SQL DDL: Limbaj de definire a datelor

SEMINAR 1

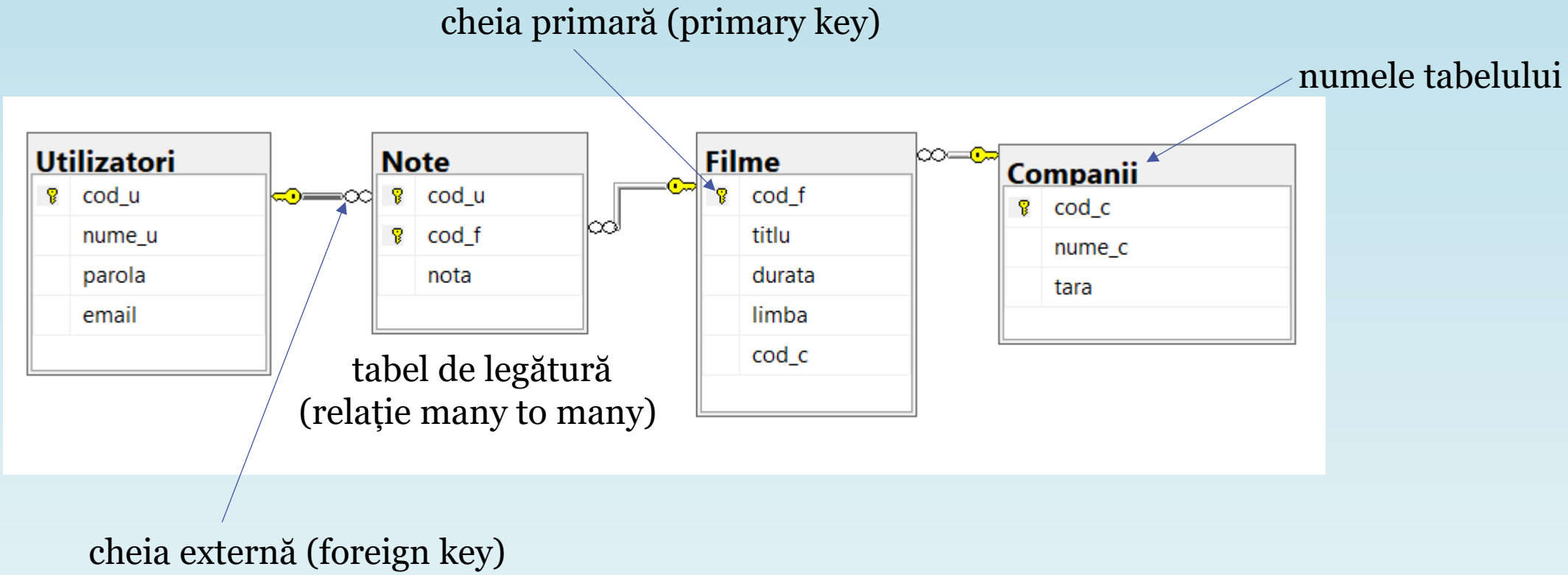
Limbajul SQL: DDL

- **DDL** (*Data Definition Language* - ***Limbaj de definire a datelor***) conține comenzi pentru creare/modificare/ștergere a:
  - bazelor de date
  - tabelelor
  - relaţiilor dintre tabele
  - constrângerilor
  - indecşilor

Problemă

- Să se creeze baza de date a unei aplicații care gestionează notele date de către utilizatori unor filme. **Entitățile** de interes pentru **domeniul problemei** sunt: *Utilizatori*, *Companii* și *Filme*. Fiecare companie are un **nume** și o **țară**. O companie poate produce mai multe filme, dar fiecare film este produs de către o singură companie. Fiecare film are un **titlu**, o **durată**, o **limbă** și este produs de către o singură **companie**. Utilizatorii pot da note mai multor filme, iar un film poate primi note de la mai mulți utilizatori. Fiecare utilizator are un **nume de utilizator**, o **parolă** și o **adresă de email**. Numele de utilizator și adresa de email sunt *unice*. Pentru numele de utilizator **nu** se poate specifica valoarea *NULL*. Fiecare utilizator poate da fiecărui film o singură notă. Valoarea notei este cuprinsă între 1 și 10.

Rezolvare: diagrama bazei de date relaționale



Rezolvare: Cod Transact-SQL

- În primul rând, este necesară crearea unei noi baze de date, utilizând instrucțiunea ***CREATE DATABASE***:

```
--Crearea bazei de date cu numele ProblemaFilme

CREATE DATABASE ProblemaFilme;

GO
```

Rezolvare: Cod Transact-SQL

- După ce am creat baza de date, este necesar să ne conectăm la aceasta înainte de a crea tabelele folosind instrucțiunea ***USE***:

```
--Conectarea la baza de date

USE ProblemaFilme;
```

- !!!Dacă **nu** ne conectăm la baza de date *ProblemaFilme*, instrucțiunile de creare a tabelelor ce urmează a fi executate vor crea aceste tabele în contextul unei alte baze de date (de obicei în baza de date *master*)

Rezolvare: Cod Transact-SQL

- Urmează crearea primului tabel din baza de date *ProblemaFilme*, folosind instrucțiunea ***CREATE TABLE***:

```
--Crearea tabelului Utilizatori
CREATE TABLE Utilizatori
(cod_u INT PRIMARY KEY IDENTITY,
nume_u VARCHAR(100) NOT NULL UNIQUE,
parola VARCHAR(100),
email VARCHAR(100)
);
```

Constrângere PRIMARY KEY

Proprietate IDENTITY

Constrângere UNIQUE

Constrângere NOT NULL

Tipul de date

Denumirea coloanei

Rezolvare: Cod Transact-SQL

- O altă variantă de creare a primului tabel din baza de date *ProblemaFilme*, folosind instrucțiunea ***CREATE TABLE:***

```
--Crearea tabelului Utilizatori
CREATE TABLE Utilizatori
(cod_u INT IDENTITY(1,1),
nume_u VARCHAR(100) NOT NULL,
parola VARCHAR(100),
email VARCHAR(100),
CONSTRAINT pk_Utilizatori PRIMARY KEY (cod_u),
CONSTRAINT uq_nume_u UNIQUE (nume_u)
);
```

Observații

- Fiecare tabel va conține mai multe coloane, iar fiecare **coloană** va avea un **tip de date**
- Fiecare tabel va avea un **identificator unic**, asigurat prin definirea unei **constrângeri PRIMARY KEY** pe tabel (aceasta poate fi definită pe una sau mai multe coloane)
- În cazul în care avem o **constrângere PRIMARY KEY** definită pe o singură coloană de tipul *INT*, putem folosi **proprietatea IDENTITY** pentru a genera valori unice în mod automat pentru această coloană
- Dacă **nu** dorim să permitem introducerea de valori *NULL* pentru o coloană, aplicăm **constrângerea NOT NULL** pe coloana respectivă

Observații

- **Constrângerea UNIQUE** asigură unicitatea valorilor la nivelul coloanei sau combinației de coloane pe care este definită
- Se pot defini mai multe **constrângeri UNIQUE** pe un tabel, dar doar o **singură constrângere PRIMARY KEY**
- Nu se pot specifica valori duplicate sau *NULL* pentru coloanele pe care a fost definită o **constrângere PRIMARY KEY**
- **Proprietatea IDENTITY(*seed*, *increment*):**
  - **seed** reprezintă valoarea folosită la inserarea primei înregistrări din tabel
  - **increment** reprezintă valoarea care se adaugă la valoarea *identity* generată pentru ultima înregistrare inserată în tabel

Rezolvare: Cod Transact-SQL

- În continuare, se va crea tabelul *Companii*, executând instrucțiunea de mai jos:

```
--Crearea tabelului Companii
CREATE TABLE Companii
(cod_c INT PRIMARY KEY IDENTITY(1,1),
nume_c NVARCHAR(100),
tara NVARCHAR(100)
);
```

Rezolvare: Cod Transact-SQL

- Apoi, se va crea tabelul *Filme*, executând instrucțiunea de mai jos:

```
--Crearea tabelului Filme
CREATE TABLE Filme
(cod_f INT PRIMARY KEY IDENTITY,
titlu NVARCHAR(200),
durata TIME,
limba NVARCHAR(100),
cod_c INT FOREIGN KEY REFERENCES Companii(cod_c) ON UPDATE CASCADE
ON DELETE CASCADE
);
```



Rezolvare: Cod Transact-SQL

- O altă variantă de a crea tabelul *Filme*, executând instrucțiunea de mai jos:

```
--Crearea tabelului Filme
CREATE TABLE Filme
(cod_f INT PRIMARY KEY IDENTITY,
titlu NVARCHAR(200),
durata TIME,
limba NVARCHAR(100),
cod_c INT,
CONSTRAINT fk_CompaniiFilme FOREIGN KEY (cod_c) REFERENCES
Companii(cod_c) ON UPDATE CASCADE ON DELETE CASCADE
);
```



Observații

- **Constrângerea FOREIGN KEY** se folosește pentru a crea relații între tabele
- O coloană pe care este definită o constrângere **FOREIGN KEY** este conectată la o coloană pe care este definită o constrângere **PRIMARY KEY** (de obicei dintr-un alt tabel)
- Constrângerea **FOREIGN KEY** este folosită pentru a preveni acțiuni care ar distruge legăturile dintre cele două tabele, dar și pentru a împiedica introducerea unor date invalide care nu se regăsesc în coloana pe care este definită constrângerea **PRIMARY KEY**
- Nu se pot face modificări în tabelul care conține constrângerea **PRIMARY KEY** dacă aceste modificări distrug legături spre date din tabelul care conține constrângerea **FOREIGN KEY**

Observații

- La crearea unei constrângeri **FOREIGN KEY**, se pot specifica acțiuni care să aibă loc în cazul operațiilor de **modificare** sau **ștergere** a valorilor din coloana pe care este definită constrângerea **PRIMARY KEY** din celălalt tabel:
  - NO ACTION (default)
  - CASCADE
  - SET NULL
  - SET DEFAULT

Rezolvare: Cod Transact-SQL

- În final, se va crea tabelul de legătură (denumit *Note*) dintre *Filme* și *Utilizatori*, în care vor fi stocate notele date filmelor de către utilizatori:

```
--Crearea tabelului Note

CREATE TABLE Note

(cod_u INT,

cod_f INT,

nota INT,

CONSTRAINT fk_UtilizatoriNote FOREIGN KEY (cod_u) REFERENCES
Utilizatori(cod_u),

CONSTRAINT fk_FilmeNote FOREIGN KEY (cod_f) REFERENCES Filme(cod_f),

CONSTRAINT pk_Note PRIMARY KEY (cod_u, cod_f)

);
```

Constrângere FOREIGN KEY

Constrângere PRIMARY KEY compusă din două coloane

Rezolvare: Cod Transact-SQL

- O altă variantă de a crea tabelul de legătură (denumit *Note*) dintre *Filme* și *Utilizatori*, în care vor fi stocate notele date filmelor de către utilizatori:

```
--Crearea tabelului Note
CREATE TABLE Note
(cod_u INT FOREIGN KEY REFERENCES Utilizatori(cod_u),
cod_f INT FOREIGN KEY REFERENCES Filme(cod_f),
nota INT,
CONSTRAINT pk_Note PRIMARY KEY (cod_u, cod_f)
);
```

Constrângere FOREIGN KEY

Constrângere PRIMARY KEY compusă din două coloane

Rezolvare: Cod Transact-SQL

- În enunțul problemei se specifică faptul că atât **numele utilizatorului** cât și **adresa de email** trebuie să fie **unice**
- Dacă am omis specificarea unei constrângeri **UNIQUE** la crearea tabelului, aceasta se poate crea și **ulterior** cu ajutorul instrucțiunii ***ALTER TABLE***:

```
/*Crearea unei constrângeri UNIQUE pe coloana email din tabelul Utilizatori
după crearea tabelului*/

ALTER TABLE Utilizatori

ADD CONSTRAINT uq_email UNIQUE (email);
```

Rezolvare: Cod Transact-SQL

- În enunțul problemei se menționează faptul că **nota** dată de către fiecare utilizator trebuie să aibă o **valoare** cuprinsă între **1** și **10**
- Această restricție se poate asigura cu ajutorul unei constrângeri **CHECK**, care poate fi impusă **ulterior** creării tabelului cu ajutorul instrucțiunii ***ALTER TABLE***:

```
/*Crearea unei constrângeri CHECK pe coloana nota din tabelul Note după ce a fost creat tabelul*/

ALTER TABLE Note

ADD CONSTRAINT ck_nota CHECK (nota>=1 AND nota<=10);
```

!!! Dacă există în tabel valori care nu se află în intervalul specificat de condiția din constrângerea CHECK de mai sus, instrucțiunea de creare a constrângerii va eșua

Observații

- **Constrângerea CHECK** se folosește pentru a limita intervalul de valori ce se pot introduce pentru o anumită coloană
- Se poate defini pe o coloană, iar în acest caz limitează valorile ce pot fi introduse pentru coloana respectivă
- Se pot crea și **constrângeri CHECK** care conțin restricții pentru mai multe coloane

Modificări la nivelul structurii bazei de date

- În anumite cazuri, vor fi necesare **modificări** la nivelul structurii bazei de date
- Dacă dorim să **adăugăm** o **coloană nouă** în tabelul *Note* în care vom stoca data și ora la care a fost adăugată nota, vom folosi din nou instrucțiunea ***ALTER TABLE***:

```
--Adăugarea unei coloane noi în tabelul Note

ALTER TABLE Note

ADD data_si_ora_adaugarii DATETIME;
```

Modificări la nivelul structurii bazei de date

- Dacă dorim să setăm o **valoare implicită** pentru o **coloană** (care să fie introdusă în mod automat atunci când nu se specifică o valoare pentru coloana respectivă la adăugarea unei înregistrări), vom folosi o constrângere **DEFAULT**:

```
--Adăugarea unei constrângeri DEFAULT pe coloana data_si_ora_adaugarii

ALTER TABLE Note

ADD CONSTRAINT df_data_si_ora_adaugarii DEFAULT GETDATE() FOR
data_si_ora_adaugarii;
```

- !!!În exemplul de mai sus, este folosită funcția sistem **GETDATE()** pentru generarea valorii implicite, deoarece se dorește folosirea datei și orei curente la momentul adăugării înregistrării în tabel

Modificări la nivelul structurii bazei de date

- Dacă dorim să modificăm tipul de date al unei coloane, vom folosi din nou instrucțiunea ***ALTER TABLE***:

```
--Modificarea tipului de date al coloanei titlu din tabelul Filme

ALTER TABLE Filme

ALTER COLUMN titlu NVARCHAR(220);
```

Modificări la nivelul structurii bazei de date

- Dacă dorim să eliminăm o constrângere dintr-un tabel, vom folosi instrucțiunea ***ALTER TABLE***:

```
/*Eliminarea constrângerii DEFAULT definită pe coloana data_si_ora_adaugarii
din tabelul Note*/

ALTER TABLE Note

DROP CONSTRAINT df_data_si_ora_adaugarii;
```

Modificări la nivelul structurii bazei de date

- Dacă dorim să ștergem o coloană dintr-un tabel, vom folosi instrucțiunea ***ALTER TABLE***:

```
--Ștergerea coloanei data_si_ora_adaugarii din tabelul Note

ALTER TABLE Note

DROP COLUMN data_si_ora_adaugarii;
```

Modificări la nivelul structurii bazei de date

- Dacă dorim să modificăm numele bazei de date, vom folosi instrucțiunea ***ALTER DATABASE***:

```
--Modificarea numelui bazei de date  ProblemaFilme

ALTER DATABASE ProblemaFilme

MODIFY Name=NoteFilme;
```

Ștergerea unui tabel din baza de date

- Dacă dorim să ștergem un tabel din baza de date, vom folosi instrucțiunea ***DROP TABLE***:

```
--Ștergerea tabelului Note

DROP TABLE Note;
```

Ștergerea bazei de date

- Dacă dorim să ștergem baza de date, vom folosi instrucțiunea ***DROP DATABASE***:

```
--Ștergerea bazei de date NoteFilme

USE master;

DROP DATABASE NoteFilme;
```

- !!!Este important să fim conectați la o altă bază de date în momentul în care ștergem baza de date *NoteFilme* de pe server, în caz contrar va apărea o eroare și ștergerea acesteia va eșua

Problemă propusă

- Să se creeze o bază de date care stochează informații despre un parc de distracții. Entitățile de interes pentru domeniul problemei sunt: **categorii de vizitatori**, **vizitatori**, **secțiuni** și **atracții**. Fiecare atracție din parcul de distracții are un **nume**, o **descriere**, o **vârstă minimă recomandată** și aparține unei singure secțiuni. Fiecare secțiune are un **nume** și o **descriere**. O secțiune poate conține mai multe atracții, dar fiecare atracție aparține unei singure secțiuni. Fiecare vizitator are un **nume**, o **adresă de email** și aparține unei singure categorii de vizitatori. Fiecare categorie de vizitatori are un **nume**. O categorie de vizitatori conține mai mulți vizitatori, dar fiecare vizitator aparține doar unei singure categorii. Fiecare vizitator poate vizita mai multe atracții, iar fiecare atracție poate fi vizitată de mai mulți vizitatori. Un vizitator poate da o singură notă fiecărei atracții pe care a vizitat-o. Nota este un număr real cuprins între 1 și 10.

Bibliografie

- https://learn.microsoft.com/en-us/sql/t-sql/statements/create-database-transact-sql?view=sql-server-ver16&tabs=sqlpool
- https://learn.microsoft.com/en-us/sql/t-sql/statements/alter-database-transact-sql?view=sql-server-ver16&tabs=sqlpool
- https://learn.microsoft.com/en-us/sql/t-sql/statements/drop-database-transact-sql?view=sql-server-ver16
- https://learn.microsoft.com/en-us/sql/t-sql/statements/create-table-transact-sql?view=sql-server-ver16
- https://learn.microsoft.com/en-us/sql/t-sql/statements/alter-table-transact-sql?view=sql-server-ver16

Bibliografie

- https://learn.microsoft.com/en-us/sql/t-sql/statements/drop-table-transact-sql?view=sql-server-ver16
- https://learn.microsoft.com/en-us/sql/relational-databases/tables/primary-and-foreign-key-constraints?view=sql-server-ver16
- https://learn.microsoft.com/en-us/sql/relational-databases/tables/unique-constraints-and-check-constraints?view=sql-server-ver16
- https://learn.microsoft.com/en-us/sql/relational-databases/tables/specify-default-values-for-columns?view=sql-server-ver16