Baze de date MySQL

Marin Bînzari
Pentalog

Cuprins

Ce este o bază de date ?

Baze de date

O bază de date este o colecţie de date centralizate, creată şi menţinută computerizat, în scopul prelucrării datelor în contextul unui set de aplicaţii. Prelucrarea datelor se referă la operaţiile de introducere, ştergere, actualizare şi interogare a datelor.

Ce oferă o bază de date :

1. Control centralizat al datelor
2. Viteză mare
3. Sunt compacte
4. Flexibilitate
5. Integritate

Tipuri de baze de date

Tipuri de baze de date :

1. Relaționale (MySQL, PostgreSQL)
2. Obiect orientate (MongoDB)
3. Graph (Neo4j, ArangoDB)
4. Key-Value (Redis, MemcacheDB)

Instalare MySQL

```
→ sudo apt-get update
→ sudo apt-get install mysql-server

→ mysql_secure_installation
```

Proiectarea bazelor de date

Definiții

- Tabelă (entitate) este o colecţie de informaţii logice relaţionale tratată ca o unitate
- Înregistrare (n-uplu). O tabelă este compusă din înregistrări sau rânduri. Fiecare înregistrare este tratată ca o simplă unitate. Fiecare înregistrare este legată de înregistrări ale altei tabele.
- Câmpuri (atribute). Înregistrările sunt constituite din câmpuri (coloane) . Un câmp este o particulă atomică a bazei de date ce reprezintă cea mai mică cantitate de informaţie care poate fi manipulată. Toate înregistrările dintr-o tabelă au aceleaşi câmpuri.

Normalizarea

Date ne-normalizate

Următorul tabel conţine date ne-normalizate. Coloanele 2, 3 şi 4 conţin liste de valori, iar coloana 5 conţine un atribut compus.

	Listă	*Listă*	*Listă*	*Valori compuse*				
Legitimaţie	Cod Calificări	Categorie Calificări	Profil	Nume	Vârstă	Birou nr.	Oraş	Superior
21	113	Sisteme	3	Mareş Ana	29	1	Iaşi	Ion
35	113 170 200	Sisteme Taxe Audit	5 7 4	Ionescu Dan	33	2	Bucureşti	Damian
50	170	Taxe	3	Mircea Călin	35	2	Bucureşti	Damian
77	150 200	Consultanţă Audit	5 8	Traian Raluca	28	1	Iaşi	Ion

Prima formă normală (FN1)

O relaţie se prezintă în FN1 dacă valorile fiecărui câmp sunt ne-decompozabile (atomice) şi fiecare tuplu este unic. Atributele care se pot partiţiona în sub-atribute sau grupurile repetitive, care sunt foarte comune în bazele de date, nu sunt permise.

Legitimaţie	Cod Calificări	Categorie Calificări	Profil	Nume	Prenume	Vârstă	Birou nr.	Oraş	Superior
21	113	Sisteme	3	Mareş	Ana	29	1	Iaşi	Ion
35	113	Sisteme	5	Ionescu	Dan	33	2	Bucureşti	Damian
35	170	Taxe	7	Ionescu	Dan	33	2	Bucureşti	Damian
35	200	Audit	4	Ionescu	Dan	33	2	Bucureşti	Damian
50	170	Taxe	3	Mircea	Călin	35	2	Bucureşti	Damian
77	150	Consultanţa	5	Traian	Raluca	28	1	Iaşi	Ion
77	200	Audit	8	Traian	Raluca	28	1	Iaşi	Ion

A doua formă normală (FN2)

O relaţie este în FN2 dacă este în FN1 şi toate atributele sale sunt dependente de întreaga cheie (adică, nici unul din atributele non-cheie nu este relaţionat doar cu o parte a cheii). Tehnica de descompunere pentru obţinerea FN2 este foarte simplă: presupune construirea unei relaţii separate care sa inglobeze dependenţele parţiale şi să înlăture atributele dependente din relaţia originală.

Relaţia **CADRE**:

Legitimaţie	Nume	Prenume	Vârstă	Birou nr.	Oraş	Superior
21	Mareş	Ana	29	1	Iaşi	Ion
35	Ionescu	Dan	33	2	Bucureşti	Damian
50	Mircea	Călin	35	2	Bucureşti	Damian
77	Traian	Raluca	28	1	Iaşi	Ion

A doua formă normală (FN2)

Relaţia **CALIFICĂRI**:

Cod Calificări	Categorie Calificări
113	Sisteme
170	Taxe
200	Audit
150	Consultanţa

Relaţia **COMPETENŢE**:

Legitimatie	Cod Calificări	Competenţe
21	113	3
35	113	5
35	170	7
35	200	4
50	170	3
77	150	5
77	170	8

A treia formă normală (FN3)

O relaţie este în FN3 dacă este în FN2 şi nu există nici un fel de dependenţe tranzitive (adică, nici unul din atributele non-cheie nu este dependent de alt atribut, care la rândul său este dependent de cheia relaţiei).

Relaţia **CADRE**:

Legitimaţie	Nume	Prenume	Vârstă	Birou nr.
21	Mareş	Ana	29	1
35	Ionescu	Dan	33	2
50	Mircea	Călin	35	2
77	Traian	Raluca	28	1

Relaţia **BIROURI**:

Birou nr.	Oraş	Superior
1	Iaşi	Ion
2	Bucureşti	Damian

A treia formă normală (FN3)

Relaţia **CALIFICĂRI**:

Cod Calificări	Categorie Calificări
113	Sisteme
170	Taxe
200	Audit
150	Consultanţa

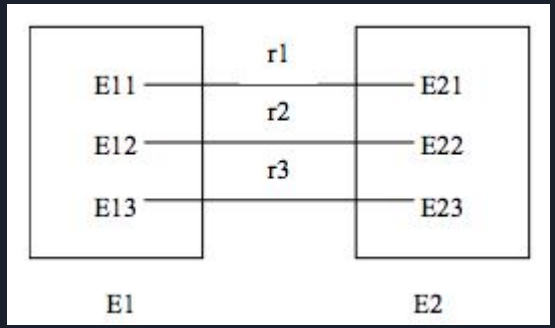
Relaţia **COMPETENŢE**:

Legitimaţie	Cod Calificări	Competenţe
21	113	3
35	113	5
35	170	7
35	200	4
50	170	3
77	150	5
77	170	8

Relațiile : 1-1, 1-N, M-N, unară

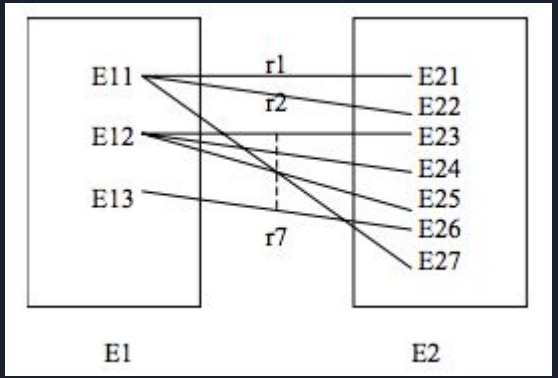
Relaţia „unul-la-unul” (1-1 sau one to one)

Relaţia „unul-la-unul” este cel mai simplu tip de relaţie. Ea este relaţia prin care unui element din mulţimea E1 îi corespunde un singur element din mulţimea E2 şi reciproc.



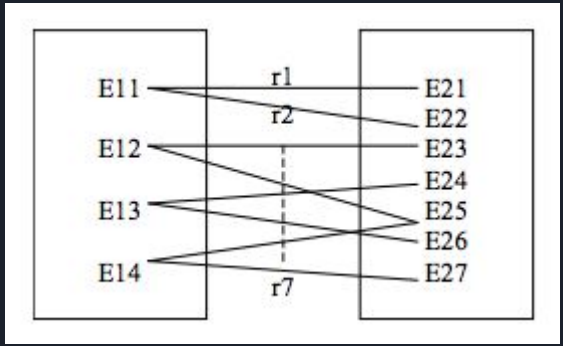
Relaţia „unul-la-multe” (1-N sau one to many)

Această relaţie este o relaţie prin care unui element din mulţimea E1 îi corespund unul sau mai multe elemente din mulţimea E2, dar unui element din mulţimea E2 îi corespunde un singur element din mulţimea E1.



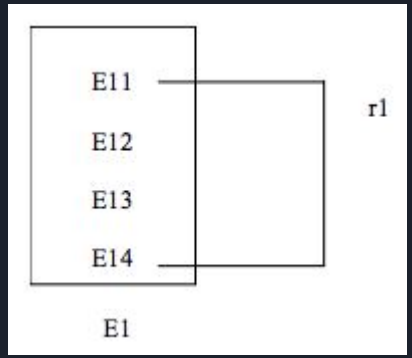
Relaţia „multe-la-multe” (M-N sau many to many)

Această relaţie este o relaţie prin care unui element din mulţimea E1 îi corespund unul sau mai multe elemente din mulţimea E2, şi reciproc.



Relaţia unară (self-referencing)

Toate relaţiile prezentate anterior sunt relaţii binare, având câte două relaţii implicate. Relaţiile unare folosesc doar o singură relaţie, aceasta fiind asociată cu ea însăşi.



Tipuri de date

Numere

Type	Length in Bytes	Minimum Value (Signed)	Maximum Value (Signed)	Minimum Value (Unsigned)	Maximum Value (Unsigned)
TINYINT	1	-128	127	0	255
SMALLINT	2	-32768	32767	0	65535
MEDIUMINT	3	-8388608	8388607 to	0	16777215
INT	4	-2147483648	2147483647	0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807	0	18446744073709551615
FLOAT	4	-3.402823466E+38	-1.175494351E-38	1.175494351E-38	3.402823466E+38
DOUBLE	8	-1.7976931348623 157E+ 308	-2.22507385850720 14E- 308	0, and 2.22507385850720 14E- 308	1.797693134862315 7E+ 308

Date

Types	Description	Display Format	Range
DATETIME	Use when you need values containing both date and time information.	YYYY-MM-DD HH:MM:SS	'1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
DATE	Use when you need only date information.	YYYY-MM-DD	'1000-01-01' to '9999-12-31'.
TIMESTAMP	Values are converted from the current time zone to UTC while storing and converted back from UTC to the current time zone when retrieved.	YYYY-MM-DD HH:MM:SS	'1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC

String

Types	Description	Range in characters
CHAR	Contains non-binary strings. Length is fixed as you declare while creating a table. When stored, they are right-padded with spaces to the specified length.	The length can be any value from 0 to 255.
VARCHAR	Contains non-binary strings. Columns are variable-length strings.	A value from 0 to 255 before MySQL 5.0.3, and 0 to 65,535 in 5.0.3 and later versions.
BINARY	Contains binary strings.	0 to 255
VARBINARY	Contains binary strings.	A value from 0 to 255 before MySQL 5.0.3, and 0 to 65,535 in 5.0.3 and later versions.

Text

Types	Description	Categories	Range in characters
BLOB	Large binary object that containing a variable amount of data. Values are treated as binary strings.You don't need to specify length while creating a column.	TINYBLOB	255 characters
		BLOB	65535 characters
		MEDIUMBLOB	16777215 characters
		LONGBLOB	4294967295 characters
TEXT	Values are treated as character strings having a character set.	TINYTEXT	255 characters (255 B)
		TEXT	65535 characters (64 KB)
		MEDIUMTEXT	16777215 characters (16 MB)
		LONGTEXT	4294967295 characters (4 GB)

ENUM

CREATE TABLE length (length ENUM('small', 'medium', 'large'));

Max 64 choices.

Index & Constrîngeri

Index

CREATE INDEX [index name] ON [table name]([column name]);

Constrîngeri de integritate

https://www.w3resource.com/mysql/creating-table-advance/constraint.php

CREATE TABLE [table name] ([column name] [data type]([size]) [column constraint] [table constraint] ([[column name]……])……);

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

NULL CONSTRAINT

```
CREATE TABLE IF NOT EXISTS author
(
id int(11) NOT NULL,
name varchar(50) NOT NULL,
email varchar(255) NOT NULL,
country varchar(25) NOT NULL,
home_city varchar(25) NOT NULL
);
```

UNIQUE

```
CREATE TABLE IF NOT EXISTS author
(
id int(11) NOT NULL UNIQUE,
name varchar(50) NOT NULL,
email varchar(255) NOT NULL,
country varchar(25) NOT NULL,
home_city varchar(25) NOT NULL,
UNIQUE (email)
);
```

PRIMARY KEY

```
CREATE TABLE IF NOT EXISTS author
(
id int(11) NOT NULL AUTO_INCREMENT,
name varchar(50) NOT NULL,
email varchar(255) NOT NULL,
country varchar(25) NOT NULL,
home_city varchar(25) NOT NULL,
PRIMARY KEY (id),
UNIQUE (email)
);
```

FOREIGN KEY

FOREIGN KEY [column list] REFERENCES [primary key table] ([column list]);

```
CREATE TABLE IF NOT EXISTS book
(
id int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
name varchar(50) NOT NULL,
isbn_no varchar(15) NOT NULL UNIQUE,
author_id int(11) NOT NULL,
publication_date date,
number_of_pages int(5) NOT NULL,
FOREIGN KEY (author_id) REFERENCES author(id)
);
```

Check

```
CREATE TABLE IF NOT EXISTS book
(
id int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
name varchar(50) NOT NULL,
isbn_no varchar(15) NOT NULL UNIQUE,
author_id int(11) NOT NULL,
publication_date date,
number_of_pages int(5) NOT NULL,
FOREIGN KEY (author_id) REFERENCES author(id),
CHECK(number_of_pages>0)
);
```

DEFAULT

```
CREATE TABLE IF NOT EXISTS author
(
id int(11) NOT NULL AUTO_INCREMENT,
name varchar(50) NOT NULL,
email varchar(255) NOT NULL,
country varchar(25) NOT NULL DEFAULT 'Moldova',
home_city varchar(25) NOT NULL DEFAULT 'Chisinau',
PRIMARY KEY (id),
UNIQUE (email)
);
```

FOREIGN KEY - CASCADE & RESTRICT

NO ACTION === RESTRICT

```
CREATE TABLE IF NOT EXISTS book
(
id int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
name varchar(50) NOT NULL,
isbn_no varchar(15) NOT NULL UNIQUE,
author_id int(11) NOT NULL,
publication_date date,
number_of_pages int(5) NOT NULL,
FOREIGN KEY (author_id) REFERENCES author(id) ON UPDATE CASCADE ON
DELETE RESTRICT,
CHECK(number_of_pages>0)
);
```

FOREIGN KEY - SET NULL

```
CREATE TABLE IF NOT EXISTS book
(
id int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
name varchar(50) NOT NULL,
isbn_no varchar(15) NOT NULL UNIQUE,
author_id int(11) NOT NULL,
publication_date date,
number_of_pages int(5) NOT NULL,
FOREIGN KEY (author_id) REFERENCES author(id) ON UPDATE CASCADE ON
DELETE SET NULL,
CHECK(number_of_pages>0)
);
```

Selectarea, inserarea, ștergerea, modificarea

Database SQL

```
mysql> SHOW DATABASES;
mysql> CREATE DATABASE web_academy;
mysql> USE web_academy;
mysql> SHOW TABLES;
mysql> DROP DATABASE web_academy;
```

INSERT

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

EX :

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');
```

SELECT

```
SELECT column1, column2, ...
FROM table_name;
```

EX :

```
SELECT * FROM Customers; # All fields
```

```
SELECT CustomerName, City FROM Customers;
```

WHERE

SELECT column1, column2, … FROM table_name
WHERE condition;

Ex:

SELECT * FROM Customers WHERE CustomerID=1;

SELECT * FROM Customers WHERE Country='Moldova';

UPDATE

```
UPDATE table_name
SET column1 = value1, column2 = value2, …
WHERE condition;
```

Ex:

```
UPDATE Customers
SET ContactName = ‘Ion Druță’, City= ‘Horodiște’
WHERE CustomerID = 1;
```

DELETE

```
DELETE FROM table_name
WHERE condition;
```

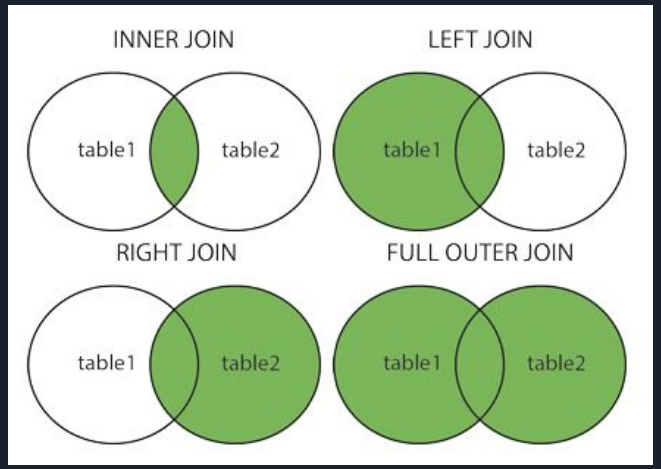
Ex:

```
DELETE FROM Customers

WHERE CustomerID = 1;
```

Join

Inner Join, Left Join, Right Join, Full Join, Self Join




```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

Ex:

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

Altele

https://www.w3schools.com/sql/default.asp

- Select Distinct
- And, Or, Not
- Order By
- Min and Max
- Count, Avg, Sum
- Like
- Wildcards (%, _)
- In
- Between
- Aliases
- Union
- Group By
- Having
- Exists

Tips & tricks

Export & Import

Dump the database
→ mysqldump -h localhost -u username -p database_name -r “/home/mbinzari/dump.sql”

Import database
→ mysql -h localhost -u username -p database_name
mysql> SOURCE “/home/mbinzari/dump.sql”;

Slow Queries

```
# File : /etc/mysql/mysql.conf.d/mysqld.cnf
slow_query_log = 1
slow_query_log_file = /var/log/mysql/mysql-slow.log
long_query_time = 1
log-queries-not-using-indexes
```

EXPLAIN

https://www.sitepoint.com/using-explain-to-write-better-mysql-queries/