Cele 12 Reguli ale lui E.F. Codd

1 Introducere

Cele 12 reguli ale lui E.F. Codd definesc principiile fundamentale ale sistemelor de gestiune a bazelor de date relaționale (DBMS). Aceste reguli asigură că un sistem relațional este implementat complet și corect.

2 Regulile lui E.F. Codd

2.1 Regula 0: Fundația Modelului Relațional

Orice sistem care pretinde că este un DBMS relațional trebuie să utilizeze doar funcționalități bazate pe modelul relațional.

Exemplu: Crearea unei tabele relaționale.

```
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    Nume VARCHAR(100),
    Varsta INT
);
```

Această comandă demonstrează utilizarea tabelelor ca structură fundamentală.

2.2 Regula 1: Informația

Toate informațiile dintr-o bază de date sunt reprezentate logic sub formă de tabele.

Exemplu: Salvarea informațiilor despre studenți într-o tabelă.

```
SELECT * FROM Student;
```

Toate datele sunt vizualizate și organizate logic în rânduri și coloane.

2.3 Regula 2: Acces Garanatizat

Toate datele trebuie să fie accesibile folosind o combinație de nume de tabele, coloane și chei primare.

Exemplu: Accesarea unui student după `StudentID`.

```
SELECT Nume, Varsta
FROM Student
WHERE StudentID = 1;
```

Fiecare înregistrare poate fi accesată în mod unic.

2.4 Regula 3: Tratamentul Sistematic al Valorilor `NULL`

Valorile `NULL` trebuie tratate consecvent, ca valori necunoscute sau inadecvate.

Exemplu: Identificarea rândurilor cu valori `NULL`.

```
SELECT *
FROM Student
WHERE Varsta IS NULL;
```

Această interogare returnează rândurile unde vârsta nu este specificată.

2.5 Regula 4: Catalog Activ

Structura bazei de date trebuie să fie stocată într-un catalog accesibil utilizatorilor.

Exemplu: Interogarea metadatelor bazei de date.

```
SELECT TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'universitate';
```

Catalogul activ permite accesul la informații despre structura bazei de date.

2.6 Regula 5: Sub-limbaj Complet de Date

Sistemul trebuie să suporte un limbaj complet pentru definirea, manipularea și interogarea datelor.

Exemplu: SQL ca limbaj complet.

```
-- Creare tabel
CREATE TABLE Curs (
    CursID INT PRIMARY KEY,
    Denumire VARCHAR(100)
);

-- Inserare date
INSERT INTO Curs VALUES (101, 'Matematica');

-- Interogare
SELECT * FROM Curs;

-- tergere date
DELETE FROM Curs WHERE CursID = 101;
```

Acest exemplu demonstrează capacitatea SQL de a defini și manipula date.

2.7 Regula 6: Actualizarea Vederilor

Vederile definite pe mai multe tabele ar trebui să poată fi actualizate.

Exemplu: Crearea și actualizarea unei vederi.

```
CREATE VIEW VedereStudent AS
SELECT StudentID, Nume
FROM Student;

-- Actualizare prin vedere
UPDATE VedereStudent
SET Nume = 'Ion-Popescu'
WHERE StudentID = 1;
```

2.8 Regula 7: Inserare, Actualizare și Ștergere la Nivel Logic

Informațiile trebuie să poată fi adăugate, actualizate și șterse utilizând tabele logice.

Exemplu: Adăugarea unui nou student.

```
INSERT INTO Student (StudentID, Nume, Varsta)
VALUES (2, 'Maria-Ionescu', 20);
```

Exemplu: Ștergerea unui student.

```
DELETE FROM Student
WHERE StudentID = 2;
```

2.9 Regula 8: Independența Fizică a Datelor

Modificările fizice ale bazei de date nu ar trebui să afecteze aplicațiile utilizatorilor.
Exemplu: Adăugarea unui index pentru optimizarea interogărilor.

```
CREATE INDEX idx_varsta
ON Student(Varsta);
```

Indexul optimizează performanța, fără a afecta structura logică.

2.10 Regula 9: Independența Logică a Datelor

Modificările logicii bazei de date nu ar trebui să afecteze aplicațiile existente.
Exemplu: Adăugarea unei coloane noi în tabelă.

```
ALTER TABLE Student
ADD Email VARCHAR(100);
```

Adăugarea unei coloane nu afectează interogările existente care nu utilizează această coloană.

2.11 Regula 10: Independența Integrității

Constrângerile de integritate trebuie să fie definite la nivel logic și nu prin aplicații.
Exemplu: Definirea unei constrângeri `CHECK`.

```
CREATE TABLE Nota (
    NotaID INT PRIMARY KEY,
    Valoare DECIMAL(4, 2) CHECK (Valoare BETWEEN 1 AND 10)
);
```

2.12 Regula 11: Independența Distribuirii

Utilizatorii ar trebui să poată accesa o bază de date distribuită fără a fi afectați de modul în care sunt distribuite datele.
Exemplu: Conectarea la o bază de date distribuită.

```
SELECT *
FROM Student
WHERE Varsta > 18;
```

Acest exemplu funcționează indiferent de distribuirea fizică a datelor.

2.13 Regula 12: Non-subversivitate

Toate operațiile asupra bazei de date trebuie să fie efectuate utilizând limbajul relațional al sistemului.
Exemplu: Ștergerea unei înregistrări utilizând SQL.

```
DELETE FROM Student
WHERE StudentID = 1;
```

Operațiile directe asupra fișierelor de sistem sau metadatelor sunt interzise.