Tutoriat 6

CREATE; DROP; INSERT; DELETE; TRUNCATE;

CREATE

Pana acum am invatat cum se pot accesa date din baza de date dupa anumite cerinte. In acest tutoriat o sa vedem cum putem creea,insera si sterge tabele si linii din ea.

Pentru inceput sa ne ocupam de creearea unui tabel. Primul lucru ce trebuie specificat este numele acestuia urmat de numele si tipul coloanelor sale. Coloanele pot fi:

- VARCAHR2(n) = Un sir de caractere de dimensiune maxima n.
- CHAR(n) = Un sir de caractere de fix lungimea n (folositor daca vrema sa retinem date ale caror dimensiune le stim cum ar fi CNP-ul).
- NUMBER(n) = Numar de maxim n cifre
- NUMBER(n,m) = Numar de maxim n cifre dintre care m cifre reprezinta partea zecimala
- DATE = O data calendaristica din intervalul 1.01.4712 i.Hr. si 31.12.9999 d.Hr.
- LONG = sir de caractere de dimensiune maxima de 2GB.

De asemenea fiecare tabel poate avea una sau mai multe constrangeri ce se aplica fie la nivel de coloana fie la nivel de tabel(mai multe coloane). Acestea pot fi:

- NOT NULL = coloana nu are voie sa aiba valoarea (null)
- UNIQUE = nicio valoare de pe coloana nu are voie sa se repete. (poate fi folosita si la nivel de tabel, pentru mai multe coloane, daca dorim ca combinatii de de valori de pe mai multe coloane sa nu se repete)
- PRIMARY KEY = coloana este cheie primara. Ea devene automat si NOT NULL + UNIQUE (poate fi folosita la nivel de tabel, pentru mai multe coloane, daca dorim ca tabelul sa aiba cheie primara multipla).
- FOREIGN KEY = coloana este o cheie externa. Trebuie specificat tabelul si coloana pentru care aceasta este cheie externa(constrangere la nivel de tabel).
- CHECK = conditie ce trebuie respectata pentru fiecare coloana sau linie(valabil numai la nivel de tabel).

Sintaxa pentru creearea unui tabel este:

```
CREATE TABLE nume_tabel(
    nume_coloana1 tip_date DEFAULT valoare constrangere_nivel_coloana1 ... con
stranere_nivel_coloanaN,
    nume_coloana2 tip_date DEFAULT valoare constrangere_nivel_coloana1 ... con
stranere_nivel_coloanaN,
    ...
    nume_coloanaN tip_date DEFAULT valoare constrangere_nivel_coloana1 ... con
stranere_nivel_coloanaN,

constrangere_nivel_tabel1
...
constrangere_nivel_tabelN
);
```

Sa cream 2 tabele pentru a intelege mai bine sintaxa:

Sa cream un tabel cu numele ANIMAL ce contine: id_persoana(PK, NUMBER(4)), nume(NOT NULL,VARCHAR(50)), rasa(NOT NULL,VARCHAR(50)), varsta(NOT NULL, NUMBER(2)) si greutate(NUMBER(3)). De asemenea numele va avea o valoare DEFAULT „John Doe”.

```
CREATE TABLE Animal(
  id_animal NUMBER(4) PRIMARY KEY,
  nume VARCHAR(50) DEFAULT 'John Doe' NOT NULL,
  rasa VARCHAR(50) NOT NULL,
  varsta NUMBER(2) NOT NULL,
  greutate NUMBER(3)
);
```

In contiunare sa cream un tabel PERSOANA ce contine: id_persoana(PK,NUMBER(10)), nume(NOT NULL,VARCHAR(50)), data_nasterii(DATE) si animal(FK,VARCHAR(50)). Acesta este legat de tabelul ANIMAL prin id_animal:

```
CREATE TABLE Persoana(
    id_persoana NUMBER(4) PRIMARY KEY,
    nume VARCHAR(50) NOT NULL,
    data_nasteri DATE,
    animal NUMBER(4),
FOREIGN KEY(animal) REFERENCES Animal(id_animal)
);
```

De asemenea, in cazul in care dorim sa cream un tabel ce sa contina rezultatele unei cereri, putem folosi operatorul AS:

```
CREATE TABLE nume_tabel AS (SELECT ...);
```

Astfel noul tabel va contine toate coloanele prezente in cerere inpreuna cu constrangerile (excluzand cele de cheie primara si externa) si tipurile lor de date. Daca selectam toate datele dintr-un tabel intr-o astfel de cerer putem face o copie exacta a acestuia.

DROP

Pentru a sterge un tabel din baza de date se foloseste comanda DROP TABLE. Impreuna cu tabelul toate datele din acesta o sa fie sterse. Sintexa este:

```
1. DROP TABLE nume_tabel;
```

Aceasta comanda este des folosita pentru a sterge tabele de care nu mai este nevoie.

INSERT

INSERT este comanda folosita pentru a insera o linie intr-un tabel. Sintaxa este:

```
1. INSERT INTO nume_tabel (nume_col1, nume_col2, ..., nume_colN) VALUES( val1,
   val2, ..., valN);
```

In comanda specificam numele tabelului in care dorim sa inseram o linie, numele coloanelor in ordinea in care dorim sa scriem valorile si lista de valori ce o sa fie inserate.Daca numele unei coloane nu apare valoarea coloanei repsective va fi setata automat ca fiind null. Daca coloana are constrangerea NOT NULL se va semnala o eroare.De asemenea numele coloanelor este optionala daca valorile sunt scrise in ordinea din declararea tabelului.

Sa exemplificam comanda INSERT adaugand valori in tabelul ANIMAL :

```
1. INSERT INTO Animal (nume, varsta, greutate, rasa, id_animale) VALUES('Jesse',6
   ,12, 'Caine',1 );
2.
3. INSERT INTO Animal VALUES(2,'Dante', 'Caine', 4, 8);
```

In primul INSERT am putut insera datele in ce ordine am dorit deoarece am specificat aceasta ordine inainte. In al doilea INSERT ordinea datelor a fost aceea cu ordinea de declarea a coloanelor (id,nume,rasa,varsta,greutate) si de aceea nu a mai trebuit specificata.

Comanda INSERT poate fi folosita si pentru a insera rezultatul unei cereri intr-un tabel astfel:

```
1. INSERT INTO nume_tabel (nume_col1, nume_col2, ..., nume_colN) ( SELECT ...);
```

ATENTIE: Ordinea si tipul de date al coloanelor care sunt declarate in SELECT este aceasi cu ce-a a coloanelor care sunt enumerate. Daca tipurile de date nu corespund se va semnala o eroare. Asemanator cu INSERT-ul de mai sus daca lista de coloane nu este indicata atunci se va folosi ordine precizata in declararea tabelului.

In continuare sa adaugam toti angajatii din tabelul EMPLOYEES ce au salariul mai mare de 8000 in tabelul PERSOANA folosind ambele metode.

```
1. INSERT INTO Persoana (id_persoana, nume) (SELECT employee_id, first_name ||' '
   || last_name
2.                                          FROM employees);
3.
4. INSERT INTO Persoana (SELECT employee_id, first_name ||' '|| last_name , null

5.                       FROM employees);
```

In primul INSERT am folosit lista de coloane ce mi-a permis sa inserez datele in ordinea specificata, toate datele nenumite avand valoarea null. Pentru al doilea INSERT ordinea din SELECT a trebuit sa fie aceasi cu ce-a din declararea tabelului (id_persoana, nume, id_animal), de aceea am adaugat o coloana null in SELECT, pentru a fi introdusa drept valoarea coloanei id_animal.

DELETE

DELETE este comanda inversa lui INSERT. Aceasta sterge toate liniile dintr-un tabel ce indeplinesc o anumita conditie. Sintaxa este:

```
1. DELETE FROM nume_tabel WHERE conditie
```

Daca conditia nu exista toate datele din tabel o sa fie sterse.

Sa exemplificam aceasta comanda facand o copie a tabelului EMPLOYEES si stergand toti angajatii al caror salariu este mai mare de 10000.

```
1. --creez copia
2. CREATE TABLE emp AS (SELECT * FROM employees);
3.
4. --sterg angajatii dupa conditie
5. DELETE FROM emp WHERE salary > 10000;
```

TRUNCATE

Asemanator lui DELETE fara conditie, comanda TRUNCATE TABLE o sa stearga toate datele dintr-un tabel. Sintaxa este:

```
1. TRUNCATE TABLE nume_tabel;
```

Exercitii se pot gasi pe Drive -> BD 2018-2019 -> Laborator7-> Exercitiile 1 - 31

-> Laborator6-> Exercitiile 1,2,3