Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та кібербезпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

АВТОМАТИЗОВАНА СИСТЕМА ОБЛІКУ ТОВАРІВ

AUTOMATED GOODS ACCOUNTING SYSTEM

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІс-21
Філіпчук Антон Олександрович

(підпис)

Керівник:
д.пед.н., професор
Чернящук Наталія Леонідівна

(підпис)

Кваліфікаційну роботу
допущено до захисту
«____» червня 2023 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2023 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та кібербезпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Н. Чернящук
«_____« _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Філіпчуку Аентону Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Автоматизована система обліку товарів*

Керівник роботи *д.пед.н., проф. Чернящук Наталія Леонідівна*

затверджені наказом закладу вищої освіти від «28» грудня 2022 року № 982/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 01.06.2023р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання , опубліковані зарубіжні та вітчизняні роботи в даній області*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Засоби адміністрування
IT інфраструктура Автоматизація задач в корпоративному середовищі
Висновки

5. Перелік графічного (ілюстративного) матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження	*Чернящук Н.Л.*		
Теоретичне дослідження та практична реалізація	*Чернящук Н.Л.*		
Практична реалізація об'єкта проектування	*Чернящук Н.Л.*		
Висновки			

7. Дата видачі завдання: 01.11.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми*	До 15.11.2022 р.	
2.	*Обгрунтування теми*	До 17.12.2022 р.	
3.	*Огляд підходів до контролю стану серверів*	До 11.01.2023 р.	
4.	*Моделювання моніторингу серверів за допомогою Zabbix*	До 04.02.2023 р.	
5.	*Функціонал оповіщення адміністратора*	До 22.02.2023 р.	
6.	*Захист інформації та криптографічні протоколи*	До 28.02.2023 р.	
7.	*Оцінка продуктивності та доступності баз даних*	До 01.03.2023 р.	
8.	*Інсталяція та конфігурація веб-сервера*	До 11.03.2023 р.	
9.	*Інтеграція PHP у середовище веб-сервера Nginx*	До 16.05.2023 р.	
10.	*Організація моніторингу продуктивності*	До 20.05.2023 р.	
11.	*Інструментальна перевірка на академічний плагіат*	До 22.05.2023 р.	
12.	*Представлення кваліфікаційної роботи бакалавра до захисту*	До 01.06.2023 р.	

Здобувач вищої освіти _____ (підпис) Філіпчук А.О. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) Чернящук Н.Л. (прізвище, ініціали)

АНОТАЦІЯ

Філіпчук А.О. Автоматизована система обліку товарів. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2023.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел.

Перший розділ присвячено аналізу завдання і вибору методу його вирішення, тут розглядається аналіз стану проблеми, та порівнялний аналіз аналогів. Також в цьому розділі здійснено аналіз методів розв'язання поставленої задачі.

В другому розділі здійснено розробку структури програмного продукту та розробки ER-молелі, інтерфейсу.

Третій розділ присвячено розробки програмного забезпечення де було використано варіантний аналіз і обґрунтування вибору мови програмування та вибору середовища розробки і розробки модулів програми. Було здійснено тестування роботи та аналізу результатів та методики, які були використані для тестування програм. Була розроблена інструкція користувача.

Об'єкт – є інформаційні технології та процеси обробки інформаційних ресурсів.

Предмет – є моделі, алгоритми, структури та методи розробки автоматизованих систем обробки інформаційних ресурсів та створення баз даних.

Метою роботи є створення автоматизованих систем обліку товарів для магазину.

Ключові слова: структура програмного продукту; ER-модель; інтерфейс; розробка модулів; інструкція для користувача; тестування програми; порівняльний аналіз; база даних.

ANNOTATION

Filipchuk A.O. Automated product accounting system. Manuscript.

Qualifying work of a bachelor of EP "Computer Engineering" specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2023.

The qualification work consists of an introduction, three sections, conclusions, and a list of used sources.

The first section is devoted to the analysis of the task and the choice of the method of its solution, here the analysis of the state of the problem and the comparative analysis of analogues are considered. Also, in this section, the methods of solving the given problem are analyzed.

In the second section, the development of the structure of the software product and the development of the ER module and interface were carried out.

The third section is devoted to the development of software, where variant analysis and justification of the choice of the programming language and the choice of the development environment and the development of the program modules were used.

Performance testing and analysis of the results and methods used to test the programs was carried out. A user manual was developed.

The object – information technology and information resource processing processes. for the store.

The subject – models, algorithms, structures and methods of developing automated systems for processing information resources and creating databases.

The method of work is to create an automated system of accounting for goods/

Keywords: software product structure; ER model; interface; development of modules; user manual; program testing; comparative analysis; Database.

ЗМІСТ

ВСТУП 7

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ПИТАНЬ РОЗРОБКИ 9

1.1 Особливості реляційних баз даних в обліку товарів підприємств 9

1.2 Порівняльний аналіз аналогів для обліку товарів 11

1.3 Аналіз методів вирішення проблеми 15

1.4 Розподіл та фрагментація даних 17

РОЗДІЛ 2 МЕТОДИ І ІНСТРУМЕНТИ РОЗРОБКИ 20

2.1 Формування вимог до автоматизованої системи обліку товарів 20

2.2 Проєктування структури бази даних підприємства 23

2.3 Моделювання логічної та фізичної схеми бази даних 25

2.4 Розробка SQL-запитів для обліку товарів, складу та замовлень 27

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ ТОВАРІВ 30

3.1 Створення таблиць та визначення зв’язків між ними 30

3.2 Реалізація обліку товарів, постачальників і складських залишків 32

3.3 Формування звітів і вибірок для аналізу товарообігу 34

3.4 Аналіз отриманих результатів та можливості масштабування системи 36

ВИСНОВКИ 38

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 40

ВСТУП

Актуальність теми дослідження полягає в тому, що впровадження сучасних інформаційних систем обліку забезпечує точність даних, зменшує ймовірність помилок, скорочує час на обробку інформації та підвищує ефективність управління товарними запасами підприємства. Особливо важливою є інтеграція баз даних з аналітичними інструментами, що дозволяє здійснювати глибокий аналіз товарообігу, планувати закупівлі та оптимізувати логістику.

Мета роботи полягає у розробці та реалізації автоматизованої системи обліку товарів підприємства на основі реляційної бази даних, яка забезпечує повний контроль за рухом товарів, постачальників та замовлень, а також формування аналітичних звітів для підтримки управлінських рішень.

Об’єктом дослідження є процеси обліку товарів, постачальників і складських залишків на підприємстві.

Предметом дослідження є методи та інструменти проектування і реалізації реляційної бази даних для автоматизації обліку товарообігу, включаючи моделювання структури, визначення зв’язків між таблицями та формування SQL-запитів для отримання аналітичних даних.

Для досягнення поставленої мети в роботі були визначені наступні завдання:

– провести аналіз особливостей реляційних баз даних та методів обліку товарів на підприємствах;

– виконати порівняльний аналіз існуючих аналогів систем обліку товарів;

– дослідити методи вирішення проблеми інтеграції та контролю даних;

– розробити логічну та фізичну схему бази даних підприємства;

– реалізувати таблиці та визначити зв’язки між ними для обліку товарів, постачальників, складу та замовлень;

– створити SQL-запити для обліку товарообігу та формування аналітичних звітів;

– провести аналіз отриманих результатів та визначити можливості масштабування системи.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ПИТАНЬ РОЗРОБКИ

1.1 Особливості реляційних баз даних в обліку товарів підприємств

Підприємцям, які вирішують питання обліку даних, важливо розуміти, що існують різні типи систем керування базами даних і кожна підходить для своїх задач.

Традиційні реляційні бази даних добре працюють з чітко структурованою інформацією, де важливо зберігати зв’язки між даними, наприклад товари, клієнти, постачальники чи замовлення. Реляційні БД використовують SQL як мову для створення, читання, оновлення та видалення даних і гарантовано підтримують цілісність через первинні та зовнішні ключі. Реляційні бази зручні для звітності, аналізу і підтримки транзакцій, коли важливо, щоб всі пов’язані зміни відбувалися узгоджено і без помилок. SQL бази традиційно підходять для невеликих і середніх бізнесів, оскільки вони забезпечують стабільність і передбачуваність при обробці даних, що особливо важливо для обліку товарів і фінансів. Через те, що схема даних у таких системах зазвичай заздалегідь визначена, вони добре працюють, коли структура даних не змінюється дуже часто. Джерело [1] пояснює головні відмінності між реляційними та нереляційними базами і їх сильні сторони (рис. 1.1).

SQL	NoSQL
Stands for Structured Query Language	Stands for Not Only SQL
Relational database management system (RDBMS)	Non-relational database management system
Suitable for structured data with predefined schema	Suitable for unstructured and semi-structured data
Data is stored in tables with columns and rows	Data is stored in collections or documents
Follows ACID properties (Atomicity, Consistency, Isolation, Durability) for transaction management	Does not necessarily follow ACID properties
Supports JOIN and complex queries	Does not support JOIN and complex queries
Uses normalized data structure	Uses denormalized data structure
Requires vertical scaling to handle large volumes of data	Horizontal scaling is possible to handle large volumes of data
Examples: MySQL, PostgreSQL, Oracle, SQL Server, Microsoft SQL Server	Examples: MongoDB, Cassandra, Couchbase, Amazon DynamoDB, Redis

Рисунок 1.1 – Порівняння баз даних [1]

Нереляційні, або NoSQL бази даних, натомість пропонують більшу гнучкість зберігання даних без заздалегідь визначеної структури. Вони не використовують традиційні таблиці з рядками і стовпцями, а можуть зберігати документи, графи чи ключ-значення, що особливо корисно, якщо дані сильно варіюються або швидко змінюються. NoSQL системи зазвичай забезпечують горизонтальну масштабованість, завдяки чому легше обробляють великі обсяги даних і розподілені навантаження.

Основою реляційної моделі даних є таблична структура, що дозволяє забезпечити логічну цілісність та взаємозв’язок між даними. Кожна таблиця має первинний ключ (primary key), який однозначно ідентифікує кожен рядок, а також може містити зовнішні ключі (foreign key) для встановлення зв’язків з іншими таблицями.

Реляційні бази даних підтримують операції створення, читання, оновлення та видалення даних (CRUD), а взаємодія з ними здійснюється за допомогою мови структурованих запитів SQL (Structured Query Language). SQL забезпечує потужні можливості для маніпуляції даними, виконання складних запитів, сортування, фільтрації та об’єднання таблиць.

Основні принципи реляційної моделі включають:

– унікальність рядків – жоден запис не повторюється у межах таблиці;

– нормалізацію – процес організації даних для мінімізації надлишковості та запобігання аномалій оновлення;

– цілісність даних, що забезпечується через первинні та зовнішні ключі, обмеження унікальності та типи даних;

– маніпулювання даними через запити, що дозволяє здійснювати вибірку, об’єднання та агрегування інформації

Відсутність єдиного стандарту запитів і необхідність вибору конкретного типу NoSQL бази може бути складнішим для підприємців без технічної підтримки. Обидва підходи мають свої переваги і недоліки, і вибір залежить від потреб бізнесу і типу даних, які він обробляє.

Серед конкретних систем, що добре підходять для малого та середнього бізнесу, є безліч реляційних СУБД з відкритим кодом, таких як PostgreSQL, MySQL, MariaDB і вбудовані рішення типу SQLite, які забезпечують стабільну роботу і потужні можливості для обліку даних, включаючи транзакції та складні запити. Такі системи підтримуються великою спільнотою, мають багато ресурсів і прикладів використання, що спрощує їх впровадження без значних витрат на ліцензії.

Перевага реляційних рішень для підприємців у першу чергу полягає в тому, що вони дозволяють організувати дані про товари, клієнтів, постачальників і транзакції у зв'язні таблиці, забезпечують стабільну роботу складних запитів і звітів, і легко інтегруються з обліковими та ERP-системами. Для невеликих завдань або початкового етапу бізнесу чудовим варіантом може бути легка база SQLite, яку можна інтегрувати напряму в програму або звітний інструмент без окремого серверу.

DB-Fiddle у цьому контексті виступає як онлайн-середовище для створення, тестування і демонстрації SQL-баз даних без необхідності встановлення локального програмного забезпечення. Воно дозволяє підприємцям і розробникам швидко експериментувати зі структурою таблиць, SQL-запитами і зв'язками між даними, перевіряти запити до реляційних баз, які будуть використовуватися в бізнес-логіці, і обмінюватися цими ділянками SQL-коду з іншими учасниками проєкту для спільної роботи.

1.2 Порівняльний аналіз аналогів для обліку товарів

Для організації обліку товарів підприємства використовують різні типи баз даних, і вибір конкретного рішення залежить від масштабу бізнесу, складності обліку та потреб у звітності. Найпоширенішими є реляційні бази даних, які зберігають інформацію у таблицях з чіткою структурою та підтримують зв'язки між даними через первинні та зовнішні ключі. До прикладу, MySQL та PostgreSQL добре підходять для невеликих і середніх підприємств, оскільки вони

забезпечують надійне зберігання даних, підтримку транзакцій і складні SQL-запити для формування звітів про залишки, продажі та постачання. Реляційні бази дозволяють легко інтегруватися з бухгалтерським та складським програмним забезпеченням, а також гарантують цілісність і узгодженість інформації.

Нереляційні бази даних типу MongoDB, CouchDB або Firebase застосовуються, якщо облік товарів вимагає більшої гнучкості у структурі даних або інтеграції з веб-сервісами і мобільними додатками. Вони дозволяють зберігати документи або ключ-значення, легко масштабуються при збільшенні обсягу даних і підтримують розподілене зберігання. Однак вони не завжди підходять для складних звітів і транзакцій, що може бути критично для обліку товарів на підприємстві.

Також існують готові рішення у вигляді ERP-систем або хмарних сервісів обліку, наприклад 1С:Підприємство, Odoo або Zoho Inventory. Вони включають готові модулі для ведення товарного обліку, автоматичного формування звітів і контролю залишків. Перевага таких систем у швидкому старті без розробки бази з нуля, але їх використання пов’язане з ліцензійними витратами та залежністю від постачальника.

DB-Fiddle в цьому контексті пропонує легкий і гнучкий інструмент для розробки власної реляційної бази даних обліку товарів. Воно дозволяє створювати таблиці, визначати зв’язки, додавати дані та тестувати SQL-запити онлайн без встановлення СУБД на локальний комп’ютер. Це особливо зручно для підприємців, які хочуть попередньо спроектувати структуру бази і перевірити логіку запитів перед впровадженням у реальне програмне забезпечення.

Для організації обліку товарів підприємства використовують різні типи баз даних, і вибір конкретного рішення залежить від масштабу бізнесу, складності обліку та потреб у звітності. Найпоширенішими є реляційні бази даних, які зберігають інформацію у таблицях з чіткою структурою та підтримують зв’язки між даними через первинні та зовнішні ключі. До прикладу, MySQL та

PostgreSQL добре підходять для невеликих і середніх підприємств, оскільки вони забезпечують надійне зберігання даних, підтримку транзакцій і складні SQL-запити для формування звітів про залишки, продажі та постачання. Реляційні бази дозволяють легко інтегруватися з бухгалтерським та складським програмним забезпеченням, а також гарантують цілісність і узгодженість інформації.

Нереляційні бази даних типу MongoDB, CouchDB або Firebase застосовуються, якщо облік товарів вимагає більшої гнучкості у структурі даних або інтеграції з веб-сервісами і мобільними додатками. Вони дозволяють зберігати документи або ключ-значення, легко масштабуються при збільшенні обсягу даних і підтримують розподілене зберігання. Однак вони не завжди підходять для складних звітів і транзакцій, що може бути критично для обліку товарів на підприємстві.

Також існують готові рішення у вигляді ERP-систем або хмарних сервісів обліку, наприклад 1С:Підприємство, Odoo або Zoho Inventory. Вони включають готові модулі для ведення товарного обліку, автоматичного формування звітів і контролю залишків. Перевага таких систем у швидкому старті без розробки бази з нуля, але їх використання пов’язане з ліцензійними витратами та залежністю від постачальника.

DB-Fiddle в цьому контексті пропонує легкий і гнучкий інструмент для розробки власної реляційної бази даних обліку товарів. Воно дозволяє створювати таблиці, визначати зв’язки, додавати дані та тестувати SQL-запити онлайн без встановлення СУБД на локальний комп’ютер. Це особливо зручно для підприємців, які хочуть попередньо спроектувати структуру бази і перевірити логіку запитів перед впровадженням у реальне програмне забезпечення.

В роботі [2] досліджено, що «з новими технологіями NoSQL та різними варіантами баз даних на ринку, розробники систем електронної комерції задаються питанням, чи є NoSQL кращим варіантом для їхніх платформ». Щоб допомогти розробникам систем електронної комерції приймати кращі рішення

щодо баз даних, це дослідження провело 9 тестів використання – 5 однопотокових тестів, 4 багатопотокові тести – з операціями CRUD для порівняння продуктивності баз даних PostgreSQL та MongoDB з реальними даними електронної комерції, експортованими з Kaggle. У цих 9 тестах PostgreSQL перевершив MongoDB майже в 7 тестах, тоді як MongoDB показав кращі результати в операціях вставки та видалення зі сценаріями одного потоку. Тому це дослідження показало, що PostgreSQL є кращим варіантом для платформ електронної комерції, де часто відбувається велика кількість одночасних запитів. Автор також підозрює, що природа моделі даних електронної комерції, яка є більш реляційною, визначає результат, що SQL працює краще, ніж NoSQL у сценаріях електронної комерції (рис. 1.2).

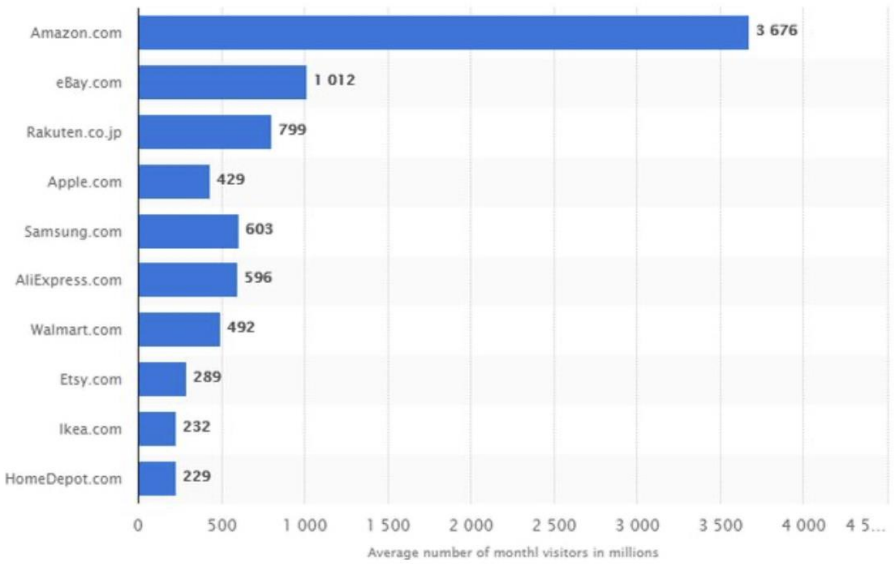

Рисунок 1.2 – Найпопулярніші онлайн-ритейл сайти у світі в 2020 році за середнім місячним трафіком [2]

Дослідження [3] порівнює продуктивність основних СУБД: MS SQL Server, Oracle, IBM DB2, MySQL, MS Access з точки зору виконання складних SQL-запитів і робочого навантаження, що може бути корисним при оцінці різних реляційних систем для обліку даних.

1.3 Аналіз методів вирішення проблеми

Вибір системи керування базами даних MySQL у межах даної роботи зумовлений її поширеним використанням у прикладних інформаційних системах, зокрема в системах обліку товарів на підприємствах. Використання мови SQL дозволяє реалізовувати запити різного рівня складності, що є необхідним для аналізу залишків товарів, обліку руху продукції та формування аналітичних звітів у межах автоматизованих систем управління.

Важливою перевагою MySQL є її адаптованість до потреб бізнес-середовища та сумісність із сучасними програмними платформами. Ця система керування базами даних широко застосовується у веборієнтованих проєктах, системах електронної комерції та корпоративних інформаційних рішеннях, що підтверджує її надійність і практичну доцільність. Завдяки цьому розроблена в роботі структура бази даних може розглядатися як така, що відповідає реальним вимогам підприємницької діяльності та здатна до подальшого розширення або інтеграції з іншими програмними компонентами.

Вибір онлайн-середовища DB-Fiddle як інструменту реалізації бази даних пояснюється його функціональністю та зручністю використання в навчальних цілях. DB-Fiddle надає можливість працювати з SQL-запитами у веббраузері без встановлення локальної СУБД, що спрощує процес проєктування та тестування структури бази даних. За допомогою цього середовища можна створювати таблиці, задавати зв’язки між ними, наповнювати базу тестовими даними та перевіряти коректність виконання запитів у режимі реального часу.

Поєднання MySQL як промислово орієнтованої системи керування базами даних та DB-Fiddle як інструменту її практичної реалізації дозволяє досягти балансу між теоретичною обґрунтованістю та практичною реалізацією дослідження. Такий підхід забезпечує відповідність роботи сучасним світовим практикам проєктування систем обліку товарів і водночас створює зручні умови для демонстрації результатів у межах бакалаврської кваліфікаційної роботи.

Нижче подано таблицю 1.1 основних методів роботи з реляційною базою даних, які використовуються під час реалізації системи обліку товарів на основі MySQL, а також пояснення її змісту.

Таблиця 1.1 – Методи формування баз даних для обліку товарів

Метод	Призначення	Значення для обліку товарів
CREATE TABLE	Створення структури таблиць у базі даних	Дозволяє визначити логічну модель зберігання даних про товари, категорії, постачальників і складські операції
INSERT	Додавання нових записів до таблиць	Забезпечує введення інформації про нові товари, надходження продукції та початкові залишки
SELECT	Отримання даних з бази даних	Використовується для формування звітів, перегляду залишків товарів, аналізу руху продукції
UPDATE	Оновлення наявних записів	Дозволяє коригувати дані про ціну, кількість товарів або інші характеристики у процесі діяльності підприємства
DELETE	Видалення записів	Забезпечує видалення застарілої або помилкової інформації з бази даних
JOIN	Об’єднання даних з кількох таблиць	Дозволяє отримувати узагальнену інформацію про товари разом із категоріями та постачальниками
PRIMARY KEY	Ідентифікація записів у таблиці	Гарантує унікальність кожного товару або облікового запису
FOREIGN KEY	Встановлення зв’язків між таблицями	Забезпечує логічну цілісність даних між товарами та пов’язаними сутностями

Наведена таблиця відображає ключові методи та механізми реляційної бази даних, які є базовими для побудови автоматизованої системи обліку товарів. Методи створення та наповнення таблиць формують основу структури бази даних і дозволяють організувати зберігання інформації відповідно до предметної області підприємства. Запити вибірки та об’єднання даних забезпечують можливість отримання аналітичної інформації, необхідної для прийняття управлінських рішень.

Методи оновлення та видалення даних відіграють важливу роль у підтримці актуальності інформації в умовах постійного руху товарів і змін у номенклатурі. Використання первинних і зовнішніх ключів дозволяє забезпечити цілісність даних та запобігти виникненню логічних помилок у базі даних. Сукупне застосування наведених методів у середовищі MySQL та DB-Fiddle створює надійну основу для реалізації системи обліку товарів, орієнтованої на практичні потреби підприємства.

1.4 Розподіл та фрагментація даних

Розподіл даних розглядається як раціональний процес розміщення окремих частин інформації, які поділяються на фрагменти, з метою підвищення продуктивності роботи бази даних та покращення доступу користувачів до інформаційних ресурсів. Основним завданням такого підходу є зменшення сумарних витрат на обробку транзакцій, а також забезпечення оперативного отримання достовірних і актуальних даних у середовищі розподілених систем керування базами даних.

Реалізація ефективного розподілу даних сприяє підвищенню швидкодії системи та надійності обробки інформації [4].

Розподіл даних є одним з визначальних етапів проєктування розподілених баз даних, оскільки саме на цьому етапі формується логіка зберігання і доступу до інформації в межах кількох вузлів системи.

Для досягнення оптимального результату зазвичай застосовуються дві базові стратегії, а саме фрагментація та реплікація даних. Зазначені підходи дозволяють підвищити ефективність використання ресурсів та забезпечити безперервність доступу до даних. Детальний аналіз цих методів розглядається у відповідних підрозділах, присвячених фрагментації та реплікації в розподілених базах даних.

Фрагментація даних визначається як процес поділу відношень або таблиць бази даних на окремі частини, які можуть розміщуватися на різних вузлах

системи. Такий підхід дозволяє розбити базу даних на менші логічні підтаблиці або підвідношення, що сприяє ефективнішому зберіганню інформації та оптимізації доступу до неї. У результаті фрагментації дані можуть оброблятися локально, що зменшує обсяг передавання інформації між вузлами системи [5].

Залежно від способу поділу інформації фрагментація бази даних може здійснюватися у двох основних формах.

При горизонтальній фрагментації кожен кортеж відношення розподіляється між одним або кількома фрагментами відповідно до заданих умов відбору.

Вертикальна фрагментація, у свою чергу, передбачає розбиття схеми відношення на декілька менших схем, які містять підмножини атрибутів та об’єднуються за допомогою спільного ключа-кандидата, що забезпечує логічну цілісність даних.

Розподілені бази даних у загальному вигляді поділяються на однорідні та гетерогенні середовища, кожне з яких характеризується власними особливостями організації та взаємодії компонентів.

Така класифікація дозволяє систематизувати підходи до побудови розподілених баз даних залежно від типу використовуваних систем керування базами даних, архітектури та способів інтеграції.

Кожен із зазначених типів, у свою чергу, має додаткові підкласи, що відрізняються рівнем сумісності, методами управління даними та механізмами доступу до інформації, що наочно представлено на рисунку 1.3.

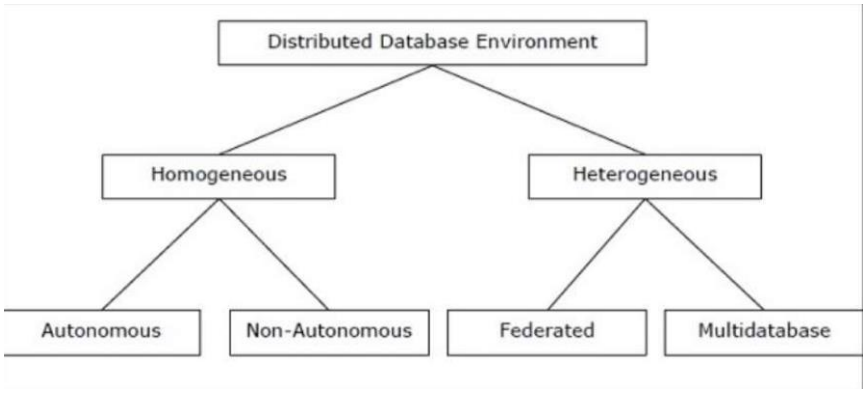

Рисунок 1.3 – Підрозділи розподілених баз даних [5]

Отримані результати аналітичного дослідження підтверджують, що використання реляційної бази даних у поєднанні з сучасними інструментами розробки дозволяє створити ефективну та масштабовану систему обліку товарів.

РОЗДІЛ 2
МЕТОДИ І ІНСТРУМЕНТИ РОЗРОБКИ

2.1 Формування вимог до автоматизованої системи обліку товарів

Автоматизована система обліку товарів призначена для зберігання, обробки та аналізу інформації про товарні позиції, їх категоризацію, постачання, складські залишки та операції замовлення. Основною метою такої системи є підвищення ефективності управління товарними потоками, зменшення помилок ручного обліку та забезпечення актуальності даних у реальному часі [6].

Система повинна забезпечувати централізоване зберігання даних у реляційній базі даних з чітко визначеною структурою та зв’язками між сутностями (табл. 2.1).

Таблиця 2.1 – Таблиця формування категорій товарів

Поле	Тип даних	Опис
CategoryID	Ціле число	Унікальний ідентифікатор категорії PK
CategoryName	Текст	Назва категорії –Електроніка, Канцелярія тощо
Description	Текст	Опис категорії

Облік товарів (табл. 2.2) має здійснюватися з урахуванням їх належності до певних категорій, що дозволяє впорядковувати номенклатуру продукції та спрощувати пошук і аналіз. Для кожного товару система повинна зберігати назву, ціну, кількість, а також зв’язок з відповідною категорією.

Таблиця 2.2 – Таблиця товарних позицій

Поле	Тип даних	Опис
ProductID	Ціле число	Унікальний ідентифікатор товару PK
ProductName	Текст	Назва товару
Price	Дробове число	Ціна за одиницю
Quantity	Ціле число	Поточна кількість товару, яку можна розраховувати зі складу
CategoryID	Ціле число	Посилання на категорію FK → Categories.CategoryID
SupplierID	Ціле число	Посилання на постачальника FK → Suppliers.SupplierID

Окремою вимогою є підтримка обліку постачальників, які забезпечують надходження товарів на підприємство (табл. 2.3).

Таблиця 2.3 – Дані про постачальників товарів

Поле	Тип даних	Опис
SupplierID	Ціле число	Унікальний ідентифікатор постачальника PK
SupplierName	Текст	Назва компанії постачальника
ContactName	Текст	Контактна особа
Phone	Текст	Телефонний номер
Email	Текст	Email для зв’язку
Address	Текст	Адреса постачальника

Система повинна зберігати контактну інформацію постачальників та забезпечувати зв’язок між постачальником і замовленнями. Це дозволяє відстежувати джерело походження товарів і аналізувати співпрацю з контрагентами.

Важливим компонентом автоматизованої системи є складський облік, який повинен відображати фактичні залишки товарів, їх розміщення та зміну кількості в результаті надходження або відпуску. Система має забезпечувати узгодженість даних між таблицями товарів і складу, що досягається шляхом використання зовнішніх ключів і обмежень цілісності.

Автоматизована система також повинна підтримувати операції формування замовлень, збереження дати замовлення та інформації про постачальника [7]. Це дає змогу відстежувати історію закупівель і забезпечує основу для подальшого аналізу товарообігу. Усі операції з даними повинні виконуватися за допомогою SQL-запитів, що забезпечує універсальність і масштабованість системи.

Розширена блок-схема автоматизованої системи обліку товарів включає такі основні логічні блоки та зв’язки між ними.

У центральній частині схеми розміщується блок «Товари», який містить інформацію про всі товарні позиції без обмеження їх кількості. Кожен товар пов’язаний із блоком «Категорії», що дозволяє класифікувати продукцію за

типами, наприклад електроніка, побутова техніка, канцелярія, офісне обладнання, витратні матеріали та інші групи.

Блок «Товари» також має прямий зв’язок із блоком «Склад», у якому відображається кількість кожного товару та його місце зберігання. Цей зв’язок забезпечує синхронізацію інформації про номенклатуру товарів і фактичні залишки на складі.

Блок «Постачальники» пов’язаний із блоком «Замовлення». Кожне замовлення містить посилання на конкретного постачальника та дату оформлення. Через блок «Замовлення» реалізується логічний зв’язок між постачальниками і товарами, що дозволяє фіксувати факт надходження товарів у систему.

Узагальнено вдосконалена блок-схема (рис. 2.1) відображає взаємодію таких сутностей, як категорії, товари, склад, постачальники та замовлення, що у сукупності формує цілісну автоматизовану систему обліку товарів, придатну для використання в реальних умовах підприємства та подальшого масштабування.

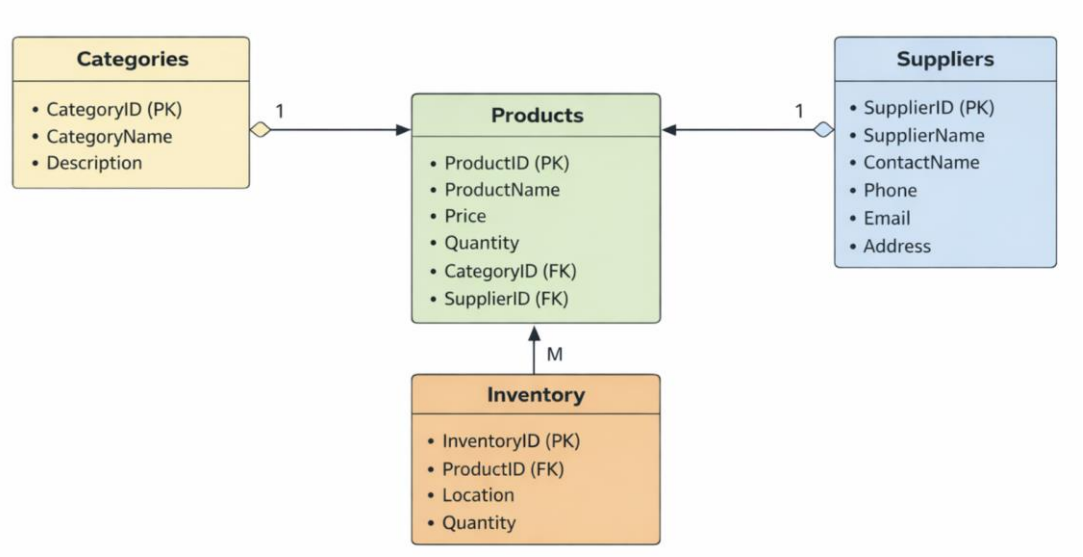

Рисунок 2.1 – ER-діаграма структури автоматизованої системи обліку товарів

Схема показує централізовану структуру бази даних із чітким відстеженням категорій, постачальників, товарів і залишків на складах, що дозволяє ефективно керувати обліком товарів.

2.2 Проєктування структури бази даних підприємства

Проєктування структури бази даних підприємства розпочинається з визначення предметної області та основних інформаційних об’єктів, що відображають діяльність організації. Для цього виділяються ключові сутності, між якими існують стійкі логічні зв’язки, а також визначається набір атрибутів, необхідних для збереження та обробки даних. У межах даного проєкту до таких сутностей належать departments, positions, employees, projects та employee_projects, які забезпечують зберігання інформації про організаційну структуру підприємства, персонал і його участь у проєктах. -модель предметної області.

Сутність departments призначена для зберігання даних про структурні підрозділи підприємства. Вона містить атрибут id, який є унікальним ідентифікатором запису та використовується як первинний ключ, а також атрибут name, що зберігає назву відділу та має обмеження унікальності для запобігання дублюванню даних.

Сутність positions використовується для опису посад, які можуть обіймати працівники. Основним атрибутом цієї таблиці є id, що виконує роль первинного ключа, та атрибут title, який зберігає назву посади. Така структура дозволяє уникнути повторного введення однакових назв посад для різних працівників.

Сутність employees є центральною в структурі бази даних і призначена для зберігання інформації про працівників підприємства. Вона включає атрибут id як первинний ключ, full_name для збереження повного імені працівника, email і phone для контактної інформації, hire_date для фіксації дати прийняття на роботу та salary для збереження розміру заробітної плати. Крім того, атрибути department_id та position_id є зовнішніми ключами, що забезпечують зв’язок працівника з відповідним відділом у таблиці departments та посадою у таблиці positions, гарантуючи цілісність даних.

Сутність projects використовується для зберігання інформації про проєкти, які реалізуються на підприємстві. Вона містить атрибут id як первинний ключ,

атрибут name для назви проєкту, а також атрибути start_date та end_date, що дозволяють фіксувати часові межі виконання проєкту.

Сутність employee_projects реалізує зв’язок типу багато-до-багатьох між працівниками та проєктами. Вона містить атрибути employee_id та project_id, які одночасно виконують роль зовнішніх ключів і формують складений первинний ключ таблиці. Додатковий атрибут role використовується для збереження інформації про роль працівника в конкретному проєкті. Такий підхід дозволяє гнучко описувати участь одного працівника в кількох проєктах і навпаки (рис. 2.2).

Рисунок 2.2 – Схема зв’язків бази даних між працівниками і проектами

Для реалізації описаної структури бази даних було використано онлайн-сервіс db-fiddle (рис. 2.3), який дозволяє створювати та перевіряти SQL-схеми без встановлення локальної системи управління базами даних.

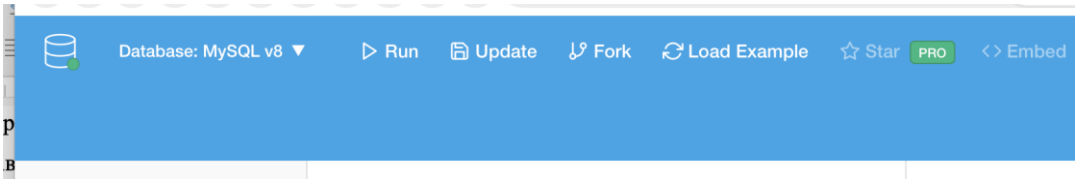

Рисунок 2.3 – Інтерфейс меню db-fiddle

У середовищі db-fiddle структура бази даних описується мовою SQL за допомогою операторів CREATE TABLE, після чого може бути перевірена шляхом виконання запитів на вставку та вибірку даних. Запропонована модель відповідає вимогам реляційного проєктування та нормалізована до третьої нормальної форми, що забезпечує відсутність надлишковості даних і підтримання їхньої цілісності.

2.3 Моделювання логічної та фізичної схеми бази даних

Моделювання логічної та фізичної схеми бази даних у кваліфікаційній роботі спрямоване на формалізацію структури зберігання даних підприємства та забезпечення їхньої цілісності, узгодженості й ефективної обробки. На етапі логічного моделювання визначаються основні сутності предметної області, їхні атрибути та взаємозв’язки без прив’язки до конкретної системи управління базами даних. Фізичне моделювання, у свою чергу, полягає в реалізації логічної моделі засобами конкретної СУБД із урахуванням типів даних, ключів, обмежень і механізмів забезпечення продуктивності.

Логічна схема бази даних включає п’ять основних таблиць: departments, positions, employees, projects та employee_projects. Таблиця departments у логічній моделі відображає організаційну структуру підприємства та містить атрибути, що ідентифікують відділи й описують їхні назви:

```
INSERT INTO departments (id, name) VALUES
(1, 'IT'),
(2, 'HR'),
(3, 'Finance'),
(4, 'Marketing');
```

Ця сутність використовується як довідкова та є базовою для формування зв’язків із персоналом. У фізичній схемі вона реалізується як таблиця з первинним ключем id та атрибутом name, для якого встановлюється обмеження унікальності з метою уникнення дублювання назв відділів.

Таблиця positions у логічній схемі призначена для представлення посад, які можуть займати працівники підприємства:

```
INSERT INTO positions (id, title) VALUES
(1, 'Developer'),
(2, 'Project Manager'),
(3, 'HR Specialist'),
(4, 'Accountant'),
(5, 'Marketing Analyst');
```

Вона забезпечує відокремлення інформації про посади від даних про конкретних працівників, що відповідає принципам нормалізації. У фізичній реалізації ця сутність представлена таблицею з первинним ключем id та атрибутом title, який зберігає назву посади та використовується під час формування зв'язків із таблицею employees.

Центральною сутністю логічної схеми є таблиця employees, яка описує персонал підприємства:

```
INSERT INTO employees
(id, full_name, email, phone, hire_date, salary, department_id, position_id)
VALUES
(1, 'Іван Петренко', 'ivan.petrenko@company.com', '+380501112233', '2021-05-10', 45000,
(2, 'Олена Коваль', 'olena.koval@company.com', '+380671234567', '2020-03-01', 60000, 1,
(3, 'Сергій Мельник', 'serhii.melnyk@company.com', '+380931112244', '2019-09-15', 35000,
(4, 'Наталія Шевчук', 'nataliia.shevchuk@company.com', '+380991223344', '2018-02-20', 40
(5, 'Андрій Литвин', 'andrii.lytvyn@company.com', '+380681998877', '2022-06-05', 38000,
```

У межах логічного моделювання вона включає атрибути, що характеризують працівника, його контактні дані та кадрову інформацію, а також зв'язки з відділом і посадою. На фізичному рівні ця сутність реалізується у вигляді таблиці з первинним ключем id та зовнішніми ключами department_id і position_id, що посилаються відповідно на таблиці departments і positions. Такий підхід дозволяє забезпечити цілісність даних і унеможливити існування працівника без прив'язки до структурного підрозділу чи посади.

Таблиця projects у логічній схемі відображає проєктну діяльність підприємства та використовується для зберігання інформації про проєкти, у яких беруть участь працівники. Вона містить атрибути, що дозволяють ідентифікувати проєкт і описати часові межі його реалізації. У фізичній схемі таблиця projects реалізується з первинним ключем id та атрибутами name,

start_date і end_date, що дає змогу виконувати аналіз тривалості проєктів і планування ресурсів.

Для реалізації зв’язку типу багато-до-багатьох між працівниками та проєктами в логічній схемі використовується таблиця employee_projects:

```
INSERT INTO employee_projects
(employee_id, project_id, role)
VALUES
(1, 101, 'Backend Developer'),
(1, 102, 'Frontend Developer'),
(2, 101, 'Project Manager'),
(2, 102, 'Project Manager'),
(3, 104, 'HR Consultant'),
(4, 103, 'Financial Analyst'),
(5, 104, 'Marketing Analyst');
```

Вона слугує проміжною сутністю, що фіксує факт участі конкретного працівника в конкретному проєкті. У фізичній схемі ця таблиця містить атрибути employee_id і project_id, які є зовнішніми ключами та одночасно формують складений первинний ключ, а також атрибут role, що зберігає інформацію про функціональну роль працівника в межах проєкту. Така структура забезпечує гнучкість моделі та дозволяє коректно відображати участь одного працівника в кількох проєктах і залучення кількох працівників до одного проєкту.

2.4 Розробка SQL-запитів для обліку товарів, складу та замовлень

Розробка SQL-запитів для обліку товарів, складських запасів та замовлень є важливим етапом створення інформаційної системи підприємства, оскільки саме на цьому рівні забезпечується практична взаємодія користувача з базою даних. SQL-запити дозволяють здійснювати додавання, оновлення, видалення та вибірку даних, необхідних для контролю руху товарів, стану складу та процесу оформлення замовлень.

Для реалізації обліку товарів використовується таблиця products, яка містить інформацію про номенклатуру продукції, її вартість та основні характеристики. SQL-запити до цієї таблиці забезпечують можливість додавання нових товарів до системи, коригування їхніх параметрів та отримання переліку

наявних позицій. Наприклад, запит на вибірку дозволяє отримати повний список товарів із зазначенням ціни, що використовується під час формування замовлень і аналізу асортименту:

```
SELECT id, name, price
FROM products;
```

Облік складських запасів реалізується за допомогою таблиці warehouse, яка зберігає інформацію про кількість кожного товару на складі. SQL-запити до цієї таблиці дозволяють контролювати залишки продукції, своєчасно виявляти дефіцит товарів і приймати управлінські рішення щодо поповнення складу. Типовим є запит для отримання інформації про товари, кількість яких є меншою за встановлений поріг:

```
SELECT p.name, w.quantity
FROM warehouse w
JOIN products p ON w.product_id = p.id
WHERE w.quantity < 10;
```

Облік замовлень здійснюється з використанням таблиць orders та order_items, які дозволяють зберігати інформацію про кожне замовлення та перелік товарів у його складі. SQL-запити до таблиці orders забезпечують реєстрацію нових замовлень, зміну їхнього статусу та отримання історії замовлень за певний період. Наприклад, запит на вибірку дозволяє отримати список усіх замовлень із зазначенням дати оформлення та поточного стану:

```
SELECT id, order_date, status
FROM orders;
```

Для отримання детальної інформації про склад конкретного замовлення використовуються SQL-запити з об’єднанням таблиць orders, order_items та products. Це дозволяє визначити, які саме товари та в якій кількості були замовлені, а також розрахувати загальну вартість замовлення:

```
SELECT o.id AS order_id,
    p.name AS product_name,
    oi.quantity,
    p.price,
    oi.quantity * p.price AS total_price
FROM order_items oi
JOIN orders o ON oi.order_id = o.id
JOIN products p ON oi.product_id = p.id
WHERE o.id = 1;
```

Окрім операцій вибірки, важливу роль відіграють запити на оновлення даних, зокрема зменшення кількості товарів на складі після оформлення замовлення. Такі SQL-запити забезпечують актуальність інформації про складські запаси та узгодженість даних у системі:

```
UPDATE warehouse
SET quantity = quantity - 2
WHERE product_id = 3.
```

Таким чином, розроблені SQL-запити забезпечують повноцінний облік товарів, контроль складських залишків та управління замовленнями. Їх використання дозволяє автоматизувати основні бізнес-процеси підприємства, підвищити точність обліку та ефективність прийняття управлінських рішень, що є важливою складовою кваліфікаційної роботи.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ ТОВАРІВ

3.1 Створення таблиць та визначення зв’язків між ними

Створення таблиць та визначення зв’язків між ними є ключовим етапом проєктування бази даних для обліку товарів, складу та замовлень. На цьому етапі здійснюється формалізація логічної моделі у вигляді SQL-інструкцій створення таблиць із визначенням первинних і зовнішніх ключів, що забезпечують цілісність та узгодженість даних.

База db-fiddle не будує ER-діаграми автоматично. Він показує зв’язки лише на рівні SQL-коду через FOREIGN KEY. Тому зв’язки “дивляться” або через структуру таблиць, або через SQL-запити.

Таблиця categories призначена для зберігання інформації про категорії товарів. Вона містить атрибут category_id, який є первинним ключем та автоматично збільшується, що забезпечує унікальну ідентифікацію кожної категорії. Атрибут category_name використовується для збереження назви категорії та має обмеження NOT NULL, що унеможливлює створення записів без зазначення назви. Дана таблиця виконує роль довідкової та використовується для класифікації товарів.

Таблиця products використовується для обліку товарів підприємства. Вона містить атрибут product_id як первинний ключ, атрибут product_name для збереження назви товару, а також атрибути price та quantity, які відповідають за ціну та кількість товару. Атрибут category_id у цій таблиці є зовнішнім ключем, що посилається на таблицю categories і забезпечує зв’язок типу багато-до-одного між товарами та категоріями. Це дозволяє кожному товару належати до певної категорії та запобігає введенню некоректних значень категорій.

Таблиця suppliers призначена для зберігання інформації про постачальників. Вона містить атрибут supplier_id як первинний ключ, а також

атрибути supplier_name, phone та email, які зберігають контактні дані постачальника. Ця таблиця використовується під час формування замовлень і дозволяє пов’язати кожне замовлення з конкретним постачальником.

Таблиця warehouse реалізує облік складських запасів. Вона містить атрибут warehouse_id як первинний ключ, атрибут product_id, що є зовнішнім ключем і посилається на таблицю products, а також атрибути quantity та location, які відображають кількість товару на складі та його місце зберігання. Завдяки зовнішньому ключу забезпечується зв’язок між складом і товарами, що дозволяє контролювати залишки кожної товарної позиції.

Таблиця orders використовується для зберігання інформації про замовлення. Вона містить атрибут order_id як первинний ключ, атрибут supplier_id, який є зовнішнім ключем і посилається на таблицю suppliers, а також атрибути order_date та total_amount, що зберігають дату оформлення замовлення і його загальну вартість. Така структура дозволяє пов’язати кожне замовлення з конкретним постачальником та вести облік фінансових операцій.

Таблиця suppliers призначена для зберігання інформації про постачальників, з якими співпрацює підприємство. Атрибут supplier_name використовується для збереження назви постачальника та має обмеження NOT NULL, що унеможливлює створення записів без зазначення назви. Додаткові атрибути phone та email забезпечують зберігання контактної інформації, необхідної для оперативної взаємодії під час формування замовлень і постачання товарів (рис. 3.1). Таблиця warehouse використовується для реалізації обліку складських залишків товарів. Вона містить атрибут warehouse_id як первинний ключ, атрибут product_id, який є зовнішнім ключем і посилається на таблицю products, а також атрибути quantity та location, що відображають кількість товару на складі та місце його зберігання. Використання зовнішнього ключа забезпечує цілісність даних і унеможливлює зберігання інформації про складські залишки для товарів, які відсутні в таблиці products.

```
CREATE TABLE suppliers (
    supplier_id INT AUTO_INCREMENT PRIMARY KEY,
    supplier_name VARCHAR(150) NOT NULL,
    phone VARCHAR(20),
    email VARCHAR(100)
);
CREATE TABLE warehouse (
    warehouse_id INT AUTO_INCREMENT PRIMARY KEY,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    location VARCHAR(100),
    FOREIGN KEY (product_id) REFERENCES
products(product_id)
```

Рисунок 3.1 – Програмний фрагмент формування таблиць suppliers та warehouse_id

Таким чином, створені таблиці та визначені між ними зв’язки формують цілісну реляційну структуру бази даних, яка забезпечує коректний облік товарів, контроль складських запасів і управління замовленнями. Використання первинних і зовнішніх ключів гарантує логічну узгодженість даних та відповідає вимогам проєктування сучасних інформаційних систем.

3.2 Реалізація обліку товарів, постачальників і складських залишків

Реалізація обліку товарів, постачальників і складських залишків у базі даних спрямована на забезпечення повного контролю за рухом матеріальних ресурсів підприємства та актуальністю інформації про наявність товарів на складі. Даний функціональний блок є основою для підтримки процесів закупівлі, зберігання та подальшого використання товарів у господарській діяльності підприємства.

Облік товарів реалізується за допомогою таблиці products, у якій зберігається інформація про номенклатуру продукції, її належність до певної категорії, вартість та кількість. Кожен товар ідентифікується унікальним первинним ключем product_id, що забезпечує однозначний доступ до запису. Зв’язок із таблицею categories через зовнішній ключ category_id дозволяє систематизувати товари за категоріями та спрощує аналіз асортименту. SQL-

запити до цієї таблиці забезпечують можливість додавання нових товарів, оновлення цін і перегляду наявних позицій.

Облік постачальників здійснюється з використанням таблиці suppliers, яка містить інформацію про контрагентів, що забезпечують підприємство товарами. Таблиця включає унікальний ідентифікатор supplier_id, назву постачальника, а також контактні дані, необхідні для оперативної взаємодії. Така структура дозволяє зберігати повну інформацію про постачальників та використовувати її під час формування і аналізу замовлень.

Облік складських залишків реалізується за допомогою таблиці warehouse, яка відображає кількість кожного товару на складі та його місце зберігання. Зовнішній ключ product_id забезпечує зв’язок із таблицею products, що унеможливлює появу записів про складські залишки для неіснуючих товарів. Актуалізація даних у цій таблиці відбувається під час надходження товарів від постачальників або їх списання, що дозволяє підтримувати достовірну інформацію про стан складу в реальному часі.

Взаємодія між таблицями products, suppliers і warehouse забезпечує цілісність обліку та узгодженість даних у системі (рис. 3.2).

Рисунок 3.2 – Схема взаємозв’язків між таблицями products, suppliers і warehouse

Завдяки використанню первинних і зовнішніх ключів реалізується контроль коректності введених даних, а SQL-запити з об’єднанням таблиць дозволяють отримувати зведену інформацію про товари, їх постачальників і наявність на складі. Таким чином, реалізована структура бази даних створює надійну основу для автоматизації процесів управління товарними запасами підприємства.

3.3 Формування звітів і вибірок для аналізу товарообігу

Формування звітів і вибірок для аналізу товарообігу в цій базі даних базується на використанні SQL-запитів з об’єднанням таблиць, агрегатних функцій та умов фільтрації. Оскільки структура БД вже містить логічно пов’язані сутності categories, products, suppliers, warehouse, orders та order_items, це дозволяє отримувати аналітичну інформацію різного рівня деталізації.

Для аналізу асортименту та товарообігу за категоріями використовується зв’язок між таблицями products і categories. На основі цього можна сформувати звіт, який показує кількість товарних позицій у кожній категорії та їх загальну вартість на складі. Такий звіт будується шляхом групування товарів за категоріями та використання агрегатних функцій SUM і COUNT. Він дає змогу оцінити, які категорії є найбільш насиченими та фінансово значущими для підприємства. Приклад вибірки для аналізу вартості товарних запасів за категоріями наведено на рисунку 3.3.

Query SQL

```
SELECT c.category_name,
       COUNT(p.product_id) AS product_count,
       SUM(p.price * p.quantity) AS total_value
FROM categories c
JOIN products p ON c.category_id = p.category_id
GROUP BY c.category_name;

```

Рисунок 3.3 – Код запиту по вартості

Таблиця 3.1 відображає результати запиту:

Таблиця 3.1 – Результати запиту по вартості

category_name	product_count	total_value
Електроніка	2	157000.00
Побутова техніка	1	21600.00
Канцелярія	1	4500.00

Для контролю складських залишків і виявлення дефіцитних позицій використовується зв’язок між таблицями products і warehouse. На основі цієї інформації формуються звіти про фактичну наявність товарів на складі, їх місце зберігання та кількість. Це дозволяє оперативно приймати рішення щодо поповнення запасів. Приклад запиту для формування звіту про складські залишки наведено на рисунку 3.4.

Query SQL

```sql
SELECT p.product_name,
       w.quantity,
       w.location
FROM warehouse w
JOIN products p ON w.product_id = p.product_id;
```

Рисунок 3.4 – Код запиту по залишкам

Для детального аналізу товарообігу за конкретними товарами використовується таблиця order_items, яка фіксує кількість і ціну кожного товару в замовленні. Це дозволяє визначати найбільш затребувані товари та оцінювати динаміку закупівель. Приклад вибірки для аналізу обсягів закупівель товарів – на рисунку 3.5.

Query SQL

```sql
SELECT p.product_name,
       SUM(oi.quantity) AS total_quantity,
       SUM(oi.quantity * oi.price) AS total_sum
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_name;
```

Рисунок 3.5 – Код аналізу обсягів закупівель товарів

Таким чином, формування звітів і вибірок у межах цієї бази даних реалізується шляхом логічного поєднання таблиць і використання SQL-агрегацій. Це забезпечує повноцінний аналіз товарообігу, контроль складських запасів та підтримку управлінських рішень у діяльності підприємства.

3.4 Аналіз отриманих результатів та можливості масштабування системи

Аналіз результатів демонструє, що система дозволяє:

- контролювати товарні запаси – завдяки таблиці warehouse можна відстежувати залишки та оперативно реагувати на дефіцитні позиції;
- аналізувати асортимент та товарообіг – поєднання таблиць products і categories забезпечує оцінку фінансової значущості категорій товарів та ефективності управління асортиментом;
- оцінювати ефективність постачальників – завдяки зв’язку між таблицями orders, suppliers та order_items можна визначати обсяги закупівель, частоту замовлень та надійність контрагентів;
- формувати оперативні та зведені звіти – використання агрегатних функцій та умов фільтрації дозволяє швидко отримувати інформацію для прийняття управлінських рішень.

Система побудована модульно, що забезпечує її масштабованість і подальший розвиток. Можливі напрямки масштабування включають:

- розширення асортименту – додавання нових категорій та товарів без змін у базовій структурі;
- підключення нових складів або відділень – таблиця warehouse легко адаптується для обліку запасів у різних локаціях;
- інтеграція з автоматизованими процесами закупівель та продажів – можна реалізувати синхронізацію даних з ERP-системами або веб-інтерфейсами;

- побудова складніших аналітичних звітів – використання SQL-запитів з об’єднанням кількох таблиць та функцій для прогнозування попиту, аналізу рентабельності та оптимізації запасів;
- впровадження контролю прав доступу та логування дій – забезпечить безпеку та аудит операцій у системі.

Таким чином, отримана база даних не лише забезпечує ефективний поточний облік товарів, постачальників та замовлень, а й створює надійну платформу для подальшого розвитку інформаційної системи підприємства, здатної підтримувати складні бізнес-процеси та масштабуватися відповідно до потреб росту підприємства.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було виконано поставлені завдання.

Проведено аналіз особливостей реляційних баз даних та методів обліку товарів на підприємствах. Було встановлено, що реляційні системи забезпечують цілісність даних, контроль залишків товарів та можливість формування аналітичних звітів для прийняття управлінських рішень. Виявлено, що застосування реляційної моделі є оптимальним для підприємств середнього та великого масштабу, де важлива точність і швидкість обробки інформації.

Виконано порівняльний аналіз існуючих аналогів систем обліку товарів. Було з’ясовано переваги та недоліки різних рішень за функціональністю, масштабованістю та зручністю інтеграції з іншими інформаційними системами. Отримані результати дозволили визначити оптимальні підходи до розробки власної автоматизованої системи, враховуючи потреби підприємства і вимоги до обліку товарообігу.

Проаналізовано методи вирішення проблеми інтеграції та контролю даних. Встановлено, що використання первинних і зовнішніх ключів у базі даних забезпечує логічну узгодженість інформації, унеможливлює створення некоректних записів та гарантує цілісність даних між усіма таблицями системи. Це створює надійну основу для автоматизованого обліку та подальшого масштабування системи.

Розроблено логічну та фізичну схему бази даних підприємства. Було визначено ключові таблиці та їх атрибути, а також зв’язки між ними, що забезпечує ефективне зберігання інформації про товари, постачальників, складські залишки та замовлення. Схема бази даних дозволяє швидко отримувати аналітичну інформацію та підтримує формування звітів різного рівня деталізації.

Реалізовано таблиці та визначено зв’язки між ними для обліку товарів, постачальників, складу та замовлень. Використання зовнішніх ключів дозволяє

контролювати коректність введення даних і забезпечує взаємозв’язок між усіма сутностями системи. Це дозволяє підтримувати актуальну інформацію в режимі реального часу та забезпечує надійну основу для автоматизації управлінських процесів.

Створено SQL-запити для обліку товарообігу та формування аналітичних звітів. Запити забезпечують можливість швидкого отримання інформації про наявність товарів, їх категорії, обсяги закупівель та залишки на складі. Реалізовані вибірки дозволяють ефективно контролювати товарообіг, аналізувати ефективність постачальників та приймати обґрунтовані управлінські рішення.

Аналіз отриманих результатів показав, що розроблена система забезпечує цілісність даних, підтримку управлінських процесів та можливість масштабування. Визначено, що система легко адаптується до збільшення кількості товарів, постачальників та складів, а також інтегрується з іншими інформаційними системами для підвищення ефективності обліку та аналітики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SQL vs NoSQL: 5 Critical Differences. *Integrate.io*. URL: https://www.integrate.io/blog/the-sql-vs-nosql-difference/?utm_source=chatgpt.com/#two (date of access: 15.12.2022).

2. Shareef T., Sharif K., Rashid B. A Survey of Comparison Different Cloud Database Performance: SQL and NoSQL. *Passer Journal of Basic and Applied Sciences*. 2022. Vol. 4, no. 1. P. 45–57.

3. Staniševic I., Vicentić M., Obradovic S. MS SQL server and MySQL response speed comparation. IJEEC - International journal of electrical engineering and computing. 2020. Vol. 4, no. 2. P. 1265-1280

4. DB-Engines Ranking. DB-Engines. URL: https://db-engines.com/en/ranking/relational+dbms (дата звернення: 05.12.2022).

5. Allocation Fragmentation and Replication In Distributed Databases: A Quick Start Guide - Learn | Hevo. Learn | Hevo. URL: https://hevodata.com/learn/fragmentation-and-replication-in-distributed-database/ (дата звернення: 05.12.2022).

6. Distributed DBMS - Quick Guide. Online Tutorials Library. URL: https://www.tutorialspoint.com/distributed_dbms/distributed_dbms_quick_guide.htm (дата звернення: 05.12.2022).

7. Raouf A. E. A., Badr N. L., Tolba M. F. Dynamic Distributed Database over Cloud Environment. Communications in Computer and Information Science. Cham, 2014. C. 67-76.