Tutoriat 7

UPDATE; ALTER TABLE; RENAME; ROLLBACK; COMMIT; VARIABILE DE SUBSTITUTIE;

In ultimul tutoriat am vazut cum putem sa cream, sa stergem si sa inseram date intr-un table. In acest tutoriat o sa vedem cum putem edita atat datele dintr-un tabel cat si structura sa. De asemenea vom invata si cum pot fi citite date de la tastatura.

UPDATE

Comanda UPDATE este folosita pentru a altera date dintr-un tabel. Ea are sintaxa:

```
1. UPDATE nume_tabel
2. SET nume_col1 = val1,
3.     nume_col2 = val2,
4.     ...
5.     nume_colN = valN
6. WHERE conditie;
```

Prin aceasta comanda liniile tabelului ce respecta coditia din WHERE vor fi updatate cu valorile din UPDATE. Nu trebuie specificate valori noi pentru toate coloanele, coloanele nenumite isi vor pastra vechea valoare.

ATENTIE: Clauza WHERE este optionala, daca aceasta lipseste TOATE coloanele din tabel specificate in UPDATE o sa fie updatate cu noile valori.

Sa exemplificam utilizarea lui UPDATE crescand salariile tuturor angajatiilor cu salari mai mici de 5000 cu 2000:

```
1. UPDATE employees
2. SET salary = salary + 2000
3. WHERE salary < 5000;
```

De asemenea UPDATE se poate folosi pentru a atribui datele rezultate dintr-o cerere uneia sau mai multor linii dintr-un tabel astfel:

```
1. UPDATE nume_tabel
2. SET (nume_col1, nume_col2, ..., nume_colN) = (SELECT...)
3. WHERE conditie;
```

In continuare sa setam salariul lui Steven King drept media salariilor tuturor angajatiilor:

```
1. UPDATE emp_rap
2. SET salary = (SELECT AVG(salary) FROM employees)
3. WHERE UPPER(first_name) = 'STEVEN'
4.    AND UPPER(last_name) = 'KING';
```

ALTER TABLE

Daca dorim sa adaugam, editam sau stergem coloane dintr-un tabel atunci putem folosi comanda ALTER TABLE pentru a face acest lucru. Sintaxa este:

```
1. --adaugare de coloana noua
2. ALTER TABLE nume_tabel
3. ADD(nume_coloana tip_de_date constrangere1, ..., constrangereN);
4.
5. --modificarea unei coloane deja existente
6. ALTER TABLE nume_tabel
7. MODIFY(nume_coloana tip_de_date constrangere1, ..., constrangereN);
8.
9. --eliminarea unei coloane
10. ALTER TABLE nume_tabel
11. DROP COLUMN nume_coloana
```

Unde constrangerile pot fi: NOT NULL, UNIQUE, PRIMARY KEY, etc. … (vezi Tutoriat 6).

Sa exemplificam acest lucru adaugand o noua coloana unica CNP in tabelul EMPLOYEES si sa stergem coloana email:

```
1. --adaug noua coloana
2. ALTER TABLE employees
3. ADD(CNP VARCHAR2(13) UNIQUE);
4.
5. --sterg coloana "email"
6. ALTER TABLE employees
7. DROP COLUMN email;
```

De asemenea ALTER TABLE mai poate fi folosit si pentru editarea constrangerile coloanelor :

```
1. --adaugare de constrangere noua
2. ALTER TABLE nume_tabel
3. ADD CONSTRAINT nume_constrnagere tip_constrangere(nume_coloana);
4.
5. --stergerea unei constrangeri
6. --Metoda 1
7. ALTER TABLE nume_tabel
8. DROP tip_constrangere(nume_coloana);
9.
```

```
--Metoda 2
ALTER TABLE nume_tabel
DROP CONSTRAINT nume_constrangere;

--pentru activarea/dezactivare de constrangere
--Metoda 1
ALTER TABLE nume_tabel
MODIFY CONSTRAINT nume_constrangere ENABLE/DISABLE

--Metoda 2
ALTER TABLE nume_tabel
ENABLE/DISABLE CONSTRAINT nume_constrangere
```

La adaugarea de constrangeri numele constrangerii(“CONSTRAINT nume_constrangere”) este optional. Totusi, daca constrangerea nu are nume, aceasta nu va putea fi dezactivata si reactivata, doar stearsa.

RENAME

Daca dorim sa redenumim un tabel o putem face astfel:

```
RENAME nume_tabel_original
TO nume_tabel_nou;
```

COMMIT SI ROLLBACK

Odata ce un UPDATE, ALTER TABLE sau RENAME a fost facut datele alterate nu sunt salvate permanent, numai local si numai pe durata sesiunii curente. Daca dorim sa salvam toate schimbarile permanent in baza de date comanda COMMIT trebuie rulata.

```
COMMIT;
```

Daca am facut o gresela si dorim sa intoarcem tabelele din sesiunea curenta la ultima stare in care s-a facut COMMIT(un fel de undo) sau, daca aceasta nu exista, la starea in care erau cand a inceput sesiunea atunci putem rula comanda ROLLBACK.

```
ROLLBACK;
```

VARIABILE DE SUBSTITUTIE

Variabilele de substitutie reprezinta un mod simplu de a putea citi date de la tastatura in timpul unei cereri. Acestea pot fi folsite in 3 moduri:

- Folosind comanda DEFINE (exemplu DEFINE nume_variabila = valoare). In acest caz trebuie folosita si comanda UNDEFINE pentru a sterge variabila. Variabila va pastra valoarea data la declarare.
- Prefixand “&” la numele variabilei (exemplu : &variabile, &p_salariu). Daca variabila nu a fost declarata atunci ea va exista numai pe durata cererii. Mereu cand se prefixeaza “&” unei variabile aceasta o sa fie citita de la tastatura daca nu a fost definita cu DEFINE.
- Prefixand “&&” la numele variabilei(exemplu: &&variabila, &&p_salariu). Folosit pentru a semnala memorarea valorii, astfel variabila va lua mereu valoarea primita prima data de la tastature(daca nu a fost definita).

Sa exemplificam aceste moduri afisand toti angajatii ce au salariul mai mare decat un numar citit de la tastatura.

```
1.  --Metoda 1(correct)
2.  SELECT *
3.  FROM employees
4.  WHERE salary > &salariu;
5.
6.  --Metoda 2(gresit)
7.  DEFINE salariu = 1000;
8.  SELECT *
9.  FROM employees
10. WHERE salary > &salariu;
11. UNDEFINE UNDEFINE salariu;
```

Prima metoda citeste variabila “salariu” de la tastatura deoarce aceasta nu a fost definite inainte si afiseaza toti angajatii cu salarii mai mari decat ea .

A doua metoda nu rezolva problema. Deoarece definesc variabila inainte sa o folosesc aceasta are o valoare declarata ce va fi folosita, astfel nu se va mai citi de la tastatura.

In continuare sa afisam toti angajatii al caror nume si prenume incep cu aceasi litera citita de la tastatura:

```
1.  --Metoda 1(gresit)
2.  SELECT *
3.  FROM employees
4.  WHERE UPPER(first_name) LIKE '&p_litera%'
5.    AND UPPER(last_name) LIKE '&p_litera%';
```

```
6.
7.  --Metoda 1(corect)
8.  SELECT *
9.  FROM employees
10. WHERE UPPER(first_name) LIKE '&&p_litera%'
11.   AND UPPER(last_name) LIKE '&p_litera%';
```

Metoda 1 este gresita deoarece implica citirea de 2 ori de la tastatura a variabilei(liniile 4 si 5) deoarece foloseste mereu prefixarea cu “&”.

Metoda a 2-a prefixeaza prima data cu “&&” numele variabilei. Astfel compilatorul este semnalat ca valoarea citita la linia 10 trebuie pastrata si refolosita, astfel este citita numai o data de la tastatura.

Un alt lucru demn de mentionat este prezenta variabilelor in string-uri. Compilatorul isi poate da seama cand o variabila se afla intr-un string si o inlocuieste cu valoarea sa (sau o citeste de la tastatura daca valoarea nu exista).

Exercitii se pot gasi pe Drive -> BD 2018-2019 -> Laborator6-> Exercitiile 17 - 23

-> Laborator7-> Exercitiile 14-21

->Laborator8->Exercitiile 4-37