|  |
| --- |
| МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ  «ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»  Кафедра програмної інженерії та інтелектуальних технологій управління  КУРСОВА РОБОТА  з дисципліни « Проєктування і розробка баз даних »  Керівники роботи:  Зав. каф. ПІІТУ Андрій КОПП  Проф. ПІІТУ Дмитро КУДІЙ  Виконавці:  Студент групи КН-222Б Артем ГОРБАСЬ  Студент групи КН-222Б Данило ОСИПЧУК  Харків – 2024 |

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Кафедра програмної інженерії та інтелектуальних технологій управління

Оцінка

голова комісії,

/ /

« » 20 р.

КУРСОВА РОБОТА

Тема: « Система управління складом логістичної компанії »

Керівники роботи:

Зав. каф. ПІІТУ Андрій КОПП

Проф. ПІІТУ Дмитро КУДІЙ

«07» червня 2024 р.

Виконавці:

студент групи КН-222Б Артем ГОРБАСЬ

студент групи КН-222Б Данило ОСИПЧУК

«07» червня 2024 р.

Харків – 2024

Національний технічний університет

«Харківський політехнічний інститут»

Кафедра «Програмної інженерії та інтелектуальних технологій управління»

Студент - Артем ГОРБАСЬ Група КН-222Б Курс 2

Студент - Данило ОСИПЧУК Група КН-222Б Курс 2

Завдання

на курсову роботу

Тема: « Система управління складом логістичної компанії »

Короткий зміст роботи:

а) Визначення вимог до програмного забезпечення

б) Проєктування та розробка бази даних

в) Проєктування та розробка застосунку для роботи з базою даних

Завдання видано 02 березня 2024р.

Студент (виконавець курсової роботи) Артем ГОРБАСЬ

Студент (виконавець курсової роботи) Данило ОСИПЧУК

Термін захисту курсової роботи 07 червня 2024 р.

Керівник курсової роботи Андрій КОПП

Керівник курсової роботи Дмитро КУДІЙ

ВІДГУК

на курсову роботу

студентів групи КН-222Б

Артем ГОРБАСЬ та Данило ОСИПЧУК

Курсова робота присвячена розв’язанню задачі …

В ході виконання роботи було …

Протягом виконання роботи було розроблено …

На завершальному етапі виконання курсової роботи …

Курсова робота повністю відповідає поставленим вимогам та заслуговує оцінки «\_\_\_\_\_\_\_\_\_\_».

Керівник КР Андрій КОПП

Керівник КР Дмитро КУДІЙ

**ЗМІСТ**

[ПЕРЕЛІК ПОЗНАК ТА СКОРОЧЕНЬ...........................................................6](#_ПЕРЕЛІК_ПОЗНАК_ТА)

[ВИЗНАЧЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.............7](#_1_ВИЗНАЧЕННЯ_ВИМОГ)

[ПРОЄКТУВАННЯ ТА РОЗРОБКА БАЗИ ДАНИХ.....................................18](#_2_ПРОЄКТУВАННЯ_ТА)

[ПРОЄКТУВАННЯ ТА РОЗРОБКА ЗАСТОСУНКУ....................................55](#_3_ПРОЄКТУВАННЯ_ТА)

# ПЕРЕЛІК ПОЗНАК ТА СКОРОЧЕНЬ

БД – база даних.

СУБД – система управління базами даних.

DFD – Data Flow Diagram.

BR – Business requirement

FR – Functional requirement

NFR – Nonfunctional requirement

С/Д – Статичне/Динамічне.

WBS – Work-Breakdown Structure

ER – Entity-Relationship.

DDL – Data Definition Language.

SQL – Structured Query Language.

**РЕФЕРАТ**

Пояснювальна записка до КР: 64 с., 58 рис., 7 табл., 5 джерел

*Ключові слова*: БІЗНЕС, ВИМОГА, МОДЕЛЬ, СХЕМА, ДІАГРАМА АНАЛІЗ, ПРОЄКТУВАННЯ, РОЗРОБКА, БАЗА ДАНИХ, ТАБЛИЦЯ, СУБД, SQL, ЗАСТОСУНОК.

# 1 ВИЗНАЧЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**Вступ**

Зараз важко уявити компанію, яка б не використовувала програмне забезпечення, яке дозволяє автоматизувати і налагодити бізнес-процеси разом з бізнес-правилами.

Мета в даному випадку - підвищення продуктивності підприємства і мінімізація можливого людського браку.

Наш новий програмний продукт спрямовано на обробку інформації, якою володіє компанія.

**Опис проблеми**

Завдання логістичної компанії полягає у зберіганні, обробці і доставці товару. Компанія володіє великим обсягом інформації, обробку якої необхідно оптимізувати. Тому потрібен програмний продукт, що надає ефективний та зрозумілий механізм обробки інформації, щоб продуктивність підприємства була максимальною.

**Запропоноване рішення**

Логістична компанія володіє складами та товарами. До компанії поступають замовлення на постачання певних продуктів. Наша система підвищує продуктивність обробки інформації за рахунок розробки відповідної бази даних та користувацького інтерфейсу для взаємодії з базою даних.

**Формат IT рішення**

Базовою платформою IT рішення буде десктопний додаток, де користувач зможе маніпулювати ресурсами компанії. Користувач взаємодіє з інтерфейсом і наприклад додає нове замовлення. IT рішення зберігає це замовлення, та надає можливість його отримання за запитом користувача. Інформація подається у вигляді списку, рядки якого позначають конкретний атрибут. Наприклад замовлення буде відображено наступним чином:

* Товар
  + Назва: Холодильник Samsung RB33J
  + Ціна: 22 500 гривень
  + Опис: Сучасний холодильник
* Склад
  + Код: 4
  + Адреса: вулиця Козацька 25, Лисиничі, 81126
* Замовник
  + Ім’я: Петренко Андрій
  + Телефон: +380 68 222 3193
  + Пошта: andrii.petrenko@ukr.net
  + Адреса: вулиця Наукова 116, Львів, 79000
* Постачальник
  + Ім’я: Гуля Віктор
  + Телефон: +380 95 111 3913
  + ІПН: ІПН123456789123

**Цілі проєкту**

**GOAL 01 -** Запроєктувати та розробити базу даних для логістичної компанії до кінця семестру

**GOAL 02 -** Запроєктувати та розробити застосунок для взаємодії за базою даних до кінця семестру

**Моделювання предметної області**

Модель діяльності організації була зображена у нотації DFD (Data flow diagram, Діаграма потоків даних).

1. Контекстна діаграма потоків даних - зображає загальний огляд діяльності та взаємодію з зовнішніми сутностями. Зображена на рисунку 1.1
2. Діаграма потоків даних рівня 1 - зображує під-процеси, які підтримуються нашим програмним забезпеченням. Зображена на рисунку 1.2

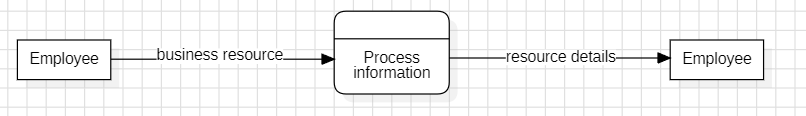
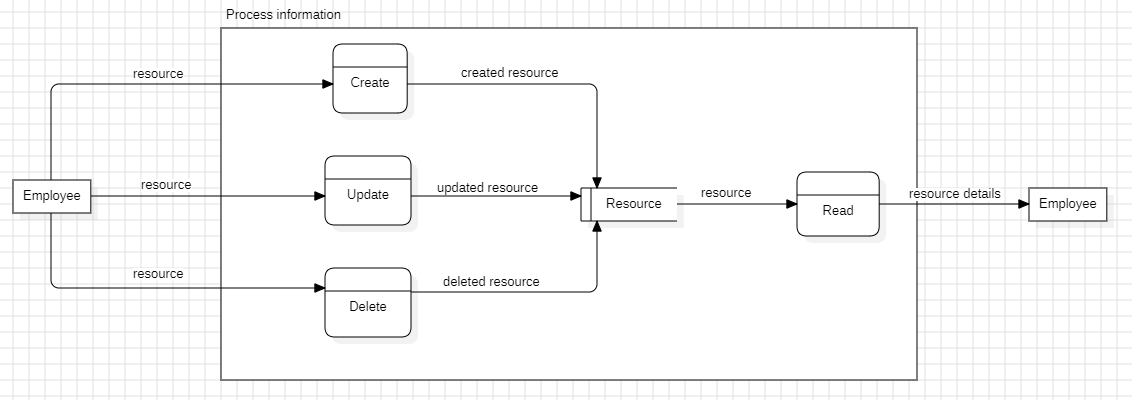


Рисунок 1.1 - Контекстна діаграма потоків даних

Рисунок 1.2 - Діаграма потоків даних рівня 1

**Бізнес вимоги**

**BR 01 Менеджмент ресурсів:** Користувач має доступ до ресурсів, якими володіє компанія**.** Програма надає ефективні механізми управління ресурсами.

**Функціональні вимоги**

**BR 01 – FR 01 Управління товарами:** Система повинна надавати користувачу можливість переглядати, додавати, змінювати та видаляти товари компанії.

**BR 01 – FR 02 Управління складами:** Система повинна надавати користувачу можливість переглядати, додавати, змінювати та видаляти зберігати склади компанії.

**BR 01 – FR 03 Управління постачальниками:** Система повинна надавати користувачу можливість переглядати, додавати, змінювати та видаляти постачальників компанії.

**BR 01 – FR 04 Управління замовниками:** Система повинна надавати користувачу можливість переглядати, додавати, змінювати та видаляти замовників компанії.

**BR 01 – FR 05 Управління замовленнями:** Система повинна надавати користувачу можливість переглядати, додавати, змінювати та видаляти замовлення, які надходять до компанії.

**Варіанти використання**

Сформульовані вимоги представленні у вигляді діаграми використання на рисунку 1.3

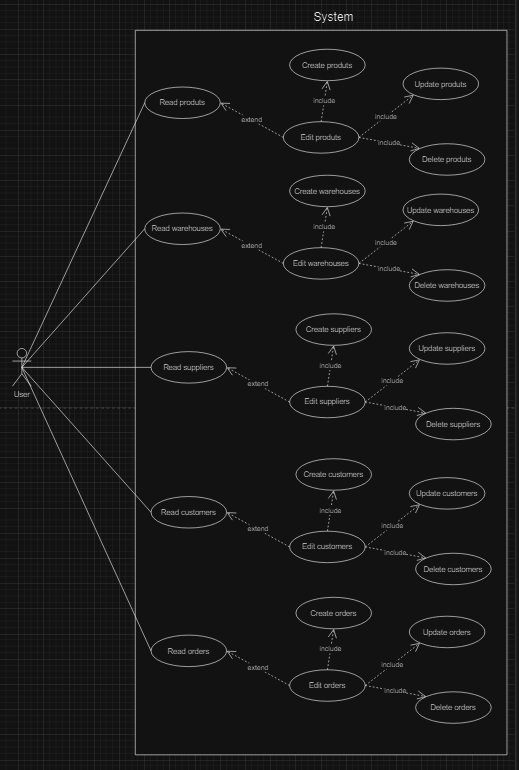


Рисунок 1.3 - Діаграма варіантів використання

**Нефункціональні вимоги**

**BR 01 – NFR 01 Зрозумілість інтерфейсу:** Інтерфейс користувача повинен бути мінімалістичним та зрозумілим для людей будь-якого віку.

**BR 01 – NFR 02 Сумісність:** IT-рішення повинно бути пристосованим до запуску на операційних системах Windows, MacOS та Linux.

**BR 01 – NFR 03 Зручність використання:** Система повинна працювати без помилок для всіх сценаріїв використання.

**BR 01 – NFR 04 Надійність:** Система повинна забезпечувати цілісність збережених даних.

Сформульовані вимоги представленні у вигляді діаграми вимог на рисунку 1.4

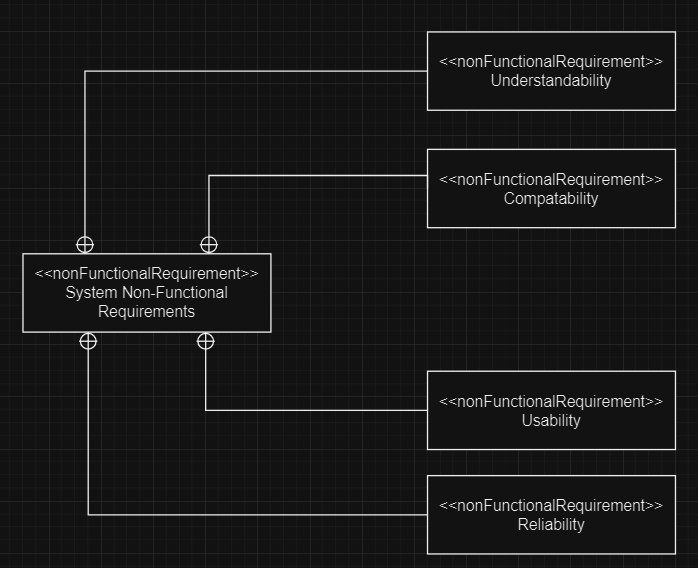


Рисунок 1.4 - Діаграма вимог

## Користувацькі історії

**US 01 – FR 02:** Як користувач програми, я хочу мати можливість перегляду списку складів разом з товарами, які на них зберігаються, щоб підвищити змістовність інформації

**US 02 – FR 04:** Як користувач програми, я хочу мати можливість перегляду списку замовників разом з замовленнями, які вони здійснили, щоб підвищити змістовність інформації

**US 03 – FR 05:** Як користувач програми, я хочу мати можливість перегляду списку постачальників разом з складами, на яких вони забирають товар, щоб підвищити змістовність інформації

**US 04 – FR 02:** Як користувач програми, я хочу мати можливість перегляду кількості товарів, які присутні на конкретному складі, щоб підвищити змістовність інформації

**US 05 – FR 03:** Як користувач програми, я хочу мати можливість перегляду кількості доставок певного постачальника, щоб підвищити змістовність інформації

**US 06 – FR 04:** Як користувач програми, я хочу мати можливість перегляду кількості замовлень певного замовника, щоб підвищити змістовність інформації

**US 07 – FR 05:** Як користувач програми, я хочу щоб при додаванні певного замовлення, запис про зберігання товару вказаного в замовлені на складі автоматично видалявся, щоб підвищити ефективність управління ресурсами

**US 08 – FR 05:** Як користувач програми, я хочу щоб при оновлені певного замовлення, вказана в замовлені дата змінювалась на дату оновлення, щоб підвищити ефективність управління ресурсами

**US 09 – FR 02:** Як користувач програми, я хочу щоб при усуванні певного складу, всі записи про зберігання товарів на цьому складі видалялися, щоб підвищити ефективність управління ресурсами

**US 10 – FR 01:** Як користувач програми, я хочу щоб при усуванні певного товару, всі записи про зберігання цього товару видалялися, щоб підвищити ефективність управління ресурсами

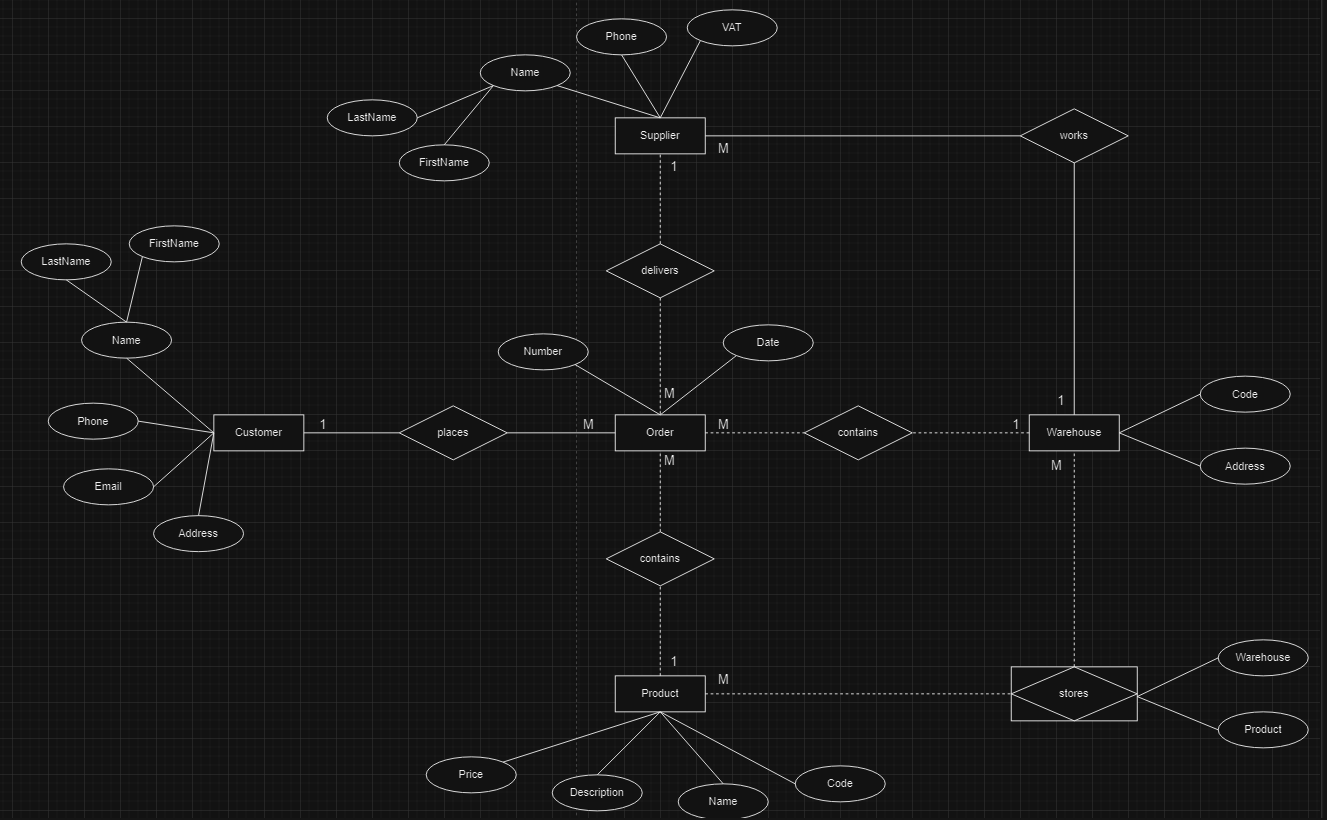
## Система бізнес-правил

В результаті аналізу та моделювання предметної області було сформовано наступну систему бізнес-правил.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Правило | Тип | С/Д |
| 01 | Товар характеризується кодом, назвою, описом та ціною | Факт | Статичне |
| 02 | Склад характеризується кодом та адресою | Факт | Статичне |
| 03 | Замовлення характеризується номером, датою, замовником, товаром, складом і постачальником | Факт | Статичне |
| 04 | Персональні дані замовника містять: ім’я, прізвище, номер телефону, електронну пошту та адресу | Факт | Статичне |
| 05 | Персональні дані постачальника містять: ім’я, прізвище, номер телефону, ІПН, та склад, з якого він постачає замовлення | Факт | Статичне |
| 06 | Замовлення містить тільки один товар | Обмеження | Статичне |
| 07 | Дата складання замовлення вказується обов’язково. У тому випадку, якщо дата не зазначена, використовується поточна дата | Обмеження | Статичне |
| 08 | Товар може бути наявним на багатьох складах | Обмеження | Статичне |
| 09 | На одному складі може бути лише одна одиниця товару | Обмеження | Статичне |
| 10 | ІПН постачальника вказується обов’язково | Обмеження | Статичне |
| 11 | ІПН та номер телефону постачальника мають бути унікальними | Обмеження | Статичне |
| 12 | Електрона пошта та номер телефону постачальника мають бути унікальними | Обмеження | Статичне |

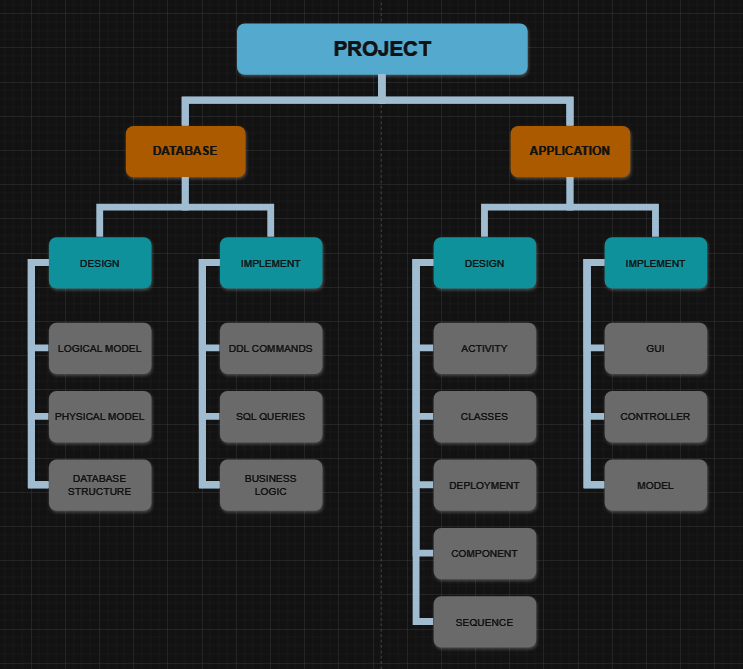
**Модель даних**

Модель предметної області у нотації ER (Entity-Relationship, Сутність-Зв’язок) наведена на рисунку 1.5

Рисунок 1.5 - ER-модель предметної області

## Постановка задачі

Задачі, які необхідно виконати для досягнення цілей, зображені у вигляді моделі WBS (Work-Breakdown Structure, Декомпозована структура праці)

Рисунок 1.6 - Work-Breakdown Structure

# 2 ПРОЄКТУВАННЯ ТА РОЗРОБКА БАЗИ ДАНИХ

## 2.1 Розробка логічної та фізичної моделей даних

Логічна модель даних у нотації Crow`s Foot наведена на рисунку 2.1

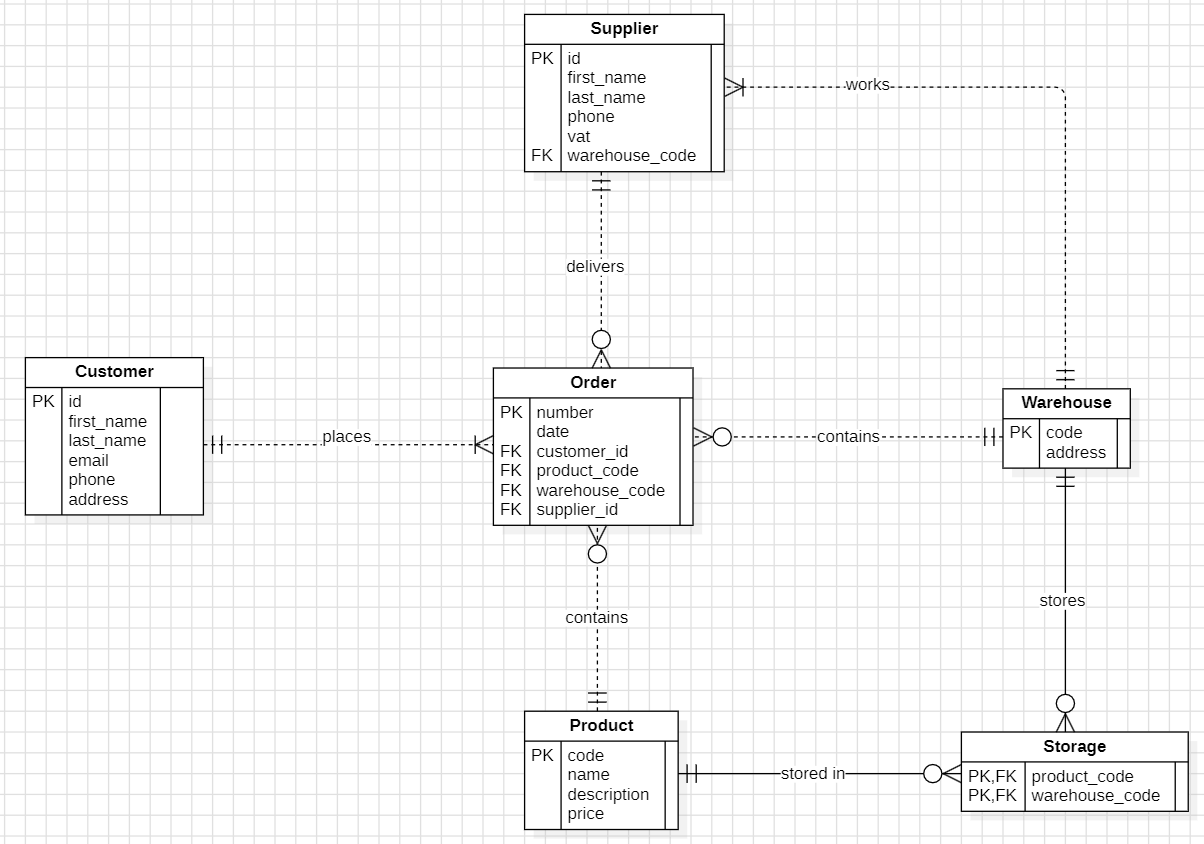


Рисунок 2.1 - Логічна модель даних

Фізична модель даних у нотації Crow`s Foot наведена на рисунку 2.2

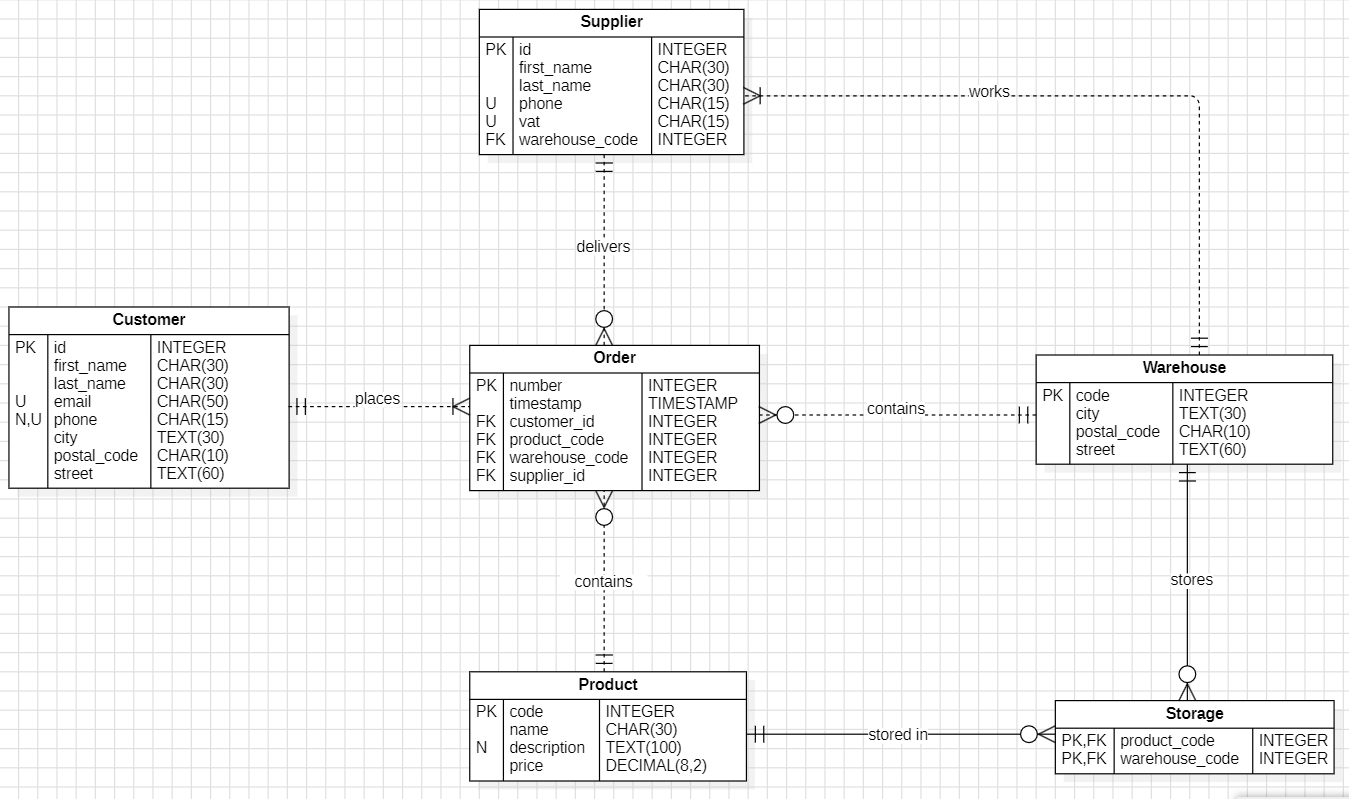


Рисунок 2.2 - Фізична модель даних

## 2.2 Опис структури бази даних

Таблиця «Product» призначена для зберігання інформації про товар, яким володіє компанія. Кожний запис таблиці складається з наступних полів, опис яких наведений в таблиці 2.1.

Таблиця 2.1 – Опис структури таблиці «Product»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Ім’я поля | Тип даних | Розмір поля | Опис |
| PK | code | Чисельний | Ціле | Код товару |
|  | name | Рядковий | 30 символів | Назва товару |
| N | description | Рядковий | 100 символів | Опис товару |
|  | price | Чисельний | Десяткове число з 6 цифрами до і 2 цифрами після десяткової точки | Ціна товару |

Таблиця «Warehouse» призначена для зберігання інформації про склади, якими володіє компанія. Кожний запис таблиці складається з наступних полів, опис яких наведений в таблиці 2.2.

Таблиця 2.2 – Опис структури таблиці «Warehouse»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Ім’я поля | Тип даних | Розмір поля | Опис |
| PK | code | Чисельний | Ціле | Код складу |
|  | city | Рядковий | 30 символів | Місто, в якому розташовано склад |
|  | postal\_code | Рядковий | 10 символів | Поштовий код міста |
|  | street | Рядковий | 60 символів | Вулиця, на якій розташовано склад |

Таблиця «Storage» призначена для реалізації зв'язку багато-до-багатьох між таблицями «Product» та «Warehouse». Містить інформацію про наявність певного товару на певному складі. Кожний запис таблиці складається з наступних полів, опис яких наведений в таблиці 2.3.

Таблиця 2.3 – Опис структури таблиці «Storage»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Ім’я поля | Тип даних | Розмір поля | Опис |
| PK, FK | product\_code | Чисельний | Ціле | Код товару |
| PK, FK | warehouse\_code | Чисельний | Ціле | Код складу |

Таблиця «Supplier» призначена для зберігання інформації про постачальників замовлень. Кожний запис таблиці складається з наступних полів, опис яких наведений в таблиці 2.4.

Таблиця 2.4 – Опис структури таблиці «Supplier»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Ім’я поля | Тип даних | Розмір поля | Опис |
| PK | id | Чисельний | Ціле | Ідентифікатор постачальника |
|  | first\_name | Рядковий | 30 символів | Ім’я постачальника |
|  | last\_name | Рядковий | 30 символів | Прізвище постачальника |
| U | phone | Рядковий | 15 символів | Телефон  постачальника |
| U | vat | Рядковий | 15 символів | ІПН постачальника |
| FK | warehouse\_code | Чисельний | Ціле | Код складу, з якого постачальник бере замовлення |

Таблиця «Сustomer» призначена для зберігання інформації про клієнтів, які замовляють товар у компанії. Кожний запис таблиці складається з наступних полів, опис яких наведений в таблиці 2.5.

Таблиця 2.5 – Опис структури таблиці «Customer»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Ім’я поля | Тип даних | Розмір поля | Опис |
| PK | id | Чисельний | Ціле | Ідентифікатор клієнта |
|  | first\_name | Рядковий | 30 символів | Ім’я клієнта |
|  | last\_name | Рядковий | 30 символів | Прізвище клієнта |
| U | email | Рядковий | 50 символів | Поштова скриня клієнта |
| N, U | phone | Рядковий | 15 символів | Номер телефону клієнта |
|  | city | Рядковий | 30 символів | Місто, до якого буде виконано доставлення |
|  | postal\_code | Рядковий | 10 символів | Поштовий код міста |
|  | street | Рядковий | 60 символів | Вулиця, на яку буде виконано доставлення |

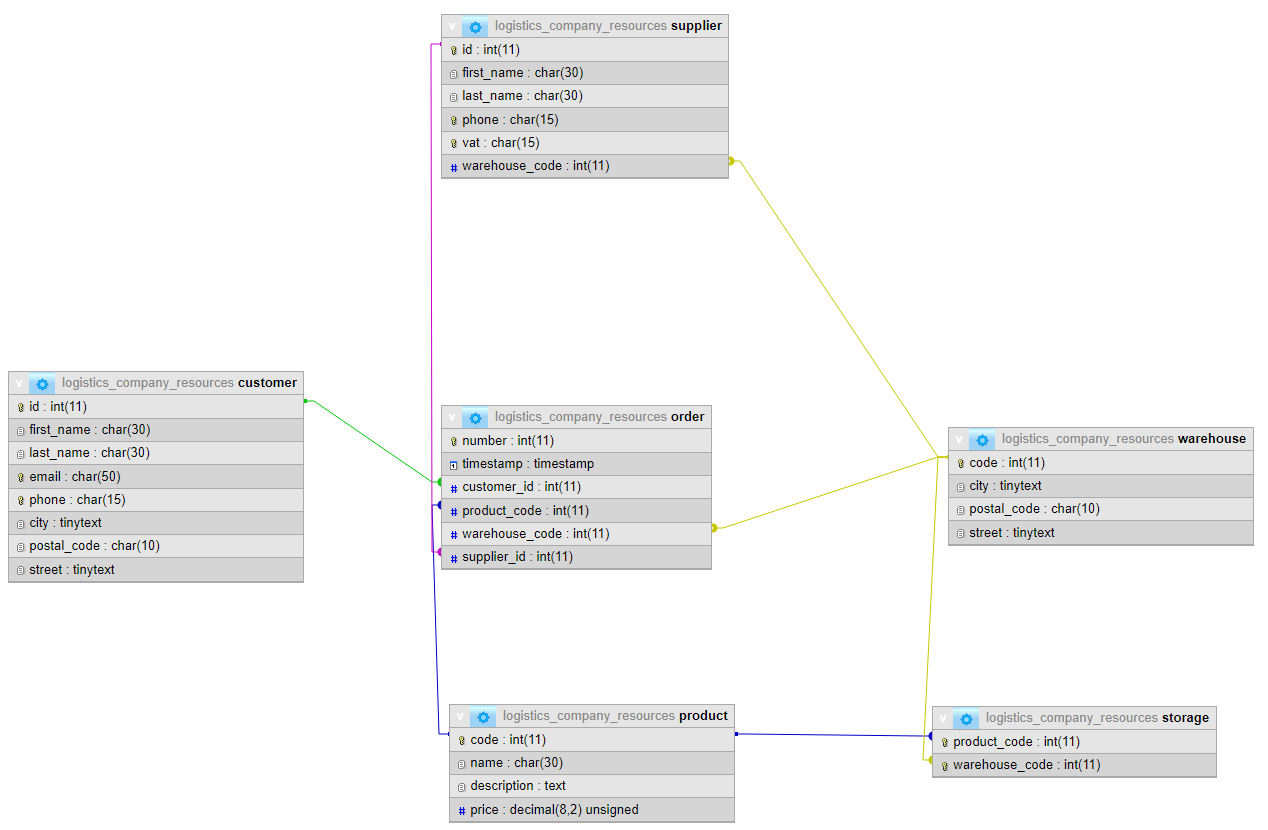
Таблиця «Order» призначена для зберігання інформації про замовлення, які поступають до компанії. Кожний запис таблиці складається з наступних полів, опис яких наведений в таблиці 2.6.

Таблиця 2.6 – Опис структури таблиці «Order»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Ім’я поля | Тип даних | Розмір поля | Опис |
| PK | number | Чисельний | Ціле | Номер замовлення |
|  | timestamp | Позначка часу | Стандартний формат | Дата складання замовлення |
| FK | сustomer\_id | Чисельний | Ціле | Ідентифікатор замовника |
| FK | product\_code | Чисельний | Ціле | Код замовленого товару |
| FK | warehouse\_code | Чисельний | Ціле | Код складу |
| FK | supplier\_id | Чисельний | Ціле | Ідентифікатор постачальника |

## 2.3 Реалізація бази даних у СУБД MySQL

Схема даних для бази даних, що реалізована засобами СУБД MySQL, наведена на рисунку 2.3.

Рисунок 2.3 - Схема розробленої бази даних

Для створення бази даних у СУБД MySQL було використано наступні команди DDL:

CREATE DATABASE `logistics\_company\_resources`

CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

USE `logistics\_company\_resources`;

CREATE TABLE `product` (

`code` INT AUTO\_INCREMENT,

`name` CHAR(30) NOT NULL,

`description` TEXT(100),

`price` DECIMAL(8,2) UNSIGNED NOT NULL,

PRIMARY KEY (`code`)

) ENGINE=INNODB;

CREATE TABLE `warehouse` (

`code` INT AUTO\_INCREMENT,

`city` TEXT(30) NOT NULL,

`postal\_code` CHAR(10) NOT NULL,

`street` TEXT(60) NOT NULL,

PRIMARY KEY (`code`)

) ENGINE=INNODB;

CREATE TABLE `storage` (

`product\_code` INT NOT NULL,

`warehouse\_code` INT NOT NULL,

PRIMARY KEY (`product\_code`, `warehouse\_code`),

FOREIGN KEY (`product\_code`) REFERENCES `product` (`code`), FOREIGN KEY (`warehouse\_code`) REFERENCES `warehouse` (`code`)

) ENGINE=INNODB;

CREATE TABLE `supplier` (

`id` INT AUTO\_INCREMENT,

`first\_name` CHAR(30) NOT NULL,

`last\_name` CHAR(30) NOT NULL,

`phone` CHAR(15) NOT NULL,

`vat` CHAR(15) NOT NULL,

`warehouse\_code` INT NOT NULL,

PRIMARY KEY(`id`),

UNIQUE(`phone`),

FOREIGN KEY (`warehouse\_code`) REFERENCES `warehouse` (`code`)

) ENGINE=INNODB;

CREATE TABLE `customer` (

`id` INT AUTO\_INCREMENT,

`first\_name` CHAR(30) NOT NULL,

`last\_name` CHAR(30) NOT NULL,

`email` CHAR(50) NOT NULL,

`phone` CHAR(15),

`city` TEXT(30) NOT NULL,

`postal\_code` CHAR(10) NOT NULL,

`street` TEXT(60) NOT NULL,

PRIMARY KEY(`id`),

UNIQUE(`email`),

UNIQUE(`phone`)

) ENGINE=INNODB;

CREATE TABLE `order` (

`number` INT AUTO\_INCREMENT,

`timestamp` TIMESTAMP NOT NULL,

`customer\_id` INT NOT NULL,

`product\_code` INT NOT NULL,

`warehouse\_code` INT NOT NULL,

`supplier\_id` INT NOT NULL,

PRIMARY KEY (`number`),

FOREIGN KEY (`customer\_id`) REFERENCES `customer` (`id`),

FOREIGN KEY (`product\_code`) REFERENCES `product` (`code`),

FOREIGN KEY (`warehouse\_code`) REFERENCES `warehouse` (`code`),

FOREIGN KEY (`supplier\_id`) REFERENCES `supplier` (`id`)

) ENGINE=INNODB;

## 2.4 Заповнення бази даних початковими записами

Для коректного відображення текстових значень у форматі Unicode необхідно виконати команду, яка повторює відповідний набір кодування і правила порівняння текстових значень:

SET NAMES utf8mb4 COLLATE utf8mb4\_unicode\_ci;

Також для коректного відображення у командному рядку необхідно виконати команду, яка змінює активне кодування на utf-8:

CHCP 65001

Наступні команди дозволяють заповнити дані про товар у створеній базі даних (рисунок 2.4):

INSERT INTO `product` (`name`, `description`, `price`)

VALUES

('Холодильник Bosch KGN39XI35', 'Двокамерний холодильник з технологією No Frost', 27999.00),

('Прачка LG F4V9BWP2E', 'Безшумна пральна машина зі швидким режимом прання', 18999.50),

('Посудомийна машина Electrolux', 'Посудомийна машина з енергоефективністю класу A++', 16999.99),

('Плита Indesit IS6E5', 'Електрична плита з керамічними конфорками та духовкою', 8499.00),

('Мікрохвильова піч Samsung ME83', 'Мікрохвильова піч з об\'ємом 23 літри та грилем', 2999.75),

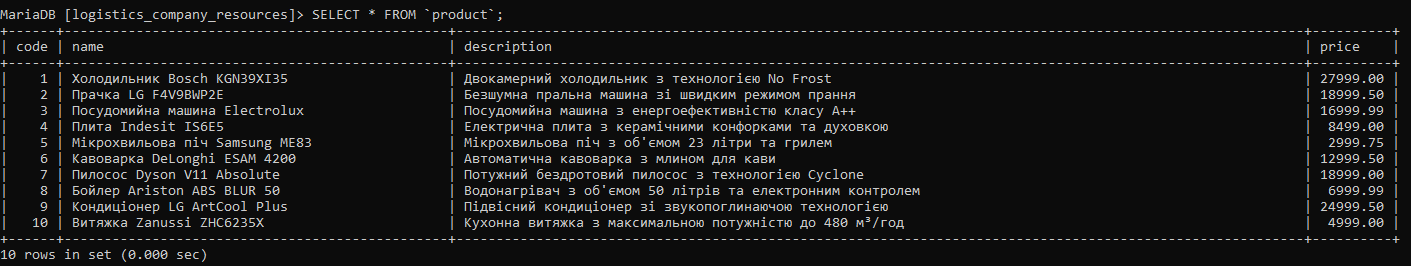
('Кавоварка DeLonghi ESAM 4200', 'Автоматична кавоварка з млином для кави', 12999.50),

('Пилосос Dyson V11 Absolute', 'Потужний бездротовий пилосос з технологією Cyclone', 18999.00),

('Бойлер Ariston ABS BLUR 50', 'Водонагрівач з об\'ємом 50 літрів та електронним контролем', 6999.99),

('Кондиціонер LG ArtCool Plus', 'Підвісний кондиціонер зі звукопоглинаючою технологією', 24999.50),

('Витяжка Zanussi ZHC6235X', 'Кухонна витяжка з максимальною потужністю до 480 м³/год', 4999.00);

Рисунок 2.4 – Записи про товар

Наступні команди дозволяють заповнити дані про склади у створеній базі даних (рисунок 2.5):

INSERT INTO `warehouse` (`city`, `postal\_code`, `street`)

VALUES

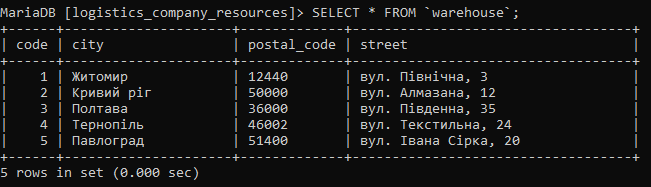
('Житомир', '12440', 'вул. Північна, 3'),

('Кривий ріг', '50000', 'вул. Алмазана, 12'),

('Полтава', '36000', 'вул. Південна, 35'),

('Тернопіль', '46002', 'вул. Текстильна, 24'),

('Павлоград', '51400', 'вул. Івана Сірка, 20');

Рисунок 2.5 – Записи про склади

Наступні команди дозволяють заповнити дані про зберігання товарів на складах у створеній базі даних (рисунок 2.6):

INSERT INTO `storage` (`product\_code`, `warehouse\_code`)

VALUES

(1, 2),

(1, 3),

(3, 1),

(3, 5),

(4, 2),

(4, 4),

(5, 3),

(6, 1),

(6, 5),

(7, 2),

(7, 3),

(7, 4),

(9, 1),

(10, 4);

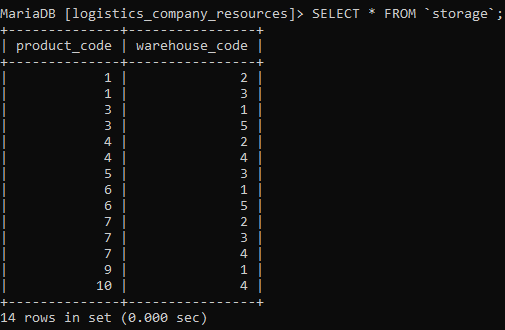


Рисунок 2.6 – Записи про зберігання товарів на складах

Наступні команди дозволяють заповнити дані про постачальників у створеній базі даних (рисунок 2.7):

INSERT INTO `supplier` (`first\_name`, `last\_name`, `phone`, `vat`, `warehouse\_code`)

VALUES

('Олексій', 'Петренко', '+380951234567', '100000000000001', 1),

('Наталія', 'Іванова', '+380978765432', '200000000000002', 1),

('Володимир', 'Сидоренко', '+380935678945', '300000000000003', 1),

('Ольга', 'Коваленко', '+380997654321', '400000000000004', 1),

('Ірина', 'Павленко', '+380963278954', '500000000000005', 2),

('Петро', 'Семененко', '+380950123456', '600000000000006', 2),

('Марина', 'Лисенко', '+380976543219', '700000000000007', 2),

('Сергій', 'Морозов', '+380931234567', '800000000000008', 2),

('Катерина', 'Ковалева', '+380978945612', '900000000000009', 3),

('Андрій', 'Попов', '+380994561237', '100000000000010', 3),

('Іван', 'Мельник', '+380995687412', '110000000000011', 3),

('Олена', 'Ковальчук', '+380977458963', '120000000000012', 3),

('Михайло', 'Бондаренко', '+380936985274', '130000000000013', 4),

('Марія', 'Ковальова', '+380999741852', '140000000000014', 4),

('Віталій', 'Петрів', '+380950632741', '150000000000015', 4),

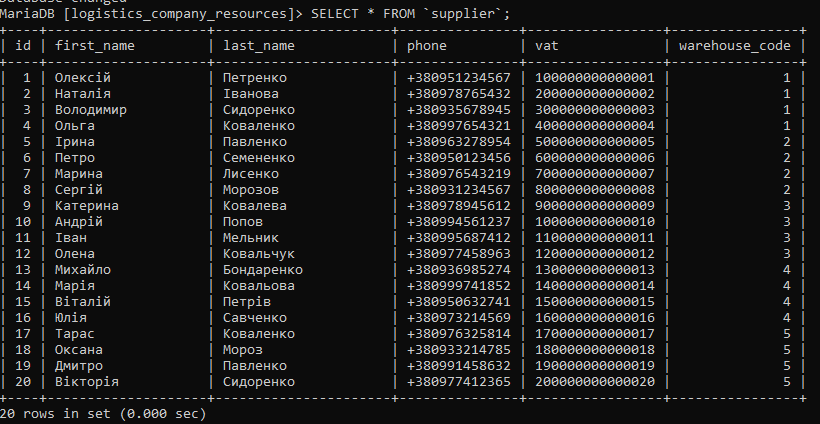
('Юлія', 'Савченко', '+380973214569', '160000000000016', 4),

('Тарас', 'Коваленко', '+380976325814', '170000000000017', 5),

('Оксана', 'Мороз', '+380933214785', '180000000000018', 5),

('Дмитро', 'Павленко', '+380991458632', '190000000000019', 5),

('Вікторія', 'Сидоренко', '+380977412365', '200000000000020', 5);

Рисунок 2.7 – Записи про постачальників

Наступні команди дозволяють заповнити дані про клієнтів у створеній базі даних (рисунок 2.8):

INSERT INTO `customer` (`first\_name`, `last\_name`, `email`, `phone`, `city`, `postal\_code`, `street`)

VALUES

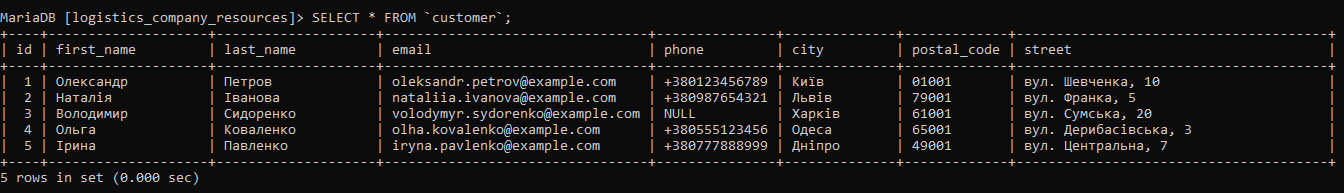
('Олександр', 'Петров', 'oleksandr.petrov@example.com', '+380123456789', 'Київ', '01001', 'вул. Шевченка, 10'),

('Наталія', 'Іванова', 'nataliia.ivanova@example.com', '+380987654321', 'Львів', '79001', 'вул. Франка, 5'),

('Володимир', 'Сидоренко', 'volodymyr.sydorenko@example.com', NULL, 'Харків', '61001', 'вул. Сумська, 20'),

('Ольга', 'Коваленко', 'olha.kovalenko@example.com', '+380555123456', 'Одеса', '65001', 'вул. Дерибасівська, 3'),

('Ірина', 'Павленко', 'iryna.pavlenko@example.com', '+380777888999', 'Дніпро', '49001', 'вул. Центральна, 7'),

Рисунок 2.8– Записи про клієнтів

Наступні команди дозволяють заповнити дані про замовлення у створеній базі даних (рисунок 2.9):

INSERT INTO `order` (`timestamp`, `customer\_id`, `product\_code`, `warehouse\_code`, `supplier\_id`)

VALUES

('2024-05-20 10:45:00', 1, 1, 1, 4),

('2024-05-21 11:15:30', 1, 10, 2, 7),

('2024-05-22 12:25:10', 2, 2, 3, 11),

('2024-05-23 13:50:00', 2, 9, 4, 15),

('2024-05-24 15:00:40', 3, 3, 1, 2),

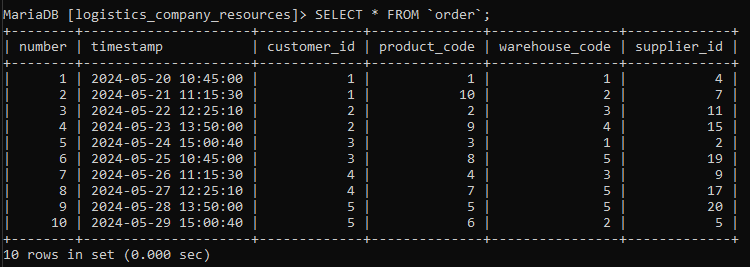
('2024-05-25 10:45:00', 3, 8, 5, 19),

('2024-05-26 11:15:30', 4, 4, 3, 9),

('2024-05-27 12:25:10', 4, 7, 5, 17),

('2024-05-28 13:50:00', 5, 5, 5, 20),

('2024-05-29 15:00:40', 5, 6, 2, 5);

Рисунок 2.9 – Записи про замовлення

## 2.5 Створення “Уявлень” у СУБД MySQL

**US 01: Перегляд списку складів разом з товарами**

CREATE VIEW `warehouse\_product` AS

SELECT `warehouse`.code AS `warehouse\_code`,

CONCAT(`warehouse`.`street`, ' ,', `warehouse`.`city`,

' ,', `warehouse`.`postal\_code`) AS `warehouse\_address`,

`product`.`name` AS `product\_name`,

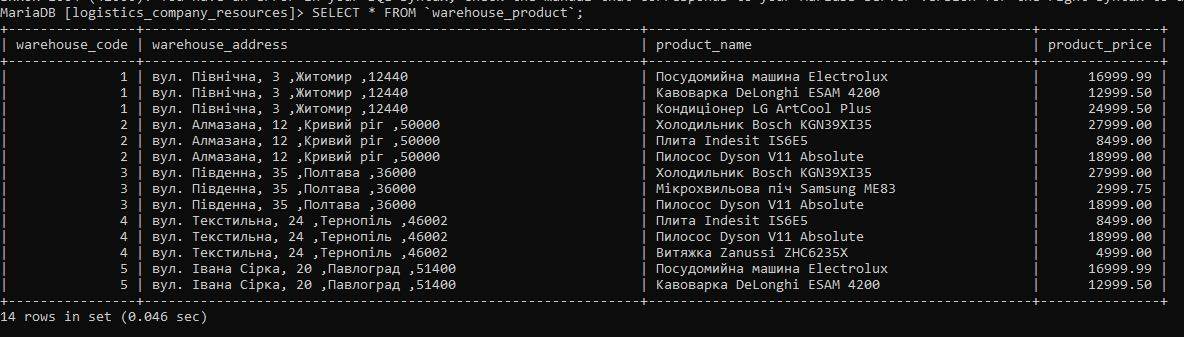
`product`.`price` AS `product\_price`

FROM `warehouse`

INNER JOIN `storage` ON `warehouse`.code = `storage`.`warehouse\_code`

INNER JOIN `product` ON `storage`.`product\_code` = `product`.code

ORDER BY warehouse\_products\_count(`warehouse`.`code`) DESC;

Рисунок 2.10 - Список складів разом з товарами

**US 02: Перегляд списку замовників разом з замовленнями**

CREATE VIEW `customer\_order` AS

SELECT `customer`.`id`,

CONCAT(`customer`.`first\_name`, ' ', `customer`.`last\_name`) AS `customer\_name`,

CONCAT(`customer`.`street`, ' ,', `customer`.`city`,

' ,', `customer`.`postal\_code`) AS `customer\_address`,

`order`.`number` AS `order\_number`,

`product`.`name` AS `product\_name`,

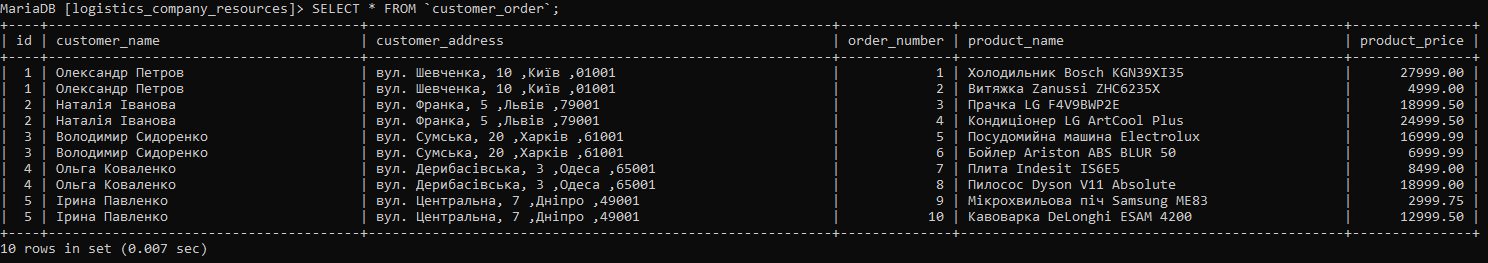
`product`.`price` AS `product\_price`

FROM `customer`

INNER JOIN `order` ON `customer`.`id` = `order`.`customer\_id`

INNER JOIN `product` ON `order`.`product\_code` = `product`.code

ORDER BY customer\_orders\_count(`customer`.`id`) DESC;

Рисунок 2.11 - Список замовників разом з замовленнями

**US 03: Перегляд списку постачальників разом з складами**

CREATE VIEW `supplier\_warehouse` AS

SELECT `supplier`.`id` AS `supplier\_id`,

CONCAT(`supplier`.`first\_name`, ' ', `supplier`.`last\_name`) AS `supplier\_name`,

`supplier`.`vat` AS `supplier\_vat`,

`warehouse`.code AS `warehouse\_code`,

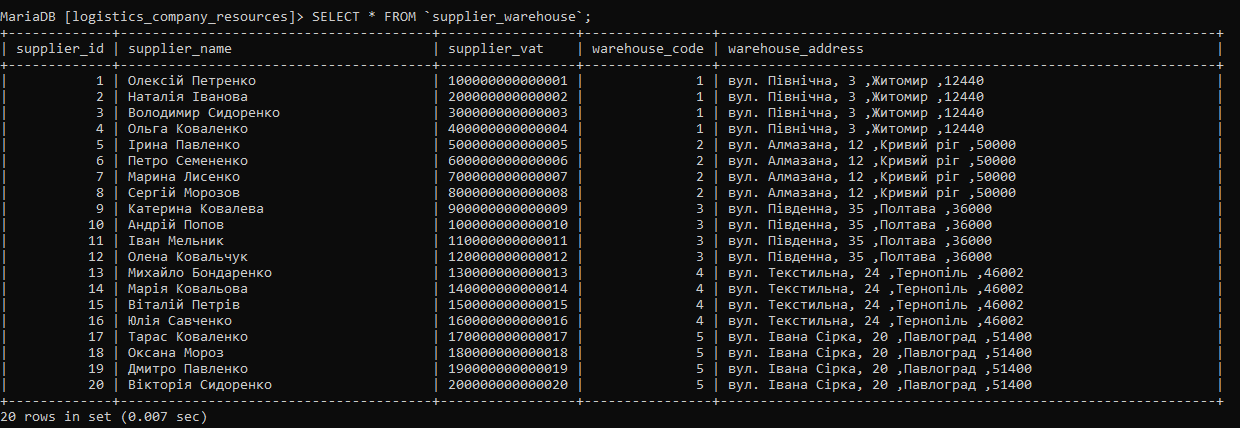
CONCAT(`warehouse`.`street`, ' ,', `warehouse`.`city`,

' ,', `warehouse`.`postal\_code`) AS `warehouse\_address`

FROM `supplier`

INNER JOIN `warehouse` ON `supplier`.`warehouse\_code` = `warehouse`.`code`

ORDER BY `supplier`.`id`;

Рисунок 2.12 - Список постачальників разом з складами

## 2.6 Створення “Збережених Функцій” у СУБД MySQL

**US 04 Перегляд кількості товарів певного складу**

DELIMITER $$

CREATE FUNCTION warehouse\_products\_count (`w\_code` INT)

RETURNS INT

NOT DETERMINISTIC

BEGIN

DECLARE `count` INT;

SELECT COUNT(\*)

INTO `count`

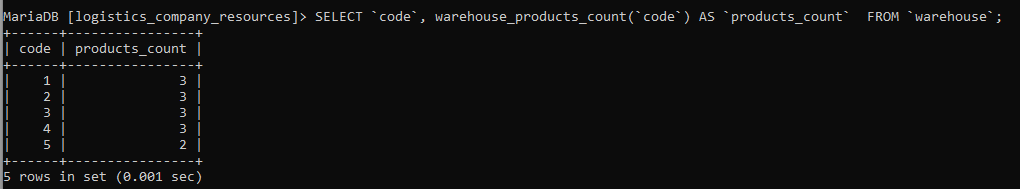
FROM `storage`

WHERE `warehouse\_code` = `w\_code`;

RETURN `count`;

END$$

DELIMITER ;

Рисунок 2.13 - Кількість товарів на складах

**US 05 Перегляд кількості доставок постачальника:**

DELIMITER $$

CREATE FUNCTION supplier\_deliveries\_count (`s\_id` INT)

RETURNS INT

NOT DETERMINISTIC

BEGIN

DECLARE `count` INT;

SELECT COUNT(`order`.`number`)

INTO `count`

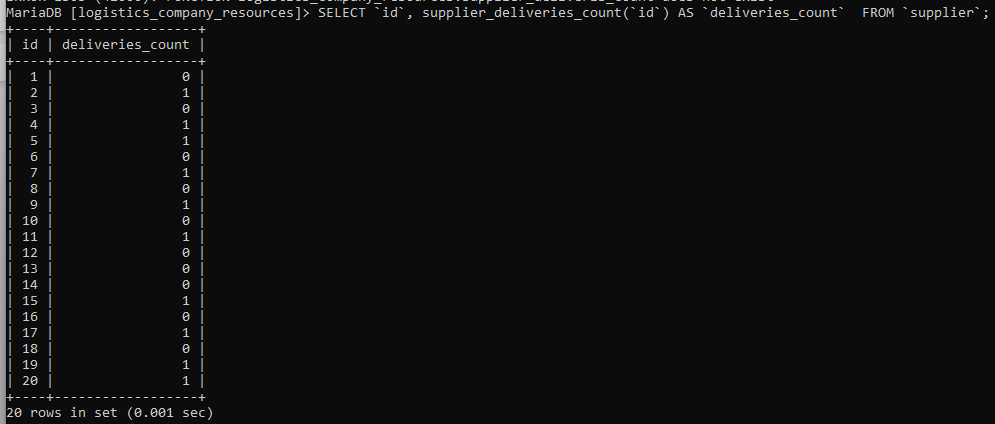
FROM `order`

WHERE `supplier\_id` = `s\_id`;

RETURN `count`;

END$$

DELIMITER ;

Рисунок 2.14 - Кількість доставок постачальників

**US 06 Перегляд кількості замовлень замовника:**

DELIMITER $$

CREATE FUNCTION customer\_orders\_count (`c\_id` INT)

RETURNS INT

NOT DETERMINISTIC

BEGIN

DECLARE `count` INT;

SELECT COUNT(`order`.`number`)

INTO `count`

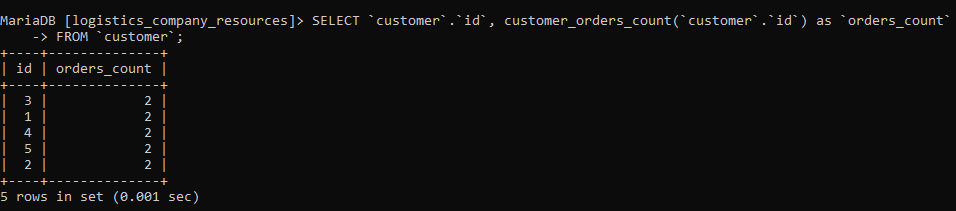
FROM `order`

WHERE `customer\_id` = `c\_id`;

RETURN `count`;

END$$

DELIMITER ;

Рисунок 2.15 - Кількість замовлень користувачів

## 2.7 Створення "Збережених Процедур” у СУБД MySQL

**FR 01 Управління товарами**

DELIMITER $$

CREATE PROCEDURE create\_product (

IN `p\_name` CHAR(30),

IN `p\_description` TEXT(100),

IN `p\_price` DECIMAL(8,2)

)

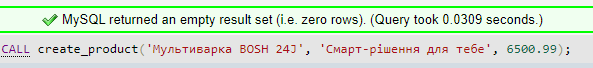
BEGIN

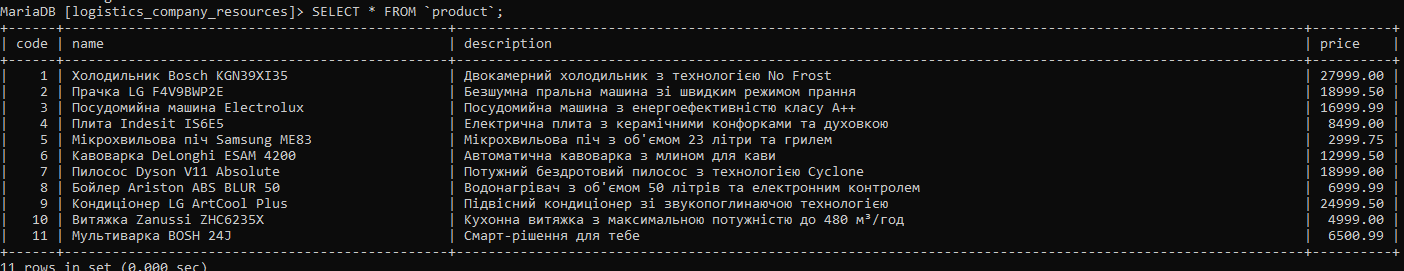
INSERT INTO `product` (`name`, `description`, `price`)

VALUES (`p\_name`, `p\_description`, `p\_price`);

END $$

DELIMITER ;



Рисунок 2.16 - Створення товару

**FR 02 Управління складами**

DELIMITER $$

CREATE PROCEDURE update\_warehouse (

IN w\_code INT,

IN w\_city TEXT(30),

IN w\_postal\_code CHAR(10),

IN w\_street TEXT(60)

)

BEGIN

UPDATE `warehouse`

SET `city` = w\_city,

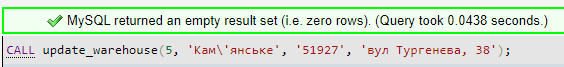
`postal\_code` = w\_postal\_code,

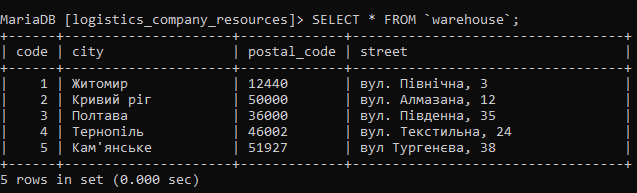
`street` = w\_street

WHERE `code` = w\_code;

END $$

DELIMITER ;



Рисунок 2.17 - Оновлення складу

**FR 03 Управління постачальниками**

DELIMITER $$

CREATE PROCEDURE delete\_supplier (IN s\_id INT)

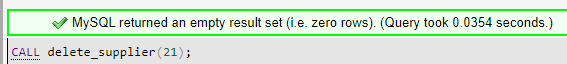
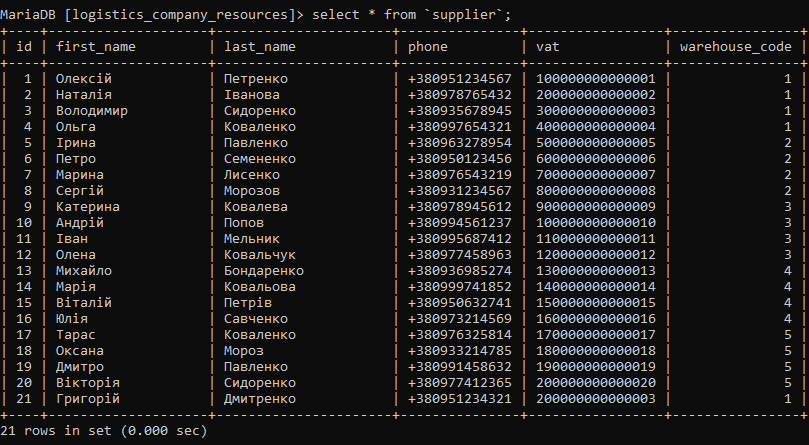
BEGIN

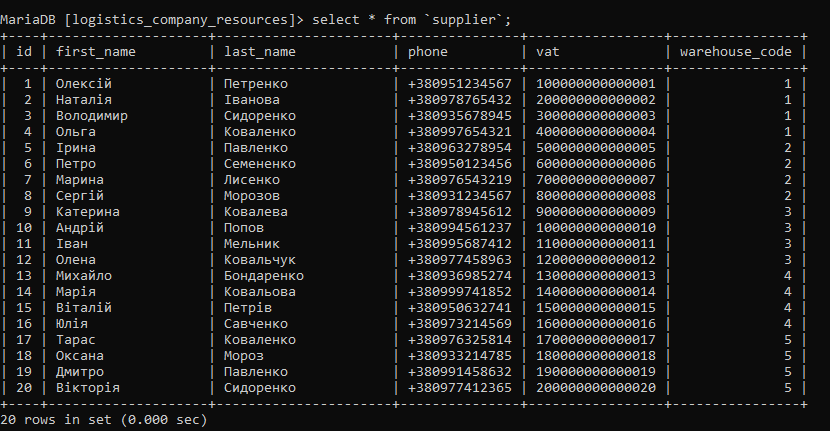
DELETE FROM `supplier`

WHERE `id` = s\_id;

END $$

DELIMITER ;



Рисунок 2.18 - Видалення постачальника

**FR 04 Управління замовниками**

DELIMITER $$

CREATE PROCEDURE read\_customer (IN c\_id INT)

BEGIN

SELECT `id`, CONCAT(`first\_name`, ' ', `last\_name`) AS `name`, `email`, `phone`,

CONCAT(`street`, ' ,', `city`, ' ,', `postal\_code`) AS `address`,

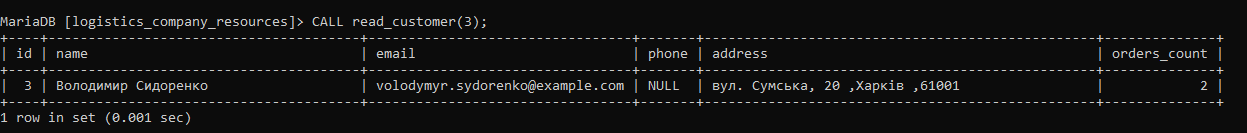
customer\_orders\_count(`id`) AS `orders\_count`

FROM `customer`

WHERE `id` = c\_id;

END $$

DELIMITER ;

Рисунок 2.19 - Читання замовника

## 2.8 Створення “Тригерів” у СУБД MySQL

**US 07 Додавання замовлення:**

DELIMITER $$

CREATE TRIGGER after\_order\_insert

AFTER INSERT ON `order`

FOR EACH ROW

BEGIN

DELETE FROM `storage`

WHERE `product\_code` = NEW.`product\_code`

AND `warehouse\_code` = NEW.`warehouse\_code`;

END $$

DELIMITER ;



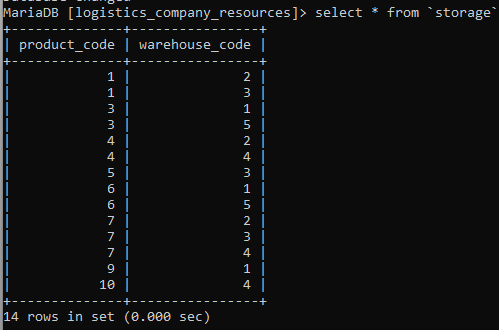


Рисунок 2.20 - Додавання замовлення

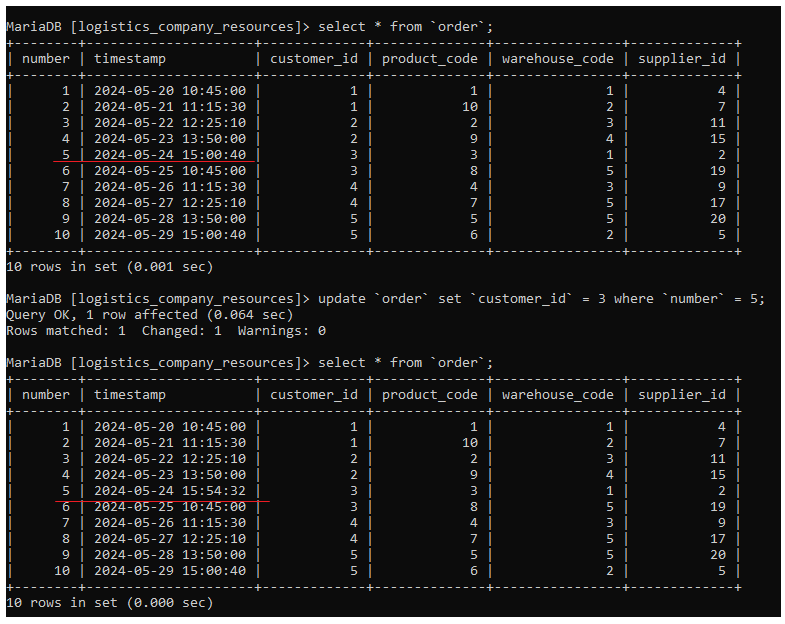
**US 08 Зміна замовлення:**

CREATE TRIGGER order\_timestamp\_update

BEFORE UPDATE ON `order`

FOR EACH ROW

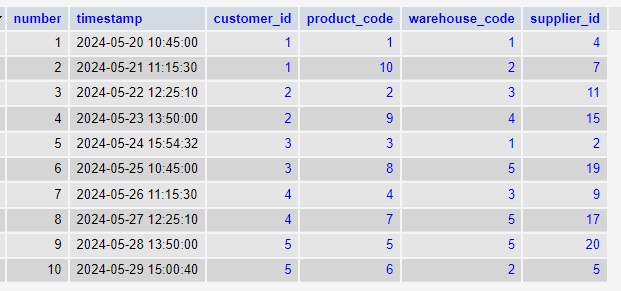
SET NEW.`timestamp` = CURRENT\_TIMESTAMP();

Рисунок 2.21 - Оновлення дати замовлення

## 2.9 Створення “Транзакцій” у СУБД MySQL

1) Транзакція для створення користувача з замовленням

Рисунок 2.22 - Таблиця користувачів до виконання транзакції

Рисунок 2.23 - Таблиця замовлень до виконання транзакції

SET AUTOCOMMIT = 0;

START TRANSACTION;

SELECT @customer\_id := MAX(`id`) + 1

FROM `customer`;

INSERT INTO `customer` (`first\_name`, `last\_name`, `email`,

`phone`, `city`, `postal\_code`, `street`)

VALUES ('Дмитро', 'Петров', '[dmitro.petrov@example.com](mailto:dmitro.petrov@example.com)',

'+380123456788', 'Харків', '61002', 'вул. Шевченка, 10');

SELECT \* FROM `customer`;

INSERT INTO `order` (`timestamp`, `customer\_id`, `product\_code`,

`warehouse\_code`, `supplier\_id`)

VALUES ('2024-06-1 15:35:30', @customer\_id, 6, 2, 6);

SELECT \* FROM `order`;

COMMIT;

SET AUTOCOMMIT = 1;

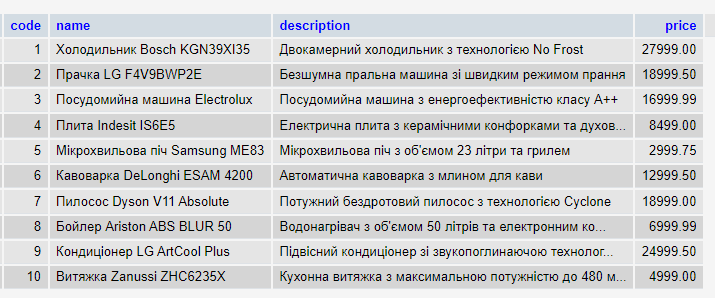
SELECT \* FROM `customer`;

SELECT \* FROM `order`;

Рисунок 2.24 - Таблиця користувачів підчас та після виконання транзакції

Рисунок 2.25 - Таблиця замовлень підчас та після виконання транзакції

2) Транзакція для створення продуктів з складом

Рисунок 2.26 - Таблиця товарів до виконання транзакції

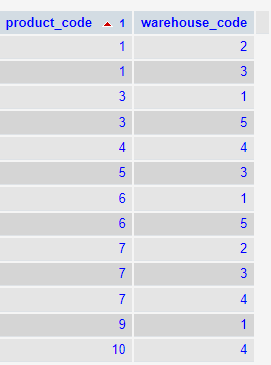


Рисунок 2.27 - Таблиця збереження до виконання транзакції

SET AUTOCOMMIT = 0;

START TRANSACTION;

SELECT @product\_code := MAX(`code`) + 1

FROM `product`;

INSERT INTO `product` (`name`, `description`, `price`)

VALUES ('Мікрохвильова піч BOSH 34K', 'З турботою до родини', 3299.99);

SELECT \* FROM `product`;

INSERT INTO `storage` (`product\_code`, `warehouse\_code`)

VALUES (@product\_code, 3);

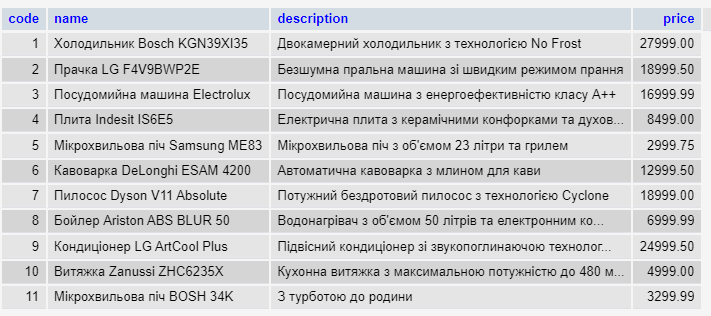
SELECT \* FROM `storage`;

ROLLBACK;

SET AUTOCOMMIT = 1;

SELECT \* FROM `product`;

SELECT \* FROM `storage`;

Рисунок 2.28 - Таблиця товарів підчас виконання транзакції

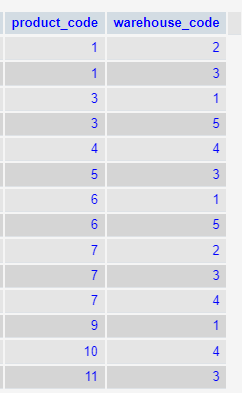
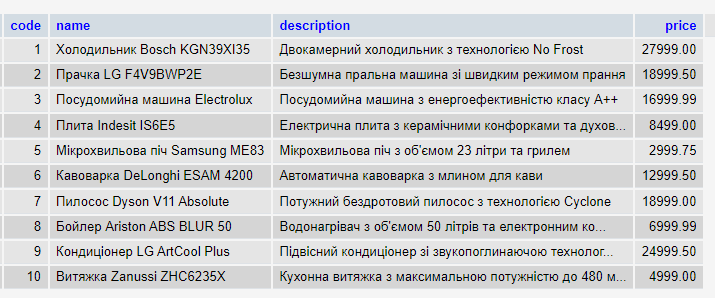


Рисунок 2.29 - Таблиця збереження підчас виконання транзакції

Рисунок 2.30 - Таблиця товарів після виконання транзакції

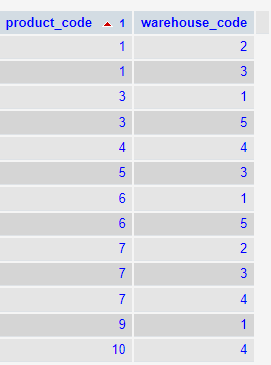


Рисунок 2.31 - Таблиця збереження після виконання транзакції

3) Транзакція для створення cкладу з постачальниками



Рисунок 2.32 - Таблиця складів до виконання транзакції

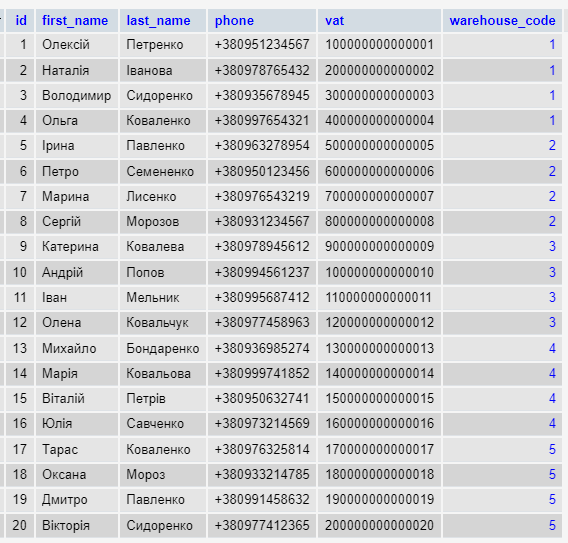


Рисунок 2.33 - Таблиця постачальників до виконання транзакції

SET AUTOCOMMIT = 0;

START TRANSACTION;

SELECT @warehouse\_code := MAX(`code`) + 1

FROM `warehouse`;

INSERT INTO `warehouse` (`city`, `postal\_code`, `street`)

VALUES ('Павлоград', '51400', 'вул. Івана Сірка, 20');

SELECT \* FROM `warehouse`;

UPDATE `supplier`

SET `warehouse\_code` = @warehouse\_code

WHERE `id` = 20;

SELECT \* FROM `supplier`;

COMMIT;

SET AUTOCOMMIT = 1;

SELECT \* FROM `warehouse`;

SELECT \* FROM `supplier`;



Рисунок 2.34 - Таблиця складів підчас та після виконання транзакції



Рисунок 2.35 - Таблиця замовників підчас та після виконання транзакції

## 2.10 Створення “Користувачів” у СУБД MySQL

Для виконання завдання, було окремо сформовано акторів, які взаємодіють з системою:

* Employee - Працівник компанії, який володіє інформацією ресурси компанії та управляє ними.
* Strategist - Стратег компанії, який відповідає за розвиток компанії. Використовує ресурси для аналізу діяльності і CRM.

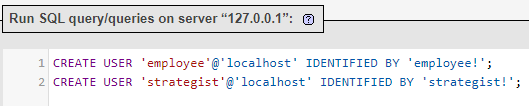


Рисунок 2.36 - Створення облікових записів

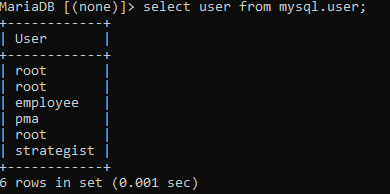


Рисунок 2.37 - Результат створення облікових записів

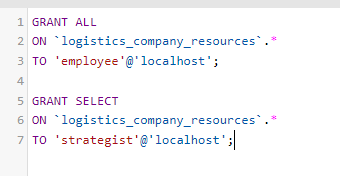
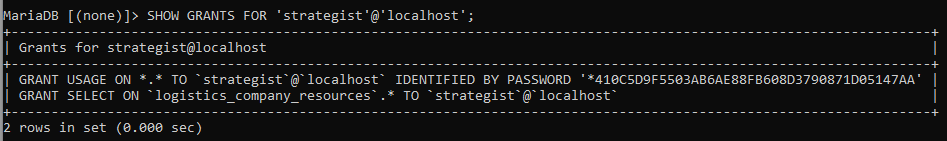
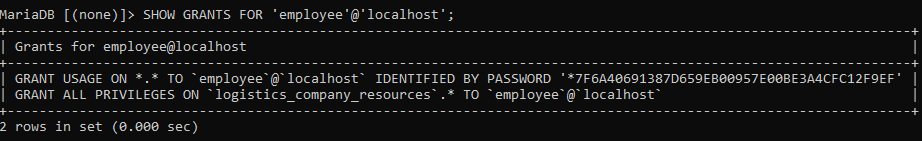


Рисунок 2.38 - Призначення необхідних повноважень

Рисунок 2.39 - Результат призначення необхідних повноважень

**2.11 Модифікація механізмів забезпечення цілісності посилань**

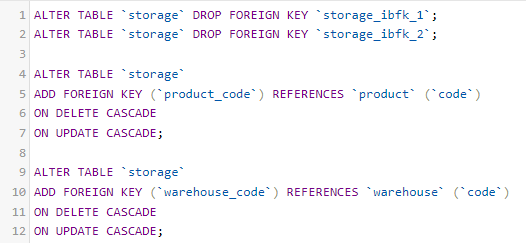
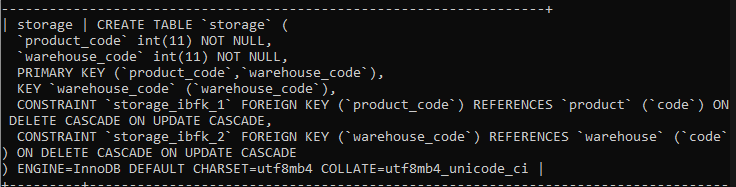


Рисунок 2.40 - Модифікація механізмів забезпечення цілісності посилань для таблиці `storage`

Рисунок 2.41 - Результат модифікації механізмів забезпечення цілісності посилань для таблиці `storage`

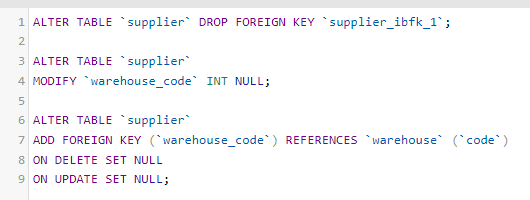
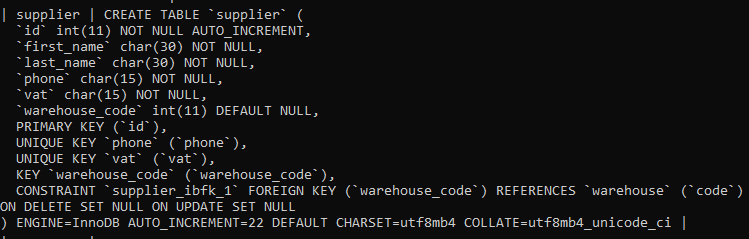


Рисунок 2.42 - Модифікація механізмів забезпечення цілісності посилань для таблиці `supplier`

Рисунок 2.43 - Результат модифікації механізмів забезпечення цілісності посилань для таблиці `storage`

# 3 ПРОЄКТУВАННЯ ТА РОЗРОБКА ЗАСТОСУНКУ ДЛЯ РОБОТИ З БАЗОЮ ДАНИХ

## Проектування застосунку для роботи з базою даних

**3.1 Діяльність програмного забезпечення**

Діаграма діяльності, яка зображає потік дій під час додавання нового замовлення до системи, зображена на рисунку 3.1

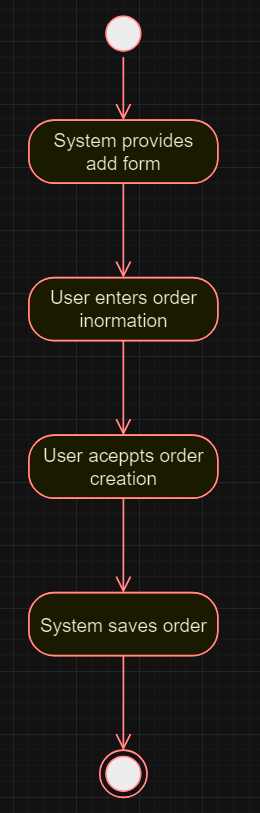
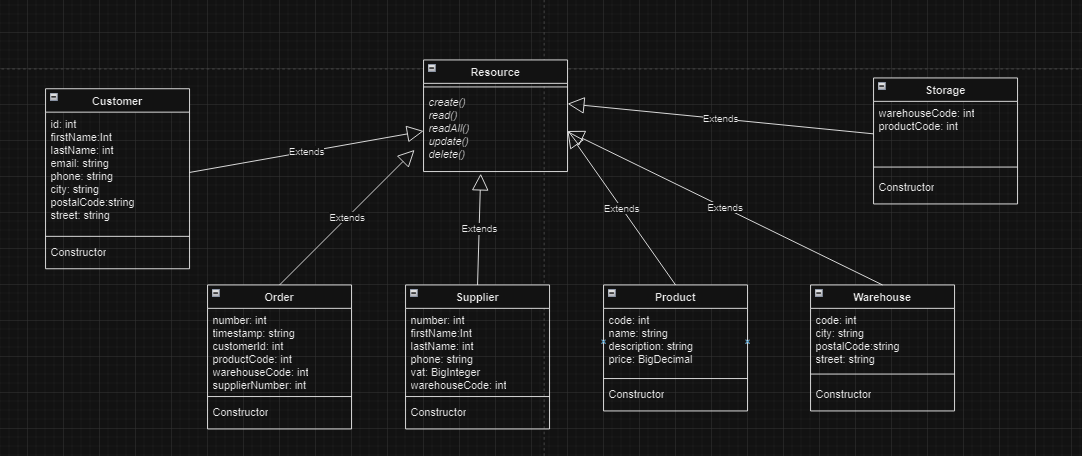


Рисунок 3.1 - Діаграма діяльності для додавання нового замовлення

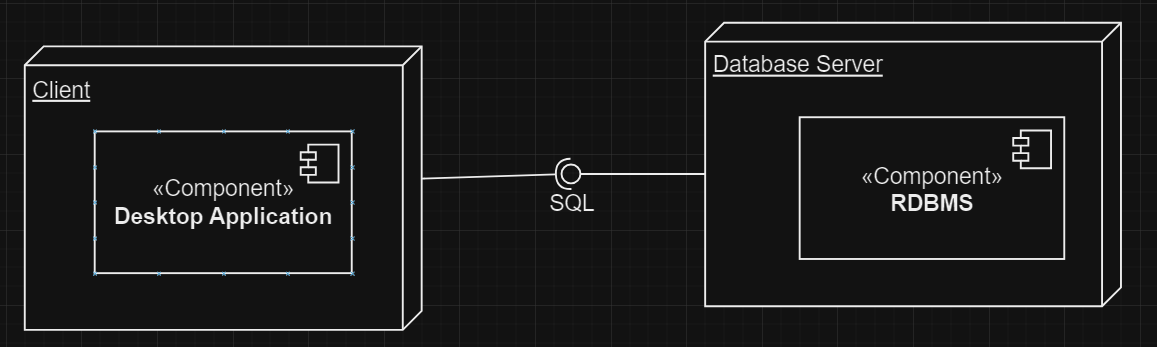
**3.2 Об’єктно-орієнтована модель програмного забезпечення**

Діаграма класів, яка відображає сутності предметної області, зображена на рисунку 3.2

Рисунок 3.2 - Діаграма класів

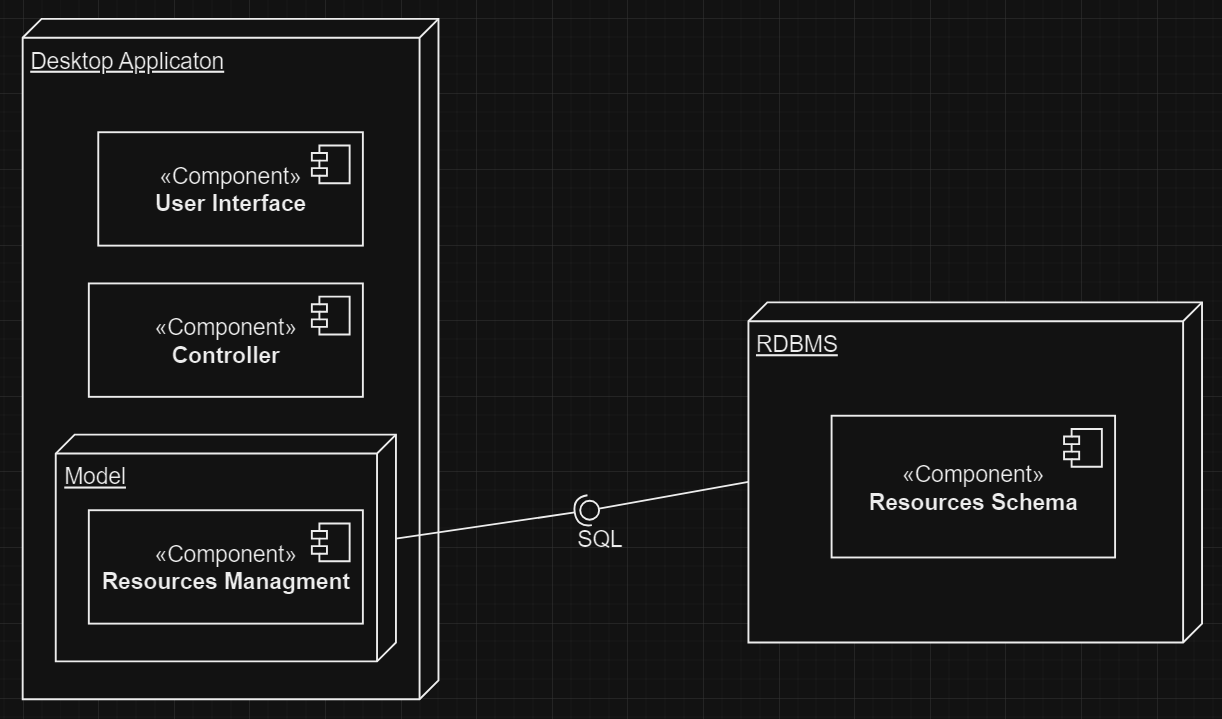
**3.3 Архітектура програмного забезпечення**

Діаграма розгортання застосунку для роботи з базою даних наведена на рисунку 3.3.

 Рисунок 3.3 - Діаграма розгортання застосунку

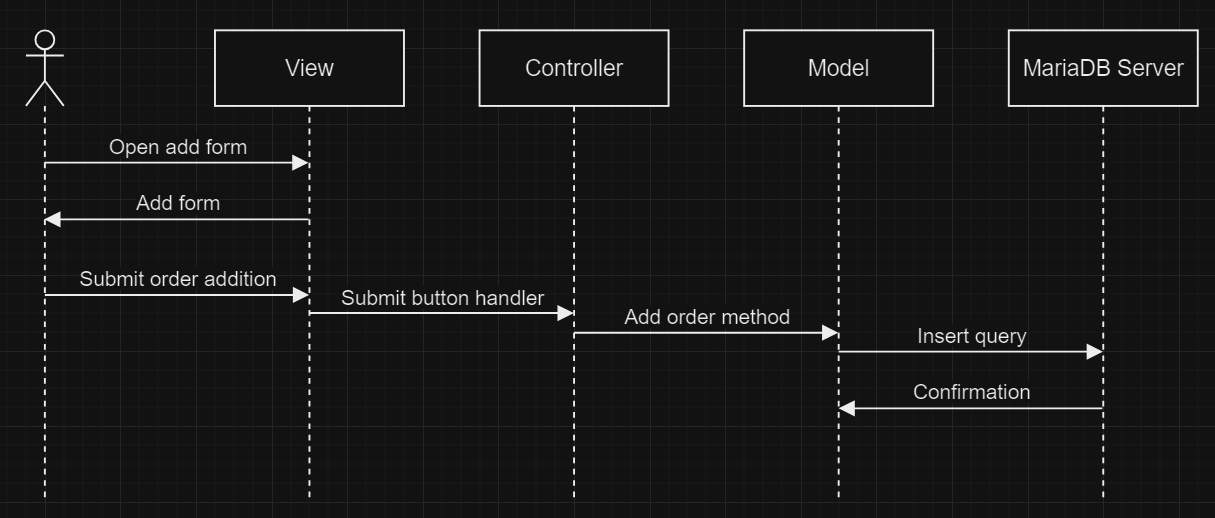
**3.4 Компоненти програмного забезпечення**

Діаграма компонентів, яка зображує детальну структуру програмного забезпечення, зображена на рисунку 3.4

Рисунок 3.4 - Діаграма компонентів

**3. 5 Поведінка програмного забезпечення**

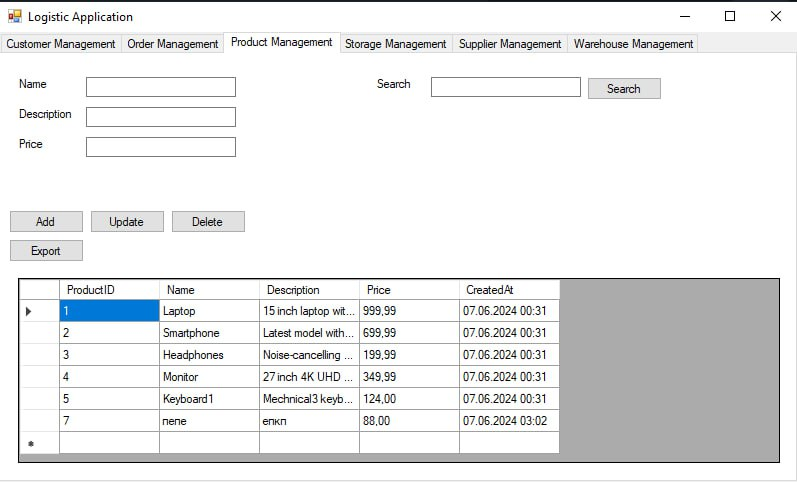
Діаграма послідовності, яка зображає взаємодію користувачем і компонентами системи під час додавання нового замовлення до системи, зображена на рисунку 3.5

Рисунок 3.5 - Діаграма послідовності

## Приклад використання застосунку для роботи з базою даних

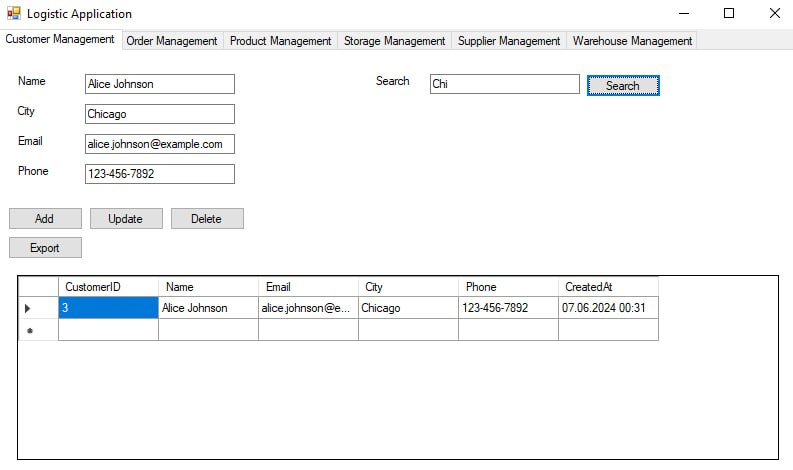
**Інтерфейс програмного забезпечення**

Приклад головної сторінки графічного інтерфейсу користувача, для взаємодії з базою даних, зображено на рисунку 3.6.

Рисунок 3.6 - Інтерфейс ПЗ

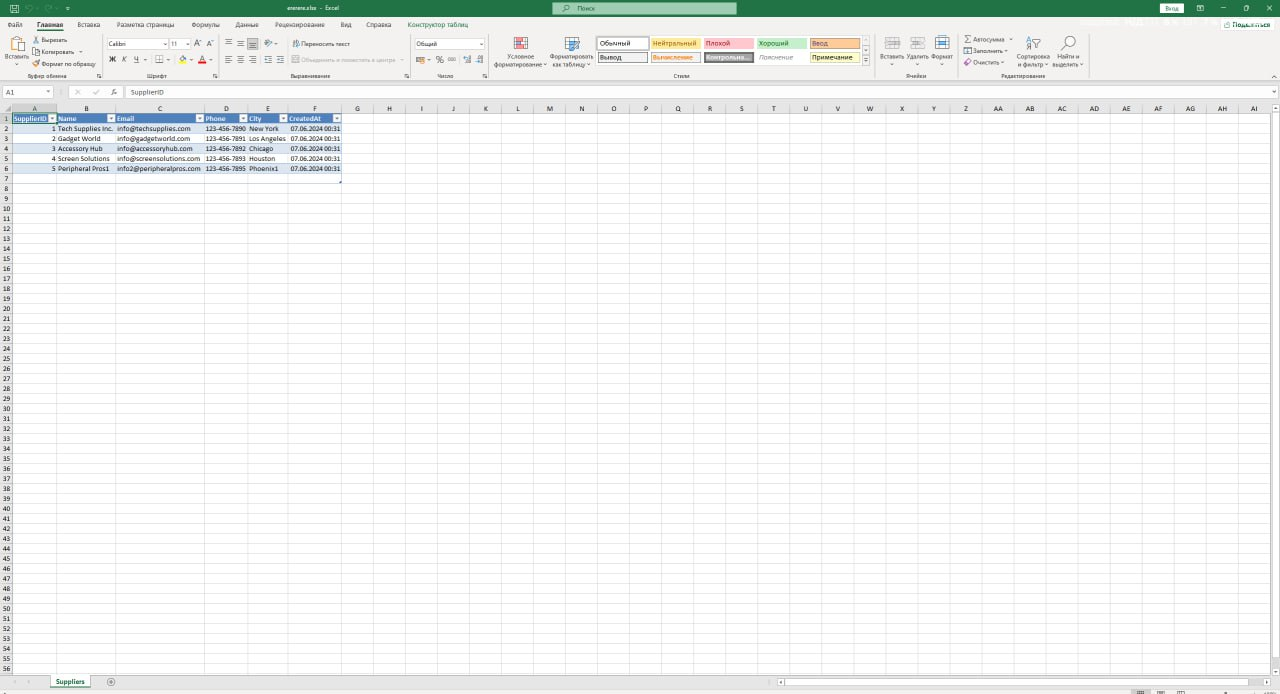
**Пошук ресурсу**

Приклад пошуку ресурсу зображено на рисунку 3.7.

Рисунок 3.7 - Пошук ресурсу

**Експорт ресурсу**

Приклад експорту ресурсу до .csv файлу зображено на рисунку 3.8.

Рисунок 3.8 - Експорт ресурсу

# ВИСНОВКИ

Для виконання проєкту було розглянуто діяльність логістичної компанії, завдання якої полягає у зберіганні, обробці і доставці товару. Компанія володіє великим обсягом інформації, тому наше IT-рішення було розроблено для підвищення продуктивності обробки інформації шляхом створення БД, яка враховує особливості визначених бізнес-правил та застосунку для взаємодії з БД.

Під час життєвого циклу проєкту було проаналізовано діяльність компанії та виконано наступні кроки:

1 Розроблено логічну модель бази даних.

2 Розроблено фізичну модель бази даних.

3 Описано структуру бази даних.

4 Реалізовано базу даних у системі управління базами даних (СУБД) MySQL за допомогою команд DDL (Data Definition Language).

5 Заповнено базу даних початковими записами та розроблено запити мовою SQL (Structured Query Language).

6 Запроєктовано застосунок для роботи з базою даних.

7 Розроблено застосунок для роботи з базою даних.

# СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1 Object-Oriented Design in Java by Stephen Gilbert and Bill McCarty

2 Computer Networking: A Top-Down Approach 6th Edition by Jim Kurose and Keith Ross

3 Software Requirements (Developer Best Practicies) 3rd Edition by Karl Wigers and Joy Beatty

4 Database Design for Mere Mortals 4th Edition by Michael J Hernandez

5 [Advanced Java Programming Course by Leo V. Ivanov](http://iwanoff.inf.ua/java_en_2/LabTraining01.html)