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 реєстру укриттів та планів евакуації міста

Студента (ки) __2__ курсу _ІП-45_____ групи
спеціальності 121 «Інженерія програмного забезпечення»
_Янова Богдана Євгенійовича_____
(прізвище та ініціали)

Керівник _Ліщук Катерина Ігорівна_____
_____
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____

Кількість балів: _____ Оцінка _____

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 – 2025 рік

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

Факультет Інформатики та обчислювальної техніки
(повна назва)

Кафедра Інформатики та програмної інженерії
(повна назва)

Дисципліна Бази даних

Курс 2 Група ІП-45 Семестр 3

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

Янову Богдану Євгенійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи База даних реєстру укриттів та планів евакуації міста

керівник роботи Ліщук Катерина Ігорівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи 15.12.2025

3. Вихідні дані до роботи
розробити базу даних реєстру укриттів та планів евакуації міста

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 26.10.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання курсового проекту	Строк виконання етапів проекту	Примітка
1	Аналіз предметного середовища	01.11.2025	
2	Побудова ER-моделі	01.11.2025	
3	Побудова реляційної схеми з ER-моделі	01.11.2025	
4	Створення бази даних, у форматі обраної системи управління базою даних	01.11.2025	
5	Створення користувачів бази даних	01.11.2025	
6	Імпорт даних з використанням засобів СУБД в створену базу даних	01.11.2025	
7	Створення мовою SQL запитів	01.11.2025	
8	Оптимізація роботи запитів	01.11.2025	
9	Оформлення пояснювальної записки	15.12.2025	
10	Захист курсової роботи	18.12.2025	

Студент _____ Янов Б.Є.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ліщук К.І.
(підпис) (прізвище та ініціали)

ЗМІСТ

ЗМІСТ **4**

ВСТУП **5**

1 АНАЛІЗ ПРЕДМЕТНОГО СЕРЕДОВИЩА **6**

1.1 Опис предметного середовища 6
1.2 Аналіз існуючих програмних продуктів 8

2 ПОСТАНОВКА ЗАВДАННЯ **10**

3 ПОБУДОВА ER-МОДЕЛІ **12**

3.1 Бізнес-правила 12
3.2 Вибір сутностей 12
3.3 Опис сутностей 13
3.4 Опис зв'язків 16
3.5 ER-Діаграма 17

4 РЕЛЯЦІЙНА МОДЕЛЬ БАЗИ ДАНИХ **19**

5 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ **24**

6 СТВОРЕННЯ КОРИСТУВАЧІВ **31**

7 РОБОТА З БАЗОЮ ДАНИХ **33**

7.1 Функції та процедури 33
7.2 Триггери 45
7.3 Представлення 50
7.4 Select-запити 53

ДЖЕРЕЛА **67**

ВИСНОВКИ **68**

ВСТУП

У сучасних умовах, коли питання безпеки цивільного населення набуває пріоритетного значення, ефективна організація системи цивільного захисту стає критично важливою задачею для органів місцевого самоврядування. Забезпечення громадян надійними укриттями та чіткими планами евакуації вимагає використання сучасних інформаційних технологій для збору, обробки та оперативного надання даних.

Актуальність теми «База даних реєстру укриттів та планів евакуації міста» обумовлена необхідністю відмовитися від застарілих паперових носіїв та розрізнених таблиць на користь централізованої системи. В умовах воєнного стану та постійних загроз повітряних атак, швидкість доступу до інформації про найближче відкрите сховище може коштувати людських життів. Високий обсяг динамічних даних, таких як поточний статус готовності укриттів, наявність вільних місць та матеріальних запасів, вимагає ефективних інструментів управління.

Метою даної курсової роботи є розробка структури бази даних для автоматизованої системи управління фондом захисних споруд та планування евакуаційних заходів з використанням системи управління базами даних PostgreSQL.

Задачі роботи включають у себе ретельний аналіз предметного середовища цивільного захисту, проектування оптимальної ER-моделі, що враховує географічну прив'язку об'єктів, розробку реляційної схеми та впровадження бізнес-логіки для автоматизації процесів моніторингу та маршрутизації.

Призначенням розробки є підвищення рівня безпеки населення шляхом забезпечення прозорості даних про стан укриттів, спрощення роботи інспекторів та балансоутримувачів, а також надання громадянам інструментів для побудови безпечних піших маршрутів. Сфера використання охоплює департаменти муніципальної безпеки, служби ДСНС та публічні сервіси оповіщення населення.

Обґрунтування вибору PostgreSQL полягає в її потужних можливостях роботи з геопросторовими даними завдяки розширенню PostGIS, що є ключовим для розрахунку дистанцій та побудови маршрутів. Ця СУБД забезпечує високу надійність, відповідність стандартам ACID та підтримує складні аналітичні запити, що є критичним для систем безпеки. Крім того, PostgreSQL є системою з відкритим кодом, що дозволяє використовувати її без додаткових ліцензійних витрат.

Отже, курсова робота спрямована на створення надійної основи для інформаційної системи, що здатна адаптуватися до змінних умов небезпеки та забезпечувати координацію дій усіх учасників процесу цивільного захисту.

1 АНАЛІЗ ПРЕДМЕТНОГО СЕРЕДОВИЩА

1.1 Опис предметного середовища

В умовах сучасних викликів безпеки, зокрема військової агресії та постійної загрози повітряних атак, питання цивільного захисту населення набуває критичного значення. Ефективність системи цивільного захисту міста безпосередньо залежить від якості інформаційного забезпечення, оперативності прийняття рішень та наявності актуальних даних про стан захисних споруд. Предметним середовищем даної курсової роботи є система обліку, управління та моніторингу фонду захисних споруд (укриттів) міста, а також планування безпечних маршрутів евакуації населення до цих споруд.

Процес управління укриттями є складним та багатокомпонентним. Він починається з інвентаризації та категоризації об'єктів. Захисні споруди можуть бути різного типу: від спеціалізованих сховищ та протирадіаційних укриттів до найпростіших укриттів (підвалів житлових будинків, підземних паркінгів) та споруд подвійного призначення. Кожен такий об'єкт має низку технічних характеристик, які визначають його здатність захистити від конкретних видів загроз (артилерійський обстріл, авіаудар, радіаційне забруднення тощо). Критично важливим параметром є місткість споруди, оскільки саме вона визначає кількість людей, які можуть знайти там порятунок, та впливає на розрахунок планів евакуації конкретного мікрорайону.

Важливим аспектом предметного середовища є розмежування майнових прав та експлуатаційної відповідальності. Часто виникає ситуація, коли об'єкт нерухомості (укриття) перебуває на балансі однієї організації (наприклад, міської ради або комунального підприємства), але фактичний контроль за доступом та поточним станом здійснює інша особа чи структура. Наприклад, підвальне приміщення може належати місту, але ключі від нього, обов'язок відчиняти двері під час тривоги та підтримувати санітарний стан покладено на майстра дільниці ЖЕКу, голову ОСББ або чергового вахтера установи. Система повинна чітко фіксувати не лише юридичного власника, а й конкретну відповідальну особу з контактними даними, що дозволить оперативно вирішувати проблеми "зачинених дверей" під час надзвичайних ситуацій.

Функціонування укриття неможливе без належного матеріально-технічного забезпечення. Предметна область охоплює облік інвентарю, необхідного для життєзабезпечення: запаси питної та технічної води, продукти харчування тривалого зберігання, медикаменти, засоби пожежогасіння, місця для сидіння/лежання, а також засоби автономного енергозабезпечення (генератори, паливо). Кожна одиниця інвентарю має свій термін придатності та необхідну кількість відповідно до норм на одну особу.

Це вимагає постійного моніторингу: відповідальні особи повинні вчасно оновлювати запаси, а контролюючі органи — перевіряти їх наявність.

Контроль за станом укриттів здійснюється через механізм регулярних перевірок (інспекцій). Уповноважені інспектори (представники ДСНС, муніципальної варти або відділу цивільного захисту) періодично відвідують об'єкти, оцінюючи їх технічний стан, доступність, справність комунікацій (вентиляція, каналізація, електромережа) та відповідність нормам. Результати таких перевірок фіксуються у вигляді актів, які визначають поточний статус укриття: готове до використання, обмежено готове або не готове. Історія перевірок дозволяє відстежувати динаміку змін та виявляти системні проблеми в обслуговуванні конкретних районів чи типів споруд.

Окремим, надзвичайно важливим пластом предметної області є логістика евакуації. В умовах щільної міської забудови та високої імовірності транспортного колапсу або небезпеки пересування транспортом під час обстрілів, основним способом евакуації до укриттів розглядається піший хід. Це накладає специфічні вимоги до побудови маршрутів. Традиційна концепція "точок збору", де люди чекають на транспорт, у даному середовищі є неефективною та небезпечною, оскільки скупчення людей на відкритій місцевості стає мішенню.

Натомість, логіка евакуації будується на основі динамічних маршрутів, які пролягають від місць проживання громадян (адресних точок) до найближчого доступного укриття відповідного типу. Маршрут являє собою послідовність проміжних точок або орієнтирів, що ведуть до безпечного місця. Система повинна враховувати відстань, яку людині доведеться подолати пішки, оскільки час підльоту ракет або снарядів часто вимірюється хвилинами. Також маршрути можуть змінюватися в залежності від статусу укриття (якщо основне укриття переповнене або пошкоджене, потік людей має бути перенаправлений до резервного).

Крім того, необхідно враховувати специфіку загроз. Різні типи небезпеки вимагають різних типів захисту. Наприклад, у випадку загрози застосування ядерної зброї, найпростіші укриття (перший поверх, звичайні підвали) можуть бути неефективними, і маршрути евакуації мають бути перебудовані виключно до протирадіаційних сховищ або станцій метрополітену глибокого залягання, навіть якщо вони знаходяться на більшій відстані.

Таким чином, інформаційна система в даному предметному середовищі виступає сполучною ланкою між адміністрацією міста, балансоутримувачами, інспекторами та населенням. Вона повинна забезпечувати актуалізацію даних про фонд захисних споруд, контроль за їхнім станом та матеріальним забезпеченням, а також надавати алгоритми для побудови оптимальних

пішохідних маршрутів порятунку, мінімізуючи час знаходження людей на небезпечній території. Відсутність централізованої бази даних призводить до хаосу, використання застарілої інформації та, як наслідок, до підвищення ризиків для життя громадян. [1, 2]

1.2 Аналіз існуючих програмних продуктів

На сьогоднішній день існує ряд програмних рішень, які частково покривають функціонал обліку укриттів та оповіщення населення. Проте, комплексний аналіз виявляє певні обмеження кожного з них у контексті поставлених завдань, особливо щодо внутрішнього менеджменту інвентаризації та специфічної маршрутизації.

Одним із найбільш відомих рішень є державний портал та додаток «Дія» (або його інтеграція з інтерактивними картами укриттів). Це рішення надає користувачам можливість бачити найближчі укриття на мапі, переглядати їх адресу, вид та фотографії.

Переваги: Широке охоплення аудиторії, зручний інтерфейс для кінцевого користувача, інтеграція з державними реєстрами.

Недоліки: Система орієнтована насамперед на відображення інформації, а не на управління процесами забезпечення. У ній відсутній детальний функціонал для балансоутримувачів щодо обліку запасів води, їжі чи палива для генераторів. Також відсутня гнучка система для інспекторів, яка б дозволяла планувати перевірки та зберігати їх детальну історію з прив'язкою до конкретних відповідальних осіб, а не просто установ. Маршрутизація реалізована через стандартні API картографічних сервісів без урахування специфіки евакуаційних коридорів [3, 4].

Другою групою продуктів є геоінформаційні системи (ГІС) загального призначення, такі як Google Maps або OpenStreetMap. Ці системи дозволяють наносити шари з об'єктами (укриттями) та будувати до них маршрути.

Переваги: Потужні алгоритми розрахунку відстаней та часу руху, висока точність геопозиціонування, можливість візуалізації великих масивів даних.

Недоліки: Це універсальні інструменти, які не мають вбудованої бізнес-логіки цивільного захисту. Вони не можуть автоматично змінювати доступність укриття в залежності від типу тривоги (наприклад, фільтрувати укриття при радіаційній загрозі). ГІС не призначені для ведення господарського обліку (інвентаризації) або кадрового обліку відповідальних осіб. Крім того, побудова маршрутів у них часто орієнтована на автомобільний або громадський транспорт, що суперечить вимозі щодо пріоритету пішої евакуації в умовах небезпеки [5, 6].

Існують також спеціалізовані системи управління майном (Asset Management Systems), які використовуються комунальними підприємствами. Вони чудово справляються з інвентаризацією та обліком технічного стану будівель.

Переваги: Детальний облік матеріальних цінностей, планування ремонтів, фіксація відповідальних.

Недоліки: Такі системи зазвичай є закритими корпоративними рішеннями, недоступними для публічного використання громадянами. Вони не мають функціоналу картографії та побудови евакуаційних маршрутів. Інтеграція таких систем з публічними сервісами є складною та дороговартісною. Крім того, вони часто перевантажені зайвим бухгалтерським функціоналом, що ускладнює роботу інспекторів цивільного захисту.

Отже, аналіз існуючих рішень показує, що жодне з них не задовольняє повною мірою специфічні потреби предметної області, які поєднують в собі:

1. Публічний доступ та гео-навігацію для населення (як у «Дії» чи Google Maps).
2. Глибокий господарський облік та інвентаризацію (як у ERP-системах).
3. Специфічну логіку розмежування відповідальності між власником та утримувачем.
4. Адаптивну систему пішої маршрутизації в залежності від типу загрози.

Це обґрунтовує необхідність розробки власної бази даних та інформаційної системи, яка б інтегрувала ці аспекти в єдине середовище, забезпечуючи надійність даних для адміністрації та безпеку для громадян.

2 ПОСТАНОВКА ЗАВДАННЯ

Метою даної курсової роботи є розробка та реалізація бази даних для ведення єдиного реєстру захисних споруд цивільного захисту та інформаційної підтримки процесів евакуації населення міста. Основними напрямками роботи є автоматизація обліку фонду захисних споруд, моніторинг їх технічного стану та матеріального забезпечення, а також моделювання та збереження оптимальних маршрутів евакуації залежно від типів загроз. Задачі роботи:

1. аналіз предметного середовища;
2. побудова ER-моделі, що відповідає потребам управління цивільним захистом міста;
3. побудова реляційної схеми на підставі розробленої ER-моделі;
4. створення бази даних, що була спроектована, з використанням СУБД PostgreSQL;
5. імпорт даних з використанням засобів СУБД в створену базу даних;
6. з використанням мови SQL автоматизація ключових бізнес-процесів, таких як реєстрація та категоризація укриттів, облік відповідальних осіб та балансоутримувачів, управління інвентаризацією запасів, фіксація результатів інспекційних перевірок, розрахунок та побудова піших евакуаційних маршрутів, а також фільтрація доступних сховищ відповідно до рівня та типу поточної загрози;
7. оптимізація роботи запитів;
8. підтримка багатокористувальницького доступу.

База даних повинна відповідати стандартам ефективного управління критичною інфраструктурою міста. Вона повинна бути стійкою до навантажень, забезпечувати швидкий пошук геопросторових даних та гарантувати цілісність інформації. Дані повинні зберігатися відповідно до вимог щодо розмежування доступу, враховуючи стратегічне значення об'єктів.

База даних включатиме в себе низку таблиць, що описують різноманітні аспекти системи цивільного захисту. Серед ключових сутностей можна виділити: укриття, їх адреси та характеристики, наявний інвентар та ресурси, типи загроз, маршрути евакуації, відповідальних осіб, організації-балансоутримувачі, інспекторів та акти перевірок. Також важливою є організація зв'язків між загрозами та здатністю конкретних типів укриттів захистити від них.

Отже, розробка та впровадження бази даних реєстру укриттів та планів евакуації сприятиме підвищенню готовності міста до надзвичайних ситуацій. Очікується, що вона стане ефективним інструментом для органів місцевого самоврядування та відділів цивільного захисту, дозволяючи оперативно

приймати рішення щодо перенаправлення потоків населення та доукомплектування захисних споруд. Розробка цієї моделі є кроком до створення безпечного міського середовища в умовах сучасних викликів.

3 ПОБУДОВА ER-МОДЕЛІ

3.1 Бізнес-правила

На основі аналізу предметного середовища було сформульовано наступні бізнес-правила, які обмежують та структурують дані в системі:

1. Кожна адреса (будівля) належить до одного конкретного району міста.
2. Адреса повинна мати географічні координати (широта та довгота) для забезпечення роботи алгоритмів маршрутизації.
3. Укриття розташоване за однією основною адресою, але може мати кілька входів, які прив’язані до різних адресних точок (наприклад, різні сторони будівлі).
4. Кожне укриття характеризується визначеною місткістю, поточним статусом готовності та типом (сховище, ПРУ, найпростіше укриття).
5. Укриття перебуває на балансі однієї організації, але за його відкриття та стан відповідає конкретна фізична особа.
6. Відповідальна особа повинна бути співробітником або мати зв'язок з організацією-балансоутримувачем.
7. Укриття може захищати від багатьох типів загроз, і кожен тип загрози може покриватися багатьма укриттями (зв'язок багато-до-багатьох).
8. Укриття може мати набір особливостей (інтернет, генератор, пандус), наявність яких фіксується у системі.
9. Облік матеріальних цінностей (води, їжі, палива) ведеться для кожного укриття окремо із зазначенням наявної кількості та терміну придатності (де це застосовно).
10. Інспектори можуть проводити багато перевірок різних укриттів; кожна інспекція завершується складанням акту з фінальним статусом та примітками.
11. Маршрут евакуації прив'язаний до одного цільового укриття і складається з послідовності адресних точок (зупинок).
12. Маршрут має статус активності та розраховану дистанцію.

3.2 Вибір сутностей

Відповідно до поставленого завдання та бізнес-правил було виділено наступні сутності:

1. City
2. District
3. Address
4. Shelter
5. Shelter_Entrance
6. Organization
7. Responsible_Person
8. Threat

9. Shelter_Threat
10. Feature
11. Shelter_Feature
12. Inventory_Item
13. Shelter_Inventory
14. Inspector
15. Inspection
16. Route
17. Route_Stop

3.3 Опис сутностей

Після детального аналізу предметного середовища були виділені атрибути сутностей та наведено їх опис (таблиця 3.1).

Таблиця 3.1 – Опис сутностей

Назва сутності	Атрибути	Опис
city	id	Унікальний ідентифікатор міста
	name	Назва міста
	population	Чисельність населення
district	id	Унікальний ідентифікатор району
	name	Назва району
	city_id	Ідентифікатор міста
address	id	Унікальний ідентифікатор адреси
	district_id	Ідентифікатор району
	street	Назва вулиці
	building	Номер будинку
	lat	Географічна широта
	lon	Географічна довгота
organization	id	Унікальний ідентифікатор організації
	name	Назва організації
	type	Тип організації (комунальна, приватна, тощо)
	contacts	Контактні дані організації
responsible_person	id	Унікальний ідентифікатор відповідальної особи
	name	ПІБ відповідального

	phone	Контактний телефон
	organization_id	Ідентифікатор організації, до якої належить особа
	role	Посада або роль особи
shelter	id	Унікальний ідентифікатор укриття
	address_id	Ідентифікатор основної адреси розташування
	capacity	Місткість (кількість осіб)
	status	Статус готовності укриття
	organization_id	Ідентифікатор організації-балансоутримувача
	responsible_id	Ідентифікатор відповідальної особи
	type	Тип захисної споруди
shelter_entrance	id	Унікальний ідентифікатор входу
	shelter_id	Ідентифікатор укриття
	address_id	Ідентифікатор адресної точки входу
	is_main	Позначка головного входу
	note	Примітка (опис входу)
threat	id	Унікальний ідентифікатор загрози
	name	Назва загрози
	desc	Опис загрози
	severity	Рівень небезпеки
shelter_threat	shelter_id	Ідентифікатор укриття
	threat_id	Ідентифікатор загрози
feature	id	Унікальний ідентифікатор особливості
	name	Назва особливості
	desc	Опис особливості
shelter_feature	shelter_id	Ідентифікатор укриття
	feature_id	Ідентифікатор особливості
	value	Значення особливості (якщо наявне)

	notes	Додаткові примітки
inventory_item	id	Унікальний ідентифікатор предмета інвентарю
	name	Назва предмета
	unit	Одиниця вимірювання
	category	Категорія предмету
shelter_inventory	shelter_id	Ідентифікатор укриття
	item_id	Ідентифікатор предмета
	value	Наявна кількість
	notes	Примітки до стану
	expiration_date	Дата закінчення терміну придатності
inspector	id	Унікальний ідентифікатор інспектора
	name	ПІБ інспектора
	phone	Контактний телефон інспектора
inspection	id	Унікальний ідентифікатор перевірки
	shelter_id	Ідентифікатор укриття
	inspector_id	Ідентифікатор інспектора
	date	Дата проведення інспекції
	status	Фінальний статус за результатами перевірки
	notes	Зауваження чи примітки інспектора
	created_at	Час створення запису про інспекцію
route	id	Унікальний ідентифікатор маршруту
	shelter_id	Ідентифікатор цільового укриття
	is_active	Статус активності маршруту
	notes	Примітки про маршрут
	distance	Загально довжина маршруту

route_stop	route_id	Ідентифікатор маршруту
	address_id	Ідентифікатор зупинки
	order	Порядковий номер зупинки в маршруті
	kind	Тип точки

3.4 Опис зв’язків

1. City – District: один до багатьох (1:N) — кожне місто може складатися з багатьох районів, але кожен район належить лише одному місту.
2. District – Address: один до багатьох (1:N) — кожен район включає багато адрес (будівель), кожна адреса територіально належить одному району.
3. Address – Shelter: один до багатьох (1:N) — за однією адресою може бути зареєстровано кілька укриттів (наприклад, у різних корпусах великого комплексу), але кожне укриття має лише одну основну адресу розташування.
4. Address – Shelter_Entrance: один до багатьох (1:N) — одна адреса може слугувати точкою для багатьох входів (рідкісний випадок, але можливий), проте кожен запис про вхід прив'язаний до конкретної адресної точки.
5. Shelter – Shelter_Entrance: один до багатьох (1:N) — кожне укриття може мати кілька входів (основний, запасний), але кожен вхід належить лише одному конкретному укриттю.
6. Organization – Shelter: один до багатьох (1:N) — одна організація може утримувати на балансі багато укриттів, кожне укриття перебуває на балансі лише однієї організації.
7. Organization – Responsible_Person: один до багатьох (1:N) — в одній організації може працювати багато відповідальних осіб, кожна відповідальна особа закріплена за однією організацією.
8. Responsible_Person – Shelter: один до багатьох (1:N) — одна особа може бути відповідальною за кілька укриттів, але за кожним укриттям закріплена одна головна відповідальна особа.
9. Shelter – Threat: багато до багатьох (M:N) — одне укриття може захищати від різних типів загроз, один тип загрози може покриватися багатьма укриттями. Зв'язок реалізовано через проміжну сутність Shelter_Threat.
10. Shelter – Feature: багато до багатьох (M:N) — укриття може мати багато особливостей (Wi-Fi, пандус), одна особливість може бути

наявною у багатьох укриттях. Зв'язок реалізовано через проміжну сутність Shelter_Feature.

11. Shelter – Inventory_Item: багато до багатьох (M:N) — укриття містить багато типів інвентарю, один тип інвентарю може зберігатися у різних укриттях. Зв'язок реалізовано через проміжну сутність Shelter_Inventory, яка також зберігає кількість та терміни придатності.
12. Inspector – Inspection: один до багатьох (1:N) — один інспектор може проводити багато перевірок, кожна перевірка проводиться одним інспектором.
13. Shelter – Inspection: один до багатьох (1:N) — укриття може проходити багато перевірок протягом часу, кожна перевірка стосується одного конкретного укриття.
14. Shelter – Route: один до багатьох (1:N) — до одного укриття може вести багато різних маршрутів евакуації (з різних точок міста), але кожен маршрут веде до одного цільового укриття.
15. Route – Route_Stop: один до багатьох (1:N) — маршрут складається з багатьох зупинок (точок маршруту), кожна зупинка належить одному маршруту.
16. Address – Route_Stop: один до багатьох (1:N) — одна адреса може бути точкою в багатьох різних маршрутах, кожна зупинка маршруту прив'язана до конкретної адреси (географічної точки).

3.5 ER-Діаграма

Побудована ER-модель зображена на рис. 3.1. Кожна таблиця має свої ключові атрибути та відносини з іншими таблицями, створюючи добре структуровану систему для зберігання та управління інформацією. Зовнішні ключі використовуються для забезпечення цілісності даних та зв’язків між сутностями.

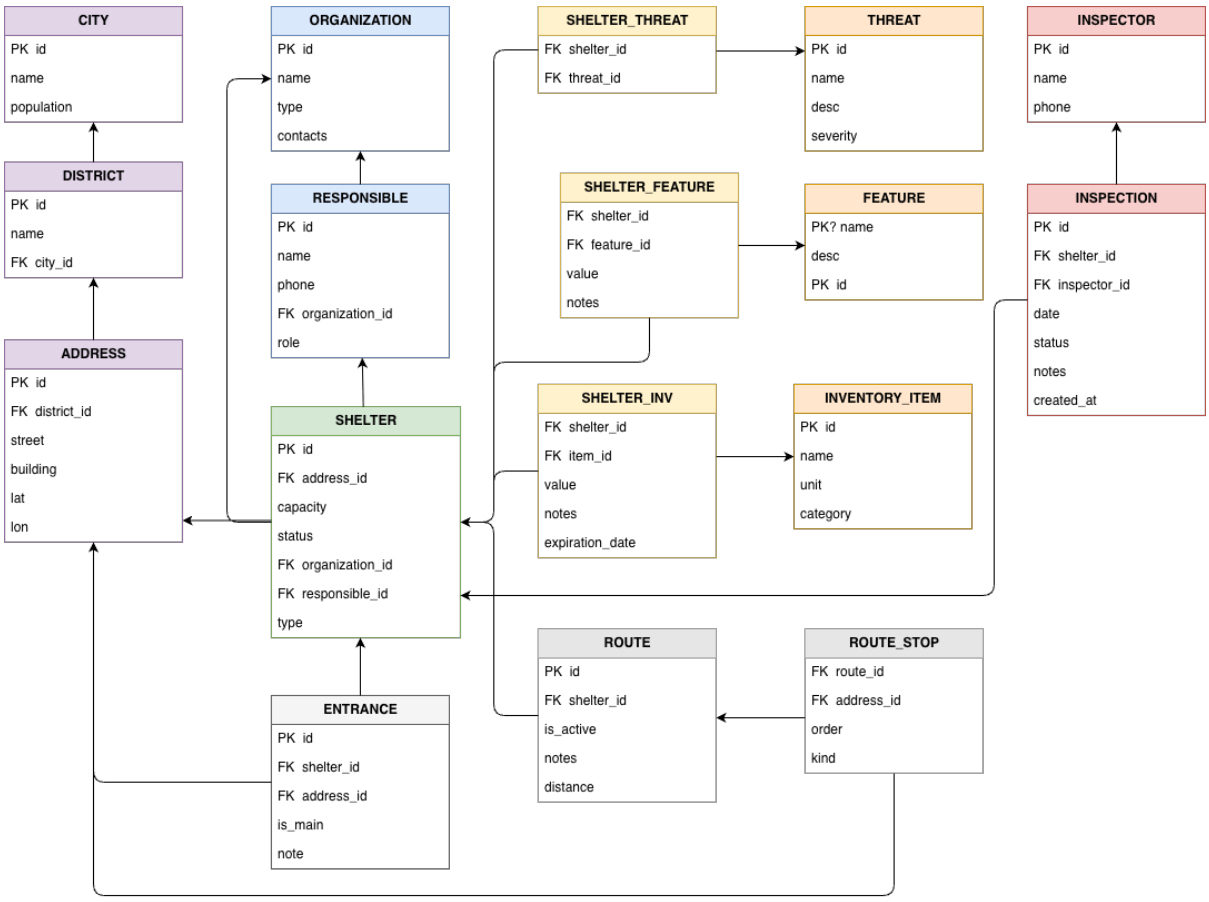


Рис. 3.1 – ER-діаграма бази даних

Отже, в цьому розділі були сформовані бізнес правила та обрані сутності для подальшої побудови бази даних реєстру укриттів та планів евакуації міста. Також була побудована ER-модель бази даних.

4 РЕЛЯЦІЙНА МОДЕЛЬ БАЗИ ДАНИХ

Забезпечення надійності зберігання інформації, швидкодії при обробці геопросторових даних та цілісності зв'язків є критично важливим етапом у проектуванні системи цивільного захисту. У даному розділі буде обґрунтовано вибір інструментарію та наведено структуру бази даних, що забезпечує функціонування реєстру укриттів та планів евакуації.

Для виконання даної курсової роботи було обрано об'єктно-реляційну систему управління базами даних PostgreSQL з ряду ключових причин, які враховують специфіку предметної області.

По-перше, визначальним фактором вибору PostgreSQL є її неперевершені можливості у роботі з географічними даними завдяки розширенню PostGIS. Оскільки система передбачає побудову евакуаційних маршрутів, пошук найближчих укриттів у заданому радіусі та розрахунок дистанцій між адресними точками, використання нативних геометричних типів даних та просторових індексів є необхідністю. Це дозволяє виконувати складні геопросторові запити безпосередньо на рівні бази даних із високою швидкістю, що є критичним для систем безпеки.

По-друге, PostgreSQL відома своєю суворою відповідністю стандартам SQL та надійністю механізмів забезпечення цілісності даних (ACID). У контексті цивільного захисту, де помилка в даних про місткість сховища або його статус може мати фатальні наслідки, сувора типізація та розвинена система обмежень (constraints) гарантують валідність інформації.

По-третє, система підтримує складні аналітичні запити та рекурсивні вирази (CTE), що є необхідним для роботи з ієрархічними структурами даних, такими як маршрути евакуації, які складаються з послідовності зупинок. Це дозволяє ефективно будувати графи маршрутів без необхідності надмірного навантаження на програмну частину застосунку.

Крім того, PostgreSQL є системою з відкритим вихідним кодом (Open Source), що дозволяє використовувати її безкоштовно та розгортати на будь-яких серверних платформах без ліцензійних обмежень. Це забезпечує економічну ефективність рішення для муніципальних органів влади.

Отже, вибір PostgreSQL обумовлений необхідністю поєднання високої надійності зберігання даних із потужним функціоналом для обробки геопросторової інформації, що є ключовим для системи евакуації населення [7, 8].

Після детального аналізу предметного середовища та побудови ER-моделі було розроблено реляційну схему бази даних. Результати проектування таблиць наведені нижче.

Таблиця 4.1 – сутність city

Атрибут	Тип	Ключ
id	serial	PK
name	varchar(1000)	
population	int	

Таблиця 4.2 – сутність district

Атрибут	Тип	Ключ
id	serial	PK
name	varchar(100)	
city_id	int	FK

Таблиця 4.3 – сутність address

Атрибут	Тип	Ключ
id	serial	PK
district_id	int	FK
street	varchar(100)	
building	varchar(20)	
location	geometry	

Таблиця 4.4 – сутність organization

Атрибут	Тип	Ключ
id	serial	PK
name	varchar(200)	
type	varchar(50)	
contacts	text	

Таблиця 4.5 – сутність responsible_person

Атрибут	Тип	Ключ
id	serial	PK
name	varchar(150)	
phone	varchar(20)	
organization_id	int	FK
role	varchar(200)	

Таблиця 4.6 – сутність shelter

Атрибут	Тип	Ключ
id	serial	PK
address_id	int	FK
capacity	int	
status	varchar(50)	
organization_id	int	FK
responsible_id	int	FK
type	varchar(50)	

Таблиця 4.7 – сутність shelter_entrance

Атрибут	Тип	Ключ
id	serial	PK
shelter_id	int	FK
address_id	int	FK
is_main	boolean	
note	text	

Таблиця 4.8 – сутність threat

Атрибут	Тип	Ключ
id	serial	PK
name	varchar(100)	
descriptions	text	
severity	int	

Таблиця 4.9 – сутність shelter_threat

Атрибут	Тип	Ключ
shelter_id	int	PK, FK
threat_id	int	PK, FK

Таблиця 4.10 – сутність feature

Атрибут	Тип	Ключ
id	serial	PK
name	varchar(100)	
description	text	

Таблиця 4.11 – сутність shelter_feature

Атрибут	Тип	Ключ
shelter_id	int	PK, FK

feature_id	int	PK, FK
value	varchar(100)	
notes	text	

Таблиця 4.12 – сутність inventory_item

Атрибут	Тип	Ключ
id	serial	PK
name	varchar(100)	
unit	varchar(20)	
category	varchar(50)	

Таблиця 4.13 – сутність shelter_inventory

Атрибут	Тип	Ключ
shelter_id	int	PK, FK
item_id	int	PK, FK
value	decimal(10,2)	
notes	text	
expiration_date	date	

Таблиця 4.14 – сутність inspector

Атрибут	Тип	Ключ
id	serial	PK
name	varchar(150)	
phone	varchar(20)	

Таблиця 4.15 – сутність inspection

Атрибут	Тип	Ключ
id	serial	PK
shelter_id	int	FK
inspector_id	int	FK
date	timestamp	
status	varchar(50)	
notes	text	
created_at	timestamp	

Таблиця 4.16 – сутність route

Атрибут	Тип	Ключ
id	serial	PK

shelter_id	int	FK
is_active	boolean	
notes	text	
distance	decimal(10,2)	

Таблиця 4.17 – сутність route_stop

Атрибут	Тип	Ключ
route_id	int	PK, FK
address_id	int	FK
stop_order	int	PK
kind	varchar(20)	

5 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

Результатом проектування бази даних є сформований SQL-скрипт, який використовується для створення об'єктів, описаних в ER-моделі та реляційній схемі. Для реалізації було обрано СУБД PostgreSQL.

Перед створенням таблиць необхідно активувати розширення PostGIS для роботи з геопросторовими даними:

```
CREATE EXTENSION IF NOT EXISTS PostGIS;
```

Тепер створимо таблиці:

```
CREATE EXTENSION IF NOT EXISTS postgis;


CREATE TABLE city (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    population INT CHECK (population > 0)
);


CREATE TABLE district (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    city_id INT NOT NULL,
    FOREIGN KEY (city_id) REFERENCES city (id) ON DELETE CASCADE
);


CREATE TABLE address (
    id SERIAL PRIMARY KEY,
    district_id INT NOT NULL,
    street VARCHAR(100) NOT NULL,
    building VARCHAR(20) NOT NULL,
    location GEOMETRY(POINT, 4326),
```

```
    FOREIGN KEY (district_id) REFERENCES district (id) ON DELETE RESTRICT
);

CREATE INDEX idx_address_location ON address USING GIST (location);

CREATE TABLE organization (
    id SERIAL PRIMARY KEY,
    name VARCHAR(200) NOT NULL,
    type VARCHAR(50) NOT NULL,
    contacts TEXT
);

CREATE TABLE responsible_person (
    id SERIAL PRIMARY KEY,
    name VARCHAR(150) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    organization_id INT NOT NULL,
    role VARCHAR(100),
    FOREIGN KEY (organization_id) REFERENCES organization (id) ON DELETE CASCADE
);

CREATE TABLE shelter (
    id SERIAL PRIMARY KEY,
    address_id INT NOT NULL,
    capacity INT NOT NULL CHECK (capacity > 0),
    status VARCHAR(50) NOT NULL CHECK (status IN ('Ready', 'Limited Ready', 'Not Ready')),
    organization_id INT NOT NULL,
```

```
    responsible_id INT NOT NULL,
    type VARCHAR(50) NOT NULL CHECK (type IN ('Bomb Shelter', 'Radiation Shelter', 'Dual Use', 'Simple Shelter')),
    FOREIGN KEY (address_id) REFERENCES address (id),
    FOREIGN KEY (organization_id) REFERENCES organization (id),
    FOREIGN KEY (responsible_id) REFERENCES responsible_person (id)
);

CREATE TABLE shelter_entrance (
    id SERIAL PRIMARY KEY,
    shelter_id INT NOT NULL,
    address_id INT NOT NULL,
    is_main BOOLEAN DEFAULT FALSE,
    note TEXT,
    FOREIGN KEY (shelter_id) REFERENCES shelter (id) ON DELETE CASCADE,
    FOREIGN KEY (address_id) REFERENCES address (id)
);

CREATE TABLE threat (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL UNIQUE,
    description TEXT,
    severity INT CHECK (severity BETWEEN 1 AND 10)
);

CREATE TABLE shelter_threat (
    shelter_id INT NOT NULL,
    threat_id INT NOT NULL,
```

```
    PRIMARY KEY (shelter_id, threat_id),
    FOREIGN KEY (shelter_id) REFERENCES shelter (id) ON DELETE CASCADE,
    FOREIGN KEY (threat_id) REFERENCES threat (id) ON DELETE CASCADE
);

CREATE TABLE feature (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL UNIQUE,
    description TEXT
);

CREATE TABLE shelter_feature (
    shelter_id INT NOT NULL,
    feature_id INT NOT NULL,
    value VARCHAR(100),
    notes TEXT,
    PRIMARY KEY (shelter_id, feature_id),
    FOREIGN KEY (shelter_id) REFERENCES shelter (id) ON DELETE CASCADE,
    FOREIGN KEY (feature_id) REFERENCES feature (id) ON DELETE CASCADE
);

CREATE TABLE inventory_item (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    unit VARCHAR(20) NOT NULL,
    category VARCHAR(50)
);
```

```
CREATE TABLE shelter_inventory (
    shelter_id INT NOT NULL,
    item_id INT NOT NULL,
    value DECIMAL(10,2) NOT NULL CHECK (value >= 0),
    notes TEXT,
    expiration_date DATE,
    PRIMARY KEY (shelter_id, item_id),
    FOREIGN KEY (shelter_id) REFERENCES shelter (id) ON DELETE CASCADE,
    FOREIGN KEY (item_id) REFERENCES inventory_item (id) ON DELETE RESTRICT
);

CREATE TABLE inspector (
    id SERIAL PRIMARY KEY,
    name VARCHAR(150) NOT NULL,
    phone VARCHAR(20) NOT NULL
);

CREATE TABLE inspection (
    id SERIAL PRIMARY KEY,
    shelter_id INT NOT NULL,
    inspector_id INT NOT NULL,
    date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(50) NOT NULL CHECK (status IN ('Passed', 'Failed', 'Needs Improvement')),
    notes TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    FOREIGN KEY (shelter_id) REFERENCES shelter (id) ON DELETE CASCADE,
    FOREIGN KEY (inspector_id) REFERENCES inspector (id)
);

CREATE TABLE route (
    id SERIAL PRIMARY KEY,
    shelter_id INT NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    notes TEXT,
    distance DECIMAL(10,2) CHECK (distance >= 0),
    FOREIGN KEY (shelter_id) REFERENCES shelter (id) ON DELETE CASCADE
);

CREATE TABLE route_stop (
    route_id INT NOT NULL,
    address_id INT NOT NULL,
    stop_order INT NOT NULL CHECK (stop_order >= 1),
    kind VARCHAR(20) NOT NULL CHECK (kind IN ('Start', 'Intermediate', 'Finish')),
    PRIMARY KEY (route_id, stop_order),
    FOREIGN KEY (route_id) REFERENCES route (id) ON DELETE CASCADE,
    FOREIGN KEY (address_id) REFERENCES address (id)
);
```

У результаті було створено базу даних та згенеровано діаграму за допомогою IDE DataGrip. Результат зображено на рисунку 5.1.

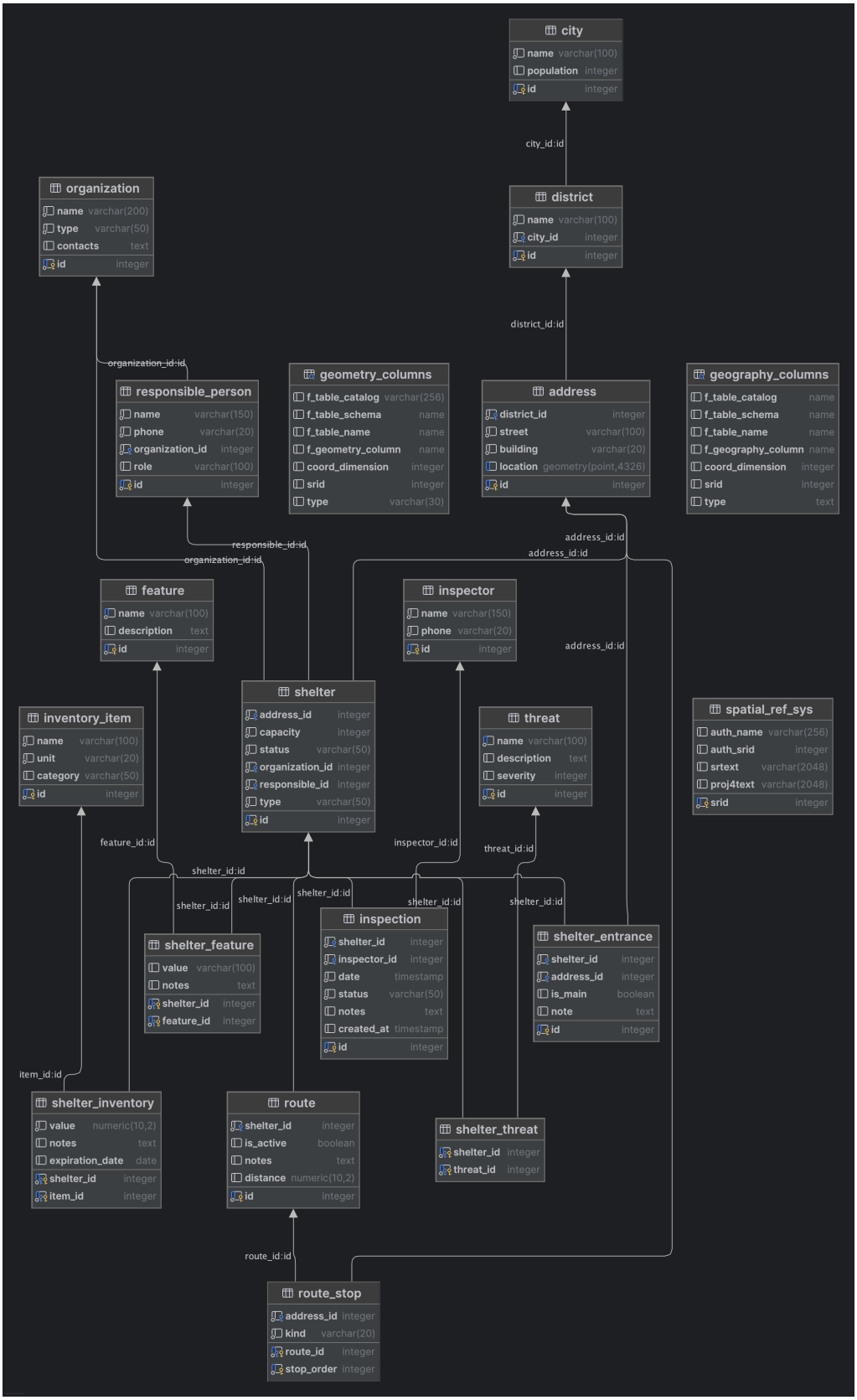


Рис. 5.1 – Діаграма бази даних

6 СТВОРЕННЯ КОРИСТУВАЧІВ

Для забезпечення захисту інформації та розмежування повноважень у системі було виділено чотири основні групи користувачів (ролі):

Адміністратор системи (Administrator). Має повний доступ до всіх об'єктів бази даних. Відповідає за створення структури БД, керування довідниками (міста, райони, типи загроз), реєстрацію нових організацій та надання доступу іншим користувачам.

Інспектор (Inspector). Співробітник контролюючого органу. Має права на перегляд інформації про укриття та їх характеристики, але не може їх змінювати. Має право додавати нові записи в таблицю перевірок (inspection) та змінювати статус інспекції.

Менеджер укриття (Shelter Manager). Представник організації-балансоутримувача. Має право оновлювати інформацію про стан закріплених укриттів, змінювати дані про відповідальних осіб та актуалізувати залишки інвентарю (shelter_inventory). Не має права видаляти укриття або змінювати глобальні довідники.

Гість / Мешканець (Guest). Користувач публічного інтерфейсу. Має права виключно на читання (SELECT) загальнодоступної інформації: адреси укриттів, їх типи, статус готовності та маршрути. Доступ до персональних даних інспекторів або детального опису запасів обмежений.

```
CREATE ROLE admin_role;

CREATE ROLE inspector_role;

CREATE ROLE manager_role;

CREATE ROLE guest_role;


GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admin_role;

GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO admin_role;


GRANT SELECT ON shelter, address, organization, responsible_person, shelter_entrance TO inspector_role;

GRANT INSERT ON inspection TO inspector_role;
```

```
GRANT USAGE, SELECT ON SEQUENCE inspection_id_seq TO inspector_role;


GRANT SELECT ON threat, feature, inventory_item TO manager_role;
GRANT SELECT, UPDATE ON shelter TO manager_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON shelter_inventory TO manager_role;
GRANT SELECT, INSERT, UPDATE ON responsible_person TO manager_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON shelter_feature TO manager_role;
GRANT USAGE, SELECT ON SEQUENCE responsible_person_id_seq TO manager_role;


GRANT SELECT ON city, district, address, shelter, shelter_entrance, route, route_stop TO guest_role;


CREATE USER sys_admin WITH PASSWORD 'secure_pass_admin';
GRANT admin_role TO sys_admin;


CREATE USER insp_petrenko WITH PASSWORD 'insp_pass_2024';
GRANT inspector_role TO insp_petrenko;


CREATE USER manager_osbb WITH PASSWORD 'manager_pass_123';
GRANT manager_role TO manager_osbb;


CREATE USER public_web_client WITH PASSWORD 'guest_access_only';
GRANT guest_role TO public_web_client;
```

7 РОБОТА З БАЗОЮ ДАНИХ

7.1 Функції та процедури

7.1.1 Знаходження найближчого укриття за координатами

```
CREATE OR REPLACE FUNCTION find_nearest_shelter(p_lon DECIMAL,
p_lat DECIMAL)
RETURNS TABLE (
   shelter_id INT,
   full_address TEXT,
   distance_meters INT
) LANGUAGE plpgsql AS $$
DECLARE
   user_point GEOMETRY;
BEGIN
   user_point := ST_SetSRID(ST_MakePoint(p_lon, p_lat), 4326);

   RETURN QUERY
   SELECT
      s.id,
      (d.name || ', ' || a.street || ', ' || a.building)::TEXT,
      ST_DistanceSphere(a.location, user_point)::INT
   FROM shelter s
   JOIN address a ON s.address_id = a.id
   JOIN district d ON a.district_id = d.id
   WHERE s.status = 'Ready'
   ORDER BY a.location <-> user_point
   LIMIT 1;
END;
$$;
```

Результат виконання зображено на рисунку 7.1.

```
coursework=# select * from find_nearest_shelter(30.513, 50.461);
 shelter_id |            full_address            | distance_meters
------------+------------------------------------+----------------
          4 | Подільський, вул. Сагайдачного, 10 |            849
(1 row)
```

Рис. 7.1 – результат виконання функції find_nearest_shelter

7.1.2 Пошук укриттів в радіусі

```
CREATE OR REPLACE FUNCTION get_shelters_in_radius(p_lon DECIMAL,
p_lat DECIMAL, p_radius_meters FLOAT)
RETURNS TABLE (
    shelter_id INT,
    type VARCHAR,
    capacity INT,
    distance_meters INT
) LANGUAGE plpgsql AS $$
DECLARE
    user_point GEOMETRY;
BEGIN
    user_point := ST_SetSRID(ST_MakePoint(p_lon, p_lat), 4326);

    RETURN QUERY
    SELECT
        s.id,
        s.type,
        s.capacity,
        ST_DistanceSphere(a.location, user_point)::INT AS dist
    FROM shelter s
    JOIN address a ON s.address_id = a.id
    WHERE ST_DWithin(a.location::geography, user_point::geography,
p_radius_meters)
      AND s.status != 'Not Ready'
```

```
    ORDER BY dist ASC;
END;
$$;
```

Результат виконання зображено на рисунку 7.2.

```
coursework=# select * from get_shelters_in_radius(30.513, 50.461, 1500);
 shelter_id |     type     | capacity | distance_meters
------------+--------------+----------+-----------------
          4 | Dual Use     |      500 |             849
          1 | Bomb Shelter |      300 |            1418
(2 rows)
```

Рис. 7.2 – результат виконання функції get_shelters_in_radius

7.1.3 Отримання повної адреси укриття

```
CREATE OR REPLACE FUNCTION get_shelter_full_address(p_shelter_id INT)
RETURNS TEXT LANGUAGE plpgsql AS $$
DECLARE
    res TEXT;
BEGIN
    SELECT
        c.name || ', ' || d.name || ', ' || a.street || ', ' || a.building ||
        ' (' || ST_Y(a.location)::NUMERIC(9,6) || ', ' ||
ST_X(a.location)::NUMERIC(9,6) || ')'
    INTO res
    FROM shelter s
    JOIN address a ON s.address_id = a.id
    JOIN district d ON a.district_id = d.id
    JOIN city c ON d.city_id = c.id
    WHERE s.id = p_shelter_id;

    RETURN res;
END;
$$;
```

Результат виконання зображено на рисунку 7.3.

```
coursework=# select * from get_shelter_full_address(4);
                    get_shelter_full_address
-------------------------------------------------------------------
 Київ, Подільський, вул. Сагайдачного, 10 (50.461000, 30.525000)
(1 row)
```

Рис. 7.3 – результат виконання функції get_shelter_full_address

7.1.4 Отримання всіх маршрутів до укриття

```
CREATE OR REPLACE FUNCTION get_routes_to_shelter(p_shelter_id INT)
RETURNS TABLE (
    route_id INT,
    stops_list TEXT,
    total_distance FLOAT
) LANGUAGE plpgsql AS $$
BEGIN
    RETURN QUERY
    SELECT
        r.id,
        STRING_AGG(a.street || ' ' || a.building, ' -> ' ORDER BY rs.stop_order),
        r.distance::FLOAT
    FROM route r
    JOIN route_stop rs ON r.id = rs.route_id
    JOIN address a ON rs.address_id = a.id
    WHERE r.shelter_id = p_shelter_id AND r.is_active = TRUE
    GROUP BY r.id;
END;
$$;
```

Результат виконання зображено на рисунку 7.4.

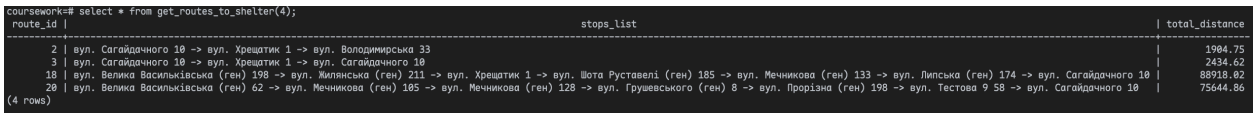


Рис. 7.4 – результат виконання функції get_routes_to_shelter

7.1.5 Процедура для перерахування дистанції маршруту

```
CREATE OR REPLACE PROCEDURE recalculate_route_distance(p_route_id
INT)
LANGUAGE plpgsql AS $$
DECLARE
   v_total_dist DECIMAL(10,2);
BEGIN
   WITH segments AS (
      SELECT
         ST_DistanceSphere(
            a.location,
            LEAD(a.location) OVER (ORDER BY rs.stop_order)
         ) AS segment_dist
      FROM route_stop rs
      JOIN address a ON rs.address_id = a.id
      WHERE rs.route_id = p_route_id
   )
   SELECT SUM(segment_dist) INTO v_total_dist FROM segments;

   UPDATE route
   SET distance = COALESCE(v_total_dist, 0)
   WHERE id = p_route_id;
END;
$$;
```

Результат виконання зображено на рисунку 7.5.

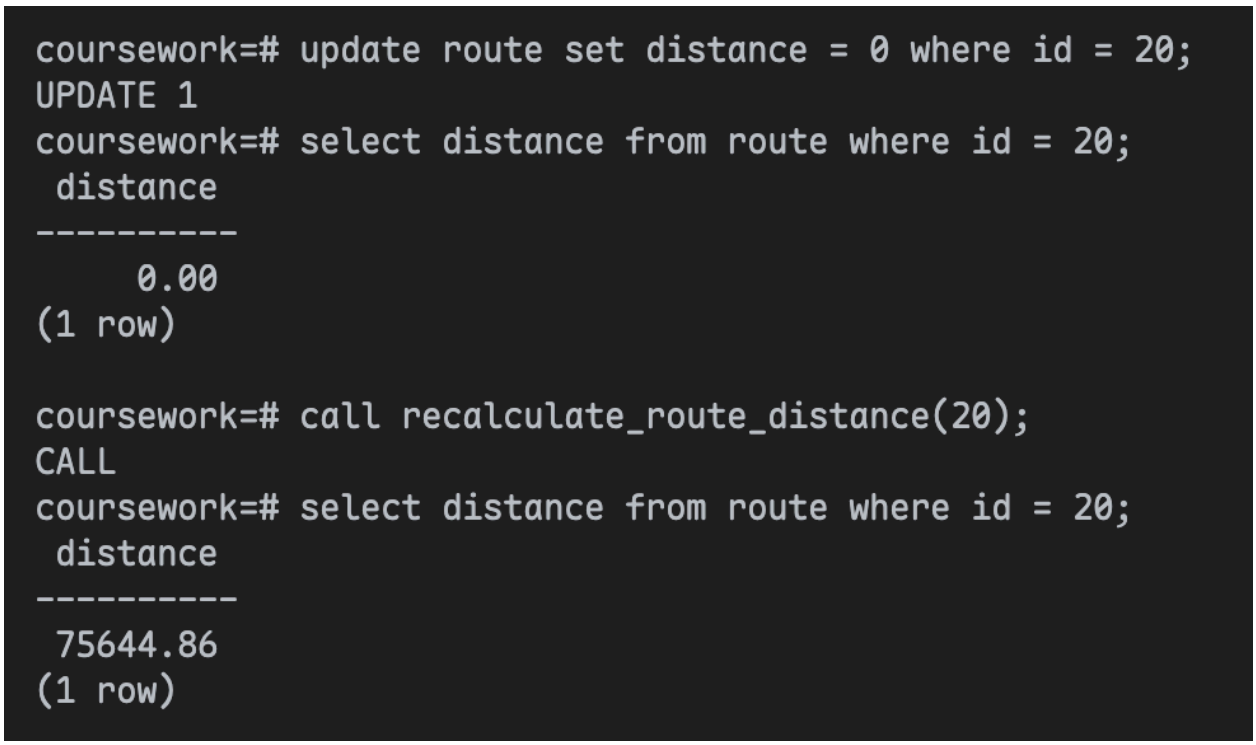


Рис. 7.5 – результат виконання процедури recalculate_route_distance

7.1.6 Функція для валідації маршруту

```
CREATE OR REPLACE FUNCTION validate_route(p_route_id INT)
RETURNS TABLE (
    is_valid BOOLEAN,
    reason TEXT
) LANGUAGE plpgsql AS $$
DECLARE
    v_has_start BOOLEAN;
    v_finish_count INT;
    v_last_kind VARCHAR;
BEGIN
    SELECT EXISTS(SELECT 1 FROM route_stop WHERE route_id = p_route_id
AND kind = 'Start')
    INTO v_has_start;

    IF NOT v_has_start THEN
        is_valid := FALSE;
```

```
    reason := 'Відсутня точка Start';
    RETURN NEXT;
    RETURN;
END IF;

SELECT COUNT(*) INTO v_finish_count
FROM route_stop
WHERE route_id = p_route_id AND kind = 'Finish';

IF v_finish_count = 0 THEN
    is_valid := FALSE;
    reason := 'Відсутня точка Finish';
    RETURN NEXT;
    RETURN;
END IF;

IF v_finish_count > 1 THEN
    is_valid := FALSE;
    reason := 'Більше однієї точки Finish';
    RETURN NEXT;
    RETURN;
END IF;

SELECT kind INTO v_last_kind
FROM route_stop
WHERE route_id = p_route_id
ORDER BY stop_order DESC
LIMIT 1;
```

```
    IF v_last_kind != 'Finish' THEN
        is_valid := FALSE;
        reason := 'Остання точка не є Finish';
        RETURN NEXT;
        RETURN;
    END IF;

    is_valid := TRUE;
    reason := 'Маршрут валідний';
    RETURN NEXT;
END;
$$;
```

Результат виконання зображено на рисунку 7.6.

```
coursework=# select * from validate_route(20);
 is_valid |      reason
----------+------------------
 t        | Маршрут валідний
(1 row)
```

Рис. 7.6 – результат виконання функції validate_route

7.1.7 Процедура для створення маршруту з масиву адрес

```
CREATE OR REPLACE PROCEDURE create_route_from_addresses(
    p_address_ids INT[],
    p_notes TEXT
) LANGUAGE plpgsql AS $$
DECLARE
    v_new_route_id INT;
    v_target_shelter_id INT;
    v_addr_id INT;
    v_order INT := 1;
    v_kind VARCHAR;
```

```
    v_len INT;
BEGIN
    v_len := array_length(p_address_ids, 1);

    IF v_len < 2 THEN
        RAISE EXCEPTION 'Маршрут повинен мати мінімум 2 точки';
    END IF;

    SELECT id INTO v_target_shelter_id
    FROM shelter
    WHERE address_id = p_address_ids[v_len]
    LIMIT 1;

    IF v_target_shelter_id IS NULL THEN
        RAISE EXCEPTION 'За останньою адресою (ID %) не знайдено зареєстрованого укриття', p_address_ids[v_len];
    END IF;

    INSERT INTO route (shelter_id, is_active, notes, distance)
    VALUES (v_target_shelter_id, TRUE, p_notes, 0)
    RETURNING id INTO v_new_route_id;

    FOREACH v_addr_id IN ARRAY p_address_ids
    LOOP
        IF v_order = 1 THEN v_kind := 'Start';
        ELSIF v_order = v_len THEN v_kind := 'Finish';
        ELSE v_kind := 'Intermediate';
        END IF;
```

```
        INSERT INTO route_stop (route_id, address_id, stop_order, kind)
        VALUES (v_new_route_id, v_addr_id, v_order, v_kind);

        v_order := v_order + 1;
    END LOOP;

    CALL recalculate_route_distance(v_new_route_id);

    RAISE NOTICE 'Маршрут % створено успішно до укриття %',
v_new_route_id, v_target_shelter_id;
END;
$$;
```

Результат виконання зображено на рисунку 7.7.

```
coursework=# call create_route_from_addresses('{1}', '');
ERROR:  Маршрут повинен мати мінімум 2 точки
CONTEXT:  PL/pgSQL function create_route_from_addresses(integer[],text) line 13 at RAISE
coursework=# call create_route_from_addresses('{1,2,3}', '');
NOTICE:  Маршрут 84 створено успішно до укриття 3
CALL
coursework=# select shelter_id from route where id = 84;
 shelter_id
------------
          3
(1 row)
```

Рис. 7.7 – результат виконання процедури create_route_from_addresses

7.1.8 Отримання загальної місткості всіх укриттів в місті

```
CREATE OR REPLACE FUNCTION get_total_city_capacity(p_city_id INT)
RETURNS BIGINT LANGUAGE plpgsql AS $$
DECLARE
    v_sum BIGINT;
BEGIN
    SELECT SUM(s.capacity) INTO v_sum
    FROM shelter s
    JOIN address a ON s.address_id = a.id
    JOIN district d ON a.district_id = d.id
```

```
    WHERE d.city_id = p_city_id
      AND s.status IN ('Ready', 'Limited Ready');

    RETURN COALESCE(v_sum, 0);
END;
$$;
```

Результат зображено на рисунку 7.8.

```
coursework=# select * from get_total_city_capacity(1);
 get_total_city_capacity
-------------------------
                    3354
(1 row)
```

Рис. 7.8 – результат виконання функції get_total_city_capacity

7.1.9 Отримання основної інформації про укриття одним рядком

```
CREATE OR REPLACE FUNCTION get_shelter_summary(p_shelter_id INT)
RETURNS TEXT LANGUAGE plpgsql AS $$
DECLARE
    v_res TEXT;
    v_last_insp TEXT;
    v_entrances INT;
BEGIN
    SELECT to_char(date, 'YYYY-MM-DD') || ' (' || status || ')'
    INTO v_last_insp
    FROM inspection
    WHERE shelter_id = p_shelter_id
    ORDER BY date DESC LIMIT 1;

    SELECT COUNT(*) INTO v_entrances FROM shelter_entrance WHERE
shelter_id = p_shelter_id;
```

```
    SELECT format(
        '[%s] %s, %s. Орг: %s. Відп: %s. Тип: %s. Статус: %s. Інспекція: %s. Входів: %s',
        c.name, d.name, a.street || ' ' || a.building,
        o.name, rp.name,
        s.type, s.status,
        COALESCE(v_last_insp, 'Не проводилась'),
        v_entrances
    ) INTO v_res
    FROM shelter s
    JOIN address a ON s.address_id = a.id
    JOIN district d ON a.district_id = d.id
    JOIN city c ON d.city_id = c.id
    JOIN organization o ON s.organization_id = o.id
    JOIN responsible_person rp ON s.responsible_id = rp.id
    WHERE s.id = p_shelter_id;

    RETURN v_res;
END;
$$;
```

Результат зображено на рисунку 7.9.

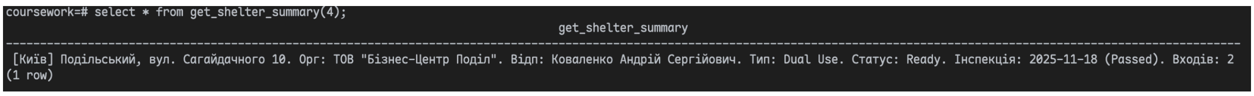


Рис. 7.9 – результат виконання функції get_shelter_summary

7.1.10 Перевірка прострочених продуктів в укритті

```
CREATE OR REPLACE FUNCTION check_expired_inventory(p_shelter_id INT)
RETURNS TABLE (
    item_name VARCHAR,
    quantity DECIMAL,
```

```
    expiration_date DATE,
    days_overdue INT
) LANGUAGE plpgsql AS $$
BEGIN
    RETURN QUERY
    SELECT
        ii.name,
        si.value,
        si.expiration_date,
        (CURRENT_DATE - si.expiration_date)::INT
    FROM shelter_inventory si
    JOIN inventory_item ii ON si.item_id = ii.id
    WHERE si.shelter_id = p_shelter_id
      AND si.expiration_date < CURRENT_DATE;
END;
$$;
```

Результат виконання зображено на рисунку 7.10.

```
coursework=# select * from check_expired_inventory(4);
  item_name  | quantity | expiration_date | days_overdue
-------------+----------+-----------------+--------------
 Бензин А-95 |   100.00 | 2025-12-01      |           13
(1 row)
```

Рис. 7.10 – результат виконання функції check_expired_inventory

7.2 Триггери

7.2.1 Автоматичне оновлення статусу після інспекції

```
CREATE OR REPLACE FUNCTION update_shelter_status_on_inspection()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.status = 'Passed' THEN
        UPDATE shelter SET status = 'Ready' WHERE id = NEW.shelter_id;
```

```
    ELSIF NEW.status = 'Failed' THEN
        UPDATE shelter SET status = 'Not Ready' WHERE id = NEW.shelter_id;
    ELSIF NEW.status = 'Needs Improvement' THEN
        UPDATE shelter SET status = 'Limited Ready' WHERE id =
NEW.shelter_id;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER trg_update_shelter_status
AFTER INSERT ON inspection
FOR EACH ROW EXECUTE FUNCTION
update_shelter_status_on_inspection();
```

Результат зображено на рисунку 7.11.

```
coursework=# SELECT id, status FROM shelter WHERE id = 1;
 id | status
----+--------
  1 | Ready
(1 row)

coursework=# INSERT INTO inspection (shelter_id, inspector_id, status, notes)  VALUES (1, 1, 'Failed', 'Виявлено серйозні порушення');
NOTICE:  Укриття 1 не готове. Маршрути деактивовано.
INSERT 0 1
coursework=# SELECT id, status FROM shelter WHERE id = 1;
 id |  status
----+-----------
  1 | Not Ready
(1 row)
```

Рис. 7.11 – результат роботи триггера trg_update_shelter_status

7.2.2 Блокування оновлень укриття, якщо його статус «готове»

```
CREATE OR REPLACE FUNCTION lock_ready_shelter_updates()
RETURNS TRIGGER AS $$
BEGIN
    IF OLD.status = 'Ready' AND NEW.status = 'Ready' THEN
        IF NEW.capacity != OLD.capacity OR NEW.type != OLD.type OR
NEW.address_id != OLD.address_id THEN
            RAISE EXCEPTION 'Редагування характеристик заборонено для
укриттів зі статусом Ready. Спочатку змініть статус.';
```

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

CREATE TRIGGER trg_lock_ready_shelter
BEFORE UPDATE ON shelter
FOR EACH ROW EXECUTE FUNCTION lock_ready_shelter_updates();
```

Результат роботи зображено на рисунку 7.12.

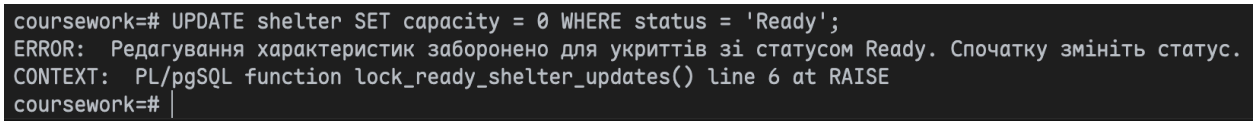


Рис. 7.12 – результат роботи триггера trg_lock_ready_shelter

7.2.3 Валідація послідовності зупинок маршруту

```
CREATE OR REPLACE FUNCTION validate_route_stops_logic()
RETURNS TRIGGER AS $$
DECLARE
    v_finish_exists BOOLEAN;
BEGIN
    IF NEW.kind = 'Start' AND EXISTS(SELECT 1 FROM route_stop WHERE
route_id = NEW.route_id AND kind = 'Start') THEN
        RAISE EXCEPTION 'Маршрут вже має точку початку (Start)';
    END IF;

    IF NEW.kind = 'Finish' AND EXISTS(SELECT 1 FROM route_stop WHERE
route_id = NEW.route_id AND kind = 'Finish') THEN
        RAISE EXCEPTION 'Маршрут вже має точку кінця (Finish)';
    END IF;
```

```
    SELECT EXISTS(SELECT 1 FROM route_stop WHERE route_id =
NEW.route_id AND kind = 'Finish')
    INTO v_finish_exists;

    IF v_finish_exists AND NEW.kind != 'Finish' THEN
        IF NEW.stop_order > (SELECT stop_order FROM route_stop WHERE
route_id = NEW.route_id AND kind = 'Finish') THEN
            RAISE EXCEPTION 'Не можна додавати зупинки після точки Finish';
        END IF;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_validate_stops
BEFORE INSERT ON route_stop
FOR EACH ROW EXECUTE FUNCTION validate_route_stops_logic();
```

7.2.4 Гарантія лише одного основного входу в укриття

```
CREATE OR REPLACE FUNCTION ensure_single_main_entrance()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.is_main = TRUE THEN
        UPDATE shelter_entrance
        SET is_main = FALSE
        WHERE shelter_id = NEW.shelter_id AND id != NEW.id AND is_main =
TRUE;
    END IF;
    RETURN NEW;
```

```
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER trg_auto_demote_entrances
AFTER INSERT OR UPDATE ON shelter_entrance
FOR EACH ROW EXECUTE FUNCTION ensure_single_main_entrance();
```

Результат роботи зображено на рисунку 7.13.

```
coursework=# INSERT INTO shelter_entrance (shelter_id, address_id, is_main) VALUES (2, 2, TRUE);
INSERT 0 1
coursework=# INSERT INTO shelter_entrance (shelter_id, address_id, is_main) VALUES (2, 3, TRUE);
INSERT 0 1
coursework=# SELECT * FROM shelter_entrance WHERE shelter_id = 2;
 id | shelter_id | address_id | is_main |           note
----+------------+------------+---------+-------------------------
  5 |          2 |         45 | f       | Додатковий вхід #2 (ген)
  4 |          2 |         11 | f       | Головний вхід (ген)
 53 |          2 |          3 | t       |
 52 |          2 |          2 | f       |
(4 rows)
```

Рис. 7.13 – результат роботи триггера trg_auto_demote_entrances

7.2.5 Автоматична зміна статусу маршруту при зміні статусу укриття

```
CREATE OR REPLACE FUNCTION sync_routes_with_shelter_status()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW.status = 'Not Ready' THEN
      UPDATE route SET is_active = FALSE WHERE shelter_id = NEW.id;
      RAISE NOTICE 'Укриття % не готове. Маршрути деактивовано.',
NEW.id;
   ELSIF NEW.status = 'Ready' AND OLD.status != 'Ready' THEN
      UPDATE route SET is_active = TRUE WHERE shelter_id = NEW.id;
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trg_sync_route_active
AFTER UPDATE OF status ON shelter
FOR EACH ROW EXECUTE FUNCTION sync_routes_with_shelter_status();
```

Результат роботи зображено на рисунку 7.14.

```
coursework=# SELECT id, shelter_id, is_active FROM route WHERE shelter_id = 1;
 id | shelter_id | is_active
----+------------+-----------
  1 |          1 | f
  5 |          1 | f
  6 |          1 | f
  7 |          1 | f
  8 |          1 | f
(5 rows)

coursework=# UPDATE shelter SET status = 'Ready' WHERE id = 1;
UPDATE 1
coursework=# SELECT id, shelter_id, is_active FROM route WHERE shelter_id = 1;
 id | shelter_id | is_active
----+------------+-----------
  1 |          1 | t
  5 |          1 | t
  6 |          1 | t
  7 |          1 | t
  8 |          1 | t
(5 rows)
```

Рис. 7.14 – результат роботи триггера trg_sync_route_active

7.3 Представлення

7.3.1 Представлення публічних укриттів

Це представлення призначене для використання у публічному веб-інтерфейсі або мобільному додатку для громадян. Воно об'єднує дані про місцезнаходження та характеристики укриття, фільтруючи об'єкти, які мають статус “Не готово”, щоб не дезорієнтувати населення під час тривоги. Також воно приховує чутливу інформацію про відповідальних осіб та інвентар.

```
CREATE OR REPLACE VIEW v_public_shelters AS
SELECT
    s.id AS shelter_id,
    d.name AS district,
    a.street || ' ' || a.building AS address,
    s.type AS shelter_type,
```

```
    s.capacity,
    s.status,
    ST_Y(a.location)::NUMERIC(10,6) AS lat,
    ST_X(a.location)::NUMERIC(10,6) AS lon
FROM shelter s
JOIN address a ON s.address_id = a.id
JOIN district d ON a.district_id = d.id
WHERE s.status IN ('Ready', 'Limited Ready');
```

На рисунку 7.15 зображено дані з представлення.

```
coursework=# select * from v_public_shelters;
 shelter_id |    district    |               address                |  shelter_type  | capacity |    status     |    lat    |    lon
------------+----------------+--------------------------------------+----------------+----------+---------------+-----------+-----------
          1 | Шевченківський | вул. Хрещатик 1                      | Bomb Shelter   |      300 | Ready         | 50.450100 | 30.523400
          2 | Шевченківський | вул. Володимирська 33                | Simple Shelter |      150 | Limited Ready | 50.447000 | 30.515000
          3 | Печерський     | бул. Лесі Українки 26                | Simple Shelter |       50 | Ready         | 50.426500 | 30.538000
          4 | Подільський    | вул. Сагайдачного 10                 | Dual Use       |      500 | Ready         | 50.461000 | 30.525000
          6 | Дарницький     | вул. Харківське шосе 121             | Bomb Shelter   |      400 | Ready         | 50.415000 | 30.665000
          7 | Подільський    | вул. Тестова 1 28                    | Simple Shelter |      423 | Ready         | 50.479130 | 30.622611
          8 | Печерський     | вул. Тестова 2 71                    | Simple Shelter |      263 | Ready         | 50.438859 | 30.679605
         10 | Дарницький     | вул. Тестова 4 77                    | Simple Shelter |      198 | Ready         | 50.463190 | 30.578900
         11 | Шевченківський | вул. Тестова 5 67                    | Simple Shelter |      146 | Ready         | 50.423020 | 30.650744
         19 | Дарницький     | вул. Тестова 13 7                    | Bomb Shelter   |      101 | Ready         | 50.468947 | 30.540517
         16 | Подільський    | вул. Прорізна (ген) 60               | Bomb Shelter   |       84 | Ready         | 50.576626 | 30.591508
         17 | Голосіївський  | вул. Велика Васильківська (ген) 219  | Bomb Shelter   |       90 | Ready         | 50.453577 | 30.585189
         18 | Печерський     | просп. Перемоги (ген) 72             | Bomb Shelter   |      100 | Ready         | 50.424972 | 30.798244
         21 | Подільський    | вул. Мечникова (ген) 128             | Bomb Shelter   |      121 | Ready         | 50.568471 | 30.699371
         20 | Шевченківський | вул. Богдана Хмельницького (ген) 122 | Bomb Shelter   |      114 | Ready         | 50.522911 | 30.547572
         13 | Печерський     | вул. Грушевського (ген) 8            | Bomb Shelter   |       79 | Ready         | 50.486035 | 30.547338
         14 | Шевченківський | вул. Саксаганського (ген) 163        | Bomb Shelter   |       83 | Ready         | 50.541983 | 30.724047
         12 | Шевченківський | вул. Велика Васильківська (ген) 75   | Bomb Shelter   |       69 | Ready         | 50.532840 | 30.673641
         15 | Голосіївський  | вул. Велика Васильківська (ген) 167  | Bomb Shelter   |       83 | Ready         | 50.481799 | 30.539637
(19 rows)
```

Рис. 7.15 – данні з представлення v_public_shelters

7.3.2 Представлення критичного стану інвентарів

Це представлення створене для менеджерів та служб забезпечення. Воно автоматично відфільтровує та показує лише ті позиції інвентарю, термін придатності яких вже сплив або спливає протягом наступних 30 днів. Це дозволяє проактивно планувати закупівлі та заміни запасів без необхідності переглядати весь список майна.

```
CREATE OR REPLACE VIEW v_critical_inventory AS
SELECT
    s.id AS shelter_id,
    a.street || ' ' || a.building AS address,
    rp.name AS responsible_person,
    rp.phone AS contact_phone,
    ii.name AS item_name,
```

```
    si.value AS quantity,
    ii.unit,
    si.expiration_date,
    CASE
        WHEN si.expiration_date < CURRENT_DATE THEN 'Expired'
        ELSE 'Expiring Soon'
    END AS status
FROM shelter_inventory si
JOIN inventory_item ii ON si.item_id = ii.id
JOIN shelter s ON si.shelter_id = s.id
JOIN address a ON s.address_id = a.id
JOIN responsible_person rp ON s.responsible_id = rp.id
WHERE si.expiration_date <= (CURRENT_DATE + INTERVAL '30 days');
```

Дані з представлення зображено на рисунку 7.16.

```
coursework=# select * from v_critical_inventory;
 shelter_id |       address        |     responsible_person      | contact_phone |      item_name      | quantity | unit | expiration_date |    status
------------+----------------------+-----------------------------+---------------+---------------------+----------+------+-----------------+---------------
          1 | вул. Хрещатик 1      | Петренко Іван Іванович      | +380501112233 | Вода питна (пляшки) |   500.00 | л    | 2026-01-01      | Expiring Soon
          1 | вул. Хрещатик 1      | Петренко Іван Іванович      | +380501112233 | Бензин А-95         |   200.00 | л    | 2025-06-01      | Expired
          4 | вул. Сагайдачного 10 | Коваленко Андрій Сергійович | +380635554433 | Бензин А-95         |   100.00 | л    | 2025-12-01      | Expired
(3 rows)
```

Рис. 7.16 – дані з представлення v_critical_inventory

7.3.3 Представлення статистики по районах

Аналітичне представлення для вищого керівництва міста та відділу планування цивільного захисту. Воно агрегує дані по районах, показуючи загальну кількість укриттів, сумарну місткість та відсоток готових до використання об'єктів. Це допомагає виявляти райони з дефіцитом захисних споруд.

```
CREATE OR REPLACE VIEW v_district_statistics AS
SELECT
    d.name AS district_name,
    COUNT(s.id) AS total_shelters,
    SUM(CASE WHEN s.status = 'Ready' THEN s.capacity ELSE 0 END) AS effective_capacity,
```

```
    ROUND(
        (COUNT(CASE WHEN s.status = 'Ready' THEN 1 END)::DECIMAL /
        NULLIF(COUNT(s.id), 0) * 100), 2
    ) AS readiness_percentage
FROM district d
LEFT JOIN address a ON d.id = a.district_id
LEFT JOIN shelter s ON a.id = s.address_id
GROUP BY d.id, d.name
ORDER BY effective_capacity ASC;
```

Дані з представлення зображено на рисунку 7.17.

```
coursework=# select * from v_district_statistics;
 district_name  | total_shelters | effective_capacity | readiness_percentage
----------------+----------------+--------------------+----------------------
 Голосіївський  |              3 |                173 |                66.67
 Печерський     |              4 |                492 |               100.00
 Дарницький     |              3 |                699 |               100.00
 Шевченківський |              6 |                712 |                83.33
 Подільський    |              5 |               1128 |                80.00
(5 rows)
```

Рис. 7.17 – дані з представлення v_district_statistics

7.4 Select-запити

7.4.1 Отримати повний список укриттів з адресами, районами та назвами міст.

```
SELECT
    s.id AS shelter_id,
    c.name AS city,
    d.name AS district,
    a.street,
    a.building,
    s.type,
    s.status
```

```
FROM shelter s
JOIN address a ON s.address_id = a.id
JOIN district d ON a.district_id = d.id
JOIN city c ON d.city_id = c.id
ORDER BY c.name, d.name;
```

shelter_id	city	district	street	building	type	status
5	Київ	Голосіївський	просп. Голосіївський	42	Radiation Shelter	Not Ready
15	Київ	Голосіївський	вул. Велика Васильківська (ген)	167	Bomb Shelter	Ready
17	Київ	Голосіївський	вул. Велика Васильківська (ген)	219	Bomb Shelter	Ready
6	Київ	Дарницький	вул. Харківське шосе	121	Bomb Shelter	Ready
10	Київ	Дарницький	вул. Тестова 4	77	Simple Shelter	Ready
19	Київ	Дарницький	вул. Тестова 13	7	Bomb Shelter	Ready
3	Київ	Печерський	бул. Лесі Українки	26	Simple Shelter	Ready
13	Київ	Печерський	вул. Грушевського (ген)	8	Bomb Shelter	Ready
8	Київ	Печерський	вул. Тестова 2	71	Simple Shelter	Ready

Рис. 7.18 – результат виконання команди

7.4.2 Вивести контактні дані відповідальних осіб для кожного укриття разом з назвою організації-балансоутримувача.

```
SELECT
    s.id AS shelter_id,
    a.street || ' ' || a.building AS address,
    o.name AS organization,
    rp.name AS responsible_name,
    rp.phone AS responsible_phone,
    rp.role
FROM shelter s
JOIN address a ON s.address_id = a.id
JOIN organization o ON s.organization_id = o.id
JOIN responsible_person rp ON s.responsible_id = rp.id;
```

shelter_id	address	organization	responsible_name	responsible_phone	role
2	вул. Володимирська 33	КП "Київжитлоспецексплуатація"	Петренко Іван Іванович	+380501112233	Головний інженер
3	бул. Лесі Українки 26	ОСББ "Затишок"	Сидоренко Олена Петрівна	+380679998877	Голова правління
4	вул. Сагайдачного 10	ТОВ "Бізнес-Центр Поділ"	Коваленко Андрій Сергійович	+380635554433	Завгосп
5	просп. Голосіївський 42	КП "Київжитлоспецексплуатація"	Петренко Іван Іванович	+380501112233	Головний інженер
6	вул. Харківське шосе 121	ЗОШ №55	Мельник Тетяна Василівна	+380971231231	Директор школи
7	вул. Тестова 1 28	ОСББ "Затишок"	Сидоренко Олена Петрівна	+380679998877	Голова правління
8	вул. Тестова 2 71	ОСББ "Затишок"	Сидоренко Олена Петрівна	+380679998877	Голова правління
9	вул. Тестова 3 34	КП "Київжитлоспецексплуатація"	Сидоренко Олена Петрівна	+380679998877	Голова правління
10	вул. Тестова 4 77	ТОВ "Бізнес-Центр Поділ"	Сидоренко Олена Петрівна	+380679998877	Голова правління

Рис. 7.19 – результат виконання команди

7.4.3 Підрахувати сумарну місткість укриттів по кожному району міста.

```
SELECT
    d.name AS district_name,
    COUNT(s.id) AS total_shelters,
    SUM(s.capacity) AS total_capacity
FROM district d
JOIN address a ON d.id = a.district_id
JOIN shelter s ON a.id = s.address_id
WHERE s.status IN ('Ready', 'Limited Ready')
GROUP BY d.name
ORDER BY total_capacity DESC;
```

```
 district_name  | total_shelters | total_capacity
----------------+----------------+----------------
 Подільський    |              4 |           1128
 Шевченківський |              6 |            862
 Дарницький     |              3 |            699
 Печерський     |              4 |            492
 Голосіївський  |              2 |            173
(5 rows)
```

Рис. 7.20 – результат виконання команди

7.4.4 Знайти всі укриття, які захищають від "Радіаційного забруднення" та є готовими до використання.

```
SELECT
    s.id,
    a.street,
    s.capacity,
    t.name AS threat_name
FROM shelter s
JOIN address a ON s.address_id = a.id
JOIN shelter_threat st ON s.id = st.shelter_id
JOIN threat t ON st.threat_id = t.id
```

```
WHERE t.name = 'Радіаційне забруднення'
   AND s.status = 'Ready';
```

```
 id |        street        | capacity |      threat_name
----+----------------------+----------+-----------------------
  1 | вул. Хрещатик        |      300 | Радіаційне забруднення
  6 | вул. Харківське шосе |      400 | Радіаційне забруднення
(2 rows)
```

Рис. 7.21 – результат виконання команди

7.4.5 Вивести список укриттів, які мають особливість "Генератор" та знаходяться в певному районі (наприклад, "Шевченківський").

```
SELECT
   s.id,
   a.street,
   sf.value AS generator_power
FROM shelter s
JOIN shelter_feature sf ON s.id = sf.shelter_id
JOIN feature f ON sf.feature_id = f.id
JOIN address a ON s.address_id = a.id
JOIN district d ON a.district_id = d.id
WHERE f.name = 'Генератор'
   AND d.name = 'Шевченківський';
```

```
 id |    street     | generator_power
----+---------------+-----------------
  1 | вул. Хрещатик | 50 кВт
(1 row)
```

Рис. 7.22 – результат виконання команди

7.4.6 Показати наявність питної води у всіх укриттях, де її менше 1000 одиниць (літрів).

```
SELECT
   s.id,
   a.street,
   ii.name AS item,
```

```
    si.value AS quantity,
    ii.unit
FROM shelter s
JOIN address a ON s.address_id = a.id
JOIN shelter_inventory si ON s.id = si.shelter_id
JOIN inventory_item ii ON si.item_id = ii.id
WHERE ii.name LIKE '%Вода%'
    AND si.value < 1000;
```

```
 id |    street     |        item          | quantity | unit
----+---------------+----------------------+----------+------
  1 | вул. Хрещатик | Вода питна (пляшки) |   500.00 | л
(1 row)
```

Рис. 7.23 – результат виконання команди

7.4.7 Звіт про прострочені продукти: адреса укриття, відповідальний та назва простроченого товару.

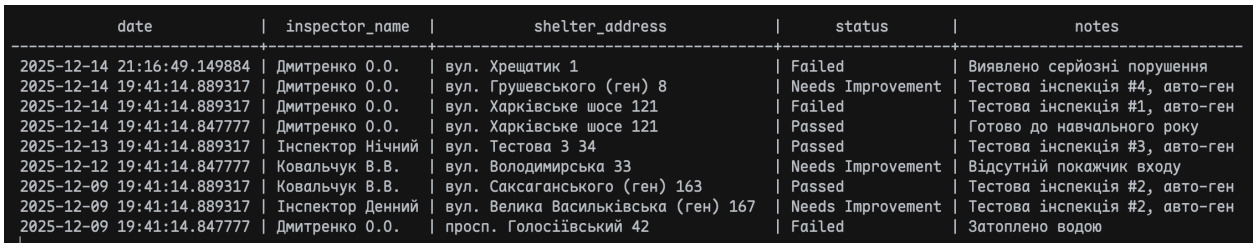
```
SELECT
    s.id,
    rp.name AS responsible,
    rp.phone,
    ii.name AS item,
    si.expiration_date
FROM shelter s
JOIN responsible_person rp ON s.responsible_id = rp.id
JOIN shelter_inventory si ON s.id = si.shelter_id
JOIN inventory_item ii ON si.item_id = ii.id
WHERE si.expiration_date < CURRENT_DATE;
```

```
 id |        responsible          |     phone     |    item     | expiration_date
----+-----------------------------+---------------+-------------+-----------------
  1 | Петренко Іван Іванович      | +380501112233 | Бензин А-95 | 2025-06-01
  4 | Коваленко Андрій Сергійович | +380635554433 | Бензин А-95 | 2025-12-01
(2 rows)
```

Рис. 7.24 – результат виконання команди

7.4.8 Вивести історію перевірок: хто перевіряв, коли, яку адресу і який результат.

```
SELECT
    i.date,
    insp.name AS inspector_name,
    a.street || ' ' || a.building AS shelter_address,
    i.status,
    i.notes
FROM inspection i
JOIN inspector insp ON i.inspector_id = insp.id
JOIN shelter s ON i.shelter_id = s.id
JOIN address a ON s.address_id = a.id
ORDER BY i.date DESC;
```

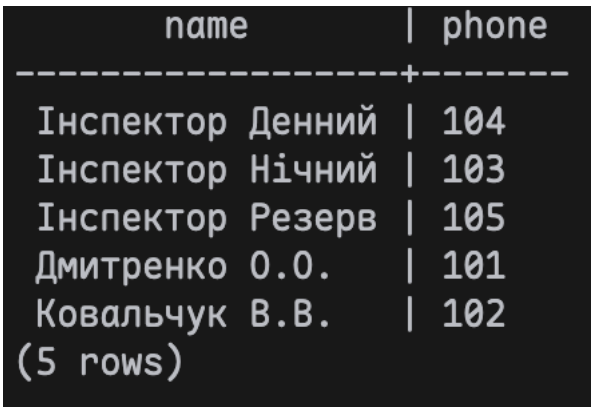


date	inspector_name	shelter_address	status	notes
2025-12-14 21:16:49.149884	Дмитренко О.О.	вул. Хрещатик 1	Failed	Виявлено серйозні порушення
2025-12-14 19:41:14.889317	Дмитренко О.О.	вул. Грушевського (ген) 8	Needs Improvement	Тестова інспекція #4, авто-ген
2025-12-14 19:41:14.889317	Дмитренко О.О.	вул. Харківське шосе 121	Failed	Тестова інспекція #1, авто-ген
2025-12-14 19:41:14.847777	Дмитренко О.О.	вул. Харківське шосе 121	Passed	Готово до навчального року
2025-12-13 19:41:14.889317	Інспектор Нічний	вул. Тестова 3 34	Passed	Тестова інспекція #3, авто-ген
2025-12-12 19:41:14.847777	Ковальчук В.В.	вул. Володимирська 33	Needs Improvement	Відсутній покажчик входу
2025-12-09 19:41:14.889317	Ковальчук В.В.	вул. Саксаганського (ген) 163	Passed	Тестова інспекція #2, авто-ген
2025-12-09 19:41:14.889317	Інспектор Денний	вул. Велика Васильківська (ген) 167	Needs Improvement	Тестова інспекція #2, авто-ген
2025-12-09 19:41:14.847777	Дмитренко О.О.	просп. Голосіївський 42	Failed	Затоплено водою

Рис. 7.25 – результат виконання команди

7.4.9 Знайти інспекторів, які ставили оцінку "Failed" укриттям комунальної форми власності.

```
SELECT DISTINCT
    insp.name,
    insp.phone
FROM inspector insp
JOIN inspection i ON insp.id = i.inspector_id
JOIN shelter s ON i.shelter_id = s.id
JOIN organization o ON s.organization_id = o.id
WHERE i.status = 'Failed'
    AND o.type = 'Комунальна';
```



name	phone
Інспектор Денний	104
Інспектор Нічний	103
Інспектор Резерв	105
Дмитренко О.О.	101
Ковальчук В.В.	102

(5 rows)

Рис. 7.26 – результат виконання команди

7.4.10 Показати деталі активних маршрутів: ID маршруту, адреса цільового укриття та загальна дистанція.

```
SELECT
    r.id AS route_id,
    a.street || ' ' || a.building AS destination_shelter,
    r.distance,
    r.notes
FROM route r
JOIN shelter s ON r.shelter_id = s.id
JOIN address a ON s.address_id = a.id
WHERE r.is_active = TRUE;
```

route_id	destination_shelter	distance	notes
10	вул. Володимирська 33	69599.92	Швидкий (умовно) → укриття #2
9	вул. Володимирська 33	14458.23	Через головні вулиці → укриття #2
17	бул. Лесі Українки 26	43367.66	Маршрут без підйомів (умовно) → укриття #3
16	бул. Лесі Українки 26	6652.96	Резервний → укриття #3
15	бул. Лесі Українки 26	46821.22	Обхід через двори → укриття #3
84	бул. Лесі Українки 26	3489.14	
18	вул. Сагайдачного 10	88918.02	Через головні вулиці → укриття #4
3	вул. Сагайдачного 10	2434.62	Поділ → укриття #4 (правильний фініш)
2	вул. Сагайдачного 10	1904.75	Поділ → БЦ (через центр)

Рис. 7.27 – результат виконання команди

7.4.11 Вивести всі зупинки для конкретного маршруту (наприклад, ID=1) з адресами зупинок.

```
SELECT
    rs.route_id,
    rs.stop_order,
    rs.kind,
```

```
    a.street,
    a.building
FROM route_stop rs
JOIN route r ON rs.route_id = r.id
JOIN address a ON rs.address_id = a.id
WHERE r.id = 4
ORDER BY rs.stop_order;
```

```
route_id | stop_order |     kind     |        street        | building
---------+------------+--------------+----------------------+----------
       4 |          1 | Start        | просп. Голосіївський | 42
       4 |          2 | Intermediate | вул. Володимирська   | 33
       4 |          3 | Intermediate | вул. Хрещатик        | 1
       4 |          4 | Finish       | вул. Харківське шосе | 121
(4 rows)
```

Рис. 7.28 – результат виконання команди

7.4.12 Знайти маршрути, які ведуть до сховищ (Bomb Shelter) з місткістю більше 100 осіб.

```
SELECT
    r.id AS route_id,
    s.type,
    s.capacity,
    a.street AS shelter_street
FROM route r
JOIN shelter s ON r.shelter_id = s.id
JOIN address a ON s.address_id = a.id
WHERE s.type = 'Bomb Shelter'
    AND s.capacity > 100;
```

route_id	type	capacity	shelter_street
24	Bomb Shelter	400	вул. Харківське шосе
25	Bomb Shelter	400	вул. Харківське шосе
26	Bomb Shelter	400	вул. Харківське шосе
75	Bomb Shelter	101	вул. Тестова 13
76	Bomb Shelter	101	вул. Тестова 13
4	Bomb Shelter	400	вул. Харківське шосе
1	Bomb Shelter	300	вул. Хрещатик
5	Bomb Shelter	300	вул. Хрещатик
77	Bomb Shelter	101	вул. Тестова 13

Рис. 7.29 – результат виконання команди

7.4.13 Вивести перелік адрес, які є початковими точками ("Start") для будь-яких активних маршрутів.

```
SELECT
    r.id AS route_id,
    a.street,
    a.building
FROM route r
JOIN route_stop rs ON r.id = rs.route_id
JOIN address a ON rs.address_id = a.id
WHERE r.is_active = TRUE
    AND rs.kind = 'Start';
```

route_id	street	building
1	вул. Володимирська	33
2	вул. Сагайдачного	10
3	вул. Сагайдачного	10
4	просп. Голосіївський	42
5	вул. Велика Васильківська (ген)	75
6	вул. Жилянська (ген)	124
7	вул. Мечникова (ген)	105
8	вул. Богдана Хмельницького (ген)	122
9	вул. Грушевського (ген)	147

:

Рис. 7.30 – результат виконання команди

7.4.14 Знайти укриття, місткість яких більша за середню місткість усіх укриттів у місті.

```
SELECT
   s.id,
   a.street,
   s.capacity,
   d.name AS district
FROM shelter s
JOIN address a ON s.address_id = a.id
JOIN district d ON a.district_id = d.id
WHERE s.capacity > (
   SELECT AVG(capacity) FROM shelter
);
```

```
 id |        street         | capacity |    district
----+-----------------------+----------+----------------
  4 | вул. Сагайдачного     |      500 | Подільський
  5 | просп. Голосіївський  |      200 | Голосіївський
  6 | вул. Харківське шосе  |      400 | Дарницький
  7 | вул. Тестова 1        |      423 | Подільський
  8 | вул. Тестова 2        |      263 | Печерський
  9 | вул. Тестова 3        |      491 | Подільський
 10 | вул. Тестова 4        |      198 | Дарницький
  1 | вул. Хрещатик         |      300 | Шевченківський
(8 rows)
```

Рис. 7.31 – результат виконання команди

7.4.15 Вивести список організацій, які не мають на балансі жодного не готового ("Not Ready") укриття.

```
SELECT
   o.name,
   o.contacts
FROM organization o
WHERE o.id NOT IN (
   SELECT DISTINCT s.organization_id
```

```
    FROM shelter s
    WHERE s.status = 'Not Ready'
);
```

```
           name            |        contacts
---------------------------+------------------------
 ОСББ "Затишок"            | osbb_zatyshok@gmail.com
 ТОВ "Бізнес-Центр Поділ"  | admin@bcpodol.ua
 ЗОШ №55                   | school55@kyiv.edu
(3 rows)
```

Рис. 7.32 – результат виконання команди

7.4.16 Знайти укриття, які жодного разу не проходили інспекцію (відсутні записи в таблиці inspection).

```
SELECT
    s.id,
    a.street,
    rp.name AS responsible_person
FROM shelter s
JOIN address a ON s.address_id = a.id
JOIN responsible_person rp ON s.responsible_id = rp.id
WHERE s.id NOT IN (
    SELECT DISTINCT shelter_id FROM inspection
);
```

```
 id |     street     |    responsible_person
----+----------------+--------------------------
  8 | вул. Тестова 2 | Сидоренко Олена Петрівна
(1 row)
```

Рис. 7.33 – результат виконання команди

7.4.17 Вивести найновішу інспекцію для кожного укриття.

```
SELECT
    s.id AS shelter_id,
    a.street,
```

```
    i.date AS inspection_date,
    i.status
FROM shelter s
JOIN address a ON s.address_id = a.id
JOIN inspection i ON s.id = i.shelter_id
WHERE i.date = (
    SELECT MAX(date)
    FROM inspection i2
    WHERE i2.shelter_id = s.id
);
```

shelter_id	street	inspection_date	status
2	вул. Володимирська	2025-12-12 19:41:14.847777	Needs Improvement
5	просп. Голосіївський	2025-12-09 19:41:14.847777	Failed
3	бул. Лесі Українки	2025-09-11 19:41:14.889317	Failed
4	вул. Сагайдачного	2025-11-18 19:41:14.889317	Passed
6	вул. Харківське шосе	2025-12-14 19:41:14.889317	Failed
7	вул. Тестова 1	2025-12-04 19:41:14.889317	Needs Improvement
9	вул. Тестова 3	2025-12-13 19:41:14.889317	Passed
10	вул. Тестова 4	2025-11-19 19:41:14.889317	Passed
11	вул. Тестова 5	2025-10-04 19:41:14.889317	Needs Improvement

Рис. 7.34 – результат виконання команди

7.4.18 Знайти укриття, в яких запаси води менші за середній показник запасів води по всіх укриттях.

```
SELECT
    s.id,
    a.street,
    si.value AS water_amount
FROM shelter s
JOIN address a ON s.address_id = a.id
JOIN shelter_inventory si ON s.id = si.shelter_id
WHERE si.item_id = (SELECT id FROM inventory_item WHERE name = 'Вода питна (пляшки)')
    AND si.value < (
        SELECT AVG(value)
```

```
    FROM shelter_inventory
    WHERE item_id = (SELECT id FROM inventory_item WHERE name =
'Вода питна (пляшки)')
);
```

```
 id |    street     | water_amount
----+---------------+--------------
  1 | вул. Хрещатик |       500.00
(1 row)
```

Рис. 7.35 – результат виконання команди

7.4.19 Вибрати райони, в яких кількість готових укриттів менша за 3 (для виявлення проблемних зон).

```
SELECT
    d.name,
    c.name AS city
FROM district d
JOIN city c ON d.city_id = c.id
WHERE d.id IN (
    SELECT a.district_id
    FROM address a
    JOIN shelter s ON a.id = s.address_id
    WHERE s.status = 'Ready'
    GROUP BY a.district_id
    HAVING COUNT(s.id) < 3
);
```

```
     name      | city
---------------+------
 Голосіївський | Київ
(1 row)
```

Рис. 7.36 – результат виконання команди

7.4.20 Показати укриття, які мають всі можливі типи особливостей (фіч), що зареєстровані в системі

```
SELECT
    s.id,
    a.street
FROM shelter s
JOIN address a ON s.address_id = a.id
WHERE NOT EXISTS (
    SELECT f.id FROM feature f
    EXCEPT
    SELECT sf.feature_id FROM shelter_feature sf WHERE sf.shelter_id = s.id
);
```

ДЖЕРЕЛА

1) Бомбосховище. URL: https://uk.wikipedia.org/wiki/%D0%91%D0%BE%D0%BC%D0%B1%D0%BE%D1%81%D1%85%D0%BE%D0%B2%D0%B8%D1%89%D0%B5 (Дата звернення: 01.11.2025)
2) Сховище цивільної оборони. URL: https://uk.wikipedia.org/wiki/%D0%A1%D1%85%D0%BE%D0%B2%D0%B8%D1%89%D0%B5_%D1%86%D0%B8%D0%B2%D1%96%D0%BB%D1%8C%D0%BD%D0%BE%D1%97_%D0%BE%D0%B1%D0%BE%D1%80%D0%BE%D0%BD%D0%B8 (Дата звернення: 01.11.2025)
3) Нова послуга в Дії: знайдіть найближчий Пункт незламності в кілька кліків. URL: https://diia.gov.ua/news/nova-posluha-v-dii-znaidit-naiblyzhchyi-punkt-nezlamnosti-v-kilka-klikiv (Дата звернення: 01.11.2025)
4) Мапа укриттів для населення міста Києва. URL. https://kyivcity.gov.ua/bezpeka_ta_pravoporiadok/vazhlyve_pid_chas_voiennoho_stanu/mapa_ukrittiv_dlya_naselennya_mista_kiyeva/ (Дата звернення: 01.11.2025)
5) Google Maps. URL: https://maps.google.com/ (Дата звернення: 01.11.2025)
6) OpenStreetMap. URL: https://www.openstreetmap.org/ (Дата звернення: 01.11.2025)
7) PostgreSQL. URL: https://www.postgresql.org/download/ (Дата звернення: 01.11.2025)
8) PostGIS. URL: https://postgis.net/ (Дата звернення: 01.11.2025)

ВИСНОВКИ

У ході виконання даної курсової роботи було проведено детальний аналіз предметного середовища системи цивільного захисту міста. Визначено основні завдання та вимоги до розроблюваної бази даних, яка має забезпечувати облік укриттів, контроль їхнього стану та планування евакуації. Робота була спрямована на створення структури, що дозволяє вирішити проблеми децентралізації даних та відсутності актуальної інформації про доступність сховищ.

В результаті аналізу бізнес-правил було спроектовано ER-модель, яка відображає ключові сутності (укриття, адреси, інспекції, маршрути) та зв'язки між ними. Реалізація бази даних виконана у середовищі СУБД PostgreSQL з використанням розширення PostGIS, що дозволило ефективно працювати з географічними координатами та будувати просторові запити.

Створено скрипти для генерації таблиць з необхідними обмеженнями цілісності. Розроблено розмежування прав доступу для різних категорій користувачів: адміністраторів, інспекторів, менеджерів укриттів та звичайних громадян. Це гарантує безпеку даних та захист від несанкціонованих змін критичної інформації.

Значну увагу приділено автоматизації процесів на стороні сервера бази даних. Розроблено комплект збережених процедур та функцій для пошуку найближчих укриттів, розрахунку місткості та перевірки термінів придатності інвентарю. Створено систему тригерів, які забезпечують валідацію маршрутів, автоматичне оновлення статусів укриттів за результатами інспекцій та блокування змін для готових до використання об'єктів. Для зручності вибірки даних реалізовано низку представлень (Views).

Проведено тестування бази даних шляхом наповнення її тестовими даними та виконання серії SQL-запитів різної складності, що підтвердило коректність роботи закладеної логіки.

Мета роботи — створення ефективної бази даних для реєстру укриттів — досягнута. Розроблена система готова до використання як бекенд-складова для міських інформаційних порталів або мобільних застосунків, забезпечуючи надійне зберігання даних та оперативний доступ до життєво важливої інформації в умовах надзвичайних ситуацій.