



UNIUNEA EUROPEANĂ



GUVERNUL ROMÂNIEI



Instrumente Structurale
2007-2013



Platformă de e-learning și curriculum e-content
pentru învățământul superior tehnic

Baze de date 1

4. SQL - Data Definition Language (DDL)

STUD

MATR	NUME	AN	GRUPA	DATAN	LOC	TUTOR	PUNCTAJ	CODS
----	-----	--	-----	-----	-----	-----	-----	----
1456	GEORGE	4	1141A	12-MAR-82	BUCURESTI		2890	11
1325	VASILE	2	1122A	05-OCT-84	PITESTI	1456	390	11
1645	MARIA	3	1131B	17-JUN-83	PLOIESTI		1400	11
3145	ION	1	2112B	24-JAN-85	PLOIESTI	3251	1670	21
2146	STANCA	4	2141A	15-MAY-82	BUCURESTI		620	21
3251	ALEX	5	2153B	07-NOV-81	BRASOV		1570	21
2215	ELENA	2	2122A	29-AUG-84	BUCURESTI	2146	890	21
4311	ADRIAN	3	2431A	31-JUL-83	BUCURESTI		450	24
3514	FLOREA	5	2452B	03-FEB-81	BRASOV		3230	24
1925	OANA	2	2421A	20-DEC-84	BUCURESTI	4311	760	24
2101	MARIUS	1	2412B	02-SEP-85	PITESTI	3514	310	24
4705	VOICU	2	2421B	19-APR-84	BRASOV	4311	1290	24

SPEC si BURSA

CODS	NUME	DOMENIU			
11	MATEMATICA	STIINTE EXACTE			
21	GEOGRAFIE	UMANIST			
24	ISTORIE	UMANIST			
TIP			PMIN	PMAX	SUMA
FARA BURSA			0	399	
BURSA SOCIALA			400	899	100
BURSA DE STUDIU			900	1799	150
BURSA DE MERIT			1800	2499	200
BURSA DE EXCEPTIE			2500	9999	300

OBIECTIV

Scopul acestui capitol este de a prezenta elementele limbajului pentru descrierea datelor (DDL) referitoare la:

- ◆ Tipurile de date permise pentru coloanele tabelelor,
- ◆ Crearea de noi tabele,
- ◆ Constrângeri de integritate,
- ◆ Dicționarul de date al sistemului.

TIPURI DE DATE

Sistemul Oracle pune la dispoziție un set optim de tipuri de date care pot fi asociate coloanelor unei tabele, grupate în mai multe categorii:

- ◆ Tipuri numerice scalare
- ◆ Tipuri scalare șir de caractere
- ◆ Tipuri scalare binare
- ◆ Tipuri pentru date calendaristice, timp și interval de timp
- ◆ Tipuri LOB (large object)
- ◆ Tipuri compuse: TABLE și VARRAY

TIPURI DE DATE (2)

- ◆ Aceste tipuri, cu unele diferențe în ceea ce privește dimensiunea maximă admisă, pot fi folosite și în limbajul procedural PL/SQL.
- ◆ În continuare sunt prezentate tipurile din primele cinci categorii.
- ◆ Crearea unei tabele cu coloane de tip TABLE și VARRAY și manipularea datelor de acest fel nu este prezentată în acest capitolul.

TIPURI NUMERICE SCALARE

Tip	Descriere
NUMBER	Număr real de dimensiune variabilă, cu 38 de cifre semnificative, având valori între 1E-130 și 10E125.
NUMBER(n)	Număr întreg cu maxim n cifre
NUMBER(n, z)	Număr real cu n cifre dintre care z zecimale
DEC, DECIMAL, NUMERIC	Subtipuri pentru NUMBER. Numere în virgulă fixă cu 38 de cifre semnificative.
DOUBLE PRECISION, FLOAT	Subtipuri pentru NUMBER. Numere în virgulă mobilă cu 38 de cifre semnificative.
REAL	Subtip pentru NUMBER. Numere în virgulă mobilă cu 18 cifre semnificative.

TIPURI NUMERICE SCALARE (2)

Tip	Descriere
INTEGER, INT, SMALLINT	Subtipuri pentru NUMBER. Numere întregi cu maxim 38 de cifre
BINARY_INTEGER	Întregi între -2^{31} și 2^{31} .
NATURAL, POSITIVE	Subtipuri pentru BINARY_INTEGER. Numere întregi non-negative, respectiv pozitive.
NATURALN, POSITIVEN	Subtipuri pentru BINARY_INTEGER. Numere întregi nenule non-negative, respectiv pozitive.
SIGNTYPE	Subtip al BINARY_INTEGER. Poate lua doar valorile -1, 0 și 1.
PLS_INTEGER	Întregi între -2^{31} și 2^{31} . Similar cu BINARY_INTEGER dar operațiile cu astfel de numere sunt mai rapide și în caz de depășire se ridică o excepție.

SIRURI

Tip	Descriere
CHAR(<i>n</i>), CHAR	Șir de caractere de lungime fixă, egală cu <i>n</i> . Valoarea maximă pentru <i>n</i> este 2000. Dacă <i>n</i> lipsește, șir de caractere de lungime 1.
CHARACTER, CHARACTER(<i>n</i>)	Identice cu cele anterioare. Introduse pentru compatibilitatea cu alte sisteme.
NCHAR(<i>n</i>)	Analog cu CHAR dar poate stoca șiruri scrise în seturi de caractere naționale (multioctet)
VARCHAR2(<i>n</i>)	Șir de caractere de lungime variabilă egală cu <i>n</i> . Valoarea maximă pentru <i>n</i> este 4000.
STRING(<i>n</i>), VARCHAR(<i>n</i>)	Identice cu VARCHAR2, introduse pentru compatibilitatea cu alte sisteme
NVARCHAR(<i>n</i>)	Analog cu VARCHAR. Poate stoca șiruri scrise în seturi de caractere naționale (multioctet)

SIRURI (2)

Tip	Descriere
LONG	Șir de caractere de maxim 2^{31} octeți. Este permisă doar o singură coloană de acest tip pentru o tabelă.
ROWID	Poate stoca un identificator pentru o linie dintr-o tabelă. Pentru conversia la/de la șir de caractere (18 caractere) se pot folosi funcțiile SQL ROWIDTOCHAR respectiv CHARTOROWID.
UROWID	Universal ROWID. Poate stoca un identificator logic și fizic de linie într-o tabelă, indexată sau nu precum și un identificator de linie extern (non-Oracle). Nu este necesară folosirea funcțiilor de conversie la/de la șir de caractere (conversie automată).

TIPURI BINARE

Tip	Descriere
RAW(<i>n</i>)	Similar cu VARCHAR2 dar conține date binare. Valoarea maximă pentru <i>n</i> este de 2000.
LONG RAW	Similar cu LONG dar conține date binare.

DATE, TIMP SI INTERVAL

Tip	Descriere
DATE	Data calendaristică (secol, an, lună, zi, oră, minut, secundă).
TIMESTAMP[(n)]	Extensie a tipului DATE. Conține și fracțiuni de secundă. Dacă este prezent, <i>n</i> specifică numărul de zecimale pentru acestea. Implicit <i>n</i> = 6
TIMESTAMP [(n)] WITH TIME ZONE	Extinde tipul TIMESTAMP conținând și o diferență între ora locală și ora universală (GMT)
TIMESTAMP [(n)] WITH LOCAL TIME ZONE	Similar cu tipul anterior dar la stocarea în baza de date valorile sunt convertite la ora zonei bazei de date iar la regăsire la ora zonei aplicației client.
INTERVAL YEAR [(n)] TO MONTH	Se stochează intervale de ani și luni. Precizia <i>n</i> specifică numărul de cifre pentru an (între 0 și 4, implicit 2).
INTERVAL DAY [(z)] TO SECOND [(s)]	Similar cu tipul anterior, dar pentru intervale de zile și secunde. valorile <i>z</i> și <i>s</i> sunt preciziile pentru zile, respectiv secunde (0-9, implicit 2 pentru <i>z</i> și 6 pentru <i>s</i>).

TIPURI LARGE OBJECT

- ◆ Aceste tipuri, introduse în ultimele versiuni ale sistemului, permit stocarea unor cantități mari de date pe coloanele unei tabele sau a unei referințe (numită și locator) către un fișier extern bazei de date.
- ◆ Manipularea valorilor de acest tip se face în PL/SQL cu ajutorul pachetelor de proceduri și funcții puse la dispoziție de sistem.
- ◆ Caracteristicile lor sunt următoarele:

TIPURI LOB – cont.

Tip	Descriere
CLOB	Șir de caractere de până la 4 GB. Se recomandă ca în aplicațiile noi să fie folosit în locul lui LONG.
NCLOB	Similar cu CLOB dar se pot stoca șiruri utilizând seturi naționale de caractere. Dimensiunea maximă este de asemenea de 4 GB
BLOB	Date binare de dimensiune până la 4 GB
BFILE	Date binare de dimensiune până la 4 GB stocate în fișiere externe. Nu participă la tranzacții, replicare și pot fi doar citite nu și modificate.

CREATE TABLE

- ◆ Cea mai simplă formă a cererii SQL de creare a unei noi tabele are următoarea sintaxă:

```
CREATE TABLE [schema.]nume_tabela
  (nume_coloana_1  tip_coloana_1 [DEFAULT
  expresie_1],
   nume_coloana_2  tip_coloana_2 [DEFAULT
  expresie_2],
   . . .
   nume_coloana_n  tip_coloana_n [DEFAULT
  expresie_n]);
```

unde:

- ◆ **nume_tabela** este numele tablei care se crează,
- ◆ **nume_coloana_1, nume_coloana_2, ...** sunt numele coloanelor acesteia

CREATE TABLE – cont.

- ◆ **tip_coloana_1, tip_coloana_2, ...** reprezintă tipurile de date pentru coloanele respective, alese dintre cele prezentate în paragraful anterior, cu specificarea, dacă este cazul, a dimensiunii maxime sau preciziei,
- ◆ clauza opțională **DEFAULT expresie_i** specifică o valoare implicită care este introdusă automat în acea coloană în cazul în care la adăugarea unei noi linii nu se specifică o valoare pe coloana respectivă,
- ◆ schema este numele de utilizator Oracle al proprietarului noii tabele. Valoarea implicită pentru acesta este numele utilizatorului care execută cererea de creare.

TREBUIE CA:

- ◆ Utilizatorul are drepturile necesare (dreptul sau privilegiul - în terminologia uzuală - de CREATE TABLE).
- ◆ Există spațiu de stocare pentru noua tabelă.
- ◆ Numele tabelii și al coloanelor respectă restricțiile uzuale Oracle (maxim 30 de caractere, începe cu o literă, conține litere, cifre și caracterele `_`, `$`, `#`, nu este cuvânt rezervat Oracle). Sunt permise și alte caractere dacă numele este inclus între ghilimele.
- ◆ Nu există deja un alt obiect cu același nume în aceeași schemă (pentru același utilizator Oracle).

LITERE MARI SI MICI

- ◆ Ca și în cazul cuvintelor cheie, literele mici și cele mari sunt considerate egale în numele de tabele și coloane.
- ◆ Exemplu: tabela de studenți din exemple este aceeași dacă la creare se folosește oricare din numele STUD, Stud sau StuD.

OBSERVATII

- ◆ Expresia din clauza opțională DEFAULT trebuie să se evalueze la o valoare de tip compatibil cu al coloanei respective. Ea poate conține:
 - ◆ Constante numerice sau șir de caractere,
 - ◆ Funcții SQL, inclusiv SYSDATE sau USER.
 - ◆ dar nu poate conține:
 - ◆ Numele unei coloane,
 - ◆ Numele unei pseudocoloane. (de exemplu pseudocoloanele definite de o secvență: NEXTVAL sau CURRVAL).

EXEMPLU

```
CREATE TABLE STUD (  
MATR NUMBER(4) ,  
NUME VARCHAR2(10) ,  
AN NUMBER(1) DEFAULT 1 ,  
GRUPA VARCHAR2(6) ,  
DATAN DATE ,  
LOC VARCHAR2(10) DEFAULT 'BUCURESTI' ,  
TUTOR NUMBER(4) ,  
PUNCTAJ NUMBER(4) DEFAULT 0 ,  
CODS NUMBER(2) ) ;
```

OBSERVAM CA:

- ◆ MATR, AN, TUTOR, PUNCTAJ și CODS care conțin valori de tip număr întreg au fost definite ca NUMBER(n) unde n este 1, 2 sau 4,
- ◆ NUME, GRUPA și LOC sunt șiruri de caractere de lungime variabilă definite ca VARCHAR2,
- ◆ DATAN care conține data nașterii studentului este de tipul DATE,
- ◆ Pentru unele dintre coloane au fost asociate valori implicite care vor fi stocate automat în acestea dacă la inserarea unei noi linii nu se specifică o valoare, nulă sau nenulă.
- ◆ Valorile implicite sunt compatibile cu tipul coloanelor respective (întreg sau șir de caractere, după caz).

- ◆ Crearea unei tabele în care se stochează date despre evenimente: momentul de început, durata și o descriere a acestora.

```
CREATE TABLE EVENIMENT (  
  COD NUMBER(10) ,  
  "MOMENT INCEPUT" TIMESTAMP(3) WITH LOCAL  
    TIME ZONE ,  
  DURATA INTERVAL DAY(2) TO SECOND(3) ,  
  "DESCRIERE (PE LARG)" LONG) ;
```

OBSERVAM CA:

- ◆ COD: un număr întreg de maxim 10 cifre,
- ◆ MOMENT ÎNCEPUT: conține momentul începutului unui eveniment în forma: data, ora, minutul, secunda și miimile de secundă: `TIMESTAMP(3)` arată că fracțiunile de secundă sunt memorate cu 3 zecimale. Numele coloanei conține un spațiu și a trebuit pus între ghilimele.
- ◆ DURATA: durata evenimentului în zile, ore, minute, secunde și fracțiuni de secundă. Numărul de zile poate avea maxim 2 cifre iar numărul de secunde maxim 3 zecimale.
- ◆ DESCRIERE (PE LARG): conține un text de descriere a evenimentului care poate avea până la 2 GB caractere. Se folosesc ghilimelele pentru că numele coloanei conține spații și paranteze.

CREATE din SELECT

```
CREATE TABLE [schema.]nume_tabela  
  [(descriere_coloana_1, ...,  
  descriere_coloana_n) ]  
AS  
  cerere_SELECT;
```

- ◆ Se crează o tabelă având numele specificat și aceeași structură cu a rezultatului returnat de SELECT.

CREATE din SELECT (2)

- ◆ **Descrierea coloanelor nu este prezentă în cerere:**
 - ◆ Numele coloanelor tabeli create precum și tipul acestora este identic cu al celor din rezultatul cererii SELECT.
 - ◆ Noua tabelă nu moștenește nici una dintre constrângerile de integritate ale tabeli/tabelelor din care provine rezultatul.
 - ◆ Dacă în lista de expresii din clauza SELECT există unele care nu returnează pentru capul de tabel al rezultatului un nume valid de coloană este obligatorie folosirea unor aliasuri de coloană.

CREATE din SELECT (3)

```
CREATE TABLE STUD11
AS
  SELECT MATR, NUME, PUNCTAJ*1.1 "PUNCTAJ MARIT"
  FROM STUD WHERE CODS = 11;
```

◆ Conținutul tabelii STUD11 va fi următorul:

MATR	NUME	PUNCTAJ MARIT
1456	GEORGE	3179
1325	VASILE	429
1645	MARIA	1540

◆ iar descrierea structurii sale este:

Nume coloana	Tip
MATR	NUMBER (4)
NUME	VARCHAR2 (10)
PUNCTAJ MARIT	NUMBER

CREATE din SELECT (4)

- ◆ Crearea tabelii se face inclusiv în cazul în care cererea SELECT nu returnează nici o linie. De exemplu:

```
CREATE TABLE STUD100
```

```
AS
```

```
SELECT MATR, NUME, PUNCTAJ*1.1 "PUNCTAJ  
MARIT"
```

```
FROM STUD
```

```
WHERE CODS = 100;
```

- ◆ va avea ca efect crearea tabelii STUD100 având aceeași structură cu STUD11 dar aceasta nu va conține nici o linie.

CREATE din SELECT (5)

- ◆ **Cererea conține descrierea coloanelor:**
- ◆ Descrierea unei coloane are următoarea formă:
`nume_coloana [DEFAULT expresie] [constrângeri_integritate]`
- ◆ Numărul de descrieri de coloană trebuie să fie egal cu numărul de coloane din rezultatul cererii SELECT.
- ◆ Numele coloanelor tabeli create este cel din descriere iar tipul lor este identic cu al celor din rezultatul cererii SELECT.
- ◆ Dacă descrierea unei coloane conține clauza DEFAULT expresie, se asociază acestei coloane valoarea implicită respectivă.
- ◆ Noua tabelă nu moștenește nici una dintre constrângerile de integritate ale tabeli/tabelelor din care provine rezultatul dar primește constrângerile de integritate din descriere, dacă acestea există

CREATE din SELECT (6)

```
CREATE TABLE STUD11
(NUMAR DEFAULT 0 NOT NULL, NUME, PUNCTE NOT NULL)
AS
  SELECT MATR, NUME, PUNCTAJ*1.1 "PUNCTAJ MARIT"
  FROM STUD
 WHERE CODS = 11;
```

- ◆ Conținutul tabelii este același, ultima coloană are alt nume, prima are o valoare implicită iar două dintre ele au asociată o constrângere de tip NOT NULL (nu se pot stoca valori nule pe acele coloane):

Nume coloana	Null?	Tip	Implicit
NUMAR	NOT NULL	NUMBER(4)	0
NUME		VARCHAR2(10)	
PUNCTE	NOT NULL	NUMBER	

CONSTRANGERI

- ◆ Constrângerile de integritate reprezintă reguli pe care valorile conținute într-o tabelă trebuie să le respecte.
- ◆ Ele previn introducerea de date eronate în baza de date și definesc forma corectă a valorilor respective dar nu iau în considerație semnificația acestora.
- ◆ Constrângerile de integritate sunt verificate automat de sistemul de gestiune atunci când au loc operații de modificare a conținutului tabelelor (adăugare, ștergere și modificare linii).
- ◆ În cazul în care noile valori nu sunt valide operația de modificare este rejectată de sistem și se generează o eroare.

TIPURI DE CONSTRANGERI

- ◆ NOT NULL: valorile nu pot fi nule
- ◆ PRIMARY KEY: definește cheia primară a unei tabele
- ◆ UNIQUE: definește o altă cheie a tabelei
- ◆ FOREIGN KEY: definește o cheie străină (externă)
- ◆ CHECK: introduce o condiție (expresie logică).

CONSTRANGERI (2)

- ◆ Fiecare constrângere de integritate poate avea asociat un nume specificat la crearea tabelului care permite activarea sau dezactivarea constrângerii și alte operații cu aceasta.
- ◆ În cazul în care nu se asociază un astfel de nume sistemul generează automat unul.
- ◆ Locul definirii unei constrângeri de integritate în cererea de creare a unui tabel poate fi:

CONSTRANGERI (3)

- ◆ În descrierea unei coloane, dacă acea constrângere se referă doar la aceasta (uzual se spune despre o astfel de constrângere că este definită *la nivel de coloană*),
- ◆ În continuarea listei de descrieri de coloane (*la nivel de tabelă*).
- ◆ În funcție de locul unde se găsește, sintaxa definirii unei constrângeri poate fi diferită.
- ◆ Pentru fiecare tip de constrângere sunt prezentate ambele sintaxe și exemple de folosire a lor.

DETALIERE CONSTRANGERI

- ◆ **NOT NULL:** valorile nu pot fi nule
- ◆ **PRIMARY KEY:** definește cheia primară a unei tabele
- ◆ **UNIQUE:** definește o altă cheie a tabelei
- ◆ **FOREIGN KEY:** definește o cheie străină (externă)
- ◆ **CHECK:** introduce o condiție (expresie logică).

NOT NULL

- ◆ Acest tip de constrângere se aplică unei coloane a noii tabele și specifică faptul că aceasta nu poate conține valori nule. El poate fi definit doar la nivel de coloană și are următoarea sintaxă:

```
coloana [CONSTRAINT nume_constr] NOT NULL
```

EXEMPLU

```
CREATE TABLE SPEC  
(CODS NUMBER(2),  
  NUME VARCHAR2(10) CONSTRAINT  
  NUMENENUL NOT NULL,  
  DOMENIU VARCHAR2(15) NOT  
  NULL);
```

DETALIERE CONSTRANGERI

- ◆ NOT NULL: valorile nu pot fi nule
- ◆ PRIMARY KEY: definește cheia primară a unei tabele
- ◆ UNIQUE: definește o altă cheie a tabelei
- ◆ FOREIGN KEY: definește o cheie străină (externă)
- ◆ CHECK: introduce o condiție (expresie logică).

PRIMARY KEY

◆ Sintaxa la nivel de coloană:

```
coloana [CONSTRAINT nume_constrangere]  
PRIMARY KEY
```

◆ Sintaxa la nivel de tabelă:

```
[, CONSTRAINT nume_constrangere] PRIMARY  
KEY (lista_coloane)
```

EXEMPLE

```
CREATE TABLE SPEC (  
  CODS NUMBER(2) CONSTRAINT SPEC_PK PRIMARY  
    KEY,  
  NUME VARCHAR2(10),  
  DOMENIU VARCHAR2(15));
```

Si la nivel de tabela:

```
CREATE TABLE SPEC (  
  CODS NUMBER(2),  
  NUME VARCHAR2(10),  
  DOMENIU VARCHAR2(15),  
  CONSTRAINT SPEC_PK PRIMARY KEY(CODS));
```

PK DIN > 1 ATRIBUT

```
CREATE TABLE BURSA (  
  PMIN NUMBER(4),  
  PMAX NUMBER(4),  
  TIP VARCHAR2(20),  
  SUMA NUMBER(4),  
  CONSTRAINT BURSA_PK PRIMARY KEY (PMIN, PMAX) );
```


DETALIERE CONSTRANGERI

- ◆ NOT NULL: valorile nu pot fi nule
- ◆ PRIMARY KEY: definește cheia primară a unei tabele
- ◆ UNIQUE: definește o altă cheie a tablei
- ◆ FOREIGN KEY: definește o cheie străină (externă)
- ◆ CHECK: introduce o condiție (expresie logică).

UNIQUE

- ◆ Sintaxa la nivel de coloană:

```
coloana [CONSTRAINT nume_constrangere]  
UNIQUE
```

- ◆ Sintaxa la nivel de tabelă:

- ◆

```
[, CONSTRAINT nume_constrangere]  
UNIQUE (lista_coloane)
```

EXEMPLUL 1

- ◆ Dacă în tabela de SPEC nu pot exista niciodată două specializări cu același nume, definiția anterioară se completează cu o constrângere de tip UNIQUE:

```
CREATE TABLE SPEC (  
  CODS NUMBER(2) CONSTRAINT SPEC_PK  
    PRIMARY KEY,  
  NUME VARCHAR2(10) CONSTRAINT NUMES_UNIC  
    UNIQUE,  
  DOMENIU VARCHAR2(15));
```

EXEMPLUL 1 – cont.

- ◆ Aceeasi constrangere, la nivel de tabela:

```
CREATE TABLE SPEC (  
  CODS NUMBER(2) CONSTRAINT SPEC_PK PRIMARY  
    KEY,  
  NUME VARCHAR2(10),  
  DOMENIU VARCHAR2(15),  
  CONSTRAINT NUMES_UNIC UNIQUE (NUME) );
```

EXEMPLUL 2

- ◆ În tabela BURSA nu pot exista două burse cu aceeași sumă. Cererea de creare anterioară devine:

```
CREATE TABLE BURSA (  
  PMIN NUMBER(4) ,  
  PMAX NUMBER(4) ,  
  TIP VARCHAR2(20) ,  
  SUMA NUMBER(4) CONSTRAINT SUMA_UNICA  
    UNIQUE ,  
  CONSTRAINT BURSA_PK PRIMARY KEY (PMIN,  
    PMAX) ) ;
```

- ◆ În acest caz în tabelă **nu pot exista** două linii care conțin aceeași valoare nenulă pe coloana SUMA dar **pot exista** oricâte linii cu valori nule pe această coloană.

CE E UNIC?

- ◆ În cazul în care o cheie definită cu UNIQUE conține mai multe coloane, verificarea se face astfel:
 - ◆ nu există două linii care au aceleași valori nenule pentru toate coloanele cheii.
 - ◆ nu există două linii care au aceleași valori nenule pentru unele coloane ale cheii și valori nule în rest.
- ◆ Exemplu: În cazul unei tabele NUMERE creată cu cererea:

```
CREATE TABLE NUMERE (  
  NUMAR1 NUMBER(4),  
  NUMAR2 NUMBER(4),  
  UNIQUE(NUMAR1, NUMAR2));
```

CONTINUT VALID

NUMAR1 NUMAR2

NULL NULL

NULL NULL

1000 2000

1000 3000

NULL 2000

1000 NULL

CONTINUT INVALID

NUMAR1 NUMAR2

NULL NULL

NULL NULL

1000 2000

1000 3000

NULL 2000

1000 NULL

1000 2000

NULL 2000

1000 NULL

DETALIERE CONSTRANGERI

- ◆ NOT NULL: valorile nu pot fi nule
- ◆ PRIMARY KEY: definește cheia primară a unei tabele
- ◆ UNIQUE: definește o altă cheie a tabelei
- ◆ FOREIGN KEY: definește o cheie străină (externă)
- ◆ CHECK: introduce o condiție (expresie logică).

FOREIGN KEY

- ◆ Prin această constrângere valorile unei coloane/unor coloane sunt forțate să fie doar dintre cele ale cheii unei tabele (*cheie primară sau UNIQUE*).
- ◆ Coloanele constrânse în acest fel formează ceea ce se numește în terminologia de specialitate o *cheie straină* sau *cheie externă* iar constrângerea mai este denumită și *de integritate referențială*.

SINTAXA

- ◆ La nivel de coloană:

```
coloana [CONSTRAINT nume_constrangere]  
REFERENCES tabela (coloana)  
[ON DELETE CASCADE |  
ON DELETE SET NULL]
```

- ◆ La nivel de tabelă:

```
[, CONSTRAINT nume_constrangere] FOREIGN  
KEY (lista_coloane)  
REFERENCES tabela (lista_coloane)  
[ON DELETE CASCADE |  
ON DELETE SET NULL]
```

- ◆ În cazul tabelii STUD avem două coloane care pot avea asociată o astfel de constrângere:
 - ◆ Coloana CODS poate fi constrânsă să conțină doar valori ale cheii primare a tabelii SPEC (formată dintr-o coloană cu același nume - CODS).
 - ◆ Coloana TUTOR poate fi constrânsă să conțină doar valori nenule ale cheii primare a lui STUD - MATR.
- ◆ În acest caz cererea anterioară de creare pentru STUD devine:

STUD:

```
CREATE TABLE STUD (  
MATR NUMBER(4) PRIMARY KEY,  
NUME VARCHAR2(10),  
AN NUMBER(1) DEFAULT 1,  
GRUPA VARCHAR2(6),  
DATAN DATE,  
LOC VARCHAR2(10) DEFAULT 'BUCURESTI',  
TUTOR NUMBER(4) REFERENCES STUD(MATR),  
PUNCTAJ NUMBER(4) DEFAULT 0,  
CODS NUMBER(2),  
CONSTRAINT CODS_FK FOREIGN KEY(CODS)  
REFERENCES SPEC(CODS));
```

OBSERVATII

- ◆ În cazul în care **tabela SPEC nu există** încă sistemul va semnala eroarea: *ORA-00942: table or view does not exist*
- ◆ În cazul în care tabela **SPEC există dar nu are cheia CODS** se semnalează eroarea: *ORA-02270: no matching unique or primary key for this column-list*
- ◆ După crearea lui STUD, dacă se încearcă **ștergerea tablei SPEC** se semnalează eroarea: *ORA-02449: unique/primary keys in table referenced by foreign keys*

OBSERVATII – cont.

- ◆ La **inserarea de noi linii** în tabela STUD, acestea trebuie să conțină pe coloanele CODS și TUTOR **fie valori care există deja** în SPEC.CODS și STUD.MATR, **fie valori nule**.
- ◆ La încercarea de a **șterge linii** din tabelele SPEC și STUD, dacă acestea au valori ale chei referite în STUD prin constrângerile de mai sus se va semnala de asemenea o **eroare**.
- ◆ La încercarea de a **modifica valorile cheii** în linii din tabelele SPEC și STUD, dacă acestea sunt referite în STUD se va semnala de asemenea o **eroare**.

CLAUZA ON DELETE

- ◆ În cazul în care se dorește ca o linie dintr-o tabelă conținând o valoare de cheie referită într-o altă tabelă (sau în aceeași tabelă) să poată fi ștearsă fără a se obține un mesaj de eroare, la definirea constrângerii se poate specifica modul în care se tratează această ștergere din perspectiva tabelului care referă valoarea:
 - ◆ Dacă se dorește ca la ștergerea liniei conținând o valoare de cheie să fie șterse suplimentar și toate liniile care referă această valoare, se specifică **ON DELETE CASCADE**
 - ◆ Dacă se dorește ca liniile care referă valoarea cheii respective să nu fie șterse, se specifică **ON DELETE SET NULL** care are următorul efect: în toate liniile care referă acea valoare ea este înlocuită cu valori nule.

EXEMPLU

```
CREATE TABLE STUD (  
MATR NUMBER(4) PRIMARY KEY,  
.  
.  
.  
TUTOR NUMBER(4) REFERENCES STUD (MATR)  
ON DELETE SET NULL,  
.  
.  
.  
CONSTRAINT CODS_FK FOREIGN KEY (CODS)  
REFERENCES SPEC (CODS)  
ON DELETE CASCADE);
```

EXEMPLU – cont.

- ◆ La **ștergerea unei specializări** din tabela SPEC vor fi șterși automat și toți studenții din tabela STUD având pe coloana CODS codul acesteia.
- ◆ La **ștergerea unui student** care este tutorul altor studenți, automat acestora li se va modifica valoarea pe coloana TUTOR la una nulă.
- ◆ Tabela **SPEC poate fi golită** - ceea ce va antrena și golirea tablei STUD - dar nu ștearsă.

FK PENTRU UNIQUE

- ◆ În cazul în care constrângerea se definește pentru o cheie de tip **UNIQUE** (care poate conține și valori nule) ea este verificată pentru cheile străine formate doar din valori nenule.

DETALIERE CONSTRANGERI

- ◆ NOT NULL: valorile nu pot fi nule
- ◆ PRIMARY KEY: definește cheia primară a unei tabele
- ◆ UNIQUE: definește o altă cheie a tablei
- ◆ FOREIGN KEY: definește o cheie străină (externă)
- ◆ CHECK: introduce o condiție (expresie logică).

CHECK

- ◆ Prin acest tip de constrângeri **valorile aflate pe o linie** a tabelii sunt forțate să verifice o condiție (expresie logică).
- ◆ Aceasta poate conține toate elementele prezentate anterior pentru conditii, inclusiv apeluri de funcții, cu câteva **excepții**:
 - ◆ apelurile unor funcții ca SYSDATE, USER, UID
 - ◆ referința la pseudocoloanele NEXTVAL, CURRVAL (specifice secvențelor), LEVEL și ROWNUM)
 - ◆ subcereri.
- ◆ O aceeași coloană poate participa **la mai multe constrângeri de acest tip**, având fiecare o altă condiție asociată.
- ◆ Condiția se evaluează **doar la nivelul liniei respective**, fără a se putea lua în considerare valori aflate pe alte linii ale tabelii.

SINTAXA

◆ La nivel de coloană:

```
coloana [CONSTRAINT nume_constrangere]  
CHECK (expresie_logica)
```

◆ La nivel de tabelă:

```
[,CONSTRAINT nume_constrangere]  
CHECK (expresie_logica)
```

EXEMPLU

- ◆ Tabela BURSA poate avea asociate mai multe constrângeri care verifică următoarele condiții:
 - ◆ PMIN și PMAX sunt pozitive
 - ◆ PMIN < PMAX
 - ◆ SUMA este între 0 și 500

- ◆ Cererea de creare va fi:

```
CREATE TABLE BURSA (  
  PMIN NUMBER(4) CHECK (PMIN >= 0),  
  PMAX NUMBER(4) CHECK (PMAX >= 0),  
  TIP VARCHAR2(20),  
  SUMA NUMBER(4) CHECK (SUMA BETWEEN 0 AND 500),  
  CONSTRAINT PMINPMAX CHECK (PMIN < PMAX));
```

EXEMPLU – cont.

- ◆ În cazul în care pe coloanele respective se găsesc **valori nule**, linia nu este rejectată ci se consideră că verifică expresia logică.
- ◆ Pentru tabela BURSA definită ca mai sus se pot introduce linii având valori nule pentru unele din coloanele PMIN, PMAX și SUMA fără a se semnala violarea constrângerilor de integritate de tip CHECK.
- ◆ Expresia logică asociată unei constrângeri CHECK **poate fi compusă**: cererea de creare de mai sus poate fi rescrisă prin combinarea tuturor condițiilor în una singură:

EXEMPLU – cont.

```
CREATE TABLE BURSA (  
  PMIN NUMBER(4), PMAX NUMBER(4),  
  TIP VARCHAR2(20),  
  SUMA NUMBER(4),  
  CONSTRAINT BURSA_CHK CHECK (PMIN < PMAX  
    AND PMIN >= 0 AND PMAX >= 0 AND SUMA  
    BETWEEN 0 AND 500));
```

DICTIONARUL DE DATE

- ◆ Una dintre cerințele fundamentale pentru ca un sistem de gestiune a bazelor de date să fie considerat cu adevărat relațional este aceea ca datele interne ale acestuia să fie organizate și stocate în același mod cu datele utilizatorilor și aplicațiilor care îl folosesc.
- ◆ În acest sens, și sistemul Oracle își ține aceste date, numite uzual *dictionarul de date al sistemului*, sub formă de tabele, iar utilizatorii care au drepturile necesare le pot accesa fie direct fie folosind o serie de vederi și sinonime puse la dispoziție de sistem.

DICTIONARUL DE DATE – cont.

- ◆ Există mai multe categorii de vederi prin care uzual aceste date sunt accesate, fiecare având ca și caracteristică un prefix al numelui care definește categoria respectivă.
- ◆ Cele mai folosite vederi intră în categoriile definite de prefixele următoare:
 - ◆ Prefixul **USER_** este comun vederilor prin care se pot accesa informații despre obiectele deținute de utilizator.
 - ◆ Prefixul **ALL_** este specific vederilor care conțin informații despre toate tabelele la care utilizatorul are acces.
 - ◆ Prefixul **DBA_** este folosit pentru vederile accesibile doar utilizatorilor care au drepturi (privilegii) de administrator al bazei de date.

EXEMPLE

- ◆ Prezentarea tuturor acestor vederi nu face obiectul lucrării de față.
- ◆ În continuare sunt prezentate câteva dintre cele mai importante, legate mai ales de inventarul de tabele și constrângeri de integritate existent la un moment dat în sistem:
 - ◆ USER_CATALOG
 - ◆ USER_OBJECTS
 - ◆ USER_CONSTRAINTS
 - ◆ USER_CONS_COLUMNS

USER_CATALOG

- ◆ utilizator. Are doar două coloane, prima conținând numele și al doilea tipul obiectului. O parte a conținutului acestei tabele, obținut cu cererea SQL:

```
SELECT * FROM USER_CATALOG;
```

- ◆ este următorul:

TABLE_NAME	TABLE_TYPE
-----	-----
BURSA	TABLE
EVENIMENT	TABLE
NUMERE	TABLE
SPEC	TABLE
STUD	TABLE

USER_OBJECTS

- ◆ Conține numele, tipul și alte informații despre obiectele deținute de utilizator, inclusiv indecși.
- ◆ Informații sumare despre conținutul vederii se pot obține cu cererea:

```
SELECT OBJECT_NAME, OBJECT_TYPE  
FROM USER_OBJECTS;
```
- ◆ Rezultatul include rezultatul obținut în urma cererii anterioare (USER_CATALOG).

USER_CONSTRAINTS

- ◆ Conține date despre constrângerile definite de utilizator. Dintre coloanele sale menționăm:
 - ◆ **CONSTRAINT_NAME** : Numele constrângerii
 - ◆ **CONSTRAINT_TYPE**: Tipul acesteia, codificat pe un caracter: P: cheie primară, U: cheie (UNIQUE), R pentru integritatea referențială și C pentru constrângerile CHECK (aici sunt incluse și cele de tip NOT NULL)
 - ◆ **SEARCH_CONDITION**: Condiția asociată constrângerii

EXEMPLU

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE,  
       SEARCH_CONDITION  
FROM USER_CONSTRAINTS;
```

Rezultat:

CONSTRAINT_NAME	C	SEARCH_CONDITION
-----	-	-----
BURSA_CK	C	PMIN < PMAX AND PMIN >= 0 AND PMAX >= 0 AND SUMA BETWEEN 0 AND 500
DOM_NENUL	U	
DOM_5	C	LENGTH(NUME) >5
NUME_NENUL	C	NUME IS NOT NULL

USER_CONS_COLUMNS

- ◆ Conține informații despre coloanele care sunt implicate în constrângerile de integritate. Cererea următoare afișează perechi (nume constrângere, nume coloană implicată) pentru constrângerea BURSA_CK:

```
SELECT CONSTRAINT_NAME, COLUMN_NAME
FROM USER_CONS_COLUMNS
WHERE CONSTRAINT_NAME = 'BURSA_CK';
```

- ◆ Rezultatul este:

CONSTRAINT_NAME	COLUMN_NAME
BURSA_CK	PMIN
BURSA_CK	PMAX
BURSA_CK	SUMA

CATE VEDERI SUNT?

- ◆ Există peste **2000 de vederi** puse la dispoziție de sistemul Oracle (V9) dintre care peste 200 în fiecare dintre categoriile USER_, ALL_ și DBA_.
- ◆ O descriere completă a lor se găsește în documentația aferentă sistemului.
- ◆ Lista tuturor vederilor poate fi obținută cu cererea:

```
SELECT VIEW_NAME FROM ALL_VIEWS;
```

Bibliografie

1. **Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer D. Widom:** *Database Systems: The Complete Book*, Prentice-Hall, Englewood Cliffs, NJ, 2002.
2. **F. Rădulescu :** *Oracle SQL, PL/SQL*, Editura Printech, ISBN 973-718-203-02005