11. Crearea şi modificarea structurii tabelelor. Constrângeri.

- Cum se creează o tabelă
- Cum se pot adăuga / şterge / modifica coloanele unei tabele
- Ce sunt constrângerile
- Cum se pot defini constrângerile la nivel de coloană
- Cum se pot defini constrângerile la nivel de tabelă
- Cum se pot dezactiva / reactiva / şterge constrângerile

- După etapa de modelare a bazelor de date, primul pas în realizarea unei aplicaţii de baze de date constă în crearea obiectelor ce compun baza de date: **tabele, indexi, vederi, sinonime** etc.
- Crearea tabelelor, presupune **stabilirea numelor** tabelelor şi a **coloanelor** ce le compun, stabilirea **tipurilor** de date pe care le au coloanele tabelei, dar şi declararea **restricţiilor** (constrângerilor) care asigură ***integritatea şi coerenţa*** informaţiilor din baza de date.

CREAREA TABELELOR

- Pentru crearea unei tabele se foloseşte comanda **CREATE TABLE**.
- Cea mai simplă formă a acestei comenzi, în care nu se definesc valori implicite pentru coloane şi nu definim nici o restricţie este:

  CREATE TABLE *numetabel*

  (*coloana1 tip1*,

  ***coloana2 tip2*,**

  …

  ***coloanan tipn*)**
- unde - ***numetabel*** este numele atribuit tabelului nou creat. Acest nume trebuie să respecte restricţiile privind definirea numelor
- - ***coloana1, coloana2, …, coloanan*** sunt numele coloanelor din tabela nou creată
- - ***tip1, tip2, …, tipn*** reprezintă tipul datelor ce vor fi reţinute în coloanele tabelei nou create şi dimensiunea (dacă este cazul).
- Pe lângă numele tipului respectiv se precizează în paranteză lungimea tipului, respectiv numărul de caractere pentru un şir de caractere, sau numărul total de cifre şi numărul de cifre de după virgulă pentru valorile numerice.

- De exemplu, pentru crearea tabelei corespunzătoare entităţii **Jucător** folosim comanda:

```
CREATE TABLE jucatori    (
  nr_legitimatie NUMBER(3),
  nume VARCHAR2(30),  prenume VARCHAR2(30),
  data_nasterii DATE, adresa VARCHAR2(50),
  telefon CHAR(13),   email VARCHAR2(30),
  cod_echipa NUMBER(3)  )
```

- Deocamdată nu am definit cheia primară şi cheia străină.

- Pentru crearea tabelei **ECHIPE** folosim comanda:

```
CREATE TABLE jucatori (
  cod NUMBER(3),
  nume VARCHAR2(30), localitate VARCHAR2(30),
  adresa_club VARCHAR2(50) )
```

- Încă un exemplu:

```
CREATE TABLE elevi (
  id NUMBER(5),
        nume VARCHAR2(30), prenume VARCHAR2(30),
        bursier CHAR(1), media NUMBER(4,2) )
```

- În acest exemplu, pentru tipul câmpului media s-au precizat două valori. Prima (**4**) reprezintă numărul total de cifre ale numărului, iar al doilea număr reprezintă numărul de cifre zecimale (**2**).
- Dacă sunt introduse mai mult de două zecimale se va face rotunjire la două zecimale.
- La partea întreagă pot exista două cifre. Dacă numărul introdus are mai mult de două cifre la partea întreagă se va semnala o eroare.
- De asemenea, am declarat un câmp **bursier**, care ne va ajuta să memorăm dacă un elev este sau nu bursier. Însă, în Oracle nu există tipul logic (sau boolean), motiv pentru care am optat pentru tipul **CHAR(1)**, pentru un elev bursier vom memora în acest câmp valoarea **'D'**, pentru ceilalţi elevi acest câmp rămânând necompletat.

- O altă metodă de creare a unei tabele defineşte structura pe baza structurii unei tabele deja existente şi în acelaşi timp copiază datele din tabela deja existentă.
- Datele care se copiază din tabela deja existentă (liniile dar şi coloanele ce se copiază) se precizează prin clauza **AS** urmată de o subinterogare.
- De exemplu comanda următoare creează tabela **bursieri** pe baza tabelei **elevi** deja existentă:

```
CREATE TABLE bursieri
AS   SELECT id, nume, prenume FROM elevi
     WHERE bursier='D'
```

- Se observă că nu sunt copiate coloanele **media** şi **bursier** din tabela **elevi**.

Definirea valorilor implicite pentru coloane

- Sintaxa comenzii **CREATE TABLE** prezentată anterior este una mult simplificată.
- În cadrul acestei comenzi putem utiliza clauza **DEFAULT** pentru a defini o valoare implicită pentru o coloană a tabelei. Această clauză precizează ce valoare va lua un atribut atunci când, la inserarea unei linii în tabelă, nu se specifică în mod explicit valoarea atributului respectiv. Clauza **DEFAULT** apare după precizarea tipului coloanei şi este urmată de constanta care defineşte valoarea implicită:

```
CREATE TABLE angajati
( nume varchar2(30), prenume varchar2(30),
  adresa varchar2(50) DEFAULT 'Necunoscuta',
  localitate varchar2(20) DEFAULT 'Bucuresti',
  data_ang date DEFAULT SYSDATE,
  salar NUMBER(5) DEFAULT 800 )
```

- După cum se vede în exemplul anterior valoarea implicită poate fi o constantă dar poate fi de asemenea o expresie, sau o una din funcţiile speciale **SYSDATE** şi **USER** (care returnează numele utilizatorului curent) dar nu poate fi numele altei coloane sau al unei funcţii definite de utilizator.
- Pentru o coloană pentru care nu s-a definit o valoare implicită, şi nu face parte din cheia primară sau dintr-o restricţie **NOT NULL** sau **UNIQUE**, sistemul va considera ca valoare implicită valoarea **NULL**.

Definirea constrângerilor

- Orice bază de date trebuie să stabilească regulile de integritate care să garanteze că datele introduse în baza de date sunt corecte şi valide.
- Aceasta înseamnă că dacă există o regulă sau restricţie asupra unei entităţi, atunci datele introduse în baza de date respectă aceste restricţii.
- Regulile de integritate se definesc la crearea tabelelor folosind **constrângerile**. Constrângerile pot fi clasificate în:
  - constrângeri de domeniu, care definesc valorile pe care le poate lua un atribut (**NOT NULL**, **UNIQUE**, **CHECK**)
  - constrângeri de integritate a tabelei, precizând cheia primară a acesteia
  - constrângeri de integritate referenţială, care asigură coerenţa între cheile primare (sau unice) şi cheile străine corespunzătoare (**FOREIGN KEY**)

- Pe de altă parte constrângerile se pot clasifica după nivelul la care sunt definite în:
  - contrângeri la nivel de tabelă care pot acţiona asupra unei combinaţii de coloane
  - constrângeri la nivel de coloană.
- Constrângerile **NOT NULL** se pot defini doar la nivel de coloană.
- Constrângerile **UNIQUE**, **PRIMARY KEY**, **FOREIGN KEY** şi **CHECK** pot fi definite atât la nivel de coloană cât şi la nivel de tabelă. Totuşi dacă aceste constrângeri implică mai multe coloane atunci trebuie să fie definite obligatoriu la nivel de tabelă.
- Dacă o restricţie se defineşte la nivel de coloană se va folosi sintaxa:

nume_coloana tip_data tip_constr

sau

nume_coloana tip_data CONSTRAINT nume_constr tip_constr

- La nivel de tabelă folosim sintaxa

tip_constr

sau

CONSTRAINT nume_constr tip_constr

- Se observă că putem decide să dăm un nume explicit unei constrângeri, ceea ce uşurează referirea ulterioară la acea constrângere, sau putem să nu definim un nume explicit, caz în care sistemul va genera un nume implicit.
- Dacă se foloseşte cuvântul **CONSTRAINT**, atunci obligatoriu acesta va fi urmat de numele dat explicit constrângerii.

RESTRICŢIA NOT NULL

- Modul de definire al fiecăreia dintre aceste constrângeri:
- **NULL** este o valoare specială. Necompletarea în tabelă a unei celule conduce la completarea ei cu valoarea **NULL**, semnificând faptul că celula respectivă are de fapt o valoare nedefinită.
- Într-un ERD, un atribut poate fi obligatoriu, lucru pe care îl marcam cu o steluţă în faţa atributului respectiv. În baza de date această condiţie se traduce prin faptul că valoarea coloanei respective trebuie obligatoriu completată, adică nu poate conţine valoarea **NULL**.
- Pentru definirea acestui tip de restricţii folosim restricţia **NOT NULL** pentru coloana respectivă, fie la crearea tabelei fie mai târziu la modificarea structurii acesteia.
- La crearea tabelei, restricţia **NOT NULL** se precizează pentru fiecare coloană ce trebuie să respecte această restricţie, după precizarea tipului coloanei respective astfel:

```
CREATE TABLE angajati
( nume varchar2(30) NOT NULL,
  prenume varchar2(30),
  localitate varchar2(20) DEFAULT 'Iasi' NOT NULL
  ... )
```

- Restricţia **NOT NULL** a putut fi folosită în combinaţie cu clauza **DEFAULT**.

RESTRICŢIILE PRIMARY KEY ŞI UNIQUE

- Cheia primară este o coloană sau o combinaţie de coloane care identifică în mod unic liniile unei tabele.
- Coloanele care fac parte din cheia primară vor fi automat de tip **NOT NULL** fără ca acest lucru să mai trebuiască precizat explicit.
- Când cheia primară este compusă dintr-o singură coloană, definirea acesteia se poate face la nivel de coloană ca în exemplul următor:

```
CREATE TABLE angajati
( cnp number(13) PRIMARY KEY
  nume varchar2(30),
  ... )
```

- Dacă dorim să atribuim un nume constrângerii putem scrie

```
CREATE TABLE angajati
( cnp number(13) CONSTRAINT angajati_pk
    PRIMARY KEY
  nume varchar2(30),
  ... )
```

- Definirea cheii primare la nivel de tabelă se poate face şi atunci când cheia este compusă dintr-un singur câmp, dar este obligatorie atunci când este compusă din mai multe coloane.
- De exemplu tabela **carti** are cheia primară compusă din combinaţia coloanelor **titlu**, **autor**, **data_aparitie**.
- Comanda de creare a acestei tabele se poate scrie:

```
CREATE TABLE carti
( titlu VARCHAR2(30),
  autor VARHAR2(30),
  data_ap DATE,
  format VARCHAR2(10),
  nr_pag NUMBER(3),
  CONSTRAINT carti_pk
PRIMARY KEY (titlu, autor, data_ap)
)
```

- sau simplu

```
CREATE TABLE carti
( titlu VARCHAR2(30),
  autor VARCHAR2(30),
  data_ap DATE,
  format VARCHAR2(10),
  nr_pag NUMBER(3),
  PRIMARY KEY (titlu, autor, data_ap)
 )
```

- Sintaxa generală de definire a cheii primare este deci **PRIMARY KEY (lista_coloane)**
- Similar se poate defini şi restricţia **UNIQUE** care precizează că valoare coloanei definită ca **UNIQUE**, sau combinaţia valorilor coloanelor ce definesc restricţia **UNIQUE** trebuie să fie unică pentru toate liniile din tabelă.
- Cu alte cuvinte, într-o coloană definită ca **UNIQUE** nu pot exista valori duplicate.
- ***Atenţie!*** Coloanele definite ca **UNIQUE** pot conţine valori **NULL**, iar acestea pot fi oricâte, adică valoare **NULL** este singura valoare ce poate fi duplicată într-o coloană **UNIQUE**.

Exemple:

```
CREATE TABLE elevi
( nr_matr NUMBER(5) PRIMARY
    KEY,
  cnp NUMBER(13) UNIQUE,
  nume VARCHAR2(30),
  prenume VARHAR2(30)
 )
```

```
CREATE TABLE elevi
( nr_matr NUMBER(5) PRIMARY
    KEY,
  cnp NUMBER(13)
    CONSTRAINT cnp_uk
    UNIQUE,
  nume VARCHAR2(30),
  prenume VARHAR2(30)
 )
```

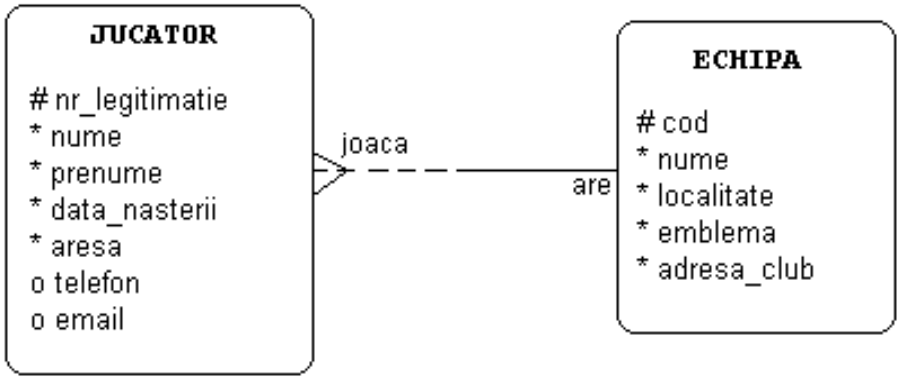
Exemple:

```
CREATE TABLE carti
( ISBN varchar2(20) PRIMARY KEY,
  titlu VARCHAR2(30),
  autor VARCHAR2(30),
  data_ap DATE,
  format VARCHAR2(10),
  nr_pag NUMBER(3),
  UNIQUE (titlu, autor, data_ap)
)
```

```
CREATE TABLE carti
( ISBN varchar2(20) PRIMARY KEY,
  titlu VARCHAR2(30),
  autor VARCHAR2(30),
  data_ap DATE,
  format VARCHAR2(10),
  nr_pag NUMBER(3),
  CONSTRAINT carti_uk UNIQUE (titlu, autor, data_ap)
 )
```

RESTRICŢIA FOREIGN KEY

- Restricţiile referenţiale sunt categoria de restricţii care creează cele mai mari probleme în gestiunea bazelor de date.
- Pentru exemplificarea modului de definire a chei străine vom relua un exemplu de ERD prezentat anterior



- Crearea tabelei **jucatori** corespunzătoare entităţii **JUCATOR** din acest ERD se va scrie:

```
CREATE TABLE jucatori
( nr_legitimatie NUMBER(5) PRIMARY KEY,
  cod_echipa NUMBER(3)
        REFERENCES echipe(cod)
  nume VARCHAR2(30) NOT NULL,
  prenume VARCHAR2(30) NOT NULL,
  datan DATE NOT NULL,
  adresa VARCHAR2(60) NOT NULL,
  telefon NUMBER(3),
  email VARCHAR2(30)
        )
```

- Sau:

```
CREATE TABLE jucatori
( nr_legitimatie NUMBER(5) PRIMARY KEY,
  cod_echipa NUMBER(3) CONSTRAINT ech_fk
        REFERENCES echipe(cod),
  nume VARCHAR2(30) NOT NULL,
  prenume VARCHAR2(30) NOT NULL,
  datan DATE NOT NULL,
  adresa VARCHAR2(60) NOT NULL,
  telefon NUMBER(3),
  email VARCHAR2(30)
 )
```

- Sau la nivel de tabela:

```
CREATE TABLE jucatori
( nr_legitimatie NUMBER(5) PRIMARY KEY,
  cod_echipa NUMBER(3),
  nume VARCHAR2(30) NOT NULL,
  prenume VARCHAR2(30) NOT NULL,
  datan DATE NOT NULL,
  adresa VARCHAR2(60) NOT NULL,
  telefon NUMBER(3),
  email VARCHAR2(30),
  FOREIGN KEY (cod_echipa)
     REFERENCES echipe(cod)
)
```

- sau
```
CREATE TABLE jucatori
( nr_legitimatie NUMBER(5) PRIMARY KEY,
  cod_echipa NUMBER(3),
  nume VARCHAR2(30) NOT NULL,
  prenume VARCHAR2(30) NOT NULL,
  datan DATE NOT NULL,
  adresa VARCHAR2(60) NOT NULL,
  telefon NUMBER(3),
  email VARCHAR2(30),
  CONSTRAINT test_fk FOREIGN KEY (cod_echipa)
       REFERENCES echipe(cod)
)
```

- Sintaxa generală este aşadar la nivel de tabelă:

[CONSTRAINT nume_const] FOREIGN KEY (lista_coloane)

REFERENCES tabela_parinte(lista_coloane_referite)

- iar la nivel de coloană

[CONSTRAINT nume_const]

REFERENCES tabela_parinte(lista_coloane_referite)

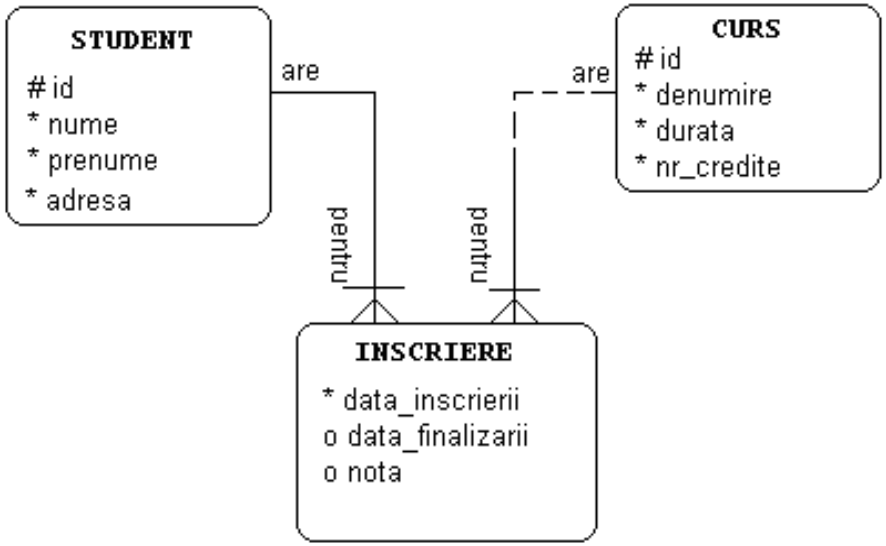
- La definirea unei chei străine se poate utiliza o clauză suplimentară **ON DELETE CASCADE** care precizează că la ştergerea unei linii din tabela părinte se vor şterge automat din tabela copil acele linii care fac referire la linia ce se şterge din tabela părinte. De exemplu, prin folosirea acestei opţiuni, la ştergerea unei echipe se vor şterge automat toţi jucătorii de la acea echipă.
- Această clauză se foloseşte astfel:

```
CREATE TABLE jucatori
( nr_legitimatie NUMBER(5) PRIMARY
    KEY,
  cod_echipa NUMBER(3)
    CONSTRAINT ech_fk
       REFERENCES echipe(cod) ON
    DELETE CASCADE,
  nume VARCHAR2(30) NOT NULL,
  prenume VARCHAR2(30) NOT NULL,
  datan DATE NOT NULL,
  adresa VARCHAR2(60) NOT NULL,
  telefon NUMBER(3),
  email VARCHAR2(30)
        )
```

- O altă opţiune este **ON DELETE SET NULL** care face ca la ştergerea unui părinte, valorile cheii străine din liniile tabelei copil care fac referire la linia ştearsă vor fi setate pe **NULL**.
- De exemplu la ştergerea unei echipe, jucătorii acesteia vor deveni liberi de contract, deci codul echipei la care joaca va fi setat pe **NULL**:

```
CREATE TABLE jucatori
( nr_legitimatie NUMBER(5) PRIMARY KEY,
  cod_echipa NUMBER(3) CONSTRAINT ech_fk
      REFERENCES echipe(cod) ON DELETE SET NULL,
  nume VARCHAR2(30) NOT NULL,
  prenume VARCHAR2(30) NOT NULL,
  datan DATE NOT NULL,
  adresa VARCHAR2(60) NOT NULL,
  telefon NUMBER(3),
  email VARCHAR2(30)
        )
```

- Implicit, fără precizarea uneia din aceste două opţiuni, Oracle va interzice ştergerea unei linii din tabela părinte atâta timp cât mai există măcar o linie în tabela copil care face referire la ea.
- Să vedem acum cum creăm tabela **inscrieri** corespunzătoare entităţii **inscriere** din figura alăturată.
- Observăm că în cheia primară intră şi coloanele ce fac parte din cheia străină.



```
CREATE TABLE inscriere (
    id_student NUMBER(5) NOT NULL
        REFERENCES studenti(id),
    id_curs NUMBER(5) NOT NULL REFERENCES
  cursuri(id),
    data_inscrierii DATE DEFAULT sysdate NOT
  NULL,
    data_finalizarii DATE,
    nota NUMBER (4,2),
    PRIMARY KEY (id_student, id_curs,
  data_inscrierii)
   )
```

RESTRICŢIA CHECK

- Acest tip de constrângeri specifică o condiţie ce trebuie să fie îndeplinită de datele introduse în coloana (sau coloanele) asupra căreia acţionează. O astfel de constrângere poate limita valorile care pot fi introduse în cadrul unei coloane.
- Iată câteva exemple de reguli de validare pentru tabela elevi care pot fi implementate cu ajutorul constrângerilor de tip **CHECK**:
  - numele şi prenumele unui elev trebuie să înceapă cu o majusculă restul literelor fiind litere mici
  - nota unui elev nu poate fi mai mare de **10**
  - câmpul bursier poate avea doar valorile **'D'** şi **NULL**
  - numărul de absenţe nemotivate va fi cel mult egal cu numărul total de absenţe

- Crearea tabelei elevi în această situaţie se poate scrie astfel:

```
CREATE TABLE elevi
( nr_matr NUMBER(5) PRIMARY KEY,
  cnp NUMBER(13) CONSTRAINT cnp_uk UNIQUE,
  nume VARCHAR2(30) NOT NULL
       CHECK nume=LTRIM(INITCAP(nume)),
  prenume VARHAR2(30) NOT NULL
       CHECK nume=LTRIM(INITCAP(nume))
  bursier CHAR(1) CHECK bursier='D',
  nota NUMBER(4,2)
    CONSTRAINT nota_ck CHECK nota<=10
  total_abs NUMBER(3),
  abs_nemotiv NUMBER(3),
  CHECK (abs_nemotiv<=total_abs)
     )
```

Modificarea structurii unei tabele

- Modificarea structurii unui tabel se realizează cu ajutorul comenzii ALTER TABLE, permiţând adăugarea sau ştergerea unei coloane, modificarea definiţiei unei coloane, crearea unei noi constrângeri sau ştergerea unor constrângeri existente.
- Vom prezenta în continuare, pe scurt, fiecare dintre aceste operaţii.

Adăugarea unei noi coloane

- Se realizează folosind clauza **ADD** a comenzii **ALTER TABLE**. Sintaxa este similară cu cea a creării unei coloane în cadrul comenzii **CREATE TABLE**. De exemplu comenda următoare adaugă o colonă **nrgoluri** la tabela **jucatori**:

ALTER TABLE jucatori

ADD nrgoluri NUMBER(4)

- Coloana nou creată va deveni ultima coloană a tabelei. Dacă tabela conţine deja date, coloana adăugată va fi completată cu **NULL** în toate liniile existente. De aceea nu vom putea adăuga o coloană cu restricţia **NOT NULL** la o tabelă ce conţine deja date.
- Aşadar o comandă de forma:

ALTER TABLE test ADD ex NUMBER(3) NOT NULL

- sau

ALTER TABLE test ADD ex NUMBER(3) PRIMARY KEY

- Sunt permise doar dacă tabela nu conţine deja date.
- Însă comanda

ALTER TABLE test ADD ex NUMBER(3) UNIQUE

- poate fi folosită în orice moment, deoarece după cum am precizat o coloană **UNIQUE** poate conţine oricâte valori **NULL**.

ŞTERGEREA UNEI COLOANE

- Se realizează folosind clauza **DROP COLUMN** a comenzii **ALTER TABLE**:

ALTER TABLE elevi DROP COLUMN bursier

- Aşa cum este şi normal, ştergerea unei coloane duce automat şi la ştergerea restricţiilor definite pentru aceasta şi care nu implică şi alte coloane.
- De exemplu dacă tabela **elevi** a fost creată cu ajutorul comenzii de la pagina 58, putem şterge fără probleme coloana **nume**:

ALTER TABLE elevi DROP COLUMN nume

- chiar dacă avem definită o restricţie de tip **CHECK** la nivelul acestei coloane. De asemenea putem şterge coloana **nr_matr**, chiar dacă aceasta este cheia primară a tabelei:

ALTER TABLE elevi DROP COLUMN nr_matr

- însă se va genera o eroare dacă încercăm să ştergem coloana **abs_nemotiv**, din cauza restricţiei definită la nivel de tabelă şi care implică coloanele **abs_nemotiv** şi **total_abs**.
- O variantă ar fi să ştergem mai întâi toate restricţiile în care apare coloana ce dorim să o ştergem, sau să folosim clauza **CASCADE CONSTRAINTS** astfel:

ALTER TABLE elevi DROP COLUMN abs_nemotiv

CASCADE CONSTRAINTS

Modificarea unei coloane

- Poate fi făcută cu clauza **MODIFY** ca în exemplul următor:

ATLER TABLE elevi MODIFY prenume VARCHAR2(50)

- Prin care am modificat tipul coloanei **prenume** de le **VARCHAR2(30)** la **VARCHAR2(50)**, deoarece am descoperit la un moment dat că există elevi al căror prenume (compus) are mai mult de 30 de caractere.
- Mărirea numărului de caractere pentru o coloană de tip şir de caractere se poate face fără nici o problemă, însă micşorarea acestei dimensiuni se poate face doar dacă tabela este goală, sau coloana respectivă conţine doar valori **NULL**.
- Tot cu opţiunea **MODIFY** se poate modifica, sau se poate stabili o valoare implicită, dacă nu exista deja una astfel:

ALTER TABLE elevi MODIFY bursier CHAR(1)

DEFAULT 'D'

- însă această valoare implicită nu va afecta liniile deja existente în tabelă, ci doar liniile ce vor fi introduse în continuare.

Adăugarea unei constrângeri

- Sintaxa comenzii pentru adăugarea unei constrângeri la nivel de tabelă este:

ALTER TABLE *nume_tabela*

ADD CONSTRAINT *nume_constr definitie_constr*

- sau

ALTER TABLE *nume_tabela*

ADD *definitie_constr*

- De exemplu comanda următoare defineşte cheia primară pentru o tabelă fictivă

ALTER TABLE tabelaexemplu

ADD PRIMARY KEY (coloana1)

- Această comandă poate fi scrisă echivalent şi

ALTER TABLE tabelaexemplu

ADD CONSTRAINT tabelaexemplu_pk PRIMARY KEY (coloana1)

- Singura constrângere ce nu poate fi adăugată în acest fel este NOT NULL, care poate fi adăugată doar prin modificarea coloanei restective folosind MODIFY:

ALTER TABLE tabelaexemplu

MODIFY coloana2 VARCHAR2(20) NOT NULL

Ştergerea unei constrângeri

- Ştergerea unei constângeri se face folosind opţiunea DROP CONSTRAINT astfel:

ALTER TABLE *nume_tabela*

DROP CONSTRAINT *nume_constrangere*

- sau

ALTER TABLE *nume_tabela* DROP PRIMARY KEY

- sau

ALTER TABLE *nume_tabela*

DROP UNIQUE(*lista_coloane*)

Activarea/dezactivarea unei constrângeri

- În unele situaţii, este necesară o dezactivare temporară şi apoi reactivarea unei constrângeri. Acest lucru se realizează astfel:

ALTER TABLE *nume_tabela* DISABLE/ENABLE

CONSTRAINT *nume_constrangere* [CASCADE]

- Sau

ALTER TABLE *nume_tabela* DISABLE/ENABLE

PRIMARY KEY [CASCADE]

- sau

ALTER TABLE *nume_tabela* DISABLE/ENABLE

UNIQUE (coloana1,coloana2,…) [CASCADE]

- Clauza **CASCADE** precizează că şi constrângerile dependente sunt de asemenea afectate.

În această lecție am învățat despre:

- Cum se creează o tabelă
- Cum se pot adăuga / şterge / modifica coloanele unei tabele
- Ce sunt constrângerile
- Cum se pot defini constrângerile la nivel de coloană
- Cum se pot defini constrângerile la nivel de tabelă
- Cum se pot dezactiva / reactiva / şterge constrângerile

SFÂRȘIT