НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

(повна назва інституту/факультету)

КАФЕДРА інформатики та програмної інженерії

(повна назва кафедри)

**КУРСОВА РОБОТА**

з дисципліни «Бази даних»

(назва дисципліни)

на тему База даних маршрутів громадського транспорту міста

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студента (ки) \_\_2\_\_ курсу \_ІП-11\_\_\_\_\_ групи

спеціальності 121 «Інженерія програмного забезпечення»

\_Головні Олександра Ростиславовича\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_Ас. Ліщук О. В. \_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка ECTS \_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2022 рік

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки

(повна назва)

Кафедра Інформатики та програмної інженерії

(повна назва)

Дисципліна Бази даних

Курс \_\_2\_\_\_ Група \_\_ІП-11 Семестр \_\_\_3\_\_

**З А В Д А Н Н Я**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

\_\_\_\_\_\_\_Головні Олександру Ростиславовичу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема роботи\_\_\_\_\_База даних маршрутів громадського транспорту міста\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

керівник роботи Ас. Ліщук О. В.

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи \_\_\_\_\_10.01.2023\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи Створена база даних відповідно до умови, sql скрипти\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)\_\_Опис предметного середовища, постановка задачі, побудова ER-діаграми, побудова Реляційної моделі бд, реалізація бд, створення користувачів, sql запити

1) Аналіз предметного середовища

2) Побудова ER-моделі

3) Побудова реляційної схеми з ER-моделі

4) Створення бази даних, у форматі обраної системи управління базою даних

5) Створення користувачів бази даних

6) Імпорт даних з використанням засобів СУБД в створену базу даних

7) Створення мовою SQL запитів

8) Оптимізація роботи запитів

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Дата видачі завдання\_\_\_\_08.11.2022**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
| 1 | Аналіз предметного середовища |  |  |
| 2 | Побудова ER-моделі |  |  |
| 3 | Побудова реляційної схеми з ER-моделі |  |  |
| 4 | Створення бази даних, у форматі обраної системи управління базою даних |  |  |
| 5 | Створення користувачів бази даних |  |  |
| 6 | Імпорт даних з використанням засобів СУБД в створену базу даних |  |  |
| 7 | Створення мовою SQL запитів |  |  |
| 8 | Оптимізація роботи запитів |  |  |
| 9 | Оформлення пояснювальної записки |  |  |
| 10 | Захист курсової роботи | 10.01.2022 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_**Головня О.Р.**\_\_\_\_\_\_\_**

(підпис ) (прізвище та ініціали)

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Ліщук О.В.**\_\_\_\_\_\_\_\_**

(підпис ) (прізвище та ініціали)

**АНОТАЦІЯ**

Пояснювальна записка до курсової роботи: 34 сторінки, 17 рисунків, 2 таблиці, 3 посилання.

Об’єкт дослідження: база даних для збереження даних про футбольні чемпіонати.

Мета роботи: закріплення навичок з проектування, реалізації реляційних баз даних та їх використання для практичних задач.

Проведено аналіз предметного середовища, визначено сутності, атриблути та зв’язки між об’єктами. Побудована ER-модель предметного середовища, реляційну схему бази даних, виділено первинні та зовнішні ключі, визначено обмеження для підтримки цілісності даних. Розроблено скрипти для побудови спроектованої бази даних, імпортовано дані в неї, виконано різноманітні запити до неї.

Виконана програмна реалізація бази даних маршрутів громадського транспорту ( варіант 7 ).

**ЗМІСТ**

[ВСТУП 9](#_Toc123257886)

[Опис предметного середовища 10](#_Toc123257887)

[Постановка задачі 11](#_Toc123257888)

[ER-діаграма 12](#_Toc123257889)

[Бізнес-правила 12](#_Toc123257890)

[Вибір сутностей 12](#_Toc123257891)

[Набори атрибутів сутностей 13](#_Toc123257892)

[Реляційна модель бази даних 16](#_Toc123257893)

[Побудова необхідних відношень та визначення первинних і зовнішніх ключів 16](#_Toc123257894)

[Визначення обмежень цілісності для спроектованих відношень 16](#_Toc123257895)

[Реалізація бази даних 17](#_Toc123257896)

[Створення бази даних у форматі системи управління базою даних mysql 17](#_Toc123257897)

[Імпортування даних в таблицю 20](#_Toc123257898)

[Створення користувачів бази даних 23](#_Toc123257899)

[Створення пасажира 23](#_Toc123257900)

[Створення перевіряльника 23](#_Toc123257901)

[SQL запити 24](#_Toc123257902)

[Створення тригерів на таблиці, що будуть змінюватись користувачами. 24](#_Toc123257903)

[Створення процедур для взаємодії користувачів з базою даних 27](#_Toc123257904)

[Процедури пасажира 27](#_Toc123257905)

[Процедури перевіряльника пасажирів 28](#_Toc123257906)

[**Інші sql запити** 29](#_Toc123257907)

[Приклад роботи індексів 33](#_Toc123257908)

[Висновок 35](#_Toc123257909)

[Перелік посилань 36](#_Toc123257910)

ВСТУП

**База даних** – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім самих даних, містить їх опис та може містити засоби для їх обробки.

Обрана мною тема є актуальною, тому що: Як мінімум, можна вести підрахунок кількості оплачених поїздок у громадському транспорті. А, якщо вести підрахунок лише кількості оплачених поїздок, то можливо дізнатись лише загальну кількість перевезених пасажирів за певний період часу. Інформацію про те, скільки людей входить та виходить на тих чи інших зупинках, коли саме, на яких ділянках маршруту та напрямах руху відбувається найбільше навантаження, отримати за рахунок обліку надходжень неможливо ніяким чином. Система збору статистики та статистичного аналізу пасажиропотоку є досить корисною інформацією для використання в будь яких сферах.

Метою цієї роботи є створити базу даних маршрутів громадського транспорту міста, яка може замінити чиюсь роботу, на самообслуговування, тобто клієнт через онлайн сервіс вдома або ж будь-де з доступом до інтернету робить бронювання місця на будь-який маршурт громадського транспорту з вільними місцями.

Опис предметного середовища

Організація маршрутів є одним з найголовніших факторів безпечної та комфортної їзди. Під кожен маршурт варто виділити найбільш підходящий транспорт, обирати який можна за різними характеристиками: кількість пасажирів, марка і чимало інших. Для цього інформацію про кожен транспорт зручно структурувати у вигляді таблиці. Для ефективного використання часу зайнятості і завантаженості найкращим рішенням буде створити графік відбуття та прибуття транспорту на маршруті.

Постановка задачі

Метою даної роботи є розробка бази даних маршрутів громадського транспорту міста. Тобто організація даних таким чином, щоб робота громадського транспорту виконувалась максимально ефективно. Для конкретики та наглядності, я вибрав доповнення і створю маршрутизацію тролейбусів. Отже основні задачі:

Пасажир повинен мати змогу:

* Забронювати місце на транспорт
* Переглянути всі свої бронювання

Також, нам потрібен контролер для перевірки, тому перевіряючий повинен мати змогу:

* Переглянути список пасажирів
* Змінити статус перевірки певному пасажиру

ER-діаграма

Після аналізу було виділено такі сутності та зв’язки між ними:

Бізнес-правила

1. Дата бронювання транспорту не може бути більшою ніж дата виїзду транспорта.
2. На транспорт не можна забронювати більше білетів, ніж максимальна кількість місць.
3. Результат перевірки є або позитивним або негативним.

Вибір сутностей

* Пасажир
* Перевірка пасажира
* Бронювання
* Перевіряючий
* Результати(довідник)
* Графік маршрутів
* Маршрут
* Місця(початок і кінець маршруту)
* Тролейбус
* Марка

Набори атрибутів сутностей

– Сутності та їхні атрибути

|  |  |
| --- | --- |
| Сутність | Атрибути |
| passanger | passanger\_id  first\_name  last\_name  passport\_number |
| passanger\_check | passanger\_check\_id  result\_id  checked\_at  passanger\_id  checker\_id |
| booking | booking\_id  passangers\_id  booking\_time  booking\_date  route\_id |

|  |  |
| --- | --- |
| route | route\_id  graphic\_id  start\_place  end\_place |
| route\_graphic | graphic\_id  time\_daparture  time\_arrival  route\_date |
| trolley | transport\_id  id\_brand  route\_id  max\_passangers  last\_date\_of\_check |
| places | place\_id  place\_name |
| сhecker | checker\_id  first\_name  last\_name |
| brand | id\_brand  brand\_name |
| results | result\_id  result\_name |

Сутність passanger буде пов’язана один до багатьох з сутностями booking passanger\_check, адже один пасажир може в різні дати та час бронювати та проходити перевірку.

Сутність booking пов’язана багато до одного з сутністю route, тому що на один маршрут є багато бронювань.

Сутність route\_graphic пов’язана один до багатьх з сутністю route, тому що всі маршрути відбуваються за одним графіком.

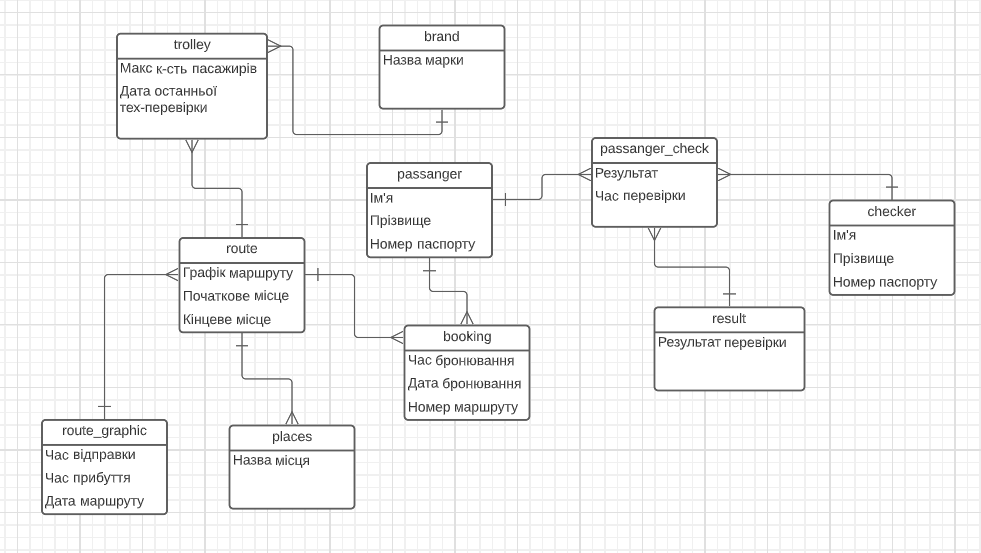
Сутність brand пов’язана один до багатьох з сутністю trolley, тому що є багато тролейбусів одного бренда.

Сутність trolley пов’язана багато до одного з сутністю route, тому що на одному маршруті може працювати декілька транспорту.

Сутність places пов’язана один до багатьох з сутністю route, тому що декілька маршрутів можуть починатися або завершуватися в одному місті.

Сутність checker пов’язана один до багатьох з сутностями passanger\_check, бо одна людина може перевірити чимало пасажирів.

Сутність results пов’язана один до багатьох з сутностями passanger\_check, бо один результат може бути характерний для багатьох перевірок.



– ER-діаграма

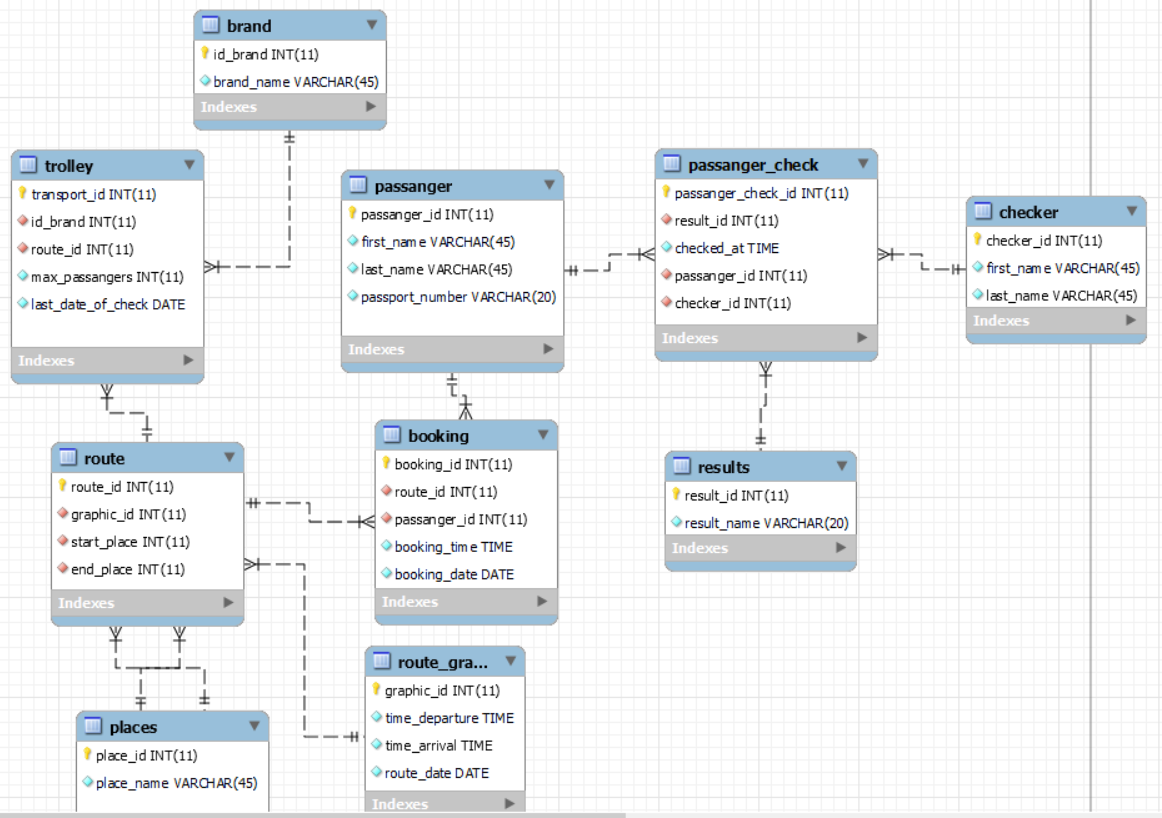
Реляційна модель бази даних

Побудова необхідних відношень та визначення первинних і зовнішніх ключів

На рисунку можна побачити, що база даних знаходить у 3 нормальній формі, оскільки: поля таблиці декомпозовані, атрибути функціально повно залежать від первинного ключа, кожен неключовий атрибут не є транзитивно залежним від первинного ключа.

Визначення обмежень цілісності для спроектованих відношень

1. Якщо не існує зовнішних ключів, що посилаються на значення головного ключа батьківського рядка, то тільки у цьому випадку його можна видалити. Для цього треба додати до таблиці обмеження з відповідними параметрами: ALTER TABLE child ADD CONSTRAINT fk\_parent\_child FOREIGN KEY (parent\_id) REFERENCES parent (id) ON UPDATE CASCADE ON DELETE RESTRICT
2. Обов’язкові артрибути таблиць мають обмеження NOT NULL для запобігання помилок при роботі з даними



– Реляційна схема бази даних

Реалізація бази даних

Створення бази даних у форматі системи управління базою даних mysql

CREATE TABLE IF NOT EXISTS results (

result\_id INT(11) NOT NULL AUTO\_INCREMENT,

result\_name VARCHAR(20) NOT NULL,

PRIMARY KEY (result\_id)

);

CREATE TABLE IF NOT EXISTS checker (

checker\_id INT(11) NOT NULL AUTO\_INCREMENT,

first\_name VARCHAR(45) NOT NULL,

last\_name VARCHAR(45) NOT NULL,

PRIMARY KEY (checker\_id)

);

CREATE TABLE IF NOT EXISTS passanger (

passanger\_id INT(11) NOT NULL AUTO\_INCREMENT,

first\_name VARCHAR(45) NOT NULL,

last\_name VARCHAR(45) NOT NULL,

passport\_number VARCHAR(20) NOT NULL,

PRIMARY KEY (passanger\_id)

);

CREATE TABLE IF NOT EXISTS passanger\_check (

passanger\_check\_id INT(11) NOT NULL AUTO\_INCREMENT,

result\_id INT(11) NOT NULL,

checked\_at TIME NOT NULL,

passanger\_id INT(11) NOT NULL,

checker\_id INT(11) NOT NULL,

PRIMARY KEY (passanger\_check\_id),

FOREIGN KEY (passanger\_id) REFERENCES passanger(passanger\_id),

FOREIGN KEY (checker\_id) REFERENCES checker(checker\_id),

FOREIGN KEY (result\_id) REFERENCES results(result\_id)

);

CREATE TABLE IF NOT EXISTS brand (

id\_brand INT(11) NOT NULL AUTO\_INCREMENT,

brand\_name VARCHAR(45) NOT NULL,

PRIMARY KEY(id\_brand)

);

CREATE TABLE IF NOT EXISTS places (

place\_id INT(11) NOT NULL AUTO\_INCREMENT,

place\_name VARCHAR(45) NOT NULL,

PRIMARY KEY (place\_id)

);

CREATE TABLE IF NOT EXISTS route\_graphic(

graphic\_id INT(11) NOT NULL AUTO\_INCREMENT,

time\_departure TIME NOT NULL,

time\_arrival TIME NOT NULL,

route\_date DATE NOT NULL,

PRIMARY KEY (graphic\_id)

);

CREATE TABLE IF NOT EXISTS route (

route\_id INT(11) NOT NULL AUTO\_INCREMENT,

graphic\_id INT(11) NOT NULL,

start\_place INT(11) NOT NULL,

end\_place INT(11) NOT NULL,

PRIMARY KEY (route\_id),

FOREIGN KEY (start\_place) REFERENCES places(place\_id),

FOREIGN KEY (end\_place) REFERENCES places(place\_id),

FOREIGN KEY ( graphic\_id) REFERENCES route\_graphic(graphic\_id)

);

CREATE TABLE IF NOT EXISTS trolley (

trolley\_id INT(11) NOT NULL AUTO\_INCREMENT,

id\_brand INT(11) NOT NULL,

route\_id INT (11) NOT NULL,

max\_passangers INT(11) NOT NULL,

last\_date\_of\_check DATE NOT NULL,

PRIMARY KEY (trolley\_id),

FOREIGN KEY (id\_brand) REFERENCES brand(id\_brand),

FOREIGN KEY (route\_id) REFERENCES route(route\_id)

);

CREATE TABLE IF NOT EXISTS booking (

booking\_id INT(11) NOT NULL AUTO\_INCREMENT,

route\_id INT(11) NOT NULL,

passanger\_id INT(11) NOT NULL,

booking\_time TIME NOT NULL,

bookind\_date DATE NOT NULL,

PRIMARY KEY (booking\_id),

FOREIGN KEY (passanger\_id) REFERENCES passanger(passanger\_id),

FOREIGN KEY (route\_id) REFERENCES route(route\_id)

);

Імпортування даних в таблицю

Для імпортування даних у СУБД mysql існує два методи:

* Імпортувати дані кожної таблиці з .csv файлів
* Імпортувати дані із файла скрипта .sql

Для першого методу було створено .csv файли для кожної таблиці.

Для другого методу було написано файл insert.sql:

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (1,'Tommi','Cleave','833214740');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (2,'Reilly','Seson','100678603');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (3,'Elna','Dami','162100167');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (4,'Aaren','Marshall','390415394');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (5,'Adelina','Tran','887588204');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (6,'Amitie',' Cole','530566612');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (7,'Barbee','Cleave','176907028');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (8,'Reilly','Hamilton','423091384');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (9,'Betti','Fisher',' 790574934');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (10,'Chad','Cleave','951038180');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (11,'Cherilynn','Price','100238603');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (12,'Issi','Ward','471535898');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (13,'Larissa','Ramos','315142864');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (14,'Leonie','Kelly','863922893');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (15,'Piper','Morales','264126011');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (16,'Tommi','Gomez','190551421');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (17,'Rosetta','Scott','486262942');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (18,'Sisile','Young','764156068');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (19,'Dabi','Cleave','489582252');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (20,'Todoroki','Martinez','31278603');

INSERT INTO `passanger` (`passanger\_id`,`first\_name`,`last\_name`,`passport\_number`) VALUES (21,'Elsa','Davis','444400167');

INSERT INTO `places` (`place\_id`,`place\_name`) VALUES (1,'Al Fujayrah');

INSERT INTO `places` (`place\_id`,`place\_name`) VALUES (2,'Ash Shāriqah');

INSERT INTO `places` (`place\_id`,`place\_name`) VALUES (3,'Nīmrūz');

INSERT INTO `places` (`place\_id`,`place\_name`) VALUES (4,'Takhār');

INSERT INTO `places` (`place\_id`,`place\_name`) VALUES (5,'Sar-e Pol');

INSERT INTO `places` (`place\_id`,`place\_name`) VALUES (6,'Samangān');

INSERT INTO `places` (`place\_id`,`place\_name`) VALUES (7,'Kabul');

INSERT INTO `places` (`place\_id`,`place\_name`) VALUES (8,'Faryab');

INSERT INTO `places` (`place\_id`,`place\_name`) VALUES (9,'Laghmān');

INSERT INTO `places` (`place\_id`,`place\_name`) VALUES (10,'Balkh');

INSERT INTO `checker` (`checker\_id`,`first\_name`, `last\_name`) VALUES (1,'liam','Consolas');

INSERT INTO `checker` (`checker\_id`,`first\_name`, `last\_name`) VALUES (2,'Oliver','Qeean');

INSERT INTO `checker` (`checker\_id`,`first\_name`, `last\_name`) VALUES (3,'Lucas','Dendi');

INSERT INTO `results` (`result\_id`,`result\_name`) VALUES (1,'+');

INSERT INTO `results` (`result\_id`,`result\_name`) VALUES (2,'-');

INSERT INTO `brand` (`id\_brand`,`brand\_name`) VALUES (1,'Acura');

INSERT INTO `brand` (`id\_brand`,`brand\_name`) VALUES (2,'Alfa');

INSERT INTO `brand` (`id\_brand`,`brand\_name`) VALUES (3,'Audi');

INSERT INTO `brand` (`id\_brand`,`brand\_name`) VALUES (4,'BMW');

INSERT INTO `brand` (`id\_brand`,`brand\_name`) VALUES (5,'Bentley');

INSERT INTO `brand` (`id\_brand`,`brand\_name`) VALUES (6,'Cadillac');

INSERT INTO `brand` (`id\_brand`,`brand\_name`) VALUES (7,'Chevrolet');

INSERT INTO `brand` (`id\_brand`,`brand\_name`) VALUES (8,'Buick');

INSERT INTO `brand` (`id\_brand`,`brand\_name`) VALUES (9,'Mazda');

INSERT INTO `brand` (`id\_brand`,`brand\_name`) VALUES (10,'Ferrari');

INSERT INTO `route\_graphic` (`graphic\_id`,`time\_departure`,`time\_arrival`,`route\_date`) VALUES (1, '09:30:00', '12:30:00','2022-03-06');

INSERT INTO `route\_graphic` (`graphic\_id`,`time\_departure`,`time\_arrival`,`route\_date`) VALUES (2, '04:30:00', '6:30:00','2022-02-16');

INSERT INTO `route\_graphic` (`graphic\_id`,`time\_departure`,`time\_arrival`,`route\_date`) VALUES (3, '03:20:00', '04:30:00','2022-12-08');

INSERT INTO `route\_graphic` (`graphic\_id`,`time\_departure`,`time\_arrival`,`route\_date`) VALUES (4, '14:50:00', '16:20:00','2021-02-06');

INSERT INTO `route\_graphic` (`graphic\_id`,`time\_departure`,`time\_arrival`,`route\_date`) VALUES (5, '16:30:00', '18:00:00','2021-03-08');

INSERT INTO `route\_graphic` (`graphic\_id`,`time\_departure`,`time\_arrival`,`route\_date`) VALUES (6, '20:20:00', '22:00:00','2022-04-01');

INSERT INTO `route\_graphic` (`graphic\_id`,`time\_departure`,`time\_arrival`,`route\_date`) VALUES (7, '05:30:00', '06:00:00','2021-05-01');

INSERT INTO `route\_graphic` (`graphic\_id`,`time\_departure`,`time\_arrival`,`route\_date`) VALUES (8, '05:10:00', '07:30:00','2023-07-26');

INSERT INTO `route\_graphic` (`graphic\_id`,`time\_departure`,`time\_arrival`,`route\_date`) VALUES (9, '06:30:00', '08:00:00','2023-03-22');

INSERT INTO `route\_graphic` (`graphic\_id`,`time\_departure`,`time\_arrival`,`route\_date`) VALUES (10, '10:15:00', '12:15:00','2021-10-23');

INSERT INTO `route\_graphic` (`graphic\_id`,`time\_departure`,`time\_arrival`,`route\_date`) VALUES (11, '09:30:00', '11:30:00','2023-03-28');

INSERT INTO `route\_graphic` (`graphic\_id`,`time\_departure`,`time\_arrival`,`route\_date`) VALUES (12, '10:30:00', '12:20:00','2020-01-08');

INSERT INTO `passanger\_check` (`passanger\_check\_id`,`result\_id`,`checked\_at`,`passanger\_id`,`checker\_id`) VALUES (1, 1, '12:30:00',1, 2);

INSERT INTO `passanger\_check` (`passanger\_check\_id`,`result\_id`,`checked\_at`,`passanger\_id`,`checker\_id`) VALUES (2, 1, '7:30:00',2, 3);

INSERT INTO `passanger\_check` (`passanger\_check\_id`,`result\_id`,`checked\_at`,`passanger\_id`,`checker\_id`) VALUES (3, 1, '13:10:00',7, 2);

INSERT INTO `passanger\_check` (`passanger\_check\_id`,`result\_id`,`checked\_at`,`passanger\_id`,`checker\_id`) VALUES (4, 2, '07:30:00',4, 1);

INSERT INTO `passanger\_check` (`passanger\_check\_id`,`result\_id`,`checked\_at`,`passanger\_id`,`checker\_id`) VALUES (5, 1, '11:20:00',1, 1);

----

INSERT INTO `route` (`route\_id`,`graphic\_id`,`start\_place`,`end\_place`) VALUES (1, 1, 1, 2);

INSERT INTO `route` (`route\_id`,`graphic\_id`,`start\_place`,`end\_place`) VALUES (2, 3, 3, 4);

INSERT INTO `route` (`route\_id`,`graphic\_id`,`start\_place`,`end\_place`) VALUES (3, 4, 5, 6);

INSERT INTO `route` (`route\_id`,`graphic\_id`,`start\_place`,`end\_place`) VALUES (4, 2, 8, 2);

INSERT INTO `route` (`route\_id`,`graphic\_id`,`start\_place`,`end\_place`) VALUES (5, 3, 1, 7);

INSERT INTO `route` (`route\_id`,`graphic\_id`,`start\_place`,`end\_place`) VALUES (6, 7, 5, 6);

INSERT INTO `route` (`route\_id`,`graphic\_id`,`start\_place`,`end\_place`) VALUES (7, 8, 2, 3);

INSERT INTO `trolley` (`trolley\_id`,`id\_brand`,`route\_id`,`max\_passangers`,`last\_date\_of\_check`) VALUES (1, 10, 7, 22, '2019-02-08');

INSERT INTO `trolley` (`trolley\_id`,`id\_brand`,`route\_id`,`max\_passangers`,`last\_date\_of\_check`) VALUES (2, 10, 1, 12, '2020-01-02');

INSERT INTO `trolley` (`trolley\_id`,`id\_brand`,`route\_id`,`max\_passangers`,`last\_date\_of\_check`) VALUES (3, 2, 2, 32, '2018-01-08');

INSERT INTO `trolley` (`trolley\_id`,`id\_brand`,`route\_id`,`max\_passangers`,`last\_date\_of\_check`) VALUES (4, 3, 3, 44, '2010-12-04');

INSERT INTO `trolley` (`trolley\_id`,`id\_brand`,`route\_id`,`max\_passangers`,`last\_date\_of\_check`) VALUES (5, 7, 7, 45, '2020-12-28');

INSERT INTO `trolley` (`trolley\_id`,`id\_brand`,`route\_id`,`max\_passangers`,`last\_date\_of\_check`) VALUES (6, 2, 5, 35, '2020-01-08');

INSERT INTO `trolley` (`trolley\_id`,`id\_brand`,`route\_id`,`max\_passangers`,`last\_date\_of\_check`) VALUES (7, 1, 3, 35, '2019-02-02');

INSERT INTO `trolley` (`trolley\_id`,`id\_brand`,`route\_id`,`max\_passangers`,`last\_date\_of\_check`) VALUES (8, 5, 6, 20, '2020-04-04');

INSERT INTO `booking` (`booking\_id`,`route\_id`,`passanger\_id`,`booking\_time`,`bookind\_date`) VALUES (16,2,12,'15:02:00','2021-01-21');

INSERT INTO `booking` (`booking\_id`,`route\_id`,`passanger\_id`,`booking\_time`,`bookind\_date`) VALUES (17,5,5,'23:30:00','2021-08-05');

INSERT INTO `booking` (`booking\_id`,`route\_id`,`passanger\_id`,`booking\_time`,`bookind\_date`) VALUES (18,5,11,'09:42:00','2021-03-06');

INSERT INTO `booking` (`booking\_id`,`route\_id`,`passanger\_id`,`booking\_time`,`bookind\_date`) VALUES (19,7,16,'23:00:00','2021-12-01');

INSERT INTO `booking` (`booking\_id`,`route\_id`,`passanger\_id`,`booking\_time`,`bookind\_date`) VALUES (20,1,5,'13:43:00','2021-08-30');

Створення користувачів бази даних

Створення пасажира

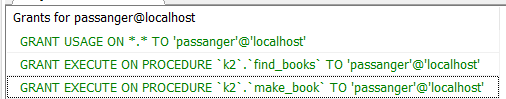
**CREATE** **USER** passanger@localhost **IDENTIFIED** **BY** 'passanger';

**GRANT** **EXECUTE** **ON** **PROCEDURE** k2.make\_book **TO** passanger@localhost;

**GRANT** **EXECUTE** **ON** **PROCEDURE** k2.find\_books **TO** passanger@localhost;

– Надання прав пасажиру

**SHOW** **GRANTS** **FOR** passanger@localhost;



– Права пасажира

Створення перевіряльника

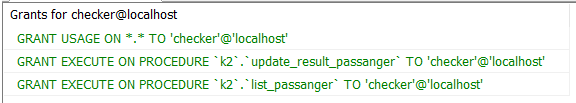
**CREATE** **USER** checker@localhost **IDENTIFIED** **BY** 'checker';

**GRANT** **EXECUTE** **ON** **PROCEDURE** k2.update\_result\_passanger **TO** checker@localhost;

**GRANT** **EXECUTE** **ON** **PROCEDURE** k2.list\_passanger **TO** checker@localhost;

– Надання прав контролеру

**SHOW** **GRANTS** **FOR** checker@localhost;



– Права контролера

SQL запити

Створення тригерів на таблиці, що будуть змінюватись користувачами.

-- booking

DROP TRIGGER IF EXISTS booking\_insert;

DELIMITER //

CREATE TRIGGER booking\_insert BEFORE INSERT

ON booking FOR EACH ROW

BEGIN

DECLARE max\_passangers INT;

IF (NEW.route\_id > (SELECT COUNT(route\_id) FROM route) OR NEW.route\_id<=0) THEN

SIGNAL SQLSTATE '99999'

SET MESSAGE\_TEXT = 'No such route';

END IF;

SET max\_passangers = (SELECT t.max\_passangers FROM trolley t WHERE t.route\_id = NEW.route\_id);

IF (max\_passangers<(SELECT COUNT(booking\_id) FROM booking WHERE route\_id = NEW.route\_id)) THEN

SIGNAL SQLSTATE '99999'

SET MESSAGE\_TEXT = 'Have not free seats';

END IF;

END //

DELIMITER ;

-- checks

DROP TRIGGER IF EXISTS passanger\_check\_update;

DELIMITER //

CREATE TRIGGER passanger\_check\_update BEFORE UPDATE

ON passanger\_check FOR EACH ROW

BEGIN

IF (NEW.passanger\_id > (SELECT COUNT(passanger\_id) FROM passanger) OR NEW.passanger\_id<=0) THEN

SET NEW.passanger\_id = OLD.passanger\_id;

END IF;

IF (NEW.result\_id > 2 OR NEW.result\_id <= 0) THEN

SET NEW.result\_id = OLD.result\_id;

END IF;

END //

DELIMITER ;

UPDATE passanger\_check SET result\_id = 5 WHERE result\_id = 2;

DROP TRIGGER IF EXISTS passanger\_check\_insert;

DELIMITER //

CREATE TRIGGER passanger\_check\_insert BEFORE INSERT

ON passanger\_check FOR EACH ROW

BEGIN

IF (NEW.passanger\_id > (SELECT COUNT(passanger\_id) FROM passanger) OR NEW.passanger\_id<=0) THEN

SIGNAL SQLSTATE '99999'

SET MESSAGE\_TEXT = 'No such passanger';

END IF;

IF (NEW.result\_id > 2 OR NEW.result\_id <= 0) THEN

SIGNAL SQLSTATE '99999'

SET MESSAGE\_TEXT = 'No such result';

END IF;

SET NEW.checked\_at = current\_time();

IF (NEW.checker\_id > (SELECT COUNT(checker\_id) FROM checker) OR NEW.checker\_id<0) THEN

SIGNAL SQLSTATE '99999'

SET MESSAGE\_TEXT = 'Invalid number of checker!';

END IF;

END //

DELIMITER ;

DROP TRIGGER IF EXISTS passangers\_insert;

DELIMITER //

CREATE TRIGGER passangers\_insert BEFORE INSERT

ON passanger FOR EACH ROW

BEGIN

IF (SELECT EXISTS(SELECT passport\_number FROM passanger WHERE passport\_number = NEW.passport\_number)) THEN

SIGNAL SQLSTATE '99999'

SET MESSAGE\_TEXT = 'Current number of passport exist!';

END IF;

END //

DELIMITER ;

DROP TRIGGER IF EXISTS passangers\_after\_insert;

DELIMITER //

CREATE TRIGGER passangers\_after\_insert AFTER INSERT

ON passanger FOR EACH ROW

BEGIN

INSERT INTO `passanger\_check` (`result\_id`,`checked\_at`,`passanger\_id`,`checker\_id`) VALUES (1,current\_time(),NEW.passanger\_id,(SELECT RAND()\*(SELECT COUNT(checker\_id)-1 FROM checker)+1));

END //

DELIMITER ;

Створення процедур для взаємодії користувачів з базою даних

Процедури пасажира

DROP PROCEDURE IF EXISTS make\_book;

DELIMITER //

CREATE PROCEDURE make\_book(IN passanger INT(11), IN route INT(11))

BEGIN

DECLARE num\_of\_booking INT;

SET num\_of\_booking = (SELECT COUNT(booking\_id) FROM booking)+1;

INSERT INTO `booking` (`booking\_id`,`route\_id`,`passanger\_id`,`booking\_time`,`bookind\_date`) VALUES (num\_of\_booking,route,passanger,current\_time(),current\_date());

END //

DELIMITER ;

**call** make\_book(21,5);



– Приклад виконання замовлення

DROP PROCEDURE IF EXISTS find\_books;

DELIMITER //

CREATE PROCEDURE find\_books(IN passanger INT(11))

BEGIN

SELECT \* FROM booking WHERE passanger\_id = passanger;

END //

DELIMITER ;

**CALL** find\_books(11);



– Пошук бронювань 11 пасажиром

Процедури перевіряльника пасажирів

DROP PROCEDURE IF EXISTS update\_result\_passanger;

DELIMITER //

CREATE PROCEDURE update\_result\_passanger(IN checker INT, IN passanger INT(11), IN result INT)

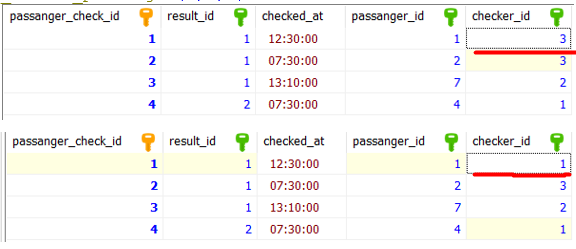
BEGIN

UPDATE passanger\_check SET result\_id = result, checker\_id = checker WHERE passanger\_id = passanger;

END //

DELIMITER ;

**call** update\_result\_passanger(2,1,2);



– Оновлення даних про перевірку пасажира

DROP PROCEDURE IF EXISTS list\_passanger;

DELIMITER //

CREATE PROCEDURE list\_passanger()

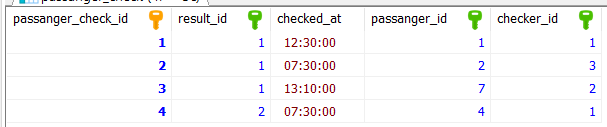
BEGIN

SELECT \* FROM passanger\_check;

END //

DELIMITER ;

**CALL** list\_passanger();



– Перегляд списку пасажирів

**Інші sql запити**

Створення VIEW для відображення номеру перевірки пасажира, її результат та час, і ім’я перевіряючого.

CREATE OR REPLACE VIEW `Passanger checks and checkers` AS

SELECT tc.passanger\_check\_id, passanger\_id, c.first\_name, c.last\_name, tc.checked\_at, r.result\_name

FROM checker c, passanger\_check tc, results r

WHERE c.checker\_id = tc.checker\_id AND tc.result\_id = r.result\_id

**SELECT** \* **FROM** k2.`passanger checks and checkers`;

 – Запит представлення

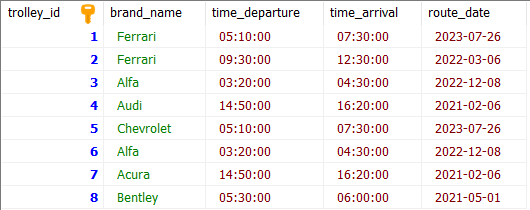
Запит для перегляду тролейбуса та час його відправлення

CREATE OR REPLACE VIEW `trolley and his time ` AS

SELECT t.trolley\_id, b.brand\_name, g.time\_departure, g.time\_arrival, g.route\_date FROM trolley t, route\_graphic g, route r, brand b

WHERE t.id\_brand = b.id\_brand AND t.route\_id = r.route\_id AND r.graphic\_id=g.graphic\_id

**SELECT** \* **FROM** k2.`trolley and his time`;



– Запит представлення

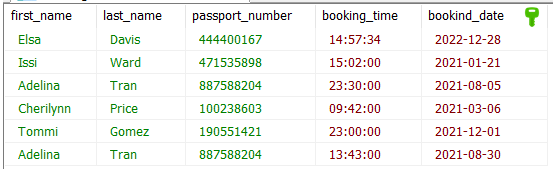
Запит щоб подивитися пасажирів та їх інформацію: паспорт, час та дата бронювання

CREATE OR REPLACE VIEW `Passanger and his book` AS

SELECT p.first\_name, p.last\_name, p.passport\_number, b.booking\_time, b.bookind\_date FROM passanger p, booking b, route r

WHERE p.passanger\_id = b.passanger\_id AND b.route\_id = r.route\_id

**SELECT** \* **FROM** k2.`Passanger and his book`;



– Запит представлення

Запит для знаходження тролейбусів та їх марки, де дата останнього тех огляду прострочена

CREATE OR REPLACE VIEW `old last date of check` AS

SELECT t.trolley\_id, b.brand\_name, t.last\_date\_of\_check

FROM trolley t, brand b

WHERE t.last\_date\_of\_check < date ‘2020-01-01’ AND b.id\_brand = t.id\_brand;

**SELECT** \* **FROM** k2.`old last date of check`;



– Запит представлення

Запит для знаходження макс кількості пасажирів на маршруті

SELECT t.route\_id, t.max\_passangers

FROM trolley t

WHERE t.route\_id = (SELECT r.route\_id FROM booking b, route r WHERE b.route\_id = r.route\_id AND b.booking\_id=6);



– Результат запиту

Створення запиту з використанням INNER та OUTER JOIN для того, щоб дізнатись, результати перевірки пасажирів та тих хто їх перевіряв.

SELECT myData.passanger\_check\_id, myData.result\_name, CONCAT(myData.first\_name,' ',myData.last\_name) AS checker\_full\_name

FROM (

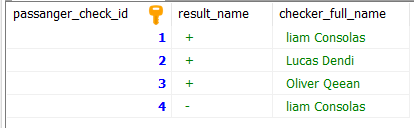
SELECT tc.passanger\_check\_id, passanger\_id, c.first\_name, c.last\_name, tc.checked\_at, r.result\_name

FROM checker c

JOIN passanger\_check tc ON c.checker\_id = tc.checker\_id

LEFT JOIN results r ON tc.result\_id = r.result\_id

) AS myData;



– Результат запиту

Створення запиту з використанням функції CONCAT для визначення імен пасажирів, які зареєструвалися пізніше 10 пасажира.

SELECT passanger\_id, CONCAT(first\_name,' ', last\_name) AS full\_name FROM passanger WHERE passanger\_id > 10;



– Результат запиту

Запит для знаходження інформації про пасажирів, які зробили хоч одне бронювання

SELECT \* FROM passanger p

WHERE EXISTS(SELECT booking\_id FROM booking b WHERE b.passanger\_id = p.passanger\_id);

SELECT \* FROM passanger p

WHERE IN(SELECT booking\_id FROM booking b WHERE b.passanger\_id = p.passanger\_id);



– Результат запиту

Запит для знаходження пасажирів та їх бронювань

SELECT p.passanger\_id, CONCAT(p.first\_name,' ',p.last\_name) AS passanger\_full\_name, booking\_id

FROM passanger p

JOIN booking b

ON p.passanger\_id = b.passanger\_id;



– Результат запиту

Створення запиту для пошуку бренда тролейбуса на маршруті

SELECT r.route\_id, brand\_name FROM route r, trolley t, brand b WHERE t.id\_brand = b.id\_brand AND t.route\_id = r.route\_id

ORDER BY route\_id ASC;



– Результат запиту

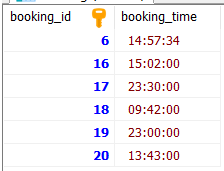
Приклад роботи індексів

Для прикладу розглянемоо простий SQL запит типу:

SELECT DISTINCT booking\_id, booking\_time

FROM booking

WHERE bookind\_date BETWEEN CONVERT('2015-01-01',DATE) AND CONVERT('2023-01-01',DATE);



З допомогою даного запиту ми можемо обрати бронювання виконані, між 2015-01-01 та 2021-01-01. Час виконання 0,047s

А зараз створимо індекс на поле `booking\_date` виконивши такий скрипт:

CREATE INDEX bookind\_date ON booking(bookind\_date);

Після повторного виконання запиту уже із створеним індексом отримаємо час виконання 0,016s

Бачимо, що різниця є(в 2,9375 рази швидше), тому що з індексами потрібно перевірити меншу кількість рядків

Ефективність збільшилась, отже використання індексів – невід’ємна частина роботи з базою даних.

Висновок

Створення бази даних – це один із фундаментів для початку ефективного та продуктивного бізнесу, адже від якості групування даних напряму залежать можливості обробки даних та кількість реалізованих одиниць товару.

У даній роботі було спроектовано базу даних для аеропорту. Спочатку було вивчено та описано предметне середовище, після чого визначені задачі для виконання.

Наступним кроком було створення ER-моделі бази даних, для кращого розуміння та представлення, які сутності будуть брати участь в роботі аеропорту.

Після чого, було створено реляційну модель бази даних приведену до третьої нормальної форми.

В завершення, для створення бази даних було виконано:

* написано SQL скрипти для створення таблиць
* написано тригери для перевірки коректності введених даних, де це необхідно
* створено користувачів бази даних та функції для них

Перелік посилань

[PostgreSQL: Documentation](https://www.postgresql.org/docs/)

[PostgreSQL Tutorial – Comprehensive Postgresql Tutorial](https://www.postgresqltutorial.com/)

[PostgresPro](https://postgrespro.ru/docs/postgresql/9.6/app-psql)