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-сервіз пошуку коду»

Виконала:

студентка групи КН-210

Кручковська Христина Юріївна

|  |  |
| --- | --- |
| **Балів** | **Дата** |
|  |  |

Викладач:

Мельникова Наталя Іванівна

**Львів – 2020**

**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 можна 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, куди може передати свої аргументи.
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.

Для створення баз даних було обрано MySQL3.

1. **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:**

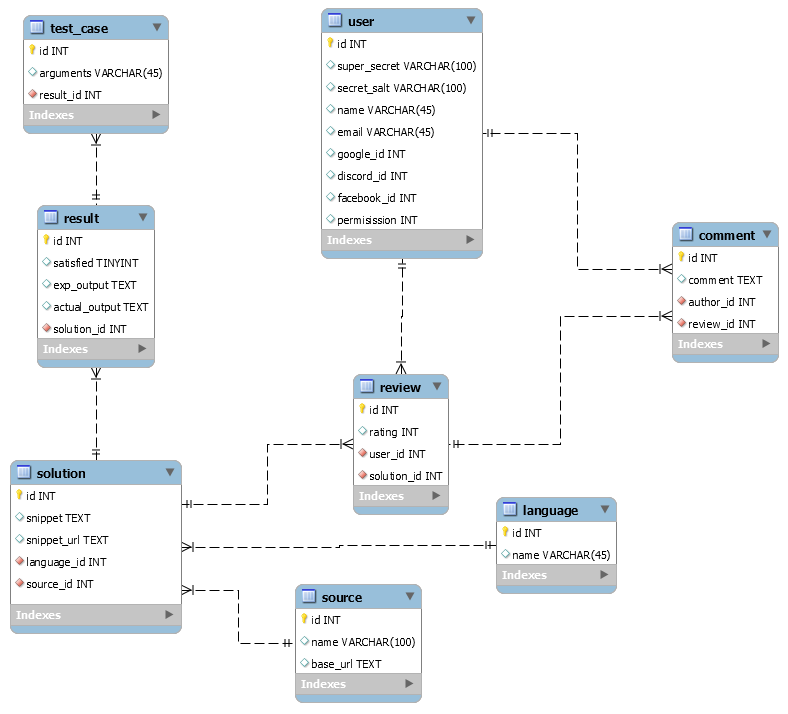
* Те, що яке-небудь джерело фрагментів коду може і допоможе вирішити будь-яку задачу.
* Те, що користувач може вводити тестові справи, які однозначно і безпомилково ідентифікують робоче рішення від непрацюючого.

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.

1. **Логічна схема БД проекту.**

Рис 1. Логічна схема Бази Даних проекту.

**4.Опис структури БД.**

Логічна схема Бази 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’зкове поле у разі, якщо користувач не використовував соціальну авторизацію, обмеження- 45 символів.
* email varchar(45) – Емейл користувача. Обов’зкове 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 рев'ю 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 кометар.
* review\_id INT - Зовнішній ключ, який посилається на таблицю рев’ю, яке кометує 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.

1. **Фізична модель БД.**

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

-- MySQL Script generated by MySQL Workbench

-- Sat May 23 16:48:31 2020

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

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

-- -----------------------------------------------------

-- -----------------------------------------------------

-- 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,

`permisission` 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;

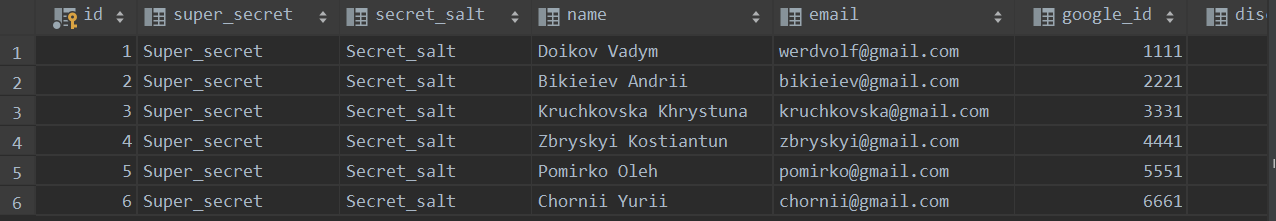
1. **Ділова модель.**

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 | \* |  |  |  |  |  |  |  |
| Додати test case |  |  |  |  |  |  | \* | \* |
| Додати аргумент |  |  |  |  |  |  | \* | \* |
| Обрати мову |  |  |  |  | \* |  | \* |  |
| Обрати ресурс |  |  |  |  |  | \* | \* |  |
| Пошук |  |  |  | \* | \* | \* | \* | \* |
| Написати відгук |  |  | \* | \* |  |  |  |  |
| Написати комент до відгуку |  | \* | \* | \* |  |  |  |  |
| Переглянути відгуки |  | \* | \* | \* |  |  |  |  |
| Поставити оцінку |  |  | \* | \* |  |  |  |  |

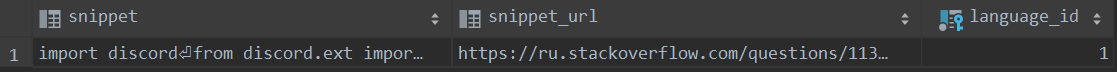
1. **Запити до БД.**
2. Для початку переглядаємо всіх користувачів.

use mydb;  
select \* from user;



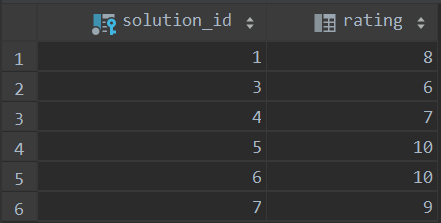
1. Переглянемо всі рішення з російськомовних сайтів:

use mydb;  
select  
 snippet,  
 snippet\_url,  
 language\_id  
from  
 solution  
where  
 snippet\_url LIKE 'https://ru';



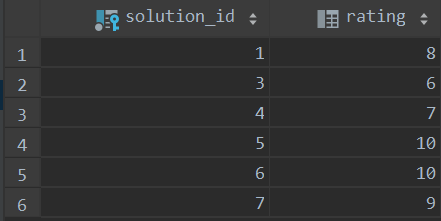
1. Переглянемо ті рішення, оцінка на які більша за 5:

use mydb;  
select  
 solution\_id,  
 rating  
from review  
where rating > 5



1. Виводимо всі рішення, які задовольнили наші очікування:

use mydb;  
  
select exp\_output,  
 actual\_output  
from result  
where satisfied = 1



1. Функція, яка з’єднує всі id прив’язаних акаунтів користувача:

use mydb;  
  
delimiter $$  
create function *account\_id*(  
 google char(45),  
 diskord char(45),  
 facebook char(45)  
)  
returns varchar(135)  
deterministic  
begin  
 return *concat*('google:', google, ' diskord: ', diskord, ' facebook: ', facebook);  
end$$  
delimiter ;

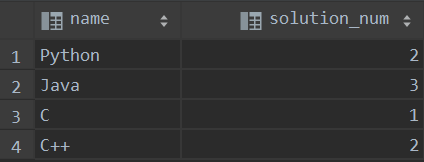
Виклик функції та результат виконання:

use mydb;  
  
select *account\_id*(google\_id, discord\_id, facebook\_id)  
from user



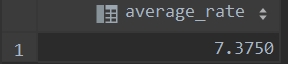
1. Дивимось на кількість рішень кожною з мов програмування.

use mydb;  
  
select name,  
 *count*(\*) as solution\_num  
from language  
inner join solution s on language.id = s.language\_id  
group by language.id



1. Знаходимо середню оцінку для всіх рішень, які знаходить наш сервіс:

use mydb;  
  
select *avg*(rating) as average\_rate  
from review



1. **Висновки**.

Виконуючи цю розрахункову роботу, а також весь проект в цілому, я навчилася працювати з реляційними базами даних. Також навчилася проектувати схеми баз даних за допомогою ER-діаграм (Entity-Relation Diagram). В цей згенерований код слід вносити правки, створювати потрібні індекси, перевіряти правильність зовнішніх ключів. Після створення бази даних навчилася 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.

**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-Х.

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