# Baze de Date

## Curs 3. Gestiunea BD cu limbajul SQL: instructiuni de definire a datelor

Ioana Ciuciu
ioana.ciuciu@ubbcluj.ro
http://www.cs.ubbcluj.ro/~oana/

# Planificare

Sunt posibile mici modificari ale planificarii in timpul semestrului

| Saptamana | Curs | Seminar | Laborator |
|---|---|---|---|
| S1 | 1. Concepte fundamentale ale bazelor de date. Modelare conceptuala | 1. Modelul Entitate-Relatie. Modelul relational | 1. Modelarea unei BD in modelul ER si implementarea ei in SQL Server |
| S2 | 2. Modelul relational de organizare a bazelor de date. Modelare conceptuala | | |
| S3 | 3. Gestiunea bazelor de date relationale cu limbajul SQL (DDL) | 2. Limbajul SQL – definirea si actualizarea datelor | 2. Interogari SQL |
| S4 | 4. Gestiunea bazelor de date relationale cu limbajul SQL (DML) | | |
| S5-6 | 5-6. Dependente functionale, forme normale | 3. Limbajul SQL – regasirea datelor | 3. Interogari SQL avansate |
| S7 | 7. JDBC (Java Database Connectivity) | 4. Proceduri stocate | 4. Proceduri stocate. View. Trigger |
| S8 | 8. Interogarea bazelor de date relationale cu operatori din algebra relationala | | |
| S9 | 9. Structura fizica a bazelor de date relationale | 5. View-uri. Functii definite de utilizator. Trigger | |
| S10-11 | 10-11. Indecsi. Arbori B. Fisiere cu acces direct | 6. Formele normale ale unei relatii. Indecsi | |
| S12 | 12. Evaluarea interogarilor in bazele de date relationale | | |
| S13 | 13. Extensii ale modelului relational si baze de date NoSQL | 7. Probleme | Examen practic |
| S14 | 14. Aplicatii | | |

# Plan

- Gestiunea bazelor de date relationale cu limbajul SQL
  - Definirea datelor - Data Definition Language (DDL)
  - Manipularea datelor - Data Manipulation Language (DML)
    - Cursurile 3 si 4

*Majoritatea slide-urilor acestui curs sunt adaptate dupa materialul didactic al Cameliei Andor*

# Limbajul SQL

- SQL (Structured Query Language) este un limbaj standard pentru accesarea și manipularea bazelor de date
- SQL este un standard ANSI (American National Standards Institute)
- Există mai multe implementări ale limbajului SQL
- Comenzile principale ( SELECT, INSERT, UPDATE, DELETE) sunt permise într-un mod aproape unitar de toate implementările, pentru compatibilitate

# Limbajul SQL

- **SQL poate extrage date din baza de date**
- SQL poate insera înregistrări în baza de date
- SQL poate actualiza înregistrări în baza de date
- SQL poate șterge înregistrări în baza de date
- SQL poate crea noi baze de date
- SQL poate crea noi tabele în baza de date
- SQL poate crea proceduri stocate, funcții, triggere
- SQL poate crea vizualizări în baza de date
- SQL poate stabili permisiuni asupra entităților din baza de date

# Limbajul SQL

- Majoritatea operațiilor efectuate asupra bazelor de date se fac cu comenzi/instrucțiuni SQL
- SQL nu ține cont de diferența dintre literele mari și mici
- Unele sisteme de gestiune a bazelor de date necesită caracterul ; după instrucțiuni

# Limbajul SQL

- **DML = Data Manipulation Language** – comenzi pentru extragere, inserare, actualizare și ștergere a datelor
  - SELECT – extrage date din baza de date
  - INSERT INTO – inserează date noi în baza de date
  - UPDATE – actualizează date în baza de date
  - DELETE – șterge date din baza de date

# Limbajul SQL

- **DDL = Data Definition Language** – comenzi pentru creare/modificare bază de date, tabele, indecși, stabilire relații între tabele, constrângeri
  - CREATE DATABASE
  - ALTER DATABASE
  - DROP DATABASE
  - CREATE TABLE
  - ALTER TABLE
  - DROP TABLE
  - CREATE INDEX
  - ALTER INDEX
  - DROP INDEX

# Limbajul SQL: DDL

- Instrucțiunea CREATE DATABASE se folosește pentru a crea o bază de date
- Sintaxa:

  CREATE DATABASE database_name
- Exemplu:

  CREATE DATABASE World

- Instrucțiunea ALTER DATABASE se folosește pentru a modifica o bază de date
- Exemplu de modificare a numelui unei baze de date:

  ALTER DATABASE World

  MODIFY Name=People

# Limbajul SQL: DDL

- Instrucțiunea DROP DATABASE se folosește pentru a șterge o bază de date

- Sintaxa:

  DROP DATABASE database_name

- Exemplu:

  DROP DATABASE People

# Limbajul SQL: DDL

- O bază de date conține cel puțin un tabel identificat prin nume
- Tabelele conțin înregistrări cu date
- Exemplu: tabela Persoane

| Id | Nume | Prenume | Localitate |
|---|---|---|---|
| 1 | Pop | Oana | Sibiu |
| 2 | Porumb | Sebastian | Oradea |
| 3 | Georgescu | Ana | București |

  - Un tabel cu 4 coloane și 3 înregistrări

# Limbajul SQL: DDL

- Instrucțiunea CREATE TABLE se folosește pentru a crea un tabel într-o bază de date

- Sintaxa:

```
CREATE TABLE table_name
 ( column_name1  data_type,
   column_name2 data_type,
   …
 )
```

# Limbajul SQL: DDL

- Dorim să creăm un tabel numit Persoane care conține câmpurile id, nume, prenume, localitate

```
CREATE TABLE Persoane
 ( id int,
   nume varchar(30),
   prenume varchar(30),
   localitate varchar(30)
 )
```

# Limbajul SQL: DDL

- Instrucțiunea ALTER TABLE se folosește pentru a modifica un tabel

- Sintaxa instrucțiunii pentru adăugarea unei coloane într-un tabel:

```
ALTER TABLE table_name
ADD column_name datatype
```

- Exemplu de adăugare a unei coloane într-un tabel:

```
ALTER TABLE Persoane
ADD  data_nașterii date
```

# Limbajul SQL: DDL

- Sintaxa instrucțiunii pentru schimbarea tipului de date al unei coloane dintr-un tabel:

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype
```

- Exemplu de schimbare a tipului de date al unei coloane dintr-un tabel:

```
ALTER TABLE Persoane
ALTER COLUMN data_nașterii datetime
```

# Limbajul SQL: DDL

- Sintaxa instrucțiunii pentru ștergerea unei coloane dintr-un tabel:

```
ALTER TABLE table_name
DROP COLUMN column_name
```

- Exemplu de ștergere a unei coloane dintr-un tabel:

```
ALTER TABLE Persoane
DROP COLUMN data_nașterii
```

# Limbajul SQL: DDL

- Instrucțiunea DROP TABLE se folosește pentru a șterge un tabel dintr-o bază de date

- Sintaxa:

  DROP TABLE table_name

- Exemplu:

  DROP TABLE Persoane

# Limbajul SQL: DDL

- În limbajul SQL, fiecare coloană, variabilă locală, expresie sau parametru are un tip (de date)
- Un tip de date este un atribut care specifică tipul de date pe care îl poate stoca obiectul respectiv
- Exemple:
  - int, tinyint, smallint, bigint, decimal, float, money, nchar, varchar, datetime, date, time

# Limbajul SQL: DDL

- **Constrângerile** se folosesc pentru a asigura integritatea datelor pe care le introducem într-un tabel

- **Integritatea** datelor se poate asigura în mod *declarativ*, ca parte din definiția tabelului sau în mod *procedural* prin proceduri stocate sau triggers

- Constrângerile se pot specifica la crearea tabelului (în instrucțiunea CREATE TABLE) dar și după ce tabelul a fost creat (cu instrucțiunea ALTER TABLE)

# Limbajul SQL: DDL

- Constrângeri:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK
  - DEFAULT

# Limbajul SQL: DDL

- În mod implicit un tabel permite inserarea de valori NULL în câmpurile sale
- Dacă nu dorim să permitem introducerea de valori NULL într-o coloană, aplicăm **constrângerea NOT NULL** pentru coloana respectivă
- Ca rezultat, nu vom putea insera sau actualiza înregistrări care nu specifică o valoare pentru coloana respectivă

# Limbajul SQL: DDL

- Constrângere NOT NULL – Exemplu:

```
CREATE TABLE Students
  (
    s_id int NOT NULL,
    FirstName  varchar(50),
    LastName varchar(50),
    City varchar(50)
  )
```

# Limbajul SQL: DDL

- **Constrângerea UNIQUE** se folosește asupra coloanelor în care nu dorim să permitem valori duplicate
- Se pot defini mai multe constrângeri UNIQUE în același tabel
- Se poate defini pe una sau mai multe coloane
- Ambele constrangeri, UNIQUE si PRIMARY KEY, garanteaza unicitatea unei coloane sau a unei multimi de coloane
- Constrangerea PRIMARY KEY contine automat si constrangerea UNIQUE

# Limbajul SQL: DDL

- Exemplu de constrângere UNIQUE pe o coloană:
- CREATE TABLE Students

```
(
  s_id int UNIQUE,
  FirstName varchar(50),
  LastName varchar(50),
  City varchar(50)
)
```

# Limbajul SQL: DDL

- Exemplu de constrângere UNIQUE pe mai multe coloane:
- CREATE TABLE Students

```
(
  s_id int NOT NULL,
  FirstName  varchar(50),
  LastName varchar(50),
  City varchar(50),
  CONSTRAINT uc_StudentID UNIQUE (s_id,
LastName)
)
```

# Limbajul SQL: DDL

- Definirea unei constrângeri UNIQUE după ce tabelul a fost creat se face cu ajutorul instrucțiunii ALTER TABLE
- Exemplu de definire a unei constrângeri UNIQUE pe o singură coloană:

```
ALTER TABLE Students
ADD UNIQUE(s_id)
```

- Exemplu de definire a unei constrângeri UNIQUE pe mai multe coloane:

```
ALTER TABLE Students
ADD CONSTRAINT uc_StudentID UNIQUE(s_id, LastName)
```

# Limbajul SQL: DDL

- O constrângere poate fi eliminată cu ajutorul instrucțiunii DROP
- Sintaxa:

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name
```

- Exemplu:

```
ALTER TABLE Students
DROP CONSTRAINT uc_StudentID
```

# Limbajul SQL: DDL

- Fiecare tabel trebuie să aibă o singură **cheie primară**
  - Cheia primară identifică în mod unic fiecare înregistrare din tabel
  - Nu permite introducerea valorilor duplicate sau NULL în coloana pe care este definită
  - Poate fi definită pe o singură coloană sau pe o combinație de coloane
  - În cazul în care este definită pe mai multe coloane, combinația de valori din acele coloane trebuie să fie unică
  - Se poate defini o singură constrângere de tip cheie primară (primary key) într-un tabel

# Limbajul SQL: DDL

- Exemplu de definire a unei constrângeri PRIMARY KEY la crearea unui tabel:

```
CREATE TABLE Students
 (
   s_id int PRIMARY KEY,
   FirstName varchar(30),
   LastName varchar(50),
   City varchar(50),
)
```

# Limbajul SQL: DDL

- Exemplu de definire a unei constrângeri PRIMARY KEY pe mai multe coloane la crearea unui tabel:

```
CREATE TABLE Students
 (
   s_id int ,
   FirstName varchar(30),
   LastName varchar(50),
   City varchar(50),
   CONSTRAINT pk_Student PRIMARY KEY (s_id, LastName)
 )
```

# Limbajul SQL: DDL

- Pentru a putea crea o cheie primară după crearea tabelului, coloana sau coloanele pe care dorim să le includem în cheia primară trebuie să aibă definită o constrângere NOT NULL
- Exemplu de definire a unei constrângeri PRIMARY KEY după crearea tabelului:

```
ALTER TABLE Students
ADD CONSTRAINT pk_Student PRIMARY KEY(s_id, FirstName)
```

- Exemplu de eliminare a unei constrângeri PRIMARY KEY:

```
ALTER TABLE Students
DROP CONSTRAINT pk_Student
```

# Limbajul SQL: DDL

- Un **foreign key (cheie externa)** pointează la un primary key (cheie primara) dintr-un alt tabel
  - Tabelul Clienți

| IDClient | Nume | Prenume | Localitate |
|---|---|---|---|
| 1 | Pop | Oana | Cluj-Napoca |
| 2 | Rus | Andrei | Sibiu |

  - Tabelul Comenzi

| IDCom | NrCom | IDClient |
|---|---|---|
| 1 | 3455 | 2 |
| 2 | 3456 | 1 |

# Limbajul SQL: DDL

- Un **foreign key (cheie externa)** pointează la un primary key (cheie primara) dintr-un alt tabel
  - Tabelul Clienți

primary key

| IDClient | Nume | Prenume | Localitate |
|---|---|---|---|
| 1 | Pop | Oana | Cluj-Napoca |
| 2 | Rus | Andrei | Sibiu |

  - Tabelul Comenzi

foreign key

| IDCom | NrCom | IDClient |
|---|---|---|
| 1 | 3455 | 2 |
| 2 | 3456 | 1 |

# Limbajul SQL: DDL

- Coloana IDClient din tabelul Comenzi pointează spre coloana IDClient din tabelul Clienți
- Coloana IDClient din tabelul Comenzi este FOREIGN KEY, iar coloana IDClient din tabelul Clienți este PRIMARY KEY

- Constrângerea FOREIGN KEY este folosită pentru a preveni acțiuni care ar distruge legăturile dintre cele două tabele, dar și pentru a împiedica introducerea unor date invalide care nu se regăsesc în coloana care este PRIMARY KEY
- **Nu se pot face modificări în tabelul care conține cheia primară dacă aceste modificări distrug legături spre date din tabelul care conține foreign key**

# Limbajul SQL: DDL

- Exemplu de definire a unei constrângeri FOREIGN KEY la crearea unui tabel:

```
CREATE TABLE Comenzi
( IDCom int PRIMARY KEY,
NrCom int,
IDClient int FOREIGN KEY REFERENCES Clienți(IDClient)
)
```

# Limbajul SQL: DDL

- Exemplu de definire a unei constrângeri FOREIGN KEY cu numele fk_Client la crearea unui tabel:

```
CREATE TABLE Comenzi
( IDCom int PRIMARY KEY,
NrCom int,
IDClient int,
CONSTRAINT fk_Client FOREIGN KEY (IDClient)
REFERENCES Clienți(IDClient)
)
```

# Limbajul SQL: DDL

- Exemplu de definire a unei constrângeri FOREIGN KEY după crearea tabelului:

```
ALTER TABLE Comenzi
ADD FOREIGN KEY (IDClient)
REFERENCES Clienți(IDClient)
```

SAU

```
ALTER TABLE Comenzi
ADD CONSTRAINT fk_Client FOREIGN KEY (IDClient)
REFERENCES Clienți(IDClient)
```

# Limbajul SQL: DDL

- Se pot specifica **acțiuni** care vor fi efectuate în cazul în care un user încearcă să șteargă sau să modifice un primary key spre care pointează un foreign key

- Următoarele acțiuni pot fi specificate în acest caz:
  - NO ACTION
  - CASCADE
  - SET NULL
  - SET DEFAULT

# Limbajul SQL: DDL

- NO ACTION – motorul bazei de date afișează o eroare și actualizarea sau ștergerea eșuează
- CASCADE – se șterge sau se actualizează înregistrarea din tabelul care conține cheia referită împreună cu înregistrările corespunzătoare din tabelul care conține foreign key-ul
- SET NULL – se va seta valoarea null pentru toate valorile care alcătuiesc foreign key-ul atunci când înregistrarea corespunzătoare din tabelul care conține cheia referită este actualizată sau ștearsă
- SET DEFAULT – toate valorile care alcătuiesc foreign key-ul sunt setate pe valoarea default (cu condiția să fie definite valori default pe coloana sau coloanele respective) atunci când înregistrarea corespunzătoare din tabelul care conține cheia referită este actualizată sau ștearsă

# Limbajul SQL: DDL

- Exemplu de definire a unei constrângeri FOREIGN KEY cu acțiuni care au loc în caz de modificare sau ștergere:

```
CREATE TABLE Comenzi
(
IDCom int PRIMARY KEY,
NrCOm int,
IDClient int FOREIGN KEY REFERENCES
Clienți(IDClient)
ON DELETE CASCADE
ON UPDATE CASCADE
 )
```

# Limbajul SQL: DDL

- **Constrângerea CHECK** se folosește pentru a limita intervalul de valori ce se pot introduce pentru o anumită coloană
  - Se poate defini pe o coloană, iar în acest caz limitează valorile ce pot fi introduse pentru coloana respectivă
  - Se poate defini pe mai multe coloane

# Limbajul SQL: DDL

- Exemplu de definire a unei constrângeri CHECK la crearea tabelului pe o coloană:

```
CREATE TABLE Clienți
(
IDClient int PRIMARY KEY CHECK(IDClient>0),
Nume varchar(50) NOT NULL,
Prenume varchar(50),
Localitate varchar(50)
)
```

# Limbajul SQL: DDL

- Exemplu de constrângere CHECK definită pe mai multe coloane la crearea unui tabel:

- CREATE TABLE Clienți
  (
  IDClient int PRIMARY KEY,
  Nume varchar(50) NOT NULL,
  Prenume varchar(50),
  Localitate varchar(50),
  CONSTRAINT ck_IDClient CHECK(IDClient>0 AND
  Localitate IN ('Cluj-Napoca','Sibiu')
  )

# Limbajul SQL: DDL

- Exemplu de adăugare a unei constrângeri CHECK după crearea tabelului:

```
ALTER TABLE Clienți
ADD CHECK (IDClient>0)
```

- Exemplu de adăugare și stabilire a unui nume pentru o constrângere CHECK după crearea tabelului:

```
ALTER TABLE Clienți
ADD CONSTRAINT ck_Client
CHECK (IDClient>0 AND Localitate IN ('Cluj-Napoca','Sibiu'))
```

# Limbajul SQL: DDL

- **Constrângerea DEFAULT** se folosește pentru a insera o valoare implicită într-o coloană
- Valoarea implicită va fi adăugată pentru toate înregistrările noi dacă nu se specifică o altă valoare
- Se poate folosi și pentru a insera valori sistem obținute prin apelul unor funcții

- Exemplu de definire a unei constrângeri DEFAULT după crearea unui tabel:

```
ALTER TABLE Clienți
ADD CONSTRAINT d_Localitate DEFAULT 'Cluj-Napoca' FOR
Localitate
```

- Eliminarea unei constrângeri DEFAULT

```
ALTER TABLE Clienți
DROP CONSTRAINT d_Localitate
```

# Limbajul SQL: DDL

- Exemplu de definire a unei constrângeri DEFAULT la crearea unui tabel:

```
CREATE TABLE Comenzi
(
IDCom int PRIMARY KEY,
 NrCOm int NOT NULL,
IDClient int,
DataCom date DEFAULT GETDATE()
)
```

# Limbajul SQL: DML

- Instrucțiunea **INSERT INTO** se folosește pentru a insera noi înregistrări într-un tabel

- Sintaxa:

```
INSERT INTO table_name
VALUES (value1, value2,…)
```

- SAU

```
INSERT INTO table_name
(column_name1, column_name2, column_name3,…)
VALUES (value1, value2, value3, ….)
```

# Limbajul SQL: DML

- Specificarea coloanelor după numele tabelului este opțională
- Prin specificarea coloanelor controlăm asocierile coloană-valoare, deci nu ne bazăm pe ordinea în care apar coloanele atunci când a fost creat tabelul sau când structura tabelului a fost modificată ultima dată
- Dacă nu specificăm o valoare pentru o coloană, SQL Server va verifica dacă există o valoare default pentru coloana respectivă iar dacă nu există și coloana nu permite NULL atunci inserarea nu va avea loc

# Limbajul SQL: DML

- Exemplu de inserare a unei noi înregistrări în tabelul Clienți

```
INSERT INTO Clienți (IDClient, Nume, Prenume,
Localitate)
VALUES (1, 'Pop', 'Anda','Sibiu')

SAU

INSERT INTO Clienți
VALUES (1, 'Pop', 'Anda','Sibiu')
```

# Limbajul SQL: DML

- Instrucțiunea **UPDATE** se folosește pentru a actualiza date într-un tabel

- Sintaxa:

```
UPDATE table_name
SET column1=value1, column2=value2, …
WHERE some_column=some_value
```

- Omiterea clauzei WHERE va rezulta în actualizarea tuturor înregistrărilor din tabel

# Limbajul SQL: DML

- Exemplu de actualizare a unei înregistrări dintr-un tabel:

```
UPDATE Clienți
SET Localitate='Cluj-Napoca'
WHERE Nume='Pop' AND Prenume='Anda'
```

# Limbajul SQL: DML

- Instrucțiunea **DELETE** se folosește pentru a șterge înregistrări dintr-un tabel

- Sintaxa:

```
DELETE FROM table_name
WHERE some_column = some_value
```

- Omiterea clauzei WHERE rezultă în ștergerea tuturor înregistrărilor din tabel

# Limbajul SQL: DML

- Exemplu de ștergere a tuturor înregistrărilor din tabelul Clienți pentru care coloana Localitate are valoarea ‘Sibiu’:

```
DELETE FROM Clienți
WHERE Localitate =‘Sibiu’
```

- Exemplu de ștergere a tuturor înregistrărilor din tabelul Clienți:

```
DELETE FROM Clienți
```

# Cursul urmator

- Curs 4 – Data Manipulation Language – continuare (extragerea datelor)

# Referinte


- [Da04] DATE, C.J., An Introduction to Database Systems (8th Edition), Addison-Wesley, 2004.
- [Ga08] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., *Database Systems: The Complete Book*, Pearson Prentice Hall, 2008
- [Mi09] MIU, L., OZSU, M.T., Encyclopedia of Database Systems, Springer 2009 (3818 pages).
- [Ra07] RAMAKRISHNAN, R., Database Management Systems. McGraw-Hill, 2007, http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- [Si10] SILBERSCHATZ A., KORTZ H., SUDARSHAN S., Database System Concepts, McGraw-Hill, 2010, http://codex.cs.yale.edu/avi/db-book/
- [Ul11] ULLMAN, J., WIDOM, J., A First Course in Database Systems (3rd Edition), Addison-Wesley + Prentice-Hall, 2011
- Leon Tambulea, curs de Baze de Date, UBB Cluj-Napoca
- Andor Camelia, seminar de Baze de Date, Matematica-Informatica, UBB Cluj-Napoca, 2016
- https://www.w3schools.com/sql/