Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему: Обробка масивів даних з використанням мікросервісів

Виконав: студент ІІ курсу
групи 1ПІ-18 м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Буковський Нікіта Валентинович
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ
Хошаба Олександр Мирославович
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. КН
Арсенюк Ігор Ростиславович
(прізвище та ініціали)

ВНТУ – 2019

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Освітньо-кваліфікаційний рівень – магістр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
"____"_____2019 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Буковському Нікіті Валентиновичу

1. Тема роботи – обробка масивів даних з використанням мікросервісів.

Керівник роботи: Хошаба Олександр Мирославович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від "__"_____2019 року №___

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи: базові методи підвищення швидкості обробки даних – методи представлення та обробки даних OLTP; вихідні дані для визначення швидкості транзакцій даних за читанням; дані про розмішування даних в таблицях у кількості 1, 2, 4 та 8 штук; кількість записів у кожній таблиці складає 10000; значення загальної кількості часу роботи скрипта; середній час обробки даних для однієї таблиці.

4. Зміст розрахунково-пояснювальної записки: обгрунтування вибору методу аналізу та постановка задачі дослідження систем з обробки даних, аналіз стану проблеми, порівняльний аналіз аналогів, постановка задачі розробки системи, дослідження методів роботи систем з обробки даних, вимоги щодо дослідження методів роботи систем з обробки даних, дослідження роботи загальної структури систем з обробки даних, дослідження серверно-клієнтської структури з обробки даних, особливості дослідження багатовимірної структури обробки даних, дослідження роботи систем на основі мікросервісних архітектур, необхідність використання кластера RabbitMQ для обробки масивів даних, дослідження створення кластера RabbitMQ для обробки даних, дослідження роботи бази даних, технологічний аудит розроблених методів та засобів процесу обробки масивів даних, розрахунок витрат на розробку методів та засобів обробки масивів даних з використанням мікросервісів, розрахунок економічного ефекту від можливого впровадження розроблених методів та засобів обробки масивів даних з використанням мікросервісів; економічна частина.

5. Перелік графічного матеріалу: актуальність проведення досліджень, постановка задачі; порівняльна характеристика аналогів; адаптивний зафарбування; нові моделі відбивних здатностей поверхонь; інтерфейс 3D – редактора світлових ефектів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Хошаба О. М., к.т.н., доцент кафедри ПЗ		
5	Глущенко Л.Д., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз моделей з обробки даних, аналіз стану проблеми, порівняльний аналіз аналогів	07.10.2018 – 27.10-2019	Вик.
2	Розробка моделей з обробки даних на основі OLTP-технологій	28.10.2019 – 8.11.2019	Вик.
3	Розробка методів впровадження інформа-ційних технологій на основі мікросервісів	9.11.2019 – 20.11.2019	Вик.
4	Виконання експериментальних даних дослідження швидкості обробки даних з використанням OLTP-технологій	21.11.2019 – 3.12.2019	Вик.
5	Економічна частина	4.12.2019 – 8.12.2019	Вик.

Студент _____ **Буковський Н.В.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Хошаба О.М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі виконано дослідження методів обробки масивів даних за допомогою використанням мікросервісів.

Сформульовано мету досліджень - підвищення продуктивності обробки масивів даних за рахунок використання OLTP-технологій та сучасних інформаційних технології мікросервісів. Запропоновано метод визначення ефективності роботи бази даних за допомогою використання OLTP-технологій з обробки даних.

Проведено аналіз стану проблеми обробки даних де визначено два різних підходи щодо побудови систем які основані на OLTP та OLAP методах.

В другому розділі визначені вимоги щодо проектування систем з обробки даних. Було визначено, що типова система з обробки даних, як правило, відрізняється від звичайної реляційної бази даних.

В третьому розділі показана створення системи обробки даних де визначені особливості розробки мікросервісних архітектур. Основна увага приділена основні принципи мікросервісних та сервіс-орієнтованих архітектур. Надана інформація щодо використання сучасних кластерних мікросервісних засобів для створення програмних продуктів, що засновані на обробках даних. Визначені механізми роботи мікросервісів з контейнерами які виконують обробку даних.

Також, в третьому розділі показана необхідність використання кластера RabbitMQ для обробки масивів даних. Визначена загальна схема роботи RabbitMQ під час обробки повідомлень та роботу менеджера RabbitMQ з веб додатком. Показано механізм роботи менеджера RabbitMQ з веб додатком.

Розроблено програмне оточення та додаток з обробки даних на основі мікросервісних технологій.

Отримані в магістерській кваліфікаційній роботі наукові та практичні результати можна використати для побудови систем з обробки даних.

ABSTRACT

In the master's qualification work the methods of processing of data arrays with the help of microservices were investigated.

The purpose of the research is to increase the productivity of data processing due to the use of OLTP-technologies and modern information technology of microservices. The method of determining the efficiency of the database using OLTP technologies for data processing is proposed.

An analysis of the state of the data processing problem is carried out, which identifies two different approaches to building systems based on OLTP and OLAP methods.

The second section defines the requirements for the design of data processing systems. It has been determined that a typical data processing system is generally different from a conventional relational database.

The third section shows the creation of a data processing system that defines the features of development of microservice architectures. The main attention is paid to the basic principles of microservice and service oriented architectures. Information is provided regarding the use of state-of-the-art cluster microservices to create data-based software products. The mechanisms of work of microservices with containers that perform data processing are defined.

Also, Section 3 shows the need to use the RabbitMQ cluster to handle data arrays. The general scheme of work of RabbitMQ during message processing and the operation of the RabbitMQ manager with a web application is defined. The mechanism of the RabbitMQ manager with the web application is shown.

Developed software and application for data processing based on microservice technologies. The scientific and practical results obtained in the master's qualification work can be used to build data processing systems.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. На сьогоднішній день важко уявити ІТ-інфраструктуру підприємства, що не використовує сучасні інформаційні технології. В основному, використання сучасних інформаційних технологій признано для обробки різних типів інформації яка необхідна для обслуговування бухгалтерії, інформаційних архівів, телефонних мереж, реєстрації документів, банківських операцій та інше [1,2].

З появою персональних комп'ютерів сучасні системи обробки інформації стали доступними для безлічі дрібних і середніх фірм, підприємств і організацій. Системи оперативної обробки інформації отримали назву OLTP (On-Line Transaction Processing - оперативна, тобто в режимі реального часу, обробка транзакцій).

Типовим прикладом поширеного застосування OLTP-систем є масове обслуговування клієнтів, наприклад бронювання авіаквитків або оплата послуг телефонних компаній. Обидві ці ситуації мають два загальних властивості: дуже велике число клієнтів і безперервне надходження інформації.

При бронюванні авіаквитків з численних пунктів продажу безперервно стікається інформація про вже продані квитки, яку вводять зі своїх робочих місць оператори-продавці. У тій же час в базі даних формується інформація про вільні місця. З точки зору даного завдання транзакція включає в себе набір таких важливих дій, як: запит оператора про наявність вільних місць на той чи інший рейс; відгук БД з наданням відповідної інформації; введення оператором інформації про клієнта, номері замовленого місця і сплаченій сумі; передача нової інформації в базу даних і внесення до неї відповідних змін; передача оператору підтвердження про те, що операція пройшла успішно. Такі транзакції виконуються тисячі разів в день в сотнях пунктів продажів білетів. Очевидно, що основним пріоритетом в даному випадку є забезпечення мінімального часу відгуку при максимальному завантаженні системи [1,2].

Мета та завдання дослідження. Метою роботи є підвищення ефективності

обробки масивів даних за рахунок використання сучасних технологій.

Основними задачами дослідження є:

- провести аналіз та дослідження існуючих методів підвищення швидкості обробки інформації у базах даних;

- запропонувати нові:

- методи підвищення продуктивності швидкості обробки даних на основі використання сучасних технології мікросервісів;

- розробити програмні компоненти з обробки даних на основі використання сучасних технології мікросервісів;

- провести експериментальні дослідження розроблених засобів обробки даних.

Об’єкт дослідження – процес обробки масивів даних.

Предмет дослідження – методи та засоби процесу обробки масивів даних.

Методи дослідження. У процесі досліджень використовувались: теорія чисел та чисельних методів, теорія диференціально-інтегрального числення; аналіз та перевірка отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Вперше запропоновано метод обробки масивів даних, особливість якого полягає у використанні OLTP технологій, що забезпечує підвищення швидкості обробки транзакцій записів у таблицях внаслідок оптимізацій даних, і як наслідок, дає можливість підвищити ефективність роботи до 17,165%.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано технології обробки інформації та розроблено програмні засоби обробки даних на основі контейнерних технологій.

Впровадження. Впровадження результатів досліджень підтверджуються відповідними актами та використовуються на таких підприємствах і організаціях:

- компанія «SoftSystems» для підвищення продуктивності роботи серверних

додатків.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалась згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві на конференції професорсько-викладацього складу ВНТУ, автору належать результати дослідження, які були отримані в ході дослідження продуктивності систем на основі мікросервісних технологій.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на конференції професорсько-викладацього складу ВНТУ (Вінниця, 2017).

Структура та обсяг роботи. Магістерська кваліфікаційна роботи складається зі вступу, чотирьох розділів, висновків, списку літератури, що містить __ найменувань, __ додатків. Робота містить __ ілюстрацій, __ таблиці.

РОЗДІЛ 1
ОБГРУНТУВАННЯ ВИБОРУ МЕТОДУ АНАЛІЗА ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ СИСТЕМ З ОБРОБКИ ДАНИХ

1.1. Аналіз стану проблеми

Існує два різних підходи щодо обробки інформації у базах даних: OLTP та OLAP. Перший підхід (Online Transaction Processing) - являє собою обробку транзакцій в реальному часі. Цей підхід розрахований на ефективну обробку даних в реальному часі.

Другий підхід (Online Analytical Processing) - полягає в аналітичній обробці інформації в реальному часі, але націлене саме на вибірку та обробку даних максимально ефективним способом [3].

У бізнесі щодня доводиться приймати безліч рішень:

- виробничі, маркетингові і кадрові рішення;
- рішення, що визначають ціни, продажу, знижки.

Ухвалення ефективних рішень має відбуватися на всіх рівнях управління організацією, що в кінцевому підсумку призводить до успіху всієї організації в цілому [3,4].

Також, OLAP (on-line analytical processing) - це набір технологій для оперативного опрацювання інформації, що включає динамічну побудову звітів в різних розрізах, аналіз даних, моніторинг і прогнозування ключових показників бізнесу. В основі OLAP-технологій лежить подання інформації у вигляді OLAP-кубів.

OLAP-куби містять бізнес-показники, що використовуються для аналізу і прийняття управлінських рішень, наприклад: прибуток, рентабельність продукції, сукупні кошти (активи), власні кошти, позикові кошти та інше.

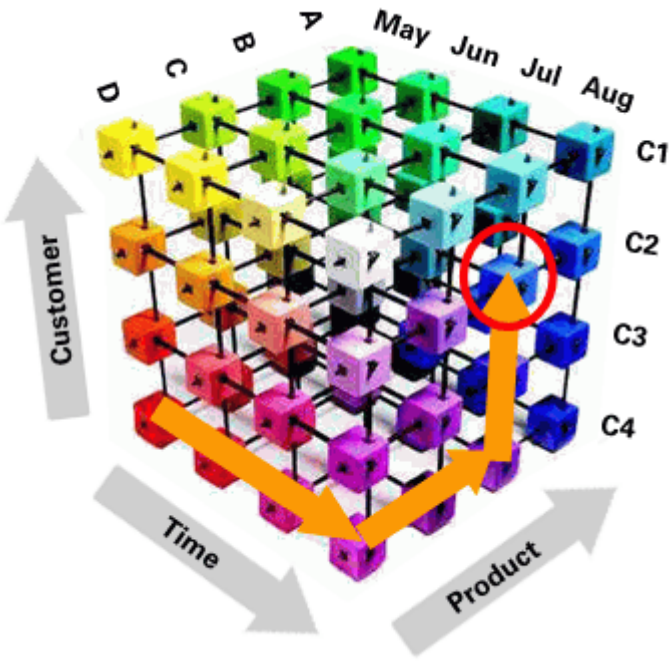


Рисунок 1.1 – Загальний вигляд моделі OLAP-кубу, що використовується під час обробки даних

Бізнес-показники зберігаються в кубах не у вигляді простих таблиць, як в звичайних системах обліку або бухгалтерських програмах, а в розрізах, що представляють собою основні бізнес-категорії діяльності організації: товари, магазини, клієнти, час продажів та інше.

Завдяки детальному структуруванні інформації OLAP-куби дозволяють оперативно здійснювати аналіз даних і формувати звіти в різних розрізах і з довільною глибиною деталізації. Звіти можуть створюватися аналітиками, менеджерами, фінансистами, керівниками підрозділів в інтерактивному режимі для того, щоб швидко отримати відповіді, на які виникають щодня питання, і прийняти правильне рішення. При цьому співробітникам, для створення звітів не потрібно вдаватися до послуг програмістів, на що зазвичай йде чимало часу.

Тому, часто в компаніях існує кілька інформаційних систем - системи складського обліку, бухгалтерські системи, ERP системи для автоматизації окремих виробничих процесів, системи збору звітності з підрозділів компанії, а також безліч файлів, які знаходиться на комп'ютерах співробітників [5].

Маючи стільки розрізнених джерел інформації, часто буває дуже складно отримати відповіді на ключові питання діяльності компанії і побачити загальну

картину. А коли потрібна інформація знаходиться в одній з використовуваних систем або локальному файлі, то вона часто виявляється застарілою або суперечить інформації, отриманої з іншої системи. Дана проблема ефективно вирішується за допомогою інформаційно-аналітичних систем, що побудовані на базі OLAP-технологій.

OLAP-системи інтегрують вже існуючі системи обліку, що надають користувачу інструменти для:

- аналізу великих обсягів даних в реальному часі;

- динамічного конструювання звітів;

- моніторингу та прогнозування ключових бізнес-показників.

Відомо, що ключову роль в управлінні компанією грає інформація. Як правило, навіть невеликі компанії використовують кілька інформаційних систем для автоматизації різних сфер діяльності. Отримання аналітичної звітності в інформаційних системах, заснованих на традиційних базах даних пов'язане з такими обмеженнями як:

- розробка кожного звіту вимагає роботи програміста;

- звіти формуються дуже повільно (найчастіше кілька годин), сповільнюючи при цьому роботу всієї інформаційної системи.

- дані, що отримані від різних структурних елементів компанії не уніфіковані і часто суперечливі.

Тому, OLAP-системи, самою ідеологією своєї побудови призначені для аналізу великих обсягів інформації, дозволяють подолати обмеження традиційних інформаційних систем. Однак, створення OLAP-системи на підприємстві дозволяє (рис. 1.2):

- інтегрувати дані різних інформаційних систем;

- проектувати нові звіти без участі програмістів;

- аналізувати дані по будь-яким категоріям і показниками бізнесу на будь-якому рівні деталізації у реальному часі.

- здійснювати моніторинг і прогнозування ключових показників бізнесу.

Таким чином, під час роботи з OLAP-системою, завжди можна знайти

відповіді, на виникаючі питання, побачити картину в цілому, проводити постійний моніторинг стану бізнесу. При цьому можна бути впевненими, що використовується тільки актуальна інформація [6,7].

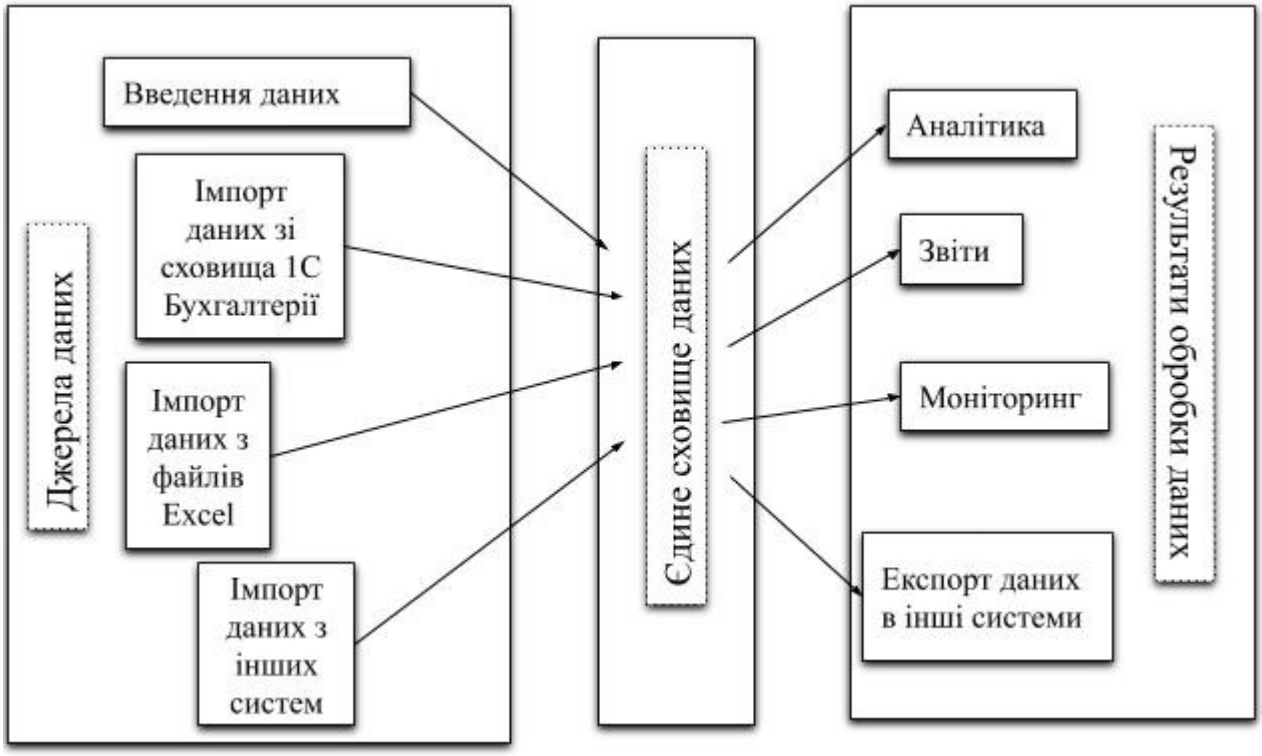


Рисунок 1.2 – Структура інформаційно-аналітичних систем з обробки даних

Інформаційні системи масштабу підприємства, як правило, містять додатки, що призначені для комплексного багатовимірного аналізу даних, їх динаміки, тенденцій та інше. Такий аналіз в кінцевому підсумку покликаний сприяти прийняттю рішень. Нерідко ці системи так і називаються - системи підтримки прийняття рішень (рис. 1.2).

Прийняти будь-яке управлінське рішення неможливо не володіючи необхідною для цього інформацією, зазвичай кількісної. Для цього необхідне створення сховищ даних (Data warehouses), тобто процес збору, відсіювання і попередньої обробки даних з метою надання результуючої інформації користувачам для статистичного аналізу (а нерідко і створення аналітичних звітів).

Ральф Кімбол (Ralph Kimball), один з авторів концепції сховищ даних, описував сховище даних як "місце, де люди можуть отримати доступ до своїх даних". Він також сформулював і основні вимоги до сховищ даних:

- підтримка високої швидкості отримання даних зі сховища;
- підтримка внутрішньої несуперечності даних;
- можливість отримання і порівняння так званих зрізів даних (slice and dice);
- наявність зручних утиліт для перегляду даних в сховищі;
- повнота і достовірність даних, що зберігаються;
- підтримка якісного процесу поповнення даних.

Задовольняти всім перерахованим вимогам в рамках одного і того ж продукту часто не вдається. Тому для реалізації сховищ даних зазвичай використовується кілька продуктів, одні з яких представляють собою власне засоби зберігання даних, інші - кошти їх вилучення та перегляду, треті - кошти їх поповнення та інше [8].

Типове сховище даних, як правило, відрізняється від звичайної реляційної бази даних. По-перше, звичайні бази даних призначені для того, щоб допомогти користувачам виконувати повсякденну роботу, тоді як сховища даних призначені для прийняття рішень. Наприклад, продаж товару і виписування рахунку здійснюються з використанням бази даних, призначеної для обробки транзакцій, а аналіз динаміки продажів за декілька років, що дозволяє спланувати роботу з постачальниками - за допомогою сховища даних.

По-друге, звичайні бази даних постійно змінюються в процесі роботи користувачів, а сховище даних відносно стабільне: дані в ньому зазвичай оновлюються за розкладом. Наприклад, щотижня, щодня або щогодини - залежно від потреб користувача. В ідеалі процес поповнення являє собою просто додавання нових даних за певний період часу без зміни колишньої інформації, що вже перебуває в сховище.

По-третє, звичайні бази даних найчастіше є джерелом даних, що потрапляють в сховище. Крім того, сховище може поповнюватися за рахунок зовнішніх джерел, наприклад статистичних звітів.

Кінцевою метою використання OLAP є аналіз даних і представлення результатів цього аналізу у вигляді, зручному для сприйняття і прийняття рішень. Основна ідея OLAP полягає в побудові багатовимірних кубів, які будуть доступні

для користувача запитів. Однак вихідні дані для побудови OLAP-кубів зазвичай зберігаються в реляційних базах даних. Нерідко це спеціалізовані реляційні бази даних, звані також сховищами даних (Data Warehouse). На відміну від так званих оперативних баз даних, з якими працюють програми, що модифікують дані, сховища даних призначені виключно для обробки і аналізу інформації. Тому проектуються вони таким чином, щоб час виконання запитів до них було мінімальним. Зазвичай дані копіюються в сховище з оперативних баз даних згідно з визначеним розкладом.

Типова структура сховища даних істотно відрізняється від структури звичайної реляційної бази даних. Як правило, ця структура денормалізована (це дозволяє підвищити швидкість виконання запитів), тому може допускати надмірність даних [9].

Системи підтримки прийняття рішень зазвичай мають засоби надання користувачеві агрегатних даних для різних вибірок з вихідного набору в зручному для сприйняття і аналізу вигляді. Як правило, такі агрегатні функції утворюють багатовимірний (і, отже, нереляційний) набір даних (нерідко званий гіперкубом або метакубом), осі якого містять параметри, а осередки - залежні від них агрегатні дані. Причому зберігатися такі дані можуть і в реляційних таблицях, але в даному випадку мова йде про логічної організації даних, а не про фізичну реалізації їх зберігання.

Уздовж кожної осі дані можуть бути організовані у вигляді ієрархії, що представляє різні рівні їх деталізації. Завдяки такій моделі даних користувачі можуть формувати складні запити, генерувати звіти, отримувати підмножини даних [10].

Технологія комплексного багатовимірного аналізу даних отримала назву OLAP (On-Line Analytical Processing). OLAP - це ключовий компонент організації сховищ даних. Концепція OLAP була описана в 1993 році Едгаром Коддом, відомим дослідником баз даних і автором реляційної моделі даних. У 1995 році на основі вимог, викладених Коддом, було сформульовано так званий тест FASMI (Fast Analysis of Shared Multidimensional Information - швидкий аналіз розділяється

багатовимірної інформації). До його складу відносяться такі вимоги щодо додатків для багатовимірного аналізу:

- надання користувачу результатів аналізу за прийнятний час (зазвичай не більше 5 с), нехай навіть ціною менш детального аналізу;

- можливість здійснення будь-якого логічного і статистичного аналізу, характерного для цього додатка, його збереження в доступному для кінцевого користувача вигляді;

- розрахований на багато користувачів доступ до даних з підтримкою відповідних механізмів блокувань і засобів авторизованого доступу;

- багатовимірне концептуальне представлення даних, включаючи повну підтримку для ієрархій і множинних ієрархій (це - ключова вимога OLAP);

- можливість звертатися до будь-якої потрібної інформації незалежно від її обсягу і місця зберігання.

Слід зазначити, що OLAP-функціональність може бути реалізована різними способами, починаючи з найпростіших засобів аналізу даних в офісних додатках і закінчуючи розподіленими аналітичними системами, що засновані на серверних продуктах [11].

1.2. Порівняльний аналіз аналогів

Під час проведення порівняльного аналіз аналогів важливо спочатку розглянути характерні риси процесів, які властиві та притаманні в тій чи іншій мірі система з обробки даних на основі OLTP та OLAP методів.

Особливістю таких систем є те, що запити та звіти що присутні як результати обробки даних є повністю регламентовані. Тому, в більшості випадків, користувач не може сформувати власний запит, щоб уточнити або проаналізувати будь-яку інформацію що підлягає обробці.

Як тільки бізнес-процес завершився, інформація про обслуговування клієнта втрачає сенс, стає не актуальною і підлягає видаленню після певного часу (тобто історичні дані не підтримуються). Тоді, операції проводяться над даними користувачів з максимальним рівнем деталізації, тобто по кожному клієнту

окремо.

Розглянемо приклад з обробкою даних в галузі транспортних перевезень. Ситуація докорінно змінюється, коли керівництво компанії приймає рішення про вивчення потоків даних з метою, наприклад, їх оптимізації. Такі дослідження можуть бути певною реакцією на інформацію. Про те, останнім часом у багатьох пунктах продажів, наприклад білетів, почастішали випадки нестачі квитків на певні маршрути, що дозволяє зробити припущення про доцільність організації додаткових рейсів на транспортні засоби.

Однак для проведення таких досліджень необхідні як мінімум три речі. По-перше, потрібні дані про продажі квитків за досить тривалий період (кілька місяців або років). По-друге, дані не повинні містити протиріч, пропусків, аномальних значень та інших чинників, які не дозволять виконати коректний аналіз. По-третє, необхідна додаткова інформація про бізнес-середовищі: про конкурентів, ринкові тенденції, ціни на паливо та інше. Очевидно, що типова OLTP-система не може забезпечити нічого з перерахованого. Саме з розумінням цих проблем приходить усвідомлення необхідності використання більш розвинених систем зберігання даних, що орієнтовані на аналіз.

Головна вимога до OLTP-систем - це швидке обслуговування відносно простих запитів великої кількості користувачів. При цьому час очікування виконання типового запиту не повинно перевищувати кілька секунд. Згодом в таких системах почали оброблялись великі обсяги даних: документи, відомості про банківські операції, інформація про клієнтів, укладених угодах, надані послуги тощо.

Поступово виникло розуміння того, що збір даних не самоціль. Зібрана інформація може виявитися досить корисною в процесі управління організацією, пошуку шляхів вдосконалення діяльності та отримання за допомогою цього конкурентних переваг. Але для цього потрібні системи, які дозволяли б виконувати не тільки найпростіші дії над даними: підраховувати суми, середні, максимальні і мінімальні значення. З'явилася потреба в інформаційних системах, які дозволяли:

- проводити глибоку аналітичну обробку;

- вирішення різних завдань;

- пошук прихованих структур і закономірностей в масивах даних;

- генерація різних висновків та правил;

- стратегічне і оперативне планування;

- прийняття рішень і прогнозування їх наслідків.

Розуміння переваг, які здатний дати інтелектуальний аналіз, призвело до появи нового класу систем: інформаційних систем підтримки прийняття рішень (інформаційних СППР). Такі системи орієнтовані на аналітичну обробку даних з метою отримання знань, необхідних для розробки рішень в галузі управління. Додатковим стимулом вдосконалення цих систем стали такі фактори, як:

- зниження вартості високопродуктивних комп'ютерів і витрат на зберігання великих обсягів інформації;

- поява можливості обробки великих масивів даних і розвиток відповідних математичних методів.

Узагальнена структурна схема інформаційної СППР що призначена для обробки даних представлена на рис. 1.3.

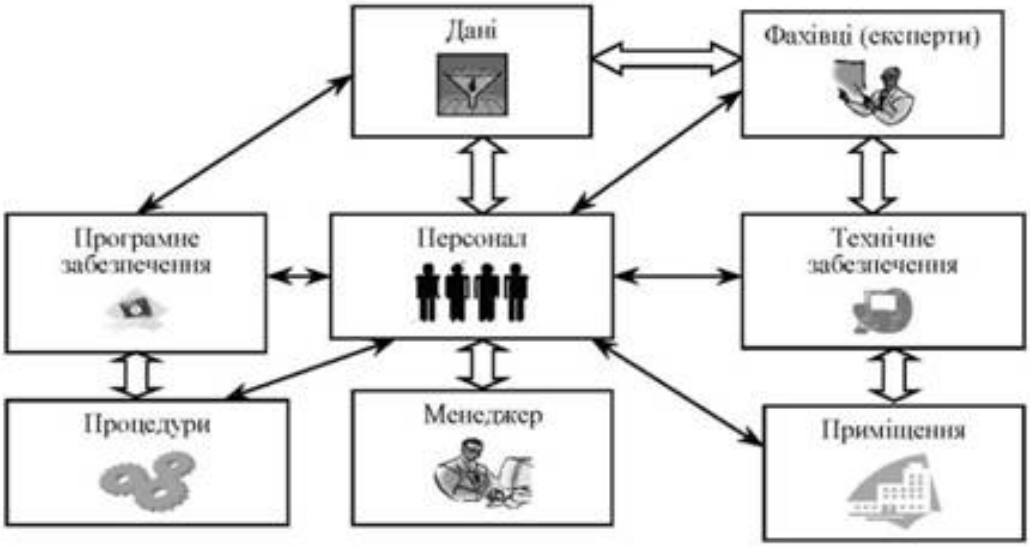


Рисунок 1.3 – Структура інформаційної системи СППР з обробки даних

В основі роботи з такою СППР лежать запити, з якими до неї звертається користувач: особа, яка приймає рішення (ОПР) - менеджер, експерт чи аналітик.

При цьому запити, що допустимі в традиційних системах оперативної обробки даних, дуже примітивні. Наприклад, для банку це може бути запит типу «Скільки грошей на рахунку клієнта» Або «Скільки грошей клієнт витратив за останній місяць». Очевидно, що цінність інформації, отриманої за допомогою подібного запиту, невелика. У той же час аналітична система може відповісти на набагато більш складні запити. Наприклад: «Визначити середній час між виставленням та сплатою рахунку для кожної категорії клієнтів».

В процесі розробки систем аналізу інформації та методології їх застосування виявляється, що для ефективного функціонування такі системи повинні бути організовані трохи іншим способом, ніж той, який застосовується в OLTP-системах. Це обумовлено наступними причинами.

Для виконання складних аналітичних запитів необхідна обробка великих масивів даних з різноманітних джерел. Для виконання запитів, що пов'язані з аналізом тенденцій, прогнозуванням протяжних в часі процесів, необхідні історичні дані, накопичені за досить тривалий період, що не забезпечується звичайними OLTP-системами.

Дані, що використовуються для цілей аналізу і обслуговування аналітичних запитів, відрізняються від звичайних даних що існують в OLTP-системах. При аналітичної обробки перевагу надають не детальним даними, а узагальненим (агрегованих). Очевидно, що для аналізу продажів великого супермаркету інтерес представляє не інформація про окремі покупки, а скоріше відомості про продажі протягом певних часових інтервалів (наприклад, тижня чи місяця).

У зв'язку з цим можна виділити ряд принципових відмінностей СППР і OLTP-систем. Ці відмінності представлені в табл. 1.1.

Як видно з табл. 1.1, вимоги щодо СППР і OLTP-систем істотно відрізняються. Тому, в СППР використовуються спеціалізовані бази даних, які називаються сховищами даних. Сховища даних орієнтовані на аналітичну обробку та задовольняють вимогам, що пред'являються до систем підтримки прийняття рішень. Розглянемо програмну реалізацію OLTP технології. Спочатку, Microsoft Analysis Services (Microsoft SQL Server) не був оригінальною розробкою цієї

корпорації. Технологія, що розроблена в 1996 році, була куплена у компанії Panorama Software Systems.

Існує декілька відомих програмних продуктів Microsoft, що створені на основі OLAP обробки даних.

Таблиця 1.1

Порівняльна характеристика OLAP і OLTP-систем

Властивість	OLTP-система	OLAP-система
1	2	3
Цілі використання даних	Швидкий пошук, найпростіші алгоритми обробки	Аналітична обробка з метою пошуку прихованих закономірностей, побудови прогнозів і моделей та інше
Рівень узагальнення (деталізації) даних	Деталізовані	Як деталізовані, так і узагальнені (агреговані)
Вимоги до якості даних	Можливі некоректні дані (помилки реєстрації, введення та інше)	Помилки в даних не допускаються, оскільки можуть призвести до некоректної роботи аналітичних алгоритмів
Формат зберігання даних	Дані можуть зберігатися в різних форматах в залежності від програми, в якому вони були створені	Дані зберігаються і обробляються в єдиному форматі
Час зберігання даних	Як правило, не більше року (в межах звітного періоду)	Роки, десятиліття

Продовження таблиці 1.1

Властивість	OLTP-система	СППР
1	2	3
Зміна даних	Дані можуть додаватися, змінюватися і вилучатися	Допускається тільки поповнення; раніше додані дані змінюватися не повинні, що дозволяє забезпечити їх хронологію
Періодичність оновлення	Часто, але в невеликих обсягах	Рідко, але в більших обсягах
Доступ до даних	Повинен бути забезпечений доступ до всіх поточних (оперативних) даними	Повинен бути забезпечений доступ до історичних (тобто накопиченим за досить тривалий період часу) даними з дотриманням їх хронології
Характер виконуваних запитів	Стандартні, налаштовані заздалегідь	Нерегламентовані, що формуються аналітиком «на льоту» в залежності від необхідного аналізу
Час виконання запиту	Декілька секунд	Кілька хвилин

Також відомо, що ринок OLAP рішень доволі поширився за рахунок вдосконаленої архітектури зберігання даних із застосуванням ROLAP / MOLAP / HOLAP, а також демпінгових цін і агресивного маркетингу. Analysis Services 2005 виграє в ціні у конкурентів, крім того, корпорація спочатку вбудовували зачатки OLAP-системи в MS Office, а в Excel 2000 спеціальний додаток PivotTables

допомагало інтегруватися у вигляді OLAP-клієнта до Microsoft Analytic Services.

У Microsoft SQL Server 2005 присутня єдина модель OLAP і реляційної звітності. Моделі можна будувати з довільних розрізах куба. При виборі одного виміру інші, пов'язані з ним, можна розглядати як вкладені таблиці, причому до кожного допускається застосування незалежних розрізів. Продукт дозволяє застосовувати OLAP-модель до глобального куба, ділити його на групи, створювати кластери в режимі реального часу. Сервер дозволяє постійно оновлювати куби, зводячи різницю між реляційними і багатовимірними базами до мінімуму.

Microsoft завжди намагалася залучати інших розробників до створення інструментарію, тому окремо варто згадати продукти, інтегровані з Microsoft Analysis Services. Один з них - ProClarity Analytics Platform, OLAP-клієнт від компанії Proclarity (колишньої Knosys), що дозволяє взаємодіяти з іншими MS додатками, такими як Excel, PowerPoint, Outlook Digital Dashboards і з web-додатками. Пакет дає можливість віддаленої роботи з базами даних. Серед користувачів продуктів Proclarity такі компанії, як Compaq, Ericsson, Hewlett-Packard, L'Oreal, Pennzoil, Siemens, а також Swiss Stock Exchange.

Ще одна платформа для швидкого розгортання систем аналітики та звітності інтегруються з Microsoft Analysis Services, - Panorama NovaView компанії Panorama Software. Цей OLAP-клієнт забезпечує зв'язку між Microsoft Reporting Services, Microsoft Excel і Microsoft Share Point Portal Server. Як операційні джерел в Panorama NovaView можуть бути використані будь-які бази даних, у тому числі Oracle, Teradata, DB2 і Microsoft.

Другим конкурентом на ринку OLAP-систем є компанія Hyperion Solutions Corporation, яка пропонує одразу кілька рішень. Hyperion Performance Suite включає в себе два програмних продукту: Hyperion Intelligence, систему формування запитів і аналізу отриманої інформації, а також Hyperion SQR, яка формує презентаційні звіти. Рішення забезпечує взаємодію з реляційними, багатовимірними і операційними базами даних усіх відомих розробників, включаючи Oracle, DB2, MS SQL Server та інші. Звіти можна формувати

практично у всіх форматах, включаючи XML, PDF, HTML, текст, PostScript, і т.д. Ще один OLAP-сервер від цього виробника, Hyperion Essbase, що використовує технологію HOLAP, підтримує обсяги даних, що вимірюються в терабайтах, доступ до яких можуть отримати тисячі користувачів одночасно. Середній час виконання запиту при цьому, за твердженням розробників, не перевищує одну секунду.

Важливо, що продукти Hyperion Solutions в тій чи іншій мірі містять технології, розроблені одинадцятьма компаніями в різний час. Тому при створенні нових версій своїх рішень, корпорація приділила особливу увагу більш глибокої інтеграції компонентів. Крім того, був розроблений більш дружній інтерфейс.

Дещо поступається OLAP-гігантам на ринку компанія Cognos, в числі найбільш популярних її рішень у цій сфері - Cognos Enterprise Planning і Cognos PowerPlay. Перше є інструментом, що дозволяє вести деталізований план доходів і витрат, здійснювати управління активами з урахуванням фінансових ризиків і погоджувати діяльність всіх підрозділів компанії. Це рішення, випущене після придбання Cognos в 2003 році компанії Adaytum, забезпечує обмін інформацією між різноманітними джерелами і одержувачами. Cognos PowerPlay - засіб OLAP-аналізу, що дозволяє будувати багатовимірний куб, що містить понад 2 мільярдів різних даних і категорій.

Користувачі можуть отримати доступ до кубів, використовуючи веб-клієнти, засоби операційних систем або Excel.

Тому, в останніх версіях своїх продуктів Cognos перейшов на нову платформу. Раніше основний акцент компанія робила саме на OLAP-звітах, зараз користувач може працювати одночасно з реляційними базами, а також з спрощеним ROLAP і повним MOLAP серверами. У цьому плані Cognos краще справляється з OLAP, ніж інші конкуренти, такі як Business Objects. При цьому нова версія надає просунуті можливості та інструменти OLAP-сервера для користувачів.

В даний час однозначного визначення сховища даних не існує, через те що розроблено велику кількість різних архітектур і технологій сховища даних. Самі

сховища використовуються для вирішення найрізноманітніших завдань. Кожен автор вкладає в це поняття своє бачення питання. Узагальнюючи вимоги, що пред'являються до СППР, можна дати наступне визначення сховища даних, яке не претендує на повноту і однозначність, але дозволяє зрозуміти основну ідею.

Тому, сховище даних - це різновид систем зберігання, орієнтована на підтримку процесу аналізу даних, що забезпечує цілісність, несуперечність і хронологію даних, а також високу швидкість виконання аналітичних запитів.

Найважливішим елементом сховища даних є семантичний шар - механізм, що дозволяє аналітику оперувати даними за допомогою бізнес-термінів предметної області. Семантичний шар дає користувачеві можливість зосередитися на аналізі і не замислюватися про механізми отримання даних.

Типові сховища даних істотно відрізняється від звичайних систем зберігання даних. Головною відмінністю є цілі використання. Наприклад, реєстрація продажів і виписка відповідних документів - завдання рівня OLTP-систем, що використовують звичайні реляційні системи управління базами даних. Аналіз динаміки продажів і попиту за кілька років, що дозволяє виробити стратегію розвитку фірми і спланувати роботу з постачальниками і клієнтами, найзручніше виконувати за підтримки сховища даних.

Інша важлива відмінність полягає в динаміці зміни даних. Бази даних в OLTP-системах характеризуються дуже високою динамікою зміни записів через повсякденної роботи великої кількості користувачів. Що стосується сховища даних, то інформаія зі сховища не видаляється, а поповнення відбувається відповідно до визначеного регламенту: раз на годину, день, тиждень, в певний час.

До основних вимог до сховища даних належать наступні . Щоб сховища даних виконувало функції, відповідні його основному завданню - підтримки процесу аналізу даних. Отже, до основних вимог концепції сховища даних належать наступні:

- висока швидкість отримання даних зі сховища;
- автоматична підтримка внутрішньої несуперечності даних;
- можливість отримання і порівняння різних зрізів даних;

- наявність зручних засобів для перегляду даних в сховищі;

- забезпечення цілісності та достовірності даних, що зберігаються.

Для впровадження перерахованих вимог з метою побудови і роботи сховища даних, як правило, використовується не один додаток, а система, в яку входить кілька програмних продуктів. Одні з них представляють собою власне систему зберігання даних, інші - засоби їх перегляду, вилучення, завантаження та інше.

Таким чином, в останні десятиліття технологія сховища даних стрімко розвивається. Десятки компаній пропонують на ринку свої рішення в області сховища даних. Тому, багато організацій вже використовують це потужний засіб підтримки аналітичних проектів.

1.3. Постановка задачі розробки системи

В роботі необхідно провести аналіз обробки даних за допомогою поширених OLTP та OLAP методів. Аналіз обробки даних за допомогою OLAP-кубів є найбільш ефективним та в загальному випадку зводиться до пошуку оптимальної структури аналітичної системи, яка забезпечила б значне скорочення часу на обробку даних і формування звітів та унікальну гнучкість в побудові звітності.

Тому, в процесі аналізу основних проблем аналізу обробки даних може бути поставлена задача розробити єдину систему звітності, що надала б можливість оптимізувати бізнес-потоки і при цьому дозволяла відмовитися від механічної ручної роботи. При цьому можуть бути сформовані основні вимоги щодо аналітичної системи з використання відомих методів обробки даних яка полягає у:

- можливості отримати будь-який аналітичний звіт без виклику на допомогу з обробки даних програміста;

- скорочення часу отримання звіту в межах секунд за будь-який період;

- існування наскрізного аналізу даних за великий період;

- наявність єдиного сховища даних і системи аналізу інформації.

Тому, система з обробки даних можує являти собою складний програмний продукт, який включає:

- ядро - систему управління базами даних (СКБД);

- механізми імпорту, експорту та перетворення даних (ETL, або data staging);

- засоби проектування систем з обробки та сховища даних;

- засоби роботи з репозиторієм метаданих;

- засоби оперативного аналітики (OLAP-засоби).

Процедура роботи системи з обробки даних може використовувати такий алгоритм роботи. Спочатку дані збираються із застосуванням технології Data Warehouse (DWH), яка відповідає за структуризацію і очищення даних. Потім, на другому етапі, ці дані готуються для обробки даних та їх оперативного аналізу за допомогою технології OLAP. Далі відбувається процес агрегації і аналіз інформації, що міститься в DWH- і OLTP-системі.

Нарешті, на третьому етапі дані передаються зацікавленому в них аналітику через Інтернет, який виступає засобом дистрибуції звітів та інформації для обробки даних та OLAP-аналізу.

Зазвичай існують суперечки серед розробників автоматизованих систем щодо відмінностей між обробкою та сховищем даних, базою даних і операційною системою. Серед цих відмінностей виділяється тема функціонального характеру системи сховища даних. Будь-яка операційна система або база даних орієнтована на операції з обробки даних. Тоді як в сховищі важливі самі інформаційні об'єкти.

В операційній системі зберігаються тільки найактуальніші дані, тоді як сховище розраховане на весь спектр історичної інформації про компанії. Ці історичні дані і є її основною інформацією. Зберігати їх просто необхідно, оскільки тільки з їх допомогою можна зробити повноцінний аналіз. Дані в сховищі збираються і зберігаються у вигляді єдиної таблиці фактів, інформацію з якої можна розглядати під різними кутами.

У банках споживачами інформації, що лежить в сховище, можуть бути не тільки аналітики, але і бухгалтерія, правління банку, управління кадрів та інші департаменти. А питання, які вирішуються за допомогою аналізу даних зі сховища, можуть бути самих різних рівнів, починаючи від складання консолідованої звітності і закінчуючи бюджетуванням і аналізом діяльності

підприємства.

Аналіз клієнтської бази і банківських продуктів, вироблений за допомогою оцінки її структури і динаміки зміни, оцінки якості активів і пасивів і бізнес-операцій в компанії, а також оцінки прибутковості клієнтів. Це дозволяє вчасно помічати зміни тенденцій і міняти стратегічні рішення.

Вирішення цих проблем полягає у використанні OLAP-кубів (оперативний багатовимірний аналіз) де основним ключовим елементом є бази даних Online Analytical Processing (OLAP). Бази даних OLAP полегшують створення запитів для обробки та аналізу інформації.

OLAP — це технологія обробки баз даних, що оптимізована для створення запитів та звітів. Вихідними даними для OLAP є бази даних Online Transactional Processing (OLTP), які часто обробляються та зберігаються в інформаційних сховищах.

Дані OLAP створюються на основі цих накопичених даних та впорядковуються в структурах, що дозволяють проводити складний аналітичний аналіз. Дані OLAP впорядковані ієрархічно та зберігаються в кубах, а не в таблицях. Це складна технологія, яка використовує багатовимірні структури, що надають швидкий доступ до даних для аналізу.

Така організація полегшує відображення зведень високого рівня у звітах зведених таблиць або зведених діаграм, наприклад, обсягів продажу в країні або в регіоні, а також відображення відомостей про місця, де рівень продаж особливо високий або низький.

Бази даних OLAP створені для прискорення обробки даних. Оскільки підрахунок сумарних значень здійснює сервер OLAP, а не електронні таблиці (наприкад Microsoft Office Excel), то менше даних потрібно надсилати дані до таблиць Excel під час створення або змінення звіту. Такий підхід надає можливість працювати з набагато більшою кількістю вихідних даних, ніж у разі, якщо б дані було оформлено в традиційному стилі, коли Excel завантажує усі окремі звіти, а тоді підраховує сумарні значення.

Таким чином, для обробки бази даних на основі OLAP методів існують два

основних типа даних:

1) показники, які є числовими даними, кількісні показники та середні значення даних, які використовуються для прийняття рішень;

2) виміри, які є категоріями, що використовуються для організації цих показників.

Для обробки бази даних на основі методів OLAP з’являється можливість організувати дані за багатьма рівнями деталізації з використанням тих самих категорій, за допомогою яких здійснюється аналіз даних.

Куб – це структура з обробки даних, яка об’єднує показники за рівнями та ієрархіями кожного з вимірів, які потрібно проаналізувати. Куби поєднують кілька вимірів — наприклад, час, розташування та лінію продуктів — з сумарними даними, такими як показники продажів або резерву.

OLAP-аналіз даних є важливою необхідністю обробки та управління даних у сучасних підприємств. Такий аналіз дозволяє керівникам узагальнювати різносторонню інформацію про роботу компанії й ухвалювати обгрунтовані тактичні й стратегічні рішення.

Для вирішення вищезазначених проблем в даній роботі обирається така інформаційна платформа для обробки даних на основі методів OLAP:

1) сервер OLAP (на прикладі MS Analyses Services 2005/2008);

2) реляційна база даних що сумісна з обраним сервером OLAP (наприклад, Microsoft SQL Server 2005/2008);

3) модуль обробки даних облікової системи що розташований на сервері OLAP;

4) типові структури OLAP-кубів що дозволяють ефективно обробляти дані;

5) клієнт OLAP для роботи менеджерів з обробки даних (на прикладі Microsoft reporting Services 2005/2008).

1.4. Висновки

В першому розділі проведено аналіз стану проблеми обробки даних де визначено два різних підходи щодо побудови систем які основані на OLTP та

OLAP методах. Перший підхід являє собою обробку даних на основі транзакцій в реальному часі та переважно використовується в системах управління базами даних. Цей підхід був розрахований на ефективний збір даних в реальному часі. Другий підхід являє собою аналітичну обробку даних в реальному часі, але націлене саме на вибірку та обробку інформації максимально зручним та ефективним способом. Це дозволяє на основі останнього підходу поширено використовувати у бізнесі, де щодня доводиться приймати безліч рішень для виробничих, маркетингових і кадрових рішень.

Тому, визначення ефективних рішень в галузі обробки даних має відбуватися на всіх рівнях управління організацією. Це в кінцевому підсумку призводить до успіху всієї організації в цілому.

Завдяки детальному структуруванні інформації OLAP-куби дозволяють оперативно здійснювати аналіз та обробку даних, формувати різні звіти у визначених розрізах і з довільною глибиною деталізації. Звіти можуть створюватися аналітиками, менеджерами, фінансистами, керівниками підрозділів в інтерактивному режимі. Це дозволяє швидко отримати відповіді, на які виникають щодня питання, і прийняти правильне рішення. При цьому співробітникам, для створення звітів не потрібно вдаватися до послуг програмістів, на що зазвичай йде чимало часу. Тому, часто в компаніях існує кілька інформаційних систем що потребують ефективної обробки даних: системи складського обліку, бухгалтерські системи, ERP системи для автоматизації окремих виробничих процесів, системи збору звітності з підрозділів компанії, а також безліч файлів, які знаходиться на комп'ютерах співробітників.

Також в першому розділі визначено, що системи з обробки даних зазвичай мають засоби надання користувачеві агрегатних даних для різних вибірок з вихідного набору в зручному для сприйняття і аналізу вигляді. Як правило, такі агрегатні функції утворюють багатовимірний (і, отже, нереляційний) набір даних. Така структура необхідна для швидкої обробки даних та називається гіперкубом або метакубом. Це відбувається внаслідок того, що такі куби містять параметри, а осередки - залежні від них агрегатні дані. Причому оброблятися та зберігатися

такі дані можуть і в реляційних таблицях, але в даному випадку мова йде про логічної організації даних, а не про фізичну реалізації їх зберігання.

Уздовж кожної осі метакубів, дані можуть бути організовані у вигляді ієрархії, що представляє різні рівні їх деталізації. Завдяки такій моделі з обробки даних користувачі можуть формувати складні запити, генерувати звіти, отримувати підмножини даних.

Під час проведення порівняльного аналіз аналогів в цьому розділі були визначені характерні риси процесів, які властиві та притаманні в тій чи іншій мірі всім OLTP-системам. Особливістю таких систем є те, що запити та звіти під час обробки даних є повністю регламентовані.

В роботі було розглянуто приклад з обробкою даних в галузі транспортних перевезень. Такі дослідження можуть бути певною реакцією на вхідну інформації що постійно змінюється. Про те, останнім часом у багатьох пунктах продажів, наприклад білетів, почастішали випадки нестачі квитків на певні маршрути, що дозволяє зробити припущення про доцільність організації додаткових рейсів на транспортні засоби. Однак для проведення таких досліджень необхідні як мінімум три речі. По-перше, потрібні дані про продажі квитків за досить тривалий період (кілька місяців або років). По-друге, дані не повинні містити протиріч, пропусків, аномальних значень та інших чинників, які не дозволять виконати коректний аналіз. По-третє, необхідна додаткова інформація про бізнес-середовищі: про конкурентів, ринкові тенденції, ціни на паливо та інше. Тому, типова OLTP-система не може забезпечити нічого з перерахованого. Саме з розумінням цих проблем приходить усвідомлення необхідності використання більш розвинених системи з обробки та зберігання даних, що орієнтовані на аналіз.

Під час постановки задачі було проведено аналіз обробки даних за допомогою OLAP-кубів, що в загальному випадку зводиться до пошуку оптимальної структури аналітичної системи, яка забезпечила б значне скорочення часу на обробку даних і формування звітів та унікальну гнучкість в побудові звітності. В процесі висвітлення основних проблем аналізу оперативних даних може бути поставлена задача розробити єдину систему звітності, що надала б

можливість оптимізувати бізнес-потоки і при цьому дозволяла відмовитися від механічної ручної обробки даних. При цьому можуть бути сформовані основні вимоги щодо аналітичної системи яка полягає у: можливості отримати будь-який аналітичний звіт без виклику програміста; скорочені часу отримання звіту в межах секунд за будь-який період; існування наскрізного аналізу даних за великий період; наявності єдиного сховища даних і системи аналізу інформації. Тому, такі сховище даних можуть являти собою складну систему, яка включає: ядро - систему управління базами даних; ресурси, що виконують імпорт, експорт та перетворення даних; засоби проектування обробки та сховища даних; засоби роботи з репозиторієм метаданих та засоби оперативного аналітики (OLAP-засоби).

РОЗДІЛ 2
ДОСЛІДЖЕННЯ МЕТОДІВ РОБОТИ СИСТЕМ З ОБРОБКИ ДАНИХ

2.1. Вимоги щодо дослідження методів роботи систем з обробки даних

На сьогоднішній день існують досить підвищені вимоги щодо роботи систем з обробки даних на підприємствах. Це відбувається внаслідок того, що інформаційні системи масштабу підприємства, як правило, містять додатки, що призначені для швидкого перетворення даних, комплексного багатовимірного аналізу даних, їх динаміки, тенденцій та інше. Такий аналіз в кінцевому підсумку покликаний сприяти прийняттю рішень. Часто, ці системи називаються системами підтримки прийняття рішень на основі сучасних методів обробки даних.

Тому, прийняти будь-яке управлінське рішення неможливо без володіння необхідною для цього інформацією, зазвичай кількісної. Для цього необхідне створення швидких методів обробки та сховищ даних (Data warehouses), тобто процес збору, відсіювання і попередньої обробки даних з метою надання результуючої інформації користувачам для статистичного аналізу та створення аналітичних звітів. В результаті цього, концепція обробки та сховища даних має такі основні вимоги до наявних інформаційних ресурсів:

- підтримка високої швидкості отримання та обробки даних зі сховища;
- підтримка внутрішньої несуперечності у структурах даних;
- можливість отримання і порівняння різні зрізи даних (slice and dice);
- наявність зручних утиліт перегляду даних в сховищі;
- повнота і достовірність даних, що зберігаються;
- підтримка якісного процесу поповнення даних.

Задовольняти всім перерахованим вимогам в рамках одного і того ж програмного продукту часто не вдається. Тому, для реалізації систем з обробки та сховищ даних зазвичай використовується кілька продуктів, одні з яких представляють собою власне засоби зберігання даних, інші - ресурси їх вилучення та перегляду, треті - ресурси їх поповнення.

Типове сховище даних, як правило, відрізняється від звичайної реляційної бази даних. По-перше, звичайні бази даних призначені для того, щоб допомогти

користувачам виконувати повсякденну роботу, тоді як сховища даних призначені для прийняття рішень. Наприклад, продаж товару і виписування рахунку здійснюються з використанням бази даних, призначеної для обробки транзакцій, а аналіз динаміки продажів за декілька років, що дозволяє планувати роботу з постачальниками - за допомогою сховища даних.

По-друге, звичайні бази даних постійно змінюються в процесі роботи користувачів, а сховище даних відносно стабільне: дані в ньому зазвичай оновлюються за розкладом (наприклад, щотижня, щодня або щогодини - залежно від потреб). Тому, процес поповнення являє собою просте додавання нових даних за певний період часу без зміни колишньої інформації, що вже перебуває в сховище.

По-третє, звичайні бази даних найчастіше є джерелом обробки даних, що потрапляють в сховище. Крім того, сховище може поповнюватися за рахунок зовнішніх джерел, наприклад статистичних звітів.

Кінцевою метою використання технології OLAP є обробка, аналіз даних і представлення результатів цього аналізу у вигляді, що зручне для сприйняття і прийняття рішень. Основна ідея OLAP полягає в побудові багатовимірних кубів, які будуть доступні для користувача запитів. Однак вихідні дані для побудови OLAP-кубів зазвичай зберігаються в реляційних базах даних. Нерідко це спеціалізовані реляційні бази даних, звані також сховищами даних (Data Warehouse). На відміну від так званих оперативних баз даних, з якими працюють програми, що модифікують дані, сховища даних призначені виключно для обробки і аналізу інформації. Тому, проектуються вони таким чином, щоб час виконання запитів до них було мінімальним. Зазвичай дані копіюються в сховище з оперативних баз даних згідно з визначеним розкладом.

Типова структура обробки та сховища даних істотно відрізняється від структури звичайної реляційної бази даних. Як правило, ця структура денормалізована так як це дозволяє підвищити швидкість виконання запитів. Тому, може допускати надмірність даних.

Системи підтримки прийняття рішень зазвичай мають засоби надання

користувачам агрегатних даних для різних вибірок з вихідного набору в зручному для обробки, сприйняття і аналізу вигляді. Як правило, такі агрегатні функції утворюють багатовимірний (і, отже, не реляційних) набір даних. Це нерідко називається гіперкубом або метакубом, осі якого містять параметри, а осередки - залежні від них агрегатні дані. Причому, зберігатися такі дані можуть і в реляційних таблицях, але в даному випадку мова йдеться про логічної організації даних, а не про фізичну реалізації їх зберігання.

Уздовж кожної осі дані можуть бути організовані у вигляді ієрархії, що представляють різні рівні їх деталізації. Завдяки такій моделі даних користувачі можуть обробляти та формулювати складні запити, генерувати звіти, отримувати підмножини даних.

Таким чином, технологія комплексного багатовимірного аналізу даних отримала назву OLAP (On-Line Analytical Processing). OLAP - це ключовий компонент організації сховищ даних. Концепція OLAP визначає такі вимоги до програмних додатків для багатовимірного аналізу:

- надання користувачу результатів аналізу за прийнятний час (зазвичай не більше 5 секунд), нехай навіть ціною менш детального аналізу;

- можливість здійснення будь-якого логічного і статистичного аналізу, характерного для цього додатка, його збереження в доступному для кінцевого користувача вигляді;

- розрахування на доступ багатьох користувачів до даних з підтримкою відповідних механізмів блокувань і засобів авторизованого доступу;

- багатовимірне концептуальне представлення даних, включаючи повну підтримку для ієрархій і множинних ієрархій;

- можливість звертатися до будь-якої потрібної інформації незалежно від її обсягу і місця зберігання.

Також, слід зазначити, що OLAP-функціональність може бути реалізована різними способами, починаючи з найпростіших засобів обробки та аналізу даних в офісних додатках і закінчуючи розподіленими аналітичними системами, заснованими на серверних продуктах.

2.2. Дослідження роботи загальної структури систем з обробки даних

Під час дослідження роботи загальної структури систем з обробки даних кінцевою метою є аналіз інформації і представлення результатів цього аналізу у вигляді, зручному для сприйняття і прийняття рішень.

Основна ідея використання OLAP технології полягає в побудові багатовимірних кубів, які будуть доступні для користувача запитів. Однак вихідні дані для побудови OLAP-кубів зазвичай зберігаються в реляційних базах даних. Нерідко це спеціалізовані реляційні бази даних, звані також сховищами даних (Data Warehouse). На відміну від так званих оперативних баз даних, з якими працюють програми, що модифікують дані, сховища даних призначені виключно для обробки і аналізу інформації. Тому проектуються вони таким чином, щоб час виконання запитів до них було мінімальним. Зазвичай дані копіюються в сховище з оперативних баз даних згідно з визначеним розкладом.

Типова структура обробки та сховища даних істотно відрізняється від структури звичайної реляційної СУБД. Як правило, ця структура денормалізована (це дозволяє підвищити швидкість виконання запитів). Тому така структура може допускати надмірність даних.

Для подальших прикладів визначимо структуру сховища даних як TestDatabase. Її структура даних приведена на рис. 2.1.

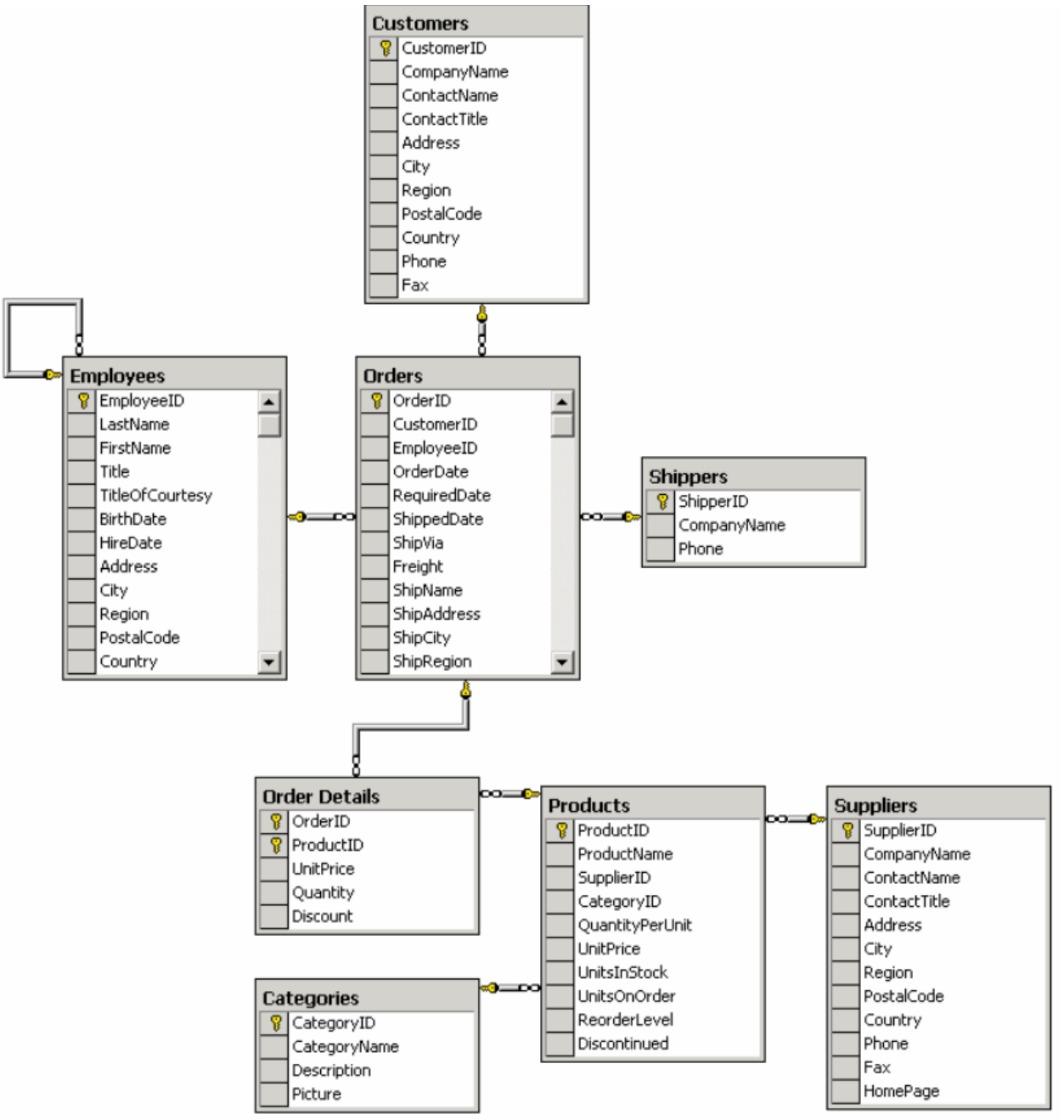


Рисунок 2.1 – Дослідження структури систем обробки даних на прикладі TestDatabase

Основними складовими структури обробки та сховищ даних є таблиця фактів (fact table) і таблиці вимірювань (dimension tables).

Таблиця фактів є основною таблицею обробки та сховища даних. Як правило, вона містить відомості про об'єкти або події, сукупність яких буде надалі аналізуватися. Зазвичай говорять про чотирьох найбільш часто зустрічаються типах фактів. До них відносяться наступні:

- факти, що пов'язані з транзакціями (Transaction facts). Вони засновані на

окремих подіях (типовими прикладами яких є телефонний дзвінок або зняття грошей з рахунку за допомогою банкомату);

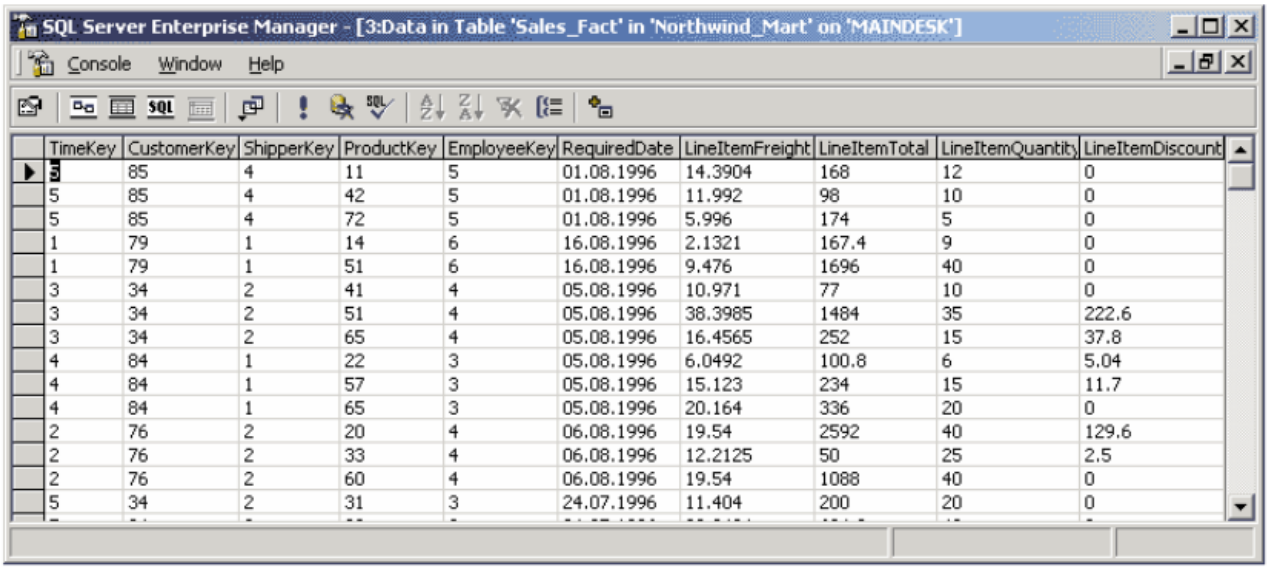
- факти, що пов'язані з «миттєвими знімками» (Snapshot facts). Засновані на стан об'єкта (наприклад, банківського рахунку) в певні моменти часу, наприклад на кінець дня чи місяця;

- факти, що пов'язані з елементами документа (Line-item facts). Засновані на тому чи іншому документі (наприклад, рахунку за товар або послуги) і містять детальну інформацію про елементи цього документа (наприклад, кількості, ціні, відсотку знижки);

- факти, що пов'язані з подіями або станом об'єкта (Event or state facts). Представляють виникнення події без подробиць про нього (наприклад, просто факт продажу або факт відсутності такої без інших подробиць).

Для прикладу розглянемо факти, що пов'язані з обробкою елементів документа (в даному випадку рахунки, виставленого за товар). Таблиця даних, як правило, містить унікальний складовий ключ, який об'єднує первинні ключі таблиць вимірів. Найчастіше це цілочисельні значення або значення типу «дата/час» - адже таблиця фактів може містити сотні тисяч або навіть мільйони записів. При цьому як ключові, так і деякі не ключових поля повинні відповідати майбутнім вимірам OLAP-куба.

Крім цього, таблиця даних містить одне або кілька числових полів, на підставі яких в подальшому будуть отримані агрегатні дані. Приклад таблиці фактів, яка може бути побудована на основі сховища даних TestDatabase, наведено на рис. 2.2.



TimeKey	CustomerKey	ShipperKey	ProductKey	EmployeeKey	RequiredDate	LineItemFreight	LineItemTotal	LineItemQuantity	LineItemDiscount
3	85	4	11	5	01.08.1996	14.3904	168	12	0
5	85	4	42	5	01.08.1996	11.992	98	10	0
5	85	4	72	5	01.08.1996	5.996	174	5	0
1	79	1	14	6	16.08.1996	2.1321	167.4	9	0
1	79	1	51	6	16.08.1996	9.476	1696	40	0
3	34	2	41	4	05.08.1996	10.971	77	10	0
3	34	2	51	4	05.08.1996	38.3985	1484	35	222.6
3	34	2	65	4	05.08.1996	16.4565	252	15	37.8
4	84	1	22	3	05.08.1996	6.0492	100.8	6	5.04
4	84	1	57	3	05.08.1996	15.123	234	15	11.7
4	84	1	65	3	05.08.1996	20.164	336	20	0
2	76	2	20	4	06.08.1996	19.54	2592	40	129.6
2	76	2	33	4	06.08.1996	12.2125	50	25	2.5
2	76	2	60	4	06.08.1996	19.54	1088	40	0
5	34	2	31	3	24.07.1996	11.404	200	20	0

Рисунок 2.2 – Приклад таблиці даних структури сховища TestDatabase

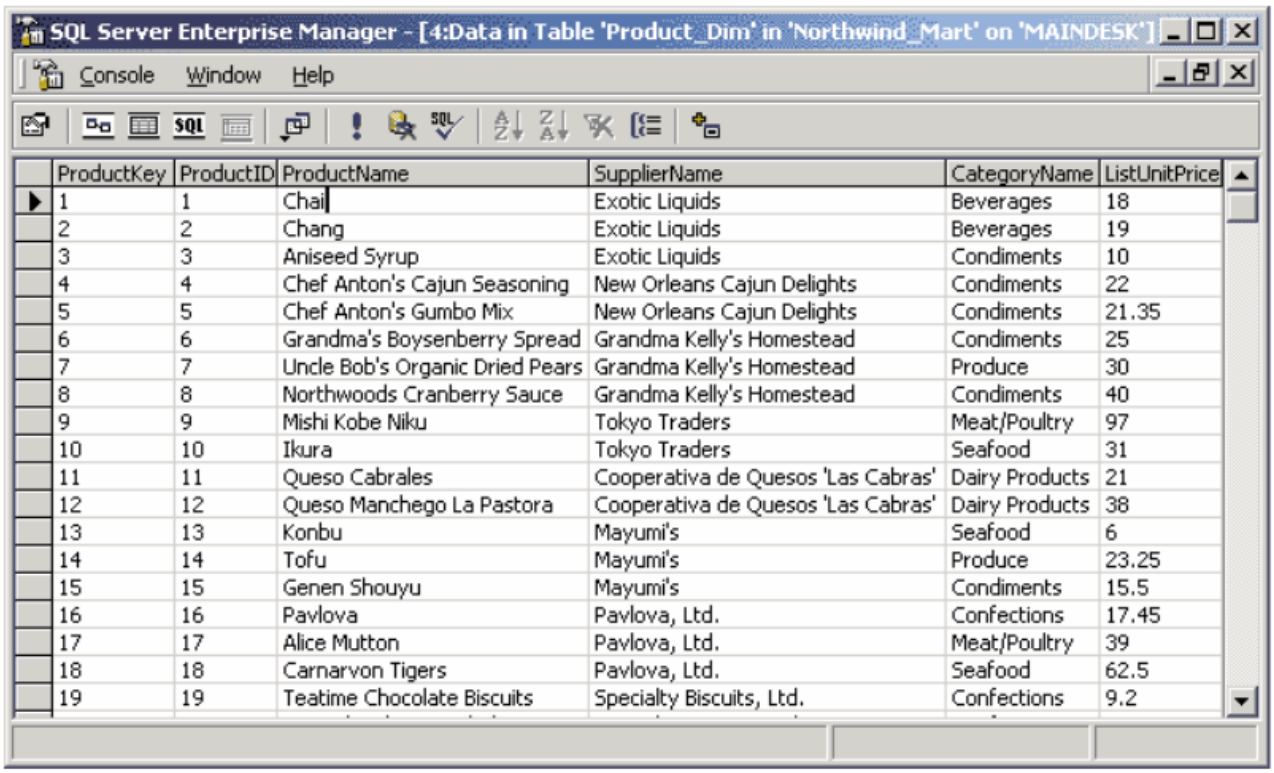
В даному прикладі (рис. 2.2) вимірам майбутнього куба відповідають перші шість полів, а агрегатним даними - останні чотири. Відзначимо, що для багатовимірного аналізу придатні таблиці фактів, що містять якомога докладніші дані (тобто відповідні членам нижніх рівнів ієрархії відповідних вимірювань). В даному випадку краще взяти за основу факти продажу товарів окремим замовникам, а не суми продажів для різних країн - останні все одно будут обчислені OLAP-засобом. Виняток можна зробити, мабуть, тільки для клієнтських OLAP-засобів, оскільки в силу ряду обмежень вони не можуть маніпулювати великими обсягами даних.

Необхідно також відзначити, що в таблиці даних немає ніяких відомостей про те, як групувати записи при обчисленні агрегатних даних. Наприклад, в ній є ідентифікатори продуктів або клієнтів, але відсутня інформація про те, до якої категорії відноситься даний продукт або в якому місті знаходиться даний клієнт. Ці відомості, в подальшому використовуються для побудови ієрархій в вимірах куба, містяться в таблицях вимірів.

Таблиці вимірювань містять незмінні або рідко змінювані дані. У переважній більшості випадків ці дані представляють собою по одному запису для кожного члена нижнього рівня ієрархії у вимірі. Таблиці вимірювань також містять як мінімум одне описову поле (зазвичай з ім'ям члена вимірювання) і, як правило, целочисленное ключове поле для однозначної ідентифікації члена

вимірювання. Якщо майбутнє вимір, засноване на даній таблиці вимірювань, містить ієрархію, то таблиця вимірювань також може містити поля, що вказують на «батька» даного члена в цій ієрархії. Нерідко (але не завжди) таблиця вимірювань може містити і поля, що вказують на «прабатьків», і інших «предків» в цій ієрархії (це зазвичай характерно для збалансованих ієрархій), а також додаткові атрибути членів вимірювань, що містилися в початковій оперативної базі даних (наприклад, адреси та телефони клієнтів).

Кожна таблиця вимірювань повинна знаходитися у відношенні «один до багатьох» з таблицею фактів. Відзначимо, що швидкість росту таблиць вимірів повинна бути незначною в порівнянні зі швидкістю зростання таблиці фактів; наприклад, додавання нового запису в таблицю вимірювань, що характеризує товари, проводиться тільки при появі нового товару, що не продавався раніше. Приклад таблиці даних з вимірювань наведено на рис. 2.3.



ProductKey	ProductID	ProductName	SupplierName	CategoryName	ListUnitPrice
1	1	Chai	Exotic Liquids	Beverages	18
2	2	Chang	Exotic Liquids	Beverages	19
3	3	Aniseed Syrup	Exotic Liquids	Condiments	10
4	4	Chef Anton's Cajun Seasoning	New Orleans Cajun Delights	Condiments	22
5	5	Chef Anton's Gumbo Mix	New Orleans Cajun Delights	Condiments	21.35
6	6	Grandma's Boysenberry Spread	Grandma Kelly's Homestead	Condiments	25
7	7	Uncle Bob's Organic Dried Pears	Grandma Kelly's Homestead	Produce	30
8	8	Northwoods Cranberry Sauce	Grandma Kelly's Homestead	Condiments	40
9	9	Mishi Kobe Niku	Tokyo Traders	Meat/Poultry	97
10	10	Ikura	Tokyo Traders	Seafood	31
11	11	Queso Cabrales	Cooperativa de Quesos 'Las Cabras'	Dairy Products	21
12	12	Queso Manchego La Pastora	Cooperativa de Quesos 'Las Cabras'	Dairy Products	38
13	13	Konbu	Mayumi's	Seafood	6
14	14	Tofu	Mayumi's	Produce	23.25
15	15	Genen Shouyu	Mayumi's	Condiments	15.5
16	16	Pavlova	Pavlova, Ltd.	Confections	17.45
17	17	Alice Mutton	Pavlova, Ltd.	Meat/Poultry	39
18	18	Carnarvon Tigers	Pavlova, Ltd.	Seafood	62.5
19	19	Teatime Chocolate Biscuits	Specialty Biscuits, Ltd.	Confections	9.2

Рисунок 2.3 – Приклад таблиці вимірювань структури сховища даних TestDatabase

Один вимір куба може міститися як в одній таблиці (в тому числі і при наявності декількох рівнів ієрархії), так і в декількох зв'язаних таблицях, що відповідають різним рівням ієрархії у вимірі. Якщо кожен вимір міститься в одній

таблиці, така схема сховища даних носить назву «зірка» (star schema). Приклад такої схеми наведений на рис.2.4.

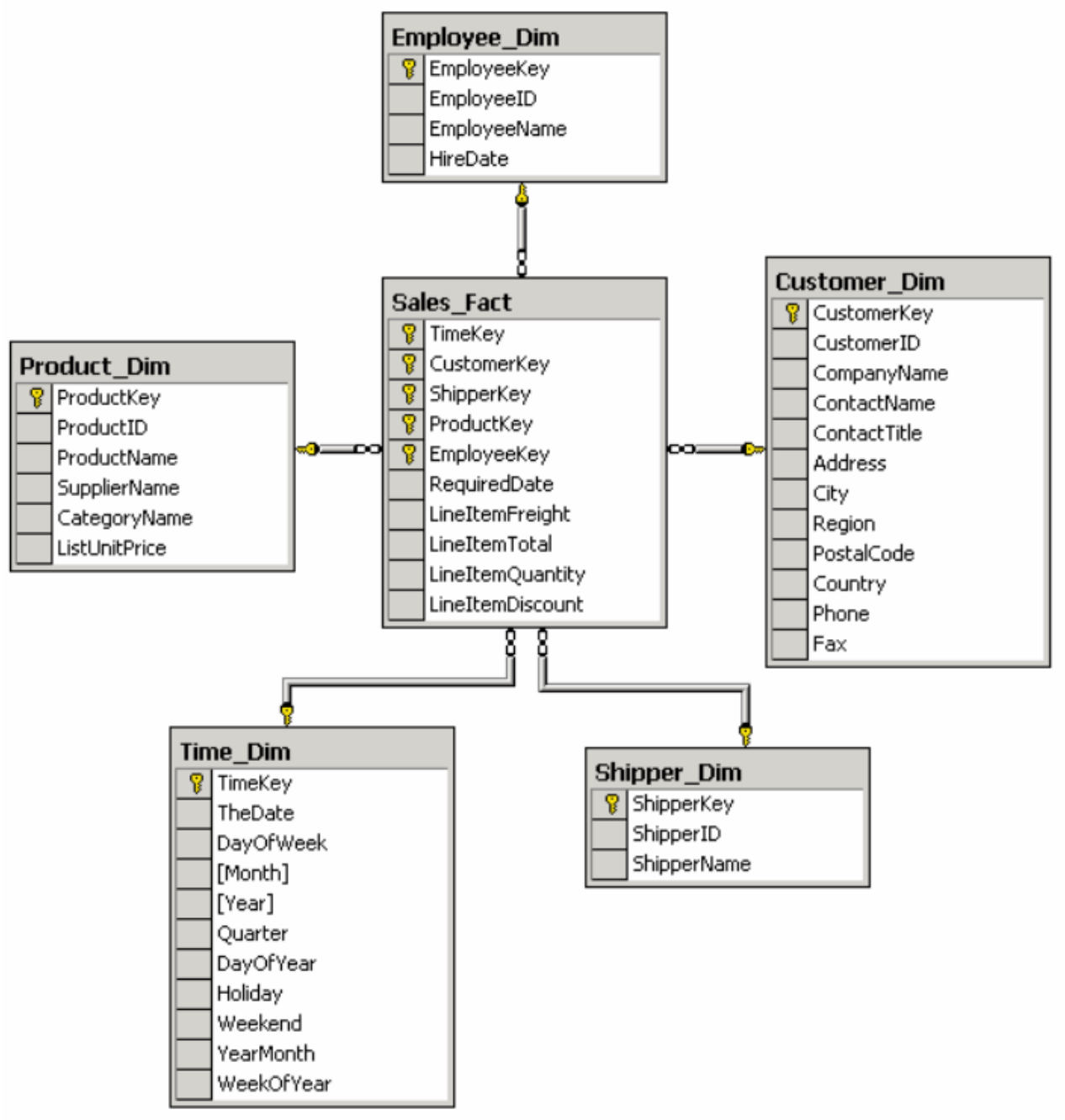


Рисунок 2.4 – Приклад таблиці вимірювань структури сховища даних типу «зірка»

Якщо ж хоча б один вимір міститься в декількох зв'язаних таблицях, така схема з обробки та сховища даних носить назву «сніжинка» (snowflake schema). Додатково таблиці вимірювань в такій схемі, зазвичай відповідні верхніх рівнів ієрархії вимірювання і знаходяться в співвідношенні «один до багатьох» в головній таблиці вимірювань, що відповідає нижньому рівню ієрархії, іноді

називають консольними таблицями (outrigger table).

Відзначимо, що при наявності ієрархічних вимірів з метою підвищення швидкості виконання запитів до сховища даних нерідко перевага віддається схемою «зірка». Однак не всі сховища даних проектуються за двома наведеними вище схемами. Так, досить часто замість ключового поля для вимірювання, що містить дані типу «дата», і відповідної таблиці вимірювань сама таблиця фактів може містити ключове поле типу «дата». У цьому випадку відповідна таблиця вимірювань просто відсутня.

У разі незбалансованої ієрархії (наприклад, такою, що може бути заснована на таблиці Employees бази даних Northwind, що має поле EmployeeID, яке одночасно є і первинним, і зовнішнім ключем і відображає підпорядкованість одних співробітників іншим (рис. 2.1) в схему «сніжинка» також слід вносити корективи. у цьому випадку зазвичай в таблиці вимірювань присутній зв'язок, аналогічна відповідній зв'язку в оперативній базі даних.

Ще один приклад відступу від правил - наявність декількох різних ієрархій для одного і того ж вимірювання. Типові приклади таких ієрархій - ієрархії для календарного та фінансового року або з різними способами угруповання членів вимірювання (наприклад, групувати товари можна за категоріями, а можна і по компаніям-постачальникам). В цьому випадку таблиця вимірювань містить поля для всіх можливих ієрархій з одними і тими ж членами нижнього рівня, але з різними членами верхніх рівнів (приклад такої таблиці наведено на рис. 2.3).

Також, таблиця вимірювань може містити поля, що не мають відношення до ієрархій і представляють собою просто додаткові атрибути членів вимірювань (member properties). Іноді такі атрибути можуть бути використані при аналізі даних.

Слід відмітити, що для створення реляційних сховищ даних нерідко застосовуються спеціалізовані бази даних, зберігання даних в яких оптимізовано з точки зору швидкості виконання запитів. Прикладом такого продукту є Sybase Adaptive Server IQ, який реалізує нетрадиційний спосіб зберігання даних в таблицях (не по рядках, а по стовпцях). Однак створювати сховища можна і в

звичайних реляційних баз даних.

2.3. Дослідження серверно-клієнтської структури з обробки даних

Дослідження серверно-клієнтської структури з обробки даних може бути проведений за допомогою різних засобів багатовимірного аналізу, які умовно можна розділити на клієнтські і серверні OLAP-засоби.

Клієнтські OLAP-засоби являють собою програми, які здійснюють обробку та обчислення агрегатних даних (сум, середніх величин, максимальних або мінімальних значень) і їх відображення, при цьому самі агрегатні дані містяться в кеші всередині адресного простору такого OLAP-засоби.

Якщо вихідні дані містяться в СУБД, то обчислення агрегатних даних виробляється самим OLAP-засобом. Якщо ж джерело вихідних даних - серверна база даних, багато з клієнтських OLAP-засобів посилають на сервер SQL-запити, що містять оператор GROUP BY, і в результаті отримують агрегатні дані, обчислені на сервері.

Як правило, OLAP-функціональність реалізована в засобах статистичної обробки даних (з продуктів цього класу на російському ринку широко поширені продукти компаній StatSoft і SPSS) і в деяких електронних таблицях. Зокрема, непоганими засобами багатовимірного аналізу володіє Microsoft Excel 2000. За допомогою цього продукту можна створити і зберегти як файл невеликий локальний багатомірний OLAP-куб і відобразити його дво- або тривимірні перетину.

Тому, існують багато засобів розробки містять бібліотеки класів або компонентів, що дозволяють створювати додатки, які реалізують найпростішу OLAP-функціональність. Для цього можна використовувати компоненти DecisionCube в Borland Delphi і Borland C ++ Builder. Крім цього багато компаній пропонують елементи управління ActiveX і інші бібліотеки, реалізують подібну функціональність.

Необхідно відзначити, що клієнтські OLAP-засоби застосовуються, як правило, при малому числі вимірів (зазвичай рекомендується не більше шести) і

невеликому різноманітності значень цих параметрів. При цьому, отримані агрегатні дані повинні вміщатися в адресному просторі подібного кошти, а їх кількість зростає експоненціально при збільшенні числа вимірювань. Тому навіть самі не ефективні клієнтські OLAP-засоби, як правило, дозволяють зробити попередній підрахунок обсягу необхідної оперативної пам'яті для створення в ній багатовимірного куба.

Багато клієнтські OLAP-засоби дозволяють зберегти вміст кеша з агрегатними даними у вигляді файлу, що, в свою чергу, дозволяє не проводити їх повторне обчислення. Також відомо, що нерідко така можливість використовується для відчуження агрегатних даних з метою передачі їх іншим організаціям або для публікації. Типовим прикладом таких агрегатних даних є статистика захворюваності в різних регіонах і в різних вікових групах, яка є відкритою інформацією, що публікується міністерствами охорони здоров'я різних країн і Всесвітньою організацією охорони здоров'я. При цьому власне вихідні дані, що представляють собою відомості про конкретні випадки захворювань.

Ідея збереження кешу з агрегатними даними у файлі отримала свій подальший розвиток в серверних OLAP-засобах, в яких збереження і зміна агрегатних даних, а також підтримка містить їхні сховища здійснюються окремим додатком або процесом, званим OLAP-сервером. Клієнтські програми можуть запитувати подібне багатовимірне сховище і у відповідь отримувати ті чи інші дані. Деякі клієнтські програми можуть також створювати такі сховища або оновлювати їх відповідно до зміненими вихідними даними.

Переваги застосування серверних OLAP-засобів в порівнянні з клієнтськими OLAP-засобами подібні з перевагами застосування серверних баз даних в порівнянні з настільними: в разі застосування серверних засобів обчислення і зберігання агрегатних даних відбуваються на сервері, а клієнтське додаток отримує лише результати запитів до них, що дозволяє в загальному випадку знизити мережевий трафік, час виконання запитів і вимоги до ресурсів, що споживаються клієнтським додатком. Відзначимо, що кошти аналізу та обробки даних масштабу підприємства, як правило, базуються саме на серверних

OLAP-засобах, наприклад, таких як Oracle Express Server, Microsoft SQL Server 2000 Analysis Services, Hyperion Essbase, продуктах компаній Crystal Decisions, BusinessObjects, Cognos, SAS Institute. Оскільки всі провідні виробники серверних баз даних виробляють ті чи інші серверні OLAP-засоби, вибір їх досить широкий і майже у всіх випадках можна придбати OLAP-сервер того ж виробника, що і у самого сервера баз даних.

Необхідно відзначити, що багато клієнтські OLAP-засоби (зокрема, Microsoft Excel 2000, Seagate Analysis і ін.) Дозволяють звертатися до серверних OLAP-сховищ, виступаючи в цьому випадку в ролі клієнтських додатків, що виконують подібні запити. Крім цього є чимало продуктів, що представляють собою клієнтські програми до OLAP-засобів різних виробників.

2.4. Особливості дослідження багатовимірної структури обробки даних

Особливості проектування багатовимірної структури з обробки та сховища даних полягають у наступному. В багатовимірних сховищах даних містяться агрегатні дані різного типу. Наприклад, обсяги продажів по днях, місяцях, роках, за категоріями товарів та інших полів.

Мета зберігання агрегатних даних - це скоротити час обробки та виконання запитів, оскільки в більшості випадків для аналізу і прогнозів цікаві не детальні, а сумарні дані. Тому при створенні багатовимірної бази даних завжди обчислюються і зберігаються деякі агрегатні дані.

Відзначимо, що збереження всіх агрегатних даних не завжди виправдано. Справа в тому, що при додаванні нових вимірів обсяг даних, що складають куб, зростає експоненціально (іноді говорять про «вибухове зростання» обсягу даних). Необхідно відзначити, що ступінь зростання обсягу агрегатних даних залежить від кількості вимірювань куба і членів вимірювань на різних рівнях ієрархій цих вимірів. Для вирішення проблеми «вибухового зростання» застосовуються різноманітні схеми, що дозволяють при обчисленні далеко не всіх можливих агрегатних даних досягти прийнятної швидкості виконання запитів.

Як вихідні, так і агрегатні дані можуть зберігатися або в реляційних, або в

багатовимірних структурах. Тому в даний час застосовуються три способи обробки та зберігання даних:

- MOLAP (Multidimensional OLAP) - вихідні і агрегатні дані зберігаються в багатовимірній базі даних. Зберігання даних в багатовимірних структурах дозволяє маніпулювати даними як багатовимірним масивом, завдяки чому швидкість обчислення агрегатних значень однакова для будь-якого з вимірів. Однак в цьому випадку багатовимірна база даних виявляється надлишкової, так як багатовимірні дані повністю містять вихідні реляційні дані.

- ROLAP (Relational OLAP) - вихідні дані залишаються в тій же реляційної базі даних, де вони спочатку і знаходилися. Агрегатні ж дані поміщають в спеціально створені для їх зберігання службові таблиці в тій же базі даних.

- HOLAP (Hybrid OLAP) - вихідні дані залишаються в тій же реляційної базі даних, де вони спочатку знаходилися, а агрегатні дані зберігаються в багатовимірної базі даних.

Деякі OLAP-засоби підтримують зберігання даних тільки в реляційних структурах, деякі - тільки в багатовимірних. Однак більшість сучасних серверних OLAP-засобів підтримують всі три способи зберігання даних. Вибір способу зберігання залежить від обсягу і структури вихідних даних, вимог до швидкості виконання запитів і частоти оновлення OLAP-кубів.

Відзначимо також, що переважна більшість сучасних OLAP-засобів не зберігає «порожніх» значень. Прикладом «порожнього» значення може бути відсутність продажів сезонного товару поза сезоном.

Джерелом даних для кубів OLAP, як правило, є реляційне сховище даних. Самі по собі аналітичні служби не містять засобів для поповнення реляційного сховища даними з оперативних баз даних, з якими працюють користувачі. Однак такі засоби містять багато сучасних серверні бази даних.

Зокрема, в Microsoft SQL Server існують ресурси для переносу даних з однієї реляційної СУБД в іншу з можливим їх перетворенням, які можна застосовувати і для поповнення сховищ даних, називаються службами перетворення даних (Data Transformation Services, DTS). Служби перетворення

даних можуть бути використані не тільки з Microsoft SQL Server, а й з будь-якими іншими джерелами даних, доступними через універсальний механізм доступу до даних OLE DB.

Відзначимо, що DTS дозволяє використовувати додаткові модулі розширення (plugins) і до складу аналітичних служб входить одне з таких розширень, що дозволяє оновлювати OLAP-куби.

При створенні описів OLAP-кубів за допомогою бібліотек SQL DSO ці описи, або метадані, зберігаються в репозитарії. Самі ж дані зберігаються в каталозі, вказаному при встановлені Analysis Services. Згодом це місце розташування можна змінити.

За замовчуванням репозитарій аналітичних служб є базою даних Access msmdrep.mdb, розташовану в каталозі Microsoft Analysis Services\bin, який при необхідності можна перенести і в базу даних Microsoft SQL Server. Поточна версія аналітичних служб не підтримує збереження репозитарія в базах даних інших баз даних, в той час як саме вихідне реляційне сховище даних може міститися в будь-якій базі даних, що доступна за допомогою універсальних механізмів доступу до даних OLE DB і ODBC.

Створення додатків для читання і запису в репозитарій за допомогою засобів, відмінних від бібліотек SQL DSO, не рекомендується, так як структура сховища не документована і може бути змінена в наступній версії аналітичних служб.

Основи OLAP (On-Line Analytical Processing) являють собою технології багатовимірного аналізу даних, типові структури сховищ даних і деякі технічні аспекти багатовимірного зберігання даних. Однією з розповсюджених типових архітектур OLAP-служб є Microsoft Analysis Services - OLAP-сервера фірми Microsoft, що входить в комплект поставки Microsoft SQL Server 2000 Enterprise Edition. На сьогоднішній день це визнане аналітиками Gartner Group одним з найбільш популярних продуктів цього класу.

Кінцевою метою використання обробки даних і OLAP є аналіз даних та представлення результатів цього аналізу в зручному для сприйняття і прийняття

рішень вигляді. Безпосереднє звернення клієнтського додатка, який відповідає за представлення результатів аналізу даних є ефективним рішенням.

Також, в цьому випадку в ньому повинні бути реалізовані засоби такого аналізу. По суті, воно повинно бути клієнтським OLAP-засобом. При всій простоті такого підходу до реалізації OLAP, ця реалізація не позбавлена недоліків, пов'язаних з обмеженнями, що накладаються на число вимірів і кількість членів в них.

У серверних OLAP-засобів таких недоліків немає. Тому, більш прогресивним представляється підхід, що заснований на застосуванні серверних OLAP-засобів в якості проміжної ланки між сховищем даних у вигляді реляційної базою даних і клієнтським додатком. В цьому випадку OLAP-сервер повинен перетворювати дані з реляційного сховища в форму, більш зручну для створення аналітичних звітів - в OLAP-куби.

Як приклад серверного OLAP-засобу розглянемо аналітичні служби Microsoft (Microsoft Analysis Services), що входять до складу Microsoft SQL Server 2000 Enterprise Edition. Основним компонентом аналітичних служб є Analysis Server - сервіс операційної системи Windows NT / 2000. Цей сервер призначений для створення OLAP-кубів на основі реляційних сховищ даних, а також для надання доступу до них з клієнтських додатків. Тому розглянемо, якими саме об'єктами маніпулює цей сервер і за допомогою яких механізмів це відбувається.

OLAP-куб, створений за допомогою аналітичних служб Microsoft, може містити всі дані з таблиці фактів плюс агрегатні значення для тих груп записів з цієї таблиці, які відповідають верхнім рівням ієрархії вимірів. При необхідності можна проводити динамічне оновлення куба, якщо в таблицю фактів були додані нові записи, а також вибрати, чи будуть дані з нижніх рівнів ієрархії зберігатися в самому кубі. Це відповідає способу зберігання даних Multidimensional OLAP. Його основним призначенням є зчитування з таблиці фактів сховища даних, що відповідає способам зберігання даних Relational OLAP і Hybrid OLAP.

З точки зору користувача відмінностей між цими способами зберігання немає, не рахуючи різниці в продуктивності звернень до цих кубів додатків.

Таким чином, аналітичні служби зберігають агрегатні дані тільки для найпростіших агрегатних функцій (сум, числа записів, максимальних і мінімальних значень). Однак у разі потреби можна створювати так звані обчислювані члени (calculated members) для отримання інших типів агрегатних значень (середніх, середньозважених, зміщених і незміщене дисперсій і т.д.). При цьому, крім застосування вбудованих засобів створення агрегатних даних, Analysis Services дозволяє використовувати для обчислення агрегатних даних функції VBA або Excel, а також створювати власні.

2.5. Висновки

В другому розділі визначені вимоги щодо дослідження систем з обробки даних. Було визначено, що типова система з обробки даних, як правило, відрізняється від звичайної реляційної бази даних. По-перше, звичайні бази даних призначені для того, щоб допомогти користувачам виконувати повсякденну роботу, тоді як системи з обробки даних даних призначені для прийняття рішень. Наприклад, продаж товару і виписування рахунку здійснюються з використанням бази даних, призначеної для обробки транзакцій, а аналіз динаміки продажів за декілька років, дозволяє планувати роботу з постачальниками - за допомогою систем з обробки даних. По-друге, звичайні бази даних постійно змінюються в процесі роботи користувачів. Системи з обробки за допомогою OLAP-технологій відносно стабільні: дані в інформаційних сховищах зазвичай оновлюються рідко та за розкладом. Тому, процес поповнення являє собою просте додавання нових даних за певний період часу без зміни колишньої інформації, що вже перебуває в сховище. По-третє, звичайні бази даних найчастіше є джерелом даних, що потрапляють в сховище яке потрібно для обробки даних. Крім того, сховище може поповнюватися за рахунок зовнішніх джерел, наприклад статистичних звітів.

Кінцевою метою використання технології використання методу OLAP є аналіз даних і представлення результатів цього аналізу у вигляді, що зручне для обробки даних, сприйняття і прийняття рішень. Основна ідея методу OLAP полягає в побудові багатовимірних кубів, які будуть доступні для обробки даних у

запитів користувача. Однак вихідні дані для побудови OLAP-кубів зазвичай зберігаються в реляційних базах даних. Нерідко це спеціалізовані реляційні бази даних, звані також сховищами даних (Data Warehouse). На відміну від так званих оперативних баз даних, з якими працюють програми, що модифікують дані, сховища даних призначені виключно для обробки і аналізу інформації. Тому, проектуються вони таким чином, щоб час виконання запитів до них було мінімальним. Зазвичай дані копіюються в сховище з оперативних баз даних згідно з визначеним розкладом.

РОЗДІЛ 3.
ДОСЛІДЖЕННЯ МЕТОДІВ З ДОСЛІДЖЕННЯ РОБОТИ СИСТЕМ З ОБРОБКИ ДАНИХ

3.1. Дослідження роботи систем на основі мікросервісних архітектур

Мікросервісна архітектура (МА) являє собою особливий варіант сервіс-орієнтованої структур програмного забезпечення. Така структура орієнтована на взаємодію невеликих, слабо пов'язаних і легко змінюваних модулів. Ці модулі отримали назву мікросервісів.

Початок поширення МА вважається [27-29] середина 2010-х років коли стали розвиватися практики гнучкої розробки програмного забезпечення та засоби DevOps.

МА з'явилася як продовження в розвитку сервіс-орієнтованих систем. Відомо [30], що сервіс-орієнтовані архітектури (СОА) є досить складними програмними системами, а взаємодія між окремими модулями які спираються на стандартизовані великовагові протоколи (такі, як SOAP, XML-RPC). На відміну від СОА [31], в МА використовуються компоненти, які виконують відносно елементарні функції. Також, в МА використовують легковагі мережеві комунікаційні протоколи, наприклад JSON, Protocol Buffers, Thrift.

Основою функціонування МА є зменшення ступеня взаємодії збільшення зв'язності між окремими модулями, що дозволяє простіше додавати і змінювати функціональність системи в будь-який час. Також, при використанні МА акцент виконується на простоту, незалежність розгортання та оновлення кожного з мікросервісів. Модулі можуть бути реалізовані за допомогою різних мов програмування, фреймворків.

Окремі модулі можуть також виконуватися в різних середовищах контейнеризації, віртуалізації, хмарних платформах, під керуванням різних операційних систем і апаратного забезпечення.

Філософія МА схожа на методологію Unix, згідно з якою:

- кожен сервіс має «робити щось одне, і робити це добре»;
- взаємодіяти з іншими модулями простими засобами;

- мікросервіси мінімальні і призначені для реалізації мінімальної функціональності.

Найбільш популярними середовищами реалізації мікросервісов є системи управління контейнерними додатками: Kubernetes, надбудови OpenShift і CloudFoundry, Docker Swarm, Apache Mesos і інші.

При роботі з такими середовищами кожен з мікросервісів ізолюється в окремий контейнер або невелику групу контейнерів, доступну з комп'ютерної мережі іншим мікросервісам або користувачам. Також, до функцій таких платформ відносяться управління середовищем оркестрації, забезпечення відмовостійкості і балансування навантаження. Типовий практикою є використання коштів DevOps, систем безперервної інтеграції, забезпечення автоматизації оновлення програмних додатків і розгортання мікросервісів.

3.2. Необхідність використання кластера RabbitMQ для обробки масивів даних

RabbitMQ є менеджер черг який обмінюватися даними між процесами, додатками і серверами [32, 33]. В його основні функції входять визначення і робота з чергами до яких можуть підключатися різні додатки і передавати або отримувати повідомлення (рис. 3.1).

Повідомлення може включати в себе будь-яку інформацію. Наприклад, це може бути просте текстове повідомлення або інформація про процес або завданню яку необхідно виконати іншим додатком.

Менеджер черг RabbitMQ може зберігати повідомлення до тих пір поки інший додаток або сервер, якій адресовано повідомлення, що не підключитися до кластеру і не забере (отримає) повідомлення з черги. Потім прокладання-одержувач обробляє повідомлення за потрібне йому чином.

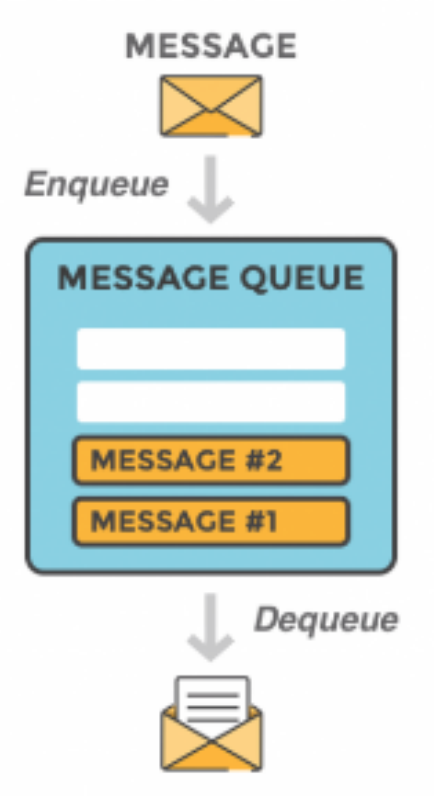


Рисунок 3.1 – Загальна схема роботи RabbitMQ

Іншим прикладом використання RabbitMQ є сценарій коли веб додаток дозволяє користувачам завантажувати інформацію на сайт. У цьому прикладі сайт повинен обробити цю інформацію, генерувати PDF і відправляти дані назад користувачеві на електронну пошту.

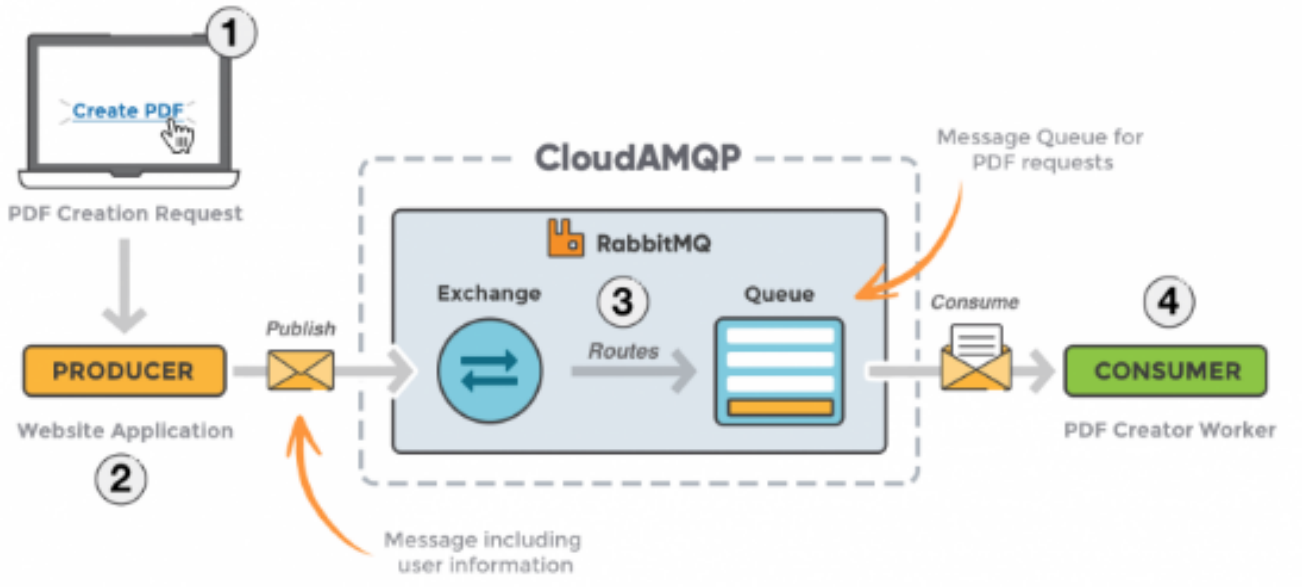


Рисунок 3.2 – Робота менеджера RabbitMQ з веб додатком

В цьому випадку, обробка інформації, генерація PDF і відправка email

зазвичай займає кілька секунд і полягає в наступному. Коли користувач введе інформацію в веб інтерфейс, додаток створить завдання на генерацію PDF і вся інформація в повідомленні і дані користувача будуть поміщені в чергу, яка визначена в кластері RabbitMQ. Даний механізм роботи менеджера RabbitMQ з веб додатком (рис. 3.2) полягає в наступному:

1. Користувач формує запит на створення PDF документа.

2. Додаток (Producer) посилає повідомлення в менеджер RabbitMQ, включаючи в запит інформацію про ім'я та електронну адресу користувача.

3. Оброблювач приймає повідомлення від програми постачальника і направляє його в потрібну чергу повідомлень RabbitMQ.

4. Воркер обробки повідомлень (Consumer) отримує завдання і починає генерацію PDF документа.

Даний приклад показує як можна ефективно використовувати черги повідомлень для розробки веб додатків.

3.3. Дослідження створення кластера RabbitMQ для обробки даних

У кластері менеджер RabbitMQ використовує кілька сервісів, у яких існують загальні користувачі, налаштування і черги. Сервіси можуть додаватися і віддалятися в процесі виконання, розташовуватися на різних мікросервісах. Однак, для підключеного клієнта вони будуть виглядати як один кластерний RabbitMQ сервіс. Такий механізм є досить ефективним для горизонтального мастшабірованія у випадках, коли користувачів стає багато для обробки запитів одного брокера.

Механізм кластеризації мікросервісів є подібним до принципів реплікації або високонавантажених ресурсів. Використання першого і другого зазначеного принципу не впливає на доступність даних і роботу сервісів при великих навантаженнях. При класичному використанні кластерів, вузли не є взаємозамінними. Хоча користувачі і налаштування прикладних систем будуть дублюватися на кожному з вузлів, де б вони не створювалися. Однак, з чергами повідомлень основні операції виконуються за іншими принципами. Ці відмінності

полягають в тому, що якщо який-небудь з вузлів перейшов в стан оффлайна, то його черги будуть недоступні для інших додатків.

Створення кластера RabbitMQ не є досить складним процесом. Для цього необхідно всім вузлам майбутнього кластера потрібно задати однакову Erlang cookie і викликати команду rabbitmqctl join_cluster. Далі, необхідно підготовити менеджер RabbitMQ вузли для мікросервісів Docker або Kubernetes.

Для створення кластера необхідно мати як мінімум два незалежних вузла (хоста). Найпростіший спосіб отримати їх - запустити два Docker контейнера з RabbitMQ всередині. У репозиторії Docker Hub для цих цілей є два образи: rabbitmq і rabbitmq: management. Більш краще використовувати другий так як в ньому вже є присутнім панель веб-адміністратора.

Також, необхідно щоб контейнери могли обмінюватися інформацією між собою в загальній комп'ютерній мережі, застосовувати доменні імена Erlang cookie. Виконується це вимоги за допомогою docker-compose. При створенні файлу конфігурації, автоматично будуть присвоювати імена хостів, виконуватися мережеві з'єднання, передаватися параметри між хостами, запускатися сервіси і т.д. Наприклад, такий файл конфігурації може мати наступний вигляд:

```
version: '2'
services:
  rabbit:
    image: rabbitmq:management
    hostname: rabbit
    ports:
      - "15677:15678"
    environment:
      - RABBITMQ_ERLANG_COOKIE='secret'
  cent01:
    image: rabbitmq:management
    hostname: hamster
```

```
ports:
  - "15673:15672"
environment:
  - RABBITMQ_ERLANG_COOKIE='secret'
```

У цьому файлі створюються два сервісу (контейнера), які після запуску формують RabbitMQ вузли кластера з іменами rabbit і cent01. Імена контейнерних хостів (hostname) задані явно. Вони будуть використовуватися в іменах RabbitMQ вузлів для взаємодії між ними.

Панель адміністратора для кластера RabbitMQ буде доступна з контейнера через порт 15678 з боку хоста. Також, в офіційний спосіб rabbitmq можна передати Erlang cookie як зміною оточення - RABBITMQ_ERLANG_COOKIE.

Далі, запускаються сервіси docker-compose.yml наступним чином:

```
docker-compose up
#Creating network "cluster_default" with the default driver
#Creating cluster_hamster_1
#...
#hamster_1  |
#hamster_1  | RabbitMQ 3.6.9. Copyright (C) 2007-2019 Pivotal Software, Inc.
#.....
```

Таким чином виконується створення, запуск і розробка мікросервісних архітектур на прикладі кластера RabbitMQ.

3.4. Дослідження роботи бази даних

Під час дослідження у дипломній роботі виконувався вимір швидкості обробки транзакцій за читанням 10 тис. записів з різних таблиць з метою визначення ефективності роботи баз даних. Для цього був створений сценарій навантаження у вигляді скрипта, який послідовно звертався до різної кількості таблиць у базах даних.

Експериментальні дослідження проводились на обчислювальному комплексі, характеристики якої зведені до таблиць 3.1 та 3.2.

Таблиця 3.1

Склад обчислювального комплексу на якому проводились дослідження (апаратні компоненти)

Апаратні компоненти	Значення апаратної компоненти
Процесор	Intel 7
Тактова частота процесора	2,2 ГГц
Оперативно-запам’ятовуючий пристрій	8 ГБ
Жорсткий диск	2 ТБ

Таблиця 3.2

Склад обчислювального комплексу на якому проводились дослідження (програмні компоненти)

Апаратні компоненти	Значення апаратної компоненти
Операційна система	Centos 7
Файлова система	Ext4
База даних	MySQL v. 5.7
OLAP-система	Pentaho v7.0

Для дослідження швидкості роботи бази даних вимірювались транзакції за читанням до записів таблиць за допомогою використання двох технології: OLTP та OLAP. Такі запити виконувались до таблиць у кількості 1, 2, 4 та 8 (табл. 3.3).

Таблиця 3.3

Результати досліджень вимірювались обробки записів у базі даних за технологіями OLTP та OLAP

Кількість таблиць що підлягали обробці	Значення вимірювань за технологією OLTP (у секундах)	Значення вимірювань за технологією OLAP (у секундах)

1	0,327	0,291
2	0,296	0,262
4	0,214	0,183
8	0,183	0,142
Загальні значення роботи скрипта	1,02	0,878
Середні значення для однієї таблиці	0,1275	0,1098

Після цього вимірювалась ефективність роботи бази даних за різними технологіями (OLTP та OLAP) та за різною кількістю таблиць (рис. 3.4) яка складала 17,165%.

Ефективність роботи бази даних була визначена як ступень швидкості в обробці даних (10 тис. записів) від використання OLTP і OLAP-технологій та в загальному вигляді мало таке співвідношення [33-35]:

$$E = (W_{oltp} - W_{olap}) / W_{oltp} * 100\%; \quad (3.1)$$

де

W_{oltp} - загальний час обробки даних за технологією OLTP;

W_{olap} - загальний час обробки даних за технологією OLAP.

За значеннями від табл. 3.3 знайдемо значення:

$$E = (3,239 - 2,683) / 3,239 * 100\% = 17,165\%;$$

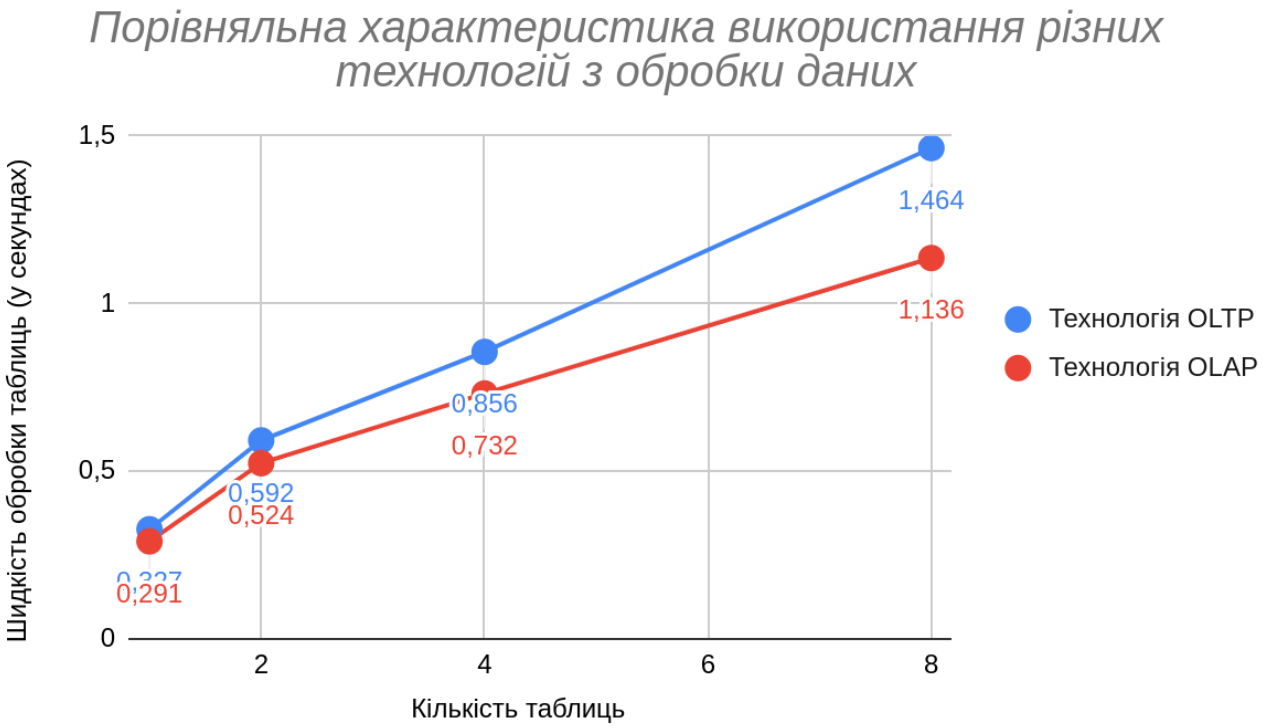


Рисунок 3.4 – Порівняльна характеристика використання різних технологій з обробки даних

Таким чином, в роботі запропоновано метод обробки даних у вигляді транзакція за читанням за допомогою технології OLAP. У порівнянні з іншою технологією, а саме OLTP ефективність за швидкістю обробки даних від використання технології OLAP становить 17,165%.

3.5. Висновки

В третьому розділі показано дослідження роботи систем з обробки даних де визначені особливості розробки мікросервісних архітектур. Основна увага приділена основним принципам мікросервісних та сервіс-орієнтованих архітектур. Надана інформація щодо використання сучасних кластерних мікросервісних засобів для створення програмних продуктів, що засновані на обробках даних. Визначені механізми роботи мікросервісів з контейнерами які виконують обробку даних.

Також, в третьому розділі показана необхідність використання кластера RabbitMQ для обробки масивів даних. Визначена загальна схема роботи RabbitMQ під час обробки повідомлень та роботу менеджера RabbitMQ з веб додатком. Показано механізм роботи менеджера RabbitMQ з веб додатком.

Після налаштування менеджера RabbitMQ з веб додатком було показано створення кластера RabbitMQ для обробки даних де описано механізм кластеризації мікросервісів який був подібним до принципів реплікації або високонавантажених ресурсів. Змістовно описано файл конфігурації мікросервісів з обробки даних та виконання створеного мікросервісу.

РОЗДІЛ 4.
ЕКОНОМІЧНИЙ РОЗДІЛ

4.1 Технологічний аудит розроблених методів та засобів процесу обробки масивів даних

Загальновідомо, що функціонування автоматизованих систем управління, інформаційних систем, вимірювальних комплексів тощо неможливе без здійснення обробки великого обсягу різноманітної інформації, причому програмне забезпечення таких систем має забезпечувати не тільки обробку, але й зберігання, тестування, введення-виведення великих масивів інформації самої різної структури.

Стрімкий розвиток інформаційних систем кожного дня висуває все нові та нові вимоги до розробки більш ефективних методів і засобів обробки інформації, які б задовольняли користувачів цієї інформації та відповідали критеріям надійності, ефективності, достовірності тощо.

Тому у виконаній нами магістерській кваліфікаційній роботі було заплановано за рахунок використання сучасних технологій мікросервісів розробити методи та засоби, застосування яких дозволило б суттєво підвищити продуктивність обробки масивів даних.

Для цього нами було: проведено аналіз існуючих методів підвищення обробки даних; запропоновано нові методи підвищення продуктивності обробки даних на основі використання сучасних технологій мікросервісів; розроблено програмні компоненти з обробки даних на основі використання сучасних технологій мікросервісів; проведено експериментальні дослідження розроблених засобів обробки даних.

В результаті, на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень, запропоновано технології обробки інформації та розроблено програмні засоби обробки даних на основі контейнерних технологій.

Проведемо технологічний аудит нашої розробки, який дозволить встановити її технічний рівень та комерційний потенціал. Для проведення аудиту запросимо 3-х експертів, які є знаними фахівцями в цій галузі знань: д.т.н. професора Романюка О.Н., , к.т.н. доцента Майданюка В.П. та к.т.н, доцента Рейду О.М.

Фахівці здійснювали оцінювання технічного рівня та комерційного потенціалу нашої розробки за методикою, [1], наведеною в табл. 4.1.

Таблиця 4.1 – Критерії оцінювання технічного рівня та комерційного потенціалу розробки та їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)

Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Крите р.	0	1	2	3	4
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший	Термін реалізації ідеї більший	Термін реалізації ідеї від 3-х до 5-	Термін реалізації ідеї менше	Термін реалізації ідеї менше 3-х років.

	за 10 років	за 5 років. Термін окупності інвестицій більше 10-ти років	ти років. Термін окупності інвестицій більше 5-ти років	3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін окупності інвестицій менше 3-х років

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Крите р.	0	1	2	3	4
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати проведеного з технологічного аудиту зведено в таблицю 4.2.

Таблиця 4.2 – Результати технологічного аудиту нашої розробки

Критерії	Прізвище, ініціали експерта		
	Романюк О.Н.	Майданюк В.П.	Рейда О.М.
	Бали, виставлені експертами:		
1	3	3	3
2	3	4	3
3	3	3	4
4	3	4	4
5	3	3	3
6	3	3	3
7	3	3	3
8	3	3	3
9	3	3	4
10	2	3	3
11	4	3	4
12	3	3	3
Сума балів	$СБ_1$ =36	$СБ_2$ = 38	$СБ_3$ = 40

Далі розрахуємо середньоарифметичну суму балів, що їх виставили запрошені експерти:

Середньоариф метична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{36+38+40}{3} = \frac{114}{3} = 38.$

На підставі рекомендацій, наведених в [1] (див. табл. 4.3), можна зробити висновок, що розроблені нами методи та засоби обробки масивів даних з використанням мікросервісів мають технічний рівень та комерційний потенціал, який є «вище середнього» (38 балів).

Таблиця 4.3 – Технічні рівні та комерційний потенціал розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Технічний рівень та комерційний потенціал розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Такий досить високий технічний рівень та комерційний потенціал нашої розробки пояснюється тим, розроблені нами методи та засобі обробки масивів даних з використанням мікросервісів успішно впроваджені низкою підприємств, зокрема, в компанії «SoftSystems», що забезпечило їй підвищення продуктивності роботи серверних додатків; на кафедрі програмного забезпечення Вінницького національного технічного університету для використання у навчальному процесі при викладанні курсів «Проектування та реінженерія програмного забезпечення» у студентів спеціальності 121 – «Інженерія програмного забезпечення» тощо.

4.2 Розрахунок витрат на розробку методів та засобів обробки масивів даних з використанням мікросервісів

Витрати на розробку методів та засобів обробки масивів даних з використанням мікросервісів складаються з таких основних статей:

А. Основна заробітна плата $З_о$ виконавців:

$$З_о = \frac{М}{Т_р} \cdot t \text{ грн.,} \qquad (4.1)$$

де М – місячний посадовий оклад конкретного виконавця, грн.

В 2019 році величини окладів виконавців у ВНТУ знаходяться в межах (4173…17000) грн за місяць;

$Т_р$ – число робочих днів в місяці; приймемо $Т_р$ = 20 днів;

t – число робочих днів роботи виконавців.

Зроблені розрахунки основної заробітної плати виконавців роботи зведемо до таблиці 4.4:

Таблиця 4.4 – Основна заробітна плата виконавців

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на оплату праці, грн	Примітка
1. Науковий керівник магістерської роботи	17000	850	25 годин	3542	-

2. Магістрант	2000	100	52	5200	-
3. Консультант з ЕЧ	14000	700	2,5 год.	292	-
4. Інші	8400	420	4	1680	
Всього				$З_о$ = 10714 грн	

б). Додаткова заробітна плата $З_д$ виконавців розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = (0{,}1...0{,}12) \cdot З_о . \qquad (4.2)$$

Для нашого випадку отримаємо:

$$З_д = 0{,}116 \times 10714 \approx 1243 \text{ грн.}$$

в). Нарахування на заробітну плату $Н_{зп}$ виконавців:

$$Н_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \qquad (4.3)$$

де β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, %. β = 22%. Тоді:

$$Н_{зп} = (10714+1243) \times 0{,}22 = 2631 \text{ грн.}$$

г). Амортизація обладнання, комп'ютерів та приміщень А, які використовувались під час виконання даної роботи:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ грн}, \qquad (4.4)$$

де Ц – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання роботи, грн.;

$Н_а$ – річна норма амортизаційних відрахувань. $Н_а$= (2,5...25)%;

Т – термін, використання обладнання, приміщень тощо, місяці.

Зроблені розрахунки зведемо у таблицю 4.5.

Таблиця 4.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, місяці	Величина амортизаційних відрахувань, грн
1. Обладнання: комп'ютери, принтери тощо	22000	20	2,25 (25% використання)	206
2. Приміщення університету	26000	3,5	2,25 (25% використання)	43
Всього				А = 249 грн

д). Витрати на матеріали М:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \text{ грн.}, \qquad (4.5)$$

де $Н_і$ – витрати матеріалу *і*-го найменування, кг; $Ц_і$ – вартість матеріалу *і*-го найменування, грн./кг.; $К_і$ – коефіцієнт транспортних витрат, $К_і$ = (1,1…1,15); $В_і$ – маса відходів матеріалу *і*-го найменування, кг; $Ц_в$ – ціна відходів матеріалу *і*-го найменування, грн/кг; n – кількість видів матеріалів.

е) Витрати на комплектуючі К:

$$К = \sum_{1}^{n} Н_і \cdot Ц_і \cdot К_і \text{ грн.}, \qquad (4.6)$$

де $Н_і$ – кількість комплектуючих *і*-го виду, шт.; $Ц_і$ – ціна комплектуючих *і*-го виду, грн; $К_і$ – коефіцієнт транспортних витрат, $К_і$ = (1,1…1,15); n – кількість видів комплектуючих.

Загальна вартість основних матеріалів та комплектуючих, які були використані під час виконання цієї роботи, складає приблизно 670 грн.

ж). Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д}, \qquad (4.7)$$

де В – вартість 1 кВт-год. електроенергії, в 2019 р. В ≈ 2,2 грн/кВт;

П – установлена потужність обладнання, кВт; П = 1,2 кВт;

Ф – фактична кількість годин роботи обладнання, годин. Приймемо, що Ф = 115 годин;

$К_п$ – коефіцієнт використання потужності; $К_п$ = 0,93.

$К_д$ – коефіцієнт корисної дії, $К_д$ = 0,84.

Тоді витрати на силову електроенергію складуть:

$$В_е = \frac{В К\,\Phi К_п}{К_д} = \frac{2,2 \cdot 1,2 \cdot 115 \cdot 0,93}{0,84} = 336 \text{ грн.}$$

и). Інші витрати $В_{ін}$ можна прийняти як (100…300)% від суми основної заробітної плати виконавців, тобто:

$$В_{ін} = (1..3) \times З_о. \qquad (4.8)$$

Для нашого випадку отримаємо:

$$В_{ін} = 1,5 \times 10714 = 16071 \text{ грн.}$$

к). Сума всіх попередніх статей витрат дає витрати на виконання цього етапу роботи магістрантом – В.

$$В = 10714 + 1243 + 2631 + 249 + 670 + 336 + 16071 = 31914 \text{ грн.}$$

л). Загальні витрати ЗВ на розробку та можливе впровадження результатів виконаної нами роботи розраховуються за формулою:

$$ЗВ = \frac{В}{\beta}, \qquad (4.9)$$

де β – коефіцієнт, який характеризує етап виконання роботи на шляху до її можливого впровадження. Оскільки наша розробка потребує доопрацювання, то згідно з [1, 2], $\beta \approx 0,5$.

Тоді: $ЗВ = \frac{31914}{0,5} = 63828,00$ грн або приблизно 64 тис. грн.

Тобто прогнозовані загальні витрати на виконання та завершення нашої розробки становлять приблизно 64 тис. грн.

4.3 Розрахунок економічного ефекту від можливого впровадження розроблених методів та засобів обробки масивів даних з використанням мікросервісів

Аналіз місткості ринку даної продукції показує, що на сьогодні кількість реальних користувачів подібних розробок становить 200 осіб. Разом з тим, не виникає сумніву, що кількість осіб, зацікавлених у нашій розробці, буде стрі-мко зростати.

Практично нашу розробку можна буде впровадити з 2021 року, і користуватися попитом на ринку наша розробка буде не менше 3-х років після впровадження. Прогноз зростання попиту на нашу розробку складе по роках:

1-й рік після впровадження (2021 р.) – приблизно на 50 шт.;

2-й рік після впровадження (2022 р.) – приблизно на 70 шт.;

3-й рік після впровадження (2023 р.) – приблизно на 90 шт.

На сьогодні існує велика кількість розробок, які виконують аналогічні функції. Найбільш поширеними є такі аналоги: «Microsoft Analysis Services» (виробник Microsoft) – ціна $4,256; «Essbase» (виробник Oracle) – ціна $2755; «TM1» (виробник IBM) – ціна $2930. В середньому подібні розробки коштують (4256+2755+2930)$ /3 = 3314$, що становить 79536 грн ≈ 79,5 тис. грн (при курсі НБУ 1$ = 24 грн станом на 24.11.2019 року)

Оскільки розроблені нами методи та засоби обробки масивів даних з використанням мікросервісів мають значно вищу продуктивність, то з метою підвищення конкурентоспроможності нашої розроообки її можна реалізовувати на ринку дещо дешевше, наприклад за 77,5 тис. грн або на 2,5 тис. грн дешевше.

Можливе збільшення чистого прибутку $\Delta П_i$, що його можна отримати потенційний інвестор від впровадження нашої розробки, розраховується за формулою [2]:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (4.10)$$

де $\Delta Ц_o$ – зміна основного якісного показника від впровадження розробки. Таким показником є зростання ціни реалізації нової розробки, тобто $\Delta Ц_o$ = (77,5 – 79,5) = -2 тис. грн;

N – основний кількісний показник, який визначає обсяг діяльності у цьому році до впровадження результатів розробки; N = 200 шт.;

Δ N – покращення основного кількісного показника від збільшення попиту на розробку. Як було зазначено вище, таке збільшення становить, відповідно по роках, +50, +70 та +90 шт.;

$Ц_o$ – основний якісний показник, який визначає обсяг діяльності (тобто ціну реалізації) у році після впровадження результатів розробки, грн.

$Ц_o$ = 77,5 тис. грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від

впровадження розробки; n = 3 роки;

λ – коефіцієнт, який враховує сплату податку на додану вартість $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати ρ = (0,2...0,5); візьмемо ρ = 0,5;

υ – ставка податку на прибуток. У 2019 році υ = 18%.

Величина чистого прибутку $\Delta П_1$, що його може отримати потенційний інвестор протягом першого року після реалізації нашої розробки (2021 р.):

$$\Delta П_1 = [-2 \cdot 200 + 77{,}5 \cdot 200] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) = 1187 \text{ тис. грн.}$$

Величина можливого чистого прибутку $\Delta П_2$ для потенційного інвестора від реалізації нашої розробки протягом другого (2022 р.) року складе:

$$\Delta П_2 = [-2 \cdot 200 + 77{,}5 \cdot 200] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) = 1717 \text{ тис. грн.}$$

Величина можливого чистого прибутку $\Delta П_3$ для потенційного інвестора протягом третього (2023 р.) року складе:

$$\Delta П_3 = [-2 \cdot 200 + 77{,}5 \cdot 200] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) = 2246 \text{ тис. грн.}$$

Теперішня вартість початкових інвестицій PV, що можуть вкладатися в нашу розробку, становитиме: PV = (2…5) × ЗВ.

Для нашого випадку PV = 5 × 64 = 320 тис. грн.

Розрахуємо абсолютний ефект вкладених інвестицій $Е_{абс}$.

$$Е_{абс} = ПП - PV, \qquad (4.11)$$

де ПП – приведена вартість всіх чистих прибутків від можливого впровадження нашої розробки, грн:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.12)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати впровадженої роботи, грн;

т – період часу, протягом якого виявляються результати впровадженої роботи, роки; т = 3 роки;

τ – ставка дисконтування (або прогнозований рівень інфляції). Для наших розрахунків приймемо, що τ = 0,11 (або 11%);

t – період часу (в роках) від моменту отримання прибутків до початку впровадження розробки.

Тоді приведена вартість всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження нашої розробки, складе:

$$ПП = \frac{1187}{(1+0,11)^2} + \frac{1717}{(1+0,11)^3} + \frac{2246}{(1+0,11)^4} = 963+1255 + 1480 = 3698 \text{ тис грн.}$$

Абсолютний ефект ід впровадження нашої розробки за 4 роки може становити::

$E_{абс}$ = 3698 –320 = 3378 тис. грн. або по 844,5 тис. грн протягом 4-х років.

Далі розрахуємо відносну ефективність $E_в$ вкладених у розробку інвестицій. Для цього скористаємося формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (4.13)$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс}$ = 3698 тис. грн;

PV – теперішня вартість початкових інвестицій PV = 320 тис. грн;

$T_ж$ – життєвий цикл розробки, роки. $T_ж$ = 5 років (2019-2023 рр.).

Для нашого випадку:

$$E_в = \sqrt[5]{1+\frac{3698}{320}} - 1 = \sqrt[5]{1+11,556} - 1 = \sqrt[5]{12,556} - 1 = 1,6587 - 1 = 0,6587 = 65,87\%.$$

Визначимо ту мінімальну дохідність, нижче за яку кошти в розробку нашої системи вкладатися не будуть.

Мінімальна дохідність $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \qquad (4.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = (0,10...0,15);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5), але може бути і значно більше.

Для нашого випадку отримаємо:

$\tau_{мін}$ = 0,15 + 0,5 = 0,65 або $\tau_{мін}$ = 65%.

Оскільки величина $E_в$ =65,87% > $\tau_{мін}$ = 65%, то інвестор у принципі може бути зацікавлений у фінансуванні нашої розробки.

Далі розраховуємо термін окупності коштів, які можуть бути вкладені у впровадження нашої розробки:

$$T_{ок} = \frac{1}{E_в}. \qquad (4.15)$$

Для нашого випадку термін окупності $T_{ок}$ коштів, вкладених у впровадження нашої розробки, складе:

$$T_{ок} = \frac{1}{0,6597} = 1,518 \text{ років,}$$

що свідчить про потенційну комерційну привабливість нашої розробки.

Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю:

Показники	Задані у ТЗ	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку методів та засобів обробки масивів даних з використанням мікросервісів	Не більше 70 тис. грн	64 тис. грн.	Досягнуто
2. Абсолютний ефект від впровадження розробки (щорічний), тис. грн.	не менше 800 тис. грн	по 844,5 тис. грн протягом 4-х років	Досягнуто
3. Внутрішня норма дохідності інвестицій, %	не менше 65%	65,87%	Досягнуто
4. Термін окупності інвестицій, роки	до 3-х років	1,518 років	Досягнуто

Таким чином, основні техніко-економічні характеристики розроблених нами методів та засобів обробки масивів даних з використанням мікросервісів, які були задані у технічному завданні, виконані.

4.4. Висновки

В четвертому розділі визначено технологічний аудит розроблених методів та засобів процесу обробки масивів даних. Проведемо технологічний аудит розробки системи з обробки даних, який дозволить встановити її технічний рівень та комерційний потенціал. Для проведення аудиту були запрошені три експерта з інформаційних технологій.

Розрахунок витрат на розробку методів та засобів обробки масивів даних з використанням мікросервісів дозволив визначити витрати на розробку системи що складалися з декількох основних статей де прогнозовані загальні витрати на виконання та завершення розробки системи становили приблизно 64 тис. грн.

Також був проведен розрахунок економічного ефекту від можливого впровадження розроблених методів та засобів обробки масивів даних з використанням мікросервісів. Цей розрахунок показав аналіз місткості ринку даної продукції де на сьогодні кількість реальних користувачів подібних розробок

становить 200 осіб. Після цього визначено прогноз зростання попиту на нашу розробку складе по роках: 1-й рік після впровадження (2021 р.) – приблизно на 50 шт.; 2-й рік після впровадження (2022 р.) – приблизно на 70 шт.; 3-й рік після впровадження (2023 р.) – приблизно на 90 шт.

Таким чином, в четвертому розділі були виконані всі задані у технічному завданні основні техніко-економічні характеристики розроблених методів та засобів обробки масивів даних з використанням мікросервісів.

ВИСНОВКИ

В першому розділі проведено аналіз стану проблеми обробки даних де визначено два різних підходи щодо побудови систем які основані на OLTP та OLAP методах. Перший підхід являє собою обробку даних на основі транзакцій в реальному часі та переважно використовуюється в системах управління базами даних. Цей підхід був розрахований на ефективний збір даних в реальному часі. Другий підхід являє собою аналітичну обробку даних в реальному часі, але націлене саме на вибірку та обробку інформації максимально зручним та ефективним способом. Це дозволяє на основі останнього підходу поширено використовувати у бізнесі, де щодня доводиться приймати безліч рішень для виробничих, маркетингових і кадрових рішень.

Тому, визначення ефективних рішень в галузі обробки даних має відбуватися на всіх рівнях управління організацією. Це в кінцевому підсумку призводить до успіху всієї організації в цілому.

Завдяки детальному структуруванні інформації OLAP-куби дозволяють оперативно здійснювати аналіз та обробку даних, формувати різні звіти у визначених розрізах і з довільною глибиною деталізації. Звіти можуть створюватися аналітиками, менеджерами, фінансистами, керівниками підрозділів в інтерактивному режимі. Це дозволяє швидко отримати відповіді, на які виникають щодня питання, і прийняти правильне рішення. При цьому співробітникам, для створення звітів не потрібно вдаватися до послуг програмістів, на що зазвичай йде чимало часу. Тому, часто в компаніях існує кілька інформаційних систем що потребують ефективної обробки даних: системи складського обліку, бухгалтерські системи, ERP системи для автоматизації окремих виробничих процесів, системи збору звітності з підрозділів компанії, а також безліч файлів, які знаходиться на комп'ютерах співробітників.

Також в першому розділі визначено, що системи з обробки даних зазвичай мають засоби надання користувачеві агрегатних даних для різних вибірок з вихідного набору в зручному для сприйняття і аналізу вигляді. Як правило, такі агрегатні функції утворюють багатовимірний (і, отже, нереляційний) набір даних.

Така структура необхідна для швидкої обробки даних та називається гіперкубом або метакубом. Це відбувається внаслідок того, що такі куби містять параметри, а осередки - залежні від них агрегатні дані. Причому оброблятися та зберігатися такі дані можуть і в реляційних таблицях, але в даному випадку мова йде про логічної організації даних, а не про фізичну реалізації їх зберігання.

Уздовж кожної осі метакубів, дані можуть бути організовані у вигляді ієрархії, що представляє різні рівні їх деталізації. Завдяки такій моделі з обробки даних користувачі можуть формувати складні запити, генерувати звіти, отримувати підмножини даних.

Під час проведення порівняльного аналіз аналогів в цьому розділі були визначені характерні риси процесів, які властиві та притаманні в тій чи іншій мірі всім OLTP-системам. Особливістю таких систем є те, що запити та звіти під час обробки даних є повністю регламентовані.

В роботі було розглянуто приклад з обробкою даних в галузі транспортних перевезень. Такі дослідження можуть бути певною реакцією на вхідну інформації що постійно змінюється. Про те, останнім часом у багатьох пунктах продажів, наприклад білетів, почастішали випадки нестачі квитків на певні маршрути, що дозволяє зробити припущення про доцільність організації додаткових рейсів на транспортні засоби. Однак для проведення таких досліджень необхідні як мінімум три речі. По-перше, потрібні дані про продажі квитків за досить тривалий період (кілька місяців або років). По-друге, дані не повинні містити протиріч, пропусків, аномальних значень та інших чинників, які не дозволять виконати коректний аналіз. По-третє, необхідна додаткова інформація про бізнес-середовищі: про конкурентів, ринкові тенденції, ціни на паливо та інше. Тому, типова OLTP-система не може забезпечити нічого з перерахованого. Саме з розумінням цих проблем приходить усвідомлення необхідності використання більш розвинених системи з обробки та зберігання даних, що орієнтовані на аналіз.

Під час постановки задачі було проведено аналіз обробки даних за допомогою OLAP-кубів, що в загальному випадку зводиться до пошуку оптимальної структури аналітичної системи, яка забезпечила б значне скорочення

часу на обробку даних і формування звітів та унікальну гнучкість в побудові звітності. В процесі висвітлення основних проблем аналізу оперативних даних може бути поставлена задача розробити єдину систему звітності, що надала б можливість оптимізувати бізнес-потоки і при цьому дозволяла відмовитися від механічної ручної обробки даних. При цьому можуть бути сформовані основні вимоги щодо аналітичної системи яка полягає у: можливості отримати будь-який аналітичний звіт без виклику програміста; скорочені часу отримання звіту в межах секунд за будь-який період; існування наскрізного аналізу даних за великий період; наявності єдиного сховища даних і системи аналізу інформації. Тому, такі сховище даних можуть являти собою складну систему, яка включає: ядро - систему управління базами даних; ресурси, що виконують імпорт, експорт та перетворення даних; засоби проектування обробки та сховища даних; засоби роботи з репозиторієм метаданих та засоби оперативного аналітики (OLAP-засоби).

В другому розділі визначені вимоги щодо дослідження систем з обробки даних. Було визначено, що типова система з обробки даних, як правило, відрізняється від звичайної реляційної бази даних. По-перше, звичайні бази даних призначені для того, щоб допомогти користувачам виконувати повсякденну роботу, тоді як системи з обробки даних даних призначені для прийняття рішень. Наприклад, продаж товару і виписування рахунку здійснюються з використанням бази даних, призначеної для обробки транзакцій, а аналіз динаміки продажів за декілька років, дозволяє планувати роботу з постачальниками - за допомогою систем з обробки даних. По-друге, звичайні бази даних постійно змінюються в процесі роботи користувачів. Системи з обробки та сховища даних відносно стабільне: дані в ньому зазвичай оновлюються за розкладом. Тому, процес поповнення являє собою просте додавання нових даних за певний період часу без зміни колишньої інформації, що вже перебуває в сховище. По-третє, звичайні бази даних найчастіше є джерелом даних, що потрапляють в сховище що потрібно для обробки даних. Крім того, сховище може поповнюватися за рахунок зовнішніх джерел, наприклад статистичних звітів.

Кінцевою метою використання технології використання методу OLAP є аналіз даних і представлення результатів цього аналізу у вигляді, що зручне для обробки даних, сприйняття і прийняття рішень. Основна ідея методу OLAP полягає в побудові багатовимірних кубів, які будуть доступні для обробки даних у запитів користувача. Однак вихідні дані для побудови OLAP-кубів зазвичай зберігаються в реляційних базах даних. Нерідко це спеціалізовані реляційні бази даних, звані також сховищами даних (Data Warehouse). На відміну від так званих оперативних баз даних, з якими працюють програми, що модифікують дані, сховища даних призначені виключно для обробки і аналізу інформації. Тому, проектуються вони таким чином, щоб час виконання запитів до них було мінімальним. Зазвичай дані копіюються в сховище з оперативних баз даних згідно з визначеним розкладом.

В третьому розділі показано дослідження роботи систем з обробки даних де визначені особливості розробки мікросервісних архітектур. Основна увага приділена основні принципи мікросервісних та сервіс-орієнтованих архітектур. Надана інформація щодо використання сучасних кластерних мікросервісних засобів для створення програмних продуктів, що засновані на обробках даних. Визначені механізми роботи мікросервісів з контейнерами які виконують обробку даних.

Також, в третьому розділі показана необхідність використання кластера RabbitMQ для обробки масивів даних. Визначена загальна схема роботи RabbitMQ під час обробки повідомлень та роботу менеджера RabbitMQ з веб додатком. Показано механізм роботи менеджера RabbitMQ з веб додатком.

Після налаштування менеджера RabbitMQ з веб додатком було показано створення кластера RabbitMQ для обробки даних де описано механізм кластеризації мікросервісів який був подібним до принципів реплікації або високонавантажених ресурсів. Змістовно описано файл конфігурації мікросервісів з обробки даних та виконання створеного мікросервісу.

В четвертому розділі визначено технологічний аудит розроблених методів та засобів процесу обробки масивів даних. Проведемо технологічний аудит

розробки системи з обробки даних, який дозволить встановити її технічний рівень та комерційний потенціал. Для проведення аудиту були запрошені три експерта з інформаційних технологій.

Розрахунок витрат на розробку методів та засобів обробки масивів даних з використанням мікросервісів дозволив визначити витрати на розробку системи що складалися з декількох основних статей де прогнозовані загальні витрати на виконання та завершення розробки системи становили приблизно 64 тис. грн.

Також був проведен розрахунок економічного ефекту від можливого впровадження розроблених методів та засобів обробки масивів даних з використанням мікросервісів. Цей розрахунок показав аналіз місткості ринку даної продукції де на сьогодні кількість реальних користувачів подібних розробок становить 200 осіб. Після цього визначено прогноз зростання попиту на нашу розробку складе по роках: 1-й рік після впровадження (2021 р.) – приблизно на 50 шт.; 2-й рік після впровадження (2022 р.) – приблизно на 70 шт.; 3-й рік після впровадження (2023 р.) – приблизно на 90 шт.

Таким чином, в четвертому розділі були виконані всі задані у технічному завданні основні техніко-економічні характеристики розроблених методів та засобів обробки масивів даних з використанням мікросервісів.

Список використаної літератури

1. Форрестер Дж. Основы кибернетики предприятия: индивидуальная динамика. — М.: Прогресс, 2001. — 340 с.

2. Грицунов О. В. Інформаційні системи та технології. Навчальний посібник. — Х.: ХНАМГ, 2010. — 222 с.

3. Економіка підприємства: навч. посібник / за заг. ред. В.Г. Герасимчука, А.Е. Розенплентера. – К.: ІВЦ «Видавництво «Політехніка», 2013. – 264 с.

4.А. Ю. Берко, О. М. Верес Організація баз даних: практичний курс : Навч. посіб. для студ.; Нац. ун-т "Львів. політехніка". - Л., 2012. - 149 с.

5.Основы конструирования и расчета химико-технологического и природоохранного оборудования. Справочник. / А.С. Тимонин Том 2, 2-е изд., перераб. и доп. - Калуга: Издательство Н. Бочкаревой, 2014. — 1030 с.

6. Цыпкин Я.З. Основи теорії автоматичнх систем. М., Наука, 2016. — 423 с.

7. Журавльова І. В. Інформаційно-комунікаційне забезпечення фінансової діяльності : навчальний посібник для студентів напряму підготовки 6.030508 "Фінанси і кредит" / І. В. Журавльова, І. Л. Латишева, О. В. Лебідь. – Х. : Вид. ХНЕУ ім. С.Кузнеця, 2014. – 424 с.

8. Годин В.В., Корнеев И.К. Управление информационными ресурсами: 17-модульная программа для менеджеров «Управление развитием организации». Модуль 17. — М.: «Инфра-М», 2000. — 352 с.

9. Гудвін Г.К., С.Ф. Гребе, М.Е. Сальдаго «Проектування систем управління». — пер. з англ. - М.: БІНОМ, Лабораторія знань, 2014. - 911 с.

10. Анхімюк В.Л., Олейко О.Ф., Міхєєв М.М. «Теорія автоматичного керування» - М.: Дизайн ПРО, 2012. - 352 с.

11.Пригорьев, Ю.А. Проблемы выбора доступа к данным при проектировании информационных систем на основе СУБД / Информационные технологии. - 2009 - №5. С. 4-10.

12. Воройский, Ф. З. Інформатика. Енциклопедичний систематизований словник-довідник. (Введення ЄІАС у сучасні інформаційні і телекомунікаційні

технології в термінах і фактах). — М.: Физматлит, 2007.— 760с.

13.А.А. Оводенко, А.М. Смирнов, А.Г. Степанов, Т.В. Третьякова Формирование информационного обеспечения для поддержки принятия решений на предприятии: учебное пособие ; РИО ГУАП. СПб., 2013. 141 с.

14. Дейт, К., Дж. Введение в системы баз данных, 6-е издание. К.; М.; Спб.: Издательский дом «Вильяме», 2014. - 848 с.

15. Скворцов, А.В. Автоматизация управления жизненным циклом продукции: Учебник для студентов учреждений высшего профессионального образования / А.В. Скворцов, А.Г. Схиртладзе, Д.А. Чмырь. - М.: ИЦ Академия, 2013. - 320 с.

16. Марк Лутц. Программирование на Python, 4-е видання, I том - Переклад з англійської. - СПб.: Символ-Плюс, 2011. - 992 с.

17. Т. М. Басюк, П. І. Жежнич Методи та засоби мультимедійних інформаційних систем : навч. посіб. ; Нац. ун-т "Львів. політехніка". - Львів : Вид-во Львів. політехніки, 2015. - 426 с. - Бібліогр.: с. 413-416.

18. DB-Engines Ranking [Електронний ресурс] // DB-engines. – 2017. – Режим доступу до ресурсу: https://db-engines.com/en/ranking

19. Branson T. 8 Major Advantages of Using MySQL [Електронний ресурс] / Tony Branson // Datamation – Режим доступу до ресурсу: http://www.datamation.com/storage/8-major-advantages-of-using-mysql.html

20. Перцовский М.И. Комплексная автоматизация промышленного предприятия: новые преимущества и новые проблемы//Мир компьютерной автоматизации. 2011.- №3 - С. 12-15.

21. Схиртладзе, А.Г. Автоматизация технологических процессов и производств: Учебник / А.Г. Схиртладзе, А.В. Федотов, В.Г. Хомченко. - М.: Абрис, 2012. - 565 с.

22. Боэм Б.У. Инженерное программирование для проектирования программного обеспечения. -М.: Радио і связь, 2005, -512с.

23. Энсор Д., Стивенсон Й. Oracle. Проектирование баз данных К.: BHV, 2009.-557 с.

24. Лисицин Н.А. ”Экономика, организация и планирование промышленного производства”, Минск, ”Вышэйшая школа”, 2010.-324 с.

25. Грекул В.И. Автоматизация деятельности предприятия розничной торговли с использованием информационной системы Microsoft Dynamics NAV: Учебное пособие / В.И. Грекул, Н.Л. Коровкина, Д.А. Богословцев. — М.: Бином, 2014. — 182 с.

26. Баин А. М. Современные информационные технологии систем поддержки принятия решений: учебное пособие / А. М. Баин. – М.: ИД "ФОРУМ", 2009. – 260 с.1. Irakli Nadareishvili. Microservice Architecture: Aligning Principles, Practices, and Culture. O'Reilly Media; 1 edition, 2018. P. 146.

27. Sam Newman. Building Microservices: Designing Fine-Grained Systems 1st Edition. O'Reilly Media; 1 edition, 2015. P. 280.

28. Susan Fowler. Production-Ready Microservices, 2017. O'Reilly Media; 1 edition. P. 172.

29. Thomas Erl. Service-Oriented Architecture (SOA): Concepts, Technology, and Design. Prentice Hall, 2005. P. 792.

30. Robert C. Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017. P. 432.

31. Gavin M. Roy. RabbitMQ in Depth. Manning Publications; 1 edition, 2017. P. 264.

32. Alvaro Videla. RabbitMQ in Action: Distributed Messaging for Everyone. Manning Publications; 1 edition, 2012. P. 312.

33. Как рассчитать эффективность работы оборудования? [Електронний ресурс] – Режим доступу до ресурсу: https://www.trn.ua/articles/8511/

34. Формула эффективности [Електронний ресурс] – Режим доступу до ресурсу: http://delo-do.com/article/formula_of_performance.html

35. Луценко И.А. Основы теории эффективности. – Канада, Altaspera

Publishing & Literary Agency Inc., 2012. – 65 с.

36. Методичні рекомендації з комерціалізації розробок, створених в результаті науково-технічної діяльності – К.: Наказ Державного комітету України з питань науки, інновацій та інформатики (Лист № 1/06-4-97 від 13.09.2010 р.).

37. Козловський В. О. Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт. – Вінниця: ВНТУ, 2012.

ДОДАТКИ

Додаток А

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"____ "_____2019 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методів і засобів моделювання світлових ефектів та графічного 3D редактора на їх основі» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____д.т.н., проф. О.Н.Романюк

"____"_____2019 р.

Виконав:

_____студент гр.2ПІ-17м В.О. Отришко

"____"_____2019 р.

Вінниця – 2019 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Обробка масивів даних з використанням мікросервісів».

Галузь застосування - системи обробки даних.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення продуктивності обробки масивів даних за рахунок використання OLTP-технологій та сучасних інформаційних технології мікросервісів. Запропоновано метод визначення ефективності роботи бази даних за допомогою використання OLTP-технологій з обробки даних.

Призначення роботи – розробка методів підвищення продуктивності обробки даних за рахунок використання OLTP-технологій, сучасних інформаційних технології мікросервісів та впровадження програмних засобів.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Грицунов О. В. Інформаційні системи та технології. Навчальний посібник. — Х.: ХНАМГ, 2010. — 222 с.
2. Економіка підприємства: навч. посібник / за заг. ред. В.Г. Герасимчука, А.Е. Розенплентера. – К.: ІВЦ «Видавництво «Політехніка», 2013. – 264 с.
3. А. Ю. Берко, О. М. Верес Організація баз даних: практичний курс : Навч. посіб. для студ.; Нац. ун-т "Львів. політехніка". - Л., 2012. - 149 с.
4. Цыпкин Я.З. Основи теорії автоматичнх систем. М., Наука, 2016. — 423 с.
5. Гудвін Г.К., С.Ф. Гребе, М.Е. Сальдаго «Проектування систем управління». — пер. з англ. - М.: БІНОМ, Лабораторія знань, 2014. - 911 с.
6. Анхімюк В.Л., Олейко О.Ф., Міхєєв М.М. «Теорія автоматичного керування» - М.: Дизайн ПРО, 2012. - 352 с.
7. Пригорьев, Ю.А. Проблемы выбора доступа к данным при проектировании информационных систем на основе СУБД / Информационные технологии. -

2009 - №5. С. 4-10.

4. Технічні вимоги

Базові методи підвищення швидкості обробки даних – методи представлення та обробки даних OLTP; вихідні дані для визначення швидкості транзакцій даних за читанням; дані про розмішування даних в таблицях у кількості 1, 2, 4 та 8 штук; кількість записів у кожній таблиці складає 10000; вихідні дані – значення загальної кількості часу роботи скрипта; середній час обробки даних для однієї таблиці.

5. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до МКР;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз моделей освітлення та вибір напрямків досліджень	
2	Розробка моделей з обробки даних на основі OLTP-технологій	
3	Розробка методів впровадження програмного оточення для мікросервісних технологій	

4	Розробка скриптів для навантажувального тестування на системи бази даних з метою визначення швидкості обробки записів	
5	Економічна частина	

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б

Програмний код додатку з обробки даних за допомогою мікросервісних технологій

```
package org.vntu.cardsystem.entity;
 public class Card
{
    private Long contactId;
    private String firstName;
    private String lastName;
    private String phone;
    private String email;

    public Card() {
    }

    public Card(String firstName, String lastName, String phone, String email) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.phone = phone;
        this.email = email;
    }

    public Card(Long contactId, String firstName, String lastName, String phone,
String email) {
        this.contactId = contactId;
        this.firstName = firstName;
        this.lastName = lastName;
        this.phone = phone;
        this.email = email;
```

```
}

public Long getCardId() {
   return contactId;
}

public void setCardId(Long contactId) {
   this.contactId = contactId;
}

public String getFirstName() {
   return firstName;
}

public void setFirstName(String firstName) {
   this.firstName = firstName;
}

public String getLastName() {
   return lastName;
}

public void setLastName(String lastName) {
   this.lastName = lastName;
}

public String getPhone() {
   return phone;
}

public void setPhone(String phone) {
   this.phone = phone;
```

```
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String toString() {
        return "Card{" + "contactId=" + contactId + ", firstName=" + firstName + ", lastName=" + lastName + ", phone=" + phone + ", email=" + email + '}';
    }
}
```

Розробка модуля управління з обробки даних картотеки службовців

```
package org.vntu.cardsystem.dao;

import org.vntu.cardsystem.entity.Card;
import java.util.List;

public interface CardDAO
{
    public Long addCard(Card contact);

    public void updateCard(Card contact);

    public void deleteCard(Long contactId);
```

```
    public Card getCard(Long contactId);
    public List<Card> findCards();
}


package org.vntu.cardsystem.dao;

public class CardDAOFactory
{
    public static CardDAO getCardDAO() {
        return new CardSimpleDAO();
    }
}

package org.vntu.cardsystem.business;

import org.vntu.cardsystem.dao.CardDAO;
import org.vntu.cardsystem.dao.CardDAOFactory;
import org.vntu.cardsystem.entity.Card;
import java.util.List;

public class CardManager
{
    private CardDAO dao;

    public CardManager() {
        dao = CardDAOFactory.getCardDAO();
    }

    public Long addCard(Card contact) {
        return dao.addCard(contact);
```

```
    }

    public void updateCard(Card contact) {
        dao.updateCard(contact);
    }
     public void deleteCard(Long contactId) {
        dao.deleteCard(contactId);
    }
     public Card getCard(Long contactId) {
        return dao.getCard(contactId);
    }
        public List<Card> findCards() {
        return dao.findCards();
    }
}
```

Додаток В. Ілюстративний матеріал

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доцент кафедри ПЗ_____ О. М. Хошаба

Рецензент, к.т.н, доцент кафедри КН_____ І. Р. Арсенюк

Нормоконтроль, к. т. н., доц. кафедри ПЗ _____О. М. Хошаба

Виконавець, студент групи 1ПІ-18м _____ Н. В. Буковський

Слайд 1 – Тема, автор, науковий керівник бакалаврської дипломної роботи:

Обробка масивів даних з використанням мікросервісів

Виконав:
студент групи 1ПІ-18м
Буковський Н.В.

Керівник:
к.т.н., доц. каф. ПЗ Хошаба О.М.

Слайд 2 – Мета, об'єкт та предмет дослідження:

Метою роботи є підвищення ефективності обробки масивів даних за рахунок використання сучасних технологій.

Об’єкт дослідження – процес обробки масивів даних.

Предмет дослідження – методи та засоби процесу обробки масивів даних.

Слайд 3 – Задачі дослідження:
Основними задачами дослідження є:
- провести аналіз та дослідження існуючих методів підвищення швидкості обробки інформації у базах даних;
- запропонувати нові:
- методи підвищення продуктивності швидкості обробки даних на основі використання сучасних технології мікросервісів;
- розробити програмні компоненти з обробки даних на основі використання сучасних технології мікросервісів;
- провести експериментальні дослідження розроблених засобів обробки даних.

Слайд 4 – Актуальність розробки:

З появою персональних комп'ютерів та мобільних пристроїв суттєво зросла роль сучасних систем обробки інформації.

Поряд з цим, декілька років тому з’явились потужні системи оперативної обробки інформації, які отримали назву OLTP. Ці технології зразу після появи почали використовувати багато компаній.

Саме цій проблемі та питанням використання сучасних інформаційних технологій присвячена дана робота.

Слайд 5 – Порівняльний аналіз аналогів:

Порівняльна характеристика OLAP і OLTP-систем

Властивість	OLTP-система	OLAP-система
1	2	3
Цілі використання даних	Швидкий пошук, найпростіші алгоритми обробки	Аналітична обробка з метою пошуку прихованих закономірностей, побудови прогнозів і моделей та інше
Рівень узагальнення (деталізації) даних	Деталізовані	Як деталізовані, так і узагальнені (агреговані)
Вимоги до якості даних	Можливі некоректні дані (помилки реєстрації, введення та інше)	Помилки в даних не допускаються, оскільки можуть призвести до некоректної роботи аналітичних алгоритмів
Формат зберігання даних	Дані можуть зберігатися в різних форматах в залежності від програми, в якому вони були створені	Дані зберігаються і обробляються в єдиному форматі
Час зберігання даних	Як правило, не більше року (в межах звітного періоду)	Роки, десятиліття

Слайд 6 – Порівняльний аналіз аналогів:

Властивість	OLTP-система	СППР
1	2	3
Зміна даних	Дані можуть додаватися, змінюватися і вилучатися	Допускається тільки поповнення; раніше додані дані змінюватися не повинні, що дозволяє забезпечити їх хронологію
Періодичність оновлення	Часто, але в невеликих обсягах	Рідко, але в більших обсягах
Доступ до даних	Повинен бути забезпечений доступ до всіх поточних (оперативних) даними	Повинен бути забезпечений доступ до історичних (тобто накопиченим за досить тривалий період часу) даними з дотриманням їх хронології
Характер виконуваних запитів	Стандартні, налаштовані заздалегідь	Нерегламентовані, що формуються аналітиком «на льоту» в залежності від необхідного аналізу
Час виконання запиту	Декілька секунд	Кілька хвилин

Слайд 7 – Перший розділ

В першому розділі було проведено обгрунтування методу анализа та постановка задачі дослідження систем з обробки даних.

Визначено аналіз стану проблеми обробки даних де визначено два різних підходи щодо побудови систем які основані на OLTP та OLAP методах.

Перший підхід являє собою обробку даних на основі транзакцій в реальному часі та переважно використовується в системах управління базами даних. Цей підхід був розрахований на ефективний збір даних в реальному часі.

Другий підхід являє собою аналітичну обробку даних в реальному часі, але націлене саме на вибірку та обробку інформації максимально зручним та ефективним способом.

Слайд 8 – Другий розділ

В другому розділі були:

- визначені вимоги щодо дослідження методів роботи систем з обробки даних;

- досліджена загальна структура систем з обробки даних;

- проведено дослідження серверно-клієнтської структури з обробки даних;

- визначені особливості дослідження багатовимірної структури обробки даних.

Слайд 9 – Третій розділ

В третьому розділі були:

- досліджена робота систем на основі мікросервісних архітектур;
- визначена необхідність використання кластера RabbitMQ для обробки масивів даних
- досліджена створення кластера RabbitMQ для обробки даних та роботи бази даних.

В результаті роботи запропоновано метод обробки даних у вигляді транзакція за читанням за допомогою технології OLAP. У порівнянні з іншою технологією, а саме OLTP ефективність за швидкістю обробки даних від використання технології OLAP становить 17,165%.

Слайд 10 – Експериментальні дослідження

Склад обчислювального комплексу (апаратні компоненти)

Апаратні компоненти	Значення апаратної компоненти
Процесор	Intel 7
Тактова частота процесора	2,2 ГГц
Оперативно-запам’ятовуючий пристрій	8 ГБ
Жорсткий диск	2 ТБ

Склад обчислювального комплексу (програмні компоненти)

Апаратні компоненти	Значення апаратної компоненти
Операційна система	Centos 7
Файлова система	Ext4
База даних	MySQL v. 5.7
OLAP-система	Pentaho v7.0

Слайд 11 – Експериментальні дослідження

Результати досліджень вимірювались обробки записів у базі даних за технологіями OLTP та OLAP

Кількість таблиць що підлягали обробці	Значення вимірювань за технологією OLTP (у секундах)	Значення вимірювань за технологією OLAP (у секундах)
1	0,327	0,291
2	0,296	0,262
4	0,214	0,183
8	0,183	0,142
Загальні значення роботи скрипта	1,02	0,878
Середні значення для однієї таблиці	0,1275	0,1098

Слайд 12 – Експериментальні дослідження