НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КПІ»

ФAКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

ЛАБОРАТОРНА РОБОТА №2

*з дисципліни*

*«Бази даних»*

**ТЕМА: «Розробка Web-орієнтованої інформаційно-пошукової системи»**

Виконала студентка

ІІ курсу групи КП-42

Білогуб Дар’я

Перевірив:

Петрашенко Андрій Васильович

Оцінка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, підпис)

Київ 2016

**Завдання**

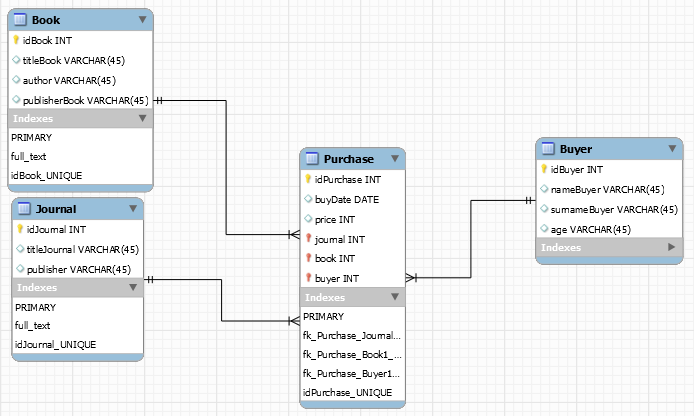
*Метою роботи* є здобуття практичних навичок проектування реляційних баз даних та створення прикладних програм щодо їх обробки.

*Завдання* роботи полягає у наступному:

1. Розробити модель «сутність-зв’язок» предметної галузі, обраної студентом, відповідно до пункту «Вимоги до ER-моделі».
2. Перетворити розроблену модель у структуру бази даних MySQL.
3. Виконати нормалізацію бази даних до 3НФ, **затвердити базу даних у викладача**.
4. Реалізувати функціональні вимоги, наведені нижче.

**Варіант 2**

****

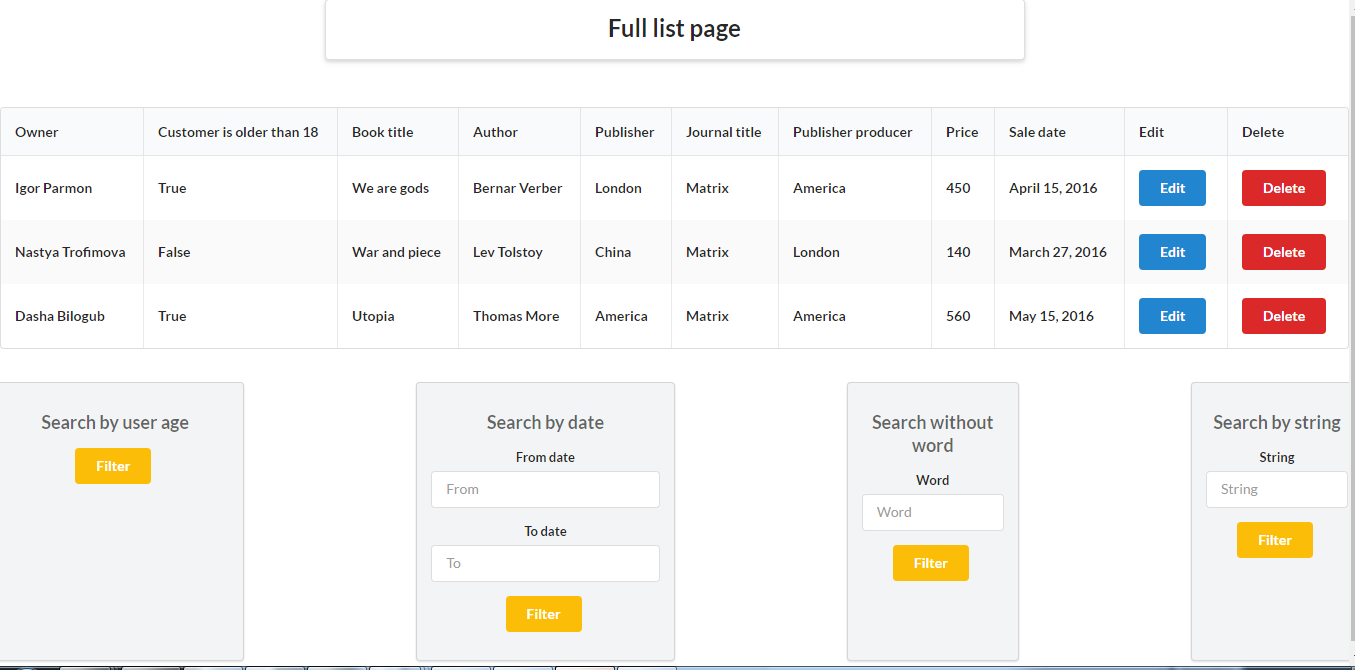
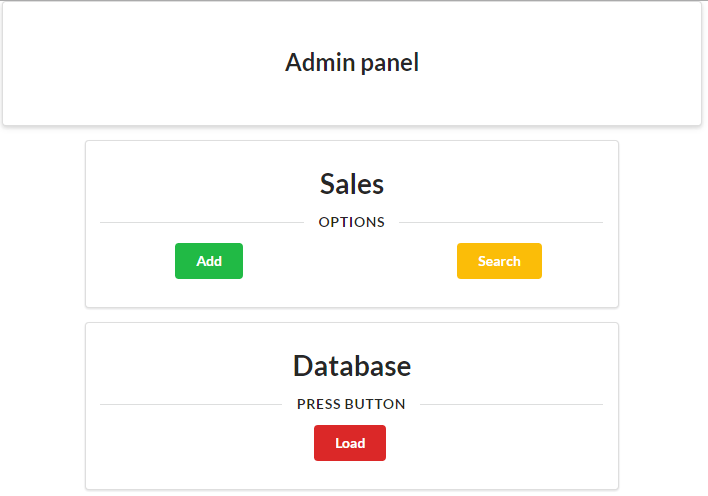
****

*Рис. 1. Графічне подання моделі*

**Фрагменти програмного коду**

**class** DB(object):  
 **def** \_\_init\_\_(self):  
 self.connection = **None  
  
 def** connect(self):  
 **if** self.connection **is not None**:  
 **return  
 try**:  
 self.connection = mdb.connect(**'127.0.0.1'**, **'root'**, **'5730518'**, **'lab2'**)  
  
 **except** mdb.Error **as** e:  
 print(**"Error %d: %s"** % (e.args[0], e.args[1]))  
 self.connection = **None  
  
 def** close(self):  
 **if** self.connection **is not None**:  
 self.connection.close()  
 self.connection = **None  
  
 def** initialization(self):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"DELETE FROM purchase"**)  
 cur.execute(**"ALTER TABLE purchase AUTO\_INCREMENT = 1"**)  
 cur.execute(**"commit"**)  
  
 cur.execute(**"DELETE FROM book"**)  
 cur.execute(**"ALTER TABLE book AUTO\_INCREMENT = 1"**)  
 cur.execute(**"commit"**)  
  
 cur.execute(**"DELETE FROM journal"**)  
 cur.execute(**"ALTER TABLE journal AUTO\_INCREMENT = 1"**)  
 cur.execute(**"commit"**)  
  
 cur.execute(**"DELETE FROM buyer"**)  
 cur.execute(**"ALTER TABLE buyer AUTO\_INCREMENT = 1"**)  
 cur.execute(**"commit"**)  
  
 data = json.load(open(**'test.json'**))  
 buyer\_list = data[**'buyer'**]  
 **for** buyer **in** buyer\_list:  
 **try**:  
 name = str(buyer[**'nameUser'**])  
 surname = str(buyer[**'surnameUser'**])  
 age = str(buyer[**'age'**])  
 cur.execute(**"INSERT INTO buyer (nameBuyer, surnameBuyer, age) VALUES('%s', '%s', '%s')"** % (name, surname, age))  
 cur.execute(**"commit"**)  
 print(name, surname, age)  
 **except** IndexError:  
 **pass  
 continue** data = json.load(open(**'test.json'**))  
 film\_list = data[**'journal'**]  
 **for** film **in** film\_list:  
 **try**:  
 titleJournal = str(film[**'titleJournal'**])  
 publisher = str(film[**'publisher'**])  
 cur.execute(**"INSERT INTO journal (titleJournal, publisher) VALUES('%s', '%s')"** %  
 (titleJournal, publisher))  
 cur.execute(**"commit"**)  
 print(titleJournal, publisher)  
 **except** IndexError:  
 **pass  
 continue** data = json.load(open(**'test.json'**))  
 book\_list = data[**'book'**]  
 **for** book **in** book\_list:  
 **try**:  
 titleBook = str(book[**'titleBook'**])  
 author = str(book[**'author'**])  
 publisherBook = str(book[**'publisher'**])  
 cur.execute(**"INSERT INTO book (titleBook, author, publisherBook) VALUES('%s', '%s', '%s')"** %  
 (titleBook, author, publisherBook))  
 cur.execute(**"commit"**)  
 print(titleBook, author)  
 **except** IndexError:  
 **pass  
 continue** data = json.load(open(**'test.json'**))  
 purchase\_list = data[**'purchase'**]  
 **for** purchase **in** purchase\_list:  
 **try**:  
 buyer = int(purchase[**'buyer'**])  
 book = int(purchase[**'book'**])  
 price = int(purchase[**'price'**])  
 saleDate = str(purchase[**'saleDate'**])  
 journal = int(purchase[**'titleJournal'**])  
 cur.execute(  
 **"INSERT INTO purchase (buyDate, price, journal, book, buyer) VALUES('%s', '%d', '%d', '%d', "  
 "'%d')"** % (saleDate, price, journal, book, buyer))  
 cur.execute(**"commit"**)  
 print(price)  
 **except** IndexError:  
 **pass  
 continue** self.close()  
  
 **def** getBuyer(self):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"SELECT \* FROM buyer"**)  
 self.close()  
 **return** cur.fetchall()  
  
 **def** getJournal(self):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"SELECT \* FROM journal"**)  
 self.close()  
 **return** cur.fetchall()  
  
 **def** getBooks(self):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"SELECT \* FROM book"**)  
 self.close()  
 **return** cur.fetchall()  
  
 **def** getPurchaseList(self):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"SELECT \* FROM purchase, journal, book, buyer WHERE purchase.buyer=buyer.idBuyer "  
 "AND purchase.book=book.idBook "  
 "AND purchase.journal=journal.idJournal"**)  
 self.close()  
 **return** cur.fetchall()  
  
 **def** getPurchase(self, id):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"SELECT \* FROM purchase, journal, book, buyer WHERE purchase.buyer=buyer.idBuyer "  
 "AND purchase.book=book.idBook "  
 "AND purchase.journal=journal.idJournal "  
 "AND purchase.idPurchase=%d"** % int(id))  
 self.close()  
 **return** cur.fetchone()  
  
 **def** savePurchase(self, buyDate, price, book, journal, buyer):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"INSERT INTO purchase (buyDate, price, journal, book, buyer) "  
 "VALUES('%s', '%d', '%d', '%d', '%d')"** % (buyDate, int(price), int(book), int(journal), int(buyer)))  
 cur.execute(**"commit"**)  
 self.close()  
  
 **def** updatePurchase(self, idPurchase, buyDate, price, book, journal, buyer):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"UPDATE purchase SET buyDate='%s', price='%d', "  
 "book='%d', journal='%d', buyer='%d' where idPurchase=%d"** %  
 (buyDate, int(price), int(book), int(journal), int(buyer), int(idPurchase)))  
 cur.execute(**"commit"**)  
 self.close()  
  
 **def** removePurchase(self, id):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"DELETE FROM purchase WHERE idPurchase = '%d' "** % (int(id)))  
 cur.execute(**"commit"**)  
 self.close()  
  
*#logicalType* **def** getPurclaseListByAge(self):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"SELECT \* FROM purchase, journal, book, buyer WHERE "  
 " (buyer.age LIKE '%s'"  
 " AND purchase.journal=journal.idJournal"  
 " AND purchase.book=book.idBook"  
 " AND purchase.buyer=buyer.idBuyer)"** % **"True"**)  
 self.close()  
 **return** cur.fetchall()  
  
*#date* **def** getPurchaceListByDate(self, fromDate, toDate):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"SELECT \* FROM purchase, journal, book, buyer WHERE purchase.buyDate BETWEEN '%s' AND '%s' "  
 "AND purchase.journal=journal.idJournal "  
 "AND purchase.buyer=buyer.idBuyer "  
 "AND purchase.book=book.idBook "** % (fromDate, toDate))  
 self.close()  
 **return** cur.fetchall()  
 **def** fullTextSearch(self, phrase):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"SELECT \* FROM purchase, journal, book, buyer WHERE "  
 "(MATCH (book.publisherBook) AGAINST ( ' \"%s\"' IN BOOLEAN MODE )"  
 " AND purchase.journal=journal.idJournal"  
 " AND purchase.book=book.idBook"  
 " AND purchase.buyer=buyer.idBuyer)"** % phrase)  
 self.close()  
 **return** cur.fetchall()  
 **def** getListWithoutWord(self, phrase):  
 self.connect()  
 **if** self.connection **is None**:  
 **return** []  
 cur = self.connection.cursor(mdb.cursors.DictCursor)  
 cur.execute(**"SELECT \* FROM purchase, journal, book, buyer WHERE "  
 " (book.publisherBook NOT LIKE '%s'"  
 " AND purchase.journal=journal.idJournal"  
 " AND purchase.book=book.idBook"  
 " AND purchase.buyer=buyer.idBuyer)"** % phrase)  
 self.close()  
 **return** cur.fetchall()

**Snapshots**

****