# *Curs 11 : Limbaje pentru prelucrarea datelor relationale*

1. Generalitati
2. Limbaje algebrice
3. Limbaje predicative
4. Utilizarea limbajelor de prelucrare a datelor relationale in contextul limbajelor de programare
5. *SQL*
6. Limbajul *MDX*

- Modelul relaţional incorporeaza mai multe limbaje de interogare (neprocedurale)

⇒ orice utilizator poate descri informatia cautata, fără a preciza modul în care este obţinut acest rezultat

- O relaţie poate fi definită:
  - ca o mulţime
  - ca un predicat

⇒ clasificarea limbajelor de prelucrare a datelor relaţionale.

## Clasificarea limbajelor de prelucrare a datelor relaţionale:

- ✓ limbaje algebrice – bazate pe teoria mulţimilor (*SEQUEL*, *SQL*);
- ✓ limbaje predicative – bazate pe calculul predicatelor:
  - orientate pe tupluri (*QUEL*, *ALPHA*);
  - orientate pe domenii (variabilele iau valori în domeniile relaţiilor, şi nu în tuplurile acestora)
    - ➢ non-grafice
    - ➢ grafice
      - ❑ cu variabile domeniu explicite (*QBE*);
      - ❑ fără variabile domeniu explicite (*LAGRIF*, *CUPID*, *VGQF*).

Limbajele grafice oferă utilizatorului o imagine sau o ilustrare a structurii relaţiei

→ utilizatorul completează un exemplu cu ceea ce doreşte

→ sistemul returnează datele cerute în acest format.

## 🕮 *SQL (Structured Query Language)*

= **limbajul standard** de descriere a datelor şi accesare a informaţiilor din BD

- creat şi dezvoltat iniţial de către compania *IBM Research*,
- implementat în cadrul prototipului *System R*;
- exista peste o sută de dialecte

⇒ dificultati si costuri de intretinere si portabilitate

⇒ s-a constituit un grup de specialisti:*SAG* (*SQL Access Group*) cu scopul de:

- ✓ a construi un limbaj *SQL* comun
- ✓ a realiza pentru fiecare dialect un program de conversie din respectivul dialect în *SQL*, şi invers,

⇒ au fost create două tehnologii puternice:

A. API

B. ODBC.

**(A.)** Tehnologia ***API*** (*Application Programming Interface*) =

= o bibliotecă de funcţii *SQL* ce se pot integra într-un program gazdă

- avantaj: este o tehnica de lucru familiara programatorilor
- dezavantaj: lipsa de interoperabilitate:

  programele trebuie preprocesate, utilizând precompilatorul asigurat de către producătorul sistemului, şi legate la biblioteca *API*, furnizată tot de către acesta =>
  - costuri mari
  - munca suplimentara de programare.

## (B.) Tehnologia ***ODBC*** (*Open Database Connectivity*)

= o mulţime de primitive ce pune la dispoziţia utilizatorilor o interfaţă comună pentru accesarea bazelor de date *SQL* eterogene

- creata in 1992, de *Microsoft* , pe baza limbajului *C*
- a devenit standardul industrial *de facto*
- avantaje: (1) grad înalt de interoperabilitate:
  - interfaţa permite programatorului să construiască şi să distribuie o aplicaţie *client-server* fără a avea în vedere un anumit SGBD
  - driverele bazei de date sunt adăugate apoi, pentru a lega aplicaţia de sistemul ales de utilizator.

    (2) grad inalt de flexibilitate
  - instrucţiunile *sql* pot fi incluse explicit în codul sursă sau construite dinamic în timpul execuţiei.
  - datele pot fi expediate şi recepţionate în formatul convenabil pentru aplicaţie.

Arhitectura interfetei ***ODBC*** include patru componente:

1. aplicaţia:
   - efectuează prelucrarea şi apelurile la funcţiile *ODBC* pentru a transmite comenzile *SQL* către SGBD şi a regăsi rezultatele furnizate de către acesta
2. administratorul de drivere:
   - încarcă driverele în contul unei aplicaţii
3. agentul pentru drivere şi baza de date:
   - procesează apelurile de funcţii ODBC,
   - transmite cererile SQL către o anumită sursă de date
   - returnează rezultatele spre aplicaţie
   - dacă este necesar: modifică cererea unei aplicaţii, astfel încât aceasta să se conformeze sintaxei acceptate de către SGBD-ul asociat
4. sursa de date:
   - este formată din datele pe care doresc să le acceseze:
     - ✓ utilizatorul şi SGBD-ul asociat
     - ✓ SO gazdă şi platforma de reţea (dacă există).

Interfata ***ODBC*** defineşte următoarele elemente:

1. o bibliotecă de de funcţii care permit:
   - conectarea unei aplicaţii la un SGBD,
   - executarea instrucţiunilor *SQL*
   - regăsirea rezultatelor;
2. o modalitate standard de conectare şi începere a unei sesiuni de lucru într-un SGBD;
3. o reprezentare standard a tipurilor de date;
4. o mulţime standard de coduri de erori;
5. o sintaxă a limbajului *SQL*, bazată pe specificaţiile sistemului *X/Open*
6. o interfaţă la nivel de apelare *CLI* (*Call Level Interface*).

# *Curs 11 : Limbaje pentru prelucrarea datelor relationale*

1. Generalitati
2. Limbaje algebrice
3. Limbaje predicative
4. Utilizarea limbajelor de prelucrare a datelor relationale in contextul limbajelor de programare
5. *SQL*
6. Limbajul *MDX*

## Limbajele algebrice

= limbaje neprocedurale care utilizează relaţii pentru a transforma datele de intrare în datele de ieşiri solicitate

(i.e.: limbaje orientate spre transformări: se exprimă ceea ce se doreşte în funcţie de ceea ce se cunoaşte)

**Exemple**:

- *SQUARE*,
- *SEQUEL*
- *SQL*;

***SEQUEL*** (*Structured English as a Query Language*) =

= limbaj algebric definit în 1974 de D. D. Chamberlin şi R.F. Boyce, tot de la Laboratorul de cercetări *IBM*, pentru prototipul relaţional *System R*.

- este singurul limbaj relaţional care are integrată închiderea tranzitivă
- operaţia fundamentală a limbajului: *SELECT*
- o extensie comercială a acestui limbaj: *SQL (SEQUEL* = parintele *SQL)*.

# *Curs 11 : Limbaje pentru prelucrarea datelor relationale*

1. Generalitati
2. Limbaje algebrice
3. Limbaje predicative
4. Utilizarea limbajelor de prelucrare a datelor relationale in contextul limbajelor de programare
5. *SQL*
6. Limbajul *MDX*

## 🕮 Limbajele predicative

= limbaje bazate pe calculul cu predicate

- cererile: mulţimi de tupluri sau valori pentru care se specifică, sub forma unor predicate, proprietăţile pe care trebuie să le îndeplinească

⇒ calculul se bazează pe utilizarea:

- ✓ variabilelor de tip tuplu, adică variabile ale căror singure valori permise sunt tupluri ale relaţiei
- ✓ cuantificatorilor:
  - ∃ este utilizat în formulele care trebuie să fie adevărate pentru cel puţin o instanţă
  - ∀ este utilizat formulele care trebuie să fie adevărate pentru toate instanţele

- variabilele de tip tuplu se numesc:
  - ✓ variabile libere dacă nu sunt calificate de către un cuantificator
  - ✓ variabile legate, altfel.

## 🕮 *QUEL* =

**=** limbaj predicativ orientat pe tupluri

- creat la Universitatea Berkeley California în 1974 pentru sistemul *INGRES*
- comercializat pentru *PDP.11* şi *VAX*
- poate fi utilizat independent sau inclus în limbajul *C*.

Limbajul este caracterizat de:

- declararea unei variabile tuplu pentru fiecare relaţie (prin *RANGE*)
- absenţa cuantificatorilor în expresii;
  - ∃ : este reprezentatimplicit prin declaraţia *RANGE*, iar cuantificatorul universal este simulat cu ajutorul.
  - ∀ : este simulat cu ajutorul funcţiei *COUNT*
- utilizarea unor operatori speciali pe mulţimi.

## *QUEL* (cont.)

- **Exemplu.** Să se listeze numele studentilor care urmeaza cursuri de tip facultativ

```
RANGE OF c IS curs
RANGE OF s IS student
RANGE OF u IS urmeaza
RETRIEVE s.nume
WHERE (s.cod_student = u.cod_student)
        AND (u.cod_curs = c.cod_curs)   AND (c.tip = facultativ)
```

- **Exemplu.** Să se listeze numele studentilor care urmeaza **toate** cursurile de tip facultativ

```
RANGE OF c IS curs
RANGE OF s IS student
RANGE OF u IS urmeaza
RETRIEVE s.nume
WHERE COUNT(f.cod_curs)
     WHERE (s.cod_student = u.cod_student)
         AND (u.cod_curs = c.cod_curs)  AND (c.tip = 'facultativ')
             = COUNT(c.cod_curs WHERE (c.tip = facultativ'))
```

## *QUEL* (cont.)

- **Exemplu.** Să se listeze numele studentilor care urmeaza cursuri de tip facultativ

```
RANGE OF c IS curs
RANGE OF s IS student
RANGE OF u IS urmeaza
RETRIEVE s.nume
WHERE (s.cod_student = u.cod_student)
        AND (u.cod_curs = c.cod_curs)   AND (c.tip = facultativ)
```

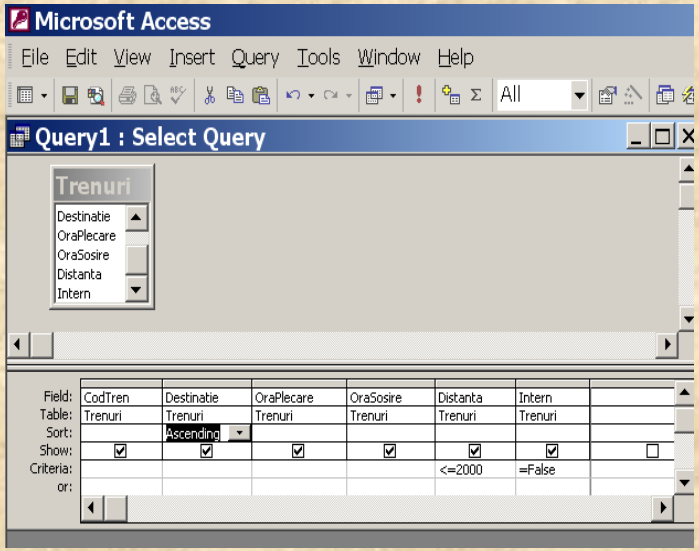
- **Exemplu.** Să se listeze numele studentilor care urmeaza **toate** cursurile de tip facultativ

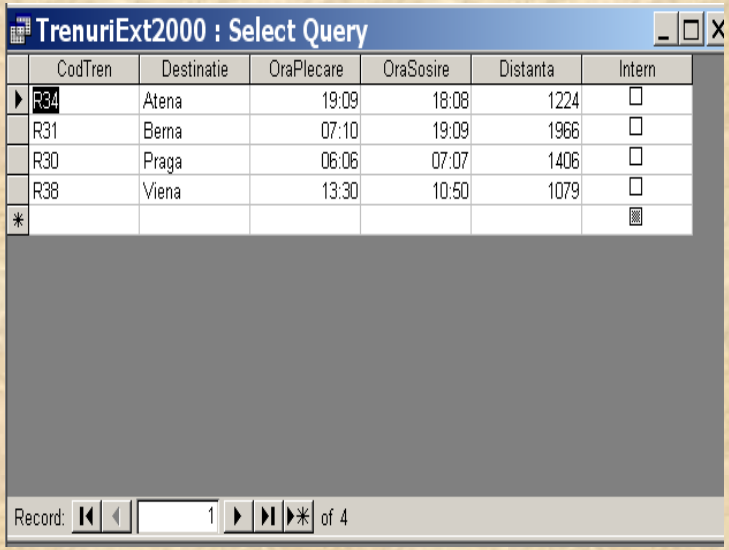
```
RANGE OF c IS curs
RANGE OF s IS student
RANGE OF u IS urmeaza
RETRIEVE s.nume
WHERE COUNT(f.cod_curs)
     WHERE (s.cod_student = u.cod_student)
         AND (u.cod_curs = c.cod_curs)  AND (c.tip = 'facultativ')
             = COUNT(c.cod_curs WHERE (c.tip = facultativ'))
```

## 🕮 *QBE* (*Query By Example*)

= un limbaj predicativ orientat spre domenii

- prezentat de M.M. Zloof în 1977 şi comercializat de *IBM* după 1980, pentru a-i ajuta utilizatorii neiniţiaţi
- extrem de accesibil => limbaj furnizat (în diferite forme) de majoritatea SGBD-urilor, inclusiv *Microsoft Access*.
- reprezintă un mod de tratare virtual pentru accesarea informaţiilor dintr-o BD, prin utilizarea şabloanelor de interogare
- limbajul dispune de primitive de programare grafică a cererilor de date:
  - ✓ utilizatorii completează o mulţime de câmpuri predefinite intr-un ecran special.
  - ✓ SGBD-ul construieşte în fundal instrucţiunea *SQL*.
- nu există un standard oficial pentru această clasă de limbaje
- funcţionalitatea este, în general, foarte asemănătoare
- limbajele sunt mai intuitive decât *SQL*.
- există posibilitatea de a crea o nouă relaţie care poate fi:
  - ✓ o schemă virtuală ce reflectă modificările din relaţiile de bază sau
  - ✓ o imagine în care modificările nu sunt stocate.



# *Curs 11 : Limbaje pentru prelucrarea datelor relationale*

1. Generalitati
2. Limbaje algebrice
3. Limbaje predicative
4. Utilizarea limbajelor de prelucrare a datelor relationale in contextul limbajelor de programare
5. *SQL*
6. Limbajul *MDX*

Două abordări posibile:
- integrarea;
- extensia.

**Integrarea LMD într-un limbaj de programare**
- se consideră limbajul de prelucrare independent de limbajul de programare
- se presupune că există construcţii speciale care permit utilizarea limbajului de prelucrare în limbajul gazdă:
  - Exemplu
    comenzile *SQL*, integrate în limbajul *PL/1* sunt prefixate de caracterul $ pentru a le distinge de comenzile *PL/1*
- pentru a realiza integrarea este necesară:
  - o precompilare a cererilor
  - o interpretare la execuţie;

**Extensia limbajului de programare cu un LMD**

se consideră un limbaj de programare şi se realizează extensii ale acestuia cu clauze specifice limbajelor de prelucrare a datelor.
- Exemplu
  extensia *PASCAL/R* care permite definirea unui nou tip de date, şi anume tipul *RELATION*.

- Exemplu:

Să se definească relaţia *curs* utilizând structura *RELATION*

```
TYPE cf = RECORD
  cod_curs: TEXT;
  cod_profesor: TEXT;
  cod_student: TEXT;
  an: INTEGER;
  tip: TEXT
       END;
lect = RELATION OF cf;
VAR curs: lect;
```

- Exemplu

Să se obţină o relaţie având numele *cursFacultativ* care sa conţina toate cursurile de tip „facultativ” care se tin în anul II

```
VAR cursFacultativ: lect;
    …
    BEGIN
       cursFacultativ:=[];
       FOR EACH x IN curs DO
          IF(x.tip = facultativ')  AND (x.an = 2)
             THEN cursFacultativ:= cursFacultativ + [x]
    END;
```

Relaţia cursFacultativ poate fi construită şi direct prin atribuirea:

```
cursFacultativ:= [EACH (x) IN curs:
      (x.tip = facultativ')  AND  (x.an = 2)];
```

# *Curs 11 : Limbaje pentru prelucrarea datelor relationale*

1. Generalitati
2. Limbaje algebrice
3. Limbaje predicative
4. Utilizarea limbajelor de prelucrare a datelor relationale in contextul limbajelor de programare
5. *SQL*
6. Limbajul *MDX*

📖 ***SQL*** (*Structured Query Language*) =

= limbaj neprocedural pentru interogarea şi prelucrarea informaţiilor din BD

= limbaj declarativ:

- utilizatorul trebuie doar să descrie CE si NU CUM trebuie obţinut
- compilatorul limbajului *SQL* generează automat o procedură care accesează BD şi execută comanda

✓ *SQL* permite:

- definirea, prelucrarea şi interogarea datelor,
- controlul accesului la date
  - la nivel logic si
  - la nivel de multime

✓ Comenzile *SQL* pot fi integrate în programe scrise în alte limbaje, de exemplu *Cobol*, *C*, *C++*, *Java* etc.

## Clasificarea instrucţiunile *SQL* :

- limbajul de definire a datelor (*LDD*);
- limbajul de prelucrare a datelor (*LMD*);
- limbajul de control al datelor (*LCD*).
  - instrucţiuni pentru controlul sesiunii;
  - instrucţiuni pentru controlul sistemului;
  - instrucţiuni *SQL* încapsulate.

## Procesarea instrucţiunilor *SQL*

- ✓ Procesarea oricărei instrucţiuni *SQL* :
  - crearea unui cursor,
  - analiza instrucţiunii,
  - legarea variabilelor,
  - executarea instrucţiunii
  - închiderea cursorului
- ✓ In plus, interogările solicită câteva etape suplimentare:
  - descrierea rezultatelor,
  - definirea modului de prezentare a acestora
  - recuperarea liniilor selectate.

## Optimizarea instrucţiunilor *SQL*

📖 **Optimizorul**

= un modul *software* al sistemului *Oracle*, care determină modalitatea cea mai eficientă pentru execuţia unei instrucţiuni *SQL*

În sistemul *Oracle* există:

- optimizori pe bază de cost (in versiunile noi)
- optimizori pe bază de reguli;

Optimizorul trateaza fiecare instructiune / interogare conform unui plan de executie;

📖 **Schemă stocată** (*stored outline*) =

= este o abstractizare a unui plan de execuţie generat de către optimizor

Utilizatorul- programator poate modifica un plan de executie (poate edita schema stocata) transmitand optimizorului specificatii precise – legate de situatia concreta modelata) prin intermediul unor comentarii incluse in corpul instructiunii LMD, numite *hint*-uri.

# *Curs 11 : Limbaje pentru prelucrarea datelor relationale*

1. Generalitati
2. Limbaje algebrice
3. Limbaje predicative
4. Utilizarea limbajelor de prelucrare a datelor relationale in contextul limbajelor de programare
5. *SQL*
6. Limbajul *MDX*

📖 ***MDX*** (*Multidimensional Expressions*) =

= limbaj multidimensional specializat

- prima specificatie:
  - 1997,
  - firma Microsoft
  - ca parte componentă a specificaţiei *ODBO*[1]
  - urmată în 1998 de:
  - versiunea comercială *Olap Services 7.0* şi
  - produsul *Analysis Services;*

= limbaj de interogare a *BD OLAP* ~ *SQL* = limbajul de interogare BD relaţionale.

[1] ***ODBO = OLE DB for OLAP =*** *Object Linking and Embedding, Database for Online Analytical Processing*

***OLE*** **=** *Object Linking and Embedding* = tehnologie Microsoft de corelare a obiectelor (documente etc.)

***OLAP*** **=** *Online Analytical Processing* ∈ *Business Intelligence*

## 🕮 Interogarile *MDX* =

= interogari care se bazeaza pe cuburi *OLAP*

- intorc ca rezultate date agregate, aflate la intersecţia oricăror niveluri de agregare din oricare ierarhii, din oricare dimensiuni;

🕮 **Cubul** (*UDM – Unified Dimensional Model*) =

= o structură specială, logica nu fizica si care va fi interogată în cadrul unui sistem multidimensional

- poate răspunde rapid la interogări care cer o anumită agregare a datelor aflate la intersecţia oricăror niveluri din oricare ierarhii din oricate dimensiuni;

🔔 Scopul unei interogări *MDX:*

obţinerea unei mulţimi de date, numită **mulţime de celule** (*cellset*) sau **cub-rezultat** (i.e.: o submulţime a cubului original).

## 🕮 Interogarile *MDX* (cont.)

forma generală:

*SELECT* <specificaţie axa1> [, < specificaţie axa2>…]
*FROM* < specificaţie cub>
WHERE < specificaţie tăietură >

🕮 **Axa** =

= o colecţie de membri dintr-una sau mai multe dimensiuni ale cubului, organizată în tupluri

- utilizata în identificarea sau filtrarea valorilor specifice dintr-un cub de-a lungul dimensiunilor membrilor pe care îi reprezintă

⌛ Exemplu

se cauta numarul de studenti inscrisi la cursurile predate de fiecare cadru didactic din FMI, in functie de tipul de curs: obligatoriu, optional, facultativ

⇒ sunt necesare maximum 3 axe, fiecare reprezentând una dintre cele trei dimensiuni: *Student, Cadru_Didactic,Curs*

🔔 numărul de axe poate fi redus dacă dimensiunile sunt imbricate (*nested*).

# *Curs 11 : Limbaje pentru prelucrarea datelor relationale*

1. Generalitati
2. Limbaje algebrice
3. Limbaje predicative
4. Utilizarea limbajelor de prelucrare a datelor relationale in contextul limbajelor de programare
5. *SQL*
6. Limbajul *MDX*