МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Направление подготовки
09.03.01 Информатика и вычислительная техника

Направленность (профиль)
«Технологии разработки программного обеспечения»

ОТЧЁТ

по реализации проекта для дисциплины «Базы данных»

Система управления библиотечными ресурсами

Преподаватель: к.ф-м.н., доцент кафедры ИТиЭО

_____
(Жуков Н. Н.)

Студенты 2 курса:

Славный Д.М. _____
Красникова Д.Я._____
Красников Д.Я._____

Санкт-Петербург
2024

Оглавление

Ответственные 3

Предметная область 3

Ход выполнения нормализации 3

Объяснение выбранной СУБД 4

ER – диаграмма 5

Исходных текст запросов 6

Исходный код программы 10

Ответственные

Красникова Д.Я. – разработчик проекта. В ее обязанности входит выбор предметной области, определение сущностей и атрибутов, необходимых для создания базы данных, нормализация полученной модели данных и разработка структуры базы данных

Красников Д.Я. – разработчик проекта. В его обязанности входит проектирование и разработка базы данных, включая создание ER-диаграммы для визуального представления структурных связей между таблицами.

Славный Д.М. – разработчик проекта. В его обязанности входит разработка функционала для работы с базой данных, а конкретнее Славный Д.М. создал программу на языке программирования Python, которая позволяет взаимодействовать с базой данных MySQL. Эта программа устанавливает соединение с базой данных, запрашивает у пользователя ввод информации о книге, включая название, автора, год публикации и статус доступности. Эта информация затем добавляется в таблицу "book" в базе данных, запрашивает у пользователя ввод названия жанра. Затем программа проверяет, существует ли уже такой жанр в базе данных. Если жанр существует, используется его ID, если нет, создается новый жанр, и используется его ID, создаётся связь между только что добавленной книгой и жанром в таблице "book_genre". После успешного добавления данных программа выводит все записи из таблицы "book". В случае возникновения ошибок во время выполнения запросов, программа выводит сообщение об ошибке.

Предметная область

Проект базы данных для информационной системы библиотеки. Предметной областью проекта является система управления библиотечными ресурсами, включая книги, аудиокниги, электронные ресурсы, читателей и персонал библиотеки.

Ход выполнения нормализации

После выделения конкретных сущностей для системы управления библиотекой, был сформирован список атрибутов каждой сущности:

КНИГА:

Каждая книга является уникальной единицей и имеет: название, автора, год издания, жанр и статус (доступна/выдана). В качестве ключевого атрибута – book_id. Все атрибуты данной сущности обязательны к заполнению. Атрибут «жанр» является внешним ключом к сущности ЖАНР.

ЧИТАТЕЛЬ:

Каждый читатель является физическим лицом и имеет: имя, контактную информацию, историю бронирования. В качестве ключевого атрибута – reader_id. Все атрибуты данной сущности обязательны к заполнению.

БРОНИРОВАНИЕ:

Каждое бронирование представляет собой уникальное событие и имеет: дату бронирования, читателя (внешний ключ), книгу (внешний ключ), срок возврата. В качестве ключевого атрибута – reservation_id. Все атрибуты данной сущности обязательны к заполнению.

СОТРУДНИК:

Каждый сотрудник является физическим лицом и имеет: имя, должность, контактную информацию. В качестве ключевого атрибута - staff_id. Все атрибуты данной сущности обязательны к заполнению.

ЖАНР:

Каждый жанр имеет уникальное название. Ключевой атрибут - genre_id.

АУДИОКНИГА:

Каждая аудиокнига имеет следующие атрибуты: book_id, e-resourse_id, продолжительность. Ключевой атрибут - audiobook_id.

ЭЛЕКТРОННЫЙ РЕСУРС:

Каждый электронный ресурс имеет следующие атрибуты: ссылку, формат файла. Ключевой атрибут - e-resourse_id.

Книга (Book) - Жанр (Genre): Одна книга может иметь несколько жанров, один жанр может быть присвоен многим книгам. Это отношение "многие ко многим".

Книга (Book) - Бронирование (Reservation): Одна книга может быть забронирована многими читателями, и одно бронирование может включать только одну книгу. Это отношение "один ко многим".

Читатель (Reader) - Бронирование (Reservation): Один читатель может сделать много бронирований, но одно бронирование может быть сделано только одним читателем. Это отношение "один ко многим".

Сотрудник (Staff) - Бронирование (Reservation): Предполагая, что каждое бронирование обрабатывается одним сотрудником, и один сотрудник может обрабатывать множество бронирований, это отношение "один ко многим".

Книга (Book) - Аудиокнига (Audiobook): Одна аудиокнига может быть только у такой же книги, но у одной книги может быть несколько аудиокниг. Это отношение "один ко многим".

Электронный ресурс (E-resource) -Аудиокнига (Audiobook): Так же, как с книгами, один электронный ресурс может быть только у одной аудиокниги, но электронного ресурса может быть несколько аудиокниг. Это отношение "один ко многим".

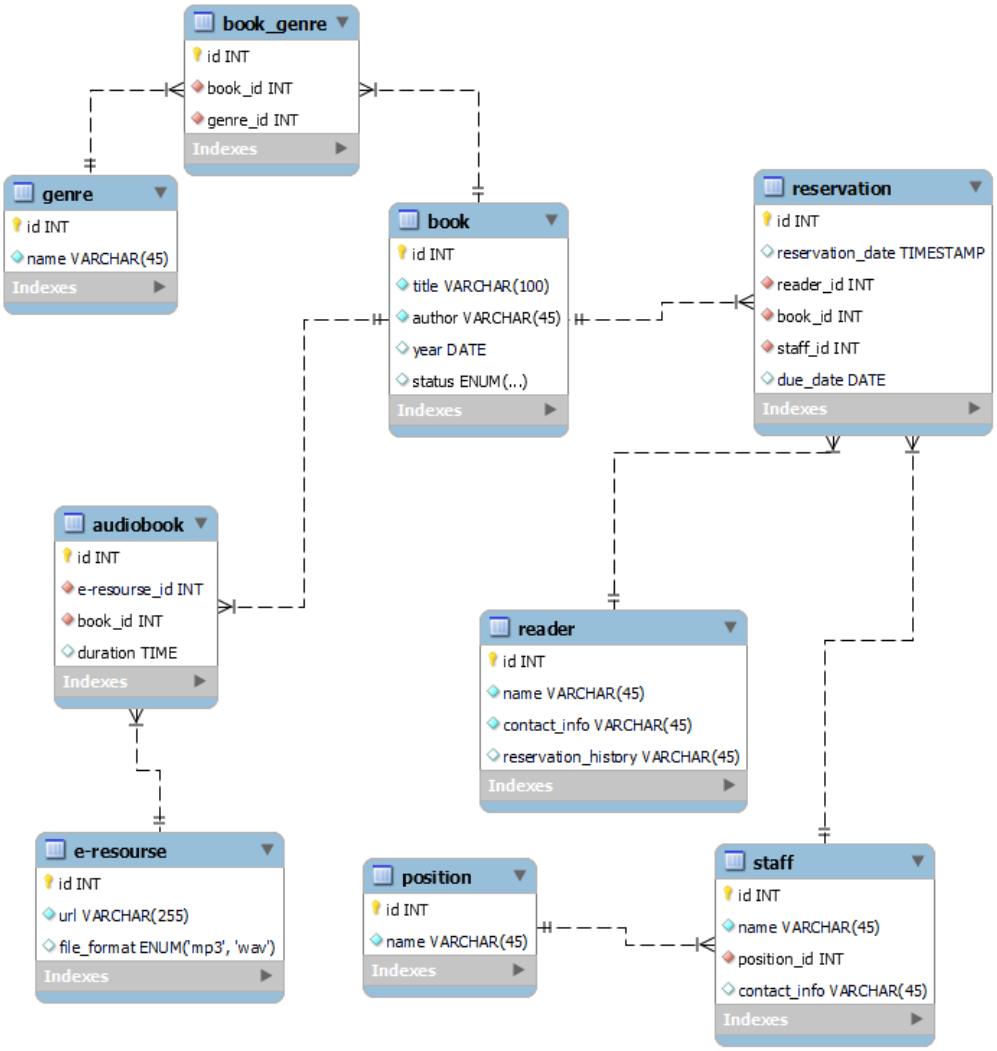
Объяснение выбранной СУБД

MySQL Workbench был выбран в качестве системы управления базами данных (СУБД) по следующим причинам:

1. Открытый исходный код: MySQL является бесплатной СУБД с открытым исходным кодом, что позволяет разработчикам настраивать и модифицировать систему по своему усмотрению.
2. Поддержка различных платформ: MySQL Workbench поддерживает широкий спектр операционных систем, включая Windows, Linux и MacOS. Это облегчает разработку и тестирование в различных средах.
3. Надежность и безопасность: MySQL известен своей надежностью и предлагает ряд функций безопасности, включая поддержку SSL и шифрование данных.
4. Производительность и масштабируемость: MySQL оптимизирован для высокой производительности и способен обрабатывать большие объемы данных, что делает его подходящим для больших проектов.
5. Интуитивный интерфейс: MySQL Workbench предлагает графический интерфейс пользователя, который упрощает создание, проектирование и управление базами данных.

В связи с этими преимуществами, мы решили использовать MySQL Workbench для реализации нашего проекта.

ER-диаграмма



Исходный текст запросов

```
-- MySQL Script generated by MySQL Workbench
-- Wed Jun 12 17:26:48 2024
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

```
-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema library_management_system
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema library_management_system
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `library_management_system` DEFAULT CHARACTER SET utf8mb3 ;
USE `library_management_system` ;

-- -----------------------------------------------------
-- Table `library_management_system`.`book`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `library_management_system`.`book` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(100) NOT NULL,
  `author` VARCHAR(45) NOT NULL,
  `year` DATE NULL DEFAULT NULL,
  `status` ENUM('available', 'unavailable') NULL DEFAULT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table `library_management_system`.`e-resourse`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `library_management_system`.`e-resourse` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `url` VARCHAR(255) NOT NULL,
  `file_format` ENUM('mp3', 'wav') NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `url_UNIQUE` (`url` ASC) VISIBLE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table `library_management_system`.`audiobook`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `library_management_system`.`audiobook` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `e-resourse_id` INT NOT NULL,
  `book_id` INT NOT NULL,
  `duration` TIME NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `e-resourse_id_idx` (`e-resourse_id` ASC) VISIBLE,
  INDEX `book_id_idx` (`book_id` ASC) VISIBLE,
  CONSTRAINT `book_audiobook_id`
    FOREIGN KEY (`book_id`)
    REFERENCES `library_management_system`.`book` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `e-resourse_id`
    FOREIGN KEY (`e-resourse_id`)
    REFERENCES `library_management_system`.`e-resourse` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table `library_management_system`.`genre`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `library_management_system`.`genre` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `name_UNIQUE` (`name` ASC) VISIBLE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table `library_management_system`.`position`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `library_management_system`.`position` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;
```

```
-- -----------------------------------------------------
-- Table `library_management_system`.`reader`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `library_management_system`.`reader` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `contact_info` VARCHAR(45) NOT NULL,
  `reservation_history` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table `library_management_system`.`staff`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `library_management_system`.`staff` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `position_id` INT NOT NULL,
  `contact_info` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `position_id_idx` (`position_id` ASC) VISIBLE,
  CONSTRAINT `position_id`
    FOREIGN KEY (`position_id`)
    REFERENCES `library_management_system`.`position` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table `library_management_system`.`reservation`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `library_management_system`.`reservation` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `reservation_date` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  `reader_id` INT NOT NULL,
  `book_id` INT NOT NULL,
  `staff_id` INT NOT NULL,
  `due_date` DATE NULL DEFAULT NULL,
```

```
  PRIMARY KEY (`id`),
  INDEX `reader_id_idx` (`reader_id` ASC) VISIBLE,
  INDEX `book_id_idx` (`book_id` ASC) VISIBLE,
  INDEX `staff_id_idx` (`staff_id` ASC) VISIBLE,
  CONSTRAINT `book_id`
    FOREIGN KEY (`book_id`)
    REFERENCES `library_management_system`.`book` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `reader_id`
    FOREIGN KEY (`reader_id`)
    REFERENCES `library_management_system`.`reader` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `staff_id`
    FOREIGN KEY (`staff_id`)
    REFERENCES `library_management_system`.`staff` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb3;


-- -----------------------------------------------------
-- Table `library_management_system`.`book_genre`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `library_management_system`.`book_genre` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `book_id` INT NOT NULL,
  `genre_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `book_genre_book_idx` (`book_id` ASC) VISIBLE,
  INDEX `book_genre_genre_idx` (`genre_id` ASC) VISIBLE,
  CONSTRAINT `book_genre_book`
    FOREIGN KEY (`book_id`)
    REFERENCES `library_management_system`.`book` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `book_genre_genre`
    FOREIGN KEY (`genre_id`)
    REFERENCES `library_management_system`.`genre` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Исходный код программы

Путь (для разработчика) - c:\Linuxoid\study\bdProject>

Config.py

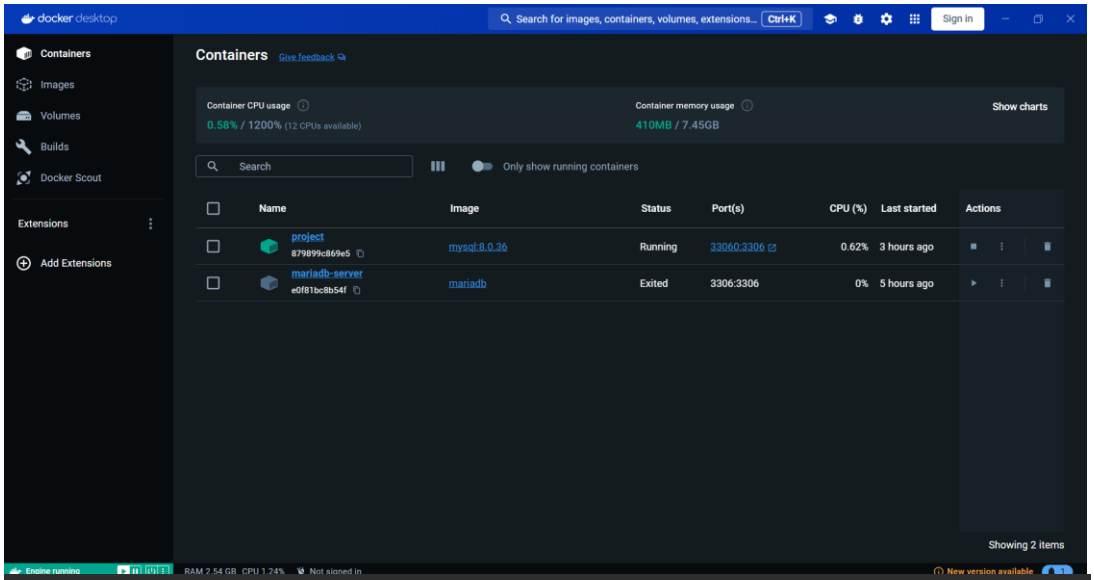
```
host = "127.0.0.1"
user = "root"
password = "my-secret-pw"
db_name = "library_management_system"
```

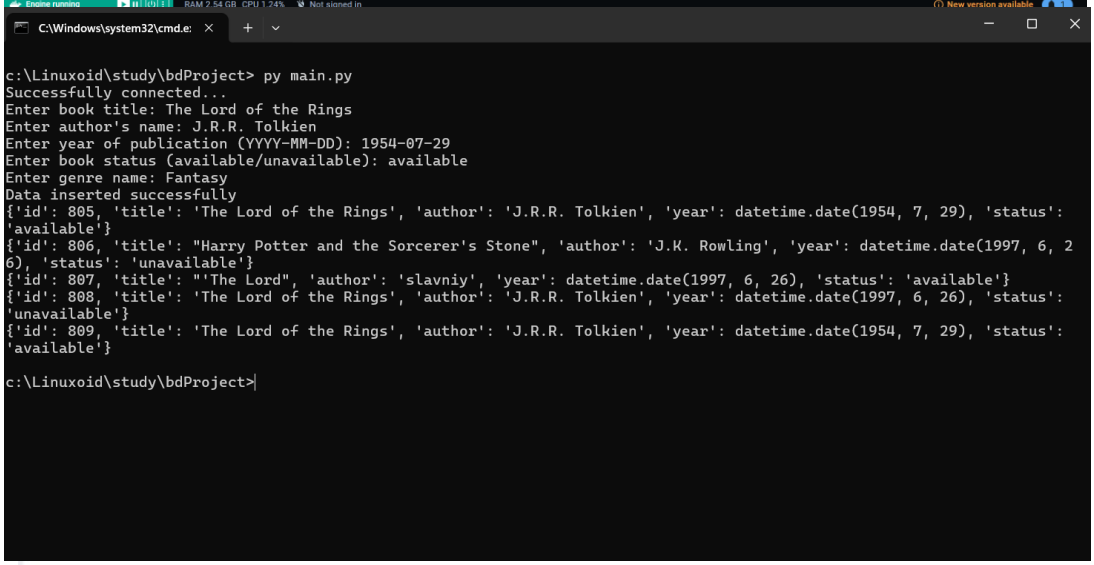
main.py

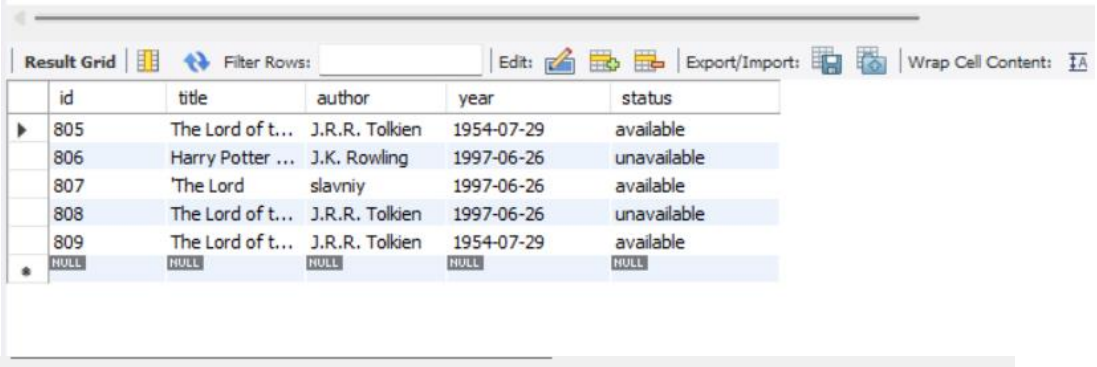
```
import pymysql
import pymysql.cursors
from config import host, user, password, db_name

try:
    connection = pymysql.connect(
        host=host,
        port=33060,
        user=user,
        password=password,
        database=db_name,
        cursorclass=pymysql.cursors.DictCursor
    )
    print("Successfully connected...")
except Exception as ex:
    print("Connection refused...")
    print(ex)

try:
    with connection.cursor() as cursor:
        # Ввод данных для таблицы book
        title = input("Enter book title: ")
```

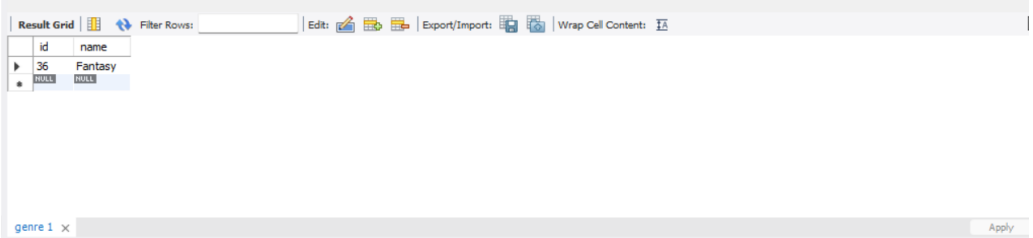
```
        author = input("Enter author's name: ")
        year = input("Enter year of publication (YYYY-MM-DD): ")
        status = input("Enter book status (available/unavailable): ")

        insert_book_query = "INSERT INTO library_management_system.book (title,
author, year, status) VALUES (%s, %s, %s, %s)"
        cursor.execute(insert_book_query, (title, author, year, status))
        book_id = cursor.lastrowid  # Получаем ID только что вставленной книги

        # Ввод данных для таблицы genre
        genre_name = input("Enter genre name: ")
        select_genre_query = "SELECT id FROM library_management_system.genre
WHERE name = %s"
        cursor.execute(select_genre_query, (genre_name,))
        result = cursor.fetchone()
        if result:
            genre_id = result['id']
        else:
            insert_genre_query = "INSERT INTO library_management_system.genre
(name) VALUES (%s)"
            cursor.execute(insert_genre_query, (genre_name,))
            genre_id = cursor.lastrowid  # Получаем ID только что вставленного
жанра

        # Ввод данных для таблицы book_genre
        insert_book_genre_query = "INSERT INTO
library_management_system.book_genre (book_id, genre_id) VALUES (%s, %s)"
        cursor.execute(insert_book_genre_query, (book_id, genre_id))

        connection.commit()
        print("Data inserted successfully")

        # Выбор всех данных из таблицы book
        select_book_query = "SELECT * FROM library_management_system.book"
        cursor.execute(select_book_query)
        result = cursor.fetchall()
        for row in result:
            print(row)

except Exception as ex:
    print("Failed to insert data into database")
    print(ex)
finally:
    connection.close()
```

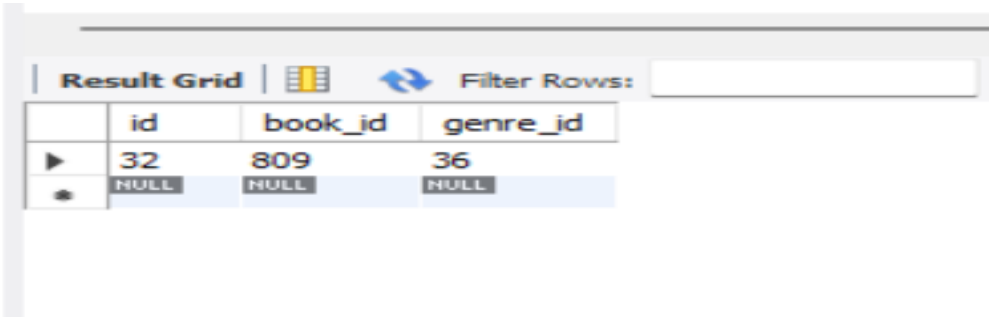
Результат программы:


docker desktop
Search for images, containers, volumes, extensions... Ctrl+K
Sign in
Containers
Images
Volumes
Builds
Docker Scout
Extensions
Add Extensions
Containers
Give feedback
Container CPU usage
0.58% / 1200% (12 CPUs available)
Container memory usage
410MB / 7.45GB
Show charts
Search
Only show running containers
Name
Image
Status
Port(s)
CPU (%)
Last started
Actions
project
879899c869e5
mysql:8.0.36
Running
33060:3306
0.62%
3 hours ago
mariadb-server
e0f81bc8b54f
mariadb
Exited
3306:3306
0%
5 hours ago
Showing 2 items


C:\Windows\system32\cmd.e:
c:\Linuxoid\study\bdProject> py main.py
Successfully connected...
Enter book title: The Lord of the Rings
Enter author's name: J.R.R. Tolkien
Enter year of publication (YYYY-MM-DD): 1954-07-29
Enter book status (available/unavailable): available
Enter genre name: Fantasy
Data inserted successfully
{'id': 805, 'title': 'The Lord of the Rings', 'author': 'J.R.R. Tolkien', 'year': datetime.date(1954, 7, 29), 'status': 'available'}
{'id': 806, 'title': "Harry Potter and the Sorcerer's Stone", 'author': 'J.K. Rowling', 'year': datetime.date(1997, 6, 26), 'status': 'unavailable'}
{'id': 807, 'title': "'The Lord", 'author': 'slavniy', 'year': datetime.date(1997, 6, 26), 'status': 'available'}
{'id': 808, 'title': 'The Lord of the Rings', 'author': 'J.R.R. Tolkien', 'year': datetime.date(1997, 6, 26), 'status': 'unavailable'}
{'id': 809, 'title': 'The Lord of the Rings', 'author': 'J.R.R. Tolkien', 'year': datetime.date(1954, 7, 29), 'status': 'available'}
c:\Linuxoid\study\bdProject>


Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
id
title
author
year
status
805
The Lord of t...
J.R.R. Tolkien
1954-07-29
available
806
Harry Potter ...
J.K. Rowling
1997-06-26
unavailable
807
'The Lord
slavniy
1997-06-26
available
808
The Lord of t...
J.R.R. Tolkien
1997-06-26
unavailable
809
The Lord of t...
J.R.R. Tolkien
1954-07-29
available
NULL


Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
id
name
36
Fantasy
NULL
genre 1
Apply


Result Grid
Filter Rows:
id
book_id
genre_id
32
809
36
NULL
NULL
NULL