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

Дойков Вадим Степанович

Балів	**Дата**

Викладач:

Мельникова Наталя Іванівна

Львів – 2020

Зміст

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.

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.

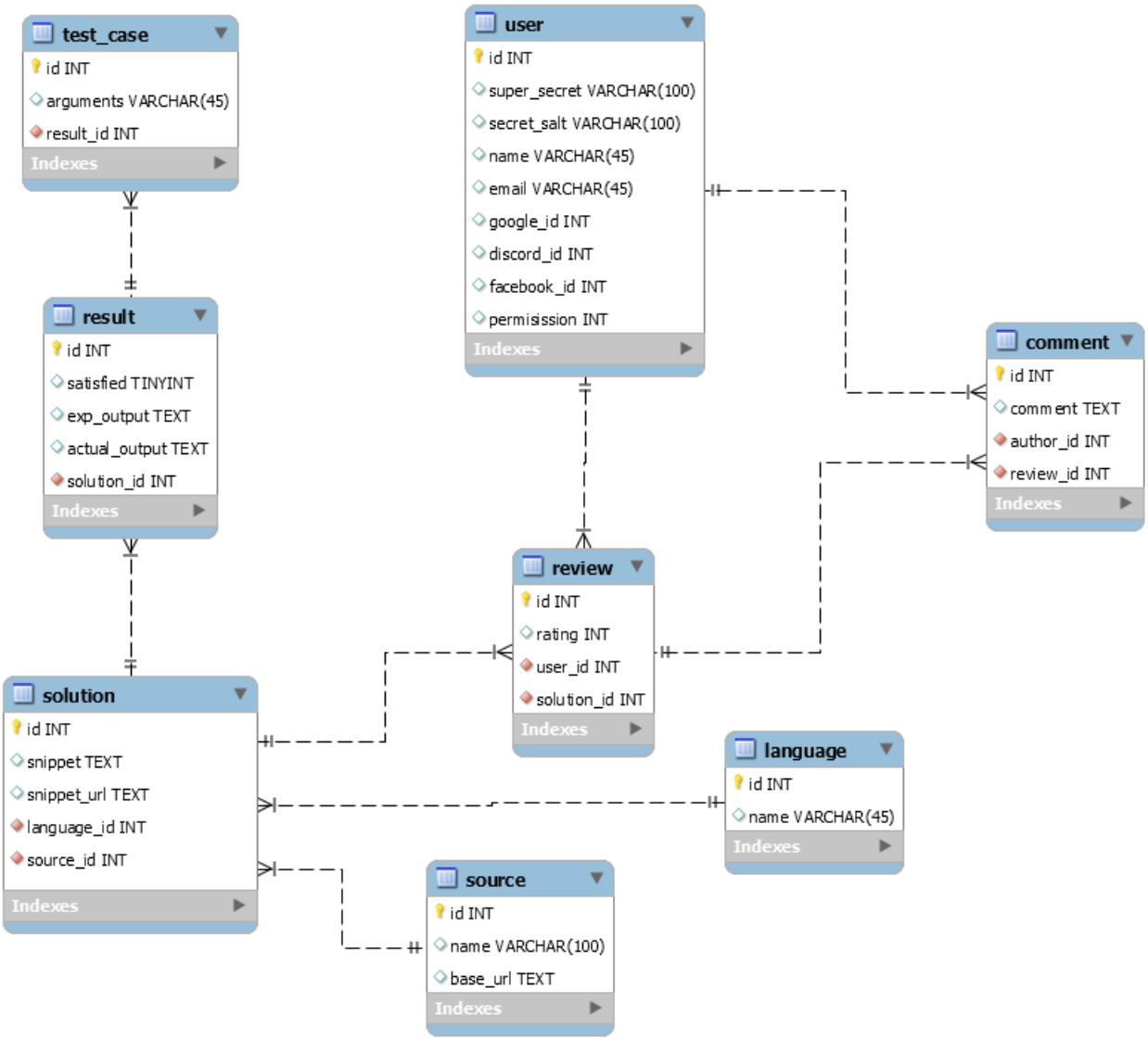

Рис 1. Логічна схема Бази Даних проекту.

4.Опис структури БД.

Логічна схема 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.

5. 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_DA
TE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUT
ION';

-- -----------------------------------------------------
-- 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;
```

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

7. Запити до БД.

1. Вивід всіх користувачів:
   select * from mydb.user;

id	super_secret	secret_salt	name	email	google_id	discord_id	facebook_id	permisission
1	Super_secret	Secret_salt	Doikov Vadym	werdvolf@gmail.com	1111	1112	1113	0
2	Super_secret	Secret_salt	Bikieiev Andrii	bikieiev@gmail.com	2221	2222	2223	0
3	Super_secret	Secret_salt	Kruchkovska Khrystuna	kruchkovska@gmail.com	3331	3332	3333	1
4	Super_secret	Secret_salt	Zbryskyi Kostiantun	zbryskyi@gmail.com	4441	4442	4443	2
5	Super_secret	Secret_salt	Pomirko Oleh	pomirko@gmail.com	5551	5552	5553	4
6	Super_secret	Secret_salt	Chornii Yurii	chornii@gmail.com	6661	6662	6663	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Вивід всіх посилань на відповіді, де мова програмування «Python»:
   select s.id, s.snippet_url, l.name
   from solution as s
   inner join mydb.language as l on l.id = s.language_id
   where l.name = "Python";

id	snippet_url	name
7	https://www.hackerrank.com/environment/writi...	Python
8	https://ru.stackoverflow.com/questions/11300...	Python

3. Підрахунок скільки кожен користувач зробив відгуків:
   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 Yurii	2

4. Підрахуємо кількість розв’язків, які користувач «Doikov Vadym» оцінив в хоча б 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 = " Doikov Vadym";
```

	name	amount
▶	Doikov Vadym	1

5. Вивід середньої оцінки відповідей для мови «Java»:
```
select l.name, avg(r.rating) as average
from review as r
inner join solution as s on r.solution_id = s.id
inner join language as l on l.id = s.language_id
where l.name = "Java";
```

	name	average
▶	Java	7.6667

6. Функція, яка перевіряє чи очікувана відповідь співпадає з дійсною:
```
delimiter //
create function check_answers(
        exp_output text,
   actual_output text)
   returns tinyint
   deterministic
   begin
            if exp_output = actual_output then
                return 1;
            else
                return 0;
            end if;
end;//
```

Виклик функції та результат виконання:

```
select r.exp_output, r.actual_output,
check_answers(r.exp_output, r.actual_output) as satisfied
from result as r;
```

exp_output	actual_output	satisfied
[4,3,2,1]	[4,3,2,1]	1
null	[2,4,4,4,1]	0
Рівно 7	[4,3,2,1]	0
myfile.txt	myfile.txt	1
0	1	0
file.txt	file.txt	1
myfile.txt	myfile.txt	1
NULL	NULL	1

7. Тригер, який перевіряє чи додана очікувана відповідь співпадає з дійсною, після додавання до таблиці та змінює поле satisfied:

```
drop trigger check_output;
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;
```

При додаванні нового поля з однаковими exp_output та actual_output:

```
insert into result(id, exp_output, actual_output, solution_id)
values (9, 123, 123, 2);
```

id	satisfied	exp_output	actual_output	solution_id
1	1	[4,3,2,1]	[4,3,2,1]	1
2	0	null	[2,4,4,4,1]	2
3	0	Рівно 7	[4,3,2,1]	3
4	1	myfile.txt	myfile.txt	4
5	0	0	1	5
6	1	file.txt	file.txt	6
7	1	myfile.txt	myfile.txt	7
8	1	NULL	NULL	8
9	1	123	123	2
NULL	NULL	NULL	NULL	NULL

При додаванні нового поля з різними exp_output та actual_output:

```
insert into result(id, exp_output, actual_output, solution_id)
values (10, 12, 123, 4);
```

id	satisfied	exp_output	actual_output	solution_id
1	1	[4,3,2,1]	[4,3,2,1]	1
2	0	null	[2,4,4,4,1]	2
3	0	Рівно 7	[4,3,2,1]	3
4	1	myfile.txt	myfile.txt	4
5	0	0	1	5
6	1	file.txt	file.txt	6
7	1	myfile.txt	myfile.txt	7
8	1	NULL	NULL	8
9	1	123	123	2
10	0	12	123	4

8. **Висновки**.

Виконуючи цю розрахункову роботу, а також весь проект в цілому, я навчився працювати з реляційними базами даних. авчився 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 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.

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