

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)

**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 ІІ-22 Семестр 3

**З А В Д А Н Н Я  
НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Панасюку Олександрю Юрійовичу

(прізвище, ім'я, по батькові)

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

керівник роботи старший викладач Марченко Олена Іванівна

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

2. Строк подання студентом роботи 30.12.2023

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)

6. Дата видачі завдання 31.11.2023

## 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**

\_\_\_\_\_  
(підпис)

**Панасюк О.Ю.**

\_\_\_\_\_  
(Прізвище та ініціали)

**Керівник роботи**

\_\_\_\_\_  
(підпис)

**Марченко О.І.**

\_\_\_\_\_  
(Прізвище та ініціали)

## ЗМІСТ

ВСТУП.....	5
1   Опис предметного середовища.....	6
1.1.   Бізнес-процеси .....	6
1.2.   Обмеження.....	8
1.3.   Користувачі .....	9
1.4.   Вхідні дані .....	9
1.5.   Вихідні дані .....	9
2   Аналіз існуючих програмних продуктів.....	10
3   Постановка Завдання .....	11
4   Опис ER-моделі .....	12
4.1.   Опис сутностей .....	12
4.2.   Опис атрибутів сутностей.....	13
4.3.   Опис зв'язків .....	14
4.4.   ER-діаграма .....	15
5   Реляційна модель.....	16
6   Створення бази даних .....	23
6.1.   Вибір СУБД.....	23
6.2.   Створення бази даних .....	23
6.3.   Наповнення бази даних .....	25
6.4.   Створення користувачів.....	26
7   Робота з Базою даних.....	28
7.1.   Генератори.....	28
7.2.   Збережені процедури та функції.....	28
7.3.   Тригери .....	40

7.4.	Представлення.....	50
7.5.	Запити .....	53
7.6.	Оптимізація .....	64
	ВИСНОВКИ.....	68
	Список використаної літератури .....	69
	Додаток А SQL скрипти створення бази даних .....	70
	Додаток Б SQL скрипти логічних обмежень .....	75
	Додаток В SQL скрипт заповнення бази даних .....	76
	Додаток Г SQL скрипти створення ролей та користувачів.....	80

## 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.

## 1 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

### 1.1. 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 менеджерів 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.

## 1.2. Обмеження

Далі наведені обмеження властивостей компонент, які логічно впливають 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

### 1.3. 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.

### 1.4. 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.

### 1.5. 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.

## 2 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.

### 3 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 декомпонувати 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.

## 4 ОПИС ER-МОДЕЛІ

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

### 4.1. 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.

#### 4.2. 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

#### 4.3. 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

#### 4.4. ER-діаграма

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

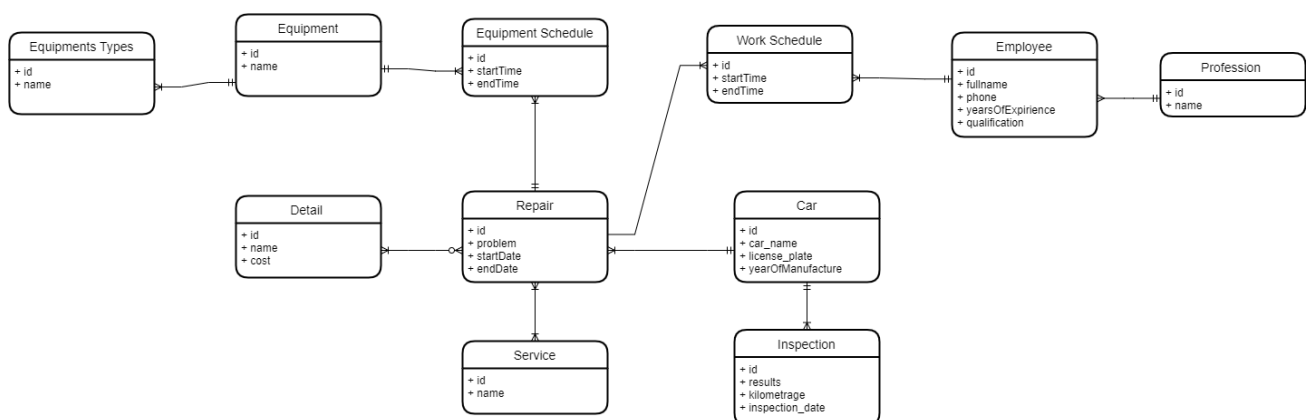


Рисунок 4.1 – ER- діаграма бази даних для підрозділу обслуговування  
автотранспорту



## 5 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	-	РК	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	-	РК	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	> startDate
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, > 0

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, > 0

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	-	РК	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, $\geq 0$
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	-	РК	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	-	РК	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	-	РК	Ідентифікатор розкладу	Identity, not null
startTime	timestamp	-	-	Час початку роботи працівника	Not null, Для одного обладнання часові проміжки не мають
endTime	timestamp	-	-	Час кінця роботи працівника	

					перетинатися
--	--	--	--	--	--------------

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.

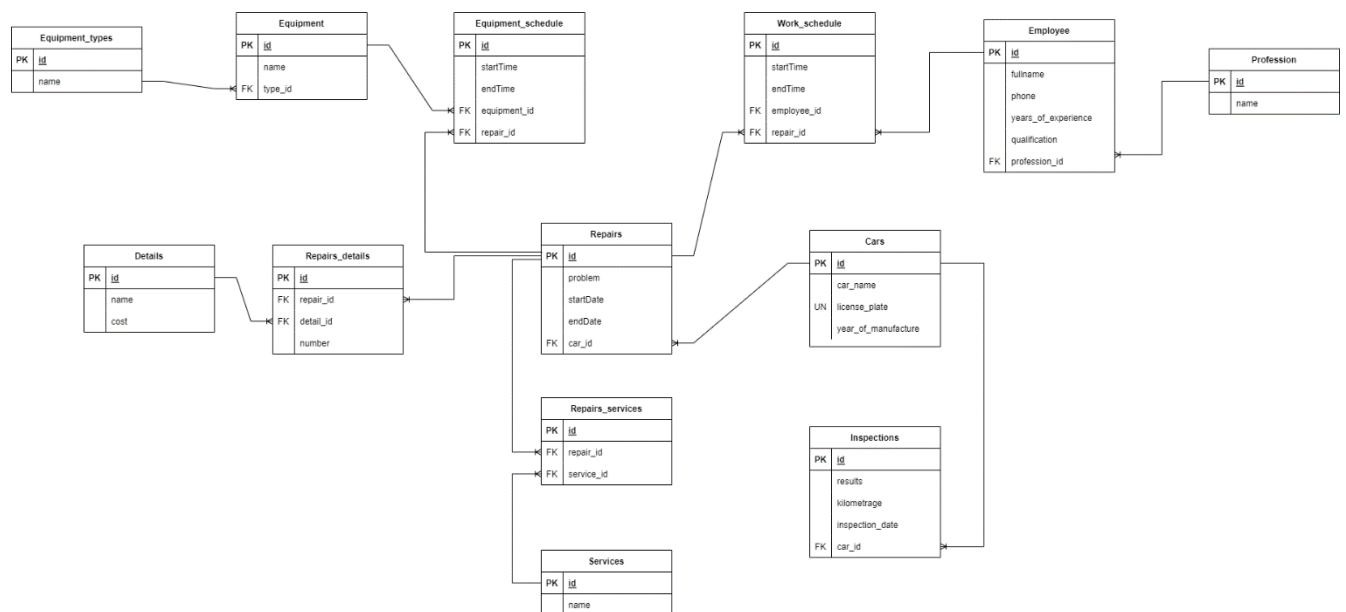


Рисунок 5.1 – Діаграма реляційної моделі для бази даних з підтримки діяльності підрозділу обслуговування автотранспорту

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.

## 6 СТВОРЕННЯ БАЗИ ДАНИХ

### 6.1. 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.

### 6.2. 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;
```



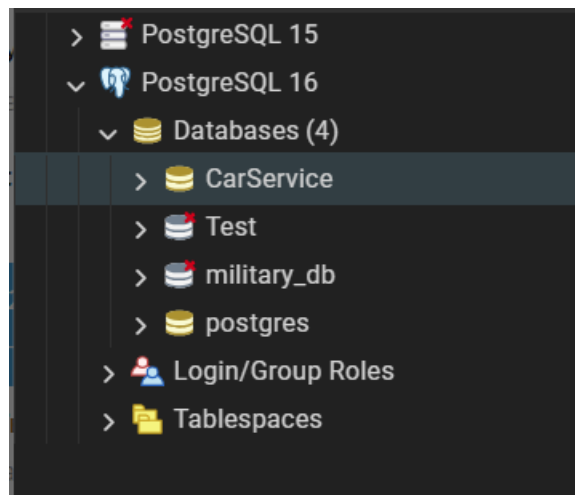


Рисунок 6.1 – список доступних баз даних в СУБД після виконання команди

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

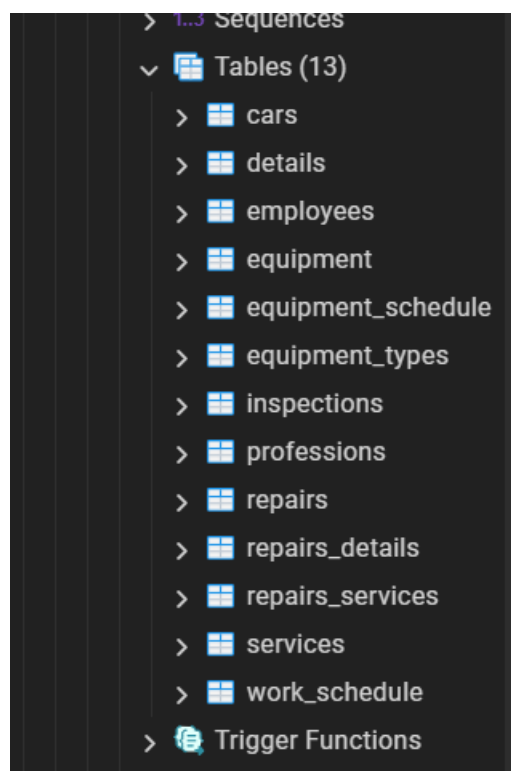


Рисунок 6.2 – список створених таблиць після виконання команди

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

Далі згенеруємо схему даних засобами СУБД.

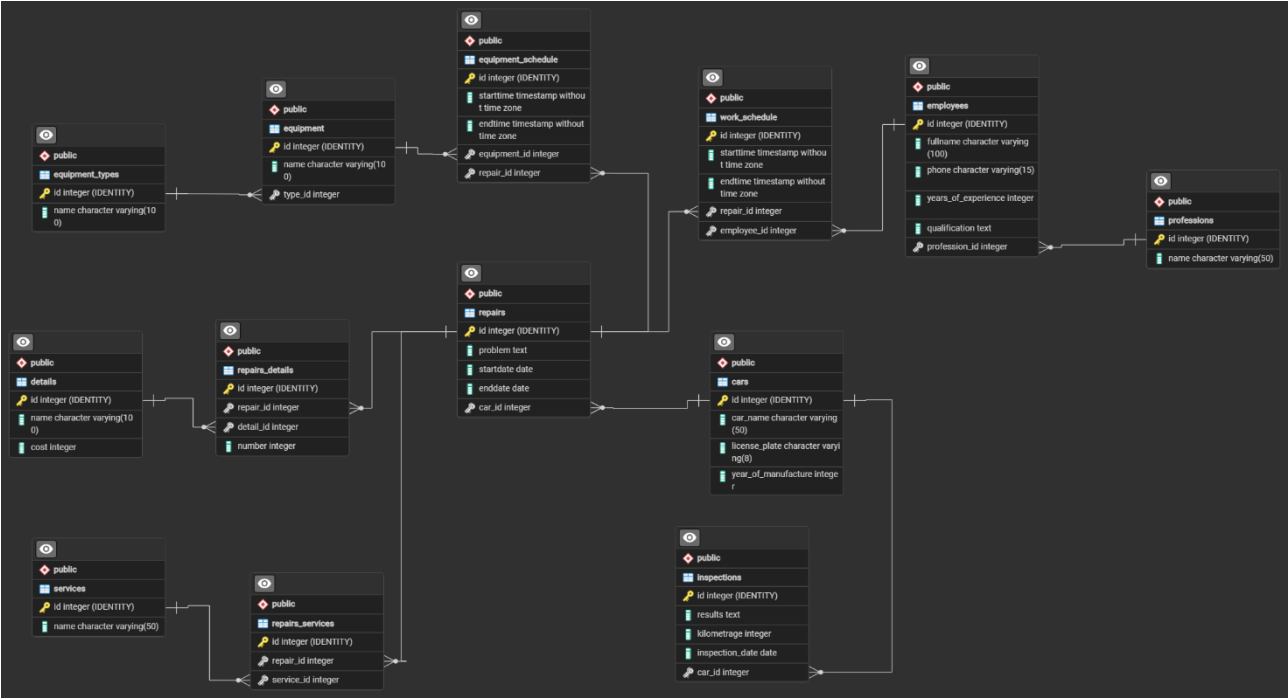


Рисунок 6.3 – Схема даних згенерована СУБД PostgreSQL

6.3. 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.

Data Output Messages Notifications						
	id [PK] integer	problem text	startdate date	enddate date	car_id integer	
1	1	Faulty Oxygen Sensor	2023-10-03	2023-10-05	159	
2	2	Engine Overheating	2023-08-17	2023-08-26	163	
3	3	Failed Water Pump	2023-10-22	2023-10-29	28	
4	4	Transmission Failure	2023-12-25	2023-12-28	35	
5	5	Ignition Coil Failure	2023-12-29	2024-01-01	87	
6	6	Ignition Coil Failure	2023-09-23	2023-09-28	192	
7	7	Ignition Coil Failure	2023-10-20	2023-10-26	125	
8	8	Fuel Pump Failure	2023-11-20	2023-11-23	96	
9	9	Broken Serpentine Belt	2023-12-11	2023-12-19	163	
10	10	Broken Suspension	2023-12-15	2023-12-25	10	

Рисунок 6.4 – Вибірка даних з таблиці “repairs”

	id [PK] integer	car_name character varying (50)	license_plate character varying (8)	year_of_manufacture integer
1	1	Mercedes-Benz M-Class	AO1399PA	2016
2	2	Chrysler 300	CA9878BP	2011
3	3	Volkswagen Golf	BX4365TB	2009
4	4	Alfa Romeo Spider	AB9871MP	2011
5	5	Jaguar XK Series	BE9024XO	2018
6	6	Infiniti I	AI7807MB	2020
7	7	Kia Spectra	CE0947MC	2017
8	8	GMC Savana 3500	BT6435EI	2010
9	9	GMC 1500 Club Coupe	AE0773OK	2011
10	10	Saturn L-Series	AX9543EE	2019
11	11	Volkswagen Cabriolet	AA2415II	2007

Рисунок 6.5 – Вибірка даних з таблиці “cars”

#### 6.4. 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**.

CarService=> SELECT \* FROM work\_schedule;

id	starttime	endtime	repair_id	employee_id
1	2023-12-05 11:50:00	2023-12-05 14:50:00	122	28
2	2023-12-05 08:00:00	2023-12-05 12:00:00	37	15
3	2023-12-07 15:10:00	2023-12-07 18:10:00	35	11
4	2023-12-07 19:10:00	2023-12-07 20:10:00	44	11
5	2023-12-07 10:00:00	2023-12-07 12:00:00	30	5
6	2023-12-07 13:00:00	2023-12-07 17:00:00	28	5
7	2023-12-25 09:00:00	2023-12-25 14:00:00	3	16
8	2023-12-25 13:30:00	2023-12-25 18:30:00	74	9
9	2023-09-14 12:20:00	2023-09-14 14:20:00	47	29
10	2023-09-14 15:40:00	2023-09-14 18:40:00	64	1
11	2023-09-22 11:50:00	2023-09-22 14:50:00	22	25
12	2023-09-22 10:40:00	2023-09-22 14:40:00	39	16
13	2023-11-13 13:50:00	2023-11-13 17:50:00	40	8
14	2023-11-13 17:50:00	2023-11-13 18:50:00	78	8
15	2023-11-13 15:30:00	2023-11-13 16:30:00	14	5

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

```
CarService=> SELECT * FROM details;
ERROR: permission denied for table details
CarService=> |
```

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

## 7 РОБОТА З БАЗОЮ ДАНИХ

### 7.1. Генератори

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

```
id int GENERATED ALWAYS AS IDENTITY
```

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

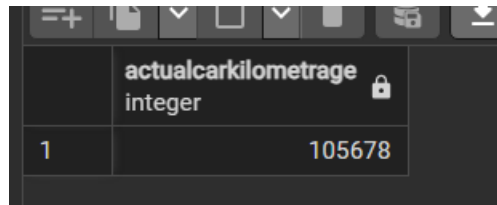
### 7.2. Збережені процедури та функції

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

#### 7.2.1. 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;
```



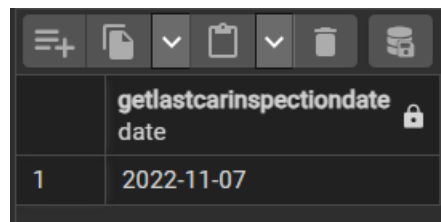
	actualcarkilometrage integer
1	105678

Рисунок 7.1 – Результат виконання ActualCarKilometrage для авто з ідентифікатором 15

#### 7.2.2. 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;
```



	getlastcarinspectiondate date
1	2022-11-07

Рисунок 7.2 - Результат виконання GetLastCarInspectionDate для авто з ідентифікатором 12

### 7.2.3. 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;
```

	car_name character varying	license_plate character varying	time_since_last_inspection interval
1	Chrysler 300	CA9878BP	10 mons
2	Saturn L-Series	AX9543EE	1 year 1 mon 14 days
3	Ford Mustang	BO1828TP	1 year 1 mon 21 days
4	Mercury Sable	AM7804MI	1 year 3 mons 11 days
5	Ford E-Series	AC6225CK	1 year 3 mons 23 days
6	Jaguar XJ Series	BI9811HX	1 year 2 mons 7 days
7	Scion tC	BA3780CX	3 mons 25 days

Рисунок 7.3 – Результат виконання ActualCarKilometrage

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

#### 7.2.4. 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;
```

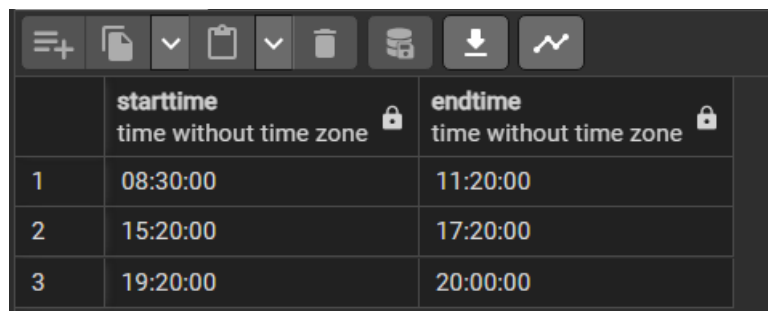


```

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;

```



	starttime time without time zone	endtime time without time zone
1	08:30:00	11:20:00
2	15:20:00	17:20:00
3	19:20:00	20:00:00

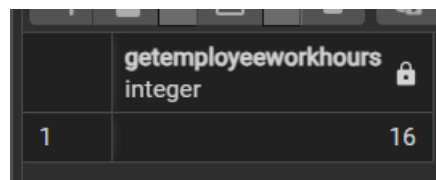
Рисунок 7.4 - Результат виконання GetTimeForEquipmentReservation

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.

#### 7.2.5. 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;
```



	getemployeeworkhours integer
1	16

Рисунок 7.6 – Результат виконання GetEmployeeWorkHours

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

#### 7.2.6. 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;

```

	fullname character varying	workload character varying
1	Ameliya Harmash	8.3%
2	Bohdan Harmatiuk	6.6%
3	Bohdan Rodyn	13.3%
4	Danylo Hryhorenko	5.8%
5	Ihor Moroz	2.5%
6	Iryna Romanyuk	4.1%
7	Kostyantyn Bilyi	2.5%
8	Kyrylo Tkachuk	4.1%
9	Leonid Antonenko	0.8%
10	Mariya Ivanenko	6.6%
11	Marta Ivanyuk	7.5%
12	Matviy Ivanov	7.5%
13	Mykhailo Tkachenko	14.1%
14	Oleksiy Petrenko	3.3%

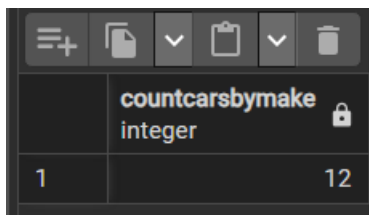
Рисунок 7.7 – Результат виконання CalculateEmpolyeesWorkLoad

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

#### 7.2.7. 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;
```



	countcarsbymake integer
1	12

Рисунок 7.8 – Результат виконання CountCarsByMake

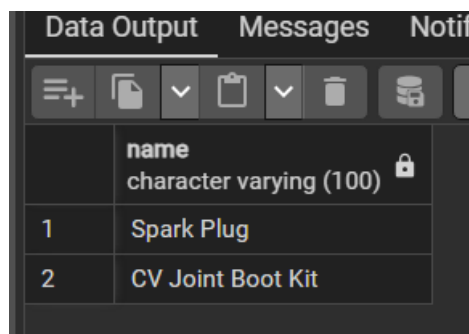
На рисунку 7.8 наведена кількість автомобілів марки “ Фольксваген”

#### 7.2.8. 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.

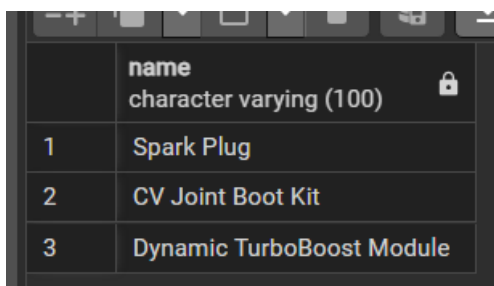
Щоб переконатись, що процедура працює, виконаємо виконаємо наступну команду:

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



	name character varying (100)
1	Spark Plug
2	CV Joint Boot Kit

Рисунок 7.9 – Список замієнених деталей до виконання процедури



	name character varying (100)
1	Spark Plug
2	CV Joint Boot Kit
3	Dynamic TurboBoost Module

Рисунок 7.10 – Список замієнених деталей після виконання процедури

#### 7.2.9. 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.

```

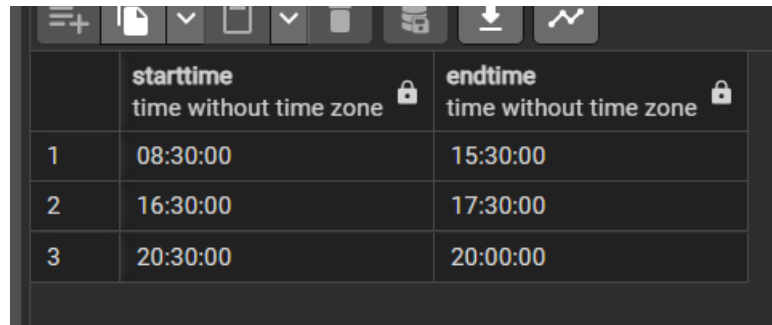
CREATE OR REPLACE FUNCTION GetTimeForEmployeeReservation
(
    workDayStart time,
    workDayEnd time,
    reservationDate date,
    employeeID int
)
RETURNS TABLE(
    startTime time,
    endTime time
)
AS $$
DECLARE
    reservation record;
    previousTime time;
BEGIN
    previousTime := workDayStart;
    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;

```



	starttime time without time zone	endtime time without time zone
1	08:30:00	15:30:00
2	16:30:00	17:30:00
3	20:30:00	20:00:00

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.

#### 7.2.10. Функція MakeReportAboutCar

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

```

REPAIR 2023-11-05
PROBLEM: Dead Battery
SERVICE:
  Brake Caliper and Rotor Replacement
REPLACED DETAILS:
  Brake Disc Set

REPAIR 2023-10-22
PROBLEM: Failed Water Pump
SERVICE:
  Oil Change and Filter Replacement
REPLACED DETAILS:

```

Рисунок 7.11 – Результат виконання MakeReportAboutCar

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

#### 7.2.11. 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 `reserveEquipment` 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;
```

#### 7.2.12. 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 `reserveEmployee` та



UpdateReservationEmployee , результат виконання функції наведений в пунктах 1.3.9 та 1.3.10 відповідно.

### 7.3. Тригери

#### 7.3.1. 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);
```

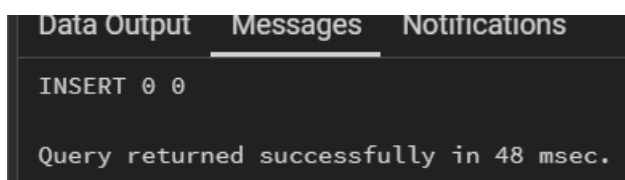


Рисунок 7.12 – Результат спрацювання тригера checkRepair

## 7.3.2. 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;
```

14	14	Saab 9-7X	AP7415XA
15	15	Honda Pilot	AC5409IA
16	16	Cadillac Eldorado	BP7325OK

Рисунок 7.13 – таблиця “cars” до видалення

	id [PK] integer	problem text	startdate date	enddate date	car_id integer
1	81	Exhaust System Failure	2023-11-10	2023-11-13	15
2	85	Flat Tire	2023-08-07	2023-08-14	15

Рисунок 7.14 – Ремонти автомобіля до видалення

	results text	kilometrage integer	inspection_date date	car_id integer
9	Engine: Ok, Tires: normal, Transmission: Ok, Test-drive: Pass...	105678	2023-12-15	15

Рисунок 7.15 – Огляди автомобіля до видалення

13	Mercury Sable	AM7804MI
14	Saab 9-7X	AP7415XA
16	Cadillac Eldorado	BP73250K

Рисунок 7.16 – таблиця “cars” після видалення

id [PK] integer	problem text	startdate date	enddate date	car_id integer

Рисунок 7.17 – Ремонти автомобіля після видалення

id [PK] integer	results text	kilometrage integer	inspection_date date	car_id integer

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.

### 7.3.3. Тригер checkLicensePlateBeforeAddition

Даний тригер виконується 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;
END;
```

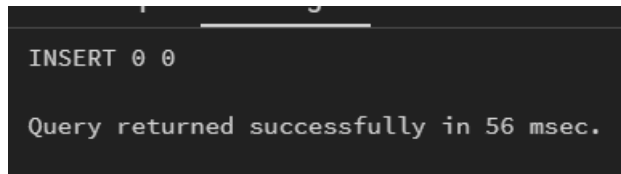
```
IF regionCode IN ('AA', 'AB', 'AC', 'AE', 'AH', 'AI', 'AM', 'AO', 'AP', 'AT', 'AX',
'BA', 'BB', 'BC', 'BE', 'BH', 'BI', 'BM', 'BO', 'BP', 'BT', 'BX',
'CA', 'CB', 'CE') THEN
    IF series ~ '[A-Z]{2}' THEN RETURN NEW;
    END IF;
END IF;
RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

```
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);



```
INSERT 0 0
Query returned successfully in 56 msec.
```

Рисунок 7.19 – Результат спрацювання тригера  
checkLicensePlateBeforeAddition

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

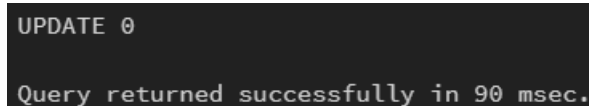
#### 7.3.4. Тригер checkLicensePlateBeforeUpdating

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 = '34AAAAA45' WHERE id = 15;
```



```
UPDATE 0
Query returned successfully in 90 msec.
```

Рисунок 7.20 – Результат спрацювання тригера  
checkLicensePlateBeforeAddition

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

#### 7.3.5. 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.

id [PK] integer	results text	kilometrage integer	inspection_date date	car_id integer
12	Engine: Unstable work, Tires: normal, Transmission: Ok, Test-drive: didn't pa...	137269	2023-09-01	75
13	Engine: Ok, Tires: normal, Transmission: Ok, Test-drive: Passed	150567	2023-10-26	75
14	Engine: Ok, Tires: Good, Transmission: Leaks observed, Test-drive: didn't pa...	172000	2023-11-20	75

Рисунок 7.21 – Технічні огляди автомобіля з id 75

Для оновлення пробігу виконаємо наступну команду:

```
UPDATE inspections SET kilometrage = 200000 WHERE id = 13;
```

```
UPDATE 0
Query returned successfully in 97 msec.
```

Рисунок 7.22 – Результат спрацювання триггеру checkKilometrageBeforeUpdating  
Як бачимо із рисунку 7.22 тригер спрацював і не дав оновити пробіг на більший ніж у наступному огляді.

### 7.3.6. 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);
```

```
INSERT 0 0

Query returned successfully in 50 msec.
```

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.

### 7.3.7. 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 CheckReservationTimeForEquipment ()
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 CheckReservationTimeForEquipment ();
```

id [PK] integer	starttime timestamp without time zone	endtime timestamp without time zone	equipment_id integer	repair_id integer
10	2023-09-01 13:40:00	2023-09-01 17:40:00	7	102
11	2023-09-01 19:40:00	2023-09-01 20:40:00	7	114

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);
```

```
INSERT 0 0
Query returned successfully in 154 msec.
```

Рисунок 7.25 – Результат спрацювання тригера TryReservateEquipment

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

### 7.3.8. Тригер UpdateReservationTimeForEquipment

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

```
CREATE OR REPLACE TRIGGER UpdateReservationTimeForEquipment
BEFORE UPDATE ON equipment_schedule
FOR EACH ROW EXECUTE FUNCTION CheckReservationTimeForEquipment();
```

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;
```

```
UPDATE 0
Query returned successfully in 47 msec.
```

Рисунок 7.26 – Результат спрацювання тригера  
UpdateReservationTimeForEquipment

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

### 7.3.9. 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 CheckReservationTimeForEmployee()
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 CheckReservationTimeForEmployee();
```

id [PK] integer	starttime timestamp without time zone	endtime timestamp without time zone	repair_id integer	employee_id integer
5	2023-12-07 10:00:00	2023-12-07 12:00:00	30	5
6	2023-12-07 13:00:00	2023-12-07 17:00:00	28	5

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);
```

```
INSERT 0 0

Query returned successfully in 46 msec.
```

Рисунок 7.28 – Результат спрацювання тригера TryReservateEmployee

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

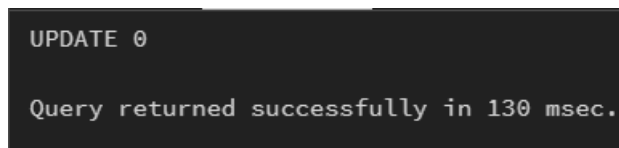
### 7.3.10. Тригер UpdateReservationTimeForEmployee

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 CheckReservationTimeForEmployee();
```

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;
```



```
UPDATE 0
Query returned successfully in 130 msec.
```

Рисунок 7.29 – Результат спрацювання тригера  
UpdateReservationTimeForEmployee

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

#### 7.4. 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.

##### 7.4.1. 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;
```




	<b>name</b> character varying (100) 	<b>cost</b> integer 	<b>number</b> bigint 
1	Brake Rotor	9876	3
2	Cabin Air Filter	876	3
3	Camshaft	3210	3
4	Camshaft Position Sensor	432	1
5	Catalytic Converter	5432	2
6	Fuel Filter	543	1
7	Fuel Pressure Sensor	8765	2
8	Fuel Tank Cap	8765	3
9	Ignition Control Module Connector	234	4
10	Ignition Starter Switch	8765	4
11	Intake Manifold Gasket	321	3
12	Mass Air Flow Sensor	987	1
13	Oxygen Sensor Relay	987	2

Рисунок 7.30 – Вибірка з представлення usedDetailsInThisMonth

#### 7.4.2. 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;
```

	name character varying (100)	hours numeric
1	Diagnostic Scanner 1	27
2	Welding machine 2	26
3	Wheel balancing stand	25
4	Oil change machine LEX 85L	21
5	Lift platform 1	18
6	Lift platform 2	17
7	Air Compressor 2	16
8	Air Compressor 1	13
9	Diagnostic Scanner 2	10
10	Lift platform 3	8

Рисунок 7.31 – Вибірка з представлення

## EquipmentOperationTimeThisMonth

## 7.4.3. 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;
```

	<b>car_name</b> character varying (50) 🔒	<b>license_plate</b> character varying (8) 🔒	<b>totalcost</b> bigint 🔒
1	GMC Sierra 1500	AA4406BA	40467
2	GMC 1500 Club Coupe	AE0773OK	35060
3	Dodge Caravan	AA6122EX	29628
4	Buick Riviera	BI8863TC	26295
5	Jaguar XJ Series	BI9811HX	17530
6	Dodge Grand Caravan	CE4176KH	12493
7	Chrysler Crossfire	AH6760OM	9630
8	Nissan 350Z	BE2886BK	6576
9	Saturn L-Series	AX9543EE	3060
10	Mazda MX-6	CE7745HI	2628

Рисунок 7.32 – Вибірка з представлення RepairCostsInThisMonth

## 7.5. 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.

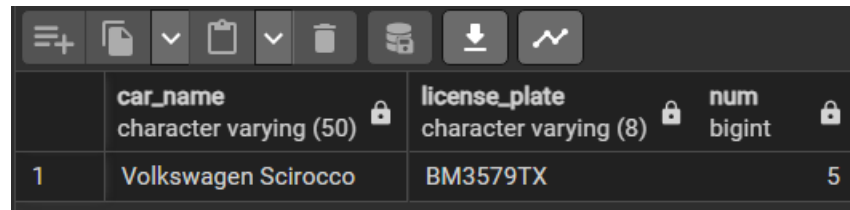
### 1.5.1. Запит, що виводить автомобілі, що найчастіше ремонтувались

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);

```



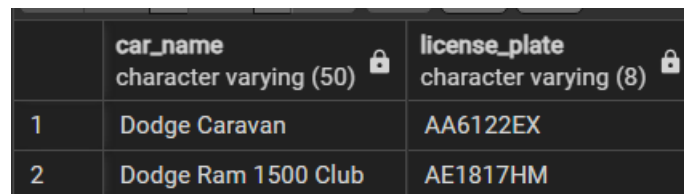
	car_name character varying (50)	license_plate character varying (8)	num bigint
1	Volkswagen Scirocco	BM3579TX	5

Рисунок 7.33 – Автомобілі, які найчастіше ремонтувались

## 1.5.2. 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;
```



	car_name character varying (50)	license_plate character varying (8)
1	Dodge Caravan	AA6122EX
2	Dodge Ram 1500 Club	AE1817HM

Рисунок 7.34 – Автомобілі, що в даний момент ремонтуються

## 1.5.3. 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;
```

car_name character varying (50)	license_plate character varyi	name character varying (100)	date date
GMC Sierra 1500	AA4406BA	Intake Manifold Gasket	2023-12-25
GMC Sierra 1500	AA4406BA	Wheel Cylinder	2023-12-25
Dodge Caravan	AA6122EX	Brake Rotor	2023-12-30
Kia Amanti	AA9184MT	Fuel Filter	2023-12-18
Ford Expedition	AA9362BC	Transmission Pan	2023-08-26
Cadillac Escalade	AC1651KH	Engine Air Intake Hose	2023-09-09
Ford E-Series	AC6225CK	Headlight Assembly	2023-10-22

Рисунок 7.35 – Автомобілі та їх замінені деталі

1.5.4. 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;
```

car_name character varying (50)	license_plate character var	name character varying (50)	date date
Toyota FJ Cruiser	AA4301EH	Transmission Rebuild	2023-08-02
GMC Sierra 1500	AA4406BA	Fuel Injector Cleaning	2023-10-12
Toyota Camry	AA5564XI	Power Steering Fluid Flush	2023-12-13
Kia Amanti	AA9184MT	Emission System Repair	2023-12-18
Pontiac Grand Prix	AB4271KK	Engine Diagnostics and Repair	2023-12-10
Pontiac Bonneville	AB5383HC	Drive Shaft and CV Joint Repair	2023-10-09
GMC Vandura G3500	AC2312HK	Thermostat Replacement	2023-12-06
Ford E-Series	AC6225CK	Engine Diagnostics and Repair	2023-10-22

Рисунок 7.36 – Автомобілі та обслуговування, проведене для них

1.5.5. 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;
```

fullname character varying	car_name character varying (50)	license_plate character varying (10)	problem text	starttime timestamp without time zone
Bohdan Rodyn	Chrysler Crossfi...	AH67600M	Broken Serpentine Belt	2023-12-29 09:00:00
Yaroslav Bilous	Ford E-Series	AC6225CK	Brake Failure	2023-12-29 13:00:00
Bohdan Rodyn	Chevrolet SSR	AI8824TM	Radiator Leak	2023-12-29 13:00:00
Yaroslav Bilous	Mercury Sable	AM7804MI	Faulty Oxygen Sensor	2023-12-29 18:00:00

Рисунок 7.37 – План роботи працівників на 29 грудня

#### 1.5.6. 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;
```

name character varying (100)	car_name character varying (50)	license_plate character varying (10)	starttime timestamp without time zone
Oil change machine LEX 85L	Ford Escort	BE6821CX	2023-12-29 11:20:00
Lift platform 2	Lexus ES	AT5667IM	2023-12-29 14:50:00
Oil change machine LEX 85L	Hyundai Azera	AM4904PE	2023-12-29 16:20:00
Lift platform 2	Pontiac Grand Prix	BH0942BM	2023-12-29 16:50:00
Oil change machine LEX 85L	GMC Sierra 1500	AA4406BA	2023-12-29 19:20:00

Рисунок 7.38 – Бронювання обладнання на 29 грудня

## 1.5.7. 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);
```

car_name	license_plate	kilometrage
character varying (50)	character vary	integer
Jaguar XK Series	BE9024XO	300120

Рисунок 7.39 – Автомобіль з найбільшим пробігом

## 1.5.8. 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;
```

	fullname	name
	character varying (100)	character varying (50)
1	Oleksiy Petrenko	Certified Climate System Expert
2	Mariya Ivanenko	Auto Electrician-Mechanic
3	Roman Kovalchuk	Air Conditioning and Refrigeration Technician
4	Yevheniya Melnyk	Automotive Structural Welding Mechanic
5	Mykhailo Tkachenko	Automotive Electronics Technician
6	Yuliya Shevchenko	Automotive Chassis Repair Technician
7	Kostyantyn Bilyi	Electronic Systems Diagnostics Technician
8	Sofiya Andriyenko	Suspension System Specialist

Рисунок 7.40 – Працівники та їх спеціальність

## 1.5.9. 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;
```

car_name character varying (50)	problem text	fullname character varying (100)	starttime timestamp without time zone
GMC Vandura G35...	Engine Overheating	Viktoriya Tretyak	2023-12-05 11:50:00
Ram 1500	Brake Failure	Bohdan Rodyn	2023-12-05 08:00:00
Saturn L-Series	Exhaust System Fail...	Matviy Ivanov	2023-12-07 15:10:00
Audi Allroad	Broken Suspension	Matviy Ivanov	2023-12-07 19:10:00
Toyota FJ Cruiser	Alternator Failure	Mykhailo Tkachenko	2023-12-07 10:00:00
Hyundai Azera	Failed Water Pump	Mykhailo Tkachenko	2023-12-07 13:00:00

Рисунок 7.42 – Автомобілі та робітники, що їх ремонтували

1.5.10. 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);
```

name character varying (100)	count bigint
Lift platform	64

Рисунок 7.42 – Тип обладнання, що найчастіше використовувався

1.5.11. 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;
```

	averagecost	
	numeric	
1	44946.000000000000	

Рисунок 7.43 – Середня вартість деталей для ремонту

1.5.12. 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
);
```

	averageinterval	
	interval	
1	2 mons 31 days 03:00:00	

Рисунок 7.44 – Середній час між технічними оглядами

1.5.13. 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
)
)
```

car_name character varying (50)	results text	kilometrage integer	inspection_date date
Jaguar XK Series	Engine: Ok, Tires: Normal, Transmission: Ok, Test-d...	150250	2022-09-03
Jaguar XK Series	Engine: Ok, Tires: Good, Transmission: Clutch slips, ...	300120	2023-10-03

Рисунок 7.45 – Технічні огляди авто з найбільшим пробігом

1.5.14. 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 можж 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;
```

fullname character varying (100)	hours numeric
Bohdan Rodyn	24

Рисунок 7.46 – Робітник, який відпрацював найбільшу кількість годин

1.5.15. 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)
```

dow	count
text	bigint
Friday	25

Рисунок 7.48 – Найбільш навантажений день тижня

1.5.16. 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
```

15	Suzuki Samurai	AC9434XP	3.000
16	Mercury Sable	AM7804MI	3.000
17	Buick Riviera	BI8863TC	3.000
18	GMC Sierra 1500	BM0904HI	3.000
19	Acura ZDX	AM2937OO	4.000
20	Pontiac Bonneville	BO5493IC	4.000
21	GMC Suburban 1500	CA1740KI	4.000
22	Dodge Ram	BP2618IA	4.000
23	Plymouth Colt	BB3651AM	4.000
24	Lamborghini Murciélago	BA1108XB	4.000
25	Acura TL	AN6411IE	4.000

Рисунок 7.49 – Середні тривалості ремонтів автомобілів у днях

1.5.17. 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%';
```

1	Volkswagen Golf	BX4365TB	Engine Overheating
2	Honda Element	BE3567OX	Engine Overheating
3	Mercedes-Benz R-Class	BE7856MK	Engine Overheating
4	Volkswagen Passat	CE2286AA	Engine Overheating
5	Chrysler Crossfire	AN6760OM	Engine Overheating

Рисунок 7.50 – Автомобілі, в яких були проблеми з двигуном

1.5.18. 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

year_of_manufacture integer	kilometrage numeric
2007	148755.000
2009	199526.000
2010	172000.000
2011	120000.000
2016	86464.500
2017	181361.000
2018	223505.000
2019	121345.000

Рисунок 7.51 – Середній пробіг автомобілів за роками їх випуску

#### 1.5.19. 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
```

8	August	198230
9	September	182022
10	October	124989
11	November	204060
12	December	189619

Рисунок 7.52 – Витрати на деталі по місяцям

#### 1.5.20. Запит, що виводить обслуговування, яке робили найчастіше у поточному місяці.

Запит відбирає всі ремонти поточного місяця, рахує в них обслуговування та виводить те, яке зустрічалось 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;
```

name	count
character varying (50)	bigint
Emission System Repair	2

Рисунок 7.53 – Послуга, яка найчастіше надавалась у грудні

## 7.6. 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.

### 1.6.1. 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.

```
CarService=# EXPLAIN ANALYZE SELECT * FROM equipment_schedule
CarService=# JOIN repairs ON repair_id = repairs.id
CarService=# JOIN cars ON car_id = cars.id
CarService=# WHERE startTime BETWEEN '2023-10-24 8:30' AND '2023-10-24 20:00';
                                QUERY PLAN

-----
Nested Loop  (cost=5.87..11.12 rows=4 width=94) (actual time=0.093..0.122 rows=3 loops=1)
-> Hash Join  (cost=5.72..9.95 rows=4 width=62) (actual time=0.083..0.107 rows=3 loops=1)
    Hash Cond: (repairs.id = equipment_schedule.repair_id)
    -> Seq Scan on repairs  (cost=0.00..3.25 rows=125 width=34) (actual time=0.013..0.025 rows=123 loops=1)
    -> Hash  (cost=5.67..5.67 rows=4 width=28) (actual time=0.052..0.053 rows=3 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 9kB
        -> Seq Scan on equipment_schedule  (cost=0.00..5.67 rows=4 width=28) (actual time=0.043..0.047 rows=3 loops=1)
            Filter: ((starttime >= '2023-10-24 08:30:00'::timestamp without time zone) AND (starttime <= '2023-10-24 20:00:00'::timestamp without time zone))
            Rows Removed by Filter: 242
-> Index Scan using pk_car_id on cars  (cost=0.14..0.29 rows=1 width=32) (actual time=0.004..0.004 rows=1 loops=3)
    Index Cond: (id = repairs.car_id)
Planning Time: 0.434 ms
Execution Time: 0.157 ms
```

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);

```
CarService=# EXPLAIN ANALYZE SELECT * FROM equipment_schedule
CarService=# JOIN repairs ON repair_id = repairs.id
CarService=# JOIN cars ON car_id = cars.id
CarService=# WHERE startTime BETWEEN '2023-10-24 8:30' AND '2023-10-24 20:00';
                                QUERY PLAN

-----
Nested Loop  (cost=5.87..11.12 rows=4 width=94) (actual time=0.045..0.060 rows=3 loops=1)
-> Hash Join  (cost=5.72..9.95 rows=4 width=62) (actual time=0.042..0.053 rows=3 loops=1)
    Hash Cond: (repairs.id = equipment_schedule.repair_id)
    -> Seq Scan on repairs  (cost=0.00..3.25 rows=125 width=34) (actual time=0.008..0.015 rows=123 loops=1)
    -> Hash  (cost=5.67..5.67 rows=4 width=28) (actual time=0.026..0.026 rows=3 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 9kB
        -> Seq Scan on equipment_schedule  (cost=0.00..5.67 rows=4 width=28) (actual time=0.022..0.024 rows=3 loops=1)
            Filter: ((starttime >= '2023-10-24 08:30:00'::timestamp without time zone) AND (starttime <= '2023-10-24 20:00:00'::timestamp without time zone))
            Rows Removed by Filter: 242
-> Index Scan using pk_car_id on cars  (cost=0.14..0.29 rows=1 width=32) (actual time=0.002..0.002 rows=1 loops=3)
    Index Cond: (id = repairs.car_id)
Planning Time: 0.233 ms
Execution Time: 0.081 ms
```

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.

### 1.6.2. Оптимізація другого запиту

Для остаточної перевірки роботи індексів оптимізуємо наступний запит:

WITH kilometres 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.

Sort (cost=38.54..38.57 rows=14 width=36) (actual time=0.197..0.198 rows=8 loops=1)
Sort Key: cars.year_of_manufacture
Sort Method: quicksort Memory: 25kB
-> HashAggregate (cost=38.06..38.27 rows=14 width=36) (actual time=0.166..0.171 rows=8 loops=1)
Group Key: cars.year_of_manufacture
Batches: 1 Memory Usage: 24kB
-> Hash Join (cost=32.53..37.06 rows=199 width=8) (actual time=0.138..0.144 rows=12 loops=1)
Hash Cond: (inspections.car_id = cars.id)
-> HashAggregate (cost=26.05..28.05 rows=200 width=8) (actual time=0.022..0.024 rows=12 loops=1)
Group Key: inspections.car_id
Batches: 1 Memory Usage: 40kB
-> Seq Scan on inspections (cost=0.00..20.70 rows=1070 width=8) (actual time=0.010..0.011 rows=19 loop...)
-> Hash (cost=3.99..3.99 rows=199 width=8) (actual time=0.080..0.080 rows=199 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 16kB
-> Seq Scan on cars (cost=0.00..3.99 rows=199 width=8) (actual time=0.029..0.054 rows=199 loops=1)
Planning Time: 2.057 ms
Execution Time: 0.261 ms

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);
```

QUERY PLAN
text
Sort (cost=7.00..7.03 rows=14 width=36) (actual time=0.124..0.125 rows=8 loops=1)
Sort Key: cars.year_of_manufacture
Sort Method: quicksort Memory: 25kB
-> HashAggregate (cost=6.52..6.73 rows=14 width=36) (actual time=0.113..0.117 rows=8 loops=1)
Group Key: cars.year_of_manufacture
Batches: 1 Memory Usage: 24kB
-> Hash Join (cost=1.90..6.43 rows=19 width=8) (actual time=0.082..0.107 rows=12 loops=1)
Hash Cond: (cars.id = kilometrages.car_id)
-> Seq Scan on cars (cost=0.00..3.99 rows=199 width=8) (actual time=0.013..0.024 rows=199 loops=1)
-> Hash (cost=1.67..1.67 rows=19 width=8) (actual time=0.025..0.025 rows=12 loops=1)
Buckets: 1024 Batches: 1 Memory Usage: 9kB
-> Subquery Scan on kilometrages (cost=1.29..1.67 rows=19 width=8) (actual time=0.020..0.023 rows=12 loop...)
-> HashAggregate (cost=1.29..1.48 rows=19 width=8) (actual time=0.020..0.022 rows=12 loops=1)
Group Key: inspections.car_id
Batches: 1 Memory Usage: 24kB
-> Seq Scan on inspections (cost=0.00..1.19 rows=19 width=8) (actual time=0.007..0.009 rows=19 loop...)
Planning Time: 0.182 ms
Execution Time: 0.162 ms

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 автосервісу, 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;
```