UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Proiect nr. 154/323 cod SMIS – 4428 cofinanțat de prin Fondul European de Dezvoltare Regională “Investiții pentru viitorul dumneavoastră”.

Programul Operaţional Sectorial Creşterea Competitivităţii Economice - POS CCE

UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

POLITEHNICA DIN BUCUREȘTI
UNIVERSITATEA
1818

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Baze de date

17. Constrângeri asupra bazelor de date. Erori posibile

Introducere

Comenzile pentru crearea şi definirea de structuri tabelare sunt comenzi de definire a datelor (Data Definition Language – DDL) şi permit crearea, dar şi relaționarea lor într-o bază de date.

Structura unei tabele este dată de următoarele specificații de definire:

- definirea coloanelor
- definirea constrângerilor de integritate
- definirea tablespace-lui unde se creează
- definirea parametrilor

Constrângerile de integritate reprezintă anumite reguli de funcționare, care trebuie respectate la nivel de tabel, sau în relațiile cu alte tabele. Aceste reguli sunt verificate automat, în cazul operațiilor de inserare, stergere sau modificare şi în cazul în care nu se validează, sistemul de gestiune generează o eroare şi tranzacția nu se efectuează.

Tipuri de constrângeri de integritate

În Oracle, constrăngerile de integritate pot fi :

- **NOT NULL** – înregistrările nu pot conține valori nule
- **UNIQUE** – defineşte o cheie unică pe una sau mai multe coloane (nu pot fi mai multe înregistrări cu aceleaşi valori pe coloanele respective)
- **PRIMARY KEY** – defineşte o cheie primară la nivel de coloană sau tabelă (nu pot fi mai multe înregistrări cu aceeaşi cheie primară).
- **FOREIGN KEY** – defineşte o cheie externă (tabela relaționează cu altă tabelă, pe o cheie unică sau cheie primară)
- **CHECK** – forțează o condiție pe coloană

Caracteristicile constrângerilor de integritate

- Fiecare constrângere va avea un nume dat de user, sau generat de sistem;
- Constrângerile pot fi activate sau dezactivate cu comanda ALTER TABLE;
- Constrângerile pot fi adăugate sau şterse şi după crearea tabelului;
- Informațiile legate de constrângeri se păstrează în dicționarul bazei de date;
- Violarea unei constrângeri generează o eroare de sistem.

Constrângerea NOT NULL

- Se aplică numai la nivel de coloane şi verifică dacă înregistrările au valori nule pe coloanele respective, forțând un cod de eroare care anulează tranzacția;
- Când se creează constrângeri pe o cheie primară, se creează automat şi o costrângere NOT NULL pe coloanele respective (o cheie primară nu trebuie să conțină valori nule pe coloanele care o definesc).
- Sintaxa la nivel de coloană este :

column_name [CONSTRAINT constraint_name] NOT NULL

unde

- column_name – este numele unei coloane din tabel;
- constraint_name – este numele constrângerii.

Exemplu:

- Să creăm un tabel pentru tipurile de funcții într-o companie:

```
SQL>CREATE TABLE functii
      ( cod_functie number(2)  CONSTRAINT NL NOT NULL,
        den_functie varchar2(20) NOT NULL,
        data_vigoare date );
```

- Funcționalitatea coloanelor cod_functie şi den_functie este aceeaşi (nu acceptă valori nule), dar construcția constrângerii pentru cod_functie permite activarea sau dezactivarea constrângerii **NL**.

Constrângerea UNIQUE

- Se foloseşte când vrem ca o coloană, sau perechi de coloane, să nu conțină valori duplicate. Verificarea se face numai pentru înregistrări cu valori nenule, deoarece constrângerea permite inserarea de valori nule în coloanele respective.
- În mod automat, se creează şi un index pe coloanele definite cheii unice, ceea ce duce la mărirea vitezei de interogare pe tabel.
- Sintaxa la nivel de coloană este:

  column_name [CONSTRAINT constraint_name] UNIQUE

- Sintaxa la nivel de tabel este:

  [, CONSTRAINT constraint_name] UNIQUE (col1, col 2,..)

Exemplu:

- Să creăm tabela functii definind unicitate pe anumite coloane:

```
SQL>CREATE TABLE functii
        ( cod_functie  number(2)    CONSTRAINT UK_FUN UNIQUE,
          den_functie  varchar2(20) UNIQUE,
          data_vigoare date ) ;
```

- Putem să facem şi următoarea construcție la nivel de tabel:

```
SQL>CREATE TABLE functii
        ( cod_functie  number(2),
          den_functie  varchar2(20),
          data_vigoare date,
          CONSTRAINT UK_FUN UNIQUE (cod_functie,den_functie) );
```

- În prima construcție, putem insera oricâte înregistrări cu valori nule pe coloanele definite unice, dar în construcția a doua, nu putem să inserăm înregistrări cu una dintre coloane nulă şi cealaltă să nu respecte unicitatea.

Constrângerea PRIMARY KEY

- Se foloseşte pentru definirea cheii primare la nivel de coloană (când cheia conține o singură coloană) sau la nivel de tabel (când cheia este compusă pe mai multe coloane). Un tabel poate avea o singură cheie primară şi nu acceptă valori nule pentru nicio coloană care o definesc. Când se creează o cheie primară, se creează în mod automat şi un index pentru a scurta timpul de răspuns în cazul unei interogări.
- Sintaxa la nivel de coloană este:

column [CONSTRAINT constraint_name] PRIMARY KEY

- Sintaxa la nivel de tabel este:

[, CONSTRAINT constraint_name] PRIMARY KEY (col1, col 2,..)

Exemplu:

- Dacă vrem să creăm un nomenclator de funcții, în care fiecare funcție să aibă cod unic, folosim următoarea construcție :

```
SQL>CREATE TABLE functii
      ( cod_functie  number(2) CONSTRAINT PK PRIMARY KEY,
        den_functie  varchar2(20),
        data_vigoare date );
```

- Când dorim ca un cod de funcție să poată fi utilizat de mai multe ori, trebuie să definim cheia pe mai multe coloane, de exemplu pe codul funcției şi data la care intră în vigoare codificarea, folosind construcția:

```
SQL>CREATE TABLE functii
     ( cod_functie  number(2) ,
       den_functie  varchar2(20),
       data_vigoare date,
       CONSTRAINT PK_FUN PRIMARY KEY (cod_functie,data_vigoare) )
```

Constrângerea FOREIGN KEY

- Acest tip de constrângere se foloseşte pentru relaționarea unui tabel cu unul sau mai multe tabele, verificând dacă valorile conținute în coloanele definite de FOREIGN KEY (cheie străină sau cheie externă) sunt cuprinse în valorile coloanelor altei tabele care trebuie să fie definite UNIQUE sau PRIMARY KEY. Mai exact, o relaționare pe cheie externă se poate face numai pe o cheie primară sau unică .
- Sintaxa la nivel de coloană este:

column [CONSTRAINT constraint_name]

REFERENCES table(column)

[ON DELETE CASCADE | ON DELETE SET NULL]

- Sintaxa la nivel de tabel este:

column [CONSTRAINT constraint_name]

FOREIGN KEY (col1, col2, ...)

REFERENCES table(col1, col2, ..)

[ON DELETE CASCADE | ON DELETE SET NULL]

Exemplu:

- Dacă vrem să creăm tabelul angajati şi să o relaționăm cu tabelele departamente şi funcții, trebuie să facem următoarea construcție la nivel de coloană:

```
SQL>CREATE TABLE angajati
    ( id_ang      number(4)     PRIMARY KEY,
      nume        varchar2(30),
      functie     varchar2(20)  REFERENCES functii(den_functie),
      id_sef      number(4)     REFERENCES angajati(id_ang),
      data_ang    date,
      salariu     number(7,2),
      comision    number(7,2),
      id_dep      number(2)     REFERENCES departamente(id_dep) );
```

- Pentru a se putea face relaționarea, trebuie ca tabela functii să aibă cheie primară sau unică pe coloana den_functie, iar tabela departamente să aibă cheie primară sau unică pe coloana id_dep. Se observă că tabelul este relaționat cu el însuşi după coloanele id_sef şi id_ang.

Reguli de funcționare

- Când se face relaționarea între două tabele, trebuie avute în vedere următoarele reguli de funcționare :
- Inserarea unei linii într-un tabel relaționat (pe care am definit FOREIGN KEY) nu se poate face dacă nu există decât o singură linie în tabelul de referință (în care am definit PRIMARY KEY sau UNIQUE) corespunzător coloanelor de relaționare;
- Ştergerea unei linii din tabelul de referință nu se poate face atâta timp cât există linii relaționate pe linia respectivă în tabelul relaționat;
- Dacă s-a folosit optiunea ON DELETE CASCADE, se poate şterge o linie din tabelul de referință, dar se şterg automat toate liniile relaționate pe linia respectivă, din toate tabelele relaționate;
- În cazul unui tabel relaționat cu el însuşi, când se şterge o linie, care este referită, toate coloanele devin nule în liniile relaționate;
- Regulile de mai sus sunt valabile şi în cazul relaționării pe coloane.

Constrângerea CHECK

- Acest tip de constrângere se foloseşte pentru a forța valorile unei coloane să verifice o condiție. Condiția poate să conțină şi funcții, cu unele excepții(sysdate, user, unele functii de tip data calendaristica e.t.c).
- Sintaxa la nivel de coloană este:

  column [CONSTRAINT constraint_name] CHECK (expr)

- Sintaxa la nivel de tabel este:

  [, CONSTRAINT constraint_name] CHECK (expr)

Exemplu:

- Dacă vrem ca în tabelul angajati, numele angajaților să fie înregistrate numai cu litere mari, salariul să fie mai mare decât zero, iar comisionul să nu depăşească salariul, folosim constrângerea CHECK :

```
SQL>CREATE TABLE angajati
( id_ang      number(4)     PRIMARY KEY,
nume          varchar2(30)  CONSTRAINT CK_NU CHECK(nume=UPPER(nume),
functie       varchar2(20)  REFERENCES  functii(den_functie),
id_sef        number(4)     REFERENCES angajati(id_ang),
data_ang      date,
salariu       number(7,2)   CHECK (salariu>0),
comision      number(7,2),
id_dep        number(2)     REFERENCES departamente(id_dep),
CONSTRAINT CK_COM CHECK (comision <= salariu) );
```