



**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

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

**Лабораторна робота № 2
з дисципліни “ Базы даних і засоби управління”
Створення додатку бази даних, орієнтованого на взаємодію з СУБД
PostgreSQL**

**Виконав
студент III курсу
групи KB-81
Гей Антон**

Київ 2020

Метою роботи є здобуття вмінь програмування прикладних додатків баз даних 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/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

Деталізоване завдання:

1. Забезпечити можливість введення/редагування/вилучення даних у таблицях бази даних з можливістю контролю відповідності 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 підлеглий таблиці. З точки зору підлеглої таблиці варто контролювати наявність відповідного 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 користувачем 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!**

Приклад генерації 100 псевдовипадкових чисел:

```
select trunc(random()*1000)::int
from generate_series(1,100)
```

	trunc integer	
1	368	
2	773	
3	29	
4	66	
5	497	
6	956	

Приклад генерації 5 псевдовипадкових рядків:

```
select chr(trunc(65+random()*25)::int) || chr(trunc(65+random()*25)::int)
from generate_series(1,5)
```

	?column? text	
1	NE	
2	MQ	
3	RN	
4	DW	
5	DA	

Приклад генерації псевдовипадкової мітки часу з діапазону [доступний за посиланням](#).

Кількість даних для генерування має вводити користувач з клавіатури. Для тесту взяти 100 000 записів для однієї-двох таблиць.

Особливу увагу слід звернути на відповідність даних вимогам зовнішніх ключів з метою уникнення помилок порушення обмежень цілісності (foreign key).

1. Для реалізації пошуку необхідно підготувати 3 запити, що 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). Приклад організації коду згідно шаблону доступний [за даним посиланням](#). При цьому модель, подання та контролер



мають бути реалізовані у окремих файлах. Для доступу до бази даних використовувати **лише мову SQL** (без ORM).

Початкові значення таблиць

Films

public.Films/postgres/postgres@PostgreSQL 12





Data Output

	 id_f [PK] integer	 name_f character varying (30)	 year_f integer	 genre_f character varying (30)	 duration_f integer
1	1	Forest Gump	1994	drama	142
2	2	Titanic	1997	drama	194
3	3	Back to the Future	1985	comedy	116
4	4	The Matrix	1999	fantasy	136
5	5	Avatar	2009	fantasy	162
6	6	Avengers	2018	action	149




Cinemas

public.Cinemas/postgres/postgres@PostgreSQL 12			
Data Output			
	id_c [PK] integer	name_c character varying (30)	street character varying (30)
1	1	Multiplex	Gnat Khotkevich street
2	2	Oskar	Peremogy Avenue
3	3	Kiev	Velyka Vasylkivska street
4	4	Cinema City	Antonovich street

Sessions

public.Sessions/postgres/postgres@PostgreSQL 12				
Data Output				
	 id_s [PK] integer	 id_f integer	 start_date character varying (30)	 hall_name character varying (30)
1	1	3	05/12/2020	Green
2	2	1	12/11/2020	Second
3	3	2	18/06/2019	Cat
4	4	4	23/01/2021	Square
5	5	6	30/11/2020	Yellow
6	6	5	19/07/2018	Dog
7	7	3	25/08/2021	First
8	8	2	08/10/2020	Circle

Schedule

public.Schedule/postgres/postgres@PostgreSQL 12				
Data Output				
	 id_sch [PK] integer	 id_s integer	 id_c integer	
1	1	1	1	
2	2	2	2	
3	3	3	3	
4	4	4	4	
5	5	5	1	
6	6	6	3	
7	7	7	2	
8	8	8	4	

CRUD (create, read, update, delete)

1. Редагувати/відобразити дані таблиці

2. Генерація випадкових даних

3. Пошук даних

4. Вихід

Виберіть один з пунктів...

1

1. Films

2. Cinemas

3. Schedule

4. Sessions

Виберіть одну з таблиць...

1

1. Додати запис

2. Оновити дані запису

3. Відобразити дані

4. Видалити дані

5. Вихід

Виберіть один з пунктів...

Додавання запису в таблицю Films

Виберіть один з пунктів...

1

Введіть значення поля id_f

Тип поля число

7

Введіть значення поля name_f

Тип поля текст

Film7

Введіть значення поля year_f

Тип поля число

2020

Введіть значення поля genre_f

Тип поля текст

best_genre

Введіть значення поля duration_f

Тип поля число

200

+++++

Успіх! Запис з ідентифікатором 7 було додано до таблиці Films !

+++++

Оновлення запису

```

Виберіть один з пунктів...
2
Введіть значення поля id_f
Тип поля число
7
Введіть значення поля name_f
Тип поля текст
New_Film7
Введіть значення поля year_f
Тип поля число
2020
Введіть значення поля genre_f
Тип поля текст
new_genre
Введіть значення поля duration_f
Тип поля число
201
---
Запис 7 було змінено на [7, 'New_Film7', 2020, 'new_genre', 201]
---
```

Видалення запису

```

Виберіть один з пунктів...
4
Введіть ключ для запису який необхідно видалити
7
-----
Елемент 7 було успішно видалено!
-----
```

Перегляд даних з таблиці

```

Виберіть один з пунктів...
3
['id_f', 'name_f', 'year_f', 'genre_f', 'duration_f']
--- ТАБЛИЦЯ FILMS ---
1. (1, 'Forest Gump', 1994, 'drama', 142)
2. (2, 'Titanic', 1997, 'drama', 194)
3. (3, 'Back to the Future', 1985, 'comedy', 116)
4. (4, 'The Matrix', 1999, 'fantasy', 136)
5. (5, 'Avatar', 2009, 'fantasy', 162)
6. (6, 'Avengers', 2018, 'action', 149)
```

Спроба ввести запис з ключем, який уже існує

public.Cinemas/postgres/postgres@PostgreSQL 12			
Data Output			
	id_c [PK] integer	name_c character varying (30)	street character varying (30)
1		1 Multiplex	Gnat Khotkevich street
2		2 Oskar	Peremogy Avenue
3		3 Kiev	Velyka Vasylkivska street
4		4 Cinema City	Antonovich street

1. Додати запис

2. Оновити дані запису

3. Відобразити дані

4. Видалити дані

5. Вихід

Виберіть один з пунктів...

1

Введіть значення поля id_c

Тип поля число

3

Введіть значення поля name_c

Тип поля текст

Lviv

Введіть значення поля street

Тип поля текст

Kovalya street

База даних вже містить елемент 3 в таблиці Cinemas !

ПОМИЛКА: повторювані значення ключа порушують обмеження унікальності "cinemas_pkey"

DETAIL: Ключ (id_c)=(3) вже існує.

Завдання 2

Генерація даних

Виберіть один з пунктів...

2

1. Films
2. Cinemas
3. Schedule
4. Sessions

Виберіть одну з таблиць...

1

Обрано таблицю Films

Скільки записів потрібно згенерувати?

3

Введіть дані для поля id_f

Введіть максимальне число

20

Введіть дані для поля name_f

Введіть довжину рядка

6

Введіть дані для поля year_f

Введіть максимальне число

2010

Введіть дані для поля genre_f

Введіть довжину рядка

6

Введіть дані для поля duration_f

Введіть максимальне число

250

public.Films/postgres/postgres@PostgreSQL 12						
Data Output						
	id_f [PK] integer	name_f character varying (30)	year_f integer	genre_f character varying (30)	duration_f integer	
1	1	Forest Gump	1994	drama	142	
2	2	Titanic	1997	drama	194	
3	3	Back to the Future	1985	comedy	116	
4	4	The Matrix	1999	fantasy	136	
5	5	Avatar	2009	fantasy	162	
6	6	Avengers	2018	action	149	
7	15	TJOROT	707	MVRLCI	188	
8	17	YWQHDO	1100	QOKLOT	13	

Завдання 3

Пошук фільмів, які будуть показувати у певному кінотеатрі після
вказаної дати

1. Редагувати/відобразити дані таблиці
2. Генерація випадкових даних
3. Пошук даних
4. Вихід

Виберіть один з пунктів...

3

1. фільми, що будуть показувати після дати у кінотеатрі

Виберіть одну з дій...

1

Введіть назву кінотеатру

Multiplex

Введіть дату після якої будуть показувати фільми (дд/мм/рр)

29/10/2019

Finished 'search_query' in 0.0041 secs

['name_f', 'genre_f', 'start_date', 'hall_name']

--- ТАБЛИЦЯ FILMS ---

1. ('Avengers', 'action', '30/11/2020', 'Yellow')

```
query = 'SELECT name_f, genre_f, start_date, hall_name ' \
        'FROM "Films", "Sessions", "Schedule", "Cinemas" ' \
        'WHERE "Films"."id_f" = "Sessions"."id_f" ' \
        'AND "Sessions"."id_s" = "Schedule"."id_s" ' \
        'AND "Schedule"."id_c" = "Cinemas"."id_c" ' \
        'AND "Cinemas".name_c = \'' + cinema + '\' AND "Sessions".start_date > \'' + after_date + \'' \
        .format(cinema, after_date)
```

Завдання 4

amdota / databases

Unwatch 1 Star 0 Fork 0

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

main databases / lab2 /

Go to file Add file

amdota Add files via upload bae35f3 now History

..

bd_func.py Add files via upload now

controller.py Add files via upload now

main.py Add files via upload now

model.py Add files via upload now

timer.py Add files via upload now

view.py Add files via upload now

<https://github.com/amdota/databases/tree/main/lab2>