press = (SELECT Id\_Press FROM inserted));

END;

GO

INSERT INTO Books

SELECT Name,Pages,YearPress,

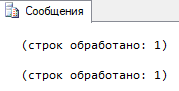
Id\_Themes,Id\_Category,

Id\_Author,Id\_Press,

Comment, Quantity

FROM DeletedBooks

WHERE Id=1;



SELECT \* FROM DeletedBooks;

