

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект бази даних організації на основі
PostgreSQL»

Виконав: студент 4-го курсу групи КІ-20-1

спеціальності 123 «Комп'ютерна інженерія»

 Тимур ГЕНСІРОВСЬКИЙ

Керівник: доцент кафедри КІ та КБ, к.пед.н., доцент

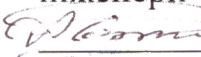
 Олена ГОЛОВНЯ

Рецензент: доцент кафедри КІ та КБ, к.т.н., доцент

 Юрій БРОДСЬКИЙ

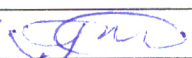
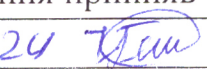
Житомир – 2024

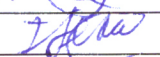
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач
кафедри комп'ютерної
інженерії та кібербезпеки
 Андрій Єфіменко
5 березня 2024 р.

ЗАВДАННЯ
на випускню кваліфікаційну роботу бакалавра
Студент: Генсіровський Тимур Валентинович
Тема роботи: «Проект бази даних організації на основі PostgreSQL»
затверджена Наказом університету від 5 березня 2024 р. № 97/с
Термін здачі студентом закінченої роботи 9 червня 2024 р.
Вихідні дані роботи: Об'єктом дослідження є процес побудови та адміністрування реляційних баз даних. Предметом дослідження є процес побудови та адміністрування бази даних в СУБД PostgreSQL, спрямований на забезпечення її доступності для аутентифікованих користувачів, продуктивності та захищеності даних.

Обладнання та програмне забезпечення:
1. VirtualBox 6.1.38
2. Ubuntu Server 22.04.4 LTS
3. PostgreSQL 14.12
Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Головня О.С.	10.03.24 	8.06.24 

Студент  Тимур ГЕНСІРОВСЬКИЙ
Керівник  Олена ГОЛОВНЯ

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	06.03.24 – 31.03.24	виконано
2	Аналіз теоретичних основ реляційних баз даних	01.04.24 – 07.04.24	викорнано
3	Вибір технічних засобів	08.04.24 – 15.04.24	викорнано
4	Проектування структури та функціоналу бази даних	16.04.24 – 30.04.24	викорнано
5	Проектування політики розмежування прав доступу	01.05.24 – 10.05.24	викорнано
6	Встановлення й налаштування СУБД	11.05.24 – 14.05.24	викорнано
7	Побудова бази даних	15.05.24 – 17.05.24	викорнано
8	Створення додаткового функціоналу бази даних	18.05.24 – 22.05.24	викорнано
9	Реалізація політики розмежування прав доступу	23.05.24 – 27.05.24	викорнано
10	Налаштування резервного копіювання бази даних	28.05.24 – 31.05.24	викорнано
11	Оформлення кваліфікаційної роботи	01.06.24 – 08.06.24	викорнано

Студент Тимур Тимур ГЕНСІРОВСЬКИЙ

Керівник Олена Олена ГОЛОВНЯ

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра на тему «Проект бази даних організації на основі PostgreSQL» складається з вступу, трьох розділів, висновків, списку використаної літератури та додатків. Текстова частина викладена на 67 сторінках друкованого тексту. Пояснювальна записка має 12 сторінок додатків. Список використаних джерел містить 26 найменувань і займає 2 сторінки. В роботі наведено 17 рисунків та 13 таблиць. Загальний обсяг роботи – 80 сторінок.

Метою кваліфікаційної роботи бакалавра є розробка бази даних організації засобами СУБД PostgreSQL.

Перший розділ роботи присвячено ознайомленню з реляційною моделлю даних, причинами її виникнення та перевагами реляційних баз даних, огляду розповсюджених реляційних СУБД та вибору СУБД для розробки бази даних.

Другий розділ роботи присвячено проектуванню структури та функціоналу бази даних, а також аналізу механізму ролей та привілеїв в PostgreSQL та розробці політики розмежування прав доступу до об'єктів бази даних.

Третій розділ роботи присвячено розгортанню та налаштуванню системи. Він включає в себе встановлення та налаштування СУБД, побудову бази даних зі всіма таблицями, збереженням процедурами, функціями, представленнями, створення всіх необхідних ролей бази даних й надання для кожної з них необхідних привілеїв. Останнім етапом роботи над розділом було налаштування резервного копіювання бази даних.

Ключові слова: СУБД, БАЗА ДАНИХ, ТАБЛИЦЯ, ФУНКЦІЯ, ПРОЦЕДУРА, ПРЕДСТАВЛЕННЯ, РОЛЬ.

					KIK.KP.B – 123 – 04 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Проект бази даних організації на основі PostgreSQL	Лім.	Арк.	Аркуші
Розроб.		Генсіровський Т.В.		26.06.24				
Керівник		Головня О.С.		26.06.24			4	80
Реценз.		Бродський Ю.Б.		26.06.24		ФІКТ, Гр. КІ-20-1		
Н. контр.		Головня О.С.		26.06.24				
Затверд.		Єфіменко А.А.		26.06.24				

ANNOTATION

The explanatory note for the bachelor's qualification work on the topic "Organization Database Project Based on PostgreSQL" consists of an introduction, three chapters, conclusions, a list of references, and appendices. The text part is presented on 67 pages of printed text. The explanatory note includes 12 pages of appendices. The list of references contains 26 titles and occupies 2 pages. The work features 17 figures and 13 tables. The total volume of the work is 80 pages.

The purpose of the bachelor's qualification work is to develop an organization database using PostgreSQL DBMS tools.

The first chapter of the work is dedicated to familiarizing with the relational data model, the reasons for its emergence, and the advantages of relational databases, reviewing popular relational DBMS, and selecting a DBMS for database development.

The second chapter of the work is dedicated to designing the structure and functionality of the database, as well as analyzing the roles and privileges mechanism in PostgreSQL and developing a policy for segregating access rights to database objects.

The third chapter of the work is dedicated to deploying and configuring the system. It includes installing and configuring the DBMS, building the database with all tables, stored procedures, functions, views, creating all necessary database roles, and granting the required privileges to each of them. The final stage of the work on the chapter was configuring database backup.

Keywords: DBMS, DATABASE, TABLE, FUNCTION, PROCEDURE, VIEW, ROLE.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП	8
РОЗДІЛ 1. ОГЛЯД ТА ВИБІР ТЕХНІЧНИХ ЗАСОБІВ ДЛЯ ОБСЛУГОВУВАННЯ БАЗИ ДАНИХ.....	10
1.1. Реляційні бази даних	10
1.2. Огляд та вибір СУБД.....	11
1.3. Обґрунтування вибору ОС для розгортання СУБД	14
Висновки до першого розділу	15
РОЗДІЛ 2. ПРОЄКТУВАННЯ БАЗИ ДАНИХ, ПІДГОТОВКА РОБОЧОГО СЕРЕДОВИЩА	16
2.1. Проєктування таблиць та зв'язків між ними	16
2.2. Проєктування функціональності бази даних	21
2.3. Розробка політики розмежування доступу	27
Висновки до другого розділу.....	32
РОЗДІЛ 3. СТВОРЕННЯ ТА НАЛАШТУВАННЯ БАЗИ ДАНИХ	33
3.1. Встановлення та ініціалізація PostgreSQL	33
3.2. Створення бази даних та її складових	36
3.3. Реалізація системи розмежування прав доступу	56
3.4. Налаштування резервного копіювання бази даних.....	60
Висновки до третього розділу	63
ВИСНОВКИ.....	64
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	66
ДОДАТКИ.....	68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – Система управління базами даних

ОС – Операційна система

SQL – Structured query language

ІС – Інформаційна система

ОЗП – Оперативний запам'ятовуючий пристрій

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. В реаліях сьогодення швидкість та ефективність є першочерговими факторами конкурентоспроможності. Впровадження цифрових технологій призвело до значного підвищення продуктивності різноманітних процесів та зменшення витрат на них. У той же час, масова цифровізація цих процесів спричиняє стрімке зростання обсягу даних, що створює необхідність в забезпеченні умов їх ефективного зберігання та обробки. І саме бази даних є основним засобом вирішення даної проблеми.

Бази даних знаходять практичне застосування у багатьох сферах людської діяльності, в тому числі й критично важливих, таких як охорона здоров'я, фінансові операції, електронні комунікації, освіта та наука тощо. Сучасні системи управління базами даних (СУБД) здатні не тільки зберігати інформацію, а й надавати швидкий доступ до неї для багатьох користувачів одночасно, швидко інтегруватися з уже наявними інформаційними системами, забезпечувати цілісність, безпеку даних та можуть містити в собі інструменти для їх аналізу та спрощення прийняття рішень. Відповідно, зі збільшенням обсягів даних у сучасному світі, потреба в ефективно спроектованих та налаштованих базах даних є стабільно високою.

Метою кваліфікаційної роботи є розробка бази даних організації засобами СУБД PostgreSQL.

Визначена мета обумовлює наступні завдання:

- Встановлення та налаштування СУБД PostgreSQL
- Проектування структури бази даних, її побудова
- Проектування й реалізація функціоналу бази даних
- Проектування й реалізація системи розмежування прав доступу до об'єктів бази даних
- Налаштування автоматичного резервного копіювання бази даних

Об'єктом дослідження є процес побудови та адміністрування реляційних баз даних.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

Предметом дослідження є процес побудови та адміністрування бази даних в СУБД PostgreSQL, спрямований на забезпечення її доступності для аутентифікованих користувачів, продуктивності та захищеності даних.

У процесі виконання кваліфікаційної роботи бакалавра було розроблено проект бази даних кадрової агенції. Процес розробки передбачає проектування структури бази даних, визначення зв'язків між її складовими, розробку функціоналу бази даних, розробку політики розмежування прав доступу для користувачів, налаштування автоматичного резервного копіювання бази даних.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ОГЛЯД ТА ВИБІР ТЕХНІЧНИХ ЗАСОБІВ ДЛЯ ОБСЛУГОВУВАННЯ БАЗИ ДАНИХ

1.1. Реляційні бази даних

Система управління базами даних (СУБД) – це вид програмного забезпечення, що дозволяє адміністраторам баз даних, програмним додаткам і кінцевим користувачам зберігати, упорядковувати, зчитувати та маніпулювати даними в базах даних.

Реляційні СУБД є основоположним засобом для керування базами даних, побудованих на основі реляційної моделі. Реляційна модель даних передбачає організацію даних у вигляді сукупності структурованих таблиць, що пов'язані одна з одною на основі попередньо визначених правил для забезпечення цілісності даних. Такий підхід сприяє ефективному зберіганню, пошуку та модифікації даних. Протягом останніх десятиліть реляційні СУБД зазнали значної еволюції, адже системам від різних розробників необхідно було постійно підлаштовуватися під зростаючі масштаби проектів та потреби користувачів.

Реляційна модель даних була розроблена для вирішення проблеми багатьох довільних структур даних. Вона стала значним кроком вперед, наприклад, у порівнянні з мережевою моделлю даних. Замість того, щоб використовувати зв'язки типу «власник-учасник», реляційна модель дозволяє пов'язати один файл з будь-яким іншим за допомогою спільного поля. Складність дизайну значно зменшилася, оскільки з'явилася можливість вносити зміни до схеми бази даних, не впливаючи на здатність системи отримувати доступ до даних [1]. Тому, оскільки доступ здійснювався не за допомогою шляхів до та з файлів, а через прямі зв'язки між цими файлами, додати нові зв'язки було відносно легко.

Реляційні бази даних також зазвичай асоціюються з транзакційними базами даних, оскільки вони реалізують механізм транзакцій. Суть транзакцій полягає в тому, що вони об'єднують кілька кроків в одну операцію за принципом «все або нічого» [2]. Проміжні стани між кроками невидимі для інших одночасних транзакцій, і якщо виникає якась помилка, що перешкоджає завершенню

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

транзакції, то жоден із кроків не вплине на базу даних взагалі. Класичним прикладом транзакції є те, що відбувається, коли клієнт знімає гроші зі свого банківського рахунку. Гроші або були зняті з банківського рахунку, або ні – проміжного стану бути не може. Транзакції визначаються чотирма найважливішими властивостями: атомарністю, узгодженістю, ізоляцією та довговічністю (англ. atomicity, consistency, isolation, durability, ACID).

- **Атомарність** гарантує, що всі операції, що складають повну транзакцію, будуть виконані як одне ціле: або виконаються усі операції, або не виконається жодна з них.
- **Узгодженість** передбачає, що система перебуватиме в узгодженому стані як до, так і після транзакції.
- **Ізоляція** передбачає, що транзакції, які відбуваються одночасно, на впливатимуть на роботу одна одної.
- **Довговічність** гарантує, що зміни, спричинені здійсненням транзакції будуть збережені, незалежно від будь-яких технічних проблем.

Таким чином, даний набір вимог можна вважати стандартом для реляційних СУБД. Втім, дотримання такої концепції несе в собі й певні недоліки. Наприклад, це суттєво ускладнює побудову великомасштабних розподілених систем, а також зумовлює проблему конкурентного оновлення даних: одночасні транзакції, які намагаються змінити одні й ті самі дані, вимагаючи, щоб одна транзакція чекала завершення іншої, знижують продуктивність системи та викликають затримки в її роботі.

1.2. Огляд та вибір СУБД

Розглянемо популярні СУБД з метою вибору СУБД, на основі якої буде розроблено базу даних організації у межах даної кваліфікаційної роботи.

Oracle Database – це пропрієтарна об’єктно-реляційна система керування базами даних від корпорації Oracle. Дана СУБД цінується за свою надійність, високу продуктивність і розширені функції, такі як Real Application Clusters (RAC) для відмовостійкості та масштабованості. Це робить її фаворитом для

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

критично важливих систем великих підприємств. Крім того, Oracle пропонує глибокі можливості інтеграції з хмарними платформами, сприяючи створенню хмарних і гібридних архітектур. Функції безпеки Oracle Database, такі як розширене шифрування та маскування даних, додатково забезпечують захист конфіденційної інформації, що робить її популярною СУБД для галузей, де потрібен високий рівень захисту даних. З недоліків можна зазначити те, що її досконале опанування займає відносно багато часу, а також платність її ліцензії для організацій.

MySQL – найрозповсюдженіша СУБД з відкритим кодом [3], яка підтримується корпорацією Oracle. Маючи відкритий код, MySQL пропонує економічні переваги, що особливо важливо для стартапів, малих та середніх підприємств. Її розповсюдженість зумовлена високою швидкістю роботи, надійністю та великою спільнотою. Крім того, широке впровадження даної СУБД призвело до появи великої кількості інструментів та інтеграцій з іншими системами. Модульна архітектура MySQL дозволяє застосовувати широкий вибір конфігурацій, задовольняючи різні потреби продуктивності. З недоліків можна зазначити, що деякі додаткові функції, доступні в інших комерційних СУБД, можуть бути відсутні в MySQL.

Microsoft SQL Server – це пропріетарна СУБД від корпорації Microsoft. Програми та інструменти підключаються і обмінюються з нею даними за допомогою мови Transact-SQL (T-SQL). Дана СУБД забезпечує зручну інтеграцію з іншими продуктами Microsoft, що робить її природним вибором для компаній, які орієнтуються на екосистему цієї корпорації. Втім, на сьогоднішній день дана СУБД може працювати і в ОС сімейства Linux [4]. Microsoft SQL Server відома простотою використання, потужними функціями безпеки та широким набором інструментів. Її хмарна інтеграція з сервісом Azure спрощує масштабованість і відновлення даних. Її інтегровані інструменти бізнес-аналітики, такі як Power BI та SSRS, забезпечують динамічне звітування та аналітику для клієнтів. Недоліком є те, що витрати на ліцензування корпоративної версії можуть бути доволі

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

високими. Крім того, можлива відносно висока складність опанування даної СУБД для спеціалістів, не знайомих з екосистемою Microsoft.

PostgreSQL – це безкоштовна об’єктно-реляційна СУБД з відкритим джерельним кодом, відома своєю розширюваністю та підтримкою як реляційних, так і не реляційних моделей даних. Її розширені типи даних та можливості індексування роблять її чудово адаптованою до різноманітних завдань [5]. PostgreSQL може працювати на всіх основних операційних системах, включаючи Linux, FreeBSD, OpenBSD, macOS і Windows, і здатна обробляти низку робочих навантажень від окремих машин до сховищ даних або веб-сервісів із багатьма одночасними користувачами. Можливі недоліки: відносно висока складність опанування, а також невелика кількість спеціалістів, що мають досвід роботи з даною СУБД, оскільки попит на цю систему почав зростати відносно нещодавно. Крім того, у середовищах із високою кількістю транзакцій PostgreSQL може потребувати додаткового налаштування для оптимальної продуктивності.

Після огляду найрозповсюдженіших реляційних СУБД було вирішено використовувати об’єктно-реляційну СУБД PostgreSQL. Розглянемо найбільш вагомні аргументи на користь використання даної системи для задоволення потреб як малих, так і великих підприємств.

Перш за все, слід зауважити, що PostgreSQL є безкоштовною системою з відкритим кодом, що дозволяє модифікувати її під унікальні потреби замовника та робить її доступною для використання без ліцензійних витрат [6]. Це може бути визначним фактором для новостворених підприємств. При цьому, PostgreSQL пропонує багатий набір функціональних можливостей, включаючи підтримку різноманітних типів даних [7], складних запитів, процедур, тригерів, геопросторових даних та інших можливостей. Це робить PostgreSQL гнучким інструментом для обслуговування різних систем, включаючи веб-додатки, аналітичні системи, системи керування контентом та інші. Не менш значною перевагою є широкі можливості масштабування даної СУБД. PostgreSQL можна масштабувати як під невеликі додатки, розраховані на одного користувача, так і для потреб великих корпоративних систем з безліччю користувачів і великими

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

обсягами даних. Вона підтримує різні методи масштабування, включаючи горизонтальне та вертикальне масштабування, а також реплікацію та розподіл даних.

Варто також відзначити, що PostgreSQL має велику спільноту розробників та довгу історію розробки та тестування, що забезпечує швидке виправлення помилок та об'ємну документацію. Це також зумовило потужні можливості PostgreSQL для інтеграції з іншими СУБД, бібліотеками та мовами програмування.

Отже, для розробки бази даних організації у межах даної дипломної роботи обрано СУБД PostgreSQL.

1.3. Обґрунтування вибору ОС для розгортання СУБД

PostgreSQL є багатоплатформною системою. Вона здатна працювати на всіх розповсюджених операційних системах, включаючи Linux, FreeBSD, OpenBSD, macOS та Windows [8]. Однак для зручності розгортання та налаштування СУБД, а також ефективнішої роботи бази даних організації, варто обрати одну конкретну ОС, яка з практичної точки зору є однією з найоптимальніших для роботи даної СУБД.

Історично PostgreSQL розроблялася в першу чергу з огляду на Unix-подібні системи, і при роботі на ОС сімейства Windows вона реалізує ту саму архітектуру. Це означає, що в процесі її роботи виконуються низки операцій, що не є природними для ОС сімейства Windows. Логічно припустити, що на одному й тому ж обладнанні PostgreSQL працюватиме дещо швидше під керівництвом Unix-подібної ОС. В наші дні більшість оптимізацій даної СУБД також орієнтовані в першу чергу на роботу в середовищі Unix [9]. Не менш важливим фактором є те, що при використанні PostgreSQL на Windows більшість антивірусних продуктів будуть виходити з ладу, оскільки вони не адаптовані до такого типу робочого навантаження (наприклад, тисяча різних процесів, які читають і записують дані у один і той самий файл через різні дескриптори). З огляду на це було вирішено зосередитися саме на Unix-подібних ОС.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

В свою чергу, серед Unix-подібних ОС було вирішено обрати саме Ubuntu Server, яка доволі часто використовується для розгортання серверних додатків. Серед основних переваг даної серверної ОС можна відзначити безкоштовність ліцензії, відносно простий процес встановлення та оновлення системи та обширну документацію [10]. Ubuntu Server надаються регулярні оновлення безпеки та підтримка тривалого терміну для версій LTS (Long Term Support). Це забезпечує безпеку та стабільність операційної системи протягом тривалого часу, що важливо для критичних серверів баз даних. Окремо варто відзначити файлову систему ext4, що використовується в Ubuntu Server за замовчуванням. Вона забезпечує високу швидкість введення-виведення, що є важливим для баз даних, оскільки в них часто виконуються операції читання та запису даних.

Висновки до першого розділу

В даному розділі було розглянуто загальні характеристики реляційних баз даних, причини їх виникнення та переваги цієї концепції. Розглянуто також розповсюджені реалізації реляційних СУБД та аспекти їх застосування. В якості СУБД для розробки бази даних було обрано об'єктно-реляційну систему PostgreSQL. В якості ключової переваги даної СУБД було відзначено досить широку функціональність при повній безкоштовності системи. Також було проаналізовано сумісність PostgreSQL з найрозповсюдженішими серверними ОС, після чого в якості системи для розгортання СУБД було обрано Ubuntu Server.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ БАЗИ ДАНИХ, ПІДГОТОВКА РОБОЧОГО СЕРЕДОВИЩА

2.1. Проєктування таблиць та зв'язків між ними

База даних проєктуватиметься як складова інформаційної системи кадрової агенції. Кадрова агенція – це організація, що надає послуги з підбору персоналу для роботодавців та пошуку роботи для претендентів.

Найважливіший крок при проєктуванні бази даних – визначення ключових понять предметної області, відповідних таблиць для зберігання інформації про них та визначення зв'язків між цими таблицями. В реляційних базах даних зв'язки між таблицями встановлюються з використанням первинних та зовнішніх ключів. Первинний ключ (англ. primary key) – це поле або сукупність полів (стовпців), що однозначно ідентифікує кожний запис таблиці. Зовнішній ключ (англ. foreign key) – це поле або сукупність полів, що посиляються на первинні ключі іншої або тієї ж таблиці, тобто містять в собі значення, що обов'язково присутні в полях первинного ключа цієї таблиці. Така концепція має назву посилальна цілісність (англ. referential integrity).

В PostgreSQL передбачено широкий спектр типів даних для визначення стовпців таблиць: числові, символні, типи дати і часу, геометричні тощо. В межах даної роботи будуть використовуватися переважно такі типи:

- integer (int) – цілі числа, на зберігання яких відводиться 4 байти
- serial – Integer з автоінкрементом
- character varying(n) (varchar(n)) – набір символів змінної довжини з обмеженням в n символів
- text – набір символів необмеженої довжини
- boolean – логічний тип, що визначає два можливих стани: true (“істина”) або false (“хибність”)
- date – дати, за замовчуванням в форматі “рік-місяць-день”
- timestamp – мітка часу Unix

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Перш за все необхідно визначити таблицю, в якій зберігатиметься технічна інформація про користувачів системи (users). Ці дані включають ім'я користувача, хеш паролю, адресу електронної пошти, а також призначення, дату і час створення облікового запису. Дана таблиця не матиме жодних зовнішніх ключів. Структуру таблиці users описано в таблиці 2.1.

Таблиця 2.1

Структура таблиці «Користувачі» (users)

Назва поля	Тип даних	РК	FK	Опис поля
id	serial	+	-	Ідентифікатор запису
username	varchar(50)	-	-	Ім'я користувача
password	varchar(100)	-	-	Хеш паролю
email	varchar(150)	-	-	Адреса електронної пошти
role	varchar(20)	-	-	Призначення облікового запису
created_at	timestamp	-	-	Дата і час створення облікового запису

Наступна таблиця має назву job_seekers (табл. 2.2). Вона призначена для зберігання персональних даних тих користувачів, що є претендентами. До такої інформації можна віднести: ім'я, прізвище, дату народження, номер телефону, дані про освіту, досвід та навички претендента. Окрім цього, кожен запис має містити посилання на службову інформацію про користувача. Для цього таблиця job_seekers має один зовнішній ключ – поле user_id, за яким вона посилається на таблицю users.

Таблиця 2.2

Структура таблиці «Претенденти» (job_seekers)

Назва поля	Тип даних	РК	FK	Опис поля
id	serial	+	-	Ідентифікатор запису
user_id	integer	-	+	Посилання на таблицю users
name	varchar(50)	-	-	Ім'я
surname	varchar(50)	-	-	Прізвище

Продовження таблиці 2.2

Назва поля	Тип даних	РК	FK	Опис поля
birth_date	date	-	-	Дата народження
phone	varchar(30)	-	-	Номер телефону
education	text	-	-	Інформація про освіту
experience	text	-	-	Досвід претендента
skills	text	-	-	Навички претендента

Для зберігання інформації про роботодавців також необхідно відвести окрему таблицю (employers). В ній мають зберігатися такі дані: назва компанії, посилання на службові дані користувача, чисельність штату, контактний номер телефону, юридична адреса, адреса веб-сайту (табл. 2.3). Таблиця employers, як і таблиця job_seekers, має зовнішній ключ user_id, за яким вона посиляється на таблицю users.

Таблиця 2.3

Структура таблиці «Роботодавці» (employers)

Назва поля	Тип даних	РК	FK	Опис поля
id	serial	+	-	Ідентифікатор запису
user_id	integer	-	+	Посилання на таблицю users
company_name	varchar(100)	-	-	Назва компанії
staff_size	integer	-	-	Кількість працівників
phone	varchar(30)	-	-	Контактний номер телефону
address	varchar(150)	-	-	Юридична адреса
website	varchar(150)	-	-	Адреса веб-сайту

Таблиця vacancies (табл. 2.4) призначена для зберігання даних про вакансії. Стосовно кожної вакансії зберігаються наступні відомості: назва, посилання на роботодавця, інформація про заробітну плату, обов'язки та вимоги до кандидата, форма зайнятості, дата та час розміщення вакансії та внесення останніх змін.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Таблиця vacancies має зовнішній ключ employer_id, за яким вона посилається на таблицю employers.

Таблиця 2.4

Структура таблиці «Вакансії» (vacancies)

Назва поля	Тип даних	РК	ФК	Опис поля
id	serial	+	-	Ідентифікатор запису
employer_id	integer	-	+	Посилання на таблицю employers
title	varchar(150)	-	-	Назва вакансії
salary	varchar(50)	-	-	Заробітна плата
responsibilities	text	-	-	Обов'язки працівника
requirements	text	-	-	Вимоги
is_remote	boolean	-	-	Чи є робота віддаленою
created_at	timestamp	-	-	Дата і час розміщення
updated_at	timestamp	-	-	Дата і час останнього редагування

Таблиця responses (табл. 2.5) призначена для зберігання відгуків на вакансії. Кожен запис цієї таблиці має містити посилання на претендента, що надіслав відгук, посилання на відповідну вакансію, статус та дату й час надсилання відгуку. Відповідно, таблиця responses має два зовнішніх ключа: job_seeker_id, за яким посилається на таблицю job_seekers, і vacancy_id, за яким посилається на таблицю vacancies.

Таблиця 2.5

Структура таблиці «Відгуки» (responses)

Назва поля	Тип даних	РК	ФК	Опис поля
id	serial	+	-	Ідентифікатор запису
job_seeker_id	integer	-	+	Посилання на таблицю job_seekers
vacancy_id	integer	-	+	Посилання на таблицю vacancies
status	varchar(20)	-	-	Статус відгуку
sent_at	timestamp	-	-	Дата і час надсилання

Таблиця recommendations (табл. 2.6) призначена для зберігання рекомендацій, наданих претендентам співробітниками агенції. Кожна рекомендація характеризується такими даними, як посилання на претендента, якому присвячена рекомендація; посилання співробітника агенції, що є автором рекомендації; посилання на відповідну вакансію; відмітка про те, чи була рекомендація прийнята та дата і час її надсилання. Таблиця recommendations має одразу три зовнішніх ключа – job_seeker_id, adviser_id та vacancy_id, за якими вона посилається на таблиці job_seekers, users та vacancies відповідно.

Таблиця 2.6

Структура таблиці «Рекомендації» (recommendations)

Назва поля	Тип даних	PK	FK	Опис поля
id	serial	+	-	Ідентифікатор запису
job_seeker_id	integer	-	+	Посилання на таблицю job_seekers
consultant_id	integer	-	+	Посилання на таблицю users
vacancy_id	integer	-	+	Посилання на таблицю vacancies
is_accepted	boolean	-	-	Чи прийнято рекомендацію
sent_at	timestamp	-	-	Дата і час надсилання

Остання таблиця (messages) призначена для зберігання повідомлень, якими будуть обмінюватися між собою користувачі. Вона містить наступні дані: посилання на користувача, що надіслав повідомлення, і користувача, якому воно призначено, текст повідомлення й дату і час надсилання (табл. 2.7). Таблиця має два зовнішніх ключа – sender_id і receiver_id, за якими посилається на одну й ту саму таблицю – users.

Таблиця 2.7

Структура таблиці «Повідомлення» (messages)

Назва поля	Тип даних	PK	FK	Опис поля
id	serial	+	-	Ідентифікатор запису
sender_id	integer	-	+	Посилання на відправника (таблиця users)

Назва поля	Тип даних	PK	FK	Опис поля
receiver_id	integer	-	+	Посилання на отримувача (теж таблиця users)
content	text	-	-	Текст повідомлення
sent_at	timestamp	-	-	Дата і час надсилання

На рисунку 2.1 зображено діаграму «сутність-зв'язок» спроектованої бази даних, з позначенням всіх зв'язків між таблицями.

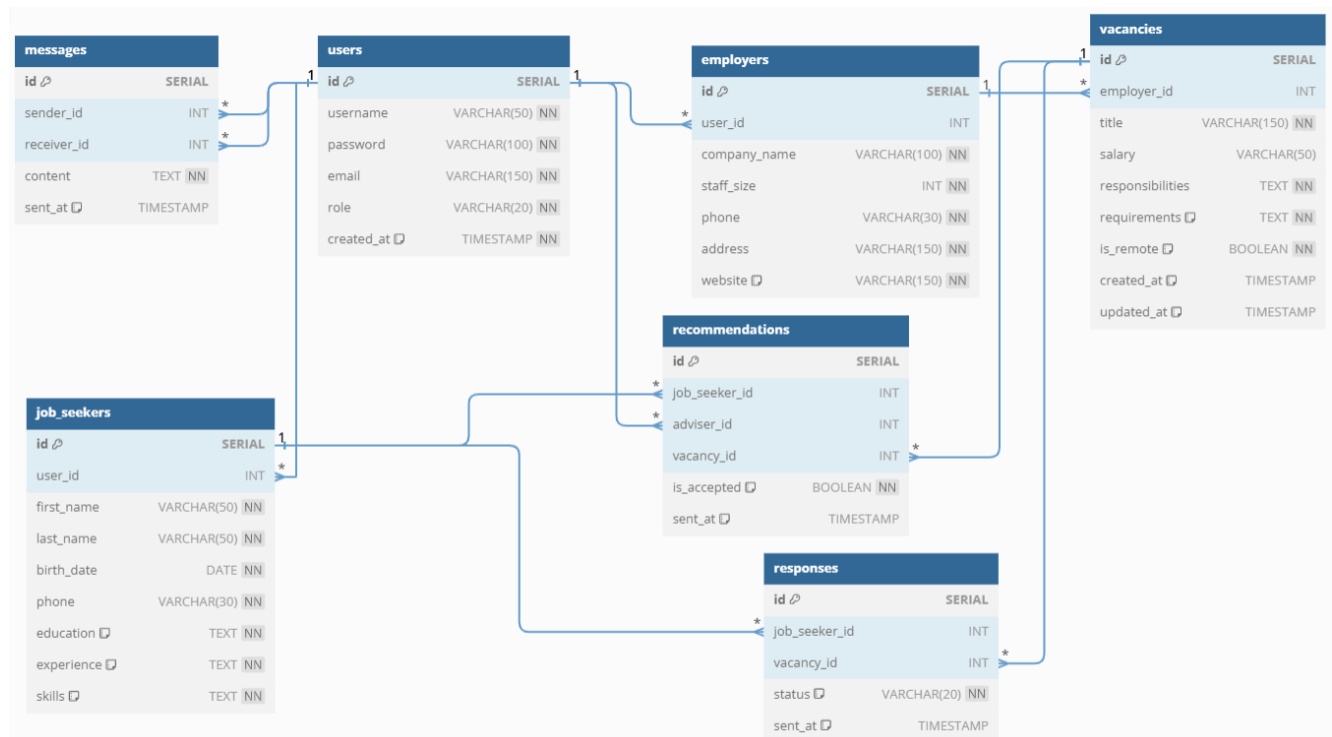


Рис. 2.1. Діаграма «сутність-зв'язок» бази даних

2.2. Проектування функціональності бази даних

Окрім визначення всіх необхідних таблиць для зберігання даних, слід також продумати механізми маніпулювання цими даними, необхідні для обслуговування бізнес-потреб організації. Для цього можна використати такі засоби, як представлення, збережені процедури та функції.

Збережена процедура – це набір інструкцій SQL, який створюється і зберігається на сервері СУБД для подальшого виконання. Процедури являють собою іменовані блоки коду, які можна запустити у будь-який час з метою виконання серії операцій. Зазвичай вони використовуються для інкапсуляції та

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

оптимізації виконання повторюваних операцій у базі даних. Переваги використання збережених процедур є наступними.

1. Інкапсуляція логіки. Збережені процедури дають можливість інкапсулювати бізнес-логіку на рівні бази даних. Це спрощує управління та підтримку функціоналу інформаційної системи, оскільки він стає зосередженим в одному місці.
2. Повторне використання. Процедури можна викликати необхідну кількість разів на різних етапах роботи ІС. Це дозволяє зменшити дублювання цілих блоків коду.
3. Забезпечення атомарності. Усі інструкції у процедурі виконуються в межах однієї транзакції, тому або всі вони будуть виконані, або не буде виконано жодної з них, і дані у базі повернуться у початковий стан. Це дозволяє уникнути порушення цілісності даних в результаті невдало виконаних команд.
4. Валідація даних. Процедури дозволяють визначати правила перевірки даних на коректність перед їх збереженням.
5. Безпека даних. Можна обмежити клієнтам безпосередній доступ до таблиць і надати права виключно на виконання процедур, що виконують чітко визначені операції відносно цих таблиць.
6. Зменшення обсягу мережевого трафіку. Оскільки весь код процедури зберігається на стороні сервера бази даних, обсяг даних, що передаються між клієнтом і сервером зменшується. Це особливо актуально при частому виконанні великих запитів.
7. Підвищення продуктивності. За рахунок попередньої компіляції та збереження планів виконання процедур, операції, що в них містяться, зазвичай виконуються швидше, ніж звичайні запити.

Створюються збережені процедури командою `CREATE PROCEDURE`. Нижче наведено перелік та опис процедур, розроблених для автоматизації деяких задач в базі даних. Загалом ці процедури можна поділити на три групи.

1. Процедури для створення нових записів в таблицях.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Процедури для оновлення вже наявних записів.

3. Процедури для видалення записів.

Перш за все варто визначити процедури для збереження нових даних в таблицях (табл. 2.8). Ці процедури інкапсулюють операції вставки даних (INSERT), а їх параметри відповідають параметрам об'єктів та суб'єктів, для збереження яких вони призначені.

Таблиця 2.8

Процедури для збереження нових записів

Назва	Таблиця	Параметри	Опис параметрів
add_user	users	username	Ім'я користувача
		password	Пароль
		email	Адреса ел. пошти
		role	Тип користувача
add_job_seeker	job_seekers	user_id	Посилання на користувача
		name	Ім'я
		surname	Прізвище
		birth_date	Дата народження
		phone	Номер телефону
		education	Інформація про освіту
		experience	Інформація про досвід
		skills	Інформація про навички
add_employer	employers	user_id	Посилання на користувача
		company_name	Назва
		staff_size	Кількість працівників
		phone	Номер телефону
		address	Адреса
		website	Веб-сайт
add_vacancy	vacancies	employer_id	Посилання на роботодавця
		title	Назва вакансії
		salary	Заробітна плата
		responsibilities	Обов'язки
		requirements	Вимоги до претендентів

Продовження таблиці 2.8

Назва	Таблиця	Параметри	Опис параметрів
add_vacancy	vacancies	is_remote	Чи є робота віддаленою
send_response	responses	job_seeker_id	Посилання на претендента
		vacancy_id	Посилання на вакансію
recommend_vacancy	recommendations	job_seeker_id	Посилання на претендента
		consultant_id	Посилання на консультанта
		vacancy_id	Посилання на вакансію
send_message	messages	sender_id	Посилання на відправника
		receiver_id	Посилання на отримувача
		content	Текст повідомлення

Наступна група процедур (табл. 2.9) призначена для модифікації вже наявних в таблицях даних. Першим параметром цих процедур є ідентифікатор (первинний ключ) запису, а інші параметри приймають значення, що заносяться в поля запису з відповідним ідентифікатором.

Таблиця 2.9

Процедури для оновлення наявних записів

Назва	Таблиця	Параметри	Опис параметрів
update_user	users	user_id	Ідентифікатор користувача
		username	Ім'я користувача
		email	Адреса ел. пошти
update_job_seeker	job_seekers	job_seeker_id	Ідентифікатор претендента
		name	Ім'я
		surname	Прізвище
		birth_date	Дата народження
		phone	Номер телефону
		education	Інформація про освіту
		experience	Інформація про досвід
update_employer	employers	skills	Інформація про навички
		employer_id	Ідентифікатор роботодавця
		company_name	Назва
		staff_size	Кількість працівників

Назва	Таблиця	Параметри	Опис параметрів
update_employer	employers	phone	Номер телефону
		address	Адреса
		website	Веб-сайт
update_vacancy	vacancies	vacancy_id	Ідентифікатор вакансії
		title	Назва вакансії
		salary	Заробітна плата
		responsibilities	Обов'язки
		requirements	Вимоги до претендентів
		is_remote	Чи є робота віддаленою
update_response	responses	response_id	Ідентифікатор відгуку
		status	Статус відгуку
update_recommendation	recommendations	recommendation_id	Ідентифікатор рекомендації
		is_accepted	Чи прийнято рекомендацію
update_message	messages	message_id	Ідентифікатор посилання
		content	Текст повідомлення

Третя група процедур призначена для видалення записів з таблиць. Кожна така процедура приймає лише один параметр – ідентифікатор запису, що підлягає видаленню. Таким чином, було визначено наступні процедури:

- delete_user – для таблиці «Користувачі»
- delete_job_seeker – для таблиці «Претенденти»
- delete_employer – для таблиці «Роботодавці»
- delete_vacancy – для таблиці «Вакансії»
- delete_response – для таблиці «Відгуки»
- delete_recommendation – для таблиці «Рекомендації»
- delete_message – для таблиці «Повідомлення»

Функція в PostgreSQL – це іменований блок коду, що приймає дані в якості аргументів, виконує набір операцій і повертає результат певного типу. Для створення функцій використовується команда CREATE FUNCTION.

Зважаючи на структуру таблиці «Користувачі», має сенс визначити функцію, що повертатиме всі дані користувача за його ідентифікатором, окрім паролю. Хоча паролі й зберігатимуться у вигляді хешу, надавати будь-яким користувачам доступ до них є зайвим ризиком. Тому для безпечнішого доступу до даних користувачів слід визначити функцію `show_user` (табл. 10), й надати усім користувачам доступ виключно до цієї функції, а безпосередній перегляд даних таблиці `users`, включаючи поле `password`, заборонити.

Ще одним корисним рішенням буде визначення функції, що обчислює вік претендента на основі його дати народженням станом на поточну дату. Зберігати вік безпосередньо у вигляді кількості років нераціонально, оскільки його значення для кожного користувача потрібно буде періодично оновлювати. Водночас, працівникам агенції та роботодавцям для спрощення відбору претендентів було б зручно бачити саме їхній вік, а не дату народження. Таким чином було вирішено визначити ще одну функцію – `calculate_age` (табл. 2.10).

Таблиця 2.10

Функції, що будуть створені в базі даних

Назва	Вхідні параметри	Типи параметрів	Тип результату
<code>show_user</code>	Ідентифікатор користувача	<code>integer</code>	<code>table</code>
<code>calculate_age</code>	Дату народження	<code>date</code>	<code>integer</code>

Представлення – це іменований SQL-запит, що зберігається на сервері СУБД. Воно виконується, коли у зовнішньому запиті вказується його ім'я. Представлення можна поділити на два типи: віртуальні та матеріалізовані. Віртуальні, або ж звичайні представлення зберігаються лише як код запитів, без результатів їхньої роботи. Результуючі набори таких представлень формуються при кожному їх виконанні. На противагу цьому, результуючі набори матеріалізованих представлень зберігаються в базі даних як звичайні таблиці. Доступ до їхніх даних, як правило, відбувається суттєво швидше, ніж у випадку з базовими таблицями чи звичайними представленнями. Однак, для підтримки актуальності цих даних, матеріалізовані представлення необхідно оновлювати

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

кожного разу, коли у пов'язаних з ними таблицях відбуваються зміни, а отже, операції оновлення даних в такому випадку будуть виконуватися повільніше.

Для полегшення отримання даних та їх аналізу було визначено представлення для відображення наступної інформації: дані про претендентів на роботу разом з їх віком; вакансії та роботодавці, що їх розмістили; розширена інформація про відгуки, що включає дані відповідних претендентів та вакансій; рекомендації претендентам від співробітників агенції; повідомлення разом з їх відправниками та отримувачами (табл. 2.11). Для створення представлень використовується команда CREATE VIEW.

Таблиця 2.11

Представлення бази даних

Назва	Тип	Базові таблиці	Що відображає
job_seekers_with_age	Віртуальне	job_seekers	дані про претендентів разом з їх віком
vacancies_employers	Віртуальне	employers, vacancies	вакансії та роботодавців, що їх розмістили
responses_ext	Віртуальне	job_seekers, responses, vacancies	розширену інформацію про відгуки на вакансії
recommendations_ext	Віртуальне	recommendations, job_seekers, users, vacancies	рекомендації від співробітників агенції
messages_users	Віртуальне	messages, users	повідомлення разом з їх відправниками та отримувачами

Наведений вище перелік збережених процедур, функцій та представлень надає необхідний і достатній функціонал для зручного керування та перегляду даних ІС організації.

2.3. Розробка політики розмежування доступу

Для авторизації та автентифікації в PostgreSQL використовується концепція «ролей». В процесі встановлення на сервері СУБД за замовчуванням створюється роль з ім'ям postgres, що має статус «суперкористувача», тобто адміністратора

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

серверу СУБД. Дана роль має бути використана для першого підключення до PostgreSQL.

В PostgreSQL для керування правами доступу (привілеями) до бази даних використовується концепція ролей. Поняття ролі можна розглядати одночасно і як користувача, і як групу користувачів бази даних, залежно від налаштувань ролі. Таким чином, підхід до керування правами доступу в PostgreSQL дещо відрізняється від концепції в Unix-подібних ОС, де користувачі та групи є окремими поняттями. Така відмінність існує починаючи з версії PostgreSQL 8.1. Ролі можуть мати певний набір прав доступу до об'єктів баз даних, бути власниками і призначати права доступу щодо цих об'єктів іншим ролям. Окрім того, можна надати одній ролі членство в іншій, дозволяючи ролі учасника використовувати права ролі, що в такому випадку виступає групою.

Ролі бази даних існують на рівні всього кластера бази даних, а не окремої бази даних. Кластер баз даних (далі просто кластер) – це сукупність баз даних, що керуються одним і тим самим екземпляром серверу СУБД. В процесі встановлення на сервері СУБД за замовчуванням створюється роль, що має статус «суперкористувача», тобто адміністратора серверу СУБД. Як правило, вона має назву «postgres». Дана роль має бути використана для першого підключення до PostgreSQL. Далі вже можна використовувати цю роль для створення інших ролей. Кожне з'єднання з сервером СУБД виконується від імені певної ролі, і ця роль визначає початкові права доступу у поточному з'єднанні. Ім'я ролі, яке слід використовувати для конкретного з'єднання з базою даних, вказується клієнтом, який ініціює запит на з'єднання, залежно від програми. Багато клієнтських додатків при підключенні до PostgreSQL в Unix-подібних ОС за замовчуванням використовують ім'я поточного користувача ОС. Тому на практиці часто підтримується відповідність між іменами ролей та користувачів ОС.

Оскільки роль визначає набір прав доступу, доступних для підключеного клієнта, важливо відповідально призначати привілеї для кожної ролі у середовищі, розрахованому на використання багатьма користувачами. У таблиці 2.12 наведено перелік привілеїв, що можуть бути надані ролям, та їх опис.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

Привілеї в PostgreSQL

Назва	Типи об'єктів	Що дозволяє
SELECT	Таблиці, стовпці, великі об'єкти	Читання даних
	Послідовності	Використання функції currval
INSERT	Таблиці, стовпці	Вставка даних
UPDATE	Таблиці, стовпці, великі об'єкти	Оновлення даних
	Послідовності	Використання функцій nextval і setval
DELETE	Таблиці	Видалення стовпців таблиці
TRUNCATE	Таблиці	Виконання операції TRUNCATE над таблицею
REFERENCES	Таблиці, стовпці	Створення зовнішніх ключів
TRIGGER	Таблиці	Створення тригерів для таблиці
CREATE	Бази даних, схеми, таблиці, індекси, тимчасові файли	Створення вказаних об'єктів
CONNECT	Бази даних	Підключення до бази даних
TEMPORARY	Бази даних	Створення тимчасових таблиць в базі даних
EXECUTE	Процедури, функції	Виклик процедур та функцій
USAGE	Схеми	Доступ до об'єктів схеми
	Процедурні мови	Створення функції на цих мовах
	Послідовності	Використання функцій currval і nextval
	Домени, типи	Використання для створення таблиць, функцій та інших об'єктів
SET	Параметри	Встановлення значень параметрів конфігурації сервера в поточній сесії
ALTER SYSTEM	Параметри	Налаштування параметрів конфігурації сервера за допомогою команди ALTER SYSTEM

Для забезпечення високого рівня безпеки даних кадрової агенції при використанні її інформаційної системи, в базі даних, окрім ролі адміністратора серверу СУБД (postgres) було визначено наступні ролі.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

1. db_admin – другорядний адміністратор бази даних організації, але не всього кластеру, на відміну від postgres. Навіть в межах даної бази даних володіє дещо меншим набором привілеїв. Наприклад, не може змінювати структуру об'єктів бази даних за допомогою команди ALTER, оскільки власником цих об'єктів залишається роль postgres.
2. consultant – роль, що являє собою співробітника агенції, посередника між претендентами та роботодавцями.
3. job_seeker – роль, призначена для використання претендентами.
4. employer – роль, призначена для використання роботодавцями.
5. guest – роль, від імені якої підключаються неавтентифіковані користувачі. Завдяки цій ролі реалізується гостьовий доступ.

Таблиця 2.13 являє собою матрицю доступу до об'єктів бази даних, рядки якої відповідають об'єктам, а стовпці – ролям. За значеннями у комірці на перехресті певного стовпця та рядка можна побачити, які права доступу має відповідна роль відносно відповідного об'єкту бази даних. Було визначено наступні умовні позначення:

0 – доступ відсутній;

1 – для таблиць і представлень – доступ до читання (операція SELECT), для функцій та процедур – доступ до виклику;

2 – повний доступ. Включає в себе безпосередній доступ до читання (SELECT), вставку (INSERT), оновлення (UPDATE), видалення записів (DELETE) відносно таблиць та представлень, виклик процедур та функцій, а також модифікацію (команди групи ALTER) та видалення (команди групи DROP) самих об'єктів бази даних.

Надання ролям прав на виконання операцій вставки, оновлення та видалення записів безпосередньо не є необхідним, оскільки для додавання нових даних в таблиці використовуються відповідні процедури. Таким чином, дозволи на опосередковане здійснення цих операцій реалізуються шляхом надання дозволів на виконання відповідних процедур.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.13

Матриця доступу до об'єктів бази даних

	postgres	db_admin	consultant	job_seeker	employer	guest
Таблиця users	2	0	0	0	0	0
Таблиця job_seekers	2	1	1	1	1	0
Таблиця employers	2	1	1	1	1	1
Таблиця vacancies	2	1	1	1	1	1
Таблиця responses	2	1	1	1	1	0
Таблиця recommendations	2	1	1	1	0	0
Таблиця messages	2	1	1	1	1	0
Процедура add_user	2	1	0	0	0	1
Процедура add_job_seeker	2	1	0	1	0	0
Процедура add_employer	2	1	0	0	1	0
Процедура add_vacancy	2	1	0	0	1	0
Процедура send_response	2	1	0	1	0	0
Процедура recommend_vacancy	2	1	1	0	0	0
Процедура send_message	2	1	1	1	1	0
Процедура update_user	2	1	1	1	1	0
Процедура update_job_seeker	2	1	0	1	0	0
Процедура update_employer	2	1	0	0	1	0
Процедура update_vacancy	2	1	0	0	1	0
Процедура update_response	2	1	0	0	1	0
Процедура update_recommendation	2	1	0	1	0	0
Процедура update_message	2	1	1	1	1	0
Процедура delete_user	2	1	0	0	0	0
Процедура delete_job_seeker	2	1	0	1	0	0
Процедура delete_employer	2	1	0	0	1	0
Процедура delete_vacancy	2	1	0	0	1	0
Процедура delete_response	2	1	0	1	0	0
Процедура delete_recommendation	2	1	1	0	0	0
Процедура delete_message	2	1	1	1	1	0

Продовження таблиці 2.13

	postgres	db_admin	consultant	job_seeker	employer	guest
Функція show_user	2	1	1	1	1	0
Функція calculate_age	2	1	1	0	1	0
Представлення job_seekers_with_age	2	1	1	0	1	0
Представлення vacancies_employers	2	1	1	1	1	1
Представлення responses_ext	2	1	1	1	1	0
Представлення recommendations_ext	2	1	1	1	0	0
Представлення messages_users	2	1	1	1	1	0

Таким чином, було розроблено політику розмежування прав доступу до об'єктів бази даних організації.

Висновки до другого розділу

Даний розділ було присвячено проектуванню бази даних. Було визначено основні поняття, якими має оперувати ІС організації, і спроектовано відповідні таблиці для зберігання їхніх даних. Також визначено зв'язки між таблицями, що відображають взаємодію між поняттями предметної області ІС.

Проектування функціоналу ІС на рівні бази даних проводилося з урахуванням засобів, що надаються процедурною мовою PL/pgSQL, а саме збережених процедур, функцій, представлень. Розглянуто переваги використання збережених процедур для інкапсуляції SQL-запитів.

Останнім етапом роботи над розділом була розробка політики доступу до об'єктів бази даних. Розглянуто концепцію розмежування прав доступу в PostgreSQL, а саме механізм ролей та привілеїв. Визначено ролі, необхідні для практичного застосування бази даних, та відповідні привілеї для кожної ролі з метою забезпечення високого рівня безпеки даних.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

РОЗДІЛ 3. СТВОРЕННЯ ТА НАЛАШТУВАННЯ БАЗИ ДАНИХ

3.1. Встановлення та ініціалізація PostgreSQL

Інсталяція СУБД PostgreSQL відбуватиметься на заздалегідь підготовлену віртуальну машину з ОС Ubuntu Server 22.04.4. Технічні характеристики віртуальної машини: обсяг ОЗП – 2 ГБ, обсяг дискового накопичувача – 20 ГБ. В якості графічного середовища робочого столу використовується LXDE (Lightweight X11 Desktop Environment).

Перш за все слід оновити списки пакунків та завантажити необхідні оновлення за допомогою команд:

```
sudo apt update
```

```
sudo apt upgrade
```

Пакунки PostgreSQL містяться в стандартних репозиторіях Ubuntu, а отже можуть бути встановленні за допомогою команди:

```
sudo apt install postgresql postgresql-contrib
```

Пакунок postgresql-contrib містить додаткові утиліти та модулі, що розширюють функціонал СУБД. Вивід команди у разі успішного встановлення зображено на рисунку 3.1.

```
The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/postgresql/14/main ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok
update-alternatives: using /usr/share/postgresql/14/man/man1/postmaster.1.gz to
provide /usr/share/man/man1/postmaster.1.gz (postmaster.1.gz) in auto mode
Setting up postgresql-contrib (14+238) ...
Setting up postgresql (14+238) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.7) ...
Scanning processes...
Scanning linux images...
```

Рис. 3.1. Результат успішного виконання команди встановлення

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Після встановлення PostgreSQL можна перевірити, чи було запущено відповідний фоновий процес, виконавши команду:

sudo service postgresql status

Результат виконання команди зображено на рисунку 3.2.

```

srvadmin@server:~$ sudo service postgresql status
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor
   Active: active (exited) since Tue 2024-05-28 13:48:33 UTC; 1h 11min ago
   Process: 3967 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 3967 (code=exited, status=0/SUCCESS)
      CPU: 2ms

May 28 13:48:33 server systemd[1]: Starting PostgreSQL RDBMS...
May 28 13:48:33 server systemd[1]: Finished PostgreSQL RDBMS.

```

Рис. 3.2. Перевірка статусу фонового процесу PostgreSQL

Якщо з тих чи інших причин процес не є активним, його потрібно запустити вручну за допомогою команди:

sudo systemctl start postgresql.service

PostgreSQL, зокрема в ОС сімейства Linux, за замовчуванням налаштовується на однорангову аутентифікацію. Цей механізм пов'язує ролі СУБД з однойменними користувачами ОС. Тобто, якщо на сервері СУБД існує роль з певним ім'ям, підключитися до серверу з використанням даної ролі може лише користувач ОС з ім'ям, ідентичним до імені цієї ролі. Однорангова автентифікація доступна лише в операційних системах, що забезпечують функцію `getpeereid()`, параметр сокету `SO_PEERCRED` чи інші подібні механізми. Тому в ОС автоматично буде створено користувача з ім'ям `postgres`, задля забезпечення можливості підключення до СУБД в якості однойменної ролі. В цьому можна переконатися шляхом пошуку цього ім'я в файлі `/etc/passwd`, в якому зберігається список користувачів та відомості про них (рис. 3.3).

```

srvadmin@server:~$ cat /etc/passwd | grep "postgres"
postgres:x:116:125:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
srvadmin@server:~$

```

Рис. 3.3. Інформація про користувача postgres

Отже, перше підключення до PostgreSQL має відбутися від імені користувача `postgres`. Для взаємодії з сервером СУБД зручно використовувати

консольний клієнт psql. Відповідно, підключитися до серверу (рис. 3.4) можна за допомогою наступної команди:

sudo -u postgres psql

```
srvadmin@server:~$ sudo -u postgres psql
[sudo] password for srvadmin:
psql (14.11 (Ubuntu 14.11-0ubuntu0.22.04.1))
Type "help" for help.

postgres=#
```

Рис. 3.4. Успішна авторизація в PostgreSQL

Однорангову аутентифікацію можна замінити на аутентифікацію за паролем. Для цього необхідно знайти конфігураційний файл pg_hba.conf (в даному випадку знаходиться в каталозі /etc/postgresql/14/main/) і встановити в ньому значення, зображені на рисунку 3.5.

```
# Database administrative login by Unix domain socket
local    all             postgres                                md5

# TYPE      DATABASE     USER        ADDRESS            METHOD

# "local" is for Unix domain socket connections only
local    all             all                                md5
```

Рис. 3.5. Налаштування аутентифікації за паролем

Після цього необхідно перезапустити сервер PostgreSQL командою:

sudo systemctl restart postgresql

За замовчуванням PostgreSQL припускає, що для кожної ролі, що використовується для входу в систему, існує база даних з тим самим ім'ям, що і в цієї ролі, в даному випадку – база даних postgres. Технічно, сервер PostgreSQL не вимагає обов'язкової наявності цієї бази даних, але багато зовнішніх службових програм взаємодіють з сервером з розрахунком на те, що вона існує. В psql назва поточної бази даних відображається на початку запрошення до вводу. Дізнатися назву поточної бази даних й іншу інформацію про підключення можна також за допомогою команди \conninfo (рис. 3.6).

```
postgres=# \conninfo
You are connected to database "postgres" as user "postgres" via socket in
"/var/run/postgresql" at port "5432".
postgres=#
```

Рис. 3.6. Інформація про підключення

Переглянути перелік усіх баз даних, наявних на кластері, та деяку інформацію про них можна за допомогою команди \l (рис. 3.7).

```
postgres=# \l
```

List of databases					
Name	Owner	Encoding	Collate	Ctype	Access privileges
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres +
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	postgres=CTc/postgres +
(3 rows)					

Рис. 3.7. Список баз даних на сервері СУБД

Як видно з виводу команди \l (рис. 3.6), на кластері, окрім postgres, наявні ще дві бази даних: template0 та template1. Це службові бази даних, що створюються під час ініціалізації кластеру й використовуються як шаблони для створення інших баз даних. Їх не варто використовувати безпосередньо.

3.2. Створення бази даних та її складових

Нехай база даних матиме назву staffing_agency. Тоді для створення бази даних потрібно виконати команду:

```
CREATE DATABASE staffing_agency;
```

Після цього варто переконатися в тому, що база даних дійсно була створена, переглянувши список баз даних на сервері (рис. 3.8).

```
postgres=# CREATE DATABASE staffing_agency;
```

```
CREATE DATABASE
```

```
postgres=# \l
```

List of databases					
Name	Owner	Encoding	Collate	Ctype	Access privileges
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
staffing_agency	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	
template0	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	=c/postgres +
template1	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	postgres=CTc/postgres +
(4 rows)					

Рис. 3.8. Перевірка наявності бази даних

Щоб підключитися до створеної бази даних в psql, можна використати команду \c staffing_agency, після чого в запрошенні до вводу відобразатиметься її назва (рис. 3.9).

```
postgres=# \c staffing_agency
You are now connected to database "staffing_agency" as user "postgres".
staffing_agency=#
```

Рис. 3.9. Підключення до створеної бази даних

Перед створенням будь-яких об'єктів в базі даних варто встановити права доступу до них за замовчуванням. Наступні команди забороняють будь-які дії відносно новостворених таблиць, функцій та процедур для всіх ролей, окрім ролі-власника:

```
ALTER DEFAULT PRIVILEGES IN SCHEMA public REVOKE ALL PRIVILEGES
ON TABLES FROM PUBLIC;
```

```
ALTER DEFAULT PRIVILEGES IN SCHEMA public REVOKE EXECUTE ON
FUNCTIONS FROM PUBLIC;
```

Тепер можна перейти до створення таблиць. Для створення таблиці users (Користувачі) необхідно ввести наступний код:

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL,
    email VARCHAR(150) UNIQUE NOT NULL,
    role VARCHAR(20) CHECK (role IN ('db_admin', 'consultant', 'employer',
    'job_seeker')) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL
);
```

Далі наведено код створення таблиці job_seekers (Претенденти):

```
CREATE TABLE job_seekers (
    id SERIAL PRIMARY KEY,
    user_id INT REFERENCES users(id) ON DELETE CASCADE,
    name VARCHAR(50) NOT NULL,
    surname VARCHAR(50) NOT NULL,
    birth_date DATE NOT NULL,
```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

```

phone VARCHAR(30) NOT NULL,
education TEXT NOT NULL DEFAULT "",
experience TEXT NOT NULL DEFAULT "",
skills TEXT NOT NULL DEFAULT ""
);

```

Код створення таблиці employers (Роботодавці):

```

CREATE TABLE employers (
    id SERIAL PRIMARY KEY,
    user_id INT REFERENCES users(id) ON DELETE CASCADE,
    company_name VARCHAR(100) NOT NULL,
    staff_size INT NOT NULL,
    phone VARCHAR(30) NOT NULL,
    address VARCHAR(150) NOT NULL,
    website VARCHAR(150) NOT NULL DEFAULT ""
);

```

Код створення таблиці vacancies (Вакансії):

```

CREATE TABLE vacancies (
    id SERIAL PRIMARY KEY,
    employer_id INT REFERENCES employers(id) ON DELETE CASCADE,
    title VARCHAR(150) NOT NULL,
    salary VARCHAR(50),
    responsibilities TEXT NOT NULL,
    requirements TEXT NOT NULL DEFAULT "",
    is_remote BOOLEAN NOT NULL DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

Код створення таблиці responses (Відгуки):

```
CREATE TABLE responses (  
    id SERIAL PRIMARY KEY,  
    job_seeker_id INT REFERENCES job_seekers(id) ON DELETE CASCADE,  
    vacancy_id INT REFERENCES vacancies(id) ON DELETE CASCADE,  
    status VARCHAR(20) CHECK (status IN ('pending', 'reviewed', 'accepted',  
'rejected')) DEFAULT 'pending' NOT NULL,  
    sent_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

Код створення таблиці recommendations (Рекомендації):

```
CREATE TABLE recommendations (  
    id SERIAL PRIMARY KEY,  
    job_seeker_id INT REFERENCES job_seekers(id) ON DELETE CASCADE,  
    consultant_id INT REFERENCES users(id) ON DELETE CASCADE,  
    vacancy_id INT REFERENCES vacancies(id) ON DELETE CASCADE,  
    is_accepted BOOLEAN NOT NULL DEFAULT FALSE,  
    sent_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

Код створення таблиці messages (Повідомлення):

```
CREATE TABLE messages (  
    id SERIAL PRIMARY KEY,  
    sender_id INT REFERENCES users(id) ON DELETE CASCADE,  
    receiver_id INT REFERENCES users(id) ON DELETE CASCADE,  
    content TEXT NOT NULL,  
    sent_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

Тепер необхідно написати й виконати код створення процедур. Нижче наведено код групи процедур, призначених для зберігання нових записів втаблицях.

Код процедури для додавання нового користувача:

```
CREATE PROCEDURE add_user(  
    IN username VARCHAR(50),  
    IN password VARCHAR(100),  
    IN email VARCHAR(150),  
    IN role VARCHAR(20)  
)  
LANGUAGE plpgsql  
AS $$  
BEGIN  
    INSERT INTO users (username, password, email, role)  
    VALUES (username, password, email, role);  
END;  
$$;
```

Код процедури, що зберігає інформацію нових про претендентів:

```
CREATE PROCEDURE add_job_seeker(  
    IN user_id INT,  
    IN name VARCHAR(50),  
    IN surname VARCHAR(50),  
    IN birth_date DATE,  
    IN phone VARCHAR(30),  
    IN education TEXT DEFAULT "",  
    IN experience TEXT DEFAULT "",  
    IN skills TEXT DEFAULT ""  
)  
LANGUAGE plpgsql
```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

AS \$\$

BEGIN

IF EXISTS (SELECT 1 FROM users WHERE id = user_id AND role = 'job_seeker')
THEN

INSERT INTO job_seekers (user_id, name, surname, birth_date, phone, education,
experience, skills)

VALUES (user_id, name, surname, birth_date, phone, education, experience,
skills);

ELSE

RAISE EXCEPTION 'User with id % does not exist or is not a job seeker', user_id;
END IF;

END;

\$\$;

Код процедури, що зберігає інформацію про нових роботодавців:

CREATE PROCEDURE add_employer(

IN user_id INT,

IN company_name VARCHAR(100),

IN staff_size INT,

IN phone VARCHAR(30),

IN address VARCHAR(150),

IN website VARCHAR(150) DEFAULT "

)

LANGUAGE plpgsql

AS \$\$

BEGIN

IF EXISTS (SELECT 1 FROM users WHERE id = user_id AND role = 'employer')
THEN

INSERT INTO employers (user_id, company_name, staff_size, phone, address,
website)

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

```

VALUES (user_id, company_name, staff_size, phone, address, website);
ELSE
    RAISE EXCEPTION 'User with id % does not exist or is not an employer',
user_id;
END IF;
END;
$$;

```

Код процедури для створення роботодавцем вакансії:

```

CREATE PROCEDURE add_vacancy(
    IN employer_id INT,
    IN title VARCHAR(150),
    IN salary VARCHAR(50),
    IN responsibilities TEXT,
    IN requirements TEXT DEFAULT "",
    IN is_remote BOOLEAN DEFAULT FALSE
)
LANGUAGE plpgsql
AS $$
BEGIN
    IF EXISTS (SELECT 1 FROM employers WHERE id = employer_id) THEN
        INSERT INTO vacancies (employer_id, title, salary, responsibilities, requirements,
is_remote)
            VALUES (employer_id, title, salary, responsibilities, requirements, is_remote);
    ELSE
        RAISE EXCEPTION 'Employer with id % does not exist', employer_id;
    END IF;
END;
$$;

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

Код процедури для надсилання відгуку на вакансію:

```
CREATE PROCEDURE send_response(  
    IN job_seeker_id INT,  
    IN vacancy_id INT  
)  
LANGUAGE plpgsql  
AS $$  
BEGIN  
    IF EXISTS (SELECT 1 FROM job_seekers WHERE id = job_seeker_id) AND  
    EXISTS (SELECT 1 FROM vacancies WHERE id = vacancy_id) THEN  
        INSERT INTO responses (job_seeker_id, vacancy_id)  
        VALUES (job_seeker_id, vacancy_id);  
    ELSE  
        RAISE EXCEPTION 'Job seeker with id % or vacancy with id % does not exist',  
        job_seeker_id, vacancy_id;  
    END IF;  
END;  
$;
```

Код процедури для надсилання рекомендації:

```
CREATE PROCEDURE recommend_vacancy(  
    IN job_seeker_id INT,  
    IN consultant_id INT,  
    IN vacancy_id INT  
)  
LANGUAGE plpgsql  
AS $$  
BEGIN  
    INSERT INTO recommendations (job_seeker_id, consultant_id, vacancy_id)  
    VALUES (job_seeker_id, consultant_id, vacancy_id);
```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

END;

\$\$;

Код процедури для надсилання повідомлення:

CREATE PROCEDURE send_message(

IN sender_id INT,

IN receiver_id INT,

IN content TEXT

)

LANGUAGE plpgsql

AS \$\$

DECLARE

sender_role VARCHAR(20);

receiver_role VARCHAR(20);

BEGIN

SELECT role INTO sender_role FROM users WHERE id = sender_id;

SELECT role INTO receiver_role FROM users WHERE id = receiver_id;

IF NOT FOUND THEN

RAISE EXCEPTION 'User with id % or % does not exist', sender_id, receiver_id;

END IF;

IF (sender_role = 'job_seeker' AND receiver_role = 'job_seeker') OR

(sender_role = 'employer' AND receiver_role = 'employer') THEN

RAISE EXCEPTION 'Messages between users with the same role (job_seeker or employer) are not allowed';

END IF;

INSERT INTO messages (sender_id, receiver_id, content)

VALUES (sender_id, receiver_id, content);

END;

\$\$;

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

Далі необхідно виконати код процедур, що призначені для оновлення наявних в таблицях записів. Код процедури для оновлення даних користувачів:

```
CREATE PROCEDURE update_user(
    IN user_id INT,
    IN username VARCHAR(50),
    IN email VARCHAR(150)
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE users
    SET username = username,
        email = email
    WHERE id = user_id;
END;
$$;
```

Код процедури для оновлення даних претендентів:

```
CREATE PROCEDURE update_job_seeker(
    IN job_seeker_id INT,
    IN name VARCHAR(50),
    IN surname VARCHAR(50),
    IN birth_date DATE,
    IN phone VARCHAR(30),
    IN education TEXT,
    IN experience TEXT,
    IN skills TEXT
)
LANGUAGE plpgsql
AS $$
```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

BEGIN

UPDATE job_seekers

SET name = name,

surname = surname,

birth_date = birth_date,

phone = phone,

education = education,

experience = experience,

skills = skills

WHERE id = job_seeker_id;

END;

\$\$;

Код процедури, що оновлює дані роботодавців:

CREATE PROCEDURE update_employer(

IN employer_id INT,

IN company_name VARCHAR(100),

IN staff_size INT,

IN phone VARCHAR(30),

IN address VARCHAR(150),

IN website VARCHAR(150)

)

LANGUAGE plpgsql

AS \$\$

BEGIN

UPDATE employers

SET company_name = company_name,

staff_size = staff_size,

phone = phone,

address = address,

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

```

        website = website
    WHERE id = employer_id;
END;
$$;

```

Код процедури, що оновлює вакансії:

```

CREATE PROCEDURE update_vacancy(
    IN vacancy_id INT,
    IN title VARCHAR(150),
    IN salary VARCHAR(50),
    IN responsibilities TEXT,
    IN requirements TEXT,
    IN is_remote BOOLEAN
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE vacancies
    SET title = title,
        salary = salary,
        responsibilities = responsibilities,
        requirements = requirements,
        is_remote = is_remote
    WHERE id = vacancy_id;
END;
$$;

```

Код процедури, що оновлює відгуки:

```

CREATE PROCEDURE update_response(
    IN response_id INT,

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        IN status VARCHAR(20)
    )
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE responses
    SET status = status WHERE id = response_id;
END;
$$;

```

Код процедури для редагування рекомендацій:

```

CREATE PROCEDURE update_recommendation(
    IN recommendation_id INT,
    IN is_accepted BOOLEAN
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE recommendations
    SET is_accepted = is_accepted
    WHERE id = recommendation_id;
END;
$$;

```

Код процедури для редагування повідомлень:

```

CREATE PROCEDURE update_message(
    IN message_id INT,
    IN content TEXT
)
LANGUAGE plpgsql

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

AS \$\$

BEGIN

UPDATE messages

SET content = content WHERE id = message_id;

END;

\$\$;

Остання група процедур – це процедури, призначені для видалення записів таблиць за їх ідентифікаторами. Код процедури для видалення користувачів:

CREATE PROCEDURE delete_user(user_id INT)

LANGUAGE plpgsql

AS \$\$

BEGIN

DELETE FROM users WHERE id = user_id;

END;

\$\$;

Код процедури для видалення даних про претендентів:

CREATE PROCEDURE delete_job_seeker(job_seeker_id INT)

LANGUAGE plpgsql

AS \$\$

BEGIN

DELETE FROM job_seekers WHERE id = job_seeker_id;

END;

\$\$;

Код процедури для видалення даних про роботодавців:

CREATE PROCEDURE delete_employer(employer_id INT)

LANGUAGE plpgsql

AS \$\$

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

BEGIN

DELETE FROM employers WHERE id = employer_id;

END;

\$\$;

Код процедури для видалення вакансій:

CREATE PROCEDURE delete_vacancy(vacancy_id INT)

LANGUAGE plpgsql

AS \$\$

BEGIN

DELETE FROM vacancies WHERE id = vacancy_id;

END;

\$\$;

Код процедури для видалення відгуків:

CREATE PROCEDURE delete_response(response_id INT)

LANGUAGE plpgsql

AS \$\$

BEGIN

DELETE FROM responses WHERE id = response_id;

END;

\$\$;

Код процедури для видалення рекомендацій:

CREATE PROCEDURE delete_recommendation(recommendation_id INT)

LANGUAGE plpgsql

AS \$\$

BEGIN

DELETE FROM recommendations WHERE id = recommendation_id;

END;

\$\$;

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

Код процедури для видалення повідомлень:

```
CREATE PROCEDURE delete_message(message_id INT)
LANGUAGE plpgsql
AS $$
BEGIN
    DELETE FROM messages WHERE id = message_id;
END;
$$;
```

Також необхідно виконати код функцій та представлень, визначених при проектуванні бази даних. Код функції, що безпечно повертає дані користувача (без пароллю):

```
CREATE FUNCTION show_user(user_id INT)
RETURNS TABLE (
    id INT,
    username VARCHAR(50),
    email VARCHAR(150),
    role VARCHAR(20),
    created_at TIMESTAMP
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT id, username, email, role, created_at
    FROM users
    WHERE id = user_id;
END;
$$;
```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

Код функції для обчислення віку претендента:

```
CREATE FUNCTION calculate_age(birth_date DATE) RETURNS INT AS $$  
DECLARE  
    user_age INT;  
BEGIN  
    SELECT DATE_PART('year', AGE(birth_date)) INTO user_age;  
    RETURN user_age;  
END;  
$$ LANGUAGE plpgsql;
```

Тепер можна визначити необхідні представлення. Представлення, що надає інформацію про претендентів разом з їх віком:

```
CREATE VIEW job_seekers_with_age AS  
SELECT  
    id, user_id, name, surname, birth_date,  
    calculate_age(birth_date) AS age,  
    phone, education, experience, skills  
FROM job_seekers;
```

Представлення, що виводить інформацію про вакансії та роботодавців, що їх розмістили:

```
CREATE VIEW vacancies_employers AS  
SELECT  
    v.id AS vacancy_id, v.title, v.salary, v.responsibilities, v.requirements,  
    v.is_remote, v.created_at, v.updated_at,  
    e.id AS employer_id, e.company_name, e.phone, e.address, e.website  
FROM  
    employers e JOIN vacancies v  
ON  
    v.employer_id = e.id;
```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

Представлення, що надає розширену інформацію про відгуки, включаючи дані відповідних претендентів та вакансій:

```
CREATE VIEW responses_ext AS
SELECT
    js.id AS job_seeker_id,
    js.name AS job_seeker_name,
    js.surname AS job_seeker_surname,
    v.id AS vacancy_id, v.title AS vacancy_title,
    r.id AS response_id, r.sent_at, r.status
FROM
    job_seekers js
JOIN
    responses r ON r.job_seeker_id = js.id
JOIN
    vacancies v ON r.vacancy_id = v.id;
```

Представлення, що виводить рекомендації претендентам від співробітників агенції на відповідні вакансії:

```
CREATE VIEW recommendations_ext AS
SELECT
    r.id AS recommendation_id,
    r.is_accepted,
    r.sent_at,
    js.id AS job_seeker_id,
    js.name AS job_seeker_name,
    js.surname AS job_seeker_surname,
    u.id AS consultant_id,
    u.username AS consultant_username,
    v.id AS vacancy_id,
    v.title AS vacancy_title
```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

```

FROM
    recommendations r
JOIN
    job_seekers js ON r.job_seeker_id = js.id
JOIN
    users u ON r.consultant_id = u.id
JOIN
    vacancies v ON r.vacancy_id = v.id;

```

Представлення, що повертає повідомлення разом з їх адресантом та отримувачем:

```

CREATE VIEW messages_users AS
SELECT
    m.id AS message_id,
    m.content, m.sent_at,
    s.id AS sender_id,
    s.username AS sender,
    r.id AS receiver_id,
    r.username AS receiver
FROM
    messages m
JOIN
    users s ON m.sender_id = s.id
JOIN
    users r ON m.receiver_id = r.id;

```

Після виконання вищезазначених операцій варто переконатися, що всі об'єкти дійсно було створено. Для перегляду таблиць бази даних в `psql` можна використовувати команду `\dt` (рис. 3.10).

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

```
staffing_agency=# \dt
```

List of relations			
Schema	Name	Type	Owner
public	employers	table	postgres
public	job_seekers	table	postgres
public	messages	table	postgres
public	recommendations	table	postgres
public	responses	table	postgres
public	users	table	postgres
public	vacancies	table	postgres

(7 rows)

Рис. 3.10. Перелік таблиць бази даних

Аналогічно за допомогою команди `\dv` можна переглянути перелік створених представлень (рис. 3.11).

```
staffing_agency=# \dv
```

List of relations			
Schema	Name	Type	Owner
public	job_seekers_with_age	view	postgres
public	messages_users	view	postgres
public	recommendations_ext	view	postgres
public	responses_ext	view	postgres
public	vacancies_employers	view	postgres

(5 rows)

Рис. 3.11. Перелік представлень бази даних

Щоб переглянути перелік користувацьких збережених процедур та функцій, можна виконати наступний запит:

```
SELECT routine_schema, routine_name, routine_type
FROM information_schema.routines WHERE routine_schema = 'public';
```

```
staffing_agency=# SELECT routine_schema, routine_name, routine_type
staffing_agency=# FROM information_schema.routines
staffing_agency=# WHERE routine_schema = 'public';
```

routine_schema	routine_name	routine_type
public	calculate_age	FUNCTION
public	show_user	FUNCTION
public	add_user	PROCEDURE
public	delete_employer	PROCEDURE
public	delete_vacancy	PROCEDURE
public	delete_response	PROCEDURE
public	delete_recommendation	PROCEDURE
public	delete_message	PROCEDURE
public	update_user	PROCEDURE
public	update_job_seeker	PROCEDURE
public	update_employer	PROCEDURE
public	update_vacancy	PROCEDURE
public	update_response	PROCEDURE
public	update_recommendation	PROCEDURE
public	update_message	PROCEDURE
public	add_job_seeker	PROCEDURE
public	add_employer	PROCEDURE
public	add_vacancy	PROCEDURE
public	send_response	PROCEDURE
public	recommend_vacancy	PROCEDURE
public	send_message	PROCEDURE
public	delete_user	PROCEDURE
public	delete_job_seeker	PROCEDURE

(23 rows)

Рис. 3.12. Перелік процедур та функцій бази даних

Таким чином, було написано та виконано код функціоналу, визначеного на етапі проектування бази даних.

3.3. Реалізація системи розмежування прав доступу

Для організації розмежування прав доступу перш за все необхідно створити визначені для використання в базі даних ролі. Для цього в PostgreSQL використовується команда CREATE ROLE. При створенні ролей важливо вказати атрибут LOGIN, щоб від їх імені можна було підключатися до серверу баз даних. Створити всі необхідні ролі можна, виконавши наступні команди:

```
CREATE ROLE db_admin LOGIN PASSWORD '8hl31jb47u';
CREATE ROLE consultant LOGIN PASSWORD '58im7u09ts';
CREATE ROLE job_seeker LOGIN PASSWORD 'nbv71k8ds4';
CREATE ROLE employer LOGIN PASSWORD '764as8bga2';
CREATE ROLE guest LOGIN PASSWORD 'hf91l7bs46';
```

Щоб переконатися в успішному створенні ролей, можна переглянути їх перелік за допомогою команди \du (рис. 3.13).

staffing_agency=# \du

Role name	List of roles Attributes	Member of
consultant		{}
db_admin		{}
employer		{}
guest		{}
job_seeker		{}
postgres	Superuser, Create role, Create DB, Replication, Bypass RLS	{}

Рис. 3.13. Перелік ролей на сервері баз даних

Наступний крок – надання користувачам привілеїв, зазначених у матриці доступу. Надання привілеїв ролі db_admin:

```
-- привілеї на перегляд таблиць
GRANT SELECT ON job_seekers TO db_admin;
GRANT SELECT ON responses TO db_admin;
GRANT SELECT ON recommendations TO db_admin;
GRANT SELECT ON messages TO db_admin;
GRANT SELECT ON employers TO db_admin;
```

```

GRANT SELECT ON vacancies TO db_admin;
-- привілеї на виконання всіх процедур та функцій
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA public TO db_admin;
GRANT EXECUTE ON ALL PROCEDURES IN SCHEMA public TO db_admin;
-- привілеї на перегляд представлень
GRANT SELECT ON job_seekers_with_age TO db_admin;
GRANT SELECT ON responses_ext TO db_admin;
GRANT SELECT ON recommendations_ext TO db_admin;
GRANT SELECT ON messages_users TO db_admin;
GRANT SELECT ON vacancies_employers TO db_admin;

```

Надання привілеїв ролі consultant:

```

-- привілеї на перегляд таблиць
GRANT SELECT ON job_seekers TO consultant;
GRANT SELECT ON responses TO consultant;
GRANT SELECT ON recommendations TO consultant;
GRANT SELECT ON messages TO consultant;
GRANT SELECT ON employers TO consultant;
GRANT SELECT ON vacancies TO consultant;
-- привілеї на виконання процедур та функцій
GRANT EXECUTE ON PROCEDURE recommend_vacancy TO consultant;
GRANT EXECUTE ON PROCEDURE send_message TO consultant;
GRANT EXECUTE ON PROCEDURE update_user TO consultant;
GRANT EXECUTE ON PROCEDURE update_message TO consultant;
GRANT EXECUTE ON PROCEDURE delete_recommendation TO consultant;
GRANT EXECUTE ON PROCEDURE delete_message TO consultant;
GRANT EXECUTE ON FUNCTION show_user(INT) TO consultant;
GRANT EXECUTE ON FUNCTION calculate_age(AGE) TO consultant;
-- привілеї на перегляд представлень
GRANT SELECT ON job_seekers_with_age TO consultant;

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

```
GRANT SELECT ON responses_ext TO consultant;
GRANT SELECT ON recommendations_ext TO consultant;
GRANT SELECT ON messages_users TO consultant;
GRANT SELECT ON vacancies_employers TO consultant;
```

Надання привілеїв ролі job_seeker:

-- привілеї на перегляд таблиць

```
GRANT SELECT ON job_seekers TO job_seeker;
GRANT SELECT ON responses TO job_seeker;
GRANT SELECT ON recommendations TO job_seeker;
GRANT SELECT ON messages TO job_seeker;
GRANT SELECT ON employers TO job_seeker;
GRANT SELECT ON vacancies TO job_seeker;
```

-- привілеї на виконання процедур та функцій

```
GRANT EXECUTE ON PROCEDURE add_job_seeker TO job_seeker;
GRANT EXECUTE ON PROCEDURE send_response TO job_seeker;
GRANT EXECUTE ON PROCEDURE send_message TO job_seeker;
GRANT EXECUTE ON PROCEDURE update_user TO job_seeker;
GRANT EXECUTE ON PROCEDURE update_job_seeker TO job_seeker;
GRANT EXECUTE ON PROCEDURE update_recommendation TO job_seeker;
GRANT EXECUTE ON PROCEDURE update_message TO job_seeker;
GRANT EXECUTE ON PROCEDURE delete_job_seeker TO job_seeker;
GRANT EXECUTE ON PROCEDURE delete_response TO job_seeker;
GRANT EXECUTE ON PROCEDURE delete_message TO job_seeker;
GRANT EXECUTE ON FUNCTION show_user(INT) TO job_seeker;
```

-- привілеї на перегляд представлень

```
GRANT SELECT ON responses_ext TO job_seeker;
GRANT SELECT ON recommendations_ext TO job_seeker;
GRANT SELECT ON messages_users TO job_seeker;
GRANT SELECT ON vacancies_employers TO job_seeker;
```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

Надання привілеїв ролі employer:

-- привілеї на перегляд таблиць

GRANT SELECT ON job_seekers TO employer;

GRANT SELECT ON responses TO employer;

GRANT SELECT ON messages TO employer;

GRANT SELECT ON employers TO employer;

GRANT SELECT ON vacancies TO employer;

-- привілеї на виконання процедур та функцій

GRANT EXECUTE ON PROCEDURE add_employer TO employer;

GRANT EXECUTE ON PROCEDURE add_vacancy TO employer;

GRANT EXECUTE ON PROCEDURE send_message TO employer;

GRANT EXECUTE ON PROCEDURE update_user TO employer;

GRANT EXECUTE ON PROCEDURE update_employer TO employer;

GRANT EXECUTE ON PROCEDURE update_vacancy TO employer;

GRANT EXECUTE ON PROCEDURE update_response TO employer;

GRANT EXECUTE ON PROCEDURE update_message TO employer;

GRANT EXECUTE ON PROCEDURE delete_employer TO employer;

GRANT EXECUTE ON PROCEDURE delete_vacancy TO employer;

GRANT EXECUTE ON PROCEDURE delete_message TO employer;

GRANT EXECUTE ON FUNCTION show_user(INT) TO employer;

GRANT EXECUTE ON FUNCTION calculate_age(AGE) TO employer;

-- привілеї на перегляд представлень

GRANT SELECT ON job_seekers_with_age TO employer;

GRANT SELECT ON responses_ext TO employer;

GRANT SELECT ON messages_users TO employer;

GRANT SELECT ON vacancies_employers TO employer;

Роль guest має найменшу кількість привілеїв. Втім, це єдина роль окрім db_admin, яка має дозвіл на виклик до процедури add_user, що дозволяє неавтентифікованим користувачам зареєструватися в системі.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

Надання привілеїв ролі guest:

GRANT SELECT ON employers TO guest;

GRANT SELECT ON vacancies TO guest;

GRANT EXECUTE ON PROCEDURE add_user TO guest;

GRANT SELECT ON vacancies_employers TO guest;

Таким чином, було реалізовано політику розмежування прав доступу до об'єктів бази даних, спроектовану у розділі 2.

3.4. Налаштування резервного копіювання бази даних

Резервне копіювання є однією з найбільш важливих практик в управлінні базами даних та забезпеченні їхньої безпеки. Резервне копіювання дає змогу відновлювати дані після нештатних ситуацій, таких як помилки обладнання, збої у програмному забезпеченні, зловмисні атаки чи людські помилки. У разі втрати даних відновлення без резервної копії може вимагати значних зусиль та часу, або взагалі бути неможливим. Резервне копіювання дозволяє швидко відновити дані та продовжити бізнес-процеси, мінімізуючи простої та втрати доходів.

В PostgreSQL для створення резервних копій баз даних передбачена вбудована утиліта `pg_dump`. Суть її роботи полягає в тому, щоб згенерувати набір SQL-інструкцій, що повністю відтворює базу даних станом на момент запуску команди. Цей код потрібно зберегти, наприклад, в файлі на диску чи на віддаленому сервері. Утиліту `pg_dump` можна виконувати з будь-якого віддаленого хоста, якщо він має доступ до бази даних. Однак, для цього вона повинна мати доступ до читання всіх таблиць, для яких необхідно створити резервну копію, тому на практиці її майже завжди необхідно запускати від імені суперкористувача, в даному випадку ролі `postgres`.

Хороша реалізація резервного копіювання має відповідати принаймні наступним вимогам.

1. Назви резервних копій повинні містити дату та час їх створення
2. Файли резервних копій мають стискатися для економії дискового простору

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Застарілі резервні копії мають видалятися автоматично

Для початку необхідно створити каталог, в якому зберігатимуться резервні копії, наприклад /backups. Для цього виконаємо команду:

```
sudo mkdir /backups
```

Зробимо власником цього каталогу користувача postgres, від імені якого відбуватиметься резервне копіювання. Це можна зробити наступною командою:

```
sudo chown postgres:postgres /backups
```

Після чого змінимо обліковий запис на postgres та перейдемо до цього каталогу:

```
sudo -i -u postgres
```

```
cd /backups
```

Знаходячись в каталозі /backups, створимо в ньому каталог postgresql та файл backup_database.sh:

```
mkdir postgresql
```

```
touch backup_database.sh
```

Таким чином, каталог /backups матиме структуру, показану на рисунку 3.14.

```
postgres@server:/backups$ ls -l
total 8
-rwxr-xr-x 1 postgres postgres 846 Jun 20 08:05 backup_database.sh
drwxr-xr-x 2 postgres postgres 4096 Jun 20 08:36 postgresql
```

Рис. 3.14. Структура каталогу /backups

В файл backup_database.sh додамо наступний код:

```
#!/bin/bash
```

```
# назва бази даних
```

```
db_name=staffing_agency
```

```
# куди зберігатимуться копії
```

```
backupfolder=/backups/postgresql
```

```
# скільки днів зберігатимуться копії
```

```
keep_day=14
```

```
sqlfile=$backupfolder/$db_name-$(date +%d-%m-%Y_%H-%M).sql
```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

```
zipfile=$backupfolder/$db_name-$(date +%d-%m-%Y_%H-%M).zip
```

```
# створення резервної копії
```

```
if pg_dump $db_name > $sqlfile ; then
```

```
    echo 'Sql dump created'
```

```
else
```

```
    echo 'pg_dump return non-zero code'
```

```
    exit
```

```
fi
```

```
# стиснення резервної копії
```

```
if gzip -c $sqlfile > $zipfile; then
```

```
    echo 'The backup was successfully compressed'
```

```
else
```

```
    echo 'Error compressing backup'
```

```
    exit
```

```
fi
```

```
# видалення sql-файла копії
```

```
rm $sqlfile
```

```
# Видалення застарілих копій
```

```
find $backupfolder -mtime +$keep_day -delete
```

Налаштуємо автоматичне виконання вищезазначеного файлу, використовуючи утиліту для планування задач cron. Для цього необхідно виконати команду:

```
crontab -e
```

Після чого додамо в файл завдань, що відкриється, наступний рядок:

```
0 * * * * /backups/backup_database.sh
```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
0 * * * * /backups/backup_database.sh
```

Рис. 3.15. Файл завдань crontab користувача postgres

Даний запис означає, що програма резервного копіювання автоматично запускатиметься кожної години. На рисунку 3.16 можна побачити перелік резервних копій, створених автоматично з моменту запису задачі в crontab.

```
postgres@server:/backups/postgresql$ ls | grep "staffing"
staffing_agency-20-06-2024_08-00.zip
staffing_agency-20-06-2024_09-00.zip
staffing_agency-20-06-2024_10-00.zip
```

Рис. 3.16. Створені резервні копії бази даних

Таким чином, було налаштовано автоматичне резервне копіювання бази даних кадрової агенції, що відбувається кожну годину. Для економії дискового простору кожна резервна копія видаляється через 14 днів з моменту її створення.

Висновки до третього розділу

В ході роботи над даним розділом було встановлено та налаштовано СУБД PostgreSQL в ОС Ubuntu Server. Зокрема, змінено метод аутентифікації з однорангового, що призначений для локальних підключень, на аутентифікацію за паролем. Було створено базу даних, призначену для зберігання даних організації. За допомогою процедурної мови PL/pgSQL було створено таблиці, збережені процедури, функції й представлення, визначені на етапі проектування бази даних. Окрім вищевказаних об'єктів, з метою забезпечення надійного доступу до даних було додано нові ролі й призначено для кожної з них відповідні привілеї.

Останнім етапом була організація резервного копіювання бази даних з використанням утиліти pg_dump, планувальника завдань cron і мови сценаріїв командної оболонки bash.

ВИСНОВКИ

В ході виконання даної кваліфікаційної роботи було спроектовано та побудовано базу даних для обслуговування потреб кадрової агенції, розроблено політику розмежування прав доступу та налаштовано резервне копіювання бази даних.

Перший розділ роботи присвячено ознайомленню з реляційною моделлю даних, причинами її виникнення та перевагами реляційних баз даних. Було розглянуто розповсюджені реляційні СУБД та аспекти їх застосування, і обрано СУБД PostgreSQL для розробки бази даних, здебільшого за широку функціональність даної системи при її повній безкоштовності. Було проаналізовано сумісність СУБД PostgreSQL з найрозповсюдженішими серверними ОС, після чого в якості системи для розгортання СУБД було обрано Ubuntu Server.

Другий розділ присвячено проектуванню бази даних. На цьому етапі було визначено основні поняття, якими має оперувати база даних організації, і спроектовано таблиці для зберігання необхідних даних. Було визначено зв'язки між таблицями, що відображають взаємодію між поняттями предметної області ІС. Проектування функціоналу ІС на рівні бази даних проводилося з урахуванням таких засобів, як збережені процедури, функції, представлення. Розглянуто переваги використання збережених процедур в базах даних. Ключові з них: спрощена підтримка функціоналу бази даних, забезпечення атомарності, можливість валідації даних перед збереженням, підвищена безпека, зменшення обсягу мережевого трафіку. Останнім етапом роботи над розділом була розробка політики доступу до об'єктів бази даних. Розглянуто механізм ролей та привілеїв в PostgreSQL, і визначено ролі, необхідні для практичного застосування бази даних, та відповідні привілеї для кожної з них з метою забезпечення високого рівня безпеки даних.

Третій розділ роботи присвячено розгортанню та налаштуванню системи. Було встановлено та налаштовано СУБД PostgreSQL. За допомогою процедурної мови PL/pgSQL було створено таблиці, збережені процедури, функції й

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

представлення, визначені на етапі проектування бази даних. Було додано всі необхідні ролі бази даних, визначені в другому розділі, й призначено для кожної з них відповідні привілеї. Останнім етапом було налаштування резервного копіювання бази даних на створення копій кожну годину, їх архівування та видалення застарілих копій.

Загалом, в процесі розробки кваліфікаційної роботи було відпрацьовано проектування, побудову та налаштування бази даних на стабільну роботу та безпечний доступ до даних.

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Understanding the Relational Database Model [Електронний ресурс] – Режим доступу до ресурсу: <https://mariadb.com/kb/en/understanding-the-relational-database-model/>
2. Transactions in PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/tutorial-transactions.html>
3. DB-Engines Ranking [Електронний ресурс] – Режим доступу до ресурсу: <https://db-engines.com/en/ranking>
4. What is SQL Server? [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver16>
5. PostgreSQL: About [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/about/>
6. PostgreSQL: License [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/about/licence/>
7. PostgreSQL Data Types [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/datatype.html>
8. Supported Platforms [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/supported-platforms.html>
9. Server Setup and Operation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/runtime.html>
10. Ubuntu Server documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://ubuntu.com/server/docs>
11. PL/pgSQL — SQL Procedural Language [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/plpgsql.html>
12. PostgreSQL User-Defined Procedures [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/xproc.html>
13. PostgreSQL Functions and Operators [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/functions.html>

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

14. PostgreSQL User-Defined Functions [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/xfunc.html>
15. PostgreSQL Constraints [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/ddl-constraints.html>
16. Views in PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/sql-createview.html>
17. Database Roles [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/user-manag.html>
18. PostgreSQL Privileges [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/ddl-priv.html>
19. PostgreSQL Indexes [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/indexes.html>
20. PostgreSQL Backup and Restore [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/backup.html>
21. PostgreSQL Client Applications [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/reference-client.html>
22. PostgreSQL: define access privileges [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/sql-grant.html>
23. PostgreSQL: define default access privileges [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/sql-alterdefaultprivileges.html>
24. Joins Between Tables [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/tutorial-join.html>
25. Table Expressions [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/queries-table-expressions.html>
26. Controlling the Planner with Explicit JOIN Clauses [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/current/explicit-joins.html>

ДОДАТКИ

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг SQL-запитів

-- створення ролей

```
CREATE ROLE db_admin LOGIN PASSWORD '8hl31jb47u';
CREATE ROLE consultant LOGIN PASSWORD '58im7u09ts';
CREATE ROLE job_seeker LOGIN PASSWORD 'nbv71k8ds4';
CREATE ROLE employer LOGIN PASSWORD '764as8bga2';
CREATE ROLE guest LOGIN PASSWORD 'hf91l7bs46';
```

-- дозволи за замовчуванням

```
ALTER DEFAULT PRIVILEGES IN SCHEMA public REVOKE ALL PRIVILEGES ON TABLES
FROM PUBLIC;
ALTER DEFAULT PRIVILEGES IN SCHEMA public REVOKE EXECUTE ON FUNCTIONS
FROM PUBLIC;
```

-- таблиця Користувачі

```
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  username VARCHAR(50) UNIQUE NOT NULL,
  password VARCHAR(100) NOT NULL,
  email VARCHAR(150) UNIQUE NOT NULL,
  role VARCHAR(20) CHECK (role IN ('db_admin', 'consultant', 'employer', 'job_seeker')) NOT
  NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL
);
```

-- таблиця Претенденти

```
CREATE TABLE job_seekers (
  id SERIAL PRIMARY KEY,
  user_id INT REFERENCES users(id) ON DELETE CASCADE,
  name VARCHAR(50) NOT NULL,
  surname VARCHAR(50) NOT NULL,
  birth_date DATE NOT NULL,
  phone VARCHAR(30) NOT NULL,
  education TEXT NOT NULL DEFAULT "",
  experience TEXT NOT NULL DEFAULT "",
  skills TEXT NOT NULL DEFAULT ""
);
```

-- таблиця Роботодавці

```
CREATE TABLE employers (
  id SERIAL PRIMARY KEY,
  user_id INT REFERENCES users(id) ON DELETE CASCADE,
  company_name VARCHAR(100) NOT NULL,
  staff_size INT NOT NULL,
  phone VARCHAR(30) NOT NULL,
  address VARCHAR(150) NOT NULL,
  website VARCHAR(150) NOT NULL DEFAULT ""
);
```

-- таблиця Вакансії

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

CREATE TABLE vacancies (
    id SERIAL PRIMARY KEY,
    employer_id INT REFERENCES employers(id) ON DELETE CASCADE,
    title VARCHAR(150) NOT NULL,
    salary VARCHAR(50),
    responsibilities TEXT NOT NULL,
    requirements TEXT NOT NULL DEFAULT "",
    is_remote BOOLEAN NOT NULL DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- таблиця Відгуки
CREATE TABLE responses (
    id SERIAL PRIMARY KEY,
    job_seeker_id INT REFERENCES job_seekers(id) ON DELETE CASCADE,
    vacancy_id INT REFERENCES vacancies(id) ON DELETE CASCADE,
    status VARCHAR(20) CHECK (status IN ('pending', 'reviewed', 'accepted', 'rejected')) DEFAULT 'pending' NOT NULL,
    sent_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- таблиця Рекомендації
CREATE TABLE recommendations (
    id SERIAL PRIMARY KEY,
    job_seeker_id INT REFERENCES job_seekers(id) ON DELETE CASCADE,
    consultant_id INT REFERENCES users(id) ON DELETE CASCADE,
    vacancy_id INT REFERENCES vacancies(id) ON DELETE CASCADE,
    is_accepted BOOLEAN NOT NULL DEFAULT FALSE,
    sent_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- таблиця Повідомлення
CREATE TABLE messages (
    id SERIAL PRIMARY KEY,
    sender_id INT REFERENCES users(id) ON DELETE CASCADE,
    receiver_id INT REFERENCES users(id) ON DELETE CASCADE,
    content TEXT NOT NULL,
    sent_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Функція, обчислює вік претендентів
CREATE FUNCTION calculate_age(birth_date DATE) RETURNS INT AS $$
DECLARE
    user_age INT;
BEGIN
    SELECT DATE_PART('year', AGE(birth_date)) INTO user_age;
    RETURN user_age;
END;
$$ LANGUAGE plpgsql;

-- Функція, що надає інформацію про користувача

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

CREATE FUNCTION show_user(user_id INT)
RETURNS TABLE (
    id INT,
    username VARCHAR(50),
    email VARCHAR(150),
    role VARCHAR(20),
    created_at TIMESTAMP
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT id, username, email, role, created_at
    FROM users
    WHERE id = user_id;
END;
$$;

```

```

-- додати користувача
CREATE PROCEDURE add_user(
    IN username VARCHAR(50),
    IN password VARCHAR(100),
    IN email VARCHAR(150),
    IN role VARCHAR(20)
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO users (username, password, email, role)
    VALUES (username, password, email, role);
END;
$$;

```

```

-- додати претендента
CREATE PROCEDURE add_job_seeker(
    IN user_id INT,
    IN name VARCHAR(50),
    IN surname VARCHAR(50),
    IN birth_date DATE,
    IN phone VARCHAR(30),
    IN education TEXT DEFAULT "",
    IN experience TEXT DEFAULT "",
    IN skills TEXT DEFAULT ""
)
LANGUAGE plpgsql
AS $$
BEGIN
    IF EXISTS (SELECT 1 FROM users WHERE id = user_id AND role = 'job_seeker') THEN
        INSERT INTO job_seekers (user_id, name, surname, birth_date, phone, education, experience,
skills)
        VALUES (user_id, name, surname, birth_date, phone, education, experience, skills);
    ELSE

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        RAISE EXCEPTION 'User with id % does not exist or is not a job seeker', user_id;
    END IF;
END;
$$;

-- додати роботодавця
CREATE PROCEDURE add_employer(
    IN user_id INT,
    IN company_name VARCHAR(100),
    IN staff_size INT,
    IN phone VARCHAR(30),
    IN address VARCHAR(150),
    IN website VARCHAR(150) DEFAULT ''
)
LANGUAGE plpgsql
AS $$
BEGIN
    IF EXISTS (SELECT 1 FROM users WHERE id = user_id AND role = 'employer') THEN
        INSERT INTO employers (user_id, company_name, staff_size, phone, address, website)
        VALUES (user_id, company_name, staff_size, phone, address, website);
    ELSE
        RAISE EXCEPTION 'User with id % does not exist or is not an employer', user_id;
    END IF;
END;
$$;

-- додати вакансію
CREATE PROCEDURE add_vacancy(
    IN employer_id INT,
    IN title VARCHAR(150),
    IN salary VARCHAR(50),
    IN responsibilities TEXT,
    IN requirements TEXT DEFAULT '',
    IN is_remote BOOLEAN DEFAULT FALSE
)
LANGUAGE plpgsql
AS $$
BEGIN
    IF EXISTS (SELECT 1 FROM employers WHERE id = employer_id) THEN
        INSERT INTO vacancies (employer_id, title, salary, responsibilities, requirements, is_remote)
        VALUES (employer_id, title, salary, responsibilities, requirements, is_remote);
    ELSE
        RAISE EXCEPTION 'Employer with id % does not exist', employer_id;
    END IF;
END;
$$;

-- надіслати відгук
CREATE PROCEDURE send_response(
    IN job_seeker_id INT,
    IN vacancy_id INT
)

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

LANGUAGE plpgsql
AS $$
BEGIN
    IF EXISTS (SELECT 1 FROM job_seekers WHERE id = job_seeker_id) AND EXISTS (SELECT
1 FROM vacancies WHERE id = vacancy_id) THEN
        INSERT INTO responses (job_seeker_id, vacancy_id)
        VALUES (job_seeker_id, vacancy_id);
    ELSE
        RAISE EXCEPTION 'Job seeker with id % or vacancy with id % does not exist', job_seeker_id,
vacancy_id;
    END IF;
END;
$$;

```

-- надіслати рекомендацію

```

CREATE PROCEDURE recommend_vacancy(
    IN job_seeker_id INT,
    IN consultant_id INT,
    IN vacancy_id INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO recommendations (job_seeker_id, consultant_id, vacancy_id)
    VALUES (job_seeker_id, consultant_id, vacancy_id);
END;
$$;

```

-- надіслати повідомлення

```

CREATE PROCEDURE send_message(
    IN sender_id INT,
    IN receiver_id INT,
    IN content TEXT
)
LANGUAGE plpgsql
AS $$
DECLARE
    sender_role VARCHAR(20);
    receiver_role VARCHAR(20);
BEGIN
    SELECT role INTO sender_role FROM users WHERE id = sender_id;
    SELECT role INTO receiver_role FROM users WHERE id = receiver_id;

    IF NOT FOUND THEN
        RAISE EXCEPTION 'User with id % or % does not exist', sender_id, receiver_id;
    END IF;

    IF (sender_role = 'job_seeker' AND receiver_role = 'job_seeker') OR
(sender_role = 'employer' AND receiver_role = 'employer') THEN
        RAISE EXCEPTION 'Messages between users with the same role (job_seeker or employer) are
not allowed';
    END IF;

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

INSERT INTO messages (sender_id, receiver_id, content)
VALUES (sender_id, receiver_id, content);
END;
$$;

```

```

-- видалення користувача
CREATE PROCEDURE delete_user(user_id INT)
LANGUAGE plpgsql
AS $$
BEGIN
    DELETE FROM users WHERE id = user_id;
END;
$$;

```

```

-- видалення інформації про претендента
CREATE PROCEDURE delete_job_seeker(job_seeker_id INT)
LANGUAGE plpgsql
AS $$
BEGIN
    DELETE FROM job_seekers WHERE id = job_seeker_id;
END;
$$;

```

```

-- видалення інформації про роботодавця
CREATE PROCEDURE delete_employer(employer_id INT)
LANGUAGE plpgsql
AS $$
BEGIN
    DELETE FROM employers WHERE id = employer_id;
END;
$$;

```

```

-- видалення вакансії
CREATE PROCEDURE delete_vacancy(vacancy_id INT)
LANGUAGE plpgsql
AS $$
BEGIN
    DELETE FROM vacancies WHERE id = vacancy_id;
END;
$$;

```

```

-- видалення відгуку
CREATE PROCEDURE delete_response(response_id INT)
LANGUAGE plpgsql
AS $$
BEGIN
    DELETE FROM responses WHERE id = response_id;
END;
$$;

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

```
-- видалення рекомендації
CREATE PROCEDURE delete_recommendation(recommendation_id INT)
LANGUAGE plpgsql
AS $$
BEGIN
    DELETE FROM recommendations WHERE id = recommendation_id;
END;
$$;
```

```
-- видалення повідомлення
CREATE PROCEDURE delete_message(message_id INT)
LANGUAGE plpgsql
AS $$
BEGIN
    DELETE FROM messages WHERE id = message_id;
END;
$$;
```

```
-- оновлення даних користувачів
CREATE PROCEDURE update_user(
    IN user_id INT,
    IN username VARCHAR(50),
    IN email VARCHAR(150)
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE users
    SET username = username,
        email = email
    WHERE id = user_id;
END;
$$;
```

```
-- оновлення даних претендентів
CREATE PROCEDURE update_job_seeker(
    IN job_seeker_id INT,
    IN name VARCHAR(50),
    IN surname VARCHAR(50),
    IN birth_date DATE,
    IN phone VARCHAR(30),
    IN education TEXT,
    IN experience TEXT,
    IN skills TEXT
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE job_seekers
    SET name = name,
        surname = surname,
        birth_date = birth_date,
```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

phone = phone,
education = education,
experience = experience,
skills = skills
WHERE id = job_seeker_id;
END;
$$;

-- оновлення даних роботодавців
CREATE PROCEDURE update_employer(
    IN employer_id INT,
    IN company_name VARCHAR(100),
    IN staff_size INT,
    IN phone VARCHAR(30),
    IN address VARCHAR(150),
    IN website VARCHAR(150)
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE employers
    SET company_name = company_name,
        staff_size = staff_size,
        phone = phone,
        address = address,
        website = website
    WHERE id = employer_id;
END;
$$;

-- оновлення вакансій
CREATE PROCEDURE update_vacancy(
    IN vacancy_id INT,
    IN title VARCHAR(150),
    IN salary VARCHAR(50),
    IN responsibilities TEXT,
    IN requirements TEXT,
    IN is_remote BOOLEAN
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE vacancies
    SET title = title,
        salary = salary,
        responsibilities = responsibilities,
        requirements = requirements,
        is_remote = is_remote
    WHERE id = vacancy_id;
END;
$$;

-- оновлення відгуків

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

CREATE PROCEDURE update_response(
    IN response_id INT,
    IN status VARCHAR(20)
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE responses
    SET status = status WHERE id = response_id;
END;
$$;

-- оновлення рекомендацій
CREATE PROCEDURE update_recommendation(
    IN recommendation_id INT,
    IN is_accepted BOOLEAN
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE recommendations
    SET is_accepted = is_accepted
    WHERE id = recommendation_id;
END;
$$;

-- редагування повідомлень
CREATE PROCEDURE update_message(
    IN message_id INT,
    IN content TEXT
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE messages
    SET content = content WHERE id = message_id;
END;
$$;

CREATE VIEW job_seekers_with_age AS
SELECT
    id, user_id, name, surname, birth_date,
    calculate_age(birth_date) AS age,
    phone, education, experience, skills
FROM job_seekers;

CREATE VIEW vacancies_employers AS
SELECT
    v.id AS vacancy_id, v.title, v.salary, v.responsibilities,
    v.requirements, v.is_remote, v.created_at, v.updated_at,
    e.id AS employer_id, e.company_name, e.phone, e.address, e.website
FROM

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

employers e JOIN vacancies v
ON
    v.employer_id = e.id;

CREATE VIEW responses_ext AS
SELECT
    js.id AS job_seeker_id,
    js.name AS job_seeker_name,
    js.surname AS job_seeker_surname,
    v.id AS vacancy_id, v.title AS vacancy_title,
    r.id AS response_id, r.sent_at, r.status
FROM
    job_seekers js
JOIN
    responses r ON r.job_seeker_id = js.id
JOIN
    vacancies v ON r.vacancy_id = v.id;

CREATE VIEW recommendations_ext AS
SELECT
    r.id AS recommendation_id,
    r.is_accepted,
    r.sent_at,
    js.id AS job_seeker_id,
    js.name AS job_seeker_name,
    js.surname AS job_seeker_surname,
    u.id AS consultant_id,
    u.username AS consultant_username,
    v.id AS vacancy_id,
    v.title AS vacancy_title
FROM
    recommendations r
JOIN
    job_seekers js ON r.job_seeker_id = js.id
JOIN
    users u ON r.consultant_id = u.id
JOIN
    vacancies v ON r.vacancy_id = v.id;

CREATE VIEW messages_users AS
SELECT
    m.id AS message_id, m.content, m.sent_at,
    s.id AS sender_id, s.username AS sender,
    r.id AS receiver_id, r.username AS receiver
FROM
    messages m
JOIN
    users s ON m.sender_id = s.id
JOIN
    users r ON m.receiver_id = r.id;

-- привілеї користувачів категорії db_admin

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

-- надання привілеїв на перегляд таблиць
GRANT SELECT ON job_seekers TO db_admin;
GRANT SELECT ON responses TO db_admin;
GRANT SELECT ON recommendations TO db_admin;
GRANT SELECT ON messages TO db_admin;
GRANT SELECT ON employers TO db_admin;
GRANT SELECT ON vacancies TO db_admin;
-- на виконання всіх процедур та функцій
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA public TO db_admin;
GRANT EXECUTE ON ALL PROCEDURES IN SCHEMA public TO db_admin;
-- на перегляд представлень
GRANT SELECT ON job_seekers_with_age TO db_admin;
GRANT SELECT ON responses_ext TO db_admin;
GRANT SELECT ON recommendations_ext TO db_admin;
GRANT SELECT ON messages_users TO db_admin;
GRANT SELECT ON vacancies_employers TO db_admin;

-- привілеї користувачів категорії consultant
-- таблиці
GRANT SELECT ON job_seekers TO consultant;
GRANT SELECT ON responses TO consultant;
GRANT SELECT ON recommendations TO consultant;
GRANT SELECT ON messages TO consultant;
GRANT SELECT ON employers TO consultant;
GRANT SELECT ON vacancies TO consultant;
-- функції, процедури
GRANT EXECUTE ON PROCEDURE recommend_vacancy TO consultant;
GRANT EXECUTE ON PROCEDURE send_message TO consultant;
GRANT EXECUTE ON PROCEDURE update_user TO consultant;
GRANT EXECUTE ON PROCEDURE update_message TO consultant;
GRANT EXECUTE ON PROCEDURE delete_recommendation TO consultant;
GRANT EXECUTE ON PROCEDURE delete_message TO consultant;
GRANT EXECUTE ON FUNCTION show_user(INT) TO consultant;
GRANT EXECUTE ON FUNCTION calculate_age(AGE) TO consultant;
-- представлення
GRANT SELECT ON job_seekers_with_age TO consultant;
GRANT SELECT ON responses_ext TO consultant;
GRANT SELECT ON recommendations_ext TO consultant;
GRANT SELECT ON messages_users TO consultant;
GRANT SELECT ON vacancies_employers TO consultant;

-- привілеї користувачів категорії job_seeker
-- таблиці
GRANT SELECT ON job_seekers TO job_seeker;
GRANT SELECT ON responses TO job_seeker;
GRANT SELECT ON recommendations TO job_seeker;
GRANT SELECT ON messages TO job_seeker;
GRANT SELECT ON employers TO job_seeker;
GRANT SELECT ON vacancies TO job_seeker;
-- функції, процедури
GRANT EXECUTE ON PROCEDURE add_job_seeker TO job_seeker;
GRANT EXECUTE ON PROCEDURE send_response TO job_seeker;

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```

GRANT EXECUTE ON PROCEDURE send_message TO job_seeker;
GRANT EXECUTE ON PROCEDURE update_user TO job_seeker;
GRANT EXECUTE ON PROCEDURE update_job_seeker TO job_seeker;
GRANT EXECUTE ON PROCEDURE update_recommendation TO job_seeker;
GRANT EXECUTE ON PROCEDURE update_message TO job_seeker;
GRANT EXECUTE ON PROCEDURE delete_job_seeker TO job_seeker;
GRANT EXECUTE ON PROCEDURE delete_response TO job_seeker;
GRANT EXECUTE ON PROCEDURE delete_message TO job_seeker;
GRANT EXECUTE ON FUNCTION show_user(INT) TO job_seeker;
-- представлення
GRANT SELECT ON responses_ext TO job_seeker;
GRANT SELECT ON recommendations_ext TO job_seeker;
GRANT SELECT ON messages_users TO job_seeker;
GRANT SELECT ON vacancies_employers TO job_seeker;

-- привілеї користувачів категорії employer
-- таблиці
GRANT SELECT ON job_seekers TO employer;
GRANT SELECT ON responses TO employer;
GRANT SELECT ON messages TO employer;
GRANT SELECT ON employers TO employer;
GRANT SELECT ON vacancies TO employer;
-- функції, процедури
GRANT EXECUTE ON PROCEDURE add_employer TO employer;
GRANT EXECUTE ON PROCEDURE add_vacancy TO employer;
GRANT EXECUTE ON PROCEDURE send_message TO employer;
GRANT EXECUTE ON PROCEDURE update_user TO employer;
GRANT EXECUTE ON PROCEDURE update_employer TO employer;
GRANT EXECUTE ON PROCEDURE update_vacancy TO employer;
GRANT EXECUTE ON PROCEDURE update_response TO employer;
GRANT EXECUTE ON PROCEDURE update_message TO employer;
GRANT EXECUTE ON PROCEDURE delete_employer TO employer;
GRANT EXECUTE ON PROCEDURE delete_vacancy TO employer;
GRANT EXECUTE ON PROCEDURE delete_message TO employer;
GRANT EXECUTE ON FUNCTION show_user(INT) TO employer;
GRANT EXECUTE ON FUNCTION calculate_age(AGE) TO employer;
-- представлення
GRANT SELECT ON job_seekers_with_age TO employer;
GRANT SELECT ON responses_ext TO employer;
GRANT SELECT ON messages_users TO employer;
GRANT SELECT ON vacancies_employers TO employer;

-- привілеї користувачів категорії guest
GRANT SELECT ON employers TO guest;
GRANT SELECT ON vacancies TO guest;
GRANT EXECUTE ON PROCEDURE add_user TO guest;
GRANT SELECT ON vacancies_employers TO guest;

```

		Генсіровський Т.В.			ККІКБ.КР.Б – 123 – 04 – ПЗ	Арк.
		Головня О.С.				80
Змн.	Арк.	№ докум.	Підпис	Дата		