

Curs 6 → BD

Linabajul SQL

S-a numit initial SEQUEL dezv. de IBM → sistem R (după 1974)
 ↓ folosit intern

DB2 (dezv din
sistem R)

1987 → apare I standard SQL (ANSI) (versiunile (7) nu
 include toate componentele SQL standardizate; (7) particula-
 rități pt diverse linbaje)

SQL - DDL (Data Definition Language) → comandă orientată pe def. struct.
 de date CREATE TABLE (crearea struct. de tabele)
 ALTER TABLE (modificarea struct.)
 DROP TABLE (stergere tabelă)

SQL - DML (Data Manipulation Language)

SELECT ... (nu înseamnă select din algebra relațională)
 INSERT
 DELETE (stergere înreg din tabelă)
 UPDATE (modif. conținut înreg.)
 INDEX

SQL - VDL (View Definition Language) → comenzi DDL și comenzi
 manipулarea de VIEW (creare și manipulare view) DML

SQL - DBA → grupare de comenzi ce permit manipularea
 drepturilor de acces.

SQL - nu este un linaj procedural, prin cerere și se indică
 mediul de BD ce dorește să obțină și anume dorește să
 le obțină ⇒ linaj declarativ; modul de execuție a cererii
 este det de SQL în funcție de structura cererii.

Se formulează cererea declarativ, iar SQL decide cum
 se execută cererea.

SQL - DDL :

Forma cea mai simplă a unei constrângeri:

CREATE TABLE [schema] numeTabelă (NumeCâmp TipDat
[DEFAULT valoare], NumeCâmp TipData [DEFAULT valoare],..
↓
val implicită a câmpului

CBG: nu s-au introdus constrângeri; în fapt de mediu de BD există restricții referitoare la NumeCâmp; lungime, incepere cu literă, etc.)

- ex:

```
CREATE TABLE Angajat(  
    Nume VARCHAR(25),  
    Prenume VARCHAR(25),  
    DataN DATE,  
    CNP CHAR(13),  
)  
  
CREATE TABLE Student(  
    Nume VARCHAR(25),  
    :  
    An INTGER DEFAULT 1,  
    :  
)
```

→ orice tabelă trebuie să aibă o cheie primară

CONSTRÂNGERI: (3) mediu care nu acceptă constrângeri

Tipuri de constrângeri:

1) Primary key = aplicată pe un câmp sau pe multime de câmpuri forțează ca la popularea tabelii să nu (3) 2 înreg pt care val pt un câmp sau asociate să nu fie identice și nici nula.

(nu poate avea val null și nu pot (7) să intreg cu același val)

2) Not Null: nu permite stocarea de intreg în tabelă având val null pt cămpul pe care a fost aplicată

3) Unique: nu permite val identice pt cămpul resp la să intreg dar permite valori nule (OBS: null sunt diferențe în ceea ce mediu BD null₁ ≠ null₂)

OBS: 1) Primary key = Not Null + Unique

2) În tabelă poate fi mai multe chei candidate (care pot fi def drept cheie primară)

Cheie primară = cheie unică (serie + nr de buletine = cheie unică - dacă se elimină un camp din cheie, ei nu mai formează o cheie)

4) Foreign key face referire la o cheie primară sau la un cămp unic al altor tabele; întotdeauna se referă la un cămp sau asociatii de cămpuri din altă tabelă care a fost def fie primary key, fie unique.

5) Check: se impune ca val. stocate în tabelă pt cămpul cu o astfel de constrângere să valideze o anumită expresie logică.

Constrângările se pot defi la nivel de cămp, fie la nivel de tabelă, decizia aparținând utilizatorului. De regulă toate constr. pot fi defi la nivel de tabelă, dar și la nivel de cămp.

PRIMARY KEY:

→ nu mai un cămp poate avea constrângerea Primary key deci nu mai multe cămpuri candidate,

La nivel de câmp

Nume Câmp Tip date [CONSTRAINT NumeConstrângere]
 PRIMARY KEY



activare / dezactivare constrângere
 aceea se poate face fără
 nume

ex: CREATE TABLE Departament(
 DepNume CHAR(20),
 DepNr INTEGER PRIMARY KEY,
 DepLoc ...)

SAU:

DepNr INTEGER CONSTRAINT P_key PRIMARY KEY,
 ...
 nume_constrângere.

Situatie in care nu e posibila def Pkey pt camp:

La nivel de tabelă: se def după ultimul câmp al tabl
 [CONSTRAINT NumeConstrângere] PRIMARY KEY(
 Câmp 1, Câmp 2, ...)



Cheia primară este formată din asoc. câmpurilor
 1, 2, ...

ex: CREATE TABLE Lucreaza(
 CNP CHAR(13),
 PNr INTEGER,
 Ore INTEGER,

CONSTRAINT cheie-pris PRIMARY KEY(
 CNP, PNr)



La def unei chei primare se crează automat index
 primar după acel câmp pt a manipula ușor tabela,

considerând că accesul la primary key se face foarte des.
Se vehiculează mai ușor cheile primare de tip numeric,
decat cele sir de caractere.

NOT NULL:

Se def de regulă la nivel de câmp; nu se impune restricție
în ceea ce privește (7) la mai multe câmpuri;

Ex: I CREATE TABLE Angajat(

Nume VARCHAR(20) NOT NULL,
Prenume VARCHAR(20) CONSTRAINT Pre_NonNull
NOT NULL,

Salariu INTEGER NOT NULL,

:

)

II CREATE TABLE Angajat(

Nume VARCHAR(20),

Prenume VARCHAR(20),

:

Salariu INTEGER,

CONSTRAINT CAMP_NENUL NOT NULL(Nume, Prenume, Salariu)

→ dezactivarea constrângерilor în cazul II se face global pt
toate câmpurile, dacă se dorește dezactiv. individual tb
se scrie CONSTRAINT pt fiecare câmp în parte
UNIQUE:

Nu permite val. identice la câmpurile specificate la 2 sau
mai multe val. nule; ale regule se atribuie celorlalte chei
candidate ce nu sunt chei primare

```

ex: CREATE TABLE Student(
    NrMatriculo INTEGER CONSTRAINT PK PRIMARY KEY,
    CNP CHAR(13) CONSTRAINT unic UNIQUE,
    Serieci CHAR(2),
    NrCi NUMBER(6),
    :
    ) CONSTRAINT Biunic UNIQUE (Serieci, NrCi)

```

Se crează automat index pt UNIQUE => 3 indexe (pt NrMat, pt CNP și asociatia <Serieci, NrCi>)

OBS: 1) La oricare se poate remota la multe constrângeri.
(nu se poate activ/dezactiv)

2) Nu (7) probleme dacă se aplică la crearea tabelei, dar dacă (7) olijă tabela ar putea apărea eroi.

FOREIGN KEY:

Face referire la cheia primară sau cîmpuri declarate cu C. UNIQUE din alte tabele.

```

CREATE TABLE Angajat(
    CNP CHAR(13) PRIMARY KEY,
    :
    DepNr INTEGER CONSTRAINT cheiestr FOREIGN KEY
    REFERENCES Departament(D_Nr) ON DELETE CASCADE |
        cîmpul din Departament
    ON DELETE SET NULL,
    :
)

```

Se indică tabela și cîmpul sau cîmpurile în care acel cîmp este declarat ca UNIQUE sau PRIMARY KEY (nu se poate pură cîmpurile care nu au aceste constrângeri)

ON DELETE → cum se manipulează întreg din tabelă în care sunt declarat-o să fie strămută în cazul în care o lărgind din tabelă de ref. este stșarsă:

→ CASCADE: se stăruiește întreg din tabelă în care era FOREIGN KEY

→ SET NULL: campul devine null;

Departament
Primary key

D_Nr	1
2	
3	
4	

Anagajat
Primary Key
Foreign Key

CNP	...	D_Nr
2		
2		
2		
2		

Cascade: dacă se stăruiește întreg din Departament, ⇒ se stăruiește toti angajații;

Set Null: dacă angajatul cu D_Nr se face null.

OBS: dacă nu se face ON DELETE; implicit se consideră SET NULL.

mai multe constrângeri Foreign Key; aceste cămpuri pot fi în același timp UNIQUE sau P Key în tabela în care sunt declarat FOREIGN KEY,

ex: CREATE TABLE Lucrează(

CNP CHAR(13) CONSTRAINT CNPfk FOREIGN KEY

REFERENCES Anagajat (CNP) ON DELETE CASCADE

// dacă pleacă angajatul din companie se stăruiește toate informațiile legate de el

P_Nr INTEGER CONSTRAINT PNfk REFERENCES

Proiect (P_Nr) ON DELETE CASCADE

Ore NUMBER(2),

CONSTRAINT PK PRIMARY KEY (CNP, P_Nr))

OBS: 1) Declarația unei c. FK poate fi făcută numai de decl. unei c. UNIQUE sau PK în tabela referată, iar popularea unei tabele în care avem FK se face după popularea tabelei referite pt a se asigura consistență

2) C. de tip FK poate fi def și la nivel de tabel

CONSTRAINT PK PRIMARY KEY (cnp, P-Nr), CONSTRAINT

cnpFK FOREIGN KEY (cnp) REFERENCES ...

(def la sfârșitul tabeli)

↳ tb inclus an. FOREIGN KEY

-ex: CREATE TABLE Angajat(

CNP CHAR(13) PRIMARY KEY

!

CNP_S CHAR(13) CONSTRAINT cnpSFK REFERENCES
(CNP supervisor)

Angajat (CNP) ON DELETE SET NULL

↓

Foreign key la aceeași tabelă, tb def după PK

C. de FK tb să urmărească compatib. d.p.v. al tipului de date asociat cāmpului cheie străină cu tipul de date a cāmpului referit din tabela coresp. (crearea apare dacă nul nu are denumire coresp cu cea referată ex: char(20) varchar(25): dacă toate sunt nul și nu de 20 de caractere nu apare eroare; dacă tipurile sunt dif. tb aplicată o funcție de conversie a tipului.)

CHECK:

Poate fi aplicată unei cāmp. sau mai multor cāmpuri reprez. o cond. de validare:

CHECK decl.: CONSTRAINT NumelConstrangere CHECK
expresie logică

numeCāmpuri, numeCāmpuri + valori

```

CREATE TABLE STUDENT(
    ;
    ANU NUMBER(1) DEFAULT 1 CONSTRAINT valAnu CHECK
    ( Anu > 1 and Anu ≤ 5)
    ;
    ANU CHECK ( Anu BETWEEN(1,5))

```

Burse (Tip Bursă, Val_min, Val_max, CnpStud, Bursă)

```

CREATE TABLE Burse(
    ;
    Valmin NUMBER(3),
    Valmax NUMBER(5),
    ;
    Bursa NUMBER(5), CHECK ( Bursa BETWEEN (Valmin
    Valmax))

```

CBS: CHECK poate fi aplicată și unei tabele.

ALTER TABLE → reunește setul de instrucțiuni SQL prin care se poate modifica o tabelă existență d.p.v. structura.

→ Adăugare cărăpuri

→ Modificarea def. unei cărăpuri (tip date, constrângeri)

→ Stergere cărăpuri

Eoperările nu sunt întotdeauna permise (vezi CASCADE), trebuie să respecte integritatea BD.

ALTER TABLE NumeTabelă TIP_OP (Definire)

↓
numeCărăp, tipData

Ex: ALTER TABLE Angajat ADD (Functia VARCHAR(20) NOT NU

(se adaugă în ordine fizică după ultimul def.) ↓
fără constrângere dacă tabela este populată

Adaugare constrângeri:

ALTER TABLE NumeTabela ADD CONSTRAINTS (Definire constrângere)

Ex: ALTER TABLE Angajat ADD CONSTRAINTS (Nume NOT NULL, CNP PRIMARY KEY, DepNr REFERENCES Departament (DNr), ...)

OBS: Definirea constrângeri poate fi validă sau nu în funcție de structura datelor din tabela.

Modificare: ALTER TABLE NumeTabela MODIFY (Definire)

↓
presupăt se modifică

tipul de date al unei câmpuri, se modifică constrângerea aferentă unei câmpuri.

Ex: ALTER TABLE Angajat MODIFY (Nume VARCHAR(30))

OBS: dacă (f) e constrângere pt câmpul Nume, după ap. de mai sus ea se pierde.

ALTER TABLE Angajat MODIFY CONSTRAINTS (Definire constrângeri noi)

Activare / Dezactivare constrângere:

ALTER TABLE NumeTabela //MODIFY// DISABLE CONSTRAINTS (NumeConstrângeri)

Ex: ALTER TABLE Angajat DISABLE CONSTRAINTS (nume_constfk, nume_constnull)

ALTER TABLE NumeTabela ENABLE CONSTRAINTS (NumeConstrângere) (se activează constrângeri aleja definite)

Stergere de cāmp:

ALTER TABLE NumeTabelă DROP COLUMN nume_colonă
 (stergerea unei zg coloane)

-II-

DROP COLUMNS (lista coloane)

OBS: → Stergerea uneia sau mai multor col. poate avea ca efect
 stergerea unei constrig. la care e implicată; stergerea e
 col. care era PK → se poate propaga efectul la mai multe
 tabele.

→ Este neutrala; stergerea poate încrește serverul BD ⇒ se
 setează coloanele ca UNUSED (nu pot fi implicate în
 nici o operatie, nu se sterge fizic; alta coloană poate fi
 amintită la fel ca cea care era UNUSED)

→ Op inversă → RECALL (s-ar putea să nu mai respecte
 constrangerile, etc.)

→ Op. facute în DDL nu pot fi revocate (au COMMIT autor
 (dacă s-a fărsă o coloană DROP COLUMN ⇒ nu se mai poate
 revoca))