Tutoriat SQL Oracle: Forme Normale

1 Introducere

În baze de date relaționale, **normalizarea** este procesul prin care organizăm datele pentru a reduce redundanța și a preveni anomaliile de inserare, ștergere sau actualizare. Scopul este să avem tabele clare, curate și ușor de întreținut.

Hai să vedem cum funcționează, cu exemple simple și concrete!

2 Prima Formă Normală (1NF)

Definiție: O tabelă este în 1NF dacă:

- Toate atributele conțin valori atomice (fără liste, fără seturi).
- Nu există grupuri de repetare.

Exemplu inițial (NE-normalizat):

Student	Cursuri
Ana	BD, Java
Ion	Java
Maria	BD, C++, Java

Table 1: Tabelă NE-normalizată

De ce nu respectă 1NF: Această tabelă nu respectă Prima Formă Normală deoarece:

- Coloana **Cursuri** conține valori non-atomice (liste de cursuri separate prin virgulă).
- Cursurile reprezintă un grup de repetare - un student poate avea mai multe cursuri.
- Acest design face dificilă interogarea, filtrarea și sortarea după un anumit curs.
- De exemplu, pentru a găsi toți studenții care studiază Java, ar trebui să folosim operatori LIKE sau funcții de tip string pentru a căuta subșiruri.

După normalizare (1NF):

Student	Curs
Ana	BD
Ana	Java
Ion	Java
Maria	BD
Maria	C++
Maria	Java

Table 2: Tabelă în 1NF

3 A Doua Formă Normală (2NF)

Definiție: O tabelă este în 2NF dacă:

- Este deja în 1NF.
- Fiecare atribut non-cheie depinde **complet** de cheia primară (nu parțial).

Exemplu inițial (doar 1NF):

(StudentID, CursID)	NumeStudent	NumeCurs
(1, BD)	Ana	Baze de Date
(1, Java)	Ana	Java
(2, Java)	Ion	Java

Table 3: Tabelă în 1NF cu dependență parțială

De ce nu respectă 2NF: Această tabelă este în 1NF (toate valorile sunt atomice), dar nu respectă 2NF deoarece:

- Cheia primară este compusă din **(StudentID, CursID)**.
- Atributul **NumeStudent** depinde doar de **StudentID**, nu de întreaga cheie primară. Acest lucru reprezintă o dependență parțială.
- Similar, **NumeCurs** depinde doar de **CursID**, nu de întreaga cheie.
- Această structură creează redundanță: numele studentului Ana apare de două ori, iar numele cursului Java apare de două ori.
- Dacă actualizăm numele unui student într-un rând, trebuie să actualizăm și celelalte rânduri pentru a menține consistența (anomalie de actualizare).

După normalizare (2NF):

- Tabelă **Studenti**(StudentID, NumeStudent)
- Tabelă **Cursuri**(CursID, NumeCurs)
- Tabelă **Inscrieri**(StudentID, CursID)

4 A Treia Formă Normală (3NF)

Definiție: O tabelă este în 3NF dacă:

- Este deja în 2NF.
- Nu există dependențe tranzitive între atribute (atributele non-cheie depind direct de cheia primară, nu prin alte atribute).

Exemplu inițial:

StudentID	FacultateID	NumeFacultate
1	10	Informatica
2	10	Informatica
3	20	Electrotehnica

Table 4: Tabelă în 2NF cu dependență tranzitivă

De ce nu respectă 3NF: Această tabelă este în 2NF (nu există dependențe parțiale), dar nu respectă 3NF deoarece:

- Există o dependență tranzitivă: **StudentID** $\rightarrow$ **FacultateID** $\rightarrow$ **NumeFacultate**.

- **NumeFacultate** depinde de cheia primară **StudentID** doar indirect, prin intermediul atributului **FacultateID**.
- Această structură cauzează redundanță: numele facultății "Informatica" apare de două ori.
- Dacă schimbăm numele unei facultăți, trebuie să actualizăm toate rândurile pentru acea facultate (anomalie de actualizare).
- Dacă ștergem ultimul student dintr-o facultate, pierdem informația despre facultatea respectivă (anomalie de ștergere).

După normalizare (3NF):

- Tabelă **Studenti**(StudentID, FacultateID)
- Tabelă **Facultati**(FacultateID, NumeFacultate)

5 Forma Normală Boyce-Codd (BCNF)

Definiție: O tabelă este în BCNF dacă pentru orice dependență funcțională X → Y, X este supercheie.
Exemplu problemă:

Profesor	Sala	Curs
Popescu	A101	BD
Ionescu	B202	Java
Popescu	A101	SQL

Table 5: Tabelă care nu este BCNF

De ce nu respectă BCNF: Această tabelă este în 3NF, dar nu respectă BCNF deoarece:

- Avem următoarele dependențe funcționale:
  - **(Profesor, Curs)** → **Sala** (un profesor predă un curs într-o anumită sală)
  - **(Profesor, Sala)** → **Curs** (un profesor într-o anumită sală predă un anumit curs)
  - **Profesor** → **Sala** (un profesor predă doar într-o singură sală, indiferent de curs)
- Observăm că **Profesor** determină **Sala**, dar **Profesor** nu este o cheie candidat.
- Cheia candidat este **(Profesor, Curs)** sau **(Sala, Curs)**.
- Această structură cauzează redundanță: sala "A101" pentru profesorul Popescu apare de două ori.
- Dacă profesorul Popescu își schimbă sala, trebuie să actualizăm toate rândurile pentru acel profesor (anomalie de actualizare).

După normalizare (BCNF):

- Tabelă **ProfesoriSali**(Profesor, Sala)
- Tabelă **CursuriProfesori**(Profesor, Curs)

6 A Patra Formă Normală (4NF)

Definiție: O tabelă este în 4NF dacă este în BCNF și nu conține dependențe multi-valuate (adică o cheie determină două seturi de valori independente).
Exemplu inițial:
De ce nu respectă 4NF: Această tabelă este în BCNF, dar nu respectă 4NF deoarece:

- Avem dependențe multi-valuate:
  - **StudentID** ↠ **Hobby** (un student poate avea mai multe hobby-uri)
  - **StudentID** ↠ **Limba** (un student poate cunoaște mai multe limbi)

StudentID	Hobby	Limba
1	Fotbal	Engleză
1	Fotbal	Franceză
1	Lectură	Engleză
1	Lectură	Franceză

Table 6: Tabelă cu dependență multi-valuată

- Limba și Hobby sunt independente între ele - nu există o relație directă între ce hobby are un student și ce limbă cunoaște.
- Această structură creează un produs cartezian: pentru fiecare hobby al unui student, tabelul conține toate limbile cunoscute de student.
- Dacă un student cunoaște 5 limbi și are 5 hobby-uri, avem 25 de rânduri pentru acest student (redundanță excesivă).
- Dacă adăugăm un nou hobby pentru un student, trebuie să adăugăm câte un rând pentru fiecare limbă cunoscută (anomalie de inserare).

După normalizare (4NF):

- Tabelă **StudentHobby**(StudentID, Hobby)
- Tabelă **StudentLimba**(StudentID, Limba)

7 A Cincea Formă Normală (5NF)

Definiție: O tabelă este în 5NF dacă este în 4NF și nu poate fi descompusă mai departe fără a pierde informație; orice dependență de tip join este doar o consecință a cheilor.

Exemplu inițial:

ProdusID	FurnizorID	PachetID
P1	F1	B1
P1	F2	B2
P2	F1	B2

Table 7: Tabelă combinată

De ce nu respectă 5NF: Această tabelă este în 4NF, dar nu respectă 5NF deoarece:

- Există o dependență de join ciclică între cele trei coloane: relația ternară între **ProdusID**, **FurnizorID** și **PachetID** conține constrângeri implicite.
- Nu toate combinațiile posibile de Produs-Furnizor-Pachet sunt valide sau prezente în tabel.
- Tabelul implică restricții care nu sunt evidente din structura sa:
  - Dacă produsul P1 este furnizat de F1 și face parte din pachetul B1
  - Și produsul P1 este furnizat de F2 și face parte din pachetul B2
  - Și produsul P2 este furnizat de F1 și face parte din pachetul B2
  - Atunci pot exista restricții pentru alte combinații care nu sunt explicit capturate
- Această structură poate cauza anomalii la join-uri multiple și redundanță la nivel de relații.
- Dacă se schimbă o relație (de exemplu, un produs nu mai este furnizat de un anumit furnizor), trebuie să actualizăm mai multe rânduri pentru a menține consistența.

După normalizare (5NF):

- Tabelă **ProdusFurnizor**(ProdusID, FurnizorID)
- Tabelă **FurnizorPachet**(FurnizorID, PachetID)
- Tabelă **ProdusPachet**(ProdusID, PachetID)

8 Concluzie

Normalizarea este cheia pentru baze de date curate și performante. În Oracle SQL, după ce modelezi corect în forme normale, tabelele sunt mai ușor de întreținut și interogările sunt mai clare!

Fiecare formă normală rezolvă un tip specific de problemă:

- **1NF**: Elimină grupurile de repetare și valorile non-atomice.
- **2NF**: Elimină dependențele parțiale față de cheia primară.
- **3NF**: Elimină dependențele tranzitive între atribute non-cheie.
- **BCNF**: Elimină anomaliile rămase din dependențele funcționale.
- **4NF**: Elimină dependențele multi-valuate independente.
- **5NF**: Elimină dependențele de join care nu rezultă direct din chei.

Acesta a fost tutoriatul nostru! Dacă vrei, îți pot adăuga exemple de `CREATE TABLE` Oracle pentru fiecare tabel normalizat.