Functii SQL de baza

March 20, 2025

Contents

1 SELECT

Instructiunea `SELECT` este folosita pentru a extrage date din baza de date.

```
SELECT coloana1, coloana2, ...
FROM nume_tabela;
```

1.1 Exemple de utilizare

```
-- Selecteaza toate coloanele din tabela "studenti"
SELECT * FROM studenti;

-- Selecteaza doar numele si prenumele studentilor
SELECT nume, prenume FROM studenti;
```

1.2 Utilitate

Instructiunea `SELECT` este fundamentala in SQL si reprezinta punctul de plecare pentru majoritatea interogarilor. Aceasta permite specificarea exacta a datelor care trebuie extrase.

2 WHERE

Clauza `WHERE` este folosita pentru a filtra inregistrarile pe baza unei conditii.

```
SELECT coloana1, coloana2, ...
FROM nume_tabela
WHERE conditie;
```

2.1 Exemple de utilizare

```
-- Selecteaza studentii cu varsta mai mare de 20 de ani
SELECT nume, prenume, varsta
FROM studenti
WHERE varsta > 20;

-- Selecteaza studentii cu numele "Popescu"
SELECT * FROM studenti
WHERE nume = 'Popescu';
```

2.2 Operatori in clauza WHERE

- `=` : Egal
- `>` : Mai mare decat
- `<` : Mai mic decat
- `>=` : Mai mare sau egal cu
- `<=` : Mai mic sau egal cu
- `<>` sau `!=` : Diferit de
- `BETWEEN` : Intre un interval
- `LIKE` : Cauta un model
- `IN` : Specifica valori multiple

3 ORDER BY

Clauza `ORDER BY` este folosita pentru a sorta rezultatele in ordine ascendenta sau descendenta.

```
SELECT coloana1, coloana2, ...
FROM nume_tabela
ORDER BY coloana1 [ASC|DESC], coloana2 [ASC|DESC], ...;
```

3.1 Exemple de utilizare

```
-- Sorteaza studentii in ordine alfabetica dupa nume
SELECT nume, prenume
FROM studenti
ORDER BY nume ASC;

-- Sorteaza studentii dupa varsta (descrescator) si apoi dupa nume (crescator)
SELECT nume, prenume, varsta
FROM studenti
ORDER BY varsta DESC, nume ASC;
```

3.2 Utilitate

Clauza `ORDER BY` este esentiala pentru prezentarea datelor intr-un mod organizat si usor de citit.

4 DISTINCT

Cuvantul cheie `DISTINCT` este folosit pentru a elimina duplicatele din rezultate.

```
SELECT DISTINCT coloana1, coloana2, ...
FROM nume_tabela;
```

4.1 Exemple de utilizare

```
-- Selecteaza toate facultatile distincte
SELECT DISTINCT facultate
FROM studenti;

-- Selecteaza toate combinatiile distincte de facultate si an
SELECT DISTINCT facultate, an
FROM studenti;
```

4.2 Utilitate

`DISTINCT` este util cand dorim sa eliminam duplicatele si sa obtinem doar valorile unice.

5 GROUP BY

Clauza `GROUP BY` este folosita pentru a grupa randurile care au aceleasi valori.

```
SELECT coloana1, coloana2, ..., functie_agregare(coloana)
FROM nume_tabela
GROUP BY coloana1, coloana2, ...;
```

5.1 Exemple de utilizare

```sql
-- Numarul de studenti per facultate
SELECT facultate, COUNT(*) as numar_studenti
FROM studenti
GROUP BY facultate;

-- Media varstei studentilor per facultate si an
SELECT facultate, an, AVG(varsta) as medie_varsta
FROM studenti
GROUP BY facultate, an;
```

5.2 Functii de agregare folosite cu GROUP BY

- `COUNT()` : Numara randurile
- `SUM()` : Calculeaza suma
- `AVG()` : Calculeaza media
- `MIN()` : Gaseste valoarea minima
- `MAX()` : Gaseste valoarea maxima

6 HAVING

Clauza `HAVING` este folosita pentru a filtra rezultatele grupate.

```sql
SELECT coloana1, coloana2, ..., functie_agregare(coloana)
FROM nume_tabela
GROUP BY coloana1, coloana2, ...
HAVING conditie;
```

6.1 Exemple de utilizare

```sql
-- Facultatile cu mai mult de 100 de studenti
SELECT facultate, COUNT(*) as numar_studenti
FROM studenti
GROUP BY facultate
HAVING COUNT(*) > 100;

-- Grupuri de studenti cu varsta medie peste 22 de ani
SELECT facultate, an, AVG(varsta) as medie_varsta
FROM studenti
GROUP BY facultate, an
HAVING AVG(varsta) > 22;
```

6.2 Diferenta intre WHERE si HAVING

`WHERE` filtreaza randurile inainte de grupare, in timp ce `HAVING` filtreaza grupurile dupa grupare.

7 LIMIT

Clauza `LIMIT` este folosita pentru a limita numarul de randuri returnate.

```sql
SELECT coloana1, coloana2, ...
FROM nume_tabela
LIMIT numar;
```

7.1 Exemple de utilizare

```sql
-- Primii 10 studenti
SELECT nume, prenume
FROM studenti
LIMIT 10;

-- Studentii de la pozitia 11 pana la pozitia 20
SELECT nume, prenume
FROM studenti
LIMIT 10 OFFSET 10;
```

7.2 Utilitate

`LIMIT` este util pentru paginare si pentru a limita cantitatea de date returnate.

8 Functii de agregare

Functiile de agregare efectueaza calcule pe un set de valori si returneaza o singura valoare.

8.1 Tipuri de functii de agregare

```sql
-- COUNT() - Numara randurile
SELECT COUNT(*) FROM studenti;
SELECT COUNT(DISTINCT facultate) FROM studenti;

-- SUM() - Calculeaza suma
SELECT SUM(credite) FROM cursuri;

-- AVG() - Calculeaza media
SELECT AVG(nota) FROM examene;

-- MIN() - Gaseste valoarea minima
SELECT MIN(varsta) FROM studenti;

-- MAX() - Gaseste valoarea maxima
SELECT MAX(nota) FROM examene;
```

8.2 Utilizare cu GROUP BY

Functiile de agregare sunt deseori folosite impreuna cu `GROUP BY` pentru a efectua calcule pe grupuri de date.

```sql
SELECT facultate,
       COUNT(*) as numar_studenti,
       AVG(varsta) as medie_varsta,
       MIN(varsta) as varsta_minima,
       MAX(varsta) as varsta_maxima
FROM studenti
GROUP BY facultate;
```

9 LIKE

Operatorul `LIKE` este folosit pentru a cauta un model specific in date.

```sql
SELECT coloana1, coloana2, ...
FROM nume_tabela
WHERE coloana LIKE pattern;
```

9.1 Caractere speciale in LIKE

- % : Reprezinta zero, unul sau mai multe caractere
- _ : Reprezinta un singur caracter

9.2 Exemple de utilizare

```
-- Gaseste studentii al caror nume incepe cu "Pop"
SELECT * FROM studenti
WHERE nume LIKE 'Pop%';

-- Gaseste studentii al caror prenume are "a" ca a doua litera
SELECT * FROM studenti
WHERE prenume LIKE '_a%';

-- Gaseste cursurile care contin cuvantul "date" in titlu
SELECT * FROM cursuri
WHERE titlu LIKE '%date%';
```

10 IN

Operatorul `IN` permite specificarea mai multor valori intr-o clauza WHERE.

```
SELECT coloana1, coloana2, ...
FROM nume_tabela
WHERE coloana IN (valoare1, valoare2, ...);
```

10.1 Exemple de utilizare

```
-- Gaseste studentii din facultatile de Informatica sau Matematica
SELECT * FROM studenti
WHERE facultate IN ('Informatica', 'Matematica');

-- Gaseste cursurile cu ID-urile 101, 205 si 310
SELECT * FROM cursuri
WHERE id_curs IN (101, 205, 310);
```

10.2 Utilizare cu subinterogari

`IN` poate fi folosit si cu subinterogari.

```
-- Gaseste studentii care au cel putin un examen cu nota 10
SELECT * FROM studenti
WHERE id_student IN (
    SELECT id_student
    FROM examene
    WHERE nota = 10
);
```

11 BETWEEN

Operatorul `BETWEEN` selecteaza valori intr-un interval specificat.

```
SELECT coloana1, coloana2, ...
FROM nume_tabela
WHERE coloana BETWEEN valoare1 AND valoare2;
```

11.1 Exemple de utilizare

```sql
-- Gaseste studentii cu varsta intre 20 si 25 de ani
SELECT * FROM studenti
WHERE varsta BETWEEN 20 AND 25;

-- Gaseste examenele din perioada 2023-01-01 pana la 2023-01-31
SELECT * FROM examene
WHERE data_examen BETWEEN '2023-01-01' AND '2023-01-31';
```

11.2 NOT BETWEEN

Operatorul `NOT BETWEEN` selecteaza valori in afara intervalului specificat.

```sql
-- Gaseste studentii cu varsta in afara intervalului 20-25
SELECT * FROM studenti
WHERE varsta NOT BETWEEN 20 AND 25;
```

12 Concluzie

Functiile SQL prezentate in acest document sunt esentiale pentru interogarea bazelor de date si reprezinta baza pentru interogari mai complexe care implica JOIN-uri. Intelegerea acestor functii este cruciala pentru a putea scrie interogari eficiente si pentru a manipula datele in mod corespunzator.

- `SELECT` extrage date din baza de date
- `WHERE` filtreaza inregistrarile pe baza unei conditii
- `ORDER BY` sorteaza rezultatele
- `DISTINCT` elimina duplicatele
- `GROUP BY` grupeaza randurile care au aceleasi valori
- `HAVING` filtreaza grupurile
- `LIMIT` limiteaza numarul de randuri returnate
- Functiile de agregare efectueaza calcule pe un set de valori
- `LIKE` cauta un model specific
- `IN` specifica valori multiple
- `BETWEEN` selecteaza valori intr-un interval