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 з підтримки діяльності підрозділу обслуговування автотранспорту

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Студента | 2 | курсу | | ІП-22 | групи | |
| спеціальності 121 «Інженерія програмного забезпечення» | | | | | | |
| Панасюк О.Ю. | | | | | | |
| (прізвище та ініціали) | | | | | | |
| Керівник: старший викладач Марченко О.І. | | | | | | |
| (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 – 2023 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 ㅤ2ㅤ Група ㅤІП-22ㅤ Семестр ㅤ3ㅤ

**ЗАВДАННЯ**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

ㅤㅤㅤㅤㅤㅤㅤㅤ ㅤ Панасюку Олександру Юрійовичу

(прізвище, ім’я, по батькові)

1. Тема роботи База даних з підтримки діяльності підрозділу обслуговування автотранспорту ㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤ

керівник роботи старший викладач Марченко Олена Іванівнаㅤㅤ ㅤㅤㅤㅤ

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

2. Строк подання студентом роботи ㅤ30.12.2023 ㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤ

3. Вихідні дані до роботи завдання на розробку бази даних з підтримки діяльності підрозділу обслуговування автотранспорту

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

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

7) Створення мовою SQL запитів

8) Оптимізація роботи запитів

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

ㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤ ㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤ

6. Дата видачі завдання ㅤ31.11.2023**ㅤㅤ**

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
| 1 | Аналіз предметного середовища | 02.12.2023 |  |
| 2 | Побудова ER-моделі | 10.12.2023 |  |
| 3 | Побудова реляційної схеми з ER-моделі | 16.12.2023 |  |
| 4 | Створення бази даних, у форматі обраної системи управління базою даних | 20.12.2023 |  |
| 5 | Створення користувачів | 24.12.2023 |  |
| 6 | Створення мовою SQL запитів | 26.12.2023 |  |
| 7 | Оптимізація роботи запитів | 29.12.2023 |  |
| 8 | Оформлення пояснювальної записки | 30.12.2023 |  |
| 9 | Захист курсової роботи | 5.01.2024 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Студент** |  |  | Панасюк О.Ю. |  |
|  | (підпис) |  | (Прізвище та ініціали) |  |
| **Керівник роботи** |  |  | Марченко О.І. |  |
|  | (підпис) |  | (Прізвище та ініціали) |  |

**ЗМІСТ**

[ВСТУП 5](#_Toc154880494)

[1 Опис предметного середовища 6](#_Toc154880495)

[1.1. Бізнес-процеси 6](#_Toc154880496)

[1.2. Обмеження 8](#_Toc154880497)

[1.3. Користувачі 9](#_Toc154880498)

[1.4. Вхідні дані 9](#_Toc154880499)

[1.5. Вихідні дані 9](#_Toc154880500)

[2 Аналіз існуючих програмних продуктів 10](#_Toc154880501)

[3 Постановка Завдання 11](#_Toc154880502)

[4 Опис ER-моделі 12](#_Toc154880503)

[4.1. Опис сутностей 12](#_Toc154880504)

[4.2. Опис атрибутів сутностей 13](#_Toc154880505)

[4.3. Опис зв’язків 14](#_Toc154880506)

[4.4. ER-діаграма 15](#_Toc154880507)

[5 Реляційна модель 16](#_Toc154880508)

[6 Створення бази даних 23](#_Toc154880509)

[6.1. Вибір СУБД 23](#_Toc154880510)

[6.2. Створення бази даних 23](#_Toc154880511)

[6.3. Наповнення бази даних 25](#_Toc154880512)

[6.4. Створення користувачів 26](#_Toc154880513)

[7 Робота з Базою даних 28](#_Toc154880514)

[7.1. Генератори 28](#_Toc154880515)

[7.2. Збережені процедури та функції 28](#_Toc154880516)

[7.3. Тригери 40](#_Toc154880517)

[7.4. Представлення 50](#_Toc154880518)

[7.5. Запити 53](#_Toc154880519)

[7.6. Оптимізація 64](#_Toc154880520)

[ВИСНОВКИ 68](#_Toc154880521)

[Список використаної літератури 69](#_Toc154880522)

[Додаток А SQL скрипти створення бази даних 70](#_Toc154880523)

[Додаток Б SQL скрипти логічних обмежень 75](#_Toc154880524)

[Додаток В SQL скрипт заповнення бази даних 76](#_Toc154880525)

[Додаток Г SQL скрипти створення ролей та користувачів 80](#_Toc154880526)

# ВСТУП

Забезпечення роботи підрозділу з обслуговування автотранспорту є важливим процесом будь-якої компанії, що працює у цій галузі. Точний моніторинг технічного стану автомобілів та автоматизація керування їхнім ремонтом сприяє оптимізації робочих процесів та підвищення якості обслуговування. Задля роботи цих механізмів необхідним є створення організованої сукупності даних, бази даних. Розробка та впровадження такої бази даних сприятиме підвищенню ефективності роботи автосервісу, зменшенню часу на пошук інформації, покращенню обслуговування та оптимізації витрат на утримання підрозділу. Варто також зазначити, що обслуговування транспорту можна розглядати не тільки в контексті компанії, а навіть безпеки держави. Військовий транспорт потребує ремонту ще частіше ніж звичайний, тому ефективний та швидкий ремонт може надати стратегічну перевагу у бою перед противником.

# Опис предметного середовища

Автомобілі компанії потребують своєчасного ремонту та технічного огляду. Огляд проводиться періодично і фіксує технічний стан деталей та систем автомобіля. Ремонт проводиться, коли водій звертається із скаргою на несправність. Ремонт може включати в себе різні процедури наприклад, заміна мастила, шиномонтаж, кузовні роботи та інше. Також в процесі ремонту можуть бути використані деталі, які потрібно встановити. Ремонт потребує використання певного обладнання, таких як підйомники, оглядові ями , шиномонтажні станки, тестові стенди тощо. А оскільки навантаженість на підрозділ, може бути великою, то з’являється потреба в ефективному використанні обладнання. Для вирішення цієї проблеми створюються розклади роботи обладнання, які допомагають планувати їх використання. Аналогічна проблема виникає із використанням людських ресурсів, тому для працівників теж складають плани проведення робіт. Ці плани несуть інформацію про те, хто та в який час має займатися ремонтом певного автомобіля.

Основні компоненти системи:

1. Автомобіль
2. Технічний огляд
3. Ремонт
4. Послуга
5. Деталь
6. Обладнання
7. Працівник підрозділу
8. Розклад роботи обладнання
9. Розклад роботи працівників

## Бізнес-процеси

Основний бізнес процес ремонту автомобіля можна описати наступним чином:

1. **Прийом водія та аналіз проблеми**. Водій приїжджає до сервісного центру та докладно описує виниклу проблему з автомобілем. Спеціалісти приймають цю інформацію та проводять первинний аналіз несправності. У випадку, якщо причина поломки неочевидна, автомобіль оглядають та визначають кореневу причину несправності. Далі менеджер підрозділу реєструє ремонт в системі.
2. **Планування та розподіл робіт**. На основі виявлених проблем складається план ремонту, визначається необхідний персонал та розклад робіт. Інформація про ремонт автомобіля, перелік послуг та графік роботи працівників реєструється.
3. **Бронювання обладнання**. Якщо для виконання робіт потрібне спеціалізоване обладнання, робітник бронює його на певний час. Інформація про бронювання обладнання вноситься в розклад.
4. **Заміна деталей**. У випадку необхідності заміни деталей робітник вносить інформацію про деталі в систему.
5. **Завершення ремонту**. Після успішного завершення робіт та тестування автомобіля, інформація про готовність транспортного засобу передається менеджеру автопарку. Менеджер автопарку зв'язується з водієм та надає всю необхідну інформацію щодо готовності автомобіля для використання.

Бізнес процес проведення планового технічного огляду:

1. **Запит на огляд від менеджера підрозділу**. Менеджер підрозділу переглядає перелік автомобілів у його відділі та виявляє транспортні засоби, які потребують технічного огляду або потенційного обслуговування. Після ідентифікації таких автомобілів, менеджер підрозділу надсилає запит менеджерові автопарку щодо необхідності проведення огляду.
2. **Пригін автомобілів до підрозділу обслуговування**. Автомобілі, що виявилися у списку для огляду, направляються до підрозділу обслуговування. Тут їх зустрічає відповідальний персонал, який забезпечує надходження транспортних засобів на обслуговування.
3. **Технічний огляд та реєстрація результатів**. Кваліфікований експерт проводить детальний огляд кожного автомобіля, визначаючи наявність технічних проблем чи потребу в обслуговуванні. Експерт перевіряє стан моторного масла, зчеплення, гальмівної системи, ходової частини, кузова та інших частин. Також він фіксує пробіг автомобіля. Результати огляду вносяться в систему.
4. **Ремонт.** У випадку виявлення несправностей, які потребують ремонту чи обслуговування, запускається бізнес-процес ремонту. Проводиться планування та організація робіт, призначається персонал та виконуються інші необхідні дії для відновлення автомобіля до робочого стану.
5. **Повернення на автостоянку**. У випадку, якщо результати технічного огляду підтверджують, що автомобіль відповідає установленим нормам, менеджер підрозділу повідомляє менеджера автопарку про закінчення огляду, після чого автомобіль забирають на автостоянку.

## Обмеження

Далі наведені обмеження властивостей компонент, які логічно випливають з їх суті. Порушення цих правил може спричинити до помилки в системі.

1. Ціна та кількість деталей не може бути від’ємним числом.
2. Рік випуску машин не може бути більшим за поточний.
3. Номерні знаки машини повинні відповідати встановленому формату.
4. Пробіг зафіксований на одному із оглядів авто повинен бути більший або рівний пробігу з попереднього огляду.
5. Дата закінчення ремонту має бути пізніше за дату початку.
6. Часові проміжки бронювання обладнання не мають перетинатися.
7. Часові проміжки роботи працівника над різними машинами не повинні перетинатися.
8. Час закінчення бронювання має бути пізніше часу початку.
9. Дата технічного огляду не може бути більша за поточну

## Користувачі

З наведених бізнес-процесів можна виділити основні ролі користувачів, які будуть взаємодіяти з системою:

1. **Менеджер підрозділу.** Виконує реєстрацію нових ремонтів та повинен отримувати інформацію про автомобілі, які не проходили технічний огляд протягом заданого терміну.
2. **Аналітик.** Роль має доступ до читання з всіх таблиць, для проведення аналізів ефективності та завантаженості персоналу та обладнання та інших показників.
3. **Планувальник.** Має доступ до плану роботи працівників та розкладу обладнання. Створює записи про бронювання обладнання та вносить інформацію про план робіт.
4. **Інспектор.** Роль має доступ тільки до інформації про технічні огляди автомобілів. Повинен вносити інформацію про них в систему.

## Вхідні дані

Вхідними даними для системи є дані про автомобілі, результати технічного огляду, розклади роботи обладнання та розклади роботи працівників. Також сюди входить інформація про несправності автомобілів та деталі, що потрібно замінити.

## Вихідні дані

Вихідними даними для системи є інформація про виконані ремонтні роботи, історія технічних оглядів та обслуговування автомобілів, звіти використаних деталей, звіти завантаженості обладнання та робітників.

# Аналіз існуючих програмних продуктів

Існує кілька продуктів та платформ, які можуть застосовуватись для підтримки роботи підрозділу з обслуговування автомобілів. Далі наведені декілька з них:

1. RemOnline. Сервіс дозволяє зберігати дані про клієнтів, про їх авто та їх стан. Забезпечує оптимальне завантаження підйомників і постів автосервісу. Надає можливість відстежувати тривалість робіт, щоб ефективно розпоряджатися часом. Також має інструменти, які рівномірно розподіляють навантаження між працівниками.
2. CarBook. Система зберігає історію всіх ремонтів та дозволяє планувати завантаження по постам і слюсарям-механікам. Має функціонал відкриття, закриття змін та роботи з чергою автомобілів.
3. SalesDrive. CRM-система для обліку ремонтів авто. Система веде базу автомобілів, має календар зайнятості спеціалістів та інформує про необхідність проведення технічного огляду. Має можливість обліку витрат та прибутків автосервісу.
4. CRM Appointer. Система включає електронний календар для запису клієнтів, детальну базу даних по клієнтам, автоматичний розрахунок зарплат співробітників, дистанційний запис в автосервіс для водіїв, облік на складі, а також надає докладні аналітичні і статистичні дані. Програма також підтримує IP-телефонію, дозволяє переносити дані з інших програм.

# Постановка Завдання

Метою даної курсової роботи є розробка бази даних для підтримки діяльності підрозділу, що обслуговує автотранспорт. Розроблене рішення відповідати наступним вимогам:

1. Цілісність даних та безпечне їх зберігання.
2. Автоматизація процесів проведення регулярного технічного огляду.
3. Збереження історії всіх ремонтів та технічних оглядів.
4. Облік використаних деталей та зроблених послуг.
5. Збереження інформації про розклади роботи обладнання та працівників.

Розробка такого програмного забезпечення є складним та комплексним процесом, тому доцільно декомпозувати роботу на наступні кроки:

1. Побудова ER-моделі.
2. Побудова реляційної моделі на основі розробленої ER-моделі.
3. Фізична реалізація бази даних, використовуючи СКБД PostgreSQL.
4. Заповнення бази тестовими даними.
5. Створити запити та процедури, що відповідають предметній області та автоматизують певні бізнес-процеси підрозділу.
6. Реалізація багатокористувацької моделі доступу.
7. Оптимізація запитів.

# Опис ER-моделі

Перед початком створення моделі “сутність-зв’язок” потрібно проаналізувати сутності, встановити їхні атрибути та описати зв’язки між сутностями.

## Опис сутностей

Далі наведені основні інформаційні об’єкти та їх описи, що були виділені на етапі аналізу предметного середовища:

* **Автомобіль (Car) –** сутність несе інформацію про саме авто та його унікальні номерні знаки.
* **Технічний огляд (Inspection)** – сутність, яка містить інформацію про результати огляду та дату, коли він був проведений. Також фіксує пробіг автомобіля.
* **Ремонт (Repair)** – об’єкт, який описує процес ремонтування авто та зберігає інформацію про виявлені несправності транспорту**.** Має зв'язок із деталями, що були замінені, та із послугами, що були надані.
* **Деталь (Detail)** – сутність, що описує деталь, яка зберігається на складі.
* **Послуга (Service)** – сутність, яка містить інформацію про обслуговування авто, яке може виконати підрозділ.
* **Обладнання (Equipment)** – інформаційний об’єкт, що репрезентує фізичне обладнання підрозділу.
* **Тип обладнання (Equipment type)** – об’єкт, що об’єднує обладнання в категорії.
* **Працівник (Employee)** – сутність, яка несе інформацію про робітника, його кваліфікацію, професію, досвід роботи та контакти.
* **Професія (Profession)** – сутність, що описує спеціалізацію працівника.
* **Розклад роботи обладнання (Equipment schedule)** – об’єкт, що містить інформацію про час, протягом якого обладнання буде задіяне в ремонті автомобіля.
* **Розклад роботи працівників (Work schedule)** – сутність, що містить інформацію про час, протягом якого працівник буде працювати над ремонтом авто.

## Опис атрибутів сутностей

Опишемо набори атрибутів сутностей.

Таблиця 4.1 – Опис атрибутів.

|  |  |
| --- | --- |
| Назва сутності | Атрибути |
| Автомобіль | Ідентифікатор авто, назва, номерний знак, рік випуску |
| Технічний огляд | Ідентифікатор огляду, результати огляду, пробіг автомобіля, дата огляду |
| Ремонт | Ідентифікатор ремонту, опис несправності, дата початку, дата кінця |
| Деталь | Ідентифікатор деталі, назва, вартість |
| Послуга | Ідентифікатор послуги, назва |
| Обладнання | Ідентифікатор обладнання, назва |
| Тип обладнання | Ідентифікатор типу, назва |
| Працівник | Ідентифікатор працівника, ПІБ, телефон, досвід роботи, кваліфікація |
| Професія | Ідентифікатор професії, назва |
| Розклад роботи обладнання | Ідентифікатор розкладу, дата початку, дата кінця |
| Розклад роботи працівників | Ідентифікатор розкладу, дата початку, дата кінця |

## Опис зв’язків

Виділені сутності мають наступні зв’язки між собою:

Таблиця 4.2 – Опис зв’язків

|  |  |  |  |
| --- | --- | --- | --- |
| Перша сутність | Друга сутність | Тип зв’язку | Опис |
| Автомобіль | Ремонт | 1:M | Автомобіль може ремонтуватися багато раз |
| Автомобіль | Технічний огляд | 1:M | Для одного автомобіля може проводитись багато технічних оглядів |
| Тип обладнання | Обладнання | 1:M | Багато одиниць обладнання можуть мати один тип |
| Професія | Працівник | 1:M | Багато працівників можуть мати одну професію |
| Деталь | Ремонт | M:M | Протягом одного ремонту можуть замінюватись багато деталей і одна деталь може бути задіяна в декількох ремонтах |
| Послуга | Ремонт | M:M | Протягом одного ремонту можуть надаватися декілька різних послуг і одна й та сама послуга може надаватися в багатьох ремонтах |
| Обладнання | Розклад роботи обладнання | 1:M | На обладнанні можуть застосовувати для багатьох ремонтів в різні проміжки часу. |
| Ремонт | Розклад роботи обладнання | 1:M | Один ремонт може відбуватися на багатьох приладах |
| Працівник | Розклад роботи працівників | 1:M | Один працівник може працювати над багатьма ремонтами в різні проміжки часу |
| Ремонт | Розклад роботи працівників | 1:M | Над одним ремонтом може працювати багато працівників |

## ER-діаграма

Візьмемо до уваги всі описи сутностей, їх атрибутів і зв’язків та побудуємо діаграму “сутність-зв’язок”.

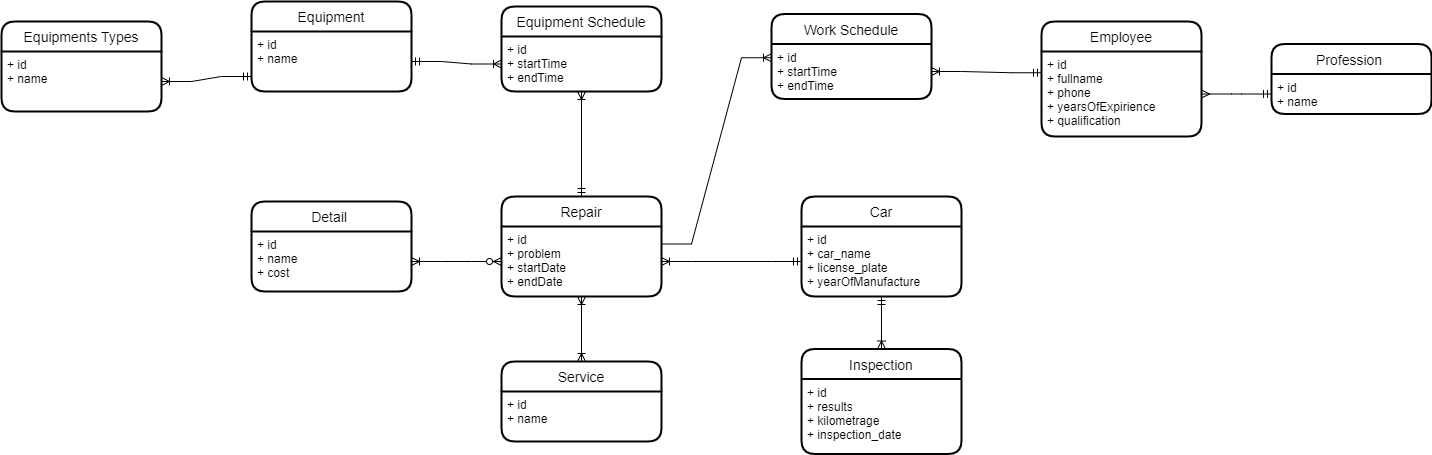


Рисунок 4.1 – ER- діаграма бази даних для підрозділу обслуговування автотранспорту

# Реляційна модель

Перед етапом фізичної реалізації бази даних слід проаналізувати всі ключові відношення між сутностями та розробити реляційну модель бази даних. У цьому розділі на основі ER-моделі будуть описані всі відношення та обмеження на атрибути, що були описані у описі предметного середовища.

Таблиця “Cars” зберігає дані про автомобілі, що є в автопарку. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор автомобіля | Identity, not null |
| car\_name | varchar | 50 | - | Марка та модель авто | Not null |
| license\_plate | varchar | 8 | - | Номерні знаки | Unique, not null |
| year\_of\_manufacture | int | - | - | Дата випуску | ≤ поточного року, not null |

Таблиця “Inspections” містить інформацію про проведений технічний огляд автомобіля, який був проведений для автомобіля. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор огляду | Identity, not null |
| results | text | - | - | Опис результатів огляду | Not null |

Продовження таблиці “Inspections”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| kilometrage | int | - | - | Пробіг автомобіля | Not null,  ≥ пробіг з попереднього огляду |
| inspection\_date | date | - | - | Дата проведення огляду | Not null,  ≤ поточна дата |
| car\_id | int | - | FK | Ідентифікатор автомобіля | Not null |

Таблиця “Repairs” призначена для зберігання даних про процес ремонту. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор ремонту | Identity, not null |
| problem | text | - | - | Опис несправності | Not null |
| startDate | date | - | - | Дата початку ремонту | Not null |
| endDate | date | - | - | Дата кінця ремонту |  |
| car\_id | int | - | FK | Ідентифікатор автомобіля | Not null |

Таблиця “Details” зберігає інформацію про деталі, що є в розпорядженні підрозділу. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор деталі | Identity, not null |
| name | varchar | 100 | - | Назва деталі | Not null |
| cost | int | - | - | Ціна закупки | Not null, |

Таблиця “Services” зберігає інформацію про послуги, що може надавати підрозділ. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор послуги | Identity, not null |
| name | varchar | 50 | - | Назва послуги | Not null |

Таблиця “Repairs\_details” містить інформацію про деталі, що були замінені під час ремонту. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор зв’язку | Identity, not null |
| repair\_id | int | - | FK | Ідентифікатор ремонту | Not null |
| detail\_id | int | - | FK | Ідентифікатор деталі | Not null |
| number | int | - | - | Кількість деталей | Not null, |

Таблиця “Repairs\_services” містить інформацію про послуги, що були виконані під час ремонту. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор зв’язку | Identity, not null |
| repair\_id | int | - | FK | Ідентифікатор ремонту | Not null |
| service\_id | int | - | FK | Ідентифікатор послуги | Not null |

Таблиця “Equipment\_types” зберігає інформацію про типи обладнання підрозділу. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор типу | Identity, not null |
| name | varchar | 100 | - | Назва типу | Not null |

Таблиця “Equipment” містить інформацію про наявне обладнання підрозділу. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор обладнання | Identity, not null |
| name | varchar | 100 | - | Назва обладнання | Not null |
| type\_id | int | - | FK | Ідентифікатор типу | Not null |

Таблиця “Employees” містить інформацію про працівників підрозділу. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор працівника | Identity, not null |
| fullname | varchar | 100 | - | ПІБ працівника | Not null |
| phone | varchar | 15 | - | Телефон працівника | Not null |
| years\_of\_experience | int | - | - | Досвід роботи в роках | Not null,  ≥ 0 |
| qualification | text | - | - | Кваліфікація | - |
| profession\_id | int | - | FK | Ідентифікатор професії | Not null |

Таблиця “Professions” зберігає інформацію про різні спеціальності працівників. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор професії | Identity, not null |
| name | varchar | 50 | - | Назва професії | Not null |

Таблиця “Equipment\_schedule” призначена для зберігання інформації про розклад роботи обладнання. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор розкладу | Identity, not null |
| startTime | timestamp | - | - | Час початку роботи обладнання | Not null,  Для одного обладнання часові проміжки не мають перетинатися |
| endTime | timestamp | - | - | Час кінця роботи обладнання |
| equipment\_id | int | - | FK | Ідентифікатор обладнання | Not null |
| repair\_id | int | - | FK | Ідентифікатор ремонту | Not null |

Таблиця “Work\_schedule” призначена для зберігання інформації про розклад роботи працівників з різними ремонтами. Структура таблиці наступна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int | - | PK | Ідентифікатор розкладу | Identity, not null |
| startTime | timestamp | - | - | Час початку роботи працівника | Not null,  Для одного обладнання часові проміжки не мають перетинатися |
| endTime | timestamp | - | - | Час кінця роботи працівника |

Продовження таблиці “Work\_schedule”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| employee\_id | int | - | FK | Ідентифікатор працівника | Not null |
| repair\_id | int | - | FK | Ідентифікатор ремонту | Not null |

Далі наведена діаграма, що репрезентує розроблену реляційну модель. Діаграма реляційної моделі даних наведена нижче відображає структуру бази даних та зв'язки між різними сутностями.

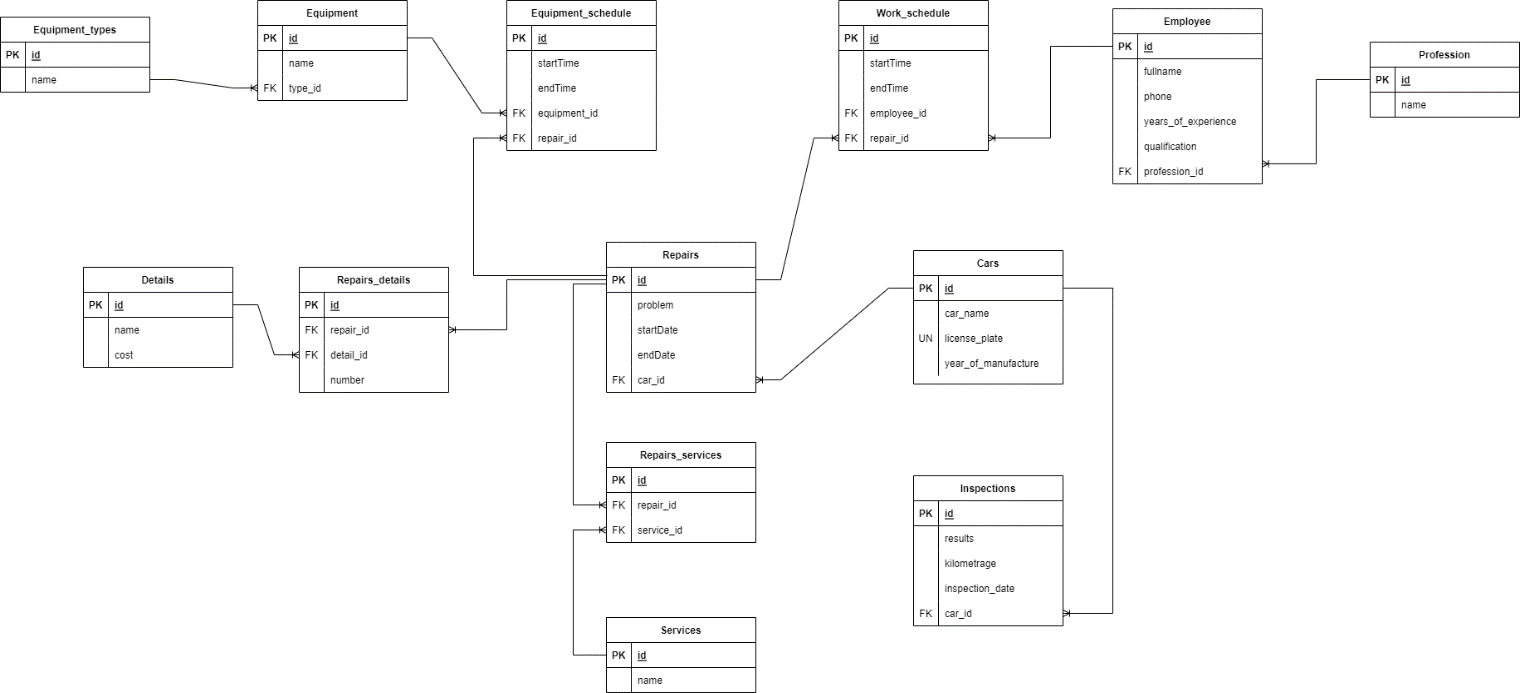


Рисунок 5.1 – Діаграма реляційної моделі для бази даних з підтримки діяльності підрозділу обслуговування автотранспорту

У даному розділі курсової роботи була розроблена реляційна модель даних, яка є ключовим концептуальним складником організації і структурування інформації в системі. Модель включає низку табличних представлень, що визначають сутності, атрибути та взаємозв'язки в базі даних та її графічне представлення в виді діаграми.

# Створення бази даних

## Вибір СУБД

Для створення фізичної бази даних було обрано СУБД PostgreSQL. Вирішальними при виборі системи стали декілька факторів.

По-перше, PostgreSQL підтримує розширений набір функцій SQL і включає ряд додаткових функцій, які роблять його чудовим вибором для проектів різної складності. Наявність розширених інструментів, таких як загальні типи, географічні об’єкти та повнотекстовий пошук, робить PostgreSQL гнучким і придатним для різноманітних завдань.

Другим важливим аспектом є висока надійність і стабільність PostgreSQL. Ця СУБД проявляє стійкість до великих обсягів даних та забезпечує високий рівень відмовостійкості, що робить її відмінним вибором для проектів з високою доступністю та критичних застосунків.

Крім того, PostgreSQL — це відкрита система з відкритим вихідним кодом, що означає, що спільнота веде активну діяльність і функції швидко розвиваються. Це надає користувачам доступ до найновіших функцій, розширену підтримку та підтримку спільноти.

Узагальнюючи, вибір PostgreSQL для реалізації бази даних обумовлений його потужністю, стабільністю та широким спектром можливостей, а також активною спільнотою, що гарантує ефективну підтримку та можливості розвитку.

## Створення бази даних

Після всіх етапів проектування можна остаточно сформулювати скрипти, що створять потрібні об’єкти. Далі наведено команду, що створить базу даних з ім’ям “CarService”.

CREATE DATABASE CarService;

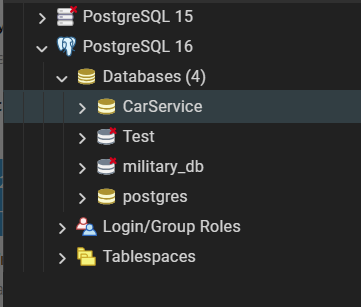


Рисунок 6.1 – список доступних баз даних в СУБД після виконання команди

Для створення всіх потрібних таблиць, виконаємо скрипт, що наведений в додатку А.

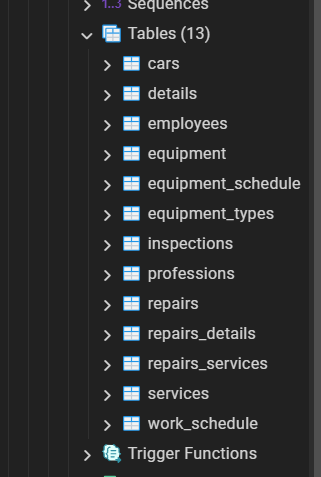


Рисунок 6.2 – список створених таблиць після виконання команди

Для виконання всіх виявлених обмежень та бізнес-правил виконаємо, скрипт, що наведений в додатку Б.

Далі згенеруємо схему даних засобами СУБД.

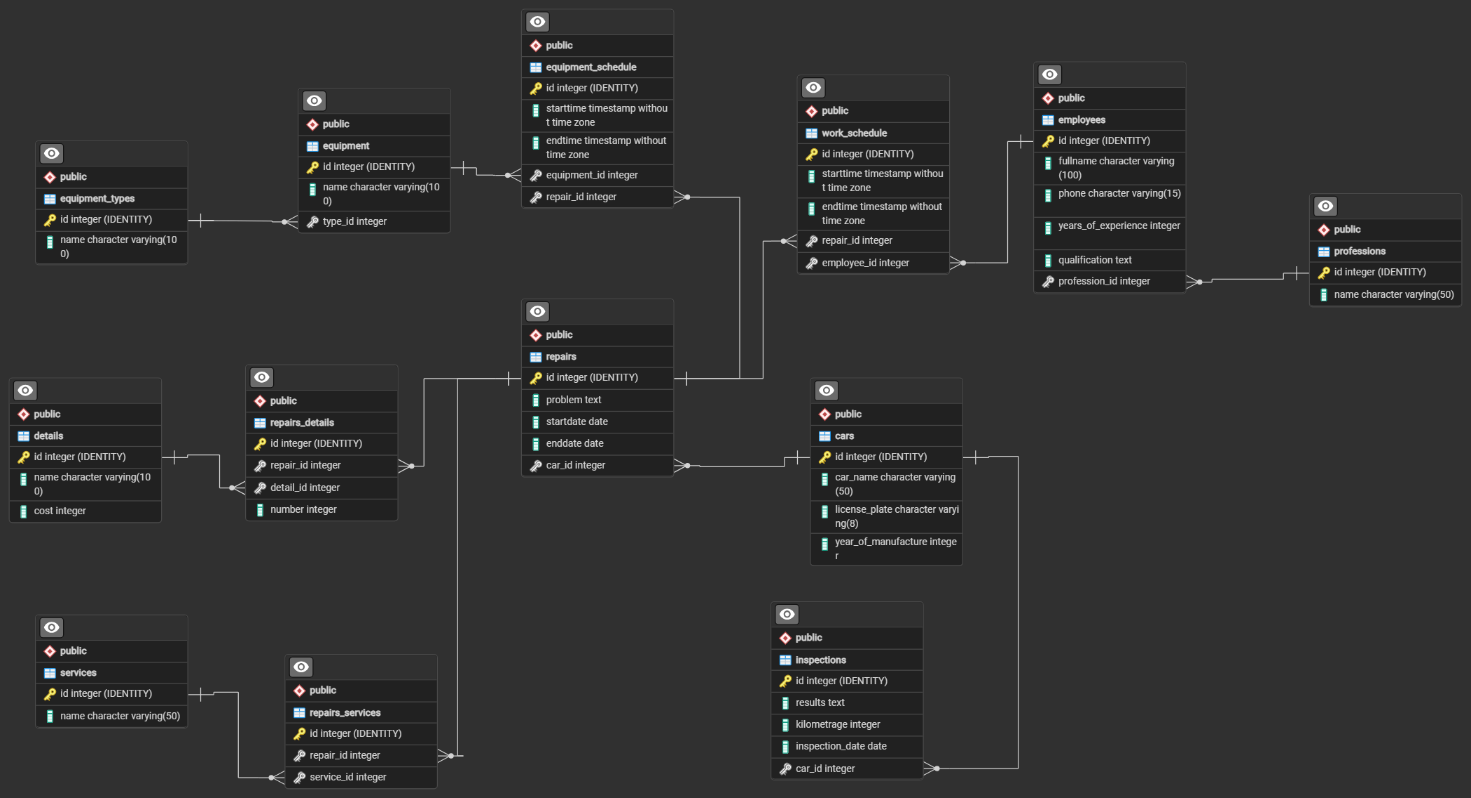


Рисунок 6.3 – Схема даних згенерована СУБД PostgreSQL

## Наповнення бази даних

Для подальшого тестування запитів та збережених процедур потрібно імпортувати дані у базу. Щоб це зробити, виконаємо скрипт, що наведений у додатку В. Далі будуть наведені вибірки даних після імпорту.

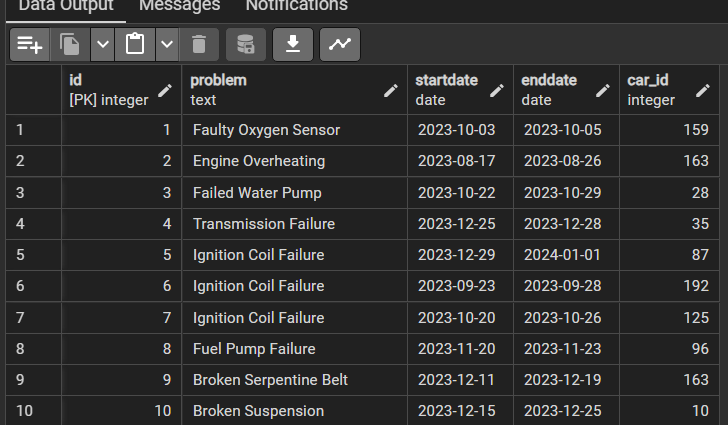


Рисунок 6.4 – Вибірка даних з таблиці “repairs”



Рисунок 6.5 – Вибірка даних з таблиці “cars”

## Створення користувачів

Для реалізації багатокористувацької моделі доступу, що була описана в описі предметного середовища, виконаємо скрипт з додатка Г.

Результатом виконання скрипта є створені користувачі:

1. Користувач аналітик **analyst\_1** з паролем 1111
2. Користувач менеджер **repair\_manager\_1** з паролем 2222
3. Користувач інспектор **inspector\_1** з паролем 4444
4. Користувач планувальник **planner\_1** з паролем 5555

Далі неведений приклад роботи з базою даних під користувачем **planner\_1**.

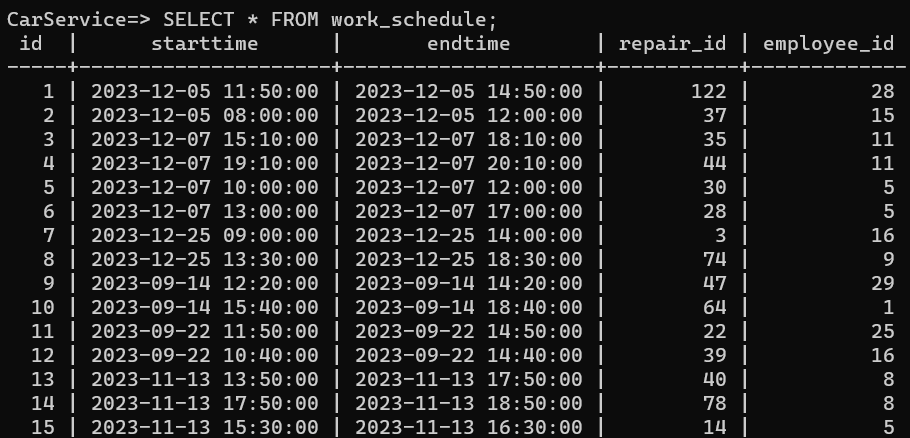


Рисунок 6.6 – Вибірка даних з таблиці, на яку користувач має дозвіл

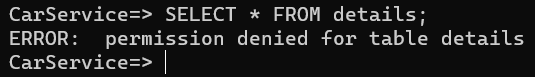


Рисунок 6.7 – Спроба вибірки із таблиці, на яку користувач не має дозвіл

# Робота з Базою даних

## Генератори

В розробленій базі даних, генератори використовуються для всіх табличок. Вони використовуються для створення унікального ідентифікатора, який є первинним ключом таблички. Далі наведений текст скрипта, що задає поле з генератором:

id int GENERATED ALWAYS AS IDENTITY

У всіх табличках поле для ідентифікатора називається “id”. Тому для інших табличок, генератор створюється аналогічно.

## Збережені процедури та функції

Відповідно до поставлених вимог та аналізу предметного середовища було розроблено наступні 12 функцій.

* + 1. Функція ActualCarKilometrage

Данна функція приймає як параметр ідентифікатор автомобіля та повертає поточний пробіг автомобіля. Результат виконання наведений на рисунку 7.1.

CREATE OR REPLACE FUNCTION ActualCarKilometrage(carID IN integer)

RETURNS int

AS $$

DECLARE

actualKilometrage int;

BEGIN

SELECT kilometrage

FROM inspections WHERE car\_id = carID

ORDER BY inspection\_date DESC LIMIT 1

INTO actualKilometrage;

RETURN actualKilometrage;

END;

$$ LANGUAGE plpgsql;

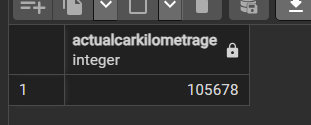


Рисунок 7.1 – Результат виконання ActualCarKilometrage для авто з ідентифікатором 15

* + 1. Функція GetLastCarInspectionDate

Функція GetLastCarInspectionDate повертає дату останнього технічного огляду автомобіля, ідентифікатор автомобіля передається як параметр. Результат виконання наведений на рисунку 7.2.

CREATE OR REPLACE FUNCTION GetLastCarInspectionDate(carID IN integer)

RETURNS date

AS $$

DECLARE

lastInspectionDate date;

BEGIN

SELECT inspection\_date FROM inspections

WHERE car\_id = carID ORDER BY inspection\_date DESC

LIMIT 1 INTO lastInspectionDate;

RETURN lastInspectionDate;

END;

$$ LANGUAGE plpgsql;

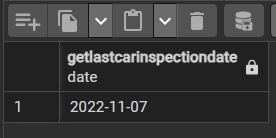


Рисунок 7.2 - Результат виконання GetLastCarInspectionDate для авто з ідентифікатором 12

* + 1. Функція GetCarsThatNeedInspection

Розроблена функція повертає набір автомобілів, які не проходили технічний огляд протягом заданого проміжку часу. Функція може бути корисна менеджеру підрозділу, щоб отримати інформацію про автомобілі, які потребують огляду. Результат виконання наведений на рисунку 7.3.

CREATE OR REPLACE FUNCTION GetCarsThatNeedInspection(timeSpan IN interval)

RETURNS TABLE(

car\_name varchar(50),

license\_plate varchar(8),

time\_since\_last\_inspection interval

)

AS $$

DECLARE

carRecord record;

last\_inspection\_date timestamp;

BEGIN

for carRecord in (SELECT id,cars.car\_name, cars.license\_plate FROM cars) LOOP

last\_inspection\_date := GetLastCarInspectionDate(carRecord.id);

time\_since\_last\_inspection := age(CURRENT\_DATE, last\_inspection\_date);

if time\_since\_last\_inspection >= timeSpan THEN

car\_name := carRecord.car\_name;

license\_plate := carRecord.license\_plate;

RETURN NEXT;

END IF;

END LOOP;

END;

$$ LANGUAGE plpgsql;

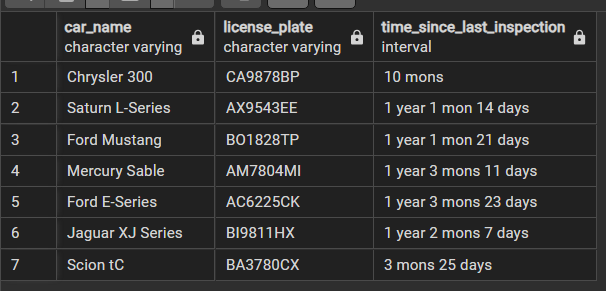


Рисунок 7.3 – Результат виконання ActualCarKilometrage

На рисунку 7.3 виведені всі автомобілі, що не мали технічного огляду більше ніж трьох місяців.

* + 1. Функція GetTimeForEquipmentReservation

Дана функція повертає вільні інтервали часу в які обладнання не використовується. Вона приймає як параметри дату, для якої обраховується час, час початку робочого дня, час кінця робочого дня та ідентифікатор обладнання. Результат виконання наведений на рисунку 7.4.

CREATE OR REPLACE FUNCTION GetTimeForEquipmentReservation

(

workDayStart time,

workDayEnd time,

reservationDate date,

equipmentID int

)

RETURNS TABLE(

startTime time,

endTime time

)

AS $$

DECLARE

reservation record;

previousTime time;

BEGIN

previousTime := workDayStart;

for reservation in (

SELECT CAST(equipment\_schedule.startTime AS time), CAST(equipment\_schedule.endTime AS time)

FROM equipment\_schedule WHERE CAST(equipment\_schedule.startTime AS date) = reservationDate

AND equipment\_id = equipmentID) LOOP

if reservation.startTime::time != previousTime THEN

startTime := previousTime;

endTime := reservation.startTime;

RETURN NEXT;

END IF;

previousTime := reservation.endTime;

END LOOP;

if workDayEnd != previousTime THEN

startTime := previousTime;

endTime := workDayEnd;

RETURN NEXT;

END IF;

END;

$$ LANGUAGE plpgsql;

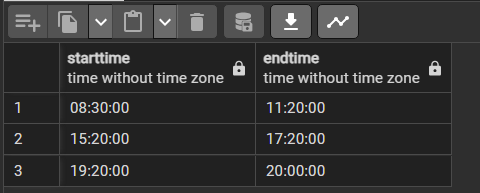


Рисунок 7.4 - Результат виконання GetTimeForEquipmentReservation

На рисунку 7.4 зображені час, протягом якого обладнання з ідентифікатором 5 вільне на 3 вересня 2023 року, якщо робочий день від 8:30 до 20:00.

* + 1. Функція GetEmployeeWorkHours

Дана функція обчислює скільки годин пропрацював працівник в заданий термін. В якості параметрів вона приймає ідентифікатор працівника, та період за який рахуються години. Результат виконання наведений на рисунку 7.6.

CREATE OR REPLACE FUNCTION GetEmployeeWorkHours

(

fromDate date,

toDate date,

employeeID int

)

RETURNS int

AS $$

DECLARE

workhours int;

BEGIN

SELECT EXTRACT(hour FROM SUM(endTime-startTime)) as hours FROM employees

JOIN work\_schedule ON work\_schedule.employee\_id = employees.id

WHERE CAST(startTime AS date) BETWEEN fromDate AND toDate

AND employee\_id = employeeID

INTO workhours;

RETURN workhours;

END;

$$ LANGUAGE plpgsql;

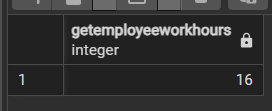


Рисунок 7.6 – Результат виконання GetEmployeeWorkHours

На рисунку 7.6 виведено скільки годин пропрацював працівник з ідентифікатором 15 з 1 по 22 грудня.

* + 1. Функція CalculateEmpolyeesWorkLoad

Функція CalculateEmpolyeesWorkLoad обчислює завантаженість робітників у відсотках. Як параметри вона приймає час початку та кінця робочого дня та часовий інтервал для якого робиться обрахунок. Результат виконання наведений на рисунку 7.7.

CREATE OR REPLACE FUNCTION CalculateEmpolyeesWorkLoad

(

workDayStart time,

workDayEnd time,

fromDate date,

toDate date

)

RETURNS TABLE(

fullname varchar(100),

workload varchar(5)

)

AS $$

DECLARE

scheduleRecord record;

maxHours decimal;

BEGIN

maxHours:= EXTRACT(hour FROM (toDate - fromDate) \* (workDayEnd - workDayStart))/2;

for scheduleRecord in (SELECT employees.fullname, SUM(endTime-startTime) as hours FROM employees JOIN work\_schedule ON work\_schedule.employee\_id = employees.id

WHERE CAST(startTime AS date) BETWEEN fromDate AND toDate

GROUP BY employees.fullname

) LOOP

fullname := scheduleRecord.fullname;

workload := CAST(ROUND((EXTRACT(hour FROM scheduleRecord.hours) / maxHours)\*100, 1) AS varchar(5)) || '%';

RETURN NEXT;

END LOOP;

END;

$$ LANGUAGE plpgsql;

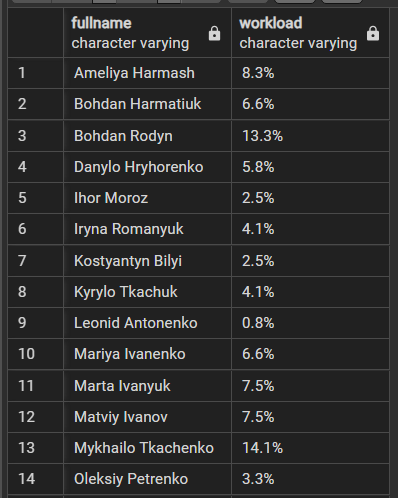


Рисунок 7.7 – Результат виконання CalculateEmpolyeesWorkLoad

На рисунку 7.7 зображена завантаженість робітників з 1 по 22 грудня, якщо робочий день з 8:30 до 20:00.

* + 1. Функція CountCarsByMake

Дана функція рахує всі автомобілі вказаної марки. Результат виконання наведений на рисунку 7.8.

CREATE OR REPLACE FUNCTION CountCarsByMake(brand IN varchar(25))

RETURNS int

AS $$

DECLARE

resultCount int;

BEGIN

SELECT COUNT(\*) FROM (SELECT car\_name FROM cars

WHERE car\_name like brand||'%') INTO resultCount;

RETURN resultCount;

END;

$$ LANGUAGE plpgsql;

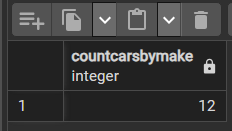


Рисунок 7.8 – Результат виконання CountCarsByMake

На рисунку 7.8 наведена кількість автомобілів марки “ Фольксваген”

* + 1. Процедура AddSpecificDetailToRepair

Розроблена функція додає до списку використаних деталей певного ремонту деталь, якої немає в базі. Як аргументи функція приймає назву деталі, її ціну, кількість деталей та ідентифікатор ремонту.

Щоб переконатись, що процедура працює, виконаєм виконаємо наступну команду:

CALL AddSpecificDetailToRepair(8,'Dynamic TurboBoost Module', 50000,1);

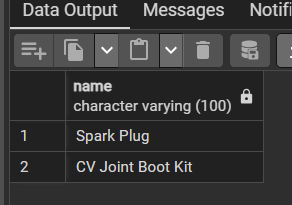


Рисунок 7.9 – Список замінених деталей до виконання процедури

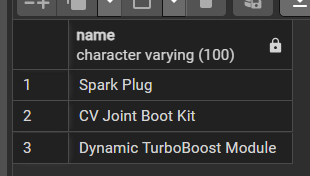


Рисунок 7.10 – Список замінених деталей після виконання процедури

* + 1. Функція GetTimeForEmployeeReservation

Розроблена функція повертає вільні інтервали часу в які працівник не працює над ремонтом. Вона приймає як параметри дату, для якої обраховується час, час початку робочого дня, час кінця робочого дня та ідентифікатор працівника. Результат виконання наведений на рисунку 7.5.

CREATE OR REPLACE FUNCTION GetTimeForEmployeeReservation

(

workDayStart time,

workDayEnd time,

reservationDate date,

employeeID int

)

RETURNS TABLE(

startTime time,

endTime time

)

AS $$

DECLARE

reservation record;

previousTime time;

BEGIN

previousTime := workDayStart;

for reservation in (

SELECT CAST(work\_schedule.startTime AS time), CAST(work\_schedule.endTime AS time)

FROM work\_schedule WHERE CAST(work\_schedule.startTime AS date) = reservationDate

AND employee\_id = employeeID) LOOP

if reservation.startTime::time != previousTime THEN

startTime := previousTime;

endTime := reservation.startTime;

RETURN NEXT;

END IF;

previousTime := reservation.endTime;

END LOOP;

if workDayEnd != previousTime THEN

startTime := previousTime;

endTime := workDayEnd;

RETURN NEXT;

END IF;

END;

$$ LANGUAGE plpgsql;

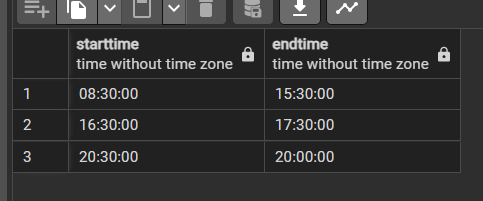


Рисунок 7.5 - Результат виконання GetTimeForEmployeeReservation

На рисунку 7.5 зображені час, протягом якого працівник з ідентифікатором 5 вільний на 13 листопада 2023 року, якщо робочий день від 8:30 до 20:00.

* + 1. Функція MakeReportAboutCar

Дана функція виводить всю історію ремонтів, обслуговування та замінених деталей вказаного автомобіля. Результат виконання наведений на рисунку 7.11.

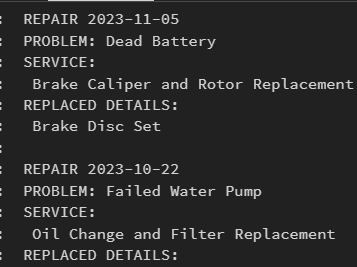


Рисунок 7.11 – Результат виконання MakeReportAboutCar

На рисунку 7.11 наведена повна історія ремонтів, замінених деталей та проведеного обслуговування для автомобіля з ідентифікатором 28.

* + 1. Функція CanReservateEquipment

Дана функція перевіряє чи можна забронювати обладнання на вказаний час. Вона перевіряє чи не пересікається вказаний часовий проміжок з іншим бронюванням. Як параметри приймає ідентифікатор обладнання, час початку бронювання та час кінця. Функція використовується в роботі тригерів reservateEquipment та UpdateReservationEquipment, результат виконання функції наведений в пунктах 1.3.7 та 1.3.8 відповідно.

CREATE OR REPLACE FUNCTION CanReservateEquipment(newStartTime IN timestamp, newEndTime IN timestamp, equipmentID in integer)

RETURNS bool

AS $$

DECLARE

reservationDate date;

reservation record;

canReservate bool := true;

BEGIN

reservationDate := CAST(newStartTime AS date);

for reservation in (

SELECT equipment\_schedule.startTime, equipment\_schedule.endTime FROM equipment\_schedule

WHERE CAST(startTime AS date) = reservationDate AND equipment\_id = equipmentID

) LOOP

if (reservation.startTime, reservation.endTime) OVERLAPS (newStartTime, newEndTime) THEN

canReservate := false;

END IF;

END LOOP;

RETURN canReservate;

END;

$$ LANGUAGE plpgsql;

* + 1. Функція CanReservateEmployee

Дана функція перевіряє чи можна додати запис в розклад працівника на вказаний час. Вона перевіряє чи не пересікається вказаний часовий проміжок з іншими записами в його розкладі. Як параметри приймає ідентифікатор працівника, час початку роботи над ремонтом та час кінця. Функція використовується в роботі тригерів reservateEmployee та UpdateReservationEmployee , результат виконання функції наведений в пунктах 1.3.9 та 1.3.10 відповідно.

## Тригери

* + 1. Тригер checkRepair

Тригер спрацьовує перед доданням до таблиці “repairs” та відповідає за збереження цілісності даних в ній. Його призначення перевіряти виконуваність бізнес-правила “В одного автомобіля не може відбуватись два ремонти одночасно”. Результати спрацювання тригера наведені на рисунку 7.12.

CREATE OR REPLACE FUNCTION checkRepair()

RETURNS trigger

AS $$

DECLARE

carID integer;

BEGIN

carID := NEW.car\_id;

IF EXISTS( SELECT \* FROM repairs WHERE car\_id = carID AND endDate IS NULL) THEN

RETURN NULL;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER checkRepair

BEFORE INSERT ON repairs

FOR EACH ROW EXECUTE FUNCTION checkRepair();

Щоб перевірити роботу тригера, спробуємо зареєструвати ремонт для машини, яка вже ремонтується. Для цього виконаємо наступну команду:

INSERT INTO repairs(problem, startdate, car\_id) VALUES('Engine overheating', '2023-12-05', 113);

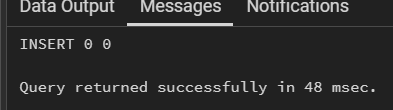


Рисунок 7.12 – Результат спрацювання тригера checkRepair

* + 1. Тригер deleteCar

Розроблений тригер спрацьовує перед видаленням з таблиці “cars”. Він виконує функцію каскадного видалення, при видалення автомобіля з бази спочатку видаляється вся інформація про його ремонти та огляди, а потім вже сама інформація про автомобіль.

CREATE OR REPLACE FUNCTION deleteCar()

RETURNS trigger

AS $$

BEGIN

DELETE FROM repairs WHERE car\_id = OLD.id;

DELETE FROM inspections WHERE car\_id = OLD.id;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER deleteCar

BEFORE DELETE On cars

FOR EACH ROW EXECUTE FUNCTION deleteCar();

Щоб перевірити, чи спрацьовує тригер, видалимо автомобіль з ідентифікатором 15.

DELETE FROM cars WHERE id = 15;

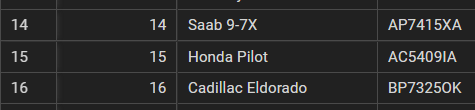


Рисунок 7.13 – таблиця “cars” до видалення

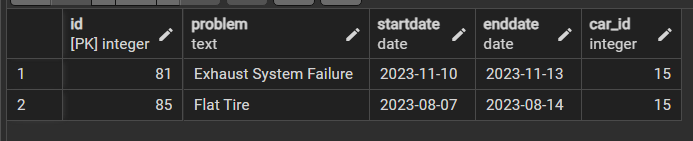


Рисунок 7.14 – Ремонти автомобіля до видалення

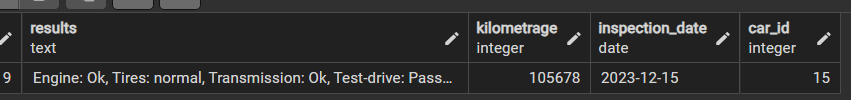


Рисунок 7.15 – Огляди автомобіля до видалення

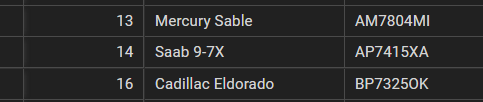


Рисунок 7.16 – таблиця “cars” після видалення

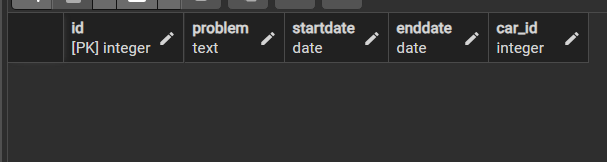


Рисунок 7.17 – Ремонти автомобіля після видалення

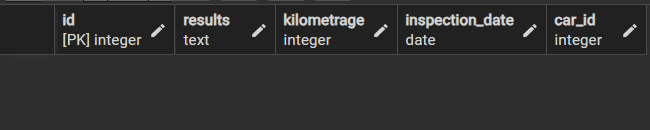


Рисунок 7.18 – Технічні огляди автомобіля після видалення

Як бачимо із рисунків 7.13 – 7.18 після видалення автомобіля інформація про його огляди та ремонти теж видалилась.

* + 1. Тригер checkLicensePlateBeforeAddition

Даний тригер виконуєтся перед додаванням в таблицю “cars” та контролює, щоб номерні знаки були коректного формату. Результат спрацювання тригера наведений на рисунку 7.19.

CREATE OR REPLACE FUNCTION checkLicensePlate() RETURNS trigger

AS $$

DECLARE

regionCode char(2);

series char(2);

numbers int;

BEGIN

regionCode := substring(new.license\_plate, 1,2);

series := substring(new.license\_plate, 7, 2);

BEGIN

numbers := substring(new.license\_plate, 3,4)::int;

EXCEPTION WHEN invalid\_text\_representation THEN RETURN NULL;

END;

IF regionCode IN ('AA', 'AB', 'AC', 'AE', 'AH', 'AI' ,'AM', 'AO', 'AP', 'АT', 'AX', 'BA', 'BB', 'BC', 'BE', 'BH', 'BI', 'BM', 'BO', 'BP', 'BT', 'BX',

'CA', 'CB', 'CE') THEN

IF series ~ '[A-Z]{2}' THEN RETURN NEW;

END IF;

END IF;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER checkLicensePlateBeforeAddition

BEFORE INSERT ON cars

FOR EACH ROW EXECUTE FUNCTION checkLicensePlate();

Спробуємо додати автомобіль з неправильними номерними знаками:

INSERT INTO cars(car\_name, license\_plate, year\_of\_manufacture)

VALUES ('Volkswagen Passat', 'GF54d5GD', 2009);

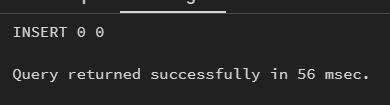


Рисунок 7.19 – Результат спрацювання тригера checkLicensePlateBeforeAddition

Як бачимо з рисунку 7.19 не вдалося додати автомобіль з некоректними номерними знаками.

* + 1. Тригер checkLicensePlateBeforeUpdating

Даний тригер аналогічний попередньому, але перевіряє умову коректності знаків перед оновленням запису. Результат спрацювання тригера наведений на рисунку 7.20.

CREATE OR REPLACE TRIGGER checkLicensePlateBeforeUpdating

BEFORE UPDATE ON cars

FOR EACH ROW EXECUTE FUNCTION checkLicensePlate();

Для того, щоб перевірити його роботу спробуємо оновити номерні знаки у автомобіля з ідентифікатором 15 на неправильні.

UPDATE cars SET license\_plate = '34АААА45' WHERE id = 15;

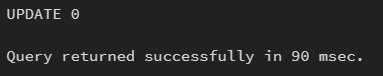


Рисунок 7.20 – Результат спрацювання тригера checkLicensePlateBeforeAddition

Як бачимо з рисунку 7.20 не вдалося оновити номерні знаки на некоректні.

* + 1. Тригер checkKilometrageBeforeUpdating

Тригер checkKilometrageBeforeUpdating спрацьовує перед оновленням пробігу автомобіля в таблиці “inspections”. Його основне призначення перевіряти, чи виконується бізнес-правило “Пробіг зафіксований на одному із оглядів авто повинен бути більший або рівний пробігу з попереднього огляду”

CREATE OR REPLACE FUNCTION checkKilometrage()

RETURNS trigger

AS $$

DECLARE

lowerBound int;

upperBound int;

BEGIN

SELECT kilometrage FROM inspections

WHERE car\_id = new.car\_id AND inspection\_date < new.inspection\_date

ORDER BY inspection\_date DESC LIMIT 1 INTO lowerBound;

SELECT kilometrage FROM inspections

WHERE car\_id = new.car\_id AND inspection\_date > new.inspection\_date

ORDER BY inspection\_date DESC LIMIT 1 INTO upperBound;

IF lowerBound IS NULL THEN

IF new.kilometrage <= upperBound THEN

RETURN new;

END IF;

ELSEIF upperBound IS NULL THEN

IF new.kilometrage >= lowerBound THEN

RETURN new;

END IF;

ELSE

IF new.kilometrage >= lowerBound AND new.kilometrage <= upperBound THEN

RETURN new;

END IF;

END IF;

RETURN null;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER checkKilometrageBeforeUpdating

BEFORE UPDATE ON inspections

FOR EACH ROW EXECUTE FUNCTION checkKilometrage();

Для перевірки тригер спробуємо оновити пробіг для одного із оглядів автомобіля з id 75. На рисунку 7.21 наведені всі огляди даного автомобіля перед оновленням.

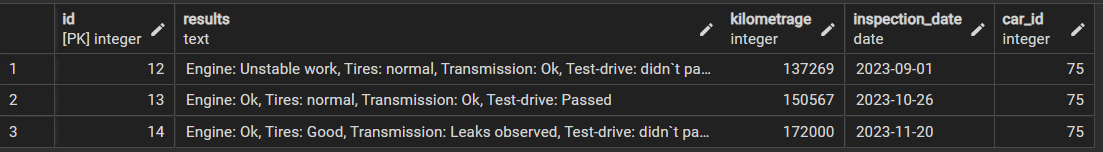


Рисунок 7.21 – Технічні огляди автомобіля з id 75

Для оновлення пробігу виконаємо наступну команду:

UPDATE inspections SET kilometrage = 200000 WHERE id = 13;

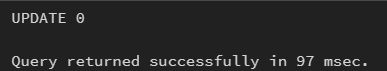


Рисунок 7.22 – Результат спрацювання тригеру checkKilometrageBeforeUpdating

Як бачимо із рисунку 7.22 тригер спрацював і не дав оновити пробіг на більший ніж у наступному огляді.

* + 1. Тригер checkKilometrageBeforeInserting

Даний тригер аналогічний попередньому, але спрацьовує перед додаванням інформації про новий технічний огляд.

CREATE OR REPLACE TRIGGER checkKilometrageBeforeInserting

BEFORE INSERT ON inspections

FOR EACH ROW EXECUTE FUNCTION checkKilometrage();

Спробуємо додати новий технічний огляд з пробігом 100000 для автомобіля з id 75.

INSERT INTO inspections(results, kilometrage, inspection\_date, car\_id)

VALUES('Test', 100000, '2023-12-15', 75);

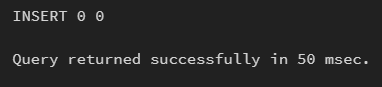


Рисунок 7.23 – Результат спрацювання тригера checkKilometrageBeforeInserting

Як бачимо із рисунку 7.23 тригер спрацював і не дав додати огляд з пробігом, що менший за попередній.

* + 1. Тригер TryReservateEquipment

Розроблений тригер спрацьовує перед додаванням нового запису до таблиці “Equipment\_schedule”. Його основне призначення – перевіряти, чи допустимий проміжок часу намагається додати користувач. Якщо час, на який користувач хоче забронювати обладнання, перетинається із іншими записами в розкладі, то такий запис не може бути доданий.

CREATE OR REPLACE FUNCTION СheckReservationTimeForEquipment ()

RETURNS trigger

AS $$

BEGIN

IF (new.startTime < new.endTime) AND CanReservateEquipment(new.startTime, new.endTime, new.equipment\_id) THEN

RETURN new;

END IF;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER TryReservateEquipment

BEFORE INSERT ON equipment\_schedule

FOR EACH ROW EXECUTE FUNCTION СheckReservateTimeForEquipment ();

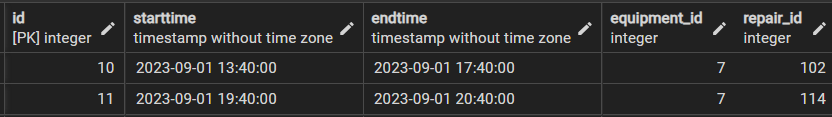


Рисунок 7.24 – Наявні бронювання для обладнання з id 7 на 1 вересня

Спробуємо додати бронювання для обладнання з id 7, що перетинається з раніше доданими бронюванням:

INSERT INTO equipment\_schedule(startTime, endTime, equipment\_id, repair\_id)

VALUES('2023-09-01 12:00', '2023-09-01 14:30', 7, 23);

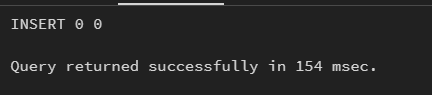


Рисунок 7.25 – Результат спрацювання тригера TryReservateEquipment

Як бачимо із рисунка 7.25, не вдалося додати бронювання з некоректним часом.

* + 1. Тригер UpdateReservationTimeForEquipment

Даний тригер аналогічний попередньому, але спрацьовує при оновлені часу бронювання обладнання.

CREATE OR REPLACE TRIGGER UpdateReservationTimeForEquipment

BEFORE UPDATE ON equipment\_schedule

FOR EACH ROW EXECUTE FUNCTION СheckReservationTimeForEquipment();

Для перевірки тригера спробуємо оновити одне з бронювань, таким чином, щоб воно перетнулося в часі з іншим.

UPDATE equipment\_schedule SET endTime = '2023-09-01 20:00' WHERE id = 10;

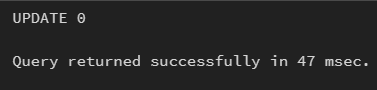


Рисунок 7.26 – Результат спрацювання тригера UpdateReservationTimeForEquipment

Як бачимо із рисунка 7.26, не вдалося оновити бронювання з некоректним часом.

* + 1. Тригер TryReservateEmployee

Даний тригер спрацьовує перед додаванням нового запису до таблиці “Work\_schedule”. Його основне призначення – перевіряти, чи допустимий проміжок часу намагається додати користувач. Якщо час, на який користувач хоче доручити ремонт працівнику, перетинається з іншими записами в його розкладу, то такий запис не може бути доданий.

CREATE OR REPLACE FUNCTION СheckReservationTimeForEmployee()

RETURNS trigger

AS $$

BEGIN

IF (new.startTime < new.endTime) AND CanReservateEmployee(new.startTime, new.endTime, new.employee\_id) THEN

RETURN new;

END IF;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER TryReservateEmployee

BEFORE INSERT ON work\_schedule

FOR EACH ROW EXECUTE FUNCTION СheckReservationTimeForEmployee();

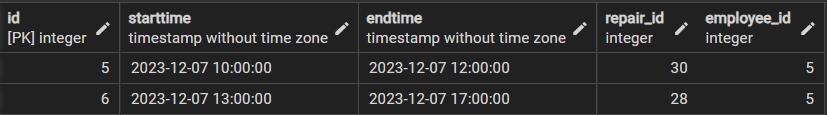


Рисунок 7.27 – Розклад роботи працівника з id 5 на 7 грудня

Щоб перевірити, чи спрацює тригер, додаймо для цього працівника запис в розклад, що перетинається в часі з іншими записами:

INSERT INTO work\_schedule(startTime, endTime, repair\_id, employee\_id)

VALUES('2023-12-07 15:00', '2023-12-07 18:00', 25, 5);

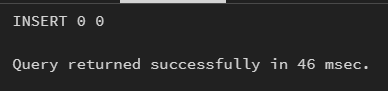


Рисунок 7.28 – Результат спрацювання тригера TryReservateEmployee

Як бачимо із рисунка 7.28, не вдалося додати запис із неправильним часом.

* + 1. Тригер UpdateReservationTimeForEmployee

Даний тригер аналогічний попередньому, але спрацьовує при оновленні запису в таблиці “work\_schedule”.

CREATE OR REPLACE TRIGGER UpdateReservationTimeForEmployee

BEFORE UPDATE ON work\_schedule

FOR EACH ROW EXECUTE FUNCTION СheckReservationTimeForEmployee();

Для перевірки спробуємо оновити час на такий, що перетинається з часом в інших записах:

UPDATE work\_schedule SET endTime = '2023-12-07 15:00' WHERE id = 5;

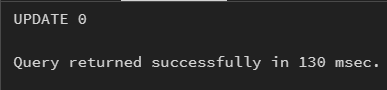


Рисунок 7.29 – Результат спрацювання тригера UpdateReservationTimeForEmployee

Як бачимо із рисунка 7.29, не вдалося оновити час запису на некоректний.

## Представлення

У цьому розділі будуть описані всі розроблені представлення. Вони є ключовим елементом структури бази даних. Представлення визначають зручний та оптимізований спосіб представлення даних для користувачів або додатків, забезпечуючи необхідний рівень абстракції та швидкий доступ до інформації. У цьому розділі детально розглядається функціональність кожного представлення, його призначення та вплив на роботу бази даних в цілому.

* + 1. Представлення usedDetailsInThisMonth

Це представлення виконує функцію звіту по використаним запчастинам за поточний місяць, а саме такі поля: назву деталі, її ціну та кількість. На рисунку 7.30 наведений результат вибірки даних з цього представлення.

CREATE OR REPLACE VIEW usedDetailsInThisMonth

AS SELECT name, cost, SUM(number) as number FROM repairs JOIN repairs\_details

ON repairs.id = repairs\_details.repair\_id

JOIN details ON details.id = repairs\_details.detail\_id

WHERE EXTRACT(month from startDate) = EXTRACT(month from CURRENT\_DATE)

GROUP BY name, cost;

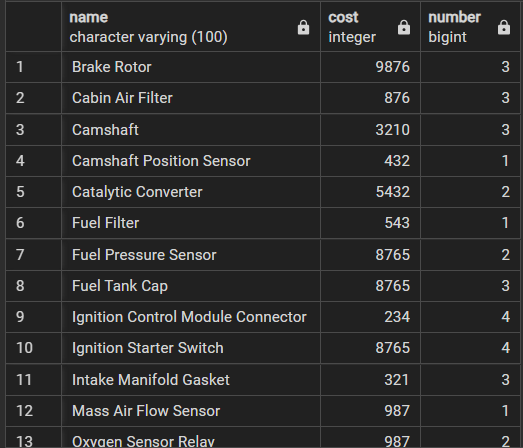


Рисунок 7.30 – Вибірка з представлення usedDetailsInThisMonth

* + 1. Представлення EquipmentOperationTimeThisMonth

Розроблене представлення відображає, скільки годин працювало обладнання за поточний місяць. Таким чином можна проаналізувати, які прилади використовуються найбільше та оптимізувати роботу. На рисунку 7.31 наведений результат вибірки з цього представлення.

CREATE OR REPLACE VIEW EquipmentOperationTimeThisMonth

AS SELECT name, SUM(EXTRACT(HOUR FROM (endTime - startTime))) as hours FROM equipment JOIN equipment\_schedule

ON equipment.id = equipment\_schedule.equipment\_id

WHERE EXTRACT(month from startTime) = EXTRACT(month from CURRENT\_DATE)

GROUP BY name

ORDER BY hours DESC;

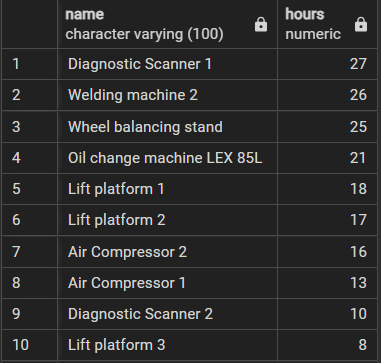


Рисунок 7.31 – Вибірка з представлення EquipmentOperationTimeThisMonth

* + 1. Представлення RepairCostsInThisMonth

Дане представлення є звітом за поточний місяць, який відображає інформацію про витрати на ремонт автомобілів. представлення надає корисний засіб для моніторингу та аналізу витрат на ремонт автомобілів протягом поточного місяця, що може бути важливим для управління та планування бюджету підприємства. На рисунку 7.32 наведений результат вибірки з цього представлення.

CREATE OR REPLACE VIEW RepairCostsInThisMonth

AS SELECT car\_name, license\_plate, SUM((cost\*number)) as totalCost FROM cars JOIN repairs

ON car\_id = cars.id

JOIN repairs\_details ON repairs.id = repairs\_details.repair\_id

JOIN details ON details.id = detail\_id

WHERE EXTRACT(month from startDate) = EXTRACT(month from CURRENT\_DATE)

GROUP BY car\_name, license\_plate

ORDER BY totalCost DESC;

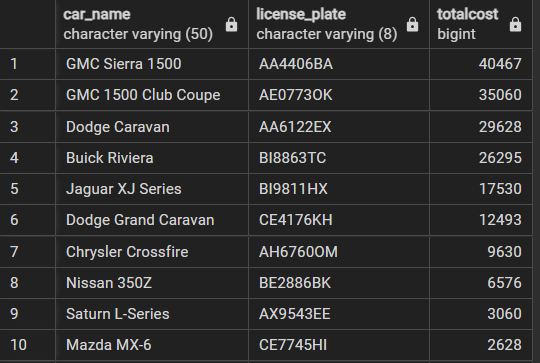


Рисунок 7.32 – Вибірка з представлення RepairCostsInThisMonth

## Запити

SQL запити – це базовий інструмент для маніпуляції з даними в СУБД. За їх допомогою можливо додавати, видаляти або оновлювати різну інформацію. У цьому розділі будуть представлені описи структури кожного запиту, використані оператори та функції, а також наведені приклади результатів виконання. Аналіз запитів зосереджений на розкритті їхнього функціонального призначення та визначенні ефективності в контексті роботи з базою даних.

* + 1. Запит, що виводить автомобілі, що найчастіше ремонтувались

Запит рахує всі ремонти автомобілів, визначає найбільшу кількість ремонтів та виводить автомобіля, в який кількість ремонтів дорівнює максимальній. На рисунку 7.33 Наведений результат роботи запита.

WITH numberOfRepairs AS(

SELECT car\_name, license\_plate, COUNT(\*) as num

FROM cars JOIN repairs ON cars.id =car\_id

GROUP BY car\_name, license\_plate

)

SELECT car\_name, license\_plate, num FROM numberOfRepairs WHERE num = (SELECT MAX(num)FROM numberOfRepairs);

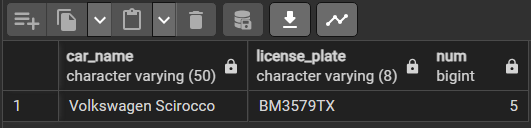


Рисунок 7.33 – Автомобілі, які найчастіше ремонтувались

* + 1. Запит, що виводить автомобілі, які ремонтуються в даний момент

Даний запит, виводить всі автомобілі, в яких є ремонт, у якому зазначена дата початку, але не зазначена дата кінця. Це означає, що автомобіль в даний момент ремонтується. На рисунку 7.34 наведений результат роботи запиту.

SELECT car\_name, license\_plate FROM cars, (SELECT car\_id FROM repairs WHERE endDate IS NULL)

WHERE cars.id = car\_id;

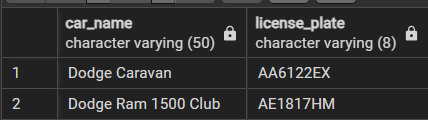


Рисунок 7.34 – Автомобілі, що в даний момент ремонтуються

* + 1. Запит, який виводить деталі, що були замінені в автомобілях

Цей запит використовує з’єднання таблиць “cars”, “repairs”, “repairs\_details”, “details” , щоб вивести таблицю, яка містить такі поля: назва авто, номерні знаки, назва деталі, дата заміни деталі. На рисунку 7.35 наведений результат виконання запиту.

SELECT car\_name, license\_plate, details.name, startDate as date FROM cars

JOIN repairs ON car\_id = cars.id

JOIN repairs\_details ON repair\_id = repairs.id

JOIN details ON detail\_id = details.id

ORDER BY license\_plate;

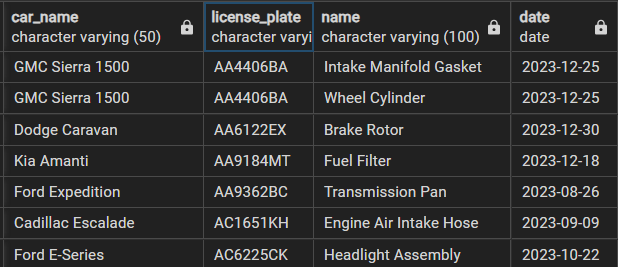


Рисунок 7.35 – Автомобілі та їх замінені деталі

* + 1. Запит, який виводить обслуговування, що було зроблене для автомобілів.

Цей запит використовує з’єднання таблиць “cars”, “repairs”, “repairs\_details”, “services” , щоб вивести таблицю з обслуговуванням, що було проведене для автомобілів. На рисунку 7.36 наведений результат роботи запиту.

SELECT car\_name, license\_plate, services.name, startDate as date FROM cars

JOIN repairs ON car\_id = cars.id

JOIN repairs\_services ON repair\_id = repairs.id

JOIN services ON service\_id = services.id

ORDER BY license\_plate;

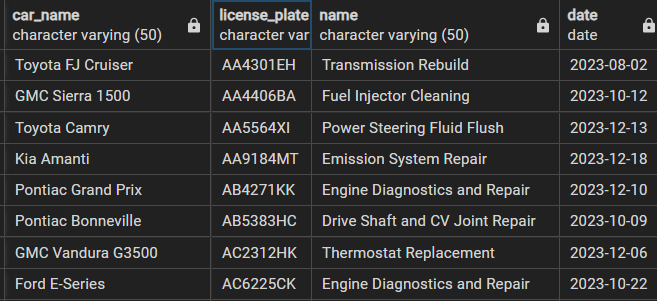


Рисунок 7.36 – Автомобілі та обслуговування, проведене для них

* + 1. Запит, що виводить план роботи працівників на сьогодні

Даний запит з’єднує таблиці “work\_schedule”, “employees”, “repairs”, “cars” та відбирає ті записи у який дата початку дорівнює поточній. Результат виконання наведений на рисунку 7.37.

SELECT fullname,car\_name, license\_plate, problem, startTime, endTime FROM work\_schedule

JOIN employees ON employee\_id = employees.id

JOIN repairs ON repair\_id = repairs.id

JOIN cars ON car\_id = cars.id

WHERE DATE(startTime) = CURRENT\_DATE

ORDER BY startTime;

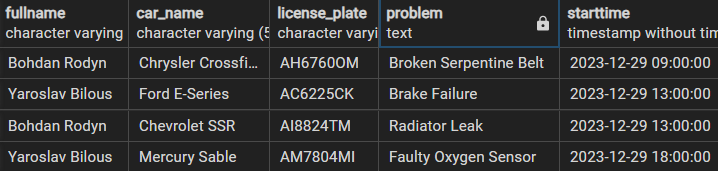


Рисунок 7.37 – План роботи працівників на 29 грудня

* + 1. Запит, що виводить бронювання обладнання на сьогодні

Даний запит з’єднує таблиці “equipment\_schedule”, “equipment”, “repairs”, “cars” та відбирає ті записи у який дата початку дорівнює поточній. Результат виконання наведений на рисунку 7.38.

SELECT name, car\_name, license\_plate, startTime, endTime FROM equipment\_schedule

JOIN equipment ON equipment\_id = equipment.id

JOIN repairs ON repair\_id = repairs.id

JOIN cars ON car\_id = cars.id

WHERE DATE(startTime) = CURRENT\_DATE

ORDER BY startTime;

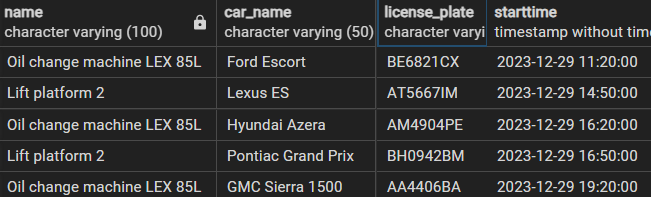


Рисунок 7.38 – Бронювання обладнання на 29 грудня

* + 1. Запит, що виводить автомобіль з найбільшим пробігом

Запит, знаходить технічний огляд, в якому був зафіксований найбільший пробіг та виводить назву автомобіля, номерні знаки та пробіг. Результат виконання запиту наведений на рисунку 7.39.

SELECT car\_name, license\_plate, kilometrage FROM cars JOIN inspections

ON car\_id = cars.id

WHERE kilometrage = (

SELECT MAX(kilometrage) FROM cars JOIN inspections

ON car\_id = cars.id);

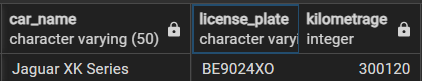


Рисунок 7.39 – Автомобіль з найбільшим пробігом

* + 1. Запит, що виводить працівників з їх спеціальностями

Запит з’єднує таблиці “employees” та “professions” та виводить результуючу таблицю. Результат виконання запиту наведений на рисунку 7.40.

SELECT fullname, professions.name FROM employees

JOIN professions ON profession\_id = professions.id;

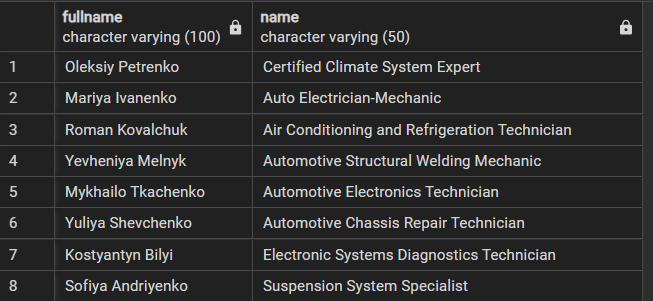


Рисунок 7.40 – Працівники та їх спеціальність

* + 1. Запит, що виводить працівників та ремонти, над якими вони працювали

Запит виводить назву автомобіля, який ремонтувався, опис несправності, ПІБ працівника та час, протягом якого працівник ремонтував автомобіль. Результат виконання запиту наведений на рисунку 7.41.

SELECT car\_name, problem, fullname, startTime, endTime FROM employees

JOIN work\_schedule ON employee\_id = employees.id

JOIN repairs ON repair\_id = repairs.id

JOIN cars ON car\_id = cars.id;

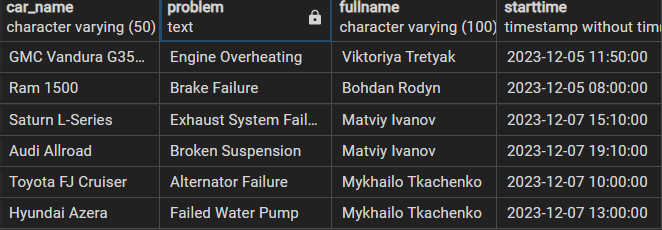


Рисунок 7.42 – Автомобілі та робітники, що їх ремонтували

* + 1. Запит, що виводить тип обладнання, яке найчастіше використовується.

Даний запит рахує кількість бронювань обладнання, визначає запис із максимальною кількістю бронювань та виводить його тип обладнання. Запит, може використовуватись для визначення пріоритету закупівлі нових приладів. Результат виконання запиту наведений на рисунку 7.42.

WITH numberOfEquipmentUsage AS(

SELECT equipment\_types.name, COUNT(\*) FROM equipment\_types

JOIN equipments ON type\_id = equipment\_types.id

JOIN equipment\_schedule ON equipment\_id = equipments.id

GROUP BY equipment\_types.name

)

SELECT name, count FROM numberOfEquipmentUsage WHERE count = (SELECT MAX(count) FROM numberOfEquipmentUsage);

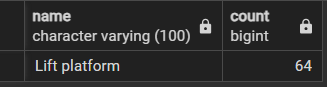


Рисунок 7.42 – Тип обладнання, що найчастіше використовувався

* + 1. Запит, що обчислює середню ціну деталей для ремонту

Запит, обчислює вартість деталей як добуток ціни на кількість та знаходить середня значення. Результат виконання запиту наведений на рисунку 7.43.

WITH detailsCost AS (

SELECT problem, SUM(cost\*number) as cost FROM repairs

JOIN repairs\_details ON repair\_id = repairs.id

JOIN details ON detail\_id = details.id

GROUP BY problem

)

SELECT AVG(cost) as AverageCost FROM detailsCost;

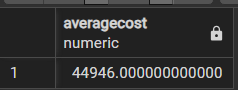


Рисунок 7.43 – Середня вартість деталей для ремонту

* + 1. Запит, що обчислює середні проміжок часу між технічними оглядами.

Запит обраховує різницю між датами двох оглядів та знаходить середня значення цих різниць. Запит може використовуватись, для оцінки дотримання норм проведення технічних оглядів. Результат виконання запиту наведений на рисунку 7.44.

SELECT AVG(interval) as averageInterval FROM (

SELECT age(f.inspection\_date, s.inspection\_date) as interval

FROM inspections f, inspections s WHERE f.car\_id = s.car\_id

AND f.inspection\_date > s.inspection\_date

);

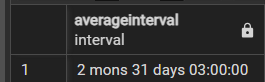


Рисунок 7.44 – Середній час між технічними оглядами

* + 1. Запит, що повертає всі технічні огляди, автомобіля, який має найбільший пробіг.

Даний запит спочатку знаходить автомобіль з найбільшим пробігом, а потім вибирає всі його технічні огляди. Результат виконання запиту наведений на рисунку 7.45.

SELECT car\_name, results, kilometrage, inspection\_date

FROM inspections JOIN cars ON car\_id = cars.id

WHERE car\_id =(

SELECT car\_id FROM inspections

WHERE kilometrage = (

SELECT MAX(kilometrage) FROM inspections

)

)

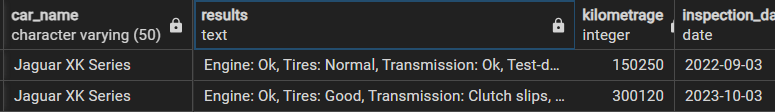


Рисунок 7.45 – Технічні огляди авто з найбільшим пробігом

* + 1. Запит, що виводить робітника, який працював найбільше годин в поточному місяці.

Запит спочатку обчислює, скільки годин працювали робітники, далі сортує їх за спаданням та відбирає першого працівника. Даний запит можк використовуватись для видачі премій та підвищення вмотивованості працівників. Результат виконання наведений на рисунку 7.46.

SELECT fullname, EXTRACT(hour FROM SUM(endTime - startTime)) as hours FROM employees

JOIN work\_schedule ON employee\_id = employees.id

WHERE EXTRACT(month from startTime) = EXTRACT(month from CURRENT\_DATE)

GROUP BY fullname

ORDER BY hours DESC LIMIT 1;

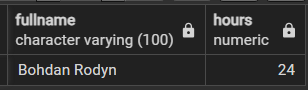


Рисунок 7.46 – Робітник, який відпрацював найбільшу кількість годин

* + 1. Запит, який виводить день тижня, в який найчастіше реєструвались нові ремонти.

Запит знаходить день тижня в який реєструвалися ремонти та знаходить день, що зустрічався найчастіше. Такий запит дозволить зрозуміти завантаженість підрозділу протягом тижня та ретельніше спланувати робочий день. Результат виконання наведений на рисунку 7.47.

WITH countOfRepairsByDay AS (

SELECT to\_char(startDate::date, 'Day') as dow, COUNT(to\_char(startDate::date, 'Day')) as count FROM repairs

GROUP BY dow

)

SELECT \* FROM countOfRepairsByDay WHERE count = (SELECT MAX(count) FROM countOfRepairsByDay)

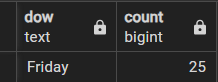


Рисунок 7.48 – Найбільш навантажений день тижня

* + 1. Запит, що обраховує середню тривалість ремноту

Даний запит обчислює середня значення різниці між датою кінця та датою початку. Він може бути застосований для оцінки ефективності підрозділу. Результат виконання наведений на рисунку 7.49.

SELECT car\_name, license\_plate, ROUND(AVG(EXTRACT(day FROM age(endDate, startDate))),3) as Duration FROM cars

JOIN repairs ON car\_id = cars.id

GROUP BY car\_name,license\_plate ORDER BY Duration



Рисунок 7.49 – Середні тривалості ремонтів автомобілів у днях

* + 1. Запит, який виводить авто, в яких був ремонт двигуна

Даний запит знаходить всі ремонти, в описі яких згадується слово “Двигун”. Результат виконання запиту наведений на рисунку 7.50.

SELECT car\_name, license\_plate, problem FROM cars

JOIN repairs ON car\_id = cars.id

WHERE problem ilike '%engine%';



Рисунок 7.50 – Автомобілі, в яких були проблеми з двигуном

* + 1. Запит, що виводить середній пробіг авто за роками їх випуску

Запит обчислює пробіг авто та знаходить середнє між авто з однаковим роком випуску. Результат виконання наведений на рисунку 7.51.

WITH kilometrages AS(

SELECT car\_id, MAX(kilometrage) as kilometrage FROM inspections

GROUP By car\_id

)

SELECT year\_of\_manufacture, ROUND(AVG(kilometrages.kilometrage),3) as kilometrage FROM cars

JOIN kilometrages ON car\_id = cars.id

GROUP BY year\_of\_manufacture

ORDER BY year\_of\_manufacture

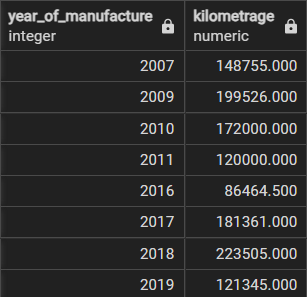


Рисунок 7.51 – Середній пробіг автомобілів за роками їх випуску

* + 1. Запит, що обчислює витрати на запчастини по місяцям

Запит обраховує вартість всіх деталей ремонту, сумує їх та групує по місяцям. Даний запит дає можливість зрозуміти, у який місяць були найбільші витрати на ремонт тощо. Результат виконання запиту наведений на рисунку 7.52.

SELECT EXTRACT(MONTH FROM startDate) AS monthNum, to\_char(startDate, 'Month'), SUM(cost\*number) FROM repairs

JOIN repairs\_details ON repair\_id = repairs.id

JOIN details ON detail\_id = details.id

GROUP BY to\_char(startDate, 'Month'), monthNum

ORDER BY monthNum



Рисунок 7.52 – Витрати на деталі по місяцям

* + 1. Запит, що виводить обслуговування, яке робили найчастіше у поточному місяці.

Запит відбирає всі ремонти поточного місяця, рахує в них обслуговування та виводить те, яке зустрічалося найбільшу кількість раз. Даний може дати розуміння, які послуги потрібно покращити в першу чергу. Результат виконання наведений на рисунку 7.53.

SELECT name, COUNT(\*) FROM repairs

JOIN repairs\_services ON repair\_id = repairs.id

JOIN services ON service\_id = services.id

WHERE EXTRACT(month from startDate) = EXTRACT(month from current\_date)

GROUP BY name ORDER BY count DESC

LIMIT 1;

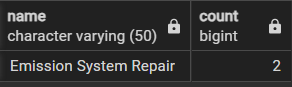


Рисунок 7.53 – Послуга, яка найчастіше надавалась у грудні

## Оптимізація

Оптимізація бази даних є ключовим етапом у забезпеченні ефективності та продуктивності системи. Загалом цей процес складається з багатьох аспектів. Перший шлях зробити базу ефективнішою – це нормалізація відношень. При цьому оптимізація досягається завдяки зменшенню зайвого дублювання інформації. Інший шлях – це оптимізувати саме запити. У цьому розділі будуть розглядатися саме оптимізація запитів.

Одним із потужним методів прискорення запитів є створення індексів. Швидкодія запиту досягається завдяки створенні додаткової структури даних, яка дозволяє робити швидкий пошук за заданими полями.

* + 1. Оптимізація першого запиту

Для оптимізації візьмемо наступний запит:

SELECT \* FROM equipment\_schedule

JOIN repairs ON repair\_id = repairs.id

JOIN cars ON car\_id = cars.id

WHERE startTime BETWEEN '2023-10-24 8:30' AND '2023-10-24 20:00';

Щоб побачити план виконання запиту використаємо конструкцію “EXPLAIN ANALYZE”. На рисунку 7.54 наведений план та час виконання запиту до оптимізації.

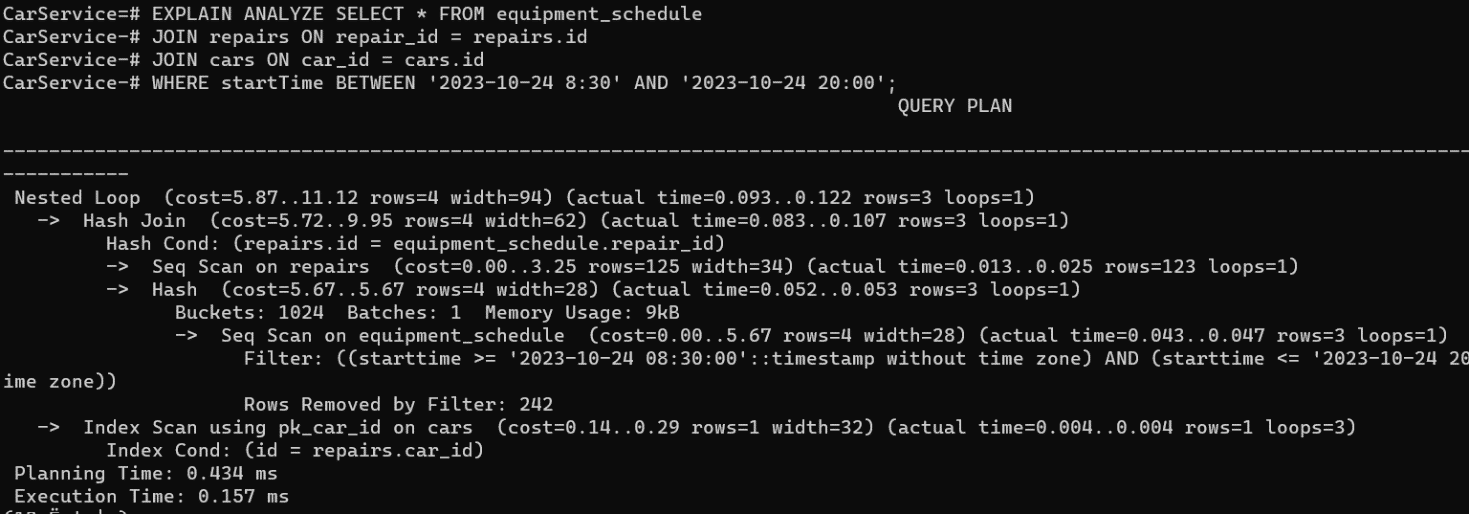


Рисунок 7.54 - План та час виконання першого запиту до оптимізації

Далі створимо індекс на таблицю “equipment\_schedule” за полем “startTime”.

CREATE INDEX startDate\_index ON equipment\_schedule(startTime);

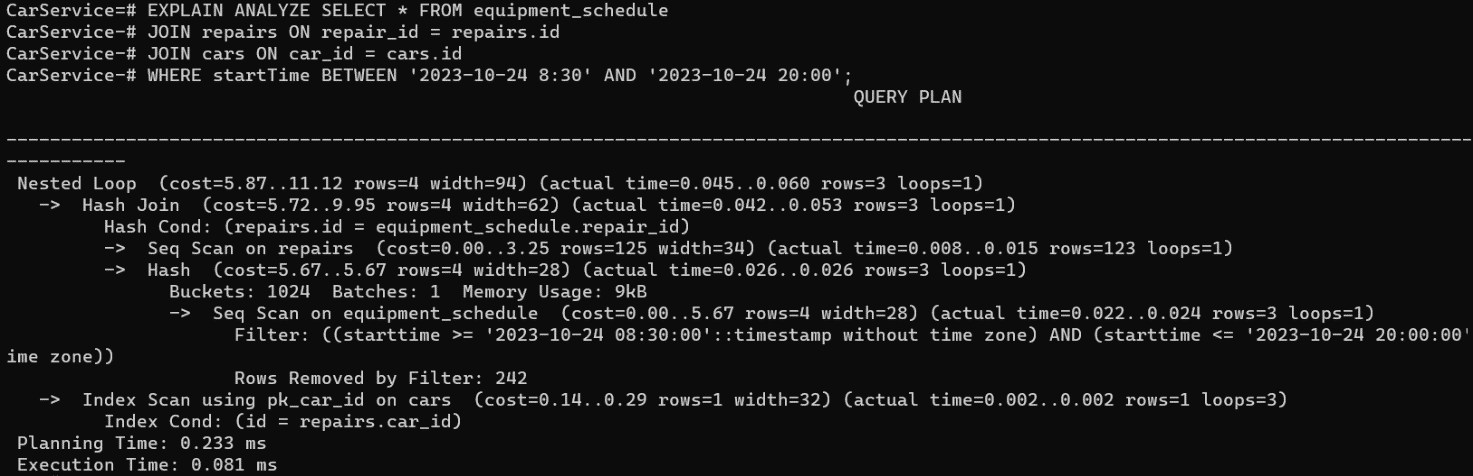


Рисунок 7.55 - План та час виконання першого запиту після оптимізації

Як бачимо із рисунків 7.54 та 7.55 час виконання запиту значно скоротився, отже оптимізація пройшла успішно.

* + 1. Оптимізація другого запиту

Для остаточної перевірки роботи індексів оптимізуємо наступний запит:

WITH kilometrages AS(

SELECT car\_id, MAX(kilometrage) as kilometrage FROM inspections

GROUP By car\_id

)

SELECT year\_of\_manufacture, ROUND(AVG(kilometrages.kilometrage),3) as kilometrage FROM cars

На рисунку 7.56 наведений план та час виконання запиту до створення індексу.

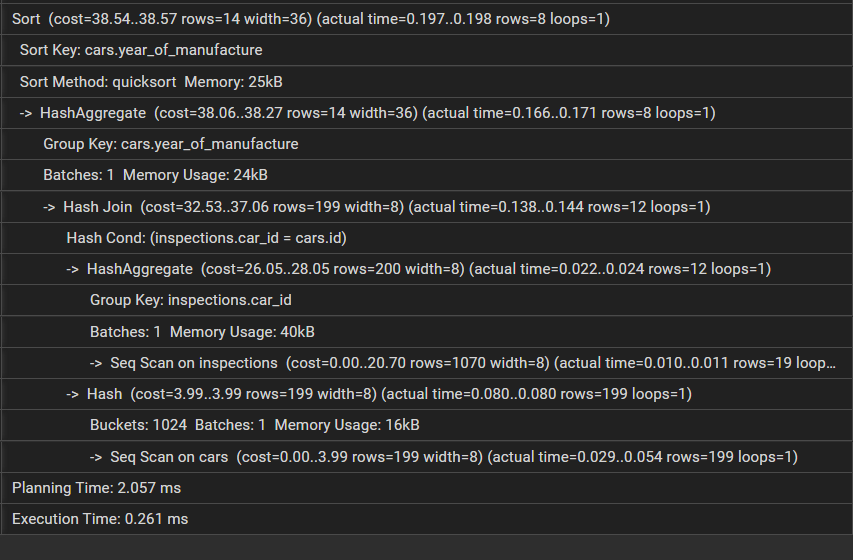


Рисунок 7.56 – План та час виконання другого запиту до оптимізації

Далі створимо індекс на таблицю “ inspections” за полем “kilometrage”:

CREATE INDEX kilometrage\_index ON inspections(kilometrage);



Рисунок 7.57 – План та час виконання другого запиту після оптимізації

З рисунків 7.56 та 7.57 можна побачити, що час виконання другого запиту теж став значно менший.

# ВИСНОВКИ

У ході виконання курсової роботи було проведено аналіз предметної області, аналіз вже існуючих програмних продуктів, були сформульовані вимоги до системи та розроблена ER-модель. Далі на основі дослідженої інформації була реалізована база даних та були написані всі необхідне програмне забезпечення для функціонування системи відповідно вимог.

Під час аналізу предметного середовища були виявлені основні компоненти системи. Також були описані основні бізнес-процеси підрозділу та бізнес-правила. Також за були виявлені основні типи користувачів, що будуть взаємодіяти з системою.

Розробка ER-моделі дала змогу виявити інформаційні об’єкти, їх атрибути та зв’язки. Цей процес значно спростив моделювання бази даних.

При реалізації бази були створені всі необхідні сутності, а також обмеження, які підтримували цілісність даних у базі. Були розроблені ролі та користувачі. Задля взаємодії із системою були написані збережені процедури, функції, представлення та запити.

Розроблена база даних розроблена відповідно до визначених вимог та оптимізує процеси реєстрації замовлень, ведення обліку технічних оглядів, планування роботи та обліку витрат на запчастини.

Отже, імплементація даної системи значно покращить управління підрозділом та надасть можливості для подальшого покращення обслуговування.

# Список використаної літератури

1. Роль служби технічного обслуговування у сучасних технологіях: функції та виклики. URL: <https://automoto.ua/uk/auto-news/sovremennaya-rol-sto-5884.html> (дата звернення: 30.12.2023).
2. PostgreSQL: documentation. URL: <https://www.postgresql.org/docs/> (date of access: 30.12.2023).
3. Chapter 43. pl/pgsql – SQL procedural language. *PostgreSQL Documentation*. URL: <https://www.postgresql.org/docs/current/plpgsql.html> (date of access: 30.12.2023).
4. Програма для автосервісу та СТО. URL: <https://remonline.ua/autoservice/> (дата звернення: 30.12.2023).
5. Софт для СТО carbook mobi | програма для автосервісу. URL: <https://carbook.mobi/> (дата звернення: 30.12.2023).
6. CRM система для автосервиса, автосалона, СТО - SalesDrive. URL: <https://salesdrive.com.ua/by-industry/auto/> (дата звернення: 30.12.2023).
7. Програма для СТО і автосервісу – CRM Appointer. *Appointer*. URL: <https://appointer.ua/programa-dlya-sto/> (дата звернення: 30.12.2023).

# Додаток А SQL скрипти створення бази даних

CREATE DATABASE CarService

DROP TABLE IF EXISTS employees CASCADE;

CREATE TABLE IF NOT EXISTS employees(

id int GENERATED ALWAYS AS IDENTITY,

fullname varchar(100) NOT NULL,

phone varchar(15) NOT NULL,

years\_Of\_Experience int NOT NULL DEFAULT 0,

qualification text,

CONSTRAINT pk\_employee\_id PRIMARY KEY(id)

);

DROP TABLE IF EXISTS professions CASCADE;

CREATE TABLE IF NOT EXISTS professions(

id int GENERATED ALWAYS AS IDENTITY,

name varchar(50) NOT NULL,

CONSTRAINT pk\_profession\_id PRIMARY KEY(id)

);

DROP TABLE IF EXISTS equipment\_types CASCADE;

CREATE TABLE IF NOT EXISTS equipment\_types(

id int GENERATED ALWAYS AS IDENTITY,

name varchar(100) NOT NULL,

CONSTRAINT pk\_equipment\_type\_id PRIMARY KEY(id)

);

DROP TABLE IF EXISTS equipment CASCADE;

CREATE TABLE IF NOT EXISTS equipment(

id int GENERATED ALWAYS AS IDENTITY,

name varchar(100) NOT NULL,

CONSTRAINT pk\_equipment\_id PRIMARY KEY(id)

);

DROP TABLE IF EXISTS details CASCADE;

CREATE TABLE IF NOT EXISTS details(

id int GENERATED ALWAYS AS IDENTITY,

name varchar(100) NOT NULL,

cost int NOT NULL,

CONSTRAINT pk\_detail\_id PRIMARY KEY(id)

);

DROP TABLE IF EXISTS services CASCADE;

CREATE TABLE IF NOT EXISTS services(

id int GENERATED ALWAYS AS IDENTITY,

name varchar(50) NOT NULL,

CONSTRAINT pk\_service\_id PRIMARY KEY(id)

);

DROP TABLE IF EXISTS inspections;

CREATE TABLE IF NOT EXISTS inspections(

id int GENERATED ALWAYS AS IDENTITY,

results text NOT NULL,

kilometrage int NOT NULL,

inspection\_date date NOT NULL DEFAULT CURRENT\_DATE,

CONSTRAINT pk\_inspection\_id PRIMARY KEY(id)

);

DROP TABLE IF EXISTS cars CASCADE;

CREATE TABLE IF NOT EXISTS cars(

id int GENERATED ALWAYS AS IDENTITY,

car\_name varchar(50) NOT NULL,

license\_plate varchar(8) NOT NULL,

year\_Of\_Manufacture int NOT NULL,

CONSTRAINT pk\_car\_id PRIMARY KEY(id)

);

DROP TABLE IF EXISTS repairs CASCADE;

CREATE TABLE IF NOT EXISTS repairs(

id int GENERATED ALWAYS AS IDENTITY,

problem text NOT NULL,

startDate date NOT NULL,

endDate date,

CONSTRAINT pk\_repair\_id PRIMARY KEY(id)

);

DROP TABLE IF EXISTS equipment\_schedule CASCADE;

CREATE TABLE IF NOT EXISTS equipment\_schedule(

id int GENERATED ALWAYS AS IDENTITY,

startTime timestamp NOT NULL,

endTime timestamp NOT NULL,

CONSTRAINT pk\_equipment\_schedule\_id PRIMARY KEY(id)

);

DROP TABLE IF EXISTS work\_schedule CASCADE;

CREATE TABLE IF NOT EXISTS work\_schedule(

id int GENERATED ALWAYS AS IDENTITY,

startTime timestamp NOT NULL,

endTime timestamp NOT NULL,

CONSTRAINT pk\_work\_schedule\_id PRIMARY KEY(id)

);

DROP TABLE IF EXISTS repairs\_details CASCADE;

CREATE TABLE IF NOT EXISTS repairs\_details(

id int GENERATED ALWAYS AS IDENTITY,

repair\_id int NOT NULL REFERENCES repairs(id) ON DELETE CASCADE,

detail\_id int NOT NULL REFERENCES details(id) ON DELETE CASCADE,

number int NOT NULL,

CONSTRAINT pk\_repairs\_details\_id PRIMARY KEY(id)

);

DROP TABLE IF EXISTS repairs\_services CASCADE;

CREATE TABLE IF NOT EXISTS repairs\_services(

id int GENERATED ALWAYS AS IDENTITY,

repair\_id int NOT NULL REFERENCES repairs(id) ON DELETE CASCADE,

service\_id int NOT NULL REFERENCES services(id) ON DELETE CASCADE,

CONSTRAINT pk\_repairs\_services\_id PRIMARY KEY(id)

);

ALTER TABLE equipment

ADD COLUMN type\_id int REFERENCES equipment\_types(id);

ALTER TABLE employees

ADD COLUMN profession\_id int NOT NULL REFERENCES professions(id);

ALTER TABLE inspections

ADD COLUMN car\_id int NOT NULL REFERENCES cars(id);

ALTER TABLE equipment\_schedule

ADD COLUMN equipment\_id int NOT NULL REFERENCES equipment(id);

ALTER TABLE equipment\_schedule

ADD COLUMN repair\_id int NOT NULL REFERENCES repairs(id) ON DELETE CASCADE;

ALTER TABLE work\_schedule

ADD COLUMN repair\_id int NOT NULL REFERENCES repairs(id) ON DELETE CASCADE;

ALTER TABLE work\_schedule

ADD COLUMN employee\_id int NOT NULL REFERENCES employees(id);

ALTER TABLE repairs

ADD COLUMN car\_id int NOT NULL REFERENCES cars(id);

# Додаток Б SQL скрипти логічних обмежень

ALTER TABLE inspections

ADD CONSTRAINT date\_check CHECK(inspection\_date <= CURRENT\_DATE);

ALTER TABLE cars

ADD CONSTRAINT year\_check CHECK(year\_of\_manufacture <= date\_part('year', CURRENT\_DATE));

ALTER TABLE cars

ADD CONSTRAINT unique\_license\_plate UNIQUE(license\_plate)

ALTER TABLE details

ADD CONSTRAINT cost\_check CHECK(cost > 0);

ALTER TABLE employees

ADD CONSTRAINT years\_of\_experience\_check CHECK(years\_of\_experience >= 0);

ALTER TABLE repairs

ADD CONSTRAINT date\_check CHECK(startDate <= endDate);

# Додаток В SQL скрипт заповнення бази даних

COPY professions(name) FROM 'C:\Users\hitec\OneDrive\Documents\Database\_CourseWork\import\_data\professions.csv' (

FORMAT CSV,

DELIMITER ';',

ENCODING 'UTF-8'

);

COPY employees(fullname, phone, years\_Of\_Experience,profession\_id) FROM 'C:\Users\hitec\OneDrive\Documents\Database\_CourseWork\import\_data\employees.csv' (

FORMAT CSV,

DELIMITER ',',

ENCODING 'UTF-8'

);

COPY equipment\_types(name) FROM 'C:\Users\hitec\OneDrive\Documents\Database\_CourseWork\import\_data\equipment\_types.csv' (

FORMAT CSV,

DELIMITER ',',

ENCODING 'UTF-8'

);

COPY equipment(name, type\_id) FROM 'C:\Users\hitec\OneDrive\Documents\Database\_CourseWork\import\_data\equipments.csv' (

FORMAT CSV,

DELIMITER ',',

ENCODING 'UTF-8'

);

COPY cars(car\_name, license\_plate, year\_Of\_Manufacture) FROM 'C:\Users\hitec\OneDrive\Documents\Database\_CourseWork\import\_data\cars.csv'(

FORMAT CSV,

DELIMITER ',',

ENCODING 'UTF-8'

);

COPY services(name) FROM 'C:\Users\hitec\OneDrive\Documents\Database\_CourseWork\import\_data\services.csv'(

FORMAT CSV,

DELIMITER ',',

ENCODING 'UTF-8'

);

COPY repairs(problem, startdate, enddate, car\_id) FROM 'C:\Users\hitec\OneDrive\Documents\Database\_CourseWork\import\_data\repairs.csv'(

FORMAT CSV,

DELIMITER ',',

ENCODING 'UTF-8'

);

COPY details(name, cost) FROM 'C:\Users\hitec\OneDrive\Documents\Database\_CourseWork\import\_data\details.csv'(

FORMAT CSV,

DELIMITER ',',

ENCODING 'UTF-8'

);

INSERT INTO inspections(results, kilometrage, inspection\_date, car\_id) VALUES

('Engine: Ok, Tires: Normal, Transmission: Ok, Test-drive: Passed', 150250, '2022-09-03', 5),

('Engine: Ok, Tires: Good, Transmission: Clutch slips, Test-drive: didn`t pass', 300120, '2023-10-03', 5),

('Engine: Unstable work, Tires: normal, Transmission: Ok, Test-drive: Passed', 163345, '2022-09-12', 45),

('Engine: Unstable work, Tires: All tires worn, Transmission: Good, Test-drive: didn`t pass', 180250, '2022-10-21', 45),

('Engine: Ok, Tires: All tires worn, Transmission: Ok, Test-drive: didn`t pass', 100856, '2022-10-09', 10),

('Engine: Ok, Tires: worn left rear tire, Transmission: Normal, Test-drive: didn`t pass', 121345, '2022-11-14', 10),

('Engine: Ok, Tires: normal, Transmission: Ok, Test-drive: Passed', 134950, '2022-09-09', 12),

('Engine: Ok, Tires: normal, Transmission: Ok, Test-drive: Passed', 162366, '2022-11-07', 12),

('Engine: Ok, Tires: normal, Transmission: Ok, Test-drive: Passed', 105678, '2023-12-15', 15),

('Engine: Ok, Tires: normal, Transmission: Ok, Test-drive: Passed', 182361, '2023-11-02', 7),

('Engine: Ok, Tires: normal, Transmission: Reverse gear does not work, Test-drive: didn`t pass', 200356, '2023-12-20', 7),

('Engine: Unstable work, Tires: normal, Transmission: Ok, Test-drive: didn`t pass', 137269, '2023-09-01', 75),

('Engine: Ok, Tires: normal, Transmission: Ok, Test-drive: Passed', 150567, '2023-10-26', 75),

('Engine: Ok, Tires: Good, Transmission: Leaks observed, Test-drive: didn`t pass', 172000, '2023-11-20', 75),

('Engine: Unstable work, Tires: Good, Transmission: Ok, Test-drive: didn`t pass', 199526, '2023-09-03', 56),

('Engine: high fuel consumption, Tires: Good, Transmission: Ok, Test-drive: didn`t pass', 77569, '2023-11-30', 123),

('Engine: Ok, Tires: worn right rear tire, Transmission: Ok, Test-drive: didn`t pass', 95360, '2023-11-25', 41),

('Engine: Ok, Tires: Good, Transmission: Ok, Test-drive: Passed', 117260, '2022-09-05', 26),

('Engine: Ok, Tires: Good, Transmission: second gear does not work, Test-drive: didn`t pass', 146890, '2022-09-17', 13),

('Engine: Ok, Tires: Good, Transmission: Gear shifting issues, Test-drive: didn`t pass', 120000, '2023-02-28', 2);

COPY repairs\_details(detail\_id, repair\_id, number) FROM 'C:\Users\hitec\OneDrive\Documents\Database\_CourseWork\import\_data\repairs\_details.csv'(

FORMAT CSV,

DELIMITER ',',

ENCODING 'UTF-8'

);

COPY repairs\_services(service\_id, repair\_id) FROM 'C:\Users\hitec\OneDrive\Documents\Database\_CourseWork\import\_data\repairs\_services.csv'(

FORMAT CSV,

DELIMITER ',',

ENCODING 'UTF-8'

);

COPY equipment\_schedule(starttime, endtime, equipment\_id, repair\_id) FROM 'C:\Users\hitec\OneDrive\Documents\Database\_CourseWork\import\_data\equipment\_schedule.csv'(

FORMAT CSV,

DELIMITER ',',

ENCODING 'UTF-8'

);

COPY work\_schedule(starttime, endtime, employee\_id, repair\_id) FROM 'C:\Users\hitec\OneDrive\Documents\Database\_CourseWork\import\_data\work\_schedule.csv'(

FORMAT CSV,

DELIMITER ',',

ENCODING 'UTF-8'

);

# Додаток Г SQL скрипти створення ролей та користувачів

CREATE ROLE analyst;

GRANT SELECT ON ALL TABLES IN SCHEMA public TO analyst;

CREATE ROLE repair\_manager;

GRANT ALL ON repairs TO repair\_manager;

GRANT SELECT ON cars TO repair\_manager;

CREATE ROLE inspector;

GRANT ALL ON inspections TO inspector;

GRANT SELECT ON cars TO inspector;

CREATE ROLE planner;

GRANT ALL ON work\_schedule, equipment\_schedule TO planner;

GRANT SELECT ON equipment, employees, repairs TO planner;

CREATE USER analyst\_1 WITH PASSWORD '1111';

GRANT analyst TO analyst\_1;

CREATE USER repair\_manager\_1 WITH PASSWORD '2222';

GRANT repair\_manager TO repair\_manager\_1;

CREATE USER inspector\_1 WITH PASSWORD '4444';

GRANT inspector TO inspector\_1;

CREATE USER planner\_1 WITH PASSWORD '5555';

GRANT planner TO planner\_1;