Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет ___Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра __Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____
Розроблення програмного модуля комп'ютеризованої системи для автоматизації керування архівацією інформації у реляційній базі даних
(тема)

Виконав:
здобувач _____2____ року навчання,
групи ____КТРСм-23-1
_____Галанін Ю.Д.
(прізвище, ініціали)

Спеціальність __174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(код і повна назва спеціальності)
Тип програми __освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Комп'ютеризовані та робототехнічні системи
(повна назва освітньої програми)
Керівник _доц. Іванов Л.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _____
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки

Рівень вищої освіти другий (магістерський)

Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютеризовані та робототехнічні системи
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Галаніну Юрію Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення програмного модуля комп'ютеризованої системи для автоматизації керування архівацією інформації у реляційній базі даних
затверджена наказом університету від 25 листопада 2024 р. № 1239Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 29 січня 2025 р.
3. Вихідні дані до роботи включають:
3.1 Сервер СКБД – MySQL
3.2 Середовище розробки – Intellij IDEA
3.3 Реалізувати функцію архівування інформації у реляційних базах даних
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Вступ
4.2 Аналіз теоретичних основ архівування даних та постановка завдання
4.3 Розробка програмного модуля
4.4 Експериментальна перевірка та оцінка ефективності
4.5 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _Діаграма компонентів, що відображає взаємодію основних частин системи. Презентація для захисту кваліфікаційної роботи._____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної області	16.09.2024	Виконано
2	Вибір технологій та засобів розробки	07.10.2024	Виконано
3	Проектування архітектури ПЗ	21.10.2024	Виконано
4	Розробка алгоритмів та основних компонентів модуля	30.11.2024	Виконано
5	Розробка інтерфейсу користувача	19.12.2024	Виконано
6	Тестування та відлагодження	10.12.2024	Виконано
7	Підготовка документації	31.12.2024	Виконано

Дата видачі завдання __30__ __серпня_____2024 р.

Здобувач _____

(підпис)

Керівник роботи _____ ___доц. Іванов Л.С._____

(підпис) (посада, прізвище, ініціали)

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на віповідне джерело.

«17» січня 2025 р.  Галанін Ю.Д.

РЕФЕРАТ

Пояснювальна записка містить: 88 с., 4 рис., 3 дод., 21 джерело.

РЕЛЯЦІЙНІ БАЗИ ДАНИХ, АРХІВУВАННЯ, СТИСНЕННЯ, ПРОГРАМНИЙ МОДУЛЬ, ЦІЛІСНІСТЬ ДАНИХ.

Об'єкт дослідження – процеси автоматизації управління архівуванням даних у реляційних базах даних. Предмет дослідження – методи, алгоритми та програмні засоби, що забезпечують ефективну автоматизацію архівування даних у реляційній базі даних.

Мета кваліфікаційної роботи – підвищення ефективності управління процесами архівування даних у реляційних базах за рахунок автоматизацій операцій архівування, зберігання та відновлення даних.

Методи дослідження – вивчення існуючих методів та підходів, експериментальне дослідження, порівняльний аналіз, тестування та верифікація.

Кваліфікаційна робота присвячена розробці програмного модуля для автоматизації архівування даних у реляційних базах даних. Модуль реалізує алгоритм вибірки даних, архівування з різними методами стиснення та їх зберігання в зашифрованому вигляді. Результати кваліфікаційної роботи апробовані у конференції «Інформаційні технології та автоматизація – 2024».

Також, отримані результати роботи можно віднести до Цілі сталого розвитку 9 "Промисловість, інновації та інфраструктура", а саме пункт 9.5 «Створити фінансову та інституційну системи (інноваційну інфраструктуру), що забезпечуватимуть розвиток наукових досліджень та науковотехнічних (експериментальних) розробок».

ABSTRACT

The explanatory note contains: 88 p., 4 images, 3 addition, 21 references.

RELATIONAL DATABASES, ARCHIVING, COMPRESSION, SOFTWARE MODULE, DATA INTEGRITY.

The object of the study is the automation of data archiving management processes in relational databases.

The subject of the study is methods, algorithms and software tools that ensure effective automation of data archiving in a relational database.

The purpose of the qualification work is to increase the efficiency of data archiving management processes in relational databases by automating archiving, storage and data recovery operations.

Research methods are the study of existing methods and approaches, experimental research, comparative analysis, testing and verification.

This qualification work is devoted to the development of a software module for automating data archiving in relational databases. The module implements an algorithm for data selection, archiving with various compression methods and their storage in encrypted form. The work also considers aspects of data security, such as archive integrity and access control, effectiveness testing, comparison with other approaches and performance evaluation.

The results of the qualification work were tested at the conference "Information Technologies and Automation, 2024".

Also, the results of the work can be attributed to Sustainable Development Goal 9 "Industry, Innovation and Infrastructure", namely item 9.5 "Create financial and institutional systems (innovation infrastructure) that will ensure the development of scientific research and scientific and technological (experimental) developments".

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних;

РБД – реляційна база даних;

СУБД – система управління базами даних;

API – прикладний програмній інтерфейс;

SQL – мова структурованих запитів.

ВСТУП

У сучасному світі, де обсяги інформації зростають у геометричній прогресії, зберігання та управління даними стають ключовими аспектами ефективної діяльності організацій. Архівування даних є важливим процесом для забезпечення їхньої збереженості, доступності та зменшення навантаження на активні інформаційні системи. Особливої уваги цей процес потребує в контексті реляційних баз даних, які широко використовуються в різних галузях для обробки структурованої інформації. Автоматизація управління архівуванням даних у таких системах дозволяє значно підвищити ефективність їхньої роботи, мінімізувати людські помилки та забезпечити відповідність нормативним вимогам.

Основною метою роботи є підвищення ефективності управління процесами архівування даних у реляційних базах за рахунок створення програмного модуля, який автоматизує операції архівування, зберігання та відновлення даних, забезпечуючи оптимізацію використання ресурсів бази даних, зменшення ручної праці та підвищення надійності збереження інформації. Для досягнення цієї мети визначено такі завдання:

– провести аналіз існуючих методів і технологій архівування даних;

– розробити алгоритми, що забезпечують ефективне архівування у реляційній базі даних;

– створити програмний модуль, який реалізує ці алгоритми;

– провести тестування розробленого модуля та оцінити його ефективність.

Об'єктом дослідження є процеси автоматизації управління архівуванням даних у реляційних базах даних. Предметом дослідження є методи, алгоритми та програмні засоби, що забезпечують ефективну автоматизацію архівування даних у реляційній базі даних.

Наукова новизна роботи полягає в розробці та застосуванні нових алгоритмів і підходів до автоматизації процесу архівування даних у реляційній базі даних. Практична значущість полягає в можливості використання створеного програмного модуля для підвищення ефективності функціонування інформаційних систем, оптимізації їхньої продуктивності та забезпечення відповідності вимогам до збереження даних.

Робота складається з чотирьох основних розділів. У першому розглядаються теоретичні основи архівування даних. У другому здійснюється обґрунтування вибору методів і постановка завдання автоматизації. Третій розділ присвячено безпосередньо розробці програмного модуля. Четвертий містить результати тестування та оцінки ефективності розробленого рішення. Крім того, у роботі наведено розділ з охорони праці, який акцентує увагу на забезпеченні безпечних умов роботи з інформаційними системами.

У процесі виконання кваліфікаційної роботи було використано Методичні вказівки з підготовки та захисту кваліфікаційної роботи, які визначають вимоги до структури та змісту роботи [1], а також документація щодо структури та правил оформлювання ДСТУ 3008:2015, що регламентують оформлення документів відповідно до національних стандартів [2].

Результати роботи були опубліковані на XVII міжнародній науково-практичній конференції «Інформаційні технології та автоматизація – 2024» [3].

Також, отримані результати роботи можно віднести до Цілі сталого розвитку 9 "Промисловість, інновації та інфраструктура", а саме пункт 9.5 «Створити фінансову та інституційну системи (інноваційну інфраструктуру), що забезпечуватимуть розвиток наукових досліджень та науковотехнічних (експериментальних) розробок».

1 ТЕОРЕТИЧНІ ОСНОВИ АРХІВУВАННЯ ДАНИХ

Архівування даних є важливим аспектом управління інформаційними ресурсами в сучасних організаціях. Цей процес включає переміщення або копіювання застарілих або менш важливих даних із основних сховищ у спеціалізовані архіви, що забезпечує оптимізацію використання ресурсів, підвищення продуктивності баз даних та забезпечення довготривалого збереження інформації. У цьому розділі розглянуто ключові поняття, методи та підходи до архівування даних, а також особливості його реалізації в реляційних базах даних.

1.1 Огляд методів і технологій архівування даних

Архівування даних є процесом перенесення або збереження даних, що не використовуються активно, на спеціальні носії або в інші формати для збереження їх цілісності і доступності. Архівування дозволяє знижувати навантаження на основну систему зберігання даних, зменшувати час пошуку та обробки інформації, а також забезпечувати збереження даних для можливості їх подальшого відновлення чи використання.

У реляційних базах даних архівування даних є важливою частиною управління великими обсягами інформації, оскільки допомагає підтримувати оптимальну продуктивність системи, одночасно забезпечуючи доступність архівованих даних для подальшої аналітики чи відновлення [4].

Існує кілька основних методів архівування даних у реляційних базах даних, кожен з яких має свої переваги та недоліки в залежності від конкретних вимог і ситуації:

– фізичне архівування. Фізичне архівування передбачає переміщення даних з основної бази даних на інші носії, як-от зовнішні диски або спеціалізовані сховища. Це може бути перенесення даних в окремі файли або таблиці, де вони зберігаються в зжатому форматі або на менш швидких носіях, що знижує навантаження на основну систему;

– логічне архівування. Логічне архівування полягає в тому, що дані залишаються в тій самій базі даних, але позначаються як "архівні" або "неактивні". Зазвичай це здійснюється за допомогою спеціальних тегів чи статусів, які дозволяють визначити, чи є ці дані актуальними для поточних операцій або потребують архівування;

– інкрементне архівування. При інкрементному архівуванні дані архівуються поетапно, зазвичай лише нові або змінені записи. Це дозволяє значно зменшити обсяг даних, які необхідно обробляти за раз, та пришвидшити процес архівування;

– диференційне архівування. Диференційне архівування схоже на інкрементне, але тут архівуються всі зміни, що сталися після останнього повного архівування. Це забезпечує швидший доступ до відновлених даних, хоча потребує більших обсягів пам'яті для збереження змін.

Сучасні технології архівування даних використовують різні підходи до забезпечення ефективності, збереження та відновлення інформації. Ось основні технології, що застосовуються для архівування даних у реляційних базах даних:

– SQL-скрипти та автоматизація. Відповідно до специфікацій СУБД, для архівування даних можуть використовуватися SQL-скрипти, які автоматизують процеси переміщення даних між таблицями або навіть між різними серверами. Це дозволяє значно зменшити людський фактор і забезпечити постійну регулярність архівування;

– рішення на основі хмарних технологій. Хмарні технології активно використовуються для архівування даних, особливо коли обсяги інформації занадто великі для традиційних методів зберігання. Використання хмарних

сервісів, таких як Amazon S3, Google Cloud Storage або Microsoft Azure, дозволяє не лише зберігати великі обсяги даних, а й знижувати витрати на фізичне обладнання [5];

– бекап і відновлення за допомогою засобів СУБД. Більшість сучасних СУБД (наприклад, PostgreSQL, MySQL, Microsoft SQL Server) мають вбудовані механізми бекапування та відновлення, які можна налаштувати для регулярного архівування даних. Вони підтримують створення резервних копій на різних етапах, що дозволяє відновлювати як повні бази даних, так і окремі їх частини;

– інструменти для роботи з великими даними (Big Data). Для роботи з великими обсягами даних, які не завжди можуть бути ефективно збережені в класичних реляційних базах даних, використовуються технології, як-от Hadoop, Apache Spark, або інші рішення для обробки даних на основі розподілених систем. Архівування в таких системах часто включає обробку та збереження даних у вигляді блоків, що забезпечує високу ефективність при збереженні великих обсягів інформації.

Для автоматизації процесу архівування в реляційних базах даних використовуються різноманітні алгоритми та програмні інструменти. Одним із основних аспектів є визначення критичних даних для архівування та налаштування регулярних перевірок [6]. Наприклад:

– періодичне архівування за розкладом. За допомогою механізмів планування завдань (cron, Task Scheduler) можна налаштувати регулярне виконання архівування в задані періоди;

– умови архівування. Алгоритми, які визначають, коли дані підлягають архівуванню (наприклад, на основі віку записів або відсутності активності);

– інтеграція з іншими системами. Використання API для інтеграції з іншими зовнішніми сервісами або системами для управління архівами, що дозволяє спростити весь процес.

1.2 Особливості реляційних баз даних у контексті архівування

Реляційні бази даних (РБД) є найбільш поширеним типом систем управління базами даних, що зберігають інформацію у вигляді таблиць, де кожна таблиця складається з рядків і стовпців. Дані в таких базах зазвичай організовані так, щоб забезпечити їх зручний доступ, модифікацію та управління за допомогою мов запитів, зокрема SQL (Structured Query Language) [6].

Однією з основних особливостей реляційних баз даних є їх здатність підтримувати цілісність і зв'язність даних за допомогою механізмів, таких як зовнішні ключі, індекси та транзакції. Вони також використовують нормалізацію для усунення надлишковості даних, що сприяє зменшенню обсягів зберігання. Реляційні бази є основою для багатьох інформаційних систем завдяки своїй гнучкості та здатності ефективно управляти даними [4].

Реляційні бази даних мають низку специфічних характеристик, які впливають на процес архівування даних. Ось основні з них:

– структурованість даних. Оскільки реляційні бази даних зберігають дані в таблицях, архівування здійснюється на рівні таблиць або окремих рядків. Це дозволяє здійснювати архівування на основі певних критеріїв, таких як вік записів або їхній статус, що не завжди можливе в інших типах баз даних. Також можна архівувати цілу таблицю або частину даних на основі виконаних запитів;

– цілісність даних. Однією з важливих характеристик реляційних баз даних є підтримка транзакційної цілісності. Кожен етап архівування має бути ретельно спланований, щоб зберегти послідовність та узгодженість даних. Під час архівування важливо забезпечити, щоб інформація, яка архівується, не стала недоступною або пошкодженою, що може вплинути на цілісність усієї бази;

– індекси та їх вплив на архівування. Індекси значно пришвидшують виконання запитів до бази даних, однак вони можуть збільшувати обсяг даних, що зберігаються. При архівуванні даних необхідно враховувати, чи потрібно

зберігати індекси на архівованих таблицях. Часто індекси для архівованих даних можуть бути відключені або збережені в окремих архівних таблицях для економії простору [7];

– продуктивність при архівуванні великих обсягів даних. Архівування в реляційних базах даних, які містять великі обсяги даних, може призводити до значного навантаження на систему. У таких випадках важливо забезпечити, щоб архівування не призводило до падіння продуктивності основної бази даних. Це часто досягається за рахунок застосування інкрементного або диференційного архівування, а також за допомогою автоматизованих процесів, які можуть виконуватися в неробочі години;

– масштабованість і резервне копіювання. Реляційні бази даних повинні підтримувати масштабованість, що дозволяє зберігати великі обсяги даних. Масштабованість особливо важлива для архівування, оскільки з часом обсяг інформації в базі може значно збільшуватися. Використання реплікаційних або кластеризованих рішень дозволяє створити резервні копії даних, що спрощує процес архівування і відновлення [7].

Реляційні бази даних використовуються в різних галузях, і вимоги до архівування можуть відрізнятися в залежності від типу даних і специфіки роботи. Наприклад, у фінансових організаціях архівування даних повинно забезпечувати відповідність регуляторним вимогам, що можуть включати зберігання фінансових звітів і транзакцій на певний період. В медичних установах архівування даних повинно відповідати вимогам законодавства щодо збереження медичних записів пацієнтів. А в онлайн-торгівлі архівування даних може бути зосереджене на збереженні історії транзакцій, користувацьких даних і замовлень для подальшої аналітики та звітності.

Попри всі переваги реляційних баз даних, існують певні проблеми, з якими можна зіткнутися під час архівування, наприклад:

– великі обсяги даних. Коли обсяг даних у реляційних базах даних досягає значних розмірів, архівування стає складним завданням через потребу у великих ресурсах для обробки та зберігання даних;

– труднощі з відновленням даних. Відновлення даних з архіву в реляційних базах може бути складним процесом, особливо якщо архівування виконувалося на різних рівнях (наприклад, на рівні таблиць або окремих рядків). Це може вимагати спеціальних інструментів для відновлення цілісності;

– залежність від конкретної СУБД. Різні системи управління базами даних можуть використовувати різні методи архівування, що ускладнює процес інтеграції та міграції між системами. Це вимагає використання спеціалізованих інструментів або налаштувань для забезпечення сумісності.

1.3 Аналіз вимог до систем автоматизації архівування

Система автоматизації архівування даних повинна відповідати ряду специфічних вимог, щоб забезпечити ефективність, надійність і безпеку процесу зберігання інформації на різних етапах її життєвого циклу. Вимоги до таких систем включають:

– високу доступність та надійність. Система архівування повинна забезпечувати високу доступність збережених даних при мінімальних затратах часу на відновлення. Це вимагає застосування надійних методів резервного копіювання, а також механізмів для швидкого відновлення даних у випадку збоїв або непередбачуваних ситуацій;

– масштабованість. Оскільки обсяги даних, що потребують архівування, можуть збільшуватися, система повинна бути здатною обробляти великі масиви даних без зниження продуктивності. Масштабованість також включає здатність ефективно використовувати ресурси, що можуть бути розширені відповідно до зростання вимог;

– інтеграцію з існуючими інформаційними системами. Автоматизація архівування повинна включати інтеграцію з іншими системами, що зберігають або обробляють дані, наприклад, з реляційними базами даних, системами зберігання даних, обробки транзакцій, а також з інформаційними системами, що підтримують звітність і аналітику;

– гнучкість та налаштовуваність. Кожен процес архівування має бути налаштований відповідно до специфіки організації та даних, які обробляються. Система повинна дозволяти користувачам налаштовувати параметри архівування, такі як періодичність, формат зберігання, обсяг даних, що архівуються, тощо;

– безпеку та відповідність нормативним вимогам. Архівування даних потребує гарантії конфіденційності, цілісності та доступності даних. Система повинна підтримувати шифрування, контроль доступу та аудит, що забезпечить відповідність нормативним вимогам і політикам організації;

– автоматизацію процесів. Основна перевага систем автоматизації архівування полягає у скороченні людської участі в процесі. Процеси архівування, відновлення та обробки даних повинні бути автоматизованими, щоб зменшити помилки, підвищити швидкість виконання завдань та знизити витрати на підтримку.

Система автоматизації архівування повинна забезпечити виконання кількох ключових функцій:

– інкрементальне архівування. Для зменшення витрат на зберігання і прискорення процесу архівування повинна бути реалізована підтримка інкрементального архівування, яке дозволяє зберігати лише змінені дані з моменту останнього архівування;

– підтримка різних форматів зберігання. Архівування даних повинно підтримувати різні формати зберігання (наприклад, компресовані файли, сховища з відкритим кодом або спеціалізовані архіви), що дозволить ефективно використовувати різні технічні рішення і знижувати витрати на зберігання;

– моніторинг та звітність. Необхідно забезпечити механізм моніторингу процесів архівування та автоматичне створення звітів про виконання архівування. Це дозволить адміністратору системи стежити за ефективністю роботи і своєчасно виявляти будь-які проблеми або збої;

– інтерфейси для управління архівами. Користувачі повинні мати змогу зручно управляти архівами через інтерфейси, які дозволяють виконувати операції з архівованими даними: відновлення, переміщення, видалення або доступ до конкретних версій даних;

– розширені можливості пошуку. Оскільки архівовані дані можуть бути великими за обсягом, система повинна підтримувати розширені можливості пошуку, що дозволяють швидко знаходити потрібні архівовані файли або записи.

Технічні вимоги до системи автоматизації архівування передбачають використання ефективних алгоритмів стиснення даних, що дозволяють зменшити обсяг архівованих файлів і оптимізувати процес зберігання, знижуючи витрати на ресурси і простір. Крім того, система повинна підтримувати розподілені системи зберігання, що дає змогу розподіляти навантаження на різні носії або хмарні сховища, забезпечуючи доступ до даних у будь-який час. Враховуючи різноманітність баз даних, система має бути сумісною з реляційними, нереляційними та іншими типами баз даних, що використовуються в організації. Також важливим аспектом є наявність механізмів відновлення після збоїв, що гарантують відновлення даних у разі непередбачених ситуацій, таких як апаратні відмови або помилки програмного забезпечення.

Інтерфейс користувача повинен бути інтуїтивно зрозумілим і зручним для використання, що дозволяє легко налаштовувати параметри архівування, такі як час виконання, періодичність та обсяг даних. Користувачі повинні мати можливість швидко отримати доступ до архівованих даних для виконання операцій відновлення, а також для аналізу вмісту архівів. Крім того, інтерфейс повинен забезпечувати систему оповіщень, яка інформує про успішне

завершення архівування або виникнення проблем, а також надавати зручні інструменти для відновлення даних через прості та зрозумілі інтерфейси.

1.4 Висновки

У першому розділі проведено аналіз теоретичних основ архівування даних. Розглянуто сучасні методи та технології архівування, їх переваги та обмеження, а також актуальність автоматизації цих процесів у контексті зростання обсягів інформації. Особливу увагу приділено реляційним базам даних, які є основою для багатьох сучасних інформаційних систем, зокрема їх властивостям, що впливають на реалізацію архівування. Проведений аналіз дозволив сформулювати вимоги до систем автоматизації архівування, зокрема щодо функціональності, інтеграції з існуючими базами даних, забезпечення безпеки даних та зручності використання. Отримані результати стали теоретичним підґрунтям для подальшого проектування та реалізації програмного модуля, спрямованого на вирішення задачі автоматизації архівування.

2 ОБҐРУНТУВАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

У сучасних умовах розвитку інформаційних технологій обсяг даних, що зберігаються та обробляються в організаціях, постійно збільшується. Це створює нові виклики у процесах збереження та управління інформацією, що вимагає впровадження ефективних інструментів для архівування. У зв'язку з цим автоматизація процесів архівування даних набуває все більшої актуальності. Це дозволяє знизити людську участь у рутинних операціях, забезпечити безпеку та доступність інформації, а також зменшити витрати на зберігання та обробку великих обсягів даних.

Основним завданням даного розділу є визначення напрямків і підходів до автоматизації архівування, а також вибір методів і засобів для розробки програмного модуля, що відповідатиме вимогам сучасних інформаційних систем. У розділі буде детально розглянуто постановку задачі автоматизації архівування, вибір інструментів і технологій для її реалізації, а також проектування структури рішення, що дозволить створити ефективну систему архівування даних, здатну інтегруватися з іншими інформаційними системами.

У першому підрозділі буде сформульовано основне завдання автоматизації архівування, визначено його ключові етапи та пріоритети, а також розглянуто вплив автоматизації на процеси зберігання та доступу до даних. Другий підрозділ присвячений вибору методів і засобів, що будуть використані для розробки програмного модуля, а також аналізу технічних і функціональних вимог до цього модуля. У третьому підрозділі буде представлено проектування структури рішення, що охоплює архітектуру системи, її компоненти та принципи взаємодії між ними.

2.1 Постановка задачі автоматизації процесів архівування

У зв’язку з експоненційним зростанням обсягів даних, що зберігаються в інформаційних системах, виникає необхідність ефективного управління цими даними протягом усього їх життєвого циклу. Архівування є важливим етапом цього процесу, оскільки дозволяє зберігати неактивні або застарілі дані в умовах обмежених ресурсів зберігання та забезпечити їх доступність у разі потреби. Однак, традиційні методи архівування, що базуються на ручному втручанні або частковій автоматизації, часто призводять до помилок, затримок та високих витрат на обробку великих обсягів інформації. Тому автоматизація процесів архівування стає необхідним кроком для забезпечення більшої надійності, швидкості та ефективності зберігання даних.

Задача автоматизації архівування полягає в розробці та впровадженні системи, яка забезпечить автоматичне виконання всіх операцій архівування на основі заздалегідь визначених правил та алгоритмів. Система повинна включати можливості для регулярного архівування даних з можливістю налаштування часу та обсягу архівування, а також підтримувати функції для відновлення даних у разі необхідності. Важливою вимогою є забезпечення безпеки архівованих даних, що вимагає застосування методів шифрування та механізмів контролю доступу.

Крім того, автоматизація має забезпечити інтеграцію з існуючими інформаційними системами, що зберігають дані, таких як реляційні бази даних. Це дозволить організації безперешкодно зберігати дані без втручання користувачів, оптимізуючи ресурси та знижуючи навантаження на IT-персонал. В результаті впровадження автоматизованого рішення організація отримає систему, що знижує ризики втрат даних, підвищує ефективність архівування та забезпечує легкість у доступі до необхідної інформації.

2.2 Вибір методів і засобів для розробки програмного модуля

Для розробки програмного модуля, що автоматизує процес архівування даних, важливим кроком є вибір оптимальних методів та засобів, які забезпечать ефективну та надійну реалізацію всіх функцій системи. Вибір інструментів та технологій залежить від кількох факторів, таких як типи баз даних, що використовуються в організації, вимоги до продуктивності та безпеки, а також потреба в інтеграції з існуючими інформаційними системами:

– мови програмування та платформи. Для розробки програмного модуля вибрано використання мови програмування Java, оскільки вона має низку переваг для створення кросплатформених рішень, що підтримуються на різних операційних системах, таких як Windows, Linux та macOS. Крім того, Java забезпечує надійність та масштабованість додатків, що є критичними для розробки програмного забезпечення для архівування даних. Для роботи з реляційними базами даних буде використано бібліотеку JDBC (Java Database Connectivity), яка дозволяє ефективно інтегрувати програмний модуль з різними СУБД, такими як MySQL, PostgreSQL або Oracle, і виконувати необхідні операції з базою даних [8];

– алгоритми архівування та стиснення. Однією з основних задач модуля є вибір ефективних алгоритмів для архівування та стиснення даних. В якості основного алгоритму архівування буде використано ZIP (за допомогою бібліотеки Java.util.zip), оскільки цей формат забезпечує достатній рівень стиснення, простоту інтеграції та підтримку багатьма інструментами. Для досягнення високого рівня стиснення можна застосувати LZ4 або Gzip, залежно від вимог до швидкості архівування та обсягу даних. Такі алгоритми дозволяють швидко обробляти великі обсяги інформації, що важливо для корпоративних середовищ з великими обсягами даних;

– безпека та шифрування даних. Забезпечення безпеки архівованих даних є важливою вимогою, що вимагає застосування шифрування для захисту інформації від несанкціонованого доступу. Для цієї мети буде використовуватися стандартне шифрування на основі алгоритмів AES (Advanced Encryption Standard), що забезпечує високий рівень безпеки і швидкість шифрування/дешифрування. Для роботи з шифруванням буде використана бібліотека Java Cryptography Architecture (JCA), яка надає засоби для реалізації алгоритмів шифрування та захисту даних [9];

– інтерфейс користувача. Для створення зручного та інтуїтивно зрозумілого інтерфейсу користувача буде використано фреймворк Vue.js, який дозволяє створювати сучасні веб-додатки з динамічними та інтерактивними інтерфейсами. Vue.js надає простоту інтеграції, гнучкість у розробці і підтримку компоненто-орієнтованого підходу, що дозволить легко створювати інтерфейс для керування процесами архівування [6]. Інтерфейс надасть можливість налаштовувати параметри архівування, переглядати архіви, а також відновлювати дані з архівів через інтуїтивно зрозумілий веб-інтерфейс. Використання Vue.js дозволяє забезпечити зручний доступ до модуля з будь-якої платформи, що має веб-браузер, забезпечуючи кросплатформеність;

– інтеграція з існуючими інформаційними системами. Одним із важливих аспектів є інтеграція програмного модуля з існуючими інформаційними системами та базами даних. Для цього буде використовуватися стандартний протокол JDBC для взаємодії з реляційними базами даних. Крім того, розроблений модуль має бути здатний працювати з різними типами зберігання даних, включаючи локальні файлові системи, хмарні сховища або розподілені системи зберігання, що дозволить організації використовувати зручні для неї технології для зберігання архівованих даних;

– тестування та верифікація. Для забезпечення високої якості програмного забезпечення будуть використовуватися стандартні методи тестування, такі як unit-тестування з використанням бібліотеки Junit [10]. Це дозволить перевіряти

правильність роботи окремих компонентів системи, таких як алгоритми стиснення, шифрування, а також взаємодія з базою даних [7]. Крім того, буде проведено тестування продуктивності та навантаження, щоб переконатися в здатності системи ефективно працювати з великими обсягами даних.

2.3 Проектування структури рішення

Проектування структури рішення є одним із ключових етапів у процесі розробки програмного модуля автоматизації архівування даних. На цьому етапі визначається основна архітектура системи, її компоненти, взаємодія між ними та необхідні технології для реалізації кожної з частин рішення.

Одним із основних аспектів проектування є поділ системи на окремі логічні блоки. Це дозволяє не тільки спростити процес розробки та тестування, але й забезпечити гнучкість та масштабованість системи в майбутньому. Основні компоненти системи будуть включати: модуль архівування, модуль шифрування, модуль роботи з базою даних, веб-інтерфейс користувача та модуль для управління налаштуваннями та моніторингу [11].

Модуль архівування є основним компонентом системи. Він відповідає за процес стиснення даних та їх збереження у відповідному форматі архіву. Відповідно до обраної архітектури, модуль буде реалізовано з використанням алгоритмів архівування ZIP та Gzip, що дозволяє досягти хорошого компромісу між швидкістю архівування та рівнем стиснення. Модуль також буде підтримувати можливість роботи з великими обсягами даних, що важливо для корпоративних систем.

Для забезпечення безпеки даних передбачено створення окремого модуля шифрування, який буде працювати за алгоритмами AES. Модуль забезпечить захист архівованих даних від несанкціонованого доступу, а також підтримуватиме функції дешифрування при відновленні даних з архівів.

Модуль взаємодії з базою даних буде відповідати за збереження та отримання метаданих архівів. Він буде реалізований за допомогою JDBC, що дозволяє інтегрувати систему з різними реляційними базами даних (MySQL, PostgreSQL та ін.). Модуль забезпечить зберігання інформації про архіви (наприклад, дати створення, розмір, тип стиснення тощо) та дозволить здійснювати ефективний пошук і фільтрацію архівів за різними критеріями.

Веб-інтерфейс користувача є важливим компонентом системи, що дозволяє кінцевим користувачам керувати процесами архівування та доступом до архівованих даних. Для створення інтерфейсу вибрано використання фреймворку Vue.js, що дозволить створювати сучасний, динамічний та зручний веб-додаток. Користувачі зможуть налаштовувати параметри архівування, переглядати метадані архівів, а також відновлювати дані з архівів без необхідності взаємодіяти з командним рядком або іншими складними інтерфейсами.

Модуль для налаштувань дозволить користувачам управляти параметрами архівування, а також здійснювати моніторинг процесів архівування та відновлення даних. Цей компонент забезпечить можливість налаштування алгоритмів архівування, рівнів стиснення, а також встановлення обмежень на максимальний розмір архіву. Крім того, він буде надавати інформацію про стан виконання архівних операцій, включаючи їх тривалість і поточний прогрес.

Рішення буде побудоване на архітектурі клієнт-сервер, де серверна частина відповідатиме за обробку запитів та виконання основних операцій з архівування та шифрування, а клієнтська частина забезпечить зручний інтерфейс для взаємодії з користувачами. Таке розподілене рішення дозволить легко масштабувати систему та забезпечити високу продуктивність при обробці великих обсягів даних.

Вся система буде інтегрована з існуючими інформаційними системами організації, що дозволить забезпечити автоматизацію процесу архівування без

потреби у втручанні з боку користувачів. Вибір технологій та інструментів забезпечить гнучкість, надійність і безпеку при роботі з архівованими даними.

Проектування структури рішення включає в себе розробку модулів та компонентів, що гарантують ефективність, безпеку та зручність використання програмного модуля автоматизації архівування. Використання перевірених технологій дозволяє створити рішення, яке відповідає вимогам до продуктивності, безпеки та простоти у використанні.

2.4 Висновки

У другому розділі здійснено обґрунтування та постановку завдання для розробки програмного модуля автоматизації архівування даних. Сформульовано ключові аспекти задачі, включно з необхідністю забезпечення ефективного управління архівами, автоматизацією рутинних операцій, збереженням цілісності та безпеки даних.

На основі аналізу обрано методи і засоби реалізації, серед яких використання мови програмування Java для серверної частини, Vue.js для створення зручного інтерфейсу користувача та бази даних MySQL як надійного засобу збереження інформації. Проведено проектування структури рішення, у рамках якого визначено основні компоненти модуля та їхню взаємодію. Розроблений підхід закладає основу для подальшої реалізації та тестування програмного забезпечення, яке відповідатиме поставленим вимогам і забезпечить високу ефективність виконання архівних операцій.

3 РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ

Розробка програмного модуля автоматизації архівування даних є центральною частиною цієї роботи. На основі визначених у попередніх розділах вимог, обраних методів і технологій розпочинається безпосередній процес створення програмного забезпечення, що забезпечить ефективність, зручність та безпеку роботи з даними.

Цей розділ присвячений аналізу функціональних вимог до модулю, проектуванню його архітектури, розробці алгоритмів, реалізації основних компонентів та вирішенню завдань, пов’язаних із забезпеченням безпеки і цілісності даних. У процесі розробки враховуються сучасні підходи до проектування програмних рішень, такі як модульність, масштабованість та відповідність принципам безпеки.

Перший підрозділ розглядає функціональні вимоги до модуля, визначаючи основні завдання, які він повинен виконувати. У другому підрозділі наводиться детальний опис архітектури, яка буде реалізована у модулі. У третьому підрозділі представлено алгоритми, розроблені для виконання процесу архівування, враховуючи ефективність та точність обробки даних. Четвертий підрозділ присвячений практичній реалізації основних компонентів модуля, зокрема модулів для архівування, шифрування та взаємодії з базою даних. У п’ятому підрозділі розглянуто методи забезпечення безпеки та цілісності даних, що є ключовим аспектом системи.

Цей розділ формує основу для подальшого тестування та впровадження розробленого програмного забезпечення, демонструючи його можливості у виконанні завдань архівування та забезпечення надійного зберігання даних.

3.1 Аналіз функціональних вимог до модуля

Аналіз функціональних вимог до програмного модуля є одним із основних етапів на шляху до розробки програмного забезпечення. У цьому підрозділі розглядаються ключові функціональні вимоги, які визначають основні завдання, що мають бути вирішені програмним модулем для ефективного автоматизованого архівування даних у реляційних базах даних. Задоволення цих вимог дозволить створити потужне та гнучке рішення, яке відповідає потребам користувачів і забезпечує безпеку та цілісність архівованих даних.

Однією з основних функцій програмного модуля є архівування даних. Модуль повинен підтримувати архівування різних типів файлів та даних, включаючи текстові, бінарні, мультимедійні файли, а також складні структури, як-от бази даних або великі колекції файлів. Важливою вимогою є підтримка форматів архівів, які дозволяють досягти оптимального співвідношення між швидкістю архівування та рівнем стиснення даних. Формати ZIP та GZIP були обрані через їх високу популярність, швидкість роботи та широке застосування у корпоративних середовищах. Крім того, модуль має бути здатний здійснювати архівування як окремих файлів, так і цілих директорій з підкаталогами, зберігаючи структуру директорій, а також відповідну метаінформацію (розмір файлів, дата модифікації, права доступу).

Для забезпечення конфіденційності архівованих даних передбачено впровадження механізму шифрування. Це надзвичайно важлива функціональність, оскільки дані, що зберігаються в архівах, можуть містити конфіденційну або чутливу інформацію, і її захист є обов’язковим. Модуль буде використовувати алгоритм AES (Advanced Encryption Standard), що гарантує високий рівень безпеки, а також підтримку різних режимів шифрування, таких як ECB (Electronic Codebook) або CBC (Cipher Block Chaining). Окрім цього, для шифрування буде реалізовано механізм генерації та збереження ключів

шифрування, що дасть можливість здійснювати дешифрування даних лише за наявності відповідного ключа. Важливо, щоб користувачі мали можливість налаштовувати рівень шифрування в залежності від своїх потреб.

Модуль має забезпечити інтеграцію з реляційною базою даних для зберігання метаданих архівів. До метаданих архіву відносяться різноманітні параметри, зокрема дата створення архіву, його розмір, формат, використані алгоритми стиснення та шифрування, а також статус виконання архівної операції. Зберігання метаданих у базі даних дозволяє здійснювати пошук, фільтрацію та сортування архівів за різними параметрами, що значно спрощує роботу з великою кількістю архівів. Для реалізації цієї функціональності буде використовуватися стандарт SQL, що забезпечує зручний доступ до метаданих через запити. Оскільки система повинна працювати з великими обсягами даних, архітектура бази даних має бути оптимізованою для забезпечення швидкого доступу до архівів навіть за умов великого навантаження.

Зручний інтерфейс користувача є ключовим елементом програмного модуля, оскільки він забезпечує простоту та ефективність взаємодії користувача з системою. Веб-інтерфейс, що буде розроблений для цього модуля, має бути інтуїтивно зрозумілим, з чітко структурованими розділами для налаштування архівування, перегляду архівів і управління процесом архівування. Користувач повинен мати можливість налаштовувати параметри архівування, такі як вибір формату архіву, рівень стиснення, налаштування шифрування, а також переглядати вже створені архіви, відновлювати файли або цілі архіви за допомогою зручного інтерфейсу. Важливою вимогою є підтримка багатьох платформ для роботи з інтерфейсом, включаючи робочі станції, ноутбуки та мобільні пристрої. Для реалізації веб-інтерфейсу вибрано фреймворк Vue.js, який забезпечує високу продуктивність і легкість у підтримці сучасних веб-застосунків [12][13].

Модуль повинен забезпечити функціональність для відновлення даних з архівів. Користувачі повинні мати можливість як повністю відновлювати архів,

так і вибірково відновлювати окремі файли або директорії, що містяться в архіві. Вибіркове відновлення дасть змогу економити час і ресурси при необхідності доступу лише до окремих частин архівованих даних. Для реалізації цієї функції буде передбачено використання спеціальних алгоритмів, що дозволяють швидко і без помилок знаходити потрібні файли в архіві і відновлювати їх без порушення цілісності всього архіву.

Моніторинг процесів архівування та відновлення є важливою складовою частиною модуля, оскільки він дозволяє користувачам стежити за виконанням операцій у реальному часі. Система моніторингу повинна надавати інформацію про поточний стан кожної операції, наприклад, відсоток виконання, залишковий час, а також повідомлення про можливі помилки або збої під час виконання операцій. Це дасть можливість своєчасно реагувати на будь-які проблеми, що можуть виникнути під час виконання архівування або відновлення.

Однією з ключових вимог є забезпечення високої продуктивності роботи модуля. Зокрема, модуль повинен забезпечувати можливість ефективної обробки великих обсягів даних без істотних затримок у часі. Для цього будуть застосовані такі технології, як мультипотокове архівування та оптимізація алгоритмів стиснення для максимального використання ресурсів комп’ютера. Крім того, важливо забезпечити використання пам'яті таким чином, щоб навіть при роботі з великими архівами система не виходила за межі ресурсів, доступних на робочій станції користувача.

Для забезпечення автоматизації архівування без необхідності втручання користувача, модуль має бути здатний інтегруватися з іншими інформаційними системами підприємства. Це може включати як внутрішні процеси (наприклад, архівування даних з баз даних), так і зовнішні системи, зокрема для резервного копіювання даних або зберігання в хмарі. Модуль має підтримувати налаштування автоматичного архівування за розкладом або в залежності від подій у системі (наприклад, при зміні даних або появі нових файлів).

Для забезпечення прозорості і підзвітності процесів архівування передбачається функціональність журналювання всіх операцій, що здійснюються з архівами. Журнал має містити інформацію про кожну операцію архівування, шифрування, відновлення або помилки, що виникли під час виконання. Звіти про виконання операцій будуть доступні для перегляду і зберігання, що дозволяє користувачам і адміністраторам здійснювати моніторинг стану системи.

3.2 Проектування архітектури програмного забезпечення

Проектування архітектури програмного забезпечення є критично важливим етапом розробки, оскільки від цього етапу залежить загальна ефективність, гнучкість і масштабованість системи. Архітектура визначає основні компоненти програмного забезпечення, взаємодію між ними, а також способи обробки даних, що забезпечують надійне та ефективне функціонування програмного модуля. У даному випадку проектування архітектури орієнтовано на створення модуля для автоматизованого архівування даних, що інтегрується з реляційною базою даних та іншими корпоративними системами.

Архітектура програмного модуля буде побудована на принципах монолітності, модульності та відокремленості інтерфейсів. Вся система буде складатися з одного монолітного додатка, що включає всі компоненти, необхідні для виконання функцій архівування. Це дозволить спростити взаємодію між компонентами та полегшить розгортання програмного модуля. Основні компоненти архітектури наступні:

– система архівування: цей компонент відповідає за стиснення і архівування файлів, а також за створення архівів різних форматів, таких як ZIP або GZIP. Він також буде здійснювати шифрування та інтеграцію з базою даних для збереження метаданих архіву;

– компонент взаємодії з базою даних: цей компонент відповідає за управління метаданими архівів, які зберігаються у реляційній базі даних. Зокрема, метадані включатимуть інформацію про розміри файлів, дату архівування, формат і статус архіву. Для цієї частини буде використовуватися стандарт SQL, що дозволяє ефективно взаємодіяти з реляційними базами даних, такими як PostgreSQL або MySQL;

– інтерфейс користувача: для взаємодії з користувачами буде розроблено веб-інтерфейс, який забезпечить доступ до функцій архівування та відновлення даних. Веб-інтерфейс буде реалізовано за допомогою Vue.js, що забезпечить високу продуктивність і зручний інтерфейс для користувачів. Інтерфейс дозволить налаштовувати параметри архівування, переглядати архіви, запускати процеси відновлення та отримувати звіти про виконання операцій;

– система моніторингу: окремий компонент буде забезпечувати моніторинг процесів архівування та відновлення, інформуючи користувачів про поточний стан операцій у реальному часі. Це дасть змогу оперативно реагувати на збої та відхилення від нормального виконання процесу;

– компонент безпеки: з метою захисту даних в архівах буде передбачено використання сучасних методів шифрування (наприклад, AES). Цей компонент забезпечить високий рівень безпеки даних, дозволяючи шифрувати архіви та здійснювати безпечне збереження та відновлення файлів.

Щоб краще зрозуміти, як саме відбувається взаємодія між компонентами, на рис 3.1 наведена діаграма, що показує основні частини системи та їх зв'язки. Діаграма ілюструє, як інтерфейс користувача, бізнес-логіка, доступ до даних та інші елементи працюють разом для реалізації процесів архівування та відновлення.

На діаграмі видно, що всі основні компоненти програми, тобто інтерфейс користувача, бізнес-логіка та база даних, взаємодіють через єдиний програмний процес. Інтерфейс користувача забезпечує введення даних і взаємодію з системою, бізнес-логіка обробляє ці дані і виконує операції архівування та

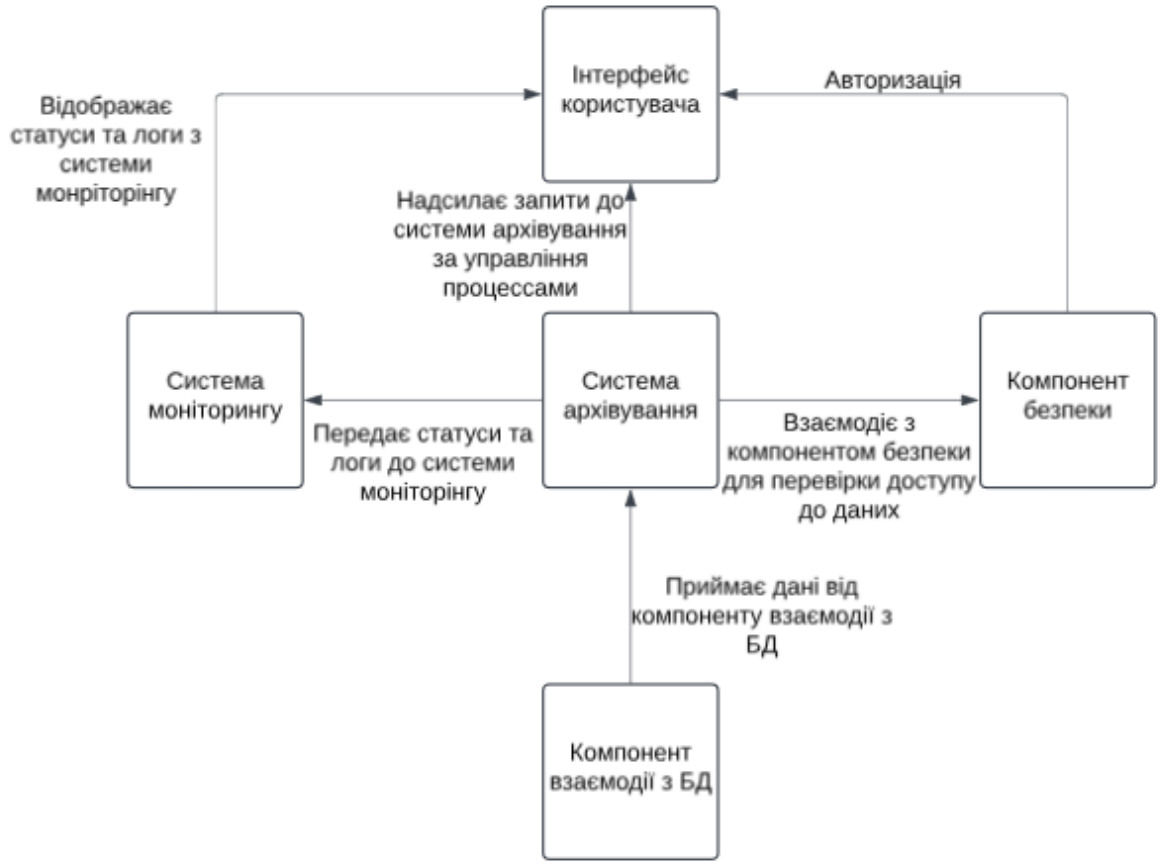

Рисунок 3.1 – Діаграма взаємодії між компонентами модуля

відновлення, а база даних зберігає метадані архівів та інші необхідні дані. Кожен компонент має чітко визначену роль, але всі вони тісно пов'язані в рамках єдиного додатку.

Архітектура програмного забезпечення для цього модуля буде побудована за принципом монолітної архітектури. У такій архітектурі всі компоненти програми інтегровані в один єдиний додаток, що зменшує складність у взаємодії між компонентами, оскільки всі частини системи функціонують у межах одного середовища. Це дозволяє спростити процес розгортання і оновлення, а також знижує вимоги до підтримки декількох сервісів.

Для реалізації архітектури програмного модуля було обрано наступні технології та інструменти, кожен з яких відповідає вимогам щодо надійності, ефективності та масштабованості системи:

– мова програмування: Java. Java була обрана як основна мова програмування для розробки програмного модуля через свою надійність, масштабованість і високу продуктивність. Java є однією з найбільш популярних мов для розробки корпоративних додатків завдяки її великій екосистемі, підтримці багатозадачності та вбудованій можливості обробки великих обсягів даних. Крім того, наявність численних бібліотек і фреймворків, таких як Spring для реалізації бізнес-логіки, дозволяє швидко створювати і підтримувати додатки;

– фреймворк для інтерфейсу: Vue.js. Для створення веб-інтерфейсу був обраний Vue.js завдяки його легкості в освоєнні, швидкості роботи та високій продуктивності. Vue.js є одним із найпопулярніших фреймворків для побудови інтерактивних інтерфейсів, що дозволяє створювати динамічні веб-застосунки з мінімальними витратами ресурсів. Його проста інтеграція з іншими технологіями, низька складність підтримки та великий набір інструментів для розробки дозволяють ефективно реалізувати функціональність для користувачів модуля, зокрема, для налаштування архівування, перегляду архівів та запуску процесів відновлення;

– база даних: MySQL. MySQL була обрана як реляційна база даних через її популярність, стабільність та підтримку широкого спектру функцій для роботи з великими обсягами даних. MySQL є відкритим програмним забезпеченням з високою продуктивністю, що дозволяє ефективно працювати з метаданими архівів, зокрема для збереження інформації про архівовані файли, їх статус та дату архівування. Крім того, MySQL підтримує стандарт SQL, що забезпечує простоту та ефективність роботи з даними, а також можливість інтеграції з іншими системами, що можуть використовуватись в рамках підприємства. Оскільки MySQL активно використовується у багатьох комерційних та відкритих проектах, вона є перевіреним і надійним вибором для цієї системи [14];

– шифрування: AES (Advanced Encryption Standard). Для забезпечення безпеки даних у архівах вибрано використання стандарту AES для шифрування. AES є одним з найбільш надійних алгоритмів шифрування, що відповідає вимогам до конфіденційності даних і захисту від несанкціонованого доступу. Оскільки безпека даних є важливим аспектом при архівуванні, AES забезпечує високий рівень захисту та підтримує достатню швидкість шифрування для обробки великих обсягів файлів без значних втрат продуктивності.

Таким чином, вибір цих технологій дозволяє забезпечити стабільність, ефективність, безпеку і масштабованість програмного модуля, що має вирішальне значення для реалізації автоматизованої системи архівування даних.

Проектоване програмне забезпечення буде організовано в три основні шари:

– шар представлення (Frontend): цей шар відповідає за інтерфейс користувача. З ним працюватимуть кінцеві користувачі для налаштування архівування, перегляду архівів та запуску процесів відновлення;

– шар бізнес-логіки (Backend): в цьому шарі реалізуються основні алгоритми архівування, шифрування та взаємодії з базою даних;

– шар даних (Database): на цьому шарі зберігаються всі метадані архівів, а також виконуються операції з базою даних, що включають пошук, збереження та оновлення даних.

Для розробки програмного модуля буде використовуватися клієнт-серверна архітектура. Клієнт (веб-браузер) взаємодіятиме з сервером через API, що забезпечить централізовану обробку запитів та ефективну роботу з даними. Такий підхід дозволить забезпечити доступність сервісу для широкої аудиторії користувачів, використовуючи сучасні веб-технології та інтерфейси для взаємодії з серверною частиною системи.

Проектування архітектури програмного забезпечення є важливим етапом, який визначає ефективність, гнучкість і масштабованість майбутнього модуля. Завдяки використанню монолітної архітектури, програмний модуль буде

простим у розгортанні, обслуговуванні і масштабуванні для вирішення завдань автоматизації архівування даних.

3.3Розробка алгоритмів для архівування

Розроблений алгоритм призначений для автоматизації процесу архівації інформації в реляційній базі даних. Він дозволяє вибирати дані на основі заданих параметрів, переміщувати їх до архівного сховища та застосовувати методи стиснення для оптимізації використання ресурсів зберігання.

Алгоритм працює за принципом послідовного виконання операцій з даними, що включає:

– підключення до основної бази даних;

– вибір даних, які підлягають архівації, на основі вхідних умов;

– переміщення вибраних даних до архівної бази або зовнішнього сховища;

– стиснення архівованих даних для зменшення обсягу зберігання;

– видалення архівованих записів з основної бази після успішного переміщення.

Для забезпечення гнучкості алгоритм приймає наступні параметри:

– URL основної бази даних. Це адреса або шлях до бази даних, з якої будуть вибиратися дані для архівування. Важливо, щоб алгоритм міг підключатися до цієї бази через відповідні права доступу, тому цей параметр також може включати облікові дані для підключення (користувач і пароль) або механізм автентифікації;

– список таблиць. Це перелік таблиць, які повинні бути піддані архівуванню. Користувач може вибрати конкретні таблиці або задати шаблон (наприклад, всі таблиці з префіксом "log_"). Це дозволяє алгоритму працювати лише з потрібними даними, не впливаючи на інші таблиці, що можуть мати важливу або часто оновлювану інформацію;

– умови вибору даних. Це SQL-умови, які вказують, які саме записи з таблиць повинні бути архівовані. Наприклад, умови можуть включати дату створення або оновлення запису, деякий статус чи інші критерії;

– URL архівної бази. Це місце, куди будуть переміщені архівовані дані. Як і в випадку з основною базою, URL архівної бази визначає місцезнаходження архіву, який може бути на іншому сервері або в окремій базі даних. У цьому параметрі можуть бути вказані налаштування підключення до архіву (наприклад, ім'я архівної бази, хост і порт, а також автентифікація). Важливою є також можливість налаштування автоматичного створення архівних таблиць або визначення наявності архіву, щоб уникнути дублювання даних;

– алгоритм стиснення. Цей параметр визначає метод стиснення, що буде використаний для зменшення обсягу даних під час архівування. Серед доступних методів є gzip, zstd та lz4.

Алгоритм реалізує кілька етапів, кожен з яких відповідає за конкретну частину процесу архівації.

На першому етапі відбувається ініціалізація параметрів. Алгоритм отримує всі необхідні вхідні параметри, які можуть бути передані користувачем або конфігураційним файлом. Це включає URL основної бази даних, список таблиць для архівації, умови вибору даних, URL архівної бази даних і метод стиснення даних. Після отримання параметрів здійснюється їх перевірка на коректність: чи існують зазначені таблиці в основній базі даних, чи відповідають умови вибору даних формату SQL-запиту, чи є доступ до обох баз даних та чи правильно вказано метод стиснення. Потім алгоритм встановлює з’єднання з основною базою даних для виконання подальших операцій, включаючи аутентифікацію і перевірку доступу.

Другий етап алгоритму починається з формування SQL-запиту. Алгоритм отримує умови, визначені на попередньому етапі (наприклад, дата створення записів або інші фільтри), і на їх основі формує запит для вибору відповідних даних з основної бази даних. Умови можуть бути складними і включати кілька

критеріїв, наприклад, вибір записів, що були створені до певної дати, або записи, що належать до певної категорії.

Після формування SQL-запиту алгоритм виконує його на основній базі даних. В залежності від обсягу даних і складності запиту, виконання може зайняти різний час. Під час виконання запиту база даних шукає всі записи, що відповідають заданим критеріям, і повертає їх як результат виконання запиту. Це може включати численні рядки, які зберігаються у тимчасовому буфері для подальшої обробки.

Збереження вибраних даних у тимчасовому буфері є важливим етапом. Цей буфер служить своєрідним проміжним зберіганням даних перед їхнім подальшим переміщенням в архівну базу даних. Оскільки кількість записів може бути великою, алгоритм використовує буфер для того, щоб зберігати їх у пам'яті і потім передавати пакетами до архіву. Це дозволяє уникнути перевантаження пам’яті, оскільки дані обробляються не всі одразу, а порціями. Також використання буфера дозволяє ефективно працювати з великими обсягами даних, знижуючи навантаження на систему.

Третій етап алгоритму відповідає за переміщення даних до архівного сховища. Після того як дані вибрані та збережені у тимчасовому буфері, алгоритм починає процес їх перенесення до архівної бази даних. Спочатку алгоритм встановлює з’єднання з архівною базою даних, що дозволяє взаємодіяти з цією базою для подальших операцій. Якщо архівна база даних розташована на іншому сервері або в іншому середовищі, можуть бути використані спеціалізовані методи з’єднання через мережу, а також забезпечення безпечного з’єднання, включаючи шифрування даних.

Наступний крок це перевірка наявності необхідних таблиць у архівній базі даних. Алгоритм порівнює структуру таблиць, що зберігають архівовані дані, із тим, що вже є в архівній базі. Якщо виявляється, що потрібні таблиці відсутні, алгоритм автоматично створює їх, підтримуючи правильну структуру даних, яка відповідає вимогам. Створення таблиць включає в себе визначення правильних

типів даних для кожного поля, а також забезпечення можливості зберігання великих обсягів даних без втрати ефективності.

Після створення необхідних таблиць або підтвердження їх наявності, алгоритм переходить до безпосереднього процесу перенесення даних. Збережені у тимчасовому буфері записи передаються в архівну базу даних. Оскільки обсяг даних може бути великим, перенесення здійснюється не за один раз, а пакетами. Алгоритм розбиває вибрані дані на невеликі порції, що дозволяє уникнути перевантаження пам’яті і зберігає стабільність роботи системи. Кількість записів у кожному пакеті може бути налаштована в залежності від доступних ресурсів пам'яті або параметрів продуктивності.

Під час переміщення даних алгоритм виконує перевірку успішності кожної операції. Якщо дані успішно записуються в архівну базу, алгоритм переходить до наступного пакету. Якщо ж виникає помилка під час запису, алгоритм намагається вирішити проблему, спробувавши повторити операцію, або ж записує помилку в журнал, якщо перенесення не вдалося після кількох спроб. Це дозволяє мінімізувати ризик втрати даних і забезпечити цілісність процесу перенесення.

Процес переміщення може включати додаткові перевірки, такі як контроль за дотриманням обмежень з цілісності даних, наприклад, перевірка унікальності значень або відповідності зовнішнім ключам. Якщо виявляються порушення цілісності даних, алгоритм може або ігнорувати ці записи, або надавати повідомлення про помилки.

Після успішного переміщення всіх даних, алгоритм завершує етап перенесення та готується до подальших операцій, таких як стиснення архівованих даних або видалення їх з основної бази.

Четвертий етап алгоритму полягає в стисненні архівованих даних після їх переміщення до архівної бази даних. Цей етап є важливим для зниження обсягу даних і забезпечення їх економічного зберігання або ефективного

транспортування. Алгоритм приступає до стиснення після того, як дані успішно переміщені і збережені в архівному сховищі.

Процес стиснення починається з вибору методу стиснення, що був заданий на початкових етапах налаштувань. Це може бути один з кількох доступних алгоритмів, gzip, zstd або lz4. Вибір методу залежить від вимог до швидкості стиснення та ступеня стиснення.

Після того як вибрано відповідне стиснення, алгоритм приступає до обробки кожної частини даних. Якщо архівовані дані зберігаються у вигляді кількох окремих файлів або таблиць, алгоритм обробляє їх по черзі, виконуючи стиснення кожного файлу окремо. Це дозволяє зберегти цілісність кожної частини архівованих даних і зменшує ймовірність помилок при стисненні великих обсягів інформації. Кожен стиснений файл може бути записаний в окремий архів або комбінований в один великий архівний файл, в залежності від вимог до організації збереження.

Залежно від методу стиснення, алгоритм може здійснювати додаткові оптимізації. Наприклад, якщо використовується алгоритм, що підтримує паралельне стиснення, алгоритм може розподіляти процес стиснення на кілька потоків, що забезпечить швидше виконання операції при обробці великих обсягів даних. Вибір рівня стиснення також є важливим аспектом: високий рівень стиснення зазвичай займає більше часу, але значно зменшує обсяг даних, тоді як низький рівень стиснення дозволяє швидше завершити операцію, але забезпечує менше зниження обсягу.

Стиснення може бути здійснене на різних етапах. Наприклад, дані можуть бути стиснуті безпосередньо після їхнього запису в архівну базу даних, або ж стиснення може бути відкладено на певний час для того, щоб спочатку забезпечити більш швидке перенесення даних і зменшити навантаження на систему. В разі обробки дуже великих обсягів даних, алгоритм може також ділити дані на блоки і стиснювати їх по черзі, щоб не перевантажити пам’ять або не викликати проблеми з продуктивністю.

Після того як стиснення завершено, отриманий архів зберігається на вибраній платформі для зберігання. Це може бути файловою системою локального сервера, спеціалізованим хмарним сховищем або іншими методами зберігання. Вибір місця для збереження архіву залежить від вимог до доступності, надійності зберігання та швидкості доступу до архівованих даних. Місце зберігання також може бути налаштоване на етапі конфігурації.

Таким чином, цей етап включає в себе не лише власне стиснення, але й оптимізацію процесу з урахуванням параметрів системи, вибору відповідного алгоритму і умов зберігання архівованих даних.

Четвертий етап алгоритму полягає в стисненні архівованих даних після їх експорту з архівної бази даних. Спочатку дані з архівної бази експортуються у файл SQL за допомогою утиліти mysqldump. Цей інструмент дозволяє створити текстовий файл, що містить усі необхідні SQL-запити для відновлення бази даних, включаючи структуру таблиць та вміст [15].

Після того як файл SQL було успішно створено, алгоритм переходить до стиснення цього файлу за допомогою одного з вибраних користувачем методів стиснення: gzip, zstd або lz4. Вибір методу стиснення залежить від налаштувань, заданих на етапі запиту. Після того як стиснення завершено, отриманий архів зберігається локальній файловій системі.

Таким чином, на цьому етапі алгоритм спочатку виконує експорт даних із архівної бази даних за допомогою mysqldump, а потім стискає отриманий SQL-файл за допомогою обраного алгоритму стиснення. Це забезпечує ефективне використання простору для зберігання архівованих даних та зручність їх транспортування або зберігання на довготривалий період.

П’ятий етап алгоритму полягає у видаленні даних з основної бази після їх успішного експорту та стиснення. Цей етап є критичним, оскільки він гарантує, що тільки дані, які були правильно архівовані, будуть видалені з основної бази, а решта залишатиметься для подальшої роботи.

Перед тим, як здійснити видалення, алгоритм перевіряє, чи було успішно завершено попередні етапи: експорт даних і їх стиснення. Це важливо для уникнення ситуацій, коли дані можуть бути видалені з основної бази, але через помилки на етапі архівації не будуть збережені або стиснуті. Алгоритм перевіряє, чи існує стиснений файл, а також чи не було помилок під час експортних операцій.

Після того, як перевірки успішності виконано, алгоритм створює SQL-запит для видалення даних з основної бази. Видалення може здійснюватися як за конкретними таблицями, так і за умовами, які визначені на етапі вибору даних для архівації.

Якщо в параметрах архівації були задані умови для вибору даних (наприклад, записи, створені до певної дати або з певним статусом), то на цьому етапі алгоритм формує SQL-запит на видалення тільки тих записів, які відповідають зазначеним критеріям.

Алгоритм підключається до основної бази даних і виконує сформовані SQL-запити для видалення даних. Для цього він використовує підключення через стандартний драйвер JDBC. Під час виконання цієї операції алгоритм ретельно контролює виконання запитів, щоб уникнути випадкового видалення важливих даних.

Оскільки видалення може стосуватися для великої кількості записів, алгоритм може реалізувати операцію пакетно, по частинах, щоб уникнути перевантаження системи. Наприклад, замість того, щоб видаляти всі записи одразу, він може видаляти записи з одної партії, потім з іншої, поки весь обсяг даних не буде очищено.

Після успішного видалення алгоритм підтверджує, що всі операції були завершені без помилок. Він фіксує час і кількість видалених записів, щоб користувач мав можливість проаналізувати ефективність цього кроку. Також на цьому етапі алгоритм перевіряє цілісність бази даних, щоб переконатися, що після видалення не виникли проблеми з іншими таблицями або залежностями.

Оскільки видалення є незворотнім процесом, перед видаленням даних користувачу рекомендується створити резервну копію таблиць, що видаляються.

Шостий етап алгоритму полягає в журналюванні операцій, що забезпечує відстеження ходу виконання алгоритму, реєстрацію важливих подій, таких як успішні та неуспішні операції, а також надає можливість діагностувати помилки, якщо вони виникли під час виконання. Журналювання є важливим для збереження прозорості роботи системи, покращення її надійності та спрощення подальшого аналізу результатів архівації.

На цьому етапі алгоритм створює журнал, в якому зберігаються записи про ключові події. Кожен запис містить точний час, коли була виконана операція, тип події (наприклад, "Ініціалізація", "Експорт даних", "Стиснення"), а також результат цієї операції, що може бути або "Успіх", або "Помилка". У разі помилки також буде збережено повідомлення, що допоможе визначити її причину, наприклад, "Помилка з'єднання з базою даних" або "Помилка при стисненні файлу". Крім того, в журналі буде вказано кількість оброблених даних, таких як кількість записів, що були переміщені або видалені, а також обсяг стиснених даних.

У випадку, якщо на будь-якому етапі виникла помилка, журнал не лише зафіксує факт помилки, але й надасть додаткову інформацію, що полегшить діагностику. Наприклад, якщо не вдалося виконати експорт даних, в журналі можна побачити не тільки код помилки, але й пояснення того, чому операція не була виконана, що допоможе швидше усунути проблему. Ця інформація стане в пригоді для подальшого усунення помилок і поліпшення роботи алгоритму.

Журнал можна використовувати і для моніторингу прогресу виконання алгоритму. Він надає змогу оцінити, скільки часу витрачається на кожен етап, що допомагає виявити проблеми з продуктивністю, якщо, наприклад, якийсь із етапів триває надто довго. Такий моніторинг може бути корисним для аналізу роботи системи в реальному часі.

Після завершення алгоритму журнал можна архівувати для збереження історії виконаних операцій. Це дає змогу зберігати записи про всі виконані архівації для подальшого аналізу та забезпечення прозорості роботи системи. Старі журнали можна стиснути або зберігати у базі даних, щоб мати до них доступ в майбутньому.

Завдяки детальному журналюванню всіх операцій алгоритм не лише забезпечує прозорість виконання, але й дозволяє швидко ідентифікувати проблеми, покращувати продуктивність і гарантовано забезпечувати правильність виконання всіх етапів архівації.

Тож, описаний алгоритм складається з кількох етапів, кожен з яких виконується з урахуванням специфіки роботи з даними в реляційних базах. На першому етапі відбувається ініціалізація, коли отримуються і перевіряються всі необхідні параметри, що дає змогу правильно налаштувати з'єднання з базою даних. Наступний етап передбачає вибір даних, що відповідають заданим умовам, за допомогою SQL-запитів, а потім їх переміщення в архівне сховище.

Особливу увагу було приділено етапу стиснення даних, де для зниження обсягу інформації застосовуються популярні алгоритми стиснення, такі як gzip, zstd або lz4. Після успішного переміщення та стиснення даних з основної бази, алгоритм передбачає їх видалення для економії ресурсів.

Завершальним етапом є журналювання операцій, яке забезпечує повну прозорість виконання алгоритму, дозволяє відстежувати успішність кожного кроку та зафіксувати можливі помилки. Це створює надійний механізм для моніторингу, аналізу та підтримки алгоритму в роботі.

Таким чином, запропонований алгоритм для архівування даних забезпечує ефективне та безпечне переміщення інформації в архіви, зберігаючи високу гнучкість і масштабованість при використанні різних форматів стиснення. Кожен етап ретельно спроектований для забезпечення високої продуктивності та надійності архівації даних у реляційних базах, що робить цей підхід ефективним і для малих, і для великих обсягів інформації.

3.4 Реалізація основних компонентів модуля

У цьому розділі буде описано основні компоненти, які є частиною модуля для автоматизації процесу архівування даних у реляційних базах даних. Кожен з компонентів відіграє важливу роль у забезпеченні ефективного функціонування та гнучкості архівації.

Система архівування є основним компонентом модуля, що реалізує логіку архівації даних. Вона відповідає за весь процес від вибору даних до їх переміщення в архівне сховище та стиснення. Система працює у взаємодії з іншими компонентами модуля для забезпечення ефективності, надійності та безпеки архівації.

Спочатку система архівування отримує від користувача необхідні параметри для налаштування процесу архівації. Це включає вибір таблиць для архівації, умови для відбору даних, а також параметри стиснення. Після цього система перевіряє правильність параметрів та встановлює з’єднання з основною базою даних. На цьому етапі вона взаємодіє з компонентом взаємодії з базою даних, який відповідає за встановлення з’єднання з базою даних і перевірку її доступності. Якщо налаштування коректні, система переходить до вибору даних для архівації.

Далі система архівування формує SQL-запит, щоб вибрати дані, що відповідають критеріям архівації. Умови для цього запиту задаються користувачем (наприклад, дата створення записів або їх статус). Вибірка даних здійснюється безпосередньо з основної бази, а отримані записи зберігаються у внутрішньому буфері. Цей етап також вимагає взаємодії з компонентом взаємодії з базою даних, оскільки для формування та виконання SQL-запитів необхідно здійснити зв'язок з основною базою.

Після того як дані вибрано, система архівування підключається до архівної бази даних. Якщо в архіві ще не існують необхідні таблиці, система автоматично створює їх. Це відбувається за допомогою SQL-запитів, що створюють структуру таблиць, аналогічну тій, що є в основній базі. На цьому етапі система архівування працює разом з компонентом взаємодії з базою даних, який забезпечує створення таблиць у архівній базі. Потім система переміщає вибрані дані в архівну базу, роблячи це пакетами, щоб уникнути перевантаження пам'яті. Для забезпечення цілісності та узгодженості даних можуть використовуватися транзакції.

Після того як дані успішно переміщено в архів, система архівування приступає до їх стиснення. Стиснення здійснюється за допомогою обраного алгоритму (gzip, zstd, lz4), що було задано на початку процесу. Для цього система використовує утиліту mysqldump, яка спочатку експортує дані у формат SQL-дампу, а потім стиснутий файл зберігається в архівному сховищі. На цьому етапі система архівування взаємодіє з компонентом моніторингу, щоб повідомляти користувача про прогрес стиснення та будь-які можливі помилки, що виникають під час цього процесу.

Після успішного стиснення даних система архівування переходить до видалення архівованих записів з основної бази даних. Це дозволяє звільнити місце в основній базі, що особливо важливо при великих обсягах даних. Всі видалення виконуються після перевірки того, що дані були успішно переміщені і стиснуті. Цей етап знову активно використовує компонент взаємодії з базою даних, який відповідає за виконання операцій видалення записів із основної бази.

Завершальний етап роботи системи архівування передбачає створення звітності та журналювання всіх етапів процесу. Система забезпечує моніторинг і інформування користувача через систему моніторингу, повідомляючи про кількість оброблених записів, обсяг стиснених даних, час виконання операцій і можливі помилки, що виникли під час роботи модуля. Всі ці дані зберігаються для подальшого аналізу та звітності.

Таким чином, система архівування є ключовим компонентом, що взаємодіє з усіма іншими модулями для виконання архіваційного процесу від початку до кінця. Вона забезпечує ефективність, автоматизацію і контроль за процесом архівації, при цьому гарантує безпеку та цілісність даних.

Компонент взаємодії з базою даних є важливим елементом, який забезпечує всі операції, пов'язані з доступом до бази даних. Цей компонент відповідає за налаштування з’єднання з основною та архівною базами даних, виконання SQL-запитів для вибору, вставки та видалення даних, а також за забезпечення цілісності та узгодженості даних протягом усіх етапів архівації.

Перше, що виконує компонент це налаштування з’єднання з основною базою даних. Після отримання параметрів від системи архівування, компонент створює з’єднання з основною базою даних, використовуючи відповідні параметри підключення, такі як URL, порт, ім’я користувача та пароль. Якщо підключення до бази не вдається, компонент генерує відповідне повідомлення про помилку, щоб система могла припинити роботу, попередивши користувача про проблему. Для цього використовується драйвер MySQL, який дозволяє встановлювати з’єднання та виконувати SQL-запити.

Після успішного підключення компонент взаємодії з базою даних передає запит на вибір даних для архівації. Це виконується через форму SQL-запиту, що визначає умови вибору (наприклад, вибір записів до певної дати, з певним статусом тощо). Зібрані дані зберігаються в буфері для подальшої обробки. Важливо, щоб компонент ефективно обробляв великі обсяги даних, тому в запитах можуть бути використані обмеження на кількість вибраних записів, що дозволяє уникнути перевантаження пам'яті.

Наступним етапом є переміщення даних до архівної бази даних. Компонент взаємодії з базою даних перевіряє, чи існують відповідні таблиці в архівній базі. Якщо таблиці не існують, компонент ініціює їх створення, використовуючи SQL-запити для генерації структури таблиць, що відповідають основній базі даних. Це важливий етап, оскільки забезпечує коректне збереження

архівованих даних у необхідному форматі, підтримуючи сумісність між основною та архівною базами. У разі потреби, компонент також може забезпечити створення індексів або обмежень для підтримки цілісності даних в архівній базі.

Після того, як структура таблиць в архівній базі готова, компонент здійснює вставку вибраних даних у ці таблиці. Для цього використовуються SQL-запити, які вставляють дані пакетами, щоб уникнути перевантаження пам’яті та зменшити час обробки. Пакетна вставка дозволяє значно збільшити швидкість операцій, оскільки зменшується кількість транзакцій і загальна навантаження на систему. Окрім цього, компонент взаємодії з базою даних забезпечує використання транзакцій, щоб забезпечити цілісність даних при їх переміщенні: у разі помилки на будь-якому етапі, транзакція може бути скасована, і дані не будуть частково переміщені.

Коли дані успішно переміщено до архівної бази даних, наступним кроком є видалення записів з основної бази. Тут компонент взаємодії з базою даних відповідає за виконання операцій видалення. Для цього він генерує SQL-запити, що видаляють тільки ті записи, які були успішно переміщені до архіву, запобігаючи їх несанкціонованому видаленню. Видалення також виконується пакетами, щоб уникнути великих навантажень на систему.

Особлива увага приділяється тому, щоб забезпечити коректне та безпечне видалення даних. Компонент взаємодії з базою даних має функції, що перевіряють успішність виконання операцій, а також забезпечують обробку можливих помилок, пов’язаних з доступом до бази або виконанням SQL-запитів.

Важливим аспектом роботи цього компонента є забезпечення безпеки і захисту даних. Всі запити, які виконуються до бази даних, мають бути захищені від SQL-ін’єкцій, а також повинні мати належну авторизацію для доступу до критичних даних. Компонент взаємодії з базою даних інтегрований з компонентом безпеки, щоб забезпечити додаткові рівні захисту, зокрема за допомогою шифрування даних при їх збереженні в архіві або на етапі передачі.

У підсумку, компонент взаємодії з базою даних забезпечує ключову роль у всьому процесі архівації. Він відповідає за налаштування з’єднання, виконання SQL-запитів для вибору та переміщення даних, створення таблиць в архівній базі, а також за видалення архівованих записів з основної бази. Завдяки своїй взаємодії з іншими компонентами, такими як система архівування та система безпеки, цей компонент гарантує стабільність, ефективність і безпеку архіваційного процесу.

Інтерфейс користувача також є важливим компонентом системи архівації, який відповідає за забезпечення зручного і ефективного взаємодії користувача з програмним модулем. Його основною задачею є надання користувачеві всіх необхідних інструментів для налаштування параметрів архівації, запуску процесу, моніторингу його ходу та отримання результатів. Взаємодія з іншими компонентами системи через інтерфейс дозволяє забезпечити ефективну та зручну роботу з модулем.

Інтерфейс користувача є інтуїтивно зрозумілим і доступним, щоб користувач міг без проблем налаштувати параметри архівації без необхідності глибоких технічних знань. На початковому етапі роботи інтерфейс дозволяє користувачеві ввести всі необхідні параметри для налаштування процесу архівації. Це включає в себе вибір основної та архівної бази даних, а також конкретних таблиць для архівації. Інтерфейс надає користувачу можливість задати умови для вибору даних, такі як діапазон дат або інші фільтри, що дозволяють точніше налаштувати, які саме записи повинні бути архівовані.

Крім цього, користувач може вибрати метод стиснення даних (gzip, zstd або lz4). Інтерфейс також дозволяє налаштувати інші опції, такі як формат збереження архівів або необхідність очищення основної бази. Параметри, задані через інтерфейс, передаються до системи архівування, де вони обробляються компонентом архівації та компонентом взаємодії з базою даних для подальшої роботи.

Після налаштування всіх параметрів інтерфейс надає користувачеві можливість запустити процес архівації. Під час виконання цього процесу інтерфейс забезпечує візуальне відображення ходу роботи модуля. Він може надавати інформацію про поточний статус архівації, кількість оброблених записів, обсяг стиснених даних та час, що залишився до завершення операції. Взаємодія з системою моніторингу забезпечує користувача актуальними даними, а також повідомляє про будь-які помилки або несправності, що виникли під час роботи. Це дозволяє користувачу своєчасно вжити необхідних заходів у разі збоїв або затримок.

Інтерфейс також дозволяє користувачеві переглядати результати архівації після її завершення. Наприклад, після стиснення та переміщення даних в архів користувач може отримати звіт, що містить інформацію про успішно архівовані записи, їх кількість, загальний обсяг стиснених даних та тривалість виконання процесу. У разі виникнення помилок або неполадок під час архівації інтерфейс надає відповідні повідомлення, що дозволяють користувачу зрозуміти причину невдачі та прийняти необхідні заходи для вирішення проблеми.

Завдяки інтеграції з компонентом безпеки, інтерфейс також може надавати можливості для налаштування рівня доступу до системи, що дозволяє обмежити доступ до критичних функцій. Це включає в себе автентифікацію користувачів та управління правами доступу до різних розділів модуля. Крім того, інтерфейс може забезпечити журналювання дій користувачів для подальшого аудиту, що є важливим для забезпечення безпеки та відповідності вимогам стандартів захисту даних.

Інтерфейс користувача реалізований як веб-додаток, тож користувач може працювати із модулем з будь-якого пристрою, підключеного до інтернету, що забезпечить більшу гнучкість і мобільність роботи з модулем.

Таким чином, інтерфейс користувача є важливим елементом системи архівації, який забезпечує зручний доступ до функцій модуля, дозволяючи користувачеві налаштовувати параметри архівації, запускати процеси та

відстежувати хід виконання. Він інтегрується з усіма іншими компонентами системи, зокрема з компонентами моніторингу та безпеки, забезпечуючи зручність та ефективність використання системи архівації.

Система моніторингу є іншим важливим компонентом, який відповідає за контроль та відстеження всіх етапів архівації, а також за повідомлення користувача про стан виконання процесу. Вона забезпечує прозорість та надійність виконання алгоритму архівації, а також дозволяє вчасно виявляти й реагувати на можливі помилки чи неполадки. Завдяки цьому компоненту можна ефективно управляти архівацією великих обсягів даних і забезпечити своєчасне завершення процесу без втрати даних.

На першому етапі, система моніторингу взаємодіє з інтерфейсом користувача, забезпечуючи візуальне відображення поточного стану архівації. Під час виконання кожного етапу алгоритму, від вибору даних до стиснення й видалення записів, система моніторингу фіксує важливі події, такі як кількість оброблених записів, обсяг стиснених даних, час виконання та інші показники. Це дозволяє користувачеві бачити прогрес роботи і оцінити ефективність процесу архівації в реальному часі.

Під час роботи алгоритму система моніторингу активно відстежує стан виконання кожної операції та оперативно фіксує будь-які збої або помилки, що виникають. У разі виявлення помилки, інформація про це потрапляє від системи моніторингу на інтерфейс користувача. Це можуть бути помилки при підключенні до бази даних, проблеми з виконанням SQL-запитів, збої при стисненні архіву чи інші неполадки. Система моніторингу відразу надає детальні повідомлення про помилки, що дозволяє користувачу швидко реагувати на проблему та, за необхідності, перезапустити операцію або вжити інших заходів.

Крім того, система моніторингу може бути налаштована для збору статистичних даних про виконання архівації. Це включає в себе дані про тривалість кожного етапу процесу, обсяги даних, що були архівовані, час, що залишився до завершення, і інші показники, які допомагають оптимізувати

подальші архіваційні операції. Ці дані можуть бути збережені для подальшого аналізу або для створення звітів про виконання архіваційних процесів, що дозволяє проводити аудит та покращувати роботу системи в майбутньому.

Інтеграція системи моніторингу з системою безпеки також важлива для виявлення спроб несанкціонованого доступу чи інших підозрілих дій під час виконання архівації. Якщо виявлено порушення безпеки, система моніторингу може миттєво зупинити процес і сповістити відповідальних осіб або адміністраторів про загрозу. Це гарантує, що дані будуть захищені на всіх етапах їх переміщення та зберігання.

Однією з важливих функцій системи моніторингу є можливість налаштування сповіщень. Користувач може отримувати сповіщення не тільки в разі помилок, але й у разі досягнення певних етапів роботи, наприклад, завершення стиснення або переміщення даних в архівну базу. Сповіщення можуть надходити через інтерфейс користувача, або напряму через точки доступу REST, що забезпечує зручний спосіб інформування про прогрес архівації.

Крім того, система моніторингу може бути використана для аналізу ефективності роботи всього модуля архівації. Вона може фіксувати час початку та завершення кожного етапу процесу, що дозволяє аналізувати швидкість виконання операцій та виявляти потенційні вузькі місця в роботі системи. Ці дані можуть бути використані для подальшої оптимізації архіваційних алгоритмів або для вдосконалення інтерфейсу користувача.

В подальшому, завдяки інтеграції з компонентом безпеки, система моніторингу може забезпечити додаткові функції для виявлення аномалій в системі та підтримки безпеки даних на всіх етапах архівації. Наприклад, вона може відстежувати не тільки помилки в роботі модуля, але й спроби доступу до критичних даних без відповідних прав, а також забезпечувати протидію можливим атакам чи порушенням безпеки.

Таким чином, система моніторингу є ключовим елементом, що дозволяє підтримувати стабільність та надійність архіваційного процесу. Вона забезпечує своєчасне виявлення проблем, сповіщення користувачів про хід виконання архівації та збереження статистичних даних для подальшого аналізу. Система моніторингу також взаємодіє з іншими компонентами модуля, такими як інтерфейс користувача та система безпеки, для забезпечення прозорості, безпеки та ефективності архіваційного процесу.

Компонент безпеки є критично важливою частиною системи архівації, оскільки він забезпечує захист даних та системи від несанкціонованого доступу, зловмисних атак і забезпечує конфіденційність, цілісність та доступність інформації. Цей компонент виконує функції автентифікації користувачів, контролю доступу, шифрування даних, а також моніторингу і запису дій користувачів та системи з метою забезпечення відповідності стандартам безпеки.

Першим етапом забезпечення безпеки є автентифікація користувачів, яка дозволяє переконатися, що до системи мають доступ лише авторизовані особи. Інтерфейс користувача запитує у користувача відповідні дані для входу, після чого система перевіряє їх на відповідність збереженим записам і лише після цього надає доступ до функцій архівації.

Після автентифікації система безпеки забезпечує контроль доступу, що визначає, які саме операції користувач може виконувати в межах модуля архівації. Це дозволяє обмежити доступ до критичних функцій, таких як налаштування архівації або видалення даних з основної бази. Наприклад, адміністратори можуть мати повний доступ до всіх функцій, тоді як звичайні користувачі можуть лише переглядати звіти про архівацію без можливості змінювати параметри процесу. Цей рівень контролю допомагає запобігти випадковим або навмисним змінам у системі з боку користувачів без необхідних прав.

Для забезпечення конфіденційності даних, які архівуються, компонент безпеки може застосовувати шифрування як під час передачі даних між

компонентами системи, так і під час зберігання архівів. Це особливо важливо для захисту чутливої інформації, що може міститися в архівованих даних. Під час створення архіву за допомогою утиліти mysqldump, зібрані дані можуть бути автоматично зашифровані перед збереженням у файловій системі або в хмарному сховищі. Для цього можуть використовуватися стандартні алгоритми шифрування, такі як AES або RSA, що гарантують захист даних навіть у разі доступу до архіву сторонніми особами.

Окрім цього, компонент безпеки інтегрується з системою моніторингу, що дозволяє виявляти спроби несанкціонованого доступу, аномальні дії або зловмисні операції в процесі архівації. Наприклад, система може зафіксувати спроби доступу до архівів без відповідних прав або спроби зміни налаштувань архівації без авторизації. Такі події будуть автоматично записані в журнал безпеки, що дозволяє здійснювати аудит діяльності в системі та відслідковувати порушення політик безпеки.

Компонент безпеки також може бути налаштований для забезпечення дотримання стандартів безпеки, що є важливим для систем, які працюють з чутливою особистою інформацією. Він дозволяє здійснювати аудит дій користувачів, фіксувати час доступу, зміни в налаштуваннях і архівованих даних, що може бути важливим для звітності та юридичних вимог.

Завдяки інтеграції з іншими компонентами системи архівації, зокрема з інтерфейсом користувача та системою моніторингу, компонент безпеки активно відслідковує всі дії, що здійснюються в рамках архівації, надаючи при цьому адекватні сповіщення про будь-які загрози або порушення безпеки. Важливим аспектом є своєчасне реагування на підозрілі дії, наприклад, відключення від бази даних, спроби скинути паролі або доступ до конфіденційних архівів.

Крім того, компонент безпеки забезпечує резервне копіювання та відновлення даних, що є важливим аспектом безпеки в разі втрати даних або пошкодження архівів. Всі архіви повинні бути доступними для відновлення в

разі необхідності, і цей процес має бути захищений від несанкціонованого доступу.

Таким чином, компонент безпеки є критично важливим для забезпечення захищеності, конфіденційності та цілісності даних під час архівації. Він гарантує, що лише авторизовані користувачі можуть отримувати доступ до системи, контролює доступ до важливих функцій, захищає дані за допомогою шифрування і моніторить всі дії з метою виявлення та запобігання загрозам. У комплексі з іншими компонентами система безпеки забезпечує цілісність та надійність роботи програмного модуля архівації.

Підсумовуючи, у цьому підрозділі було розглянуто ключові складові програмного модуля архівації, які забезпечують його ефективну та безпечну роботу. Основними компонентами є система архівування, компонент взаємодії з базою даних, інтерфейс користувача, система моніторингу та компонент безпеки. Кожен із цих елементів виконує важливу роль у забезпеченні безперебійного процесу архівації даних, їх безпеки, доступності та моніторингу.

Система архівування відповідає за безпосереднє виконання процесу архівації, включаючи вибір даних, їх експортування та стиснення. Компонент взаємодії з базою даних забезпечує стабільне підключення до джерела даних і архівної бази, а також управління виконанням SQL-запитів. Інтерфейс користувача надає можливість зручної взаємодії з системою, дозволяючи користувачам налаштовувати процес архівації та відстежувати його хід.

Система моніторингу відповідає за постійне відслідковування стану архівації та повідомлення користувача про важливі події та помилки в процесі виконання операцій. Взаємодія з компонентом безпеки забезпечує захист архівованих даних від несанкціонованого доступу, шифрування та аудиторський контроль, що підвищує надійність і відповідність стандартам безпеки.

Таким чином, реалізація цих компонентів є важливою умовою для ефективного виконання архівації даних, забезпечення її безпеки, стабільності та зручності користування системою. Інтеграція кожного компонента в загальний

процес дозволяє досягти високого рівня автоматизації архівації, знижуючи ймовірність помилок та забезпечуючи надійний захист даних.

3.5 Забезпечення безпеки та цілісності даних у процесі архівування

Цей підрозділ зосереджений на важливих аспектах забезпечення захисту і збереження даних у процесі їх архівації. Оскільки архівація є важливою складовою частиною системи збереження та управління даними, забезпечення їх безпеки та цілісності є критично важливим для запобігання втратам даних, несанкціонованому доступу та інших загроз. У цьому підрозділі описуються основні підходи та методи, що використовуються для забезпечення безпеки архівованих даних.

Перше і найважливіше завдання полягає в захисті даних під час їх передачі. Це є критично важливим аспектом забезпечення безпеки архівованих даних, оскільки під час передавання через мережу існує ризик перехоплення або модифікації інформації сторонніми особами. Для запобігання таким загрозам використовується сучасний криптографічний протокол TLS (Transport Layer Security). Цей протокол забезпечує шифрування всіх даних, які передаються між клієнтом і сервером, тим самим захищаючи їх від несанкціонованого доступу під час трансферу.

При використанні TLS вся передана інформація шифрується з використанням асиметричних і симетричних криптографічних алгоритмів. Це означає, що навіть якщо зловмисник перехопить мережевий трафік, він не зможе прочитати або змінити вміст повідомлень, оскільки для дешифрування даних необхідний спеціальний ключ, який є лише у відправника та одержувача. Протокол TLS також забезпечує автентифікацію сторін, гарантуючи, що обидві сторони (клієнт і сервер) можуть бути впевненими у тому, що вони взаємодіють саме з тими партнерами, з якими планували [16].

Крім того, реалізовано перевірку сертифікатів для TLS з'єднань, щоб підтвердити автентичність сервера і захистити систему від атаки "чоловіка посередині" (Man-in-the-Middle). Це забезпечує впевненість у тому, що дані передаються між авторизованими сторонами і немає можливості їх фальсифікації або втручання сторонніх осіб.

Захист даних під час передачі є одним з найважливіших етапів безпеки архівації, особливо коли дані переміщуються через незахищені канали зв'язку або зберігаються в хмарних сховищах. Використання сучасних криптографічних стандартів та протоколів гарантує, що дані залишаються захищеними навіть під час їх передавання через потенційно небезпечні мережі.

Ще одним важливим аспектом є забезпечення цілісності даних. Забезпечення цілісності даних у процесі архівування є одним з ключових аспектів для гарантування того, що архівовані файли зберігають свою точність і не були змінені або пошкоджені під час транспортування, зберігання або обробки. Порушення цілісності даних може виникнути внаслідок збоїв у системах, помилок користувача або зовнішніх загроз, таких як несанкціоновані зміни або атаки. Для цього під час архівування використовуються методи хешування та контрольних сум, що дозволяє гарантувати відсутність змін у даних під час їх архівації та зберігання. Для цього кожен архівований файл або набір даних проходить через процес хешування, де обчислюється контрольна сума з використанням алгоритму SHA-256. Цей алгоритм обирається через свою високу стійкість до колізій і надійність у забезпеченні цілісності даних.

Під час архівації модуль генерує контрольну суму для кожного архівованого файлу, яку зберігає у базі даних разом з іншою метаінформацією, такою як розмір файлу та дату створення. Після завершення процесу архівування та збереження даних, контрольні суми зберігаються у спеціальній таблиці архівної бази даних, що дозволяє в подальшому здійснювати перевірку цілісності даних під час відновлення або під час їх передачі.

Коли архівовані дані потрібно відновити або передати, система обчислює нову контрольну суму для кожного файлу на основі поточного вмісту і порівнює її з оригінальною контрольною сумою, збереженою в архівній базі. Якщо контрольні суми збігаються, це означає, що дані не були змінені або пошкоджені. Якщо ж значення контрольних сум не співпадають, модуль автоматично повідомляє про проблему, що дозволяє оперативно вжити заходів для відновлення даних або їх перевірки.

Для забезпечення додаткової безпеки та перевірки цілісності даних, також використано цифрові підписи. Кожен архів підписується приватним ключем, що дозволяє користувачам перевіряти походження архівів і переконатися в тому, що вони не були змінені з моменту створення. Підпис застосовується до кожного архіву перед його збереженням, що додає ще один рівень захисту.

Цей підхід забезпечує високу надійність і цілісність даних протягом всього процесу архівування, зберігання та передачі, гарантуючи, що архівовані файли залишаються незмінними і можуть бути безпечно відновлені в будь-який момент.

Для архівування даних передбачено шифрування архівованих файлів як важливий етап забезпечення їх безпеки. Шифрування дозволяє захистити архіви від несанкціонованого доступу та гарантує, що тільки авторизовані користувачі з відповідними правами можуть отримати доступ до вмісту архівованих файлів.

Для цього використовується стандарт AES-256 (Advanced Encryption Standard з 256-бітним ключем), який є одним з найнадійніших алгоритмів симетричного шифрування. AES-256 широко використовується в промисловості для захисту даних і має високий рівень безпеки завдяки великому розміру ключа, що ускладнює його зламування навіть за допомогою сучасних методів атак.

Процес шифрування архівів починається після того, як дані були успішно експортовані з бази даних за допомогою утиліти mysqldump і після того, як дані були стиснуті за допомогою обраного алгоритму (наприклад, gzip, zstd або lz4). Перед збереженням архівовані файли шифруються, і для цього використовується

секретний ключ, який генерується або вводиться користувачем під час налаштування процесу архівації. Цей ключ необхідно зберігати в безпечному місці, оскільки він використовується для дешифрування файлів під час їх відновлення. Система забезпечує, щоб цей ключ не зберігався разом з архівованими файлами, що підвищує рівень безпеки.

Шифрування виконується за допомогою бібліотеки, що підтримує AES-256, наприклад OpenSSL або libsodium. Процес шифрування файлів проходить наступним чином: кожен архівований файл або пакет даних обробляється через алгоритм AES-256, де дані шифруються в потоковому або блочному режимі, залежно від розміру файлу та налаштувань системи. Шифрований файл зберігається в архівному сховищі з відповідним розширенням, що вказує на його зашифрований статус (наприклад, .enc або .aes).

Під час відновлення архівованих даних, система використовує той самий ключ для дешифрування файлів. Для цього виконується зворотний процес: зашифрований файл зчитується і передається в алгоритм AES-256, який відновлює початкові дані. Якщо ключ введений неправильно або відсутній, дешифрування не відбудеться, і користувач отримає повідомлення про помилку. Таким чином, шифрування гарантує, що дані можуть бути відновлені тільки в тому випадку, якщо користувач має відповідний ключ доступу.

В подальшому, для посилення безпеки архівів під час їх зберігання в мережевих чи хмарних сховищах, шифрування може виконуватися в поєднанні з іншими технологіями захисту, такими як TLS або SSL для забезпечення безпечного каналу передачі даних. Таким чином, файли не тільки шифруються на момент архівації, але й передаються по зашифрованому каналу, що забезпечує комплексний рівень захисту під час усього процесу.

Загалом, шифрування архівованих файлів в розробленому модулі є критично важливим для захисту чутливих даних і гарантує їх конфіденційність і безпеку на всіх етапах архівації, зберігання та передачі.

Не менш важливим є забезпечення автентифікації користувачів є автентифікація користувачів, оскільки вона гарантує, що тільки авторизовані особи можуть здійснювати операції з архівуванням, доступом до архівів і їх відновленням. Це особливо критично в контексті обробки чутливих даних, коли неправильний доступ до архівів може призвести до втрати або витоку важливої інформації.

Для забезпечення автентифікації використовано стандартний механізм входу за паролем. Під час входу в систему користувач вводить свій логін та пароль, які перевіряються на відповідність збереженій інформації в базі даних. Всі паролі зберігаються в зашифрованому вигляді, що виключає їх прямий доступ навіть для адміністраторів системи.

Крім того, сюди може бути легко інтегровано LDAP (Lightweight Directory Access Protocol) або OAuth для аутентифікації користувачів у корпоративних середовищах, де зберігається централізована інформація про користувачів. Це дозволить адміністраторам більш ефективно керувати доступом, забезпечуючи єдину точку контролю і централізоване управління правами доступу.

Аутентифікація користувачів також включає в себе підтримку різних рівнів доступу до функцій модуля. Наприклад, адміністратори мають повний доступ до всіх операцій, таких як архівування, відновлення, налаштування шифрування, тоді як звичайні користувачі можуть мати обмежений доступ тільки до перегляду архівів чи їх відновлення. Ці рівні доступу визначаються через рольову модель доступу, де кожен користувач отримує певну роль (наприклад, «адміністратор», «користувач» або «гость»), що визначає, які операції він може виконувати.

Також важливою частиною автентифікації є логування спроб доступу до системи. Кожен вхід у систему, зміна пароля або спроба доступу до архіву фіксуються в журналах безпеки, що дозволяє адміністраторам системи вчасно виявляти підозрілі дії або спроби несанкціонованого доступу.

Також було застосовуано механізм автоматичного блокування після кількох неуспішних спроб введення пароля. Це зменшує ризики атак на систему, таких як атаки методом підбору пароля.

Для підвищення безпеки автентифікації користувачів також використано шифрування паролів в базі даних. Замість того, щоб зберігати паролі у відкритому вигляді, дані шифруються компонентом безпеки, що робить паролі стійкими до атак методом грубої сили або словникових атак.

Забезпечення надійної автентифікації користувачів є одним із ключових елементів безпеки в системі архівування. Цей підхід гарантує, що лише уповноважені особи можуть взаємодіяти з архівами, і дозволяє захистити важливі дані від несанкціонованого доступу та змін.

Ще однією важливою складовою безпеки є моніторинг та аудит операцій, які здійснюються з архівованими даними. Для забезпечення повного та ефективного контролю за процесом архівації необхідно постійно відслідковувати кожну дію, що виконується з архівами — від доступу до них до будь-яких змін чи видалення. Програмний модуль реалізує систему моніторингу, яка фіксує всі операції, що стосуються архівованих даних. Це дозволяє вчасно виявити будь-які підозрілі дії, що можуть свідчити про несанкціоновані спроби доступу або порушення політики безпеки.

Кожна подія, що відбувається в системі, будь то спроби входу, відкриття архівів, зміни в їхньому вмісті або навіть спроби їх видалення, автоматично записується в спеціальний журнал. Журнали містять детальну інформацію про те, хто, коли і яку операцію виконав, а також деталі про результат кожної спроби. Це дає змогу відстежувати повний ланцюг подій, що стосуються архівів, і відновлювати ситуацію навіть у разі виникнення проблем чи інцидентів безпеки.

Ці журнали виконують ключову роль у забезпеченні прозорості всіх операцій з архівованими даними. Вони є важливим інструментом для аналізу дій користувачів та виявлення можливих порушень політики доступу чи спроб зловживань. Завдяки такому підходу адміністраторами системи можна

ефективно перевіряти та перевіряти наявність будь-яких порушень, а також швидко реагувати на непередбачувані ситуації, мінімізуючи ризики безпеки. Усі ці заходи значно підвищують рівень захисту архівованих даних, роблячи їх обробку та зберігання більш надійними та контрольованими.

Для забезпечення довготривалої цілісності архівованих даних також важливо організувати надійне резервне копіювання архівних файлів. Хоча в розробленому модулі реалізовано всі необхідні механізми для збереження архівів у захищеному вигляді, резервне копіювання даних є важливою додатковою процедурою, яка в цьому випадку покладена на користувача. Ця процедура дозволяє захистити архіви від можливих фізичних пошкоджень носіїв, випадкових видалень або інших непередбачених ситуацій, що можуть призвести до втрати важливих даних.

Користувач повинен самостійно налаштувати регулярне копіювання архівів на додаткові носії або в інші безпечні місця зберігання, такі як інші фізичні диски чи хмарні сховища. Це дозволяє мати резервні копії архівів, які можна буде відновити в разі потреби. Регулярність і надійність таких копій повинні визначатися виходячи з важливості даних і вимог до збереження їх цілісності протягом тривалого часу.

3.6 Висновки

В цьому розділі було розглянуто основні принципи та заходи, які гарантують безпеку і цілісність даних під час їх архівування. Забезпечення захисту архівованих даних є ключовим аспектом для збереження конфіденційності, доступності та цілісності інформації на всіх етапах обробки. Важливими заходами для досягнення цієї мети є шифрування даних під час їх

передачі та зберігання, що дозволяє захистити архіви від можливих атак або несанкціонованого доступу.

Окрім цього, на етапі доступу до архівів застосовується система автентифікації користувачів, що забезпечує тільки авторизованим особам можливість здійснювати операції з даними. Це дозволяє ефективно контролювати доступ і запобігати порушенням політики безпеки. Не менш важливими є моніторинг та аудит операцій, що дають змогу відстежувати всі дії з архівами і своєчасно виявляти будь-які несанкціоновані спроби доступу або зміни в даних.

Додатково, було акцентовано на необхідності організації резервного копіювання архівних файлів для забезпечення їх довготривалої цілісності. Хоча сам програмний модуль не включає функцію резервного копіювання, це є важливою складовою загальної стратегії безпеки, яку користувач повинен організувати окремо. Регулярне створення резервних копій дає змогу захистити дані від випадкових пошкоджень або втрат.

В сукупності, ці заходи гарантують, що архівовані дані зберігаються в безпечному і цілісному вигляді. Вони дозволяють не тільки захистити дані від зовнішніх загроз, але й забезпечують високий рівень контрольованості та прозорості операцій з архівами. Таким чином, даний розділ охоплює всі аспекти, необхідні для забезпечення високої безпеки і цілісності даних у процесі архівування.

4 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ТА ОЦІНКА ЕФЕКТИВНОСТІ

У цьому розділі буде розглянуто експериментальну перевірку та оцінку ефективності розробленого програмного модуля для архівування даних, зокрема, перевірку його здатності до виконання запитів у реальних умовах. Метою експерименту є не тільки тестування функціональності системи, але й аналіз її ефективності, продуктивності та масштабованості при роботі з великими обсягами даних. Оскільки система повинна забезпечувати автоматизацію процесу архівування для бази даних MySQL, особливу увагу буде приділено тестуванню алгоритмів вибору, переміщення та стиснення даних, а також перевірці взаємодії з іншими компонентами системи.

Тестування проводитиметься на реальних даних, що будуть заповнені випадковими значеннями, для того щоб наблизити умови до реального середовища та відтворити ситуації, з якими система може зіткнутися у процесі архівування великих баз даних. Кожна база даних, з якою працюватиме модуль, буде наповнена випадковими записами, що дозволить перевірити, як система справляється з різноманітними структурами даних і як вона веде себе при обробці записів з різними значеннями.

Запити на початок оброблення будуть виконуватись через REST-ендпоінти, що дозволить симулювати реальні сценарії взаємодії з системою в умовах інтеграції з іншими програмними продуктами чи платформами. Завдяки такій архітектурі, тестування буде зручним і гнучким, дозволяючи зберігати дані в різних форматах та взаємодіяти з ними в стандартних режимах роботи.

У підрозділі буде розглянуто план-програму експерименту, яка включатиме кілька тестових сценаріїв, що дозволять оцінити ключові аспекти роботи модуля: від швидкості архівування та ефективності алгоритмів до надійності роботи під час великого навантаження. Окремо будуть оцінюватися

продуктивність системи при обробці великих обсягів даних та її здатність до масштабування в умовах зростаючого обсягу інформації.

Додатково буде проведено порівняння з іншими підходами до архівування даних, що дозволить зрозуміти переваги і недоліки розробленого рішення в контексті існуючих методів. Це дозволить більш об'єктивно оцінити його ефективність і дати точну характеристику його конкурентоспроможності на ринку подібних інструментів.

4.1 Постановка план-програми експерименту

План-програма експерименту була розроблена з метою ретельної перевірки всіх функцій і характеристик розробленого програмного модуля для архівування даних з бази даних MySQL, що надало змогу оцінити його працездатність, ефективність і безпеку. Основним завданням експерименту є вивчення її поведінки в реальних умовах з різними обсягами даних та умовами архівації. Для цього було розроблено чітке планування етапів тестування, яке включало кілька ключових компонентів: перевірку коректності основних функцій, вимірювання продуктивності, тестування безпеки даних та аналіз масштабованості системи.

Експеримент має складатись з кількох основних етапів. Першим завданням є перевірка правильності архівування даних: чи відбираються правильні записи для архівації, чи коректно виконується їх переміщення в архівну базу та чи забезпечується необхідне стиснення архівів за допомогою алгоритмів gzip, zstd чи lz4, вибраних на етапі налаштування. Для цього мають використовуватись бази даних, що наповнені випадковими значеннями для імітації реальних умов роботи. Бази мають бути заповнені великими обсягами даних, включаючи різноманітні таблиці з різними типами даних, щоб перевірити, як система буде працювати з різноманітними структурами інформації.

Далі необхідно виміряти ефективність обробки даних, зокрема швидкість архівування та стиснення даних, а також час, необхідний для вибірки великих обсягів даних. Для цього має бути проведена серія тестів, на яких можна перевірити швидкість виконання всіх основних операцій: вибірки даних, стиснення та перенесення в архів. Зокрема, такий експеримент має полягати у вимірюванні часу, необхідного для виконання кожної операції при різних обсягах даних, щоб оцінити як система справляється з навантаженням.

Особливу увагу такж приділено тестуванню алгоритмів стиснення. Для цього має бути проведено серію експериментів із застосуванням різних алгоритмів стиснення (gzip, zstd і lz4), що дозволить порівняти їх ефективність і вибрати оптимальний варіант для різних типів даних. У ході тестів має вимірюватись швидкість стиснення, а також розмір архівованих файлів для кожного з алгоритмів. Це дозволить оцінити, наскільки швидко здійснюється стиснення даних і який обсяг пам'яті займають результати після стиснення для кожного з варіантів.

Крім того, одним із важливих завдань також виступає тестування системи безпеки, а саме перевірка захисту даних під час їх архівування та передачі. Оскільки архівовані дані можуть містити чутливу інформацію, тестування має включати перевірку методів шифрування та захисту архівів під час передачі через мережу. Важливими аспектами тестування безпеки є використання криптографічних методів для захисту файлів на етапах архівування і зберігання, а також моніторинг усіх дій з даними в системі, щоб уникнути можливих спроб несанкціонованого доступу до них.

Масштабованість системи також є одним з основних завдань експерименту. В ході тестування має бути проведено оцінювання того, як система обробляє запити при великому навантаженні та в умовах різних обсягів даних. Для цього необхідно створити великі бази даних з випадковими значеннями, на основі яких буде виконано запити через REST-ендпоінти для симуляції реального навантаження. Важливо перевірити, чи зберігається

продуктивність системи при збільшенні обсягів даних і чи зменшується швидкість обробки запитів під великим навантаженням.

Таким чином, експеримент має бути спрямований на всебічну перевірку розробленого модуля, охоплюючи всі технічні та безпекові аспекти його роботи. Важливим є не тільки перевірка основних функцій і алгоритмів, таких як вибірка даних для архівації, стиснення та переміщення даних, але й забезпечення захисту та цілісності даних на всіх етапах процесу архівування. Під час тестування має бути акцентовано увагу на вимірюванні продуктивності системи за різних умов роботи, зокрема при великих обсягах даних, а також на перевірці ефективності застосованих методів стиснення. Крім того, особлива увага має бути приділена тестуванню безпеки, що включає захист даних від несанкціонованого доступу та перевірку процесів шифрування і автентифікації.

Усі результати тестів мають бути детально зафіксовані та проаналізовані для подальшої оптимізації розробленого модуля. Для цього після кожного етапу експерименту має здійснюватись ретельний аналіз результатів, що дозволить виявити можливі слабкі місця та покращити як технічні, так і безпекові характеристики системи. Оцінка працездатності та ефективності модуля має проводитись не лише через вимірювання часу виконання операцій, але й через порівняння з іншими підходами до архівування даних. Завдяки цьому експеримент дозволить отримати об'єктивні дані для оцінки, чи відповідає розроблений модуль вимогам, що ставляться до нього, в умовах реального навантаження.

4.2 Тестування та верифікація розробленого модуля

Після завершення розробки програмного модуля важливим етапом стало його тестування та верифікація для перевірки коректності роботи всіх його компонентів згідно попередньо створеній план-програмі. Основна мета цього

етапу полягала у впевненості, що модуль функціонує відповідно до визначених вимог і може обробляти архівацію даних у заданих умовах. Тестування охоплювало як функціональні, так і нефункціональні аспекти, що дозволило оцінити не лише працездатність, а й ефективність, стабільність і надійність системи.

Для виконання тестування було організовано набір тестових сценаріїв, які охоплювали ключові аспекти роботи модуля. Серед них перевірка коректності виконання операцій архівації, переміщення даних між базами, обробка великих обсягів інформації, а також реакція модуля на можливі збої та нестандартні ситуації. Окремо було протестовано систему моніторингу та захисту даних, зокрема шифрування та контроль доступу.

Нижче наведено декілька основних тестів, що було проведено.

Приклад 1: запит на архівацію даних із зазначенням таблиці та умов.

Для перевірки працездатності модуля було створено тестову базу даних source_database з таблицею orders, яка містить 100,000 записів. Ці дані були згенеровані випадковим чином і включають такі поля:

– id (унікальний ідентифікатор запису);

– customer_id (ідентифікатор клієнта);

– created_at (дата створення замовлення);

– amount (сума замовлення).

Записи мали значення created_at у межах від 1 січня 2020 року до 31 грудня 2024 року. З них 10,000 записів відповідають умові архівації: created_at < '2025-01-01'.

Цільовою базою даних була archive_database, яка спочатку була порожньою. Завдання полягало в архівації даних із таблиці orders, що відповідають зазначеній умові, та переміщенні їх у базу archive_database.

Для запуску тесту було виконано перший REST-запит, який можна побачити на рисунку 4.1.

Після отримання запиту модуль ініціює підключення до бази даних source_database та перевіряє наявність таблиці orders. Система формує SQL-запит для вибору даних, що відповідають умові created_at < '2025-01-01'. У результаті було вибрано 10,000 записів.

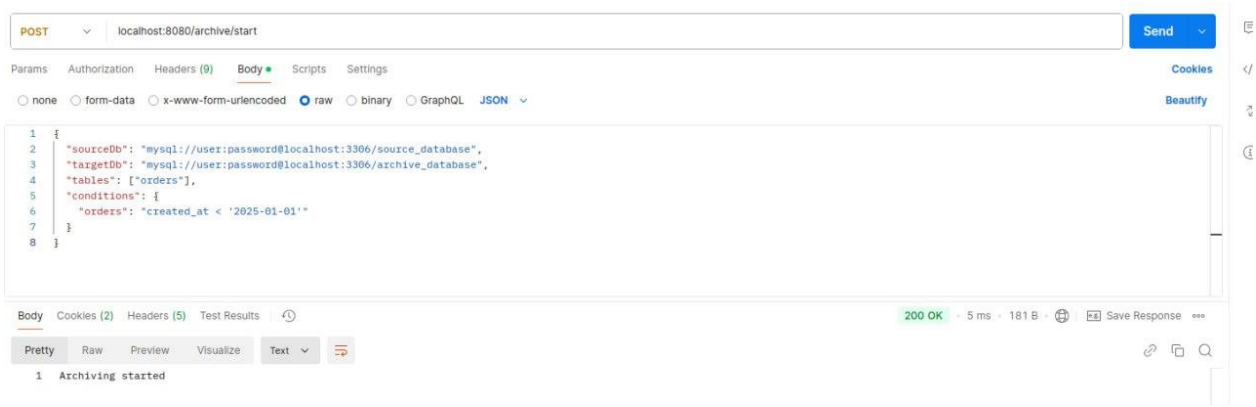

Рисунок 4.1 – Запит на початок архівування

Дані передаються у вигляді пакетів по 1,000 записів для мінімізації використання оперативної пам’яті. Кожен пакет записів обробляється, і відповідні записи вставляються у таблицю orders у базі archive_database. Якщо таблиці orders в архівній базі не існувало, модуль створює її з відповідною структурою.

Після успішного переміщення всіх записів модуль виконує перевірку, порівнюючи кількість переміщених записів із початковою вибіркою, щоб переконатися в коректності операції. У кінці процесу записи, які були переміщені, видаляються з таблиці orders у базі source_database.

Очікуваний результат експерименту:

– таблиця orders у базі source_database зменшується на 10,000 записів, залишаючи 90,000 записів;

– у таблиці orders бази archive_database з’являється 10,000 записів із тим самим вмістом;

– система фіксує у лог-файлі такі дані: кількість оброблених записів, час виконання, розмір переміщених даних, можливі помилки (якщо виникли).

Фактичний результат: запит виконався успішно. Після завершення операції таблиця orders у source_database містила 90,000 записів, а в archive_database 10,000 записів. Лог-файл зафіксував наступну інформацію:

Task ID: task_12345

Start Time: 2025-01-15T14:00:00Z

End Time: 2025-01-15T14:01:30Z

Processed Records: 10,000

Data Size: 25 MB

Errors: None

Час виконання операції склав 1 хвилину 30 секунд, а обсяг переданих даних 25 МБ. Усі результати відповідають очікуваним, що підтверджує працездатність модуля у даному сценарії.

Приклад 2: запит на перевірку статусу архівації.

Після виконання архівації користувач може перевірити статус виконання процесу за допомогою REST-запиту до відповідного ендпоінта модуля. Для тесту було створено таблицю transactions із 10,000 записів, з яких 8,000 відповідали умовам архівації (date < '2024-01-01'). Архівація була успішно виконана, і модуль перемістив відповідні записи до архівної бази. Завданням було перевірити, чи модуль правильно відображає статус цього процесу.

Таблиця transactions мала такі поля:

– id (унікальний ідентифікатор);

– amount (сума транзакції);

– date (дата транзакції);

– status (статус транзакції).

Дані були створені із датами у межах від 1 січня 2020 року до 31 грудня 2024 року.

Запит, що було надіслано до ендпоінта /archive/status, зображено на рисунку 4.2. Там же можна побачити тіло запиту, а також відповідь сервера.

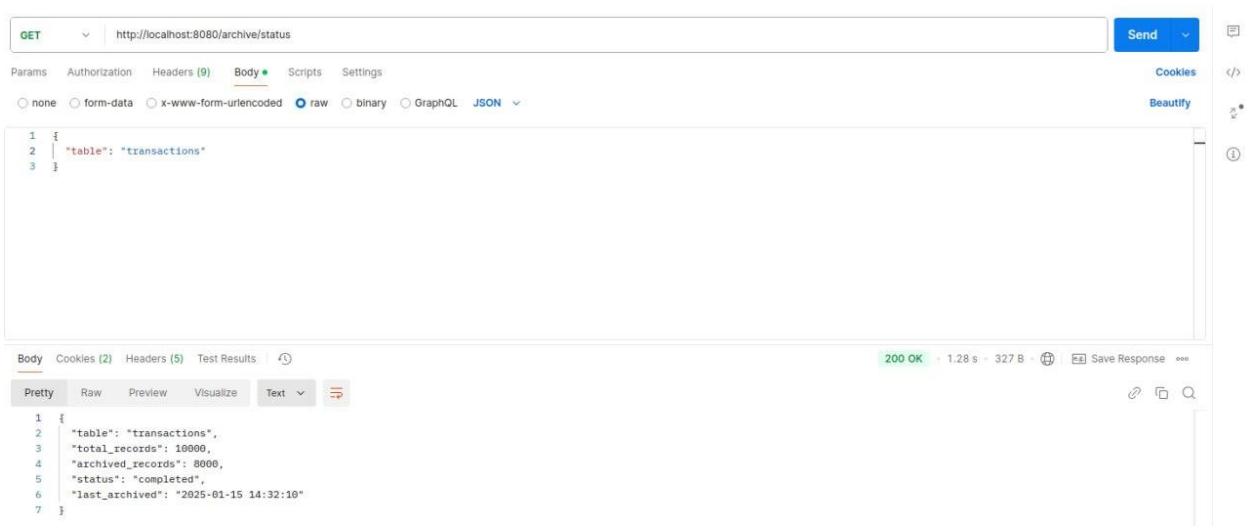

Рисунок 4.2 – Запит на отримання статусу архівації

Як видно, модуль обробив запит, звернувшись до журналу виконаних операцій, і повернув JSON-відповідь із деталями статусу архівації. Відповідь модуля підтверджує, що із таблиці transactions було оброблено всі 10,000 записів, із яких 8,000 були заархівовані. Поле status демонструє, що архівація завершена, а поле last_archived вказує точну дату й час останньої операції.

Тест успішно довів, що модуль коректно обробляє запити на перевірку статусу архівації, надаючи точну інформацію про кількість записів, дату останньої операції й загальний статус процесу.

Приклад 3: тест на некоректний запит.

Для перевірки обробки некоректних запитів модуль був протестований на сценарії, коли користувач надсилає запит із неправильними параметрами. Завданням було перевірити, чи модуль коректно виявляє помилки, надає зрозуміле повідомлення про них і не виконує жодних дій, які можуть порушити роботу системи.

Користувач надіслав запит до ендпоінта /archive/start без обов’язкового параметра table. У цьому випадку очікувалося, що модуль згенерує повідомлення про помилку, вказавши на відсутність необхідного параметра.

Запит до модуля, його тіло та відповідь серверу зображено на рисунку 4.3.

Рисунок 4.3 – Некоректний запит на старт архівації

Відповідь модуля підтверджує, що помилка була коректно оброблена. Код статусу 400 Bad Request чітко вказує на помилку з боку клієнта, а детальне повідомлення дозволяє користувачеві зрозуміти причину. При цьому жодних дій із базою даних не виконувалося, що гарантує стабільність системи.

Також для повного охоплення було перевірено аналогічні сценарії при:

– відсутності параметра conditions;

– некоректному форматі URL для параметра sourceDb;

– некоректному форматі URL для параметра targetDb;

– передачі неіснуючої таблиці.

У кожному випадку модуль коректно ідентифікував помилку та повертав відповідні повідомлення.

Даний тест продемонстрував, що модуль стійкий до некоректних запитів, надає користувачеві чітку інформацію про виявлені помилки й не виконує небажаних дій. Це підтверджує відповідність модуля вимогам до надійності й безпеки.

Проведені тести охопили ключові аспекти роботи системи, включаючи обробку коректних запитів, перевірку статусу виконання архівації та стійкість до некоректних запитів. Для кожного тесту було наведено конкретні приклади REST-запитів, що дозволяють детально оцінити поведінку модуля в різних сценаріях.

Тестування показало, що модуль здатний успішно виконувати завдання архівації навіть при значному обсязі даних, забезпечуючи коректне переміщення та обробку записів відповідно до заданих умов. Запити на перевірку статусу виконуються оперативно й інформативно, дозволяючи користувачеві відстежувати хід процесу архівації.

Окрема увага була приділена обробці некоректних запитів. Модуль продемонстрував здатність виявляти помилки, наприклад, відсутність обов’язкових параметрів чи некоректний формат введення, і коректно повідомляти про це користувача. Жодних небажаних дій у разі помилкових запитів не виконувалося, що свідчить про високу надійність і захищеність системи.

Результати тестів підтвердили, що розроблений модуль відповідає заявленим вимогам до функціональності, стабільності та безпеки. Це дає підстави вважати, що система готова до практичного використання у реальних умовах, забезпечуючи ефективне і безпечне архівування даних.

4.3 Оцінка продуктивності та масштабованості

Після завершення етапів тестування та верифікації наступним важливим кроком є оцінка продуктивності та масштабованості розробленого модуля. Цей підрозділ спрямований на аналіз, як система реагує на навантаження різної інтенсивності та обсягу даних, а також на визначення меж її ефективності в умовах масштабування.

Для проведення цього дослідження було сформовано низку експериментальних сценаріїв, кожен з яких передбачав виконання операцій архівації за різних умов. Тестування продуктивності здійснювалося шляхом вимірювання часу виконання ключових операцій, таких як вибірка даних, їх переміщення в архівну базу, а також стиснення архівів. Масштабованість

оцінювалася на основі обробки даних у базах різного розміру, починаючи від невеликих (кілька тисяч записів) до великих, що містять мільйони рядків.

Під час експериментів база даних наповнювалася тестовими даними, які включали випадкові записи із заданою структурою. Для кожного тесту обсяг даних зростав поступово, що дозволило визначити залежність продуктивності від кількості оброблюваних записів. Наприклад, для оцінки роботи системи з великими масивами даних проводилися операції архівації таблиць, які містили від 100 тисяч до 10 мільйонів рядків.

Проведений аналіз дозволив детально оцінити час виконання кожної операції модуля. Для тестування використовувалися таблиці, що включали кілька типів полів: числові (INTEGER, DECIMAL), текстові (VARCHAR, TEXT), а також дата-часові (DATETIME, TIMESTAMP). Структура таблиць була спеціально розроблена для моделювання реальних сценаріїв використання, де текстові поля займали приблизно 60% обсягу даних, числові – 30%, а решта припадала на дата-часові значення.

Вибірка даних із таблиці, що містила 1 мільйон записів, займала в середньому 10 секунд. При цьому розмір таблиці становив близько 250 МБ. У випадку таблиці з 10 мільйонами записів, загальний обсяг якої досягав 2.5 ГБ, час вибірки зростав до 90 секунд. Підвищення часу виконання було очікуваним і відповідало збільшенню обсягу даних.

Процес стиснення архівів був протестований всіх для трьох доступних алгоритмів: gzip, zstd та lz4. Таблиця з 1 мільйоном записів після експорту у формат SQL мала розмір близько 250 МБ. Для її стиснення gzip витрачав близько 16 секунд, зменшуючи обсяг файлу до 50 МБ (80% зменшення). Алгоритм zstd показав схожий рівень стиснення (розмір стисненого файлу – 48 МБ), але час обробки був меншим – 12 секунд. Алгоритм lz4 працював найшвидше, виконуючи стиснення за 6 секунд, проте рівень стиснення був нижчим – файл мав розмір 60 МБ (76% зменшення) [17][18][19].

При тестуванні таблиці з 10 мільйонами записів gzip витрачав приблизно 140 секунд, досягаючи зменшення розміру з 2 ГБ до 500 МБ. Алгоритм zstd забезпечував схожий результат за 110 секунд, тоді як lz4 справлявся за 60 секунд, зменшуючи файл до 600 МБ.

Ці результати підтвердили ефективність використання різних алгоритмів стиснення в залежності від пріоритетів системи: gzip та zstd підходять для сценаріїв, де важлива максимальна економія місця, тоді як lz4 є оптимальним вибором для швидкісної обробки.

Окрім цього, було проведено тести в умовах підвищеного навантаження, коли кілька операцій архівації запускалися паралельно. Це дало можливість оцінити, наскільки система здатна ефективно обробляти запити у багатозадачному середовищі та як впливає навантаження на стабільність роботи. Виявилося, що навіть за умов одночасної архівації кількох великих таблиць час обробки збільшувався незначно, завдяки оптимізації роботи з базою даних і паралельному виконанню операцій.

Оцінка масштабованості також включала перевірку роботи системи при зміні конфігурації серверного обладнання. На основі результатів було підтверджено, що модуль може ефективно використовувати додаткові ресурси, такі як багатоядерні процесори та високошвидкісні дискові системи, для забезпечення більшої продуктивності.

Таким чином, проведені експерименти з оцінки продуктивності та масштабованості підтвердили здатність системи ефективно обробляти великі обсяги даних і працювати у багатозадачному середовищі. Це свідчить про її готовність до використання у реальних умовах із динамічним збільшенням навантаження.

4.4 Порівняння з іншими підходами до архівування

Розроблений модуль для архівування даних у базі даних MySQL має ряд схожих аспектів із комерційними і відкритими інструментами для резервного копіювання, такими як MySQL Enterprise Backup, Percona XtraBackup чи ClusterControl. Порівняння цих підходів дозволяє оцінити переваги і можливі недоліки розробленого рішення в контексті існуючих альтернатив.

MySQL Enterprise Backup є комерційним продуктом від Oracle, що надає інструменти для резервного копіювання та відновлення MySQL-баз даних. Цей продукт підтримує гарячі резервні копії для InnoDB, а також пропонує інкрементальні резервні копії, шифрування і стиснення даних. Однією з основних переваг MySQL Enterprise Backup є його інтеграція з іншими продуктами Oracle, що дозволяє забезпечити високу надійність і підтримку для корпоративних середовищ. Однак важливим обмеженням є ліцензійна модель, оскільки для використання цього інструменту необхідно придбати платну ліцензію на MySQL Enterprise Edition, що робить його менш доступним для малих та середніх підприємств [20].

Порівнюючи його з розробленим модулем, можна зазначити, що використаний підхід орієнтований на використання відкритих технологій, що дозволяє знизити витрати на ліцензування. Розроблений модуль також підтримує гарячі резервні копії та використовує сучасні алгоритми стиснення (gzip, zstd, lz4), що дозволяє ефективно зберігати великі обсяги даних. Однак, на відміну від MySQL Enterprise Backup, розроблений модуль не включає спеціальних можливостей для шифрування або автоматизованого керування процесом резервного копіювання на рівні інтерфейсу користувача. Це означає, що в реалізованому рішенні користувач має більше гнучкості в налаштуванні, але також несе більшу відповідальність за налаштування безпеки.

Percona XtraBackup є відкритим інструментом для створення резервних копій MySQL, що підтримує функціональність гарячого резервування та інкрементальних резервних копій, а також дозволяє працювати з великими обсягами даних. Однією з ключових переваг Percona XtraBackup є його можливість працювати з базами даних у форматах InnoDB і XtraDB, забезпечуючи високу швидкість і надійність операцій. Також, на відміну від MySQL Enterprise Backup, Percona XtraBackup є безкоштовним і відкритим продуктом, що робить його доступним для широкого кола користувачів [21].

У порівнянні з Percona XtraBackup, розроблений модуль має схожу функціональність, що стосується виконання гарячих резервних копій, стиснення і вибору даних для архівації. Однак, створений модуль більше орієнтований на автоматизацію процесів і інтерфейс для користувачів, де кожен етап архівування може бути налаштований через REST-інтерфейс. Це дозволяє досягти високої гнучкості у використанні модуля в різних середовищах та сценаріях, зокрема для інтеграції в більш складні програмні системи. Крім того, використання REST-ендпоінтів дає змогу легко автоматизувати процеси архівування, що є важливою перевагою в порівнянні з більш класичними інтерфейсами для роботи з резервними копіями, які використовує Percona XtraBackup.

ClusterControl це ще одна комерційна платформа для управління базами даних, яка надає широкий набір інструментів для адміністрування, моніторингу, резервного копіювання та відновлення баз даних, зокрема для MySQL. ClusterControl підтримує автоматизоване резервне копіювання, шифрування, моніторинг продуктивності та збереження цілісності даних для MySQL-кластерів. Вона також дозволяє проводити автоматичне відновлення даних у разі збоїв і включає зручний графічний інтерфейс для управління та контролю за усіма процесами [22].

Порівнюючи ClusterControl з розробленим модулем, можна зазначити кілька ключових відмінностей. ClusterControl пропонує більш комплексне рішення для управління великими MySQL-кластерами, зокрема для

адміністрування кластерних середовищ Galera, MySQL Replication, а також інструменти для резервного копіювання на рівні всього кластера. Це робить ClusterControl особливо корисним для великих підприємств з високими вимогами до доступності та безпеки даних. Однак, на відміну від розробленого модуля, який зосереджений на архівуванні та стисненні даних в окремих таблицях або наборах таблиць, ClusterControl більше орієнтований на резервне копіювання всього кластера.

Розроблений модуль, на відміну від ClusterControl, має значно більшу гнучкість і простоту в налаштуванні. Він підтримує архівування лише вибраних даних, що дозволяє користувачам більш точно контролювати, які саме таблиці або записи будуть заархівовані. Це може бути корисно в умовах, коли необхідно архівувати лише частину даних (наприклад, застарілі записи або тимчасові дані), що є більш обмеженим варіантом порівняно з широким спектром функцій резервного копіювання для всього кластера, що надається ClusterControl.

Ще однією важливою відмінністю є те, що ClusterControl надає повний набір інструментів для адміністрування баз даних, включаючи функціональність для автоматичного масштабування, управління реплікацією і відновлення після аварій. Це робить ClusterControl підходящим вибором для підприємств, які потребують комплексного рішення для керування базами даних у великих масштабах, зокрема для кластерних архітектур. Натомість, розроблений модуль більше фокусується на задачах архівування та стиснення даних, що робить його більш вузькоспеціалізованим інструментом.

Підсумовуючи порівняння розробленого модуля з такими популярними системами архівування, як MySQL Enterprise Backup, Percona XtraBackup та ClusterControl, можна зробити кілька ключових висновків.

Розроблений модуль має очевидні переваги в простоті використання та гнучкості, оскільки він орієнтований на архівування та стиснення конкретних таблиць чи наборів даних, що дозволяє користувачам більш точно налаштовувати процес архівування залежно від їхніх потреб. У порівнянні з

MySQL Enterprise Backup та Percona XtraBackup, які більше зосереджені на резервному копіюванні та відновленні даних на рівні всієї бази даних, розроблений модуль пропонує спеціалізоване рішення для архівування, що може бути корисним у сценаріях, де необхідно працювати лише з певними частинами бази даних. Він також включає підтримку декількох алгоритмів стиснення, що забезпечує збереження цілісності даних при зменшенні їхнього обсягу.

У той же час, ClusterControl, що надає повний набір інструментів для управління кластерними середовищами та резервним копіюванням, є набагато більш комплексним рішенням. Воно пропонує широкі можливості для масштабування, управління реплікацією, а також зручний інтерфейс для адміністрування баз даних у великих середовищах. Однак, для організацій, які потребують простого та ефективного інструменту для архівування та стиснення окремих таблиць або записів, розроблений модуль може бути більш зручним, оскільки він має менші вимоги до інфраструктури та не потребує складних налаштувань.

Таким чином, вибір між цими рішеннями залежить від конкретних вимог організації. Якщо необхідне комплексне управління базами даних з функціональністю для резервного копіювання, відновлення та масштабування, ClusterControl або Percona XtraBackup можуть стати кращим варіантом. Якщо ж мета полягає у спеціалізованому архівуванні та стисненні частини даних, розроблений модуль може бути більш гнучким і ефективним рішенням.

4.5 Висновки

У четвертому розділі проведено експериментальну перевірку та оцінку ефективності розробленого програмного модуля автоматизації архівування даних. Було сформульовано план-програму експерименту, яка визначила основні

критерії перевірки, зокрема коректність виконання архівних операцій, продуктивність системи та відповідність функціональним вимогам.

Результати тестування та верифікації показали, що модуль успішно виконує всі основні функції: архівування, зберігання та відновлення даних із забезпеченням їх цілісності та безпеки. Проведено оцінку продуктивності, яка засвідчила високу швидкість виконання операцій навіть за значних обсягів даних, а також здатність системи масштабуватися під зростаючі потреби.

Порівняння з іншими підходами до архівування виявило переваги запропонованого рішення в аспектах інтеграції з реляційними базами даних, автоматизації рутинних процесів та забезпечення зручності користувацького інтерфейсу. Таким чином, розроблений модуль підтвердив свою ефективність і доцільність впровадження в практичну діяльність.

5 ОХОРОНА ПРАЦІ

Даний розділ зосереджений на забезпеченні безпечних і здорових умов праці для всіх учасників процесу, від розробників до користувачів системи. Враховуючи, що робота з інформаційними системами передбачає тривале сидіння за комп'ютером, тестування, обробку великих обсягів даних та взаємодію з різними компонентами, важливо дотримуватися вимог до робочого середовища. У цьому розділі будуть розглянуті умови праці при використанні модуля для архівування даних, а також рекомендації щодо організації робочого місця оператора, щоб знизити ризики фізичного та психоемоційного перенавантаження.

5.1 Аналіз умов праці при роботі з інформаційними системами

Під час експлуатації модуля оператори системи будуть виконувати різні операції з архівування, зберігання та обробки великих обсягів даних. Тому важливо забезпечити комфортні та безпечні умови праці для всіх користувачів.

Оператори системи, які працюють з архівуванням даних, часто взаємодіють з великими масивами інформації, що вимагає тривалих сесій за комп'ютером. Це може призвести до перевантаження зору та фізичних напруг, оскільки тривала робота за монітором може негативно впливати на зір, поставу та загальний фізичний стан. Тому важливо забезпечити належне робоче середовище, зокрема, зручні робочі місця з ергономічними меблями та пристроями. Важливим аспектом є також контроль за освітленням та вентиляцією приміщень, де здійснюється робота з системою, щоб зменшити ризики перевтоми [23].

Система повинна бути спроектована таким чином, щоб зменшити ймовірність стресових ситуацій у процесі експлуатації. Наприклад, наявність чітких повідомлень про статус виконання операцій, логування подій та автоматизоване повідомлення про помилки дозволяють оператору своєчасно реагувати на потенційні проблеми, що в свою чергу знижує психоемоційне навантаження. Протягом тривалих сесій роботи також має бути можливість робити регулярні перерви для відпочинку, що дозволяє уникнути фізичної та розумової перевтоми.

Нарешті, важливо розглянути умови доступу до системи для всіх користувачів. З огляду на специфіку роботи з даними, доступ до модуля має бути обмежений відповідно до рівня доступу та відповідальності користувачів, щоб зменшити ризики випадкових або несанкціонованих змін даних. Система повинна передбачати можливість налаштування прав доступу та надання чітких інструкцій щодо того, як працювати з даними без шкоди для їх цілісності та безпеки.

5.2 Вимоги до організації робочого місця оператора

Оскільки операція архівування часто вимагає тривалих сесій за комп'ютером, важливо організувати робоче місце таким чином, щоб мінімізувати ризики для здоров'я оператора та забезпечити комфортне і продуктивне середовище для роботи.

Одним з основних аспектів є ергономіка робочого місця. Оператор повинен мати зручний стіл та стілець, що забезпечують правильну поставу і мінімізують навантаження на спину, шию та руки. Важливо, щоб висота столу і стільця регулювалася таким чином, щоб оператор міг працювати в природній позі, зберігаючи комфорт протягом довгих годин роботи [23]. Крім того,

клавіатура і миша повинні бути розташовані на такій висоті, щоб запобігти перенавантаженню зап'ясть.

Оператори повинні працювати в добре освітлених приміщеннях з достатнім рівнем світла, що дозволяє чітко розглядати інформацію на екрані комп'ютера. Відсутність відблисків на екрані та правильне розташування джерела світла допомагають уникнути зорового навантаження та втрати зору. Важливо також, щоб освітлення було м'яким, щоб зменшити напругу на очі під час тривалого використання комп'ютера.

Для тривалої роботи важливо створити умови для регулярних перерв. Робочі місця повинні бути обладнані так, щоб оператори могли легко здійснювати фізичні вправи, ходити по офісу або просто відпочивати протягом коротких перерв. Регулярний рух допомагає зняти м'язове напруження та знизити ризики розвитку хронічних захворювань, пов'язаних з тривалим сидінням.

Ще одним важливим аспектом є належне організування робочого місця з точки зору технічного оснащення. Оператор повинен мати доступ до всіх необхідних інструментів для ефективної роботи з програмним модулем, зокрема комп'ютер з достатньою потужністю, двома моніторами (для зручного перегляду документації та даних одночасно), а також надійний доступ до мережі для роботи з базою даних та іншими ресурсами. У разі необхідності архівування даних великого обсягу, система повинна підтримувати швидкий доступ до інформації без збоїв у роботі.

Не менш важливою є організація безпеки на робочому місці. Для цього необхідно передбачити систему доступу до робочих станцій, щоб лише авторизовані користувачі могли взаємодіяти з даними. Це дозволить знизити ризики несанкціонованого доступу або змін до архівованих даних. Робочі місця повинні бути обладнані належним чином для підтримки захисту інформації, включаючи системи резервного копіювання та антивірусного захисту.

ВИСНОВКИ

У процесі дослідження та розробки програмного модуля для архівування даних для реляційних баз даних було вирішено низку важливих завдань, пов'язаних з ефективним збереженням і переміщенням даних. Модуль, що був розроблений, відповідає вимогам збереження та забезпечення цілісності даних, а також пропонує можливості для вибору різних алгоритмів стиснення, що дає змогу досягати оптимальних результатів за часом і обсягом даних.

Особливу увагу було приділено безпеці даних під час їх архівування та зберігання, де були реалізовані механізми шифрування, автентифікації користувачів, а також створено систему моніторингу і журналювання для відслідковування та контролю всіх операцій, пов'язаних з архівуванням. Завдяки цьому система забезпечує не лише ефективність, а й високий рівень захисту архівованих даних від несанкціонованого доступу та потенційних загроз.

Під час експериментального тестування модуля було перевірено його працездатність за різних умов, включаючи роботу з великими обсягами даних. Результати тестів продемонстрували ефективність розробленого рішення та його здатність до масштабування в умовах зростаючих вимог. Програмний модуль здатний справлятися з великими обсягами даних, а вибір між різними методами стиснення дозволяє оптимізувати роботу в залежності від конкретних потреб.

Також було здійснено порівняння з існуючими підходами, такими як MySQL Enterprise Backup, Percona XtraBackup і ClusterControl, що показало конкурентоспроможність розробленого рішення. Розроблена система має низку переваг, зокрема більш гнучкий підхід до архівування та здатність працювати з різними алгоритмами стиснення, що дозволяє вибрати оптимальний варіант для кожної конкретної ситуації.

Завершенням роботи стало визначення вимог до організації робочого місця оператора та охорони праці при роботі з інформаційними системами, що є важливим аспектом для забезпечення безпеки і здоров'я користувачів системи.

Таким чином, розроблений програмний модуль для архівування даних відповідає встановленим вимогам і демонструє високий рівень ефективності та безпеки. Однак для досягнення оптимальних результатів рекомендується враховувати специфічні умови використання, такі як обсяг даних і вимоги до швидкості архівування, що може вплинути на вибір алгоритмів стиснення та конфігурацію системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології, освітньо-професійних програм: «Автоматизоване управління технологічними процесами», «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. – Харків: ХНУРЕ, 2021. – 55 с.

2. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання.

3. Галанін Юрій, Іванов Леонід. Алгоритмізація та імплементація у коді процесу архівування даних у реляційній БД. "Інформаційні технології і автоматизація", 2024. 268 - 270.

4. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom. Database Systems: The Complete Book, 2008. – 1248p.

5. Darren Quick. Cloud Storage Forensics, 2013. – 208p.

6. W. Curtis Preston. Backup & Recovery: Inexpensive Backup Solutions for Open Systems, 2007. – 758p.

7. Markus Winand. SQL Performance Explained, 2012. – 204 p.

8. Java Database Connectivity (JDBC) documentation // Сайт. URL: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/ (дата звернення: 12.11.2024).

9. Java Cryptography Architecture (JCA) documentation. // Сайт. URL: https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html (дата звернення: 18.11.2024).

10. Catalin Tudose. JUnit in Action, Third Edition, 2020. 525p.

11. Guidelines for Data Encryption // National Institute of Standards and Technology, USA. URL: https://csrc.nist.gov/pubs/sp/800/175/b/r1/final. (дата звернення: 16.11.2024).

12. Vue.js documentation // Сайт. URL: https://vuejs.org/guide (дата звернення: 26.11.2024).

13. Bill Scott, Theresa Neil. Designing Web Interfaces: Principles and Patterns for Rich Interactions, 2009. – 334p.

14. Michael Kofler. MySQL 8 for Developers, 2008. – 659p.

15. MySql documentation // Сайт. URL: https://dev.mysql.com/doc/ (дата звернення: 10.11.2024).

16. Internet Security Research Group. TLS and HTTPS Best Practices // Сайт. URL: https://community.letsencrypt.org/t/best-practices-for-a-tls-server/75517 (дата звернення: 16.11.2024).

17. GZIP documentation // Сайт. URL: https://www.gnu.org/software/gzip/ (дата звернення: 7.11.2024).

18. ZSTD documentation // Сайт. URL: https://facebook.github.io/zstd/ (дата звернення: 7.11.2024).

19. LZ4 documentation // Сайт. URL: https://lz4.github.io/lz4/ (дата звернення: 8.11.2024).

20. MySQL Enterprise Backup documentation // Сайт. URL: https://dev.mysql.com/doc/mysql-enterprise-backup/en/ (дата звернення: 12.12.2024).

21. Percona XtraBackup documentation // Сайт. URL: https://www.percona.com/doc/percona-xtrabackup (дата звернення: 12.12.2024).

22. ClusterControl documentation // Сайт. URL: https://severalnines.com/clustercontrol (дата звернення: 13.12.2024).

23. Методичні вказівки до самостійної роботи з дисципліни "Охорона праці в галузі" для студентів усіх спеціальностей галузі автоматизації та приладобудування денної форми навчання / упоряд.: Г. В. Пронюк, Н. М.

Сердюк, Т. Є. Стиценко ; М-во освіти і науки України, ХНУРЕ. – Харків : ХНУРЕ, 2016. – 116 с.