

ALTER TABLE Nume-tablea { ADD | DROP | MODIFY } Definiri { cumpărare | constrângeri }

```
ALTER TABLE Angajat ADD (Funcția VARCHAR(30) [DEFAULT val] [NOT NULL],  
                          studii VARCHAR(20))  
                          CHAR(40)
```

ALTER TABLE Angajat (Funcția VARCHAR(40)) // modificare tip dată

babeie să fie compatibil cu cel definit anterior dacă există înregistrări

- diminuarea lungimii  $\rightarrow$  posibilă dacă nu se trezorează înregistrările
  - dacă nu sunt înregistrări, tipul de date poate fi incompatibil
  - adăugare contrângere:

```
ALTER TABLE Angajat ADD CONSTRAINT (Fundia NOT NULL)
```

```
ALTER TABLE Lucreză ADD CONSTRAINT (cnp CONSTRAINT cnp_pk REFERENCES Angajat(c  
P_nr CONSTRAINT P_nr_fk REFERENCES Proiect (P_nr),  
CONSTRAINT pk PRIMARY KEY (cnp, P_nr))
```

**DRoP :**

**ALTER TABLE NomeTabela DROP COLUMN Nome\_coluna [CASCADE]**

↑  
numai dacă este Primary key / Unique

```
ALTER TABLE Department DROP D-MR CASCADE
```

**ALTER TABLE NumeTabelă DROP ( lista coloane ) [CASCADE]**

- nu ne ating cămpurile ci ne declară urmăred (Oracle)

ALTER TABLE NumeTabelă SET UNUSED ( lista coloane ) [CASCADE]

- sunt găsite în bază dar nu mai sunt vizibile

ALTER TABLE Angajat SET UNUSED (Funcția, D\_nr)

- atât DROP cât și SET UNUSED nu pot fi aplicate dacă în urma lui Drop sau set unused se stergă ultimul camp din tabelă.

SET UNUSED - run-time

ALTER TABLE Angajat DROP UNUSED COLUMNS;

- dacă nu sunt coloane unused nu apar eroi
  - la drop column și coloana nu există  $\Rightarrow$  eroare

ALTER TABLE Nume\_tabelă DROP [CONSTRAINT] Nume\_Constrângere  
ALTER TABLE Angajat DROP [CONSTRAINT] (D\_nr REFERENCES Departament (D\_nr))

### Activare / Deactivare constrângere:

ALTER TABLE Nume\_tabelă MODIFY DISABLE [CONSTRAINT] → optional  
ALTER TABLE Nume\_tabelă MODIFY DISABLE CONSTRAINT (listă constrângeri) → obligatoriu

pt. ENABLE la fel

TRUNCATE Nume\_tabelă [REUSE STORAGE] (tabela pierdeaza spatiul alocat)

RENAME Nume\_vechi TO Nume\_nou

RENAME Angajat TO ang

- vizualizarea structurii unei tabele:

DESCRIBE Nume\_tabelă

SQL - DML: influențarea performanțe bazei de date (limbaj de cerere)  
query ↓  
algebra relatională

- limbaj reprocedural ~ declarativ

SELECT ... } Fraza SQL implicită în clauze SQL < obligatoriu  
optionale

SELECT lista cîmpuri, constante, funcții FROM lista tabele  
[WHERE condiții tip S, condiții join] ← logic (T/F)

select

[GROUP BY cîmpuri de grupare]

[HAVING condiții pe funcții (alg. rel.)]

[ORDER BY ordonare rezultat]

### Cerere SQL la o singură tabelă:

- rezultatul unei cereri SQL este o tabelă

SELECT Nume, Prenume FROM Angajat;  
P<Nume, Prenume>(Angajat)

SELECT DISTINCT Nume, Prenume FROM Angajat;

- se pot utiliza și constante:
  - siruri de caractere "" ; o astfel de const. devine cîmp nou în rezultat
  - separate prin virgulă
  - dacă este blank, se scrie între-un singur cuvânt (în rezultat)

SELECT 'Nume angajat', Nume, Prenume FROM Angajat;

	Numeangajat	Nume	Prenume
1.	Nume angajat	Popescu	Ion
2.			

↑  
constantă  
CHAR(12)

SELECT Nume, Prenume, Salariu \* 1.1 FROM Angajat

Nume Prenume Salariu \* 1.1 ← denumire cămpuri

operatorul de concatenare: - între cămpuri  
- între cămpuri și constante } CHAR

SELECT Nume || Prenume FROM Angajat

SELECT Nume || ' ' || Prenume FROM Angajat

SELECT \* FROM Angajat  
ALL

WHERE: - cerere pe o singură tabelă  
- condiție de tip select

WHERE (expresie logică)

SELECT Nume, Prenume, Funcția FROM Angajat WHERE Salariu > 1000 AND D\_nr = 3;

- se poate utiliza operatorul BETWEEN

SELECT Nume, Prenume FROM Angajat WHERE Salariu BETWEEN 500 AND 1000

- operatorul IN (op. de set) ~ se verifică dacă val. unui câmp se găsește într-o gamă de valori specificată.

SELECT Nume, Prenume FROM Angajat WHERE D\_nr IN {2,4,6}

ALIAS:

AS nume-mov

SELECT nume AS Nume\_familie, Prenume FROM Angajat

SELECT Nume AS 'Nume\_familie', Prenume FROM Angajat

SELECT 'Text-lung' AS Camp1, ... FROM Angajat

LIKE: - comparație între siruri de caractere

SELECT Nume, Prenume FROM Angajat WHERE Oras LIKE 'Buc%'

% - nu lungime ne specificată

'\_-' - înlocuire caracter

'i.o.'

Nume A...C LIKE 'A% C%'

Nume BETWEEN 'A%' AND 'D%' mai simplu

Nume LIKE 'A%' OR Nume LIKE 'B%' OR ...

NULL:

SELECT Nume, Prenume FROM Angajat WHERE cnpS = NULL FALS  
cnpS = ''

fiecarem IS NULL

SELECT Nume, Prenume FROM Angajat WHERE cnpS IS NULL.  
IS NOT NULL

Data: {xx/xx/xxxx}

în oracle: 'xx - xx - xxxx' 21 - Dec - 2007  
zi luna an

SELECT Nume, Prenume FROM Angajat WHERE Data.m > '21-DEC-1990'  
dec

nume = 'Popescu'  
'popescu' } case sensitive

ORDER BY :- se definește ordinea în rezultat

- poate utiliza oricare dintre numele de câmpuri din rezultat sau care nu se află în rezultat.

SELECT Nume, Prenume FROM Angajat ORDER BY Nume, Prenume  
ASC DESC

SELECT Nume, Prenume, Media FROM Student ORDER BY Media DESC, Nume [ASC],  
Prenume

SELECT Nume, Prenume FROM Angajat ORDER BY Salariu DESC, Nume, Prenume.

• aliasul nu poate fi utilizat decât în order by (în where nu se poate)

• se definește un câmp compus Nume + Prenume căruia îl ne precizează un alias și în rezultat se mai adaugă salariul ...

SELECT Nume||Prenume AS Numef, Salariu FROM Angajat  
WHERE D\_nr IN (2,3,7) ORDER BY Nume, Prenume câmpuri inițiale  
Numef alias

când în where apără doi primari sau unice, execuția cererii este mult mai rapidă fiindcă are index intern.

dacă în where apar cămpuri ce nu sunt unice sau pk, execuția este mai lentă (nu are index).

Cereri la mai multe tabele:

FROM lista tabele

-produs cartezian (nu se respectă)

-join

SELECT \* FROM Angajat, Departament; (permis în Oracle - produs cartezian)

SELECT \* FROM Angajat, Departament WHERE  $D\_m1 = D\_m2$  AND  $D\_n2 = 3$   
nu este confuzie      cauză de  
join                        tipuri

ALTER TABLE nume\_tabelă { ADD  
DROP  
MODIFY } definiții { comuni  
constrângeri }

ALTER TABLE Angajat ADD ( Functie VARCHAR(30) [DEFAULT] val  
[NOT NULL], Stadii VARCHAR(20))  
constrângere

{ -val. default  
-definire constrângere }

ALTER TABLE Angajat ( Functie VARCHAR(40) )

→ s-a făcut numai modificare tip date (de la varchar(30) la (40)).

OBS: modificarea tipului de date să nu fie incompatible cu cel inițial doar în tabelă există înregistrări.

- VARCHAR(40) = CHAR(40).

OBS: de modif. tipului de date are se efect modificarea urmărește corecțare, astă lucea este posibil numai de valoare comului respectiv este devenită definitie.

La o tabelă goală, modificarea tipului de date al unui camp este permisă și în situația în care noul tip de date de incompatible cu cel vechi.

Se le un camp se adaugă doar o constrângere, fără modif. tipului de date se folosește:

ALTER TABLE Angajat ADD CONSTRAINT

( Functie NOT NULL )

↑  
campul      ↑  
constrângerea

ALTER TABLE lucrăză ADD CONSTRAINT

( CNP CONSTRAINT cmp\_fk

REFERENCES Angajat (emp),

project\_nr CONSTRAINT project\_nr\_fk,

REFERENCES Proiect ( project\_nr ),

CONSTRAINT PK PRIMARY KEY (CNP, project\_nr) )

→ adăugare de constrângeri pt. tabelă lucrăză (setate cnp și project\_nr ca foreign keys dar nu ca primary keys pt tabelă lucrăză).

## DROP

ALTER TABLE nume-table DROP COLUMN nume-colonă  
[CASCADE]

→ poate fi optional

OBS: Stergerea unei constrângeri nu sterge necesar totate referințele la ea.

→ CASCADE se folosește de către compilator și este:

→ primary key

→ unique

Exemplu: Stergem depart-ur din tabelă ~~departament~~

ALTER TABLE departament DROP COLUMN depart-ur.

[CASCADE]

→ precizăm că CASCADE pt a se sterge depart-ur, din tabelă departament și că se sterge proprietatea lui de foreign key în tabelă angajat.

→ de nu precizezi CASCADE, constrângerea rămâne să fie neutilizată;

- DROP pt. mai multe colonă:

ALTER TABLE nume-table DROP (listă colonă) [CASCADE]

Problema care poate apărea în acest caz:

a) Stergerea constrângerii unei tabele este o operatie expensivă care încoreacă serverul. În Oracle, nu se sterg efectiv compunii cînd declară se fund unused. În SQL, avem posibilitatea de declararea unui camp/ set de compuniri se fund unused.

ALTER TABLE nume-table SET UNUSED (listă colonă) [CASCADE]

- DBMS moșcheoză lista colonă în UNUSED, sunt păstrate în baza de date și sunt vizibile.

Exemplu: declarăm în Angajat, funcție și depart-ur se fund unused.

ALTER TABLE ~~SEP-HAUSER~~ ~~departament~~ SET UNUSED

( Funcție , depart-ur.)

↑  
FK, deci următoare să punem  
CASCADE

- Drop și Set Unused nu pot fi aplicate în situație în care în nume op. Drop sau în nume op. Unused se produce stergerea ultimului camp din tabelă.

- Comenzile se au fapt declarate ca fiind Unused comenzi baza de date; pot defini compunere sau nu și să fie sălăi sălăi; nume cu ale stării; compunere nu sălăi; nume de acest tip sunt vizibile ca fiind ≠.

- Set Unused se face în etape de RUNTIME (în produsele BD). Dacă dorești să stergi efectiv compunere declarată UNUSED folosești:

ALTER TABLE Angajat DROP UNUSED COLUMNS

OBS: - de la tabelele în care se văd stergerea compunerilor declarate Unused ~~nu~~ și astfel de compunere, op. nu are efect și nu produce mesaj de eroare

- de la drop se invocă un camp care nu este în tabelă, SGBD produce mesaj de eroare.

Stergere de constrângere:

ALTER TABLE nume\_tabelă DROP [CONSTRAINT] nume\_constrângere  
[Stergerea unei constrângeri sărante își-a străbuit nume în BD (deci este un obiect pt BD);

- putem sterge și constrângerei foarte nume:

ALTER TABLE Angajat DROP [CONSTRAINT]

(depart - nr REFERENCES departament (depart - nr))

Activare / dezactivare constrângeri:

- face parte din categoria ALTER TABLE MODIFY;  
- structură:

ALTER TABLE nume\_tabelă MODIFY DISABLE nume\_constrângere

OBS: SQL suportă și între DISABLE și nume\_constrângere să nu avanțeze CONSTRAINT.

ALTER TABLE nume\_tabelă MODIFY DISABLE

CONSTRAINTS (listă - constrângere)

→ dezactivare a constrângerii la același tabelă simplifică utilizarea obligatorie a lui CONSTRAINTS.

→ pt setivore → ENABLE

→ ENABLE pt o singură constrangere  
pt o listă de constrangeri - analog cu DISABLE.

• MODIFY după ce modificările vor fi comp.

- pt un comp definit, nu putem schimba numele; și putem face efectiv sau il declarăm sau nu declarăm compul nou;

→ creare tabelă fără nicio înregistrare:

TRUNCATE Nume\_tabelă [ REUSE STORAGE]

→ stergere înregistrări, fără a modifica structura tabelă de la [REUSE STORAGE] pt tabelă care rămâne sparță doar (deși nu mai are înregistrări în ea).

→ O tabelă poate fi redenumită:

RENAME Nume\_Vechi TO Nume\_Nou

Exemplu:

RENAME Angajat TO Ang.

→ problema: PK-ul din Angajat a fost utilizată în REFERENCES; Angajare are în REFERENCES datele modificărilor introduse pe care să le facă SGBD.

→ Structura unei tabele se poate vizualiza în Oracle cu:

DESCRIBE Nume\_tabelă

- se descrie structura tabelă după ce numele compilor, tipuri de date și constrangeri;

DESCRIBE Nume\_BD → afișarea tuturor tabelilor care fac parte din acea BD.

OBS: Fiecare de standardele interioare, Standardul 2005 adună plus comunitatea referitoare la constrangerile fără nume care pot fi ignorate.

Comandă de stergere trebuie să fie identică cu ~~aceea~~ model de definire al constrângeri.