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 КВ-94
Гераймович Д. Ю.
Перевірив:

Київ – 2021

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).

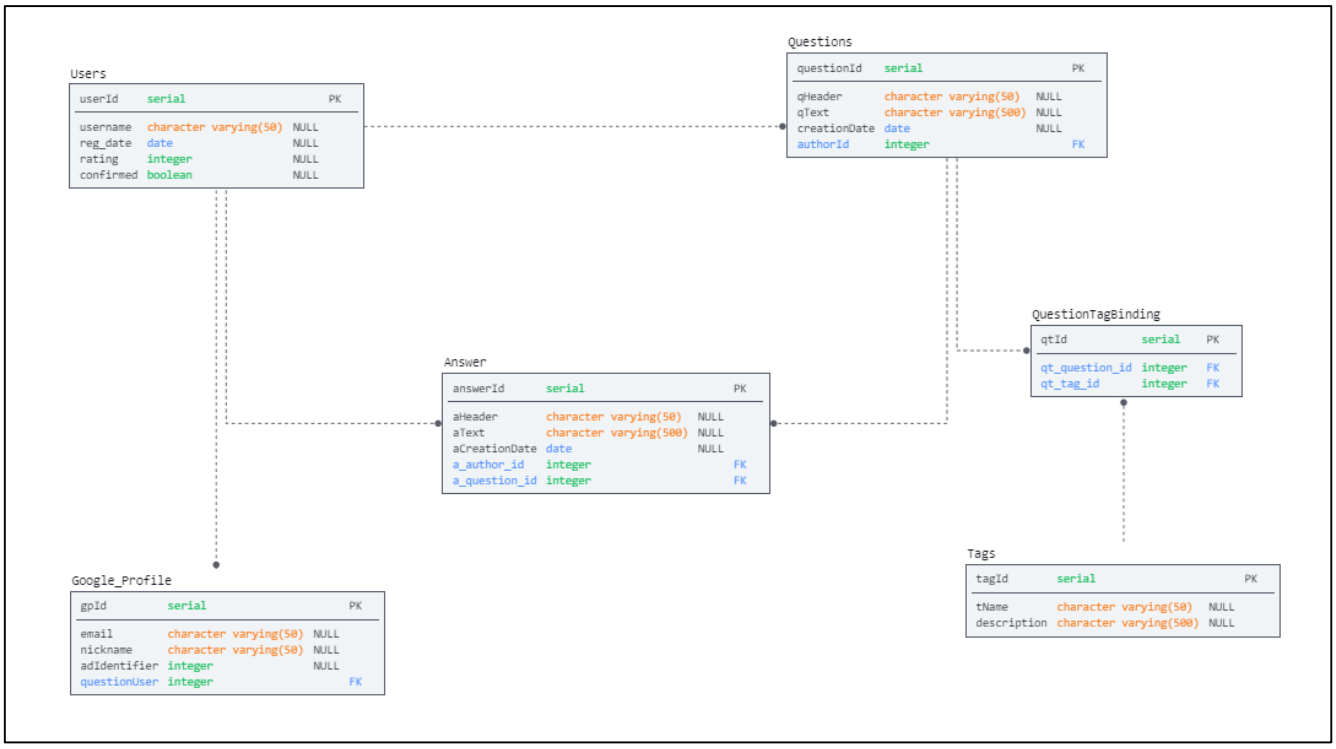
Деталізоване завдання:

1. Забезпечити можливість уведення/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 підлеглій 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**!

Логічна модель бази даних

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



Зміни у порівнянні з першою лабораторною роботою:

1. В таблицю Users додано атрибут **confirmed** типу **boolean.**
2. Для зручності, було перейменовано назви атрибутів у таблицях.

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» класс **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: 'questions',
    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.

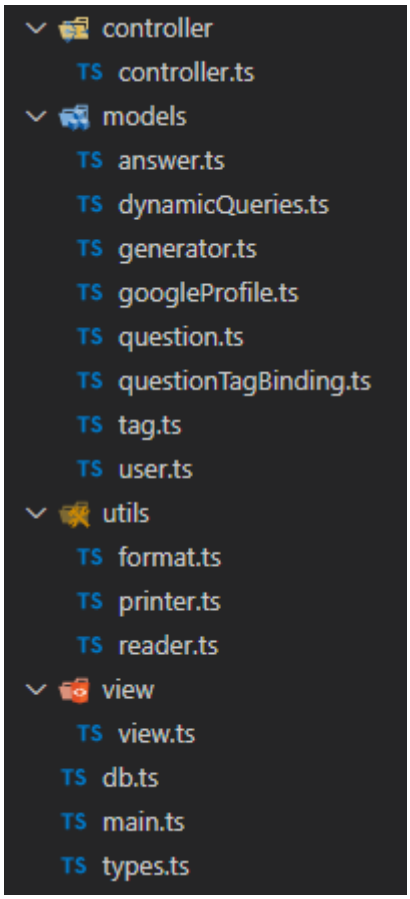
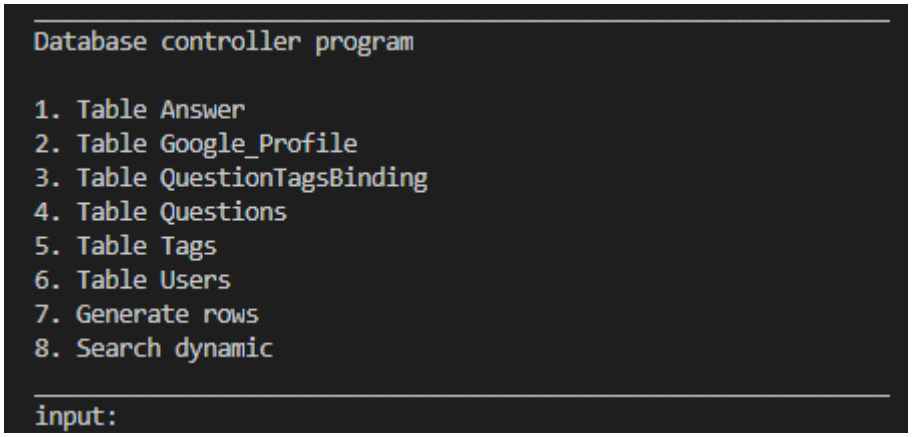


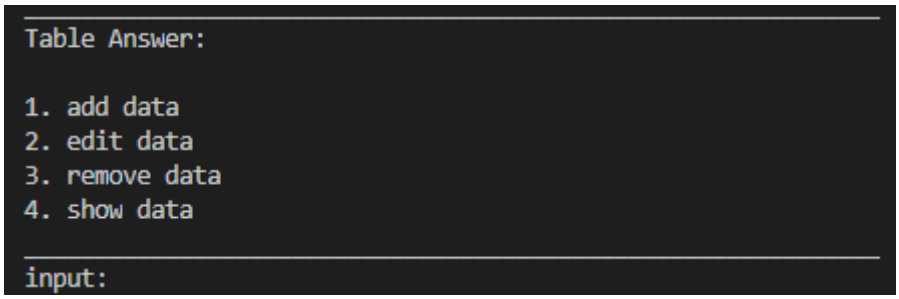
Рис. 2 – Структура програмного коду

Структура меню програми

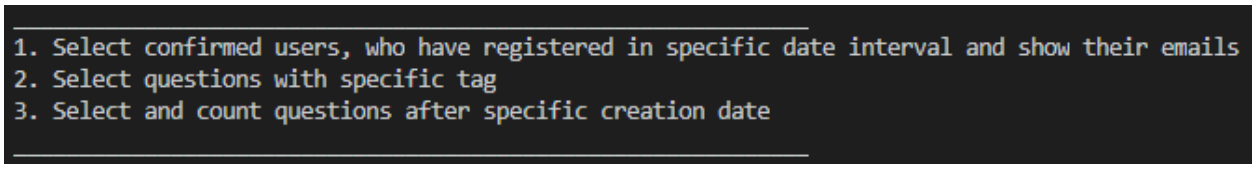
Головне меню



Меню для таблиці



Меню для вибору динамічних запитів



Меню вибору кількості рядків для генерації

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 Answer:

```
static async addDataAnswer() {
    try {
        const text: string = 'INSERT INTO "Answer" ("aHeader", "aText", "aCreationDate", "a_author_id", "a_question_id") VALUES ($1, $2, $3, $4, $5)';
        const ans: Answer = Reader.prepareDataAnswer();
        const user = await pool.query('SELECT * FROM "Users" WHERE "userId" = $1', [ans.a_author_id]);
        const quest = await pool.query('SELECT * FROM "Questions" WHERE "questionId" = $1', [ans.a_question_id]);

        if (!user.rows.length || !quest.rows.length) {
            console.log(`There is no user with id ${ans.a_author_id} or question with id ${ans.a_question_id} in database`);
        } else {
            await pool.query(text, [ans.aHeader, ans.aText, ans.aCreationDate, ans.a_author_id, ans.a_question_id]);

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

```
answer header: test header
answer text: test text
author id: 4
question id: 2
Added 1 element to table Answer
```

Результат:

	answerId [PK] integer	aHeader character varying (50)	aText character varying (500)	aCreationDate date	a_author_id integer	a_question_id integer
1	1	Try *	Example: 7*2	2019-11-02	5	1
2	2	LOL	There is no biggest number	2020-09-20	8	2
3	3	Read this article	link: https://link.com	2020-09-20	7	2
4	4	Need for speed	Underground	2020-07-24	4	3
5	5	FarCry	All chapters	2020-10-20	3	3
6	7	test header	test text	2020-12-17	4	2

Додавання запису у таблицю Google_Profile:

```
static async addDataGoogleProfile() {
    try {
        const newProfile: GoogleProfile = Reader.prepareDataGoogleProfile();
        const text: string = 'INSERT INTO "Google_Profile" ("email", "nickname", "adIdentifier", "questionUser") VALUES($1, $2, $3, $4)';

        await pool.query(text, [newProfile.email, newProfile.nickname, newProfile.adIdentifier, newProfile.questionUser]);

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

```
email: test@mail.com
nickname: testnick
ad id: 123123123
user id: 5
Added 1 row to table Google_Profile
```

Результат:

	gpId [PK] bigint	email character varying (50)	nickname character varying (50)	adIdentifier integer	questionUser integer
1	2	somemail@email.com	google_droid	123536	2
2	3	vitakdu@gmail.com	another_talk	64314	3
3	4	anothermail@mail.ru	bulbaster228	41245	4
4	5	ukrainian@i.ua	akinator	23512	7
5	6	ram@rambler.ru	muslimKA	351245	8
6	9	test@mail.com	testnick	123123123	5

Додавання запису у таблицю QuestionTagsBinding:

```
static async addDataQTB() {
    try {
        const qtb: QuestionTagsBinding = Reader.prepareDataQuestionTagsBinding();
        const text: string = 'INSERT INTO "QuestionTagsBinding" ("qt_tag_id", "qt_question_id") VALUES ($1, $2)';
        const checkTag = await pool.query('SELECT * FROM "Tags" WHERE "tagId" = $1', [qtb.qt_tag_id]);
        const checkQuestion = await pool.query('SELECT * FROM "Questions" WHERE "questionId" = $1', [qtb.qt_question_id]);

        if (!checkTag.rows.length || !checkQuestion.rows.length) {
            console.log(`There is no tag with id ${qtb.qt_tag_id} or question with id ${qtb.qt_question_id}`);
        } else {
            await pool.query(text, [qtb.qt_tag_id, qtb.qt_question_id]);

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

```
input: 1
question id: 2
tag id: 3
Added 1 row to table QuestionTagsBinding
```

Результат:

	qtId [PK] integer	qt_tag_id integer	qt_question_id integer
1	1	1	1
2	2	1	2
3	3	2	2
4	4	3	3
5	6	3	2

Додавання запису у таблицю Questions:

```
static async addDataQuestion() {
    try {
        const quest: Question = Reader.prepareDataQuestion();
        const text: string = 'INSERT INTO "Questions" ("qHeader", "qText", "creationDate", "authorId") VALUES ($1, $2, $3, $4)';
        const checkUser = await pool.query('SELECT * FROM "Users" WHERE "userId" = $1', [quest.authorId]);

        if (!checkUser.rows.length) {
            console.log(`There is no user with id ${quest.authorId}`);
        } else {
            await pool.query(text, [quest.qHeader, quest.qText, quest.creationDate, quest.authorId]);

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

```
input: 1
question header: test header
question text: test question
author id: 5
Added 1 row to table Questions
```

Результат:

	questionId [PK] integer	qHeader character varying (50)	qText character varying (500)	creationDate date	authorId integer
1	1	How to multiply in c#?	Hello! Want to multiply two di...	2019-10-02	2
2	2	What is the biggest number?	idk help	2020-09-20	3
3	3	Your favourite PC game?	Lets talk about something else	2020-07-07	5
4	5	test header	test question	2020-12-17	5

Додавання запису у таблицю Tags:

```
static async addDataTag() {
    try {
        const newTag: Tags = Reader.prepareDataTag();
        const text: string = 'INSERT INTO "Tags" ("tName", "description") VALUES($1, $2)';

        await pool.query(text, [newTag.tName, newTag.description]);

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

```
input: 1
tName: test tag
description: test description
Added 1 row to table Tags
```

Результат:

	tagId [PK] integer	tName character varying (50)	description character varying (500)
1	1	Programming	Here you can ask about progr...
2	2	Mathematics	Algebra, geometry, etc.
3	3	Free themes	Everything you want
4	4	Biology	Natural things
5	6	test tag	test description

Додавання запису у таблицю Users:

```
static async addDataUser() {
    try {
        const newUser: Users = Reader.prepareDataUser();
        const text: string = 'INSERT INTO "Users" (username, reg_date, rating) VALUES($1, $2, $3);';

        await pool.query(text, [newUser.username, newUser.reg_date, newUser.rating]);

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

```
input: 1
username: test username
rating: 122
Added 1 row to table Users
```

Результат:

	userId [PK] integer	username character varying (50)	reg_date date	rating integer	confirmed boolean
4	5	somethinganother	2019-02-14	135	false
5	6	bulek	2004-04-02	3	true
6	7	terminator228	2018-02-03	19	false
7	8	teremok2	2002-02-02	86	false
8	10	rtyuiopasd	2018-01-10	895	true
9	11	uiopasdfgh	2018-01-11	472	false
10	12	zxcvbnmQWE	2018-01-18	460	false
11	13	QWERTYUIOP	2018-01-18	163	false
12	14	fghjklzxcv	2018-01-18	242	true
13	15	rtyuiopasd	2018-01-12	95	false
14	16	test username	2020-12-17	122	false

Функції редагування

Редагування запису в таблиці Answer:

```
static async editDataAnswer() {
    try {
        const id: number = +question('answer id: ');
        const newAns: Answer = Reader.prepareDataAnswer();
        const text: string = 'UPDATE "Answer" SET "aHeader" = $1, "aText" = $2, "aCreationDate" = $3, "a_author_id" = $4, "a_question_id" = $5 WHERE "answerId" = $6';
        const check = await pool.query('SELECT * FROM "Answer" WHERE "answerId" = $1', [id]);

        if (!check.rows.length) {
            console.log(`There is no answer with id ${id}`);
        } else {
            const checkUser = await pool.query('SELECT * FROM "Users" WHERE "userId" = $1', [newAns.a_author_id]);
            const checkQuestion = await pool.query('SELECT * FROM "Questions" WHERE "questionId" = $1', [newAns.a_question_id]);

            if (!checkUser.rows.length || !checkQuestion.rows.length) {
                console.log(`There is no user with id ${newAns.a_author_id} or question with id ${newAns.a_question_id}`);
            } else {
                await pool.query(text, [newAns.aHeader, newAns.aText, newAns.aCreationDate, newAns.a_author_id, newAns.a_question_id, id]);

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

```
input: 2
answer id: 7
answer header: edited header
answer text: edited text
author id: 4
question id: 2
Row with id 7 has been updated
```

Результат:

	answerId [PK] integer	aHeader character varying (50)	aText character varying (500)	aCreationDate date	a_author_id integer	a_question_id integer
1	1	Try *	Example: 7*2	2019-11-02	5	1
2	2	LOL	There is no biggest number	2020-09-20	8	2
3	3	Read this article	link: https://link.com	2020-09-20	7	2
4	4	Need for speed	Underground	2020-07-24	4	3
5	5	FarCry	All chapters	2020-10-20	3	3
6	7	edited header	edited text	2020-12-17	4	2

Редагування запису в таблиці Google_Profile:

```
static async editDataGoogleProfile() {
    try {
        const id: number = +question('profile id: ');
        const updateProfile: GoogleProfile = Reader.prepareDataGoogleProfile();
        const text: string = 'UPDATE "Google_Profile" SET "email" = $1, "nickname" = $2, "adIdentifier" = $3, "questionUser" = $4 WHERE "gpId" = $5';
        const check = await pool.query('SELECT * FROM "Google_Profile" WHERE "gpId" = $1', [id]);

        if (!check.rows.length) {
            console.log(`There is no profile with id ${id} in database`);
        } else {
            const userCheck = await pool.query('SELECT * FROM "Users" WHERE "userId" = $1', [updateProfile.questionUser]);

            if (!userCheck.rows.length) {
                console.log(`There is no user with id ${updateProfile.questionUser} in database`);
            } else {
                await pool.query(text, [updateProfile.email, updateProfile.nickname, updateProfile.adIdentifier, updateProfile.questionUser, id]);

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

```
input: 2
profile id: 9
email: newtest@mail.com
nickname: newtestnickname
ad id: 123123
user id: 5
Row with id 9 has been updated
```

Результат:

	gpId [PK] bigint	email character varying (50)	nickname character varying (50)	adIdentifier integer	questionUser integer
1	2	somemail@email.com	google_droid	123536	2
2	3	vitakdu@gmail.com	another_talk	64314	3
3	4	anothermail@mail.ru	bulbaster228	41245	4
4	5	ukrainian@i.ua	akinator	23512	7
5	6	ram@rambler.ru	muslimKA	351245	8
6	9	newtest@mail.com	newtestnickname	123123	5

Редагування запису в таблиці QuestionTagsBinding:

```
static async editDataQTB() {
    try {
        const id: number = +question('QTB id: ');
        const newQtb: QuestionTagsBinding = Reader.prepareDataQuestionTagsBinding();
        const text: string = 'UPDATE "QuestionTagsBinding" SET "qt_tag_id" = $1, "qt_question_id" = $2 WHERE "qtId" = $3';
        const checkQTB = await pool.query('SELECT * FROM "QuestionTagsBinding" WHERE "qtId" = $1', [id]);
        const checkTag = await pool.query('SELECT * FROM "Tags" WHERE "tagId" = $1', [newQtb.qt_tag_id]);
        const checkQuestion = await pool.query('SELECT * FROM "Questions" WHERE "questionId" = $1', [newQtb.qt_question_id]);

        if (!checkQTB.rows.length) {
            console.log(`There is no QTB with id ${id}`);
        } else {
            if (!checkTag.rows.length || !checkQuestion.rows.length) {
                console.log(`There is no tag with id ${newQtb.qt_tag_id} or question with id ${newQtb.qt_question_id}`);
            } else {
                await pool.query(text, [newQtb.qt_tag_id, newQtb.qt_question_id, id]);

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

```
input: 2
QTB id: 6
question id: 1
tag id: 2
Row with id 6 has been updated
```

Результат:

	qtId [PK] integer	qt_tag_id integer	qt_question_id integer
1	1	1	1
2	2	1	2
3	3	2	2
4	4	3	3
5	6	2	1

Редагування запису в таблиці Questions:

```
static async editDataQuestion() {
    try {
        const id: number = +question('question id: ');
        const text: string = 'UPDATE "Questions" SET "qHeader" = $1, "qText" = $2, "creationDate" = $3, "authorId" = $4 WHERE "questionId" = $5';
        const newQuest: Question = Reader.prepareDataQuestion();
        const checkUser = await pool.query('SELECT * FROM "Users" WHERE "userId" = $1', [newQuest.authorId]);
        const checkQuestion = await pool.query('SELECT * FROM "Questions" WHERE "questionId" = $1', [id]);

        if (!checkQuestion.rows.length) {
            console.log(`There is no question with id ${id}`);
        } else {
            if (!checkUser.rows.length) {
                console.log(`There is no user with id ${newQuest.authorId}`);
            } else {
                await pool.query(text, [newQuest.qHeader, newQuest.qText, newQuest.creationDate, newQuest.authorId, id]);

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

```
input: 2
question id: 5
question header: new test header
question text: new text edited
author id: 5
Row with id 5 has been updated
```

Результат:

	questionId [PK] integer	qHeader character varying (50)	qText character varying (500)	creationDate date	authorId integer
1	1	How to multiply in c#?	Hello! Want to multiply two di...	2019-10-02	2
2	2	What is the biggest number?	idk help	2020-09-20	3
3	3	Your favourite PC game?	Lets talk about something else	2020-07-07	5
4	5	new test header	new text edited	2020-12-17	5

Редагування запису в таблиці Tags:

```
static async editDataTag() {
    try {
        const id: number = +question('tag id: ');
        const updateTag: Tags = Reader.prepareDataTag();
        const text: string = 'UPDATE "Tags" SET "tName" = $1, "description" = $2 WHERE "tagId" = $3';
        const check = await pool.query('SELECT * FROM "Tags" WHERE "tagId" = $1', [id]);

        if (!check.rows.length) {
            console.log(`There is no tag with id ${id} in database`);
        } else {
            await pool.query(text, [updateTag.tName, updateTag.description, id]);

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

```
input: 2
tag id: 6
tName: new test tag
description: edited description
Record with id 6 has been updated
```

Результат:

	tagId [PK] integer	tName character varying (50)	description character varying (500)
1	1	Programming	Here you can ask about progr...
2	2	Mathematics	Algebra, geometry, etc.
3	3	Free themes	Everything you want
4	4	Biology	Natural things
5	6	new test tag	edited description

Редагування запису в таблиці Users:

```
static async editDataUser() {
    try {
        const id: number = +question('user id: ');
        const updateUser: Users = Reader.prepareDataUser();
        const text: string = 'UPDATE "Users" SET username = $1, reg_date = $2, rating = $3 WHERE "userId" = $4';
        const check = await pool.query('SELECT * FROM "Users" WHERE "userId" = $1', [id]);

        if (!check.rows.length) {
            console.log(`There is no user with id ${id} in database`);
        } else {
            await pool.query(text, [updateUser.username, updateUser.reg_date, updateUser.rating, id]);

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

```
input: 2
user id: 16
username: edited username
rating: 212
Record with id 16 has been updated
```

Результат:

	userId [PK] integer	username character varying (50)	reg_date date	rating integer	confirmed boolean
3	4	Ivan	2020-04-02	2	true
4	5	somethinganother	2019-02-14	135	false
5	6	bulek	2004-04-02	3	true
6	7	terminator228	2018-02-03	19	false
7	8	teremok2	2002-02-02	86	false
8	10	rtyuiopasd	2018-01-10	895	true
9	11	uiopasdfgh	2018-01-11	472	false
10	12	zxcvbnmQWE	2018-01-18	460	false
11	13	QWERTYUIOP	2018-01-18	163	false
12	14	fghjklzxcv	2018-01-18	242	true
13	15	rtyuiopasd	2018-01-12	95	false
14	16	edited username	2020-12-17	212	false

Функції видалення

Видалення в таблиці Answer:

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

        if (!check.rows.length) {
            console.log(`There is no answer 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
answer id: 7
Row with id 7 has been deleted
```

Результат:

	answerId [PK] integer	aHeader character varying (50)	aText character varying (500)	aCreationDate date	a_author_id integer	a_question_id integer
1	1	Try *	Example: 7*2	2019-11-02	5	1
2	2	LOL	There is no biggest number	2020-09-20	8	2
3	3	Read this article	link: https://link.com	2020-09-20	7	2
4	4	Need for speed	Underground	2020-07-24	4	3
5	5	FarCry	All chapters	2020-10-20	3	3

Видалення в таблиці Google_Profile:

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

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

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

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

Результат:

	gpId [PK] bigint	email character varying (50)	nickname character varying (50)	adIdentifier integer	questionUser integer
1	2	somemail@email.com	google_droid	123536	2
2	3	vitakdu@gmail.com	another_talk	64314	3
3	4	anothermail@mail.ru	bulbaster228	41245	4
4	5	ukrainian@i.ua	akinator	23512	7
5	6	ram@rambler.ru	muslimKA	351245	8

Видалення в таблиці QuestionTagsBinding:

```
static async deleteDataQTB() {
    try {
        const id: number = +question('QTB id: ');
        const text: string = 'DELETE FROM "QuestionTagsBinding" WHERE "qtId" = $1';
        const checkQTB = await pool.query('SELECT * FROM "QuestionTagsBinding" WHERE "qtId" = $1', [id]);

        if (!checkQTB.rows.length) {
            console.log(`There is no QTB 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
QTB id: 6
Row with id 6 has been deleted
```

Результат:

	qtId [PK] integer	qt_tag_id integer	qt_question_id integer
1	1	1	1
2	2	1	2
3	3	2	2
4	4	3	3

Видалення в таблиці Questions:

```
static async deleteDataQuestion() {
    try {
        const id: number = +question('question id: ');
        const text: string = 'DELETE FROM "Questions" WHERE "questionId" = $1';
        const checkQuestion = await pool.query('SELECT * FROM "Questions" WHERE "questionId" = $1', [id]);

        if (!checkQuestion.rows.length) {
            console.log(`There is no question 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
question id: 5
Row with id 5 has been deleted
```

Результат:

	questionId [PK] integer	qHeader character varying (50)	qText character varying (500)	creationDate date	authorId integer
1	1	How to multiply in c#?	Hello! Want to multiply two di...	2019-10-02	2
2	2	What is the biggest number?	idk help	2020-09-20	3
3	3	Your favourite PC game?	Lets talk about something else	2020-07-07	5

Видалення в таблиці Tags:

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

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

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

```
input: 3
tag id: 6
Record with id 6 has been deleted
```

Результат:

	tagId [PK] integer	tName character varying (50)	description character varying (500)
1	1	Programming	Here you can ask about progr...
2	2	Mathematics	Algebra, geometry, etc.
3	3	Free themes	Everything you want
4	4	Biology	Natural things

Видалення в таблиці Users:

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

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

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

```
input: 3
user id: 16
Record with id 16 has been deleted
```

Результат:

	userId [PK] integer	username character varying (50)	reg_date date	rating integer	confirmed boolean
1	2	drioder	2001-10-06	12	true
2	3	rodriges	2014-09-08	51	false
3	4	ivan	2020-04-02	2	true
4	5	somethinganother	2019-02-14	135	false
5	6	bulek	2004-04-02	3	true
6	7	terminator228	2018-02-03	19	false
7	8	teremok2	2002-02-02	86	false
8	10	rtyuiopasd	2018-01-10	895	true
9	11	uiopasdfgh	2018-01-11	472	false
10	12	zxcvbnmQWE	2018-01-18	460	false
11	13	QWERTYUIOP	2018-01-18	163	false
12	14	fghjklzxcv	2018-01-18	242	true
13	15	rtyuiopasd	2018-01-12	95	false

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

Рандомізоване внесення даних до таблиці Users

Було створено клас Generate із методом generateUsers. Задана кількість записів генерується у таблиці **Users**:

```
const pool = require('../db.js');
import { question } from 'readline-sync';

import { Reader } from '../utils/reader.js';
import { Printer } from '../utils/printer.js';
import { Users } from '../types.js';

export class Generate {
    static async generateUsers() {
        try {
            const num: number = +question('number of records: ');
            const text: string = `
                insert into "Users" (username, reg_date, rating, confirmed)
                    select substr(characters, (random() * length(characters) + 1)::integer, 10),
                    timestamp '2018-01-10' + random() * (timestamp '2018-01-20' - timestamp '2018-01-10'),
                    trunc(random() * 1000)::int,
                    cast(cast(round(random()) as character varying) as boolean)
                    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 Users`);
            console.log(`query time: ${queryTime}`);
        } catch (err) {
            console.log(err);
        }
    }
}
```

Таблиця Users до генерації:

	userId [PK] integer	username character varying (50)	reg_date date	rating integer	confirmed boolean
1	2	drioder	2001-10-06	12	true
2	3	rodriges	2014-09-08	51	false
3	4	ivan	2020-04-02	2	true
4	5	somethinganother	2019-02-14	135	false
5	6	bulek	2004-04-02	3	true
6	7	terminator228	2018-02-03	19	false
7	8	teremok2	2002-02-02	86	false

Генеруємо записи:

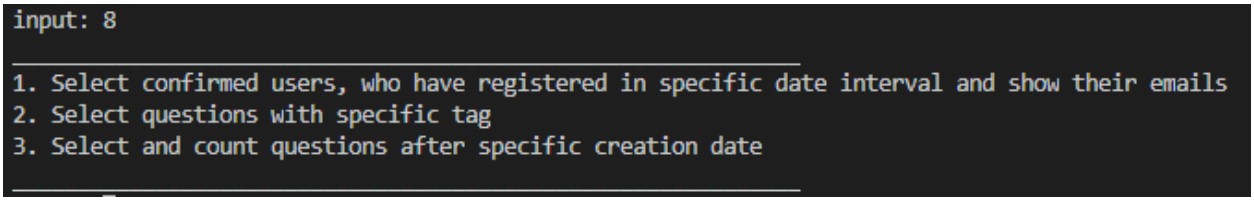
```
input: 7
number of records: 15
15 rows has been generated in table Users
query time: 55
```

Таблиця Users після генерації:

	userId [PK] integer	username character varying (50)	reg_date date	rating integer	confirmed boolean
1	2	drioder	2001-10-06	12	true
2	3	rodriges	2014-09-08	51	false
3	4	ivan	2020-04-02	2	true
4	5	somethinganother	2019-02-14	135	false
5	6	bulek	2004-04-02	3	true
6	7	terminator228	2018-02-03	19	false
7	8	teremok2	2002-02-02	86	false
8	17	zxcvbnmQWE	2018-01-15	372	true
9	18	opasdfghjk	2018-01-18	206	false
10	19	UIOPASDFGH	2018-01-18	964	false
11	20	sdfghjklzx	2018-01-10	606	true
12	21	tyuiopasdf	2018-01-19	344	true
13	22	mQWERTYUIO	2018-01-10	148	false
14	23	GHJKLZXCVB	2018-01-14	532	true
15	24	rtyuiopasd	2018-01-16	537	true
16	25	WERTYUIOPA	2018-01-13	578	true
17	26	VBNM	2018-01-16	334	false
18	27	uiopasdfgh	2018-01-10	661	false
19	28	ZXCVBNM	2018-01-18	659	true
20	29	tyuiopasdf	2018-01-12	443	false
21	30	ZXCVBNM	2018-01-11	490	true
22	31	wertyuiopa	2018-01-12	76	false

Виконання динамічних запитів бази даних

Меню вибору запиту:



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 specDate() {
    const date1: string = question('first date: ');
    const date2: string = question('second date: ');
    const text: string = `
        with confirmedUsers as (
            select * from "Users" where "confirmed" = true
            and "userId" in (select "questionUser" from "Google_Profile")
        ),

        specDate as (
            select * from confirmedUsers where "reg_date" > $1 and "reg_date" < $2
        )

        select "username", "reg_date", "confirmed", m."email" from specDate join "Google_Profile" as m on m."questionUser" = specDate."userId";
    `;

    try {
        const start: number = Date.now();
        const result = await pool.query(text, [date1, date2]);
        const queryTime: number = Date.now() - start;

        if (!result.rows.length) {
            console.log('No result');
        } else {
            console.log('username   |  reg_date  | confirmed | email');
            console.log('_____________________________________________________');

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

                if (item.username.length > 11) {
                    modUName = item.username.substr(0, 8) + '...';
                } else {
                    modUName = Format.toField(11, item.username);
                }

                console.log(`${modUName}|${Format.toField(12, Format.formatDate(item.reg_date))}|${Format.toField(11, item.confirmed.toString())}|${item.email}`);
                console.log('_____________________________________________________');
            });
            console.log(`query time: ${queryTime}ms`);
        }
    } catch (err) {
        console.log(err);
    }
}
```

Результат:

```
input: 1
first date: 20010101
second date: 20190101
username   |  reg_date  | confirmed | email
_____________________________________________________
drioder    |06.10.01    |true       |somemail@email.com
_____________________________________________________
query time: 1ms
```

Другий запит:

```
static async specTag() {
    const tag: string = question('tag: ');
    const text: string = `
        with tagsIds as (
            select "tagId", "tName" from "Tags" where "tName" = $1
        ),

        questionIds as (
            select "tagId", "tName", qtb."qt_question_id" from tagsIds join "QuestionTagsBinding" as qtb on "tagId" in (select "qt_tag_id" from "QuestionTagsBinding")
        ),

        tagAndQ as (
            select distinct "tName", q."qHeader"
            from questionIds
            join "Questions" as q
            on "qt_question_id" in (select "questionId" from "Questions")
        )

        select * from tagAndQ;
    `;

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

        if (!result.rows.length) {
            console.log('No result');
        } else {
            console.log('     tName      |       qHeader');
            console.log('_____________________________________');

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

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

                if (item.qHeader.length > 20) {
                    modHeader = item.qHeader.substr(0, 17) + '...';
                } else {
                    modHeader = Format.toField(20, item.qHeader);
                }

                console.log(`${modTName}|${modHeader}`);
                console.log('_____________________________________');
            });
            console.log(`query time: ${queryTime}ms`);
        }
    } catch (err) {
        console.log(err);
    }
}
```

Результат:

```
input: 2
tag: Programming
     tName      |       qHeader

_____________________________________
Programming     |What is the bigge...

_____________________________________
Programming     |How to multiply i...

_____________________________________
Programming     |Your favourite PC...

_____________________________________
query time: 2ms
```

Третій запит:

```
static async specQDate() {
    try {
        const date: string = question('date: ');
        const text: string = `
            select count(*), "creationDate" from "Questions" where "creationDate" > $1
            group by "creationDate"
        `;
        const start: number = Date.now();
        const result = await pool.query(text, [date]);
        const queryTime: number = Date.now() - start;

        if (!result.rows.length) {
            console.log('No result');
        } else {
            console.log(' count |   creationDate');
            console.log('_____________________');

            result.rows.forEach((item: any) => {

                console.log(`${Format.toField(7, item.count.toString())}|${Format.formatDate(item.creationDate)}`);
                console.log('_____________________');
            });
            console.log(`query time: ${queryTime}ms`);
        }
    } catch (err) {
        console.log(err);
    }
}
```

Результат:

```
input: 3
date: 20100101
 count |   creationDate
_____________________
1      |07.07.20
_____________________
1      |02.10.19
_____________________
1      |20.09.20
_____________________
query time: 1ms
```

Обробка виняткових ситуацій (помилок) 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
answer header: new text
answer text: new text
author id: 10001
question id: 10001
There is no user with id 10001 or question with id 10001 in database
```

Введення рядка з полем, тип якого не відповідає дійсному:

```
input: 1
question id: asdas
tag id: asdas
error: неверный синтаксис для типа integer: "NaN"
```

Видалення рядка, ключ якого є зовнішнім ключем іншої таблиці:

```
input: 3
user id: 7
error: UPDATE или DELETE в таблице "Users" нарушает ограничение внешнего ключа "fk_user" таблицы "Google_Profile"
```

Дослідження режимів обмеження ON DELETE

Дослідження режимів будемо проводити на таблиці Users (батьківська) та Google_Profile (дочірня). Використаємо запис із таблиці Google_Profile з gpId = 5 та зовнішнім ключем questionUser = 7:

Таблиця Google_Profile:

	gpId [PK] bigint	email character varying (50)	nickname character varying (50)	adIdentifier integer	questionUser integer
1	2	somemail@email.com	google_droid	123536	2
2	3	vitakdu@gmail.com	another_talk	64314	3
3	4	anothermail@mail.ru	bulbaster228	41245	4
4	5	ukrainian@i.ua	akinator	23512	7

Таблиця Users:

	userId [PK] integer	username character varying (50)	reg_date date	rating integer	confirmed boolean
1	2	drioder	2001-10-06	12	true
2	3	rodriges	2014-09-08	51	false
3	4	ivan	2020-04-02	2	true
4	5	somethinganother	2019-02-14	135	false
5	6	bulek	2004-04-02	3	true
6	7	terminator228	2018-02-03	19	false
7	8	teremok2	2002-02-02	86	false

Режим NO ACTION

При видаленні запису із таблиці Users, id якого присутній в таблиці Google_Profile отримуємо повідомлення про помилку:

```
input: 3
user id: 7
error: UPDATE или DELETE в таблице "Users" нарушает ограничение внешнего ключа "fk_user" таблицы "Google_Profile"
```

Режим SET NULL

При видаленні запису із таблиці Users, id якого присутній в таблиці Google_Profile, якщо на questionUser таблиці Google_Profile NOT NULL, отримуємо повідомлення:

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

Якщо прибрати NOT NULL, то questionUser прийме значення NULL.

Режим SET DEFAULT

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

В налаштуваннях Users 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 Users, id якого присутній в таблиці Google_Profile отримуємо таке саме повідомлення про помилку, як і в режимі NO ACTION:

```
input: 3
user id: 7
error: UPDATE или DELETE в таблице "Users" нарушает ограничение внешнего ключа "fk_user" таблицы "Google_Profile"
```

Режим CASCADE

При видаленні запису із таблиці Users, id якого присутній в таблиці Google_Profile, видаляється запис таблиці Google_Profile та запис таблиці Users із відповідним userId:

```
input: 3
user id: 7
Record with id 7 has been deleted
```

Таблиця Users:

	userId [PK] integer	username character varying (50)	reg_date date	rating integer	confirmed boolean
1	2	drioder	2001-10-06	12	true
2	3	rodriges	2014-09-08	51	false
3	4	ivan	2020-04-02	2	true
4	5	somethinganother	2019-02-14	135	false
5	6	bulek	2004-04-02	3	true
6	8	teremok2	2002-02-02	86	false

Таблиця Google_Profile:

	gpId [PK] bigint	email character varying (50)	nickname character varying (50)	adIdentifier integer	questionUser integer
1	2	somemail@email.com	google_droid	123536	2
2	3	vitakdu@gmail.com	another_talk	64314	3
3	4	anothermail@mail.ru	bulbaster228	41245	4