

Baze de date

Universitatea “Transilvania” din Brasov

Lect.dr. Costel Aldea
costel.aldea@gmail.com

Proiectarea logică a bazei de date relaționale

Pas1: Crearea unui model conceptual local, pentru vederile utilizatorilor

- Identificarea tipurilor de entități.
- Identificarea tipurilor de relații.
- Identificarea și atribuirea de attribute la tipurile de entități și tipurile de relații.
- Determinarea domeniilor de definiție a atributelor.
- Determinarea atributelor care compun cheile candidate și primare.
- Specializare/generalizare (pas opțional).
- Desenarea diagramei entity-relationship.
- Verificarea modelului conceptual local cu ajutorul utilizatorului.

Proiectarea logică a bazei de date relaționale

Pas2: Crearea și validarea modelului logic local.

- Proiectarea modelului conceptual local pe un model logic local.
- Crearea relațiilor pentru modelul logic local.
- Validarea modelului, utilizând normalizarea.
- Validarea modelului din nou, utilizând tranzacțiile.
- Desenarea diagramei ER.
- Definirea regulilor de integritate a bazei de date.
- Verificarea modelului logic local cu ajutorul utilizatorului.

Proiectarea logică a bazei de date relaționale

Pas3: Crearea și validarea modelului logic global de date.

- Compunerea medelelor logice locale într-un model logic global.
- Validarea modelului logic global.
- Verificarea posibilității de a completa baza de date în viitor.
- Desenarea diagramei ER finale.
- Verificarea modelului logic global cu ajutorul utilizatorului.

Proiectarea logică a bazei de date relaționale

Pas4: Translatarea modelului logic global în SGBD.

- Proiectarea relațiilor de bază în SGBD.
- Crearea regulilor de integritate în SGBD.

Proiectarea logică a bazei de date relaționale

Pas5: Proiectarea și implementarea reprezentării fizice.

- Analizarea tranzacțiilor.
- Alegerea organizării fișierelor.
- Alegerea indecșilor secundari.
- Introducerea unei redundanțe controlate.
- Estimarea spațiului pe disc.

Proiectarea logică a bazei de date relaționale

Pas6: Proiectarea și implementarea unui mechanism de securitate.

- Crearea view-urilor pentru utilizatori.
- Proiectarea regulilor de acces la baza de date.

Proiectarea logică a bazei de date relaționale

1. Crearea unui model conceptual local, pentru vederile utilizatorilor
2. Crearea și validarea modelului logic local
3. Crearea și validarea modelului logic global de date
4. Translatarea modelului logic global în SGBD
5. Proiectarea și implementarea reprezentării fizice
6. Proiectarea și implementarea unui mecanism de securitate
7. Verificarea sistemului operațional

Limbajele relaționale

- Limbajele bazelor de date sunt împărțite în 2 categorii
 - limbaje de definire a datelor (**DDL**)
 - limbaje de manipulare a datelor (**DML**)
- DDL este utilizat pentru a specifica schema bazei de date
- DML este utilizat pentru citirea și reactualizarea bazei de date
- **DML** asigură un set de procedee ce permit operații de bază pentru manipularea datelor din baza de date:
 - inserarea de date noi
 - modificări de date
 - regăsirea datelor
 - ștergerea de date

Limbajele relaționale

- Limbajele DML pot fi de două tipuri
 - procedurale
 - neprocedurale
- **Limbajele DML procedurale** specifică modul cum trebuie să fie obținut rezultatul unei instrucțiuni DML
- **Limbajele DML neprocedurale** descriu numai ce rezultat trebuie obținut
- La baza limbajelor relaționale stă algebra relațională și calculul relațional
- Algebra relațională și calculul relațional constituie fundament pentru DML

Algebra relațională

- **Algebra relațională** este un limbaj teoretic, cu operații care acționează asupra uneia sau mai multor relații, pentru a defini o altă relație, fără modificarea celor inițiale
- Atât operanzii cât și rezultatele sunt relații, așa că, ieșirea unei operații poate deveni intrare pentru o alta
 - Aceasta permite imbricarea expresiilor, la fel ca la operațiile matematice.
- Algebra relațională este un limbaj de tip câte-o-relație-o-dată, în care toate tuplurile sunt manipulate într-o singură instrucțiune, fără ciclare
- interogările sunt expresii compuse din operații care au ca operanzi relații și rezultatul este o relație
- Operațiile algebrei relationale: operații pe mulțimi și operații speciale, la care se adaugă operația de redenumire (*rename*) a atributelor (*E.Codd*)

Operațiile algebrei relationale

- Operațiile relationale pe mulțimi acționează asupra relațiilor văzute ca mulțimi (de tupluri), fără a lua în considerație compoziția fiecărui tuplu; acestea sunt:
 - Reuniunea
 - Intersecția
 - Diferența
 - Produsul cartezian
- Operațiile relaționale speciale iau în considerație compoziția tuplurilor, formate din valori ale atributelor relațiilor; acestea sunt:
 - Restricția
 - Proiecția
 - Joncțiunea
 - Diviziunea
- Proprietatea de închidere: operanzii (unul sau doi operanzi) sunt relații, rezultatul este o relație; această proprietate permite operații imbricate: proiecția unei joncțiuni etc.

Limbajele relaționale - *Algebra relațională*

- Cele 5 operații fundamentale din algebra relațională sunt
 - selecția
 - proiecția
 - produsul cartezian
 - reuniunea
 - diferența

Limbajele relaționale - *Algebra relațională*

- Algebra relațională definește, suplimentar, operațiile de
 - *uniune*
 - *intersecție*
 - *împărțire*
- Acestea pot fi exprimate prin intermediul celor 5 operații fundamentale
- Operațiile de selecție și proiecție sunt **unare** pentru că operează asupra unei singure relații
- Celelalte acționează asupra unor perechi de relații și se numesc operații **binare**

Algebra relațională - Selecția (restricția)

$$\sigma_{\text{predicat}}(R)$$

- Operația de selecție acționează asupra unei singure relații R și definește o relație care conține numai acele tupluri ale lui R care satisfac condiția specificată (predicatul)

- Exemplu

$$\sigma_{\text{Salariul} > 1000}(\text{Angajati})$$

Algebra relațională - Proiecția

$$\Pi_{col1, \dots, coln}(R)$$

- Operația de proiecție acționează asupra unei singure relații R și definește o relație care conține un subset vertical al lui R , extrăgând valorile atributelor specificate și eliminând dublurile

- Exemplu

$$\Pi_{NrMarca, Nume, Salariul}(Angajati)$$

Algebra relațională - Produsul cartezian

$$R \times S$$

- Produsul cartezian definește o relație care reprezintă o concatenare a fiecărui tuplu din relația R cu fiecare tuplu din relația S
- Produsul cartesian (Cartesian-Product) a două relații $r(R)$ și $s(S)$:
$$q = r \times s = \{ tp \mid t \in r \text{ and } p \in s \}, Q = R \cup S$$
- Se presupune ca multimile R și S sunt disjuncte, adică $R \cap S = \emptyset$
- Dacă attributele din R și S nu sunt disjuncte, atunci (unele):
 - se pot redenumi (RENAME nume_atribut AS noul_nume_atribut) sau
 - se pot califica cu numele relației careia îi aparțin (folosind operatorul “punct”)

Produsul cartezian

- Tuplurile relației rezultat se obțin prin concatenarea valorilor atributelor fiecărui tuplu din prima relație cu valorile atributelor tuturor tuplurilor din a doua relație
- Relația rezultată are numărul de tupluri (cardinalitatea) egal cu produsul numărului de tupluri ale relațiilor operand

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

r x s

Algebra relațională - Reuniunea

$$R \cup S$$

- Reuniunea a două relații cu i , respectiv j tupluri, reprezintă o relație obținută prin concatenarea celor două și având maxim $i+j$ tupluri, tuplurile duble fiind eliminate. R și S trebuie să fie compatibile la reuniune

Reuniunea (union)

- *Reuniunea(union) a două relații compatibile $r(R)$ și $s(S)$:*

$$q = r \cup s = \{ t \mid t \in r \text{ or } t \in s \}$$

- Pentru ca r și s să fie compatibile, trebuie ca:
 - r și s să aibă același grad (același număr de atribute)
 - Atributele corespondente (în ordine pozițională) să fie compatibile
- Tuplurile care aparțin ambelor relații se introduc în relația rezultat o singură dată (nu se duplică)
- Relația rezultat are un număr de tupluri (cardinalitatea) mai mic sau egal cu suma numerelor de tupluri ale celor doi operanzi

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
α	2
β	1
β	3

$r \cup s$

Algebra relațională - Diferența

$R - S$

- Diferența definește o relație ce constă din tuplurile care sunt în R și nu sunt în S . R și S trebuie să fie compatibile la diferență
- Diferența (set-difference) a două relații compatibile $r(R)$ și $s(S)$:

$$q = r - s = \{ t \mid t \in r \text{ and } t \notin s \}$$

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	1
β	1

$r - s$

Intersectia

Intersectia (set-intersection) a două relații compatibile $r(R)$ și $s(S)$:

$$q = r \cap s = \{ t \mid t \in r \text{ and } t \in s \}$$

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

A	B
α	2

$r \cap s$

Limbajul SQL

- ❑ Limbajul IBM Sequel dezvoltat ca parte a proiectului System R la IBM San Jose Research Laboratory (1970)
- ❑ Redenumit Structured Query Language (SQL)
- ❑ Standarde SQL -ANSI si ISO:

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

- ❑ Fiecare SGBDR implementează un dialect al limbajului SQL, ceea ce micșorează gradul de portabilitate a aplicațiilor
- ❑ În diferitele implementări ale limbajului SQL pot să lipsească unele comenzi prevăzute în standard, dar pot exista extensii specifice

Caracteristicile ale limbajului SQL

- ❑ Limbajul SQL folosește reprezentarea prin tabele a relațiilor, reprezentare care este mai simplă și mai intuitivă (folosește termenii tabel, linie, coloană)
- ❑ Limbajul SQL cuprinde:
 - 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 tranzacțiilor, controlul securității, protecția datelor etc.
- ❑ Limbajul SQL este un limbaj neprocedural:
 - o instrucțiune SQL specifică ce informații trebuie să fie setate sau obținute, nu modul (procedura) în care se operează
 - limbajul SQL nu conține instrucțiuni de control al fluxului execuției (instrucțiuni ca for, while, if, etc)
- ❑ Pentru aplicațiile de baze de date, s-au dezvoltat extensii procedurale ale limbajului SQL, biblioteci și interfețe de programare care integrează instrucțiunile SQL

Structura lexicala a limbajului SQL

- O instrucțiune SQL (*statement*) este o secvență de elemente - de regula terminată cu semnul punct și virgulă (;)
- Fiecare instrucțiune SQL conține o comandă SQL (*command*), care specifică ce acțiune se efectuează, urmată de alte elemente, care specifică operații, clauze, parametri etc.
 - Exemplu: **SELECT * FROM ANGAJATI;**
- Elementele (*tokens*) instrucțiunilor SQL
 - cuvinte cheie (*key words*): **CREATE, INSERT, SELECT, WHERE, FROM** etc.
 - identificatori (*identifiers*):
 - simpli - numai caractere alfa-numerice și underscore (_): ANGAJATI, Nume, Prenume etc.
 - delimitați (*quoted*) - pot conține orice caracter, folosește ghilimele : 'Nume', 'Prenume', etc.

Structura lexicala a limbajului SQL

- constante (literali): 1000, 100.5, 'Ionescu', NULL
- caractere speciale: *, .. ;
- Spațiile albe (*whitespaces*) separa elementele: spațiu, linie nouă, tab
- O instrucțiune se poate scrie pe una sau mai multe linii, iar într-o linie se pot introduce una sau mai multe instrucțiuni
- Limbajul SQL este *case-insensitive* (nu deosebește literele mici de cele mari) cu excepția identificatorilor delimitați (*quoted*) care sunt *case-sensitive*

Expresii si operatori in SQL

- O expresie SQL constă dintr-unul sau mai mulți operanzi, operatori și paranteze.
 - Parantezele se pot folosi pentru a preciza o anumită ordine a operațiilor, dacă aceasta este diferită de ordinea implicită data de precedenta operatorilor.
- Un operand poate fi:
 - numele unei coloane - in acest caz se foloseste valoarea memorata in acea colona intr-una sau mai multe linii ale tabelului
 - o constantă (literal)
 - valoarea returnată de o functie
- Un operator SQL este exprimat prin:
 - unul sau mai mai multe caractere speciale; exemple: +, -, *, /, %, <<, >> etc.

Expresii si operatori in SQL

- un cuvânt cheie; exemple: AND, OR, NOT, LIKE etc.
- Operatori SQL: binari sau unari (dupa numarul de operanzi)
- Operatori SQL: aritmetici, logici, de comparație SQL, relationali
 - Operatori aritmetici de operatii cu numere intregi sau reale: +, -, *, /, %, ^
 - Operatori aritmetici orientati pe biti: ~, &, |, #
 - Operatori aritmetici de comparatie: <, >, =, <> (sau !=), <=, >=
 - Operatori de comparatie SQL: IS NULL, IS NOT NULL, BETWEEN, IN, LIKE
 - Operatori relationali: UNION, INTERSECT, MINUS

Operatori SQL

- ❑ Toti operatorii de comparație returneaza valori logice:
 - true (1), dacă condiția este îndeplinită
 - false (0) dacă condiția nu este îndeplinită
 - null dacă ambii operanzi au valoarea null
- ❑ Operatorii logici (NOT, AND, OR):
 - se aplică unor valori logice trivalente (cu 3 valori: true (1), false (0) și null - lipsa de informatie)
 - returnează o valoare logică trivalentă

Funcții SQL

- ❑ Funcții SQL: funcții de agregare și funcții scalare.
- ❑ Funcțiile de agregare calculează un rezultat din mai multe linii ale unui tabel
 - Aceste funcții vor fi detaliate ulterior, la descrierea instrucțiunii SELECT.
- ❑ Funcțiile scalare:
 - Primesc unul sau mai multe argumente și returnează valoarea calculată sau NULL în caz de eroare
 - Argumentele funcțiilor pot fi constante (literale) sau valori ale atributelor specificate prin numele coloanelor corespunzătoare

Funcții SQL

- Tipuri de funcții scalare SQL:
 - Funcții de calcul trigonometric (**sin, cos, tan**etc.), **funcții de calcul al logaritmului (ln, log), al puterii (power), funcții de rotunjire (floor, ceil), etc.**
 - Funcții pentru manipularea șirurilor de caractere: **concat, replace, upper**etc.
 - Funcții pentru data calendaristică și timp: **add_months, next_day, last_day** etc.
 - Funcții de conversie: **to_number, to_cha**etc.
- Funcțiile scalare se folosesc în expresii, care pot să apară în diferite clauze ale instrucțiunilor SQL

Tipuri de date SQL

- ❑ Tipuri de date SQL2: numeric, șiruri de caractere, șiruri de biți, data (calendaristică), timp
- ❑ Tipul numeric:
 - numere întregi: **integer** sau **int(4 octeți)**, **smallint(2 octeți)**
 - numere reale reprezentate în virgulă flotanta: **float (4 octeți)**, **real** și **double [precision](8 octeți)**
 - numere zecimale reprezentate cu precizia dorită (tipul numeric sau decimal, memorate ca șir de caractere): **numeric[(p,s)]** (sau **decimal [(p,s)]**), unde p (precizia) este numărul total de cifre, iar s (scara) este numărul de cifre după punctul zecimal

Tipuri de date SQL

- Siruri de caractere:
 - **character(n)**, prescurtat, **char(n)**-șir de caractere de lungime fixă (n)
 - **character varying(n)**, prescurtat **varchar(n)**-șir de caractere de lungime variabilă, maximum n
- Siruri de biți -secvențe de cifre binare (care pot lua valoarea 0 sau 1):
- **bit(n)** - sir de biti de lungime fixă (n)
- **bit varying(n)** sir de biti lungime variabilă, maxim n

Tipuri de date SQL

- Tipurile SQL pentru data calendaristică și timp sunt: date, time, timestamp, interval:
 - Tipul **date**: memorarea datelor calendaristice prin utilizarea a trei câmpuri (year, month, day), în formatul yyyy-mm-dd; se admit numai date valide
 - Tipul **time**: memorarea timpului, folosind trei câmpuri (hour, minute, second) în formatul HH:MM:SS; se admit numai valori valide
 - Tipul **timestamp(p)**: memorarea combinată a datei calendaristice și a timpului, cu precizia p pentru câmpul second. Valoarea implicită a lui p este 6
 - Tipul **interval** este utilizat pentru memorarea intervalelor de timp

Tipuri de date SQL

- Variante de tipuri de date SQL specifice în diferite sisteme SGBD; Exemple:
 - SGBD Microsoft SQL Server: **tinyint** -număr întreg pe 1 octet
 - SGBD Oracle: **varchar2** -șir de caractere de lungime variabilă
- Standardul SQL2 nu suportă tipuri de date și operații definite de utilizator
- Standardul SQL3 suportă tipuri de date și operații definite de utilizator, care sunt caracteristice ale modelului de date obiect-relațional
- Actualmente, producătorii de sisteme de baze de date relaționale introduc treptat diferite caracteristici ale modelului obiect-relațional cuprinse în SQL3

Domenii SQL

- ❑ In SQL2 domeniile atributelor se specifică pe baza tipurilor de date predefinite ale limbajului SQL, deci nu corespund intru totul cunoștința de domeniu relațional
- ❑ Standardul SQL2 prevede comanda CREATE DOMAIN, care definește un domeniu pe baza unui tip predefinit SQL2 și cu unele constrângeri
- ❑ Standardul SQL3 prevede comanda CREATE TYPE care creează tipuri definite de utilizator (user-defined types),

Domenii SQL

- In SGBD-urile actuale sunt implementate diferite versiuni din standarde:
 - In SQL Server se pot crea domenii ale atributelor cu comanda SQLCREATE DOMAIN
 - În Oracle (8i, 9i, 10g, 11g) se pot crea tipuri de date noi, folosind comanda CREATE TYPE, care permite gruparea sub un anumit nume a mai multor attribute si operatii
 - In PostgreSQL de asemenea se pot crea tipuri de date noi, folosind comanda CREATE TYPE

Conventii de notatie

- Pentru prezentarea limbajului SQL si a altor limbaje, bibliotecisi interfete
- Caracterele folosite pentru a specifica o anumită convenție sintactică (paranteze, bara verticală, virgula, etc.) nu apar în instrucțiunile propriu-zise
- Listele de elemente (compuse din mai multe elemente separate prin virgulă) vor fi exprimate fie folosind una cele trei din construcțiile de mai sus, care se potrivește cel mai bine instrucțiunii respective

[] (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 același tip; elementele repetate sunt separate prin virgulă
lista_elemente	Listă de elemente de același tip separate prin virgulă

Instrucțiuni SQL

- Componenta de definire a datelor din SQL (LDD -Limbajul de Definiere a Datelor):
 - Crearea (**CREATE**), modificarea (**ALTER**) și distrugerea (**DROP**) obiectelor bazei de date
 - Obiectele bazei de date sunt: tabele de bază (**TABLE**), tabele vedere (**VIEW**), indexuri (**INDEX**), proceduri (**PROCEDURE**), trigere (**TRIGGER**), utilizatori (**USER**)
- Exemple de 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**

Instrucțiuni SQL

- **DROP TABLE, DROP VIEW, DROP INDEX, DROP USER**
- **DROP FUNCTION, DROP PROCEDURE, DROP TRIGGER**
- Componenta de manipulare a datelor din limbajul SQL (Limbajul de Manipulare a Datelor -LMD) conține comenzile: SELECT, INSERT, UPDATE și DELETE
- Instrucțiunile SQL se transmit SGBD-ului:
 - de către diferite programe client (client grafic, linie de comandă, program executabil)
 - SGBD-ul execută instrucțiunea SQL
 - și returnează un răspuns (rezultatul operației sau un cod de eroare)

Crearea tabelelor

- Instrucțiunea CREATE TABLE are următoarea sintaxă:

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

- Constrângerile impuse fiecărei coloane (atribut), ca și constrângerile de tabel, sunt opționale și vor fi discutate în secțiunea următoare.

Crearea tabelelor

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

- ❑ Instrucțiunea **CREATE TABLE** definește atât tipul relației cât și o variabilă relație care inițial este vidă (nu conține nici un tuplu)

Crearea vederilor

- Tabelele create cu instrucțiunea CREATE TABLE:
 - se numesc și *tabele de bază(base tables)*
 - ele sunt memorate în fișierele bazei de date și pot fi accesate pentru introducerea, modificarea și regăsirea (interogarea) datelor
- *Un tabel vedere(view) este un tabel virtual care:*
 - nu este memorat fizic în fișiere
 - reprezintă o selecție (după un anumit criteriu) a datelor memorate în unul sau mai multe tabele de bază

Crearea vederilor

- Un tabel vedere se creeaza cu instrucțiunea SQL:
CREATE VIEW nume_vedere AS (SELECT...);
- Formatul comenzii SELECT va fi descris în capitolul următor
- Datele (valorile atributelor) sunt memorate o singură dată, în tabelele de bază, dar pot fi accesate atât prin tabelele de bază cât și prin tabelele vederi
- Un tabel vedere este întotdeauna actualizat ("la zi"), adică orice modificare efectuată în tabelele de bază se regăsește imediat în orice tabel vedere creat pe baza acestora

Modificarea si stergerea tabelelor si a vederilor

- ❑ Comanda de modificare a tabelelor (ALTER TABLE) permite:
 - adăugarea sau ștergerea unor attribute
 - modificarea domeniilor unor attribute
 - adăugarea, modificarea sau ștergerea unor constrângeri ale tabelului
- ❑ Pentru adaugare unei coloane intr-un tabel se foloseste clauza ADD, urmata de numele coloanei si numele domeniului (tipul SQL) atributului corespunzator.

Modificarea si stergerea tabelelor si a vederilor

**ALTER TABLE ANGAJATI ADD DataAngajarii
date;**

- Pentru ștergerea unei coloane dintr-un tabel se folosește clauza DROP, urmata de numele coloanei care se va șterge.

**ALTER TABLE ANGAJATI DROP
DataAngajarii;**

- Instrucțiunile de ștergere a tabelelor de bază și a vederilor sunt:

DROP TABLE nume_tabel;

DROP VIEW nume_vedere;

Instrucțiunea SELECT

- SELECT -instrucțiune de interogare, prin care se regăsesc informațiile din unul sau mai multe tabele ale bazei de date după un criteriu (condiție) dat

- Sintaxa generală:

SELECT [DISTINCT] lista_coloane
[FROM lista_tabele]
[WHERE condiție]
[clauze_secundare];

- SELECT returnează un tabel cu coloanele din “lista_coloane”

Instrucțiunea SELECT

- ale acelor linii (tupluri) ale produsului cartezian al tabelelor din “lista_tabele” pentru care expresia logică “conditie” este adevărată (are valoarea TRUE).
- Instrucțiunea SELECT are următoarele secțiuni (clauze):
 - Clauza **SELECT** definește lista de coloane a tabelului rezultat
 - Clauza **FROM** indică lista de tabele din care se selectează rezultatul
 - Clauza **WHERE** definește condiția pe care trebuie să o îndeplinească fiecare linie a tabelului rezultat
 - Clauze secundare (**ORDER BY, GROUP BY, HAVING**): permit ordonări sau grupări ale tuplurilor (liniilor) rezultate

Clauza SELECT

- Clauza SELECT specifica:
 - lista coloanelor unor tabele (date in “lista_tabele”)
 - expresii care vor fi calculate și afișate
- Exemple:

**SELECT ID, Name, CountryCode,
District from city;**

SELECT 3*4, cos(45), floor(12.45);

Clauza SELECT

- ❑ Eliminarea liniilor duplicat cu parametrul DISTINCT:
SELECT [DISTINCT] CountryCode FROM city;
- ❑ Selectarea tuturor coloanelor produsului cartezian al tabelelor date -cu caracterul * ca si “lista_coloane” :
SELECT * FROM city;
- ❑ În clauza SELECT se pot redenumi coloanele tabelelor sau se pot specifica nume pentru expresii, folosind următoarea sintaxă:
SELECT nume1 [AS] noul_nume1 [...n] FROM lista_tabele [alte_clauze];
SELECT ID, Name Oras, CountryCode ‘Cod Tara’FROM city;

Funcții de agregare

- În clauza SELECT se pot introduce și funcții agregat (totalizatoare)

- Exemple:

SELECT COUNT(*) FROM city;

SELECT MAX(Population) FROM city;

SELECT MIN(Population) FROM city;

SELECT AVG(Population) FROM city;

Funcția	Valoarea returnată
COUNT	Numarul de linii al tabelului rezultat
SUM	Suma valorilor din coloana dată ca argument
MAX	Valoarea maximă din coloana dată ca argument
MIN	Valoarea minimă din coloana dată ca argument
AVG	Valoarea medie din coloana dată ca argument

Clauzele FROM si WHERE

- ❑ *Clauza FROM specifica “lista_tabele” din care se selecteaza rezultatul*
- ❑ Numele coloanelor din “lista_coloane” (clauza SELECT) trebuie să fie distincte
- ❑ Dacă nu sunt distincte, se califică unele coloane cu numele tabelului caruia îi aparțin -folosind operatorul “punct”(.). De exemplu:

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

Clauzele FROM si WHERE

- *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) și paranteze
 - o valoare logică se obtine ca rezultat al comparației între doi operanzi folosind un operator de comparație
 - un operand poate fi un atribut (nume de coloană), o constantă, valoarea unei expresii aritmetice sau o valoare returnată de o funcție
 - operatorii de comparație pot fi operatori aritmetici sau operatori SQL de comparație

SELECT * FROM city WHERE Population > 1000;

SELECT * FROM city

**WHERE Population BETWEEN 1000 AND 100000 AND
CountryCode='NLD';**

Alte clauze

- *Clauza **ORDER BY** specifica numele atributului după care se face ordonarea liniilor tabelului rezultat*
SELECT * FROM city order by CountryCode;
- Ordonarea în ordine crescătoare: parametrul ASC (implicit); în ordine descrescătoare: DESC.
SELECT * FROM city order by CountryCode DESC;
- *Clauza **GROUP BY** se folosește pentru gruparea rezultatelor funcțiilor agregat după valoarea uneia sau mai multor coloane.*
SELECT CountryCode, AVG(Population) FROM city GROUP BY CountryCode;
- *Clauza **HAVING** înlocuiește clauza **WHERE** atunci când în condiția care trebuie să fie îndeplinită se folosesc funcții agregat.*
SELECT CountryCode, AVG(Population) FROM city GROUP BY CountryCode HAVING AVG(Population) >800000;

Instrucțiunea INSERT

- Instrucțiunea INSERT se folosește pentru introducerea datelor în tabele și are următoarea sintaxă:

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

- Între valori și numele de coloane trebuie să existe o corespondență pozițională. De exemplu:

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

Instructiunea INSERT

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

- ordinea valorilor introduse trebuie să respecte ordinea coloanelor tabelului
 - ordinea coloanelor provine din ordinea de definire a atributelor prin instrucțiunea CREATE TABLE, precum și din operațiile ulterioare de alterare a tabelului
 - ordinea coloanelor se poate afla prin instrucțiunea DESCRIBE nume_tabel.
- De exemplu, introducerea unei linii în tabelul ANGAJATI(IdAngajat, Nume, Prenume, DataNasterii, Adresa, Functia, Salariu) se poate face cu instrucțiunea:
- INSERT INTO ANGAJATI VALUES(100,'Mihailescu',
'Mihai','1950-04-05','Craiova','Inginer', 3000);**

Instrucțiunile UPDATE și DELETE

- *Instrucțiunea UPDATE permite actualizarea valorilor coloanelor (atributelor) din una sau mai multe linii ale unui tabel și are sintaxa:*

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

- Clauza WHERE: actualizarea valorilor se efectuează numai asupra acelor linii care îndeplinesc condiția dată. Exemplu:

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

Instrucțiunile UPDATE și DELETE

- *Instrucțiunea DELETE permite ștergerea uneia sau mai multor linii dintr-un tabel și are sintaxa:*

DELETE FROM nume_tabel [WHERE conditie];

- Din tabel se șterg acele linii care îndeplinesc condiția dată în clauza WHERE. Dacă este omisă clauza WHERE, vor fi sterse toate liniile din tabel.

DELETE FROM ANGAJATI WHERE Nume = 'Popescu';

- Dacă este omisă clauza WHERE, vor fi modificate valorile coloanelor din toate liniile tabelului.