Laborator 2

Crearea si manipularea bazelor de date si a tabelelor în MySQL

În cadrul acestui capitol se va trata modul în care se creează o bază de date, precum şi tabelelele din aceasta, bazându-ne pe un exemplu concret şi anume cel prezentat în cadrul laboratorului 1

Operații de creare și manipulare a bazelor de date

- Instrucțiunea **SHOW DATABASES**

Pentru a afla ce baze de date există într-un anumit moment pe server se va folosi instrucțiunea

```
SHOW DATABASES
```

Această instrucțiune va genera ca rezultat o listă a bazelor de date asupra căreia utilizatorul are acces.

- Instrucțiunea **CREATE DATABASE**

Pentru a putea folosi o bază de date, aceasta trebuie să fie mai întâi creată. Comanda pentru crearea unei baze de date este:

```
CREATE DATABASE <nume_bază_de_date>
```

unde <nume_bază_de_date> este numele bazei de date ce se doreşte a fi creată.

- Instrucțiunea **USE**

Odată ce s-a creat o bază de date, trebuie create şi tabelele care intră în componența acesteia. Pentru fiecare sesiune MySQL, trebuie selectată baza de date ce urmează a fi folosită, prin comanda:

```
USE <nume_ bază _de_date>
```

unde <nume_bază_de_date> este numele bazei de date ce se doreşte a fi folosită.

Se poate afla care este baza de date ce este selectată în sesiunea MySQL curentă, prin rularea comenzii `SELECT DATABASE()`

- Instrucțiunea **DROP DATABASE**

Comanda DROP DATABASE şterge toate tabelele dintr-o bază de date precum şi baza de date care le conține. Pentru a putea şterge o bază de date, este nevoie ca utilizatorul curent să aibă acest privilegiu. Comanda completă pentru a şterge o bază de date este:

DROP DATABASE [IF EXISTS] <nume_bază_de_date>

unde < nume_bază_de_date > este numele bazei de date ce se doreşte să se şteargă.

Tabele în MySQL

MySQL nu se utilizează numai pentru manipularea datelor din tabele, ci și pentru crearea și manipularea tabelelor în sine.

Există două modalități de creare a tabelelor dintr-o bază de date și anume:

- majoritatea programelor de gestiune a bazelor de date sunt prevăzute cu instrumente de administrare, care pot fi utilizate la crearea și gestionarea interactivă a tabelelor din baza de date
- tabelele pot fi create și manipulate în mod direct, prin folosirea instrucțiunilor MySQL, modalitate care va fi utilizată în cele ce urmează.

Pentru crerea unei baze de date este necesară determinarea în prealabil a colecției de tabele și a caracteristicilor acestora.

În momentul creării unor tabele, se dorește stabilirea unor detalii de proiectare precum: tipurile de date care vor fi stocate în tabel, coloanele care nu acceptă valori de NULL, valori implicite, precum și reguli și valori necesare pentru cheile primare și cele secundare.[7]

- **Tipuri de date în limbajul MySQL**

Orice obiect care conține o dată are asociat un tip de date prin care se definește ce fel de valori poate stoca obiectul. Specificarea unui tip de date presupune definirea mai multor caracteristici:

- natura datelor (caracter, întreg, binar, etc.)
- dimensiunea valorilor stocate (numărul de octeți)
- precizia (pentru valorile numerice, numărul de cifre zecimale)

Tipuri numerice

MySQL suportă toate tipurile de date numerice ale standardului SQL. Aceste tipuri de date includ tipurile numerice exacte (integer, smallint, decimal şi numeric), precum şi cele cu zecimale (float, real, double precision).

Dacă se specifică atributul ZEROFILL pentru un tip numeric, atunci MYSQL adaugă automat atributul UNSIGNED coloanei, adică tipul este fără semn. Implicit, tipurile care sunt cu semn au atributul SIGNED.

Tabel 1. Tipuri numerice

Tipuri	Descriere
Bit[(M)]	M indică numărul de biți per valoare, de la 1 la 64
BOOL, BOOLEAN	Valoarea 0 = fals. Valori nenule = adevărat
SMALLINT[(M)] **[UNSIGNED] [ZEROFILL]**	Valori cu semn: [-32768, 32767]. Valori fără semn: [0, 65535].
MEDIUMINT[(M)] **[UNSIGNED] [ZEROFILL]**	Valori cu semn: [-8388608, 8388607]. Valori fără semn: [0, 16777215].
INT[(M)] **INTEGER[(M)**	Valori cu semn: [-2147483648, 2147483647]. Valori fără semn: [0, 4294967295].
TINYINT[(M)] **[UNSIGNED] [ZEROFILL]**	Un integer foarte mic: Valori cu semn: [2147483648, 2147483647]. Valori fără semn: [0, 4294967295].
BIGINT[(M)]	Valori cu semn: [-9223372036854775808, 9223372036854775807]. Valori fără semn: [0, 18446744073709551615].
FLOAT **FLOAT[(M,D)]**	Valori cu zecimale de mărimi mici. [-3.402823466E+38, -1.175494351E-38] [1.175494351E-38, 3.402823466E+38] M – numărul total de cifre D- numărul total de cifre zecimale
DOUBLE **DOUBLE[(M,D)]**	Valori cu zecimale de mărimi mici. [1.7976931348623E+308, -2.2250738585072E-308, 0] [2.2250738585072014E-308, 1.7976931348623157E+308] [1.175494351E-38, 3.402823466E+38] M – numărul total de cifre D- numărul total de cifre zecimale.
DECIMAL[(M[,D])]	Număr “exact” cu zecimale. M – numărul total de cifre D- numărul total de cifre zecimale

Tipuri dată calendaristică

Tipurile dată calendaristică definesc tipuri destinate pentru a reprezenta data calendaristică şi ora. În tabelul 2 sunt prezentate cele mai utilizate tipuri de acest gen.[10]

Tabel 5.2. Tipuri dată calendaristică

Tipuri	**Descriere**
DATE	Pentru a descrie o dată în formatul 'YYYY-MM-DD'. Interval ['1000-01-01', '9999-12-31'].
DATETIME	Pentru a descrie dată şi ora în formatul 'YYYY-MM-DD HH:MM:SS'. Interval permis ['1000-01-01 00:00:00', '9999-12-31 23:59:59'].
TIMESTAMP	Reprezinta numarul de secunde trecute de la ('1970-01-01 00:00:00') Interval permis ['1970-01-01 00:00:01.000000', '2038-01-19 03:14:07.999999']

TIME	Pentru a descrie ore, minute şi secunde în formatul 'HH:MM:SS'. Interval permis ['-838:59:59', '838:59:59'].
YEAR[(2|4)]	Reprezinta anul cu 2 sau 4 cifre, predefinite fiind valorile cu 4 cifre. YEAR(2) sau YEAR(4) difer[prin modul de afisare, dar au acelasi interval pentru valori. In formatul de 2 cifre, valorile afisate de la 70 la 69, reprezinta anii de la 1970 la 2069.

Tipuri șir de caractere

Tipurile şir de caractere definesc şiruri de caractere de lungime fixă sau variabilă. Mai jos sunt prezentate cele mai utilizate tipuri:[10]

Tabel 3.Tipuri şir de caractere

Tipuri	Descriere
CHAR[(M)]	Un şir de caractere de lungime fixă, căruia îi sunt adăugate spaţii la dreapta până ajunge la lungimea specificată. M- numărul de caractere. Valorea implicită este 1
VARCHAR(M)	Un şir de caractere de lungime variabilă. M- numărulmaximde caractere. Intervalul permis este [0, 65535]
BINARY **VARBINARY**	Aceste tipuri sunt similare cu CHAR si VARCHAR, dar acestea contin siruri de caractere binare.Acest lucru inseamna ca sortarea si comparatia dintre valori se bazeaza pe valorile numerice ale octetior din componenta lor.
BLOB[(M)]	Reprezintă un tip de dată Blog de o lungime maximă de 65,535 octeţi. M- lungimea în octeţi
TEXT[(M)]	Reprezintă un tip de dată Text de o lungime maximă de 65,535 caractere. M- lungimea în octeţi.
ENUM('value1','value2',...)	Pentru a reprezenta o enumerare. Un obiect de acest tip poate să fie un şir de caractere, a cărui valoare poate să ia numai valori specificate de lista de valori 'value1', 'value2', ..., sau NULL.
SET('value1','value2',...)	Pentru a reprezenta o mulţime. Un obiect de acest tip, poate să aibă zero sau mai multe valori, fiecare trebuind să fie aleasă numai din lista de valori 'value1', 'value2', etc.

- **Operaţii de creare şi manipulare a tabelelor**

Instrucţiunea **CREATE TABLE,** utilizată pentru crearea unui tabel, are sintaxa:

```
CREATE TABLE[IF NOT EXISTS]<[nume_bază_de_date.]nume_tabelă>
{ <nume_coloană1> definiţie_coloană1;
<nume_coloană2> definiţie_coloană2;
<nume_coloană3> definiţie_coloană3;
.....
PRIMARY KEY (<nume_coloană>)...
UNIQUE <nume_coloană>...
FOREIGN KEY (index_nume_coloană,...) REFERENCES<nume_tabela_referita>.<parametru_referit>}
```

În definiţie_coloană pot apărea următorii parametri:

```
tip_date [NOT NULL | NULL] [DEFAULT valoare_implicită][AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
index_nume_coloană: <nume_coloană> [(lungime)] [ASC | DESC]
tip_date: BIT[(lungime)] | TINYINT[(lungime)] | SMALLINT[(lungime)] … etc
```

unde:

- < **nume_bază_de_date** > reprezintă numele bazei de date din care face parte tabelul care se dorește a fi creat
- **<nume_tabelă>** reprezintă numele tabelei ce va fi creată
- **<nume_coloană1>** reprezintă numele unic al coloanei ce va fi creată în tabelă
- **definiţie_coloană1** va conține tipul de date al valorilor din coloană, dacă acestea au valoare implicită și restricțiile pentru valorile acesteia dacă este cazul (ex. NULL, NOT NULL).
- **<index_nume_coloană>** se foloseşte pentru definirea câmpului din tabela curentă ce va fi folosit ca şi cheie străină.
- **PRIMARY KEY:** este constrângerea de integritate a relaţiei care impune valori unice şi nenule pentru coloanele care fac parte din cheia primară; poate exista o singură cheie primară pentru o tabelă dată.
- **UNIQUE**: impune valori unice pentru una sau mai multe coloane, este de fapt mecanismul de definire al cheilor candidate din tabelă; pot exista mai multe constrângeri de tip UNIQUE la nivelul unei tabele.
- **FOREIGN KEY… REFERENCES**: această constrângere impune condiţia ca orice valoare a coloanei curente să se regăsească printre valorile coloanelor din tabela referită; coloana referită trebuie să îndeplinească fie constrângerea de primary key, fie cea de UNIQUE.[9]

- Instrucțiunea **SHOW TABLES**: cu ajutorul acestei instrucțiuni utilizatorul poate vizualiza o listă cu toate tabelele incluse în baza de date pe care o folosește în momentul apelării acestei funcții. Tabelele create pot fi vizualizare și în partea dreaptă a ecranului în fereastra Object Browser.
- Instrucțiunea **DESCRIBE**: apelarea acestei instrucțiuni va afișa coloanele tabelului cu tipul de date al datelor conținute în acestea și alte caracteristici care ajută la verificarea acurateței cu care tabelul a fost creat. Comanda completă este:

```
DESCRIBE<nume tabel>
```

- Instrucțiunea **ALTER TABLE:** modifică definiţia unei tabele realizând orice combinaţie a următoarelor categorii de operaţii:

    - Modificarea definiţiei unei coloane sau constrângeri;

- Adăugarea/ ştergerea unei coloane sau constrângeri;
- Activarea/ dezactivarea unei constrângeri sau a unui trigger

```
ALTER TABLE<nume_tabela>
definitie_alter [, definitie_alter] ...
definitie_alter:
 ADD [COLUMN]<nume_coloana>definiţie_coloană [FIRST | AFTER <nume_coloana2> ]
 | ADD [COLUMN] (<nume_coloana>definiţie_coloană, ...)
 | ADD {INDEX|KEY} [index_name] (index_nume_coloană, ...)
 | ADD [CONSTRAINT [symbol]] PRIMARY KEY (index_nume_coloană,...)
| ADD [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY] [index_name] index_nume_coloană,...)
 | ADD [CONSTRAINT [symbol]] FOREIGN KEY (index_nume_coloană,...) reference_definition
 | ALTER [COLUMN] <nume_coloana> {SET DEFAULTliteral | DROP DEFAULT}
 |CHANGE[COLUMN]<nume_vechi_coloană><nume_nou_coloană> definiţie_coloană
 | MODIFY [COLUMN] <nume_coloana>definiţie_coloană
 | DROP [COLUMN] <nume_coloana>
 | DROP PRIMARY KEY
 | DROP {INDEX|KEY} <nume_index>
 | DROP FOREIGN KEY fk_symbol
 | RENAME [TO] <nume_nou_tabelă>
```

Pentru instrucţiunea de mai sus, semnificaţia parametrilor este următoarea:

- **<Nume_tabelă>:** numele tabelei ce dorim să o modificăm;
- În **definiţie_alter** se vor preciza toate câmpurile, constrângerile ce se doresc a fi adăugate, şterse sau modificate în tabelă;
- **definiţie_coloană, index_nume_coloană** au aceeaşi structură şi semnificaţie ca şi la CREATE TABLE.
- **<nume_vechi_coloană>:** numele vechi al coloanei în cazul în care se doreşte redenumirea coloanei;
- **<nume_nou_coloană>:** numele nou al coloanei în cazul în care se doreşte redenumirea coloanei;
- **<nume_coloana2>:** numele tabelei după care să se adaugă noul câmp, definit de <nume_tabelă>;
- **<nume_ nou_tabelă>:** numele nou al tabelei în cazul în care se doreşte redenumirea tabelei;
- **symbol** şi **fk_symbol:** sunt nume simbolice ce vor fi asociate unor constrângeri, respectiv unei chei străine;
- **literal:** valoare explicită ce se doreşte a fi setată ca valoare implicită pentru un câmp.

- Instrucţiunea **DROP TABLE**

```
DROP [TEMPORARY] TABLE [IF EXISTS]
Nume_tabel1 [, nume_tabel2] ...
```

Această instrucțiune șterge unul sau mai multe tabele, împreună cu toate datele conţinute, constrângeri, indecşii şi triggerele asociate. Dacă unul dintre tebelele din listă nu există, MySQL va returna o eroare care specifică numele tabelului care nu a putut fi șters, dar șterge tabelele celelalte.

Folosirea **IF EXISTS** previne eroarea care ar putea apărea dacă tabelele nu există în baza de date.

Comentarii ale codului: limbajul SQL acceptă două tipuri de comentarii şi anume:

--text comentariu

care se foloseşte pentru comentarii care sunt conţinute pe o singură linie de cod, textul conţinut după "--" fiind ignorant şi

/*text comentariu*/

care se foloseşte pentru comentarii care se întind pe mai multe linii de cod, textul conţinut între "/*" şi "*/" fiind ignorat. [4]

Aplicații

Pentru a rula exemplele ce urmează este nevoie să se pornească serverul MySQL şi utilitarul MySQL Workbench.

Mai jos este prezentată diagrama ce va fi folosită în exerciţiile din laboratoarele de BDIE.

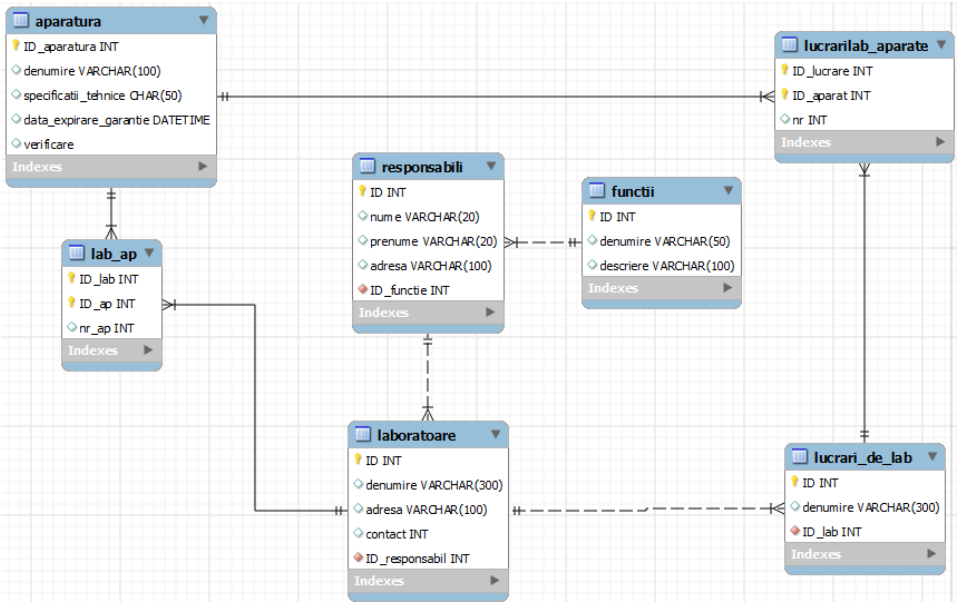

Fig.1.Diagrama bazei de date.

Aplicaţia 1: Să se creeze o nouă bază de date denumită gestiune_laboratoare

Rezolvare

```
CREATE DATABASE IF NOT EXISTS gestiune_laboratoare;
```

Aplicaţia 2: Să se selecteze baza de date creată anterior şi să se creeze o nouă tabelă intitulată **APARATURA** care să aibă următorii parametri:

Rezolvare

ID_aparatura –integer – cheie primară
Denumire –varchar(100)
Specificaț ii_tehnice – char(50)
Data_expirare_garantie – datetime
Verificare – bool

```
USE gestiune_laboratoare;
CREATE TABLE IF NOT EXIST aparatura(ID_aparatura INTEGER NOT NULL
                AUTO_INCREMENT,
                        denumire VARCHAR(100) NULL,
                        specificatii_tehnice CHAR(50) NULL,
                        data_expirare_garantie DATETIME NULL,
                        verificare BOOL,
                        PRIMARY KEY(ID_aparatura));
```

Aplicația 3 Să se selecteze baza de date creată anterior şi să se creeze o nouă tabelă intitulată **FUNCTII** care să aibă următorii parametri

Rezolvare

ID – integer-cheie primară
Denumire – varchar(50)
Descriere – varchar(100)

```
USE gestiune_laboratoare;
CREATE TABLE IF NOT EXISTS functii(ID INTEGER NOT NULL AUTO_INCREMENT,
                                   denumire VARCHAR(50) NULL,
                                   descriere VARCHAR(100) NULL,
                                   PRIMARY KEY(ID));
```

Aplicația 4 Să se selecteze baza de date creată anterior şi să se creeze o nouă tabelă intitulată **RESPONSABILI** care să aibă următorii parametri:

Rezolvare

ID –integer-cheie primară
Nume – varchar(20)
Prenume – varchar(20)
Adresa – varchar(100)
ID_functie – integer- cheie străină,care se referă la parametrul ID din tabelul Functii

```
USE gestiune_laboratoare;
CREATE TABLE if not exists responsabili(ID INTEGER NOT NULL
                AUTO_INCREMENT,
                        nume VARCHAR(20) NULL,
                        prenume VARCHAR(20) NULL,
                        adresa VARCHAR(100) NULL,
                        ID_functie INTEGER NOT NULL,
                        PRIMARY KEY(ID),
                        FOREIGN KEY(ID_functie) REFERENCES
                                functii(ID));
```

Aplicația 5 Folosiți instrucțiunea ALTER TABLE în tabelul Responsabili pentru a adăuga o coloană vârstă de tip integer

Rezolvare

```
ALTER TABLE responsabili ADD COLUMN (varsta INTEGER NOT NULL);
```

Aplicație 6 Folosiți instrucțiunea ALTER TABLE în tabelul Aparatura pentru a:

-șterge câmpul specificatii_tehnice

-redenumi câmpul Data_expirare_garantie în Expirare_garantie.

- adăuga o nouă coloană spec_tehn char(50) null

-modifica tipului datelor din coloane spec_tehn din char(50) în varchar(50)

Pentru a observa dacă modificările asupra tabelului au fost efectuate se poate folosi instrucțiunea **DESCRIBE**

Rezolvare

```
ALTER TABLE aparatura DROP specificatii_tehnice,
CHANGE COLUMN data_expirare_garantie Expirare_garantie DATETIME,
ADD COLUMN (spec_tehn char(50) null);
ALTER TABLE aparatura MODIFY spec_tehn varchar(50) null;
```

Exerciții propuse

1. Să se selecteze baza de date creată anterior şi să se creeze o nouă tabelă intitulată **LABORATOARE** care să aibă următorii parametri:
   ID – integer-cheie primară
   Denumire-varchar(300)
   Adresa – varchar(100)
   Contact – integer
   Suprafata – integer
   Capacitate – integer
   ID_responsabil – integer - cheie străină,care se referă la parametrul ID din tabelul Responsabili

2. Să se selecteze baza de date creată anterior şi să se creeze o nouă tabelă intitulată **LAB_AP** care să aibă următorii parametri:
   ID_lab - integer- cheie primară ș i cheie străină care se referă la parametrul ID din tabelul Laboratoare
   ID_ap – integer- cheie primară ș i cheie străină care se referă la parametrul ID_aparatura din tabelul Aparatură
   nr_ap – integer

3. Să se selecteze baza de date creată anterior şi să se creeze o nouă tabelă intitulată **LUCRARI_DE_LAB** care să aibă următorii parametri:
   ID – integer – cheie primara
   Denumire – varchar(300)
   ID_lab – integer-cheie străină care se referă la parametrul ID din tabelul Laboratoare

4. Să se selecteze baza de date creată anterior şi să se creeze o nouă tabelă intitulată **LUCRARILAB_APARATE** care să aibă următorii parametri:

ID_lucrare – integer-cheie primară - cheie străină care se referă la parametrul ID din tabelul Lucrari_de_lab

ID_aparat – integer – cheie primară - cheie străină care se referă la parametrul ID_aparatură din tabelul Aparatură

Nr – integer

5. Să se folosească instrucțiunea **SHOW DATABASES; și SELECT DATABASE().** Bazele de date existente se pot observa și in fereastra Object Browser.

6. Să se folosească instrucțiunea **SHOW TABLES;**

7. Să se folosească instrucțiunea **DESCRIBE** pentru tabelul Lucrari_de_lab

Aplicații

1. Creati diagrama bazei de date care să conțină toate tabelele create anterior

- se salvează modificările la baza de date creată
- in MySQL Workbench se selectează fereastra HOME și se alege opțiunea **Create EER Model from SQL Script**

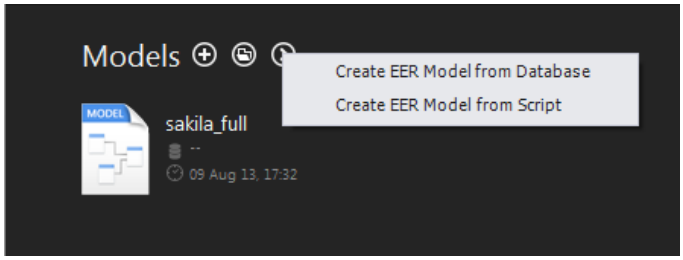

Fig.2.Primul pas în crearea diagramei unei baze de date cu MySQL Workbench.

- se selectează baza de date salvată anterior și opțiunea **Add Diagram**
- cu **Drag and Drop** se duc toate tabelele din fereastra Catalog Tree în fereastra de lucru, iar conexiunile vor fi create automat de către program

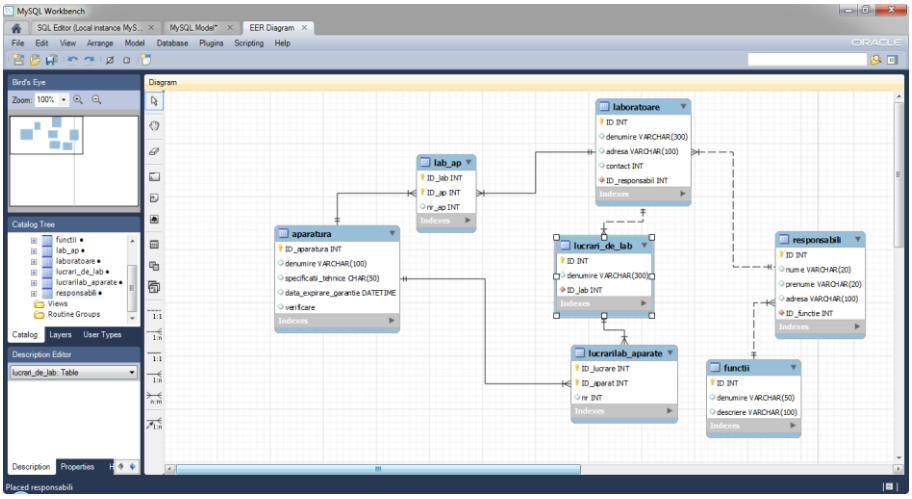

Fig.3.Diagrama bazei de date.

- la suprapunerea mouse-ului pe fiecare dintre conexiunile existente se va afisa legătura dintre tabele, cheia primară şi cea străină prin care se face conexiunea fiind evidenţiate; de asemenea, tipul de legătură (1-1, 1-N, M-N) este precizat cu ajutorul tipului de linie de legătură prezent în diagramă