



Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут»

Факультет прикладної математики  
Кафедра системного програмування і спеціалізованих комп'ютерних  
систем

**Розрахунково-графічна робота**  
*з дисципліни*  
**«Бази даних та засоби управління»**

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

Виконав студент 3 курсу

ФПМ групи КВ-11

Парієнко Віктор

**Київ – 2023**

*Метою роботи є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL*

*Загальне завдання роботи полягає у наступному:*

1. Реалізувати функції перегляду, 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/False, для дат - у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер)

*GitHub репозиторій: [https://github.com/SkaLe3/KPI\\_class\\_DataBase\\_Labs](https://github.com/SkaLe3/KPI_class_DataBase_Labs)*

*Контакт в Telegram: <https://t.me/SkaLe9> (@SkaLe9)*

## Модель “сутність-зв’язок” предметної галузі “Музичний каталог для зберігання даних про виконавців та пісні”

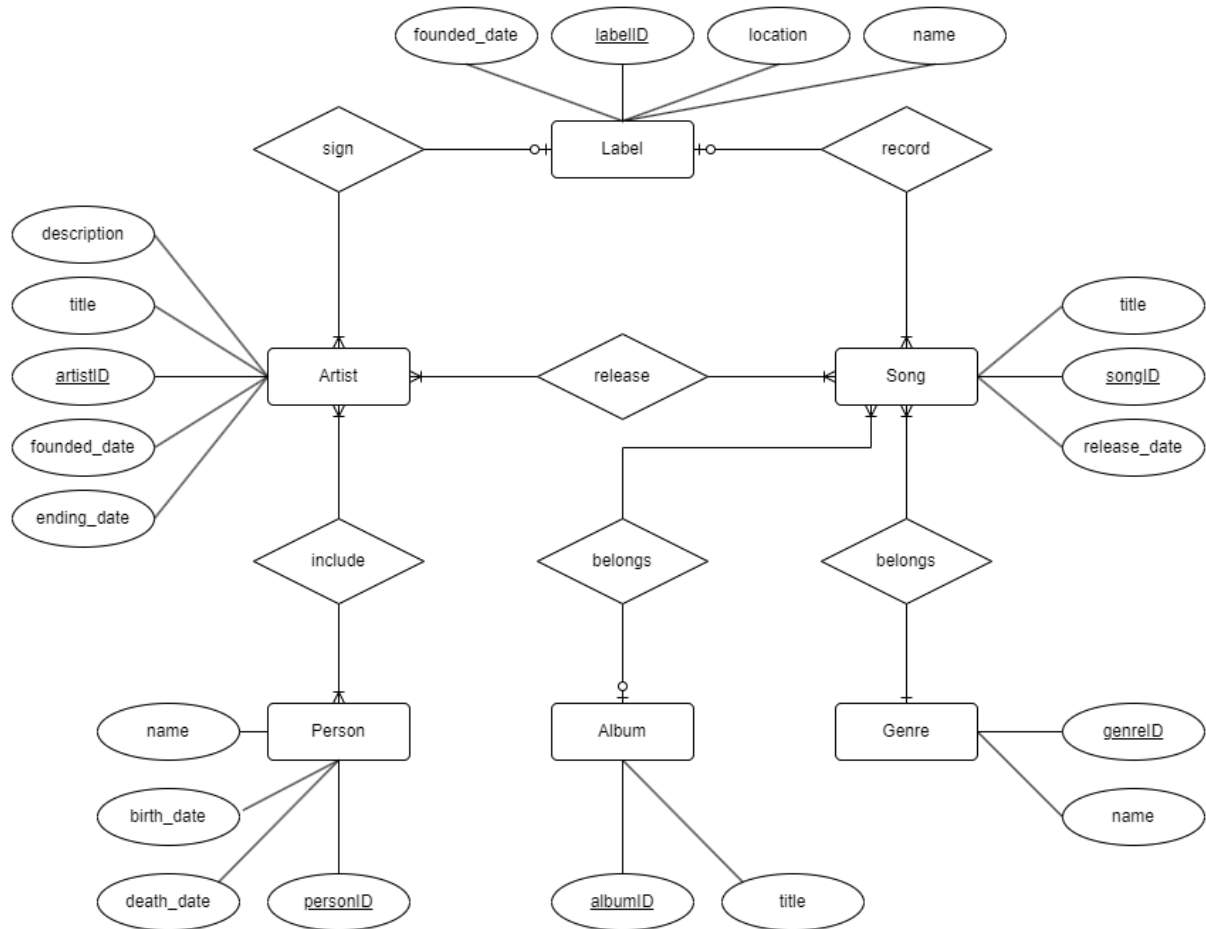


Рис 1. ER-діаграма з використанням нотації “Crow’s foot”

### Короткий опис бази даних

#### Перелік сутностей і їх атрибутів:

бази даних в предметній галузі “Музичний каталог” включає 6 сутностей з наступними атрибутами:

- **Artist**
  - artist\_id, title, description, founded\_date, closed\_date
- **Person**
  - person\_id, name, birth\_date, death\_date
- **Label**
  - label\_id, founded\_date, location
- **Song**
  - song\_id, title, release\_date
- **Genre**
  - genre\_id, name
- **Album**
  - album\_id, title

### Опис сутностей:

Artist - сутність для представлення музичного проєкту що може бути групою або однією людиною. Причому одна людина може мати декілька сольних проєктів. Кожен такий проєкт має id, назву, опис, дату створення, і дату закінчення.

Person - сутність для представлення людини. Людина є учасником певного музичного проєкту, або декількох. Атрибути людини це ID, ім'я, дата народження, дата смерті.

Label - сутність для представлення звукозаписуючої компанії, що є брендом а також має право власності на пісні і співпрацює з авторами. Містить id, назву, локацію, дату створення

Song - сутність для представлення пісні яку випускають виконавці, містить id, назву, дату випуску

Genre - сутність для представлення жанру пісні. Складається лише з 2 атрибутів id і назва

Album - сутність для представлення альбому з піснями. Має id і назву.

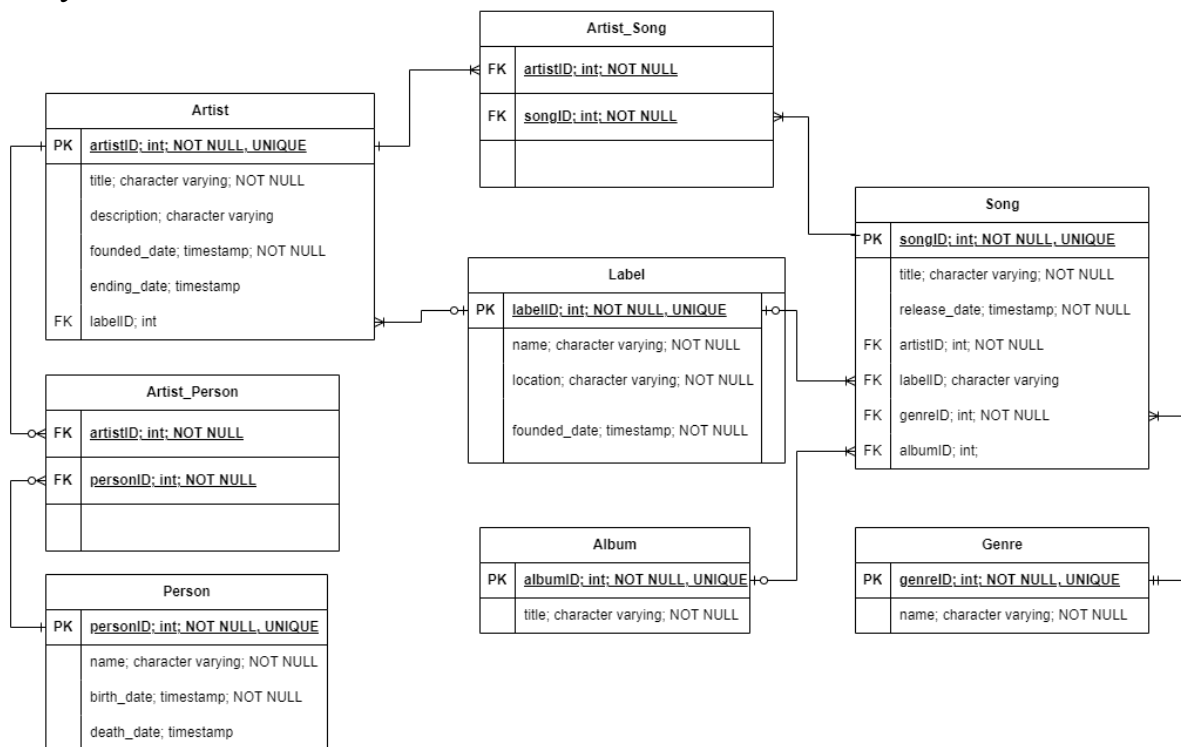


Рис 2. Схема бази даних

*Для створення застосунку було використано:*

Мова програмування: C++ 20

Середовище розробки: Visual Studio 2022

Бібліотеки:

- GLFW - для створення вікна і прийняття введення
- Glad - для завантаження вказівників на функції OpenGL
- VieM - власна бібліотека, на базі Walnut, для основи програми
- ImGui - для створення графічного інтерфейсу
- libpq - для взаємодії з PostgreSQL

# Схема меню користувача

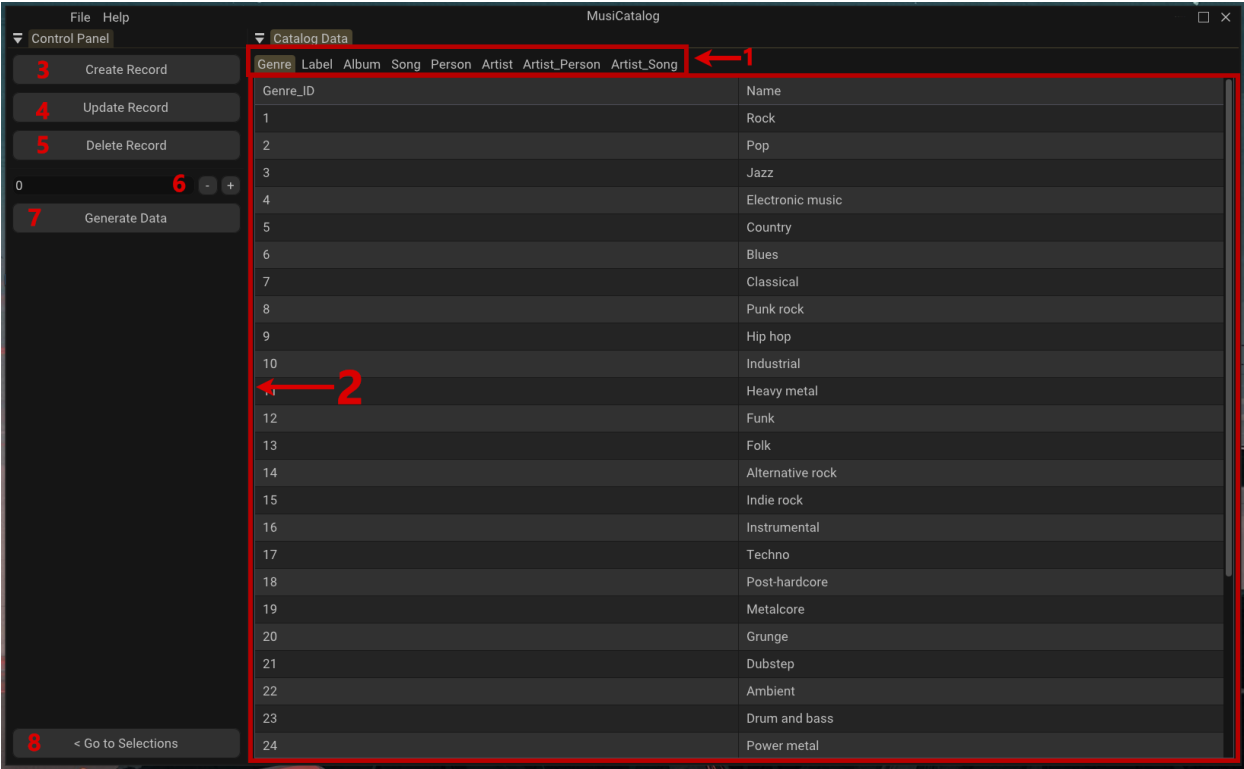


Рис 3. Меню управління даними

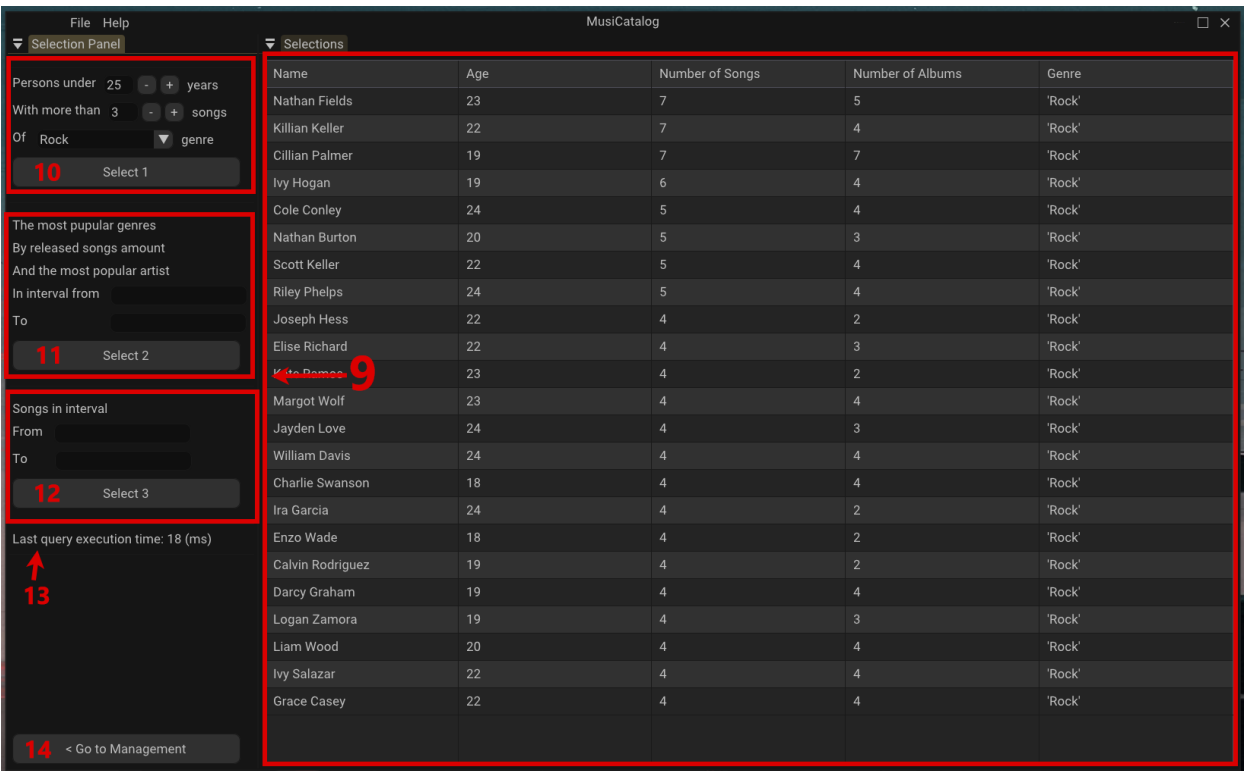


Рис 4. Меню вибору даних

## Опис функціоналу:

1. Tabs - Вкладки з таблицями для відображення даних
2. Table - Результат виведення вибраної таблиці з даними (виконується при переключенні між вкладками)

Для наступних елементів управління, взаємодія відбувається з даними таблиць з вибраної вкладки

3. Create Record - Кнопка для внесення даних:

При натисненні цієї кнопки з'являється наступне меню для внесення даних.

The first screenshot shows a dialog box titled 'Add Record' with a close button (X). It contains two input fields: 'Genre\_ID' with a dropdown menu showing 'Default', and 'Name'. Below these fields is a 'Submit' button.

The second screenshot shows a more complex 'Add Record' dialog box. It has a table with six columns: 'Song\_ID', 'Title', 'Release\_Date', 'Label\_ID', 'Genre\_ID', and 'Album\_ID'. Each column has a 'Default' value and a small '+' button next to it. The values are: 'new song', '2023-11-12', '0', '0', and '0'. Below the table is a 'Submit' button.

Користувач вносить дані у відведені поля. (Дата підсвічується зеленим, якщо формат дати введено вірно, червоним - не вірно). Після внесення даних користувач натискає кнопку Submit

4. Update Record - Кнопка редагування даних:

При натисненні цієї кнопки з'являється наступне меню для пошуку запису

The 'Update Record' dialog box has a title bar with a close button (X). Below the title bar is a label 'Find Record To Update'. There is an input field for 'Genre\_ID' with a dropdown menu showing '0'. Below this field is a 'Find Record' button.

Користувач вносить дані за якими можна знайти запис у таблиці і натискає кнопку Find Record.

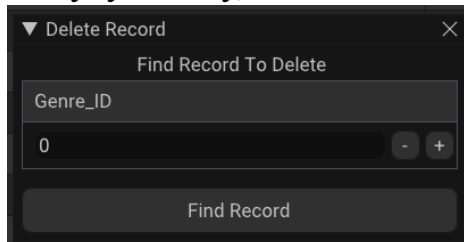
Якщо шуканий запис існує, з'являється наступне меню, аналогічне до меню внесення даних.

The 'Update Record' dialog box shows the 'Genre\_ID' dropdown menu with 'Folk' selected. Below the table is a 'Record Found' message and a 'Submit' button.

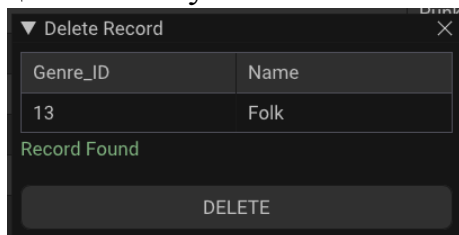
Користувач редагує дані і натискає кнопку Submit для їх занесення до таблиці.

5. Delete Record - Кнопка видалення даних:

При натисненні цієї кнопки з'являється наступне меню для пошуку запису, аналогічне меню з попереднього пункту



Після успішного знаходження запису з'являється меню з даними цього запису і кнопкою DELETE



6. Generate amount - Поле для вводу кількості записів для генерації:

(для таблиць Genre і Label кількість генерованих даних стала)

7. Generate Data - Кнопка для генерації даних:

При натисненні кнопки відбувається генерація вказаної кількості записів для вибраної таблиці.

8. Go to Selections - Кнопка для переходу на вікно для вибору даних:

Наступні пункти описують елементи управління цього вікна

9. Search Table - Результат виконання запиту пошуку

10. Area 1 - Область першого запиту:

Користувач вводить максимальний вік людини, мінімальну кількість пісень і бажаний жанр. Після чого натискає кнопку Select 1 і результат пошуку з'являється в таблиці праворуч.

11. Area 2 - Область другого запиту:

Користувач вводить початок і кінець часового інтервалу у вигляді двох дат для пошуку даних.

12. Area 3 - Область третього запиту:

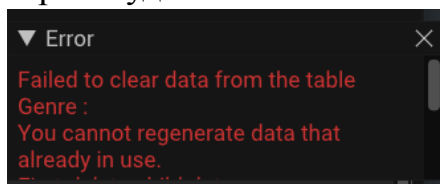
Користувач вводить початок і кінець часового інтервалу у вигляді двох дат для пошуку даних.

13. Search Time - Час виконання останнього запиту пошуку даних, у мілісекундах

14. Go to Management - Кнопка для повернення в меню управління даними.



У разі будь-якої помилки виводиться наступне вікно



(в цьому випадку була спроба регенерувати таблицю з жанрами, коли існують пісні з цими жанрами)

## Завдання 1

### *Видалення даних*

Результат виконання операції видалення записів з таблиці Label при наявних залежних даних в таблиці Song, з режимом каскадного видалення

File Help

Control Panel

Catalog Data

Genre Label Album Song Person Artist Artist\_Person Artist\_Song

Song_ID	Title	Release_Date	Label_ID	Genre_ID	Album_ID
1	way how be home	2014-05-14	5	15	267
2	lonely memory tidal impos	2001-05-02		22	296
3	I oblivion left	1990-10-08	2	8	409
4	you innocent last	2014-03-11		6	445
5	last foxglove alive I fight	2014-09-16		24	492
6	negative the fool life adrena	1997-01-08	13	27	531
7	light you just	2019-12-04		11	659
8	you at fire hate	1997-10-09		2	742
9	king memories old	1985-09-08	3	11	915
10	I of dark I	2005-06-05	12	25	988

10

Generate Data

Last data generation time: 6490 (ms)

< Go to Selections

File Help

Control Panel

Catalog Data

Genre Label Album Song Person Artist Artist\_Person Artist\_Song

Song_ID	Title	Release_Date	Label_ID	Genre_ID	Album_ID
2	lonely memory tidal impos	2001-05-02		22	296
4	you innocent last	2014-03-11		6	445
5	last foxglove alive I fight	2014-09-16		24	492
7	light you just	2019-12-04		11	659
8	you at fire hate	1997-10-09		2	742

10

Generate Data

Last data generation time: 32 (ms)

< Go to Selections

Результат виконання операції вилучення записів таблиці Genre при наявних залежних даних в таблиці Song, з режимом заборони видалення

File Help

▼ Control Panel

Create Record

Update Record

Delete Record

10

Generate Data

Last data generation time: 32 (ms)

▼ Catalog Data

Genre	Label	Album	Song	Person	Artist	Artist_Person	Artist_Song		
Song_ID			Title			Release_Date	Label_ID	Genre_ID	Album_ID
2			lonely memory tidal impos			2001-05-02		22	296
4			you innocent last			2014-03-11		6	445
5			last foxglove alive I fight			2014-09-16		24	492
7			light you just			2019-12-04		11	659
8			you at fire hate			1997-10-09		2	742

< Go to Selections

File Help

▼ Control Panel

Create Record

Update Record

Delete Record

10

Generate Data

▼ Catalog Data

Genre	Label	Album	Song	Person	Artist	Artist_Person	Artist_Song		
Song_ID			Title			Release_Date	Label_ID	Genre_ID	Album_ID
2			lonely memory tidal impos			2001-05-02		22	296
4			you innocent last			2014-03-11		6	445
5			last foxglove alive i fight			2014-09-16		24	492
7			light you just			2019-12-04		11	659
8			you at fire hate			1997-10-09		2	742

▼ Error

Failed to clear data from the table Genre : You cannot regenerate data that already in use.

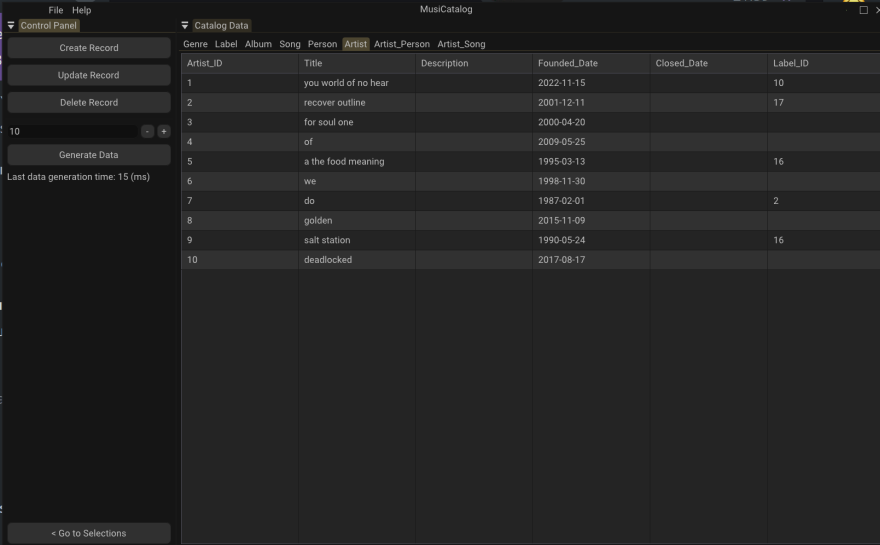
(Завуваження: було використано генерацію даних, для чого спочатку всі наявні дані таблиці видаляються. Через що в повідомленні помилки сказано перегенерація даних, а не видалення, бо технічно була натиснена кнопка генерації даних)

▼ Error

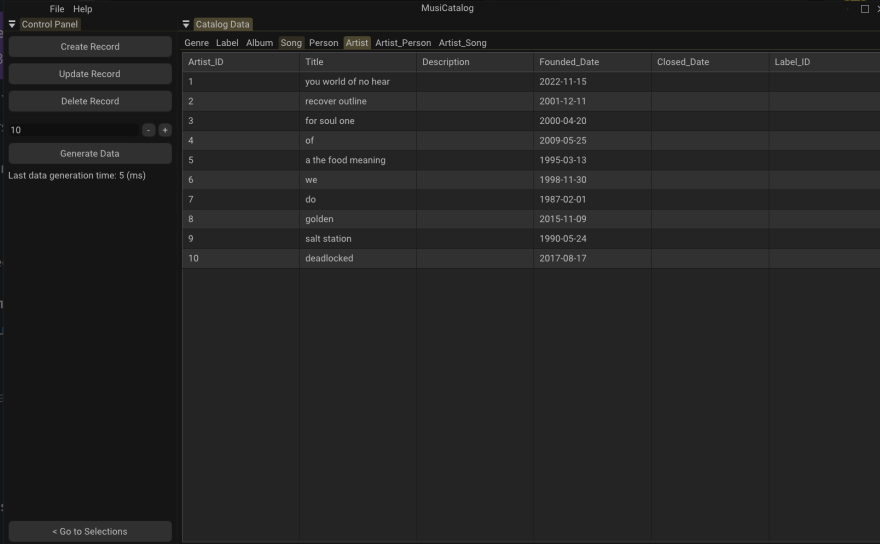
Delete Record Query execution failed for table: Genre Deletion is not possible if there is dependent data

(Видалення)

Результат виконання операції вилучення записів таблиці Genre при наявних залежних даних в таблиці Song, з режимом встановлення в NULL при видаленні



Artist_ID	Title	Description	Founded_Date	Closed_Date	Label_ID
1	you world of no hear		2022-11-15		10
2	recover outline		2001-12-11		17
3	for soul one		2000-04-20		
4	of		2009-05-25		
5	a the food meaning		1995-03-13		16
6	we		1998-11-30		
7	do		1987-02-01		2
8	golden		2015-11-09		
9	salt station		1990-05-24		16
10	deadlocked		2017-08-17		



Artist_ID	Title	Description	Founded_Date	Closed_Date	Label_ID
1	you world of no hear		2022-11-15		
2	recover outline		2001-12-11		
3	for soul one		2000-04-20		
4	of		2009-05-25		
5	a the food meaning		1995-03-13		
6	we		1998-11-30		
7	do		1987-02-01		
8	golden		2015-11-09		
9	salt station		1990-05-24		
10	deadlocked		2017-08-17		

*Листинг*

```
bool Model::DeleteRecord(Table table, std::vector<Column> pkeyColumns,
std::vector<std::string> pkeysData, std::string& errorMessage)
{
    std::string query = "DELETE FROM " + TableSpecs::GetName(table) + "
WHERE ";

    for (size_t i = 0; i < pkeyColumns.size(); ++i)
    {
        if (i > 0)
            query += " AND ";
    }
}
```

```

        query += pkeyColumns[i].Name + " = ";
        query += "'" + pkeysData[i] + "'";
    }
    query += ";";

    PGresult* res = PQexec(m_Connection, query.c_str());

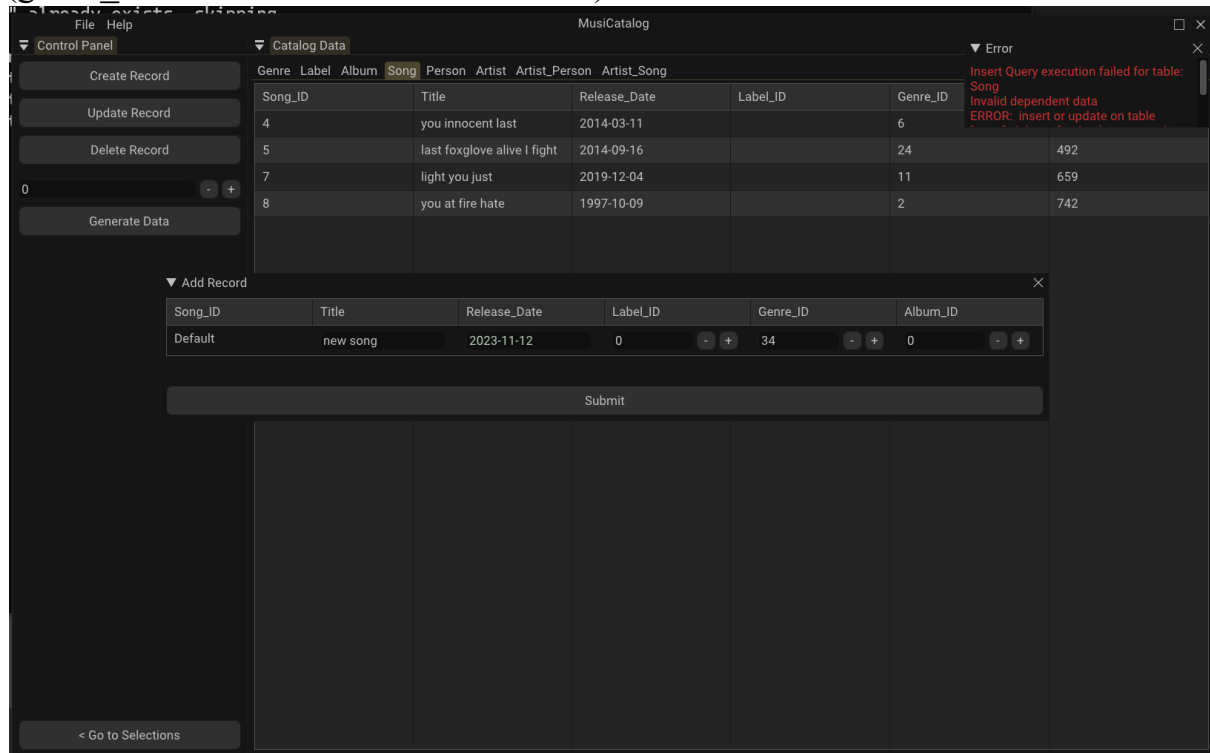
    if (PQresultStatus(res) != PGRES_COMMAND_OK)
    {
        errorMessage = std::string("Delete Record Query execution failed for
table: ") + TableSpecs::GetName(table) + "\n"
        "Deletion is not possible if there is dependent data\n"+
PQerrorMessage(m_Connection);
        VM_ERROR(errorMessage);
        VM_ERROR("Query text was: ", query, "\n");
        PQclear(res);
        return false;
    }

    PQclear(res);
    return true;
}

```

## Внесення даних

Спроба внесення неіснуючого(або NULL) genre\_id в таблицю Song (genre\_id має обмеження NOT NULL)



## Лістинг

```
bool Model::CreateRecord(Table table, std::vector<std::string> data,
std::string& errorMessage)
{
    std::string query;
    query+= "INSERT INTO " + TableSpecs::GetName(table) + " (";
    auto columns = TableSpecs::GetColumns(table);
    for (int32_t col = 0, size = columns.size(); col < size; col++)
    {
        if (columns[col].Type == ColumnType::Serial)
            continue;
        query += columns[col].Name;
        if (size - col != 1)
            query += ",";
    }
    query += ") VALUES (";
    for (int32_t col = 0, size = data.size(); col < size; col++)
    {
        if (columns[col].Type == ColumnType::Serial)
            continue;
```

```

if (data[col].empty())
{
    query += "NULL";
}
else
{
    query += "'" + data[col] + "'";
}
if (size - col != 1)
    query += ",";
}
query += ")";

```

```
PGresult* result = PQexec(m_Connection, query.c_str());
```

```

if (PQresultStatus(result) != PGRES_COMMAND_OK) {

    errorMessage = std::string("Insert Query execution failed for table: ") +
    TableSpecs::GetName(table) + "\n"
    "Invalid dependent data\n" + PQerrorMessage(m_Connection);
    VM_ERROR(errorMessage);
    VM_ERROR("Query text was: ", query, "\n");
    PQclear(result);
    return false;
}
PQclear(result);
errorMessage = "";
return true;
}

```

## Завдання 2

### Автоматичне генерування “рандомізованих” даних

#### Спосіб генерації

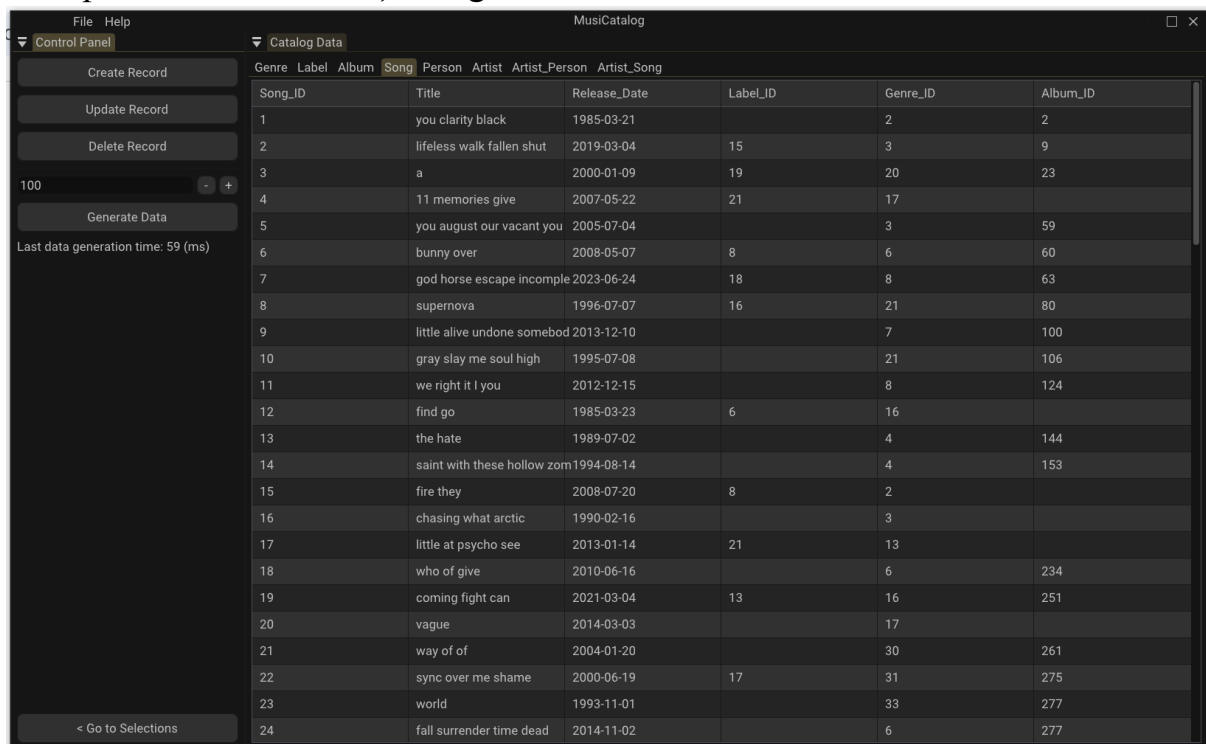
Для генерації текстових даних було використано підготовлені зразки даних.

Для таблиці Genre та Label дані вносяться з текстових файлів що знаходяться в директорії assets/data. В цих таблицях дані не рандомізовані.

Для всіх інших таблиць також використано зразки даних з текстових файлів. При запуску програми створюються допоміжні таблиці з даними, якщо вони ще не існують, і в них заносяться дані з іменами, прізвищами, і словами. Далі ці таблиці використовуються як джерело даниї для генерації рандомізованих даних основних таблиць.

#### Приклади генерації

##### Згенеровані дані таблиці Song



The screenshot shows the MusiCatalog application window. On the left is a 'Control Panel' with buttons for 'Create Record', 'Update Record', and 'Delete Record'. Below these is a slider set to 100 and a 'Generate Data' button. A status message reads 'Last data generation time: 59 (ms)'. At the bottom left is a '< Go to Selections' button. The main area is titled 'Catalog Data' and contains a table with columns: Genre, Label, Album, Song, Person, Artist, Artist\_Person, and Artist\_Song. The table displays 24 rows of generated data.

Genre	Label	Album	Song	Person	Artist	Artist_Person	Artist_Song
			Song_ID				Title
			Release_Date				Label_ID
			Genre_ID				Album_ID
		1	you clarity black				1985-03-21
		2	lifeless walk fallen shut				2019-03-04
		3	a				2000-01-09
		4	11 memories give				2007-05-22
		5	you august our vacant you				2005-07-04
		6	bunny over				2008-05-07
		7	god horse escape incomple				2023-06-24
		8	supernova				1996-07-07
		9	little alive undone somebod				2013-12-10
		10	gray slay me soul high				1995-07-08
		11	we right it i you				2012-12-15
		12	find go				1985-03-23
		13	the hate				1989-07-02
		14	saint with these hollow zom				1994-08-14
		15	fire they				2008-07-20
		16	chasing what arctic				1990-02-16
		17	little at psycho see				2013-01-14
		18	who of give				2010-06-16
		19	coming fight can				2021-03-04
		20	vague				2014-03-03
		21	way of of				2004-01-20
		22	sync over me shame				2000-06-19
		23	world				1993-11-01
		24	fall surrender time dead				2014-11-02

##### Згенеровані дані таблиці Artist\_Person

File Help

Control Panel

Create Record

Update Record

Delete Record

100 - +

Generate Data

Last data generation time: 19 (ms)

< Go to Selections

MusiCatalog

Catalog Data

Genre	Label	Album	Song	Person	Artist	Artist_Person	Artist_Song
Artist_ID					Person_ID		
1					12		
1					132		
1					854		
1					1021		
1					1640		
1					2099		
1					2626		
1					2670		
1					2930		
1					3647		
2					522		
2					658		
2					1523		
2					1795		
2					1844		
2					2115		
2					2478		
2					2935		
2					2937		
2					3120		
2					3260		
2					3412		
3					447		
3					968		

Згенеровані дані таблиці Person

File Help

Control Panel

Create Record

Update Record

Delete Record

100 - +

Generate Data

Last data generation time: 77 (ms)

< Go to Selections

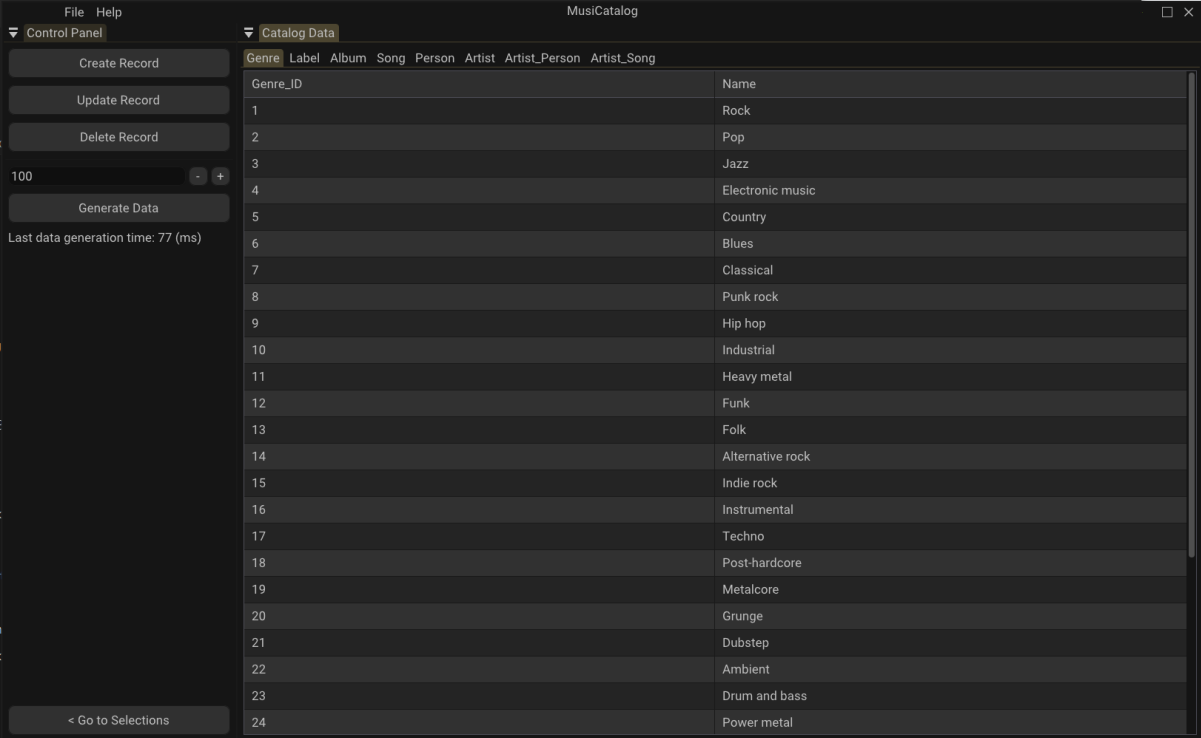
MusiCatalog

Catalog Data

Genre	Label	Album	Song	Person	Artist	Artist_Person	Artist_Song
Person_ID				Name	Birth_Date	Death_Date	
1				Ivy Smith	2004-01-22		
2				Evan Brown	1971-06-11		
3				Oliver Janes	1983-04-13		
4				Grace Garcia	1988-05-10		
5				Maria Davis	2002-04-21		
6				Finn Rodriguez	1984-12-06		
7				Jason Wilson	1973-06-18		
8				Remy Clark	1987-08-19		
9				Aiden Walker	1999-07-13		
10				Charlie King	1981-06-16		
11				Logan Rivera	1997-06-17		
12				Lucas Rivera	1976-07-30		
13				Dexter Evans	1975-11-02		
14				Kate Diaz	1983-04-09		
15				Nikita Morris	1970-08-21		
16				Harvey Cooper	1984-01-15		
17				Martin Bailey	1991-12-03		
18				Felix Price	1968-07-27		
19				Nova Ross	1975-05-21		
20				Charlie Foster	1966-03-06		
21				Zoey Powell	1999-07-30		
22				Ira Perry	1992-10-12		
23				Kai Perry	1974-05-22		
24				Lilian Russel	1991-09-26		



## Занесені дані таблиці Genre



Genre_ID	Name
1	Rock
2	Pop
3	Jazz
4	Electronic music
5	Country
6	Blues
7	Classical
8	Punk rock
9	Hip hop
10	Industrial
11	Heavy metal
12	Funk
13	Folk
14	Alternative rock
15	Indie rock
16	Instrumental
17	Techno
18	Post-hardcore
19	Metalcore
20	Grunge
21	Dubstep
22	Ambient
23	Drum and bass
24	Power metal

## SQL запити

### Запит генерації даних для таблиці Album

#### CREATE OR REPLACE FUNCTION

generate\_random\_sequence(max\_number **INT**)

**RETURNS TABLE**(amount\_of\_words **INT**, word\_id **INT**) **AS** \$\$

**BEGIN**

**FOR** i **IN** 1..max\_number **LOOP**

**FOR** j **IN** 1..**floor**(**random**() \* 5) + 1 **LOOP**

amount\_of\_words := i;

word\_id := **floor**(**random**() \* (**select count**(\*) **from** words)::int + 1);

**RETURN NEXT**;

**END LOOP**;

**END LOOP**;

**END**;

\$\$ **LANGUAGE** plpgsql;

**WITH** tab **AS**(

**SELECT** \* **FROM** generate\_random\_sequence(\$1)

)

, new\_words **AS**(

```

SELECT tab.amount_of_words, words.data
FROM tab
  INNER JOIN words on tab.word_id = words.words_id
)

, only_titles AS(
SELECT title FROM
(
  SELECT amount_of_words as title_id, string_agg(data, ' ' order by
random()) AS title
  FROM new_words
  GROUP BY amount_of_words
) AS new_titles
)

INSERT INTO album (title)
SELECT* FROM only_titles

```

*Занум генерації даних для таблиці Song*

```

CREATE OR REPLACE FUNCTION
generate_random_sequence(max_number INT)
RETURNS TABLE(amount_of_words INT, word_id INT) AS $$
BEGIN
  FOR i IN 1..max_number LOOP
    FOR j IN 1..floor(random() * 5) + 1 LOOP
      amount_of_words := i;
      word_id := floor(random() * (select count(*) from words)::int + 1);
      RETURN NEXT;
    END LOOP;
  END LOOP;
END;
$$ LANGUAGE plpgsql;

```

```

WITH tab AS( "
  SELECT * FROM generate_random_sequence($1)),

--Join tables to replace word id with actual word
new_words AS(
SELECT tab.amount_of_words, words.data
  FROM tab
  INNER JOIN words ON tab.word_id = words.words_id

```

) "

--Concatenate words into titles

```
, only_titles AS(  
SELECT amount_of_words AS title_id, string_agg(data, '' order by  
random()) AS title  
FROM new_words  
GROUP BY amount_of_words)
```

--Label Table with id and random number in range 1 - count label

```
, pick_label AS(  
SELECT number, floor(random() * (SELECT count(*) FROM label) +  
1) as rnd  
FROM generate_series(1, $1) as number  
) "
```

--Label Table with numbered labels from 1 to count

```
, label_number AS(  
SELECT  
ROW_NUMBER() OVER(ORDER BY label.label_id) AS number,  
label.label_id  
FROM label  
)
```

--Label Table with id and label\_id

```
, picked_label AS(  
SELECT pick_label.number AS id, label_number.label_id  
FROM pick_label  
LEFT JOIN label_number ON pick_label.rnd = label_number.number  
)
```

--Label Table with nulls

```
, ready_label AS(  
SELECT id,  
CASE  
WHEN(id + label_id) % 2 = 0 THEN NULL  
ELSE label_id  
END AS label_id  
FROM picked_label  
)
```

--Genre Table with id and random number in range 1 - count genre

```
, pick_genre AS(  
SELECT number, floor(random() * (SELECT count(*) FROM genre) +  
1) as rnd
```

```

    FROM generate_series(1, $1) as number
)
--Genre Table with numbered genres from 1 to count
, genre_number AS(
    SELECT
        ROW_NUMBER() OVER(ORDER BY genre.genre_id) AS number,
        genre.genre_id
    FROM genre
)
--Genre Table with id and genre_id
, ready_genre AS(
    SELECT pick_genre.number AS id, genre_number.genre_id
    FROM pick_genre
    INNER JOIN genre_number ON pick_genre.rnd = genre_number.number
)

--Album Table with id and random number in range 1 - count album
, pick_album AS(
    SELECT number, floor(random() * (SELECT count(*) FROM album) +
1) as rnd
    FROM generate_series(1, $1) as number
)
--Album Table with numbered albums from 1 to count
, album_number AS(
    SELECT
        ROW_NUMBER() OVER(ORDER BY album.album_id) AS number,
        album.album_id
    FROM album
)
--Album Table with id and album_id
, picked_album AS(
    SELECT pick_album.number AS id, album_number.album_id
    FROM pick_album
    LEFT JOIN album_number ON pick_album.rnd = album_number.number
)

--Album Table with nulls
, ready_album AS(
    SELECT id,
    CASE
    WHEN(id + album_id) % 5 = 0 THEN NULL
    ELSE album_id
    END AS album_id

```

```

    FROM picked_album
)

, results AS
(
SELECT
    only_titles.title,
    date_trunc('day', current_date - (random() * 365 * 40 + 10 ||
'days')::interval)::date AS date,
    ready_label.label_id,
    ready_genre.genre_id,
    ready_album.album_id

FROM only_titles
    INNER JOIN ready_label ON only_titles.title_id = ready_label.id
    INNER JOIN ready_genre ON only_titles.title_id = ready_genre.id
    INNER JOIN ready_album ON only_titles.title_id = ready_album.id)

INSERT INTO song (title, release_date, label_id, genre_id, album_id)
SELECT * FROM results

```

*Заняття генерації даних для таблиці Person*

```

WITH rnd_numbers AS(
SELECT
    floor(random() * (SELECT count(*) FROM names) + 1) as rnd_name,
    floor(random() * (SELECT count(*) FROM surnames) + 1) as
rnd_surname
FROM generate_series(1, $1) "
)

, results AS(
SELECT
    names.data || ' ' || surnames.data,
    date_trunc('day', current_date - (random() * 365 * 40 + 365 * 18 ||
'days')::interval)::date AS date
FROM rnd_numbers
    INNER JOIN names ON rnd_name = names.names_id
    INNER JOIN surnames ON rnd_surname = surnames.surnames_id
)
INSERT INTO person (name, birth_date)
SELECT* FROM results

```

### *Занум генерації даних для таблиці Artist*

#### **CREATE OR REPLACE FUNCTION**

generate\_random\_sequence(max\_number **INT**)

**RETURNS TABLE**(amount\_of\_words **INT**, word\_id **INT**) **AS** \$\$

**BEGIN**

**FOR** i **IN** 1..max\_number **LOOP**

**FOR** j **IN** 1..floor(random() \* 5) + 1 **LOOP**

            amount\_of\_words := i;

            word\_id := floor(random() \* (select count(\*) from words)::int + 1);

**RETURN NEXT**;

**END LOOP**;

**END LOOP**;

**END**;

\$\$ **LANGUAGE** plpgsql;

**WITH** tab **AS**(

**SELECT** \* **FROM** generate\_random\_sequence(\$1)

)

--Join tables to replace word id with actual word

, new\_words **AS**(

**SELECT** tab.amount\_of\_words, words.data

**FROM** tab

**INNER JOIN** words **ON** tab.word\_id = words.words\_id

)

--Concatenate words into titles

, only\_titles **AS**(

**SELECT** amount\_of\_words **AS** title\_id, string\_agg(data, '' **order by**  
random()) **AS** title

**FROM** new\_words

**GROUP BY** amount\_of\_words

)

--Label Table with id and random number in range 1 - count label

, pick\_label **AS**(

**SELECT** number, floor(random() \* (SELECT count(\*) FROM label) + 1)  
**as** rnd

**FROM** generate\_series(1, \$1) **as** number

)

--Label Table with numbered labels from 1 to count

, label\_number **AS**(

```

SELECT
    ROW_NUMBER() OVER(ORDER BY label.label_id) AS number,
    label.label_id
FROM
    label
)

--Label Table with id and label_id
, picked_label AS(
SELECT pick_label.number AS id, label_number.label_id
FROM pick_label
LEFT JOIN label_number ON pick_label.rnd = label_number.number
)

--Label Table with nulls
, ready_label AS(
SELECT id,
CASE
WHEN(id + label_id) % 2 = 0 THEN NULL
ELSE label_id
END AS label_id
FROM picked_label
)

, results AS(
SELECT
    only_titles.title,
    NULL AS description,
    date_trunc('day', current_date - (random() * 365 * 40 + 10 ||
'days')::interval)::date AS founded_date,
    NULL::date AS closed_date,
    ready_label.label_id
FROM only_titles
INNER JOIN ready_label ON only_titles.title_id = ready_label.id
)
INSERT INTO artist (title, description, founded_date, closed_date, label_id)
SELECT * FROM results

```

### *Занум генерації даних для таблиці Artist\_Person*

--Artist Table with numbered artists from 1 to count

```
WITH artist_number AS(  
SELECT  
    ROW_NUMBER() OVER(ORDER BY artist.artist_id) AS number,  
    artist.artist_id  
FROM  
    artist  
)
```

--Artist Table with id and random number in range 1 - count artist

```
, pick_artist AS(  
    SELECT number, floor(random() * (SELECT count(*) FROM artist) +  
1) as rnd  
    FROM generate_series(1, $1) as number  
)
```

--Artist Table with id and artist\_id

```
, ready_artist AS(  
    SELECT pick_artist.number AS id, artist_number.artist_id  
    FROM pick_artist  
    LEFT JOIN artist_number ON pick_artist.rnd = artist_number.number  
    ORDER BY id  
)
```

--Person Table with numbered persons from 1 to count

```
, person_number AS(  
SELECT  
    ROW_NUMBER() OVER(ORDER BY person.person_id) AS number,  
    person.person_id  
FROM  
    person  
)
```

--Person Table with id and random number in range 1 - count person

```
, pick_person AS(  
    SELECT number, floor(random() * (SELECT count(*) FROM person) +  
1) as rnd  
    FROM generate_series(1, $1) as number  
)
```

--Person Table with id and person\_id

```
, ready_person AS(  
    SELECT pick_person.number AS id, person_number.person_id
```



```

FROM pick_person
    LEFT JOIN person_number ON pick_person.rnd =
person_number.number
    ORDER BY id
)

, results AS(
    SELECT ready_artist.artist_id, ready_person.person_id
    FROM ready_artist
        INNER JOIN ready_person ON ready_artist.id = ready_person.id
        ORDER BY artist_id, person_id
)
INSERT INTO artist_person (artist_id, person_id)
SELECT DISTINCT * FROM results

```

### *Занум генерації даних для таблиці Artist\_Song*

```

--Artist Table with numbered artists from 1 to count
WITH artist_number AS(
SELECT
    ROW_NUMBER() OVER(ORDER BY artist.artist_id) AS number,
    artist.artist_id
FROM
    artist
)

--Artist Table with id and random number in range 1 - count artist
, pick_artist AS(
    SELECT number, floor(random() * (SELECT count(*) FROM artist) +
1) as rnd
    FROM generate_series(1, $1) as number
)

--Artist Table with id and artist_id
, ready_artist AS(
    SELECT pick_artist.number AS id, artist_number.artist_id
    FROM pick_artist
        LEFT JOIN artist_number ON pick_artist.rnd = artist_number.number
        ORDER BY id
)

--Song Table with numbered songs from 1 to count
, song_number AS(
SELECT

```

```

    ROW_NUMBER() OVER(ORDER BY song.song_id) AS number,
    song.song_id
FROM
    song
)
--Song Table with id and random number in range 1 - count song
, pick_song AS(
    SELECT number, floor(random() * (SELECT count(*) FROM song) +
1) as rnd
    FROM generate_series(1, $1) as number
)

--Song Table with id and song_id
, ready_song AS(
    SELECT pick_song.number AS id, song_number.song_id
    FROM pick_song
    LEFT JOIN song_number ON pick_song.rnd = song_number.number
    ORDER BY id
)

, results AS(
    SELECT ready_artist.artist_id, ready_song.song_id
    FROM ready_artist
    INNER JOIN ready_song ON ready_artist.id = ready_song.id
    ORDER BY artist_id, song_id
)
INSERT INTO artist_song (artist_id, song_id)
SELECT DISTINCT* FROM results

```

# Пошук даних

## Результати

## Виконання першого пошуку

FileHelp

Selection Panel

Persons under 24 - + years

With more than 3 - + songs

Of Metalcore genre

Select 1

The most popular genres

By released songs amount

And the most popular artist

In interval from

To

Select 2

Songs in interval

From

To

Select 3

Last query execution time: 9 (ms)

< Go to Management

Selections

Name	Age	Number of Songs	Number of Albums	Genre
Aurora Kirk	23	8	7	'Metalcore'
Peter Greer	21	8	6	'Metalcore'
Luke Dennis	22	7	5	'Metalcore'
Enzo Dickerson	22	7	7	'Metalcore'
Miles Mendez	18	6	3	'Metalcore'
Serena Boone	18	6	6	'Metalcore'
Emerson Jackson	22	6	5	'Metalcore'
Jayden Carey	23	6	3	'Metalcore'
Emmett Wells	19	5	5	'Metalcore'
Joseph Padilla	22	5	3	'Metalcore'
Molly Chase	20	5	4	'Metalcore'
Sage Weber	20	5	4	'Metalcore'
Kyra Short	21	4	3	'Metalcore'
Serena Yang	18	4	2	'Metalcore'
Travis Little	18	4	3	'Metalcore'
Caleb Alvarez	22	4	3	'Metalcore'
Vivian Leon	22	4	4	'Metalcore'
Theo Rocha	18	4	3	'Metalcore'
Hunter Rice	23	4	1	'Metalcore'
Aiden Baker	19	4	4	'Metalcore'
Remi Edwards	19	4	2	'Metalcore'
Maeve Bruce	20	4	3	'Metalcore'
Margot Snow	20	4	4	'Metalcore'
Ivan Gill	20	4	4	'Metalcore'

## Виконання другого пошуку

FileHelp

Selection Panel

Persons under 24 - + years

With more than 3 - + songs

Of Metalcore genre

Select 1

The most pupular genres

By released songs amount

And the most popular artist

In interval from 2007-06-11

To 2019-02-17

Select 2

Songs in interval

From

To

Select 3

Last query execution time: 22 (ms)

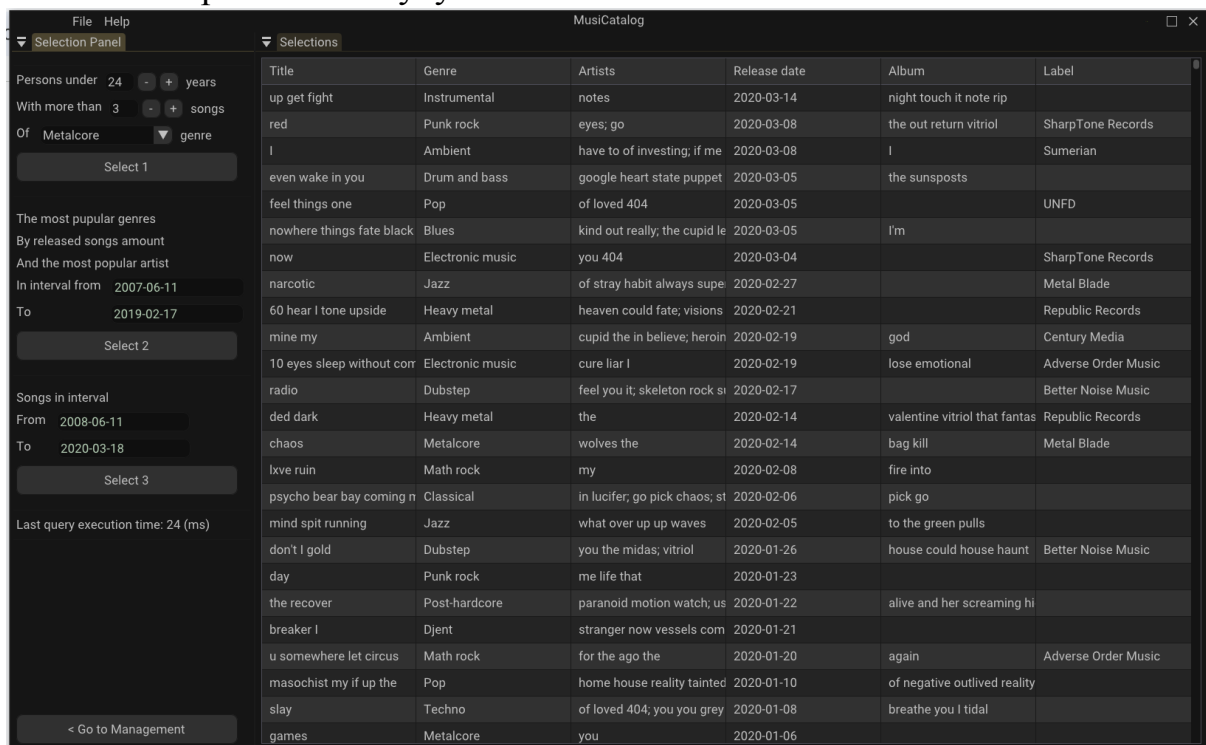
< Go to Management

MusiCatalog

Selections

Genre	Number of Songs	Number of Artists	The Most Popular Artist
Rock	84	77	I that another tone coming
Pop	76	73	escape thang 11 me mine
Jazz	78	74	bunny king we pass aura
Electronic music	83	67	the hell reality
Country	71	63	the
Blues	107	84	what you wolves
Classical	107	91	hell rotoscope suffering to ruins
Punk rock	93	82	divide paranoid one call me
Hip hop	75	67	hurt down fon't like
Industrial	77	68	in breathe narcotic
Heavy metal	98	78	flat
Funk	83	65	on down wraith name
Folk	81	62	aerials you
Alternative rock	79	79	let screaming hold
Indie rock	89	77	has effect into it jerk
Instrumental	90	80	heart
Techno	81	61	bad a I flat
Post-hardcore	93	93	if me a maybe dark
Metalcore	82	76	eyes rust me fool fell
Grunge	106	82	dust are liar salt
Dubstep	95	86	prove the spit if
Ambient	86	69	google heart state puppet
Drum and bass	88	84	you
Power metal	73	64	friends gate
Math rock	95	86	everything lose like clean

## Виконання третього пошуку



## SQL запити

### Перший запит

```
WITH person_under_age AS(
SELECT
    person_id,
    name,
    (DATE_PART('year', age(current_date, birth_date))) AS age
FROM person
WHERE (DATE_PART('year', age(current_date, birth_date))) < $1
ORDER BY age
)
```

```
, artists_of_persons AS(
SELECT artist_person.artist_id, artist_person.person_id
FROM artist_person
INNER JOIN person_under_age ON person_under_age.person_id =
artist_person.person_id
)
, artist_song_of_persons AS(
```

```

SELECT DISTINCT artists_of_persons.artist_id, artist_song.song_id FROM
artist_song
  INNER JOIN artists_of_persons ON artist_song.artist_id =
artists_of_persons.artist_id
  INNER JOIN song ON artist_song.song_id = song.song_id
  WHERE song.genre_id = $3
)
, artist_countsongs AS(
SELECT artist_id, count(song_id) AS countsongs
FROM artist_song_of_persons
  GROUP BY artist_id
  ORDER BY artist_id
)
, artist_countalbums AS(
SELECT artist_id, count(album_id) AS countalbums
FROM artist_song_of_persons
  INNER JOIN song ON artist_song_of_persons.song_id = song.song_id
  GROUP BY artist_id
  ORDER BY artist_id
)
, person_count AS(
SELECT
  artists_of_persons.person_id,
  sum(artist_countsongs.countsongs) as countsong,
  sum(artist_countalbums.countalbums) as countalbum
FROM artists_of_persons
  INNER JOIN artist_countsongs ON artists_of_persons.artist_id =
artist_countsongs.artist_id
  INNER JOIN artist_countalbums ON artists_of_persons.artist_id =
artist_countalbums.artist_id
  GROUP BY artists_of_persons.person_id
)
, results AS(
SELECT person.name, age, countsong, countalbum, $4 as genre
FROM person_count
  INNER JOIN person ON person_count.person_id = peson.person_id
  INNER JOIN person_under_age ON person_count.person_id =
person_under_age.person_id
  WHERE countsong > $2
  ORDER BY countsong DESC
)

SELECT * FROM results

```

## *Другой запуск*

```
WITH song_in_interval AS(
SELECT *
FROM song
    WHERE release_date > $1 AND release_date < $2
    ORDER BY song_id
)
, genre_countsongs AS(
SELECT genre_id, count(song_in_interval.song_id) as countsongs
FROM song_in_interval
    GROUP BY genre_id
    ORDER BY countsongs DESC
)
, genre_countartists AS(
SELECT genre_id, count(DISTINCT artist_song.artist_id) as countartists
FROM song_in_interval
    INNER JOIN artist_song ON song_in_interval.song_id =
artist_song.song_id
    GROUP BY genre_id
    ORDER BY countartists DESC
)
, artist_most_songs AS(
SELECT
    song_in_interval.genre_id,
    artist_song.artist_id,
    RANK() OVER(PARTITION BY song_in_interval.genre_id ORDER BY
COUNT(song_in_interval.song_id) DESC) AS artist_rank
FROM song_in_interval
    INNER JOIN artist_song ON song_in_interval.song_id =
artist_song.song_id
    GROUP BY artist_song.artist_id, song_in_interval.genre_id
)
, results AS(
SELECT
    max(genre.name) AS genre,
    max(genre_countsongs.countsongs) AS song_count,
    max(genre_countartists.countartists) AS artist_count,
    min(artist.title) AS most_popular_artist
FROM genre_countsongs
    JOIN genre_countartists ON genre_countsongs.genre_id =
genre_countartists.genre_id
```

```

    JOIN artist_most_songs ON genre_countsongs.genre_id =
    artist_most_songs.genre_id AND artist_most_songs.artist_rank = 1
    JOIN artist ON artist_most_songs.artist_id = artist.artist_id
    JOIN genre ON genre_countsongs.genre_id = genre.genre_id
    GROUP BY genre_countsongs.genre_id
    ORDER BY genre_countsongs.genre_id
)

SELECT * FROM results

```

### *Трети занад*

```

WITH new_song AS(
SELECT song_id, song.title, genre.name as genre, song.release_date,
album.title AS album, label.name AS label
FROM song
    INNER JOIN genre ON song.genre_id = genre.genre_id
    LEFT JOIN album ON song.album_id = album.album_id
    LEFT JOIN label ON song.label_id = label.label_id
    WHERE release_date > $1 AND release_date < $2
)
, results AS(
SELECT
    new_song.title,
    new_song.genre,
    STRING_AGG(artist.title, '; ') AS artists,
    new_song.release_date,
    new_song.album,
    new_song.label
FROM new_song
    INNER JOIN artist_song ON new_song.song_id = artist_song.song_id
    INNER JOIN artist ON artist_song.artist_id = artist.artist_id
    GROUP BY new_song.song_id, new_song.title, new_song.genre,
    new_song.release_date, new_song.album, new_song.label
    ORDER BY new_song.release_date DESC
)

SELECT* FROM results

```

## Завдання 4

### Модуль Model згідно MVC

#### *Шаблон MVC*

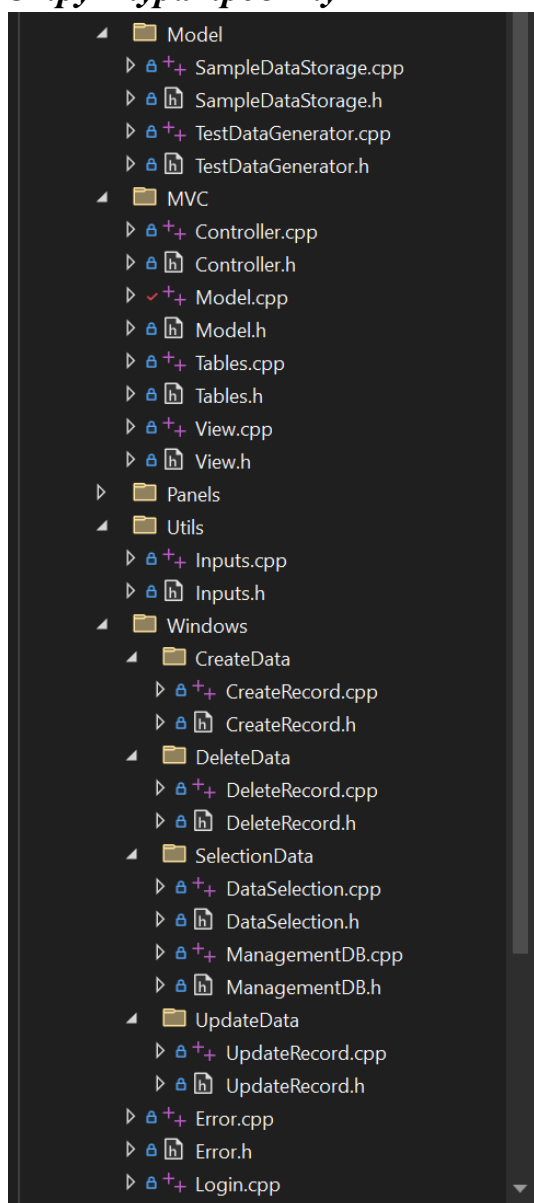
Згідно завдання для архітектури програми було використано шаблон проектування MVC, для розподілення частин програми на модулі.

Model - Модуль взаємодії з базою даних

View - Модуль інтерфейсу користувача, для відображення і введення даних

Controller - Модуль для взаємодії View і Model шляхом передачі, обробки даних, а також передачі сповіщень.

#### *Структура проекту*



Директорія Windows відноситься до модулю View



## Программный код модуля *Model*

### Файл Model.h

```
1  #pragma once
2  #include <libpq-fe.h>
3  #include <memory>
4  #include <string>
5  #include "Tables.h"
6  #include "Model/SampleDataStorage.h"
7  #include "Model/TestDataGenerator.h"
8
9  class Model
10 {
11 public:
12     ~Model();
13     void Finish();
14     bool Connect(const std::string& dbname, const std::string& username, const std::string& password, std::string& errorMessage);
15
16     bool CreateTables(std::string& errorMessage);
17
18     std::shared_ptr<TableData> FetchTableData(Table table, std::vector<std::string> pkeysTitles, std::string& errorMessage);
19     bool CreateRecord(Table table, std::vector<std::string> recordData, std::string& errorMessage);
20     bool UpdateRecord(Table table, std::vector<std::string> recordData, std::vector<Column> pkeyColumns, std::vector<std::string> pkeysData, std::string& errorMessage);
21     bool DeleteRecord(Table table, std::vector<Column> pkeyColumns, std::vector<std::string> pkeysData, std::string& errorMessage);
22
23     std::vector<std::string> GetRecordIfExists(Table table, std::vector<Column> pkeyColumns, std::vector<std::string> pkeysData, std::string& errorMessage);
24     std::vector<std::string> GetPKeyColumnTitles(Table table, std::string& errorMessage);
25
26     bool GenerateData(Table table, int32_t rowCount, std::string& errorMessage);
27     bool CheckTableMinRecords(Table table, int32_t count);
28
29     // Selections
30     std::vector<std::string> GetListOfGenres(std::string& errorMessage);
31     std::pair<std::shared_ptr<TableData>, std::vector<std::string>> ExecuteFirstSelection(int32_t age, int32_t songCount, int32_t genreId, std::string genre, std::string& errorMessage);
32     std::pair<std::shared_ptr<TableData>, std::vector<std::string>> ExecuteSecondSelection(const std::string& fromDate, const std::string& toDate, std::string& errorMessage);
33     std::pair<std::shared_ptr<TableData>, std::vector<std::string>> ExecuteThirdSelection(const std::string& fromDate, const std::string& toDate, std::string& errorMessage);
34 public:
35     // Test Data
36     bool LoadTestDataSamples(std::string& errorMessage);
37     bool CreateAuxiliaryTablesForTestData(std::string& errorMessage);
38
39 private:
40     std::string CreateTableGenre();
41     std::string CreateTableLabel();
42     std::string CreateTableSong();
43     std::string CreateTableAlbum();
44     std::string CreateTablePerson();
45     std::string CreateTableArtist();
46     std::string CreateTableArtist_Person();
47     std::string CreateTableArtist_Song();
48
49     std::string CheckCreateResult(PGresult* res, ExecStatusType status, const std::string& text);
50 private:
51     PGconn* m_Connection;
52
53     SampleDataStorage m_SampleDataStorage;
54     TestDataGenerator m_Generator;
55
56 };
57
```

### Файл TestDataGenerator.h

```
1  #pragma once
2  #include <vector>
3  #include <string>
4  #include <libpq-fe.h>
5  #include "MVC/Tables.h"
6
7  class TestDataGenerator
8  {
9  public:
10     bool GenerateGenres(std::vector<std::string>& genres, std::string& errorMessage);
11     bool GenerateLabels(std::vector<std::string>& labels, std::vector<std::string>& locations, std::string& errorMessage);
12     bool GenerateAlbums(int32_t rowCount, std::string& errorMessage);
13     bool GenerateSong(int32_t rowCount, std::string& errorMessage);
14     bool GeneratePerson(int32_t rowCount, std::string& errorMessage);
15     bool GenerateArtist(int32_t rowCount, std::string& errorMessage);
16     bool GenerateArtist_Person(int32_t rowCount, std::string& errorMessage);
17     bool GenerateArtist_Song(int32_t rowCount, std::string& errorMessage);
18
19 public:
20     bool DeleteData(Table table, std::string& errorMessage);
21     void SetConnection(PGconn* connection);
22 private:
23
24     PGconn* m_Connection;
25
26 };

```

## Файл SampleDataStorage.h

```
1  #pragma once
2  #include <vector>
3  #include <string>
4  #include <libpq-fe.h>
5  #include "MVC/Tables.h"
6
7  class SampleDataStorage
8  {
9  public:
10     // Data
11     bool LoadFiles();
12     std::vector<std::string> LoadFile(const std::string& fileName);
13
14     // Tables
15     bool CreateTables();
16     bool CreateTable(AuxiliaryTable tableName, std::vector<std::string> data);
17
18     void SetConnection(PGconn* connection);
19 public:
20     std::vector<std::string> GetNames() { return m_Names; }
21     std::vector<std::string> GetSurnames() { return m_Surnames; }
22     std::vector<std::string> GetGenres() { return m_Genres; }
23     std::vector<std::string> GetLabels() { return m_Labels; }
24     std::vector<std::string> GetLocations() { return m_Locations; }
25     std::vector<std::string> GetWords() { return m_Words; }
26 private:
27     std::vector<std::string> m_Names;
28     std::vector<std::string> m_Surnames;
29     std::vector<std::string> m_Genres;
30     std::vector<std::string> m_Labels;
31     std::vector<std::string> m_Locations;
32     std::vector<std::string> m_Words;
33
34     PGconn* m_Connection;
35 };
```

В звіті приведено програмний код лише оголошення класів, так як програмний код їх реалізації налічує більше 1600 рядків.

Повний код за посиланням

[https://github.com/SkaLe3/KPI\\_DataBase\\_Labs/tree/main/MusCat/src](https://github.com/SkaLe3/KPI_DataBase_Labs/tree/main/MusCat/src)

## *Опис функцій*

class SampleDataStorage:

Клас для збереження зразків даних

- LoadFiles() - Функція для завантаження всіх даних з усіх файлів у вектори рядків, для подальшого заповнення допоміжних таблиць для генерації даних.
- LoadFile() - Функція читає файл і вставляє рядки у вектор який потім повертає
- CreateTable() - Функція створення допоміжної таблиці яка буде використовуватись для генерації даних
- CreateTables() - Функція створення всіх допоміжних таблиць
- SetConnection() - Встановлення зв'язку до бази даних який тримає Model
- Get%name\_of\_vector%() - Функції повернення векторів з даними

class TestDataGenerator:

Клас для генерації рандомізованих даних

- Generate%table\_name%() - Функції для генерації даних для кожної таблиці, виконанням SQL запиту
- DeleteData() - функція видалення всіх даних з таблиці
- SetConnection() - Встановлення зв'язку до бази даних який тримає Model

class Model

Клас взаємодії з базою даних

- Connect() - Виконання запиту встановлення зв'язку з базою даних.
- Finish() - Розрив зв'язку з базою даних
- CreateTables() - Функція яка викликає створення для кожної таблиці бази даних
- FetchTypeData() - Функція для отримання всіх даних вказаної таблиці SQL запитом
- CreateRecord() - Виконання запиту для створення нового запису у вказаній таблиці з наданими даними для внесення
- UpdateRecord() - Виконання запиту для оновлення існуючого запису у вказаній таблиці з наданими даними для внесення
- DeleteRecord() - Виконання запиту для видалення існуючого запису у вказаній таблиці за наданими даними для пошуку запису
- GetRecordIfExists() - Функція повертає запис який було знайдено за наданим ключем, якщо такий запис існує
- GetPrimaryKeyTitles() - Функція повертає вектор назв стовпців які входять в первинний ключ вказаної таблиці

- `GenerateData()` - Функція для передачі виклику генерації даних конкретної вказаної таблиці
- `CheckTableMinRecords()` - Перевірка на те чи містить таблиця вказану кількість записів
- `GetListOfGenres()` - Функція для отримання вектору з назвами всіх жанрів з таблиці `Genre`
- `ExecuteFirstSelection()` - Виконання першого запиту пошуку даних
- `ExecuteSecondSelection()` - Виконання другого запиту пошуку даних
- `ExecuteThirdSelection()` - Виконання третього запиту пошуку даних
- `LoadTestDataSamples` - Функція обгортка для завантаження даних з файлів
- `CreateAuxiliaryTablesForTestData()` - Функція для створення допоміжних таблиць для генерації даних
- `CreateTable%table_name%()` - Функції для створення конкретних таблиць