

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 з підтримки волонтерської діяльності

Студента (ки) 2 курсу ІП-21 групи спеціальності 121 «Інженерія програмного забезпечення»

Бойка Бориса Дмитровича

(прізвище та ініціали)

Керівник к.т.н., доцент, Ліщук Катерина Ігорівна

(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 ІІІ-21 Семестр 3

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

Бойку Борису Дмитровичу

(прізвище, ім'я, по батькові)

1. Тема роботи База даних з підтримки волонтерської діяльності

керівник роботи к.т.н., доцент, Ліщук Катерина Ігорівна

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

2. Строк подання студентом роботи 26.01.2024

3. Вихідні дані до роботи

завдання на розробку бази даних для підтримки волонтерської діяльності

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 08.11.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання курсового проекту	Строк виконання етапів проекту	Примітка
1	Аналіз предметного середовища	10.01.2024	
2	Побудова ER-моделі	11.01.2024	
3	Побудова реляційної схеми з ER-моделі	16.01.2024	
4	Створення бази даних, у форматі обраної системи управління базою даних	17.01.2024	
5	Створення користувачів бази даних	23.01.2024	
6	Імпорт даних з використанням засобів СУБД в створену базу даних	23.01.2024	
7	Створення мовою SQL запитів	23.01.2024	
8	Оптимізація роботи запитів	25.01.2024	
9	Оформлення пояснювальної записки	26.01.2024	
10	Захист курсової роботи	27.01.2024	

Студент

_____ Бойко Б.Д.
(підпис) (прізвище та ініціали)

Керівник роботи

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

ЗМІСТ

ЗМІСТ	4
ВСТУП.....	5
1 ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА.....	6
1.1 Розбір об'єкту дослідження	6
1.2 Вхідні і вихідні дані.....	7
1.3 Основні бізнес-процеси використання даних	8
2 АНАЛІЗ СТОРОННІХ ПРОГРАМНИХ ПРОДУКТІВ	9
2.1 Сервіс VolunteerHub.....	9
2.2 Сервіс VicNet	10
2.3 Сервіс GivePulse.....	11
3 ПОСТАНОВКА ЗАДАЧІ	12
4 ОПИС КОНЦЕПТУАЛЬНОЇ МОДЕЛІ.....	13
4.1 Опис інформаційних об'єктів	13
4.2 Опис відношень	16
4.3 Діаграма відношень	19
5 ПОБУДОВА БАЗИ ДАНИХ	20
5.1 Обґрунтування вибору системи управління базами даних	20
5.2 Опис сутностей в табличному вигляді.....	21
5.3 SQL-скрипт	30
5.4 Згенерована схема бази даних	36
6 РОБОТА З БАЗОЮ ДАНИХ.....	37
6.1 Тексти генераторів.....	37
6.2 Тексти функцій/процедур	38
6.3 Тексти тригерів.....	69
6.4 Тексти представлень	79
6.5 Тексти запитів.....	83
6.6 Оптимізація.....	84
6.7 Додавання користувачів.....	85
ВИСНОВКИ	90
ДОДАТОК.....	91
СПИСОК ДЖЕРЕЛ.....	92

ВСТУП

Волонтерство дозволяє людям відчутися своєю соціальною відповідальністю та зробити внесок у світ навколо себе. Це сприяє формуванню активної громадянської позиції та надає можливість отримати нові навички та досвід у різних сферах, що може бути корисним у професійній кар'єрі. [1]

З лютого 2022 року в Україні розпочалася нова хвиля волонтерської активності. Фонд "Демократичні ініціативи" імені І. Кучеріва провів опитування кількох сотень респондентів. За його результатами з'ясувалося, що близько 60% — приблизно 12 млн — жителів України брали участь у волонтерських справах чи благодійності, а 11% планують долучитися [2].

Мета даної курсової роботи полягає у створенні інформаційної складової інструменту для координації людських зусиль із забезпеченням регулярного підбиття підсумків роботи та оцінки ефективності, разом з тим забезпечуючи спрощення заходів комплексного юридичного адміністрування та фінансової звітності. Утілення встановленої мети залежить від розв'язання переліку задач:

1. Перенесення понять реального життя у реляційну модель.
 2. Підбір збалансованої структури зв'язків між відношеннями.
 3. Проектування важелів маніпуляції над даними поряд з другорядними механізмами масштабування й персоналізації.
 4. Ізоляції приватних даних системи від неавторизованих користувачів.
- Розробка має стати в пригоді як окремим фізичним особам, так і благодійним організаціям, що потребують оптимізації керування ресурсами.[2]

При виборі програмного забезпечення особлива увага приділялася некомерційним засадам справи. Система управління з відкритим кодом позбавляє видавця та користувачів від додаткових витрат на ліцензію та гарантує високий рівень інтегрованості з іншими продуктами.

1 ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА

1.1 Розбір об'єкту дослідження

Згідно із закону України № 3236-VI «Про волонтерську діяльність», волонтерська діяльність — добровільна, соціально спрямована, неприбуткова діяльність, що здійснюється волонтерами шляхом надання безоплатної допомоги. [3].

Одиночні волонтери за різних обставин гуртуються у відмінні за ладом і розміром групи. Мотивують на такий крок амбітні плани, які складно чи неможливо виконати наодинці. За різними показниками можна виділити такі об'єднання:

- Громадська ініціатива

Необмежене за кількістю учасників, але зазвичай середньочисленне об'єднання, що згуртувалося навколо конкретної ідеї або завдання, може проводити діяльність від лиця громадської організації. Неформальне або володіє обмеженими ознаками формальності, відповідно не має чіткої ієрархії. Представляє інтереси учасників або групи осіб на яких направлена допомога. Ініціативні зібрання можуть бути одноразовими чи систематично, але здебільшого нетривалі.[4]

- Група

Для таких об'єднань притаманний попередній відбір серед загальної маси активних осіб. У підсумку, волонтери заключають договори з організацією, що ініціювала створення групи, на взаємовигідних умовах. Усередині групи формуються команди, кожна з яких опікується окремою задачею чи її компонентом. Таким чином корисна робота в декількох напрямках може здійснюватися одночасно і довготривало. Представники організації стежать за прогресом та відповідають за навчання й забезпечення добровольців протягом всього терміну співробітництва.[5]

- **Фонд**

Такий тип об'єднання повністю самостійний. Юридичний статус наділяє його легальним правом проводити публічні збори коштів. Члени об'єднання ухвалюють рішення про вкладення грошей у вигляді проектів, що є реакцією на сигнали замовників. На ім'я фонду укладаються контракти з комерційними постачальниками, видаються дозволи на ввезення матеріальної допомоги з-за кордону тощо. Про всі витрати фонд звітує перед фіскальною службою та донатерами. [6]

1.2 Вхідні і вихідні дані

Вхідні дані:

- **Активності:** сукупність домовленостей про часові рамки, ідейне спрямування, доступні бонуси, адресу або облікові дані електронного майданчика, в залежності від узгодженого формату.
- **Персональні характеристики:** інформація, що стосується не лише корисних якостей, а й деяких обмежень добровольців, якщо такі є.
- **Звернення:** ключові пункти з розлогих заявок, із зазначенням ініціалів контактної особи, обсягів необхідної допомоги та ін.
- **Контракти:** важливі деталі договорів про постачання продукції, що увійде в склад допомоги.

Вихідні дані:

- **Моніторинг:** порівняння актуальних потреб аудиторії зі спроможностями команди; оцінка успішності замовлень від обраних постачальників.
- **Журналювання:** реєстрація осіб чи компаній, що профінансували волонтерську діяльність
- **Дослідження та корисні матеріали:** Історії отримувачів, аналітичні статті про вплив на спільноту.

1.3 Основні бізнес-процеси використання даних

Опрацювання вхідних та проміжних даних супроводжується низкою бізнес-процесів:

- ***Авторизація користувачів:*** додавання нових облікових записів до системи з охороною довірчих даних.
- ***Присвоєння задач:*** відбір виконавців з числа активних волонтерів з урахуванням володіння критичними компетенціями. Завдання без критичних вимог доступні для вибору за бажанням.
- ***Нормування праці:*** серія перевірок для попередження перенавантажень, що сприяють злагодженій роботі. Зважаючи на брак законодавчих правил для волонтерів, за зразок буде взятий чинний трудовий кодекс.
- ***Декларування надходжень та витрат:*** скринінг історії нарахувань за кварталний період або в обраних межах. Переспрямування залишкових накопичень.
- ***Оцінка результативності:*** відстеження позитивних і негативних тенденцій від партнерства зі третіми особами.

Висновок: у розділі викладено належні для розуміння області інтересу міркування. Наведені підходи до оперування внутрішніми даними у подальшому послугують орієнтиром процедурного дизайну.

2 АНАЛІЗ СТОРОННІХ ПРОГРАМНИХ ПРОДУКТІВ

2.1 Сервіс VolunteerHub

VolunteerHub — сервіс направлений на комплексну оптимізацію управління волонтерами і розрахований на великі організації, які прагнуть підтримати репутацію і покращити досвід волонтерства.

Основний функціонал:

- *Спільнота та рекрутинг*: проста реєстрація приваблює увагу широкого кола осіб до сервісу, де вони мають змогу відгукуватися на пропозиції за інтересами. Таким чином компанії скорочують маркетингові витрати.
- *Планування*: вбудований календар, окрім дат початку і закінчення, зберігає короткий опис події. Будь-яку подію можна зробити повторюваною.
- *Координація та розподіл обов'язків*: люди, які виявили бажання брати участь у події автоматично отримують регулярні нагадування та слова подяки. Організатори особисто вирішують чи власноруч делегувати обов'язки чи надати учасникам вільний вибір за умови дотримання вимог вакансії.
- *Збори пожертв*: платформа підтримує розміщення благодійних зборів незалежно від суми та просуває їх згідно спрямування. Усі, благодійники отримують квитанцію з вказанням призначення донату.
- *Аналітика та звітність*: широкий вибір шаблонів, що містять статистику продуктивності та вподобань волонтерів, дані про швидкість накопичення коштів, списки постійних учасників тощо. За бажанням можна видозмінити звіти та зберегти користувацькі шаблони.

Переваги:

- Підбірка індивідуальної статистики про кожного учасника.
- Інтеграція більшості провідних платіжних систем.
- Зв'язок з хмарним сховищем.
- Профілі з посиланням на соціальні мережі учасників.
- Підтвердження статусу волонтера під час реєстрації.
- Різноманіття навчальних матеріалів для новачків.
- Необмежена кількість спеціалізованих груп.

Недоліки:

- Ризики збоїв через надмірне навантаження.
- Відсутність мобільного офлайн застосунку.
- Посередні стандарти безпеки. [7]

2.2 Сервіс VicNet

VicNet – допоміжний сервіс орієнтований на невеликі команди, що бажають швидко пристосуватися до особливості колективного волонтерства, маючи надійну підтримку з боку адміністраторів.

Основний функціонал:

- **Поширення й заохочення:** платформа оперує сторінками-візитівками волонтерських об'єднань, які легко розповсюдити в мережі. Кожен, кого зацікавила інформація на сторінці може скористатися заготовленими формами і приєднатися до команди.
- **Спостереження та визнання:** всередині своїх профілів волонтери вибирають робочі години, а система зі свого боку контролює перекриття всіх запланованих змін. За блискуче виконання своїх обов'язків можуть отримувати нагороди. Відзначення волонтерів автоматизується за умовленими критеріями.

- **Комунікація:** учасники групи не мають доступу до текстових спілкування інших груп. Присутність волонтера на локації фіксується надходженням збережених сповіщень.

- **Безпека:** компанія-партнер Verified First пропонує послугу перевірки біографії, аби мати впевненість в порядності добровольців.

Переваги:

- Оснащення протоколами SSL шифрування.
- Мобільний застосунок.
- Необмежена кількість користувацьких типів вступних форм.
- Закрита екосистема повідомлень.
- Ізольоване сховище в кожному профілі під файли різних форматів.

Недоліки:

- Брак вбудованого благодійного рахунку.
- Перенасичення вузькоспрямованими додатками.
- Слабкий аналітичний потенціал. [8]

2.3 Сервіс GivePulse

GivePulse — сучасний сервіс з багатим функціоналом, за загальними рисами ідентичний до VolunteerHub, при цьому вирізняється унікальними для сектору peer-to-peer донатами/подарунками. [9]

Висновок: У цьому розділі згадано функціонал трьох програмних продуктів, їхні сильні і слабкі сторони. На ґрунті цього дослідження будуть сформовані вимоги до поточної розробки.

3 ПОСТАНОВКА ЗАДАЧІ

Мета: втілення доступної бази даних, що посприє налагодженню координації волонтерського сегменту. Процес побудови такої конструкції залучає злиття множини шаблонів дій, а отже потребує зваженої декомпозиції. Задачі з якими доведеться стикнутися безпосередньо пов'язані з бізнес-логікою, котра зі свого боку обумовлена наступними вимогами:

1. ***Розумне архівування:*** збереження неактуальних даних протягом встановленого терміну з розрахунку на економію місця та попередження випадкової втрати.
2. ***Засвідчення відповідальності:*** маркування факту ознайомлення з обов'язками.
3. ***Сегрегація профілів:*** передбачення програмного обмеження на численні підключення до єдиного акаунту.
4. ***Адаптивність:*** простір можливостей має бути модульним, масштабованим і розкриватися у міру потреби користувача.
5. ***Контроль версій:*** Кожна маніпуляція над завданням мусить бути занотована.

Завдання:

1. Транслювати концепції в сутності.
2. Специфікувати рольову модель
3. Реалізувати важелі управління через функції та процедури
4. Підготувати набір затребуваних запитів

Висновок: у розділі остаточно сформульовано мету та завдання проекту. Поставлені задачі враховують необхідність ефективного управління даними. Сформульовані вимоги вказують на очікувані параметри рішення, яке вважатиметься успішним, тому вони є фундаментом для моделі відношень.

4 ОПИС КОНЦЕПТУАЛЬНОЇ МОДЕЛІ

4.1 Опис інформаційних об'єктів

1. Отримувач:

- Назва групи
- Ім'я відповідальної особи
- Прізвище відповідальної особи
- Номер телефону
- Електронна пошта

2. Запит

- Дата подання
- Час подання
- Деталі

3. Активність

- Мета
- Опис
- Організатор
- Дата початку
- Дата закінчення

4. Рахунок

- Поточна сума
- Цільова сума

5. Надходження

- Назва банківської системи
- Дата
- Час
- Кількість грошей
- Коментар

6. Предмет

- Назва
- Ціна
- Кількість

7. Контракт

- Заголовок
- Дата заключення
- Дата виконання
- Відгук

8. Постачальник

- Назва
- Міжнародний рахунок
- Країна

10. Локація

- Назва
- Вулиця
- Номер будівлі
- Місто
- Країна

11. Категорія

- Назва

12. Статус

- Назва

13. Завдання

- Опис
- Тривалість
- Пріоритетність

14. Важлива компетенція

- Назва
- Рівень володіння

15. Журнал задач

- Дата запису
- Час запису
- Статус

16. Волонтер

- Ім'я
- Прізвище
- Дата народження
- Стать
- Номер телефона
- Електронна пошта
- Пароль
- Дата останнього входу
- Дата реєстрації
- Маркер активності

4.2 Опис відношень

1. Отримувач – подає – Запит:

Кожен отримувач може мати один або багато запитів на допомогу. Кожен запит на допомогу належить лише одному отримувачу.

2. Активність – задовольняє – Запит:

Активність може не залежати від запитів або поривати один і більше. Кожен запит належить конкретній активності.

3. Рахунок – виділений для – Активність:

Активність може мати один або не мати рахунку. Рахунок може належати одній активності або бути незалежним (головний рахунок).

4. Предмет – придбаний на – Рахунок:

На кошти одного рахунка може бути придбано багато предметів або не придбано нічого. Кожний предмет придбаний за кошти конкретного рахунка.

5. Запит – включає – Предмет:

Кожний запит може включати багато або жодного предмету. Предмет можуть бути придбані для якогось запиту або незалежно від нього.

6. Предмет – придбаний за - Контракт:

Предмет може бути придбаний лише за конкретним контрактом або не придбані взагалі. За одним контрактом може бути придбано від одного до багатьох предметів.

7. Предмет – належить – Категорія:

Кожен предмет належить до однієї категорії.

8. Надходження – на – Рахунок:

Кожен рахунок може бути без або мати багато надходжень. Надходження нараховуються на конкретний рахунок.

9. Контракт – укладається з – Постачальник:

Кожен постачальник мусить мати хоча б один контракт. Кожен контракт укладається з одним постачальником.

10. Контракт – має – Статус:

Кожен контракт має один поточний статус. Статус може належати багатьом контрактам або бути не задіяним.

11. Активність – проводиться – Локація:

Активність може відбуватися онлайн або на декількох локаціях одночасно. Кожна локація призначена для конкретної активності (події).

12. Завдання – виконується в рамках – Активність:

Кожне завдання належить відбуватися в рамках визначеної активності. Активність зі свого боку може мати щонайменше одне завдання.

13. Завдання – має – Статус:

Кожна завдання має один поточний статус. Статус може належати багатьом завданням або бути не задіяним.

14. Запис в журналі завдань – має – Статус:

Кожен запис має один поточний статус. Статус може належати багатьом записам або бути не задіяним.

15. Волонтер – виконує – Завдання:

Кожен волонтер може бути вільним чи виконувати декілька завдань. Завдання може бути не призначене до виконання або виконуватися кількома волонтерами одночасно.

16. Важлива компетенція – необхідна для – Завдання:

Завдання може вимагати володіння декількома компетенціями або обходитись без них. Компетенція може ставати в нагоді в одному чи багатьох завданнях.

17. Волонтер – володіє – Важлива компетенція:

Волонтер може володіти одразу багатьма компетенціями або не знати жодної. Кожна компетенція належить одній особі чи цілій групі.

18. Запис в журналі завдань – реєструє – Завдання:

Кожна зміна стану завдання реєструється. Один запис посилається на одне завдання.

19. Запис в журналі завдань – відмічає – Волонтер:

Волонтер може мати щонайменше одну відмітку в журналі. Кожен запис призначений для одного волонтера.

4.3 Діаграма відношень

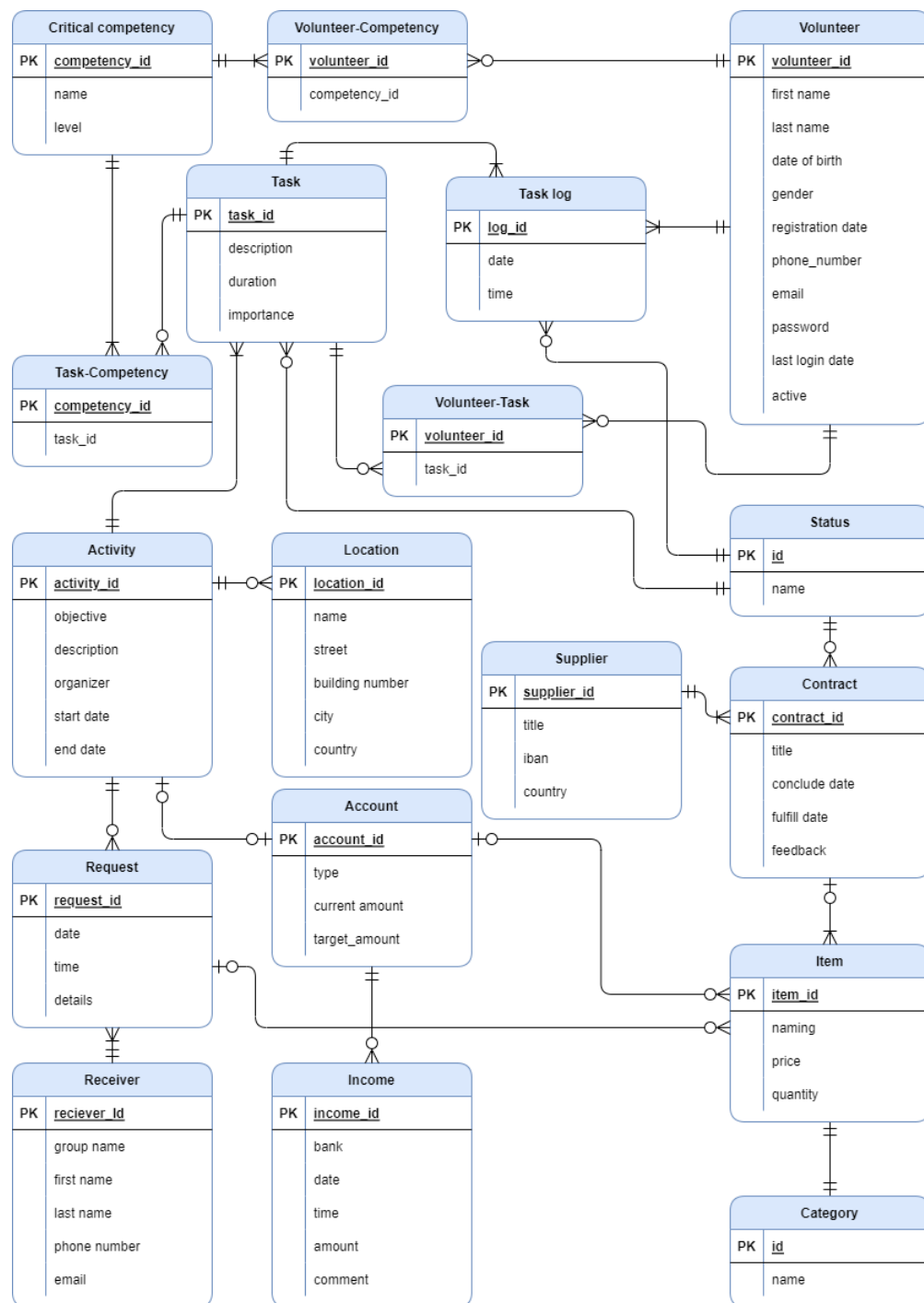


Рис 4.1 ER діаграма бази даних з підтримки волонтерської діяльності

Висновок: У цьому розділі представлено інформаційні об'єкти, які входять до складу концептуальної моделі системи. Наведені відношення між об'єктами надають уявлення про причини взаємодії й залежності між ними. За допомогою отриманої діаграми база буде відтворена в обраній СУБД.

5 ПОБУДОВА БАЗИ ДАНИХ

5.1 Обґрунтування вибору системи управління базами даних

Серед наявних технічних засобів реляційна СУБД PostgreSQL найкраще пасувала до тематичного середовища завдяки ряду особливостей:

- **Відповідність принципам ACID:** система для організації роботи людей потребує довірного рівня цілісності даних. Чітке дотримання стандартів ACID (атомарність, послідовність, ізоляція та довговічність) спростовує сумніви щодо імовірного псування інформації під час стандартних операцій.
- **Відкритість коду:** як вже було зазначено зрозумілий вихідний код суттєво зменшує витрати, а окрім цього сприяє створенню спільноти розробників, які зацікавлені у вдосконаленні та оновленнях.
- **Структурна гнучкість:** бази соціального спрямування покладаються на різноманітні джерела неоднорідних даних. Багата номенклатура вбудованих типів, включаючи JSON, та здатність утримувати поля користувацьких типів прискорюють перехід від абстрактного уявлення до фізичного представлення.
- **Глибина і масштабованість:** така база даних очікувано кратно зростатиме зі збільшенням кількості учасників. Закладена в PostgreSQL вертикальна і горизонтальна масштабованість обіцяє стабільну експлуатацію протягом тривалого часу.
- **Багатокористувацька взаємодія:** провідні асинхронні механізми уможливають безперебійну без шкоди для узгодженості даних.
- **Шифрування:** SSL-сертифікати і детальний контроль доступу, гарантують безпечне зберігання і передачу даних волонтерів, захищаючи їх від несанкціонованого доступу.
- **Сумісність:** некомерційні продукти часто потребують інтеграції зі сторонніми інструментами чи платформами, пріоритезуючи відсутність штучних перешкод, тим самим підвищуючи універсальність.[10]

5.2 Опис сутностей в табличному вигляді

Таблиця «Отримувач» призначена для зберігання інформації про одиночних осіб або групи, яким надано допомогу:

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор отримувача
group_name	VARCHAR	255		Означення групової приналежності отримувача
first_name	VARCHAR	255		Ім'я відповідальної особи
last_name	VARCHAR	255		Прізвище відповідальної особи
phone_number	VARCHAR	30		Номер телефону
email	VARCHAR	255		Адреса електронної пошти

Рис 5.1 Таблиця «Отримувач»

Таблиця «Активність» використовується для зберігання історії про різні проекти чи події.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор активності
objective	VARCHAR	255		Мета проекту або події
description	TEXT			Короткий опис події
organizer	VARCHAR	255		Організатор події
start_date	DATE			Дата початку
end_date	DATE			Дата закінчення

Рис 5.2 Таблиця «Активність»

Таблиця «Запит» існує для обліку звернень від користувачів. Після розгляду, на запит може бути відкрито активність.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор запиту
receiver_id	INT			Користувача, що створив запит
activity_id	INT			Події протягом якої надається допомога
date	DATE			Дата подання
time	TIME			Час подання
details	TEXT			Уточнення або коментарі до запиту

Рис 5.3 Таблиця «Запит»

Таблиця «Локація» запам'ятовує місця проведення різних активностей чи подій.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор місця
activity_id	INT		FK	Активності, що проходить на цій локація
name	VARCHAR	255		Назва приміщення, території
street	VARCHAR	255		Назва вулиці
building_number	SMALLINT			Номер будівлі
city	VARCHAR	255		Назва міста
country	VARCHAR	255		Назва країни

Рис 5.4 Таблиця «Локація»

Таблиця «Рахунок» призначена для зберігання інформації про рахунки, пов'язані зі збором коштів для різних активностей. Ця таблиця дозволяє відстежувати фінансові успіхи зборів коштів і зупиняти їх при досягненні цільової суми.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор рахунку
activity_id	INT		FK	Активність під час якої проводиться збір коштів
type	VARCHAR	15		Тип рахунку
current_amount	DECIMAL	(20,8)		Поточна сума на рахунку
target_amount	DECIMAL	(20,8)		Цільова сума для рахунку

Рис 5.5 Таблиця «Рахунок»

Таблиця «Надходження» декларує кожну пожертву коштів на рахунок, визначаючи банк, дату, час, суму та коментар.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор надходження на рахунок
account_id	INT		FK	Рахунок на який поступає надходження
bank	VARCHAR	255		Назва банківської системи
date	DATE			Дата отримання доходу
time	TIME			Час отримання доходу
amount	DECIMAL	(18,8)		Сума доходу

comment	VARCHAR	255		Коментар відправника
---------	---------	-----	--	----------------------

Рис 5.6 Таблиця «Надходження»

Таблиця «Статус» містить інформацію про різні статуси та їхні унікальні ідентифікатори, що утворюють множину можливих станів інших відношень.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор
name	VARCHAR	100		Назва статусу

Рис 5.7 Таблиця «Статус»

Таблиця «Контракт» відведена під зберігання згадок про контракти, що були укладені з постачальниками. Завдяки цьому відношенню уповноважені зможуть прогнозувати об'єми й строки поставок.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор
supplier_id	INT		FK	Зовнішній ключ, посилається на supplier(id)
status_id	INT		FK	Зовнішній ключ, посилається на status(id)
title	VARCHAR	255		Назва контракту
conclude_date	DATE			Дата укладання контракту
fulfill_date	DATE			Дата виконання контракту
feedback	SMALLINT			Оцінка або зворотний зв'язок

Рис 5.8 Таблиця «Контракт»

Таблиця «Категорія» класифікує товари або послуги. Ця таблиця може використовуватися для управління каталогами в межах аналітичних процедур.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор
name	VARCHAR	255		Назва категорії

Рис 5.9 Таблиця «Категорія»

Таблиця «Постачальник» дозволяє системі акумулювати зведення про різних постачальників, що може бути використано при ухваленні рішень про майбутню роботи з ними.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор постачальника
title	VARCHAR	255		Назва організації постачальника
iban	VARCHAR	34		Міжнародний номер банківського рахунку
country	VARCHAR	255		Назва країни, з якої постачальник

Рис 5.10 Таблиця «Постачальник»

Таблиця «Предмет» призначена для зберігання інформації про обіг товарів/послуг.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор товару або послуги

account_id	INT		FK	Рахунок на кошти якого заповується товар
request_id	INT		FK	Запит за яким заповується товар
contract_id	INT		FK	Контракт на закупівлю
naming	VARCHAR	255		Назва товару чи послуги
price	DECIMAL	(18,8)		Ціна за одиницю
estimated price	DECIMAL	(18,8)		
quantity	INT			Кількість одиниць т/п
category_id	INT		FK	Категорія до якої належить товар чи послуга

Рис 5.11 Таблиця «Предмет»

Таблиця «Волонтер» відтворює профілі волонтерів і дає змогу обмінювати сигналами з колегами.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		РК	Ідентифікатор волонтера
first_name	VARCHAR	255		Ім'я
last_name	VARCHAR	255		Прізвище
date_of_birth	DATE			Дата народження
gender	CHAR	1		Стать волонтера
phone_number	VARCHAR	30		Номер телефону
email	VARCHAR	255		Поштова адреса
registration_date	TIMESTAMP			Дата реєстрації
password	VARCHAR	255		Пароль
last_login_date	TIMESTAMP			Дата останнього входу в аккаунт

active	BOOLEAN			Прапорець, що вказує чи готовий волонтер до роботи
--------	---------	--	--	--

Рис 5.12 Таблиця «Волонтер»

Таблиця «Важлива компетенція» містить інформацію про різні затребувані вміння волонтерів. Ця таблиця дозволяє оцінювати спроможності команди учасників та обирати найкращу стратегію роботи.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор вміння
name	VARCHAR	255		Назва вміння
level	VARCHAR	100		Рівень володіння вмінням

Рис 5.13 Таблиця «Важлива компетенція»

Таблиця «Завдання» містить інформацію про різні завдання (плани, тривалість, важливість та поточний статус) фіксуючи поточних виконавців.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор завдання
activity_id	INT		FK	Активність для якої призначене завдання
description	TEXT			План завдання
duration	SMALLINT			Орієнтовний час виконання завдання
importance	SMALLINT			Рівень важливості завдання

status_id	INT		FK	Статус у якому перебуває завдання
-----------	-----	--	----	-----------------------------------

Рис 5.14 Таблиця «Завдання»

Таблиця «Журнал завдань» призначена для дослідження історії завдань. Ця таблиця надає можливість фіксувати кожен змін характеристик із вказанням точного часу.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	INT		PK	Унікальний ідентифікатор запису в журналі
task_id	INT		FK	Завдання вказане в записі
volunteer_id	INT		FK	Волонтер, що має стосунок до завдання
date	DATE			Дата занесення запису
time	TIME			Час занесення запису
status_id	INT		FK	Статус завдання на момент запису

Рис 5.15 Таблиця «Журнал завдань»

Таблиця "Завдання-Компетенція" відображення кореляцію між завданнями та компетенціями (знаннями чи навичками) в контексті волонтерської діяльності.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
competency_id	INT		PK, FK	Певне важливе знання чи навичка
task_id	INT		PK, FK	Завдання яке потребує цього вміння

Рис 5.16 Таблиця «Завдання-Компетенція»

Таблиця "Волонтер-Компетенція" дарує можливість відстежувати вартісні вміння членів команди.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
volunteer_id	INT		РК, FK	Волонтер, який володіє особливістю
competency_id	INT		РК, FK	Знання чи навичка, що належить волонтеру

Рис 5.17 Таблиця «Волонтер-Компетенція»

Таблиця "Волонтер-Завдання" призначена для відображення взаємозв'язку між волонтерами та завданнями в контексті обраної активності.

Ім'я поля	Тип даних	Розмір	Ключ	Опис
volunteer_id	INT		РК, FK	Волонтер, який має стосунок до завдання
task_id	INT		РК, FK	Завдання призначене для виконання

Рис 5.18 Таблиця «Волонтер-Завдання»

5.3 SQL-ckpunm

-- Receiver block

```
CREATE TABLE receiver (
  id SERIAL PRIMARY KEY,
  group_name VARCHAR(255),
  first_name VARCHAR(255) NOT NULL,
  last_name VARCHAR(255) NOT NULL,
  phone_number VARCHAR(30) UNIQUE NOT NULL,
  email VARCHAR(255) UNIQUE
);
```

-- Activity block

```
CREATE TABLE activity(
  id SERIAL PRIMARY KEY,
  objective VARCHAR(255) NOT NULL,
  description TEXT,
  organizer VARCHAR(255) NOT NULL,
  start_date DATE NOT NULL,
  end_date DATE
);
```

```
CREATE TABLE request (
  id SERIAL PRIMARY KEY,
  receiver_id INT REFERENCES receiver(id) NOT NULL,
  activity_id INT REFERENCES activity(id),
  date DATE NOT NULL,
  time TIME NOT NULL,
  details TEXT
);
```

```

CREATE TABLE location (
id SERIAL PRIMARY KEY,
activity_id INT REFERENCES activity(id) NOT NULL,
name VARCHAR(255) NOT NULL,
street VARCHAR(255) NOT NULL,
building_number SMALLINT,
city VARCHAR(255) NOT NULL,
country VARCHAR(255)
);

--Account block

CREATE TABLE account (
id SERIAL PRIMARY KEY,
activity_id INT REFERENCES activity(id) UNIQUE,
type VARCHAR(15) NOT NULL,
current_amount DECIMAL(20,8) DEFAULT 0 CHECK(current_amount >= 0),,
target_amount DECIMAL(20,8) DEFAULT 0 CHECK(target_amount >= 0),
reserved_amount DEFAULT 0 CHECK(reserved_amount >= 0)
);

CREATE TABLE income (
id SERIAL PRIMARY KEY,
account_id INT REFERENCES account(id) NOT NULL,
bank VARCHAR(255) NOT NULL,
date DATE NOT NULL,
time TIME NOT NULL,
amount DECIMAL(18,8) NOT NULL CHECK (amount > = 0),
comment VARCHAR(255)
);

```

--Contract block

```
CREATE TABLE status (  
  id SERIAL PRIMARY KEY,  
  name VARCHAR(100) UNIQUE  
);  
  
CREATE TABLE supplier (  
  id SERIAL PRIMARY KEY,  
  title VARCHAR(255) NOT NULL,  
  iban VARCHAR(34) NOT NULL UNIQUE,  
  country VARCHAR(255)  
);  
  
CREATE TABLE contract (  
  id SERIAL PRIMARY KEY,  
  supplier_id INT REFERENCES supplier(id) NOT NULL,  
  status_id INT REFERENCES status(id) NOT NULL,  
  title VARCHAR(255) NOT NULL,  
  conclude_date DATE NOT NULL,  
  fulfill_date DATE NOT NULL,  
  feedback SMALLINT  
);
```

--Item block

```
CREATE TABLE category (  
  id SERIAL PRIMARY KEY,  
  name VARCHAR(255) NOT NULL UNIQUE  
);
```



```

CREATE TABLE item (
id SERIAL PRIMARY KEY,
account_id INT REFERENCES account(id),
request_id INT REFERENCES request(id),
contract_id INT REFERENCES contract(id),
naming VARCHAR(255) NOT NULL,
price DECIMAL(18,8) CHECK (price >= 0),
estimated_price DECIMAL(18,8) NOT NULL CHECK(estimated_price >=0),
quantity INT NOT NULL CHECK(quantity>0),
category_id INT REFERENCES category(id)
);
--Volunteer block
CREATE TABLE volunteer (
id SERIAL PRIMARY KEY,
first_name VARCHAR(255) NOT NULL,
last_name VARCHAR(255) NOT NULL,
date_of_birth DATE NOT NULL,
gender CHAR(1),
phone_number VARCHAR(30) UNIQUE NOT NULL,
email VARCHAR(255) UNIQUE,
registration_date TIMESTAMP DEFAULT NOW(),
password VARCHAR(255) NOT NULL,
last_login_date TIMESTAMP DEFAULT NOW(),
active BOOLEAN DEFAULT true
);

```

--Competency block

```
CREATE TABLE critical_competency (
id SERIAL PRIMARY KEY,
name VARCHAR(255) NOT NULL,
level VARCHAR(100) NOT NULL,
CONSTRAINT comp_level_identity UNIQUE (name,level)
);
```

--Task block

```
CREATE TABLE task (
id SERIAL PRIMARY KEY,
activity_id INT REFERENCES activity(id) NOT NULL,
description TEXT NOT NULL,
duration SMALLINT NOT NULL,
importance SMALLINT,
status_id INT REFERENCES status(id) NOT NULL
);
```

```
CREATE TABLE task_log (
id SERIAL PRIMARY KEY,
task_id INT REFERENCES task(id) NOT NULL,
volunteer_id INT REFERENCES volunteer(id) NOT NULL,
date DATE NOT NULL,
time TIME NOT NULL,
status_id INT REFERENCES status(id) NOT NULL
);
```

--Many-to-many tables

```
CREATE TABLE task_competency (  
competency_id INT REFERENCES critical_competency(id),  
task_id INT REFERENCES task(id),  
CONSTRAINT task_competency_dual_pkey PRIMARY KEY(competency_id,  
task_id)  
);  
  
CREATE TABLE volunteer_competency (  
volunteer_id INT REFERENCES volunteer(id),  
competency_id INT REFERENCES critical_competency(id),  
CONSTRAINT volunteer_competency_dual_pkey PRIMARY KEY(volunteer_id,  
competency_id)  
);  
  
CREATE TABLE volunteer_task (  
volunteer_id INT REFERENCES volunteer(id),  
task_id INT REFERENCES task(id),  
CONSTRAINT volunteer_task_dual_pkey PRIMARY KEY(volunteer_id, task_id)  
);
```

5.4 Згенерована схема бази даних

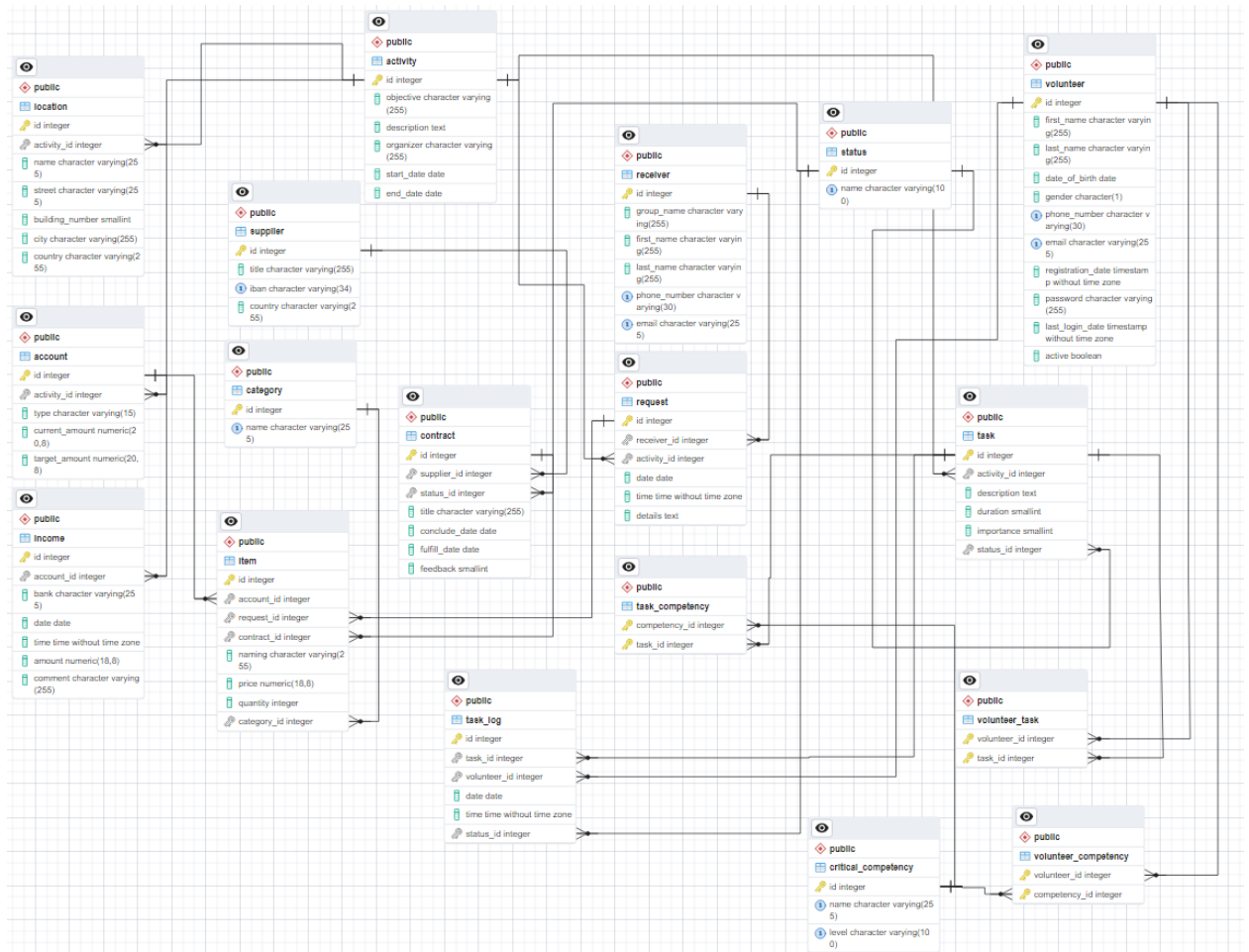


Рис 5.1 Схема бази згенерована утилітою pgadmin4

Висновок: у матеріалах цього розділу приділено час поясненню мотивації до вибору СУБД PostgreSQL. Продемонстровано скрипт написання фізичної версії бази даних. Додатково до текстової документації прикладено таблиці до кожного сутнісного утворення. У підсумку отримано візуальну проекцію даної керувальної системи.

6 РОБОТА З БАЗОЮ ДАНИХ

6.1 Тексти генераторів

CREATE TABLE receiver (

id SERIAL PRIMARY KEY

CREATE TABLE activity(

id SERIAL PRIMARY KEY

CREATE TABLE request (

id SERIAL PRIMARY KEY

CREATE TABLE location (

id SERIAL PRIMARY KEY

CREATE TABLE account (

id SERIAL PRIMARY KEY

CREATE TABLE income (

id SERIAL PRIMARY KEY

CREATE TABLE status (

id SERIAL PRIMARY KEY

CREATE TABLE supplier (

id SERIAL PRIMARY KEY

CREATE TABLE contract (

id SERIAL PRIMARY KEY

CREATE TABLE category (

id SERIAL PRIMARY KEY

```
CREATE TABLE task_log (
```

```
id SERIAL PRIMARY KEY
```

```
CREATE TABLE item (
```

```
id SERIAL PRIMARY KEY
```

```
CREATE TABLE volunteer (
```

```
id SERIAL PRIMARY KEY
```

```
CREATE TABLE critical_competency (
```

```
id SERIAL PRIMARY KEY
```

```
CREATE TABLE task (
```

```
id SERIAL PRIMARY KEY
```

```
CREATE TABLE task_log (
```

```
id SERIAL PRIMARY KEY
```

6.2 Тексти функцій/процедур

Функція №1

- *Призначення:* отримання інформації про години зайнятості волонтера.
- *Бізнес-правило:* запобігання перенавантаженню волонтерів
- *Словесний опис:* функція повертає кількість годин, яку волонтер витратив на завдання, відрізняючи волонтера за його ідентифікатором. Функція враховує існування волонтера. Якщо волонтер не має завдань, повертається 0.
- *Скрипт:*

```
CREATE OR REPLACE FUNCTION get_hours(volunteer_id INT) RETURNS INT
```

```
AS $body$
```

```

DECLARE

    total_hours INT;

BEGIN

    IF (SELECT id FROM volunteer

        WHERE volunteer.id = target_volunteer) IS NULL THEN

        RAISE EXCEPTION 'Volunteer with id:% does not exist', target_volunteer;

    END IF;

    SELECT COALESCE(SUM(t.duration), 0)

    INTO total_hours

    FROM volunteer v

    LEFT JOIN volunteer_task vt ON v.id = vt.volunteer_id

    LEFT JOIN task t ON vt.task_id = t.id

    WHERE v.id = target_volunteer;

    RETURN total_hours;

END

$body$

LANGUAGE plpgsql;

```

- *Тест роботи:*
SELECT get_hours(6);

get_hours	integer
	7

Рис. 6.2.1 Функція №1

Функція №2

- *Призначення:* перевірка завдання на наявність особливих вимог для його виконання
- *Бізнес-правило:* розумний розподіл завдань, що враховує персональні якості виконавця
- *Словесний опис:* у запиті обраховується кількість спеціальних навичок, пов'язаних із завданням. Якщо кількість навичок більше 0, функція повертає true, інакше – false
- *Скрипт:*

```
CREATE OR REPLACE FUNCTION need_special_skills(task_id INT)
RETURNS BOOLEAN AS
$body$
DECLARE
    skill_count INT;
BEGIN
    SELECT COUNT(*) INTO skill_count
    FROM task_competency tc
    WHERE tc.task_id = $1;
    RETURN CASE
        WHEN skill_count > 0 THEN true
        ELSE false
    END;
END
$body$
LANGUAGE plpgsql;
```


- *Тест роботи:*

```
SELECT task_id FROM task_competency WHERE competency_id = 4;
```

task_id	integer
	7

Рис. 6.2.2 Функція №2

```
SELECT need_special_skills(7);
```

need_special_skills	boolean
	true

Рис. 6.2.3 Функція №2

Функція №3

- *Призначення:* створює текстову квитанцію для благодійника
- *Бізнес-процес:* публічне визнання волонтерської діяльності
- *Словесний опис:* ця функція корисна для отримання звіту про пожертвування. У звіт-квитанцію входять банківські дані про переказ та подяка донору за підтримку групи волонтерів.

- Скрипт:

```
CREATE OR REPLACE FUNCTION generate_receipt(income_id INT, org_name TEXT)
```

```
RETURNS TEXT AS
```

```
$body$
```

```
DECLARE
```

```
    receipt_text TEXT;
```

```
    donor_name VARCHAR(255);
```

```
    amount DECIMAL(18,8);
```

```
BEGIN
```

```
    SELECT i.amount, COALESCE(i.comment, "")
```

```
    INTO amount, donor_name
```

```

FROM income i
WHERE i.id = $1;

receipt_text := 'Receipt for Donation' || E'\n\n';
receipt_text := receipt_text || org_name || E'\n\n';
receipt_text := receipt_text || 'Dear contributor' || ',' || E'\n\n';
receipt_text := receipt_text || 'We sincerely thank you for your generous
donation of $' || amount || '.' || E'\n';
receipt_text := receipt_text || 'Your contribution plays a vital role in supporting
our initiatives.' || E'\n\n';
receipt_text := receipt_text || 'Transaction Details:' || E'\n';
receipt_text := receipt_text || 'Date: ' || (SELECT date FROM income WHERE id
= $1) || E'\n';
receipt_text := receipt_text || 'Time: ' || (SELECT time FROM income WHERE
id = $1) || E'\n';
receipt_text := receipt_text || 'Amount: $' || amount || E'\n';
receipt_text := receipt_text || 'Comment: ' || COALESCE((SELECT comment
FROM income WHERE id = $1), '') || E'\n\n';
receipt_text := receipt_text || 'Thank you for your continued support!' || E'\n';
receipt_text := receipt_text || org_name;

RETURN receipt_text;
END
$body$
LANGUAGE plpgsql;

```

- *Текст повідому:*

```

SELECT id FROM income
LIMIT 3;

```

id	
[PK] integer	
	7
	8
	10

Рис. 6.2.4 Функція №3

```
SELECT generate_receipt(8);
```

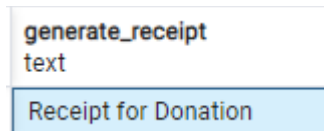


Рис. 6.2.5 Функція №3

Функція №4

- *Призначення:* відсіювання неактуальних активностей
- *Бізнес-процес:* економія часу на перегляд актуальних заходів
- *Словесний опис:* функція оцінює дату завершення роботи відносно поточної, стан пов'язаних з нею завдань і статус завершення контрактів, пов'язаних з елементами, щоб визначити, чи завершено роботу.

- *Скрипт:*

```
CREATE OR REPLACE FUNCTION is_activity_finished(target_activity_id INT)
RETURNS BOOLEAN AS
$body$
DECLARE
    activity_end_date activity.end_date%TYPE;
    number_of_tasks INT;
    associated_account INT;
    item_contract_check REFCURSOR;
    activity_associated_item item%ROWTYPE;
BEGIN
```

```
SELECT end_date INTO activity_end_date
FROM activity a WHERE a.id = $1;
```

```
IF activity_end_date IS NULL OR activity_end_date > CURRENT_DATE
THEN
```

```
    RETURN false;
END IF;
```

```
SELECT COUNT(*) INTO number_of_tasks
FROM task t
WHERE t.activity_id = $1;
```

```
IF (SELECT COUNT(*) FROM task
    WHERE task.activity_id = $1
    AND task.status_id = (SELECT s.id FROM status s WHERE name =
'Finished'))
```

```
    <> number_of_tasks THEN
    RETURN false;
END IF;
```

```
SELECT account.id INTO associated_account
FROM account
WHERE account.activity_id = $1;
```

```
OPEN item_contract_check FOR
SELECT * FROM item i WHERE i.account_id = associated_account;
```

```
LOOP
```

```
    FETCH item_contract_check INTO activity_associated_item;
```

```

EXIT WHEN NOT FOUND;

IF activity_associated_item.contract_id IS NULL OR
  (SELECT c.status_id FROM contract c
   WHERE c.id = activity_associated_item.contract_id)
  <> (SELECT s.id FROM status s WHERE name = 'Finished') THEN
  CLOSE item_contract_check;

  RETURN false;

END IF;

END LOOP;

CLOSE item_contract_check;

RETURN true;

END
$body$
LANGUAGE plpgsql;

```

- *Тест роботи:*

```
SELECT is_activity_finished(10);
```

is_activity_finished	🔒
boolean	
false	

Рис. 6.2.6 Функція №4

Функція №5

- *Призначення:* перевірка чи присвоєне певне завдання до певного волонтера
- *Бізнес-процес:* відіграє допоміжну роль у механізмі призначень.
- *Словесний опис:* функція звертається до таблиці призначень та шукає запис із вказаною сигнатурою. Результатом цього пошуку є значення «істина» або «хиба».

- *Скрипт:*

```
CREATE OR REPLACE FUNCTION is_volunteer_assigned(examined_volunteer
INT, questionable_tast INT)
RETURNS BOOLEAN AS
$body$
BEGIN
RETURN EXISTS(SELECT 1 FROM volunteer_task vt
                WHERE vt.volunteer_id = $1 AND
                vt.task_id = $2);
END
$body$
LANGUAGE plpgsql;
```

- *Тест роботи*

```
SELECT is_volunteer_assigned(3,2);
```

is_volunteer_assigned	🔒
boolean	
true	

Рис. 6.2.7 Функція №5

Процедура №1

- *Призначення:* допоміжна процедура для закріплення волонтера за справою
- *Бізнес-процес:* процедура є компонентом одного з центральних процесів, а саме процесу багатофакторного підбору кандидатів на роль учасників активності
- *Словесний опис:* процедура перевіряє, чи не існує вже відповідного запису у таблиці `volunteer_task` для вказаної пари волонтера та завдання. Якщо запис не існує, то виконується вставка нового запису з вказаними значеннями волонтера та завдання.

- *Скритт:*

```
CREATE OR REPLACE PROCEDURE assign_for_task(IN task_id INT, IN volunteer_id
INT)
AS $body$
BEGIN
IF NOT EXISTS (
    SELECT 1
    FROM volunteer_task vt
    WHERE vt.volunteer_id = $2
    AND vt.task_id = $1
)
THEN
    INSERT INTO volunteer_task (volunteer_id, task_id)
    VALUES ($2, $1);
END IF;
END
$body$
LANGUAGE plpgsql;
```

- *Тест роботи:*

```
SELECT * FROM volunteer_task WHERE task_id = 6
```

volunteer_id [PK] integer	task_id [PK] integer
------------------------------	-------------------------

Рис. 6.3.1 Процедура №1

```
CALL assign_for_task(6,10);
```

volunteer_id [PK] integer	task_id [PK] integer
10	6

Рис. 6.3.2 Процедура №1

Процедура №2

- *Призначення:* ведуча процедура роздачі завдань волонтерам, виступає в ролі агрегатора.
- *Бізнес-процес:* прийняття зважених рішень щодо призначення тих чи інших кандидатів на виконання доручень.
- *Словесний опис:* процедура автоматизує призначення волонтерів до завдань з урахуванням навичок та обмеження годин. У якості параметрів приймає ідентифікатор завдання, список волонтерів та регламентований ліміт годин. Ретельно аналізує стан завдання та наявність необхідних навичок. Якщо потрібно, оцінює компетентність волонтерів та призначає їх, дотримуючись обмежень.

У разі успіху змінює стан завдання та виводить повідомлення. Коли щось порушує рекомендації, генерує помилку зі стислим тлумаченням походження.

- *Скрипт:*

```
CREATE OR REPLACE PROCEDURE assign_validated(IN task_id INT, IN
volunteer_ids INT[], IN hours_limit INT DEFAULT 8)
```

```
AS
```

```
$body$
```

```
DECLARE
```

```
    report TEXT DEFAULT '';
```

```
    task_state TEXT;
```

```
    examined_volunteer INT;
```



```

    cu_volunteer_competency REFCURSOR;
    volunteer_competency_pair RECORD;
BEGIN
    SELECT s.name FROM task t JOIN status s ON t.status_id = s.id
    WHERE t.id = $1 INTO task_state;

    IF task_state NOT IN ('Planned', 'Cancelled') THEN
        RAISE EXCEPTION 'Task already %', task_state;
    ELSE
        IF need_special_skills($1) = true THEN
            report := report || 'Assignment report' || E'\n';
            FOREACH examined_volunteer IN ARRAY $2
            LOOP
                OPEN cu_volunteer_competency FOR
                SELECT u.volunteer_id, tc.competency_id
                FROM task_competency tc
                CROSS JOIN unnest(volunteer_ids) AS u(volunteer_id)
                WHERE tc.task_id = $1 AND u.volunteer_id =
examined_volunteer;
            LOOP
                FETCH cu_volunteer_competency INTO
volunteer_competency_pair;
                IF volunteer_competency_pair IS NULL THEN
                    EXIT;
                END IF;

                IF EXISTS (

```

```

SELECT 1
FROM volunteer_competency vc
WHERE vc.volunteer_id =
volunteer_competency_pair.volunteer_id
AND vc.competency_id =
volunteer_competency_pair.competency_id
) AND
get_hours(volunteer_competency_pair.volunteer_id) < $3
THEN
CALL assign_for_task($1,
volunteer_competency_pair.volunteer_id);
ELSE
report := report || 'Volunteer
' || volunteer_competency_pair.volunteer_id
|| 'does not have
competency:' || volunteer_competency_pair.competency_id || E'\n';
END IF;
END LOOP;
CLOSE cu_volunteer_competency;
END LOOP;
ELSE
FOREACH examined_volunteer IN ARRAY $2
LOOP
CALL assign_for_task($1, examined_volunteer);
END LOOP;
report := report || 'Assignment does not need special skills';
END IF;
END IF;
IF (SELECT COUNT(*) FROM volunteer_task vt

```

```

WHERE vt.task_id = $1) = 0

OR

(SELECT COUNT(*) FROM volunteer_task vt
WHERE vt.task_id = $1) <> array_length($2, 1)
THEN

    RAISE EXCEPTION 'Assignment failed. See report: %',report;
ELSE UPDATE task SET status_id = 2 WHERE task.id = $1;
report := report || E'\n';
report := report || 'Task succesfully assigned';
RAISE NOTICE '%', report;
END IF;
END
$body$
LANGUAGE plpgsql;

```

•*Текст роботи:*

```
SELECT * FROM task_competency WHERE competency_id = 4
```

competency_id [PK] integer	task_id [PK] integer
4	7

Рис. 6.3.3 Процедура №2

```
SELECT * FROM task_competency WHERE competency_id = 4
```

volunteer_id [PK] integer	competency_id [PK] integer
21	4

Рис. 6.3.4 Процедура №2

```
CALL assign_validated(7, ARRAY[50]);
```

```

ERROR:  Assignment failed. See report: Assignment report
Volunteer 50does not have competency:4

```

Рис. 6.3.5 Процедура №2

```
CALL assign_validated(7, ARRAY[21]);
```

```
ЗАМЕЧАНИЕ: Assignment report
Task succesfully assigned
CALL
```

Рис. 6.3.6 Процедура №2

volunteer_id [PK] integer	task_id [PK] integer
21	7

Рис. 6.3.7 Процедура №2

Процедура №3

- *Призначення:* сигнал про згоду або незгоду на виконання завдання
- *Бізнес-процес:* розмежування відповідальності між особою, що призначила завдання та особою, що прийняла його
- *Словесний опис:* процедура призначена для зміни статусу вказаного завдання в базі даних. `Accept_task` встановлює статус "В процесі". За аналогією, процедура `cancel_task` встановлює статус «Скасоване», а `finish_task` – «Завершене».
- *Скрипт:*

```
CREATE OR REPLACE PROCEDURE accept_task(IN volunteer_id INT, IN task_id INT)
AS
$body$
BEGIN
    IF NOT is_volunteer_assigned($1, $2) THEN
        RAISE EXCEPTION 'Access denied! You are not associated with this task';
    END IF;
    IF (SELECT t.status_id FROM task t WHERE t.id = $2)
    IN (SELECT s.id FROM status s WHERE s.name = 'Assigned')
    THEN
        UPDATE task
```

```

        SET status_id = (SELECT id FROM status WHERE name = 'In progress')
        WHERE task.id = $2;

    ELSE

        RAISE EXCEPTION 'This task can not be accepted';

    END IF;

END

$body$

LANGUAGE plpgsql;

```

```

CREATE OR REPLACE PROCEDURE finish_task(IN volunteer_id INT, IN task_id INT)
AS
$body$
BEGIN
    IF NOT is_volunteer_assigned($1, $2) THEN
        RAISE EXCEPTION 'Access denied! You are not associated with this task';
    END IF;

    IF (SELECT t.status_id FROM task t WHERE t.id = $2)
    IN (SELECT s.id FROM status s WHERE s.name = 'In progress')
    THEN
        UPDATE task
        SET status_id = (SELECT id FROM status WHERE name = 'Finished')
        WHERE task.id = $2;

    ELSE
        RAISE EXCEPTION 'This task can not be finished';
    END IF;

END

```

\$body\$

LANGUAGE plpgsql;

CREATE OR REPLACE PROCEDURE cancel_task(IN volunteer_id INT, IN task_id INT)

AS

\$body\$

BEGIN

IF NOT is_volunteer_assigned(\$1, \$2) THEN

RAISE EXCEPTION 'Access denied! You are not associated with this task';

END IF;

IF (SELECT t.status_id FROM task t WHERE t.id = \$2)

IN (SELECT s.id FROM status s WHERE s.name = 'Assigned' OR s.name = 'In progress')

THEN

UPDATE task

SET status_id = (SELECT id FROM status WHERE name = 'Cancelled')

WHERE id = \$2;

ELSE

RAISE EXCEPTION 'This task can not be cancelled';

END IF;

END

\$body\$

LANGUAGE plpgsql;

- *Течт роботу:*

SELECT * FROM task WHERE id = 7

id [PK] integer	activity_id integer	description text	duration smallint	importance smallint	status_id integer
7	3	Review project updates and compile reports	1	3	2

Рис. 6.3.8 Процедура №3

CALL accept_task(21,7);

id [PK] integer	activity_id integer	description text	duration smallint	importance smallint	status_id integer
7	3	Review project updates and compile reports	1	3	3

Рис. 6.3.9 Процедура №3

CALL accept_task(2,7);

ERROR: Access denied! You are not associated with this task

ERROR: This task can not be accepted

Рис. 6.3.10 Процедура №3

CALL finish_task(21,7);

id [PK] integer	activity_id integer	description text	duration smallint	importance smallint	status_id integer
7	3	Review project updates and compile reports	1	3	4

Рис. 6.3.11 Процедура №3

CALL assign_validated(8, ARRAY[7]);

ЗАМЕЧАНИЕ: Assignment does not need special skills
Task succesfully assigned
CALL

Рис. 6.3.12 Процедура №3

SELECT * FROM volunteer_task WHERE task_id = 8

volunteer_id [PK] integer	task_id [PK] integer
7	8

Рис. 6.3.13 Процедура №3

CALL cancel_task(7,8);

SELECT * FROM volunteer_task WHERE task_id = 8;

volunteer_id [PK] integer	task_id [PK] integer
------------------------------	-------------------------

Рис. 6.3.14 Процедура №3

Процедура №4

- *Призначення:* ініціалізація інфраструктури для оперування запитами.
- *Бізнес-процес:* прийняття рішення стосовно прийняття запиту в роботу.
- *Словесний опис:* ця процедура допомагає закріпити ствердне рішення щодо кількох запитів, одночасно створюючи корельовану активності і за потреби рахунок.

- *Скрипт:*

```
CREATE OR REPLACE PROCEDURE accept_requests(request_ids INT[],
objective VARCHAR(255), organizer VARCHAR(255), start_date DATE)
AS
$body$
DECLARE
    new_activity_id INT;
    targ_request_id INT;
    new_account_id INT;
BEGIN
    FOR targ_request_id IN SELECT unnest(request_ids)
    LOOP
        IF EXISTS (SELECT 1 FROM request r WHERE r.id = targ_request_id
AND r.activity_id IS NOT NULL) THEN
            RAISE EXCEPTION 'Request % already accepted.', targ_request_id;
        END IF;
    END LOOP;

    INSERT INTO activity (objective, organizer, start_date)
    VALUES ($2, $3, $4)
    RETURNING id INTO new_activity_id;

    UPDATE request SET activity_id = new_activity_id
    WHERE id = ANY(request_ids);
```



```

FOR targ_request_id IN SELECT unnest(request_ids)
LOOP
    IF EXISTS (SELECT 1 FROM item WHERE request_id =
targ_request_id) THEN
        IF NOT EXISTS (SELECT 1 FROM account WHERE activity_id =
new_activity_id) THEN
            INSERT INTO account (activity_id, type)
            VALUES (new_activity_id, 'Secondary')
            RETURNING id INTO new_account_id;
        END IF;
    END IF;
END LOOP;

IF new_account_id IS NOT NULL THEN
    UPDATE item SET account_id = new_account_id
    WHERE request_id = ANY(request_ids);

    UPDATE account SET target_amount =
        (SELECT SUM(it.estimated_price) FROM item it
        WHERE it.account_id = new_account_id)
    WHERE id = new_account_id;
END IF;
END
$body$
LANGUAGE plpgsql;

```

- *Течн роботу:*

```
SELECT * FROM request
```

id [PK] integer	receiver_id integer	activity_id integer	date date	time time without time zone	details text
1	1	2	2022-11-01	10:00:00	Some details
2	2	[null]	2023-10-01	10:00:00	Need help in game rendering
3	2	[null]	2023-10-01	11:00:00	Help me in game rendering. Please

Рис. 6.3.15 Процедура №4

CALL accept_requests(ARRAY[2,3], 'Help to the writer Alan Wake', 'University
ord % FCB', DATE'2023-10-02');

SELECT * FROM request;

id [PK] integer	receiver_id integer	activity_id integer	date date	time time without time zone	details text
2	2	5	2023-10-01	10:00:00	Need help in game rendering
3	2	5	2023-10-01	11:00:00	Help me in game rendering. Please

Рис. 6.3.16 Процедура №4

id [PK] integer	objective character varying (255)	description text	organizer character varying (255)	start_date date	end_date date
5	Help to the writer Alan Wake	[null]	University ord % FCB	2023-10-02	[null]

Рис. 6.3.17 Процедура №4

Процедура №5

- *Призначення:* додавання запису про укладення угоди, включно з супутніми деталями
- *Бізнес-процес:* контрагування необхідних товарів/послуг, які неможливо відтворити руками добровольців
- *Словесний опис:* процедура ефективно укладає контракти, адресує автоматичне додавання постачальника, статусу та нішеві фінансові розрахунки. Вона забезпечує цілісність даних завдяки систематичним перевіркам, включаючи перевірку існування постачальника та ретельну обробку помилок. Узагальнюючи, процедура спрощує процес формалізації контракту.

- *Скритум:*

```

CREATE OR REPLACE PROCEDURE conclude_contract(
    supplier_name VARCHAR(255),
    supplier_iban CHAR(34),
    title VARCHAR(255),
    items JSONB[],
    fulfill_date DATE,
    conclude_date DATE DEFAULT CURRENT_DATE
) AS
$body$
DECLARE
    new_supplier_id INT;
    contract_status INT;
    new_contract_id INT;
    price_per_unit DECIMAL(18,8);
    bought_item INT;
    connected_account INT;
    item_data JSON;
BEGIN
    SELECT id INTO new_supplier_id
    FROM supplier s
    WHERE s.title = $1 AND s.iban = $2;

    IF new_supplier_id IS NULL THEN
        INSERT INTO supplier(title, iban)
        VALUES($1, $2)
        RETURNING id INTO new_supplier_id;
    END IF;

```

```
SELECT status.id INTO contract_status FROM status WHERE status.name =
'Assigned';
```

```
INSERT INTO contract(supplier_id, status_id, title, conclude_date, fulfill_date)
VALUES(new_supplier_id, contract_status, $3, $6, $5)
RETURNING id INTO new_contract_id;
```

```
FOREACH item_data IN ARRAY items LOOP
```

```
    bought_item := (item_data->>'item_id')::INT;
```

```
    price_per_unit := (item_data->>'price')::DECIMAL(18,8);
```

```
SELECT account_id INTO connected_account FROM item WHERE id =
bought_item;
```

```
CASE
```

```
    WHEN (connected_account) IS NULL THEN
```

```
        RAISE EXCEPTION 'Item % does not attach to an account',
bought_item;
```

```
    WHEN (SELECT price FROM item WHERE id = bought_item) IS NOT
NULL THEN
```

```
        RAISE EXCEPTION 'Item % is already contracted', bought_item;
```

```
    WHEN (SELECT current_amount FROM account WHERE id =
connected_account) < price_per_unit THEN
```

```
        RAISE EXCEPTION 'Account % does not have enough money',
connected_account;
```

```
    ELSE
```

```
        UPDATE item
```

```
        SET contract_id = new_contract_id,
```

```
        price = price_per_unit
```

```

WHERE id = bought_item;

END CASE;

UPDATE account

SET current_amount = current_amount - price_per_unit,
    reserved_amount = reserved_amount + price_per_unit

WHERE id = connected_account;

END LOOP;

END

$body$

LANGUAGE plpgsql;

```

- *Тест роботи:*

```
SELECT * FROM item
```

id [PK] integer	account_id integer	request_id integer	contract_id integer	naming character varying (255)	price numeric (18,8)	quantity integer	category_id integer	estimated_price numeric (18,8)
3	1	1	[null]	Mavic 3T	[null]	1	4	290000.00000000

Рис. 6.3.18 Процедура №5

```
CALL conclude_contract('DJI', 'NL45RABO9510165298', 'Another_attemp to buy
Mavic', ARRAY[1], DATE'2024-01-28',285000)
```

```
ERROR: Account 1 does not have enough money
```

Рис. 6.3.19 Процедура №5

Поповнюємо рахунок.

```
CALL conclude_contract('DJI', 'NL45RABO9510165298', 'Another_attemp to buy
Mavic', ARRAY[3], DATE'2024-01-28',285000)
```

```
SELECT * FROM item
```

id [PK] integer	account_id integer	request_id integer	contract_id integer	naming character varying (255)	price numeric (18,8)	quantity integer	category_id integer	estimated_price numeric (18,8)
3	1	1	17	Mavic 3T	285000.00000000	1	4	290000.00000000

Рис. 6.3.20 Процедура №5

SELECT * FROM contract

id [PK] integer	supplier_id integer	status_id integer	title character varying (255)	conclude_date date	fulfill_date date	feedback smallint
1	1	5	Chinese UAV	2024-01-23	2024-01-27	[null]
17	1	2	Another_attempt to buy Mavic	2024-01-25	2024-01-28	[null]

Рис. 6.3.21 Процедура №5

SELECT * FROM supplier

id [PK] integer	title character varying (255)	iban character varying (34)	country character varying (255)
1	DJI	NL45RABO9510165298	[null]

Рис. 6.3.22 Процедура №5

SELECT * FROM account

id [PK] integer	activity_id integer	type character varying (15)	current_amount numeric (20,8)	target_amount numeric (20,8)	reserved_amount numeric (20,8)
2	[null]	Main	0.00000000	[null]	[null]
1	2	Secondary	36650.00000000	290000.00000000	285000.00000000

Рис. 6.3.23 Процедура №5

Процедура №6

- *Призначення:* скасування контракту з постачальником та повернення коштів на рахунок
- *Бізнес-процес:* управління документацією для здійснення прозорих фінансових операцій
- *Словесний опис:* процедура скасовує контракт, встановлюючи відповідний статус та повертає зарезервовані під цей контракт кошти
- *Скрипт:*

```
CREATE OR REPLACE PROCEDURE cancel_contract(
    contract_id INT
)
```

AS

\$body\$

DECLARE

 canceled_contract_status INT;

 finished_contract_status INT;

 canceled_price DECIMAL(18,8);

 connected_account INT;

 canceled_item INT;

BEGIN

 SELECT id INTO canceled_contract_status

 FROM status

 WHERE name = 'Cancelled';

 SELECT id INTO finished_contract_status

 FROM status

 WHERE name = 'Finished';

 IF (SELECT c.status_id FROM contract c

 WHERE c.id = \$1) = finished_contract_status THEN

 RAISE EXCEPTION 'This contract already succesfully finished and can not be cancelled!';

 ELSEIF(SELECT c.status_id FROM contract c

 WHERE c.id = \$1) = canceled_contract_status THEN

 RAISE EXCEPTION 'This contract already cancelled!';

 END IF;

 UPDATE contract

 SET status_id = canceled_contract_status

 WHERE id = \$1;

 FOR canceled_item IN (SELECT i.id FROM item i WHERE i.contract_id = \$1)

LOOP

```

SELECT price, account_id
INTO canceled_price, connected_account
FROM item
WHERE id = canceled_item;

UPDATE item
SET contract_id = NULL,
    price = NULL
WHERE id = canceled_item;

UPDATE account
SET current_amount = current_amount + canceled_price,
    reserved_amount = reserved_amount - canceled_price
WHERE id = connected_account;

END LOOP;

END

$body$

LANGUAGE plpgsql;

```

- Тест роботи:

```
SELECT * FROM contract
```

id [PK] integer	supplier_id integer	status_id integer	title character varying (255)	conclude_date date	fulfill_date date	feedback smallint
1	1	5	Chinese UAV	2024-01-23	2024-01-27	[null]
17	1	2	Another_attempt to buy Mavic	2024-01-25	2024-01-28	[null]

Рис. 6.3.23 Процедура №6

```
SELECT * FROM item
```

id [PK] integer	account_id integer	request_id integer	contract_id integer	naming character varying (255)	price numeric (18,8)	quantity integer	category_id integer	estimated_price numeric (18,8)
3	1	1	17	Mavic 3T	285000.00000000	1	4	290000.00000000

Рис. 6.3.24 Процедура №6

```
SELECT * FROM account;
```


id [PK] integer	activity_id integer	type character varying (15)	current_amount numeric (20,8)	target_amount numeric (20,8)	reserved_amount numeric (20,8)
2	[null]	Main	0.00000000	[null]	[null]
1	2	Secondary	36650.00000000	290000.00000000	285000.00000000

Рис. 6.3.25 Процедура №6

CALL cancel_contract(17);

SELECT * FROM contract;

id [PK] integer	supplier_id integer	status_id integer	title character varying (255)	conclude_date date	fulfill_date date	feedback smallint
1	1	5	Chinese UAV	2024-01-23	2024-01-27	[null]
17	1	5	Another_attempt to buy Mavic	2024-01-25	2024-01-28	[null]

Рис. 6.3.26 Процедура №6

SELECT * FROM item;

id [PK] integer	account_id integer	request_id integer	contract_id integer	naming character varying (255)	price numeric (18,8)	quantity integer	category_id integer	estimated_price numeric (18,8)
3	1	1	[null]	Mavic 3T	[null]	1	4	290000.00000000

Рис. 6.3.27 Процедура №6

SELECT * FROM account

id [PK] integer	activity_id integer	type character varying (15)	current_amount numeric (20,8)	target_amount numeric (20,8)	reserved_amount numeric (20,8)
2	[null]	Main	0.00000000	[null]	[null]
1	2	Secondary	321650.00000000	290000.00000000	0.00000000

Рис. 6.3.28 Процедура №6

Процедура №7

Призначення: процедура призначена для деактивації рахунків та переадресації фінансових залишків

Бізнес-процес: контроль за обігом фінансових ресурсів

Словесний опис: процедура перевіряє чи активність пов'язана з рахунком завершення, у разі ствердного результату рахунок може бути деактивовано

Скрипт:

CREATE OR REPLACE PROCEDURE deactivate_account(target_account INT)

AS

\$body\$

```

DECLARE
money_residue account.current_amount%TYPE;
connected_activity INT;
BEGIN
SELECT ac.activity_id INTO connected_activity FROM account ac
WHERE ac.id = target_account;
IF target_account IS NOT NULL THEN
    IF is_activity_finished(connected_activity) THEN
        SELECT ac.current_amount INTO money_residue FROM account ac
WHERE
        ac.id = target_account;

        UPDATE account SET current_amount =
current_amount+money_residue
        WHERE account.type = 'Main';

        UPDATE account SET current_amount = 0
        WHERE account.id = target_account;
    ELSE
        RAISE EXCEPTION 'Associated activity is not finished yet.';
    END IF;
ELSE
    RAISE EXCEPTION 'Can not deactivate MAIN account';
END IF;
END
$body$
LANGUAGE plpgsql;

```

- Тест роботи:
CALL deactivate_account(1);

ERROR: Associated activity is not finished yet.

Рис. 6.3.29 Процедура №7

SELECT * FROM contract

id [PK] integer	supplier_id integer	status_id integer	title character varying (255)	conclude_date date	fulfill_date date	feedback smallint
1	1	5	Chinese UAV	2024-01-23	2024-01-27	[null]
17	1	5	Another_attempt to buy Mavic	2024-01-25	2024-01-28	[null]
18	11	4	Agreement with Righmetal for Mavic	2024-01-25	2024-01-28	10

Рис. 6.3.30 Процедура №7

CALL deactivate_account(1);

id [PK] integer	activity_id integer	type character varying (15)	current_amount numeric (20,8)	target_amount numeric (20,8)	reserved_amount numeric (20,8)
2	[null]	Main	36650.00000000	[null]	[null]
1	2	Secondary	0.00000000	290000.00000000	285000.00000000

Рис. 6.3.31 Процедура №7

Процедура №8

- Призначення: взяття коштів з основного рахунку на користь інших
- Бізнес-процес: заключна частина механізму внутрішнього перерозподілу ресурсів
- Словесний опис: процедура знімає вказану кількість грошей з основного рахунку та переводить її на цільовий. Перед транзакцією проводиться перевірка наявності достатньої кількості коштів.
- Скрипт:

```
CREATE OR REPLACE PROCEDURE grab_from_main(target_account INT,
amount DECIMAL DEFAULT -1)
```

```
AS
```

```
$body$
```

```
DECLARE
```

```

money_resource DECIMAL;
BEGIN
    SELECT current_amount INTO money_resource
    FROM account
    WHERE type = 'Main';
    IF(money_resource <> 0) THEN
        IF amount = -1 THEN
            UPDATE account
            SET current_amount = current_amount + money_resource
            WHERE id = target_account;
        ELSIF amount > 0 AND amount <= money_resource THEN
            UPDATE account
            SET current_amount = current_amount + amount
            WHERE id = target_account;
        ELSE
            RAISE EXCEPTION 'Insufficient funds in Main account';
        END IF;
    ELSE
        RAISE EXCEPTION 'Main account is empty';
    END IF;
END
$body$
LANGUAGE plpgsql;

```

- *Текст побору:*

```
SELECT * FROM account;
```

id [PK] integer	activity_id integer	type character varying (15)	current_amount numeric (20,8)	target_amount numeric (20,8)	reserved_amount numeric (20,8)
2	[null]	Main	36650.00000000	[null]	[null]
1	2	Secondary	0.00000000	290000.00000000	285000.00000000
10	10	Secondary	0.00000000	2700.00000000	0.00000000

Рис. 6.3.32 Процедура №8

```
CALL grab_from_main(10,1000);
SELECT * FROM account;
```

id [PK] integer	activity_id integer	type character varying (15)	current_amount numeric (20,8)	target_amount numeric (20,8)	reserved_amount numeric (20,8)
2	[null]	Main	36650.00000000	[null]	[null]
1	2	Secondary	0.00000000	290000.00000000	285000.00000000
10	10	Secondary	1000.00000000	2700.00000000	0.00000000

Рис. 6.3.33 Процедура №8

6.3 Тексти тригерів

Тригер №1

- *Призначення:* попередження реєстрації осіб, яких не можна легально залучати до праці, незважаючи на добровільні основи
- *Бізнес-процес:* контроль за дотриманням юридичних нормативів при веденні волонтерської діяльності
- *Словесний опис:* цей тригер застерігає від додавання до бази даних особи вік, якої становить менше 14 років. Якщо це так, то вхідний кортеж відкидається, натомість генерується виключення.
- *Скрипт:*

```
CREATE OR REPLACE FUNCTION check_volunteer_age()
```

```
RETURNS TRIGGER AS
```

```
$body$
```

```
BEGIN
```

```
    IF NEW.date_of_birth > CURRENT_DATE - INTERVAL '14 years' THEN
```

```
        RAISE EXCEPTION 'Volunteer must be at least 14 years old. Due to lawful restrictions' ;
```

```
    END IF;
```

```
    RETURN NEW;
```

```
END
```

```
$body$
```

```
LANGUAGE plpgsql;
```

```
CREATE TRIGGER volunteer_age_check
BEFORE INSERT ON volunteer
FOR EACH ROW
EXECUTE FUNCTION check_volunteer_age();
```

- *Тест роботи:*

```
INSERT INTO volunteer (first_name, last_name, date_of_birth, gender,
phone_number, email, password)
VALUES ('John', 'Doe', '2015-05-15', 'M', '+1234567890',
'john.doe@example.com', 'hashed_password');
```

```
ERROR: Volunteer must be at least 14 years old. For the reasons of law
```

Рис 6.4.1 Тригер №1

Тригер №2

- *Призначення:* відбиття усіх змін поля доручення
- *Бізнес-процес:* контроль стадій завершеності завдання
- *Словесний опис:* тригер заносить в журнал записи про стан завдання та людей пов'язаних із завданням після кожного оновлення статусу
- *Скрипт:*

```
CREATE OR REPLACE FUNCTION task_update_trigger()
RETURNS TRIGGER AS
$body$
DECLARE
    temp_id_spec INT;
BEGIN
```

```

FOR temp_id_spec IN SELECT DISTINCT vt.volunteer_id FROM
volunteer_task vt

WHERE task_id = NEW.id

LOOP

INSERT INTO task_log (task_id, volunteer_id, date, time, status_id)

VALUES (NEW.id, temp_id_spec, CURRENT_DATE,
CURRENT_TIME, NEW.status_id); END LOOP;

IF NEW.status_id = (SELECT id FROM status WHERE name = 'Cancelled')

OR NEW.status_id = (SELECT id FROM status WHERE name = 'Finished')

THEN

DELETE FROM volunteer_task WHERE task_id = NEW. id;

END IF; RETURN NEW;

END; $body$ LANGUAGE plpgsql;

CREATE TRIGGER after_task_update
AFTER UPDATE ON task FOR EACH ROW
EXECUTE FUNCTION task_update_trigger();

```

- *Течт роботу:*

id [PK] integer	task_id integer	volunteer_id integer	date date	time time without time zone	status_id integer
1	2	3	2024-01-22	22:07:50.478451	2
2	4	6	2024-01-22	22:11:22.161257	2
3	4	8	2024-01-22	22:11:22.161257	2
4	2	3	2024-01-22	23:14:48.978505	3
5	7	21	2024-01-25	18:51:17.445911	2
6	7	21	2024-01-25	19:06:35.564075	3
9	7	21	2024-01-25	19:12:55.568948	4
10	8	7	2024-01-25	19:17:06.753634	2
11	8	7	2024-01-25	19:19:49.064632	5
12	8	7	2024-01-25	19:20:15.946365	2
13	8	7	2024-01-25	19:21:30.73573	5

Рис 6.4.2 Тригер №2

SELECT * FROM volunteer_task;

volunteer_id [PK] integer	task_id [PK] integer
1	1
5	3
15	3
3	2
8	4
6	4
10	6

Рис 6.4.3 Тригер №2

CALL accept_task(5,3);

SELECT * FROM task_log;

id [PK] integer	task_id integer	volunteer_id integer	date date	time time without time zone	status_id integer
14	3	5	2024-01-26	01:14:46.621126	3
15	3	15	2024-01-26	01:14:46.621126	3

Рис 6.4.4 Тригер №2

CALL finish_task(5,3);

16	3	5	2024-01-26	01:17:20.953732	4
17	3	15	2024-01-26	01:17:20.953732	4

Рис 6.4.5 Тригер №2

Тригер №3

- *Призначення:* слідкування за приналежністю предмета чи послуги
- *Бізнес-процес:* є складовою нагляду за звітністю та обігом товарів/послуг
- *Словесний опис:* позиція може мати прив'язку лише до рахунку, коли закупівля ініційована зсередини, або мати прив'язку лише до запиту, коли запит ще не взято на опрацювання, або обидві прив'язки, тож мусить належати хоч кудись.
- *Скрипт:*

```
CREATE OR REPLACE FUNCTION tfu_item_belongs_check()
```

```
RETURNS TRIGGER AS
```

```
$body$
```

```
BEGIN
```

```
    IF (NEW.account_id IS NULL AND NEW.request_id IS NULL) THEN
```

```
        RAISE EXCEPTION 'Item must belongs to either an account or a request, or both, but not neither.';
```

```
    END IF;
```

```
    RETURN NEW;
```

```
END
```

```
$body$
```

```
LANGUAGE plpgsql;
```

```
CREATE TRIGGER tg_item_belongs_check
```

```
BEFORE INSERT ON item
FOR EACH ROW
EXECUTE PROCEDURE tfu_item_belongs_check();
```

- *Тест роботи:*

```
INSERT INTO item(request_id,naming,quantity,estimated_price)
VALUES(NULL,'Roof panel',15,2500);
```

```
ERROR: Item must belong to either an account or a request, or both, but not neither.
```

Рис 6.4.6 Тригер №3

Тригер №4

- *Призначення:* оновлення лічильника зібраних коштів на пов'язаному рахунку
- *Бізнес-процес:* проведення благодійних зборів коштів на допомогу
- *Словесний опис:* щоразу, коли надходить пожертва, тригер викликає функцію, яка збільшує поточну суму на рахунку на суму пожертви.
- *Скрипт:*

```
CREATE OR REPLACE FUNCTION refresh_collected_money()
```

```
RETURNS TRIGGER AS
```

```
$body$
```

```
DECLARE
```

```
temp_money DECIMAL(18,8);
```

```
BEGIN
```

```
SELECT a.current_amount INTO temp_money
```

```
FROM account a
```

```

WHERE a.id = NEW.account_id;

UPDATE account

SET current_amount = NEW.amount + temp_money

WHERE account.id = NEW.account_id;

RETURN NEW;

END

$body$

LANGUAGE plpgsql;

```

```

CREATE OR REPLACE TRIGGER register_income

AFTER INSERT ON income

FOR EACH ROW

EXECUTE FUNCTION refresh_collected_money();

```

- *Тест роботу*

```

INSERT INTO income(account_id,bank, date, time,amount,comment)

VALUES(1,'Pumb', CURRENT_DATE, CURRENT_TIME,320000,'For mavic drone');

```

id [PK] integer	activity_id integer	type character varying (15)	current_amount numeric (20,8)	target_amount numeric (20,8)	reserved_amount numeric (20,8)
2	[null]	Main	0.00000000	[null]	[null]
1	2	Secondary	321650.00000000	290000.00000000	[null]

Рис 6.4.7 Тригер №4

```

SELECT amount FROM income WHERE account_id = 1;

```

amount	🔒
numeric (18,8)	
100.00000000	
1200.00000000	
320000.00000000	

Рис 6.4.8 Тригер №4

Тригер №5

- *Призначення:* запобігти знищенню основного накопичувального рахунка
- *Бізнес-процес:* є частиною процесу переадресації невикористаних коштів
- *Словесний опис:* цей тригер налаштовано на виконання перевірки перед операцією видалення в таблиці 'account' чи обліковий запис, який видаляється, є головним рахунком. Якщо так, вона генерує виключення, запобігаючи видаленню і зберігаючи цілісність даних.
- *Скрипт:*

```

CREATE OR REPLACE FUNCTION check_account_before_deletion()
RETURNS TRIGGER
AS
$body$
BEGIN
    IF OLD.type = 'Main' THEN
        RAISE EXCEPTION 'Deletion of main account is not allowed';
    END IF;
    RETURN OLD;
END
$body$

```

```
LANGUAGE plpgsql;
```

```
CREATE TRIGGER prevent_deletion
```

```
BEFORE DELETE ON account
```

```
FOR EACH ROW
```

```
EXECUTE PROCEDURE check_account_before_deletion();
```

- *Тест роботи:*

```
SELECT * FROM account
```

id [PK] integer	activity_id integer	type character varying (15)	current_amount numeric (20,8)	target_amount numeric (20,8)	reserved_amount numeric (20,8)
2	[null]	Main	0.00000000	[null]	[null]

Рис 6.4.9 Тригер №5

```
DELETE FROM account WHERE id = 2;
```

```
ERROR:  Deletion of main account is not allowed
```

Рис 6.4.10 Тригер №5

Тригер №6

- *Призначення:* збільшення показника цільової суми разом зі збільшенням сумарної ціни предметів, які заплановано придбати
- *Бізнес-процес:* є частиною процесу контролю обігу ресурсів
- *Словесний опис:* цей тригер гарантує, що кожного разу, коли в таблицю предметів/послуг вставляється нова позиція, цільова сума відповідного рахунку автоматично оновлюється, щоб відобразити додану розрахункову ціну.
- *Скрипт:*

```
CREATE OR REPLACE FUNCTION update_target_amount()
```

RETURNS TRIGGER AS

\$body\$

BEGIN

 UPDATE account

 SET target_amount := target_amount + NEW.estimated_price

 WHERE id = NEW.account_id;

 RETURN NEW;

END;

\$body\$

LANGUAGE plpgsql;

CREATE TRIGGER item_insert_trigger

AFTER INSERT ON item

FOR EACH ROW

EXECUTE FUNCTION update_target_amount();

- *Тест роботи:*

id [PK] integer	activity_id integer	type character varying (15)	current_amount numeric (20,8)	target_amount numeric (20,8)	reserved_amount numeric (20,8)
2	[null]	Main	36650.00000000	[null]	[null]
1	2	Secondary	0.00000000	290000.00000000	285000.00000000
10	10	Secondary	0.00000000	2500.00000000	0.00000000

Рис 6.4.11 Тригер №6

INSERT INTO item(account_id, request_id,naming,quantity,estimated_price)

VALUES(10, 6,'Nails',100,200);

id [PK] integer	activity_id integer	type character varying (15)	current_amount numeric (20,8)	target_amount numeric (20,8)	reserved_amount numeric (20,8)
2	[null]	Main	36650.00000000	[null]	[null]
1	2	Secondary	0.00000000	290000.00000000	285000.00000000
10	10	Secondary	0.00000000	2700.00000000	0.00000000

Рис 6.4.12 Тригер №6

6.4 Тексти представлень

Представлення №1

- *Призначення:* відсів тимчасово або повністю завершених активностей
- *Бізнес-процес:* швидкий доступ до даних про поточні активності
- *Словесний опис:* відображає поточні активності через призму функції оцінки
- Скрипт:

```
CREATE OR REPLACE VIEW v_actual_activities
```

```
AS
```

```
SELECT
```

```
  a.objective, a.description,
```

```
  a.organizer,
```

```
  ac.target_amount - ac.current_amount + ac.reserved_amount AS needed_money
```

```
FROM activity a
```

```
LEFT JOIN account ac ON ac.activity_id = a.id
```

```
WHERE a.id IS NULL OR is_activity_finished(a.id) = false;
```

- *Тест роботи:*
SELECT * FROM v_actual_activities;

objective character varying (255)	description text	organizer character varying (255)	needed_money numeric
Repair roof for school	[null]	University org & UkrBuild	1700.00000000
Art Exhibition	Showcasing local artists and their creative works	City Arts Council	200.00000000
Annual EasterFestival	We will cheer with childrens	Little hope foundation	[null]
Ecology flashmob	Clean the local forest. Open recycle fest	Green peace	[null]
Tech Innovators Conference	Showcasing the latest in technology and innovation	TechHub Organization	[null]
Community Health Fair	Promote healthy living with workshops and free checku...	Local Health Department	[null]
Help to the writer Alan Wake	Teach him how do rendering in Blender 3D	University org & Federal Control	[null]
Test activity	Only for testing purposes	Testing Department	[null]
Meeting with Team	Discuss project updates and timelines	CDPR	[null]

Рис 6.4.1 Представлення №1

Представлення №2

- *Призначення:* відображає рейтинг волонтерів у поточному місяці
- *Бізнес-процес:* представлення допомагає волонтерам отримати визнання їхньої праці
- *Словесний опис:* представлення створене для швидкого та зручного перегляду топ-50 волонтерів. Його основа - обчислення балів, які кожен волонтер отримав за виконання завдань. Підрахунок здійснюється на основі тривалості та кількості завдань, які вони успішно виконали.
- *Скрипт:*

```
CREATE OR REPLACE VIEW v_volunteer_leaderboard_top_50 AS
WITH id_points AS (SELECT vol.id, SUM(t.duration/task_occurence) AS points
FROM volunteer vol
JOIN task_log tl ON tl.volunteer_id = vol.id
JOIN task t ON t.id = tl.task_id
JOIN (SELECT task_id, COUNT(*) AS task_occurence FROM task_log
WHERE task_log.status_id = (SELECT id FROM status WHERE name =
'Finished'))
GROUP BY task_id) AS task_count
ON task_count.task_id = t.id
WHERE EXTRACT(MONTH FROM tl.date) = EXTRACT(MONTH FROM
CURRENT_DATE)
```



```

AND EXTRACT(YEAR FROM tl.date) = EXTRACT(YEAR FROM
CURRENT_DATE)

AND tl.status_id = (SELECT id FROM status WHERE name = 'Finished')
GROUP BY vol.id)
SELECT volunteer.first_name || ' ' || last_name AS full_name,
points FROM id_points
JOIN volunteer
ON volunteer.id = id_points.id
ORDER BY points DESC LIMIT 50;

```

- *Тест роботи:*

```
SELECT * FROM v_volunteer_leaderboard_top_50;
```

full_name text	points numeric
David Brown	3
William Anderson	3
Licha Blewis	1

Рис 6.4.2 Представлення №2

Представлення №3

- *Призначення:* показ інформації про волонтерів та їхні корисні вміння
- *Бізнес-процес:* є складовою механізму підбору найліпших кандидатів на виконання доручення
- *Словесний опис:* комбінує ініціали особи, контактний телефон та опис навички. Такий підхід дозволяє спростити роботу організатора.
- *Скрипт:*

```

CREATE VIEW v_volunteer_competencies AS

SELECT vol.id, vol.first_name || ' ' || vol.last_name AS full_name,
vol.phone_number, cc.name AS skill, cc.level

```

FROM volunteer vol

JOIN volunteer_competency vc

ON vc.volunteer_id = vol.id

JOIN critical_competency cc

ON cc.id = vc.competency_id

- *Тест роботи*

SELECT * FROM v_volunteer_competencies;

id integer	full_name text	phone_number character varying (30)	skill character varying (255)	level character varying (100)
1	John Doe	+380971234567	Driver lisence	A
10	Sophia Hill	+380971234568	Driver lisence	A
5	David Brown	+380941234567	Driver lisence	B
15	William Anderson	+380981234568	English	B1
21	Licha Blewis	982-610-9071	Java Programing	Junior
30	Felecia Bruck	834-983-0496	Blender 3D	Basic
33	Michaeline Macr...	796-723-0864	Blender 3D	Basic
50	Dyana Hearne	924-414-7042	Blender 3D	Advanced
55	Octavia Labone	921-560-8775	Blender 3D	Advanced
62	Kellen Duran	758-518-4870	Driver lisence	A
23	Sandor Mahon	297-445-1769	Driver lisence	C
57	Manon Marquese	947-776-7000	Driver lisence	C
71	Thorndike Dockrill	760-387-7804	Photography	Basic
120	Cathy Gillis	788-627-5694	Photography	Basic
2	Jane Smith	+380991234567	Photography	Basic
2	Jane Smith	+380991234567	Photography	Advanced
3	Michael Johnson	+380931234567	Photography	Advanced
140	Muffin Huyche	548-974-6916	Photography	Advanced
127	Harlin Lammerdi...	905-878-3682	German	A2
128	Rice Dobrowski	457-972-7647	German	A2
7	Daniel Miller	+380901234567	German	A2
9	James Moore	+380921234567	German	B1

Рис 6.4.3 Представлення №3

6.5 Тексти запитів

Запит №1

- Призначення: вивести перелік всіх товарів послуг разом з інформацією про контракт та постачальника
- Бізнес-процес: аналіз постачальників та об'ємів закупівель
- Скрипт:

```
SELECT i.id AS item_id, i.naming AS item_name, c.title AS contract_title, s.title AS supplier_title
```

```
FROM item i
```

```
JOIN contract c ON i.contract_id = c.id
```

```
JOIN supplier s ON c.supplier_id = s.id;
```

item_id integer	item_name character varying (255)	contract_title character varying (255)	supplier_title character varying (255)
3	Mavic 3T	Agreement with Righmetal for Mavic	Reignmetal

Рис 6.5.1 Запит №1

Запит №2

- Призначення: вибірка волонтерів разом із описом завдань, які вони виконують
- Бізнес-процес: спостереження за зайнятістю добровольців
- Скрипт:

```
SELECT v.first_name || ' ' || v.last_name AS volunteer_name, t.description AS task_description
```

```
FROM volunteer v
```

```
JOIN volunteer_task vt ON v.id = vt.volunteer_id
```

```
JOIN task t ON vt.task_id = t.id;
```

volunteer_name text	task_description text
John Doe	Test task 1
Michael Johnson	Test task 2
Olivia Davis	Test task 4
Emily Jones	Test task 4
Sophia Hill	Prepare agenda for the meeting

Рис 6.5.2 Запит №1

6.6 Оптимізація

EXPLAIN ANALYZE SELECT date_of_birth FROM volunteer WHERE
date_of_birth = DATE'2005-08-20';

Статистичні дані пошуку для неоптимізованої таблиці:


QUERY PLAN	
text	
Seq Scan on volunteer (cost=0.00..29.74 rows=1 width=4) (actual time=0.301..0.302 rows=1 loops...	
Filter: (date_of_birth = '2005-08-20'::date)	
Rows Removed by Filter: 1018	
Planning Time: 0.129 ms	
Execution Time: 0.321 ms	

Рис 6.6.1 Послідовний пошук

Послідовний пошук має лінійну часову складність, а отже займає час пропорційний до розміру таблиці.

Для прискорення пошуку за затребуваним полем є сенс створити індекс.

CREATE INDEX volunteer_birthday_idx ON volunteer(date_of_birth);


QUERY PLAN	
text	
Index Only Scan using volunteer_birthday_idx on volunteer (cost=0.28..4.29 rows=1 width=4) (actual time=0.021..0.022 rows=1 loop...	
Index Cond: (date_of_birth = '2005-08-20'::date)	
Heap Fetches: 0	
Planning Time: 0.130 ms	
Execution Time: 0.041 ms	

Рис 6.6.2 Індексний пошук

Пошук за індексом наочно демонструє перевагу над попередником. Час планування пошуку в обох випадках майже однаковий, однак час виконання другого пошуку майже у вісім разів менший. При цьому важливим нюансом

використання індексного підходу є його підсумковий вплив на розмір бази даних. А отже, треба розважливо вирішувати, який саме домен слід індексувати

6.7 Додавання користувачів

```
CREATE ROLE volunteer_manager;  
  
GRANT SELECT,  
  
INSERT,  
  
UPDATE ON v_actual_activities TO volunteer_manager;  
  
GRANT SELECT,  
  
INSERT,  
  
UPDATE,  
  
DELETE ON task TO volunteer_manager;  
  
GRANT SELECT,  
  
INSERT,  
  
UPDATE,  
  
DELETE ON location TO volunteer_manager;  
  
GRANT EXECUTE ON PROCEDURE assign_validated(INT, INT[], INT) TO  
volunteer_manager;  
  
GRANT SELECT ON v_volunteer_competencies TO volunteer_manager;  
  
GRANT SELECT ON task_log TO volunteer_manager;  
  
GRANT SELECT ON v_volunteer_leaderboard_top_50 TO volunteer_manager;  
  
CREATE ROLE volunteer;
```

```
GRANT SELECT,
INSERT,
UPDATE,
DELETE ON volunteer TO volunteer;

GRANT SELECT,
INSERT,
UPDATE,
DELETE ON critical_competency TO volunteer;

GRANT EXECUTE ON PROCEDURE accept_task(INT, INT) TO volunteer;
GRANT EXECUTE ON PROCEDURE cancel_task(INT, INT) TO volunteer;
GRANT EXECUTE ON PROCEDURE finish_task(INT, INT) TO volunteer;
GRANT SELECT ON task TO volunteer;
GRANT SELECT ON volunteer_task TO volunteer;


CREATE ROLE help_seeker;

GRANT SELECT,
INSERT,
UPDATE ON receiver TO help_seeker;

GRANT SELECT,
INSERT,
UPDATE,
DELETE ON request TO help_seeker;
```

GRANT SELECT,

INSERT,

UPDATE ON item TO help_seeker;

CREATE ROLE request_operator;

GRANT SELECT ON request TO request_operator;

GRANT SELECT ON account TO request_operator;

GRANT SELECT ON income TO request_operator;

GRANT SELECT ON activity TO request_operator;

GRANT EXECUTE ON PROCEDURE accept_requests(INT[], VARCHAR(255),
VARCHAR(255), DATE) TO request_operator;

CREATE ROLE supply_manager;

GRANT SELECT,

INSERT,

UPDATE ON supplier TO supply_manager;

GRANT SELECT,

UPDATE,

DELETE ON contract TO supply_manager;

GRANT SELECT,

UPDATE ON item TO supply_manager;

GRANT SELECT,

UPDATE ON account TO supply_manager;

```

GRANT SELECT,
UPDATE ON category TO supply_manager;

GRANT EXECUTE ON PROCEDURE conclude_contract(VARCHAR(255),
CHAR(34), VARCHAR(255), INT[], DATE, DECIMAL(18,8), DATE)
TO supply_manager;

GRANT EXECUTE ON PROCEDURE cancel_contract(INT) TO supply_manager;

GRANT EXECUTE ON PROCEDURE deactivate_account(INT) TO
supply_manager;

GRANT EXECUTE ON PROCEDURE grab_from_main(INT, DECIMAL(20,8))
TO supply_manager;


CREATE ROLE contributor;

GRANT SELECT ON activity TO contributor;

GRANT SELECT ON account TO contributor;

GRANT SELECT ON item TO contributor;

GRANT SELECT ON income TO contributor;

GRANT EXECUTE ON FUNCTION generate_receipt(INT) TO contributor;


CREATE USER test_volunteer_manager WITH PASSWORD 'volunteer_manager';
CREATE USER test_volunteer WITH PASSWORD 'volunteer';
CREATE USER test_help_seeker WITH PASSWORD 'help_seeker';
CREATE USER test_request_operator WITH PASSWORD 'request_operator';
CREATE USER test_supply_manager WITH PASSWORD 'supply_manager';

```



```
CREATE USER test_contributor WITH PASSWORD 'contributor';
```

```
GRANT volunteer_manager TO test_volunteer_manager;
```

```
GRANT volunteer TO test_volunteer;
```

```
GRANT help_seeker TO test_help_seeker;
```

```
GRANT request_operator TO test_request_operator;
```

```
GRANT supply_manager TO test_supply_manager;
```

```
GRANT contributor TO test_contributor;
```

Висновок: у цьому розділі було описано поетапну розбудову інфраструктури бази даних. Усі набуті функціональні властивості ідейно діляться на два типи: керуючі та оглядові. До першого типу належать функції, процедури й тригери. Вони утворюють автоматизовані ланки в просторі бізнес-процесів. Окрім економії часу, цей дієвий комплект запобігає грубим порушенням цілісності, чого не завжди можна досягти внутрішньотабличними обмеженнями. Функціонал другої категорії підлаштований до рольової схеми, ізолюючи область видимості одного користувача від іншого. І наостанок, з огляду на потенційне зростання об'єму даних було випробувано класичний спосіб оптимізації у вигляді пошукової структури індексу.

ВИСНОВКИ

Даний курсовий проект охоплює всі стадії розробки інформаційної системи. У конкретному випадку реляційної бази даних. До того як взятися за реалізацію коду було проведено розбір предметного середовища на окремі компоненти. Для розуміння ринку порівняно близьке порівняння кращих технологічних напрацювань. Зваживши всі за та проти було прийнято рішення про використання системи управління базами даних PostgreSQL. Ця СУБД пропонує гнучку типізацію, підтримку стандартів ACID, широку сумісність з різноманітними платформами і ще багато приємних дрібниць за публічною ліцензією. Таким чином будь-який продукт на її основі апріорі є безкоштовним, якщо не враховувати ресурси, що витрачені на розробку. Структурно базу даних створено з урахуванням встановлених вимог. Усі таблиці інтегровані між собою та обладнані засобами збереження цілісності. Реалізовано SQL-запити, генератори, збережені процедури, тригери та інші елементи, що сприяють оптимізації та ефективності взаємодії з базою даних. Результатом прикладених зусиль є модульна, масштабована система з глибокого продуманими рольовими напрямками та здатна витримувати одночасне використання багатьма користувачами. Загалом, робота цілком відповідає усім пунктам постановки завдання.

ДОДАТОК

Нижче наведено список файлів, що містять інформацію з однойменних таблиць в базі даних:

1. receiver.csv
2. activity.csv
3. request.csv
4. location.csv
5. account.csv
6. income.csv
7. status.csv
8. supplier.csv
9. contract.csv
- 10.category.csv
- 11.item.csv
- 12.volunteer.csv
- 13.critical_competency.csv
- 14.task.csv
- 15.task_log.csv
- 16.task_competency.csv
- 17.volunteer_competency.csv
- 18.volunteer_task.csv

СПИСОК ДЖЕРЕЛ

1. Про волонтерство простими словами

<https://platforma.volunteer.country/posts/pro-volonterstvo-prostymy-slovamy>

(Дата звернення 10.01.2024)

2. Горінов П., Драпушко Р. "Волонтерська діяльність в Україні: соціально-правове дослідження. Монографія." Київ: Державний інститут сімейної та молодіжної політики, 2022.

3. ЗАКОН УКРАЇНИ Про волонтерську діяльність

<https://zakon.rada.gov.ua/laws/show/3236-17#Text>

(Дата звернення 10.01.2024)

4. Матвійчик А. Громадянська ініціатива як чинник самоорганізації громадянського суспільства в сучасній Україні. Наукові записки, 3-4, 2016

5. Менеджмент волонтерських груп від А до Я: навч.-метод. посібник / За ред. Т.Л. Лях; авт.-кол.: З.П. Бондаренко, Т.В. Журавель, Т.Л. Лях та ін. – К.: Версо-04, 2012. – 288 с.

6. Charity organizations or NGO <https://babel.ua/en/profit/77389-i-want-to-volunteer-during-the-war-how-to-start-should-i-launch-some-organization-can-i-raise-money-via-my-personal-bank-card-the-lawyer-explains-the-rules-in-ukraine>

(Дата звернення 11.01.2024)

7. VolunteerHub BetterGood Product <https://volunteerhub.com/>

(Дата звернення 12.01.2024)

8. Volunteer Logistics <https://www.volgistics.com/vicnet.htm>

(Дата звернення 12.01.2024)

9. Volunteer management. Tools For Success

<https://learn.givepulse.com/volunteer-management>

(Дата звернення 12.01.2024)

10. PostgreSQL Documentation <https://www.postgresql.org/docs/>

(Дата звернення 14.01.2024)