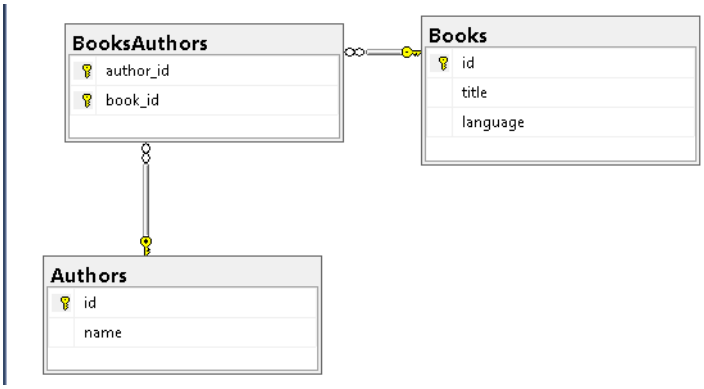
Lab 3

All the requirements have to be solved in SQL SERVER.

Creaţi o procedură stocată ce inserează date pentru entităţi ce se află într-o relaţie m-n. Dacă o operaţie de inserare eşuează, trebuie făcut *roll-back* pe întreaga procedură stocată. (nota: 6)

-

We consider the database



Ca notă generală, nu se va transmite niciun ID ca parametru de intrare a unei proceduri stocate şi toţi parametrii trebuie să fie validaţi (utilizaţi funcţii acolo unde este nevoie).

Create functions for validation: for example - check the language to have some values (for table Books)

```
CREATE FUNCTION uf_ValidateLanguage (@language varchar(100)) RETURNS INT AS
BEGIN
DECLARE @return INT
SET @return = 0
IF(@language  IN ('English','Romanian','French'))
SET @return = 1
RETURN @return
END
```

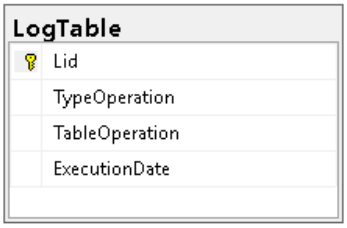
Create the stored procedure with the following restrictions:

- Do not take the Id's as parameters (here id from Authors, id from Books, author_id and book_id from BooksAuthors)
- Take the parameters all the rest of the fields from the tables (here title, language, name)
- Create validation functions for the parameters (all you consider necessary), like:
  - a field apart to a domain of values (language IN ('English','Romanian','French')))
  - the fields of varchar type to be not null, start with a upper type, ..
  - the fields of int to be positive, …
  - validation functions for telephone numbers, e-mail, …
  - or, whatever do you need
- first we insert values in the tables Authors and Books (the order is not important) and then in BooksAuthors (the intermediate table), by taking the id from both of the tables. We can take the id from one of the tables in a variable or if the field is identity like the maximum value of that field.

De asemenea, pentru toate scenariile trebuie să stabiliţi un sistem de logare ce vă va permite să memoraţi istoricul acţiunilor executate. Pentru detectarea erorilor se recomandă folosirea clauzei try-catch.

The store procedure must include all the fields from the tables (3 tables) involved, except the id's of these tables (the primary key's, that can be extracted with MAX value introduced, SCOPE_IDENTITY(), …), and these fields must be validated.

For the log system, one can verify with SELECT or save in a log table.

SELECT * FROM Authors SELECT * FROM Books SELECT * FROM BooksAuthors EXEC …. SELECT * FROM Authors SELECT * FROM Books SELECT * FROM BooksAuthors	CREATE TABLE LogTable(Lid INT IDENTITY PRIMARY KEY, TypeOperation VARCHAR(50), TableOperation VARCHAR(50), ExecutionDate DATETIME) Where TypeOperation can be Update, Select, …. 
Or any other method…	

Next, we give an example for a stored procedure for table Books.

```
CREATE PROCEDURE AddBookAuthor @title varchar(50), @language varchar(50) AS
BEGIN

BEGIN TRAN
BEGIN TRY
IF(dbo.uf_ValidateLanguage(@language)<>1)
BEGIN
        RAISERROR('Language must be Romanian, English or French',14,1)
END
INSERT INTO Books (title, language) VALUES (@title, @language)
COMMIT TRAN
SELECT 'Transaction committed'
END TRY

BEGIN CATCH
ROLLBACK TRAN
SELECT 'Transaction rollbacked'
END CATCH
END
```

But, pay attention, you have to insert values in all the tables from the relation m-n considered. First, you insert data in the table(s) Books and Authors, and then in the intermediate table (BooksAuthors). All these Insert's operations (for all these 3 tables), will be done in a single transaction.

Pentru prezentarea laboratorului pregătiţi teste ce acoperă scenarii de succes şi cu erori. Pregătiţi explicaţii detaliate ale scenariilor şi implementării.

You have to prepare scenarios for the verification of this homework: one with commit and one with rollback. The rollback can be obtain from the validation conditions given by the validation functions. You have to return the history of the operations executed. You can use Select/PRINT messages, or, use Select * from table_name, or, Select * from LogTable, or, any other solution that you consider.
Execution:

- with success (commit)

```
Select * from Books
EXEC AddBookAuthor 'Harry Potter and The Chamber of Secrets','English'
Select * from Books
```

	id	title	language
1	4	Panda	English
2	7	Codul lui Davinci	English

	(No column name)
1	Transaction committed

	id	title	language
1	4	Panda	English
2	7	Codul lui Davinci	English
3	9	Harry Potter an...	English

- without success (rollback)

```
Select * from Books
EXEC AddBookAuthor 'Inferno','Spanish'
Select * from Books
```

	id	title	language
1	4	Panda	English
2	7	Codul lui Davinci	English
3	9	Harry Potter and The Chamber of Secrets	English

	(No column name)
1	Transaction rollbacked

	ErrorNumber	ErrorMessage
1	50000	Language must be Romanian, English or French

	id	title	language
1	4	Panda	English
2	7	Codul lui Davinci	English
3	9	Harry Potter an...	English

Creaţi o procedură stocată ce inserează date pentru entităţi ce se află într-o relaţie m-n. Dacă o operaţie de inserare eşuează va trebui să se păstreze cât mai mult posibil din ceea ce s-a modificat până în acel moment. De exemplu, dacă se încearcă inserarea unei cărţi şi a autorilor acesteia, iar autorii au fost inseraţi cu succes însă apare o problemă la inserarea cărţii, atunci să se facă *roll-back* la inserarea de carte însă autorii acesteia să rămână în baza de date. (nota: 10)

Here, the transaction will be split into 3 transactions in the same stored procedure:

- First for the table Authors - with the validation also
- Second for the table Books – with the validation also
- Third for the table BooksAuthors – with the id's taken from both of the previous tables

The idea is that one can insert separately in each of the table. If we can add in Books, we add, and in Authors we cannot add, but this won't affect the add from Books. Each table is treat separately and do not affect the add of the others tables.
The execution has to be done for a success case and also for an un-success case.