

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
Ім. Ігоря Сікорського»
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

КУРСОВА РОБОТА

з дисципліни
«Організація баз даних і знань»

на тему: «Проектування та створення реляційної бази даних для інтернет-магазину товарів різної категорії із інтеграцією з інструментом звітності Report Builder на базі платформи Microsoft SQL Server Reporting Services (SSRS)»

Студентки 3 курсу групи КА-23
Галузь знань *12 Інформаційні технології*
Спеціальність *124 Системний аналіз*
Кабанова Марина Ігорівна
Керівник: доцент, к. т. н. Зінченко А. Ю.
Національна оцінка _____
Кількість балів: _____
Оцінка ECTS: _____

Засвідчую, що у цій курсовій роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студентка Кабанова Марина Ігорівна

НТУУ „КПІ ім. І. Сікорського” НН ІПСА
(назва вищого закладу освіти)

Кафедра	<i>Математичних методів системного аналізу</i>
---------	--

Дисципліна	<i>Організація баз даних та знань</i>
------------	---------------------------------------

Галузь знань	<i>124 Системний аналіз</i>
-----------------	-----------------------------

Курс	<i>3</i>	Група	<i>КА—23</i>	Семестр	<i>другий</i>
------	----------	-------	--------------	---------	---------------

ЗАВДАННЯ
на курсовий проект(роботу) студента

Кабанова Марина Ігорівна
(прізвище, ім'я, по батькові)

1.Тема проекту(роботи)	Тема №11. Проектування та створення реляційної бази даних для інтернет-магазину товарів різної категорії із інтеграцію з інструментом звітності Report Builder на базі платформи Microsoft SQL Server Reporting Services (SSRS).

2. Строк здачі студентом закінченого проекту(роботи)	20.05.2025 р.
--	---------------

3. Вихідні дані до проекту (роботи)	<p>Спроектувати та розробити реляційну базу даних для автоматизації основних бізнес-процесів предметної області інтернет-магазинів. Структура бази даних повинна відображати основні сутності магазину, такі як замовник, постачальник, склад, товар, замовлення, система доставки та платежу.</p> <p>В базі повинна бути передбачена можливість зберігання інформації про користувачів системи. В розробленій базі даних передбачити логування помилок та подій змін (створення, видалення, зміна) об'єктів, зокрема дату зміни. Запити для пошуку за постачальником, типом товару, номером та статусом замовлення. Перевірка коректного введення рахунку, адреси користувачів, контактної інформації; та інші операції доставки, оформлення замовлення та здійснення оплати товару; відстеження кількості товарів у магазині зі складом постачання повинні бути реалізовані окремими процедурами чи функціями в пакеті.</p> <p>Крім того, потрібно передбачити пакет із SQL-запитами для перевірки коректності функціонування бази даних в цілому. Продемонструвати логіку роботи обраної предметної області інтернет-магазину.</p> <p>На створені таблиці потрібно передбачити власну журнальну таблицю, послідовність, тригер, пакет із процедурами/функціями відбору даних, оновлення, вставлення та видалення.</p>
-------------------------------------	--

	Забезпечити функціонал для внутрішнього/зовнішнього використання, безпеки доступу, подань, звітів і бізнес-аналітики.
--	---

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)	
1. Титульний аркуш.	
2. Листок завдання з описом постановки задачі.	
3. Діаграми потоків DFD0, DFD1 та DFD3.	
4. Життєві цикли основних сутностей (блок-схеми).	
5. ERD для інфологічної моделі.	
6. ERD для даталогічної моделі у третій нормальній формі.	
7. Опис вибору СУБД та БД.	
8. Опис основних алгоритмів (за наявності), складових БД.	
9. Результати роботи створених об'єктів БД.	
10. Висновки.	
11. Список літератури.	
12. Додаток А. Лістинг реалізації структури БД і заповнення таблиць даними;	
13. Додаток Б. Лістинг скриптів до створення об'єктів БД (схеми, тригерів, індексів, представлень, збережених процедур, користувацьких функцій тощо).	

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

1. Діаграми потоків DFD0, DFD1 та DFD3;	
2. Життєві цикли основних сутностей (блок-схеми);	
3. ERD для інфологічної моделі;	
4. ERD для даталогічної моделі у третій нормальній формі;	
5. Загальні блок-схеми алгоритмів (за наявності).	
6. Ілюстрації роботи скриптів до створених об'єктів БД.	

6. Дата видачі завдання	25.02.2025
-------------------------	------------

КАЛЕНДАРНИЙ ПЛАН

/п	Назва етапів курсового проекту (роботи)	Кінцевий строк виконання етапів роботи	Примітка
.	Вибір теми курсової роботи. Оформлення листа Завдання. Формулювання вимог.	25.02.2025	
.	Аналіз постановки задачі. Підбор та вивчення літератури.	04.03.2025	
.	Проведення функціонального моделювання бізнес-процесів предметної області з використанням методології DFD (Data Flow Diagrams).	11.03.2025	
.	Проведення інфологічного та даталогічного моделювання баз даних (побудова ERD). Перше узгодження з керівником.	18.03.2025	
.	Реалізація структури БД в реляційній СУБД згідно з фізичною ER-діаграмою. Заповнення таблиць даними. Друге узгодження з керівником.	25.03.2025	
.	Створення об'єктів бази даних.	15.04.2025	

.	Тестування скриптів.	29.04.2025	
.	Оформлення пояснювальної записки.	13.05.2025	
.	Подання курсового проекту (роботи) на перевірку.	20.05.2025	
0.	Захист та демонстрація курсового проекту (роботи).	26.05.2025 – 28.05.2025 – «А» 02.06.2025 – «В, С», 03.06.2025 – 07.06.2025 – «D, E»	

Студент		Кабанова Марина Ігорівна
	(підпис)	

Керівник		Зінченко Артем Юрійович
	(підпис)	(прізвище, ім'я, по батькові)
	(дата)	

ЗМІСТ

ВСТУП	1
1. РОЗДІЛ 1. ПОСТАНОВКА ЗАВДАННЯ	2
1.1. Основні теоретичні відомості.	3
1.2. Вибір програмного забезпечення для проектування та створення бази даних	7
2. РОЗДІЛ 2. ПРОЕКТУВАННЯ СТРУКТУРИ БАЗИ ДАНИХ	10
2.1. Функціональне моделювання.	10
2.2. Інфологічна модель предметної області.	12
2.3. Даталогічна модель предметної області.	18
3. РОЗДІЛ 3. РОЗРОБЛЕННЯ БАЗИ ДАНИХ	21
3.1. Створення бази даних.	21
3.2. Створення та забезпечення посилкової цілісності таблиць.	22
3.3. Створення подань.	26
3.4. Створення тригерів.	30
3.5. Створення функцій.	34
3.6. Створення курсорів.....	35
3.7. Створення процедур.	37
3.8. Налаштування безпеки доступу до об'єктів бази даних.	41
3.8.1. Призначення ролей на рівні бази даних (Database-level Roles).....	42
3.8.2. Створення програмних ролей (Application Roles).....	44
4. РОЗДІЛ 4. ПРАКТИЧНЕ ЗАСТОСУВАННЯ БАЗИ ДАНИХ	50
4.1. Налаштування SQL Server Reporting Services (SSRS).....	52
4.2. Налаштування безпеки доступу та розмежування прав у (SSRS).....	54
4.3. Створення звітів на основі бази даних.....	56
ВИСНОВКИ	65
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	67
Додаток А. Лістинг реалізації структури БД і заповнення таблиць даними;	68
Додаток Б. Лістинг скриптів до створення об'єктів БД (схеми, тригерів, індексів, представлень, збережених процедур, користувацьких функцій)	80

ВСТУП

Предметною областю курсової роботи є розробка та проектування реляційної бази даних для онлайн-продажу товарів різної категорії із реалізацією системи звітності на базі веб-технологій.

Запропонована база даних дозволяє структурувати дані та забезпечує зручний доступ до введення та виведення інформації, зокрема: особисті дані клієнтів, історію замовлень, каталог товарів, дані постачальників, статуси та рахунки платежів тощо. Важливим аспектом проєкту є побудова якісної системи звітності на основі запитів бази даних, яка дозволяє надавати користувачам інформацію про діяльність магазину в зручній візуальній формі.

Для реалізації роботи були використані наступні інформаційні інструменти: програмний продукт draw.io для проектування діаграм та блок-схем; інтегроване середовище для роботи з базою даних SQL Server Management Studio 20 (SSMS); стандартна мова запитів SQL; SQL Server Reporting Services (SSRS) для створення аналітичних звітів на базі SQL-запитів. Для підготовки та оформлення курсової роботи використаний текстовий редактор Microsoft Word.

Курсова робота складається зі вступу, основної частини, висновку, списку використаних джерел та додатків. У першому розділі розглянута постановка задачі, включно з теоретичними відомостями та обґрунтуванням обраного програмного забезпечення для проектування бази даних. Другий розділ містить спроектовані діаграми потоків даних (DFD0, DFD1, DFD2), інфологічну та даталогічну моделі у вигляді ER-діаграм для цільової предметної області. У третьому розділі реалізовано логіку функціонування бази даних, створено об'єкти бази, подання, процедури, функції, тригери та налаштована система безпеки доступу. Четвертий розділ містить опис реалізації аналітичних звітів.

1. ПОСТАНОВКА ЗАВДАННЯ

Метою роботи є розробка та реалізація реляційної бази даних для автоматизації ключових процесів інтернет-магазину та побудова системи звітності для аналізу даних.

Інформаційна система створюється з цілями полегшення процесу різного вигляду документації та відстеження бізнес-процесів, подій, пов'язаних із продажом, управлінням замовленнями та обліком фінансових операцій. Реалізація за рахунок інтуїтивно зрозумілого механізму введення, редагування та швидкого пошуку необхідних даних.

Система повинна надавати зрозумілий механізм введення, редагування, фільтрації та пошуку даних, а також процедури для формування аналітичної звітності для прийняття рішень.

Основні функції системи:

- введення, збереження та редагування контактної інформації користувачів із фіксацією дати змін у базі даних;
- документація усіх операцій з товарами, замовленнями, доставкою;
- фіксація платежів, статусів замовлення;
- реалізація механізму автоматичного оновлення статусів після внесення змін в інший з них;
- синхронізація кількості товарів магазину та складу, з автоматичним оновленням даних;
- реалізація необхідних перевірок для ефективного здійснення замовлень: наявності товарів на складі, необхідних даних клієнтів для доставки та здійснення покупок, перевірка коректності введених даних;
- гнучке сортування та пошук товарів за ціною, брендом, категорією;
- автоматизація процесу доставки товарів, видачу рахунків та оновлення статусів замовлень з урахуванням актуальної дати операцій;
- створення збережених процедур, функцій, курсорів і тригерів для оптимізації обробки даних;

- забезпечення цілісності даних, захист від втрати інформації та стабільну роботу системи при подальших оновленнях;
- реалізація подань і звітів, що дозволяють візуалізувати результати аналізу та приймати обґрунтовані управлінські рішення;
- налаштування безпеки доступу до об'єктів бази даних за допомогою розмежування прав внутрішніх користувачів та ролей для зовнішнього використання.

Для досягнення поставленого завдання необхідно виконати наступні етапи: розглянути теоретичну частину, проаналізувати предметну область, побудувати інформаційно-логічну модель бази даних, розробити модель з урахуванням характеристик обраної системи управління базами даних. При розробці визначити необхідні вимоги, функціонал, обґрунтувати створенні зв'язки, обмеження та алгоритми. Заповнити отримані таблиці даними та реалізувати політику доступу та систему безпеки, розробити необхідні запити та процедури. Провести тестову перевірку працездатності та інтегрувати з системою звітності.

1.1. Основні теоретичні відомості.

Для коректного визначення вимог для реалізації бази даних необхідно визначити основні поняття.

База даних (БД) – це множина взаємопов'язаних даних, об'єднаних спільним середовищем зберігання, спільним застосуванням, єдиною формою представлення, єдиним и методами і засобами керування[1].

При наявності готової бази даних можна здійснити у ній пошук даних по заданих користувачем критеріях. Розглядають кілька аспектів моделювання в обробці даних: [5]

- системний аналіз предметної області.
- концептуальне проектування;
- логічне проектування;

- фізичне проектування.

Системний аналіз передбачає мовний опис реальних об'єктів предметної області, визначення зв'язків між об'єктами, дослідження характеристик об'єктів і зв'язків.

На етапі концептуального проектування визначаються об'єкти, зв'язки між об'єктами, атрибути, ключові атрибути.

Логічне проектування полягає в створенні логічної моделі на основі вибраної моделі даних.

Фізичні моделі визначають засоби розміщення даних в середовищі зберігання і засоби доступу до цих даних, які підтримуються на фізичному рівні.

Система управління базами даних (СУБД) – сукупність мовних та програмних засобів, призначених для створення, ведення і сумісного використання бази даних багатьма користувачами.

Реляційна модель (RM) – це набір простих таблиць, між якими встановлені зв'язки (відношення) з допомогою числових кодів. Елементи реляційної моделі [5]:

- відношення – таблиці, де рядки представляють записи, а стовпці представляють атрибути;
- кортеж – рядок таблиці, який містить один запис;
- атрибут – заголовок стовпця таблиці;
- ключ – сукупність атрибутів, які унікально визначають кожен рядок таблиці, виконують функції зв'язування таблиць, дозволяють прискорити операції над таблицями;
- домен – множина значень атрибуту;
- схема відношення – рядок заголовків стовпців таблиці, тобто назва відношення з його атрибутами.

Для однозначної ідентифікації рядків, для зв'язування таблиць між собою, та для прискорення операцій над даними застосовують ключі:

- потенційний ключ (Potential Key) – мінімальна підмножина атрибутів відношення, які єдиним чином ідентифікують кортеж даного відношення;
- первинний ключ (Primary Key) - потенційний ключ, який обрано для унікальної ідентифікації кортежів відношення;
- вторинний ключ (Secondary Key) – ключ, кожному значенню якого може відповідати більш ніж один екземпляр індексованих даних;
- зовнішній ключ (Foreign Key) – сукупність атрибутів відношення, значення яких є одночасно і значеннями первинного або потенційного ключа іншого відношення.

Переваги моделі реляційної бази даних [3]:

- однорідність подання даних у моделі, що обумовлює простоту сприйняття її конструкцій користувачами БД;
- наявність розвиненої математичної теорії реляційних БД, що обумовлює коректність її застосування.

На вхід процесу проектування БД подаються[3]:

- інформаційна модель : діаграми "сутність зв'язок" (ER-діаграми);
- функціональна модель: бізнес-модель процесів, діаграми потоку даних (DF-діаграми), діаграми станів, діаграми життєвих циклів сутностей, специфікації на системи (вимоги), бізнес-правила;
- загальносистемні вимоги й обмеження;
- завдання зворотного впливу

База даних має задовольняти наступним вимогам:

- цілісність, що визначає повноту і правильність інформації, яка вміщується в БД;

- відновлювання;
- ефективність, що характеризує мінімальний час реакції на запит користувача і мінімальні потреби в пам'яті;
- безпека, що передбачає захист даних від сторонньої модифікації.
- задовільнення потребам організації та вимогам користувачів;
- адаптація при зміні програмного й апаратного середовища
- можливість подальшого практичного застосування бази даних для візуалізації аналітики, бізнес-логіки і надання звітності;

Дозволи для управління базами даних керуються на рівні сервера (server level) за допомогою логінів і ролей сервера, та на рівні бази даних (database level) — за допомогою користувачів бази даних і ролей бази даних.

Принципал безпеки (security principal) — це ідентифікатор, який використовує SQL Server, якому можна призначити дозвіл на виконання дій. Принципали безпеки зазвичай є людьми або групами людей, які можна створювати та керувати за допомогою SQL Server Management Studio.

Логіни (logins) — це окремі облікові записи користувачів для входу в обробник баз даних SQL Server. SQL Server і база даних SQL підтримують входи на основі автентифікації Windows і входи на основі автентифікації SQL Server.

У SQL Server **фіксовані ролі сервера (fixed server roles)** — це набір попередньо налаштованих ролей, які надають зручну групу дозволів на рівні сервера. Логіни можна додавати до ролей за допомогою оператора *ALTER SERVER ROLE ... ADD MEMBER*. База даних SQL не підтримує фіксовані ролі сервера, але має дві ролі в *master* базі даних (dbmanager і loginmanager), які діють як ролі сервера.

Аналогічно, SQL Server надає можливість створення власних ролей серверу (**user-defined server roles**) та надання їм дозволу на рівні сервера. Логіни можна додавати до ролей за допомогою оператора *ALTER SERVER ROLE ... ADD MEMBER*.

Для легкого керування дозволами на рівні бази даних, у SQL Server використовується поняття учасників (*principals*), які можуть запитувати ресурси SQL Server, і яким можуть надаватися дозволи на використання таких ресурсів.

Щоб додати або видалити користувачів до ролі бази даних, використовується параметри *ADD MEMBER* і *DROP MEMBER* оператора *ALTER ROLE*.

Виділяють два типи ролей на рівні бази даних: фіксовані ролі бази даних (**fixed database roles**), які попередньо визначені для кожної бази даних, і ролі бази даних, визначені користувачем (**user-defined database roles**). Дозволи ролей, визначених користувачем, можна налаштувати за допомогою операторів *GRANT*, *DENY* і *REVOKE*.

Кожен користувач бази даних належить до **public** ролі бази даних, членство в якій змінити не можна. Якщо користувачеві не надано або заборонено певні дозволи на захищений об'єкт, користувач успадковує дозволи, надані **public** для цього об'єкта.

Крім цього, в SQL Server існують ролі програми (**application roles**) — це учасники бази даних, які дозволяють програмі працювати з власними, подібними до користувача дозволами. Ролі програми вмикаються за допомогою *sp_setapprole*, для якої потрібен пароль. Оскільки ролі програми є учасники на рівні бази даних, вони можуть отримати доступ до інших баз даних лише через дозволи, надані в цих базах даних гостю (*guest*). Таким чином, будь-яка база даних, у якій було вимкнено гостьову систему, недоступна для ролей програми в інших базах даних. У SQL Server ролі програми не можуть отримати доступ до метаданих рівня сервера (*server-level metadata*), оскільки вони не пов'язані з учасником рівня сервера [7].

1.2. Вибір програмного забезпечення для проектування та створення бази даних.

Для реалізації бази даних була обрана мова програмування Structured Query Language (SQL) та управління базами даних Microsoft SQL Server у середовищі SQL Server Management Studio 20 (SSMS).

Мова SQL є загальноприйнятим стандартом для роботи з реляційними базами даних, що забезпечує виконання операцій зі збереження, редагування, видалення та аналізу даних.

Microsoft SQL Server – це потужна система управління базами даних (СУБД), розроблена корпорацією Microsoft. Призначена для зберігання та обробки великих обсягів даних.

Основні переваги програми[6]:

- Масштабованість – система може працювати як на персональних комп'ютерах, так і на високопродуктивних серверах із великим навантаженням.
- Оптимізація зберігання даних – використання сторінок розміром до 8 КБ забезпечує швидке отримання інформації та ефективне управління складними наборами даних.
- Розвинений механізм транзакцій – підтримка багаторівневого блокування даних дозволяє обробляти транзакції в режимі реального часу.
- Гнучкий пошук – можливість здійснювати пошук за ключовими словами, фразами або текстовими фрагментами із застосуванням індексів.
- Інтеграція з іншими продуктами Microsoft – забезпечує взаємодію з Excel, Access, Power BI та іншими бізнес-інструментами.

Для демонстрації практичного застосування створеної бази даних було використано інструмент звітування на базі Microsoft SQL Server Reporting Services (SSRS). SSRS дозволяє користувачам створювати інтерактивні, табличні, графічні звіти або звіти у довільній формі з джерел реляційних, багатовимірних даних або даних на основі XML. Система звітності реалізована

на базі сучасного веб-порталу – односторінкова програма HTML5, яка працює з усіма основними браузерами, включаючи Microsoft Edge, Internet Explorer 10 і 11, Chrome, Firefox і Safari.

Вміст порталу звітів SSRS впорядковується за такими типами об'єктів:

- Пагіновані звіти (сторінково-орієнтовані документи для друку);
- KPI (ключові показники ефективності);
- Робочі книги Excel;
- Спільні набори даних;
- Спільні джерела даних;

Основні переваги використання SSRS[8]:

- Бездоганна інтеграція з іншими інструментами та програмами Microsoft, зокрема з пакетом SQL Server. Це робить його привабливим вибором для організацій, які вже використовують технології Microsoft, забезпечуючи плавний і ефективний процес звітування даних.
- Динамічні та інтерактивні звіти: можливість створювати різноманітні звіти, включаючи табличні, графічні звіти та звіти у довільній формі, використовуючи низку джерел даних.
- Централізоване керування звітами: SSRS дозволяє централізовано керувати звітами, що спрощує для адміністраторів контроль доступу та забезпечення безпеки даних.
- Налаштування форматування звітів: включає можливість експорту звітів у різні формати, такі як PDF, Excel і Word.

2. ПРОЕКТУВАННЯ СТРУКТУРИ БАЗИ ДАНИХ

2.1. Функціональне моделювання.

Діаграма потоків даних або DFD (Data Flow Diagram) – це методологія графічного структурного аналізу, що описує зовнішні стосовно системи джерела та адресати даних, логічні функції, потоки даних та сховища даних, до яких здійснюється доступ[2].

Головна ціль DFD – продемонструвати, як кожен процес перетворює вхідні дані і вихідні, а також виявити відношення між цими процесами. Різниця між DFD0, DFD1 та DFD2 полягає в рівні деталізації, який вони представляють у моделюванні бізнес-процесів за допомогою діаграм потоків даних (DFD) [4].

На основі аналізу предметної області інтернет-магазину було побудовано діаграми потоків даних.

1. Діаграма DFD0 представляє загальний огляд інформаційної системи продажу товарів, показуючи взаємодію системи із зовнішніми сутностями (акторами) та потоки даних між ними. Основні зовнішні сутності:

- Замовник – взаємодіє з системою, здійснюючи пошук товарів, оформлення замовлення та оплату.
- Постачальник – постачає продукцію та отримує реквізити замовлень.
- Склад – відповідає за збереження продукції.



Рис.2.1. Контекстна діаграма DFD0 процесу продажу товарів.

2. Діаграма DFD1 деталізує процеси, що складають систему, а також відповідні потоки даних між ними та зовнішніми сутностями. Основні функціональні блоки:

- Здійснення замовлення.
- Підтвердження платежу
- Доставка продукції.



Рис.2.2. Діаграма DFD1 процесу продажу товару.

3. Діаграма DFD2 деталізує підпроцеси, показані на DFD1, додаючи потоки та сховища даних:

- Обробка замовлення – перевірка наявності товару, збір деталей замовлення;
- Оплата замовлення – реєстрація та оновлення статусу платежу;
- Підтвердження платежу – перевірка оплати та надсилання рахунку;
- Доставка продукції – доставка товару та оновлення статусу замовлення.

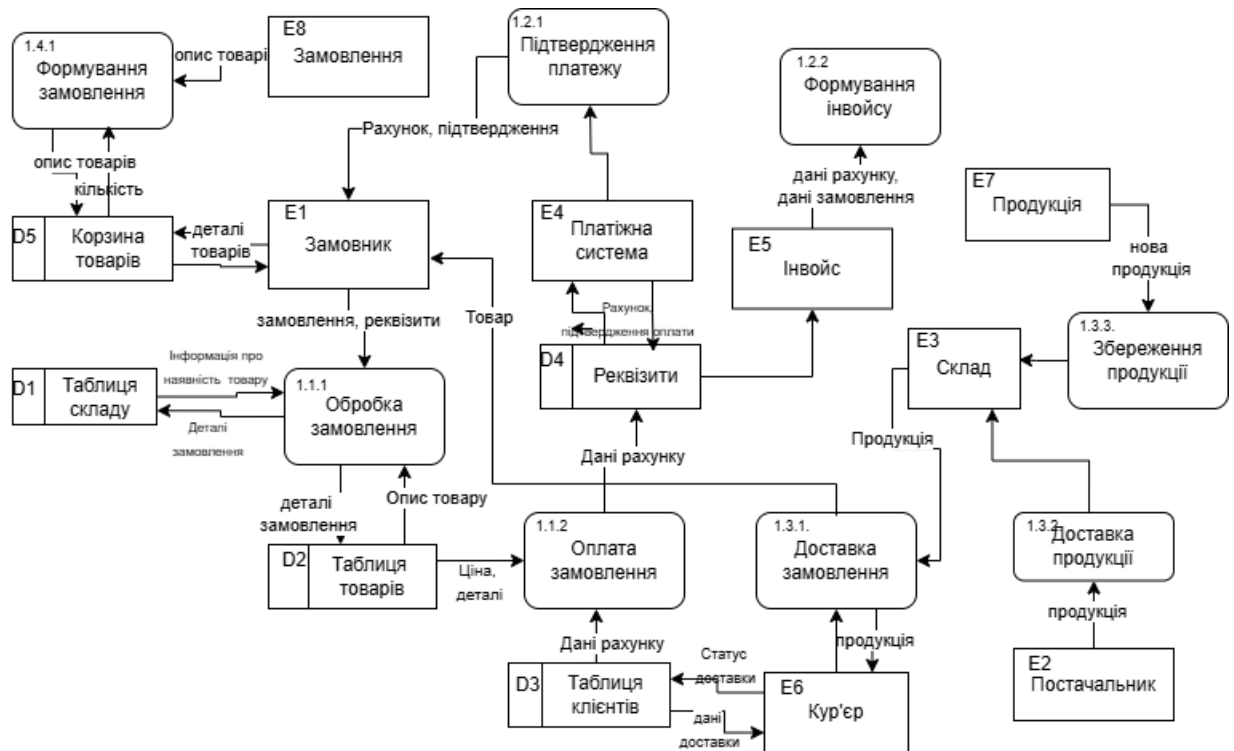


Рис.2.3. Діаграма DFD2 процесу продажу товару.

2.2. Інфологічна модель предметної області.

Опис предметної області із зображенням взаємозв'язків між основними об'єктами та даними, виконаний без орієнтації на використовувані надалі програмні і технічні засоби, називається інфологічною моделлю предметної області. Найбільш поширеною є модель "сутність – зв'язок" (Entity Relationship model, ER-модель)[4].

Розробка ER-моделі складається з наступних етапів:

- визначення сутностей;
- визначення зв'язків між сутностями;
- визначення атрибутів кожної сутності;
- визначення первинних (унікальних) ключів;
- визначення ступеня зв'язку між сутностями;
- визначення класу приналежності сутностей.

Для вирішення поставленого завдання визначено наступні сутності та їхні зв'язки: Товар (Product), Замовлення (Order), Замовник (Client), Постачальник

(Supplier), Доставка (Delivery), Склад (Warehouse), Платіжна система (PaymentSys), Рахунок (Invoices), Користувач (Users).

- Постачальник (Supplier) володіє Складом (Warehouse);
- Товар (Product) зберігається на Складі (Warehouse);
- Постачальник (Supplier) продає Товар (Product);
- Замовник (Client) робить Замовлення (Order);
- Замовлення (Order) містить один або кілька Товарів (Product);
- Доставка (Delivery) отримує інформацію про Замовлення (Order);
- Платіжна система (PaymentSys) обробляє оплату Замовлення (Order);
- Замовник (Client) здійснює оплату через Платіжну систему (PaymentSys);
- Платіжна система (PaymentSys) створює Рахунок (Invoices);
- Користувач (Users) вносить зміни у систему.

Встановлені ступені зв'язків між сутностями, які вказують яким чином сутності взаємодіють між собою.

- 1) Бінарний зв'язок один до одного (1:1) – кожному представнику сутності *A* відповідає 1 або 0 представників сутності *B*.

Зв'язок мають наступні сутності:

- Замовлення (Order) із Доставка (Delivery);
- Замовлення (Order) із Платіжною системою (PaymentSys);
- Замовник (Client) із Платіжною системою (PaymentSys).

- 2) Бінарний зв'язок один до багатьох (1:n): одному представнику сутності *A* відповідають 0, 1 або декілька представників сутності *B*.

Зв'язок мають наступні сутності:

- Постачальник (Supplier) із Товар (Product);
- Постачальник (Supplier) із Склад (Warehouse);
- Товар (Product) із Склад (Warehouse);
- Товар (Product) із Замовлення (Order);
- Замовник (Client) із Замовлення (Order);
- Платіжна система (PaymentSys) із Рахунок (Invoices);

- Користувач (Users) з усіма сутностями.

Нижче наведено основні атрибути та ключі для кожної сутності:

- Користувач (Users):
 - Ідентифікаційний номер (ID_User) – внутрішній ключ;
 - IP-адреса (IPAdresse) – числовий тип INTEGER;
 - Дата внесення змін у систему (ChangeDate) – тип дата/час DATETIME.
- Замовник (Client):
 - Ідентифікаційний номер (ID_Client) - внутрішній ключ;
 - Ідентифікаційний номер користувача (ID_User) – зовнішній ключ;
 - Ім'я (FirstName) – текстовий тип VARCHAR(size);
 - Прізвище (LastName) – текстовий тип VARCHAR(size);
 - Адреса доставки (AddressCl) – текстовий тип VARCHAR(size);
 - Номер телефону (Phone) – текстовий тип VARCHAR(size);
 - Електронна пошта (Email) – текстовий тип VARCHAR(size);
 - Платіжні реквізити (Invoice) – текстовий тип VARCHAR(size).
- Постачальник (Supplier):
 - Ідентифікаційний номер (ID_Supplier) – внутрішній ключ;
 - Ідентифікаційний номер користувача (ID_User) – зовнішній ключ;
 - Назва компанії (CompanyName) – текстовий тип VARCHAR(size);
 - Адреса (AddressSp) – текстовий тип VARCHAR(size);
 - Номер телефону (Phone) – текстовий тип VARCHAR(size).
- Товар (Product):
 - Ідентифікаційний номер (ID_Product) – внутрішній ключ;
 - Ідентифікаційний номер користувача (ID_User) – зовнішній ключ;
 - Ідентифікаційний номер постачальника (ID_Supplier) – зовнішній ключ;
 - Назва товару (Name) – текстовий тип VARCHAR(size);
 - Назва виробника (Brand) – текстовий тип VARCHAR(size);
 - Ціна товару (Price) – числовий тип FLOAT;

- Кількість наявного товару (Quantity) – числовий тип INTEGER.
- **Замовлення (Order):**
 - Ідентифікаційний номер (ID_Order) – внутрішній ключ;
 - Ідентифікаційний номер користувача (ID_User) – зовнішній ключ;
 - Ідентифікаційний номер товару (ID_Product) – зовнішній ключ;
 - Ідентифікаційний номер замовника (ID_Client) – зовнішній ключ;
 - Статус замовлення (OrderStatus) – текстовий тип VARCHAR(size);
 - Сума замовлення (Price) – числовий тип FLOAT;
 - Дата замовлення (OrderDate) – тип дата/час DATETIME.
- **Доставка (Delivery):**
 - Ідентифікаційний номер (ID_Delivery) - внутрішній ключ;
 - Ідентифікаційний номер користувача (ID_User) - зовнішній ключ;
 - Ідентифікаційний номер замовлення (ID_Order) - зовнішній ключ;
 - Адреса доставки (AddressDl) – текстовий тип VARCHAR(size);
 - Дата доставки (DeliveryDate) – тип дата/час DATETIME.
 - Статус доставки (DeliveryStatus) – текстовий тип VARCHAR(size);.
- **Платіжна система (PaymentSys):**
 - Ідентифікаційний номер (ID_PaymentSys) – внутрішній ключ;
 - Ідентифікаційний номер користувача (ID_User) – зовнішній ключ;
 - Ідентифікаційний номер замовлення (ID_Order) – зовнішній ключ;
 - Ідентифікаційний номер замовника (ID_Client) – зовнішній ключ;
 - Статус оплати (PaymentStatus) – текстовий тип VARCHAR(size);
 - Сума рахунку (Price) – числовий тип FLOAT;
 - Дата оплати (PaymentDate) – тип дата/час DATETIME.
- **Склад (Warehouse):**
 - Ідентифікаційний номер (ID_Warehouse) – внутрішній ключ;
 - Ідентифікаційний номер користувача (ID_User) – зовнішній ключ;
 - Ідентифікаційний номер товару (ID_Product) – зовнішній ключ;
 - Ідентифікаційний номер постачальника (ID_Supplier) – зовнішній ключ;
 - Кількість товару (Quantity) – числовий тип INTEGER;

- Ціна товару (Price) – числовий тип FLOAT.
- Рахунок (Invoices):
 - Ідентифікаційний номер (ID_Invoice) – внутрішній ключ;
 - Ідентифікаційний номер користувача (ID_User) – зовнішній ключ;
 - Ідентифікаційний номер платіжної системи (ID_PaymentSys) – зовнішній ключ;
 - Сума рахунку (Price) – числовий тип FLOAT.
 -

На основі визначених сутностей і зв'язків можна побудувати ER-діаграму (Рис. 2.5), що відображає логіку функціонування бази даних інтернет-магазину.

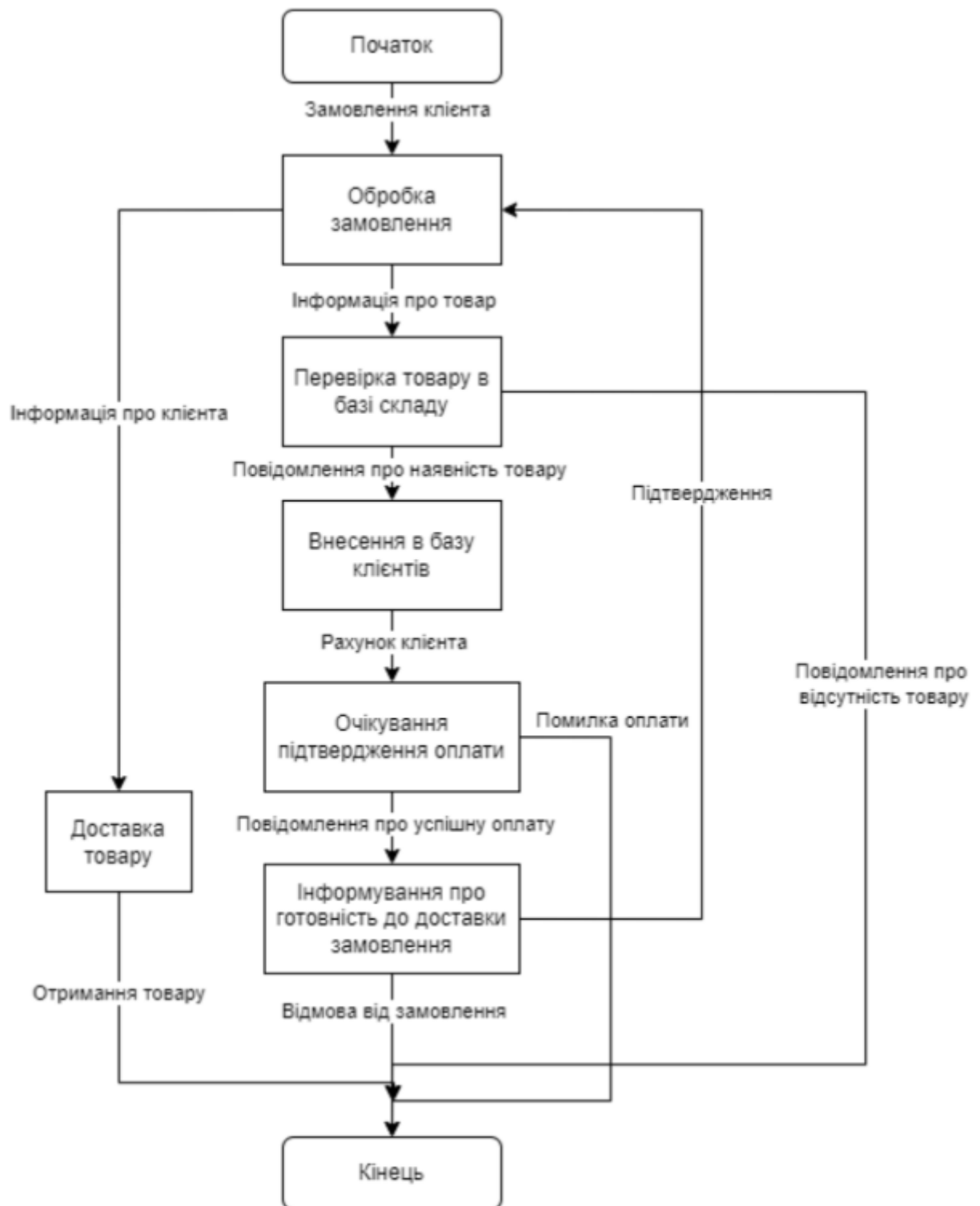


Рис. 2.4. Діаграма життєвого циклу продавця інтернет-магазину.

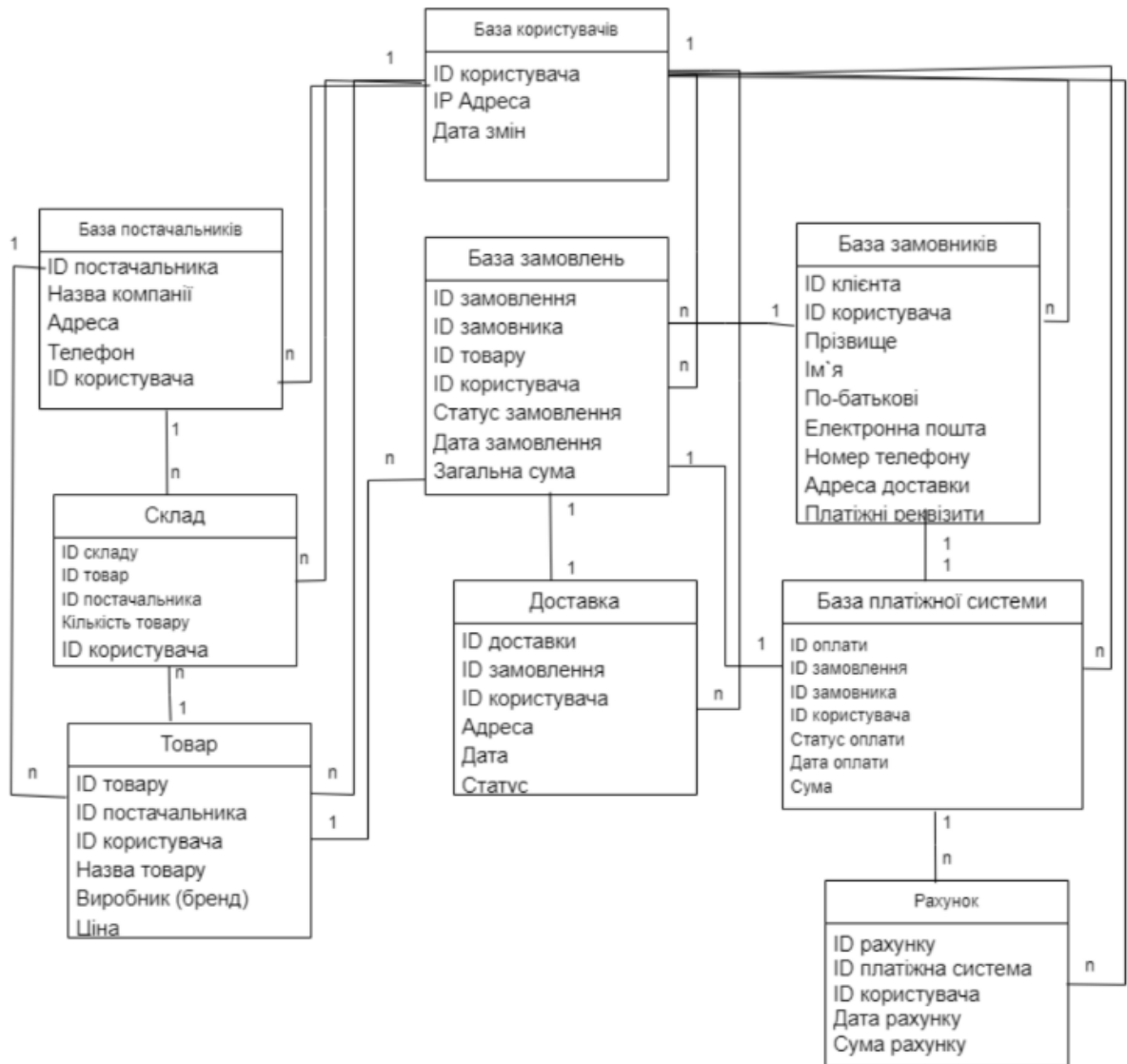


Рис. 2.5. ERD для інфологічної моделі предметної області інтернет-магазину.

2.3. Даталогічна модель предметної області.

Створений на попередніх етапах набір відношень логічної моделі БД повинен бути перевірений на коректність об'єднання атрибутів у кожному відношенні. Перевірка виконується шляхом застосування до кожного відношення процедури послідовної нормалізації. Нормалізація гарантує, що отримана модель не буде мати протиріччя і буде мати мінімальну збитковість. Атрибути в результаті нормалізації будуть згруповані відповідно до існуючих між ними логічних зв'язків.

Процес проектування БД з використанням декомпозиції являє собою процес послідовної нормалізації схем відношень, при цьому кожна наступна ітерація відповідає нормальній формі більш вищого рівня і має кращі властивості у порівнянні з попередньою. Кожній нормальній формі (НФ) відповідає деякий набір обмежень. При виконанні декомпозиції зберігається множина вихідних функціональних залежностей між атрибутами і виконується зворотність.

Відношення знаходиться в третій нормальній формі (3НФ), якщо виконуються наступні вимоги:

- атрибути є атомарними. що означає, що є неподільними у всіх застосуваннях;
- кожен непервинний атрибут функціонально повно залежить від первинного ключа. Повна функціональна залежність передбачає залежність неключового атрибуту від всіх атрибутів одночасно, що входять до складу ключа.
- жоден з непервинних атрибутів у відношенні не є транзитивно залежним від первинного ключа. Вважається, що атрибут С транзитивно залежить від атрибуту А, якщо для атрибутів А, В, С виконуються такі умови $A \rightarrow B$ і $B \rightarrow C$, однак зворотня залежність відсутня[5].

Після процедури нормалізації можна побудувати ERD для даталогічної моделі у третій нормальній формі для заданої предметної області (Рис.2.6).

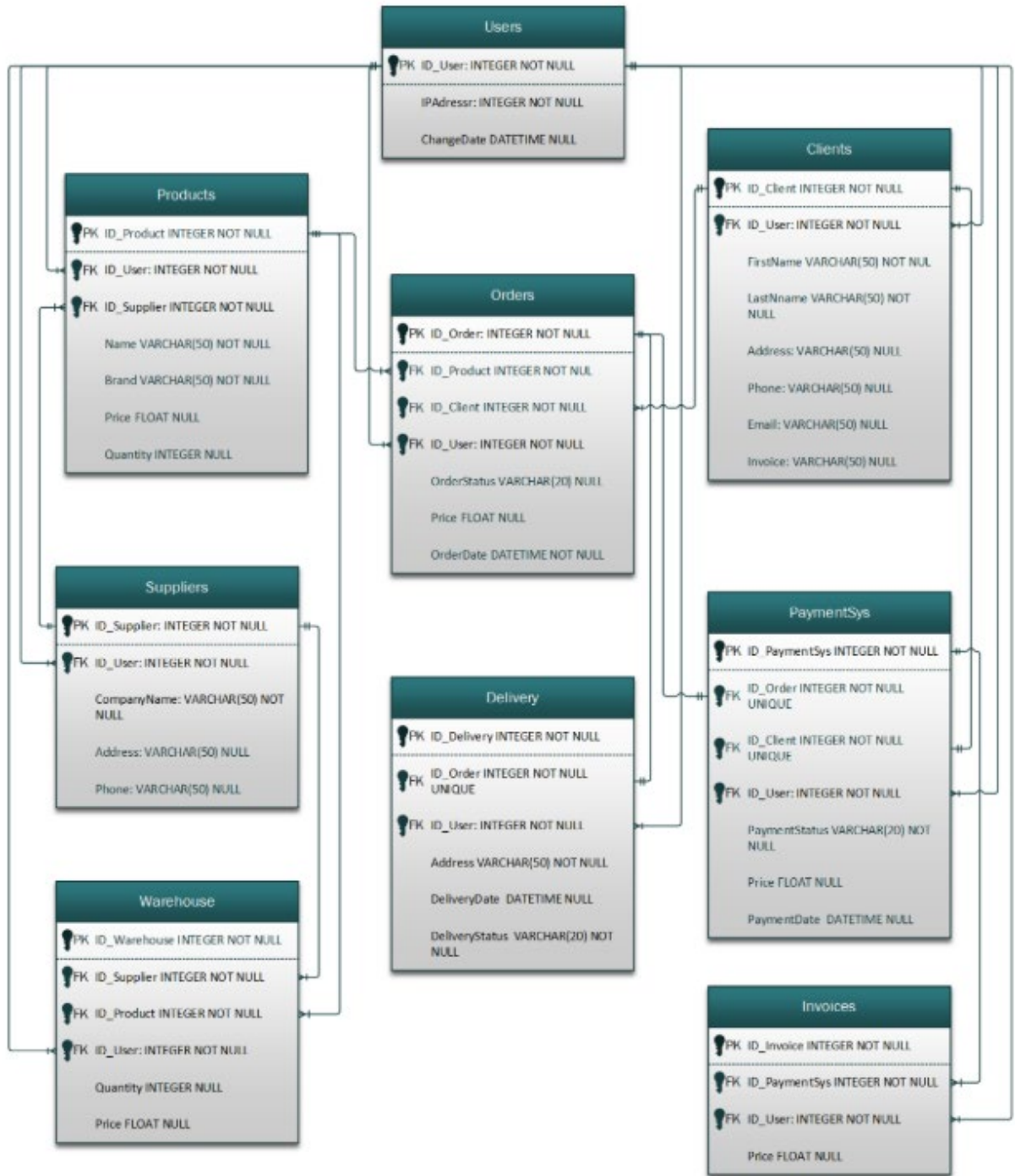


Рис. 2.6. ERD для даталогічної моделі у третій нормальній формі предметної області інтернет-магазин.

3. РОЗРОБЛЕННЯ БАЗИ ДАНИХ

Використовуючи розроблену фізичну ER-діаграму заданої предметної області, можна реалізувати наступні етапи розробки бази даних:

- створення таблиць;
- визначення полів та їхніх типів даних;
- визначення формату відображення для поля та призначення ключів;
- визначити зв'язки між таблицями;
- заповнення таблиць даними;
- створення об'єктів бази даних та запитів: схеми, тригери, індекси, представлення, збережені процедури, користувацькі функції тощо;

Таблиці складають основу бази даних, саме в них зберігаються всі дані. Запис даних — це сукупність значень, пов'язаних між собою. Кожен запис складається з визначеного набору полів, де кожне поле містить дані одного типу.

Тип даних характеризує кожний з елементів запису, визначає вид і межі допустимих значень, які можуть бути введені в поле, а також об'єм пам'яті, який виділяється для цього поля. При виборі типів даних необхідно враховувати можливості СУБД. Microsoft SQL Server підтримує широкий набір типів даних, включаючи символічні, числові, бітові рядки, спеціалізовані числові дані (наприклад, грошові суми), а також дані спеціального формату (дата, час тощо).

3.1. Створення бази даних.

Для створення бази даних використовується оператор CREATE DATABASE. Була створена база даних 'WebShopDB', як показано на коді нижче:

```
CREATE DATABASE WebShopDB
ON(
NAME = 'WebShop',
FILENAME = 'C:\DataBases\WebShopDB.mdf',
SIZE = 100MB,
```

```

MAXSIZE = 100MB,
FILEGROWTH = 10MB
)
LOG ON (
NAME = 'LogWebShopDB.ldf',
FILENAME = 'C:\DataBases\WebShopDB.ldf',
SIZE = 5MB,
MAXSIZE = 50MB,
FILEGROWTH = 5MB
)
COLLATE Cyrillic_General_CI_AS

```

3.2. Створення та забезпечення посилкової цілісності таблиць.

Для кожної сутності предметної області створено відповідну таблицю з визначенням типів даних та властивостей полів. Використовується оператор CREATE TABLE.

При створенні таблиць реалізовані механізми забезпечення цілісності даних:

- обмеження первинного ключа (PRIMARY KEY);
- обмеження зовнішнього ключа (FOREIGN KEY);
- можливість зберігання значень NULL;
- значення за умовчанням (DEFAULT);
- обмеження на унікальність значень (UNIQUE);
- обмеження на умову (CHECK).

Структура таблиць:

Поле	Тип даних	Обмеження
Таблиця Users		
ID_User	INTEGER IDENTITY(1,1) NOT NULL	PRIMARY KEY
IPAdresse	VARCHAR(50)	UNIQUE CHECK (IPAdresse LIKE '%.%.%.%.%')

ChangeDate	DATETIME NOT NULL	DEFAULT CURRENT_TIMESTAMP
Таблица Clients		
ID_Client	INTEGER IDENTITY(1,1) NOT NULL	PRIMARY KEY
ID_User	INTEGER NOT NULL	FOREIGN KEY REFERENCES Users(ID_User)
FirstName	VARCHAR(50) NOT NULL	
LastName	VARCHAR(50) NOT NULL	
AddressCl	VARCHAR(50) NOT NULL	
Phone		CHECK (Phone LIKE '[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]')
Email		CHECK (Email LIKE '%_@_%.__%')
Invoice	VARCHAR(50) NULL	UNIQUE
		UNIQUE (ID_Client, Phone, Email)
Таблица OrdersCl		
ID_Order	INTEGER IDENTITY(1,1) NOT NULL	PRIMARY KEY
ID_Product	INTEGER NOT NULL	FOREIGN KEY REFERENCES Products(ID_Product)
ID_Client	INTEGER NOT NULL	FOREIGN KEY REFERENCES Clients(ID_Client)
ID_User	INTEGER NOT NULL	FOREIGN KEY REFERENCES Users(ID_User)
OrderStatus	VARCHAR(50)	CHECK (OrderStatus IN ('Completed', 'Pending', 'Cancelled'))
Price	FLOAT NOT NULL	CHECK (Price >= 0)
OrderDate	DATETIME NULL	
Таблица Suppliers		
ID_Supplier	INTEGER IDENTITY(1,1) NOT NULL	PRIMARY KEY

ID_User	INTEGER NOT NULL	FOREIGN KEY REFERENCES Users(ID_User)
CompanyName	VARCHAR(50) NOT NULL	
AddressSp	VARCHAR(50) NULL	
Phone	VARCHAR(50)	UNIQUE CHECK (Phone LIKE '[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9]')
Таблица Products		
ID_Product	INTEGER IDENTITY(1,1) NOT NULL	PRIMARY KEY
ID_User	INTEGER NOT NULL	FOREIGN KEY REFERENCES Users(ID_User)
ID_Supplier	INTEGER NOT NULL	FOREIGN KEY REFERENCES Suppliers (ID_Supplier)
Name	VARCHAR(50) NOT NULL	
Brand	VARCHAR(50) NOT NULL	
Price	FLOAT NOT NULL	CHECK (Price >= 0)
Quantity	INTEGER NOT NULL	CHECK (Quantity >= 0)
Таблица Delivery		
ID_Delivery	INTEGER IDENTITY(1,1) NOT NULL	PRIMARY KEY
ID_User	INTEGER NOT NULL	FOREIGN KEY REFERENCES Users(ID_User)
ID_Order	INTEGER NOT NULL	UNIQUE FOREIGN KEY REFERENCES OrdersCl(ID_Order)
AddressDl	VARCHAR(50) NOT NULL	
DeliveryDate	DATETIME NULL	
DeliveryStatus	VARCHAR(50)	CHECK (DeliveryStatus IN ('Shipped', 'Delivered', 'Cancelled'))
Таблица PaymentSys		
ID_PaymentSys	INTEGER IDENTITY(1,1) NOT NULL	PRIMARY KEY
ID_Order	INTEGER NOT NULL	UNIQUE FOREIGN KEY REFERENCES OrdersCl(ID_Order)

ID_Client	INTEGER NOT NULL	UNIQUE FOREIGN KEY REFERENCES Clients(ID_Client)
ID_User	INTEGER NOT NULL	UNIQUE FOREIGN KEY REFERENCES Users(ID_User)
PaymentStatus	VARCHAR(50)	CHECK (PaymentStatus IN ('Success','Failed'))
PaymentDate	DATETIME NOT NULL	DEFAULT CURRENT_TIMESTAMP
Price	FLOAT NOT NULL	CHECK (Price >= 0)
Таблиця Warehouse		
ID_Warehouse	INTEGER IDENTITY(1,1) NOT NULL	PRIMARY KEY
ID_Supplier	INTEGER NOT NULL	UNIQUE FOREIGN KEY REFERENCES Suppliers(ID_Supplier)
ID_Product	INTEGER NOT NULL	UNIQUE FOREIGN KEY REFERENCES Products(ID_Product)
ID_User	INTEGER NOT NULL	UNIQUE FOREIGN KEY REFERENCES Users(ID_User)
Price	FLOAT NOT NULL	CHECK (Price >= 0)
Quantity	INTEGER NOT NULL	CHECK (Quantity >= 0)
Таблиця Invoice		
ID_Invoice	INTEGER IDENTITY(1,1) NOT NULL	PRIMARY KEY
ID_PaymentSys	INTEGER NOT NULL	UNIQUE FOREIGN KEY REFERENCES PaymentSys(ID_PaymentSys)
ID_User	INTEGER NOT NULL	UNIQUE FOREIGN KEY REFERENCES Users(ID_User)
Price	FLOAT NOT NULL	CHECK (Price >= 0)

Таблиця 3.1. Таблиці бази даних «Інтернет-магазин».

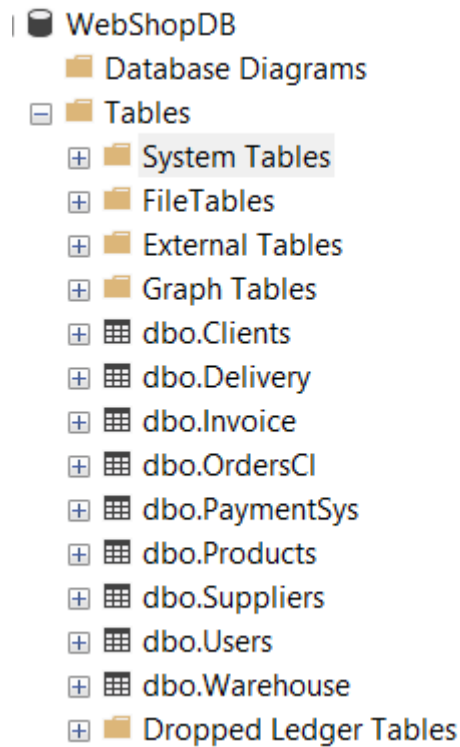


Рисунок 3.1. Список створених таблиць бази даних «Інтернет-магазин».

Усі таблиці були зв'язані за допомогою зовнішніх ключів.

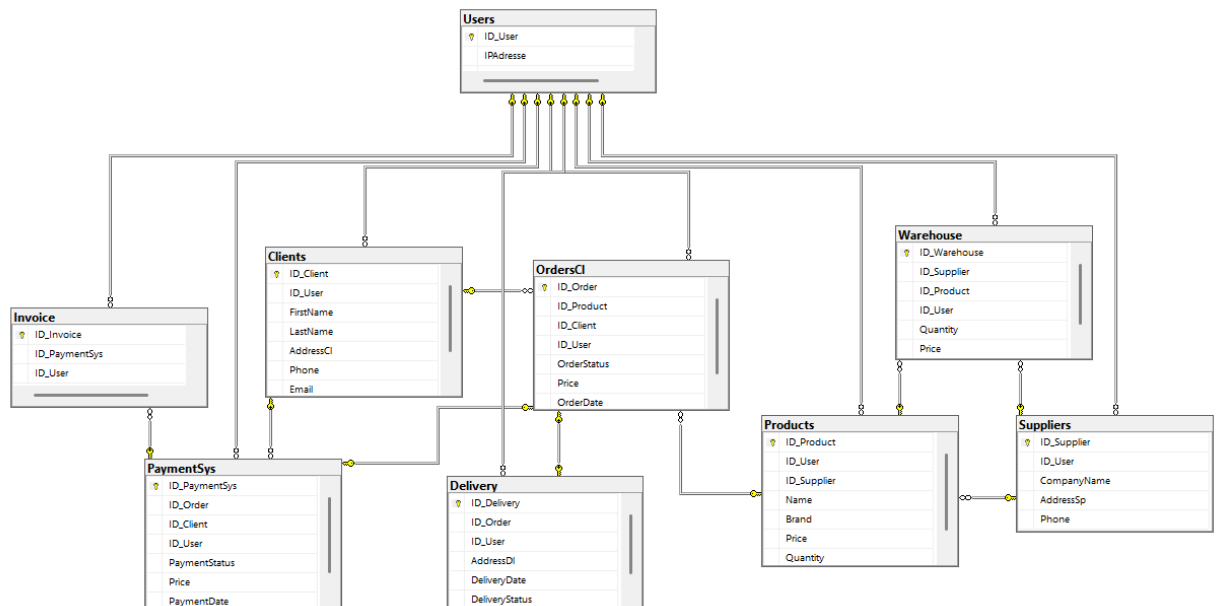


Рисунок 3.2. Діаграма зв'язків бази даних.

3.3. Створення подань.

Були написані запити, які забезпечують ефективний доступ до необхідної інформації бази даних. Вони допомагають отримати структуровані дані та полегшують аналітичну обробку.

1) Приклад 1. Виведення інформації про замовлення магазину.

```
CREATE OR ALTER VIEW ViewClientOrders AS
SELECT
    o.ID_Order, C.LastName + ' ' + C.FirstName AS ClientName,
    P.Name AS ProductName, P.Brand AS ProductBrand,
    FORMAT(O.OrderDate, 'dd/MM/yyyy') as OrderDate,
    O.OrderStatus,
    D.DeliveryStatus,
    O.Price
FROM OrdersCl O
JOIN Clients C ON O.ID_Client = C.ID_Client
JOIN Products P ON O.ID_Product = P.ID_Product
LEFT JOIN Delivery D ON D.ID_Order = O.ID_Order;
```

	ID_Order	ClientName	ProductName	ProductBrand	OrderDate	OrderStatus	Price
1	1	Petrov Ivan	Laptop	HP	01/03/2025	Completed	1000
2	2	Ivanova Olga	Laptop	Apple	02/03/2025	Pending	500
3	3	Kovalenko Andriy	Laptop	Lenovo	03/03/2025	Cancelled	100
4	4	Tarasenko Svitlana	Monitor	HP	04/03/2025	Completed	300
5	5	Melnyk Viktor	Printer	HP	05/03/2025	Pending	250
6	6	Kuzmenko Natalia	Sofa	IKEA	06/03/2025	Completed	25
7	7	Moroz Viktor	Sofa	IKEA	07/03/2025	Completed	15
8	8	Nikolenko Tatiana	Bicycle	SportX	08/03/2025	Cancelled	10
9	9	Vasylenko Marina	Helmet	SafeRide	09/03/2025	Completed	20
10	10	Vasylenko Marina	Sneakers	Nike	10/03/2025	Pending	1,5

Рис.3.3. Інформація про замовлення магазину.

2) Приклад 2. Виведення інформації про товар на складі.

```
CREATE VIEW ViewStockProducts
AS SELECT w.ID_Product, p.Name, p.Brand, s.CompanyName, SUM(w.Quantity) AS
Total_Quantity, SUM(w.Price) AS TotalPrice
FROM Warehouse w
JOIN OrdersCl o ON w.ID_Product = o.ID_Product
LEFT JOIN Products p ON w.ID_Product = p.ID_Product
JOIN Suppliers s ON s.ID_Supplier = p.ID_Supplier
GROUP BY w.ID_Product, p.Name, p.Brand , s.CompanyName;

SELECT *
FROM ViewStockProducts
ORDER BY Total_Quantity DESC
```

Results		Messages				
	ID_Product	Name	Brand	CompanyName	TotalQuantity	TotalPrice
1	1	Laptop	HP	Tech Supplies	150	3000
2	2	Laptop	Apple	Home Goods	30	500
3	3	Laptop	Lenovo	Auto Parts	100	100
4	4	Monitor	HP	Tech Supplies	70	300
5	5	Printer	HP	Tech Supplies	40	250
6	6	Sofa	IKEA	Electronics Co.	200	25
7	7	Sofa	IKEA	Furniture Ltd.	50	16
8	8	Bicycle	SportX	Sports Gear	150	10
9	9	Helmet	Safe...	Sports Gear	500	40
10	10	Snea...	Nike	Sports Gear	1000	1.5

Рис.3.4. Інформація про товар на складі.

- 3) Приклад 3. Виведення інформації про успішні оплати для подальшої доставки товару до клієнтів.

```

ALTER VIEW ViewSuccessPayments AS
SELECT pr.Name AS ProductName, c.FirstName, c.LastName, c.Phone, c.AddressCl
FROM Clients c
JOIN PaymentSys p
  ON p.ID_Client = c.ID_Client
JOIN OrdersCl o
  ON o.ID_Client = c.ID_Client
JOIN Products pr
  ON pr.ID_Product = o.ID_Product
WHERE p.PaymentStatus = 'Success'

SELECT * FROM ViewSuccessPayments
ORDER BY ProductName

```

	ID_Order	ProductName	FirstName	LastName	Phone	AddressCl
1	1	Laptop	Ivan	Petrov	050-123-4567	Kyiv, St. Shevchenka, 10
2	3	Laptop	Andriy	Kovalenko	050-345-6789	Lviv, St. Franka, 15
3	4	Monitor	Svitlana	Tarasenko	050-456-7890	Kharkiv, St. Shevchenka, 20
4	6	Sofa	Natalia	Kuzmenko	050-678-9012	Dnipro, St. Kotsiubynskoho, 8
5	7	Sofa	Viktor	Moroz	050-789-0123	Kyiv, St. Bessarabka, 10
6	9	Helmet	Marina	Vasylenko	050-901-2345	Kharkiv, St. Kurchatova, 40
7	10	Sneakers	Marina	Vasylenko	050-012-3456	Odesa, St. Derybasivska, 5

Рис.3.5. Інформація про успішні оплати клієнтів.

- 4) Приклад 5. Інформація про компанії з найбільшою виручкою. Виведення їхньої загальної кількість покупців. Пошук компанії з найбільшою виручкою.

```

ALTER VIEW ViewCompanyProductSales AS

```

```

SELECT
    s.CompanyName,
    p.Name AS ProductName,
    p.Brand,
    ps.PaymentDate,
    SUM(p.Quantity) AS TotalSales,
    COUNT(DISTINCT c.ID_Client) AS TotalClients,
    SUM(i.Price) AS TotalRevenue
FROM OrdersCl o
LEFT JOIN Clients c ON c.ID_Client = o.ID_Client
LEFT JOIN Products p ON o.ID_Product = p.ID_Product
LEFT JOIN PaymentSys ps ON o.ID_Order = ps.ID_Order
LEFT JOIN Invoice i ON ps.ID_PaymentSys = i.ID_PaymentSys
LEFT JOIN Suppliers s ON s.ID_Supplier = p.ID_Supplier
WHERE ps.PaymentStatus = 'Success'
GROUP BY
    s.CompanyName,
    p.Name,
    p.Brand,
    ps.PaymentDate

```

	CompanyName	ProductName	Brand	PaymentDate	TotalSales	TotalClients	TotalRevenue
1	Auto Parts	Laptop	Lenovo	2025-03-04 00:00:00.000	40	1	100
2	Electronics Co.	Sofa	IKEA	2025-03-07 00:00:00.000	10	1	25
3	Furniture Ltd.	Sofa	IKEA	2025-03-08 00:00:00.000	12	1	15
4	Sports Gear	Helmet	SafeRide	2025-03-10 00:00:00.000	50	1	20
5	Sports Gear	Sneakers	Nike	2025-03-11 00:00:00.000	35	1	1,5
6	Tech Supplies	Laptop	HP	2025-03-02 00:00:00.000	50	1	1000
7	Tech Supplies	Monitor	HP	2025-03-05 00:00:00.000	25	1	300

Рис.3.6. Компанії з найбільшою виручкою.

- 1) Приклад 6. Інформація про успішні замовлення магазину за місяць для кожної компанії. Розрахунок загальної кількості продажів, прибутку та клієнтів для кожного бренду компанії.

```

CREATE OR ALTER VIEW ViewCompanyMonthSales
AS SELECT
    s.CompanyName,
    p.Name AS ProductName,
    p.Brand,
    FORMAT(MONTH(CAST(ps.PaymentDate AS DATE)), '00') AS PaymentMonth,
    SUM(p.Quantity) AS TotalSales,
    COUNT(DISTINCT c.ID_Client) AS TotalClients,
    SUM(i.Price) AS TotalRevenue,
    SUM(SUM(i.Price)) OVER (PARTITION BY s.CompanyName) AS TotalCompanyRevenue
FROM OrdersCl o
LEFT JOIN Clients c ON c.ID_Client = o.ID_Client
LEFT JOIN Products p ON o.ID_Product = p.ID_Product
LEFT JOIN PaymentSys ps ON o.ID_Order = ps.ID_Order
LEFT JOIN Invoice i ON ps.ID_PaymentSys = i.ID_PaymentSys
LEFT JOIN Suppliers s ON s.ID_Supplier = p.ID_Supplier

```

```

WHERE ps.PaymentStatus = 'Success' AND o.OrderStatus = 'Completed'
GROUP BY
    s.CompanyName,
    p.Name,
    p.Brand,
    FORMAT(MONTH(CAST(ps.PaymentDate AS DATE)), '00');

```

	CompanyName	ProductName	Brand	PaymentMonth	TotalSales	TotalClients	TotalRevenue	TotalCompanyRevenue
1	Auto Parts	Laptop	Lenovo	03	40	1	100	100
2	Electronics Co.	Sofa	IKEA	03	10	1	25	25
3	Furniture Ltd.	Sofa	IKEA	03	12	1	15	15
4	Home Goods	Laptop	Apple	03	30	1	500	500
5	Sports Gear	Bicycle	SportX	03	15	1	10	31,5
6	Sports Gear	Helmet	SafeRide	03	50	1	20	31,5
7	Sports Gear	Sneakers	Nike	03	35	1	1,5	31,5
8	Tech Supplies	Laptop	HP	03	50	1	1000	1550
9	Tech Supplies	Monitor	HP	03	25	1	300	1550
10	Tech Supplies	Printer	HP	03	20	1	250	1550

Рис. 3.7. Інформація про успішні замовлення магазину за місяць для кожної компанії.

3.4. Створення тригерів.

Для підвищення рівня автоматизації бази даних були визначені тригери для забезпечення автоматичного виконання певних дій. Тригер — це особливий різновид процедури, який автоматично викликається під час виконання операцій INSERT, UPDATE, DELETE [4].

1) Приклад 1. Оновлення дати у таблиці Users при будь-яких змін у системі.

```

CREATE TRIGGER trg_changes_monitoring_products
ON Products
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE Users
    SET ChangeDate = GETDATE()
    FROM Users u
    INNER JOIN inserted i ON u.ID_User = i.ID_User;
END;

```

Тригер був створений для кожної таблиці. Демонстрація автоматичного оновлення поточної дати після внесення змін у таблицю:

-- Перевірка роботи триггеру -----

```

SELECT * FROM Products WHERE ID_Product = 1;
SELECT * FROM Users WHERE ID_User = 1;

UPDATE Products
SET Price = 1500
WHERE ID_Product = 1;

SELECT * FROM Products WHERE ID_Product = 1;
SELECT * FROM Users WHERE ID_User = 1;

```

89 %

Results Messages

	ID_Product	ID_User	ID_Supplier	Name	Brand	Price	Quantity
1	1	1	1	Laptop	HP	1500	50

	ID_User	IPAdresse	ChangeDate
1	1	192.168.0.1	2024-03-01 16:00:00.000

	ID_Product	ID_User	ID_Supplier	Name	Brand	Price	Quantity
1	1	1	1	Laptop	HP	1500	50

	ID_User	IPAdresse	ChangeDate
1	1	192.168.0.1	2025-03-24 09:08:15.213

Рис. 3.8. Перевірка роботи триггеру.

2) Приклад 2. Оновлення статусу замовлення після здійснення платежу.

Якщо платіж успішний, статус замовлення набуває статусу 'Pending'. В іншому разі – скасування замовлення.

```

CREATE TRIGGER trg_changes_monitoring_orders
ON OrdersCl
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE Users
    SET ChangeDate = GETDATE()
    FROM Users u
    INNER JOIN inserted i ON u.ID_User = i.ID_User;
END;

```

```
-- Перевірка триггеру --
SELECT * FROM OrdersCl WHERE ID_Client IN (1, 2);
SELECT * FROM PaymentSys WHERE ID_PaymentSys IN (1, 2)

UPDATE PaymentSys
SET PaymentStatus = 'Failed'
WHERE ID_PaymentSys = 1;

UPDATE PaymentSys
SET PaymentStatus = 'Success'
WHERE ID_PaymentSys = 2;

SELECT * FROM OrdersCl WHERE ID_Client IN (1, 2);
SELECT * FROM PaymentSys WHERE ID_PaymentSys IN (1, 2)
-----
```

9 %

Results Messages

	ID_Order	ID_Product	ID_Client	ID_User	OrderStatus	Price	OrderDate
1	1	1	1	1	Completed	1000	2025-03-01 00:00:00.000
2	2	2	2	2	Pending	500	2025-03-02 00:00:00.000

	ID_PaymentSys	ID_Order	ID_Client	ID_User	PaymentStatus	Price	PaymentDate
1	1	1	1	1	Success	1000	2025-03-02 00:00:00.000
2	2	2	2	2	Failed	500	2025-03-03 00:00:00.000

	ID_Order	ID_Product	ID_Client	ID_User	OrderStatus	Price	OrderDate
1	1	1	1	1	Cancelled	1000	2025-03-01 00:00:00.000
2	2	2	2	2	Pending	500	2025-03-02 00:00:00.000

	ID_PaymentSys	ID_Order	ID_Client	ID_User	PaymentStatus	Price	PaymentDate
1	1	1	1	1	Failed	1000	2025-03-02 00:00:00.000
2	2	2	2	2	Success	500	2025-03-03 00:00:00.000

Рис. 3.9. Перевірка роботи триггеру.

3) Приклад 3. Оновлення статусу доставки після здійснення замовлення. Якщо замовлення в статусі очікування, відбувається процес доставки. Після доставки товару до клієнта, замовлення є завершеним, в іншому випадку – скасоване.

```
CREATE TRIGGER trg_delivery_order_status
ON Delivery
AFTER INSERT, UPDATE
AS
BEGIN
```



```
SET NOCOUNT ON;
```

```
UPDATE Delivery
SET DeliveryStatus = 'Shipped'
FROM Delivery d
JOIN OrdersCl o ON d.ID_Order = o.ID_Order
WHERE o.OrderStatus = 'Pending';
```

```
UPDATE OrdersCl
SET OrderStatus = 'Completed'
FROM OrdersCl o
JOIN inserted i ON o.ID_Order = i.ID_Order
WHERE i.DeliveryStatus = 'Delivered';
```

```
UPDATE OrdersCl
SET OrderStatus = 'Cancelled'
FROM OrdersCl o
JOIN inserted i ON o.ID_Order = i.ID_Order
WHERE i.DeliveryStatus = 'Cancelled';
```

```
END;
```

The screenshot shows a SQL query editor with the following queries:

```
UPDATE OrdersCl
SET OrderStatus = 'Pending'
WHERE ID_Client = 3

UPDATE Delivery
SET DeliveryStatus = 'Delivered'
WHERE ID_Delivery = 5;

UPDATE Delivery
SET DeliveryStatus = 'Cancelled'
WHERE ID_Delivery = 6;

SELECT o.ID_Client, d.ID_Delivery, o.OrderStatus, d.DeliveryStatus
FROM OrdersCl o
JOIN Delivery d
ON d.ID_Order = o.ID_Order
WHERE ID_Delivery IN (3, 5, 6) AND ID_Client IN (3, 5, 6)
```

Below the queries, there is a 'Results' tab showing the output of the last query. The results are as follows:

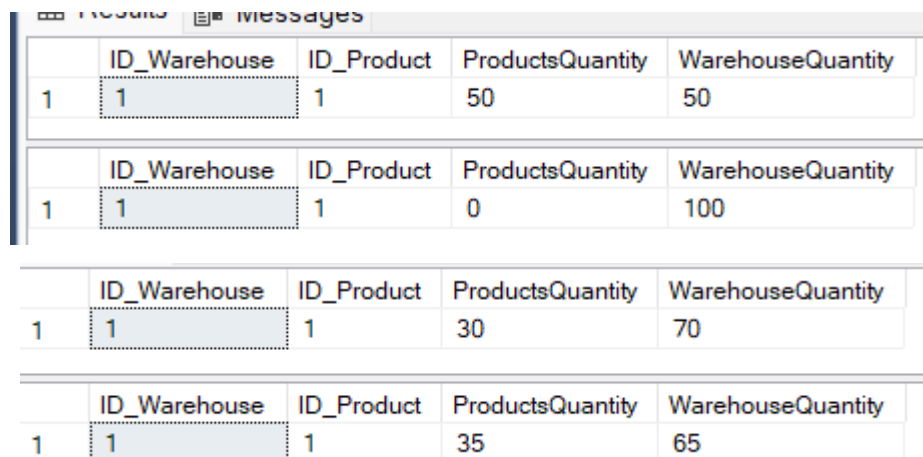
	ID_Client	ID_Delivery	OrderStatus	DeliveryStatus
1	3	3	Pending	Shipped
2	5	5	Completed	Delivered
3	6	6	Cancelled	Cancelled

Рис. 3.10. Перевірка роботи триггеру.

4) Приклад 4. Оновлення кількості товарів на складі після видалення клієнтом замовлення або зміни кількості товарів.

```
CREATE TRIGGER trg_update_warehouse_after_products_modification
ON Products
AFTER DELETE, UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE w
        SET w.Quantity = w.Quantity + d.Quantity
        FROM Warehouse w
        JOIN DELETED d ON w.ID_Product = d.ID_Product;

    UPDATE w
        SET w.Quantity = w.Quantity - i.Quantity
        FROM Warehouse w
        JOIN DELETED d ON w.ID_Product = d.ID_Product
        JOIN INSERTED i ON d.ID_Product = i.ID_Product;
END;
```



	ID_Warehouse	ID_Product	ProductsQuantity	WarehouseQuantity
1	1	1	50	50
1	1	1	0	100
1	1	1	30	70
1	1	1	35	65

Рис. 3.11. Перевірка роботи триггеру.

3.5. Створення функцій.

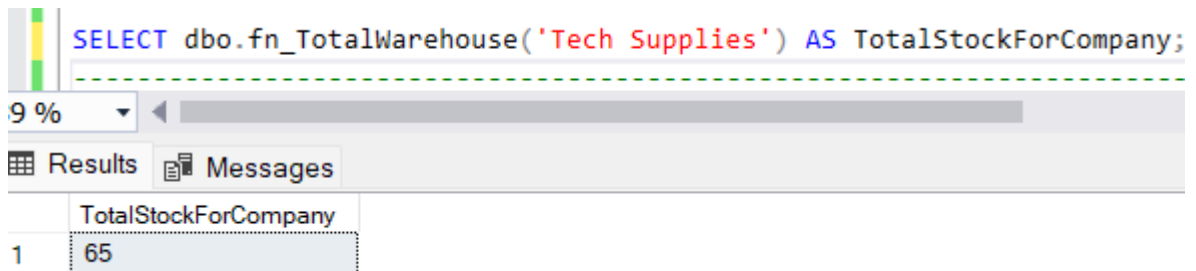
1) Приклад 1. Підрахунок загальної кількості товарів на складі для певних постачальників.

```
CREATE FUNCTION fn_TotalWarehouse(@SupplierCompanyName NVARCHAR(255))
RETURNS INT
AS
BEGIN
    DECLARE @TotalStock INT;
    SELECT @TotalStock = SUM(Quantity)
    FROM Warehouse w
```

```

JOIN Suppliers s ON w.ID_Supplier = s.ID_Supplier
WHERE s.CompanyName = @SupplierCompanyName;
RETURN @TotalStock;
END;

```



	TotalStockForCompany
1	65

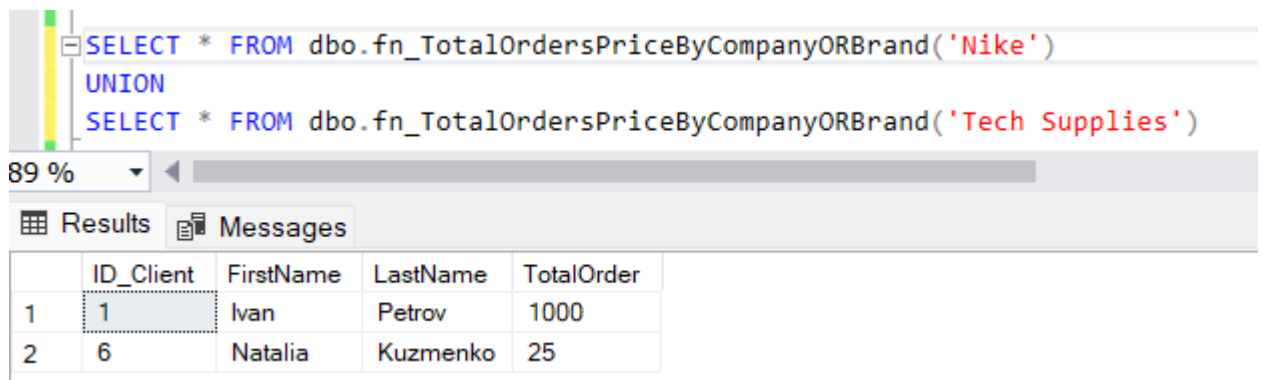
Рис. 3.12. Перевірка роботи функції.

2) Приклад 2. Інформація про замовлення клієнта для заданої компанії або назви бренду.

```

CREATE FUNCTION fn_TotalOrdersPriceByCompanyORBrand(@BrandOrCompanyName VARCHAR(255))
RETURNS TABLE
AS
RETURN (
    SELECT c.ID_Client, c.FirstName, c.LastName, SUM(o.Price) AS TotalOrder
    FROM Clients c
    JOIN OrdersCl o ON c.ID_Client = o.ID_Client
    JOIN Products p ON p.ID_Product = o.ID_Product
    JOIN Suppliers s ON s.ID_Supplier = p.ID_Supplier
    WHERE s.CompanyName = @BrandOrCompanyName OR p.Brand = @BrandOrCompanyName
    GROUP BY c.ID_Client, c.FirstName, c.LastName);

```



	ID_Client	FirstName	LastName	TotalOrder
1	1	Ivan	Petrov	1000
2	6	Natalia	Kuzmenko	25

Рис. 3.13. Перевірка роботи функції.

3.6. Створення курсорів.

1) Приклад 1. Перевірка наявності товару перед обробкою замовлення.

Якщо недостатня кількість наявного товару, замовлення залишається у статусі обробки.

```

DECLARE @OrderID INT, @ProductID INT, @RequiredQuantity INT, @StockQuantity INT;

DECLARE order_cursor CURSOR FOR
SELECT o.ID_Order, o.ID_Product, p.Quantity
FROM OrdersCl o
JOIN Products p ON o.ID_Product = p.ID_Product
WHERE o.OrderStatus = 'Pending';
OPEN order_cursor;
FETCH NEXT FROM order_cursor INTO @OrderID, @ProductID, @RequiredQuantity;
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Перевірка кількості товару на складі
    SELECT @StockQuantity = Quantity
    FROM Warehouse
    WHERE ID_Product = @ProductID;
    IF @StockQuantity >= @RequiredQuantity
    BEGIN
        -- Оновлення статусу замовлення
        UPDATE OrdersCl
        SET OrderStatus = 'Completed'
        WHERE ID_Order = @OrderID;
        -- Зменшення кількості товару на складі
        UPDATE Warehouse
        SET Quantity = Quantity - @RequiredQuantity
        WHERE ID_Product = @ProductID;
    END
    FETCH NEXT FROM order_cursor INTO @OrderID, @ProductID, @RequiredQuantity;
END;
CLOSE order_cursor;
DEALLOCATE order_cursor;

```

2) Приклад 2. Загальний процес доставки замовлення.

Після успішної оплати замовлення перебуває у стані обробки. У разі наявної інформації про адресу замовника, початок процесу доставки та зміна статусу доставки на відправлено. Також встановлення орієнтовної дати доставки (п'ять днів від поточної дати).

```

DECLARE @OrdersID INT, @DeliveryID INT, @ClientID INT, @AddressDelivery VARCHAR(100),
@AddressClient VARCHAR(100);
DECLARE delivery_cursor CURSOR FOR
SELECT o.ID_Order, d.ID_Delivery, c.ID_Client, d.AddressDl, c.AddressCl
FROM OrdersCl o
JOIN Delivery d ON o.ID_Order = d.ID_Order
JOIN PaymentSys p ON o.ID_Order = p.ID_Order
JOIN Clients c ON o.ID_Client = c.ID_Client
WHERE p.PaymentStatus = 'Success';
OPEN delivery_cursor;
FETCH NEXT FROM delivery_cursor INTO @OrdersID, @DeliveryID, @ClientID, @AddressDelivery,
@AddressClient
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Після успішної оплати оновлення статусу замовлення
    UPDATE OrdersCl
    SET OrderStatus = 'Pending'
    WHERE ID_Order = @OrdersID;
    -- Перевірка наявності адреси покупця та відповідності до адреси доставки
    IF @AddressDelivery IS NULL AND @AddressClient IS NOT NULL
    BEGIN
        UPDATE Delivery
        SET AddressDl = @AddressClient
        WHERE ID_Delivery = @DeliveryID;
        SET @AddressDelivery = @AddressClient;
    END
    -- Оновлення статусу доставки та встановлення приблизної дати отримання
    IF @AddressDelivery IS NOT NULL
    BEGIN
        UPDATE Delivery
        SET DeliveryStatus = 'Shipped',
            DeliveryDate = DATEADD(DAY, 5, GETDATE())
        WHERE ID_Delivery = @DeliveryID;
    END
    FETCH NEXT FROM delivery_cursor INTO @OrdersID, @DeliveryID, @ClientID,
@AddressDelivery, @AddressClient;
END;
CLOSE delivery_cursor;
DEALLOCATE delivery_cursor;

```

3) Приклад 3. Видача рахунку покупцю після успішної оплати.

```

DECLARE @PaymentID INT, @OrderID INT, @ID_User INT, @Price FLOAT

```

```

DECLARE invoice_cursor CURSOR FOR
SELECT ID_PaymentSys, ID_Order, ID_User, Price
FROM PaymentSys
WHERE PaymentStatus = 'Success'
      AND ID_PaymentSys NOT IN (SELECT ID_PaymentSys
                                FROM Invoice)

OPEN invoice_cursor;
FETCH NEXT FROM invoice_cursor INTO @PaymentID, @OrderID, @ID_User, @Price;

WHILE @@FETCH_STATUS = 0
BEGIN
    -- Створення рахунку

    INSERT INTO Invoice(ID_PaymentSys, ID_User, Price)
    VALUES (@PaymentID, @ID_User, @Price);
END;

CLOSE invoice_cursor;
DEALLOCATE invoice_cursor

```

3.7. Створення процедур.

1) Приклад 1. Знаходження найдорожчого та найдешевшого товару.

```

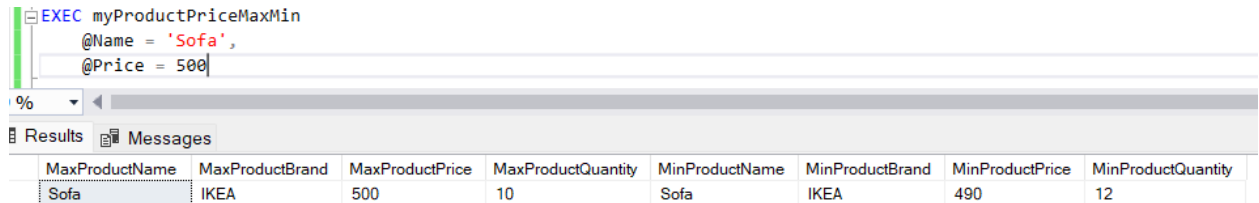
CREATE OR ALTER PROCEDURE myProductPriceMaxMin
    @Name VARCHAR(75) = null,
    @Price FLOAT = null
AS
BEGIN
    SELECT MaxProduct.Name AS MaxProductName,
           MaxProduct.Brand AS MaxProductBrand,
           MaxProduct.Price AS MaxProductPrice,
           MaxProduct.Quantity AS MaxProductQuantity,
           MinProduct.Name AS MinProductName,
           MaxProduct.Brand AS MinProductBrand,
           MinProduct.Price AS MinProductPrice,
           MinProduct.Quantity AS MinProductQuantity
    FROM (SELECT TOP 1 Name, Brand, Price, Quantity
          FROM Products
          WHERE Name LIKE @Name AND Price <= @Price
          ORDER BY Price DESC) AS MaxProduct

    CROSS JOIN( SELECT TOP 1 Name, Brand, Price, Quantity
                FROM Products
                WHERE Name LIKE @Name AND Price <= @Price
                ORDER BY Price ASC) AS MinProduct
END;
GO
-- Виконання процедури

```

```
DECLARE @retminbrand VARCHAR(75), @retmin FLOAT, @retmax FLOAT, @retmaxbrand
VARCHAR(75), @retrows INT;
```

```
EXEC myProductPriceMaxMin
    @Name = 'Sofa',
    @Price = 500
```

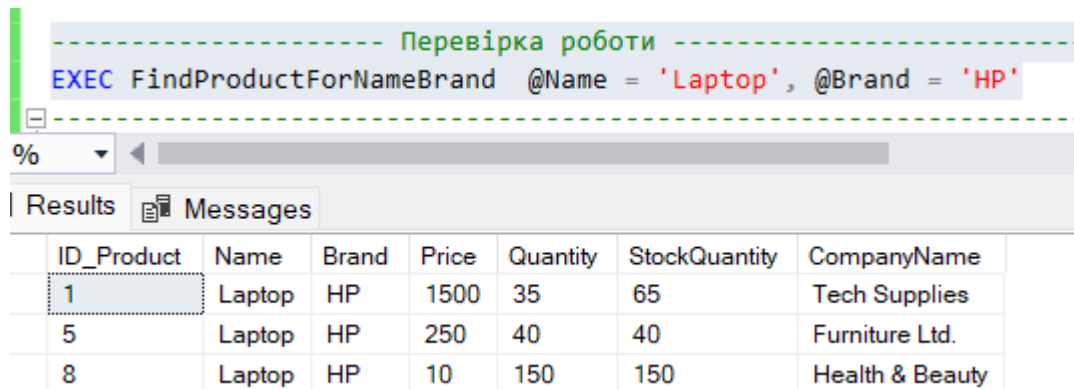


MaxProductName	MaxProductBrand	MaxProductPrice	MaxProductQuantity	MinProductName	MinProductBrand	MinProductPrice	MinProductQuantity
Sofa	Ikea	500	10	Sofa	Ikea	490	12

Рис. 3.14. Результат роботи процедури.

2) Приклад 2. Інформація про товар за заданим брендом та типом. Виведення різних постачальників такого товару.

```
ALTER PROCEDURE FindProductForNameBrand
    @Name VARCHAR(75), @Brand VARCHAR(75)
AS
BEGIN
    SELECT p.ID_Product, p.Name, p.Brand, p.Price, p.Quantity, w.Quantity AS
    StockQuantity, s.CompanyName
    FROM Products p
    JOIN Suppliers s ON s.ID_Supplier = p.ID_Supplier
    JOIN Warehouse w ON w.ID_Product = p.ID_Product
    WHERE Name LIKE @Name AND Brand LIKE @Brand
    ORDER BY p.Price DESC
END;
```



----- Перевірка роботи -----

```
EXEC FindProductForNameBrand @Name = 'Laptop', @Brand = 'HP'
```

ID_Product	Name	Brand	Price	Quantity	StockQuantity	CompanyName
1	Laptop	HP	1500	35	65	Tech Supplies
5	Laptop	HP	250	40	40	Furniture Ltd.
8	Laptop	HP	10	150	150	Health & Beauty

Рис. 3.15. Результат роботи процедури.

3) Приклад 3. Інформація про замовлення за заданим статусом оплати.

Аналогічна реалізація можлива за заданими окремо або разом статусами доставки або замовлення.

```
ALTER PROCEDURE FindFailedPayments
    @PaymentStatus VARCHAR(75)
AS BEGIN
    SELECT ps.ID_PaymentSys, o.ID_Order,
           c.ID_Client, c.FirstName, c.LastName, c.Email, c.Phone,
           c.Invoice, ps.Price, ps.PaymentDate
    FROM PaymentSys ps
    JOIN OrdersCl o ON o.ID_Order = ps.ID_Order
    JOIN Clients c ON c.ID_Client = o.ID_Client
    JOIN Invoice i ON i.ID_PaymentSys = ps.ID_PaymentSys
    JOIN Products p ON p.ID_Product = o.ID_Product
    WHERE PaymentStatus LIKE @PaymentStatus
    ORDER BY ps.Price DESC
END;
```

----- Перевірка роботи -----
 EXEC FindFailedPayments @PaymentStatus = 'Failed'

89 %

	ID_PaymentSys	ID_Order	ID_Client	FirstName	LastName	Email	Phone	Invoice	Price	PaymentDate
1	1	1	1	Ivan	Petrov	ivan@example.com	050-123-4567	INV001	1000	2025-03-02 00:00:00.000
2	5	5	5	Dmytro	Melnyk	dmytro@example.com	050-567-8901	INV005	250	2025-03-06 00:00:00.000
3	8	8	8	Tatiana	Nikolenko	tatiana@example.com	050-890-1234	INV008	10	2025-03-09 00:00:00.000

Рис. 3.16. Результат роботи процедури.

4) Приклад 4. Процедура для зміни статусу доставки

```
CREATE PROCEDURE ChangeStatusDelivery
    @DeliveryID INT, @NewDeliveryStatus VARCHAR(75)
AS BEGIN
    UPDATE Delivery
    SET DeliveryStatus = @NewDeliveryStatus
    WHERE ID_Delivery = @DeliveryID;
END;
```


----- Перевірка роботи -----

```
EXEC ChangeStatusDelivery @DeliveryID = 2, @NewDeliveryStatus = 'Delivered'
```

```
SELECT * FROM Delivery
```

39 %

Results Messages

	ID_Delivery	ID_Order	ID_User	AddressDI	DeliveryDate	DeliveryStatus
1	1	1	1	Kyiv, St. Shevchenka, 10	2025-03-02 00:00:00.000	Shipped
2	2	2	2	Kyiv, St. Lesya, 5	2025-03-29 10:28:44.707	Delivered

Рис. 3.17. Результат роботи процедури.

3.8. Налаштування безпеки доступу до об'єктів бази даних.

Безпека бази даних є ключовим аспектом для захисту конфіденційної інформації, цілісності даних і контролю доступу до функціональності системи. У системі керування базами даних SQL Server безпека доступу реалізується шляхом призначення ролей користувачам бази даних. Один і той самий обліковий запис на сервері може мати різні ролі в різних базах даних. Кожен користувач може мати індивідуальні права, або успадковувати їх через ролі, зокрема:

- Ролі рівня бази даних (database-level roles) — використовуються для внутрішніх користувачів, які працюють безпосередньо з базою даних;
- Програмні ролі (application roles) — призначені для зовнішніх користувачів, які взаємодіють із базою даних через програму, сайт або систему звітності.

Application roles забезпечують контрольований доступ через додатки, дозволяючи визначати дозволи на рівні самого застосунку, а не конкретних користувачів. Це підвищує безпеку системи та спрощує адміністрування доступу.

Для активації програмної ролі застосовується збережена процедура `sp_setapprole`, яка приймає пароль ролі. Усі дії в межах сесії здійснюються згідно з дозволами, заданими для ролі. Програма може повернутися до попереднього контексту за допомогою `sp_setapprole`.

Оскільки ролі програми є ідентифікаторами на рівні бази даних, вони можуть отримати доступ до інших баз даних лише через дозволи, надані в цих базах даних гостю. Таким чином, будь-яка база даних, у якій було вимкнено гостьову систему, недоступна для ролей програми в інших базах даних[7].

Для кожної з цих ролей налаштовані конкретні права доступу до таблиць, подань та процедур на рівні бази даних, що гарантує мінімальні права для кожного користувача.

Подання та процедури обмежують доступу до таблиць напряму;

- агрегують або фільтрують дані відповідно до потреб користувачів;
- обмежують доступ до чутливої інформації: до електронних адрес, номерів телефонів, адрес проживання, рахунків клієнтів; особистих даних постачальників;
- дозволяють виконувати лише дозволені дії відповідно до ролі.

3.8.1. Призначення ролей на рівні бази даних (Database-level Roles).

Внутрішніми користувачами є співробітники інтернет-магазину, наприклад менеджери з продажу, аналітики, адміністратори складу, які мають постійні облікові записи в системі безпеки SQL Server. Для них створені облікові записи та призначені відповідні ролі, які визначають права доступу до об'єктів бази даних.

- 1) **Системний адміністратор** – повний доступ до всіх об'єктів бази даних (таблиць, подань, процедур). Має роль db_owner.

```
CREATE LOGIN sys_admin_user WITH PASSWORD = 'SysPa$$w0rd';
USE WebShopDB;
CREATE USER sys_admin_user FOR LOGIN sys_admin_user;
ALTER ROLE db_owner ADD MEMBER sys_admin_user;
```

- 2) **Аналітик** – роль db_datareader, має доступ до читання всіх таблиць і представлень користувачів. Об'єкти користувача можуть існувати в будь-якій схемі, крім sys і INFORMATION_SCHEMA.

```
CREATE LOGIN analyst_user WITH PASSWORD = 'AnalystPa$$w0rd';
CREATE USER analyst_user FOR LOGIN analyst_user;
ALTER ROLE db_datareader ADD MEMBER analyst_user;
```

- 3) Керівник інтернет-магазину (менеджер)** – роль db_ddladmin, може змінювати структуру об'єктів бази даних (DDL).

```
CREATE LOGIN e_manager_user WITH PASSWORD = 'ManagerPa$$w0rd';
CREATE USER e_manager_user FOR LOGIN e_manager_user;
ALTER ROLE db_ddladmin ADD MEMBER e_manager_user;
```

- 4) Бухгалтер** – стежить за витрачанням коштів, веденням документації та своєчасним складанням звітності, веде повний облік товарів. Доступ до фінансових даних та операцій зі складом.

```
CREATE LOGIN sales_user WITH PASSWORD = 'SalesPa$$w0rd';
CREATE USER sales_user FOR LOGIN sales_user;
```

```
GRANT SELECT ON PaymentSys TO sales_user;
GRANT SELECT ON Clients TO sales_user;
GRANT SELECT ON OrdersCl TO sales_user;
GRANT SELECT ON Invoice TO sales_user;
GRANT SELECT ON Products TO sales_user;
GRANT SELECT ON Warehouse TO sales_user;
```

- 5) Відділ логістики** – відповідає за керуванням відправки товарів замовникам магазину, веде комунікацію зі спеціалістами доставки та відповідними поштовими службами. Має доступ до даних клієнтів, замовлення та доставки.

```
CREATE LOGIN logistic_user WITH PASSWORD = 'LogisticPa$$w0rd';
CREATE USER logistic_user FOR LOGIN logistic_user;
```

```
GRANT SELECT ON Delivery TO logistic_user;
GRANT SELECT ON OrdersCl TO logistic_user;
GRANT SELECT ON Clients TO logistic_user;
```









Server name:	Marina\MSSSQL
View server permissions	
Database name:	WebShopDB
Users or roles:	<input type="text" value="Search..."/>
Name	Type
 analyst_user	User
 e_manager_user	User
 logistic_user	User
 Marina\marak	User
 public	Database role
 sales_user	User
 sys_admin_user	User
 USR_DDL	User

Рис.3.18. Внутрішні користувачі бази даних «Інтернет-магазин».

Таким чином, кожен користувач отримує лише ті дозволи, що відповідають його посадовим обов'язкам.

3.8.2. Створення програмних ролей (Application Roles).

Ролі були створені для зовнішніх користувачів, які працюють із базою даних через додатки або веб-інтерфейси, а саме клієнтів, постачальників та кур'єрів. Кожна application role активується через відповідний логін програми і має обмежений набір дозволів на виконання тільки тих процедур, які потрібні для її функціоналу.

- 1) **Клієнт** – доступ до реєстрації, здійснення замовлення, перегляду та оновленню особистих даних та історії замовлень, пошуку товарів.

```
CREATE APPLICATION ROLE client_app_role WITH PASSWORD = 'ClientPa$$w0rd';
```

```
GRANT EXECUTE ON pr_RegisterAsCustomer TO client_app_role;
GRANT EXECUTE ON pr_AddInvoice TO client_app_role;
GRANT EXECUTE ON pr_UpdateIfoAsCustomer TO client_app_role;
GRANT EXECUTE ON FindProductForNameBrand TO client_app_role;
GRANT EXECUTE ON pr_MakeOrder TO client_app_role;
GRANT EXECUTE ON pr_UpdateOrderQuantity TO client_app_role;
GRANT EXECUTE ON pr_ViewPersonalOrders TO client_app_role;
GRANT SELECT ON vw_Catalog TO client_app_role;
```

1. Приклад процедури реєстрації нового користувача:

```

CREATE OR ALTER PROCEDURE pr_RegisterAsCustomer
    @IPAdresse VARCHAR(50),
    @FirstName VARCHAR(50),
    @LastName VARCHAR(50),
    @AddressCl VARCHAR(50),
    @Phone VARCHAR(50),
    @Email VARCHAR(50),
    @Invoice VARCHAR(50) = NULL
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @NewUserID INT;

    SELECT @NewUserID = ID_User
    FROM Users
    WHERE IPAdresse = @IPAdresse;

    IF @NewUserID IS NULL
    BEGIN INSERT INTO Users(IPAdresse)
        VALUES(@IPAdresse);
        SET @NewUserID = SCOPE_IDENTITY();
    END

    -- Додаємо клієнта з прив'язкою до ID_User
    INSERT INTO Clients (ID_User, FirstName, LastName, AddressCl, Phone, Email,
Invoice)
    VALUES (@NewUserID, @FirstName, @LastName, @AddressCl, @Phone, @Email, @Invoice);
END;
GO

DECLARE @cookie VARBINARY(8000);
-- Вмикаємо application role
EXEC sp_setapprole
    @rolename = 'client_app_role',
    @password = 'ClientPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;

EXEC pr_RegisterAsCustomer
    @IPAdresse = '192.168.0.6',
    @FirstName = 'Oleg',
    @LastName = 'Ivanov',
    @AddressCl = 'Lviv, St. Rynok, 12',
    @Phone = '093-123-4567',
    @Email = 'oleg@gmail.com',
    @Invoice = 'INV12';
-- Вимикаємо application role
EXEC sp_unsetapprole @cookie;

```

2. Приклад процедури здійснення замовлення.

```

CREATE OR ALTER PROCEDURE pr_MakeOrder
    @IPAdresse VARCHAR(50),
    @Name VARCHAR(50) = null,
    @Brand VARCHAR(50) = null,
    @ID_Product INTEGER = null,
    @ID_Client INTEGER,
    @Quantity INTEGER
AS BEGIN
    SET NOCOUNT ON;
    DECLARE @NewUserID INT;

    SELECT @NewUserID = ID_User

```

```

FROM Users
WHERE IPAdresse = @IPAdresse;

IF @NewUserID IS NULL
BEGIN INSERT INTO Users(IPAdresse)
VALUES(@IPAdresse);
SET @NewUserID = SCOPE_IDENTITY();
END

IF @ID_Product IS NULL
BEGIN SELECT @ID_Product = ID_Product
FROM Products
WHERE Name = @Name AND Brand = @Brand;
END

INSERT INTO OrdersCl (ID_Product, ID_Client, ID_User, OrderStatus, Price,
OrderDate)
SELECT @ID_Product, @ID_Client, @NewUserID, 'Pending', Price * @Quantity,
GETDATE()
FROM Products
WHERE ID_Product = @ID_Product;
END;
GO

DECLARE @cookie VARBINARY(8000);
EXEC sp_setapprole
    @rolename = 'client_app_role',
    @password = 'ClientPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;

EXEC pr_MakeOrder
    @IPAdresse = '192.168.0.1',
    @ID_Client = 3,
    @Name = 'Laptop',
    @Brand = 'HP',
    @Quantity = 2;
-- Вимикаємо application role
EXEC sp_unsetapprole @cookie;

```

3. Приклад процедури зміни кількості товару у замовленні. Передбачено також оновлення суми замовлення.

```

CREATE OR ALTER PROCEDURE pr_UpdateOrderQuantity
    @ID_Order INTEGER,
    @ID_Product INTEGER,
    @Quantity INTEGER
AS BEGIN
SET NOCOUNT ON;
UPDATE p
    SET p.Quantity = @Quantity
    FROM Products p
    JOIN OrdersCl o ON o.ID_Product = p.ID_Product
    WHERE o.ID_Order = @ID_Order AND o.ID_Product = @ID_Product

UPDATE o
    SET o.Price = @Quantity * p.Price
    FROM OrdersCl o
    JOIN Products p ON o.ID_Product = p.ID_Product
    WHERE o.ID_Order = @ID_Order AND o.ID_Product = @ID_Product

```

```

END;

DECLARE @cookie VARBINARY(8000);
EXEC sp_setapprole
    @rolename = 'client_app_role',
    @password = 'ClientPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;

EXEC pr_UpdateOrderQuantity
    @ID_Order = 12,
    @ID_Product = 9,
    @Quantity = 1;
-- Вимикаємо application role
EXEC sp_unsetapprole @cookie;

```

5. Приклад процедури перегляду замовлення.

```

CREATE OR ALTER PROCEDURE pr_ViewPersonalOrders
    @ID_Order INTEGER
AS BEGIN
    SELECT * FROM ViewClientOrders vw
    WHERE vw.ID_Order = @ID_Order
END;
GO

DECLARE @cookie VARBINARY(8000);
EXEC sp_setapprole
    @rolename = 'client_app_role',
    @password = 'ClientPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;

EXEC pr_ViewPersonalOrders
    @ID_Order = 3;
-- Вимикаємо application role
EXEC sp_unsetapprole @cookie;

```

2) Постачальник магазину – доступ до оновлення складу, контактних даних, перегляду товарів. Процедури реєстрації нових постачальників.

```

CREATE APPLICATION ROLE supplier_app_role WITH PASSWORD = 'SupplierPa$$w0rd';

GRANT EXECUTE ON pr_RegisterAsSupplier TO supplier_app_role;
GRANT EXECUTE ON pr_UpdateInfoAsSupplier TO supplier_app_role;
GRANT EXECUTE ON pr_UpdateStock TO supplier_app_role;
GRANT EXECUTE ON pr_ProductStock TO supplier_app_role;

```

1. Приклад процедури перегляду товарів на складі.

```

CREATE OR ALTER PROCEDURE pr_ProductStock
    @ID_Supplier INTEGER
AS BEGIN
    SELECT p.Name, p.Brand, w.Quantity, w.Price
    FROM Warehouse w
    LEFT JOIN Suppliers s ON s.ID_Supplier = w.ID_Supplier
    LEFT JOIN Products p ON p.ID_Supplier = w.ID_Supplier
    WHERE @ID_Supplier = w.ID_Supplier
END;
GO

```

```

DECLARE @cookie varbinary(8000);
EXEC sp_setapprole
    @rolename = 'supplier_app_role',
    @password = 'SupplierPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;

EXEC pr_ProductStock
    @ID_Supplier = 1
EXEC sp_unsetapprole @cookie;

```

3) Кур'єр – доступ до перегляду даних про доставку та оновлення її статусу.

```

CREATE APPLICATION ROLE delivery_app_user WITH PASSWORD = 'DeliveryPa$$w0rd';

GRANT EXECUTE ON pr_UpdateDeliveryAddress TO delivery_app_user;
GRANT EXECUTE ON vw_ClientDeliveryData TO delivery_app_user;

```

1. Приклад процедур перегляду інформації для доставки та оновлення статусу.

--- 1. Подання перегляд

```

CREATE PROCEDURE vw_ClientDeliveryData
    @ID_Order INTEGER

```

AS BEGIN

```

    SELECT * FROM ViewLogisticInfo
    WHERE ID_Order = @ID_Order;

```

END;

--- 2. Процедура оновлювати (для статусу) інформацію про доставку

```

CREATE PROCEDURE pr_UpdateDeliveryAddress

```

```

    @ID_Delivery INTEGER,
    @NewStatus VARCHAR(50)

```

AS BEGIN

```

    UPDATE Delivery
    SET DeliveryStatus = @NewStatus, DeliveryDate = GETDATE()
    WHERE ID_Delivery = @ID_Delivery;

```

END;

```

DECLARE @cookie varbinary(8000);
EXEC sp_setapprole
    @rolename = 'delivery_app_user',
    @password = 'DeliveryPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;

```



```
EXEC vw_ClientDeliveryData
    @ID_Order = 4

EXEC pr_UpdateDeliveryAdress
    @ID_Delivery = 4,
    @NewStatus = 'Delivered'

EXEC vw_ClientDeliveryData
    @ID_Order = 4
EXEC sp_unsetapprole @cookie;
```

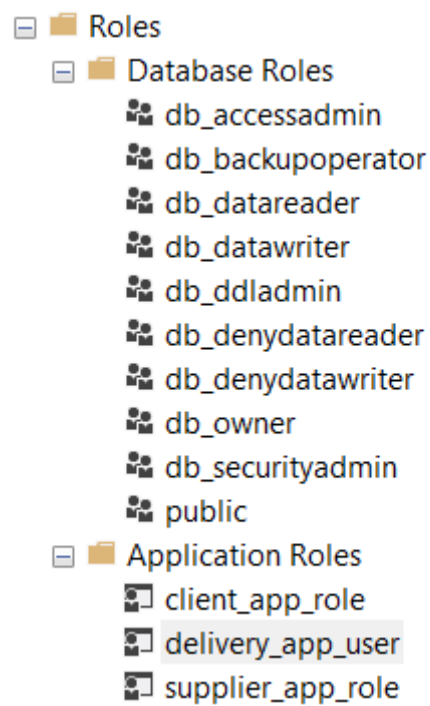


Рис.3.19. Ролі бази даних «Інтернет-магазин».

4. ПРАКТИЧНЕ ЗАСТОСУВАННЯ БАЗИ ДАНИХ

Важливим етапом життєвого циклу інформаційної системи є практичне застосування бази даних для аналітики, створення звітності та візуалізації бізнес-процесів. Побудова якісної системи звітності дозволяє приймати обґрунтовані управлінські рішення, підвищує безпеку діяльності підприємства та забезпечує ефективний контроль за ключовими показниками.

Для реалізації функцій звітності у рамках даного проєкту було використано **Microsoft Report Builder** — інструмент для створення звітів із розбивкою на сторінки, який є частиною платформи SQL Server Reporting Services (SSRS). Звіти створюються у форматі .rdl та публікуються на сервері звітів або на сайті SharePoint, після чого стають доступними для кінцевих користувачів відповідно до їх ролей і повноважень. [9].

Етапи створення звіту:

- **Визначити ціль та призначення звіту.** Добре продуманий звіт надає читачам інформацію, яка спонукає до розуміння та дій. Рішення, прийняті на цьому етапі, впливають на вибір параметрів звіту та дизайн макета звіту.
- **Вибрати тип запиту.** Визначити чи використовувати спільний запит на набір даних, чи специфічний для конкретного репорту. Спільний набір даних із узагальненим запитом легко підтримувати для використання кількома звітами, але кожен розробник звітів повинен фільтрувати дані за потреби для свого конкретного набору звітів.
- **Налаштувати права доступу до звітів.** Загальною стратегією є створення структури папок на сервері звітів і надання доступу до звітів і пов'язаних зі звітами ролей на основі елементів і захисту папок.
- **Підключитися до джерела даних та створити набори даних для побудови звіту.**
- **Форматування звітів.**
- **Збереження та публікація звіту на сервері відповідно до внутрішньої логіки підприємства.**

Архітектура SSRS є наступною [12]:

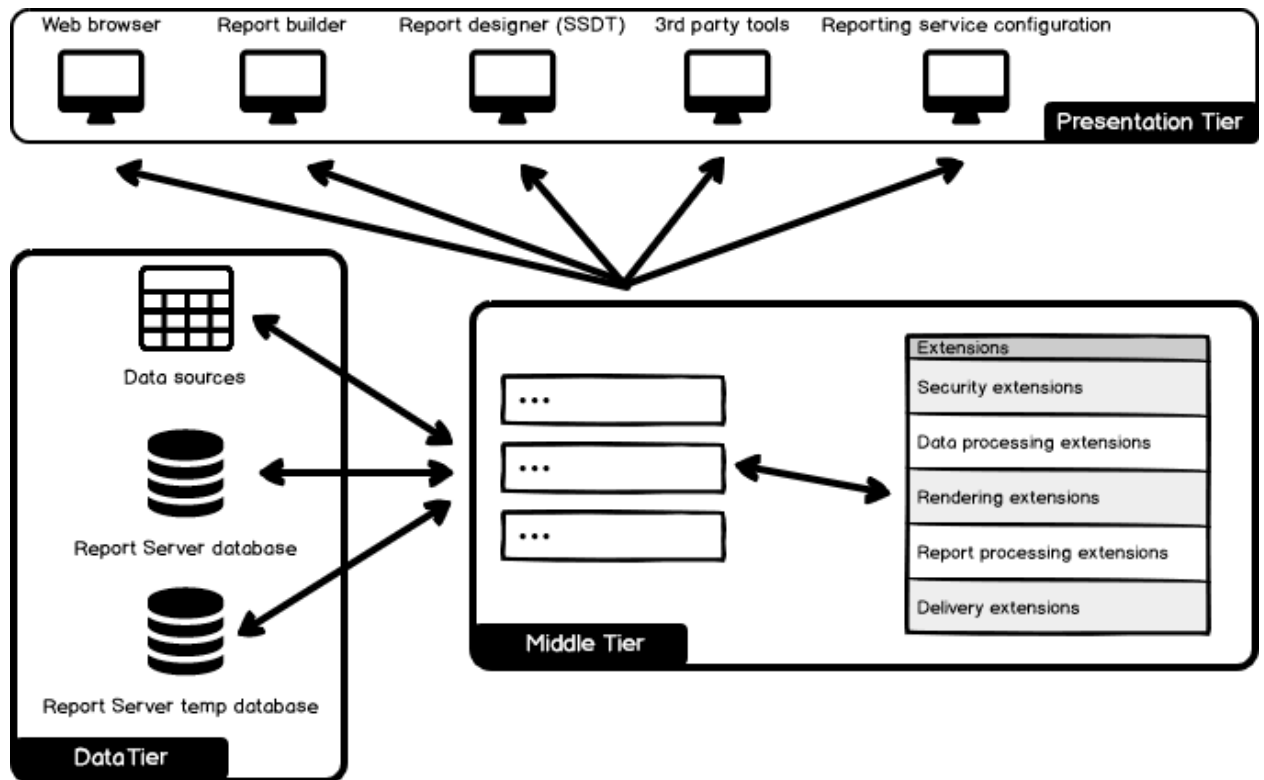


Рис. 4.1. Архітектура SQL Server Reporting Services (SSRS).

Принцип роботи SSRS [12]:

1. Користувачі звітів – це люди, які працюють з даними та хочуть отримати певну аналітичну інформацію з цих даних. Вони надсилають запит на сервер SSRS.
2. Сервер SSRS знаходить метадані звіту та надсилає запит на дані до джерела даних.
3. Дані з джерела даних об'єднуються з визначенням, тобто запитом, звіту у репорт.
4. Після створення аналітичного звіту він повертається клієнту.

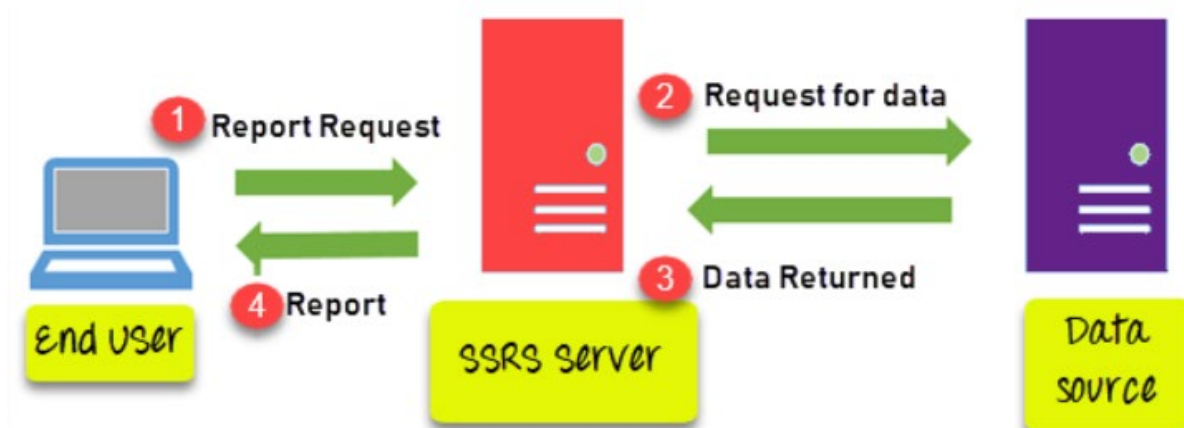


Рис. 4.2. Принцип роботи SQL Server Reporting Services (SSRS).

Кожна організація дотримується стандартного **життєвого циклу** звітності, який складається з наступних етапів[12]:

- Створення (**Authoring**): на цьому етапі автор звіту визначає макет і синтаксис даних. Звіти можна створювати за допомогою Report Builder.
- Керування (**Management**): включає управління опублікованим звітом. Автор публікує певний звіт у централізованому місці, де адміністратор звіту ретельно перевіряє його на безпеку та місце призначення. Після публікації звіту адміністратор може використовувати Report Manager, SharePoint або SQL Server Management Studio для керування цими опублікованими звітами.
- Доставка (**Delivery**): фактичний звіт розповсюджується серед цільових користувачів і також доступний у багатьох різних форматах. Звіти структуровані як окремі елементи в папці, що додатково спрощує перегляд та пришвидшує виконання.

4.1. Налаштування SQL Server Reporting Services (SSRS).

Для забезпечення функціональності звітності була встановлена та налаштована служба SSRS, включно з URL-адресами веб-порталу та веб-служби звітів. Ці адреси дозволяють швидко розгортати та використовувати інструменти звітності без додаткових кроків налаштування.

Адреса веб-порталу: <http://marina/Reports/browse/ManaderFolder>.

Після встановлення та реєстрації сервера у SQL Server Management Studio, було створено спільне джерело даних до бази «Інтернет-магазин».

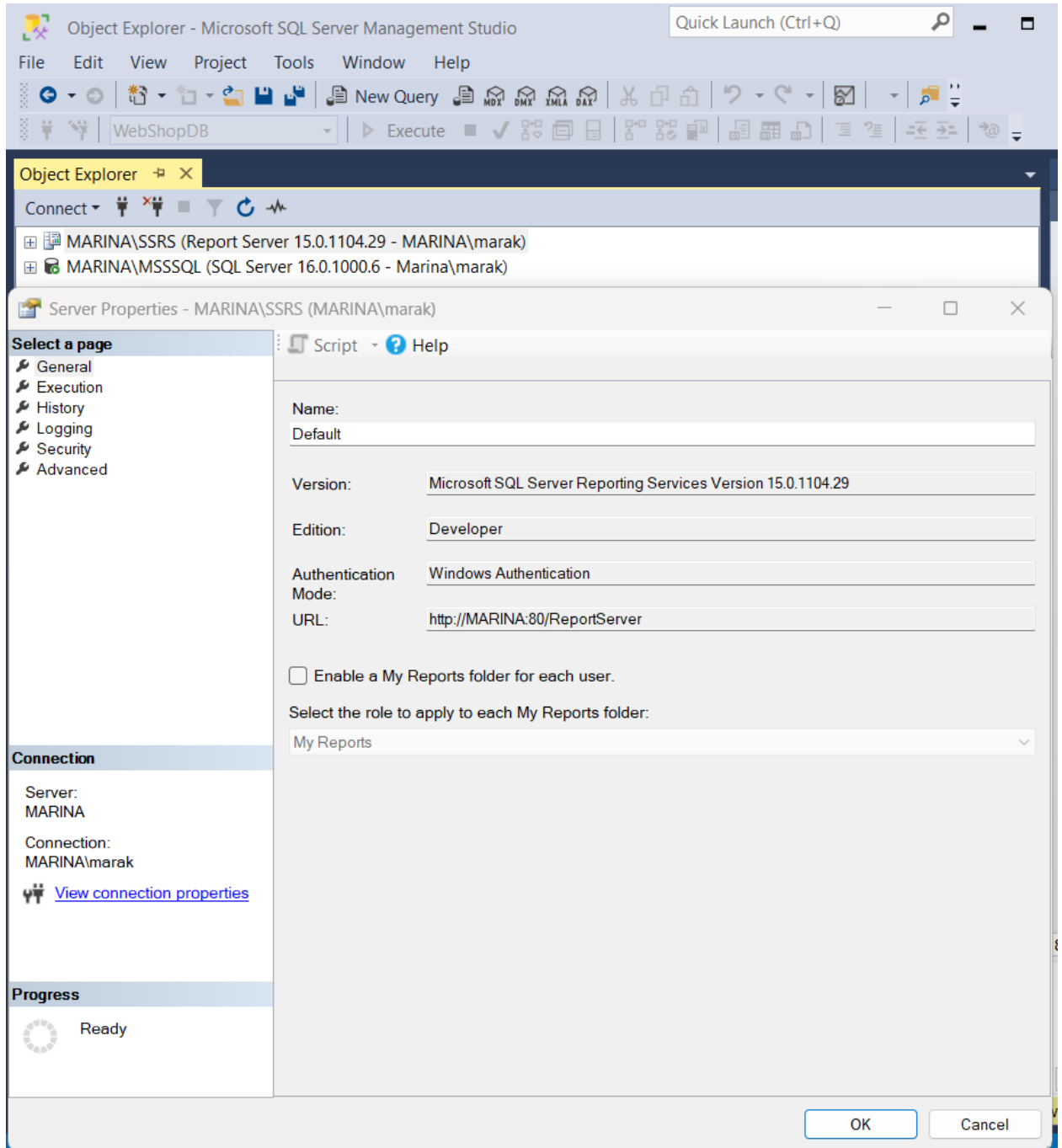


Рис. 4.3. Реєстрація сервера у SQL Server Management Studio.

Спільне джерело даних передбачає набір властивостей підключення, на який можна посилатися в кількох звітах, моделях і керованих даними підписах, які виконуються на сервері звітів Reporting Services. Після створення спільного

джерела даних на сервері звітів можна створити призначення ролей, щоб керувати доступом до нього, перемістити його в інше розташування, перейменувати або перевести в автономний режим. Перевагою створення спільного джерела даних є автоматичне оновлення інформації про шлях у всіх звітах або підписах, які на нього посилаються, у разі перейменування або переміщення спільного джерела даних в інше розташування в ієрархії папок сервера звітів.

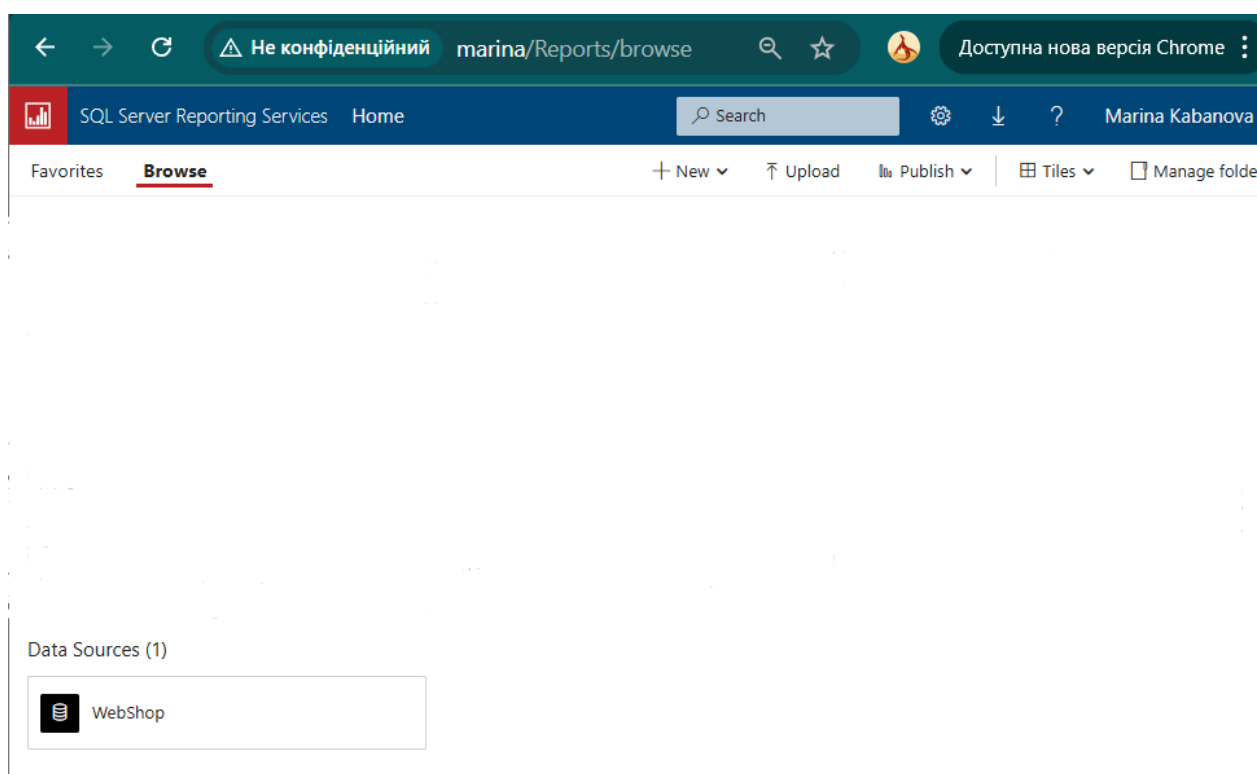


Рис.4.4. Підключення бази даних «Інтернет-магазин» для спільного використання.

4.2. Налаштування безпеки доступу та розмежування прав у SSRS.

Безпека доступу до звітів забезпечується завдяки:

- Ролям у базі даних: database-level roles та application roles.
- Системним ролям веб-порталу, які забезпечують можливість призначати ролі іншим користувачам.

Щоб додати роль необхідно ввести обліковий запис користувача або групи домену Windows у форматі: `<domain>\<account>`.

- Роль System Administrator – призначена для системного адміністратора та менеджера;
- Роль System User – для аналітика, відділу логістики та бухгалтера.
- Шляхом призначення вбудованих ролей Browser, Content Manager, Publisher, Report Builder залежно від обов’язків.

Крім цього, у SSRS були створені тематичні папки, що відповідають функціональним підрозділам магазину. Це забезпечує ефективну організацію звітів та розмежування доступу.

- ManagerFolder – для керівника інтернет-магазину.
- LogisticFolder – для відділу логістики.
- AnalystFolder – для відділу аналітики.
- SalesFolder – для відділу фінансів.

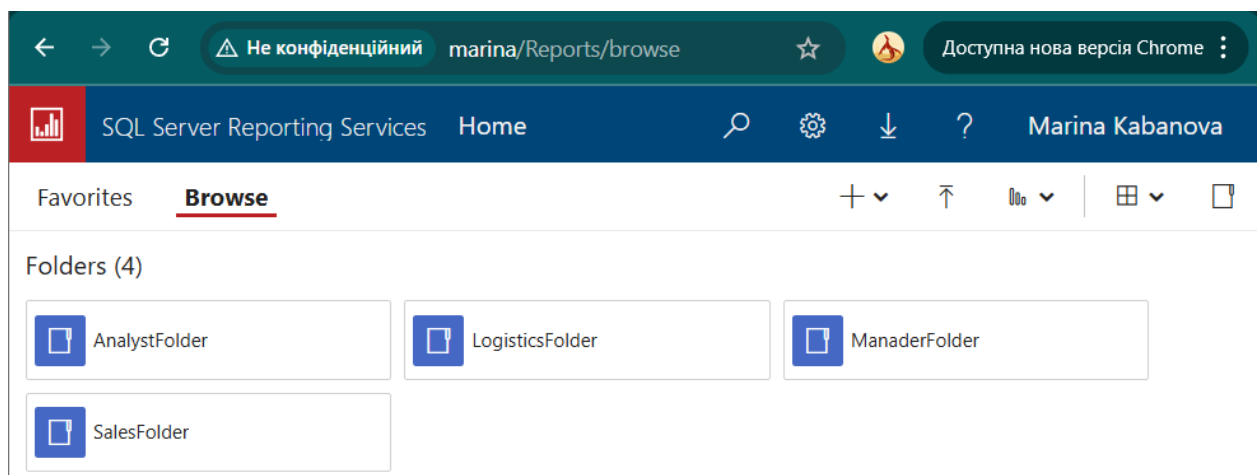


Рис.4.5. Папки на сервері звітів Reporting Services.

Користувачі звітів отримують різні рівні деталізації в залежності від призначеної їм ролі. За замовчуванням лише користувачі, які є членами локальної групи адміністраторів, можуть запускати звіти, переглядати ресурси, змінювати властивості та видаляти елементи. Для всіх інших

користувачів мають бути створені ролі, що дозволяють отримати доступ до звіту чи ресурсу.

4.3. Створення звітів на основі бази даних.

Кожен звіт створюється на основі подань та збережених процедур в SQL Server Management Studio, які відображують необхідну інформацію. Для створення звітів було здійснення підключення до спільного джерела даних «WebShop», після чого сформуванні набори даних, які потім використовувалися в дизайні звітів. Звіти зберігаються у створених папках ManagerFolder, AnalystFolder, SalesFolder, LogisticFolder, відповідно до потреб кожного підрозділу та різних сценаріїв аналітики.

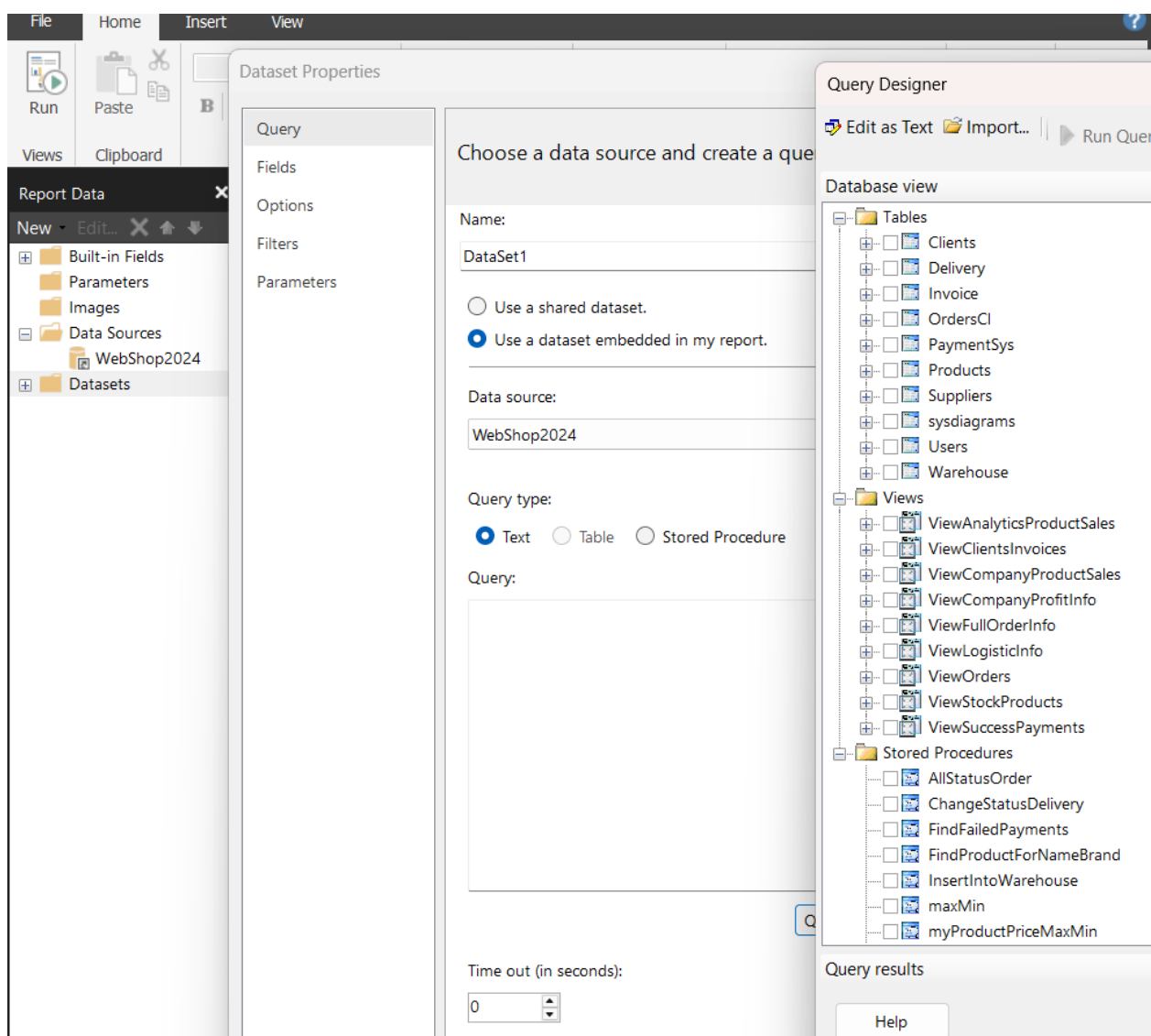


Рис.4.6. Етап створення набору даних для бази даних «Інтернет-магазин».

1) Приклад 1. Звіт «ShopOrdersReport» - замовлення клієнтів.

Звіт створений на основі подання ViewClientOrders, доступ до якого має керівник магазину та системний адміністратор, аналітик може переглядати дані. Репорт зберігається у папці менеджеру магазину.

```
GRANT SELECT, UPDATE, INSERT ON ViewClientOrders TO e_manager_user, sys_admin_user  
GRANT SELECT ON ViewClientOrders TO analyst_user
```

Крім цього, у репорті забезпечено вибір параметру ID_Order, за яким фільтрується вивід інформації. Якщо не задане значення параметру, виводиться таблиця всіх даних.

ID_Order

☒ NULL

Shop Orders

Reporting Date | 13.04.2025 9:29:53

ID	CUSTOMER		PRODUCT		STATUS		PRICE
ID Order	First Name	Last Name	Product Name	Product Brand	Order Date	Order Status	Price
1	Ivan	Petrov	Laptop	HP	01.03.2025	Completed	1000
2	Olga	Ivanova	Laptop	Apple	02.03.2025	Pending	500
3	Andriy	Kovalenko	Laptop	Lenovo	03.03.2025	Cancelled	100
4	Svitlana	Tarasenko	Monitor	HP	04.03.2025	Completed	300
5	Viktor	Melnyk	Printer	HP	05.03.2025	Pending	250
6	Natalia	Kuzmenko	Sofa	IKEA	06.03.2025	Completed	25
7	Viktor	Moroz	Sofa	IKEA	07.03.2025	Completed	15
8	Tatiana	Nikolenko	Bicycle	SportX	08.03.2025	Cancelled	10
9	Marina	Vasylenko	Helmet	SafeRide	09.03.2025	Completed	20
10	Marina	Vasylenko	Sneakers	Nike	10.03.2025	Pending	1,5
							2221,5

Рис.4.7.1. Репорт «ShopOrdersReport» про замовлення у магазині у загальному вигляді.

ID_Order

5

☐ NULL

Shop Orders

Reporting Date | 13.04.2025 9:31:43

ID	CUSTOMER		PRODUCT		STATUS		PRICE
ID Order	First Name	Last Name	Product Name	Product Brand	Order Date	Order Status	Price
5	Viktor	Melnyk	Printer	HP	05.03.2025	Pending	250
							250

Рис.4.7.2. Репорт «ShopOrdersReport» про замовлення у магазині для заданого ID.

2) Приклад 2. Звіт StockProductReport - товари на складі.

Доступ до потрібних даних забезпечує параметризована процедура FindProductForNameBrand, право модифікації якої має керівник магазину та системний адміністратор, аналітик може переглядати дані. Репорт зберігається у папці менеджера магазину.

```
GRANT EXECUTE ON FindProductForNameBrand TO e_manager_user, sys_admin_user
GRANT EXECUTE ON FindProductForNameBrand TO analyst_user
```

У репорті забезпечено вибір одного або декількох параметрів Name та Brand. Параметри передбачають можливість нульових значень.

Stock Products Info

Reporting Date | 08.04.2025 17:01:15

PRODUCT				PRICE	QUANTITY		
ID Product	Type	Brand	Company Name	Price	Shop Quantity	Stock Quantity	Total Quantity
10	Sneakers	Nike	Sports Gear	100	35	1000	1000
7	Sofa	IKEA	Furniture Ltd.	490	12	500	500
9	Helmet	SafeRide	Sports Gear	80	50	250	250
6	Sofa	IKEA	Electronics Co.	500	10	200	200
8	Bicycle	SportX	Sports Gear	250	15	150	150
3	Laptop	Lenovo	Auto Parts	900	40	100	100
4	Monitor	HP	Tech Supplies	300	25	70	70
1	Laptop	HP	Tech Supplies	1000	50	50	50
5	Printer	HP	Tech Supplies	200	20	40	40
2	Laptop	Apple	Home Goods	1200	30	30	30

Рис.4.8.1. Репорт «StockProductReport» про товар на складі у загальному вигляді.

Name ☒ NULL
Brand ☐ NULL

Stock Products Info

Reporting Date | 08.04.2025 17:05:32

PRODUCT				PRICE	QUANTITY		
ID Product	Type	Brand	Company Name	Price	Shop Quantity	Stock Quantity	Total Quantity
4	Monitor	HP	Tech Supplies	300	25	70	70
1	Laptop	HP	Tech Supplies	1000	50	50	50
5	Printer	HP	Tech Supplies	200	20	40	40

Рис.4.8.2. Репорт «StockProductReport» про товар на складі із заданим параметром.

3) Приклад 3. Звіти в папці «SalesFolder».

Подання ViewSuccessPayments слугує для виведення інформації про замовлення магазину зі статусом «Completed», ViewClientsInvoices - виведення рахунків клієнтів.

Бухгалтер, менеджер та системний адміністратор магазину мають доступ до перегляду, модифікації. Аналітик має доступ до перегляну звіту з успішними замовленнями.

```
GRANT SELECT, UPDATE ON ViewClientsInvoices TO e_manager_user, sales_user
GRANT SELECT, UPDATE ON ViewClientsInvoices TO e_manager_user, sales_user
GRANT SELECT ON ViewSuccessPayments TO analyst_user
GRANT SELECT, UPDATE ON ViewSuccessPayments TO e_manager_user, sales_user
GRANT SELECT, UPDATE, INSERT ON ViewSuccessPayments TO sys_admin_user
```

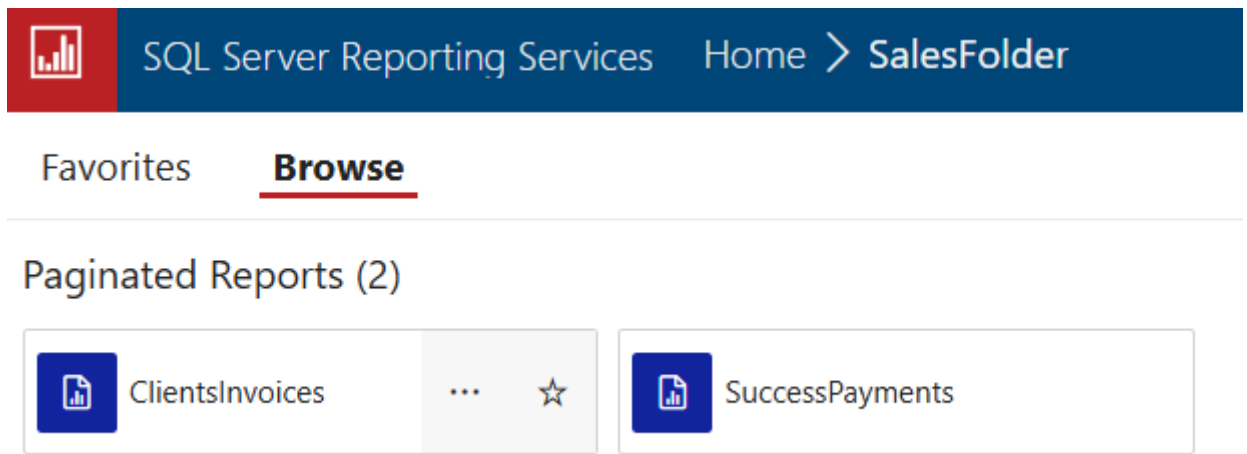


Рис.4.9. Звіти в папці «SalesFolder».

4) Приклад 4. Звіт «LogisticInfoReport» дані для доставки товару.

Відділ доставки, менеджер та системний адміністратор магазину має доступ до перегляду, модифікації подання ViewLogisticInfo, на основі якого був створений звіт.

```
GRANT SELECT, UPDATE ON ViewLogisticInfo TO e_manager_user,logistic_user
GRANT SELECT, UPDATE, INSERT ON ViewLogisticInfo TO sys_admin_user
```

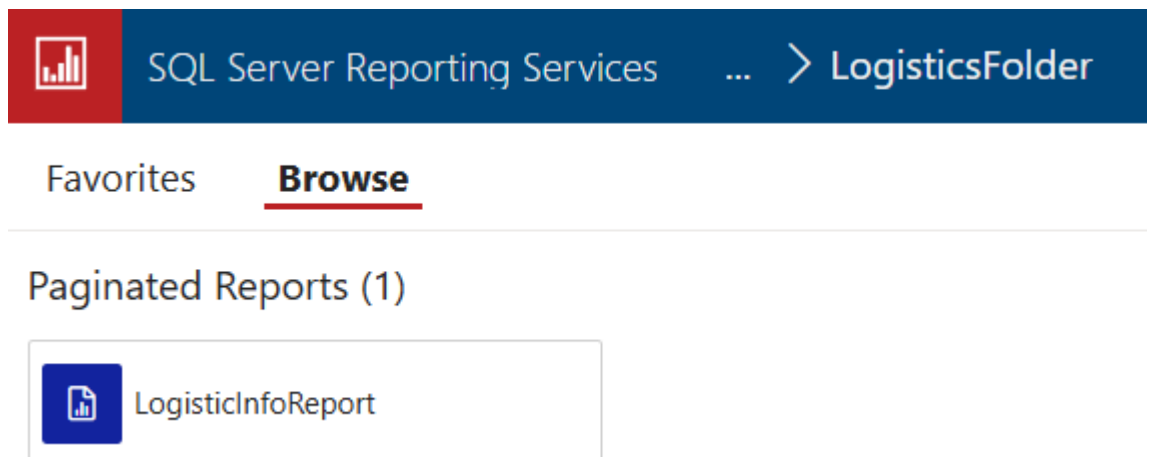


Рис.4.10. Звіти в папці «LogisticFolder».

5) Приклад 5. Звіт «CompanyMonthSalesReport».

Основою є подання ViewCompanyMonthSales, що відображає інформацію про успішні продажі магазину за місяць для кожного бренду компанії.

Доступ до перегляду мають крівник магазину та аналітик; до модифікації системний адміністратор. Розташування звіту в папці «AnalystFolder».

```
GRANT SELECT ON ViewCompanyMonthSales TO analyst_user, e_manager_user
GRANT SELECT, UPDATE, INSERT ON ViewOrders TO sys_admin_user
```

Successful Company Sales Per Month

Reporting Date | 08.04.2025 18:16:19

Brand	Product Name	Payment Month	Total Sales	Total Clients	Total Revenue
Auto Parts					
Lenovo	Laptop	03	40	1	100
Total					100
E lectronics Co.					
IKEA	Sofa	03	10	1	25
Total					25
Furniture Ltd.					
IKEA	Sofa	03	12	1	15
Total					15
Home Goods					
Apple	Laptop	03	30	1	500
Total					500
Sports Gear					
Nike	Sneakers	03	35	1	1,5
SafeRide	Helmet	03	50	1	20
SportX	Bicycle	03	15	1	10
Total					31,5

Рис.4.11. Звіт «CompanyMonthSalesReport» в папці «AnalystFolder».

6) Приклад 6. Звіт «ChangeStatusDelivery».

Базується на процедурі ChangeStatusDelivery для зміни статусу доставки для заданого замовлення. Доступ надано користувачу зовнішнього застосунку доставки:

```
GRANT EXECUTE ON ChangeStatusDelivery TO delivery_app_user ;
```

Order ID

New Delivery Status

Select the Order ID to change the Delivery Status

Changing Date | 08.04.2025 18:50:33

ID	ID Order	Address DI	Delivery Date	STATUS
1	1	Kyiv, St. Shevchenka, 10	02.03.2025	Shipped
2	2	Kyiv, St. Lesya, 5	03.03.2025	Shipped
3	3	Lviv, St. Franka, 15	04.03.2025	Cancelled
4	4	Kharkiv, St. Shevchenka, 20	05.03.2025	Cancelled
5	5	Odesa, St. Pushkina, 30	06.03.2025	Cancelled
6	6	Dnipro, St. Kotsiubynskoho, 8	07.03.2025	Shipped
7	7	Kyiv, St. Bessarabka, 10	08.03.2025	Delivered
8	8	Lviv, St. Soborna, 12	09.03.2025	Cancelled
9	9	Kharkiv, St. Kurchatova, 40	10.03.2025	Shipped
10	10	Odesa, St. Derybasivska, 5	11.03.2025	Shipped

Рис.4.12. Репорт «ChangeStatusDelivery».

7) Приклад 7. Найприбутковіші постачальники магазину.

Основою є подання ViewCompanyProductSales.

```
GRANT SELECT ON ViewCompanyProductSales TO sales_user, analyst_user,
e_manager_user;
```

Supplier sales report

Reporting Date | 13.04.2025 11:17:07

COMPANY			DATE	SALES	CLIENTS	REVENUE
Company Name	Product Name	Brand	Payment Date	Total Sales	Total Clients	Total Revenue
Sports Gear	Sneakers	Nike	11.03.2025	35	1	1,50€
Furniture Ltd.	Sofa	IKEA	08.03.2025	12	1	15,00€
Sports Gear	Helmet	SafeRide	10.03.2025	50	1	20,00€
Electronics Co.	Sofa	IKEA	07.03.2025	10	1	25,00€
Auto Parts	Laptop	Lenovo	04.03.2025	40	1	100,00€
Tech Supplies	Monitor	HP	05.03.2025	25	1	300,00€
Tech Supplies	Laptop	HP	02.03.2025	50	1	1 000,00€



Рис.4.13. Репорт «SuppliersSalesReport».

ВИСНОВКИ

У межах курсової роботи була розроблена та реалізована реляційна база даних інтернет-магазину товарів різної категорії, разом із системою звітності. Для створення бази було використано мову програмування Structured Query Language (SQL) та систему управління базами даних Microsoft SQL Server у середовищі SQL Server Management Studio 20 (SSMS). Для демонстрації практичного застосування бази даних реалізовано інтеграцію з інструментом звітності Report Builder на базі платформи Microsoft SQL Server Reporting Services (SSRS).

Структура бази даних відображає основні сутності магазину, такі як замовник, постачальник, склад, товар, замовлення, система доставки та платежу. База даних була створена з цілями полегшення процесу різного вигляду документації та відстеження бізнес-процесів, подій, пов'язаних із продажом, управлінням замовленнями та обліком фінансових операцій. Реалізація за рахунок інтуїтивно зрозумілого механізму введення, редагування та швидкого пошуку необхідних даних.

Система звітності дозволяє формувати динамічні звіти для аналізу даних та прийняття бізнес-рішень, з урахуванням рівнів доступу, що підвищує зручність використання та посилює інформаційну безпеку.

Були проведені попередній аналіз предметної області, побудовані діаграми потоків даних, проведено інфологічне та даталогічне моделювання, що дозволило подальше створення об'єктів бази даних. Були наведені приклади використання послідовностей, тригерів, процедур, подань та функцій, передбачене логування змін та підтримка актуальності даних. Також особливу увагу приділено налаштуванню безпеки доступу для взаємодії з базою, з розмежуванням прав доступу до даних через внутрішні та програмні ролі відповідно до типів користувачів.

Отримані результати підтверджують ефективність спроектованої системи та її здатність вирішувати завдання, пов'язані з автоматизацією

обліку, аналітики та надання інформації у зручному для користувача форматі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Берко А.Ю., Верес О.М. / Організація баз даних: практичний курс: Навч. посібник / За наук. ред. В.В. Пасічника. – Львів: Видавництво Національного університету "Львівська політехніка", 2003. – 152 с.
URL: https://library.kre.dp.ua/Books/2-4%20kurs/%D0%9E%D1%80%D0%B3%D0%B0%D0%BD%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F%20%D0%B1%D0%B0%D0%B7%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85/Orghanizatsiia%20baz%20danikh%20-%20A.Iu.Bierko%2CO.M.Vieries_2003.pdf.
2. Інформаційна безпека бізнесу: керівництво з вивчення дисципліни. Лекція 15. Нотація DFD. Національна бібліотека України імені Ярослава Мудрого. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%9A%D0%BE%D0%BD%D0%B4%D1%96%D1%83%D1%81%20%20%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%B2%D0%B0/page18.html
3. Організація баз даних та знань: конспект лекцій для студентів заочної форми навчання/ Укладач А.В. Неня. – Суми: Вид-во СумДУ, 2010.– 109 с.
URL: <https://essuir.sumdu.edu.ua/bitstream-download/123456789/465/1/Nenya%5b1%5d.pdf;jsessionid=E184FC7B526AE4CE006D87AA10F644C1>
4. Організація баз даних та знань: методичні вказівки до виконання лабораторних робіт №1- №3. Проектування баз даних. Робоча програма навчальної дисципліни 3-го курсу, осіннього семестру. Керівника курсу: Зінченко Артем Юрійович. Розміщення курсу: <https://classroom.google.com/c/NzExNjk0OTY4ODQz?cjc=mtv4nmj>
5. Проектування баз даних: опорний конспект лекцій. – Тернопіль: Західноукраїнський національний економічний університет Факультет комп'ютерних інформаційних технологій. 2021. – 24 с. URL: http://dSPACE.wunu.edu.ua/bitstream/316497/41356/1/FCIT_kKN_bIPZ_dPBD_LEC.pdf

6. Публікація: Microsoft SQL Server: переваги та недоліки. Огляд СУБД. «АСТВ.РУ». 2012-2024. URL: <https://astv.ru/news/materials/microsoft-sql-server-preimushhestva-i-nedostatki>.
7. Microsoft Learn / Microsoft SQL documentation / SQL Server technical documentation. Authentication Access. Article: Get started with Database Engine permissions. 03.03.2025. URL: <https://learn.microsoft.com/en-us/sql/relational-databases/security/authentication-access/getting-started-with-database-engine-permissions?view=sql-server-ver16>.
8. Radheya Zunjur. SSRS(Tutorial)-Learn SRS Step By Step. Medium. 30.10.2023. URL: <https://radheyzunjur77.medium.com/ssrs-tutorial-learn-ssrs-step-by-step-b050cd1dcf31> Radheya Zunjur
9. Microsoft Learn / Microsoft SQL documentation / What is SQL Server Reporting Services (SSRS) / Reporting Services report (SSRS). Authentication Access. Article: Get started with Database Engine permissions. 09.27.2024. URL: <https://learn.microsoft.com/en-us/sql/reporting-services/reports/reporting-services-reports-ssrs?view=sql-server-ver16>
10. Microsoft Learn / Microsoft SQL documentation / SQL Server technical documentation. Authentication Access / Database-level roles. Article: Get started with Database Engine permissions. 28.02.2025. URL: <https://learn.microsoft.com/en-us/sql/relational-databases/security/authentication-access/database-level-roles?view=sql-server-ver16>
11. Microsoft Learn / Microsoft SQL documentation / SQL Server technical documentation. Authentication Access / Application Roles. Article: Get started with Database Engine permissions. 30.06.2023. URL: <https://learn.microsoft.com/en-us/sql/relational-databases/security/authentication-access/application-roles?view=sql-server-ver16>

12. Guru99. SSRS Tutorial: What is SQL Server Reporting Services? Fiona Brown. 28.06.2024. URL: <https://www.guru99.com/ssrs-tutorial.html>

Додаток А. Лістинг реалізації структури БД і заповнення таблиць даними.

```
CREATE DATABASE WebShopDB
ON(
NAME = 'WebShop',
FILENAME = 'C:\DataBases\WebShopDB.mdf',
SIZE = 100MB,
MAXSIZE = 100MB,
FILEGROWTH = 10MB
)
LOG ON (
NAME = 'LogWebShopDB.ldf',
FILENAME = 'C:\DataBases\WebShopDB.ldf',
SIZE = 5MB,
MAXSIZE = 50MB,
FILEGROWTH = 5MB
)
COLLATE Cyrillic_General_CI_AS
```

-- Створити у базі даних таблиці. Встановити зв'язки між таблицями.
Передбачити умови цілісності посилання та створити обмеження користувача.

```
DROP TABLE IF EXISTS Invoice;
DROP TABLE IF EXISTS Warehouse;
DROP TABLE IF EXISTS PaymentSys;
DROP TABLE IF EXISTS Delivery;
DROP TABLE IF EXISTS OrdersCl;
DROP TABLE IF EXISTS Products;
DROP TABLE IF EXISTS Suppliers;
DROP TABLE IF EXISTS Clients;
```

```
DROP TABLE IF EXISTS Users;
```

```
CREATE TABLE Users (
  ID_User INTEGER IDENTITY(1,1) NOT NULL PRIMARY KEY,
  IPAdresse VARCHAR(50) UNIQUE CHECK (IPAdresse LIKE '%.%.%.%'),
  ChangeDate DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE TABLE Clients (
  ID_Client INTEGER IDENTITY(1,1) NOT NULL PRIMARY KEY,
  ID_User INTEGER NOT NULL FOREIGN KEY REFERENCES
  Users(ID_User),
  FirstName VARCHAR(50) NOT NULL,
  LastName VARCHAR(50) NOT NULL,
  AddressCl VARCHAR(50) NOT NULL,
  Phone VARCHAR(50) CHECK (Phone LIKE '[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]'),
  Email VARCHAR(50) CHECK (Email LIKE '%_@_%.__%'),
  Invoice VARCHAR(50) UNIQUE NULL,
  UNIQUE (ID_Client, Phone, Email)
);
```

```
CREATE TABLE Suppliers (
  ID_Supplier INTEGER IDENTITY(1,1) NOT NULL PRIMARY KEY,
  ID_User INTEGER NOT NULL FOREIGN KEY REFERENCES
  Users(ID_User),
  CompanyName VARCHAR(50) NOT NULL,
  AddressSp VARCHAR(50) NULL,
  Phone VARCHAR(50) UNIQUE CHECK (Phone LIKE '[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]')
```

);

```
CREATE TABLE Products (
ID_Product INTEGER IDENTITY(1,1) NOT NULL PRIMARY KEY,
ID_User  INTEGER  NOT  NULL  FOREIGN  KEY  REFERENCES
Users(ID_User),
ID_Supplier  INTEGER  NOT  NULL  FOREIGN  KEY  REFERENCES
Suppliers(ID_Supplier),
Name VARCHAR(50) NOT NULL,
Brand VARCHAR(50) NOT NULL,
Price FLOAT NOT NULL CHECK (Price >= 0),
Quantity INTEGER NOT NULL CHECK (Quantity >= 0)
);
```

```
CREATE TABLE OrdersCl (
ID_Order INTEGER IDENTITY(1,1) NOT NULL PRIMARY KEY,
ID_Product  INTEGER  NOT  NULL  FOREIGN  KEY  REFERENCES
Products(ID_Product),
ID_Client  INTEGER  NOT  NULL  FOREIGN  KEY  REFERENCES
Clients(ID_Client),
ID_User  INTEGER  NOT  NULL  FOREIGN  KEY  REFERENCES
Users(ID_User),
OrderStatus  VARCHAR(50)  CHECK  (OrderStatus  IN  ('Completed',
'Pending','Cancelled')),
Price FLOAT NOT NULL CHECK (Price >= 0),
OrderDate DATETIME NULL
);
```

```
CREATE TABLE Delivery (
ID_Delivery INTEGER IDENTITY(1,1) NOT NULL PRIMARY KEY,
```



```

ID_Order INTEGER NOT NULL UNIQUE FOREIGN KEY REFERENCES
OrdersCl(ID_Order),
ID_User  INTEGER NOT NULL FOREIGN KEY REFERENCES
Users(ID_User),
AddressDl VARCHAR(50) NOT NULL,
DeliveryDate DATETIME NULL,
DeliveryStatus VARCHAR(50) CHECK (DeliveryStatus IN ('Shipped',
'Delivered', 'Cancelled'))
);

```

```

CREATE TABLE PaymentSys (
ID_PaymentSys INTEGER IDENTITY(1,1) NOT NULL PRIMARY KEY,
ID_Order INTEGER NOT NULL UNIQUE FOREIGN KEY REFERENCES
OrdersCl(ID_Order),
ID_Client  INTEGER NOT NULL FOREIGN KEY REFERENCES
Clients(ID_Client),
ID_User  INTEGER NOT NULL FOREIGN KEY REFERENCES
Users(ID_User),
PaymentStatus VARCHAR(50) CHECK (PaymentStatus IN ('Success','Failed')),
Price FLOAT NOT NULL CHECK (Price >= 0),
PaymentDate DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
);

```

```

CREATE TABLE Warehouse (
ID_Warehouse INTEGER IDENTITY(1,1) NOT NULL PRIMARY KEY,
ID_Supplier  INTEGER NOT NULL FOREIGN KEY REFERENCES
Suppliers(ID_Supplier),
ID_Product  INTEGER NOT NULL FOREIGN KEY REFERENCES
Products(ID_Product),

```

```
ID_User  INTEGER  NOT  NULL  FOREIGN  KEY  REFERENCES
Users(ID_User),
Quantity INTEGER NOT NULL CHECK (Quantity >= 0),
Price FLOAT NOT NULL CHECK (Price >= 0)
);
```

```
CREATE TABLE Invoice (
ID_Invoice INTEGER IDENTITY(1,1) NOT NULL PRIMARY KEY,
ID_PaymentSys INTEGER NOT NULL FOREIGN KEY REFERENCES
PaymentSys(ID_PaymentSys),
ID_User INTEGER FOREIGN KEY REFERENCES Users(ID_User),
Price FLOAT NOT NULL CHECK (Price >= 0)
);
```

```
-- Вставка даних у таблицю Users
INSERT INTO Users (IPAdresse, ChangeDate)
VALUES
('192.168.0.1', '2024-03-01 16:00:00'),
('192.168.0.2', '2024-03-02 16:00:00'),
('192.168.0.3', '2024-03-03 16:00:00');
```

```
-- Вставка даних у таблицю Clients
INSERT INTO Clients (ID_User, FirstName, LastName, AddressCl, Phone,
Email, Invoice)
VALUES
(1, 'Ivan', 'Petrov', 'Kyiv, St. Shevchenka, 10', '050-123-4567',
'ivan@example.com', 'INV001'),
(2, 'Olga', 'Ivanova', 'Kyiv, St. Lesya, 5', '050-234-5678', 'olga@example.com',
'INV002'),
```

(3, 'Andriy', 'Kovalenko', 'Lviv, St. Franka, 15', '050-345-6789',
'andriy@example.com', 'INV003'),
(1, 'Svitlana', 'Tarasenko', 'Kharkiv, St. Shevchenka, 20', '050-456-7890',
'svitlana@example.com', 'INV004'),
(2, 'Viktor', 'Melnyk', 'Odesa, St. Pushkina, 30', '050-567-8901',
'dmytro@example.com', 'INV005'),
(3, 'Natalia', 'Kuzmenko', 'Dnipro, St. Kotsiubynskoho, 8', '050-678-9012',
'natalia@example.com', 'INV006'),
(1, 'Viktor', 'Moroz', 'Kyiv, St. Bessarabka, 10', '050-789-0123',
'viktor@example.com', 'INV007'),
(2, 'Tatiana', 'Nikolenko', 'Lviv, St. Soborna, 12', '050-890-1234',
'tatiana@example.com', 'INV008'),
(3, 'Marina', 'Vasylenko', 'Kharkiv, St. Kurchatova, 40', '050-901-2345',
'pavlo@example.com', 'INV009'),
(1, 'Marina', 'Vasylenko', 'Odesa, St. Derybasivska, 5', '050-012-3456',
'marina@example.com', 'INV010');

-- Вставка даних у таблицю Suppliers

INSERT INTO Suppliers (ID_User, CompanyName, AddressSp, Phone)
VALUES

(1, 'Tech Supplies', 'Kyiv, St. Bessarabka, 12', '050-111-2222'),
(2, 'Home Goods', 'Kyiv, St. Yevhena, 20', '050-222-3333'),
(3, 'Auto Parts', 'Lviv, St. Franka, 10', '050-333-4444'),
(1, 'Electronics Co.', 'Odesa, St. Soborna, 25', '050-444-5555'),
(2, 'Furniture Ltd.', 'Kharkiv, St. Kurchatova, 5', '050-555-6666'),
(3, 'Sports Gear', 'Kyiv, St. Shevchenka, 30', '050-666-7777'),
(1, 'Books & Stationery', 'Odesa, St. Shevchenka, 40', '050-777-8888'),
(2, 'Health & Beauty', 'Dnipro, St. Pivdenna, 15', '050-888-9999'),
(3, 'Supplies', 'Lviv, St. Soborna, 8', '050-999-0000'),
(1, 'Food & Drinks', 'Kharkiv, St. Soborna, 5', '050-000-1111');

-- Вставка данных у таблицу Products

-- Вставка данных у таблицу Products

```
INSERT INTO Products (ID_User, ID_Supplier, Name, Brand, Price, Quantity)
VALUES
```

```
(1, 1, 'Laptop', 'HP', 1000.0, 50),
(2, 2, 'Laptop', 'Apple', 1200.0, 30),
(3, 3, 'Laptop', 'Lenovo', 900.0, 40),
(1, 1, 'Monitor', 'HP', 300.0, 25),
(2, 1, 'Printer', 'HP', 200.0, 20),
(3, 4, 'Sofa', 'IKEA', 500.0, 10),
(1, 5, 'Sofa', 'IKEA', 490.0, 12),
(2, 6, 'Bicycle', 'SportX', 250.0, 15),
(3, 6, 'Helmet', 'SafeRide', 80.0, 50),
(1, 6, 'Sneakers', 'Nike', 100.0, 35),
(2, 7, 'Notebook', 'Moleskine', 25.0, 60),
(3, 7, 'Pen', 'Pilot', 5.0, 100),
(1, 8, 'Shampoo', 'Loreal', 10.0, 200),
(2, 8, 'Conditioner', 'Loreal', 12.0, 150),
(3, 10, 'Dog Food', 'Pedigree', 20.0, 300),
(1, 10, 'Cat Food', 'Pedigree', 22.0, 250),
(2, 9, 'Dog Food', 'HomeCo', 21.0, 100),
(3, 2, 'Washing Machine', 'HomeCo', 450.0, 8);
```

-- Вставка данных у таблицу OrdersCl

```
INSERT INTO OrdersCl (ID_Product, ID_Client, ID_User, OrderStatus, Price,
OrderDate)
```

```
VALUES
```

```
(1, 1, 1, 'Completed', 1000.0, '2025-03-01'),
```

```
(2, 2, 2, 'Pending', 500.0, '2025-03-02'),
(3, 3, 3, 'Cancelled', 100.0, '2025-03-03'),
(4, 4, 1, 'Completed', 300.0, '2025-03-04'),
(5, 5, 2, 'Pending', 250.0, '2025-03-05'),
(6, 6, 3, 'Completed', 25.0, '2025-03-06'),
(7, 7, 1, 'Completed', 15.0, '2025-03-07'),
(8, 8, 2, 'Cancelled', 10.0, '2025-03-08'),
(9, 9, 3, 'Completed', 20.0, '2025-03-09'),
(10, 10, 1, 'Pending', 1.5, '2025-03-10');
```

-- Вставка даних у таблицю Delivery

```
INSERT INTO Delivery (ID_Order, ID_User, AddressDl, DeliveryDate,
DeliveryStatus)
```

```
VALUES
```

```
(1, 1, 'Kyiv, St. Shevchenka, 10', '2025-03-02', 'Shipped'),
(2, 2, 'Kyiv, St. Lesya, 5', '2025-03-03', 'Shipped'),
(3, 3, 'Lviv, St. Franka, 15', '2025-03-04', 'Cancelled'),
(4, 1, 'Kharkiv, St. Shevchenka, 20', '2025-03-05', 'Delivered'),
(5, 2, 'Odesa, St. Pushkina, 30', '2025-03-06', 'Shipped'),
(6, 3, 'Dnipro, St. Kotsiubynskoho, 8', '2025-03-07', 'Shipped'),
(7, 1, 'Kyiv, St. Bessarabka, 10', '2025-03-08', 'Delivered'),
(8, 2, 'Lviv, St. Soborna, 12', '2025-03-09', 'Cancelled'),
(9, 3, 'Kharkiv, St. Kurchatova, 40', '2025-03-10', 'Shipped'),
(10, 1, 'Odesa, St. Derybasivska, 5', '2025-03-11', 'Shipped');
```

-- Вставка даних у таблицю PaymentSys

```
INSERT INTO PaymentSys (ID_Order, ID_Client, ID_User, PaymentStatus,
Price, PaymentDate)
```

VALUES

```
(1, 1, 1, 'Success', 1000.0, '2025-03-02'),
(2, 2, 2, 'Failed', 500.0, '2025-03-03'),
(3, 3, 3, 'Success', 100.0, '2025-03-04'),
(4, 4, 1, 'Success', 300.0, '2025-03-05'),
(5, 5, 2, 'Failed', 250.0, '2025-03-06'),
(6, 6, 3, 'Success', 25.0, '2025-03-07'),
(7, 7, 1, 'Success', 15.0, '2025-03-08'),
(8, 8, 2, 'Failed', 10.0, '2025-03-09'),
(9, 9, 3, 'Success', 20.0, '2025-03-10'),
(10, 10, 1, 'Success', 1.5, '2025-03-11');
```

-- Вставка даних у таблицю Warehouse

```
INSERT INTO Warehouse (ID_Supplier, ID_Product, ID_User, Quantity, Price)
```

VALUES

```
(1, 1, 1, 50, 1000.0),
(2, 2, 2, 30, 500.0),
(3, 3, 3, 100, 100.0),
(4, 4, 1, 70, 300.0),
(5, 5, 2, 40, 250.0),
(6, 6, 3, 200, 25.0),
(7, 7, 1, 500, 15.0),
(8, 8, 2, 150, 10.0),
(9, 9, 3, 250, 20.0),
(10, 10, 1, 1000, 1.5);
```

-- Вставка даних у таблицю Invoice

```
INSERT INTO Invoice(ID_PaymentSys, ID_User, Price)
```

VALUES

(1, 1, 1000.0),
(2, 2, 500.0),
(3, 3, 100.0),
(4, 1, 300.0),
(5, 2, 250.0),
(6, 3, 25.0),
(7, 1, 15.0),
(8, 2, 10.0),
(9, 3, 20.0),
(10, 1, 1.5);

SELECT * FROM Users;
SELECT * FROM Products;
SELECT * FROM OrdersCl;
SELECT * FROM Clients;
SELECT * FROM Suppliers;
SELECT * FROM Delivery;
SELECT * FROM PaymentSys;
SELECT * FROM Warehouse;
SELECT * FROM Invoice;

Додаток Б. Лістинг скриптів до створення об'єктів БД (схеми, тригерів, індексів, представлень, збережених процедур, користувацьких функцій тощо).

```
=====
```

```
-- TRIGGER --
```

```
=====
```

--- 1) Приклад 1. Оновлення дати у таблиці Users при будь-яких змін у системі

```
CREATE TRIGGER trg_changes_monitoring_products
```

```
ON Products
```

```
AFTER UPDATE
```

```
AS
```

```
BEGIN
```

```
    SET NOCOUNT ON;
```

```
    UPDATE Users
```

```
    SET ChangeDate = GETDATE()
```

```
    FROM Users u
```

```
    INNER JOIN inserted i ON u.ID_User = i.ID_User;
```

```
END;
```

```
GO
```

```
CREATE TRIGGER trg_changes_monitoring_orders
```

```
ON OrdersCl
```

```
AFTER UPDATE
```

```
AS
```

```
BEGIN
```

```
    SET NOCOUNT ON;
```

```
    UPDATE Users
```

```
    SET ChangeDate = GETDATE()
```



```
FROM Users u
  INNER JOIN inserted i ON u.ID_User = i.ID_User;
END;
GO
```

```
CREATE TRIGGER trg_changes_monitoring_clients
ON Clients
AFTER UPDATE
AS
BEGIN
  SET NOCOUNT ON;
```

```
  UPDATE Users
  SET ChangeDate = GETDATE()
  FROM Users u
  INNER JOIN inserted i ON u.ID_User = i.ID_User;
END;
GO
```

```
CREATE TRIGGER trg_changes_monitoring_suppliers
ON Suppliers
AFTER UPDATE
AS
BEGIN
  SET NOCOUNT ON;
```

```
  UPDATE Users
  SET ChangeDate = GETDATE()
  FROM Users u
  INNER JOIN inserted i ON u.ID_User = i.ID_User;
```

END;

GO

CREATE TRIGGER trg_changes_monitoring_delivery

ON Delivery

AFTER UPDATE

AS

BEGIN

 SET NOCOUNT ON;

 UPDATE Users

 SET ChangeDate = GETDATE()

 FROM Users u

 INNER JOIN inserted i ON u.ID_User = i.ID_User;

END;

GO

CREATE TRIGGER trg_changes_monitoring_paymentSys

ON PaymentSys

AFTER UPDATE

AS

BEGIN

 SET NOCOUNT ON;

 UPDATE Users

 SET ChangeDate = GETDATE()

 FROM Users u

 INNER JOIN inserted i ON u.ID_User = i.ID_User;

END;

GO

```

CREATE TRIGGER trg_changes_monitoring_warehouse
ON Warehouse
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE Users
    SET ChangeDate = GETDATE()
    FROM Users u
    INNER JOIN inserted i ON u.ID_User = i.ID_User;
END;
GO

```

```

CREATE TRIGGER trg_changes_monitoring_invoice
ON Invoice
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE Users
    SET ChangeDate = GETDATE()
    FROM Users u
    INNER JOIN inserted i ON u.ID_User = i.ID_User;
END;
GO

```

```

----- Перевірка роботи триггеру -----
SELECT * FROM Products WHERE ID_Product = 1;

```

```
SELECT * FROM Users WHERE ID_User = 1;
```

```
UPDATE Products
```

```
SET Price = 1500
```

```
WHERE ID_Product = 1;
```

```
SELECT * FROM Products WHERE ID_Product = 1;
```

```
SELECT * FROM Users WHERE ID_User = 1;
```

```
-----
```

--- 2)Приклад 2. Оновлення статусу замовлення після здійснення платежу ---

-

```
CREATE TRIGGER trg_order_status_after_payment
```

```
ON PaymentSys
```

```
AFTER INSERT, UPDATE
```

```
AS
```

```
BEGIN
```

```
    SET NOCOUNT ON;
```

```
    UPDATE OrdersCl
```

```
    SET OrderStatus = 'Pending'
```

```
    FROM OrdersCl o
```

```
    INNER JOIN inserted i ON o.ID_Order = i.ID_Order
```

```
    WHERE i.PaymentStatus = 'Success';
```

```
    UPDATE OrdersCl
```

```
    SET OrderStatus = 'Cancelled'
```

```
    FROM OrdersCl o
```

```
    INNER JOIN inserted i ON o.ID_Order = i.ID_Order
```

```
    WHERE i.PaymentStatus = 'Failed';
```

END;

GO

----- Перевірка роботи триггеру -----

SELECT * FROM OrdersCl WHERE ID_Client IN (1, 2);

SELECT * FROM PaymentSys WHERE ID_PaymentSys IN (1, 2)

UPDATE PaymentSys

SET PaymentStatus = 'Failed'

WHERE ID_PaymentSys = 1;

UPDATE PaymentSys

SET PaymentStatus = 'Success'

WHERE ID_PaymentSys = 2;

SELECT * FROM OrdersCl WHERE ID_Client IN (1, 2);

SELECT * FROM PaymentSys WHERE ID_PaymentSys IN (1, 2)

--- 3)Приклад 3. Оновлення статусу доставки після здійснення замовлення --

-

CREATE TRIGGER trg_delivery_order_status

ON Delivery

AFTER INSERT, UPDATE

AS

BEGIN

SET NOCOUNT ON;

UPDATE Delivery

SET DeliveryStatus = 'Shipped'

```

FROM Delivery d
JOIN OrdersCl o ON d.ID_Order = o.ID_Order
WHERE o.OrderStatus = 'Pending';

UPDATE OrdersCl
SET OrderStatus = 'Completed'
FROM OrdersCl o
JOIN inserted i ON o.ID_Order = i.ID_Order
WHERE i.DeliveryStatus = 'Delivered';

UPDATE OrdersCl
SET OrderStatus = 'Cancelled'
FROM OrdersCl o
JOIN inserted i ON o.ID_Order = i.ID_Order
WHERE i.DeliveryStatus = 'Cancelled';
END;
GO

----- Перевірка роботи триггеру -----
SELECT o.ID_Client, d.ID_Delivery, o.OrderStatus, d.DeliveryStatus
FROM OrdersCl o
JOIN Delivery d
ON d.ID_Order = o.ID_Order
WHERE ID_Delivery IN (3, 5, 6) AND ID_Client IN (3, 5, 6)

UPDATE OrdersCl
SET OrderStatus = 'Pending'
WHERE ID_Client = 3

UPDATE Delivery

```

```
SET DeliveryStatus = 'Delivered'
WHERE ID_Delivery = 5;
```

```
UPDATE Delivery
SET DeliveryStatus = 'Cancelled'
WHERE ID_Delivery = 6;
```

```
SELECT o.ID_Client, d.ID_Delivery, o.OrderStatus, d.DeliveryStatus
FROM OrdersCl o
JOIN Delivery d
ON d.ID_Order = o.ID_Order
WHERE ID_Delivery IN (3, 5, 6) AND ID_Client IN (3, 5, 6)
```

--- 4) Приклад 4. Оновлення кількості товарів на складі після видалення клієнтом замовлення або зміни кількості товарів ---

```
CREATE TRIGGER trg_update_warehouse_after_products_modification
ON Products
AFTER DELETE, UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE w
        SET w.Quantity = w.Quantity + d.Quantity
    FROM Warehouse w
        JOIN DELETED d ON w.ID_Product = d.ID_Product;

    UPDATE w
        SET w.Quantity = w.Quantity - i.Quantity
    FROM Warehouse w
```

```

JOIN DELETED d ON w.ID_Product = d.ID_Product
JOIN INSERTED i ON d.ID_Product = i.ID_Product;
END;

```

----- Перевірка роботи триггеру -----

```

SELECT w.ID_Warehouse, p.ID_Product, p.Quantity AS ProductsQuantity,
w.Quantity AS WarehouseQuantity
FROM Warehouse w
JOIN Suppliers s ON s.ID_Supplier=w.ID_Supplier
JOIN Products p ON p.ID_Supplier = s.ID_Supplier
WHERE p.ID_Product =1 AND w.ID_Warehouse =1

```

```

UPDATE Products
SET Quantity = 35
WHERE ID_Product = 1

```

```

SELECT w.ID_Warehouse, p.ID_Product, p.Quantity AS ProductsQuantity,
w.Quantity AS WarehouseQuantity
FROM Warehouse w
JOIN Suppliers s ON s.ID_Supplier= w.ID_Supplier
JOIN Products p ON p.ID_Supplier = s.ID_Supplier
WHERE p.ID_Product =1 AND w.ID_Warehouse = 1

```

--

```

=====
=====

```

-- CURSOR --


```
--
=====

=====

---- 1) Приклад 1. Перевірка наявності товару перед обробкою замовлення --
--

DECLARE @OrderID INT, @ProductID INT, @RequiredQuantity INT,
@StockQuantity INT;

DECLARE order_cursor CURSOR FOR
SELECT o.ID_Order, o.ID_Product, p.Quantity
FROM OrdersCl o
JOIN Products p ON o.ID_Product = p.ID_Product
WHERE o.OrderStatus = 'Pending';
OPEN order_cursor;
FETCH NEXT FROM order_cursor INTO @OrderID, @ProductID,
@RequiredQuantity;
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Перевірка кількості товару на складі
    SELECT @StockQuantity = Quantity
    FROM Warehouse
    WHERE ID_Product = @ProductID;
    IF @StockQuantity >= @RequiredQuantity
    BEGIN
        -- Оновлення статусу замовлення
        UPDATE OrdersCl
        SET OrderStatus = 'Completed'
        WHERE ID_Order = @OrderID;
        -- Зменшення кількості товару на складі
        UPDATE Warehouse
```

```

SET Quantity = Quantity - @RequiredQuantity
WHERE ID_Product = @ProductID;
END
FETCH NEXT FROM order_cursor INTO @OrderID, @ProductID,
@RequiredQuantity;
END;
CLOSE order_cursor;
DEALLOCATE order_cursor;

```

---- 2) Приклад 2. Загальний процес доставки замовлення ----

```

DECLARE @OrdersID INT, @DeliveryID INT, @ClientID INT,
@AddressDelivery VARCHAR(100), @AddressClient VARCHAR(100);

```

```

DECLARE delivery_cursor CURSOR FOR
SELECT o.ID_Order, d.ID_Delivery, c.ID_Client, d.AddressDl, c.AddressCl
FROM OrdersCl o
JOIN Delivery d ON o.ID_Order = d.ID_Order
JOIN PaymentSys p ON o.ID_Order = p.ID_Order
JOIN Clients c ON o.ID_Client = c.ID_Client
WHERE p.PaymentStatus = 'Success';

```

```

FETCH NEXT FROM delivery_cursor INTO @OrdersID, @DeliveryID,
@ClientID, @AddressDelivery, @AddressClient

```

```

WHILE @@FETCH_STATUS = 0
BEGIN

```

```

-- Після успішної оплати оновлення статусу замовлення

```

```

UPDATE OrdersCl
SET OrderStatus = 'Pending'
WHERE ID_Order = @OrdersID;

-- Перевірка наявності адреси покупця та відповідності до адреси доставки
IF @AddressDelivery IS NULL AND @AddressClient IS NOT NULL
BEGIN
    UPDATE Delivery
    SET AddressDl = @AddressClient
    WHERE ID_Delivery = @DeliveryID;

    SET @AddressDelivery = @AddressClient;
END

-- Оновлення статусу доставки та встановлення приблизної дати
отримання
IF @AddressDelivery IS NOT NULL
BEGIN
    UPDATE Delivery
    SET DeliveryStatus = 'Shipped',
        DeliveryDate = DATEADD(DAY, 5, GETDATE())
    WHERE ID_Delivery = @DeliveryID;
END

FETCH NEXT FROM delivery_cursor INTO @OrdersID, @DeliveryID,
@ClientID, @AddressDelivery, @AddressClient;
END;

CLOSE delivery_cursor;
DEALLOCATE delivery_cursor;

```

---- 3) Приклад 3. Видача рахунку покупцю після успішної оплати ---

DECLARE @PaymentID INT, @OrderID INT, @ID_User INT, @Price FLOAT

DECLARE invoice_cursor CURSOR FOR

SELECT ID_PaymentSys, ID_Order, ID_User, Price

FROM PaymentSys

WHERE PaymentStatus = 'Success'

 AND ID_PaymentSys NOT IN (SELECT ID_PaymentSys

 FROM Invoice)

OPEN invoice_cursor;

FETCH NEXT FROM invoice_cursor INTO @PaymentID, @OrderID,

@ID_User, @Price;

WHILE @@FETCH_STATUS = 0

BEGIN

 -- Створення рахунку

 INSERT INTO Invoice(ID_PaymentSys, ID_User, Price)

 VALUES (@PaymentID, @ID_User, @Price);

END;

CLOSE invoice_cursor;

DEALLOCATE invoice_cursor;

GO

=====

-- FUNCTION --

=====

---- 1) Приклад 1. Підрахунок загальної кількості товарів на складі для певних постачальників ----

```
CREATE      FUNCTION      fn_TotalWarehouse(@SupplierCompanyName
NVARCHAR(255))
RETURNS INT
AS
BEGIN
    DECLARE @TotalStock INT;

    SELECT @TotalStock = SUM(Quantity)
        FROM Warehouse w
        JOIN Suppliers s ON w.ID_Supplier = s.ID_Supplier
        WHERE s.CompanyName = @SupplierCompanyName;

    RETURN @TotalStock;
END;
```

----- Перевірка роботи -----

```
SELECT dbo.fn_TotalWarehouse('Tech Supplies') AS TotalStockForCompany,
FROM ;
```

---- 2) Приклад 2. Інформація про замовлення клієнта для заданої компанії або назви бренду ----

```
CREATE                                          FUNCTION
fn_TotalOrdersPriceByCompanyORBrand(@BrandOrCompanyName
VARCHAR(255))
RETURNS TABLE
AS
RETURN (
```

```

SELECT
    c.ID_Client,
    c.FirstName,
    c.LastName,
    SUM(o.Price) AS TotalOrder
FROM Clients c
JOIN OrdersCl o ON c.ID_Client = o.ID_Client
JOIN Products p ON p.ID_Product = o.ID_Product
JOIN Suppliers s ON s.ID_Supplier = p.ID_Supplier
WHERE s.CompanyName = @BrandOrCompanyName OR p.Brand =
@BrandOrCompanyName
GROUP BY c.ID_Client, c.FirstName, c.LastName
);

```

----- Перевірка роботи -----

```

SELECT * FROM dbo.fn_TotalOrdersPriceByCompanyORBrand('Nike')
UNION
SELECT * FROM dbo.fn_TotalOrdersPriceByCompanyORBrand('Tech
Supplies')

```

=====

-- PROCEDURE --

=====

```

CREATE OR ALTER PROCEDURE myProductPriceMaxMin
    @Name VARCHAR(75) = null,
    @Price FLOAT = null
AS

```

BEGIN

```
SELECT MaxProduct.Name AS MaxProductName,
       MaxProduct.Brand AS MaxProductBrand,
       MaxProduct.Price AS MaxProductPrice,
       MaxProduct.Quantity AS MaxProductQuantity,
       MinProduct.Name AS MinProductName,
       MaxProduct.Brand AS MinProductBrand,
       MinProduct.Price AS MinProductPrice,
       MinProduct.Quantity AS MinProductQuantity
```

```
FROM (SELECT TOP 1 Name, Brand, Price, Quantity
      FROM Products
```

```
      WHERE Name LIKE @Name AND Price <= @Price
      ORDER BY Price DESC) AS MaxProduct
```

```
CROSS JOIN( SELECT TOP 1 Name, Brand, Price, Quantity
            FROM Products
```

```
            WHERE Name LIKE @Name AND Price <= @Price
            ORDER BY Price ASC) AS MinProduct
```

END;

GO

-- Виконання процедури

```
DECLARE @retminbrand VARCHAR(75), @retmin FLOAT, @retmax FLOAT,
        @retmaxbrand VARCHAR(75), @retrows INT;
```

EXEC myProductPriceMaxMin

```
@Name = 'Sofa',
```

```
@Price = 500
```

-- Вивід результату

```
SELECT * FROM Products
```

--- 2) Приклад 2. Інформація про товар за заданою брендом та типом.
Виведення різних постачальників такого товару---

```
CREATE OR ALTER PROCEDURE FindProductForNameBrand
    @Name VARCHAR(75) = null, @Brand VARCHAR(75) = null
AS
BEGIN
    SELECT w.ID_Product, p.Name, p.Brand, s.CompanyName, p.Price,
           p.Quantity AS ShopQuantity, w.Quantity AS StockQuantity,
SUM(w.Quantity) AS Total_Quantity
    FROM Warehouse w
    JOIN OrdersCl o ON w.ID_Product = o.ID_Product
    LEFT JOIN Products p ON w.ID_Product = p.ID_Product
    JOIN Suppliers s ON s.ID_Supplier = p.ID_Supplier
    WHERE ((@Name IS NULL OR LOWER(p.Name) LIKE LOWER('%' +
@Name + '%'))
        AND
        (@Brand IS NULL OR LOWER(p.Brand) LIKE LOWER('%' + @Brand
+ '%'))))
    GROUP BY w.ID_Product, p.Name, p.Brand, s.CompanyName, p.Price,
           p.Quantity, w.Quantity
    ORDER BY Total_Quantity DESC
END;
GO
```

----- Перевірка роботи -----

```
EXEC FindProductForNameBrand @Name = 'Laptop', @Brand = 'HP'
```

--- 3) Приклад 3. Інформація про замовлення за заданим статусом оплати ---

```
CREATE OR ALTER PROCEDURE FindPayments
```

```
  @PaymentStatus VARCHAR(75)
```

```
AS
```

```
BEGIN
```

```
  SELECT ps.ID_PaymentSys, o.ID_Order,
         c.ID_Client, c.FirstName, c.LastName, c.Email, c.Phone,
         c.Invoice, ps.Price, ps.PaymentDate
```

```
  FROM PaymentSys ps
```

```
    JOIN OrdersCl o ON o.ID_Order = ps.ID_Order
```

```
    JOIN Clients c ON c.ID_Client = o.ID_Client
```

```
    JOIN Invoice i ON i.ID_PaymentSys = ps.ID_PaymentSys
```

```
    JOIN Products p ON p.ID_Product = o.ID_Product
```

```
  WHERE PaymentStatus LIKE @PaymentStatus
```

```
  ORDER BY ps.Price DESC
```

```
END;
```

```
GO
```

----- Перевірка роботи -----

```
EXEC FindFailedPayments @PaymentStatus = 'Failed'
```

```
UNION
```

```
EXEC FindFailedPayments @PaymentStatus = 'Success'
```

--- 4) Приклад 4. Повна інформація про замовлення за заданими статусами -

--

```
CREATE OR ALTER PROCEDURE AllStatusOrder
```

```
  @PaymentStatus VARCHAR(75) = null,
```

```
  @OrderStatus VARCHAR(75) = null,
```

```
  @DeliveryStatus VARCHAR(75) = null
```

```
AS
```

```
BEGIN
```

```

SELECT c.LastName, c.FirstName, c.Email, c.Phone, c.AddressCl,
p.Name AS ProductName, p.Brand AS ProductBrand, p.Quantity,
c.Invoice, i.Price AS InvoiceSum, convert(varchar, o.OrderDate, 4)
AS OrderDate, convert(varchar, ps.PaymentDate, 4) AS PaymentDate,
o.OrderStatus, ps.PaymentStatus, d.DeliveryStatus
FROM PaymentSys ps
LEFT JOIN OrdersCl o ON o.ID_Order = ps.ID_Order
LEFT JOIN Clients c ON c.ID_Client = o.ID_Client
LEFT JOIN Invoice i ON i.ID_PaymentSys = ps.ID_PaymentSys
LEFT JOIN Products p ON p.ID_Product = o.ID_Product
LEFT JOIN Delivery d ON o.ID_Order = d.ID_Order
WHERE ((PaymentStatus LIKE @PaymentStatus) OR (@PaymentStatus IS
NULL)
AND (OrderStatus LIKE @OrderStatus) OR (@OrderStatus IS NULL)
AND (DeliveryStatus LIKE @DeliveryStatus) OR (@DeliveryStatus
IS NULL));
END;
GO

```

--- 5) Приклад 5. Процедура для зміни статусу доставки

```
ALTER PROCEDURE ChangeStatusDelivery
```

```
    @OrderID INT, @NewDeliveryStatus VARCHAR(75)
```

```
AS BEGIN
```

```
    UPDATE Delivery
```

```
        SET DeliveryStatus = @NewDeliveryStatus
```

```
        WHERE ID_Order = @OrderID;
```

```
END;
```

```
GO
```

----- Перевірка роботи -----

```
EXEC ChangeStatusDelivery @OrderID = 2, @NewDeliveryStatus = 'Shipped'
```

```
SELECT * FROM Delivery
```

```
-----
```

```
-- 5) Процедура для додавання нового рядка в таблиці складу Warehouse
```

```
CREATE PROCEDURE InsertIntoWarehouse
```

```
    @NewIDSupplier INT, @NewIDProduct INT, @NewIDUser INT,  
    @NewIDQuantity INT, @NewPrice FLOAT
```

```
AS BEGIN
```

```
    INSERT INTO Warehouse (ID_Supplier, ID_Product, ID_User, Quantity,  
Price)
```

```
        VALUES (@NewIDSupplier, @NewIDProduct, @NewIDUser,  
@NewIDQuantity, @NewPrice);
```

```
    END;
```

```
GO
```

```
----- Перевірка роботи -----
```

```
SELECT * FROM Warehouse
```

```
EXEC InsertIntoWarehouse 10, 7, 9, 57, 1000.0
```

```
SELECT * FROM Warehouse
```

```
=====
```

```
-- VIEW --
```

```
=====
```

```
-- 1) Інформація про продажі товарів
```

```
CREATE VIEW ViewProductSalesSummary AS
```

```
SELECT
```

```
    P.Name AS ProductName,
```

```
    P.Brand,
```

```

SUM(p.Quantity) AS TotalQuantitySold,
SUM(p.Price * p.Quantity) AS TotalRevenue
FROM Products p
JOIN OrdersCl o ON p.ID_Product = o.ID_Product
GROUP BY P.Name, P.Brand;

```

-- 2) Чек замовлення клієнтів

```

CREATE VIEW ViewInvoicet AS
SELECT
    I.ID_Invoice,
    C.LastName + ' ' + C.FirstName AS ClientName,
    PS.Price AS PaymentAmount,
    PS.PaymentStatus,
    PS.PaymentDate
FROM Invoice I
JOIN PaymentSys PS ON I.ID_PaymentSys = PS.ID_PaymentSys
JOIN Clients C ON PS.ID_Client = C.ID_Client;

```

-- 3) Історія замовлення клієнтів

```

ALTER VIEW ViewClientOrders AS
SELECT
    o.ID_Order, C.LastName + ' ' + C.FirstName AS ClientName,
    P.Name AS ProductName, P.Brand AS ProductBrand,
    FORMAT(O.OrderDate, 'dd/MM/yyyy') as OrderDate,
    O.OrderStatus,
    O.Price
FROM OrdersCl O
JOIN Clients C ON O.ID_Client = C.ID_Client

```

```
JOIN Products P ON O.ID_Product = P.ID_Product;
```

```
SELECT * FROM ViewClientOrders
```

```
-- 4) Виведення інформації про товар на складі--
```

```
CREATE VIEW ViewStockProducts
```

```
AS SELECT w.ID_Product, p.Name, p.Brand, s.CompanyName,  
SUM(w.Quantity) AS Total_Quantity
```

```
FROM Warehouse w
```

```
JOIN OrdersCl o ON w.ID_Product = o.ID_Product
```

```
LEFT JOIN Products p ON w.ID_Product = p.ID_Product
```

```
JOIN Suppliers s ON s.ID_Supplier = p.ID_Supplier
```

```
GROUP BY w.ID_Product, p.Name, p.Brand , s.CompanyName;
```

```
GO
```

```
-- 5) Статуси замовлень клієнтів
```

```
CREATE VIEW ViewClientOrdersHistory AS
```

```
SELECT o.ID_Order, C.LastName + ' ' + C.FirstName AS ClientName, p.Name  
AS Product,
```

```
o.OrderStatus, ps.PaymentStatus, d.DeliveryStatus, o.Price
```

```
FROM OrdersCl o
```

```
JOIN Clients c ON o.ID_Client = c.ID_Client
```

```
JOIN Products p ON o.ID_Product = p.ID_Product
```

```
LEFT JOIN Delivery d ON o.ID_Order = d.ID_Order
```

```
LEFT JOIN PaymentSys ps ON o.ID_Order = ps.ID_Order
```

```
LEFT JOIN Invoice i ON i.ID_PaymentSys = ps.ID_PaymentSys
```

```
ORDER BY LastName, Price DESC;
```

```
-- 6) Виведення інформації про успішні оплати для подальшої доставки  
товару до клієнтів --
```

```

CREATE VIEW ViewSuccessPayments AS
SELECT  o.ID_Order, pr.Name AS ProductName, c.FirstName, c.LastName,
c.Phone, c.AddressCl
FROM Clients c
JOIN PaymentSys p
  ON p.ID_Client = c.ID_Client
JOIN OrdersCl o
  ON o.ID_Client = c.ID_Client
JOIN Products pr
  ON pr.ID_Product = o.ID_Product
WHERE p.PaymentStatus = 'Success'
GO

```

-- 7) Виведення рахунків замовлень клієнтів --

```

CREATE VIEW ViewClientsInvoices AS
SELECT  c.LastName + ' ' + c.FirstName AS ClientName, c.Invoice,
SUM(DISTINCT i.Price) AS TotalSum, p.PaymentDate
FROM Invoice i
JOIN PaymentSys p
  ON i.ID_PaymentSys = p.ID_PaymentSys
JOIN Clients c
  ON c.ID_Client = p.ID_Client
JOIN OrdersCl o ON c.ID_Client = p.ID_Client
WHERE p.PaymentStatus = 'Success'
GROUP BY  c.Invoice, p.PaymentDate, c.LastName, c.FirstName;

```

-- 8) Інформація про клієнта для відділу логістики доставки --

```

ALTER VIEW ViewLogisticInfo AS (

```

```

SELECT d.ID_Order, c.FirstName, c.LastName, c.Phone, d.AddressDl,
convert(varchar, d.DeliveryDate, 4) AS DeliveryDate, d.DeliveryStatus
FROM Delivery d
JOIN OrdersCl o ON o.ID_Order = d.ID_Order
JOIN Clients c ON o.ID_Client = c.ID_Client
)

```

```

SELECT * FROM ViewLogisticInfo

```

-- 9) Інформація про компанії, ціль - визначити найприбутковіших
постачальників магазину

```

ALTER VIEW ViewCompanyProductSales AS

```

```

SELECT

```

```

    s.CompanyName,

```

```

    p.Name AS ProductName,

```

```

    p.Brand,

```

```

    ps.PaymentDate,

```

```

    SUM(p.Quantity) AS TotalSales,

```

```

    COUNT(DISTINCT c.ID_Client) AS TotalClients,

```

```

    SUM(i.Price) AS TotalRevenue

```

```

FROM OrdersCl o

```

```

LEFT JOIN Clients c ON c.ID_Client = o.ID_Client

```

```

LEFT JOIN Products p ON o.ID_Product = p.ID_Product

```

```

LEFT JOIN PaymentSys ps ON o.ID_Order = ps.ID_Order

```

```

LEFT JOIN Invoice i ON ps.ID_PaymentSys = i.ID_PaymentSys

```

```

LEFT JOIN Suppliers s ON s.ID_Supplier = p.ID_Supplier

```

```

WHERE ps.PaymentStatus = 'Success'

```

```

GROUP BY

```

```

    s.CompanyName,

```

```

p.Name,
p.Brand,
ps.PaymentDate
GO

```

```

SELECT * FROM ViewCompanyProductSales
ORDER BY TotalRevenue;

```

```

=====
-- Database-level roles для надання доступу реальним користувачам (для
внутрішньої користуваців БД)
--
=====

```

```

-- Системний адміністратор (вносить всі зміни)
-- Access all files and data within the corporate network
CREATE LOGIN sys_admin_user WITH PASSWORD = 'SysPa$$w0rd';
USE WebShopDB;
CREATE USER sys_admin_user FOR LOGIN sys_admin_user;
ALTER ROLE db_owner ADD MEMBER sys_admin_user;

```

```

-- Аналітик
-- доступ для читання таблиць, уявлень і функцій
CREATE LOGIN analyst_user WITH PASSWORD = 'AnalystPa$$w0rd';
CREATE USER analyst_user FOR LOGIN analyst_user;
ALTER ROLE db_datareader ADD MEMBER analyst_user;

```

```

-- Менеджер (або керівник інтернет-магазину)

```


-- Виконання DDL-інструкцій (створення, зміна, видалення об'єктів бази даних)

```
CREATE LOGIN e_manager_user WITH PASSWORD = 'ManagerPa$$w0rd';
CREATE USER e_manager_user FOR LOGIN e_manager_user;
ALTER ROLE db_ddladmin ADD MEMBER e_manager_user;
```

-- бухгалтер – доступ лише до фінансових таблиць та товарів

```
CREATE LOGIN sales_user WITH PASSWORD = 'SalesPa$$w0rd';
CREATE USER sales_user FOR LOGIN sales_user;
GRANT SELECT ON PaymentSys TO sales_user;
GRANT SELECT ON Clients TO sales_user;
GRANT SELECT ON OrdersCl TO sales_user;
GRANT SELECT ON Invoice TO sales_user;
GRANT SELECT ON Products TO sales_user;
GRANT SELECT ON Warehouse TO sales_user;
GO
```

-- Відділ логістики – доступ до інформації про замовлення, клієнтів і доставку

```
CREATE LOGIN logistic_user WITH PASSWORD = 'LogisticPa$$w0rd';
CREATE USER logistic_user FOR LOGIN logistic_user;
GRANT SELECT ON Delivery TO logistic_user;
GRANT SELECT ON OrdersCl TO logistic_user;
GRANT SELECT ON Clients TO logistic_user;
GO
```

=====

-- Application Role для зовнішніх користувачів (клієнти, постачальники, доставка)

-- не мають прямого SQL-доступу
-- працюють через зовнішні додатки (які ти передбачаєш, що будуть у майбутньому),
-- повинні мати мінімальні права, строго обмежені.

=====

-- Клієнти магазину
-- переглядати та оновлювати замовлення, переглядати інформацію про товари
-- переглядати свою особисту інформацію про себе, доставку, інвойси,
CREATE APPLICATION ROLE client_app_role WITH PASSWORD =
'ClientPa\$\$w0rd';

-- Постачальник магазину
-- переглядати рахунки; змінювати інформацію про склад, продукти, особисті дані
CREATE APPLICATION ROLE supplier_app_role WITH PASSWORD =
'SupplierPa\$\$w0rd';

-- кур'єр
-- переглядати та оновлювати (для статусу) інформацію про доставку
CREATE APPLICATION ROLE delivery_app_user WITH PASSWORD =
'DeliveryPa\$\$w0rd';

=====

-- Надання дозволу --

=====

----- 1. репорт 1

GRANT SELECT, UPDATE, INSERT ON ViewClientOrders TO
e_manager_user, sys_admin_user

GRANT SELECT ON ViewClientOrders TO analyst_user

----- 2. репорт 2

GRANT EXECUTE ON FindProductForNameBrand TO e_manager_user,
sys_admin_user

GRANT EXECUTE ON FindProductForNameBrand TO analyst_user

----- 3. репорт 3

GRANT SELECT, UPDATE ON ViewClientsInvoices TO e_manager_user,
sales_user

GRANT SELECT, UPDATE ON ViewClientsInvoices TO e_manager_user,
sales_user

GRANT SELECT ON ViewSuccessPayments TO analyst_user

GRANT SELECT, UPDATE ON ViewSuccessPayments TO e_manager_user,
sales_user

GRANT SELECT, UPDATE, INSERT ON ViewSuccessPayments TO
sys_admin_user

----- 4. репорт 4

GRANT SELECT, UPDATE ON ViewLogisticInfo TO
e_manager_user, logistic_user

GRANT SELECT, UPDATE, INSERT ON ViewLogisticInfo TO sys_admin_user

----- 5. репорт 5

GRANT SELECT ON ViewCompanyMonthSales TO analyst_user,
e_manager_user

```
GRANT SELECT, UPDATE, INSERT ON ViewCompanyMonthSales TO
sys_admin_user
```

----- 6. репорт 6

```
GRANT EXECUTE ON ChangeStatusDelivery TO delivery_app_user ;
```

----- 7. репорт 7

```
GRANT SELECT ON ViewCompanyProductSales TO analyst_user,
e_manager_user, sales_user
```

```
=====
```

```
-- client_app_role --
```

```
=====
```

```
-- Клієнти магазину
```

```
-- переглядати та оновлювати замовлення, переглядати інформацію про
товари
```

```
-- переглядати свою особисту інформацію про себе, доставку, інвойси,
```

```
-----
```

```
CREATE APPLICATION ROLE client_app_role WITH PASSWORD =
'ClientPa$$w0rd';
```

```
GRANT EXECUTE ON pr_RegisterAsCustomer TO client_app_role;
```

```
GRANT EXECUTE ON pr_AddInvoice TO client_app_role;
```

```
GRANT EXECUTE ON pr_UpdateIfoAsCustomer TO client_app_role;
```

```
GRANT EXECUTE ON FindProductForNameBrand TO client_app_role;
```

```
GRANT EXECUTE ON pr_MakeOrder TO client_app_role;
```

```
GRANT EXECUTE ON pr_UpdateOrderQuantity TO client_app_role;
```

```
GRANT EXECUTE ON pr_ViewPersonalOrders TO client_app_role;
```

```
GRANT SELECT ON vw_Catalog TO client_app_role;
```

--- 1. Процедура для реєстрації користувача

```
CREATE OR ALTER PROCEDURE pr_RegisterAsCustomer
```

```
    @IPAdresse VARCHAR(50),
```

```
    @FirstName VARCHAR(50),
```

```
    @LastName VARCHAR(50),
```

```
    @AddressCl VARCHAR(50),
```

```
    @Phone VARCHAR(50),
```

```
    @Email VARCHAR(50),
```

```
    @Invoice VARCHAR(50) = NULL
```

```
AS
```

```
BEGIN
```

```
    SET NOCOUNT ON;
```

```
    DECLARE @NewUserID INT;
```

```
        SELECT @NewUserID = ID_User
```

```
        FROM Users
```

```
        WHERE IPAdresse = @IPAdresse;
```

```
    IF @NewUserID IS NULL
```

```
        BEGIN INSERT INTO Users(IPAdresse)
```

```
            VALUES(@IPAdresse);
```

```
        SET @NewUserID = SCOPE_IDENTITY();
```

```
    END
```

```
-- Додаємо клієнта з прив'язкою до ID_User
```

```
INSERT INTO Clients (ID_User, FirstName, LastName, AddressCl, Phone,
Email, Invoice)
```

```
VALUES (@NewUserID, @FirstName, @LastName, @AddressCl, @Phone,
@Email, @Invoice);
END;
GO
```

```
DECLARE @cookie VARBINARY(8000);
```

```
-- Вмикаємо application role
```

```
EXEC sp_setapprole
```

```
    @rolename = 'client_app_role',
```

```
    @password = 'ClientPa$$w0rd',
```

```
    @fCreateCookie = 1,
```

```
    @cookie = @cookie OUTPUT;
```

```
EXEC pr_RegisterAsCustomer
```

```
    @IPAdresse = '192.168.0.6',
```

```
    @FirstName = 'Oleg',
```

```
    @LastName = 'Ivanov',
```

```
    @AddressCl = 'Lviv, St. Rynok, 12',
```

```
    @Phone = '093-123-4567',
```

```
    @Email = 'oleg@gmail.com',
```

```
    @Invoice = 'INV12';
```

```
-- Вимикаємо application role
```

```
EXEC sp_unsetapprole @cookie;
```

--- 2. Процедура для того щоб додавати рахок користувачу

```
CREATE OR ALTER PROCEDURE pr_AddInvoice
```

```
    @Invoice VARCHAR(50),
```

```
    @ID_Client INTEGER
```

```
AS BEGIN
```

```

UPDATE Clients
SET Invoice = @Invoice
WHERE ID_Client = @ID_Client;
END;
GO

```

```

DECLARE @cookie VARBINARY(8000);
-- Вмикаємо application role
EXEC sp_setapprole
    @rolename = 'client_app_role',
    @password = 'ClientPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;

```

```

EXEC pr_AddInvoice
    @ID_Client = 3,
    @Invoice = 'INV12';
-- Вимикаємо application role
EXEC sp_unsetapprole @cookie;

```

--- 3. Процедура для оновлення особистої інформації клієнта

```

CREATE OR ALTER PROCEDURE pr_UpdateIfoAsCustomer
    @ID_Client INTEGER,
    @AddressCl VARCHAR(50)= NULL,
    @Phone VARCHAR(50)= NULL,
    @Email VARCHAR(50)= NULL,
    @FirstName VARCHAR(50),
    @LastName VARCHAR(50)
AS BEGIN

```

IF @Phone IS NOT NULL

UPDATE Clients

SET Phone = @Phone WHERE ID_Client = @ID_Client

IF @AddressCl IS NOT NULL

UPDATE Clients

SET AddressCl = @AddressCl WHERE ID_Client = @ID_Client

IF @Email IS NOT NULL

UPDATE Clients

SET Email = @Email WHERE ID_Client = @ID_Client

END;

GO

DECLARE @cookie VARBINARY(8000);

-- Вмикаємо application role

EXEC sp_setapprole

@rolename = 'client_app_role',

@password = 'ClientPa\$\$w0rd',

@fCreateCookie = 1,

@cookie = @cookie OUTPUT;

EXEC pr_UpdateIfoAsCustomer

@ID_Client = 3,

@AddressCl = 'Lviv, St. Rynok, 12',

@Phone = '093-123-4567',

@Email = 'oleg@gmail.com';

-- Вимикаємо application role

EXEC sp_unsetapprole @cookie;

--- 4. Процедура пошуку товару за заданими користувачем параметрами бренду або типу товару

```
CREATE OR ALTER PROCEDURE FindProductForNameBrand
```

```
  @Name VARCHAR(75) = null, @Brand VARCHAR(75) = null
```

```
AS
```

```
BEGIN
```

```
  SELECT w.ID_Product, p.Name, p.Brand, s.CompanyName, p.Price,
         p.Quantity AS ShopQuantity, w.Quantity AS StockQuantity,
SUM(w.Quantity) AS Total_Quantity
```

```
  FROM Warehouse w
```

```
  JOIN OrdersCl o ON w.ID_Product = o.ID_Product
```

```
  LEFT JOIN Products p ON w.ID_Product = p.ID_Product
```

```
  JOIN Suppliers s ON s.ID_Supplier = p.ID_Supplier
```

```
  WHERE ((@Name IS NULL OR LOWER(p.Name) LIKE LOWER('%' +
@Name + '%'))
```

```
    AND
```

```
    (@Brand IS NULL OR LOWER(p.Brand) LIKE LOWER('%' + @Brand
+ '%'))))
```

```
  GROUP BY w.ID_Product, p.Name, p.Brand, s.CompanyName, p.Price,
         p.Quantity, w.Quantity
```

```
  ORDER BY Total_Quantity DESC
```

```
END;
```

```
GO
```

```
EXEC FindProductForNameBrand @Name = 'Laptop', @Brand = 'HP'
```

```
EXEC FindProductForNameBrand
```

--- 5. Процедура здійснення замовлення

CREATE OR ALTER PROCEDURE pr_MakeOrder

 @IPAdresse VARCHAR(50),

 @Name VARCHAR(50) = null,

 @Brand VARCHAR(50) = null,

 @ID_Product INTEGER = null,

 @ID_Client INTEGER,

 @Quantity INTEGER

AS BEGIN

 SET NOCOUNT ON;

 DECLARE @NewUserID INT;

 SELECT @NewUserID = ID_User

 FROM Users

 WHERE IPAdresse = @IPAdresse;

 IF @NewUserID IS NULL

 BEGIN INSERT INTO Users(IPAdresse)

 VALUES(@IPAdresse);

 SET @NewUserID = SCOPE_IDENTITY();

END

 IF @ID_Product IS NULL

 BEGIN SELECT @ID_Product = ID_Product

 FROM Products

 WHERE Name = @Name AND Brand = @Brand;

END

INSERT INTO OrdersCl (ID_Product, ID_Client, ID_User, OrderStatus, Price, OrderDate)

```

SELECT @ID_Product, @ID_Client, @NewUserID, 'Pending', Price *
@Quantity, GETDATE()
FROM Products
WHERE ID_Product = @ID_Product;
END;
GO

```

```

DECLARE @cookie VARBINARY(8000);
EXEC sp_setapprole
    @rolename = 'client_app_role',
    @password = 'ClientPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;

```

```

EXEC pr_MakeOrder
    @IPAdresse = '192.168.0.1',
    @ID_Client = 3,
    @Name = 'Laptop',
    @Brand = 'HP',
    @Quantity = 2;
-- Вимикаємо application role
EXEC sp_unsetapprole @cookie;

```

```

SELECT * FROM OrdersCl

```

--- 6. Процедура зміни кількості товару у замовленні --> оновлення суми замовлення

```

CREATE OR ALTER PROCEDURE pr_UpdateOrderQuantity
    @ID_Order INTEGER,

```

```

@ID_Product INTEGER,
@Quantity INTEGER
AS BEGIN
    SET NOCOUNT ON;

    UPDATE p
        SET p.Quantity = @Quantity
        FROM Products p
        JOIN OrdersCl o ON o.ID_Product = p.ID_Product
        WHERE o.ID_Order = @ID_Order AND o.ID_Product = @ID_Product

    UPDATE o
        SET o.Price = @Quantity * p.Price
        FROM OrdersCl o
        JOIN Products p ON o.ID_Product = p.ID_Product
        WHERE o.ID_Order = @ID_Order AND o.ID_Product = @ID_Product

END;
GO

DECLARE @cookie VARBINARY(8000);
EXEC sp_setapprole
    @rolename = 'client_app_role',
    @password = 'ClientPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;

EXEC pr_UpdateOrderQuantity
    @ID_Order = 12,
    @ID_Product = 9,
    @Quantity = 1;
-- Вимикаємо application role

```

```
EXEC sp_unsetapprole @cookie;
```

```
SELECT  p.Quantity,  o.Price,  o.ID_Order,  o.ID_Product  AS  OdPrOrd,
p.ID_Product AS OdPrPr
FROM Products p
JOIN OrdersCl o ON o.ID_Product = p.ID_Product
```

--- 7. Процедура перегляду замовлення

```
CREATE OR ALTER PROCEDURE pr_ViewPersonalOrders
    @ID_Order INTEGER
AS BEGIN
    SELECT * FROM ViewClientOrders vw
    WHERE vw.ID_Order = @ID_Order
END;
GO
```

```
DECLARE @cookie VARBINARY(8000);
EXEC sp_setapprole
    @rolename = 'client_app_role',
    @password = 'ClientPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;
```

```
EXEC pr_ViewPersonalOrders
    @ID_Order = 3;
-- Вимикаємо application role
EXEC sp_unsetapprole @cookie;
```

--- 8. Перегляд всіх товарів

```
CREATE VIEW vw_Catalog
AS SELECT Name, Brand, Price, Quantity
FROM Products;
```

```
=====
```

```
-- supplier_app_role --
```

```
=====
```

```
-- Постачальник магазину
```

```
-- переглядати рахунки; змінювати інформацію про склад, продукти, особисті дані
```

```
CREATE APPLICATION ROLE supplier_app_role WITH PASSWORD =
'SupplierPa$$w0rd';
```

```
GRANT EXECUTE ON pr_RegisterAsSupplier TO supplier_app_role;
GRANT EXECUTE ON pr_UpdateIfoAsSupplier TO supplier_app_role;
GRANT EXECUTE ON pr_UpdateStock TO supplier_app_role;
GRANT EXECUTE ON pr_ProductStock TO supplier_app_role;
```

```
--- 1. Процедура для реєстрації постачальника
```

```
CREATE OR ALTER PROCEDURE pr_RegisterAsSupplier
```

```
    @IPAdresse VARCHAR(75),
```

```
    @CompanyName VARCHAR(75),
```

```
    @AddressSp VARCHAR(75),
```

```
    @Phone VARCHAR(75)
```

```
AS BEGIN
```

```
    SET NOCOUNT ON;
```

```
    DECLARE @NewUserID INT;
```

```

SELECT @NewUserID = ID_User
FROM Users
WHERE IPAdresse = @IPAdresse;

IF @NewUserID IS NULL
BEGIN INSERT INTO Users(IPAdresse)
VALUES(@IPAdresse);
SET @NewUserID = SCOPE_IDENTITY();
END

INSERT INTO Suppliers (ID_User, CompanyName, AddressSp, Phone)
VALUES (@NewUserID, @CompanyName, @AddressSp,@Phone)
END;
GO

DECLARE @cookie varbinary(8000);
EXEC sp_setapprole
    @rolename = 'supplier_app_role',
    @password = 'SupplierPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;

EXEC pr_RegisterAsSupplier
    @IPAdresse = '192.168.0.6',
    @CompanyName = 'Auto Parts',
    @AddressSp = 'Odessa, St. Romashka, 1',
    @Phone = '067-224-1363'
EXEC sp_unsetapprole @cookie;

```

--- 2. Процедура для перегляду складу

```
CREATE OR ALTER PROCEDURE pr_ProductStock
    @ID_Supplier INTEGER
AS BEGIN
    SELECT p.Name, p.Brand, w.Quantity, w.Price
    FROM Warehouse w
    LEFT JOIN Suppliers s ON s.ID_Supplier = w.ID_Supplier
    LEFT JOIN Products p ON p.ID_Supplier = w.ID_Supplier
    WHERE @ID_Supplier = w.ID_Supplier
END;
GO
```

```
DECLARE @cookie varbinary(8000);
EXEC sp_setapprole
    @rolename = 'supplier_app_role',
    @password = 'SupplierPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;
```

```
EXEC pr_ProductStock
    @ID_Supplier = 1
EXEC sp_unsetapprole @cookie;
```

--- 3. Процедура для оновлення контактних даних про постачальника

```
CREATE OR ALTER PROCEDURE pr_UpdateIfoAsSupplier
    @ID_Supplier INTEGER,
    @CompanyName VARCHAR(75) = null,
    @AddressSp VARCHAR(100) = null,
```



```

        @Phone INTEGER= null
AS BEGIN
    IF @CompanyName IS NOT NULL
        UPDATE Suppliers
        SET CompanyName = @CompanyName
        WHERE ID_Supplier = @ID_Supplier

    IF @AddressSp IS NOT NULL
        UPDATE Suppliers
        SET AddressSp = @AddressSp
        WHERE ID_Supplier = @ID_Supplier

    IF @Phone IS NOT NULL
        UPDATE Suppliers
        SET Phone = @Phone
        WHERE ID_Supplier = @ID_Supplier
END;
GO

DECLARE @cookie varbinary(8000);
EXEC sp_setapprole
    @rolename = 'supplier_app_role',
    @password = 'SupplierPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;

EXEC pr_UpdateStock
    @ID_Supplier = 8,
    @CompanyName = 'Food & Drinks'
    @AddressSp = 'Dnipro, St. Pivdenna, 15',

```

```

@Phone = '050-8457-1617'
EXEC sp_unsetapprole @cookie;

```

--- 3. Процедура для оновлення товарів та складі

```

CREATE OR ALTER PROCEDURE pr_UpdateStock

```

```

    @ID_Supplier INTEGER,
    @ID_Product INTEGER,
    @ID_Warehouse INTEGER,
    @Quantity INTEGER = null,
    @Price FLOAT = null

```

```

AS BEGIN

```

```

    IF @Quantity IS NOT NULL

```

```

        UPDATE Warehouse

```

```

        SET Quantity = @Quantity

```

```

        WHERE @ID_Warehouse = ID_Warehouse AND ID_Product =
@ID_Product AND ID_Supplier = @ID_Supplier

```

```

    IF @Price IS NOT NULL

```

```

        UPDATE Warehouse

```

```

        SET Price = @Price

```

```

        WHERE @ID_Warehouse = ID_Warehouse AND ID_Product =
@ID_Product AND ID_Supplier = @ID_Supplier

```

```

END;

```

```

GO

```

```

DECLARE @cookie varbinary(8000);

```

```

EXEC sp_setapprole

```

```

    @rolename = 'supplier_app_role',

```

```

@password = 'SupplierPa$$w0rd',
@fCreateCookie = 1,
@cookie = @cookie OUTPUT;

```

```
EXEC pr_UpdateStock
```

```

    @ID_Supplier = 7,
    @ID_Product = 7,
    @ID_Warehouse = 7,
    @Quantity = 50,
    @Price = 16

```

```
EXEC sp_unsetapprole @cookie;
```

```
SELECT * FROM Warehouse
```

```
SELECT * FROM Delivery
```

```
=====
```

```
-- delivery_app_user --
```

```
=====
```

```
-- кур'єр
```

```
-- переглядати та оновлювати (для статусу) інформацію про доставку
```

```
CREATE APPLICATION ROLE delivery_app_user WITH PASSWORD =
'DeliveryPa$$w0rd';
```

```
GRANT EXECUTE ON pr_UpdateDeliveryAdress TO delivery_app_user;
```

```
GRANT EXECUTE ON vw_ClientDeliveryData TO delivery_app_user;
```

```
--- 1. Процедура оновлювати (для статусу) інформацію про доставку
```

```
CREATE PROCEDURE pr_UpdateDeliveryAdress
```

```
    @ID_Delivery INTEGER,
```

```

        @NewStatus VARCHAR(50)
    AS BEGIN
        UPDATE Delivery
        SET DeliveryStatus = @NewStatus, DeliveryDate = GETDATE()
        WHERE ID_Delivery = @ID_Delivery;
    END;

```

--- 2. Подання перегляд

```

CREATE PROCEDURE vw_ClientDeliveryData
    @ID_Order INTEGER
AS BEGIN
    SELECT * FROM ViewLogisticInfo
    WHERE ID_Order = @ID_Order;
END;

```

```

DECLARE @cookie varbinary(8000);
EXEC sp_setapprole
    @rolename = 'delivery_app_user',
    @password = 'DeliveryPa$$w0rd',
    @fCreateCookie = 1,
    @cookie = @cookie OUTPUT;

```

```

EXEC vw_ClientDeliveryData
    @ID_Order = 4

```

```

EXEC pr_UpdateDeliveryAdress
    @ID_Delivery = 4,
    @NewStatus = 'Delivered'

```

```
EXEC vw_ClientDeliveryData  
    @ID_Order = 4  
EXEC sp_unsetapprole @cookie;
```