НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
(повна назва інституту/факультету)

КАФЕДРА інформатики та програмної інженерії
(повна назва кафедри)

КУРСОВА РОБОТА

з дисципліни «Бази даних»
(назва дисципліни)

на тему: БАЗА ДАНИХ ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ CALL-ЦЕНТРУ

Студента (ки) 2 курсу ІП-35 групи спеціальності 121 «Інженерія програмного забезпечення»
Адаменка А.Б.
(прізвище та ініціали)

Керівник доц. Ліщук К. І
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____

Кількість балів: _____ Оцінка ECTS _____

Члени комісії

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

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

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

Київ – 2024 рік

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

Факультет Інформатики та обчислювальної техніки
(повна назва)

Кафедра Інформатики та програмної інженерії
(повна назва)

Дисципліна Бази даних

Курс __2___ Група __ІП-35__ Семестр ___3__

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

Адаменку Арсену Богдановичу
(прізвище, ім'я, по батькові)

1. Тема роботи ___БАЗА ДАНИХ ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ___ CALL-ЦЕНТРУ

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

2. Строк подання студентом роботи 15.12.2024

3. Вихідні дані до роботи ____Завдання на розробку база даних для роботи комапнії типу Call-цетру

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____23.11.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання курсового проекту	Строк виконання етапів проекту	Примітка
1	Аналіз предметного середовища	03.12.2024	
2	Побудова ER-моделі	05.12.2024	
3	Побудова реляційної схеми з ER-моделі	06.12.2024	
4	Створення бази даних, у форматі обраної системи управління базою даних	08.12.2024	
5	Створення користувачів бази даних	09.12.2024	
6	Імпорт даних з використанням засобів СУБД в створену базу даних	10.12.2024	
7	Створення мовою SQL запитів	18.12.2024	
8	Оптимізація роботи запитів	20.12.2024	
9	Оформлення пояснювальної записки	22.12.2023	
10	Захист курсової роботи	25.12.2024	

Студент _____ ___Адаменко А.Б._____
(підпис) (прізвище та ініціали)

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

ЗМІСТ

Вступ

В сучасному світі бази даних відіграють ключову роль у забезпеченні ефективної організації, зберігання та обробки інформації. Вони стали невід'ємною складовою діяльності будь-якої організації, що працює з великими обсягами даних. Одним із прикладів таких організацій є Call-центри [1], які щодня обробляють велику кількість звернень клієнтів, запитів на підтримку та інформаційних запитів. Для ефективної роботи таких центрів необхідна розробка оптимізованої бази даних, яка дозволить систематизувати дані, забезпечити багатокористувацький доступ і підтримувати високу швидкість обробки запитів.

Метою даної курсової роботи є створення бази даних для підтримки діяльності Call-центру. Ця база повинна забезпечувати зберігання інформації про клієнтів, дзвінки, запити, операторів та інші аспекти роботи Call-центру. Окрім цього, важливо враховувати вимоги до цілісності даних, зручності використання і масштабованості системи.

У процесі виконання роботи буде проведено аналіз предметної області, сформульовано бізнес-правила, розроблено ER-модель бази даних, побудовано реляційну схему та реалізовано функціональність для виконання ключових операцій. Результати роботи дозволять забезпечити ефективну підтримку роботи Call-центру, а також сприятимуть підвищенню якості обслуговування клієнтів і продуктивності операторів.

1 ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА

Предметне середовище Call-центру - це комплекс процесів, даних і взаємодій, які забезпечують обробку запитів клієнтів через різні канали комунікації. Ефективна робота Call-центру вимагає автоматизації таких процесів, як прийом дзвінків, реєстрація запитів, управління інформацією про клієнтів та підготовка звітності для аналізу продуктивності.

Основні аспекти предметного середовища:

1. Клієнти:
    1. Фізичні або юридичні особи, які звертаються до Call-центру для отримання допомоги;
    2. Інформація про клієнтів включає контактні дані, історію взаємодій, статус обробки запитів.
2. Оператори:
    1. Співробітники Call-центру, які приймають дзвінки, реєструють запити та надають консультації;
    2. Для кожного оператора фіксуються дані про навантаження, продуктивність і графік роботи.
3. Дзвінки та комунікації:
    1. Кожен вхідний або вихідний дзвінок фіксується в системі разом із деталями (тривалість, час, ідентифікатор оператора та клієнта);
    2. Можуть також враховуватися електронні повідомлення чи чати.
4. Запити клієнтів:
    1. Запит - це проблема чи питання, яке потрібно вирішити;
    2. Мають статуси (наприклад, "Новий", "В процесі", "Закрито") та результати вирішення.
5. Аналітика і звітність:
    1. Збір статистичних даних про кількість запитів, їхній статус, швидкість вирішення, оцінку клієнтів;
    2. Формування звітів для аналізу ефективності роботи операторів та покращення якості обслуговування.

6. Перевірки та обмеження полів:
    1. ФІО для клієнтів та операторів мають бути унікальними;
    2. Назви кожного департамента мають бути унікальними;
    3. Номер телефону та адреса електронної пошти мають бути унікальними для кожного клієнта;
    4. Рейтинг може находитися лише в певних числових границях: від 0 до 10.

База даних буде реалізовано на СУБД PostgreSQL [2].

PostgreSQL [2] є однією з найпопулярніших систем керування базами даних з відкритим вихідним кодом. Вибір PostgreSQL для розробки даної бази даних зумовлений її багатим функціоналом, високою продуктивністю та стабільністю. Основні причини вибору PostgreSQL описані нижче:

1. Підтримка складних бізнес-правил

PostgreSQL забезпечує широкі можливості для реалізації складних бізнес-правил завдяки:

1. Підтримці збережених процедур та функцій (PL/pgSQL), які дозволяють автоматизувати виконання логіки на рівні бази даних;
2. Реалізації тригерів, які забезпечують автоматичне реагування на зміну даних у таблицях;
3. Використанню обмежень цілісності даних, таких як унікальність, перевірки, зв’язки зовнішніх ключів і інші можливості.

2. Гнучкість моделювання даних

PostgreSQL підтримує різні типи даних, включаючи спеціалізовані, що дозволяє створювати оптимальну структуру даних. Це є важливим для моделювання таких типів, як робочий графік або типи змін.

3. Продуктивність та масштабованість:

PostgreSQL забезпечує високу продуктивність навіть для великих обсягів даних, що критично для автоматизації роботи Call-центру. Система також підтримує масштабованість, дозволяючи ефективно працювати як у невеликих проектах, так і в масштабних корпоративних рішеннях.

4. Багатокористувацький доступ:

PostgreSQL дозволяє налаштовувати права доступу для різних груп користувачів, що забезпечує:

1. Надійний контроль доступу до даних.
2. Високий рівень безпеки для розподілу ролей між операторами, адміністраторами та іншими користувачами.

5. Підтримка транзакцій:

PostgreSQL забезпечує повну підтримку транзакцій (ACID), що гарантує:

1. Надійність і відновлення даних у разі збоїв;
2. Узгодженість при виконанні багатокрокових операцій.

6. Інструменти для аналізу даних:

PostgreSQL включає розширені можливості роботи з запитами:

1. Оптимізатор запитів автоматично покращує їх виконання;
2. Підтримуються складні підзапити, агрегатні функції, представлення (VIEW) та індекси, що забезпечує швидкий доступ до даних;

7. Відкритий код та активна спільнота:

Відкритий вихідний код дозволяє адаптувати PostgreSQL під потреби конкретного проекту, а активна спільнота забезпечує доступ до багатої документації, регулярних оновлень та вирішення потенційних проблем.

PostgreSQL підтримує різні операційні системи (Linux, Windows, macOS), що робить її зручною для розгортання у різних середовищах.

Сутності в предметному середовищі мають наступні зв’язки:

1. Оператор 0,n --- 1 Департамент;
2. Оператор 1 --- 0,n Графік;
3. Оператор 1--- 0,n Інцидент;
4. Оператор 1--- 0,n Виклик;
5. Клієнт 1--- 0,n Виклик;
6. ТипВиклику 0,1 --- 1 Виклик;
7. Виклик 1 --- 0,n Запит;
8. Запит 1 --- 0,1 РезультатЗапиту;

9. Відгук 1 --- 0,1 Виклик;

Тому пираючись на основні сутності предметного середовища та його зв’язки можна побудувати наступну ER-модель бази даних для втоматичзованого обслуговування Call-центру в таблиці 1.1.

Таблиця 1.1 — таблиця сутностей та їх атрибутів

Таблиця	Ім’я поля	Тип даних	Розмір	Ключ	Опис
Department	ID_	int		PK	Ідентифікатор департаменту
Department	Name_	varchar	64		Назва департаменту
Department	Address_	varchar	64		Адреса знаходження
Department	Description_	varchar	128		Короткий опис
Operator	ID_	int		PK	Індентифікатор ператора
Operator	FIO	varchar	48		Повне ФІО
Operator	Department	int		FK	Принадлежність до департаменту
Client	ID_	int		PK	Індентифікатор клієнта
Client	FIO	varchar	48		Повне ФІО

Таблиця	Ім’я поля	Тип даних	Розмір	Ключ	Опис
Client	Phone	varchar	24		Номер телефону
Client	Email	varchar	64		Електронна пошта
Call_	ID_	int		PK	Індентифікатор дзвінка
Call_	Client	int		FK	Клієнт дзвінка
Call_	Operator	int		FK	Оператор дзвінка
Call_	StartDate	date			Дата початку дзвінка
Call_	Duration	decimal			Довжина дзвінка
CallType	ID_	int		PK	Ідентифікатор типа дзвінка
CallType	Call_	int		FK	Ключ на дзвінок
CallType	Type_	varchar	16		
CallType	Description_	varchar	48		
Feedback	ID_	int		PK	
Feedback	Client	int		FK	
Feedback	Call	int		FK	

Таблиця	Ім’я поля	Тип даних	Розмір	Ключ	Опис
Feedback	Rating	decimal			
Feedback	Comment	varchar	48		
Request	ID_	int		PK	Ідентифікатор запиту
Request	Call_	int		FK	Ключ на дзвінок
Request	RequestDate	date			Дата запиту
Request	Notes	varchar	72		Примітки для запиту
RequestResult	ID_	int		PK	Ідентифікатор результату запиту
RequestResult	Request	int		FK	Ключ на запит
RequestResult	ResolutionDate	date			Дата вирішення запиту
RequestResult	Result	varchar	64		Текстовий результат запиту
Shedule	ID_	int		PK	Ідентифікатор зміни
Shedule	Operator	int		FK	Оператор зміни
Shedule	ShiftDate	enum			Тип дати зміни

Таблиця	Ім’я поля	Тип даних	Розмір	Ключ	Опис
Shedule	ShiftType	enum			Тип часу зміни
Incident	ID_	int		PK	Ідентифікатор інцидента
Incident	Operator	enum		FK	Оператор інцидента
Incident	Date_	date			Дата інциденту
Incident	Description_	varchar	64		Опис інцидента
Incident	IsResolved	boolean			Вказує, чи є інцидент рзв’язанним

2 АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ ПРОДУКТІВ, КОТРІ ВИКОРИСТОВУЮТЬСЯ ДЛЯ АВТОМАТИЗАЦІЇ ЗАДАЧ ЗГІДНО ДОСЛІДЖУВАНОГО ПРЕДМЕТНОГО СЕРЕДОВИЩА

Автоматизація діяльності Call-центрів є ключовим фактором ефективної роботи організацій, що обробляють великий обсяг клієнтських запитів. Для цього на ринку доступний широкий спектр програмних продуктів, які забезпечують управління дзвінками, реєстрацію запитів, аналітику та управління базами даних клієнтів. Нижче наведено аналіз основних рішень, які використовуються в сучасних Call-центрах.

Популярні програмні продукти для автоматизації роботи Call-центрів:

1. Zoho Desk [3]:
    1. Опис: Багатофункціональна платформа для управління запитами клієнтів;
    2. Функціональність:
        1. Реєстрація та управління запитами;
        2. Автоматизація робочих процесів (workflow automation);
        3. Інтеграція з іншими системами (CRM, соціальні мережі, електронна пошта);
        4. Звіти та аналітика.
    3. Переваги:
        1. Простий у використанні інтерфейс;
        2. Підтримка багатоканального обслуговування;
    4. Недоліки:
        1. Обмежені функції в безкоштовній версії.
2. Freshdesk [4]:
    1. Опис: Хмарний програмний продукт для автоматизації Call-центрів та підтримки клієнтів;
    2. Функціональність:
        1. Управління дзвінками та запитами;

2. Підтримка бази знань для швидкого вирішення повторюваних запитів;
3. Можливість автоматизації завдань і маршрутизації запитів.

3. Переваги:
    1. Інтеграція зі сторонніми інструментами (Slack, Zoom, CRM);
    2. Підтримка мультиканального обслуговування.
4. Недоліки:
    1. Складність налаштувань для нових користувачів.

3. Bitrix24 [5]:
    1. Опис: Комплексна платформа для автоматизації бізнесу, включаючи Call-центри;
    2. Функціональність:
        1. CRM-система для управління клієнтськими базами;
        2. Інтеграція телефонії для обробки дзвінків;
        3. Інструменти для командної роботи та управління задачами.
    3. Переваги:
        1. Широкий функціонал;
        2. Доступність безкоштовної версії для малих команд.
    4. Недоліки:
        1. Висока складність інтерфейсу для новачків.
4. Zendesk [6]:
    1. Опис: Програмний продукт для підтримки клієнтів і автоматизації Call-центрів;
    2. Функціональність:
        1. Управління запитами через різні канали (телефон, електронна пошта, соціальні мережі);
        2. Аналітичні звіти про ефективність роботи операторів;
        3. Інтеграція з іншими продуктами для управління даними.

3. Переваги:
    1. Гнучкість налаштувань;
    2. Підтримка багатоканального обслуговування.
4. Недоліки:
    1. Висока вартість у порівнянні з конкурентами.

5. Asterisk [7]:
    1. Опис: Відкрите програмне забезпечення для управління телефонією;
    2. Функціональність:
        1. Управління телефонними дзвінками;
        2. Підтримка інтерактивного голосового меню (IVR);
        3. Запис дзвінків та маршрутизація викликів.
    3. Переваги:
        1. Відкритий код та можливість кастомізації;
        2. Низька вартість впровадження.
    4. Недоліки:
        1. Необхідність технічних знань для налаштування.

Порівняльний аналіз наведено вижче в таблиці 2.1:

Програмний продукт	Основні переваги	Основні недоліки	Цільова аудиторія
Zoho Desk	Інтеграція з CRM, аналітика	Обмежена безкоштовна	Малі та середні бізнеси
Freshdesk	Простота використання	Складність глибоких налаштувань	Малі бізнеси та стартапи
Bitrix24	Комплексний функціонал	Складність для початківців	Малі, середні та великі бізнеси
Zendesk	Гнучкість налаштувань	Висока вартість	Великі бізнеси
Asterisk	Відкритий код,	Потребує технічної	Технічно

Програмний продукт	Основні переваги	Основні недоліки	Цільова аудиторія
	кастомізація	експертизи	підготовлені компанії

Головними перевагами цієї розробки, що використовує цю бази даних, є простота, легкість використання її використання та можливість її розширити у подальшому.

3 ПОСТАНОВКА ЗАВДАННЯ

Метою курсової роботи є розробка бази даних для підтримки діяльності Call-центру, яка забезпечить систематизацію, зберігання та обробку даних про клієнтів, дзвінки, запити та результати їх вирішення. База даних повинна сприяти автоматизації роботи Call-центру, підвищенню продуктивності операторів та покращенню якості обслуговування клієнтів.

Основними завданнями під час написання курсової роботи є:

1. Аналіз предметної області:
    1. Вивчення основних бізнес-процесів Call-центру.
    2. Формулювання бізнес-правил для побудови бази даних.
    3. Визначення основних сутностей і зв’язків між ними.
2. Розробка бази даних:
    1. Побудова ER-моделі бази даних із визначенням сутностей, атрибутів і зв’язків.
    2. Опис бізнес-правил та обмежень для забезпечення цілісності даних.
    3. Розробка реляційної схеми бази даних із нормалізацією до третьої нормальної форми.
3. Створення бази даних у СУБД.
    1. Реалізація зв’язків між таблицями з використанням первинних і зовнішніх ключів.
    2. Визначення обмежень цілісності даних (унікальність, обов'язковість, перевірки).
4. Наведення прикладів маніпуляцій над базою даних та показати її можливості.

4 ПОБУДОВА ER-МОДЕЛІ

На малюнку 4.1 зображено ER-модель бази даних на основі аналізу предметної області в першому розділі курсової роботи. Зв’язки між об’єктами були створені на основі бізнес правил та вимог першого розділу на рисунку 4.1.

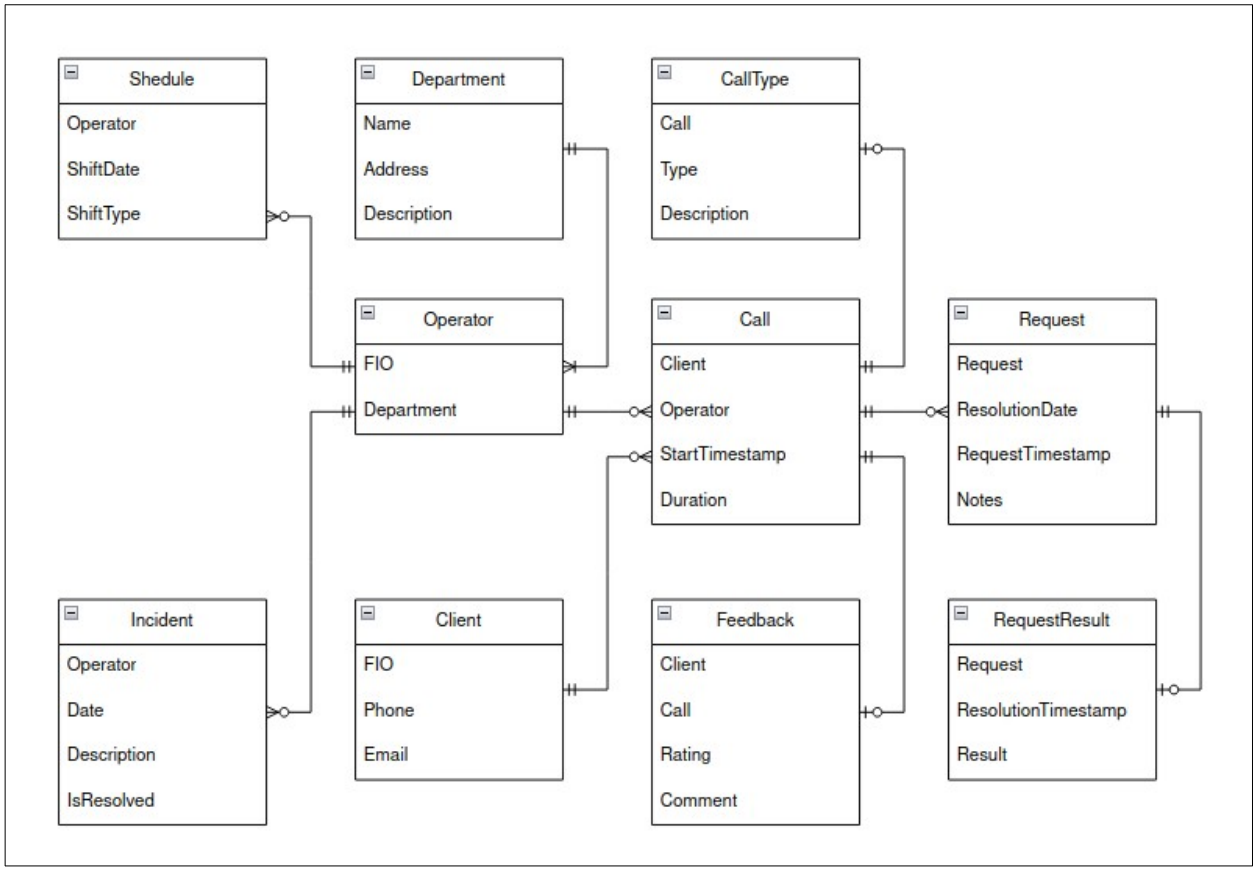

Рисунок 4.1 — ER-модель бази даних

На рисунку 4.2 навеждено генрацію ER-моделі вбудованими засобами СУБД після її створення за допомогою скриптів створення таблиць та їхніх обмежень.

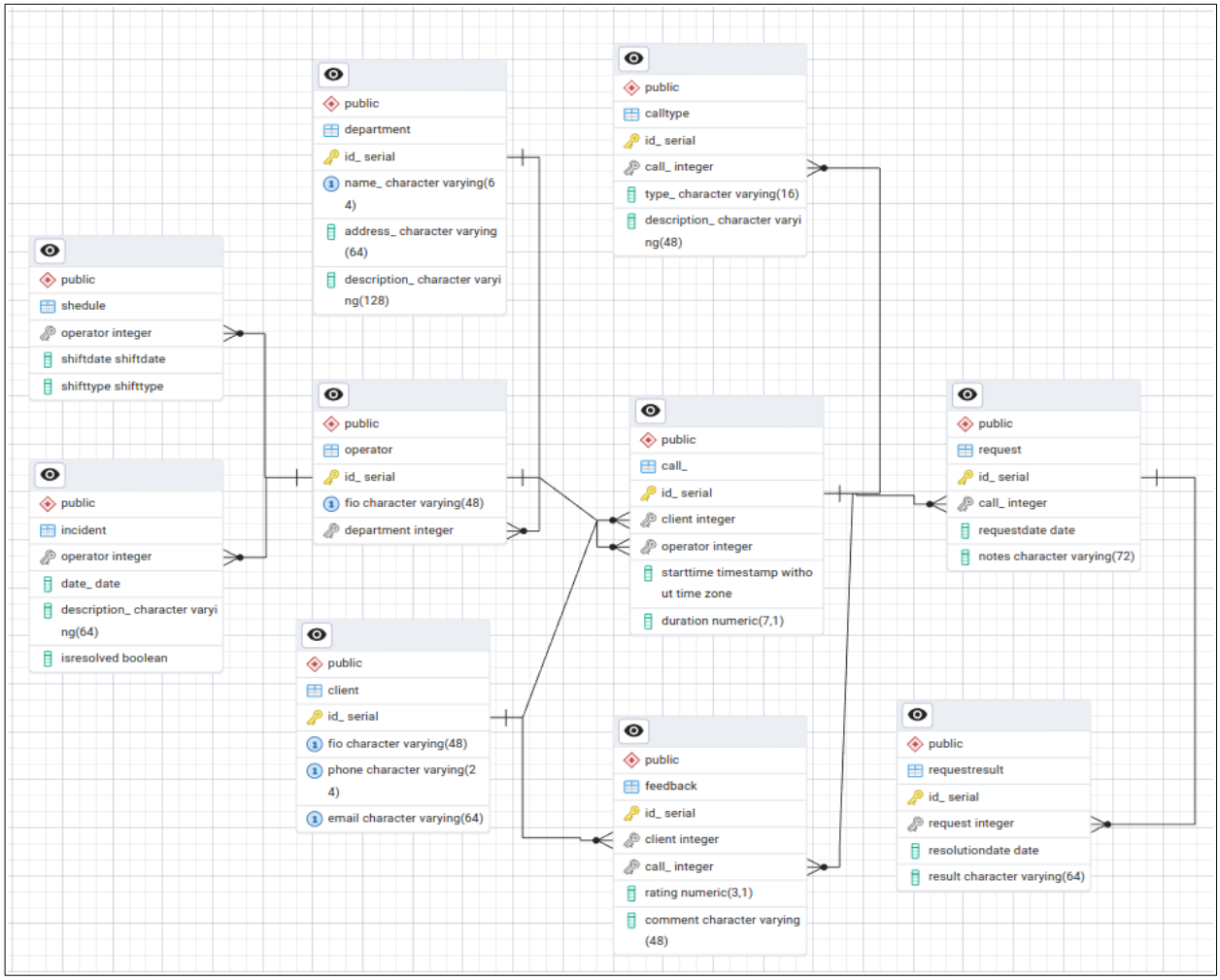

Рисунок 4.2 — побудова ER-моделі вбудованими засобами СУБД

5 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

Першим етапом програмної реалізації бази даних є створенням її каркаса. Нижче наведено вихідний код «скелета» бази даних, до нього входить створення таблиць та зв’язків між ними, а також задання бізнес правил нижче:

```
-- Buildings and invididuals

create table Department(
    ID_             serial primary key,
    Name_           varchar(64) unique,
    Address_        varchar(64),
    Description_    varchar(128)
);

create table Operator(
    ID_         serial primary key,
    FIO         varchar(48) not null unique,
    Department  int references Department(ID_) not null
);

create table Client(
    ID_         serial primary key,
    FIO         varchar(48) not null unique,
    Phone       varchar(24) unique,
    Email       varchar(64) unique
);

-- Call info

create table Call_(
    ID_              serial primary key,
    Client           int references Client(ID_)
        on delete cascade,
```

```
    Operator         int references Operator(ID_)
        on delete cascade,
    StartTimestamp   timestamp not null,
    Duration         decimal(7, 1)
);

create table CallType(
    ID_              serial primary key,
    Call_            int references Call_(ID_)
        on delete cascade,
    Type_            varchar(16) not null,
    Description_     varchar(48)
);

create table Feedback(
    ID_              serial primary key,
    Client           int references Client(ID_)
        on delete cascade,
    Call_            int references Call_(ID_)
        on delete cascade,
    Rating           decimal(2, 1)
            not null
            check (Rating between 0 and 10), -- from 0 to 10
    Comment          varchar(48)
);

-- Call requests

create table Request(
    ID_                  serial primary key,
    Call_                int references Call_(ID_)
        on delete cascade,
    RequestTimestamp     timestamp not null,
```

```
    Notes                varchar(72)
);

create table RequestResult(
    ID_                  serial primary key,
    Request              int references Request(ID_)
        on delete cascade,
    ResolutionTimestamp  timestamp,
    Result               varchar(64)
);

-- Operator working time

create type ShiftType as enum(
    'during_morning',
    'during_afternoon',
    'during_evening',
    'during_night'
);

create type ShiftDate as enum(
    'every_monday',
    'every_tuesday',
    'every_wednesday',
    'every_fourday',
    'every_friday',
    'every_saturday',
    'every_sunday'
);

create table Shedule(
    Operator        int references Operator(ID_)
        on delete cascade,
```

```
    ShiftDate       ShiftDate not null,
    ShiftType       ShiftType not null
);


create table Incident(
    Operator        int references Operator(ID_)
        on delete cascade,
    Date_           date not null,
    Description_    varchar(64),
    IsResolved      boolean not null
);
```

Нижче знаходиться запит для створення користувачів бази даних. Користувачами бази даних, їх ролі, їх привілеї а можливості та їх запити для створення:

1. Адміністратор:

```
CREATE ROLE admin_role WITH LOGIN PASSWORD 'adminka1234';

GRANT ALL PRIVILEGIES ON TABLES IN SCHEMA public TO admin_role;

ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL PRIVILEGIES ON TABLES TO
admin_role;

CREATE USER admin_user WITH PASSWORD 'adminka1234'; GRANT admin_role TO
admin_user;

GRANT SELECT ON ALL TABLES IN SCHEMA public TO admin_user;
```

2. Менеджер департаментів:

```
CREATE ROLE manager_role WITH LOGIN PASSWORD 'bossxyzabc123' NOSUPERUSER
NOCREATEDB NOCREATEROLE NOINHERIT;

GRANT SELECT ON ALL TABLES IN SCHEMA public TO manager_role;

ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO
manager_role;

GRANT UPDATE ON TABLE operator, schedule, incident TO manager_role;

CREATE USER manager_user WITH PASSWORD 'adminka1234'; GRANT manager_role TO
manager_user;

GRANT SELECT ON ALL TABLES IN SCHEMA public TO manager_user;
```

3. Оператор:

```
-- 1. Створення ролі для оператора
```

```
CREATE ROLE operator_role
WITH LOGIN PASSWORD 'ololo123'
NOSUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO operator_role;
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO
operator_role;
GRANT UPDATE ON TABLE request, call_ TO operator_role;
GRANT INSERT ON TABLE requestresult, call_, calltype TO operator_role;
GRANT ALL PRIVILEGES ON TABLE incident TO operator_role;
CREATE USER operator_user WITH PASSWORD 'secure_password';
GRANT operator_role TO operator_user;
```

4. Клієнт:

```
-- 1. Створення ролі для клієнта з обмеженими правами
CREATE ROLE client_role
WITH LOGIN PASSWORD 'client_password'
NOSUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT;
GRANT SELECT ON TABLE schedule, operator, call_, request, requestresult,
feedback, client, calltype TO client_role;
GRANT INSERT ON TABLE feedback TO client_role;
CREATE USER client_user WITH PASSWORD 'slava_ukraine';
GRANT client_role TO client_user;
GRANT SELECT ON TABLE schedule, operator, call_, request, requestresult,
feedback, client, calltype TO client_user;
```

6 РОБОТА З БАЗОЮ ДАНИХ

Тексти, опис запитів типу SELECT для отримання різноманітної інформації з бази даних Call-центру:

Запит №1:

Призначення: Отримання переліку всіх операторів та кількості дзвінків, які вони обробили.

Бізнес-правило: Оператор повинен фіксувати всі оброблені дзвінки для аналізу продуктивності.

Опис: Цей запит об'єднує таблиці Operator та Call_, щоб визначити кількість дзвінків, оброблених кожним оператором.

Результат:

Текст запиту:

```
SELECT o.FIO AS OperatorName, COUNT(c.ID_) AS TotalCalls
FROM Operator o
LEFT JOIN Call_ c ON o.ID_ = c.Operator
GROUP BY o.FIO;
```

Запит №2:

Призначення: Пошук клієнтів, які залишили відгуки із низькими оцінками.

Бізнес-правило: Компанія повинна реагувати на негативні відгуки для покращення якості обслуговування.

Опис: Використовуються таблиці Client та Feedback, щоб знайти клієнтів із оцінками нижче 4.5.

Результат:

Текст запиту:

```
SELECT c.FIO AS ClientName, f.Rating AS FeedbackRating, o.FIO AS
OperatorName, d.Name_ AS DepartmentName
SELECT cl.FIO AS ClientName, fb.Rating, fb.Comment
FROM Client cl
INNER JOIN Feedback fb ON cl.ID_ = fb.Client
```

```
WHERE fb.Rating < 4.5;
```

Запит №3:

Призначення: Отримання списку операторів, які працювали у певну зміну.

Бізнес-правило: Керівництво має знати, хто працював у конкретний період.

Опис: Використовуються таблиці Operator та Shedule для фільтрації операторів за типом зміни.

Результат:

Текст запиту:

```
SELECT o.FIO AS OperatorName, s.ShiftType
FROM Operator o
INNER JOIN Shedule s ON o.ID_ = s.Operator
WHERE s.ShiftType = 'during_evening';
```

Запит №4:

Призначення: Пошук дзвінків, що були здійснені певним клієнтом у конкретний період.

Бізнес-правило: Інформація про дзвінки клієнта повинна бути доступна для аналізу його активності.

Опис: Таблиці Client та Call_ використовуються для пошуку дзвінків за клієнтом та датою.

Результат:

Текст запиту:

```
SELECT c.StartDate, c.Duration, op.FIO AS OperatorName
FROM Call_ c
INNER JOIN Client cl ON c.Client = cl.ID_
INNER JOIN Operator op ON c.Operator = op.ID_
WHERE cl.FIO = 'Іван Петренко' AND c.StartDate BETWEEN '2024-01-01' AND
'2024-12-31';
```

Опис:

Результат виконання:

Запит №5:

Запит 5

Призначення: Визначення операторів, які отримували відгуки із середнім рейтингом вище 6.5.

Бізнес-правило: Компанія має заохочувати операторів із високими результатами обслуговування.

Опис: Використовується підзапит для обчислення середнього рейтингу по кожному оператору.

Результат:

Текст запиту:

```
SELECT o.FIO AS OperatorName, AVG(fb.Rating) AS AverageRating
FROM Operator o
INNER JOIN Call_ c ON o.ID_ = c.Operator
INNER JOIN Feedback fb ON fb.Call_ = c.ID_
GROUP BY o.FIO
HAVING AVG(fb.Rating) > 6.5;
```

Запит №6:

Призначення: Перегляд дзвінків, що мають відкриті запити на обробку.

Бізнес-правило: Всі запити повинні мати статус завершення або бути відкритими для обробки.

Опис: Об'єднуються таблиці Call_ та Request, фільтруються запити без дати завершення.

Результат:

Текст запиту:

```
SELECT c.ID_ AS CallID, r.RequestDate, r.Notes
FROM Call_ c
INNER JOIN Request r ON c.ID_ = r.Call_
WHERE r.ResolutionDate IS NULL;
```

Запит №7:

Призначення: Пошук клієнтів, які здійснили дзвінки тривалістю більше 1 години.

Бізнес-правило: Довгі дзвінки можуть свідчити про складні проблеми клієнтів.

Опис: Таблиці Call_ та Client використовуються для аналізу дзвінків клієнтів.

Результат:

Текст запиту:

```
SELECT cl.FIO AS ClientName, c.StartDate, c.Duration
FROM Call_ c
INNER JOIN Client cl ON c.Client = cl.ID_
WHERE c.Duration > 2400;
```

Запит №8:

Призначення: Отримання списку клієнтів із кількістю відкритих запитів.

Бізнес-правило: Керівництво повинно знати, які клієнти мають невирішені проблеми.

Опис: Підзапит рахує кількість запитів для кожного клієнта.

Результат:

Текст запиту:

```
SELECT cl.FIO AS ClientName, (
  SELECT COUNT(*)
  FROM Request r
  WHERE r.Call_ IN (
    SELECT c.ID_
    FROM Call_ c
    WHERE c.Client = cl.ID_
  ) AND r.ResolutionDate IS NULL
) AS OpenRequests
FROM Client cl;
```

Запит №9:

Призначення: Перевірка графіку роботи операторів для уникнення конфліктів змін.

Бізнес-правило: Оператори не повинні працювати більше однієї зміни за день.

Опис: Використовується групування для перевірки кількості змін оператора на день.

Результат:

Текст запиту:

```
SELECT s.Operator, s.ShiftDate, COUNT(s.ShiftType) AS ShiftsPerDay
FROM Shedule s
GROUP BY s.Operator, s.ShiftDate
HAVING COUNT(s.ShiftType) > 1;
```

Запит №10:

Призначення: Отримання кількості дзвінків, які отримали оцінку вище 7, по кожному відділу.

Бізнес-правило: Відділи з високими оцінками мають відзначатися як приклад для інших.

Опис: Об'єднання таблиць Department, Operator, Call_ та Feedback для аналізу оцінок.

Результат:

Текст запиту:

```
SELECT d.Name_ AS DepartmentName, COUNT(fb.ID_) AS HighRatedCalls
FROM Department d
INNER JOIN Operator o ON d.ID_ = o.Department
INNER JOIN Call_ c ON o.ID_ = c.Operator
INNER JOIN Feedback fb ON c.ID_ = fb.Call_
WHERE fb.Rating > 7
GROUP BY d.Name_;
```

Запит №11:

Призначення: Отримання списку операторів, які обробляли дзвінки певного клієнта.

Бізнес-правило: Інформація про взаємодії клієнтів з операторами повинна бути доступною.

Опис: Таблиці Call_ та Operator використовуються для визначення операторів, які працювали з клієнтом.

Результат:

Текст запиту:

```
SELECT DISTINCT o.FIO AS OperatorName
FROM Call_ c
INNER JOIN Operator o ON c.Operator = o.ID_
WHERE c.Client = (SELECT ID_ FROM Client WHERE FIO = 'Іван Петренко');
```

Запит №12:

Призначення: Пошук клієнтів, які не залишали жодного відгуку.

Бізнес-правило: Компанія повинна стимулювати клієнтів залишати відгуки для аналізу якості обслуговування.

Опис: Використовується підзапит для перевірки відсутності записів у таблиці Feedback.

Результат:

Текст запиту:

```
SELECT cl.FIO AS ClientName
FROM Client cl
WHERE NOT EXISTS (
  SELECT 1
  FROM Feedback fb
  WHERE fb.Client = cl.ID_
);
```

Запит №13:

Призначення: Перегляд середньої тривалості дзвінків по кожному оператору.

Бізнес-правило: Керівництво має знати, які оператори витрачають більше часу на обслуговування.

Опис: Використовується агрегація для обчислення середньої тривалості дзвінків.

Результат:

Текст запиту:

```
SELECT o.FIO AS OperatorName, AVG(c.Duration) AS AverageDuration
FROM Operator o
INNER JOIN Call_ c ON o.ID_ = c.Operator
GROUP BY o.FIO;
```

Запит №14:

Призначення: Пошук клієнтів, які здійснили дзвінки із певного відділу.

Бізнес-правило: Взаємодії клієнтів із конкретними відділами повинні бути відстежені.

Опис: Об'єднання таблиць Client, Call_ та Operator для фільтрації за відділом.

Результат:

Текст запиту:

```
SELECT cl.FIO AS ClientName
FROM Client cl
INNER JOIN Call_ c ON cl.ID_ = c.Client
INNER JOIN Operator o ON c.Operator = o.ID_
WHERE o.Department = (SELECT ID_ FROM Department WHERE Name_ LIKE '%Cats%');
```

Запит №15:

Призначення: Визначення найбільш активного клієнта за кількістю дзвінків.

Бізнес-правило: Компанія має знати своїх ключових клієнтів для кращого обслуговування.

Опис: Групування таблиці Call_ для обчислення кількості дзвінків по кожному клієнту.

Результат:

Текст запиту:

```
SELECT cl.FIO AS ClientName, COUNT(c.ID_) AS TotalCalls
FROM Client cl
INNER JOIN Call_ c ON cl.ID_ = c.Client
GROUP BY cl.FIO
ORDER BY COUNT(c.ID_) DESC
LIMIT 1;
```

Запит №16:

Призначення: Аналіз частоти звернень клієнтів із певним типом дзвінків.

Бізнес-правило: Необхідно знати, які типи дзвінків найбільш популярні серед клієнтів.

Опис: Об'єднання таблиць Call_, CallType та Client для аналізу популярності типів дзвінків.

Результат:

Текст запиту:

```
SELECT ct.Type_, COUNT(c.ID_) AS TotalCalls
FROM CallType ct
INNER JOIN Call_ c ON ct.Call_ = c.ID_
INNER JOIN Client cl ON c.Client = cl.ID_
GROUP BY ct.Type_
ORDER BY COUNT(c.ID_) DESC;
```

Запит №17:

Призначення: Підрахунок середнього рейтингу дзвінків для кожного відділу.

Бізнес-правило: Відділи повинні забезпечувати якісний сервіс на основі зворотного зв'язку клієнтів.

Словесний опис: Цей запит обчислює середній рейтинг, який клієнти залишили для дзвінків операторів кожного відділу.

Результат:

Текст запиту:

```
SELECT
```

```
    d.Name_ AS DepartmentName,
    AVG(f.Rating) AS AverageRating
FROM
    Department d
JOIN
    Operator o ON d.ID_ = o.Department
JOIN
    Call_ c ON o.ID_ = c.Operator
JOIN
    Feedback f ON c.ID_ = f.Call_
GROUP BY
    d.Name_;
```

Запит №18:

Призначення: Пошук операторів, які мають досвід роботи в усіх можливих типах змін.

Бізнес-правило: Оператори повинні бути готові працювати в різних робочих змінах для забезпечення гнучкості графіка.

Словесний опис: Цей запит знаходить операторів, які працювали в кожному типі змін (ранкова, денна, вечірня, нічна).

Результат:

Текст запиту:

```
SELECT
    s.Operator
FROM
    Shedule s
GROUP BY
    s.Operator
HAVING
    COUNT(DISTINCT s.ShiftType) = (SELECT COUNT(DISTINCT ShiftType) FROM
Shedule);
```

Запит №19:

-- Призначення: Визначення найдовших дзвінків для кожного оператора.

Бізнес-правило: Оператори повинні підтримувати якісний сервіс незалежно від тривалості дзвінків.

Словесний опис: Цей запит показує максимальну тривалість дзвінка для кожного оператора.

Результат:

Текст запиту:

```
SELECT
    o.FIO AS OperatorName,
    MAX(c.Duration) AS LongestCall
FROM
    Operator o
JOIN
    Call_ c ON o.ID_ = c.Operator
GROUP BY
    o.FIO;
```

Представлення №1:

Призначення: Обчислення мінімального, середнього та максимального рейтингу дзвінків для кожного оператора з виключенням екстремальних оцінок.

Бізнес-правило: Ранжуються оцінки дзвінків оператора по зростанню та спаданні, та обчислюються статистики за виключенням найгірших і найкращих 20% оцінок.

Опис: Для обчислення використовуються лише оцінки середнього діапазону, виключаючи 20% найгірших та 20% найкращих для деякої кількісті величин для запобіганню використання надто рідких даних для того, щоб вони були використані.

Результат:

Текст запиту:

```
CREATE VIEW OperatorRatingStatistic
WITH RankedFeedback AS (
    SELECT
        o.FIO AS OperatorFIO,
```

```
        f.Rating,
        ROW_NUMBER() OVER (PARTITION BY o.FIO ORDER BY f.Rating ASC) AS
RowAsc,
        ROW_NUMBER() OVER (PARTITION BY o.FIO ORDER BY f.Rating DESC) AS
RowDesc,
        COUNT(*) OVER (PARTITION BY o.FIO) AS TotalRows
    FROM Operator o
    INNER JOIN Call_ c ON o.ID_ = c.Operator
    INNER JOIN Feedback f ON c.ID_ = f.Call_
)
SELECT
    OperatorFIO,
    ROUND(MIN(CASE WHEN RowAsc > TotalRows * 0.1 AND RowDesc > TotalRows *
0.2 THEN Rating END), 1) AS MinimalCallRating,
    ROUND(AVG(CASE WHEN RowAsc > TotalRows * 0.1 AND RowDesc > TotalRows *
0.2 THEN Rating END), 1) AS AverageCallRating,
    ROUND(MAX(CASE WHEN RowAsc > TotalRows * 0.1 AND RowDesc > TotalRows *
0.2 THEN Rating END), 1) AS MaximumCallRating
FROM RankedFeedback
GROUP BY OperatorFIO
ORDER BY AverageCallRating DESC, OperatorFIO;
```

Представлення №2:

Призначення: Представлення історії дзвінків для аналізу якості обслуговування.

Бізнес-правило: Кожен дзвінок має бути зареєстрований, можливий аналіз якості обслуговування.

Опис: Показує список дзвінків клієнтів із деталями оператора та зворотного зв'язку (якщо є).

Результат:

Текст запиту:

```
CREATE VIEW ClientCallHistory AS
SELECT
    c.FIO AS ClientName,
```

```
    o.FIO AS OperatorName,
    call.StartDate,
    call.Duration,
    fb.Rating,
    fb.Comment
FROM
    Call_ call
    JOIN Client c ON call.Client = c.ID_
    JOIN Operator o ON call.Operator = o.ID_
    LEFT JOIN Feedback fb ON call.ID_ = fb.Call_;
```

Представлення №3:

Призначення: Відображення графіка роботи операторів.

Бізнес-правило: Кожен оператор повинен мати заздалегідь визначений графік роботи.

Опис: Відображає інформацію про графік роботи кожного оператора.

Результат:

Текст запиту:

```
CREATE VIEW OperatorSchedules AS
SELECT
    o.FIO AS OperatorName,
    s.ShiftDate,
    s.ShiftType
FROM
    Shedule s
    JOIN Operator o ON s.Operator = o.ID_;
```

Тригер №1:

Призначення: Забезпечення цілісності даних у таблиці `Feedback` шляхом перевірки рейтингу.

Бізнес-правило: Рейтинг повинен бути в межах від 0 до 10 включно, щоб уникнути некоректних даних.

Опис: Тригер перевіряє значення рейтингу перед вставкою або оновленням запису в таблиці `Feedback`. Якщо значення не відповідає правилам, операція скасовується.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION validate_feedback_rating() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.Rating < 0 OR NEW.Rating > 10 THEN
        RAISE EXCEPTION 'Rating must be between 0 and 10.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_feedback_rating
BEFORE INSERT OR UPDATE ON Feedback
FOR EACH ROW
EXECUTE FUNCTION validate_feedback_rating();
```

Тригер №2:

Призначення: Автоматичне створення запису в логах під час змін у таблиці `Incident`.

Бізнес-правило: Всі зміни в інцидентах повинні бути задокументовані для подальшого аналізу.

Опис: Після оновлення запису в таблиці `Incident`, тригер додає інформацію про старі значення до таблиці `IncidentLog`.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION log_incident_changes() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO IncidentLog (Operator, Date_, Description_, IsResolved,
ChangeTime)
```

```
    VALUES (OLD.Operator, OLD.Date_, OLD.Description_, OLD.IsResolved,
now());
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER track_incident_changes
AFTER UPDATE ON Incident
FOR EACH ROW
EXECUTE FUNCTION log_incident_changes();
```

Тригер №3:

Призначення: Запобігання створенню перекриття змін для операторів у таблиці `Shedule`.

Бізнес-правило: Оператор не може працювати у кількох змінах одночасно в один день.

Опис: Перед вставкою або оновленням запису в таблиці `Shedule`, тригер перевіряє наявність перекриття змін. Якщо таке перекриття існує, операція скасовується.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION prevent_shift_overlap() RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1 FROM Shedule
        WHERE Operator = NEW.Operator
          AND ShiftDate = NEW.ShiftDate
          AND ShiftType = NEW.ShiftType
    ) THEN
        RAISE EXCEPTION 'Shift overlap detected for Operator ID %.',
NEW.Operator;
    END IF;
    RETURN NEW;
END;
```

```
$$ LANGUAGE plpgsql;


CREATE TRIGGER check_shift_overlap
BEFORE INSERT OR UPDATE ON Shedule
FOR EACH ROW
EXECUTE FUNCTION prevent_shift_overlap();
```

Тригер №4:

Призначення: Автоматично коригувати рейтинг у відгуках клієнтів, щоб значення залишалося в межах від 0 до 10.

Бізнес-правило: Рейтинг клієнта повинен бути валідним і завжди перебувати в межах дозволеного діапазону (0–10).

Опис: Цей тригер використовується для перевірки даних перед їх додаванням у таблицю feedback або під час оновлення. Якщо клієнт вводить значення рейтингу, що виходить за межі дозволеного діапазону (менше 0 або більше 10), тригер автоматично коригує його до найближчого граничного значення (0 або 10 відповідно).

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION enforce_feedback_rating()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.Rating < 0 THEN
        NEW.Rating := 0;
    ELSIF NEW.Rating > 10 THEN
        NEW.Rating := 10;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER enforce_feedback_rating_trigger
BEFORE INSERT OR UPDATE ON feedback
FOR EACH ROW
EXECUTE FUNCTION enforce_feedback_rating();
```

Тригер №5:

Призначення: Логувати зміни даних операторів у спеціальну таблицю, щоб забезпечити контроль за оновленнями.

Бізнес-правило: Всі зміни в таблиці operator повинні бути зареєстровані, щоб у разі виникнення проблем можна було відстежити, хто і коли їх вніс.

Опис: Триггер записує зміни в полі FIO (повне ім'я оператора) до таблиці operator_log. У цій таблиці фіксується ідентифікатор оператора, старе і нове значення імені, а також дата внесення змін. Це дозволяє зберігати історію оновлень даних для аудиту.

Результат:

Текст запиту:

```
CREATE TABLE OperatorLog(
    LogID       serial PRIMARY KEY,
    OperatorID  int NOT NULL,
    OldFIO      varchar(48),
    NewFIO      varchar(48),
    ChangeDate  timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TRIGGER log_operator_changes_trigger
AFTER UPDATE OF FIO ON operator
FOR EACH ROW
EXECUTE FUNCTION log_operator_changes();
```

Функція №1:

Призначення: Забезпечити автоматичне позначення інциденту як вирішеного, якщо у полі Description_ вказано слово "resolved".

Бізнес-правило: Якщо інцидент позначається як вирішений, це повинно відображатись у базі даних.

Опис: Функція перевіряє зміст поля Description_ на наявність ключового слова "resolved" та встановлює значення IsResolved = TRUE.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION set_incident_resolved()
RETURNS TRIGGER AS $$
BEGIN
    IF POSITION('resolved' IN NEW.Description_) > 0 THEN
        NEW.IsResolved := TRUE;
```

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

Функція №2:

Призначення: Запобігти внесенню змін у запис після його остаточного затвердження (Notes містить слово "finalized").

Бізнес-правило: Остаточно затверджені запити не можуть бути змінені.

Опис: Перед оновленням перевіряє вміст поля Notes. Якщо запис має статус "finalized", оновлення забороняється.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION restrict_request_updates()
RETURNS TRIGGER AS $$
BEGIN
    IF POSITION('finalized' IN OLD.Notes) > 0 THEN
        RAISE EXCEPTION 'Request updates are restricted after finalization';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №3:

Призначення: Гарантувати, що рейтинг знаходиться у межах від 0 до 10.

Бізнес-правило: Відгуки мають бути валідними за числовою шкалою.

Опис: Функція автоматично виправляє значення рейтингу до допустимих меж.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION enforce_feedback_rating()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.Rating < 0 THEN
        NEW.Rating := 0;
    ELSIF NEW.Rating > 10 THEN
        NEW.Rating := 10;
    END IF;
    RETURN NEW;
```

```
END;
$$ LANGUAGE plpgsql;
```

Функція №4:

Призначення: Логувати зміни імен операторів для забезпечення прозорості операцій.

Бізнес-правило: Ім'я оператора є критичною інформацією, всі зміни мають бути відстежені.

Опис: Функція створює запис у таблиці operator_log із зазначенням старого та нового імені після кожної зміни.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION log_operator_changes()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO operator_log (OperatorID, OldFIO, NewFIO)
    VALUES (OLD.ID_, OLD.FIO, NEW.FIO);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №5:

Призначення: Гарантувати, що кожен дзвінок має асоційований тип.

Бізнес-правило: Жоден дзвінок не може залишитись без типу у системі.

Опис: При створенні нового дзвінка функція автоматично додає запис у таблицю CallType із зазначенням типу за замовчуванням "general".

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION auto_assign_calltype()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO CallType (Call_, Type_, Description_)
    VALUES (NEW.ID_, 'general', 'Automatically assigned call type');
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №6:

Призначення: Гарантувати, що кожен запит має пов’язаний результат.

Бізнес-правило: Жоден запит не може залишитися без відповідного запису результату.

Опис: При створенні нового запису в таблиці Request автоматично додається запис у таблицю RequestResult із початковими значеннями.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION auto_create_request_result()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO RequestResult (Request, ResolutionDate, Result)
    VALUES (NEW.ID_, NULL, 'Pending');
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №7:

Призначення: Автоматично обчислювати тривалість дзвінка після його завершення.

Бізнес-правило: Тривалість дзвінка має бути правильно зафіксована для аналізу роботи операторів.

Опис: При оновленні StartDate функція обчислює та оновлює значення Duration.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION calculate_call_duration()
RETURNS TRIGGER AS $$
BEGIN
    NEW.Duration := EXTRACT(EPOCH FROM (NOW() - NEW.StartDate)) / 60.0; -- в
хвилинах
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №8:

Призначення: Заборонити додавання клієнтів із однаковими телефоном та електронною адресою.

Бізнес-правило: У базі даних не повинно бути дублюючих записів клієнтів.

Опис: Функція перевіряє новий запис на унікальність телефону та email. Якщо знайдено дублюючі дані, операція скасовується.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION prevent_duplicate_clients()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1 FROM Client
        WHERE Phone = NEW.Phone OR Email = NEW.Email
    ) THEN
        RAISE EXCEPTION 'Duplicate client entry is not allowed';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №9:

Призначення: Забезпечити каскадне видалення типів дзвінків при видаленні запису дзвінка.

Бізнес-правило: Типи дзвінків не повинні залишатися у базі без відповідних записів у таблиці Call_.

Опис: Після видалення дзвінка автоматично видаляються записи у таблиці CallType.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION cascade_delete_calltype()
RETURNS TRIGGER AS $$
BEGIN
    DELETE FROM CallType WHERE Call_ = OLD.ID_;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

Функція №10:

Призначення: Перераховувати середній рейтинг клієнта після додавання нового відгуку.

Бізнес-правило: Рейтинг клієнта повинен відображати середнє значення його оцінок у таблиці Feedback.

Опис: При додаванні нового запису у Feedback функція обчислює новий середній рейтинг клієнта та оновлює відповідне поле у таблиці Client.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION update_client_rating()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Client
    SET AverageRating = (
        SELECT AVG(Rating) FROM Feedback WHERE Client = NEW.Client
    )
    WHERE ID_ = NEW.Client;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №10:

Призначення: Даний індекс прискорює виконання запитів, які включають фільтрацію або сортування за стовпцями StartDate і Operator.

Бізнес-правило: Запити на аналіз дзвінків мають виконуватись швидко, враховуючи часові рамки (StartDate) та залучених операторів (Operator).

Опис: Цей індекс створений для оптимізації типових запитів, наприклад:

1. Пошук дзвінків, здійснених конкретним оператором у певний період.
2. Сортування дзвінків за датою початку та оператором.
3. Індекс зменшує кількість дискових операцій при вибірці даних, що значно підвищує продуктивність системи при роботі з великим обсягом інформації.

Результат:

Текст запиту:

```
CREATE INDEX Call_Idx
ON Call_(StartDate, Operator);
```

Висновок

У межах курсової роботи було розроблено базу даних для підтримки діяльності Call-центру. Основною метою проекту є забезпечити ефективну організацію та автоматизацію процесів обробки дзвінків, управління операторами та клієнтськими запитами, а також збору й аналізу даних для прийняття рішень.

Основні результати роботи:

1. Проектування бази даних:
    1. Створено логічну та фізичну моделі бази даних, що відповідають бізнес-вимогам предметного середовища.
    2. Розроблено 10 основних таблиць, які забезпечують зберігання та обробку даних про дзвінки, операторів, клієнтів, запити, робочі зміни та відгуки.
2. Реалізація функціоналу:
    1. Впроваджено 10 збережених процедур та функцій, що забезпечують автоматизацію рутинних операцій (наприклад, обчислення рейтингу клієнта, додавання пов’язаних записів тощо).
    2. Реалізовано 5 тригерів різного типу, які підтримують цілісність даних, автоматично обчислюють значення або здійснюють каскадні операції.
    3. Створено 3 представлення для зручного отримання агрегованих даних та спрощення аналізу інформації.
    4. Реалізовано 20 SQL-запитів для отримання аналітичної інформації, із яких 6 містять підзапити та працюють з кількома таблицями.
3. Управління доступом:
    1. Налаштовано ролі для різних категорій користувачів (менеджери, оператори, клієнти), що дозволяє чітко контролювати доступ до даних і забезпечувати безпеку.

Таким чином, розроблена база даних є потужним інструментом для ефективного управління діяльністю Call-центру, забезпечуючи автоматизацію процесів, підтримку бізнес-рішень та високий рівень безпеки.

Список використаної літератури

Call Center визначення та приклади на Wikipedia URL: https://en.wikipedia.org/wiki/Call_centre (дата звернення 22.12.2024)

PostgreSQL database management system home page. URL: https://www.postgresql.org/ (дата звернення 22.12.2024).

Zoho desk домашня сторінка. URL: https://www.zoho.com/desk/omni-channel-customer-service.html (дата звернення 22.12.2024).

Freshdesk домашня сторінка. URL: https://www.freshworks.com/freshcaller-cloud-pbx/features/ (дата звернення 22.12.2024).

Bitrix24 домашній сторінка. URL: https://www.bitrix24.com/uses/call-center-software.php (дата звернення 22.12.2024).

Zendesk домашній сторінка. URL: https://www.zendesk.com/ (дата звернення 22.12.2024).

Asterisk домашній сторінка. URL: https://www.asterisk.org/ (дата звернення 22.12.2024).

ДОДАТОК А ПРОГРАМНІ КОДИ ДЛЯ БАЗИ ДАНИХ

Програмні коди для бази даних

(Найменування програми (документа))

студента групи ІП-35 ІІ курсу

Адаменко А.Б.

Запит №1:

Призначення: Отримання переліку всіх операторів та кількості дзвінків, які вони обробили.

Бізнес-правило: Оператор повинен фіксувати всі оброблені дзвінки для аналізу продуктивності.

Опис: Цей запит об'єднує таблиці Operator та Call_, щоб визначити кількість дзвінків, оброблених кожним оператором.

Результат:

Текст запиту:

```
SELECT o.FIO AS OperatorName, COUNT(c.ID_) AS TotalCalls
FROM Operator o
LEFT JOIN Call_ c ON o.ID_ = c.Operator
GROUP BY o.FIO;
```

Запит №2:

Призначення: Пошук клієнтів, які залишили відгуки із низькими оцінками.

Бізнес-правило: Компанія повинна реагувати на негативні відгуки для покращення якості обслуговування.

Опис: Використовуються таблиці Client та Feedback, щоб знайти клієнтів із оцінками нижче 4.5.

Результат:

Текст запиту:

```
SELECT c.FIO AS ClientName, f.Rating AS FeedbackRating, o.FIO AS OperatorName,
d.Name_ AS DepartmentName
SELECT cl.FIO AS ClientName, fb.Rating, fb.Comment
FROM Client cl
INNER JOIN Feedback fb ON cl.ID_ = fb.Client
WHERE fb.Rating < 4.5;
```

Запит №3:

Призначення: Отримання списку операторів, які працювали у певну зміну.

Бізнес-правило: Керівництво має знати, хто працював у конкретний період.

Опис: Використовуються таблиці Operator та Shedule для фільтрації операторів за типом зміни.

Результат:

Текст запиту:

```
SELECT o.FIO AS OperatorName, s.ShiftType
FROM Operator o
INNER JOIN Shedule s ON o.ID_ = s.Operator
WHERE s.ShiftType = 'during_evening';
```

Запит №4:

Призначення: Пошук дзвінків, що були здійснені певним клієнтом у конкретний період.

Бізнес-правило: Інформація про дзвінки клієнта повинна бути доступна для аналізу його активності.

Опис: Таблиці Client та Call_ використовуються для пошуку дзвінків за клієнтом та датою.

Результат:

Текст запиту:

```
SELECT c.StartDate, c.Duration, op.FIO AS OperatorName
FROM Call_ c
INNER JOIN Client cl ON c.Client = cl.ID_
INNER JOIN Operator op ON c.Operator = op.ID_
WHERE cl.FIO = 'Іван Петренко' AND c.StartDate BETWEEN '2024-01-01' AND '2024-12-31';
```

Опис:

Результат виконання:

Запит №5:

Запит 5

Призначення: Визначення операторів, які отримували відгуки із середнім рейтингом вище 6.5.

Бізнес-правило: Компанія має заохочувати операторів із високими результатами обслуговування.

Опис: Використовується підзапит для обчислення середнього рейтингу по кожному оператору.

Результат:

Текст запиту:

```
SELECT o.FIO AS OperatorName, AVG(fb.Rating) AS AverageRating
FROM Operator o
INNER JOIN Call_ c ON o.ID_ = c.Operator
INNER JOIN Feedback fb ON fb.Call_ = c.ID_
GROUP BY o.FIO
HAVING AVG(fb.Rating) > 6.5;
```

Запит №6:

Призначення: Перегляд дзвінків, що мають відкриті запити на обробку.

Бізнес-правило: Всі запити повинні мати статус завершення або бути відкритими для обробки.

Опис: Об'єднуються таблиці Call_ та Request, фільтруються запити без дати завершення.

Результат:

Текст запиту:

```
SELECT c.ID_ AS CallID, r.RequestDate, r.Notes
FROM Call_ c
INNER JOIN Request r ON c.ID_ = r.Call_
WHERE r.ResolutionDate IS NULL;
```

Запит №7:

Призначення: Пошук клієнтів, які здійснили дзвінки тривалістю більше 1 години.

Бізнес-правило: Довгі дзвінки можуть свідчити про складні проблеми клієнтів.

Опис: Таблиці Call_ та Client використовуються для аналізу дзвінків клієнтів.

Результат:

Текст запиту:

```
SELECT cl.FIO AS ClientName, c.StartDate, c.Duration
FROM Call_ c
INNER JOIN Client cl ON c.Client = cl.ID_
WHERE c.Duration > 2400;
```

Запит №8:

Призначення: Отримання списку клієнтів із кількістю відкритих запитів.

Бізнес-правило: Керівництво повинно знати, які клієнти мають невирішені проблеми.

Опис: Підзапит рахує кількість запитів для кожного клієнта.

Результат:

Текст запиту:

```
SELECT cl.FIO AS ClientName, (
  SELECT COUNT(*)
  FROM Request r
  WHERE r.Call_ IN (
    SELECT c.ID_
    FROM Call_ c
    WHERE c.Client = cl.ID_
  ) AND r.ResolutionDate IS NULL
) AS OpenRequests
FROM Client cl;
```

Запит №9:

Призначення: Перевірка графіку роботи операторів для уникнення конфліктів змін.

Бізнес-правило: Оператори не повинні працювати більше однієї зміни за день.

Опис: Використовується групування для перевірки кількості змін оператора на день.

Результат:

Текст запиту:

```
SELECT s.Operator, s.ShiftDate, COUNT(s.ShiftType) AS ShiftsPerDay
FROM Shedule s
GROUP BY s.Operator, s.ShiftDate
HAVING COUNT(s.ShiftType) > 1;
```

Запит №10:

Призначення: Отримання кількості дзвінків, які отримали оцінку вище 7, по кожному відділу.

Бізнес-правило: Відділи з високими оцінками мають відзначатися як приклад для інших.

Опис: Об'єднання таблиць Department, Operator, Call_ та Feedback для аналізу оцінок.

Результат:

Текст запиту:

```
SELECT d.Name_ AS DepartmentName, COUNT(fb.ID_) AS HighRatedCalls
FROM Department d
INNER JOIN Operator o ON d.ID_ = o.Department
INNER JOIN Call_ c ON o.ID_ = c.Operator
INNER JOIN Feedback fb ON c.ID_ = fb.Call_
WHERE fb.Rating > 7
GROUP BY d.Name_;
```

Запит №11:

Призначення: Отримання списку операторів, які обробляли дзвінки певного клієнта.

Бізнес-правило: Інформація про взаємодії клієнтів з операторами повинна бути доступною.

Опис: Таблиці Call_ та Operator використовуються для визначення операторів, які працювали з клієнтом.

Результат:

Текст запиту:

```
SELECT DISTINCT o.FIO AS OperatorName
FROM Call_ c
INNER JOIN Operator o ON c.Operator = o.ID_
WHERE c.Client = (SELECT ID_ FROM Client WHERE FIO = 'Іван Петренко');
```

Запит №12:

Призначення: Пошук клієнтів, які не залишали жодного відгуку.

Бізнес-правило: Компанія повинна стимулювати клієнтів залишати відгуки для аналізу якості обслуговування.

Опис: Використовується підзапит для перевірки відсутності записів у таблиці Feedback.

Результат:

Текст запиту:

```
SELECT cl.FIO AS ClientName
FROM Client cl
WHERE NOT EXISTS (
  SELECT 1
  FROM Feedback fb
  WHERE fb.Client = cl.ID_
);
```

Запит №13:

Призначення: Перегляд середньої тривалості дзвінків по кожному оператору.

Бізнес-правило: Керівництво має знати, які оператори витрачають більше часу на обслуговування.

Опис: Використовується агрегація для обчислення середньої тривалості дзвінків.

Результат:

Текст запиту:

```
SELECT o.FIO AS OperatorName, AVG(c.Duration) AS AverageDuration
FROM Operator o
INNER JOIN Call_ c ON o.ID_ = c.Operator
GROUP BY o.FIO;
```

Запит №14:

Призначення: Пошук клієнтів, які здійснили дзвінки із певного відділу.

Бізнес-правило: Взаємодії клієнтів із конкретними відділами повинні бути відстежені.

Опис: Об'єднання таблиць Client, Call_ та Operator для фільтрації за відділом.

Результат:

Текст запиту:

```
SELECT cl.FIO AS ClientName
FROM Client cl
INNER JOIN Call_ c ON cl.ID_ = c.Client
INNER JOIN Operator o ON c.Operator = o.ID_
WHERE o.Department = (SELECT ID_ FROM Department WHERE Name_ LIKE '%Cats%');
```

Запит №15:

Призначення: Визначення найбільш активного клієнта за кількістю дзвінків.

Бізнес-правило: Компанія має знати своїх ключових клієнтів для кращого обслуговування.

Опис: Групування таблиці Call_ для обчислення кількості дзвінків по кожному клієнту.

Результат:

Текст запиту:

```
SELECT cl.FIO AS ClientName, COUNT(c.ID_) AS TotalCalls
FROM Client cl
INNER JOIN Call_ c ON cl.ID_ = c.Client
GROUP BY cl.FIO
ORDER BY COUNT(c.ID_) DESC
LIMIT 1;
```

Запит №16:

Призначення: Аналіз частоти звернень клієнтів із певним типом дзвінків.

Бізнес-правило: Необхідно знати, які типи дзвінків найбільш популярні серед клієнтів.

Опис: Об'єднання таблиць Call_, CallType та Client для аналізу популярності типів дзвінків.

Результат:

Текст запиту:

```
SELECT ct.Type_, COUNT(c.ID_) AS TotalCalls
FROM CallType ct
INNER JOIN Call_ c ON ct.Call_ = c.ID_
INNER JOIN Client cl ON c.Client = cl.ID_
GROUP BY ct.Type_
```

```
ORDER BY COUNT(c.ID_) DESC;
```

Запит №17:

Призначення: Підрахунок середнього рейтингу дзвінків для кожного відділу.

Бізнес-правило: Відділи повинні забезпечувати якісний сервіс на основі зворотного зв'язку клієнтів.

Словесний опис: Цей запит обчислює середній рейтинг, який клієнти залишили для дзвінків операторів кожного відділу.

Результат:

Текст запиту:

```
SELECT
    d.Name_ AS DepartmentName,
    AVG(f.Rating) AS AverageRating
FROM
    Department d
JOIN
    Operator o ON d.ID_ = o.Department
JOIN
    Call_ c ON o.ID_ = c.Operator
JOIN
    Feedback f ON c.ID_ = f.Call_
GROUP BY
    d.Name_;
```

Запит №18:

Призначення: Пошук операторів, які мають досвід роботи в усіх можливих типах змін.

Бізнес-правило: Оператори повинні бути готові працювати в різних робочих змінах для забезпечення гнучкості графіка.

Словесний опис: Цей запит знаходить операторів, які працювали в кожному типі змін (ранкова, денна, вечірня, нічна).

Результат:

Текст запиту:

```
SELECT
```

```
    s.Operator
FROM
    Shedule s
GROUP BY
    s.Operator
HAVING
    COUNT(DISTINCT s.ShiftType) = (SELECT COUNT(DISTINCT ShiftType) FROM
Shedule);
```

Запит №19:

-- Призначення: Визначення найдовших дзвінків для кожного оператора.

Бізнес-правило: Оператори повинні підтримувати якісний сервіс незалежно від тривалості дзвінків.

Словесний опис: Цей запит показує максимальну тривалість дзвінка для кожного оператора.

Результат:

Текст запиту:

```
SELECT
    o.FIO AS OperatorName,
    MAX(c.Duration) AS LongestCall
FROM
    Operator o
JOIN
    Call_ c ON o.ID_ = c.Operator
GROUP BY
    o.FIO;
```

Представлення №1:

Призначення: Обчислення мінімального, середнього та максимального рейтингу дзвінків для кожного оператора з виключенням екстремальних оцінок.

Бізнес-правило: Ранжуються оцінки дзвінків оператора по зростанню та спаданні, та обчислюються статистики за виключенням найгірших і найкращих 20% оцінок.

Опис: Для обчислення використовуються лише оцінки середнього діапазону, виключаючи 20% найгірших та 20% найкращих для деякої кількісті

величин для запобіганню використання надто рідких даних для того, щоб вони були використані.

Результат:

Текст запиту:

```
CREATE VIEW OperatorRatingStatistic
WITH RankedFeedback AS (
    SELECT
        o.FIO AS OperatorFIO,
        f.Rating,
        ROW_NUMBER() OVER (PARTITION BY o.FIO ORDER BY f.Rating ASC) AS
RowAsc,
        ROW_NUMBER() OVER (PARTITION BY o.FIO ORDER BY f.Rating DESC) AS
RowDesc,
        COUNT(*) OVER (PARTITION BY o.FIO) AS TotalRows
    FROM Operator o
    INNER JOIN Call_ c ON o.ID_ = c.Operator
    INNER JOIN Feedback f ON c.ID_ = f.Call_
)
SELECT
    OperatorFIO,
    ROUND(MIN(CASE WHEN RowAsc > TotalRows * 0.1 AND RowDesc > TotalRows * 0.2
THEN Rating END), 1) AS MinimalCallRating,
    ROUND(AVG(CASE WHEN RowAsc > TotalRows * 0.1 AND RowDesc > TotalRows * 0.2
THEN Rating END), 1) AS AverageCallRating,
    ROUND(MAX(CASE WHEN RowAsc > TotalRows * 0.1 AND RowDesc > TotalRows * 0.2
THEN Rating END), 1) AS MaximumCallRating
FROM RankedFeedback
GROUP BY OperatorFIO
ORDER BY AverageCallRating DESC, OperatorFIO;
```

Представлення №2:

Призначення: Представлення історії дзвінків для аналізу якості обслуговування.

Бізнес-правило: Кожен дзвінок має бути зареєстрований, можливий аналіз якості обслуговування.

Опис: Показує список дзвінків клієнтів із деталями оператора та зворотного зв'язку (якщо є).

Результат:

Текст запиту:

```
CREATE VIEW ClientCallHistory AS
SELECT
    c.FIO AS ClientName,
    o.FIO AS OperatorName,
    call.StartDate,
    call.Duration,
    fb.Rating,
    fb.Comment
FROM
    Call_ call
    JOIN Client c ON call.Client = c.ID_
    JOIN Operator o ON call.Operator = o.ID_
    LEFT JOIN Feedback fb ON call.ID_ = fb.Call_;
```

Представлення №3:

Призначення: Відображення графіка роботи операторів.

Бізнес-правило: Кожен оператор повинен мати заздалегідь визначений графік роботи.

Опис: Відображає інформацію про графік роботи кожного оператора.

Результат:

Текст запиту:

```
CREATE VIEW OperatorSchedules AS
SELECT
    o.FIO AS OperatorName,
    s.ShiftDate,
    s.ShiftType
FROM
    Shedule s
    JOIN Operator o ON s.Operator = o.ID_;
```

Тригер №1:

Призначення: Забезпечення цілісності даних у таблиці `Feedback` шляхом перевірки рейтингу.

Бізнес-правило: Рейтинг повинен бути в межах від 0 до 10 включно, щоб уникнути некоректних даних.

Опис: Тригер перевіряє значення рейтингу перед вставкою або оновленням запису в таблиці `Feedback`. Якщо значення не відповідає правилам, операція скасовується.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION validate_feedback_rating() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.Rating < 0 OR NEW.Rating > 10 THEN
        RAISE EXCEPTION 'Rating must be between 0 and 10.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_feedback_rating
BEFORE INSERT OR UPDATE ON Feedback
FOR EACH ROW
EXECUTE FUNCTION validate_feedback_rating();
```

Тригер №2:

Призначення: Автоматичне створення запису в логах під час змін у таблиці `Incident`.

Бізнес-правило: Всі зміни в інцидентах повинні бути задокументовані для подальшого аналізу.

Опис: Після оновлення запису в таблиці `Incident`, тригер додає інформацію про старі значення до таблиці `IncidentLog`.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION log_incident_changes() RETURNS TRIGGER AS $$
```

```
BEGIN
    INSERT INTO IncidentLog (Operator, Date_, Description_, IsResolved,
ChangeTime)
    VALUES (OLD.Operator, OLD.Date_, OLD.Description_, OLD.IsResolved, now());
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER track_incident_changes
AFTER UPDATE ON Incident
FOR EACH ROW
EXECUTE FUNCTION log_incident_changes();
```

Тригер №3:

Призначення: Запобігання створенню перекриття змін для операторів у таблиці `Shedule`.

Бізнес-правило: Оператор не може працювати у кількох змінах одночасно в один день.

Опис: Перед вставкою або оновленням запису в таблиці `Shedule`, тригер перевіряє наявність перекриття змін. Якщо таке перекриття існує, операція скасовується.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION prevent_shift_overlap() RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1 FROM Shedule
        WHERE Operator = NEW.Operator
          AND ShiftDate = NEW.ShiftDate
          AND ShiftType = NEW.ShiftType
    ) THEN
        RAISE EXCEPTION 'Shift overlap detected for Operator ID %.',
NEW.Operator;
    END IF;
    RETURN NEW;
```

```
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER check_shift_overlap
BEFORE INSERT OR UPDATE ON Shedule
FOR EACH ROW
EXECUTE FUNCTION prevent_shift_overlap();
```

Тригер №4:

Призначення: Автоматично коригувати рейтинг у відгуках клієнтів, щоб значення залишалося в межах від 0 до 10.

Бізнес-правило: Рейтинг клієнта повинен бути валідним і завжди перебувати в межах дозволеного діапазону (0–10).

Опис: Цей тригер використовується для перевірки даних перед їх додаванням у таблицю feedback або під час оновлення. Якщо клієнт вводить значення рейтингу, що виходить за межі дозволеного діапазону (менше 0 або більше 10), тригер автоматично коригує його до найближчого граничного значення (0 або 10 відповідно).

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION enforce_feedback_rating()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.Rating < 0 THEN
        NEW.Rating := 0;
    ELSIF NEW.Rating > 10 THEN
        NEW.Rating := 10;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER enforce_feedback_rating_trigger
BEFORE INSERT OR UPDATE ON feedback
FOR EACH ROW
EXECUTE FUNCTION enforce_feedback_rating();
```

Тригер №5:

Призначення: Логувати зміни даних операторів у спеціальну таблицю, щоб забезпечити контроль за оновленнями.

Бізнес-правило: Всі зміни в таблиці operator повинні бути зареєстровані, щоб у разі виникнення проблем можна було відстежити, хто і коли їх вніс.

Опис: Триггер записує зміни в полі FIO (повне ім'я оператора) до таблиці operator_log. У цій таблиці фіксується ідентифікатор оператора, старе і нове значення імені, а також дата внесення змін. Це дозволяє зберігати історію оновлень даних для аудиту.

Результат:

Текст запиту:

```
CREATE TABLE OperatorLog(
    LogID       serial PRIMARY KEY,
    OperatorID  int NOT NULL,
    OldFIO      varchar(48),
    NewFIO      varchar(48),
    ChangeDate  timestamp DEFAULT CURRENT_TIMESTAMP
);

CREATE TRIGGER log_operator_changes_trigger
AFTER UPDATE OF FIO ON operator
FOR EACH ROW
EXECUTE FUNCTION log_operator_changes();
```

Функція №1:

Призначення: Забезпечити автоматичне позначення інциденту як вирішеного, якщо у полі Description_ вказано слово "resolved".

Бізнес-правило: Якщо інцидент позначається як вирішений, це повинно відображатись у базі даних.

Опис: Функція перевіряє зміст поля Description_ на наявність ключового слова "resolved" та встановлює значення IsResolved = TRUE.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION set_incident_resolved()
RETURNS TRIGGER AS $$
BEGIN
    IF POSITION('resolved' IN NEW.Description_) > 0 THEN
        NEW.IsResolved := TRUE;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №2:

Призначення: Запобігти внесенню змін у запис після його остаточного затвердження (Notes містить слово "finalized").

Бізнес-правило: Остаточно затверджені запити не можуть бути змінені.

Опис: Перед оновленням перевіряє вміст поля Notes. Якщо запис має статус "finalized", оновлення забороняється.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION restrict_request_updates()
RETURNS TRIGGER AS $$
BEGIN
    IF POSITION('finalized' IN OLD.Notes) > 0 THEN
        RAISE EXCEPTION 'Request updates are restricted after finalization';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №3:

Призначення: Гарантувати, що рейтинг знаходиться у межах від 0 до 10.

Бізнес-правило: Відгуки мають бути валідними за числовою шкалою.

Опис: Функція автоматично виправляє значення рейтингу до допустимих меж.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION enforce_feedback_rating()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.Rating < 0 THEN
        NEW.Rating := 0;
    ELSIF NEW.Rating > 10 THEN
        NEW.Rating := 10;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №4:

Призначення: Логувати зміни імен операторів для забезпечення прозорості операцій.

Бізнес-правило: Ім'я оператора є критичною інформацією, всі зміни мають бути відстежені.

Опис: Функція створює запис у таблиці operator_log із зазначенням старого та нового імені після кожної зміни.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION log_operator_changes()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO operator_log (OperatorID, OldFIO, NewFIO)
    VALUES (OLD.ID_, OLD.FIO, NEW.FIO);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №5:

Призначення: Гарантувати, що кожен дзвінок має асоційований тип.

Бізнес-правило: Жоден дзвінок не може залишитись без типу у системі.

Опис: При створенні нового дзвінка функція автоматично додає запис у таблицю CallType із зазначенням типу за замовчуванням "general".

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION auto_assign_calltype()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO CallType (Call_, Type_, Description_)
    VALUES (NEW.ID_, 'general', 'Automatically assigned call type');
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №6:

Призначення: Гарантувати, що кожен запит має пов’язаний результат.

Бізнес-правило: Жоден запит не може залишитися без відповідного запису результату.

Опис: При створенні нового запису в таблиці Request автоматично додається запис у таблицю RequestResult із початковими значеннями.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION auto_create_request_result()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO RequestResult (Request, ResolutionDate, Result)
    VALUES (NEW.ID_, NULL, 'Pending');
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №7:

Призначення: Автоматично обчислювати тривалість дзвінка після його завершення.

Бізнес-правило: Тривалість дзвінка має бути правильно зафіксована для аналізу роботи операторів.

Опис: При оновленні StartDate функція обчислює та оновлює значення Duration.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION calculate_call_duration()
RETURNS TRIGGER AS $$
BEGIN
    NEW.Duration := EXTRACT(EPOCH FROM (NOW() - NEW.StartDate)) / 60.0; -- в
хвилинах
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №8:

Призначення: Заборонити додавання клієнтів із однаковими телефоном та електронною адресою.

Бізнес-правило: У базі даних не повинно бути дублюючих записів клієнтів.

Опис: Функція перевіряє новий запис на унікальність телефону та email. Якщо знайдено дублюючі дані, операція скасовується.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION prevent_duplicate_clients()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
```

```
        SELECT 1 FROM Client
        WHERE Phone = NEW.Phone OR Email = NEW.Email
    ) THEN
        RAISE EXCEPTION 'Duplicate client entry is not allowed';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №9:

Призначення: Забезпечити каскадне видалення типів дзвінків при видаленні запису дзвінка.

Бізнес-правило: Типи дзвінків не повинні залишатися у базі без відповідних записів у таблиці Call_.

Опис: Після видалення дзвінка автоматично видаляються записи у таблиці CallType.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION cascade_delete_calltype()
RETURNS TRIGGER AS $$
BEGIN
    DELETE FROM CallType WHERE Call_ = OLD.ID_;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

Функція №10:

Призначення: Перераховувати середній рейтинг клієнта після додавання нового відгуку.

Бізнес-правило: Рейтинг клієнта повинен відображати середнє значення його оцінок у таблиці Feedback.

Опис: При додаванні нового запису у Feedback функція обчислює новий середній рейтинг клієнта та оновлює відповідне поле у таблиці Client.

Результат:

Текст запиту:

```
CREATE OR REPLACE FUNCTION update_client_rating()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Client
    SET AverageRating = (
        SELECT AVG(Rating) FROM Feedback WHERE Client = NEW.Client
```

```
    )
    WHERE ID_ = NEW.Client;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Функція №10:

Призначення: Даний індекс прискорює виконання запитів, які включають фільтрацію або сортування за стовпцями StartDate і Operator.

Бізнес-правило: Запити на аналіз дзвінків мають виконуватись швидко, враховуючи часові рамки (StartDate) та залучених операторів (Operator).

Опис: Цей індекс створений для оптимізації типових запитів, наприклад:

1. Пошук дзвінків, здійснених конкретним оператором у певний період.
2. Сортування дзвінків за датою початку та оператором.
3. Індекс зменшує кількість дискових операцій при вибірці даних, що значно підвищує продуктивність системи при роботі з великим обсягом інформації.

Результат:

Текст запиту:

```
CREATE INDEX Call_Idx
ON Call_(StartDate, Operator);
```

ДОДАТОК Б ДАНІ ДЛЯ ІМПОРТУВАННЯ ДО БАЗИ ДАНИХ

_____Дані для імпортування до бази даних_____

(Найменування програми (документа))

_____GIT_____

(тип носія)

____https://github.com/adamenko-arsen/kpi/blob/master/kr_db/Import/*.csv___

студента групи ІП-35 ІІ курсу

Адаменко А.Б.