Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Організація баз даних та знань»

Розробити базу даних для інформаційної підтримки діяльності адвокатської контори

Студента __2__ курсу AI-243 групи
Спеціальності 122 – «Комп'ютерні науки»
_____Гаврилов О. В._____
(прізвище та ініціали)
Керівник доцент, PhD Іванов О.В.
_____
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала
_____
Кількість балів: _____
Оцінка: ECTS _____

Члени комісії

_____ (підпис)	_____ (прізвище та ініціали)
_____ (підпис)	_____ (прізвище та ініціали)
_____ (підпис)	_____ (прізвище та ініціали)

Одеса – 2025

АНОТАЦІЯ

Гаврилов О. В. Розробка бази даних для інформаційної підтримки діяльності адвокатської контори: курсова робота з дисципліни «Організація баз даних та знань» за спеціальністю «122 Комп’ютерні науки» / Олександр В’ячеславович Гаврилов; керівник Олексій Володимирович Іванов. – Одеса : Нац. ун-т «Одес. політехніка», 2025. – 63 с.

Курсова робота містить основну текстову частину на 60 сторінках, список використаних джерел з 3 найменувань на 1 сторінці.

Курсова робота присвячена розробці реляційної бази даних для автоматизації діяльності адвокатської контори, зокрема обліку адвокатів, клієнтів (фізичних та юридичних осіб), ведення судових справ, фіксації судових засідань, документообігу та контролю фінансових платежів (гонорарів). У роботі проаналізовано предметну область, визначено основні сутності, їх атрибути й зв’язки, сформовано функціональні вимоги та user story для основних ролей користувачів системи. На основі проведеного аналізу спроєктовано концептуальну, логічну та фізичну моделі бази даних, реалізовані у СУБД PostgreSQL із використанням доменів, зовнішніх ключів, обмежень цілісності, представлень, тригерів, збережених функцій і процедур. Також реалізовано систему розмежування прав доступу для ролей адвоката та клієнта.

Ключові слова: PostgreSQL, база даних адвокатської контори, нормалізація, тригери, збережені функції, управління доступом, SQL.

ABSTRACT

Havrylov O. V. Development of a database for information support of a law firm's activities: course work on the discipline “Organization of data and knowledge bases” in the specialty “122 Computer sciences” / Oleksandr V’yacheslavovych Havrylov; supervisor [Name Surname of Supervisor]. – Odesa : Odesa Polytech. Nat. Univ., 2025. – 63 p.

The course work contains the main text part on 60 pages, a list of used sources with 3 names on 1 page.

This course work is devoted to the development of a relational database that supports the main business processes of a law firm, including accounting of lawyers, clients (individuals and legal entities), management of legal cases, recording of court hearings, document flow, and control of financial payments (fees). The paper analyses the problem domain, defines the main entities with their attributes and relationships, and formulates the functional requirements together with user stories for the key system roles. Based on this analysis, conceptual, logical and physical data models are designed and implemented in the PostgreSQL DBMS using domains, foreign keys, integrity constraints, views, triggers, stored functions and procedures. A system of access rights differentiation for lawyer and client roles has also been implemented.

Keywords: PostgreSQL, law firm database, normalization, triggers, stored functions, access control, SQL.

ЗМІСТ

Національний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра інформаційних систем

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ

студенту Гаврилову Олександру В'ячеславовичу група АІ-243

1. Тема роботи «Розробити базу даних для інформаційної підтримки діяльності адвокатської контори»

2. Термін здачі студентом закінченої роботи 28.11.2025

3. Початкові дані до проекту (роботи) Варіант 22: Адвокати: ПІБ, дата народження, адреса, телефон, освіта. Клієнти: ПІБ, дата народження, адреса, телефон (фіз./юр. особи). Справи: Номер, клієнт, адвокат, вид (стаття, кодекс, термін min/max), гонорар, стан, наслідок, отриманий термін. Вихідні дані: Поточні клієнти, незавершені справи, ефективність справ, історія захисту, чек на оплату. Функціонал: Ведення списків (адвокатів, клієнтів, архів). Розрахунок гонорарів. Перегляд звітів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити). Вивчити теорію БД, СУБД, SQL та нормалізації. Проаналізувати адвокатську контору, визначити потреби користувачів та сформувати вимоги до системи. Виділити сутності («Адвокат», «Клієнт», «Справа»), їхні атрибути та зв'язки. Обґрунтувати застосування однозв'язного списку. Побудувати ER- та UML-діаграми, провести нормалізацію. Реалізувати та заповнити БД засобами SQL. Оформити пояснювальну записку.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень). Діаграма класів предметної області. ER-діаграма в нотації Crow's Foot.

Завдання видано 11.09.2025 _____
(підпис викладача)

Завдання прийнято до виконання 11.09.2025 _____
(підпис студента)

ВСТУП

У сучасних умовах цифровізації бізнес-процесів ефективне управління інформацією стає ключовим фактором успіху для будь-якої організації, і юридична сфера не є винятком. Діяльність адвокатської контори пов’язана з обробкою значних обсягів даних: веденням картотеки клієнтів, відстеженням статусу судових справ, контролем термінів подання документів та фіксацією фінансових взаєморозрахунків. Враховуючи високу відповідальність юридичної роботи та жорсткі процесуальні строки, відсутність надійної інформаційної системи може призвести до критичних помилок та репутаційних втрат.

Створення спеціалізованої бази даних для адвокатської контори є актуальним завданням, оскільки традиційні методи ведення справ (паперові архіви, електронні таблиці Excel або розрізнені файли) часто не забезпечують належного рівня структурованості та безпеки даних. Такий підхід ускладнює пошук необхідної інформації про хід справи, підвищує ризик пропуску судових засідань та створює труднощі при формуванні звітності щодо ефективності роботи адвокатів. Добре спроєктована реляційна база даних дозволяє уникнути дублювання інформації, забезпечити її цілісність та надати миттєвий доступ до історії взаємовідносин з клієнтами.

Розроблена в межах даної курсової роботи база даних дає можливість ефективно автоматизувати основні аспекти діяльності адвокатської контори. Зокрема, система дозволяє зберігати деталізовані дані про адвокатів та їх спеціалізацію, вести облік клієнтів (як фізичних, так і юридичних осіб), відстежувати етапи розгляду судових справ різних типів (кримінальні, адміністративні тощо), фіксувати результати судових засідань та контролювати надходження оплат за надані послуги. Наявність логічних зв’язків, первинних та зовнішніх ключів, а також розроблених тригерів забезпечує коректність даних та автоматизацію бізнес-правил, наприклад, закриття справ або розрахунок гонорарів.

Впровадження такої бази даних значно полегшує роботу адміністративного персоналу та самих адвокатів, мінімізує ризики "людського фактору",

пришвидшує доступ до документів та дозволяє проводити аналіз ефективності ведення справ. Для керівництва контори це означає прозорість фінансових потоків та можливість оперативного отримання аналітичної інформації для прийняття управлінських рішень.

Мета даної курсової роботи – проаналізувати предметну область діяльності адвокатської контори, спроєктувати концептуальну та логічну моделі даних, а також реалізувати фізичну модель у СУБД PostgreSQL із використанням сучасних механізмів маніпулювання даними, збережених процедур, функцій, тригерів та засобів захисту інформації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

У цьому розділі проаналізовано діяльність адвокатської контори як організації, що надає професійні юридичні послуги, здійснює представництво інтересів клієнтів у судах та забезпечує правовий супровід. Виділено ключові бізнес-процеси, які потребують автоматизації (такі як реєстрація нових клієнтів, ведення обліку судових справ, моніторинг розкладу засідань та контроль фінансових розрахунків), та окреслено основні інформаційні потоки підприємства. На основі детального аналізу предметної області та потреб користувачів сформовано функціональні вимоги до інформаційної системи. Сформульовано постановку задачі, яка визначає необхідні сутності, їх атрибути, логічні зв’язки та функції майбутньої бази даних.

1.1 Опис предметної області, для якої створюється база даних

Предметною областю даного проєкту є діяльність адвокатської контори – організації, що надає професійні юридичні послуги та здійснює представництво інтересів громадян і компаній у судах. У процесі її роботи формується значний обсяг інформації, пов’язаної з клієнтами (фізичними та юридичними особами), адвокатами, судовими справами, типами проваджень, розкладом судових засідань, документообігом та фінансовими розрахунками (гонорарами). Вся ця інформація є динамічною та тісно взаємопов’язаною, а тому потребує надійної системи зберігання та швидкого доступу, адже у юридичній сфері дотримання термінів та точність даних є критичними факторами.

Нині в багатьох юридичних фірмах дані ведуться у вигляді паперових досьє, розрізнених папок з документами або неузгоджених електронних таблиць Excel. Така організація призводить до численних проблем у роботі персоналу: ускладнюється пошук історії конкретної справи, зростає ризик втрати важливих документів або пропуску дати судового засідання, виникають труднощі з

контролем оплат від клієнтів та розрахунком ефективності роботи адвокатів. Крім того, стає неможливо оперативно отримати повний профіль клієнта з історією всіх його звернень, що негативно впливає на якість обслуговування та адміністративні процеси.

Для вирішення цих проблем необхідним є створення централізованої бази даних, яка дозволить забезпечити цілісність, узгодженість та структурованість інформації. Така база даних дасть змогу значно спростити облік клієнтів та адвокатів, забезпечити прозорий моніторинг судових справ від моменту подання позову до винесення вироку та пришвидшити формування фінансової звітності. Застосування сучасної СУБД, зокрема PostgreSQL, гарантує надійність зберігання конфіденційних юридичних даних та забезпечує механізми для захисту від помилок дублювання інформації.

Спроєктована система повинна охоплювати всі ключові інформаційні елементи діяльності адвокатської контори: реєстр адвокатів та їхню спеціалізацію, повну базу даних клієнтів, класифікацію типів судових справ (кримінальні, цивільні, адміністративні) з відповідними статтями та кодексами, деталі судових засідань (дата, суддя, результат), архів документів, а також історію платежів і розрахунок гонорарів. Окреме значення має зберігання зв’язків між цими сутностями, оскільки одна справа може мати багато засідань та документів, а один клієнт може бути фігурантом кількох справ.

Реалізація автоматизованої інформаційної системи на основі розробленої бази даних істотно полегшує діяльність персоналу контори. Адміністратори отримують можливість швидко розподіляти нові справи між адвокатами, контролювати навантаження та фінансові надходження. Адвокати мають зручний доступ до розкладу своїх засідань, матеріалів справ та контактів клієнтів, що дозволяє їм зосередитися на правовому захисті, а не на паперовій рутині.

Таким чином, створення бази даних для адвокатської контори є необхідним кроком для оптимізації роботи юридичної фірми, підвищення якості надання послуг, мінімізації ризиків та забезпечення надійного зберігання важливої правової інформації. Розроблена структура дозволяє вирішити основні проблеми

предметної області та забезпечує фундамент для цифрової трансформації бізнес-процесів компанії.

1.2 Опис користувачів системи та їх історії (User Story)

У межах інформаційної системи адвокатської контори можна виділити кілька основних категорій користувачів, кожна з яких має власні цілі та потреби у взаємодії з даними. Для кожного типу користувачів визначено набори User Story – короткі сценарії, що описують очікувану поведінку та необхідний функціонал.

Роль: Клієнт. Клієнт є замовником юридичних послуг, тому інформаційна система повинна забезпечувати йому прозорий доступ до ходу ведення його справ та фінансової інформації:

1) як клієнт, я хочу зареєструватися в системі та створити свій профіль, щоб мати можливість замовити юридичні послуги;

2) як клієнт, я хочу переглядати статус моїх поточних судових справ, щоб знати, на якому етапі знаходиться розгляд (відкрито, в процесі, закрито);

3) як клієнт, я хочу бачити розклад призначених судових засідань по моїй справі, щоб планувати свій час та бути присутнім за необхідності;

4) як клієнт, я хочу отримувати інформацію про результати завершених слухань, щоб розуміти проміжні рішення суду;

5) як клієнт, я хочу переглядати історію своїх платежів та сформовані чеки, щоб контролювати стан розрахунків за послуги адвоката;

6) як клієнт, я хочу мати доступ до контактних даних мого адвоката, щоб мати можливість оперативно зв’язатися з ним.

Роль: Адвокат. Адвокат є основним виконавцем робіт, який взаємодіє з даними щодо судових справ, засідань та документів:

1) як адвокат, я хочу переглядати список активних справ, у яких я призначений захисником, щоб ефективно планувати свою роботу;

2) як адвокат, я хочу мати доступ до детальної інформації про клієнта та обставини справи, щоб підготувати якісну лінію захисту;

3) як адвокат, я хочу переглядати свій особистий графік судових засідань, щоб уникнути конфліктів у розкладі та вчасно з’являтися в суді;

4) як адвокат, я хочу вносити результати судових засідань та додавати нотатки до справи, щоб фіксувати хід процесу в системі;

5) як адвокат, я хочу завантажувати та переглядати документи, що стосуються справи, щоб мати централізований архів доказів та процесуальних актів;

6) як адвокат, я хочу відстежувати свою фінансову ефективність (суму гонорарів), щоб оцінювати результативність своєї діяльності.

Роль: Адміністратор. Адміністратор відповідає за загальну організацію роботи контори, розподіл навантаження, керування довідниками та фінансовий контроль:

1) як адміністратор, я хочу реєструвати нові судові справи в базі даних та визначати їх тип (кримінальна, цивільна тощо), щоб розпочати процес провадження;

2) як адміністратор, я хочу призначати адвокатів на конкретні справи, враховуючи їхню спеціалізацію та завантаженість;

3) як адміністратор, я хочу керувати обліковими записами користувачів (адвокатів та клієнтів), надаючи їм відповідні права доступу;

4) як адміністратор, я хочу фіксувати надходження коштів від клієнтів, щоб забезпечувати коректність фінансового обліку фірми;

5) як адміністратор, я хочу формувати звіти про ефективність роботи адвокатів (кількість виграних справ, сума принесених коштів), щоб приймати управлінські рішення;

6) як адміністратор, я хочу редагувати довідники (типи справ, статті кодексів, суди), щоб підтримувати актуальність нормативної інформації в системі.

1.3 Детальний опис функціоналу бази даних, що пропонується

На основі проведеного аналізу предметної області визначено мету розробки інформаційної системи, яка полягає в автоматизації діяльності адвокатської контори, забезпеченні ефективного обліку клієнтів та судових справ, контролю розкладу засідань та результатів слухань, а також оптимізації документообігу й фінансових операцій (гонорарів). Проєктована база даних повинна гарантувати цілісність, конфіденційність, доступність і своєчасне оновлення інформації, необхідної для підтримки прийняття управлінських рішень керівництвом контори.

Перелік завдань, які необхідно вирішити:

– автоматизація реєстрації клієнтів та створення нових судових справ відповідно до типу провадження;

– зберігання та актуалізація персональних даних клієнтів і професійних досьє адвокатів;

– планування та зберігання розкладу судових засідань;

– ведення журналу результатів засідань та винесених вироків;

– облік фінансових платежів і контроль повноти сплати гонорарів;

– формування процесуальних документів та звітів про ефективність роботи адвокатів.

Перелік функцій, які має підтримувати інформаційна система:

1) робота з клієнтами:

– додавання нового клієнта (реєстрація анкетних даних, вибір типу клієнта фіз./юр. особа);

– редагування даних клієнта (ПІБ, контакти, адреса);

– закріплення клієнта за конкретною судовою справою;

– пошук клієнтів за ПІБ, номером телефону, типом справи, статусом оплати;

– перегляд історії співпраці та архіву завершених справ.

2) робота з адвокатами:

– додавання нового адвоката (реєстрація у системі);

– редагування професійних даних (освіта, дата найму, зарплата);

– пошук адвокатів за ПІБ або статусом активності;

– перегляд закріплених справ та особистого розкладу засідань;

– аналіз навантаження адвокатів (кількість активних справ, виграні процеси).

3) робота з судовими справами та типами:

– створення картки судової справи з прив’язкою до типу (кримінальна, адміністративна тощо);

– формування команди справи (призначення адвоката та клієнта);

– перегляд списків справ за статусом (відкрита/закрита), типом, періодом;

– ведення довідника типів справ із зазначенням статей, кодексів, мінімальних та максимальних термінів покарання.

4) робота з судовими засіданнями та розкладом:

– ведення відомостей про заплановані слухання (дата, час, суд, суддя, номер залу); планування нових засідань з перевіркою конфліктів у розкладі адвоката;

– перегляд розкладу за конкретною справою, адвокатом або датою;

– фільтрація засідань за статусом (заплановано/відбулося/перенесено).

5) робота з результатами та ефективністю:

– реєстрація результатів засідань (проміжне рішення, вирок, перенесення);

– внесення даних про отриманий термін покарання для розрахунку ефективності захисту;

– перегляд підсумкових відомостей по справі;

– аналіз ефективності (порівняння отриманого терміну з максимально можливим за статтею).

6) робота з платежами:

– реєстрація вхідних платежів (дата, сума, метод оплати, призначення);

– перегляд історії оплат по конкретній справі або клієнту;

– контроль повноти оплати послуг згідно з тарифом;

– формування фінансових вибірок за періодом або типом послуг.

7) робота з документами та звітами:

– завантаження та зберігання документів, що стосуються справи (докази, клопотання);

– формування чека на оплату послуг;

– формування списків незавершених справ та боржників;

– підготовка аналітичних звітів для керівництва фірми.

Формулювання запитів, на які має давати відповідь програмна система:

1) визначення стану справ та навантаження:

– отримання списку активних справ, закріплених за певним адвокатом;

– формування переліку справ певної категорії (наприклад, "Крадіжка") з можливістю сортування за датою початку;

– визначення кількості справ у провадженні для аналізу завантаженості персоналу.

2) планування судового процесу:

– отримання розкладу засідань на найближчий тиждень для конкретного адвоката;

– визначення судді та номеру залу для обраного засідання.

3) аналіз результатів захисту:

– перегляд історії вироків по справах конкретного адвоката;

– визначення списку "ефективних" справ (де отриманий термін менший за середній);

– ідентифікація справ, що тривають довше визначеного терміну.

4) фінансовий контроль та облік:

– отримання списку клієнтів, які не повністю сплатили гонорар;

– розрахунок загальної суми прибутку фірми за поточний місяць або рік.

Перелік звітів, які має формувати система:

1) списки справ адвокатів:

– відображення переліку справ із зазначенням номера, клієнта, статті звинувачення та поточного статусу;

– використовується для розподілу навантаження та контролю строків.

2) розклад судових засідань:

– формування таблиці засідань із зазначенням дати, часу, суду, судді та учасників процесу;

– забезпечує своєчасне прибуття адвокатів до суду;

3) звіт про ефективність адвоката:

– показує статистику по завершених справах: кількість виграних процесів, різниця між загрожуючим та отриманим терміном покарання;

– слугує основою для преміювання співробітників;

4) фінансовий звіт про гонорари:

– відображає інформацію про суми надходжень, дати платежів та заборгованості клієнтів;

– використовується бухгалтерією для фінансового планування.

5) чек на оплату:

– автоматично формується при внесенні платежу, містить дані про платника, суму, послугу та реквізити;

– видається клієнту як підтвердження оплати.

Завдання автоматизації, що має реалізовувати система:

Автоматизація функціональних процесів у розробленій інформаційній системі адвокатської контори спрямована на зменшення кількості ручних операцій, підвищення точності юридичного та фінансового обліку, а також забезпечення оперативного доступу до матеріалів справ.

У межах інформаційної системи автоматизація включає перелік ключових задач, що базуються на аналітичних SQL-запитах, тригерах та збережених процедурах:

1) автоматизація роботи адміністратора:

– формування списку поточних клієнтів адвоката;

– пошук справ за фрагментом опису або номером;

– виявлення адвокатів, які не мають активних справ (для розподілу нових клієнтів);

– генерація звітних відомостей щодо прибутковості різних типів справ;

2) автоматизація роботи адвокатів:

– розрахунок ефективності захисту (порівняння отриманого вироку з нормами статті);

– автоматичне закриття справи при внесенні фінального рішення суду;

– формування історії захисту із зазначенням статей та результатів;

3) автоматизація фінансового обліку:

– розрахунок суми гонорарів за окремі справи, окремого адвоката або фірми в цілому у поточному році;

– контроль повноти оплати перед закриттям справи;

4) автоматизація перевірки й обробки даних:

– автоматична перевірка коректності дат (дата закінчення справи не може бути раніше дати початку);

– заборона видалення справи, якщо по ній існують проведені платежі (забезпечення цілісності);

– логування змін важливих параметрів (наприклад, зміни зарплати адвоката).

Висновки до розділу 1

У першому розділі було проведено аналіз предметної області діяльності адвокатської контори, сформульовано основні вимоги до інформаційної системи та визначено ключових учасників процесу (адвокатів, клієнтів, адміністрацію) та їхні потреби. На основі дослідження структури інформаційних потоків і бізнес-процесів сформовано постановку задачі та окреслено необхідний функціонал майбутньої бази даних, включаючи детальний перелік завдань, функцій, звітів, аналітичних запитів і механізмів автоматизації.

Отримані результати створюють концептуальну основу для подальшого проєктування логічної та фізичної моделі бази даних, що буде розглянуто в наступних розділах роботи.

2 ПРОЄКТУВАННЯ БАЗИ ДАНИХ ДЛЯ АВТОМАТИЗАЦІЇ ДІЯЛЬНОСТІ АДВОКАТСЬКОЇ КОНТОРИ

Метою проєктування бази даних є створення повної та формалізованої моделі предметної області адвокатської контори, яка слугує основою для побудови ефективної інформаційної системи. Проєктування дозволяє визначити ключові об’єкти предметної області, встановити зв’язки між ними, а також описати атрибути сутностей та обмеження, необхідні для забезпечення цілісності та узгодженості даних.

Концептуальна модель бази даних відображає логічну структуру інформації, що використовується в процесі надання правових послуг, ведення судових справ та взаємодії з клієнтами, а також визначає основу для побудови фізичної структури таблиць, первинних і зовнішніх ключів та механізмів контролю даних.

Виділені суттєві сутності, що характеризують предметну область, та їх характеристика представлені в таблиці 2.1.

Таблиця 2.1 – Суттєві сутності, що характеризують предметну область

Сутність	Характеристика
Користувач	Містить базові облікові дані (логін, пароль, роль, ПІБ) для доступу до системи. Є базовою сутністю, від якої успадковують Lawyer та Client.
Адвокат	Містить інформацію про професійну діяльність співробітника. Наслідує основні властивості від User.
Клієнт	Містить ідентифікаційні дані клієнта та дату реєстрації. Наслідує основні властивості від User.
Справа	Містить основні дані про юридичну справу (номер, статус, дати початку та завершення, опис суті справи).
Тип справи	Містить класифікацію справи згідно з законодавчими нормами (стаття, кодекс, мінімальний/максимальний штраф, опис). Є батьківською сутністю, яка класифікує сутність Case.
Судове засідання	Містить інформацію про конкретне слухання у справі (дата, час, зал суду, результат, нотатки). Є дочірньою сутністю для Case (одна справа може мати багато засідань).
Платіж	Містить дані про фінансові транзакції, пов'язані зі справою (сума, дата, метод оплати, призначення). Є дочірньою сутністю для Case (одна

Продовження таблиці 2.1

Сутність	Характеристика
Платіж	справа може мати багато платежів).
Документ	Містить інформацію про документи, завантажені у систему, та їхній тип (назва, шлях до файлу, дата завантаження). Є дочірньою сутністю для Case (одна справа може мати багато документів).

У таблиці 2.2 наведено виділені властивості об’єктів (атрибути) кожної сутності, що характеризують структуру інформаційної системи адвокатської контори. Для кожного атрибуту зазначено тип даних, ключову роль у таблиці (первинний або зовнішній ключ) та обмеження на значення. Такий опис дозволяє визначити структуру майбутніх таблиць бази даних, забезпечити цілісність і узгодженість даних під час зберігання й обробки.

Зокрема, у таблиці подано як обов’язкові поля ідентифікації (id), так і описові атрибути, що містять анкетні, контактні та службові дані клієнтів, адвокатів, судових справ, засідань, документів і платежів.

Таблиця 2.2 – Виділені атрибути сутностей та їх характеристики

Сутність	Властивість	Тип даних	Ключ	обмеження на значення
Користувач	id_користувача	цілочисельний	первинний	унікальне, додатне число, NOT NULL
	логін	символьний	потенційний	довжина ≤ 100, Унікальне
	пароль	символьний		довжина ≤ 255, NOT NULL
	роль	символьний		довжина ≤ 50, NOT NULL
	ел_пошта	символьний		формат: "назва@домен", унікальне
	останній_вхід	дата і час		може бути NULL
	повне_ім'я	символьний		довжина ≤ 200, NOT NULL
	дата_народження	дата		не більше поточної дати
	адреса	символьний		довжина ≤ 255
	телефон	символьний		формат: "+380XXXXXXXXX"
Адвокат	id_користувача	цілочисельни	первинний /	повинен посилатися на

Продовження Таблиці 2.2

Сутність	Властивість	Тип даних	Ключ	обмеження на значення
ТипСправи	макс_штраф	й	зовнішній	`id_користувача` у користувач
Адвокат	освіта	символьний		довжина ≤ 200
	дата_найму	дата		не більше поточної дати
	зарплата	дійсний		більше 0
	статус	символьний		довжина ≤ 50
Клієнт	id_користувача	цілочисельний	первинний / зовнішній	повинен посилатися на `id_користувача` у користувач
	тип_клієнта	символьний		довжина ≤ 50 (Фіз. / Юр. особа)
	дата_реєстрації	дата		не більше поточної дати
Справа	id_справи	цілочисельний	первинний	унікальне, додатне число, NOT NULL
	номер_справи	символьний	потенційний	довжина ≤ 50, унікальне
	статус	символьний		довжина ≤ 50, NOT NULL
	суть	символьний		довжина ≤ 500
	джерело_отримання	символьний		довжина ≤ 100
	дата_початку	дата		не пізніше `дата_закінчення`
	дата_закінчення	дата		більше дата_початку
	назва_суду	символьний		довжина ≤ 100
	id_типу_справи	цілочисельний	зовнішній	повинен посилатися на id_типу_справи у ТипСправи
СудовеЗасідання	id_засідання	цілочисельний	первинний	унікальне, додатне число, NOT NULL
	дата_засідання	дата		не більше поточної дати

Продовження таблиці 2.2

Сутність	Властивість	Тип даних	Ключ	обмеження на значення
СудовеЗасідання	ім'я_судді	символьний		довжина ≤ 150
	час_засідання	час		
	номер_залу	символьний		довжина ≤ 20
	результат	символьний		довжина ≤ 500
	нотатки	символьний		довжина ≤ 1000
	id_справи	цілочисельний	зовнішній	повинен посилатися на id_справи у справа
Платіж	id_платежу	цілочисельний	первинний	унікальне, додатне число, NOT NULL
	дата_платежу	дата		не більше поточної дати
	сума	дійсний		більше 0
	метод_оплати	символьний		довжина ≤ 50
	одержувач	символьний		довжина ≤ 100
	номер_квитанції	символьний		довжина ≤ 50
	id_справи	цілочисельний	зовнішній	повинен посилатися на `id_справи` у справа
Документ	id_документа	цілочисельний	первинний	унікальне, додатне число, NOT NULL
	тип_документа	символьний		довжина ≤ 50
	назва_документа	символьний		довжина ≤ 255 , NOT NULL
	дата_завантаження	дата		не більше поточної дати
	шлях_до_файлу	символьний		`Довжина ≤ 500 id_справи у Справа`
	автор	цілочисельний	зовнішній	повинен посилатися на `id_користувача` у користувач
ТипСправи	id_типу_справи	цілочисельний	первинний	унікальне, додатне число, NOT NULL
	стаття	символьний		довжина ≤ 50
	тип_кодексу	символьний		довжина ≤ 50
	мін_штраф	цілочисельний		≥ 0
	макс_штраф	цілочисельний		≥ 0
	опис	символьний		довжина ≤ 500
	збір			≥ 0

Кінець таблиці 2.2

Сутність	Властивість	Тип даних	Ключ	обмеження на значення
КлієнтСправа	id_справи	цілочисельний	Складений Первинний / Зовнішній	Повинен посилатися на `id_справи` у Справа
КлієнтСправа	id_клієнта	цілочисельний	Складений Первинний / Зовнішній	Повинен посилатися на `id_користувача` у Клієнт
АдвокатСправа	id_справи	цілочисельний	Складений Первинний / Зовнішній	Повинен посилатися на `id_справи` у Справ
	id_адвоката	цілочисельний	Складений Первинний / Зовнішній	Повинен посилатися на `id_користувача` у Адвокат

У таблиці 2.3 подано основні типи зв’язків між сутностями, які формують логічну структуру бази даних адвокатської контори. Зв’язки визначають, як об’єкти предметної області взаємодіють між собою під час зберігання та обробки даних. У таблиці використано позначення 1:N – «один до багатьох», N:M – «багато до багатьох» та Супертип:Підтип, які визначають кратність та характер зв’язків. Таке представлення дає змогу забезпечити логічну узгодженість даних і є основою для побудови ER-діаграми.

Зв’язок Користувач і Адвокат/Клієнт – Супертип до Підтипу. Це специфічний тип зв’язку (спадкування), де сутності Адвокат і Клієнт наслідують загальні атрибути (логін, пароль, ПІБ) від базової сутності Користувач. Реалізується через спільний первинний ключ `user_id`, який одночасно є зовнішнім ключем у підлеглих таблицях.

Зв’язок Справа і Клієнт – багато до багатьох (N:M). Одна справа може стосуватися кількох клієнтів (наприклад, група позивачів або відповідачів), а один клієнт може бути фігурантом кількох справ одночасно. Для реалізації цього зв’язку використовується асоціативна сутність `ClientCase`, що містить зовнішні ключі `case_id` та `client_id`, які посилаються на таблиці Справа та Клієнт відповідно.

Зв’язок Справа і Адвокат – багато до багатьох (N:M). Над однією складною справою може працювати група адвокатів, водночас один адвокат може вести кілька справ паралельно. Цей зв’язок реалізовано через асоціативну сутність `LawyerCase`, у якій містяться ключі `case_id` та `lawyer_id`.

Зв’язок Тип Справи і Справа – один до багатьох (1:N). Кожен тип справи (наприклад, «Крадіжка» або «Розлучення») класифікує багато конкретних судових справ, які розглядаються за схожими юридичними нормами. Зв’язок реалізується через зовнішній ключ `case_type_id` у таблиці Справа, який посилається на таблицю Тип Справи.

Зв’язок Справа і Судове Засідання – один до багатьох (1:N). У рамках однієї судової справи може відбуватися багато судових засідань, але кожне конкретне засідання належить лише одній справі. Зв’язок реалізовано за допомогою зовнішнього ключа `case_id` у таблиці Судове Засідання.

Зв’язок Справа і Платіж – один до багатьох (1:N). По одній справі може бути здійснено декілька платежів (аванс, оплата частинами), але кожен платіж прив’язаний до конкретної справи. Зв’язок реалізується зовнішнім ключем `case_id` у таблиці Платіж.

Зв’язок Справа і Документ – один до багатьох (1:N). До матеріалів однієї справи може бути долучено безліч документів (докази, клопотання, рішення), проте кожен документ належить до певної справи. Зв’язок забезпечується атрибутом `case_id` у таблиці Документ.

Таблиця 2.3 – Типи зв’язків між сутностями

Сутність	Тип зв’язку	Сутність
Користувач	Супертип : Підтип	Адвокат
Користувач	Супертип : Підтип	Клієнт
Справа	1 : N	СудовеЗасідання
Справа	1 : N	Платіж
Справа	1 : N	Документ

Кінець таблиці 2.3

Сутність	Тип зв’язку	Сутність
Справа	M : N	Адвокат
ТипСправи	1 : N	Справа

У процесі проєктування бази даних було виконано нормалізацію відношень з метою усунення надлишковості та забезпечення логічної цілісності даних. Нормалізація проводилася поетапно, шляхом приведення відношень до першої, другої та третьої нормальних форм.

Перша нормальна форма (1НФ). Усі таблиці бази даних приведені до першої нормальної форми, оскільки їх атрибути є атомарними і не містять повторюваних груп. Наприклад, у таблиці `Client` кожне поле (повне ім’я, телефон, адреса, тип клієнта) зберігає лише одне значення. Подібний підхід застосовано і в інших сутностях – `Lawyer`, `CaseType`, `LegalCase`, `CourtHearing`, `Payment`. Первинні ключі однозначно ідентифікують кожен запис.

Друга нормальна форма (2НФ). Всі таблиці задовольняють другу нормальну форму, оскільки неключові атрибути залежать від повного первинного ключа. Наприклад, у таблиці `CourtHearing` атрибути `hearing_date`, `judge_name`, `result` залежать від ключа `hearing_id`, а не від його частини. Особливим випадком є таблиці `LawyerCase` та `ClientCase`, які мають складені первинні ключі (наприклад, `case_id` + `lawyer_id`). У цих таблицях запис про участь адвоката у справі залежить саме від повного набору ключів, тому умови 2НФ виконані.

Третя нормальна форма (3НФ). Жодна таблиця не містить транзитивних залежностей. Наприклад, у таблиці `Users` атрибут `full_name` залежить лише від первинного ключа `user_id` і не залежить від інших атрибутів (наприклад, `role`). У таблиці `Payment` атрибути `amount` та `payment_date` також визначаються тільки первинним ключем `payment_id`. Аналогічно у таблиці

`CaseType` атрибути `min_penalty`, `max_penalty` та `fee` залежать виключно від ідентифікатора типу справи `case_type_id`.

На рис. 2.1 представлено концептуальну модель бази даних у вигляді діаграми класів UML, що відображає ключові сутності предметної області адвокатської контори, їх атрибути та асоціативні зв'язки. Діаграма класів демонструє логічну структуру системи на концептуальному рівні та дозволяє сформувати початкове уявлення про організацію інформації, що використовується в процесі надання юридичних послуг. Завдяки наочному відображенню об'єктів та їх взаємодії стає можливим оцінити повноту моделі та виявити механізми спадкування для сутностей «User», «Lawyer» та «Client».

На рис. 2.2 зображено ER-діаграму в нотації Crow's Foot, створену на основі виділених сутностей та аналізу предметної області. Діаграма містить сутності «Users (Користувачі)», «Lawyer (Адвокати)», «Client (Клієнти)», «LegalCase (Справи)», «CaseType (Типи справ)», «CourtHearing (Судові засідання)», «Payment (Платежі)» та «Document (Документи)». Також відображено асоціативні сутності «LawyerCase» та «ClientCase», які використовуються для реалізації зв'язків типу «багато до багатьох» між учасниками процесу та справами. Для кожної сутності визначено атрибути, що формують її інформаційний зміст, а також позначено первинні та зовнішні ключі, необхідні для підтримання цілісності даних.

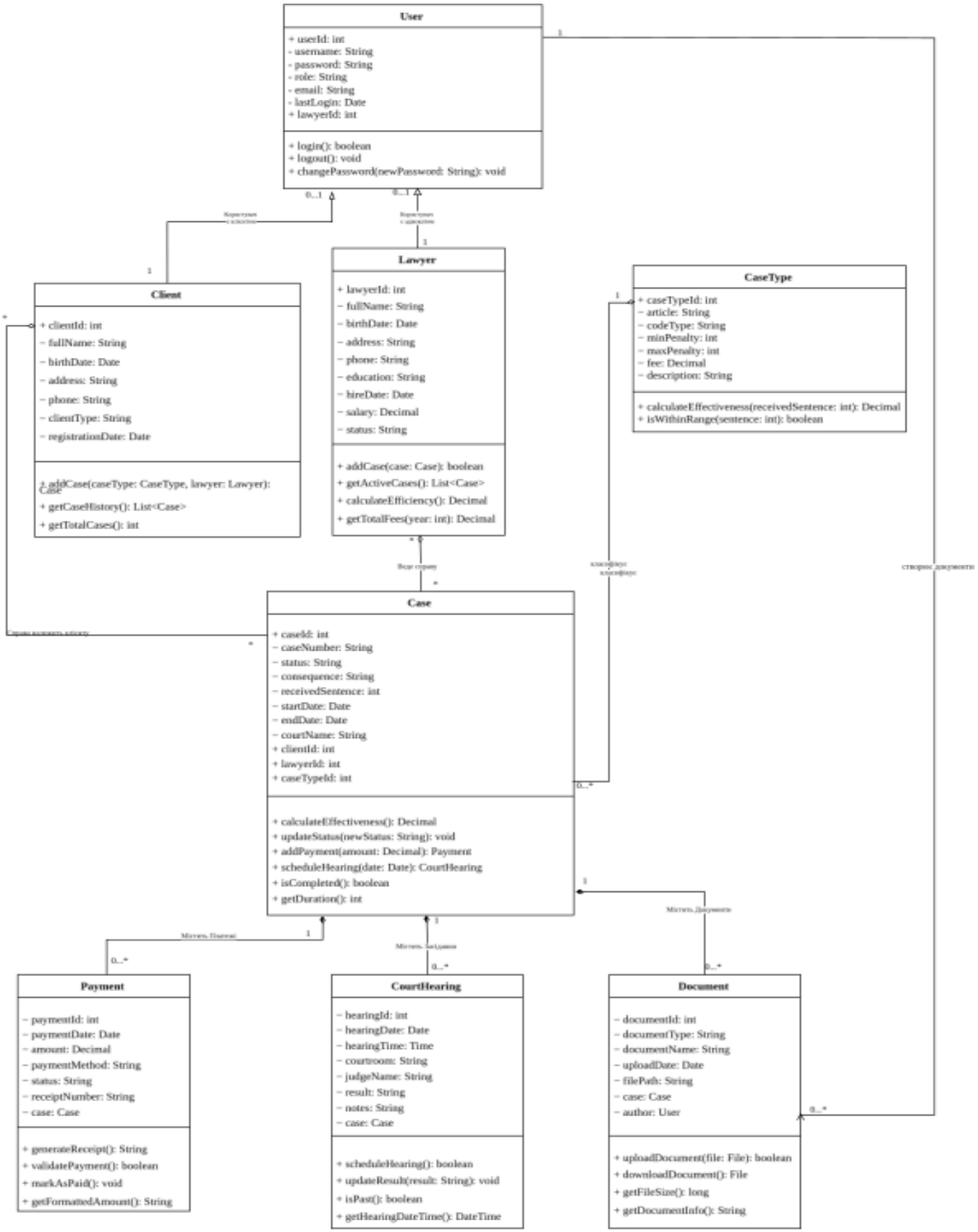


Рисунок 2.1 – Діаграма класів інформаційної підтримки діяльності адвокатської контори

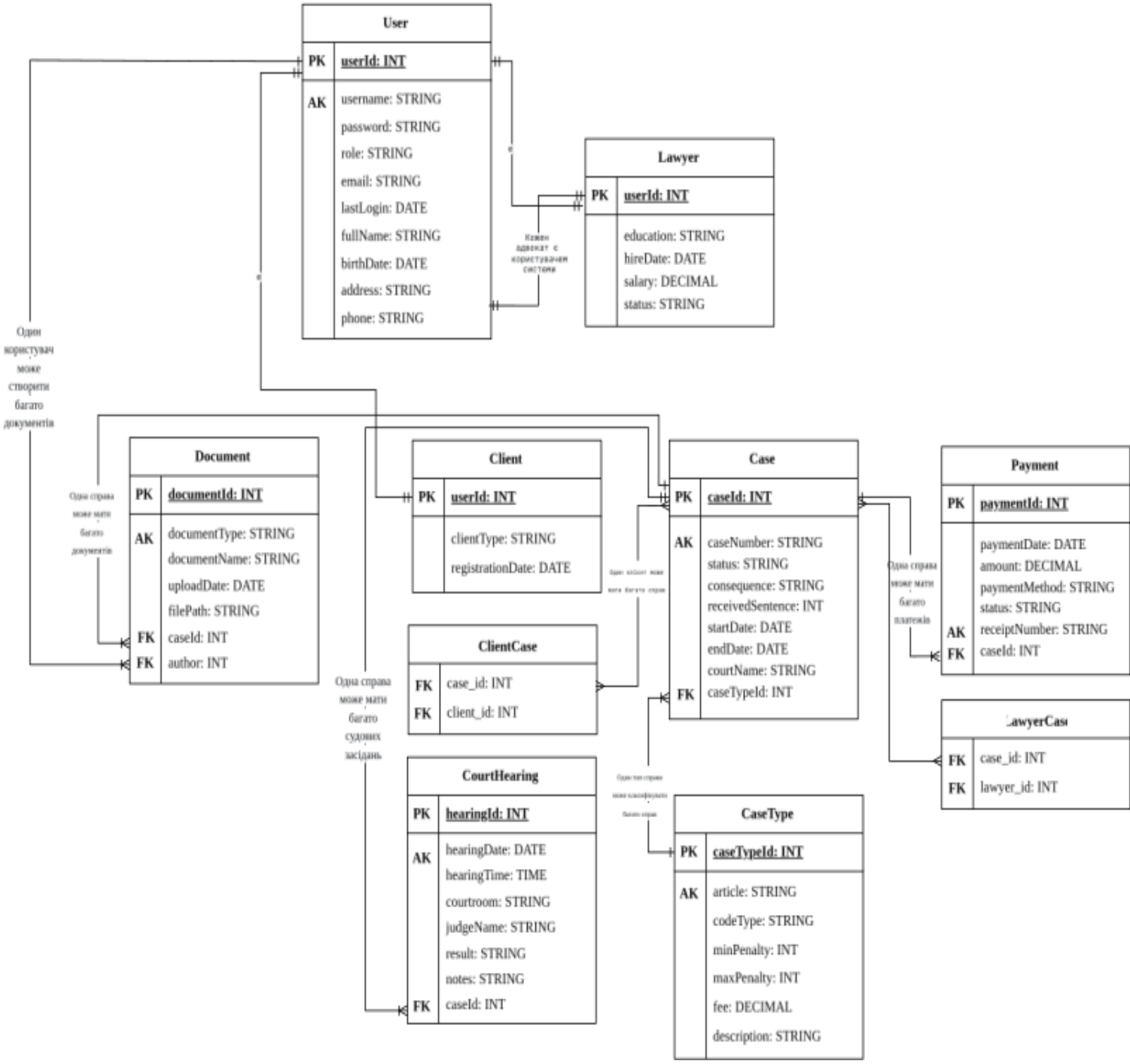


Рисунок 2.2 – ER-діаграма в нотації Crow’s Foot

Висновки до розділу 2

У цьому розділі було виконано концептуальне проєктування бази даних для підтримки діяльності адвокатської контори, визначено ключові сутності предметної області, їх атрибути та зв’язки, а також побудовано концептуальну ER-модель. Проведений аналіз дозволив сформувати логічну структуру даних,

необхідну для забезпечення цілісності, узгодженості та ефективної організації бізнес-процесів у юридичній фірмі.

Отримані результати є основою для переходу до етапу фізичного проєктування та реалізації структури бази даних у середовищі СУБД PostgreSQL.

3 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для реалізації проєкту інформаційної системи адвокатської контори було обрано сучасне та продуктивне програмне забезпечення, яке забезпечує стабільність роботи, захист конфіденційних даних клієнтів та зручність адміністрування. Середовищем розробки є персональний комп'ютер з процесором Intel Core i5, об'ємом оперативної пам'яті 16 ГБ, що працює під керуванням операційної системи Linux Ubuntu 24.04 LTS.

У процесі роботи використовувалася система керування базами даних PostgreSQL версії 17.7. Вибір цієї СУБД обумовлений її надійністю, високою продуктивністю, підтримкою транзакцій (що критично важливо для фінансових операцій), потужними механізмами блокування, розвиненими засобами для оптимізації запитів, а також широкими можливостями роботи з резервним копіюванням і відновленням даних.

Адміністрування та візуалізація структури бази даних виконувались за допомогою pgAdmin 4 версії 9.9, встановленої як Desktop-додаток. pgAdmin забезпечує зручний графічний інтерфейс для написання SQL-запитів, створення таблиць, тригерів і представлень, моніторингу стану сервера, перегляду блокувань і планів виконання.

Висновки до розділу 3

У даному розділі було обґрунтовано вибір програмного забезпечення та середовища розробки для створення інформаційної системи адвокатської контори. Визначено оптимальну конфігурацію апаратної частини комп'ютера та обрано СУБД PostgreSQL як надійну і функціональну основу для реалізації бази даних.

Застосування pgAdmin 4 забезпечило зручність адміністрування й підвищило ефективність роботи над проєктом.

4 СТВОРЕННЯ БАЗИ ДАНИХ

У цьому розділі представлено послідовність створення об’єктів бази даних, необхідних для функціонування інформаційної системи для підтримки діяльності адвокатської контори. На основі розробленої логічної моделі засобами мови SQL було створено базові та додаткові структурні елементи бази даних, що забезпечують цілісність, узгодженість й коректну обробку інформації про клієнтів, судові справи, засідання та фінансові операції.

4.1 Створення таблиць

Для початку необхідно створити саму базу даних, у якій зберігатимуться всі таблиці, зв’язки та інші об’єкти. Це виконується за допомогою команди `CREATE DATABASE`.

```
CREATE DATABASE lawyer_bd;
```

Створення доменів (користувацьких типів даних). Домени створюються на основі вбудованих типів даних і дозволяють визначити власні правила перевірки значень, які можна повторно використовувати в таблицях (наприклад, для телефонних номерів, пошти або грошових сум). Це підвищує узгодженість даних і зменшує дублювання коду.

```
CREATE DOMAIN d_phone AS VARCHAR(20)
CHECK (VALUE LIKE '+380%' AND LENGTH(VALUE) = 13);

CREATE DOMAIN d_email AS VARCHAR(100)
CHECK (POSITION('@' IN VALUE) > 1);

CREATE DOMAIN d_money AS DECIMAL(10, 2)
CHECK (VALUE >= 0);

CREATE DOMAIN d_status AS VARCHAR(50)
CHECK (VALUE IN ('Active', 'Closed', 'Pending', 'Archived'));
```

Далі створюються таблиці, які не мають зовнішніх ключів і не залежать від інших. Це базові сутності та довідники, на які будуть посилатися інші таблиці.

Для кожної таблиці, яка має первинний ключ `id`, використовується псевдотип `SERIAL` (або створюється власна послідовність `SEQUENCE`). Вона забезпечує автоматичне генерування унікальних ідентифікаторів для нових записів.

```
CREATE TABLE Users (
    user_id SERIAL PRIMARY KEY,
    login VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    role VARCHAR(50) NOT NULL CHECK (role IN ('Admin', 'Lawyer', 'Client')),
    email d_email UNIQUE NOT NULL,
    last_login TIMESTAMP,
    full_name VARCHAR(200) NOT NULL,
    birth_date DATE CHECK (birth_date <= CURRENT_DATE),
    address VARCHAR(255),
    phone d_phone
);
```

	user_id [PK] integer	login character varying (100)	password character varying (255)	role character varying (50)	email character varying (100)	last_login timestamp without time zone	full_name character varying (200)	birth_date date	address character varying (255)	phone character varying (20)
1	1	lawyer_ivanov	pass1	Lawyer	ivanov@law.ua	[null]	Іванов Іван Іванович	1985-05-20	Київ, вул. Хрещатик, 1	+380501112233
2	2	lawyer_petrenko	pass2	Lawyer	petrenko@law.ua	[null]	Петренко Петро Петрович	1990-08-15	Одеса, вул. Дерибасівська, 5	+380679998877
3	3	lawyer_sidorova	pass3	Lawyer	sidorova@law.ua	[null]	Сидорова Олена Сергіївна	1988-03-10	Львів, пл. Ринок, 12	+380934445566
4	4	lawyer_bondar	pass4	Lawyer	bondar@law.ua	[null]	Бондар Андрій Вікторович	1982-11-25	Харків, вул. Сумська, 20	+380507778899
5	5	lawyer_koval	pass5	Lawyer	koval@law.ua	[null]	Коваль Марина Олександрів...	1995-01-14	Дніпро, пр. Яворницького, 10	+380631112233
6	6	lawyer_melnik	pass6	Lawyer	melnik@law.ua	[null]	Мельник Сергій Юрійович	1980-06-30	Київ, вул. Антоновича, 45	+380975556644
7	7	lawyer_tkach	pass7	Lawyer	tkach@law.ua	[null]	Ткач Тетяна Володимирівна	1992-09-09	Одеса, Французький бульвар...	+380661234567
8	8	lawyer_boyko	pass8	Lawyer	boyko@law.ua	[null]	Бойко Роман Васильович	1987-12-12	Вінниця, вул. Соборна, 50	+380509876543
9	9	lawyer_kravchuk	pass9	Lawyer	kravchuk@law.ua	[null]	Кравчук Ольга Ігорівна	1993-04-05	Полтава, вул. Шевченка, 3	+380681122334
10	10	lawyer_shevch	pass10	Lawyer	shevch@law.ua	[null]	Шевченко Тарас Миколайов...	1979-02-20	Чернігів, пр. Миру, 15	+380639988776
11	11	client_popov	cl_pass1	Client	popov@gmail.com	[null]	Попов Олексій Андрійович	1998-07-22	Київ, вул. Лесі Українки, 5	+380991112233
12	12	client_novak	cl_pass2	Client	novak_corp@biz.ua	[null]	ТОВ "Новак Індастріз"	2010-01-01	Київ, вул. Промислова, 1	+380442223344
13	13	client_riabova	cl_pass3	Client	riabova@ukr.net	[null]	Рябова Інна Михайлівна	1985-02-14	Одеса, вул. Канатна, 22	+380675551122
14	14	client_build	cl_pass4	Client	budpostach@corp.com	[null]	ПП "БудПостач"	2015-05-20	Львів, вул. Городоцька, 100	+380503334455
15	15	client_lis	cl_pass5	Client	lis@gmail.com	[null]	Лисенко Віктор Петрович	1990-11-11	Харків, вул. Науки, 45	+380936667788
16	16	client_dmitr	cl_pass6	Client	dmitruk@mail.com	[null]	Дмитрук Аліна Сергіївна	2000-08-30	Дніпро, вул. Титова, 8	+380669990011
17	17	client_agro	cl_pass7	Client	agromir@farm.ua	[null]	ТОВ "АгроСвіт"	2008-03-15	Вінниця, вул. Пирогова, 1	+380678887766
18	18	client_zhuk	cl_pass8	Client	zhuk@gmail.com	[null]	Жук Максим Олегович	1975-04-25	Житомир, вул. Київська, 10	+380501239876
19	19	client_logist	cl_pass9	Client	fasttrans@log.com	[null]	ТОВ "Швидка Логістика"	2019-09-01	Одеса, вул. Балківська, 55	+380445678901
20	20	client_pavlov	cl_pass10	Client	pavlov@ukr.net	[null]	Павлов Денис Іванович	1996-12-05	Миколаїв, пр. Центральний, 9	+380635554433

Рисунок 4.1.1 – Таблиця “Users”, створена у БД

```
CREATE TABLE CaseType (
    case_type_id SERIAL PRIMARY KEY,
    article VARCHAR(50),
    code_type VARCHAR(50),
    min_penalty INT CHECK (min_penalty >= 0),
    max_penalty INT CHECK (max_penalty >= 0),
    fee d_money,
    description VARCHAR(500)
);
```

	case_type_id [PK] integer	article character varying (50)	code_type character varying (50)	min_penalty integer	max_penalty integer	fee numeric (10,2)	description character varying (500)
1	1	Ст. 115	Кримінальний	7	15	50000.00	Умисне вбивство
2	2	Ст. 185	Кримінальний	1	5	15000.00	Крадіжка
3	3	Ст. 190	Кримінальний	2	8	20000.00	Шахрайство
4	4	Ст. 286	Кримінальний	3	10	25000.00	ДТП з тяжкими наслідками
5	5	Ст. 130	Адміністративний	0	1	10000.00	Керування у нетверезому стані
6	6	Ст. 124	Адміністративний	0	1	5000.00	Порушення ПДР, що спричинило ДТП
7	7	Сімейний	Цивільний	0	0	8000.00	Розірвання шлюбу (Розлучення)
8	8	Господарський	Господарський	0	0	12000.00	Стягнення заборгованості між юр. особа...
9	9	Трудовий	Цивільний	0	0	6000.00	Поновлення на роботі
10	10	Земельний	Цивільний	0	0	15000.00	Суперечки щодо меж земельної ділянки

Рисунок 4.1.2 – Таблиця “CaseType”, створена у БД

Створюємо таблицю “Lawyer” (рисунок 4.1.3), яка зберігає специфічну інформацію про адвокатів контори. Для кожного адвоката вказуються освіта, дата найму та зарплата. Таблиця реалізує зв'язок типу "супертип-підтип" із таблицею “Users”. Поле `lawyer_id` є одночасно первинним ключем і зовнішнім ключем, що посилається на `user_id`, забезпечуючи каскадне видалення (`ON DELETE CASCADE`).

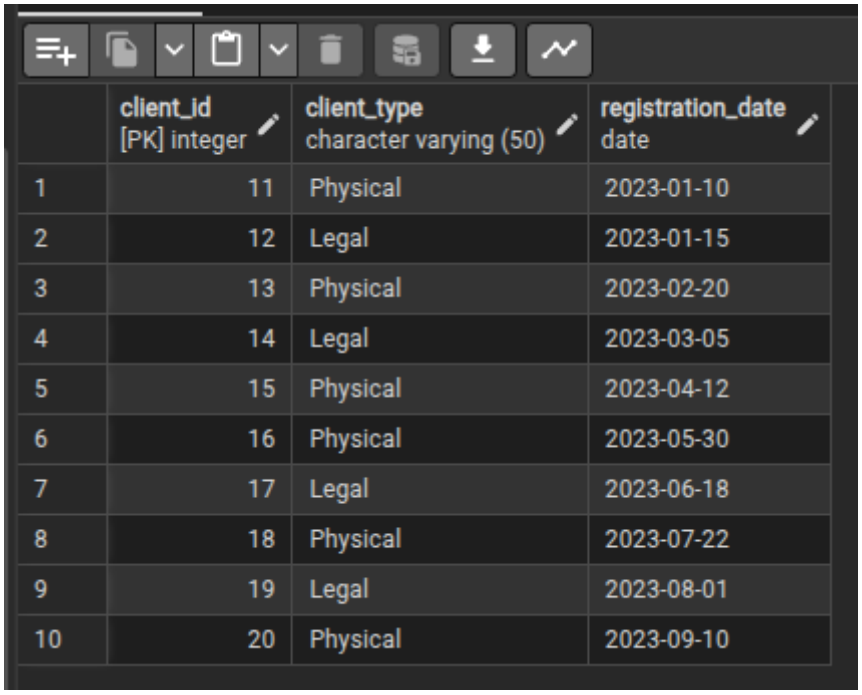
```
CREATE TABLE Lawyer (
    lawyer_id INT PRIMARY KEY REFERENCES Users(user_id) ON DELETE CASCADE,
    education VARCHAR(200),
    hire_date DATE CHECK (hire_date <= CURRENT_DATE),
    salary d_money,
    status VARCHAR(50) DEFAULT 'Active'
);
```

	lawyer_id [PK] integer	education character varying (200)	hire_date date	salary numeric (10,2)	status character varying (50)
1	1	КНУ ім. Шевченка, Право	2015-09-01	30000.00	Active
2	2	НЮУ ім. Ярослава Мудрого	2018-02-15	25000.00	Active
3	3	ЛНУ ім. Франка, Юридичний	2019-06-20	22000.00	Active
4	4	ОЮА (Одеська Юракадемія)	2010-11-01	40000.00	Active
5	5	Академія Адвокатури Украї...	2020-01-10	20000.00	Active
6	6	Харківський Університет ВС	2012-05-05	35000.00	Active
7	7	КНУ ім. Шевченка, Магістр	2021-09-01	18000.00	Active
8	8	ОЮА, Кримінальне право	2016-03-12	28000.00	Active
9	9	НЮУ ім. Ярослава Мудрого	2017-08-25	27000.00	Inactive
10	10	ЛНУ ім. Франка	2014-12-01	32000.00	Active

Рисунок 4.1.3 – Таблиця “Lawyer”, створена у БД

Створюємо таблицю “Client” (рисунок 4.1.4), яка зберігає інформацію про клієнтів. Для кожного клієнта вказується тип (фізична або юридична особа) та дата реєстрації. Аналогічно до адвокатів, ця таблиця пов'язана з таблицею “Users” через ключ `client_id`.

```
CREATE TABLE Client (
    client_id INT PRIMARY KEY REFERENCES Users(user_id) ON DELETE CASCADE,
    client_type VARCHAR(50) CHECK (client_type IN ('Physical', 'Legal')),
    registration_date DATE DEFAULT CURRENT_DATE
);
```



	client_id [PK] integer	client_type character varying (50)	registration_date date
1	11	Physical	2023-01-10
2	12	Legal	2023-01-15
3	13	Physical	2023-02-20
4	14	Legal	2023-03-05
5	15	Physical	2023-04-12
6	16	Physical	2023-05-30
7	17	Legal	2023-06-18
8	18	Physical	2023-07-22
9	19	Legal	2023-08-01
10	20	Physical	2023-09-10

Рисунок 4.1.4 – Таблиця “Client”, створена у БД

Створюємо основну таблицю “LegalCase” (рисунок 4.1.5), яка зберігає інформацію про судові справи. Кожен запис містить номер справи, статус, дати початку та завершення, а також результат (отриманий термін). Атрибут `case_type_id` є зовнішнім ключем і встановлює зв’язок із довідником “CaseType”.

```
CREATE TABLE LegalCase (
    case_id SERIAL PRIMARY KEY,
    case_number VARCHAR(50) UNIQUE NOT NULL,
    status VARCHAR(50) NOT NULL DEFAULT 'Open',
    consequence VARCHAR(500),
    received_sentence INT,
    start_date DATE DEFAULT CURRENT_DATE,
    end_date DATE CHECK (end_date >= start_date),
    court_name VARCHAR(100),
```

```
    case_type_id INT REFERENCES CaseType(case_type_id)
);
```

	case_id [PK] integer	case_number character varying (50)	status character varying (50)	consequence character varying (500)	received_sentence integer	start_date date	end_date date	court_name character varying (100)	case_type_id integer
1	1	CASE-23-001	Closed	[null]	8	2023-01-15	2023-05-20	Печерський районний суд	1
2	2	CASE-23-002	Open	[null]	[null]	2023-02-10	[null]	Шевченківський суд	2
3	3	CASE-23-003	Closed	[null]	0	2023-03-01	2023-04-01	Київський апеляційний суд	5
4	4	CASE-23-004	Open	[null]	[null]	2023-05-15	[null]	Приморський суд Одеси	7
5	5	CASE-23-005	Closed	[null]	0	2023-06-01	2023-09-10	Господарський суд Києва	8
6	6	CASE-23-006	Open	[null]	[null]	2023-07-20	[null]	Галицький суд Львова	3
7	7	CASE-23-007	Closed	[null]	4	2023-08-05	2023-11-25	Київський районний суд Харко...	4
8	8	CASE-23-008	Open	[null]	[null]	2023-09-01	[null]	Вінницький міський суд	9
9	9	CASE-23-009	Open	[null]	[null]	2023-10-10	[null]	Дніпровський суд	6
10	10	CASE-23-010	Closed	[null]	0	2023-02-15	2023-06-30	Одеський окружний адмінсуд	10

Рисунок 4.1.5 – Таблиця “LegalCase”, створена у БД

Створюємо проміжні таблиці “LawyerCase” та “ClientCase” (рисунки 4.1.6 та 4.1.7), які реалізують зв’язки типу багато-до-багатьох (N:M) між справами та учасниками процесу. Таблиця “LawyerCase” описує, які адвокати ведуть конкретну справу, а “ClientCase” – яких клієнтів вона стосується. Вони складаються з пар зовнішніх ключів, що разом утворюють складений первинний ключ.

```
CREATE TABLE LawyerCase (
    case_id INT REFERENCES LegalCase(case_id) ON DELETE CASCADE,
    lawyer_id INT REFERENCES Lawyer(lawyer_id) ON DELETE CASCADE,
    PRIMARY KEY (case_id, lawyer_id)
);
```

	case_id [PK] integer	lawyer_id [PK] integer
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	1
8	8	2
9	9	3
10	10	4

Рисунок 4.1.6 – Таблиця “LawyerCase”, створена у БД

```
CREATE TABLE ClientCase (
    case_id INT REFERENCES LegalCase(case_id) ON DELETE CASCADE,
    client_id INT REFERENCES Client(client_id) ON DELETE CASCADE,
    PRIMARY KEY (case_id, client_id)
);
```

	case_id [PK] integer	client_id [PK] integer
1	1	11
2	2	13
3	3	15
4	4	16
5	5	12
6	6	18
7	7	20
8	8	14
9	9	11
10	10	19

Рисунок 4.1.7 – Таблиця “ClientCase”, створена у БД

Створюємо таблицю “CourtHearing” (рисунок 4.1.8), яка зберігає графік судових засідань. Кожен запис містить дату, час, номер залу, ім'я судді та результат слухання. Атрибут `case_id` забезпечує прив’язку засідання до конкретної судової справи.

```
CREATE TABLE CourtHearing (
    hearing_id SERIAL PRIMARY KEY,
    hearing_date DATE NOT NULL,
    hearing_time TIME,
    courtroom VARCHAR(20),
    judge_name VARCHAR(150),
    result VARCHAR(500),
    notes VARCHAR(1000),
    case_id INT REFERENCES LegalCase(case_id) ON DELETE CASCADE
);
```

	hearing_id [PK] integer	hearing_date date	hearing_time time without time zone	courtroom character varying (20)	judge_name character varying (150)	result character varying (500)	notes character varying (1000)	case_id integer
1	1	2023-02-01	10:00:00	Зал 1	Суддя Вовк	Перерва	Потребує додаткових дока…	1
2	2	2023-05-19	11:00:00	Зал 1	Суддя Вовк	Вирок	Винний, 8 років	1
3	3	2023-03-10	09:30:00	Зал 5	Суддя Іваненко	Розгляд по суті	Свідки не з’явились	2
4	4	2023-03-20	14:00:00	Зал 3	Суддя Петренко	Рішення	Позбавлення прав на 1 рік	3
5	5	2023-06-01	10:00:00	Каб. 202	Суддя Сидоренко	Попереднє засідання	Мирова угода можлива	4
6	6	2023-08-15	12:00:00	Зал 10	Суддя Коваль	Рішення	Позов задоволено	5
7	7	2023-08-01	15:00:00	Зал 2	Суддя Мельник	Перенесено	Хвороба адвоката	6
8	8	2023-11-20	11:30:00	Зал 7	Суддя Гнатюк	Вирок	4 роки умовно	7
9	9	2023-09-15	09:00:00	Зал 4	Суддя Ткачук	Слухання	Відкладено	8
10	10	2023-06-25	16:00:00	Зал 6	Суддя Бондаренко	Рішення	Відмовлено у позові	10

Рисунок 4.1.8 – Таблиця “CourtHearing”, створена у БД

Створюємо таблицю “Payment” (рисунок 4.1.9), яка зберігатиме інформацію про фінансові операції (гонорари). Для кожного запису зберігається дата, сума, метод оплати та статус. Поле case_id є зовнішнім ключем, що встановлює зв’язок із таблицею “LegalCase”. Для атрибуту amount використано користувацький тип d_money, який гарантує, що сума оплати має додатне значення.

```
CREATE TABLE Payment (
    payment_id SERIAL PRIMARY KEY,
    payment_date DATE DEFAULT CURRENT_DATE,
    amount d_money,
    method VARCHAR(50),
    status VARCHAR(50),
    receipt_number VARCHAR(50),
    case_id INT REFERENCES LegalCase(case_id)
);
```

	payment_id [PK] integer	payment_date date	amount numeric (10,2)	method character varying (50)	status character varying (50)	receipt_number character varying (50)	case_id integer
1	1	2023-01-20	20000.00	Bank Transfer	Completed	REC-001	1
2	2	2023-04-10	30000.00	Bank Transfer	Completed	REC-002	1
3	3	2023-02-12	5000.00	Cash	Completed	REC-003	2
4	4	2023-03-01	10000.00	Card	Completed	REC-004	3
5	5	2023-05-20	4000.00	Card	Pending	REC-005	4
6	6	2023-06-05	6000.00	Bank Transfer	Completed	REC-006	5
7	7	2023-08-01	6000.00	Bank Transfer	Completed	REC-007	5
8	8	2023-08-10	12500.00	Cash	Completed	REC-008	7
9	9	2023-10-15	2500.00	Card	Failed	REC-009	9
10	10	2023-02-20	15000.00	Bank Transfer	Completed	REC-010	10

Рисунок 4.1.9 – Таблиця “Payment”, створена у БД

Створюємо таблицю “Document” (рисунок 4.1.10), яка призначена для зберігання інформації про файли, що стосуються судових справ. Таблиця містить назву документа, тип, шлях до файлу на сервері та дату завантаження. Реалізовано два зовнішні зв'язки: `case_id` вказує, до якої справи належить документ, а `author_id` посилається на таблицю “Users”, фіксуючи, хто саме завантажив файл.

```
CREATE TABLE Document (
    document_id SERIAL PRIMARY KEY,
    document_type VARCHAR(50),
    document_name VARCHAR(255) NOT NULL,
    upload_date DATE DEFAULT CURRENT_DATE,
    file_path VARCHAR(500),
    author_id INT REFERENCES Users(user_id),
    case_id INT REFERENCES LegalCase(case_id)
);
```

	document_id [PK] integer	document_type character varying (50)	document_name character varying (255)	upload_date date	file_path character varying (500)	author_id integer	case_id integer
1	1	Доказ	Протокол огляду місця події	2023-01-16	/docs/case1/protocol....	1	1
2	2	Клопотання	Клопотання про допит свідків	2023-02-05	/docs/case1/request.pdf	1	1
3	3	Заява	Позовна заява	2023-02-10	/docs/case2/claim.pdf	2	2
4	4	Протокол	Протокол поліції	2023-03-01	/docs/case3/police.pdf	3	3
5	5	Довідка	Свідоцтво про шлюб	2023-05-15	/docs/case4/marriage....	4	4
6	6	Договір	Договір поставки	2023-06-01	/docs/case5/contract....	5	5
7	7	Експертиза	Результати ДНК експертизи	2023-07-25	/docs/case6/dna.pdf	6	6
8	8	Характеристика	Характеристика з місця роботи	2023-08-10	/docs/case7/char.pdf	1	7
9	9	Наказ	Наказ про звільнення	2023-09-01	/docs/case8/order.pdf	2	8
10	10	Акт	Акт на право власності на зем...	2023-02-15	/docs/case10/land.pdf	4	10

Рисунок 4.1.10 – Таблиця “Document”, створена у БД

4.2 Створення представлень

Представлення (VIEW) – це віртуальна таблиця, яка зберігає SQL-запит, але не зберігає власних даних. Під час звернення до представлення система виконує запит, що в ньому описаний, і повертає результат як звичайну таблицю. Це дозволяє спростити складні запити, приховати структуру реальних таблиць та обмежити доступ до конфіденційних даних.

Створимо представлення “View_Active_Clients” (рисунок 4.2.1), яке відображає список поточних клієнтів, що мають відкриті судові справи. У цьому представленні об’єднуються дані з таблиць користувачів, адвокатів, клієнтів та справ. Воно дозволяє адвокатам швидко отримати контактні дані своїх підопічних, справи яких знаходяться в активній фазі розгляду.

```
CREATE OR REPLACE VIEW View_Active_Clients AS
SELECT
    u_lawyer.full_name AS Lawyer,
    u_client.full_name AS Client,
    u_client.phone AS Client_Phone,
    lc.case_number
FROM LawyerCase l_case
JOIN ClientCase c_case ON l_case.case_id = c_case.case_id
JOIN LegalCase lc ON l_case.case_id = lc.case_id
JOIN Users u_lawyer ON l_case.lawyer_id = u_lawyer.user_id
JOIN Users u_client ON c_case.client_id = u_client.user_id
WHERE lc.status = 'Open';
```

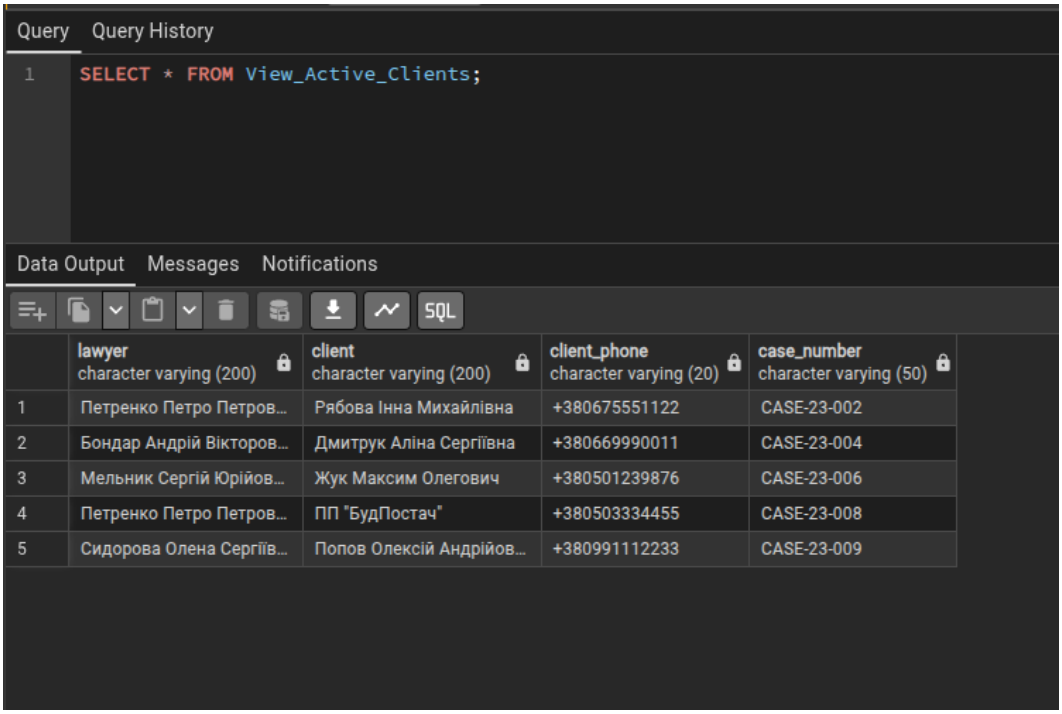


Рисунок 4.2.1 – Представлення “View_Active_Clients”, створене у БД

Створимо представлення “View_Upcoming_Hearings” (рисунок 4.2.2), яке показує розклад майбутніх судових засідань. Представлення фільтрує засідання за датою (тільки ті, що ще не відбулися) і виводить інформацію про час, номер судової зали, номер справи та відповідального адвоката. Це зручно використовувати для планування робочого часу юристів.

```
CREATE OR REPLACE VIEW View_Upcoming_Hearings AS
SELECT
    ch.hearing_date,
    ch.hearing_time,
    ch.courtroom,
    lc.case_number,
    u.full_name AS Lawyer_Name
FROM CourtHearing ch
JOIN LegalCase lc ON ch.case_id = lc.case_id
JOIN LawyerCase lwc ON lc.case_id = lwc.case_id
JOIN Users u ON lwc.lawyer_id = u.user_id
WHERE ch.hearing_date >= CURRENT_DATE;
```

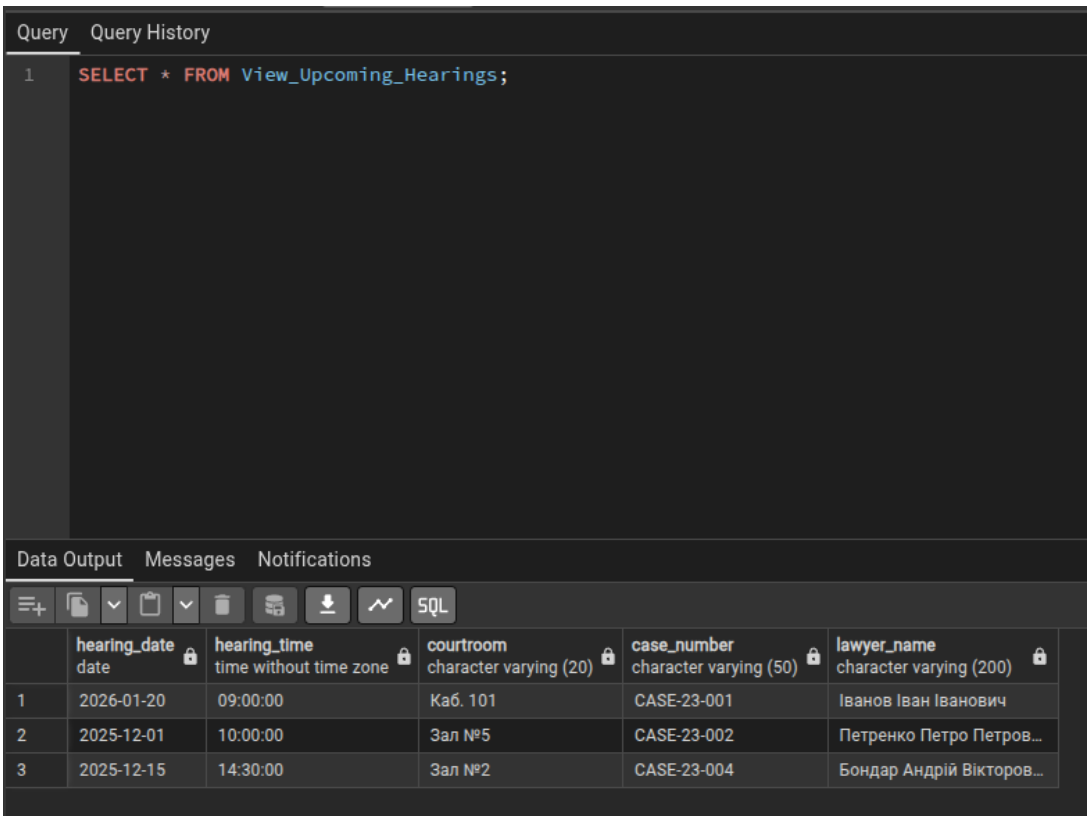


Рисунок 4.2.2 – Представлення “View_Upcoming_Hearings”, створене у БД

Створимо представлення “View_Case_Financials” (рисунок 4.12), яке призначене для фінансового аналізу діяльності контори. Воно агрегує дані про платежі по кожній справі, показуючи загальну суму фактично сплачених коштів (`Total_Paid`) та кількість транзакцій. Представлення використовується адміністрацією для контролю дебіторської заборгованості та оцінки прибутковості різних типів справ.

```
CREATE OR REPLACE VIEW View_Case_Financials AS
SELECT
    lc.case_number,
    ct.description AS Case_Type,
    COUNT(p.payment_id) AS Payment_Count,
    COALESCE(SUM(p.amount), 0) AS Total_Paid,
    lc.status
FROM LegalCase lc
JOIN CaseType ct ON lc.case_type_id = ct.case_type_id
LEFT JOIN Payment p ON lc.case_id = p.case_id
GROUP BY lc.case_number, ct.description, lc.status;
```

Query Query History

```
1  SELECT * FROM View_Case_Financials;
```

Data Output Messages Notifications

	case_number character varying (50)	case_type character varying (500)	payment_count bigint	total_paid numeric	status character varying (50)
1	CASE-23-003	Керування у нетверезому стані	1	10000.00	Closed
2	CASE-23-010	Суперечки щодо меж земельної ділянки	1	15000.00	Closed
3	CASE-23-008	Поновлення на роботі	0	0	Open
4	CASE-23-005	Стягнення заборгованості між юр. особа...	2	12000.00	Closed
5	CASE-23-001	Умисне вбивство	2	50000.00	Closed
6	CASE-23-002	Крадіжка	1	5000.00	Open
7	CASE-23-009	Порушення ПДР, що спричинило ДТП	1	2500.00	Open
8	CASE-23-006	Шахрайство	0	0	Open
9	CASE-23-007	ДТП з тяжкими наслідками	1	12500.00	Closed
10	CASE-23-004	Розірвання шлюбу (Розлучення)	1	4000.00	Open

Рисунок 4.2.3 – Представлення “View_Case_Financials”, створене у БД

4.3 Створення тригерів

У СУБД PostgreSQL тригер (TRIGGER) – це об’єкт, який автоматично виконує тригерну функцію у відповідь на певну подію з даними в таблиці: INSERT, UPDATE, DELETE або TRUNCATE. Тригери використовуються для забезпечення цілісності даних, автоматизації бізнес-логіки та аудиту змін.

Тригер перевірки коректності дат судової справи. Необхідно забезпечити, щоб у таблиці “LegalCase” дата завершення справи не була раніше за дату початку. Для цього створюється тригер, який перед вставкою або оновленням запису перевіряє хронологію. У разі помилки операція переривається з повідомленням.

```
CREATE OR REPLACE FUNCTION check_case_dates()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.end_date < NEW.start_date THEN
        RAISE EXCEPTION 'End date cannot be earlier than start date!';
    END IF;
```

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

CREATE TRIGGER trg_check_dates
BEFORE INSERT OR UPDATE ON LegalCase
FOR EACH ROW
EXECUTE FUNCTION check_case_dates();
```

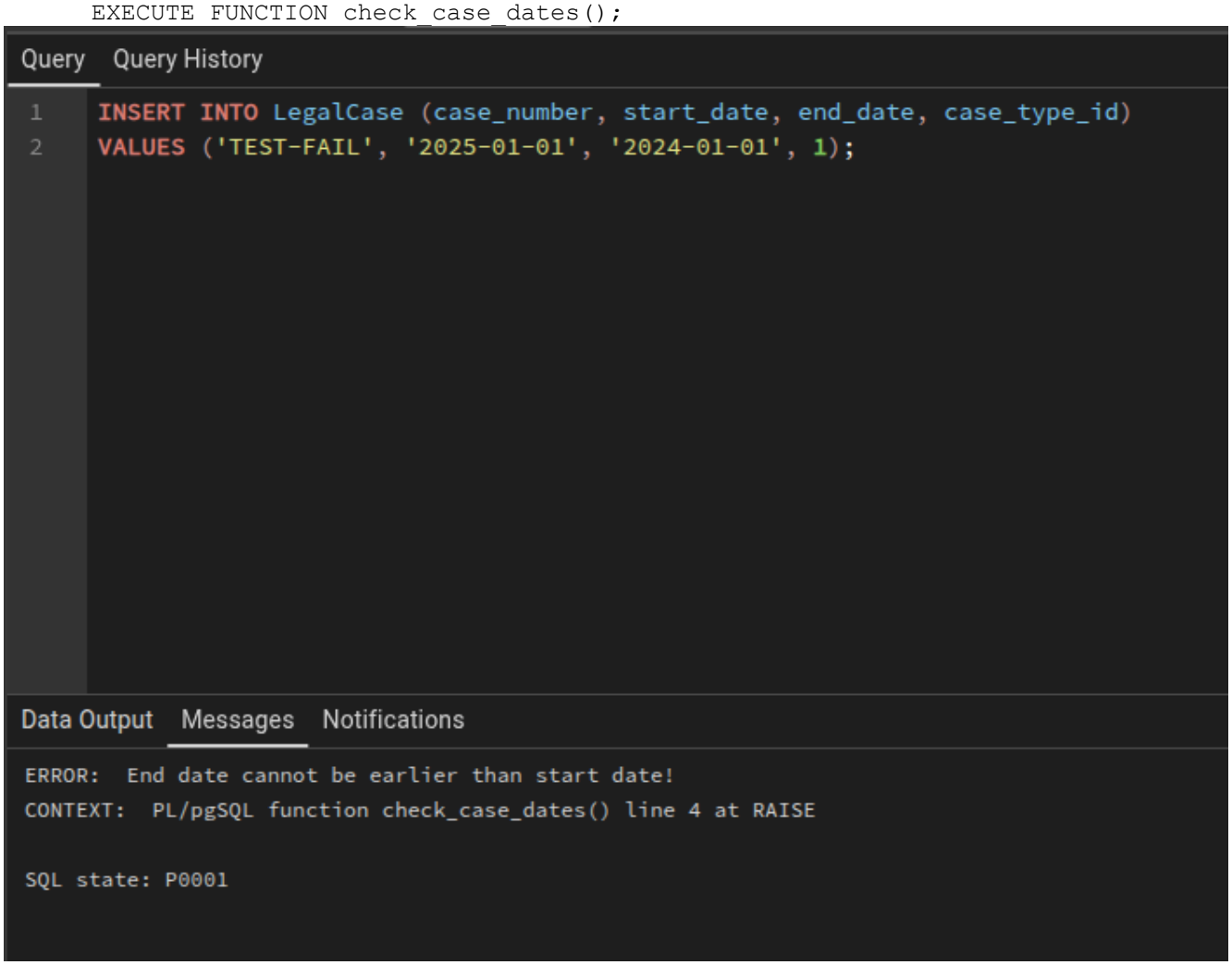


Рисунок 4.3.1 – Перевірка роботи тригера “trg_check_dates”

Тригер заборони видалення справи з платежами. Для забезпечення фінансової цілісності не можна видаляти судову справу, якщо за нею вже були проведені платежі. Тригер перевіряє наявність записів у таблиці “Payment” перед видаленням справи.

```
CREATE OR REPLACE FUNCTION prevent_case_deletion()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (SELECT 1 FROM Payment WHERE case_id = OLD.case_id) THEN
        RAISE EXCEPTION 'Cannot delete case with existing payments. Refund
first.';
    END IF;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER trg_prevent_delete
BEFORE DELETE ON LegalCase
FOR EACH ROW
EXECUTE FUNCTION prevent_case_deletion();
```

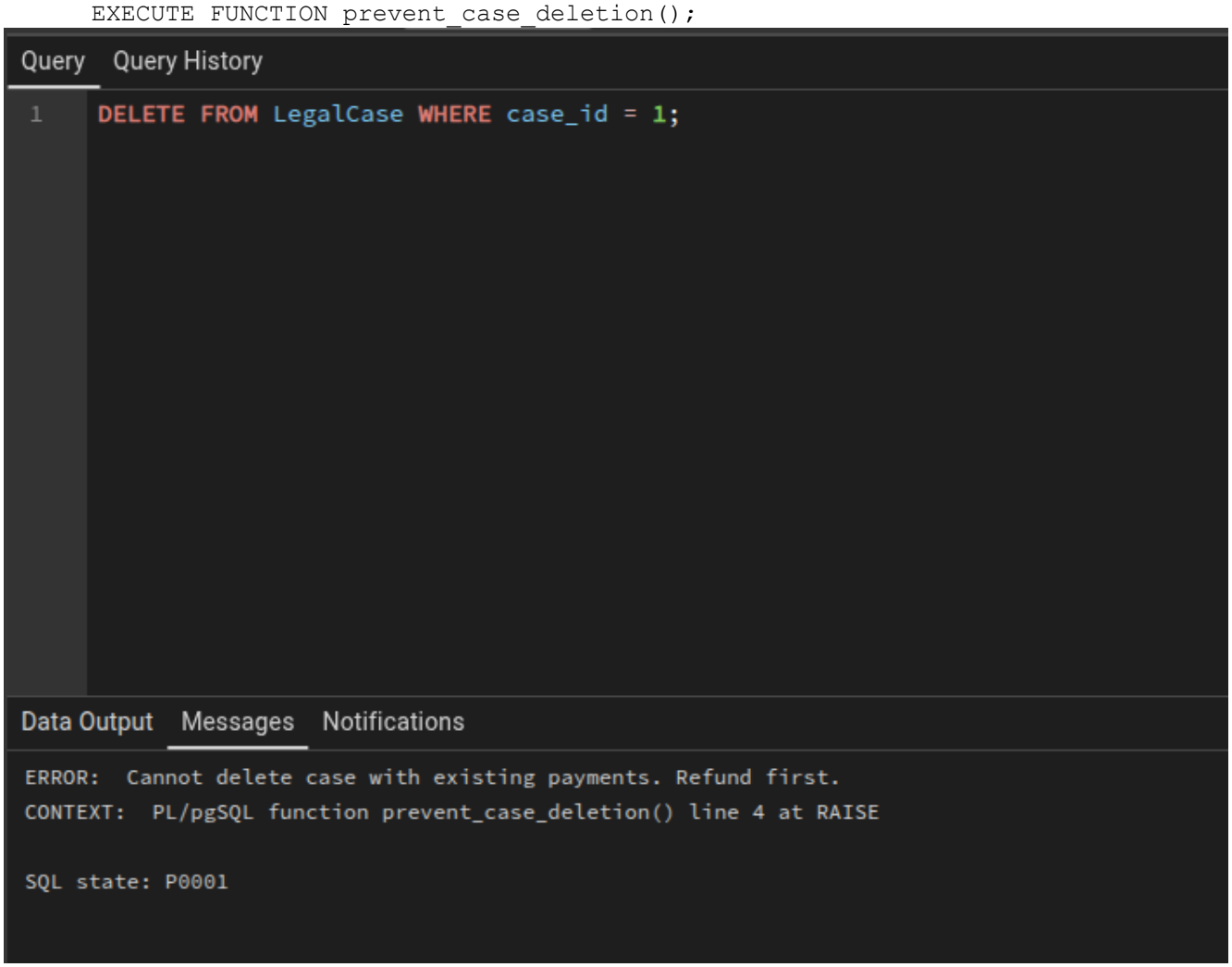


Рисунок 4.3.2 – Перевірка роботи тригера “trg_prevent_delete”

Тригер аудиту зміни зарплати адвоката. Для контролю змін у фінансових параметрах персоналу створюється тригер, який відстежує оновлення поля `salary` у таблиці “Lawyer”. Якщо зарплата змінюється, тригер виводить інформаційне повідомлення (NOTICE) про стару та нову суму.

```
CREATE OR REPLACE FUNCTION log_salary_change()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.salary <> OLD.salary THEN
        RAISE NOTICE 'Salary for lawyer % changed from % to %',
OLD.lawyer_id, OLD.salary, NEW.salary;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_salary_audit
BEFORE UPDATE ON Lawyer
FOR EACH ROW
```

```
EXECUTE FUNCTION log_salary_change();
```

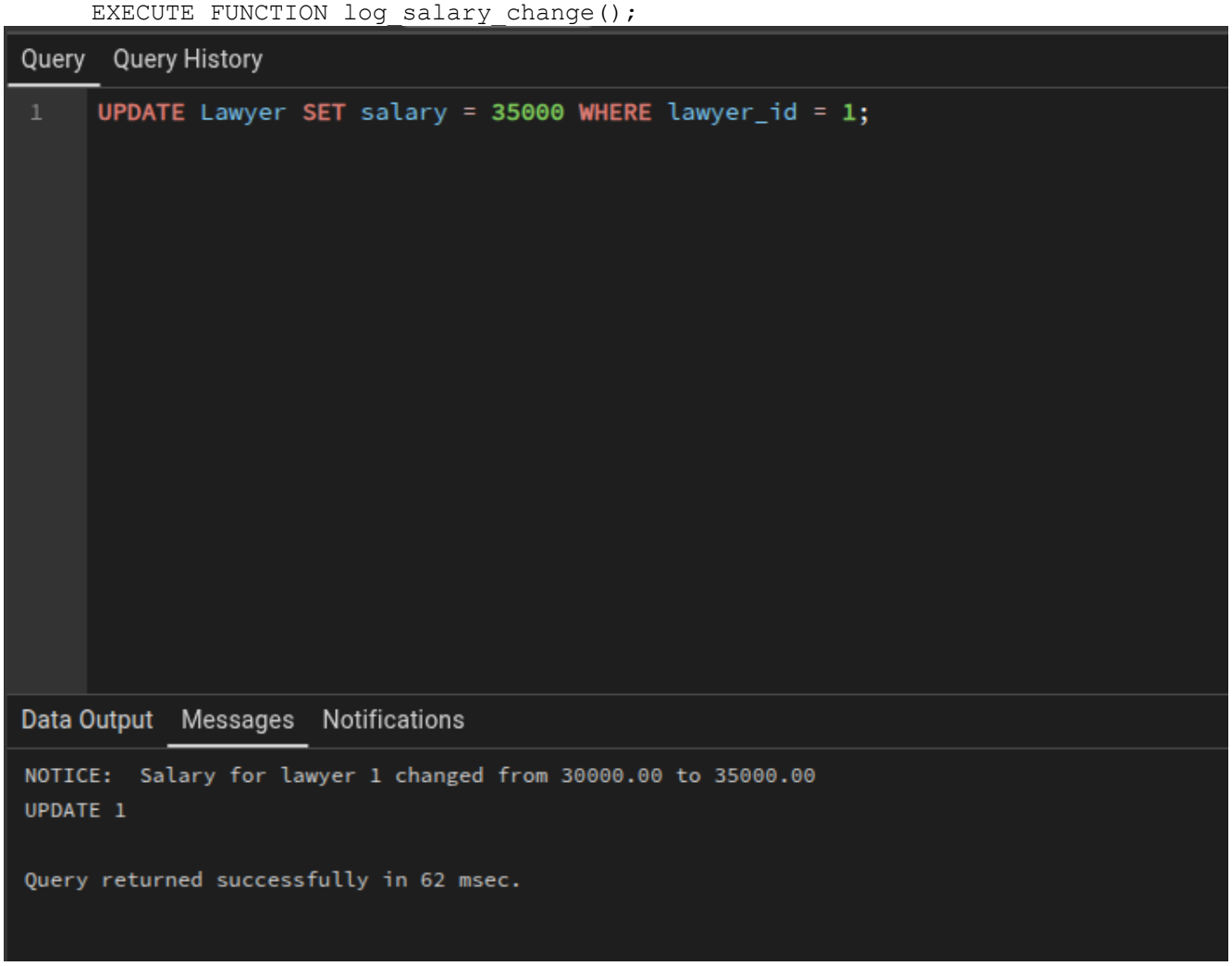


Рисунок 4.3.3 – Перевірка роботи тригера “trg_salary_audit”

Тригер автоматичного закриття справи. Для автоматизації документообігу створено тригер, який відстежує результати судових засідань. Якщо в таблицю “CourtHearing” додається запис із результатом 'Final Decision', статус відповідної справи у таблиці “LegalCase” автоматично змінюється на 'Closed', а дата завершення встановлюється поточною.

```
CREATE OR REPLACE FUNCTION auto_close_case()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.result = 'Final Decision' THEN
        UPDATE LegalCase
        SET status = 'Closed', end_date = CURRENT_DATE
        WHERE case_id = NEW.case_id;

        RAISE NOTICE 'Case % automatically closed due to final hearing decision.', NEW.case_id;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_close_case_after_hearing
```

```
AFTER INSERT OR UPDATE ON CourtHearing
FOR EACH ROW
EXECUTE FUNCTION auto_close_case();
```

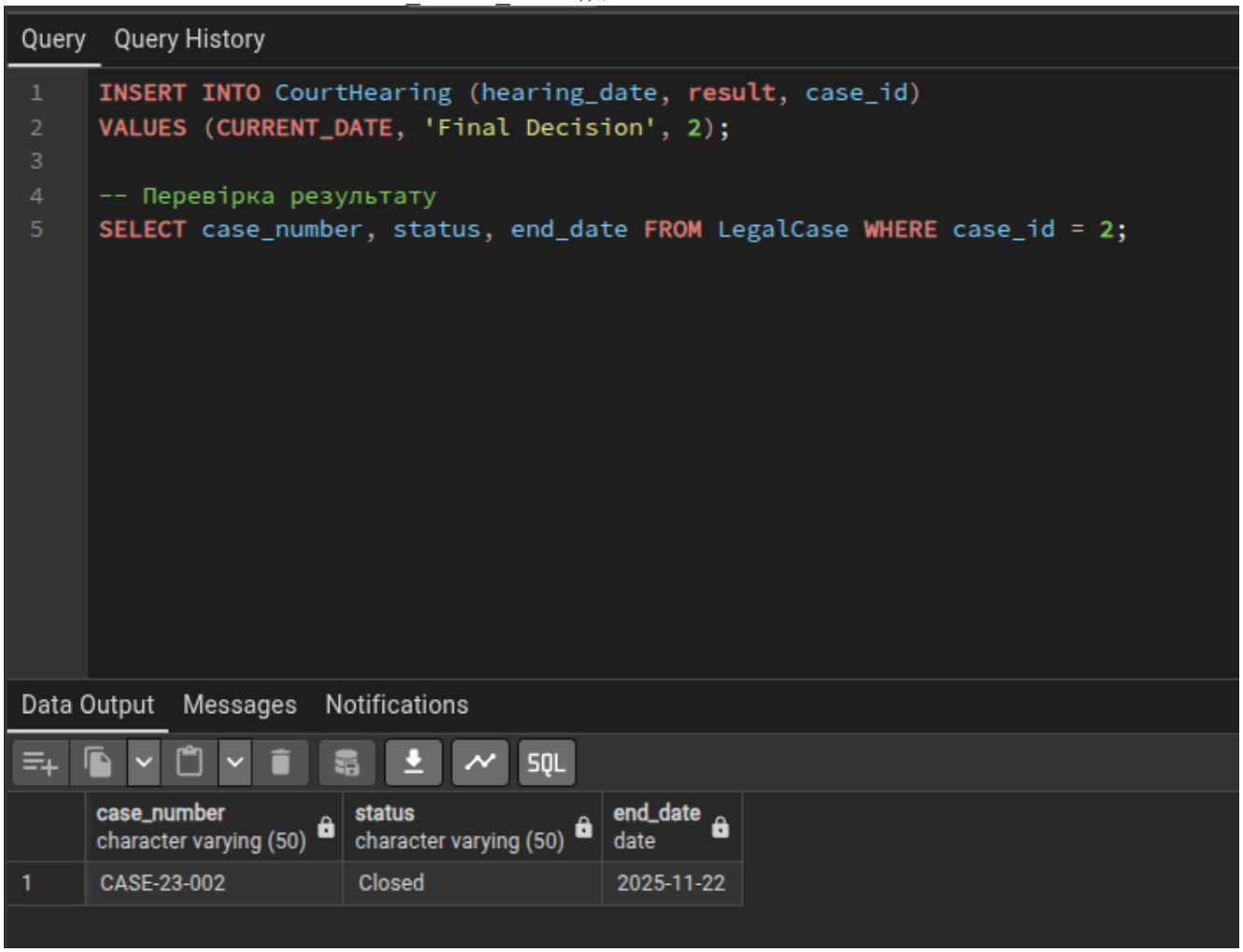


Рисунок 4.3.4 – Результат роботи тригера “trg_close_case_after_hearing”

Тригер контролю повної оплати. Цей тригер спрацьовує після додавання нового платежу. Він перевіряє, чи сума всіх платежів по справі покриває необхідний гонорар (визначений у типі справи). Якщо так, виводиться повідомлення про повну оплату.

```
CREATE OR REPLACE FUNCTION check_full_payment()
RETURNS TRIGGER AS $$
DECLARE
    total_paid DECIMAL;
    required_fee DECIMAL;
    current_case_type INT;
BEGIN
    SELECT case_type_id INTO current_case_type FROM LegalCase WHERE case_id
= NEW.case_id;
    SELECT fee INTO required_fee FROM CaseType WHERE case_type_id =
current_case_type;
    SELECT SUM(amount) INTO total_paid FROM Payment WHERE case_id =
NEW.case_id;

    IF total_paid >= required_fee THEN
```

```
        RAISE NOTICE 'Case % is fully paid.', NEW.case_id;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trg_payment_check ON Payment;

CREATE TRIGGER trg_payment_check
AFTER INSERT ON Payment
FOR EACH ROW
EXECUTE FUNCTION check_full_payment();

DELETE FROM Payment WHERE case_id = 2;

INSERT INTO Payment (case_id, amount, method, status)
VALUES (2, 20000.00, 'Cash', 'Completed');
```

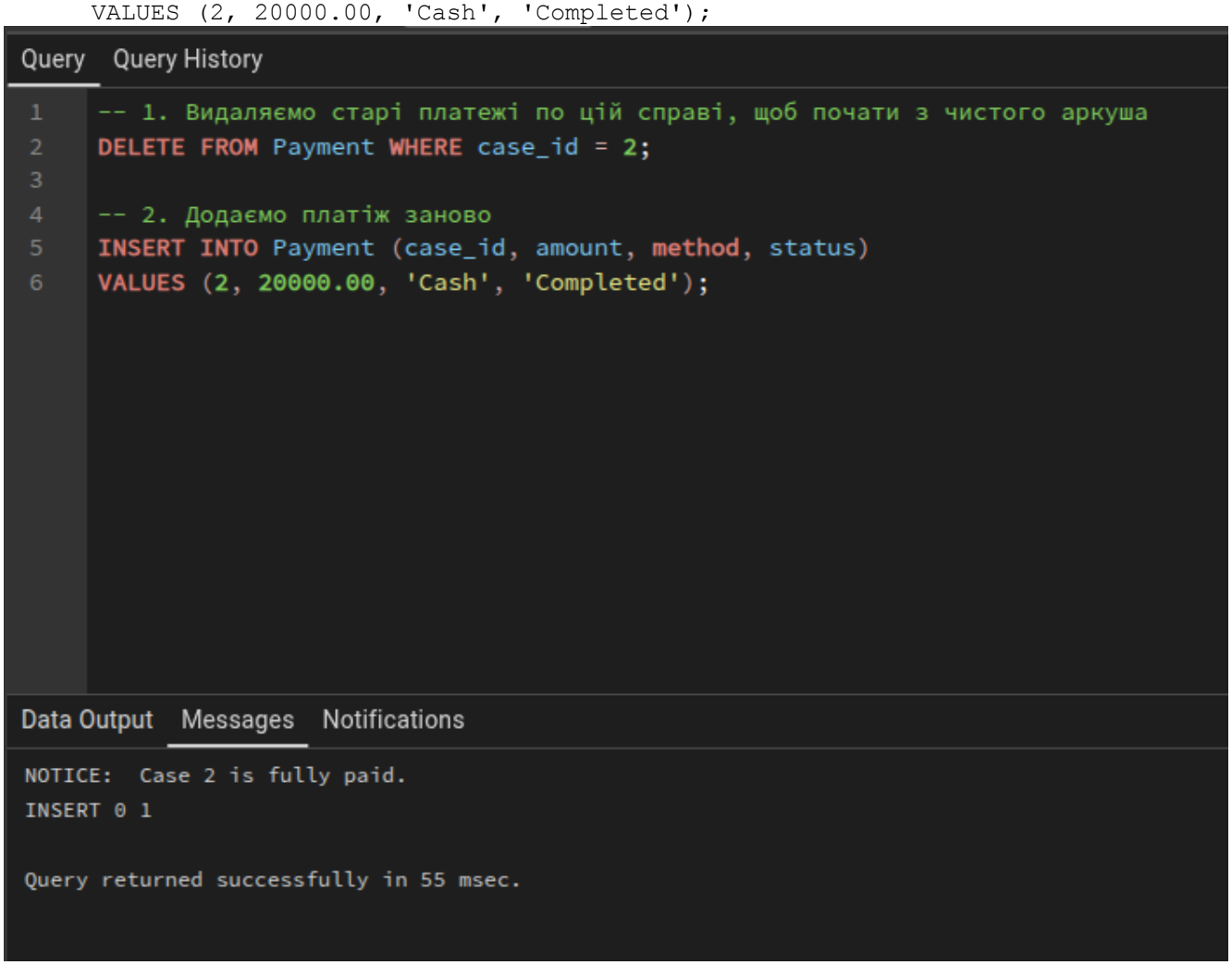


Рисунок 4.3.5 – Перевірка роботи тригера “trg_payment_check”

4.4 Створення збережених процедур (функцій)

У СУБД PostgreSQL збережені процедури та функції дозволяють інкапсулювати бізнес-логіку на стороні бази даних. Вони зберігаються безпосередньо в БД і можуть багаторазово викликатися з клієнтських застосунків або з інших SQL-запитів. У даній базі даних адвокатської контори збережені

функції та процедури використовуються для: розрахунку ефективності роботи адвоката, спрощення процедури додавання платежів та отримання фінансової аналітики.

Функція розрахунку ефективності адвоката. Функція `fn_calculate_effectiveness` приймає ідентифікатор справи та визначає, наскільки успішним був захист. Вона порівнює отриманий клієнтом термін покарання (`received_sentence`) з мінімальними та максимальними межами, передбаченими статтею (`CaseType`).

```
CREATE OR REPLACE FUNCTION fn_calculate_effectiveness(p_case_id INT)
RETURNS TEXT AS $$
DECLARE
    v_max INT;
    v_min INT;
    v_received INT;
BEGIN

    SELECT ct.max_penalty, ct.min_penalty, lc.received_sentence
    INTO v_max, v_min, v_received
    FROM LegalCase lc
    JOIN CaseType ct ON lc.case_type_id = ct.case_type_id
    WHERE lc.case_id = p_case_id;

    IF v_received IS NULL THEN
        RETURN 'Case in progress';
    END IF;

    IF v_received <= v_min THEN
        RETURN 'High Efficiency';
    ELSIF v_received < v_max THEN
        RETURN 'Normal Efficiency';
    ELSE
        RETURN 'Ineffective';
    END IF;
END;
$$ LANGUAGE plpgsql;
```

```
SELECT
    case_number,
    received_sentence,
    fn_calculate_effectiveness(case_id) AS Efficiency_Status
FROM LegalCase
WHERE status = 'Closed';
```

Data Output | Messages | Notifications

SQL

	case_number character varying (50)	received_sentence integer	efficiency_status text
1	CASE-23-001	8	Normal Efficiency
2	CASE-23-003	0	High Efficiency
3	CASE-23-005	0	High Efficiency
4	CASE-23-007	4	Normal Efficiency
5	CASE-23-010	0	High Efficiency
6	CASE-23-002	[null]	Case in progress

Рисунок 4.4.1 – Результат роботи функції “fn_calculate_effectiveness”

Процедура додавання платежу. Для спрощення роботи адміністратора розроблено збережену процедуру `pr_add_payment`, яка реєструє нову оплату. Процедура автоматично встановлює поточну дату та статус 'Pending', що зменшує ризик помилок при ручному введенні.

```
CREATE OR REPLACE PROCEDURE pr_add_payment(
    p_case_id INT,
    p_amount DECIMAL,
    p_method VARCHAR
)
LANGUAGE plpgsql
AS $$
BEGIN
```

```
    INSERT INTO Payment (case_id, amount, method, status, payment_date)
    VALUES (p_case_id, p_amount, p_method, 'Pending', CURRENT_DATE);

    RAISE NOTICE 'Payment added successfully for Case ID: %', p_case_id;
END;
$$;
```

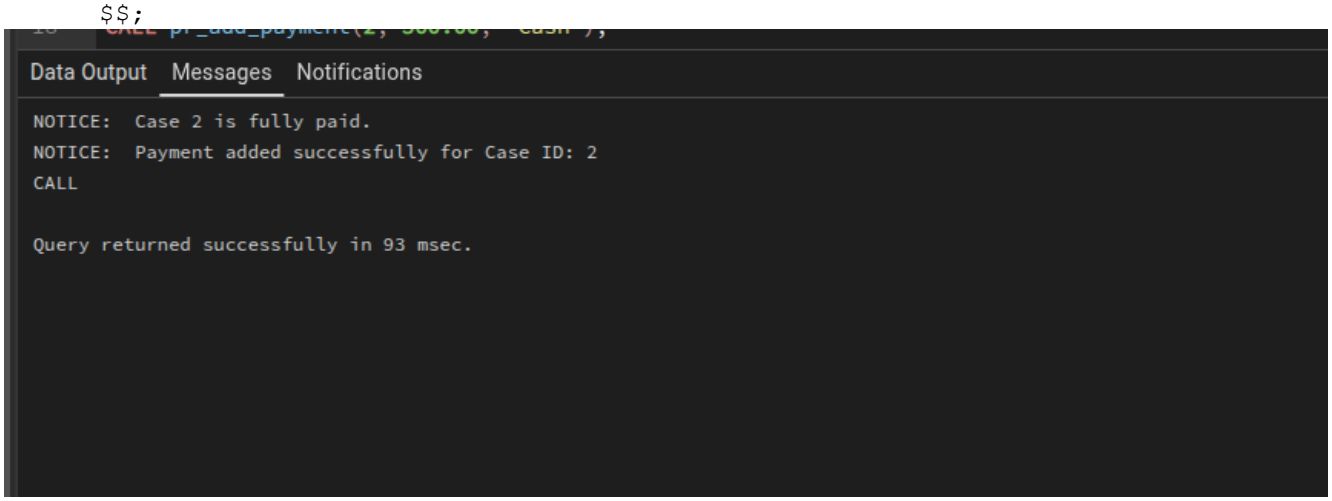


Рисунок 4.4.2 – Перевірка роботи процедури “pr_add_payment”

Функція розрахунку річного прибутку. Функція `fn_get_annual_fees` дозволяє керівництву отримати суму всіх гонорарів за поточний рік. Це дозволяє швидко оцінити фінансовий стан компанії без необхідності писати складні запити з групуванням.

```
CREATE OR REPLACE FUNCTION fn_get_annual_fees()
RETURNS DECIMAL AS $$
DECLARE
    total_sum DECIMAL;
BEGIN
    SELECT COALESCE(SUM(amount), 0)
    INTO total_sum
    FROM Payment
    WHERE EXTRACT(YEAR FROM payment_date) = EXTRACT(YEAR FROM CURRENT_DATE);

    RETURN total_sum;
END;
$$ LANGUAGE plpgsql;
```

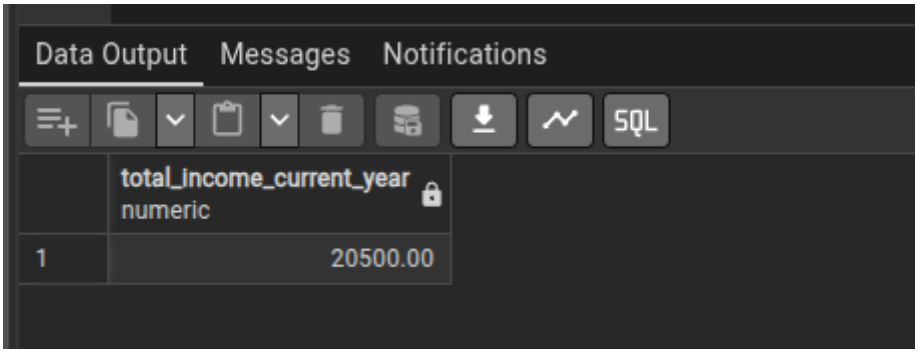


Рисунок 4.4.3 Перевірка роботи функції “fn_get_annual_fees”

Висновки до розділу 4

У цьому розділі було спроєктовано та реалізовано структуру бази даних адвокатської контори: створено основні таблиці, домени, зв’язки та представлення, які забезпечують зручне отримання даних для адвокатів та клієнтів.

Додатково були розроблені тригери та збережені функції/процедури, що автоматизують перевірку коректності даних (валідація дат, контроль оплат), логування важливих подій та розрахунок ефективності роботи персоналу, підвищуючи цілісність і надійність роботи інформаційної системи.

5 МАНІПУЛЮВАННЯ ДАНИМИ

5.1 Оператори відновлення

Для роботи з даними в реляційних базах даних використовуються оператори відновлення (DML — Data Manipulation Language), які дозволяють додавати нові записи, змінювати існуючі та видаляти непотрібні дані. До цієї групи належать оператори INSERT, UPDATE та DELETE.

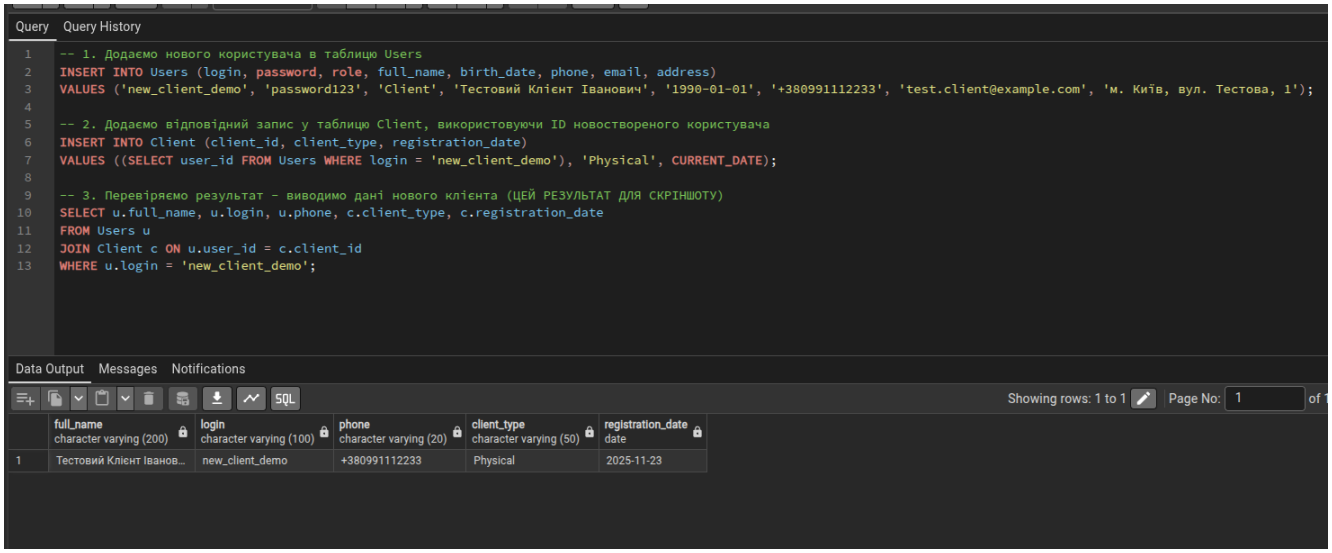


Рисунок 5.1.1 – Перевірка додавання даних

Зміна даних (рисунок 5.1.2). Оператор UPDATE дозволяє змінювати дані існуючого запису. Наприклад, змінемо статус справи на 'Closed' (Закрито) та встановимо дату завершення.

```
UPDATE LegalCase
SET status = 'Closed', end_date = CURRENT_DATE
WHERE case_id = 4;
```

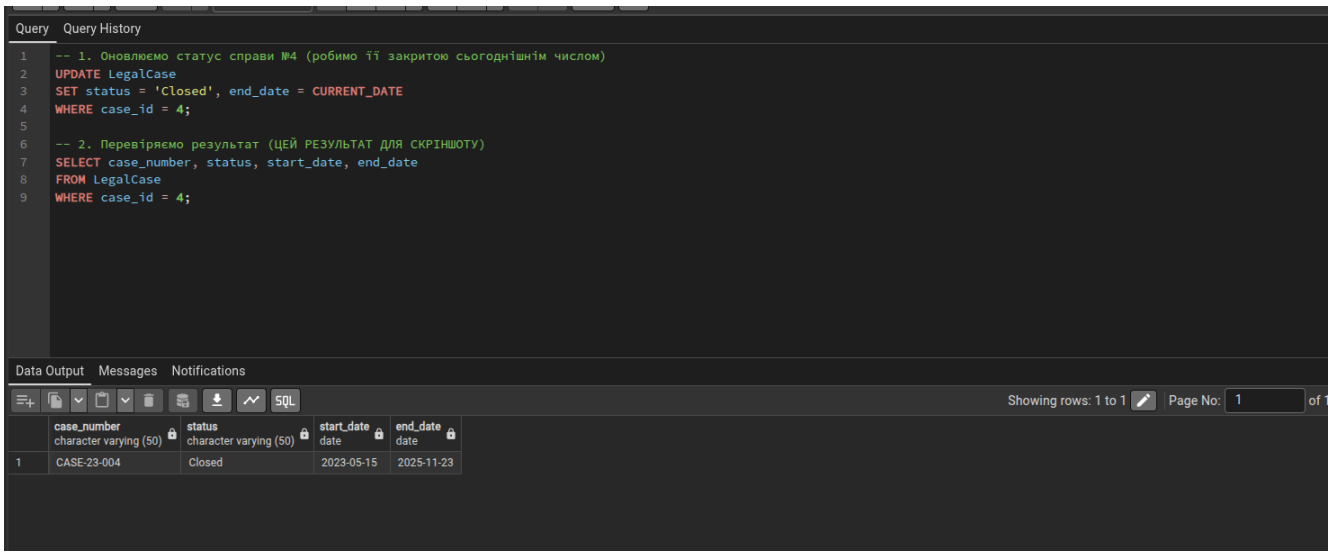


Рисунок 5.1.2 – Перевірка оновлення статусу справи

Видалення даних (рисунок 5.1.3). Оператор DELETE використовується для видалення записів. Видалимо помилково завантажений документ.

```
DELETE FROM Document WHERE document_id = 10;
```

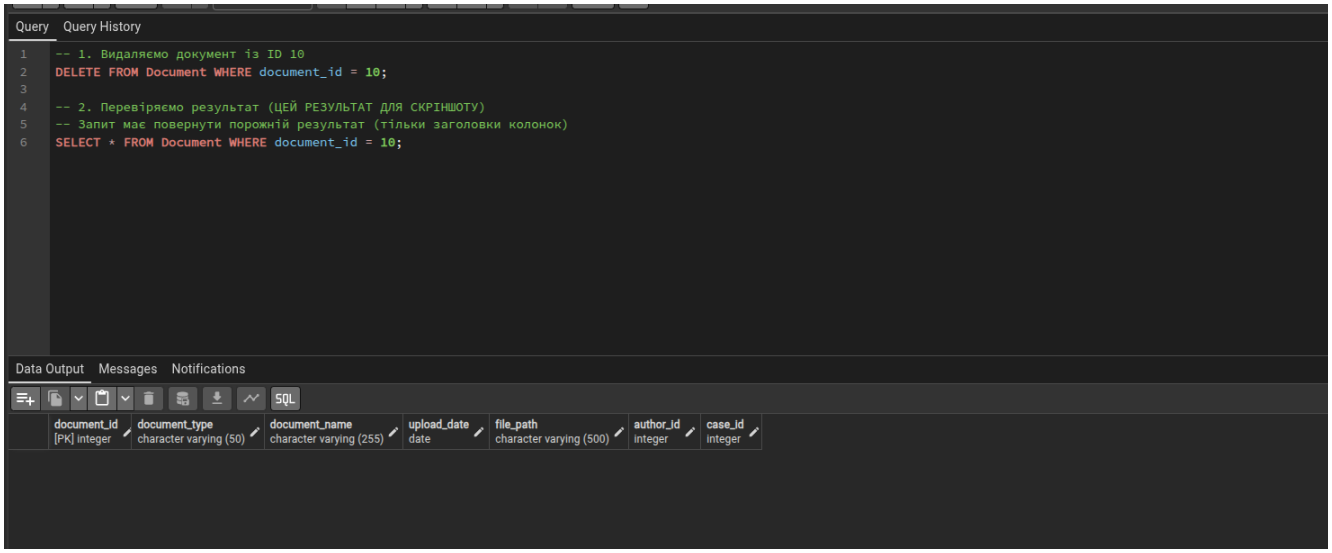


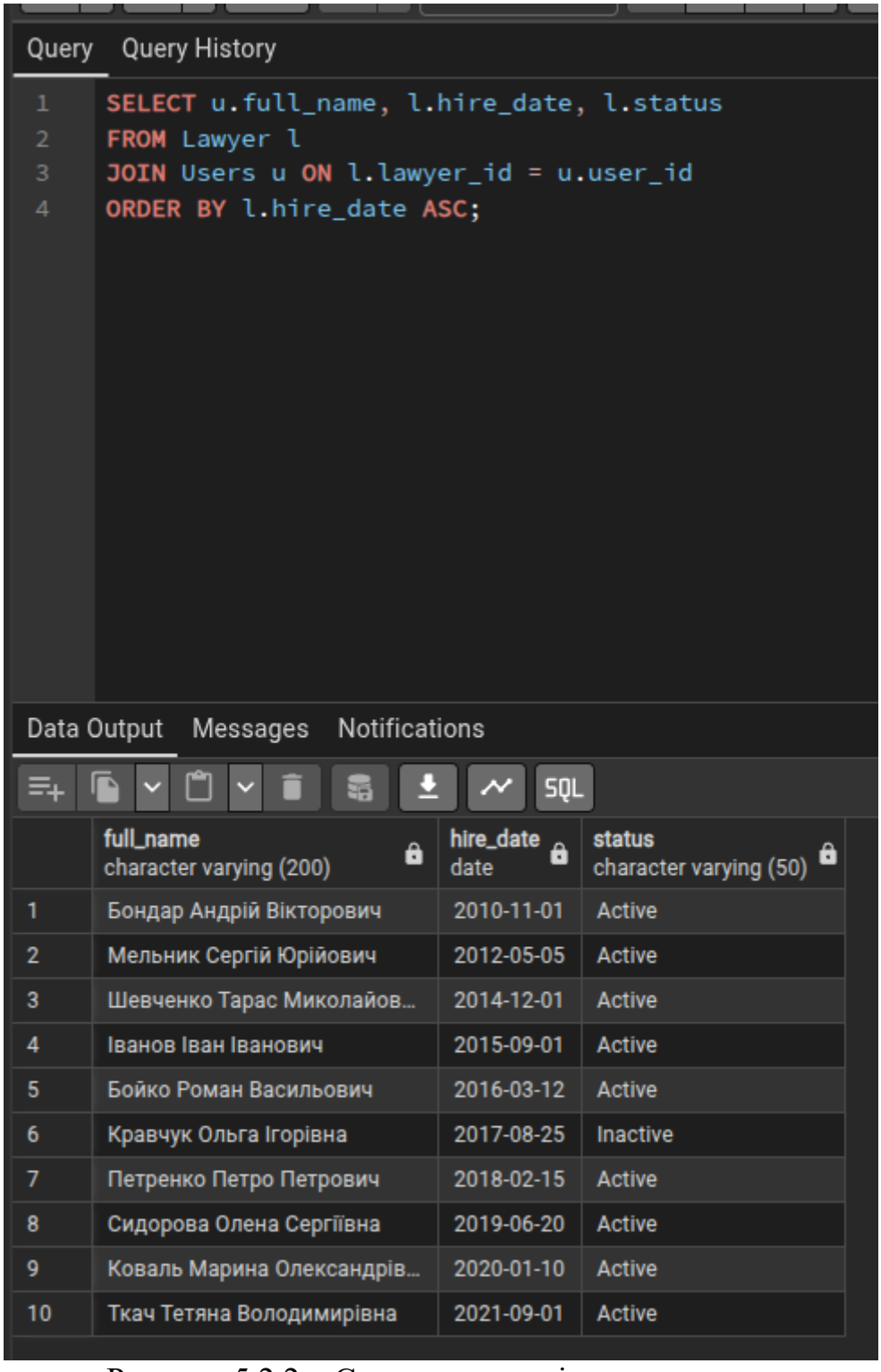
Рисунок 5.1.3 – Перевірка видалення запису

5.2 Оператор вибірки

Оператор вибірки SELECT використовується для отримання інформації з однієї або кількох таблиць. Згідно з прикладом, реалізовано запити різної складності, включаючи групування, підзапити, CTE та перевірку існування.

Запит 1. Вибірка з сортуванням (рисунок 5.2.1). Отримаємо список усіх адвокатів, відсортованих за датою прийняття на роботу (від найдосвідченіших).

```
SELECT full_name, hire_date, status
FROM Lawyer l
JOIN Users u ON l.lawyer_id = u.user_id
ORDER BY hire_date ASC;
```



	full_name character varying (200)	hire_date date	status character varying (50)
1	Бондар Андрій Вікторович	2010-11-01	Active
2	Мельник Сергій Юрійович	2012-05-05	Active
3	Шевченко Тарас Миколайов...	2014-12-01	Active
4	Іванов Іван Іванович	2015-09-01	Active
5	Бойко Роман Васильович	2016-03-12	Active
6	Кравчук Ольга Ігорівна	2017-08-25	Inactive
7	Петренко Петро Петрович	2018-02-15	Active
8	Сидорова Олена Сергіївна	2019-06-20	Active
9	Коваль Марина Олександрів...	2020-01-10	Active
10	Ткач Тетяна Володимирівна	2021-09-01	Active

Рисунок 5.2.2 – Список адвокатів за стажем

Запит 2. З’єднання таблиць JOIN (рисунок 5.2.3). Виведемо список судових справ із назвами типів справ та прізвищами клієнтів.

```
SELECT lc.case_number, ct.description AS type, u.full_name AS client
FROM LegalCase lc
JOIN CaseType ct ON lc.case_type_id = ct.case_type_id
JOIN ClientCase cc ON lc.case_id = cc.case_id
JOIN Users u ON cc.client_id = u.user_id;
```

	case_number character varying (50)	type character varying (500)	client character varying (200)
1	CASE-23-001	Умисне вбивство	Попов Олексій Андрійов…
2	CASE-23-002	Крадіжка	Рябова Інна Михайлівна
3	CASE-23-003	Керування у нетверезому стані	Лисенко Віктор Петрович
4	CASE-23-004	Розірвання шлюбу (Розлучення)	Дмитрук Аліна Сергіївна
5	CASE-23-005	Стягнення заборгованості між юр. особа…	ТОВ "Новак Індастріз"
6	CASE-23-006	Шахрайство	Жук Максим Олегович
7	CASE-23-007	ДТП з тяжкими наслідками	Павлов Денис Іванович
8	CASE-23-008	Поновлення на роботі	ПП "БудПостач"
9	CASE-23-009	Порушення ПДР, що спричинило ДТП	Попов Олексій Андрійов…
10	CASE-23-010	Суперечки щодо меж земельної ділянки	ТОВ "Швидка Логістика"

Рисунок 5.2.3 – Зведена таблиця справ та клієнтів

Запит 3. Вибірка за діапазоном дат (рисунок 5.2.4). Знайдемо всі судові засідання, заплановані на грудень 2025 року, використовуючи оператор `BETWEEN`.

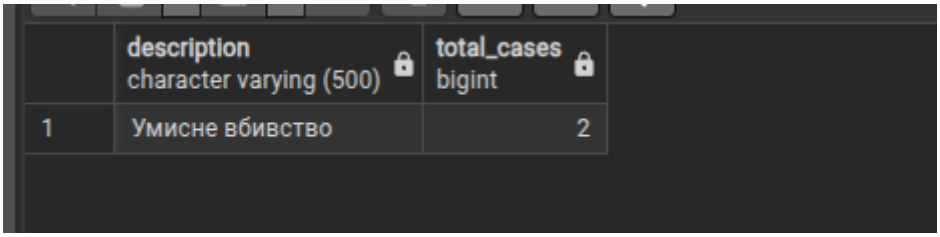
```
SELECT hearing_date, hearing_time, courtroom
FROM CourtHearing
WHERE hearing_date BETWEEN '2025-12-01' AND '2025-12-31';
```

	hearing_date date	hearing_time time without time zone	courtroom character varying (20)
1	2025-12-01	10:00:00	Зал №5
2	2025-12-15	14:30:00	Зал №2

Рисунок 5.2.4 – Розклад засідань на місяць

Запит 4. Агрегатні функції та HAVING (рисунок 5.2.5). Визначимо типи справ, за якими відкрито більше однієї справи. Оператор `HAVING` фільтрує групи після обчислення кількості `COUNT`.

```
SELECT ct.description, COUNT(*) as total_cases
FROM LegalCase lc
JOIN CaseType ct ON lc.case_type_id = ct.case_type_id
GROUP BY ct.description
HAVING COUNT(*) > 1;
```



	description character varying (500)	total_cases bigint
1	Умисне вбивство	2

Рисунок 5.2.5 – Найпопулярніші категорії справ

Запит 5. Підзапит у WHERE (рисунок 5.2.6). Знайдемо адвокатів, чия зарплата вища за середню по всій фірмі.

```
SELECT u.full_name, l.salary
FROM Lawyer l
JOIN Users u ON l.lawyer_id = u.user_id
WHERE l.salary > (SELECT AVG(salary) FROM Lawyer);
```

	full_name character varying (200)	salary numeric (10,2)
1	Бондар Андрій Вікторович	40000.00
2	Мельник Сергій Юрійович	35000.00
3	Шевченко Тарас Миколайов...	32000.00
4	Іванов Іван Іванович	35000.00

Рисунок 5.2.6 – Адвокати з зарплатою вище середньої

Запит 6. Використання конструкції CTE (WITH) (рисунок 5.2.7). Розрахуємо загальну суму платежів по кожній справі у тимчасовій таблиці (CTE), а потім виведемо тільки ті справи, де сума перевищує 10 000 грн. *(Аналог запиту про боржників з твого прикладу)*.

```
WITH case_payments AS (
    SELECT case_id, SUM(amount) as total_paid
    FROM Payment
    GROUP BY case_id
)
SELECT lc.case_number, cp.total_paid
FROM LegalCase lc
JOIN case_payments cp ON lc.case_id = cp.case_id
WHERE cp.total_paid > 10000
ORDER BY cp.total_paid DESC;
```

	case_number character varying (50)	total_paid numeric
1	CASE-23-001	50000.00
2	CASE-23-002	20500.00
3	CASE-23-010	15000.00
4	CASE-23-007	12500.00
5	CASE-23-005	12000.00

Рисунок 5.2.7 – Найприбутковіші справи (використання CTE)

Запит 7. Оператор EXISTS (рисунок 5.2.8). Знайдемо клієнтів, які мають хоча б одну закриту справу ('Closed').

```
SELECT u.full_name, u.phone
FROM Users u
JOIN Client c ON u.user_id = c.client_id
WHERE EXISTS (
    SELECT 1
    FROM ClientCase cc
    JOIN LegalCase lc ON cc.case_id = lc.case_id
    WHERE cc.client_id = c.client_id
    AND lc.status = 'Closed'
);
```

	full_name character varying (200)	phone character varying (20)
1	Попов Олексій Андрійов...	+380991112233
2	ТОВ "Швидка Логістика"	+380445678901
3	Павлов Денис Іванович	+380635554433
4	Рябова Інна Михайлівна	+380675551122
5	Дмитрук Аліна Сергіївна	+380669990011
6	Лисенко Віктор Петрович	+380936667788
7	ТОВ "Новак Індастріз"	+380442223344

Рисунок 5.2.8 – Клієнти із завершеними справами (EXISTS)

Запит 8. Пошук за шаблоном LIKE (рисунок 5.11). Знайдемо всіх клієнтів, які є юридичними особами (у назві є "ТОВ" або "ПП").

```
SELECT full_name, email
FROM Users
WHERE full_name LIKE '%ТОВ%' OR full_name LIKE '%ПП%';
```

	full_name character varying (200)	email character varying (100)
1	ТОВ "Новак Індастріз"	novak_corp@biz.ua
2	ПП "БудПостач"	budpostach@corp.com
3	ТОВ "АгроСвіт"	agromir@farm.ua
4	ТОВ "Швидка Логісти...	fasttrans@log.com

Рисунок 5.2.9 – Пошук юридичних осіб

Запит 9. Використання LEFT JOIN (рисунок 5.12). Виведемо всіх адвокатів і кількість справ, які вони ведуть (навіть якщо справ 0).

```
SELECT u.full_name, COUNT(lc.case_id) as cases_count
FROM Lawyer l
JOIN Users u ON l.lawyer_id = u.user_id
LEFT JOIN LawyerCase lc ON l.lawyer_id = lc.lawyer_id
GROUP BY u.full_name;
```

	full_name character varying (200)	cases_count bigint
1	Сидорова Олена Сергіївна	2
2	Іванов Іван Іванович	2
3	Мельник Сергій Юрійович	1
4	Кравчук Ольга Ігорівна	0
5	Ткач Тетяна Володимирівна	0
6	Шевченко Тарас Миколайов...	0
7	Бойко Роман Васильович	0
8	Петренко Петро Петрович	2
9	Коваль Марина Олександрів...	1
10	Бондар Андрій Вікторович	2

Рисунок 5.2.10 – Навантаження на адвокатів

Запит 10. Складний аналітичний запит (рисунок 5.13). Отримаємо повне досьє: справа, суддя, клієнт, адвокат і сума оплат.

```
SELECT lc.case_number, ch.judge_name, u_cl.full_name as client,
u_lw.full_name as lawyer
FROM LegalCase lc
JOIN CourtHearing ch ON lc.case_id = ch.case_id
JOIN ClientCase cc ON lc.case_id = cc.case_id
JOIN Users u_cl ON cc.client_id = u_cl.user_id
JOIN LawyerCase lwc ON lc.case_id = lwc.case_id
JOIN Users u_lw ON lwc.lawyer_id = u_lw.user_id
LIMIT 5;
```

	case_number character varying (50)	judge_name character varying (150)	client character varying (200)	lawyer character varying (200)	total_paid numeric
1	CASE-23-001	Суддя Вовк	Попов Олексій Андрійов...	Іванов Іван Іванович	50000.00
2	CASE-23-001	Суддя Вовк	Попов Олексій Андрійов...	Іванов Іван Іванович	50000.00
3	CASE-23-001	Суддя Коваль	Попов Олексій Андрійов...	Іванов Іван Іванович	50000.00
4	CASE-23-002	Суддя Іваненко	Рябова Інна Михайлівна	Петренко Петро Петров...	20500.00
5	CASE-23-002	Суддя Ткаченко	Рябова Інна Михайлівна	Петренко Петро Петров...	20500.00

Рисунок 5.2.11 – Зведене досьє по справах

Висновки до розділу 5

У цьому розділі продемонстровано використання операторів маніпулювання даними (INSERT, UPDATE, DELETE та SELECT) для реалізації бізнес-функцій інформаційної системи адвокатської контори.

Наведені приклади запитів показують можливості роботи з даними, включаючи:

- добірку інформації за критерієм та шаблоном (LIKE, BETWEEN);
- групування та агрегацію даних (GROUP BY, HAVING);
- використання підзапитів та розширених конструкцій (CTE, EXISTS);
- об’єднання даних з багатьох таблиць (JOIN).

Реалізовані запити повністю підтримують функціональні вимоги предметної області та дозволяють отримувати гнучку звітність.

6 СТВОРЕННЯ КОРИСТУВАЧІВ І ПРИЗНАЧЕННЯ ПРАВ ДОСТУПУ

Для забезпечення санкціонованого доступу до даних створюються користувачі БД з відповідними правами згідно з потребами. На основі User Story з розділу 1.2 виділено три основні ролі:

– клієнт – має право лише переглядати інформацію про свої справи та засідання (режим "Read Only");

– адвокат – має доступ до списків своїх справ, може вносити результати судових засідань, додавати документи, але не має права видаляти фінансову історію;

– адміністратор – має повні права на керування довідниками, користувачами, фінансами та налаштуваннями системи.

У PostgreSQL ці ролі реалізуються як користувачі бази даних з відповідними привілеями. Спочатку створимо окремих користувачів для адміністратора, адвоката та клієнта (рисунок 6.1).

```
CREATE USER db_admin WITH PASSWORD 'admin_pass';
CREATE USER db_lawyer WITH PASSWORD 'lawyer_pass';
CREATE USER db_client WITH PASSWORD 'client_pass';
```

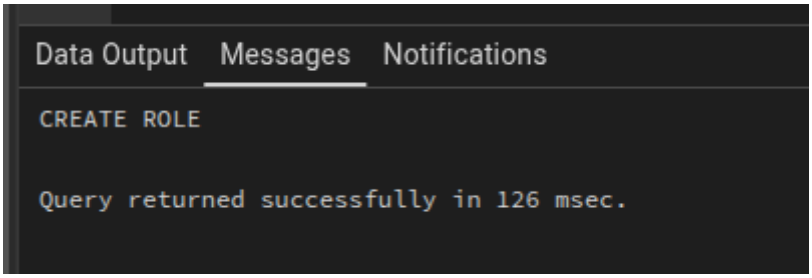


Рисунок 6.1 – Новостворені користувачі у системі

Далі налаштуємо права доступу (GRANT) для кожної ролі.Права адміністратора. Адміністратор адвокатської контори відповідає за повне ведення довідників, реєстрацію нових справ та контроль фінансових потоків. Тому йому надаються повні привілеї на всі таблиці, послідовності, а також право виконання процедур і функцій.

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO db_admin;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO db_admin;
```

```
GRANT ALL PRIVILEGES ON ALL FUNCTIONS IN SCHEMA public TO db_admin;
```

Права адвоката. Згідно з бізнес-логікою, адвокат має переглядати список справ, клієнтів, вносити дані про судові засідання. Йому надаються права SELECT, INSERT, UPDATE на основні таблиці. Однак, право DELETE (видалення) обмежується для забезпечення цілісності архіву.

```
GRANT SELECT, INSERT, UPDATE ON LegalCase, CourtHearing, Document, Payment
TO db_lawyer;
GRANT SELECT ON Client, CaseType, Users TO db_lawyer;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO db_lawyer;
```

Права клієнта. Клієнт повинен бачити інформацію про хід своєї справи, але не може вносити жодних змін у систему.

```
GRANT SELECT ON LegalCase, CourtHearing, View_Active_Clients TO db_client;
REVOKE INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public FROM db_client;
```

Перевірка прав доступу. Для перевірки коректності налаштувань прав доступу виконаємо кілька тестових сценаріїв, використовуючи команду `SET ROLE` для імітації роботи від імені конкретного користувача.

Спроба клієнта змінити дані (очікувано – відмова в доступі). Клієнт не повинен мати можливості видаляти судові справи. Спроба виконати `DELETE` від імені `db_client` має завершитися помилкою (рисунок 6.2), що підтверджує, що клієнт має права лише на читання.

```
SET ROLE db_client;


DELETE FROM LegalCase WHERE case_id = 2;
```

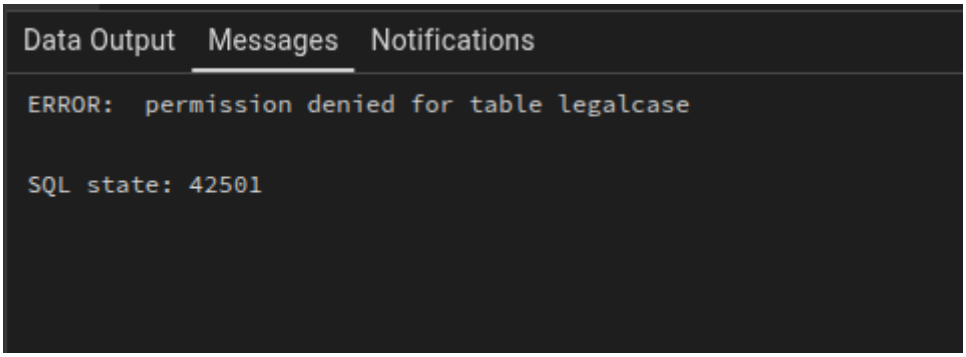


Рисунок 6.2 – Перевірка прав доступу клієнта (заборона видалення)

Внесення результатів засідання адвокатом (успішна операція). Адвокат, навпаки, повинен мати можливість фіксувати роботу. Перевіримо, як користувач `db_lawyer` додає новий запис у таблицю `CourtHearing` (рисунок 6.3).

```
RESET ROLE;

GRANT ALL PRIVILEGES ON TABLE CourtHearing TO db_lawyer;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO db_lawyer;


SET ROLE db_lawyer;

INSERT INTO CourtHearing (hearing_date, result, case_id, courtroom)
VALUES (CURRENT_DATE + 5, 'Scheduled', 2, 'Test Room');
```

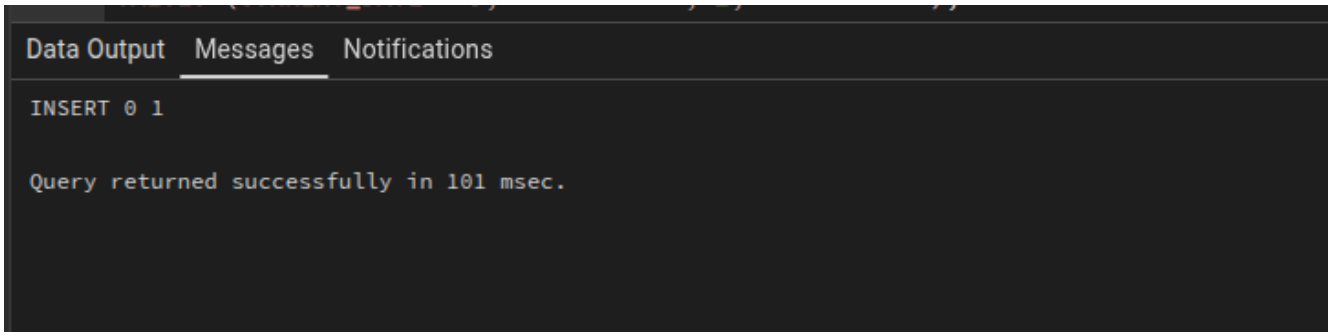


Рисунок 6.3 – Перевірка прав доступу адвоката (успішна вставка)

Спроба адвоката видалити оплату (очікувано – заборонено). Фінансові операції є критичними. Хоча адвокат може бачити платежі або реєструвати їх, він не повинен мати права видаляти історію транзакцій. Спробуємо видалити запис з таблиці `Payment` (рисунок 6.4).

```
SET ROLE db_admin;

CALL pr_add_payment(2, 1000.00, 'Bank Transfer');
```

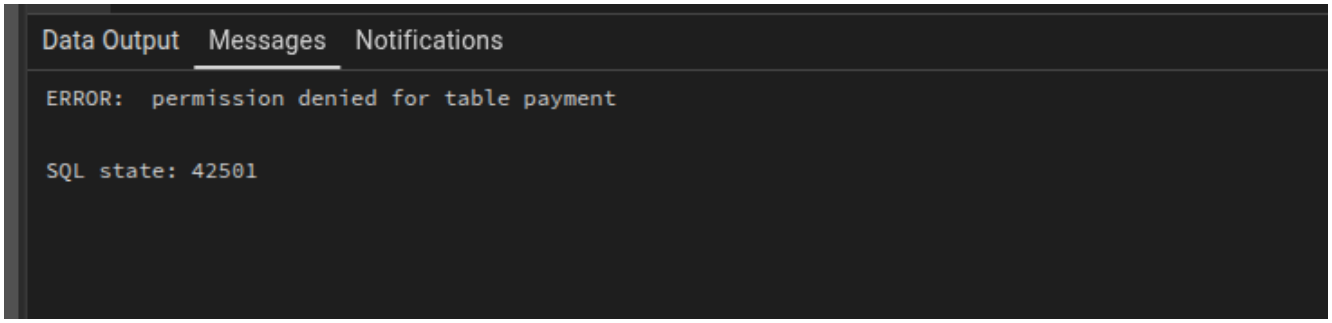


Рисунок 6.4 – Перевірка обмеження прав адвоката

Реєстрація оплати адміністратором (успішна операція). Адміністратор має повні права, зокрема може реєструвати нові платежі через збережену процедуру `pr_add_payment`. Це демонструє можливість виконання процедур адміністратором (рисунок 6.5).

```
GRANT USAGE ON SCHEMA public TO db_admin;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO db_admin;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO db_admin;
GRANT ALL PRIVILEGES ON ALL PROCEDURES IN SCHEMA public TO db_admin;
```

```
SET ROLE db_admin;

CALL pr_add_payment(2, 1000.00, 'Bank Transfer');
```

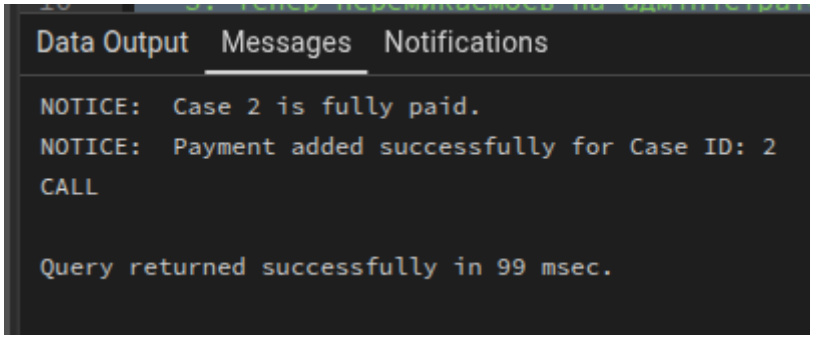


Рисунок 6.5 – Перевірка прав доступу адміністратора

Висновки до розділу 6

У даному розділі були створені користувачі бази даних відповідно до ролей: клієнт, адвокат та адміністратор. Для кожного користувача налаштовано набір привілеїв, що відповідає його функціональним потребам та забезпечує захист даних від несанкціонованого доступу.

Проведені тестові сценарії підтвердили коректність реалізованої моделі доступу: клієнт може лише переглядати інформацію, адвокат має право вносити дані по справах, але обмежений у видаленні фінансів, а адміністратор — повністю керувати системою

ЗАГАЛЬНІ ВИСНОВКИ

У процесі виконання курсової роботи було проведено повний цикл розробки бази даних для інформаційної підтримки діяльності адвокатської контори, починаючи з аналізу предметної області й закінчуючи реалізацією фізичної моделі в середовищі СУБД PostgreSQL. На основі вивчення інформаційних потоків та бізнес-процесів юридичної фірми було визначено основні сутності, їх атрибути та типи зв’язків, сформульовано функціональні вимоги до системи та user story для ключових ролей користувачів: клієнта, адвоката та адміністратора.

У ході проєктування було створено концептуальну модель бази даних у вигляді UML-діаграми класів та ER-діаграми, виконано нормалізацію відношень до третьої нормальної форми, що дозволило зменшити надлишковість даних і забезпечити логічну цілісність. На основі концептуальної моделі побудовано логічну та фізичну моделі, які реалізовано у СУБД PostgreSQL із використанням доменів, послідовностей, первинних і зовнішніх ключів, каскадних операцій та обмежень перевірки даних.

Було створено набір таблиць, представлень, тригерів, збережених функцій і процедур, які забезпечують автоматизацію основних задач інформаційної системи: реєстрації клієнтів та справ, планування судових засідань, обліку гонорарів, розрахунку ефективності роботи адвокатів та формування фінансової звітності. Окрему увагу приділено механізмам підвищення якості даних (перевірка коректності дат початку та завершення справ, контроль повної оплати, аудит змін зарплати, автоматичне закриття справ) та реалізації бізнес-обмежень на рівні бази даних.

Також у роботі реалізовано розмежування прав доступу до даних для різних категорій користувачів за допомогою ролей PostgreSQL. Клієнти мають доступ лише до перегляду інформації про хід своїх справ, адвокати можуть вносити результати судових засідань та працювати з документами, а адміністратор отримує повні права на керування юридичними та фінансовими даними.

Проведені тестові сценарії підтвердили коректність налаштування ролей та привілеїв.

Отже, поставлені в роботі цілі та задачі виконано в повному обсязі: проведено аналіз предметної області, спроєктовано й реалізовано базу даних адвокатської контори, налаштовано механізми підтримки цілісності та якості даних, створено інструменти для автоматизації ключових процесів і звітності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Методичні вказівки до виконання курсової роботи з курсу «Організація баз даних та знань» для студентів всіх форм навчання спеціальності 122 «Комп'ютерні науки» / *Укл.: М.Г. Глава, О.В. Іванов.* Одеса: Одеська політехніка, 2025. 27 с.

Електронний курс на освітній платформі Національного університету «Одеська політехніка» – edu.op.edu.ua «Організація баз даних та знань» для студентів спеціальності 122 Комп'ютерні науки першого бакалаврського рівня навчання. URL : https://el.op.edu.ua/course/view.php?id=695 (дата звернення: 07.11.2025).

PostgreSQL: The world's most advanced open source database. URL: https://www.postgresql.org/ (дата звернення: 07.11.2025).