Laborator 4

Instrucțiuni pentru interogarea bazelor de date în MySQL

Instrucţiunea SELECT reprezintă blocul de interogare de bază şi selectează informaţiile dorite din tabelele bazei de date. Ca rezultat al instrucţiunii SELECT se pot obţine una sau mai multe înregistrări după un criteriu dat.

Toate spaţiile albe care apar în instrucţiune vor fi ignorate în momentul procesării acesteia. De asemenea, instrucţiunea poate fi scrisă pe un rând sau despărţită pe mai multe rânduri, lucru recomandat pentru o mai bună urmărire a acţiunilor care vor fi defăşurate. Denumirea instrucţiunii poate fi scrisă SELECT, Select sau select, programul considerându-le unul şi acelaşi lucru.

Instrucțiunea SELECT

```
SELECT [ALL | DISTINCT | DISTINCTROW] <nume_coloana>
[, <expresie> ...]
FROM <nume_tabelă>
[WHERE <condiţie_de_căutare>]
[GROUP BY {<expresie_de_grupare>}[ASC | DESC]
[HAVING <condiţii_de_selecţie_grup>]]
[ORDER BY <expresie_de_ordonare> [ASC | DESC]]
```

În general, clauzele folosite trebuie să fie în aceeaşi ordine prezentată în descrierea sintaxei. Fiecare atribut nume_coloana indică o coloană ce se doreşte a fi returnată, fiecare expresie fiind necesar să aparţină uneia dintre tabelele specificate în clauza FROM nume_tabele. Dacă lista de tabele (definită de clauza FROM) conţine mai mult de o tabelă, atunci numele coloanelor din clauza SELECT trebuie să fie diferit.

Urmărind cuvintele cheie din clauza SELECT, se pot folosi anumite opțiuni care vor determina valorile returnate.

- Opțiunile **ALL** și **DISTINCT** specifică dacă rândurile duplicat vor fi returnate. ALL este opțiunea predefinită și specifică faptul că vor fi returnatele și înregistrările duplicat. DISTINCT va determina returnarea numai a rândurilor unice. Se consideră că folosirea ambelor opțiuni este o eroare. DISTINCTROW este sinonim cu DISTINCT.
- Folosind *****, vor fi incluse toate atributele din toate tabelele sau vederile menţionate în clauza FROM. Atributele vor apare în ordinea specificării tabelelor şi vederilor din clauza FROM şi după aceea în ordinea în care se află atributele în tabele şi vederi.

- **<nume_coloană>** – este numele coloanei a cărui valori se doresc a fi introduse în rezultat. În cazul în care numele tabelelor nu sunt diferite între ele, atunci se califică numele coloanei cu numele tabelei căreia îi aparţine (precedând numele atributului cu numele tabelei urmat de operatorul "punct" (.)).
- **<expresie>** – poate fi numele de coloană, o constantă, o funcţie, orice combinaţie a acestora realizată prin operatori ori poate fi o frază SELECT imbricată.

- **Clauza FROM**

FROM <nume_tabelă> - specifică tabelele, vederile sau frazele SELECT care constituie sursa de tuple pentru fraza SELECT.

FROM <nume_tabelă> [[AS] <alias_tabelă>] – specifică numele tabelei şi un alias opţional

- **Clauza WHERE**

Clauza **Where** indică condiția sau condițiile care trebuie îndeplinite de către înregistrări pentru a fi selectate. <**condiţie_căutare**> este condiția care trebuie îndeplinită pentru ca o anumită înregistrare să fie afișată.

{[NOT] <predicat> | <condiţie_căutare>}
[{AND|OR} [NOT] {<predicat>| <condiţie_căutare>}], unde <predicat> poate lua următoarele valori:
<expresie> {= | <> | != | > | >= | !> | < | <= | !<} <expresie>
| <expresie_şir> [NOT] LIKE <expresie_şir>
| <expresie> [NOT] BETWEEN <expresie>AND <expresie>
| <expresie>IS [NOT] NULL
| <expresie> [NOT] IN (<subinterogare> | <expresie>)
| <expresie>{= | <> | != | > | >= | !> | < | <= | !<} {ALL | SOME | ANY} (<subinterogare>)
| EXISTS (<subinterogare>)

Semnificaţia opţiunilor pentru clauza WHERE este următoarea:

- **<expresie>** - este un nume de coloană, o constantă, o funcţie, o variabilă, o subinterogare
- **<expresie_şir>** - un şir de caractere.
- **<subinterogare>** - este o frază SELECT restricţionată, care nu poate să conţină clauzele ORDER BY sau INTO.
- **[NOT] LIKE** – este operatorul de comparare inexactă pentru şiruri de caractere.
- **[NOT] BETWEEN** – specifică un interval închis de valori

- **IS [NOT] NULL** – specifică o căutare de valori NULL (sau diferită de NULL, în cazul în care se specifică cuvântul cheie NOT).
- **[NOT] IN** – specifică o căutare a unei expresii (constantă sau nume de coloană) într-o listă de valori constante sau în relaţia rezultat returnată de o subinterogare.
- **ALL** – poate fi folosit exclusiv ca prefix al unei subinterogări în operaţii de comparare. Predicatul în care este inclus returnează TRUE dacă toate valorile din rezultatul returnat de subinterogare satisfac condiţia de comparare şi FALSE, în caz contrar.
- **SOME | ANY** – poate fi folosit exclusiv ca prefix al unei subinterogări în operaţii de comparare. Predicatul în care este inclus returnează TRUE dacă cel puţin una dintre valorile din rezultatul returnat de subinterogare satisface condiţia de comparare şi FALSE, dacă nici o valoare nu satisface condiţia de comparare sau dacă subinterogarea returnează ca rezultat o relaţie vidă.
- **EXISTS** – poate fi folosit exclusiv ca prefix al unei subinterogări şi returnează TRUE dacă rezultatul returnat de subinterogare conţine cel puţin o tuplă şi FALSE, în caz contrar.

- **Clauza GROUP BY**

Această clauză specifică grupurile care se formează din înregistrările selectate prin caluzele anterioare. Elementele din clauza SELECT trebuie să producă o valoare unică la nivel de grup. Orice funcţie de agregare care apare în funcţia SELECT acţionează la nivelul fiecărui grup în parte calculând agregarea corespunzătoare pentru înregistrările care fac parte din grup.[9]

GROUP BY [ALL] <expresie[,n]>

- **ALL**- indică faptul că toate grupurile vor fi incluse în relaţia rezultat, inclusiv cele care nu conţin nici o înregistrare care să satisfacă condiţiile din clauza WHERE
- **<*expresie*>** -este expresia după care se face gruparea

- **Clauza HAVING**

Specifică o condiţie de căutare la nivelul grupurilor. Aceasta poate fi folosită în sintaxa SELECT doar dacă există anterior o clauză GROUP BY. O frază SELECT care conţine clauzele WHERE, GROUP BY şi HAVING este prelucrată în ordinea apariţiei acestor clauze:întâi se elimină înregistrările care nu satisfac condiţia din clauza WHERE, apoi se grupează după expresia din GROUP BY şi se elimină grupurile care nu satisfac condiţia din HAVING.

HAVING <conditie_de_cautare>

- **<conditie_de_cautare>**-este simulată cu WHERE, cu menţiunea că este exprimată doar în termenii acelor elemente care au o valoare unică la nivel de grup

- **Clauza ORDER BY**

Specifică operaţia de sortare a relaţiei rezultat. Clauza ORDER BY nu poate să apară în subinterogări şi în definiţiile de vederi.[9]

ORDER BY <expresie_ordonare>[ASC|DESC][,…n]

- **<expresie_ordonare>**- specifică o coloană pe baza căreia se face sortarea
- **ASC** - specifică ordonarea crescătoare
- **DESC** – specifică ordonarea descrescătoare

Funcțiile de agregare

Aceste funcţii sunt adesea folosite împreună cu clauza GROUP BY şi efectuează un anumit calcul asupra unui set de valori, returnând ca rezultat o singură valoare. Cu excepţia funcţiei COUNT, toate funcţiile de agregare ignoră valorile NULL. Aceste funcţii sunt prezentate în cadrul umătorului tabel.

Tabel 1. Funcţiile de agregare

Denumire	Descriere
AVG([DISTINCT] *expr*)	Returnează media valorilor unui grup. Opţiunea DISTINCT poate fi folosită pentru a returna media valorilor distincte din acest grup
COUNT([DISTINCT] *expr*)	Returnează numărul de elemente ale unui grup. În cazul în care avem COUNT(*), rezultatul este diferit deoarece returnează numărul de rânduri, incluzându-le şi pe cele care au valoarea NULL.
GROUP_CONCAT(*expr*)	Această funcţie returnează un şir de caractere concatenat din valori nenule dintr-un grup de şiruri de caractere
MAX([DISTINCT] *expr*)	Returnează valoarea maximă dintr-un grup. Această funcţie poate fi folosită şi pentru şiruri de caractere. Cuvântul cheie DISTINCT poate fi folosit, dar funcţia va avea aceeaşi valoare returnată indiferent dacă acest cuvânt e prezent sau nu.
MIN([DISTINCT] *expr*)	Această funcţie returnează valoarea minimă dintr-un grup. Această funcţie poate fi de asemenea folosită şi pentru şiruri de caractere
SUM([DISTINCT] *expr*)	Returnează suma unui grup de argumente

Aplicații

Pentru a putea vizualiza rezultatele obţinute cu ajutorul SELECT se va accesa meniul EDIT şi opţiunea PREFERENCES, după care din SQL Queries se va debifa opţiunea ”Safe Updates”. Se va folosi baza de date creată anterior şi anume gestiune_laboratoare.

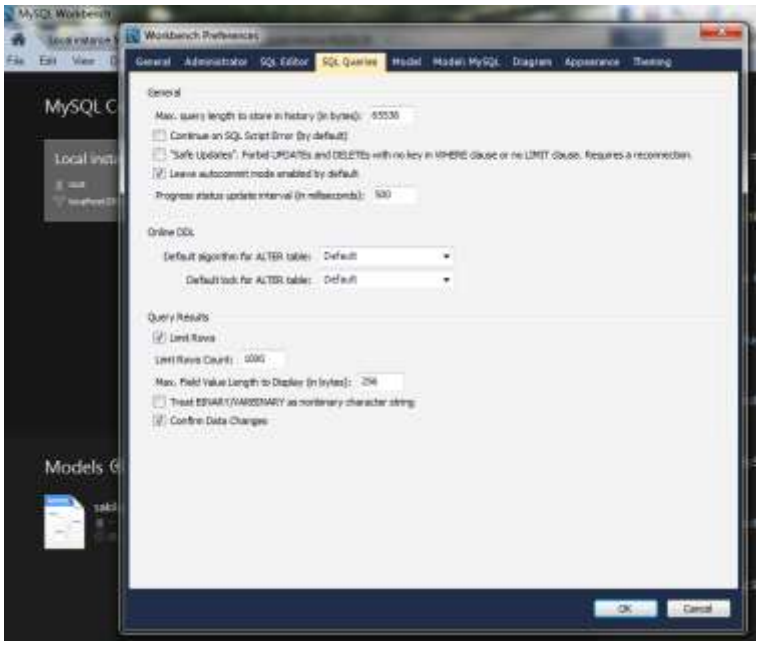

Fig.1.Setarea opțiunilor pentru vizualizarea rezultatelor.

Aplicația 1 Afișați toată aparatura existentă

Rezolvare

a)fără a aminti baza de date cu care se lucrează(se va afișa din baza de date curentă)

```
Select denumire
From aparatura;
```

b)amintind baza de date din care face parte tabelul

```
Select denumire
From gestiune_laboratoare.aparatura;
```

Aplicația 2 Afișați toate informațiile despre toată aparatura

Rezolvare

```
Select *
From aparatura;
```

Aplicația 3 Să se returneze denumirile din tabela laboratoare si lucrari_de_lab (pentru ca nu există o legatura intre acestea se va afisa de 10 ori(câte lucrari de laborator sunt) lista de laboratoare, astfel încat pentru fiecare laborator corespunde fiecare lucrare de lab)

Rezolvare

a) folosind explicit numele tabelelor

```
select laboratoare.denumire, lucrari_de_lab.denumire
from laboratoare, lucrari_de_lab;
```

b)folosind alias pentru tabele

```
select lab.denumire, luc.denumire
from laboratoare lab, lucrari_de_lab luc;
```

Aplicația 4 Inserați în tabela responsabili încă o persoana cu numele Georgescu Ioan și:

Rezolvare

a)afișați toate numele persoanelor care sunt responsabili de laboratoare

```
insert into responsabili
values(5,'Georgescu','Ioan','Str.Cosbuc nr.5',1,30);
SELECT ALL nume
FROM responsabili;
```

b)afișați doar numele distincte ale persoanelor care sunt responsabile de laboratoare

```
SELECT DISTINCT nume
FROM responsabili;
```

Aplicația 5 Echipamentele care nu sunt verificate

Rezolvare

```
Select denumire
From aparatura a
Where verificare=false;
```

Aplicația 6 Să se afișeze numele și prenumele persoanelor din tabela responsabili care au ID_functie=1

Rezolvare

```
select nume, prenume
```

```
from responsabili
where ID_functie=1;
```

Aplicația 7 Să se afișeze denumirea laboratoarelor care se află la una dintre adresele 'Str.Baritiu nr.26','Str Baritiu, 356'

Rezolvare

```
Select denumire
From laboratoare
Where adresa IN ('Str.Baritiu nr.26','Str Baritiu, 356');
```

Aplicația 8 Să se afișeze denumirea aparaturii care are ID_aparatură mai mare de 30

Rezolvare

```
select denumire
from aparatura
where ID_aparatura>30;
```

Aplicația 9 Numele laboratorului şi aparaturii care aparține acestui laborator

Rezolvare

```
select lab.denumire,ap.denumire
from laboratoare lab, aparatura ap,lab_ap
where lab_ap.ID_lab=lab.ID
and lab_ap.ID_ap=ap.ID_aparatura;
```

Aplicația 10 Laboratoarele de care se ocupa Georgescu Alin care are ID=1

Rezolvare

```
SELECT lab.denumire
FROM laboratoare lab, responsabili r
WHERE lab.ID_responsabil=r.ID AND lab.ID_responsabil=1;
```

Aplicația 11 Sa se afișeze valoarea medie a suprafetelor laboratoarelor existente în baza de date.

Rezolvare

```
select avg(suprafata)
from laboratoare;
```

Aplicația 12 Să se afișeze suprafaţa maximă dintre suprafeţele laboratoarelor.

Rezolvare

```
select max(suprafata)
from laboratoare;
```

Aplicația 13 Câte laboratoare există cu aceeaşi adresă şi care e media capacităţilor acestora?

Rezolvare

```
select adresa, count(adresa), avg(capacitate)
from laboratoare
group by adresa;
```

Aplicația 14 Afişaţi adresele la care se găsesc laboratoarele, alături de suma suprafeţelor tuturor laboratoarelor care se găsesc la aceeaşi adresă, folosind de asemenea opţiunea WITH ROLLUP.

Rezolvare

```
SELECT adresa, sum(suprafata)
from laboratoare
group by adresa with rollup;
```

Aplicația 15 Să se afişeze data expirării garanţiei şi numărul de aparate care au aceeaşi dată de expirare a acesteia pentru expirare_garantie>'2011-11-05'. Ordinea acestor înregistrărilor trebuie să fie făcută în funcţie de expirare garanţie, în ordine temporală.

Rezolvare

```
select Expirare_garantie, count(*)
from aparatura
group by Expirare_garantie
having Expirare_garantie>'2011-11-05'
order by Expirare_garantie;
```

Aplicația 16 Afişaţi numele aparatelor din lucrarea de laborator cu ID=2, care au data de expirare a garanţiei după '2011-11-05'

Rezolvare

```
select a.denumire,a.Expirare_garantie
from aparatura a, lab_ap x, laboratoare lab
where (a.ID_aparatura=x.ID_ap) and (x.ID_lab=lab.ID) and (lab.ID=2)
group by a.denumire
having a.Expirare_garantie>'2011-11-05';
```

Aplicaţia 17 Să se afişeze tabela Aparatura ordonată după 2 criterii, primul dintre ele fiind valoarea verificării, iar cel de al doilea denumirea.

Rezolvare

```
select *
from aparatura
order by verificare, denumire;
```

Aplicaţia 18 Să se afişeze înregistrările din tabela aparatura unde data de expirare a garanţiei este '2013-22-2' sau verificarea are valoarea 1.

Rezolvare

```
select *
from aparatura
where Expirare_garantie='2013-22-2' or verificare=1;
```

Exerciții propuse

1. Să se afișeze coloanele denumire și adresa din tabela laboratoare.
2. Să se afișeze toate câmpurile din tabela lucrari_de_lab.
3. Să se afișeze denumirea aparaturii care este verificată.
4. Să se afișeze numele laboratorului și numele și prenumele persoanelor care se ocupă de acesta.
5. Să se afişeze coloanele nume, prenume şi adresă din tabelul responsabili.
6. Arătaţi ce este greşit în cazul următoarelor aplicaţii:

```
Select *, denumire
From aparatură;

Select  nume
From funcții;

Select all distinct spec_tehn
From aparatura;

Select denumire, adresa
From laboratoare lab,responsabili r
Where lab.denumire='Georgescu';
```

7. Să se afişeze aparatura ce se găseşte în laboratorul cu ID=2 şi specificaţiile tehnice ale acestora.

8. Să se afişeze denumirea laboratorului unde se desfăşoară lucrarea de laborator cu ID-ul egal cu 5.
9. Să se afişeze aparatura folosită pentru lucrarea de laborator 3.
10. Să se afişeze responsabilii cu denumirea şi descrierea funcţiei acestora.
11. Să se afişeze toate echipamentele cu expirare_garantie între ‘2011-09-03’ si ‘2012-06-05’.
12. Să se afişeze toată aparatura a cărei dată de expirare a garanţiei a trecut deja
13. Să se afişeze toate laboratoarele care au în componenţă un număr mai mare de 4 aparate.
14. Să se afişeze numărul de tipuri de aparate care există folosind count. De la această aplicaţie se vor folosi clauzele GROUP BY, HAVING şi ORDER BY.
15. Să se afişeze denumirea laboratoarelor cu adresa pe 'Str. Dorobantilor 71-73' şi 'Str. Baritiu nr.26', grupate după adresă.
16. Să de afişeze denumirea aparatelor şi numărul total de aparate de acel tip conţinute în laboratoarele universităţii.
17. Afişaţi aparatura care are verificarea la zi şi expirarea garanţiei după data de '2011-01-01'. Aparatura se va grupa după denumire şi prima condiţie va fi conţinută într-o clauză WHERE, iar cea de a doua în clauza HAVING.
18. Afişaţi numărul de lucrări de laborator care se desfăşoară în laboratorul cu ID-ul egal cu 5.
19. Să se afişeze numele şi prenumele responsabililor alături de numărul de laboratoare pentru care răspund.
20. Să se afişeze numele şi prenumele responsabililor alături de numărul de laboratoare pentru care răspund, dar de data aceasta să se ordoneze după nume responsabili.
21. Numărul total de aparate care se folosesc pentru lucrarea de laborator cu ID-ul 2 şi denumirea laboratorului cu acel ID.

Operatori de control

- **Case**

Sintaxa acestui operator este:

```
CASE valoare WHEN [val_de_comparat1] THEN rezultat1
[WHEN [val_de_comparat2] THEN rezultat2 ...]
 [ELSE rezultat3] END
```

În cazul în care valoare = val_de_comparat1 atunci operatorul CASE va returna rezultat1. În caz contrar, se va trece la compararea valorii din CASE cu val_de_comparat2 şi dacă acestea sunt egale va fi returnat rezultat2. În toate celelalte cazuri valoarea retrnată va fi rezultat3. Dacă clauza ELSE nu există şi nici una dintre condiţii nu se adevereşte, se va returna NULL.[10]

Exemplul 1:

```
SELECT CASE x WHEN 1 THEN 'unu'
```

```
WHEN 2 THEN 'doi' ELSE 'mai mult' END;
```

Instrucţiunea de mai sus va returna valoarea 1 dacă x=1 valoarea 2 dacă x=2 sau “mai mult” dacă x are oricare altă valoare numerică.

- **IF**

Sintaxa acestui operator se poate observa în cele ce urmează:

```
IF(expr1,expr2,expr3)
```

Dacă expr1 este adevărată cu condiţia ca aceasta să fie diferită de 0 sau NULL, atunci instrucţiunea va returna valoarea expr2; în caz contrar, se va returna expr3. [10]

Exemplul 2:

SELECT IF(1>2,2,3); va returna valoarea 3 deoarece condiţia 1>2 nu este adevărată

- **IFNULL**

```
IFNULL(expr1,expr2)
```

Dacă expr1 nu este NULL, atunci instrucţiunea va returna expr1, altfel se va returna expr2.[10]

Exemplul 3:

SELECT IFNULL(1,0); va returna valoarea 1 deoarece prima valoare este diferită de null.

- **NULLIF**

```
NULLIF(expr1,expr2)
```

Va returna valoare NULL dacă expr1 = expr2, iar în caz contrar returnează expr1. Aceasta va returna acelaşi rezultat ca şi CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END.[10]

Exemplul 4:

SELECT NULLIF(1,1); va returna NULL deoarece 1=1

Operatorul ISNULL

O valoare NULL indică faptul că acea valoare este necunoscută. O valoare null este diferită de un şir vid sau de o valoare egală cu 0. Două valori null nu sunt niciodată egale între ele.

La operarea cu valori NULL, trebuie să se ţină seama de unele reguli:

- pentru a testa în clauza where a unei interogări dacă o valoare este sau nu NULL, se folosesc operatorii IS NULL şi IS NOT NULL
- toate funcţiile de agregare ignoră valorile NULL existente în setul de valori asupra căreia se aplică
- dacă există mai multe valori NULL printre valorile unui atribut specificat într-o clauză GROUP BY, acestea vor forma un singur grup

Valorile NULL din cadrul tabelelor vor fi specificate în momentul definirii acestora, fiind interzisă folosirea unei astfel de valori pentru cheile primare şi cele străine.[10]

În cadrul instrucţiunii select există o clauză specială WHERE care se poate utiliza pentru detecţia coloanelor care conţin valori NULL, şi anume ISNULL care are următoarea sintaxă:

ISNULL(expresie, valoare_de_înlocuire)

Exemplul 5:

Dacă am avea un tabel care să conțină informații despre angajații unei firme, valorile returnate pentru instrucţiunea

```
Select  nume, prenume
From angajati
Where salariu is null;
```

Vor fi doar numele și prenumele angajaților cărora nu le este specificat salariul.

Operatorul NOT

Operatorul NOT al clauzei WHERE are funcţia de a nega orice condiţie pe care o precede. În clauzele simple folosirea operatorului NOT nu este foarte eficientă, dar în clauze complexe, mai ales in combinaţie cu IN, facilitează găsirea tuturor rândurilor care nu corespund unei liste de criterii.

Operatorul LIKE

Operatorul LIKE vine în ajutorul utilizatorilor care doresc să afişeze date de tip şir de caractere precizând doar o mică parte din acestea. De exemplu dacă dorim să afişăm dintr-o bază de date toate persoanele cu prenumele Camelia, indiferent dacă acestea mai au şi un alt prenume. Acest lucru se poate face prin intermediul caracterelor de înlocuire, prin crearea unor modele de căutare care pot fi comparate cu datele dintr-o bază de date.

Pentru a folosi caracterele de înlocuire în clauzele de căutare conţinute în clauza WHERE, trebuie utilizat operatorul LIKE. Acesta indică programului să compare modelul de căutare următor folosind caracterele de înlocuire în locul operatorului de egalitate.

- **Caractere de înlocuire % şi _**

  - **Operatorul %** este cel mai folosit caracter de înlocuire. În interiorul unui şir de căutare, % are rolul de a găsi orice caracter indiferent de câte ori apare. Acesta poate fi folosit pentru a găsi toate înregistrările care încep cu o literă, un cuvânt sau un grup de cuvinte, care au în componenţă un şir de caractere sau încep cu o literă şi se termină cu altă literă precizate de către utilizator.[4]
  - **Operatorul _** se foloseşte în acelaşi mod ca şi %, dar acesta substituie doar un singur character

Cuplarea externă SQL JOIN

Cuvântul cheie Join este folosit în SQL pentru a interoga două sau mai multe tabele, între care există relaţii la nivel de coloane. Tabelele din baza de date sunt de cele mai multe ori cuplate între ele prin chei. În MySQL, există 4 tipuri de JOIN:

- **INNER JOIN-** returnează înregistrările, atunci când există cel puţin o potrivire în ambele tabele, adică o înregistrare este adăugată în rezultat numai în cazul în care există o înregistrare care îndeplineşte condiţia de join atât în tabela din stânga, cât şi în cea din dreapta.
- **LEFT JOIN-**este folosită în cazul în care se dorește să fie returnate toate înregistrările din tabela din stânga (nume_tabelă1) chiar dacă nu există tuple în tabela din dreapta (nume_tabelă2) care să îndeplinească condiţia
- **RIGHT JOIN-**este folosită în cazul în care se doresc să fie returnate toate înregistrările din tabela din dreapta (nume_tabelă2) chiar dacă nu există tuple în tabela din stânga (nume_tabelă1) care să îndeplinească condiţia de JOIN.

- **FULL JOIN**-se foloseşte atunci când se doreşte ca rezultatul operaţiei să fie produsul cartezian între toate înregistrările din tabela din dreapta şi toate înregistrările din tabela din stânga ce îndeplinesc o anumită condiţie.

Sintaxa acestor interogări este aproximativ identică şi este prezentată în cele ce urmează:

```
SELECT <nume_coloană>, <nume_coloană> ,…
FROM <nume_tabelă1>
INNER JOIN| LEFT JOIN| RIGHT JOIN| FULLJOIN <nume_tabelă2>
ON <condiţie_de_potrivire>
```

Semnificaţia opţiunilor pentru clauzele JOIN este următoarea:

- **<nume_coloană>** - numele coloanei ce se doreşte a fi returnată în rezultat.
- **<nume_tabelă1>, <nume_tabelă2>** - numele tabelelor pentru care dorim să returnăm rezultatele.
- **<condiţie_de_potrivire>** - condiţia ce trebuie să fie satisfăcută de cel puţin tabela din dreapta pentru ca înregistrarea să fie adăugată în rezultat

UNION

Operatorul UNION se foloseşte atunci când se doreşte combinarea rezultatelor a două sau mai multe rezultate SELECT.

```
SELECT <nume_coloane> FROM <nume_tabelă1>
UNION [ALL]
SELECT <nume_coloane> FROM <nume_tabelă2>
```

Diferenţa dintre UNION şi UNION ALL este faptul că UNION ALL permite valori duplicate, pe când UNION nu permite acest lucru, ele trebuie să fie distincte.

Semnificaţia opţiunilor pentru clauza UNION şi UNION ALL este următoarea:

- **<nume_coloane>** - numele coloanelor ce se doresc a fi returnate în rezultat.
- **<nume_tabelă1>, <nume_tabelă2>** - numele tabelelor pentru care dorim să returnăm rezultatele.

Trebuie să se ţina cont de constrângerile următoare atunci când se definesc numele coloanelor pentru frazele SELECT:

- Numărul de coloane trebuie să fie la fel în toate frazele SELECT.
- Ordinea coloanelor în frazele SELECT trebuie să fie la fel în toate frazele.
- Tipul coloanelor din frazele SELECT trebuie să fie la fel în toate frazele.

Aplicaţii

Aplicația 1: Afişaţi angajaţii din tabelul responsabili care nu au numele Gog Grigore

Rezolvare

```
select nume, prenume
from responsabili
where not (nume='Gog' and prenume='Grigore');
```

Aplicația 2:Să se afişeze tate numele şi prenumele responsabililor a căror nume încep cu litera G

Rezolvare

```
select nume, prenume
from responsabili
where nume like 'G%';
```

Aplicația 3:Să se afişeze denumirea aparaturii care are în denumire şirul de caractere ‘curent continuu’

Rezolvare

```
select denumire
from aparatura
where denumire like '%curent continuu%';
```

Aplicația 4:Să se afişeze toate datele despre laboratoarele care au adresa pe strada Baritiu şi în adresă dupa stradă mai există 5 spaţii

Rezolvare

```
select *
from laboratoare
where adresa like 'Str. Baritiu _____';
```

În continuare se va demonstra modul de utilizare a funcţiilor JOIN şi UNION.

Aplicația 5:Să se creeze o nouă bază de date cu denumirea Personal conţinând două tabele cu datele de mai jos .Primul dintre acestea va avea denumirea angajati, iar cel de al doilea articole

ID	Nume	Prenume
1	Baciu	Ion
2	Dan	Daniel
3	Gal	Ionut
4	Nemes	Alex

ID	Nr_artic	ID_personal
1	23	1
2	15	1
3	11	4
4	45	2
5	43	2
6	12	6

Aplicația 6: Se dorește afișarea persoanelor care răspund de cel puțin un articol (folosind INNER JOIN)

Rezolvare

```
SELECT a.nume, a.prenume, ar.nr_artic
FROM angajati a
INNER JOIN articole ar
ON ar.ID_personal = a.ID;
```

Aplicația 7: Se dorește să se afișeze toate persoanele și articolele de care răspund, chiar dacă există persoane cărora nu le corespunde nici un articol (LEFT JOIN).

Rezolvare

```
SELECT a.nume, a.prenume, ar.nr_artic
FROM angajati a
LEFT JOIN articole ar
ON ar.ID_personal = a.ID;
```

Aplicația 8: Se dorește să se afișeze toate articolele și persoanele care răspund de acestea, chiar dacă sunt articole de care nu răspunde nimeni (RIGHT JOIN)

Rezolvare

```
SELECT a.nume, a.prenume, ar.nr_artic
FROM angajati a
RIGHT JOIN articole ar
ON ar.ID_personal = a.ID;
```

Aplicația 9: Creați cele două tabele de mai jos pentru a prezenta folosirea operatorului UNION după care uniți cele două tabele folosind UNION și UNION ALL. Cele două tabele se vor numi persoane1 și persoane2

NUME	PRENUME
Dan	Alina
Ilies	Corina
Dolha	Bianca
Marian	Cristian

NUME	PRENUME
Moldovan	Andreea
Ilies	Corina
Muresan	Maria

Rezolvare

```
SELECT p1.nume, p1.prenume
FROM persoane1 p1
UNION
SELECT p2.nume, p2.prenume
FROM persoane2 p2;

SELECT p1.nume, p1.prenume
FROM persoane1 p1
UNION ALL
SELECT p2.nume, p2.prenume
FROM persoane2 p2;
```

Exerciții propuse

1. Să se ruleze următoarele instrucţiuni şi să se comenteze rezultatul obţinut.

```
SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END;

SELECT CASE BINARY 'B'
    WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;

    SELECT IF(1<2,'yes','no');

    SELECT IF(STRCMP('test','test1'),'no','yes');

    SELECT IFNULL(NULL,10);

    SELECT IFNULL(1/0,10);

   SELECT IFNULL(1/0,'yes');

   SELECT NULLIF(1,2);
```

2. Să se afişeze din tabela aparatură denumire şi id-ul aparaturii căreia nu îi sunt precizate specificaţiile tehnice
3. Să se afişeze din tabela aparatură denumirea şi câmpul verificare aparaturii căreia ii sunt precizate specificaţiile tehnice
4. Să se afişeze denumirea aparaturii a cărei denumire se termină cu şirul de caractere ‚curent alterativ’
5. Să se afişeze toate aparatele (denumirea) din tabelul aparatură a căror denumire începe cu A ordonate după denumire

6. Se doreşte să se afişeze toţi responsabilii (nume, prenume) şi laboratoarele de care răspund (LEFT JOIN).
7. Se doreşte să se afişeze toţi responsabilii (nume, prenume) şi laboratoarele de care răspund (LEFT JOIN), ordonate după prenumele responsabililor