**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ  БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

(повна назва інституту/факультету)

кафедра   БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

(повна назва кафедри)

|  |  |
| --- | --- |
| **КУРСОВА РОБОТА** | |
|  | |
| з дисципліни (кредитного модуля) | **Системи баз даних** |

|  |  |
| --- | --- |
| спеціальність | ***122 Комп’ютерні науки*** |
|  |  |
| спеціалізація | ***Комп’ютерні технології в біології та медицині*** |

|  |  |
| --- | --- |
| **На тему** | **Облік побутової техніки на складах** |
|  |  |
|  | (тема індивідуального завдання) |

**Виконав студент \_3-го курсу гр. БС-03**

**ЗАТУЛОВСЬКИЙ ГЕОРГІЙ АНДРІЙОВИЧ**

***Засвідчую, що у роботі немає запозичень з праць інших авторів без відповідних посилань.***

|  |  |
| --- | --- |
| **Студент (-ка)** |  |

**\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Перевірив ст.викл. Віталій СЕРДАКОВСЬКИЙ**

**Бали за роботу студента відповідно до РСО за**

|  |  |
| --- | --- |
| Оформлення роботи (із 18 балів) |  |
| Обґрунтування (із 12 балів) |  |
| Захист роботи (із 60 балів) |  |
| ВСЬОГО |  |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |
| --- |
|  |
| (підпис викладачів) |

**Київ – 2022р.**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

|  |  |
| --- | --- |
| Інститут (факультет) | БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ |
|  | (повна назва) |
| Кафедра | БІОМЕДИЧНОЇ КІБЕРНЕТИКИ |
|  | (повна назва) |

**ЗАВДАННЯ**

**на курсову роботу студенту**

|  |  |
| --- | --- |
| **ЗАТУЛОВСЬКИЙ ГЕОРГІЙ АНДРІЙОВИЧ** | |
| (прізвище, ім’я, по батькові) | |
| 1. Тема роботи | **Облік побутової техніки на складах** |
|  | |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| 2. Термін подання студентом роботи | | | ***20 грудня 2022 року*** |
| 3. Вихідні дані до роботи | |  | |
|  | | | |
|  | | | |
| 4. Зміст роботи |  | | |
|  | | | |
|  | | | |

5. Дата видачі завдання  ***20 вересня 2022 р.***

**Календарний план**

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів виконання  курсової роботи | Термін виконання  етапів роботи | Примітка |
| 1 | Отримати завдання на КР | 20 вересня 2022р. |  |
|  | Оформлення розділу з |  |  |
|  | Оформлення розділу з |  |  |
|  | Оформлення розділу з |  |  |
|  | Оформлення КР |  |  |
|  | Подання в електронному вигляді КР та анотації до неї на сайт кафедри. | До 26 грудня 2022р |  |
|  | Подання пакету документів по КР до захисту | 26 грудня 2022р. |  |
|  | Захист КР | 31 грудня 2022р |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | Ім’я ПРІЗВИЩЕ |
|  |  | (підпис) |  | (ініціали, прізвище) |
| Керівник роботи |  |  |  | Віталій СЕРДАКОВСЬКИЙ |
|  |  | (підпис) |  | (ініціали, прізвище) |

**Анотація**

«Курсова робота (Курсовий проект)» з дисципліни «Системи баз даних» являється частиною циклу **Загальної підготовки** дисциплін першого (бакалаврського) рівня вищої освіти ступеня «**бакалавр**», зі спеціальності 122 «Комп’ютерні науки» за спеціалізацією «Комп’ютерні технології в біології та медицині»).

Загальна трудомісткість освоєння модуля становить 1 кредитів (ЕКТС), 30 годин.

Курсову роботу виконав **Затуловський Георгій Андрійович** студент 3 курсу, гр. БС-03 кафедри Біомедичної кібернетики факультету Біомедичної інженерії НТУУ «КПІ ім. Ігоря Сікорського» .

Тема роботи : Розробка бази даних для супроводу даних про ціни на медичну продукцію.

Розглянуті питання: концептуального, логічного та фізичного проектування баз даних.

Результати по роботі:

- придбані навички з проектування та розробки бази даних

- проведено практичне застосування набутих знань шляхом застосування засобів проектування та програмування SQL

- реалізовано поставленні задачі з проектування та розробки відповідної бази даних

Зміст звіту практики:

обсяг роботи 69 ст.; кількість розділів 3; додатків 3; використаних джерел 16; кількість графічних об'єктів та / або додатків містить робота 48 (графіки, схеми, малюнки, таблиці і інший ілюстративний і інформативний матеріал)

Ключові слова: бази даних, проектування, побутова техніка, склад, заказ, корисутвач

**Annotation**

"Coursework (Course Project)" from the discipline "Database Systems" is part of the cycle of **General training** of disciplines of the first (bachelor) level of higher education of the "**Bachelor**" degree, from the specialty 122 "Computer Sciences" with the specialization "Computer Technologies in biology and medicine").

The total complexity of learning the module is 1 credit (ECTS), 30 hours.

The course work was completed by **Zatulovskyi Georgy** Andriyovych, a 3rd-year student, gr. BS-03 of the Department of Biomedical Cybernetics, Faculty of Biomedical Engineering, NTUU "KPI named after Igor Sikorsky".

Work topic: Development of a database to support data on the prices of medical products.

Issues considered: conceptual, logical and physical design of databases.

Work results:

- acquired skills in database design and development

- practical application of acquired knowledge was carried out through the use of SQL design and programming tools

- the set tasks for the design and development of the corresponding database have been implemented

Content of the practice

report: scope of work, 69th century; number of sections 3; applications 3; used sources 16; the work contains 48 graphic objects and/or applications (graphs, diagrams, drawings, tables and other illustrative and informative material)

Keywords: databases, design, household appliances, warehouse, order, user

**ЗМІСТ**

[**СПИСОК СКОРОЧЕНЬ 7**](#_Toc124290518)

[**ВСТУП 8**](#_Toc124290519)

[**РОЗДІЛ 1.**](#_Toc124290520)[**ТЕОРИТИЧНА ЧАСТИНА 9**](#_Toc124290521)

[**1.1. Бази даних 9**](#_Toc124290522)

[**1.2. Cистема управління базами даних 10**](#_Toc124290523)

[**1.2.1. Microsoft SQL Server 11**](#_Toc124290524)

[**1.3. Процес проектування бази даних 12**](#_Toc124290525)

[**1.3.1 Концептуальне проектування 13**](#_Toc124290526)

[**1.3.2. Логічне проектування 13**](#_Toc124290527)

[**1.3.3. Фізичне проектування 15**](#_Toc124290528)

[**1.4. Реляційна логічна модель ІС 16**](#_Toc124290529)

[**1.4.1. Ключі 16**](#_Toc124290530)

[**1.4.2. Нормалізація 17**](#_Toc124290531)

[**1.4.3. Функціональна та багатозначна залежність 19**](#_Toc124290532)

[**1.5. Представлення 19**](#_Toc124290533)

[**1.6. Тригери 20**](#_Toc124290534)

[**1.7. Індекси 22**](#_Toc124290535)

[**Висновок до розділу 1 24**](#_Toc124290536)

[**РОЗДІЛ 2.**](#_Toc124290537)[**АНАЛІТИЧНА ЧАСТИНА 25**](#_Toc124290538)

[**2.1 Актуальність розроблюваної бази даних 25**](#_Toc124290539)

[**2.2 Діаграма “Сутність-зв’язок” 25**](#_Toc124290540)

[**2.3 Бізнес-процеси предметної області 26**](#_Toc124290541)

[**2.4. Морфологічний аналіз 30**](#_Toc124290542)

[**2.5 Реляційна логічна модель ІС 31**](#_Toc124290543)

[**2.5.1. Перелік атрибутів 31**](#_Toc124290544)

[**2.5.2. Декомпозиція 33**](#_Toc124290545)

[**Висновок до розділу 2 35**](#_Toc124290546)

[**РОЗДІЛ 3.**](#_Toc124290547)[**ПРАКТИЧНА ЧАСТИНА 36**](#_Toc124290548)

[**3.1. Фізична модель 36**](#_Toc124290549)

[**3.1.1. Оператори створення таблиць 36**](#_Toc124290550)

[**3.1.2. Створення коментарів 38**](#_Toc124290551)

[**3.1.3 Опис обмежень 39**](#_Toc124290552)

[**3.2. Представлення 42**](#_Toc124290553)

[**3.3. Тригери 45**](#_Toc124290554)

[**3.4. Індекси 52**](#_Toc124290555)

[**3.4.1. Плани виконання запитів без використання індексів 52**](#_Toc124290556)

[**3.4.2. Плани виконання запитів з використанням індексів 56**](#_Toc124290557)

[**Висновок до розділу 3 60**](#_Toc124290558)

[**ВИСНОВОК 61**](#_Toc124290559)

[**СПИСОК ВИКОРИЧТАНИХ ДЖЕРЕЛ 62**](#_Toc124290560)

[**ДОДАТОК A 64**](#_Toc124290561)

[**ДОДАТОК Б 67**](#_Toc124290562)

[**ДОДАТОК В 68**](#_Toc124290563)

# 

# 

# **СПИСОК СКОРОЧЕНЬ**

БД – база даних

Рис. – рисунок

Табл. – таблиця

СУБД – система управління базами даних

ФЗ – функціональна залежність

БЗ – багатозначна залежність

ІС – інформаційна система

# **ВСТУП**

У ції роботі ми створюємо базу даних для отримання та зберігання інформації о користувачів, їх заказів та наявність побутовуї техніки на складі. Цей додаток повинен спростити роботу та запобігти виникнення неточностей, адже бази даних влаштовані так, щоб вони могли легко та швидко знаходить, записувати, переписувати та знову знаходить дані. Такі додатки є невід’ємною частиною бізнесу та застосовуються у всіх сферах життя людини.

Курсова робота передбачає розробку бази даних котра буде робити облік побутової техніки на складах. Розробка такої бази даних буде спрощувати контроль над активними заказами та вмістом певної побутової техніки на складі. Це допоможе своєчасно відправляти товар заказнику та комірникам поповнювати у постачальників. Тому для цього треба зберігати інформацію о користувачів (логіни, паролі, тип користувача), заказів(номер заказу, кількість та загальну ціну) та побутової техніки (її номер, ціну, загальну характеристику).

Основна мета цієї курсової роботи це закріплення знань та навичок в області систем баз даних та мови SQL, його плануванням, розробкою, проектуванням та реалізацією.

# **РОЗДІЛ 1.**

# **ТЕОРИТИЧНА ЧАСТИНА**

## Бази даних

База даних – це сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім самих даних, містить їх опис та може містити засоби для їх обробки.

Проектування баз даних:

Зазвичай з базами даних працюють дві категорії користувачів. Перша категорія – проектувальники. Їх завдання полягає в розробці структури таблиць бази даних і узгодження її із замовником. Окрім таблиць проектувальники розробляють і інші об'єкти бази даних, призначені, з одного боку, для автоматизації роботи з базою, а з іншого боку – для обмеження функціональних можливостей роботи з базою (якщо це необхідно з міркувань безпеки).

Проектувальники не наповнюють базу конкретними даними (замовник може вважати їх конфіденційним і не надавати стороннім особам). Виняток становить експериментальне наповнення модельними даними на етапі відладки об'єктів бази. Друга категорія виконавців, що працюють з базами даних, - користувачі. Вони отримують вихідну базу даних від проектувальників і займаються її наповненням і обслуговуванням. У загальному випадку користувачі не мають засобів доступу до управління структурою бази – лише до даних, да і то не до всіх, а до тих, робота з якими передбачена на конкретному робочому місці.

Відповідно СУБД має два режими роботи: проектувальний і призначений для користувача.

Перший режим призначений для створення або зміни структури бази і створення її об'єктів. У другому режимі відбувається використання раніше підготовлених об'єктів для наповнення бази або здобуття даних з неї.

## Cистема управління базами даних

Система управління базами даних (СУБД) — це комплекс програмних засобів, призначений для інтегрованого зберігання та обробки даних. За загальне управління системою бази даних відповідає адміністратор бази даних, котрий виконує такі функції: • визначення інформаційного змісту бази даних; • визначення структури зберігання даних; • взаємодія з користувачами; 382 • забезпечення перевірки достовірності інформації; • визначення повноважень доступу; • визначення методів архівації та поновлення даних; • управління ефективністю функціонування СУБД.

Програмні компоненти середовища СУБД:

* процесор запитів: перетворює запити у послідовність низькорівневих інструкцій для контролера бази даних;
* контролер бази даних: взаємодіє з запущеними користувачами прикладними програмами і запитами;
* контролер файлів: маніпулює призначеними для зберігання даних файлами та відповідає за розподіл доступного дискового простору, створює і підтримує список структур та індексів, визначених у внутрішній схемі, викликає функції хешування для генерації адрес записів, але не керує фізичним вводом та виводом даних безпосередньо;
* препроцесор мови DML: перетворює впроваджені в прикладні програми DML-оператори у викликах стандартних функцій базової мови;
* компілятор мови DDL: перетворює команди в набір таблиць, що містять метадані;
* контролер словника: управляє доступом до системного каталогу і забезпечує роботу з ним.

### 1.2.1. Microsoft SQL Server

MicrosoftSQLServer — система управління базами даних, яка розробляється корпорацією Microsoft. Як сервер даних виконує головну функцію по збереженню та наданню даних у відповідь на запити інших застосунків, які можуть виконуватися як на тому ж самому сервері, так і у мережі. Мова, що використовується для запитів — Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних

Основні переваги Microsoft SQL Server:

* Масштабування системи. Взаємодіяти з нею можна як на простих ноутбуках, так і на ПК з потужним процесором, здатним обробляти великий обсяг запитів.
* Підтримка постійної пам'яті (РМЕМ). Це швидка пам'ять, що має можливість зберігати дані після відключення живлення. Вона дозволяє обробляти дані in-memory, позбавляючи необхідності передавати дані по каналах передачі і прискорюючи обробку запитів на 30% для інтенсивних робочих навантажень введення-виведення.
* Гібридна транзакційна/аналітична обробка (НТАР). Модель HTAP дозволяє одночасно здійснювати операційні транзакції та аналітику на тих самих даних в одній і тій же пам'яті, також реалізуючи підхід in memory.
* Інтелектуальна обробка запитів Паралелізація запитів та покращене масштабування частих запитів завдяки механізмам інтелектуальної обробки запитів роблять продуктивність значно вищою. Відкладена компіляція табличних змінних більш ніж 50% прискорює обробку запитів.
* Безпека та відповідність вимогам Захист конфіденційних даних за допомогою технології Always Encrypted із захищеними анклавами. Шифрування дома дозволяє виконувати криптографічні операції з конфіденційними даними без їх переміщення межі бази даних.
* Azure Data Studio (колишній SQL Operations Studio) – це спрощений кросплатформовий графічний засіб керування та редактор коду. Дозволяє створювати запити до реляційних та нереляційних баз даних за допомогою різних операційних систем та джерел даних. Azure Data Studio дозволяє підключатися до SQL Server локально та у хмарі, у Windows, macOS та Linux.
* Інтелектуальний аналіз даних Правильний аналіз та ефективне подання результатів безпосередньо впливає на ефективність аналізу даних та можливість приймати на їх основі управлінські рішення.

## Процес проектування бази даних

Проектування бази даних - це ітераційний процес, що має свій початок, але не має кінця й складається з нескінченного ряду уточнень. Його варто розглядати, насамперед, як процес пізнання. Як тільки проектувальник приходить до розуміння роботи підприємства й змісту оброблюваних даних, а також виражає це розуміння засобами обраної моделі даних, придбані знання можуть показати, що потрібне уточнення й в інших частинах проекту. Особливо важливу роль у загальному процесі успішного створення системи грає концептуальне й логічне проектування бази даних. Якщо на цих етапах не вдасться одержати повне розуміння про діяльність підприємства, то задача визначення всіх необхідних користувальницьких подань або забезпечення захисту бази даних стає надмірно складної або навіть нездійсненною. До того ж може виявитися скрутним визначення способів фізичної реалізації або досягнення прийнятної продуктивності системи. З іншого боку, здатність адаптуватися до змін є одним з ознак удалого проекту бази даних. Тому має сенс затратити час і енергію, необхідні для підготовки найкращого можливого проекту.

### 1.3.1 Концептуальне проектування

Концептуальне проектування бази даних – це процес створення моделі, використовуваної на підприємстві інформації, що не залежить від будь-яких фізичних аспектів її представлення. Він полягає в створенні концептуальної моделі даних для аналізованої частини підприємства. Ця модель даних створюється на основі інформації, записаної в специфікаціях вимог користувачів. Концептуальне проектування бази даних абсолютно не залежить від таких подробиць її реалізації як тип обраної цільовий СУБД, набору створюваних прикладних програм, використовуваних мов програмування, типу обраної обчислювальної платформи, а також від будь-яких інших особливостей фізичної реалізації. При розробці концептуальна модель даних постійно піддається тестуванню й перевірці на відповідність вимогам користувачів. Створена концептуальна модель даних підприємства є джерелом інформації для етапу логічного проектування бази даних.

### 1.3.2. Логічне проектування

Логічне проектування бази даних - це процес створення моделі використовуваної на підприємстві інформації на основі обраної моделі організації даних, але без врахування типу цільовий СУБД і інших фізичних аспектів реалізації. Логічне проектування полягає у визначенні числа й структури таблиць, формування запитів до БД, визначенні типів звітних документів, розробці алгоритмів обробки інформації, створенні форм для уведення й редагування даних у базі й рішенні ряду інших задач. Його ціль складається в створенні логічної моделі даних для досліджуваної частини підприємства. Концептуальна модель даних, створена на попередньому етапі, уточнюється й перетвориться в логічну модель даних. Логічна модель даних ураховує особливості обраної моделі організації даних у цільовий СУБД (наприклад, реляційна модель). Якщо концептуальна модель даних не залежить від будь-яких фізичних аспектів реалізації, то логічна модель даних створюється на основі обраної моделі організації даних цільовий СУБД. Інакше кажучи, на цьому етапі вже повинне бути відомо, яка СУБД буде використовуватися в якості цільовій - реляційна, мережна, ієрархічна або об’ектно-орієнтована. Однак на цьому етапі ігноруються всі інші характеристики обраної СУБД, наприклад, будь-які особливості фізичної організації її структур зберігання даних і побудова індексів. У процесі розробки логічна модель даних постійно тестується й перевіряється на відповідність вимогам користувачів. Для перевірки правільносгі логічної моделі даних використовується метод *нормалізації.*Нормалізація гарантує, що відносшення, виведені з існуючої моделі даних, не будуть мати надмірність даних, здатної викликати порушення в процесі відновлення даних після їхньої фізичної реалізації. Створена логічна модель даних є джерелом інформації для етапу фізичного проектування й забезпечує розроблювача фізичної бази даних засобами пошуку компромісів, необхідних для досягнення поставлених цілей, що дуже важливо для ефективного проектування. Логічна модель даних грає також важливу роль на етапі експлуатації й супроводу вже готової системи. При правильно організованому супроводі підтримувана в актуальному стані модель даних дозволяє точно й наочно представити будь-які внесені в базу дані зміни, а також оцінити їхній вплив на прикладні програми й використання даних, уже наявних у базі.

### 1.3.3. Фізичне проектування

Фізичне проектування бази даних – це процес підготовки опису реалізації бази даних на вторинних запам'ятовувальних пристроях; на цьому етапі розглядаються основні відношення, організація файлів і індексів, призначених для забезпечення ефективного доступу до даних, а також усе пов'язане із цим обмеження цілісності й засоби захисту. Під час попереднього етапу проектування була визначена логічна структура бази даних (яка описує відношення й обмеження в розглянутій прикладній області). Хоча ця структура не залежить від конкретної цільової СУБД, вона створюється з урахуванням обраної моделі зберігання даних, наприклад реляційної, мережної або ієрархічної. Однак, приступаючи до фізичного проектування бази даних, насамперед, необхідно вибрати конкретну цільову СУБД. Тому фізичне проектування нерозривно пов'язане з конкретної СУБД. Між логічним і фізичним проектуванням існує постійний зворотний зв'язок, тому що рішення, прийняті на етапі фізичного проектування з метою підвищення продуктивності системи, здатні вплинути на структуру логічної моделі даних.

Як правило, основною метою фізичного проектування бази даних є опис способу фізичної реалізації логічного проекту бази даних. У випадку реляційної моделі даних під цим мається на увазі наступне:

створення набору реляціних таблиць і обмежень для них на основі інформації, представленої в глобальній логічній моделі даних;

визначення конкретних структур зберігання даних і методів доступу до них, що забезпечують оптимальну продуктивність СУБД;

розробка засобів захисту створюваної системи.

Етапи концептуального й логічного проектування великих систем варто відокремлювати від етапів фізичного проектування. На це є кілька причин.

Вони пов'язані із зовсім різними аспектами системи, оскільки відповідають на запитання, *що*робити, а не *як*робити.

Вони виконуються в різний час, оскільки зрозуміти, *що*треба зробити, треба перш, ніж вирішити, *як*це зробити.

Вони вимагають зовсім різних навичок і досвіду, тому вимагають залучення фахівців різного профілю.

## Реляційна логічна модель ІС

Реляційна логічна модель — це тип логічної моделі, яка представляє зв’язки між різними елементами або факторами в системі. Він часто використовується для візуалізації та аналізу взаємозв’язків між різними змінними чи факторами, які можуть впливати на конкретну проблему чи проблему.

У контексті інформаційних систем (ІС) модель реляційної логіки може бути використана для розуміння зв’язків між різними компонентами системи та того, як вони взаємодіють один з одним. Наприклад, модель реляційної логіки може бути використана для представлення зв’язків між різними таблицями бази даних або для розуміння потоку даних через різні системи.

Модель реляційної логіки зазвичай представлена як діаграма або графічне представлення, яке показує зв’язки між різними елементами в системі. Його можна використовувати для виявлення потенційних проблем або областей для вдосконалення системи, а також для розробки стратегій вирішення цих проблем.

### 1.4.1. Ключі

**Ключ** – це стовпець (може бути декілька стовпців), що додається до таблиці і дозволяє встановити зв’язок із записами в іншій таблиці. Існують ключі двох типів: первинні і вторинні (зовнішні).

**Первинний ключ** – це одне або кілька полів (стовпців), комбінація значень яких однозначно визначає кожний запис у таблиці. Первинний ключ не допускає значень Null і завжди повинен мати унікальний індекс. Первинний ключ використовується для зв’язування таблиці з зовнішніми ключами в інших таблицях.

**Зовнішній (вторинний) ключ** – це одне або кілька полів (стовпців) у таблиці, що містять посилання на поле або поля первинного ключа в іншій таблиці. Зовнішній ключ визначає спосіб об’єднання таблиць.

З двох логічно пов’язаних таблиць одну називають таблицею первинного ключа або головною таблицею, а іншу таблицею вторинного (зовнішнього) ключа або підпорядкованою таблицею. СУБД дозволяють зіставити споріднені записи з обох таблиць і спільно вивести їх у формі, звіті або запиті. Існує три типи первинних ключів: ключові поля лічильника (лічильник), простий ключ і складовий ключ.

* Поле лічильника (Тип даних «Счетчик»). Для кожного запису цього поля таблиці автоматично заноситься унікальне числове значення.
* **Простий ключ.** Якщо поле містить унікальні значення, такі як коди чи інвентарні номери, то це поле можна визначити як первинний ключ. В якості ключа можна визначити всі поля, що містить дані, якщо це поле не містить повторювані значення або значення Null.
* **Складові ключі.** У випадках, коли неможливо гарантувати унікальність значень кожного поля, існує можливість створити ключ, що складається з *декількох полів.* Найчастіше така ситуація виникає для таблиці, використовуваної для зв’язування двох таблиць відношенням “багато – до – багатьох”.

### 1.4.2. Нормалізація

Нормалізація схеми бази даних — покроковий процес розбиття одного відношення (на практиці: таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей.

Нормальна форма — властивість відношення в реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення.

Таким чином, схема реляційної бази даних переходить у першу, другу, третю і так далі нормальні форми. Якщо відношення відповідає критеріям нормальної форми *n* та всіх попередніх нормальних форм, тоді вважається, що це відношення знаходиться у нормальній формі рівня *n*.

До нормальних форм належать

* НФБК – нормальна форма Бойса-Кодда використовуэться в нормалізації баз даних та являється трохи сильнішою версією третьої нормальної форми. Таблиця знаходиться в НФБК тоді і тілько тоді, коли для кожної її нетривіальної функціональної залежності X→Y, X це суперключ, тобто Х, або потенційний ключ, або його надмножина.
* Нормальна форма домен/ключ. Ця нормальна форма вимагає, аби в схемі не було інших обмежень окрім ключів та доменів.
* Перша нормальна форма (1НФ, 1NF) утворює ґрунт для структурованої схеми бази даних:
* Друга нормальна форма (2НФ, 2NF) вимагає, аби дані, що зберігаються в таблицях із композитним ключем, не залежали лише від частини ключа:
* Третя нормальна форма (3НФ, 3NF) вимагає, аби дані в таблиці залежали винятково від основного ключа:
* Четверта нормальна форма (4НФ, 4NF) потребує, аби в схемі баз даних не було нетривіальних багатозначних залежностей множин атрибутів від будь чого, окрім надмножини ключа-кандидата.
* П'ята нормальна форма (5НФ, 5NF, PJ/NF) вимагає, аби не було нетривіальних залежностей об'єднання, котрі б не витікали із обмежень ключів.
* Шоста нормальна форма (6НФ, 6NF).Таблиця знаходиться у 6NF, якщо вона знаходиться у 5NF та задовольняє вимозі відсутності нетривіальних залежностей. Зазвичай 6NF ототожнюють з DKNF.

### 1.4.3. Функціональна та багатозначна залежність

ФЗ (Функціональна залежність) – бінарне відношення між множинами атрибутів даного відношення і є, по суті, зв’язком «один-до-багатьох». Функціональна залежність виконує роль основи для наукового підходу до розв’язання проблем

БЗ (багатозначна залежність )– повне обмеження між двома множинами атрибутів у відношенні. На відміну від функціональної залежності, багатозначна залежність вимагає наявність певних кортежів у відношенні. Отже, багатозначна залежність це особливий випадок кортеж-твірної залежності. Поняття багатозначної залежності використовується при визначенні четвертої нормальної форми.

## Представлення

Представлення – це динамічний результат однієї або декількох реляційних операцій, виконаних над відношеннями БД для одержання нового відношення.

Представлення бази даних - результат надісланих запитів до даних, які містяться в таблиці бази даних. На відміну від звичайних таблиць у реляційній базі даних, представлення є віртуальною таблицею, розрахованою або зібраною динамічно з даних BD при запиті доступу до неї. Зміни, застосовані до даних відповідної базової таблиці, відтворюються на даних, які будуть показані у разі виклику подання.

Переваги подання над таблицями:

* Представлення можуть надати підмножини даних, які містяться у таблиці.
* Представлення можуть об'єднати та збільшити безліч таблиць в одну віртуальну таблицю.

Подання не потрібно багато місць; база даних містить лише визначення подання, а не копію всіх даних, які вона надає.

Ви можете маніпулювати вкладеними уявленнями, оскільки одне уявлення може зібрати дані з інших уявлень.Дані представлені не впорядковані. Представлення є реляційною таблицею, відповідно, надає лише набір даних. Однак ви можете отримати відсортовані дані з подання таким же чином, як ви це робитье, працюючи з будь-якою іншою таблицею — як частина запиту для цього подання.робите, працюючи з будь-якою іншою таблицею — у якості частини запиту для цього представлення.

## Тригери

Тригер – це збережена процедура спеціального типу, що автоматично виконується при настанні заданої події. Тригери не містять явних вхідних або вихідних параметрів і не можуть бути запущені явно. SQL Server підтримує три класи тригерів: тригери мови обробки даних (тригери DML), тригери мови опису даних (тригери DDL) та тригери входу до системи.

Тригери DML запускаються при спробі змінити дані в таблиці або поданні за допомогою операторів INSERT, UPDATE, DELETE або їх сукупність. Тригери DML у ході роботи мають доступ до спеціальних таблиць inserted і deleted, що зберігає результати змін, зроблених оператором, що викликав тригер. У табл. 1.1 наведено дані про те, що саме в цих таблицях зберігається залежно від оператора, для якого визначено тригер.

Табл. 1.1

|  |  |  |
| --- | --- | --- |
| Оператор | Таблиця | Зміст таблиці |
| INSERT | inserted | Вставлені в таблицю нові рядки |
| DELETE | deleted | Віддалені рядки |
|  | inserted | Новий стан змінених рядків |
| UPDATE | deleted | Старий стан змінених рядків |

За допомогою тригерів можна реалізувати перевірку складних логічних умов, виконання яких забезпечує несуперечність даних. Наприклад, потрібно при внесенні запису в таблицю перевірити наявність пов'язаних даних в іншій таблиці, що знаходиться в іншій БД.

Тригер запускається сервером автоматично при спробі зміни даних у таблиці, з якою він пов'язаний. Всі здійснені ним модифікації даних розглядаються як виконані в транзакції, в якій виконано дію, що викликало спрацьовування тригера. Відповідно, у разі виявлення помилки або порушення цілісності даних може відбутися відкат цієї транзакції.

Момент запуску тригера визначається за допомогою ключових слів BEFORE (тригер запускається до виконання пов'язаного з ним події; наприклад, до додавання запису) або AFTER (після події). У випадку, якщо тригер викликається до події, він може внести зміни у модифікований подією запис (звичайно, за умови, що подія — не вилучення запису). Деякі СУБД накладають обмеження на оператори, які можуть бути використані в тригері (наприклад, може бути заборонено вносити зміни в таблицю, на якій «висить» тригер тощо).

Крім того, тригери можуть бути прив'язані не до таблиці, а до розрізу (VIEW). В цьому випадку за їхньої допомоги реалізується механізм «оновлюваного виду». В цьому випадку ключові слова BEFORE і AFTER впливають лише на послідовність виклику тригерів, бо власне подія (вилучення, вставка або оновлення) не відбувається.

## Індекси

Індекси – це спеціальні таблиці, які можна використовувати пошуковим двигуном бази даних (далі – БД), прискорення отримання даних. Необхідно просто додати індекс індексу в таблицю. Індекс у БД вкрай схожий з індексом наприкінці книги.

Припустимо, ми хочемо мати посилання на всі сторінки книги, які стосуються певної теми. Для цього ми в першу чергу посилаємося на індекс, який вказано в кінці книги і переходимо на будь-яку зі сторінок, яка стосується необхідної теми. Індекс допомагає прискорити запити на отримання даних (SELECT [WHERE]), але уповільнює процес додавання та зміни записів (INSERT, UPDATE). Індекси можуть бути додані або видалені без впливу на дані.

Для того, щоб додати індекс, нам необхідно використовувати команду CREATE INDEX, що дозволить нам вказати ім'я індексу і визначити таблицю і колонку або індекс колонки і визначити, чи використовується індекс зростання або спадання.

Індекс також можуть бути унікальними, так само як і констрейнт UNIQUE. У цьому випадку індекс запобігає додаванню даних, що повторюються, в колонку або комбінацію колонок, на які вказує індекс.

Індекс також можуть бути унікальними, так само як і констрейнт UNIQUE. У цьому випадку індекс запобігає додаванню даних, що повторюються, в колонку або комбінацію колонок, на які вказує індекс.

Індекси використовуються в трьох основних випадках:

* Прискорення виконання запитів. Індекси створюються для полів, які використовуються в умовах пошуку SQL-запитів.
* Забезпечення унікальності значень в полях. Обмеження первинного ключа вимагає, щоб в усій таблиці не знайшлося двох однакових значень полів, що входять до первинного ключа. Щоб виконати цю умову, необхідно при кожній вставці нового запису здійснювати пошук такого ж значення, яке буде вставлене.
* Забезпечення цілісності посилань. Обмеження зовнішніх ключів Foreign key використовується для перевірки того, щоб значення, які вставляються до таблиці, обов’язково існували в іншій таблиці. При створенні зовнішнього ключа автоматично створюється індекс, який використовується як для прискорення запитів, що використовують з’єднання таблиць, так і для перевірки умов зовнішнього ключа.

Індекси використовуються для збільшення продуктивності БД, але є випадки, коли нам варто уникати їх використання:

* Не слід використовувати індекси для невеликих таблиць.
* Не варто використовувати індекси для таблиць, в яких, як передбачається, часто додаватимуться нові дані, або ці дані будуть змінюватися.
* Не варто використовувати індекс для колонок, з якими проводитимуться часті маніпуляції.
* Не варто використовувати індекси для стовпчиків, які мають багато значень NULL.

## Висновок до розділу 1

У першому розділі нашої курсової роботи ми розібрали всю теоретичну частину та проаналізували інформацію котру будемо застосовувати у створені нашої бази даних.

# **РОЗДІЛ 2**

# **АНАЛІТИЧНА ЧАСТИНА**

## 2.1 Актуальність розроблюваної бази даних

База даних, яка розробляється у цій курсовій роботі на тему обліку побутової техніки на складах є популярною та потрібною у багатьох компаніях світу. Всі магазини мають свою базу даних, котра допомогає регулювати наявність та потребу певного товару. Це призводить до того, що зменшується загальний обсяг роботи та кількість часу для її виконання. Цей факт насам перед підвищує ефективність самої компанії, збільшує кількість продажу товарів та їх обсягу.

У цьому програмному додатку користувач може створувати заказ котрий буде мати номер цього заказу, ім’я заказника, загальну ціну товару, кількість товару та докладну характеристику побутової техніки котру заказують. Крім цього користувач має можливість змінювати всю інформацію о товарах та заказниках. Це допоможе в організації складу та надасть докладну інформацію о побутовій техниці, котра користується великим попитом.

## 2.2 Бізнес-процеси предметної області

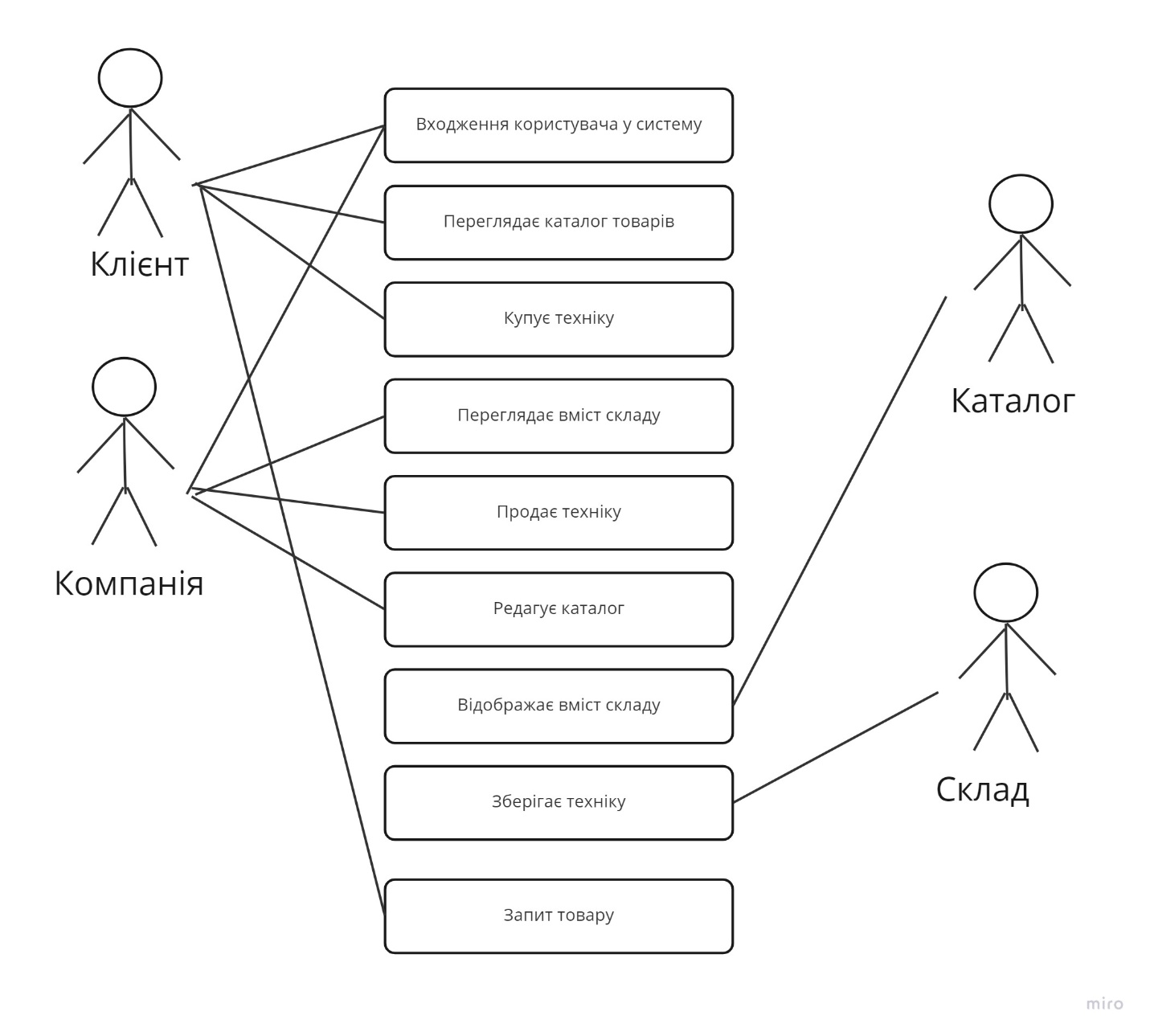
****

Рис. 2.1 - Діаграми “прецендентів“

Перелік використаних прецедентів:

* Входження користувача у систему
* Клієнт переглядає каталог товарів
* Клієнт купує техніку
* Компанія переглядає вміст складу
* Компанія продає техніку
* Компанія редагує каталог
* Каталог відображає вміст складу
* Склад зберігає техніку
* Запит товару

Опис прецедентів використання в стислому форматі:

1. Прецедент: Входження користувача у систему
   1. Користувач – Людина

Мета: Входження в систему або створення нового обліку для подальшої роботи з нею

Завдання: Надати особисті дані (ПІБ, пароль, електронна пошта, номер телефону)

1. Прецедент: Клієнт переглядує каталог товарів
   1. Користувач – Клієнт

Мета: Надання можливості клієнту познайомитись з асортиментом складу

Завдання: Показати інформацію про кількість, наявність та ціну товару

1. Прецедент: Клієнт купує техніку
   1. Користувач – Клієнт

Мета: Купівля товару

Завдання: Створення ордеру на побутову техніку (ціни, кількість), що знаходиться на складі

1. Прецедент: Компанія переглядає вміст складу
   1. Користувач – Компанія

Мета: Перегляд кількості побутової техніки на складі

Завдання: Перевірка складу на наявність техніки

1. Прецедент: Компанія продає техніку
   1. Користувач – Компанія

Мета: Продаж техніки зі складу

Завдання: Отримання інформації про побутову техніку та продаж клієнту за замовленням

1. Прецедент: Компанія редагує каталог
   1. Користувач – Компанія

Мета: Редагування каталогу товарів

Завдання: Змінення інформації товару, її кількість, наявність та ціну

1. Прецедент: Каталог відображає вміст складу
   1. Користувач – Каталог

Мета: Перегляд складу на наявність товару

Завдання: Перевірка кількості товару на складі

1. Прецедент: Склад зберігає техніку
   1. Користувач – Склад

Мета: Збереження побутової техніки

Завдання: Зберігати інформацію про побутову техніку (назву, тип, виробник, модель, дата)

1. Прецедент: Запит товару
   1. Користувач – Клієнт

Мета: Продати товар

Завдання: Виставити запит на купівлю товару

## 2.4. Морфологічний аналіз

Провидемо морфологічний аналіз текстів з виділенням кандидатів сутностей, зв'язків та атрибутів:

Аналіз іменників:

* Людина – сутність, синонім до “користувача”
* Пароль – атрибут “Людина”
* Ім’я – атрибут “Людина”
* Компанія – сутність
* Клієнт – сутність
* Корзина – атрибут “Клієнт”
* Запит – сутність
* Склад – сутність
* Побутова техніка – сутність
* Каталог - сутність
* Ціна – атрибут “Запит”, “Товар” та “Каталогу”
* Кількість – атрибут “Склад ” та “Замовлення”
* Назва - атрибут “Побутовий товар” та “Компанія”
* Модель - атрибут “Побутовий товар”
* Тип - атрибут “Побутовий товар”
* ID товару - атрибут “Побутовий товар” та “Замовлення”
* ID заказу – атрибут “Замовлення”
* ID аккаунту – атрибут “Компанії” ,”Клієнта” та “ Замовлення”
* Виробник - атрибут “Побутовий товар”
* Розмір - атрибут “Побутовий товар”

Дієслова в тексті та їх аналіз:

Надання людиною особистих даних (для регістрації нового обліку: ПІБ, пароль, електронна пошта, номер телефону; для входження у систему: ПІБ, пароль)

* Клієнт переглядає каталог (ціни, кількість)
* Компанія продає побутову техніку
* Клієнт купує побутову техніку (указує кількість товару, отримує ціну)
* Зберігання побутової техніки на складі (кількість товару на складі)
* Замовлення містить побутову техніку (для подальшого продажу клієнту)
* Відображення товарів складу
* Компанія редагує каталог (змінює параметри товарів: ціна)
* Компанія переглядає склад

## 2.3 Діаграма “Сутність-зв’язок”

На рис. 2.1 зображенна діаграма “Сутність-зв’язок” нашої бази даних:

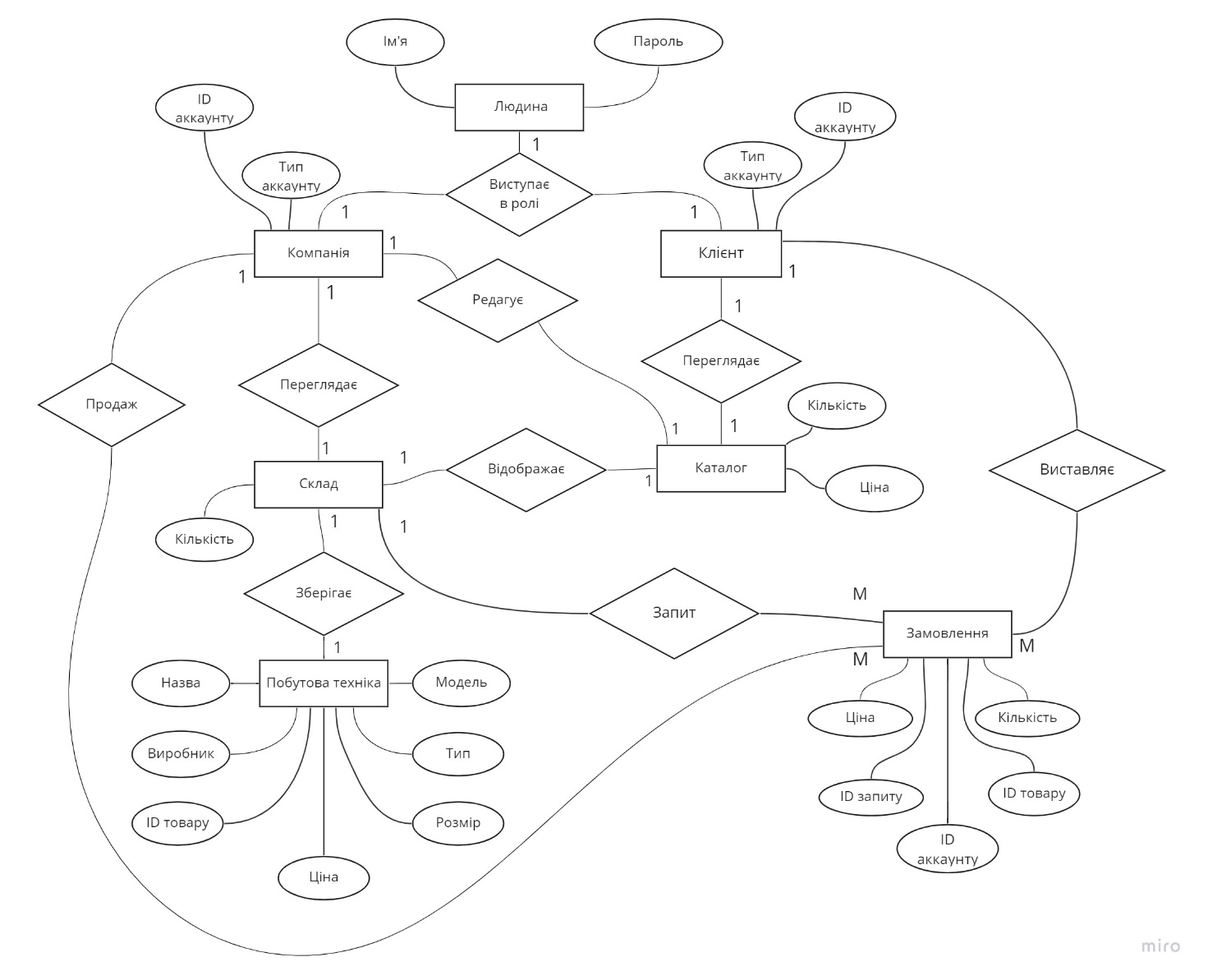


Рис. 2.2 - Діаграма «сутність зв'язок» бази даних “Облік побутової техніки на складах”

## 2.4 Реляційна логічна модель ІС

### 2.4.1. Перелік атрибутів

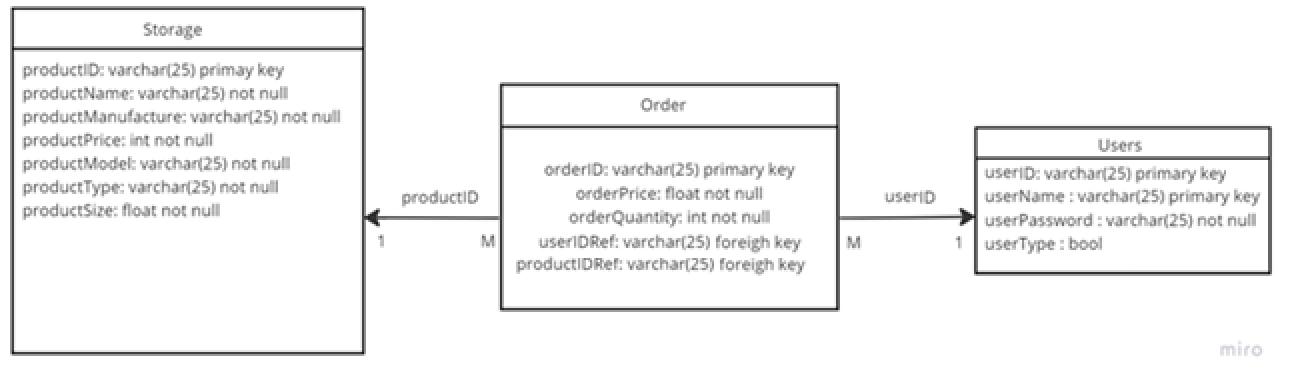


Рис. 2.3 - Діаграма класів UML етапу логічного проектування

Перелік атрибутів:

* userName – ім’я аккаунту
* userPassword – пароль аккаунту
* userType – тип аккаунту
* userID – ID аккаунту
* productName – назва побутової техніки
* productManufacture – виробник побутової техніки
* productID – ID побутової техніки
* productPrice – ціна побутової техніки
* productModel – модель побутової техніки
* productType – тип побутової техніки
* productSize – вага побутової техніки
* quantityProduct – кількість побутової техніки
* orderPrice – ціна заказу
* orderQuantity – кількість заказаної побутової техніки
* orderID – ID заказу
* userIDRef – ID аккаунту, що створив запит
* productIDRef – ID побутової техніки, що містить запит

Опис атрибутів:

Побутова техніка, яка має свою назву, виробника, ціну, модель, тип, розмір та кількість.

productID → productName(1), productManufacture(2) , productPrice(3), productModel(4) , productType(5), productSize(6), quantityProduct(7)

Аккаунт має свій пароль та тип доступу.

userID→ userName(8), userPassword(9), userType(10)

Кожен заказ містить в собі загальну ціну ордеру, номер товару та кількість покупляємої побутової техніки.

orderID –orderPrice(11), orderQuantity(12), userIDRef(13), productIDRef(14)

### 2.4.2. Декомпозиція

Проведемо декомпозицію нашої діаграми "сутність-зв'язок" етапу концептуального проектування:

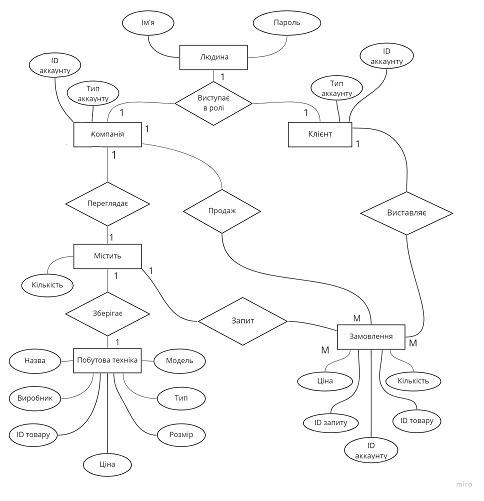


Рис. 2.4 – Декомпозиція діаграми "сутність-зв'язок"

Проведення декомпозиції:

R**(**userName, userPassword, userType, userID, productName, productManufacture, productID, productPrice, productModel, productType, productSize, quantityProduct, orderID, productID, orderPrice, orderQuantity, userIDRef, productIDRef**)**

Ключ userID= {userName, userPassword, userType, productID, productName, productManufacture, productPrice, productModel, productType, productSize, quantityProduct, orderID , orderPrice, orderQuantity, userIDeRef, productIDRef }

ФЗ 1-4 на R1

R1(userID,userName,userPassword, userType)

R2(productID, productName, productManufacture, productPrice, productModel, productType, productSize, quantityProduct, orderID , orderPrice, orderQuantity, userIDRef, productIDRef)

Ключ на R1: userID ФЗ 1-4

Ключ на R2: productID ФЗ 5-17

Ф3 5-13 на R2

R3(productID, productName, productManufacture, productPrice, productModel, productType, productSize, quantityProduct)

R4(orderID, orderPrice, orderQuantity, useIDeRef, productIDRef)

Ключ на R3: productID ФЗ 5-13

Ключ на R4: orderID ФЗ 13-17

Результати декомпозиції:

* User (userID, userName, userPassword, userType) – R1
* Storage(productID, productName, productManufacture, productPrice, productModel, productType, productSize, quantityProduct) – R3
* Order(orderID, orderPrice, orderQuantity, userIDRef, productIDRef) – R5

## Висновок до розділу 2

У другому розділі нашої курсової роботи ми провели концептуальне та логічне проектування реляційної бази даних, зробили опис предметної області та прецедентів використання, розробили ІС та створили бізнес-процеси нашої бази даних

# **РОЗДІЛ 3**

# **ПРАКТИЧНА ЧАСТИНА**

## 3.1. Фізична модель

### 3.1.1. Оператори створення таблиць

Проаналізувавши декомпозицію, створимо таблиці з урахуванням загальних обмежень та приватних ключів завдяки операторам створення таблиць мови програмування SQL.

Створення таблиці Users:

CREATE TABLE Users

(

userID varchar(25) not null primary key,

userName varchar(25) not null,

userPassword varchar(25) not null,

userType binary not null

)

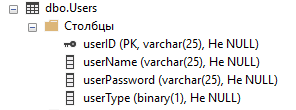


Рис. 3.1 – Таблиця Users

Створення таблиці Orders:

CREATE TABLE Orders

(

orderID varchar(25) primary key not null,

orderPrice float not null,

orderQuantit int not null,

userIDRef varchar(25) not null,

productIDRef varchar (25) not null,

foreign key (userIDRef) references Users(userID),

foreign key (productIDRef) references Storage(productID)

)

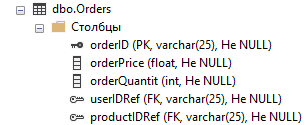


Рис. 3.2 – Таблиця Orders

Створення таблиці Storage:

CREATE TABLE Storage

(

productID varchar(25) primary key not null,

productName varchar(25) not null,

productManufacture varchar(25) not null,

productPrice float not null,

productModel varchar(25) not null,

productType varchar(25)not null,

productSize float not null,

quantityProduct int not null

)

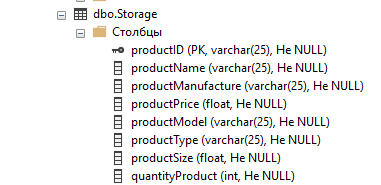


Рис. 3.3 – Таблиця Storage

Дані для заповення таблиць описані в додатку А

Табл. 3.1

Представлення характеристики сутності Users:

|  |  |  |
| --- | --- | --- |
| Атрибути | Тип даних | Характеристики атрибутів |
| userID | varchar | primary key, not null, обмеження у 25 символів |
| userName | varchar | not null, обмеження у 25 символів |
| userPassword | varchar | not null, обмеження у 25 символів |
| userType | binary | Може приймати тільки значення true або false |

Табл. 3.2

Представлення характеристики сутності Orders:

|  |  |  |
| --- | --- | --- |
| Атрибути | Тип даних | Характеристики атрибутів |
| orderID | varchar | primary key, not null, обмеження у 25 символів |
| orderPrice | float | not null |
| orderQuantit | Int | not null |
| productIDRes | varchar | foreign key, обмеження у 25 символів, not null |
| userIDRes | varchar | foreign key, обмеження у 25 символів, not null |

Табл. 3.3

Представлення характеристики сутності Storage:

|  |  |  |
| --- | --- | --- |
| Атрибути | Тип даних | Характеристики атрибутів |
| productID | varchar | Primary key, not null, обмеження у 25 символів |
| productName | varchar | not null, обмеження у 25 символів |
| productManufacture | varchar | not null, обмеження у 25 символів |
| productPrice | float | not null |
| productModel | varchar | not null, обмеження у 25 символів |
| productType | varchar | not null, обмеження у 25 символів |
| productSize | float | not null |
| quantityProduct | int | not null |

\

### 3.1.2. Створення коментарів

Приклади команд для створення коментарів до таблиць та до стовпців:

exec sys.sp\_addextendedproperty

@name = N'MS\_Description',

@value = N'Users table, columns: userID - primary key, userName, userPassword, userType',

@level0type = N'SCHEMA', @level0name = N'dbo',

@level1type = N'TABLE', @level1name = N'Users'

exec sys.sp\_addextendedproperty

@name = N'MS\_Description',

@value = N'Varchar userName can contain 25 symbols',

@level0type = N'SCHEMA', @level0name = N'dbo',

@level1type = N'TABLE', @level1name = N'Users',

@level2type = N'COLUMN', @level2name = N'userName'

exec sys.sp\_addextendedproperty

@name = N'MS\_Description',

@value = N'Varchar userPassword can contain 25 symbols',

@level0type = N'SCHEMA', @level0name = N'dbo',

@level1type = N'TABLE', @level1name = N'Users',

@level2type = N'COLUMN', @level2name = N'userPassword'

exec sys.sp\_addextendedproperty

@name = N'MS\_Description',

@value = N'Varchar userType can contain 0 or 1',

@level0type = N'SCHEMA', @level0name = N'dbo',

@level1type = N'TABLE', @level1name = N'Users',

@level2type = N'COLUMN', @level2name = N'userType'

exec sys.sp\_addextendedproperty

@name = N'MS\_Description',

@value = N'Varchar userID can contain 25 symbols',

@level0type = N'SCHEMA', @level0name = N'dbo',

@level1type = N'TABLE', @level1name = N'Users',

@level2type = N'COLUMN', @level2name = N'userID'

### 3.1.3 Опис обмежень

* ID користувача не може перевищувати 25 символів, не може дорівнювати NULL, первинний ключ
* Ім’я користувача не може перевищувати 25 символів, не може дорівнювати NULL
* Пароль користувача не може перевищувати 25 символів, не може дорівнювати NULL
* Тип користувача може мати значення 1 або 0
* ID заказу не може перевищувати 64 символів, не може дорівнювати NULL, первинний ключ
* Ціна заказу не може бути менше або дорівнювати 0
* Кількість заказу не може бути менше або дорівнювати 0
* ID побутової техніки не може користувача не може перевищувати 25 символів, дорівнювати NULL, первинний ключ
* Номер побутової техніки користувача не може перевищувати 25 символів, не може дорівнювати NULL
* Виробник побутової техніки не може користувача не може перевищувати 25 символів, дорівнювати NULL
* Ціна за побутову техніку не може бути менше або дорівнювати 0
* Модель побутової техніки не може дорівнювати NULL
* Тип побутової техніки не може користувача не може перевищувати 25 символів, дорівнювати NULL
* Кількість побутової техніки не може бути менше або дорівнювати 0

Навести опис обмежень у вигляді generalassertions мовою SQL:

create assertion assert\_OrderPrice(

0 < any (select orderPrice from Order)

);

create assertion assert\_ OrderQuantit (

0 < any (select orderQuantit from Order)

);

create assertion assert\_ProductPrice (

0 < any (select productPrice from Product)

);

create assertion assert\_ProductQuantit (

0 < any (select quantityProduct from Product)

);

create assertion assert\_ProductSize (

0 < any (select productSize from Product)

);

Створення таблиць з обмеженнями, які реалізовані засобами constraints:

Створення таблиці Orders:

CREATE TABLE Orders

(

orderID varchar(25) primary key not null,

orderPrice float not null check(orderPrice > 0),

orderQuantit int not null check(orderQuantit > 0),

userIDRef varchar(25) not null,

productIDRef varchar (25) not null,

foreign key (userIDRef) references Users(userID),

foreign key (productIDRef) references Storage(productID)

)

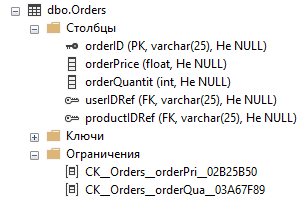


Рис. 3.4 – Таблиця Orders засобами constraints

Створення таблиці Storage:

CREATE TABLE Storage

(

productID varchar(25) primary key not null,

productName varchar(25) not null,

productManufacture varchar(25) not null,

productPrice float not null check(productPrice > 0),

productModel varchar(25) not null,

productType varchar(25)not null,

productSize float not null check(productSize > 0),

quantityProduct int not null check(quantityProduct >= 0)

)

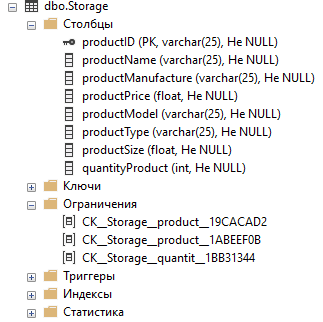


Рис. 3.5 – Таблиця Storage засобами constraints

## 3.2. Представлення

Розробимо представлення для таблиць нашої бази даних, які будуть показувати тількі певні обрані за логікою дані з таблиць. На рисунках 3.6-3.13 можно побачити результати утворення представлень.

Створення представлення, що зображує всіх користувачів з типом доступу звичайних клієнтів:

create view Standard\_users (userName,userPassword,userType)

as select userName,userPassword,userType

from Users

where userType = 0

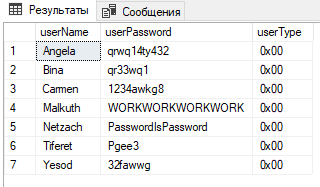


Рис. 3.6 – Представлення Standard\_users

Створення представлення, що зображує всіх користувачів з типом доступу адмінистрації компанії:

create view Admin\_users (userName,userPassword,userType)

as select userName,userPassword,userType

from Users

where userType = 1;

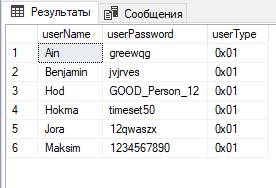


Рис. 3.7 – Представлення Admin\_users

Створення представлення, що зображує вибіркові дані заказів та користувачів:

create view Users\_Order (userName,userPassword , userType, orderPrice, orderQuantit)

as select Users.userName ,Users.userPassword, Users.userType, Orders.orderPrice, Orders.orderQuantit from Users

left outer join Orders on Users.userName = Orders.orderID

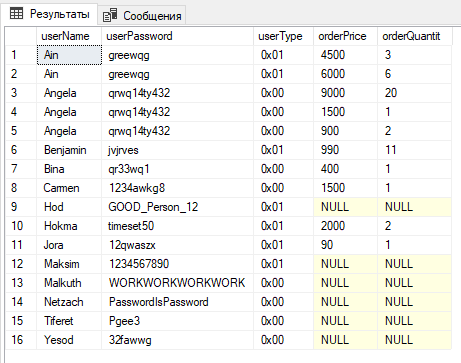


Рис. 3.8 – Представлення Users\_Order

Створення представлення, що зображує побутову техніку, якої на складі залишилось дуже мало:

create view Low\_amount\_of\_product as

select productName,productManufacture,productModel,quantityProduct from Storage

where quantityProduct <= 5

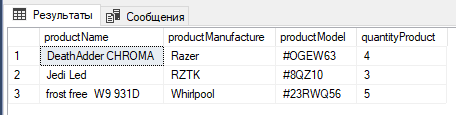


Рис. 3.9 – Представлення Low\_amount\_of\_product

Створення представлення, що зображує вибіркові дані артивних закалів:

create view Active\_order as select \* from Users

join Orders on Users.userID = Orders.userIDRef

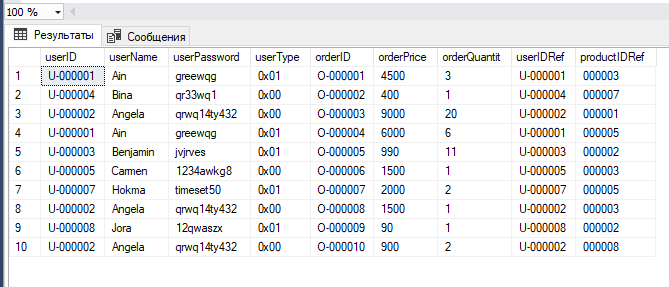


Рис. 3.10 – Представлення Active\_order

Створення представлення, що зображує всі заказів звичайних клієнтів

create view Standard\_orders (userName,userPassword , userType, orderPrice, orderQuantit)

as select Users.userName ,Users.userPassword, Users.userType, Orders.orderPrice, Orders.orderQuantit from Users

left outer join Orders on Users.userID = Orders.userIDRef

where userType = 0

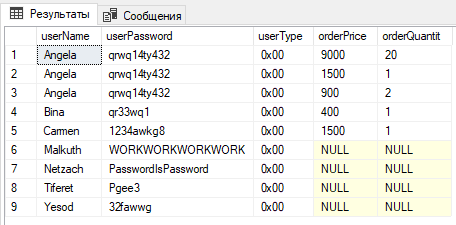


Рис. 3.11 – Представлення Standard\_orders

Створення представлення, що зображує всі заказів з типом доступу адмінистрації компанії:

create view Admin\_orders (userName,userPassword , userType, orderPrice, orderQuantit)

as select Users.userName ,Users.userPassword, Users.userType, Orders.orderPrice, Orders.orderQuantit from Users

left outer join Orders on Users.userID = Orders.userIDRef

where userType = 1

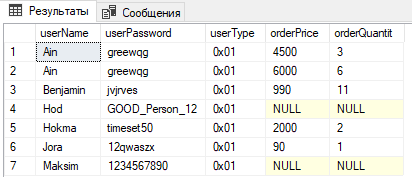


Рис. 3.12 – Представлення Admin\_orders

## 3.3. Тригери

Створюємо таблицю для збереження виконаних заказів (Додаток В):

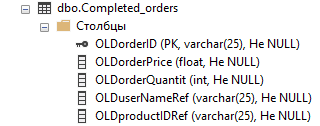


Рис. 3.13 – Таблиця для збереження інформації про виконані закази

Тригер AFTER для оператору DELETE для збереження даних про виконані закази:

create trigger Clear\_order on Orders

after delete

as begin

insert into Completed\_orders(OLDorderID ,OLDorderPrice ,OLDorderQuantit ,OLDuserIDRef ,OLDproductIDRef )

select orderID, orderPrice, orderQuantit, userIDRef, productIDRef from deleted

end

Результат:

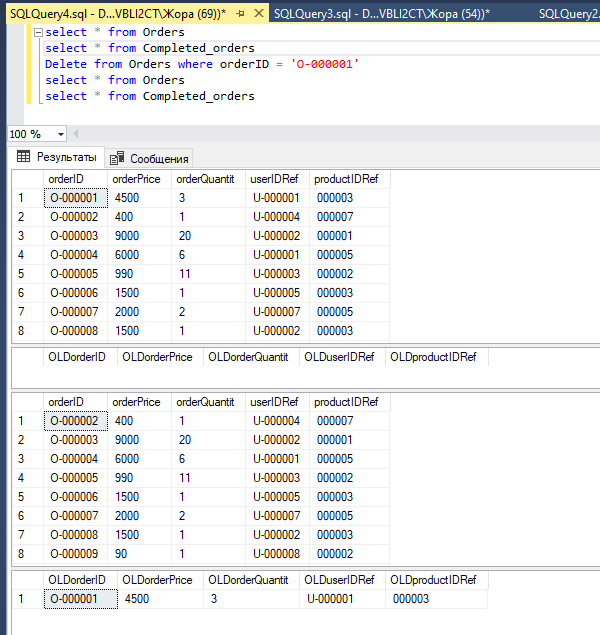


Рис. 3.14 – Запит для тригера Clear\_order

Створюємо таблицю для збереження інформації про поставлення нової побутової техніки на склад (Додаток В):

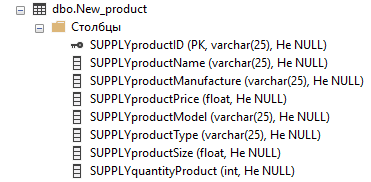


Рис. 3.15 – Таблиця для збереження інформації про поставлення нової побутової техніки на склад

Тригер AFTER для оператору INSERT, щоб зберігати накладні про поставлення нової побутової техніки на склад:

create trigger Product\_delivery on Storage

after insert

as

insert into New\_product(SUPPLYproductID, SUPPLYproductName, SUPPLYproductManufacture, SUPPLYproductPrice, SUPPLYproductModel, SUPPLYproductType, SUPPLYproductSize,SUPPLYquantityProduct )

select productID,productName ,productManufacture , productPrice,productModel , productType, productSize, quantityProduct

from inserted

Результат:

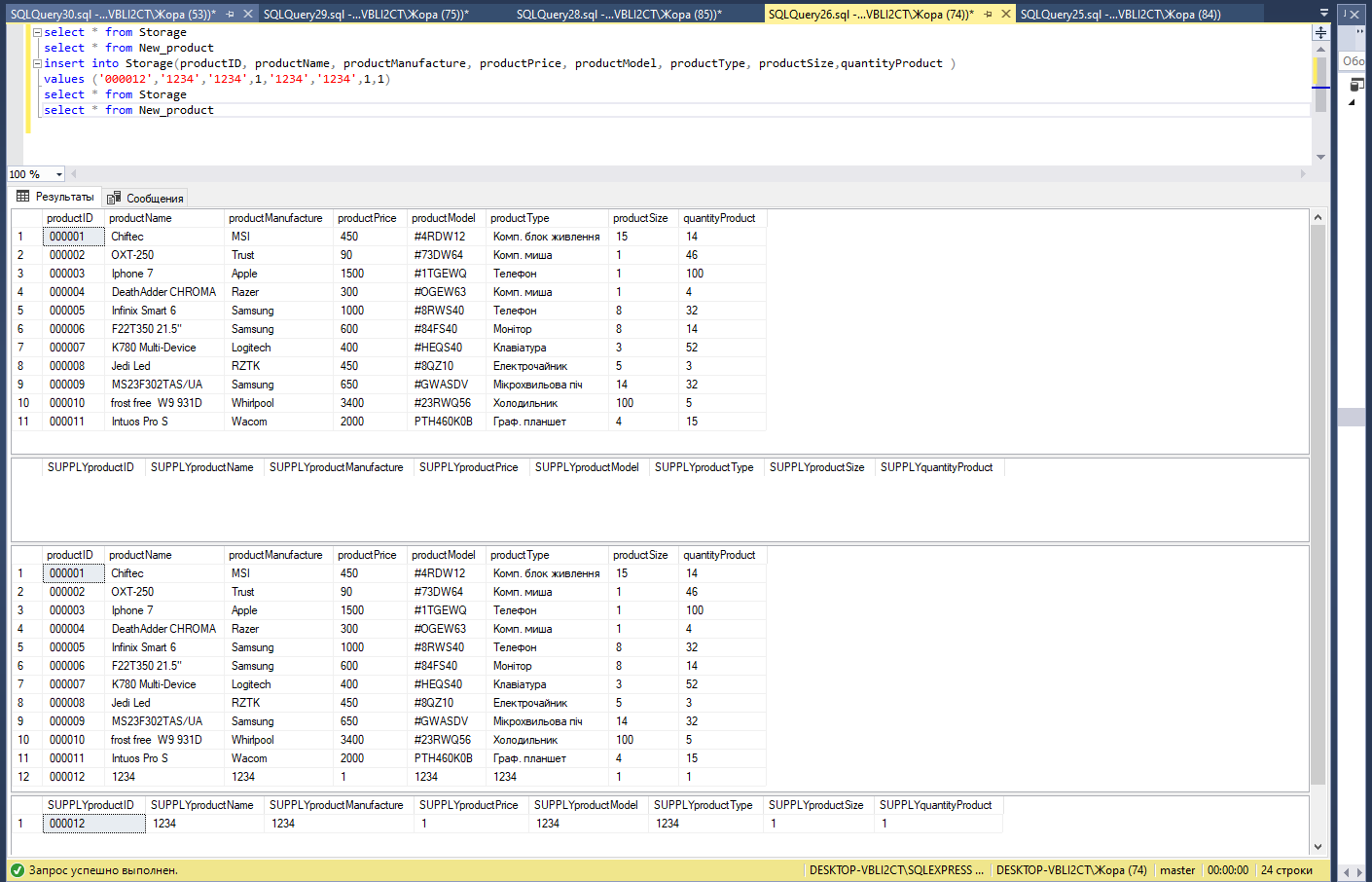


Рис. 3.16 - Запит для тригера Product\_delivery

Створюємо таблицю для збереження інформації про зміни кількості побутової техніки на складі (Додаток В):

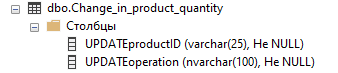


Рис. 3.17 – Таблиця для збереження інформації про зміни кількості побутової техніки на складі

Тригер AFTER для оператору UPDATE, щоб зберігати накладні про змінення кількості побутової техніки на склад:

create trigger Update\_product

on Storage

after update

as

insert into Change\_in\_product\_quantity(UPDATEproductID,UPDATEproductName,UPDATEoperation)

select productID, productName,quantityProduct

from inserted

Результат:

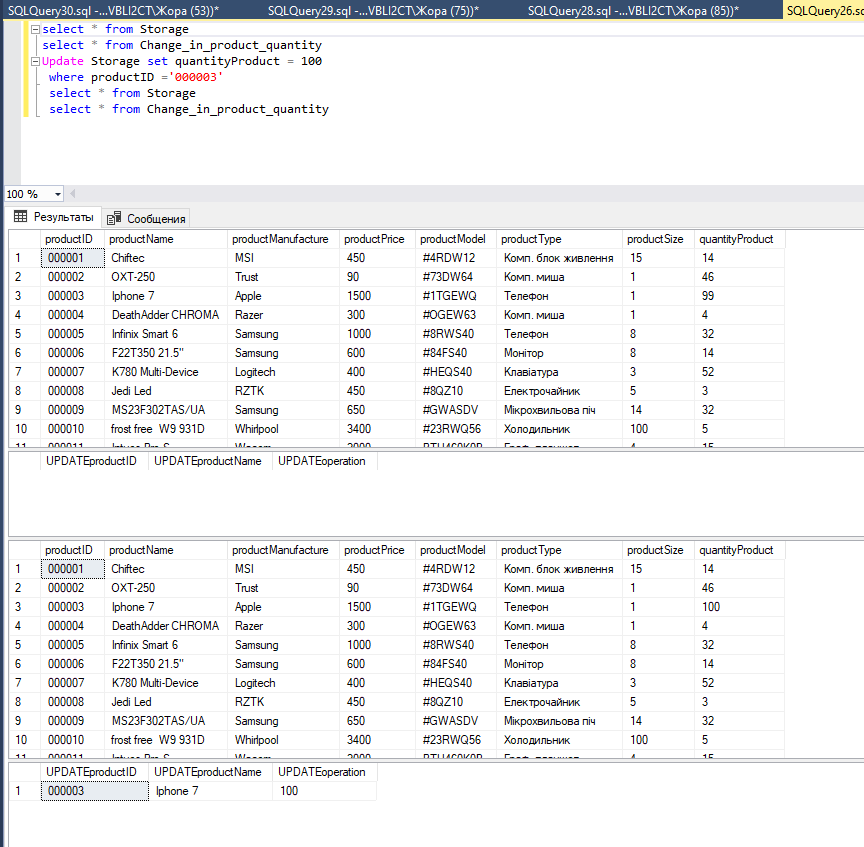


Рис. 3.18 – Запит для тригера Update\_product

Тригер обмеження, для кількості, ваги та ціни техніки

create trigger Cheak\_limits\_Product on Storage

instead of insert

as begin

if (select productPrice from inserted) < 0

raiserror('Product price must be more then 0', 16, 1)

else if(select quantityProduct from inserted) < 0

raiserror('Product quantity must be more then 0', 16, 1)

else if(select productSize from inserted) < 0

raiserror('Product weight must be more then 0', 16, 1)

else

begin

insert into Storage(productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct)

select productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct from inserted

end

end

Результат:

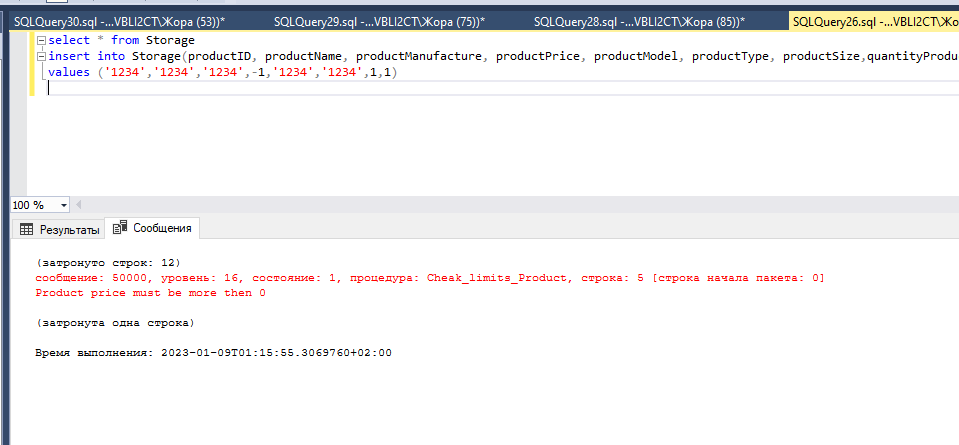


Рис. 3.19 – Запит для тригера Cheak\_limits\_ Product, випадок 1

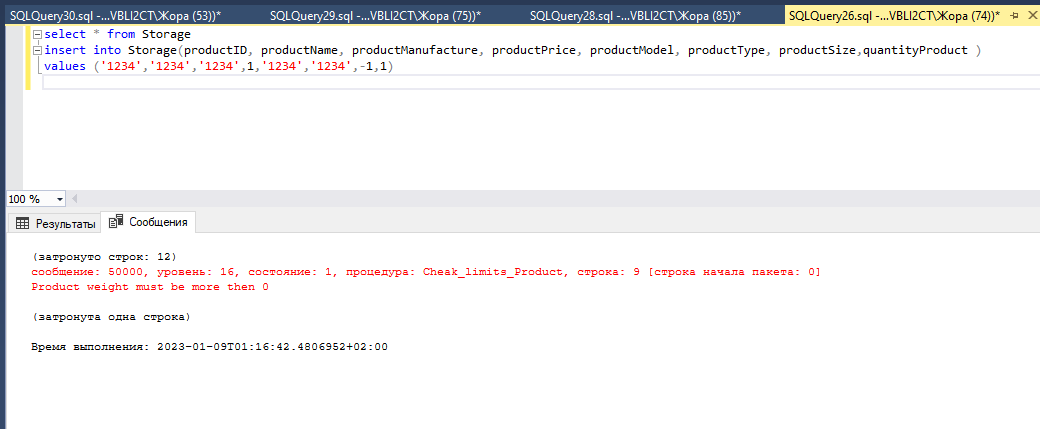


Рис. 3.20 – Запит для тригера Cheak\_limits\_ Product, випадок 2

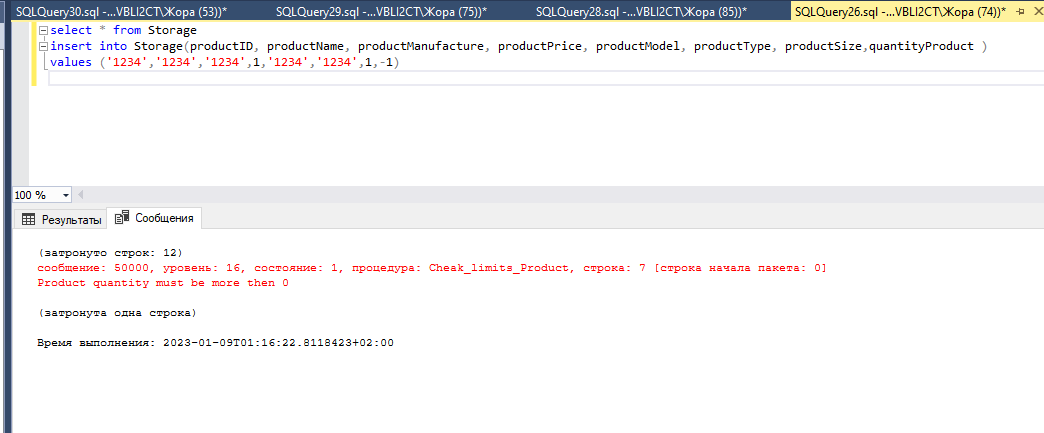


Рис. 3.21 – Запит для тригера Cheak\_limits\_Product, випадок 3

Тригер обмеження, для кількості та ціни заказу

create trigger Cheak\_limits\_Orders on Orders

instead of insert

as begin

if (select orderPrice from inserted) < 0

raiserror('Order price must be more then 0', 16, 1)

else if(select orderQuantit from inserted) < 0

raiserror('Order quantity must be more then 0', 16, 1)

else

begin

insert into Orders(userIDRef,productIDRef,orderID, orderPrice, orderQuantit)

select userIDRef,productIDRef,orderID, orderPrice, orderQuantit from inserted

end

end

Результат:



Рис. 3.22 – Запит для тригера Cheak\_limits\_Orders, випадок 1

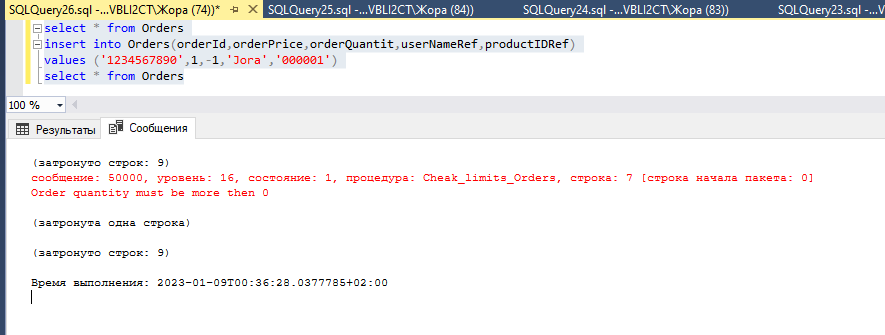


Рис. 3.23 – Запит для тригера Cheak\_limits\_Orders, випадок 2

Тригер обмеження, для величини паролю користувачів:

create trigger Cheak\_limits\_User on Users

instead of insert

as begin

if (select len(userPassword) from inserted) < 5

raiserror('Password is too short (minimum 5 symbols)', 16, 1)

else

begin

insert into Users(userName,userPassword,userType,userID)

select userName,userPassword,userType,userID from inserted

end

end

Результат:

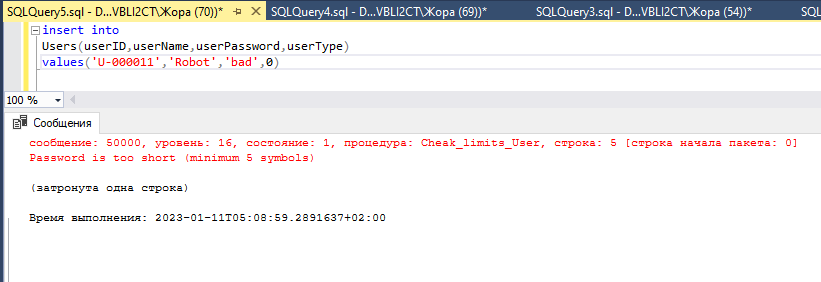


Рис. 3.24 – Запит для тригера Cheak\_limits\_User

Реалізації запитів до тригерів описані у додатку Б

## 3.4. Індекси

### 3.4.1. Плани виконання запитів без використання індексів

Запити коритувача:

1. Запит на сортування побутової техніки на складі, котра коштує менше за 1000:

select productName as Tech\_name, productManufacture as Manufacture, productPrice as Price, quantityProduct as Quatity from Storage

group by productPrice, productName,productManufacture , quantityProduct

Having productPrice < 1000

Результат:

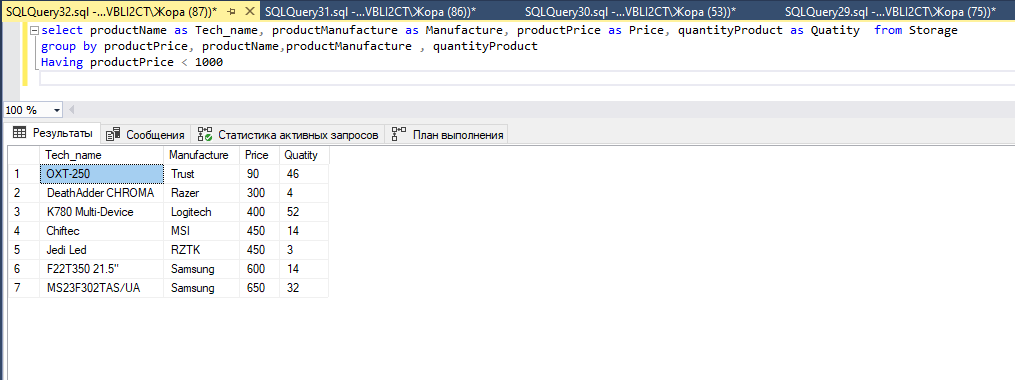


Рис. 3.25 – Результати виконання першого запиту користувача без використання індексів

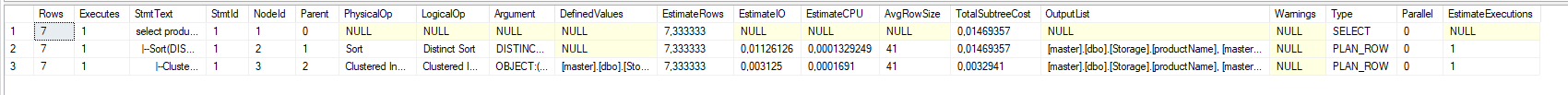


Рис. 3.26 - План виконання першого запиту користувача без використання індексів

1. Запит на сортування заказів обутової техніки за їхнім ID номером:

select orderID as ID,userIDRef as Client, orderPrice as Price, orderQuantit as Quantity, productIDRef as ID\_of\_technics

from Orders

order by orderID, userIDRef, productIDRef,orderPrice,orderQuantit

Результат:

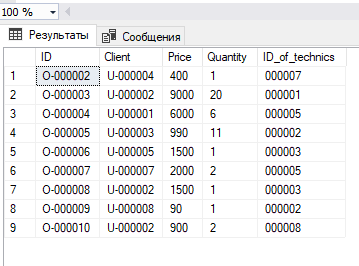


Рис. 3.27 –– Результати виконання другого запиту користувача без використання індексів

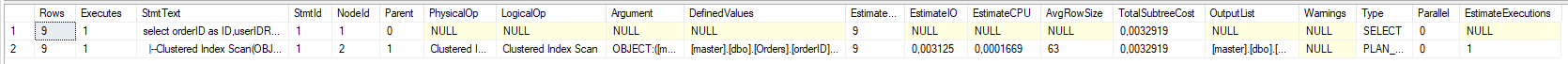


Рис. 3.28 – План виконання другого запиту користувача без використання індексів

1. Запит на вивід приналежність користувачів до певного типу статусу (компанії,клієнт)

select userName as Nickname, userType as Type\_of\_use, 'Company' as Belonging from Users

where userType = 1

union

select userName as Nickname, userType as Type\_of\_use, 'Client' as Belonging from Users

where userType = 0

Результат:

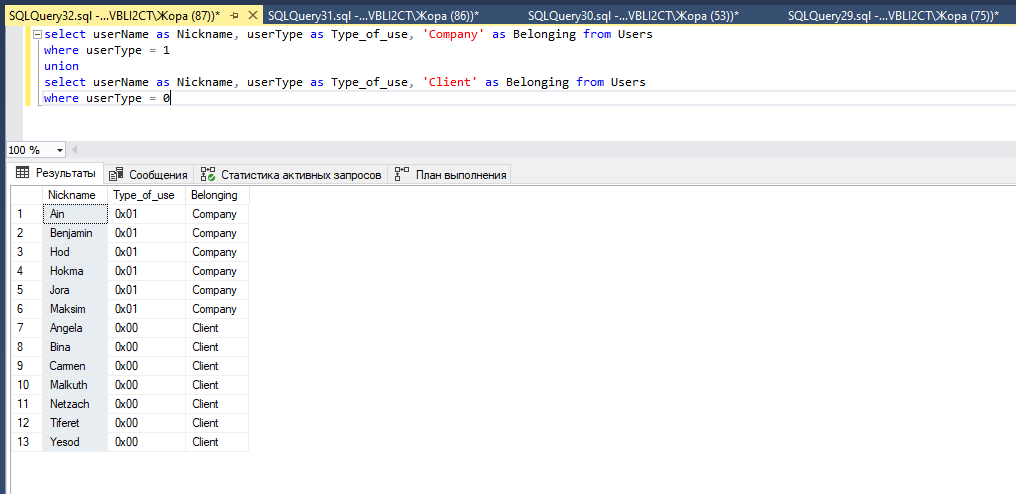
****

Рис. 3.29 – Результати виконання третього запиту користувача без використання індексів

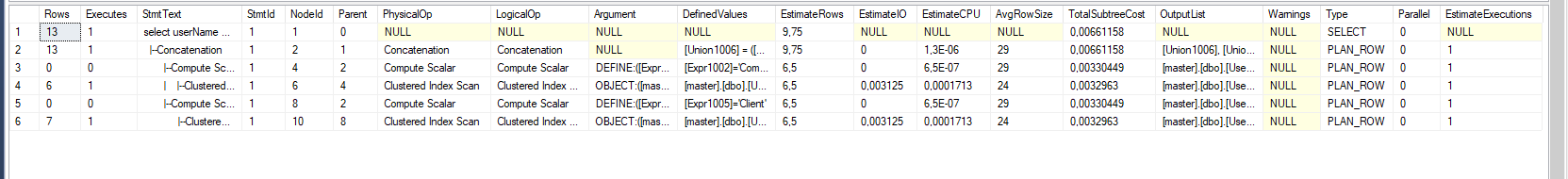
****

Рис. 3.30 – План виконання третього запиту користувача без використання індексів

Запити представлень:

1. Запит представлення, що зображує вибіркові дані активних закалів:

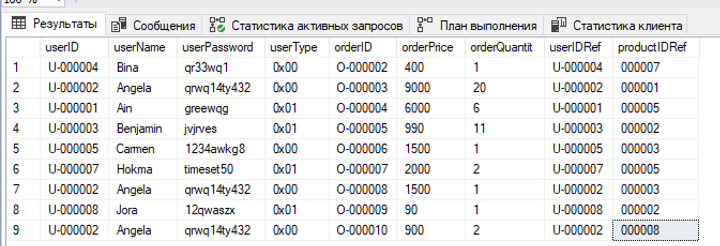


Рис. 3.31 – Результати виконання першого запиту представлення без використання індексів

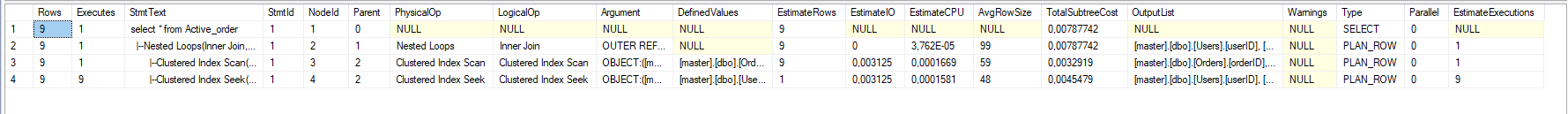


Рис. 3.32 – План виконання першого запиту представлення без використання індексів

1. Запит представлення, що зображує побутову техніку, якої на складі залишилось дуже мало:

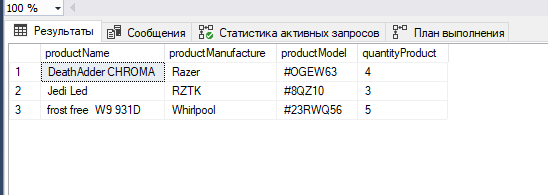


Рис. 3.33 – Результати виконання першого запиту представлення без використання індексів

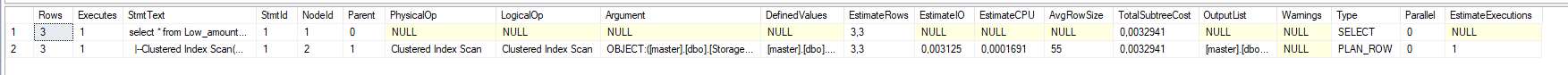


Рис. 3.34 – План виконання першого запиту представлення без використання індексів

### 3.4.2. Плани виконання запитів з використанням індексів

Створині індекси для наших запитів:

Для запитів користувача:

create index Index\_StoragePrice on Storage(productPrice)

create index Index\_OrdersID on Orders(orderID)

create index Index\_UserType on Users(userType)

Для запитів представлень:

create index Index\_OrderActive on Users(userID)

create index Index\_StorageLowQ on Storage(quantityProduct)

Запити коритувача:

1. Запит на сортування побутової техніки на складі, котра коштує менше за 1000:

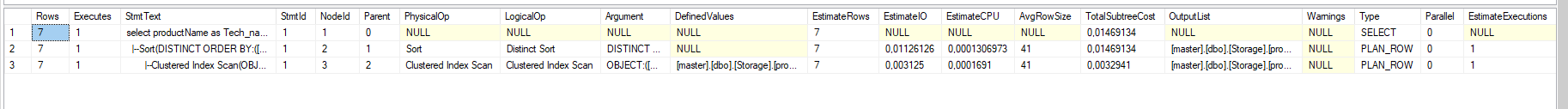


Рис. 3.35 – План виконання першого запиту користувача без використання індексів

1. Запит представлення , що зображує побутову техніку, якої на складі залишилось дуже мало:

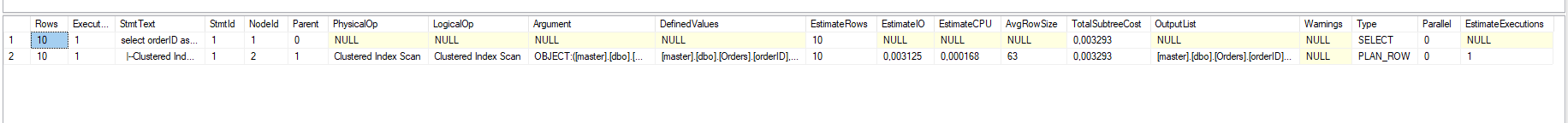


Рис. 3.36 – План виконання другого запиту користувача без використання індексів

1. Запит на вивід приналежність користувачів до певного типу статусу (компанії,клієнт)

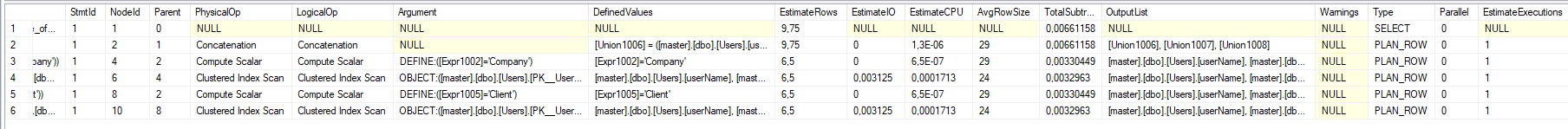


Рис. 3.37 – План виконання третього запиту користувача без використання індексів

Запити представлень:

1. Запит представлення, що зображує вибіркові дані активних закалів:

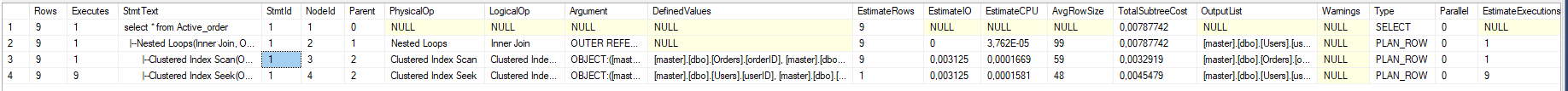


Рис. 3.38 – План виконання першого запиту представлень з використання індексів

1. Запит представлення , що зображує побутову техніку, якої на складі залишилось дуже мало:

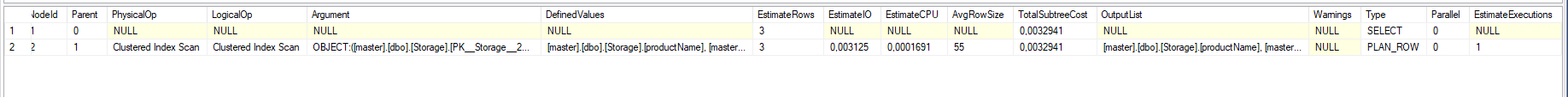


Рис. 3.39 – План виконання другого запиту користувача з використання індексів

Розглянувши результати запитів з індексами та без них можна зробити висновок, що індексація прискорює виконання запистів, але їх треба використовувати під час роботи з великими обсягами даних тому, як в деяких випадках індексація не пришвидвувала роботу.

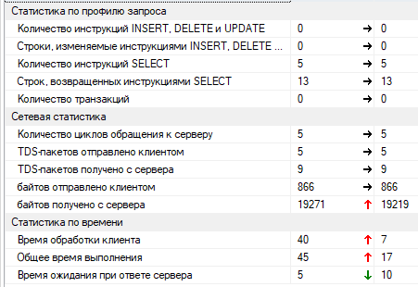


Рис. 3.40 – Статистика виконання без та з індексом для першого запиту користувача

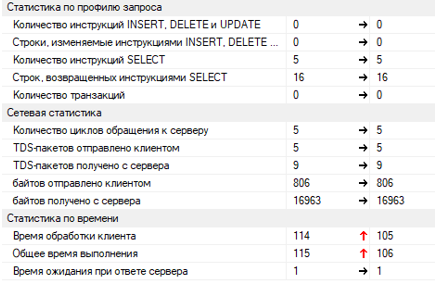


Рис. 3.41 – Статистика виконання без та з індексом для другого запиту користувача

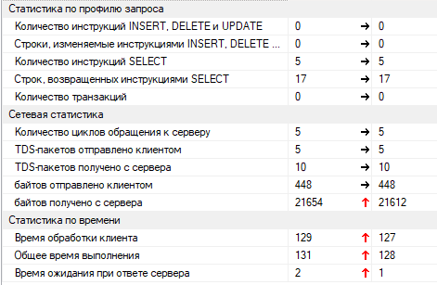


Рис. 3.42 – Статистика виконання без та з індексом для третьего запиту

користувача

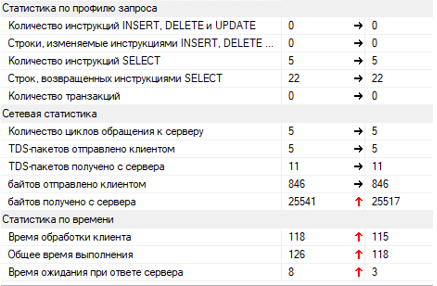


Рис. 3.43 – Статистика виконання без та з індексом для першого запиту представлень

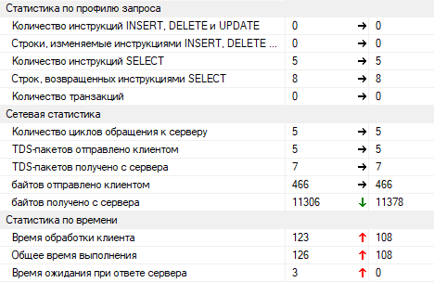


Рис. 3.44 – Статистика виконання без та з індексом для другого запиту представлень

## Висновок до розділу 3

У практичній частині ми для забезпечення узгодженості та цілісності у відповідності із сформульованими обмеженнями та бізнес-правилами предметної області розроблено 18 обмежень цілісності, з них: 13 — унікальності (3 — первинного ключа), 2 — зовнішнього ключа; 7 представлень (3 — матеріалізованих), 6 тригерів та 5 індексів.

# **ВИСНОВОК**

Була поставлена задача реалізувати базу даних для обліку побутової техніки на складі. Використовуючи програмне забезпечення легко можна робити облік товарів на складі, регулювати запити та вести накладні.

В ході роботи було звернуто увагу на просту доступність до певнох бази даних , сортування за різними критеріями , визначення обмеження предметної області, розробці бізнес-правила, розробці логічної модель інформаційної системи. Для реалізування нашої бази даних була обрана система управління базами даних Microsoft SQL Server з розробленням таблиць, представлень, перевірок, обмеженням цілісності, індексів та тригерів.

На прикладі курсової роботи була виконана основна ціль: навчитися використовувати різні підходи систиматизації даних та закріпити отримані за курс навчальної дисципліни “Система бази даних” знання на практицы.

# **СПИСОК ВИКОРИЧТАНИХ ДЖЕРЕЛ**

1. Основні відомості про бази даних [Електронний ресурс] – Режим доступу до ресурсу: <https://support.microsoft.com/uk-ua/office/основні-відомості-про-бази-даних-a849ac16-07c7-4a31-9948-3c8c94a7c204>
2. Введение в Microsoft SQL Server [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Microsoft_SQL_Server>
3. Етапи проектування БД. [Електронний ресурс] – Режим доступу до ресурсу: <http://um.co.ua/8/8-2/8-228774.html>
4. Реляційна модель даних. Поняття реляційної БД [Електронний ресурс] – Режим доступу до ресурсу: <https://stud.com.ua/43428/informatika/relyatsiyna_model_danih_ponyattya_relyatsiynoyi>
5. Основні поняття реляційних БД: нормалізація, зв’язок та ключі [Електронний ресурс] – Режим доступу до ресурсу: <https://bondarenko.dn.ua/osnovni-ponyattya-relyatsijnih-bd-normalizatsiya-zv-yazok-ta-klyuchi/>
6. [Електронний ресурс] – Режим доступу до ресурсу:
7. Триггеры для операций INSERT, UPDATE, DELETE [Електронний ресурс] – Режим доступу до ресурсу: <https://metanit.com/sql/sqlserver/12.2.php>.
8. Предикати SQL [Електронний ресурс] / Режим доступу: <https://docs.microsoft.com/en-us/sql/t-sql/queries/predicates>
9. Представление Базы Данных [Електронний ресурс] – Режим доступу до ресурсу: <https://anylogic.help/ru/anylogic/connectivity/view.html>
10. Нормалізація баз даних [Електронний ресурс] – Режим доступу до ресурсу: <https://www.wikiwand.com/uk/Нормалізація_баз_даних>
11. Microsoft SQL Server [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/ru-ru/sql/tools/overview-sql-tools?view=sql-server-ver16>
12. [Електронний ресурс] – Режим доступу до ресурсу: <https://studfile.net/preview/972381/page:5/>
13. Діаграма сутність-зв’язок [Електронний ресурс] – Режим доступу до ресурсу: <https://studfile.net/preview/5462334/page:12/>
14. Основные сведения о базах данных [Електронний ресурс] – Режим доступу до ресурсу: <https://support.microsoft.com/ru-ru/office/основные-сведения-о-базах-данных-a849ac16-07c7-4a31-9948-3c8c94a7c204>
15. Оптимізація роботи БД - Реляційні бази даних [Електронний ресурс] – Режим доступу до ресурсу: <https://rdb.dp.ua/uk/chapter_10>
16. Руководство по SQL. Индексы. [Електронний ресурс] – Режим доступу до ресурсу: <https://proselyte.net/tutorials/sql/sql-indexes/>

# **ДОДАТОК A**

Users:

insert into

Users(userID,userName,userPassword,userType)

values ('U-000001','Ain', 'greewqg',1)

insert into

Users(userID,userName,userPassword,userType)

values ('U-000002','Angela', 'qrwq14ty432',0)

insert into

Users(userID,userName,userPassword,userType)

values ('U-000003','Benjamin', 'jvjrves',1)

insert into

Users(userID,userName,userPassword,userType)

values ('U-000004','Bina', 'qr33wq1',0)

insert into

Users(userID,userName,userPassword,userType)

values ('U-000005','Carmen', '1234awkg8',0)

insert into

Users(userID,userName,userPassword,userType)

values ('U-000006','Hod', 'GOOD\_Person\_12',1)

insert into

Users(userID,userName,userPassword,userType)

values ('U-000007','Hokma', 'timeset50',1)

insert into

Users(userID,userName,userPassword,userType)

values ('U-000008','Jora', '12qwaszx',1)

insert into

Users(userID,userName,userPassword,userType)

values ('U-000009','Maksim', '1234567890',1)

insert into

Users(userID,userName,userPassword,userType)

values ('U-000010','Malkuth', 'WORKWORKWORKWORK',0)

insert into

Users(userID,userName,userPassword,userType)

values ('U-000011','Netzach', 'PasswordIsPassword',0)

insert into

Users(userID,userName,userPassword,userType)

values ('U-000012','Tiferet', 'Pgee3',0)

insert into

Users(userID,userName,userPassword,userType)

values ('U-000013','Yesod', '32fawwg',0)

Orders:

select \* from Users

insert into

Orders(userIDRef,productIDRef,orderID, orderPrice, orderQuantit)

values ('U-000001','000003','O-000001', 4500, 3)

insert into

Orders(userIDRef,productIDRef,orderID, orderPrice, orderQuantit)

values ('U-000002','000001','O-000003', 9000, 20)

insert into

Orders(userIDRef,productIDRef,orderID, orderPrice, orderQuantit)

values ('U-000001','000005','O-000004', 6000, 6)

insert into

Orders(userIDRef,productIDRef,orderID, orderPrice, orderQuantit)

values ('U-000005','000003','O-000006', 1500,1)

insert into

Orders(userIDRef,productIDRef,orderID, orderPrice, orderQuantit)

values ('U-000003','000002','O-000005', 990, 11)

insert into

Orders(userIDRef,productIDRef,orderID, orderPrice, orderQuantit)

values ('U-000004','000007','O-000002', 400, 1)

insert into

Orders(userIDRef,productIDRef,orderID, orderPrice, orderQuantit)

values ('U-000007','000005','O-000007', 2000, 2)

insert into

Orders(userIDRef,productIDRef,orderID, orderPrice, orderQuantit)

values ('U-000002','000003','O-000008', 1500, 1)

insert into

Orders(userIDRef,productIDRef,orderID, orderPrice, orderQuantit)

values ('U-000008','000002','O-000009', 90, 1)

insert into

Orders(userIDRef,productIDRef,orderID, orderPrice, orderQuantit)

values ('U-000002','000008','O-000010', 900, 2)

Storage:

insert into Storage(productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct)

values ('000001','MSI','#4RDW12','Chiftec', 450 ,15,'Комп. блок живлення', 14 )

insert into Storage(productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct)

values ('000002','Trust','#73DW64','OXT-250', 90 ,1,'Комп. миша', 46 )

insert into Storage(productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct)

values ('000003','Apple','#1TGEWQ','Iphone 7', 1500 ,1,'Телефон', 100 )

insert into Storage(productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct)

values ('000004','Razer','#OGEW63','DeathAdder CHROMA', 300 ,1,'Комп. миша', 4 )

insert into Storage(productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct)

values ('000005','Samsung','#8RWS40','Infinix Smart 6', 1000 ,8 ,'Телефон', 32 )

insert into Storage(productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct)

values ('000006','Samsung','#84FS40','F22T350 21.5"', 600 ,8 ,'Монітор', 14 )

insert into Storage(productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct)

values ('000007','Logitech','#HEQS40','K780 Multi-Device', 400 ,3 ,'Клавіатура', 52 )

insert into Storage(productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct)

values ('000008','RZTK','#8QZ10','Jedi Led', 450 , 5 ,'Електрочайник', 3 )

insert into Storage(productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct)

values ('000009','Samsung','#GWASDV','MS23F302TAS/UA', 650 ,14 ,'Мікрохвильова піч', 32 )

insert into Storage(productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct)

values ('000010','Whirlpool','#23RWQ56','frost free W9 931D', 3400 ,100 ,'Холодильник', 5 )

insert into

Storage(productID,productManufacture,productModel,productName,productPrice,productSize,productType,quantityProduct)

values ('000011','Wacom','PTH460K0B','Intuos Pro S', 2000 , 4 ,'Граф. планшет', 15 )

# **ДОДАТОК Б**

Запит до тригера Clear\_order:

Select \* from Orders

Select \* from Сompleted\_orders

Delete from Orders where orderID = 'O-000001'

Select \* from Orders

Select \* from Сompleted\_orders

Запит до тригера Product\_delivery:

select \* from Storage

select \* from New\_product

insert into Storage(productID, productName, productManufacture, productPrice, productModel, productType, productSize,quantityProduct )

values ('000012','1234','1234',1,'1234','1234',1,1)

select \* from Storage

select \* from New\_product

Запит до тригера Update\_product:

select \* from Storage

select \* from Change\_in\_product\_quantity

Update Storage set quantityProduct = 100

where productID ='000003'

select \* from Storage

select \* from Change\_in\_product\_quantity

Запит до тригера Cheak\_limits\_Users:

insert into

Users(userID,userName,userPassword,userType)

values('U-000011','Robot','bad',0)

Запит до тригера Cheak\_limits\_Orders:

select \* from Orders

insert into Orders(orderId,orderPrice,orderQuantit,userIDRef,productIDRef)

values ('1234567890',1,-1,'Jora','000001')

select \* from Orders

select \* from Orders

insert into Orders(orderId,orderPrice,orderQuantit,userIDRef,productIDRef)

values ('1234567890',-1,1,'Jora','000001')

select \* from Orders

Запит до тригера Cheak\_limits\_Product:

select \* from Storage

insert into Storage(productID, productName, productManufacture, productPrice, productModel, productType, productSize,quantityProduct )

values ('1234','1234','1234',1,'1234','1234',-1,1)

select \* from Storage

insert into Storage(productID, productName, productManufacture, productPrice, productModel, productType, productSize,quantityProduct )

values ('1234','1234','1234',1,'1234','1234',1,-1)

select \* from Storage

insert into Storage(productID, productName, productManufacture, productPrice, productModel, productType, productSize,quantityProduct )

values ('1234','1234','1234',-1,'1234','1234',1,1)

# **ДОДАТОК В**

Таблиця для збереження інформації про поставлення нової побутової техніки на склад:

CREATE TABLE New\_product

(

SUPPLYproductID varchar(25) not null,

SUPPLYproductName varchar(25) not null,

SUPPLYproductManufacture varchar(25) not null,

SUPPLYproductPrice float not null check(SUPPLYproductPrice >= 0),

SUPPLYproductModel varchar(25) not null,

SUPPLYproductType varchar(25)not null,

SUPPLYproductSize float not null check(SUPPLYproductSize >= 0),

SUPPLYquantityProduct int not null check(SUPPLYquantityProduct >= 0)

);

Створюємо таблицю для збереження виконаних заказів (Додаток В):

CREATE TABLE Сompleted\_orders

(

OLDorderID varchar(25) not null,

OLDorderPrice float not null check(OLDorderPrice > 0),

OLDorderQuantit int not null check(OLDorderQuantit > 0),

OLDuserIDRef varchar(25) not null,

OLDproductIDRef varchar (25) not null,

);

Таблиця для збереження інформації про зміни кількості побутової техніки на складі :

CREATE TABLE Change\_in\_product\_quantity

(

UPDATEproductID varchar(25) not null,

UPDATEproductName varchar(25) not null,

UPDATEoperation nvarchar(100) not null,

);