Constrangeri SQL - Teorie, Exemple Rezolvate si Probleme Propuse

1 Partea Teoretica

Constrangerile SQL sunt reguli aplicate coloanelor unei baze de date pentru a asigura corectitudinea, integritatea si validitatea datelor. Ele sunt definite in timpul crearii tabelelor si pot fi aplicate pentru a controla actiunile asupra datelor.

Principalele constrangeri SQL sunt:

- `NOT NULL`: Asigura ca o coloana nu poate avea valoarea `NULL`.
- `UNIQUE`: Asigura ca toate valorile dintr-o coloana sunt distincte.
- `PRIMARY KEY`: Identifica in mod unic fiecare rand dintr-un tabel.
- `FOREIGN KEY`: Mentine integritatea referentiala intre tabele.
- `CHECK`: Impune o conditie care trebuie respectata de valorile unei coloane.
- `DEFAULT`: Stabileste o valoare implicita pentru o coloana.

2 Exemple Rezolvate

2.1 Exemplul 1: Constrangerea `NOT NULL`

Sa presupunem ca dorim sa cream o tabela `Student` in care toate valorile pentru `StudentID` si `Nume` trebuie sa fie obligatorii.

Cod SQL:

```
CREATE TABLE Student (
StudentID INT NOT NULL,
Nume VARCHAR(50) NOT NULL,
Email VARCHAR(100)
);
```

Rezultat: Tabela `Student` este creata astfel incat `StudentID` si `Nume` nu pot fi `NULL`, dar `Email` poate accepta valori `NULL`.

2.2 Exemplul 2: Constrangerea `PRIMARY KEY`

Dorim sa cream o tabela `Produs` care sa aiba o coloana `ProdusID` ce identifica unic fiecare produs.

Cod SQL:

```
CREATE TABLE Produs (
ProdusID INT PRIMARY KEY,
NumeProdus VARCHAR(100),
Pret DECIMAL(10, 2)
);
```

Rezultat: `ProdusID` devine cheia primara a tabelului `Produs`, asigurand unicitatea fiecarei inregistrari.

2.3 Exemplul 3: Constrangerea `FOREIGN KEY`

Vrem sa gestionam relatia intre tabelele `Client` si `Comanda`, unde fiecare comanda este asociata unui client.

Cod SQL:

```
CREATE TABLE Client (
ClientID INT PRIMARY KEY,
Nume VARCHAR(100)
);

CREATE TABLE Comanda (
ComandaID INT PRIMARY KEY,
ClientID INT,
DataComanda DATE,
FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
);
```

Rezultat: `ClientID` din tabela `Comanda` face referinta la `ClientID` din tabela `Client`, mentinand integritatea referentiala.

2.4 Exemplul 4: Constrangerea `CHECK`

Dorim sa cream o tabela `Angajat` unde salariul fiecarui angajat trebuie sa fie pozitiv.

Cod SQL:

```
CREATE TABLE Angajat (
AngajatID INT PRIMARY KEY,
Nume VARCHAR(100),
Salariu DECIMAL(10, 2) CHECK (Salariu > 0)
);
```

Rezultat: Nu se pot introduce valori negative pentru coloana `Salariu`.

2.5 Exemplul 5: Constrangerea `DEFAULT`

Cream o tabela `Factura` in care statusul implicit al unei facturi este `'Neachitat'`.

Cod SQL:

```
CREATE TABLE Factura (
FacturaID INT PRIMARY KEY,
DataFactura DATE DEFAULT CURRENT_DATE,
Status VARCHAR(20) DEFAULT 'Neachitat'
);
```

Rezultat: Coloana `DataFactura` are valoarea implicita `CURRENT_DATE`, iar `Status` are valoarea `'Neachitat'` daca nu se specifica alte valori.

3 Probleme Propuse

1. **Exercitiul 1: Validare complexa cu constrangeri `CHECK`**
   Creati o tabela `Angajat` cu urmatoarele coloane:

   - `AngajatID` (`INT`, `PRIMARY KEY`),
   - `Salariu` (`DECIMAL(10, 2)`, `CHECK` (`Salariu` ¿ 0)),
   - `NivelExperienta` (`INT`, `CHECK` (`NivelExperienta` BETWEEN 1 AND 10)).

   Adaugati o constrangere `CHECK` astfel incat `Salariu` sa fie cel putin 3000 pentru `NivelExperienta` mai mare de 5.

2. **Exercitiul 2: Relatii ciclice cu chei externe**
   Creati doua tabele, `Angajat` si `Manager`. Fiecare angajat poate avea un manager, iar fiecare manager este totodata angajat. Definiti o cheie externa in tabelul `Angajat` care face referinta la `ManagerID` din acelasi tabel.

3. **Exercitiul 3: Constrangeri conditionale**
   Creati o tabela `Evaluare` cu coloanele:

   - `EvaluareID` (`INT`, `PRIMARY KEY`),
   - `TipEvaluare` (`VARCHAR(20)`, valorile posibile fiind `'Teorie'` sau `'Practica'`),
   - `Nota` (`DECIMAL(4, 2)`).

   Adaugati o constrangere `CHECK` astfel incat notele pentru `Teorie` sa fie intre 1 si 10, iar cele pentru `Practica` intre 1 si 5.

4. **Exercitiul 4: Stergerea in cascada cu `ON DELETE CASCADE`**
   Creati doua tabele, `Client` si `Comanda`. Tabelul `Client` contine `ClientID` (`PRIMARY KEY`). Tabelul `Comanda` contine `ComandaID` (`PRIMARY KEY`) si `ClientID` (`FOREIGN KEY` catre `Client` cu optiunea `ON DELETE CASCADE`). Testati ce se intampla cand un client este sters.

5. **Exercitiul 5: Chei externe multiple**
   Creati un tabel intermediar `AlocareProiect` intre tabelele `Angajat`, `Proiect` si `Departament`. Coloanele tabelului `AlocareProiect` trebuie sa respecte urmatoarele:

   - `AngajatID` (`FOREIGN KEY` catre `Angajat`),
   - `ProiectID` (`FOREIGN KEY` catre `Proiect`),
   - `DepartamentID` (`FOREIGN KEY` catre `Departament`),
   - `OraAlocare` (`TIME`, intre `08:00` si `18:00`).

6. **Exercitiul 6: Validare complexa cu `CHECK` pe mai multe coloane**
   Creati o tabela `Comanda` cu urmatoarele coloane:

   - `ComandaID` (`INT`, `PRIMARY KEY`),
   - `ProdusID` (`INT`),
   - `Cantitate` (`INT`),
   - `PretTotal` (`DECIMAL(10, 2)`).

   Adaugati o constrangere `CHECK` care sa valideze ca `PretTotal = Cantitate * PretUnitar`, unde `PretUnitar` este o valoare implicita de 100 pentru fiecare produs.

7. **Exercitiul 7: Constrangeri compuse `UNIQUE`**
   Creati o tabela `InscriereStudent` cu urmatoarele coloane:

   - `StudentID` (`INT`, cheie externa referita la tabela `Student`),
   - `CursID` (`INT`, cheie externa referita la tabela `Curs`),
   - `DataInscriere` (`DATE`),
   - `NotaFinala` (`DECIMAL(4, 2)`).

   Asigurati-va ca fiecare student nu se poate inscrie la acelasi curs de mai multe ori, utilizand o constrangere compusa `UNIQUE`.

8. **Exercitiul 8: Relatii complexe intre tabele**
   Creati urmatoarele tabele:

   - `Profesor` (`ProfesorID`, `Nume`),

- `Curs` (`CursID`, `NumeCurs`, `ProfesorID`, `FOREIGN KEY` catre `Profesor`),
- `StudentCurs` (`StudentID`, `CursID`, `FOREIGN KEY` catre `Curs`).

Asigurati-va ca fiecare student poate participa doar la cursuri sustinute de un profesor.

9. **Exercitiul 9: Combinarea constrangerilor**
   Creati o tabela `Furnizor` cu urmatoarele coloane:
   - `FurnizorID` (`INT`, `PRIMARY KEY`),
   - `Nume` (`VARCHAR(100)`, `NOT NULL`),
   - `CUI` (`VARCHAR(10)`, `UNIQUE`),
   - `Rating` (`DECIMAL(3, 1)`, `CHECK` (`Rating` BETWEEN 0 AND 5)).
10. **Exercitiul 10: Adaugarea unei constrangeri dupa creare**
    Creati o tabela `Plata` cu coloanele:
    - `PlataID` (`INT`, `PRIMARY KEY`),
    - `Suma` (`DECIMAL(10, 2)`).

    Adaugati ulterior o constrangere `CHECK` care sa valideze ca `Suma` este mai mare decat 0.