

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Кафедра Систем Штучного Інтелекту

РОЗРАХУНКОВА РОБОТА

з дисципліни «Організація баз даних та знань»

на тему:

«Веб-сервіс пошуку коду»

Виконав:

студент групи КН-210

Бікєєв Андрій Олексійович

Балів	Дата

Викладач:

Мельникова Наталя Іванівна

Зміст

1. Тема проекту.....	3
2. Вступ.....	4
3. Логічна схема БД проекту.....	5
4. Опис структури БД.....	6
5. Фізична модель БД.....	9
6. Ділова модель.....	15
7. Запити до БД.....	16
8. Висновки.....	19
9. Список використаних джерел інформації.....	20

1. Тема проекту.

Темою проекту нашої команди A-Team є веб – сервіс, який дозволяє користувачу знаходити робочі фрагменти коду за заданими критеріями, такими як мова, на якій цей код написаний, або ж/і джерело, на якому цей фрагмент коду розміщений.

Основна ідея проекту полягає в тому, що просіювання безлічі неробочих/недопрацьованих рішень вручну займає багато часу, тоді як перевірку працездатності коду – ні, а отже її можна з легкістю автоматизувати.

Принцип роботи нашого проекту полягає у тому, що:

1. Користувач авторизується або вручну, або за допомогою соціальних медіа.
2. Авторизований користувач заходить на наш веб – сервіс та вводить ключовий пошуковий запит
3. Далі користувач обирає мову програмування та/або першоджерело для пошуку рішення.
4. Користувач отримує декілька варіантів готових рішень, які може фільтрувати на «якість», за допомогою тестових випадків.
5. Задоволений користувач забирає найбільш підходяще для нього рішення, ділиться своїм враженням від роботи сервісу, можливо коментує враження інших та іде вставляти фрагмент коду у свій проект.

2. Вступ

При виборі теми для розрахункової роботи ми в першу чергу керувались власним досвідом та хотіли вирішити проблему, з якою стикались самі.

Нашою метою стало створити веб-сервіс, здатний знаходити код за пошуковим запитом, що задовольняє задані тестові випадки.

Ось **цілі**, які ми поставили перед собою:

- ✓ Створити відповідальний та юзабельний інтерфейс для введення тестових справ, які визначають якість рішення.
- ✓ Створити аналізатор для збору фрагментів коду з різних джерел, таких як - StackOverflow, Github тощо.
- ✓ Створити мовний процесор, щоб визначити, на яких залежностях працює фрагмент коду, і перетворить фрагмент у робоче рішення.
- ✓ Створити сервер Sandbox для безпечного тестування рішення.

Нашим критерієм успіху було те, що усі компоненти працюють синхронізовано, створюють рутину введення проблеми -> введення тестових випадків -> отримання робочого рішення.

Проте були і **сумніви**:

- ✓ Те, що яке-небудь джерело фрагментів коду може і допоможе вирішити будь-яку задачу.
- ✓ Те, що користувач може вводити тестові справи, які однозначно і безпомилково ідентифікують робоче рішення від непрацюючого.

Серед **ризиків та перешкод** ми виділили:

- ✓ Безпечний запуск будь-якого фрагмента коду.
- ✓ Створення інтуїтивного середовища для користувача.
- ✓ Правильно визначити залежності фрагмента коду.

Обрана нами тема є актуальною, адже кожен хто хоч раз шукав необхідний фрагмент коду стикався з тим щоб власноруч його перевіряти та продовжувати пошуки. Дуже рідко необхідна нам інформація з'являється першою, тому ми вирішили полегшити це завдання як собі, так і нашим колегам. Також користувач може залишити свій відгук на певний фрагмент коду і ми обов'язково врахуємо дану інформацію що полегшить подальші пошуки іншим користувачам.

3. Логічна схема БД проекту.

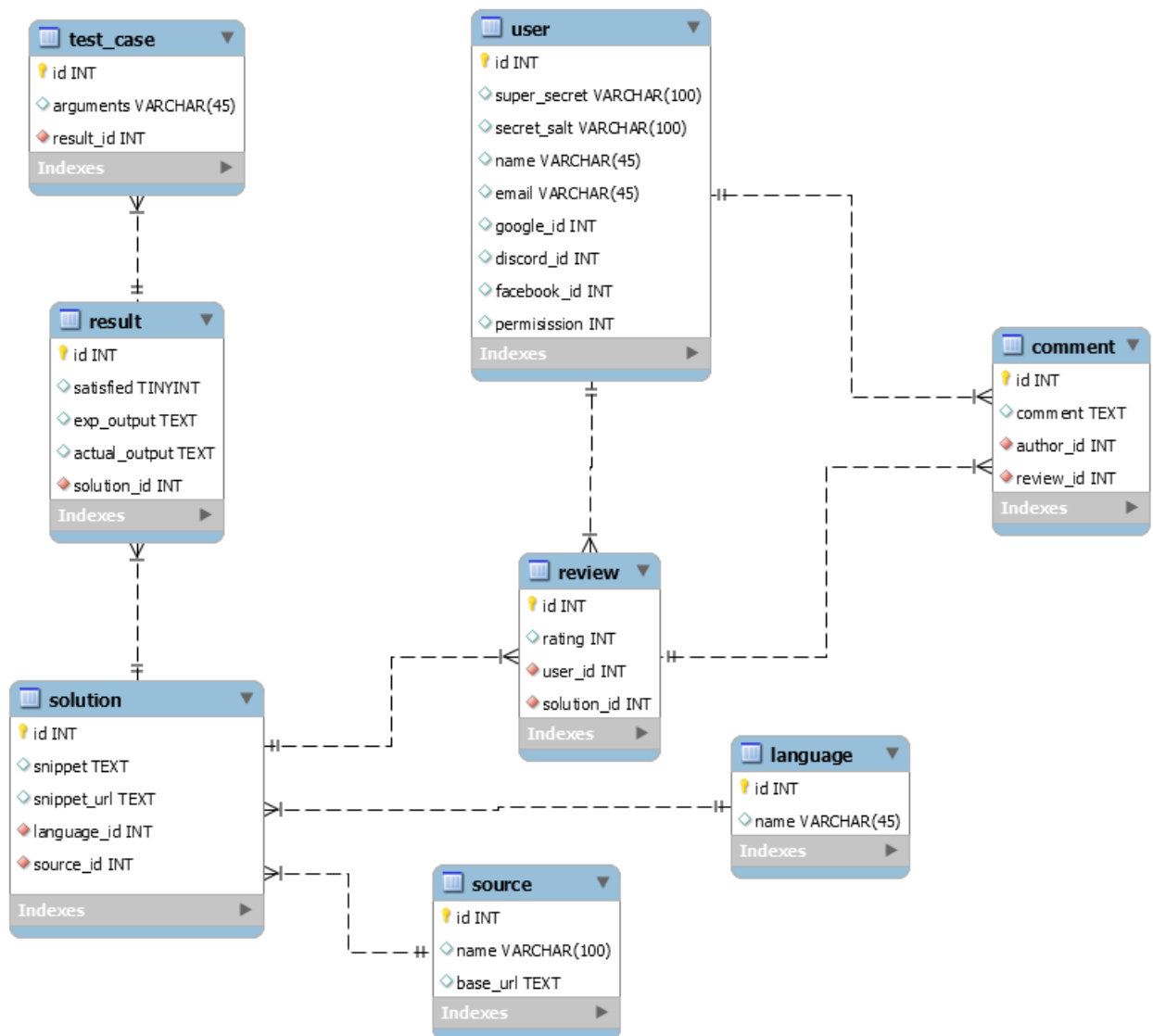


Рис 1. Логічна схема Бази Даних проекту.

4.Опис структури БД.

Логічна схема Баз Даних подана на попередній сторінці. Почнемо з однієї із головних таблиць у структурі БД – таблиці *user*.

Структура цієї таблиці:

- *id* INT - Внутрішній ключ, цілочисельний унікальний ідентифікатор.
- *super_secret* varchar(100) - Персональний ідентифікуючий токен юзера, згенерований сервісом у разі, якщо користувач не використовував соціальну авторизацію. Обмеження - 100 символів.
- *name* varchar(45) – Ім'я користувача. Обов'язкове поле у разі, якщо користувач не використовував соціальну авторизацію, обмеження- 45 символів.
- *email* varchar(45) – Емейл користувача. Обов'язкове поле у разі, якщо користувач не використовував соціальну авторизацію, обмеження - 45 символів.
- *google_id* INT - Персональний ідентифікуючий токен юзера, отриманий сервісом через спеціальний запит у разі, якщо користувач використовував соціальну авторизацію через Google.
- *discord_id* INT - Персональний ідентифікуючий токен юзера, отриманий сервісом через спеціальний запит у разі, якщо користувач використовував соціальну авторизацію через Discord.
- *facebook_id* INT - Персональний ідентифікуючий токен юзера, отриманий сервісом через спеціальний запит у разі, якщо користувач використовував соціальну авторизацію через facebook.
- *permission* INT - бітмаска дозволів. Наприклад, "видалити коментар", або "видалити користувача".

Проте, який сервіс з користувачами може існувати, без надавання цим користувачам певних можливостей виділятися та взаємодіяти. Тому окрім самої таблиці *User*, створено таблиці:

reviews – для збереження ревію користувача на рішення, яке він отримав.

Тут є поля:

- *id* INT - Внутрішній ключ, цілочисельний унікальний ідентифікатор.
- *rating* INT – Оцінка в певному діапазоні, яку виставляє користувач певному рішенню.

- user_id INT – Зовнішній ключ, який посилається на користувача, який залишив дане рев'ю.
- solution_id INT - Зовнішній ключ, який посилається на рішення, яке отримав користувач і на яке він залишає своє рев'ю.

comments – для збереження коментарів інших користувачів на рев'ю, яке залишив певний користувач. Тут відношення багато до багатьох, оскільки багато користувачів може коментувати одне рев'ю, так і один користувач може коментувати багато рев'ю.

Тут є поля:

- id INT - Внутрішній ключ, цілочисельний унікальний ідентифікатор.
- comment TEXT – коментар, який певний користувач залишає щодо рев'ю іншого користувача та його рішення.
- author_id INT - Зовнішній ключ, який посилається на користувача, який залишає даний коментар.
- review_id INT - Зовнішній ключ, який посилається на таблицю рев'ю, яке коментує даний користувач

На наш ресурс користувач приходить за певним рішенням. Для цього в нас створена таблиця Solution, куди зберігаються певні параметри пошуку, та виводиться результат з парсеру. Для цього тут створені поля:

- id INT - Внутрішній ключ, цілочисельний унікальний ідентифікатор.
- snippet TEXT – поле для певного отриманого фрагменту коду
- snippet_url TEXT – поле для посилання на першоджерело отриманого коду.
- languages_id INT - Зовнішній ключ, який посилається на таблицю мови програмування, яка була обрана для пошуку рішення.
- sources_id INT - Зовнішній ключ, який посилається на таблицю джерела, яке користувач обрав для пошуку.

Для повного опису сутності solution в нас створено ще поле result.

Тут є поля:

- id INT - Внутрішній ключ, цілочисельний унікальний ідентифікатор.
- satisfied TINYINT – бінарне значення яке повертає '0' або '1' в залежності від валідності результату.
- exp_output TEXT - поле очікуваного результату.
- actual_output TEXT – поле реального результату, який отримує користувач.
- solutions_id INT - Зовнішній ключ, який посилається на таблицю рішення, яке користувач має отримати.

Для повного опису сутності result в нас створено поле test_case, де вводяться аргументи, які вписуються в функцію результату для обрахунку.

- id INT - Внутрішній ключ, цілочисельний унікальний ідентифікатор.
- arguments VARCHAR(45) – поле для введення аргументів.
- results_id INT - Зовнішній ключ, який посилається на таблицю результату, куди вписуватимуться аргументи.

Для налаштувань таблиці рішення в нас створено дві таблиці:

Для вибору мови програмування – language з полями:

- id INT - Внутрішній ключ, цілочисельний унікальний ідентифікатор.
- name VARCHAR (45) – ім'я мови програмування, якою шукатиметься рішення.

Для вибору джерела для пошуку – source з полями:

- id INT - Внутрішній ключ, цілочисельний унікальний ідентифікатор.
- name VARCHAR (100) – найменування джерела, на якому шукатиметься рішення.
- base_url TEXT – посилання на обране джерело, на якому шукатиметься рішення.

5. Фізична модель БД.

Текст файлу створення БД з оголошенням обмежень, індексів та ключів:

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

```
-- -----
-- Schema mydb
-- -----
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8
;
USE `mydb` ;
```

```
-- -----
-- Table `mydb`.`language`
-- -----
CREATE TABLE IF NOT EXISTS `mydb`.`language` (
  `id` INT NOT NULL,
  `name` VARCHAR(45) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

```
-- -----
-- Table `mydb`.`source`
-- -----
```

```

CREATE TABLE IF NOT EXISTS `mydb`.`source` (
  `id` INT NOT NULL,
  `name` VARCHAR(100) NULL,
  `base_url` TEXT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

```

```

-- -----
-- Table `mydb`.`solution`
-- -----
CREATE TABLE IF NOT EXISTS `mydb`.`solution` (
  `id` INT NOT NULL,
  `snippet` TEXT NULL,
  `snippet_url` TEXT NULL,
  `language_id` INT NOT NULL,
  `source_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_solution_language1_idx` (`language_id` ASC) VISIBLE,
  INDEX `fk_solution_source1_idx` (`source_id` ASC) VISIBLE,
  CONSTRAINT `fk_solution_language1`
  FOREIGN KEY (`language_id`)
  REFERENCES `mydb`.`language` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
  CONSTRAINT `fk_solution_source1`
  FOREIGN KEY (`source_id`)
  REFERENCES `mydb`.`source` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table `mydb`.`result`
-- -----
CREATE TABLE IF NOT EXISTS `mydb`.`result` (
  `id` INT NOT NULL,
  `satisfied` TINYINT NULL,

```

```

`exp_output` TEXT NULL,
`actual_output` TEXT NULL,
`solution_id` INT NOT NULL,
PRIMARY KEY (`id`),
INDEX `fk_results_solutions1_idx` (`solution_id` ASC) VISIBLE,
CONSTRAINT `fk_results_solutions1`
FOREIGN KEY (`solution_id`)
REFERENCES `mydb`.`solution` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table `mydb`.`test_case`
-- -----
CREATE TABLE IF NOT EXISTS `mydb`.`test_case` (
`id` INT NOT NULL,
`arguments` VARCHAR(45) NULL,
`result_id` INT NOT NULL,
PRIMARY KEY (`id`),
INDEX `fk_test_case_results1_idx` (`result_id` ASC) VISIBLE,
CONSTRAINT `fk_test_case_results1`
FOREIGN KEY (`result_id`)
REFERENCES `mydb`.`result` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table `mydb`.`user`
-- -----
CREATE TABLE IF NOT EXISTS `mydb`.`user` (
`id` INT NOT NULL,
`super_secret` VARCHAR(100) NULL,
`secret_salt` VARCHAR(100) NULL,
`name` VARCHAR(45) NULL,
`email` VARCHAR(45) NULL,

```

```

`google_id` INT NULL,
`discord_id` INT NULL,
`facebook_id` INT NULL,
`permission` INT NULL,
PRIMARY KEY (`id`))
ENGINE = InnoDB;

```

```

-- -----
-- Table `mydb`.`review`
-- -----
CREATE TABLE IF NOT EXISTS `mydb`.`review` (
`id` INT NOT NULL,
`rating` INT NULL,
`user_id` INT NOT NULL,
`solution_id` INT NOT NULL,
PRIMARY KEY (`id`),
INDEX `fk_reviews_users_idx` (`user_id` ASC) VISIBLE,
INDEX `fk_reviews_solutions1_idx` (`solution_id` ASC) VISIBLE,
CONSTRAINT `fk_reviews_users`
FOREIGN KEY (`user_id`)
REFERENCES `mydb`.`user` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_reviews_solutions1`
FOREIGN KEY (`solution_id`)
REFERENCES `mydb`.`solution` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table `mydb`.`comment`
-- -----
CREATE TABLE IF NOT EXISTS `mydb`.`comment` (
`id` INT NOT NULL,
`comment` TEXT NULL,
`author_id` INT NOT NULL,

```

```

`review_id` INT NOT NULL,
PRIMARY KEY (`id`),
INDEX `fk_comments_users1_idx` (`author_id` ASC) VISIBLE,
INDEX `fk_comments_reviews1_idx` (`review_id` ASC) VISIBLE,
CONSTRAINT `fk_comments_users1`
FOREIGN KEY (`author_id`)
REFERENCES `mydb`.`user` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_comments_reviews1`
FOREIGN KEY (`review_id`)
REFERENCES `mydb`.`review` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

6. Ділова модель.

Тут відображена ділова модель бази даних. Вона встановлює відповідність між певною функцією, яку виконує наш сервіс, та сутністю в базі даних, яка бере участь в запиті.

Таблиця\ Функція	user	comment	review	solution	language	source	result	test_case
Аутентифікація	*							
Авторизація	*							
Обрати відповідь за мовою					*		*	
Обрати відповідь за ресурсом						*	*	

Пошук				*	*	*	*	*
Зберігти відгук			*	*				
Зберігти коментар до відгуку		*	*	*				
Пергляну ти відгуки		*	*	*				
Зберегти оцінку до розв'язку			*	*				

7. Запити до БД.

1. Вивід всіх «мов», які опрацьовуються цим сервісом:

*select * from language;*

```
mysql> select * from language;
+----+-----+
| id | name   |
+----+-----+
| 1  | Python |
| 2  | Java   |
| 3  | JavaScript |
| 4  | C       |
| 5  | C++    |
| 6  | MySql  |
+----+-----+
6 rows in set (0.00 sec)
```

2. Вивід всіх посилань, на «рішення», які були знайдені за допомогою веб-сервісу “StackOverflow”

*select S.snippet_url, SO.name from solution as S
inner join source SO
on S.source_id = SO.id
where SO.name = “stackoverflow”;*

```
mysql> select S.snippet_url, SO.name from solution as S
-> inner join source SO
-> on S.source_id = SO.id
-> WHERE SO.name = "stackoverflow"
-> ;
+-----+-----+
| snippet_url | name |
+-----+-----+
| https://stackoverflow.com/questions/2137755/how-do-i-reverse-an-int-array-in-java | stackoverflow |
| https://stackoverflow.com/questions/5862761/java-random-array-of-numbers | stackoverflow |
| https://ru.stackoverflow.com/questions/1138012/%d0%91%d0%be%d1%82-%d0%bd%d0%be%d0%bf%d0%be%d1%81%d0%b0%d0%bd%d1%8b%d0%b9-%d0%bd%d0%b0-discord-api-%d0%bf%d0%b8%d1%88%d0%b5%d1%82-%d0%b2-%d1%87%d0%be%d1%82-%d1%82%d0%be%d0%bb%d1%8c%d0%ba%d0%be-%d0%be%d0%b4%d0%bd%d0%be-%d1%81%d0%be%d0%be%d0%b1%d1%89%d0%b5%d0%bd%d0%b8%d0%b5 | stackoverflow |
+-----+-----+
3 rows in set (0.00 sec)
```

3. Підрахуємо кількість відгуків від кожного з користувачів:

```
select u.id, u.name, count(*) as amount_of_reviews  
from user as u  
inner join review as r on r.user_id = u.id  
group by u.id;
```

```
mysql> select u.id, u.name, count(*) as amount_of_reviews  
-> from user as u  
-> inner join review as r on r.user_id = u.id  
-> group by u.id;
```

id	name	amount_of_reviews
1	Doikov Vadym	2
2	Bikieiev Andrii	1
3	Kruchkovska Khrystuna	1
4	Zbryskyi Kostiantun	1
5	Pomirko Oleh	1
6	Chornii Yuri	2

```
6 rows in set (0.00 sec)
```

4. Підрахуємо кількість розв'язків, які користувач "Bikieiev Andrii" оцінив в більше ніж 5 «зірок».

```
select u.name, count(*) as amount from user as u  
inner join review as r  
on r.user_id = u.id  
where r.rating > 5 and u.name = "Bikieiev Andrii";
```

```
mysql> select u.name, COUNT(*) as amount from user as u  
-> inner join review as r  
-> on r.user_id = u.id  
-> where r.rating > 5 and u.name = "Bikieiev Andrii";
```

name	amount
Bikieiev Andrii	1

```
1 row in set (0.00 sec)
```

5. Створимо функцію, яка буде аутентифіковувати користувача за «паролем», та його айді.

```
delimiter //  
create function check_password(  
    user_id int,  
    password varchar(100))  
    returns tinyint  
    deterministic  
    begin  
        declare secret varchar(100);
```

```

declare salt varchar(100);

set secret = (select super_secret from user where id = user_id);
set salt = (select secret_salt from user where id = user_id);
if SHA2(concat(password, salt), 512) = SHA2(concat(secret, salt), 512)
then
    return 1;
else
    return 0;
end if;
end;

```

Перевіримо роботу цієї функції:

```

select u.id, u.super_secret, check_password(u.id, "Super_secret") as correct
from user as u;

```

```

mysql> select u.id, u.super_secret, check_password(u.id, "Super_secret") as correct from user as u;
+----+-----+-----+
| id | super_secret | correct |
+----+-----+-----+
| 1  | q1w151s9bn9w010s0zk | 0       |
| 2  | 34w151s9b1;a;s0s0zk | 0       |
| 3  | 30lsdlgs;as;`x'a    | 0       |
| 4  | 5g;;;sas;mmsd1as1a  | 0       |
| 5  | 50s0ads021dl1dsf1   | 0       |
| 6  | Super_secret        | 1       |
+----+-----+-----+
6 rows in set (0.00 sec)

```

- Підрахуємо середню оцінку рішень для кожного з користувачів:

```

select u.name, avg(r.rating) as avg_rating from user as u
inner join review as r
on u.id = r.user_id
group by u.name;

```

```

mysql> select u.name, avg(r.rating) as avg_rating from user as u
-> inner join review as r
-> on u.id = r.user_id
-> group by u.name;
+-----+-----+
| name | avg_rating |
+-----+-----+
| Doikov Vadym | 6.5000 |
| Bikiiev Andrii | 6.0000 |
| Kruchkovska Khrystuna | 7.0000 |
| Zbryskyi Kostiantun | 10.0000 |
| Pomirko Oleh | 10.0000 |
| Chornii Yurii | 6.5000 |
+-----+-----+
6 rows in set (0.00 sec)

```


7. Створимо тригер, який встановлює поле `satisfied` автоматично, при додаванні полів `exp_output`, та `actual_output` до таблиці `result`.

```
create trigger check_output
before insert on result
for each row
set new.satisfied = case
    when new.exp_output = new.actual_output then 1
    when new.exp_output <> new.actual_output then 0
end;
```

```
mysql> create trigger check_output
-> before insert on result
-> for each row
-> set new.satisfied = case
->     when new.exp_output = new.actual_output then 1
->     when new.exp_output <> new.actual_output then 0
->     end;
Query OK, 0 rows affected (0.01 sec)
```

Перевіримо цей тригер, додавши результат з однаковими очікуванням та дійсним виводами, та й навпаки:

```
insert into result(id, exp_output, actual_output, solution_id)
values (11, 214, 214, 2);
```

```
mysql> insert into result(id, exp_output, actual_output, solution_id)
-> values (11, 214, 214, 2);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> select exp_output, actual_output, satisfied from result where id = 11;
+-----+-----+-----+
| exp_output | actual_output | satisfied |
+-----+-----+-----+
| 214       | 214          | 1        |
+-----+-----+-----+
1 row in set (0.00 sec)
```

```
insert into result(id, exp_output, actual_output, solution_id)
values (12, "[1, 2, 3]", "[3, 2, 1]", 4);
```

```
mysql> insert into result(id, exp_output, actual_output, solution_id)
-> values (12, "[1, 2, 3]", "[3, 2, 1]", 4);
Query OK, 1 row affected (0.01 sec)

mysql> select exp_output, actual_output, satisfied from result where id = 12;
+-----+-----+-----+
| exp_output | actual_output | satisfied |
+-----+-----+-----+
| [1, 2, 3] | [3, 2, 1]    | 0        |
+-----+-----+-----+
1 row in set (0.00 sec)
```

8. Створимо процедуру, яка б повертала усі тестові випадки, які мають відношення до того чи іншого користувача(Наприклад, були написані, або використані ним)

```

delimiter //
create procedure get_test_cases(IN users_id int)
begin
    select tc.id as test_case_id from user as u
    inner join review as r
    on u.id = r.user_id
    inner join solution as s
    on s.id = r.solution_id
    inner join result as re
    on re.solution_id = s.id
    inner join test_case as tc
    where u.id = users_id and tc.result_id = re.id;
end; //

```

```

mysql> create procedure get_test_cases(IN users_id int)
-> begin
-> select tc.id as test_case_id from user as u
-> inner join review as r
-> on u.id = r.user_id
-> inner join solution as s
-> on s.id = r.solution_id
-> inner join result as re
-> on re.solution_id = s.id
-> inner join test_case as tc
-> where u.id = users_id and tc.result_id = re.id;
-> end; //
Query OK, 0 rows affected (0.01 sec)

```

Наприклад, для користувача 1:

```

mysql> call get_test_cases(1);
+-----+
| test_case_id |
+-----+
|           1 |
|           2 |
+-----+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

```

8. Висновки.

Виконуючи цю розрахункову роботу, а також весь проект в цілому, я навчився працювати з реляційними базами даних. Спершу навчився проектувати схеми баз даних за допомогою ER-діаграм (Entity-Relation Diagram). На основі ER-діаграми я навчився будувати моделі бази даних. Після створення бази даних навчився записувати дані в БД, і діставати їх простими запитами. Крім цього, була проведена робота над оптимізацією, що включала в себе створення індексів та застосування певних обмежень.

Сам проект побудований на основі Flask, Python, бібліотека flask-mysql яких надає високорівневий доступ до більшості функцій бази даних. Тому вся робота над проектом була зроблена без використання синтаксису SQL. Для виконання розрахункової роботи була використана аналогічна модель бази даних, але в середовищі MySQL, для відображення роботи створеної командою бази даних.

9. Список використаних джерел інформації.

1. Пасічник В.В., Резніченко В.А. Організація баз даних та знань - К.: Видавнича група BHV, 2006. — 384 с.: іл. — ISBN 966-552-156-X.
2. Coronel C., Morris S. Database Systems: Design, Implementation, and Management. 12th ed. – Cengage Learning, 2017. – 818 p.
3. Connolly T.M., Begg C.E. Database Systems: A Practical Approach to Design, Implementation and Management: Global Edition. – 6th Edition. – Pearson Education, 2015. – 1440 p.
4. Kroenke D.M., Auer D.J. Database Processing: Fundamentals, Design, and Implementation. 14th ed. – Pearson Education Ltd., 2016. – 638 p.
5. <https://www.w3schools.com/sql/>
6. <https://www.tutorialspoint.com/sql/index.htm>
7. <http://www.sql-tutorial.ru/>
8. <https://www.codecademy.com/learn/learn-sql>
9. <https://www.mysqltutorial.org/>
10. <https://www.tutorialspoint.com/mysql/index.htm>