



BAZE DE DATE

CURS 10

SQL

Structured Query Language

- limbaj universal care poate fi utilizat pentru a **defini**, **interoga**, **reactualiza** și **gestiona** baze de date relaționale.
 - LDD + LMD + LCD + alte comenzi (de administrare etc.)
- SQL este un limbaj **neprocedural**, adică se specifică **CE** informație este solicitată, dar nu modul **CUM** se obține această informație.
- 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 **confidențialității și integrității** datelor, **salvarea** informației, realizarea fizică a **modificărilor** în baza de date, rezolvarea unor probleme de **concurență**.

LDD

- O BD este alcătuită din scheme.
- O schemă este o mulțime de structuri logice de date, numite obiecte. Ea aparține unui utilizator al bazei de date și poartă numele său.
- Specificarea bazelor de date și a obiectelor care le compun se realizează prin intermediul limbajului de definire a datelor (LDD).
- Definirea unui obiect presupune crearea, modificarea și suprimarea sa. Limbajul de definire a datelor cuprinde instrucțiunile SQL care permit realizarea acestor operații (CREATE, ALTER, DROP).
- Instrucțiunile LDD au efect imediat asupra bazei de date și înregistrează informația în dicționarul datelor.
- LDD conține și instrucțiunile RENAME, TRUNCATE.



LDD

Tipuri de date

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

LDD

Modele de format

- Un model de format este un literal caracter care descrie formatul valorilor de tip DATE sau NUMBER stocate într-un șir de caractere.
- Atunci când se convertește un șir de caractere într-o dată calendaristică sau într-un număr, modelul de format indică sistemului cum să interpreteze șirul respectiv.
- În instrucțiunile SQL se poate folosi un model de format ca argument al funcțiilor **TO_CHAR** și **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 funcției 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 șterge valorile acestora.
- LEVEL returnează nivelul liniilor rezultat ale unei cereri ierarhice.
- CURRVAL și NEXTVAL sunt pseudocoloane utile în lucrul cu secvențe și sunt tratate în secțiunea corespunzătoare acestora.
- ROWNUM returnează numărul de ordine al liniilor rezultate în urma execuției 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 și definirea caracteristicilor sale (se definesc coloanele, se definesc constrângerile de integritate, se specifică parametrii de stocare, etc).
- Comanda CREATE TABLE permite crearea unui **tabel relațional** sau a unui tabel obiect. Tabelul relațional reprezintă **structura fundamentală pentru stocarea datelor utilizatorului**.

Tabele

- Pentru a crea un tabel, utilizatorul trebuie să aibă acest **privilegiu** și să dispună de **spațiul 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 conține opțional clauza **TABLESPACE**, care specifică spațiul tabel în care va fi stocat tabelul.
- De asemenea, poate conține opțional clauza **STORAGE** care este folosită pentru setarea parametrilor de stocare prin intermediul cărora se specifică mărimea și modul de alocare a extinderilor segmentului tabel.

Constrângeri

- Constrângerea este un mecanism care asigură că valorile unei coloane sau ale unei mulțimi de coloane satisfac o **condiție declarată**.
- Unei constrângeri 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 **șterse**, 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

- Definițiile și numele constrângerilor definite se pot fi consulta prin interogarea vizualizărilor **USER_CONSTRAINTS** și **ALL_CONSTRAINTS** din dicționarul datelor.
- Există posibilitatea ca o constrângere să fie amânată (**DEFERRABLE**).
 - În acest caz, mai multe comenzi SQL pot fi executate fără a se verifica restricția, aceasta fiind verificată numai la sfârșitul tranzacției, atunci când este executată comanda COMMIT.
 - Dacă vreuna din comenzile tranzacției încalcă restricția, atunci întreaga tranzacție este derulată înapoi și este returnată o eroare. Opțiunea 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 conținutului acestuia**, este **ALTER TABLE**.
- Comanda **nu schimbă conținutul 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 și 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 ștergerea unui tabel este utilizată comanda DROP TABLE:
DROP TABLE [nume_schema.]nume_tabel [CASCADE CONSTRAINTS] ;
- Clauza **CASCADE CONSTRAINTS** permite suprimarea tuturor constrângerilor de integritate referențială corespunzătoare cheilor primare și unice din tabelul supus ștergerii.
 - Altfel, sistemul returnează o eroare și nu suprimă tabelul.
- Suprimarea unui tabel presupune:
 - suprimarea definiției sale în dicționarul datelor;
 - suprimarea indecșilor asociați;
 - suprimarea privilegiilor conferite în legătură cu tabelul;
 - recuperarea spațiului ocupat de tabel;
 - permanentizarea tranzacțiilor în așteptare;
 - **invalidarea** (dar nu suprimarea) funcțiilor, procedurilor, vizualizărilor, secvențelor, sinonimelor referitoare la tabel.

Suprimarea unui tabel

- Odată executată, instrucțiunea DROP TABLE este **ireversibilă**.
- Ca și în cazul celorlalte instrucțiuni ale limbajului de definire a datelor, această comandă nu poate fi anulată (*ROLLBACK*).
- Oracle 10g a introdus o nouă manieră pentru suprimarea unui tabel.
 - Când se șterge un tabel, baza de date nu eliberează imediat spațiul asociat tabelului.
 - Ea redenumeste 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 ștergerea întregului conținut al unui tabel și eliberarea spațiului de memorie ocupat de acesta, sistemul Oracle oferă instrucțiunea:

```
TRUNCATE TABLE nume_tabel;
```

Suprimarea unui tabel

TRUNCATE sau DELETE?

- Fiind o instrucțiune 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 spațiul de memorie.
- TRUNCATE este mai rapidă deoarece nu generează informație ROLLBACK și nu activează declanșatorii asociați operației de ștergere.
 - Dacă tabelul este „părintele” unei constrângeri de integritate referențială, el nu poate fi trunchiat.
 - Pentru a putea fi aplicată instrucțiunea TRUNCATE, constrângerea trebuie să fie mai întâi dezactivată.

Informații în DD

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

TABLE_NAME	Numele tabelului
TABLESPACE_NAME	Spațiul tabel în care se află tabelul
CLUSTER_NAME	Numele cluster-ului din care face parte tabelul
PCT_FREE	Procentul de spațiu păstrat liber în interiorul fiecărui bloc
PCT_USED	Procentul de spațiu ce poate fi utilizat în fiecare bloc
INI_TRANS	Numărul inițial de tranzacții 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 conțin date
TEMPORARY	Y sau N, indică dacă tabelul este temporar (sau nu)

Informatii in DD

- Exemplu:

```
DESCRIBE USER_TABLES;  
SELECT    TABLE_NAME, NUM_ROWS, TEMPORARY  
FROM      USER_TABLES;
```

- Exemplu:

```
SELECT 'DROP TABLE' || OBJECT_NAME || ';'   
FROM USER_OBJECTS  
WHERE OBJECT_TYPE = 'TABLE';
```

Indecși

- Un index este un obiect al schemei unei baze de date care **crește viteza de execuție 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 și din identificatorul adresă ROWID**
- Indecșii sunt utilizați și întreținuți automat de către server-ul Oracle. Odată creat indexul, el nu necesită o acțiune directă din partea utilizatorului

Indecși

- Indecșii pot fi creați 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 definește o constrângere PRIMARY KEY sau UNIQUE asupra unei coloane sau unui grup de coloane.
- Numele indexului va fi același cu numele constrângerii.
- Indecșii, fiind obiecte ale schemei bazei, beneficiază de procesul de definire a unui obiect.

Indecși

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>;
```

Indecși

Mai mulți indecși asupra unui tabel nu implică întotdeauna interogări mai rapide.

Fiecare operație LMD care este permanentizată asupra unui tabel cu indecși asociați presupune actualizarea indecșilor respectivi

Prin urmare, **este recomandată** crearea de indecși numai în anumite situații:

- coloana conține o varietate mare de valori;
- coloana conține un număr mare de valori *null*;
- una sau mai multe coloane sunt utilizate frecvent împreună într-o clauză WHERE sau într-o condiție de JOIN;
- tabelul este mare și este de așteptat ca majoritatea interogărilor asupra acestuia să regăsească mai puțin de 2-4% din linii;

Indecși

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

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

Indecși

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

- indexați tabelele pentru care interogările selectează **un număr redus de rânduri** (sub 5%);
- indexați tabelele care sunt interogate folosind **clauze SQL simple**;
- nu indexați tabelele ce conțin puține înregistrări (accesul secvențial este mai simplu);
- nu indexați tabelele care sunt frecvent actualizate, deoarece **ștergerile, inserările și modificările sunt îngreunate de indecși**;
- indexați coloanele folosite frecvent în clauza **WHERE** sau în clauza **ORDER BY**;
- nu indexați coloanele ce conțin date asemănătoare (puține valori distincte);

Indecși

Exemplu:

```
CREATE INDEX upper_num_idx ON artist (UPPER(num));
```

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

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

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

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

Indecși

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),
                    ...);
```

- **Ștergerea** 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 șterge indexul (DROP INDEX) și se recreează (CREATE INDEX);
 - se utilizează comanda ALTER INDEX cu opțiunea REBUILD.

Secvențe

- O secvență este un obiect în baza de date care servește pentru a genera **întregi unici** în sistemele multi-utilizator, evitând apariția conflictelor și a blocării.
- Secvențele sunt memorate și generate **independent de tabele**; aceeași secvență poate fi utilizată pentru mai multe tabele.
- O secvență poate fi creată de un utilizator și poate fi partajată de mai mulți utilizatori.
- Crearea unei secvențe 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);

Secvențe

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

Secvențe

NEXTVAL și **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;
- condiția 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 mulțimi (UNION, INTERSECT, MINUS).

Secvențe

Din dicționarul datelor pot fi obținute informații despre secvențe folosind vizualizarea USER_SEQUENCES.

Exemplu:

- Să se creeze o secvență *domeniuseq* care să fie utilizată pentru a insera noi domenii în tabelul domeniu și să se insereze un nou domeniu.
- Să se afișeze informațiile referitoare la secvența *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, moștenind caracteristicile obiectelor la care se referă.
- Vizualizarea, fiind virtuală, **nu solicită o alocare de memorie** pentru date.
- Textul cererii (SELECT) care definește 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 și execută cererea.
 - Dacă sunt utilizate clauzele UNION, GROUP BY și CONNECT BY, atunci Oracle nu determină fuzionarea, el va rezolva vizualizarea și apoi va aplica cererea rezultatului obținut.

Vizualizări

- O vizualizare reflectă la orice moment **conținutul** exact al tabelelor de bază.
- Orice modificare efectuată asupra tabelelor se realizează instantaneu și asupra vizualizării.
- Ștergerea unui tabel implică **invalidarea vizualizărilor asociate** tabelului și nu ștergerea acestora.
- Vizualizările sunt definite pentru:
 - furnizarea unui nivel mai înalt de **securizare a bazei** (restricționarea accesului la date);
 - simplificarea formulării unei cereri;
 - mascarea complexității datelor;
 - afișarea datelor într-o altă reprezentare decât cea a tabelelor de bază;
 - asigurarea confidențialității anumitor informații;

Vizualizări

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

Vizualizări

- Operațiile LMD asupra vizualizărilor complexe sunt restricționate de următoarele **reguli**:
 - nu se poate insera, actualiza sau șterge o linie dintr-o vizualizare dacă aceasta conține **funcții grup, clauza GROUP BY, cuvântul cheie DISTINCT sau pseudocoloana ROWNUM**;
 - nu se poate adăuga sau modifica o linie dintr-o vizualizare, dacă aceasta conține **coloane definite prin expresii**;
- Nu pot fi adăugate linii printr-o vizualizare, dacă tabelul de bază conține coloane care au **constrângerea NOT NULL** și 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 referință la obiecte care încă nu sunt create. Deși vizualizarea va fi creată, utilizatorul nu poate să o folosească.
- **NO FORCE** este implicită și se referă la faptul că vizualizarea este creată numai dacă tabelele de bază există.
- Cererea este o comandă SELECT care poate să conțină 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 nici o operație LMD nu poate fi executată asupra vizualizării.

Vizualizări

Reactualizarea tabelelor implică reactualizarea corespunzătoare a vizualizărilor. Reactualizarea vizualizărilor implică reactualizarea tabelelor de bază? NU! Există restricții 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, șterse sau actualizate date din vizualizări ce conțin:
 - operatorul DISTINCT;
 - clauzele GROUP BY, HAVING, START WITH, CONNECT BY;
 - pseudo-coloana ROWNUM;
 - funcții grup;
 - operatori de mulțimi.
- Nu pot fi inserate sau actualizate date care ar încălca constrângerile din tabellele de bază.

Vizualizări

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

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

- Regula fundamentală este că orice operație 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 și cheie a vizualizării.
 - Deci, un tabel protejat prin cheie este un tabel ale cărui chei se păstrează și la nivel de vizualizare.

Vizualizări

- Asupra unui *join view* pot fi aplicate instrucțiunile INSERT, UPDATE sau DELETE, doar dacă sunt îndeplinite următoarele condiții:
 - instrucțiunea LMD afectează numai unul dintre tabelele de bază;
 - în cazul instrucțiunii **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 instrucțiunii **DELETE**, rândurile unei vizualizări pot fi șterse numai dacă există un tabel în join protejat prin cheie și numai unul (în caz contrar, Oracle nu ar ști din care tabel să șteargă);
 - în cazul instrucțiunii **INSERT**, toate coloanele în care sunt inserate valori trebuie să provină dintr-un tabel protejat prin cheie.