Introducere în DML și LC

1. Obiectivul lucrării:

   Formarea abilităților de lucru cu DML și LC.

2. Breviar teoretic cu exerciții și probleme rezolvate

Comanda INSERT

Pentru a insera noi linii într-un tabel sau într-o vedere se utilizează comanda INSERT. O linie nouă inserată va ocupa în tabel o locaţie arbitrară (independenţa logică a datelor în bazele de date relaţionale). Comanda INSERT are una din următoarele două forme:

```
INSERT [INTO] denumire_tabel [(denumire_coloană[,…])]
VALUES (expresie [,… ])[,… ]
```

```
INSERT [INTO] denumire_tabel[(denumire_coloană[,…])]
  Comanda_Select
```

Prima formă permite inserarea directă a unui rând sau a mai multor rânduri într-un tabel al bazei de date, iar a doua formă permite inserarea mai multor rânduri care sunt rezultatul unei comenzi SELECT.

expresie poate fi o expresie constantă, **null**(dacă este acceptată) sau **default** (dacă este definită o valoare implicită).

Lista coloanelor trebuie să facă abstracţie de coloanele ale căror valori se autogenerează (coloane de tip *identity*, *timestamp* sau coloane calculate).
Dacă lista coloanelor lipseşte, atunci valorile din clauza *values* trebuie introduse în ordinea în care se găsesc coloanele în tabel. Nu se vor indica valori pentru coloanele de tip *identity*, *timestamp* sau calculate .

Dacă se introduc date doar în anumite coloane, atunci lista acestor coloane trebuie specificată, în plus în restul coloanelor se încarcă valoarea implicită (*null* sau valoarea definită prin clauza *default)*. De remarcat că valorile de tip caracter sau de tip dată calendaristică vor fi incluse între apostrofuri.

Dacă *Insert* nu respectă constrângerile de integritate sau de validare sau furnizează o valoare incompatibilă pentru o coloană, atunci întreaga operaţie eşuează şi se afişează un mesaj de eroare.

Exemplu

```
insert into tBonuriConsum (NrBon,DataBon,CodGest,CodConsumator)
     values (101,'03/10/08','g1','c2')

insert into tBonuriConsum (NrBon,CodGest,CodConsumator,DataBon)
```

```
     values (102,'g2','c1','16/12/08')
insert into tBonuriConsum (NrBon,CodGest,CodConsumator,DataBon)
     values (103,'g2','c1','16/12/08'), (104,'g2','c1','03/12/08')

insert into tBonuriConsum
     values (105,'17/15/02','g1','c1')

insert into tBonuriConsum
   (NrBon,CodGest,CodConsumator)
     values (106,'g2','c1')
```

Aici am presupus că este definită valoarea implicită pentru coloana *DataBon*:

```
DataBon datetime default getdate()
```

Comanda SELECT

Comanda SELECT este principalul instrument pentru regăsirea datelor dintr-o bază de date relaţională. SELECT este destul de complexă, dar principalele clauze se pot sintetiza astfel:

```
SELECT  [ALL | DISTINCT] [TOP n [Percent] [WITH TIES]]
      { * | lista_de_coloane_rezultat}
[INTO tabel_nou]
FROM   lista_de_tabele
[WHERE conditie]
[GROUP BY lista_de_expresii_de_grupare
 [HAVING  conditie]
]
[ORDER BY{expresie_de_ordonare [ ASC|DESC]}[,…]]
```

Clauzele **SELECT** şi **FROM** sunt obligatorii şi permit specificarea listei coloanelor prezente în rezultat şi a listei tabelelor din care vor fi consultate informaţii. Numele unui tabel poate fi urmat de un *alias* al acestuia. Restul clauzelor sunt opţionale, permiţând rafinarea comenzii şi oferind anumite servicii suplimentare.

Optiunea **ALL** este implicită şi permite selectarea tuturor liniilor,

Optiunea **DISTINCT** impune suprimarea dublurilor (de linii) din rezultat. La eliminarea duplicatelor valorile NULL se consideră ca fiind egale între ele.

TOP n [PERCENT] - vor fi returnate numai primele n tuple din rezultat. Dacă se specifică şi opţiunea PERCENT, atunci numai primele n procente din tuplele rezultatului sunt returnate. In acest caz n trebuie să fie între 0 şi 100. Dacă interogarea conţine clauza ORDER BY, atunci sunt returnate primele n tuple (sau n procente de tuple) din rezultatul ordonat după criteriile clauzei ORDER BY. În lipsa clauzei ORDER BY, ordinea tuplelor este arbitrară.

WITH TIES - dacă există mai multe tuple cu aceeaşi valoare a criteriilor din clauza ORDER BY şi sunt printre ultimele candidate pentru a face parte din cele TOP n (PERCENT) tuple, atunci ele vor fi incluse în rezultatul returnat. TOP ...WITH TIES se poate folosi numai în interogările care conţin clauza ORDER BY.

Clauza **INTO** *tabel_nou* creează un nou tabel (care nu trebuie să existe), unde se inserează relaţia rezultat.

Opţiunea * impune includerea în rezultat a tuturor coloanelor tabelelor menţionate în clauza FROM.

lista_de_coloane_rezultat este formată din coloane ale tabelelor clauzei FROM, coloane calculate sau funcţii agregat. Coloanele calculate sau obţinute pe baza funcţiilor agregat nu au denumiri implicite.

Denumirile coloanelor rezultat pot fi impuse prin construcţii de forma

expresie AS *alias_coloana*

sau

alias_coloana = expresie.

Dacă aliasul este format din mai multe cuvinte separate prin spaţiu va trebui să-l scriem între apostrofuri, ghilimele sau paranteze patrate

Exemple:

```
codCl AS CodClient
codF AS 'Cod furnizor'
codStd AS  "Cod student"
pret AS [Pret unitar]
cantitate * pret as valoare
"cod student"=codStd
'cod student'=codStd
[cod student]=codStd
```

alias_coloana poate fi folosit în clauza ORDER BY dar nu şi în clauzele WHERE, GROUP BY sau HAVING.

Denumirea de coloană trebuie prefixată cu denumirea sau aliasul tabelului din care provine ori de câte ori există ambiguităţi: ***tabel.coloană sau alias. coloană***

Construcţiile de forma *denumire_tabel.** sau *alias_tabel.** implică includerea în rezultat a tuturor coloanelor tabelului specificat

Sistemul SQL Server acceptă următoarele funcţii agregat (de grup): COUNT, SUM, MAX, MIN, AVG (valoare medie), VAR şi VARP (dispersia), STDEV şi STDEVP (deviaţia standard).

Clauza **FROM** specifică sursele din care se selectează tuplele (tabele, vederi, fraze SELECT imbricate). Clauza FROM este obligatorie ori de câte ori în clauza SELECT se face referire la cel puţin o coloană.

Clauza **WHERE** specifică o condiţie care este folosită pentru a selecta rândurile. Rândurile care nu satisfac condiţia specificată prin WHERE sunt ignorate.

Prin **GROUP BY** se constituie subansamble, iar prin clauza **HAVING** se selectează subansamble. Clauza GROUP BY se introduce dacă SELECT include funcţii(prelucrări) la nivel de grupuri de linii. In acest caz coloanele citate în clauza SELECT vor fi fie argumente ale funcțiilor de agregare(**sum, count, max, min, avg**, etc.), fie vor apărea în clauza GROUP BY pentru definirea nivelului de grupare. Grupurile care nu satisfac condiţia din clauza HAVING sunt eliminate din rezultat.

Clauza **ORDER BY** specifică ordinea în care trebuie returnate rândurile din rezultat. Ordinea implicită este ASC (crescătoare), iar pentru o secvenţă crescătoare valorile *null* sunt afişate ultimele. Dacă nu se face nicio specificaţie atunci ordinea de returnare a rândurilor este la latitudinea *server*-ului. Optiunea DISTINCT implică faptul că orice criteriu de ordonare din ORDER BY trebuie să fie prezent și în lista coloanelor rezultat a clauzei SELECT.

Funcţii de agregare

Funcţiile de agregare efectuează un anumit calcul asupra unui set de valori şi returnează ca rezultat o singură valoare. Cu excepţia funcţiei COUNT, toate funcţiile de agregare ignoră valorile NULL. Funcţiile de agregare sunt adesea folosite împreună cu clauza GROUP BY pentru a calcula valori agregate la nivelul grupurilor

de tuple. Folosirea funcţiilor de agregare este permisă numai în lista SELECT a unei fraze SELECT principală sau imbricată sau într-o clauză HAVING.
Principalele funcţii de agregare disponibile în SQL Server sunt:

Sintaxă	Semnificaţie
AVG([ALL | DISTINCT] expresie numerică)	media valorilor unui grup;
Count(*)	Numărul de elemente ale unui grup, cuprinzând toate duplicatele şi tuplele cu valori NULL
COUNT([ALL] expresie)	numărul de elemente ale unui grup - include toate duplicatele, dar fără valorile NULL;
COUNT(DISTINCT expresie)	numărul de elemente ale unui grup - fără duplicate şi fără valori NULL.
MAX(expresie)	maximul dintr-un grup de valori.
MIN (expresie)	minimul dintr-un grup de valori.
SUM([ALL | DISTINCT] expresie numerică)	suma valorilor unui grup.
VAR(expresie_numerică)	returnează dispersia estimată pentru valorile expresiei dată ca argument: $\frac{1}{n-1}\sum(x-\bar{x})^2$, $\bar{x}$ reprezintă media valorilor.
VARP(expresie_numerică)	returnează dispersia calculată pentru valorile expresiei dată ca argument: $\frac{1}{n}\sum(x-\bar{x})^2$, $\bar{x}$ reprezintă media valorilor.
STDEV(expresie_numerică)	returnează deviaţia standard estimată pentru valorile expresiei dată ca argument. $\sqrt{\frac{1}{n-1}\sum(x-\bar{x})^2}$, $\bar{x}$ reprezintă media valorilor.
STDEVP(expresie_numerică)	returnează deviaţia standard calculată pentru valorile expresiei dată ca argument. $\sqrt{\frac{1}{n}\sum(x-\bar{x})^2}$, $\bar{x}$ reprezintă media valorilor.

unde:
ALL - indică faptul că agregarea se aplică tuturor valorilor. ALL este opţiunea implicită.

DISTINCT - specifică faptul că agregarea se aplică doar asupra unei singure instanţe a fiecărei valori din grup, duplicatele fiind ignorate.
expresie - este o expresie de orice tip. Expresia nu poate conţine funcţii de agregare sau fraze SELECT imbricate.

```
select * from tAngajati

select CNP,nume,Prenume from tAngajati

select Nume+' '+Prenume as "Nume complet"  from tAngajati

select year(DataNasterii) as An from tAngajati

select distinct year(DataNasterii) as An from tAngajati

select Nume,Prenume,year(DataNasterii) as "Anul nasterii"
from tAngajati

select Nume,Prenume,year(DataNasterii) as "Anul nasterii"
from tAngajati
where year(DataNasterii)>1980
order by year(DataNasterii) desc

select COUNT(Nume) "Nr. angajati",
       year(DataNasterii) as "Anul nasterii"
from tAngajati
group by year(DataNasterii)
```