

**Міністерство освіти та науки України
Національний технічний університет України “Київський
політехнічний інститут імені Ігоря Сікорського”
Факультет прикладної математики
Кафедра системного програмування і спеціалізованих комп’ютерних
систем**

РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА
з дисципліни
“Бази даних та засоби управління”
**“Створення додатку бази даних, орієнтованого на взаємодію з СУБД
PostgreSQL”**

Виконала Лукашук Ю. А.
Студентка групи KB-22
github: [yuliya2409/db-kpi \(github.com\)](https://github.com/yuliya2409/db-kpi)
telegram: @sunshine_o9

Київ 2024

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).

Структура бази даних

Тема: “Платформа для бронювання та управління майданчиками для подій”

Перелік сутностей з описом їх призначення

Організатор (Manager) - сутність організатора події

Подія (Event) - сутність події, якою керує організатор

Майданчик (Venue) - сутність майданчика, на якому може проходити подія.

Між сутностями проходять наступні зв'язки

Організатор(1) -> влаштовує -> подію(0...N)

Подія(0...N) -> відбувається на -> майданчику(1)

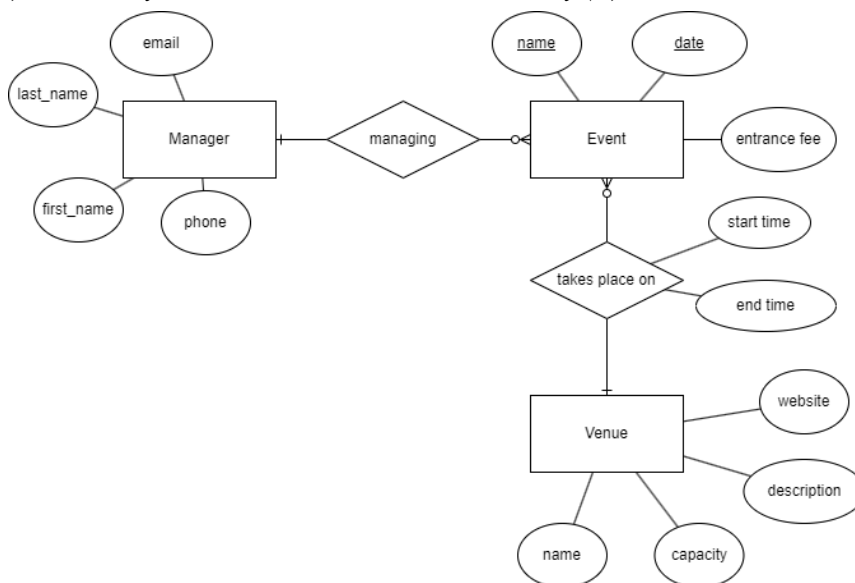
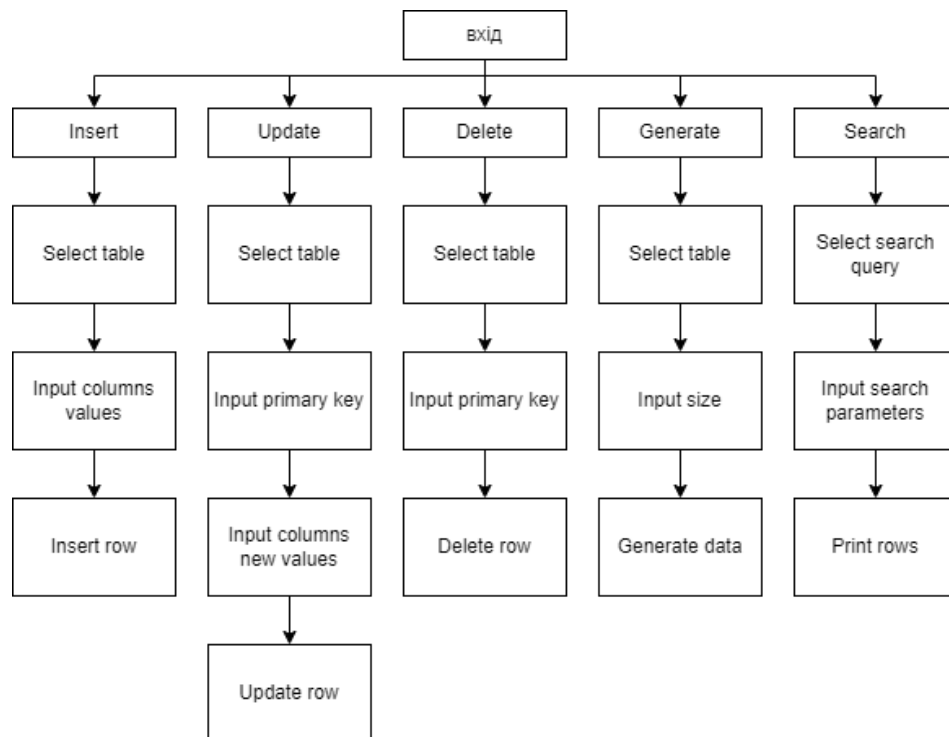


Схема меню користувача

1. Insert - пункт для вставки даних за заданими користувачами значеннями
2. Update - пункт для редагування даних за заданими користувачами значеннями
3. Delete - пункт для видалення даних за заданими користувачами значенням
4. Generate - пункт для генерації даних заданого користувачем розміру
5. Search - пункт для пошуку даних за допомогою різних пошукових запитів
6. Вихід



Мова програмування та бібліотеки

Для виконання роботи було використано мову Java та JDBC як інструмент доступу до даних, org.postgresql.Driver як драйвер бази даних для JDBC.

Пункт №1

Скріншоти результатів виконання операції вилучення запису батьківської таблиці:

```
Table:
1. event
2. manager
3. venue
4. events_venue
2
Primary key:
manager_id:
1
DELETE FROM "manager" WHERE manager_id='1'
org.postgresql.util.PSQLException: ОШИБКА: UPDATE или DELETE в таблице "manager" нарушает ограничение внешнего ключа "fk_manager_id" таблицы "e
  Подробности: На ключ (manager_id)=(1) всё ещё есть ссылки в таблице "event".
```

СУБД не може видалити запис з батьківської таблиці, оскільки це порушує обмеження зовнішнього ключа, тому видає помилку, яку перехоплює програма.

```
Table:
1. event
2. manager
3. venue
4. events_venue
2
Primary key:
manager_id:
25349
DELETE FROM "manager" WHERE manager_id='25349'
```

СУБД успішно видалила запис з батьківської таблиці

```
Primary key:
manager_id:
20047
Columns:
manager_id:
1
first_name:
1
last_name:
1
email:
1
phone:
1
UPDATE "manager" SET manager_id='1',phone='1',last_name='1',first_name='1',email='1' WHERE manager_id='20047'
org.postgresql.util.PSQLException: ОШИБКА: повторяющееся значение ключа нарушает ограничение уникальности "manager_pkey"
  Подробности: Ключ "(manager_id)=(1)" уже существует.
```

СУБД не може відредагувати запис, оскільки це порушує обмеження первинного ключа, отже видає помилку, яку перехоплює програма.

```
Primary key:
manager_id:
20047
Columns:
manager_id:
20047
first_name:
TestName
last_name:
1
email:
1
phone:
1
UPDATE "manager" SET manager_id='20047',phone='1',last_name='1',first_name='TestName',email='1' WHERE manager_id='20047'
```

СУБД успішно відредагувала запис

Скріншоти результатів виконання операції вставки запису в дочірню таблицю:

```
Table:
1. event
2. manager
3. venue
4. events_venue
1
Columns:
name:
test
date:
2024-01-11
entrance_fee:
100
manager_id:
99999
INSERT INTO "event"(date,manager_id,name,entrance_fee) VALUES ('2024-01-11','99999','test','100')
org.postgresql.util.PSQLException: ОШИБКА: INSERT или UPDATE в таблице "event" нарушает ограничение внешнего ключа "fk_manager_id"
Подробности: Ключ (manager_id)=(99999) отсутствует в таблице "manager".
```

СУБД не може виконати операцію вставки запису в дочірню таблицю, оскільки атрибут manager_id порушує обмеження зовнішнього ключа, тому видає помилку, яка перехоплюється програмою

```

Table:
1. event
2. manager
3. venue
4. events_venue
1
Columns:
name:
Test
date:
2024-01-01
entrance_fee:
100
manager_id:
2
INSERT INTO "event"(date,manager_id,name,entrance_fee) VALUES ('2024-01-01','2','Test','100')

```

СУБД успішно вставила запис до дочірньої таблиці

Пункт №2

Текст запиту генерації випадкових даних до таблиці events:

```

insert into manager(manager_id, first_name, last_name, email, phone)

select * from (select (random()*100000)::int as manager_id,
chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int)
as first_name,

chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int)
as last_name,

chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int)
as email,





to_char(random() * 10000000000, 'FM(000) 000-0000') as phone

from generate_series(1,100000)) as _





where (manager_id not in (select manager_id from manager))

limit 100000;

```

	name [PK] character varying 	date [PK] date 	entrance_fee integer 	manager_id integer 
1	Anna Trincher	2024-11-01	700	2
2	DREVO	2024-10-23	450	3
3	KOLA	2024-10-19	700	1
4	Olya Polyakova	2024-10-26	0	3

```
Main menu
1. Insert
2. Update
3. Delete
4. Generate
5. Search
6. Exit
4
Table:
1. manager
2. event
2
Dataset:
Size:
5
```

	name [PK] character varying 	date [PK] date 	entrance_fee integer 	manager_id integer 
1	Anna Trincher	2024-11-01	700	2
2	DREVO	2024-10-23	450	3
3	FRW	2025-09-16	41	2
4	KOLA	2024-10-19	700	1
5	Olya Polyakova	2024-10-26	0	3
6	OSB	2025-07-16	21	1
7	REK	2025-05-10	61	3
8	UET	2025-07-30	27	1
9	YYM	2025-09-09	77	2

Текст запиту для генерації випадкових значень до таблиці manager:

```
INSERT INTO \"event\"(name, date, entrance_fee, manager_id)

SELECT chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() *
25)::int) || chr(trunc(65 + random() * 25)::int) as event_name,

DATE(CURRENT_DATE + random() * interval '10 months') as date,

(random() * 100)::int as entrance_fee,

manager_id

from generate_series(1,100000)

cross join (select manager_id from manager) as m_id

LIMIT 10000;
```

	manager_id [PK] integer	first_name character varying	last_name character varying	email character varying	phone character varying
1	1	Ivan	Petrov	petrov.manager@gmail.com	(380) 999-1929
2	2	Petro	Sydorov	sydorov.p@gmail.com	(380) 994-6586
3	3	Anna	Bila	anya_bila@ukr.net	(380) 970-4859

```
Main menu
1. Insert
2. Update
3. Delete
4. Generate
5. Search
6. Exit
4
Table:
1. manager
2. event
1
Dataset:
Size:
5
```

	manager_id [PK] integer	first_name character varying	last_name character varying	email character varying	phone character varying
1	1	Ivan	Petrov	petrov.manager@gmail.com	(380) 999-1929
2	2	Petro	Sydorov	sydorov.p@gmail.com	(380) 994-6586
3	3	Anna	Bila	anya_bila@ukr.net	(380) 970-4859
4	2484	OR	SR	VQ	(212) 091-9035
5	60487	DH	VL	BN	(973) 565-6471
6	80895	XC	VW	EA	(144) 005-3043
7	83426	LP	MK	WE	(344) 261-0022
8	91494	WL	SX	CN	(729) 037-4118

Пункт №3

Запит №1:

```
select manager.manager_id, first_name, last_name, email, count(*) from
manager

join "event"

on event.manager_id = manager.manager_id

where first_name like 'Ivan'

group by manager.manager_id, first_name, last_name, email;
```

```
Select a search query:
1. Search count of events by manager's name
2. Search venues by count of events
3. Search venues by date
1
Search params:
first_name:
Ivan
Time: 20384000ns
manager_id | first_name | last_name | email | count
-----+-----+-----+-----+-----
1 | Ivan | Petrov | petrov.manager@gmail.com | 3
```

Запит №2

```
select venue.venue_id, venue.name, venue.capacity, count(event.name) as
events from venue

join "events_venue"

on venue.venue_id = events_venue.venue_id

join event

on events_venue.event_name = event.name

group by venue.venue_id, venue.name, venue.capacity

having count(event.name) = 2;
```

```
Select a search query:
1. Search count of events by manager's name
2. Search venues by count of events
3. Search venues by date
2
Search params:
events:
2
Time: 5519500ns
venue_id | name | capacity | events
-----+-----+-----+-----
3 | Origin stage | 1000 | 2
1 | Palats Sportu | 13000 | 2
```

Занум №3

```
select venue.venue_id, venue.name, venue.capacity, count(event.name) as
events from venue

left join "events_venue"

on venue.venue_id = events_venue.venue_id

left join event

on events_venue.event_name = event.name

where date between '2024-01-01' and '2025-01-01'

group by venue.venue_id, venue.name;
```

Select a search query:

1. Search count of events by manager's name
2. Search venues by count of events
3. Search venues by date

3

Search params:

first:

2024-01-01

last:

2025-01-01

Time: 1761200ns

venue_id	name	capacity	events
-----+-----+-----+-----			
3	Origin stage	1000	2
1	Palats Sportu	13000	2

Пункт №4

Ілюстрації програмного коду модуля “Model”, згідно із шаблоном MVC

```
1 usage
public Model() throws SQLException, ClassNotFoundException {
    connection = connect();
}

1 usage
public Connection connect() throws SQLException, ClassNotFoundException {
    Class.forName(DRIVER);

    Connection c = DriverManager.getConnection(URL, USERNAME, PASSWORD);
    return c;
}

1 usage
public void disconnect() throws SQLException {
    connection.close();
}
```

Конструктор та методи для підключення до БД та відключення

```
public void insert(String tableName, Map<String, String> columns) {
    StringBuilder q = new StringBuilder("INSERT INTO \"" + tableName + "\"(");
    StringBuilder v = new StringBuilder(" VALUES (");
    int i = 0;
    for (Map.Entry<String, String> columnData : columns.entrySet()) {
        q.append(columnData.getKey()).append(i == columns.size() - 1 ? "" : ",");
        v.append("").append(columnData.getValue()).append("").append(i == columns.size() - 1 ? "" : ",");
        i++;
    }
    q.append(")");
    v.append(")");
    q.append(v);
    System.out.println(q.toString());
    try {
        Statement statement = connection.createStatement();
        statement.execute(q.toString());
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

Метод генерації insert-запиту

```

public void update(String t, Map<String, String> pk, Map<String, String> c) {
    StringBuilder q = new StringBuilder("UPDATE \"" + t + "\" SET ");
    int i = 0;
    for (Map.Entry<String, String> columnData : c.entrySet()) {
        q.append(columnData.getKey())
          .append("=").append(columnData.getValue()).append("'")
          .append(i == c.size() - 1 ? "" : ",");
        i++;
    }
    q.append(" WHERE ");
    i = 0;
    for (Map.Entry<String, String> columnData : pk.entrySet()) {
        q.append(columnData.getKey())
          .append("=").append(columnData.getValue()).append("'")
          .append(i == pk.size() - 1 ? "" : ",");
        i++;
    }
    System.out.println(q);
    try {
        Statement statement = connection.createStatement();
        statement.execute(q.toString());
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}

```

метод для генерації update-запиту

```

public void delete(String t, Map<String, String> pk) {
    StringBuilder q = new StringBuilder("DELETE FROM \"" + t + "\" WHERE ");
    int i = 0;
    for (Map.Entry<String, String> columnData : pk.entrySet()) {
        q.append(columnData.getKey())
          .append("=").append(columnData.getValue()).append("'")
          .append(i == pk.size() - 1 ? "" : " AND ");
        i++;
    }
    System.out.println(q.toString());
    try {
        Statement statement = connection.createStatement();
        statement.execute(q.toString());
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}

```

метод для генерації delete-запису

```

public void generate(String table, int size) {
    String query = "";
    if(table.equals("manager")) {
        query = "insert into manager(manager_id, first_name, last_name, email, phone) \n" +
            "select * from (select (random()*100000)::int as manager_id,\n" +
            "    chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int) as first_name,\n" +
            "    chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int) as last_name,\n" +
            "    chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int) as email,\n" +
            "    to_char(random() * 10000000000, 'FM(000) 000-0000') as phone\n" +
            "    from generate_series(1,100000)) as _\n" +
            "where (manager_id not in (select manager_id from manager))\n" +
            "limit " + size + " ";
    }
    else if(table.equals("event")) {
        query = "INSERT INTO \"event\"(name, date, entrance_fee, manager_id) " +
            "SELECT chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int) as name,\n" +
            "    DATE(CURRENT_DATE + random() * interval '10 months') as date,\n" +
            "    (random() * 100)::int as entrance_fee,\n" +
            "    manager_id\n" +
            "    from generate_series(1,100000)\n" +
            "    cross join (select manager_id from manager) as m_id" +
            " LIMIT " + size + " ";
    }
}

else throw new RuntimeException("Invalid table name");
try {
    connection.createStatement().execute(query);
} catch (SQLException e) {
    throw new RuntimeException(e);
}
}

```

метод для генерації випадкових даних

```

public List<String> search(Map<String, String> params, int queryNum) {
    List<String> rows = new ArrayList<>();
    String query = switch (queryNum) {
        case 1 -> "select manager.manager_id, first_name, last_name, email, count(*) from manager\n" +
            "    join \"event\"\n" +
            "    on event.manager_id = manager.manager_id\n" +
            "    where first_name like " + params.get("first_name") + " '\n" +
            "    group by manager.manager_id, first_name, last_name, email";
        case 2 -> "select venue.venue_id, venue.name, venue.capacity, count(event.name) as events from venue\n" +
            "join \"events_venue\"\n" +
            "on venue.venue_id = events_venue.venue_id\n" +
            "join event\n" +
            "    on events_venue.event_name = event.name\n" +
            "group by venue.venue_id, venue.name, venue.capacity\n" +
            "having count(event.name) = " + params.get("events");
        case 3 -> "select venue.venue_id, venue.name, venue.capacity, count(event.name) as events from venue\n" +
            "left join \"events_venue\"\n" +
            "on venue.venue_id = events_venue.venue_id\n" +
            "left join event\n" +
            "    on events_venue.event_name = event.name\n" +
            "where date between " + params.get("first") + " and " + params.get("last") + "\n" +
            "group by venue.venue_id, venue.name";
        default -> throw new RuntimeException("Invalid option");
    };
}

```

```

try {
    ResultSet resultSet = connection.createStatement().executeQuery(query);
    switch (queryNum) {
        case 1 -> {
            rows.add(String.format("%11s | %15s | %15s | %26s | %9s", "manager_id", "first_name", "last_name", "email", "count"));
            rows.add("-----+-----+-----+-----+-----");
        }
        case 2, 3 -> {
            rows.add(String.format("%11s | %15s | %15s | %20s ", "venue_id", "name", "capacity", "events"));
            rows.add("-----+-----+-----+-----");
        }
    }

    while (resultSet.next()) {
        switch (queryNum) {
            case 1 -> {
                int managerId, count;
                String firstName, lastName, email;
                managerId = resultSet.getInt( columnLabel: "manager_id");
                firstName = resultSet.getString( columnLabel: "first_name");
                lastName = resultSet.getString( columnLabel: "last_name");
                email = resultSet.getString( columnLabel: "email");
                count = resultSet.getInt( columnLabel: "count");
                rows.add(String.format("%11d | %15s | %15s | %26s | %9d", managerId, firstName, lastName, email, count));
            }
            case 2, 3 -> {
                int venueId, events, capacity;
                String name = resultSet.getString( columnLabel: "name");
                venueId = resultSet.getInt( columnLabel: "venue_id");
                events = resultSet.getInt( columnLabel: "events");
                capacity = resultSet.getInt( columnLabel: "capacity");

                rows.add(String.format("%11d | %15s | %15d | %20d ", venueId, name, capacity, events));
            }
        }
    }
} catch (SQLException e) {
    throw new RuntimeException(e);
}

System.out.println("Time: " + (end - start) + "ns");
return rows;
}

```

метод для пошуку даних різними пошуковими запитами

