

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 для морського вантажного терміналу

Студента 2 курсу ІІІ-33 групи
спеціальності 121 «Інженерія програмного
забезпечення»

Бресківа Давида Андрійовича
(прізвище та ініціали)

Керівник Ліщук Катерина Ігорівна, доц., к.т.н.

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

Національна шкала _____

Кількість балів: _____ Оцінка ECTS _____

Члени комісії

(підпис) _____ (вчене звання, науковий ступінь, прізвище та ініціали)

(підпис) _____ (вчене звання, науковий ступінь, прізвище та ініціали)

(підпис) _____ (вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2024 рік

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки
(повна назва)

Кафедра Інформатики та програмної інженерії
(повна назва)

Дисципліна Бази даних

Курс 2 Група ІІІ-33 Семестр 3

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

Бресківу Давиду Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи База даних для морського вантажного терміналу

керівник роботи Ліщук Катерина Ігорівна, к.т.н.

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

2. Строк подання студентом роботи 23.12.2024

3. Вихідні дані до роботи завдання на розробку база даних складського обліку підприємства

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

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

7) Створення мовою SQL запитів

8) Оптимізація роботи запитів

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

6. Дата видачі завдання 06.11.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/П	Назва етапів виконання курсового проекту	Строк виконання етапів проекту	Примітка
1	Аналіз предметного середовища	18.11.2024	
2	Побудова ER-моделі	10.12.2024	
3	Побудова реляційної схеми з ER-моделі	11.12.2024	
4	Створення бази даних, у форматі обраної системи управління базою даних	14.12.2024	
5	Створення користувачів бази даних	18.12.2024	
6	Імпорт даних з використанням засобів СУБД в створену базу даних	20.12.2024	
7	Створення мовою SQL запитів	22.12.2024	
8	Оптимізація роботи запитів	25.12.2024	
9	Оформлення пояснювальної записки	30.12.2024	
10	Захист курсової роботи	28.01.2025	

Студент

(підпис) Давид БРЕСКІВ
(прізвище та ініціали)

Керівник роботи

(підпис) Катерина ЛІЩУК
(прізвище та ініціали)

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 94 сторінки, 56 рисунків, 12 таблиць, 9 посилань.

Об'єкт дослідження: база даних для управління морським вантажним терміналом.

Мета роботи: розробка та реалізація бази даних для управління морським вантажним терміналом.

У межах роботи проведено аналіз предметного середовища, на основі якого побудовано ER-модель та реляційну схему. Далі була створена база даних у форматі PostgreSQL, у якій налаштовані користувачі та їхні ролі. Дані були імпортовані до цієї бази. Також розроблено функції, процедури, тригери, представлення та SQL-запити для забезпечення ефективної роботи з базою даних. Окрім цього, виконано оптимізацію запитів шляхом додавання індексів для підвищення продуктивності роботи бази даних.

Реалізація бази даних для управління морським вантажним терміналом була виконана успішно, відповідно до поставлених завдань.

Зміст

ВСТУП	5
1 АНАЛІЗ ПРЕДМЕТНОГО СЕРЕДОВИЩА	7
1.1 Опис предметного середовища	7
1.2 Аналіз існуючих програмних продуктів	9
2 ПОСТАНОВКА ЗАВДАННЯ	12
3 ПОБУДОВА ER-МОДЕЛІ	13
3.1 Бізнес-правила	13
3.2 Вибір сутностей	14
3.3 Опис сутностей	15
4 РЕАЛЯЦІЙНА МОДЕЛЬ БАЗИ ДАНИХ.....	22
5 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ	31
5.1 Створення бази даних	31
6 СТВОРЕННЯ КОРИСТУВАЧІВ БАЗИ ДАНИХ	35
6.1 Адміністратор	35
6.2 Менеджер	36
6.3 Оператор.....	37
6.4 Аналітик	37
6.5 Керівник	38
6.6 Технік.....	38
6.7 Оператор дока	39
6.8 Клієнт.....	40
7 РОБОТА З БАЗОЮ ДАНИХ.....	42
7.1 Тригери	42
7.2 Тексти представлень	52
7.3 Тексти збережених процедур та функцій.....	57
7.4 SQL-запити.....	71
7.5 Індeksi та результати оптимізації	89
ВИСНОВКИ.....	92
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	94

ВСТУП

Сучасний світ активно стикається зі зростанням обсягів міжнародної торгівлі, що зумовлює підвищення навантаження на інфраструктуру морських вантажних терміналів. У таких умовах ефективне управління логістичними процесами набуває вирішального значення для забезпечення швидкої обробки вантажів, мінімізації витрат та підвищення конкурентоспроможності портів. Розробка бази даних для морського вантажного терміналу є актуальною задачею, оскільки дозволяє автоматизувати та оптимізувати управління ключовими процесами, такими як облік кораблів, розкладів, вантажів, клієнтів та ресурсів терміналу.

На глобальному рівні морські порти є критично важливими вузлами для транспортування товарів, що забезпечують понад 80% світової торгівлі. Успішні рішення в галузі цифровізації портової діяльності вже продемонстровані провідними портами, такими як Роттердам, Сінгапур та Шанхай. Ці порти активно впроваджують інформаційні системи для моніторингу, аналізу та управління вантажопотоками, що дозволяє значно підвищити продуктивність їхньої роботи. Наприклад, у Роттердамі використовуються комплексні бази даних, інтегровані з системами штучного інтелекту, для прогнозування потоку вантажів та оптимізації розкладів [1]. У Сінгапурі реалізовано проекти, які дозволяють автоматично керувати розподілом причалів залежно від типу кораблів і вантажів. Ці тенденції демонструють важливість цифрових технологій у морській логістиці.

На сьогоднішній день багато наукових і практичних розробок присвячені автоматизації портових операцій. Проте, більшість таких рішень є специфічними для великих портів і мають високу вартість, що ускладнює їх впровадження в середніх та малих терміналах. Водночас, зростає потреба у створенні універсальних, масштабованих та доступних баз даних, які могли б бути адаптовані до потреб різних терміналів. Провідні наукові установи та організації, такі як Європейська організація портової логістики (ESPO) [2] та Міжнародна асоціація портів і гаваней (IAPH) [3], акцентують увагу на

необхідності створення інноваційних рішень для підвищення ефективності портових операцій.

Розроблена база даних може знайти широке застосування в управлінні портовою інфраструктурою, зокрема для обліку кораблів, планування розкладів, моніторингу стану причалів, управління вантажами та координації діяльності персоналу. Вона також може бути використана для автоматизації процесів прийняття рішень, підвищення прозорості в управлінні та інтеграції з іншими інформаційними системами, такими як митні чи транспортні сервіси. Таким чином, впровадження бази даних для морського вантажного терміналу є важливим кроком у напрямку підвищення ефективності та стійкості сучасної логістики.

1 АНАЛІЗ ПРЕДМЕТНОГО СЕРЕДОВИЩА

1.1 Опис предметного середовища

Морський вантажний термінал є важливою складовою глобальної логістичної інфраструктури, яка забезпечує перевезення товарів між континентами. Його діяльність охоплює обробку вантажів, їх тимчасове зберігання, завантаження та розвантаження суден, а також взаємодію з іншими видами транспорту, такими як автомобільний, залізничний і повітряний. Для забезпечення ефективності цих процесів термінал має бути оснащений сучасними технічними засобами, професійним персоналом і надійною інформаційною системою.

Морські термінали виконують функцію посередника між морським і сухопутним транспортом. Їх основними завданнями є прийом суден, обробка вантажів і організація їх подальшого транспортування. Основними елементами інфраструктури терміналу є причали, склади, підйомно-транспортне обладнання, а також зони для сортування й тимчасового зберігання вантажів. Ефективність роботи терміналу значною мірою залежить від оптимізації цих елементів та їхньої взаємодії.

Сучасні морські вантажні термінали стикаються з низкою викликів. Зростання обсягів міжнародної торгівлі вимагає підвищення пропускнуєї спроможності портів, впровадження інноваційних технологій і автоматизації процесів [4]. У цьому контексті використання інформаційних систем стає ключовим фактором підвищення продуктивності та забезпечення конкурентоспроможності. Дані системи дозволяють забезпечити управління потоками вантажів, оптимізувати розклад прибуття і відправлення суден, відстежувати стан складів і прогнозувати завантаженість терміналу.

Розробка бази даних для морського вантажного терміналу є важливим етапом в організації його роботи. Така база даних повинна включати інформацію про всі ключові елементи: судна, вантажі, клієнтів, транспортні засоби, розклади і персонал. Крім того, вона має підтримувати функції

відстеження статусу вантажів, планування використання ресурсів і формування звітності.

Системи управління базами даних для морських терміналів також повинні враховувати специфіку різних типів вантажів. Наприклад, контейнери, небезпечні вантажі або швидкопсувні товари потребують особливих умов обробки і зберігання. Інформаційна система повинна не лише відображати ці особливості, але й надавати інструменти для прийняття рішень щодо їхньої обробки та транспортування.

Суттєвою складовою предметного середовища є клієнти терміналу. Це можуть бути як великі корпорації, так і індивідуальні підприємці. Відповідно, інформаційна система повинна забезпечувати обробку запитів різного рівня складності, відстеження історії взаємодії з клієнтами, а також формування спеціалізованих пропозицій на основі аналізу їхніх потреб.

Крім внутрішньої організації, морський вантажний термінал функціонує у складному мережевому середовищі, взаємодіючи з іншими портами, логістичними компаніями, митними органами та страхувальниками. Це потребує створення інтегрованої бази даних, яка дозволяє швидко обмінюватися інформацією між усіма учасниками ланцюга постачання. Наприклад, вчасне оновлення статусу вантажу чи надання інформації про доступність причалів може значно скоротити час простою і знизити витрати.

Розробка бази даних для морського терміналу також повинна враховувати питання безпеки та надійності. Інформація про вантажі, клієнтів і персонал має бути захищена від несанкціонованого доступу, а система повинна забезпечувати резервне копіювання та швидке відновлення даних у разі збоїв. Крім того, важливим є забезпечення відповідності законодавчим нормам, які регулюють діяльність у сфері міжнародних перевезень.

Зважаючи на вищезазначене, предметне середовище морського вантажного терміналу є надзвичайно складним і багатограним. Воно охоплює технічні, логістичні, економічні та інформаційні аспекти, які необхідно враховувати при проектуванні бази даних. Успішна реалізація такої системи може суттєво підвищити ефективність роботи терміналу, знизити

витрати та забезпечити високу якість обслуговування клієнтів, що робить її важливим елементом сучасної логістичної інфраструктури.

1.2 Аналіз існуючих програмних продуктів

Сучасний ринок програмного забезпечення для управління морськими вантажними терміналами пропонує низку рішень, що спрямовані на автоматизацію логістичних процесів, зменшення операційних витрат і підвищення ефективності роботи портової інфраструктури. У цьому підпункті буде розглянуто основні існуючі програмні продукти, їх переваги, недоліки та особливості застосування.

Одним із провідних рішень у цій галузі є програмні комплекси типу Terminal Operating System (TOS). Такі системи, як Körber Supply Chain's TOS [5], та CyberLogitec OPUS Terminal [6], є широко визнаними на глобальному рівні. Вони пропонують комплексний підхід до управління операціями терміналу, включаючи планування розміщення контейнерів, моніторинг суднових розкладів, управління транспортними потоками та обробкою вантажів. Основною перевагою цих систем є їх інтегрованість із зовнішніми інформаційними системами, такими як портові спільноти та системи управління логістикою. Проте їх вартість, складність налаштування та необхідність залучення кваліфікованих фахівців для підтримки залишаються значними перешкодами для малих і середніх терміналів.

Ще однією групою програмного забезпечення є спеціалізовані рішення для управління окремими аспектами портової діяльності. Наприклад, CATOS (Computer-Aided Terminal Operation System) фокусується на оптимізації розташування контейнерів і плануванні роботи вантажного обладнання. Ця система дозволяє суттєво зменшити витрати часу на пошук вантажів та оптимізувати використання обладнання. Однак, її функціональність обмежується контейнерними терміналами і не враховує специфіку обробки інших типів вантажів, таких як сипучі або рідкі.

Для великих портів, що мають високий рівень автоматизації, застосовуються індивідуально розроблені рішення, такі як DP World's CARGOES TOS [7] або власні платформи для аналізу даних і підтримки прийняття рішень. Ці продукти надають можливість реалізувати специфічні потреби, але вимагають значних інвестицій у розробку, впровадження та підтримку. До їх переваг можна віднести масштабованість і гнучкість, однак такі рішення менш доступні для невеликих портів із обмеженими фінансовими ресурсами.

Значну роль на ринку відіграють відкриті платформи та рішення з відкритим вихідним кодом, такі як Port-IO [8] чи OpenTOS. Ці продукти дають можливість налаштовувати систему під індивідуальні потреби користувачів, що знижує витрати на придбання. Водночас, для їх ефективного використання потрібні висококваліфіковані розробники, а також додаткові ресурси для інтеграції з іншими системами.

Аналіз існуючих рішень дозволяє зробити висновок, що жодна з представлених систем не є універсальною. Наприклад, для портів із високою завантаженістю та різноманітними типами вантажів необхідні комплексні рішення, тоді як невеликі термінали часто потребують бюджетних інструментів із базовою функціональністю. Додатково, більшість сучасних систем приділяє недостатню увагу забезпеченню прозорості та інтеграції з державними регуляторними органами, що є важливим аспектом у багатьох країнах.

Світові тенденції свідчать про зростаючу роль технологій штучного інтелекту, машинного навчання та великих даних у розвитку систем для управління морськими терміналами. Прогнозується, що впровадження цих технологій дозволить створити більш адаптивні та прогнозовані системи управління, які будуть автоматично реагувати на зміни у вантажопотоках та попиті на логістичні послуги.

Таким чином, аналіз існуючих програмних продуктів підкреслює необхідність розробки інноваційних рішень, які б враховували специфіку роботи морських вантажних терміналів, забезпечували високу ефективність і були доступними для широкого кола користувачів. Запропонована у курсовій роботі база даних орієнтована на вирішення таких завдань і може стати основою для створення ефективної інформаційної системи.

2 ПОСТАНОВКА ЗАВДАННЯ

У межах даної курсової роботи було поставлено завдання розробити базу даних для морського вантажного терміналу, яка б забезпечувала ефективне управління основними бізнес-процесами. Основною метою є створення інформаційної системи, здатної автоматизувати облік вантажів, планування розкладу кораблів, управління зберіганням та моніторинг роботи терміналів, з урахуванням реальних умов експлуатації морських портів.

Усі розроблені рішення базуються на сучасних можливостях СУБД PostgreSQL [9], що гарантує їх стабільність, масштабованість і зручність у використанні.

Задачі роботи:

- 1) аналіз предметного середовища;
- 2) побудова ER-моделі, що відповідає потребам приймальної комісії університету;
- 3) побудова реляційної схеми на підставі розробленої ER-моделі;
- 4) створення бази даних, що була спроектована, з використання СУБД PostgreSQL;
- 5) імпорт даних з використанням засобів СУБД в створену базу даних;
- 6) з використанням мови SQL автоматизація ключових бізнес-процесів;
- 7) оптимізація роботи запитів;
- 8) підтримка багатокористувальницького доступу.

База даних також має включати швидке надання та безпеку даних що в ній зберігаються для всіх користувачів. Також має враховувати надання подальшої статистики по товарам, їх цінам, продажам, попиту в часових рамках.

3 ПОБУДОВА ER-МОДЕЛІ

3.1 Бізнес-правила

До основних бізнес-правил належить:

- 1) Кожен порт має унікальну назву, що гарантує однозначну ідентифікацію в системі.
- 2) Кожен термінал обов'язково належить одному порту, а його місткість має бути більшою за нуль, що дозволяє уникнути некоректних даних.
- 3) Клієнти класифікуються за типами: "Фізична особа", "Корпоративний замовник", "Урядовий замовник", що дає змогу ефективно управляти їх потребами.
- 4) Вантажі мають чітко визначені типи: "Контейнерний", "Небезпечний", "Сипучий", "Рідкий" тощо. Це дозволяє враховувати специфіку обробки різних типів вантажів.
- 5) Якщо тип вантажу позначено як "Dangerous" (Небезпечний), його вага не повинна перевищувати 50 тонн.
- 6) Довжина і осадка корабля перед внесенням до розкладу перевіряються на відповідність довжині та глибині причалу, щоб уникнути логістичних проблем.
- 7) Один причал може обслуговувати лише один корабель у визначений проміжок часу, що виключає конфлікти у розкладі.
- 8) Дата відправлення корабля у розкладі має бути пізнішою за дату прибуття, що забезпечує логічність та послідовність операцій.
- 9) Загальна вага вантажів, що зберігаються в терміналі, не може перевищувати його місткість, що дозволяє уникнути перевантаження інфраструктури.
- 10) Початок зберігання вантажу в терміналі повинен бути після його доставки, щоб забезпечити правильний порядок операцій.
- 11) У кожному терміналі встановлено максимальну кількість активних працівників (20 осіб), щоб забезпечити ефективну організацію роботи.

- 12) Усі зміни в даних про співробітників, зокрема їх позиції, статус активності або термінал роботи, мають бути прозорими та логуватись.
- 13) Для вантажів типу 'Perishable' (швидкопсувні) обов'язково має бути вказана дата завершення.
- 14) Для вантажів зі статусом "В очікуванні" або "Скасовано" дата відправлення може бути відсутньою, що відповідає бізнес-логіці.
- 15) Для вантажів зі статусом "В процесі" або "Доставлено" обов'язковим є вказання дати відправлення, що гарантує точність даних.
- 16) Причал може належати лише одному терміналу, і цей термінал повинен входити до складу порту, що забезпечує ієрархічну структуру даних.
- 17) Вантажі можуть бути відправлені лише через причали, що відповідають характеристикам кораблів, які їх обслуговують (довжина, глибина).
- 18) У розкладі кораблів заборонено перетин часових інтервалів прибуття та відправлення на одному причалі, що виключає дублювання.
- 19) Кожен транспортний засіб може перевозити лише той вантаж, вага якого не перевищує його місткість, щоб забезпечити безпеку транспортування.
- 20) У розкладі кораблів кожна пара "корабель - дата прибуття" повинна бути унікальною, щоб запобігти дублюванню записів і забезпечити коректність планування.
- 21) Усі дати, пов'язані з доставкою вантажів, мають бути перевірені на відповідність послідовності: дата відправлення повинна передувати даті початку зберігання.
- 22) Сума ваги всіх вантажів, які зберігаються в терміналі, не може перевищувати його загальну місткість.

3.2 Вибір сутностей

Відповідно до поставленого завдання та предметного середовища було виділено наступні сутності:

- 1) Ports

- 2) Terminals
- 3) Clients
- 4) Cargo
- 5) Ships
- 6) Berths
- 7) Schedules
- 8) Transporters
- 9) Customer
- 10) Shipments
- 11) Employees

3.3 Опис сутностей

Після детального аналізу опису предметного середовища були виділені наступні атрибути сутностей (таблиця 3.1).

Таблиця 3.1 – Опис сутностей

Назва сутності	Атрибути	Опис атрибута
Ports	PortID	Унікальний ідентифікатор порту
	PortName	Назва порту
Terminals	TerminalID	Унікальний ідентифікатор терміналу
	Name	Назва терміналу
	PortID	Ідентифікатор порту, до якого належить термінал
	Capacity	Місткість терміналу (у тоннах)

Продовження Таблиці 3.1

Назва сутності	Атрибути	Опис атрибута
Clients	ClientID	Унікальний ідентифікатор клієнта
	Name	Ім'я клієнта
	PhoneNumber	Контактний номер клієнта
	Type	Тип клієнта (Індивідуальний, Корпоративний тощо)
Cargo	CargoID	Унікальний ідентифікатор вантажу.
	Weight	Вага вантажу (у тоннах)
	CargoType	Тип вантажу (Контейнерний, Небезпечний тощо)
	ClientID	Ідентифікатор клієнта, який є власником вантажу
Ships	ShipID	Унікальний ідентифікатор корабля
	Name	Назва корабля
	Flag	Прапор країни реєстрації корабля
	Capacity	Місткість корабля (у тоннах)
	ShipType	Тип корабля (Танкер, Контейнеровоз тощо)

Продовження Таблиці 3.1

Назва сутності	Атрибути	Опис атрибута
	RequiredLength	Необхідна довжина причалу для корабля (у метрах)
	Draft	Осадка корабля (у метрах)
Berths	BerthID	Унікальний ідентифікатор причалу
	TerminalID	Ідентифікатор терміналу, до якого належить причал
	Length	Довжина причалу (у метрах)
	Depth	Глибина причалу (у метрах)
	Status	Статус причалу (Доступний, Зайнятий тощо)
Schedules	ScheduleID	Унікальний ідентифікатор розкладу
	ShipID	Ідентифікатор корабля
	TerminalID	Ідентифікатор терміналу
	BerthID	Ідентифікатор причалу
	DestinationPortID	Ідентифікатор порту призначення
	ArrivalDate	Дата прибуття
	DepartureDate	Дата відправлення.

Продовження Таблиці 3.1

Назва сутності	Атрибути	Опис атрибута
Storage	StorageID	Унікальний ідентифікатор складу
	TerminalID	Ідентифікатор терміналу, де зберігається вантаж
	CargoID	Ідентифікатор вантажу
	StartDate	Дата початку зберігання
	EndDate	Дата завершення зберігання
Transporters	TransporterID	Унікальний ідентифікатор транспортера
	Name	Назва транспортної компанії
	VehicleType	Тип транспортного засобу (Вантажівка, Літак тощо)
	Capacity	Місткість транспортного засобу (у тоннах)
Shipments	ShipmentID	Унікальний ідентифікатор доставки
	CargoID	Ідентифікатор вантажу
	TransporterID	Ідентифікатор транспортера
	ShipmentDate	Дата відправлення

Продовження Таблиці 3.1

Назва сутності	Атрибути	Опис атрибута
	Status	Статус доставки (В очікуванні, Доставлено тощо)
Employees	EmployeeID	Унікальний ідентифікатор працівника
	Name	Ім'я працівника
	Position	Посада працівника
	TerminalID	Ідентифікатор терміналу, де працює працівник
	IsActive	Статус активності працівника

Опис зв'язків між сутностями:

- 1) Ports – Terminals: один до багатьох (1:N) – порт може мати кілька терміналів, кожен термінал прив'язаний до одного порту.
- 2) Terminals – Berths: один до багатьох (1:N) – термінал може включати кілька причалів, причал належить лише одному терміналу.
- 3) Terminals – Employees: один до багатьох (1:N) – термінал має персонал, працівник працює в одному терміналі.
- 4) Clients – Cargo: один до багатьох (1:N) – клієнт може володіти кількома вантажами, кожен вантаж має одного клієнта.
- 5) Cargo – Shipments: один до багатьох (1:N) – вантаж може мати кілька доставок, кожна доставка пов'язана з одним вантажем.
- 6) Transporters – Shipments: один до багатьох (1:N) – транспортер може здійснювати кілька доставок, кожна доставка виконується одним транспортером.

7) Terminals – Storage: один до багатьох (1:N) – термінал може зберігати кілька вантажів, кожен вантаж прив’язаний до одного терміналу для зберігання.

8) Cargo – Storage: один до одного (1:1) – кожен вантаж має лише одне місце зберігання, і кожне місце зберігання зберігає один вантаж.

9) Ships – Schedules: один до багатьох (1:N) – корабель може мати кілька розкладів, кожен розклад належить до одного корабля.

10) Terminals – Schedules: один до багатьох (1:N) – термінал може мати кілька розкладів, кожен розклад прив’язаний до одного терміналу.

11) Berths – Schedules: один до багатьох (1:N) – причал може бути частиною кількох розкладів, кожен розклад пов’язаний з одним причалом.

12) Ports – Schedules: один до багатьох (1:N) – порт призначення може мати кілька розкладів, кожен розклад має один порт призначення.

Побудована ER-модель зображена на рис. 3.1.

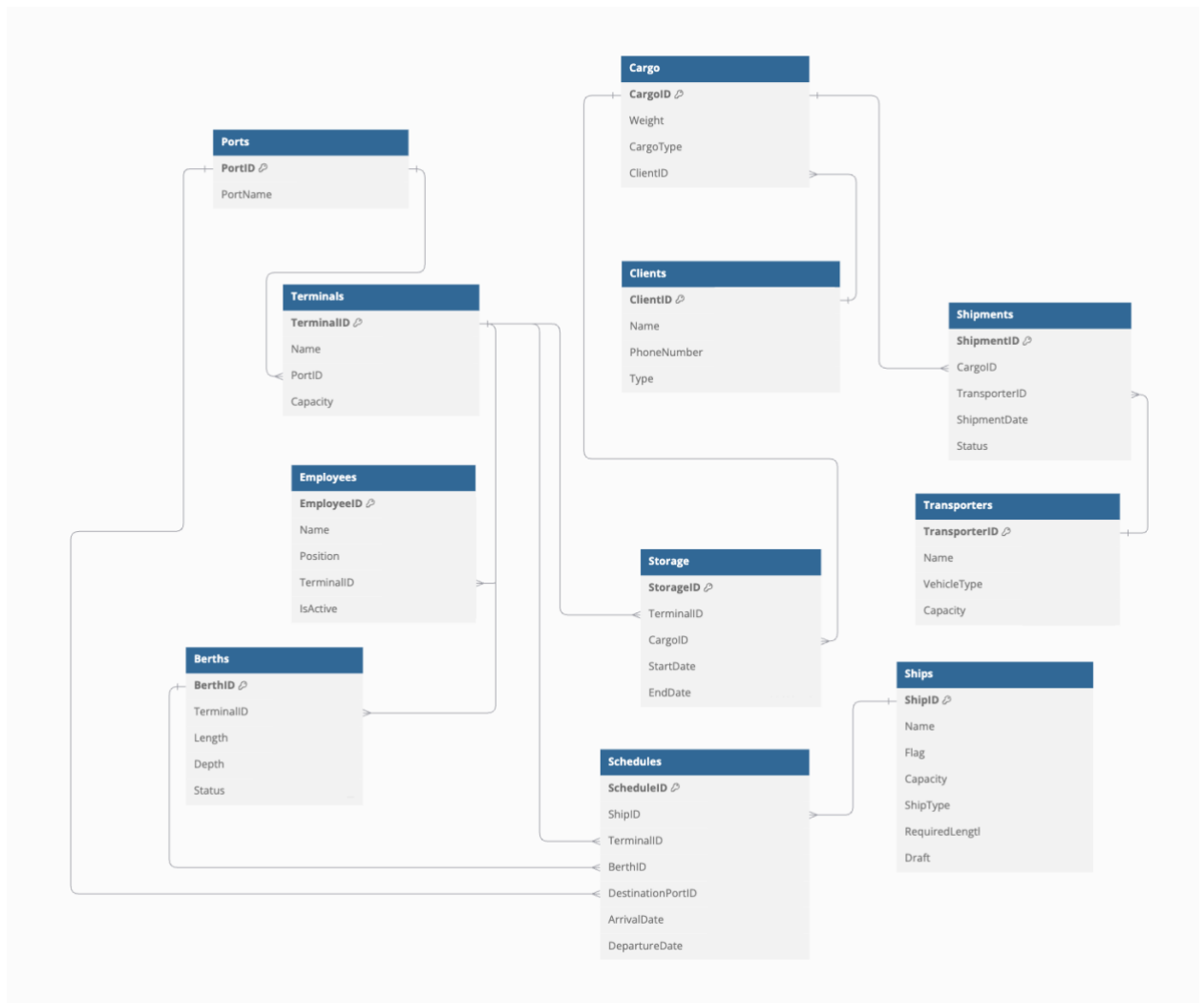


Рис. 3.1 – ER-модель бази даних складського обліку підприємства

Отже, в цьому розділі були сформовані бізнес правила та обрані сутності для подальшої побудови бази для морського вантажного терміналу. Також була побудована ER-модель бази даних.

4 РЕАЛЯЦІЙНА МОДЕЛЬ БАЗИ ДАНИХ

Розробка реляційної моделі бази даних для морського вантажного терміналу є ключовим етапом у забезпеченні автоматизації бізнес-процесів, підвищенні ефективності роботи підприємства та точності обліку інформації. У цьому розділі описуються основні сутності, атрибути, взаємозв'язки та особливості проектування бази даних, які були враховані під час її створення.

Для виконання завдань курсової роботи було обрано систему управління базами даних PostgreSQL. Це рішення забезпечує високу функціональність і продуктивність завдяки підтримці складних запитів, транзакцій, обмежень цілісності даних, зберігання великих обсягів інформації та роботі зі складними типами даних. Розширюваність PostgreSQL дозволяє створювати власні функції, типи даних і методи індексації, що особливо важливо для специфічних потреб морського терміналу. Завдяки багатоплатформеності PostgreSQL може бути розгорнуто в будь-якому операційному середовищі. Система забезпечує високий рівень безпеки через механізми аутентифікації, управління ролями та контроль доступу до даних. Велика спільнота користувачів і розробників сприяє постійній підтримці, доступу до документації та додаткових навчальних матеріалів.

Після аналізу предметного середовища було виділено ключові сутності, атрибути та взаємозв'язки. Ці сутності включають Ports, Terminals, Clients, Cargo, Ships, Berths, Schedules, Storage, Transporters, Shipments та Employees. Було визначено всі необхідні атрибути, такі як унікальні ідентифікатори, ключові характеристики та властивості, які відображають основні аспекти бізнес-процесів морського терміналу. Визначені взаємозв'язки між сутностями враховують як обов'язкові зв'язки, так і додаткові обмеження, необхідні для забезпечення цілісності даних.

Особливу увагу було приділено реалізації бізнес-правил, таких як обмеження цілісності, забезпечення відповідності вантажів параметрам терміналів і причалів, а також перевірка допустимості дат для складування та відправлення. Для цього в базі даних впроваджено використання первинних і

зовнішніх ключів, додаткових обмежень і тригерів, які автоматизують перевірку бізнес-логіки.

Реляційна модель розроблена з урахуванням вимог масштабованості, високої продуктивності та надійності. Усі проєктовані таблиці оптимізовані для швидкого виконання запитів, забезпечення простоти інтеграції та підтримки в багатокористувацькому середовищі. У результаті було створено систему, яка відповідає потребам морського вантажного терміналу та забезпечує ефективну обробку інформації. Результати моделювання представлені в таблицях 4.1 – 4.11.

Таблиця 4.1 – Сутність "Ports" створена для репрезентації портів. Структура таблиці наступна:

Ім'я поля	Тип даних	Розмір	Ключ	Опис
portid	SERIAL		Первинний	Унікальний ідентифікатор порту
portname	VARCHAR	50	Унікальний	Назва порту

Таблиця 4.2 – Сутність "Terminals" створена для репрезентації терміналів. Структура таблиці наступна:

Ім'я поля	Тип даних	Розмір	Ключ	Опис
terminalid	SERIAL		Первинний	Унікальний ідентифікатор терміналу
name	VARCHAR	50	Унікальний	Назва терміналу
portid	INT		Зовнішній	Посилання на ідентифікатор порту

Продовження Таблиці 4.2

Ім'я поля	Тип даних	Розмір	Ключ	Опис
capacity	DECIMAL	10,2		Максимальна місткість терміналу

Таблиця 4.3 – Сутність "Clients" створена для репрезентації клієнтів. Структура таблиці наступна:

Ім'я поля	Тип даних	Розмір	Ключ	Опис
clientid	SERIAL		Первинний	Унікальний ідентифікатор клієнта
name	VARCHAR	50		Ім'я клієнта
phonenummer	VARCHAR	20	Унікальний	Номер телефону клієнта
type	VARCHAR	50		Тип клієнта

Таблиця 4.4 – Сутність "Cargo" створена для репрезентації вантажів. Структура таблиці наступна:

Ім'я поля	Тип даних	Розмір	Ключ	Опис
cargoid	SERIAL		Первинний	Унікальний ідентифікатор вантажу
weight	DECIMAL	10,2		Вага вантажу
cargotype	VARCHAR	50		Тип вантажу
clientid	INT		Зовнішній	Посилання на ідентифікатор клієнта

Таблиця 4.5 – Сутність "Ships" створена для репрезентації кораблів.

Структура таблиці наступна:

Ім'я поля	Тип даних	Розмір	Ключ	Опис
shipid	SERIAL		Первинний	Унікальний ідентифікатор корабля
name	VARCHAR	50	Унікальний	Назва корабля
flag	VARCHAR	3		Прапор країни
capacity	DECIMAL	10,2		Максимальна вантажопідйомність
shiptype	VARCHAR	50		Тип корабля
requiredlength	DECIMAL	8,2		Мінімальна довжина причалу для змоги пришвартуватись
draft	DECIMAL	5,2		Мінімальна глибина причалу

Таблиця 4.6 – Сутність "Berths" створена для репрезентації причалів.

Структура таблиці наступна:

Ім'я поля	Тип даних	Розмір	Ключ	Опис
berthid	SERIAL		Первинний	Унікальний ідентифікатор причалу
terminalid	INT		Зовнішній	Посилання на ідентифікатор терміналу
length	DECIMAL	8,2		Довжина причалу

Продовження Таблиці 4.6

Ім'я поля	Тип даних	Розмір	Ключ	Опис
depth	DECIMAL	5,2		Глибина причалу
status	VARCHAR	20		Статус причалу

Таблиця 4.7 – Сутність "Schedules" створена для репрезентації розкладу кораблів. Структура таблиці наступна:

Ім'я поля	Тип даних	Розмір	Ключ	Опис
scheduleid	SERIAL		Первинний	Унікальний ідентифікатор розкладу
shipid	INT		Зовнішній	Посилання на ідентифікатор корабля
terminalid	INT		Зовнішній	Посилання на ідентифікатор терміналу
berthid			Зовнішній	Посилання на ідентифікатор причалу
arrivaldate	TIMESTAMP			Дата і час прибуття
departuredate	TIMESTAMP			Дата і час відправлення
destinationportid	INT		Зовнішній	Посилання на порт призначення

Таблиця 4.8 – Сутність "Storage" створена для репрезентації зберігання вантажів. Структура таблиці наступна:

Ім'я поля	Тип даних	Розмір	Ключ	Опис
storageid	SERIAL		Первинний	Унікальний ідентифікатор складу
terminalid	INT		Зовнішній	Посилання на ідентифікатор терміналу
cargoid	INT		Зовнішній	Посилання на ідентифікатор вантажу
startdate	TIMESTAMP			Дата початку зберігання
enddate	TIMESTAMP			Дата завершення зберігання

Таблиця 4.9 – Сутність "Transporters" створена для репрезентації транспортних засобів. Структура таблиці наступна:

Ім'я поля	Тип даних	Розмір	Ключ	Опис
transporterid	SERIAL		Первинний	Унікальний ідентифікатор транспортного засобу
name	VARCHAR	50	Унікальний	Назва транспортного засобу

Продовження Таблиці 4.9

Ім'я поля	Тип даних	Розмір	Ключ	Опис
vehicletype	VARCHAR	50		Тип транспортного засобу
capacity	DECIMAL	10,2		Вантажопідйомність транспортного засобу

Таблиця 4.10 – Сутність "Shipments" створена для репрезентації перевезень вантажів. Структура таблиці наступна:

Ім'я поля	Тип даних	Розмір	Ключ	Опис
shipmentid	SERIAL		Первинний	Унікальний ідентифікатор перевезення
cargoid	INT		Зовнішній	Посилання на ідентифікатор вантажу
transporterid	INT		Зовнішній	Посилання на ідентифікатор транспортного засобу
shipmentdate	TIMESTAMP			Дата перевезення
status	VARCHAR	50		Статус перевезення

Таблиця 4.11 – Сутність "Employees" створена для репрезентації співробітників. Структура таблиці наступна:

Ім'я поля	Тип даних	Розмір	Ключ	Опис
employeeid	SERIAL		Первинний	Унікальний ідентифікатор співробітника
name	VARCHAR	50		Ім'я співробітника
position	VARCHAR	50		Посада співробітника
terminalid	INT		Зовнішній	Посилання на ідентифікатор терміналу
isactive	BOOLEAN			Чи є співробітник активним

Відповідно до табличних описів було створено реляційну схему (рис. 4.1) в третій нормальній формі. Це досягнуто завдяки декомпозиції полів, відсутності зайвих даних, транзитивної незалежності атрибутів від первинного ключа та функціональної повної залежності атрибутів від первинного ключа.

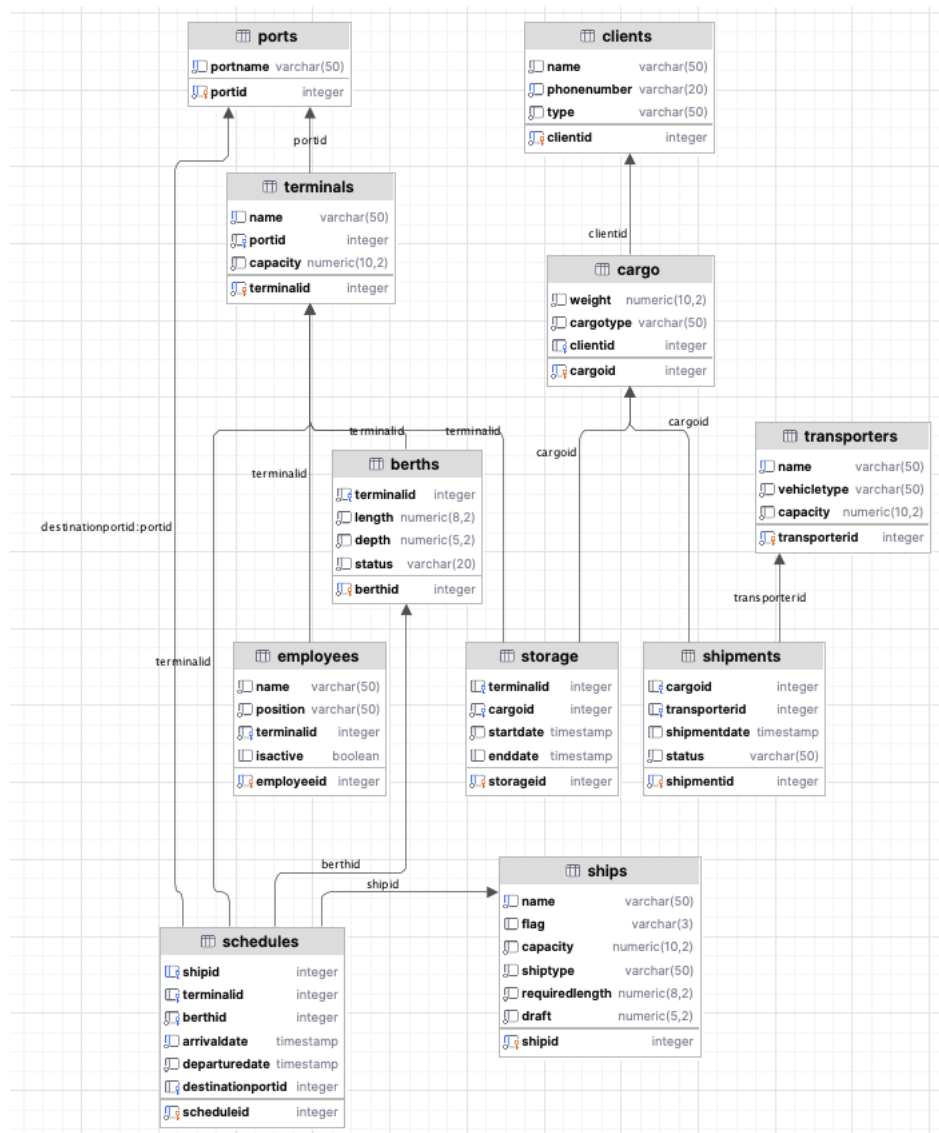


Рис. 4.1 – Реляційна схема бази даних реалізована засобами PostgreSQL,

У цьому розділі було наведено обґрунтування вибору PostgreSQL в якості СУБД, були побудовані необхідні відношення та обмеження, створена реляційна схема засобами PostgreSQL (рис. 4.1).

5 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

5.1 Створення бази даних

Результатом проектування бази даних є сформований SQL-скрипт, який використовується для створення об'єктів, які були наведені в ER-моделі та табличних описах:

```
CREATE TABLE Ports
```

```
(  
    PortID SERIAL PRIMARY KEY,  
    PortName VARCHAR(50) NOT NULL UNIQUE  
);
```

```
CREATE TABLE Terminals
```

```
(  
    TerminalID SERIAL PRIMARY KEY,  
    Name VARCHAR(50) NOT NULL UNIQUE,  
    PortID INT NOT NULL REFERENCES Ports (PortID) ON DELETE  
CASCADE,  
    Capacity DECIMAL(10, 2) NOT NULL CHECK (Capacity > 0)  
);
```

```
CREATE TABLE Clients
```

```
(  
    ClientID SERIAL PRIMARY KEY,  
    Name VARCHAR(50) NOT NULL,  
    PhoneNumber VARCHAR(20) NOT NULL UNIQUE CHECK (PhoneNumber  
~ '^\\d{7,20}$'),  
    Type VARCHAR(50) NOT NULL CHECK (Type IN ('Individual',  
'Corporate', 'Government'))  
);
```


CREATE TABLE Cargo

```
(
  CargoID SERIAL PRIMARY KEY,
  Weight DECIMAL(10, 2) NOT NULL CHECK (Weight > 0),
  CargoType VARCHAR(50) NOT NULL CHECK (CargoType IN
    ('General', 'Container', 'Bulk', 'Liquid', 'RoRo',
    'Perishable',
    'Dangerous', 'Project')),
  ClientID INT REFERENCES Clients (ClientID) ON DELETE SET
  NULL
);
```

CREATE TABLE Ships

```
(
  ShipID SERIAL PRIMARY KEY,
  Name VARCHAR(50) NOT NULL UNIQUE,
  Flag VARCHAR(3),
  Capacity DECIMAL(10, 2) NOT NULL CHECK (Capacity > 0),
  ShipType VARCHAR(50) NOT NULL CHECK (ShipType IN ('Cargo',
  'Tanker', 'Container', 'Bulk', 'RoRo')),
  RequiredLength DECIMAL(8, 2) NOT NULL CHECK (RequiredLength > 0),
  Draft DECIMAL(5, 2) NOT NULL CHECK (Draft > 0)
```

CREATE TABLE Berths

```
(
  BerthID SERIAL PRIMARY KEY,
  TerminalID INT NOT NULL REFERENCES Terminals (TerminalID)
  ON DELETE CASCADE,
  Length DECIMAL(8, 2) NOT NULL CHECK (Length > 0),
  Depth DECIMAL(5, 2) NOT NULL CHECK (Depth > 0),
  Status VARCHAR(20) NOT NULL CHECK (Status IN ('Available',
```

'Occupied', 'Under Maintenance'))

);

CREATE TABLE Schedules

(

ScheduleID SERIAL PRIMARY KEY,

ShipID INT REFERENCES Ships (ShipID) ON DELETE SET NULL,

TerminalID INT REFERENCES Terminals (TerminalID) ON DELETE
SET NULL,

BerthID INT NOT NULL REFERENCES Berths (BerthID),

ArrivalDate TIMESTAMP NOT NULL,

DepartureDate TIMESTAMP NOT NULL,

CHECK (DepartureDate > ArrivalDate),

UNIQUE (ShipID, ArrivalDate)

);

ALTER TABLE Schedules

ADD COLUMN DestinationPortID INT REFERENCES Ports(PortID);

CREATE TABLE Storage

(

StorageID SERIAL PRIMARY KEY,

TerminalID INT REFERENCES Terminals (TerminalID) ON DELETE SET
NULL,

CargoID INT NOT NULL REFERENCES Cargo (CargoID) ON DELETE
CASCADE,

StartDate TIMESTAMP NOT NULL,

EndDate TIMESTAMP,

CHECK (EndDate IS NULL OR EndDate > StartDate)

);

```
CREATE TABLE Transporters
```

```
(
  TransporterID SERIAL PRIMARY KEY,
  Name          VARCHAR(50)  NOT NULL UNIQUE,
  VehicleType   VARCHAR(50)  NOT NULL CHECK (VehicleType IN
('Truck', 'Train', 'Plane', 'Ship', 'Van')),
  Capacity      DECIMAL(10, 2) NOT NULL CHECK (Capacity > 0)
);
```

```
CREATE TABLE Shipments
```

```
(
  ShipmentID   SERIAL PRIMARY KEY,
  CargoID      INT          REFERENCES Cargo (CargoID) ON DELETE SET
NULL,
  TransporterID INT          REFERENCES Transporters (TransporterID) ON
DELETE SET NULL,
  ShipmentDate TIMESTAMP NOT NULL,
  Status       VARCHAR(50) NOT NULL CHECK (Status IN ('Pending', 'In
Progress', 'Delivered', 'Canceled'))
);
```

```
ALTER TABLE Shipments
```

```
  ALTER COLUMN ShipmentDate DROP NOT NULL;
```

```
ALTER TABLE Shipments
```

```
  ADD CONSTRAINT check_shipment_status_date
```

```
  CHECK (
```

```
    (Status = 'Pending' AND ShipmentDate IS NULL) OR
```

```
    (Status = 'Canceled' AND ShipmentDate IS NULL) OR
```

```
    (Status IN ('In Progress', 'Delivered') AND ShipmentDate IS NOT NULL)
```

```
  );
```

```
alter table shipments
```

```
add constraint check_shipment_date_valid
```

```
check (shipmentdate >= (select startdate from storage where storage.cargoid =
shipments.cargoid));
```

```
CREATE TABLE Employees
```

```
(
```

```
EmployeeID SERIAL PRIMARY KEY,
```

```
Name VARCHAR(50) NOT NULL,
```

```
Position VARCHAR(50) NOT NULL CHECK (Position IN ('Manager',
'Supervisor', 'Worker', 'Admin')),
```

```
TerminalID INT NOT NULL REFERENCES Terminals (TerminalID) ON
DELETE CASCADE,
```

```
IsActive BOOLEAN DEFAULT TRUE
```

```
);
```

6 СТВОРЕННЯ КОРИСТУВАЧІВ БАЗИ ДАНИХ

6.1 Адміністратор

```
CREATE ROLE admin WITH
```

```
LOGIN
```

```
SUPERUSER
```

```
CREATEDB
```

```
CREATE ROLE  
PASSWORD 'admin_password';
```

6.2 Менеджер

```
CREATE ROLE manager WITH
```

```
LOGIN
```

```
NOSUPERUSER
```

```
NOCREATEDB
```

```
NOCREATEROLE
```

```
PASSWORD 'manager_password';
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE schedules TO  
manager;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE cargo TO manager;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE employees TO  
manager;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE shipments TO  
manager;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE clients TO  
manager;
```

```
GRANT SELECT ON TABLE terminals TO manager;
```

```
GRANT SELECT ON TABLE berths TO manager;
```

```
GRANT SELECT ON TABLE transporters TO manager;
```

```
GRANT USAGE, SELECT ON SEQUENCE schedules_scheduleid_seq TO  
manager;
```

```
GRANT USAGE, SELECT ON SEQUENCE cargo_cargoid_seq TO manager;
```

```
GRANT USAGE, SELECT ON SEQUENCE employees_employeeid_seq TO  
manager;
```

```
GRANT USAGE, SELECT ON SEQUENCE shipments_shipmentid_seq TO
```

```
manager;  
GRANT USAGE, SELECT ON SEQUENCE clients_clientid_seq TO manager;  
;
```

6.3 Оператор

```
CREATE ROLE operator WITH  
    LOGIN  
    NOSUPERUSER  
    NOCREATEDB  
    NOCREATEROLE  
    PASSWORD 'operator_password';
```

```
GRANT SELECT, INSERT, UPDATE ON TABLE schedules TO operator;  
GRANT SELECT, INSERT, UPDATE ON TABLE cargo TO operator;
```

```
GRANT SELECT ON TABLE ships TO operator;  
GRANT SELECT ON TABLE terminals TO operator;  
GRANT SELECT ON TABLE berths TO operator;  
GRANT SELECT ON TABLE clients TO operator;
```

```
GRANT USAGE, SELECT ON SEQUENCE schedules_scheduleid_seq TO  
operator;  
GRANT USAGE, SELECT ON SEQUENCE cargo_cargoid_seq TO operator;
```

6.4 Аналітик

```
CREATE ROLE analyst WITH  
    LOGIN  
    NOSUPERUSER  
    NOCREATEDB  
    NOCREATEROLE  
    PASSWORD 'analyst_password';
```

```
GRANT SELECT ON TABLE schedules TO analyst;  
GRANT SELECT ON TABLE cargo TO analyst;  
GRANT SELECT ON TABLE clients TO analyst;  
GRANT SELECT ON TABLE ships TO analyst;  
GRANT SELECT ON TABLE terminals TO analyst;  
GRANT SELECT ON TABLE berths TO analyst;  
GRANT SELECT ON TABLE transporters TO analyst;  
GRANT SELECT ON TABLE employees TO analyst;
```

6.5 Керівник

```
CREATE ROLE supervisor WITH  
    LOGIN  
    NOSUPERUSER  
    NOCREATEDB  
    NOCREATEROLE  
    PASSWORD 'supervisor_password';
```

```
GRANT SELECT, UPDATE ON TABLE schedules TO supervisor;  
GRANT SELECT, UPDATE ON TABLE cargo TO supervisor;  
GRANT SELECT, UPDATE ON TABLE employees TO supervisor;  
GRANT SELECT ON TABLE ships TO supervisor;  
GRANT SELECT ON TABLE terminals TO supervisor;  
GRANT SELECT ON TABLE berths TO supervisor;
```

6.6 Технік

```
CREATE ROLE technician WITH  
    LOGIN  
    NOSUPERUSER  
    NOCREATEDB  
    NOCREATEROLE
```

```
PASSWORD 'technician_password';
```

```
GRANT SELECT, UPDATE ON TABLE berths TO technician;
```

```
GRANT SELECT ON TABLE terminals TO technician;
```

```
GRANT SELECT ON TABLE ports TO technician;
```

6.7 Оператор дока

```
CREATE ROLE dock_operator WITH
```

```
LOGIN
```

```
NOSUPERUSER
```

```
NOCREATEDB
```

```
NOCREATEROLE
```

```
PASSWORD 'dock_password';
```

```
GRANT SELECT, UPDATE ON TABLE berths TO dock_operator;
```

```
GRANT SELECT ON TABLE terminals TO dock_operator;
```


6.8 Клієнт

CREATE ROLE client WITH

LOGIN

NOSUPERUSER

NOCREATEDB

NOCREATEROLE

PASSWORD 'client_password';

GRANT SELECT ON TABLE cargo TO client;

GRANT SELECT ON TABLE shipments TO client;

GRANT SELECT ON TABLE transporters TO client;

Результат виконання команд для створення користувачів та їх привілеїв командами SQL на рис. 6.1 та 6.2.

	grantee name	privilege_type character varying	table_schema name	table_name name
1	analyst	SELECT	public	schedules
2	analyst	SELECT	public	berths
3	analyst	SELECT	public	terminals
4	analyst	SELECT	public	transporters
5	analyst	SELECT	public	employees
6	analyst	SELECT	public	cargo
7	analyst	SELECT	public	ships
8	analyst	SELECT	public	clients
9	client	SELECT	public	transporters
10	client	SELECT	public	cargo
11	client	SELECT	public	shipments
12	dock_operator	SELECT	public	terminals
13	dock_operator	SELECT	public	berths
14	dock_operator	UPDATE	public	berths
15	manager	DELETE	public	schedules
16	manager	DELETE	public	clients
17	manager	DELETE	public	cargo
18	manager	DELETE	public	shipments
19	manager	DELETE	public	employees
20	manager	INSERT	public	clients
21	manager	INSERT	public	cargo
22	manager	INSERT	public	employees
23	manager	INSERT	public	shipments
24	manager	INSERT	public	schedules
25	manager	SELECT	public	berths
26	manager	SELECT	public	terminals
27	manager	SELECT	public	cargo
28	manager	SELECT	public	transporters
29	manager	SELECT	public	shipments

Рис. 6.1 – Створені користувачі та їх привілеїв в базі даних

30	manager	SELECT	public	clients
31	manager	SELECT	public	schedules
32	manager	SELECT	public	employees
33	manager	UPDATE	public	shipments
34	manager	UPDATE	public	employees
35	manager	UPDATE	public	clients
36	manager	UPDATE	public	schedules
37	manager	UPDATE	public	cargo
38	operator	INSERT	public	schedules
39	operator	INSERT	public	cargo
40	operator	SELECT	public	clients
41	operator	SELECT	public	cargo
42	operator	SELECT	public	ships
43	operator	SELECT	public	schedules
44	operator	SELECT	public	terminals
45	operator	SELECT	public	berths
46	operator	UPDATE	public	schedules
47	operator	UPDATE	public	cargo
48	supervisor	SELECT	public	employees
49	supervisor	SELECT	public	cargo
50	supervisor	SELECT	public	ships
51	supervisor	SELECT	public	berths
52	supervisor	SELECT	public	terminals
53	supervisor	SELECT	public	schedules
54	supervisor	UPDATE	public	employees
55	supervisor	UPDATE	public	schedules
56	supervisor	UPDATE	public	cargo
57	technician	SELECT	public	terminals
58	technician	SELECT	public	ports
59	technician	SELECT	public	berths
60	technician	UPDATE	public	berths
Total rows: 60 of 60 Query complete 00:00:00.110 Ln 10, Col 1				

Рис. 6.2 – Створені користувачі та їх привілеїв в базі даних

7 РОБОТА З БАЗОЮ ДАНИХ

7.1 Тригери

Відповідно до моїх бізнес правил були розроблені наступні тригери на таблицях.

7.1.1 Тригер check_shipment_storage_date

Тригер check_shipment_storage_date за допомогою функції validate_shipment_and_storage_date() забезпечує перевірку відповідності дати відправлення вантажу (shipmentdate) даті початку його зберігання (startdate). Цей механізм гарантує, що дата відправлення вантажу не може перевищувати дату початку його зберігання.

Приклад роботи показано на рис. 7.1.1.1 та 7.1.1.2.

```
create or replace function validate_shipment_and_storage_date()
  returns trigger as $$
declare
  v_startdate timestamp;
begin
  select startdate
  into v_startdate
  from storage
  where storage.cargoid = new.cargoid;

  if v_startdate is not null and new.shipmentdate > v_startdate then
    raise exception 'shipmentdate % cannot be later than storage startdate % for
cargoid %',
      new.shipmentdate, v_startdate, new.cargoid;
  end if;

  return new;
```

end;

\$\$ language plpgsql;

create trigger check_shipment_storage_date

before insert or update on shipments

for each row

execute function validate_shipment_and_storage_date();

```
seaport.public> INSERT INTO shipments (cargoid, transporterid, shipmentdate, status)
VALUES (1, 1, '2024-12-25 10:00:00', 'In Progress')
[2024-12-21 01:52:56] [P0001] ERROR: shipmentdate 2024-12-25 10:00:00 cannot be later than storage startdate 2024-06-01 00:00:00 for cargoid 1
[2024-12-21 01:52:56] Where: PL/pgSQL function validate_shipment_and_storage_date() line 11 at RAISE
```

Рис. 7.1.1.1 – Спроба імпортувати дані, де дата надсилання вантажу більша, ніж дата, встановлена при отриманні вантажу на склад

```
seaport.public> UPDATE shipments
SET shipmentdate = '2024-12-20 10:00:00'
WHERE shipmentid = 1
[2024-12-21 01:56:21] [P0001] ERROR: shipmentdate 2024-12-20 10:00:00 cannot be later than storage startdate 2024-11-02 00:00:00 for cargoid 618
[2024-12-21 01:56:21] Where: PL/pgSQL function validate_shipment_and_storage_date() line 11 at RAISE
```

Рис. 7.1.1.2 – Спроба оновити дані так, щоб дата надсилання вантажу була більша, ніж дата, встановлена при отриманні вантажу на склад

7.1.2 Тригер before_insert_employee

Тригер before_insert_employee працює у парі з функцією check_employee_limit, щоб забезпечити виконання бізнес-правила обмеження кількості активних працівників у кожному терміналі.

Приклад роботи показано на рис. 7.1.2.1 та рис. 7.1.2.2.

create or replace function check_employee_limit()

returns trigger as \$\$

declare

v_max_limit int := 20;

v_current_count int;

begin

select count(employeeid)

```

into v_current_count
from employees
where terminalid = new.terminalid and isactive = true;

if v_current_count >= v_max_limit then
    raise exception 'terminal % already has the maximum allowed active
employees (%).', new.terminalid, v_max_limit;
end if;

return new;
end;
$$ language plpgsql;

```

```

create trigger before_insert_employee
before insert on employees
for each row
execute function check_employee_limit();

```

	terminalid ▾	÷	total_employees ▾	÷
1		202		20
2		121		18
3		26		18

Рис. 7.1.2.1 – Додаткова демонстрація, що термінал з ідентифікатором 202 вже має 20 працівників

```

seaport.public> insert into employees (name, position, terminalid, isactive)
values ('Paul Lee', 'Worker', 202, true)
[2024-12-21 02:16:31] [P0001] ERROR: terminal 202 already has the maximum allowed active employees (20).
[2024-12-21 02:16:31] Where: PL/pgSQL function check_employee_limit() line 13 at RAISE

```

Рис. 7.1.2.2 – Спроба додати працівника до терміналу, який вже має 20 активних працівників

7.1.3 Тригер `trg_check_dangerous_cargo_weight`

Тригер `trg_check_dangerous_cargo_weight` за допомогою функції `check_dangerous_cargo_weight` контролює вагу небезпечних вантажів у таблиці `cargo`. Він перевіряє, щоб вага вантажів типу "Dangerous" не перевищувала 50 тонн. Це забезпечує дотримання безпечних умов для обробки небезпечних матеріалів у терміналах.

Приклад роботи показано на рис. 7.1.3.1 та рис. 7.1.3.2.

```
CREATE OR REPLACE FUNCTION check_dangerous_cargo_weight()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.cargotype = 'Dangerous' AND NEW.weight > 50 THEN
        RAISE EXCEPTION 'Dangerous cargo weight % exceeds the allowed limit
of 50 tons', NEW.weight;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trg_check_dangerous_cargo_weight
BEFORE INSERT OR UPDATE ON cargo
FOR EACH ROW
EXECUTE FUNCTION check_dangerous_cargo_weight();
```

```
seaport.public> INSERT INTO cargo (weight, cargotype, clientid)
VALUES (60, 'Dangerous', 1)
[2024-12-21 02:51:10] [P0001] ERROR: Dangerous cargo weight 60.00 exceeds the allowed limit of 50 tons
[2024-12-21 02:51:10] Where: PL/pgSQL function check_dangerous_cargo_weight() line 4 at RAISE
```

Рис. 7.1.3.1 – Спроба присвоїти клієнту небезпечний вантаж вагою більше за 50 тонн

```
seaport.public> INSERT INTO cargo (weight, cargotype, clientid)
VALUES (40, 'Dangerous', 1)
[2024-12-21 02:59:53] 1 row affected in 8 ms
```

Рис. 7.1.3.2 – Успішне додавання небезпечного вантажу допустимої ваги

7.1.4 Тригер `trg_check_schedule_constraints`

Тригер `trg_check_schedule_constraints` забезпечує цілісність і логічну коректність даних у таблиці `schedules`. Його функція полягає в перевірці коректності даних перед додаванням або оновленням записів у таблиці. Спершу перевіряється, чи дата відправлення корабля (`departuredate`) є пізнішою за дату його прибуття (`arrivaldate`). Далі тригер переконується, що причал, зазначений у полі `berthid`, існує в таблиці `berths`, і що він належить до вказаного терміналу (`terminalid`). Перевіряється також, чи порт призначення (`destinationportid`) існує в таблиці `ports`.

Особлива увага приділяється відповідності фізичних параметрів корабля і причалу: довжина корабля не повинна перевищувати довжину причалу, а осадка корабля має бути меншою за глибину причалу. Тригер також запобігає конфліктам у розкладі: він перевіряє, чи вказаний причал не зайнятий іншим кораблем у зазначений період, і чи корабель уже не має запису з такою самою датою прибуття.

Приклад роботи показано на рис. 7.1.4.1, 7.1.4.2 та рис. 7.1.4.3.

```
create or replace function check_schedule_constraints()
```

```
    returns trigger as $$
```

```
declare
```

```
    v_terminal_portid int;
```

```
    v_berth_terminalid int;
```

```
    v_required_length numeric;
```

```
    v_required_draft numeric;
```

```
    v_berth_length numeric;
```

```
    v_berth_depth numeric;
```

```
begin
```

```
    if new.departuredate <= new.arrivaldate then
```

```

        raise exception 'departure date % must be later than arrival date %',
new.departuredate, new.arrivaldate;
    end if;

```

```

select terminalid
into v_berth_terminalid
from berths
where berthid = new.berthid;

```

```

if not found then
    raise exception 'berth with id % does not exist', new.berthid;
end if;

```

```

select portid
into v_terminal_portid
from terminals
where terminalid = new.terminalid;

```

```

if v_terminal_portid is null then
    raise exception 'terminal with id % does not belong to any port',
new.terminalid;
end if;

```

```

if v_berth_terminalid != new.terminalid then
    raise exception 'berth % does not belong to terminal %', new.berthid,
new.terminalid;
end if;

```



```

if not exists (
    select 1
    from ports
    where portid = new.destinationportid
) then
    raise exception 'destination port with id % does not exist',
new.destinationportid;
end if;

select requiredlength, draft
into v_required_length, v_required_draft
from ships
where shipid = new.shipid;

select length, depth
into v_berth_length, v_berth_depth
from berths
where berthid = new.berthid;

if v_required_length > v_berth_length then
    raise exception 'berth % is too short for ship %', new.berthid, new.shipid;
end if;

if v_required_draft > v_berth_depth then
    raise exception 'berth % is too shallow for ship %', new.berthid, new.shipid;
end if;

if exists (
    select 1

```

```

from schedules
where berthid = new.berthid
and (
    (new.arrivaldate between arrivaldate and departuredate)
    or
    (new.departuredate between arrivaldate and departuredate)
)
and scheduleid != new.scheduleid
) then
    raise exception 'berth % is already occupied during the specified time',
new.berthid;
end if;

if exists (
    select 1
    from schedules
    where shipid = new.shipid
        and arrivaldate = new.arrivaldate
        and scheduleid != new.scheduleid
    ) then
    raise exception 'ship with id % already has a schedule at %', new.shipid,
new.arrivaldate;
end if;

return new;
end;
$$ language plpgsql;

create trigger trg_check_schedule_constraints
before insert or update on schedules

```

for each row
execute function check_schedule_constraints();

```
seaport.public> insert into schedules (shipid, terminalid, berthid, arrivaldate, departuredate, destinationportid)
values (2, 2, 3, '2024-12-20 10:00:00', '2024-12-20 09:00:00', 5)
[2024-12-21 03:05:51] [P0001] ERROR: departure date 2024-12-20 09:00:00 must be later than arrival date 2024-12-20 10:00:00
[2024-12-21 03:05:51] Where: PL/pgSQL function check_schedule_constraints() line 12 at RAISE
```

Рис. 7.1.4.1 – Спроба додати розклад, в якому дата відправлення менша, ніж вже визначена дата по прибутті корабля

```
seaport.public> insert into schedules (shipid, terminalid, berthid, arrivaldate, departuredate, destinationportid)
values (1, 2, 3, '2024-12-20 10:00:00', '2024-12-28 18:00:00', 5)
[2024-12-21 03:07:50] [P0001] ERROR: berth 3 does not belong to terminal 2
[2024-12-21 03:07:50] Where: PL/pgSQL function check_schedule_constraints() line 36 at RAISE
```

Рис. 7.1.4.2 – Спроба додати розклад, в терміналі не існує заданого причалу

```
seaport.public> insert into schedules (shipid, terminalid, berthid, arrivaldate, departuredate, destinationportid)
values (1, 2, 2, '2024-12-20 10:00:00', '2024-12-28 18:00:00', 5)
[2024-12-21 03:09:34] [P0001] ERROR: berth 2 is already occupied during the specified time
[2024-12-21 03:09:34] Where: PL/pgSQL function check_schedule_constraints() line 79 at RAISE
```

Рис. 7.1.4.2 – Спроба додати розклад на період, коли вказаний причал вже зайнятий

7.1.5 Тригер trg_validate_perishable_cargo

Тригер trg_validate_perishable_cargo забезпечує перевірку коректності збереження даних для швидкопсувних вантажів у таблиці Storage. Цей тригер виконується перед виконанням операцій INSERT або UPDATE у таблиці Storage.

Приклад роботи показано на рис. 7.1.5.1 та рис. 7.1.5.2.

```
CREATE OR REPLACE FUNCTION validate_perishable_cargo()
```

```
RETURNS TRIGGER AS $$
```

```
DECLARE
```

```
v_cargotype VARCHAR;
```

```
BEGIN
```

```
SELECT cargotype
```

```

    INTO v_cargotype
  FROM cargo
  WHERE cargoid = NEW.cargoid;

  IF v_cargotype = 'Perishable' AND NEW.enddate IS NULL THEN
    RAISE EXCEPTION 'Perishable cargo must have an enddate specified.';
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

create trigger trg_validate_perishable_cargo
  before insert or update on storage
  for each row
execute function validate_perishable_cargo();

```

1002	1004	10.00	Perishable	10
------	------	-------	------------	----

Рис. 7.1.5.1 – Демонстрація, що вантаж з ідентифікатором 1004 має тип "Perishable"

```
seaport.public> insert into storage (terminalid, cargoid, startdate) values (1, 1004, '2024-12-20 10:00:00')
[2024-12-21 03:29:23] [P0001] ERROR: Perishable cargo must have an enddate specified.
[2024-12-21 03:29:23] Where: PL/pgSQL function validate_perishable_cargo() line 11 at RAISE
```

Рис. 7.1.5.2 – Спроба додати на склад вантаж з типом "Perishable", зберігання якого немає дати закінчення

7.2 Тексти представлень

7.2.1 Представлення client_cargo_status

Представлення client_cargo_status створено для швидкого отримання інформації про вантаж клієнтів, включаючи тип вантажу, вагу, стан доставки, а також дату доставки (або зазначення її відсутності). Це дозволяє легко контролювати статус вантажів клієнтів і їх логістичний процес.

Приклад роботи на рис. 7.2.1.

create or replace view client_cargo_status as

select c.cargoid,

c.weight,

c.cargotype,

clients.name as Client_Name,

shipments.status as Shipment_Status,

coalesce(shipments.shipmentdate::text, 'Not Scheduled') as shipment_date

from cargo c

join clients on c.clientid = clients.clientid

left join shipments on c.cargoid = shipments.cargoid;

	cargoid	weight	cargotype	client_name	shipment_status	shipment_date
1		618	33.77 Dangerous	Maritime Partners 369	Delivered	2024-12-28 04:42:00
2		775	58.87 Dangerous	Northern Trade Hub 544	Delivered	2024-02-22 03:42:00
3		196	33.21 Perishable	Horizon Lines 668	Delivered	2024-11-28 14:13:00
4		303	24.76 Liquid	Portside Associates 726	Delivered	2024-10-21 21:13:00
5		6	1.85 Project	Northern Trade Hub 304	Delivered	2024-05-01 14:28:00
6		1000	43.39 Liquid	Blue Ocean Shipping 652	Delivered	2024-11-04 04:26:00
7		106	48.50 Bulk	Atlantic Trade 162	Canceled	Not Scheduled
8		762	6.29 Project	Western Freight 679	Canceled	Not Scheduled
9		930	28.29 Dangerous	Horizon Lines 799	In Progress	2024-10-27 08:50:00
10		979	79.79 Liquid	Oceanic Trade 670	Canceled	Not Scheduled
11		968	21.57 Container	Eastern Logistics 946	Delivered	2024-12-21 07:27:00
12		419	34.46 General	Pacific Ventures 195	Canceled	Not Scheduled
13		680	79.43 General	Northern Trade Hub 943	Canceled	Not Scheduled
14		373	67.33 General	Wave Logistics 239	Canceled	Not Scheduled
15		952	41.87 RoRo	Transworld Shipping 670	In Progress	2024-09-17 17:28:00

Рис. 7.2.1 – Приклад роботи представлення client_cargo_status

7.2.2 Представлення terminal_load_status

Представлення terminal_load_status створено для моніторингу завантаженості терміналів. Воно дозволяє отримати інформацію про поточний обсяг вантажів, що зберігаються в кожному терміналі, порівняно з його максимальною пропускною здатністю.

Приклад роботи показано на рис. 7.2.2.

create or replace view terminal_load_status as

```
select t.name                                as terminal_name,
       t.capacity                            as terminal_capacity,
       coalesce(sum(c.weight), 0)            as total_cargo_weight,
       round((coalesce(sum(c.weight), 0) / t.capacity) * 100, 2) as load_percentage
```

from terminals t

```
left join storage s on t.terminalid = s.terminalid
```

```
left join cargo c on s.cargoid = c.cargoid
```

group by t.terminalid, t.name, t.capacity

order by load_percentage desc;

	terminal_name	terminal_capacity	total_cargo_weight	load_percentage
1	Loading Dock 75	661.93	507.78	76.71
2	Bulk Goods Terminal 64	784.31	482.62	61.53
3	Pier C 72	620.21	343.02	55.31
4	Main Terminal 92	917.69	491.01	53.5
5	Cargo Center 17	1119.46	545.95	48.77
6	Bulk Goods Terminal 18	1502.54	574.9	38.26
7	Pier B 80	1258.98	476.9	37.88
8	Freight Center 43	1233.76	428.29	34.71
9	South Dock 1	1477.40	359.1	24.31
10	Pier A 100	640.10	145.29	22.7
11	Logistics Terminal 22	1090.90	238.4	21.85
12	Bulk Goods Terminal 75	1566.22	339.02	21.65
13	East Dock 50	989.24	204.41	20.66
14	Cargo Center 74	2179.52	439.43	20.16
15	Pier B 38	3859.72	754.54	19.55

Рис. 7.2.2 – Приклад роботи представлення terminal_load_status

7.2.3 Представлення ship_load_status

Представлення `ship_load_status` створено для аналізу завантаженості кораблів у терміналах. Воно дозволяє оцінити, наскільки кораблі завантажені вантажами, а також відображає їх заплановані дати прибуття і відправлення.

Приклад роботи показано на рис. 7.2.3.

create or replace view `ship_load_status` as

select sh.shipid,

sh.name as ship_name,

sh.capacity as ship_capacity,

coalesce(sum(cargo.weight), 0) as total_cargo_weight,

*round((coalesce(sum(cargo.weight), 0) / sh.capacity) * 100, 2)* as

load_percentage,

schedules.arrivaldate,

schedules.departuredate

from ships sh

join schedules on sh.shipid = schedules.shipid

left join storage on schedules.terminalid = storage.terminalid

left join cargo on storage.cargoid = cargo.cargoid

group by sh.shipid, sh.name, sh.capacity, schedules.arrivaldate,

schedules.departuredate

order by load_percentage desc, sh.name;

shipid	ship_name	ship_capacity	total_cargo_weight	load_percentage	arrivaldate	departuredate
1	725 Merchant Wave 315	536.73	639.88	119.07	2024-03-13 06:37:00.000000	2024-03-19 08:37:00.000000
2	831 Neptunes Pride 821	611.33	668.92	109.42	2024-07-05 13:04:00.000000	2024-07-06 15:04:00.000000
3	37 Sea Star 791	532.62	576.97	108.33	2024-06-20 00:09:00.000000	2024-06-22 22:09:00.000000
4	275 Sea Star 825	566.21	606.96	107.2	2024-12-10 15:17:00.000000	2024-12-13 17:17:00.000000
5	725 Merchant Wave 315	536.73	572.01	106.57	2024-03-06 08:40:00.000000	2024-03-07 05:40:00.000000
6	994 Freight King 926	695.64	705.94	101.48	2024-09-17 08:22:00.000000	2024-09-19 04:22:00.000000
7	774 Cargo Titan 941	580.11	531.36	91.6	2024-05-15 23:56:00.000000	2024-05-25 14:56:00.000000
8	12 Neptunes Pride 808	741.51	668.92	90.21	2024-08-28 14:22:00.000000	2024-09-04 00:22:00.000000
9	26 Pacific Mariner 574	566.89	508.28	89.66	2024-07-24 22:29:00.000000	2024-07-27 06:29:00.000000
10	581 Sea Breeze 936	644.72	574.9	89.17	2024-11-08 00:51:00.000000	2024-11-15 17:51:00.000000
11	633 Cargo Titan 621	502.34	446.11	88.81	2024-02-12 04:52:00.000000	2024-02-13 03:52:00.000000
12	831 Neptunes Pride 821	611.33	541.74	88.62	2024-02-21 06:00:00.000000	2024-02-27 14:00:00.000000
13	581 Sea Breeze 936	644.72	556.26	86.28	2024-03-13 18:27:00.000000	2024-03-18 11:27:00.000000
14	581 Sea Breeze 936	644.72	550.16	85.33	2024-05-24 18:21:00.000000	2024-05-27 10:21:00.000000
15	26 Pacific Mariner 574	566.89	466.47	82.29	2024-05-20 20:47:00.000000	2024-05-21 20:47:00.000000
16	774 Cargo Titan 941	580.11	470.06	81.03	2024-09-26 18:11:00.000000	2024-10-01 19:11:00.000000
17	862 Wave Rider 342	877.42	705.94	80.46	2024-06-12 09:03:00.000000	2024-06-13 09:03:00.000000
18	12 Neptunes Pride 808	741.51	595.89	80.36	2024-02-13 06:10:00.000000	2024-02-14 04:10:00.000000
19	831 Neptunes Pride 821	611.33	476.89	78.01	2024-01-01 00:57:00.000000	2024-01-02 11:57:00.000000
20	12 Neptunes Pride 808	741.51	559.9	75.51	2024-12-01 15:35:00.000000	2024-12-08 00:35:00.000000

Рис. 7.2.3 – Приклад роботи `ship_load_status`

7.2.4 Представлення cargo_status_summary

Представлення cargo_status_summary створено для підрахунку статистики вантажів за їх типами. Воно надає інформацію про загальну кількість вантажів, їхні статуси та загальну вагу.

Приклад роботи на рис. 7.2.4.

```
create or replace view cargo_status_summary as
select c.cargotype                                as cargo_type,
       count(c.cargoid)                          as total_cargo_count,
       coalesce(sum(case when s.status = 'Pending' then 1 else 0 end), 0) as
pending_count,
       coalesce(sum(case when s.status = 'In Progress' then 1 else 0 end), 0) as
in_progress_count,
       coalesce(sum(case when s.status = 'Delivered' then 1 else 0 end), 0) as
delivered_count,
       coalesce(sum(case when s.status = 'Canceled' then 1 else 0 end), 0) as
canceled_count,
       coalesce(sum(c.weight), 0)                as total_weight
from cargo c
       left join shipments s on c.cargoid = s.cargoid
group by c.cargotype
order by total_weight desc;
```

	cargo_type ▾	total_cargo_count ▾	pending_count ▾	in_progress_count ▾	delivered_count ▾	canceled_count ▾	total_weight ▾
1	RoRo	355	82	85	102	73	19221.35
2	Bulk	355	77	75	89	98	18424.14
3	Liquid	327	80	85	81	64	15984.4
4	Container	296	79	63	66	77	15796.61
5	Dangerous	334	80	72	77	93	15585.04
6	General	320	66	80	80	77	14541.97
7	Perishable	272	55	70	60	76	11320.02
8	Project	225	50	57	50	59	11148.55

Рис. 7.2.4 – Приклад роботи представлення cargo_status_summary

7.2.5 Представлення berth_occupancy_schedule

Представлення `berth_occupancy_schedule` створено для відображення розкладу зайнятості причалів. Воно об'єднує інформацію про розклад суден, доступні причали, термінали та порти призначення.

Приклад роботи показано на рис. 7.2.5.

```
create or replace view berth_occupancy_schedule as
select schedules.scheduleid,
       ships.name           as ship_name,
       ports.portname       as destination_port,
       terminals.name       as terminal_name,
       berths.berthid       as berth_id,
       berths.length        as berth_length,
       berths.depth         as berth_depth,
       berths.status        as berth_status,
       schedules.arrivaldate as arrival_date,
       schedules.departuredate as departure_date
from schedules
      join ships on schedules.shipid = ships.shipid
      join berths on schedules.terminalid = berths.terminalid
      join terminals on berths.terminalid = terminals.terminalid
      join ports on terminals.portid = ports.portid
where berths.status = 'Available'
      and berths.length >= ships.requiredlength
      and berths.depth >= ships.draft
      and ports.portid = schedules.destinationportid
order by schedules.arrivaldate, berths.berthid;
```

	scheduleid	ship_name	destination_port	terminal_name	berth_id	berth_length	berth_depth	berth_status	arrival_date	depar
1	1078	Neptunes Pride 502	St. George	Bulk Goods Terminal 33	272	423.03	18.69	Available	2024-01-01 11:31:00.000000	2024-01
2	1078	Neptunes Pride 502	St. George	Bulk Goods Terminal 33	803	494.87	20.87	Available	2024-01-01 11:31:00.000000	2024-01
3	846	Freight King 299	Townsville	Loading Dock 75	107	412.53	26.96	Available	2024-01-24 13:55:00.000000	2024-01
4	1889	Unity Fleet 504	Port Louis	Central Pier 3	448	449.84	19.24	Available	2024-02-09 02:44:00.000000	2024-02
5	1889	Unity Fleet 504	Port Louis	Central Pier 3	1015	425.21	17.23	Available	2024-02-09 02:44:00.000000	2024-02
6	1553	Unity Fleet 773	Fremantle	South Dock 29	49	562.84	20.39	Available	2024-02-17 04:32:00.000000	2024-02
7	1553	Unity Fleet 773	Fremantle	South Dock 29	721	550.99	27.81	Available	2024-02-17 04:32:00.000000	2024-02
8	3193	Oceanic Pioneer 873	Qingdao	Pier C 34	654	459.50	27.78	Available	2024-03-04 12:29:00.000000	2024-03
9	3193	Oceanic Pioneer 873	Qingdao	Pier C 34	1140	586.82	24.03	Available	2024-03-04 12:29:00.000000	2024-03
10	1344	Trade Wind 559	Victoria	East Dock 37	938	409.32	29.54	Available	2024-04-07 22:02:00.000000	2024-04
11	1344	Trade Wind 559	Victoria	East Dock 37	1131	383.48	25.15	Available	2024-04-07 22:02:00.000000	2024-04
12	3540	Oceanic Pioneer 700	Qingdao	Pier C 34	654	459.50	27.78	Available	2024-04-16 06:13:00.000000	2024-04
13	3540	Oceanic Pioneer 700	Qingdao	Pier C 34	1140	586.82	24.03	Available	2024-04-16 06:13:00.000000	2024-04
14	3717	Merchant Wave 588	Savannah	Logistics Terminal 35	74	596.84	18.20	Available	2024-05-08 01:15:00.000000	2024-05
15	2844	Atlantic Explorer 326	Tripoli	Bulk Goods Terminal 42	630	571.98	23.90	Available	2024-05-13 11:45:00.000000	2024-05
16	359	Oceanic Pioneer 193	Calais	West Terminal 3	453	426.24	19.14	Available	2024-06-04 08:25:00.000000	2024-06
17	2482	Pacific Mariner 628	St. George	Bulk Goods Terminal 33	272	423.03	18.69	Available	2024-06-13 09:44:00.000000	2024-06
18	2482	Pacific Mariner 628	St. George	Bulk Goods Terminal 33	803	494.87	20.87	Available	2024-06-13 09:44:00.000000	2024-06
19	87	Blue Horizon 588	Limassol	Pier C 71	866	596.18	27.40	Available	2024-06-26 13:11:00.000000	2024-06
20	87	Blue Horizon 588	Limassol	Pier C 71	896	516.26	15.03	Available	2024-06-26 13:11:00.000000	2024-06
21	2006	Blue Horizon 298	Chennai	Loading Dock 36	405	534.90	21.32	Available	2024-07-14 00:49:00.000000	2024-07
22	2006	Blue Horizon 298	Chennai	Loading Dock 36	1156	428.80	16.11	Available	2024-07-14 00:49:00.000000	2024-07

Рис. 7.2.5 – Приклад роботи представлення berth_occurency_schedule

7.2.6 Представлення client_type_cargo_summary

Представлення client_type_cargo_summary створено для підрахунку кількості та загальної ваги вантажів, що належать клієнтам різних типів.

Приклад роботи на рис. 7.2.6.

create or replace view client_type_cargo_summary as

select clients.type as client_type,

count(cargo.cargoid) as total_cargo_count,

coalesce(sum(cargo.weight), 0) as total_cargo_weight

from clients

left join cargo on clients.clientid = cargo.clientid

group by clients.type

order by total_cargo_weight desc;

	client_type	total_cargo_count	total_cargo_weight
1	Individual	353	17093.06
2	Corporate	338	16304.28
3	Government	311	15829.13

Рис. 7.2.6 – Приклад роботи представлення client_type_cargo_summary

7.3 Тексти збережених процедур та функцій.

Відповідно до моїх бізнес правил були розроблені наступні функції та процедури.

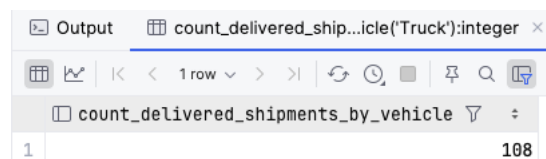
7.3.1 Функція `count_delivered_shipment_by_vehicle`

Функція `count_delivered_shipment_by_vehicle` використовується для визначення кількості вантажів, які були успішно доставлені за допомогою певного типу транспортного засобу. Ця функція дозволяє оперативно отримати кількісну статистику доставки для аналізу ефективності логістичних процесів.

Приклад роботи показано на рис. 7.3.1.1.

```
create or replace function count_delivered_shipment_by_vehicle(p_vehicle_type
VARCHAR)
returns int as $$
declare v_count int;
begin
    select count(shipments.shipmentid)
    into v_count
    from shipments
        join transporters on shipments.transporterid = transporters.transporterid
    where transporters.vehicletype = p_vehicle_type
    and shipments.status = 'Delivered';

    return v_count;
end;
$$ language plpgsql;
```



The screenshot shows a database interface with a tab titled 'Output' and a sub-tab 'count_delivered_ship...icle('Truck'):integer x'. Below the sub-tab, there is a table with one row and one column. The column header is 'count_delivered_shipments_by_vehicle' and the value in the row is '108'.

count_delivered_shipments_by_vehicle
108

Рис. 7.3.1.1 – Приклад результату від функції `count_delivered_shipment_by_vehicle` для обчислення кількості виконаних доставок вантажівками

7.3.2 Процедура update_employee_status

Процедура `update_employee_status` використовується для масового оновлення статусу активності співробітників у таблиці `employees`. Вона дозволяє змінювати статус декількох співробітників одночасно, що значно спрощує управління даними персоналу.

Приклад роботи показано на рис. 7.3.2.1 та рис. 7.3.2.2.

```
create or replace procedure update_employee_status
(
    p_employee_ids int[],
    p_isactive boolean
)
language plpgsql
as $$
begin
    update employees
    set isactive = p_isactive
    where employeeid = any(p_employee_ids);

    raise notice 'Employee status updated successfully';
end;
$$;
```

	employeeid	name	position	terminalid	isactive
1	1	Chris 68	Supervisor	194	false
2	2	John 329	Manager	190	false
3	3	Logan 31	Manager	98	false

Рис. 7.3.2.1 – Демонстрація статусу співробітників з ідентифікаторами 1, 2 та

employeeid	name	position	terminalid	isactive
1	1 Chris 68	Supervisor	194	• true
2	2 John 329	Manager	190	• true
3	3 Logan 31	Manager	98	• true

Рис. 7.3.2.2 – Статус співробітників з ідентифікаторами 1, 2 та 3 після виклику процедури `update_employee_status` з переданим параметром `p_isactive = true`

7.3.3 Функція `is_berth_available`

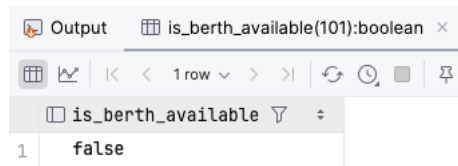
Функція `is_berth_available` визначає чи доступний причал з вказаним ідентифікатором для використання. Ця функція забезпечує швидкий механізм перевірки статусу причалу, що важливо для планування портових операцій.

Приклад роботи показано на рис. 7.3.3.1 та рис. 7.3.3.2.

```
create or replace function is_berth_available(p_berth_id int)
    returns boolean as $$
declare
    v_status varchar;
begin
    if not exists (select 1 from berths where berthid = p_berth_id) then
        raise exception 'Berth with ID % does not exist.', p_berth_id;
    end if;

    select status
    into v_status
    from berths
    where berthid = p_berth_id;

    return v_status = 'Available';
end;
$$ language plpgsql;
```



Output is_berth_available(101):boolean	
is_berth_available	
1	false

Рис. 7.3.3.1 – Перевірка чи доступний зараз причал з ідентифікатором 101

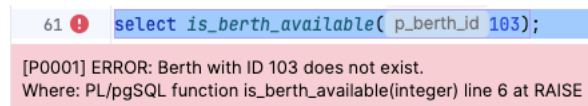


Рис. 7.3.3.2 – Спроба перевірити статус причалу з не існуючим в базі ідентифікатором

7.3.4 Функція count_client_cargo

Функція count_client_cargo дозволяє підрахувати кількість вантажів, пов'язаних із конкретним клієнтом, з можливістю фільтрації за типом вантажу. Це забезпечує зручність отримання статистики про вантажі клієнтів для аналітичних або операційних цілей.

Приклад роботи показано на рис. 7.3.4.1.

```
create or replace function count_client_cargo(p_client_id int, p_cargo_type
varchar default null)
```

```
returns int as $$
```

```
declare
```

```
v_cargo_count int;
```

```
begin
```

```
if not exists (select 1 from clients where clientid = p_client_id) then
```

```
raise exception 'Client with ID % does not exist.', p_client_id;
```

```
end if;
```

```
select count(*)
```

```
into v_cargo_count
```

```
from cargo
```

```
where clientid = p_client_id
```

```

and (p_cargo_type is null or cargotype = p_cargo_type);

return v_cargo_count;

end;

$$ language plpgsql;

```

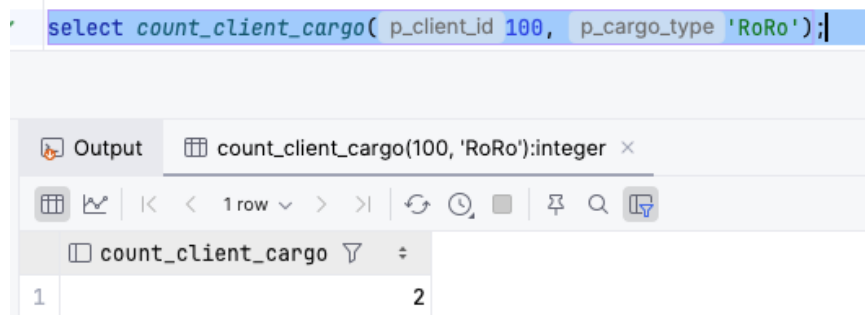


Рис. 7.3.4.1 – Перевірка кількості вантажу типу "RoRo" клієнта з ідентифікатором 100

7.3.5 Функція mark_cargo_as_delivered

Функція `mark_cargo_as_delivered` призначена для оновлення статусу вантажу в таблиці `shipments`, який позначає процес доставки. Основна мета цієї функції – змінити статус обраного вантажу на "Delivered" (доставлений) і встановити дату доставки, забезпечуючи точність і актуальність даних у базі.

Спочатку функція виконує перевірку на існування вантажу з вказаним ідентифікатором `p_cargo_id` у таблиці `cargo`. Якщо такий вантаж не знайдено, функція негайно припиняє виконання і видає відповідну помилку з повідомленням, що вказаний вантаж не існує. Це важливий механізм захисту від спроб оновлення неіснуючих даних.

Далі функція звертається до таблиці `shipments` для пошуку запису з цим вантажем і статусом "In Progress". Ця перевірка гарантує, що тільки активні перевезення можуть бути відмічені як доставлені. У разі, якщо запис про перевезення зі статусом "In Progress" не знайдено, функція знову припиняє виконання і видає помилку, повідомляючи, що немає активного перевезення для вказаного вантажу.

Якщо всі перевірки успішно пройдені, функція виконує оновлення запису в таблиці shipments. Вона змінює статус на "Delivered" і додає дату доставки, яку отримує як параметр p_delivery_date. Це оновлення гарантує, що система правильно відображає стан вантажу після завершення його доставки.

Приклад роботи показано на рис. 7.3.5.1, рис. 7.3.5.2 та рис. 7.3.5.3.

```
create or replace function mark_cargo_as_delivered(p_cargo_id int,
p_delivery_date timestamp)
returns text as $$
declare
    v_shipment_id int;

begin
    if not exists (select 1 from cargo where cargoid = p_cargo_id) then
        raise exception 'cargo id % does not exist', p_cargo_id;
    end if;

    select shipmentid
    into v_shipment_id
    from shipments
    where cargoid = p_cargo_id and status = 'In Progress'
    order by shipmentdate desc
    limit 1;

    if not found then
        raise exception 'no shipment with status "in progress" found for cargoid %',
p_cargo_id;
    end if;

    update shipments
    set status = 'Delivered', shipmentdate = p_delivery_date
```



```
where shipmentid = v_shipment_id;

return format('cargo id %s has been marked as delivered on %s', p_cargo_id,
p_delivery_date);
end;
$$ language plpgsql;
```

shipmentid	cargoid	transporterid	shipmentdate	status
1	1796	815	1 2024-08-15 23:50:00.000000	In Progress

Рис. 7.3.5.1 – Демонстрація, що вантаж з ідентифікатором 815 має статус "In Progress"

```
select mark_cargo_as_delivered( p_cargo_id 815, p_delivery_date '2024-09-15 23:50:00');
```

Output

mark_cargo_as_delive...09-15 23:50:00'):text

1 row

mark_cargo_as_delivered

1 cargo id 815 has been marked as delivered on 2024-09-15 23:50:00

Рис. 7.3.5.2 – Помітка вантажу з ідентифікатором 815 як доставлений

shipmentid	cargoid	transporterid	shipmentdate	status
448	1796	815	1 2024-09-15 23:50:00.000000	Delivered

Рис. 7.3.5.3 – Демонстрація успішного встановлення статусу "Delivered" вантажу з ідентифікатором 815

7.3.6 Функція can_add_cargo_to_storage

Функція can_add_cargo_to_storage перевіряє можливість додавання нового вантажу до зберігання на терміналі, враховуючи поточну заповненість і максимальну вантажопідйомність терміналу.

Приклад роботи показано на рис. 7.3.6.1.

```

create or replace function can_add_cargo_to_storage(p_terminal_id int,
p_cargo_weight decimal)
    returns boolean as $$
declare
    v_current_weight decimal;
    v_capacity decimal;

begin
    select capacity
    into v_capacity
    from terminals
    where terminalid = p_terminal_id;

    if not found then
        raise exception 'terminal with id % does not exist', p_terminal_id;
    end if;

    select coalesce(sum(cargo.weight), 0)
    into v_current_weight
    from storage
        join cargo on storage.cargoid = cargo.cargoid
    where storage.terminalid = p_terminal_id;

    return v_current_weight + p_cargo_weight <= v_capacity;
end;
$$ language plpgsql;

```

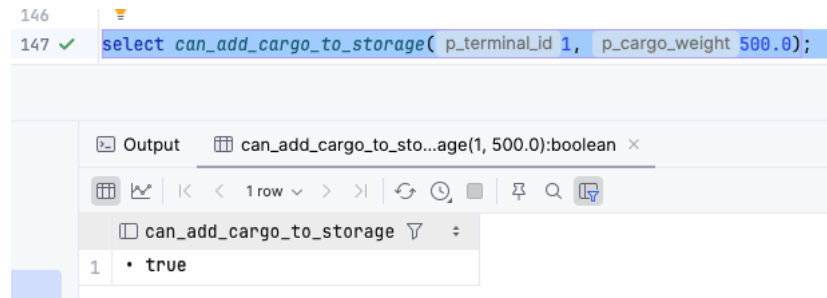


Рис. 7.3.6.1 – Демонстрація роботи функції `can_add_cargo_to_storage`

7.3.7 Функція `count_ships_in_port`

Функція `count_ships_in_port` підраховує кількість унікальних кораблів, що прибули до зазначеного порту за вказаний часовий проміжок.

Приклад роботи показано на рис. 7.3.7.1.

```
create or replace function count_ships_in_port(p_port_id int, p_start_date date,
p_end_date date)
```

```
returns int as $$
```

```
declare
```

```
v_ship_count int;
```

```
begin
```

```
if not exists (select 1 from ports where portid = p_port_id) then
```

```
raise exception 'port with id % does not exist', p_port_id;
```

```
end if;
```

```
select count(distinct schedules.shipid)
```

```
into v_ship_count
```

```
from schedules
```

```
join terminals on schedules.terminalid = terminals.terminalid
```

```
join ports on terminals.portid = ports.portid
```

```
where ports.portid = p_port_id
```

```
and schedules.arrivaldate >= p_start_date
```

```
and schedules.departuredate <= p_end_date;
```

```

return v_ship_count;
end;
$$ language plpgsql;

```

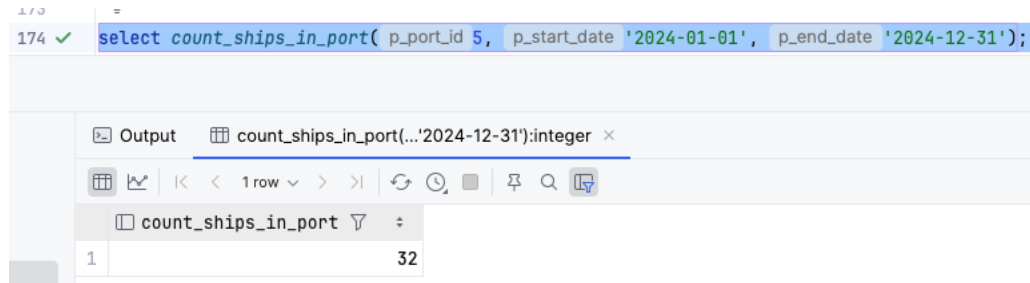


Рис. 7.3.7.1 – Демонстрація кількості кораблів, які відвідали порт з ідентифікатором 5 за 2024 рік, за допомогою функції `count_ships_in_port`

7.3.8 Функція `count_ships_in_port`

Функція `count_ships_in_port` підраховує кількість унікальних кораблів, що прибули до зазначеного порту за вказаний часовий проміжок.

Приклад роботи показано на рис. 7.3.8.1.

```

create or replace function find_longest_available_berth(p_port_id int)
returns table (
    berthid    int,
    terminalid  int,
    length     decimal,
    depth      decimal
) as $$

```

```

begin

```

```

    if not exists (select 1 from ports where portid = p_port_id) then
        raise exception 'Port with ID % does not exist', p_port_id;
    end if;

```

```

return query

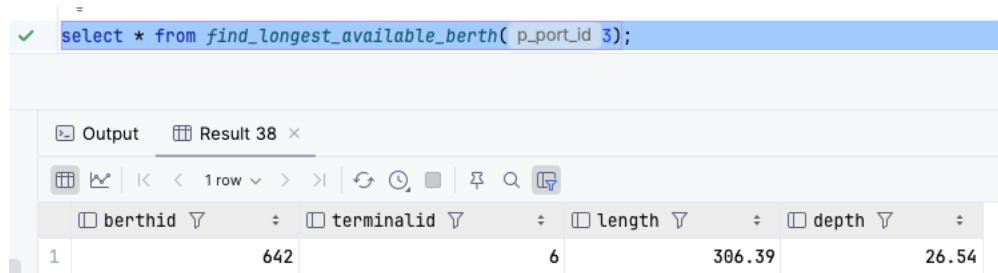
```

```

select berths.berthid, berths.terminalid, berths.length, berths.depth
from berths
    join terminals on berths.terminalid = terminals.terminalid
    join ports on terminals.portid = ports.portid
where ports.portid = p_port_id
    and berths.status = 'Available'
order by berths.length desc
limit 1;

if not found then
    raise notice 'No available berths found for port ID %', p_port_id;
end if;
end;
$$ language plpgsql;

```



The screenshot shows a SQL query execution interface. The query is: `select * from find_longest_available_berth(p_port_id 3);`. The result is displayed in a table with 4 columns: `berthid`, `terminalid`, `length`, and `depth`. There is one row of data.

berthid	terminalid	length	depth
1	642	6	306.39

Рис. 7.3.8.1 – Знаходження найдовшого причалу в порту з ідентифікатором 3

7.3.9 Функція `check_schedule_date_conflicts`

Функція `check_schedule_date_conflicts` визначає конфлікти в розкладі для заданого причалу, якщо час прибуття та відправлення двох суден перетинаються.

Приклад роботи показано на рис. 7.3.9.1.

```

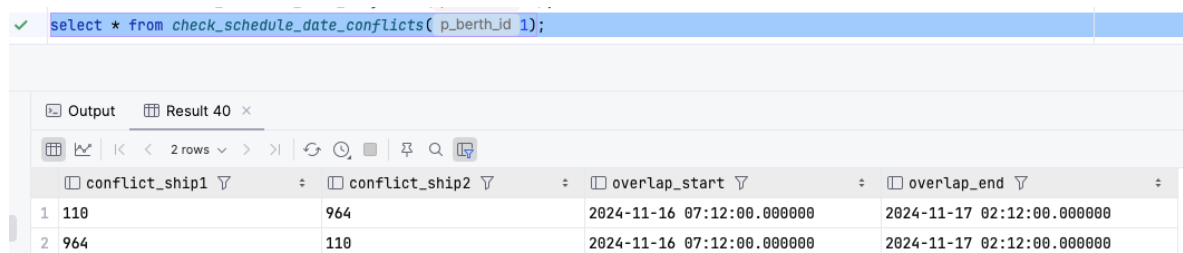
create or replace function check_schedule_date_conflicts(p_berth_id int)
returns table (
    conflict_ship1   varchar,

```

```

        conflict_ship2    varchar,
        overlap_start     timestamp,
        overlap_end       timestamp
    ) as $$
begin
    return query
        select s1.shipid::varchar as conflict_ship1,
            s2.shipid::varchar as conflict_ship2,
            greatest(s1.arrivaldate, s2.arrivaldate) as overlap_start,
            least(s1.departuredate, s2.departuredate) as overlap_end
        from schedules s1
            join schedules s2
                on s1.berthid = s2.berthid
                and s1.scheduleid <> s2.scheduleid
                and s1.arrivaldate < s2.departuredate
                and s2.arrivaldate < s1.departuredate
        where s1.berthid = p_berth_id
        order by overlap_start;
end;
$$ language plpgsql;

```



	conflict_ship1	conflict_ship2	overlap_start	overlap_end
1	110	964	2024-11-16 07:12:00.000000	2024-11-17 02:12:00.000000
2	964	110	2024-11-16 07:12:00.000000	2024-11-17 02:12:00.000000

Рис. 7.3.9.1 – Результат виконання функції `check_schedule_date_conflicts` для визначення конфліктів у розкладі суден на причалі з ID 1

7.3.10 Функція `average_delivery_time_by_vehicle_type`

Функція `average_delivery_time_by_vehicle_type` призначена для обчислення середнього часу доставки вантажів залежно від типу транспортного засобу. Вона повертає результат у вигляді інтервалу, що відображає середній час між початком зберігання вантажу та датою його доставки.

Приклад роботи показано на рис. 7.3.10.1.

```
create or replace function average_delivery_time_by_vehicle_type(p_vehicle_type
varchar)
returns interval as $$
declare
    v_average_time interval;
begin

    select avg(shipments.shipmentdate - storage.startdate)
    into v_average_time
    from shipments
        join transporters on shipments.transporterid = transporters.transporterid
        join storage on shipments.cargoid = storage.cargoid
    where transporters.vehicletype = p_vehicle_type
        and shipments.status = 'Delivered';

    return v_average_time;
end;
$$ language plpgsql;
```

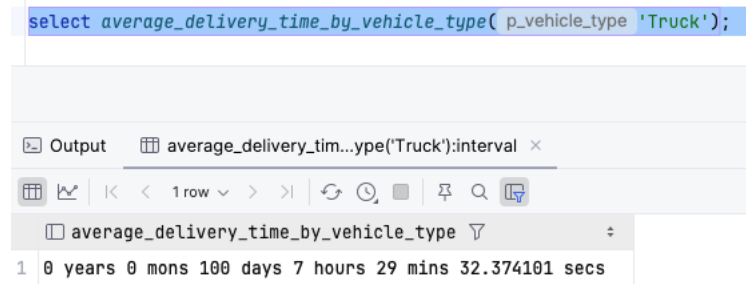


Рис. 7.3.10.1 – Результат виконання функції

`average_delivery_time_by_vehicle_type` для транспортного засобу типу "Truck"

7.4 SQL-запити

1) Запит на отримання інформації про кораблі, які швартуються у терміналах порту "Rotterdam".

Результат показано на рис. 7.4.1.

```
select distinct ships.name    as Ship_Name,
               terminals.name as Terminal_Name,
               ports.portname
from schedules
  join ships on schedules.shipid = ships.shipid
  join berths on schedules.berthid = berths.berthid
  join terminals on berths.terminalid = terminals.terminalid
  join ports on terminals.portid = ports.portid

where terminals.terminalid in (
  select terminalid from terminals
  join ports on terminals.portid = ports.portid
  where ports.portname = 'Rotterdam'
);
```


	ship_name	terminal_name	portname
1	Anchor Titan 427	Industrial Bay 87	Rotterdam
2	Blue Horizon 226	Freight Center 50	Rotterdam
3	Blue Horizon 686	Industrial Bay 87	Rotterdam
4	Freight King 233	Industrial Bay 87	Rotterdam
5	Freight King 643	Industrial Bay 87	Rotterdam
6	Freight King 778	Freight Center 50	Rotterdam
7	Global Carrier 653	Freight Center 50	Rotterdam
8	Harbor Queen 747	Industrial Bay 87	Rotterdam
9	Liberty Sailor 138	Industrial Bay 87	Rotterdam
10	Liberty Sailor 190	Industrial Bay 87	Rotterdam
11	Liberty Sailor 384	Industrial Bay 87	Rotterdam
12	Liberty Sailor 761	Industrial Bay 87	Rotterdam
13	Liberty Sailor 779	Freight Center 50	Rotterdam
14	Merchant Wave 315	Freight Center 50	Rotterdam
15	Merchant Wave 439	Industrial Bay 87	Rotterdam
16	Merchant Wave 962	Industrial Bay 87	Rotterdam
17	Ocean Voyager 316	Industrial Bay 87	Rotterdam
18	Ocean Voyager 992	Freight Center 50	Rotterdam
19	Oceanic Pioneer 289	Industrial Bay 87	Rotterdam
20	Oceanic Pioneer 939	Industrial Bay 87	Rotterdam
21	Pacific Mariner 414	Industrial Bay 87	Rotterdam
22	Pacific Mariner 991	Freight Center 50	Rotterdam
23	Sea Breeze 257	Industrial Bay 87	Rotterdam
24	Sea Breeze 530	Freight Center 50	Rotterdam
25	Sea Star 155	Industrial Bay 87	Rotterdam
26	Sea Star 855	Freight Center 50	Rotterdam
27	Wave Rider 176	Industrial Bay 87	Rotterdam

Рис. 7.4.1 – Результат запиту на отримання інформації про кораблі у порту "Rotterdam"

2) Запит на отримання інформації про доставлені вантажі, перевезені транспортними засобами типу "Train"

Результат можна побачити на рис. 7.4.2.

```

select cargo.cargoid,
       cargo.weight,
       cargo.cargotype,
       clients.Name    as Client_Name,
       transporters.name as Transporter_Name
from cargo
      join clients on cargo.clientid = clients.clientid
      join shipments on cargo.cargoid = shipments.cargoid

```

join transporters on shipments.transporterid = transporters.transporterid

where transporters.vehicletype = 'Train' and shipments.status = 'Delivered';

	cargoid	weight	cargotype	client_name	transporter_name
1	264	52.14	Perishable	Transworld Shipping 557	Freight Masters 43
2	588	51.27	General	Blue Ocean Shipping 801	Seaway Transporters 32
3	926	90.87	RoRo	Blue Ocean Shipping 663	Prime Cargo 37
4	459	81.26	Perishable	Blue Ocean Shipping 769	Prime Cargo 84
5	526	26.00	RoRo	Western Freight 604	Anchor Logistics 1
6	487	54.19	Dangerous	Northern Trade Hub 657	RailLink Carriers 51
7	487	54.19	Dangerous	Northern Trade Hub 657	Prime Cargo 37
8	942	50.73	Perishable	Maritime Group 137	Freight Masters 99
9	570	0.93	Bulk	Cargo Solutions 179	Rapid Shipping 17
10	186	56.84	Bulk	Cargo Solutions 179	Blue Horizon Freight 3
11	186	56.84	Bulk	Cargo Solutions 179	Seaway Transporters 25
12	389	20.43	Project	Sea Freight Inc. 237	Speedy Logistics 33
13	847	16.02	Project	Maritime Group 508	Seaway Transporters 61
14	847	16.02	Project	Maritime Group 508	Seaway Transporters 25
15	171	30.80	Dangerous	Sea Freight Inc. 620	Blue Horizon Freight 3
16	119	71.28	Liquid	Sea Freight Inc. 620	Rapid Shipping 17
17	688	31.59	Project	Wave Logistics 674	Speedy Logistics 33
18	642	50.49	Liquid	Northern Trade Hub 997	Swift Move 25
19	279	65.39	Project	Oceanic Trade 340	RailLink Carriers 51
20	727	36.61	Dangerous	Atlantic Trade 705	Cargo Express 82

Рис. 7.4.2 Результат запиту на отримання інформації про доставлені вантажі, перевезені "Train"

3) Запит на отримання клієнтів, які перевезли більше 50 тонн вантажу, із зазначенням загальної ваги вантажу.

Результат можна побачити на рис. 7.4.3.

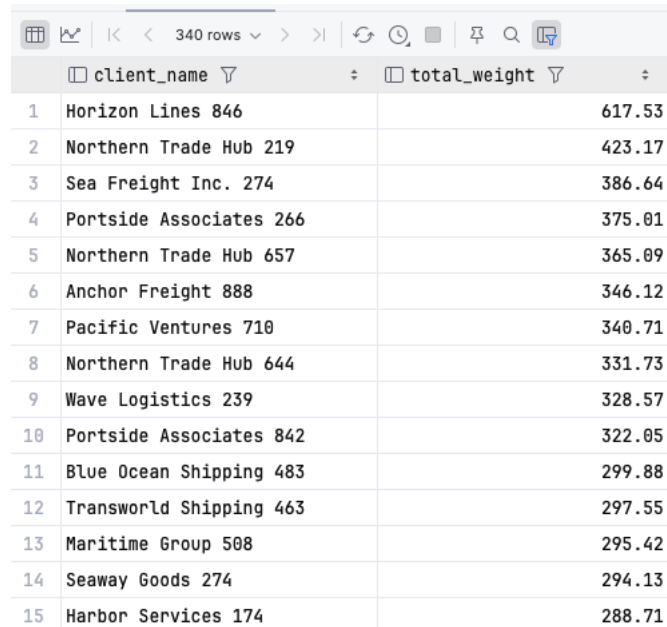
```
select clients.name    as Client_Name,
       sum(cargo.weight) as Total_Weight
from clients
       join cargo on clients.clientid = cargo.clientid
```

```
where clients.clientid in (
       select clientid
       from cargo
       group by clientid
       having sum(weight) > 50
```

)

group by clients.name

order by Total_Weight desc;



	client_name	total_weight
1	Horizon Lines 846	617.53
2	Northern Trade Hub 219	423.17
3	Sea Freight Inc. 274	386.64
4	Portside Associates 266	375.01
5	Northern Trade Hub 657	365.09
6	Anchor Freight 888	346.12
7	Pacific Ventures 710	340.71
8	Northern Trade Hub 644	331.73
9	Wave Logistics 239	328.57
10	Portside Associates 842	322.05
11	Blue Ocean Shipping 483	299.88
12	Transworld Shipping 463	297.55
13	Maritime Group 508	295.42
14	Seaway Goods 274	294.13
15	Harbor Services 174	288.71

Рис. 7.4.3 – Результат запиту на отримання клієнтів із загальною вагою перевезеного вантажу понад 50 тонн

4) Запит на отримання причалів у порту "Reykjavik", які можуть прийняти судно "Harbor Queen 533" відповідно до його вимог довжини та осадки

Результат можна побачити на рис. 7.4.4.

```

select berths.berthid,
       terminals.name   as Terminal_Name,
       berths.length    as Berth_Lenght,
       berths.depth     as Bearth_Depth
from berths
      join terminals on berths.terminalid = terminals.terminalid
      join ports on terminals.portid = ports.portid

where ports.portname = 'Reykjavik'

```

```

and berths.length >= (
    select ships.requiredlength
    from ships
    where ships.name = 'Harbor Queen 533'

)

```

```

and berths.depth >= (
    select ships.draft
    from ships
    where ships.name = 'Harbor Queen 533'

);

```

	berthid	terminal_name	berth_lenght	bearth_depth
1	767	Loading Dock 47	585.98	26.88
2	1162	Loading Dock 47	394.13	28.54
3	171	Loading Dock 47	361.40	15.95

Рис. 7.4.4 Результат запиту на отримання відповідних причалів для судна "Harbor Queen 533" у порту "Reykjavik"

5) Запит на отримання клієнтів, які мають скасовані відправлення, із підрахунком кількості таких відправлень.

Результат можна побачити на рис. 7.4.5.

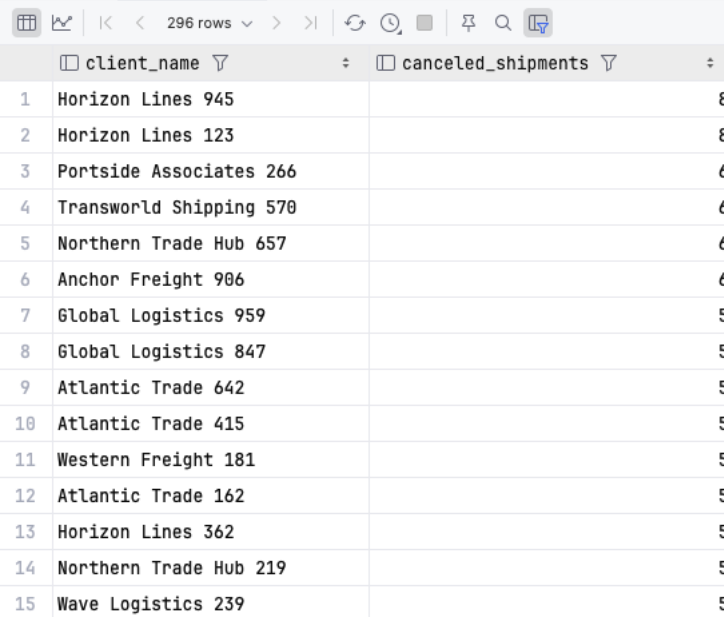
```

select clients.name           as Client_Name,
    count(shipments.shipmentid) as Canceled_Shipments
from clients
    join cargo on clients.clientid = cargo.clientid
    join shipments on cargo.cargoid = shipments.cargoid

where shipments.status = 'Canceled'
group by clients.name

```

having *count*(shipments.shipmentid) > 0
 order by Canceled_Shipments desc;



	client_name	canceled_shipments
1	Horizon Lines 945	8
2	Horizon Lines 123	8
3	Portside Associates 266	6
4	Transworld Shipping 570	6
5	Northern Trade Hub 657	6
6	Anchor Freight 906	6
7	Global Logistics 959	5
8	Global Logistics 847	5
9	Atlantic Trade 642	5
10	Atlantic Trade 415	5
11	Western Freight 181	5
12	Atlantic Trade 162	5
13	Horizon Lines 362	5
14	Northern Trade Hub 219	5
15	Wave Logistics 239	5

Рис. 7.4.5 – Результат запиту на отримання клієнтів із кількістю скасованих відправлень

б) Запит на отримання терміналів, де кількість активних працівників перевищує 70% від загальної кількості працівників.

Результат можна побачити на рис. 7.4.6.

```

select terminals.name          as Terminal_Name,
       activeworkers.total_active    as Active_Workers,
       totalworkers.total_count     as Total_Workers
from terminals
join (
       select terminalid,
              count(employeeid)    as Total_Active
       from employees
       where isactive = TRUE
       group by terminalid
) activeworkers on terminals.terminalid = activeworkers.terminalid

```

```

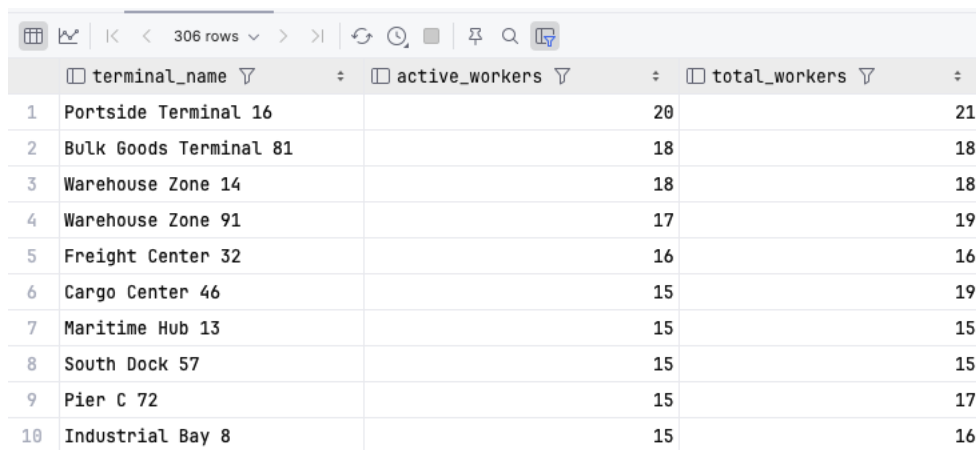
join (
    select terminalid,
           count(employeeid) as Total_Count
    from employees
    group by terminalid
) totalworkers on terminals.terminalid = totalworkers.terminalid

```

```

where activeworkers.Total_Active > (totalworkers.Total_Count * 0.7)
order by activeworkers.Total_Active desc;

```



	terminal_name	active_workers	total_workers
1	Portside Terminal 16	20	21
2	Bulk Goods Terminal 81	18	18
3	Warehouse Zone 14	18	18
4	Warehouse Zone 91	17	19
5	Freight Center 32	16	16
6	Cargo Center 46	15	19
7	Maritime Hub 13	15	15
8	South Dock 57	15	15
9	Pier C 72	15	17
10	Industrial Bay 8	15	16

Рис. 7.4.6 – Результат запиту на отримання терміналів із понад 70% активних працівників

7) Запит на отримання транспортних засобів із загальною кількістю виконаних перевезень.

Результат можна побачити на рис. 7.4.7.

```

select transporters.name as Transporter_Name,
       count(shipments.shipmentid) as Total_Shipments
from transporters
left join shipments on transporters.transporterid = shipments.transporterid
group by transporters.name

```

order by Total_Shipments desc;

	transporter_name	total_shipments
1	Prime Cargo 57	23
2	TransWorld Carriers 41	22
3	Freight Masters 18	21
4	Cargo Express 75	21
5	Global Transport 36	21
6	Metro Haul 52	21
7	Freight Masters 14	20
8	Speedy Logistics 8	20
9	Bridge Freight 84	19
10	RailLink Carriers 51	19
11	Portside Delivery 70	19
12	HaulPro 97	18
13	Highway Haulers 62	18
14	Cargo Express 2	18
15	HaulPro 13	18

Рис. 7.4.7 – Результат запиту на отримання транспортних засобів із кількістю виконаних перевезень

8) Запит на отримання загальної ваги вантажів, що зберігаються на кожному терміналі.

Результат запуску можна побачити на рис. 7.4.8.

```

select terminals.name      as Terminal_Name,
       sum(cargo.weight)   as Total_Stored_Weight
from storage
       left join cargo on storage.cargoid = cargo.cargoid
       left join terminals on storage.terminalid = terminals.terminalid
group by terminals.name
order by Total_Stored_Weight desc;
```

	terminal_name	total_stored_weight
1	Pier A 83	982.08
2	South Dock 69	816.78
3	North Yard 28	763.19
4	Pier B 38	754.54
5	West Terminal 12	752.47
6	Pier C 69	747.71
7	Maritime Hub 42	709.68
8	Pier C 41	705.94
9	Cargo Center 86	705.25
10	Cargo Center 59	693.02
11	Bulk Goods Terminal 42	691.05
12	Central Pier 47	687.92
13	Pier C 82	677.28
14	South Dock 30	676.42
15	East Dock 1	672.97

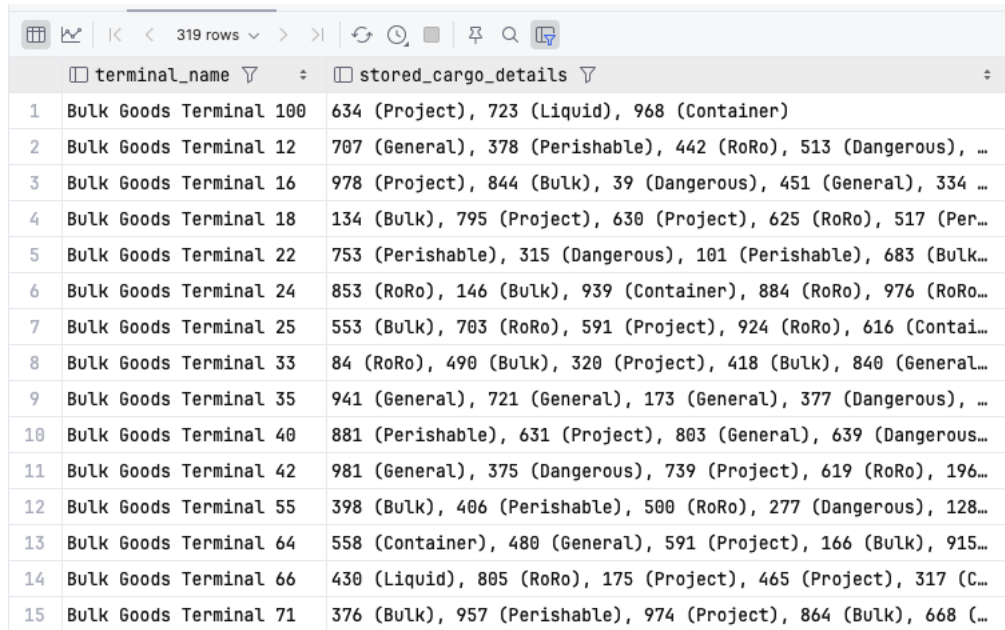
Рис. 7.4.8 – Результат запиту на отримання загальної ваги вантажів, що зберігаються на терміналах

9) Запит на отримання списку вантажів, що зберігаються на кожному терміналі із зазначенням їх типів.

Результат виконання запиту можна побачити на рис. 7.4.9.

```

select terminals.name                                as Terminal_Name,
       string_agg(concat(cargo.cargoid, '(', cargo.cargotype, ')), ', ') as
Stored_Cargo_Details
from storage
       join cargo on storage.cargoid = cargo.cargoid
       join terminals on storage.terminalid = terminals.terminalid
group by terminals.name
order by terminals.name;
```

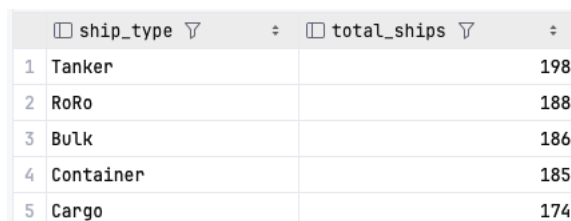
	terminal_name	stored_cargo_details
1	Bulk Goods Terminal 100	634 (Project), 723 (Liquid), 968 (Container)
2	Bulk Goods Terminal 12	707 (General), 378 (Perishable), 442 (RoRo), 513 (Dangerous), ...
3	Bulk Goods Terminal 16	978 (Project), 844 (Bulk), 39 (Dangerous), 451 (General), 334 ...
4	Bulk Goods Terminal 18	134 (Bulk), 795 (Project), 630 (Project), 625 (RoRo), 517 (Per...
5	Bulk Goods Terminal 22	753 (Perishable), 315 (Dangerous), 101 (Perishable), 683 (Bulk...
6	Bulk Goods Terminal 24	853 (RoRo), 146 (Bulk), 939 (Container), 884 (RoRo), 976 (RoRo...
7	Bulk Goods Terminal 25	553 (Bulk), 703 (RoRo), 591 (Project), 924 (RoRo), 616 (Contai...
8	Bulk Goods Terminal 33	84 (RoRo), 490 (Bulk), 320 (Project), 418 (Bulk), 840 (General...
9	Bulk Goods Terminal 35	941 (General), 721 (General), 173 (General), 377 (Dangerous), ...
10	Bulk Goods Terminal 40	881 (Perishable), 631 (Project), 803 (General), 639 (Dangerous...
11	Bulk Goods Terminal 42	981 (General), 375 (Dangerous), 739 (Project), 619 (RoRo), 196...
12	Bulk Goods Terminal 55	398 (Bulk), 406 (Perishable), 500 (RoRo), 277 (Dangerous), 128...
13	Bulk Goods Terminal 64	558 (Container), 480 (General), 591 (Project), 166 (Bulk), 915...
14	Bulk Goods Terminal 66	430 (Liquid), 805 (RoRo), 175 (Project), 465 (Project), 317 (C...
15	Bulk Goods Terminal 71	376 (Bulk), 957 (Perishable), 974 (Project), 864 (Bulk), 668 (...)

Рис. 7.4.9 – Результат запиту на отримання списку вантажів із зазначенням їх типів на терміналах

10) Запит на отримання кількості кораблів кожного типу, які мають розклад.

Результат можна побачити на рис. 7.4.10.

```
select ships.shiptype          as Ship_Type,
       count(distinct ships.shipid) as Total_Shops
from ships
       join schedules on ships.shipid = schedules.shipid
group by ships.shiptype
order by Total_Shops desc;
```



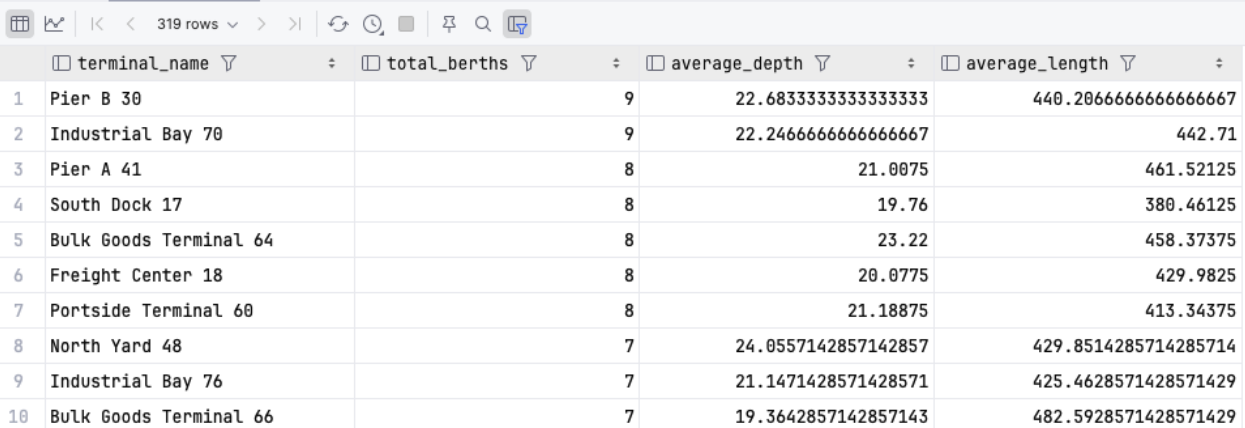
	ship_type	total_ships
1	Tanker	198
2	RoRo	188
3	Bulk	186
4	Container	185
5	Cargo	174

Рис. 7.4.10 – Результат запиту на отримання кількості кораблів кожного типу з розкладом

11) Запит на отримання інформації про кількість причалів, середню глибину та довжину причалів для кожного термінала.

Результат показано на рис. 7.4.11.

```
select terminals.name      as Terminal_Name,
       count(berths.berthid) as Total_Berths,
       avg(berths.depth)    as Average_Depth,
       avg(berths.length)   as Average_Length
from terminals
      join berths on terminals.terminalid = berths.terminalid
group by terminals.name
order by Total_Berths desc;
```



	terminal_name	total_berths	average_depth	average_length
1	Pier B 30	9	22.683333333333333	440.28666666666667
2	Industrial Bay 70	9	22.246666666666667	442.71
3	Pier A 41	8	21.0075	461.52125
4	South Dock 17	8	19.76	380.46125
5	Bulk Goods Terminal 64	8	23.22	458.37375
6	Freight Center 18	8	20.0775	429.9825
7	Portside Terminal 60	8	21.18875	413.34375
8	North Yard 48	7	24.055714285714285	429.85142857142857
9	Industrial Bay 76	7	21.147142857142857	425.46285714285714
10	Bulk Goods Terminal 66	7	19.364285714285714	482.59285714285714

Рис. 7.4.11 – Результат запиту на отримання кількості причалів, середньої глибини та довжини причалів для кожного термінала

12) Запит на отримання інформації про кораблі, їх причали, термінали та порти призначення, які відповідають вимогам по довжині та глибині причалів.

Результат показано на рис. 7.4.12.

```
SELECT ships.name      AS Ship_Name,
       berths.berthid,
       terminals.name   AS Terminal_Name,
       ports.portname   AS Destination_Port
```

FROM schedules

JOIN ships ON schedules.shipid = ships.shipid

JOIN berths ON schedules.berthid = berths.berthid

JOIN terminals ON berths.terminalid = terminals.terminalid

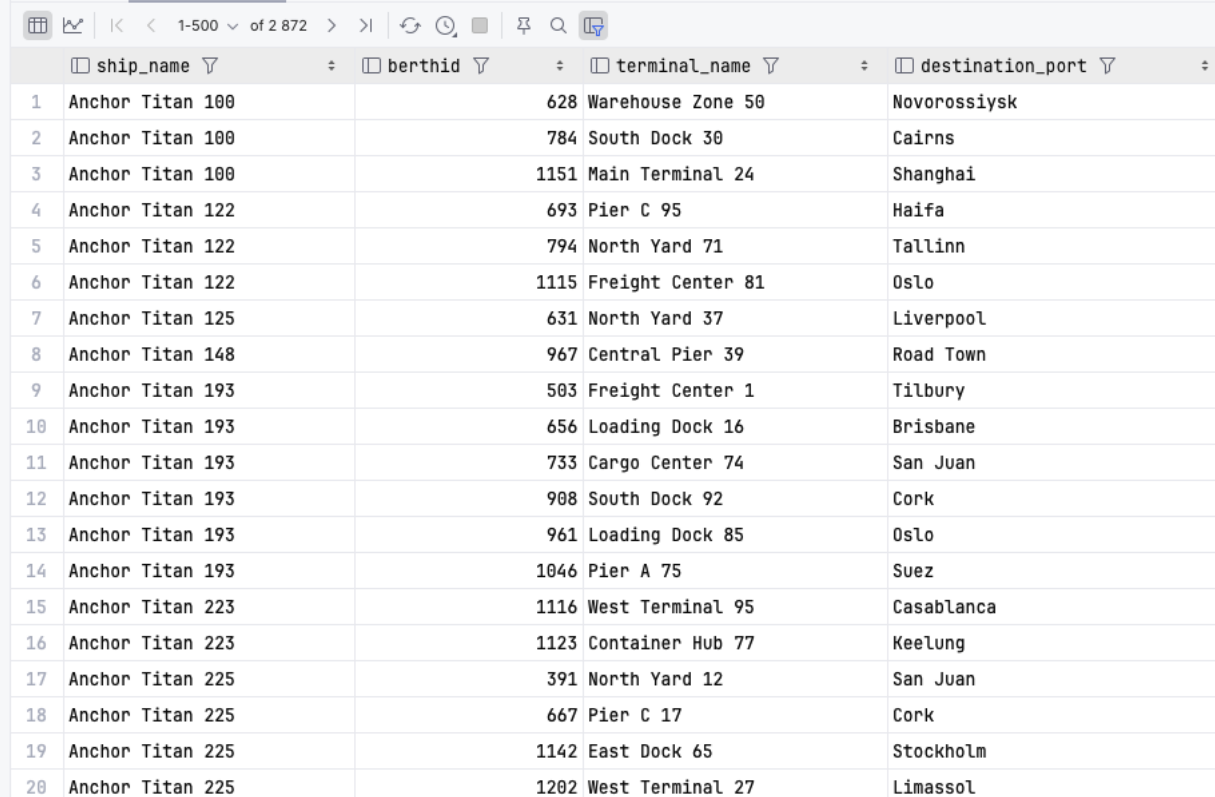
JOIN ports ON terminals.portid = ports.portid

WHERE ports.portid = schedules.destinationportid

AND berths.length >= ships.requiredlength

AND berths.depth >= ships.draft

ORDER BY ships.name, berths.berthid;



	ship_name	berthid	terminal_name	destination_port
1	Anchor Titan 100		628 Warehouse Zone 50	Novorossiysk
2	Anchor Titan 100		784 South Dock 30	Cairns
3	Anchor Titan 100		1151 Main Terminal 24	Shanghai
4	Anchor Titan 122		693 Pier C 95	Haifa
5	Anchor Titan 122		794 North Yard 71	Tallinn
6	Anchor Titan 122		1115 Freight Center 81	Oslo
7	Anchor Titan 125		631 North Yard 37	Liverpool
8	Anchor Titan 148		967 Central Pier 39	Road Town
9	Anchor Titan 193		503 Freight Center 1	Tilbury
10	Anchor Titan 193		656 Loading Dock 16	Brisbane
11	Anchor Titan 193		733 Cargo Center 74	San Juan
12	Anchor Titan 193		908 South Dock 92	Cork
13	Anchor Titan 193		961 Loading Dock 85	Oslo
14	Anchor Titan 193		1046 Pier A 75	Suez
15	Anchor Titan 223		1116 West Terminal 95	Casablanca
16	Anchor Titan 223		1123 Container Hub 77	Keelung
17	Anchor Titan 225		391 North Yard 12	San Juan
18	Anchor Titan 225		667 Pier C 17	Cork
19	Anchor Titan 225		1142 East Dock 65	Stockholm
20	Anchor Titan 225		1202 West Terminal 27	Limassol

Рис. 7.4.12 – Результат запиту на отримання інформації про кораблі, причали, термінали та порти призначення, які відповідають вимогам по довжині та глибині причалів

13) Запит на отримання інформації про кількість активних та неактивних працівників залежно від їхньої посади.

Результат показано на рис. 7.4.13.

```

SELECT employees.position                                AS Employee_Type,
       SUM(CASE WHEN employees.isactive = TRUE THEN 1 ELSE 0 END) AS
Active_Workers,
       SUM(CASE WHEN employees.isactive = FALSE THEN 1 ELSE 0 END) AS
Inactive_Workers
FROM employees
GROUP BY employees.position
ORDER BY Active_Workers DESC, Inactive_Workers DESC;

```

	□ employee_type ▾	÷	□ active_workers ▾	÷	□ inactive_workers ▾	÷
1	Worker		1734		79	
2	Admin		381		78	
3	Manager		380		93	
4	Supervisor		361		86	

Рис. 7.4.13 – Результат запиту на отримання кількості активних та неактивних працівників за їхніми посадами

14) Запит на отримання топ-10 клієнтів за кількістю доставлених відправлень протягом 2024 року.

Результат показано на рис. 7.4.14.

```

SELECT clients.name                                AS Client_Name,
       COUNT(shipments.shipmentid) AS Total_Delivered_Shipments
FROM clients
      JOIN cargo ON clients.clientid = cargo.clientid
      JOIN shipments ON cargo.cargoid = shipments.cargoid
WHERE shipments.status = 'Delivered'
      AND shipments.shipmentdate BETWEEN '2024-01-01' AND '2024-12-31'
GROUP BY clients.name
ORDER BY Total_Delivered_Shipments DESC
LIMIT 10;

```

	client_name ▾	total_delivered_shipments ▾
1	Blue Ocean Shipping 475	9
2	Sea Freight Inc. 274	8
3	Portside Associates 726	8
4	Northern Trade Hub 644	7
5	Anchor Freight 234	6
6	Oceanic Trade 566	6
7	Harbor Services 676	6
8	Sea Freight Inc. 500	6
9	Blue Ocean Shipping 586	6
10	Eastern Logistics 236	6

Рис. 7.4.14 – Результат запиту на отримання топ-10 клієнтів за кількістю доставлених відправлень у 2024 році

15) Запит на отримання загальної ваги та кількості відправлень для кожного типу вантажу.

Результат показано на рис. 7.4.15.

```
SELECT cargo.cargotype,
       SUM(cargo.weight)      AS Total_Weight,
       COUNT(shipments.shipmentid) AS Number_Of_Shipments
FROM cargo
      JOIN shipments ON cargo.cargoid = shipments.cargoid
GROUP BY cargo.cargotype;
```

	cargotype ▾	total_weight ▾	number_of_shipments ▾
1	Bulk	17590.36	339
2	Liquid	15193.4	310
3	Dangerous	14955.97	322
4	Perishable	10881.21	261
5	Container	15091.18	285
6	General	13912.41	303
7	RoRo	18603.44	342
8	Project	10766.95	216

Рис. 7.4.15 – Результат запиту на отримання загальної ваги та кількості відправлень для кожного типу вантажу

16) Запит на отримання ID вантажів та номерів телефонів клієнтів для вантажів, які залишаються на зберіганні без вказаної кінцевої дати.

Результат показано на рис. 7.4.16.

```
SELECT cargo.cargoid,
       clients.phonenumber
FROM cargo
      JOIN clients ON cargo.clientid = clients.clientid
      LEFT JOIN storage ON cargo.cargoid = storage.cargoid
WHERE storage.enddate IS NULL;
```

	cargoid	phononenumber
1	1	5299928144
2	84	4186946864
3	779	4004426572
4	732	1800354263
5	936	4186946864
6	446	2159009343
7	510	8817271556
8	268	7269560878
9	942	9712441161
10	141	5045059614
11	47	4112072381
12	731	2490113432
13	336	8370756333
14	684	3283219706
15	371	1926205005

Рис. 7.4.16 – Результат запиту на отримання ID вантажів та номерів телефонів клієнтів для вантажів, які залишаються на зберіганні та не мають кінцевої дати

17) Запит на отримання інформації про термінали, їхню загальну місткість та вагу небезпечних вантажів, які вони зберігають.

Результат показано на рис. 7.4.17.

```
SELECT terminals.name      AS Terminal_Name,
       terminals.capacity  AS Terminal_Capacity,
       SUM(cargo.weight)   AS Total_Dangerous_Weight
```

FROM terminals

JOIN storage ON terminals.terminalid = storage.terminalid

JOIN cargo ON storage.cargoid = cargo.cargoid

WHERE cargo.cargotype = 'Dangerous'

GROUP BY terminals.terminalid, terminals.name, terminals.capacity;

	terminal_name	terminal_capacity	total_dangerous_weight
1	Logistics Terminal 45	6643.61	9.74
2	Freight Center 18	12876.26	109.05
3	West Terminal 46	17371.54	53.58
4	Pier B 91	8845.49	105.92
5	Bulk Goods Terminal 33	4060.59	125.75
6	Pier C 69	19335.41	14.74
7	East Dock 20	15757.72	70.11
8	Bulk Goods Terminal 42	6771.12	89.09
9	Maritime Hub 95	12617.47	58.17
10	South Dock 86	14803.04	79.54
11	Freight Center 25	7125.46	85.01
12	Pier A 41	16518.31	8.36
13	Freight Center 9	4120.74	64.47
14	Pier C 20	18016.56	110.12
15	Main Terminal 75	4279.84	86.64
16	Warehouse Zone 52	18837.82	106.66
17	Logistics Terminal 76	6705.79	66.67
18	West Terminal 42	11284.15	82.71
19	Warehouse Zone 91	17202.88	77.96
20	Main Terminal 82	9026.70	144.33

Рис. 7.4.17 – Результат запиту на отримання інформації про термінали, їхню місткість та вагу небезпечних вантажів

18) Запит на отримання інформації про причали, які знаходяться на технічному обслуговуванні.

Результат показано на рис. 7.4.18.

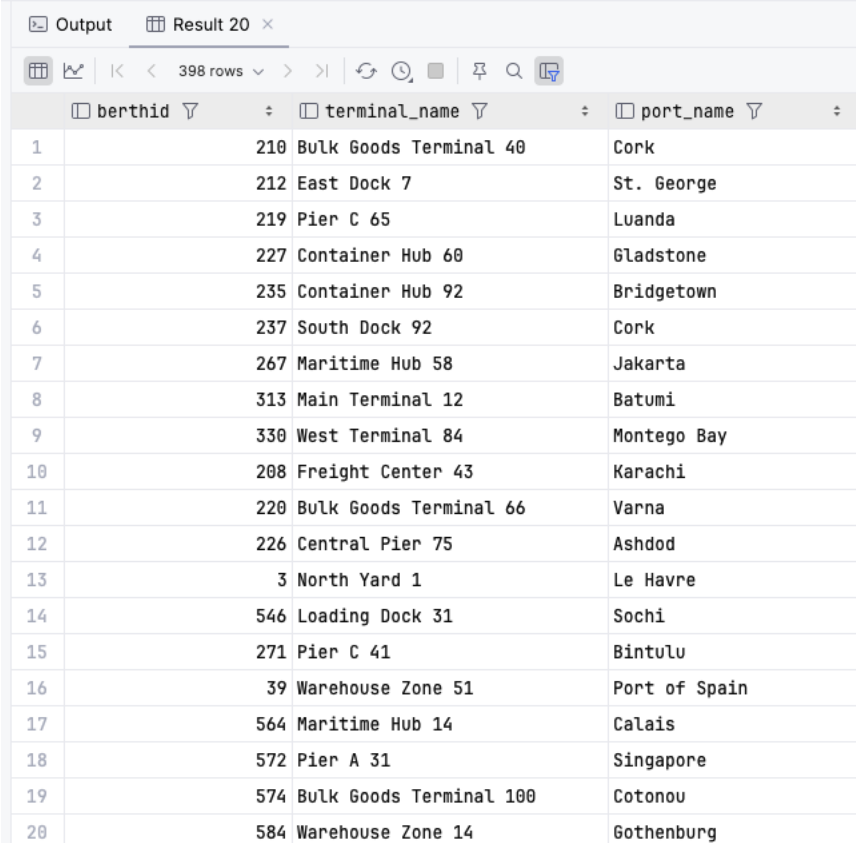
SELECT berths.berthid,

terminals.name AS Terminal_Name,

```

ports.portname      AS Port_Name
FROM berths
      JOIN terminals ON berths.terminalid = terminals.terminalid
      JOIN ports ON terminals.portid = ports.portid
WHERE berths.status = 'Under Maintenance';

```



berthid	terminal_name	port_name
1	210 Bulk Goods Terminal 40	Cork
2	212 East Dock 7	St. George
3	219 Pier C 65	Luanda
4	227 Container Hub 60	Gladstone
5	235 Container Hub 92	Bridgetown
6	237 South Dock 92	Cork
7	267 Maritime Hub 58	Jakarta
8	313 Main Terminal 12	Batumi
9	330 West Terminal 84	Montego Bay
10	208 Freight Center 43	Karachi
11	220 Bulk Goods Terminal 66	Varna
12	226 Central Pier 75	Ashdod
13	3 North Yard 1	Le Havre
14	546 Loading Dock 31	Sochi
15	271 Pier C 41	Bintulu
16	39 Warehouse Zone 51	Port of Spain
17	564 Maritime Hub 14	Calais
18	572 Pier A 31	Singapore
19	574 Bulk Goods Terminal 100	Cotonou
20	584 Warehouse Zone 14	Gothenburg

Рис. 7.4.18 – Результат запиту на отримання інформації про причали на технічному обслуговуванні.

19) Запит на отримання інформації про кораблі, тривалість перебування яких у портах була найбільшою.

Результат показано на рис. 7.4.19.

```

SELECT ships.name      AS Ship_Name,
      ports.portname    AS Port_Name,

```



```

schedules.arrivaldate      AS Arrival_Date,

schedules.departuredate    AS Departure_Date,

extract(epoch FROM (schedules.departuredate - schedules.arrivaldate)) / 3600

      AS Hours_In_Port

FROM schedules

      JOIN ships ON schedules.shipid = ships.shipid

      JOIN terminals ON schedules.terminalid = terminals.terminalid

      JOIN ports ON terminals.portid = ports.portid

ORDER BY Hours_In_Port DESC

LIMIT 10;

```

	ship_name ▾	port_name ▾	arrival_date ▾	departure_date ▾	hours_in_port ▾
1	Freight King 623	Kingston	2024-07-16 10:00:00.000000	2024-07-26 10:00:00.000000	240
2	Global Carrier 207	Basse-Terre	2024-12-15 03:44:00.000000	2024-12-25 03:44:00.000000	240
3	Blue Horizon 287	Kaohsiung	2024-04-28 18:52:00.000000	2024-05-08 18:52:00.000000	240
4	Maritime Spirit 883	Shanghai	2024-01-13 18:59:00.000000	2024-01-23 18:59:00.000000	240
5	Global Carrier 959	Khabarovsk	2024-05-25 05:02:00.000000	2024-06-04 05:02:00.000000	240
6	Pacific Mariner 909	Castries	2024-07-12 02:07:00.000000	2024-07-22 02:07:00.000000	240
7	Oceanic Pioneer 777	Gladstone	2024-06-13 04:50:00.000000	2024-06-23 04:50:00.000000	240
8	Poseidons Grace 958	Port Klang	2024-04-08 02:01:00.000000	2024-04-18 02:01:00.000000	240
9	Global Carrier 748	Road Town	2024-09-02 16:49:00.000000	2024-09-12 16:49:00.000000	240
10	Maritime Spirit 757	Helsinki	2024-03-10 00:30:00.000000	2024-03-20 00:30:00.000000	240

Рис. 7.4.19 – Результат запиту на отримання інформації про кораблі з найбільшою тривалістю перебування у портах

20) Запит на визначення перевантажених кораблів та обчислення обсягу перевантаження.

Результат показано на рис. 7.4.20.

```

SELECT ships.name      AS Ship_Name,

```

```

ships.capacity          AS Ship_Capacity,

SUM(cargo.weight)       AS Total_Cargo_Weight,

(SUM(cargo.weight) - ships.capacity) AS Overload

FROM ships

JOIN schedules ON ships.shipid = schedules.shipid

JOIN terminals ON schedules.terminalid = terminals.terminalid

JOIN storage ON terminals.terminalid = storage.terminalid

JOIN cargo ON storage.cargoid = cargo.cargoid

GROUP BY ships.shipid, ships.name, ships.capacity

HAVING SUM(cargo.weight) > ships.capacity;

```

	ship_name ▾	÷	ship_capacity ▾	÷	total_cargo_weight ▾	÷	overload ▾	÷
1	Merchant Wave 149		3362.47		3794.93		432.46	
2	Anchor Titan 997		1534.78		2040.43		505.65	
3	Neptunes Pride 252		1648.69		2356.78		708.09	
4	Pacific Mariner 542		2900.10		2980.15		80.05	
5	Neptunes Pride 808		2387.66		3893.87		1506.21	
6	Freight King 984		1861.56		2679.71		818.15	
7	Blue Horizon 628		1893.84		2133.51		239.67	
8	Freight King 926		1119.99		1808.39		688.4	
9	Pacific Mariner 574		912.70		1259.92		347.22	
10	Trade Wind 109		1763.23		1925.86		162.63	
11	Merchant Wave 315		864.13		1469.1		604.97	
12	Neptunes Pride 821		1968.48		2578.26		609.78	
13	Sea Star 957		1214.15		1912.64		698.49	
14	Harbor Queen 560		1286.70		1998.7		712	
15	Cargo Titan 621		808.77		899.09		90.32	
16	Cargo Titan 941		933.97		1490.93		556.96	
17	Liberty Sailor 632		1434.02		2039.41		605.39	
18	Sea Breeze 936		2076.00		2859.85		783.85	
19	Oceanic Pioneer 700		2316.47		2675.09		358.62	
20	Maritime Spirit 267		1807.40		1938.42		131.02	

Рис. 7.4.20 – Результат запиту на визначення перевантажених кораблів та обчислення обсягу перевантаження

7.5 Індeksi та результати оптимізації

В процесі оптимізації запитів у базі даних особливу увагу було приділено покращенню продуктивності складних операцій, які включають з'єднання таблиць із великим обсягом даних. Одним із таких запитів є

обчислення перевантаження кораблів, що вимагає з'єднання таблиць ships, schedules, terminals, storage та cargo. Такий запит може мати тривалість виконання через значний обсяг даних та складність з'єднань. Результат його виконання без оптимізації можна побачити на рис. 7.5.1.

[2024-12-22 05:46:14] 20 rows retrieved starting from 1 in 54 ms (execution: 28 ms, fetching: 26 ms)

Рис. 7.5.1 – Час виконання запиту без оптимізації

Для покращення продуктивності було додано індекси до ключових стовпців, які часто використовуються у з'єднаннях і умовах запитів. Індекси дозволяють скоротити кількість операцій сканування таблиць, забезпечуючи швидший доступ до даних та зменшуючи час виконання запитів.

Зокрема, було створено індекси для ідентифікаторів кораблів, вантажів, терміналів, а також для ваги вантажів. Вони прискорили доступ до даних під час виконання з'єднань між таблицями та оптимізували процес підрахунку агрегованих значень. Також було враховано, що індексація є важливим аспектом для баз даних із динамічно зростаючим обсягом інформації, що дозволяє уникнути затримок у майбутньому.

Код для створення індексів:

```
create index idx_ships_shipid on ships(shipid);

create index idx_schedules_shipid on schedules(shipid);

create index idx_storage_cargoid on storage(cargoid);

create index idx_cargo_weight on cargo(weight);

create index idx_terminals_terminalid on terminals(terminalid);
```

Однак у поточній базі даних розмір і обсяг таблиць залишаються відносно невеликими, що зменшує вплив індексації на продуктивність. Проведені вимірювання часу виконання запиту до і після додавання індексів продемонстрували незначну різницю у продуктивності через невелику кількість записів у таблицях. Це пояснюється тим, що на малих обсягах даних

час сканування таблиці залишається прийнятним навіть без індексації. Результат після застосування індексації можна побачити на рис. 7.5.2.

[2024-12-22 05:44:22] 20 rows retrieved starting from 1 in 37 ms (execution: 23 ms, fetching: 14 ms)

Рис. 7.5.1 – Час виконання запиту з оптимізацією

Незважаючи на те, що на поточному розмірі бази даних індексація не дала значного приросту швидкості, це важливий крок для забезпечення масштабованості системи. У міру зростання обсягів даних індекси відіграватимуть важливу роль у підтримці стабільної продуктивності. Правильний вибір стовпців для індексації дозволяє заздалегідь підготувати базу даних до збільшення навантаження, забезпечуючи її ефективне функціонування в майбутньому.

ВИСНОВКИ

У межах курсової роботи було виконано повний цикл розробки інформаційної системи для управління морським вантажним терміналом. Основною метою роботи було створення ефективної бази даних, яка б відповідала сучасним вимогам автоматизації бізнес-процесів у морських портах. У результаті проведених досліджень і реалізації завдань можна зробити такі висновки.

Було здійснено аналіз предметного середовища, що дозволило визначити ключові сутності, бізнес-правила та взаємозв'язки, які мають враховуватися в інформаційній системі. На основі цього аналізу побудовано ER-модель, яка точно відображає структуру даних і забезпечує логічну цілісність системи.

Реалізована реляційна модель бази даних включає всі необхідні сутності та атрибути, які відображають основні процеси терміналу: облік вантажів, управління судновими розкладами, моніторинг роботи терміналів та зберігання вантажів. Особливу увагу було приділено забезпеченню цілісності даних за допомогою використання зовнішніх ключів, обмежень та перевірок.

Розроблені бізнес-правила і тригери дозволили автоматизувати перевірку відповідності введених даних реальним вимогам, що значно підвищує надійність роботи системи. Наприклад, було впроваджено механізми контролю завантаження терміналів і суден, перевірки безпечної ваги небезпечних вантажів та забезпечення коректності розкладів суден. Це дозволяє уникнути помилок і збоїв у роботі терміналу.

У ході роботи було розроблено ряд SQL-запитів і представлень, які дозволяють отримувати оперативну інформацію про стан вантажів, розклади суден, завантаженість терміналів та інші ключові показники роботи. Це сприяє підвищенню ефективності управління терміналом і прийняття оперативних рішень.

Також було проведено оптимізацію бази даних за допомогою індексів, що значно покращило швидкодію запитів для великих обсягів даних. Хоча поточний розмір бази не демонструє критичних затримок, підготовка до можливого збільшення даних є важливим кроком для забезпечення майбутньої масштабованості системи.

Проведене дослідження підтверджує важливість комплексного підходу до розробки баз даних для складних бізнес-процесів. Запропонована інформаційна система є гнучкою, масштабованою та адаптованою до потреб користувачів, що робить її ефективним інструментом для автоматизації роботи морського вантажного терміналу.

Таким чином, результати цієї роботи можуть бути використані як основа для подальшого вдосконалення системи, включаючи інтеграцію з іншими інформаційними системами, розширення функціональності та впровадження сучасних технологій, таких як машинне навчання та прогнозна аналітика.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Port of Rotterdam Official Website URL: <https://www.portofrotterdam.com/en>
- 2) European Sea Ports Organisation (ESPO) URL: <https://www.espo.be/>
- 3) International Association of Ports and Harbors (IAPH) URL: <https://www.iaphworldports.org/>
- 4) The Global Economic Significance of Ports URL: <https://porteconomicsmanagement.org/pemp/contents/part11/port-and-economic-development/>
- 5) Körber Supply Chain URL: <https://koerber-supplychain.com/>
- 6) CyberLogitec OPUS Terminal URL: https://www.cyberlogitec.com/en/sub/solution/port/opus_terminal.php
- 7) DP World's CARGOES TOS URL: <https://www.dpworld.com/digital-solutions/terminal-operating-system>
- 8) Port-IO URL: <https://www.getport.io/>
- 9) PostgreSQL 17.2 Documentation URL: <https://www.postgresql.org/docs/current/index.html>