

Limbajul de definire a datelor (LDD) (partea I)

- În general, instrucțiunile *LDD* sunt utilizate pentru **definirea structurii corespunzătoare obiectelor unei scheme**: tabele, vizualizări, vizualizări materializate, indecși, sinonime, clustere, proceduri și funcții stocate, declanșatori, pachete stocate etc.
- Aceste instrucțiuni permit:
 - **crearea, modificarea și suprimarea** obiectelor unei scheme și a altor obiecte ale bazei de date, inclusiv baza însăși și utilizatorii acesteia (**CREATE**, **ALTER**, **DROP**);
 - **modificarea numelor obiectelor** unei scheme (**RENAME**);
 - **ștergerea datelor din obiectele unei scheme, fără suprimarea structurii** obiectelor respective (**TRUNCATE**).
- Implicit, o instrucțiune *LDD* permanentizează (**COMMIT**) efectul tuturor instrucțiunilor precedente și marchează începutul unei noi tranzacții.
- Instrucțiunile *LDD* au **efect imediat asupra bazei de date** și înregistrează informația în dicționarul datelor.
- Definirea unui obiect presupune: **crearea** (CREATE), **modificarea** (ALTER) și **suprimarea** sa (DROP).
- **Reguli de numire a obiectelor bazei de date**
 - Identificatorii obiectelor trebuie să înceapă cu o literă și să aibă maximum 30 de caractere, cu excepția numelui bazei de date care este limitat la 8 caractere și celui al legăturii unei baze de date, a cărui lungime poate atinge 128 de caractere.
 - Numele poate conține caracterele A-Z, a-z, 0-9, _, \$ și #.
 - Două obiecte ale aceluiași utilizator al server-ului Oracle nu pot avea același nume.
 - Identificatorii nu pot fi cuvinte rezervate ale server-ului Oracle.
 - Identificatorii obiectelor nu sunt *case-sensitive*.

Definirea tabelelor

1. Crearea tabelelor

- Formele simplificate ale comenzii de creare a tabelelor sunt:

```
CREATE TABLE nume_tabel (  
    coloana_1 tip_date [DEFAULT valoare]  
                                [constrangere_nivel_coloana [constrangere_nivel_coloana]...],  
    .....  
    coloana_n tip_date [DEFAULT valoare]  
                                [constrangere_nivel_coloana [constrangere_nivel_coloana]...],  
    [constrangeri_nivel_tabel]  
);
```

SAU

CREATE TABLE nume_tabel [(coloana_1,..., coloana_n)]
AS subcerere;

➤ **Constrângerile** definite asupra unui tabel pot fi de următoarele tipuri:

- **NOT NULL** - coloana nu poate conține valoarea *Null*; (**NOT NULL**)
- **UNIQUE** – pentru coloane sau combinații de coloane care trebuie să aibă valori unice în cadrul tabelului; (**UNIQUE (col1, col2, ...)**)
- **PRIMARY KEY** - identifică în mod unic orice înregistrare din tabel. Implică **NOT NULL + UNIQUE**; (**PRIMARY KEY (col1, col2, ...)**)
- **FOREIGN KEY** - stabilește o relație de cheie externă între o coloană a tabelului și o coloană dintr-un tabel specificat.

[**FOREIGN KEY** nume_col]
REFERENCES nume_tabel(nume_coloana)
 [**ON DELETE** {**CASCADE**| **SET NULL**}]

- **FOREIGN KEY** este utilizat într-o constrângere la nivel de tabel pentru a defini coloana din tabelul „copil”;
- **REFERENCES** identifică tabelul „părinte” și coloana corespunzătoare din acest tabel;
- **ON DELETE CASCADE** determină ca, odată cu ștergerea unei linii din tabelul „părinte”, să fie șterse și liniile dependente din tabelul „copil”;
- **ON DELETE SET NULL** determină modificarea automată a valorilor cheii externe la valoarea *null*, atunci când se șterge valoarea „părinte”.
- **CHECK**- o condiție care să fie adevărată la nivel de coloană sau linie (**CHECK (conditie)**).

Obs:

- Constrângerile pot fi create o dată cu tabelul sau adăugate ulterior cu o comandă **ALTER TABLE**.
- Constrângerile de tip **CHECK** se pot implementa la nivel de coloană doar dacă nu referă o altă coloană a tabelului.
- În cazul în care cheia primară (sau o cheie unică) este compusă, ea nu poate fi definită la nivel de coloane, ci doar la nivel de tabel.
- Constrângerea de tip **NOT NULL** se poate declara doar la nivel de coloană.

➤ Principalele **tipuri de date** pentru coloanele tabelelor sunt următoarele:

Tip de date	Descriere
VARCHAR2(n) [BYTE CHAR]	Definește un șir de caractere de dimensiune variabilă, având lungimea maximă de <i>n</i> octeți sau caractere. Valoarea maximă a lui <i>n</i> corespunde la 4000 octeți, iar cea minimă este de un octet sau un caracter.
CHAR(n) [BYTE CHAR]	Reprezintă un șir de caractere de lungime fixă având <i>n</i> octeți sau caractere. Valoarea maximă a lui <i>n</i> corespunde la 2000 octeți. Valoarea implicită și minimă este de un octet.
NUMBER(p, s)	Reprezintă un număr având <i>p</i> cifre, dintre care <i>s</i> cifre formează partea zecimală
LONG	Conține șiruri de caractere având lungime variabilă, care nu pot ocupa mai mult de 2GB.
DATE	Reprezintă date calendaristice valide, între 1 ianuarie 4712 î.Hr. și 31 decembrie 9999 d.Hr.

2. Modificarea (structurii) tabelelor

➤ **Modificarea structurii unui tabel** se face cu ajutorul comenzii **ALTER TABLE**. Forma comenzii depinde de tipul modificării aduse:

- adăugarea unei noi coloane (nu se poate specifica poziția unei coloane noi în structura tabelului; o coloană nouă devine automat ultima în cadrul structurii tabelului)

```
ALTER TABLE nume_tabel
```

```
ADD (coloana tip_de_date [DEFAULT expr][, ...]);
```

- modificarea unei coloane (schimbarea tipului de date, a dimensiunii sau a valorii implicite a acesteia; schimbarea valorii implicite afectează numai inserările care succed modificării)

```
ALTER TABLE nume_tabel
```

```
MODIFY (coloana tip_de_date [DEFAULT expr][, ...]);
```

- eliminarea unei coloane din structura tabelului:

```
ALTER TABLE nume_tabel
```

```
DROP COLUMN coloana;
```

Obs:

- dimensiunea unei coloane numerice sau de tip caracter poate fi mărită, dar nu poate fi micșorată decât dacă acea coloană conține numai valori **null** sau dacă tabelul nu conține nici o linie.
- tipul de date al unei coloane poate fi modificat doar dacă valorile coloanei respective sunt **null**.
- o coloană **CHAR** poate fi convertită la tipul de date **VARCHAR2** sau invers, numai dacă valorile coloanei sunt **null** sau dacă nu se modifică dimensiunea coloanei.

➤ Comanda **ALTER** permite **adăugarea unei constrângeri într-un tabel existent, eliminarea, activarea sau dezactivarea constrângerilor**.

- Pentru adăugare de constrângeri, comanda are forma:

```
ALTER TABLE nume_tabel
```

```
ADD [CONSTRAINT nume_constr] tip_constr (coloana);
```

- Pentru eliminare de constrângeri:

```
ALTER TABLE nume_tabel
```

```
DROP PRIMARY KEY | UNIQUE(col1, col2, ...) | CONSTRAINT nume_constr;
```

- Pentru activare/dezactivare constrângere:

```
ALTER TABLE nume_tabel
```

```
MODIFY CONSTRAINT nume_constr ENABLE/DISABLE;
```

sau

```
ALTER TABLE nume_tabel
```

```
ENABLE/ DISABLE CONSTRAINT nume_constr;
```

3. Suprimarea tabelelor

➤ Ștergerea fizică a unui tabel, inclusiv a înregistrărilor acestuia, se realizează prin comanda:

```
DROP TABLE nume_tabel;
```

➤ Pentru ștergerea conținutului unui tabel și păstrarea structurii acestuia se poate utiliza comanda:

```
TRUNCATE TABLE nume_tabel;
```

!!!Obs: Fiind operație **LDD**, comanda **TRUNCATE** are efect definitiv. De asemenea, se **reseteaza** și număratoarea pentru coloanele cu autoincrement.

4. Redenumirea tabelelor

Comanda **RENAME** permite redenumirea unui tabel, vizualizare sau secvență.

RENAME nume1_obiect **TO** nume2_obiect;

Obs:

- În urma redenumirii sunt transferate automat constrângerile de integritate, indecșii și privilegiile asupra vechilor obiecte.
- Sunt invalidate toate obiectele ce depind de obiectul redenumit, cum ar fi vizualizări, sinonime sau proceduri și funcții stocate.

5. Consultarea dicționarului datelor

Informații despre tabelele create se găsesc în vizualizările:

- **USER TABLES** – informații complete despre tabelele utilizatorului.
- **TAB** – informații de bază despre tabelele existente în schema utilizatorului.

Informații despre constângeri găsim în **USER CONSTRAINTS**, iar despre coloanele implicate în constrângeri în **USER CONS COLUMNS**.

Exerciții

1. Să se creeze tabelul **ANGAJATI_pnu** (pnu se alcatuiește din prima literă din prenume și primele două din numele studentului) corespunzător schemei relaționale:

ANGAJATI_pnu(cod_ang number(4), nume varchar2(20), prenume varchar2(20), email char(15), data_ang date, job varchar2(10), cod_sef number(4), salariu number(8, 2), cod_dep number(2))

În următoarele moduri:

- a) cu precizarea cheilor primare **la nivel de coloană** și a constrângerilor NOT NULL pentru coloanele nume și salariu;
 - b) cu precizarea cheii primare **la nivel de tabel** și a constrângerilor NOT NULL pentru coloanele nume și salariu.
- Se presupune că valoarea implicită a coloanei **data_ang** este SYSDATE.

Obs: Nu pot exista două tabele cu același nume în cadrul unei scheme, deci recrearea unui tabel va fi precedată de suprimarea sa prin comanda:

DROP TABLE ANGAJATI_pnu;

2. Adăugați următoarele înregistrări în tabelul **ANGAJATI_pnu**:

Cod_ang	Nume	Prenume	Email	Data_ang	Job	Cod_sef	Salariu	Cod_dep
100	Nume1	Prenume1	Null	Null	Director	null	20000	10
101	Nume2	Prenume2	Nume2	02-02-2004	Inginer	100	10000	10
102	Nume3	Prenume3	Nume3	05-06-2000	Analist	101	5000	20
103	Nume4	Prenume4	Null	Null	Inginer	100	9000	20
104	Nume5	Prenume5	Nume5	Null	Analist	101	3000	30

Prima si a patra înregistrare vor fi introduse specificând coloanele pentru care introduceți date efectiv, iar celelalte vor fi inserate fără precizarea coloanelor în comanda INSERT. Salvați comenzile de inserare.

3. Introduceți coloana comision în tabelul ANGAJATI_pnu. Coloana va avea tipul de date NUMBER(4,2).
4. Este posibilă modificarea tipului coloanei salariu în NUMBER(6,2) – 6 cifre si 2 zecimale?
5. Setati o valoare **DEFAULT** pentru coloana salariu.
6. Modificati tipul coloanei comision în NUMBER(2, 2) și al coloanei salariu la NUMBER(10,2), în cadrul aceleiași instrucțiuni.
7. Actualizati valoarea coloanei comision, setând-o la valoarea 0.1 pentru salariații al căror job începe cu litera A. (UPDATE)
8. Modificati tipul de date al coloanei email în VARCHAR2.
9. Adăugați coloana nr_telefon în tabelul ANGAJATI_pnu, setându-i o valoare implicită.
10. Vizualizați înregistrările existente. Suprimați coloana nr_telefon.

Ce efect ar avea o comandă ROLLBACK în acest moment?

11. Creați și tabelul **DEPARTAMENTE_pnu**, corespunzător schemei relaționale:

DEPARTAMENTE_pnu (cod_dep# number(2), nume varchar2(15), cod_director number(4)) specificând doar constrângerea NOT NULL pentru nume (nu precizați deocamdată constrângerea de cheie primară).

```
CREATE TABLE departamente_pnu ( ... );
DESC departamente_pnu
```

12. Introduceți următoarele înregistrări în tabelul DEPARTAMENTE_pnu:

Cod_dep	Nume	Cod_director
10	Administrativ	100
20	Proiectare	101
30	Programare	Null

13. Se va preciza apoi cheia primara cod_dep, fără suprimarea și recrearea tabelului (comanda ALTER).

Obs:

- Introducerea unei constrângeri după crearea tabelului, presupune ca toate liniile existente în tabel la momentul respective să satisfacă noua constrângere.
- Acest mod de specificare a constrângerilor permite numirea acestora.
- În situația în care constrângerile sunt precizate la nivel de coloană sau tabel (în CREATE TABLE) ele vor primi implicit nume atribuite de sistem, dacă nu se specifică vreun alt nume într-o clauză CONSTRAINT.

Exemplu : CREATE TABLE alfa (
X NUMBER
CONSTRAINT nn_x NOT NULL,
Y VARCHAR2 (10) NOT NULL);

14. Să se precizeze **constrângerea de cheie externă** pentru coloana **cod_dep** din **ANGAJATI_pnu**:

a) fără suprimarea tabelului (ALTER TABLE);

b) prin suprimarea și recrearea tabelului, cu precizarea noii constrângeri la nivel de coloană ({DROP, CREATE} TABLE). De asemenea, se vor mai preciza constrângerile (la nivel de coloană, dacă este posibil):

- PRIMARY KEY pentru cod_ang;
- FOREIGN KEY pentru cod_sef;
- UNIQUE pentru combinația nume + prenume;
- UNIQUE pentru email;
- NOT NULL pentru nume;
- verificarea cod_dep > 0;
- verificarea ca salariul sa fie mai mare decat comisionul*100.

15. Suprimați și recreați tabelul, specificând toate constrângerile la nivel de tabel (în măsura în care este posibil).

■ Adaugarea constrangerilor la nivel de tabel:

```
CREATE TABLE ANGAJATI_PNU
(cod_ang number(4),
 nume varchar2(20) constraint nume_pnu not null,
 prenume varchar2(20),
 email char(15),
 data_ang date default sysdate,
 job varchar2(10),
 cod_sef number(4),
 salariu number(8, 2) constraint salariu_pnu not null,
 cod_dep number(2),
 comision number(2,2),
 constraint nume_prenume_unique_pnu unique(nume,prenume),
 constraint verifica_sal_pnu check(salariu > 100*comision),
 constraint pk_angajati_pnu primary key(cod_ang),
 unique(email),
 constraint sef_pnu foreign key(cod_sef) references angajati_pnu(cod_ang),
 constraint fk_dep_pnu foreign key(cod_dep) references departamente_pnu (cod_dep),
 check(cod_dep > 0)
);
```

16. Reintroduceți date în tabel, utilizând (și modificând, dacă este necesar) comenzile salvate anterior.

```
INSERT INTO angajati_pnu
VALUES(100, 'nume1', 'prenume1', 'email1', sysdate, 'Director ',null, 20000,10, 0.1);
```

```
INSERT INTO angajati_pnu
VALUES(101,'nume2' , 'prenume2', 'email2',to_date('02-02-2004','dd-mm-
yyyy'),'Inginer', 100, 10000 ,10, 0.2);
```

```
INSERT INTO angajati_pnu
VALUES(102,'nume3' , 'prenume3', 'email3',to_date('05-06-2000','dd-mm-
yyyy'),'Analist', 101, 5000 ,20, 0.1);
```

```
INSERT INTO angajati_pnu
VALUES(103,'nume4','prenume4', 'email4', sysdate, 'Inginer ',100,9000,20, 0.1);
```

```
INSERT INTO angajati_pnu
VALUES(104,'nume5', 'prenume5', 'email5', sysdate, 'Analist', 101, 3000 ,30, 0.1);
```

17. Analizați structura vizualizărilor USER_TABLES, TAB, USER_CONSTRAINTS.

Obs: Pentru a afla informații despre tabelele din schema curentă, sunt utile cererile:

```
SELECT * FROM tab;
```

sau

```
SELECT table_name FROM user_tables;
```

18. a) Listați informațiile relevante (cel puțin nume, tip și tabel) despre constrângerile asupra tabelor angajati_pnu și departamente_pnu.

```
SELECT constraint_name, constraint_type, table_name
FROM user_constraints
WHERE lower(table_name) IN ('angajati_pnu', 'departamente_pnu');
```

Obs: Tipul constrângerilor este marcat prin:

- P - pentru cheie primară
- R – pentru constrângerea de integritate referențială (cheie externă);
- U – pentru constrângerea de unicitate (UNIQUE);
- C – pentru constrângerile de tip CHECK.

b) Aflați care sunt coloanele la care se referă constrângerile asupra tabelor angajati_pnu și departamente_pnu.

```
SELECT table_name, constraint_name, column_name
FROM user_cons_columns
WHERE LOWER(table_name) IN ('angajati_pnu', 'departamente_pnu');
```

19. Introduceți constrângerea NOT NULL asupra coloanei email.

```
ALTER TABLE angajati_pnu
```

```
MODIFY(email not null);
```

20. (Încercați să) adăugați o nouă înregistrare în tabelul ANGAJATI_pnu, care să corespundă codului de departament 50. Se poate?

21. Adăugați un nou departament, cu numele Analiza, codul 60 și directorul null în DEPARTAMENTE_pnu. COMMIT.

```
INSERT INTO departamente_pnu  
VALUES (60, 'Analiza', null);
```

```
SELECT * FROM departamente_pnu;  
COMMIT;
```

22. (Încercați să) ștergeți departamentul 20 din tabelul DEPARTAMENTE_pnu. Comentăți.

```
DELETE FROM departamente_pnu  
WHERE cod_dep = 20;
```

23. Ștergeți departamentul 60 din DEPARTAMENTE_pnu. ROLLBACK.

```
DELETE FROM departamente_pnu  
WHERE cod_dep = 60;
```

```
SELECT * FROM departamente_pnu;
```

24. Se dorește ștergerea automată a angajaților dintr-un departament, odată cu suprimarea departamentului. Pentru aceasta, este necesară introducerea clauzei **ON DELETE CASCADE** în definirea constrângerii de cheie externă. Suprimați constrângerea de cheie externă asupra tabelului **ANGAJATI_pnu** și reintroduceți această constrângere, specificând clauza **ON DELETE CASCADE**.

25. Ștergeți departamentul 20 din DEPARTAMENTE_pnu. Ce se întâmplă? Rollback.

```
DELETE FROM departamente_pnu  
WHERE cod_dep = 20;
```

26. Introduceți constrângerea de cheie externă asupra coloanei *cod_director* a tabelului DEPARTAMENTE_pnu. Se dorește ca ștergerea unui angajat care este director de departament să atragă după sine setarea automată a valorii coloanei *cod_director* la *null*.

```
ALTER TABLE departamente_pnu  
ADD CONSTRAINT cod_director_fk FOREIGN KEY(cod_director)  
REFERENCES angajati_pnu (cod_ang) ON DELETE SET NULL;
```

27. Actualizați tabelul **DEPARTAMENTE_PNU**, astfel încât angajatul având codul 102 să devină directorul departamentului 30. Ștergeți angajatul având codul 102 din tabelul **ANGAJATI_pnu**. Analizați efectele comenzii. Rollback.


```
ALTER TABLE departamente_pnu
ADD CONSTRAINT cod_director_fk FOREIGN KEY(cod_director)
REFERENCES angajati_pnu (cod_ang) ON DELETE SET NULL;
```

```
UPDATE departamente_pnu
SET cod_director = 102
WHERE cod_dep = 30;
```

```
SELECT * FROM departamente_pnu;
SELECT * FROM angajati_pnu;
```

```
DELETE from angajati_pnu where cod_ang = 102;
```

- Putem sterge angajatul 102 pentru ca nu are subalterni si in momentul stergerii, conform constrangerii ON DELETE SET NULL, cod_director devine NULL;

```
ROLLBACK;
```

Este posibilă suprimarea angajatului având codul 101? Comentăți.

- Nu putem sterge angajatul 101 deoarece are subalterni;

28. Adăugați o constrângere de tip **check** asupra coloanei salariu, astfel încât acesta să nu poată depăși 30000.

```
ALTER TABLE angajati_pnu
ADD CONSTRAINT v_sal_pnu CHECK (salariu <= 30000);
```

```
UPDATE angajati_pnu
SET salariu = 35000
where cod_ang = 100; -- nu putem adauga un salariu mai mare de 30000, conform noii constrangeri adaugate.
```

29. Dezactivați constrângerea creată anterior.

```
ALTER TABLE angajati_pnu DISABLE CONSTRAINT v_sal_pnu;
```