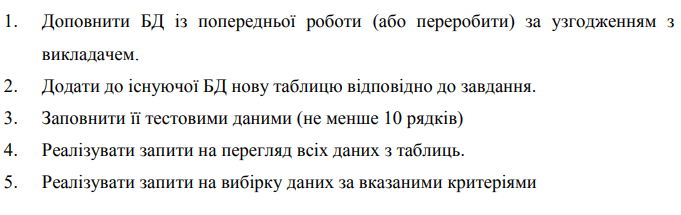
**Лабораторна робота №4**

Створення бази даних та sql запитів для обробки даних. Продовження

**Мета роботи:** отримати навички створення таблиці БД, ознайомитись з типами

даних, що можуть зберігатись в БД

*Завдання*



**Варіант 5:**

Book: id, Назва, Автор, Видавництво, Рік видання, Кількість сторінок, Ціна.

Вивести:

a) список книг заданого автора в порядку зростання року видання;

b) список книг, що видані заданим видавництвом;

c) список книг, що випущені після заданого року;

d) список авторів в алфавітному порядку;

e) список видавництв, книги яких зареєстровані в системі без повторів;

f) для кожного видавництва визначити список книг, виданих ним

**Хід роботи**

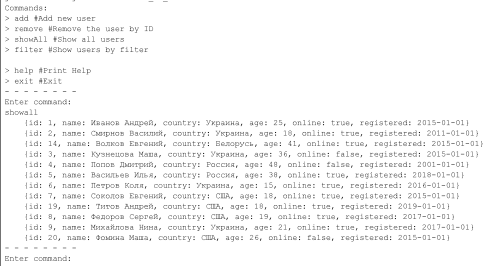
1. **Текст програми**

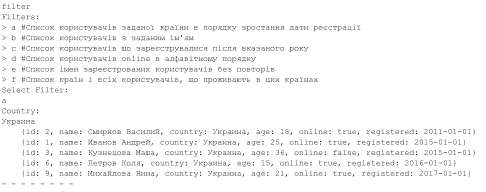
***com.App.java***

**public class** App {  
 **private** Connection **connection**;  
  
 **private static final** String ***sqlSelectAll*** = **"SELECT book.id, book.name, author.name as author,\n"** +  
 **" publisher.name AS publisher, book.publis\_date, book.pages, book.price\n"** +  
 **"FROM book\n"** +  
 **"LEFT JOIN book\_author ON book.id = book\_author.book\_id\n"** +  
 **"LEFT JOIN author ON author.id = book\_author.author\_id\n"** +  
 **"LEFT JOIN publisher ON book.publisher\_id = publisher.id\n "**;  
  
 **private int limitToShow** = 15;  
  
 **public static void** main(String[] args) {  
 App app = **new** App();  
 app.run();  
 }  
  
 **public void** run() {  
 connect();  
 runCli();  
 }  
  
 **private void** runCli() {  
 cliPrintHelp();  
 *hr*();  
  
 **while** (**true**) {  
 String command = *promptLine*(**"Enter command: "**).trim().toLowerCase();  
  
 **if** (command.equals(**"exit"**)) **break**;  
 **else** useCommand(command);  
  
 *hr*();  
 }  
  
 exitApp();  
 }  
  
 **private void** useCommand(String command) {  
 **switch** (command) {  
 **case "showall"**:  
 cliShowAllBooks();  
 **break**;  
 **case "remove"**:  
 cliRemoveBook();  
 **break**;  
 **case "filter"**:  
 cliShowBooksByFilter();  
 **break**;  
 **case "limit"**:  
 cliSetLimit();  
 **break**;  
 **case "help"**:  
 cliPrintHelp();  
 **break**;  
 **default**:  
 System.***out***.println(**"Command not found. Try again: "**);  
 **break**;  
 }  
 }  
  
 **private void** cliRemoveBook() {  
 **int** bookId = *promptInt*(**"Enter the book id"**);  
  
 **try** (**var** preparedSt = **connection**.prepareStatement(**"DELETE FROM book WHERE id = ?"**)) {  
 preparedSt.setInt(1, bookId);  
  
 preparedSt.executeUpdate();  
 System.***out***.println(**"Success"**);  
 } **catch** (SQLException e) {  
 System.***out***.println(**"Can't remove the book"**);  
 e.printStackTrace();  
 }  
  
 }  
  
 **private void** cliShowAllBooks() {  
 **try** (**var** preparedSt = **connection**.prepareStatement(***sqlSelectAll*** + **"LIMIT ?"**)) {  
 preparedSt.setInt(1, **limitToShow**);  
  
 **var** rs = preparedSt.executeQuery();  
 **var** books = createBooksFromRs(rs);  
 printBooks(books);  
 } **catch** (SQLException e) {  
 System.***out***.println(**"Can't load books"**);  
 e.printStackTrace();  
 }  
 }  
  
 **private void** cliShowBooksByFilter() {  
 printFiltersHelp();  
 String filter = *promptLine*(**"Select Filter:"**);  
  
 **try** {  
  
 **if** (filter.equalsIgnoreCase(**"a"**)) {  
 String author = *promptLine*(**"Author:"**);  
  
 **var** prepareSt = **connection**.prepareStatement(  
 ***sqlSelectAll*** + **"WHERE author.name LIKE ? ORDER BY book.publis\_date LIMIT ?"** );  
  
 prepareSt.setString(1, **"%"** + author + **"%"**);  
 prepareSt.setInt(2, **limitToShow**);  
  
 **var** rs = prepareSt.executeQuery();  
 printBooks(rs);  
  
 } **else if** (filter.equalsIgnoreCase(**"b"**)) {  
 **var** prepareSt = **connection**.prepareStatement(***sqlSelectAll*** + **"WHERE publisher.name LIKE ? LIMIT ?"**);  
  
 String publisher = *promptLine*(**"Publisher:"**);  
 prepareSt.setString(1, **"%"** + publisher + **"%"**);  
 prepareSt.setInt(2, **limitToShow**);  
  
 **var** rs = prepareSt.executeQuery();  
 printBooks(rs);  
  
 } **else if** (filter.equalsIgnoreCase(**"c"**)) {  
 **var** prepareSt = **connection**.prepareStatement(***sqlSelectAll*** + **"WHERE publis\_date > ? LIMIT ?"**);  
  
 LocalDate date = *promptDate*(**"Publish Date:"**);  
 prepareSt.setString(1, date.toString());  
 prepareSt.setInt(2, **limitToShow**);  
  
 **var** rs = prepareSt.executeQuery();  
 printBooks(rs);  
  
 } **else if** (filter.equalsIgnoreCase(**"d"**)) {  
 **var** prepareSt = **connection**.prepareStatement(  
 **"SELECT DISTINCT author.name AS author FROM book\n"** +  
 **"LEFT JOIN book\_author ON book.id = book\_author.book\_id\n"** +  
 **"LEFT JOIN author ON author.id = book\_author.author\_id\n"** +  
 **"ORDER BY author.name LIMIT ?"** );  
  
 prepareSt.setInt(1, **limitToShow**);  
 **var** rs = prepareSt.executeQuery();  
  
 List<String> authors = **new** ArrayList<>();  
  
 **while** (rs.next()) {  
 authors.add(rs.getString(**"author"**));  
 }  
  
 authors.forEach(System.***out***::println);  
  
 } **else if** (filter.equalsIgnoreCase(**"e"**)) {  
 **var** prepareSt = **connection**.prepareStatement(  
 **"SELECT DISTINCT publisher.name AS publisher FROM book\n"** +  
 **"INNER JOIN publisher ON book.publisher\_id = publisher.id\n "** +  
 **"LIMIT ?"** );  
  
 prepareSt.setInt(1, **limitToShow**);  
 **var** rs = prepareSt.executeQuery();  
  
 List<String> publishers = **new** ArrayList<>();  
  
 **while** (rs.next()) {  
 publishers.add(rs.getString(**"publisher"**));  
 }  
  
 publishers.forEach(System.***out***::println);  
  
 } **else if** (filter.equalsIgnoreCase(**"f"**)) {  
 **var** prepareSt = **connection**.prepareStatement(***sqlSelectAll*** + **"LIMIT ?"**);  
  
 prepareSt.setInt(1, **limitToShow**);  
 **var** rs = prepareSt.executeQuery();  
  
 **var** books = createBooksFromRs(rs);  
 Map<String, HashSet<Book>> map = books.stream()  
 .collect(Collectors.*groupingBy*(Book::getPublisher, Collectors.*toCollection*(HashSet::**new**)));  
  
 map.forEach((publisher, publishersBook) -> {  
 System.***out***.println(**" - "** + publisher);  
 printBooks(publishersBook);  
 });  
  
 } **else** {  
 System.***out***.println(**"Can't find the filter"**);  
  
 }  
  
 } **catch** (SQLException e) {  
 System.***out***.println(**"Filter error"**);  
 e.printStackTrace();  
 }  
 }  
  
 **private void** cliSetLimit() {  
 System.***out***.println(**"Current limit: "** + **limitToShow**);  
 **limitToShow** = *promptInt*(**"Enter new limit: "**);  
 }  
  
 **private** Book createNewBook() {  
 String name = *promptLine*(**"Name:"**);  
 String author = *promptLine*(**"Author:"**);  
 String publisher = *promptLine*(**"Publisher"**);  
 LocalDate publishDate = *promptDate*(**"Publish Date:"**);  
 **int** pages = *promptInt*(**"Pages:"**);  
 **double** price = *promptDouble*(**"Price:"**);  
  
 **return new** Book(0, name, author, publisher, publishDate, pages, price);  
 }  
  
 **private** ArrayList<Book> createBooksFromRs(ResultSet rs) **throws** SQLException {  
 HashMap<Integer, Book> booksMap = **new** HashMap<>();  
  
 **while** (rs.next()) {  
 **int** id = rs.getInt(**"id"**);  
 String author = rs.getString(**"author"**);  
  
 **if** (booksMap.containsKey(id)) {  
 booksMap.get(id).addAuthor(author);  
 **continue**;  
 }  
  
 String name = rs.getString(**"name"**);  
 String publisher = rs.getString(**"publisher"**);  
 LocalDate publishDate = rs.getDate(**"publis\_date"**).toLocalDate();  
 **int** pages = rs.getInt(**"pages"**);  
 **double** price = rs.getDouble(**"price"**);  
  
 Book book = **new** Book(id, name, author, publisher, publishDate, pages, price);  
 booksMap.put(book.getId(), book);  
 }  
  
 **return new** ArrayList<>(booksMap.values());  
 }  
  
 **private void** printBooks(Collection<Book> books) {  
 **if** (books.isEmpty()) {  
 System.***out***.println(**"There are no users"**);  
 **return**;  
 }  
  
 books.forEach((item) -> System.***out***.println(**"\t"** + item));  
 }  
  
 **private void** printBooks(ResultSet rs) **throws** SQLException {  
 **var** books = createBooksFromRs(rs);  
 printBooks(books);  
 }  
  
 **private void** exitApp() {  
 disconnect();  
 }  
  
 **private void** connect() {  
 **try** {  
 **connection** = DriverManager.*getConnection*(  
 **"jdbc:mysql://localhost:3306/java\_sem4\_lab4?serverTimezone=UTC"**,  
 **"db\_user"**, **"qqqqqqqqww"** );  
  
 } **catch** (SQLException e) {  
 System.***out***.println(**"Can't connect to DB"**);  
 e.printStackTrace();  
 }  
 }  
  
 **private void** disconnect() {  
 **try** {  
 **connection**.close();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **private void** cliPrintHelp() {  
 System.***out***.println(**"Commands:"**);  
 System.***out***.println(**"> showAll #Show all books"**);  
 System.***out***.println(**"> remove #Remove the book by ID"**);  
 System.***out***.println(**"> filter #Show books by filter"**);  
 System.***out***.println();  
 System.***out***.println(**"> limit #Set Limit to show"**);  
 System.***out***.println(**"> help #Print Help"**);  
 System.***out***.println(**"> exit #Exit"**);  
 }  
  
 **private void** printFiltersHelp() {  
 System.***out***.println(**"Filters:"**);  
 System.***out***.println(**"> a #Список книг заданого автора в порядку зростання року видання"**);  
 System.***out***.println(**"> b #Список книг, що видані заданим видавництвом"**);  
 System.***out***.println(**"> c #Список книг, що випущені після заданої дати"**);  
 System.***out***.println(**"> d #Список авторів в алфавітному порядку"**);  
 System.***out***.println(**"> e #Список видавництв, книги яких зареєстровані в системі без повторів"**);  
 System.***out***.println(**"> f #Для кожного видавництва визначити список книг, виданих ним"**);  
 }  
  
}

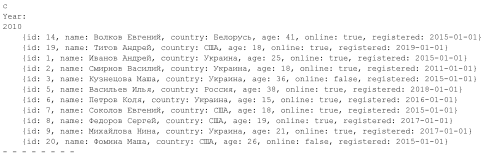
***com.Book.java***  
  
**public class** Book **implements** Comparable<Book>, Serializable {  
 **private int id**;  
 **private** String **name**;  
 **private** ArrayList<String> **authors** = **new** ArrayList<>();  
 **private** String **publisher**;  
 **private** LocalDate **publishDate**;  
 **private int pages**;  
 **private double price**;  
  
 **public** Book(**int** id, String name, String author, String publisher, LocalDate publishDate, **int** pages, **double** price) {  
 setId(id);  
 setName(name);  
 addAuthor(author);  
 setPublisher(publisher);  
 setPublishDate(publishDate);  
 setPages(pages);  
 setPrice(price);  
 }  
  
 @Override  
 **public** String toString() {  
 **return "id="** + **id** +  
 **", name='"** + **name** + **'\''** +  
 **", authors='"** + authorsToString() + **'\''** +  
 **", publisher='"** + **publisher** + **'\''** +  
 **", publishDate="** + **publishDate** +  
 **", pages="** + **pages** +  
 **", price="** + **price**;  
 }  
  
 **private** String authorsToString() {  
 **return** String.*join*(**", "**, **authors**);  
 }  
  
 **public** List<String> getAuthors() {  
 **return authors**;  
 }  
  
 **public void** addAuthor(String author) {  
 **this**.**authors**.add(author);  
 }  
  
 **public** String getPublisher() {  
 **return publisher**;  
 }  
  
 **public void** setPublisher(String publisher) {  
 **this**.**publisher** = publisher;  
 }  
  
 **public** LocalDate getPublishDate() {  
 **return publishDate**;  
 }  
  
 **public void** setPublishDate(LocalDate publishDate) {  
 **this**.**publishDate** = publishDate;  
 }  
  
}

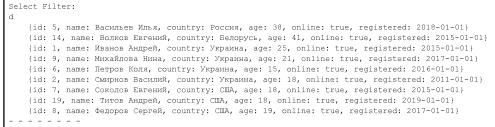
1. **Результат виконання роботи**



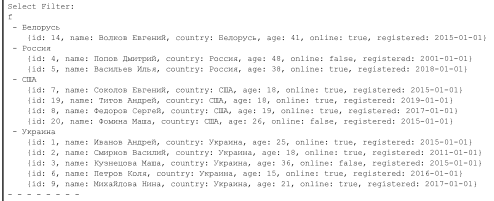












**Висновок:** на цій лабораторній роботі я отримав навички роботи зі зв’язаними таблицями БД.