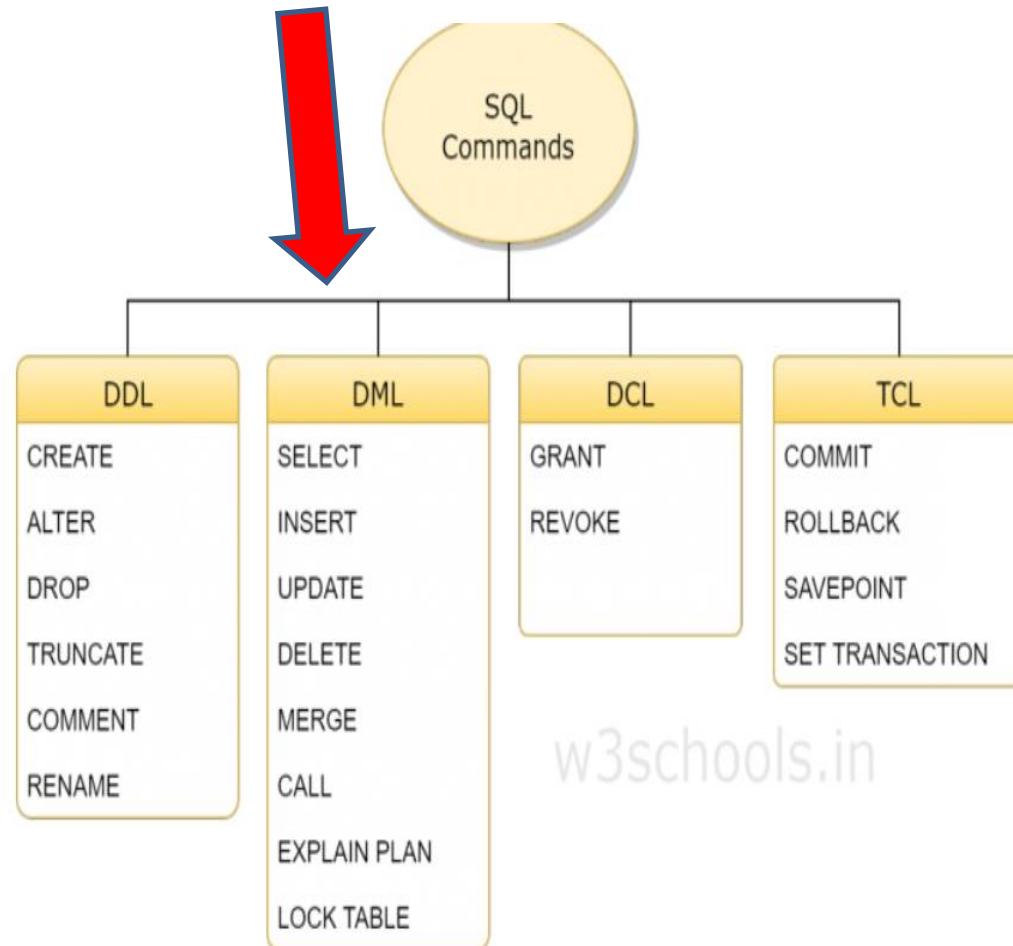


Limbaajul SQL

- **Funcțiile unui SGBD**
 - **DDL** – definirea datelor (în dicționarul de date)
 - **DML** – manipularea datelor
 - a. operații **CRUD (Create, Read, Update, Delete)**
 - **CDL** – regăsirea datelor: beneficiari, utilizatori, administratorii bazei de date
 - **TCL** – limbajul de control al tranzacțiilor la BD
 - administrarea bazei de date



Limbajul SQL

1. Limbajul de manipulare al datelor (LMD)

1. Adăugare **INSERT** o nouă înregistrare
2. Actualizarea **UPDATE** datelor dintr-o tabelă
3. Ștergerea **DELETE** tuplurilor dintr-o tabelă
4. Asocieri ale tabelelor unei BD, **JOIN**
5. Instrucțiunea **MERGE**

2. Limbajul de control al datelor (LCD). Tranzacții

Limbajul de manipulare al datelor (LMD)

Limbajul de **M**anipulare al **D**atelor este nucleul limbajului **SQL**.

Când doriți să adăugați, să actualizați, sau să ștergeți date din baza de date, executați comenzi **DML(Data Manipulation Language)**.

*O colecție de comenzi **DML** care formează o unitate logică reprezintă o tranzacție.*

Limbajul de manipulare al datelor (LMD)

Limbajul de **M**anipulare al **D**atelor (**DML**) care ne permite:

1. să adăugăm
2. să modificăm
3. sau să distrugem datele din baza de date și
conține următoarele comenzi:

- **INSERT**
- **UPDATE**
- **DELETE**
- **JOIN**
- **MERGE**

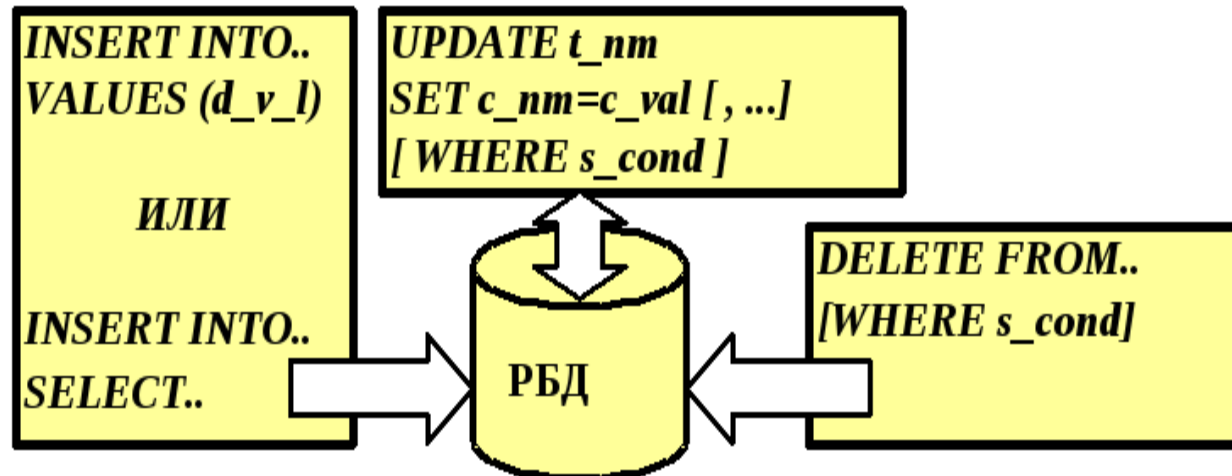
Limbaajul SQL

1. Limbaajul de manipulare al datelor (**LMD**)

1. Adăugare **INSERT** o nouă înregistrare
2. Actualizarea **UPDATE** datelor dintr-o tabelă
3. Ștergerea **DELETE** tuplurilor dintr-o tabelă
4. Asocieri ale tabelelor unei BD, **JOIN**
5. Instrucțiunea **MERGE**

2. Limbaajul de control al datelor (**LCD**).

Tranzacții



Adăugare o nouă înregistrare

Sintaxa este:

```
INSERT INTO tabela [ (coloana [, coloana . . . ])]  
VALUES ( valoare [, valoare . . . ]);
```



tabela – numele tabelului

coloana – numele coloanei din tabela

valoare – valoarea corespunzătoare coloanei

Notă: Se poate adăuga o singură linie o dată.

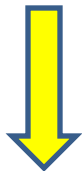
Adăugare o nouă înregistrare

Exemplu

Să se introducă un nou oras în tabela Departamente.

INSERT INTO departamente (nr_dept, denumire, localitate)
VALUES (50,'FINANCIAR','Bucuresti');

Autor



id_autor	nume_autor
1	Petru
2	Diuma
3	Vieru
4	Lena
5	Ioana
6	Ghita
7	Colea
8	Nicu
9	Valea

Carti

idcarte	autor	titlu	pret	cantitatea	id_autor
2-2222-222-10	Petru	GFGF Laborator Mysql-Php vbvbvb	950.00	23	1
2-2222-222-13	Vieru	MAMA	2000.00	200	3
2-2222-2222-22	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-23	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-6	Vica	Cei trei muschetari Php	930.00	40	4
2-2222-2222-8	Eminescu	Ghid Php	1000.00	3	6
2-2222-2222-9	Eminescu	Ghid Php	1000.00	3	2

Adăugare o nouă înregistrare

- Deoarece se poate insera o nouă linie ce conține valori pentru fiecare coloană, lista coloanelor nu mai este necesară în clauza **INSERT**.
- Totuși dacă nu utilizăm lista de coloane, valorile trebuie să fie listate în ordinea coloanelor din tabelă, iar o valoare trebuie utilizată pentru fiecare coloană.
- Pentru o utilizare mai ușoară putem folosi comanda **DESCRIBE departamente**, care ne afișează câmpurile tabelii în ordinea lor, precum și tipul fiecărui câmp.

Adăugare o nouă înregistrare

Inserarea linilor ce conțin valori NULL

Inserarea liniilor ce conțin valori **NULL** se poate face prin două metode:

1. metoda implicită: *Omiterea unor câmpuri din lista câmpurilor existente în tabela respectivă.*

```
INSERT INTO carti (autor, titlu)  
VALUES ('Vica', 'Luna mai');
```

Adăugare o nouă înregistrare

2. metoda explicită: Specificarea cuvântului NULL în clauza **VALUES**.

INSERT INTO departamente (nr_dept, *denumire*, localitate)
VALUES (50, *NULL*, 'Bucuresti');



Campul DENUMIRE este
in lista, dar valoarea
este NULL

idcarte	autor	titlu	pret	cantitatea	id_autor
2-2222-222-10	Petru GFGF	Laborator Mysql-Php vbvbyb	950.00	23	1
2-2222-222-13	Vieru	MAMA	2000.00	200	3
2-2222-2222-22	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-23	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-6	Vica	Cei trei muschetari Php	930.00	40	4
2-2222-2222-8	Eminescu	Ghid Php	1000.00	3	6
2-2222-2222-9	Eminescu	Ghid Php	1000.00	3	2

Adăugare o nouă înregistrare

Inserarea unor valori speciale

Funcția **SYSDATE()** înregistrează *data curentă și ora*.

Putem utiliza diferite funcții pentru a insera valori speciale în tabela noastră.

Adăugare o nouă înregistrare

Exemplu

Inserează în tabela Angajati datele personale, precum și data când acestea au fost introduse, prin utilizarea comenzii **`SYSDATE()`**, care reprezintă data sistemului.

```
ALTER TABLE carti ADD COLUMN data_n  
DATE AFTER autor;
```

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.3331 seconds.)

```
ALTER TABLE carti ADD COLUMN data_n DATE AFTER autor
```

idcarte	autor	titlu	pret	cantitatea	id_autor
2-2222-222-10	Petru GFGF	Laborator Mysql-Php vbvbnb	950.00	23	1
2-2222-222-13	Vieru	MAMA	2000.00	200	3
2-2222-2222-22	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-23	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-6	Vica	Cei trei muschetari Php	930.00	40	4
2-2222-2222-8	Eminescu	Ghid Php	1000.00	3	6
2-2222-2222-9	Eminescu	Ghid Php	1000.00	3	2



idcarte	autor	data_n	titlu	pret	cantitatea	id_autor
2-2222-222-10	Petru GFGF	NULL	Laborator Mysql-Php vbvbnb	950.00	23	1
2-2222-222-13	GHITA	NULL	MAMA	2000.00	200	3
2-2222-2222-22	Eminescu	NULL	Ghid Php	1000.00	3	2
2-2222-2222-23	Eminescu	NULL	Ghid Php	1000.00	3	2
2-2222-2222-6	Vica	NULL	Cei trei muschetari Php	930.00	40	4
2-2222-2222-8	Eminescu	NULL	Ghid Php	1000.00	3	2
2-2222-2222-9	Eminescu	NULL	Ghid Php	1000.00	3	2

Adăugare o nouă înregistrare

```
INSERT INTO angajati (nr_angajat, nume, functie,  
manager, data_ang, salariu, comision, nr_dept)  
VALUES (7658, 'IONESCU', 'ANALIST', 7566, SYSDATE,  
1000, NULL, 20); SYSDATE == SYSDATE()
```

```
INSERT INTO carti ('data_n')  
VALUES (SYSDATE());
```

✓ 1 row inserted. (Query took 0.1082 seconds.)

```
INSERT INTO carti (data_n) VALUES (SYSDATE())
```

✓ Showing rows 0 - 7 (8 total, Query took 0.0022 seconds.)

```
SELECT * FROM `carti`
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP]

☐ Show all

Number of rows: 25

Filter rows

Search this table

Sort by key:

None

+ Options

	idcarte	autor	data_n	titlu	pret	cantitatea	id_autor
☐ Edit ☐ Copy ☐ Delete		NULL	2021-11-16	NULL	NULL	NULL	NULL
☐ Edit ☐ Copy ☐ Delete	2-2222-222-10	Petru GFGR	NULL	Laborator Mysql-Php vbvbvb	950.00	23	1
☐ Edit ☐ Copy ☐ Delete	2-2222-222-13	GHITA	NULL	MAMA	2000.00	200	3
☐ Edit ☐ Copy ☐ Delete	2-2222-2222-22	Eminescu	NULL	Ghid Php	1000.00	3	2
☐ Edit ☐ Copy ☐ Delete	2-2222-2222-23	Eminescu	NULL	Ghid Php	1000.00	3	2
☐ Edit ☐ Copy ☐ Delete	2-2222-2222-6	Vica	NULL	Cei trei muschetari Php	930.00	40	4
☐ Edit ☐ Copy ☐ Delete	2-2222-2222-8	Eminescu	NULL	Ghid Php	1000.00	3	2
☐ Edit ☐ Copy ☐ Delete	2-2222-2222-9	Eminescu	NULL	Ghid Php	1000.00	3	2

Adăugare date din fisier

LOAD DATA INFILE 'C:\data\autor.txt'
INTO TABLE autor
FIELDS TERMINATED BY ',';

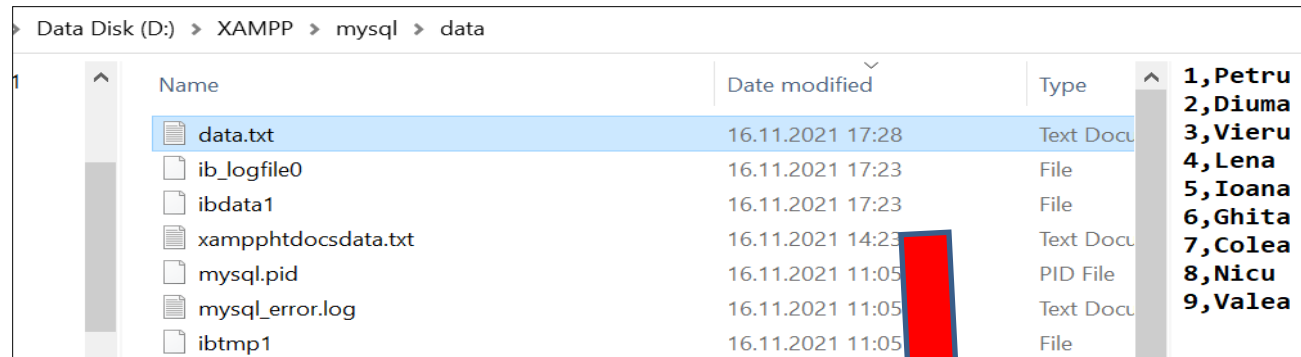
autor.txt - Notepad
File Edit Format View Help
12,PETRU
13,ION
14,COLEA

id_autor	nume_autor
1	Petru
2	Diuma
3	Vieru
4	Lena
5	Ioana
6	Ghita
7	Colea
8	Nicu
9	Valea

	id_autor	nume_autor
☐ Edit ☐ Copy ☐ Delete	1	Petru
☐ Edit ☐ Copy ☐ Delete	2	Diuma
☐ Edit ☐ Copy ☐ Delete	3	Vieru
☐ Edit ☐ Copy ☐ Delete	4	Lena
☐ Edit ☐ Copy ☐ Delete	5	Ioana
☐ Edit ☐ Copy ☐ Delete	6	Ghita
☐ Edit ☐ Copy ☐ Delete	7	Colea
☐ Edit ☐ Copy ☐ Delete	8	Nicu
☐ Edit ☐ Copy ☐ Delete	9	Valea
☐ Edit ☐ Copy ☐ Delete	12	PETRU
☐ Edit ☐ Copy ☐ Delete	13	ION
☐ Edit ☐ Copy ☐ Delete	14	COLEA

Export de date in fisier

```
SELECT * INTO OUTFILE 'D:\XAMPP\mysql\data\autor.csv'  
FIELDS TERMINATED BY ','  
FROM autor
```



Name	Date modified	Type
data.txt	16.11.2021 17:28	Text Document
ib_logfile0	16.11.2021 17:23	File
ibdata1	16.11.2021 17:23	File
xampphtdocsdata.txt	16.11.2021 14:23	Text Document
mysql.pid	16.11.2021 11:05	PID File
mysql_error.log	16.11.2021 11:05	Text Document
ibtmp1	16.11.2021 11:05	File

PhpMyadmin **EXPORT** Csv → autor.csv SE DUCE IN DOWNLOAD

IL Copiem autor.csv in 'D:\XAMPP\mysql\data\autor.csv'

DELETE FROM autor

ALTERAM TABELUL CARTI

ALTER TABLE carti

ADD CONSTRAINT FK_carti_id_autor

FOREIGN KEY (id_autor)

REFERENCES autor

(id_autor)

ON DELETE CASCADE ON UPDATE CASCADE;

Error

SQL query

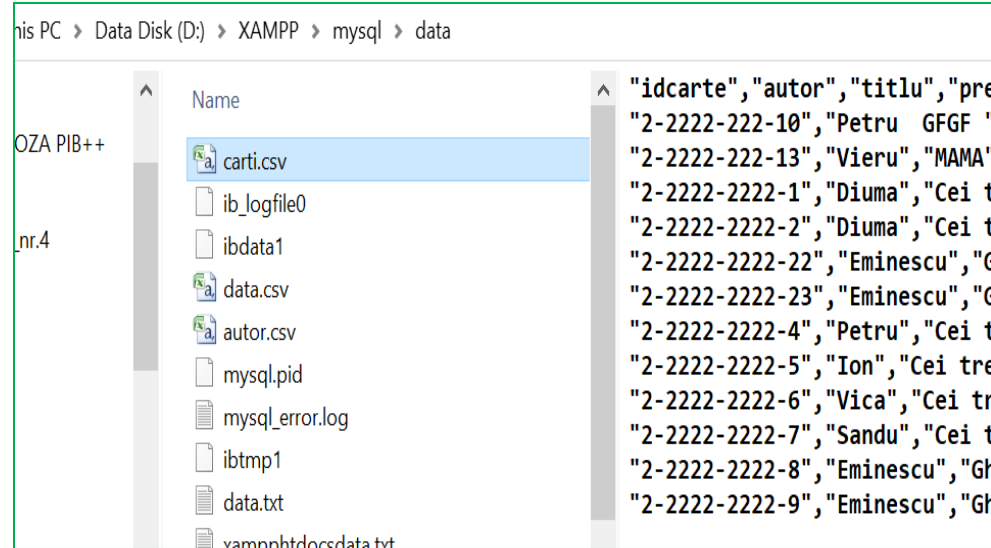
DELETE

MySQL sai

#1451 - C
FOREIGN K

Export de date in fisier

SELECT * INTO **OUTFILE 'D:\XAMPP\mysql\data\carti.csv'**
FIELDS TERMINATED BY ','
FROM carti



PhpMyadmin **EXPORT** Csv → **carti.csv** **SE DUCE IN DOWNLOAD**
IL Copiem **carti.csv** in 'D:\XAMPP\mysql\data\carti.csv'

DELETE de date in fisier

până la comanda DELETE →

DELETE from autor



Show query box

✓ 9 rows affected. (Query took 0.2498 seconds.)

```
DELETE FROM `autor`
```

DUPA comanda DELETE →

MySQL returned an empty result set

```
T * FROM `autor`
```

tor nume_autor

y results operations

11/22/2021

idcarte	autor	data_n	titlu	pret
	NULL	2021-11-16	NULL	NULL
2-2222-222-10	Petru GFGF	NULL	Laborator Mysql-Php vbvbvb	950.00
2-2222-222-13	GHITA	NULL	MAMA	2000.00
2-2222-2222-22	Eminescu	NULL	Ghid Php	1000.00
2-2222-2222-23	Eminescu	NULL	Ghid Php	1000.00
2-2222-2222-25	Fusu	NULL	NULL	NULL
2-2222-2222-6	Vica	NULL	Cei trei muschetari Php	930.00
2-2222-2222-8	Eminescu	NULL	Ghid Php	1000.00
2-2222-2222-9	Eminescu	NULL	Ghid Php	1000.00

Query took 0.0013 seconds.)

☐ Profiling [Edit inline] [Edit] [

ws: 25 ▼

Filter rows: Search this table

Sort by key: Non

idcarte	autor	data_n	titlu	pret	cantitatea	id_autor
	NULL	2021-11-16	NULL	NULL	NULL	NULL
2-2222-2222-25	Fusu	NULL	NULL	NULL	NULL	NULL

IMPORT de date in fisier

LOAD DATA INFILE 'D:/XAMPP/mysql/data/autor.csv'

INTO TABLE *autor*

FIELDS TERMINATED BY ',';



autor	nume_autor
1	Petru
2	Diuma
3	Vieru
4	Lena
5	Ioana
6	Ghita
7	Colea
8	Nicu
9	Valea



idcarte	autor	data_n	titlu	pret	cantitatea	id_autor
	NULL	2021-11-16	NULL	NULL	NULL	NULL
2-2222-2222-25	Fusu	NULL	NULL	NULL	NULL	NULL

Show query box

✓ 9 rows inserted. (Query took 0.1405 seconds.)

```
LOAD DATA INFILE 'D:/XAMPP/mysql/data/data.csv' INTO TABLE autor FIELDS TER
```

LOAD DATA INFILE 'D:/XAMPP/mysql/data/carti.csv'

INTO TABLE *carti*

FIELDS TERMINATED BY ',';

Adăugare date din fisier

INSERT INTO carti SET idcarte ='2-2222-2222-25',
autor='Fusu';

idcarte	autor	data_n	titlu	pret	cantitatea	id_autor
	NULL	2021-11-16	NULL	NULL	NULL	NULL
2-2222-222-10	Petru GFGF	NULL	Laborator Mysql-Php vbvbvb	950.00	23	1
2-2222-222-13	GHITA	NULL	MAMA	2000.00	200	3
2-2222-222-22	Eminescu	NULL	Ghid Php	1000.00	3	2
2-2222-222-23	Eminescu	NULL	Ghid Php	1000.00	3	2
2-2222-2222-25	Fusu	NULL	NULL	NULL	NULL	NULL
2-2222-2222-6	Vica	NULL	Cei trei muschetari Php	930.00	40	4
2-2222-2222-8	Eminescu	NULL	Ghid Php	1000.00	3	2
2-2222-2222-9	Eminescu	NULL	Ghid Php	1000.00	3	2

Adăugare date din fisier

Dacă dintr-un motiv oarecare trebuie să omiteți erorile de introducere a datelor, atunci puteți utiliza cuvântul cheie **IGNORE**. De exemplu, următoarea comandă va genera o eroare, deoarece o intrare cu acel ID există deja.

```
INSERT INTO carti SET idcarte ='2-2222-2222-25',  
autor='Fusu';
```

```
INSERT IGNORE INTO carti SET idcarte ='2-2222-2222-  
25', autor='Fusu';
```

Adăugare date din fisier

Interesant dar care va fi rezultatul comenzii

INSERT INTO carti VALUES();

Interesant ar putea să fie și așa

INSERT INTO autor VALUES(DEFAULT, DEFAULT);

Am putea efectua și operații în timpul executării
comenzii INSERT

```
mysql> INSERT INTO employee VALUES(50*2, 'Thomas', 'Sales', 5000+id);
```

```
mysql> select * from employee;
```

id	name	dept	salary
100	Thomas	Sales	5100

Adăugare date din fisier

```
INSERT LOW_PRIORITY INTO employee VALUES(100,  
'Thomas', 'Sales', 5000);
```

```
INSERT HIGH_PRIORITY INTO employee VALUES(100,  
'Thomas', 'Sales', 5000);
```

Dacă tabelul în care dorim să inserăm date este supus la o mulțime de operațiuni de citire, atunci prin setarea unei **priorități scăzute**, operația de scriere poate să nu funcționeze pentru o perioadă destul de lungă de timp.

Adăugare date din fisier

Actualizări de rând la repetare

Dacă în timpul procesului de înregistrare este găsită o înregistrare cu o valoare ID duplicat, operațiunea va fi oprită cu o eroare corespunzătoare.

```
INSERT INTO employee VALUES(100,'Thomas','Sales',5000);  
ERROR 1062 (23000): Duplicate entry '100' for key 'PRIMARY'
```

Dar, putem actualiza câmpurile unei astfel de înregistrări (dacă se găsește un duplicat) folosind comanda

ON DUPLICATE KEY UPDATE.

În exemplul de mai jos, actualizăm valoarea câmpului **salary** atunci când apare o înregistrare duplicată, mărand-o cu 500.

```
mysql> INSERT INTO employee VALUES(100,'Thomas','Sales',5000) on DUPLICATE KEY UPDATE salary=salary+500;
```

```
mysql> select * from employee;
```

id	name	dept	salary
100	Thomas	Sales	5500

Adăugare o nouă înregistrare

Copierea informațiilor dintr-o altă tabelă cu ajutorul unei *SUBINTEROGARI*

- Se scrie comanda **INSERT** cu ajutorul unui *subquery*.
- Nu se utilizează clauza **VALUES**.
- **Potriviți numărul de câmpuri din clauza INSERT** cu cel din *subquery*.
- Se poate folosi clauza **INSERT** pentru a adăuga linii într-o tabelă unde valorile sunt dintr-o altă tabelă.
- În loc de clauza **VALUES**, folosim un *subquery*.

Adăugare o nouă înregistrare

Sintaxa

```
INSERT INTO tabela [ coloana (, coloana) ]  
subquery(subcerere);
```

tabela – numele tablei

coloana – numele câmpului din tabelă

subquery (subinterogare) – subquery-ul care returnează câmpurile din cealaltă tabelă

Adăugare o nouă înregistrare

- *Numărul de coloane și tipurile de date din lista câmpurilor din clauza **INSERT** trebuie să se potrivească cu valorile și tipurile de date din **subquery**.*
- *Pentru a crea o copie a linilor unei tabele, vom folosi **SELECT *** în **subquery**.*

Adăugare o nouă înregistrare

Exemplu

Se introduc datele din tabela Angajati într-o altă tabelă numită ***Copie_angajati***.

```
INSERT INTO copie_angajati  
SELECT * FROM angajati;
```

Subinterogarea
selectează toate
câmpurile tabeli de
unde se preiau date

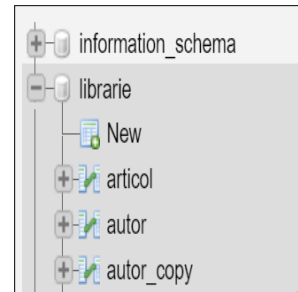
idcarte	autor	titlu
2-2222-222-10	Petru GFGF	Laborator Mysql-Php vbvbnb
2-2222-222-13	Vieru	MAMA
2-2222-2222-22	Eminescu	Ghid Php
2-2222-2222-23	Eminescu	Ghid Php
2-2222-2222-6	Vica	Cei trei muschetari Php
2-2222-2222-8	Eminescu	Ghid Php
2-2222-2222-9	Eminescu	Ghid Php

Adăugare o nouă înregistrare

Exemplu

Precautam tabelul **CUSTOMERS_BKP** cu o structură similară sub forma unui tabel pentru clienți. Acum, pentru a copia întregul tabel **CUSTOMERS** în tabelul **CUSTOMERS_BKP**, puteți utiliza următoarea sintaxă.

```
INSERT INTO CUSTOMERS_BKP SELECT * FROM  
CUSTOMERS WHERE ID IN (SELECT ID FROM  
CUSTOMERS) ;
```



DUBLAM UN TABEL SI IL OBȚINEM CURAT, FĂRĂ DATE

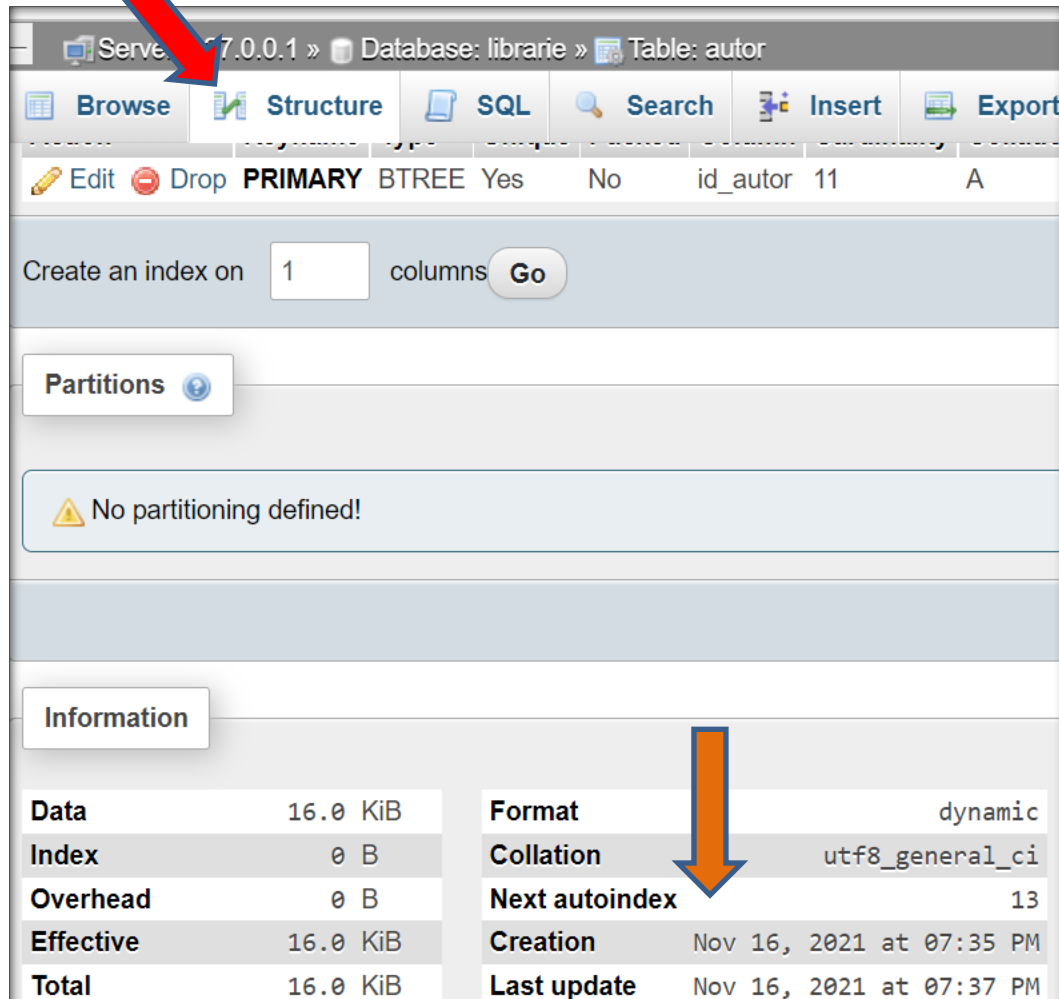
```
CREATE TABLE autor_copy LIKE autor;
```

```
INSERT INTO autor_copy SELECT * FROM autor WHERE  
id_autor IN (SELECT id_autor FROM autor)  
DELETE FROM autor_copy
```

Adăugare o nouă înregistrare

Exemplu

**INSERT INTO autor(nume_autor) SELECT autor
FROM carti WHERE cantitatea >= 40;**



Server: 7.0.0.1 » Database: librerie » Table: autor

Browse Structure SQL Search Insert Export

Edit Drop PRIMARY BTREE Yes No id_autor 11 A

Create an index on 1 columns Go

Partitions

No partitioning defined!

Information

Data	16.0 KiB	Format	dynamic
Index	0 B	Collation	utf8_general_ci
Overhead	0 B	Next autoindex	13
Effective	16.0 KiB	Creation	Nov 16, 2021 at 07:35 PM
Total	16.0 KiB	Last update	Nov 16, 2021 at 07:37 PM

id_autor	nume_autor
1	Petru
2	Diuma
3	Vieru
4	Lena
5	Ioana
6	Ghita
7	Colea
8	Nicu
9	Valea
10	GHITA
11	Vica

idcarte	aut
2-2222-222-10	Petru
2-2222-222-13	Vieru
2-2222-2222-22	Emil
2-2222-2222-23	Emil
2-2222-2222-6	Vica
2-2222-2222-8	Emil
2-2222-2222-9	Emil

Adăugare o nouă înregistrare

Exemplu

DELETE id_autor FROM autor WHERE id_autor > 9;

ALTER TABLE AUTOR AUTO_INCREMENT = 10;

**INSERT INTO autor(ume_autor) SELECT autor
FROM carti WHERE cantitatea >= 40;**

lu	pret	cantitatea	id_autor
laborator Mysql-Php vbvbvb	950.00	23	1
AMA	2000.00	200	3
id Php	1000.00	3	2
id Php	1000.00	3	2
ei trei muschetari Php	930.00	40	4
id Php	1000.00	3	6
id Php	1000.00	3	2

id_autor	nume_autor
1	Petru
2	Diuma
3	Vieru
4	Lena
5	Ioana
6	Ghita
7	Colea
8	Nicu
9	Valea
10	GHITA
11	Vica



Server: 127.0.0.1 »

Browse Structure

Edit Drop PRIMARY

Create an index on 1

Partitions ⓘ

⚠ No partitioning defined

Information

Data	16.0
Index	0
Overhead	0
Effective	16.0
Total	16.0

Optimize table

Space usage

Limbajul SQL

1. Limbajul de manipulare al datelor (LMD)

1. Adăugare o nouă înregistrare
2. **Actualizarea datelor dintr-o tabelă**
3. Ștergerea tuplurilor dintr-o tabelă
4. Instrucțiunea Merge

2. Limbajul de control al datelor (LCD).

Tranzacții

Actualizarea datelor dintr-o tabelă

Schimbarea liniilor existente folosind clauza **UPDATE**.

Sintaxa

```
UPDATE tabela  
SET coloana = valoare  
    [, coloana = valoare, . . .]  
[WHERE conditie ];
```

Actualizarea datelor dintr-o tabelă

În sintaxă:

UPDATE *tabela*

SET *coloana = valoare*

**[, coloana = valoare, . . .] [WHERE
conditie];**

- ***tabela*** - numele tablei
- ***coloana*** - numele coloanei în care vor fi introduse datele
- ***valoare*** - valoarea corespunzătoare din subquery (subinterogare)
- ***condiție*** - identificarea câmpurilor care vor fi actualizate

Actualizarea datelor dintr-o tabelă

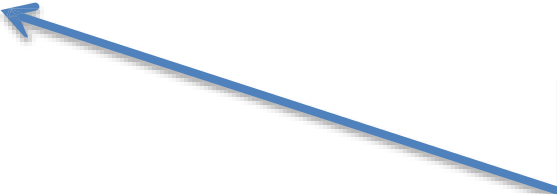
Notă:

- În general se folosește cheia primară pentru a identifica o linie.
- Utilizarea altei coloane poate duce la actualizarea mai multor linii.
- De exemplu într-o relație numita *persoane* putem avea de două sau mai multe persoane cu același nume.

Actualizarea datelor dintr-o tabelă

Exemplu 2

UPDATE angajati
SET nr_dept = 70
WHERE nr_angajat = 7499;



S-a modificat nr_dept
pentru angajatul cu
nr_angajat specificat

Actualizarea datelor dintr-o tabelă

Exemplu 1

```
UPDATE angajati  
SET functie='VANZATOR', data_angajarii = SYSDATE()  
WHERE nume = 'IONESCU';
```

În următorul exemplu, comanda UPDATE va seta coloana PRET la valoarea 200 pentru toate rândurile din tabelul de lucru.

```
UPDATE carti SET pret=200;
```

```
UPDATE carti  
SET id_autor= 7, data_n = SYSDATE()  
WHERE autor = ""Eminescu"";
```

Actualizarea datelor dintr-o tabelă

Exemplu 1

✓ 4 rows affected. (Query took 0.1832 seconds.)

```
UPDATE carti SET id_autor= DEFAULT, data_n = D
```

Mai întâi, să actualizăm coloana de **salariu**] in mod implicit folosind cuvântul cheie DEFAULT, așa cum se arată mai jos.

```
UPDATE lucrător SET salariu = DEFAULT;
```

```
UPDATE carti SET id_autor= DEFAULT, data_n =  
DEFAULT
```

```
WHERE autor = "Eminescu";
```

Revenim

```
UPDATE carti SET id_autor= 7, data_n = SYSDATE()
```

```
WHERE autor = "Eminescu";
```

idcarte	a
"2-2222-222-10"	"
"2-2222-222-13"	"
"2-2222-2222-1"	"
"2-2222-2222-2"	"
"2-2222-2222-22"	"
"2-2222-2222-23"	"
"2-2222-2222-4"	"
"2-2222-2222-5"	"
"2-2222-2222-6"	"
"2-2222-2222-7"	"
"2-2222-2222-8"	"
"2-2222-2222-9"	"
"idcarte"	"
2-2222-2222-01	F
2-2222-2222-25	F

Actualizarea datelor dintr-o tabelă

Exemplu 3

UPDATE carti

SET titlu = "Eminescu"

WHERE id_autor = 7;

id_autor	nume_autor
1	Petru
2	Diuma
3	Vieru
4	Lena
5	Ion
6	Ghita
7	Eminescu
8	Nicu
9	Valea

Exemplu 4

UPDATE carti

SET pret = pret+50

WHERE id_autor = 7;

idcarte	autor	data_n	titlu	pret	cantitatea	id_autor
"2-2222-222-10"	"Petru"	2021-11-18	"950.00"	200.00	255	1
"2-2222-222-13"	"Vieru"	0000-00-00	"2000.00"	200.00	255	3
"2-2222-2222-1"	"Diuma"	2021-11-18	"775.00"	200.00	255	2
"2-2222-2222-2"	"Diuma"	2021-11-18	"775.00"	200.00	255	2
"2-2222-2222-22"	"Eminescu"	2021-11-18	"1000.00"	200.00	255	7
"2-2222-2222-23"	"Eminescu"	2021-11-18	"1000.00"	200.00	255	7
"2-2222-2222-4"	"Petru"	2021-11-18	"155.00"	200.00	255	1
"2-2222-2222-5"	"Ion"	0000-00-00	"310.00"	200.00	255	5
"2-2222-2222-6"	"Vica"	0000-00-00	"930.00"	200.00	255	1
"2-2222-2222-7"	"Sandu"	0000-00-00	"465.00"	200.00	255	7
"2-2222-2222-8"	"Eminescu"	2021-11-18	"1000.00"	200.00	255	7
"2-2222-2222-9"	"Eminescu"	2021-11-18	"1000.00"	200.00	255	7
"idcarte"	"Petru"	2021-11-18	"pret"	200.00	255	1
2-2222-2222-01	Petru	2021-11-18	NULL	200.00	255	1
2-2222-2222-25	Fusu	NULL	NULL	200.00	255	2

Actualizarea datelor dintr-o tabelă

Exemplu 5

UPDATE carti

SET autor = "'Vica'", titlu='In lumea copiilor'

WHERE id_autor = 5;



idcarte	autor	data_n	titlu	pret	cantitatea	id_autor
"2-2222-222-10"	"Petru"	2021-11-18	"950.00"	200.00	255	1
"2-2222-222-13"	"Vieru"	0000-00-00	"2000.00"	200.00	255	3
"2-2222-2222-1"	"Diuma"	2021-11-18	"775.00"	200.00	255	2
"2-2222-2222-2"	"Diuma"	2021-11-18	"775.00"	200.00	255	2
"2-2222-2222-22"	"Eminescu"	2021-11-18	"1000.00"	200.00	255	7
"2-2222-2222-23"	"Eminescu"	2021-11-18	"1000.00"	200.00	255	7
"2-2222-2222-4"	"Petru"	2021-11-18	"155.00"	200.00	255	1
"2-2222-2222-5"	"Vica"	0000-00-00	In lumea copiilor	200.00	255	5
"2-2222-2222-6"	"Vica"	0000-00-00	"930.00"	200.00	255	1
"2-2222-2222-7"	"Sandu"	0000-00-00	"465.00"	200.00	255	7
"2-2222-2222-8"	"Eminescu"	2021-11-18	"1000.00"	200.00	255	7
"2-2222-2222-9"	"Eminescu"	2021-11-18	"1000.00"	200.00	255	7
"idcarte"	"Petru"	2021-11-18	"pret"	200.00	255	1
2-2222-2222-01	Petru	2021-11-18	NULL	200.00	255	1
2-2222-2222-25	Fusu	NULL	NULL	200.00	255	2

Actualizarea datelor dintr-o tabelă

Exemplu 6

UPDATE carti

SET pret = "200" **Limit 3**



Exemplu 7

UPDATE carti

SET pret = "500"



idcarte	autor	data_n	titlu	pret	cantitatea	id_autor
"2-2222-222-10"	"Petru"	2021-11-18	"950.00"	200.00	255	1
"2-2222-222-13"	"Vieru"	0000-00-00	"2000.00"	200.00	255	3
"2-2222-2222-1"	"Diuna"	2021-11-18	"775.00"	200.00	255	2
"2-2222-2222-2"	"Diuna"	2021-11-18	"775.00"	200.00	255	2
"2-2222-2222-22"	"Eminescu"	2021-11-18	"1000.00"	200.00	255	7
"2-2222-2222-23"	"Eminescu"	2021-11-18	"1000.00"	200.00	255	7
"2-2222-2222-4"	"Petru"	2021-11-18	"155.00"	200.00	255	1
"2-2222-2222-5"	"Vica"	0000-00-00	In lumea copiilor	200.00	255	5
"2-2222-2222-6"	"Vica"	0000-00-00	"930.00"	200.00	255	1
"2-2222-2222-7"	"Sandu"	0000-00-00	"465.00"	200.00	255	7
"2-2222-2222-8"	"Eminescu"	2021-11-18	"1000.00"	200.00	255	7
"2-2222-2222-9"	"Eminescu"	2021-11-18	"1000.00"	200.00	255	7
"idcarte"	"Petru"	2021-11-18	"pret"	200.00	255	1
2-2222-2222-01	Petru	2021-11-18	NULL	200.00	255	1
2-2222-2222-25	Fusu	NULL	NULL	200.00	255	2

Actualizarea datelor dintr-o tabelă

Exemplu 7

UPDATE carti

SET pret =

Case autor

When "'Vica'", then cantitatea = 200

When "'Vica'", then cantitatea = 300

Else cantitatea

End;

idcarte	autor	data_n
"2-2222-222-10"	"Petru"	2021-11-18
"2-2222-222-13"	"Vieru"	0000-00-00
"2-2222-2222-1"	"Diuma"	2021-11-18
"2-2222-2222-2"	"Diuma"	2021-11-18
"2-2222-2222-22"	"Eminescu"	2021-11-18
"2-2222-2222-23"	"Eminescu"	2021-11-18
"2-2222-2222-4"	"Petru"	2021-11-18
"2-2222-2222-5"	"Vica"	0000-00-00
"2-2222-2222-6"	"Vica"	0000-00-00
"2-2222-2222-7"	"Sandu"	0000-00-00
"2-2222-2222-8"	"Eminescu"	2021-11-18
"2-2222-2222-9"	"Eminescu"	2021-11-18
"idcarte"	"Petru"	2021-11-18
2-2222-2222-01	Petru	2021-11-18
2-2222-2222-25	Fusu	NULL

Actualizarea datelor dintr-o tabelă

Exemplu 7



Presupunem că avem un tabel CUSTOMERS_BKP care este o copie de rezervă a tabelului Clienți. Următorul exemplu actualizează SALARIUL de 0,25 ori în tabelul de clienți pentru toți clienții care au vârsta mai mare sau egală cu 37.

```
UPDATE CUSTOMERS SET SALARY = SALARY * 1.2  
WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP  
WHERE AGE >= 37 );
```

Creați o copie a tabelului *carti*

Apoi modificați **pretul** pentru care **cantitatea** de carti este mai mare decit 10

Limbajul SQL

1. Limbajul de manipulare al datelor (LMD)

1. Adăugare o nouă înregistrare
2. Actualizarea datelor dintr-o tabelă
3. Ștergerea tuplurilor dintr-o tabelă
4. Instrucțiunea Merge

2. Limbajul de control al datelor (LCD).

Tranzacții

Ștergerea tuplurilor dintr-o tabelă

Se pot șterge tupluri dintr-o tabelă utilizând clauza **DELETE**.

Sintaxa

DELETE [FROM] *tabela*

[WHERE *conditie*];

Ștergerea tuplurilor dintr-o tabelă

În sintaxa:

DELETE [FROM] *tabela*
[WHERE *conditie*];

- **tabela** - numele tablei
- **condiție** - identifică liniile care trebuie șterse și este compusă din:
 1. nume de câmpuri
 2. expresii
 3. constante
 4. subquery-uri
 5. și operatori de comparație

Ștergerea tuplurilor dintr-o tabelă

- Șterge anumite tupluri dintr-o tabelă specificând clauza **WHERE** în declarația funcției **DELETE**.
- Se poate confirma operația de ștergere prin afișarea tuplurilor șterse cu ajutorul declarației lui **SELECT**.

Ștergerea tuplurilor dintr-o tabelă

În exemplul următor șterge angajații care lucrează în departamentul 10 din tabela Angajati.

```
DELETE FROM angajati  
WHERE nr_dept = 10;
```

Ștergerea tuplurilor dintr-o tabelă

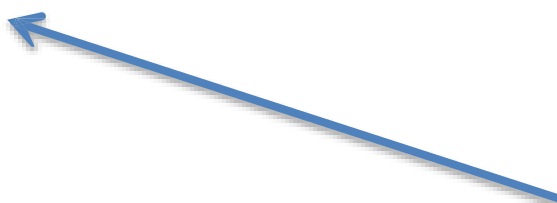
- Dacă se omite clauza **WHERE** toate câmpurile din tabelă vor fi șterse.
- Al doilea exemplu șterge toate tuplurile(înregistrările) din tabela Copie_angajati deoarece nu a fost specificată clauza **WHERE**.

DELETE FROM copie_angajati;

Ștergerea tuplurilor dintr-o tabelă

Se pot șterge și mai multe linii.

```
DELETE FROM angajati  
WHERE nr_dept IN (10, 20);
```



S-au sters angajatii care lucreaza in departamentele identificate cu nr_dept specificate

Ștergerea tuplurilor dintr-o tabelă

Exemplu 7

Presupunând că avem un tabel **CUSTOMERS_BKP** care este o copie de rezervă a tabelului **Clienți**. Următorul exemplu șterge înregistrările din tabelul **CLIENTI** pentru toți clienții a căror vârstă este de 37 de ani sau mai mult.

```
DELETE FROM CUSTOMERS WHERE AGE IN (SELECT  
AGE FROM CUSTOMERS_BKP WHERE AGE >= 37 );
```

Limbajul SQL

1. Limbajul de manipulare al datelor (LMD)

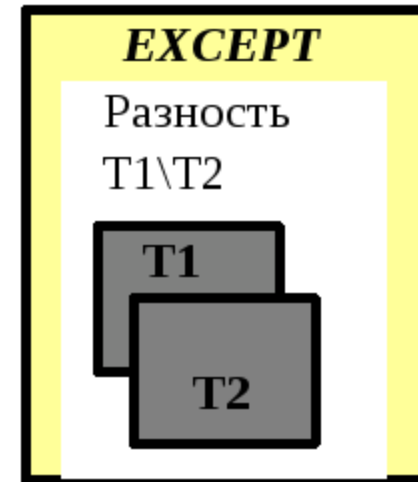
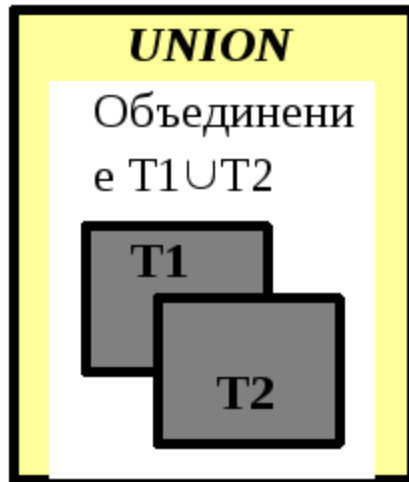
1. Adăugare o nouă înregistrare
2. Actualizarea datelor dintr-o tabelă
3. Ștergerea tuplurilor dintr-o tabelă
4. **UNION, INTERSECT, EXEPT**
5. Asocieri ale tabelelor unei BD, **JOIN**
6. Instrucțiunea Merge

2. Limbajul de control al datelor (LCD).

Tranzacții

REUNIUNE, INTERSECȚIE ȘI DIFERENȚĂ

Три типа операций с результатом подзапросов



Autor

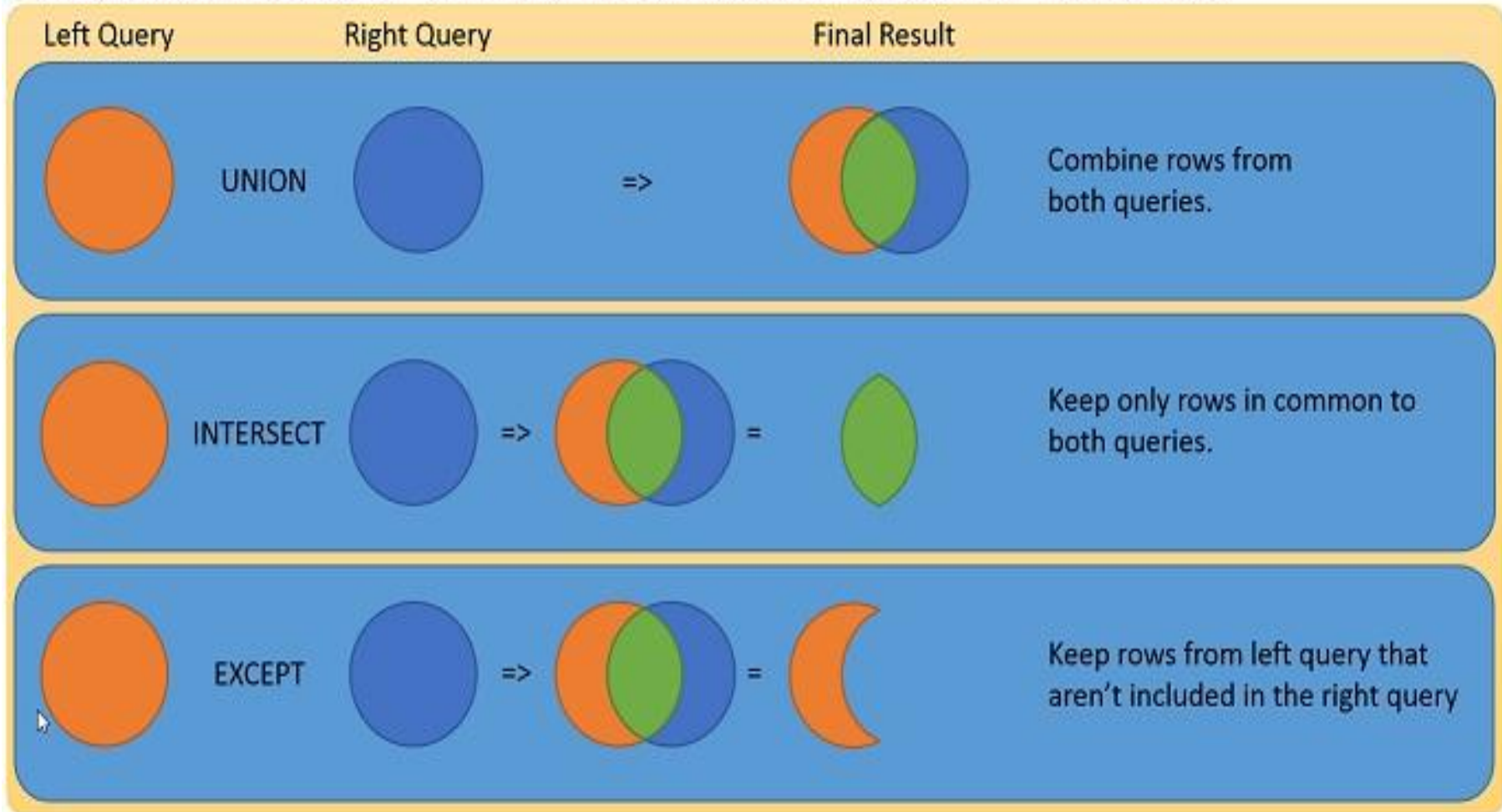
id_autor	nume_autor
1	Petru
2	Diuma
3	Vieru
4	Lena
5	Ioana
6	Ghita
7	Colea
8	Nicu
9	Valea

Carti

idcarte	autor	titlu	pret	cantitatea	id_autor
2-2222-222-10	Petru	GFGF Laborator Mysql-Php vbvbnb	950.00	23	1
2-2222-222-13	Vieru	MAMA	2000.00	200	3
2-2222-2222-22	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-23	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-6	Vica	Cei trei muschetari Php	930.00	40	4
2-2222-2222-8	Eminescu	Ghid Php	1000.00	3	6
2-2222-2222-9	Eminescu	Ghid Php	1000.00	3	2

REUNIUNE, INTERSECȚIE ȘI DIFERENȚĂ

Visual Explanation of UNION, INTERSECT, and EXCEPT operators



REUNIUNE, intersecție și diferență

UNION (reuniune) se folosește pentru a îmbina rezultatele a două sau mai multe interogări într- un singur rezultat-set

Sintaxa:

```
SELECT <col1>, <col2>, <col3> FROM table1
```

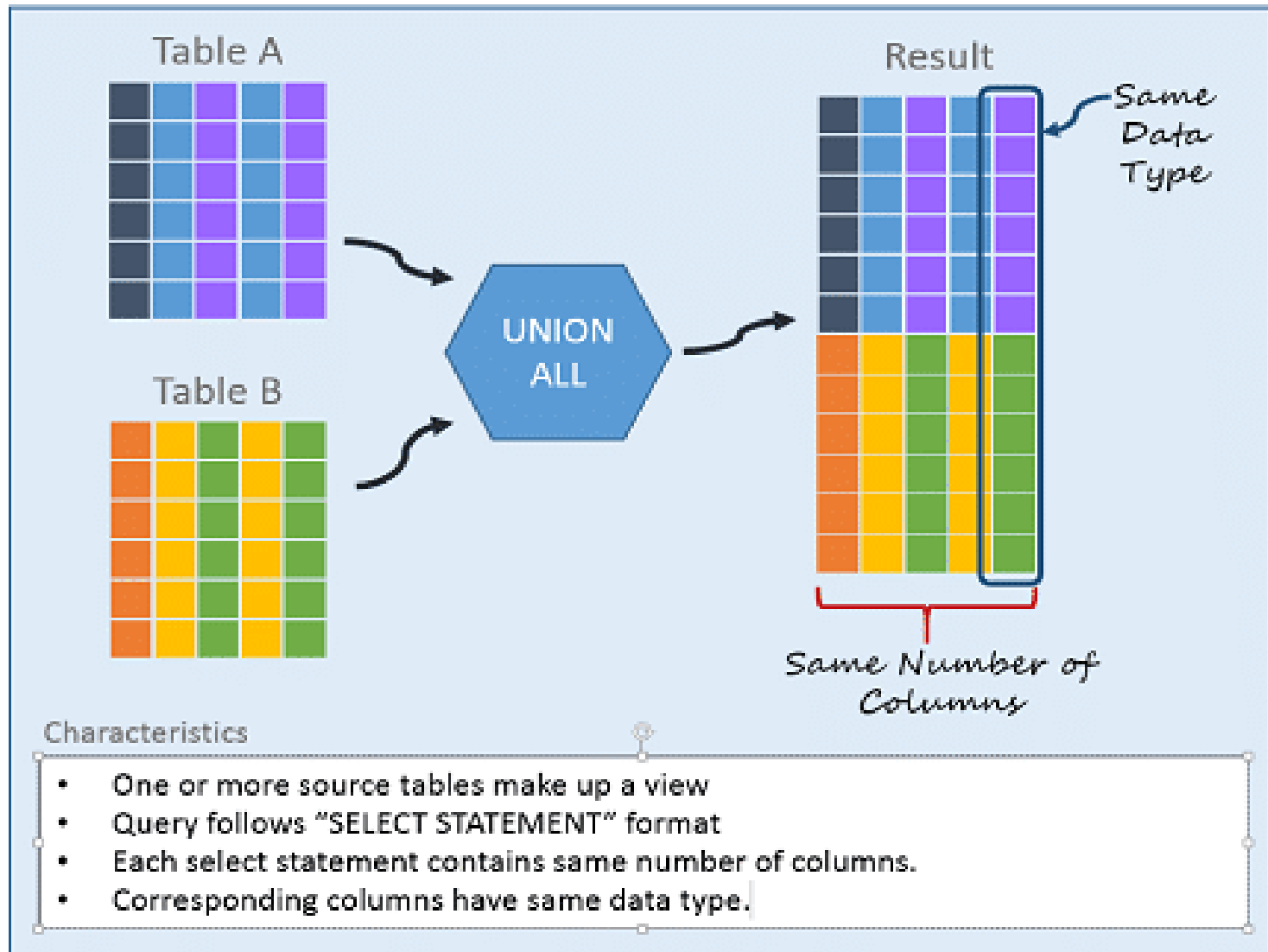
```
UNION [ALL]
```

```
SELECT <col4>, <col5>, <col6> FROM table2;
```

Fiecare interogare trebuie să conțină același număr de coloane, iar tipurile coloanelor trebuie să fie compatibile

REUNIUNE. intersectie si diferentă

Anatomy of a UNION



REUNIUNE, intersecție și diferență

```
DROP TABLE IF EXISTS t1;  
DROP TABLE IF EXISTS t2;
```

```
CREATE TABLE t1 (  
  id INT PRIMARY KEY  
);
```

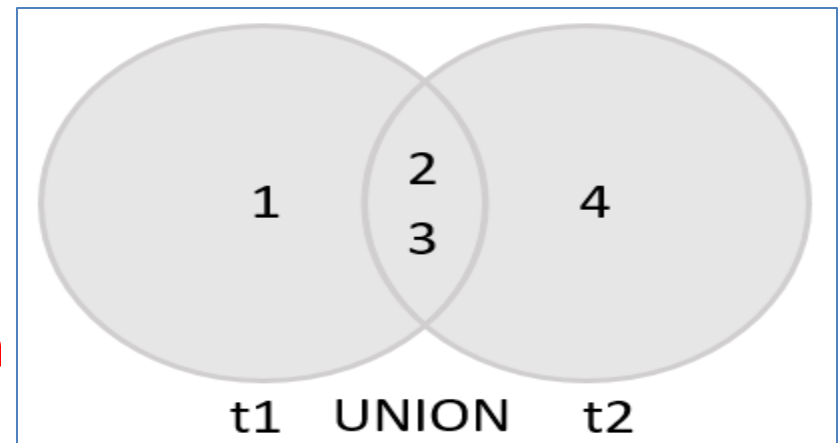
```
CREATE TABLE t2 (  
  id INT PRIMARY KEY  
);
```

```
INSERT INTO t1 VALUES (1),(2),(3);  
INSERT INTO t2 VALUES (2),(3),(4);
```

```
SELECT id FROM t1  
UNION  
SELECT id FROM t2;  
[ ORDER BY <lista> ] !! La al doilea  
SELECT
```

Dacă utilizăm UNION ALL în mod explicit, rândurile duplicate, dacă sunt disponibile, rămân în rezultat. Deoarece UNION ALL nu trebuie să gestioneze duplicatele, funcționează mai rapid decât UNION DISTINCT .

```
SELECT id FROM t1  
UNION ALL  
SELECT id FROM t2;
```



REUNIUNE, intersecție și diferență

SELECT id FROM t1
UNION
SELECT id FROM t2;

ATENȚIE!!!

id
1
2
3

UNION

id
2
3
4



id
1
2
3
4

Append
result sets
vertically

id
1
2
3

INNER
JOIN

id
2
3
4



id	id
2	2
3	3

Append
result sets
horizontally

customers
* customerNumber
customerName
contactLastName
contactFirstName
phone
addressLine1
addressLine2
city
state
postalCode
country
salesRepEmployeeNumber
creditLimit

employees
* employeeNumber
lastName
firstName
extension
email
officeCode
reportsTo
jobTitle

SELECT firstName, lastName
FROM employees

UNION

SELECT contactFirstName,
contactLastName
FROM customers;

După cum putem vedea din ieșire,
MySQL **UNION** utilizează numele de
coloane ale primei instrucțiuni SELECT
pentru titlurile de coloană ale ieșirii.

REUNIUNE, intersecție și diferență

UNION ALL va include înregistrări duplicate

Exemplu (cu duplicate):

```
SELECT nume FROM Clienți  
UNION ALL  
SELECT nume FROM Angajați;
```

Exemplu (fără duplicate):

```
SELECT nume FROM Clienți  
UNION  
SELECT nume FROM Angajați;
```

```
SELECT 'Vendor'  
FROM Vendor V  
UNION  
SELECT 'Customer'  
FROM Customer C  
BY Name
```

REUNIUNE, intersecție și diferență

```
SELECT 'Person' AS  
Source, FirstName + '  
' + LastName AS Name  
FROM person.Person  
UNION  
SELECT 'Vendor', Name  
FROM Purchasing.Vendor  
UNION  
SELECT 'Store', Name  
FROM Sales.Store  
ORDER BY Name;
```

```
SELECT 'Vendor',  
V.Name  
FROM Vendor V  
UNION  
SELECT 'Customer',  
C.Name  
FROM Customer C  
ORDER BY Name
```

REUNIUNE, intersecție și diferență

```
SELECT NUME_AUTOR  
FROM AUTOR  
UNION  
SELECT AUTOR  
FROM CARTI  
ORDER BY NUME_AUTOR
```

+ Options
NUME_AUTOR ▲ 1
Diuma
Eminescu
Ghita
Ion
Lena
Nicu
Petru
Valea
Vica
Vieru

```
SELECT NUME_AUTOR  
FROM AUTOR  
UNION ALL  
SELECT AUTOR  
FROM CARTI  
ORDER BY NUME_AUTOR
```

+ Options
NUME_AUTOR ▲ 1
Diuma
Diuma
Diuma
Diuma
Eminescu
Eminescu
Eminescu
Eminescu
Ghita
Ion
Lena
Lena
Nicu
Petru
Petru
Petru
Petru
Petru
Valea
Valea

id_autor	nume_autor
1	Petru
2	Diuma
3	Vieru
4	Lena
5	Ioana
6	Ghita
7	Colea
8	Nicu
9	Valea

idcarte	autor	titlu
2-2222-222-10	Petru	GFGF Laborator Mysql-Php
2-2222-222-13	Vieru	MAMA
2-2222-2222-22	Eminescu	Ghid Php
2-2222-2222-23	Eminescu	Ghid Php
2-2222-2222-6	Vica	Cei trei muschetari
2-2222-2222-8	Eminescu	Ghid Php
2-2222-2222-9	Eminescu	Ghid Php

Reuniune, INTERSECȚIE și diferență

INTERSECT – intersecția este folosită pentru a returna într-un singur rezultat-set acele înregistrări care apar atât în rezultatul-set al interogării din **partea dreaptă** cât și în cel al interogării din **partea stângă**

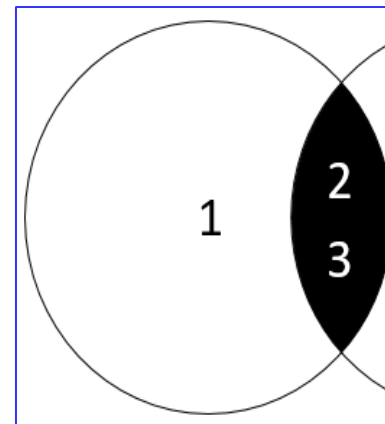
1. **Ordinea și numărul de coloane din lista de selecție a interogărilor trebuie să fie aceleași.**
2. **Tipurile de date ale coloanelor corespunzătoare trebuie să fie compatibile**

Sintaxa:

```
SELECT <col1>, <col2>, <col3> FROM  
table1
```

INTERSECT

```
SELECT <col4>, <col5>, <col6> FROM  
table2;
```



Reuniune, INTERSECȚIE și diferență

```
(SELECT NUME_AUTOR  
FROM AUTOR)
```

INTERSECT

```
(SELECT AUTOR FROM CARTI)  
ORDER BY NUME_AUTOR
```

```
(SELECT column_list  
FROM table_1)
```

INTERSECT

```
(SELECT column_list  
FROM table_2);
```



+ Options
NUME_AUTOR
Diuma
Eminescu
Lena
Petru
Valea
Vieru

idcarte	autor	titlu	preț
2-2222-222-10	Petru GFGF	Laborator Mysql-Php vbvbvb	950.0
2-2222-222-13	Vieru	MAMA	2000.0
2-2222-2222-22	Eminescu	Ghid Php	1000.0
2-2222-2222-23	Eminescu	Ghid Php	1000.0
2-2222-2222-6	Vica	Cei trei muschetari Php	930.0
2-2222-2222-8	Eminescu	Ghid Php	1000.0
2-2222-2222-9	Eminescu	Ghid Php	1000.0

Exemplu:

```
SELECT nume, prenume FROM Clienți
```

INTERSECT

```
SELECT nume, prenume FROM Angajați
```

INTERSECT

```
SELECT nume, prenume FROM Furnizori;
```

Reuniune, INTERSECȚIE și diferență

Emularea INTERSECT in MYSQL

1. Emularea INTERSECT folosind DISTINCT si clauza INNER JOIN

```
SELECT DISTINCT id  
FROM t1  
INNER JOIN t2 USING(id);
```

2. Emularea INTERSECT folosind IN si un subquery

```
SELECT DISTINCT id  
FROM t1  
WHERE id IN (SELECT id FROM t2);
```

Exemplu:

```
SELECT nume, prenume FROM Clienți
```

```
INTERSECT
```

```
SELECT nume, prenume FROM Angajați
```

```
INTERSECT
```

```
SELECT nume, prenume FROM Furnizori;
```

idcarte	autor	titlu
2-2222-222-10	Petru GFGF	Laborator My
2-2222-222-13	Vieru	MAMA
2-2222-2222-22	Eminescu	Ghid Php
2-2222-2222-23	Eminescu	Ghid Php
2-2222-2222-6	Vica	Cei trei musch
2-2222-2222-8	Eminescu	Ghid Php
2-2222-2222-9	Eminescu	Ghid Php

Reuniune, INTERSECȚIE și diferență

Emularea INTERSECT in MYSQL

1. Emularea INTERSECT folosind DISTINCT si clauza INNER JOIN

```
SELECT DISTINCT nume_autor  
FROM autor A  
INNER JOIN carti B  
ON B.autor=A.nume_autor;
```

+ Options
nume_autor
Petru
Vieru
Diuma
Eminescu
Valea
Lena

idcarte	autor	titlu
2-2222-222-10	Petru	GFGF Laborator My
2-2222-222-13	Vieru	MAMA
2-2222-2222-22	Eminescu	Ghid Php
2-2222-2222-23	Eminescu	Ghid Php
2-2222-2222-6	Vica	Cei trei musch
2-2222-2222-8	Eminescu	Ghid Php
2-2222-2222-9	Eminescu	Ghid Php

2. Emularea INTERSECT folosind IN si un subquery

```
SELECT DISTINCT nume_autor  
FROM AUTOR  
WHERE nume_autor IN (SELECT autor FROM CARTI);
```

nume_autor
Petru
Diuma
Vieru
Lena
Eminescu
Valea

Reuniune, intersecție și DIFERENȚĂ

EXCEPT (diferență) este folosită pentru a returna acele înregistrări care apar în rezultatul-set al interogării din **partea stângă** dar **nu apar** în rezultatul-set al interogării din **partea dreaptă**

Sintaxa:

```
SELECT <col1>, <col2>, <col3>  
FROM table1  
EXCEPT  
SELECT <col4>, <col5>, <col6>  
FROM table2;
```

Reuniune, intersecție și DIFERENȚĂ

```
DROP TABLE IF EXISTS t1;  
DROP TABLE IF EXISTS t2;
```

```
CREATE TABLE t1 (  
  id INT PRIMARY KEY  
);
```

```
CREATE TABLE t2 (  
  id INT PRIMARY KEY  
);
```

```
INSERT INTO t1 VALUES (1),(2),(3);  
INSERT INTO t2 VALUES (2),(3),(4);
```

```
SELECT id FROM t1  
MINUS / EXCEPT →  
SELECT id FROM t2;  
[ ORDER BY <lista> ] !! La al doilea  
SELECT
```

id
1
2
3

t1 table

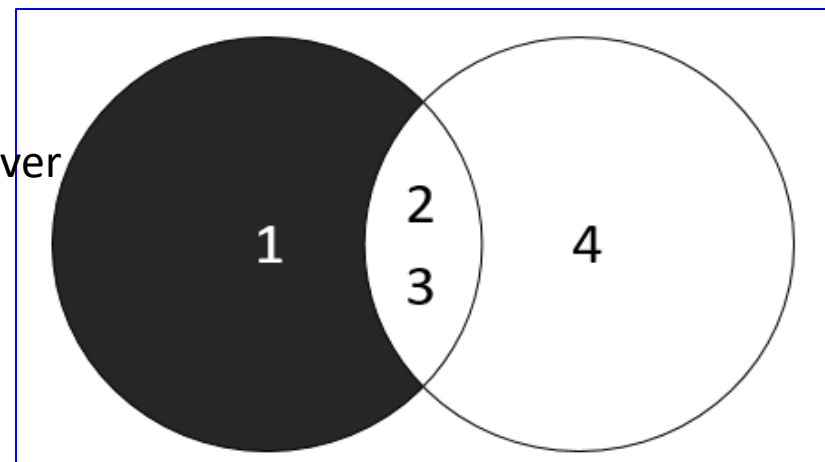
MINUS

id
2
3
4

t2 table



id
1



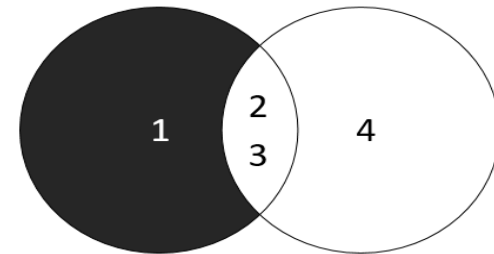
Reuniune, intersecție și DIFERENȚĂ

Emularea comenzii MINUS / EXCEPT

1 ABORDARE

```
SELECT select_list  
FROM table1  
LEFT JOIN table2  
ON join_predicate  
WHERE table2.column_name IS NULL;
```

```
SELECT id  
FROM t1  
LEFT JOIN t2  
USING (id)  
WHERE t2.id IS NULL;
```



id_autor	nume_autor
1	Petru
2	Diuma
3	Vieru
4	Lena
5	Ioana
6	Ghita
7	Colea
8	Nicu
9	Valea

```
SELECT NUME_AUTOR  
FROM AUTOR  
EXCEPT  
SELECT AUTOR  
FROM CARTI  
ORDER BY NUME_AUTOR
```

+ Options

NUME_AUTOR

Ghita

Ion

Nicu

idcarte	autor	titlu	pret	cantitatea	id_autor
2-2222-222-10	Petru	GFGF Laborator Mysql-Php vbvbvb	950.00	23	1
2-2222-222-13	Vieru	MAMA	2000.00	200	3
2-2222-2222-22	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-23	Eminescu	Ghid Php	1000.00	3	2
2-2222-2222-6	Vica	Cei trei muschetari Php	930.00	40	4
2-2222-2222-8	Eminescu	Ghid Php	1000.00	3	6
2-2222-2222-9	Eminescu	Ghid Php	1000.00	3	2

Reuniune, intersecție și DIFERENȚĂ

```
SELECT select_list  
FROM table1  
LEFT JOIN table2  
ON join_predicate  
WHERE table2.column_name IS NULL;
```



```
SELECT NUME_AUTOR  
FROM AUTOR A  
LEFT JOIN CARTI B  
ON A.NUME_AUTOR=B.AUTOR  
WHERE B.AUTOR IS NULL  
ORDER BY NUME_AUTOR;
```

Exemplu

```
SELECT nume, prenume FROM Clienți  
EXCEPT  
SELECT nume, prenume FROM Angajați;
```

Exemplu

```
SELECT id_client FROM Clienți  
EXCEPT  
SELECT id_client FROM Comenzi;
```

+ Optio
NUME
Ion
Ghita
Nicu

idcarte	autor
2-2222-222-10	Petru GFG
2-2222-222-13	Vieru
2-2222-222-22	Eminescu
2-2222-222-23	Eminescu
2-2222-222-6	Vica
2-2222-222-8	Eminescu
2-2222-222-9	Eminescu

Limbajul SQL

1. Limbajul de manipulare al datelor (LMD)

1. Adăugare o nouă înregistrare
2. Actualizarea datelor dintr-o tabelă
3. Ștergerea tuplurilor dintr-o tabelă
4. UNION, INTERSEPT, EXEPT
5. Asocieri ale tabelelor unei BD, JOIN
6. Instrucțiunea Merge

2. Limbajul de control al datelor (LCD).

Tranzacții

LIMBAJUL SQL

Selectie

Tabela 1

Proiectie

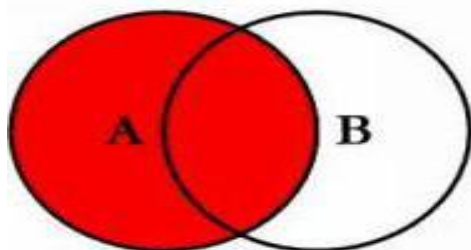
Tabela 1

Join

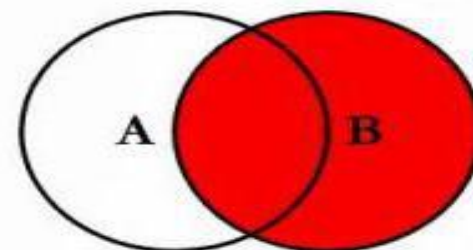
Tabela 1

Tabela 2

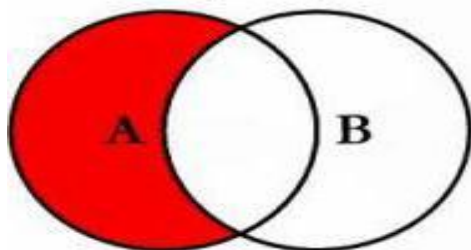
SQL JOINS



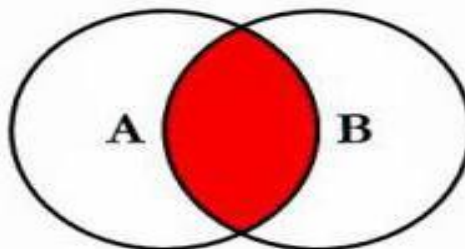
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



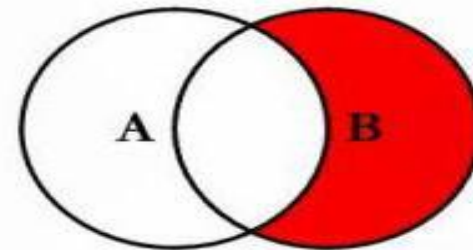
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



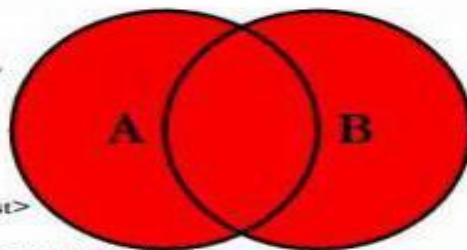
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



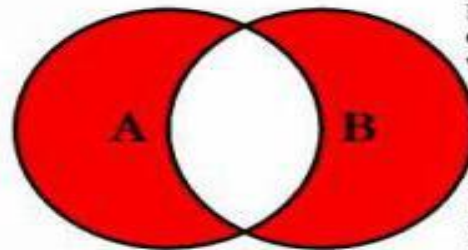
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Asocieri, **JOIN**-urile se aplică minimum asupra a 2 tabele R și S, care au în structura lor un atribut comun, denumit atribut de asociere și generează o nouă tabelă T cu o structură complexă formată implicit din multimea atributelor tabelelor sursă/componente.

Asocieri, JOIN-uri

Operatorul de Asociere **JOIN** utilizează în comanda **SELECT**, atunci cind sursa de date se prezintă prin mai multe tabele.

SELECT lista

FROM SALARIU A, DEPARTAMENT B

Operatorii traditionali de asociere sunt:

INNER JOIN

LEFT JOIN

RIGHT JOIN

OUTER JOIN

FULL JOIN, CROSS JOIN, NATURAL JOIN

Asocieri, JOIN-uri

Sintaxa:

SELECT <lista>

FROM <lista tabel> ①

[cuvint cheie] **JOIN** <nume tabel> ②

ON <criteriu de asociere> **USING** <clause>

[Criteriu de selectie **WHERE** <expr>]

[**ORDER BY** <lista cimpuri>]

Cuvint cheie

INNER

LEFT

RIGHT

OUTER

FULL

CROSS

NATURAL

Exista 2 abordări în explicarea cum lucrează **JOIN**

- Prin teoria multimilor
- Prin intermediul diagramelor
- Dacă numele coloanelor sunt aceleași în ambele tabele pentru care utilizăm Asocierea JOIN, atunci in loc de ON <conditie> utilizam USING (cimp)
- **SELECT * FROM carti JOIN autor USING(id_autor)**

Asocieri, JOIN-uri

Sintaxa:

```
CREATE DATABASE dbjoin;  
CREATE TABLE TabelulA(  
    id INT NOT NULL AUTO_INCREMENT,  
    nume VARCHAR(30) NOT NULL,  
    PRIMARY KEY (id )  
);  
CREATE TABLE TabelulB(  
    id INT NOT NULL AUTO_INCREMENT,  
    nume VARCHAR(30) NOT NULL,  
    PRIMARY KEY (id )  
);
```

Asocieri, JOIN-uri

Sintaxa:

INSERT INTO tabelula(id, nume)
VALUES (1, 'Ion'), (2, 'Nata'), (3, 'Petru'), (4, 'Ghita');

INSERT INTO tabelulb(id, nume)
VALUES (1, 'Nicolai'), (2, 'Ion'), (3, 'Elena'), (4, 'Petru');

TabelulA

id	nume
1	Ion
2	Nata
3	Petru
4	Ghita

TabelulB

id	nume
1	Nicolai
2	Ion
3	Elena
4	Petru

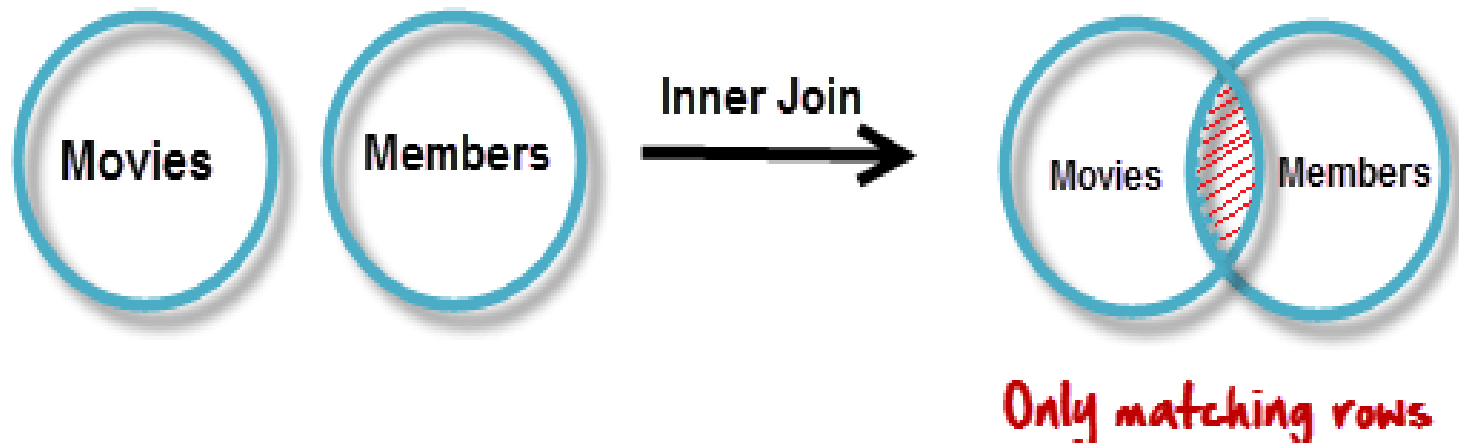
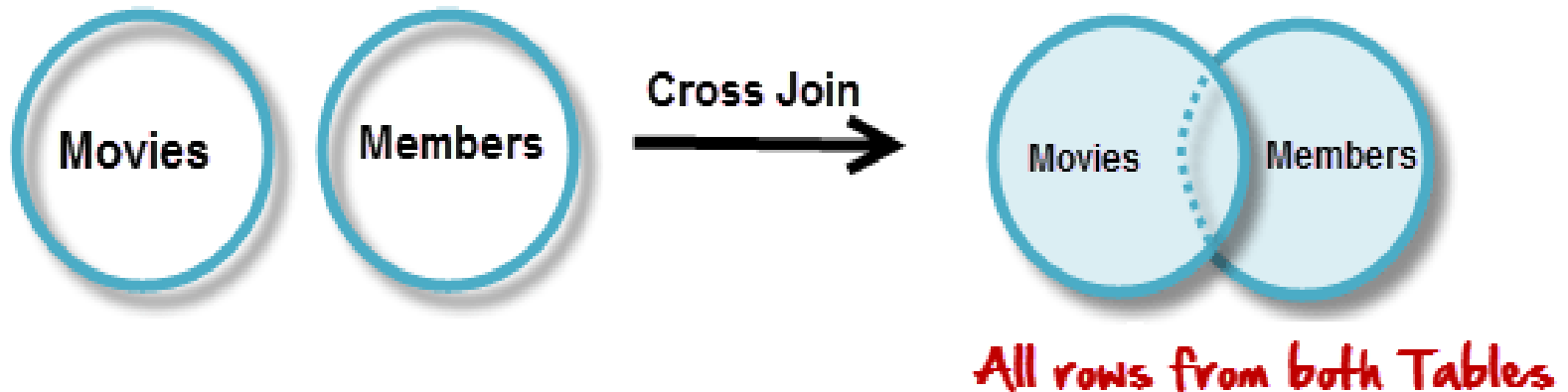
Tabelul A

id	nume
1	Ion
2	Nata
3	Petru
4	Ghita

Tabelul B

id	nume
1	Nicolai
2	Ion
3	Elena
4	Petru

CROSS JOIN / INNER JOIN



Asocieri, JOIN-uri

Reamintim că în laboratorul 2, Anexa 3, ati utilizat

- **CROSS JOIN** - mai realizează si produsul cartezian
- **NATURAL JOIN** – tabelele trebuie să aibă coincidente numele unor coloane
- Vom precăuta in continuare 2 tabele A si B nelegate intre ele, adica

SELECT *

FROM TabelulA

INNER JOIN TabelulB

ON TabelulA.numa=TabelulB.numa

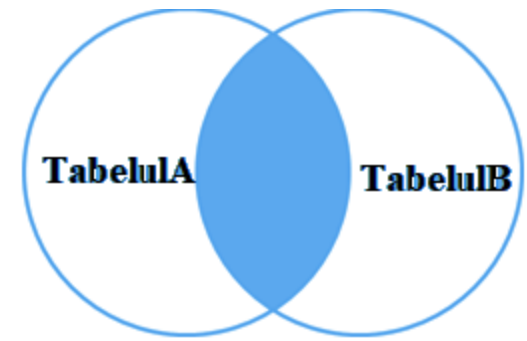
Rezultatul

id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru

+ Options

id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru

Tabelul A		Tabelul B	
id	numa	id	numa
1	Ion	1	Nicolai
2	Nata	2	Ion
3	Petru	3	Elena
4	Ghita	4	Petru



Asocieri, JOIN-uri, INNER

```
SELECT *  
FROM TabelulA  
INNER JOIN TabelulB  
ON TabelulA.numa=TabelulB.numa
```

```
SELECT *  
FROM TabelulA  
CROSS JOIN TabelulB  
ON TabelulA.numa=TabelulB.numa
```

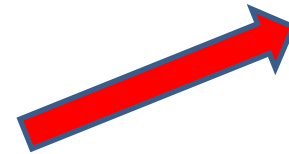
CONCLUZIE

Asocierea INNER este echivalentă cu Asocierea CROSS JOIN care efectuează produsul cartezian TabelulA x TabelulB, pentru care clauza ON este TRUE

id	nume	id	nume
1	Ion	2	Ion
3	Petru	4	Petru

+ Options

id	nume	id	nume
1	Ion	2	Ion
3	Petru	4	Petru



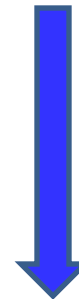
Showing rows 0 - 1 (2 total, Query to)

```
SELECT * FROM TabelulA CROSS JOIN
```

☐ Show all | Number of rows: 2

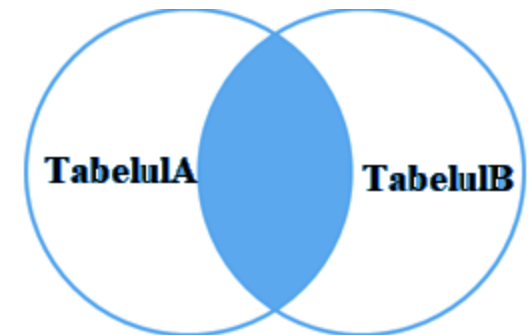
+ Options

id	nume	id	nume
1	Ion	2	Ion
3	Petru	4	Petru

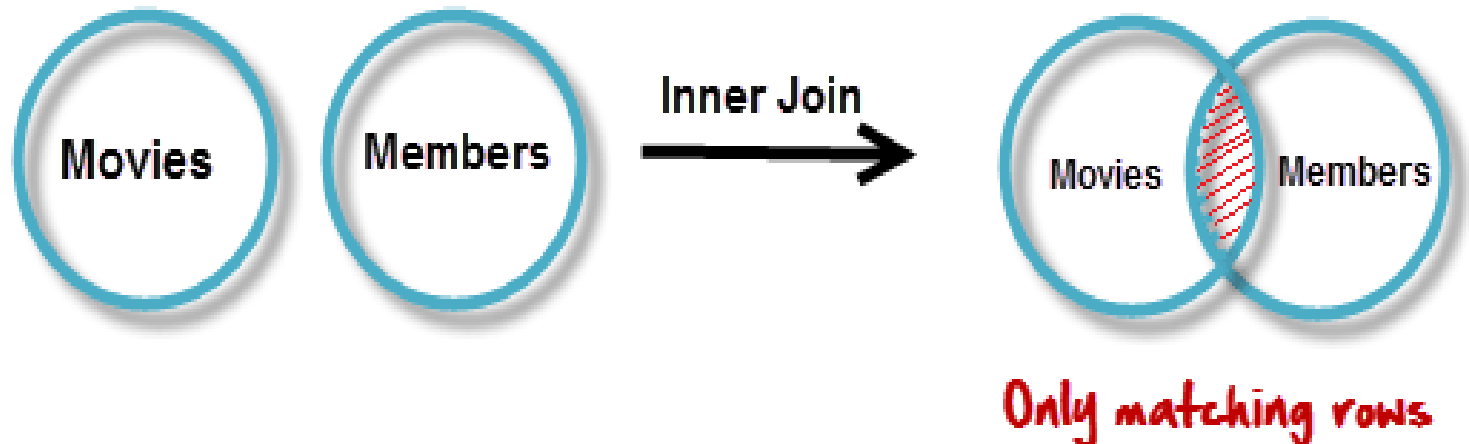


Tabelul A	
id	nume
1	Ion
2	Nata
3	Petru
4	Ghita

Tabelul B	
id	nume
1	Nicolai
2	Ion
3	Elena
4	Petru



CROSS JOIN / LEFT JOIN



Asocieri, JOIN-uri, LEFT OUTER

SELECT *
FROM TabelulA
LEFT OUTER JOIN TabelulB
ON TabelulA.nume=TabelulB.nume

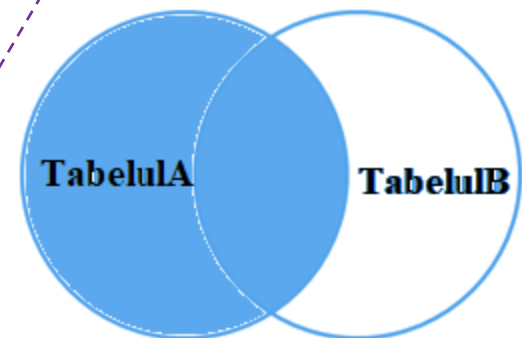
SELECT *
FROM TabelulA
INNER JOIN TabelulB
ON TabelulA.nume=TabelulB.nume

CONCLUZIE

Asocierea **LEFT OUTER JOIN** este echivalentă cu Asocierea **INNER JOIN**, numai că după ce obținem rezultatul **INNER**, în rândurile din tabelul din STINGA (**TabelulA**) pentru care **NU EXISTĂ** coincidență cu cel din DREAPTA (**TabelulB**) se completează cu **NULL** în Tabelul din DREAPTA (**TabelulB**)

+ Options			
id	nume	id	nume
1	Ion	2	Ion
3	Petru	4	Petru
2	Nata	NULL	NULL
4	Ghita	NULL	NULL

+ Options			
id	nume	id	nume
1	Ion	2	Ion
3	Petru	4	Petru



Tabelul A	
id	nume
1	Ion
2	Nata
3	Petru
4	Ghita

Tabelul B	
id	nume
1	Nicolai
2	Ion
3	Elena
4	Petru

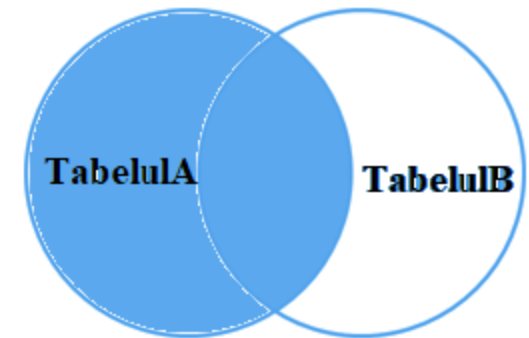
id	nume	id	nume
1	Ion	2	Ion
3	Petru	4	Petru
2	Nata	NULL	NULL
4	Ghita	NULL	NULL

Asocieri, JOIN-uri, LEFT OUTER

SELECT *
FROM TabelulA
LEFT OUTER JOIN TabelulB
ON TabelulA.numa=TabelulB.numa
Rezultatul

Tabelul A	
id	numa
1	Ion
2	Nata
3	Petru
4	Ghita

Tabelul B	
id	numa
1	Nicolai
2	Ion
3	Elena
4	Petru

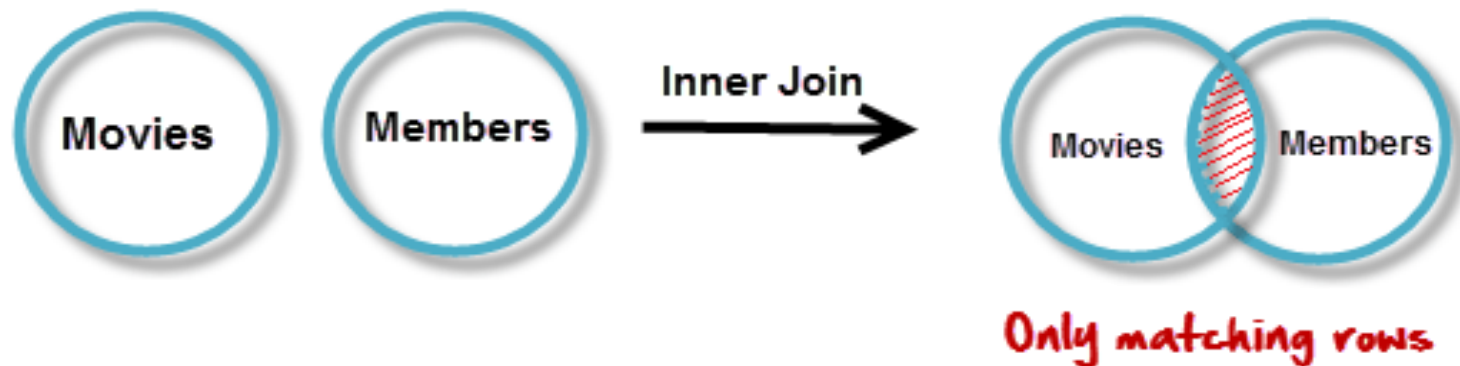


+ Options

id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
2	Nata	NULL	NULL
4	Ghita	NULL	NULL

id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
2	Nata	NULL	NULL
4	Ghita	NULL	NULL

CROSS JOIN / RIGHT JOIN



<https://www.guru99.com/joins.html>

Asocieri, JOIN-uri, RIGHT OUTER

SELECT *
FROM TabelulA
RIGHT OUTER JOIN TabelulB
ON TabelulA.nume=TabelulB.nume

SELECT *
FROM TabelulA
INNER JOIN TabelulB
ON TabelulA.nume=TabelulB.nume

CONCLUZIE

Asocierea **RIGHT OUTER JOIN** este echivalentă cu Asocierea **INNER JOIN**, numai că după ce obținem rezultatul **INNER**, în rândurile din tabelul din DREAPTA (**TabelulB**) pentru care **NU EXISTĂ** coincidență cu cel din STINGA (**TabelulA**) se completează cu **NULL** în Tabelul din STINGA(**TabelulA**)

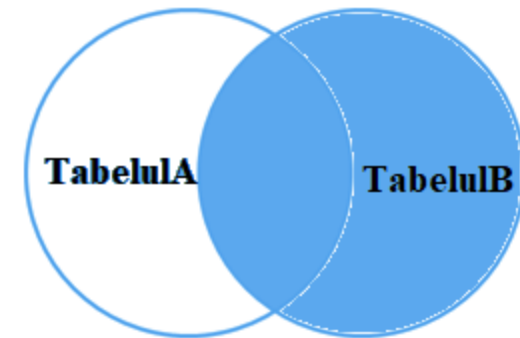
Tabelul A	
id	nume
1	Ion
2	Nata
3	Petru
4	Ghita

Tabelul B	
id	nume
1	Nicolai
2	Ion
3	Elena
4	Petru

id	nume	id	nume
1	Ion	2	Ion
3	Petru	4	Petru
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

+ Options			
id	nume	id	nume
1	Ion	2	Ion
3	Petru	4	Petru
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

+ Options	
id	nume
1	Ion
3	Petru
2	Nata
4	Ghita

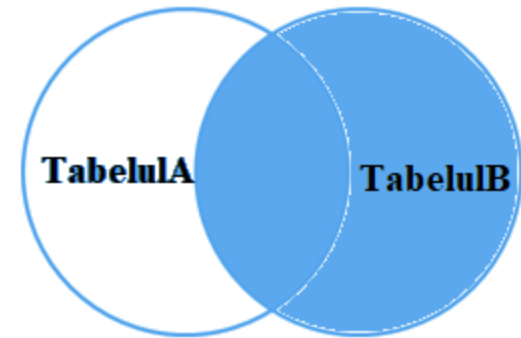


Asocieri, JOIN-uri, RIGHT OUTER

SELECT *
FROM TabelulA
RIGHT OUTER JOIN TabelulB
ON TabelulA.numa=TabelulB.numa
Rezultatul

Tabelul A	
id	numa
1	Ion
2	Nata
3	Petru
4	Ghita

Tabelul B	
id	numa
1	Nicolai
2	Ion
3	Elena
4	Petru



+ Options

id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

Asocieri, JOIN-uri, FULL OUTER/UNION

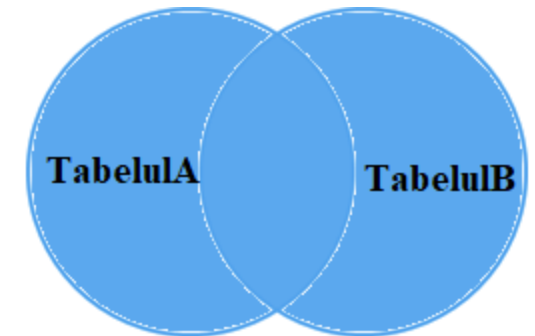
SELECT *
FROM tabelula
LEFT JOIN tabelulb
ON tabelula.numa=tabelulb.numa

UNION

SELECT *
FROM tabelula
RIGHT JOIN tabelulb
ON tabelula.numa=tabelulb.numa

REZULTATUL

Tabelul A		Tabelul B	
id	numa	id	numa
1	Ion	1	Nicolai
2	Nata	2	Ion
3	Petru	3	Elena
4	Ghita	4	Petru



LEFT -- RIGHT

+ Options			
id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
2	Nata	NULL	NULL
4	Ghita	NULL	NULL
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
2	Nata	NULL	NULL
4	Ghita	NULL	NULL
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

Asocieri, JOIN-uri

SELECT *

FROM tabelula

RIGHT JOIN tabelulb

ON tabelula.numa=tabelulb.numa

UNION

SELECT *

FROM tabelula

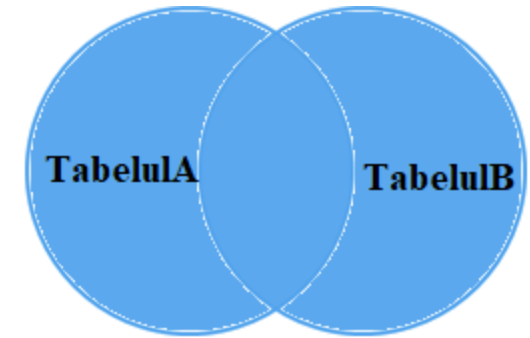
LEFT JOIN tabelulb

ON tabelula.numa=tabelulb.numa

REZULTATUL

Tabelul A	
id	nume
1	Ion
2	Nata
3	Petru
4	Ghita

Tabelul B	
id	nume
1	Nicolai
2	Ion
3	Elena
4	Petru



RIGHT --- LEFT

+ Options			
id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena
2	Nata	NULL	NULL
4	Ghita	NULL	NULL

id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena
2	Nata	NULL	NULL
4	Ghita	NULL	NULL

Asocieri, JOIN-uri cu **WHERE**

Cimpurile din TabelulA ce nu sunt in TabelulB

SELECT *

FROM TabelulA

LEFT JOIN TabelulB

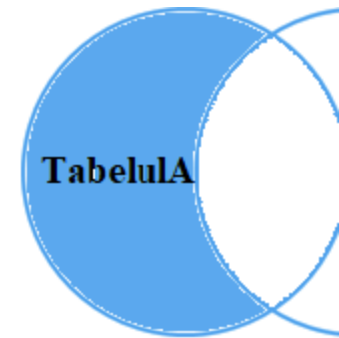
ON TabelulA.numa=TabelulB.numa

WHERE TabelulB.id IS NULL (IS NOT NULL)

REZULTATUL ↕

Tabelul A	
id	numa
1	Ion
2	Nata
3	Petru
4	Ghita

Tabelul B	
id	numa
1	Nicolai
2	Ion
3	Elena
4	Petru



+ Options			
id	numa	id	numa
2	Nata	NULL	NULL
4	Ghita	NULL	NULL

+ Options			
id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
2	Nata	NULL	NULL
4	Ghita	NULL	NULL

id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
2	Nata	NULL	NULL
4	Ghita	NULL	NULL

Asocieri, JOIN-uri cu **WHERE**

Cimpurile din TabelulB ce nu sunt in TabelulA

SELECT *

FROM TabelulA

RIGHT JOIN TabelulB

ON TabelulA.numa=TabelulB.numa

WHERE TabelulA.id IS NULL

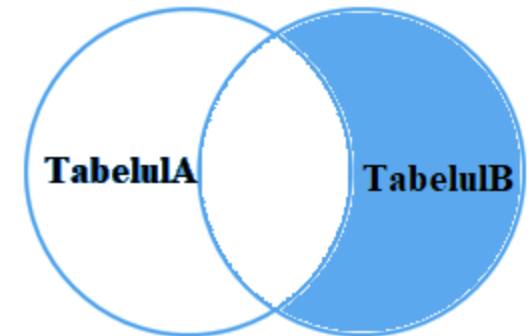
REZULTATUL ↓

+ Options

id	numa	id	numa
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

Tabelul A	
id	numa
1	Ion
2	Nata
3	Petru
4	Ghita

Tabelul B	
id	numa
1	Nicolai
2	Ion
3	Elena
4	Petru



+ Options

id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

SELECT * Asocieri, JOIN-uri, **FULL cu WHERE**

FROM TabelulA

LEFT JOIN TabelulB

ON TabelulA.numa=TabelulB.numa

WHERE TabelulB.id IS NULL

UNION

SELECT *

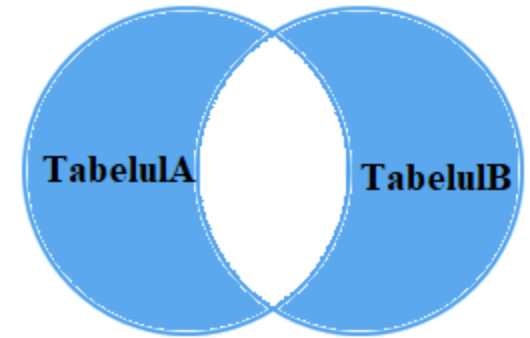
FROM TabelulA

RIGHT JOIN TabelulB

ON TabelulA.numa=TabelulB.numa

WHERE TabelulA.id IS NULL

Tabelul A		Tabelul B	
id	numa	id	numa
1	Ion	1	Nicolai
2	Nata	2	Ion
3	Petru	3	Elena
4	Ghita	4	Petru



+ Options			
id	numa	id	numa
2	Nata	NULL	NULL
4	Ghita	NULL	NULL

+ Options			
id	numa	id	numa
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

+ Options			
id	numa	id	numa
2	Nata	NULL	NULL
4	Ghita	NULL	NULL
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

id	numa	id	numa
1	Ion	2	Ion
3	Petru	4	Petru
2	Nata	NULL	NULL
4	Ghita	NULL	NULL
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

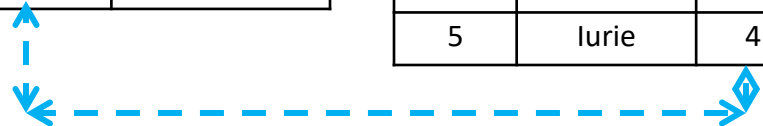
Asocieri, JOIN-uri, TABELE LEGATE

```
CREATE TABLE Departament(  
  id INT NOT NULL AUTO_INCREMENT,  
  nume VARCHAR(30) NOT NULL,  
  PRIMARY KEY (id )  
) ENGINE=INNODB;
```

Departament	
id	nume
1	IT
2	Programare
3	Tehnic
4	Finante

Angajat		
id	nume	d_id
1	Vlad	1
2	Anton	2
3	Alex	6
4	Boris	2
5	Iurie	4

```
CREATE TABLE Angajat(  
  id INT NOT NULL AUTO_INCREMENT,  
  nume VARCHAR(30) NOT NULL,  
  PRIMARY KEY (id ),  
  d_id int,  
  FOREIGN KEY(d_id) REFERENCES Departament(id)  
) ENGINE=INNODB;
```



Asocieri, JOIN-uri, TABELE LEGATE

id	nume
1	IT
2	Programare
3	Tehnic
4	Finante

id	nume	d_id
1	Vlad	1
2	Anton	2
3	Alex	4
4	Boris	2
5	Iurie	4

Departament	
id	nume
1	IT
2	Programare
3	Tehnic
4	Finante

Angajat		
id	nume	d_id
1	Vlad	1
2	Anton	2
3	Alex	6
4	Boris	2
5	Iurie	4



```
INSERT INTO Departament (id, nume)
VALUES (1, 'IT'), (2, 'Programare'), (3, 'Tehnic'), (4, 'Finante');
```

```
ALTER TABLE Angajat AUTO_INCREMENT = 10;
```

```
INSERT INTO Angajat (id, nume, d_id)
VALUES (1, 'Vlad',1), (2, 'Anton',2), (3, 'Alex',4), (4, 'Boris',2), (5,
'Iurie',4);
```

Asocieri, JOIN-uri, TABELE LEGATE

Departament

id	nume
1	IT
2	Programare
3	Tehnic
4	Finante

Angajat

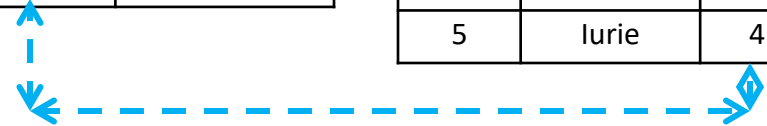
id	nume	d_id
1	Vlad	1
2	Anton	2
3	Alex	4
4	Boris	2
5	Iurie	4

Departament

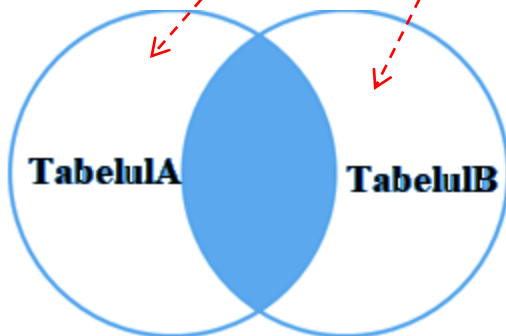
id	nume
1	IT
2	Programare
3	Tehnic
4	Finante

Angajat

id	nume	d_id
1	Vlad	1
2	Anton	2
3	Alex	4
4	Boris	2
5	Iurie	4



SELECT A.id, A.nume, B.nume AS Departament
FROM Angajat A
INNER JOIN Departament B
ON A.d_id=B.id



+ Options

id	nume	Departament
1	Vlad	IT
2	Anton	Programare
3	Alex	Finante
4	Boris	Programare
5	Iurie	Finante

Asocieri, JOIN-uri, TABELE LEGATE

Departament

id	nume
1	IT
2	Programare
3	Tehnic
4	Finante

Angajat

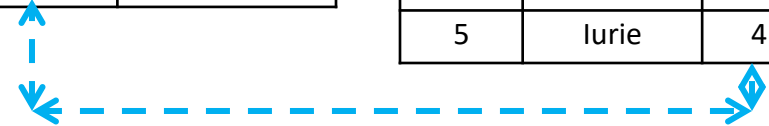
id	nume	d_id
1	Vlad	1
2	Anton	2
3	Alex	4
4	Boris	2
5	Iurie	4

Departament

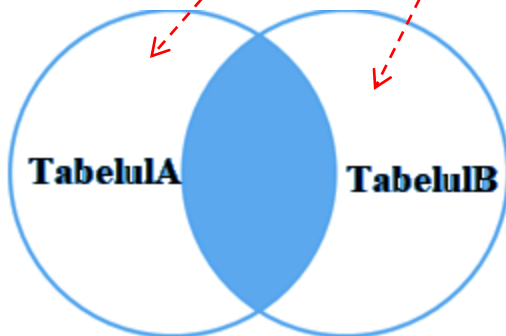
id	nume
1	IT
2	Programare
3	Tehnic
4	Finante

Angajat

id	nume	d_id
1	Vlad	1
2	Anton	2
3	Alex	4
4	Boris	2
5	Iurie	4



SELECT A.id, A.nume, B.nume AS Departament
FROM Departament A
INNER JOIN Angajat B
ON A.id=B.d_id



+ Options

id	nume	Departament
1	IT	Vlad
2	Programare	Anton
4	Finante	Alex
2	Programare	Boris
4	Finante	Iurie

Asocieri, JOIN-uri, TABELE LEGATE

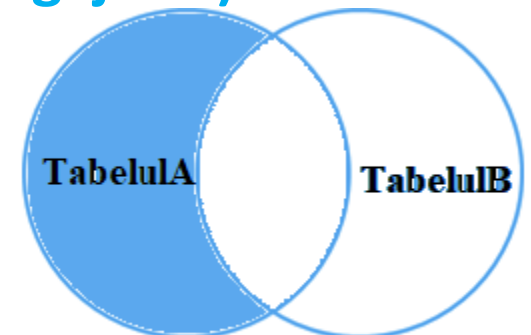
Interogarea SELECT impreuna cu operatiile de asociere JOIN uneste datele intr-o selectie dupa conditia din ON indicata in clauza [optiune] JOIN , unde [optiune] = {INNER, LEFT, RIGHT....} datele din 2 tabele.

Pentru a primi datele ce **NU** coincid dupa condiția ON este necesar de utilizat Asocierea Externă **OUTER JOIN**. **OUTER** – poate fi omis!

Această asociere va returna datele din ambele tabele care vor coincide cu una din condiții. Există 2 tipuri de Asocieri externe **OUTER JOIN – LEFT OUTER JOIN, RIGHT OUTER JOIN**

Ambele lucrează in același mod. Diferența constă in faptul că LEFT arată că Tabelul Extern va fi cel din “STÂNGA” (**Angajat A**)

```
SELECT A.id, A.num, B.num AS Departament
FROM Angajat A
INNER JOIN Departament B
ON A.d_id=B.id
```



Asocieri, JOIN-uri, TABELE LEGATE

Departament

id	nume
1	IT
2	Programare
3	Tehnic
4	Finante

Angajat

id	nume	d_id
1	Vlad	1
2	Anton	2
3	Alex	4
4	Boris	2
5	Iurie	4

Departament

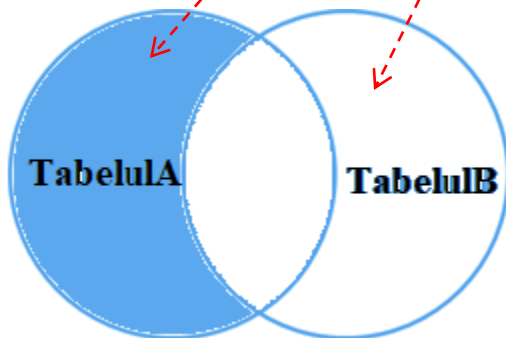
id	nume
1	IT
2	Programare
3	Tehnic
4	Finante

Angajat

id	nume	d_id
1	Vlad	1
2	Anton	2
3	Alex	4
4	Boris	2
5	Iurie	4



SELECT A.id, A.nume, B.nume AS Departament
FROM Angajat A
LEFT JOIN Departament B
ON A.d_id=B.id



+ Options

id	nume	Departament
1	Vlad	IT
2	Anton	Programare
3	Alex	Finante
4	Boris	Programare
5	Iurie	Finante

Asocieri, JOIN-uri, TABELE LEGATE

Departament

id	nume
1	IT
2	Programare
3	Tehnic
4	Finante

Angajat

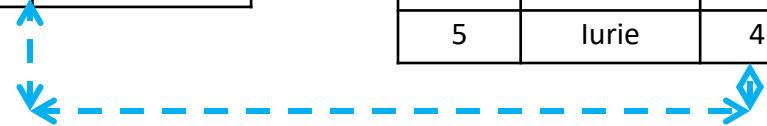
id	nume	d_id
1	Vlad	1
2	Anton	2
3	Alex	4
4	Boris	2
5	Iurie	4

Departament

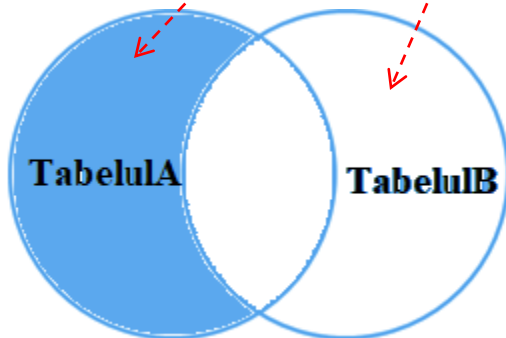
id	nume
1	IT
2	Programare
3	Tehnic
4	Finante

Angajat

id	nume	d_id
1	Vlad	1
2	Anton	2
3	Alex	4
4	Boris	2
5	Iurie	4



SELECT A.id, A.nume AS Departamentul, B.nume AS Angajatul
FROM Departament A
LEFT JOIN Angajat B
ON A.id=B.d_id



+ Options

id	Departamentul	Angajatul
1	IT	Vlad
2	Programare	Anton
4	Finante	Alex
2	Programare	Boris
4	Finante	Iurie
3	Tehnic	NULL

Asocieri, JOIN-uri, TABELE LEGATE, 3 TAB.

USE *bdjoin*;

```
CREATE TABLE Students(  
  sid INT NOT NULL AUTO_INCREMENT,  
  name VARCHAR(30) NOT NULL,  
  email VARCHAR(30),  
  age INT(3),  
  gr VARCHAR (4),  
  PRIMARY KEY (Sid )  
);
```

```
CREATE TABLE Courses(  
  cid varchar(5),  
  cname VARCHAR(30) NOT NULL,  
  credits int,  
  PRIMARY KEY (cid )
```

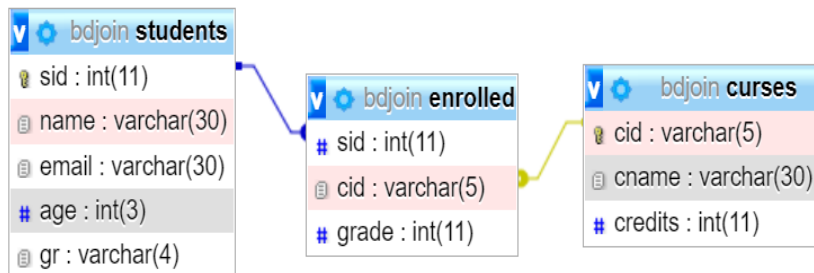
```
CREATE TABLE Enrolled(  
  sid INT,  
  cid varchar(5),  
  grade int,  
  FOREIGN KEY(sid) REFERENCES  
  Students(sid),  
  FOREIGN KEY(cid) REFERENCES Courses(cid)  
);
```

```
INSERT INTO Students (sid, name, email, age, gr)  
VALUES (1234, 'John', 'j@cs.md',21, 331),  
(1235, 'Smith', 's@cs.md',22, 331),  
(1236, 'Anne', 'a@cs.md',21, 331);
```

```
INSERT INTO Courses(cid, cname, credits)  
VALUES ('Alg1', 'Algorithms1',7),  
('DB1', 'Databases1',6), ('DB2', 'Databases2',6);
```

```
INSERT INTO Enrolled(sid, cid, grade)  
VALUES (1234, 'Alg1',9), (1235, 'Alg1',10),  
(1236, 'DB2',9);
```

<i>sid</i>	<i>name</i>
1234	John
1235	Smith
1236	Anne



Asocieri, JOIN-uri, TABELE LEGATE, 3 TAB.

REMARCA!!!

DACA DORIM SA DISTRUGEM/SA MODIFICAM UN TABEL LEGAT DEJA CU CHEI EXTERNE URMEAZA SA EFECTUAM URMATOARELE OPERATII:

1. Adăugarea constrângerilor cheie străine Adding Foreign Key Constraints

```
ALTER TABLE tbl_name ADD [CONSTRAINT [symbol]] FOREIGN KEY [index_name]  
(col_name, ...) REFERENCES tbl_name (col_name,...) [ON DELETE reference_option] [ON  
UPDATE reference_option]
```

EX:

```
ALTER TABLE table_name RENAME TO  
new_table_name;
```

```
ALTER TABLE carti
```

```
ADD CONSTRAINT FK_carti_id_autor
```

```
FOREIGN KEY (id_autor)
```

```
REFERENCES autor
```

```
(id_autor)
```

```
ON DELETE CASCADE [ON UPDATE CASCADE];
```

2. Modificarea tipului coloanei:

```
Alter TABLE `tableName` MODIFY COLUMN `ColumnName` datatype(length);
```

EX:

```
Alter TABLE `tbl_users` MODIFY COLUMN `dup` VARCHAR(120);
```

3. Schimbarea numelui coloanei:

```
ALTER TABLE student_info CHANGE roll_no identity_no VARCHAR(255);
```

Asocieri, JOIN-uri, TABELE LEGATE, 3 TAB.

Students

sid	name	email	age	gr
1234	John	j@cs.md	21	331
1235	Smith	s@cs.md	22	331
1236	Anne	a@cs.md	21	331
1237	Alexa	a@cs.md	21	331

Enrolled

sid	cid	grade
1234	Alg1	9
1235	Alg1	10
1236	DB2	9
1237	DB2	9

Courses

<i>cid</i>	<i>cname</i>	<i>credits</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

INNER JOIN

SELECT S.name, C.cname
FROM Students S, Enrolled E,
Courses C

WHERE S.sid = E.sid AND
E.cid = C.cid



SELECT S.name, C.cname
FROM Students S

INNER JOIN Enrolled E

ON S.sid = E.sid,

INNER JOIN Courses C

ON E.cid = C.cid

Students

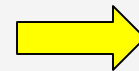
<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Courses

<i>cid</i>	<i>cname</i>	<i>cre</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

Enrolled

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1236	DB2	9
1237	DB2	9



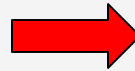
Rezultat

<i>name</i>	<i>cname</i>
John	Algorithms1
Smith	Algorithms1
Anne	Databases2
Alexa	Databases2

LEFT OUTER JOIN

```
INSERT INTO Students (sid, name, email, age, gr)  
VALUES (1237, 'Alexa', 'a@cs.md', 21, 331);
```

Daca dorim sa regasim
și studentii fără nici o
notă la vre-un curs,
atunci utilizăm comanda:



```
SELECT S.name, C.cname  
FROM Students S  
LEFT OUTER JOIN Enrolled E  
ON S.sid = E.sid,  
LEFT OUTER JOIN Courses C  
ON E.cid = C.cid
```

Students

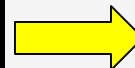
sid	name	email	age	gr
1234	John	j@cs.md	21	331
1235	Smith	s@cs.md	22	331
1236	Anne	a@cs.md	21	331
1237	Alexa	a@cs.md	21	331

Courses

cid	cname	credits
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

Enrolled

sid	cid	grade
1234	Alg1	9
1235	Alg1	10
1236	DB2	9



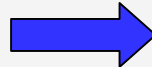
Rezultat

name	cname
John	Algorithms1
Smith	Algorithms1
Anne	Databases2
Alexa	NULL

```
DELETE FROM Enrolled where sid=1237
```

RIGHT OUTER JOIN

Pentru a gasi notele
ofeite unor studenti
inexistenti urmează să
utilizăm comanda:



```
SELECT S.name, C.cname  
FROM Students S  
INNER JOIN Enrolled E  
ON E.Sid = S.Sid  
RIGHT OUTER JOIN Courses C  
ON C.Cid = E.Cid
```

Students

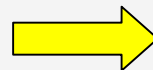
sid	name	email	age	gr
1234	John	j@cs.md	21	331
1235	Smith	s@cs.md	22	331
1236	Anne	a@cs.md	21	331
1237	Alexa	a@cs.md	21	331

Courses

<i>cid</i>	<i>cname</i>
Alg1	Algorithms1
DB1	Databases1
DB2	Databases2

Enrolled

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1236	DB2	9



Rezultat

name	cname
John	Algorithms1
Smith	Algorithms1
NULL	Databases1
Anne	Databases2

FULL OUTER JOIN

LEFT+RIGHT OUTER

JOIN In majoritatea

SGBD OUTER e optional
Students

sid	name	email	age	gr
1234	John	j@cs.md	21	331
1235	Smith	s@cs.md	22	331
1236	Anne	a@cs.md	21	331
1237	Alexa	a@cs.md	21	331

Enrolled

sid	cid	grade
1234	Alg1	9
1235	Alg1	10
1236	DB2	9

Courses

cid	cname	credits
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

```
SELECT S.name, C.cname
FROM Students S
LEFT JOIN Enrolled E
ON S.sid = E.sid
LEFT JOIN Courses C
ON E.cid = C.cid
WHERE E.sid IS NULL
UNION
```

```
SELECT S.name, C.cname
FROM Enrolled E
RIGHT JOIN Students S
ON S.sid = E.sid
RIGHT JOIN Courses C
ON C.cid = E.cid
WHERE S.sid IS NULL
```

Rezultat

name	cname
Alexa	NULL
NULL	Databases1

```
SELECT S.name,
FROM Students S
FULL OUTER JOIN
ON S.sid = E.sid,
FULL OUTER JOIN
ON E.cid = C.cid
```

SARCINI

Să se determine departamentele care nu au angajați!

```
SELECT A.id, A.num AS Departamentul, B.d_id, B.num AS  
Angajatul  
FROM Departament A  
LEFT JOIN Angajat B  
ON A.id=B.d_id  
Where B.d_id IS NULL
```

✓ Showing rows 0 - 0 (1 total, Query took 0.0047 seconds.)

```
SELECT A.id, A.num AS Departamentul, B.d_id, B.num AS Angajatul FROM departament A LEFT JOIN angajat B ON A.id=B.d_id Where  
B.d_id IS NULL
```

☐ Profiling [\[Edit inline\]](#) [\[Edit\]](#) [\[Explain SQL\]](#) [\[Create PHP code\]](#) [\[Refresh\]](#)

☐ Show all | Number of rows: 25 ▾ Filter rows:

+ Options

id	Departamentul	d_id	Angajatul
3	Tehnic	NULL	NULL

SARCINI

Să se determine departamentele care nu au angajați!

```
SELECT A.id, A.num, B.num AS Departament  
FROM Angajat A  
RIGHT JOIN Departament B  
ON A.d_id=B.id  
WHERE A.Id IS NULL
```

✓ Showing rows 0 - 0 (1 total, Query took 0.0044 seconds.)

```
SELECT A.id, A.num, B.num AS Departament FROM Angajat A RIGHT JOIN Departament B ON A.d_id=B.id WHERE A.Id IS NULL
```

☐ Profiling [\[Edit inline\]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP \]](#)

☐ Show all | Number of rows: 25 Filter rows:

+ Options

id	num	Departament
NULL	NULL	Tehnic

SARCINI

Să se determine persoanele care sunt in TabelulA si nu sunt in TabelulB

SELECT *

FROM TabelulA

LEFT JOIN TabelulB

ON TabelulA.numa=TabelulB.numa

WHERE TabelulB.id IS NULL

✓ Showing rows 0 - 1 (2 total, Query took 0.0033 seconds.)

```
SELECT * FROM TabelulA LEFT JOIN TabelulB ON TabelulA.numa=TabelulB.numa WHERE TabelulB.id IS NULL
```

☐ Profiling [\[Edit inline\]](#) [\[Edit \]](#) [\[Explain](#)

☐ Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

id	numa	id	numa
2	Nata	NULL	NULL
4	Ghita	NULL	NULL

SARCINI

Să se determine persoanele care sunt in TabelulB si nu sunt in TabelulA

SELECT *

FROM TabelulA

RIGHT JOIN TabelulB

ON TabelulA.numa=TabelulB.numa

WHERE TabelulA.id IS NULL

✓ Showing rows 0 - 1 (2 total, Query took 0.0053 seconds.)

```
SELECT * FROM TabelulA RIGHT JOIN TabelulB ON TabelulA.numa=TabelulB.numa WHERE TabelulA.id IS NULL
```

☐ Profiling [Edit inline] [Edit] [Explain]

☐ Show all | Number of rows: 25 ▾ Filter rows: Search this table Sort by key: None

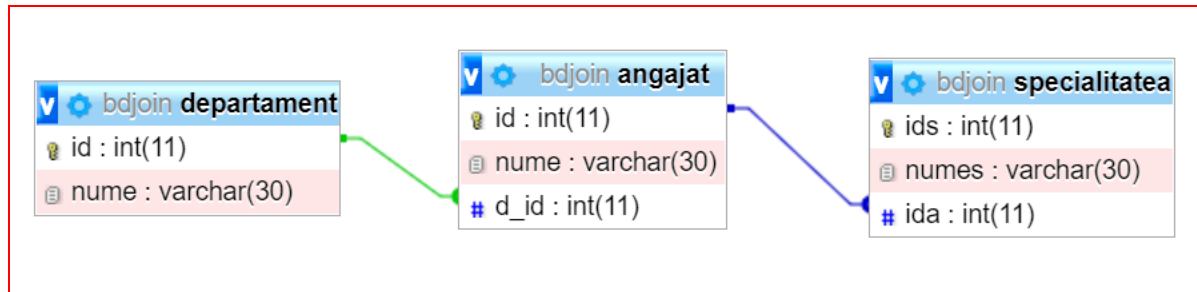
+ Options

id	numa	id	numa
NULL	NULL	1	Nicolai
NULL	NULL	3	Elena

SARCINI

```
CREATE TABLE Specialitatea(  
    ids INT NOT NULL AUTO_INCREMENT,  
    numes VARCHAR(30) NOT NULL,  
    PRIMARY KEY (ids )  
    ida INT,  
    FOREIGN KEY (ida) REFERENCES Angajat(id) )  
ENGINE=INNODB;
```

```
INSERT INTO Specialitatea (ids,numes,ida) VALUES  
(1,'Ing.progr.',1),(2,'Administrator',1),(3,'Contabil',5),(4,'Inginer  
cat.I',2), (5,'Proj.Mang',3), (6,'Tehnic',4);
```

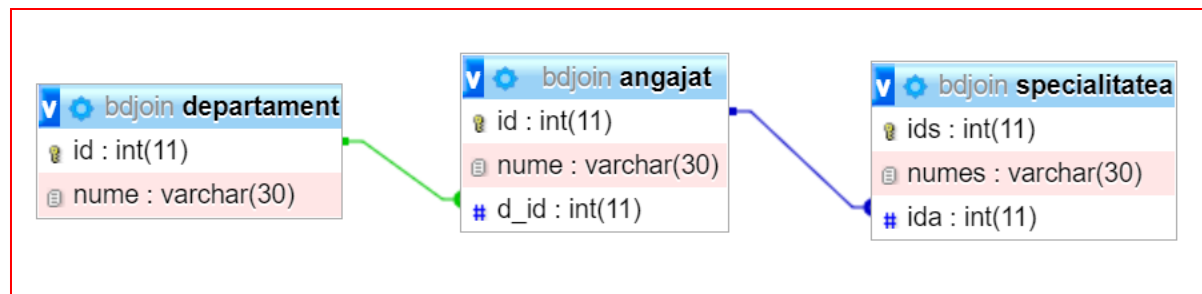


SARCINI

De prezentat toate **Departamentele** și **Specialitățile** care se regăsesc în aceste **Departamente**. Pentru fiecare **Departament** de afișat numele lui, numele **Angajatului**, precum și Specialitatea pe care o are. De prezentat chiar și **Departamentele** fără angajați cu **Specialități**.

```
SELECT A.id, A.num AS Departament, B.num AS Angajat,  
C.num AS Specialitatea  
FROM Departament A  
INNER JOIN Angajat B  
ON B.d_id=A.id  
INNER JOIN Specialitatea C  
ON C.id=B.id  
ORDER BY A.id
```

+ Options				
id	1	Departament	Angajat	Specialitatea
1	IT	Vlad	Ing.progr.	
2	Programare	Boris	Tehnic	
2	Programare	Anton	Inginer cat.I	
2	Programare	Anton	Administrator	
4	Finante	Alex	Proj.Mang	
4	Finante	Iurie	Contabil	



SARCINI

Deși ordinea JOIN-urilor în **INNER JOIN** nu este importantă, nu se poate spune același lucru și pentru **LEFT JOIN**. Când folosim **LEFT JOIN** pentru a asocia mai multe tabele, este important să ne amintim că această îmbinare va include toate rândurile din tabelul din partea STÂNGA a JOIN. Să rearanjăm interogarea anterioară, pentru a soluționa următoarea sarcină: ***prezentati numele tuturor Specialitatilor, Angajatii, precum și Departamentele în care sunt prezente aceste Specialități. Returnați chiar și Specialitățile fără angajați și Departamente.***

```
SELECT A.ids,A.numes AS Specialitatea,B.num AS Angajat,C.num  
AS Depatament  
FROM Specialitatea A  
LEFT JOIN Angajat B  
ON A.ida=B.id  
LEFT JOIN Departament C  
ON B.d_id =C.id  
ORDER BY A.ids
```

11/22/2021

Cum funcționează LEFT JOIN? Acesta ia primul tabel (Specialitatea) și unește toate rândurile sale (6 dintre ele) cu următorul tabel (Angajat) (5 inscrieri). Rezultatul este 6 rânduri, deoarece mai multe Specialitati pot aparține unui Angajat. Apoi unim aceste 6 rânduri la următorul tabel (Departament), și din nou avem 6 rânduri pentru că Angajatul ar putea aparține doar unui Departament

REMARCĂ

Folosind **INNER JOIN** de prezentat lista **autorilor** si **cartile** pe care le-au scris

```
SELECT A.nume_autor, B.titlu  
From Autor A  
INNER JOIN Carti B  
On A.id_autor=B.id_autor
```

```
SELECT A.nume_autor, B.titlu  
From Autor A  
LEFT JOIN Carti B  
On A.id_autor=B.id_autor
```

[Where B.titlu is null] /autorii care nu au carti/



+ Options	
nume_autor	titlu
Ghita	NULL
Nicu	NULL

```
SELECT A.nume_autor, B.titlu  
From Autor A  
RIGHT JOIN Carti B  
On A.id_autor=B.id_autor
```

REMARCĂ

Folosind **INNER JOIN** de prezentat lista autorilor si cantitatea de cărți pe care le-au scris

SELECT A.nume_autor, SUM(B.Cantitatea) AS Cantitatea
From Autor A

INNER JOIN Carti B

On A.id_autor=B.id_autor
GROUP BY NUME_AUTOR
ORDER BY A.nume_autor
(0.0074)

+ Options	
nume_autor	Cantitatea
Diuma	215
Eminescu	355
Ion	55
Petru	209
Vieru	20

id_autor	nume_autor
1	Petru
2	Diuma
3	Vieru
4	Lena
5	Ion
6	Ghita
7	Eminescu
8	Nicu
9	Valea

idcarte	autor	data_n	titlu	pret	cantita
"2-2222-222-10"	Petru	2021-11-18	Ghid PHP	200.00	
"2-2222-222-13"	Vieru	0000-00-00	Guguta	200.00	
"2-2222-2222-1"	Diuma	2021-11-18	Trei muschetari	200.00	
"2-2222-2222-12"	Petru	2021-11-18	Programarea OO	200.00	
"2-2222-2222-2"	Diuma	2021-11-18	Trei muschetari	200.00	
"2-2222-2222-22"	Eminescu	2021-11-18	Luceafarul	200.00	
"2-2222-2222-23"	Eminescu	2021-11-18	Luceafarul	200.00	
"2-2222-2222-4"	Petru	2021-11-18	Ghid Mysql	200.00	
"2-2222-2222-5"	Vica	0000-00-00	In lumea copiilor	200.00	
"2-2222-2222-6"	Valea	0000-00-00	"930.00"	200.00	
"2-2222-2222-7"	Lena	0000-00-00	"465.00"	200.00	
"2-2222-2222-8"	Eminescu	2021-11-18	Filosofia lui Heghel	200.00	
"2-2222-2222-9"	Eminescu	2021-11-18	Ion Creanga	200.00	
2-2222-2222-01	Petru	2021-11-18	Web Design	200.00	
2-2222-2222-25	Diuma	2020-10-17	SAPD - proiectare	200.00	

REMARCĂ

Folosind **LEFT JOIN** de prezentat lista autorilor si cantitatea de cărți pe care le-au scris, INCLUSIV SI A CELOR CE NU AU SCRIS CARTI

SELECT A.nume_autor, SUM(B.Cantitatea) AS Cantitatea

From Autor A

LEFT JOIN Carti B

On A.id_autor=B.id_autor

GROUP BY NUME_AUTOR

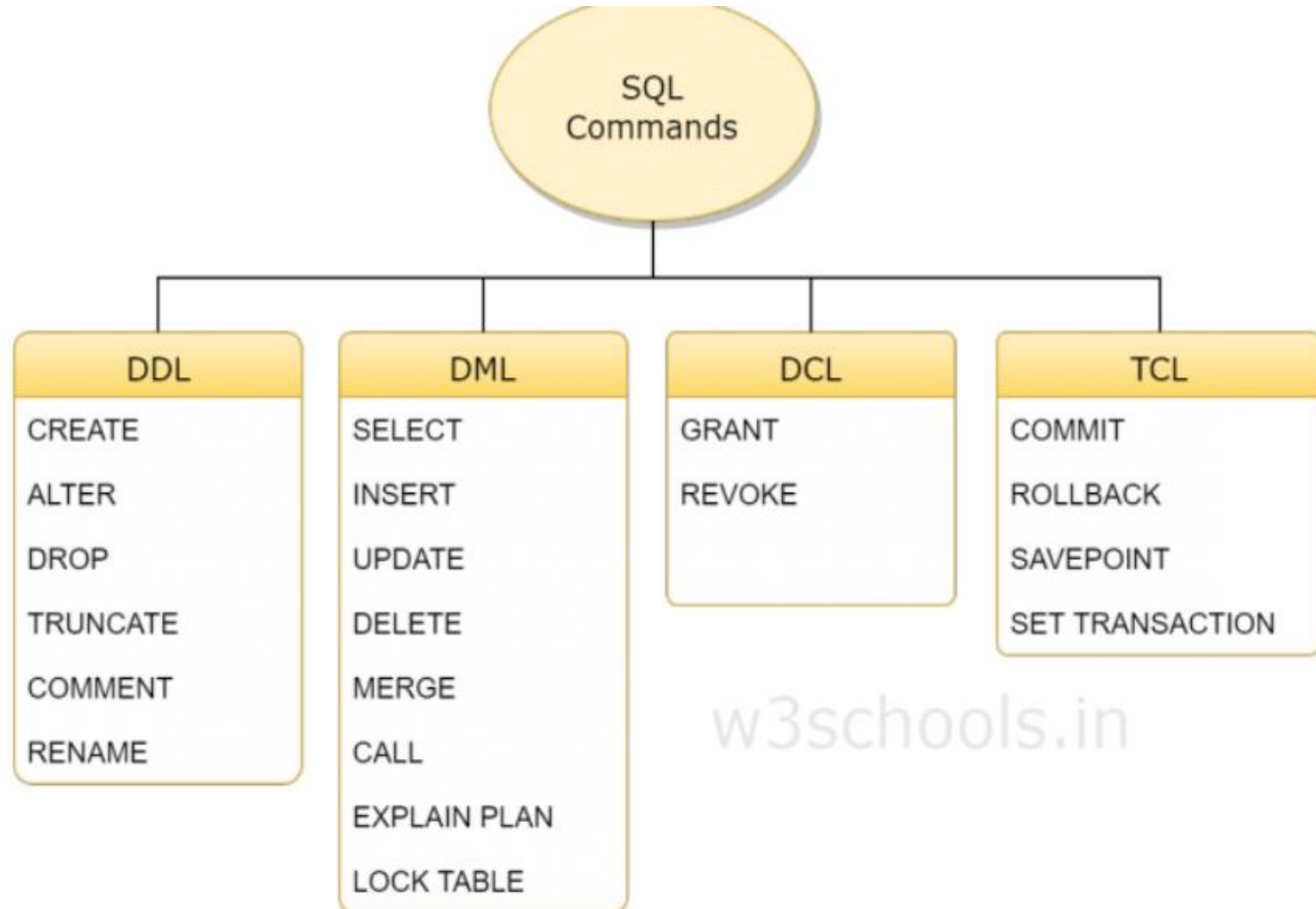
**ORDER BY A.nume_autor
(0.0049)**

+ Options		
nume_autor	▲ 1	Cantitatea
Diuma		215
Eminescu		355
Ghita		NULL
Ion		55
Lena		NULL
Nicu		NULL
Petru		209
Valea		NULL
Vieru		20

id_autor	nume_autor
1	Petru
2	Diuma
3	Vieru
4	Lena
5	Ion
6	Ghita
7	Eminescu
8	Nicu
9	Valea

idcarte	autor	data_n	titlu	pret	cantitate
"2-2222-222-10"	"Petru"	2021-11-18	Ghid PHP	200.00	10
"2-2222-222-13"	"Vieru"	0000-00-00	Guguta	200.00	2
"2-2222-2222-1"	"Diuma"	2021-11-18	Trei muschetari	200.00	2
"2-2222-2222-12"	"Petru"	2021-11-18	Programarea OO	200.00	1
"2-2222-2222-2"	"Diuma"	2021-11-18	Trei muschetari	200.00	15
"2-2222-2222-22"	"Eminescu"	2021-11-18	Luceafarul	200.00	13
"2-2222-2222-23"	"Eminescu"	2021-11-18	Luceafarul	200.00	3
"2-2222-2222-4"	"Petru"	2021-11-18	Ghid Mysql	200.00	4
"2-2222-2222-5"	"Vica"	0000-00-00	In lumea copiilor	200.00	5
"2-2222-2222-6"	"Vica"	0000-00-00	"930.00"	200.00	1
"2-2222-2222-7"	"Sandu"	0000-00-00	"465.00"	200.00	6
"2-2222-2222-8"	"Eminescu"	2021-11-18	Filosofia lui Heghel	200.00	4
"2-2222-2222-9"	"Eminescu"	2021-11-18	Ion Creanga	200.00	7
2-2222-2222-01	Petru	2021-11-18	Web Design	200.00	4
2-2222-2222-25	Fusu	2020-10-17	SAPD - proiectare	200.00	3

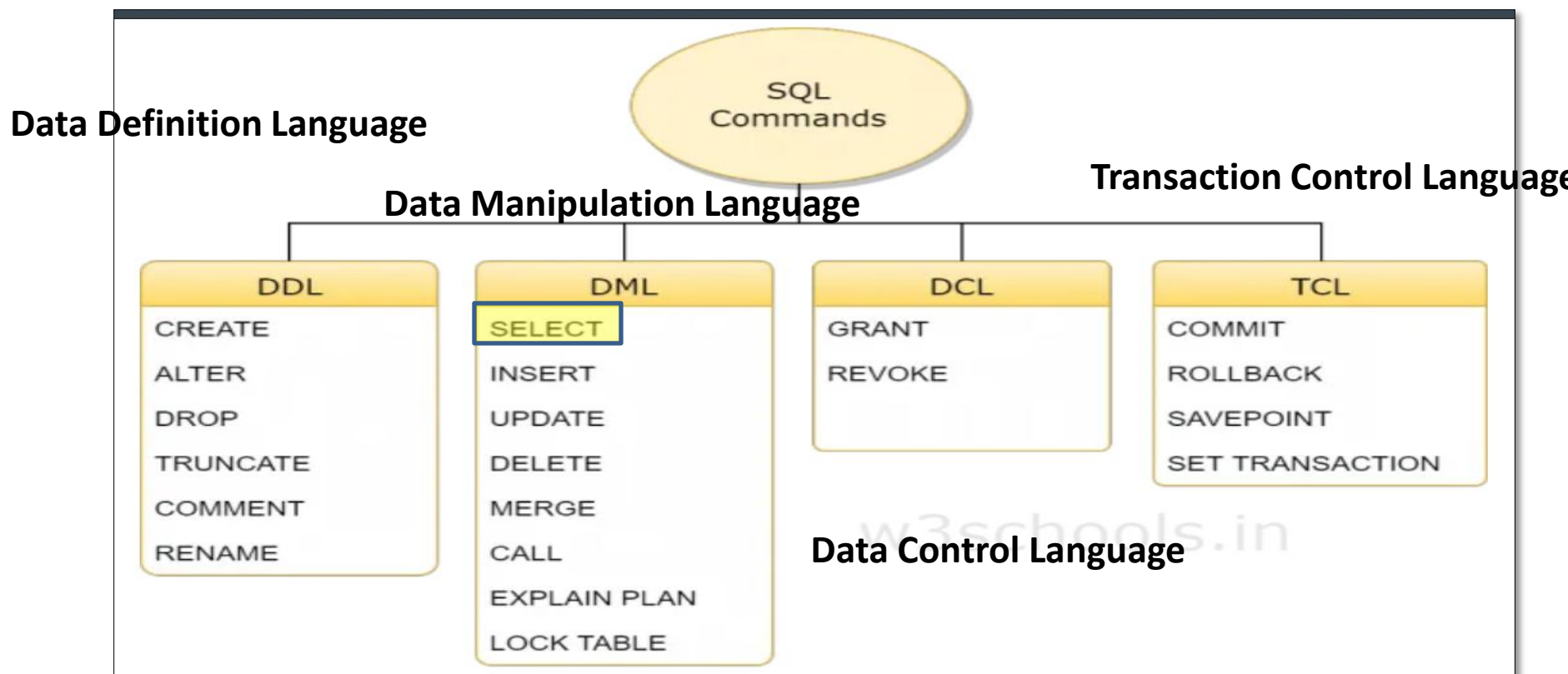
Limbajul SQL



Limbajul SQL

SQL utilizează o sintaxă foarte simplă și ușor de folosit.

Comenzile **SQL** sunt grupate în cinci categorii, astfel:





Limbajul SQL



1. **Limbajul de interogare** permite regăsirea liniilor memorate în tabelele bazelor de date.

Comanda utilizată este **SELECT**.

SELECT [**DISTINCT** | **ALL**] { * | [fieldExpression [**AS** newName]] }

FROM tableName [alias]

[**WHERE** condition]

[**GROUP BY** fieldName(s)]

[**HAVING** condition]

ORDER BY fieldName(s)



Limbajul SQL

2. Limbajul de manipulare a datelor (LMD) permite modificarea conținutului tabelelor.

Comenzile utilizate sunt:

- 1. INSERT** – pentru adăugarea de noi linii într-o tabelă
- 2. UPDATE** – pentru modificarea valorilor memorate într-o tabelă
- 3. DELETE** – pentru ștergerea liniilor dintr-o tabelă



Limbaajul SQL

3. Limbaajul de definire a datelor (LDD) permite definirea structurii tabelor ce compun bazele de date.

Comenzile utilizate sunt:

1. **CREATE** – pentru crearea structurii unei baze de date sau a unei tabele
2. **ALTER** – pentru modificarea structurii unei baze de date sau a unei tabele
3. **DROP** – pentru ștergerea structurii bazei de date
4. **RENAME** – schimbarea numelui unei tabele
5. **TRUNCATE** – ștergerea conținutului unei tabele