НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського» ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування та спеціалізованих комп’ютерних систем

**Лабораторна робота №2**

з дисципліни

# «Бази даних і засоби управління»

Тема: «Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»

Виконала: студентка III курсу ФПМ

групи КВ-04

Ільїнчик А.О.

Перевірив: доц. Петрашенко А. В.

Київ – 2023

*Мета роботи:* здобуття вмінь програмування прикладних додатків баз данних PostgreSQL.

*Загальне завдання роботи полягає у наступному*:

1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/Fаlse, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

*Деталізоване завдання*:

1. Забезпечити можливість уведення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідність типів даних атрибутів таблиць (рядків, чисел, дати/часу). Для контролю пропонується два варіанти: контроль при введенні (валідація даних) та перехоплення помилок (try..except) від сервера PostgreSQL при виконанні відповідної команди SQL. Особливу увагу варто звернути на дані таблиць, що мають зв’язок 1:N. При цьому з боку батьківської таблиці необхідно контролювати **вилучення** рядків за умови наявності даних у підлеглій таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідності рядка у батьківській таблиці при виконанні **внесення** нових даних. Унеможливити виведення програмою системних помилок на екрані шляхом їх перехоплення і адекватної обробки. Внесення даних виконується у консольному вікні програми.
2. Забезпечити можливість автоматичної генерації великої кількості даних у таблицях за допомогою вбудованих у PostgreSQL функцій роботи з псевдовипадковими числами. Дані мають бути згенерованими **не мовою програмування, а відповідним SQL-запитом**!
3. Для реалізації пошуку необхідно підготувати 3 запити, включають дані з декількох таблиць і фільтрують рядки за 3-4 атрибутами цих таблиць. Забезпечити можливість введення конкретних значень констант для фільтрації з клавіатури користувачем. Крім того, після виведення даних необхідно вивести час виконання запиту у

мілісекундах. Перевірити швидкодію роботи запитів на попередньо згенерованих даних.

1. Програмний код організувати згідно шаблону Model-View-Controller(MVC). При цьому модель, подання та контролер мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** (без ORM).

# Інформація про модель та структуру бази даних

# 

Рис. 1 - Концептуальна модель предметної області “Облік книгозбірні” Нижче (Рис. 2) наведено логічну модель бази даних:

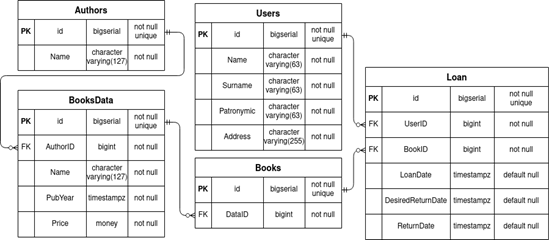


Рис. 2 – Логічна модель бази даних

Зміни у порівнянні з першою лабораторною роботою відсутні.

# Середовище розробки та налаштування підключення до бази даних

Для виконання лабораторної роботи використовувалась мова програмування Python та текстовий редактор Sublime Text 3.

Для підключення до серверу бази даних PostgreSQL використано модуль «psycopg2».

# Опис структури програми

Програма містить 5 основних модулів: **Lab, model, view, controller, utils**. Файл для запуску - «lab2.py3».

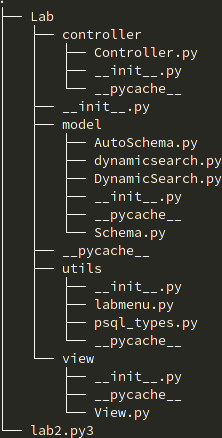
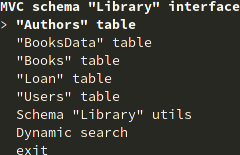


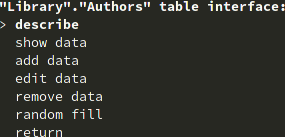
Рис. 3 – Структура програмного коду

# Структура меню програми

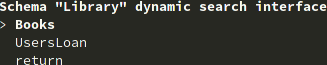
Головне меню

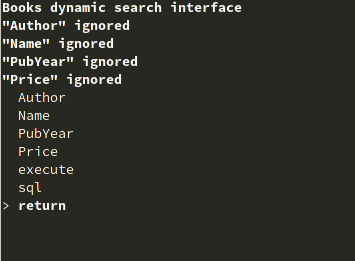


Меню для таблиці



Меню для вибору динамічних запитів





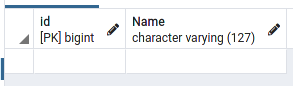
Меню вибору кількості рядків для генерації

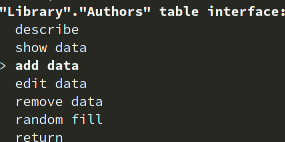


# Пункт 1

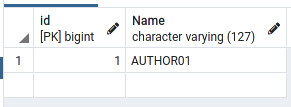
Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.

# Внесення даних

Створення нового автора:

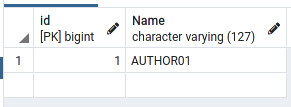






# Видалення даних

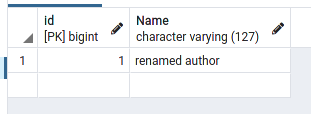




Якщо уведено неіснуючий id рядку:

При видаленні рядку програма завжди використовує ключове слово “CASCADE” мови SQL. Ключове слово “CASCADE” дає дозвіл СУБД автоматично видаляти залежні рядки в дочірній таблиці, коли відповідні рядки видаляються в батьківській таблиці.

Редагування даних

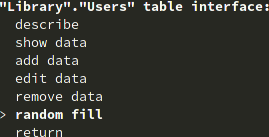


# Пункт 2

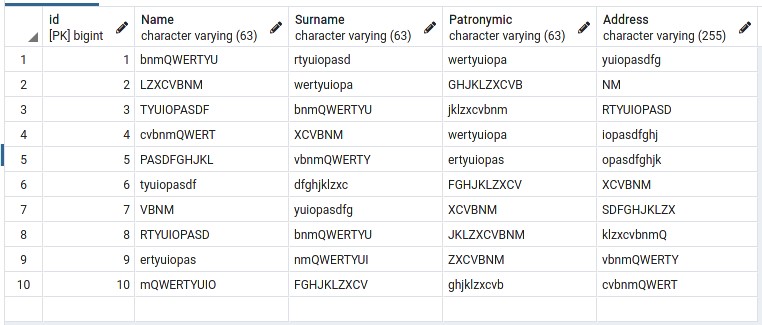
Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.

У програмі передбачено рандомізоване заповнення кожної таблиці окремо(з вказаннями кількості рандомізованих рядків для генерації), та пакетне рандомізоване заповнення таблиць схеми(без можливості зміни кількості рандомізованих рядків користувачем).

# Рандомізоване заповнення таблиці “Users”:





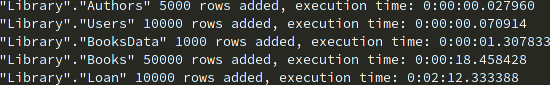


**Пакетне заповнення таблиць схеми:**

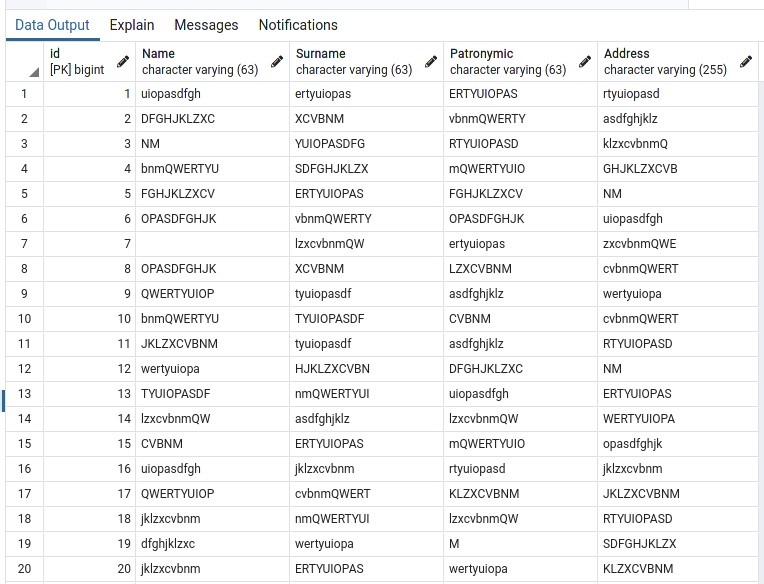
Кількість заданих рядків пакетного заповнення таблиць схеми:

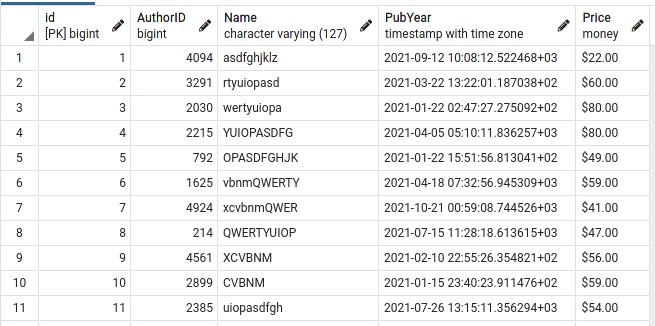
|  |  |
| --- | --- |
| Authors | 5000 |
| Users | 10000 |
| BooksData | 1000 |

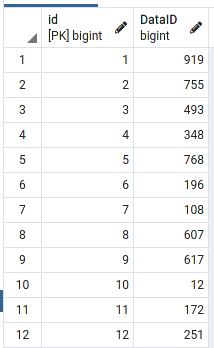
|  |  |
| --- | --- |
| Books | 50000 |
| Loan | 10000 |

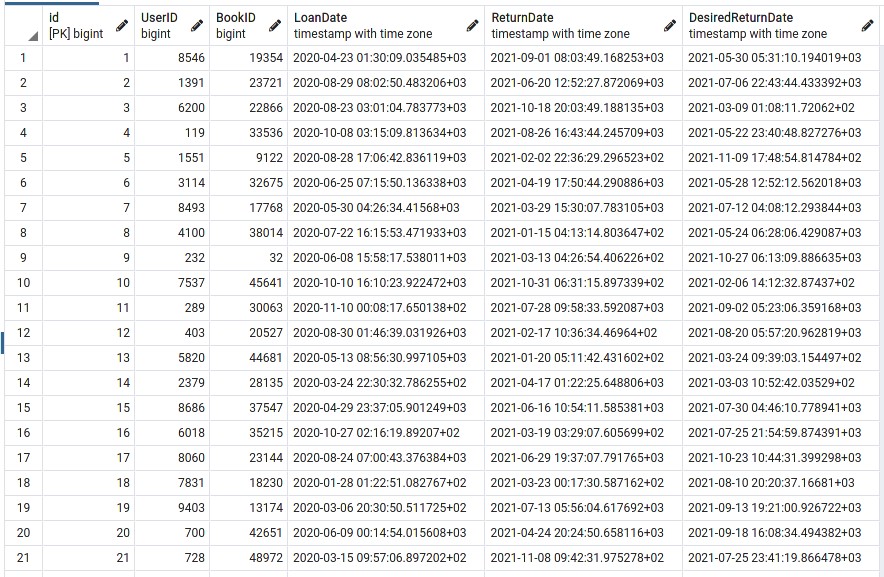


**Authors**

**Users**

**BooksData**

**Books**

**Loan**

**SQL запити рандомізованого заповнення:**

**INSERT INTO** "Library"."Authors"("Name")

**SELECT**

substr(characters, (random() \* length(characters) + 1)::**integer**, 10)

**FROM**

(**VALUES**('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) **as**

symbols(characters), generate\_series(1, 5000) **as** q;

**INSERT INTO** "Library"."Users"("Name", "Surname", "Patronymic", "Address")

**SELECT**

substr(characters, (random() \* length(characters) + 1)::**integer**, 10), substr(characters, (random() \* length(characters) + 1)::**integer**, 10), substr(characters, (random() \* length(characters) + 1)::**integer**, 10), substr(characters, (random() \* length(characters) + 1)::**integer**, 10)

**FROM**

(**VALUES**('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) **as**

symbols(characters), generate\_series(1, 10000) **as** q;

**INSERT INTO** "Library"."BooksData"("AuthorID", "Name", "PubYear", "Price")

**SELECT**

(**SELECT** "id" **FROM** "Library"."Authors" **ORDER BY** random()\*q **LIMIT** 1), substr(characters, (random() \* length(characters) + 1)::**integer**, 10),

timestamp '2021-01-01' + random() \* (timestamp '2021-11-11' - timestamp '2021-01-01'), trunc(random() \* 100)::int

**FROM**

(**VALUES**('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) **as**

symbols(characters), generate\_series(1, 1000) **as** q;

**INSERT INTO** "Library"."Books"("DataID")

**SELECT**

(**SELECT** "id" **FROM** "Library"."BooksData" **ORDER BY** random()\*q **LIMIT** 1)

**FROM**

(**VALUES**('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) **as**

symbols(characters), generate\_series(1, 50000) **as** q;

**INSERT INTO** "Library"."Loan"("UserID", "BookID", "LoanDate", "ReturnDate", "DesiredReturnDate")

**SELECT**

(**SELECT** "id" **FROM** "Library"."Users" **ORDER BY** random()\*q **LIMIT** 1), (**SELECT** "id" **FROM** "Library"."Books" **ORDER BY** random()\*q **LIMIT** 1),

timestamp '2020-01-01' + random() \* (timestamp '2020-11-11' - timestamp '2020-01-01'), timestamp '2021-01-01' + random() \* (timestamp '2021-11-11' - timestamp '2021-01-01'), timestamp '2021-01-01' + random() \* (timestamp '2021-11-11' - timestamp '2021-01-01')

**FROM**

(**VALUES**('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) **as**

symbols(characters), generate\_series(1, 10000) **as** q;

# Пункт 3

Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.

Було підготовлено два SQL запити:

* Пошук екземплярів книги за атрибутами:
  + Ім’я автора
  + Назва книги
  + Рік видання
  + Ціна
* Пошук користувачів та книг, що вони запозичили за наступними

атрибутами:

* + ім’я користувача
  + прізвище
  + по батькові
  + адреса проживання
  + дата видачі книги
  + дата коли книгу було повернено
  + дата до якої потрібно повернути книгу
  + повна назва книги

Пошук екземплярів книги:

SQL запит без фільтрації рядків:

**SELECT**

"c"."Name" **as** "Author",

"b"."Name" **as** "Name",

"b"."PubYear" **as** "PubYear",

"b"."Price" **as** "Price"

**FROM**

"Library"."Books" **as** "a"

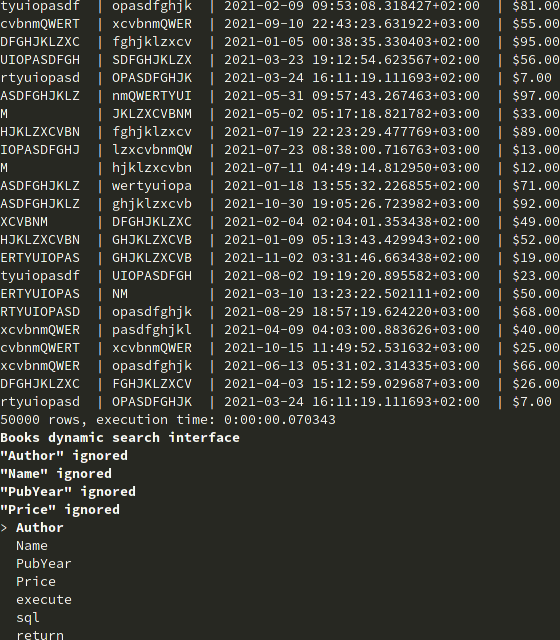
**INNER JOIN** "Library"."BooksData" **as** "b"

**ON** "a"."DataID" = "b"."id"

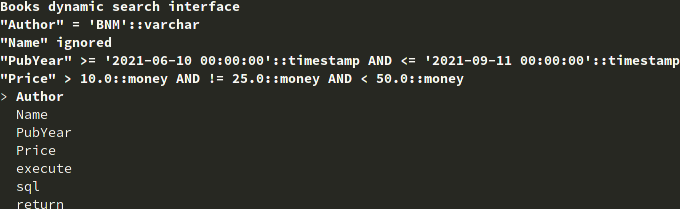
**INNER JOIN** "Library"."Authors" **as** "c"

**ON** "b"."AuthorID" = "c"."id"

;

Результат:

Налаштування фільтрування рядків:

* ім’я автора = ‘BNM’
* ціна книги > 10.00
* ціна книги != 25.00
* ціна книги < 50.00
* дата видання книги >= 2021-06-10
* дата видання книги <= 2021-09-11 Введені критерії фільтрування:

SQL запит з заданими налаштуваннями фільтрування рядків:

**SELECT**

"c"."Name" **as** "Author",

"b"."Name" **as** "Name",

"b"."PubYear" **as** "PubYear",

"b"."Price" **as** "Price"

**FROM**

"Library"."Books" **as** "a"

**INNER JOIN** "Library"."BooksData" **as** "b"

**ON** "a"."DataID" = "b"."id"

**INNER JOIN** "Library"."Authors" **as** "c"

**ON** "b"."AuthorID" = "c"."id"

**WHERE**

("c"."Name" = 'BNM'::varchar) **AND**

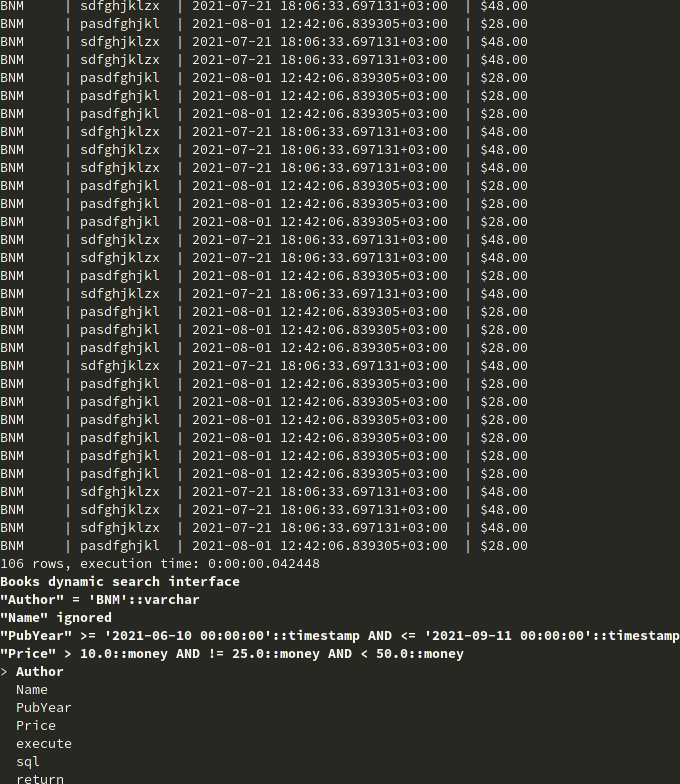
("b"."PubYear" >= '2021-06-10 00:00:00'::timestamp **AND** "b"."PubYear" <= '2021-09-11

00:00:00'::timestamp) **AND**

("b"."Price" > 10.0::money **AND** "b"."Price" != 25.0::money **AND** "b"."Price" < 50.0::money);

Результат:

...



Пошук користувачів та книг, що вони запозичили: SQL запит без фільтрації рядків:

**SELECT**

"a"."Name" **as** "Name",

"a"."Surname" **as** "Surname", "a"."Patronymic" **as** "Patronymic", "a"."Address" **as** "Address",

"b"."LoanDate" **as** "LoanDate", "b"."ReturnDate" **as** "ReturnDate",

"b"."DesiredReturnDate" **as** "DesiredReturnDate", "d"."Name" **as** "BookName"

**FROM**

"Library"."Users" **as** "a"

**INNER JOIN** "Library"."Loan" **as** "b"

**ON** "a"."id" = "b"."UserID"

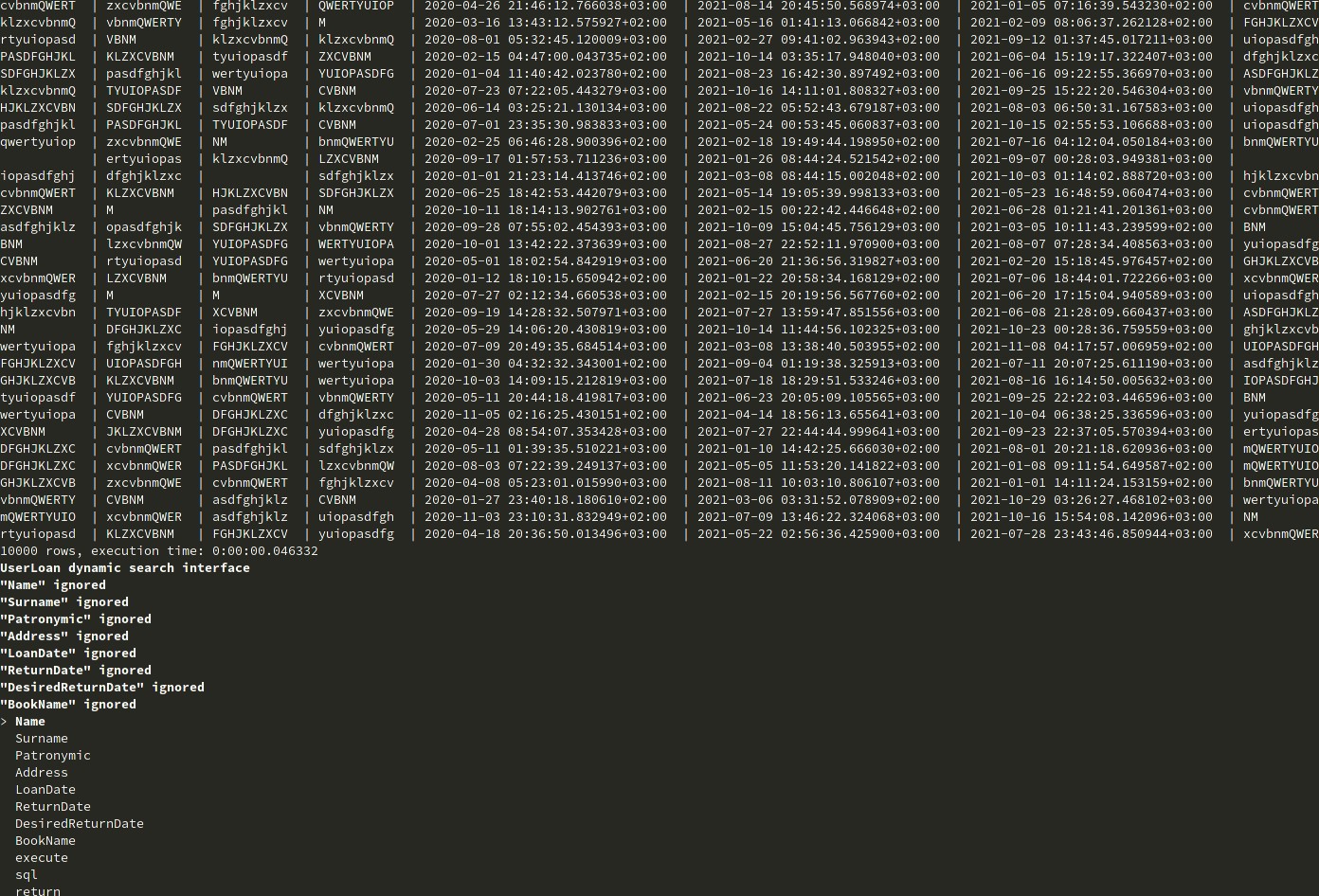
**INNER JOIN** "Library"."Books" **as** "c"

**ON** "b"."BookID" = "c"."id"

**INNER JOIN** "Library"."BooksData" **as** "d"

**ON** "c"."DataID" = "d"."id"

;

Результат