НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.2:339.173

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету	Завідувач кафедри комп'ютерних наук
інформаційних технологій	
Глазунова О.Г., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
_____	_____
____ _____ 2023 р.	**____ _____ 2023 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Аналітична система обліку складських операцій

_____

Спеціальність 122 «Комп'ютерні науки»

_____
(код і назва)

Освітня програма «Інформаційні управляючі системи та технології»

_____
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

кандидат технічних наук, доцент _____ Голуб Белла Львівна
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

кандидат технічних наук, доцент _____ Голуб Белла Львівна
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Пазій Олександра Олександрівна
(підпис) (ПІБ студента)

КИЇВ-2023

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних наук
к.т.н., доцент _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"____" _____2023 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Пазій Олександрі Олександрівні
(прізвище, ім'я, по батькові)

Спеціальність 122 «Комп'ютерні науки»
(код і назва)

Освітня програма «Інформаційні управляючі системи та технології»
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи: Аналітична система обліку складських операцій

затверджена наказом ректора НУБіП України від " 30 " грудня 2023р. № 1940-С

Термін подання завершеної роботи на кафедру 5 листопада 2023 року
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи: для дослідження були використані дані, які були надані підприємством «ТОВ ГРІН ФУД» про постачальників, клієнтів, товари та інформацію про їх постачання та вибуття зі складу.

Перелік питань, що підлягають дослідженню:

1. Аналіз та моделювання предметної області.
2. Проектування системи.
3. Дослідження можливостей використання обраних технологій для вирішення аналітичних задач обліку складських технологій.

Перелік графічного матеріалу (за потреби) _____

Дата видачі завдання " 28 " грудня 2022 р.

Керівник магістерської кваліфікаційної роботи _____ Голуб Б.Л.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Пазій О.О.
(підпис) (прізвище та ініціали студент

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DM – Data Mining

DB – Data Base

DW – Data Warehouse

CRUD – Create Read Update Delete

UML – Unified Modeling Language

ООП – об’єктно орієнтоване проектування

СУБД – система управляння базами даних

SQL – Structured Query Language

OLAP – On-Line Analytical Processing

MOLAP – Multidimensional On-Line Analytical Processing

ROLAP – Relational On-Line Analytical Processing

HOLAP – Hybrid On-Line Analytical Processing

SSAS – SQL Server Analysis Services

SSIS – SQL Server Integration Services

DSV – Data Source View

IDE – Integrated Development Environment

ВСТУП

Актуальність аналітичної системи обліку складських операцій на сьогоднішній день є надзвичайно високою. Розширення ринкових можливостей та впровадження різноманітних форм власності в Україні вимагають постійного вдосконалення бухгалтерського обліку. Це стає ключовим інструментом контролю за господарською діяльністю підприємств, які адаптуються до нових умов. Реформування економічних відносин і розвиток ринкових відносин визначають необхідність постійного оновлення підходів до обліку та звітності. Забезпечення точності і достовірності фінансової інформації стає найважливішим завданням для ефективного управління підприємствами в умовах постійної зміни економічного середовища. Сьогодні підприємства, особливо, які не використовують інноваційні технології, а пристосовані до вітчизняної економіки, перебувають не в кращому стані. За таких обставин необхідним є пошук способів і засобів підвищення їх стійкості. Одним з таких способів є використання систем обліку та аналізу складських рішень.

Раціональна організація складських операцій дає змогу керівництву підприємства мати необхідні зведення про наявність товаро матеріальних цінностей на складах і вчасно приймати рішення про їхнє поповнення та безперервне забезпечення виробництва.

Облік складських операцій переслідує завдання забезпечення збереженості матеріально товарних цінностей, які надійшли на підприємство. Дані, акумульовані в обліковій системі, дозволяють визначити, чи є в компанії неходові позиції, чи надмірність матеріалів. З роботи облікової системи ясно, наскільки велика оборотність не тільки в загальному, але і щодо конкретної різновиди матеріалу. Нарешті, в будь-який момент база даних дає відповідь на питання: «Як багато залишилося позицій на складі?» [1].

Облік і вдосконалення складських операцій – запорука якісної роботи будь-якого сучасного підприємства. Саме те, як налагоджена робота складу, визначає, наскільки раціонально компанія працює, як ефективно використовуються виробничі ресурси. Налагодження, поліпшення системи, автоматизований бухгалтерський облік складських операцій дозволяють домогтися кращої рентабельності і продуктивності, а готовий продукт стає якіснішим.

Предмет дослідження: фактори, що так чи інакше впливають на аналіз обліку складських операцій.

Об’єкт дослідження: облік складських операцій.

Метою наукового дослідження є виявлення факторів, що так чи інакше впливають на аналіз обліку складських операцій, а також розробка рекомендацій та стратегій для підвищення ефективності та якості обліку та управління складськими операціями. Дослідження спрямоване на ідентифікацію потенційних проблем та надання практичних рекомендацій для оптимізації процесів складського управління з метою підвищення конкурентоспроможності та ефективності підприємства на ринку.

Зміст поставлених завдань дослідження має наступний характер зазначений нижче.

Аналіз сучасного стану системи обліку складських операцій:

- Оцінка поточних методів та підходів до обліку складських операцій на підприємстві.
- Визначення основних аспектів інформаційного забезпечення обліку складських операцій.

Визначення ключових факторів, що впливають на облік складських операцій:

- Вивчення внутрішніх та зовнішніх чинників, які можуть впливати на ефективність та точність обліку складських операцій.

Аналіз проблем та недоліків існуючої системи обліку:

- Виявлення прогалин і недоліків у поточному обліку складських операцій.
- Оцінка впливу цих проблем на функціонування підприємства.

Розробка рекомендацій та стратегій для оптимізації обліку та управління складськими операціями:

- Визначення кращих практик та інноваційних підходів до обліку складських операцій.
- Розробка конкретних рекомендацій для підвищення ефективності та якості обліку на основі результатів дослідження.

Завершення дослідження та підготовка висновків і рекомендацій:

- Аналіз результатів експерименту і визначення їх впливу на ефективність складського управління.
- Формулювання загальних висновків та розробка рекомендацій для підприємства з метою підвищення ефективності обліку складських операцій.

Ці завдання спрямовані на глибокий аналіз, вдосконалення та оптимізацію системи обліку складських операцій з метою підвищення ефективності та конкурентоспроможності підприємства [2].

Для проведення дослідження обліку складських операцій і вирішення поставлених завдань можуть використовуватися наступні методи:

Аналіз літературних джерел де дослідник проводить огляд наукових статей, книг, журналів і інших джерел для ознайомлення з теоретичними аспектами обліку складських операцій та знайомства зі сучасними підходами і методами.

Спостереження. Дослідник може вивчати практичну роботу складу підприємства, спостерігаючи за процесами обліку та управління складськими операціями в реальному часі.

Аналіз фінансової та облікової звітності. Дослідник аналізує фінансові звіти та дані обліку складських операцій підприємства для виявлення тенденцій, проблем та можливих областей покращення.

Наукова новизна даного дослідження полягає в комплексному підході до аналізу та оптимізації системи обліку складських операцій на підприємстві з використанням передових методів Data Mining та мови програмування R. Вперше було розроблено методологію, яка охоплює аналіз сучасного стану обліку, виявлення факторів, що впливають на його ефективність, а також розробку конкретних рекомендацій для підвищення якості та ефективності управління складськими операціями. Цей комплексний підхід дозволяє отримати нові наукові знання та практичні рекомендації, які можуть бути використані підприємствами для підвищення ефективності управління складськими ресурсами і покращення їхньої конкурентоспроможності на ринку. Такий підхід є інноваційним і відкриває нові можливості для оптимізації складського управління в умовах сучасної економічної динаміки.

Результати дослідження були апробовані на науково-практичній конференції студентів і аспірантів під назвою «Теоретичні та прикладні аспекти розробки комп’ютерних систем 2023» [3] та XIV міжнародній науково-практичній конференції молодих вчених «Інформаційні технології: економіка, техніка, освіта» [4].

В структурі пояснювальної записки визначено перелік умовних позначень, вступ, та чотири розділи. У вступі вказано актуальність, проблематику, предмет, об’єкт, мету, завдання, методи дослідження, а також зазначено апробації роботи.

У першому розділі подана інформація про предметну область, розглянуті інформаційні джерела, оцінено існуючі рішення та сформульовані основні завдання майбутньої роботи. В другому розділі розглянута теоретична частина,

зокрема, моделювання системи та використані технології. Також представлені діаграми, що описують розроблену інформаційно-аналітичну систему та описано вузли, з яких збирається інформація. У третьому розділі описано розробку системи, включаючи створення сховища даних, підготування та обробку інформації, заповнення сховища даних. В четвертому розділі роботи буде здійснено дослідження кількох методів Data Mining з метою визначення їх ролі та важливості в контексті використання. У висновках наведено короткі підсумки кожного розділу, висвітлено результати дослідження та виконані завдання, і надані рекомендації.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Складські операції відіграють важливу роль у функціонуванні будь-якого підприємства та є критичним елементом у ланцюзі постачання. Ця область включає в себе широкий спектр процесів, що охоплюють отримання, зберігання та розподіл товарів або матеріалів. Ці операції становлять суттєву частину діяльності підприємства, впливаючи на якість обслуговування клієнтів, загальну ефективність та конкурентоспроможність на ринку.

Системи обліку складських операцій не тільки спрямовані на забезпечення належного управління запасами, але й включають процеси контролю якості, планування замовлень, оптимізацію простору та безперервність постачання. Вони є критичними для забезпечення надійної роботи підприємства та задоволення потреб клієнтів, вирішуючи такі важливі завдання, як забезпечення наявності необхідних товарів та уникнення надмірного запасу.

Посилення ефективності управління складськими операціями вимагає глибокого розуміння сучасних викликів, з якими стикаються підприємства. Вибір оптимальних стратегій управління та використання сучасних інструментів стають основними у завданні оптимізації робочих процесів.

В сучасному бізнес-середовищі, системи обліку складських операцій відчувають тиск для впровадження інновацій та автоматизації. Нові технології,

такі як системи штучного інтелекту, аналітики даних та програмне забезпечення для автоматизації процесів, стають ключовими для покращення продуктивності та відповідності попиту. Такі вимоги відкривають нові перспективи у використанні інноваційних підходів та впровадженні передових рішень у сфері управління складськими операціями.

1.2 Аналіз існуючих рішень та їх проблем

Огляд існуючих методів та підходів до обліку складських операцій відображає багатогранність стратегій, які використовують підприємства та організації для управління запасами та оптимізації складських операцій. Одним з найважливіших методів є Just-In-Time (точно вчасно), який розвинувся в рамках японського управління якістю. Цей підхід передбачає постачання товарів на склад лише в момент їхньої актуальної потреби для виробництва чи продажу. Він допомагає підприємствам знизити витрати на утримання запасів, оптимізувати логістичні процеси та зменшити ризики пов’язані з застою запасів. Однак цей метод вимагає дуже точного планування та співпраці з надійними постачальниками, оскільки недооблік або затримки можуть призвести до дефіциту товарів. І.Ю. Вольвач у своєму науковому досліджені наводить приклад практичного застосування системи «точно вчасно», що приносить японським машинобудівним фірмам значну вигоду. А саме: те що система вперше була запропонована та випробувана на підприємствах Toyota. Засновником концепції вважається Т.Оно, який почав роботу в Toyota Motor Co в 1943 році. Протягом декількох десятків років автомобільна корпорація створювала свою систему управління, інтегруючи кращі світові практики. Система just-in-time була введена у 1962 р [5].

Ще одним поширеним методом є ABC-аналіз. В рамках цього підходу товари класифікуються на категорії A, B і C в залежності від їхньої важливості та популярності. Товари категорії A є найважливішими та мають найвищий попит, тому їх облік та управління здійснюються найретельніше. Товари категорії B є менш важливими, але все ще потребують уваги, тоді як товари

категорії С мають менший вплив на підприємство і потребують менше уваги та ресурсів. І. В. Кулакоський у своїй роботі про «Логістика та методи логістичного аналізу» зазначає що, ідея методу АВС полягає в тому, щоб з цілої безлічі схожих об’єктів виділити найбільш значущі з точки зору визначеної мети. Таких об’єктів, як правило, небагато, і саме на них необхідно зосередити основну увагу й сили [6].

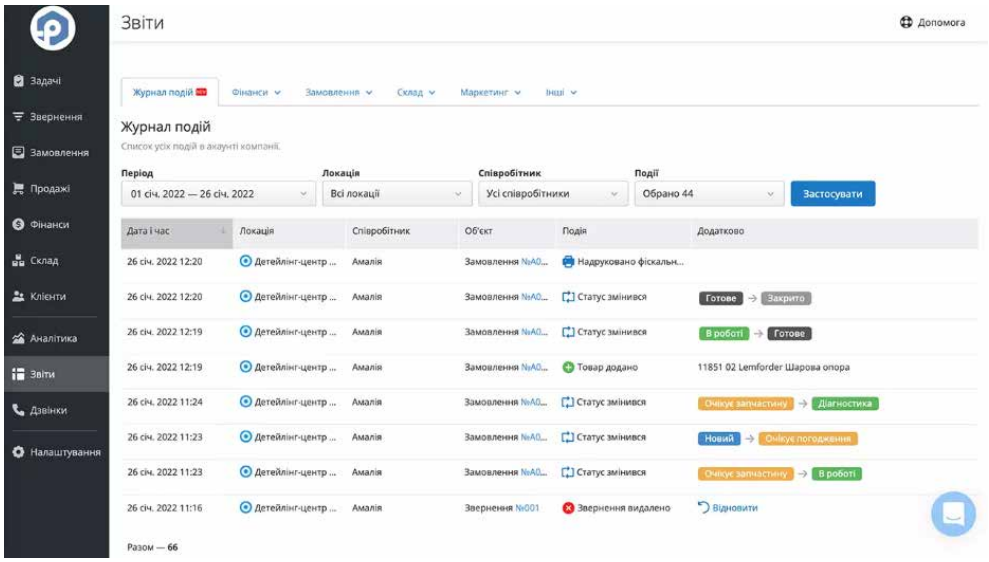
Цей метод допомагає зосередити увагу на найважливіших позиціях та оптимізувати управління запасами, концентруючись на ефективному розподілі ресурсів [7].

Додатково, багато компаній використовують методи інвентаризації, які можуть бути періодичними або постійними. Періодична інвентаризація включає в себе перевірку запасів з певною періодичністю, що допомагає виявляти розбіжності між реальними запасами та даними у системі обліку. Постійна інвентаризація, у свою чергу, передбачає постійний моніторинг запасів та їх оновлення в режимі реального часу, що дозволяє уникнути дефіциту або перевищення запасів.

Засновництво ефективної системи управління складськими операціями в сучасному бізнес-середовищі вимагає від компаній обирати серед великої кількості наявних програмних рішень та технологій. Велика частина існуючих систем, хоч і відповідають базовим потребам, проте зазвичай обмежуються обов’язковими CRUD-операціями (створення, читання, оновлення, видалення), і не завжди відповідають потребам сучасних високопродуктивних підприємств.

Проте, серед цього розмаїття рішень можна виокремити системи, аналогічні RemOnline [8], які відзначаються більш потужним функціоналом. Ці системи надають можливість генерувати розширені звіти, полегшують прийняття управлінських рішень і сприяють оптимізації складських операцій. Такі рішення стають кроком уперед, але вони досі не враховують важливого аспекту — аналітику на рівні передбачення. Інтерфейс даної системи представлений на рис. 1.2.1. – 1.2.2.

1.2.1 Скріншот програми RemOnline(1)



1.2.2 Скріншот програми RemOnline(2)

1.3 Постановка завдання для магістерської роботи

Постановка завдання для магістерської роботи є ключовим етапом у визначенні напрямків та цілей дослідження. Головною метою цієї магістерської роботи є проведення аналізу та оптимізація системи обліку складських операцій. Для досягнення цієї мети передбачено кілька конкретних завдань.

Перше завдання полягає в проведенні огляду та детального аналізу сучасного стану системи обліку складських операцій на обраному підприємстві. Цей етап передбачає вивчення існуючих методів та підходів, які вже застосовуються на практиці.

Друге завдання включає в себе визначення ключових факторів, які впливають на ефективність обліку складських операцій та їхній вплив на діяльність підприємства. Цей етап передбачає аналіз внутрішніх та зовнішніх чинників, що впливають на облік, а також виявлення можливих проблем.

Третє завдання – розробка конкретних рекомендацій для підвищення якості та ефективності управління складськими операціями на підприємстві з використанням результатів аналізу. Ця частина дослідження передбачає розробку конкретних стратегій та рекомендацій для оптимізації обліку складських операцій.

Ця магістерська робота спрямована на розробку інноваційного підходу до обліку складських операцій, що дозволить підприємству підвищити ефективність управління ресурсами та збільшити конкурентоспроможність на ринку.

Згідно з вказаними завданнями для магістерської роботи, планується використання OLAP технологій для аналізу складських операцій. Ці технології дозволяють проводити багатовимірний аналіз даних, що є важливим для отримання комплексної картини обліку на складах. Використання OLAP дозволить розглядати дані під різними оглядовими кутами та аналізувати їх для виявлення кореляцій та тенденцій, що сприятиме зрозумінню основних параметрів обліку складських операцій та формуванню стратегій оптимізації.

До того ж, для виконання цілей дослідження передбачається використання методів Data Mining, що є ключовим аспектом магістерської роботи. Застосування методів Data Mining дозволить провести детальний аналіз та моделювання даних складських операцій для виявлення різноманітних закономірностей, кореляцій та складних взаємозв’язків між різними параметрами. Ці методи дозволять не лише знайти спільні ознаки між даними, а й виявити сховані шаблони та тенденції, що сприятиме у формулюванні конкретних рекомендацій для підвищення якості та ефективності управління складськими операціями на підприємстві.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Опис та аналіз методології системного аналізу

Оскільки, аналітична система обліку складських операцій відноситься до області інформаційної системи, а сама інформаційна система згідно до міжнародного стандарту – це система обробки інформації разом із відповідними організаційними ресурсами, такими, як людські, технічні та фінансові, які надають і розподіляють інформацію. Виходячи із вказаного вище означення, інформаційна система включає такі основні елементи, як інформаційні потоки (інформація, дані бази даних, сховища тощо і відповідне програмне забезпечення), технічні засоби, кадрові ресурси, керівництво.

Зрозуміло, що для роботи інформаційної системи потрібні спеціалісти, які забезпечують роботу її складових. Кожен спеціаліст має доступ до обмеженої частини системи, а отже, має різне уявлення про систему. Наприклад, програміст бачить тільки роботу програмного забезпечення підприємства, системний адміністратор – організацію роботи та налаштування комп'ютерного обладнання, бухгалтер – фінансово-економічний стан підприємства тощо.

Задачею керівника підприємства є прийняття рішень. Але для прийняття рішень потрібно не просто володіти інформацією, звітами і іншими документами від різних підрозділів, але й розуміти, як ця інформація поєднана між собою. І це лише половина справи, оскільки стосується тільки внутрішньої інформації. Утім, кожне підприємство працює для задоволення певних потреб ринку і залежить від

ситуації на ринку, яка визначається різноманітними складними економічними показниками. Крім цього, жодна фірма не існує окремо від суспільства чи держави, а отже, потрібно враховувати також суспільно-політичні фактори і тенденції. Якщо підприємство має досить великі розміри, то вказані задачі є не тільки досить складними, щоб їх виконувати одноосібно керівнику підприємства, але і потребують знання спеціальних методів, засобів і програмного забезпечення.

Це все привело до виникнення в сучасній науці такого напряму, як системний аналіз.

Облік складських операцій у будь-якої сучасної компанії, яка бажає прийти до успіху і уникнути втрат, повинен будуватися, виходячи із специфіки роботи фірми, її масштабів і особливостей товарів, з яким потрібно взаємодіяти. А працівники мають бачити та уявляти систему всі з одного боку, та розуміти як воно все працює. Для цього і прибігають до використання системного аналізу.

Системний аналіз – це методологія теорії систем, що полягає в дослідженні будь-яких об’єктів, що представляються в якості систем, проведенні їх структуризації і подальшого аналізу. Головна особливість системного аналізу полягає в тому, що він включає в себе не тільки методи аналізу, а й методи синтезу, що означає з’єднання елементів в єдине ціле. Головною метою являється – виявлення та усунення невизначеності при вирішенні складної проблеми на основі пошуку найкращого рішення з існуючих альтернатив [9].

В основі методології системного аналізу лежать операції кількісного порівняння і вибору альтернатив у процесі прийняття рішення, що підлягає реалізації. Якщо вимога критеріїв якості альтернатив виконано, то можуть бути отримані їх кількісні оцінки. Для того щоб кількісні оцінки дозволяли вести порівняння альтернатив, вони повинні відображати беруть участь у порівнянні критерії вибору альтернатив (результат, ефективність, вартість та ін.).

У системному аналізі рішення проблеми визначається як діяльність, яка зберігає або поліпшує характеристики системи або створює нову систему із

заданими якостями. Прийоми і методи системного аналізу спрямовані на розробку альтернативних варіантів вирішення проблеми, виявлення масштабів невизначеності по кожному варіанту і зіставлення варіантів по їх ефективності (критеріями). Причому критерії шикуються па пріоритетній основі. Системний аналіз можна представити у вигляді сукупності основних логічних елементів:

- Мета дослідження – вирішення проблеми та отримання результату;
- Ресурси – наукові засоби вирішення проблеми (методи);
- Альтернативи – варіанти рішень і необхідність вибору одного з декількох рішень;
- Критерії – засіб (ознака) оцінки вирішеності проблеми;
- Модель створення нової системи.

Причому формулювання мети системного аналізу відіграє визначальну роль, так як вона дає дзеркальне відображення існуючої проблеми, бажаний результат її вирішення і опис ресурсів, за допомогою яких можна досягти цього результату.

Основні завдання системного аналізу:

- декомпозиція, тобто розкладання системи (проблеми) на окремі підсистеми (завдання);
- аналіз полягає у визначенні законів і закономірностей поведінки системи за допомогою виявлення системних властивостей і атрибутів;
- синтез вод до створення нової моделі системи, визначенню її структури і параметрів на основі отриманих при вирішенні завдань знань та інформації.

У концепції системного аналізу процес вирішення будь-якої складної проблеми розглядається як рішення системи взаємопов’язаних завдань, кожна з

яких вирішується своїми предметними методами, а потім здійснюється синтез цих рішень, оцінюваний критерієм (або критеріями) досягнення вирішення даної задачі.

2.2 Загальні відомості про моделювання

2.2.1 Визначення UML. Уніфікована мова моделювання (UML) є стандартом для визначення та специфікації програмних систем, систем ділового проектування та інших складних системних структур. Вона стала незамінним інструментом для розробки, аналізу та документування програмних проектів. UML надає стандартні засоби для створення графічних моделей систем, що полегшує спільне розуміння проектів між різними учасниками команди, а також сприяє створенню більш точної та докладної документації.

Основними компонентами UML є діаграми, які дозволяють відобразити різні аспекти системи та її структуру. Серед найпоширеніших видів діаграм UML можна виділити діаграми класів, які використовуються для моделювання структури класів, їх властивостей та взаємозв’язків. Діаграми послідовності дозволяють представити взаємодію об’єктів та компонентів системи у часовому порядку. Діаграми активності використовуються для моделювання процесів та послідовностей подій в системі.

UML також має багато інших типів діаграм, включаючи діаграми прецедентів, діаграми стану, діаграми компонентів та багато інших. Кожен з цих видів діаграм призначений для конкретного типу моделювання і дозволяє враховувати різні аспекти системи. Отже, UML надає широкі можливості для детального та комплексного аналізу та проектування систем будь-якої складності.

2.2.2 Об’єктно-орієнтоване проектування. Об’єктно-орієнтоване проектування (ООП) – це методологія проектування, яка базується на ідеї моделювання програмних систем за допомогою об’єктів, що взаємодіють між собою. У цій методології програмна система розглядається як набір окремих об’єктів, які мають властивості і поведінку, і взаємодіють один з одним для виконання певних завдань. ООП дозволяє моделювати реальний світ у програмному коді, що робить розробку більш структурованою, легко зрозумілою і підтримуваною.

UML реалізує об’єктно-орієнтований підхід до розробки складних систем за допомогою таких інструментів і концепцій:

- Самостійні об’єкти: UML дозволяє представити програмну систему як набір індивідуальних об’єктів, які мають свої властивості та поведінку.
- Класи: В UML класи групують об’єкти, які мають подібні властивості та методи. Класи визначають структуру об’єктів.
- Інкапсуляція: В UML дані та методи об’єктів зазвичай інкапсульовані в межах класу, що дозволяє приховувати деталі реалізації та надавати загальний інтерфейс для взаємодії з об’єктами.
- Поведінка об’єктів: UML дозволяє моделювати поведінку об’єктів, включаючи їх взаємодію з навколишнім середовищем та іншими об’єктами.
- Успадкування: UML підтримує концепцію успадкування, де один клас може успадковувати властивості та методи іншого класу, створюючи ієрархію класів.
- Поліморфізм: UML дозволяє описувати поліморфізм, що включає в себе перевизначення поведінки об’єктів через операції та методи.

- Групи класів: Для кращої організації великих систем UML дозволяє об’єднувати класи в групи або використовувати модульний підхід при проектуванні.

UML надає стандартизований набір символів і діаграм для моделювання об’єктно-орієнтованих систем, що сприяє зрозумілості і спільному розумінню проекту всередині команди розробників.

Об’єктно-орієнтоване проектування і структурне проектування мають спільну мету – спростити розробку складних систем. Проте підхід об’єктно-орієнтованого проектування використовує описи класів і об’єктів як ключовий засіб для досягнення цієї мети. За визначенням Г. Буча, об’єктно-орієнтоване проектування – це методологія, яка об’єднує процес об’єктної декомпозиції та використання логічних і фізичних, статичних і динамічних моделей проектуємої системи [25].

Об’єктно-орієнтоване проектування логічно продовжує об’єктно-орієнтований аналіз і використовує його результати як основу. Однак, важливо відзначити, що результати будь-якого з методів структурного аналізу також можуть бути використані для об’єктно-орієнтованого проектування. У такому випадку відбувається інтеграція діаграм потоків даних з класами та об’єктами в процесі проектування.

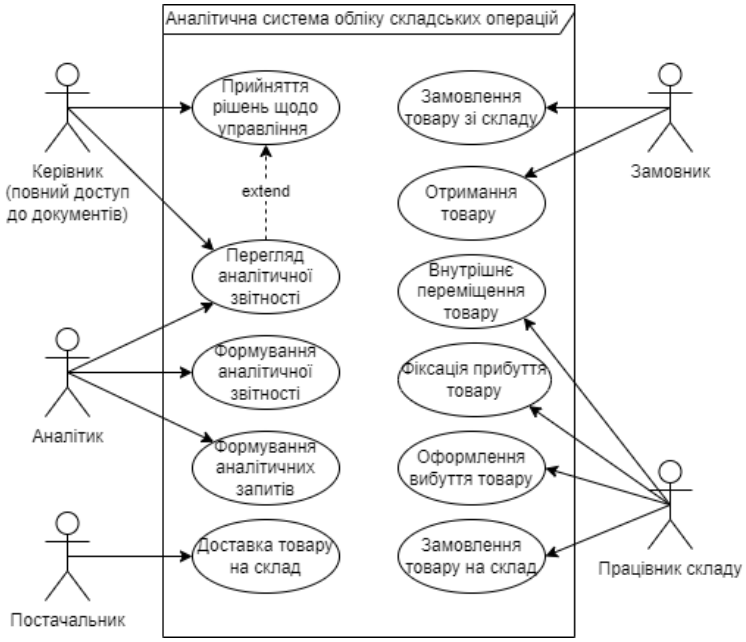
2.3 Діаграма прецедентів

Діаграма прецедентів є важливим інструментом моделювання системи в рамках UML. Вона використовується для графічного відображення функціональних можливостей системи та взаємодії зовнішніх сутностей з системою.

Діаграма прецедентів є однією з ключових частин UML і вона відображає функціональний аспект системи. Прецеденти у цій діаграмі представляють собою конкретні функціональні можливості або дії, які може виконувати система. Вони допомагають визначити, як система взаємодіє зі своїм оточенням та які завдання вона виконує.

Окремі актори виступають як зовнішні сутності, які взаємодіють з системою. Вони можуть бути користувачами, іншими системами або навіть зовнішніми процесами. Діаграма прецедентів дозволяє визначити, які актори мають доступ до яких прецедентів та як ця взаємодія відбувається [10].

У результаті аналізу було створено діаграму, зображену на рис. 2.3.1.



2.3.1 Діаграма прецедентів

Запропонована діаграма прецедентів відображає основні взаємодії та функціональні можливості системи управління складом. У цій діаграмі виокремлені різні «актори», або учасники, які взаємодіють з системою, і «прецеденти», що представляють собою конкретні функціональні можливості, які система надає. Це допомагає краще розуміти, як користувачі та зовнішні сторони спілкуються з системою та які завдання вони можуть виконувати.

Актори, такі як «Працівник складу», «Постачальник», «Замовник» і «Аналітик», представляють собою основні ролі в системі. Кожен актор має свої власні прецеденти, або функціональні операції, які він може виконувати. Наприклад, «Працівник складу» може створювати замовлення на новий товар, «Постачальник» може здійснювати доставку товару, а «Замовник» може робити замовлення товару.

Діаграма також показує, що система підтримує аналітичні можливості, представлені «Аналітиком». Він може формувати аналітичні запити, створювати аналітичну звітність та переглядати результати аналізу. Крім того, є можливість «extend» для прецеденту «Перегляд аналітичної звітності», що дозволяє «Керівнику» приймати рішення щодо управління на основі аналітичної інформації.

Отже, ця діаграма прецедентів служить важливим інструментом для розуміння функціональних вимог до системи управління складом і визначення того, як користувачі будуть взаємодіяти з нею. У цьому дослідженні будемо використовувати діаграму прецедентів для аналізу та проектування системи. Такий підхід спрощує процес проектування і дозволяє створити більш ефективну систему з точки зору функціональності та взаємодії з користувачами.

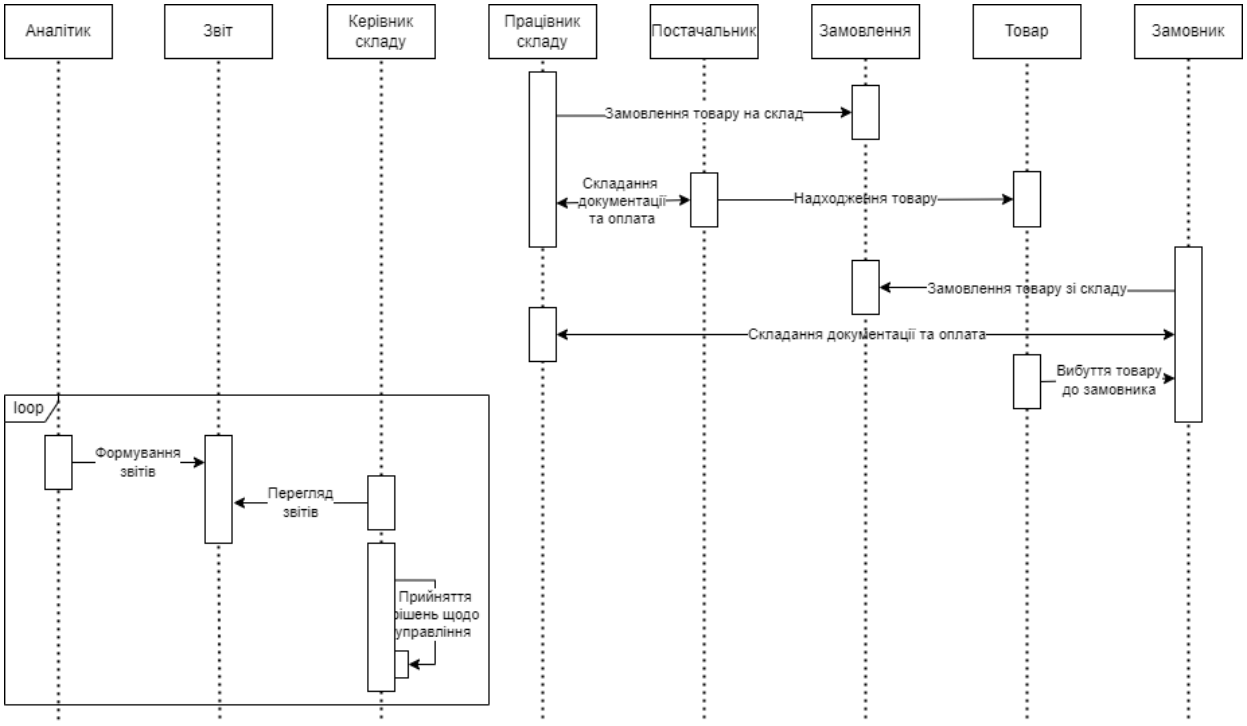
2.4 Діаграма послідовності

Діаграма послідовності є ще однією важливою складовою моделювання системи управління складом. Ця діаграма допомагає візуалізувати послідовність взаємодій між різними об’єктами або елементами системи під час виконання конкретного сценарію або функціонального процесу. У контексті управління складом, діаграма послідовності може використовуватися для уточнення та ілюстрації процесів, пов’язаних із замовленнями, вдачею товарів та багатьма іншими аспектами управління складом.

Ця діаграма включає в себе акторів та об’єкти, які беруть участь у взаємодії. Актори можуть бути користувачами системи, зовнішніми сутностями або іншими членами організації, які мають роль у цьому процесі. Об’єкти представляють конкретні елементи системи, такі як замовлення, товари на складі тощо [11].

Діаграма послідовності відображає послідовність повідомлень та взаємодій між цими акторами та об’єктами у певному порядку. Вона дозволяє деталізувати та уточнити, як саме відбуваються конкретні операції та взаємодії під час виконання конкретного сценарію або завдання.

Згідно з інформацією зазначеною у попередніх розділах, було створено діаграму представлену на рис 2.4.1.



2.4.1 Діаграма послідовності

В даній діаграмі послідовності зображено різні сценарії взаємодії між акторами та сутностями в контексті управління складом та аналітичної діяльності. Перший сценарій включає в себе спільну роботу «Працівника складу» та «Замовлення», де «Працівник складу» ініціює процес «Замовлення товару на склад» створюючи «Замовлення товару на склад» і відправляючи відповідний запит.

Другий сценарій об’єднує «Працівника складу» та «Постачальника» у спільній дії «Складання документації та оплата» «Працівник складу» і «Постачальник» спільно здійснюють оформлення необхідної документації та оплату. Цей процес може виникнути при закупівлі товарів, коли обидва боки повинні згодитися на умови та документи.

Третій сценарій – «Надходження товару» – стосується «Постачальника» та «Товару», який надходить на склад. «Постачальник» додає товари на склад, і ця подія реєструється в системі складу.

Четвертий сценарій включає взаємодію між «Замовником» та «Замовленням», де «Замовник» ініціює процес «Замовлення товару зі складу» а «Працівник складу» допомагає відправити необхідний товар. Це може стосуватися клієнтів, які замовляють товари зі складу.

П’ятий сценарій – «Вибуття товару до замовника» - описує, як «Товар» надсилається «Замовнику». Тут «Товар» виступає як окремий об’єкт, і його відправка до «Замовника» підтверджується в системі.

Окремий цикл аналітичних процесів включає в себе дії «Аналітика» та «Звіту», а також «Керівника складу». «Аналітик» формує звіти, а «Керівник складу» переглядає їх та приймає управлінські рішення на основі аналізу. Цей цикл може бути важливим для контролю та прийняття стратегічних рішень в управлінні складськими процесами.

Всі ці сценарії та цикл аналітики відображають послідовність подій та взаємодій між акторами та сутностями у контексті управління складом та аналізу даних.

У підсумку можна сказати, що діаграми послідовності є важливі для візуалізації послідовностей подій та взаємодій між об’єктами та акторами у системі. Вони дозволяють ретельно проаналізувати процеси, які відбуваються в системі, визначити послідовність подій та об’єктів, які взаємодіють між собою, і надають змогу зрозуміти, як система функціонує на рівні взаємодій між її складовими частинами.

Ці діаграми особливо корисні при проектуванні та аналізі систем, де важливо враховувати послідовність та умови взаємодій між різними елементами. Вони допомагають виявити потенційні проблеми, оптимізувати процеси та зробити систему більш ефективною.

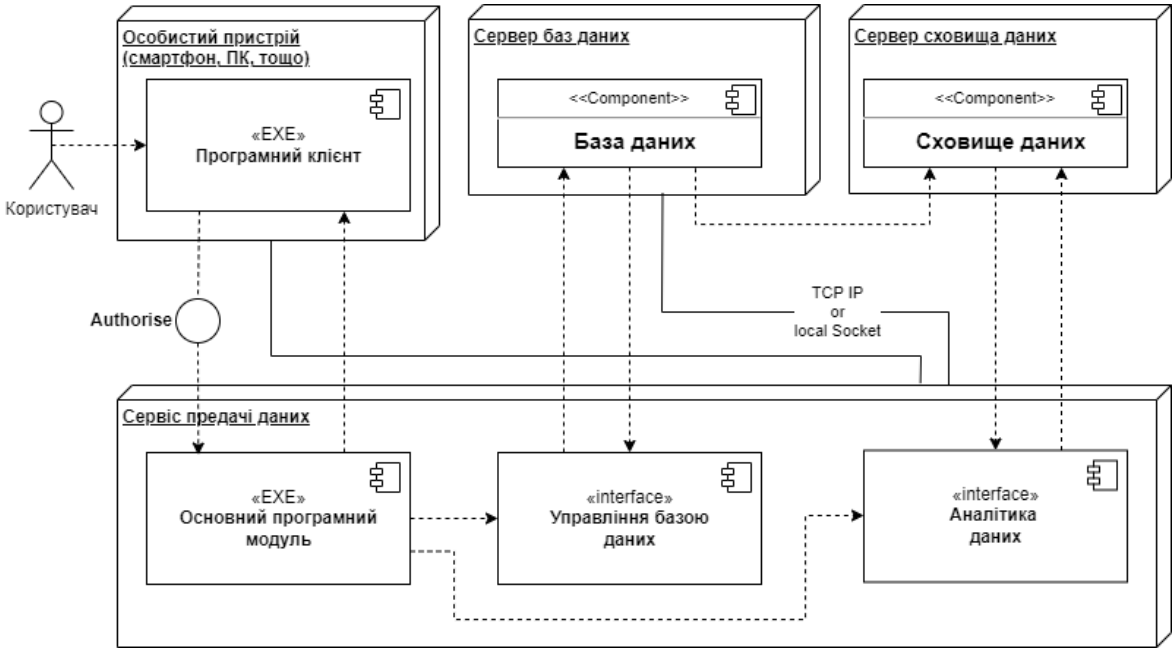
Таким чином, діаграми послідовності є необхідним інструментом для інженерів систем та розробників програмного забезпечення, які прагнуть створити добре структуровані та функціональні системи з урахуванням всіх взаємодій та послідовностей подій у них.

2.5 Діаграма розгортання і компонентів

Діаграма розгортання і компонентів є частиною моделювання системи в області інформаційних технологій та програмного забезпечення. Ця діаграма дозволяє інженерам систем розглядати аспекти фізичної реалізації програмних систем та апаратного забезпечення, яке їх підтримує. Вона надає змогу відобразити взаємодію між апаратними та програмними компонентами системи, а також їх фізичне розташування на реальних пристроях та серверах.

Діаграма розгортання дозволяє визначити, які програмні компоненти працюють на яких апаратних пристроях і як вони взаємодіють один з одним через мережу. Вона стає незамінним інструментом для архітекторів систем, які мають великі мережі компонентів та серверів [12].

Компонентна частина діаграми надає можливість детально визначити структуру програмних систем, їхні компоненти та залежності між ними. Це допомагає інженерам розробляти, розгортати та керувати програмами та компонентами системи з більшою ефективністю та точністю.



2.5.1 Діаграма розгортання і компонентів

Діаграма розгортання і компонентів представлена на рис. 2.5.1 відображає важливі аспекти фізичної реалізації системи та її компонентів. На ній

виділяються різні вузли, компоненти та інтерфейси, які взаємодіють між собою для забезпечення функціональності системи.

Актором на діаграмі є «Користувач», який взаємодіє з системою через «Особистий пристрій» з «Програмним клієнтом». Це вказує на те, що користувачі взаємодіють з системою через додаток на своєму особистому пристрої.

Слідом за цим є «Сервіс передачі даних», в якому присутній «Основний програмний модуль». Ця частина відповідає за обробку та передачу даних між «Програмним клієнтом» та іншими частинами системи. Вона також здійснює авторизацію, що може бути важливим аспектом для забезпечення безпеки системи.

Далі є інтерфейси «Управління базою даних» і «Аналітика даних». Вони підключаються до «Сервісу передачі даних» і вказують на те, що система включає в себе функціональність управління базою даних та аналітики.

Слідом за цим маємо «Сервіс баз даних», який включає «Базу даних». Ця частина відповідає за зберігання та обробку даних у системі. Вона з’єднується з «Інтерфейсом Управління базою даних», що вказує на можливість взаємодії з базою даних через цей інтерфейс.

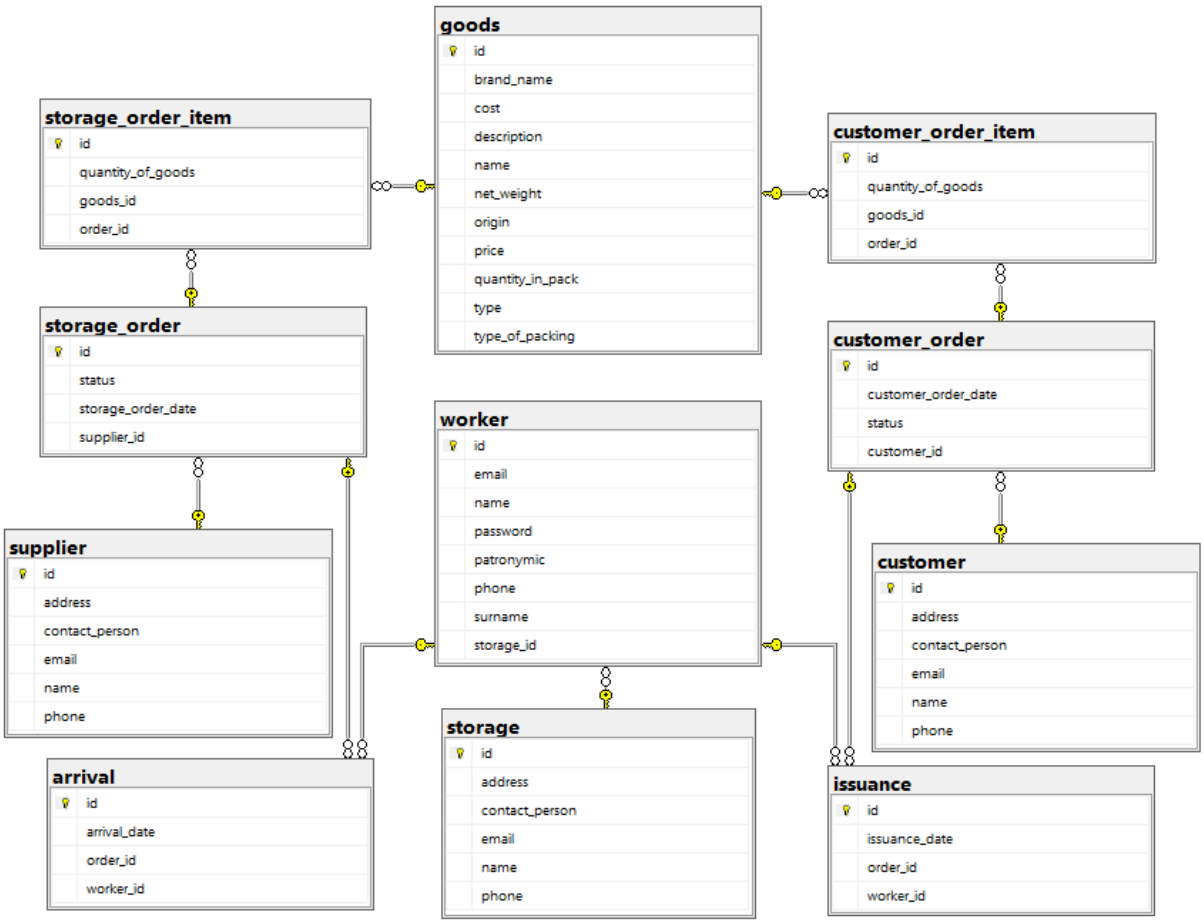
Нарешті, є «Сервер сховища даних», який має компонент «Сховище даних». Він також підключений до «Інтерфейсу Аналітики даних», що показує, що аналітика може взаємодіяти з цим сховищем даних.

Узагальнюючи, ця діаграма надає візуальне представлення структури та взаємодії компонентів системи та вказує на ключові з’єднання та інтерфейси для обміну даними та функціональністю.

3 РОЗРОБКА СИСТЕМИ

3.1 Структура джерела інформації для проведення інтелектуального аналізу

3.1.1 Структура оперативної бази даних. Дана база даних створена для детального відстеження різних аспектів бізнес-процесів, пов'язаних з управлінням та обліком товарів, замовленнями та клієнтами. Таблиця «storage» містить інформацію про склади, «worker» – дані про працівників, які прикріплені до складу. Таблиця «goods» включає дані про товари, «supplier» – дані про постачальників. Таблиця «storage_order» відображає інформацію про замовлення на склад, а «storage_order_item» – замовлені товари. Таблиця «arrival» містить дані про надходження товару на склад, інформацію про клієнтів зберігає таблиця «customer», а замовлення від клієнтів відображає таблиця «customer_order», а «customer_order_item» – замовлені товари. На кінець, таблиця «issuance» містить інформацію про видачу товарів клієнтам.



3.1.1.1 Структура оперативної бази даних

Представлена база даних на рис. 3.1.1.1 є добре розробленою і структурованою системою, яка ефективно відображає основні аспекти бізнес-процесів, пов’язаних з управлінням та обліком товарів на складі, їх постачанням і продажем клієнтам. Ця база даних відповідає трьом важливим аспектам баз даних: третій нормальній формі, правилам Бойса-Кодда та має оптимальну структуру для забезпечення ефективності обробки інформації.

Одним із головних переваг цієї бази даних є її відповідність третій нормальній формі. Це означає, що дані в базі даних розділені на окремі таблиці таким чином, що кожна таблиця містить унікальні та атомарні дані про певний аспект бізнес-процесу. Такий підхід робить базу даних добре організованою та легко зрозумілою, що полегшує її обслуговування та розширення.

Крім того, ця база даних також відповідає правилам Бойса-Кодда, що є важливим аспектом при проектуванні баз даних для забезпечення їхньої стійкості. Дані у базі зберігаються в незалежних таблицях з використанням зовнішніх ключів для забезпечення цілісності та зв’язності даних.

Окрім цього, ця база даних добре підходить для оптимізованої обробки даних у відповідності до бізнес-потреб. Вона дозволяє ефективно зберігати та відстежувати дані про товари, постачальників, клієнтів та замовлення, що сприяє точному контролю за запасами і забезпеченню якості обслуговування клієнтів.

У цій роботі було використано Систему Управління Базами Даних (СУБД) SQL Server [26] як основну платформу для зберігання та управління даними, пов’язаними з обліком складських операцій. Використання SQL Server надало можливість створення ефективної та надійної бази даних, яка відображає різноманітні аспекти бізнес-процесів у сфері управління та обліку товарів, постачальників, клієнтів та замовлень на складі.

SQL Server надає велику швидкість обробки запитів, що важливо для складських операцій, де час є критичним чинником. Його оптимізована робота з реляційними даними сприяє швидкому доступу до інформації.

Ще однією перевагою є висока рівень безпеки даних, який забезпечується за допомогою різноманітних методів шифрування та аутентифікації, що важливо для зберігання конфіденційної інформації про товари, постачальників, клієнтів та замовлення.

SQL Server також має можливості для створення резервних копій та відновлення даних, що важливо для забезпечення надійності та безпеки інформації, особливо в умовах великої кількості транзакцій, що характерне для операцій на складі.

Крім того, SQL Server має широкий спектр засобів для створення ефективних запитів та оптимізації запитів до бази даних, що сприяє підвищенню продуктивності та швидкості обробки інформації в системі обліку складських операцій.

База даних розташована на спеціалізованому сервері, який забезпечує централізоване зберігання та управління великим обсягом інформації, пов’язаної з управлінням складськими операціями. До цієї бази даних можна звертатися з різних пристроїв, використовуючи програмний клієнт, встановлений на користувацькому пристрої як показано на діаграмі рис. 2.5.1. Цей програмний клієнт дозволяє внесення даних в базу даних та отримання необхідної інформації для управління складськими операціями.

3.1.2 Загальні поняття за напрямку OLAP-технології

OLAP, або аналітична обробка у реальному часі, представляє собою інтерактивну систему, яка дозволяє швидко аналізувати багатовимірні дані і отримувати результати запитів практично миттєво. Основоположником цієї концепції є Едгар Кодд, автор реляційної моделі даних. Він запропонував правила для аналітичної обробки в реальному часі, які забезпечують ефективну роботу з багатовимірними даними.

Однією з головних мотивацій для створення OLAP була несумісність класичної реляційної моделі з потребами аналітиків, які потребують швидких та складних аналітичних запитів. Реляційні бази даних підходять для операційних

систем, але виконання складних аналітичних запитів у них може бути повільним. OLAP дозволяє створити віртуальний багатовимірний «куб» з даних та виконувати аналітику швидко та ефективно.

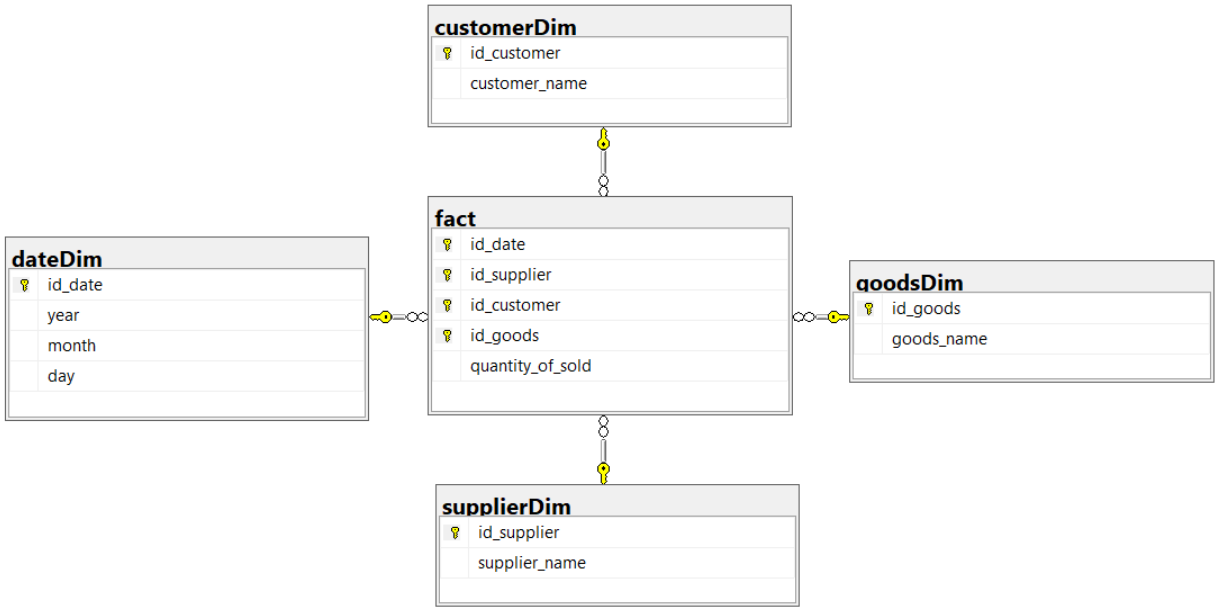
Ключовою ідеєю OLAP є використання багатовимірних кубів для представлення та аналізу даних. Осі цих кубів відображають числові або лінгвістичні дані, що характеризують предметну область. Ця концепція дозволяє використовувати різні види даних у аналітичних запитах. OLAP може бути розділений на багато видів, таких як багатовимірний OLAP (MOLAP), реляційний OLAP (ROLAP) і гібридний OLAP (HOLAP), кожен з яких має свої особливості та переваги. Наприклад, MOLAP використовує підсумовані дані та готові агрегати, ROLAP працює безпосередньо з реляційними даними, а HOLAP поєднує обидва підходи [13].

OLAP дозволяє легко аналізувати та використовувати багатовимірні дані у реальному часі, спрощуючи завдання аналітиків та допомагаючи приймати обґрунтовані рішення.

3.1.3 Структура сховища даних. Сховище даних, також відоме як Data Warehouse (DW) є централізованою та інтегрованою системою зберігання та управління великим обсягом даних з різних джерел. Основною метою створення сховища даних є забезпечення спрощеного та швидкого доступу до інформації для аналітичних операцій та прийняття управлінських рішень. Ця концепція стала популярною завдяки своїм можливостям забезпечити зручний аналіз та звітність для великих корпорацій та організацій.

Сховище даних відрізняється від традиційних операційних баз даних тим, що його структура оптимізована для аналітичних запитів і підсумкової обробки даних. Дані в сховищі зазвичай попередньо очищені, інтегровані та згруповані для спрощення аналітичного процесу. Вони також зберігаються в історичному контексті, дозволяючи аналітикам аналізувати та вивчати динаміку даних з часом.

Сховища даних грають важливу роль у великих підприємствах, банках, логістиці, медицині, роздрібній торгівлі та інших галузях. Вони допомагають підвищувати продуктивність бізнесу, приймати обґрунтовані рішення на основі даних і прогнозів, а також забезпечують надійне зберігання цінної інформації. У сучасному світі, де дані є одним з найцінніших активів, сховища даних стають невід'ємною частиною стратегії бізнесу та прийняття управлінських рішень.



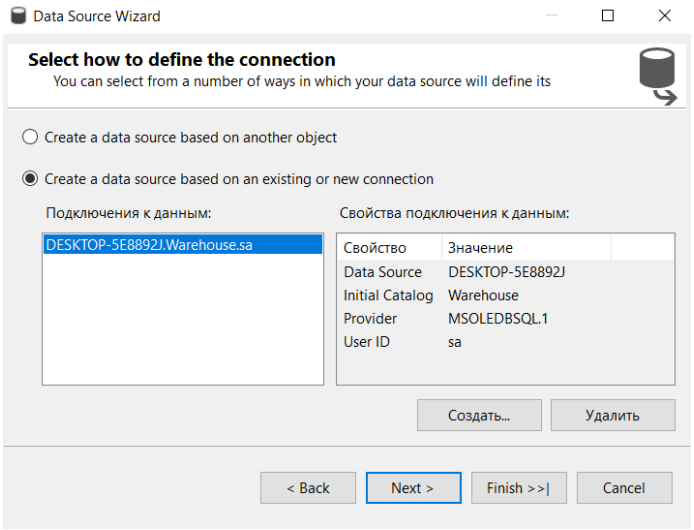
3.1.3.1 Структура сховища даних

Сховище даних у цьому контексті включає чотири ключові таблиці для аналітичного аналізу даних. Перша таблиця, «dateDim» містить інформацію про дати, розділені на рік, місяць і день, що дозволяє аналізувати дані з урахуванням часового аспекту. Друга таблиця, «goodsDim» зосереджена на даних про товари та їх найменуваннях, що полегшує аналіз товарних показників. Третя таблиця, «supplierDim» містить інформацію про постачальників і їх назви, що сприяє аналізу постачальницьких відносин. Нарешті, четверта таблиця, «customerDim» містить дані про клієнтів та їх імена, що полегшує аналіз покупців і споживчого попиту. Ці таблиці допомагають структурувати дані для подальшого аналізу і прийняття управлінських рішень. Продемонстровано на рис. 3.1.3.1.

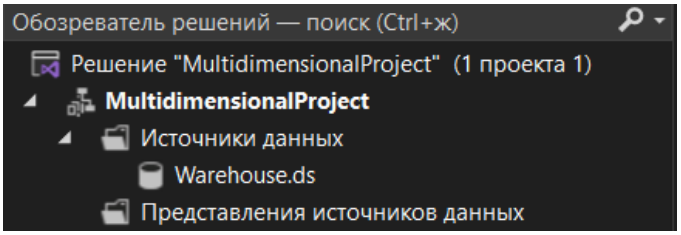
Таблиця «fact» представляє собою фактичні дані, де визначається взаємодія між датами, товарами, постачальниками, клієнтами і кількістю проданих товарів. Ця таблиця відображає основні транзакції, пов'язані з продажем товарів і дозволяє відстежувати, які товари були куплені в який день від якого постачальника та яким клієнтом. Вона є ключовою для аналітичного аналізу, оскільки дозволяє враховувати попит на товари відповідно до різних факторів, таких як час, постачальники та клієнти. Завдяки цій таблиці можна виявити тенденції в продажах, розробляти стратегії для покращення взаємодії з клієнтами та оптимізувати угоди з постачальниками.

3.1.4 Механізм вилучення, обробки і передачі даних. SQL Server Analysis Services (SSAS) – це служба, яка спрощує роботу з OLAP-кубами, надаючи можливість створювати і налаштовувати їх для аналізу даних. OLAP-куб – як зазначалось вище, це спеціальна структура даних, яка дозволяє проводити швидкий аналіз великих об'ємів і складних даних, виходячи за межі можливостей звичайних реляційних баз даних. Робота з SSAS виконується через середовище Visual Studio, де визначається джерело даних (база даних OLAP або сховище даних) та налаштовується підключення до нього для обробки та аналізу даних [14].

Спочатку було виконано підключення джерела даних та встановлено з'єднання з базою даних, як це наведено нижче на рис. 3.1.4.1 та 3.1.4.2.



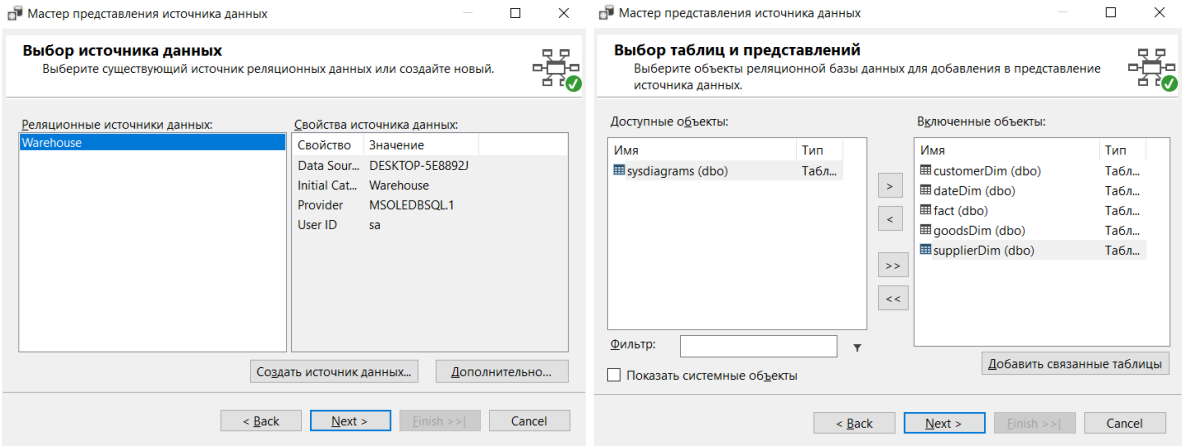
3.1.4.1 Підключення до джерела даних (1)



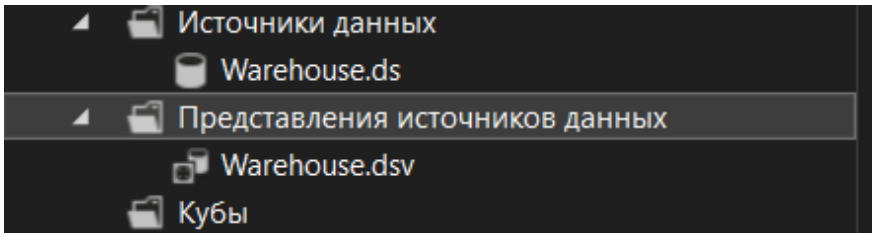
3.1.4.2 Підключення до джерела даних (2)

Створення Data Source View (DSV). Під Data Source View розуміється зріз джерела, яке буде використовуватися для заповнення сховища, при цьому в нього можуть входити як таблиці, так і уявлення (view) реляційної бази – джерела даних.

Для створення зрізу джерела необхідно: обрати джерело даних та таблиці, які необхідні вирішення задачі, як продемонстровано на рис. 3.1.4.3 та 3.1.4.4.



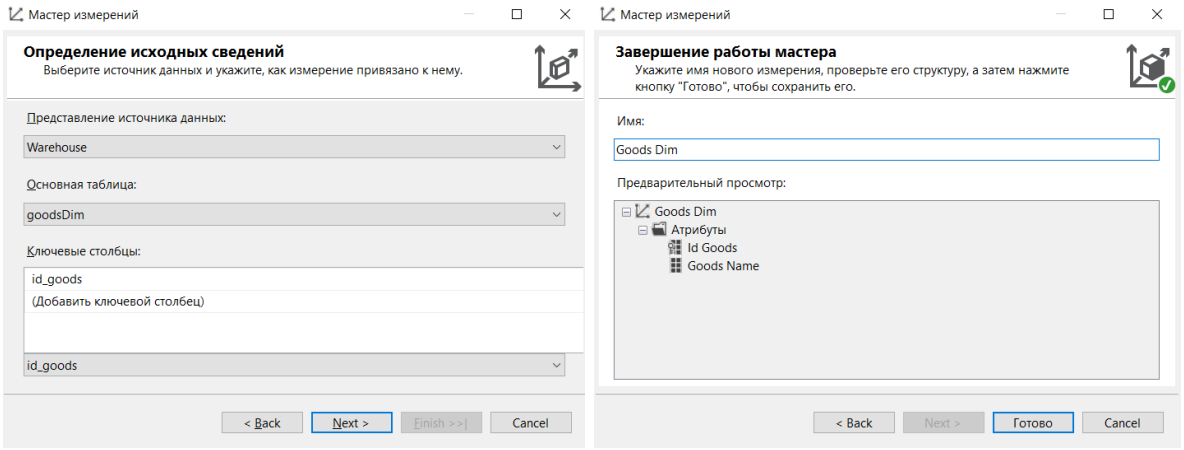
3.1.4.3 Створення уявлення (1)



3.1.4.4 Створення уявлення (2)

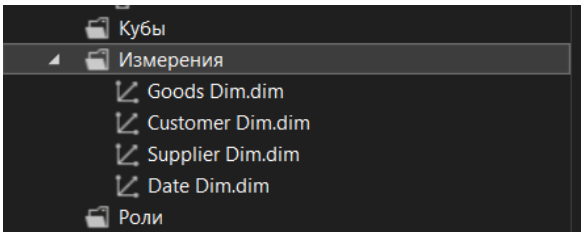
Після успішного підключення до джерела даних і визначення Data Source View, наступним кроком є створення вимірів. Виміри представляють собою ключові аспекти даних, які будуть використовуватися для агрегації та аналізу. Наприклад, це можуть бути часові періоди, категорії товарів або інші важливі атрибути даних.

Для створення вимірів використовуються дані з Data Source View, показано на рис. 3.1.4.5, і це дозволяє визначити, як саме дані будуть групуватися та агрегуватися в майбутньому аналізі. Виміри грають ключову роль у визначенні різних аспектів даних, які можна буде вивчати в OLAP-кубі.



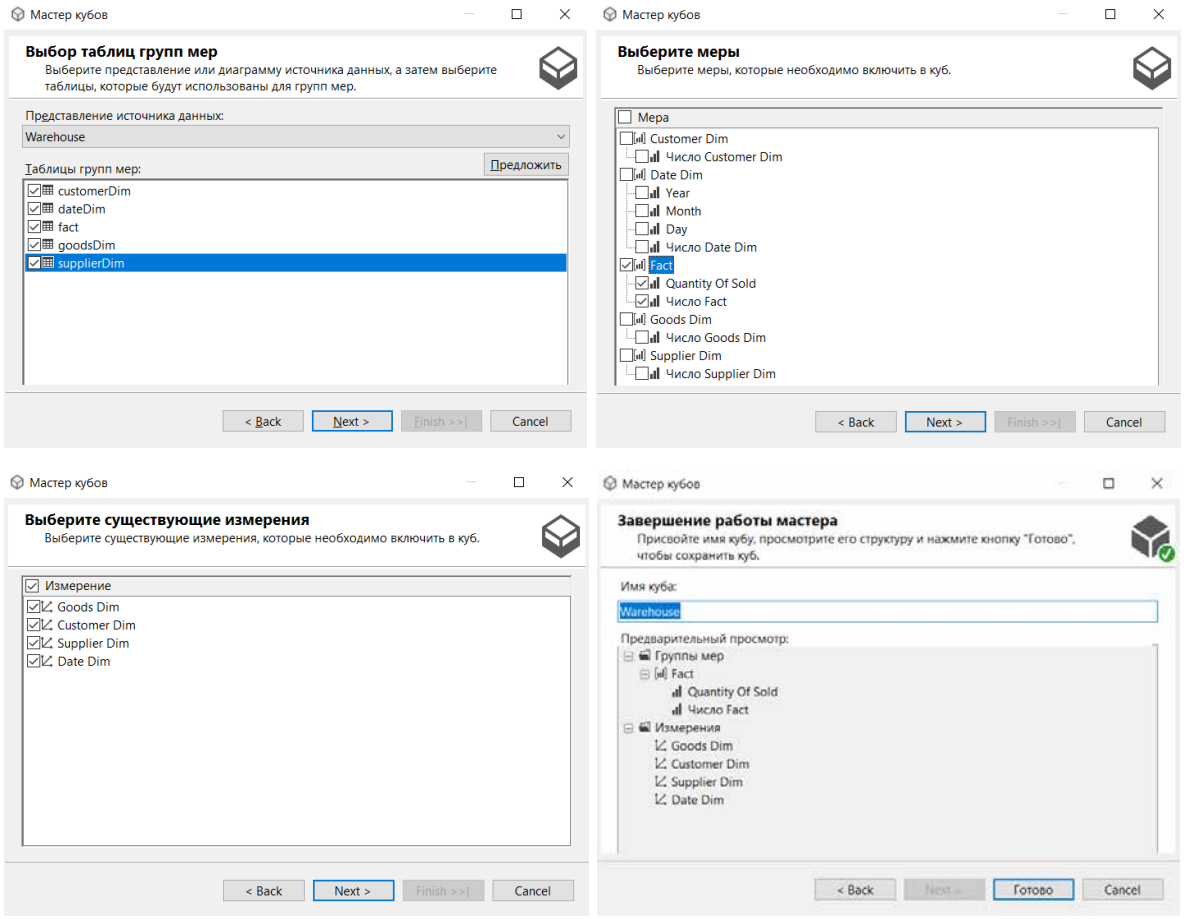
3.1.4.5 Створення виміру goodsDim

Так само створюємо інші виміри, визначаючи ключові аспекти даних, які будуть використовуватися для аналізу та агрегації. Це може включати в себе виміри, які описують характеристики клієнтів, властивості товарів або будь-які інші атрибути даних, які важливі для вашого аналізу. На рис. 3.1.4.6 продемонстровані всі створені виміри.



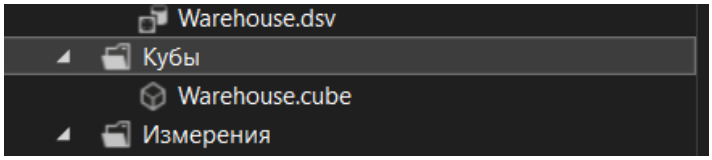
3.1.4.6 Створені виміри

На основі створеного зрізу джерела даних і визначених вимірів, використовуючи Cube Wizard (Майстер створення кубів), формуємо OLAP-куб, як зображено на рис. 3.1.4.7 – 3.1.4.9. Цей процес включає в себе створення структури куба, визначення залежностей між вимірами і фактами, а також встановлення правил агрегації даних.

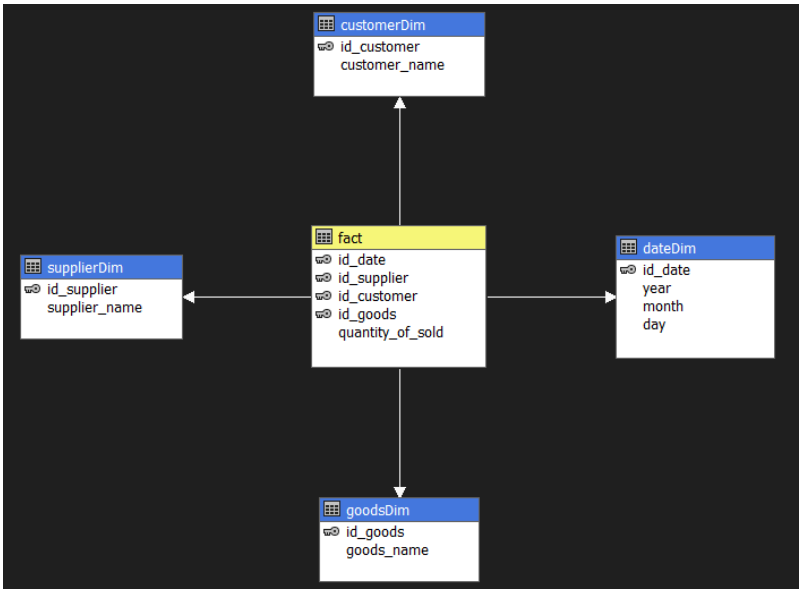


3.1.4.7 Розгортання куба (1)

На основі OLAP-кубу, можемо розпочати аналіз даних.



3.1.4.8 Розгортання куба (2)



3.1.4.9 Розгорнутий куб

Далі, для організації процесу інтеграції даних в багатовимірний куб, створюємо проект Integration Services Project, який входить до складу служби SQL Server Integration Services (SSIS). Цей проект дозволяє нам налаштовувати та керувати процесом збору та трансформації даних з різних джерел перед завантаженням їх в OLAP-куб [15].

У рамках Integration Services Project налаштовуємо підключення до джерел даних, створюємо завдання та пакети для ефективного збору та обробки даних. Важливо враховувати структуру джерел даних і їх формати.

Завдяки можливостям SSIS, можемо автоматизувати багато етапів інтеграції даних та забезпечити їхню якість та цілісність. Цей підхід дозволяє ефективно підготувати дані для подальшого аналізу в OLAP-кубі, забезпечуючи користувачів актуальними та надійними даними для прийняття рішень.

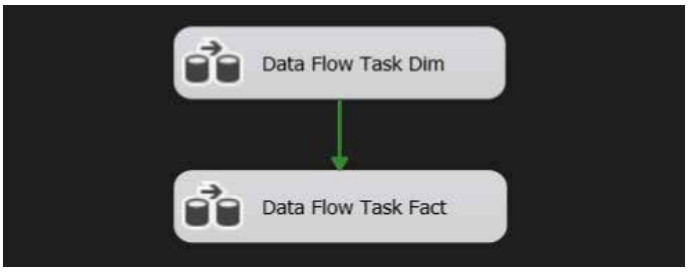
Для успішної інтеграції даних в багатовимірний куб необхідно додати завдання потоку даних до нашого проекту. Це завдання буде відповідати за ефективну передачу та обробку даних з джерела до OLAP-куба. Оскільки маємо врахувати структуру сховища даних, в нашому випадку, потрібно реалізувати два рівні потоків даних.

На першому рівні потоку даних будемо заповнювати таблиці вимірів. Цей крок включає в себе збір та трансформацію даних, необхідних для створення

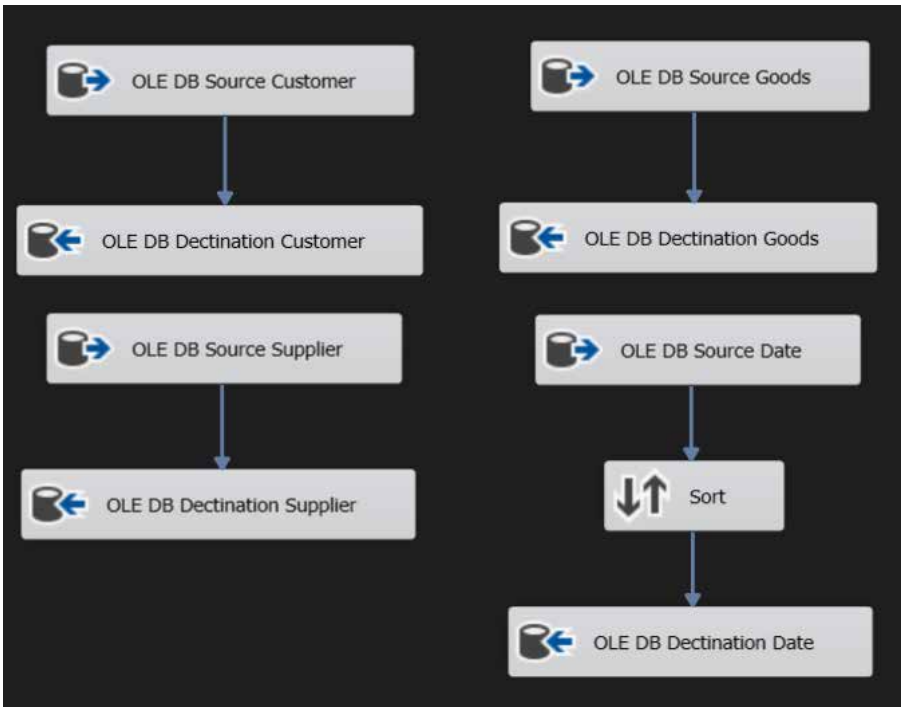
вимірів, які визначають структуру OLAP-куба та дозволяють користувачам аналізувати дані по різних аспектах, таких як час, товари, постачальники та клієнти.

На другому рівні потоку даних будемо заповнювати таблицю фактів, яка міститиме самі дані про продажі або інші обрані факти. Цей рівень включає в себе обробку деталей фактів та їх зв'язку з вимірами. Це дозволить побудувати зв'язаний OLAP-куб, який дозволить користувачам аналізувати дані та отримувати цінну інформацію для прийняття рішень.

Завдяки цим двом рівням потоку даних (рис. 3.1.4.10), забезпечимо правильну та ефективну інтеграцію даних в OLAP-куб, що є важливим етапом для подальшого аналізу та використання даних.



3.1.4.10 Рівні потоків даних



3.1.4.11 Рівень вимірів

На першому рівні потоку даних (рис. 3.1.4.11) заповнюються таблиці «goodsDim», «supplierDim» та «customerDim», які є важливими для створення вимірів в OLAP-кубі. Це дозволяє підготувати дані, які будуть використовуватися для аналізу товарів, постачальників та клієнтів.

Щодо таблиці «dateDim», використовується SQL-запит який продемонстровано на рис. 3.1.4.12 , для заповнення її даними з таблиці «issuance». Цей запит розділяє дату на рік, місяць та день, щоб створити вимір для аналізу даних за часом. Важливою частиною цього процесу є сортування дат для уникнення повторів. Сортування дат допоможе правильно побудувати OLAP-куб і забезпечити точний аналіз за часом. Цей етап є важливим для підготовки даних та їх подальшого використання в OLAP-кубі для аналізу.

Текст команды SQL:

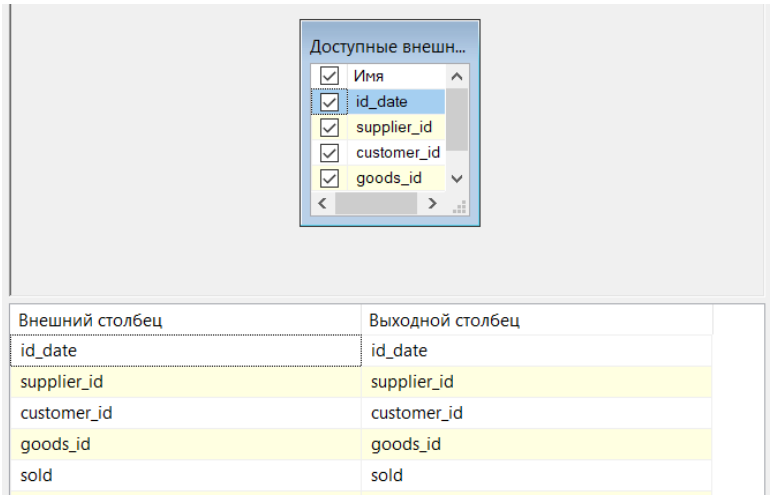
```
SELECT DISTINCT [issuance_date] as [date],
      YEAR([issuance_date]) as [year],
      MONTH([issuance_date]) as [month],
            DAY([issuance_date]) as day
FROM [Storage].[dbo].[issuance]
```

3.1.4.12 SQL запит для отримання дат

Заповнення таблиці фактів є ключовим етапом у процесі створення OLAP-куба, оскільки саме в цій таблиці зберігаються фактичні дані, які будуть аналізуватися у майбутньому.

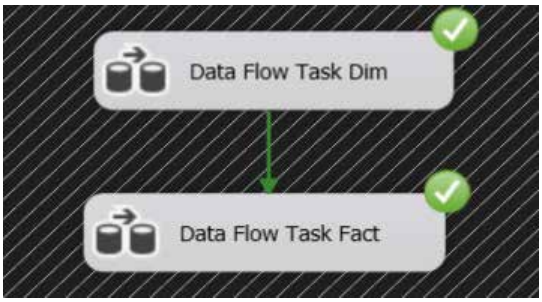
Цей етап є критичним, оскільки від якості та точності даних у таблиці фактів залежить результат аналізу у подальших операціях з OLAP-кубом.

Код SQL запиту для заповнення сховища даних з оперативної бази даних продемонстрований у додатку А.



3.1.4.13 Мапінг заповнення таблиці фактів

Успішне завершення потоків даних (рис. 3.1.4.14) свідчить про те, що інтеграція даних у проекті пройшла без помилок і дані були успішно завантажені в відповідні таблиці.



3.1.4.14 Успішна інтеграція даних

3.2 Загальні поняття технології Data Mining

Технологія Data Mining, також відома як видобування даних або аналіз даних, представляє собою процес виявлення корисних та практичних знань із великих обсягів даних. Ця технологія дозволяє відкривати різноманітні залежності, закономірності та шаблони в наборах даних, що інакше були б важко або неможливо виявити за допомогою традиційних методів аналізу.

Історія Data Mining налічує багато десятиліть розвитку, і її коріння пов’язане з областями статистики, штучного інтелекту та обробки даних. Перші спроби використання аналітичних методів для видобування знань з даних відбувалися в середині 20-го століття, але справжній розквіт Data Mining настав в 1990-х роках завдяки зростанню потужності обчислювальних систем та розвитку баз даних.

Сьогодні Data Mining є важливою складовою галузей, таких як бізнес-аналітика, фінанси, маркетинг, наука про матеріали, медицина та інші. Вона використовується для виявлення тенденцій в ринковому споживанні, покращення стратегій продажу, виявлення шахрайства, прогнозування попиту, аналізу клієнтських взаємодій, а також для вирішення багатьох інших завдань.

У майбутньому Data Mining матиме ще більше застосувань, оскільки кількість даних, доступних для аналізу, постійно зростає, і разом із нею розвиватиметься і сама ця технологія [16].

Загальні поняття технології Data Mining знайшли своє втілення в аналізі та оптимізації наших операцій в галузі управління складом. Data Mining стає надзвичайно корисним інструментом для розуміння і прогнозування попиту на товари, виявлення тенденцій та аномалій в постачанні та розподілі продукції.

Один з ключових аспектів використання Data Mining у нашій галузі – аналіз історичних даних про продажі та постачання. Можемо використовувати ці дані для виявлення сезонних коливань у попиті на товари, щоб належним чином планувати запаси та поставки. Крім того, Data Mining допомагає виявити незвичайні зміни в попиті або постачанні, що може бути знаком проблем або можливостей для оптимізації.

Додатково, Data Mining може використовуватися для класифікації товарів за різними критеріями, такими як прибутковість, швидкість обороту або ризик позбавлення. Це допомагає приймати більш обґрунтовані рішення щодо закупівель та розміщення товарів на складах.

У великих сховищах даних, які містять інформацію про клієнтів, постачальників та продукцію, Data Mining також допомагає виявляти нові можливості для розширення бізнесу та оптимізації логістичних процесів. Відкриття нових ринків або вдосконалення відносин з постачальниками може бути результатом аналізу даних за допомогою цієї технології.

Найбільша потреба сучасного бізнесу полягає у передбаченні майбутніх тенденцій та оптимальному управлінні запасами. У цьому контексті Data Mining

стає стратегічним інструментом. Вона надає можливість аналізувати великі обсяги даних, виявляти корисні закономірності і шаблони, а також прогнозувати попит на товари. Інтеграція Data Mining у системи управління складськими операціями може революціонізувати бізнес-процеси, зробити їх більш стратегічними і передбачуваними.

Застосування Data Mining дозволить підприємствам аналізувати ринкові тенденції, передбачати попит на товари, та оптимізувати управління запасами. Завдяки цій інноваційній можливості, бізнес стає більш конкурентоспроможним, знижуючи витрати та збільшуючи ефективність управління складськими операціями. Основним завданням Data Mining є аналітика, що допомагає не лише реагувати на події, але і передбачати їх, що дозволяє приймати більш обґрунтовані та стратегічні рішення в управлінні запасами та складськими процесами.

Класифікація є однією з ключових задач в області Data Mining. Ця задача полягає в призначенні категорій чи міток для об’єктів на підставі їхніх характеристик. Для класифікації використовуються алгоритми, такі як 1-Rule, Наївний Байес, Метод Опорних Векторів, Рішення Дерев та К-найближчих сусідів. Наприклад, Наївний Байес застосовується для призначення категорій на основі ймовірностей характеристик, під час як Рішення Дерев розбивають дані на деревоподібні структури для призначення категорій.

Асоціативні правила використовуються для пошуку цікавих зв’язків та асоціацій між різними елементами в даних. Головною метою цієї задачі є виявлення відносин між різними об’єктами або подіями в даних. Основні алгоритми для асоціативних правил включають Apriori, Eclat та FP-Growth. Наприклад, алгоритм Apriori використовується для виявлення часто відбуваючихся зв’язків між елементами та генерації правил асоціацій.

Кластеризація – це метод розділення об’єктів на групи або кластери на підставі їхніх характеристик чи властивостей. Для кластеризації використовуються алгоритми, такі як k-середніх, ієрархічна кластеризація,

агломеративна кластеризація, та DBSCAN. Наприклад, k-середніх розділяє об’єкти на k кластерів залежно від схожості характеристик об’єктів, прагнучи зменшити внутрішні варіації кожного кластера.

Ці алгоритми використовуються для аналізу даних в галузі управління складом, де вони допомагають класифікувати товари за різними критеріями, виявляти зв’язки між різними елементами бізнес-процесів та групувати товари залежно від їхньої подібності. Комбінація цих методів дозволяє виявляти тенденції, прогнозувати попит та виявляти можливості для оптимізації управління запасами та складськими процесами.

У підсумку, Data Mining в галузі управління складом стає невід’ємною складовою для прийняття стратегічних та оперативних рішень, спрямованих на оптимізацію процесів та досягнення конкурентної переваги.

3.3 Огляд інструментарію для реалізації задач Data Mining

В даній роботі для реалізації задач Data Mining була обрана мова програмування R [17] та інтегроване середовище розробки RStudio [18] яке надає зручний інтерфейс для роботи з даними та аналізу даних.

Мова програмування R з’явилася в 1993 році як діалект мови S і була розроблена статистиками з університету Окленда, Нова Зеландія. З тих пір R виріс у потужну мову для статистичного аналізу даних та графічного візуалізації. Завдяки відкритому коду та безкоштовній ліцензії R використовується в науці, бізнесі та в інших сферах, де потрібні статистичний аналіз та візуалізація даних. Основною перевагою R є велика кількість доступних пакетів, що дозволяють виконувати різноманітні статистичні операції та графічні візуалізації. R також підтримує програмування з використанням функцій, що дозволяє легко створювати нові функції для виконання різноманітних завдань. Іншою важливою особливістю мови R є те, що вона працює з даними у векторній формі, що дозволяє виконувати операції над великими наборами даних дуже швидко та ефективно.

Для роботи з даними в R є певні зрозумілі кроки. Після завантаження даних, перш за все, варто оглянути їх з допомогою функцій, таких як head(), tail(), str(), summary() тощо. Це допоможе отримати загальне уявлення про дані та їх структуру. Наступним кроком є обробка та очищення даних. Для цього в R є безліч функцій, таких як subset(), filter(), mutate(), arrange() та інші. Ці функції дозволяють вибирати необхідні дані, змінювати їх та сортувати за різними параметрами. І нарешті, візуалізація даних дозволяє легко та зрозуміло відображати дані в графічному вигляді. Для цього в R є багато пакетів, таких як «ggplot2», «lattice», «plotly» та інші. За допомогою цих пакетів можна створювати різні типи графіків, включаючи лінійні, кругові, стовпчикові тощо.

Щоб зрозуміло представити та відобразити дані в R їх демонструють у вигляді графіків. І можливості R досить широкі в цьому плані. Адже є певні пакети які можуть створювати різні типи, можуть їх суміщати і персоналізувати для зручності.

У R доступно безліч пакетів для статистичного аналізу даних, що дозволяють проводити різноманітні тестування гіпотез, побудову як лінійних так і множинних регресійних моделей, аналіз варіансів, кластеризацію тощо.

Машинне навчання є однією з основних галузей застосування R, оскільки мова має вбудовану підтримку для багатьох методів машинного навчання.

Деякі з методів машинного навчання, які можна використовувати в R, включають:

- Дерева рішень (Decision trees) – це метод, що використовується для прогнозування значень на основі дерева рішень, яке визначає рішення, яке необхідно прийняти на кожному кроці моделювання.
- Випадковий ліс (Random forests) – це метод, що використовується для прогнозування значень, який використовується для поєднання декількох дерев рішень випадковим чином.

- Нейронні мережі (Neural networks) – це метод, що використовується для моделювання складних взаємозв’язків між даними, що містять багато характеристик.
- Метод опорних векторів (Support vector machines) – це метод, який використовується для розділення даних на дві або більше груп, що допомагає у вирішенні задач класифікації.
- Кластеризація (Clustering) – це метод, що використовується для групування даних на основі спільних характеристик.
- Аналіз текстів та обробка природньої мови: Методи та пакети R дозволяють виконувати аналіз текстових даних, такий як аналіз емоційного забарвлення тексту, класифікація текстів, виявлення тем та багато іншого.
- Аналіз мереж та графів: У R є багато пакетів, які дозволяють аналізувати мережі та графи. Наприклад, пакет «igraph» дозволяє виконувати різноманітні операції з графами, такі як пошук найкоротших шляхів, класифікація вершин та багато іншого.
- Аналіз зображень: Методи машинного навчання можуть бути застосовані для обробки зображень, наприклад, для класифікації зображень, виявлення об’єктів на зображеннях та багато іншого. У R є пакети, які дозволяють виконувати такі завдання, наприклад, пакет «kerasR».
- Обробка аудіо- та відеоданих: R можна використовувати для обробки аудіо- та відеоданих, наприклад, для розпізнавання мови та аналізу відтворення музики.

- Оптимізація: У R є багато пакетів, які дозволяють виконувати оптимізацію функцій та виконувати пошук мінімумів та максимумів.

R має ряд пакетів, які дозволяють з'єднуватися з різними базами даних і виконувати на них операції з використанням мови R. Пакети «dplyr» та «dbplyr» надають можливість виконувати SQL-подібні запити до баз даних, а також спрощують процес роботи з базами даних в R. Додатково, пакет «RMySQL» надає можливість з'єднуватися з MySQL, а пакет «RPostgreSQL» – з PostgreSQL базами даних.

Отже R є популярною мовою програмування, яка використовується в Data Mining, зокрема для статистичного аналізу та візуалізації. Вона надає широкий спектр пакетів та функцій, які можуть використовуватися для виконання різних завдань, пов'язаних з Data Mining. R має багато переваг, таких як його відкрите джерело, простий синтаксис та наявність багатьох корисних пакетів.

RStudio є інтегрованою середою розробки (IDE) для R, яка надає користувачам зручний інтерфейс для аналізу та візуалізації даних. Вона дозволяє користувачам писати та запускати R-код, керувати проектами та переглядати дані в різноманітних форматах. RStudio також містить кілька корисних функцій, таких як підсвічування синтаксису, автозаповнення коду та інструменти для налагодження.

Загалом, Data Mining є важливою галуззю, яка грає ключову роль у багатьох галузях промисловості, а з наявністю потужних інструментів, таких як R та RStudio, стало легше вилучати корисну інформацію з великих наборів даних та приймати обґрунтовані рішення на основі цієї інформації.

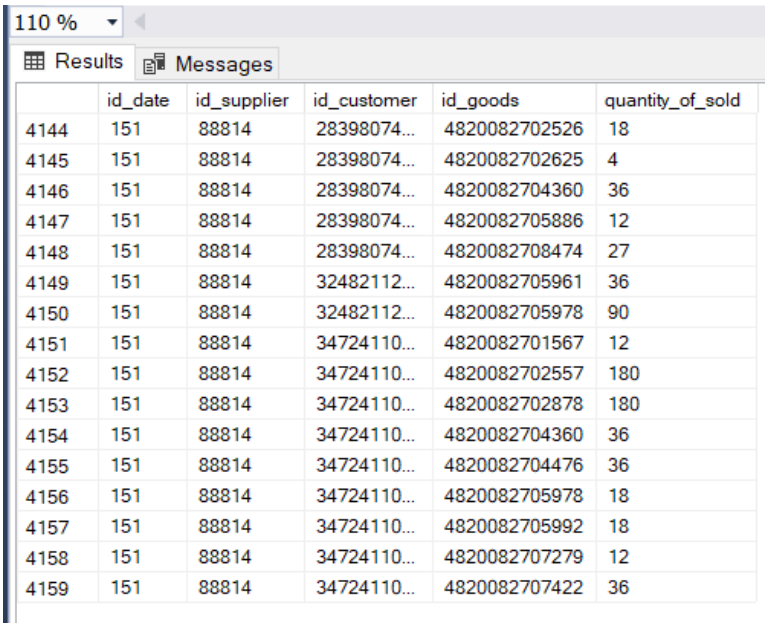
3.4 Дані для аналізу

Для дослідження були використані дані, які були надані підприємством «ТОВ ГРІН ФУД» [19]. Зважаючи на обмежену доступність великих обсягів даних від компаній, обмежмося інформацією за декілька місяців, що вже було достатньо для наших досліджень, використовуючи методи Data Mining. Первинні дані були надані у форматі xlsx та мали неструктурований вигляд. Тож

довелось вручну переструктурувати ці дані і створити SQL-код для імпорту інформації до операційної бази даних. Після цього було проведено обробку даних, яку детально описано в розділі про механізм вилучення, обробки та передачі даних вище.

Однією з найважливіших складових дослідження стали дані про продажі, які містять інформацію про товари, постачальників та клієнтів. Загалом, було опрацьовано 727 записів, що представляє собою значну кількість даних для подальшого аналізу та використання методів Data Mining.

Однак, для подальшого аналізу та застосування методів Data Mining, найбільш важливою є таблиця фактів із сховища даних. Вона містить значну кількість записів, а саме 4159, що робить її ключовою для наших досліджень. Приклад даних можна побачити на рис. 3.4.1.



110 % Results Messages

	id_date	id_supplier	id_customer	id_goods	quantity_of_sold
4144	151	88814	28398074...	4820082702526	18
4145	151	88814	28398074...	4820082702625	4
4146	151	88814	28398074...	4820082704360	36
4147	151	88814	28398074...	4820082705886	12
4148	151	88814	28398074...	4820082708474	27
4149	151	88814	32482112...	4820082705961	36
4150	151	88814	32482112...	4820082705978	90
4151	151	88814	34724110...	4820082701567	12
4152	151	88814	34724110...	4820082702557	180
4153	151	88814	34724110...	4820082702878	180
4154	151	88814	34724110...	4820082704360	36
4155	151	88814	34724110...	4820082704476	36
4156	151	88814	34724110...	4820082705978	18
4157	151	88814	34724110...	4820082705992	18
4158	151	88814	34724110...	4820082707279	12
4159	151	88814	34724110...	4820082707422	36

3.4.1 Результат запиту про дані з таблиці фактів

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Дослідження використання задач класифікації

4.1.1 Використання 1-Rule для класифікації. Алгоритм побудови елементарних правил (1-Rule) – простий алгоритм побудови правил для класифікації об’єкта, цей алгоритм будує правила за значенням тільки однієї незалежної змінної. Алгоритм по суті є пошуком змінної, яка дозволила б максимальною точністю класифікувати об’єкти. Для цього необхідно перевірити кожну змінну шляхом обчислення «помилки» – кількості об’єктів, що не задовольняють правилу з тих, що мають значення цієї змінної. У підсумку обирається змінна з найменшою помилкою.

1-Rule – це алгоритм машинного навчання, який зазвичай використовується для задач класифікації. Основна ідея 1-Rule полягає у тому, щоб використовувати одне правило для класифікації даних. Це правило може бути вибране згідно з певними статистичними метриками, такими як точність, чутливість, специфічність тощо.

Алгоритм 1-Rule складається з наступних кроків:

- Обчислити статистичні метрики для кожного можливого правила.
- Вибрати правило з найвищим значенням статистичної метрики.
- Використовувати це правило для класифікації нових даних.

Наприклад, якщо маємо набір даних з двох класів (наприклад, позитивний і негативний), то можна використовувати правило, яке класифікує всі дані, що мають значення більше або менше заданого порогового значення, як позитивні або негативні. При цьому, значення порогу може бути вибране згідно зі статистичними метриками.

Алгоритм 1-Rule може бути корисним у випадках, коли набір даних досить простий і не містить складних залежностей між ознаками. Однак, він може бути менш ефективним для складних наборів даних, де потрібна більш складна модель класифікації [20].

Розглянемо приклад, використовуючи наведене вище сховище даних та програмування на мові R. В ході аналізу цієї задачі було виділено два класи для подальшої класифікації: клас «Високий рівень продажів», який включає в себе дані з кількістю продажів, що перевищують середнє значення за весь період спостереження, та клас «Низький рівень продажів», в якому кількість продажів менше за середнє значення за весь період спостереження.

Залежна змінна у цій задачі – це кількість продажів, яка підлягає аналізу та класифікації. Незалежні змінні включають в себе інформацію про постачальників та клієнтів, яка може бути використана для побудови моделі класифікації та передбачення рівня продажів.

Першим етапом є імпорт необхідних бібліотек та встановлення з'єднання з базою даних, використовуючи бібліотеку «odbc» (рис. 4.1.1.1). Ця бібліотека надає зручний інтерфейс для підключення до різних джерел даних.

```r
# Завантаження бібліотек
library(odbc) # для підключення до сховища даних
library(dplyr) # для реалізації алгоритму
library(ggplot2) # для виведення графіку

# Підключення до бази даних
con <- dbConnect(odbc(),
                 Driver = "ODBC Driver 17 for SQL Server",
                 Server = "DESKTOP-5E8892J",
                 Database = "Warehouse",
                 UID = "sa",
                 PWD = "sa")
```

4.1.1.1 Завантаження бібліотек та підключення до сховища даних

Після отримання даних з таблиці «fact» сховища даних «Warehouse», перейдемо до обчислення середньої кількості продажів за весь період спостереження. Для цього спершу завантажили дані з таблиць «supplierDim» та «customerDim», щоб мати можливість ідентифікувати постачальників і клієнтів за їхніми ідентифікаторами. Наступним кроком було визначення обрізаного середнього, яке допомогло уникнути впливу викидів на обчислену величину (рис. 4.1.1.2).

Для обрізаного середнього було використано квантилі, де параметр «trim_percent» визначає відсоток видалення викидів з обох кінців розподілу даних. За допомогою цих квантилів були видалені екстремальні значення, що

допомогло отримати більш стабільну та репрезентативну середню кількість продажів. У нашому випадку обрізане середнє складає приблизно 74.70 одиниць, що є результатом аналізу даних з таблиці «fact».

```
# SQL запит запит для отримання даних
query <- "SELECT * FROM Warehouse.dbo.fact"

# Отримання даних з таблиці supplier та customer
supplier_data <- dbGetQuery(con, "SELECT id_supplier, supplier_name AS supplier_name FROM supplierDim")
customer_data <- dbGetQuery(con, "SELECT id_customer, customer_name AS customer_name FROM customerDim")

# Завантаження даних з бази даних
fact <- dbGetQuery(con, query)

# Визначте відсоток видалення викидів (наприклад, 10% з обох кінців)
trim_percent <- 10

# Знайдіть квантилі для видалення
lower_quantile <- trim_percent / 200
upper_quantile <- 1 - (trim_percent / 200)

# Обчислення обрізаного середнього
mean <- mean(fact$quantity_of_sold[fact$quantity_of_sold >= quantile(fact$quantity_of_sold, lower_quantile) & fact$quantity_
of_sold <= quantile(fact$quantity_of_sold, upper_quantile)])

data <- fact$quantity_of_sold[fact$quantity_of_sold >= quantile(fact$quantity_of_sold, lower_quantile) & fact$quantity_of_so
ld <= quantile(fact$quantity_of_sold, upper_quantile)]

mean
```

```
## [1] 74.70143
```

4.1.1.2 Обчислення середньої кількості продажів

Пояснимо детальніше, чому було прийнято рішення використовувати обрізане середнє значення для аналізу кількості продажів. Для цього спростимо це на прикладі гістограми розсіювання кількості продажів. На рис. 4.1.1.3 представлений графік на якому видно, що більшість значень концентрується на рівні до 500 одиниць, але при цьому існують викиди, які досягають значень, що перевищують 9000 одиниць. Якщо брати звичайне середнє значення (середнє арифметичне), то воно буде дуже вразливим до таких викидів, і в результаті переважний рівень продажів буде неправильно класифікуватися, як низький.

Саме тому було використано обрізане середнє, відкидаючи 10% найнижчих і 10% найвищих значень, щоб уникнути впливу цих викидів на показник. Такий підхід дозволяє отримати більш об’єктивне та стабільне значення середньої кількості продажів, яке коректно відображає переважний рівень продажів в наборі даних.

4.1.1.3 Діаграма розсіювання

Давайте докладніше розглянемо діаграму на рис. 4.1.1.4. Червоними лініями позначені верхня та нижня границя після відкидання 10% викидів, а зеленою – обрізане середнє значення. Тепер бачимо, що середнє значення виглядає значно більш об'єктивним і стабільним після відсіву викидів.

4.1.1.4 Діаграма розсіювання

В відрізку коду на рис. 4.1.15 було проведено класифікацію даних за певним критерієм. Класифікація полягає у визначенні рівня продажів для кожного запису в таблиці фактів. Порівнюємо кількість продажів кожного запису з середнім значенням.

Якщо кількість продажів більше за середнє значення, то цей запис класифікується як «High» (високий рівень продажів). В іншому випадку, якщо кількість продажів менше за середнє значення, запис класифікується як «Low» (низький рівень продажів).

Після класифікації додаємо новий стовпчик «level» до таблиці фактів, в якому зберігається результат класифікації для кожного запису (рис. 4.1.1.5). В результаті отримуємо таблицю, де кожен запис має вказаний рівень продажів: «High» або «Low».

```r
# Класифікація
n <- nrow(fact)
results <- character(n)
for (i in 1:n) {
  if (fact[i, 5] > mean) {
    results[i] <- "High"
  } else {
    results[i] <- "Low"
  }
}

# Додавання класифікації до даних
fact$level <- results

head(fact)
```

```
##   id_date id_supplier id_customer      id_goods quantity_of_sold level
## 1     129         333    19421419 4820082700621               6   Low
## 2     129         333    19421419 4820082702861             252  High
## 3     129         333    19421419 4820082705152              12   Low
## 4     129         333    19421419 4820082707446             126  High
## 5     129         333    19421419 4820082708108              48   Low
## 6     129         333    32974402 4820082705152              12   Low
```

4.1.1.5 Класифікація даних

Далі використовуємо бібліотеку «dplyr» для реалізації алгоритму 1-Rule. Цей алгоритм використовується для класифікації даних на основі найчастіших класів для кожного значення незалежної змінної.

Спершу було створено набір даних «data», в якому містяться дані з незалежною змінною «independent_variable» (у цьому випадку, це «id_supplier») та класовою змінною «class_variable» («level» рівень продажів).

Потім використовуємо функцію group_by() для групування даних за значенням незалежної змінної. Наступний крок – обчислення найчастіших класів для кожної групи. Для цього використовується функція summarise(), яка визначає найчастіший клас для кожної групи за допомогою функції table(), яка підраховує кількість входжень кожного класу.

Результат обчислення зберігається в змінній «rule», де для кожного значення незалежної змінної маємо визначений найчастіший клас «most_frequent_class» (рис. 4.1.1.6).

```
independent_variable <- fact$id_supplier
class_variable <- fact$level

data <- data.frame(independent_variable, class_variable)

# Реалізація алгоритму 1-Rule
rule <- data %>%
  group_by(independent_variable) %>% # групує дані за значенням незалежної змінної
  summarise(most_frequent_class = names(sort(-table(class_variable))[1])) # обчислює найчастіший клас для кожного значення н
езалежної змінно
```

4.1.1.6 Реалізація алгоритму

Після цього проводимо підрахунок кількості спостережень для кожного правила, використовуючи функцію inner_join(). Перш за все, об'єднуємо дані зі змінної «data» з даними змінної «rule» за допомогою спільного стовпця «independent_variable».

Після об'єднання даних використовуємо функцію group_by(), щоб згрупувати дані за значеннями незалежної змінної та найчастішого класу, який було обчислено на попередньому кроці.

Наступною операцією є використання функції summarise() для підрахунку кількості спостережень, які відповідають найчастішому класу для кожної групи. Використовуємо функцію sum(), щоб підрахувати кількість спостережень, де значення класової змінної «class_variable» дорівнює найчастішому класу «most_frequent_class» (рис. 4.1.1.7).

Результат підрахунку зберігається в змінній «rule_count», і він містить кількість спостережень для кожного правила (незалежна змінна та найчастіший клас) у форматі таблиці зі стовпцями «independent_variable», «most_frequent_class» та «count» (кількість спостережень).

```
# Підрахунок кількості спостережень для кожного правила
rule_count <- data %>%
  inner_join(rule, by = "independent_variable") %>%
  group_by(independent_variable, most_frequent_class) %>%
  summarise(count = sum(class_variable == most_frequent_class))
```

4.1.1.7 Підрахунок кількості спостережень для кожного правила

Розраховуємо відносну частоту кожного класу («High» або «Low») для кожного значення незалежної змінної на основі попередньо підрахованих кількостей спостережень для кожного правила. Для цього було використано функцію group_by(), щоб групувати дані за значеннями незалежної змінної.

Далі за допомогою функції mutate() було розраховано відносну частоту кожного класу для кожного значення незалежної змінної. Також було обчислено загальну кількість спостережень для кожного значення незалежної змінної.

Результати розрахунків зберігаються у змінній «class_prob», яка містить стовпці: «independent_variable», «most_frequent_class», «count» (кількість спостережень для найчастішого класу), «total_count» (загальна кількість спостережень для даної незалежної змінної), та «class_probability» (відносна частота кожного класу в форматі відсотків).

Далі об'єднуємо результати з таблицею «supplier_data», щоб включити імена постачальників у вивід, та перейменовуємо стовпці для зручності аналізу. В результаті отримуємо таблицю «class_prob_supplier», яка містить інформацію про відносну частоту класів («High» або «Low») для кожного постачальника, а також загальну кількість спостережень та відсоткову відносну частоту кожного класу для кожного постачальника (рис. 4.1.1.8).

```
# Підрахунок відносної частоти кожного класу для кожного значення незалежної змінної
class_prob <- rule_count %>%
  group_by(independent_variable) %>%
  mutate(total_count = sum(data$independent_variable == independent_variable),
         class_probability = paste(round(count / total_count * 100, 2), "%"))

# Включення імен постачальників у результати за id_supplier
class_prob_supplier <- class_prob %>%
  left_join(supplier_data, by = c("independent_variable" = "id_supplier")) %>%
  select(supplier_name, independent_variable, most_frequent_class, count, total_count, class_probability)

# Перейменування стовпців на виводі
class_prob_supplier <- class_prob_supplier %>%
  rename(
    supplier = supplier_name,
    id = independent_variable,
    level = most_frequent_class,
    cout = count,
    total = total_count,
    probability = class_probability
  )

print(class_prob_supplier)
```

4.1.1.8 Підрахунок відносної частоти та налаштування виводу

За результатами аналізу на рис. 4.1.1.9 видно, що товари деяких постачальників, таких як «JAMES FINLAY (ME) DMCC» та «CEYLON TEA LAND (PVT) LTD» мають високий рівень продажу і класифікуються як «High» з відсотковою відносною частотою 100% та 74% відповідно. У той час як інші постачальники, наприклад, «ZHANGZHOU CHANGLU FOOD CO., LTD» та «HUNAN TEA GROUP COMPANY LIMITED», класифікуються як «Low» з відсотковою відносною частотою від 64.66% до 66.47%.

```
## # A tibble: 5 × 6
## # Groups:   id [5]
##   supplier                                id level  cout total probability
##   <chr>                                <dbl> <chr> <int> <int> <chr>
## 1 "ZHANGZHOU CHANGLU FOOD CO., LTD "     333 Low     333   501 66.47 %
## 2 "JAMES FINLAY (ME) DMCC"               558 High     60    60 100 %
## 3 "CEYLON TEA LAND (PVT) LTD"            560 High     40    54 74.07 %
## 4 "A.F. JONES EXPORTERS CEYLON (PVT) LTD" 1135 Low    1557  2101 74.11 %
## 5 "HUNAN TEA GROUP COMPANY LIMITED"     88814 Low     933  1443 64.66 %
```

4.1.1.9 Результати для постачальників

З аналізу клієнтів на рис. 4.1.1.10 видно, що деякі, такі як «КАРНАУХОВА НАТАЛІЯ ОЛЕГІВНА ФОП» та «ПОЛЯРУШ ОЛЕКСАНДР ГРИГОРОВИЧ ФОП» мають велику кількість скуповувань товарів і класифікуються як «High» з відсотковою відносною частотою від 70.59% до 100%. У той час як інші клієнти, наприклад, «МОНОМАХ ПАТ» та «ЧАЙНИЙ ДІМ ТОВ» мають меншу кількість спостережень та скуповувань, тому класифікуються як «Low» з відсотковою відносною частотою від 55.96% до 94.74%.

```
## # A tibble: 12 × 6
## # Groups:   id [12]
##    customer                               id level  cout total probability
##    <chr>                               <dbl> <chr> <int> <int> <chr>
##  1 МОНОМАХ ПАТ                      19421419 Low     291   520 55.96 %
##  2 ЧАЙНИЙ ДІМ ТОВ                   32974402 Low     144   152 94.74 %
##  3 ТІТРЕЙД ТОВ                      34408235 Low     476   616 77.27 %
##  4 РОЗЕТКА. УА ТОВ                  37193071 Low     451   591 76.31 %
##  5 ЮНІОН ТРЕЙД 2013 ТОВ             38467650 Low     160   246 65.04 %
##  6 ЮНІТІ 90 ТОВ                     44606332 Low     440   600 73.33 %
##  7 ЗІНЧИК СВІТЛАНА МИКОЛАЇВНА ФОП  2301118107 Low     170   300 56.67 %
##  8 ПОЛЯРУШ ОЛЕКСАНДР ГРИГОРОВИЧ ФОП 2498802470 High    120   170 70.59 %
##  9 ПОЛІЩУК СЕРГІЙ ВАЛЕНТИНОВИЧ ФОП  2830117633 High     80   130 61.54 %
## 10 КАРНАУХОВА НАТАЛІЯ ОЛЕГІВНА ФОП  2839807409 Low     152   152 100 %
## 11 ЕПІК ОЛЬГА ЛЕОНІДІВНА ФОП        3248211238 High    174   208 83.65 %
## 12 УСТИМЕНКО ОЛЬГА МАКСИМІВНА ФОП   3472411041 Low     419   474 88.4 %
```

4.1.1.10 Результати для клієнтів

Метод 1-Rule, хоч і простий у реалізації, має свої обмеження та недоліки, які роблять його менш ефективним у деяких ситуаціях. Одним із головних

обмежень є використання лише одного найчастішого класу для класифікації кожного значення незалежної змінної. Це може призвести до втрати інформації про можливі варіації в класах та недостатньо точних результатів.

Крім того, метод 1-Rule не враховує складні зв'язки між змінними та не робить врахування різних факторів, що можуть впливати на класифікацію. В результаті, цей метод може бути менш ефективним для задач, де потрібно враховувати багато змінних та їх взаємодію.

Таким чином, важливо розуміти, що метод 1-Rule може бути корисним для простих задач класифікації, але в більш складних сценаріях варто розглядати більш потужні та точні методи машинного навчання для досягнення кращих результатів.

4.1.2 Використання методу наївного Байеса. Наївний байесівський класифікатор один з найпростіших з алгоритмів класифікації. Тим не менш, досить часто він працює краще за більш складні алгоритми.

Метод наївного Байеса – це алгоритм машинного навчання, який використовується для класифікації даних, оснований на теорії ймовірності та правилах Байеса. Він широко застосовується в різних галузях, таких як обробка природних мов, аналіз даних, рекомендаційні системи та інші.

Основна ідея методу наївного Байеса полягає в тому, щоб використовувати теорію ймовірності для визначення того, до якого класу належить новий набір даних. Метод наївного Байеса передбачає, що кожна ознака в наборі даних незалежна від інших, що дозволяє легко розрахувати ймовірності.

Для застосування методу наївного Байеса до задачі класифікації, спочатку необхідно побудувати модель, використовуючи тренувальний набір даних. Модель будується шляхом розрахунку ймовірності для кожного класу та кожної ознаки в наборі даних. Після цього, для класифікації нового набору даних, метод наївного Байеса використовує цій моделі, щоб обчислити ймовірність того, що набір даних належить до кожного класу, та вибрати клас з найвищою ймовірністю [21].

Для продовження дослідження наших даних методом наївного Байеса, необхідно встановити ще одну бібліотеку «e1071» (рис. 4.1.2.1).

```
# Завантаження бібліотек
# Завантаження бібліотек
library(odbc)
library(dplyr)
library(e1071) # для використання фуекції naiveBayes
```

4.1.2.1 Підключення нової бібліотеки

Тож було створено модель за допомогою алгоритму наївного Байеса (рис. 4.1.2.2). Модель навчена за залежністю рівня «level» від ідентифікаторів постачальника «id_supplier» та клієнта «id_customer». Це важливий крок, який дозволяє нам передбачати рівень продажів на підставі даних про постачальників та клієнтів.

```
# Навчити модель за допомогою алгоритму Naïve Bayes
model <- naiveBayes(level ~ id_supplier + id_customer, data = fact)
model
```

4.1.2.2 Навчання моделі

Отримані результати моделі наївного Байеса включають в себе наступну інформацію, продемонстровану на рис. 4.1.2.3:

A-priori probabilities: Ця частина вказує на ймовірності належності до кожного з класів «High» і «Low». Наприклад, ймовірність належності до класу «High» становить близько 31.79%, тоді як до класу «Low» - приблизно 68.21%.

Conditional probabilities: Ця частина включає умовні ймовірності належності до класів «High» і «Low» в залежності від значень ідентифікаторів постачальника і клієнта. Ці ймовірності дозволяють моделі приймати рішення на підставі вхідних даних. Наприклад, для класу «High» ідентифікатор постачальника має середнє значення навколо 34,814.22, під час коли для класу «Low» це значення близько 29,872.89. Аналогічно, для ідентифікатора клієнта, середні значення різняться між класами «High» і «Low».

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##      High       Low
## 0.3178649 0.6821351
##
## Conditional probabilities:
##       id_supplier
## Y          [,1]     [,2]
##   High 34814.22 42812.63
##   Low  29872.89 41267.70
##
##       id_customer
## Y           [,1]       [,2]
##   High 1215403281 1409783381
##   Low   960169253 1409793918
```

4.1.2.3 Виведення моделі

Далі був створений новий набір даних для прогнозування, який містить всі можливі комбінації значень «id_supplier» та «id_customer» з таблиці фактів (рис. 4.1.2.4). Для цього було використано функцію expand.grid(), яка з'єднує унікальні значення обох стовпців. Отриманий набір даних «new_data«»» представляє собою всі можливі поєднання постачальників і клієнтів з наявних даних.

```
# Створення нового набору даних для прогнозування
new_data <- expand.grid(id_supplier = unique(fact$id_supplier), id_customer = unique(fact$id_customer))
```

4.1.2.4 Всі можливі комбінації постачальників та клієнтів

Після створення «new_data» була використана навчена модель наївного Байеса для прогнозування класифікації для цих нових даних. Використовуючи функцію predict() для кожної пари постачальника і клієнта окремо, отримуючи прогнози для класів "High" і "Low". Після цього додано результати прогнозу до таблиці «result_data», в якій вказані постачальник, клієнт, ймовірність класу «High» та ймовірність класу «Low» для кожної пари (рис. 4.1.2.5).

Ці результати можуть бути корисними для подальшого аналізу та прийняття рішень у сфері бізнесу.

```
# Створення фрейму даних для результатів прогнозування
result_data <- data.frame(supplier = integer(0), customer = integer(0), `Class H` = numeric(0), `Class L` = numeric(0))

# Прогноз класу для кожної пари постачальника і клієнта окремо
for (i in 1:nrow(new_data)) {
  current_supplier <- new_data$id_supplier[i]
  current_customer <- new_data$id_customer[i]

  # Створення окремого набору даних для прогнозування
  current_data <- data.frame(id_supplier = current_supplier, id_customer = current_customer)

  # Прогноз для поточної пари
  current_prediction <- predict(model, newdata = current_data, type = "raw") * 100


  # Додавання результатів до фрейму даних result_data
  result_data <- rbind(result_data, c(current_supplier, current_customer, current_prediction[, "High"], current_prediction[,
"Low"]))
}
```

4.1.2.5 Передбачення для кожної пари

Виведення результатів продемонстровано в додатку Б.

Використання методу наївного Байеса дало нам змогу докладно проаналізувати ймовірність належності кожної пари до певних класів, що може бути корисним для прийняття бізнес-рішень. За допомогою цього методу можемо виділити ті постачальників і клієнтів, які мають високу ймовірність належності до класу «High», і навпаки, ті, які мають високу ймовірність належності до класу «Low».

Метод Байеса корисний в аналізі даних, оскільки він дозволяє враховувати ймовірності та залежності між різними змінними. В даному випадку він допомагає прогнозувати класифікацію на основі історичних даних та встановлювати, які постачальники і клієнти мають високий рівень продажів і низький рівень продажів.

Застосування методу Байеса допомагає нам визначити, які товари постачальників найкраще продавати конкретним клієнтам для отримання високих обсягів продажів.

4.2 Дослідження використання методу асоціативних правил

Метод асоціативних правил – це алгоритм машинного навчання, який використовується для виявлення корисних залежностей між різними елементами в наборі даних. Він широко використовується в бізнесі та економіці для аналізу

покупок, а також в інших галузях, де необхідно зрозуміти залежності між різними елементами.

Основна ідея методу асоціативних правил полягає в тому, щоб знайти зв'язки між різними елементами в наборі даних шляхом виявлення комбінацій які часто відбуваються. Наприклад, якщо користувач купує хліб, то йому можуть також потрібні молоко та яйця. Ці зв'язки можна використовувати для прогнозування покупок та підвищення продажів.

Для застосування методу асоціативних правил до задачі аналізу даних, спочатку необхідно побудувати модель, використовуючи тренувальний набір даних. Модель будується шляхом знаходження комбінацій елементів які часто відбуваються, які входять до складу різних транзакцій. Для цього використовуються метрики, такі як підтримка та достовірність.

Після побудови моделі, можна використовувати її для прогнозування майбутніх покупок або для рекомендацій продуктів, які часто купують разом. Наприклад, якщо багато користувачів купують хліб та молоко разом, то можна рекомендувати молоко користувачам, які купують хліб.

Метод асоціативних правил має кілька переваг, таких як можливість виявлення складних зв'язків між різними елементами та високою швидкість обробки великих обсягів даних. Крім того, він є простим у використанні та може бути застосований до різних галузей, таких як банківська справа, маркетинг, медицина та інші.

Проте, метод асоціативних правил також має свої недоліки. Наприклад, він може давати зайво велику кількість результатів, які можуть бути непотрібними або не корисними. Крім того, він може пропустити важливі зв'язки, які не зустрічаються часто в даних [22].

Першим ділом підключимо нові бібліотеки для використання методу асоціативних правил, а саме «arules» та «arulesViz», як показано на рис. 4.2.1.

```
# Завантаження бібліотек
library(odbc)
library(dplyr)
library(arules) # для використання алгоритму Apriori
library(arulesViz) # для візуалізації асоціативних правил
```

4.2.1 Підключення нових бібліотек

Після чого створюється приклад вхідних даних для навчання моделі асоціативних правил (рис. 4.2.2). Для цього дані з таблиці «fact» перетворюються у відповідний формат, включаючи ідентифікатори постачальників, клієнтів, товарів та рівень, який представляє важливу інформацію про відносини між постачальниками та клієнтами.

```
# Створення прикладу вхідних даних для навчання моделі
data <- data.frame(
  fact.id_supplier = as.character(fact$id_supplier),
  fact.id_customer = as.character(fact$id_customer),
  fact.id_goods = as.character(fact$id_goods),
  fact.level = fact$level
)
```

4.2.2 Створення вхідних даних для навчальної моделі

Після підготовки даних використовується алгоритм Apriori для пошуку корисних асоціацій між продуктами (рис. 4.2.3). У параметрах алгоритму задається підтримка «supp» та достовірність «conf» асоціативних правил. Підтримка визначає, яка частка правил у вибірці містить даний набір айтемів, тоді як достовірність вказує на ймовірність того, що, якщо правило містить A, то воно також містить B.

Наприклад, якщо підтримка встановлена на 0.1, це означає, що лише правила, які мають підтримку 10% або більше, будуть розглядатися. А достовірність 0.5 вказує, що лише правила з достовірністю 50% або більше будуть враховані.

У результаті виконання алгоритму Apriori генерується набір асоціативних правил для подальшого аналізу.

```
# Використання алгоритму Apriori для пошуку корисних асоціацій між продуктами
rules <- apriori(data, parameter = list(supp = 0.1, conf = 0.5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 415
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[158 item(s), 4159 transaction(s)] done [0.00s].
## sorting and recoding items ... [10 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [9 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

4.2.3 Використання алгоритму Apriori

Далі виводяться знайдені асоціативні правила за допомогою функції inspect() (рис. 4.2.4). Кожне правило включає ліву (lhs) і праву (rhs) частини, показники підтримки (support), достовірності (confidence), покриття (coverage), показник підняття (lift), та кількість входжень (count).

Наприклад, перше правило не має лівої частини (пуста множина) і вказує на те, що «fact.id_supplier» зі значенням 1135 часто входить у транзакції. Його підтримка становить близько 50.52%, і він має достовірність 50.52%.

Друге правило вказує, що «fact.level» зі значенням «Low» часто входить у транзакції з підтримкою приблизно 68.21% та достовірністю 68.21%.

Третє правило вказує на асоціацію між ідентифікатором клієнта «fact.id_customer» зі значенням 3472411041 та рівнем «Low». Ця асоціація має підтримку близько 10.07% та достовірність приблизно 88.40%. В контексті аналізу даних, це означає, що клієнти з ідентифікатором 3472411041 часто здійснюють покупки з рівнем «Low».

```
# Виведення знайдених асоціативних правил
inspect(rules)
```

```
##     lhs                                rhs                     support
## [1] {}                              => {fact.id_supplier=1135} 0.5051695
## [2] {}                              => {fact.level=Low}        0.6821351
## [3] {fact.id_customer=3472411041} => {fact.level=Low}        0.1007454
## [4] {fact.id_customer=37193071}   => {fact.level=Low}        0.1084395
## [5] {fact.id_customer=44606332}   => {fact.level=Low}        0.1057947
## [6] {fact.id_customer=34408235}   => {fact.level=Low}        0.1144506
## [7] {fact.id_supplier=88814}      => {fact.level=Low}        0.2243328
## [8] {fact.id_supplier=1135}       => {fact.level=Low}        0.3743688
## [9] {fact.level=Low}              => {fact.id_supplier=1135} 0.3743688
##     confidence coverage  lift      count
## [1] 0.5051695  1.0000000 1.0000000 2101
## [2] 0.6821351  1.0000000 1.0000000 2837
## [3] 0.8839662  0.1139697 1.2958814  419
## [4] 0.7631134  0.1421015 1.1187129  451
## [5] 0.7333333  0.1442654 1.0750558  440
## [6] 0.7727273  0.1481125 1.1328067  476
## [7] 0.6465696  0.3469584 0.9478615  933
## [8] 0.7410757  0.5051695 1.0864060 1557
## [9] 0.5488192  0.6821351 1.0864060 1557
```

4.2.3 Виведення асоціативних правил

Після візуалізуємо деякі асоціативні правила, які містять умову «fact.level=Low». Для цього скористалися функцією subset(), яка дозволила нам відфільтрувати правила зі складною лівою стороною, що включають цю конкретну умову (рис. 4.2.4).

Перше правило в результаті виводу показує загальну асоціацію між будь-якими елементами та «fact.level=Low». Це правило має підтримку близько 68.21% і достовірність 68.21%, що вказує на те, що рівень «Low» відображається в більшості транзакцій.

Слідуючі правила виводять асоціації, де умова «fact.level=Low» поєднана з конкретними ідентифікаторами постачальників та клієнтів. Наприклад, правило з «fact.id_supplier=1135» має підтримку близько 37.44% та достовірність 74.11%, що вказує на те, що постачальник з ідентифікатором 1135 часто постачає продукти з рівнем «Low» клієнтам.

Ця фільтрація правил дозволяє нам краще зрозуміти асоціації між рівнем продукту «Low» та конкретними постачальниками та клієнтами, що може бути корисним для оптимізації продажів та маркетингових стратегій.

```
# Фільтрація правил
rulesDem <- subset(rules, subset = rhs %in% "fact.level=Low")
inspect(head(rulesDem, n = 10, by = "support"))
```

```
##     lhs                              rhs              support   confidence
## [1] {}                            => {fact.level=Low} 0.6821351 0.6821351
## [2] {fact.id_supplier=1135}       => {fact.level=Low} 0.3743688 0.7410757
## [3] {fact.id_supplier=88814}      => {fact.level=Low} 0.2243328 0.6465696
## [4] {fact.id_customer=34408235}   => {fact.level=Low} 0.1144506 0.7727273
## [5] {fact.id_customer=37193071}   => {fact.level=Low} 0.1084395 0.7631134
## [6] {fact.id_customer=44606332}   => {fact.level=Low} 0.1057947 0.7333333
## [7] {fact.id_customer=3472411041} => {fact.level=Low} 0.1007454 0.8839662
##     coverage  lift      count
## [1] 1.0000000 1.0000000 2837
## [2] 0.5051695 1.0864060 1557
## [3] 0.3469584 0.9478615  933
## [4] 0.1481125 1.1328067  476
## [5] 0.1421015 1.1187129  451
## [6] 0.1442654 1.0750558  440
## [7] 0.1139697 1.2958814  419
```

4.2.4 Відфільтровані правила

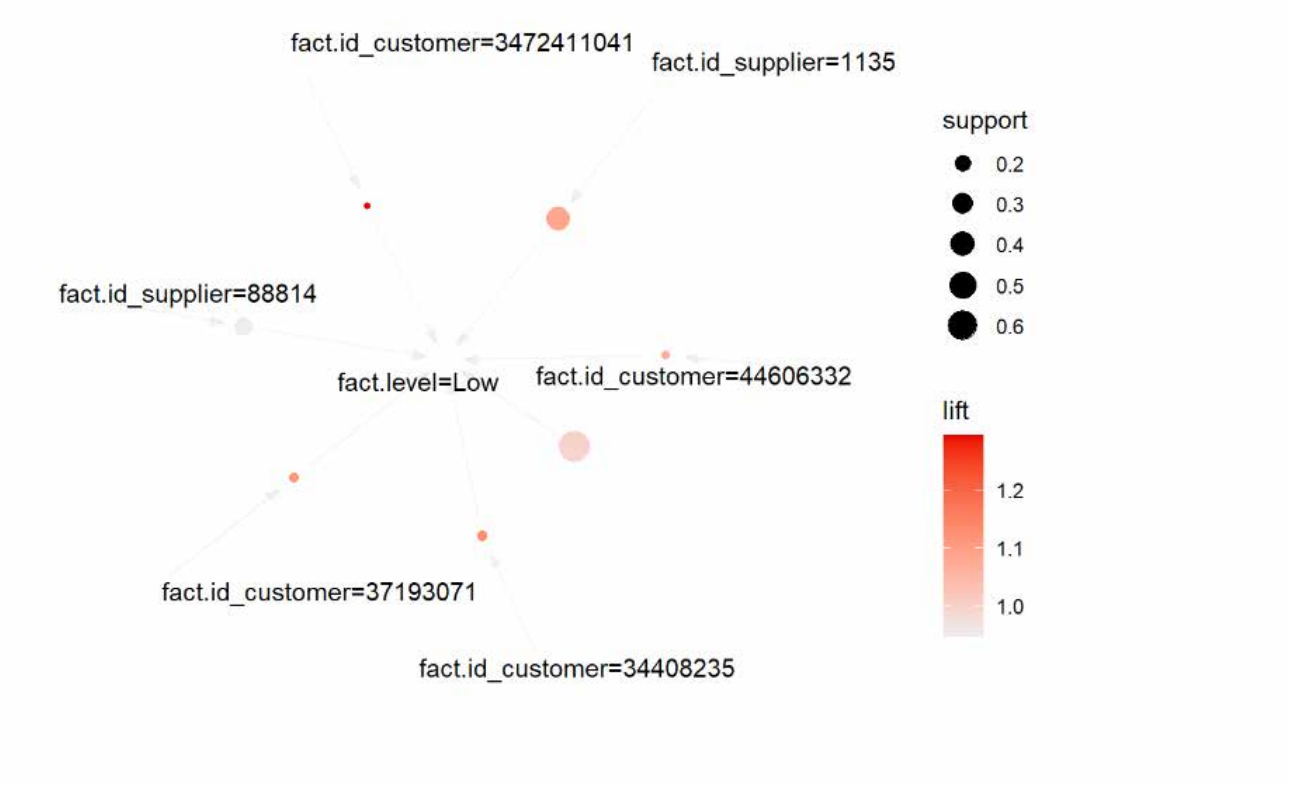
Побудуємо граф який представляє собою схему мережі асоціативних правил, яка виникла під час аналізу даних про постачальників, клієнтів та рівнів продуктів. Граф відображає важливі зв'язки та асоціації між цими об'єктами, що допомагають в розумінні взаємодій в даних.

На графу рис. 4.2.5 можна побачити вузли, які відповідають асоціативним правилам. Кожен вузол містить інформацію про це правило, зокрема його ліву та праву сторону, підтримку, достовірність, покриття, підвищення та кількість відповідних асоціацій. Ця інформація дозволяє нам зрозуміти, які асоціації були знайдені та наскільки вони сильні і важливі.

Представлений граф показує важливі асоціації та зв'язки між різними факторами у наборі даних. Нульовий вузол {} => {fact.level=Low} вказує на загальний зв'язок між усіма об'єктами і рівнем "Low", з високою підтримкою та достовірністю, свідчачи про поширену наявність рівня "Low" в транзакціях.

Подальші вузли представляють асоціації між конкретними постачальниками та клієнтами з рівнем «Low». Вони підтверджують, що певні постачальники та клієнти мають сильний зв'язок з продуктами «Low». Достовірність близько 74% для постачальників і близько 73-88% для клієнтів свідчить про високу вірогідність цих асоціацій.

Цей графік надає корисний інсайт щодо взаємодій між об’єктами у вибірці та допомагає виділити ключові асоціації, які можуть бути використані для стратегічного планування та оптимізації бізнесу.



4.2.5 Граф візуалізації асоціативних правил

Отже, використання методу асоціативних правил дозволяє виявляти корисні зв’язки та асоціації між різними елементами в даних, що є важливим інструментом для бізнесу та інших галузей. Цей метод сприяє підвищенню ефективності прийняття рішень, оптимізації стратегій та прогнозуванню майбутніх подій, що робить його корисним для підвищення продажів, рекомендаційних систем, маркетингових досліджень та багатьох інших сфер.

4.3 Дослідження використання алгоритмів кластеризації

Кластеризація — це автоматичне розбиття елементів деякої множини на групи в залежності від їх подібності. Елементами множини може бути все, що завгодно, наприклад, дані або вектори характеристик. Власне групи прийнято називати кластерами.

Кластеризація (об’єднання в групи схожих об’єктів) є однією із фундаментальних задач Data Mining. Список прикладних областей, де вона

застосовується, широкий: сегментація зображень, маркетинг, боротьба з шахрайством, прогнозування, аналіз текстів тощо. Задачу кластеризації в тому чи іншому вигляді формулювали в таких наукових напрямках, як статистика, розпізнавання образів, оптимізація, машинне навчання. Звідси різноманіття синонімів поняттю кластер — клас, таксон, згущення.

Крім того, кластеризація є розділом сучасної теоретичної інформатики, що бурхливо розвивається, і в цій області можна отримати ряд цікавих дослідницьких результатів.

Кластеризація розбиває множину об’єктів на групи, які визначаються лише її результатом. Класифікація відносить кожен об’єкт до однієї із заздалегідь визначених груп.

Кластеризація включає в себе наступні етапи:

- Виділення характеристик
- Визначення метрики
- Розбиття об’єктів на групи
- Представлення результатів

Алгоритми кластеризації – це методи машинного навчання, які дозволяють групувати схожі об’єкти разом в кластери на основі їхніх властивостей та характеристик. Вони знаходять широке застосування в різних галузях, таких як біоінформатика, медицина, соціологія, фінанси, маркетинг та інші [23].

Одним з основних застосувань алгоритмів кластеризації є сегментація ринку. Наприклад, в маркетингу вони дозволяють визначити групи споживачів з подібними потребами та поведінкою, що дає можливість спрямовувати рекламні кампанії на ці групи та підвищувати ефективність маркетингових стратегій.

Також алгоритми кластеризації використовуються в медицині для класифікації хвороб та пацієнтів за симптомами та іншими ознаками. Вони допомагають виявляти залежності між хворобами та лікуваннями, а також забезпечувати більш ефективну діагностику та лікування.

Для розв’язання задачі кластеризації можна використовувати різні алгоритми, такі як k-середніх, ієрархічну кластеризацію, агломеративну кластеризацію та інші. Вибір конкретного алгоритму залежить від характеру даних та задачі, яку необхідно вирішити.

Алгоритм кластеризації k-середніх – це один з найпоширеніших алгоритмів кластеризації, який дозволяє групувати об’єкти на основі їхньої відстані до середини кожного кластера.

Алгоритм k-середніх працює наступним чином:

- Обирається кількість кластерів, яку необхідно сформувати.
- Випадковим чином обираються початкові центри кожного кластера.
- Кожен об’єкт призначається до того кластера, до якого його відстань до центру найближчого кластера є найменшою.
- Обчислюється новий центр кожного кластера шляхом вирахування середнього значення всіх об’єктів, які належать до цього кластера.
- Кроки 3-4 повторюються до тих пір, поки кластери не стабілізуються.

Отже, за допомогою алгоритму k-середніх можна групувати об’єкти в кластери на основі їхньої відстані до центру кожного кластера. Цей алгоритм має кілька переваг, таких як простота та ефективність, що дозволяє його використовувати в багатьох галузях, таких як медицина, фінанси, маркетинг та інші [24].

Першим чином імпортуємо нову бібліотеку «stats», за допомогою такою якої і буде проведено кластеризацію. Після чого створюється приклад вхідних даних для навчання моделі, в якому містяться дві змінні: кількість проданих товарів «quantity_of_sold» і рівень «level» (рис. 4.3.1). Ці дані можуть бути використані для розв’язання завдань аналізу та прогнозування, де залежність між кількістю продажів і рівнем товарів може грати важливу роль.

```r
# Завантаження бібліотек
library(odbc)
library(dplyr)
library(stats) # для кластеризації методом k-середніх
```

```r
# Створення прикладу вхідних даних для навчання моделі
data <- data.frame(fact$quantity_of_sold, fact$level)
```

4.3.1 Підключення бібліотеки та формування набору даних

Після чого створюється вектор «k.values» з числовими значеннями від 1 до 10 включно (рис. 4.3.2). Цей вектор визначає можливу кількість кластерів, яку буде розглядати алгоритм k-середніх під час процесу кластеризації. Від 1 до 10 кластерів охоплюють різні можливі сценарії кількості кластерів, ініціюючи процес вибору оптимальної кількості кластерів за допомогою методу ліктя.

```r
# Від 1 до 10 кластерів
k.values <- 1:10
```

4.3.2 Вектор кількості кластерів

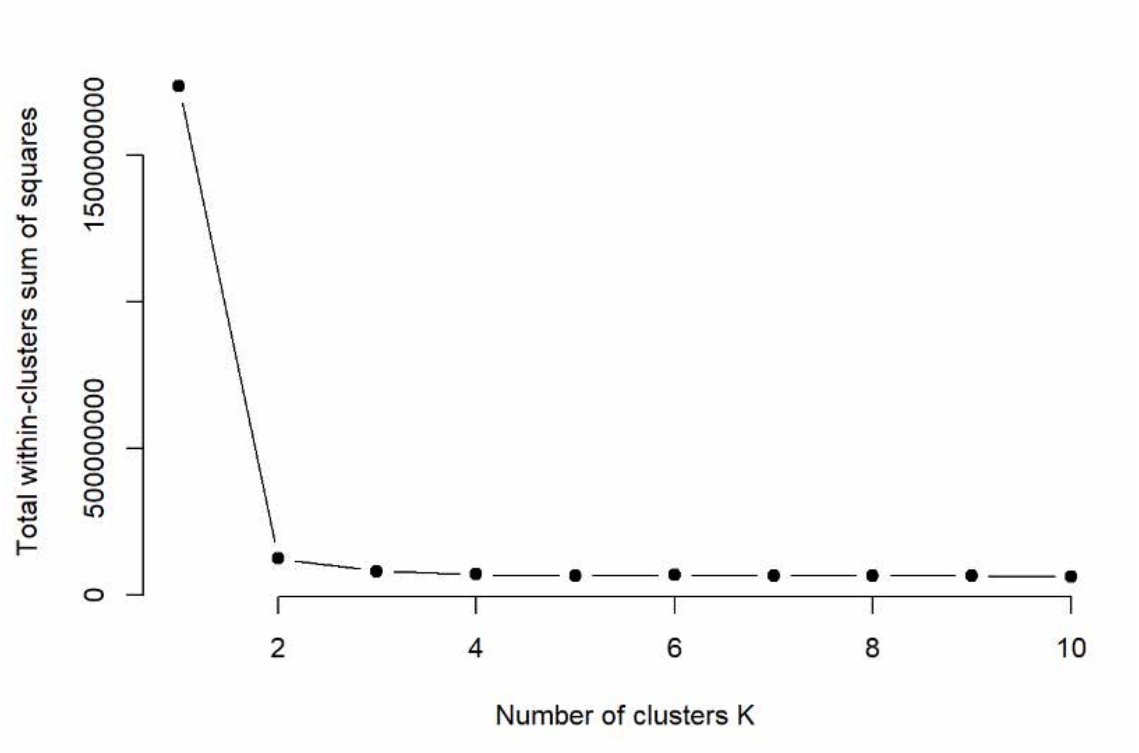
Далі створюється вектор «wss_values», який буде містити значення внутрішньокластерної дисперсії (значення розкиду) для кожної кількості кластерів. Для цього використовується цикл, який проходить по кожній кількості кластерів у векторі «k.values». Для кожної кількості кластерів проводиться кластеризація методом k-середніх (функція kmeans()) з центрами, випадково обраними з вибірки. Результат кластеризації містить внутрішньокластерну дисперсію, яка зберігається в векторі «wss_values» під відповідним індексом (рис. 4.3.4).

```r
# Вектор для збереження внутрішньокластерної дисперсії
wss_values <- vector(length = length(k.values))

# Розрахунок внутрішньокластерної дисперсії для кожної кількості кластерів
for(i in k.values) {
  kmeans_result <- kmeans(data, centers = i)
  wss_values[i] <- kmeans_result$tot.withinss
}
```

4.3.4 Розрахунок внутрішньокластерної дисперсії

Візуалізація змінної внутрішньокластерної дисперсії залежно від кількості кластерів. Змінні «k.values» та «wss_values» виступають в якості вісей графіку. Продемонстровано на рис. 4.3.5.



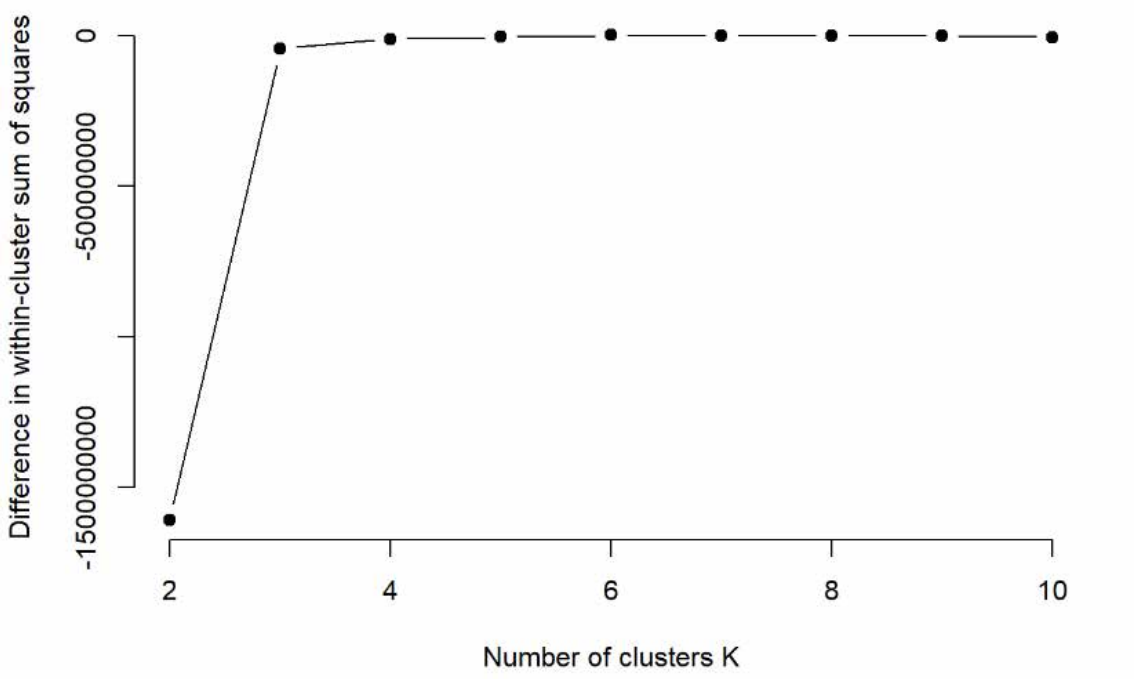
4.3.5 Внутрішньокластерна дисперсія

Використовується метод ліктя для визначення оптимальної кількості кластерів. Для цього знаходиться різниця між внутрішньокластерною дисперсією для кожної кількості кластерів (вектор «dist») та попередньою. Для знаходження вектору «dist» використовується різниця між елементами вектору «wss_values», що знаходяться на відстані 1 один від одного. Код для знаходження вектору «dist» має наступний вигляд як на рис. 4.3.6.

```
# Використання методу "ліктя" для визначення оптимальної кількості кластерів
# За допомогою цього методу ми шукаємо точку на графіку, де зміна внутрішньокластерної дисперсії
# зменшується значно менше, ніж при більшій кількості кластерів
dist <- wss_values[2:length(wss_values)] - wss_values[1:(length(wss_values)-1)]
```

4.3.6 Визначення оптимальної кількості кластерів

Далі, виводиться графік залежності різниці внутрішньокластерної дисперсії від кількості кластерів, який наведений на рис. 4.3.7. Перший елемент вектору «k.values» був вилучений, тому що для першої кількості кластерів розрахунок внутрішньокластерної дисперсії не має сенсу.



4.3.7 Різниця внутрішньокластерної дисперсії

На графіку можна побачити «лікоть», який відповідає точці, де зміна внутрішньокластерної дисперсії зменшується значно менше, ніж при більшій кількості кластерів. Цю точку можна знайти за допомогою функції which.max(), яка знаходить індекс максимального елемента вектору «dist». Індекс потрібно збільшити на 1, так як розглядаємо різницю внутрішньокластерної дисперсії для кожної кількості кластерів, починаючи з двох.

У наступному коді рис. 4.3.8 було використано функцію which.max(), яка повертає індекс максимального значення у векторі «dist». Оскільки вектор складається з різниць між внутрішньокластерною дисперсією для кожної кількості кластерів, індекс максимального значення у ньому буде відповідати точці на графіку, де зміна внутрішньокластерної дисперсії зменшується значно менше, ніж при більшій кількості кластерів. Оскільки було відкидано перший елемент вектору, додаємо 1 до результату функції which.max(), щоб отримати оптимальну кількість кластерів. У даному випадку оптимальна кількість це 6.

```
# Знаходження оптимальної кількості кластерів за допомогою методу "ліктя"
# За замовчуванням, mi - це точка на графіку, де зміна внутрішньокластерної дисперсії зменшується значно менше, ніж при більшій кількості кластерів
mi <- which.max(dist) + 1

# Результат
cat("Optimal number of clusters:", mi)
```

```
## Optimal number of clusters: 6
```

4.3.8 Знаходження оптимальної кількості кластерів

Наступний код виконує кластеризацію методом k-середніх за допомогою оптимальної кількості кластерів, рис. 4.3.9, що була знайдена раніше. У цьому коді було використано функцію kmeans(), щоб виконати кластеризацію методом k-середніх. Параметр «data» містить дані, які будуть кластеризовані, а параметр «centers» визначає кількість кластерів.

```
# Кластеризація методом k-середніх
kmeans_result <- kmeans(data, centers=mi)
```

4.3.9 Кластеризація методом k-середніх

Отримані результати вказують на результати кластеризації даних методом k-середніх з 6 кластерами (рис. 4.3.10). Кожен кластер має свої центроїди, які представлені у вигляді середніх значень ознак для об’єктів цього кластера.

За цими результатами видно, що кожен кластер має свої характеристики:

Перший кластер має центр з ознаками:

- Середня кількість проданих товарів (fact.quantity_of_sold) приблизно 94.69.
- Рівень (fact.level) приблизно 0.64.

Другий кластер має центр з ознаками:

- Середня кількість проданих товарів приблизно 36.67.
- Рівень 0.0.

Третій кластер має центр з ознаками:

- Середня кількість проданих товарів приблизно 10.68.
- Рівень 0.0.

Четвертий кластер має центр з ознаками:

- Дуже велика середня кількість проданих товарів, приблизно 7440.
- Рівень 1.0.

П'ятий кластер має центр з ознаками:

- Значення ознаки «fact.level» 1.0, що вказує на високий рівень.
- Середня кількість проданих товарів приблизно 452.45.

Шостий кластер має центр з ознаками:

- Значення ознаки «fact.level» 1.0.
- Середня кількість проданих товарів приблизно 212.95.

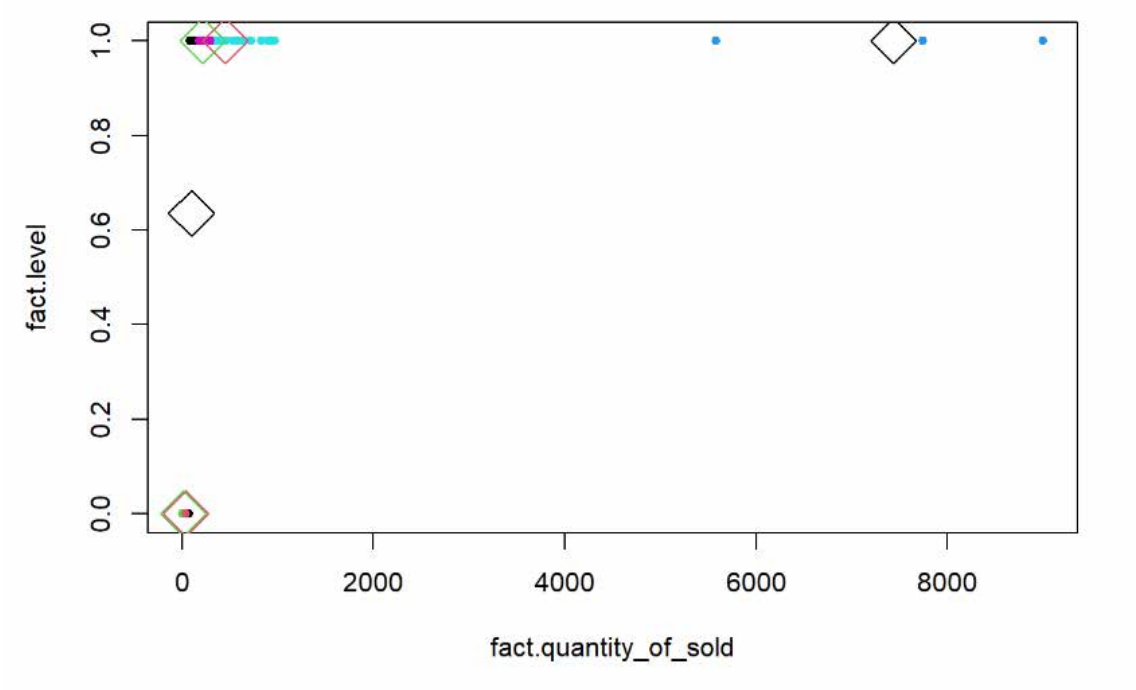
```
# Виведення результатів кластеризації
print(kmeans_result)
```

```
## K-means clustering with 6 clusters of sizes 809, 1258, 1284, 30, 290, 488
##
## Cluster means:
##   fact.quantity_of_sold fact.level
## 1              94.69221  0.6353523
## 2              36.66773  0.0000000
## 3              10.68458  0.0000000
## 4            7440.00000  1.0000000
## 5             452.44828  1.0000000
## 6             212.95492  1.0000000
```

4.3.10 Результати кластеризації

На графіку рис. 4.3.11 видно, що дані розділяються на дві головні категорії: низький рівень продажів, який представлений значеннями нуль, та високий рівень продажів, який відповідає значенню одиниця. Крім того, спостерігається наявність двох додаткових груп: одна з них відображає посередні продажі, а інша містить викиди з дуже високими значеннями.

Загалом, розподіл продажів можна класифікувати на шість основних категорій: дві низькі, одна посередня та три категорії високих продажів. Але З цього графіку можна зробити висновок, що більшість об’єктів відноситься до категорії низьких продажів, що може бути корисною інформацією для розробки маркетингових та стратегічних рішень.



4.3.11 Візуалізація результатів кластеризації

Проте, алгоритм k-середніх має деякі недоліки. Зокрема, він може збитися з курсу, якщо початкові центри кластерів випадково вибрані погано, та не завжди дозволяє знайти глобальний мінімум. Також цей алгоритм не ефективний при роботі з великими об’ємами даних.

ВИСНОВКИ

У даній магістерській роботі було проведено аналіз для розробки інформаційно-аналітичної системи для аналізу систем обліку складських операцій. У науковому дослідженні було виявлено фактори, що так чи інакше впливають на аналіз обліку складських операцій, а також розроблені рекомендації та стратегії для підвищення ефективності та якості обліку та управління складськими операціями. Дослідження було спрямоване на ідентифікацію потенційних проблем та надання практичних рекомендацій для оптимізації процесів складського управління з метою підвищення конкурентоспроможності та ефективності підприємства на ринку.

Було проведено дослідження методів Data Mining для покращення аналітики систем обліку на складах.

У першому розділі була проведена характеристика предметної області, де розглянуті основні аспекти Data Mining для аналізу складських операцій. Було проаналізовано існуючі рішення та сформульовано завдання для дослідження.

У наступному розділі надано теоретичний огляд методології системного аналізу та моделювання системи. Вивчено основні поняття моделювання та використані технології, зокрема UML. Були створені діаграми для опису та кращого розуміння процесів розробленої інформаційно-аналітичної системи.

Розділ три охопив розробку системи, включаючи створення оперативної бази даних, сховища даних, підготовку та обробку інформації, заповнення сховища даними.

У четвертому розділі було проведено дослідження використання методів Data Mining та визначено їх потрібність.

Першим було досліджено використання задач класифікації. Один із методів, а саме використання алгоритму 1-Rule для класифікації, дозволив точно визначити категорії для деяких об’єктів дослідження. Однак, при цьому методі було виявлено обмеження щодо точності класифікації в разі складних даних.

Також, використання методу наївного Байеса показало високу точність класифікації, особливо при наявності великої кількості даних.

У підрозділі дослідження використання методу асоціативних правил було виявлено, цей метод дозволив виявити цікаві зв’язки та асоціації між даними. А саме було розглянуто, що саме впливає на низький рівень продажів. Отримані асоціативні правила можуть бути корисними для усунення цих проблем та оптимізації рівня продажів.

І останнім було проведено дослідження використання алгоритмів кластеризації. Результати кластеризації показали, що дані діляться на декілька груп з різним рівнем продажів та інших параметрів. Це може бути корисним для ідентифікації груп товарів та розробки індивідуальних стратегій управління для кожної групи.

Загалом, дана магістерська робота дала можливість розширити розуміння про використання методів Data Mining для аналізу складських операцій. Отримані результати вказують на потенціал для покращення аналітики складських операцій та можливості подальшого використання отриманих результатів у практиці.

У майбутньому рекомендується подальше дослідження та вдосконалення методів Data Mining для аналізу складських операцій, а також розробка та впровадження інформаційно-аналітичних систем на складах з метою оптимізації управлінських рішень та підвищення ефективності обліку та контролю на складах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Гриньова В.М., «Організація виробництва». [Онлайновий]. Доступно: https://westudents.com.ua/knigi/159-organzatsya-virobnitstva-grinova-vm.html.

[2] BIZREVIEV., «Облік складських операцій: документи, вимоги, відповідальність.» [Онлайновий]. Доступно: https://bizreview.com.ua/oblik-skladskih-operatsij-dokumenti-vimogi-vidpovidalnist-1s-sklad/.

[3] Збірник доповідей конференції "Теоретичні та прикладні аспекти розробки комп'ютерних систем 2023". [Онлайновий]. Доступно: https://drive.google.com/file/d/1VJLglGUk4dEHKqpdzq-Hx6fPQq4ERMsV/view?usp=sharing

[4] Збірник доповідей XIV міжнародної науково-практичної конференції молодих вчених "Інформаційні технології: економіка, техніка, освіта".

[5] І. Ю. Вольвач "Досвід впровадження логістичної концепції виробництва "Just-in-Time"." [Онлайновий]. Доступно: http://journals.khnu.km.ua/vestnik/pdf/ekon/2009_4_2/pdf/250-253.pdf.

[6] І. В. Кулаковська, "Логістика та методи логістичного аналізу." [Онлайновий]. Доступно: http://surl.li/msvqw

[7] Чайка, Т. Ю. (2011). "ABC-аналіз у системі методів аналізу асортименту продукції роздрібних підприємств". [Онлайновий]. Доступно: https://core.ac.uk/download/pdf/161789461.pdf

[8] ТОВ "ОРДЕРІ" (2023). Программа для управлінського обліку. [Онлайновий]. Доступно: https://remonline.ua/features/management-accounting/

[9] Державна наукова установа «Енциклопедичне видавництво» за участі Інституту програмних систем НАН України, 2015-2023. [Онлайновий]. Доступно: http://surl.li/manpb.

[10] um.co.ua - учбові матеріали та реферати, «Діаграма прецедентів. Роль прецедентів при розробці ПС.» [Онлайновий]. Доступно: http://um.co.ua/8/8-2/8-213194.htm.

[11] Діаграма послідовності (Sequence Diagrams) Maxym Zosym © 2023. [Онлайновий]. Доступно: https://www.maxzosim.com/sequence-diagrams/

[12] Vuzlit - архив студенческих работ (info{at}vuzlit.com) © 2017 - 2023. [Онлайновий]. Доступно: https://vuzlit.com/1009782/diagrama_rozgortannya

[13] «Що таке OLAP?» [Онлайновий]. Доступно: https://uk.education-wiki.com/5528785-what-is-olap.

[14] Microsoft (2023). Основные сведения об Analysis Services. [Онлайновий]. Доступно: https://learn.microsoft.com/ru-ru/analysis-services/analysis-services-overview?view=asallproducts-allversions

[15] Microsoft (2023). SQL Server Integration Services. [Онлайновий]. Доступно: https://learn.microsoft.com/ru-ru/sql/integration-services/sql-server-integration-services?view=sql-server-ver16

[16] Архів інтернету. "Технології інтелектуальних обчислень - стан проблеми, нові рішення." [Онлайновий] Доступно: https://web.archive.org/web/20170724125601/http://www.victoria.lviv.ua/html/oio/html/theme2.htm#2_1_1 (Дата звернення: 27.09.2023).

[17] The R Project for Statistical Computing. [Online]. Available: https://www.r-project.org/. © The R Foundation.

[18] RStudio. [Online]. Available: https://www.rstudio.com/. © RStudio, PBC.

[19] Opendatabot. "ТОВ «ГРІН ФУД»." [Онлайновий] Доступно: https://opendatabot.ua/c/37269590 (Дата звернення: 02.102023).

[20] "1R Classification Algorithm" - Стаття, автор: James Tharpe, [Онлайновий] Доступно: https://www.jamestharpe.com/1r-algorithm/ (Дата звернення: 07.09.2023).

[21] "An Introduction to Naive Bayes Algorithm for Beginners" - Стаття, © 2023 Turing, Адреса: 1900 Embarcadero Road Palo Alto, CA, 94303. [Онлайновий] Доступно: https://www.turing.com/kb/an-introduction-to-naive-bayes-algorithm-for-beginners (Дата звернення: 07.09.2023).

[22] "Association Rules in Data Mining" - All Rights Reserved, Copyright 2010-2023, TechTarget. [Онлайновий] Доступно: http://surl.li/msvzh.

[23] "Exploring Clustering Algorithms: Explanation and Use Cases" - Copyright © 2023 Neptune Labs. [Онлайновий] Доступно: https://neptune.ai/blog/clustering-algorithms

[24] "Метод К-середніх (K-means)" - © Підручники для студентів онлайн. [Онлайновий]Доступно:https://stud.com.ua/10870/marketing/metod_serednih_means

[25] Гради Буч. "Объектно-ориентированный анализ и проектирование с примерами." [Онлайновий]. Доступно: https://coollib.com/b/198321-gradi-buch-obektno-orientirovannyiy-analiz-i-proektirovanie-s-primerami-prilozheniy-na-s/read.

[26] Microsoft SQL Server. [Онлайновий]. Доступно: https://www.microsoft.com/ru-ru/sql-server/sql-server-downloads. © Microsoft 2023.

ДОДАТОК А

КОД SQL ЗАПИТУ ДЛЯ ЗАПОВНЕННЯ СХОВИЩА ДАНИХ З ОПЕРАТИВНОЇ БАЗИ ДАНИХ

Сторінок – 1

```
SELECT
    wd.id_date,
    so.supplier_id,
    co.customer_id,
    coi.goods_id,
    SUM(coi.quantity_of_goods) AS sold
FROM
    Warehouse.dbo.dateDim wd
JOIN
    Storage.dbo.customer_order co
ON
    wd.year = YEAR(co.customer_order_date)
    AND wd.month = MONTH(co.customer_order_date)
JOIN
    Storage.dbo.customer_order_item coi
ON
    co.id = coi.order_id
JOIN
    Storage.dbo.storage_order_item soi
ON
    coi.goods_id = soi.goods_id
JOIN
    Storage.dbo.storage_order so
ON
    soi.order_id = so.id
GROUP BY
    wd.id_date,
    so.supplier_id,
    co.customer_id,
    coi.goods_id
```

ДОДАТОК Б

РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ МЕТОДУ НАЇВНОГО БАЙЕСА

Сторінок – 2

```
##    supplier   customer       High        Low
## 1       333   19421419 26.78241 % 73.21759 %
## 2       558   19421419  26.7889 %  73.2111 %
## 3       560   19421419 26.78896 % 73.21104 %
## 4      1135   19421419 26.80574 % 73.19426 %
## 5     88814   19421419 32.89301 % 67.10699 %
## 6       333   32974402 26.81655 % 73.18345 %
## 7       558   32974402 26.82305 % 73.17695 %
## 8       560   32974402 26.82311 % 73.17689 %
## 9      1135   32974402 26.83991 % 73.16009 %
## 10    88814   32974402 32.93144 % 67.06856 %
## 11      333   34408235 26.82017 % 73.17983 %
## 12      558   34408235 26.82667 % 73.17333 %
## 13      560   34408235 26.82672 % 73.17328 %
## 14     1135   34408235 26.84352 % 73.15648 %
## 15    88814   34408235 32.93551 % 67.06449 %
## 16      333   38467650  26.8304 %  73.1696 %
## 17      558   38467650  26.8369 %  73.1631 %
## 18      560   38467650 26.83696 % 73.16304 %
## 19     1135   38467650 26.85376 % 73.14624 %
## 20    88814   38467650 32.94703 % 67.05297 %
## 21      333   44606332 26.84588 % 73.15412 %
## 22      558   44606332 26.85238 % 73.14762 %
## 23      560   44606332 26.85244 % 73.14756 %
## 24     1135   44606332 26.86925 % 73.13075 %
## 25    88814   44606332 32.96445 % 67.03555 %
## 26      333 2839807409 34.44546 % 65.55454 %
## 27      558 2839807409 34.45293 % 65.54707 %
## 28      560 2839807409   34.453 %   65.547 %
## 29     1135 2839807409 34.47232 % 65.52768 %
## 30    88814 2839807409  41.3178 %  58.6822 %
## 31      333 3472411041 36.30208 % 63.69792 %
```

```
## 32      558 3472411041 36.30974 % 63.69026 %
## 33      560 3472411041 36.30981 % 63.69019 %
## 34     1135 3472411041  36.3296 %  63.6704 %
## 35    88814 3472411041 43.30018 % 56.69982 %
## 36      333 3248211238 35.63906 % 64.36094 %
## 37      558 3248211238 35.64666 % 64.35334 %
## 38      560 3248211238 35.64673 % 64.35327 %
## 39     1135 3248211238 35.66636 % 64.33364 %
## 40    88814 3248211238 42.59482 % 57.40518 %
## 41      333   37193071 26.82719 % 73.17281 %
## 42      558   37193071 26.83369 % 73.16631 %
## 43      560   37193071 26.83375 % 73.16625 %
## 44     1135   37193071 26.85055 % 73.14945 %
## 45    88814   37193071 32.94341 % 67.05659 %
## 46      333 2301118107 32.90075 % 67.09925 %
## 47      558 2301118107 32.90805 % 67.09195 %
## 48      560 2301118107 32.90812 % 67.09188 %
## 49     1135 2301118107 32.92701 % 67.07299 %
## 50    88814 2301118107 39.65131 % 60.34869 %
## 51      333 2498802470 33.46355 % 66.53645 %
## 52      558 2498802470 33.47092 % 66.52908 %
## 53      560 2498802470 33.47099 % 66.52901 %
## 54     1135 2498802470 33.49004 % 66.50996 %
## 55    88814 2498802470 40.26031 % 59.73969 %
## 56      333 2830117633 34.41737 % 65.58263 %
## 57      558 2830117633 34.42484 % 65.57516 %
## 58      560 2830117633  34.4249 %  65.5751 %
## 59     1135 2830117633 34.44422 % 65.55578 %
## 60    88814 2830117633 41.28764 % 58.71236 %
```

ДОДАТОК В

КОД РЕАЛІЗАЦІЇ АЛГОРИТМУ 1-RULE

Сторінок – 4

```
# Опція для відображення даних без експоненційної форми
options(scipen=999)

# Завантаження бібліотек
library(odbc) # для підключення до сховища даних
library(dplyr) # для реалізації алгоритму
library(ggplot2) # для виведеня графіку

# Підключення до бази даних
con <- dbConnect(odbc(),
                 Driver = "ODBC Driver 17 for SQL Server",
                 Server = "DESKTOP-5E8892J",
                 Database = "Warehouse",
                 UID = "sa",
                 PWD = "sa")

# SQL запит запит для отримання даних
query <- "SELECT * FROM Warehouse.dbo.fact"

# Отримання даних з таблиці supplier та customer
supplier_data <- dbGetQuery(con, "SELECT id_supplier, supplier_name AS supplier_name FROM supplierDim")
customer_data <- dbGetQuery(con, "SELECT id_customer, customer_name AS customer_name FROM customerDim")

# Завантаження даних з бази даних
fact <- dbGetQuery(con, query)

# Визначте відсоток видалення викидів (наприклад, 10% з обох кінців)
trim_percent <- 10

# Знайдіть квантилі для видалення
lower_quantile <- trim_percent / 200
upper_quantile <- 1 - (trim_percent / 200)

# Обчислення обрізаного середнього
```

```
mean <- mean(fact$quantity_of_sold[fact$quantity_of_sold >= quantile(fact$quantity_of_sold, lower_quantile) & fact$quantity_of_sold <= quantile(fact$quantity_of_sold, upper_quantile)])

data <- fact$quantity_of_sold[fact$quantity_of_sold >= quantile(fact$quantity_of_sold, lower_quantile) & fact$quantity_of_sold <= quantile(fact$quantity_of_sold, upper_quantile)]
# Створення діаграми розсіювання для всіх значень
ggplot(data = fact, aes(x = 1, y = quantity_of_sold)) +
  geom_jitter(width = 0.1, height = 0, alpha = 0.5, color = "gray60") +
  geom_hline(yintercept = max(data), linetype = "dashed", color = "indianred1", size = 1) +
  geom_hline(yintercept = mean, linetype = "solid", color = "palegreen", size = 1) +
  geom_hline(yintercept = min(data), linetype = "dashed", color = "indianred1", size = 1) +
  labs(title = "Діаграма розсіювання для кількості продажів", x = NULL, y = "Кількість продажів")
 +  theme_minimal() +
  theme(axis.text.x = element_blank())

# Створення діаграми розсіювання для всіх значень
ggplot(data = fact, aes(x = 1, y = quantity_of_sold)) +
  geom_jitter(width = 0.1, height = 0, alpha = 0.5, color = "gray60") +
  geom_hline(yintercept = max(data), linetype = "dashed", color = "indianred1", size = 1) +
  geom_hline(yintercept = mean, linetype = "solid", color = "palegreen", size = 1) +
  geom_hline(yintercept = min(data), linetype = "dashed", color = "indianred1", size = 1) +
  labs(title = "Діаграма розсіювання для кількості продажів", x = NULL, y = "Кількість продажів")
+ theme_minimal() +
  theme(axis.text.x = element_blank()) +
  ylim(0, 1000)

# Класифікація
n <- nrow(fact)
results <- character(n)
for (i in 1:n) {
  if (fact[i, 5] > mean) {
    results[i] <- "High"
  } else {
    results[i] <- "Low"
  }
}

# Додавання класифікації до даних
fact$level <- results

independent_variable <- fact$id_supplier
```

```
class_variable <- fact$level

data <- data.frame(independent_variable, class_variable)

# Реалізація алгоритму 1-Rule
rule <- data %>%
  group_by(independent_variable) %>% # групує дані за значенням незалежної змінної
  summarise(most_frequent_class = names(sort(-table(class_variable))[1])) # обчислює найчастіший клас для кожного значення незалежної змінно

# Підрахунок кількості спостережень для кожного правила
rule_count <- data %>%
  inner_join(rule, by = "independent_variable") %>%
  group_by(independent_variable, most_frequent_class) %>%
  summarise(count = sum(class_variable == most_frequent_class))

# Підрахунок відносної частоти кожного класу для кожного значення незалежної змінної
class_prob <- rule_count %>%
  group_by(independent_variable) %>%
  mutate(total_count = sum(data$independent_variable == independent_variable),
         class_probability = paste(round(count / total_count * 100, 2), "%"))

# Включення імен постачальників у результати за id_supplier
class_prob_supplier <- class_prob %>%
  left_join(supplier_data, by = c("independent_variable" = "id_supplier")) %>%
  select(supplier_name, independent_variable, most_frequent_class, count, total_count, class_probability)

# Перейменування стовпців на виводі
class_prob_supplier <- class_prob_supplier %>%
  rename(
    supplier = supplier_name,
    id = independent_variable,
    level = most_frequent_class,
    cout = count,
    total = total_count,
    probability = class_probability
  )

print(class_prob_supplier)

independent_variable <- fact$id_customer
```

```
class_variable <- fact$level

data <- data.frame(independent_variable, class_variable)

rule <- data %>%
  group_by(independent_variable) %>%
  summarise(most_frequent_class = names(sort(-table(class_variable))[1]))
rule_count <- data %>%
  inner_join(rule, by = "independent_variable") %>%
  group_by(independent_variable, most_frequent_class) %>%
  summarise(count = sum(class_variable == most_frequent_class))

class_prob <- rule_count %>%
  group_by(independent_variable) %>%
  mutate(total_count = sum(data$independent_variable == independent_variable),
         class_probability = paste(round(count / total_count * 100, 2), "%"))

class_prob_customer <- class_prob %>%
  left_join(customer_data, by = c("independent_variable" = "id_customer")) %>%
  select(customer_name, independent_variable, most_frequent_class, count, total_count, class_probability)

class_prob_customer <- class_prob_customer %>%
  rename(
    customer = customer_name,
    id = independent_variable,
    level = most_frequent_class,
    cout = count,
    total = total_count,
    probability = class_probability
  )

print(class_prob_customer)

dbDisconnect(con)
```

ДОДАТОК Г

КОД РЕАЛІЗАЦІЇ АЛГОРИТМУ НАЇВНОГО БАЙЕСА

Сторінок – 2

```
# Опція для відображення даних без експоненційної форми
options(scipen=999)

# Підключення до бази даних
con <- dbConnect(odbc(),
                 Driver = "ODBC Driver 17 for SQL Server",
                 Server = "DESKTOP-5E8892J",
                 Database = "Warehouse",
                 UID = "sa",
                 PWD = "sa")
# SQL запит запит для отримання даних
query <- "SELECT * FROM Warehouse.dbo.fact"

# Отримання даних з таблиці supplier та customer
supplier_data <- dbGetQuery(con, "SELECT id_supplier, supplier_name AS supplier_name FROM supplierDim")
customer_data <- dbGetQuery(con, "SELECT id_customer, customer_name AS customer_name FROM customerDim")
goods_data <- dbGetQuery(con, "SELECT id_goods, goods_name AS goods_name FROM goodsDim")

# Завантаження даних з бази даних
fact <- dbGetQuery(con, query)

# Визначте відсоток видалення викидів (наприклад, 10% з обох кінців)
trim_percent <- 10

# Знайдіть квантилі для видалення
lower_quantile <- trim_percent / 200
upper_quantile <- 1 - (trim_percent / 200)

# Обчислення обрізаного середнього
mean <- mean(fact$quantity_of_sold[fact$quantity_of_sold >= quantile(fact$quantity_of_sold, lower_quantile) & fact$quantity_of_sold <= quantile(fact$quantity_of_sold, upper_quantile)])

# Класифікація
n <- nrow(fact)
```

```
results <- character(n)
for (i in 1:n) {
 if (fact[i, 5] > mean) {
  results[i] <- "High"
 } else {
  results[i] <- "Low"
 }}

# Додавання класифікації до даних
fact$level <- results

# Навчити модель за допомогою алгоритму Naïve Bayes
model <- naiveBayes(level ~ id_supplier + id_customer, data = fact)
model

# Створення нового набору даних для прогнозування
new_data <- expand.grid(id_supplier = unique(fact$id_supplier), id_customer = unique(fact$id_customer))

# Створення фрейму даних для результатів прогнозування
result_data <- data.frame(supplier = integer(0), customer = integer(0), `Class H` = numeric(0), `Class L` = numeric(0))

# Прогноз класу для кожної пари постачальника і клієнта окремо
for (i in 1:nrow(new_data)) {
 current_supplier <- new_data$id_supplier[i]
 current_customer <- new_data$id_customer[i]

 # Створення окремого набору даних для прогнозування
 current_data <- data.frame(id_supplier = current_supplier, id_customer = current_customer)

 # Прогноз для поточної пари
 current_prediction <- predict(model, newdata = current_data, type = "raw") * 100
 # Додавання результатів до фрейму даних result_data
 result_data <- rbind(result_data, c(current_supplier, current_customer, current_prediction[, "High"], current_prediction[, "Low"]))}

# Встановлення назв стовпців для таблиці
colnames(result_data) <- c("supplier", "customer", "High", "Low")
result_data$High <- paste(round(result_data$High, 5), "%")
result_data$Low <- paste(round(result_data$Low, 5), "%")
```

```
# Вивід результатів у вигляді таблиці
print(result_data)

# Закриття з'єднання з базою даних
dbDisconnect(con)
```

ДОДАТОК Ґ

КОД РЕАЛІЗАЦІЇ АЛГОРИТМУ АСОЦІАТИВНИХ ПРАВИЛ

Сторінок – 2

```
# Завантаження бібліотек
library(odbc)
library(dplyr)
library(arules) # для використання алгоритму Apriori
library(arulesViz) # для візуалізації асоціативних правил

# Опція для відображення даних без експоненційної форми
options(scipen=999)

# Підключення до бази даних
con <- dbConnect(odbc(),
                 Driver = "ODBC Driver 17 for SQL Server",
                 Server = "DESKTOP-5E8892J",
                 Database = "Warehouse",
                 UID = "sa",
                 PWD = "sa")
# SQL запит запит для отримання даних
query <- "SELECT * FROM Warehouse.dbo.fact"

# Отримання даних з таблиці supplier та customer
supplier_data <- dbGetQuery(con, "SELECT id_supplier, supplier_name AS supplier_name FROM supplierDim")
customer_data <- dbGetQuery(con, "SELECT id_customer, customer_name AS customer_name FROM customerDim")
goods_data <- dbGetQuery(con, "SELECT id_goods, goods_name AS goods_name FROM goodsDim")

# Завантаження даних з бази даних
fact <- dbGetQuery(con, query)

# Визначте відсоток видалення викидів (наприклад, 10% з обох кінців)
trim_percent <- 10

# Знайдіть квантилі для видалення
lower_quantile <- trim_percent / 200
upper_quantile <- 1 - (trim_percent / 200)
```

```
# Обчислення обрізаного середнього
mean <- mean(fact$quantity_of_sold[fact$quantity_of_sold >= quantile(fact$quantity_of_sold, lower_quantile) &
fact$quantity_of_sold <= quantile(fact$quantity_of_sold, upper_quantile)])

# Класифікація
n <- nrow(fact)
results <- character(n)
for (i in 1:n) {
  if (fact[i, 5] > mean) {
    results[i] <- "High"
  } else {
    results[i] <- "Low"
  }
}

# Додавання класифікації до даних
fact$level <- results

# Створення прикладу вхідних даних для навчання моделі
data <- data.frame(
  fact.id_supplier = as.character(fact$id_supplier),
  fact.id_customer = as.character(fact$id_customer),
  fact.id_goods = as.character(fact$id_goods),
  fact.level = fact$level
)

# Використання алгоритму Apriori для пошуку корисних асоціацій між продуктами
rules <- apriori(data, parameter = list(supp = 0.1, conf = 0.5))

# Виведення знайдених асоціативних правил
inspect(rules)

# Фільтрація правил
rulesDem <- subset(rules, subset = rhs %in% "fact.level=Low")
inspect(head(rulesDem, n = 10, by = "support"))

# Візуалізація
graph <- plot(head(sort(rulesDem, by = "support"), 10), method = "graph",
            control = list(nodeCol = grey.colors(10),
                           edgeCol = grey(.7), alpha = 1))
plot(graph)
```

```
# Закриття з'єднання з базою даних
dbDisconnect(con)
```

ДОДАТОК Д

КОД РЕАЛІЗАЦІЇ АЛГОРИТМУ КЛАСТЕРИЗАЦІЇ

Сторінок – 3

```
# Завантаження бібліотек
library(odbc)
library(dplyr)
library(stats) # для кластеризації методом k-середніх

# Підключення до бази даних
con <- dbConnect(odbc(),
                 Driver = "ODBC Driver 17 for SQL Server",
                 Server = "DESKTOP-5E8892J",
                 Database = "Warehouse",
                 UID = "sa",
                 PWD = "sa")
# SQL запит запит для отримання даних
query <- "SELECT * FROM Warehouse.dbo.fact"

# Завантаження даних з бази даних
fact <- dbGetQuery(con, query)

# Отримання даних з таблиці supplier та customer
supplier_data <- dbGetQuery(con, "SELECT id_supplier, supplier_name AS supplier_name FROM supplierDim")
customer_data <- dbGetQuery(con, "SELECT id_customer, customer_name AS customer_name FROM customerDim")
goods_data <- dbGetQuery(con, "SELECT id_goods, goods_name AS goods_name FROM goodsDim")

# Визначте відсоток видалення викидів (наприклад, 10% з обох кінців)
trim_percent <- 10

# Знайдіть квантилі для видалення
lower_quantile <- trim_percent / 200
upper_quantile <- 1 - (trim_percent / 200)

# Обчислення обрізаного середнього
mean <- mean(fact$quantity_of_sold[fact$quantity_of_sold >= quantile(fact$quantity_of_sold, lower_quantile) &
fact$quantity_of_sold <= quantile(fact$quantity_of_sold, upper_quantile)])

# Класифікація
```

```
n <- nrow(fact)
results <- character(n)
for (i in 1:n) {
  if (fact[i, 5] > mean) {
    results[i] <- 1 # High
  } else {
    results[i] <- 0 # Low
  }
}

# Додавання класифікації до даних
fact$level <- results

# Створення прикладу вхідних даних для навчання моделі
data <- data.frame(fact$quantity_of_sold, fact$level)

# Від 1 до 10 кластерів
k.values <- 1:10
# Вектор для збереження внутрішньокластерної дисперсії
wss_values <- vector(length = length(k.values))

# Розрахунок внутрішньокластерної дисперсії для кожної кількості кластерів
for(i in k.values) {
  kmeans_result <- kmeans(data, centers = i)
  wss_values[i] <- kmeans_result$tot.withinss
}

# Візуалізація зміни внутрішньокластерної дисперсії залежно від кількості кластерів
plot(k.values, wss_values, type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters K", ylab = "Total within-clusters sum of squares")

# Використання методу "ліктя" для визначення оптимальної кількості кластерів
# За допомогою цього методу ми шукаємо точку на графіку, де зміна внутрішньокластерної дисперсії
# зменшується значно менше, ніж при більшій кількості кластерів
dist <- wss_values[2:length(wss_values)] - wss_values[1:(length(wss_values)-1)]

plot(k.values[-1], dist, type = "b", pch = 19, frame = FALSE,
     xlab = "Number of clusters K",
     ylab = "Difference in within-cluster sum of squares")
# Знаходження оптимальної кількості кластерів за допомогою методу "ліктя"
```

```
# За замовчуванням, mi - це точка на графіку, де зміна внутрішньокластерної дисперсії зменшується значно менше, ніж при більшій кількості кластерів
mi <- which.max(dist) + 1

# Результат
cat("Optimal number of clusters:", mi)
# Кластеризація методом k-середніх
kmeans_result <- kmeans(data, centers=mi)
# Виведення результатів кластеризації
print(kmeans_result)

# Візуалізація результатів кластеризації
plot(data, col=kmeans_result$cluster, pch=20)
points(kmeans_result$centers, col=1:3, pch=23, cex=3)
# Закриття з'єднання з базою даних
dbDisconnect(con)
```