Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: Сервіс для організації операцій над часовими рядами для SCADA-системи

Виконала: студентка 6 курсу, групи СНмз-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

		Яремцьо І.І.
	(підпис)	(прізвище та ініціали)
Керівник		доц. Баран І.О.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		доц. Мацюк О.В.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		доц. Боднарчук І.О.
	(підпис)	(прізвище та ініціали)
Рецензент		доц. Гладьо Ю.Б.
	(підпис)	(прізвище та ініціали)

м. Тернопіль – 2020

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Боднарчук І.О.
(підпис) (прізвище та ініціали)
« » 20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

студентці Яремцьо Ірині Іванівні

1. Тема роботи Сервіс для організації операцій над часовими рядами для SCADA-системи

Керівник роботи Баран Ігор Олегович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом по університету від «__06__» _листопада_ 2020 року №__4/7-829_

2. Термін подання студентом роботи 25.12.2020

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1 Аналітичний огляд використовуваних технологій. 2 Теоретико-практичне дослідження. 3. Реалізація сервісу. 4. Охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема роботи. 2. Актуальність. 3. Мета, задачі, об'єкт, предмет дослідження. 4. Стандарт OPC. 5. Переваги / недоліки застосування стандарту OPC. 6. Протокол OPC UA. 7. Вимоги до сховища даних. 8. Аналіз основних спеціалізованих БД та сховищ даних для часових рядів. 9. Тестування сховищ даних. 10. Архітектура сервера історії. 11 Структура сервера історії. 12. Діаграма класів розробленого сервісу. 13. Реалізація Модуля підтримки операцій. 14. Програмні коди функцій Модуля підтримки операцій. 15. Діаграма взаємодії для операції readRaw ().16. Скрін-шоти вікон сервісу. 17.Основні результати дослідження

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Дмитроца Л. П., доцент	01.11.20	07.12.20
Безпека в НС	Стадник І. Я., професор	01.11.20	09.12.20

7. Дата видачі завдання 06 листопада 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми кваліфікаційної роботи	06.11.20	Виконано
2	Аналіз літературних джерел	07.11.20-18.11.20	Виконано
3	Обґрунтування актуальності дослідження	18.11-21.11.20	Виконано
4	Аналіз предмету дослідження та предметної області	21.11-26.11.20	Виконано
5	Проведення дослідження методів та засобів аналітичного опрацювання даних	22.11-30.11.20	Виконано
6	Оформлення розділу «Аналітичний огляд використовуваних технологій»	20.11-26.11.20	Виконано
7	Оформлення розділу «Теоретико-практичне дослідження»	27.11-02.12.20	Виконано
8	Оформлення розділу «Реалізація сервісу»	03.12-10.12.20	Виконано
9	Оформлення розділу «Охорона праці та безпека в надзвичайних ситуаціях»	26.11-12.12.20	Виконано
10	Нормоконтроль	11.12-15.12.20	Виконано
11	Попередній захист роботи	18.12.20	Виконано
12	Захист кваліфікаційної роботи	26.12.20	

Студент _____ Яремцьо І.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Баран І. О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сервіс для організації операцій над часовими рядами для SCADA-системи // Кваліфікаційна робота освітнього рівня «Магістр» // Яремцьо Ірина Іванівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра комп'ютерних наук, група СНмз–61 // Тернопіль, 2020 // С. – 67, рис. – 21 , табл.– 8, слайдів – 17, додат. – 1, бібліогр. – 32.

Ключові слова: OPC UA, SCADA, БАЗИ ДАНИХ, СЕРВЕР ІСТОРИЧНИХ ДАНИХ, ЧАСОВІ РЯДИ

Кваліфікаційна робота присвячена проектуванню та створенню програмного сервісу для реалізації операцій над часовими рядами для SCADA-системи.

Для цього виконано огляд стандарту OPC як основного способу взаємодії SCADA-системи з зовнішнім світом, проаналізовано специфікацію OPC UA. Зроблено докладний опис стратегії побудови системи на базі OPC UA. Також наведено можливості, властиві SCADA-системам, та етапи проектування SCADA-систем. Порівняно основні спеціалізовані БД для часових рядів та сховищ «ключ-значення» або вбудовуваних СУБД. Проаналізовано їх основні функціональні можливості та параметри, а також протестовано продуктивність операції запису. Виходячи з них, вибір був зроблений на користь LevelDB.

Також докладно описано структуру сигналу відповідно до специфікації OPC UA. Наведено схему взаємодії сервера історії з компонентами SCADA-системи та архітектуру серверу історії. Розроблено програмну архітектуру сервісу, наведені особливості реалізації. Відображені результати опрацювання часових рядів даних.

ANNOTATION

Service for time serieses operations arrangement for SCADA-system // Master thesis // Yaremtso Iryna // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science // Ternopil, 2020 // P. - 67, Fig. - 21, Table – 8, Slide - 17, References - 32.

Keywords: OPC UA, SCADA, DATA BASES, HISTORY SERVER, TIME SERIES

Thesis deals with the design and creation of a software service for the implementation of time series operations for the SCADA-system.

To do this, an overview of the OPC standard as the main way of interaction of the SCADA-system with the outside world, analyzed the specification of OPC UA. A detailed description of the strategy for building a system based on OPC UA are made The possibilities inherent in SCADA-systems and stages of designing SCADA-systems are also given. The main specialized databases for time series and key-value repositories or embedded DBMSs are compared. Their main functionalities and parameters are analyzed, as well as the performance of the recording operation is tested. The choice was made in favor of LevelDB.

The signal structure according to the OPC UA specification is also described in details. The scheme of interaction of the history server with the components of the SCADA-system and the architecture of the history server is given. The software architecture of the service is developed, features of realization are resulted. The results of time series data processing are displayed.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

ACID (Atomicity, Consistency, Isolation, Durability) – це набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність

API (Application Programming Interface) – інтерфейс прикладного програмування (набір готових класів, процедур, функцій, структур і констант, які надаються додатком (бібліотекою, сервісом) для використання в зовнішніх програмних продуктах).

DDE (Dynamic Data Exchange) – динамічний обмін даними.

OPC (Open Platform Communications) – набір стандартів і специфікацій, який використовується в промислових засобах зв'язку.

OPC UA (OPC Unified Architecture) – специфікація, що визначає передачу даних в промислових мережах і взаємодію пристроїв в них.

SOA (Service Oriented Architecture) – архітектура, орієнтована на сервіси.

SCADA (Supervisory Control And Data Acquisition) – диспетчерське управління і збір даних (програмний пакет, призначений для забезпечення роботи в реальному часі систем збору, обробки, відображення і архівування інформації про об'єкт моніторингу або управління).

АСУ ТП – автоматизована система управління технологічним процесом.

БД – база даних.

ОС – операційна система

ПЗ – програмне забезпечення.

Сервер історії – програмний компонент, призначений для збору і зберігання історії технологічного процесу і забезпечення доступу клієнтських додатків до архіву історичної інформації, необхідної для аналізу подій, що відбулися і аварій в технологічних та виробничих процесах підприємств.

Сигнал – елемент тимчасового ряду.

СУБД – система управління базами даних

ТП – технологічний процес.

ФС – файлова система.

ЗМІСТ

ВСТУП

Актуальність теми. В даний час без застосування АСУ ТП не обходиться жодна галузь виробництва. Обумовлено це, перш за все, ускладненням виробничих і технологічних процесів і вимогою до підвищення продуктивності праці. Впровадження такого роду систем так само тягне за собою зменшення трудомісткості ТП, оптимізацію якісних і кількісних показників ТП, економію ресурсів, зменшення впливу людського фактора [1].

Одним з видів АСУ ТП є SCADA-системи, які широко застосовуються в енергетиці, металургії, водопостачанні, нафтогазовій, харчовій промисловості і т.д. Вони забезпечують безперервний контроль і управління технологічним обладнанням, інформування про події та аварії, зберігання історії технологічного процесу, обмін даними між вузлами розподіленого виробництва і т.д. Однією з ключових вимог до SCADA-системи є зберігання даних історії технологічних процесів. Такого роду дані необхідні в аналітичних цілях, зокрема, для: аналізу даних за заданий проміжок часу; аналізу ланцюжка подій; складання звітів. Окрім усього іншого, такі дані бувають корисні при розгляді різних позаштатних ситуацій. Використовуючи їх, можна проводити аналіз ТП (наприклад, шляхом візуалізації повторного відтворення процесу). Крім того, при застосуванні методів математичної статистики, на основі історичних даних можна прогнозувати хід розвитку ТП [2].

Проблема в тому, що більшість існуючих SCADA-систем розроблені під OS Windows, яка є пропрієтарним ПЗ. Упродовж останнього часу з'явилася задача створити якісний високотехнологічний програмно-інструментальний комплекс для реалізації АСУ ТП на базі вільного ПЗ. Тому розробка базових модулів сервера історії такої системи є надзвичайно актуальною.

Мета дослідження: реалізувати OPC UA-операції над часовими рядами технологічних процесів в СУБД для SCADA-системи.

Для досягнення вказаної мети, в роботі поставлено та розв’язано наступні **задачі**:

– провести аналіз різних сховищ даних і вибрати те, яке найкраще задовольняє необхідні вимоги;

– реалізувати «надбудову» над сховищем з функцією збереження і зміни історичних технологічних даних, які надходять від різних джерел;

– створити API модуль підтримки операцій в у відповідності до специфікації OPC UA;

– реалізувати функції доступу до історичних технологічних даних за допомогою розробленого ПЗ.

Об’єкт дослідження: система зберігання історичних технологічних даних.

Предмет дослідження: механізми організації операцій над часовими рядами.

Наукова новизна роботи:

– протестовано продуктивність операції запису основних спеціалізованих БД та сховищ даних для часових рядів;

– на базі специфікації OPC UA розроблений API- механізм зберігання даних з підтримкою OPC UA-операцій;

– програмно розроблено сервіс для роботи з часовими рядами (механізм зберігання даних, приймач даних; модуль підтримки операцій; колектор серверу історій).

Практичне значення одержаних результатів. Впровадження розробки дозволить аналізувати ТП та прогнозувати його хід його розвитку.

Апробація. Частково результати роботи були представлені на VIII науково-технічній конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя (9-10 грудня 2020 р.) у вигляді опублікованих тез [21].

1 АНАЛІТИЧНИЙ ОГЛЯД ВИКОРИСТОВУВАНИХ ТЕХНОЛОГІЙ

1.1 Стандарт OPC

Властиво основною ідеєю запровадження стандарту OPC було забезпечення єдиного визначеного інтерфейсу між АСУ ТП, які працюють з різноманітними апаратними засобами (платформами) та у різноманітних промислових мережах і випускаються відмінними вендорами.

До випуску цього стандарту кожна АСУ ТП розроблялася під конкретне обладнання, і в разі його модернізації або заміни необхідно було проводити адаптацію драйверів, оскільки кожен з них забезпечував протокол обміну даними тільки з одним клієнтом. Ступінь зв'язаності була високою, а можливість повторного використання - вкрай низькою

Але після переходу на стандарт OPC ситуація змінилася в кращу сторону: існуючі SCADA-системи були переорієнтовані на підтримку OPC та фактично стали OPC-клієнтами, а виробники обладнання додали підтримку OPC своїх пристроїв, кожен з яких ставав OPC сервером [2].

Тепер обладнання та SCADA-системи взаємодіють через єдиний уніфікований інтерфейс. Він дає змогу під’єднювати який-небудь датчик, контролер або апаратний модуль до будь-якої SCADA-системи, якщо технічні характеристики цих пристроїв дають змогу підтримувати згаданий стандарт.

Необхідно розглянути основні принципи взаємодії SCADA-системи з зовнішнім середовищем.

На сучасному етапі свого розвитку SCADA-системи не лімітують вибір апаратури на нижньому рівні, так як володіють великим набором програмних засобів чи серверів для виконання операцій вводу-виводу та використовують потужні засоби для створення власного ПЗ для драйверів. В вже самі ці драйвери створюються на стандартних мовах програмування.

В даний час можна застосувати такі механізми для якісного під'єднання драйверів до системи:

– DDE, що став фактично стандартом. Однак в сучасних SCADA-системах DDE застосовується рідко;

– власні протоколи, які розроблені фірмами виробниками SCADA-систем. Перевагою таких протоколів є найвища швидкість обміну даними;

– сам протокол OPC. Він є стандартним і більшість SCADA- систем забезпечують його підтримку. Він був створений на базі механізму OLE.

Необхідно навести наступні переваги OPC [3]:

– поєднує на об'єктному рівні різноманітні системи для управління та контролю, які працюють в розподіленому середовищі, в якому взаємодіють різноманітні апаратні та програмні платформи;

– не потрібно використовувати нестандартні протоколи обміну даними для взаємодії між пристроєм та SCADA-системою.

Таким чином, основною метою стандарту OPC є створення і забезпечення функціонування універсального механізму доступу до будь-якого апаратного пристрою з прикладної програми. Згаданий стандарт дає змогу виробникам обладнання постачати компоненти ПЗ, які визначеним способом забезпечують зв'язок ПЗ з технологічним контролером на нижчому рівні.

OPC - інтерфейс допускає використовувати такі варіанти обміну даними:

– отримання даних з фізичних пристроїв;

– між частинами розподіленого додатка;

– між різними додатками.

В першу чергу в якості серверів OPC виступають драйвери, написані відповідно до стандарту OPC і які здійснюють обмін даними з компонентами систем автоматизованого управління через відповідне комунікаційне обладнання. Крім того, будь-яка програма, забезпечена стандартним OPC-інтерфейсом, може виступати як OPC-сервер.

Стосовно до SCADA-систем, OPC-сервери, розташовані на всіх комп'ютерах системи управління, стандартним чином можуть постачати дані в програму візуалізації, БД і т.д.

При обміні даними з OPC - сервером можливо два режими:

– періодичний режим, коли із заданою частотою дані запитуються OPC - клієнтом;

– режим обміну по зміні значення, коли обмін відбувається при зміні значення змінної на заздалегідь задану величину.

Кращим є другий тип обміну. При обміні через OPC-інтерфейс дані передаються у вигляді особливих структур, званих пакетами, які містять такі поля:

– Value (значення);

– Quality (якість);

– Timestamp (відмітка часу).

Такий пакет в термінології OPC називається «елемент даних». Поле Quality дозволяє визначити, чи не сталася помилка в момент вимірювання величини або під час передачі даних.

У всіх сучасних SCADA-системах при обміні даними здійснюється перевірка поля Quality. Причому в різних системах реакція на «незадовільне» значення якості одержуваних даних може бути реалізована по-різному. Обробляти поле Quality може або додаток користувача, або сама SCADA-система.

Поле Quality може набувати різних значень: - UNCERTAIN (НЕ визначено), GOOD (задовільно), BAD (незадовільно). Якщо поле Quality приймає значення BAD, в цьому полі міститься додаткова ознака, що дозволяє уточнити причину неполадки.

В рамках стандарту OPC всі елементи даних об'єднуються в групи. Кожен елемент даних і група мають своє унікальне ім'я. Елементи даних і групи можуть бути організовані ієрархічно. Усі елементи в кожній групі оновлюються

періодично, через однакові часові проміжки, причому оновлення елементів відбувається синхронно [4].

Елементи даних часто називають тегами (TAG). Саме ці теги і є технологічними змінними в SCADA-системі.

OPC-сервер повинен здійснювати буферизацію даних, запитуваних різними клієнтськими додатками, і оптимізувати їх передачу так, щоб комунікація з фізичними пристроями була найбільш ефективною. Буферизація даних необхідна для того, щоб виключити їх втрату, і щоб була можливість їх багаторазового зчитування.

Важливою перевагою OPC є можливість перетворення системи управління в свого роду «конструктор», різнотипні елементи якого можуть бути підключені до системи стандартним чином - через OPC-інтерфейс.

Використання технології OPC в якості процедури забезпечення цілісного доступу до виробничих даними дає наступні переваги:

– виробники пристроїв мають можливість створення універсальних «перехідників» від свого пристрою до стандартизованого інтерфейсу;

– виробники ПЗ SCADA можуть орієнтувати свої програмні продукти на роботу зі стандартним інтерфейсом, що не залежать від типу пристрою;

– споживачі (замовники) комплектують свої системи такими пристроями та ПЗ, яке найбільш підходить для вирішення поставлених завдань.

Технологічні пристрої представляються керуючому ПО у вигляді серверів OPC і в загальному випадку є «чорними ящиками».

Звичайно, не буває засобу, який вирішить відразу всі проблеми. OPC може використовуватися тільки в тих ОС, де підтримується механізм Microsoft DCOM.

OPC не забезпечує роботу в жорсткому режимі реального часу, оскільки в DCOM відсутні поняття якості обслуговування, крайніх термінів і т.д. Водночас контроль за «старінням» даних є - кожне передане значення супроводжується міткою часу. Незважаючи на те, що вимоги жорсткого реального часу, строго

кажучи, не виконуються, реальний час передачі даних близько 50 мілісекунд досягається без всяких спеціальних заходів.

Не потрібно думати, що будь-який пристрій можна просто так «через OPC» підключити до будь-якої SCADA-системи - для цього треба мати OPC-сервер для даного пристрою. Сервер можна отримати або разом з пристроєм, або купити, або написати самостійно. Для написання OPC-серверів в складі деяких SCADA постачається спеціальне ПЗ.

При розробці прикладних програм OPC сервер представляється в вигляді DCOM-об'єкта, який інкапсулює всю складність поточного взаємозв’язку з апаратурою при «спілкуванні» з нею і надає користувачеві зручний та простий інтерфейс для забезпечення доступу до використовуваної апаратури [2].

DCOM (Distributed Component Object Model) є програмною архітектурою, розробленою компанією Microsoft для розподілу додатків між декількома комп'ютерами в мережі. Програмний компонент на одній з машин може використовувати DCOM для передачі повідомлення (його називають віддаленим викликом процедури) до компоненту на іншій машині. DCOM автоматично встановлює з'єднання, передає повідомлення і повертає відповідь віддаленого компонента.

1.2 Протокол OPC UA

Однак, незважаючи на очевидні переваги, технологія OPC має ряд недоліків [4, 5]:

– дана технологія доступна тільки на ОС Windows;

– DCOM має закриті sourse-коди, що зменшує надійність програмних компонентів, а також ускладнює виявлення і усунення виникаючих програмних відмов;

– виникають проблеми з налаштуванням та використанням конфігурації, пов'язані напряму з DCOM;

– сповіщення DCOM щодо переривання зв'язку бувають неточними;

– DCOM не пристосований для здійснення обміну поточними даними з використанням мережі Internet;

– DCOM має проблеми з контролем за безпекою інформації.

З врахуванням тих особливостей, які наведені вище, OPC Foundation було запропоновано новий уніфікований протокол (чи специфікацію) для якісного обміну поточними даними в АСУ ТП – OPC UA. Цей стандарт визначає способи обміну інформаційними повідомленнями, які проходять між OPC UA сервером і OPC UA клієнтом, що не залежать від апаратного забезпечення та ПЗ. Також наявна незалежність від типу систем і мереж, які взаємодіють між собою. OPC UA гарантує стабільне та безпечне з'єднання, захист від вірусів. Також гарантує, що інформації клієнта і сервера є ідентичною.

У розглядуваному стандарті є одним з ключових поняття об'єкта (схоже з поняттям об'єкта в ООП). Об'єкт - це екземпляр класу, а клас - це абстрактний тип даних. Об'єкт інкапсулює стан (змінні) і поведінку (методи і події). Практично, під об'єктом розуміється деякий фізичний або абстрактний елемент системи. Об'єктами можуть бути апаратні пристрої, системи і підсистеми. Давачі температури, наприклад, можна уявити об'єктом, що вмикає значення температури, набір параметрів сигналізацій і межі їх спрацьовування.

OPC UA має високу робастність даних і повідомлень про події. Це забезпечується механізмом швидкого виявлення помилок комунікації та відновлення даних. Під робастністю мається на увазі - нечутливість даних до перешкод, збоїв, обчислювальним помилок.

OPC UA базується на SOA. Це забезпечує сумісність програмних і апаратних засобів та платформ від різноманітних вендорів. Сервіс у поняттях в OPC UA - це якась програмна функціональність, котру можна використати безпосередньо на клієнті, сервері чи транспортувати від сервера клієнту (або навпаки) і викликати віддалено. Процедура виклику сервісу - те ж саме, що і

процедура виклику методу в ООП. Набір сервісів - фактично інтерфейс між клієнтом та сервером [10].

1.3 Стратегія побудови системи на основі OPC UA

Властиво система на основі OPC UA може складатися із безкінечної множини клієнтів і серверів, причому кожен клієнт має змогу одночасно працювати з різними серверами, відповідно кожен сервер, в свою чергу, має змогу обслуговувати кілька клієнтів [4]. Користувацький додаток має змогу поєднувати комбінації груп клієнтів-серверів з метою забезпечення надсилання повідомлень, за допомогою яких здійснюється обмін з іншими клієнтами-серверами, як це наведено на рисунку 1.1.

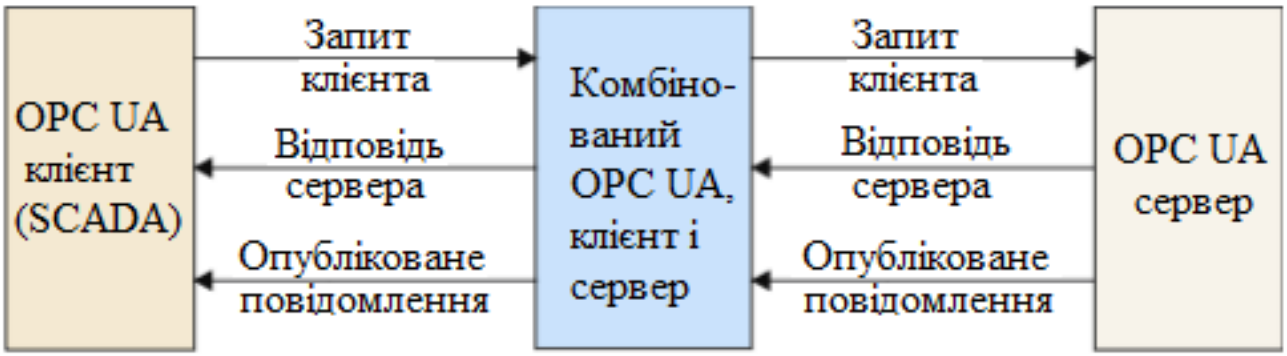

Рисунок 1.1 - Система на основі OPC UA

Прикладна програма при взаємодії з OPC сервером і є клієнтом, наприклад, SCADA. Структура клієнта показана на рисунку 1.2. Клієнтське ПЗ здійснює запити сервісів OPC сервера через внутрішній інтерфейс, який є ізолюючим прошарком між програмним і комунікаційним стеками. Комунікаційний стек конвертує запити клієнтської прикладної програми в повідомлення для виклику необхідного сервісу, які посилає серверу. Після отримання відповіді на запити комунікаційний стек передає їх в клієнтську програму.

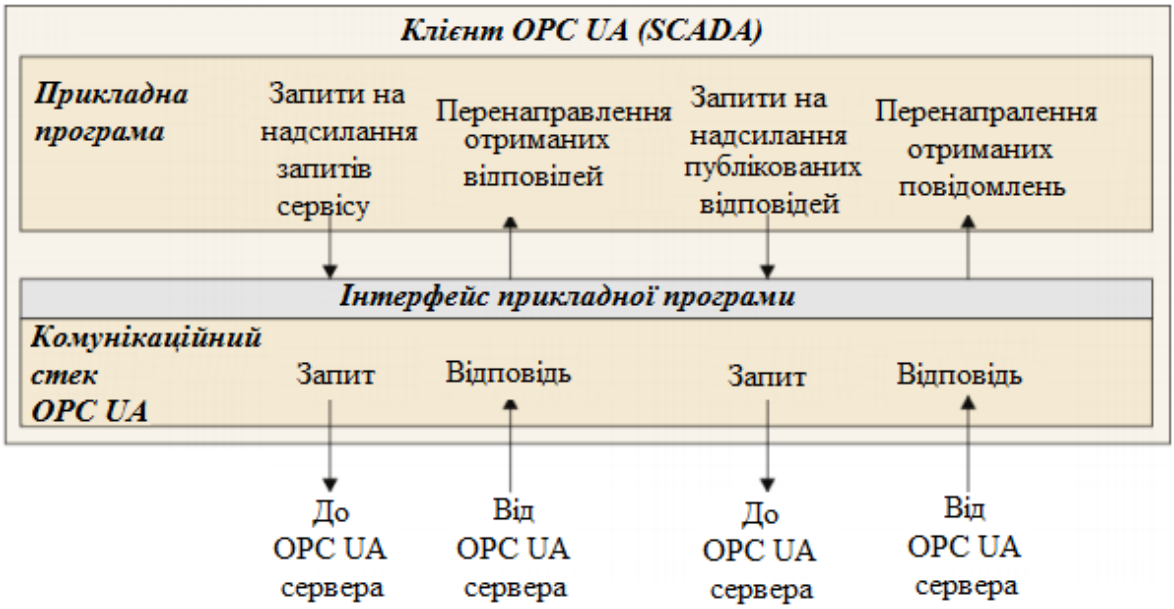

Рисунок 1.2 – Структура клієнтської частини OPC UA

Структура серверної частини OPC UA показана на рисунку 1.3.

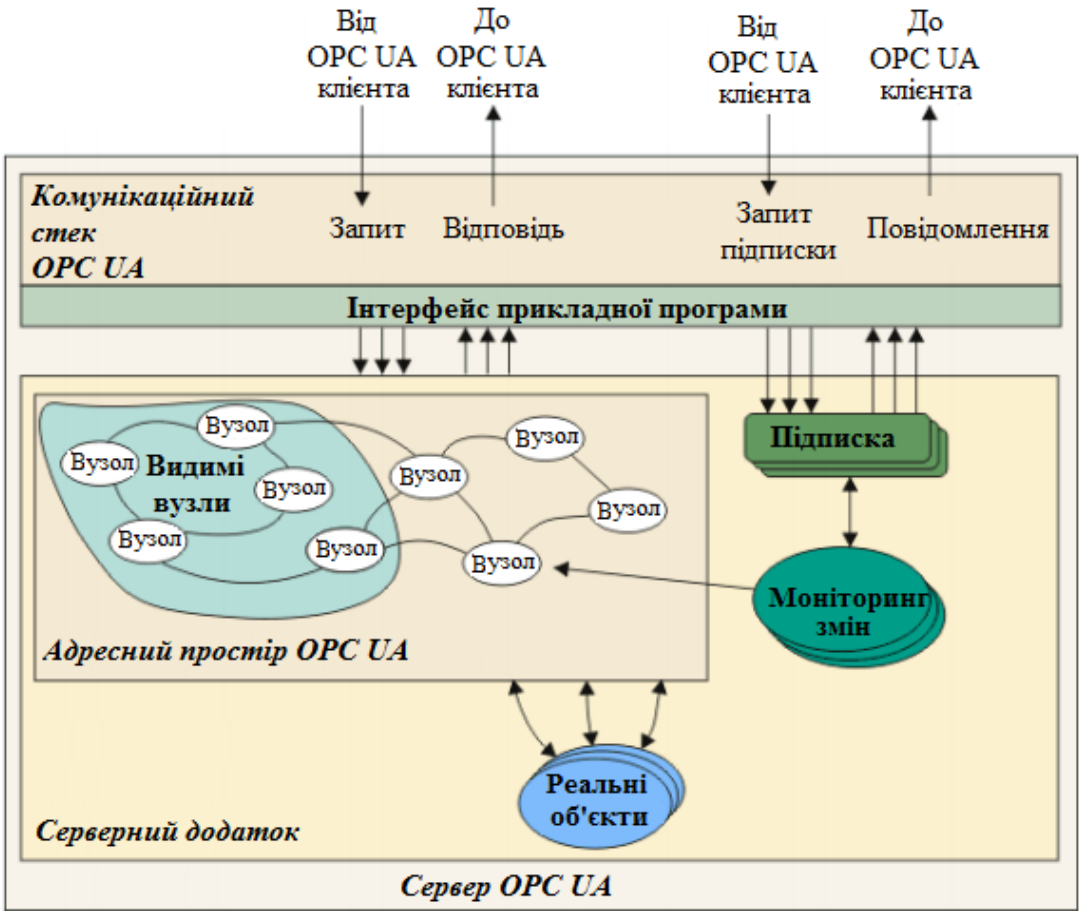

Рисунок 1.3 – Структура серверної частини OPC UA

Тут «реальні об'єкти» - складові частини вводу/виводу, програмно-логічні компоненти, спеціалізовані пристрої і програми, котрі мають змогу постачати дані безпосередньо через OPC сервер.

Програмний додаток для сервера є реалізацію тих функцій, які повинні бути виконаними.

Інтерфейс прикладної програми призначений для здійснення взаємодії OPC UA-сервера та клієнта за допомогою надсилання запитів та одержання на них відповідей.

Простір адрес OPC-сервера являє собою нескінченну множину вузлів, які доступні програмі клієнта за використання сервісів OPC UA. Такі «вузли» в цьому просторі адрес застосовуються, для представлення реальних об'єктів. В просторі адрес особливе місце займає підпростір тих вузлів, які сервер буде робити для клієнта програмно «видимими». Вже ці вузли для забезпечення зручнішої навігації їх при використанні клієнтської програми мають ієрархію у своїй структурі.

Обмін даними в середовищі «клієнт-сервер» може відбуватися як за допомогою одержання відповідей на запити миттєво, також і за схемою «видавець-підписник» («publisher-subscriber»). Цей процес наведено на рисунку 1.4. Для другого випадку програма клієнта виконує «підписку» на отримання певних даних, які сервер повинен буде надавати в міру їх появи. Для реалізації режиму підписки сервер здійснює безперервний контроль (моніторинг) вузлів і відповідних їм реальних об'єктів з метою виявлення змін. При виявленні змін в даних, події або аварійні сигнали (АЛАРМ) сервер генерує повідомлення, яке передається клієнту по каналу підписки.

Протокол OPC UA дозволяє обмін поточними даними навіть між двома серверами. З цією метою один сервер стає клієнтом, другий залишається сервером. Саме так дозволено поєднати декілька серверів в ланцюжок послідовно, завдяки цьому кожен сервер з буде з одного боку ланцюжка

виконувати роль клієнта, з іншого буде сервером, що і було показано на рисунку 1.1 [3 - 6].

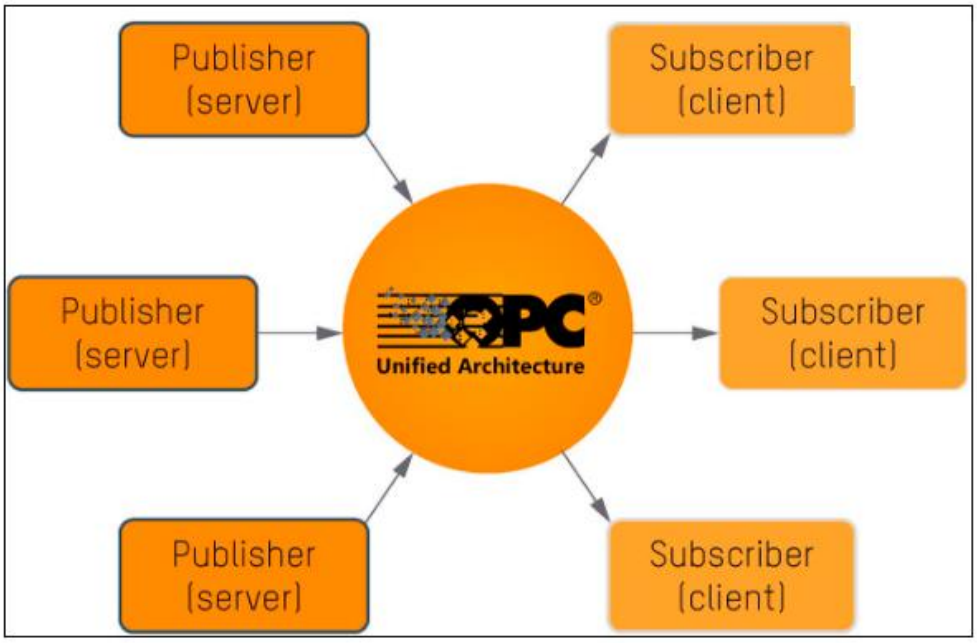

Рисунок 1.4 – Процес обміну даними клієнт-сервером для OPC UA

1.4 Огляд основних можливостей, засобів та етапів проектування SCADA-систем

Варто перерахувати основні можливості і засоби, які притаманні всім без вийнятку SCADA-системам [7]:

– застосування автоматизованого розроблення, яке дає змогу створити ПЗ системи без використання стандартних мов програмування;

– наявність засобів збору первинної технологічної інформації від існуючих пристроїв на нижньому рівні;

– присутність засобів реєстрації аварійних ситуацій і видачі сигналів про аварії;

– наявність засобів архівування та зберігання інформації (як правило, реалізуються на основі інтерфейсів з найбільш популярними БД);

– наявність засобів опрацювання первинної вхідної інформації;

– наявність засобів візуалізації поточної та історичної інформації у вигляді таблиць, графіків, гістограм, динамічних мнемосхем, анімації та ін .;

– друк звітів і протоколів довільної форми в задані моменти часу;

– ввід і передавання команд оператора в програмовані логічні контролери та інші пристрої системи;

– вирішення прикладних задач користувача і їх взаємозв'язок з поточною вимірюваної інформацією і управлінськими рішеннями;

– інформаційні зв'язки з серверами і робочими станціями через структуру мережі.

Необхідно навести всі основні етапи проектування розробки для автоматизації процесів, в основі якої є SCADA-система [8, 11].

1. Побудова системної архітектури у загальному. Цей етап визначає функціональність кожного складового елементу автоматизації за призначенням.

2. Вирішення проблем використання ймовірної розподіленої системної архітектури, забезпечення використання окремих вузлів та елементів з «гарячим резервуванням».

3. Розробка та реалізація прикладної системи керування для кожного вузла. Цей етап дозволяє фахівцю в царині автоматизованих технологічних процесів забезпечити вузлові елементи архітектури раніше створеними алгоритмами, власне сукупність яких і вирішить поставлену задачу на проектування.

4. Установка зв'язку між параметрами прикладної системи (внутрішніми змінними) та інформацією, що надходить від наприклад, програмованих логічних контролерів чи інших пристроїв низького рівня. Таким чином, приводиться у відповідність стан реального об'єкта управління і стан його відображення в прикладній програмі, що дозволить уникнути колізій при використанні.

5. Налагодження в прикладній програмі, яка створена в режимі емуляції, та в реальному варіанті використання.

Зазначені вище функціональні можливості SCADA в основному і визначають вартість розробки прикладного ПЗ, а також терміни окупності всієї системи [12].

1.5 Висновки до першого розділу

У першому розділі виконано огляд стандарту OPC як основного способу взаємодії SCADA-системи з зовнішніми пристроями різних фірм-виробників, проаналізовано специфікацію OPC UA. Зроблено докладний опис стратегії формування системи на базі OPC UA. Також наведено можливості, які притаманні SCADA-систем та основні етапи їх проектування.

2 ТЕОРЕТИКО-ПРАКТИЧНЕ ДОСЛІДЖЕННЯ

2.1 Порівняльний аналіз основних спеціалізованих БД та сховищ даних для часових рядів

Відповідно до задач проведення дослідження, були виділені наступні ключові вимоги до сховища даних:

– воно має бути вбудовуваним, тобто реалізовано у вигляді бібліотеки, яка під’єднюється;

– мати високу продуктивність операції записи (на рівні 700 тисяч операцій в секунду);

– наявність API для мови C++, оскільки розробка проекту ведеться на C++;

– мати BSD, MIT або Apache 2-ліцензію, оскільки Сервер історії - комерційна розробка, і, отже, необхідно вибрати сховище даних з ліцензією, що допускає вбудовування в пропрієтарні комерційні продукти без відкриття вихідного коду;

– доступність для ОС UNIX і ФС ext4;

– актуальність, тобто чи розвивається проект, як часто виходять релізи і коли вийшов останній;

– має розташовуватися на диску (on-disk), а не в оперативній пам'яті (in-memory).

На початковому етапі було прийнято рішення використовувати спеціалізовану БД часових рядів (Time Series Databases), в якій дані будуть зберігатися в упорядкованому за часом вигляді. У 2019 році було проведено масштабний аналіз і складено рейтинг сховищ такого типу [13]. На основі результатів, отриманих в ході дослідження, з'ясувалося, що з 16 протестованих БД лише 3 можуть забезпечити продуктивність, яка обчислюється сотнями тисяч операцій в секунду. Вони і представлені в таблиці 2.1.

Таблиця 2.1 – Порівняльний аналіз БД для часових рядів

Критерій	DalmatinerDB	InfluxDB	Prometheus
Сайт	dalmatiner.io	influxdata.com	prometheus.io
Ліцензія	MIT	MIT	Apache 2
Підтримувані виміри	Метрики	Метрики, події	Метрики
Метрика фіксується з точністю до	Мілісекунд	Наносекунд	Мілісекунд
Написано на	Erlang	Go	Go
Підтримувані типи даних	int64, float64	int64, float64?, bool, string	float64
Швидкість запису (1 вузол)	2.5-3.5 млн. метрик / с	470 тис. метрик / с	800 тис. метрик / с
Швидкість запису (кластер з 5 вузлів)	15-20 млн. метрик / с	-	-
Швидкість виконання запитів	Висока	Висока	Середня
Переваги	Кластеризація, висока відмовостійкість, висока швидкість зчитування і запису, виразна мова запитів	Зручна в експлуатації, гнучка в налаштуванні, вичерпна документація	Зручна в експлуатації, висока продуктивність, виразна мова запитів
Недоліки	Працює тільки з ФС ZFS, недолік документації	Не підтримує кластеризацію, велика історія багів	Не призначена для використання в бекенді

Найкращим вибором є, очевидно, DalmatinerDB, яка здатна проводити запис з рекордною швидкістю (в середньому 3 млн метрик /с). Однак такі значення досягаються при 16 процесорах, 60 Гб оперативної пам'яті і SSD-диску з ФС ZFS. У нас же, згідно вимог, на сервері буде HDD-диск з ФС ext4, отже, DalmatinerDB нам не підходить.

Наступним кандидатом (по продуктивності) є Prometheus. Він показує хороші результати, має виразну мову запитів і, на відміну від DalmatinerDB, працює з ФС ext4 і не вимагає SSD-диска, але не має офіційної бібліотеки для C++. Існує, проте, бібліотека, розроблена третьою стороною, але вона перебуває в стадії тестування, що не дозволяє використовувати її в продакшені.

InfluxDB, при всіх його перевагах, на жаль, не може забезпечити необхідну продуктивність і не має бібліотеки для C ++.

Оскільки серед спеціалізованих БД жодна нам не підійшла, було прийнято рішення продовжити пошук відповідного варіанту серед вбудовуваних СУБД чи сховищ «ключ-значення». Для сховищ такого типу існують численні бенчмарки [14 -20], однак вони сильно суперечливі. Так, в [14, 15] BangDB практично у всіх тестах значно перевершує LevelDB, тоді як в [18-20] обидві БД показують приблизно однакові результати. З огляду на це, було принято рішення провести власні тести продуктивності з попередньо складеного списку кандидатів, до якого увійшли: LevelDB, HyperLevelDB, BangDB, Sophia і SQLite. Зупинимося на кожній детальніше [21].

2.1.1 LevelDB

Це сховище «ключ-значення», розроблене компанією Google.

Переваги [18]:

– ключі та значення представляються довільним набором байт;

– дані зберігаються в відсортованому до ключу вигляді;

–розробнику надається можливість перевизначити функцію порівняння щоб задати потрібний порядок сортування;

– простий інтерфейс для базових операцій - Put (key, value), Get (key), Delete (key);

– кілька операції можна виконати як одну (batch);

– можливість зробити знімок БД;

– стиснення даних через Snappy (бібліотеку для компресії даних від Google);

–підтримка обходу ітератором в прямому і зворотному порядку.

Обмеження:

–не підтримує реляційну модель даних, індекси і мову SQL;

– тільки один процес може мати доступ до БД в певний момент часу.

2.1.2 HyperLevelDB

Це форк LevelDB, в якому були збережені всі її переваги і покращено багатопотокову взаємодію з БД.

LevelDB влаштована таким чином, що в певний момент часу тільки один потік може щось записувати в БД, а якщо таких потоків декілька, то для їх синхронізації використовується м'ютекс (семафор) і запис проводитися за принципом черги (FIFO - first in, first out - першим увійшов, першим вийшов): доступ до БД отримує потік, що знаходиться в «голові» черги, а всі інші чекають. Це гарантує, що одночасна робота цих потоків з БД буде безпечна і цілісність даних не порушиться.

HyperLevelDB дозволяє всім потокам одночасно записувати дані в БД, тим самим роблячи паралельну роботу більш ефективною, а синхронізація досягається за рахунок іншого механізму [17].

2.1.3 BangDB

Сховище «ключ-значення», котре може перебувати і в оперативній пам'яті (in-memory), і на диску. Основна його перевага - висока продуктивність при багатопотоковій роботі.

Дана БД підтримує ACID-транзакції і використовує компоненти з допомогою яких досягається оптимізація ресурсів, що гарантує високу продуктивність і низьку затримку (latency). BangDB реалізує свій власний буферний кеш, систему управління буфером, механізм управління пам'яттю і т.д., а так само має багатий API, який крім стандартних CRUD-операцій забезпечує гнучкий інтерфейс запитів (з використанням B-дерев) [14].

Дане сховище може бути серверним або вбудовуваним. Для тестування в даній роботі використовувалося останнє.

2.1.4 Sophia

Сучасне сховище «ключ-значення» з підтримкою транзакцій, яке може розташовуватися як в оперативній пам'яті, так і на диску. Воно розроблено для забезпечення його оптимальної роботи без деградації в часі і гарантує складність порядку 0 (1) в найгіршому для операцій читання і запису [22].

Переваги:

– підтримка ACID-транзакцій;

– управління паралельним доступом за допомогою багатоверсійності;

– оптимістичне блокування;

– 0 (1) в найгіршому для читання / запису;

– стиснення даних;

– простий інтерфейс (API);

– не має зовнішніх залежностей.

2.1.5 SQLite

Відкрита вбудована реляційна СУБД.

Переваги [20]:

– вбудована (звернення до БД відбувається не за рахунок клієнт-серверної взаємодії, а як виклик бібліотечної функції, що збільшує продуктивність);

– БД є одним файлом;

– підтримка ACID-транзакцій;

– підтримка багатопотокового читання;

– підтримка динамічно типізованих типів даних;

– підтримка SQL.

Недоліки:

– відсутність призначеного для користувача управління;

– бідні можливості додаткового налаштування.

SQLite тестувалася ще з метою експериментально перевірити твердження про те, що реляційні СУБД не годяться для завдання збереження часових рядів в умовах високого навантаження (великий потік даних) [7].

2.2 Тестування сховищ даних

Для всіх сховищ даних та БД, перерахованих в п. 2.1 кваліфікаційної роборти, були розроблені і проведені тести продуктивності.

Метою тестування було визначення продуктивності тільки операції запису, оскільки на операції читання і модифікації даних замовником обмежень накладено не було.

Характеристики комп'ютера, на якому проводилося тестування, такі:

– процесор: Intel® Core ™ i5-4460 CPU @ 3.20GHz × 4;

– оперативна пам’ять 16 Гб;

– жорсткий диск (HDD) WDC WD10EFRX 1 Тб;

– ОС Ubuntu 16.04.2 LTS (64 bit);

– ФС ext4.

Там, де це можливо, для збільшення продуктивності здійснювалося підналаштовування параметрів БД, зокрема:

– для SQLite були виставлені наступні параметри: PRAGMA synchronous = OFF і PRAGMA journal_mode = MEMORY.

Значення OFF першого параметра означає, що транзакція вважається

такою, що завершилася після того, як SQLite передав їх ОС (а не після того, як вони фактично були записані на диск), отже, якщо станеться збій живлення, то цілісність даних може бути порушена.

Другий параметр вказує, що файл журналу буде розташовуватися в оперативної пам'яті з такими ж наслідками, як і для першого параметру (взагалі, наявність журналу означає, що транзакція ще не завершена) [23];

– у Sophia є параметр cache size, значення якого для досягнення необхідної швидкості запису слід встановити в залежності, по-перше, від типу диска (HDD або SSD), а по-друге, від його об’єму [24]. Для HDD-диска об’ємом 1 Тб значення даного параметра для досягнення максимальної швидкості запису рівне 1365320 Мб. Встановлення цього параметра здійснюється через змінну db.compaction.cache;

– у BangDB параметру persist type (BANGDB_PERSIST_TYPE) було встановлено значення INMEM_PERSIST, яке означає, що дані за замовчуванням будуть записуватися в буфер, а при переповненні буфера - на диск (з буфера);

– у LevelDB і HyperLevelDB параметру write_buffer_size встановлені різні значення, однак істотного впливу на продуктивність вони не мали [14].

Дані, які записувалися в сховище - сигнали, описані в п. 2.4. Структура сигналу в JSON-форматі наведена в лістингу 2.1.

Лістинг 2.1 - Структура сигналу в JSON-форматі

```
"Signal": {
  "NodeId": {
      "NamespaceIndex": "quint16",
      "IdentifierType": "quint8",
      "Identifier": "QString"
  },
 "DataValue": {
      "SourceTimestamp": "quint64",
      "ServerTimestamp": "quint64",
      "Value": "QVariant",
      "StatusCode": "quint32"
  }
}
```

Ключем є сукупність об'єкта NodeId і значення DataValue.sourceTimestamp (23 байта), а значенням DataValue.serverTimestamp, DataValue.value і DataValue.statusCode (29 байт).

На рисунку 2.1 представлена блок-схема процедури тестування.

Рисунок 2.1 – Блок-схема процедури тестування

Деталізація кроків алгоритму тестування наведена нижче по тексту.

1. На колекторі сервера історії генерувалося велика кількість (2 млн.) сигналів (таким чином імітувалося пікове навантаження). В одних серія тестів сигнали були впорядковані по часу (що дуже близько до реальної ситуації), в інші – не впорядковані.

2. Знову згенеровані сигнали записувалися в буфер сервера історичних даних.

3. Після закінчення прийому чергової порції даних сервер історичних даних здійснював десеріалізацію.

4. Створювалася структура десеріалізованних даних для запису в БД (для кожного движка БД - своя структура).

Наприклад, для LevelDB - ключ і значення в вигляді масиву байт, для SQLite формувалася команда INSERT і т.д.

5. Було зроблено запис в БД: в одних серіях - по одному, в інших - пакетами (LevelDB, HyperLevelDB) або об'єднувалися в одну транзакцію (SQLite, Sophia).

Замір часу запису даних в БД відбувався з кроку 3 по крок 5 включно. Для цього використовувався клас QElapsedTimer.

Для достовірності результатів дана процедура виконувалася близько 40 разів, після чого обчислювалося середнє значення.

Підсумкові результати представлені в таблиці 2.2. Виходячи з них, вибір був зроблений на користь LevelDB.

Таблиця 2.2 – Результати тестів (тис. операцій / с)

Тип тесту	LevelDB	BangDB	HyperLevelDB	Sophia	SQLite
Послідовний запис одиночними сигналами	372	550	388	65	51
Випадковий запис одиночними сигналами	70	150	218	47	52
Послідовний запис пакетами сигналів (по 26/52/104 Мб)	861/862/848	Можливість пакетного запису відсутня	694/676/690	85/92/92	140/129/125
Випадковий запис пакетами сигналів (по 26/52/104 Мб)	422/404/373	Можливість пакетного запису відсутня	500/323/286	52/48/72	136/138/129

2.3 Структура сигналу

Згідно зі специфікацією OPC UA, кожен сигнал відноситься до якого-небудь вузла (Node) в адресному просторі (AddressSpace) OPC UA сервера [5].

Основне призначення адресного простору - надати уніфікований спосіб доступу клієнта до вузлів (які знаходяться на сервері).

Вузол - це уявлення об'єкта або групи об'єктів в адресному просторі OPC UA сервера. Кожен вузол є екземпляром класу вузла (NodeClass), серед яких є: "Змінні" (Variable), «Метод» (Method), «Тип даних» (DataType) і т.д.

Вузли описуються атрибутами, збираються в ієрархії і можуть посилатися один на одного [25]. На рисунку 2.4 представлена структура вузла.

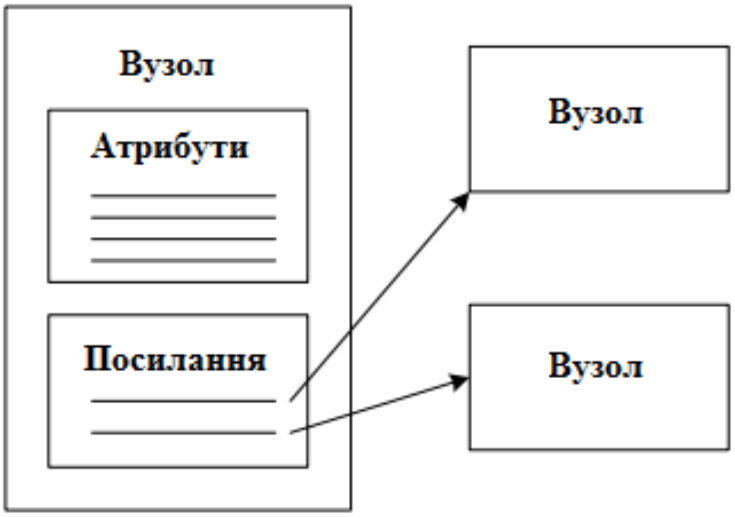

Рисунок 2.4 – Структура вузла

Приклад адресного простору OPC UA сервера наведений на рисунку 2.5.

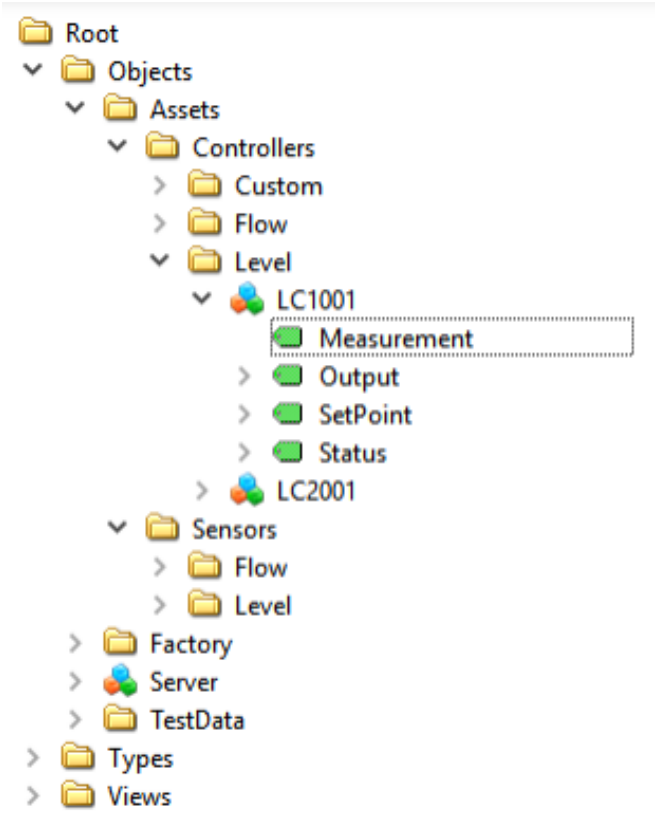

Рисунок 2.5 - Адресний простір OPC UA сервера

Вузли з піктограмами папок і кубиків - це об'єкти, а Measurement, Output, SetPoint і Status - змінні.

На рисунку 2.6 представлені приклади значень змінної Measurement в різні моменти часу.

#	Node Id	Value	Source Timestamp	Server Timestamp	Statuscode
1	NS2|String|1:LC1001?Measurement	-2.67537e+009	10:33:06.832	10:33:06.832	Good

#	Node Id	Value	Source Timestamp	Server Timestamp	Statuscode
1	NS2|String|1:LC1001?Measurement	-1.49619e+006	10:33:55.856	10:33:55.856	Good

#	Node Id	Value	Source Timestamp	Server Timestamp	Statuscode
1	NS2|String|1:LC1001?Measurement	-8.48687e+011	10:33:58.858	10:33:58.858	Good

Рисунок 2.6 - Значення змінної Measurement в різні моменти часу

На рисунку 2.7 показані всі атрибути вузла.

Attribute	Value
NodeId	NodeId
NamespaceIndex	2
IdentifierType	String
Identifier	1:LC1001?Measurement
NodeClass	Variable
BrowseName	2, "Measurement"
DisplayName	"", "Measurement"
Description	"", "A generic value."
WriteMask	0
UserWriteMask	0
Value	
SourceTimestamp	01.04.2017 23:30:52.595
SourcePicoseconds	0
ServerTimestamp	01.04.2017 23:30:53.017
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	7.49723e-15
DataType	Float
NamespaceIndex	0
IdentifierType	Numeric
Identifier	10
ValueRank	-1
ArrayDimensions	Null
AccessLevel	CurrentRead
UserAccessLevel	CurrentRead
MinimumSamplingInterval	0
Historizing	false

Рисунок 2.7 - Атрибути вузла Measurement

Для ідентифікації вузла в адресному просторі використовується структура даних NodeId. NodeId - це первинний ключ, що складається з трьох елементів, перерахованих в таблиці 2.3.

Таблиця 2.3 - Структура NodeId

Ім’я	Тип	Опис
NamespaceIndex	UInt16	Індекс ідентифікатора простору імен
IdentifierType	Enum	Формат і тип даних ідентифікатора (наведено в табл. 2.4)
Identifier	*	Ідентифікатора вузла в адресному просторі OPC UA сервера

Таблиця 2.4 - Значення IdentifierType

Значення	Опис
NUMERIC_0	Числово значення
STRING_1	Стрічка
GUID_2	Глобально унікальний ідентифікатор
OPAQUE_3	Байтова стрічка

Для задачі збереження історичних даних, згідно специфікації OPC UA, зі всієї множини атрибутів потрібно зберігати тільки NodeId і складений тип DataValue [25]. Структура DataValue представлена таблиці 2.5.

Таблиця 2.5 - Структура DataValue згідно специфікації OPC UA

Ім’я	Тип	Опис
SourceTimestamp	UtcTime	Часова мітка джерела сигналу, тобто час, в яке відбулася реєстрація значення на джерелі даних. Вимірюється з точністю до мілісекунд
ServerTimestamp	UtcTime	Часова мітка сервера, тобто час, в який сервер отримав дані від джерела сигналу. Вимірюється з точністю до мілісекунд
Value	BaseDataType	Значення, отримане від джерела реальних даних. Дане значення отримує OPC UA клієнт (SCADA-система). Якщо Status-Code - Bad, то значення виставляється в Null
StatusCode	Enum	Параметр, що характеризує ступінь достовірності значення. Може набувати значення, перечислені в таблиці 2.6

Таблиця 2.6 – Значення StatusCode

Значення	Опис
Good	Значення сигналу є коректним
Bad	Дані з контролера були отримані з критичними помилками або не було отримано взагалі
Uncertain	При отриманні даних виникли некритичні помилки

Тепер розглянемо представлення сигналу в програмному коді.

Для значень IdentifierType (таблиця 2.4) використовується просте перераховування, наведене в лістингу 2.2.

Лістинг 2.2 – Представлення значень IdentifierType в коді

```
enum IdentifierType: quint8 {
NUMERIC_0,
STRING_1,
GUID_2,
OPAQUE_3
};
```

Структура NodeId (таблиця 2.1) представлена в лістингу 2.3.

Лістинг 2.3 – Представлення структури NodeId в коді

```
struct NodeId {
quint16 namespaceIndex;
IdentifierType identifierType;
QString identifier;
};
```

quint8 і quint16 - визначені в заголовочному файлі QtGlobal типи даних для подання беззнакових цілих чисел, які на будь-який платформі займають 8 і 16 біт відповідно, а QString - рядок символів юнікоду.

Структура DataValue (таблиця 2.5) представлена в лістингу 2.4.

Лістинг 2.4 – Представлення структури DataValue в коді

```
struct DataValue {
QVariant value;
quint32 statusCode;
quint64 sourceTimestamp;
quint64 serverTimestamp;
};
```

Змінна типу QVariant може зберігати значення будь-якого типу, що для нас особливо актуально, так як значення сигналу може бути логічним, цілочисельним або числом з плаваючою крапкою.

Значенням змінної statusCode, рівному 0 відповідає Good (таблиця 2.4), 1 - Bad, 2 - Uncertain.

Як можна бачити, час (sourceTimestamp і serverTimestamp) в нашій реалізації представлено не в UTC-форматі, а цілим числом.

Це число UNIX-час (воно показує, скільки секунд пройшло з 00:00:00 UTC 1 січня 1970 року) до якого, з метою підвищення точності, додані три молодші розряди для подання мілісекунд.

І, нарешті, сам сигнал є агрегатом структур NodeId і DataValue (лістинг 2.5):

Лістинг 2.5 – Представлення агрегату структур NodeId і DataValue

```
struct Signal {
NodeId nodeId;
DataValue dataValue;
};
```

2.4 Висновки до другого розділу

У другому розділі порівняно основні спеціалізовані БД для часових рядів (DalmatinerDB, InfluxDB, Prometheus) та сховищ «ключ-значення» або вбудовуваних СУБД (LevelDB, HyperLevelDB, BangDB, Sophia і SQLite). Проаналізовано їх основні функціональні можливості та параметри, а також протестовано продуктивність операції запису. Виходячи з них, вибір був зроблений на користь LevelDB. Докладно описано структуру сигналу відповідно до специфікації OPC UA.

3 РЕАЛІЗАЦІЯ СЕРВІСУ

3.1 Архітектура сервера історії

На рисунку 3.1 наведена схема взаємодії Сервера історії з компонентами SCADA-системи.

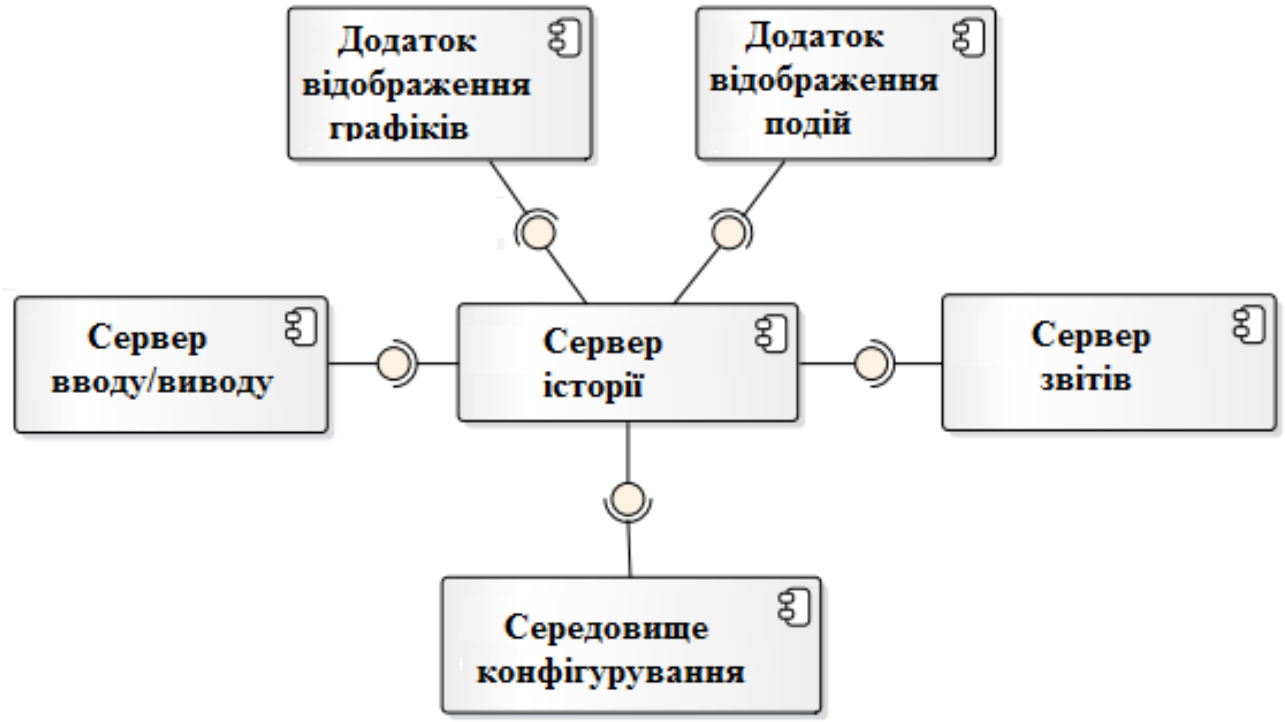

Рисунок 3.1 - Схема взаємодії сервера історії з компонентами SCADA-системи

Опис компонентів SCADA-системи, що взаємодіють з Сервером історії (за специфікацію OPC UA) [26]:

– сервер вводу / виводу оперативних даних є джерелом оперативних даних і подій, одержуваних сервером історії;

– сервер звітів є споживачем історичних даних і подій, переданих сервером історії;

– додаток відображення подій є споживачами історичних подій, переданих сервером історії;

– додаток відображення графіків є споживачами історичних подій, переданих сервером історії;

– середовище конфігурації забезпечує конфігурацію Сервера історії.

Сам Сервер історії складається з двох основних компонентів: колектора і Сервера історичних даних (рисунок 3.2). Колектор Сервера історії займається збором оперативних даних і подій від джерел даних, і, в разі відсутності зв'язку з сервером історичних даних, буферизує дані для подальшої їх передачі на сервер історичних даних після появи сервера історичних даних на зв'язку. Будова колектора для вирішення задачі даної кваліфікаційної роботи не суттєвий, і його розглядати не варто.

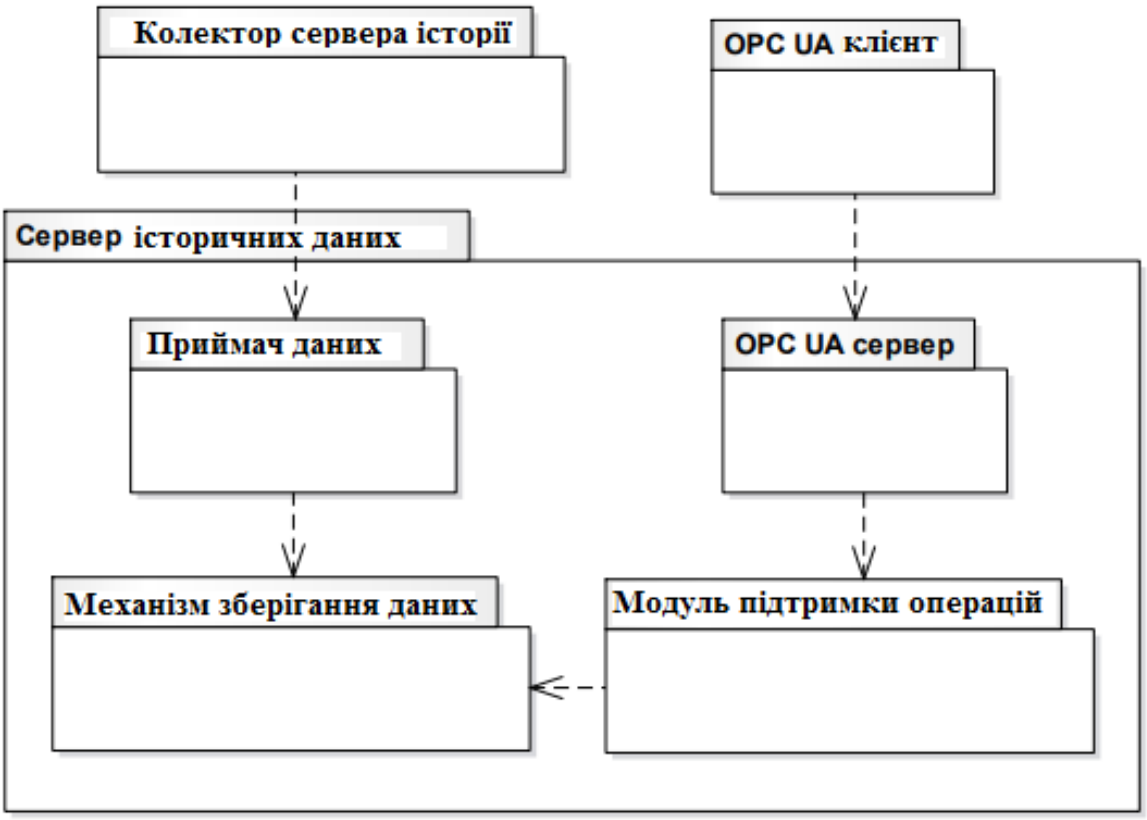

Рисунок 3.2 - Структура сервера історії

Сервер історичних даних, отримуючи дані від Колектора сервера історії, зберігає їх у БД історії, а також надає до них доступ Споживачам даних [27].

Сервер історичних даних має наступні програмні компоненти (модулі):

– приймач даних. Приймач даних на базі протоколу TCP / IP реалізує прийом даних від основного і / або резервного Колекторів сервера історії. Після того, як дані отримані, вони закладаються в БД історії за допомогою Механізму зберігання даних (Движка сховища).

Вхідними та вихідними даними модуля є технологічні дані;

– механізм зберігання даних. Механізм зберігання даних відповідає за розміщення даних в файлах певного формату (або оперативній пам'яті) і організацію конкурентного доступу до них.

Механізм зберігання даних надає низькорівневий інтерфейс з підтримкою простих операцій, використовуючи який відповідні модулі Сервера історичних даних можуть маніпулювати даними в БД історії. Також механізм зберігання даних може надавати додаткові можливості, зокрема створення і підтримку структури індексі для підвищеня продуктивності пошуку даних, підтримку блокувань при одночасній роботі з файлами БД і ін.

Файли даних, якими оперує Механізм зберігання даних, поділяються на первинні та архівні. Первинні файли складають БД історії і допускають вставку даних (наприклад, сигналів, отриманих «заднім числом»). Після закінчення терміну зберігання в первинному вигляді, дані переводяться в Архів. Архівні файли оптимізовані тільки для виконання операцій читання. Періодично відповідно до параметру, що визначає загальний термін зберігання даних, виконується процедура очищення сховища, тобто архівні файли, для яких закінчився термін зберігання, видаляють.

Вхідними даними модуля є низькорівневі запити отримання, збереження, обробку історичних даних або зберігання технологічних даних, а вихідними – історичні дані;

– модуль підтримки операцій. Модуль підтримки операцій є «обгорткою» навколо Механізму зберігання даних для реалізації різних операцій над тимчасовими рядами (впорядкованими в часі сукупностями значень спостережуваних сигналів і їх властивостей), в тому числі операції

фільтрації даних при запису в БД історії, операції інверсного збереження даних, операції зберігання даних в стислому вигляді без втрати інформації.

Вхідними даними модуля є високорівневі запити на отримання, зберігання, обробку історичних даних або зберігання технологічних даних, а вихідними – історичні дані;

– OPC UA сервер. Він дозволяє споживачам отримати доступ до історичних даних або подій за специфікацією OPC UA на базі елементів простору адрес, які витягуються з БД конфігурації сервера історії.

Вхідними даними модуля є запити від споживачів даних на отримання, зберігання, обробку історичних даних, а вихідними - історичні дані.

3.2 Програмна архітектура сервісу

Діаграма класів розроблюваного сервісу представлена на рисунку 3.3.

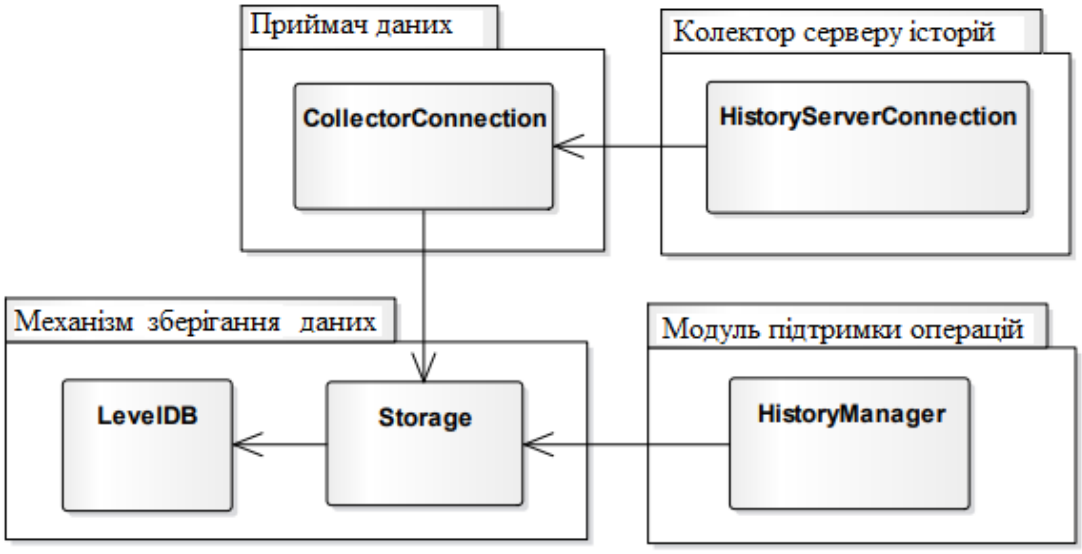

Рисунок 3.3 - Діаграма класів сервісу

Як видно з рисунку 3.3, програмна архітектура розроблюваного сервісу складається з наступних компонентів:

– Механізм зберігання даних;

– Приймач даних;
– Модуль підтримки операцій;
– Колектор серверу історій.

Більш докладно дані пакети розглянуті в наступних розділах.

3.3 Реалізація Механізму зберігання даних

Діаграма класів Механізму зберігання даних представлена на рисунку 3.4.

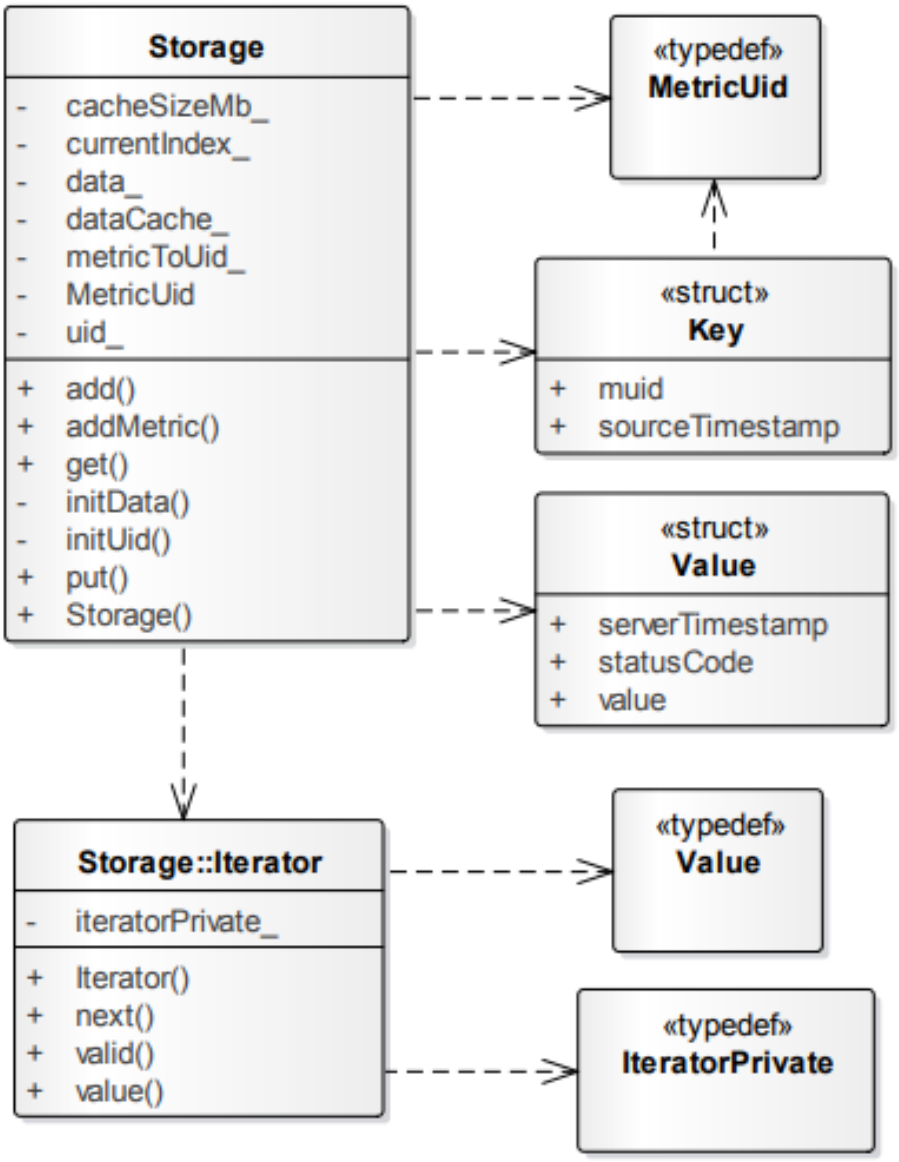

Рисунок 3.4 - Діаграма класів

Перш за все, сигнал розділяється на ключ (структура Key) і значення (Структура Value). Варто звернути увагу, що ключ - це не тільки NodeId, але ще і час, в який сигнал був зафіксований на джерелі. Якби ключ складався тільки з NodeId, то він був би ідентифікатором вузла, а не окремого сигналу.

Поле muid структури Key має тип MetricUid (size_t) і є ідентіфікатором метрики.

При додаванні сигналів (за допомогою функції add ()) відбувається серіалізація структури NodeId (функція serializeNodeIdToString ()) і відображення в лічильник метрик. metricToUid_ - словник, який оголошений так:

std :: unordered_map <std :: string, MetricUid> metricToUid_. Змінна currentIndex - кількість метрик. Якщо для поточного десеріалізованого значення відсутнє значення в словнику (metricToUid_), то воно додається туди за допомогою функції addMetric ().

Діаграма послідовностей для операції add () зображена на рисунку 3.5.

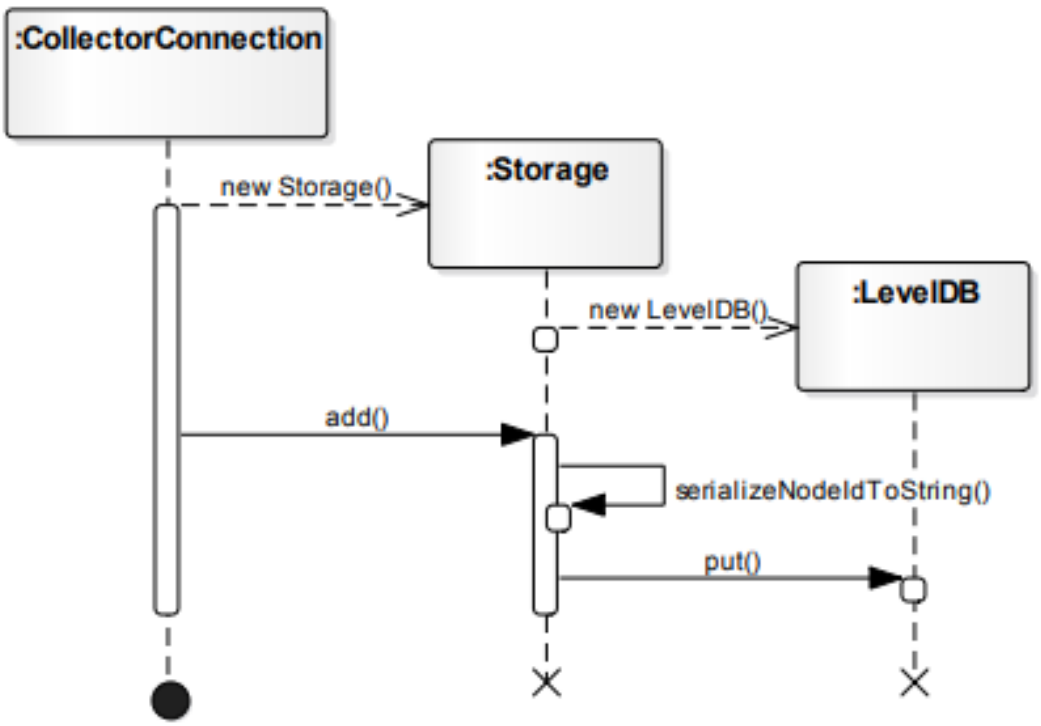

Рисунок 3.5 - Діаграма послідовностей

uid_ і data_ - покажчики на сховища метрик і даних, які ініціалізіруются за допомогою функцій initUid () і initData () відповідно.

cacheSizeMb_ - розмір кеша сховища.

Для обходу БД LevelDB використовується ітератор (клас leveldb :: Iterator).

Функція get () повертає ітератор, який проходить по заданому діапазону ключів.

3.4 Реалізація Приймача даних

Діаграма Приймача даних представлена на рисунку 3.6.

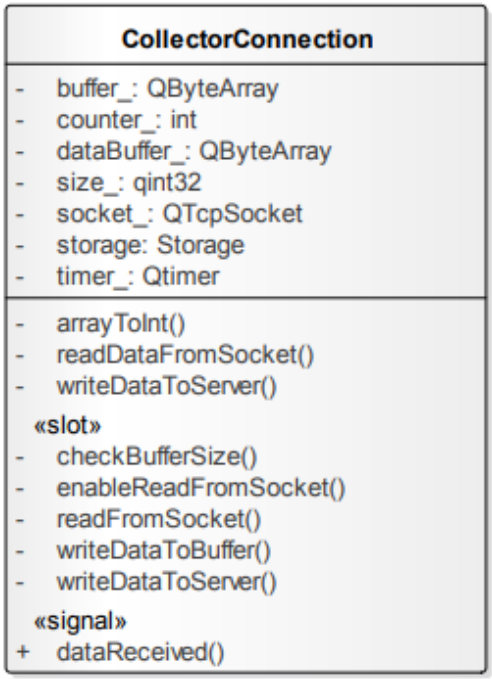

Рисунок 3.6 – Клас CollectorConnection

Призначення класу CollectorConnection - дати можливість колектору сервера історії підключатися до Сервера історичних даних і відправляти історичні технологічні дані з метою подальшого їх збереження в БД.

Для того, щоб Сервер історичних даних міг отримувати дані, які відправляються через TCP-з'єднання, використовуються сигнали і слоти.

В термінах Qt, будь-який слот –це певна функція, яка викликається як відповідь на визначений сигнал, а сигнал відбувається, коли проходить певна подія.

В нашому випадку 4 сигналу з'єднуються з 4 різними портами:

1. connect (socket_, SIGNAL (readyRead ()), SLOT (readFromSocket ()), Qt :: UniqueConnection);

2. connect (socket_, SIGNAL (disconnected ()), SLOT (deleteLater ()));

3. connect (this, SIGNAL (dataReceived (QByteArray)), SLOT (writeDataToBuffer (QByteArray)));

4. connect (timer_, SIGNAL (timeout ()), SLOT (writeDataToServer ()));

У першому випадку, при сигналі readyRead () викликається функція readFromSocket (), яка записує дані, що приходять на сокет (змінна socket_) в буфер (змінна buffer_), при цьому збільшується значення змінної size_, яка показує розмір даних (в байтах) прийшли на сервер.

У другому випадку, при від'єднанні колектора від сервера видаляється змінна socket_.

У третьому випадку, після того як пакет даних був цілком отриманий з колектора, викликається функція writeDataToBuffer (QByteArray), яка записує отримані дані в буфер (dataBuffer_), посилає на сокет повідомлення «COMMIT_TRANSACTION», яке свідчить про те, що дані були отримані сервером в повному обсязі. Після чого викликається функція checkBufferSize (), яка перевіряє розмір буфера і, якщо він перевищує певний розмір, викликає функцію writeDataToServer ().

У четвертому випадку, запис даних в БД відбувається після виходу часу таймера (timer_).

3.5 Реалізація Модуля підтримки операцій

Цей модуль фактично є адаптером між OPC UA сервером і Механізмом зберігання даних. Він перетворює інтерфейс OPC UA сервера до інтерфейсу Механізму зберігання даних. В результаті, OPC UA клієнти для отримання історичних даних звертаються до OPC UA сервера за специфікацією OPC UA, а вже сервер перенаправляє запит Модуля підтримки операцій. В результаті, зменшується зв'язаність, оскільки OPC UA клієнти нічого не знають про те, як організовано зберігання даних на сервері історичних даних.

Для реалізації були обрані дві операції читання (ReadRaw і ReadAtTime), операція модифікації (Update), а також операція, яка обчислює середнє значення на певному часовому відрізку (Average). Всі ці функції визначені в [23, 25].

Діаграма класів Модуля підтримки операцій наведена на рисунку 3.7.

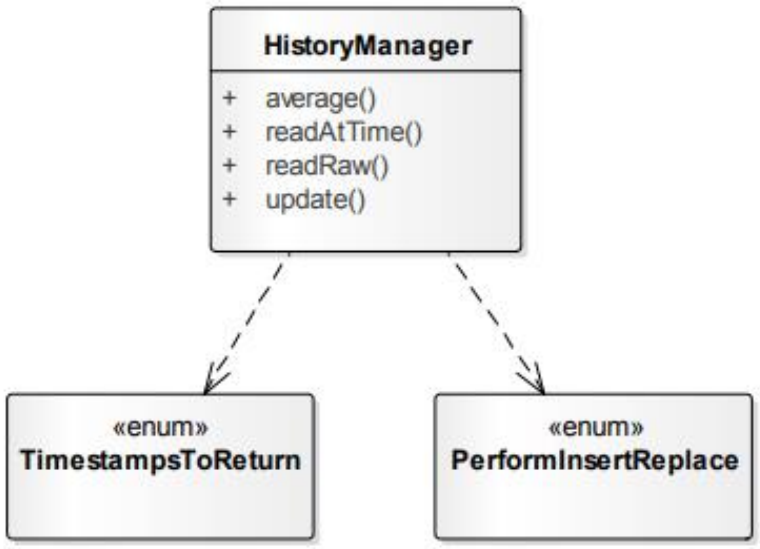

Рисунок 3.7 - Діаграма класів

3.5.1 Функція ReadRaw

Функція ReadRaw призначена для читання сирих історичних даних вузла (сигналів) на певному тимчасово інтервалі. Код сигнатури функції предсталено в лістингу 3.1.

Лістинг 3.1 – Код функції ReadRaw

```
quint32 HistoryManager :: readRaw (
Core :: TimestampsToReturn timestampsToReturn,
quint32 maxValues,
uint64 startTime,
quint64 endTime,
bool returnBounds,
NodeId nodeId,
QQueue <DataValue> & dataValues);
```

Опис параметрів функції:

– timestampsToReturn задається перерахуванням (див. Таблицю 3.1);

– maxValues - задає найбільшу кількість значень, які повертаються (якщо 0 - повертаються всі значення з тимчасового інтервалу);

– startTime - початок часового відрізку;

– endTime - кінець часового відрізку відповідно;

– returnBounds - якщо значення true - то будуть повернуті значення з відрізка [startTime, endTime], інакше - з півінтервалу [StartTime, endTime);

– nodeId - див. П. 2.3;

– cdataValues - див. П. 2.3;

Таблиця 3.1 – Перерахунок TimestampsToReturn

Значення	Опис
SOURCE_0	Повертає тільки час клієнта
SERVER_1	Повертає тільки час сервера
BOTH_2	Повертає і час сервера, і клієнта

Функція ReadRaw повертає статус-код, який свідчить про те, що функція виконана успішно (або неуспішно).

Діаграма взаємодії для операції readRaw () представлена на рисунку 3.8.

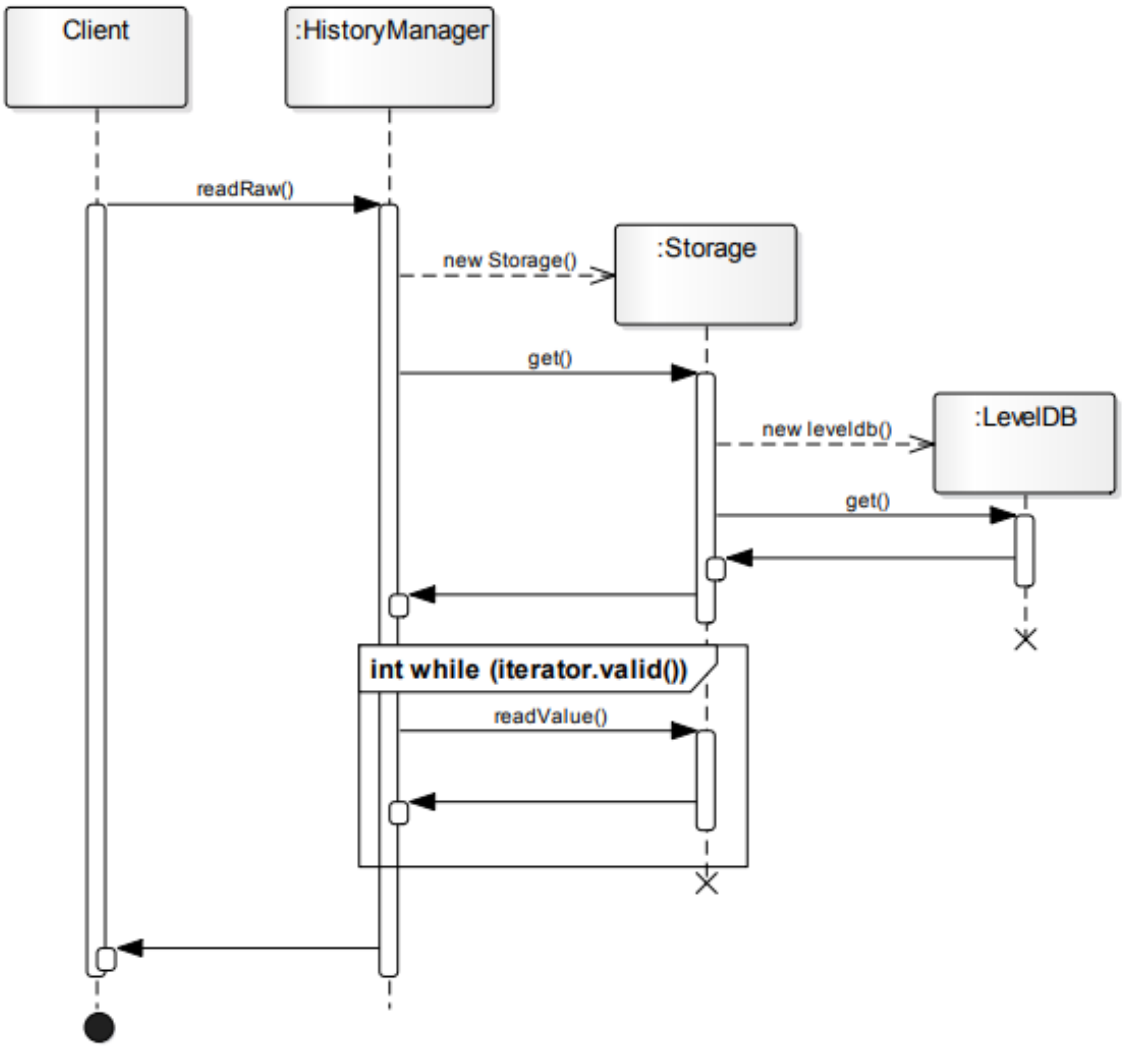

Рисунок 3.8 – Діаграма взаємодії

3.5.2 Функція ReadAtTime

Функція ReadAtTime повертає значення історичних даних вузла для певних тимчасових міток. Сигнатура функції наведена в лістингу 3.2.

Лістинг 3.2 – Код функції ReadAtTime

```
quint32 HistoryManager :: readAtTime (
Core :: TimestampsToReturn timestampsToReturn,
QVector <quint64> reqTimes,
bool useSimpleBounds,
NodeId nodeId,
QQueue <DataValue> & dataValues)
```

Тут , reqTimes - масив тимчасових міток, для яких потрібно повернути історичні дані, useSimpleBounds - параметр, який визначає, чи використовувати чи інтерполяцію, якщо немає даних для деяких тимчасових міток.

Всі інші параметри аналогічні до функції ReadRaw (п. 3.5.1).

3.5.3 Функція Update

Функція Update дозволяє виконувати операцію модифікації історичних даних. Сигнатура функції наведена в лістингу 3.3.

Лістинг 3.3 – Код функції Update

```
quint32 HistoryManager :: update (
NodeId nodeId,
Core :: PerformInsertReplace performInsertReplace,
QQueue <DataValue> values,
QQueue <quint32> & results);
```

Перерахунок PerformInsertReplace описано в таблиці 3.2.

Таблиця 3.2 – Перерахунок PerformInsertReplace

Значення	Опис
INSERT_1	Проводиться запис в БД для виділених часових міток. Якщо для будь-якої мітки в БД вже існує запис, то вставка НЕ проводиться
REPLACE_2	Виконує заміну існуючих значень в БД для певних тимчасових міток. Якщо для будь часової мітки запису в БД не існує, то вставка НЕ проводиться
UPDATE_3	Виконує вставку або заміну значенm в БД. Якщо значення для визначеної часової мітки вже існує, то буде проведена заміна, якщо немає - вставка.

В ній values, тобто якщо вставка values [i] була проведена успішно, то results [i] містить «0», що відповідає успішному виконанню операції.

3.5.4 Функція Average

Average - функція, яка здійснює пошук середнього значення на певному часовому відрізку з деяким кроком. Програмний код сигнатури функції наведено в лістингу 3.4.

Лістинг 3.4 – Код функції Average

```
quint32 HistoryManager :: average (
NodeId nodeId,
quint64 startTime,
quint64 endTime,
quint64 interval,
double & value);
```

startTime і endTime - початок і кінець часового відрізку відповідно.

interval - інтервал розбиття відрізка.

Результат операції - value -повертається за посиланням.

3.6 Симулятор клієнта сервера історії

З метою тестування був розроблений симулятор клієнта сервера історії. Форма для відображення результатів функції History ReadRaw представлена на рисунку 3.9.

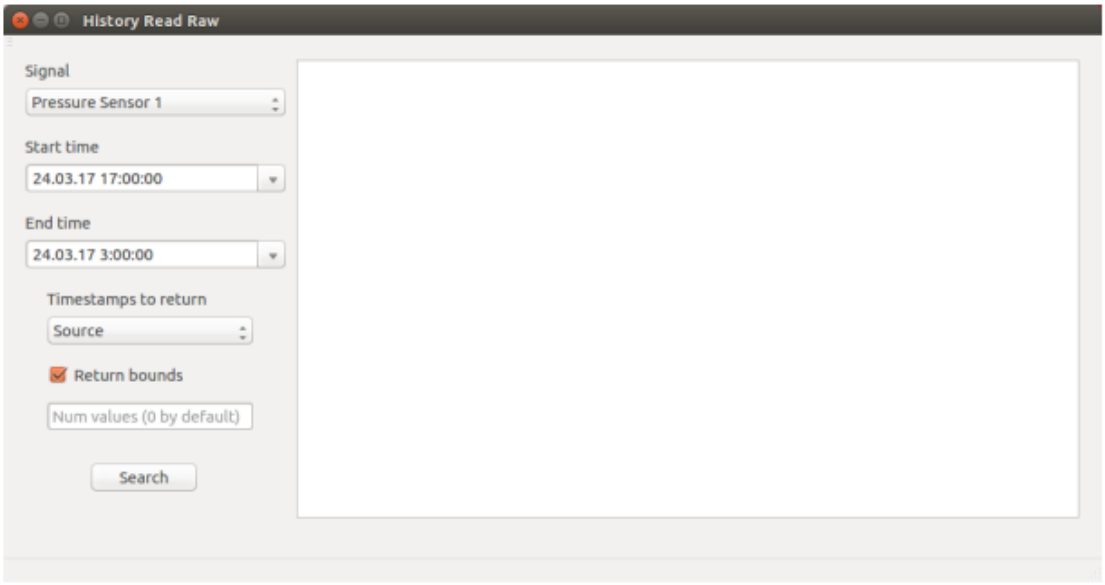

Рисунок 3.9 – Головна форма

У випадаючому списку, позначеному лейблом Signal розташовуються ідентифікатори сигналів - поле identifier структури NodeId (рисунок 3.10).

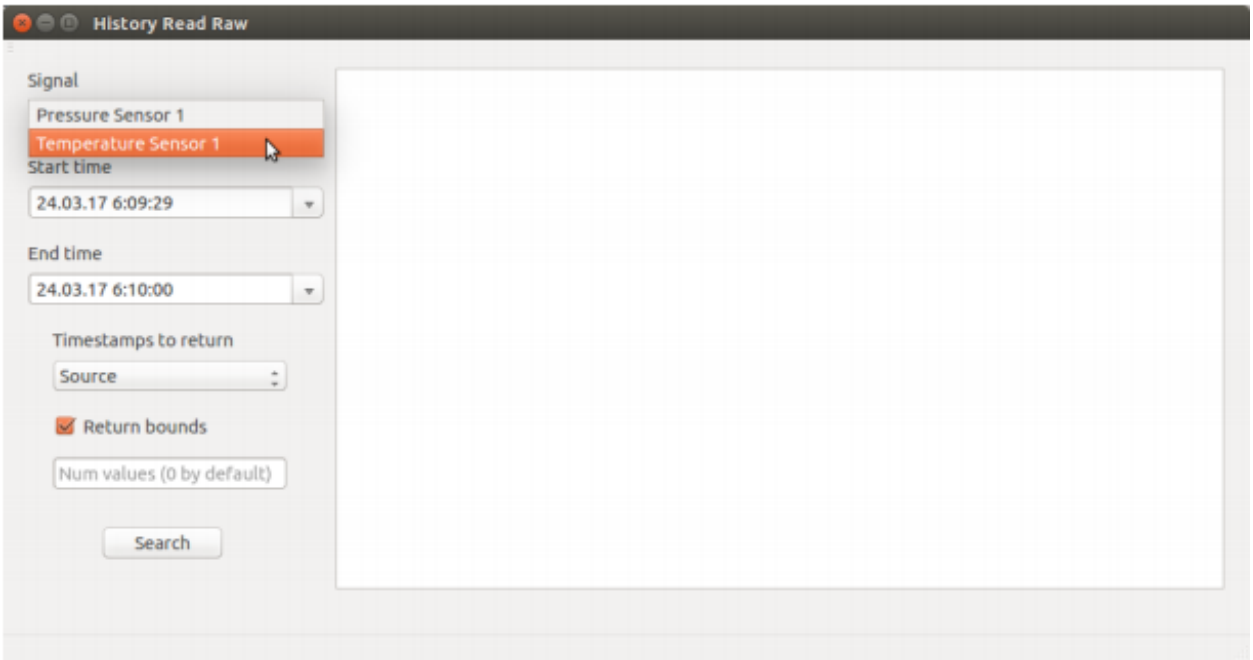

Рисунок 3.10 - Вибір сигналів

У полях Start time і End time задаються початок і кінець часового відрізку.

Відзначено параметр Return bounds - це значить, що кордони часового відрізку будуть включені в результати.

Num values - 0, тобто обмежень на кількість результатів накладено не буде.

У списку Tiemstamps to return можна вибрати відповідний параметр, представлений в таблиці 3.1 (рисунок 3.11).

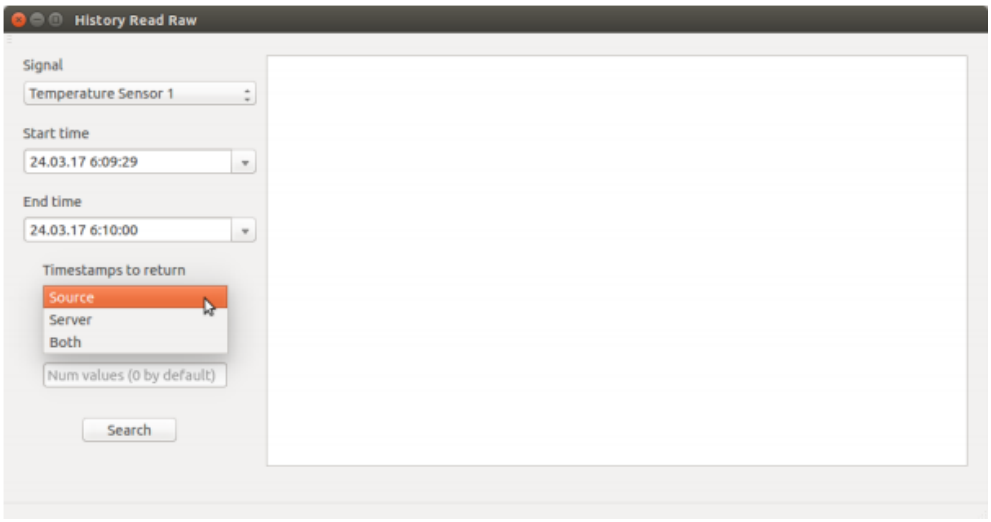

Рисунок 3.11 - Вибір параметра Timestamps to return

На рисунку 3.12 представлено результат виконання функції History ReadRaw.

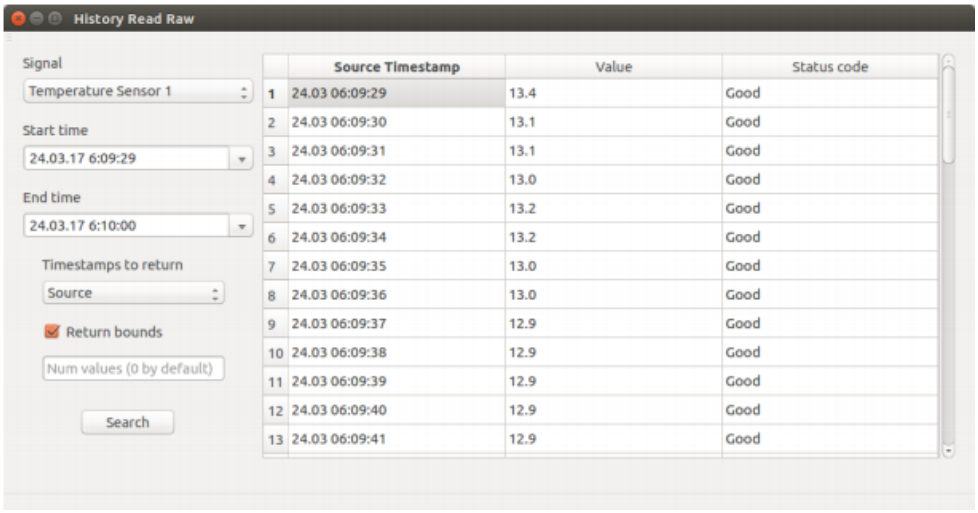

Рисунок 3.12 - Результат виконання функції History ReadRaw

3.7 Висновки до третього розділу

У третьому розділі приведена схема взаємодії сервера історії з компонентами SCADA-системи та архітектура серверу історії. Зображена програмна архітектура розроблюваного сервісу (діаграми класів всіх складових компонент), наведені особливості реалізації. Також показані скрін-шоти основних вікон сервісу та результати опрацювання часових рядів даних.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Режими праці і відпочинку при роботі з ЕОМ

Нервово-емоційне напруження, втома очей, гіподинамія, підвищене навантаження на кисті верхніх кінцівок та хребет – усе це негативний вплив на організм людини при роботі з комп'ютером.

В даному випадку для збереження здоров'я працюючих, запобігання професійним захворюванням і підтримки працездатності передбачаються внутрішньозмінні регламентовані перерви для відпочинку [28].

Основним нормативно-правовим документом, який регламентує всі питання, пов'язані із охороною праці, в т.ч. і при роботі з ЕОМ, є [29].

Вимоги при роботі з ЕОМ визначають Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин - ДСанПІН 3.3.2.007-98.

Внутрішньозмінні режими праці і відпочинку містять додаткові нетривалі перерви в періоди, що передують появі стомлення і зниження працездатності.

При виконанні робіт, що належать до різних видів трудової діяльності, за основну роботу з візуальними дисплейними терміналами (ВДТ) слід вважати таку, що займає не менше 50% робочого часу. Впродовж робочої зміни мають передбачатися:

– перерви для відпочинку і вживання їжі (обідні перерви);

– перерви для відпочинку і особистих потреб (згідно з трудовими нормами);

– додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

Тривалість обідньої перерви визначається чинним законодавством про працю і правилами внутрішнього трудового розпорядку підприємства, організації

чи установи. Як правило, тривалість такої перерви становить 40 – 60 хвилин. Тривалість та кількість інших внутрішньозмінних регламентованих перерв залежить від характеру трудової діяльності, напруженості і важності праці і визначається диференційовано для кожної професії.

За характером трудової діяльності розрізняють три професійні групи, згідно з діючим класифікатором професій [30]:

– розробники програм (інженери-програмісти) виконують роботу переважно з відеотерміналом та документацією при необхідності інтенсивного обміну інформацією з ЕОМ і високою частотою прийняття рішень;

– оператори ЕОМ виконують роботу, пов’язану з обліком інформації, одержаної з ВДТ за попереднім запитом, або тієї, що надходить з нього;

– оператор комп’ютерного набору виконує одноманітні за характером роботи з документацією та клавіатурою і нечастими нетривалими переключеннями погляду на екран дисплея, з введенням даних з високою швидкістю.

Правилами встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні в залежності від характеру праці:

– для розробників програм із застосуванням ЕОМ слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за ВДТ;

– для операторів із застосуванням ЕОМ слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

– для операторів комп’ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи.

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 години.

При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перервам при 8-годинній робочій зміні, а протягом останніх чотирьох годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин. Необхідно зазначити, що перерви під час роботи не повинні бути строго визначені за часом, а необхідно передбачати певний творчий індивідуальний підхід.

Для запобігання втоми під час деяких перерв доцільно виконувати спеціальні вправи, які наведені у ДСанПІН 3.3.2.007-98. Також ці санітарні правила визначають Відстань від екрана до ока фахівців, які працюють за комп'ютером.

Окрім ДНАОП та ДСанНіП, які регламентують вимоги безпеки та санітарно-гігієнічні вимоги до обладнання робочих місць користувачів ВДТ, є ще ціла низка нормативних актів загального призначення, які необхідно враховувати під час організації роботи користувачів ВДТ. Важливим нормативним актом є [31].

Гігієнічна класифікація праці необхідна для оцінки конкретних умов та характеру праці на робочих місцях. На основі такої оцінки приймаються рішення, спрямовані на запобігання або максимальне обмеження впливу несприятливих виробничих факторів.

4.2 Вплив електромагнітного імпульсу (ЕМІ) ядерного вибуху на елементи виробництва та заходи захисту

У воєнний час при застосуванні ядерної зброї проти України на електронно-обчислювальне обладнання в першу чергу буде впливати ЕМІ ядерного вибуху у вигляді короткого імпульсу, який вражає головним чином електричну та електронну апаратуру.

ЕМІ виникають в основному в результаті взаємодії гамма-випромінювання з атомами навколишнього середовища. На утворення ЕМІ йде невелика кількість

ядерної енергії, але він здатен викликати високі імпульси струмів та напруг в кабелях повітряних і підземних ліній зв'язку, сигналізації, управління, електропередачі, в антенах радіостанцій. Вплив ЕМІ може привести до згортання чутливих електронних та електричних елементів, зв'язаних з великими антенами чи відкритими дротами, а також до порушень в обчислювальних пристроях. Вплив ЕМІ необхідно враховувати для всіх електричних та електронних систем. Для найбільш важливих приладів треба використовувати засоби захисту і підвищувати їх стійкість до ЕМІ [28].

Особливістю ЕМІ, як вражаючого фактору є його здатність розповсюджуватись на десятки і сотні кілометрів в оточуючому середовищі. Тому ЕМІ може вплинути своєю дією на об'єкти, там де вибухова хвиля, світлове випромінювання, проникаюча радіація втрачають своє значення, як вражаючі фактори. При наземних та низьких повітряних вибухах в лініях зв'язку та електрозабезпечення виникають напруги, які можуть викликати пробій ізоляції провідників та кабелів відносно землі, пробій ізоляції елементів приладів підключених до повітряних і підземних ліній. Степінь враження залежить від наведеного імпульсу напруги чи струму і також електричної міцності обладнання.

Найбільш піддані впливу ЕМІ системи зв'язку, сигналізації, управління. Використані в цих системах кабелі та апаратура мають обмежену електричну міцність не більше 10кВ імпульсної напруги, тоді як наведені імпульси напруги від ЕМІ можуть перевищувати ці значення. Найбільш піддана впливу ЕМІ апаратура виконана на напівпровідниках та інтегральних схемах, працюючих на малих струмах і напругах, і значить відчутних до впливу зовнішніх електричних і магнітних кіл, в тому числі і елементи програмного засобу для управління процесом міграції віртуальних машин в обчислювальній хмарі. ЕМІ пробиває ізоляцію, спалює елементи електричних схем радіоапаратури, викликає коротке замикання в радіопристроях, іонізацію діелектриків, змінює або повністю стирає магнітний запис. Встановлено, що при дії ЕМІ на апаратуру найбільша напруга

наводиться на вході. В транзисторах відбувається така залежність: чим більший коефіцієнт підсилення транзистора, тим менша його електрична міцність.

ЕМІ пошкоджує також резистори, викликає іскріння в їх міжконтактних з'єднаннях і деяких областях провідної поверхні. Найбільшу небезпеку ЕМІ представляє для апаратури, яка встановлена в особливо міцних спорудах, які витримують великі тиски ударної хвилі. В цих спорудах апаратура не виходить з ладу від механічних пошкоджень, але ЕМІ може вивести з ладу всю незахищену апаратуру системи зв'язку, сигналізації і керування. Найбільших значень досягають напруги, які наводяться між кабелем і землею. Напруженість електромагнітного поля всередині споруди в деяких випадках недостатня для того, щоб вивести з ладу апаратуру, але такі поля в змозі викликати короткочасний збій роботи радіотехнічних пристроїв.

Розглянемо можливі шляхи рішення задачі захисту від ЕМІ. Ідеальним захистом від ЕМІ виявилося б повне укриття приміщення, в якому розміщена радіоелектронна апаратура, металевим екраном [32]. Водночас зрозуміло, що практично забезпечити такий захист у ряді випадків неможливо, тому що для роботи апаратури часто потрібно забезпечити її електричний зв'язок із зовнішніми пристроями. Тому використовуються менш надійні засоби захисту, такі, як струмопровідні сітки, або плівкові покриття для вікон, щільникові металеві конструкції для повітрезабірників і вентиляційних отворів і контактні пружинні прокладки, розміщувані по периметру дверей і люків.

Більш складною технічною проблемою рахується захист від проникнення ЕМІ в апаратуру через різноманітні кабельні входи. Радикальним рішенням даної проблеми міг би стати перехід від електричних мереж зв'язку до практично не схильних до впливу ЕМІ волоконно-оптичних. Проте заміна напівпровідникових приладів у всьому спектрі виконуваних ними функцій електронно-оптичними пристроями можлива тільки у віддаленому майбутньому. Тому в даний час в якості засобів захисту кабельних входів найбільші широко використовуються фільтри, у тому числі волоконні, а також

іскрові розрядники, металлоокисні варистори і високошвидкісні зенеровські діоди [32].

Всі ці засоби мають як переваги, так і недоліки. Так, ємнісно-індуктивні фільтри достатньо ефективні для захисту від ЕМІ малої інтенсивності, волоконні фільтри захищають у відносно вузькому діапазоні надвисоких частот. Іскрові розрядники мають значну інерційність й в основному придатні для захисту від перевантажень, що виникають під впливом напруг і струмів, що наводяться в обшивці літака, кожусі апаратури й оплетенні кабеля.

Металоокисні варистори є напівпровідниковими приладами, що різко підвищують свою провідність при високій напрузі. Проте, при застосуванні цих приладів у якості засобів захисту від ЕМІ варто враховувати їх недостатньо високу швидкодію і погіршення характеристик при кількаразовому впливі навантажень. Ці недоліки відсутні у високошвидкісних зенеровських діодах, дія яких заснована на різкій лавиноподібній зміні опору від високого значення практично до нуля, при перевищенні прикладеної до них напруги граничного розміру. Крім того на відміну від варисторів характеристики зенеровських діодів після багатократних впливів високих напруг і переключень режимів не погіршуються.

Найбільш раціональним підходом до проектування засобів захисту від ЕМІ кабельних входів є створення таких роз'ємів у конструкції яких передбачені спеціальні заходи, що забезпечують формування елементів фільтрів і установку вмонтованих зенеровських діодів. Подібне рішення сприяє одержанню дуже малих значень ємності й індуктивності, що необхідно для забезпечення захисту від імпульсів, що мають незначну тривалість і, отже, потужну високочастотну складову. Використання роз'ємів подібної конструкції дозволить вирішити проблему обмеження малогабаритних характеристик пристрою захисту.

Складність рішення задачі захисту від ЕМІ і висока вартість розроблених для цих цілей засобів і методів змушують піти по шляху їхнього вибіркового

застосування в особливо важливих системах зброї і військової техніки. Такий же шлях обраний і для захисту систем, що мають велику протяжність, керування і зв'язку. Проте основним методом рішення даної проблеми спеціалісти вважають створення так званих розподілених мереж зв'язку [28].

4.3 Висновки до четвертого розділу

В цьому розділі розглянуто важливі питання охорони праці та безпеки в надзвичайних ситуаціях, зокрема описані вимоги до режимів праці і відпочинку при роботі з ЕОМ, а також вплив ЕМІ ядерного вибуху на елементи виробництва та заходи захисту.

ВИСНОВКИ

В результаті виконання кваліфікаційної були реалізовані модулі сервера історії, що дозволяють зберігати історичні технологічні дані на сервері історії і забезпечувати до них доступ за протоколом OPC UA.

При цьому одержані такі основні результати:

– зроблено порівняльний аналіз існуючих сховищ даних на предмет використання в якості движка БД;

– зроблені дослідження продуктивності п'яти обраних сховищ даних.

– на базі специфікації OPC UA розроблений API- механізм зберігання даних з підтримкою OPC UA-операцій;

– реалізовано прототип сервера історії в частині функції збору оперативних технологічних даних, а також в частині підтримки операцій над історичними технологічними даними у відповідності зі специфікацією OPC UA.

Надалі планується додати можливість збереження і виконання OPC UA-операцій над подіями.

ПЕРЕЛІК ДЖЕРЕЛ

1. Святний В.А., Бровкіна Д.Ю. Сучасні тенденції в автоматизації промислових комплексів // Системні дослідження та інформаційні технології. – 2016. – № 1. – С. 32-39.

2. Денисенко В. В. Компьютерное управление технологическим процессом, экспериментом, оборудованием. – М.: Горячая линия-Телеком, 2009. – 608 с.

3. OPCFoundation. The Industrial Interoperability Standard. [Електронний ресурс] – Режим доступу: www.opcfoundation.org (дата звернення 21.11.2020).

4. Энциклопедия АСУ ТП. [Електронний ресурс] – Режим доступу: https://www.bookasutp.ru/Chapter9_2_4.aspx (дата звернення 22.11.2020).

5. Специфікація OPC UA. [Електронний ресурс] – Режим доступу: https://studfile.net/preview/7786751/page:67/ (дата звернення 22.11.2020).

6. OPC Unified Architecture Specification. Part 1: Overview and Concepts OPC Foundation. [Електронний ресурс] – Режим доступу: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-1-overview-and-concepts (дата звернення 23.11.2020).

7. Матвейкин В.Г., Фролов С.В., Шехтман М.Б. Применение SCADA-систем при автоматизации технологических процессов. - М: Машиностроение, 2000. -176 с.

8. Кангин В. В., Кангин М. В., Ямолдинов Д. Н. Разработка SCADA-систем. Учебное пособие. – М.: Инфра-Инженерия, 2019. – 564 с.

9. Елизаров И.А. Интегрированные системы проектирования и управления: SCADA-системы. — Тамбов : ТГТУ, 2015. — 160 с.

10. Программные системы и инструменты: Тематический сборник/ Под ред. Королева Л.Н. - М: Издательский отдел факультета ВМиК МГУ (МАКС Пресс). 2011. - № 12.-268 с.

11. Деменков Н.П. SCADA-системы как инструмент проектирования АСУ ТП: Учеб. пособие. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. — 328 с.

12. Бобух А. О. Автоматизовані системи керування технологічними процесами: Навч. посібник. — Харків: ХНАМГ, 2006. — 185 с

13. Best Time Series Databases Software. [Електронний ресурс] – Режим доступу: https:// https://www.g2.com/categories/time-series-databases/ (дата звернення 21.11.2020).

14. BangDB – Performance Comparison with LevelDB and BerkeleyDB. [Електронний ресурс] – Режим доступу: http://bangdb.com/resources_download php ?name=Bangdb-Embedded-PerfComp.pdf (дата звернення 26.11.2020).

15. BangDB vs LevelDB – Performance Comparison. [Електронний ресурс] – Режим доступу: http://www.iqlect.com/blog/2016/07/12/bangdb-vs-leveldb-performance-comparison/ (дата звернення 26.11.2020).

16. Embedded Key-Value Store Performance Benchmark. [Електронний ресурс] – Режим доступу: http://www.storagereview.com/embedded_keyvalue _store _performance_benchmark (дата звернення 26.11.2020).

17. HyperLevelDB Performance Benchmarks. [Електронний ресурс] – Режим доступу: http://hackingdistributed.com/2013/06/17/hyperleveldb/ (дата звернення 26.11.2020).

18. LevelDB Benchmarks Електронний ресурс] – Режим доступу: https://github.com/google/leveldb/blob/master/doc/ benchmark.html (дата звернення 26.11.2020).

19. Performance Data For LevelDB, Berkley DB And BangDB For Random Operations [Електронний ресурс] – Режим доступу: http://highscalability.com/blog /2012/11/29/performance- data-forleveldb-berkley-db-and-bangdb-for-rando.html (дата звернення 26.11.2020).

20. Pragma statements supported by SQLite [Електронний ресурс] – Режим доступу: https://www.sqlite.org/pragma.html (дата звернення 26.11.2020).

21. Яремцьо І.І. Аналіз сховищ даних та СУБД для роботи з часовими рядами // Інформаційні моделі, системи та технології: Праці VIII наук.-техн. конф. (Тернопіль, 09-10 грудня 2020 р.) Тернопіль, 2020. – С. 72.

22. Memory Requirements GitBook. Sophia - modern transactional key-value/row storage library. [Електронний ресурс] – Режим доступу: http://sophia.systems/v2.2/admin/memory_requirements.html (дата звернення 27.11.2020).

23. OPC Unified Architecture Specification. Part 11: Historical Access [Електронний ресурс] – Режим доступу: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-11-historical-access дата звернення 29.11.2020).

24. Performance Data For LevelDB, Berkley DB And BangDB For Random Operations. [Електронний ресурс] – Режим доступу: http://highscalability.com/blog /2012/11/29/ performance-data-forleveldb-berkley-db-and-bangdb-for-rando.html (дата звернення 29.11.2020).

25. OPC Unified Architecture Specification. Part 4: Historical Access [Електронний ресурс] – Режим доступу: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-4-historical-access (дата звернення 29.11.2020).

26. Интеллектуальные системы хранения данных в АСУ ТП. [Електронний ресурс] – Режим доступу: https://www.cta.ru/cms/f/441322.pdf (дата звернення 29.11.2020).

27. Замятин А.В., Острасть П.М., Телицын Е.А., Тренькаев В.Н., Яновский В.Д. Высокопроизводительный сервер истории системы диспетчерского управления и сбора данных //Промышленные АСУ и контроллеры. 2017. № 9. С. 20-28.

28. Зеркалов Д.В. Безпека життєдіяльності та основи охорони праці. Навчальний посібник. К.: «Основа». 2016. 267 с.

29. Закон України «Про охорону праці». [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12 - (дата звертання 26.11.2020).

30. Класифікатор професій ДК 003:2010/ [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/rada/show/va327609-10 - (дата звертання 26.11.2020).

31. Гігієнічна класифікація умов праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу. – К.: МОЗ України, 1998. – 34 с.

32. Сакевич В.Ф., Поліщук О.В. Цивільна оборона. Теоретичні основи. Навчальний посібник. — Вінниця : ВНТУ, — 2009. — 136 с.

ДОДАТКИ

ДОДАТОК А
Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

VII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ »

9–10 грудня 2020 року

ТЕРНОПІЛЬ
2020

УДК 004.65
І.І. Яремцьо
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

АНАЛІЗ СХОВИЩ ДАНИХ ТА СУБД ДЛЯ РОБОТИ З ЧАСОВИМИ РЯДАМИ

UDC004.65
I.I. Yaremtso

ANALYSIS DATA WAREHOUSE AND DBMS TO WORK WITH TIME SERIES

Для аналізу обмежимося вбудованими СУБД та сховищами типу «ключ-значення». Вся інформація бралася з офіційної документації.

LevelDB. Це сховище «ключ-значення» від Google. Переваги: ключі та значення представляються довільним набором байт; дані зберігаються у відсортованому до ключа вигляді; розробнику надається можливість перевизначити функцію порівняння, щоб задати потрібний порядок сортування; простий інтерфейс для базових операцій – Put (key, value), Get (key), Delete (key); кілька операцій можна виконати як одну (batch); можливість зробити знімок БД; стиснення даних через Snappy (бібліотеку для компресії даних від Google); підтримка обходу ітератором в прямому і зворотному порядку. Обмеження: не підтримує реляційну модель даних, індекси і мову SQL; тільки один процес може мати доступ до БД в певний момент часу.

HyperLevelDB. Це форк LevelDB, в якому були збережені всі її переваги і покращено багатопотокову взаємодію з БД. LevelDB влаштована таким чином, що в певний момент часу тільки один потік може щось записувати в БД, а якщо таких потоків декілька, то для їх синхронізації використовується м'ютекс (семафор) і запис проводиться за принципом черги (FIFO – first in, first out – першим увійшов, першим вийшов): доступ до БД отримує потік, що знаходиться в «голові» черги, а всі інші чекають. Це гарантує, що одночасна робота цих потоків з БД буде безпечна і цілісність даних не порушиться. HyperLevelDB дозволяє всім потокам одночасно записувати дані в БД, тим самим роблячи паралельну роботу більш ефективною, а синхронізація досягається за рахунок іншого механізму.

BangDB. Сховище «ключ-значення», котре може перебувати і в оперативній пам'яті, і на диску. Основна перевага – висока продуктивність при багатопотоковій роботі. Дана БД підтримує ACID-транзакції і використовує компоненти, з допомогою яких досягається оптимізація ресурсів, що гарантує високу продуктивність і низьку затримку. BangDB реалізує свій буферний кеш, систему управління буфером, механізм управління пам'яттю і т.д., також має багатий API, який крім стандартних CRUD-операцій забезпечує гнучкий інтерфейс запитів (з використанням B-дерев). Дане сховище може бути серверним або вбудованим.

Sophia. Сучасне сховище «ключ-значення» з підтримкою транзакцій, яке може розташовуватися як в оперативній пам'яті, так і на диску. Розроблене для забезпечення його оптимальної роботи без деградації в часі і гарантує складність порядку 0 (1) в найгіршому для операцій читання і запису. Переваги: підтримка ACID-транзакцій; управління паралельним доступом за допомогою багатоверсійності; оптимістичне блокування; стиснення даних; простий інтерфейс; не має зовнішніх залежностей.

SQLite. Відкрита вбудована реляційна СУБД. Переваги: вбудована (звернення до БД відбувається не за рахунок клієнт-серверної взаємодії, а як виклик бібліотечної функції, що збільшує продуктивність); БД є одним файлом; підтримка ACID-транзакцій; підтримка багатопотокового читання; підтримка динамічно типізованих типів даних; підтримка SQL. Недоліки: відсутність призначеного для користувача управління; бідні можливості додаткового налаштування.