НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

(повна назва інституту/факультету)

КАФЕДРА інформатики та програмної інженерії

(повна назва кафедри)

**КУРСОВА РОБОТА**

з дисципліни «Бази даних»

(назва дисципліни)

на тему: Підтримка діяльності call-центру

Студента (ки) \_\_2\_\_ курсу \_ІП-33\_\_ групи

спеціальності 121 «Інженерія програмного забезпечення»

\_Шульги Нікіти Сергійовича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_Ліщук Катерина Ігорівна\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка ECTS \_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Ліщук Катерина Ігорівна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Марченко Олена Іванівна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2024 рік

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки

(повна назва)

Кафедра Інформатики та програмної інженерії

(повна назва)

Дисципліна Бази даних

Курс \_\_2\_\_\_ Група \_\_ІП-33\_\_ Семестр \_\_\_3\_\_

**З А В Д А Н Н Я**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

\_\_\_\_\_\_\_Шульзі Нікіта Сергійовичу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема роботи \_\_\_\_\_Підтримка діяльності call-центру\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

керівник роботи \_\_\_\_Ліщук Катерина Ігорівна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

2. Строк подання студентом роботи \_\_\_\_\_22.12.2024\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_завдання на розробку бази даних для підтримки діяльності call-центру\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1) Аналіз предметного середовища

2) Побудова ER-моделі

3) Побудова реляційної схеми з ER-моделі

4) Створення бази даних, у форматі обраної системи управління базою даних

5) Створення користувачів бази даних

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

7) Створення мовою SQL запитів

8) Оптимізація роботи запитів

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Дата видачі завдання\_\_\_\_08.11.2024**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
| 1 | Аналіз предметного середовища | 12.12.2024 |  |
| 2 | Побудова ER-моделі | 17.12.2024 |  |
| 3 | Побудова реляційної схеми з ER-моделі | 17.12.2024 |  |
| 4 | Створення бази даних, у форматі обраної системи управління базою даних | 17.12.2024 |  |
| 5 | Створення користувачів бази даних | 18.12.2024 |  |
| 6 | Імпорт даних з використанням засобів СУБД в створену базу даних | 18.12.2024 |  |
| 7 | Створення мовою SQL запитів | 19.12.2024 |  |
| 8 | Оптимізація роботи запитів | 19.12.2024 |  |
| 9 | Оформлення пояснювальної записки | 22.12.2024 |  |
| 10 | Захист курсової роботи | 25.12.2024 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_**Шульга Н.С.**\_\_\_\_\_\_\_**

(підпис ) (прізвище та ініціали)

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_**Ліщук К.І.**\_\_\_\_\_\_\_\_**

(підпис ) (прізвище та ініціали)

**ЗМІСТ**

[**ВСТУП** 5](#_Toc185613040)

[**1.** **ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА** 6](#_Toc185613041)

[**2.** **АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ПРОДУКТІВ** 10](#_Toc185613042)

[**3.** **ПОСТАНОВКА ЗАДАЧІ** 17](#_Toc185613043)

[**4.** **ПОБУДОВА ER-МОДЕЛІ** 19](#_Toc185613044)

[**5.** **РЕАЛІЗАЦІЯ БАЗИ ДАНИХ** 26](#_Toc185613045)

[**6.** **РОБОТА З БАЗОЮ ДАНИХ** 39](#_Toc185613046)

[6.1 Генератори 39](#_Toc185613047)

[6.2 SQL-запити 39](#_Toc185613048)

[6.3 Функції/Збережені процедури 61](#_Toc185613049)

[6.4 Тригери 79](#_Toc185613050)

[6.5 Представлення 86](#_Toc185613051)

[6.6 Індекси 91](#_Toc185613052)

[**7.** **КОРИСТУВАЧІ СИСТЕМИ** 96](#_Toc185613053)

[**ВИСНОВКИ** 101](#_Toc185613054)

[**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ** 103](#_Toc185613055)

# **ВСТУП**

Сучасний світ неможливо уявити без ефективного обслуговування клієнтів, яке є ключовим елементом для успішної діяльності компаній у різних сферах. Кол-центри виступають важливим інструментом для забезпечення якісного зв’язку між організацією та її клієнтами. Вони дозволяють оперативно вирішувати проблеми, консультувати клієнтів, приймати замовлення та підтримувати лояльність до бренду.

Актуальність теми визначається тим, що швидка обробка запитів клієнтів і доступ до необхідної інформації є критично важливими для підвищення конкурентоспроможності компаній. Правильно спроектована база даних для кол-центру дозволяє не лише зберігати велику кількість даних про дзвінки, клієнтів та їхні запити, але й забезпечує ефективне управління процесами, покращуючи продуктивність працівників і задоволеність клієнтів.

Мета роботи – розробка бази даних для кол-центру, яка дозволить систематизувати та автоматизувати процеси обробки дзвінків і запитів клієнтів

Сфера використання розробки включає компанії, які надають послуги через кол-центри, незалежно від галузі: інтернет-провайдери, банки, служби доставки, технічна підтримка тощо

PostgreSQL – сучасна, дуже популярна СУБД, яка має низку переваг, таких як підтримка великих обсягів даних, висока продуктивність, багатофункціональність та безкоштовна ліцензія. Для цієї роботи було обрано саме її, вона добре підходить для розробки складних реляційних структур і забезпечує гнучкі можливості для подальшої масштабованості системи.

# **ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА**

**Кол-центр або центр обробки дзвінків [1]** - це керований центр, який може бути централізованим або віддаленим, що використовується для прийому або передачі великого обсягу запитів по телефону. Вхідний кол-центр управляється компанією для адміністрування вхідних запитів на підтримку продуктів чи послуг або інформаційних запитів від споживачів. Вихідні кол-центри, як правило, працюють для цілей продажу, таких як телемаркетинг, для залучення благодійних або політичних пожертв, стягнення боргів, маркетингових досліджень, екстрених повідомлень тощо.

**Контакт-центр[2]** - це розширення можливостей кол-центрів, за якого керування взаємодією з клієнтами відбувається через кілька каналів, наприклад електронна пошта, соц-мережі чати і тд. Розроблювана система буде мати можливість нотифікації клієнтів через email, а тому її можна назвати контакт-центром.

**CRM-Система[3] -** це інструмент, який допомагає компаніям організовувати та контролювати всі контакти з клієнтами та потенційними покупцями. Вони дають змогу організувати базу контактів та компаній, сегментувати її, стежити за перебігом угод (будувати воронку продажів), створювати звіти для відстеження прогресу продажу та ефективності команди.

Кол-центр розроблюваної системи буде частиною організації провайдера інтернет послуг і усі бізнес-процеси будуть базуватися тільки на цій ідеї.

Тип кол-центру, взятого за основну розроблюваної системи, буде вхідним. Це означає, що будуть оброблюватися тільки вхідні дзвінки, коли клієнти телефонують, звертаючись з певними проблемами або питаннями. Зворотний зв’язок з клієнтом для повідомлення про закінчення контракту і подібні проблеми організовуватиметься за допомогою пошти.

Вхідними даними в системі є інформація про дзвінки, клієнтів, операторів та запити, яка зберігається в базі даних.

Вихідними даними в системі є результати запитів та операцій, які обробляються базою даних. Це можуть бути списки дзвінків із фільтруванням за датою чи статусом, інформація про клієнтів, активні проблеми або звіти про роботу операторів.

Система має багато бізнес-процесів, кожен з яких важливий для її коректного і безперешкодного функціонування, а тому їх опис є важливим моментом для проєктування системи загалом. Опираючись на те, що організація, частиною якої є розроблювана система кол-центру, це провайдер інтернет послуг, можна виділити наступні бізнес-процеси:

*Реєстрація клієнта*

Коли клієнт звертається у компанію вперше, тобто для підписання контракту з інтернет-провайдером для подальшого надання останнім послуг, його потрібно зареєструвати в системі. Для цього має існувати таблиця Clients, де якраз і зберігається основна інформація про клієнтів.

*Обробка запитів клієнтів під час розмови*

Під час розмови клієнт задає певне питання оператору, останній в цей момент повинен мати змогу швидко знайти відповідь на запит по категорії, для цього мають існувати відповідні таблиці Issues(для збереження запиту користувача), Issue\_Categories(збереження усіх категорій запитів клієнтів), Issue\_Solutions(потрібна оператору таблиця, в якій, по категорії, можна знайти різноманітні відповіді на запити).

*Управління дзвінками клієнтів*

Цей бізнес-процес може включати моніторинг, коригування інформації і інші дії пов’язані з дзвінками клієнтів.

*Управління запитами та їх архівація*

Запити клієнтів мають 2 стани «Оброблюється» та «Закрито», перший говорить про те, що запит ще в обробці, бо оператор не зміг на місці допомогти клієнту, а другий значить, що клієнт отримав допомогу. Запити потрібно зберігати для подальшого звітування чи інших проблем з якими стикається компанія.

*Управління графіком операторів*

Звісно, кожен оператор повинен мати визначений графік роботи, щоб заздалегідь розуміти о котрій та в який день йому виходити на роботу. В системі має бути таблиця Operator\_Schedule, яка якраз слугуватиме для того, щоб визначати графік роботи операторів. В ній визначатимуться дні тижня, по яким він працює, а також початок і кінець зміни. Графік зазвичай формується при прийомі на роботу, але також може змінюватись під час роботи через певні події або прохання робітника.

*Управління телефонами операторів*

Як і в випадку з графіками, телефони визначаються при прийнятті оператора на роботу. Вони, очевидно, слугують для того, щоб робітник мав змогу отримувати вхідні дзвінки та робити вихідні.

*Управління звітністю кол-центру*

Звіти – невід’ємна частина будь якої системи, вона є чи не основним доказом діяльності будь-яких організацій. Таблиця Operators\_Report відповідатиме за основні звіти по кожному оператору за певний місяць, в ній мають бути вказані загальний час у дзвінках, дата та зп за поточний місяць. Також можливі інші організаційні звіти

Висновки до розділу

В даному розділі був наведений детальний опис об’єкту дослідження, було визначено в загальному що таке кол-центр, чим він відрізняється від контакт-центру, були наведені вхідні та вихідні дані, а також описані основні бізнес-процеси використання даних.

# **АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ПРОДУКТІВ**

Кол-центри відіграють ключову роль у забезпеченні якісного обслуговування клієнтів, надаючи можливість оперативно реагувати на запити, вирішувати проблеми та підтримувати комунікацію. У сучасному світі автоматизація та ефективність кол-центрів стали одними з основних критеріїв успішності організацій, незалежно від того, чи є вони частиною великої компанії, чи аутсорсинговими структурами.

На ринку програмного забезпечення існує велика кількість рішень, які покликані автоматизувати та оптимізувати роботу кол-центрів. Ці системи можуть виконувати різноманітні завдання: від маршрутизації дзвінків і запису розмов до інтеграції з CRM-системами та аналізу даних.

У цьому розділі буде розглянуто кілька програмних продуктів, що використовуються для автоматизації кол-центрів, їх функціональні можливості, переваги та недоліки. Для аналізу обрано тільки аутсорсингові рішення, оскільки доволі складно знайти опис кол-центру, який є частиною більшої організації, бо ключову увагу ніколи на цьому не загострюють.

Отже, для аналізу обрано eCall, ОНЛАЙНМАРКЕТ та contactis.

**eCall [4]** – це міжнародний контактний центр, який уже 18 років забезпечує надійне опрацювання вхідних дзвінків і продажів, а також надає високоякісні послуги колл-центру по всьому світу.

*Основні послуги E-Call:*

Вхідні дзвінки:

* Компанія приймає дзвінки від клієнтів своїх замовників (бізнесів).
* Серед основних задач: реєстрація звернень, консультації, технічна підтримка, прийом заявок.
* Вхідні дзвінки можуть бути організовані з урахуванням пріоритетності або рівня критичності.

Вихідні дзвінки:

* Організація вихідних кампаній: опитування, маркетингові акції, нагадування про борги чи підписки.
* Можливість масштабування дзвінків для великих кампаній.

Підтримка через різні канали:

* Робота не лише з дзвінками, а й з іншими засобами комунікації: email, чати, месенджери.
* Ціль — забезпечення повного супроводу клієнта незалежно від каналу звернення.

Моніторинг та оцінка роботи операторів:

* Запис дзвінків для аналізу якості.
* Надання статистичних даних замовникам про ефективність комунікацій.
* Використання систем автоматичного розподілу дзвінків між операторами.

На рисунку 2.1 показані послуги eCall, показані на головній сторінці сайту



Рисунок 2.1 – послуги кол-центру eCall, показані на головній сторінці сайту

З переваг даної компанії можна виділити складання повного звіту для замовників, який допоможе їм проаналізувати правильність роботи будь-яких процесів в своїй організації, а також можливість нагадування клієнтам про будь-які події за допомогою пошти, чати, месенджери.

В компанії, що працює більше 18 років, складно знайти будь-які недоліки в системі, але усі недоліки, що можна придумати, загалом зводяться до того, що компанія аутсорс, що значить, що вона не є частиною іншої більшої компанії, а тому не всі бізнеси можуть передати свої завдання, а також можливість втрати контролю над якістю обслуговування, якщо замовник не взаємодіє достатньо активно.

Аутсорсинговий кол-центр **ОНЛАЙНМАРКЕТ [5]** спеціалізується на наданні рішень для бізнесів, що потребують підтримки в обробці вхідних та вихідних дзвінків, а також автоматизації рутинних процесів.

Основні послуги ОНЛАЙНМАРКЕТ:

*Вхідні дзвінки:*

* Фіксація звернень клієнтів із врахуванням специфіки бізнесу.
* Графік роботи, налаштований під потреби замовника (можливість цілодобової роботи).
* Організація "гарячих ліній" та обробки заявок.

*Вихідні дзвінки:*

* Проведення опитувань.
* Обробка замовлень та інших клієнтських запитів.

*Додаткові рішення:*

* Віртуальні IP-АТС, голосові меню IVR.
* Розсилки: голосові повідомлення, SMS, email.
* Інтеграція з CRM та інші автоматизовані рішення.
* Номери 0-800 і Skype-лінії для зручності зв'язку.

На рисунку 2.2 показані послуги ОНЛАЙНМАРКЕТ, показані на головній сторінці сайту

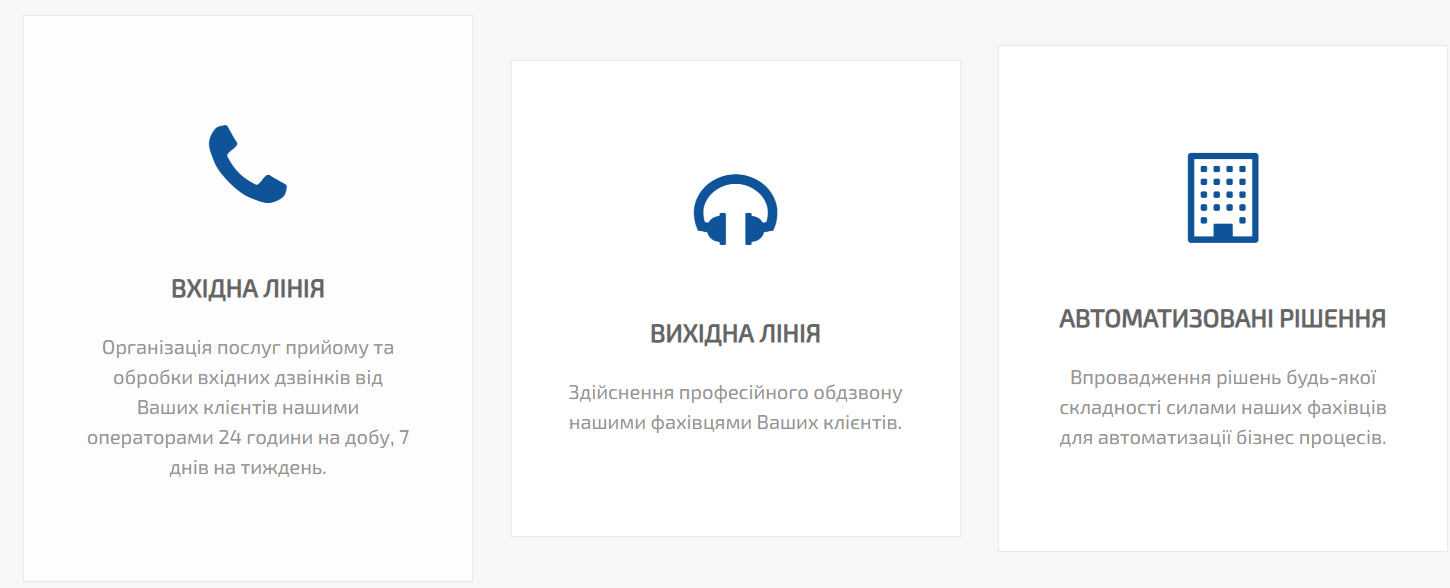


Рисунок 2.2 – послуги ОНЛАЙНМАРКЕТ

З переваг комплексне обслуговування клієнтів через кілька каналів, що дозволяє організувати ефективну комунікацію з користувачами. Це зручно для компаній, які хочуть інтегрувати різні способи взаємодії з клієнтами (телефонні дзвінки, чати, електронну пошту) в одну систему. Також компанія забезпечує звітування про ефективність роботи кол-центру, що дозволяє замовникам оцінювати результати та вдосконалювати процеси на основі даних

Основним недоліком є те, що ОНЛАЙНМАРКЕТ працює як аутсорсинговий кол-центр, що означає, що компанії можуть втратити контроль над процесами обслуговування та якістю виконання задач, якщо не взаємодіють із постачальником на достатньому рівні. Також може бути проблема з безпекою даних через зберігання інформації в сторонніх системах, що потребує додаткових заходів для захисту персональних даних клієнтів

Аутсорсинговий контакт-центр **CONTACTIS [6]** на ринку з 2003 року. Колл-центр знаходиться в Києві і веде свою діяльність на території України вже більше 21 років.

Основні послуги цього коль-центру наведені на Рисунках 2.3, 2.4 і 2.4



Рисунок 2.3 – основні послуги Contactis, показані на головній сторінці сайту(1)



Рисунок 2.4 – основні послуги Contactis, показані на головній сторінці сайту(2)



Рисунок 2.5 – основні послуги Contactis, показані на головній сторінці сайту(3)

Contactis пропонує автоматизовану обробку дзвінків і запитів, що дозволяє швидко обслуговувати великий потік клієнтів. Їхній функціонал може бути інтегрований із системами CRM, що сприяє зберіганню й аналізу даних клієнтів для кращого планування роботи кол-центру.

З недоліків можливі труднощі в адаптації їхньої автоматизації до унікальних процесів певної компанії. Крім того, передача даних до аутсорсингового кол-центру потребує додаткових заходів безпеки для уникнення витоків чи втрати конфіденційної інформації.

Висновки до розділу

Аналізуючи аналоги кол-центрів можна зробити висновок, що зазвичай усюди присутні кілька каналів для обслуговування клієнтів, що значить що цей фактор є важливим під час створенні своєї системи кол-центрів. Мається на увазі, що варто додати якийсь додатковий інструмент для зв’язку з клієнтом. Тому буде вибраний спосіб відправки повідомлень клієнту через mail, це значить, що варто додати відповідну таблицю, яка відповідадиме за збереження відправлених повідомлень.

Також варто зазначити, що в будь-яких аутсорсингових процесах існує ризик витоку чи втрати конфіденційної інформації. Для зниження цих ризиків необхідно впроваджувати надійні заходи інформаційної безпеки. Але коли кол-центр існує всередині іншої компанії, тобто є її частиною, цьому набагато легше запобігти.

# **ПОСТАНОВКА ЗАДАЧІ**

В сучасних умовах ефективна робота кол-центрів є невід'ємною складовою успішного функціонування компаній, що орієнтуються на взаємодію з клієнтами. Автоматизація обробки інформації, оперативний аналіз даних та забезпечення безпеки стають ключовими аспектами роботи таких систем. Основною метою цієї роботи є створення бази даних, яка стане фундаментом для автоматизації та оптимізації роботи кол-центру.

Головна задача полягає у розробці структури бази даних, яка відповідатиме потребам кол-центру, дозволятиме зберігати інформацію про дзвінки, операторів, клієнтів, результати обробки запитів та забезпечуватиме ефективну взаємодію з аналітичними інструментами. База даних має бути побудована таким чином, щоб підтримувати гнучке масштабування та адаптацію до змін у вимогах бізнесу.

Проаналізувавши існуючі програмні продукти, було зроблено висновок про те, що майже кожна використовує різні канали зв’язку(не тільки використовуючи мобільний зв’язок), тому було прийнято рішення додати можливість нотифікації клієнтів через пошту, для чого варто буде створити окрему таблицю для збереження шаблонів листів, які будуть відправлятися.

До майбутньої бази даних можна придумати багато вимог, таких як забезпечення безпеки, щоб не було витоку інформації, або її знищення. Це реалізуватиметься шляхом створення ролей, де доступ до бд у кожного користувача обмежений їх правами, а тому ніхто, без прав, не зможе користуватися нею.

Майбутня база даних має доволі швидко виконувати запити і повертати результат, бо дзвінок виконується у реальному часі, а клієнт не бажає довго чекати на відповідь.

Також, варто сказати, що під час змін якихось важливих даних, бд має правильно реагувати на них. Наприклад, видалення даних, які пов’язані з іншими даними, не має викликати помилки, або додавання данних правильного типу, але неправильного формату, навпаки, має створити виключення. Те ж саме і під час оновлення даних.

Висновки до розділу

Постановка задачі є важливою частиною створення будь-якої системи. Конкретно в даному випадку було визначено основну мету, задачу та вимоги до майбутньої бази даних, зокрема це забезпечення безпеки, швидкість виконання запитів, коректне реагування на зміни даних та обробка виключень. Усі ці вимоги спрямовані на створення надійної, ефективної та безпечної системи, яка відповідатиме потребам кол-центру і буде готова до масштабування в майбутньому.

# **ПОБУДОВА ER-МОДЕЛІ**

Модель «сутність-зв'язок» (ER-модель)  — модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель — це метамодель даних, тобто засіб опису моделей даних.

**Опис виділених інформаційних об’єктів:**

1. Сутність Mail

Атрибути:

* id
* Topic
* Text

Відношення:

* (One-to-Many) до сутності Mail\_Client, створений для організації зв’язку (Many-to-Many)

1. Сутність Clients

Атрибути:

* id
* Name
* Surname
* Email
* Phone
* Address

Відношення:

* (One-to-Many) до сутності Mail\_Client, створений для організації зв’язку (Many-to-Many)
* (One-to-Many) до сутності Calls. Один клієнт може багато разів телефонувати в компанії

1. Сутність Mail\_Client

Атрибути:

* id
* Client\_id
* Mail\_id
* Date

Відношення:

* (Many-to-One) до сутностей Mail та Clients, організація зв’язку багато до багатьох для сутностей Mail і Clients

1. Сутність Calls

Атрибути:

* id
* Client\_id
* Operator\_id
* Call\_date
* Duration

Відношення:

* (Many-to-One) до сутності Clients, багато звінків можуть надходити від одного клієнта
* (Many-to-One) до сутності Operator, багато дзвінків відносяться до одного оператора
* (One-to-Many) до сутності Issues, в одному дзвінку клієнт може робити різні запити

1. Operators

Атрибути:

* id
* Name
* Surname
* Email
* Hire\_date
* Hourly\_rate

Відношення:

* (One-to-Many) до сутності Calls, один оператор може мати багато дзвінків
* (One-to-One) до сутності Operator\_Schedule, один оператор може мати один графік роботи
* (One-to-One) до сутності Operator\_Phones, один оператор може мати один робочий номер телефону
* (One-to-Many) до сутності Operators\_Report, один оператор може мати декілька звітів(за різні місяці)

1. Сутність Operator\_Schedule

Атрибути:

* id
* Operator\_id
* Days\_of\_week
* Shift\_start
* Shift\_end

Відношення:

* (One-to-One) до сутності Operator, один графік роботи відноситься до одного оператора

1. Сутність Operators\_Phones

Атрибути:

* id
* Operator\_id
* Phone

Відношення:

* (One-to-One) до сутності Operators, один робочий номер телефони відноситься до одного оператора

1. Сутність Issues

Атрибути:

* id
* Issue\_Category\_id
* Description
* Status
* Call\_id

Відношення:

* (Many-to-One) до сутності Issues\_Categories, багато запитів можуть мати одну категорію
* (Many-to-One) до сутності Calls, багато запитів можуть відноситись до одного дзвінку

1. Сутність Issue\_Categories

Атрибути:

* id
* Name

Відношення:

* (One-to-Many) до сутності Issue\_Solutions, певна категорія запитів може відноситись до декількох можливих рішень(відповідей).
* (One-to-Many) до сутності Issues, одна категорія може відноситись до багатьох запитів

1. Сутність Issue\_Solutions

Атрибути:

* id
* Issue\_Category\_id
* Solution

Відношення:

* (Many-to-One) до сутності Issue\_Categories, багато рішень можуть відноситися до однієї категорії

1. Сутність Operators\_Report

Атрибути:

* id
* Operator\_id
* Calls\_duration
* Total\_salary
* Date

Відношення:

* (Many-to-One) до сутності Operators, багато звітів можуть відноситись до одного оператора

На рисунку 4.1 показана спроектована ER-модель графічно у вигляді діаграми

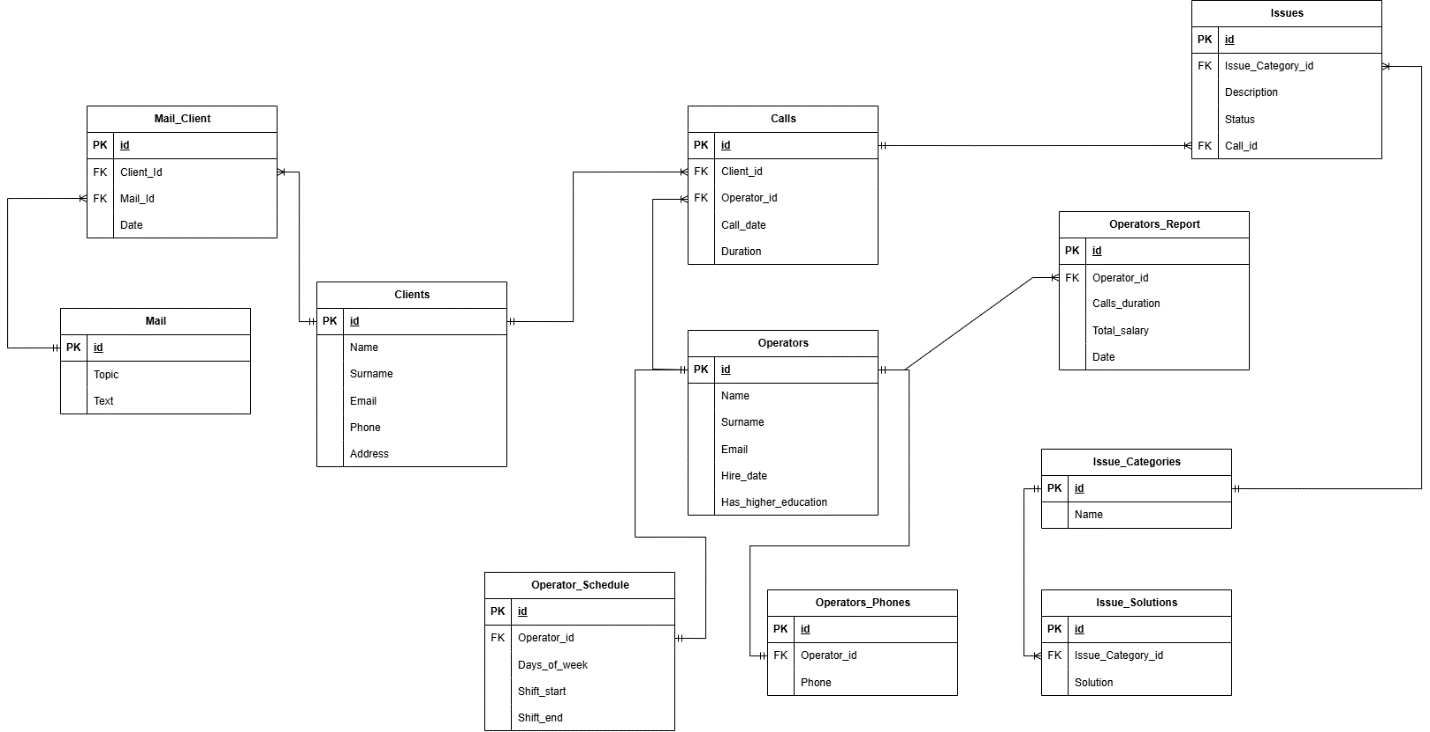


Рисунок 4.1 – спроектована ER-модель

Висновки до розділу

ER-модель дозволяє чітко ідентифікувати основні сутності системи, їх атрибути та зв'язки між ними. Це важливий крок у проектуванні бази даних, оскільки на основі ER-моделі визначаються всі необхідні таблиці, а також їх взаємозв'язки, що забезпечує правильну структуру та ефективну роботу системи

# **РЕАЛІЗАЦІЯ БАЗИ ДАНИХ**

* 1. Обгрунтування вибору СУБД

**PostgreSQL[7]** - ͏це ͏одна з найпотужніших і найбільш відомих с͏исте͏м баз да͏них з відкритим кодом,͏ яка͏ є наслі͏дком͏ більше 35 років роботи. Її вибирають через поєднання надійності, гнучкос͏ті, функцій ͏та ефективності. Однією з великих переваг є підтри͏мка стандартів SQL͏ і можливі͏сть роботи с велик͏ими об’є͏мами даних що кр͏итично важливо для задач кол-центру де потрібн͏о швидко обробляти великі потоки інформації

Серед п͏ричин вибору PostgreSQL тако͏ж треба відзнач͏ити її велику ͏гнучкість і підт͏римку додатків, що дозволяє налашту͏вати систему під с͏пецифічні потре͏би б͏ізнесу. Наприклад, доповнення PostGIS дає сильні можливості роботи з гео͏даними, що може бути корисн͏о у деяких ви͏падках аналітики. Також, Post͏gre͏SQL підтримує складні пр͏авила для бізнесу через функції, тригер͏и та перегляди,͏ що забезпечують вис͏окий рівень кастомізації͏ б͏ази даних.  
 Важливою є система безпеки PostgreSQL, яка забезпечує надійний контроль доступу до даних за допомогою ролей і прав доступу. Це дає можливість створити чітку ієрархію доступу до бази даних у кол-центрі, захистивши конфіденційну інформацію клієнтів і забезпечивши дотримання стандартів захисту даних. PostgreSQL також підтримує транзакції на рівні ACID, що гарантує цілісність даних навіть у випадках системних збоїв.

Додатково PostgreSQL є мультиплатформною системою, що дозволяє її запуск на різних операційних системах, включаючи Windows, Linux та macOS. Це особливо зручно для компаній, які використовують різноманітне програмне забезпечення. Також слід згадати про активну спільноту користувачів і розробників PostgreSQL, яка забезпечує постійну підтримку, регулярні оновлення та випуск нових функціональних можливостей.

Таким чином, вибір PostgreSQL для бази даних у межах цього проекту є обґрунтованим і відповідає сучасним вимогам до продуктивності, безпеки та масштабованості інформаційних систем.

* 1. Опис структур бази даних у табличному вигляді

Таблиця Mail призначена для зберігання даних про шаблони листів, що відправлятимуться по пошті клієнтам. Структура таблиці наведена в таблиці 5.1

Таблиця 5.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім'я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int |  | PK | ідентифікатор | primary key |
| Topic | varchar | 100 |  | Тема повідомлення |  |
| Text | varchar | 1000 |  | Текст повідомлення |  |

SQL-запит для створення таблиці:

CREATE TABLE Mail  
(  
 id SERIAL PRIMARY KEY,  
 Topic VARCHAR(100),  
 Text VARCHAR(1000),  
);

Таблиця Clients призначена для зберігання даних про клієнтів, що звертаються до компанії. Структура таблиці наведена в таблиці 5.2

Таблиця 5.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім'я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | serial |  | PK | Унікальний ідентифікатор | primary key |
| Name | varchar | 50 |  | Ім'я |  |
| Surname | varchar | 50 |  | Прізвище |  |
| Email | varchar | 50 |  | Електронна пошта | unique |
| Phone | varchar | 50 |  | Телефон | unique, not null |
| Address | varchar | 100 |  | Адреса |  |

SQL-запит для створення таблиці:

CREATE TABLE Clients  
(  
 id SERIAL PRIMARY KEY,  
 Name VARCHAR(50),  
 Surname VARCHAR(50),  
 Email VARCHAR(50) UNIQUE,  
 Phone VARCHAR(50) UNIQUE NOT NULL,  
 Address VARCHAR(100)  
);

Таблиця Mail\_Client призначена для зберігання даних про те, яке повідомлення якому клієнту і коли було відправлено. Структура таблиці наведена в таблиці 5.3

Таблиця 5.3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім'я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | serial |  | PK | Унікальний ідентифікатор | primary key |
| Client\_id | int |  | FK | Ідентифікатор клієнта | not null, foreign key (Clients.id) on delete cascade |
| Mail\_id | int |  | FK | Ідентифікатор листа | not null, foreign key (Mail.id) on delete cascade |
| Date | date |  |  | Дата відправлення | default now() |

SQL-запит для створення таблиці:

CREATE TABLE Mail\_Client  
(  
 id SERIAL PRIMARY KEY,  
 Client\_id INT NOT NULL,  
 Mail\_id INT NOT NULL,

Date DATE,  
 FOREIGN KEY (Client\_id) REFERENCES Clients (id) ON DELETE CASCADE,  
 FOREIGN KEY (Mail\_id) REFERENCES Mail (id) ON DELETE CASCADE  
);

Таблиця Operators призначена для зберігання даних про операторів. Структура таблиці наведена в таблиці 5.4

Таблиця 5.4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім'я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | serial |  | PK | Унікальний ідентифікатор | primary key |
| Name | varchar | 50 |  | Ім'я оператора |  |
| Surname | varchar | 50 |  | Прізвище оператора |  |
| Email | varchar | 50 |  | Електронна пошта оператора | unique |
| Hire\_date | date |  |  | Дата найму | default CURRENT\_DATE |
| Has\_higher\_education | bit |  |  | Чи має оператор вищу освіту |  |

SQL-запит для створення таблиці:

CREATE TABLE Operators  
(  
 id SERIAL PRIMARY KEY,  
 Name VARCHAR(50),  
 Surname VARCHAR(50),  
 Email VARCHAR(50) UNIQUE,  
 Hire\_date DATE DEFAULT CURRENT\_DATE,  
 Has\_higher\_education BIT  
);

Таблиця Operator\_Schedule призначена для зберігання графіків роботи операторів. Структура таблиці наведена в таблиці 5.5

Таблиця 5.5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім'я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | serial |  | PK | Унікальний ідентифікатор | primary key |
| Operator\_id | int |  | FK | Ідентифікатор оператора | foreign key (Operators.id) on delete cascade, not null |
| Days\_of\_week | varchar |  |  | Дні тижня, коли працює оператор |  |
| Shift\_start | time |  |  | Початок зміни |  |
| Shift\_end | time |  |  | Кінець зміни |  |

Обмеження для декількох полів:

* Перевірка на те, щоб початок зміни був раніше кінця, і тривалість зміни не перевищувала 9 годин.

SQL-запит для створення таблиці:

CREATE TABLE Operator\_Schedule  
(  
 id SERIAL PRIMARY KEY,  
 Operator\_id INT NOT NULL,  
 Days\_of\_week VARCHAR,  
 Shift\_start TIME,  
 Shift\_end TIME,  
 FOREIGN KEY (Operator\_id) REFERENCES Operators (id) ON DELETE CASCADE,  
 CHECK (Shift\_start < Shift\_end AND (Shift\_end - Shift\_start) < INTERVAL '9 hours')  
);

Таблиця Operators\_Phones призначена для зберігання даних про робочі телефони операторів. Структура таблиці наведена в таблиці 5.6

Таблиця 5.6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім'я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int |  | PK | ідентифікатор | primary key |
| Operator\_id | int |  | FK | ідентифікатор оператора | foreign key (Operators.id) on delete set null |
| Phone | varchar | 50 |  | телефон оператора | unique, not null |

SQL-запит для створення таблиці:

CREATE TABLE Operators\_Phones  
(  
 id SERIAL PRIMARY KEY,  
 Operator\_id INT,  
 Phone VARCHAR(50) UNIQUE NOT NULL,  
 FOREIGN KEY (Operator\_id) REFERENCES Operators (id) ON DELETE SET NULL  
);

Таблиця Calls призначена для зберігання даних про дзвінки. Структура таблиці наведена в таблиці 5.7

Таблиця 5.7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім'я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int |  | PK | ідентифікатор виклику | primary key |
| Operator\_id | int |  | FK | ідентифікатор оператора | foreign key (Operators.id) on delete set null |
| Client\_id | int |  | FK | ідентифікатор клієнта | foreign key (Clients.id) on delete set null |
| Date | date |  |  | дата виклику |  |
| Duration | time |  |  | тривалість виклику |  |

SQL-запит для створення таблиці:

CREATE TABLE Calls  
(  
 id SERIAL PRIMARY KEY,  
 Operator\_id INT,  
 Client\_id INT,  
 Date DATE DEFAULT CURRENT\_DATE  
 Duration TIME,  
 FOREIGN KEY (Operator\_id) REFERENCES Operators (id) ON DELETE SET NULL,  
 FOREIGN KEY (Client\_id) REFERENCES Clients (id) ON DELETE SET NULL  
);

Таблиця Issue\_Categories призначена для зберігання даних про категорії запитів. Структура таблиці наведена в таблиці 5.8

Таблиця 5.8

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім'я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int |  | PK | ідентифікатор категорії | primary key |
| Name | varchar | 30 |  | назва категорії |  |

SQL-запит для створення таблиці:

CREATE TABLE Issue\_Categories  
(  
 id SERIAL PRIMARY KEY,  
 Name VARCHAR(30)  
);

Таблиця Issue\_Solutions призначена для зберігання даних про рішення(відповіді до)запитів. Структура таблиці наведена в таблиці 5.9

Таблиця 5.9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім'я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int |  | PK | ідентифікатор рішення | primary key |
| Issue\_Category\_Id | int |  | FK | категорія проблеми | foreign key (Issue\_Categories.id) on delete cascade, not null |
| Solution | varchar | 1000 |  | опис рішення |  |

SQL-запит для створення таблиці:

CREATE TABLE Issue\_Solutions  
(  
 id SERIAL PRIMARY KEY,  
 Issue\_Category\_Id INT NOT NULL,  
 Solution VARCHAR(1000),  
 FOREIGN KEY (Issue\_Category\_Id) REFERENCES Issue\_Categories (id) ON DELETE CASCADE  
);

Таблиця Issues призначена для зберігання даних про запити. Структура таблиці наведена в таблиці 5.10

Таблиця 5.10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім'я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int |  | PK | ідентифікатор проблеми | primary key |
| Issue\_Category\_Id | int |  | FK | категорія проблеми | foreign key (Issue\_Categories.id) on delete set null |
| Description | varchar | 100 |  | опис проблеми |  |
| Status | varchar | 30 |  | статус проблеми |  |
| Call\_id | int |  | FK | ідентифікатор дзвінка | foreign key (Calls.id) on delete set null |

SQL-запит для створення таблиці:

CREATE TABLE Issues  
(  
 id SERIAL PRIMARY KEY,  
 Issue\_Category\_Id INT,  
 Description VARCHAR(100),  
 Status VARCHAR(30),  
 Call\_id INT,  
 FOREIGN KEY (Issue\_Category\_Id) REFERENCES Issue\_Categories (id) ON DELETE SET NULL,  
 FOREIGN KEY (Call\_id) REFERENCES Calls (id) ON DELETE SET NULL  
);

Таблиця Operators\_Report призначена для зберігання даних про звіти про операторів за місяць. Структура таблиці наведена в таблиці 5.11

Таблиця 5.11

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім'я поля** | **Тип даних** | **Розмір** | **Ключ** | **Опис** | **Обмеження** |
| id | int |  | PK | ідентифікатор звіту | primary key |
| Operator\_id | int |  | FK | ідентифікатор оператора | foreign key (Operators.id) on delete set null |
| Calls\_duration | interval |  |  | тривалість дзвінків |  |
| Total\_salary | decimal |  |  | загальна зарплата оператора |  |
| Date | date |  |  | дата звіту |  |

SQL-запит для створення таблиці:

CREATE TABLE Operators\_Report  
(  
 id SERIAL PRIMARY KEY,  
 Operator\_id INT NOT NULL,  
 Calls\_duration INTERVAL,  
 Total\_salary DECIMAL,  
 Date DATE,  
 FOREIGN KEY (Operator\_id) REFERENCES Operators (id) ON DELETE CASCADE  
);

На рисунку 5.1 зображена схема даних, побудована засобами СУБД postgresql

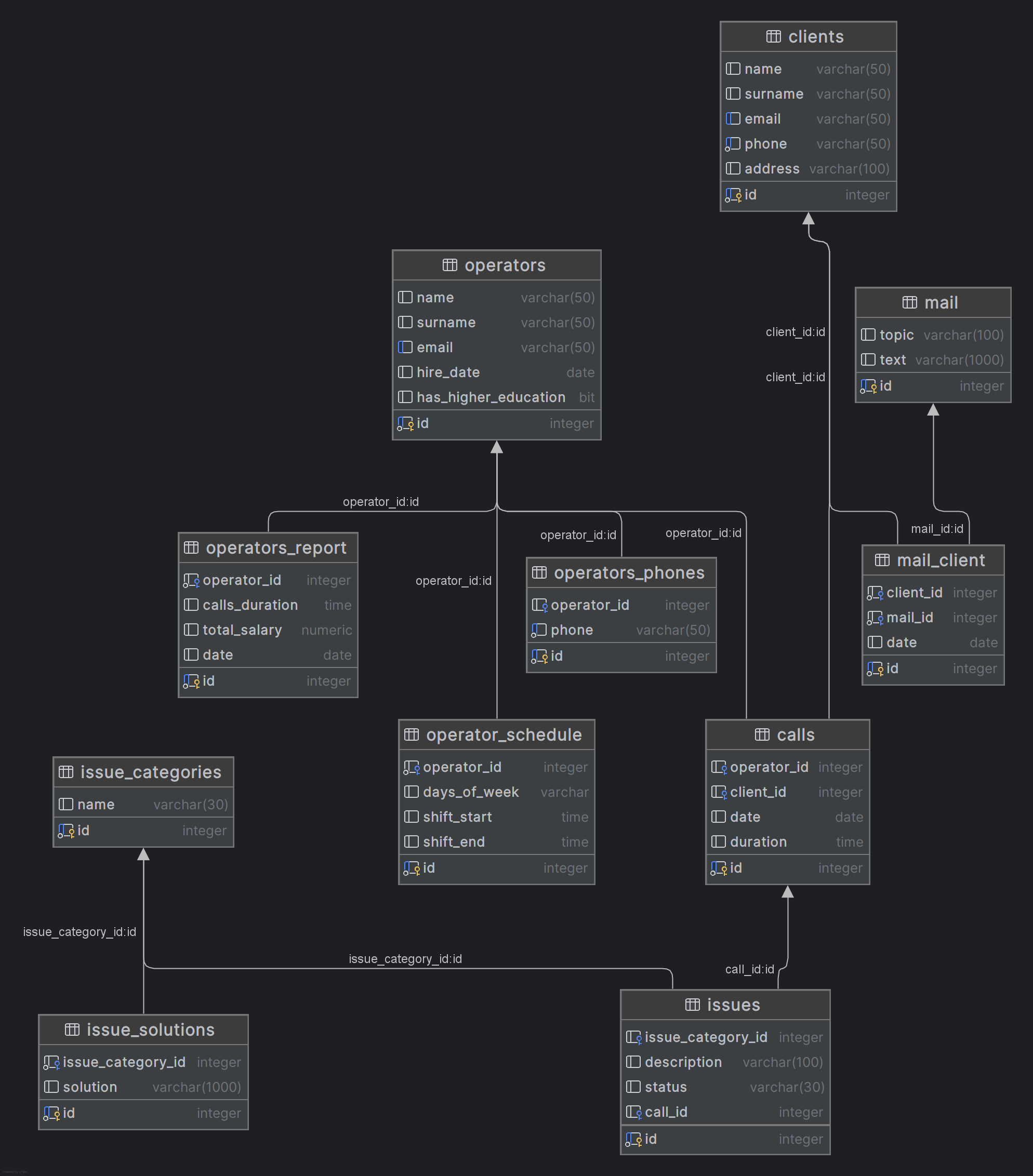


Рисунок 5.1 – Схема даних, побудована засобами СУБД postgreSQL

Висновки до розділу

В ході написання данного розділу було описано майбутні таблиці, опираючись на ER-діаграму, а також створено sql скрипти. Це один з основних моментів під час розробки бази даних, бо на даний момент вже можна створити основну стуктури майбутньої бази даних, а саме основні таблиці, зв’язки між ними та обмеження для полів.

# **РОБОТА З БАЗОЮ ДАНИХ**

В цьому розділі будуть описані тексти генераторів, збережених процедур та функцій, тригерів, представлень, SQL Select-запитів різного виду та результати оптимізації.

## Генератори

Під час створення таблиць був використаний генератор SERIAL. Він автоматично створює унікальні значення для стовпця під час вставки значень. Його було використано в кожній таблиці бази даних на полі унікального ідентифікатора.

Приклад SQL-запиту з використанням SERIAL:

CREATE TABLE issue\_categories  
(  
 id SERIAL PRIMARY KEY,  
 Name VARCHAR(30)  
);

В запиті для стовпця id(унікального ідентифікатора) під час створення нових записів формується нове значення.

Також в роботі був використаний генератор RANDOM. Він генерує випадкові числа в діапазоні від 0 до 1.

Приклад SQL-скрипту, де використовувався RANDOM

CREATE OR REPLACE PROCEDURE callIn()  
 LANGUAGE plpgsql  
AS $$  
BEGIN  
 INSERT INTO calls (operator\_id, client\_id, date, duration)  
 WITH random\_data AS (  
 SELECT  
 FLOOR(1 + (14) \* RANDOM())::INT AS operator\_id,  
 (DATE '2023-01-01' + (RANDOM() \* (DATE '2024-11-30' - DATE '2023-01-01'))::INT) AS random\_date  
 FROM generate\_series(1, 20000)  
 )  
 SELECT  
 rd.operator\_id,  
 FLOOR(1 + (94) \* RANDOM())::INT AS client\_id,  
 rd.random\_date,  
 INTERVAL '10 seconds' + (RANDOM() \* (INTERVAL '10 minutes' - INTERVAL '10 seconds')) AS duration  
 FROM random\_data rd  
 WHERE is\_operator\_working(rd.operator\_id, rd.random\_date);  
END;  
$$;

В запиті для стовпця створення запису в таблиці calls генеруються випадковим чином id оператора, id клієнта, дата та тривалість.

## SQL-запити

a)

SELECT solution FROM issue\_solutions  
WHERE issue\_category\_id = (  
 SELECT id FROM issue\_categories  
 WHERE name='Технічні проблеми'  
);

**Призначення**: Цей запит призначений для отримання рішень з таблиці issue\_solutions, які належать до категорії "Технічні проблеми" з таблиці issue\_categories.

**Опис**: Запит спочатку шукає в таблиці issue\_categories id категорії з назвою "Технічні проблеми", а потім використовує це id для отримання всіх рішень з таблиці issue\_solutions, що належать до цієї категорії.

**Бізнес-процеси**: Обробка запитів клієнтів під час розмови

Результат наведений на рисунку 6.1

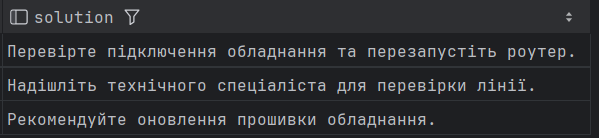


Рисунок 6.1 – результат роботи запиту a)

b)

SELECT o.name, o.surname, o.email, cs.date  
FROM operators o  
 INNER JOIN calls cs ON cs.operator\_id = o.id  
WHERE cs.id IN (  
 SELECT call\_id  
 FROM issues  
 WHERE status != 'Закрите'  
)

**Призначення:** Цей запит призначений для отримання інформації про операторів, які обробляли дзвінки, що містять незакриті запити клієнтів. Він повертає ім’я, прізвище, email оператора та дату дзвінка.

**Опис:** Запит виконується за допомогою з’єднання таблиць operators та calls. Спочатку він знаходить всі дзвінки з таблиці calls, де ідентифікатор дзвінка присутній у підзапиті, що перевіряє наявність незакритих запитів клієнтів у таблиці issues. Після цього виводяться ім’я, прізвище та email оператора, а також дата дзвінка.

**Бізнес-процеси**: Управління дзвінками клієнтів, звітністю, запитами

Результат наведений на рисунку 6.2

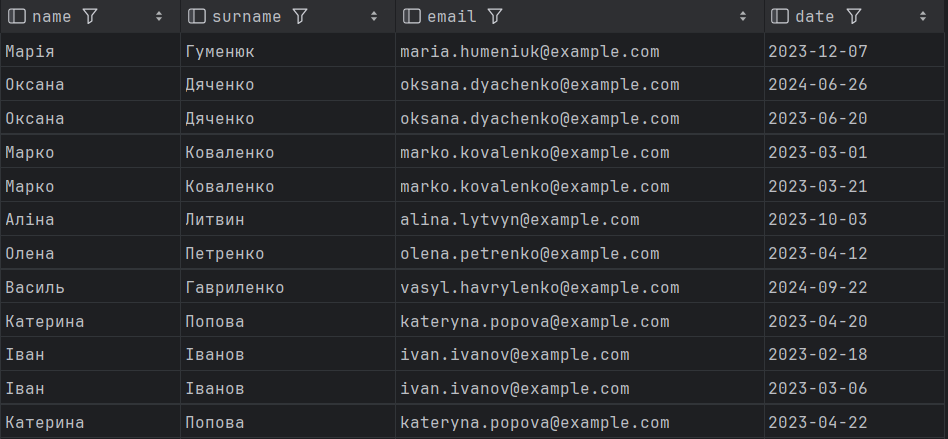


Рисунок 6.2 – результат роботи запиту b)

c)

SELECT o.name, o.surname, c.duration FROM operators o  
INNER JOIN calls c ON c.operator\_id=o.id  
WHERE c.duration = (  
 SELECT MIN(duration)  
 FROM calls  
 WHERE operator\_id=o.id  
)  
ORDER BY duration;

**Призначення:** Цей запит призначений для отримання імен, прізвищ операторів та тривалості їх найкоротших дзвінків.

**Опис:** Запит вибирає ім’я, прізвище та тривалість дзвінка для кожного оператора, де тривалість дзвінка є мінімальною серед усіх дзвінків цього оператора. Для цього використовується з’єднання таблиць operators і calls, а також підзапит для знаходження мінімальної тривалості дзвінка для кожного оператора. Результати сортуються за тривалістю.

**Бізнес-процеси:** Управління дзвінками клієнтів

Результат наведений на рисунку 6.3

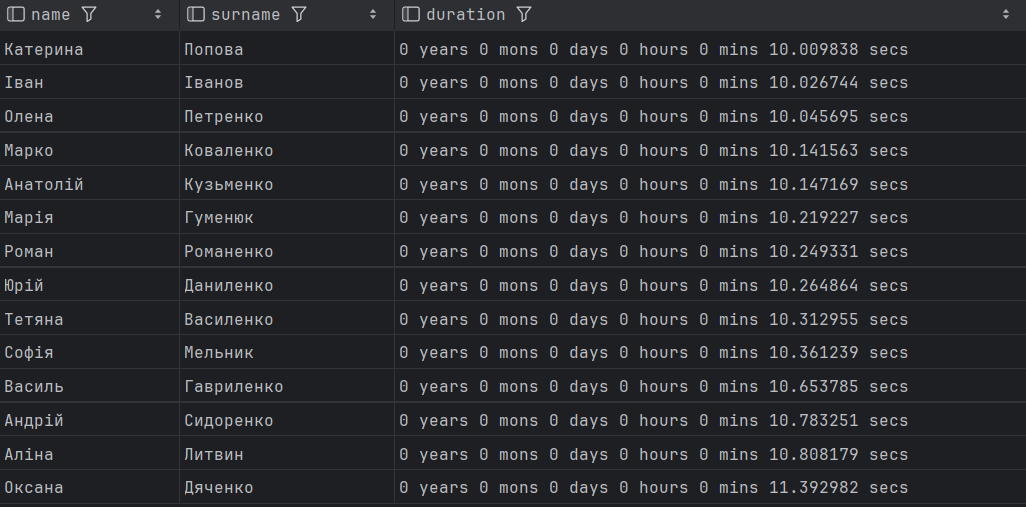


Рисунок 6.3 – результат роботи запиту c)

d)

SELECT ic.name AS category\_name,  
ROUND((SELECT COUNT(\*)  
 FROM issues i  
 WHERE i.status = 'Закрите'  
 AND ic.id = i.issue\_category\_id) \* 100 / (SELECT COUNT(\*)  
 FROM issues i  
 WHERE ic.id = i.issue\_category\_id)) AS solved\_percentage  
FROM issue\_categories ic  
ORDER BY solved\_percentage;

**Призначення:** Цей запит призначений для обчислення відсотка вирішених запитів за кожною категорією запитів та сортування результатів за цим відсотком.

**Опис:** Запит вибирає назви категорій запитів з таблиці issue\_categories та обчислює відсоток вирішених запитів для кожної категорії. Для цього підзапитами підраховується кількість вирішених запитів (статус "Закрите") та загальна кількість запитів у кожній категорії, після чого обчислюється відсоток вирішених запитів. Результати сортуються за відсотком вирішених запитів.

**Бізнес-процеси:** Управління звітністю кол-центру, запитами

Результат наведений на рисунку 6.4

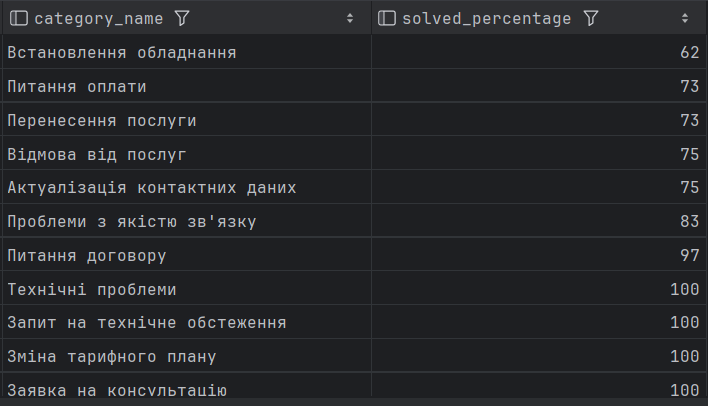


Рисунок 6.4 – результат роботи запиту d)

e)

SELECT c.name AS category\_name, COUNT(i.id) AS in\_process FROM issue\_categories c  
INNER JOIN issues i ON i.issue\_category\_id=c.id  
WHERE i.status!='Закрите'  
GROUP BY c.name;

**Призначення**: Цей запит призначений для підрахунку кількості незакритих запитів за кожною категорією запитів.

**Опис**: Запит виконується за допомогою з’єднання таблиць issue\_categories та issues, щоб отримати категорії запитів та підрахувати кількість незакритих запитів у кожній категорії. Для цього в умові WHERE перевіряється, що статус запиту не є "Закрите". Результати групуються за назвою категорії запиту.

**Бізнес-процеси**: Управління запитами

Результат наведений на рисунку 6.5

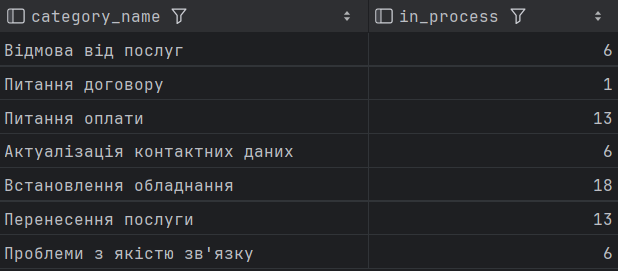


Рисунок 6.5 – результат роботи запиту e)

f)

SELECT c.id, c.name, c.surname, c.email, cs.duration FROM clients c  
INNER JOIN calls cs ON cs.client\_id=c.id  
ORDER BY cs.duration DESC  
LIMIT 10;

**Призначення**: Цей запит призначений для отримання інформації про 10 клієнтів з найдовшими дзвінками, включаючи їх ідентифікатор, ім'я, прізвище, email та тривалість дзвінка.

**Опис**: Запит виконується за допомогою з’єднання таблиць clients та calls, щоб отримати дані про клієнтів і тривалість їх дзвінків. Результати сортуються за тривалістю дзвінка в порядку спадання, і вибираються лише перші 10 записів.

**Бізнес-процеси**: Управління дзвінками клієнтів

Результат наведений на рисунку 6.6

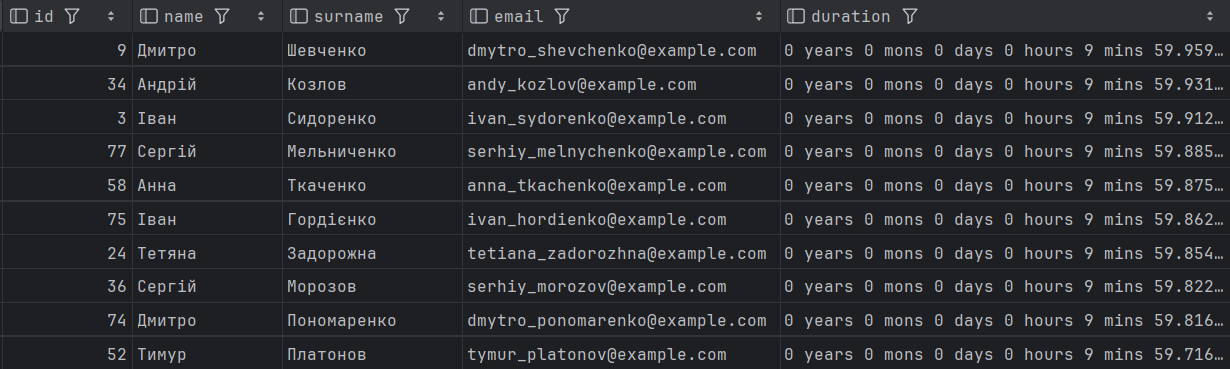


Рисунок 6.6 – результат роботи запиту f)

g)

SELECT cl.id, cl.name,cl.surname, COUNT(c.client\_id) AS number\_of\_calls FROM clients cl  
INNER JOIN calls c ON cl.id = c.client\_id  
GROUP BY cl.id, cl.name, cl.surname  
ORDER BY COUNT(c.client\_id) DESC  
LIMIT 10;

**Призначення:** Цей запит призначений для отримання списку 10 клієнтів, які здійснили найбільшу кількість дзвінків, з їх ідентифікатором, іменем, прізвищем та кількістю дзвінків.

**Опис:** Запит здійснюється за допомогою з’єднання таблиць clients та calls, щоб підрахувати кількість дзвінків для кожного клієнта. Результати групуються за ідентифікатором, ім'ям та прізвищем клієнта. Дані сортуються за кількістю дзвінків у порядку спадання, і вибираються тільки перші 10 клієнтів.

**Бізнес-процеси:** Управління дзвінками клієнтів

Результат наведений на рисунку 6.7

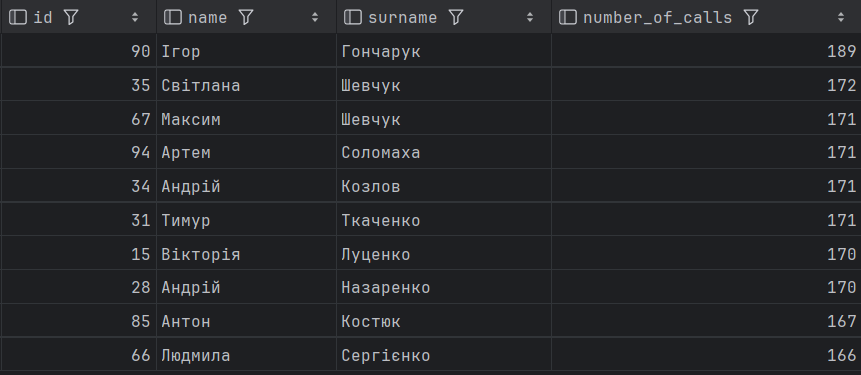


Рисунок 6.7 – результат роботи запиту g)

h)

SELECT  
 o.name AS operator\_name,  
 o.surname AS operator\_surname,  
 c.name AS client\_name,  
 c.surname AS client\_surname,  
 cs.duration AS calls\_duration  
FROM operators o  
 INNER JOIN calls cs ON cs.operator\_id=o.id  
 INNER JOIN clients c ON cs.client\_id = c.id  
 WHERE cs.duration BETWEEN '3 mins' AND '6 mins';

**Призначення**: Цей запит призначений для отримання інформації про операторів, клієнтів та тривалість дзвінків у діапазоні від 3 до 6 хвилин.

**Опис**: Запит виконується за допомогою з’єднання таблиць operators, calls та clients. Він вибирає дані про імена операторів і клієнтів, а також тривалість дзвінка для тих записів, де тривалість дзвінка знаходиться між 3 та 6 хвилинами.

**Бізнес-процеси**: Управління дзвінками клієнтів

Результат наведений на рисунку 6.8

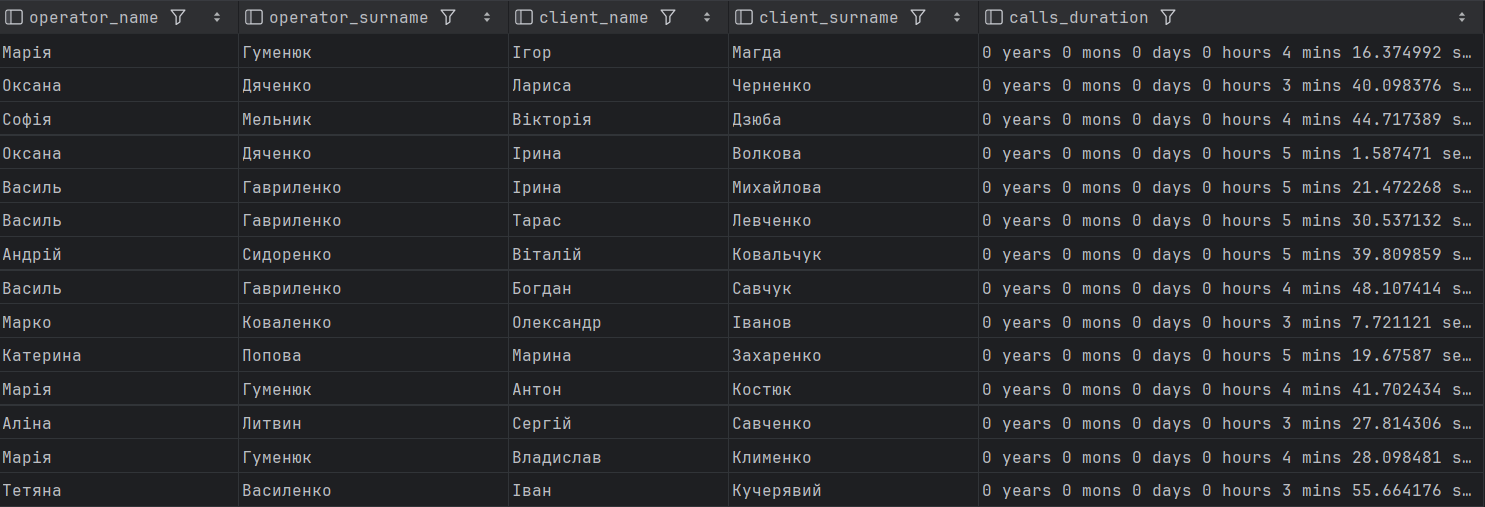


Рисунок 6.8 – результат роботи запиту h)

i)

SELECT c.name, c.surname, c.email, COUNT(DISTINCT ca.id) AS total\_calls  
FROM clients c  
 INNER JOIN calls ca ON c.id = ca.client\_id  
 INNER JOIN issues i ON ca.id = i.call\_id  
WHERE i.issue\_category\_id = 2  
GROUP BY c.name, c.surname, c.email  
HAVING COUNT(DISTINCT ca.id) > 1;

**Призначення**: Цей запит призначений для отримання інформації про клієнтів, які зверталися з конкретною категорією проблем, і мають більше одного дзвінка з цією проблемою. Запит виводить їхні імена, прізвища, email та загальну кількість дзвінків.

**Опис**: Запит виконується через з’єднання таблиць clients, calls та issues, де підраховується кількість унікальних дзвінків клієнтів, які належать до певної категорії проблем. Результати групуються за клієнтами, і виводяться лише ті клієнти, у яких більше одного дзвінка, пов'язаного з цією категорією проблем.

**Бізнес-процеси**: Управління дзвінками клієнтів, звітністю, запитами

Результат наведений на рисунку 6.9

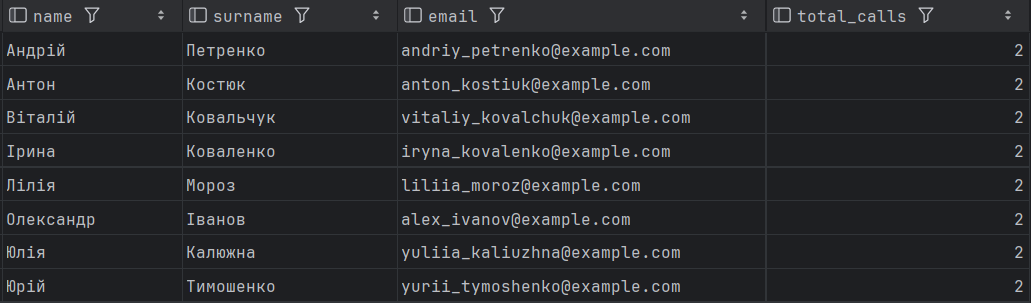


Рисунок 6.9– результат роботи запиту i)

j)

SELECT o.name, o.surname, COUNT(DISTINCT c.client\_id) FROM operators o  
INNER JOIN calls c ON o.id = c.operator\_id  
GROUP BY o.name, o.surname

**Призначення**: Цей запит призначений для отримання інформації про кількість унікальних клієнтів, з якими працював кожен оператор, з виведенням імені та прізвища оператора.

**Опис**: Запит виконується через з’єднання таблиць operators та calls, підраховуючи кількість унікальних клієнтів, з якими працював кожен оператор. Результати групуються за іменем та прізвищем оператора.

**Бізнес-процеси**: Управління звітністю кол-центру

Результат наведений на рисунку 6.10

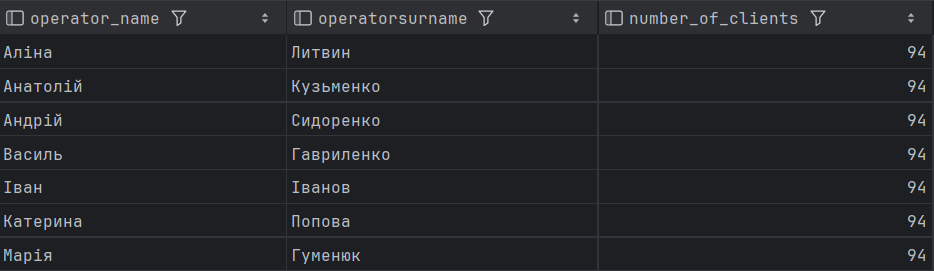


Рисунок 6.10 – результат роботи запиту j)

k)

SELECT DISTINCT ON (c.id) c.name, c.surname, c.email, cs.duration  
FROM clients c  
INNER JOIN calls cs ON cs.client\_id = c.id  
ORDER BY c.id, cs.date;

**Призначення**: Запит призначений для отримання інформації про клієнтів та тривалість їх дзвінків.

**Опис**: Запит вибирає унікальних клієнтів разом з їх ім'ям, прізвищем, email та тривалістю останнього дзвінка. Дані сортуються за клієнтом та датою дзвінка, щоб вивести тільки останній дзвінок кожного клієнта.

**Бізнес-процеси**: Управління дзвінками клієнтів

Результат наведений на рисунку 6.11

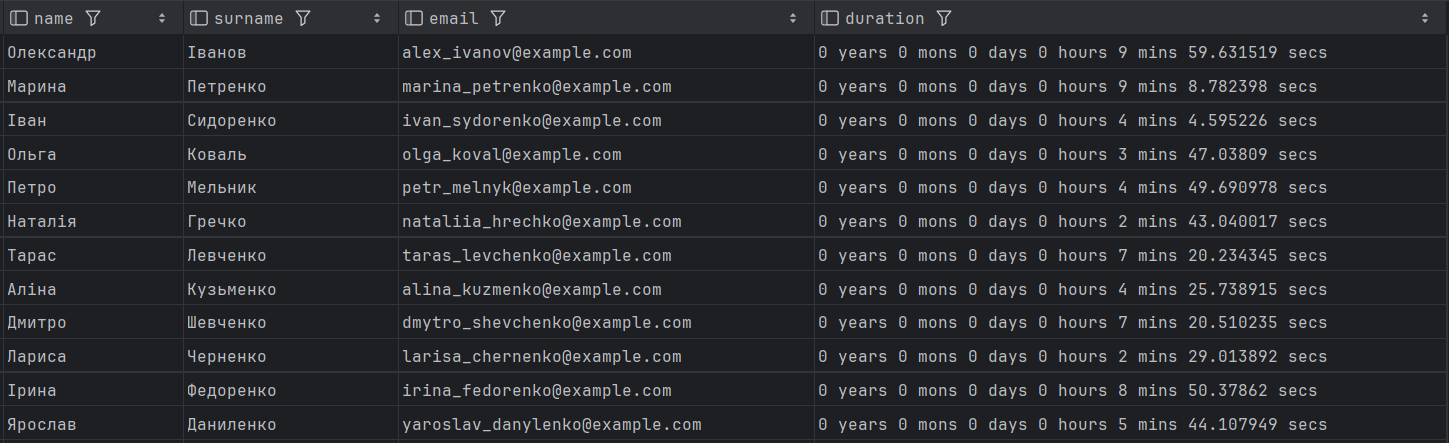


Рисунок 6.11 – результат роботи запиту k)

l)

SELECT m.topic, m.text, c.email, mc.date FROM mail m  
INNER JOIN mail\_client mc ON mc.mail\_id=m.id  
INNER JOIN clients c ON c.id=mc.client\_id  
WHERE mc.date >= NOW() - INTERVAL '1 year';

**Призначення**: Запит призначений для отримання інформації про листи, відправлені клієнтам протягом останнього року, разом з їхнім вмістом та електронною адресою клієнта.

**Опис**: Запит вибирає тему, текст листа, email клієнта та дату відправлення листа з таблиць mail, mail\_client та clients, фільтруючи дані за останній рік.

**Бізнес-процеси**: Управління звітністю кол-центру

Результат наведений на рисунку 6.12

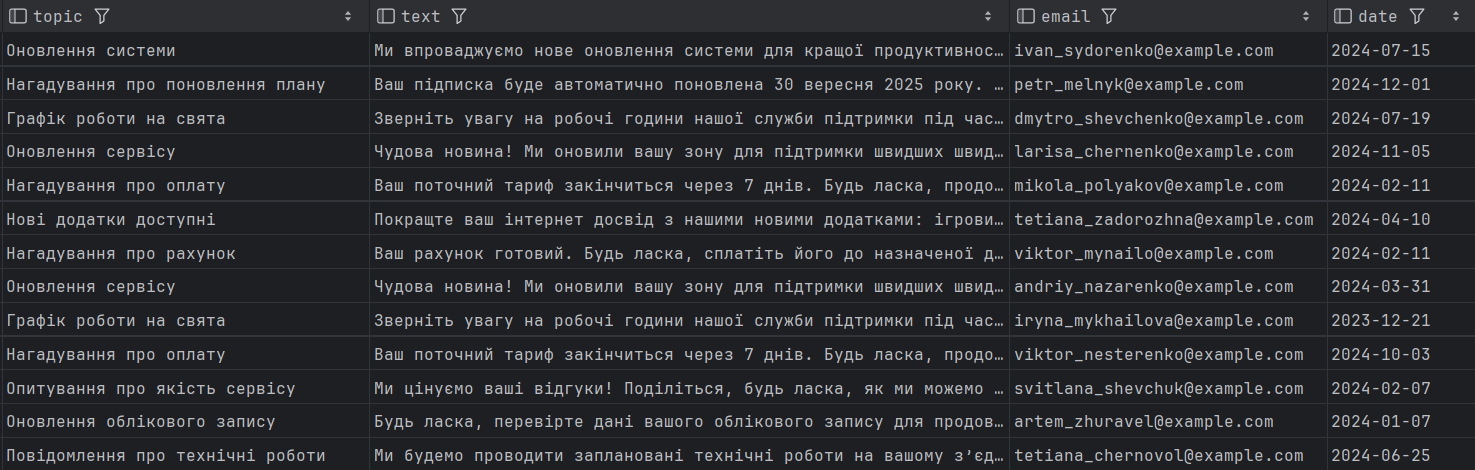


Рисунок 6.12 – результат роботи запиту l)

m)

SELECT c.name, c.surname, c.email, cs.date, COUNT(cs.id) FROM clients c  
INNER JOIN calls cs ON cs.client\_id=c.id  
GROUP BY c.name, c.surname, c.email, cs.date  
HAVING COUNT(cs.id) > 3;

**Призначення**: Запит призначений для отримання інформації про клієнтів, які здійснили більше трьох дзвінків в одну дату, щоб оцінити частоту їх звернень.

**Опис**: Запит вибирає ім'я, прізвище, email клієнта, дату дзвінка та кількість дзвінків для кожного клієнта, який здійснив більше трьох дзвінків в одну й ту саму дату. Дані фільтруються за умовою, щоб вибрати тільки клієнтів з кількістю дзвінків, більшою за три.

**Бізнес-процеси**: Управління дзвінками клієнтів, звітністю

Результат наведений на рисунку 6.13

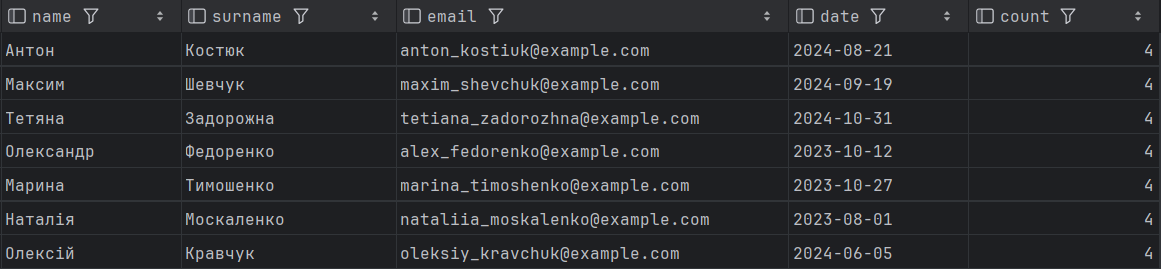


Рисунок 6.13 – результат роботи запиту m)

n)

SELECT c.name, COUNT(i.id) AS issue\_count  
FROM issue\_categories c  
JOIN issues i ON c.id = i.issue\_category\_id  
GROUP BY c.name  
ORDER BY issue\_count DESC  
LIMIT 8;

**Призначення**: Запит призначений для формування звіту про категорії запитів, що найбільш часто виникають, для визначення найбільш поширених проблем серед клієнтів.

**Опис**: Запит вибирає назву категорії запиту та кількість таких запитів, групуючи за категорією. Виводяться лише топ-8 категорій з найбільшим числом запитів, відсортованих за спаданням кількості.

**Бізнес-процеси**: Управління звітністю кол-центру

Результат наведений на рисунку 6.14

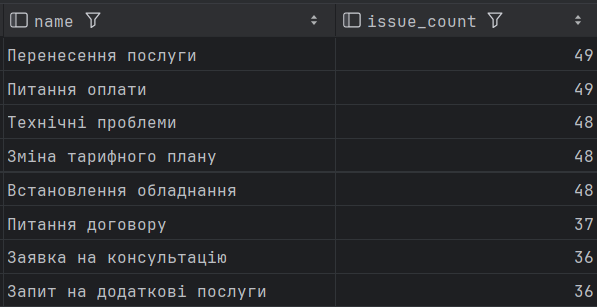


Рисунок 6.14 – результат роботи запиту n)

o)

SELECT o.id, o.name, COUNT(c.id) FROM operators o  
INNER JOIN calls c ON c.operator\_id=o.id  
GROUP BY o.id, o.name;

**Призначення**: Запит призначений для підрахунку кількості дзвінків, оброблених кожним оператором, що дозволяє оцінити їх робоче навантаження.

**Опис**: Запит вибирає ідентифікатор та ім'я оператора, а також кількість дзвінків, оброблених ним, групуючи результат за оператором. Це дозволяє визначити, скільки дзвінків кожен оператор обробив.

**Бізнес-процеси**: Управління дзвінками клієнтів, звітністю

Результат наведений на рисунку 6.15

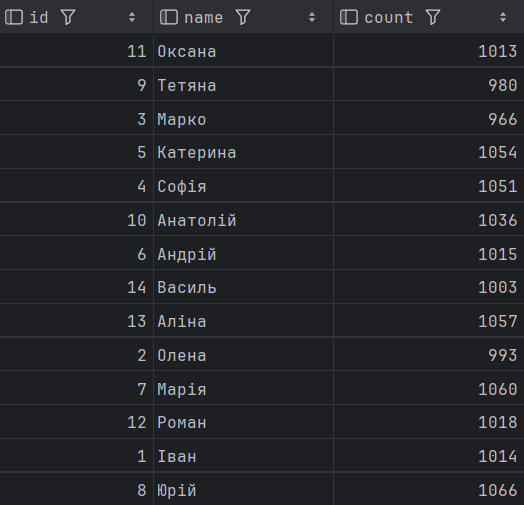


Рисунок 6.15 – результат роботи запиту o)

p)

SELECT  
 c.id AS client\_id, c.name AS client\_name, c.surname AS client\_surname,  
 tas.id AS operator\_id, tas.name AS operator\_name, tas.surname AS operator\_surname  
 FROM clients c  
INNER JOIN calls cs ON cs.client\_id=c.id  
INNER JOIN top\_average\_salary tas ON tas.id=cs.operator\_id  
WHERE c.name='Олександр' AND cs.date >= '2024-10-01' AND cs.date <= '2024-10-31'  
ORDER BY tas.avg  
LIMIT 1;

**Призначення**: Запит призначений для знаходження оператора з найвищою середньою зарплатою, який обробляв дзвінки клієнта на ім'я "Олександр" в межах жовтня 2024 року.

**Опис**: Запит вибирає інформацію про клієнта "Олександр" та оператора, який обробляв його дзвінки в межах вказаного періоду, і повертає дані про оператора з найвищою середньою зарплатою. Результат сортується за середньою зарплатою оператора

Результат наведений на рисунку 6.16

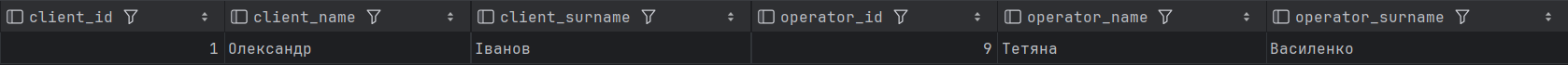


Рисунок 6.16 – результат роботи запиту p)

q)

SELECT o.name, o.surname, p.phone FROM operators o  
INNER JOIN operators\_phones p ON p.operator\_id=o.id;

**Призначення**: Запит призначений для отримання імен, прізвищ операторів та їхніх телефонних номерів.

**Опис**: Запит об'єднує таблиці операторів та їхніх телефонів за ідентифікатором оператора і повертає ім'я, прізвище та номер телефону кожного оператора.

**Бізнес-процеси**: Управління телефонами операторів

Результат наведений на рисунку 6.17

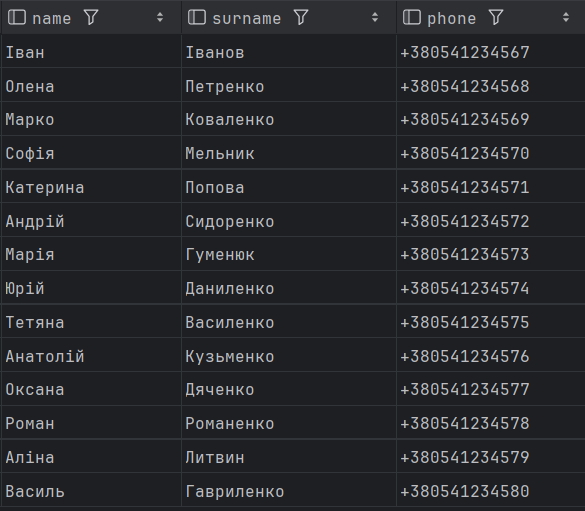


Рисунок 6.17 – результат роботи запиту q)

r)

SELECT o.name, o.surname, days\_of\_week FROM operators o  
INNER JOIN operator\_schedule os ON os.operator\_id=o.id  
WHERE days\_of\_week LIKE '%6%' OR days\_of\_week LIKE '%7%';

**Призначення**: Запит призначений для отримання імен та прізвищ операторів, які працюють у с.

**Опис**: Запит об'єднує таблиці операторів та їхніх графіків роботи на основі ідентифікатора оператора. За допомогою умови LIKE відбираються ті записи, у яких графік роботи містить 6 або 7 день тижня (субота або неділя).

**Бізнес-процеси**: Управління графіком операторів

Результат наведений на рисунку 6.18

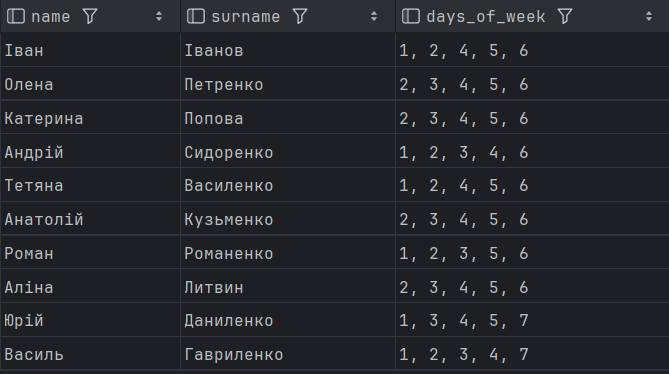


Рисунок 6.18 – результат роботи запиту r)

s)

SELECT o.name, o.surname, o.email, os.days\_of\_week FROM operators o  
INNER JOIN operator\_schedule os ON os.operator\_id=o.id  
WHERE shift\_start='08:00' AND shift\_end='17:00';

**Призначення**: Запит призначений для отримання інформації про операторів, які працюють у зміні з 08:00 до 17:00.

**Опис**: Запит об'єднує таблиці операторів та їхніх графіків роботи на основі ідентифікатора оператора. Умова WHERE обмежує результати тими графіками, де початок зміни дорівнює 08:00, а кінець — 17:00. В результаті виводяться ім'я, прізвище, email та дні тижня, коли оператор працює у цій зміні.

**Бізнес-процеси**: Управління графіком операторів.

Результат наведений на рисунку 6.19

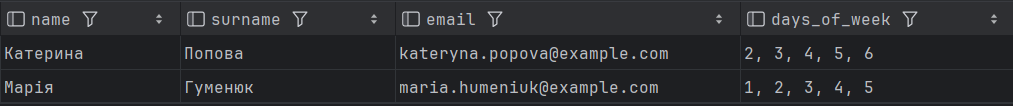


Рисунок 6.19 – результат роботи запиту s)

t)

SELECT c.id, c.name, c.surname, c.email  
FROM clients c  
 LEFT JOIN calls cs ON c.id = cs.client\_id  
 AND date >= NOW() - INTERVAL '21 day'  
 AND date <= NOW()  
WHERE cs.id IS NULL;

**Призначення**: Запит призначений для отримання списку клієнтів, які не здійснювали дзвінків протягом останніх 21 дня.

**Опис**: Виконується LEFT JOIN таблиці клієнтів з таблицею дзвінків на основі ідентифікатора клієнта. Додаткові умови перевіряють, чи дата дзвінка знаходиться в інтервалі останніх 21 дня. Умова WHERE cs.id IS NULL забезпечує вибір лише тих клієнтів, у яких не знайдено відповідних записів у таблиці дзвінків за цей період. В результаті повертаються ідентифікатор, ім'я, прізвище та email таких клієнтів.

**Бізнес-процеси**: Управління дзвінками клієнтів

Результат наведений на рисунку 6.20

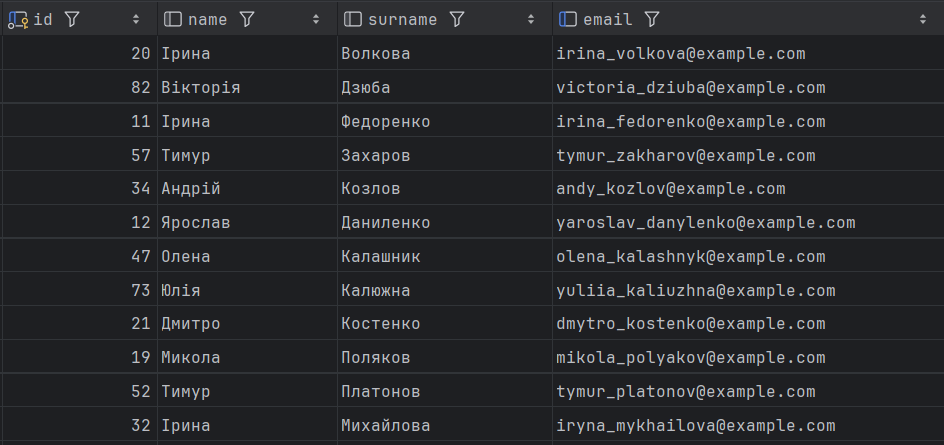


Рисунок 6.20 – результат роботи запиту t)

## Функції/Збережені процедури

a)

CREATE OR REPLACE PROCEDURE create\_client(  
 \_name VARCHAR,  
 \_surname VARCHAR,  
 \_email VARCHAR,  
 \_phone VARCHAR,  
 \_address VARCHAR,  
 \_operator\_id INT,  
 \_duration INTERVAL  
)  
 LANGUAGE plpgsql  
AS  
$$  
 DECLARE  
 \_client\_id INT;  
 \_call\_id INT;  
 \_date DATE:= TO\_CHAR(NOW(), 'YYYY-MM-DD');  
 BEGIN  
  
 IF \_phone NOT LIKE '+380%' OR LENGTH(\_phone) != 13 THEN  
 RAISE EXCEPTION 'Телефон має бути справжнім';  
 END IF;  
  
 IF \_email NOT LIKE '%@%.%' THEN  
 RAISE EXCEPTION 'Некоректна пошта';  
 END IF;  
  
 INSERT INTO clients  
 VALUES(default, \_name, \_surname, \_email, \_phone, \_address)  
 RETURNING id INTO \_client\_id;  
  
 INSERT INTO calls  
 VALUES(default, \_operator\_id, \_client\_id, \_date, \_duration)  
 RETURNING id INTO \_call\_id;  
  
 INSERT INTO issues  
 VALUES(default, 6, 'Клієнт хоче підписати новий договір', 'Оброблюється', \_call\_id);  
 END;  
$$;

**Призначення:**  
Процедура автоматизує реєстрацію нового клієнта, збереження даних про його дзвінок та створення запиту про підписання нового договору

**Опис:**  
Процедура create\_client перевіряє коректність введених даних, таких як номер телефону та електронна пошта. Після успішної валідації вона додає нового клієнта до таблиці clients. Далі реєструє дзвінок від клієнта в таблиці calls та створює новий запит у таблиці issues зі статусом "Оброблюється" і категорією "Клієнт хоче підписати новий договір"

**Бізнес-процес:**  
Реєстрація клієнта

Приклад роботи наведений на рисунку 6.21

CALL create\_client('Andriy', 'Ponchick', 'andriy\_ponchick@example.com', '+380877852398', 'вул. Червоноармійська 15, Київ', 2, '8 mins')

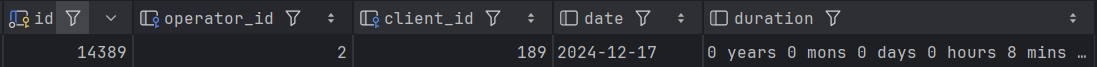
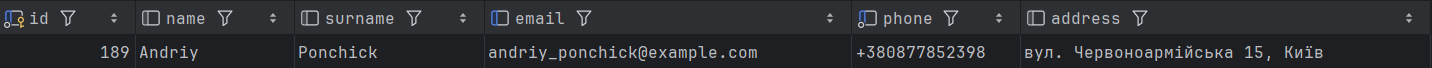
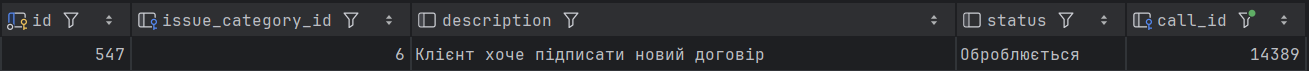
  

Рисунок 6.21 – приклад роботи процедури a)

b)

CREATE OR REPLACE PROCEDURE callIn()  
 LANGUAGE plpgsql  
AS $$  
BEGIN  
 INSERT INTO calls (operator\_id, client\_id, date, duration)  
 WITH random\_data AS (  
 SELECT  
 FLOOR(1 + (14) \* RANDOM())::INT AS operator\_id,  
 (DATE '2023-01-01' + (RANDOM() \* (DATE '2024-11-30' - DATE '2023-01-01'))::INT) AS random\_date  
 FROM generate\_series(1, 20000)  
 )  
 SELECT  
 rd.operator\_id,  
 FLOOR(1 + (94) \* RANDOM())::INT AS client\_id,  
 rd.random\_date,  
 INTERVAL '10 seconds' + (RANDOM() \* (INTERVAL '10 minutes' - INTERVAL '10 seconds')) AS duration  
 FROM random\_data rd  
 WHERE is\_operator\_working(rd.operator\_id, rd.random\_date);  
END;  
$$;

**Призначення:**  
Процедура автоматично генерує та додає записи про дзвінки операторів до таблиці calls за допомогою випадкових даних. Ця процедура використовується для тестування

**Опис:**  
Процедура callIn створює 20,000 записів у таблиці calls. Для кожного запису випадковим чином вибирається оператор (ID в діапазоні від 1 до 14), клієнт (ID від 1 до 94), дата дзвінка (між 2023-01-01 і 2024-11-30), а також тривалість дзвінка (від 10 секунд до 10 хвилин). Запис додається тільки тоді, коли оператор працює в обраний день (це перевіряється за допомогою функції is\_operator\_working)

**Бізнес-процес:**  
Обробка дзвінків клієнтів

Приклад роботи наведений на рисунку 6.22

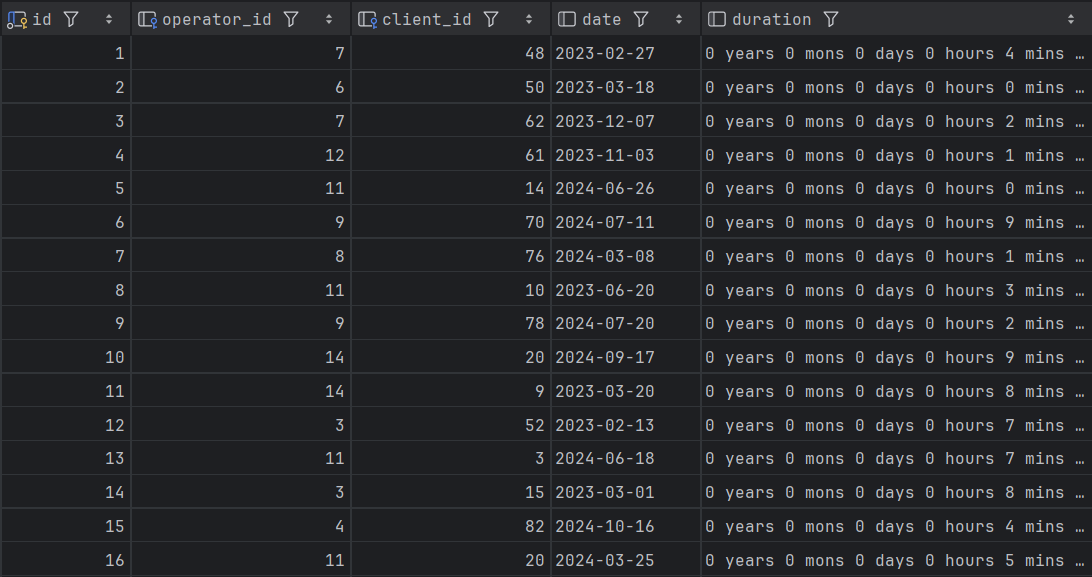


Рисунок 6.22 – приклад роботи процедури b)

c)

CREATE OR REPLACE PROCEDURE createNewReport(\_date DATE)  
 LANGUAGE plpgsql  
AS  
$$  
 DECLARE  
 \_id integer;  
 cdur INTERVAL;  
 ys DECIMAL;  
 hhe BIT;  
 hheCef DECIMAL:=0.07;  
 dhm DECIMAL:=50.0;  
 hm DECIMAL;  
 ts DECIMAl;  
BEGIN  
 FOR \_id IN  
 SELECT id FROM operators  
 LOOP  
 SELECT SUM(duration) INTO cdur FROM calls c  
 WHERE operator\_id=\_Id AND c.date <= \_date AND c.date >= \_date - INTERVAL '1 month';  
  
 SELECT getEx(\_id) INTO ys;  
  
 SELECT has\_higher\_education INTO hhe  
 FROM operators  
 WHERE id=\_id;  
  
  
 RAISE NOTICE 'YEAR: %', ys;  
  
 IF(CAST(hhe AS INTEGER) = 1) THEN  
 hm := dhm + dhm\*hheCef + dhm\*(ys/100);  
 ELSE  
 hm := dhm + dhm\*(ys/100);  
 END IF;  
  
 RAISE NOTICE 'Value: %', hm;  
  
  
 ts := hm \* (EXTRACT(EPOCH FROM cdur) / 60);  
  
 INSERT INTO operators\_report  
 VALUES(default,\_id, cdur, ts, \_date);  
 END LOOP;  
END;  
$$;

**Призначення:**  
Процедура створює новий звіт для кожного оператора за вказану дату, враховуючи кількість дзвінків, їхню тривалість, рівень освіти оператора, та додаткові коефіцієнти для обчислення зарплати

**Опис:**  
Процедура createNewReport обчислює звіт за вказану дату для кожного оператора. Для цього збирається сумарна тривалість дзвінків оператора за останній місяць, перевіряється рівень освіти оператора та обчислюється зарплата в залежності від наявності вищої освіти та спеціальних коефіцієнтів. Потім результат обчислення (зарплата, тривалість дзвінків та інші дані) записується в таблицю operators\_report

**Бізнес-процес:**  
Управління звітністю кол-центру

Приклад роботи наведений на рисунку 6.23

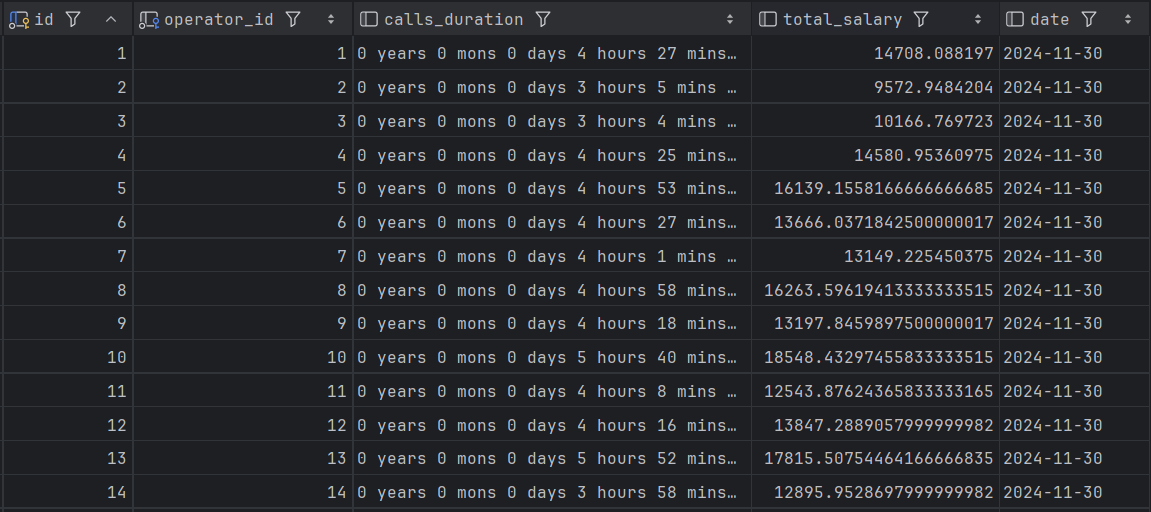


Рисунок 6.23 – приклад роботи процедури c)

d)

CREATE OR REPLACE FUNCTION is\_operator\_working(\_operator\_id INT, check\_date DATE)  
RETURNS BOOLEAN  
LANGUAGE plpgsql  
AS $$  
 DECLARE  
 work\_days INT[];  
 day\_of\_week INT;  
BEGIN  
  
SELECT string\_to\_array(days\_of\_week, ',')::INT[]  
INTO work\_days  
FROM operator\_schedule  
WHERE operator\_id = \_operator\_id;  
  
IF work\_days IS NULL THEN  
 RETURN FALSE;  
END IF;  
  
 day\_of\_week := EXTRACT(ISODOW FROM check\_date);  
  
RETURN day\_of\_week = ANY(work\_days);  
END;  
$$;

**Призначення**  
Функція перевіряє, чи працює оператор у вказаний день на основі його графіку роботи

**Опис**  
Функція is\_operator\_working перевіряє, чи працює оператор в зазначений день. Вона отримує розклад роботи оператора з таблиці operator\_schedule, порівнює день тижня з доступними робочими днями оператора і повертає значення TRUE, якщо оператор працює в цей день, і FALSE в іншому випадку

**Бізнес-процес**  
Управління графіком операторів

Приклад роботи наведений на рисунку 6.24

SELECT is\_operator\_working(1, '2024-11-09');

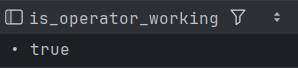


Рисунок 6.24 – приклад роботи функції d)

e)

CREATE OR REPLACE PROCEDURE delete\_old\_closed\_issues()  
LANGUAGE plpgsql

AS

$$  
BEGIN  
DELETE FROM issues  
WHERE status = 'Закрите'  
 AND call\_id IN (  
 SELECT id FROM calls  
 WHERE date < NOW() - INTERVAL '18 month'  
 );  
  
RAISE NOTICE 'Старі завершені задачі успішно видалено';  
END;  
$$;

**Призначення**  
Процедура видаляє старі завершені задачі, що пов'язані з дзвінками, які мали місце більше 18 місяців тому

**Опис**  
Процедура delete\_old\_closed\_issues видаляє записи в таблиці issues, де статус задачі є "Завершено", і дзвінок, до якого вона прив'язана, був здійснений більше 18 місяців тому. Після виконання процедури виводиться повідомлення про успішне видалення старих завершених задач

**Бізнес-процес**  
Управління запитами та їх архівація

Приклад роботи наведений на рисунках 6.25 і 6.26

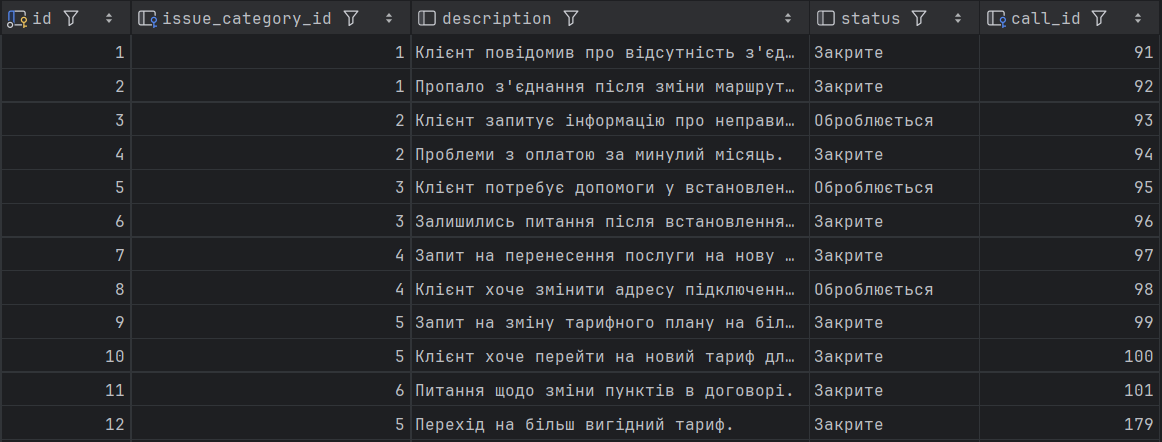


Рисунок 6.25 – приклад до роботи процедури e)

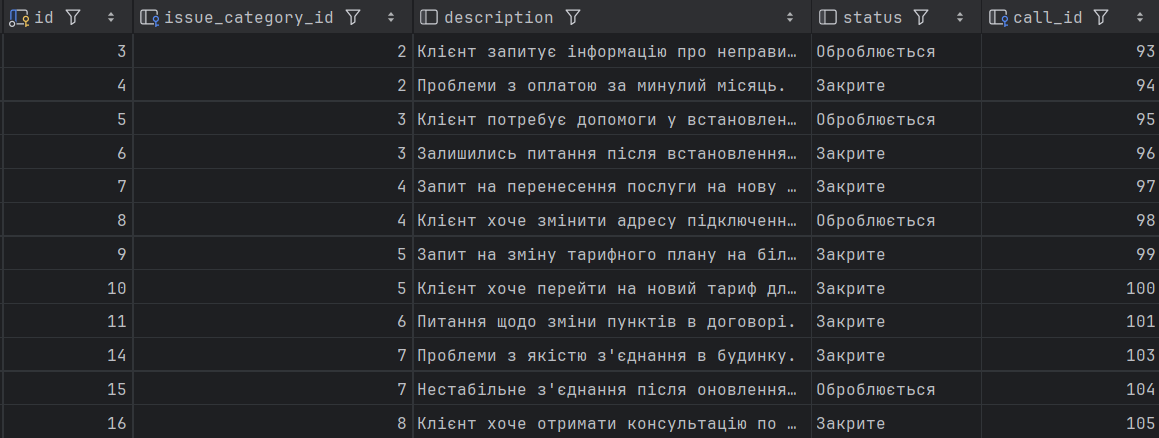


Рисунок 6.26 – приклад після роботи процедури e)

f)

CREATE OR REPLACE FUNCTION dynamic\_get(\_columns text, \_table varchar)  
 RETURNS SETOF RECORD  
 LANGUAGE plpgsql  
AS  
$$  
BEGIN  
 RETURN QUERY EXECUTE 'SELECT ' || \_columns || ' FROM ' || \_table;  
END;  
$$;

**Призначення**  
Функція виконує динамічний запит для отримання вказаних стовпців з таблиці

**Опис**  
Функція dynamic\_get дозволяє виконувати динамічний SQL-запит на вибірку даних з вказаної таблиці та стовпців. Користувач передає список стовпців та ім'я таблиці як параметри, і функція виконує запит, що повертає результати відповідно до вказаних параметрів

Приклад роботи наведений на рисунку 6.27

SELECT \* FROM dynamic\_get('id, name, surname, email', 'clients')  
 AS (id INT, name VARCHAR, surname VARCHAR, email VARCHAR);

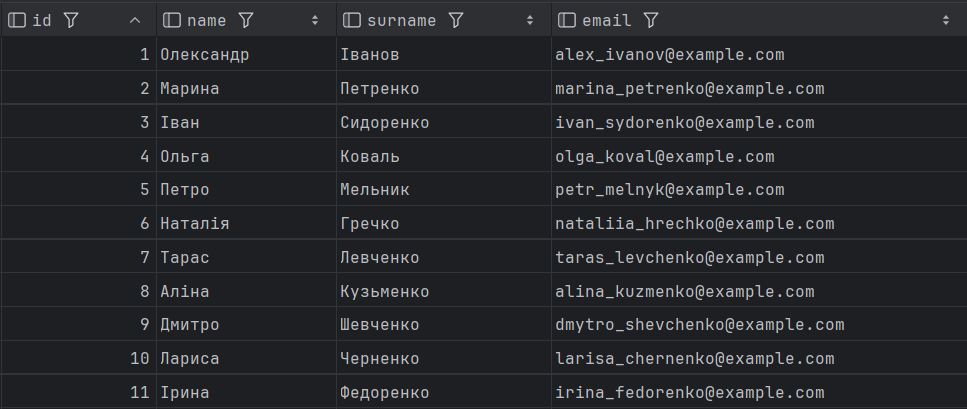


Рисунок 6.27 – приклад роботи функції f)

g)

CREATE OR REPLACE FUNCTION get\_rep(\_date DATE)  
 RETURNS TABLE(call\_duration INTERVAL, number\_of\_calls INT, unsolved\_problems INT)  
 LANGUAGE plpgsql  
AS  
$$  
 DECLARE  
 cd INTERVAL;  
 noc INT;  
 up INT;  
BEGIN  
 SELECT SUM(duration), COUNT(duration) INTO cd,noc FROM calls c  
 WHERE c.date <= \_date AND c.date >= \_date - INTERVAL '1 month';  
  
 SELECT COUNT(\*) INTO up FROM issues i  
 INNER JOIN calls c ON i.call\_id=c.id  
 WHERE c.date <= \_date AND c.date >= \_date - INTERVAL '1 month' AND i.status!='Закрите';  
  
 RETURN QUERY  
 SELECT cd, noc, up;  
END;  
$$;

**Призначення**  
Функція надає зведену інформацію про кількість дзвінків, їхню тривалість та кількість невирішених проблем за останній місяць відносно вказаної дати

**Опис**  
Функція get\_rep приймає дату як параметр і виконує два основні запити: перший обчислює загальну кількість дзвінків та сумарну їхню тривалість за останній місяць до цієї дати, другий підраховує кількість невирішених проблем, пов'язаних із дзвінками в цей самий період. В результаті функція повертає тривалість дзвінків, кількість дзвінків і кількість невирішених проблем

**Бізнес-процес**  
Управління звітністю кол-центру

Приклад роботи наведений на рисунку 6.28

SELECT \* FROM get\_rep('2024-06-25');

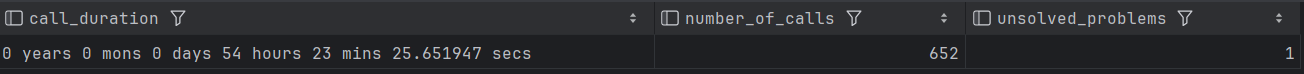


Рисунок 6.28 – приклад роботи функції g)

h)

CREATE OR REPLACE FUNCTION get\_schedule(\_operator\_id INT)  
 RETURNS TABLE  
 (  
 operator\_name VARCHAR,  
 operator\_surname VARCHAR,  
 current\_days\_of\_week VARCHAR,  
 current\_shift\_start TIME,  
 current\_shift\_end TIME  
 )  
 LANGUAGE plpgsql  
AS  
$$  
DECLARE  
 op\_name VARCHAR;  
 op\_surname VARCHAR;  
 \_days\_of\_week\_OLD VARCHAR;  
 \_days\_of\_week\_NEW VARCHAR;  
 \_shift\_start TIME;  
 \_shift\_end TIME;  
  
BEGIN  
 SELECT name, surname  
 INTO op\_name, op\_surname  
 FROM operators  
 WHERE id = \_operator\_id;  
  
 SELECT days\_of\_week, shift\_start, shift\_end  
 INTO \_days\_of\_week\_OLD, \_shift\_start, \_shift\_end  
 FROM operator\_schedule  
 WHERE operator\_id=\_operator\_id;  
  
 RAISE NOTICE 'Value: %', \_days\_of\_week\_OLD;  
  
 SELECT string\_agg(  
 CASE  
 WHEN day = '1' THEN 'Понеділок'  
 WHEN day = '2' THEN 'Вівторок'  
 WHEN day = '3' THEN 'Середа'  
 WHEN day = '4' THEN 'Четвер'  
 WHEN day = '5' THEN 'П''ятниця'  
 WHEN day = '6' THEN 'Субота'  
 WHEN day = '7' THEN 'Неділя'  
 END, ', '  
 ) INTO \_days\_of\_week\_NEW  
 FROM unnest(string\_to\_array( \_days\_of\_week\_OLD, ', ')) AS day;  
  
 RAISE NOTICE 'Value: %', \_days\_of\_week\_NEW;  
  
 RETURN QUERY  
 SELECT op\_name, op\_surname, \_days\_of\_week\_NEW, \_shift\_start, \_shift\_end;  
END  
$$;

**Призначення**  
Функція надає розклад роботи оператора, включаючи дні тижня, коли оператор працює, та час початку й закінчення зміни

**Опис**  
Функція get\_schedule приймає ідентифікатор оператора як параметр і повертає таблицю з інформацією про ім’я оператора, його прізвище, робочі дні тижня, а також час початку і закінчення його зміни. Дні тижня, в яких оператор працює, представлені в зрозумілому форматі (приклади: Понеділок, Вівторок тощо)

**Бізнес-процес**  
Управління графіком операторів

Приклад роботи наведений на рисунку 6.29

SELECT \* FROM get\_schedule(2);

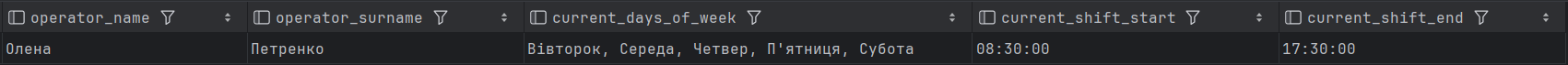


Рисунок 6.29– приклад роботи функції h)

i)

CREATE Or REPLACE PROCEDURE get\_processing()  
 LANGUAGE plpgsql  
AS  
$$  
BEGIN  
 CREATE TEMP TABLE old\_inprocessing (  
 id SERIAL,  
 issue VARCHAR,  
 date DATE  
 );  
  
 INSERT INTO old\_inprocessing (issue, date)  
 SELECT i.description, c.date  
 FROM issues i  
 JOIN calls c ON i.call\_id = c.id  
 WHERE i.status = 'Оброблюється'  
 AND c.date < NOW() - INTERVAL '1 month';  
END;  
$$;

**Призначення**  
Процедура створює тимчасову таблицю, яка містить дані про задачі, що знаходяться в статусі "Оброблюється" і мають дату обробки більше одного місяця тому

**Опис**  
Процедура get\_processing створює тимчасову таблицю old\_inprocessing з полями id, issue та date. Вона заповнюється даними про задачі, що знаходяться в статусі "Оброблюється", та мають дату обробки більше ніж місяць назад. Дані отримуються шляхом об'єднання таблиць issues та calls, фільтруючи за статусом і датою

**Бізнес-процес**

Управління запитами та їх архівація

Приклад роботи наведений на рисунку 6.30

SELECT \* FROM old\_inprocessing  
ORDER BY date;

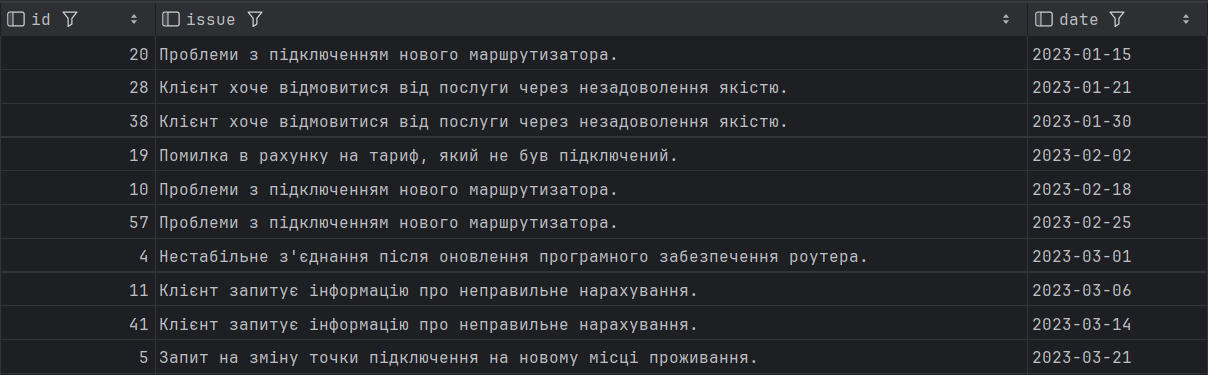


Рисунок 6.30– приклад роботи процедури i)

j)

CREATE FUNCTION getEX(\_id INT)  
 RETURNS INTEGER  
 LANGUAGE plpgsql  
AS  
$$  
DECLARE  
 hd DATE;  
 years INTEGER;  
BEGIN  
 SELECT hire\_date INTO hd  
 FROM operators o  
 WHERE o.id = \_id;  
  
 years := EXTRACT(YEAR FROM AGE(NOW(), hd));  
  
 RETURN years;  
END;  
$$;

**Призначення**  
Функція getEX визначає кількість років досвіду роботи оператора, використовуючи дату його прийому на роботу

**Опис**  
Функція отримує дату прийому на роботу оператора з таблиці operators, використовуючи його id. Потім вона обчислює різницю між поточним часом та датою прийому на роботу, визначаючи кількість років досвіду

Приклад роботи наведений на рисунку 6.31

SELECT getEX(2)

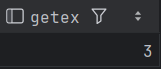


Рисунок 6.31– приклад роботи функції j)

k)

CREATE OR REPLACE PROCEDURE new\_call(  
 \_operator\_id INT,  
 \_client\_id INT,  
 \_duration INTERVAL,  
 \_issue\_category\_id INT,  
 \_description VARCHAR,  
 \_status VARCHAR  
)  
 LANGUAGE plpgsql  
AS $$  
 DECLARE  
 \_call\_id INT;  
BEGIN  
 INSERT INTO calls  
 VALUES(default, \_operator\_id, \_client\_id, NOW(), \_duration)  
 RETURNING id INTO \_call\_Id;  
  
 INSERT INTO issues  
 VALUES(default, \_issue\_category\_id, \_description, \_status, \_call\_id);  
END;  
$$;

**Призначення**  
Процедура new\_call створює новий дзвінок та запит до нього на основі переданих даних

**Опис**  
Процедура отримує усі дані, необхідні для створення дзвінка та запиту до нього, після чого спочатку створюється новий запис в таблиці calls, після чого на основі його id створюється запис в таблиці issues

**Бізнес-процес**

Управління дзвінками

## Тригери

a)

CREATE OR REPLACE FUNCTION before\_insert\_calls()  
 RETURNS TRIGGER AS  
$$  
BEGIN  
 IF NOT is\_operator\_working(NEW.operator\_Id, NEW.date) THEN  
 RAISE NOTICE 'Це не робочий день оператора';  
 RETURN NULL;  
 END IF;  
  
 DELETE FROM calls  
 WHERE date = (  
 SELECT MIN(date) FROM calls  
 WHERE date < NOW() - INTERVAL '20 month'  
 );  
  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE OR REPLACE TRIGGER trigger\_before\_delete\_calls  
 BEFORE INSERT ON calls  
 FOR EACH ROW  
EXECUTE FUNCTION before\_insert\_calls();

**Призначення**  
Функція before\_insert\_calls перевіряє, чи є день, в який додається дзвінок, робочим для оператора, та видаляє старі записи з таблиці дзвінків, якщо вони не відповідають критеріям.

**Опис**  
Функція виконується перед вставкою нового запису в таблицю calls. Вона перевіряє, чи є день, на який призначений дзвінок, робочим для оператора, використовуючи функцію is\_operator\_working. Якщо це не робочий день, вставка не відбувається, і виводиться повідомлення. Крім того, функція видаляє з таблиці calls найстаріший запис, де дата дзвінка менша ніж 20 місяців від поточної дати.

**Бізнес-процес**

Управління дзвінками та графіками операторів

Приклад роботи наведений на рисунку 6.32

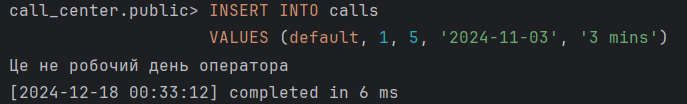


Рисунок 6.32– приклад роботи триггера a)

b)

CREATE OR REPLACE FUNCTION before\_update\_clients()  
 RETURNS TRIGGER AS  
$$  
BEGIN  
 RAISE NOTICE 'Value: %', LENGTH(NEW.phone);  
 IF NEW.phone NOT LIKE '+380%' OR LENGTH(NEW.phone) != 13 THEN  
 RAISE NOTICE 'Некоректний номер телефону, правильний має починатися з +380 і мати 13 символів';  
 RETURN NULL;  
 END IF;  
  
 IF NEW.email NOT LIKE '%@%' THEN  
 RAISE NOTICE 'Некоректний імейл';  
 RETURN NULL;  
 END IF;  
  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE OR REPLACE TRIGGER trigger\_before\_update\_clients  
 BEFORE UPDATE ON clients  
 FOR EACH ROW  
EXECUTE FUNCTION before\_update\_clients();

**Призначення**  
Тригер викликає функцію before\_update\_clients, яка перевіряє коректність номера телефону та електронної пошти клієнта перед оновленням даних у таблиці clients.

**Опис**  
Функція виконується перед оновленням записів у таблиці clients. Вона перевіряє, чи номер телефону починається з +380 і має довжину 13 символів, а також чи є електронна пошта коректною (має символ @). Якщо будь-яка з перевірок не пройдена, запис не оновлюється, і виводиться відповідне повідомлення.

Приклад роботи наведений на рисунку 6.33

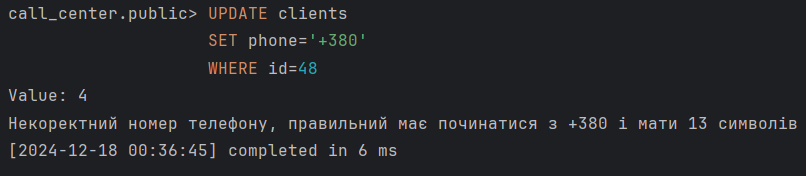


Рисунок 6.33– приклад роботи триггера b)

c)

CREATE OR REPLACE FUNCTION before\_insert\_issues()  
 RETURNS TRIGGER AS  
$$  
 DECLARE  
 \_client\_id INT;  
 inserting\_date DATE;  
 expected\_date DATE;  
BEGIN  
 SELECT date, client\_Id INTO inserting\_date, \_client\_id FROM calls  
 WHERE id=NEW.call\_id;  
  
 SELECT c.date INTO expected\_date FROM calls c  
 WHERE c.id IN (  
 SELECT call\_id FROM issues  
 WHERE status='Оброблюється' AND issue\_category\_id=NEW.issue\_category\_id  
 )  
 AND inserting\_date - INTERVAL '1 month' < c.date AND c.client\_id = \_client\_id  
 LIMIT 1;  
  
 RAISE NOTICE 'expdate: %', expected\_date;  
 IF expected\_date IS NOT NULL THEN  
 RAISE NOTICE 'Не можна вставляти проблему, якщо вже існує проблема в того-ж клієнта з такою-ж категорією, що не була вирішена впродовж останнього місяця';  
 RETURN NULL;  
 END IF;  
  
 RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE OR REPLACE TRIGGER trigger\_before\_insert\_issues  
 BEFORE INSERT ON issues  
 FOR EACH ROW  
EXECUTE FUNCTION before\_insert\_issues();

**Призначення**  
Функція before\_insert\_issues перевіряє, чи не існує у клієнта ще одна проблема тієї ж категорії, яка не була вирішена протягом останнього місяця перед вставкою нової записи у таблицю issues.

**Опис**  
Функція виконується перед вставкою нової проблеми у таблицю issues. Вона перевіряє, чи існує проблема з тією ж категорією у того ж клієнта, яка була відкрита в останній місяць і ще не вирішена. Якщо така проблема є, вставка нової проблеми блокується, і виводиться відповідне повідомлення.

Приклад роботи наведений на рисунку 6.34

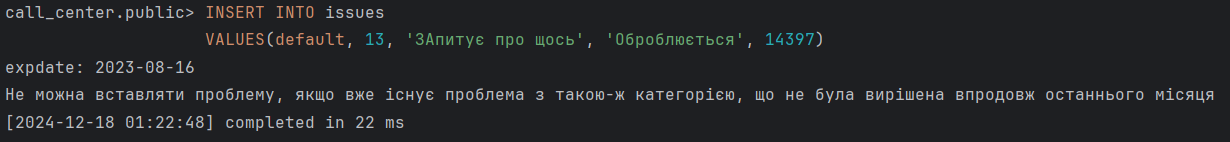


Рисунок 6.34– приклад роботи триггера c)

d)

CREATE OR REPLACE FUNCTION before\_delete\_operator()  
 RETURNS TRIGGER AS  
$$  
 DECLARE  
 opid INT;  
BEGIN  
 SELECT operator\_id INTO opid FROM calls  
 WHERE date > NOW() - INTERVAL '1 month' AND operator\_id=OLD.id;  
  
 IF opid IS NOT NULL THEN  
 RAISE NOTICE 'Не можна видаляти опаратора, який мав дзвінки впродовж останнього місяця';  
 RETURN NULL;  
 END IF;  
  
 RETURN OLD;  
END  
$$ LANGUAGE plpgsql;  
  
CREATE OR REPLACE TRIGGER trigger\_before\_delete\_operator  
 BEFORE DELETE ON operators  
 FOR EACH ROW  
EXECUTE FUNCTION before\_delete\_operator();

**Призначення**  
Функція before\_delete\_operator перевіряє, чи існують записи про дзвінки для оператора в останній місяць, і якщо такі є, забороняє видалення оператора

**Опис**  
Ця функція виконується перед видаленням запису з таблиці operators. Вона перевіряє, чи є дзвінки за участю оператора в останній місяць. Якщо такі дзвінки є, функція забороняє видалення оператора, виводячи відповідне повідомлення

Приклад роботи наведений на рисунку 6.35

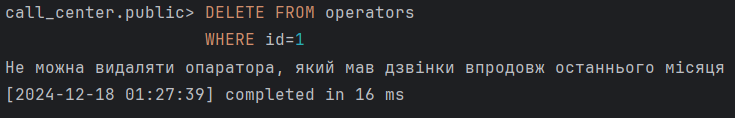


Рисунок 6.35– приклад роботи триггера d)

e)

CREATE OR REPLACE FUNCTION before\_delete\_report()  
 RETURNS TRIGGER AS  
$$  
BEGIN  
 IF(OLD.date > NOW() - INTERVAL '6 month') THEN  
 RAISE NOTICE 'Не можна видаляти звіти, що були сформовані за останні 6 місяців';  
 RETURN NULL;  
 END IF;  
  
 RETURN OLD;  
END  
$$ LANGUAGE plpgsql;  
  
CREATE OR REPLACE TRIGGER trigger\_before\_delete\_report  
 BEFORE DELETE ON operators\_report  
 FOR EACH ROW  
EXECUTE FUNCTION before\_delete\_report();

**Призначення**  
Функція before\_delete\_report перевіряє, чи не є звіт сформованим протягом останніх 6 місяців, і забороняє його видалення, якщо це так.

**Опис**  
Ця функція виконується перед видаленням запису з таблиці operators\_report. Вона перевіряє, чи дата звіту менша за 6 місяців від поточної. Якщо звіт був сформований за останні 6 місяців, функція виводить повідомлення і забороняє видалення.

**Бізнес-процес**

Управління звітністю кол-центру

Приклад роботи наведений на рисунку 6.36

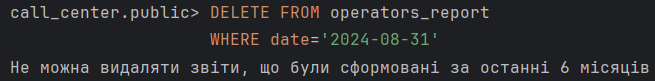


Рисунок 6.36– приклад роботи триггера e)

## Представлення

a)

CREATE VIEW total\_category\_durations AS  
SELECT  
 CASE  
 WHEN cs.duration < INTERVAL '3 mins' THEN 'Short'  
 WHEN cs.duration BETWEEN '3 mins' AND '7 mins' THEN 'Medium'  
 ELSE 'Long'  
 END AS call\_duration\_category, COUNT(\*)  
FROM calls cs  
GROUP BY call\_duration\_category;

**Призначення**  
Представлення total\_category\_durations категоризує тривалість дзвінків у три групи: короткі, середні та довгі, і підраховує кількість дзвінків у кожній категорії.

**Опис**  
Це представлення агрегує дані з таблиці calls, категоризуючи кожен дзвінок на основі його тривалості. Дзвінки діляться на три категорії: "Short" (короткі), "Medium" (середні), і "Long" (довгі), залежно від їх тривалості. Для кожної категорії підраховується кількість дзвінків.

**Бізнес-процес**

Управління звітністю та дзвінками

Приклад роботи наведений на рисунку 6.37

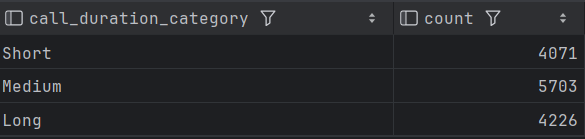


Рисунок 6.37– приклад роботи представлення a)

b)

CREATE VIEW durationsByMonth AS  
SELECT date,  
 SUM(calls\_duration)  
FROM operators\_report op  
GROUP BY date  
ORDER BY SUM(calls\_duration) DESC  
LIMIT 10;

**Призначення**  
Представлення durationsByMonth підсумовує тривалість дзвінків за кожен місяць і відображає 10 місяців з найбільшими загальними тривалістями дзвінків.

**Опис**  
Це представлення групує дані з таблиці operators\_report за датою і підсумовує тривалість дзвінків для кожного місяця. Потім воно сортує результати за сумарною тривалістю дзвінків у порядку спадання і обмежує виведення до 10 записів з найбільшими значеннями.

**Бізнес-процес**

Управління звітністю, дзвінками

Приклад роботи наведений на рисунку 6.38

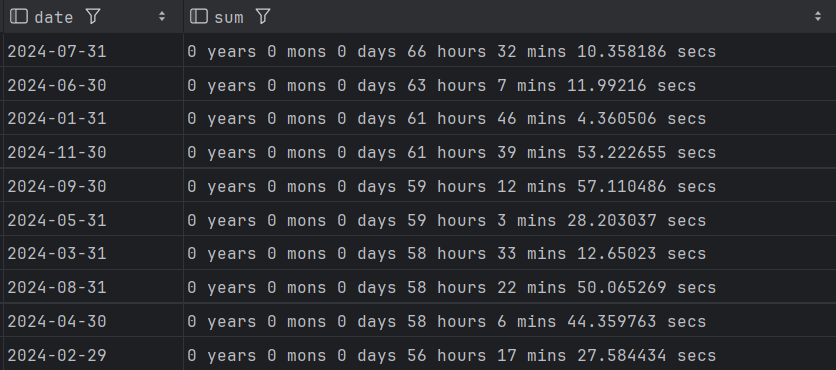


Рисунок 6.38– приклад роботи представлення b)

c)

CREATE VIEW number\_of\_calls\_by\_month AS  
SELECT o.name AS operator\_name, o.surname AS operator\_surname, TO\_CHAR(c.date, 'yyyy-mm') AS month, COUNT(c.operator\_id) AS number\_of\_calls  
FROM operators o  
INNER JOIN calls c ON o.id = c.operator\_id  
GROUP BY o.name, o.surname, month  
ORDER BY month DESC;

**Призначення**  
Представлення number\_of\_calls\_by\_month підсумовує кількість дзвінків, виконаних кожним оператором за кожен місяць.

**Опис**  
Це представлення об'єднує таблиці operators і calls, групуючи дані за ім'ям, прізвищем оператора та місяцем. Кількість дзвінків для кожного оператора обчислюється за допомогою функції COUNT, при цьому дата дзвінка перетворюється у формат 'yyyy-mm' за допомогою функції TO\_CHAR. Результати сортуються за місяцем у порядку спадання.

**Бізнес-процес**

Управління звітністю, дзвінками

Приклад роботи наведений на рисунку 6.39

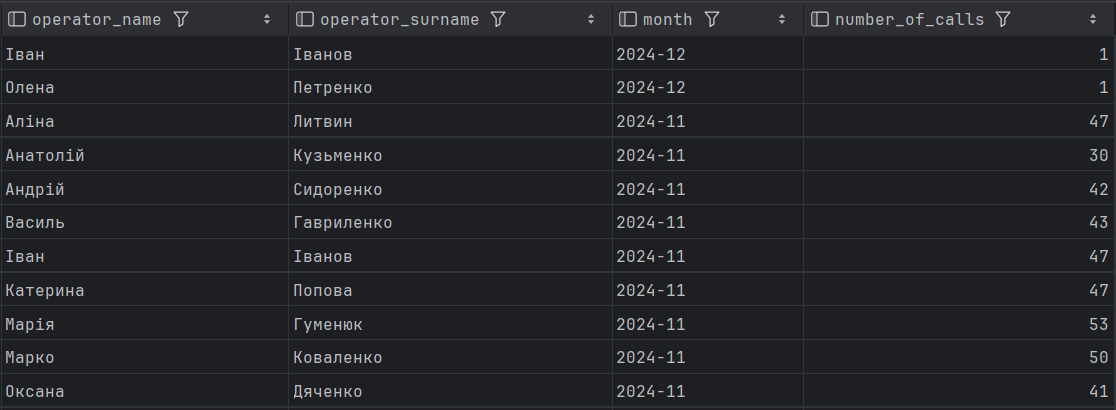


Рисунок 6.39– приклад роботи представлення c)

d)

CREATE VIEW top\_average\_salary AS  
SELECT o.id, o.name AS operator\_name, o.surname AS operator\_surname, AVG(opr.total\_salary) AS avg\_salary FROM operators o  
INNER JOIN operators\_report opr ON opr.operator\_id=o.id  
GROUP BY o.id, o.name, o.surname  
ORDER BY AVG(opr.total\_salary) DESC;

**Призначення**  
Представлення top\_average\_salary обчислює середню заробітну плату операторів на основі даних у таблиці operators\_report і надає рейтинг операторів за середнім доходом.

**Опис**  
Це представлення об'єднує таблиці operators та operators\_report через оператор INNER JOIN на основі operator\_id. Для кожного оператора обчислюється середнє значення total\_salary за допомогою функції AVG, і результати групуються за id, name та surname оператора. Дані сортуються за середнім значенням заробітної плати в порядку спадання.

**Бізнес-процес**

Управління звітністю кол-центру

Приклад роботи наведений на рисунку 6.40

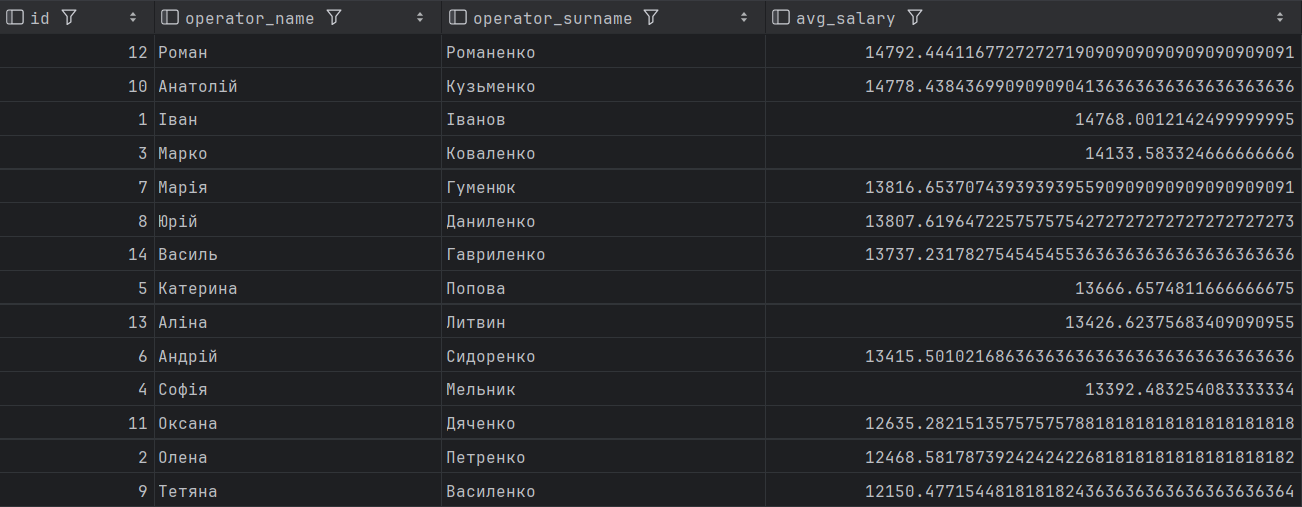


Рисунок 6.40– приклад роботи представлення d)

## Індекси

Індекси в базах даних є одним із ключових інструментів для оптимізації запитів. Вони дозволяють значно зменшити час виконання операцій пошуку та з'єднання даних, забезпечуючи швидкий доступ до необхідної інформації. Як своєрідний вказівник, індекс містить інформацію про розташування даних у таблиці, що дозволяє уникнути повного сканування таблиці.

a)

Створимо індекси для чотирьох полів таблиці calls(client\_id, operator\_id, date та duration)

SQL-скрипти для створення індексів:

CREATE INDEX idx\_on\_client\_id ON calls(client\_id);

CREATE INDEX idx\_on\_date\_calls ON calls USING btree (date);

CREATE INDEX idx\_on\_duration\_call ON calls USING btree (duration);

CREATE INDEX idx\_on\_operator\_id ON calls(operator\_id);

SQL-запит для перевірки оптимізації

EXPLAIN ANALYZE SELECT  
 o.name AS operator\_name,  
 o.surname AS operator\_surname,  
 c.name AS client\_name,  
 c.surname AS client\_surname,  
 cs.duration AS calls\_duration  
FROM operators o  
 INNER JOIN calls cs ON cs.operator\_id=o.id  
 INNER JOIN clients c ON cs.client\_id = c.id  
 WHERE cs.duration BETWEEN '3 mins' AND '6 mins';

Результат роботи до створення індексу наведений на рисунку 6.41

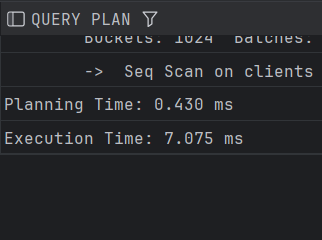


Рисунок 6.41– результат роботи запиту без індексів

Результат роботи після створення індексів наведений на рисунку 6.42

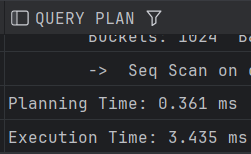


Рисунок 6.42– результат роботи запиту з індексами

b)

Створимо індекс для 2 полів таблиці issues(call\_Id, issue\_category\_id)

SQL-скрипт для створення індексу:

CREATE INDEX idx\_on\_call\_id ON issues USING btree (call\_id);

SQL-запит для перевірки оптимізації

EXPLAIN ANALYZE SELECT c.name, c.surname, c.email, COUNT(DISTINCT ca.id) AS total\_calls  
FROM clients c  
 INNER JOIN calls ca ON c.id = ca.client\_id  
 INNER JOIN issues i ON ca.id = i.call\_id  
WHERE i.issue\_category\_id = 2  
GROUP BY c.name, c.surname, c.email  
HAVING COUNT(DISTINCT ca.id) > 1;

Результат роботи до створення індексу наведений на рисунку 6.43

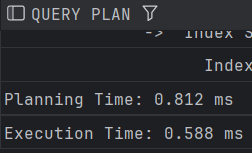


Рисунок 6.43– результат роботи запиту без індексів

Результат роботи після створення індексів наведений на рисунку 6.44

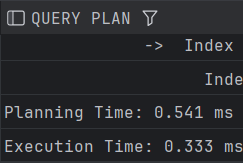


Рисунок 6.44– результат роботи запиту з індексами

Висновки до розділу

В ході виконання даного розділу було проведено основну роботу, а саме написання усіх sql select запитів, створення функцій/процедур, тригерів та представлень. Кожен з них несе за собою певну ціль, з якою він був створений. Для кожного запиту було описано призначення, його опис та бізнес процес, який він описує. В кінці було створено декілька індексів для оптимізації роботи запитів, що мають оператори WHERE, JOIN, GROUP BY та ORDER BY, було показано результат оптимізації, для цього використовувалася команда EXPLAIN ANALYZE, що надає характеристику запиту, до якого була використана.

# **КОРИСТУВАЧІ СИСТЕМИ**

Користувачі в базі даних є важливими для забезпечення належного управління доступом до даних та контролю за операціями, які можуть виконувати різні ролі. Розподіл прав доступу дозволяє зберігати безпеку та ефективність роботи з базою, адже кожен користувач має доступ лише до тих ресурсів, які йому необхідні для виконання своїх завдань, що мінімізує ризик несанкціонованих змін або витоку даних.

В даній системі було визначено 3 користувачі: оператор, адміністратор бази даних та менеджер.

**Оператор**

В його можливості входить тільки читання 3 таблиць, а саме issues, issue\_category та issue\_solutions. Ці три таблиці можуть тим чи іншим чином знадобитись оператору для відповідей на питання клієнтів під час дзвінків

SQL-код для створення користувача та надання йому відповідних прав:

CREATE ROLE call\_operator WITH LOGIN PASSWORD 'op\_password';

GRANT USAGE, SELECT, UPDATE ON ALL SEQUENCES IN SCHEMA public TO call\_manager;

GRANT USAGE ON SCHEMA public TO call\_operator;  
GRANT SELECT ON issue\_solutions, issues, issue\_categories TO call\_operator;

Приклад роботи:

Спроба звернутися до таблиці, що не визначена у правах доступу користувача показана на рисунку 7.1

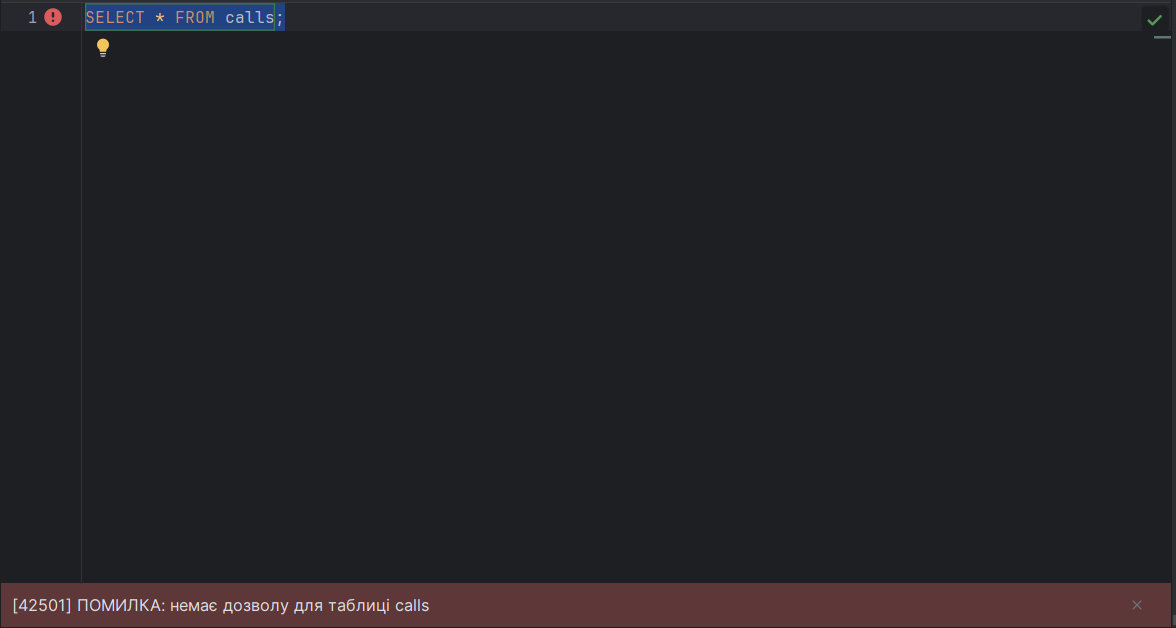


Рисунок 7.1 – спроба витягнути дані з таблиці calls

Спроба виконати недопустимі дії над таблицею,до якої у користувача є доступ тільки на читання показана на рисунку 7.2

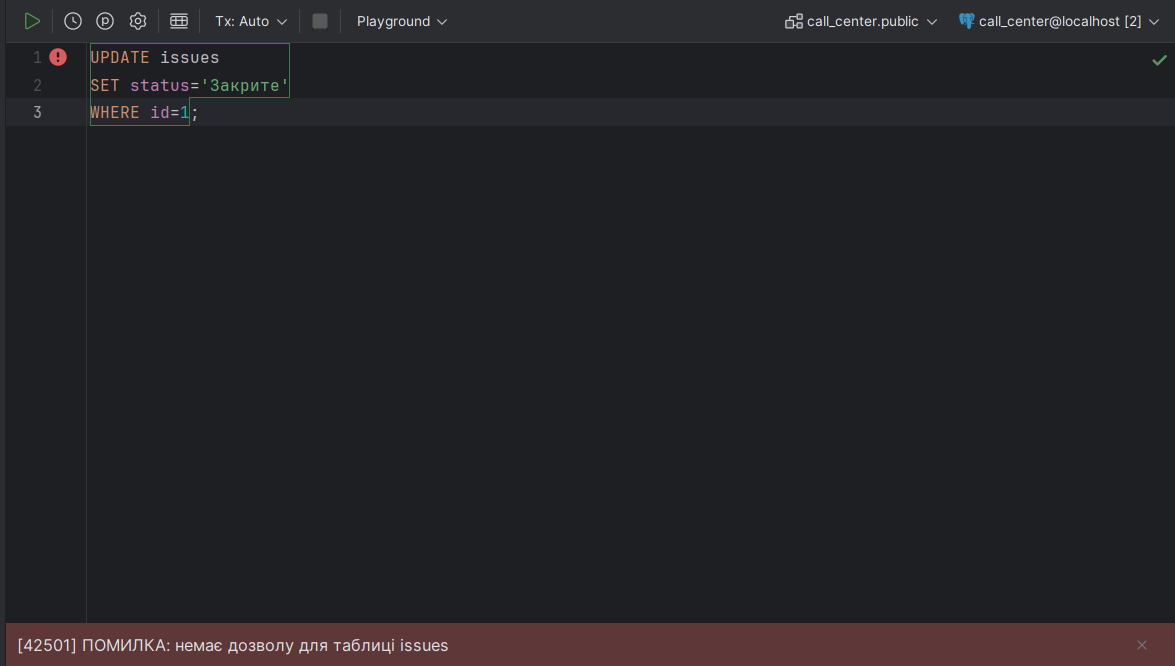


Рисунок 7.2 – спроба оновити дані в таблиці issues

Спроба виконати допустимі дії над доступної користувачеві таблицею наведена на рисунку 7.3

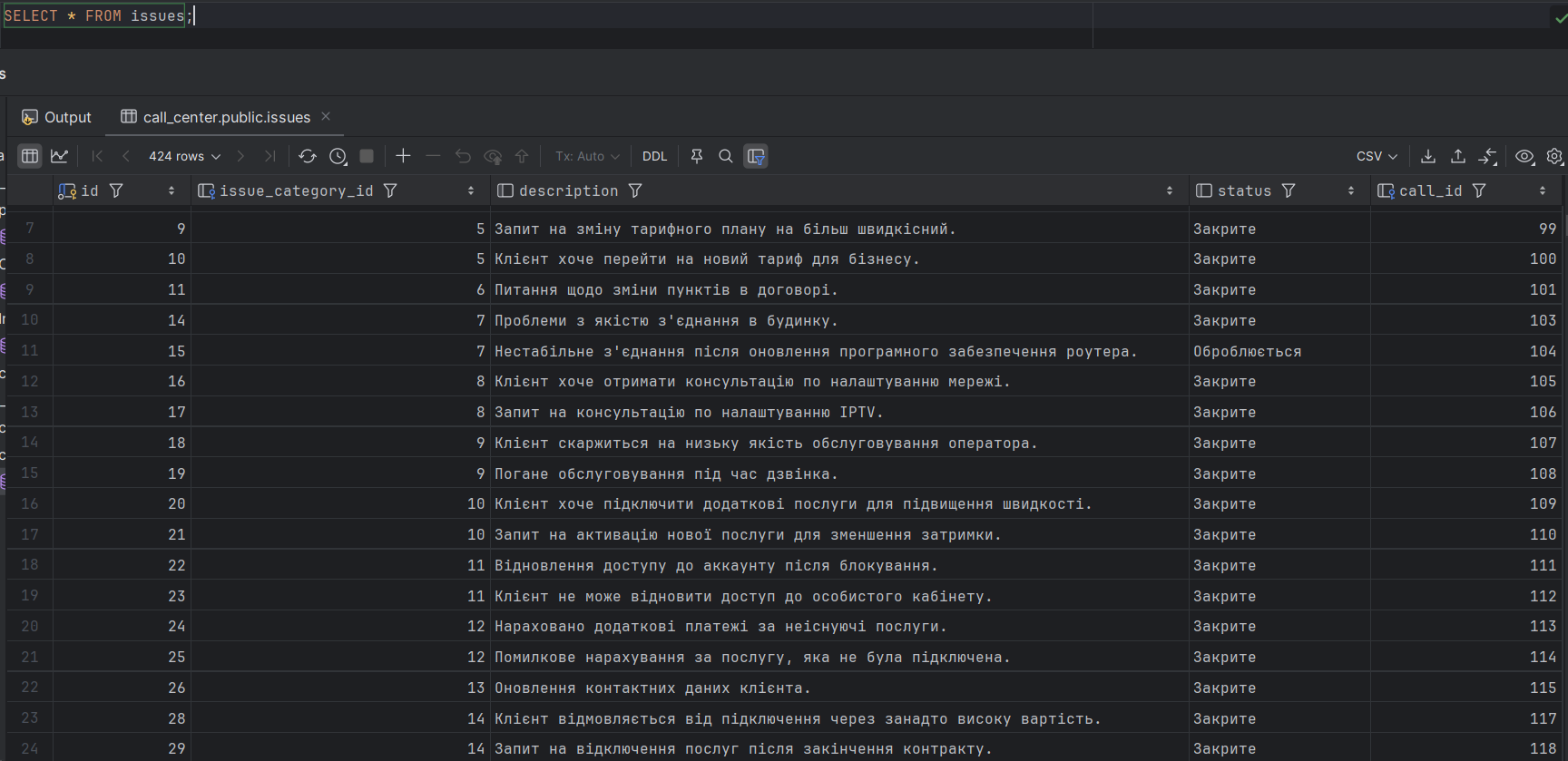


Рисунок 7.3 – спроба витягнути дані з таблиці issues

**Менеджер**

В його можливості входить вибірка, зміна та вставка значень в усі таблиці бази даних.

SQL-код для створення користувача та надання йому відповідних прав:

CREATE ROLE call\_manager WITH LOGIN PASSWORD 'manager\_password';

GRANT USAGE ON SCHEMA public TO call\_manager;  
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA public TO call\_manager;

Приклад роботи:

Спроба видалити дані з таблиці в базі даних наведена на рисунку 7.4

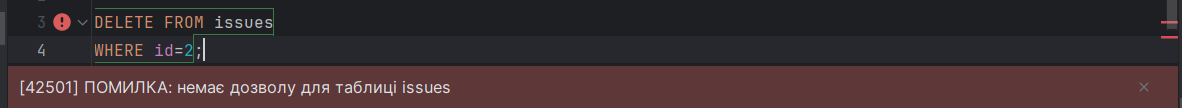


Рисунок 7.4 – спроба видалити дані з таблиці issues

Спроба зробити будь-які допустимі дії наведена на рисунку 7.5

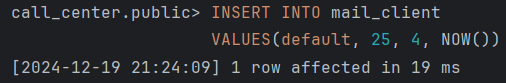


Рисунок 7.5 – спроба додати запис до таблиці mail\_client

**Адміністратор бази даних**

В його можливості входять будь-які дії над схемою в базі даних. Йому доступні майже всі можливості роботи з бд від оновлення таблиць до створення процедур та функцій.

SQL-код для створення користувача та надання йому відповідних прав:

GRANT CREATE, USAGE ON SCHEMA public TO db\_administrator;  
GRANT USAGE, SELECT, UPDATE ON ALL SEQUENCES IN SCHEMA public TO db\_administrator;  
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO db\_administrator;

Приклад роботи:

Спроба створення нової таблиці наведена на рисунку 7.6

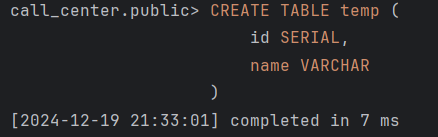


Рисунок 7.6 – спроба створення нової таблиці

Спроба видалення таблиці наведена на рисунку 7.7

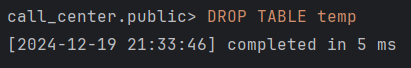


Рисунок 7.7 – спроба видалення таблиці

Висновки до розділу:

У цьому розділі було розглянуто принципи управління користувачами в базі даних та розподілу прав доступу між ними. Продемонстровано приклади створення користувачів, надання їм необхідних привілеїв і перевірки доступу до таблиць та інших об'єктів бази даних.

# **ВИСНОВКИ**

У результаті виконання лабораторної роботи було розроблено базу даних для підтримки діяльності кол-центру. У ході розробки була створена ER-модель, яка повністю описує структуру усіх таблиць, її атрибутів та зв’язків між таблицями. Після було розроблено саму базу даних, що складається з 11 табличок, кожна з яких виконує свою роль. Було визначено типи даних для кожного атрибута та зв’язки для кожної таблиці.

Кожна з таблиць мала обмеження на атрибутах для виконання умови цілісності даних. З обмежень можна виділити: первинний та зовнішній ключі, обмеження not null, unique, default, також було використане обмеження check для перевірки особливих кейсів. Кожен зовнішній ключ мав свою поведінку на видалення, зокрема cascade та set null. Також, до кожного унікального ідентифікатора кожної таблиці було додано генератор SERIAL, який відповідає за автоінкрементування,.

Після побудови та створення таблиць в базі даних було створено 20 sql select запитів, 10 процедур/функцій, 5 тригерів та 4 представлення, кожне з яких відіграє свою роль.

Також було додано індекси до часто використовуваних стовпців для більш швидкого доступу до них.

Для забезпечення безпеки в системі були додані користувачі, кожен з яких має свої права. Кожен з них може виконувати тільки ті дії з таблицями бд, які визначені їх привілеями, це забезпечує контроль над операціями та мінімізує ризик витоку даних.

Практичне використання бази даних передбачає інтеграцію її з інформаційними системами кол-центру, що дозволить автоматизувати роботу операторів, менеджерів і адміністраторів. Розроблена структура таблиць та реалізовані функції можуть бути легко адаптовані до конкретних потреб підприємства. Зокрема, база даних підтримує швидкий доступ до записів, ефективну обробку запитів клієнтів і забезпечення своєчасного аналізу даних для прийняття управлінських рішень.

Розробка даної системи допомогла значно покращити практичні навички з написання коду на sql, а також розуміння того, як правильно продумувати структуру таблиць, зв’язків між ними, знаходити помилки ще під час створення діаграми, а не в процесі реалізації бд.

# **СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Call centre – Wikipedia. URL: <https://en.wikipedia.org/wiki/Call_centre> (Дата звернення 15.12.2024)
2. ServiceNow. What is a contact center? URL: <https://www.servicenow.com/products/customer-service-management/what-is-a-contact-center.html> (Дата звернення 15.12.2024)
3. NetHunt. Що таке CRM-система? URL: <https://nethunt.ua/blog/shcho-takie-crm-sistiema-povnii-ghid-po-viboru-crm-dlia-pochatkivtsiv/> (Дата звернення 15.12.2024)
4. ECall. Інтелектуальні рішення для контакт-центрів. URL: <https://ecall.ua/> (Дата звернення 16.12.2024)
5. Call-Center. Послуги контакт-центрів в Україні. URL: <https://call-center.com.ua/uk/> (Дата звернення 16.12.2024)
6. Contactis. Професійні послуги контакт-центрів. URL: <https://contactis.ua/> (Дата звернення 16.12.2024)
7. PostgreSQL. Офіційна документація системи управління базами даних. URL: <https://www.postgresql.org/> (Дата звернення 16.12.2024)