

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”

Виконав: студент III курсу

ФПМ групи KB-94

Анохін К.

Київ – 2021

Метою роботи є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL. Загальне завдання роботи полягає у наступному:

1. Реалізувати функції перегляду, внесення, редагування та видалення даних у таблицях бази даних, створених у лабораторній роботі No1, 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).

### Логічна модель бази даних

Нижче (Рис. 1) наведено логічну модель бази даних:

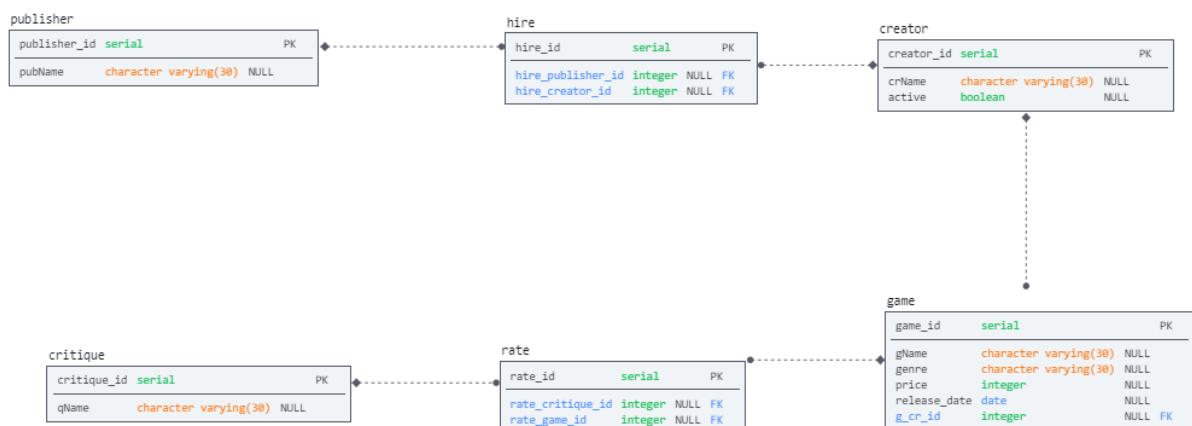


Рис. 1 – Логічна модель бази даних



## Середовище розробки та налаштування підключення до бази даних

Для виконання лабораторної роботи використовувалась мова програмування TypeScript та текстовий редактор Visual Studio Code.

Для підключення до серверу бази даних PostgreSQL використовувався npm пакет **pg**. Для цього він підключався до проекту, з нього було «дістано» клас **Pool** на його основі було створено змінну для керування базою даних:

```
const { Pool } = require('pg');

const pool = new Pool({
  user: 'postgres',
  host: '127.0.0.1',
  database: 'games',
  password: 'qwerty',
  port: 5432
});

module.exports = pool;
```

## Опис структури програми

Програма складається із: 8 моделей, що містять класи для взаємодії з базою даних, 3 допоміжних модуля в папці **utils**, модуль **view** для відображення меню, модуль **db**, що забезпечує підключення до бази даних, модуль **types** із оголошеними типами для зручності роботи з таблицями, **controller**, який містить в собі меню та забезпечує взаємодію моделей та представлення, модуль **main**, який запускає метод модуля **controller** та розпочинає роботу програми.

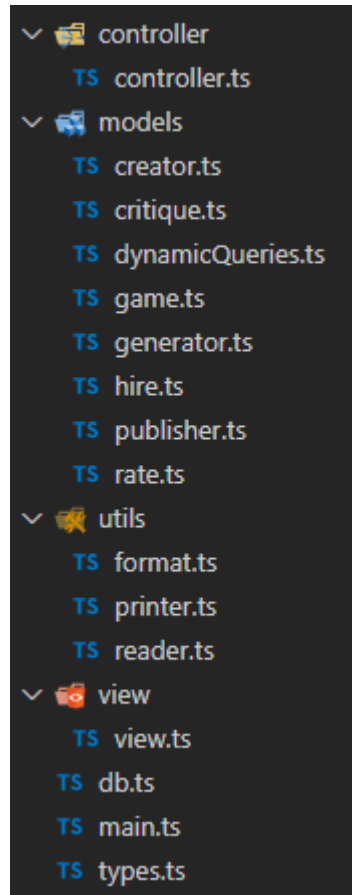


Рис. 2 – Структура програмного коду

## Структура меню програми

### Головне меню

```
Database controller program
```

1. Table Creator
2. Table Critique
3. Table Game
4. Table Hire
5. Table Publisher
6. Table Rate
7. Generate rows
8. Search dynamic

```
input: 
```

### Меню для таблиці

```
Table Creator:
```

1. add data
2. edit data
3. remove data
4. show data

```
input: 
```

### Меню для вибору динамічних запитів

1. Select creators by publisher name
2. Select games by creator name
3. Show top critiques by rate count

```
input: 
```

### Меню вибору кількості рядків для генерації

```
input: 7
```

```
number of records: 
```

## Посилання для навігації

1. Лістинги програми з директивами внесення, редагування та вилучення даних у базі даних та результати виконання цих директив.
  - a. Функції внесення
  - b. Функції редагування
  - c. Функції видалення
2. Лістинги програми з директивами рандомізованого внесення даних до таблиці Game та результати.
3. Лістинги програми з директивами динамічних запитів та валідації даних.
4. Обробка виняткових ситуацій(помилки) при введенні/вилученні та валідації даних.
5. Дослідження режимів ON DELETE.
6. Ілюстрації програмного коду на Github.

## Лістинги програми з директивами внесення, редагування та вилучення даних у базі даних та результати виконання цих директив

Для кожної таблиці було створено клас-модель, що містить методи для роботи з відповідною таблицею бази даних.

### Функції внесення

Додавання запису у таблицю Creator:

```
static async addDataCreator() {
  try {
    const text: string = 'INSERT INTO "creator" ("crName", "active") VALUES ($1, $2)';
    const creator: Creator = Reader.prepareDataCreator();

    await pool.query(text, [creator.crName, creator.active]);

    console.log('Added 1 row to table Creator');
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 1
creator name: new creator
active creator: 1
Added 1 row to table Creator
```

Результат:

	creator_id [PK] integer	crName character varying (30)	active boolean
1	1	Riot Games	true
2	2	Blizzard North	true
3	3	Nintendo R&D1	false
4	4	BioWare	true
5	5	Piranha Bytes	false
6	6	Vatra Games	false
7	7	Epic Games	true
8	10	new creator	true



Додавання запису у таблицю Critique:

```
static async addDataCritique() {
  try {
    const text: string = 'INSERT INTO "critique" ("qName") VALUES ($1)';
    const critique: Critique = Reader.prepareDataCritique();

    await pool.query(text, [critique.qName]);

    console.log('Added 1 row to table Critique');
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 1
critique name: new critique
Added 1 row to table Critique
```

Результат:

	 critique_id [PK] integer 	qName character varying (30) 
1	1	MatPat
2	2	Niko Nirvi
3	3	TotalBiscuit
4	4	Seanbaby
5	5	Jeff Rovin
6	6	JonTron
7	7	Arin Hanson
8	9	new critique

## Додавання запису у таблицю Game:

```
static async addDataGame() {
  try {
    const text: string = 'INSERT INTO "game" ("gName", "genre", "price", "release_date", "g_cr_id") VALUES ($1, $2, $3, $4, $5)';
    const game: Game = Reader.prepareDataGame();
    const checkCr: string = 'SELECT * FROM "creator" WHERE "creator_id" = $1';
    const crObj = await pool.query(checkCr, [game.g_cr_id]);

    if (!crObj.rows.length) {
      console.log('There is no creator with id ${game.g_cr_id}');
    } else {
      await pool.query(text, [game.gName, game.genre, game.price, game.release_date, game.g_cr_id]);

      console.log('Added 1 element to table Game');
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 1
game name: new game
game genre: any
game price: 12
creator id: 10
Added 1 element to table Game
```

## Результат:

	game_id [PK] integer	gName character varying (30)	genre character varying (30)	price integer	release_date date	g_cr_id integer
1	1	League of Legends	MOBA	0	2009-10-27	1
2	2	Diablo II	ARPG	10	2000-06-29	2
3	3	Donkey Kong	Platform	11	1981-07-09	3
4	4	Dragon Age:Origins	Role-playing	5	2009-11-03	4
5	5	Risen	ARPG	6	2009-10-02	5
6	6	Silent Hill	Survival Horror	60	1999-01-23	6
7	7	Fortnite	Battle Rotale	0	2017-07-25	7
8	13	new game	any	12	2021-01-01	10

Додавання запису у таблицю Publisher:

```
static async addDataPublisher() {  
  try {  
    const text: string = 'INSERT INTO "publisher" ("pubName") VALUES ($1)';  
    const publisher: Publisher = Reader.prepareDataPublisher();  
  
    await pool.query(text, [publisher.pubName]);  
  
    console.log('Added 1 row to table Publisher');  
  } catch (err) {  
    console.log(err);  
  }  
}
```

```
input: 1  
publisher name: new publisher  
Added 1 row to table Publisher
```

Результат:

		<b>publisher_id</b> [PK] integer 	<b>pubName</b> character varying (30) 
1		1	Tencent
2		2	BLizzard Entertainment
3		3	EA
4		4	Deep Silver
5		5	Konami
6		6	Epic Games
7		7	Nintendo
8		9	new publisher

Додавання запису у таблицю Hire:

```
static async addDataHire() {
  try {
    const text: string = 'INSERT INTO "hire" ("hire_publisher_id", "hire_creator_id") VALUES ($1, $2)';
    const hire: Hire = Reader.prepareDataHire();
    const checkPub: string = 'SELECT * FROM "publisher" WHERE "publisher_id" = $1';
    const checkCr: string = 'SELECT * FROM "creator" WHERE "creator_id" = $1';
    const pubObj = await pool.query(checkPub, [hire.hire_publisher_id]);
    const crObj = await pool.query(checkCr, [hire.hire_creator_id]);

    if (!pubObj.rows.length || !crObj.rows.length) {
      console.log(`There is no publisher with id ${hire.hire_publisher_id} or creator with id ${hire.hire_creator_id}`);
    } else {
      await pool.query(text, [hire.hire_publisher_id, hire.hire_creator_id]);

      console.log('Added 1 row to table Hire');
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 1
creator id: 10
publisher id: 9
Added 1 row to table Hire
```

Результат:

	hire_id [PK] integer	hire_publisher_id integer	hire_creator_id integer
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	9	1	3
9	10	9	10

Додавання запису у таблицю Rate:

```
static async addDataRate() {
  try {
    const text: string = 'INSERT INTO "rate" ("rate_game_id", "rate_critique_id") VALUES ($1, $2)';
    const rate: Rate = Reader.prepareDataRate();
    const checkGame: string = 'SELECT * FROM "game" WHERE "game_id" = $1';
    const checkCritique: string = 'SELECT * FROM "critique" WHERE "critique_id" = $1';
    const crObj = await pool.query(checkCritique, [rate.rate_critique_id]);
    const gameObj = await pool.query(checkGame, [rate.rate_game_id]);

    if (!crObj.rows.length || !gameObj.rows.length) {
      console.log(`There is no game with id ${rate.rate_game_id} or critique with id ${rate.rate_critique_id}`);
    } else {
      await pool.query(text, [rate.rate_game_id, rate.rate_critique_id]);

      console.log('Added 1 row to table Rate');
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 1
game id: 13
critique id: 9
Added 1 row to table Rate
```

Результат:

	rate_id [PK] integer	rate_game_id integer	rate_critique_id integer
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	9	7	2
9	10	13	9

## Функції редагування

Редагування запису в таблиці Creator:

```
static async editDataCreator() {
  try {
    const id: number = +question('creator id: ');
    const check: string = 'SELECT * FROM "creator" WHERE "creator_id" = $1';
    const text: string = 'UPDATE "creator" SET "crName" = $1, "active" = $2 WHERE "creator_id" = $3';
    const crCheck = await pool.query(check, [id]);

    if (!crCheck.rows.length) {
      console.log(`There is no creator with id ${id}`);
    } else {
      const creator: Creator = Reader.prepareDataCreator();

      await pool.query(text, [creator.crName, creator.active, id]);

      console.log(`Row with id ${id} has been updated`);
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 2
creator id: 10
creator name: new name
active creator:
Row with id 10 has been updated
```

Результат:

	 creator_id [PK] integer 	crName character varying (30) 	active boolean 
1	1	Riot Games	true
2	2	Blizzard North	true
3	3	Nintendo R&D1	false
4	4	BioWare	true
5	5	Piranha Bytes	false
6	6	Vatra Games	false
7	7	Epic Games	true
8	10	new name	false

## Редагування запису в таблиці Critique:

```
static async editDataCritique() {
  try {
    const id: number = +question('critique id: ');
    const check: string = 'SELECT * FROM "critique" WHERE "critique_id" = $1';
    const text: string = 'UPDATE "critique" SET "qName" = $1 WHERE "critique_id" = $2';
    const critiqueCheck = await pool.query(check, [id]);

    if (!critiqueCheck.rows.length) {
      console.log(`There is no critique with id ${id}`);
    } else {
      const critique: Critique = Reader.prepareDataCritique();

      await pool.query(text, [critique.qName, id]);

      console.log(`Row with id ${id} has been updated`);
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 2
critique id: 9
critique name: test name
Row with id 9 has been updated
```

## Результат:

	 critique_id [PK] integer	 qName character varying (30)
1	1	MatPat
2	2	Niko Nirvi
3	3	TotalBiscuit
4	4	Seanbaby
5	5	Jeff Rovin
6	6	JonTron
7	7	Arin Hanson
8	9	test name

## Редагування запису в таблиці Game:

```
static async editDataGame() {
  try {
    const id: number = +question('game id: ');
    const check: string = 'SELECT * from "game" WHERE "game_id" = $1';
    const text: string = 'UPDATE "game" SET "gName" = $1, "genre" = $2, "price" = $3, "release_date" = $4, "g_cr_id" = $5 WHERE "game_id" = $6';
    const gameCheck = await pool.query(check, [id]);

    if (!gameCheck.rows.length) {
      console.log(`There is no game with id ${id}`);
    } else {
      const game: Game = Reader.prepareDataGame();
      const checkCr: string = 'SELECT * FROM "creator" WHERE "creator_id" = $1';
      const crObj = await pool.query(checkCr, [game.g_cr_id]);

      if (!crObj.rows.length) {
        console.log(`There is no creator with id ${game.g_cr_id}`);
      } else {
        await pool.query(text, [game.gName, game.genre, game.price, game.release_date, game.g_cr_id, id]);
        console.log(`Row with id ${id} has been updated`);
      }
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 2
game id: 13
game name: new game
game genre: no genre
game price: 0
creator id: 10
Row with id 13 has been updated
```

## Результат:

	game_id [PK] integer	gName character varying (30)	genre character varying (30)	price integer	release_date date	g_cr_id integer
1	1	League of Legends	MOBA	0	2009-10-27	1
2	2	Diablo II	ARPG	10	2000-06-29	2
3	3	Donkey Kong	Platform	11	1981-07-09	3
4	4	Dragon Age:Origins	Role-playing	5	2009-11-03	4
5	5	Risen	ARPG	6	2009-10-02	5
6	6	Silent Hill	Survival Horror	60	1999-01-23	6
7	7	Fortnite	Battle Royale	0	2017-07-25	7
8	13	new game	no genre	0	2021-01-01	10



## Редагування запису в таблиці Publisher:

```
static async editDataPublisher() {
  try {
    const id: number = +question('publisher id: ');
    const check: string = 'SELECT * FROM "publisher" WHERE "publisher_id" = $1';
    const text: string = 'UPDATE "publisher" SET "pubName" = $1 WHERE "publisher_id" = $2';
    const pubCheck = await pool.query(check, [id]);

    if (!pubCheck.rows.length) {
      console.log(`There is no publisher with id ${id}`);
    } else {
      const publisher: Publisher = Reader.prepareDataPublisher();

      await pool.query(text, [publisher.pubName, id]);

      console.log(`Row with id ${id} has been updated`);
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 2
publisher id: 9
publisher name: test pub
Row with id 9 has been updated
```

## Результат:

	<b>publisher_id</b> [PK] integer	<b>pubName</b> character varying (30)
1	1	Tencent
2	2	BLizzard Entertainment
3	3	EA
4	4	Deep Silver
5	5	Konami
6	6	Epic Games
7	7	Nintendo
8	9	new publisher

## Редагування запису в таблиці Hire:

```
static async editDataHire() {
  try {
    const id: number = +question('hire id: ');
    const text: string = 'UPDATE "hire" SET "hire_publisher_id" = $1, "hire_creator_id" = $2 WHERE "hire_id" = $3';
    const hire: Hire = Reader.prepareDataHire();
    const checkHire: string = 'SELECT * FROM "hire" WHERE "hire_id" = $1';
    const hireObj = await pool.query(checkHire, [id]);

    if (!hireObj.rows.length) {
      console.log(`There is no hire with id ${id}`);
    } else {
      const checkPub: string = 'SELECT * FROM "publisher" WHERE "publisher_id" = $1';
      const checkCr: string = 'SELECT * FROM "creator" WHERE "creator_id" = $1';
      const pubObj = await pool.query(checkPub, [hire.hire_publisher_id]);
      const crObj = await pool.query(checkCr, [hire.hire_creator_id]);

      if (!pubObj.rows.length || !crObj.rows.length) {
        console.log(`There is no publisher with id ${hire.hire_publisher_id} or creator with id ${hire.hire_creator_id}`);
      } else {
        await pool.query(text, [hire.hire_publisher_id, hire.hire_creator_id, id]);

        console.log(`Row with id ${id} has been updated`);
      }
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 2
hire id: 10
creator id: 4
publisher id: 9
Row with id 10 has been updated
```

## Результат:

	hire_id [PK] integer	hire_publisher_id integer	hire_creator_id integer
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	9	1	3
9	10	9	4

## Редагування запису в таблиці Rate:

```
static async editDataRate() {
  try {
    const id: number = +question('rate id: ');
    const text: string = 'UPDATE "rate" SET "rate_game_id" = $1, "rate_critique_id" = $2 WHERE "rate_id" = $3';
    const rate: Rate = Reader.prepareDataRate();
    const checkRate: string = 'SELECT * FROM "rate" WHERE "rate_id" = $1';
    const checkGame: string = 'SELECT * FROM "game" WHERE "game_id" = $1';
    const checkCritique: string = 'SELECT * FROM "critique" WHERE "critique_id" = $1';
    const rateObj = await pool.query(checkRate, [id]);

    if (!rateObj.rows.length) {
      console.log(`There is no rate with id ${id}`);
    } else {
      const crObj = await pool.query(checkCritique, [rate.rate_critique_id]);
      const gameObj = await pool.query(checkGame, [rate.rate_game_id]);

      if (!crObj.rows.length || !gameObj.rows.length) {
        console.log(`There is no game with id ${rate.rate_game_id} or critique with id ${rate.rate_critique_id}`);
      } else {
        await pool.query(text, [rate.rate_game_id, rate.rate_critique_id, id]);

        console.log(`Row with id ${id} has been updated`);
      }
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 2
rate id: 10
game id: 7
critique id: 9
Row with id 10 has been updated
```

## Результат:

	rate_id [PK] integer	rate_game_id integer	rate_critique_id integer
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	9	7	2
9	10	7	9

## Функції видалення

Видалення в таблиці Hire:

```
static async deleteDataHire() {  
  try {  
    const id: number = +question('hire id: ');  
    const text: string = 'DELETE FROM "hire" WHERE "hire_id" = $1';  
    const checkHire: string = 'SELECT * FROM "hire" WHERE "hire_id" = $1';  
    const hireObj = await pool.query(checkHire, [id]);  
  
    if (!hireObj.rows.length) {  
      console.log(`There is no hire with id ${id}`);  
    } else {  
      await pool.query(text, [id]);  
  
      console.log(`Row with id ${id} has been deleted`);  
    }  
  } catch (err) {  
    console.log(err);  
  }  
}
```

```
input: 3  
hire id: 10  
Row with id 10 has been deleted
```

Результат:

	hire_id [PK] integer	hire_publisher_id integer	hire_creator_id integer
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	9	1	3

Видалення в таблиці Rate:

```
static async deleteDataRate() {
  try {
    const id: number = +question('rate id: ');
    const text: string = 'DELETE FROM "rate" WHERE "rate_id" = $1';
    const checkRate: string = 'SELECT * FROM "rate" WHERE "rate_id" = $1';
    const rateObj = await pool.query(checkRate, [id]);

    if (!rateObj.rows.length) {
      console.log(`There is no rate with id ${id}`);
    } else {
      await pool.query(text, [id]);

      console.log(`Row with id ${id} has been deleted`);
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 3
rate id: 10
Row with id 10 has been deleted
```

Результат:

	rate_id [PK] integer	rate_game_id integer	rate_critique_id integer
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	9	7	2

## Видалення в таблиці Game:

```
static async deleteDataGame() {
  try {
    const id: number = +question('game id: ');
    const check: string = 'SELECT * from "game" WHERE "game_id" = $1';
    const text: string = 'DELETE FROM "game" WHERE "game_id" = $1';
    const gameCheck = await pool.query(check, [id]);

    if (!gameCheck.rows.length) {
      console.log(`There is no game with id ${id}`);
    } else {
      await pool.query(text, [id]);

      console.log(`Row with id ${id} has been deleted`);
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 3
game id: 13
Row with id 13 has been deleted
```

## Результат:

	game_id [PK] integer	gName character varying (30)	genre character varying (30)	price integer	release_date date	g_cr_id integer
1	1	League of Legends	MOBA	0	2009-10-27	1
2	2	Diablo II	ARPG	10	2000-06-29	2
3	3	Donkey Kong	Platform	11	1981-07-09	3
4	4	Dragon Age:Origins	Role-playing	5	2009-11-03	4
5	5	Risen	ARPG	6	2009-10-02	5
6	6	Silent Hill	Survival Horror	60	1999-01-23	6
7	7	Fortnite	Battle Royale	0	2017-07-25	7

Видалення в таблиці Creator:

```
static async deleteDataCreator() {
  try {
    const id: number = +question('creator id: ');
    const check: string = 'SELECT * FROM "creator" WHERE "creator_id" = $1';
    const text: string = 'DELETE FROM "creator" WHERE "creator_id" = $1';
    const crCheck = await pool.query(check, [id]);

    if (!crCheck.rows.length) {
      console.log(`There is no creator with id ${id}`);
    } else {
      await pool.query(text, [id]);

      console.log(`Row with id ${id} has been deleted`);
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 3
creator id: 10
Row with id 10 has been deleted
```

Результат:

	creator_id [PK] integer	crName character varying (30)	active boolean
1	1	Riot Games	true
2	2	Blizzard North	true
3	3	Nintendo R&D1	false
4	4	BioWare	true
5	5	Piranha Bytes	false
6	6	Vatra Games	false
7	7	Epic Games	true

Видалення в таблиці Publisher:

```
static async deleteDataPublisher() {
  try {
    const id: number = +question('publisher id: ');
    const check: string = 'SELECT * FROM "publisher" WHERE "publisher_id" = $1';
    const text: string = 'DELETE FROM "publisher" WHERE "publisher_id" = $1';
    const pubCheck = await pool.query(check, [id]);

    if (!pubCheck.rows.length) {
      console.log(`There is no publisher with id ${id}`);
    } else {
      await pool.query(text, [id]);

      console.log(`Row with id ${id} has been deleted`);
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 3
publisher id: 9
Row with id 9 has been deleted
```

Результат:

	<b>publisher_id</b> [PK] integer	<b>pubName</b> character varying (30)
1	1	Tencent
2	2	BLizzard Entertainment
3	3	EA
4	4	Deep Silver
5	5	Konami
6	6	Epic Games
7	7	Nintendo



## Видалення в таблиці Critique:

```
static async deleteDataCritique() {
  try {
    const id: number = +question('critique id: ');
    const check: string = 'SELECT * FROM "critique" WHERE "critique_id" = $1';
    const text: string = 'DELETE FROM "critique" WHERE "critique_id" = $1';
    const critiqueCheck = await pool.query(check, [id]);

    if (!critiqueCheck.rows.length) {
      console.log(`There is no critique with id ${id}`);
    } else {
      await pool.query(text, [id]);

      console.log(`Row with id ${id} has been deleted`);
    }
  } catch (err) {
    console.log(err);
  }
}
```

```
input: 3
critique id: 9
Row with id 9 has been deleted
```

## Результат:

	<b>critique_id</b> [PK] integer	<b>qName</b> character varying (30)
1	1	MatPat
2	2	Niko Nirvi
3	3	TotalBiscuit
4	4	Seanbaby
5	5	Jeff Rovin
6	6	JonTron
7	7	Arin Hanson

# Лістинги програми з директивами внесення рандомізованих даних і виконання динамічних запитів у базі даних та результати виконання цих директив

## Рандомізоване внесення даних до таблиці Game

Було створено клас Generate із методом generateGames. Задана кількість записів генерується у таблиці **Game**:

```
static async generateGames() {
  try {
    const num: number = +question('number of records: ');
    const text: string = `
      insert into "game" ("gName", "genre", "price", "release_date")
      select substr(characters, (random() * length(characters) + 1)::integer, 10),
      substr(characters, (random() * length(characters) + 1)::integer, 10),
      trunc(random() * 100)::int,
      timestamp '2018-01-10' + random() * (timestamp '2018-01-20' - timestamp '2018-01-10')
      from (values('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) as symbols(characters), generate_series(1, $1);
    `;

    const start: number = Date.now();
    await pool.query(text, [num]);
    const queryTime: number = Date.now() - start;

    console.log(`${num} rows has been generated in table Game`);
    console.log(`query time: ${queryTime}ms`);
  } catch (err) {
    console.log(err);
  }
}
```

Таблиця Game до генерації:

	game_id [PK] integer	gName character varying (30)	genre character varying (30)	price integer	release_date date	g_cr_id integer
1	1	League of Legends	MOBA	0	2009-10-27	1
2	2	Diablo II	ARPG	10	2000-06-29	2
3	3	Donkey Kong	Platform	11	1981-07-09	3
4	4	Dragon Age:Origins	Role-playing	5	2009-11-03	4
5	5	Risen	ARPG	6	2009-10-02	5
6	6	Silent Hill	Survival Horror	60	1999-01-23	6
7	7	Fortnite	Battle Rotale	0	2017-07-25	7

Генеруємо записи:

```
input: 7
number of records: 10
10 rows has been generated in table Game
query time: 82ms
```

Таблиця Game після генерації:

	game_id [PK] integer	gName character varying (30)	genre character varying (30)	price integer	release_date date	g_cr_id integer
1	1	League of Legends	MOBA	0	2009-10-27	1
2	2	Diablo II	ARPG	10	2000-06-29	2
3	3	Donkey Kong	Platform	11	1981-07-09	3
4	4	Dragon Age:Origins	Role-playing	5	2009-11-03	4
5	5	Risen	ARPG	6	2009-10-02	5
6	6	Silent Hill	Survival Horror	60	1999-01-23	6
7	7	Fortnite	Battle Rotale	0	2017-07-25	7
8	14	mQWERTYUIO	yuiopasdfg	86	2018-01-19	[null]
9	15	sdfghjklzx	sdfghjklzx	20	2018-01-11	[null]
10	16	fghjklzxcv	vbnmQWERTY	80	2018-01-13	[null]
11	17	LZXCVBNM	HJKLZXCVBN	44	2018-01-10	[null]
12	18	TYUIOPASDF	lzxcvbnmQW	24	2018-01-16	[null]
13	19	cvbnmQWERT	ertyuiopas	19	2018-01-12	[null]
14	20	IOPASDFGHJ	ERTYUIOPAS	97	2018-01-11	[null]
15	21	IOPASDFGHJ	HJKLZXCVBN	70	2018-01-12	[null]
16	22	lzxcvbnmQW	klzxcvbnmQ	18	2018-01-11	[null]
17	23	sdfghjklzx	SDFGHJKLZX	58	2018-01-17	[null]

## Виконання динамічних запитів бази даних

Меню вибору запиту:

1. Select creators by publisher name
2. Select games by creator name
3. Show top critiques by rate count

input:

Обрахунок часу запиту відбувається безпосередньо у кожному методі моделі, що обробляє запит. Береться поточний час ДО відсилання запиту до бази даних і виводиться різниця між поточним часом ПІСЛЯ запиту і часом ДО запиту, щоб не враховувати час на форматування виводу у консоль.

Перший запит:

```
static async specPub() {
  try {
    const publisher: string = question('publisher name: ');
    const text: string = `
      with pub as (
        select s."pubName", "hire_creator_id" from "hire" join
        (select * from "publisher" where "pubName" = $1) s on "hire_publisher_id" = s."publisher_id"
      )

      select "pubName", s."crName" from pub join "creator" s on pub."hire_creator_id" = s."creator_id";
    `;

    const start: number = Date.now();
    const result = await pool.query(text, [publisher]);
    const queryTime: number = Date.now() - start;

    if (!result.rows.length) {
      console.log('No result');
    } else {
      console.log('  pubName      |      crName      ');
      console.log('_____');

      result.rows.forEach((item: any) => {
        let modPName: string = '';
        let modCName: string = '';

        if (item.pubName.length > 17) {
          modPName = item.pubName.substr(0, 14) + '...';
        } else {
          modPName = Format.toField(17, item.pubName);
        }

        if (item.crName.length > 16) {
          modCName = item.crName.substr(0, 13) + '...';
        } else {
          modCName = Format.toField(16, item.crName);
        }

        console.log(`${modPName}|${modCName}`);
        console.log('_____');
      });
      console.log(`query time: ${queryTime}ms`);
    }
  } catch (err) {
    console.log(err);
  }
}
```

Результат:

```
input: 1
publisher name: EA
  pubName      |      crName
-----
EA              |Nintendo R&D1
-----
query time: 78ms
```

Другий запит:

```
static async gamesByCr() {
  try {
    const creator: string = question('creator name: ');
    const text: string = `
      select "gName", "genre", s."crName" from "game" join
      (select * from "creator" where "crName" = $1) s on s."creator_id" = "g_cr_id"
    `;

    const start: number = Date.now();
    const result = await pool.query(text, [creator]);
    const queryTime: number = Date.now() - start;

    if (!result.rows.length) {
      console.log('No result');
    } else {
      console.log('   gName          |   genre   |   crName   ');
      console.log('_____');

      result.rows.forEach((item: any) => {
        let modGName: string = '';
        let modGenre: string = '';
        let modCName: string = '';

        if (item.gName.length > 15) {
          modGName = item.gName.substr(0, 12) + '...';
        } else {
          modGName = Format.toField(15, item.gName);
        }

        if (item.genre.length > 14) {
          modGenre = item.genre.substr(0, 11) + '...';
        } else {
          modGenre = Format.toField(14, item.genre);
        }

        if (item.crName.length > 14) {
          modCName = item.crName.substr(0, 11) + '...';
        } else {
          modCName = Format.toField(14, item.crName);
        }

        console.log(`${modGName}|${modGenre}|${modCName}`);
        console.log('_____');
      });
      console.log(`query time: ${queryTime}ms`);
    }
  } catch (err) {
    console.log(err);
  }
}
```

Результат:

```
input: 2
creator name: Riot Games
  gName      | genre      | crName
-----
League of Le...|MOBA        |Riot Games
query time: 3ms
```

Третій запит:

```
static async topCrit() {
  try {
    const text: string = `
      select "qName", s."cnt" from "critique" join
      (select "rate_critique_id", count("rate_critique_id") as cnt from "rate"
      group by ("rate_critique_id")) as s
      on "critique_id" = s."rate_critique_id"
      order by "cnt" desc
    `;

    const start: number = Date.now();
    const result = await pool.query(text);
    const queryTime: number = Date.now() - start;

    if (!result.rows.length) {
      console.log('No result');
    } else {
      console.log('  qName      |  cnt ');
      console.log('_____');

      result.rows.forEach((item: any) => {
        let modQName: string = '';

        if (item.qName.length > 15) {
          modQName = item.qName.substr(0, 12) + '...';
        } else {
          modQName = Format.toField(15, item.qName);
        }

        console.log(`${modQName}|${item.cnt}`);
        console.log('_____');
      });
      console.log(`query time: ${queryTime}ms`);
    }
  } catch (err) {
    console.log(err);
  }
}
```



Результат:

```
input: 3
  qName      | cnt
-----
Niko Nirvi   | 2
TotalBiscuit | 1
MatPat       | 1
Jeff Rovin   | 1
JonTron      | 1
Arin Hanson  | 1
Seanbaby     | 1
query time: 3ms
```

## Обробка виняткових ситуацій (помилки) при введенні/вилученні та валідації даних

Обробка виняткових ситуацій при введенні та видаленні даних виконується за допомогою блоків try-catch та перевірочних запитів перед основним запитом.

Додавання рядка із неіснуючим зовнішнім ключем:

```
input: 1  
creator id: 23423  
publisher id: 23423  
There is no publisher with id 23423 or creator with id 23423
```

Введення рядка з полем, тип якого не відповідає дійсному:

```
input: 1  
game name: 123  
game genre: 123  
game price: aasd  
creator id: 3  
error: неверный синтаксис для типа integer: "NaN"
```







Видалення рядка, ключ якого є зовнішнім ключем іншої таблиці:

```
input: 3  
creator id: 1  
error: UPDATE или DELETE в таблице "creator" нарушает ограничение внешнего ключа "fk_creator" таблицы "hire"
```

## Дослідження режимів обмеження ON DELETE

Дослідження режимів будемо проводити на таблиці Creator (батьківська) та Game (дочірня). Використаємо запис із таблиці Game з `game_id = 7` та зовнішнім ключем `g_cr_id = 7`:

Таблиця Game:

		<b>game_id</b> [PK] integer 	<b>gName</b> character varying (30) 	<b>genre</b> character varying (30) 	<b>price</b> integer 	<b>release_date</b> date 	<b>g_cr_id</b> integer
1		1	League of Legends	MOBA	0	2009-10-27	1
2		2	Diablo II	ARPG	10	2000-06-29	2
3		3	Donkey Kong	Platform	11	1981-07-09	3
4		4	Dragon Age:Origins	Role-playing	5	2009-11-03	4
5		5	Risen	ARPG	6	2009-10-02	5
6		6	Silent Hill	Survival Horror	60	1999-01-23	6
7		7	Fortnite	Battle Royale	0	2017-07-25	7

Таблиця Creator:

		<b>creator_id</b> [PK] integer 	<b>crName</b> character varying (30) 	<b>active</b> boolean 
1		1	Riot Games	true
2		2	Blizzard North	true
3		3	Nintendo R&D1	false
4		4	BioWare	true
5		5	Piranha Bytes	false
6		6	Vatra Games	false
7		7	Epic Games	true

## Режим NO ACTION

При видаленні запису із таблиці Creator, id якого присутній в таблиці Game отримуємо повідомлення про помилку:

```
input: 3
creator id: 7
error: UPDATE или DELETE в таблице "creator" нарушает ограничение внешнего ключа "fk_creator" таблицы "game"
```

## Режим SET NULL

При видаленні запису із таблиці Creator, id якого присутній в таблиці Game, якщо на g\_cr\_id таблиці Game NOT NULL, отримуємо повідомлення:

```
input: 3
creator id: 7
error: нулевое значение в столбце "g_cr_id" нарушает ограничение NOT NULL
```

Якщо прибрати NOT NULL, то g\_cr\_id прийме значення NULL.

## Режим SET DEFAULT

```
input: 3
creator id: 7
error: нулевое значение в столбце "g_cr_id" нарушает ограничение NOT NULL
```

В налаштуваннях Creator поле DEFAULT не заповнено, SET DEFAULT намагається встановити його як null, проте це порушує обмеження поля NOT NULL.

## Режим RESTRICT

При видаленні запису із таблиці Creator, id якого присутній в таблиці Game отримуємо таке саме повідомлення про помилку, як і в режимі NO ACTION:

```
input: 3
creator id: 7
error: UPDATE или DELETE в таблице "creator" нарушает ограничение внешнего ключа "fk_creator" таблицы "game"
```

## Режим CASCADE

При видаленні запису із таблиці Creator, id якого присутній в таблиці Game, видалається запис таблиці Game та запис таблиці Creator із відповідним creator\_id:

```
input: 3
creator id: 7
Row with id 7 has been deleted
```

Таблиця Creator:

	creator_id [PK] integer	crName character varying (30)	active boolean
1	1	Riot Games	true
2	2	Blizzard North	true
3	3	Nintendo R&D1	false
4	4	BioWare	true
5	5	Piranha Bytes	false
6	6	Vatra Games	false

Таблиця Game:

	game_id [PK] integer	gName character varying (30)	genre character varying (30)	price integer	release_date date	g_cr_id integer
1	1	League of Legends	MOBA	0	2009-10-27	1
2	2	Diablo II	ARPG	10	2000-06-29	2
3	3	Donkey Kong	Platform	11	1981-07-09	3
4	4	Dragon Age:Origins	Role-playing	5	2009-11-03	4
5	5	Risen	ARPG	6	2009-10-02	5
6	6	Silent Hill	Survival Horror	60	1999-01-23	6

## **Ілюстрації програмного коду на Github**