Laborator 3

De asemenea, creaţi o nouă tabelă care să memoreze versiunea structurii bazei de date (presupunând că această versiune este pur şi simplu un număr întreg).
Se creaza un tabel Versiune (Vid int primary key identity, Nrversiune int) sau Versiune (nrversiune int) sau… cum se doreste

Uneori, după proiectarea unei baze de date, este necesară actualizarea structurii acesteia; din păcate însă nu toate actualizările efectuate se dovedesc a fi corecte, si modificările trebuie să fie anulate. În acest context, se cere dezvoltarea unui mecanism de versionare care să faciliteze tranziţia de la o versiune a bazei de date la alta.
Scrieţi un script SQL care va:
- modifica tipul unei coloane;
- adauga o costrângere de "valoare implicită" pentru un câmp;
- creea/şterge o tabelă;
- adăuga un câmp nou;
- creea/şterge o constrângere de cheie străină.

Aceste scripturi pot fi generate, însă va trebui să puteţi explica structura lor.
Pentru fiecare dintre scripturile de mai sus scrieţi/generaţi un alt script care implementează inversul operaţiei.

Versiune 1: modifica tipul unei coloane
Versiune 2: adauga o costrângere de "valoare implicită" pentru un câmp
Versiune 3: creeaza o tabelă
Versiune 4: adăuga un câmp nou
Versiune 5: creeaza o constrângere de cheie străină

Consideram un tabel:

STUDENT	CAIET
*Sid INT	*Cid
Nume VARCHAR(50)	Tip
Prenume	NrPagini
Varsta INT	Sid INT

Versiune	Invers Versiune
V1 ALTER TABLE Student ALTER COLUMN Varsta SMALLINT -- NOT NULL se poate pune doar daca este setat deja pentru coloana sau daca nu sunt introduce valori in tabel -- se recomanda schimbari din int-smallint-float sau varchar-nvarchar de dimensiuni diferite sau la fel, … conversii care se pot executa fara erori	IV1 ALTER TABLE Student ALTER COLUMN Varsta INT

V2 ALTER TABLE Student ADD CONSTRAINT df_Student DEFAULT 'Nume Student' FOR Nume	IV2 ALTER TABLE Student DROP CONSTRAINT df_Student
V3 CREATE TABLE Caiet(Cid INT PRIMARY KEY IDENTITY, Tip VARCHAR(30))	IV3 DROP TABLE Caiet
V4 ALTER TABLE Caiet ADD NrPagini INT	IV4 ALTER TABLE Caiet DROP COLUMN NrPagini
V5 -- daca s-a adaugat coloana pe care putem defini cheie straina la V3+V4, atunci realizam doar setarea acesteia; altfel, prima data adaugam coloana si apoi o setam ca si cheie straina ALTER TABLE Caiet ADD Sid INT NOT NULL ALTER TABLE Caiet ADD CONSTRAINT fk_Caiet FOREIGN KEY (Sid) REFERENCES Student(Sid) -- daca vrem sa adaugam intr-un tabel deja existent o cheie externa si acel tabel contine deja inregistrari, trebuie sa: adaugam coloana, introducem date pe ea prin cod si preluate de la campul ce este cheie primara din tabela ce va fi referita, setam aceasta coloana sa fie NOT NULL, adaugam constrangerea de cheie straina. La IV5 (inversiul versiunii 5) vom sterge fiecare element in ordine inversa: stergem constrangerea cheie straina, stergem coloana care a fost setata ca si cheie externa.	IV5 ALTER TABLE Caiet DROP CONSTRAINT fk_Caiet ALTER TABLE Caiet DROP COLUMN Sid

Creaţi pentru fiecare din scripturi câte o procedură stocată şi asiguraţi-vă ca numele acestora să fie simplu şi clar.

Fiecare script va fi pus intr-o procedura stocata, care se creaza si doar apoi se executa (ruleaza)

```
CREATE PROCEDURE V1 AS
        ALTER TABLE Student
        ALTER COLUMN Varsta SMALLINT
GO
```

Daca exista mai multe intructiuni se pune si BEGIN … END

Se recomanda si folosirea instructiunii PRINT 'mesaj' pentru a vizualiza mai bine rezultatul.

```
CREATE PROCEDURE V1 BEGIN AS
        ALTER TABLE Student
        ALTER COLUMN Varsta SMALLINT

        PRINT 'Coloana Varsta a fost adaugata in tabelul Student…'
END
```

GO

De asemenea, scrieţi o altă procedură stocată ce primeşte ca parametru un număr de versiune şi aduce baza de date la versiunea respectivă.

Procedura stocata va avea un parametru si se va executa astfel:

La inceput baza de date se presupune a fi intr-o versiune initiala. Consideram versiunea 0.

EXEC Principala 3 -> se verifica versiunea actuala a bazei de date – in tabelul Versiune – este 0 si atunci se executa V1, V2, V3, dupa care se actualizeaza si in tabelul Versiune

EXEC Principala 5 -> se verifica versiunea actuala a bazei de date – in tabelul Versiune – este 3 si atunci se executa V4, V5 dupa care se actualizeaza si in tabelul Versiune

EXEC Principala 2 -> se verifica versiunea actuala a bazei de date – in tabelul Versiune – este 5 si atunci se executa IV5, IV4, IV3, dupa care se actualizeaza si in tabelul Versiune

EXEC Principala -9 -> Mesaj de eroare (nu exista aceasta versiune) – trebuie tratate toate exceptiile

EXEC Principala 2 -> Mesaj ca deja se afla in aceasta versiune

Codul poate contine si mesaje legate de versiune si de schimbarea acesteia.

Declararea unei variabile se realizeaza folosind DECLARE @var INT si atribuirea unei valori se realizeaza cu SET @var=0.

Aceasta procedura stocata ar arata cam asa:

- validare parametru
- extragere intr-o variabila a numarului versiunii din tabelul Versiune (select @verNoua=…)
- daca versiunea din tabelul Versiune < @verNoua atunci se executa versionarile directe (V1, V2, …)
- altfel se executa versionarile inverse (IV5, IV4, …)
- se modifica versiunea in tabelul Versiune (update Versiune set…)

Versionarea necesita o intructiune repetitiva (gen WHILE) – in care cel mai usor se lucreaza cu o variabila in care se salveaza, pe rand, numele procedurilor stocate necesare versionarii – daca avem acelasi nume si, de exemplu, ultima parte diferita, se poate face o concatenare …

```
DECLARE @comanda VARCHAR(50);
SET @comanda='v'+ convert(…);  -- v1, v2, …
EXEC(@comanda)
-- se creste acea variabila de la v1, v2, …
```

Mult spor ☺