Baze de date

Curs 2

Modelul relațional de organizare a bazelor de date

Modelul relațional de organizare a bazelor de date

- propus de Edgar Codd în 1970, 1981 – Turing Award
- modelul de date dominant azi
- reprezentat de SGBD-uri de top: DB2, Oracle, SQL Server etc
- avantaje: **reprezentarea simplă a datelor** & ușurința cu care pot fi exprimate interogările (chiar și cele mai complexe) pe date utilizând un **limbaj simplu, de nivel înalt**
- permite și utilizatorilor neinițiați să înțeleagă conținutul unei baze de date

Modelul relațional de organizare a bazelor de date

1. relația
2. restricțiile de integritate
3. bazele de date relaționale
4. gestiunea bazelor de date relaționale

1. Relația

- într-o aplicație, o **colecție de date** se folosește conform unui **model de organizare**
- în **modelul relațional** colecția de date este organizată ca o mulțime de **relații** (tabele)
- o relație cuprinde o **schemă** a relației și o **instanță** a relației
- instanța relației este un tabel, iar schema relației descrie capetele de coloană ale tabelului

1. Relația

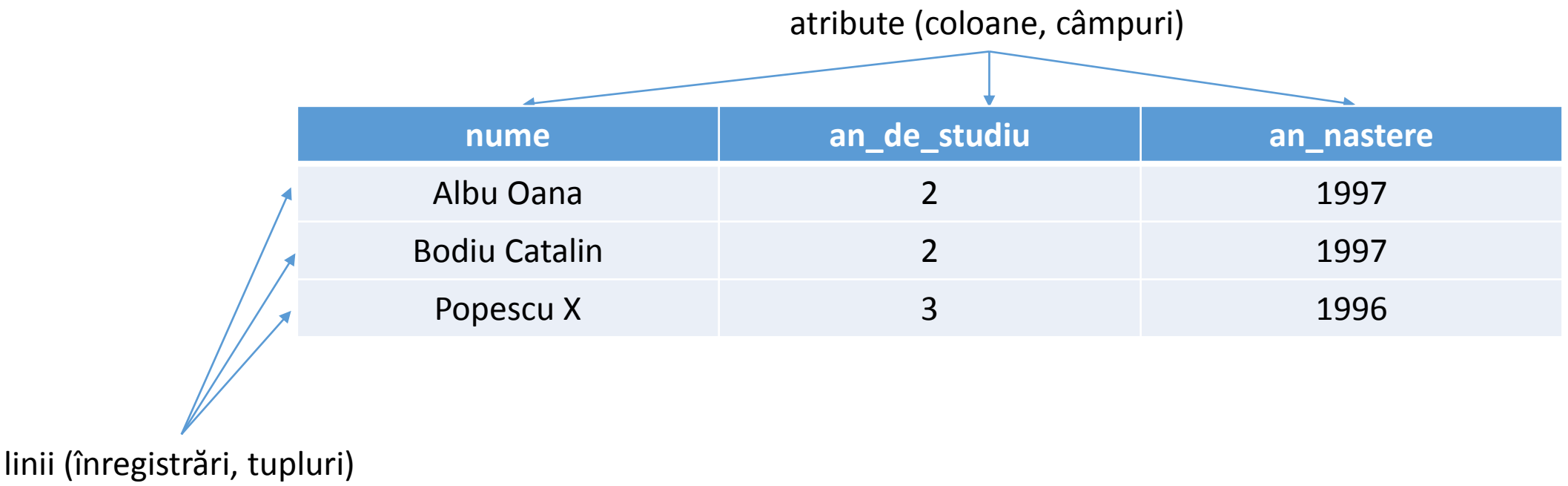
- schema precizează numele relației și numele și domeniul fiecărui câmp
- domeniul unui câmp este precizat printr-un nume de domeniu și are o mulțime de valori asociate
- exemplu de schemă pentru relația Studenti:

  Studenti[*nume*, *an_de_studiu*, *an_nastere*]
  sau notația:

  Studenti(*nume*: string, *an_de_studiu*: integer, *an_nastere*: integer)
- domeniul câmpului *nume* se numește *string*, iar mulțimea de valori asociate acestui domeniu este mulțimea tuturor șirurilor de caractere

1. Relația

- exemplu de instanță pentru relația *Studenti*:



nume	an_de_studiu	an_nastere
Albu Oana	2	1997
Bodiu Catalin	2	1997
Popescu X	3	1996

1. Relația

- **A_1, A_2, ..., A_n** o mulțime de atribute
- **D_i = Dom(A_i)** U **{?} domeniul valorilor posibile** pentru **atributul A_i,** unde prin "**?**" s-a notat valoarea de "**nedefinit**" (**null**); valoarea de *nedefinit* se folosește pentru a verifica daca unui atribut i s-a atribuit o valoare sau dacă nu are valoare (sau are valoarea "nedefinit"); această valoare nu are un anumit tip de dată, se pot compara cu această valoare atribute de diferite tipuri (numerice, șiruri de caractere etc)
- **relație de gradul (aritatea) n**: $R \subseteq D_1 \times D_2 \times \ldots \times D_n$
- relația R poate fi considerată ca o mulțime de vectori cu câte n valori, câte o valoare pentru fiecare din atributele A_i
- **R[A_1, A_2, ... , A_n]** este **schema relației**

1. Relația

- relația se poate memora într-un tabel de forma:

R	A_1	...	A_j	...	A_n
r_1	a_{11}	...	a_{1j}	...	a_{1n}
...	...	...	...	...	...
r_i	a_{i1}	...	a_{ij}	...	a_{in}
...	...	...	...	...	...
r_m	a_{m1}	...	a_{mj}	...	a_{mn}

unde $a_{ij} \in D_j$, $j = 1,...,n$, $i = 1,...,m$.

1. Relația

- valorile unui atribut sunt **atomice** (nu se pot descompune în valori pentru alte atribute) și **scalare** (nu se pot memora tablouri)
- liniile din tabel **nu sunt ordonate**
    - e.g., două reprezentări ale aceleiași instanțe a relației *Studenti*

nume	an_de_studiu	an_nastere
Albu Oana	2	1997
Bodiu Catalin	2	1997
Popescu X	3	1996

nume	an_de_studiu	an_nastere
Bodiu Catalin	2	1997
Albu Oana	2	1997
Popescu X	3	1996

- liniile din tabel **sunt distincte** - o relație este definită ca fiind o mulțime de tupluri distincte (în practică, sistemele comerciale permit tabelelor să conțină duplicate)

1. Relația

- **cardinalitatea** unei instanțe este numărul de tupluri pe care aceasta le conține

1. Relația

- exemplu de relație care nu e bine proiectată:

  Studenti[nume, sectie, an_de_studiu, grupa]

nume	sectie	an_de_studiu	grupa
Roman Ariadna	Info romana	2	226
Szabo Alexandru	Info romana	2	226
Roman Adrian	Info romana	2	227
Trifa Maria	Info romana	3	227

  - asocierea grupa=226 și (sectie='Info romana', an_de_studiu=2) se păstrează de două ori, pentru primii doi studenți
  - ultimii doi studenți sunt în aceeași grupă, dar în ani diferiți, ceea ce este o eroare
- pentru studiul unor astfel de situații este necesară **normalizarea relațiilor**

2. Restricțiile de integritate

- **constrângerile (restricțiile) de integritate** sunt **condiții specificate la definirea schemei bazei de date**; restricționează datele care pot fi stocate într-o instanță a bazei de date
- sunt **verificate când au loc modificări** asupra datelor, i.e., nu pot avea loc schimbări care violează restricțiile (e.g., introducerea unui student cu un CNP identic cu al unui student existent, ștergerea unui student pentru care mai există discipline stocate în contractul său de studiu, nerespectarea domeniului unui câmp la introducerea datelor etc)
- în anumite situații, SGBD nu va interzice modificările care nu respectă restricțiile, dar va opera anumite schimbări asupra datelor pentru a se asigura că baza de date satisface toate restricțiile specificate
- exemple de constrângeri: constrângeri de **domeniu**, constrângeri de **unicitate**, constrângeri de **integritate referențială**

2. Restricțiile de integritate

- constrângeri de **domeniu** – reprezintă o condiție importantă care trebuie să fie satisfăcută de fiecare instanță de relație: valorile care apar într-o coloană trebuie să facă parte din domeniul asociat coloanei respective

2. Restricțiile de integritate

- constrângere (de tip) **cheie** – o constrângere conform căreia o **submulțime de atribute** din relație este un **identificator unic** pentru orice tuplu; această submulțime este minimală
- $K \subseteq \{A_1, A_2, ..., A_n\}$ este **cheie** pentru relația $R[A_1, A_2, ... , A_n]$ dacă atributele din K:
  - se pot folosi pentru identificarea fiecărei linii din R
  - nu există nicio submulțime din K cu această proprietate
- sistemele de gestiune a bazelor de date nu permit existența a două elemente distincte într-o relație cu aceeași valoare pentru **oricare cheie** (două înregistrări diferite nu pot avea valori identice în toate câmpurile unei chei), deci precizarea unei chei constituie o restricție pentru baza de date

2. Restricțiile de integritate

- e.g., în relația Medici[sectie, codm, nume, prenume] putem avea constrângerea: un medic este identificat în mod unic de secția din care face parte și de codul său, i.e., doi medici diferiți pot avea fie aceeași secție, fie același cod, dar nu amândouă; cheia este grupul de atribute **{sectie, codm}**
- e.g., Carti[autor, titlu, editura, an_aparitie]

autor	titlu	editura	an_aparitie
C. J. Date	An Introduction to Database Systems	Addison-Wesley Publishing Comp.	2004
P. G. Wodehouse	Greseala lordului Emsworth	Polirom	2003
Simon Singh	Marea teorema a lui Fermat	Humanitas	2012
Stephen Hawking	Scurta istorie a timpului	Humanitas	2012

- se poate alege ca și cheie grupul de attribute **{autor, titlu, editura, an_aparitie},** deci sunt necesare toate atributele relației; în astfel de situații este utilă includerea unui atribut suplimentar, care să aibă valori distincte (aceste valori se pot genera automat la adăugarea de înregistrări) ->

2. Restricțiile de integritate

codc	autor	titlu	editura	an_aparitie
457	C. J. Date	An Introduction to Database Systems	Addison-Wesley Publishing Comp.	2004
4	P. G. Wodehouse	Greseala lordului Emsworth	Polirom	2003
34	Simon Singh	Marea teorema a lui Fermat	Humanitas	2012
769	Stephen Hawking	Scurta istorie a timpului	Humanitas	2012

- acum se poate alege cheia **{codc}**
- în specificarea unei constrângeri cheie utilizatorul trebuie să fie sigur că această constrângere nu va împiedica introducerea unor seturi de tupluri "corecte" în contextul aplicației, e.g., dacă **{titlu}** se declară ca fiind cheie, relația *Carti* nu mai poate conține cărți (diferite!) care au același titlu
- mulțimea **{codc, titlu}** nu e o cheie, deoarece conține cheia **{codc}**; este o supercheie, i.e., o mulțime de câmpuri care conține cheia

2. Restricțiile de integritate

codc	autor	titlu	editura	an_aparitie
457	C. J. Date	An Introduction to Database Systems	Addison-Wesley Publishing Comp.	2004
4	P. G. Wodehouse	Greseala lordului Emsworth	Polirom	2003
34	Simon Singh	Marea teorema a lui Fermat	Humanitas	2012
769	Stephen Hawking	Scurta istorie a timpului	Humanitas	2012

- două tupluri au întotdeauna valori diferite pentru *codc*, cheie a relației; acest lucru nu e adevărat și pentru câmpurile non-cheie: relația poate conține două cărți diferite care au apărut în același an, i.e., două tupluri cu *an_aparitie = 2004*; sau trei cărți diferite scrise de același autor, i.e., trei tupluri cu *autor = Stephen Hawking* etc

2. Restricțiile de integritate

- pentru anumite relații pot fi stabilite mai multe chei; una dintre chei (un atribut sau un grup de atribute) se alege ***cheie principală (primară)***, iar celelalte se vor considera ***chei secundare (chei candidat)***

- e.g., Orar[zi, ora, sala, profesor, formatie, disciplina]

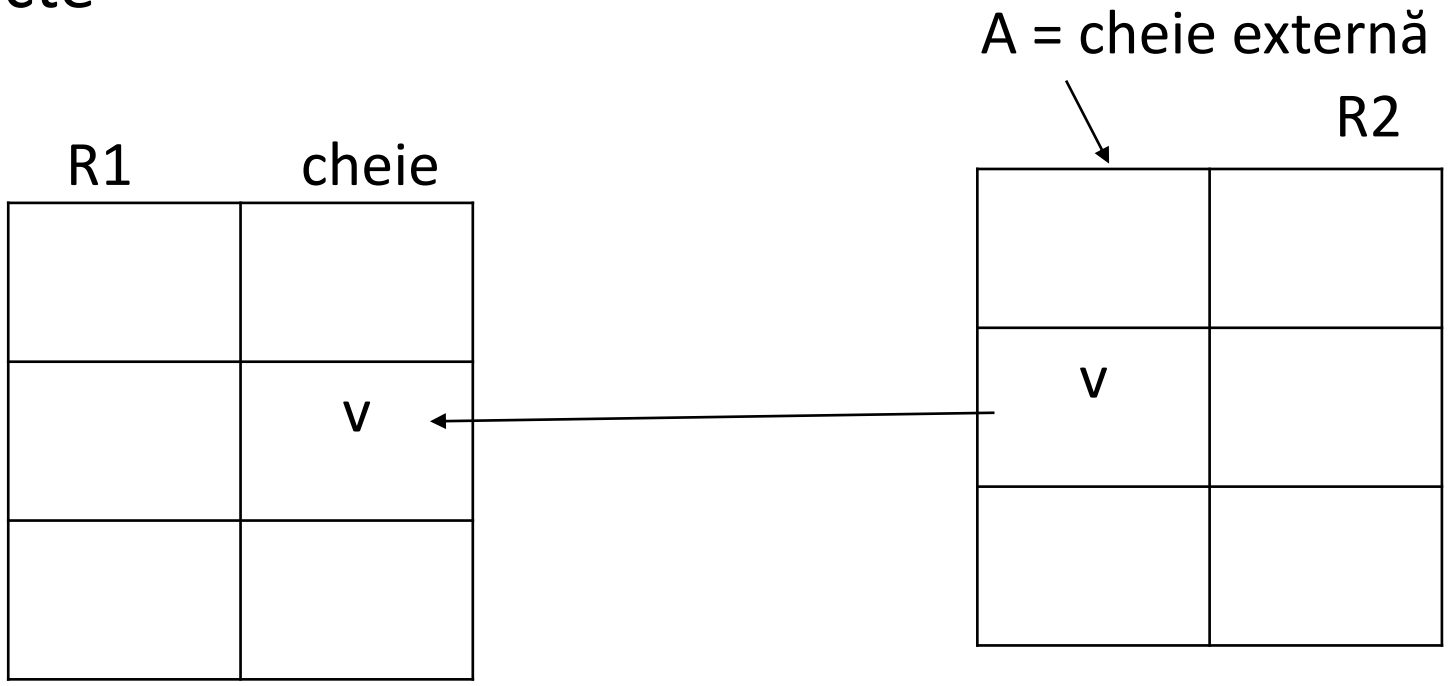
  cu orarul pe o săptămână

- se pot alege ca și chei următoarele mulțimi de atribute:

  {zi, ora, sala} ; {zi, ora, profesor} ; {zi, ora, formatie}

2. Restricțiile de integritate

- **(constrângeri) cheie externă –** valorile unor atribute dintr-o relație apar în altă relație; plecând de la o relație *R2* se pot căuta înregistrările dintr-o relație *R1* după valorile unui astfel de atribut (simplu sau compus); în relația *R2* se stabilește un atribut *A*, numit **cheie externă**; valorile atributului *A* se caută printre valorile cheii din relația *R1*; nu este obligatoriu ca cele două relații *R1* și *R2* să fie distincte

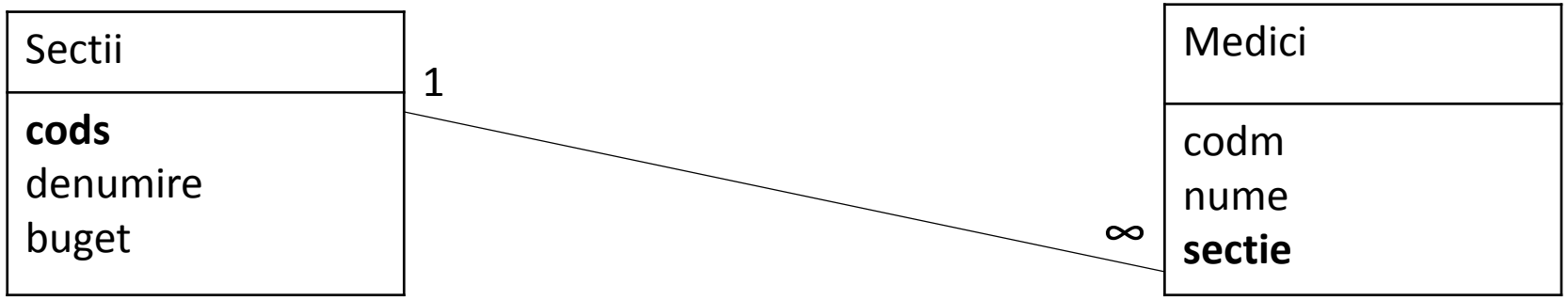


2. Restricțiile de integritate

- e.g.,

  Sectii[cods, denumire, buget]

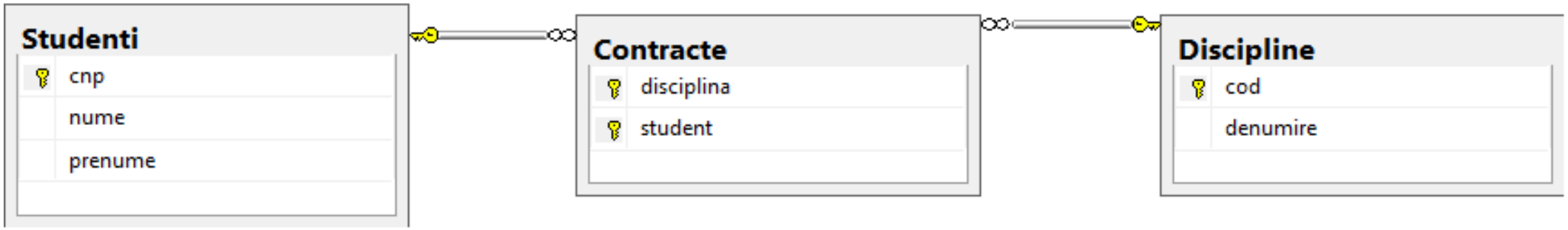
  Medici[codm, nume, sectie]



- legătura o putem stabili între relația *Sectii* (considerată ca părinte pentru relație) și relația *Medici* (ca membru pentru legătură) prin egalitatea **Sectii.cods = Medici.sectie**; unei anumite secții (memorată în relația *Sectii*), identificată printr-un cod, îi corespund toți medicii din secție cu codul respectiv; atributul *sectie* din relația *Medici* este cheie externă
- prin cheie externă se pot memora **legături 1:n** între entități: la o secție corespund oricâți medici, iar unui medic îi este asociată cel mult o secție

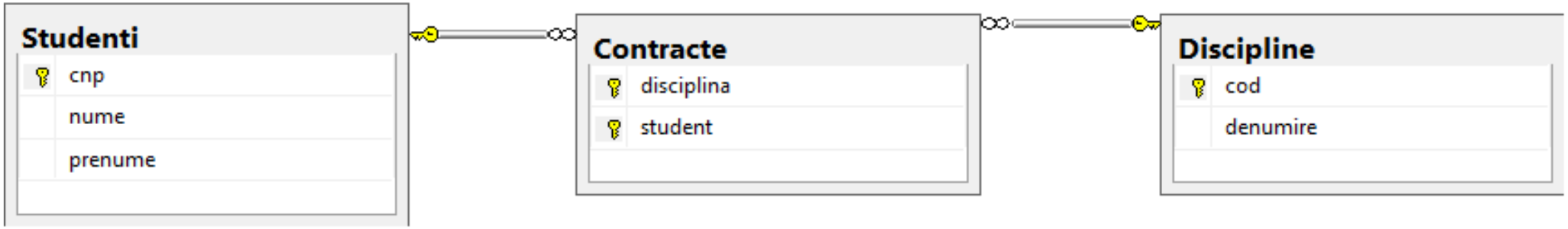
2. Restricțiile de integritate

- cheia externă se poate folosi și pentru a memora **legături m:n** între entități
- fie două entități: discipline și studenți; la o disciplină sunt „înscriși" mai mulți studenți, iar un student are asociate mai multe discipline; varianta de memorare cuprinde o relație intermediară



- orice valoare care apare în câmpul *student* al relației *Contracte* trebuie să apară de asemenea în câmpul *cnp* al relației *Studenti*; dar putem avea studenți pentru care nu s-a completat deocamdată contractul de studiu, i.e., pot exista valori în coloana *cnp* din *Studenti* care să nu apară în coloana *student* din *Contracte*

2. Restricțiile de integritate



- cheia externă din relația *Contracte* trebuie să aibă același număr de coloane cu cheia din *Studenti* la care se referă, preferabil, cheia primară; tipurile de date ale coloanelor trebuie să fie compatibile, dar numele acestora pot să difere

2. Restricțiile de integritate

respectarea restricțiilor de integritate

- când se efectuează operații asupra datelor, SGBD respinge modificările care încalcă restricțiile de integritate; în anumite situații, SGBD nu respinge comenzile, ci remediază problema operând anumite modificări asupra datelor
- exemple: introducerea în Studenti a unui tuplu al cărui cnp există deja în alt tuplu; introducerea unui tuplu în Contracte pentru care nu există un student corespunzător în Studenti; ștergerea unui tuplu din Studenti pentru care există înregistrări corespunzătoare în Contracte (se poate respinge operația sau se pot șterge toate tuplurile din Contracte referitoare la acel student)

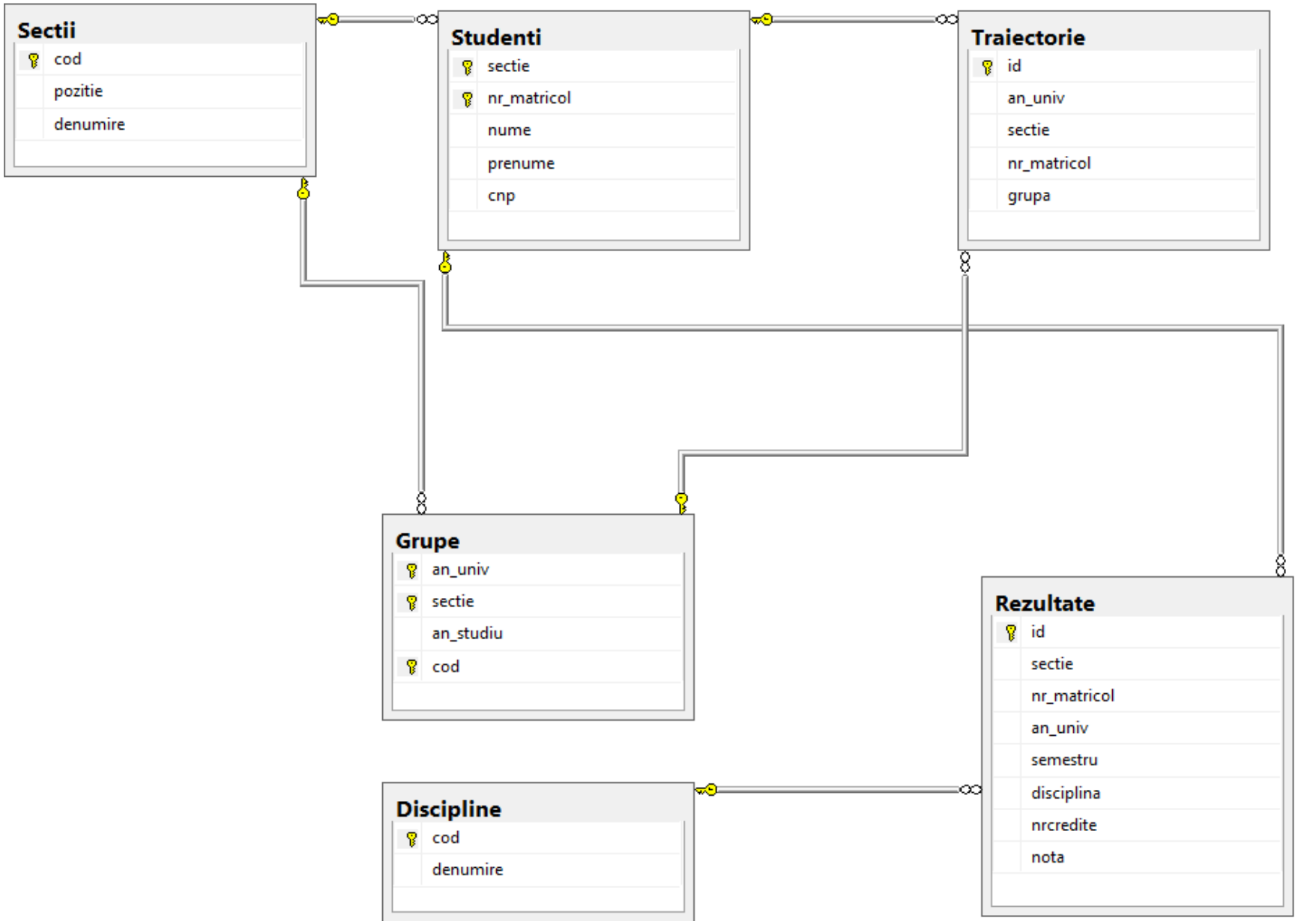
3. Bazele de date relaționale

- o **bază de date relațională** este o colecție de relații având nume distincte
- **schema bazei de date** relaționale este colecția schemelor pentru relațiile din baza de date
- o **instanță a unei baze de date** relaționale este o colecție de instanțe de relație, câte una pentru fiecare schemă relațională din schema bazei de date
- instanța unei baze de date trebuie să satisfacă toate restricțiile de integritate specificate la nivelul schemei bazei de date – instanță *legală*

3. Bazele de date relaționale

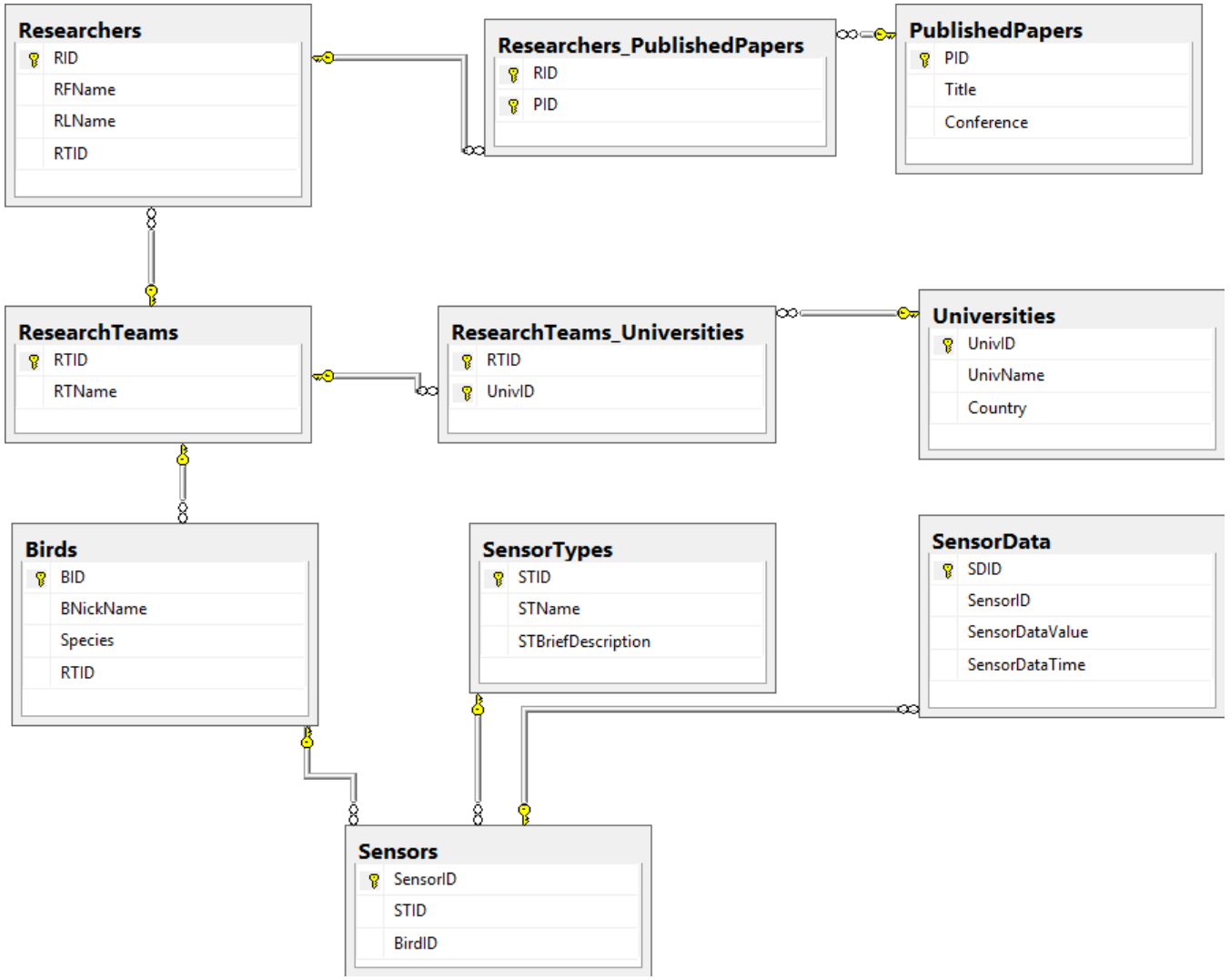
- exemplu de bază de date pentru evidența unor informații dintr-o facultate
- informații care se gestionează:
  - studenți: (nume, prenume, cnp, secție și număr matricol)
  - grupe de studiu (pentru mai mulți ani universitari) și apartenența studenților la aceste formații
  - rezultatele studenților (disciplina, nota și numărul de credite, anul universitar și semestrul)
- determinarea relațiilor din baza de date:
  - pentru o problemă dată soluția nu este unică
  - relații posibile:
    - secții, studenți, grupe, traiectorie, discipline, rezultate

3. Bazele de date relaționale



3. Bazele de date relaționale

- exemplu de bază de date pentru evidența rezultatelor obținute în activitatea de cercetare de mai multe universități în domeniul ornitologiei



4. Gestiunea bazelor de date relaționale cu limbajul SQL

- **SQL** – Structured Query Language
- crearea, manipularea și interogarea datelor într-un SGBD relațional
- scurt istoric:
  - 1970 - E.F. Codd formalizează modelul relațional
  - 1974 - la IBM (San Jose) se definește limbajul SEQUEL (Structured English Query Language)
  - 1976 - la IBM se definește o versiune modificată a limbajului SEQUEL, cu numele SEQUEL/2; după o revizuire, devine limbajul SQL
  - 1986 - SQL devine standard ANSI (American National Standards Institute)
  - 1987 - SQL devine standard ISO (International Standards Organization)
  - revizuiri: SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2006, SQL:2008, SQL:2011, SQL:2016

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- instrucțiunile SQL pot fi grupate în mai multe categorii:
  - definire componente: CREATE, ALTER, DROP
  - gestiune & regăsire date: INSERT, UPDATE, DELETE, SELECT
  - gestiune tranzacții: START TRANSACTION, COMMIT, ROLLBACK

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- CREATE TABLE – definirea unui nou tabel

CREATE TABLE nume_tabel

(definire_coloana [, definire_coloana] ... [, restrictii_tabel])

definire_coloana: nume_coloana tip_data[(lungime)] [DEFAULT valoare] [restrictie_coloana]

```
CREATE TABLE Studenti
     (cods INT PRIMARY KEY,
      cnp CHAR(13),
      nume VARCHAR(50),
      prenume VARCHAR(50) DEFAULT 'Se completeaza
ulterior.',
      varsta INT CHECK (varsta >= 18))
```

- categorii de tipuri de date: numerice, șir de caractere, data-timp etc

* reținem varsta ca atribut pentru simplitatea exemplelor; este de preferat să reținem data nașterii, întrucât aceasta nu se schimbă de la an la an

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- **restricții asociate coloanei**:
  - NOT NULL - nu poate să aibă valori nedefinite
  - PRIMARY KEY - coloana se definește cheie primară
  - UNIQUE - valorile pentru coloană sunt unice
  - CHECK(condiție) - condiția pe care trebuie să o îndeplinească valorile coloanei (condiții simple cu valoarea *true* sau *false*)
  - FOREIGN KEY REFERENCES tabel_parinte [(nume_coloana)] [ON UPDATE *actiune*] [ON DELETE *actiune*]
- **restricții asociate tabelului**:
  - PRIMARY KEY(lista_coloane) - definire cheie primară pentru tabel
  - UNIQUE(lista_coloane) - valorile pentru lista de coloane sunt unice
  - CHECK(condiție) - condiția pe care trebuie să o îndeplinească valorile unei linii

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- **restricții asociate tabelului**:
  - FOREIGN KEY (lista_coloane) REFERENCES tabel_parinte [(lista_coloane)] [ON UPDATE *actiune*] [ON DELETE *actiune*]

- **acțiuni care se pot asocia** unei chei externe:
  - *NO ACTION* – operația nu se poate executa daca se încalcă restricțiile de integritate
  - *SET NULL* - valoarea pentru cheia externă se înlocuiește cu null
  - *SET DEFAULT* - valoarea pentru cheia externă se înlocuiește cu valoarea implicită
  - *CASCADE* - se permite ștergerea sau modificarea în tabelul părinte, dar operația generează ștergeri sau modificări corespunzătoare și în tabelul fiu

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- exemplul 1 (în SQL Server):

```
CREATE TABLE Sectii
  (cod SMALLINT PRIMARY KEY,
  pozitie INT,
  denumire VARCHAR(70))

CREATE TABLE Studenti
  (sectie SMALLINT REFERENCES Sectii(cod),
  nr_matricol CHAR(10),
  nume CHAR(30),
  prenume CHAR(30),
  cnp CHAR(13) UNIQUE,
  PRIMARY KEY(sectie, nr_matricol))

CREATE TABLE Grupe
  (an_univ SMALLINT,
  sectie SMALLINT REFERENCES Sectii(cod),
  an_studiu SMALLINT,
  cod CHAR(10),
  PRIMARY KEY (an_univ, sectie, cod))

CREATE TABLE Traiectorie
  (id INT PRIMARY KEY IDENTITY(1,1),
  an_univ SMALLINT,
  sectie SMALLINT,
  nr_matricol CHAR(10),
  grupa CHAR(10),
  FOREIGN KEY(sectie, nr_matricol) REFERENCES Studenti(sectie,
nr_matricol),
  FOREIGN KEY(an_univ, sectie, grupa) REFERENCES Grupe(an_univ, sectie,
cod))

CREATE TABLE Discipline
  (cod CHAR(10) PRIMARY KEY,
  denumire VARCHAR(70))

CREATE TABLE Rezultate
  (id INT PRIMARY KEY IDENTITY(1,1),
  sectie SMALLINT,
  nr_matricol CHAR(10),
  an_univ SMALLINT,
  semestru SMALLINT,
  disciplina CHAR(10) REFERENCES Discipline(cod),
  nrcredite DECIMAL(4,2),
  nota SMALLINT,
  FOREIGN KEY(sectie, nr_matricol) REFERENCES Studenti(sectie,
nr_matricol))
```

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- exemplul 2:

```
CREATE TABLE Universities
  (UnivID SMALLINT PRIMARY KEY IDENTITY(1,1),
  UnivName NVARCHAR(100) NOT NULL,
  Country NVARCHAR(40))

CREATE TABLE ResearchTeams
  (RTID SMALLINT IDENTITY(1,1),
  RTName NVARCHAR(100) UNIQUE,
  CONSTRAINT PK_ResearchTeams PRIMARY KEY(RTID))

CREATE TABLE ResearchTeams_Universities
  (RTID SMALLINT FOREIGN KEY REFERENCES ResearchTeams(RTID) ,
  UnivID SMALLINT REFERENCES Universities(UnivID),
  PRIMARY KEY(RTID, UnivID))

CREATE TABLE SensorTypes
  (STID TINYINT PRIMARY KEY IDENTITY(1,1),
  STName NVARCHAR(50) UNIQUE,
  STBriefDescription NVARCHAR(300) DEFAULT 'TBW')

CREATE TABLE Birds
  (BID INT PRIMARY KEY IDENTITY(1,1),
  BNickName NVARCHAR(50),
  Species VARCHAR(100),
  RTID SMALLINT,
  CONSTRAINT FK_Birds_ResearchTeams FOREIGN KEY(RTID) REFERENCES
ResearchTeams(RTID))

CREATE TABLE Sensors
  (SensorID INT PRIMARY KEY IDENTITY(1,1),
  STID TINYINT REFERENCES SensorTypes ON DELETE NO ACTION ON UPDATE
CASCADE,
  BirdID INT REFERENCES Birds(BID))

CREATE TABLE SensorData
  (SDID INT PRIMARY KEY IDENTITY(1,1),
  SensorID INT REFERENCES Sensors(SensorID),
  SensorDataValue REAL,
  SensorDataTime DATETIME)

CREATE TABLE Researchers
  (RID SMALLINT PRIMARY KEY IDENTITY(1,1),
  RFName NVARCHAR(90) NOT NULL,
  RLName NVARCHAR(90) NOT NULL,
  RTID SMALLINT REFERENCES ResearchTeams(RTID))

CREATE TABLE PublishedPapers
  (PID INT PRIMARY KEY IDENTITY(1,1),
  Title NVARCHAR(200),
  Conference NVARCHAR(200))

CREATE TABLE Researchers_PublishedPapers
  (RID SMALLINT REFERENCES Researchers(RID) ,
  PID INT REFERENCES PublishedPapers(PID),
  PRIMARY KEY(RID, PID))
```

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- ALTER TABLE – modificarea structurii unui tabel definit

  ALTER TABLE nume_tabel operatie

```
ALTER TABLE Studenti
ADD SimfoniePreferata VARCHAR(50)
```

- câteva operații posibile (există diferențe între SGBD-uri):
  - adăugare / modificare / eliminare coloane:
    - ADD definire_coloana
    - {ALTER COLUMN | MODIFY COLUMN} definire_coloana
    - DROP COLUMN nume_coloana

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- adăugare / ștergere restricție:
  - ADD [CONSTRAINT nume_constrangere] PRIMARY KEY(lista_coloane)
  - ADD [CONSTRAINT nume_constrangere] UNIQUE(lista_coloane)
  - ADD [CONSTRAINT nume_constrangere] FOREIGN KEY (lista_coloane) REFERENCES nume_tabel[(lista_coloane)] [ON UPDATE *actiune*] [ON DELETE *actiune*]
  - DROP [CONSTRAINT] nume_constrangere

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- DROP TABLE – ștergerea unui tabel

  DROP TABLE nume_tabel

```
DROP TABLE Studenti
```

- subsetul SQL care permite crearea, ștergerea, modificarea diverselor componente (e.g., tabelelor) poartă denumirea de **Limbaj de definire a datelor** (LDD)

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- **modificarea datelor dintr-un tabel**
- INSERT – adăugarea de înregistrări

  INSERT INTO nume_tabel [(lista_coloane)] VALUES (lista_valori)

  INSERT INTO nume_tabel [(lista_coloane)] subinterogare

  , unde *subinterogare* se referă la o mulțime de înregistrări (generată cu ajutorul instrucțiunii SELECT)

```
INSERT INTO Studenti (sid, cnp, nume, prenume, varsta)
VALUES (1, '123456789012', 'Popescu', 'Maria', 20)
```

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- **modificarea datelor dintr-un tabel**
- UPDATE – modificarea unor înregistrări

UPDATE nume_tabel

SET nume_coloana=expresie [, nume_coloana=expresie] ... [WHERE conditie]

- se modifică înregistrările din tabel pentru care condiția din clauza WHERE se evaluează la *adevărat*; dacă se omite clauza WHERE se consideră toate înregistrările din tabel; valorile coloanelor precizate în SET se schimbă la valoarea expresiilor asociate

```
UPDATE Studenti
SET varsta = varsta + 1
WHERE cnp = '123456789012'
```

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- **modificarea datelor dintr-un tabel**
- DELETE – ștergerea de înregistrări

  DELETE FROM nume_tabel [WHERE conditie]
- se elimină din tabel înregistrările pentru care condiția din clauza WHERE se evaluează la *adevărat*; dacă se omite clauza WHERE, se șterg toate înregistrările din tabel

```
DELETE
FROM Studenti
WHERE nume = 'Popescu'
```

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- **condiții de filtrare**
- se folosesc în contextul unei linii curente din tabel
  - expresie operator_relational expresie
  - expresie [NOT] BETWEEN min AND max
  - expresie [NOT] LIKE 'sablon' ("%" – orice număr de caractere, "_" – un caracter)
  - expresie IS [NOT] NULL
  - expresie [NOT] IN (valoare [, valoare] ...)
  - expresie [NOT] IN (subselectie)
  - expresie operator_relational {ALL | ANY} (subselectie)

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- **condiții de filtrare**
- condiții echivalente:

  "expresie IN (subselectie)" $\Longleftrightarrow$ "expresie = ANY (subselectie)"

  "expresie NOT IN (subselectie)" $\Longleftrightarrow$ "expresie <> ALL (subselectie)"
- [NOT] EXISTS (subselectie)
- o condiție de filtrare poate fi:
  - condiție elementară (descrisă anterior)
  - (condiție)
  - NOT condiție
  - condiție$_1$ AND condiție$_2$
  - condiție$_1$ OR condiție$_2$

4. Gestiunea bazelor de date relaționale cu limbajul SQL

- logica trivalenta (valori de adevăr: *adevărat*, *fals*, *nedeterminat*)

	TRUE	FALSE	NULL
NOT	FALSE	TRUE	NULL

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

5. Referințe

- [Ta13] ȚÂMBULEA, L., Curs Baze de date, Facultatea de Matematică și Informatică, UBB, 2013-2014
- [Ra00] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems (2nd Edition), McGraw-Hill, 2000
- [Da03] DATE, C.J., An Introduction to Database Systems (8th Edition), Addison-Wesley, 2003
- [Ga08] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., Database Systems: The Complete Book, Prentice Hall Press, 2008
- [Ha96] HANSEN, G., HANSEN, J., Database Management And Design (2nd Edition), Prentice Hall, 1996
- [Ra07] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems, McGraw-Hill, 2007, http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- [Si10] SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S., Database System Concepts, McGraw-Hill, 2010, http://codex.cs.yale.edu/avi/db-book/
- [Ta03] ȚÂMBULEA, L., Baze de date, Litografiat Cluj-Napoca 2003