МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА УТИЛІТИ МОНІТОРИНГУ СТАТУСУ РЕПЛІКАЦІЇ БАЗ ДАНИХ»**

Виконав: студент 4 курсу, групи 6.1261-з
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)
освітньої програми Інформаційні системи та технології
(назва освітньої програми)
В.С. Свириденко
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, к.т.н. Добровольський Г.А.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри програмної інженерії, к.ф.-м.н. Мильцев О. М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 126 Інформаційні системи та технології
(шифр і назва)

Освітня програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

" ____ " _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Свириденку Василю Станіславовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка утиліти моніторингу статусу реплікації баз даних

керівник роботи Добровольський Геннадій Анатолійович, к.т.н.
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2002-с

2. Строк подання студентом роботи 15.05.2025

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка утиліти моніторингу статусу реплікації баз даних.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 22.12.2024

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	23.12.2024	
2.	Збір вихідних даних.	25.12.2024	
3.	Обробка методичних та теоретичних	30.12.2024	
	джерел.		
4.	Розробка першого та другого розділу.	10.01.2025	
5.	Розробка третього розділу.	12.02.2025	
6.	Оформлення та нормоконтроль	10.05.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	29.05.2025	

Студент _____ В.С. Свириденко
(підпис) (ініціали та прізвище)

Керівник роботи _____ Г.А. Добровольський
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка утиліти моніторингу статусу реплікації баз даних»: 38 с., 9 рис., 8 джерел, 1 додаток.

MS SQL SERVER, TELEGRAM BOT API, АВТОМАТИЗАЦІЯ МОНІТОРИНГУ, МОНІТОРИНГ РЕПЛІКАЦІЇ, РЕПЛІКАЦІЯ БАЗ ДАНИХ, СПОВІЩЕННЯ.

Об’єкт дослідження: процес моніторингу статусу реплікації баз даних MS SQL Server.

Мета роботи: розробка утиліти, яка забезпечує автоматизований моніторинг статусу реплікації баз даних MS SQL Server, виявлення та своєчасне сповіщення про проблеми.

Метод дослідження: аналіз літератури, проектування, розробка, тестування.

У кваліфікаційній роботі розроблено утиліту для автоматизованого моніторингу статусу реплікації баз даних MS SQL Server. Актуальність роботи обумовлена необхідністю забезпечення високої доступності та надійності інформаційних систем, в яких реплікація є критично важливою технологією.

Проаналізовано існуючі методи та інструменти моніторингу реплікації. Визначено вимоги до утиліти та розроблено її архітектуру, яка базується на модульному принципі. Реалізовано основні модулі утиліти, включаючи модулі підключення до бази даних, отримання даних про реплікацію, аналізу стану реплікації, та надсилання сповіщень в Telegram.

Утиліта забезпечує автоматизоване отримання даних про статус реплікації, виявлення та аналіз помилок та затримок синхронізації, а також надсилання сповіщень про проблеми в Telegram. Для забезпечення надійності роботи утиліта веде журнали подій. Результати тестування підтвердили ефективність розробленої утиліти. Її використання дозволяє підвищити оперативність реагування на збої, мінімізувати простої системи та зменшити ризики втрати даних.

SUMMARY

Bachelor's Qualification Theses: "Development of the Utility Application to Monitor Database Replication Status": 38 pages, 9 figures, 8 references, 1 supplements.

MS SQL SERVER, TELEGRAM BOT API, MONITORING AUTOMATION, REPLICATION MONITORING, DATABASE REPLICATION, NOTIFICATIONS.

Object of study: the process of monitoring the replication status of MS SQL Server databases.

Purpose of the work: development of a utility that provides automated monitoring of the replication status of MS SQL Server databases, detection and timely notification of problems.

Research method: literature analysis, design, development, testing.

This bachelor's qualification work describes the development of a utility for automated monitoring of the replication status of MS SQL Server databases. The relevance of the work is due to the necessity of ensuring high availability and reliability of information systems, in which replication is a critically important technology.

Existing methods and tools for replication monitoring were analyzed. The requirements for the utility were defined, and its architecture, based on a modular principle, was developed. The main modules of the utility were implemented, including modules for database connection, replication data retrieval, replication status analysis, and sending notifications to Telegram.

The utility provides automated data retrieval on the replication status, detection and analysis of errors and synchronization delays, as well as sending notifications about problems via Telegram. For ensuring the reliability of the utility's operation, it maintains event logs.

The testing results confirmed the effectiveness of the developed utility. Its use allows for increased responsiveness to failures, minimization of system downtime, and reduction of the risk of data loss.

ЗМІСТ

Завдання 2
Реферат 4
Summary 5
Вступ 8
1 Огляд технологій реплікації баз даних 10
1.1 Поняття та принципи реплікації баз даних 10
1.2 Типи реплікації баз даних 11
1.3 Методи моніторингу реплікації 12
1.4 Огляд інструментів для моніторингу реплікації 13
1.5 Висновки за розділом 14
2 Реалізація утиліти мониторингу реплікації 15
2.1 Архітектура та загальний підхід 15
2.1.1 Визначення основних модулів 15
2.1.2 Опис функцій кожного модуля 15
2.1.3 Правила написання модуля та взаємодія з основною системою 16
2.1.4 Взаємодія між модулями 17
2.1.5 Структурна схема інформаційної системи 18
2.1.6 Технології та стек розробки 19
2.1.7 Як користувачі взаємодіють з системою 20
2.1.8 Масштабованість, надійність, безпека 21
2.2 Основні функціональні можливості 22
2.3 Взаємодія з базою даних 24
2.4 Інтеграція з Telegram-ботом 26
3 Тестування та аналіз результатів 27
3.1 Тестування утиліти 27
3.2 Аналіз продуктивності 30
3.3 Результати роботи 30

3.4 Можливі покращення..... 32
Висновки 33
Перелік посилань..... 35
Додаток А Приклади коду..... 36

ВСТУП

В епоху стрімкого розвитку цифрових технологій, коли інформація стала найціннішим ресурсом, забезпечення надійного та безперебійного функціонування баз даних набуває критичного значення. Бази даних, як фундаментальний елемент будь-якої інформаційної системи, використовуються в найрізноманітніших сферах діяльності людини. Від невеликих локальних систем управління, що обслуговують потреби окремих підприємств, до масштабних корпоративних рішень, які обробляють колосальні обсяги даних і забезпечують функціонування критично важливих сервісів, бази даних відіграють ключову роль у забезпеченні ефективності та конкурентоспроможності сучасних організацій.

В умовах постійно зростаючих вимог до доступності, цілісності та узгодженості даних, забезпечення безперервного доступу до інформації стає першочерговим завданням. Будь-які збої або перерви в роботі баз даних можуть призвести до значних фінансових втрат, пошкодження репутації компанії та навіть загрози безпеці. Тому, поряд із забезпеченням захисту даних від несанкціонованого доступу та кібератак, особлива увага приділяється механізмам, що забезпечують високу доступність та відмовостійкість баз даних.

Одним із найефективніших та найпоширеніших механізмів, що дозволяють досягти цих цілей, є реплікація баз даних. Реплікація, по суті, є процесом створення та підтримки декількох копій однієї і тієї ж бази даних на різних серверах. Це дозволяє не тільки забезпечити збереження даних у разі виходу з ладу одного з серверів, але й значно підвищити продуктивність системи за рахунок розподілу навантаження між декількома серверами. Крім того, реплікація забезпечує можливість створення резервних копій даних, що є критично важливим для відновлення системи після збоїв або аварій.

Проте, незважаючи на численні переваги, процес реплікації є досить

складним і може супроводжуватися низкою проблем. Затримки синхронізації між копіями бази даних, порушення узгодженості даних, конфлікти при одночасній зміні даних на різних серверах, а також інші збої та помилки можуть негативно впливати на продуктивність і коректність роботи всієї системи. Особливо гостро ці проблеми проявляються у великих розподілених системах, де реплікація охоплює велику кількість серверів і обробляє величезні обсяги даних.

У зв'язку з цим, забезпечення ефективного моніторингу та контролю за процесом реплікації є надзвичайно важливою задачею. Необхідно мати надійний інструмент, який дозволяє в режимі реального часу відстежувати стан реплікації, виявляти та оперативно усувати потенційні проблеми, а також запобігати виникненню критичних ситуацій. Використання традиційних методів адміністрування, таких як ручна перевірка журналів подій, періодичний запуск SQL-запитів для перевірки статусу реплікації або візуальний аналіз моніторингових панелей бази даних, є не тільки неефективним, але й надзвичайно трудомістким. У випадках, коли необхідно контролювати велику кількість серверів та активних процесів реплікації, такий підхід стає практично неможливим і значно ускладнює оперативне реагування на виникаючі проблеми. Це, в свою чергу, може призвести до серйозних збоїв у роботі системи, втрати даних та інших негативних наслідків.

1 ОГЛЯД ТЕХНОЛОГІЙ РЕПЛІКАЦІЇ БАЗ ДАНИХ

1.1 Поняття та принципи реплікації баз даних

Реплікація баз даних є процесом створення та підтримки копій даних на різних серверах для забезпечення їх узгодженості та доступності. Основними цілями реплікації є підвищення відмовостійкості, збільшення швидкості доступу до даних і балансування навантаження між серверами. [1]

Реплікація є ключовою технологією в сучасних системах управління базами даних (СУБД), яка дозволяє підвищити надійність, доступність та продуктивність інформаційних систем. Вона забезпечує можливість розподілу даних між декількома серверами, що дозволяє уникнути простою системи у разі виходу з ладу одного з серверів, а також оптимізувати доступ до даних для користувачів, розташованих у різних географічних регіонах.

Основні принципи реплікації включають:

Консистентність (Consistency) – усі копії бази даних повинні містити актуальні та узгоджені дані. Це означає, що будь-які зміни, внесені в одну копію бази даних, повинні бути відображені у всіх інших копіях. Забезпечення консистентності є однією з найскладніших задач у реплікації, особливо в розподілених системах, де затримки передачі даних можуть призводити до тимчасової розбіжності між копіями.

Доступність (Availability) – система повинна бути готова до обробки запитів користувачів у будь-який момент часу. Реплікація підвищує доступність системи за рахунок того, що у разі виходу з ладу одного з серверів, інші сервери продовжують працювати та обслуговувати запити.

Толерантність до розділення (Partition Tolerance) – система повинна залишатися працездатною навіть у разі виникнення проблем з мережевим зв'язком, коли окремі вузли системи виявляються ізольованими один від одного. Реплікація забезпечує можливість продовження роботи системи навіть

у таких умовах, хоча при цьому може виникнути тимчасова розбіжність між копіями даних.

Ці три принципи, відомі як CAP-теорема, визначають компроміси, які необхідно враховувати при проектуванні систем реплікації. CAP-теорема стверджує, що в розподіленій системі неможливо одночасно забезпечити всі три властивості: консистентність, доступність і толерантність до розділення. При проектуванні системи необхідно зробити вибір між цими властивостями, залежно від вимог конкретного застосування.

1.2 Типи реплікації баз даних

Реплікація в Microsoft SQL Server представлена кількома основними підходами [2]:

Snapshot Replication (Знімкова реплікація) – це найпростіший тип реплікації, при якому на підписників передається повна копія даних на певний момент часу. Знімкова реплікація використовується, коли оновлення даних відбуваються рідко або немає необхідності в постійній синхронізації між сервером-джерелом та підписниками. Цей тип реплікації є найменш вимогливим до ресурсів, але забезпечує найменшу узгодженість даних.

Transactional Replication (Транзакційна реплікація) – цей тип реплікації дозволяє передавати транзакції в режимі реального часу або з мінімальною затримкою, що забезпечує високу узгодженість даних між сервером-джерелом та підписниками. Транзакційна реплікація використовується в системах, де критично важливо мати актуальні дані на всіх підписниках. Цей тип реплікації є більш вимогливим до ресурсів, ніж знімкова реплікація, але забезпечує більш високу узгодженість даних.

Merge Replication (Реплікація злиттям) – цей тип реплікації забезпечує двосторонню синхронізацію змін між кількома вузлами, дозволяючи вносити зміни як на основному сервері, так і на репліках. Реплікація злиттям

використовується в системах, де користувачі можуть працювати з даними в автономному режимі, а потім синхронізувати зміни з іншими вузлами. Цей тип реплікації є найбільш складним у реалізації та управлінні, але забезпечує найбільшу гнучкість.

Peer-to-Peer Replication (Однорангова реплікація) – цей тип реплікації реалізує рівноправний обмін даними між декількома серверами, що дозволяє розподілити навантаження та підвищити відмовостійкість системи. Однорангова реплікація використовується в системах, де немає виділеного сервера-джерела, і всі сервери мають рівні права на зміну даних.

Вибір конкретного типу реплікації залежить від вимог конкретного застосування, таких як частота оновлень даних, вимоги до узгодженості даних, доступність ресурсів та інші фактори.

1.3 Методи моніторингу реплікації

Моніторинг реплікації є критично важливим процесом, який дозволяє відстежувати стан синхронізації даних між серверами та вчасно виявляти помилки або збої, що можуть призвести до порушення узгодженості даних або простою системи. Ефективний моніторинг реплікації дозволяє адміністраторам оперативно реагувати на виникаючі проблеми, мінімізувати час простою системи та забезпечити надійне функціонування інформаційної системи. [3] Існують різні методи моніторингу реплікації, кожен з яких має свої переваги та недоліки.

Журнал змін (log-based monitoring) – цей метод передбачає аналіз журналів транзакцій бази даних для відстеження змін, що відбуваються в процесі реплікації. Аналіз журналів дозволяє отримати детальну інформацію про кожну транзакцію, що була реплікована, та виявити потенційні проблеми, такі як затримки, помилки або конфлікти.

Перевірка стану реплікації (replication status queries) – цей метод

передбачає виконання спеціальних SQL-запитів до системних таблиць бази даних для отримання інформації про поточний стан реплікації. Запити можуть повертати інформацію про стан підписників, час останньої синхронізації, наявність помилок або затримок.

Системи моніторингу (Monitoring Tools) – цей метод передбачає використання спеціалізованих програмних засобів, які автоматично збирають, аналізують та візуалізують інформацію про стан реплікації. Системи моніторингу можуть забезпечувати комплексну картину стану реплікації, включаючи графічне відображення даних, оповіщення про проблеми та інші корисні функції.

Вибір конкретного методу моніторингу залежить від вимог конкретного застосування, доступних ресурсів та технічних можливостей.

1.4 Огляд інструментів для моніторингу реплікації

Для забезпечення ефективного моніторингу реплікації баз даних існує широкий спектр інструментів, які відрізняються за своїми функціональними можливостями, вартістю та складністю використання. Вибір конкретного інструменту залежить від вимог конкретного застосування, розміру та складності інфраструктури, а також наявності ресурсів та технічних знань.

Серед найбільш популярних інструментів для моніторингу реплікації баз даних можна виділити:

Zabbix – це потужна система моніторингу з відкритим вихідним кодом, яка забезпечує моніторинг різноманітних параметрів IT-інфраструктури, включаючи сервери, мережі, операційні системи та бази даних. Zabbix підтримує моніторинг багатьох популярних СУБД, таких як MySQL, PostgreSQL, Oracle та Microsoft SQL Server. Однак, для моніторингу реплікації Zabbix часто потребує складного налаштування та використання додаткових скриптів або плагінів, що може ускладнити його використання для користувачів

з обмеженим досвідом (Zabbix Documentation – Database Monitoring, URL : https://www.zabbix.com/
documentation/current/en/manual/config/items/itemtypes/odbc_checks).

Datadog – це хмарний сервіс для моніторингу та аналізу IT-інфраструктури, який забезпечує збір, візуалізацію та аналіз даних з різних джерел, включаючи бази даних. Datadog пропонує інтеграції з багатьма популярними СУБД та забезпечує моніторинг ключових показників реплікації. Однак, використання Datadog передбачає фінансові витрати на підписку та залежність від стороннього постачальника послуг, що може бути неприйнятним для деяких організацій з міркувань безпеки або економічних причин (Datadog Database Monitoring, URL: https://www.datadoghq.com/database-monitoring/).

1.5 Висновки за розділом

У даному розділі було розглянуто основні аспекти реплікації баз даних, включаючи поняття, принципи, типи та методи моніторингу. Було підкреслено важливість реплікації для забезпечення надійності, доступності та продуктивності інформаційних систем, а також розглянуто основні проблеми, пов'язані з процесом реплікації.

Аналіз існуючих інструментів моніторингу показав, що вони не завжди відповідають специфічним вимогам, таким як простота налаштування, можливість централізованого моніторингу кількох серверів, автоматичне виявлення помилок та інтеграція з системами оповіщення, такими як Telegram.

Саме тому було прийнято рішення про розробку власної утиліти моніторингу, яка буде адаптована під конкретні потреби, забезпечить гнучкість, масштабованість та автономність. Розробка власної утиліти дозволить вирішити проблеми, пов'язані з обмеженнями існуючих рішень, і створити ефективний інструмент для моніторингу реплікації баз даних.

2 РЕАЛІЗАЦІЯ УТИЛІТИ МОНИТОРИНГУ РЕПЛІКАЦІЇ

2.1 Архітектура та загальний підхід

Утиліта моніторингу реплікації баз даних побудована за модульним принципом, що забезпечує її гнучкість та розширюваність. Модульна архітектура дозволяє легко додавати нові функції або змінювати існуючі, не впливаючи на роботу інших компонентів системи.

2.1.1 Визначення основних модулів

Утиліта включає наступні основні модулі:

EncryptionManager: Відповідає за шифрування та дешифрування конфіденційних даних, таких як параметри підключення до бази даних та API-ключі.

DatabaseManager: Забезпечує підключення до бази даних MS SQL та виконання запитів для отримання інформації про реплікацію та стан системи.

NotificationService: Відповідає за надсилання повідомлень про стан реплікації в Telegram.

ConfigManager: Керує завантаженням та збереженням налаштувань утиліти, забезпечуючи їхнє шифрування.

ReplicationMonitor: Основний модуль, який виконує моніторинг реплікації, координуючи роботу інших модулів.

2.1.2 Опис функцій кожного модуля

EncryptionManager:

- генерує ключ шифрування, якщо він відсутній;
- завантажує ключ шифрування з файлу;

– шифрує дані за допомогою алгоритму Fernet;
– дешифрує дані.

DatabaseManager:

– встановлює з'єднання із сервером MS SQL;
– отримує налаштування реплікації з системних таблиць;
– перетворює час з числового формату в строковий;
– перевіряє наявність помилок синхронізації;
– виявляє застарілі записи синхронізації.

Notificationservice:

– надсилає повідомлення в Telegram через Telegram Bot API;
– формує текст повідомлень про помилки, затримки та успішну синхронізацію.

ConfigManager:

– завантажує налаштування з файлів, якщо вони існують;
– запитує налаштування у користувача, якщо вони відсутні;
– шифрує та зберігає налаштування у файли;
– дешифрує та повертає налаштування.

ReplicationMonitor:

– запускає моніторинг реплікації в окремому потоці;
– отримує налаштування реплікації через DatabaseManager;
– періодично перевіряє наявність помилок та затримок;
– надсилає повідомлення про стан реплікації через NotificationService;
– обчислює інтервал між перевірками на основі частоти синхронізації;
– реалізує логіку для надсилання щоденних звітів.

2.1.3 Правила написання модуля та взаємодія з основною системою

Правила написання модуля:

– інкапсуляція: Кожен модуль має чітко визначені обов'язки та

приховує свою внутрішню реалізацію. Це дозволяє змінювати внутрішню логіку модуля без впливу на інші частини системи;

- незалежність: Модулі повинні бути максимально незалежними один від одного. Зв'язки між модулями повинні бути мінімальними та чітко визначеними;
- інтерфейси: Модулі взаємодіють між собою через чітко визначені інтерфейси. Це дозволяє легко замінювати один модуль іншим, якщо вони підтримують той самий інтерфейс;
- відповідальність: Кожен модуль відповідає за виконання конкретної задачі. Це спрощує розробку, тестування та налагодження.

Взаємодія модулів з основною системою:

- події: Модулі можуть сповіщати інші модулі про певні події. Наприклад, DatabaseManager може сповістити ReplicationMonitor про зміни в стані реплікації;
- виклики методів: Модулі можуть викликати методи інших модулів для отримання даних або виконання певних операцій. Наприклад, ReplicationMonitor викликає методи DatabaseManager для отримання інформації про реплікацію;
- потоки даних: Дані передаються між модулями у вигляді структур даних або об'єктів. Наприклад, налаштування підключення до бази даних передаються від ConfigManager до DatabaseManager.

2.1.4 Взаємодія між модулями

Взаємодія між модулями в утиліті моніторингу реплікації здійснюється наступним чином (рис. 2.1):

- main() → Модулі: Головна функція main() створює екземпляри всіх модулів та передає їм необхідні залежності. Наприклад, екземпляр DatabaseManager передається в ReplicationMonitor для забезпечення доступу до бази даних;

– ConfigManager ↔ EncryptionManager: Модуль ConfigManager використовує EncryptionManager для шифрування та дешифрування налаштувань, забезпечуючи їхнє безпечне зберігання;
– ReplicationMonitor → DatabaseManager: Модуль ReplicationMonitor використовує DatabaseManager для отримання інформації про стан реплікації з бази даних. Надсилає запити та обробляє отримані результати;
– ReplicationMonitor → NotificationService: Модуль ReplicationMonitor використовує NotificationService для надсилання повідомлень про події, пов'язані з реплікацією, в Telegram;
– потоки даних: дані про налаштування, стан реплікації та повідомлення передаються між модулями у вигляді словників, списків та рядків.

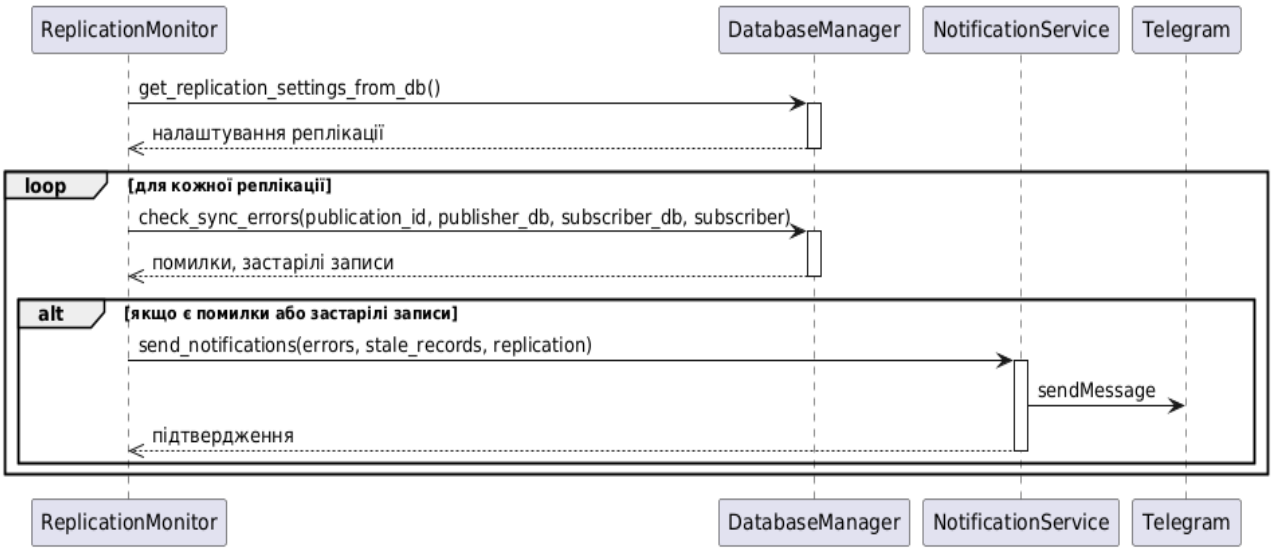


Рисунок 2.1 – Схема взаємодії класів

2.1.5 Структурна схема інформаційної системи

Нижче надана схема інформаційної системи (рис. 2.2), яка описує основні класи програми.

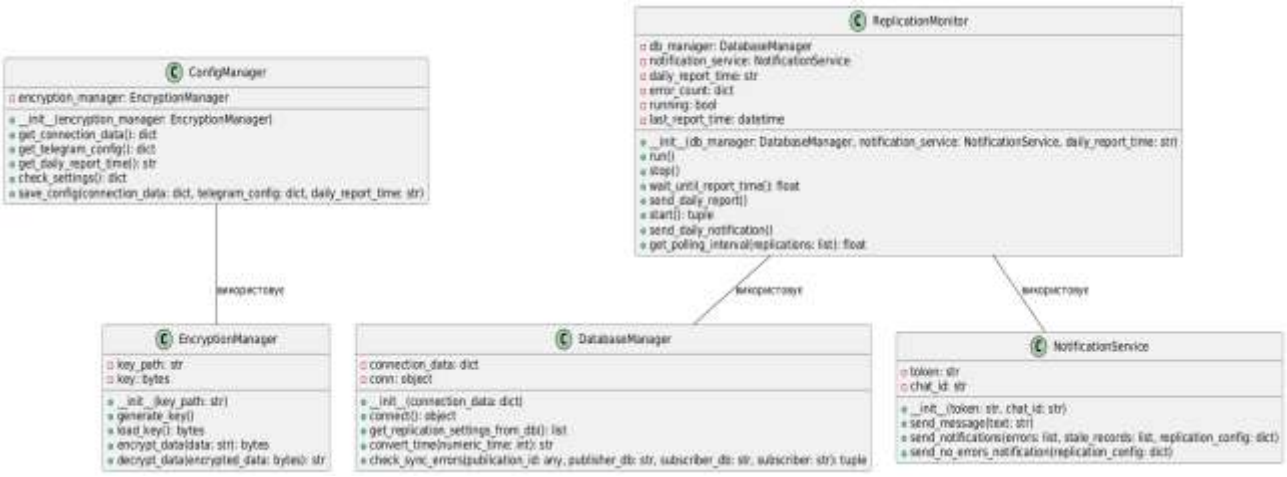


Рисунок 2.2 – Структурна схема класів

2.1.6 Технології та стек розробки

Утиліта моніторингу реплікації розроблена з використанням наступних технологій:

Python: Основна мова програмування.

- Python – це потужна та універсальна мова програмування, яка відрізняється простотою синтаксису, великою кількістю бібліотек та фреймворків, а також широкою підтримкою спільноти розробників. Він добре підходить для розробки утиліт, які потребують швидкої розробки, читабельності коду та легкої підтримки;
- pyodbc: Бібліотека для підключення до баз даних MS SQL Server (Посилання на документацію: https://pypi.org/project/pyodbc/) Ця бібліотека забезпечує простий та ефективний спосіб підключення до баз даних MS SQL Server з Python. Вона дозволяє виконувати SQL-запити, отримувати та обробляти дані, що є необхідним для моніторингу статусу реплікації;
- requests: Бібліотека для виконання HTTP-запитів до Telegram Bot API (Посилання на документацію: https://pypi.org/project/requests/). Ця бібліотека спрощує виконання HTTP-запитів, що необхідно для взаємодії з Telegram Bot API. Вона дозволяє легко надсилати

повідомлення про стан реплікації через Telegram;

- cryptography: Бібліотека для шифрування та дешифрування даних (Посилання на документацію: https://pypi.org/project/cryptography/). Забезпечення безпеки є важливим аспектом будь-якої системи, особливо тієї, що працює з даними. Бібліотека cryptography надає засоби для шифрування та дешифрування даних, що дозволяє захистити конфіденційну інформацію, таку як параметри підключення до бази даних;
- logging: Модуль для ведення журналів роботи утиліти. Модуль logging надає гнучкі можливості для ведення журналів роботи утиліти. Це дозволяє записувати інформацію про події, помилки та інші важливі дані, що полегшує налагодження та аналіз роботи утиліти;
- threading: Модуль для підтримки багатопоточності, що дозволяє паралельно виконувати моніторинг та інші завдання. Моніторинг реплікації може бути ресурсоємним процесом, особливо при роботі з великою кількістю серверів. Модуль threading дозволяє реалізувати багатопоточну обробку, що дозволяє паралельно виконувати моніторинг, надсилання повідомлень та інші завдання, підвищуючи продуктивність та ефективність утиліти.

Вибраний стек розробки є оптимальним для створення ефективної, надійної та безпечної утиліти моніторингу реплікації баз даних MS SQL Server.

2.1.7 Як користувачі взаємодіють з системою

Користувач взаємодіє з системою на етапі налаштування (рис. 2.3).

Утиліта запитує в користувача параметри підключення до бази даних (сервер, ім'я користувача, пароль) та параметри Telegram Bot API (токен, chat ID).

Користувач вводить ці дані через консоль. Тобто користувачеві потрібно знати ці дані для використання утиліти моніторингу. Також треба розуміти та

вміти виправляти помилки у роботі реплікації, які можуть виникати. Тобто ця утиліта призначена для адміністраторів баз даних MS SQL.

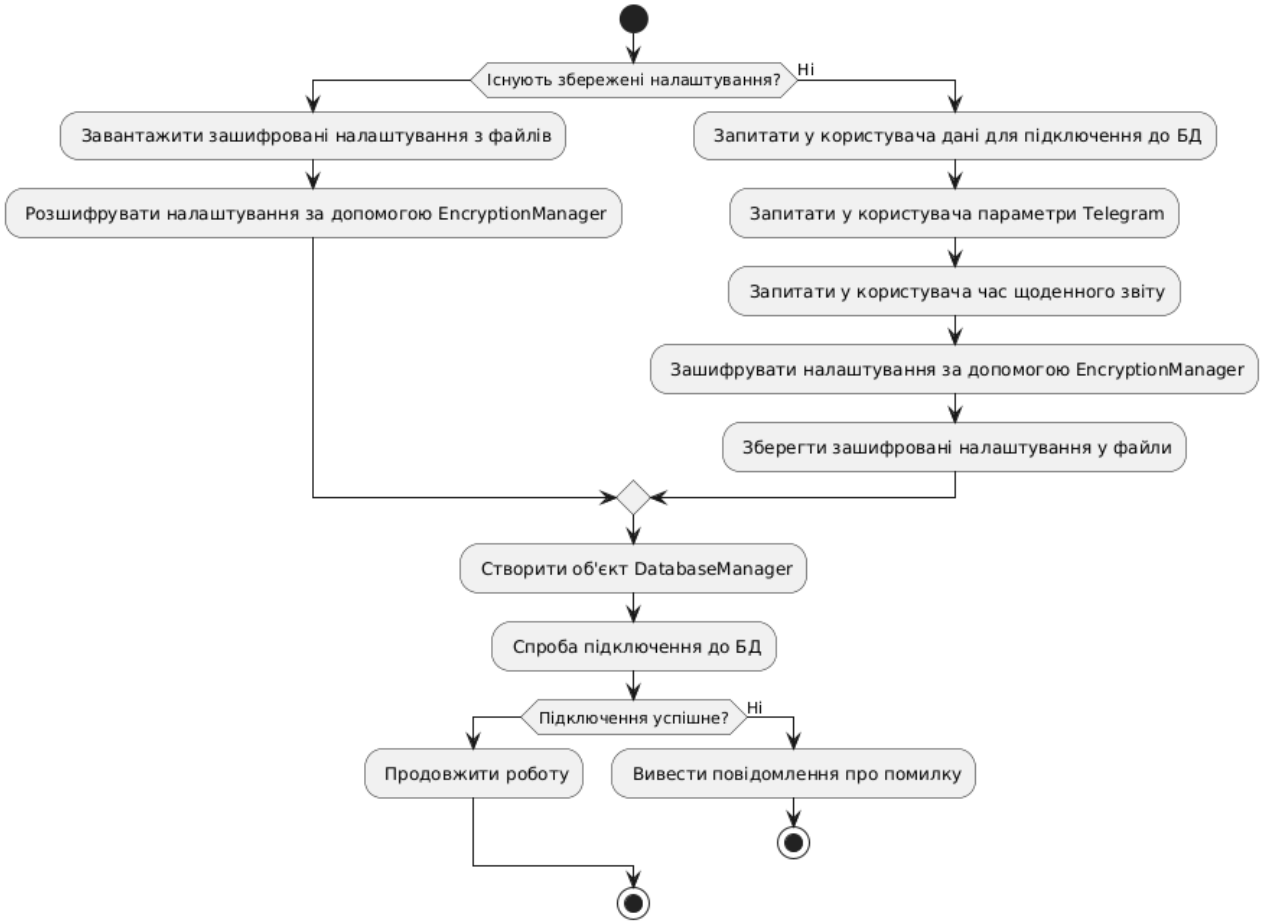


Рисунок 2.3 – Схема взаємодії користувача з програмою

Утиліта зберігає введені дані у зашифрованому вигляді для подальшого використання.

2.1.8 Масштабованість, надійність, безпека

Масштабованість: Модульна архітектура спрощує масштабування утиліти. Додавання нових модулів для моніторингу додаткових параметрів або інтеграції з іншими системами може бути виконане без значних змін в існуючому коді.

Надійність: Використання бібліотеки logging забезпечує реєстрацію всіх

важливих подій та помилок, що полегшує діагностику та усунення проблем. Обробка винятків (try-except блоки) допомагає запобігти аварійному завершенню роботи утиліти.

Безпека.

Паролі та інші конфіденційні дані зберігаються у зашифрованому вигляді за допомогою бібліотеки cryptography.

Доступ до сервера бази даних налаштовується згідно з найкращими практиками безпеки (мінімальні необхідні права, обмеження мережевого доступу тощо.)

Оцінка ризиків.

Ризик: несанкціонований доступ до файлів з налаштуваннями.

Заходи: шифрування файлів, обмеження прав доступу до файлів на рівні операційної системи.

Ризик: перехоплення даних під час передачі між утилітою та сервером бази даних або Telegram.

Заходи: використання захищених протоколів (наприклад, TLS) для з'єднання з базою даних та Telegram API.

Ризик: помилки в коді утиліти, що призводять до неправильної обробки даних або витоку інформації.

Заходи: ретельне тестування коду, використання практик безпечного кодування.

2.2 Основні функціональні можливості

Утиліта здійснює збір даних про реплікацію шляхом виконання SQL-запитів до системних таблиць бази даних MS SQL Server. Для підключення до бази даних використовується бібліотека pyodbc, яка забезпечує зручний та ефективний доступ до даних.

Розроблена утиліта моніторингу реплікації баз даних має наступні

основні функціональні можливості:

Автоматичний моніторинг статусу реплікації на кількох серверах – утиліта забезпечує автоматичний збір інформації про статус реплікації на заданих серверах баз даних з заданою періодичністю. Це дозволяє адміністраторам постійно бути в курсі стану реплікації та оперативно реагувати на будь-які зміни або проблеми.

Логування отриманих даних для подальшого аналізу – утиліта аналізує отримані з бази даних повідомлення про помилки та виявляє нові помилки, що виникли протягом останнього періоду. Це дозволяє оперативно реагувати на проблеми та мінімізувати їх вплив на роботу системи. Для кожної реплікації перевіряються наявність помилок та затримки синхронізації. Якщо виявляються помилки або затримки, утиліта формує та надсилає відповідне повідомлення через Telegram-бот..

Сповіщення адміністраторів через Telegram-бот – утиліта забезпечує надсилання повідомлень адміністраторам про виявлені помилки, збої або інші критичні події через Telegram-бот. Це дозволяє адміністраторам оперативно отримувати інформацію про проблеми та вживати необхідних заходів для їхнього усунення.

Утиліта формує два типи звітів:

Звіти про помилки: містять детальну інформацію про виявлені помилки синхронізації, включаючи час виникнення, опис помилки та назви баз даних.

Звіти про затримки: містять інформацію про реплікації, для яких час останньої синхронізації перевищує заданий поріг (наприклад, 5 хвилин).

Звіти надсилаються через Telegram-бот

Гнучке налаштування інтервалів перевірки та критеріїв виявлення помилок – утиліта забезпечує гнучкі можливості налаштування інтервалів перевірки статусу реплікації, а також критеріїв виявлення помилок або збоїв. Це дозволяє адміністраторам адаптувати утиліту до конкретних потреб та особливостей інфраструктури.

Формат повідомлень – для надсилання сповіщень використовується

Telegram Bot API. Утиліта надсилає повідомлення у вказаний Telegram-чат у разі виявлення помилок синхронізації або значних затримок. Повідомлення містять стислу та інформативну інформацію про проблему, що дозволяє адміністраторам швидко зорієнтуватися та вжити необхідних заходів."

Приклад формату повідомлення про помилку (рис. 2.4).

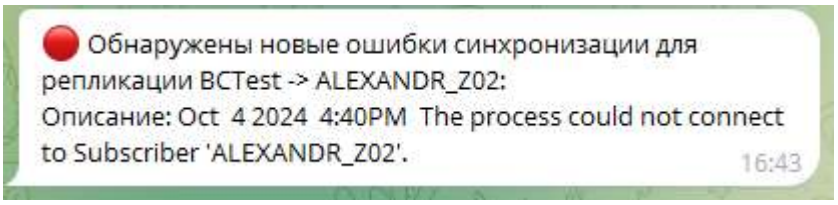


Рисунок 2.4 – Приклад повідомлення про помилку в реплікації

Приклад формату повідомлення про затримку (рис. 2.5).

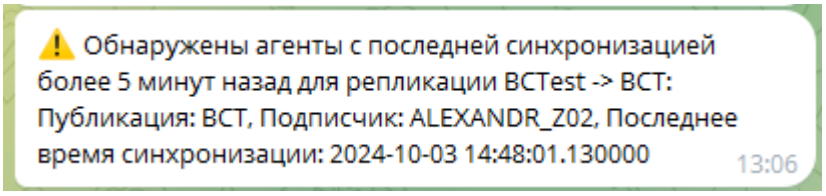


Рисунок 2.5 – Приклад повідомлення про перевищення часу синхронізації в реплікації

Приклад формату повідомлення про відсутність помилок (рис. 2.6).

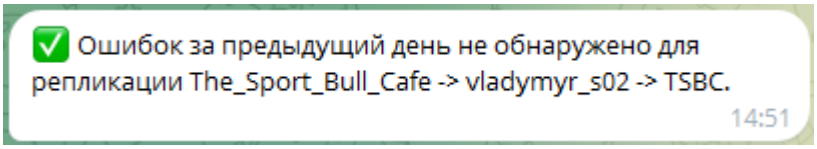


Рисунок 2.6 – Приклад повідомлення про відсутність помилок (добово) в реплікації

2.3 Взаємодія з базою даних

Для перевірки статусу реплікації утиліта виконує SQL-запити до системних таблиць сервера, таких як msdb.dbo.MSreplication_subscriptions та sys.dm_repl_traninfo (SQL Server System Views for Replication Monitoring, URL: https://learn.microsoft.com/en-us/sql/relational-databases/views/views?view=sql-server-ver16). Ці системні таблиці містять інформацію про налаштування реплікації, стан підписників, час останньої синхронізації, наявність помилок або затримок. Виконання SQL-запитів до цих таблиць дозволяє утиліті отримувати актуальну інформацію про стан реплікації та використовувати її для моніторингу та виявлення проблем.

Приклад SQL-запиту для отримання налаштувань реплікації:

“SELECT ms.publication_id, ma.publisher_db, ma.subscriber_db, ma.subscriber_name,

ss.active_start_time, ss.active_end_time, ss.freq_subday_interval

FROM [distribution].[dbo].[MSmerge_agents] ma

LEFT JOIN [distribution].[dbo].[MSmerge_subscriptions] ms on ms.publisher_db = ma.publisher_db

LEFT JOIN [msdb].[dbo].[sysjobs] sj on sj.name = ma.name

LEFT JOIN [msdb].[dbo].[sysjobschedules] sjs on sjs.job_id = sj.job_id

LEFT JOIN [msdb].[dbo].[sysschedules] ss on ss.schedule_id = sjs.schedule_id”

Цей запит отримує інформацію про ідентифікатор публікації, назви баз даних видавця та підписника, ім'я підписника, час початку та закінчення синхронізації, а також інтервал синхронізації.

Приклад SQL-запиту для отримання інформації про помилки синхронізації:

SELECT top(1) concat(max(time), ' ', error_text)

FROM distribution.dbo.MSrepl_errors

```
WHERE DATEDIFF(second, time, GETDATE()) <= 300
AND source_name like '%REPL%' and error_text like ‘%{subscriber}%’
GROUP BY time, error_text
```

Цей запит отримує текст останньої помилки, що сталася протягом останніх 300 секунд, для конкретного підписника.

2.4 Інтеграція з Telegram-ботом

Сповіщення адміністраторів про виявлені проблеми або критичні події реалізовано через Telegram API (Telegram Bot API, URL: https://core.telegram.org/bots/api). Використання Telegram-бота забезпечує зручний та оперативний спосіб сповіщення адміністраторів, дозволяючи їм отримувати інформацію про проблеми на мобільні пристрої або комп'ютери та оперативно реагувати на них.

Для інтеграції з Telegram-ботом утиліта використовує спеціальні бібліотеки, які забезпечують взаємодію з Telegram API. Адміністратори можуть налаштувати утиліту для надсилання повідомлень у конкретний Telegram-чат або групу, що дозволяє їм отримувати інформацію про проблеми в зручному для них форматі.

3 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Тестування утиліти

Для перевірки працездатності та надійності розробленої утиліти моніторингу реплікації баз даних було проведено комплексне тестування в різних умовах та сценаріях. Тестування включало перевірку основних функціональних можливостей утиліти, її продуктивності, стійкості до помилок та здатності коректно обробляти різні ситуації, що можуть виникнути в процесі реплікації.

Для проведення тестування було розгорнуто тестову інстанцію на окремому сервері. Тестова інстанція включала дві бази даних, налаштовані в режимі реплікації, та два підписники. Таке тестове середовище дозволило імітувати реальні умови роботи системи та провести тестування утиліти в умовах, максимально наближених до реальних.

Основні сценарії тестування включали:

Перевірку підключення до бази даних і коректність виконання SQL-запитів – на цьому етапі тестування перевірялася здатність утиліти встановлювати з'єднання з базами даних, налаштованими в режимі реплікації, та коректно виконувати SQL-запити до системних таблиць для отримання інформації про статус реплікації. Тестування включало перевірку коректності синтаксису SQL-запитів, а також перевірку обробки можливих помилок підключення або виконання запитів.

Імітацію помилок реплікації, таких як втрата зв’язку між Паблішером і Підписником – на цьому етапі тестування імітувалися різні помилки або збої, що можуть виникнути в процесі реплікації, такі як втрата мережевого зв'язку між сервером-видавцем (Publisher) та сервером-підписником (Subscriber), затримки передачі даних, конфлікти при одночасній зміні даних на різних серверах, а також інші помилки або збої. Тестування включало перевірку

здатності утиліти виявляти ці помилки, коректно їх обробляти та надсилати відповідні повідомлення адміністраторам через Telegram-бот.

Добовий моніторинг нормального стану реплікації, коли відсутні помилки – на цьому етапі тестування проводився безперервний моніторинг стану реплікації протягом доби в умовах нормальної роботи системи, коли відсутні будь-які помилки або збої. Тестування включало перевірку стабільності роботи утиліти, її здатності коректно збирати та обробляти інформацію про статус реплікації, а також перевірку відсутності помилкових спрацьовувань або інших проблем.

Аналіз реакції системи на виявлені збої та коректність надсилання повідомлень у Telegram – на цьому етапі тестування аналізувалася реакція утиліти на виявлені помилки або збої, а також перевірялася коректність надсилання повідомлень адміністраторам через Telegram-бот. Тестування включало перевірку змісту повідомлень, їхньої своєчасності, а також перевірку доставки повідомлень до заданого Telegram-чату або групи.

Приклади вхідних даних та очікуваних результатів:

Тестування підключення до БД:

Вхідні дані: коректні параметри підключення (сервер, ім'я користувача, пароль).

Очікуваний результат: успішне підключення до бази даних, виведення повідомлення "З'єднання з базою даних встановлено успішно".

Вхідні дані: некоректні параметри підключення (наприклад, невірний пароль).

Очікуваний результат: виникнення помилки підключення, виведення повідомлення "Не вдалося підключитися до бази даних: [повідомлення про помилку]", запит на повторне введення данних.

Тестування отримання даних про реплікацію:

Вхідні дані: відсутність налаштувань реплікації в базі даних.

Очікуваний результат: виведення повідомлення "Налаштування реплікації не знайдено".

Вхідні дані: наявність коректних налаштувань реплікації в базі даних.

Очікуваний результат: успішне отримання даних про реплікацію у вигляді списку об'єктів, кожен з яких містить інформацію про публікацію, підписника, час синхронізації тощо.

Тестування обробки помилок реплікації:

Вхідні дані: наявність записів про помилки реплікації за останній час.

Очікуваний результат: виявлення помилок та формування повідомлення про помилки для надсилання в Telegram.

Вхідні дані: відсутність записів про помилки реплікації за останній час.

Очікуваний результат: відсутність повідомлень про помилки (або надсилання повідомлення про відсутність помилок, якщо записано час щоденного відправлення повідомлення).

Тестування надсилання сповіщень:

Вхідні дані: коректний токен Telegram-бота та ID чату.

Очікуваний результат: успішне надсилання повідомлення в Telegram, виведення повідомлення "Повідомлення успішно відправлено!".

Вхідні дані: некоректний токен Telegram-бота.

Очікуваний результат: виникнення помилки при надсиланні повідомлення, виведення повідомлення "Помилка при відправленні повідомлення: [повідомлення про помилку від Telegram API]".

Тестування граничних умов:

Вхідні дані: велика кількість реплікацій (наприклад, 10).

Очікуваний результат: утиліта коректно обробляє всі реплікації, не зависає та не споживає надмірну кількість ресурсів.

Вхідні дані: відсутність зв'язку з сервером бази даних.

Очікуваний результат: утиліта коректно обробляє втрату з'єднання, намагається перепідключитися або виводить інформативне повідомлення про помилку.

3.2 Аналіз продуктивності

Під час роботи утиліти моніторингу реплікації баз даних вимірюються наступні ключові показники продуктивності:

Час виконання SQL-запитів до бази даних – цей показник характеризує швидкість отримання інформації про статус реплікації з бази даних. Час виконання SQL-запитів залежить від багатьох факторів, таких як складність запитів, розмір таблиць, навантаження на сервер бази даних, а також ефективність реалізації запитів в утиліті.

Час обробки результатів і відправлення повідомлень – цей показник характеризує швидкість обробки отриманої з бази даних інформації про статус реплікації, виявлення помилок або збоїв, а також формування та відправлення повідомлень адміністраторам через Telegram-бот. Час обробки результатів залежить від складності алгоритмів обробки даних, ефективності реалізації коду утиліти, а також швидкості роботи Telegram API.

Вплив утиліти на навантаження сервера – цей показник характеризує вплив роботи утиліти на навантаження сервера, на якому вона запущена. Робота утиліти може впливати на навантаження сервера за рахунок споживання ресурсів процесора, пам'яті, дискової підсистеми, а також мережевого трафіку.

3.3 Результати роботи

Зразки записів у лог-файл:

2024-12-25 13:09:10,141 - INFO - Запуск программы...

2024-12-25 13:09:26,320 - ERROR - Ошибка подключения к базе данных: ('08001', '[08001] [Microsoft][ODBC Driver 17 for SQL Server]Named Pipes Provider: Could not open a connection to SQL Server [53]. (53) (SQLDriverConnect); [08001] [Microsoft][ODBC Driver 17 for SQL Server]Login timeout expired (0); [08001] [Microsoft][ODBC Driver 17 for SQL Server]A network-related or instance-specific error has occurred while establishing a connection to SQL Server. Server is not found or not accessible. Check if instance name is correct and if SQL Server is configured to allow remote connections.

2025-04-22 13:49:20,388 - INFO - Запуск программы...
2025-04-22 13:50:19,362 - INFO - Успешно подключено к серверу MS SQL.
2025-04-22 13:50:19,381 - ERROR - Ошибка при получении настроек репликации: ('HY000', '[HY000] [Microsoft][ODBC Driver 17 for SQL Server]Connection is busy with results for another command (0) (SQLExecDirectW)')

2024-09-26 14:29:05,806 - INFO - Список репликаций и их подписчиков:
2024-09-26 14:29:05,807 - INFO - Репликация: NEWBEREZKA, База: NEWBEREZKA на подписчике DESKTOP-QMD8SL0
2024-10-07 18:33:48,686 - INFO - Файл с настройками подключения и Telegram конфигурацией найден.

Приклад формату повідомлення про помилку (рис. 3.1).

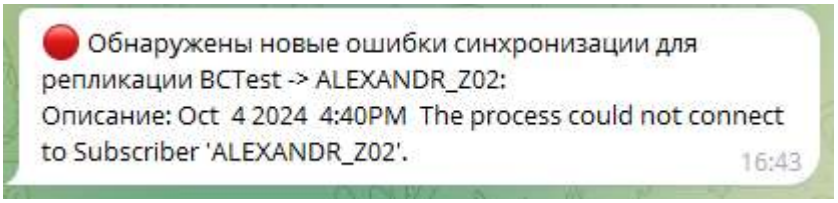


Рисунок 3.1– Приклад повідомлення про помилку в реплікації

Приклад формату повідомлення про затримку (рис. 3.2).

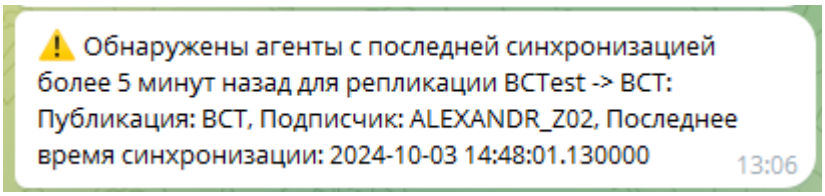


Рисунок 3.2 – Приклад повідомлення про перевищення часу синхронізації в реплікації

Приклад формату повідомлення про відсутність помилок (рис. 3.3).

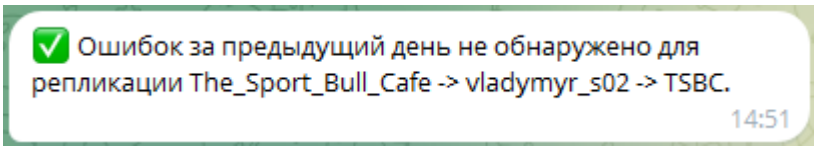


Рисунок 3.3 – Приклад повідомлення про відсутність помилок (добово) в реплікації

3.4 Можливі покращення

Подальший розвиток утиліти може бути спрямований на розширення її функціональних можливостей, підвищення ефективності та адаптацію до потреб різних середовищ. Можливі напрямки покращень включають:

Підтримка інших СУБД: Розширення утиліти для моніторингу реплікації не тільки MS SQL Server, але й інших популярних СУБД, таких як MySQL, PostgreSQL, Oracle. Це значно розширить область застосування утиліти та зробить її більш універсальною.

Розширення функціональності моніторингу: Додавання можливості моніторингу додаткових параметрів реплікації, таких як затримки реплікації, обсяг переданих даних, продуктивність реплікації. Це дозволить отримати більш повну картину стану реплікації та швидше виявляти потенційні проблеми.

Покращення інтерфейсу: Розробка графічного інтерфейсу користувача (GUI) для спрощення взаємодії з утилітою. Це зробить її більш зручною у використанні, особливо для користувачів, які не мають досвіду роботи з командним рядком.

Оптимізація продуктивності: Проведення оптимізації коду утиліти для підвищення її швидкодії та зменшення навантаження на сервер бази даних. Це особливо важливо при моніторингу великої кількості реплікацій.

Інтелектуальний аналіз даних: Впровадження елементів машинного навчання для аналізу історичних даних моніторингу та прогнозування можливих збоїв у реплікації. Це дозволить адміністраторам вживати профілактичних заходів та запобігати виникненню критичних ситуацій.

Розширені можливості сповіщень: Розробка більш гнучкої системи сповіщень, яка дозволить налаштовувати різні рівні критичності сповіщень, способи сповіщень (наприклад, електронна пошта, SMS) та правила фільтрації сповіщень.

ВИСНОВКИ

В ході виконання даної кваліфікаційної роботи було розроблено спеціалізовану утиліту для моніторингу статусу реплікації баз даних, що дозволяє автоматизувати контроль за синхронізацією даних між серверами та оперативно повідомляти адміністраторів про можливі збої.

Сучасні інформаційні системи потребують ефективних механізмів контролю за станом баз даних, особливо у випадках, коли використовується реплікація для забезпечення відмовостійкості та розподілу навантаження. Реплікація є ключовою технологією, що забезпечує високу доступність, надійність та продуктивність інформаційних систем, дозволяючи розподіляти дані між декількома серверами та забезпечувати їхню узгодженість.

У процесі дослідження були проаналізовані доступні на ринку рішення для моніторингу реплікації баз даних, серед яких Zabbix, Datadog, Percona Monitoring and Management та інші. Було виявлено, що жоден з розглянутих інструментів не повністю відповідає вимогам проекту через складність налаштування, відсутність гнучкості, обмежені можливості автоматичного виявлення помилок або необхідність використання сторонніх сервісів.

Розроблена утиліта моніторингу реплікації баз даних має наступні ключові можливості:

Автоматичний збір інформації про статус реплікації баз даних на заданих серверах з заданою періодичністю, що забезпечує постійний контроль за станом реплікації.

Виявлення збоїв у синхронізації та їх оперативне логування, що дозволяє адміністраторам аналізувати проблеми та вживати заходів для їхнього усунення.

Надсилання повідомлень адміністраторам про виявлені помилки або критичні події через Telegram-бот, що забезпечує оперативне сповіщення та швидке реагування на проблеми.

Гнучке налаштування періодичності перевірок та критеріїв виявлення помилок, що дозволяє адаптувати утиліту до конкретних потреб та особливостей інфраструктури.

Для перевірки працездатності та надійності утиліти було розгорнуто тестове середовище з двома базами даних у реплікації та двома підписниками. Тестування включало моделювання різних сценаріїв, таких як втрата зв’язку між сервером-джерелом та підписниками, і показало, що утиліта коректно ідентифікує збої та надсилає відповідні повідомлення адміністраторам через Telegram-бот. Також був проведений добовий моніторинг без збоїв, що підтвердило стабільність роботи утиліти у нормальних умовах.

Розроблена утиліта є гнучкою та масштабованою, що дозволяє легко адаптувати її до нових вимог та розширювати її функціональні можливості. Подальший розвиток утиліти може включати:

Оптимізацію продуктивності SQL-запитів для ще швидшої перевірки реплікації та зменшення навантаження на сервер бази даних.

Розширення підтримуваних типів реплікації та інтеграцію з додатковими СУБД, що дозволить розширити область застосування утиліти та зробити її більш універсальною.

Впровадження машинного навчання для аналізу історичних даних та прогнозування можливих збоїв у реплікації, що дозволить адміністраторам вживати профілактичних заходів та запобігати виникненню критичних ситуацій.

Таким чином, результати роботи підтверджують актуальність та ефективність створеного рішення. Розроблена утиліта забезпечує централізований контроль за реплікацією, мінімізує ризики втрати даних та знижує навантаження на адміністраторів, що робить її важливим інструментом для управління сучасними інформаційними системами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th Edition. McGraw-Hill, 2019. С. 750–780.

2. Microsoft Docs – Replication in SQL Server. URL : https://learn.microsoft.com/en-us/sql/relational-databases/replication/sql-server-replication?view=sql-server-ver16 (дата звернення : 20.01.2024).

3. Programmatically Monitor Replication. Microsoft Docs. URL : https://learn.microsoft.com/en-us/sql/relational-databases/replication/monitor/programmatically-monitor-replication?view=sql-server-ver16 (дата звернення : 20.01.2025).

4. Percona Monitoring and Management Documentation. URL : https://docs.percona.com/percona-monitoring-and-management//3/ (дата звернення : 10.01.2025).

5. Datadog Database Monitoring. URL : https://www.datadoghq.com/product/database-monitoring/ (дата звернення : 10.01.2025).

6. Zabbix Documentation – Database Monitoring. URL : https://www.zabbix.com/documentation/current/en/manual/config/items/itemtypes/odbc_checks (дата звернення : 10.01.2025).

7. Telegram Bot API. URL : https://core.telegram.org/bots/api (дата звернення : 07.02.2025).

8. SQL Server System Views for Replication Monitoring. Microsoft Docs. URL : https://learn.microsoft.com/en-us/sql/relational-databases/views/views?view=sql-server-ver16 (дата звернення : 25.01.2025).

ДОДАТОК А

Приклади коду

```
import json
import os
import time
import pyodbc
import requests
import logging
import threading
from datetime import datetime, timedelta
from threading import Thread
from cryptography.fernet import Fernet
logging.basicConfig(filename='monitoring.log', level=logging.INFO,
format='%(asctime)s - %(levelname)s - %(message)s', encoding='utf-8')


# Основная функция программы
def main():
    logging.info("Запуск программы...")
    encryption_manager = EncryptionManager()
    config_manager = ConfigManager(encryption_manager)
    config_data = config_manager.check_settings()
    while True:
        if config_data:
            connection_data = config_data["connection_data"]
            telegram_config = config_data["telegram_config"]
            daily_report_time = config_data["daily_report_time"]
        else:
            connection_data = config_manager.get_connection_data()
            telegram_config = config_manager.get_telegram_config()
            daily_report_time = config_manager.get_daily_report_time()
            config_manager.save_config(connection_data, telegram_config, daily_report_time)
        db_manager = DatabaseManager(connection_data)
        try:
            # Тестируем соединение
            conn = db_manager.connect()
            if conn:
                print("Соединение с базой данных установлено успешно.")
            break  # Выходим из цикла, если подключение прошло успешно
        except Exception as e:
            print(f"Не удалось подключиться к базе данных: {e}")
            print("Пожалуйста, проверьте введённые данные и попробуйте снова.")
            config_data = None  # Сбрасываем данные, чтобы запросить их снова
    os.system('cls' if os.name == 'nt' else 'clear')
   notification_service = NotificationService(telegram_config["token"], telegram_config["chat_id"])
        monitor = ReplicationMonitor(db_manager, notification_service, daily_report_time)
```

```
    try:
        monitor.start()
        monitoring_thread, report_thread = monitor.start()
        monitoring_thread.join()
        report_thread.join()
    except KeyboardInterrupt:
        monitor.stop()
if __name__ == "__main__":
    main()

# Класс для управления базой данных
class DatabaseManager:
    def __init__(self, connection_data):
        self.connection_data = connection_data
        self.conn = None
    def connect(self):
        """Подключение к базе данных MS SQL"""
        try:
            self.conn = pyodbc.connect(
                f"DRIVER=ODBC Driver 17 for SQL Server;"
                f"SERVER={self.connection_data['server']};"
                f"UID={self.connection_data['username']};"
                f"PWD={self.connection_data['password']}"
            )
            logging.info("Успешно подключено к серверу MS SQL.")
        except Exception as e:
            logging.error(f"Ошибка подключения к базе данных: {e}")
            self.conn = None
        return self.conn

# Класс для отправки уведомлений
class NotificationService:
    def __init__(self, token, chat_id):
        self.token = token
        self.chat_id = chat_id
    def send_message(self, text):
        """Отправка сообщения в Telegram"""
        url = f"https://api.telegram.org/bot{self.token}/sendMessage"
        params = {"chat_id": self.chat_id, "text": text}
        response = requests.post(url, json=params)
        if response.status_code == 200:
            logging.info("Сообщение успешно отправлено!")
        else:
            logging.error(f"Ошибка при отправке сообщения: {response.text}")
    def send_notifications(self, errors, stale_records, replication_config):
        publication = replication_config["publisher_db"]
        subscriber_db = replication_config["subscriber_db"]
        subscriber = replication_config["subscriber"]
        if errors:
            error_message = f"● Обнаружены новые ошибки синхронизации для репликации
{publication} -> {subscriber}:\n"
            for error in errors:
```

```
            error_message += f"Описание: {error}\n"
        self.send_message(error_message)
    if stale_records:
        stale_message = f"⚠ Обнаружены агенты с последней синхронизацией более 5 минут назад для репликации {publication} -> {subscriber} -> {subscriber_db}:\n"
        for record in stale_records:
            stale_message += (f"Публикация: {record['publication']}, "
            f"Подписчик: {record['subscriber_name']}, "
f"Последнее время синхронизации: {record['last_sync_time']}\n")
        self.send_message(stale_message)
  def send_no_errors_notification(self, replication_config):
    publication = replication_config["publisher_db"]
    subscriber_db = replication_config["subscriber_db"]
    subscriber = replication_config["subscriber"]
        message = f"✓ Ошибок за предыдущий день не обнаружено для репликации {publication} -> {subscriber} -> {subscriber_db}."
    self.send_message(message)
```