Cap.2. Baze de date relationale

CUPRINS

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

1. Relatii – Atribute – Domenii

DEFINITII

DEF 1. Bază de date relaţională : colectie de mai multe tabele (table), legate între ele.

DEF.2: Bază de date relaţională: o **mulţime finită de relaţii**, in care:

- fiecare **relaţie** este unica si reprezinta **o multime de entitati** si/sau **o multime de asocieri**
- **Relatia= tabel** care se defineste prin intermediul **atributelor=coloanele** sale

Atributele unei relaţii: atributele tipului de entitate sau de asociere

- fiecare **atribut** (coloana) are un **nume** (Ai) si un **domeniu** de definiţie D(Ai)
- pentru o entitate data, un atribut poate lua o singură valoare (scalar)

Atributele pot fi: **simple** (un element) sau **compuse** (o submultime de atribute)

1. Relatii – Atribute – Domenii

DEFINITII

Domeniu: multime de valori D = {di| i = 1,…, n }, definit printr-o specificare de tip, unde:

- D este **numele domeniului**
- di este un **element al domeniului** care satisface anumite constrangeri

- Elementele domeniilor sunt indivizibile
- null (**lipsa de informatie / valoare necunoscuta**), apartine oricarui domeniu

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

SCHEMA RELATIEI

DEFINITII

Schema relaţiei (descriere a unei relaţii): **R(A1,A2,...Ai,...An)**

unde:

- R = **numele schemei relaţiei**
- A1,A2,...Ai,..An = **lista ordonată a atributelor**
- D(Ai) = atribut Ai definit pe **domeniul** său
- n =**gradul relaţiei**: numărul de atribute ale schemei acelei relatii

SCHEMA RELATIEI

DEFINITII

Tabel STUDENTI

Nume	Prenume	Data Nasterii	Adresa	Facultatea
Pop	Ioan	31.12.1990	Cluj-Napoca	IE
Albu	Damian	01.02.1991	Oradea	AC

Exemplu

Schema relatiei

5 =gradul relaţiei

Ex: STUDENTI (Nume, Prenume, DataNasterii, Adresa, Facultatea)

Numele schemei relatiei

atributele relatiei

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TUPLURILE RELATIEI

DEFINITII

Relaţia r definita prin schema R(A1,A2,...Ai,...An) = mulţime finita de n- **tupluri** t

Tuplul t = listă ordonată de n valori: t = <v1,v2,...vi,...vn>,

unde :

- $1 \leq i \leq n$
- vi este o valoare a atributului Ai, $vi \in D(Ai)$

Tabel STUDENTI

Nume	Prenume	Data Nasterii	Adresa	Facultatea
Pop	Ioan	31.12.1990	Cluj-Napoca	IE
Albu	Damian	01.02.1991	Oradea	AC

Tuplu

STAREA & CARDINALITATEA RELATIEI

DEFINITII

Relaţia r definita prin schema R(A1,A2,...Ai,...An) = mulţime finita de n- **tupluri** t

Instanta schemei (tipului) R= relatia r(R) **variabila** .

- Valoarea variabilei: **starea relatiei**
- Numarul de tupluri ale unei relatii: **cardinalitatea relatiei**
- Fiecare tuplu este unic intr-o relatie (nu exista tupluri duplicat)

Tabel STUDENTI

Cardinalitatea relatiei: 2

Nume	Prenume	Data Nasterii	Adresa	Facultatea
Pop	Ioan	31.12.1990	Cluj-Napoca	IE
Albu	Damian	01.02.1991	Oradea	AC

Starea relatiei

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

2. Reprezentarea relatiilor prin tabele

DEFINITII

Un **tabel** (table): reprezentarea grafica a unei relaţii;

Componente tabel:

- **Numele tabelului** = numele relaţiei
- **Coloanele** = atributele relatiei
- **Capul tabelului** =numele atributelor (coloanelor) → **schema relatiei**
- Mulţime de **linii**, fiecare linie corespunde unui tuplu → **starea relatiei**
- **Valori** ale atributelor fiecarui tuplu

2. Reprezentarea relatiilor prin tabele

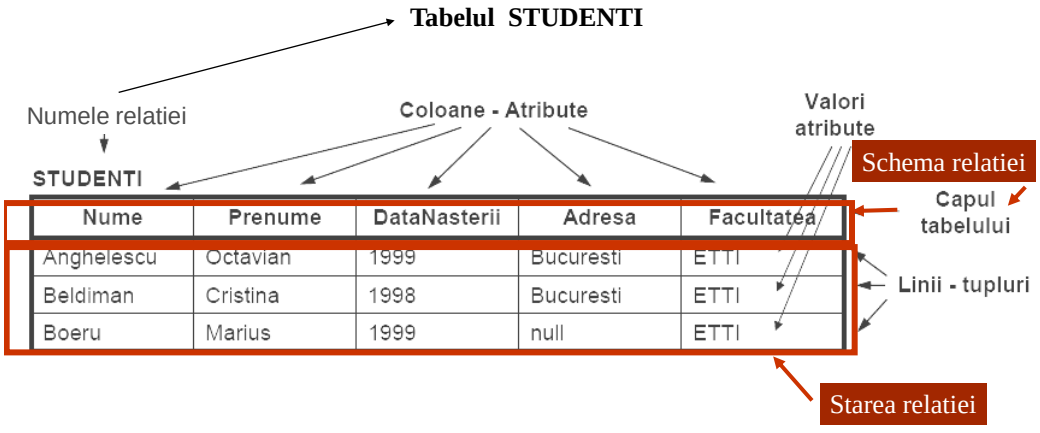
DEFINITII

Fiecare tabel, individual, are următoarele proprietăţi:

- Nu există rânduri duplicate
- Nu există nume de coloane identice (duplicate)
- Ordinea rândurilor NU este importantă
- Ordinea coloanelor NU este importantă
- Valorile (câmpurile) sunt atomice (nedecompozabile).

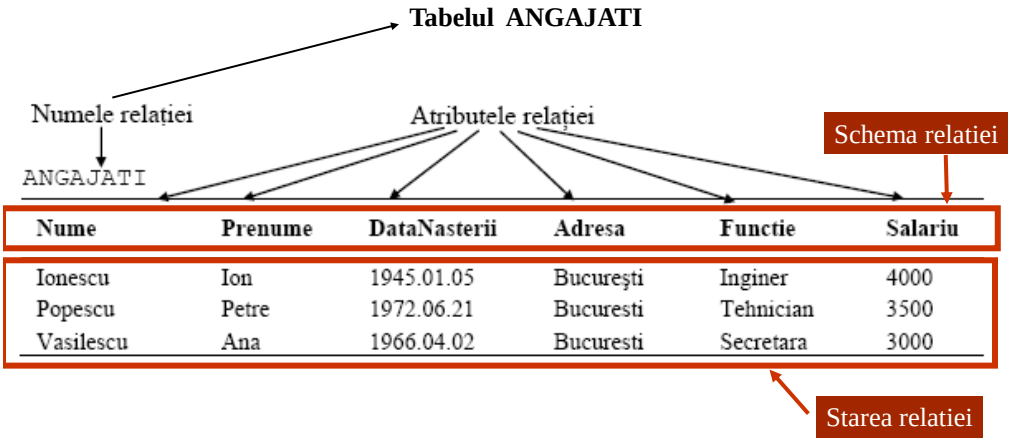
Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

EXEMPLE



Nume	Prenume	DataNasterii	Adresa	Facultatea
Anghelescu	Octavian	1999	Bucuresti	ETTI
Beldiman	Cristina	1998	Bucuresti	ETTI
Boeru	Marius	1999	null	ETTI

EXEMPLE



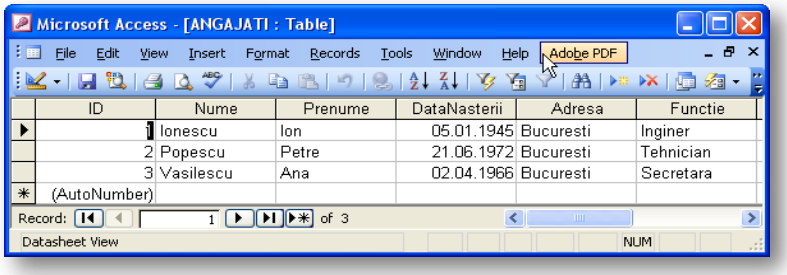
Nume	Prenume	DataNasterii	Adresa	Functie	Salariu
Ionescu	Ion	1945.01.05	Bucureşti	Inginer	4000
Popescu	Petre	1972.06.21	Bucuresti	Tehnician	3500
Vasilescu	Ana	1966.04.02	Bucuresti	Secretara	3000

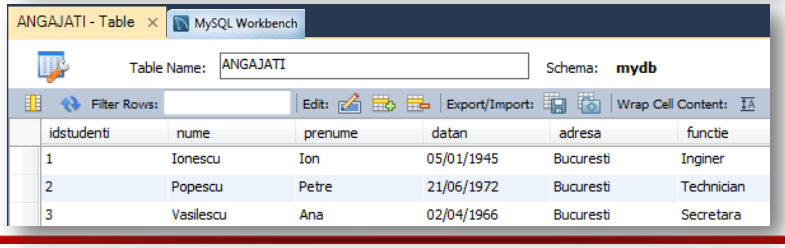
Starea relatiei

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

AFISAREA TABELELOR

SGBD-urile ofera instrumente de proiectare si afisare a tabelelor





3. Limbajul SQL

DEFINITII

Limbajul SQL (Structured Query Language): este limbajul utilizat de majoritatea SBD relationale pentru definirea şi manipularea datelor.

- face parte din limbajele relationale: QBE, QUEL, SQUARE, ALPHA, ISBL
- limbajul cel mai frecvent utilizat pentru a accesa o bază de date relaţională;
- poate fi utilizat de orice SGBD (Access, Oracle, SQL-Server, MS SQL Server, MySQL,etc.);
- este un limbaj simplu, ne-procedural, cu comenzi intuitive în limba engleză;
- este un limbaj standard pentru bazele de date

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

3. Limbajul SQL

ISTORIC

Anul	Denumire	Caracteristici
1986	SQL-86	Publicat de ANSI (SQL1); ratificat de ISO in 1987
1989	SQL-89	Revizii minore
1992	**SQL-92**	**Revizii majore, redenumit SQL2**
1999	**SQL-1999**	**Redenumit SQL3, adauga unele caracteristici obiect-relationale**
2003	SQL-2003	Adauga unele trasaturi referitoare la limbajul XML
2006	SQL-2006	Utilizare SQL in conjunctie cu XML
2008	SQL- 2008	Se introduce TRUNCATE si triggeri INSTEAD Of
2011	SQL: 2011	Revizii minore

CARACTERISTICILE GENERALE ALE LIMBAJULUI SQL

SQL utilizeaza **reprezentarea prin tabele a relaţiilor**: termenii **linie, coloană**

Componentele limbajului SQL:

- **componenta de descriere** a datelor (Limbaj de Descriere a Datelor-LDD)
- **componenta de manipulare** a datelor (Limbaj de Manipulare a Datelor-LMD)
- **alte componente**: controlul tranzactiilor, controlul securitatii,protectia datelor

SQL este un limbaj neprocedural:

- **nu conţine instrucţiuni de control** (for, while, if, etc)

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

AVANTAJELE LIMBAJULUI SQL

- NU este un limbaj comercial si este proiectat doar pentru: citirea si scrierea in BD
- sta la baza utilizarii tuturor SGBD
- usor de invatat, putine cuvinte descriptive
- limbaj extrem de puternic : permite realizarea unor operatii complexe asupra BD

- o **instructiune se poate scrie pe una/mai multe linii**
- **într-o linie se pot introduce una/mai multe instructiuni**
- Limbajul SQL este **case-insensitive** (nu deosebeste literele mici de cele mari) cu exceptia identificatorilor delimitati (quoted).

STRUCTURA LEXICALA A LIMBAJULUI SQL

Componente

1) Tipuri de date
2) Cuvinte cheie
3) Identificatori
4) Instrucţiuni/Comenzi
5) Variabile
6) Operanzi
7) Operatori
8) Functii

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE IN LIMBAJUL SQL

1) TIPURI DE DATE

a) numeric
b) şiruri de caractere
c) şiruri de biţi
d) data calendaristică,timp

a) TIPURI NUMERICE

- numere întregi:
- numere reale
- numere zecimale

TIPURI DE DATE IN LIMBAJUL SQL

a) TIPURI NUMERICE

1. **numere întregi**:
   - `integer` sau `int`(4 octeţi) , -2^31÷2^31-1
   - `smallint`(2 octeţi), -2^15÷2^15-1
   - `bigint`(8 octeţi), -2^63÷2^63-1
   - `tinynt`(1 octet), 0÷255

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE IN LIMBAJUL SQL

a) TIPURI NUMERICE

2. numere reale reprezentate în virgulă mobila:

- `float[(n)]=` nr. flotant in intervalul -1.79E+38÷ 1.79E+38 , n intreg in intervalul 1÷53

n	precizie	reprezentare	corespondenta
1-24	7 cifre zecimale	4 octeti	flotant simpla precizie
25-53	15 cifre zecimale	8 octeti	flotant dubla precizie

- `real` =nr flotant in intervalul -3.40E+38÷ 3.40E+38 reprezentat pe 4 octeti si sinonim cu `float`

TIPURI DE DATE IN LIMBAJUL SQL

a) TIPURI NUMERICE

3. numere zecimale: reprezentate cu precizia dorită :

- `decimal [(p,[s])]`-interval -10^38 +1÷ 10^38-1
- `numeric [(p[,s])]` echivalent cu tipul decimal

unde

- `p` (precizia) = valoare intre 1 ÷ 38, specifica nr. max de cifre zecimale atat la stanga cat si la dreapta punctului zecimal
- `s` (scala) = valoare intre 0 ÷ p, specifica nr. de zecimale, implicit =0

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE NUMERICE IN MYSQL

TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

TIPURI DE DATE IN LIMBAJUL SQL

b) TIPURI SIR DE CARACTERE

1. siruri de caractere de lungime fixa /variabila:

- `char([n)], character[(n)]` : şir de caractere de lungime fixă n caractere (n octeti), n in interval 1 ÷8000
- `varchar[(n)], character varying[(n)],` : şir de caractere de lungime variabilă, poate fi 0 sau max. n (max. n octeti)

OBS:

- Daca n nu este specificat implicit n=1
- Constantele sir pot fi declarate fie cu "" fie cu ' '. Ex.: "abc", 'abc'

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE CARACTER IN MYSQL

Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted.

TIPURI DE DATE IN LIMBAJUL SQL

c) TIPURI SIR DE BITI

2. siruri binare de lungime fixa sau variabila:

- `binary([n])`: data binara cu lungimea fixa n octeti, n in interval 1 ÷8000
- `varbinary[(n)]` : data binara cu lungimea variabila de max n octeti, n in interval 1 ÷8000, lungimea poate fi =0
- `image` : data binara cu lungimea variabila de max 2^31-1 octeti, pentru reprezentarea imaginilor

OBS:

- Daca n nu este specificat implicit n=1

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE IN LIMBAJUL SQL

d) TIPURI DATA CALENDARISTICA

1. Data calendaristica si ora:

- `datetime`: data si ora in intervalul 1.01.1753 ÷9999
- `smalldatetime`: data si ora in intervalul 1.01.1900 ÷6.06.2079

OBS:

- Constantele de tip data calendaristica se reprezinta intre apostroafe. Ex.: ‘04/15/98’, ‘15 April, 2010’

TIPURI DE DATE CALENDARISTICE IN MYSQL

DATE()	A date. Format: YYYY-MM-DD **Note:** The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MI:SS **Note:** The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. **Note:** Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE IN LIMBAJUL SQL

Exista diferente intre tipurile de date SQL în diferite sisteme SGBD.
Ex:

- Oracle: `varchar2` -şir de caractere de lungime variabilă (in loc de `varchar[(n)]`)

- Standardul SQL2 nu suportă tipuri de date şi operaţii definite de utilizator, spre deosebire de SQL3.
- In MS SQL Server, **se pot crea tipuri definite de utilizator** (user-defined types), echivalente cu domeniile create cu comanda SQL CREATE DOMAIN
- În Oracle si PostgreSQL **se pot crea tipuri de date noi**, folosind comanda CREATE TYPE, care permite gruparea sub un anumit nume a mai multor atribute, de diferite tipuri (predefinite sau definite de utilizator)

CONVENTII SINTACTICE IN LIMBAJUL SQL

[ ] (paranteze drepte)	Element opțional al instrucțiunii
{ } (acolade)	Element obligatoriu al instrucțiunii
| (bară verticală)	Separă elementele din parantezele drepte sau acolade; numai unul dintre acestea se poate introduce în instrucțiunea respectivă
[, . . . n]	Elementul precedent poate fi repetat de n ori; elementele repetate sunt separate prin virgulă
element1, ... elementn	Listă de n elemente de acelaşi tip; elementele repetate sunt separate prin virgulă
lista_elemente	Listă de elemente de acelaşi tip separate prin virgulă

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

STRUCTURA LEXICALA A LIMBAJULUI SQL

Componente

1) Tipuri de date ☑
2) **Cuvinte cheie**
3) Identificatori
4) Instrucţiuni/Comenzi
5) Variabile
6) Operanzi
7) Operatori
8) Functii

TIPURI DE DATE IN LIMBAJUL SQL

2) CUVINTE CHEIE

sunt elemente componente cu semnificaţie fixă si pot fi :

- **comenzi**: `SELECT, UPDATE, INSERT`, etc.
- **operatori**: `AND, OR, NOT, LIKE` , etc.
- **clauze**: `WHERE, SET, VALUES`, etc

Ex.: cuvinte cheie (bold)

```
SELECT * FROM SECTII WHERE IdSectie = 1;
INSERT INTO SECTII VALUES (2,'Productie',500);
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

STRUCTURA LEXICALA A LIMBAJULUI SQL

Componente

1) Tipuri de date ☑
2) Cuvinte cheie ☑
3) **Identificatori**
4) Instrucţiuni/Comenzi
5) Variabile
6) Operanzi
7) Operatori
8) Functii

TIPURI DE DATE IN LIMBAJUL SQL

3) IDENTIFICATORI

sunt elemente componente care denumesc tabele, coloane, etc. ale BD. Pot avea max. 30 caractere, cifre ,alte caractere (#,$,_, etc) , primul obligatoriu litera

Ex: `SELECT * FROM SECTII WHERE IdSectie = 1;`

`SECTII, IdSectie` sunt identificatori.

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

STRUCTURA LEXICALA A LIMBAJULUI SQL

Componente

1) Tipuri de date ☑
2) Cuvinte cheie ☑
3) Identificatori ☑
4) **Instrucţiuni/Comenzi**
5) Variabile
6) Operanzi
7) Operatori
8) Functii

TIPURI DE DATE IN LIMBAJUL SQL

4) INSTRUCTIUNI, COMENZI

❑ **Instrucţiune SQL** (statement): o secvenţă de elemente terminată cu **;** Fiecare instrucţiune SQL conţine o comanda.

❑ **Comandă SQL** (command): specifică ce acţiune se efectuează si operaţii, clauze, parametri etc.

Ex.: `SELECT * FROM ANGAJATI;`

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE IN LIMBAJUL SQL

4) INSTRUCTIUNI, COMENZI

Elementele (tokens) **instrucţiunilor SQL**:

❑ **cuvinte cheie rezervate** (key words): `CREATE,INSERT,SELECT,WHERE, FROM`

❑ **identificatori** (identifiers):

❖**simpli**: numai caractere alfa-numerice si "_": ANGAJATI, Nume, Prenume

❖**delimitati** (quoted):pot contine orice caracter, cu ghilimele : 'Nume', 'Prenume'

❑ **constante** (literal):valori fixe ce nu pot fi modificate (case senzitive ptr contante sir de caractere): 1000, 100.5, 'Ionescu', NULL

❑ **caractere speciale**: *, ., ;

❑ **spaţiile albe** (whitespaces) separa elementele: spaţiu, linie nouă, tab

STRUCTURA LEXICALA A LIMBAJULUI SQL

Componente

1) Tipuri de date ☑
2) Cuvinte cheie ☑
3) Identificatori ☑
4) Instrucţiuni/Comenzi ☑
5) **Variabile**
6) Operanzi
7) Operatori
8) Functii

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE IN LIMBAJUL SQL

5) VARIABILE

sunt date care pot avea in timp valori diferite. O variabila are intotdeauna un nume pentru a fi referita.

SQL accepta 2 tipuri de variabile:

- ❑ variabile asociate numelor coloanelor
- ❑ variabile sistem

STRUCTURA LEXICALA A LIMBAJULUI SQL

Componente

1) Tipuri de date ☑
2) Cuvinte cheie ☑
3) Identificatori ☑
4) Instrucţiuni/Comenzi ☑
5) Variabile ☑
6) **Operanzi**
7) Operatori
8) Functii

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE IN LIMBAJUL SQL

6) OPERANZI

Expresie SQL: **variabile**, **constante** unul/mai mulţi **operanzi**, **operatori** şi **paranteze** si /sau **functii**.

Operand poate fi:

- nume coloana : valoarea memorata in acea colona intr-una sau mai multe linii ale tabelului
- o constantă (literal)
- valoarea returnată de o functie

STRUCTURA LEXICALA A LIMBAJULUI SQL

Componente

1) Tipuri de date ☑
2) Cuvinte cheie ☑
3) Identificatori ☑
4) Instrucţiuni/Comenzi ☑
5) Variabile ☑
6) Operanzi ☑
7) **Operatori**
8) Functii

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE IN LIMBAJUL SQL

7) OPERATORI

Operator SQL :

- unul/mai multe caractere speciale; Ex: `+, -, *, /, %, <> (diferit), !=, =, !<(nu mai mic), !>(nu mai mare),` etc.
- un cuvânt cheie; Ex.: `AND, OR, NOT, LIKE` etc.

Clasificare operatori SQL dupa numarul de operanzi:

- binari
- unari

TIPURI DE DATE IN LIMBAJUL SQL

7) OPERATORI

Clasificare operatori SQL dupa operatii:

- Operatori **aritmetici de operatii** cu numere intregi / reale: `+,-, *, /, %,^`
- Operatori **aritmetici orientati pe biti**: `&, |`
- Operatori **aritmetici de comparatie**: `<,>,=,<>(sau !=),<=,>=, !<(nu mai mic decat), !>(nu mai mare decat)`
- Operatori **de comparatie SQL**:`IS NULL,IS NOT NULL,IN,LIKE, BETWEEN`
- Operatori **relationali**: `UNION, INTERSECT, MINUS`

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE IN LIMBAJUL SQL

OPERATORI COMPARATIE (IN , BETWEEN, IS NULL, IS NOT, LIKE)

Toti **operatorii de comparaţie** returneaza valori logice:

- **true** (1), dacă condiţia este îndeplinită
- **false** (0) dacă condiţia nu este îndeplinită
- **null** dacă ambii operanzi au valoarea null

Operator	Operaţia efectuată
`A BETWEEN min AND max`	compară A cu două valori, min şi max
`A IN (v1,v2, ...vn)`	compară A cu o lista de valori (v_1, v_2,...v_n)
`A IS NULL`	compară A cu NULL
`A IS NOT NULL`	compară A cu NOT NULL
`A LIKE model_sir`	compară A cu un model de şir de caractere

TIPURI DE DATE IN LIMBAJUL SQL

OPERATORI LOGICI (AND , OR, NOT)

- se aplică unor variabile logice trivalente (cu 3 valori: true (1), false (0) şi null - lipsa de informatie)
- returnează o valoare logică trivalentă

A	B	A and B	A or B	A	not A
true	true	true	true	true	false
true	false	false	true	false	true
true	null	null	true	null	null
false	false	false	false		
false	null	false	null		
null	null	null	null		

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE IN LIMBAJUL SQL

PRECEDENTA OPERATORILOR (descrescatoare)

- Operatori unari: +,-,~
- Operatori aritmetici multiplicativi: *,/,%
- Operatori aritmetici aditivi: +,-
- Operatori de comparare: =,<,>, <=,>=,<>, !=, !>, !<
- Operatori pe biti: &, |
- Operatorul de negatie logica: NOT
- Operatorul SI logic: AND
- Alti operatori logici: BETWEEN, IN, LIKE
- Operatorul de atribuire: =

STRUCTURA LEXICALA A LIMBAJULUI SQL

Componente

1) Tipuri de date ☑
2) Cuvinte cheie ☑
3) Identificatori ☑
4) Instrucţiuni/Comenzi ☑
5) Variabile ☑
6) Operanzi ☑
7) Operatori ☑
8) **Functii**

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

TIPURI DE DATE IN LIMBAJUL SQL

8) FUNCTII

- ❑ funcţii scalare
- ❑ funcţii agregat

A. Funcţiile scalare: se folosesc în expresii în clauzele instrucţiunilor SQL:

- ❑ returnează valoarea calculată sau NULL în caz de eroare
- ❑ argumentele pot fi constante/ valori ale atributelor specificate prin numele coloanelor corespunzatoare

B. Funcţiile agregat: calculează un rezultat din mai multe linii ale unui tabel

FUNCTII IN LIMBAJUL SQL

A. FUNCTII SCALARE

a. **funcţii numerice**

b. **funcţii pentru şiruri de caractere**

c. **funcţii pentru data calendaristică şi timp**

d. **funcţii de conversie**

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

FUNCTII IN LIMBAJUL SQL

A. FUNCTII SCALARE

a) **FUNCTII NUMERICE**

- `ABS(x):` modulul
- `CEILING(x):` rotunjire (prin adăugare)
- `FLOOR(x):` rotunjire (prin trunchiere)
- `MOD(x,y):` rezultatul împărţirii intregi x la y
- `POWER(x,y):` x la puterea y

FUNCTII IN LIMBAJUL SQL

A. FUNCTII SCALARE

a) FUNCTII NUMERICE -continuare

- `ROUND(x,y)`: rotunjeşte pe x la y zecimale
- `TRUNC(x,y)`: trunchează pe x la y zecimale
- `SIGN(x)`: returnează -1 dacă x e negativ, 1 dacă x e pozitiv şi 0 dacă x este nul
- `SQRT(x)`: rădăcina pătrată; `SQUARE(x)`: x pătrat (x^2);
- `LN(x)`: logaritm natural
- `EXP(x)`: exponentul numărului x (e^x)

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

FUNCTII IN LIMBAJUL SQL

A. FUNCTII SCALARE

b) FUNCTII PENTRU SIRURI DE CARACTERE

- `LEFT(<string>,x)`: extrage cele mai din stanga x caractere din sir
- `RIGHT(<string>,x)`: extrage cele mai din dreapta x caractere din sir
- `UPPER(<string>)`: converteşte in litere mari
- `LOWER(<string>)`: converteşte in litere mici
- `LEN(<string>)`: returnează numărul de caractere din şir
- `<string>||<string>`: concatenează cele două şiruri, primul şir fiind urmat de al doilea

FUNCTII IN LIMBAJUL SQL

A. FUNCTII SCALARE

b) FUNCTII PENTRU SIRURI DE CARACTERE -continuare

- `LPAD(<string>,x,'*')`: completează şirul la stânga cu caracterul dintre apostroafe până ce şirul ajunge la lungimea de x caractere
- `RPAD(<string>,x,'*')`: completează şirul la dreapta cu caracterul dintre apostroafe până ce şirul ajunge la lungimea de x caractere
- `SUBSTR(<string>,x,y)`: extrage y caractere din şir începând cu poziţia x
- `NVL(<column>,<value>)`: înlocuieşte toate valorile nule din coloana specificată cu valoarea dată
- `CONCAT(<string1>,<string2>)`: concatenează cele două şiruri
- `INSTR(<string>,ch)`: returnează poziţia caracterului ch în şir

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

FUNCTII IN LIMBAJUL SQL

A. FUNCTII SCALARE

b) FUNCTII PENTRU SIRURI DE CARACTERE -continuare

- `REPLACE(<string>,a,b)`: înlocuieşte în şirul de caractere secvenţa a prin secvenţa b iar dacă b lipseşte are loc ştergerea lui a
- `LTRIM(<string>)`: şterge spaţiile libere din stânga şirului de caractere
- `RTRIM(<string>)`: şterge spaţiile libere din dreapta şirului de caractere
- `TO_NUMBER(string,['format'])`: realizează conversia din şir de caractere în numeric

FUNCTII IN LIMBAJUL SQL

A. FUNCTII SCALARE

c) FUNCTII PENTRU DATE CALENDARISTICE SI TIMP

- `MONTHS_BETWEEN(data1,data2)`: returneaza numărul de luni între cele 2 date (pozitiv sau negativ)
- `ADD_MONTHS(data1,n)`: adaugă un număr de luni n la o dată data1
- `NEXT_DAY(data1,string)`: returneaza următoarea dată corespunzătoare zilei specificate in data1
- `LAST_DAY(data1)`: returneaza ultima zi a lunii corespunzătoare datei specificate

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

FUNCTII IN LIMBAJUL SQL

A. FUNCTII SCALARE

c) FUNCTII PENTRU DATE CALENDARISTICE SI TIMP -continuare

- `DAY(data)`: returneaza un intreg =zi din data
- `GETDATE()`: returneaza data curenta din sistem in format standard
- `MONTH(data)`: returneaza un intreg =luna din data
- `YEAR(data)`: returneaza un intreg = an din data

FUNCTII IN LIMBAJUL SQL

B. FUNCTII AGREGAT

Se utilizeaza cu SELECT:

- **AVG():** returneaza media valorilor dintr-o coloana
- **COUNT():** returneaza nr randurilor dintr-o coloana
- **MAX():** returneaza val max dintr-o coloana
- **MIN():** returneaza val min dintr-o coloana
- **SUM():** returneaza suma val dintr-o coloana

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

FUNCTII IN LIMBAJUL SQL

TEST-Kahoot

INSTRUCTIUNI IN LIMBAJUL SQL

Instructiunile SQL permit:

- interogari in BD
- returnare valori din BD
- inserare articole in BD
- modificare valori ale articolelor din BD
- stergere articole din BD
- crearea de noi BD si noi tabele in BD
- crearea de proceduri stocate in BD
- creearea de vederi (view) in BD
- stabilirea de permisiuni asupra tabelelor, vederilor si procedurilor, etc.

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI IN LIMBAJUL SQL

LMD	Pentru manipularea datelor	SELECT	Extragerea datelor din baza de date
		INSERT	Adăugarea de noi linii într-un tabel
		DELETE	Stergerea de linii dintr-un tabel
		UPDATE	Modificarea valorilor unor atribute
LDD	Pentru definirea bazei de date	CREATE TABLE	Adăugarea unui nou tabel la baza de date
		DROP TABLE	Stergerea unui tabel din baza de date
		ALTER TABLE	Modificarea structurii unei baze de date
		CREATE VIEW	Crearea unuitabel virtual (vedere)
		DROP VIEW	Stergerea unui tabel virtual
LCD	Pentru controlul accesului la baza de date	GRANT	Acordarea unor drepturi pentru utilizatori
		REVOKE	Revocarea unor drepturi pentru utilizatori
	Pentru controlul tranzacțiilor	COMMIT	Marchează sfârşitul unei tranzacții
		ROLLBACK	Abandonează tranzacția în curs.

INSTRUCTIUNI IN LIMBAJUL SQL

DEFINITII

1. Componenta de definire a datelor din SQL
(LDD – Limbajul de Definire a Datelor):

- **crearea** (CREATE) obiectelor BD
- **modificarea** (ALTER) obiectelor BD
- **distrugerea** (DROP) obiectelor BD

Obiectele BD:

- tabele de bază (TABLE),
- tabele vedere (VIEW),
- indexuri (INDEX),
- proceduri (PROCEDURE),
- trigere (TRIGGER),
- utilizatori (USER)

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI IN LIMBAJUL SQL

DEFINITII

Ex: Instructiuni (comenzi) SQL de definire a datelor:

```
CREATE TABLE, CREATE VIEW, CREATE INDEX, CREATE USER
CREATE FUNCTION, CREATE TRIGGER, CREATE PROCEDURE

ALTER TABLE, ALTER VIEW, ALTER FUNCTION, ALTER PROCEDURE

DROP TABLE, DROP VIEW, DROP INDEX, DROP USER
DROP FUNCTION, DROP PROCEDURE, DROP TRIGGER
```

INSTRUCTIUNI IN LIMBAJUL SQL

DEFINITII

2. Componenta de manipulare a datelor din limbajul SQL (Limbajul de Manipulare a Datelor -LMD)

Ex: Instructiuni (comenzi) SQL de manipulare a datelor:

```
SELECT, INSERT, UPDATE si DELETE
```

Instructiunile SQL se transmit SGBD-ului:

- de catre diferite programe client (client grafic, linie de comanda, program executabil)
- SGBD-ul executa instructiunea SQL si returneaza un raspuns (rezultatul operatiei sau un cod de eroare)

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI SQL DE DEFINIRE A DATELOR

Creearea BD: CREATE DATABASE

DEFINITII

Baza de date (data base): se creeaza cu **CREATE DATABASE** , avand privilegii de administrator

Sintaxa

```
CREATE DATABASE NUMEBD ;
```

Afisarea BD existente:

```
SHOW DATABASES;
```

EXEMPLU

Ex:

```
CREATE DATABASE TEST;
```

INSTRUCTIUNI SQL DE DEFINIRE A DATELOR

Stergerea BD: DROP DATABASE

DEFINITII

Baza de date (data base): se sterge cu **DROP DATABASE** , avand privilegii de administrator

Sintaxa:

```
DROP DATABASE NUMEBD ;
```

Verificarea stergerii BD :

```
SHOW DATABASES;
```

EXEMPLU

Ex:

```
DROP DATABASE TEST;
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI SQL DE DEFINIRE A DATELOR

Creearea tabelelor : CREATE TABLE

DEFINITII

Tabelele de baza (base table): se creeaza cu **CREATE TABLE** sunt memorate în fişierele bazei de date şi pot fi accesate pentru introducerea, modificarea şi regăsirea (interogarea) datelor:

Sintaxa

```
CREATE TABLE nume_tabel (
col1 domeniu1 [constrangeri_coloana],
col2 domeniu2 [constrangeri_coloana],
 . . . . . .
coln domeniun [constrangeri_coloana],
[constrangeri_tabel] );
```

Constrângerile impuse fiecărei coloane (atribut) /de tabel, sunt opţionale.

INSTRUCTIUNI DE DEFINIRE A DATELOR

Creearea tabelelor : CREATE TABLE

EXEMPLU

Ex:

```
CREATE TABLE Angajati(
Nume varchar(20),
Prenume varchar(20),
DataNasterii date,
Adresa varchar(50),
Functie varchar(20),
Salariu numeric);
```

Instrucţiunea CREATETABLE defineşte atât tipul relaţiei cât şi o variabilă relaţie care iniţial este vidă (nu conţine nici un tuplu)

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI SQL DE DEFINIRE A DATELOR

Creearea tabelelor pe baza tabelelor existente

DEFINITII

Sintaxa

```
CREATE TABLE new_table_name AS
       SELECT column1, column2,...
       FROM existing_table_name
       WHERE ....;
```

EXEMPLU

Ex:

```
CREATE TABLE Contab AS
   SELECT Nume , Prenume
   FROM  Angajati;
```

INSTRUCTIUNI DE DEFINIRE A DATELOR

Creearea view-uri : CREATE VIEW

DEFINITIE

Tabel view : tabel virtual care reprezintă o selecţie (după un anumit criteriu) a datelor memorate în unul sau mai multe tabele de bază

Sintaxa: `CREATE VIEW nume_vedere AS (SELECT...);`

- Datele (valorile atributelor) sunt memorate o singură dată, în tabelele de bază, dar pot fi accesate atât prin tabelele de bază cât şi prin tabelele view
- Un tabel view este întotdeauna actualizat ("la zi"), adică orice modificare efectuată în tabelele de bază se regăseşte imediat în orice tabel view creat pe baza acestora

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE DEFINIRE A DATELOR

Modificarea tabelelor: ALTER TABLE

DEFINITIE

`ALTER TABLE` : instructiune de modificare a tabelelor permite:

- adăugarea sau ştergerea unor atribute
- modificarea domeniilor unor atribute
- adăugarea, modificarea sau ştergerea unor constrângeri ale tabelului

INSTRUCTIUNI DE DEFINIRE A DATELOR

Modificarea tabelelor: ADAUGARE COLOANA

Adaugare coloana in tabel

Sintaxa:

```
ALTER TABLE nume_tabel
ADD nume_coloana domeniu;
```

EXEMPLE

Ex.: adaugarea unei coloane intr-un tabel cu clauza **ADD**

```
ALTER TABLE Angajati ADD DataAngajarii date;
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE DEFINIRE A DATELOR

Modificarea tabelelor: STERGERE COLOANA

DEFINITIE

Stergere coloana din tabel

Sintaxa:

```
ALTER TABLE nume_tabel
DROP nume_coloana domeniu;
```

EXEMPLE

Ex.: ştergerea unei coloane dintr-un tabel cu clauza **DROP**

```
ALTER TABLE Angajati DROP DataAngajarii;
```

INSTRUCTIUNI DE DEFINIRE A DATELOR

Modificarea tabelelor: SCHIMBARE TIP COLOANA

DEFINITIE

Modificare tip date in coloana din tabel

Sintaxa:

```
ALTER TABLE nume_tabel
ALTER COLUMN nume_coloana domeniu;
```

EXEMPLE

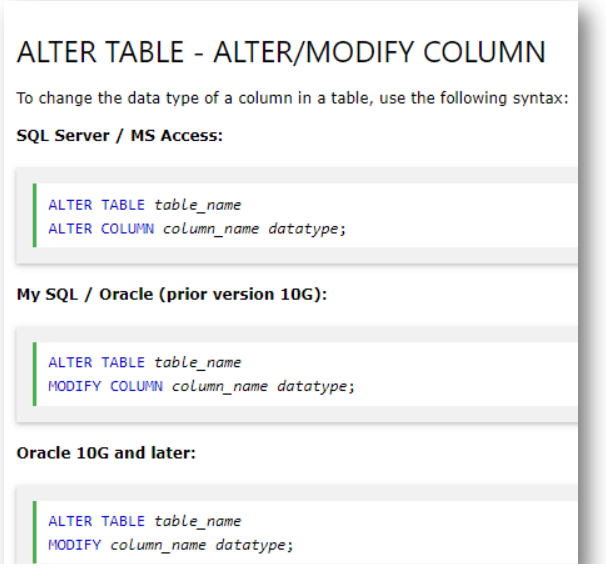
Ex.: modificarea unei coloane dintr-un tabel cu clauza ALTER COLUMN din data calendaristica (date) in an (year)

```
ALTER TABLE Angajati
ALTER COLUMN DataAngajarii year;
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE DEFINIRE A DATELOR

Modificarea tipului coloanei din tabel: ALTER TABLE



ALTER TABLE - ALTER/MODIFY COLUMN

To change the data type of a column in a table, use the following syntax:

SQL Server / MS Access:

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

My SQL / Oracle (prior version 10G):

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

Oracle 10G and later:

```
ALTER TABLE table_name
MODIFY column_name datatype;
```

INSTRUCTIUNI DE DEFINIRE A DATELOR

Stergerea tabelelor/view-urilor: DROP TABLE/VIEW

DEFINITIE

DROP TABLE, DROP VIEW: instructiuni de stergere a tabelelor de baza si respectiv a view-urilor .
TRUNCATE: instructiune care sterge articolele din BD dar nu sterge tabelul
Sintaxa:

```
DROP TABLE nume_tabel; TRUNCATE TABLE nume_tabel;
DROP VIEW nume_vedere;
```

EXEMPLE

Ex.1.: stergerea unui tabel :

```
DROP TABLE Angajati; TRUNCATE TABLE Angajati;
```

Ex.2. stergerea unui view:

```
DROP VIEW TABEL;
```


INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT

DEFINITIE

SELECT : instrucţiune de interogare prin care se regăsesc informaţiile din unul sau mai multe tabele ale BD dupa un criteriu (conditie) dat

Sintaxa :

```
SELECT [DISTINCT] lista_coloane
[FROM lista_tabele]
[WHERE conditie]
[clauze_secundare];
```

SELECT returneaza un tabel cu coloanele din “lista_coloane” ale acelor linii (tupluri) ale produsului cartezian al tabelelor din “lista_tabele” pentru care expresia logică “conditie” este adevărată (TRUE)

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT

DEFINITIE

`SELECT` are urmatoarele **clauze**:

- `SELECT` defineşte lista de coloane a tabelului rezultat
- `FROM` indică lista de tabele din care se selectează rezultatul
- `WHERE` defineşte condiţia pe care trebuie să o îndeplinească fiecare linie a tabelului rezultat

Clauze secundare permit ordonări sau grupări ale tuplurilor (liniilor) rezultate:

- `ORDER BY, GROUP BY, HAVING`

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT

DEFINITIE

`SELECT` specifica:

- lista coloanelor unor tabele (date in "lista_tabele")
- expresii care vor fi calculate şi afişate

EXEMPLU

Ex.:
```
SELECT ID, Name, CountryCode FROM CITY;
SELECT 3*4, cos(45), floor(12.45);
```

DEFINITIE

`SELECT` cu parametrul `DISTINCT` : eliminarea liniilor duplicat

EXEMPLU

Ex.:
```
SELECT DISTINCT CountryCode FROM CITY;
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT

EXEMPLE

Ex.1: Selectarea tuturor coloanelor din tabelul "CITY".

```
SELECT * FROM CITY;
```

Ex.2.: În clauza SELECT se pot redenumi coloanele tabelelor sau se pot specifica expresii pentru nume de coloane, cu sintaxa:

```
SELECT nume1 [AS] nume1_nou [,...n] FROM lista_tabele [alte_clauze];
SELECT ID, Nume Oras, CountryCode AS 'Cod Tara' FROM CITY;
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. Clauzele FROM si WHERE

DEFINITIE

Clauza FROM : specifica "lista_tabele"din care se selecteaza rezultatul.

- Numele coloanelor din "lista_coloane" (SELECT) trebuie să fie distincte.
- Dacă nu sunt distincte, se utilizeaza unele coloane cu numele tabelului caruia îi aparţin -folosind operatorul "."

EXEMPLU

Ex.: `SELECT ANGAJATI.Nume, SECTII.Nume FROM ANGAJATI, SECTII;`

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. Clauzele FROM si WHERE

DEFINITIE

Clauza WHERE: specifica "**conditia**" pe care trebuie sa o îndeplinesca rezultatul

- **conditia** este o expresie logică compusa din valori logice, operatori logici (NOT, AND, OR) şi paranteze
- o valoare logică se obtine ca rezultat al comparaţiei între doi operanzi folosind un operator de comparaţie
- un operand poate fi un atribut (nume de coloană), o constantă, valoarea unei expresii aritmetice sau o valoare returnată de o funcţie

EXEMPLU

Ex.:

```
SELECT * FROM CITY WHERE Populatie > 1000;
SELECT * FROM CITY WHERE Populatie BETWEEN 1000 AND
100000 AND CountryCode='RO';
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. Operatorul LIKE

DEFINITIE

LIKE: se utilizeaza cu clauza WHERE si specifica paternul care se cauta cu SELECT in coloana specificata. Poate contine caracterele % si _

Sintaxa:

```
SELECT column1, column2, ... FROM nume_tabel
 WHERE columnN LIKE pattern;
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. Operatorul LIKE

EXEMPLE

Operatorul LIKE	Descriere
WHERE Nume LIKE 'a%'	Gaseste orice valoare care incepe cu litera "a"
WHERE Nume LIKE '%a'	Gaseste orice valoare care se termina cu litera "a"
WHERE Nume LIKE '%or%'	Gaseste orice valoare care are "or" in orice pozitie
WHERE Nume LIKE '_r%'	Gaseste orice valoare care are "r" in a doua pozitie
WHERE Nume LIKE 'a_%_%'	Gaseste orice valoare care incepe cu litera "a" si are cel putin 3 caractere lungime
WHERE Nume LIKE 'a%o'	Gaseste orice valoare care incepe cu litera "a" si se termina cu "o"

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. OPERATORUL LIKE

EXEMPLU

Ex.1: selecteaza toate articolele din tabelul Angajati cu toate coloanele pentru care valoare din coloana Nume incepe cu litera A

```
SELECT * FROM Angajati WHERE Nume LIKE ‘A%';
```

Ex.2: selecteaza toate articolele din tabelul Angajati cu toate coloanele pentru care valoarea din coloana Nume NU incepe cu litera A

```
SELECT * FROM Angajati WHERE Nume NOT LIKE ‘A%';
```

Ex.3: selecteaza toate articolele din tabelul Angajati cu toate coloanele pentru care valoarea din coloana Nume incepe cu “Popa”

```
SELECT * FROM Angajati WHERE Nume LIKE 'Popa%';
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. OPERATORUL LIKE

EXEMPLU

Ex.4: selecteaza toate articolele din tabelul Angajati cu toate coloanele pentru care valoarea din coloana Nume incepe cu orice litera si se termina cu “ascu”

```
SELECT * FROM Angajati WHERE Nume LIKE '_ascu';
```

Ex.5: selecteaza toate articolele din tabelul Angajati cu toate coloanele pentru care valoarea din coloana Prenume incepe cu litera A, P sau V

```
SELECT * FROM Angajati WHERE Prenume LIKE '[APV]%';
```

Ex.6: selecteaza toate articolele din tabelul Angajati cu toate coloanele pentru care valoarea din coloana Prenume NU incepe cu una din literele A, P sau V

```
SELECT * FROM Angajati WHERE Prenume LIKE '[!APV]%';
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. Operatorul LIKE

DEFINITIE

IN: se utilizeaza cu clauza WHERE pentru a explicita conditii multiple SAU

Sintaxa:

```
SELECT coloane FROM nume_tabel WHERE coloana IN (value1, value2,
  ...);
```

EXEMPLU

Ex.1: selecteaza toate articolele din tabelul Angajati cu toate coloanele pentru care valoare din coloana Nume incepe este unul din lista:

```
SELECT * FROM Angajati WHERE Nume IN (‘Popa’,’Pop’,’Popescu’);
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. Operatorul BETWEEN

DEFINITIE

BETWEEN: selecteaza valorile (pot fi date numerice, sir caractere si date calendaristice) dintr-un interval:

Sintaxa:

```
SELECT coloane FROM nume_tabel
  WHERE coloana BETWEEN value1 AND value2;
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. Operatorul BETWEEN/NOT BETWEEN

EXEMPLU

Ex.1: selecteaza toate articolele din tabelul CITY cu toate coloanele pentru care valoarea din coloana Populatie este in intervalul [10000,200000]:

```
SELECT * FROM CITY WHERE Populatie BETWEEN 10000 AND 200000;
```

EXEMPLU

Ex.1: selecteaza toate articolele din tabelul CITY cu toate coloanele pentru care valoarea din coloana Populatie NU este in intervalul [10000,200000]:

```
SELECT * FROM CITY WHERE Populatie NOT BETWEEN 10000 AND 200000;
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. Aliases

DEFINITIE

SQL aliases: se utilizeaza pentru a redenumi temporar un tabel sau o coloana pe durata unei interogari cu SELECT

Sintaxa alias coloana:

```
SELECT coloana AS alias_coloana FROM nume_tabel;
```

Sintaxa alias tabel:

```
SELECT coloana FROM nume_table AS alias_tabel;
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. Aliases

EXEMPLU

Ex.1: selecteaza toate articolele din tabelul CITY cu toate coloanele pentru care valoarea din coloana Populatie este in intervalul [10000,200000]:

```
SELECT Coutry AS Tara, Populatie AS Nr_locuitori FROM CITY;
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. Clauzele FROM si WHERE

Ce instructiune SQL permite selectarea tuturor coloanelor din tabelul numit "Persoane"?

a) `SELECT Persoane`

b) `SELECT *.Persoane`

c) `SELECT [all] FROM Persoane`

d) `SELECT * FROM Persoane`

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. Clauzele FROM si WHERE

Ce instructiune SQL selecteaza toate articolele dintr-un tabel numit "Persoane" pentru care coloana "Prenume" este "Petre" iar coloana "Nume" este "Ionescu"?

a) `SELECT Prenume ='Petre' AND Nume='Ionescu' FROM Persoane`

b) `SELECT * FROM Persoane WHERE Prenume ='Petre' AND Nume='Ionescu'`

c) `SELECT * FROM Persoane WHERE Prenume<>'Petre' AND Nume<>'Ionescu'`

d) `SELECT FROM Persoane WHERE Prenume="Petre" AND Nume="Ionescu"`

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT . Functii agregat

DEFINITIE

În SELECT se pot introduce şi **funcţii agregat** (totalizatoare):

Functia	Valoarea returnata
COUNT	Numarul de linii al tabelului rezultat
SUM	Suma valorilor din coloana data ca argument
MAX	Valoarea maxima din coloana data ca argument
MIN	Valoarea minima din coloana data ca argument
AVG	Valoarea medie din coloana data ca argument

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT . Functii agregat

EXEMPLE

Ex: 1. COUNT() returneaza numarul de linii din tabelul CITY

```
SELECT COUNT(*) FROM CITY;
```

Ex. 2. COUNT() returneaza nr de linii distincte din coloana Country a tabelului CITY

```
SELECT COUNT(DISTINCT Country) FROM CITY;
```

Ex. 3. MAX() si MIN() returneaza valoarea maxima si respectiv minima din coloana Populatie a tabelului CITY

```
SELECT MAX(Populatie) FROM CITY;
SELECT MIN(Populatie) FROM CITY;
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT . Functii agregat

EXEMPLE

Ex. 4. AVG() returneaza media artimetica a valorilor din coloana Populatie a tabelului CITY:

```
SELECT AVG(Populatie) FROM CITY;
```

Ex. 5. SUM() returneaza suma valorilor din coloana Populatie a tabelului CITY:

```
SELECT SUM(Populatie) FROM CITY;
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT . Clauze secundare

DEFINITIE

În SELECT se pot introduce şi clauze secundare impreuna cu **funcţii agregat** (totalizatoare): GROUP BY , ORDER BY, HAVING

Implicit ordonarea se face ascendent (ASC)

Sintaxa :

```
SELECT lista_coloane
FROM nume_tabel
WHERE conditie
GROUP BY lista_coloane
HAVING conditie
ORDER BY lista_coloane ASC|DESC;
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT . Clauze secundare

EXEMPLE

Ex. 1. GROUP BY selecteaza si afiseaza lista cu numarul clientilor (CustomerID) din fiecare tara (Country):

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT . Clauze secundare

EXEMPLE

Ex. 2. ORDER BY selecteaza si afiseaza tarile din coloana Country din tabela CITY ordonate alfabetic ascendent dupa coloana Country, sau dupa Country si apoi Town:

```
SELECT * FROM CITY ORDER BY Country;
SELECT * FROM CITY ORDER BY Country, Town;
```

Ex. 3. ORDER BY selecteaza si afiseaza tarile din coloana Country din tabela CITY ordonate alfabetic descrescator dupa coloana Country:

```
SELECT * FROM CITY ORDER BY Country DESC;
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT . Clauze secundare

EXEMPLE

Ex. 4. GROUP BY si ORDER BY selecteaza si afiseaza nr clientilor grupati pe fiecare tara si ordonati descrescator de la cel mai mare numar la cel mai mic:

```
SELECT COUNT (CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT . Clauze secundare

EXEMPLE

Ex. 5. GROUP BY si HAVING selecteaza si afiseaza nr clientilor grupati pe fiecare tara si doar pentru tarile care indeplinesc conditia ca nr clientilor >5:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT . IS NULL/NOT NULL

EXEMPLE

Ex. 1. SELECT poate determina daca valoarea dintr-o coloana este NULL , dar nu prin intermediul operatorilor =,>,<, etc ci prin IS NULL sau IS NOT NULL:

```
SELECT coloane
FROM nume_tabel
WHERE coloana IS NULL;
sau
WHERE coloana IS NOT NULL;
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : SELECT. Clauza SELECT. Functii agregat

Care dintre instructiunile SQL de mai jos are ca efect returnarea numarului de articole din tabelul "Persoane" ?

```
a) SELECT COUNT(*) FROM Persoane
b) SELECT * FROM Persoane
c) SELECT COUNT() FROM Persoane
d) SELECT SUM(*) FROM Persoane
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : INSERT

DEFINITIE

INSERT :se utilizeaza pentru introducerea datelor în tabele. Între valori şi numele de coloane trebuie să existe o corespondenţă pozitionala.

Sintaxă:

```
INSERT INTO nume_tabel(col1,col2,..coln) VALUES(val1,val2,...valn);
```

EXEMPLU

Ex.:

```
INSERT INTO SECTII(Numar,Nume,Buget) VALUES (1,'Productie',40000);
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : INSERT

Lista de coloane poate să lipsească dacă se introduc valori în toate coloanele tabelului:

- **ordinea valorilor trebuie să respecte ordinea coloanelor tabel**ului
- ordinea coloanelor provine din ordinea de definire a atributelor prin instrucţiunea `CREATE TABLE`, şi din operaţiile ulterioare cu ALTER
- ordinea coloanelor se poate afla prin instrucţiunea `DESCRIBE nume_tabel`.

EXEMPLU

Ex.: introducerea unui articol (linie) în tabelul ANGAJATI (IdAngajat, Nume, Prenume, DataNasterii, Adresa,Functia, Salariu) :

```
INSERT INTO ANGAJATI VALUES(100,'Mihailescu','Mihai', '1950-04-05',
'Craiova','Inginer', 3000);
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea : INSERT

Care instructiune SQL insereaza sirul "Muresan" in campul "Nume" al tabelului "Persoane"?

a) `INSERT ('Muresan') INTO Angajati (Nume)`

b) `INSERT INTO Persoane (Nume) VALUES ('Muresan')`

c) `INSERT INTO Persoane ('Muresan') INTO Nume`

d) `INSERT INTO Persoane VALUES ('Muresan')`

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea: UPDATE . Clauza: WHERE

DEFINITIE

UPDATE : permite actualizarea valorilor coloanelor (atributelor) din una sau mai multe linii ale unui tabel

Sintaxa:

```
UPDATE nume_tabel SET col1 = expr1 [, . . . n] [WHERE conditie];
```

Clauza WHERE: actualizarea valorilor se efectueaza numai asupra acelor linii care îndeplinesc condiţia dată. **Dacă este omisă clauza WHERE, vor fi modificate valorile coloanelor din toate liniile tabelului.**

EXEMPLU

Ex.: actualizarea liniilor din tabelul ANGAJATI pentru care Numele este Popescu

```
UPDATE ANGAJATI SET Adresa = 'Bucuresti' WHERE Nume = 'Popescu';
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea: UPDATE . Clauza: WHERE

Care dintre instructiunile SQL de mai jos are ca efect modificarea coloanei "Nume" in "Popescu" pentru toate articolele din tabelul "Persoane" pentru care coloana "Nume" are valoarea initiala "Popa" ?

a) `UPDATE Persoane SET Nume='Popescu' INTO Nume=Popa`

b) `MODIFY Persoane SET Nume='Popescu' INTO Nume='Popa'`

c) `MODIFY Persoane SET Nume='Popa' WHERE Nume='Popescu'`

d) `UPDATE Persoane SET Nume='Popescu' WHERE Nume='Popa'`

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea: DELETE . Clauza: WHERE

DEFINITIE

DELETE: permite ştergerea uneia sau mai multor linii dintr-un tabel

Sintaxa: `DELETE FROM nume_tabel [WHERE conditie];`

Din tabel se şterg acele linii (articole) care îndeplinesc condiţia dată în clauza WHERE. **Dacă este omisă clauza WHERE, vor fi sterse toate liniile din tabel.**

EXEMPLU

Ex.:

```
DELETE FROM ANGAJATI WHERE Nume ='Ionescu';
```

INSTRUCTIUNI DE MANIPULARE A DATELOR

Instructiunea: DELETE . Clauza: WHERE

Care dintre instructiunile SQL de mai jos sterge toate articolele dintr-un tabel numit "Persoane" pentru care coloana "Prenume" este "Petre"?

a) `DELETE Prenume='Petre ' FROM Persoane`

b) `DELETE ROW Prenume='Petre' FROM Persoane`

c) `DELETE FROM Persoane WHERE Prenume='Petre'`

d) `DELETE * FROM Persoane WHERE Prenume<>Petre`

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

INSTRUCTIUNI DE MANIPULARE A DATELOR

TEST-Kahoot

4. Constrangeri de integritate a relatiilor

DEFINITIE

Constrângerile de integritate (integrity constraints) sunt reguli care:
- ❑ se definesc la proiectarea BD
- ❑ trebuie să fie respectate astfel incat datele memorate să corespundă cat mai bine celor din realitate

CLASIFICARI

a. **Clasificare dupa locul unde se definesc**
b. **Clasificare dupa nr de relatii implicate**
c. **Clasificare dupa modul de definire**

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

4. Constrangeri de integritate a relatiilor

CLASIFICARI

a. Clasificare dupa locul unde se definesc:
- ❑ **constrangeri de coloana**
- ❑ **constrangeri de tabel** (CREATE TABLE)

EXEMPLU

Ex.:

```
CREATE TABLE ANGAJATI (
Nume varchar(20) NOT NULL,
Prenume varchar(20) NOT NULL,
DataNasterii date NULL,
Adresa varchar(50) NOT NULL,
Functie varchar(20),
Salariu numeric);
```

4. Constrangeri de integritate a relatiilor

CLASIFICARI

b. Clasificare dupa nr de relatii implicate:

❑ **constrângeri intra-relaţie**: reguli care se impun în cadrul unei singure relaţii; sunt 3 categorii:

- **Constrângeri de domeniu** -condiţiile se impun valorilor din domenii
- **Constrângeri de tuplu**-condiţiile se impun tuplurilor unei relaţii prin **chei (primare sau secundare).**
- **Constrângeri prin dependenţe de date** (dependenţe funcţionale, multivalorice,de joncţiune)-constrângeri intre valorile atributelor relaţiei

❑ **constrângeri inter-relaţii**: reguli care se impun între 2 sau mai multe relaţii; asigura integritarea referenţială prin intermediul **cheilor străine (foreign keys)**

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

4. Constrangeri de integritate a relatiilor

CLASIFICARI

c. Clasificare dupa modul de definire:

❑ **Constrângerile inerente:** constrangeri ale modelului de date însuşi, care nu trebuie să fie specificate la definirea relaţiilor, dar sunt respectate prin modul în care se construiesc relaţiile

Ex.: valoarea fiecărui atribut să fie atomică (indivizibilă)=constrângere inerentă

❑ **Constrângerile implicite:** reguli care se definesc odată cu definirea schemelor relaţiilor, sunt memorate în BD şi SGBD verifică şi impune automat respectarea lor

Ex.: constrângerile de domeniu, de tuplu şi de integritate referenţială =constrângeri implicite.

❑ **Constrângerile explicite**: constrângeri suplimentare pe care trebuie să le respecte relaţiile unei BD care nu sunt impuse automat de SGBD, ci necesită proceduri speciale de verificare şi impunere

Ex.: constrangeri de date care nu sunt determinate de cheile relaţiilor

CONSTRÂNGERI INTRA-RELATIE: DE DOMENIU

Constrângerile de domeniu:

- **constrangerea `NULL`**: valoarea atributului nu este cunoscută pentru acel tuplu.
- **constrângerea `NOT NULL`:** atributul nu poate lua valoarea `NULL` în nici un tuplu al relaţiei.
- **constrângerea de valoare implicită `DEFAULT`**
- **constrângerea de verificare `CHECK`**

Ex: constrangerea NULL

- nu se cunoaste data de nastere a unei personalitati istorice;
- nu se cunoaşte valoarea unui atribut in momentul inserarii tuplului, dar aceasta va fi cunoscuta si completată ulterior

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

CONSTRÂNGERI INTRA-RELATIE: DE DOMENIU

Constrangerea `NULL`

- La crearea unui tabel:
  - opţiunea NULL este implicită dar se poate introduce explicit;
  - optiunea NOT NULL se introduce explicit.
- Optiunile NULL si NOT NULL se introduc ca si constrangeri de coloana in instructiunea SQL : CREATE TABLE.

EXEMPLU

Ex.:

```
CREATE TABLE ANGAJATI (
Nume varchar(20) NOT NULL,
Prenume varchar(20) NOT NULL,
DataNasterii date NULL,
Adresa varchar(50) NOT NULL,
Functie varchar(20),
Salariu numeric);
```

CONSTRÂNGERI INTRA-RELATIE: DE DOMENIU

DEFINITIE

Constrangerea DEFAULT : specifica **valoarea implicita** a unui atribut.
Daca la inserarea unui tuplu nu se specifică valoarea unui atribut, atunci:

- atributul primeşte valoarea implicită (DEFAULT) dacă a fost definită, sau valoarea NULL
- dacă nu a fost definită o valoare implicită şi nici nu sunt admise valori NULL, se generează o eroare.

EXEMPLU

Ex.:

```
CREATE TABLE STUDENTI (
Nume varchar (20) NOT NULL,
Prenume varchar (20) NOT NULL,
Tara varchar (20) DEFAULT ('Romania') NULL ) ;
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

CONSTRÂNGERI INTRA-RELATIE: DE DOMENIU

DEFINITIE

Constrângerea CHECK : pentru verificarea valorilor atributelor printr-o conditie care trebuie sa ia valoarea TRUE.

Se introduce ca o constrangere de tabel in instructiunea CREATE TABLE:

```
[CONSTRAINT nume_constrangere] CHECK (conditie);
```

EXEMPLU

Ex.:

```
CREATE TABLE ANGAJATI (
Nume varchar(20) NOT NULL,
Prenume varchar(20) NOT NULL,
Salariu numeric,
CONSTRAINT Verificare_Salariu CHECK (Salariu >= 700 ));
```

CONSTRÂNGERI INTRA-RELATIE: DE TUPLU

DEFINITIE

Relaţie = mulţime de tupluri

Tuplurile unei relaţii **trebuie să fie distincte** (nu pot exista două sau mai multe tupluri identice)

Supercheie (superkey=SK) este o submulţime SK de atribute ale relaţiei care prezintă **proprietatea de unicitate** (orice combinaţie de valori ale atributelor supercheii este unică pentru orice stare a relaţiei), adică:

ti[SK] ≠ tj[SK] dacă i ≠j, unde ti si tj sunt 2 tupuri ale relatiei

Dacă se cunoaşte valoarea supercheii, atunci tuplul poate fi identificat în mod unic

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

CONSTRÂNGERI INTRA-RELATIE: DE TUPLU

DEFINITIE

Cheie candidată (candidate key-CK) este o supercheie ireductibilă:

- **Unicitate**: nu există două tupluri diferite ale relaţiei care să conţină aceeaşi combinaţie de valori ale atributelor cheii CK;
- **Ireductibilitate**: nu există nici o submulţime proprie, nevidă a cheii CK care să aibă proprietatea de unicitate.

Cheie candidată = o supercheie minimală (ireductibilă) si poate fi :
- **simplă** (un singur atribut), sau
- **compusă** (mai multe atribute)

Proprietatea de **unicitate a cheii** (candidate) este o constrângere de integritate a tuplurilor

CONSTRÂNGERI INTRA-RELATIE: DE TUPLU

DEFINITIE

Atunci când există mai multe chei candidate:

- una dintre ele se alege ca şi **cheie primară**,
- celelalte chei candidate se numesc **chei secundare** (sau unice)

Cheie primară (primary key): o cheie candidată cu rol de identificare a tuplurilor

Restricţii pentru chei primare:

- **nici o valoare a atributelor cheii primare nu poate fi modificată prin operaţii de actualizare**
- **nu se admit valori NULL pentru nici unul dintre atributele cheii primare**

Cheie secundară (alternativă, unică) (secondary, alternate, unique key): o cheie candidată care nu a fost desemnată ca şi cheie primară; **Cheile secundare admit valori NULL** pentru unele din atributele lor

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

CONSTRÂNGERI INTRA-RELATIE: DE TUPLU

DEFINITIE

Alegerea **cheii primare** dintre mai multe chei candidate este arbitrară, dar, din motive de eficienta, se alege **cheia cu cel mai mic număr de atribute**

Tipuri de chei primare:

- **cheie primara naturala**: o cheie primară alcatuita din atributele existente ale tipului de entitate (in general, cheile naturale = chei compuse)
- **cheie primară artificială**: un atribut care se adaugă în schema relaţiei special pentru identificarea unică a tuplurilor

CONSTRÂNGERI INTRA-RELATIE: DE TUPLU

DEFINITIE

Definire cheie primară: prin instructiunea **CREATE TABLE** ca o constrângere de tabel sub forma:

```
[CONSTRAINT nume_constr] PRIMARY KEY (lista_atribute)
```

Dacă cheia primară este simplă (un singur atribut), se poate specifica si ca o constrângere de coloana :

Ex.1.:

```
CREATE TABLE SECTII (
IdSectie int PRIMARY KEY NOT NULL,
Nume varchar(50) NOT NULL,
Buget numeric);
```

Definire cheie secundara: prin specificatorul **UNIQUE** în locul specificatorului **PRIMARY KEY**.

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

CONSTRÂNGERI INTRA-RELATIE: DE TUPLU

EXEMPLE

Ex.1.:

```
CREATE TABLE SECTII (
IdSectie int PRIMARY KEY NOT NULL,
Nume varchar(50) NOT NULL,
Buget numeric);
```

Ex.2.:

```
CREATE TABLE ANGAJATI (
IdAngajat int PRIMARY KEY AUTO_INCREMENT,
Nume varchar(20)NOT NULL,
Prenume varchar(20)NOT NULL,
DataNasterii Date,
Adresa varchar(50),
Salariu numeric,
CONSTRAINT UK UNIQUE(Nume,Prenume,DataNasterii,Adresa));
```

→ CHEIE PRIMARA

→ CHEIE SECUNDARA

CONSTRÂNGERI INTRA-RELATIE: DE TUPLU
CHEIE PRIMARA

MySQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

MySQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
);
```

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);
```

MySQL:

```
ALTER TABLE Persons
DROP PRIMARY KEY;
```

SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
DROP CONSTRAINT PK_Person;
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

CONSTRÂNGERI INTRA-RELATIE: DE TUPLU
CHEIE SECUNDARA

MySQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    UNIQUE (ID)
);
```

MySQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT UC_Person UNIQUE (ID,LastName)
);
```

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

```
ALTER TABLE Persons
ADD CONSTRAINT UC_Person UNIQUE (ID,LastName);
```

Cheie primara

1. **Care instructiune SQL creeaza un tabel cu numele "Adrese" cu campurile "ID","Strada", "Numar","Cod_postal", care are ca si cheie primara campul ID:**

```
a)CREATE TABLE (ID int NOT NULL,Strada varchar(40)NULL,Numar
int NULL,Cod_postal int NULL, PRIMARY KEY(ID) );
```

```
b)CREATE TABLE ADRESE (ID int NOT NULL,Strada varchar(40)
NULL,Numar int NULL,Cod_postal int NULL,PRIMARY KEY(ID));
```

```
c)CREATE TABLE ADRESE(ID , Strada, Numar, Cod_postal, PRIMARY
KEY(ID) );
```

```
d)CREATE TABLE ADRESE(ID int NOT NULL, Strada varchar(40) NULL,
Numar int NULL, Cod_postal int NULL);
```

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

CONSTRÂNGERI INTER-RELATII

Chei straine: chei prin care se realizeaza asocierile 1:N între multimile de entităţi (din modelul Entitate-Asociere) în modelul relaţional

EXEMPLU

Ex.: pentru a realiza asocierea 1:N dintre relaţiile `SECTII` si `ANGAJATI`, se adaugă în relaţia `ANGAJATI` atributul `IdSectie`, care reprezintă identificatorul (numărul) secţiei în care lucrează angajatul respectiv:

```
SECTII(IdSectie,Nume, Buget)
ANGAJATI(IdAngajat, Nume, Prenume,DataNasterii, Adresa, Salariu, IdSectie)
```

Diagrama E-A

CONSTRÂNGERI INTER-RELATII

SECTII

IdSectie	Nume	Buget
1	Productie	400000
2	Proiectare	300000
3	Cercetare	200000
4	Documentare	100000

ANGAJATI

IdAngajat	Nume	Prenume	DataNasterii	Adresa	Salariul	IdSectie
1	Ionescu	Ion	1960.01.05	Bucuresti	4000	1
2	Popescu	Petre	1965.02.97	Bucuresti	3200	1
3	Vasilescu	Ana	1961.03.06	Bucuresti	2000	2
4	Ionescu	Ion	1970.03.98	Bucuresti	2000	3

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

CONSTRÂNGERI INTER-RELATII

DEFINITIE

Fie doua relatii R1 si R2, intre care exista o asociere cu raportul 1:N

Cheie straina (foreign key=FK): **o submulţime FK de atribute ale relaţiei R2 care referă relaţia R1 (relatia referita) şi satisface următoarele condiţii:**

- **atributele cheii străine** FK **sunt definite pe domenii compatibile cu cele ale atributelor unei cheii candidate** CK a relaţiei R1
- **valorile atributelor** FK **într-un tuplu din relaţia** R2, **fie sunt identice cu valorile atributelor** CK **a unui tuplu oarecare din starea curentă a relaţiei** R1**, fie sunt NULL**

Definire cheia străină :la crearea tabelului prin constrângere de tabel:

```
[CONSTRAINT nume_constr] FOREIGN KEY (cheie_straina)
      REFERENCES relatia_referita (cheie_candidata)
```

CONSTRÂNGERI INTER-RELATII

- Cheia străină reprezintă o constrângere referenţială intre cele 2 relatii.
- Două domenii sunt compatibile dacă sunt comparabile dpdv semantic (are sens sa fie comparate)
- În limbajul SQL verificarea domeniilor = verificarea tipurilor de date, iar compatibiltatea semantică trebuie să fie asigurată de proiectant

EXEMPLU

Ex.:

```
CREATE TABLE ANGAJATI (
IdAngajat int PRIMARY KEY,
Nume varchar(20) NOT NULL,
Prenume varchar(20) NOT NULL,
IdSecţie int,
CONSTRAINT UK UNIQUE (Nume,Prenume),
CONSTRAINT FK FOREIGN KEY (IdSectie) REFERENCES SECTII(IdSectie));
```

→ CHEIE PRIMARA

→ CHEIE SECUNDARA

→ CHEIE STRAINA

Curs: BAZE DE DATE ÎN INGINERIE ELECTRICĂ

CONSTRÂNGERI INTER-RELATII

MySQL:

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Orders
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Orders
ADD CONSTRAINT FK_PersonOrder
FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

MySQL:

```
ALTER TABLE Orders
DROP FOREIGN KEY FK_PersonOrder;
```