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 \_\_ІП-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](#_Toc157165634)

[ВСТУП 5](#_Toc157165635)

[1 ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА 6](#_Toc157165636)

[1.1 Розбір об’єкту дослідження 6](#_Toc157165637)

[1.2 Вхідні і вихідні дані 7](#_Toc157165638)

[1.3 Основні бізнес-процеси використання даних 8](#_Toc157165639)

[2 АНАЛІЗ СТОРОННІХ ПРОГРАМНИХ ПРОДУКТІВ 9](#_Toc157165640)

[2.1 Сервіс VolunteerHub 9](#_Toc157165641)

[2.2 Сервіс VicNet 10](#_Toc157165642)

[2.3 Сервіс GivePulse 11](#_Toc157165643)

[3 ПОСТАНОВКА ЗАДАЧІ 12](#_Toc157165644)

[4 ОПИС КОНЦЕПТУАЛЬНОЇ МОДЕЛІ 13](#_Toc157165645)

[4.1 Опис інформаційних об’єктів 13](#_Toc157165646)

[4.2 Опис відношень 16](#_Toc157165647)

[4.3 Діаграма відношень 19](#_Toc157165648)

[5 ПОБУДОВА БАЗИ ДАНИХ 20](#_Toc157165649)

[5.1 Обґрунтування вибору системи управління базами даних 20](#_Toc157165650)

[5.2 Опис сутностей в табличному вигляді 21](#_Toc157165651)

[5.3 SQL-скрипт 30](#_Toc157165652)

[5.4 Згенерована схема бази даних 36](#_Toc157165653)

[6 РОБОТА З БАЗОЮ ДАНИХ 37](#_Toc157165654)

[6.1 Тексти генераторів 37](#_Toc157165655)

[6.2 Тексти функцій/процедур 38](#_Toc157165656)

[6.3 Тексти тригерів 69](#_Toc157165657)

[6.4 Тексти представлень 79](#_Toc157165658)

[6.5 Тексти запитів 83](#_Toc157165659)

[6.6 Оптимізація 84](#_Toc157165660)

[6.7 Додавання користувачів 85](#_Toc157165661)

[ВИСНОВКИ 90](#_Toc157165662)

[ДОДАТОК 91](#_Toc157165663)

[СПИСОК ДЖЕРЕЛ 92](#_Toc157165664)

# **ВСТУП**

Волонтерство дозволяє людям відчути свою соціальну відповідальність 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].

Початок форми

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 ввійде 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 квартальний 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 посприяє 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 відбувати онлайн або на декількох локаціях одночасно. Кожна локація призначена для конкретної активності (події).

12. **Задання** – виконується в рамках – **Активність:**

Кожне завдання належить відбувається 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*

Зображення, що містить знімок екрана, Прямокутник, Паралель, ряд

Автоматично згенерований опис

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 довірчого 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 уможливлюють 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 проведенням 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 закуповується товар |
| request\_id | INT |  | FK | Запит за яким закуповується 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 |  | PK | 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 кожну зміну 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 |  | PK, FK | Волонтер, який володіє особливістю |
| competency\_id | INT |  | PK, 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 |  | PK, FK | Волонтер, який має стосунок до завдання |
| task\_id | INT |  | PK, FK | Завдання призначене для виконання |

Рис 5.18 Таблиця «Волонтер-Завдання»

*5.3 SQL-скрипт*

-- 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*

Зображення, що містить текст, схема, Паралель, число

Автоматично згенерований опис

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 занятості 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);

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Автоматично згенерований опис

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;

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Автоматично згенерований опис

Рис. 6.2.2 Функція №2

SELECT need\_special\_skills(7);

Зображення, що містить текст, Шрифт, знімок екрана, білий

Автоматично згенерований опис

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;

* *Тест роботи:*

SELECT id FROM income

LIMIT 3;

Зображення, що містить текст, квитанція, Шрифт, знімок екрана

Автоматично згенерований опис

Рис. 6.2.4 Функція №3

SELECT generate\_receipt(8);

Зображення, що містить текст, Шрифт, знімок екрана, ряд

Автоматично згенерований опис

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);

Зображення, що містить текст, Шрифт, знімок екрана, білий

Автоматично згенерований опис

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);

Зображення, що містить текст, Шрифт, знімок екрана, білий

Автоматично згенерований опис

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.

* *Скрипт:*

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

Зображення, що містить текст, Шрифт, знімок екрана, білий

Автоматично згенерований опис

Рис. 6.3.1 Процедура №1

CALL assign\_for\_task(6,10);

Зображення, що містить текст, Шрифт, ряд, знімок екрана

Автоматично згенерований опис

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 || 'Assigment 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 || 'Assigment 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 'Assigment 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

Зображення, що містить текст, Шрифт, ряд, квитанція

Автоматично згенерований опис

Рис. 6.3.3 Процедура №2

SELECT \* FROM task\_competency WHERE competency\_id = 4

Зображення, що містить текст, Шрифт, ряд, знімок екрана

Автоматично згенерований опис

Рис. 6.3.4 Процедура №2

CALL assign\_validated(7, ARRAY[50]);



Рис. 6.3.5 Процедура №2

CALL assign\_validated(7, ARRAY[21]);

Зображення, що містить текст, Шрифт, знімок екрана, алгебра

Автоматично згенерований опис

Рис. 6.3.6 Процедура №2

Зображення, що містить текст, Шрифт, ряд, знімок екрана

Автоматично згенерований опис

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 'Accsess 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 'Accsess 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 'Accsess 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;

* *Тест роботи:*

SELECT \* FROM task WHERE id = 7

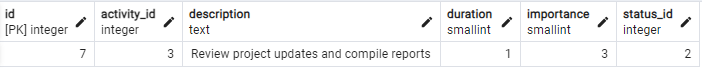
**

Рис. 6.3.8 Процедура №3

CALL accept\_task(21,7);

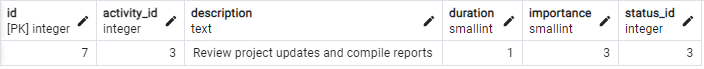


Рис. 6.3.9 Процедура №3

CALL accept\_task(2,7);





Рис. 6.3.10 Процедура №3

CALL finish\_task(21,7);

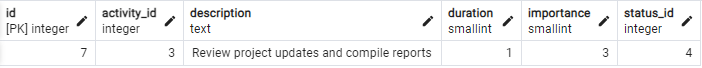


Рис. 6.3.11 Процедура №3

CALL assign\_validated(8, ARRAY[7]);

Зображення, що містить текст, Шрифт, квитанція, знімок екрана

Автоматично згенерований опис

Рис. 6.3.12 Процедура №3

SELECT \* FROM volunteer\_task WHERE task\_id = 8

Зображення, що містить текст, Шрифт, ряд, білий

Автоматично згенерований опис

Рис. 6.3.13 Процедура №3

CALL cancel\_task(7,8);

SELECT \* FROM volunteer\_task WHERE task\_id = 8;

Зображення, що містить текст, Шрифт, білий

Автоматично згенерований опис

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;

* *Тест роботи:*

SELECT \* FROM request

Зображення, що містить текст, Шрифт, знімок екрана, число

Автоматично згенерований опис

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;





Рис. 6.3.16 Процедура №4





Рис. 6.3.17 Процедура №4

**Процедура №5**

* *Призначення:* додавання запису про укладення угоди, включно з супутніми деталями
* *Бізнес-процес:* контрактування 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.
* *Скрипт:*

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

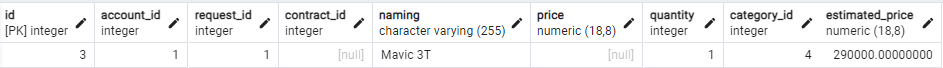


Рис. 6.3.18 Процедура №5

CALL conclude\_contract('DJI', 'NL45RABO9510165298', 'Another\_attemp to buy Mavic', ARRAY[1], DATE'2024-01-28',285000)



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



Рис. 6.3.20 Процедура №5

SELECT \* FROM contract

Зображення, що містить текст, Шрифт, ряд, число

Автоматично згенерований опис

Рис. 6.3.21 Процедура №5

SELECT \* FROM supplier

Зображення, що містить текст, Шрифт, ряд, знімок екрана

Автоматично згенерований опис

Рис. 6.3.22 Процедура №5

SELECT \* FROM account

Зображення, що містить текст, Шрифт, ряд, Паралель

Автоматично згенерований опис

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

Зображення, що містить текст, Шрифт, ряд, знімок екрана

Автоматично згенерований опис

Рис. 6.3.23 Процедура №6

SELECT \* FROM item



Рис. 6.3.24 Процедура №6

SELECT \* FROM account;

Зображення, що містить текст, Шрифт, ряд, знімок екрана

Автоматично згенерований опис

Рис. 6.3.25 Процедура №6

CALL cancel\_contract(17);

SELECT \* FROM contract;



Рис. 6.3.26 Процедура №6

SELECT \* FROM item;

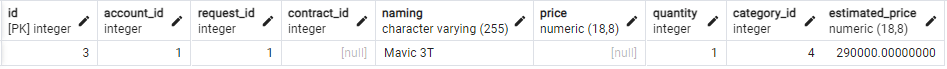


Рис. 6.3.27 Процедура №6

SELECT \* FROM account

Зображення, що містить текст, Шрифт, ряд, число

Автоматично згенерований опис

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);



Рис. 6.3.29 Процедура №7

SELECT \* FROM contract

Зображення, що містить текст, Шрифт, ряд, число

Автоматично згенерований опис

Рис. 6.3.30 Процедура №7

CALL deactivate\_account(1);

Зображення, що містить текст, Шрифт, квитанція, знімок екрана

Автоматично згенерований опис

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;

* *Тест роботи:*

SELECT \* FROM account;

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис. 6.3.32 Процедура №8

CALL grab\_from\_main(10,1000);

SELECT \* FROM account;

Зображення, що містить текст, Шрифт, знімок екрана, квитанція

Автоматично згенерований опис

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 додержанням 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');



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();

* *Тест роботи:*

Зображення, що містить текст, число, Шрифт, знімок екрана

Автоматично згенерований опис

Рис 6.4.2 Тригер №2

SELECT \* FROM volunteer\_task;

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис 6.4.3 Тригер №2

CALL accept\_task(5,3);

SELECT \* FROM task\_log;



Зображення, що містить текст, знімок екрана, ряд, Шрифт

Автоматично згенерований опис

Рис 6.4.4 Тригер №2

CALL finish\_task(5,3);



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);



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 роботи*

INSERT INTO income(account\_id,bank, date, time,amount,comment)

VALUES(1,'Pumb', CURRENT\_DATE, CURRENT\_TIME,320000,'For mavic drone');

Зображення, що містить текст, знімок екрана, Шрифт, ряд

Автоматично згенерований опис

Рис 6.4.7 Тригер №4

SELECT amount FROM income WHERE account\_id = 1;

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

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

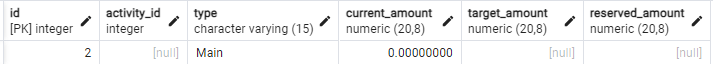


Рис 6.4.9 Тригер №5

DELETE FROM account WHERE id = 2;



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:*

*Зображення, що містить текст, Шрифт, знімок екрана, число

Автоматично згенерований опис*

Рис 6.4.11 Тригер №6

INSERT INTO item(account\_id, request\_id,naming,quantity,estimated\_price)

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

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

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;

Зображення, що містить текст, Шрифт, число, квитанція

Автоматично згенерований опис

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;

Зображення, що містить текст, Шрифт, знімок екрана, квитанція

Автоматично згенерований опис

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;

Зображення, що містить текст, число, меню, знімок екрана

Автоматично згенерований опис

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;

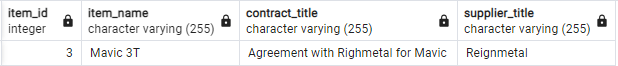
******

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;

***Зображення, що містить текст, знімок екрана, Шрифт, квитанція

Автоматично згенерований опис***

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:

Зображення, що містить текст, квитанція, знімок екрана, Шрифт

Автоматично згенерований опис

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);

Зображення, що містить текст, Шрифт, ряд, число

Автоматично згенерований опис

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 технологічний 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)