BAZE DE DATE

CURS 12

SQL

Structured Query Language

- limbaj universal care poate fi utilizat pentru a **defini**, **interoga**, **reactualiza** şi **gestiona** baze de date relaţionale.
    - LDD + LMD + LCD + alte comenzi (de administrare etc.)
- SQL este un limbaj **neprocedural**, adică se specifică **CE** informaţie este solicitată, dar nu modul **CUM** se obţine această informaţie.
- SQL poate fi utilizat autonom sau prin inserarea comenzilor sale într-un limbaj de programare.

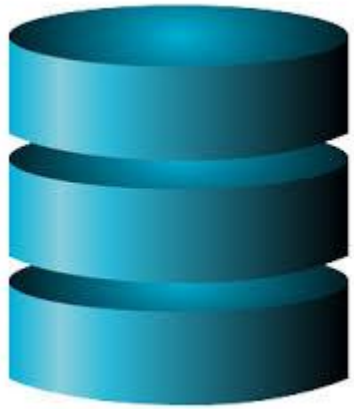
SQL

3 familii de comenzi:

- Comenzi pentru definirea datelor (LDD) - permit **descrierea** (definirea) **structurii obiectelor** ce modelează sistemul studiat.
- Comenzi pentru prelucrarea datelor (LMD) - ce permit **consultarea, reactualizarea, suprimarea sau inserarea** datelor.
- Comenzi pentru controlul datelor (LCD) - permit asigurarea **confidenţialităţii şi integrităţii** datelor, **salvarea** informaţiei, realizarea fizică a **modificărilor** în baza de date, rezolvarea unor probleme de **concurenţă**.

LDD

- O BD este alcătuită din scheme.
- O schemă este o mulţime de structuri logice de date, numite obiecte. Ea aparţine unui utilizator al bazei de date şi poartă numele său.
- Specificarea bazelor de date şi a obiectelor care le compun se realizează prin intermediul limbajului de definire a datelor (LDD).
- Definirea unui obiect presupune crearea, modificarea şi suprimarea sa. Limbajul de definire a datelor cuprinde instrucţiunile SQL care permit realizarea acestor operaţii (CREATE, ALTER, DROP).
- Instrucţiunile LDD au efect imediat asupra bazei de date şi înregistrează informaţia în dicţionarul datelor.
- LDD contine și instructiunile RENAME, TRUNCATE.



LDD

Tipuri de date

- Pentru memorarea datelor numerice, tipurile cele mai frecvent folosite sunt: **NUMBER**, **INTEGER**, **FLOAT**, **DECIMAL**.
- Pentru memorarea şirurilor de caractere, cele mai frecvent tipuri de date utilizate sunt: **CHAR**, **VARCHAR2** şi **LONG (**inlocuit de **CLOB)**.
- Informaţii relative la timp sau dată calendaristică se obţin utilizând tipul **DATE**.
  - anul, luna, ziua, ora, minutul, secunda
  - pentru o coloană de tip DATE sistemul rezervă 7 bytes

Modele de format

- Un model de format este un literal caracter care descrie formatul valorilor de tip DATE sau NUMBER stocate într-un şir de caractere.
- Atunci când se converteşte un şir de caractere într-o dată calendaristică sau într-un număr, modelul de format indică sistemului cum să interpreteze şirul respectiv.
- În instrucţiunile SQL se poate folosi un model de format ca argument al funcţiilor **TO_CHAR** şi **TO_DATE**.

LDD

Valoarea NULL

- Valoarea *null*, reprezentând lipsa datelor, nu este egală cu nicio altă valoare sau nu este diferită de nicio altă valoare, inclusiv *null*;

- Sistemul Oracle consideră două valori *null* ca fiind egale la evaluarea funcţiei DECODE

LDD

Pseudocoloane

- O pseudocoloană se comportă ca o coloană a unui tabel, dar nu este stocată efectiv într-un tabel.
- Se pot face interogări asupra pseudocoloanelor, dar nu se pot insera, actualiza sau şterge valorile acestora.
- LEVEL returnează nivelul liniilor rezultat ale unei cereri ierarhice.
- CURRVAL şi NEXTVAL sunt pseudocoloane utile în lucrul cu secvenţe şi sunt tratate în secţiunea corespunzătoare acestora.
- ROWNUM returnează numărul de ordine al liniilor rezultate în urma execuţiei unei cereri. Pseudocoloana poate fi utilizată pentru a limita numărul de linii returnate.

Tabele

Crearea unui tabel

- constă din generarea **structurii** sale, adică atribuirea unui nume tabelului şi definirea caracteristicilor sale (se definesc coloanele, se definesc constrângerile de integritate, se specifică parametrii de stocare, etc).

- Comanda CREATE TABLE permite crearea unui **tabel relaţional** sau a unui tabel obiect. Tabelul relaţional reprezintă **structura fundamentală pentru stocarea datelor utilizatorului**.

Tabele

- Pentru a crea un tabel, utilizatorul trebuie să aibă acest **privilegiu** şi să dispună de **spaţiul de memorie** în care să creeze obiectul.
- La nivelul schemei sale, un utilizator are toate privilegiile.

CREATE TABLE [<nume_schema>.] <nume_tabel> (
<nume_coloana_1> <tip_date> [DEFAULT <expresie>], …
<nume_coloana_n> <tip_date> [DEFAULT <expresie>])
[CLUSTER <nume_cluster> (<coloana_1>,…, <coloana_m>)][ENABLE | DISABLE <clause>];

- Comanda poate conţine opţional clauza **TABLESPACE**, care specifică spaţiul tabel în care va fi stocat tabelul.
- De asemenea, poate conţine opţional clauza **STORAGE** care este folosită pentru setarea parametrilor de stocare prin intermediul cărora se specifică mărimea şi modul de alocare a extinderilor segmentului tabel.

Constrângeri

- Constrângerea este un mecanism care asigură că valorile unei coloane sau ale unei mulţimi de coloane satisfac o **condiţie declarată**.
- Unei constrâgeri i se poate da un **nume unic** în cadrul schemei.
  - Dacă nu se specifică un nume explicit atunci sistemul automat îi atribuie un nume de forma **SYS_C<n>**, unde **n** reprezintă numărul constrângerii.
- Constrângerile pot fi **şterse**, pot fi **adăugate**, pot fi **activate sau dezactivate**, dar *nu pot fi modificate*.

Constrângeri

- **Constrângeri declarative**: constrângeri de domeniu, constrângerea de integritate a entităţii, constrângerea de integritate referenţială.
- **Constrângerile de domeniu** definesc valori luate de un atribut (DEFAULT, CHECK, UNIQUE, NOT NULL).
- **Constrângerea de integritate a entităţii** precizează cheia primară a unui tabel.
  - Când se creează cheia primară se generează automat un **index unic**.
  - **Valorile cheii primare sunt distincte şi diferite de valoarea *null*.**
- **Constrângerea de integritate referenţială** asigură coerenţa între cheile primare şi cheile externe corespunzătoare.
  - se verifică dacă a fost definită o cheie primară pentru tabelul referit de cheia externă;
  - dacă numărul coloanelor ce compun cheia externă corespunde numărului de coloane ale cheii primare;
  - dacă tipul şi lungimea fiecărei coloane a cheii externe corespunde cu tipul şi lungimea fiecărei coloane a cheii primare.

Constrângeri

- Definiţiile şi numele constrângerilor definite se pot fi consulta prin interogarea vizualizărilor **USER_CONSTRAINTS** şi **ALL_CONSTRAINTS** din dicţionarul datelor.
- Există posibilitatea ca o constrângere să fie amânată (**DEFERRABLE**).
  - În acest caz, mai multe comenzi SQL pot fi executate fără a se verifica restricţia, aceasta fiind verificată numai la sfârşitul tranzacţiei, atunci când este executată comanda COMMIT.
  - Dacă vreuna din comenzile tranzacţiei încalcă restricţia, atunci întreaga tranzacţie este derulată înapoi şi este returnată o eroare. Opţiunea implicită este **NOT DEFERRABLE**.

Modificarea structurii unui tabel

- Comanda care realizează modificarea structurii tabelului (la nivel de coloană sau la nivel de tabel), dar **nu modificarea conţinutului acestuia**, este **ALTER TABLE**.

- Comanda **nu schimbă conţinutul tabelului**.

Modificarea structurii unui tabel

Comanda **ALTER TABLE** permite:

- adăugarea (**ADD**) de coloane, chei (primare sau externe), constrângeri într-un tabel existent;
- modificarea (**MODIFY**) coloanelor unui tabel, inclusiv specificarea unei valori implicite pentru o coloană existentă;
- suprimarea unei coloane (**DROP COLUMN**);
- adăugarea de constrângeri (**ADD CONSTRAINT**);
- suprimarea (**DROP CONSTRAINT**) cheii primare, a cheii externe sau a altor constrângeri;
- activarea şi dezactivarea (**ENABLE**, **DISABLE**) unor constrângeri;

Modificarea structurii unui tabel

Comanda ALTER TABLE are următoarea sintaxă simplificată:

ALTER TABLE [<nume_schema>.] <nume_tabel>
[ADD(<nume_coloana> <tip_date>, <constrângere>)
| MODIFY (<nume_coloana_1>,..., <nume_coloana_n>)
| DROP <clauza_drop>,] [ENABLE | DISABLE
<clause>];

Suprimarea unui tabel

- Pentru ştergerea unui tabel este utilizată comanda DROP TABLE:

```
DROP TABLE [nume_schema.]nume_tabel [CASCADE CONSTRAINTS];
```

- Clauza ***CASCADE CONSTRAINTS*** permite suprimarea tuturor constrângerilor de integritate referenţială corespunzătoare cheilor primare şi unice din tabelul supus ştergerii.
  - Altfel, sistemul returnează o eroare şi nu suprimă tabelul.
- Suprimarea unui tabel presupune:
  - suprimarea definiţiei sale în dicţionarul datelor;
  - suprimarea indecşilor asociaţi;
  - suprimarea privilegiilor conferite în legătură cu tabelul;
  - recuperarea spaţiului ocupat de tabel;
  - permanentizarea tranzactiilor in asteptare;
  - **invalidarea** (dar nu suprimarea) funcţiilor, procedurilor, vizualizărilor, secventelor, sinonimelor referitoare la tabel.

Suprimarea unui tabel

- Odată executată, instrucţiunea DROP TABLE este **ireversibilă**.
- Ca şi în cazul celorlalte instrucţiuni ale limbajului de definire a datelor, această comandă nu poate fi anulată (*ROLLBACK*).
- Oracle 10g a introdus o noua manieră pentru suprimarea unui tabel.
  - Când se șterge un tabel, baza de date nu eliberează imediat spațiul asociat tabelului.
  - Ea redenumește tabelul și acesta este plasat într-un *recycle bin* de unde poate fi eventual recuperat ulterior prin comanda **FLASHBACK TABLE**.

Suprimarea unui tabel

Exemplu:

```
DROP TABLE exemplu;
SELECT …
INSERT…
FLASHBACK TABLE exemplu TO BEFORE DROP;
```

- Ștergerea unui tabel se poate face simultan cu eliberarea spațiului asociat tabelului, dacă este utilizată clauza **PURGE** în comanda **DROP TABLE**.
  - Nu este posibil un *rollback* pe o comanda DROP TABLE cu clauza PURGE.
- Pentru ştergerea întregului conţinut al unui tabel şi eliberarea spaţiului de memorie ocupat de acesta, sistemul Oracle oferă instrucţiunea:

```
TRUNCATE TABLE nume_tabel;
```

Suprimarea unui tabel

TRUNCATE sau DELETE?

- Fiind o instrucţiune LDD, TRUNCATE nu poate fi anulată ulterior (prin ROLLBACK).
- Ea reprezintă o alternativă a comenzii DELETE din limbajul de prelucrare a datelor.
- DELETE nu eliberează spaţiul de memorie.
- TRUNCATE este mai rapidă deoarece nu generează informaţie ROLLBACK şi nu activează declanşatorii asociaţi operaţiei de ştergere.
  - Dacă tabelul este „părintele“ unei constrângeri de integritate referenţială, el nu poate fi trunchiat.
  - Pentru a putea fi aplicată instrucţiunea TRUNCATE, constrângerea trebuie să fie mai întâi dezactivată.

Informații în DD

- În DD, informaţiile despre tabele se găsesc în vizualizarea USER_TABLES. Dintre cele mai importante coloane ale acesteia, se remarcă:

TABLE_NAME	Numele tabelului
TABLESPACE_NAME	Spaţiul tabel în care se află tabelul
CLUSTER_NAME	Numele cluster-ului din care face parte tabelul
PCT_FREE	Procentul de spaţiu păstrat liber în interiorul fiecărui bloc
PCT_USED	Procentul de spaţiu ce poate fi utilizat în fiecare bloc
INI_TRANS	Numărul iniţial de tranzacţii concurente în interiorul unui bloc
NUM_ROWS	Numărul de înregistrări din tabel
BLOCKS	Numărul de blocuri utilizate de tabel
EMPTY_BLOCKS	Numărul de blocuri ce nu conţin date
TEMPORARY	Y sau N, indică dacă tabelul este temporar (sau nu)

Indecşi

- Un index este un obiect al schemei unei baze de date care **creşte viteza de execuţie a cererilor**
- Server-ul Oracle utilizează identificatorul ROWID pentru regăsirea liniilor în structura fizică a bazei de date
- Indexul, din punct de vedere logic, este compus **dintr-o valoare cheie şi din identificatorul adresă ROWID**
- Indecşii sunt utilizaţi şi întreţinuţi automat de către server-ul Oracle. Odată creat indexul, el nu necesită o acţiune directă din partea utilizatorului

Indecşi

- Indecşii pot fi creaţi în două moduri:
  - automat, de server-ul Oracle (**PRIMARY KEY, UNIQUE KEY**);
  - manual, de către utilizator (CREATE INDEX, CREATE TABLE).
- Server-ul Oracle creează automat un **index unic** atunci când se defineşte o constrângere PRIMARY KEY sau UNIQUE asupra unei coloane sau unui grup de coloane.
- Numele indexului va fi acelaşi cu numele constrângerii.
- Indecşii, fiind obiecte ale schemei bazei, beneficiază de procesul de definire a unui obiect.

Indecşi

Crearea unui index (care nu este obligatoriu unic) pe una sau mai multe coloane ale unui tabel se face prin comanda:

CREATE [UNIQUE] INDEX <nume_index> ON [<nume_schema>.] <nume_tabel> (<nume_col> [ASC | DESC], <nume_col> [ASC | DESC], …) | CLUSTER <nume_cluster>];

Indecşi

Mai mulţi indecşi asupra unui tabel nu implică întotdeuna interogări mai rapide.

Fiecare operaţie LMD care este permanentizată asupra unui tabel cu indecşi asociaţi presupune actualizarea indecşilor respectivi

Prin urmare, **este recomandată** crearea de indecşi numai în anumite situaţii:

- coloana conţine o varietate mare de valori;
- coloana conţine un număr mare de valori *null*;
- una sau mai multe coloane sunt utilizate frecvent împreună într-o clauză WHERE sau într-o condiţie de JOIN;
- tabelul este mare şi este de aşteptat ca majoritatea interogărilor asupra acestuia să regăsească mai puţin de 2-4% din linii;

Indecşi

Crearea unui index **nu este recomandată** în următoarele cazuri:

- tabelul este mic;
- coloanele nu sunt utilizate des în cadrul unei condiţii dintr-o cerere;
- majoritatea interogărilor regăsesc mai mult de 2-4% din liniile tabelului;
- tabelul **este actualizat frecvent**;
- coloanele indexate sunt referite în expresii;

Indecşi

In concluzie, ce tabele sau ce coloane trebuie (sau nu) indexate?

- indexaţi tabelele pentru care interogările selectează **un număr redus de rânduri** (sub 5%);
- indexaţi tabelele care sunt interogate folosind **clauze SQL simple**;
- nu indexaţi tabelele ce conţin puţine înregistrări (accesul secvenţial este mai simplu);
- nu indexaţi tabelele care sunt frecvent actualizate, deoarece **ştergerile, inserările şi modificările sunt îngreunate de indecşi**;
- indexaţi coloanele folosite frecvent în clauza **WHERE** sau în clauza **ORDER BY**;
- nu indexaţi coloanele ce conţin date asemănătoare (puţine valori distincte);

Indecşi

Exemplu:

```
CREATE INDEX upper_nume_idx ON artist (UPPER(nume));
```

- Indexul creat prin instrucţiunea precedentă facilitează prelucrarea unor interogări precum:

```
SELECT * FROM artist
WHERE UPPER(nume) = 'GRIGORESCU';
```

- Pentru a asigura că server-ul Oracle **utilizează indexul** şi nu efectuează o căutare asupra întregului tabel, valoarea funcţiei corespunzătoare expresiei indexate **trebuie să nu fie null** în interogările ulterioare creării indexului. Următoarea instrucţiune garantează utilizarea indexului dar, în absenţa clauzei WHERE, serverul Oracle ar putea cerceta întreg tabelul.

```
SELECT   *  FROM     artist
WHERE    UPPER(nume) IS NOT NULL
ORDER BY UPPER(nume);
```

Indecşi

Exemplu:
Să se creeze tabelul *artist*, specificându-se indexul asociat cheii primare.

```
CREATE TABLE artist (cod_artist NUMBER PRIMARY KEY
                                    USING INDEX
                                (CREATE INDEX artist_cod_idx
                                 ON artist(cod_artist)
                                ),
                     nume VARCHAR2(30) NOT NULL,
                     prenume VARCHAR2(30),
                    …);
```

- **Ştergerea** unui index se face prin comanda:

```
DROP INDEX nume_index [ON [nume_schema.] nume_tabel]
```

  - Pentru a suprima indexul trebuie ca acesta să se găsească în schema personală sau să avem privilegiul de sistem DROP ANY INDEX.

- Pentru a **reconstrui un index** se pot folosi două metode:
  - se şterge indexul (DROP INDEX) şi se recreează (CREATE INDEX);
  - se utilizează comanda ALTER INDEX cu opţiunea REBUILD.

Secvenţe

- O secvenţă este un obiect în baza de date care serveşte pentru a genera **întregi unici** în sistemele multi-utilizator, evitând apariţia conflictelor şi a blocării.
- Secvenţele sunt memorate şi generate **independent de tabele**; aceeaşi secvenţă poate fi utilizată pentru mai multe tabele.
- O secvenţă poate fi creată de un utilizator şi poate fi partajată de mai mulţi utilizatori.
- Crearea unei secvenţe se face cu ajutorul comenzii:

```
CREATE SEQUENCE [<nume_schema>.]<nume_secventa>
[INCREMENT BY n]  [START WITH m]   [{MAXVALUE n |
NOMAXVALUE}]  [{MINVALUE n | NOMINVALUE}]   [{CACHE k
| NOCACHE}]
```

  - CACHE k | NOCACHE specifică numărul de valori alocate de server-ul Oracle pe care le va păstra în memoria *cache* pentru a oferi utilizatorilor un acces rapid (implicit sunt alocate 20 de valori);

Secvenţe

- O secvenţă este referită într-o comandă SQL cu ajutorul pseudo-coloanelor:
  - NEXTVAL – referă valoarea următoare a secvenţei;
  - CURRVAL – referă valoarea curentă a secvenţei.

- NEXTVAL şi CURRVAL pot fi folosite în:
  - clauza VALUES a unei comenzi INSERT;
  - clauza SET a unei comenzi UPDATE;
  - lista SELECT a unei subcereri dintr-o comanda INSERT;
  - lista unei comenzi SELECT.

Secvenţe

NEXTVAL şi **CURRVAL** **nu pot fi folosite** în:

- subinterogare în SELECT, DELETE sau UPDATE;
- interogarea unei vizualizări;
- comandă SELECT cu operatorul DISTINCT;
- comandă SELECT cu clauza GROUP BY, HAVING sau ORDER BY;
- clauza WHERE a unei comenzi SELECT;
- condiţia unei constrângeri CHECK;
- valoarea DEFAULT a unei coloane într-o comandă CREATE TABLE sau ALTER TABLE;
- comandă SELECT care este combinată cu altă comandă SELECT printr-un operator pe mulţimi (UNION, INTERSECT, MINUS).

Secvenţe

Din dicţionarul datelor pot fi obţinute informaţii despre secvenţe folosind vizualizarea USER_SEQUENCES.

Exemplu:

- Să se creeze o secvenţă *domeniuseq* care să fie utilizată pentru a insera noi domenii în tabelul domeniu şi să se insereze un nou domeniu.
- Să se afişeze informaţiile referitoare la secvenţa *domeniuseq*.

```
CREATE SEQUENCE domeniuseq
START WITH 1
INCREMENT BY 1;

INSERT INTO domeniu  VALUES
(domeniuseq.NEXTVAL,'Informatica');

SELECT INCREMENT, START, MAXVALUE, MINVALUE
FROM   USER_SEQUENCES
WHERE  SEQUENCE_NAME = 'domeniuseq';
```

Vizualizări

- Vizualizarea (view) este un **tabel logic (virtual)** relativ la date din una sau mai multe tabele sau vizualizări.
- Vizualizarea este definită plecând de la o **cerere** a limbajului de interogare a datelor, moştenind caracteristicile obiectelor la care se referă.
- Vizualizarea, fiind virtuală, **nu solicită o alocare de memorie** pentru date.
- Textul cererii (SELECT) care defineşte vizualizarea este salvat în DD.
- Oracle transformă cererea referitoare la o vizualizare într-o cerere relativă la tabelele de bază.
  - Nucleul Oracle determină **fuzionarea** cererii relative la vizualizare cu comanda de definire a vizualizării, analizează rezultatul fuziunii în zona partajată şi execută cererea.
  - Dacă sunt utilizate clauzele UNION, GROUP BY şi CONNECT BY, atunci Oracle nu determină fuzionarea, el va rezolva vizualizarea şi apoi va aplica cererea rezultatului obţinut.

Vizualizări

- O vizualizare reflectă la orice moment **conţinutul** exact al tabelelor de bază.
- Orice modificare efectuată asupra tabelelor se realizeaza instantaneu si asupra vizualizării.
- Ştergerea unui tabel implică **invalidarea vizualizărilor asociate** tabelului şi nu ştergerea acestora.

- Vizualizările sunt definite pentru:
  - furnizarea unui nivel mai înalt de **securizare a bazei** (restricţionarea acesului la date);
  - simplificarea formulării unei cereri;
  - mascarea complexităţii datelor;
  - afişarea datelor într-o altă reprezentare decât cea a tabelelor de bază;
  - asigurarea confidenţialităţii anumitor informaţii;

Vizualizări

- Vizualizările pot fi **simple** sau **complexe**
- O vizualizare **simplă**:
  - extrage date dintr-un singur tabel;
  - nu conţine funcţii sau grupări de date;
  - asupra ei pot fi efectuate operaţii LMD;
- O vizualizare este considerată **complexă** dacă:
  - extrage date din mai multe tabele;
  - conţine funcţii sau grupări de date;
  - nu permite întotdeauna (prin intermediul său) operaţii LMD asupra tabelelor de bază;

Vizualizări

- Operaţiile LMD asupra vizualizărilor complexe sunt restricţionate de următoarele **reguli**:
  - nu se poate insera, actualiza sau şterge o linie dintr-o vizualizare dacă aceasta conţine **funcţii grup, clauza GROUP BY, cuvântul cheie DISTINCT sau pseudocoloana ROWNUM**;
  - nu se poate adăuga sau modifica o linie dintr-o vizualizare, dacă aceasta conţine **coloane definite prin expresii**;
- Nu pot fi adăugate linii printr-o vizualizare, dacă tabelul de bază conţine coloane care au **constrângerea NOT NULL şi nu apar în lista SELECT a vizualizării**

Vizualizări

Crearea unei vizualizări se realizează cu ajutorul comenzii:

```
CREATE [OR REPLACE][FORCE | NOFORCE] VIEW
[<nume_schema>.]<nume_view> [(<alias>[,<alias>]…)]
AS <cerere_SELECT>
[WITH {CHECK OPTION [CONSTRAINT <nume_constrangere>]
|READ ONLY }];
```

– **OR REPLACE** recreează vizualizarea dacă aceasta deja există.
– **FORCE** creează vizualizarea chiar dacă tabelul de bază nu există sau chiar dacă vizualizarea face referinţă la obiecte care încă nu sunt create. Deşi vizualizarea va fi creată, utilizatorul nu poate să o folosească.
– **NO FORCE** este implicită şi se referă la faptul că vizualizarea este creată numai dacă tabelele de bază există.
– Cererea este o comandă SELECT care poate să conţină alias pentru coloane.
– **WITH CHECK OPTION** specifică faptul că reactualizarea datelor din tabele (inserare sau modificare) se poate face numai asupra datelor selectate de vizualizare (care apar în clauza WHERE).
– **WITH READ ONLY** asigură că nicio operaţie LMD nu poate fi executată asupra vizualizării.

Vizualizări

Reactualizarea tabelelor implică reactualizarea corespunzătoare a vizualizărilor. Reactualizarea vizualizărilor implică reactualizarea tabelelor de bază? NU! Există restricţii care trebuie respectate.

- Nu pot fi modificate date din vizualizare sau adaugate date prin vizualizare, dacă aceasta conține coloane definite prin expresii.

- Nu pot fi inserate, şterse sau actualizate date din vizualizări ce conţin:
  - operatorul DISTINCT;
  - clauzele GROUP BY, HAVING, START WITH, CONNECT BY;
  - pseudo-coloana ROWNUM;
  - funcţii grup;
  - operatori pe mulţimi.

- Nu pot fi inserate sau actualizate date care ar încălca constrângerile din tabelele de bază.

Vizualizări

- Nu pot fi inserate sau actualizate valorile coloanelor care rezultă prin calcul
- Nu se pot face operaţii LMD asupra coloanelor calculate cu DECODE

Alături de restricţiile prezentate anterior, aplicabile tuturor vizualizărilor, există restricţii specifice, aplicabile vizualizărilor bazate pe mai multe tabele.

- Regula fundamentală este că orice operaţie INSERT, UPDATE sau DELETE pe o vizualizare bazată pe mai multe tabele poate modifica datele doar din unul din tabelele de bază.
- Un tabel de bază al unei vizualizări este protejat prin cheie (key preserved table) dacă orice cheie selectată a tabelului este de asemenea şi cheie a vizualizării.
  - Deci, un tabel protejat prin cheie este un tabel ale cărui chei se păstrează şi la nivel de vizualizare.

Vizualizări

- Asupra unui *join view* pot fi aplicate instrucţiunile INSERT, UPDATE sau DELETE, doar dacă sunt îndeplinite următoarele condiţii:
  - instrucţiunea LMD afectează numai unul dintre tabelele de bază;
  - în cazul instrucţiunii **UPDATE**, toate coloanele care pot fi reactualizate trebuie să corespundă coloanelor dintr-un tabel protejat prin cheie (în caz contrar, Oracle nu va putea identifica unic înregistrarea care trebuie reactualizată);
  - în cazul instrucţiunii **DELETE**, rândurile unei vizualizări pot fi şterse numai dacă există un tabel în join protejat prin cheie şi numai unul (în caz contrar, Oracle nu ar şti din care tabel să şteargă);
  - în cazul instrucţiunii **INSERT**, toate coloanele în care sunt inserate valori trebuie să provină dintr-un tabel protejat prin cheie.