4. Constrângeri de integritate

Cuprins

4. Constrângeri de integritate

- O constrângere este un mecanism care asigură că valorile unei coloane sau ale unei mulțimi de coloane satisfac o condiție declarată.
- Constrângerile sunt utilizate pentru a preveni existența datelor invalide în tabele.
- Tipuri de constrângeri declarative:
  - constrângerea de domeniu
    - definește valorile ce pot fi luate de un atribut
    - *CHECK*, *UNIQUE*, *NOT NULL*
  - constrângerea de integritate a entității
    - precizează cheia primară a unei tabele
    - valorile cheii primare trebuie să fie distincte și diferite de valoarea *null*
    - *PRIMARY KEY*
  - constrângerea de integritate referențială
    - asigură coerența dintre cheile externe și cheile primare / cheile unice referite
    - la adăugarea unei chei externe se verifică dacă:
      - a fost definită o cheie primară sau o cheie unică pentru tabela referită de cheia externă;
      - dacă numărul coloanelor ce compun cheia externă corespunde numărului de coloane ale cheii primare / cheii unice;
      - dacă tipul și lungimea fiecărei coloane a cheii externe corespunde cu tipul și lungimea fiecărei coloane a cheii primare / cheii unice.
    - *FOREIGN KEY*
- Unei constrângeri i se poate specifica un nume unic în cadrul schemei.
  - Dacă nu se specifică un nume explicit, atunci sistemul îi va atribui automat un nume de forma *SYS_n*, unde *n* reprezintă numărul constrângerii.
- Constrângerile pot fi definite pentru:
  - tabele (*table*);
  - vizualizări (*view*).

- Constrângerile pot fi:
  - adăugate

    ```
    ALTER TABLE/VIEW ... ADD CONSTRAINT ...
    ```
  - șterse

    ```
    ALTER TABLE/VIEW ... DROP CONSTRAINT ...
    ```
  - activate / dezactivate

    ```
    ALTER TABLE ... MODIFY CONSTRAINT ...
    ```
  - modificate la starea *RELY* sau *NORELY*

    ```
    ALTER TABLE ... MODIFY CONSTRAINT ...
    ```

- ❖ Pentru vizualizări se pot defini constrângeri:
  - de tipul *UNIQUE*, *PRIMARY KEY* și *FOREIGN KEY*;
  - doar cu opțiunile *DISABLE NOVALIDATE*;
  - cu opțiunile *RELY / NORELY*.
- ❖ Deoarece constrângerile vizualizărilor pot fi definite doar cu opțiunile *DISABLE NOVALIDATE*, acestea nu implică nicio operație de verificare a datelor și, prin urmare, niciun consum de resurse.
- ❖ Aceste constrângeri sunt utilizate de sistem în etapa de optimizare, pentru a ajuta optimizatorul să determine planuri de execuție mai bune.

- Vizualizări din dicționarul datelor ce oferă informații despre constrângerile definite în schema curentă:
  - *USER_CONSTRAINTS*
  - *USER_CONS_COLUMNS*

- ❖ Există posibilitatea ca aplicarea unei constrângeri să fie amânată (*DEFERRABLE*).
- ❖ În acest caz, pot fi executate mai multe comenzi *SQL* fără a se verifica restricția impusă de constrângere, aceasta fiind verificată numai la sfârșitul tranzacției, atunci când este executată comanda *COMMIT*.
- ❖ Dacă vreuna dintre comenzile tranzacției încalcă restricția, atunci întreaga tranzacție este derulată înapoi (*rollback*) și este întoarsă o eroare.
- ❖ Opțiunea implicită este *NOT DEFERRABLE*.

- ❖ Într-un depozit de date informațiile sunt adăugate sau modificate în timpul procesului *ETL*.
- ❖ În mod normal, utilizatorii nu actualizează în mod direct datele, așa cum se întâmplă în sistemele *OLTP*.
- ❖ Din acest motiv bazele de date depozit reprezintă un model special, pentru care au fost introduse mai multe concepte referitoare la constrângeri.

4.1. Opțiuni de definire a unei constrângeri

- O constrângere poate fi creată cu următoarele opțiuni:
  - *ENABLE* sau *DISABLE*
  - *VALIDATE* sau *NOVALIDATE*
  - *RELY* sau *NORELY*

Opțiunile *ENABLE* / *DISABLE*

- Pentru ca o constrângere să verifice dacă datele o satisfac aceasta trebuie să fie activă (*ENABLE*).
- O constrângere activă asigură că toate modificările datelor dintr-unul sau mai multe tabele satisfac condițiile constrângerii.
- Operațiile de modificare a datelor care încalcă o constrângere activă vor fi respinse și se va trimite un mesaj de eroare.
- O constrângere dezactivată (*DISABLE*) nu implică verificări în cazul niciunei operații.

- ❖ O constrângere activă nu presupune validarea datelor existente deja în tabele.
- ❖ O constrângere activă va determina verificări în cazul operațiilor care sunt lansate ulterior adăugării constrângerii.
- ❖ Implicit, constrângerile sunt definite cu opțiunea *ENABLE*.

Opțiunile *VALIDATE / NOVALIDATE*

- Dacă o constrângere este creată cu opțiunea *VALIDATE*, atunci toate datele curente din tabela vizată trebuie să satisfacă constrângerea respectivă.
- Dacă pică testul de validare, atunci constrângerea nu poate fi adăugată și este întors un mesaj de eroare.

- ❖ Validarea este independentă față de activare.
- ❖ Implicit, constrângerile sunt definite cu opțiunea *VALIDATE*.

Opțiunile *RELY / NORELY*

- În unele cazuri, utilizatorii știu că pentru o anumită constrângere restricțiile sunt îndeplinite, astfel că nu este necesară validarea datelor și activarea constrângerii. Totuși utilizatorul poate dori și în aceste cazuri să utilizeze constrângerea pentru a îmbunătăți optimizarea și performanța cererilor.
- Opțiunea *RELY* reprezintă un mecanism prin care se comunică *server*-ului că acea constrângere este satisfăcută (chiar dacă nu se realizează nicio operație de verificare).
- Implicit, constrângerile sunt definite cu opțiunea *NORELY*.

- ❖ O constrângere definită cu opțiunile *DISABLE VALIDATE* nu permite operații *LMD* asupra tabelei pentru care a fost definită.

CONSTRÂNGERE	OPERAȚII LMD PERMISE
ENABLE VALIDATE	DA
DISABLE VALIDATE	NU
ENABLE NOVALIDATE	DA
DISABLE NOVALIDATE	DA

- ❖ În această situație pot fi utilizate două strategii pentru modificarea datelor din tabelă:
  - folosirea comenzilor *LDD* pentru a adăuga date în tabelă (de exemplu *exchanging partitions*);
  - ștergerea (suprimarea) constrângerii înainte de modificarea tabelei;
    - după ce constrângerea a fost ștearsă se realizează modificările asupra datelor, iar la final se recreează constrângerea;
    - această abordare nu asigură că datele adăugate sau modificate respectă restricțiile; dacă restricțiile sunt încălcate, atunci operația de validare a datelor se termină fără succes, iar constrângerea nu poate fi adăugată.

- În mod normal constrângerile trebuie să fie definite cu opțiunile *ENABLE VALIDATE* pentru a asigura consistența bazei de date.
- Deoarece depozitele de date au dimensiuni mari, crearea și menținerea constrângerilor pot determina costuri considerabile. Din acest motiv, în cazul depozitelor de date există o serie de recomandări în ce privește opțiunile de definire a constrângerilor.

4.2. Constrângerea de unicitate în depozitele de date

- Constrângerea de unicitate este aplicată utilizând un index unic (de tip *B*Tree*). Acest index unic este definit automat de sistem, doar dacă acea constrângere este creată cu opțiunea *ENABLE*.
- Implicit, numele indexului unic astfel definit este același cu numele constrângerii.

- Dacă o constrângere de unicitate este definită cu opțiunea *DISABLE*, atunci indexul unic nu este creat.
- Dacă o constrângere de unicitate activă (*ENABLE*) este dezactivată (*DISABLE*), atunci implicit sistemul șterge indexul unic. Acesta va fi recreat de sistem la reactivarea constrângerii respective. Dacă se dorește păstrarea indexului unic atunci trebuie să se precizeze opțiunea *KEEP INDEX*. În acest mod se păstrează și statisticile asociate indexului respectiv.
- Opțiunea *KEEP INDEX* se poate preciza și atunci când constrângerea este ștearsă, efectele acesteia fiind aceleași.

- În mod asemănător constrângerii de unicitate și constrângerea de cheie primară (deoarece implică unicitatea valorilor cheii) este aplicată de sistem folosind un index unic.

Exemplul 4.1

Numele indexului poate fi diferit de numele constrângerii. Indexul definit de utilizator pentru a întreține constrângerea poate fi un index unic sau un index neunic.

```
CREATE TABLE clienti
(id_client NUMBER(4) CONSTRAINT pk_clienti PRIMARY KEY
                  USING INDEX (CREATE UNIQUE INDEX client_id_idx
                               ON clienti(id_client)),
```

```
 nume       VARCHAR2(20),
 prenume    VARCHAR2(30),
 email      VARCHAR2(50) CONSTRAINT u_clienti_email UNIQUE
                    USING INDEX (CREATE INDEX client_email_idx
                                 ON clienti(email))
);
```

Exemplul 4.2

La dezactivarea constrângerii se poate preciza dacă se va păstra indexul.

```
ALTER TABLE clienti
DISABLE CONSTRAINT pk_clienti KEEP INDEX;
```

Exemplul 4.3

La ștergerea constrângerii se poate preciza dacă se va păstra indexul.

```
ALTER TABLE clienti
DROP CONSTRAINT u_clienti_email KEEP INDEX;
```

❖ Deoarece depozitele de date au dimensiuni mari, crearea și menținerea constrângerilor pot determina costuri considerabile. Din acest motiv, în cazul depozitelor de date există o serie de recomandări în ce privește opțiunile de definire a constrângerilor.

Exemplul 4.4

Se dă tabela de fapte *vanzari* având următoarele atribute: *id#*, *id_produs, id_client, id_timp, nr_factura*, *cantitate*, *pret_unitar_vanzare*. Coloanele *id_produs, id_client, id_timp* și *nr_factura* identifică în mod unic o înregistrare din această tabelă.

a. Definirea constrângerii de unicitate folosind opțiunile implicite.

```
ALTER TABLE vanzari
ADD CONSTRAINT u_vanzari
UNIQUE(id_produs, id_client, id_timp, nr_factura);
```

Observații:

- Deoarece constrângerea a fost definită folosind opțiunile implicite rezultă că aceasta a fost de fapt definită cu următoarele opțiuni: *ENABLE*, *VALIDATE* și *NORELY*.
- Deoarece constrângerea a fost definită cu opțiunea implicită *ENABLE,* sistemul a creat automat un index unic (denumit *u_vanzari*) pentru a întreține această constrângere.

- Acest index unic poate fi problematic în cazul depozitelor de date pentru că:
  - poate fi foarte mare, deoarece tabela *vanzari* poate avea milioane sau poate chiar miliarde de linii;
  - este rar utilizat pentru procesarea cererilor, deoarece cele mai multe cereri nu utilizează predicate pe coloane unice; din acest motiv un index unic nu va mări performanțele;
  - dacă tabela *vanzari* este partiționată după o coloană diferită de colonele ce compun cheia unică, atunci indexul trebuie să fie global; aceasta va afecta toate operațiile de întreținere a tabelei *vanzari*.

b. Pentru depozitele de date, o alternativă de definire a constrângerii de unicitate este următoarea:

```
ALTER TABLE vanzari
ADD CONSTRAINT u_vanzari
UNIQUE(id_produs, id_client, id_timp, nr_factura)
DISABLE VALIDATE;
```

Observații:

- Deoarece constrângerea a fost definită cu opțiunea *DISABLE,* acesta nu a determinat sistemul să creeze indexul unic.
- Această abordare este avantajoasă în cazul depozitelor de date, deoarece constrângerea asigură unicitatea fără costurile unui index unic.
- Deoarece constrângerea nu este activă, dar s-a realizat validarea datelor (*DISABLE VALIDATE*), nu sunt premise operații *LMD* asupra tabelei *vanzari*. În această situație, trebuie aleasă o strategie de actualizare a tabelei prin procesul *ETL*.

4.3. Constrângerea de cheie externă în depozitele de date

- Într-o schemă stea, constrângerile de cheie externă validează relațiile dintre tabela de fapte și tabelele dimensiune.

Exemplul 4.5

Tabela de fapte *vanzari* conține atributul *id_client* care referă atributul cu același nume din tabela dimensiune *clienti*.

a. Definirea constrângerii de cheie externă folosind opțiunile implicite.

```
ALTER TABLE vanzari ADD CONSTRAINT fk_vanz_clienti
FOREIGN KEY (id_client)
REFERENCES clienti (id_client);
```

Observații:

- Deoarece constrângerea a fost definită folosind opțiunile implicite rezultă că aceasta a fost de fapt definită cu următoarele opțiuni: *ENABLE*, *VALIDATE* și *NORELY*.
- Deoarece constrângerea a fost definită cu opțiunea implicită *VALIDATE,* sistemul va verifica dacă datele existe în cele două tabele respectă restricțiile impuse de constrângere.
- În cazul depozitelor de date se poate întâmpla frecvent ca datele existe în tabele să nu treacă temporar pasul de validare.
  - Presupunem că în fiecare zi sunt încărcate date noi în tabela de fapte, dar actualizarea tabelelor dimensiune se realizează doar o dată pe săptămână.
  - În timpul săptămânii, tabelele dimensiune și tabela de fapte pot încălca constrângerea de cheie externă.

b. Pentru depozitele de date, o alternativă de definire a constrângerii de cheie externă este următoarea:

```
ALTER TABLE vanzari ADD CONSTRAINT fk_vanzari_clienti
FOREIGN KEY (id_client)
REFERENCES clienti (id_client)
ENABLE NOVALIDATE;
```

Observații:

- Deoarece constrângerea a fost definită cu opțiunea *NOVALIDATE,* acesta nu a determinat sistemul să verifice dacă datele existente în tabele sunt valide relativ la restricțiile impuse de constrângere.
- Constrângerea este definită cu opțiunea *ENABLE*, adică constrângerea este activă și va determina verificarea integrității datelor în cazul operațiilor *LDM* realizate asupra tabelelor implicate.
- Administratorul poate crea constrângerea cu opțiunile *ENABLE NOVALIDATE* atunci când:
  - tabelele conțin date care în acel moment nu satisfac constrângerea, dar se dorește crearea unei constrângeri care va fi ulterior utilizată;
  - constrângerea este cerută imediat, iar pasul de validare implică un consum mare de timp și resurse;

- deși datele sursă respectă constrângerea, iar importul de date asigura această constrângere prin procesul *ETL*, totuși se dorește menținerea constrângerii și la nivelul depozitului de date pentru a preveni orice modificări ce pot afecta constrângerea de cheie externă în afara procesului *ETL*; în această situație, constrângerea de cheie externă se va șterge înaintea de a se realiza importul de date și se va redefini după finalizarea procesului *ETL*.

▪ Dacă procesul *ETL* verifică îndeplinirea constrângerii de cheie externă, atunci este recomandat ca sistemul să nu realizeze încă o dată aceleași verificări, deoarece ar determina un consum inutil de timp și resurse.

4.4. Constrângeri *RELY* în depozitele de date

- În mod normal, procesul *ETL* verifică dacă anumite constrângeri sunt îndeplinite.
  - De exemplu, acesta poate valida toate cheile externe pentru datele care vor intra în tabela de fapte.
  - Administratorul poate avea încredere în procesul *ETL* privind verificarea constrângerilor, în loc să le implementeze la nivelul depozitului de date.
- Crearea unei constrângeri cu opțiunea *RELY* nu implică niciun consum de resurse, dar se poate dori precizarea acesteia din motive de optimizare.
- Constrângerile *RELY*:
  - determină rescrierea cererilor complexe folosind vizualizările materializate;
  - permit altor utilitare de *data warehouse* să regăsească informația în raport cu constrângerile, direct din dicționarul datelor.

Exemplul 4.6

Tabela de fapte *vanzari* conține atributul *id_timp* care referă atributul cu același nume din tabela dimensiune *timp*. Pentru depozitele de date, o alternativă de definire a constrângerii de cheie externă între cele două tabele este următoarea:

```
ALTER TABLE vanzari ADD CONSTRAINT fk_vanzari_timp
FOREIGN KEY (id_timp)
REFERENCES timp (id_timp)
RELY DISABLE NOVALIDATE;
```

Observații:

- Deoarece constrângerea a fost definită cu opțiunile *RELY DISABLE NOVALIDATE*, acesta nu implică validarea datelor existente în cele două tabele și nici verificări ulterioare în cazul comenzilor *LMD* executate pe acestea.
- Deoarece constrângerea a fost definită cu opțiunea *RELY* optimizatorul va putea utiliza constrângerea pentru a determina un plan optim.

Bibliografie

1. Connolly T.M., Begg C.E., *Database Systems*: *A Practical Approach to Design, Implementation and Management*, 5th edition, Pearson Education, 2005
2. Dollinger R., Andron L., *Baze de date și gestiunea tranzacțiilor*, Editura Albastră, Cluj-Napoca, 2004
3. Inmon W.H., *Building the Data Warehouse*, 4th Edition, Wiley, 2005
4. Kimball R., *The Data Warehouse Toolkit*, 3rd Edition, Wiley, 2013
5. Kimball R., Ross M., Thornthwaite W., Mundy J., Becker B., *The Data Warehouse Lifecycle Toolkit*, 2nd Edition Wiley, 2008
6. Oracle and/or its affiliates, *Oracle Database Concepts*, 1993, 2024
7. Oracle and/or its affiliates, *Oracle Database Administrator's Guide*, 2001, 2025
8. Oracle and/or its affiliates, *Oracle Database Performance Tuning Guide*, 2013, 2025
9. Oracle and/or its affiliates, *Oracle Database SQL Language Reference*, 1996, 2025
10. Oracle and/or its affiliates, *Oracle Database PL/SQL Language Reference*, 1996, 2025
11. Oracle and/or its affiliates, *Oracle Database: SQL Tuning Workshop*, 2010, 2025
12. Oracle and/or its affiliates, *Oracle OLAP Customizing Analytic Workspace Manager*, 2006, 2019
13. Oracle and/or its affiliates, *Oracle OLAP DML Reference*, 1994, 2019
14. Oracle and/or its affiliates, *Oracle OLAP User's Guide*, 2003, 2019
15. Oracle and/or its affiliates, *Oracle Warehouse Builder Concepts*, 2000, 2021
16. Oracle and/or its affiliates, *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide*, 2000, 2021
17. Oracle and/or its affiliates, *Oracle Database Data Warehousing Guide*, 2001, 2021
18. Oracle University, *Oracle Database*: *PL/SQL Fundamentals*, *Student Guide*, 2009, 2025
19. Poe V., Klauer P., Brobst S., *Building A Data Warehouse for Decision Support*, 2nd Edition, Prentice Hall; 1997
20. Popescu I., Alecu A., Velcescu L., Florea (Mihai) G., *Programare avansată în Oracle9i*, Ed. Tehnică, 2004