МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Кваліфікаційна наукова праця на
правах рукопису

ВОЛОХ БОГДАН ЮРІЙОВИЧ

УДК 004.6; 004.8; 004.9;

ДИСЕРТАЦІЯ

ІНТЕЛЕКТУАЛЬНЕ КЕШУВАННЯ ДАНИХ В СИСТЕМІ ПІДТРИМКИ ПРОЦЕСУ ВІДНОВЛЕННЯ ПОШКОДЖЕНИХ ОБ'ЄКТІВ НЕРУХОМОСТІ

126 – Інформаційні системи та технології
12 – Інформаційні технології

Подається на здобуття наукового ступеня **доктора філософії**

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ Б. Ю. Волох

Науковий керівник Єременко Богдан Михайлович**,** кандидат технічних наук, доцент кафедри інформаційних технологій проєктування та прикладної математики Київського національного університету будівництва і архітектури

Київ – 2025

АНОТАЦІЯ

Волох Б.Ю. *Інтелектуальне кешування даних в системі підтримки процесу відновлення пошкоджених об'єктів нерухомості*. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 126 «Інформаційні системи та технології». – Київський національний університет будівництва і архітектури. – Київ, 2025.

Дисертацію присвячено вирішенню науково-практичної задачі підвищення ефективності обробки і зберігання даних в інформаційній інфраструктурі галузі архітектури і будівництва.

Наукова новизна одержаних результатів полягає в розробці методу інтелектуального управління кешем в інформаційній системі підтримки процесу відновлення пошкоджених об'єктів нерухомості, що вирішує ключові задачі кешування, базується на моделях машинного навчання і здатен забезпечити ефективний доступ до актуально важливих даних з урахуванням суттєвих ознак користувачів і пошкоджених об'єктів нерухомості.

вперше розроблено:

– гібридний метод інтелектуального управління кешем, що вирішує ключові задачі кешування, базується на моделях машинного навчання та враховує суттєві ознаки користувачів і пошкоджених об'єктів нерухомості;

удосконалено:

– модель інформаційної системи підтримки процесу відновлення пошкоджених об'єктів нерухомості шляхом інтеграції модуля інтелектуального кешування, що підвищує продуктивність системи і забезпечує швидкий доступ до актуально важливих даних;

– процес інформаційної підтримки користувачів системи підтримки процесу відновлення пошкоджених об'єктів нерухомості шляхом розширення колекцій документів для нереляційної бази даних, що призначена для зберігання даних про

пошкоджені об’єкти нерухомості, історії запитів користувачів і статистику пошкоджень;

набули подальшого розвитку:

– підходи до обробки і зберігання даних в інформаційній інфраструктурі галузі архітектури і будівництва через поєднання технологій інтелектуального кешування з нереляційними базами даних.

Основний зміст дисертаційної роботи

Запропоновано гібридний метод інтелектуального управління кешем, орієнтований на інформаційні високонавантажені системи з динамічним характером запитів, зокрема на систему підтримки процесу відновлення пошкоджених об’єктів нерухомості. Метод охоплює три ключові задачі: оцінка доцільності додавання об’єкта до кешу, вибір об’єкта для видалення у разі заповнення кешу та періодичну інвалідацію неактуальних об’єктів у кеші. Для вирішення цих задач використано поєднання моделей машинного навчання: табличного Q-Learning з механізмом відкладеного асинхронного оновлення значень моделі та логістичної регресії з механізмом відкладеного асинхронного накопичення буфера навчальних прикладів для подальшого навчання моделі та періодичного оновлення її параметрів. Проведено експериментальну оцінку ефективності методу, яка підтвердила перевагу інтелектуального підходу порівняно з існуючими методами кешування LRU та LFU, зокрема в умовах обмеженого розміру вхідних даних. Запропонований метод має високий потенціал для використання в інших інформаційних системах завдяки можливості адаптації вектора ознак моделей машинного навчання відповідно до характеристик об’єктів і контексту їх використання.

За результатами дослідження розроблено:

– гібридний метод інтелектуального управління кешем, що вирішує ключові задачі кешування, базується на моделях машинного навчання та враховує суттєві ознаки користувачів і пошкоджених об’єктів нерухомості;

– систему експериментального тестування ефективності гібридного методу інтелектуального управління кешем, що включає модулі для симуляції запитів,

реалізації логіки роботи методу, а також взаємодії з моделями машинного навчання, технологією кешування та нереляційною базою даних.

У ***першому розділі*** здійснено аналіз основних особливостей організації та виконання процесу будівельно-технічної експертизи в сучасних умовах, зокрема після початку масштабних руйнувань інфраструктури України внаслідок бойових дій; окреслено основні етапи роботи експертів; визначено та описано проблеми, пов'язані з неефективною організацією зберігання, обробки й повторного використання даних у цьому процесі; сформульовано та описано ключові проблеми роботи з даними в інформаційних високонавантажених системах, визначено вплив цих проблем на продуктивність і складність їх вирішення; запропоновано підходи до вирішення виявлених проблем роботи з даними шляхом впровадження нереляційних баз даних і технології кешування як ефективних рішень, що здатні забезпечити гнучкість, масштабованість і швидкий доступ до даних; наведено порівняльний аналіз реляційних та нереляційних баз даних для кращого обґрунтування доцільності використання нереляційних баз даних; наведено загальну інформацію про технологію кешування; описано потенційні переваги для роботи з даними в разі використання цієї технології для будівельно-технічної експертизи та інформаційних високонавантажених систем; проведений аналіз слугує основою для формування вимог до роботи з даними в інформаційній системі підтримки процесу відновлення пошкоджених об'єктів нерухомості; виконано постановку задачі.

У ***другому розділі*** визначено основні функції, структурні та технічні вимоги для бази даних системи підтримки процесу відновлення пошкоджених об'єктів нерухомості; описано основні типи запитів на читання та запис, які виконуватимуть користувачі цієї системи з різними рівнями доступу; обґрунтовано пріоритетні властивості бази даних системи відповідно до CAP-теореми; визначено ключові критерії, за якими оцінюється доцільність використання різних типів і технологій нереляційних баз даних для потреб системи; обґрунтовано доцільність використання СУБД MongoDB як бази даних для системи підтримки процесу відновлення пошкоджених об'єктів нерухомості на основі порівняльного аналізу з іншими типами та технологіями нереляційних баз даних; спроектовано структуру основних

колекцій документів бази даних MongoDB для системи, зокрема: «Пошкоджені будівлі», «Міська статистика», «Користувачі» та «Історія запитів»; побудовано схему логічних взаємозв'язків між цими колекціями як основу для розробки гібридного методу інтелектуального управління кешем, що дозволяє суттєво підвищити ефективність роботи з даними; обґрунтовано доцільність використання інтелектуального кешування як засіб подолання обмежень існуючих методів кешування; описано роль кешування в сучасних інформаційних високонавантажених системах.

У ***третьому розділі*** розроблено гібридний метод інтелектуального управління кешем, який вирішує три ключові задачі: оцінка доцільності додавання об'єкта до кешу, вибір об'єкта для видалення з заповненого кешу та періодична інвалідація неактуальних об'єктів у кеші; подано визначення методу управління кешем як основи для реалізації гібридного підходу; обґрунтовано застосування моделей машинного навчання – табличного Q-Learning та логістичної регресії для вирішення відповідних задач запропонованого методу інтелектуального управління кешем; запропоновано представлення стану об'єкта у вигляді вектора ознак; спроектовано додаткові колекції документів для бази даних системи підтримки процесу відновлення пошкоджених об'єктів нерухомості MongoDB, необхідні для коректної роботи запропонованих моделей машинного навчання; обґрунтовано доцільність використання Redis як основної технології кешування для системи; описано алгоритм роботи гібридного методу інтелектуального управління кешем; показано фінальну блок-схему його роботи із застосуванням моделей машинного навчання табличного Q-Learning і логістичної регресії, технології нереляційної бази даних MongoDB і технології кешування Redis; представлено оновлену модель системи з інтегрованим модулем інтелектуального кешування, що реалізовує розроблений гібридний метод.

У ***четвертому розділі*** описано структуру та основні функції системи експериментального тестування ефективності запропонованого гібридного методу інтелектуального управління кешем; наведено детальний опис компонентів системи, зокрема бази даних MongoDB, кеш Redis, двох моделей машинного навчання –

табличного Q-Learning і логістичної регресії та модуля керування кешем, що реалізовує алгоритм роботи гібридного методу; реалізовано повноцінну програмну архітектуру системи, включно з агентами, буферами, оновленням моделей у фоновому асинхронному режимі та синхронізацією з базою даних MongoDB і кешем Redis; продемонстровано результати роботи системи у двох типах експериментів: аналіз частки кеш-хітів після одного запуску та серії запусків з накопичувальним навчанням моделей з фіксованим набором даних; у кожному експерименті проведено порівняння ефективності запропонованого гібридного методу управління кешем з найпоширенішими існуючими методами кешування: LRU та LFU; візуалізовано результати у вигляді графіків і рисунків, що підтверджують перевагу гібридного методу за умов обмеженого розміру кешу та кількості запитів.

Ключові слова: база даних, будівельно-технічна експертиза, інтелектуальне кешування, інформаційна високонавантажена система, машинне навчання, нереляційна база даних, технологія кешування, штучний інтелект.

ABSTRACT

Volokh B. *Intelligent Data Caching for the System Supporting the Process of Damaged Buildings Restoration.* – Qualifying scientific work presented as a manuscript.

Dissertation for obtaining the scientific degree of Doctor of Philosophy in specialty 126 "Information systems and technologies". - Kyiv National University of Construction and Architecture. - Kyiv, 2025.

The dissertation is devoted to solving the scientific and practical problem of improving the efficiency of data processing and storage in the information infrastructure of the architecture and construction industry.

The scientific novelty of the obtained results. The scientific novelty of the dissertation consists in the fact that:

developed for the first time:

– the hybrid method of intelligent cache management that solves the key caching tasks, is based on machine learning models, and takes into account the essential characteristics of users and damaged real estate objects;

improved:

– the model of the information system supporting the process of damaged buildings restoration by integrating an intelligent caching module, which improves a performance of the system and provides efficient access to high-priority data;

– the process of providing information support to the users of the system supporting the process of damaged buildings restoration by expanding the collections of documents for a non-relational database designed to store data about damaged real estate objects, history of user requests, and damage statistics;

further advance:

– approaches to data processing and storage in the information infrastructure of the architecture and construction industry by combining intelligent caching technologies with non-relational databases.

Main content of the dissertation

The hybrid method of intelligent cache management is presented, focused on highly loaded information systems with dynamically changing requests, in particular on the system supporting the process of damaged buildings restoration. The method covers three key tasks: assessing the expediency of adding an object to the cache, selecting an object for deletion when the cache is full, and periodically invalidating obsolete objects in the cache. To solve these tasks, a combination of machine learning models is used: table-based Q-Learning with a mechanism for deferred asynchronous updating of model values and logistic regression with a mechanism for deferred asynchronous accumulation of a buffer of training examples for further training of the model and periodic updating of its parameters. An experimental evaluation of the method's effectiveness was performed, which confirmed the advantage of the intelligent approach over existing LRU and LFU caching methods, particularly in conditions of limited input data size. The proposed method has high potential for usage in other information systems due to the ability to adapt the feature vector of machine learning models to the features of objects and the context of their usage.

Based on the results of the study, the following have been developed:

– the hybrid method of intelligent cache management that solves the key caching tasks, is based on machine learning models, and takes into account the essential characteristics of users and damaged real estate objects;

– the system for experimental testing of the effectiveness of the hybrid intelligent cache management method, which includes the modules for simulating requests, implementing logic of the method, and interacting with the machine learning models, the caching technology, and the non-relational database.

In the first chapter, the main features of organizing and performing the process of building-technical expertise in modern conditions are analyzed, with particular attention to the challenges that have emerged after the beginning of large-scale infrastructure destruction in Ukraine as a result of military operations; the key stages of expert activities are outlined; problems related to inefficient organization of data storage, processing, and reuse in this process are identified and described; the key problems of working with data in high-load information systems are formulated and described, and the impact of these

problems on performance and the complexity of their solution are determined; approaches to solving identified data processing issues have been proposed through the integration of non-relational databases and caching technology as effective solutions capable of ensuring flexibility, scalability, and efficient access to data; the comparative analysis of relational and non-relational databases is provided to better substantiate the appropriateness of using non-relational databases; general information about caching technology is provided; the potential advantages for working with data when using this technology for construction and technical expertise and high-load information systems are described; the conducted analysis serves as the basis for forming requirements for data processing in the information system supporting the process of damaged buildings restoration; the task is set.

In the second chapter, the main functions, structural and technical requirements for the database of the system supporting the process of damaged buildings restoration are defined; the primary types of read and write queries expected to be performed by users with varying levels of access are described; the priority properties of the database are substantiated in accordance with the CAP theorem; the key criteria for assessment the feasibility of using different types and technologies of non-relational databases for the needs of the system are defined; the expediency of using MongoDB as database for the system supporting the process of damaged buildings restoration based on a comparative analysis with other types and technologies of non-relational databases is substantiated; the structure of the main MongoDB document collections for the system is designed, including “Damaged Buildings,” “Urban Statistics,” “Users,” and “Query History”; the scheme of logical relationships between these collections is built as a basis for the development of a hybrid method of intelligent cache management, which can significantly increase the efficiency of data processing; the expediency of using intelligent caching as a means of overcoming the limitations of existing caching methods is substantiated; the role of caching in modern information high-load systems is described.

In the third chapter, the hybrid method of intelligent cache management is developed, which solves three key tasks: assessing the feasibility of adding an object to the cache, selecting an object to remove from the full cache, and periodically invalidating

obsolete objects in the cache; the definition of the cache management method as a basis for the implementation of a hybrid approach is presented; the application of machine learning models - table-based Q-Learning and logistic regression for solving the relevant tasks of the proposed method of intelligent cache management is substantiated; a feature vector representation of the object state is proposed; additional collections of documents for the MongoDB database of the system supporting the process of damaged buildings restoration were designed, which are necessary for the correct functioning of the proposed machine learning models; the expediency of using Redis as the main caching technology for the system is substantiated; the algorithm of the hybrid method of intelligent cache management is described in detail; a final block diagram of its operation is presented, illustrating the use of table-based Q-Learning and logistic regression machine learning models, the non-relational MongoDB database, and the Redis caching technology; an updated model of the system with the integrated module of intelligent caching that implements the developed hybrid method is presented.

In the fourth chapter, the structure and core functionalities of the experimental testing system for evaluating the effectiveness of the proposed hybrid method of intelligent cache management are described; a detailed description of the system components, including the MongoDB database, the Redis cache, two machine learning models - table-based Q-Learning and logistic regression, and the cache management module that implements the hybrid method algorithm is presented; a complete software architecture of the system was implemented, including agents, buffers, updating models in the background asynchronously and synchronization with the MongoDB database and Redis cache; the results of the system are demonstrated in two types of experiments: analysis of the proportion of cache hits after one run and a series of runs with accumulative training of models with a fixed data set; in each experiment, the comparison of the efficiency of the proposed hybrid cache management method with the most common existing caching methods LRU and LFU is made; the results are visualized in the form of graphs and figures, which confirm the advantage of the hybrid method under conditions of limited cache size and number of requests.

Keywords: database, building-technical expertise, intelligent caching, high-load information system, machine learning, non-relational database, caching technology, artificial intelligence.

СПИСОК ОПУБЛІКОВАНИХ НАУКОВИХ ПРАЦЬ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Статті у наукових періодичних виданнях інших держав та у виданнях України, які включено до міжнародних наукометричних баз

1. **Волох Б.** Інтелектуальне кешування у високонавантажених системах як відповідь на обмеження класичних методів керування кешем. *Шляхи підвищення* ефективності *будівництва.* Київ, 2023, № 3(52). С. 227–236, https://doi.org/10.32347/2707-501x.2023.52(3).227-236. (Фахове видання категорії «Б»)

2. Бугров А. А., **Волох Б. Ю.**, Босенко І. В., Теренчук С. А. Система підтримки процесу відновлення об’єктів нерухомості: обробка і збереження даних. *Управління розвитком складних систем.* Київ, 2024. № 60. С. 136 –145, https://doi.org/10.32347/2412-9933.2024.60.136-145. (Фахове видання категорії «Б»)

Статті у періодичних наукових виданнях, проіндексованих у базах даних Web of Science Core Collection та/або Scopus

3. Pasko, R., Klochko, A., Petrochenko, O., Ploskyi, V., **Volokh B.,** Intelligent Supporting System to Building-Technical Expertise Process, *CEUR Workshop Proceedings, 3rd International Workshop on Information Technologies: Theoretical and* **Applied** *Problems, ITTAP 2023,* Ternopil, Ukraine, 22-24 November 2023, Vol. 3628, P. 702 – 708. URL: https://ceur-ws.org/Vol-3628/short9.pdf (**Scopus**, ISSN: 1613-0073)

4. Yehorov, O., **Volokh, B.,** Bosenko, I., Yeremenko, B., Terenchuk, S., Modeling of highly loaded distributed system supporting the process of assessing the technical condition of objects, *CEUR Workshop Proceedings, 4th International Workshop on Information* **Technologies***: Theoretical and Applied Problems, ITTAP 2024*, Ternopil, Ukraine, Opole, Poland, 23-25 October 2024, Vol. 3896, P. 173 – 185. URL: https://ceur-ws.org/Vol-3896/paper10.pdf (**Scopus**, ISSN 1613-0073)

Наукові праці, що представлені як тези доповіді у міжнародних науково-технічних конференціях

5. **Volokh, B.,** Bosenko, I., Pasko, R., Molodid, O., Zapryvoda, V., Terenchuk, S. Modeling the Process of Assessing the Technical Condition of Damaged Real Estate Objects. *2023 IEEE International Conference on Smart Information Systems and Technologies (SIST)*, Astana, Kazakhstan, 4–6 May 2023 DOI: https://doi.org/10.1109/sist58284.2023.10223547 (**Scopus**, ISBN: 979-835033504-0)

6. Terenchuk, S., **Pasko**, R., Bosenko, I., Buhrov, A., Yaschenko, A., **Volokh B.** Ontology **Formation** of Support System for Restoration of Buildings, Property and Infrastructure Objects. *2023 IEEE 4th KhPI Week on Advanced Technology (KhPIWeek)*, Kharkiv, Ukraine, 2–6 October 2023. DOI: https://doi.org/10.1109/khpiweek61412.2023.10313006 (**Scopus**, ISBN: 979-835039553-2)

Тези доповідей наукових та науково-практичних конференцій

7. **Bosenko** I., **Volokh B., Pasko** R. Database modeling for the infocommunication support system for rebuilding of damaged real estate objects. *Міжнародна науково-практична конференція "Перспективні напрямки розвитку науки, освіти, технологій та суспільства: теорія і практика"*, м. Полтава. 19 квітня 2023. С.35-39. URL: http://www.economics.in.ua/2023/04/19-2023.html

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

БТЕ – будівельно-технічна експертиза;

СППВПОН – система підтримки процесу відновлення пошкоджених об’єктів нерухомості;

LRU – Least Recently Used (Найменш нещодавно використаний);

LFU – Least Frequently Used (Найменш часто використаний);

JSON – JavaScript Object Notation (Нотація об’єктів JavaScript);

SQL – Structured Query Language (Мова структурованих запитів);

NoSQL – Not Only SQL (Не лише SQL).

ВСТУП

Актуальність теми. Актуальність теми дисертаційного дослідження обумовлена необхідністю швидкого відновлення об’єктів нерухомості, пошкоджених внаслідок повномасштабного вторгнення рф на територію України з 24 лютого 2022 року. Це, своєю чергою, потребує підвищення ефективності обробки великої кількості різнорідних даних в інформаційній інфраструктурі галузі архітектури і будівництва, зокрема в умовах високого навантаження.

Зв’язок роботи з науковими програмами, планами, темами. Наукова робота відповідає напрямку, визначеному у Законі України «Про схвалення Концепції розвитку штучного інтелекту в Україні» (2020 р.), відповідає цілям Плану повоєнного відновлення України (2023-2032 рр.) і може бути інтегрована у програму «Цифрова Україна». Дисертація відповідає паспорту спеціальності 126 – Інформаційні системи та технології.

Дисертаційне дослідження було виконано на кафедрі інформаційних технологій проектування та прикладної математики факультету автоматизації і інформаційних технологій Київського національного університету будівництва і архітектури та пов’язане з науковим дослідженням в рамках комплексної теми «Інформаційна технологія оцінки технічного стану об’єктів будівництва в умовах невизначеності із застосуванням методики кластеризації» (Державний реєстраційний номер 0124U004532).

Результати дослідження впроваджено в навчальний процес в межах Київського національного університету будівництва і архітектури.

Зокрема, в навчальні програми дисциплін:

– «Інформаційні технології представлення обробки та розпізнавання зображень» для магістрів спеціальності «Інформаційні системи та технології» (ІСТм-23) кафедри інформаційних технологій проектування та прикладної математики.

– «Теорія алгоритмів» для магістрів спеціальності «Інформаційні системи та технології. Штучний інтелект. Когнітивні технології» (ІСТ-ШІКТм-23, ІСТм-23)

кафедри управління проєктів і кафедри інформаційних технологій проектування та прикладної математики;

– «Прикладна теорія графів» для магістрів спеціальності «Автоматизація та комп’ютерно-інтегровані технології» (АКІТм-24) кафедри автоматизації технологічних процесів.

Дисертація містить наукові положення, нові науково обґрунтовані теоретичні результати проведених досліджень, які мають істотне значення для галузі знань 12 – Інформаційні технології.

Мета роботи полягає в розробці методу інтелектуального управління кешем в інформаційній системі підтримки процесу відновлення пошкоджених об'єктів нерухомості, що базується на моделях машинного навчання і здатен забезпечити ефективний доступ до актуально важливих даних з урахуванням суттєвих ознак користувачів і об’єктів.

Для досягнення цієї мети пропонується розв’язати наступні задачі:

1. Проаналізувати проблеми роботи з даними у сучасному процесі будівельно-технічної експертизи та інформаційних високонавантажених системах, а також існуючі методи кешування;

2. Спроектувати базу даних системи підтримки процесу відновлення пошкоджених об'єктів нерухомості;

3. Розробити гібридний метод інтелектуального управління кешем системи підтримки процесу відновлення пошкоджених об’єктів;

4. Оцінити ефективність запропонованого гібридного методу інтелектуального управління кешем.

Методи дослідження, що використовувалися під час роботи: методи системного аналізу, порівняння, формалізація, моделювання, експеримент.

Об’єкт дослідження – процес роботи з даними під час виконання будівельно-технічної експертизи та у інформаційних високонавантажених системах.

Предмет дослідження – моделі машинного навчання, методи управління кешем і технології кешування.

Обґрунтованість та достовірність отриманих результатів. Теоретичні результати дисертаційного дослідження аспіранта підтверджено і засвідчено актами про апробації.

Практичне значення одержаних результатів полягає у тому що:

– запропонована структурна модель документів для нереляційних баз даних, оптимізована для зберігання інформації про пошкоджені об’єкти нерухомості, історію запитів до них від користувачів і статистику пошкоджень надає можливість підвищити зручність та ефективність роботи експертів;

– спроектований гібридний метод інтелектуального управління кешем дозволяє підвищити ефективність роботи користувачів з актуально важливими даними.

Основний зміст дисертаційної роботи

Запропоновано гібридний метод інтелектуального управління кешем, орієнтований на інформаційні високонавантажені системи з динамічним характером запитів, зокрема на систему підтримки процесу відновлення пошкоджених об’єктів нерухомості. Метод охоплює три ключові задачі: прийняття рішення про доцільність кешування нового об’єкта, вибір об’єкта для видалення у разі заповнення кешу та періодичну інвалідацію неактуальних об’єктів. Для вирішення цих задач використано поєднання моделей машинного навчання: табличного Q-Learning з механізмом відкладеного асинхронного оновлення значень моделі та логістичної регресії з механізмом відкладеного асинхронного накопичення буфера навчальних прикладів для подальшого навчання моделі та періодичного оновлення її параметрів. Проведено експериментальну оцінку ефективності методу, яка підтвердила перевагу інтелектуального підходу порівняно з існуючими методами управління кешем LRU та LFU, зокрема в умовах обмеженого розміру вхідних даних. Запропонований метод має високий потенціал для використання в інших інформаційних системах завдяки можливості адаптації вектора ознак моделей машинного навчання відповідно до характеристик об’єктів і контексту їх використання.

За результатами дослідження розроблено:

– гібридний метод інтелектуального управління кешем, що базується на моделях машинного навчання та враховує суттєві ознаки користувачів і пошкоджених об’єктів нерухомості;

– систему експериментального тестування ефективності гібридного методу інтелектуального управління кешем, що включає модулі для симуляції запитів, реалізації логіки роботи методу, а також взаємодії з моделями машинного навчання, технологією кешування та нереляційною базою даних.

У ***першому розділі*** здійснено аналіз основних особливостей організації та виконання процесу будівельно-технічної експертизи в сучасних умовах, зокрема після початку масштабних руйнувань інфраструктури України внаслідок бойових дій; окреслено основні етапи роботи експертів; визначено та описано проблеми, пов’язані з неефективною організацією зберігання, обробки й повторного використання даних у цьому процесі; сформульовано та описано ключові проблеми роботи з даними в інформаційних високонавантажених системах, визначено вплив цих проблем на продуктивність і складність їх вирішення; запропоновано підходи до вирішення виявлених проблем роботи з даними шляхом впровадження нереляційних баз даних і технології кешування як ефективних рішень, що здатні забезпечити гнучкість, масштабованість і швидкий доступ до даних; наведено порівняльний аналіз реляційних та нереляційних баз даних для кращого обґрунтування доцільності використання нереляційних баз даних; наведено загальну інформацію про технологію кешування; описано потенційні переваги для роботи з даними в разі використання цієї технології для будівельно-технічної експертизи та інформаційних високонавантажених систем; проведений аналіз слугує основою для формування вимог до роботи з даними в інформаційній системі підтримки процесу відновлення пошкоджених об’єктів нерухомості; виконано постановку задачі..

У ***другому розділі*** визначено основні функції, структурні та технічні вимоги для бази даних системи підтримки процесу відновлення пошкоджених об’єктів нерухомості; описано основні типи запитів на читання та запис, які виконуватимуть користувачі цієї системи з різними рівнями доступу; обґрунтовано пріоритетні властивості бази даних системи відповідно до CAP-теореми; визначено ключові

критерії, за якими оцінюється доцільність використання різних типів і технологій нереляційних баз даних для потреб системи; обґрунтовано доцільність використання СУБД MongoDB як бази даних для системи підтримки процесу відновлення пошкоджених об’єктів нерухомості на основі порівняльного аналізу з іншими типами та технологіями нереляційних баз даних; спроектовано структуру основних колекцій документів бази даних MongoDB для системи, зокрема: «Пошкоджені будівлі», «Міська статистика», «Користувачі» та «Історія запитів»; побудовано схему логічних взаємозв’язків між цими колекціями як основу для розробки гібридного методу інтелектуального управління кешем, що дозволяє суттєво підвищити ефективність роботи з даними; обґрунтовано доцільність використання інтелектуального кешування як засіб подолання обмежень існуючих методів кешування; описано роль кешування в сучасних інформаційних високонавантажених системах;

У ***третьому розділі*** розроблено гібридний метод інтелектуального управління кешем, який вирішує три ключові задачі: оцінка доцільності додавання об’єкта до кешу, вибір об’єкта для видалення з заповненого кешу та періодична інвалідація неактуальних об’єктів у кеші; подано визначення методу управління кешем як основи для реалізації гібридного підходу; обґрунтовано застосування моделей машинного навчання – табличного Q-Learning та логістичної регресії для вирішення відповідних задач запропонованого методу інтелектуального управління кешем; запропоновано представлення стану об’єкта у вигляді вектора ознак; спроектовано додаткові колекції документів для бази даних системи підтримки процесу відновлення пошкоджених об’єктів нерухомості MongoDB, необхідні для коректної роботи запропонованих моделей машинного навчання; обґрунтовано доцільність використання Redis як основної технології кешування для системи; описано алгоритм роботи гібридного методу інтелектуального управління кешем; показано фінальну блок-схему його роботи із застосуванням моделей машинного навчання табличного Q-Learning і логістичної регресії, технології нереляційної бази даних MongoDB і технології кешування Redis; представлено оновлену модель системи з

інтегрованим модулем інтелектуального кешування, що реалізовує розроблений гібридний метод.

У ***четвертому розділі*** описано структуру та основні функції системи експериментального тестування ефективності запропонованого гібридного методу інтелектуального управління кешем; наведено детальний опис компонентів системи, зокрема бази даних MongoDB, кеш Redis, двох моделей машинного навчання – табличного Q-Learning і логістичної регресії та модуля керування кешем, що реалізовує алгоритм роботи гібридного методу; реалізовано повноцінну програмну архітектуру системи, включно з агентами, буферами, оновленням моделей у фоновому асинхронному режимі та синхронізацією з базою даних MongoDB і кешем Redis; продемонстровано результати роботи системи у двох типах експериментів: аналіз частки кеш-хітів після одного запуску та серії запусків з накопичувальним навчанням моделей з фіксованим набором даних; у кожному експерименті проведено порівняння ефективності запропонованого гібридного методу управління кешем з найпоширенішими існуючими методами кешування: LRU та LFU; візуалізовано результати у вигляді графіків і рисунків, що підтверджують перевагу гібридного методу за умов обмеженого розміру кешу та кількості запитів.

Ключові слова: база даних, будівельно-технічна експертиза, інтелектуальне кешування, інформаційна високонавантажена система, машинне навчання, нереляційна база даних, технологія кешування, штучний інтелект.

Публікації. За темою дисертації опубліковано 7 робіт. Основні результати досліджень викладені в 2 статтях у друкованих виданнях, включених до переліку фахових видань України, 4 робіт представлених як тези доповіді у міжнародних науково-технічних конференціях та 1 тези конференції з апробацією.

Апробація результатів дисертації. Основні положення та результати дисертаційної роботи висвітлені та обговоренні на наступних конференціях:

1. Міжнародна науково-практична конференція "Перспективні напрямки розвитку науки, освіти, технологій та суспільства: теорія і практика", м. Полтава. 19 квітня 2023.

2. 2023 IEEE International Conference on Smart Information Systems and Technologies (SIST), Astana, Kazakhstan, 4-6 May 2023.

3. 2023 IEEE 4th KhPI Week on Advanced Technology (KhPIWeek), Kharkiv, Ukraine, 2-6 October 2023.

4. 3rd International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2023, Ternopil, Ukraine, 22-24 November 2023.

5. 4th International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2024, Ternopil, Ukraine, Opole, Poland, 23-25 October 2024.

В повному обсязі дисертація доповідалась на науковому семінарі кафедри інформаційних технологій проектування та прикладної математики, кафедри інформаційних технологій, Київського національного університету будівництва і архітектури (м. Київ, 2025 рік).

Особистий внесок здобувача. Основні результати та положення, які становлять суть (зміст) дисертації виконані автором самостійно. З наукових праць, опублікованих у співавторстві, у дисертації використано лише ті ідеї та положення, які є результатом особистої роботи здобувача.

Структура та обсяг дисертації. Дисертаційна робота складається із вступу, чотирьох розділів, загальних висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 160 сторінок, у тому числі, основна частина складає 125 сторінки. Основна частина, крім тексту, включає 18 таблиць, 25 рисунків.

Подяка. Висловлюю глибоку подяку науковому керівнику – кандидату технічних наук, доценту Єременку Богдану Михайловичу.

РОЗДІЛ 1. РОБОТА З ДАНАМИ У ПРОЦЕСІ БТЕ ТА ІНФОРМАЦІЙНИХ ВИСОКОНАВАНТАЖЕНИХ СИСТЕМАХ

У першому розділі здійснено аналіз основних особливостей організації та виконання процесу будівельно-технічної експертизи в сучасних умовах, зокрема після початку масштабних руйнувань інфраструктури України внаслідок бойових дій; окреслено основні етапи роботи експертів; визначено та описано проблеми, пов’язані з неефективною організацією зберігання, обробки й повторного використання даних у цьому процесі; сформульовано та описано ключові проблеми роботи з даними в інформаційних високонавантажених системах, визначено вплив цих проблем на продуктивність і складність їх вирішення; запропоновано підходи до вирішення виявлених проблем роботи з даними шляхом впровадження нереляційних баз даних і технології кешування як ефективних рішень, що здатні забезпечити гнучкість, масштабованість і швидкий доступ до даних; наведено порівняльний аналіз реляційних та нереляційних баз даних для кращого обґрунтування доцільності використання нереляційних баз даних; наведено загальну інформацію про технологію кешування; описано потенційні переваги для роботи з даними в разі використання цієї технології для будівельно-технічної експертизи та інформаційних високонавантажених систем; проведений аналіз слугує основою для формування вимог до роботи з даними в інформаційній системі підтримки процесу відновлення пошкоджених об’єктів нерухомості; виконано постановку задачі.

1.1. Ключові аспекти та проблеми роботи з даними у процесі БТЕ

У результаті повномасштабної збройної агресії російської федерації за підтримки республіки білорусь, що розпочалася 24 лютого 2022 року, на території України пошкоджено, частково або повністю зруйновано багато будівель, споруд і об'єктів інфраструктури [1]. Наразі продовжують зазнавати шкоди та руйнувань будинки і споруди різного призначення майже в усіх регіонах країни.

Станом на серпень 2024 року нинішня лінія фронту є найдовшою в Європі з часів Другої світової війни і становить понад 3000 км, а лінія активних бойових дій – близько 1000 км. Зважаючи на це стає зрозуміло, що однією з першочергових задач українського народу після успішного закінчення збройного конфлікту є швидке й якісне відновлення багатьох зруйнованих і пошкоджених об’єктів.

Для порівняння у мирних умовах обсяги будівельної діяльності збільшувались за рахунок:

– Фізичного старіння об'єктів нерухомості;

– Зростання цін;

– Зміни власників нерухомості;

– Необхідності реконструкції промислових підприємств, малоповерхових будівель;

– Активізації нового будівництва в районах старої забудови.

В сучасних умовах воєнного стану темпи будівельної активності значно збільшуються внаслідок масових руйнувань будівель, майна та об'єктів інфраструктури (рис. 1.1.).

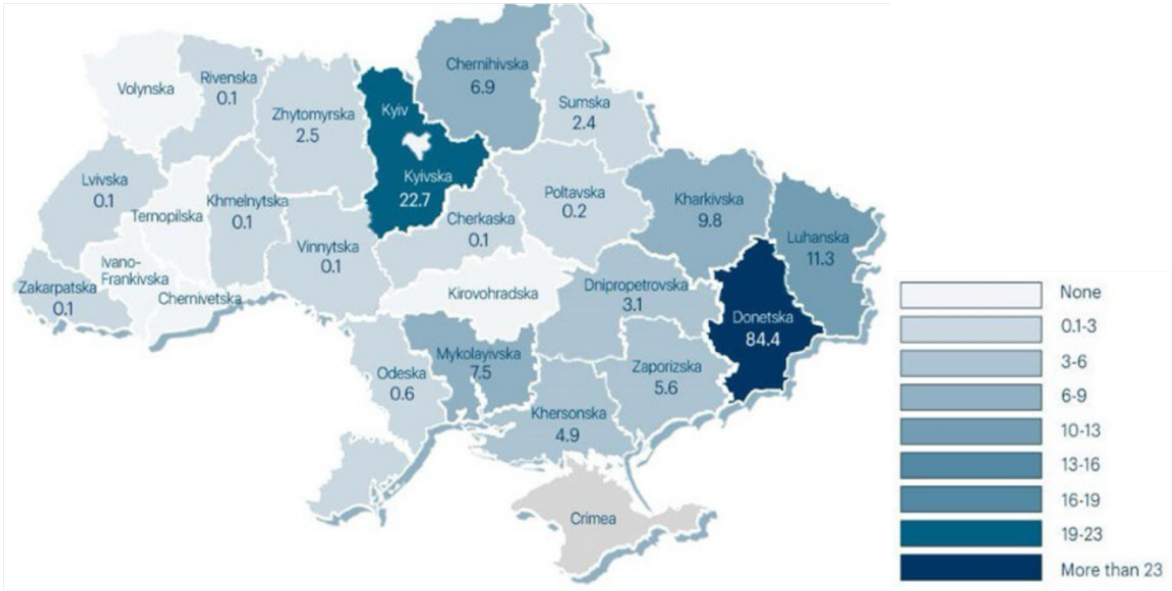

Рисунок 1.1. Розподіл областей за кількістю зруйнованих/пошкоджених будівель у тисячах одиниць [2]

Відновлення нерухомого майна та забезпечення житлом громадян України на територіях, де завершено військові дії, потребує проведення БТЕ об'єктів, які були зруйновані або пошкоджені. При цьому процес обстеження таких об'єктів вже змінювався, адаптуючись до реалій сьогодення [3], [4].

БТЕ належить до основних видів інженерно-технічних експертиз і являє собою дослідження експертом явищ і процесів, які містять інформацію про обставини справи, що знаходяться в органах досудового розслідування або суді, на підставі спеціальних знань у галузі [5], [6]:

– будівництва;
– проєктування;
– реконструювання будівель;
– відповідності будівельним нормам, правилам і явищам;
– оцінки технічного стану об’єктів, пошкоджених унаслідок бойових дій;
– використання сучасних технологій для фіксації пошкоджень, зокрема безпілотних літальних апаратів (БПЛА) та супутникових знімків.

Основні етапи проведення БТЕ відображено на рис. 1.2.

До повномасштабного вторгнення рф до основних питань БТЕ належало визначення:

– технічного стану будівель, споруд та інженерних мереж;
– причин пошкоджень та руйнувань БіС та їх елементів;
– вартості будівельних робіт, що пов’язані з переобладнанням, усуненням наслідків залиття, пожежі, стихійного лиха, механічного впливу тощо.

Однак в умовах повномасштабного вторгнення БТЕ зіткнулася з низкою викликів, що потребують адаптації до реалій воєнного часу. До таких викликів належать:

– Масштабність та кількість об’єктів, що потребують обстеження;
– Обмежений доступ до територій, які перебувають у зоні активних бойових дій або є замінованими;
– Проблеми з фінансуванням експертних робіт: значний дефіцит ресурсів для проведення повномасштабних експертиз;

– Недостатня кількість кваліфікованих експертів: через мобілізацію, евакуацію та фізичні ризики багато фахівців не можуть виконувати свою роботу;

– Нестача сучасного технічного обладнання: дефіцит дронів, лазерних сканерів, 3D-моделювання, що ускладнює проведення експертиз на великих територіях;

– Ризики подальшого руйнування будівель: багато споруд у критичному стані, а їхнє обстеження пов’язане з небезпекою для експертів;

– Розрив логістичних ланцюгів: труднощі з доступом до будівельних матеріалів і техніки для проведення відновлювальних робіт.

Рисунок 1.2. Основні етапи проведення БТЕ

Тепер роботи щодо фіксування доказів руйнувань об’єктів нерухомості проводяться із залученням судових експертів Науково-дослідних установ судових

експертиз Міністерства юстиції України, у тому числі й Київського науково-дослідного інституту судових експертиз [3]. Крім вітчизняних фахівців, в цьому процесі беруть участь судові експерти з інших країн, аби цей процес не викликав жодних сумнівів.

Робота, що проводиться криміналістичними експертами безпосередньо на місці події, в першу чергу спрямована на:

– Фіксація шкоди органам досудового розслідування та подальше проведення судових експертиз для встановлення причин та наслідків шкоди;

– Розрахунок розміру матеріального збитку.

По завершенню судових експертиз БТЕ проводяться обстеження будівель і споруд з метою надання висновку про ліквідацію або відновлення цих об'єктів і обґрунтування виду і вартості проведених відновлювальних робіт.

Таке обстеження включає в себе [4]:

– Ідентифікація зруйнованих та пошкоджених об'єктів;

– Визначення дефектів і пошкоджень будівельних конструкцій з визначенням сфери їх застосування;

– Оцифрування даних про кількість та характер руйнувань;

– Прийняття рішення про подальшу експлуатацію або ліквідацію об'єкта у зв'язку з пошкодженнями, завданими позапроектними впливами (військові дії та спричинені ними пожежі);

– Оцінка технічного стану будівельних конструкцій та об'єкта в цілому для проведення подальших проектних робіт з реставрації, реконструкції або капітального ремонту.

У БТЕ дані – це не лише вихідна інформація, а основа всього експертного висновку. Саме на основі об’єктивних відомостей про об’єкти, їхній стан, попередні огляди, документи, фотографії, запити, технічні характеристики та зміни експерт формує висновки, які мають юридичну й практичну силу.

Для фахівця надзвичайно важливо мати доступ до повної, узгодженої та актуальної інформації [7]. Навіть незначне спотворення або втрата контексту може призвести до неточних рішень або до необхідності повторного проведення частини

експертизи. Крім того, чим більшою є кількість об’єктів та учасників процесу (заявників, експертів, технічних спеціалістів), тим складніше підтримувати контроль над усіма потоками даних.

Водночас в реальній практиці робота з інформацією часто має фрагментований, неуніфікований характер: частина даних зберігається у вигляді файлів, частина – у таблицях Excel, частина – на паперових носіях або в електронній пошті [8]. Це створює умови для дублювання, втрат, помилок у версіях і значно ускладнює як доступ до інформації, так і її повторне використання.

Неефективна робота з даними призводить до цілого спектру проблем: уповільнення процесу, збільшення витрат часу на пошук, неузгодженість між версіями документів, складність масштабування процесу для великої кількості запитів. Особливо критичною є ситуація в умовах обмежених ресурсів, коли потрібна автоматизація та швидке реагування.

Відсутність уніфікованого підходу до зберігання інформації зумовлює труднощі на наступних етапах, а саме: аналізі, звітності, повторної перевірки або співпраці кількох експертів. Дані часто не узгоджені за форматом, не синхронізовані, а в різних джерелах можуть міститись дублікати або суперечлива інформація. Через це пошук потрібного фрагмента може вимагати значного часу, а ризик використання неактуальної або застарілої версії документа зростає.

Особливо складною є ситуація, коли об’єкт досліджувався кілька разів, а інформація про це розміщена у вигляді декількох звітів або таблиць без єдиного джерела правди. У таких випадках стає практично неможливим простежити повну картину змін у технічному стані об’єкта без ручного звірення та повторного аналізу.

Також слід зазначити, що відсутність автоматизованих механізмів контролю чи перевірки інформації створює передумови для помилок, неточностей або втрати даних. Усе це ускладнює не лише внутрішню експертну діяльність, але й формування якісних аналітичних висновків на рівні галузі.

Після початку повномасштабного вторгнення рф в Україну навантаження на систему БТЕ істотно зросло. Значна кількість об’єктів зазнала ушкоджень або повного знищення, що спричинило масштабний запит на фіксацію стану, обґрунтування збитків та підготовку висновків для подальшого юридичного або будівельного реагування [9]. До цього додалися стислі строки, обмежені ресурси, віддалена робота фахівців та необхідність обробки даних у нестабільних умовах, що ще більше загострило проблеми, пов’язані з неефективною організацією інформації.

Замість поступового впровадження єдиних стандартів та цифрових рішень, багатьом командам довелося адаптуватися “в польових умовах”, покладаючись на доступні засоби фіксації й обміну даними. Це поглибило фрагментованість даних, ускладнило верифікацію отриманих відомостей та зробило критично важливою здатність системи ефективно реагувати на запити. Таким чином, війна не лише збільшила обсяг роботи, але й чітко виявила системні недоліки, які раніше залишалися прихованими або вторинними.

Таким чином, основними проблемами роботи з даними у сучасній практиці БТЕ є:

1. Зберігання даних у різних форматах. Дана проблема означає, що одна й та сама інформаційна сутність, технічний стан об’єкта або хід виконання експертизи, може бути зафіксована у вигляді табличних даних, текстових описів, сканованих копій, фотографій чи листування. Формати істотно відрізняються за структурою, обсягом, придатністю до аналізу та автоматизованої обробки. Через це важко об’єднати всі ці частини в єдиний потік, а системне представлення даних стає ускладненим або неможливим без попереднього ручного опрацювання. Відсутність цілісності даних не лише затягує роботу експерта [10], змушуючи витрачати час на пошук необхідного файлу серед десятків, а іноді й сотень джерел, а й підвищує ризик помилок: деякі важливі частини інформації можуть бути просто пропущені. Це критично, особливо коли йдеться про судову експертизу, де важливе значення має кожна деталь. Крім того, фрагментарність унеможливлює ефективне застосування автоматичних аналітичних інструментів або підключення систем

підтримки прийняття рішень, оскільки такі інструменти потребують впорядкованих і доступних даних.

2. Відсутність єдиного джерела даних: У багатьох випадках дані, пов’язані з одним і тим самим об’єктом, не зберігаються централізовано, а розкидані по різних сховищах: окремих файлах, документах, листуваннях або базах. Відсутність єдиного джерела правди означає, що жодна з наявних точок доступу до інформації не гарантує її повноти, цілісності та актуальності. Через це фахівцям доводиться щоразу самостійно збирати фрагменти даних із різних джерел, витрачаючи час на пошук, звіряння та уточнення. Такий підхід ускладнює перевірку рішень, повторне використання наявної інформації та взаємодію між учасниками процесу. У разі змін в одному з джерел ці оновлення не завжди автоматично поширюються на інші, що породжує суперечності між версіями. Наприклад, у звіті може бути одне формулювання про характер пошкодження, а в супровідній таблиці інше. Подібні неузгодженості створюють ризик некоректних висновків або затримок в ухваленні рішень.

3. Повторна обробка одних і тих самих даних. У практиці будівельно-технічної експертизи нерідко трапляються ситуації, коли одні й ті самі запити або дії виконуються по кілька разів [11]. Наприклад, якщо до одного об’єкта звертаються кілька різних експертів або якщо той самий фахівець виконує кілька подібних завдань з інтервалом у часі, йому щоразу доводиться знову переглядати документи, повторно аналізувати фотофіксацію, знаходити технічні характеристики або перераховувати параметри. Це відбувається тому, що попередні результати не зберігаються в зручному для повторного використання вигляді, або ж доступ до них ускладнений Повторна обробка не тільки знижує загальну продуктивність роботи, але й підвищує ризик помилок, пов’язаних з людським фактором: експерт може зробити інші висновки на основі тих самих даних або просто не помітити певні нюанси, які вже були опрацьовані раніше. Крім того, надмірне дублювання зусиль марнує цінний час, який можна було б використати на інші, більш пріоритетні завдання.

4. Погана робота системи при великій кількості запитів:

У ситуаціях, коли експертна система обробляє велику кількість запитів одночасно, наприклад, після масових руйнувань або під час активного періоду реконструкції – виникають суттєві труднощі з її стабільною роботою [12]. Система, яка була ефективною при невеликому навантаженні, раптом починає сповільнюватися, давати збої або не встигає обробляти нові звернення вчасно. Це стосується як часу відповіді при пошуку інформації, так і загальної стабільності роботи інтерфейсів, бази даних або допоміжних сервісів. Причина часто полягає в тому, що обробка кожного запиту вимагає повного звернення до всієї інформаційної бази, навіть якщо йдеться про повторювані дії або часто використовувані дані. Система не здатна адаптуватися до зростаючого навантаження й не використовує можливостей для оптимізації ресурсів. У таких умовах експерт втрачає дорогоцінний час, а швидкість ухвалення рішень знижується. Це особливо критично в ситуаціях, коли оперативність експертизи прямо впливає на хід будівельних або відновлювальних робіт.

5. Відсутність врахування пріоритетності і популярності даних:

У багатьох випадках до різних об’єктів експертизи звертаються з неоднаковою частотою: деякі об’єкти перевіряються неодноразово, інші ж – лише один раз [13]. Попри це, система зазвичай обробляє всі запити однаково, незалежно від того, наскільки часто або наскільки терміново потрібно звертатися до конкретного об’єкта. В результаті інформація про менш важливі чи рідко використовувані об’єкти може перебувати «в центрі уваги» системи, тоді як дані про популярні чи критичні об’єкти не мають переваги в обробці. Таке однакове ставлення до всіх даних створює ситуацію, коли система неефективно використовує свої ресурси: час, пам’ять, обчислювальну потужність. Замість того, щоб швидко надавати доступ до інформації, яку часто запитують, вона змушена щоразу обробляти всі звернення, навіть якщо деякі з них повторюються багато разів на день. Це уповільнює роботу, підвищує навантаження та не дозволяє системі адаптуватись до реальних потреб користувачів.

6. Неактуальність інформації при прийнятті рішень. У практиці БТЕ часто трапляється, що рішення приймаються на основі даних, які вже втратили свою

актуальність. Це може бути спричинено як технічними затримками в оновленні інформації, так і організаційними труднощами. Як наслідок, експерт працює з інформацією, яка вже не відображає реальний стан речей. Це створює значні ризики для точності висновків [14]. Наприклад, пошкодження, зафіксоване під час попереднього обстеження, могло бути частково усунене, але якщо оновлені дані не були вчасно враховані, рішення базуватиметься на застарілій картині. Така невідповідність особливо критична у випадках, коли експертиза стосується питань безпеки, розподілу фінансування або юридичної відповідальності.

Таким чином, перелічені проблеми свідчать про те, що сучасна практика роботи з даними у сфері БТЕ залишається недостатньо ефективною і не відповідає зростаючим вимогам до швидкості, масштабованості та точності. Фрагментарність зберігання, дублювання операцій, затримки в оновленні інформації та відсутність адаптивності до навантажень суттєво знижують якість прийнятих рішень і уповільнюють процес відновлення. Ці виклики потребують не лише організаційних змін, а й впровадження сучасних технологічних рішень. Зокрема, використання нереляційних баз даних та інтелектуального кешування дозволяє принципово інакше побудувати обробку даних – з урахуванням реального навантаження, потреб користувачів і динаміки запитів.

1.2. Проблеми роботи з даними в інформаційних високонавантажених системах

Зростання обсягів даних, кількості користувачів і рівня вимог до швидкості обробки створює в сучасних інформаційних системах низку проблем, які традиційні технології зберігання вже не здатні ефективно вирішувати. Особливо це стосується високонавантажених систем, що працюють у режимі реального часу, обслуговують паралельні запити та потребують стабільності при піковому навантаженні.

Однією з найбільш поширених проблем є затримки, пов’язані з великою кількістю звернень до великих масивів даних. У системах, які не мають проміжних рівнів зберігання або оптимізованого доступу, кожен запит змушує виконувати повний цикл звернення до бази даних. Це призводить до перевантаження сховища,

зростання часу відповіді та зниження загальної продуктивності. При одночасній роботі десятків або сотень користувачів такі затримки можуть бути критичними [15].

Ще одним суттєвим викликом є неоптимальна організація доступу до часто використовуваних даних. У багатьох системах немає розділення між часто запитуваними та рідко потрібними об’єктами. Як наслідок, ресурси системи витрачаються рівномірно, без урахування реальної ваги або частоти використання інформації. Це знижує ефективність і створює зайве навантаження на інфраструктуру [13].

Крім того, в умовах динамічного середовища часто виникає проблема відставання в оновленні даних. Система може приймати рішення або формувати відповіді, базуючись на застарілих даних, якщо не реалізовано механізмів фонової синхронізації або частотного оновлення. В умовах високого навантаження це призводить до великого накопичення помилок і збільшення кількості неактуальних об’єктів в обігу [14].

Також поширеною є висока вартість складних запитів, які містять об’єднання великих таблиць, фільтрацію або сортування за багатьма критеріями. Такі операції, особливо при відсутності індексації або при роботі з гнучкою структурою даних, значно сповільнюють систему. Вони можуть блокувати доступ для інших користувачів і створювати колізії при паралельному читанні та записі.

Не менш важливою є трудність адаптації традиційних систем до горизонтального масштабування. Хоча обчислювальні ресурси часто можна масштабувати шляхом додавання нових вузлів, більшість класичних реляційних баз даних не підтримують розподілену архітектуру у базовій реалізації. Це ускладнює балансування навантаження, резервування та зберігання даних у хмарних середовищах.

Усі перелічені проблеми (табл.1.1.) свідчать про необхідність перегляду підходів до організації роботи з даними у високонавантажених системах. Стрімке зростання обсягів інформації, складність запитів і потреба в адаптивності до

змінного навантаження вимагають використання рішень, здатних забезпечити гнучкість, масштабованість і швидкий доступ до найбільш актуальної інформації.

Таблиця 1.1.

Вплив основних проблем роботи з даними у інформаційних високонавантажених системах та складність їх вирішення

Проблема	Наслідки для системи	Складність вирішення
Затримки при зверненні до великих обсягів даних	Суттєве зниження продуктивності, критичні затримки для користувачів	Висока
Відсутність поділу на «гарячі» та «холодні» дані	Неефективне використання ресурсів, зайве навантаження	Середня
Застарілі дані в обігу через відсутність синхронізації	Помилкові рішення, накопичення неактуальних записів	Середня
Складні запити з об’єднанням великих таблиць	Блокування доступу, конфлікти при паралельному доступі	Висока
Неможливість горизонтального масштабування в традиційних базах даних	Обмежена масштабованість, складність з хмарною інтеграцією	Висока

1.3. Нереляційні бази даних і кешування як підходи до вирішення проблем роботи з даними

У сучасних умовах, коли БТЕ все частіше виконується у великих масштабах, а кількість даних у високонавантажених системах рості, постає потреба у більш ефективних та гнучких підходах до зберігання і обробки даних. Традиційні

реляційні бази даних, які десятиліттями були основою інформаційних систем, виявляються недостатньо адаптивними до вимог динамічного, різноманітного і швидкозмінного інформаційного середовища, характерного для БТЕ. У таких умовах дедалі більше уваги привертають нереляційні бази даних (NoSQL), що побудовані за принципами гнучкості, масштабованості та здатності ефективно працювати з різнорідними або неструктурованими даними.

Однією з основних переваг NoSQL-систем є відсутність жорсткої схеми таблиць, яка є обов'язковою у реляційних базах даних. У контексті БТЕ це означає, що система зберігання даних може легко адаптуватися до нових типів інформації без необхідності змінювати структуру всієї бази. Наприклад, якщо в процесі експертизи з'являються нові характеристики об'єкта, нові типи пошкоджень або супровідні документи, їх можна додати до запису без необхідності створення нових таблиць або складних зв'язків. Така гнучкість особливо важлива в умовах, коли різні об'єкти мають неоднакові набори характеристик і потребують індивідуального підходу [16].

Ще однією суттєвою перевагою нереляційних баз є здатність зберігати всю інформацію про об'єкт у межах одного документа. Це дозволяє уникнути складних операцій зі зв'язуванням таблиць (JOIN) і пришвидшує обробку запитів, що критично важливо при великій кількості об'єктів. Наприклад, усі дані про конкретну будівлю – загальні характеристики, зафіксовані пошкодження, результати попередніх експертиз, прикріплені фото та супровідні файли можуть зберігатись у межах одного документа. Це не лише спрощує доступ до повної картини, а й зменшує кількість звернень до бази при отриманні потрібної інформації [17].

З технічної точки зору NoSQL-системи краще адаптовані до горизонтального масштабування. Якщо реляційні бази зазвичай вимагають потужнішого обладнання (вертикальне масштабування), то нереляційні можна розподіляти на кілька серверів, що дозволяє краще справлятись з великим потоком запитів та обробкою даних у режимі реального часу. У випадку широкомасштабної відновлення пошкоджених об'єктів та масового надходження запитів на експертизу це є суттєвою перевагою.

Окремо варто згадати і про зручність роботи з вкладеними даними, які часто супроводжують процес експертизи. Йдеться про фотофіксації, таблиці з характеристиками, примітки експертів, історію змін, геопросторові мітки тощо. Нереляційна модель дозволяє органічно зберігати ці елементи в межах одного запису, що значно полегшує аналіз і автоматичну обробку. У свою чергу, це відкриває перспективи для використання аналітики, машинного навчання та інтелектуального пошуку [18].

Крім того, NoSQL-підходи полегшують централізацію і стандартизацію зберігання даних, дозволяючи створювати єдине джерело правди (*single source of truth*). Це означає, що замість зберігання окремих фрагментів інформації у вигляді звітів, Excel-файлів чи сканів у різних місцях, можна інтегрувати всю інформацію про об’єкт у цілісний і доступний запис. Така уніфікація критично важлива для забезпечення узгодженості даних, спрощення співпраці між фахівцями та мінімізації помилок у висновках.

Таким чином, у випадку БТЕ перехід до використання нереляційних баз даних є не просто доцільним, а й стратегічно виправданим кроком. Це дозволяє суттєво покращити гнучкість, ефективність і масштабованість роботи з даними, особливо в умовах високого навантаження та потреби в динамічному оновленні інформації.

На основі порівняння реляційних та нереляційних баз даних (табл. 1.2.) з’ясовано, що використання останніх може суттєво покращити ефективність роботи з даними в процесі сучасної БТЕ.

Таблиця 1.2.

Порівняльний аналіз реляційних та нереляційних баз даних

Критерій	Реляційна БД (SQL)	Нереляційна БД (NoSQL)
Формат зберігання даних	Таблиці з чіткою структурою	Документи довільної структури (наприклад, JSON)
Цілісність інформації про об’єкт	Інформація розподілена між таблицями	Уся інформація зберігається в одному документі
Актуальність даних при доступі	Дані можуть бути неузгодженими між таблицями	Дані зберігаються цілісно, що знижує ризик неактуальності
Масштабованість при високих навантаженнях	Переважно вертикальна, обмежена	Горизонтальна, адаптована до зростання кількості об’єктів
Робота з часто запитуваними даними	Всі запити обробляються однаково, без урахування частоти	Можлива інтеграція з кешами та адаптація до частоти запитів
Зручність повторного використання даних	Повторний аналіз вимагає складних запитів	Дані готові до повторного використання без трансформацій
Повнота доступу до даних об’єкта	Потрібні складні JOIN-запити або окремі інструменти	Уся інформація доступна через один запит до документа

Кешування – це технологія, яка використовується для підвищення продуктивності та швидкодії інформаційних систем за рахунок тимчасового зберігання даних у швидкому проміжному сховищі (кеші). Основна ідея кешування

полягає в тому, щоб мінімізувати звернення до основного джерела даних (наприклад, бази даних або зовнішнього сервісу) при повторних запитах. Це досягається шляхом збереження результатів запитів або частин даних у кеші, який забезпечує швидший доступ до інформації [19].

Ключовими перевагами кешування є:

– Підвищення швидкості доступу до інформації завдяки зберіганню часто використовуваних даних у пам’яті;

– Зменшення навантаження на основне сховище: Наприклад, на базу даних, що важливо для інформаційних високонавантажених систем;

– Поліпшення масштабованості системи шляхом зменшення залежність від основного сховища, система може обробляти більше запитів.

Однією з головних особливостей кешу є обмежений розмір. З одного боку, саме це забезпечує високу швидкодію, а з іншого – вимагає раціонального управління вмістом. Наразі існують класичні методи кешування, такі як LRU або LFU), які фокусуються виключно на виборі об’єкта для видалення у разі заповнення кешу:

– LRU видаляє об’єкт, до якого зверталися не звертались найдовше;

– LFU видаляє об’єкт із найменшою частотою звернень.

Попри свою простоту, ці методи вирішують лише одну з трьох ключових задач кешування. Вони не враховують, чи варто додавати об’єкт до кешу, а також не забезпечують інвалідацію неактуальних даних.

Кешування може бути класифіковане за різними критеріями. Основними видами є:

1. Кешування за місцем розташування:
   – Клієнтське кешування. Дані зберігаються на пристрої користувача, наприклад, у браузері. Це підходить для статичних ресурсів, таких як зображення чи файли стилів, і зменшує затримки між клієнтом і сервером;

- Серверне кешування. Дані зберігаються на сервері, що дозволяє прискорити виконання запитів на стороні серверу та знизити навантаження на основну базу даних [21].

2. Кешування за типом реалізації [22]:
   - Cache-Aside для читання даних. В цьому підході додаток спочатку перевіряє, чи потрібні дані є в кеші. Якщо даних немає (*cache miss*), додаток отримує їх з бази даних і додає в кеш. Надалі ці дані використовуються при наступних запитах. Це підходить для сценаріїв із частими операціями читання та повторюваними запитами, забезпечуючи швидший доступ до часто запитуваних даних;
   - Cache-Aside для запису даних. В цьому підході додаток спочатку змінює дані в базі даних. Після цього додаток або оновлює ці дані в кеші, або видаляє їх з кешу, залежно від того, чи потрібні ці дані в майбутньому. Це забезпечує контроль над тим, які дані залишаються актуальними в кеші після оновлення основної бази;
   - Read-Through Cache. Додаток завжди звертається до кешу. Якщо даних немає в кеші (*cache miss*), кеш самостійно завантажує їх із бази даних, зберігає для наступних звернень і повертає додатку. Підхід зручний, оскільки автоматизує процес наповнення кешу, але менш гнучкий, ніж Cache-Aside;
   - Write-Through Cache. При оновленні даних у додатку вони одночасно записуються в кеш і базу даних. Це забезпечує узгодженість між кешем і базою, але може спричиняти затримки через синхронний запис;
   - Write-Behind Cache. Дані спочатку записуються в кеш, а потім із певною затримкою оновлюються в базі даних. Це підхід із високою продуктивністю для запису, але з ризиком тимчасової неконсистентності [23].

3. Кешування за типом даних:

- Дані додатків. Наприклад, результати обчислень, статистика або сесії користувачів;
- Медіафайли. Для зберігання великих файлів, таких як зображення чи відео, можуть використовуватися спеціалізовані кеші або CDN (Content Delivery Network).

4. Кешування за часом життя даних:
   - Постійне кешування. Дані зберігаються до явного видалення;
   - Тимчасове кешування (TTL). Дані автоматично видаляються після закінчення певного часу.

У сфері БТЕ кешування може суттєво покращити ефективність обробки запитів до даних, особливо в умовах масового навантаження. У практиці експертів часто виникають ситуації, коли одні й ті самі об’єкти аналізуються кілька разів різними фахівцями або коли потрібні фрагменти інформації вже неодноразово використовувалися раніше. Повторний доступ до таких даних без кешування передбачає кожного разу повне звернення до основного сховища, що займає час і навантажує систему. Завдяки кешуванню ці повторювані звернення можуть бути оброблені значно швидше. Особливо це важливо в умовах масштабної відновлення інфраструктури, коли кількість запитів значно зростає. У таких випадках навіть незначне скорочення часу доступу до даних має велике значення для загальної продуктивності системи. Кешування дозволяє не тільки зменшити навантаження на базу даних, а й скоротити затримки в роботі з інформацією, що безпосередньо впливає на швидкість прийняття рішень.

Крім того, сучасні підходи до кешування дозволяють враховувати динаміку звернень до даних: найпопулярніші об’єкти можуть залишатися в кеші довше, а менш актуальні витіснятися. Це створює підґрунтя для адаптивних систем, які автоматично підлаштовуються під потреби користувачів та зменшують обсяг зайвих операцій. Завдяки цьому кешування стає не просто засобом пришвидшення, а ключовим компонентом інтелектуальної обробки запитів у БТЕ та в інформаційних високонавантажених системах.

1.4. Загальна характеристика системи підтримки процесу відновлення пошкоджених об’єктів нерухомості

У відповідь на суттєві виклики, пов’язані з масштабними руйнуваннями об’єктів нерухомості внаслідок бойових дій, а також виявлені проблеми у роботі з даними у процесі виконання будівельно-технічної експертизи, виникла потреба у створенні загальнодоступної інформаційно-комунікаційної системи, що вперше була запропонована в [24]. Така система покликана забезпечити централізоване, зручне та масштабоване середовище для збору, зберігання, обробки й аналізу інформації про пошкоджені об’єкти нерухомості.

Пропонована система отримала умовну назву Система Підтримки Процесу Відновлення Пошкоджених Об’єктів Нерухомості (СППВПОН) Основною метою системи є підтримка прийняття рішень на різних етапах процесу відновлення – від первинної фіксації пошкоджень до експертної оцінки технічного стану та формування планів реконструкції. Система орієнтована на високу навантаженість, розподілену архітектуру, інтеграцію з системами супутникового і безпілотного спостереження, а також з державними реєстрами та фондами відновлення. На рис. 1.3 показана модель СППВПОН, яка була вперше запропонована і детально описана в [24].

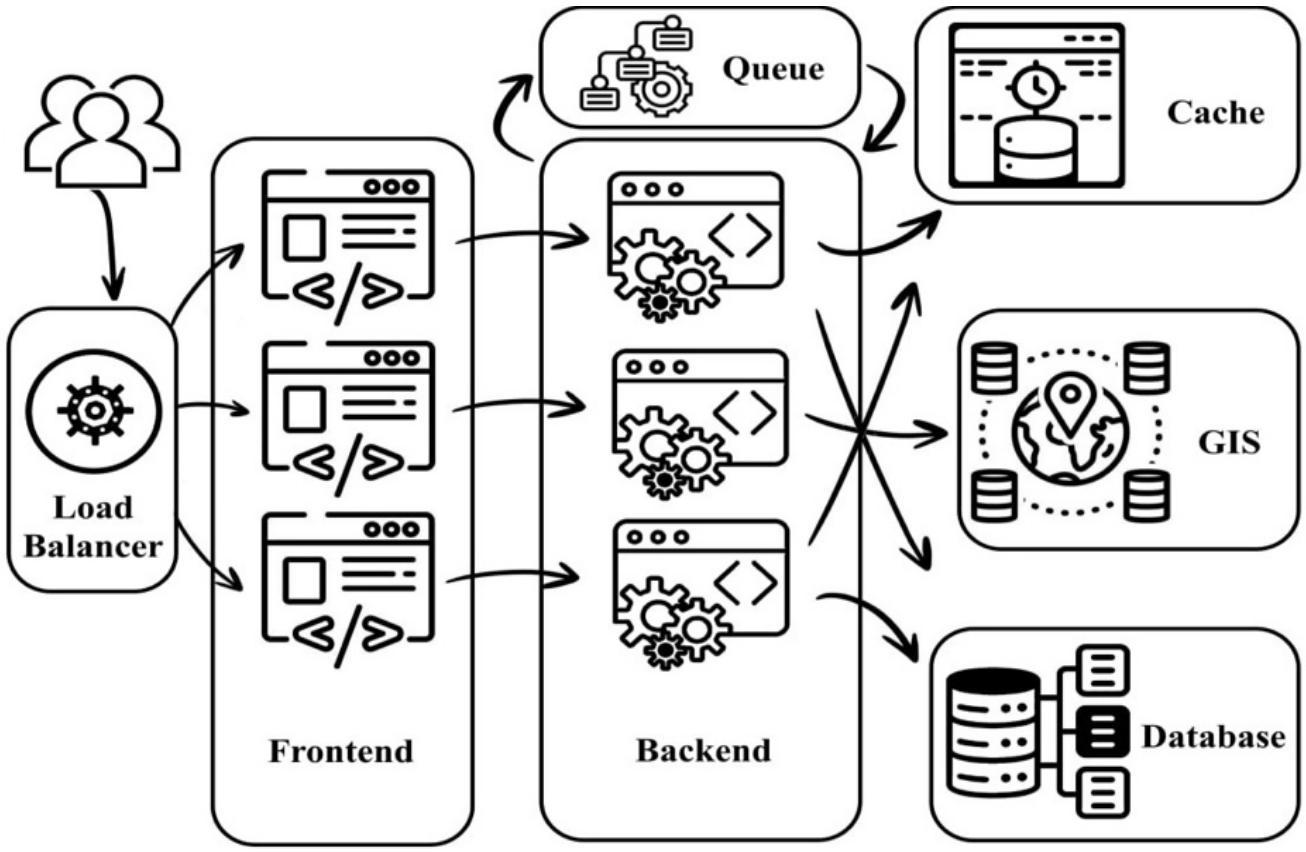

Рисунок 1.3. Початкова модель інформаційної високонавантаженої СППВПОН [24]

До основних завдань системи належать [25]:

– Оцифрувати руйнування та пошкодження об'єктів, щоб побудувати покроковий план відновлення;

– Інформувати світ про реальну ситуацію та наслідки збройної агресії Росії в Україні;

– Дозволити запустити процес визначення збитків з метою отримання репарацій та залучення фінансування;

– Надавати громадам аналітику та допомагати їм отримувати фінансування на відбудову;

– Забезпечити належні державні установи, будівельні організації та інвесторів інформацією про стан пошкоджених населених пунктів або окремих об'єктів з метою формулювання концепції їх відновлення;

– Забезпечити структуроване, централізоване та довгострокове зберігання інформації про пошкоджені об’єкти з можливістю подальшого оновлення;

– Підтримувати процес оцінки технічного стану об’єктів, включно з фіксацією стану конструктивних елементів та ступеня небезпеки;

– Надавати можливість фільтрації, сортування та аналізу даних за різними критеріями (регіон, тип об’єкта, ступінь руйнування тощо);

– Інтегруватися з іншими державними платформами, реєстрами та гуманітарними системами, забезпечуючи уніфікацію даних.

СППВПОН будується на основі мікросервісної архітектури, що передбачає поділ її функціоналу на низку незалежних, але взаємопов’язаних компонентів. Такий підхід дозволяє досягти гнучкості, масштабованості та підвищеної надійності системи, адже кожен мікросервіс відповідає за окремий функціональний напрям: від обробки запитів користувачів до управління базами даних і інтеграції модуля інтелектуального кешування. Важливою особливістю є можливість незалежного розгортання й оновлення окремих сервісів без зупинки всієї системи.

На зображенні нижче подано приклад одного з ключових елементів архітектури – модуля обробки і збереження даних [26]. Цей модуль відповідає за отримання, збереження, обробку та передачу даних між різними компонентами системи, забезпечуючи при цьому актуальність і цілісність інформації. Він виконує роль посередника між нереляційним сховищем даних, модулем інтелектуального кешування та бізнес-логікою додатку, а також забезпечує взаємодію з користувачем через API-запити. У рамках мікросервісної структури такий модуль може бути масштабований окремо, з урахуванням обсягу навантаження на підсистему зберігання й обробки інформації.

Рисунок 1.4. Схема мікросервісної архітектури модуля обробки і збереження даних

Таким чином, СППВПОН не обмежується лише збором фактів про пошкодження чи створенням бази об’єктів. Вона покликана виконувати значно ширші функції – бути інструментом аналізу, координації та комунікації між усіма учасниками відновлення. Завдяки правильній організації даних система може допомогти не тільки експертам, а й громадам, державним органам, інвесторам, донорам і міжнародним партнерам – кожному, хто має бути залучений у процес відновлення. Її ефективність, масштабованість і зручність напряму залежать від того, наскільки добре вирішені проблеми організації роботи з даними, описані в цьому розділі, і які технологічні рішення буде обрано для їх подолання.

1.5. Постановка задачі

Основним завданням дисертаційного дослідження є розробка методу інтелектуального управління кешем в інформаційній високонавантаженій системі підтримки процесу відновлення пошкоджених об'єктів нерухомості, що базується на машинному навчанні і здатен забезпечити ефективний доступ до актуально важливих даних з урахуванням суттєвих характеристик користувачів і об’єктів.

Запропонований метод вирішуватиме основні задачі кешування:

1. Оцінка доцільності додавання об’єкта до кешу;
2. Вибір об’єкта для видалення з заповненого кешу;
3. Періодична інвалідація неактуальних об’єктів у кеші.

Вирішення цього завдання передбачає:

1. Проаналізувати проблеми роботи з даними у сучасному процесі БТЕ та інформаційних високонавантажених системах та існуючі методи кешування;
2. Спроектувати базу даних СППВПОН;
3. Розробити гібридний метод інтелектуального управління кешем СППВПОН;
4. Оцінити ефективність запропонованого гібридного методу інтелектуального управління кешем.

Висновки до розділу 1

1. Окреслено послідовність виконання будівельно-технічної експертизи пошкоджених об’єктів в Україні та наголошено на критичній ролі даних у цьому процесі. Встановлено, що в умовах масштабного руйнування інфраструктури проблеми, пов’язані з обробкою даних, набувають системного характеру та потребують ефективного технологічного вирішення.

2. Визначено ключові проблеми, характерні для роботи з даними у інформаційних високонавантажених системах. Показано, що ці проблеми суттєво обмежують продуктивність систем та знижують якість прийнятих рішень, що підтверджує потребу у впровадженні альтернативних технологічних рішень.

3. Обґрунтовано доцільність застосування нереляційних баз даних і технологій кешування як ефективних засобів вирішення виявлених проблем роботи з даними. Показано, що ці підходи забезпечують гнучку схему зберігання даних, масштабованість і прискорення доступу до інформації, що є критично важливим у динамічних і високонавантажених інформаційних системах.

4. Описано загальну концепцію інформаційної системи підтримки процесу відновлення пошкоджених об’єктів нерухомості (СППВПОН), яка має розподілену мікросервісну архітектуру, орієнтовану на роботу з великим навантаженням, інтеграцію з різними джерелами даних і підтримку різних типів користувачів. Запропонована система призначена для вирішення проблеми роботи з даними у сфері БТЕ та забезпечення єдиного цифрового середовища для збору, обробки й аналітики інформації про пошкоджені об’єкти нерухомості.

РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ СИСТЕМИ ПІДТРИМКИ ПРОЦЕСУ ВІДНОВЛЕННЯ ПОШКОДЖЕНИХ ОБ'ЄКТІВ НЕРУХОМОСТІ

У другому визначено основні функції, структурні та технічні вимоги для бази даних системи підтримки процесу відновлення пошкоджених об'єктів нерухомості; описано основні типи запитів на читання та запис, які виконуватимуть користувачі цієї системи з різними рівнями доступу; обґрунтовано пріоритетні властивості бази даних системи відповідно до CAP-теореми; визначено ключові критерії, за якими оцінюється доцільність використання різних типів і технологій нереляційних баз даних для потреб системи; обґрунтовано доцільність використання СУБД MongoDB як бази даних для системи підтримки процесу відновлення пошкоджених об'єктів нерухомості на основі порівняльного аналізу з іншими типами та технологіями нереляційних баз даних; спроектовано структуру основних колекцій документів бази даних MongoDB для системи, зокрема: «Пошкоджені будівлі», «Міська статистика», «Користувачі» та «Історія запитів»; побудовано схему логічних взаємозв'язків між цими колекціями як основу для розробки гібридного методу інтелектуального управління кешем, що дозволяє суттєво підвищити ефективність роботи з даними; обґрунтовано доцільність використання інтелектуального кешування як засіб подолання обмежень існуючих методів кешування; описано роль кешування в сучасних інформаційних високонавантажених системах.

2.1. Загальна характеристика бази даних системи

Щоб СППВПОН могла ефективно вирішувати поставлені завдання, зокрема, фіксацію пошкоджень, підтримку технічного обстеження, формування звітності та аналітики, то база даних цієї системи, що лежить в її основі, повинна забезпечувати низку ключових функцій. Ці функції безпосередньо визначають, наскільки СППВПОН буде здатна працювати з великими обсягами даних, витримувати навантаження та бути масштабованою в умовах реального впровадження.

Основними функціями бази даних є [25]:

– Обробляти велику кількість запитів за короткий проміжок часу, включаючи паралельні звернення від користувачів із різними правами доступу;

– Коректно реагувати на запити користувачів, незалежно від складності запиту та обсягу пов’язаної з ним інформації;

– Зберігати великі обсяги даних із динамічною або непередбачуваною структурою, що є типовим для описів пошкоджень об’єктів, польової інформації, свідчень, геопросторових координат тощо;

– Забезпечувати надійність роботи системи, включаючи здатність до відновлення після збоїв, а також роботу при одночасному виході з ладу окремих компонентів інфраструктури;

– Масштабуватись відповідно до потреб системи, як у плані обсягу даних, так і кількості користувачів, що звертаються до системи;

– Інтегруватися з іншими інформаційними ресурсами, включаючи державні реєстри, картографічні сервіси, супутникові дані та системи автоматизованого моніторингу;

– Забезпечувати швидкий пошук, фільтрацію та агрегацію даних, що необхідно для побудови звітів, формування статистики та підтримки прийняття рішень на різних рівнях управління.

Саме тому важливою задачею цієї роботи є вибір найбільш підходящої технології бази даних і побудова її структурної моделі документів в контексті СППВПОН.

Для виконання цього завдання необхідно:

1. Визначити, які вимоги повинна задовольняти база даних СППВПОН, щоб забезпечити виконання описаних вище функцій;

2. Проаналізувати типові запити на запис і читання, які здійснюватимуть користувачі системи, з метою врахування їх для визначення найкращої технології бази даних та побудові моделі даних для неї;

3. Визначити бажані властивості згідно CAP-теореми (узгодженість, доступність, стійкість) та оцінити, яке поєднання властивостей є пріоритетним для СППВПОН з урахуванням особливостей її ключових функції;

4. Сформулювати критерії оцінки придатності різних типів та технологій нереляційних баз даних для основних задач системи (структура даних, масштабованість, гнучкість, швидкодія, стійкість тощо);

5. Дослідити головні типи та технології нереляційних баз даних, які відповідають зазначеним критеріям і найкраще підходять для використання в умовах розподіленої архітектури СППВПОН та потенційно високого навантаження;

6. Обрати найбільш доцільну технологію нереляційної бази даних для бази даних СППВПОН та обґрунтувати;

7. Спроектувати модель даних для обраної бази даних, описавши структуру основних колекцій, їхні поля, взаємозв’язки та призначення у межах функціонування системи.

Щоб база даних інформаційної системи СППВПОН могла забезпечити виконання її функцій, необхідно, щоб вона відповідала низці як структурних, так і технічні вимог. Структурні вимоги пов’язані з тим, які дані зберігаються і як вони організовані, тоді як технічні з особливостями обробки, масштабування, надійності й швидкодії.

Основними структурними вимогами є:

1. Динамічна структура даних. Система має зберігати об’єкти з різною структурою: будівлі можуть мати різну кількість пошкоджень, різні джерела даних, варіанти технічної оцінки. Це вимагає від БД здатність підтримувати гнучку схему або її відсутність, що дозволяє адаптувати формат даних під змінні умови.

2. Вкладені структури. Наприклад, один документ про пошкоджену будівлю може містити вкладений об’єкт “контактна інформація” або множину посилань на медіа файли. Підтримка вкладених полів дозволяє зберігати пов’язані дані разом і зменшує потребу в об’єднаннях (join) [27].

3. Підтримка геопросторових даних. У системі кожен об’єкт нерухомості має бути точно прив’язаний до географічного розташування. Для цього база даних

повинна підтримувати зберігання координат у форматі широти та довготи, обробку геопросторових типів даних і виконання просторових запитів, зокрема фільтрацію об’єктів за належністю до певної зони або регіону. Це критично важливо для візуалізації на карті, прив’язки до адміністративних меж та організації відновлення за територіальним принципом.

4. Підтримка мультимедійних матеріалів. У процесі проведення експертиз накопичується велика кількість супровідної візуальної інформації: фотографії пошкоджень, відеоогляди, скановані документи тощо. Оскільки такі файли займають значний обсяг, зберігати їх безпосередньо в базі даних недоцільно. Натомість у базі повинні зберігатися лише посилання на відповідні медіафайли, які розміщені у зовнішньому хмарному середовищі – наприклад, у сховищі типу Amazon S3 [28]. Такий підхід забезпечує швидкий доступ до матеріалів без перевантаження бази, а також дозволяє централізовано керувати візуальними даних у СППВПОН.

Основними технічними вимогами є:

1. Горизонтальне масштабування (шардінг). У разі збільшення кількості користувачів і даних, система повинна розподіляти навантаження між кількома вузлами. Це забезпечується шардінгом – розбиттям даних на частини, які зберігаються на різних серверах [29]. Такий підхід дозволяє зберігати продуктивність при рості.

2. Реплікація даних. Система має залишатися доступною навіть при відмові окремих серверів. Для цього необхідна реплікація – автоматичне дублювання даних на кількох вузлах [30]. Це забезпечує як надійність, так і безперервність роботи.

3. Автоматичне перемикання. У разі виходу з роботи основного вузла система має автоматично переключатись на резервний без втручання адміністратора [31]. Це дозволяє уникнути простоїв та втрати даних.

4. Багатопотокова обробка запитів. Оскільки система передбачає одночасну роботу багатьох користувачів – державних структур, експертів, техніків, то база даних СППВПОН має підтримувати одночасну обробку великої кількості запитів без зниження продуктивності.

5. Індексація. Для швидкого пошуку за геолокацією, типом об’єкта, ступенем пошкодження, користувачем тощо, база даних повинна мати можливість створювати індекси по ключових полях, що значно пришвидшує виконання запитів [32].

6. Підтримка транзакцій ACID. У ситуаціях, коли важлива повна цілісність даних (наприклад, під час оновлення відразу кількох частин документа), база даних має забезпечувати атомарність, узгодженість, ізоляцію та довговічність [33]. Навіть у нереляційній моделі іноді необхідно зберігати цілісність при складних оновленнях.

Користувачами системи будуть органи державної влади, архітектори та інженери, будівельники, експерти, спеціалізовані фонди відновлення України, власники зруйнованих або пошкоджених об'єктів [24], які можуть одночасно виконувати запити на запис і читання даних, такі як [34] і [35]:

– Додавання фото та відео пошкоджених об'єктів, зроблених із супутників та безпілотного літального апарату;

– Додавання даних від спеціалізованих державних структур, які генерують інформацію про кожну окрему будівлю та/або від організацій та експертів, які займаються процесом відновлення;

– Додавання або оновлення результатів повторної технічної експертизи об’єкта, що фіксують зміни його стану після проведення ремонтних чи відновлювальних робіт;

– Огляд основних базових характеристик пошкоджених об'єктів;

– Огляд основних технічних характеристик пошкоджених об'єкті;

– Огляд різних пошкоджень, завданих об'єктам в результаті бойових дій;

– Огляд оцінок технічного стану об'єктів;

– Формування зведених звітів щодо кількості пошкоджених об’єктів у межах певного регіону, категорії пошкоджень або статусу їх відновлення;

– Огляд фінансових втрат, завданих власникам пошкоджених об'єктів [25].

Таким чином, база даних СППВПОН працюватиме з великою кількістю запитів на запис та читання через критично велику кількість пошкоджених будівель,

споруд та інфраструктури. Крім того, ще одним важливим зауваженням є те, що кількість операцій читання буде більшою, ніж кількість операцій запису.

Все розподілене сховище даних може мати тільки дві з трьох властивостей: узгодженість даних, стійкість і доступність [36]. Саме тому на першому етапі моделювання баз даних (БД) для СППВПОН необхідно визначити, які з цих властивостей є найбільш важливими [37], [38].

Властивість узгодженості даних означає, що в будь-який момент часу різні користувачі будуть отримувати одні і ті ж дані [36], [37]. Ця властивість має важливе значення для бази даних СППВПОН.

Властивість стійкості означає, що система продовжує працювати навіть при виході з ладу деяких серверів. Ця якість також відіграє значну роль для бази даних, адже стабільність СППВПОН у всіх можливих сценаріях є однією з основних вимог.

Властивість доступності є гарантією того, що на кожен запит буде отримана відповідь, без гарантії, що ця відповідь є останньою версією даних [36] і [37]. Для СППВПОН вона менш важлива, ніж властивості узгодженості і стійкості, так як число запитів на запис значно менше, ніж число запитів на читання, отже, дані будуть оновлюватися рідше, ніж прочитані. Крім того, багато нереляційні бази даних мають механізм реплікації "Господар-слуга", який працює за принципом, що «господар» може записувати і читати файли, а «слуги» призначені тільки для резервного копіювання і, отже, з сервера пристрою «слуги» дозволені тільки операції читання [24], [37]. Ці два фактори мінімізують ймовірність появи проблеми доступності.

Отже, належна продуктивність бази для СППВПОН в першу чергу залежить від властивостей узгодженості даних і стійкості.

З огляду на вимоги до системи СППВПОН, було сформульовано набір критеріїв, які дозволяють об’єктивно оцінити придатність різних типів і технологій нереляційних баз даних. Ці критерії враховують як функціональні вимоги до бази даних системи, так і технічні.

Основними критеріями оцінки придатності різних типів та технологій нереляційних баз даних для основних задач системи є:

1. Гнучкість структури даних – можливість зберігати динамічні, вкладені або неоднорідні записи без жорсткого схемного обмеження [18];

2. Продуктивність – ефективність обробки великої кількості запитів, що критично для інформаційних високонавантажених систем [39];

3. Масштабованість – здатність бази розширюватися горизонтально без втрати стабільності або продуктивності [40];

4. Наявність підтримки транзакцій – забезпечення узгодженості даних у випадку складних операцій або оновлень;

5. Наявність механізмів реплікації та відмовостійкісті – можливість автоматичного дублювання даних і продовження роботи у разі відмови окремих вузлів [30], [31];

6. Підтримка геопросторових запитів – здатність ефективно працювати з координатами та просторовими операціями [41];

7. Інтеграційна сумісність – доступність інтерфейсів і механізмів для взаємодії з іншими сервісами, включно з хмарними та аналітичними платформами.

2.2. Обґрунтування вибору технології MongoDB

На сьогодні в сучасних високонавантажених системах використовуються такі типи нереляційних баз даних (табл. 2.1.): бази даних з широкою колонкою, ключ-значення і документально-орієнтовані бази даних [38], [42], [43] і [44]. Кожен із цих типів має свої переваги залежно від характеру даних та вимог до масштабованості, швидкодії й гнучкості структури.

Бази даних з широкою колонкою найбільш схожі на реляційні бази даних. Цей тип бази даних також використовує рядки та таблиці, але ім’я та кількість колонок можуть відрізнятися від рядка до рядка в таблиці. Це забезпечує більшу гнучкість у порівнянні з класичними SQL-схемами, особливо при роботі з великими наборами різнорідних даних. Прикладом такої бази є Apache Cassandra, яка використовується у високонавантажених розподілених системах.

Таблиця 2.1.

Результати порівняльного аналізу технологій нереляційних баз даних

Критерій	Cassandra	MongoDB	Redis
Тип бази даних	Широка колона	Документо-орієнтованість	Ключ-значення
Мова програмування	Java	C++	C
Призначення	Зберігання великих обсягів даних	Робота з документами і динамічною структурою	Обробка швидко змінюваних даних
Наявність ACID-транзакцій	Ні	Так (з версії 4.0)	Так
Модель реплікації	Децентралізована	"Слуга-Господар"	"Слуга-Господар"
CAP-властивості	Доступність і стійкість	Стійкість і узгодженість	Стійкість і узгодженість
Геопросторові функції	Так	Так	Так
Масштабованість	Горизонтальна через партиціонування	Горизонтальна через шардінг	Вертикальна, обмежена
Гнучкість структури даних	Помірна	Висока	Низька
Швидкодія при складних запитах	Низька	Висока	Низька
Підтримка вкладених структур	Ні	Так	Ні
Багатопоточність	Обмежена	Повноцінна	Обмежена

Найпростішим типом є сховище ключ-значення. Кожен елемент даних в базі даних зберігається у вигляді пари ключ-значення, що складається з імені атрибута

(«ключ») і значення [38]. Такі бази забезпечують надзвичайно швидкий доступ до інформації, однак мають обмежену функціональність у випадках складних зв’язків між даними. Вони здебільшого використовуються в кешах або як частина гібридних архітектур. Прикладом є Redis або Amazon DynamoDB.

Документно-орієнтовані БД розглядаються як перехід від простих БД «ключ-значення» до більш складних БД, які можуть зберігати пару «ключ-документ». Ця БД набула популярності завдяки тому, що вони не мають жорсткого обмеження структури [38]. Це робить їх особливо зручними для зберігання об’єктів із вкладеною або змінною структурою, що актуально, наприклад, для опису пошкоджених будівель у системі СППВПОН. Прикладами таких баз є MongoDB і Couchbase.

У табл. 2.1 наведені результати порівняльного аналізу найбільш поширених нереляційних БД, які найкраще зарекомендували себе серед користувачів високонавантажених розподілених інформаційно-комунікаційних систем різного призначення по критеріях визначених раніше [37], [38], [42], [45], [46].

За результатами аналізу нереляційних баз даних Cassandra, Redis та MongoDB, та враховуючи специфіку СППВПОН, зроблено висновок про найбільш доцільне використання MongoDB як бази даних для СППВПОН.

MongoDB – це документоорієнтована база даних з відкритим вихідним кодом, яка призначена для зберігання великих обсягів даних і дозволяє працювати з ними дуже ефективно. Оскільки дані зберігаються в документах, а не в таблицях, MongoDB класифікується як нереляційна БД.

Ця база даних працює наступним чином [47], [48]:

1. Середовище MongoDB надає сервер, на якому ви можете створювати кілька баз даних;
2. Дані зберігаються в колекціях і документах (рис. 2.1.);
3. Колекції містять документи, а ті, в свою чергу, містять дані. Одна колекція може включати кілька документів, а відсутність попередньо визначеної схеми означає, що один документ не обов'язково схожий на інший;

4. Документи дозволяють зберігати вкладені дані, що дозволяє створювати складні зв'язки між даними та зберігати їх в одному документі. Це робить обробку та пошук даних ефективними;

5. Документи створюються за допомогою полів. Поля є парами ключ-значення.

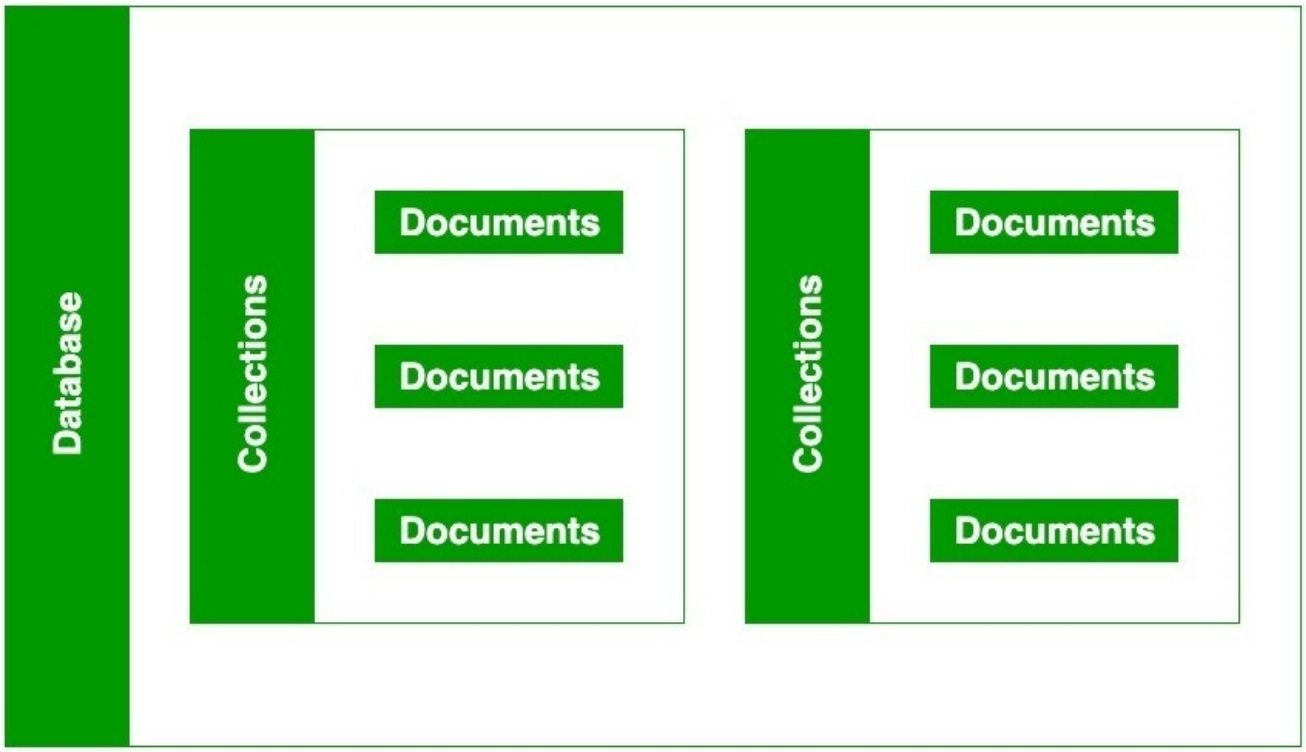

Рисунок 2.1. Ієрархія зберігання даних у MongoDB

MongoDB часто застосовується для створення веб-додатків, обробки великих обсягів даних, роботи з геопросторовими даними та IoT-систем [49]. Ця база даних особливо корисна в проєктах, де структура даних може змінюватися або потрібна висока швидкість виконання запитів.

Основними перевагами MongoDB, які визначили вибір, є [43], [44]:

– Індексація;
– Масштабованість;
– Реплікація;
– Наявність підтримки документо-орієнтованої моделі баз даних;
– Багатопоточність;

– Підтримка транзакцій ACID;
– Геопросторові функції;
– Інтеграція з хмарними платформами;
– Можливість взаємодії з БПЛА і супутниками.

Індексація забезпечує високу швидкість виконання запитів. Кожен документ має ідентифікатор, який використовується для створення унікального індексу. Він може складатися з декількох полів з однієї колекції і будується з використанням структури B-дерева, так як B-дерево працює швидше для запиту на читання, ніж для запиту на запис [44] і [47]. Документ ідентифікується унікальним ключем документа, який може бути комбінацією ідентифікатора документа та часу його створення.

Індексація в MongoDB підтримує різні типи індексів, включаючи складені (з декількох полів), текстові та геопросторові індекси [50]. Текстові індекси дозволяють ефективно виконувати пошук у текстових полях, наприклад, за ключовими словами чи фразами, що особливо корисно для аналізу контенту. Геопросторові індекси використовуються для роботи з координатами та забезпечують швидке виконання геозапитів, таких як пошук об’єктів у радіусі або побудова маршрутів.

Горизонтальна масштабованість MongoDB забезпечується шардингом. Мається на увазі розподіл даних на декількох серверах [47], [48]. Шардинг дозволяє MongoDB динамічно розподіляти запити між різними серверами, знижуючи навантаження на кожен окремий вузол [51]. Це забезпечує стабільну продуктивність навіть при збільшенні обсягів даних або кількості запитів. Основні компоненти шардингу включають конфігураційні сервери для зберігання метаданих про шард-ключі та розподіл даних, а також маршрутизатори (mongos), які спрямовують клієнтські запити до відповідного шарду. Завдяки цьому, додавання нових серверів до кластеру виконується без необхідності зупиняти роботу системи. Такий підхід ідеально підходить для великих високонавантажених додатків, які потребують масштабованої інфраструктури для обробки великих обсягів даних у реальному часі.

Завдяки функції реплікації, MongoDB забезпечує високу доступність і резервування даних. Кілька копій даних створюються і відправляються на інший сервер таким чином, щоб у разі виходу з ладу одного сервера можна було отримати дані з іншого [52].

Реплікація реалізується через реплікаційні набори, які складаються з одного основного вузла і декількох вторинних вузлів. Основний вузол обробляє всі операції запису, тоді як вторинні вузли автоматично синхронізуються з основним, копіюючи дані в реальному часі. У випадку збою основного вузла відбувається автоматичне перемикання, і один із вторинних вузлів стає новим основним без втручання адміністратора. Крім того, вторинні вузли можна налаштувати для виконання операцій читання, що допомагає розподілити навантаження на кластер. Такий підхід забезпечує не лише надійність і безперебійність роботи, але й підвищує продуктивність у системах із високим рівнем паралельних запитів [53].

Наявність підтримки документоорієнтованої моделі баз даних означає, що одна колекція може містити різні типи документів, причому ці документи можуть мати різну кількість полів, зміст і розмір. Документоорієнтована модель дозволяє зберігати дані у вигляді документів. У цих документах дані організовані у полях типу “ключ-значення”. Такий підхід забезпечує більшу гнучкість у роботі з даними, ніж традиційні реляційні бази даних, де використовується сувора структура таблиць [42], [45].

Документоорієнтована модель MongoDB дозволяє зберігати складні структури даних у форматі BSON (Binary JSON), який підтримує вкладені об’єкти та масиви [54]. Це значно спрощує роботу з ієрархічними даними, такими як списки, карти або багаторівневі зв’язки, без необхідності створення додаткових таблиць чи виконання складних JOIN-запитів, як у реляційних базах. Завдяки цьому дані зберігаються у цілісному вигляді в межах одного документа, що забезпечує швидкий доступ і спрощує обробку. Такий підхід ідеально підходить для динамічних додатків, у яких структура даних може змінюватися або розширюватися під час розробки чи використання.

Багатопоточність у MongoDB забезпечує одночасну обробку великої кількості запитів, що є критично важливим для високонавантажених систем. MongoDB використовує асинхронний ввід-вивід і багатопоточну архітектуру, яка дозволяє ефективно розподіляти завдання між ядрами процесора [55]. Це забезпечує можливість одночасного виконання операцій запису, читання та індексації, зберігаючи високу продуктивність навіть за умов значного навантаження. Завдяки цьому MongoDB може обробляти тисячі запитів на секунду, що робить її придатною для реального часу додатків, таких як аналітичні платформи чи онлайн-сервіси.

Гнучка модель блокувань також сприяє оптимізації багатопоточності. MongoDB використовує блокування на рівні документа, а не всієї колекції чи бази даних, як це було в ранніх версіях [56]. Це означає, що паралельні операції можуть виконуватися незалежно, за винятком тих, що стосуються одного й того самого документа. Такий підхід мінімізує конфлікти між потоками та дозволяє максимально використовувати обчислювальні ресурси. Крім того, багатопоточна архітектура MongoDB інтегрується з сучасними платформами, що підтримують розподілені системи, забезпечуючи стабільну роботу навіть у великих кластерах.

MongoDB забезпечує підтримку транзакцій ACID [57]. Ця функція, впроваджена у версії 4.0, забезпечує атомарність, узгодженість, ізоляцію та довговічність (рис. 2.2.). Завдяки транзакціям всі операції в межах одного або декількох документів виконуються як єдиний логічний блок. Якщо одна з операцій не вдається, всі інші скасовуються, забезпечуючи, що дані залишаються у передбачуваному стані.

Транзакції в MongoDB підтримують як локальні, так і розподілені операції [59]. Локальні транзакції виконуються в межах одного сервера чи шарда, тоді як розподілені транзакції дозволяють виконувати операції, що охоплюють кілька шардованих вузлів. Це досягається завдяки механізму двофазного підтвердження, який гарантує, що всі залучені вузли синхронно закінчать операцію. Крім того, транзакції MongoDB легко інтегруються з іншими функціями бази даних, такими як індексація чи реплікація, що забезпечує високу продуктивність і стабільність навіть у складних сценаріях використання.

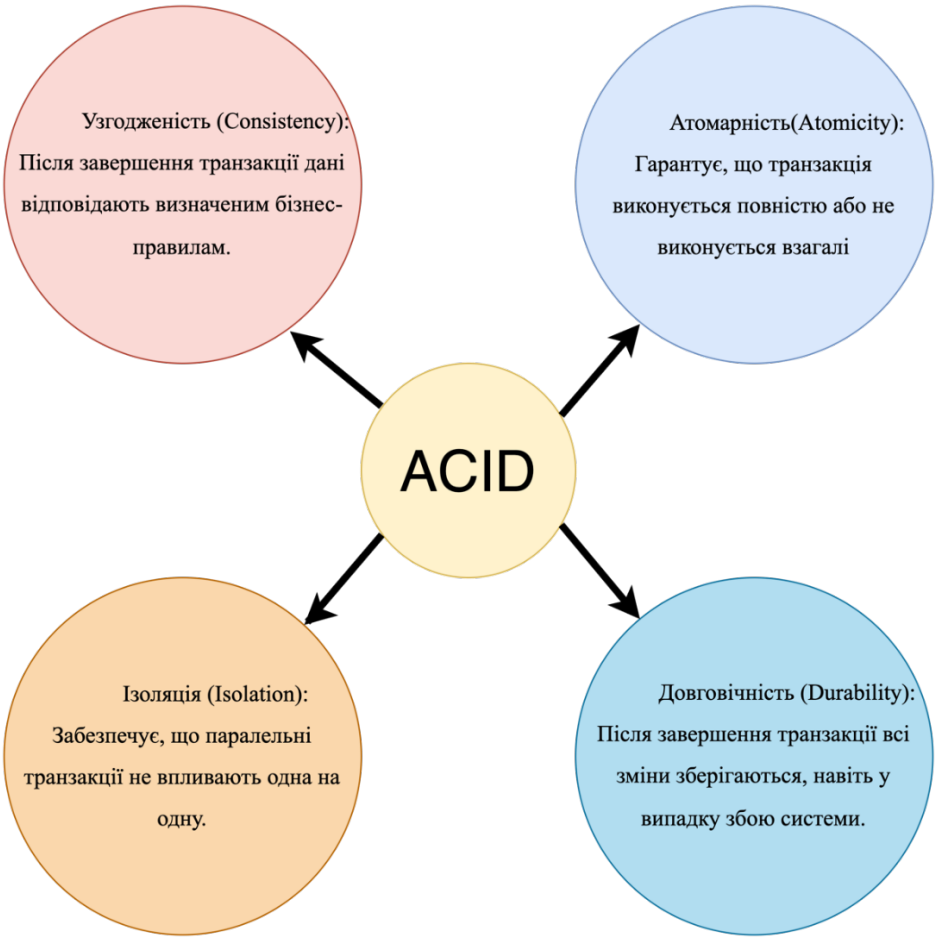

Рисунок 2.2. Властивості ACID та їх визначення

Геопросторові запити легко інтегруються з іншими функціями MongoDB [61]. Наприклад, вони можуть комбінуватися з агрегативними операціями для виконання складного аналізу, такого як підрахунок кількості об’єктів у певній зоні або визначення середньої відстані між точками. Дані зберігаються у спеціалізованих форматах, таких як GeoJSON, який дозволяє описувати точки, лінії, багатокутники та їхні комбінації. MongoDB також підтримує запити з декартовими координатами (2D) або сферичними координатами (2D-sphere), що забезпечує точність і продуктивність для різноманітних застосувань.

MongoDB легко інтегрується з хмарними платформами, такими як AWS, Azure та Google Cloud Platform, завдяки MongoDB Atlas [62]. MongoDB Atlas – це хмарний сервіс, що дозволяє створювати, масштабувати та управляти базами даних у хмарі без складної інфраструктури. Користувач може за лічені хвилини

налаштувати кластер, вибрати географічне розташування для зберігання даних і встановити рівень доступності. Це особливо корисно для додатків, які потребують глобальної доступності, низької затримки та дотримання регуляторних вимог, таких як GDPR.

Інтеграція з хмарними платформами забезпечує високу гнучкість і безпеку. MongoDB Atlas автоматично виконує балансування навантаження, забезпечує резервне копіювання та підтримує багаторегіональне розгортання для покращення доступності даних [63]. Крім того, платформа пропонує вбудовані засоби шифрування даних як під час передачі, так і на диску, а також розширені інструменти моніторингу продуктивності. Завдяки цьому MongoDB у хмарі може ефективно обслуговувати сучасні веб- і мобільні додатки, які потребують масштабованої та надійної інфраструктури.

MongoDB є потужним інструментом для роботи з даними, отриманими з безпілотних літальних апаратів (БПЛА) та супутників. Завдяки підтримці геопросторових індексів і можливості зберігати вкладені структури, MongoDB ідеально підходить для обробки великих обсягів географічної та візуальної інформації. Зображення, відеозаписи та інші дані, зібрані дронами або супутниками, можуть бути збережені у MongoDB як документи разом із метаданими (час зйомки, координати, висота польоту тощо) [64]. Це дозволяє швидко виконувати запити, наприклад, визначати об’єкти в заданому радіусі або аналізувати зміни в місцевості за певний період.

Інтеграція MongoDB із системами БПЛА та супутниками забезпечує реальний час обробки даних. Дані, отримані з дронів, можуть одразу передаватися в систему, де MongoDB використовується як центральне сховище для аналізу та доступу [65]. Завдяки своїй масштабованості MongoDB підтримує високі обсяги одночасних запитів, дозволяючи аналітичним платформам або системам моніторингу працювати без збоїв навіть за умов високих навантажень. Крім того, MongoDB легко інтегрується з хмарними платформами для зберігання великих файлів, таких як Amazon S3, та інструментами аналізу великих даних, такими як Apache Spark, що робить її універсальним рішенням для складних завдань.

На основі наведених характеристик узагальнено ключові переваги MongoDB та можливості їх застосування в межах СППВПОН [Додаток А].

Таким чином, вибір MongoDB як бази даних для СППВПОН є цілком виправданим. Ця система забезпечує гнучке зберігання даних завдяки документоорієнтованій моделі, швидкий доступ до інформації через ефективну індексацію, високу масштабованість завдяки шардингу, стійкість до збоїв через реплікацію, а також підтримку геопросторових запитів, що особливо важливо для роботи з просторовими даними. Додатково, інтеграція з хмарними платформами робить MongoDB зручним і надійним рішенням для потенційно високонавантаженої інформаційної СППВПОН.

2.3. Проєктування структурної моделі документів для MongoDB

Процес моделювання бази даних у системах з нереляційною базою даних передбачає проектування структурної моделі документів. Під структурною моделлю документів мається на увазі спосіб організації даних у форматі JSON-документів у колекціях MongoDB. Така модель визначає, які саме поля містяться в кожному документі, яким чином у ньому зберігаються пов’язані об’єкти та додаткова інформація, а також наскільки така структура відповідає потребам системи.

Документоорієнтована модель даних дозволяє уникнути складних зв’язків між таблицями, характерних для реляційного підходу, й натомість ізолювати пов’язану інформацію в межах одного документа. [66]. Такий підхід спрощує читання, збільшує швидкість обробки запитів та є оптимальним для систем з різнорідними, часто змінюваними даними, як у випадку опису пошкоджених будівель, міської статистики або активності користувача.

Відповідно до завдань, які виконує система, а також на основі визначених технічних і структурних вимог, у базі даних буде реалізовано набір основних колекцій документів, кожна з яких відповідає окремому інформаційному аспекту СППВПОН. Основні колекції документів для бази даних СППВПОН вказані в табл. 2.2.

Таблиця 2.2.

Основні колекції документів бази даних СППВПОН

Назва	Зміст
Пошкоджені будівлі	Базові характеристики, пошкодження, технічний стан
Користувачі	Ідентифікація та роль користувачів
Історія запитів	Час, тип і об’єкт запитів
Статистика пошкоджень	Пошкодження і втрати на рівні населених пунктів

Таблиця 2.3.

Структура документа в колекції «Статистика пошкоджень»

Поле	Опис	Тип даних (укр / англ)
Населений пункт	Назва міста або населеного пункту	Рядок / String
Відсоток пошкоджених об’єктів	Частка пошкоджених об’єктів	Число з плаваючою комою / Double
Відсоток зруйнованих об’єктів	Частка непридатних до відновлення об’єктів	Число з плаваючою комою / Double
Орієнтовна вартість збитку	Оціночна сума матеріальних збитків	Число з плаваючою комою / Double
Кількість постраждалих мешканців	Особи, які втратили житло	Ціле число / Integer
Кількість загиблих жителів	Втрати серед цивільного населення	Ціле число / Integer

Колекція «Пошкоджені будівлі» запропонована як основна колекція документів у базі даних системи. Кожен документ у цій колекції відповідає одному пошкодженій або зруйнованому об’єкту нерухомості. Документ включає ключові

поля, що відображають базові характеристики будівлі, види пошкоджень та її технічний стан [Додаток Б].

Таблиця 2.4.

Структура документа в колекції «Користувачі»

Поле	Опис	Тип даних (укр / англ)
ID	Унікальний ідентифікатор користувача	Рядок / ObjectId
Ім’я	Повне ім’я користувача	Рядок / String
Роль	Тип користувача (наприклад, експерт, представник органу влади, інвестор)	Рядок / String
Організація	Організація, яку представляє користувач	Рядок / String
Електронна пошта	Контактна електронна адреса	Рядок / String
Дата реєстрації	Дата створення облікового запису	Дата / Date
Активний	Ознака, чи обліковий запис активний	Булеве / Boolean

Більшість суттєвих даних про основні характеристики пошкодженої будівлі надходять з безпілотних літальних апаратів, систем супутникового спостереження та повідомлень ЗМІ [24], [67]. Тому вважається, що фото і відеофайли найефективніше зберігаються на сторонньому сервісі зберігання даних типу Amazon S3, а шляхи до відповідних файлів рекомендується зберігати в самій базі даних. Дані про місцезнаходження надходять з GPS.

Технічні характеристики об'єкта визначаються і додаються фахівцями в спеціалізовані державні системи, які генерують інформацію про кожну окрему будівлю.

Колекція «Міська статистика» містить агреговану інформацію про наслідки руйнувань в окремих населених пунктів. Ця інформація використовується для аналітики, прийняття рішень та планування відновлення.

Колекція «Користувачі» містить інформацію про зареєстрованих користувачів СППВПОН. Ці дані дозволяють ідентифікувати тип користувача, історію його активності, та формують основу для подальшого використання машинного навчання для інтелектуального управління кешем.

Колекція «Історія запитів» зберігає інформацію про всі запити, зроблені користувачами СППВПОН. Ця інформація використовується як для аналітики, так і для навчання моделей машинного навчання, що приймають рішення про кешування даних.

Таблиця 2.5.

Структура документа в колекції «Історія запитів»

Поле	Опис	Тип даних (укр / англ)
ID	Унікальний ідентифікатор запиту	Рядок / ObjectId
ID користувача	Зовнішній ключ на користувача, який виконав запит	Рядок / ObjectId
Час запиту	Дата та точний час запиту	Дата і час / DateTime
Тип запиту	Тип операції (читання або запис)	Рядок / String
ID об’єкта	Зовнішній ключ на об’єкт, до якого стосувався запит	Рядок / ObjectId
Тип об’єкта	Категорія об'єкта: лікарня, школа тощо	Рядок / String
Місто	Місто або населений пункт, в якому знаходиться об'єкт	Рядок / String

Запропоновані структури колекцій не є ізольованими, а навпаки логічно пов’язані між собою. На рис. 2.3. представлено логічні зв’язки між усіма запропонованими основними колекціями бази СППВПОН.

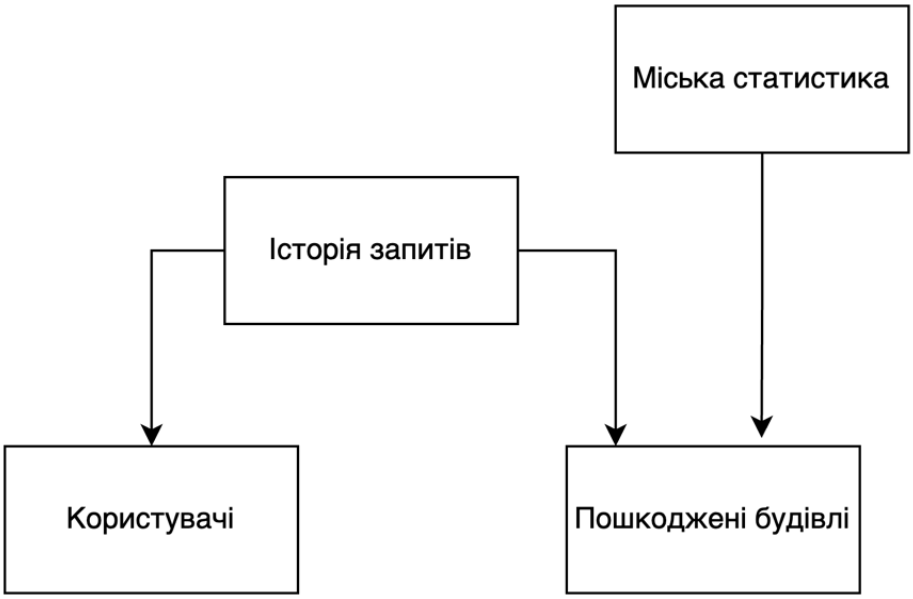

Рисунок 2.3. Логічні зв’язки між колекціями документів бази даних СППВПОН

Кожна з колекцій відіграє окрему роль у структурі зберігання даних. Колекція «Історія запитів» формує динамічний зв’язок між «Користувачі» та «Пошкоджені будівлі», зберігаючи релевантну інформацію про запити до об’єктів від користувачів. «Міська статистика» слугує джерелом додаткового контексту для колекції «Пошкоджені будівлі», зокрема щодо загальної ситуації в конкретному регіоні або районі.

Таким чином, на рис. 2.4 узагальнено процес обробки запиту до спроектованої бази даних СППВПОН, що охоплює як операції читання, так і запису даних.

Схема відображає основні етапи: отримання запиту, визначення відповідних колекцій документів у MongoDB, формування критеріїв вибірки (фільтри, індекси, поля) та визначення типу запиту.

У разі читання відбувається: виконання операції пошуку, формування отриманих даних: агрегація, сортування, доповнення, повернення результату користувачу чи модулю системи.

У разі запису здійснюється: виконання операції запису, підтвердження її успішності від MongoDB. Фінальним кроком є збереження інформації про виконаний запит до колекції «Історія запитів». Такий підхід дозволяє забезпечити надійну, структуровану та масштабовану взаємодію з даними в межах інформаційної високонавантаженої СППВПОН.

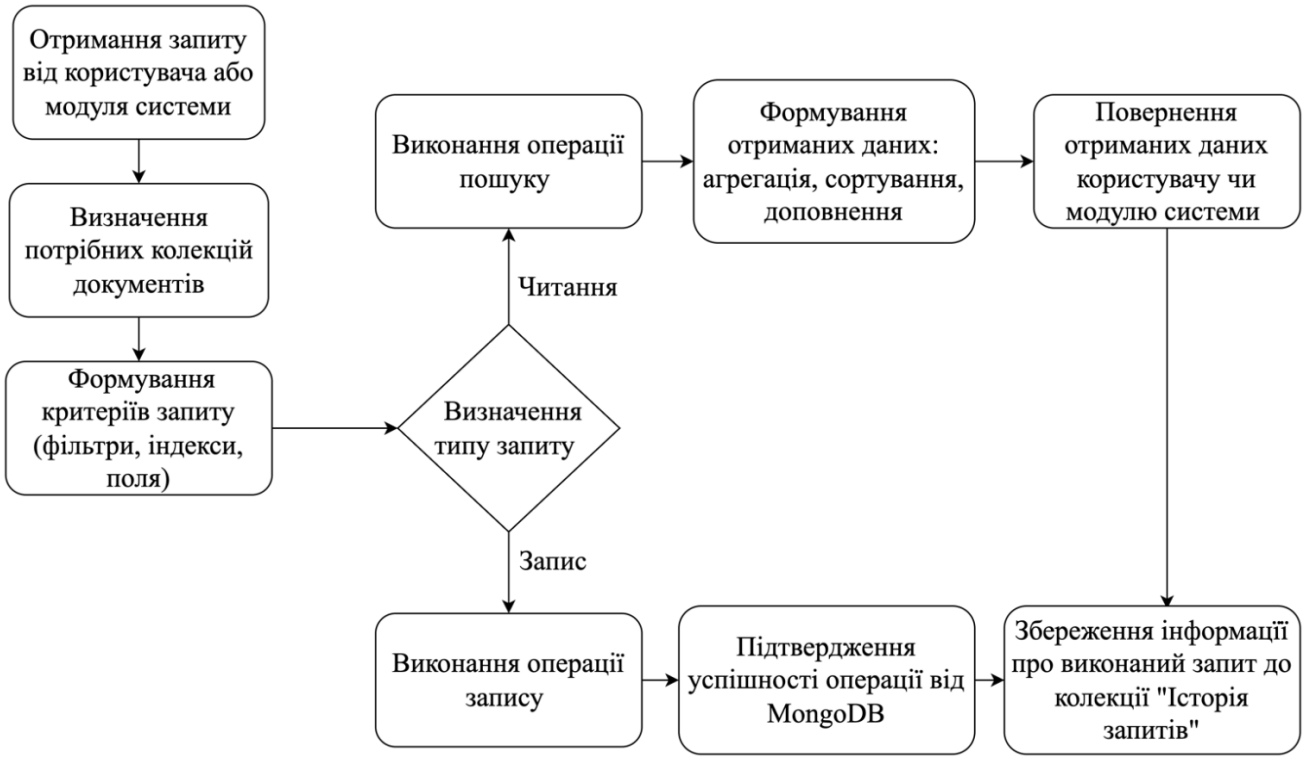

Рисунок 2.4. Схема обробки запиту в базі даних СППВПОН

2.4. Інтелектуальне кешування як відповідь на обмеження існуючих методів кешування

2.4.1. Роль кешування в інформаційних високонавантажених системах

У контексті інформаційних високонавантажених систем, де обробляються тисячі або навіть мільйони запитів щодня, кешування виконує не лише роль оптимізації, а стає критичною умовою стабільності та масштабованості системи. Підвищення ефективності доступу до даних, зменшення затримок, балансування навантаження та забезпечення стійкості – усе це напряму залежить від правильно організованого механізму кешування.

Одним із ключових викликів у таких системах є баланс між швидкістю доступу до даних та їх узгодженістю. Наприклад, при великій кількості одночасного доступу до даних одна й та сама інформація може бути затребувана десятки разів протягом хвилини. У таких випадках кешування дозволяє уникнути надмірних

запитів до основної бази даних, що істотно знижує навантаження на неї та підвищує загальну пропускну здатність системи.

Крім цього, кешування відіграє ключову роль у зменшенні затримки відповіді (*latency*) [68]. Час доступу до даних із пам’яті або локального кешу в десятки або навіть сотні разів менший, ніж з бази даних, особливо якщо вона розміщена на іншому сервері або у хмарній інфраструктурі. Це особливо важливо для інтерфейсів, які потребують миттєвого реагування, або сервісів, які забезпечують доступ до великих масивів аналітичної інформації.

Ще одним критичним аспектом є забезпечення стійкості в періоди пікових навантажень. Наприклад, у моменти великій кількості одночасних звернень до даних, кеш дозволяє витримати навантаження без значної втрати продуктивності [69]. У протилежному випадку, коли кешування відсутнє або реалізоване неефективно, система ризикує втратити стабільність або навіть припинити обслуговування запитів.

У системах, орієнтованих на відбудову, таких як СППВПОН, кешування також дозволяє реалізовувати інтелектуальні сценарії персоналізації. Наприклад, часто повторювані запити конкретних користувачів або регіональні дані можуть бути передчасно збережені в кеші для швидшого доступу до них пізніше. Таким чином, кешування стає не лише засобом підвищення швидкодії, але й інструментом адаптації системи до реальних потреб користувачів.

Також важливим є питання відмовостійкості: при тимчасовій недоступності основного сховища система може повернути останні збережені в кеші дані, що дозволяє уникнути повної недоступності сервісу. Це критично важливо для державних або кризових систем, де безперервність доступу є умовою ефективності роботи.

У системах із мікросервісною або хмарною архітектурою дедалі ширше використовується розподілене кешування, наприклад на базі Redis Cluster або Memcached [70], [71]. Такий підхід дозволяє тримати кеш паралельно на кількох вузлах, синхронізувати дані між ними, зберігати стійкість до збоїв окремих компонентів та забезпечувати горизонтальне масштабування [72].

Таким чином, основні функції кешування у інформаційних високонавантажених системах включають:

1. Зменшення навантаження на БД – кешування дозволяє уникати повторних звернень до основного сховища при частих запитах одних і тих самих даних;
2. Швидші відповіді на запити – отримання даних з кешу відбувається в рази швидше, ніж з диска або віддаленого сервера;
3. Забезпечення стабільної роботи під час пікових навантажень шляхом ролі кеш як буферу, що допомагає системі справлятися з високим трафіком;
4. Адаптація до поведінки користувачів – зберігання персоналізованих або регіональних даних дозволяє системі підлаштовуватися під запити в реальному часі;
5. Підтримка роботи при збоях основного сховища даних – кеш дозволяє тимчасово обробляти запити навіть за відсутності доступу до БД;
6. Зниження витрат у хмарній інфраструктурі – зменшення трафіку і кількості звернень до БД дозволяє оптимізувати використання ресурсів.
7. Оптимізація обміну даними між сервісами – кеш зменшує затримки при взаємодії мікросервісів і мінімізує дублювання запитів.
8. Збереження результатів складних обчислень – попередньо обчислені або фільтровані дані не потребують повторного розрахунку.

2.4.2. Обмеження існуючих методів та потреба в інтелектуальних підходах

У більшості інформаційних систем переважно застосовуються методи кешування, що базуються на простих евристиках та припущеннях щодо поведінки користувачів або характеру запитів. На основі аналізу таких підходів було виконано порівняльний огляд найбільш поширених методів кешування [73], [74], [75], [Додаток В].

Ці методи застосовувались упродовж десятиліть, особливо в умовах обмежених обчислювальних ресурсів і стабільних шаблонів запитів. Проте в реаліях сучасних розподілених, динамічних і високонавантажених інформаційних систем виникає все більше ситуацій, у яких ці підходи стають недостатніми або навіть

неефективними. Нижче наведено групи обмежень, що унеможливлюють використання класичних методів кешування у нових умовах.

1. Відсутність реакції до змін в системі. Алгоритми на кшталт LRU та LFU працюють за фіксованими правилами й не адаптуються до змін у поведінці користувачів, рівні навантаження чи зміні структури даних. Не мають механізму самонавчання або автоматичного пристосування до нових умов. Наприклад, у період пікових навантажень користувачі можуть масово запитувати дані про об’єкти в одному місті. Проте існуючі стратегії не здатні підлаштовуватися на нові пріоритети, зберігаючи в кеші об’єкти, що більше не використовуються [76];

2. Ігнорування різниці між користувачами. Сучасні інформаційні системи, зокрема такі, як СППВПОН, обслуговують різні групи користувачів – державні органи, архітектори, волонтери, представники міжнародних організацій. Кожна з цих груп має власні запити, різні періоди активності та важливість інформації. У класичних методах управління кешем усі користувачі вважаються однаковими, що призводить до кешування нерелевантних для конкретного користувача об’єктів і частих промахів кешу [77];

3. Ігнорування контексту об’єктів. Ні LFU, ні LRU не здатні розрізняти, які об’єкти мають критично важливе значення і працюють лише з лічильниками чи часовими мітками, ігноруючи тип об’єкта, географічну прив’язку, його зв’язок з іншими даними чи роль у процесі прийняття рішень. Унаслідок цього в кеші можуть залишатися об’єкти, які втратили актуальність, тоді як справді важливі дані видаляються з кешу. LFU може залишити в кеші менш важливий об’єкт тому, що до нього стабільно часто зверталися раніше [78].

4. Обмеження у передбаченні майбутніх запитів. Класичні методи здатні лише реагувати на те, що вже сталося, тобто вони приймають рішення на основі минулої активності, не враховуючи, що багато систем потребують проактивних дій. Без здатності прогнозування в кеш додається те, що було корисним раніше, а не те, що буде потрібне далі.

5. Неврахування тимчасових піків активності. Реальні системи стикаються зі піками активності, які мають тимчасовий характер. Кеш має адаптуватися до таких

хвиль, тимчасово збільшуючи пріоритетність певних об’єктів. Класичні існуючі методи не розрізняють постійну популярність і короткочасні сплески, тому реагують із затримкою або взагалі не реагують [79].

6. Відсутність механізмів самонавчання. Найголовніше обмеження класичних стратегій полягає у відсутності навички до навчання на основі існуючої історії запитів. Вони не аналізують поведінку системи, не виявляють закономірностей, зв’язків між запитами різних користувачів і не оновлюють свої правила роботи. На відміну від інтелектуальних підходів, існуючі методи управління кешем залишаються незмінними, навіть якщо ситуація повторюється багато разів [80].

Усі перелічені обмеження вказують на непридатність традиційних методів кешування для задач, де важливо враховувати:

– особливості поведінки різних груп користувачів;

– контекст системи;

– зв’язки між об’єктами та даними в системі;

– динаміку запитів на основі історії.

Це підводить до необхідності впровадження інтелектуальних методів управляння кешем, що дозволяють:

– формувати рішення на основі багатьох ознак одночасно, зокрема типу об’єкта, його контексту, частоти доступу, пріоритету користувача чи об’єкта тощо;

– автоматично адаптуватися до змін у середовищі, змінюючи стратегії в реальному часі залежно від ситуації;

– прогнозувати запити на основі історичних патернів, виявляючи закономірності у зверненнях користувачів і завчасно готуючи відповідні об’єкти.

З урахування виявлених обмежень існуючих методів кешування та зростаючі можливості машинного навчання, постає потреба в новому, більш адаптивному підході до кешування в інформаційних системах. Особливо це актуально для СППВПОН, де кількість і характер запитів до об’єктів нерухомості постійно змінюються. У таких умовах ключовим стає створення інтелектуального методу управління кешем, що базується на моделях машинного навчання і враховує суттєві ознаки як користувачів, так і об’єктів нерухомості.

Висновки до розділу 2

1. Визначено структурні та технічні вимоги до бази даних СППВПОН, необхідні для реалізації основних функцій системи. Проаналізовано типові запити на запис і читання, які виконуватимуть користувачі з різними рівнями доступу, з метою врахування їх при виборі найбільш ефективної технології бази даних та побудові відповідної моделі даних. Показано, що для стабільної роботи системи в умовах високого навантаження критично важливою є підтримка властивостей узгодженості й стійкості з урахуванням принципів CAP-теореми у проєктуванні бази даних СППВПОН.

2. На основі аналізу сучасних типів нереляційних баз даних та їх відповідності вимогам для бази даних СППВПОН обґрунтовано доцільність використання MongoDB як бази даних системи. Визначено ключові переваги MongoDB: гнучка документоорієнтована модель, висока масштабованість, підтримка геопросторових функцій, ACID-транзакцій, а також розвинені механізми індексації. Сукупність цих властивостей робить MongoDB оптимальним рішенням для побудови надійної та ефективної бази даних СППВПОН.

3. Спроектовано структурну модель документів бази даних СППВПОН у формі чотирьох колекцій у MongoDB: «Пошкоджені будівлі», «Міська статистика», «Користувачі» та «Історія запитів». Сформовано логічну схему взаємозв’язків між ними як основу для проектування гібридного методу інтелектуального управління кешем, який дозволяє суттєво підвищити ефективність роботи з даними для СППВПОН.

4. Описано роль кешування в інформаційних високонавантажених системах. Показано, що ефективне управління кешем відіграє критичну роль у забезпеченні швидкого доступу до актуальних об’єктів і зменшенні навантаження на базу даних. Окреслено основні недоліки існуючих методів кешування та описано переваги інтелектуальних підходів.

РОЗДІЛ 3. ГІБРИДНИЙ МЕТОД ІНТЕЛЕКТУАЛЬНОГО УПРАВЛІННЯ КЕШЕМ СИСТЕМИ ПІДТРИМКИ ПРОЦЕСУ ВІДНОВЛЕННЯ ПОШКОДЖЕНИХ ОБ'ЄКТІВ НЕРУХОМОСТІ

У третьому розділі розроблено гібридний метод інтелектуального управління кешем, який вирішує три ключові задачі: оцінка доцільності додавання об'єкта до кешу, вибір об'єкта для видалення з заповненого кешу та періодична інвалідація неактуальних об'єктів у кеші; подано визначення методу управління кешем як основи для реалізації гібридного підходу; обґрунтовано застосування моделей машинного навчання – табличного Q-Learning та логістичної регресії для вирішення відповідних задач запропонованого методу інтелектуального управління кешем; запропоновано представлення стану об'єкта у вигляді вектора ознак; спроектовано додаткові колекції документів для бази даних системи підтримки процесу відновлення пошкоджених об'єктів нерухомості MongoDB, необхідні для коректної роботи запропонованих моделей машинного навчання; обґрунтовано доцільність використання Redis як основної технології кешування для системи; описано алгоритм роботи гібридного методу інтелектуального управління кешем; показано фінальну блок-схему його роботи із застосуванням моделей машинного навчання табличного Q-Learning і логістичної регресії, технології нереляційної бази даних MongoDB і технології кешування Redis; представлено оновлену модель системи з інтегрованим модулем інтелектуального кешування, що реалізовує розроблений гібридний метод.

3.1. Метод управління кешем

Метод управління кешем – це стратегія, що визначає:

– які дані доцільно додати до кешу;
– які об'єкти слід видалити у разі заповнення кешу;

– коли та яким чином оновлювати вміст кешу (інвалідація).

Інвалідація кешу – це процес визначення застарілих об’єктів в кеші з подальшим їх видаленням навіть якщо кеш ще не заповнений. Вона дозволяє підтримувати актуальність даних у кеші. Ця процедура може виконуватись періодично або за певних умов, зокрема при зміні даних в основному сховищі чи низькій ймовірності повторного запиту.

Ефективність методу управління кешем зазвичай оцінюється за такими метриками:

1. Відсоток кеш-хітів (*cache hit*) – частка запитів, при яких необхідні дані вже наявні в кеші;
2. Відсоток кеш-місів (*cache miss*) – частка запитів, при яких потрібні дані відсутні в кеші і мають бути завантажені з основного сховища.

У цій роботі розроблено гібридний метод інтелектуального управління кешем, який вирішує ключові задачі: оцінка доцільності додавання об’єкта до кешу, вибір об’єкта для видалення з заповненого кешу та періодична інвалідація неактуальних об’єктів у кеші.

3.2. Оцінка доцільності додавання об’єкта до кешу

3.2.1. Формалізація

Ця задача виникає щоразу, коли до системи надходить новий запит до певного об’єкта. Необхідно оцінити, чи варто зберігати відповідний об’єкт у кеші, якщо його там ще немає. Це рішення повинно враховувати не лише частоту звернень, а й тип об’єкта, його значення в конкретному контексті, історію взаємодій з користувачами системи і навіть ймовірність майбутніх запитів. Правильне розв’язання цієї задачі дозволяє уникати зайвих запитів до бази даних і підвищує швидкодію СППВПОН.

У рамках СППВПОН задача прийняття рішення щодо доцільності збереження нового об’єкта в кеші має критичне значення. Це пояснюється кількома ключовими факторами:

1. Обмеженість кешу. Оскільки кеш має фіксований розмір, кожне збереження нового об’єкта автоматично створює потенційну конкуренцію за ресурси з об’єктами, що вже присутні у кеші. Неправильне рішення призводить до зниження загального коефіцієнта попадання в кеш (*cache hit ratio*).

2. Динамічність запитів. У системі спостерігається зміна шаблонів доступу до об’єктів, що обумовлено, наприклад, часовими піками популярності, сезонними факторами або іншими неочікуваними ефектами. За таких умов традиційні методи управління кешем, що базуються на частоті запитів або давності останнього доступу, є неефективними.

3. Нерівноцінність об’єктів. У кеш можуть надходити об’єкти, які або запрошуються лише один раз, або, навпаки, мають високу ймовірність повторного доступу. Визначення потенційної користі від об’єкта для кешу вимагає прогнозу, що базується на складній комбінації ознак.

4. Вплив на довгострокову продуктивність. Кожне рішення щодо кешування впливає не лише на поточний запит, а й на послідовність майбутніх рішень, оскільки кеш змінюється. Тому для розв’язання цієї задачі важливо орієнтуватися не на миттєву вигоду, а на досягнення максимальної користі в довгостроковій перспективі.

Урахування вищезазначених аспектів зумовлює необхідність застосування методів машинного навчання, які здатні формувати адаптивну політику кешування з урахуванням довгострокових наслідків кожного рішення.

Задача визначення доцільності кешування певного об’єкта заздалегідь є прикладом задачі прийняття рішення в умовах невизначеності, де необхідно вибрати одну з двох можливих дій: кешувати об’єкт чи ні. Формально, це можна представити як бінарну задачу прийняття рішення, яка у найпростішій формі відповідає задачі класифікації з простором дій:

$$A = \{0, 1\},$$

де:

– $a = 0$ – дія не кешувати об’єкт;

– $a = 1$ – дія кешувати об’єкт.

Кожне рішення приймається на основі вектора ознак об’єкта $x = [x_1, x_2, \dots, x_n]$, де x_i – це числове або категоріальне значення, що описує поведінку об’єкта в системі (наприклад, частота запитів, час останнього доступу, розмір, пріоритет користувача, тощо).

Задача може бути формалізована у вигляді марковського процесу прийняття рішень (*Markov Decision Process, MDP*). Такий підхід дозволяє враховувати не лише поточний стан об’єкта, але й довгострокові наслідки прийнятого рішення [81].

Марковський процес визначається п’ятіркою:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle,$$

де:

– $\mathcal{S}$ – множина станів, які в цій задачі описуються вектором ознак об’єкта $x = [x_1, x_2, \dots, x_n]$;

– $\mathcal{A} = \{0,1\}$ – множина дій:

– $\mathcal{P}(s'|s,a)$ – ймовірність переходу в новий стан s' після виконання дії a у стані s;

– $\mathcal{R}(s,a) \in R$ – функція винагороди за виконання дії a в стані s;

– $\gamma \in [0,1]$ – коефіцієнт дисконтування, що визначає вагу майбутньої винагороди.

Приклад реалізації елементів MDP у системі інтелектуального кешування для СППВПОН:

– Стан s – це набір ознак об’єкта. Наприклад, $x = [x_1$:час останнього доступу, x_2:частота, x_3:розмір].

– Дія a – прийняте рішення: кешувати або ні.

– Винагорода r – наприклад, +1, якщо в майбутньому запит на об’єкт призвів до *cache hit*, і 0 або -1 у випадку *cache miss*.

– Стан переходу s' – новий набір ознак, після оновлення кешу.

– Ціль – знайти оптимальну політику $\pi^*(s)$, яка максимізує очікувану сумарну винагороду.

3.2.2. Обґрунтування вибору методу розв’язання

Для ефективного розв’язання задачі прийняття рішення щодо кешування нового об’єкта було проаналізовано різні підходи, які відрізняються складністю, адаптивністю та здатністю враховувати довгострокові наслідки. Нижче наведено порівняння трьох основних методів, які можуть бути застосовані в даному контексті.

1. Евристичні методи. Ці методи базуються на простих фіксованих правилах, які не враховують змін у поведінці користувачів або властивостей об’єктів:

– LRU – перевага надається об’єктам, які використовувалися найпізніше [82];
– LFU – перевага надається об’єктам із найвищою частотою доступу до них [83];
– Random – рішення про кешування або витіснення об’єкта робляться випадковим чином [75].

Такі методи не потребують навчання, однак мають обмежену здатність до адаптації та не враховують складні сценарії поведінки в системі.

2. Класичне машинне навчання**.** У цьому підході задача формалізується як бінарна класифікація: кожен об’єкт описується вектором ознак, а моделі (наприклад, логістична регресія, дерева рішень або градієнтний бустинг) навчаються визначати, чи буде об’єкт запитано повторно найближчим часом [84].

Хоча такі моделі можуть враховувати складні взаємозв’язки між ознаками, вони зазвичай оптимізуються за локальними метриками (наприклад, точністю класифікації), не беручи до уваги глобальний ефект на кеш.

3. Методи машинного навчання з підкріпленням (*Reinforcement Learning*). Такі методи дають змогу системі поступово вдосконалювати свої рішення на основі постійної взаємодії з середовищем: на кожному кроці система приймає рішення, отримує зворотний сигнал (винагороду) та коригує свою стратегію дій [85].

Основна перевага цього підходу – це можливість врахування довгострокових наслідків кожного рішення, що є критично важливим для підвищення загального коефіцієнта влучання в кеш у динамічному середовищі.

Порівняння даних підходів подано в табл. 3.1, де узагальнено їхні ключові характеристики.

Таблиця 3.1.

Порівняльна таблиця методів вирішення задачі

Критерій	Евристичні методи	Класичне машинне навчання	Табличний Q-Learning
Адаптація до змін	Немає	Часткова	Висока
Урахування довгострокової вигоди	Немає	Немає	Є
Врахування ознак об’єкта	Немає	Є	Є
Точність рішень після навчання	Низька	Помірна	Висока
Складність реалізації	Низька	Середня	Висока

З урахуванням динамічного характеру СППВПОН, потреби в адаптації до змін і довгострокової оптимізації стратегії кешування, у рамках цієї роботи обрано метод навчання з підкріпленням, а саме табличну реалізацію Q-Learning.

Ннавчання з підкріпленням (*Reinforcement Learning*) – це клас машинного навчання, в якому агент навчається приймати рішення, взаємодіючи з навколишнім середовищем.

Агент спостерігає стан (*state*), виконує дію (*action*), і отримує винагороду (*reward*), яка сигналізує, наскільки його дія була корисною. Основною задачею агента є навчитися політиці (*policy*), яка максимізує сумарну очікувану винагороду з часом [86].

Такий підхід дозволяє системі самостійно формувати політику прийняття рішень на основі накопиченого досвіду, з урахуванням відкладених винагород і поступової адаптації до змін у поведінці користувачів.

Для розв’язання задачі застосовується таблична реалізація алгоритму Q-Learning. У цьому підході функція корисності $Q(s, a)$ асоціює кожну пару «стан–

дія» із числовим значенням, яке відображає очікувану довгострокову винагороду при виконанні дії a у стані s і подальшому дотриманні оптимальної стратегії.

Оновлення значення Q-функції якості відбувається згідно з формулою Беллмана [87]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \cdot \max_{a'} Q\,(s_{t+1}, a') - Q(s_t, a_t) \right], \qquad (3.1)$$

де:

– s_t – поточний стан, вектор ознак об'єкта $x^{(t)} = \left[x_1^{(t)}, \dots, x_n^{(t)} \right]$);

– a_t – вибрана дія (0 або 1);

– r_t – винагорода, отримана після виконання дії;

– s_{t+1} – наступний стан після дії;

– $\alpha \in (0,1]$ – коефіцієнт навчання (*learning rate*);

– $\gamma \in [0,1]$ – коефіцієнт дисконтування (*discount factor*);

– $\max_{a'} Q\,(s_{t+1}, a')$ – оцінка найкращої майбутньої дії.

3.2.3. Представлення стану об'єкта у вигляді вектора ознак

Кожен стан s об'єкта задається як вектор ознак:

$$s \equiv x = [x_1, x_2, \dots, x_n],$$

де кожен елемент x_i характеризує об'єкт чи контекст його використання. Для ефективної роботи моделі табличного Q-Learning доцільно використовувати невелику кількість ознак, які мають максимальний вплив на прийняття рішення (табл. 3.2.).

Таблиця 3.2.

Ознаки, що використовуються для представлення стану в моделі

Позначення	Назва ознаки	Опис
x_1	Кількість унікальних користувачів	Скільки різних користувачів зверталися до об’єкта протягом доби
x_2	Загальна кількість запитів	Загальна кількість звернень до об’єкта за останні 24 години
x_3	Тип об’єкта	Наприклад, 0 – житловий, 1 – школа, 2 – лікарня тощо
x_4	Кількість годин з моменту останнього запиту	Відображає давність взаємодії з об’єктом
x_5	Пріоритетність об’єкта або зони	Наприклад, 0 – низький, 1 – середній, 2 – високий
x_6	Локація об’єкта	Категоріальна ознака району / регіону, або його кодоване значення
x_7	Роль користувача	Категоріальна ознака, що відображає тип користувача, який зробив запит до об’єкта
x_8	Час доби	Вказує на період доби, коли відбувся останній запит до об’єкта

У результаті такого підходу модель враховує як загальні характеристики об’єкта, зокрема частоту запитів та час останнього звернення, так і специфічні для СППВПОН – тип об’єкта, його локацію та пріоритет у контексті відновлення інфраструктури. Це дозволяє точніше прогнозувати корисність об’єкта в кеші з огляду на його функціональну роль та важливість для користувачів. За потреби

набір ознак може бути доповнений додатковими параметрами, такими як розмір об’єкта, час перебування в кеші або середній інтервал між запитами.

3.2.4. Роль Q-таблиці та її структура

У даній задачі, де для кожного запиту об’єкта потрібно ефективно вирішувати додавати його в кеш чи ні, застосування табличного Q-Learning дозволяє агенту накопичувати досвід. Q-таблиця містить для кожної унікальної комбінації параметрів стану s, що представляється вектором ознак та дії a значення $Q(s,a)$ – оцінку очікуваної сукупної винагороди.

Ця оцінка дозволяє формувати політику:

$$\pi(s) = \arg\max_{a} Q\,(s,a), \tag{3.2}$$

тобто агент обирає дію з максимальною оцінкою, що забезпечує оптимальне управління кешем з урахуванням майбутньої вигоди.

У рамках реалізації СППВПОН, Q-таблиця фізично зберігається у вигляді окремої колекції документів в базі даних MongoDB із назвою «Q-таблиця». Кожен запис цієї колекції відповідає унікальній парі «стан–дія» та містить значення корисності, яке постійно оновлюється в процесі навчання (табл. 3.3).

Важливо зауважити, що при оновленні Q-значення порівнюється лише значення з вектора ознак x, а ідентифікатори конкретних об’єктів (*object_id*) не включаються до цього вектора, що дозволяє узагальнювати знання для схожих об’єктів.

Колекція «Q-таблиця» формується на основі ознак, отриманих із колекцій «Історія Запитів» та «Пошкоджені будівлі» (рис. 3.1.). Отриманий вектор ознак використовується для формування стану s у парі Q(s,a).

Таблиця 3.3.

Структура документу в колекції «Q-таблиця»

Поле	Тип даних	Опис
s	Об’єкт або рядок	Комплексне представлення стану – вектор ознак $x = [x_1, x_2, \dots, x_n]$.
a	Ціле число	Дія: 0 – не кешувати, 1 – кешувати.
q_value	Дійсне число	$Q(s, a)$: Очікувана сукупна винагорода за виконання дії a у стані s

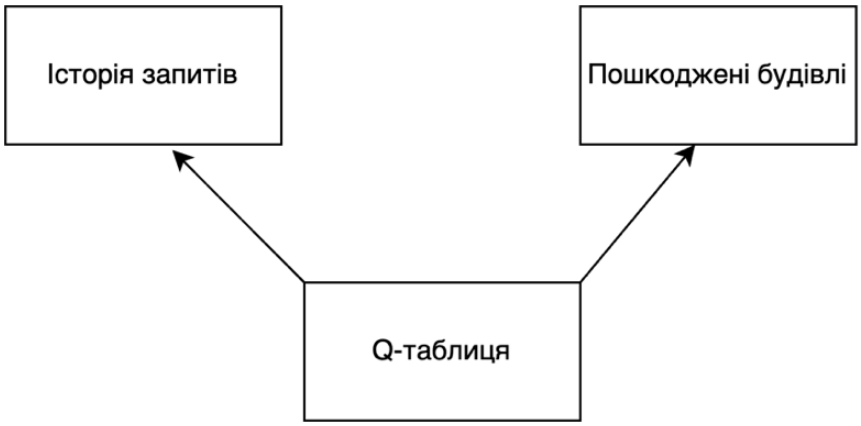

Рисунок 3.1. Логічні взаємозв’язки між основними колекціями документів в базі даних СППВПОН MongoDB, що використовуються для формування даних в колекції «Q-таблиця».

3.2.5. Формування винагороди за дію

Визначення величини винагороди є ключовим аспектом у навчанні агента, оскільки саме вона формує зворотний зв’язок, який система отримує після кожного рішення щодо кешування. Винагорода повинна стимулювати збереження тих об’єктів, які згодом будуть повторно використані, і в той же час відображати негативні наслідки неефективного використання кешу (табл. 3.4.).

Таблиця 3.4.

Класифікація винагороди для моделі на основі результату кешування

№	Ситуація	Винагорода $r^{(t)}$	Пояснення
1	Об’єкт був у кеші та повторно запитувався	+1	Кешування було ефективним, об’єкт використано повторно
2	Об’єкт був у кеші, але не був повторно запитаний	-0.5	Кешування було марним: об’єкт зайняв місце, але не був використаний
3	Об’єкт не був у кеші, і запиту до нього не було	0	Нейтральна ситуація: нічого не втрачається і не виграється
4	Об’єкт не був у кеші, але пізніше був запитаний	-1	Втрата потенційного кеш-хіту: об’єкт міг бути корисним, але його не додали в кеш

Після кожної дії виконується аналіз подальшої активності щодо об’єкта протягом заданого періоду часу і на основі цього визначається, наскільки обґрунтованим було рішення. Така стратегія дозволяє моделі навчатися уникати неефективного використання кешу та фокусуватись на об’єктах з високою ймовірністю повторного звернення.

3.2.6. Вибір дії за ε-жадібною стратегією

Для того, щоб агент моделі табличного Q-Learning не обмежувався лише вже відомими діями з високим значенням Q-функції якості, у процесі навчання застосовується ε-жадібна стратегія вибору дій [88]. Вона дозволяє поєднати два типи поведінки:

1. Дослідження (*exploration*) – випадкове виконання дії, щоб вивчити середовище;

2. Використання (*exploitation*) – вибір дії з максимальним $Q(s,a)$ у даному стані.

На кожному кроці, перебуваючи в стані $s^{(t)}$, система діє за наступним правилом [88]:

– з імовірністю ε_t обирається випадкова дія;

– з імовірністю $1-\varepsilon_t$ обирається дія, яка максимізує функцію якості, тобто: $a^{(t)} = \arg\max_a Q\left(s^{(t)}, a\right)$

На початку навчання ε_t встановлюється високим (наприклад, 0.9), що дозволяє досліджувати можливі стани. Згодом параметр зменшується за експоненціальною формулою:

$$\varepsilon_t = \max(\varepsilon_{\text{мін}}, \varepsilon_0 \cdot e^{-kt}), \tag{3.3}$$

де:

– ε_0 – початкове значення;

– $\varepsilon_{\text{мін}}$ – мінімальне допустиме значення;

– k – коефіцієнт темпу зменшення;

– t – номер ітерації алгоритму.

Застосування такої стратегії є важливою частиною процесу навчання, адже вона запобігає монотонності у виборі дій: навіть якщо на початку певна дія здається найкращою, без періодичного дослідження система може не знайти ще кращих варіантів. Зменшення параметра ε_t з часом дає змогу поступово переходити від дослідження до стабільного використання знань, що особливо актуально в системі підтримки процесу відновлення, де динаміка запитів можуть змінюватися швидко.

3.2.7. Збереження дій для подальшої оцінки ефективності кешування

В процесі вирішення цієї задачі кешування результат дії не проявляється відразу. Повторний запит до об’єкта може з’явитися лише через певний час, тому

має зберігатися інформація про попередні дії для подальшого аналізу. Для цього використовується спеціальний список переходів – черга, у яку на момент прийняття рішення, тобто у час t, зберігається кортеж:

$$\left(s^{(t)}, a^{(t)}, \text{object\_id}, t\right),$$

де:

– $s^{(t)}$ – стан об’єкта у вигляді вектора ознак;

– $a^{(t)}$ – прийняте рішення (0 або 1);

– object_id – ідентифікатор об’єкта, необхідний для подальшого виявлення повторного запиту до об’єкта;

– t – мітка часу дії.

Після проходження фіксованого інтервалу T, у момент $t + T$, відбувається звернення до цього запису, щоб визначити, чи був об’єкт повторно запитаний. На основі цього присвоюється відповідна винагороду r_t і виконується оновлення Q-значення згідно з формулою Беллмана (формула 3.1).

Такий механізм дозволяє агенту коригувати свою поведінку не лише на основі миттєвих результатів, а й з урахуванням реальної ефективності рішень у майбутньому.

3.2.8. Алгоритм вирішення

Основні кроки алгоритму вирішення задачі про оцінку доцільності додавання об’єкта до кешу показані на рис. 3.2.

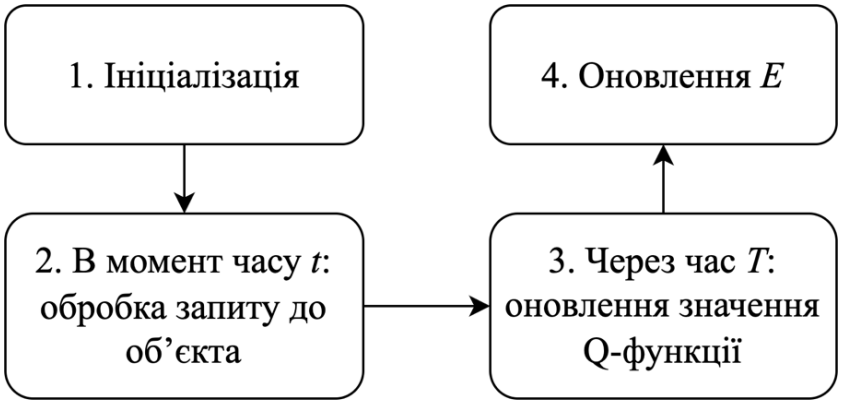

Рисунок 3.2. Основні кроки алгоритму розв’язання задачі

Детальний алгоритм вирішення задачі виглядає наступним чином:

1. Ініціалізація:

– Створити порожню Q-таблицю у MongoDB;

– Встановити параметри:

1. α – коефіцієнт навчання;
2. γ – коефіцієнт дисконтування;
3. $\varepsilon_0, \varepsilon_{\text{мін}}, k$ – параметри ε-жадібної політики;
4. T – час очікування для оцінки результату дії кешування.

2. В момент часу t: обробка запиту до об’єкта:

– Сформувати вектор ознак об’єкта $x^{(t)}$, тобто стан $s^{(t)}$;

– Обрати дію $a^{(t)}$ за ε-жадібною політикою;

– Виконати дію (наприклад, додавати об’єкт у кеш);

– Зберегти запис у список переходів:

$$\left(s^{(t)}, a^{(t)}, \text{object\_id}, t\right)$$

3. Через час T: оновлення значення Q-функції:

– Витягти з списку переходів запис, мітка часу якого дорівнює $t = t_{\text{поточний}} - T$;

– За object_id сформувати новий стан $s^{(t+T)}$, отримавши новий вектор ознак об’єкта;

– Визначити винагороду $r^{(t)}$ на основі використання об’єкта після виконаної дії;

– Оновити значення функції якості відповідно до класичної формули Беллмана [87]:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot \left[r_t + \gamma \cdot \max_a Q\,(s_{t+T}, a) - Q(s_t, a_t)\right], \qquad (3.4)$$

де:

– $Q(s_t, a_t)$ – поточне значення функції якості для пари «стан–дія» на момент t;

– $\alpha \in (0,1]$ – коефіцієнт навчання, який визначає вагу нової інформації при оновленні;

– r_t – винагорода, отримана за дію a_t у стані s_t;

– $\gamma \in [0,1]$ – коефіцієнт дисконтування, що визначає важливість майбутніх винагород;

– $\max_a Q\,(s_{t+T}, a)$ – максимальне значення функції якості серед усіх можливих дій a у новому стані s_{t+T}, тобто через час T після виконання дії;

– s_{t+T} – новий стан, сформований через фіксований проміжок часу T після виконання дії у s_t;

– a_t – дія, яка була виконана у стані s_t;

– a – довільна допустима дія, яка може бути виконана у стані s_{t+T}.

4. Оновлення ε:

Після кожного кроку значення ε зменшується згідно з формулою 3.3 [88].

3.2.9. Підсумок роботи

Побудований алгоритм вирішення задачі оцінки доцільності додавання об’єкта до кешу дозволяє СППВПОН навчитися приймати ефективні рішення щодо ефективності кешування певного об’єкта на основі накопиченого досвіду. Завдяки

використанню відкладеного оновлення функції якості, модель враховує реальні наслідки дій із часовою затримкою, адаптуючись до поведінки користувачів.

Після завершення фази навчання значення функції якості, збережені у колекції «Q-таблиця» використовується як політика, що показана на формулі 3.2. та дозволяє у реальному часі приймати рішення без додаткових обчислень. Збережена інформація про типові ситуації дозволяє моделі узагальнювати досвід на подібні об’єкти, що зменшує потребу в повторному обчисленні для кожного нового випадку. У разі змін у динаміці запитів або шаблонах поведінки користувачів можлива повторна фаза навчання моделі.

3.3. Вибір об’єкта для видалення з заповненого кешу

3.3.1. Формалізація

Дана задача виникає в ситуації, коли кеш уже заповнений і потрібно звільнити місце для нових об’єктів. Тут важливо обрати об’єкт для видалення таким чином, щоб мінімізувати втрати потрібної інформації. Це вимагає аналізу актуальності кожного об’єкта в кеші, врахування його ролі, патернів поведінки користувачів і очікуваного використання в майбутньому. Актуальність задачі обумовлена такими основними чинниками:

1. Обмеженість кешу. Кеш має фіксований розмір, тому при спробі зберегти новий об’єкт у повному кеші необхідно обрати один із наявних об’єктів для видалення. Невдалий вибір може призвести до видалення корисного об’єкта, що буде повторно потрібен у найближчому майбутньому, що знижує загальну ефективність кешу.

2. Невизначеність щодо майбутніх запитів. У момент видалення неможливо точно знати, які саме об’єкти будуть запитані згодом. Два об’єкти можуть виглядати однаково з точки зору історії використання, але лише один з них насправді буде потрібний у майбутньому. Прості евристики не здатні враховувати такі нюанси.

3. Вартість помилкового рішення. Видалення об’єкта, який згодом буде повторно запитаний, призводить до ситуації, коли дані доводиться повторно завантажувати з БД, бо вони відсутні в кеші. Це створює додаткове навантаження на СППВПОН та збільшує час відповіді користувачу.

4. Необхідність адаптації до змін у поведінці користувачів. Статистичні характеристики об’єктів змінюються з часом: з’являються нові шаблони використання, зміни в популярності тощо. Це робить неефективними підходи, що спираються лише на фіксовані правила або прості метрики. Натомість потрібна модель, яка здатна адаптуватися та прогнозувати на основі актуальних даних.

Задача визначення об’єкта, який слід видалити з кешу у разі його заповнення, є прикладом бінарної класифікації [89]. Для кожного об’єкта, що перебуває в кеші, система має оцінити ймовірність того, що об’єкт буде повторно використаний у майбутньому. Виходячи з цього прогнозу, обирається об’єкт, який з найменшою ймовірністю буде запитано і саме його слід видалити.

Формально, для кожного об’єкта обчислюється цільова змінна:

$$y \in \{0, 1\},$$

де:

– $y = 1$ – об’єкт буде запитано у найближчому майбутньому;

– $y = 0$ – об’єкт не буде запитано, і його можна видалити.

Кожен об’єкт описується вектором ознак:

$$x = [x_1, x_2, \dots, x_n],$$

де x_i – числове або категоріальне значення, що відображає історію запитів, тип об’єкта, профіль користувача тощо.

3.3.2. Обґрунтування вибору методу вирішення

Для ефективного розв’язання задачі вибору об’єкта для видалення з кешу проаналізовано кілька підходів, які відрізняються за рівнем складності, адаптивності, здатністю працювати з ознаками об’єктів та враховувати динаміку

змін у запитах. У цій задачі важливо не просто звільнити місце в кеші, а зробити це з мінімальними втратами для подальшої продуктивності системи. Це передбачає оцінку ймовірності повторного використання кожного з об’єктів у кеші на основі їхніх характеристик. Нижче представлено порівняння основних методів, які можуть бути використані для прийняття рішення про видалення об’єкта.

1. Евристичні методи. Це прості правила, що не використовують навчання на даних. Наприклад:

– LRU – видаляється об’єкт, що не використовувався найдовше [82];
– LFU – видаляється об’єкт з найменшою кількістю запитів [83];
– Random – рішення про кешування або витіснення об’єкта робляться випадковим чином [75].

Такі методи мають низьку складність реалізації, але не враховують повний контекст і не адаптуються до змін.

2. Класичне машинне навчання. У цьому підході задача подається як бінарна класифікація: для кожного об’єкта в кеші формується вектор ознак, і модель прогнозує ймовірність майбутнього запиту. Найчастіше використовуються:

– Логістична регресія – проста, інтерпретована модель з лінійним ядром [90];
– Дерева рішень / Random Forest – дозволяють враховувати взаємодії між ознаками [91];
– Градієнтний бустинг (наприклад, XGBoost) – потужна модель із високою точністю, але складніша в налаштуванні [92].

Перевага цього підходу – це швидка побудова моделей, можливість оновлення на основі нових прикладів, і хороша адаптивність до змінних шаблонів запитів.

Порівняння можливих методів вирішення цієї задачі наведено в табл. 3.5.

Методи класичного машинного навчання, зокрема логістична регресія, демонструють помітну перевагу над класичними евристиками за ключовими критеріями: здатністю враховувати ознаки об’єкта, прогнозувати ймовірність повторного використання, адаптуватися до нових патернів використання та масштабуватися до великих обсягів даних. Враховуючи ці переваги для вирішення

цієї задачі використовується метод класичного машинного навчання, а саме логістична регресія.

Таблиця 3.5.

Порівняльна таблиця можливих методів вирішення задачі

Критерій	Евристичні методи	Класичне машинне навчання
Урахування ознак об’єкта	Немає	Є
Прогноз ймовірності повторного запиту	Немає	Є
Адаптивність до змін	Немає	Є
Простота реалізації	Висока	Середня
Масштабованість на велику кількість об’єктів	Обмежена	Висока

Логістична регресія – це статистичний метод класифікації, який використовується для прогнозування ймовірності належності об’єкта до певного класу [90]. Вона моделює залежність між входом (вектором ознак) і бінарною цільовою змінною, використовуючи сигмоїдальну функцію (рис. 3.3.). На відміну від лінійної регресії, логістична обмежує результат у межах [0; 1], що дозволяє інтерпретувати його як ймовірність [93].

Ключова формула, яка обчислює ймовірність, що об’єкт буде ще запитаний має вигляд [90]:

$$\hat{y} = \frac{1}{1+e^{-\left(\sum_{j=1}^{n} w_j \cdot x_j + b\right)}}, \tag{3.5}$$

де:

– $\hat{y} \in [0,1]$ – прогнозована ймовірність повторного запиту;
– $x_j \in R$ – j-та ознака об’єкта (наприклад, частота запитів);
– $w_j \in R$ – вага ознаки x_j, що відображає її значущість;

– $b \in R$ – зміщення (*bias*), що зсуває поріг класифікації;

– $\sum w_j x_j$ – лінійна комбінація ознак.

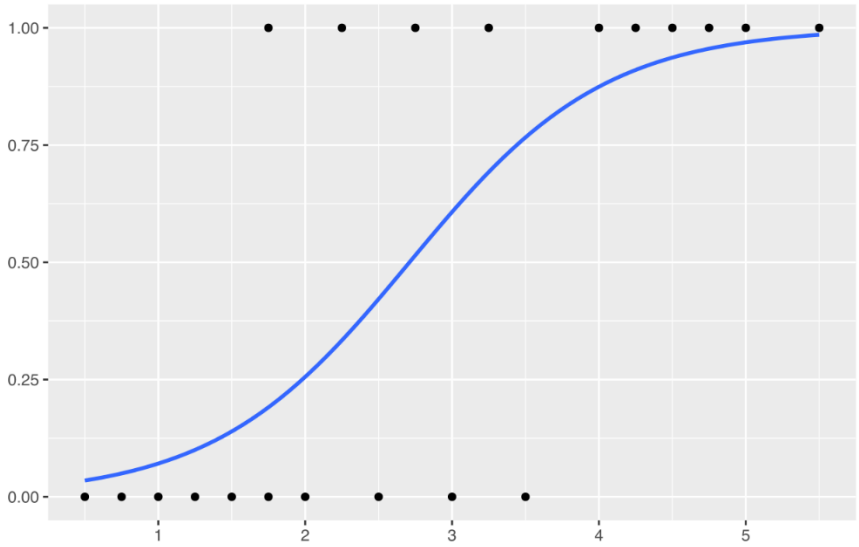

Рисунок 3.3. Графік логістичної регресії [90]

Таким чином, кожна ознака впливає на підсумкову оцінку згідно з тим, як вона вивчена моделлю на історичних прикладах. Такий підхід дозволяє обчислити ймовірність повторного запиту на основі вектора ознак об’єкта. Модель працює однаково для всіх об’єктів, що дозволяє масштабувати рішення, підтримуючи при цьому адаптивність.

3.3.3. Представлення стану об’єкта у вигляді вектора ознак

Для кожного об’єкта, що перебуває в кеші, на момент прийняття рішення про видалення формується вектор ознак – набір характеристик, які описують об’єкт або його поведінку в СППВПОН. Саме вони використовуються для побудови прогнозу в моделі логістичної регресії (табл. 3.6.).

Стан об’єкта позначається як:

$x = [x_1, x_2, x_3, x_4, x_5, x_6\ ]$,

де кожен x_i – це числове або категоріальне значення, що відображає важливу характеристику об’єкта чи контекст його використання.

Таблиця 3.6.

Ознаки, що використовуються для представлення стану об’єкта в моделі

Позначення	Назва ознаки	Опис
x_1	Кількість унікальних користувачів	Скільки різних користувачів зверталися до об’єкта протягом доби
x_2	Загальна кількість запитів	Загальна кількість звернень до об’єкта за останні 24 години
x_3	Тип об’єкта	Наприклад, 0 – житловий, 1 – школа, 2 – лікарня тощо
x_4	Кількість годин з моменту останнього запиту	Відображає давність взаємодії з об’єктом
x_5	Пріоритетність об’єкта або зони	Наприклад, 0 – низький, 1 – середній, 2 – високий
x_6	Локація об’єкта	Категоріальна ознака району / регіону, або його кодоване значення
x_7	Роль користувача	Категоріальна ознака, що відображає тип користувача, який зробив запит до об’єкта
x_8	Час доби	Вказує на період доби, коли відбувся останній запит до об’єкта

Цей набір ознак є компактним, але достатньо інформативним для прийняття рішень про доцільність подальшого зберігання об’єкта в кеші. Вектор ознак використовується на усіх етапах: формування списку прикладів, побудова навчальних прикладів, прогнозування та оновлення моделі.

3.3.4. Роль та структура додаткових колекцій

Для забезпечення функціонування моделі та її навчання в СППВПОН використовуються дві окремі колекції: «Навчальні приклади» та «Параметри моделі».

Колекція «Навчальні приклади» зберігає дані, що формуються після спостереження за об’єктами протягом періоду T після їх видалення з кешу. Вона слугує джерелом для навчання логістичної регресії. Кожен запис відповідає одному прикладу навчання – об’єкту, який був видалений, та інформації про те, чи був він запитаний повторно.

Таблиця 3.7.

Структура документа в колекції «Навчальні приклади»

Поле	Тип даних	Опис
x	масив чисел	Вектор ознак об’єкта на момент видалення
y	число (0 або 1)	Мітка: чи був об’єкт запитаний повторно у період $[t, t+T]$

Ці дані накопичуються асинхронно у фоновому режимі, що дозволяє не переривати основний процес обробки запитів і поступово оновлювати модель, зберігаючи її актуальність.

Колекція «Параметри моделі» містить актуальні значення параметрів моделі, які використовуються для передбачення повторного запиту кожного об’єкта в кеші. Оскільки модель однакова для всіх об’єктів, параметри зберігаються в одному екземплярі, що дозволяє узагальнити знання між різними об’єктами СППВПОН.

Таблиця 3.8.

Структура документа в колекції «Параметри моделі»

Поле	Тип даних	Опис
w	масив чисел	Массив ваг: $[w_1, w_2, \dots, w_n]$
b	число	Значення зміщення

Дана колекція містить масив ваг, кожен з яких відповідає певній ознаці об’єкта, а також значення зміщення, що зсуває поріг прийняття рішення. Під час прогнозування для кожного об’єкта обчислюється сума ознак, до якої додається зміщення, після чого результат передається до функції логістичної регресії.

Параметри зберігаються у зручному вигляді для подальшого використання в онлайн-режимі. Вони оновлюються після кожного циклу навчання на даних з колекції «Навчальні приклади».

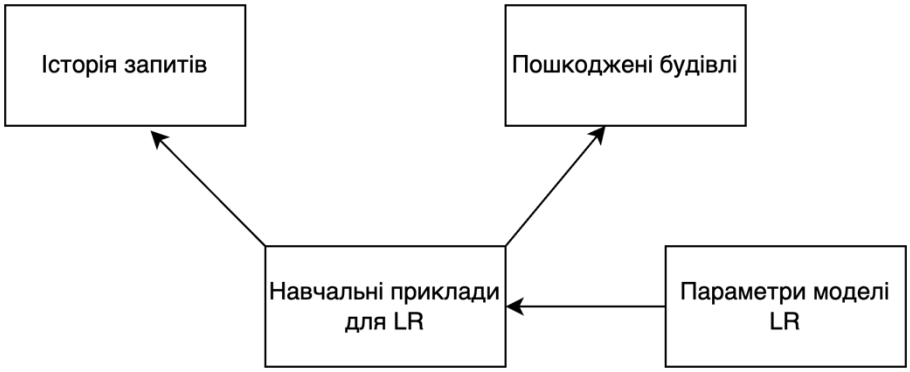

Рисунок 3.4. Структура взаємозв’язків між основними колекціями документів в базі даних СППВПОН MongoDB для формування даних в колекціях «Навчальні приклади» та «Параметри моделі».

Колекція «Навчальні приклади» формується на основі ознак, отриманих із колекцій «Історія запитів» та «Пошкоджені будівлі» (рис. 3.4.). Вона використовується для побудови тренувальної вибірки, на якій навчається модель. Колекція «Параметри моделі» зберігає актуальні коефіцієнти моделі логістичної регресії, отримані в результаті навчання.

3.3.5. Список прикладів для подальшого формування мітки

Цей компонент є критичним для того, щоб навчання моделі відповідало реальній поведінці користувачів. Мітка y для кожного прикладу не є відомою одразу після видалення об'єкта з кешу, її можна визначити лише після проходження певного фіксованого інтервалу часу T, протягом якого спостерігається чи був повторний запит до об'єкта.

Для цього використовується список прикладів, який зберігає необхідні дані для подальшого формування мітки *y,* необхідної для навчання логістичної регресії і знаходиться в оперативній пам'яті, оскільки дані потрібні лише тимчасово та мають бути доступні для швидкої обробки у фоновому режимі.

Таблиця 3.9.

Структура списку прикладів

Поле	Опис
$x^{(t)}$	Вектор ознак об'єкта на момент видалення з кешу
object_id	Ідентифікатор об'єкта, потрібний для перевірки повторного запиту
t	Мітка часу, коли відбулося видалення об'єкта з кешу

Послідовність дій:

1. У момент видалення об'єкта з кешу (час t) формується вектор ознак $x^{(t)}$, що описує об'єкт у момент дії. Разом з ідентифікатором об'єкта та міткою часу t запис додається до списку.

2. Через інтервал часу T, виконується перевірка чи був запит до object_id у проміжку $[t, t+T]$:

– якщо так – встановлюється мітка $y = 1$;

– інакше – $y = 0$.

3. Після цього з списку формується повноцінний навчальний приклад $(x^{(t)}, y)$, який додається до колекції «Навчальні приклади».

3.3.6. Алгоритм навчання моделі

Оновлення ваг моделі є ключовим процесом, що дозволяє адаптуватись до змін у запитах користувачів. На основі даних в колекції «Навчальні приклади», модель логістичної регресії оновлює свої параметри, поступово покращуючи точність прогнозування. Це дозволяє з часом краще відрізняти об'єкти, які ймовірно будуть запитані знову, від тих, що не матимуть подальшого використання.

Процес навчання реалізується у кілька послідовних кроків, а саме:

Крок 1. Обчислення передбачення для кожного прикладу. На цьому етапі використовуються поточні значення ваг і зміщення, щоб обчислити прогнозовану ймовірність повторного запиту для кожного прикладу. Це потрібно, щоб порівняти передбачення моделі з реальними мітками y і побудувати градієнти [90], [93].

$$\widehat{y^{(i)}} = \frac{1}{1+e^{-\left(\sum_{j=1}^{n} w_j \cdot x_j^{(i)} + b\right)}}, \tag{3.6}$$

де:

– $x^{(i)} = \left[x_1^{(i)}, \dots, x_n^{(i)}\right]$ – вектор ознак для і-го прикладу;

– $w_j \in R$ – поточне значення ваги j-тої ознаки;

– $b \in R$ – поточне значення зміщення;

– $\widehat{y^{(i)}} \in [0,1]$ – прогнозована ймовірність повторного запиту.

Крок 2. Обчислення логарифмічної функції втрат

Цей крок дозволяє оцінити, наскільки поточна модель добре виконує своє завдання на основі накопичених прикладів. Логарифмічна функція втрат є стандартною для задач бінарної класифікації. Вона визначає, наскільки сильно

передбачення моделі $\widehat{y^{(i)}}$ відхиляються від правильних міток $y^{(i)}$, і є критерієм, який модель намагається мінімізувати [90], [93].

$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \cdot \log\left(\widehat{y^{(i)}}\right) + \left(1 - y^{(i)}\right) \cdot \log\left(1 - \widehat{y^{(i)}}\right)\right], \quad (3.7)$$

де:

– m – кількість нових прикладів у колекції «Навчальні приклади»;

– $y^{(i)} \in \{0,1\}$ – правильна мітка для і-го прикладу;

– $\widehat{y^{(i)}} \in [0,1]$ – прогнозована ймовірність;

– log – натуральний логарифм.

Хоча сама функція втрат є числовою оцінкою якості моделі, для оновлення параметрів використовуються її похідні, а саме градієнти.

Крок 3. Обчислення градієнтів

Градієнти дозволяють моделі зрозуміти, в якому напрямку і наскільки потрібно змінити кожен параметр, щоб зменшити загальну похибку. Вони показують, яка саме вага зробила найбільший внесок у помилку. Якщо модель сильно помилилась і певна ознака мала велике значення, то її вага повинна змінитися сильніше.

Градієнт кожної ваги – це похідна логарифмічної функції втрат по цьому параметру, яка визначає коректний напрямок оновлення [94]:

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}\left(\widehat{y^{(i)}} - y^{(i)}\right) \cdot x_j^{(i)}, \quad (3.8)$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{m}\sum_{i=1}^{m}\left(\widehat{y^{(i)}} - y^{(i)}\right), \quad (3.9)$$

де:

– $\widehat{y^{(i)}} \in [0,1]$ – передбачення моделі для і-го прикладу;

– $y^{(i)} \in \{0,1\}$ – правильна відповідь;

– $x_j^{(i)} \in R$ – значення j-тої ознаки у і-му прикладі;

– m – кількість прикладів у навчальному наборі;

– $w_j \in R$ – поточне значення ваги j-тої ознаки;

– $b \in R$ – поточне значення зміщення.

Таким чином, кожна вага w_j змінюється тим сильніше, чим більшою була помилка $(\hat{y} - y)$ і чим вагомішою була відповідна ознака x_j. Градієнти фактично вказують, які параметри слід коригувати і в якому напрямку [90], [93].

Крок 4. Оновлення параметрів моделі

Цей завершальний крок дозволяє моделі вчитися шляхом зміни своїх параметрів в напрямку зменшення похибки. Оновлення відбувається за методом градієнтного спуску: кожна вага w_j та зміщення b оновлюються з урахуванням відповідного градієнта та швидкості навчання η [94], [95]:

$$w_j = w_j - \eta \cdot \frac{\partial \mathcal{L}}{\partial w_j}, \tag{3.10}$$

$$b = b - \eta \cdot \frac{\partial \mathcal{L}}{\partial b}, \tag{3.11}$$

де:

– $\eta \in (0,1]$ – коефіцієнт навчання, який визначає величину кроку оновлення;

– інші позначення – як у попередньому кроці.

Система зменшу кожну вагу в напрямку, який вказує градієнт, і на величину, визначену коефіцієнтом навчання η. Це гарантує, що модель поступово вчиться до таких значень параметрів, які мінімізують похибку. Цей механізм забезпечує поступове покращення моделі на основі досвіду: чим більше прикладів, тим краще модель розуміє, які ознаки справді впливають на повторне використання об’єкту.

Таким чином, модель логістичної регресії здатна ефективно оцінювати ймовірність повторного використання кожного об’єкта на основі його вектора

ознак. Завдяки списку прикладів і регулярному оновленню параметрів моделі, відбувається поступова адаптація до нових варіантів запитів і приймаються більш обґрунтовані рішення щодо видалення об’єктів з кешу.

3.3.7. Алгоритм вирішення

Ключові кроки алгоритму вирішення задачі про вибір об’єкта для видалення з заповненого кешу показані на рис. 3.5.

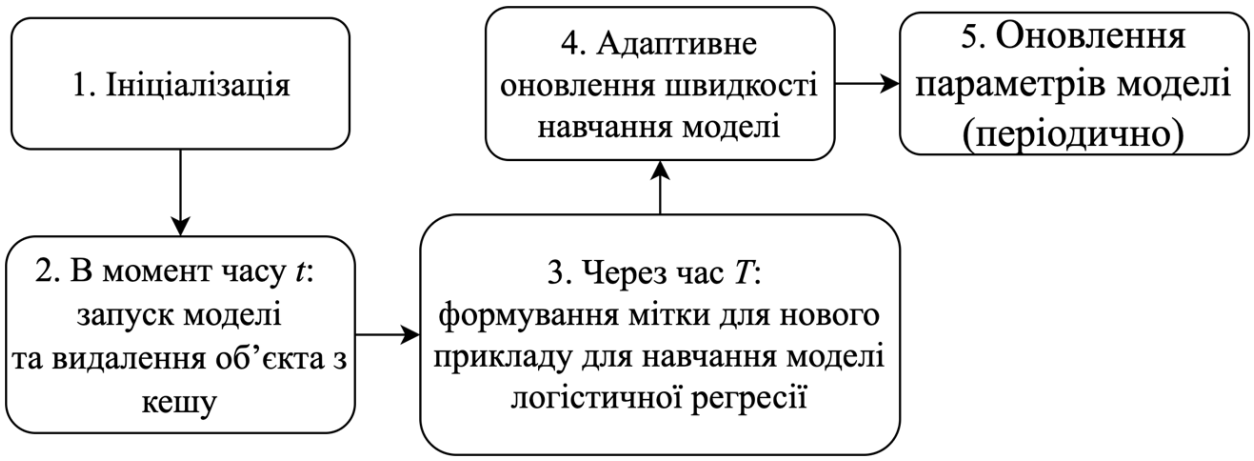

Рисунок 3.5. Основні кроки алгоритму вирішення задачі

Детальний алгоритм вирішення виглядає наступним чином:

1. Ініціалізація
   - Створити:
     1. Список прикладів у оперативній пам’яті;
     2. Колекцію «Навчальні приклади» та «Параметри моделі» у базі даних MongoDB.
   - Встановити параметри:
     1. Початкові значення ваг w_j та зміщення b;
     2. Коефіцієнт навчання η;
     3. Час очікування T для формування мітки y.
2. В момент часу t: запуск моделі та видалення об’єкта з кешу:

– Модель табличного Q-Learning визначила, що об’єкт потрібно додати в кеш, проте він заповнений;

– Для кожного об’єкта в кеші сформувати вектор ознак

$$x^{(t)} = \left[x_1^{(t)}, x_2^{(t)}, \dots, x_n^{(t)}\right];$$

– Запустити модель логістичної регресії для оцінки ймовірності повторного використання кожного об’єкта;

– Об’єкт з найнижчою ймовірністю повторного використання видаляється з кешу;

– Вектор ознак видаленого об’єкта, його object_id та час t зберігаються у список прикладів у вигляді кортежу:

$$\left(x^{(t)}, \text{object\_id}, t\right)$$

3. Через час T: формування мітки для нового прикладу для навчання моделі логістичної регресії:

– Перевірити список прикладів і обробити ті, для яких з моменту додавання до списку минув інтервал T;

– Для кожного об’єкта визначити, чи був він запитаний у проміжку $[t, t+T]$ та виставити відповідну мітку y;

– Додати приклад $\left(x^{(t)}, y\right)$ до колекції «Навчальні приклади»;

– Видалити оброблений запис із списку.

4. Адаптивне оновлення швидкості навчання

Для поступового зменшення швидкості навчання використовується формула:

$$\eta_t = \max(\eta_{\text{мін}}, \eta_0 \cdot e^{-kt}), \tag{3.12}$$

де:

– η_0 – початкове значення коефіцієнта навчання;

– $\eta_{\text{мін}}$ – мінімальне допустиме значення;

– k – коефіцієнт темпу зменшення;

– t – номер ітерації.

5. Оновлення параметрів моделі (періодично)

На основі накопичених прикладів відбувається оновлення параметрів моделі.

3.3.8. Підсумок роботи

Використання вищеописаного алгоритму із застосуванням моделі логістичної регресії дозволяє автоматично адаптуватися до змін у поведінці користувачів і приймати обґрунтовані рішення про те, який об’єкт слід видалити з кешу. Список прикладів забезпечує ефективне визначення мітки y, а модель логістичної регресії поступово навчається на реальних прикладах.

Після фази навчання оновлена модель зберігається у вигляді ваг і зміщення, які використовуються для реального прогнозування. Можливість повторного навчання дозволяє підтримувати її актуальність в умовах динамічного навантаження та патернів доступу.

3.4. Періодична інвалідація неактуальних об’єктів у кеші

3.4.1. Формалізація

Ця задача полягає в періодичному аналізі об’єктів, що вже зберігаються в кеші для того, щоб визначити доцільность їх подальшого зберігання. На відміну від задачі вибору об’єкта для видалення, яка активується лише в момент заповнення кешу, ця задача виконується регулярно, незалежно від поточного рівня заповнення. Її ціль полягає в тому, щоб виявляти об’єкти до яких, ймовірно, більше не буде запитів від користувачів СППВПОН, і завчасно видаляти їх, щоб звільнити кеш для даних з вищою потенційною цінністю. Такий підхід дозволяє підтримувати актуальність кешу та забезпечує більш ефективне використання його обмеженого розміру.

Задача формалізується як бінарна класифікація [89]. Для кожного об’єкта в кеші обчислюється цільова змінна:

$$y \in \{0, 1\},$$

де:

– $y = 1$ – об’єкт буде запитано у найближчому майбутньому;

– $y = 0$ – об’єкт не буде запитано, і його можна видалити.

Кожен об’єкт описується вектором ознак:

$$x = [x_1, x_2, \dots, x_n],$$

де x_i – числове або категоріальне значення, що відображає історію запитів, тип об’єкта, профіль користувача тощо.

3.4.2. Обґрунтування вибору методу вирішення

Задача визначення доцільності подальшого зберігання об’єкта в кеші має дуже схожу логіку та формалізацію із задачею вибору об’єкта для видалення, коли кеш заповнений. В обох задачах система потрібно оцінити чи буде об’єкт повторно використаний у найближчому майбутньому.

З огляду на схожість між двома задачами, порівняння можливих методів вирішення цієї задачі наведено в табл. 3.5. Методи класичного машинного навчання, зокрема логістична регресія, демонструють помітну перевагу над класичними евристиками за ключовими критеріями: здатністю враховувати ознаки об’єкта, прогнозувати ймовірність повторного використання, адаптуватися до нових патернів використання та масштабуватися до великих обсягів даних. Враховуючи ці переваги для вирішення цієї задачі використовується метод класичного машинного навчання, а саме логістична регресія [90], [93].

3.4.3. Представлення стану об’єкта у вигляді вектора ознак

Оскільки дана задача має схожу формалізацію до задачі вибору об’єкта для видалення при заповненому кеші для її вирішення доцільно використовувати той

самий вектор ознак. Він поєднує як загальні характеристики поведінки об’єкта (наприклад, частота використання), так і ознаки, специфічні для СППВПОН (напр. локація). Повний перелік ознак наведено у табл. 3.6.

3.4.4. Роль та структура додаткових колекцій

У межах цієї задачі не передбачається створення окремих колекцій документів. Для задачі інвалідації кешу використовується та сама модель логістичної регресії, що була навчена для задачі вибору об’єкта на видалення у разі заповнення кешу. Це можливо завдяки спільній формалізації задач як бінарної класифікації з однаковим набором ознак. Таким чином, система може виконувати інвалідацію об’єктів на основі існуючої моделі без потреби у дублюванні або зміні структури БД.

3.4.5. Перевірка актуальності об’єктів в кеші через фіксований інтервал часу

На відміну від задачі вибору об’єкта для видалення, що з’являється лише, коли кеш заповнений, інвалідація або видалення застарілих об’єктів у цій задачі відбувається проактивно – через фіксований інтервал часу T. Такий підхід дозволяє СППВПОН самостійно контролювати актуальність вмісту кешу, не чекаючи нових запитів на додавання об’єктів.

Основна ідея полягає в тому, що через кожен інтервал часу T гібридний метод інтелектуального управління кешем проходить усі наявні об’єкти в кеші, оцінює ймовірність їх повторного використання за допомогою логістичної регресії, і видаляє ті, чия ймовірність не перевищує заздалегідь визначений поріг. Це дозволяє поступово звільняти кеш від застарілої інформації, навіть якщо нових об’єктів до нього не додається.

Таким чином, механізм інвалідації доповнює інші задачі гібридного методу: він забезпечує автономне оновлення кешу в умовах, коли поведінка користувачів змінюється, а частина даних втрачає актуальність із часом.

3.4.6. Алгоритм вирішення

Детальний алгоритм вирішення задачі виглядає наступним чином:

1. Ініціалізація:
   - Завантажити модель логістичної регресії;
   - Встановити порогове значення θ для інвалідації об'єктів;
   - Встановити фіксований інтервал перевірки актуальності T.
2. Кожні T одиниць часу для кожного об'єкта в кеші:
   1. Сформувати вектор ознак $x = [x_1, x_2, \dots, x_n]$;
   2. Обчислити ймовірність повторного використання:

$$\hat{y} = \sigma\left(\sum_{j=1}^{n} w_j \cdot x_j + b\right), \tag{3.13}$$

де:

– $\hat{y} \in [0,1]$ – передбачена ймовірність того, що об'єкт буде повторно використаний;

– $\sigma(z) = \frac{1}{1+e^{-z}}$ – функція активації;

– $z = \sum_{j=1}^{n} w_j \cdot x_j + b$ – лінійна сума ознак зі зміщенням;

– x_j – значення j-тої ознаки об'єкта;

– w_j – вага j-тої ознаки, що показує її важливість для прогнозу;

– b – зміщення, що впливає на рішення моделі незалежно від значень ознак;

– n – кількість ознак.

3. Якщо $\hat{y} < \theta$, то об'єкт вважається неактуальним і видаляється з кешу.

3.4.7. Підсумок роботи

Запропонований алгоритм вирішення задачі дозволяє СППВПОН визначати доцільність подальшого зберігання об’єктів у кеші. Регулярна перевірка актуальності забезпечує своєчасну інвалідацію непотрібних даних, що дозволяє звільняти місце під об’єкти з вищим пріоритетом.

Модель логістичної регресії використовує ті самі ознаки та параметри, що й у задачі вибору об’єкта для видалення, коли кеш заповнений тому не потребує окремого навчального процесу. Це забезпечує ефективну інтеграцію в загальний цикл роботи СППВПОН без додаткових витрат ресурсів.

3.5. Обґрунтування вибору технології Redis

Сучасна технологія кешування повинна відповідати ряду вимог для ефективної роботи в системах із високим навантаженням. Серед основних вимог: підтримка низької затримки доступу до даних, здатність працювати в розподілених середовищах, масштабованість для обробки великих обсягів даних і гнучке налаштування існуючих методів кешування. Крім того, вона має забезпечувати можливості для реплікації даних і стійкості до збоїв, що є критично важливими для стабільної роботи таких інформаційних високонавантажених систем, як СППВПОН.

Кеш працює за принципом збереження даних у вигляді пар “ключ-значення” [70], що дозволяє швидко отримувати потрібну інформацію за унікальним ключем. Архітектурно кеш інтегрується як окремий компонент системи, що взаємодіє з БД і додатком. Завдяки цьому підходу кеш дозволяє знизити навантаження на основну сховище даних і пришвидшити обробку запитів, особливо для даних, які часто використовуються або є критично важливими для роботи системи.

Технології кешування класифікуються за їхньою архітектурою та способом обробки даних. Основними типами є:

1. Технології кешування в оперативній пам’яті (*In-Memory Cache Technologies*). Цей тип технологій зберігає дані виключно в оперативній пам’яті, що

забезпечує високу швидкість доступу. Такі технології використовуються для сценаріїв, де затримки повинні бути мінімальними, а обсяги даних помірними, щоб відповідати доступній пам’яті. Вони ідеально підходять для швидкого читання та обробки невеликих обсягів даних. Приклади: Redis, Memcached [96].

2. Розподілені кеш-технології (*Distributed Cache Technologies*). Ці технології розподіляють дані між кількома вузлами або серверами, що дозволяє масштабувати обсяг кешу та забезпечувати стійкість до збоїв. Розподілені кеші забезпечують високу доступність даних і використовуються у великих системах із розподіленими навантаженнями. Вони підтримують реплікацію, кластеризацію та інші механізми забезпечення узгодженості. Приклади: Hazelcast, Redis Cluster [97].

Redis – це популярна технологія кешування з відкритим кодом, яка працює в оперативній пам’яті і забезпечує високу швидкість та масштабованість. Завдяки підтримці різноманітних структур даних, таких як рядки, списки, множини, упорядковані множини, хеші та навіть геопросторові дані, Redis є універсальним інструментом для реалізації кешування, черг повідомлень та інших задач [98].

Redis пропонує розширені можливості для забезпечення стійкості та високої доступності. Це досягається завдяки функціям реплікації, кластеризації та механізмам автоматичного відновлення після збоїв. Реплікація дозволяє створювати копії даних на різних серверах для захисту від втрат, а кластеризація забезпечує розподіл даних між кількома вузлами для масштабування системи.

Redis може працювати у двох режимах [98]:

1. В оперативній пам’яті. Дані зберігаються в пам’яті одного сервера, що забезпечує мінімальні затримки доступу;

2. Розподілений кеш через кластеризацію У цьому режимі Redis розподіляє дані між кількома вузлами, що дозволяє масштабувати обсяг кешу і забезпечити стійкість до збоїв. Така архітектура робить Redis ідеальним для великих систем із високими навантаженнями.

Memcached – це проста та ефективна кеш-технологія, яка спеціалізується на зберіганні пар “ключ-значення” в оперативній пам’яті [99]. Основний акцент Memcached робить на максимальній швидкодії читання та запису, що досягається

завдяки легкій архітектурі та відсутності складних структур даних. Memcached ідеально підходить для кешування динамічних даних, таких як результати запитів або сесії користувачів.

Простота Memcached є його перевагою, але й обмеженням. Технологія не підтримує збереження даних після перезапуску або складні структури даних, тому більше підходить для сценаріїв, де кеш є тимчасовим сховищем. Memcached також дозволяє горизонтально масштабувати систему через додавання нових вузлів, однак він не має вбудованої кластеризації або реплікації, що може бути обмеженням для складних розподілених систем.

Hazelcast – це розподілена технологія кешування, що також функціонує як платформа для обробки даних у реальному часі [100]. Основною особливістю Hazelcast є здатність інтегруватися з іншими сховищами даних і працювати як у оперативній пам’яті, так і з постійними даними. Це робить Hazelcast ідеальним рішенням для складних систем, які потребують високого рівня узгодженості та підтримки транзакцій.

Hazelcast підтримує розподілену архітектуру, забезпечуючи масштабованість і високу доступність. Він також пропонує розширені функції, такі як підтримка розподілених транзакцій, що є важливим для систем із великими обсягами даних і вимогою до узгодженості. Hazelcast активно використовується для зберігання сесій, управління чергами повідомлень і обробки потоків даних у реальному часі [100].

В результаті аналізу вищезазначених технологій кешування [Додаток Г] найкращим варіантом для СППВПОН є Redis, оскільки в даній технології присутні:

1. Декілька режимів роботи. Redis може працювати як локальний кеш в оперативній пам’яті (*In-Memory Mode*) для прискорення обробки невеликих даних [101]. Окрім цього, кластеризація (*Distributed Mode*) дозволяє Redis функціонувати як розподілений кеш, що забезпечує масштабованість і стійкість до збоїв, необхідну для СППВПОН;

2. Швидкодія та продуктивність. Redis забезпечує мінімальні затримки для операцій читання та запису, що особливо важливо для систем із високими навантаженнями та великою кількістю одночасних запитів [102];

3. Підтримка складних структур даних. Redis дозволяє зберігати різні типи даних, включаючи рядки, списки, множини та геопросторові координати. Це робить його ідеальним для зберігання метаданих та інших ключових елементів СППВПОН;

4. Можливість автоматичного видалення даних. Функція дозволяє автоматично очищати кеш від застарілих даних, забезпечуючи оптимальне використання пам’яті [103];

5. Ефективна інтеграція з MongoDB. Redis легко інтегрується з MongoDB, що дозволяє організувати взаємодію між БД і кешем без додаткової складності [104];

6. Масштабованість. Завдяки кластеризації Redis може легко адаптуватися до зростаючих обсягів даних і збільшення кількості користувачів системи [105];

7. Надійність. Реплікація даних у Redis забезпечує стійкість до збоїв окремих вузлів, гарантуючи стабільну роботу системи навіть за умови високих навантажень [106];

8. Універсальність. Redis чудово підходить для роботи з даними, які активно використовуються в СППВПОН, такими як текстові описи пошкоджень, координати об’єктів, результати аналітики стану об’єктів нерухомості, технічні характеристики та дані про оцінку ступеня пошкоджень. Це дозволяє ефективно кешувати метадані та результати обчислень, забезпечуючи швидкий доступ до ключової інформації, необхідної для прийняття рішень у процесі відновлення пошкоджених об’єктів нерухомості [107].

Організація запису та зчитування інформації в кеші для СППВПОН буде виконуватись за допомогою техніки Cache-Aside, що дозволяє контролювати процес оновлення кешу [23]. Коли до системи додаються нові дані, наприклад, координати об’єкта чи опис пошкоджень, вони спочатку записуються в базу даних. Після успішного збереження в MongoDB, додаток записує відповідні дані в Redis. Зазвичай у кеш додаються ті поля, які найчастіше запитуються, наприклад: текстові описи, координати або результати аналітики. Це забезпечує швидкий доступ до ключових даних у наступних запитах.

Redis, як основна технологія кешування, добре інтегрується з мікросервісною архітектурою СППВПОН. Завдяки своїй здатності обробляти дані в оперативній

пам’яті та підтримувати розподілену кластерну архітектуру, Redis може стати основним компонентом для спільного використання мікросервісами. У системі СППВПОН кожен із них відповідає за різні функції, такі як: маршрутизація, авторизація, географічна інформація, обробка даних та інші [24], [25], [26], [108], [109]. Інтеграція Redis дозволяє кожному з них отримувати безпосередньо кешовані дані, зменшуючи кількість запитів до MongoDB і покращуючи загальну продуктивність системи.

3.6. Алгоритм та інтеграція гібридного методу інтелектуального управління кешем

У межах запропонованого гібридного методу інтелектуального управління кешем реалізовано послідовність дій, що охоплює три ключові задачі кешування: прийняття рішення про доцільність додавання об’єкта до кешу, вибір об’єкта для видалення у разі заповнення кешу, а також періодичну інвалідацію неактуальних даних. На рис. 3.6 подано загальну блок-схему роботи алгоритму гібридного методу інтелектуального управління кешем, яка демонструє взаємодію між кешем Redis, базою даних MongoDB і двома моделями машинного навчання – табличним Q-Learning та логістичною регресією.

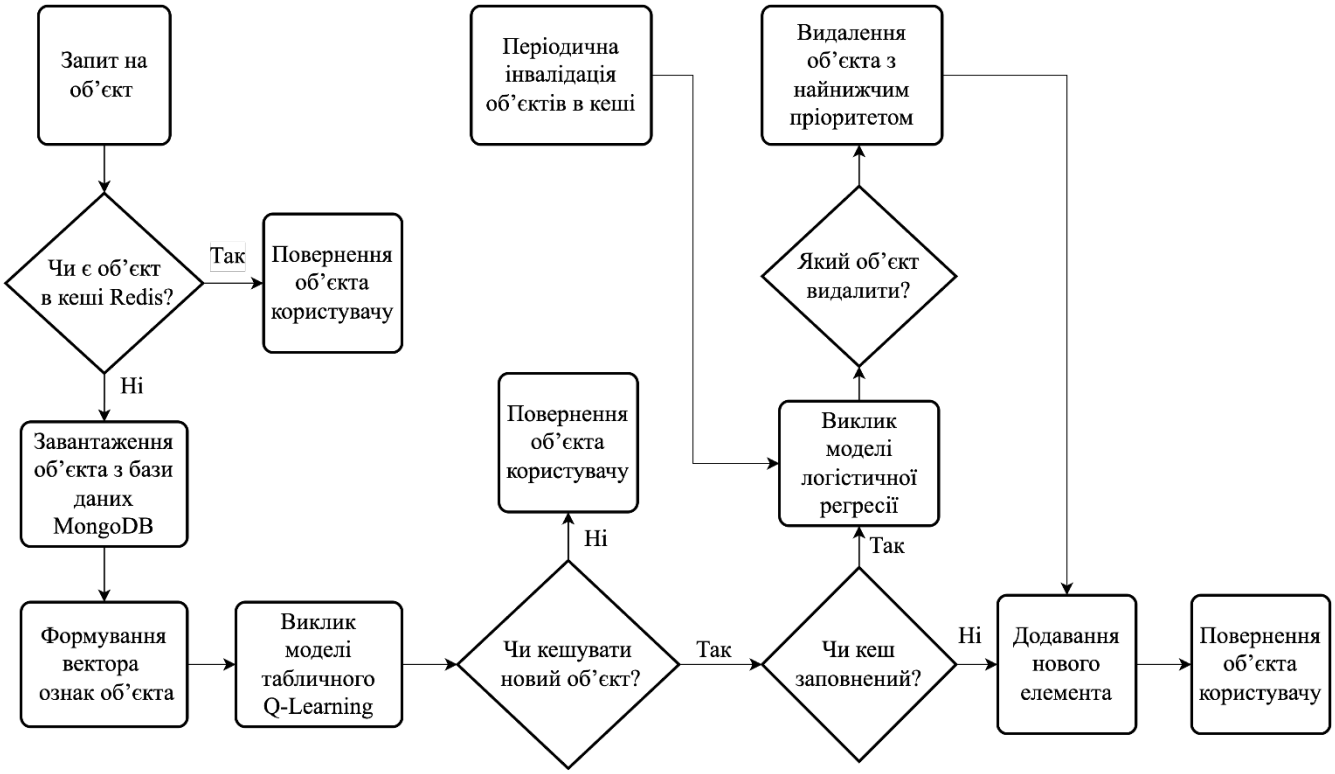

Рисунок 3.6. Блок-схема роботи гібридного методу інтелектуального управління кешем

Детальний опис алгоритму роботи запропонованого гібридного методу інтелектуального кешування виглядає наступним чином:

1. Отримання запиту:

   СППВПОН отримує запит на певний об'єкт від користувача та запускає процедуру обробки.

2. Перевірка об'єкта в технології кешування Redis:

– Якщо об'єкт знайдено (*cache hit*) – повертається користувачу та алгоритм завершується;

– Якщо об'єкта немає (*cache miss*) – обробка продовжується.

3. Завантаження об'єкта з бази даних MongoDB:

   СППВПОН отримує дані про об'єкт із бази.

4. Формування вектора ознак:

   На основі об'єкта, статистики та контексту формується вектор ознак.

5. Рішення моделі табличного Q-Learning:

– Модель оцінює чи варто додавати новий об’єкт у кеш;
– Якщо ні – об’єкт повертається користувачу та алгоритм завершується;
– Якщо так – перевіряється стан кешу.

6. Перевірка розміру кешу:
– Якщо є вільне місце – об’єкт додається до Redis, повертається користувачу та алгоритм завершується;
– Якщо кеш заповнений – викликається модель логістичної регресії.
7. Рішення моделі логістичної регресії:
– Для кожного об’єкта в кеші обчислюється ймовірність повторного використання;
– Визначається об’єкт, ймовірність повторного запиту до якого є найнижчою.
8. Оновлення кешу:
– Видаляється об’єкт, вибраний логістичною регресією;
– Новий об’єкт додається до Redis.
9. Періодична інвалідація об’єктів в кеші:
– За допомогою моделі логістичної регресії для кожного об’єкта в кеші обчислюється прогнозована ймовірність повторного використання $\hat{y}$;
– Видаляються всі об’єкти, для яких $\hat{y}$ менша за порогове значення θ;
10. Завершення обробки запиту:
– Об’єкт повертається користувачу;
– Модель табличного Q-Learning оновлює Q-значення;
– Періодично відбувається оновлення моделі логістичної регресії на основі нових навчальних прикладів.

Інтеграція модуля інтелектуального управління кешем у загальну модель СППВПОН дозволяє значно підвищити її продуктивність. Завдяки використанню запропонованого гібридного методу інтелектуального управління кешем, що поєднує моделі машинного навчання, СППВПОН може забезпечити ефективне управління кешем у режимі реального часу. Це дозволяє зменшити навантаження на

базу даних MongoDB, скоротити час обробки запитів та гарантувати швидкий доступ до найбільш важливої актуальної інформації.

На рис. 3.7. представлено оновлену модель СППВПОН із вбудованим модулем інтелектуального кешування AI Cache Manager, що реалізовує спроєктований гібридний метод інтелектуального управління кешем.

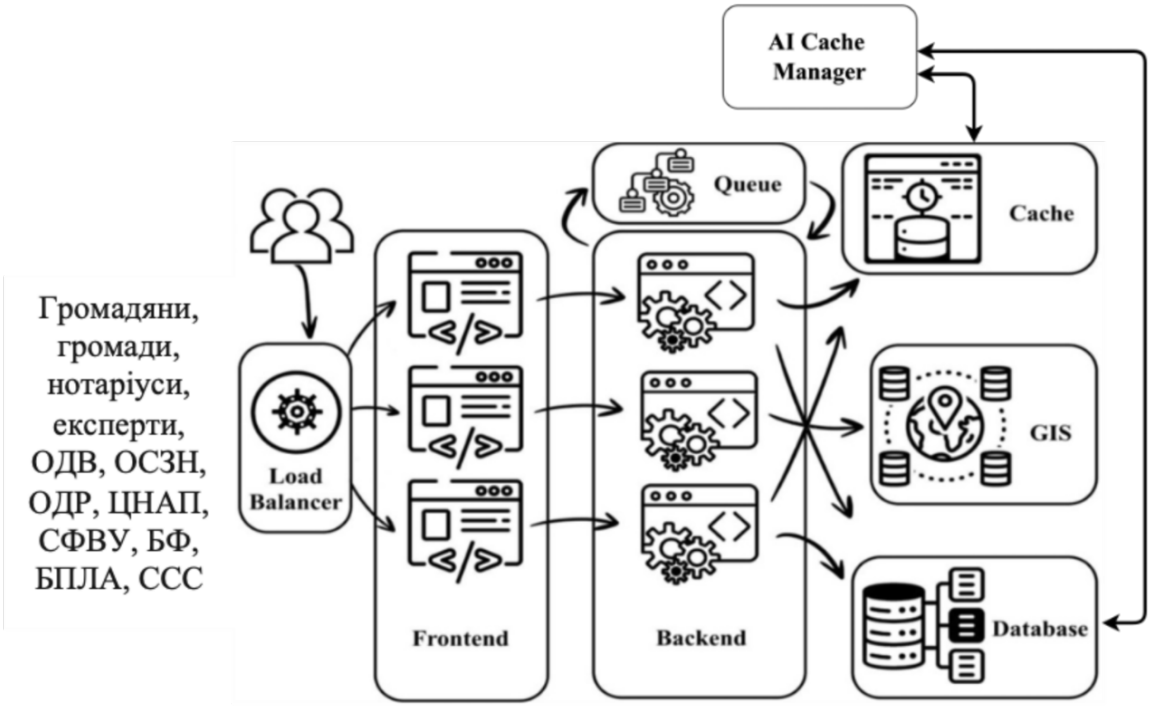

Рисунок 3.7. Інтеграція модуля інтелектуального кешування у модель інформаційної СППВПОН

Висновки до розділу 3

1. Сформульовано визначення методу управління кешем як узагальненого підходу до реалізації логіки кешування в інформаційній системі. Виокремлено три ключові задачі, які він охоплює: додавання об’єктів до кешу, видалення об’єктів у разі його заповнення та інвалідація застарілих даних. Визначено ключові метрики ефективності методу та обґрунтовано його роль як концептуальної основи для побудови гібридного інтелектуального підходу.

2. Формалізовано задачу оцінки доцільності додавання об’єкта до кешу у вигляді марковського процесу прийняття рішень. Запропоновано метод вирішення цієї задачі за допомогою використання моделі машинного навчання табличного Q-Learning. Визначено простір можливих дій, функцію винагороди та принцип відкладеного оновлення Q-значень через фіксований період часу. Сформовано вектор ключових ознак об’єкта в системі, що використовується для представлення стану в моделі та дозволяє її агенту приймати обґрунтовані рішення щодо доцільності додавання об’єкта до кешу. Спроектовано додаткову колекцію документів «Q-таблиця» для бази даних СППВПОН, необхідну для повноцінної роботи моделі табличного Q-Learning. Описано алгоритм вирішення задачі.

3. Формалізовано задачу вибору об’єкта для видалення з заповненого кешу як задачу бінарної класифікації. Запропоновано метод вирішення цієї задачі за допомогою використання моделі машинного навчання логістичної регресії. Визначено набір ознак, що описують об’єкт на момент видалення, а також механізм асинхронного накопичення навчальних прикладів у фоновому режимі, необхідний для ефективної роботи моделі. Спроектовано додаткову колекцію документів «Навчальні приклади» та «Параметри моделі» для бази даних СППВПОН, необхідних для повноцінної роботи моделі логістичної регресії. Описано алгоритм вирішення задачі.

4. Формалізовано задачу періодичної інвалідації неактуальних об’єктів у кеші. Запропоновано метод вирішення на основі логістичної регресії з використанням тієї ж

моделі, що і для задачі видалення у разі заповненого кешу. Визначено механізм періодичної перевірки актуальності об’єктів у кеші через фіксований інтервал часу. Описано алгоритм прийняття рішень про інвалідацію об’єктів залежно від ймовірності їх повторного використання, що обчислюється за допомогою моделі логістичної регресії. Описано алгоритм вирішення задачі.

5. Обґрунтовано доцільність використання Redis як технології кешування в СППВПОН. Показано, що Redis забезпечує необхідну продуктивність, гнучкість і масштабованість для інтеграції з гібридним методом інтелектуального управління кешем.

6. Описано алгоритм роботи гібридного методу інтелектуального управління кешем. Показано фінальну блок-схему його роботи із застосуванням моделей машинного навчання – табличного Q-Learning і логістичної регресії, технології бази даних MongoDB і технології кешування Redis. Обґрунтовано доцільність інтеграції модуля інтелектуального управління кешем у загальну модель СППВПОН.

РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ГІБРИДНОГО МЕТОДУ ІНТЕЛЕКТУАЛЬНОГО УПРАВЛІННЯ КЕШЕМ

У розділі описано структуру та основні функції системи експериментального тестування ефективності запропонованого гібридного методу інтелектуального управління кешем; наведено детальний опис компонентів системи, зокрема бази даних MongoDB, кеш Redis, двох моделей машинного навчання – табличного Q-Learning і логістичної регресії та модуля керування кешем, що реалізовує алгоритм роботи гібридного методу; реалізовано повноцінну програмну архітектуру системи, включно з агентами, буферами, оновленням моделей у фоновому асинхронному режимі та синхронізацією з базою даних MongoDB і кешем Redis; продемонстровано результати роботи системи у двох типах експериментів: аналіз частки кеш-хітів після одного запуску та серії запусків з накопичувальним навчанням моделей з фіксованим набором даних; у кожному експерименті проведено порівняння ефективності запропонованого гібридного методу управління кешем з найпоширенішими існуючими методами кешування: LRU та LFU; візуалізовано результати у вигляді графіків і рисунків, що підтверджують перевагу гібридного методу за умов обмеженого розміру кешу та кількості запитів.

4.1. Система експериментального тестування ефективності гібридного методу інтелектуального управління кешем

Для перевірки ефективності запропонованого гібридного методу інтелектуального управління кешем запропоновано створити окрему систему тестування. Вона дозволяє окремо аналізувати ефективність роботи кешу, БД, моделей машинного навчання та результати обробки запитів.

Важливим аспектом цієї системи є те, що вона імітує реальні умови використання: всі дані про об’єкти генеруються один раз на початку і зберігаються між запусками, а запити формуються псевдовипадковим чином із фіксованим

генератором випадкових чисел. Це забезпечує відтворюваність експериментів, дозволяючи оцінювати ефективність моделей у стабільному середовищі.

Система експериментального тестування інтелектуального кешування складається з наступних компонентів:

4.1.1. MongoDB

У системі тестування MongoDB використовується як БД, що містить усі необхідні дані для перевірки ефективності запропонованого гібридного методу інтелектуального упрравління кешем. Звідти завантажуються об’єкти з усіх основних колекцій документів, що дозволяє моделювати реальні запити і будувати ключові ознаки для прийняття рішень щодо кешування.

MongoDB забезпечує двосторонню взаємодію з програмним модулем: дані з колекцій пошкоджених об’єктів нерухомості, історії запитів і користувачів використовуються для створення динамічного потоку запитів до об’єктів, який наближено відтворює реальні сценарії використання СППВПОН, а результати виконання кешування зберігаються у спеціальних колекціях для подальшого аналізу. Це дозволяє фіксувати перебіг експериментів, порівнювати ефективність різних стратегій кешування та виявляти закономірності в їх поведінці.

4.1.2. Redis

Redis у системі тестування виступає в ролі технології кешування, де тимчасово зберігаються об’єкти, до яких були згенеровані запити. Основна функція цього компоненту – забезпечити доступ до даних з мінімальною затримкою, не навантажуючи БД.

Кеш постійно оновлюється відповідно до результатів роботи агентів моделей машинного навчання, які приймають рішення про збереження або видалення конкретних об’єктів. Зміни у вмісті кешу можна відстежити через стандартні

інструменти моніторингу Redis, що також дає змогу оцінювати продуктивність системи загалом.

4.1.3. Модуль керування

Модуль керування виконує основну функцію – реалізацію алгоритму роботи запропонованого гібридного методу інтелектуального управління кешем. Його завдання полягає в послідовній обробці всіх запитів до об’єктів і забезпеченні правильної взаємодії між кешем Redis, БД MongoDB і моделями машинного навчання.

4.1.4. Моделі машинного навчання

У системі тестування реалізовано дві окремі моделі машинного навчання, кожна з яких вирішує свою підзадачу керування кешем.

Перша модель – це агент з навчанням з підкріпленням (Reinforcement Learning), побудований за принципом табличного Q-Learning. Вона приймає рішення про доцільність кешування нового об’єкта на основі сформованого вектора ознак, що відображає поточний стан об’єкта в контексті системи експериментального тестування. Модель навчається під час роботи, оновлюючи свої Q-значення залежно від отриманої винагороди.

Друга модель – це логістична регресія, використовується у випадках, коли об’єкт слід додати в кеш, але він вже заповнений. Вона виконує бінарну класифікацію для об’єктів, що перебувають у кеші, оцінюючи ймовірність їх подальшого використання.

Обидві моделі взаємодіють із модулем керування, який забезпечує обчислення ознак об’єкта, виклик моделей у потрібний час та інтерпретацію результатів для подальших дій.

4.1.5. Архітектура системи

Усі описані компоненти системи експериментального тестування взаємодіють між собою, що забезпечує повний цикл обробки запиту, а саме: від запиту до певного об’єкта до прийняття рішень про доцільність кешування на основі результатів роботи моделей машинного навчання.

У структурному плані архітектура побудована за принципом розподілу відповідальності, де кожен компонент виконує чітко визначену функцію. Дані передаються між компонентами через послідовні етапи обробки, що дозволяє легко масштабувати систему або змінювати окремі частини без впливу на загальну логіку. Такий підхід забезпечує гнучкість і узгодженість між компонентами, а також дозволяє проводити ізольоване тестування кожного з них.

З технічної точки зору, архітектура відповідає принципу слабкої зв’язаності, коли компоненти взаємодіють через чітко визначені інтерфейси й не залежать від внутрішньої реалізації один одного. Це дає змогу проводити окреме профілювання Redis, MongoDB або моделей машинного навчання, а також швидко адаптувати структуру під різні сценарії навантаження.

На рис. 4.1. зображено архітектуру системи експериментального тестування ефективності гібридного методу інтелектуального управління кешем, де модуль керування виступає центральним елементом, що відповідає за процедуру обробки запиту та координує роботу між кешем Redis, базою даних MongoDB та моделями машинного навчання.

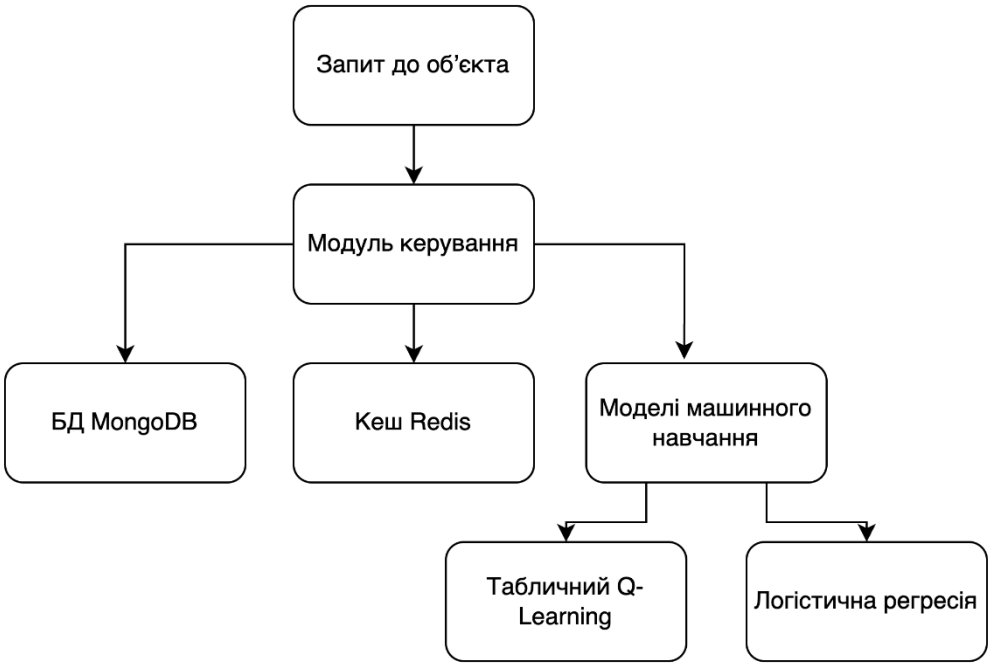

Рисунок 4.1. Архітектура системи експериментального тестування ефективності гібридного методу інтелектуального управління кешем

4.2. Програмна реалізація системи

На основі визначеної архітектури та компонентів було реалізовано повноцінну систему тестування, що дозволяє оцінити ефективність запропонованого гібридного методу інтелектуального управління кешем у режимі, наближеному до реальних умов експлуатації. Система охоплює модулі для обробки запитів, управління кешем, роботи з БД, а також взаємодії з моделями машинного навчання.

4.2.1. Мова програмування

Для реалізації системи експериментального тестування було обрано мову програмування Python. Ця мова є однією з найпопулярніших у світі, особливо в науковому, освітньому та прикладному середовищі. Основна її перевага полягає в простоті синтаксису, що дозволяє зосередитися на логіці програми, а не на технічних деталях реалізації. Завдяки цьому Python активно використовується в

дослідницьких розробках, навчальних проєктах та створенні прототипів складних систем [110].

Python має низький поріг входу, тому вважається однією з найбільш зручних мов для створення систем, які постійно змінюються або розвиваються. Висока читабельність коду та структурованість дозволяють швидко орієнтуватися в проєкті навіть після тривалих перерв, а також легко підтримувати код у довгостроковій перспективі.

Ще однією важливою перевагою Python є його універсальність [111]. Ця мова підходить як для невеликих навчальних завдань, так і для побудови повноцінних інформаційних програмних систем. Вона однаково добре працює на різних операційних системах і підтримується більшістю сучасних середовищ розробки.

У межах цього проєкту було використано версію Python 3.9, яка є стабільною та достатньо сучасною для реалізації складної логіки системи. Вибір саме цієї версії зумовлений її сумісністю з усіма необхідними можливостями мови, що необхідні для реалізації запропонованого гібридного методу інтелектуального управління кешем[112]. Таким чином, Python став надійною основою для програмної імплементації системи експериментального тестування.

4.2.2. Середовище розробки

Під час створення програмної частини важливим кроком було вибрати зручне середовище, в якому можна було б ефективно писати, перевіряти та запускати код. Такі середовища часто називають середовищами розробки або IDE (*Integrated Development Environment*). Вони об’єднують в одному вікні усе необхідне для програміста: редактор коду, засоби перевірки помилок, запуску програм, а також підтримку структури проєкту й керування файлами [113].

Серед багатьох доступних середовищ було обрано PyCharm – одну з найпопулярніших програм у світі для написання коду на Python. Її розробляє компанія JetBrains, і вона має дві основні версії: платну та безкоштовну. У цьому

проєкті використовувалась саме безкоштовна версія, оскільки вона містить усі функції, необхідні для реалізації системи тестування [114].

PyCharm – це програма, яка дозволяє зручно працювати з усім кодом у межах одного проєкту. У ній зручно створювати нові файли, групувати їх у папки, швидко перемикатися між модулями та бачити структуру всієї системи. Це особливо важливо тоді, коли проєкт має багато компонентів.

Однією з головних переваг PyCharm є автоматичне підсвічування синтаксису. Це означає, що код одразу виглядає структуровано: змінні, функції та ключові слова мають різні кольори. Це допомагає швидше орієнтуватися в коді та уникати помилок. Крім того, PyCharm підказує можливі продовження коду під час набору, що економить час і дозволяє писати точніше.

Для запуску програм PyCharm має вбудовану панель управління. Це означає, що будь-який файл можна запустити безпосередньо з інтерфейсу програми, без необхідності відкривати термінал. Крім того, можна легко задавати аргументи запуску, вибирати середовище виконання або запускати проєкт з різними налаштуваннями.

У цьому проєкті важливу роль відігравало також використання так званого віртуального середовища. Це спеціальна ізольована оболонка, в якій встановлюються лише ті бібліотеки, які потрібні саме для цього проєкту. PyCharm підтримує створення та використання таких середовищ автоматично, що дозволяє уникнути конфліктів між різними версіями програмних пакетів. У межах системи тестування було створено окреме середовище, в якому були встановлені всі необхідні бібліотеки для роботи з MongoDB, Redis моделями машинного навчання та обробкою результатів.

Окремо варто відзначити зручність у роботі з файлами. У PyCharm можна швидко відкривати будь-який файл за назвою, переглядати останні зміни, а також зручно переміщувати функції або класи між різними частинами проєкту. Це дає змогу підтримувати порядок у структурі системи, що дуже важливо при роботі з кількома модулями одночасно.

Порівняльна таблиця (табл. 4.1.) дає змогу оцінити, наскільки PyCharm підходить для реалізації системи експериментального тестування у порівнянні з іншими популярними середовищами розробки.

Таблиця 4.1.

Порівняльний аналіз найпопулярніших середовищ розробки мовою Python

Критерій	PyCharm Community	Visual Studio Code	Spyder
Зручність роботи з великим проєктом	Висока	Висока (з розширеннями)	Середня
Підсвічування та підказки коду	Так	Так	Частково
Підтримка віртуального середовища	Повна	Повна	Часткова
Вбудований запуск і налагодження	Так	Так	Так
Проста інтеграція з базами даних	Так	Через розширення	Немає
Інтеграція з технологіями кешування (Redis)	Так	Частково	Немає
Інтеграція з моделями штучного інтелекту	Так	Частково	Частково
Наявність готових шаблонів проєктів	Так	Частково	Ні
Підходить для навчання і досліджень	Так	Так	Так

Таким чином, PyCharm забезпечив повноцінне середовище для реалізації, тестування та підтримки системи експериментального тестування гібридного методу інтелектуального управління кешем.

4.2.3. MongoDB

У реалізованій системі експериментального тестування використано MongoDB версії 6.0, що забезпечує стабільну підтримку роботи з великими обсягами вкладених даних. База даних функціонує у форматі BSON (розширений двійковий формат JSON), який дозволяє зберігати вкладені об’єкти, списки, логічні значення, часові мітки та інші типи, необхідні для моделювання складної структури об’єктів.

З’єднання з базою реалізовано через Python-драйвер pymongo, що підтримує усі необхідні операції: зчитування, фільтрацію, вставку та оновлення документів [115].

Загальна структура охоплює сім колекцій. Чотири з них пов’язані з вхідними даними, що описують поведінку системи в конкретний момент часу:

1. damaged_buildings – зберігає детальну інформацію про пошкоджені об’єкти. Кожен документ містить дані про місцезнаходження. Зокрема, координати, місто, район і вулицю, тип будівлі, рівень ушкоджень, дату фіксації, опис проблеми, джерело інформації, контактну особу. Також фіксуються дані про технічний стан і наявність огляду;

2. request_history – містить історію запитів до об’єктів. Фіксується тип запиту, ідентифікатор користувача та об’єкта, час звернення, категорія об’єкта та регіон. Ця колекція дозволяє простежити динаміку звернень і використовувати її як джерело ознак для прийняття рішень;

3. city_stats – надає агреговану інформацію про регіони: відсоток пошкоджених і зруйнованих об’єктів, орієнтовні збитки, кількість постраждалих і загиблих;

4. users – містить відомості про користувачів системи: ім’я, роль, організацію, контактну інформацію, дату реєстрації та стан активності. Ці дані дають можливість враховувати тип і авторитетність джерела звернень.

На базі цих чотирьох колекцій формуються вектори ознак, які подаються на вхід для моделей машинного навчання. Ознаки включають тип об’єкта, інтенсивність запитів, пріоритетність, відстань у часі від останнього запиту,

контекст регіону тощо. Формат зберігання дозволяє виконувати вибірки з урахуванням складених фільтрів, що наближує систему до умов реального навантаження.

Три інші колекції мають службове призначення та підтримують роботу моделей машинного навчання:

1. q_table – містить зв’язки між станами, діями та оцінками значущості (Q-значеннями), які використовуються в моделі табличного Q-Learning. Формат документів компактний, у вигляді трійок «стан–дія–оцінка».

2. lr_training_examples – містить історичні приклади, що використовуються для навчання моделі логістичної регресії. Кожен документ – це вектор ознак об’єкта та відповідна мітка, що вказує, чи буде повторний запит до нього після видалення з кешу.

3. lr_model_params – зберігає поточні параметри логістичної моделі, зокрема вагові коефіцієнти та зміщення. Ці значення можуть періодично оновлюватися після надходження нових прикладів до колекції lr_training_examples.

З технічної точки зору, структура документів побудована з урахуванням вкладеності, підтримки масивів і часових міток. Це забезпечує повноцінну імітацію реального застосування без потреби у зовнішньому форматуванні або трансформації даних. Усі колекції можуть вільно переглядатися через графічне середовище MongoDB Compass, що дозволяє контролювати цілісність даних під час тестування [116], [117].

На рис. 4.2. зображено приклад повноцінного документа в колекції damaged_buildings разом із загальним виглядом структури всієї бази даних. Наведений приклад ілюструє використання вкладених об’єктів, масивів і часових міток, що відображають логіку зберігання даних у системі та відповідають вимогам до вхідної інформації для моделей.

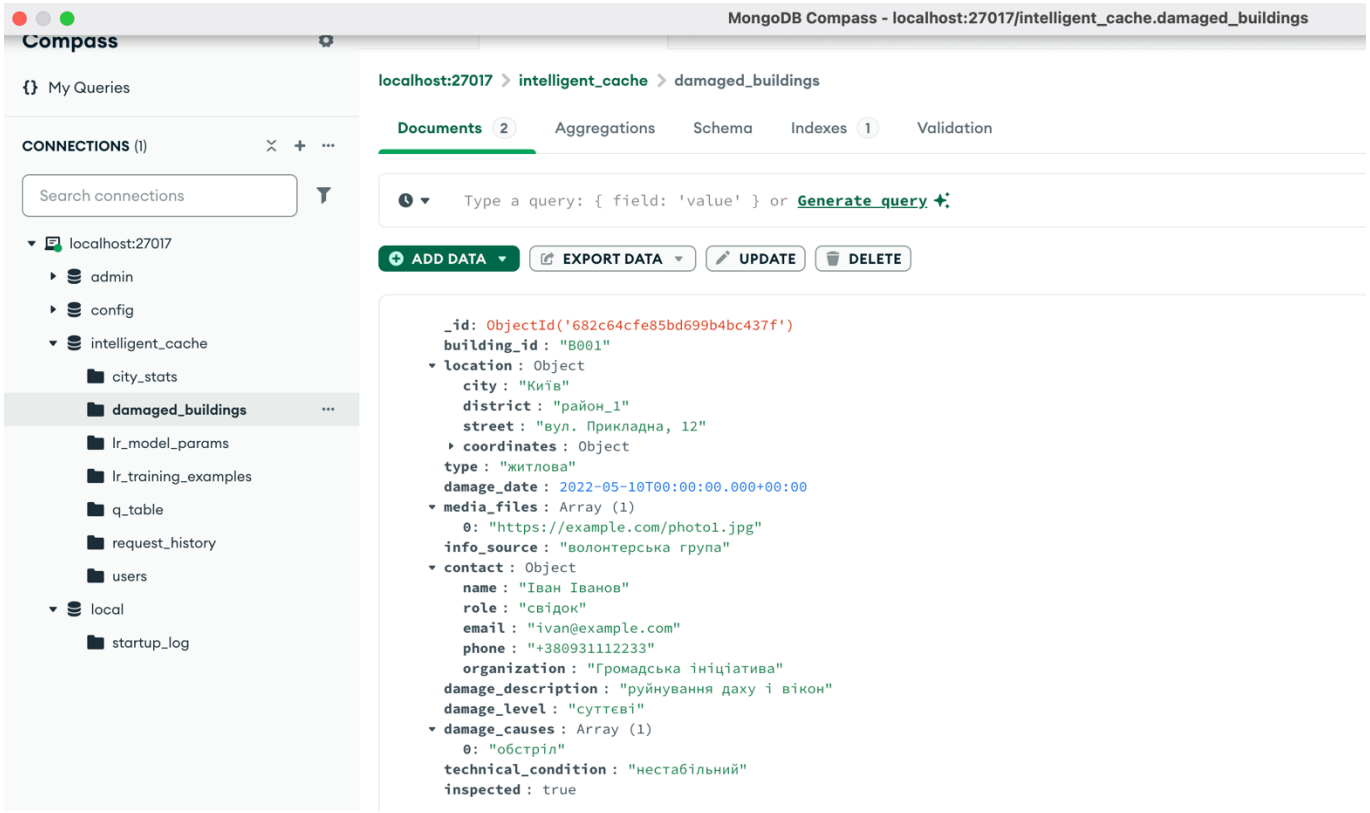

Рисунок 4.2. Документ у колекції damaged_buildings та структура бази даних MongoDB

4.2.4. Redis

Redis у системі експериментального тестування інтегровано як технологію кешування, з якою відбувається пряме з'єднання з коду. Взаємодія реалізована за допомогою бібліотеки redis-py, яка дозволяє виконувати базові операції GET, SET, DEL, EXISTS у межах логіки роботи моделей машинного навчання [118], [119].

У Redis зберігаються серіалізовані об'єкти у форматі JSON [120]. Ключі формуються за шаблоном building:<ID>, що забезпечує однозначну адресацію кожного об'єкта у кеші. Значенням є повний об'єкт з колекції «damaged_buildings» з MongoDB, що зберігає структуру вкладеності, масивів та типів без втрати інформації.

На рис. 4.3. зображено повний вміст одного з ключів кешу, що демонструє приклад повноцінного документа.

```
redis-cli set building:B001 '{
  "building_id": "B001",
  "location": {
    "city": "Київ",
    "district": "район_1",
    "street": "вул. Прикладна, 12",
    "coordinates": {
      "lat": 50.45,
      "lon": 30.52
    }
  },
  "type": "житлова",
  "damage_date": "2022-05-10T00:00:00Z",
  "media_files": ["https://example.com/photo1.jpg"],
  "info_source": "волонтерська група",
  "contact": {
    "name": "Іван Іванов",
    "role": "свідок",
    "email": "ivan@example.com",
    "phone": "+380931112233",
    "organization": "Громадська ініціатива"
  },
  "damage_description": "руйнування даху і вікон",
  "damage_level": "суттєві",
  "damage_causes": ["обстріл"],
  "technical_condition": "нестабільний",
  "inspected": true
}'
```

Рисунок 4.3. Додавання об’єкта до Redis через термінал за допомогою команди SET

У реалізації об’єкти в кеші не мають фіксованого часу життя. Додавання та видалення ключів виконується виключно за рішеннями агентів моделей машинного навчання. Це дає змогу повністю контролювати вміст Redis з боку системи, а саме – через модуль керування, який реалізовує алгоритм гібридного методу інтелектуального управління кешем.

У процесі тестування структура кешу контролюється через графічне середовище Another Redis Desktop Manager, яке надає можливість переглядати ключі, значення та внутрішню структуру об’єктів [121]. На рис. 4.4. показано огляд активних ключів, присутніх у графічному середовищі на момент перевірки.

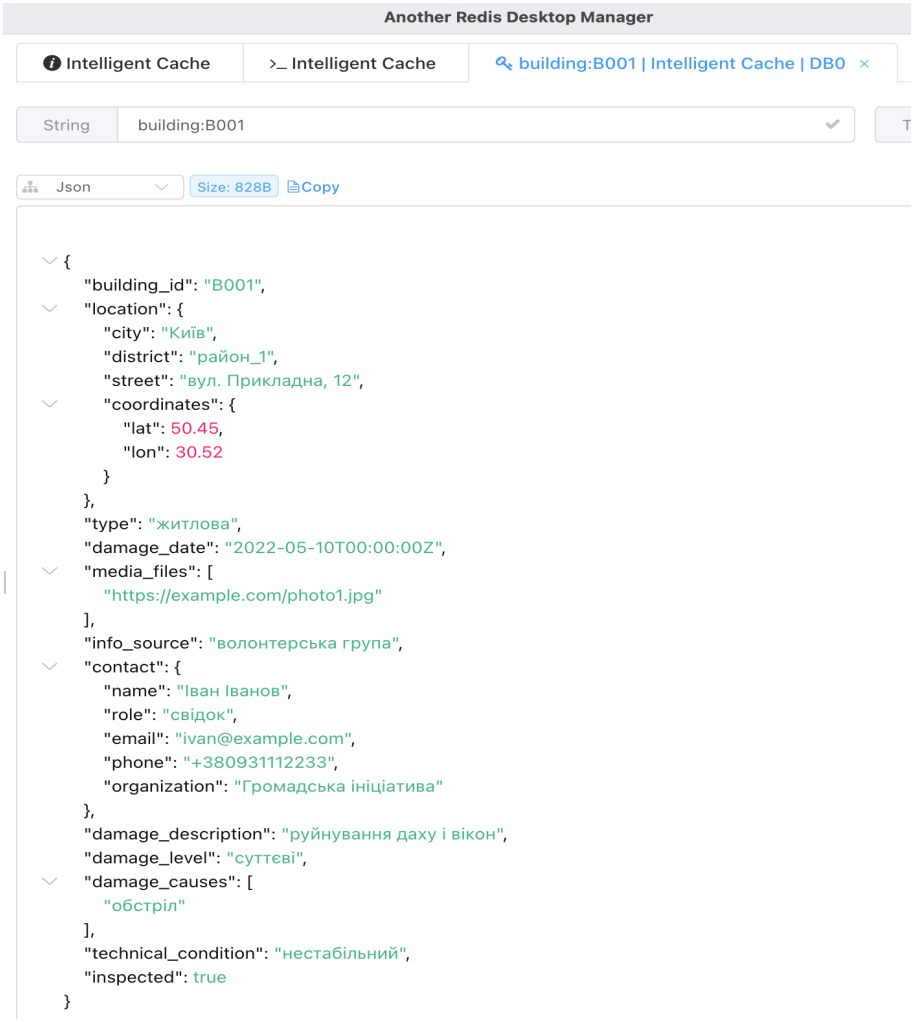

Рисунок 4.4. Приклад об’єкта у кеші Redis, відображеного у графічному середовищі Another Redis Desktop Manager

4.2.5. Моделі машинного навчання

У межах системи реалізовано дві моделі машинного навчання – табличний Q-Learning та логістичну регресію, кожна з яких представлена як окремий клас в коді із чітко визначеним інтерфейсом для виклику та оновлення. Моделі функціонують як незалежні агенти, інтегровані у загальний цикл обробки запитів.

Модель табличного Q-Learning побудована на основі словникової структури, де ключами є серіалізовані стани, а значеннями – списки оцінок дій. У коді реалізовано логіку вибору дії на основі ε-жадібної стратегії, а також оновлення значень в Q-таблиці за результатами взаємодії системи з даними. Усі зміни зберігаються в MongoDB для збереження оновлених даних між сесіями.

Окрему увагу приділено стратегії оновлення Q-значень: у системі використовується відкладене оновлення, яке виконується не після кожного запиту, а з певною періодичністю в окремому потоці. Це дозволяє зменшити навантаження на пам’ять і стабілізувати процес навчання.

Логістична регресія реалізована з використанням бібліотеки scikit-learn як адаптована модель із накопичуваним буфером прикладів**.** У процесі роботи до нього додаються нові приклади, отримані під час кожного рішення про видалення об’єкта з кешу. Через фіксований інтервал T, відбувається перенавчання моделі. Модель оновлюється на основі нових прикладів, що дозволяє поступово адаптуватися до змін у динаміці та характеру запитів.

Після навчання оновлені параметри моделі, а саме: ваги та зміщення зберігаються у колекції «lr_model_params». У момент запуску системи передбачено механізм ініціалізації: якщо модель та її параметри присутні, вони автоматично завантажуються з БД; якщо відсутні або пошкоджені, то створюється новий екземпляр моделі з початковими значеннями параметрів. Це забезпечує постійну роботи логістичної регресії та можливість її поступового навчання.

На рис. 4.5. зображено структурну схема класів, задіяних у реалізації гібридного методу інтелектуального управління кешем. Зв’язки між класами позначено відповідно до нотації UML-діаграми класів. LogRegAgent і QLearningAgent реалізовують спільний інтерфейс IAgent, що забезпечує уніфікований підхід до виклику моделей у процесі обробки запитів. Відповідно, LogRegUpdater і QLearningUpdater реалізовують інтерфейс IUpdater та відповідають за періодичне оновлення параметрів моделей.

QTable містить логіку роботи з Q-значеннями, а LogRegModel реалізовує логістичну модель з параметрами. LogRegBuffer і QLearningBuffer зберігають навчальні приклади для подальшого навчання та оновлення моделей. Центральним елементом є Controller, який виконує функції модуля керування та координує взаємодію між агентами, оновлювачами, буферами та моделями, забезпечуючи цілісну логіку гібридного методу.

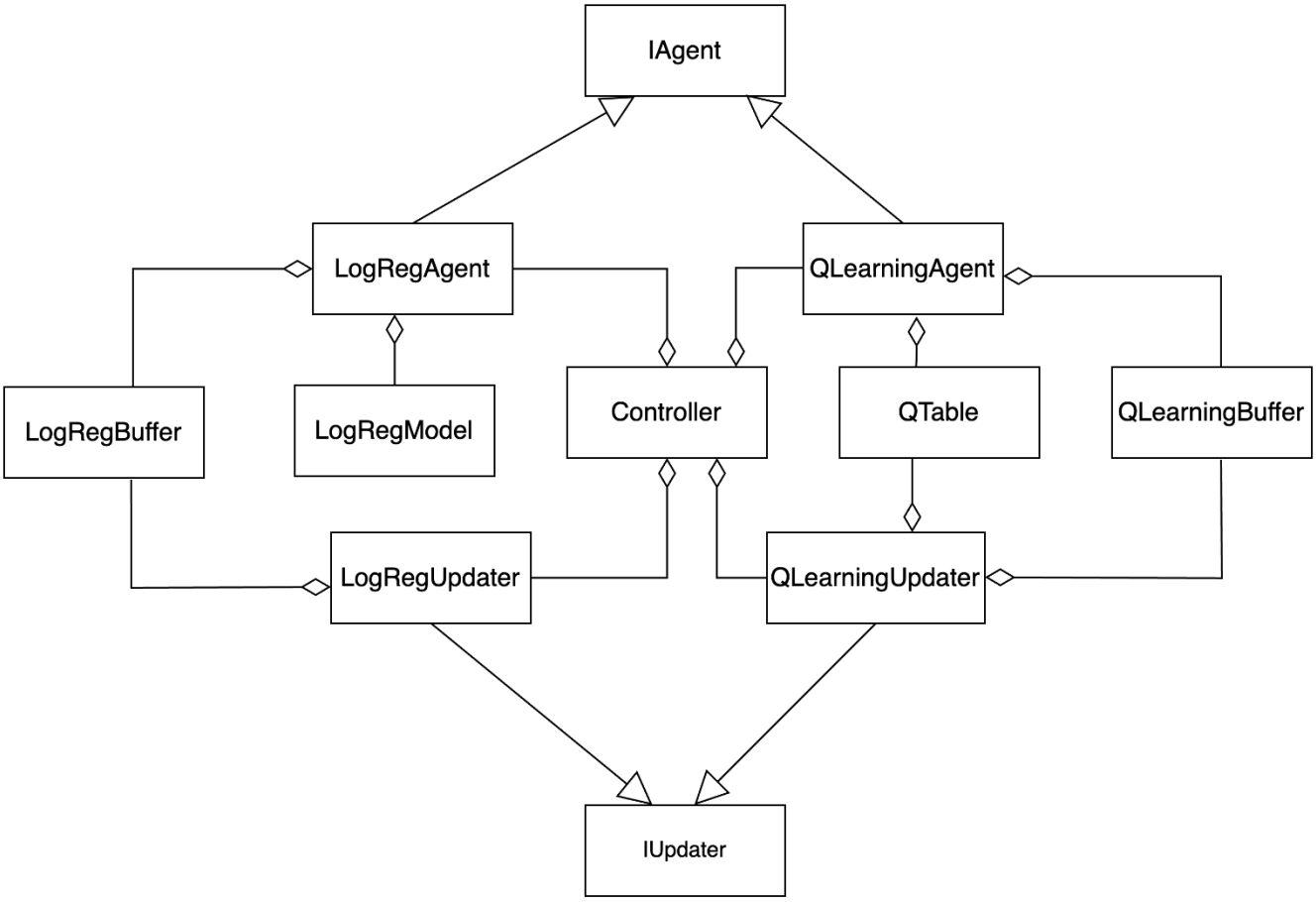

Рисунок 4.5. Структурна схема класів, задіяних у реалізації гібридного методу інтелектуального управління кешем

Таким чином, реалізація обох моделей машинного навчання побудована з урахуванням вимог до гнучкості та стабільності в умовах багаторазових ітерацій. Завдяки чіткій модульній структурі та можливості незалежного оновлення, ці компоненти легко масштабуються і можуть бути адаптовані до інших інформаційних систем, де кешування відіграє важливу роль, а динаміка та контекст запитів часто змінюються.

4.2.6. Модуль керування

Модуль керування реалізує всі етапи логіки гібридного методу інтелектуального управління кешем. Саме він послідовно обробляє згенеровані

запити до об’єктів, координує взаємодію з Redis, MongoDB і моделями машинного навчання, а також відповідає за оновлення моделей у фоновому режимі.

Центральною частиною модуля є функція обробки запиту, яка включає наступні кроки:

1. Перевірка наявності об’єкта в кеші Redis;
2. Завантаження об’єкта з MongoDB у разі cache miss;
3. Формування вектора ознак для моделей машинного навчання;
4. Застосування моделі табличного Q-Learning для прийняття рішення щодо додавання нового об’єкта в кеш;
5. Перевірка заповненості кешу;
6. Виклик моделі логістичної регресії, якщо необхідно звільнити місце;
7. Оновлення кешу Redis відповідно до прийнятого рішення;
8. Фіксація результатів для подальшого аналізу частки кеш-хітів.

Окремі процеси виконуються у фоновому або відкладеному режимі:

– Оновлення Q-таблиці: відбувається з фіксованою затримкою після дії. Для цього використовується список переходів, що зберігає стан, дію, ідентифікатор об’єкта та час виконання. Через кожен інтервал часу відбувається перевірка, які записи готові до оновлення;

– Оновлення моделі логістичної регресії: здійснюється періодично та її параметрів виконується періодично. Протягом симуляції формується буфер навчальних прикладів. Після накопичення достатнього обсягу прикладів модель оновлюється за методом градієнтного спуску, а нові параметри зберігаються в колекції.

На рис. 4.6. подано фрагмент коду, що виконує повний цикл обробки одного запиту та демонструє взаємодію всіх компонентів системи.

```
# --- Основний цикл ---
for i in range(len(requests)):
    req = requests[i]
    object_id = req["object_id"]
    obj = all_objects[object_id]

    shared_index.set(i)

    # --- Підрахунок хітів ---
    if lru_cache.get(object_id) is not None:
        lru_hits += 1
    else:
        lru_cache.add(obj)
    if lfu_cache.get(object_id) is not None:
        lfu_hits += 1
    else:
        lfu_cache.add(obj)
    if int_cache.get(object_id) is None:
        action = q_agent.act(object_data=obj, current_index=i)
        if action == 1:
            if int_cache.is_full():
                to_evict = logreg_agent.act(current_index=i)
                if to_evict:
                    int_cache.delete(to_evict)
            int_cache.add(obj)
    else:
        int_hits += 1
```

Рисунок 4.6. Основний цикл обробки запитів і підрахунок кеш-хітів для стратегій LRU, LFU та інтелектуального кешу

4.3. Перевірка ефективності гібридного методу інтелектуального управління кешем

Усі експерименти проводилися в окремому середовищі із симульованими даними. Об’єкти генерувалися випадково з фіксованими закономірностями, які відображали найбільш ймовірні типи запитів у СППВПОН. Основна увага приділялась сценаріям з обмеженим числом унікальних об’єктів (до 100) та відносно малим розміром кешу (до 20% від загальної кількості об’єктів). У таких умовах запропонований гібридний метод демонструє помітну перевагу над LRU та LFU методами завдяки здатності адаптуватися до характеру запитів навіть у рамках невеликої вибірки.

Для оцінки ефективності реалізованого гібридного методу інтелектуального управління кешем було проведено два типи експериментів. Перший тип (рис. 4.7.) відображає зміну частки кеш-хітів після одного запуску системи залежно від кількості запитів. Запити формувались на основі випадково згенерованих об’єктів із фіксованими ознаками. У цьому сценарії можна простежити, як система адаптується до характеру запитів: стратегія, що використовує комбінацію Q-Learning і логістичної регресії показує вищу стабільність та вищий частку кеш-хітів порівняно з класичними LRU і LFU (рис. 4.8.).

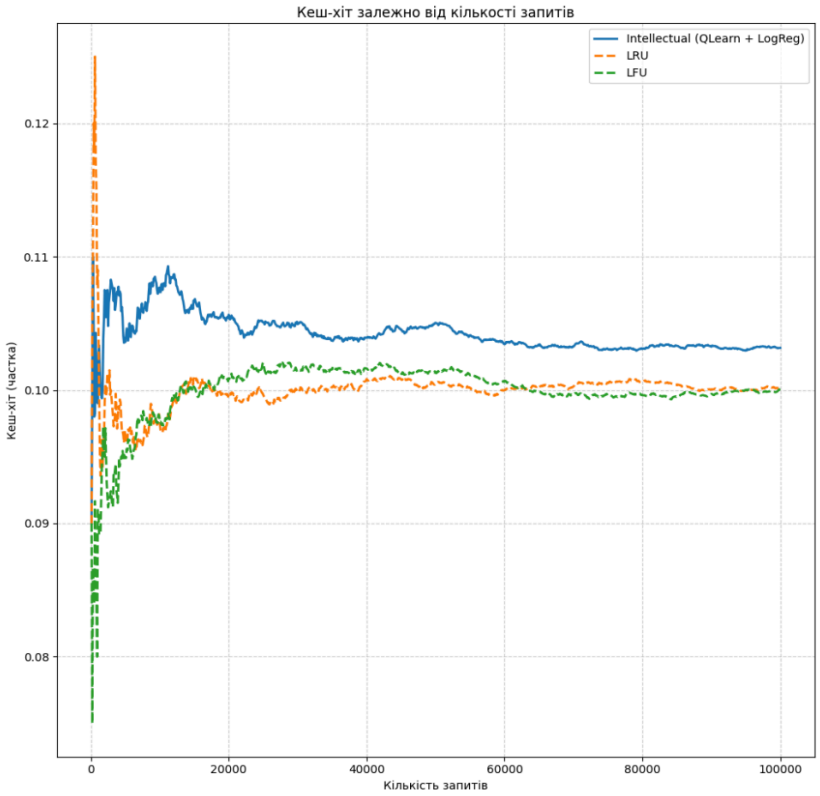

Рисунок 4.7. Графік частки кеш-хітів для трьох методів управління кешем у системі експериментального тестування залежно від кількості запитів

```
--- Конфігурація експерименту ---
Кількість унікальних об’єктів:   100
Об’єм кешу:                      10
Загальна кількість запитів:      100000

--- Результати ---
Кеш-хіти (Intellectual):        10316
Кеш-хіти (LRU):                 10015
Кеш-хіти (LFU):                 10007

Покращення порівняно з LRU:     3.01%
Покращення порівняно з LFU:     3.09%
```

Рисунок 4.8. Статистика частки кеш-хітів для трьох методів управління кешем після 1 запуску системи

У другому типі експерименту (рис. 4.9.) система запускалась багаторазово з том самим набором об’єктів. Єдиною змінною була черговість запитів, яка формувалась випадково для кожного запуску. При цьому всі інші параметри залишались незмінними: ознаки об’єктів, початкові умови, розмір кешу, кількість запитів. Важливо, що між всіма запусками системи зберігалися: буфер нових навчальних прикладів, параметри моделі логістичної регресії, а також оновлені значення в Q-таблиці. Таким чином, моделі продовжували вчитися між всіма запусками системи.

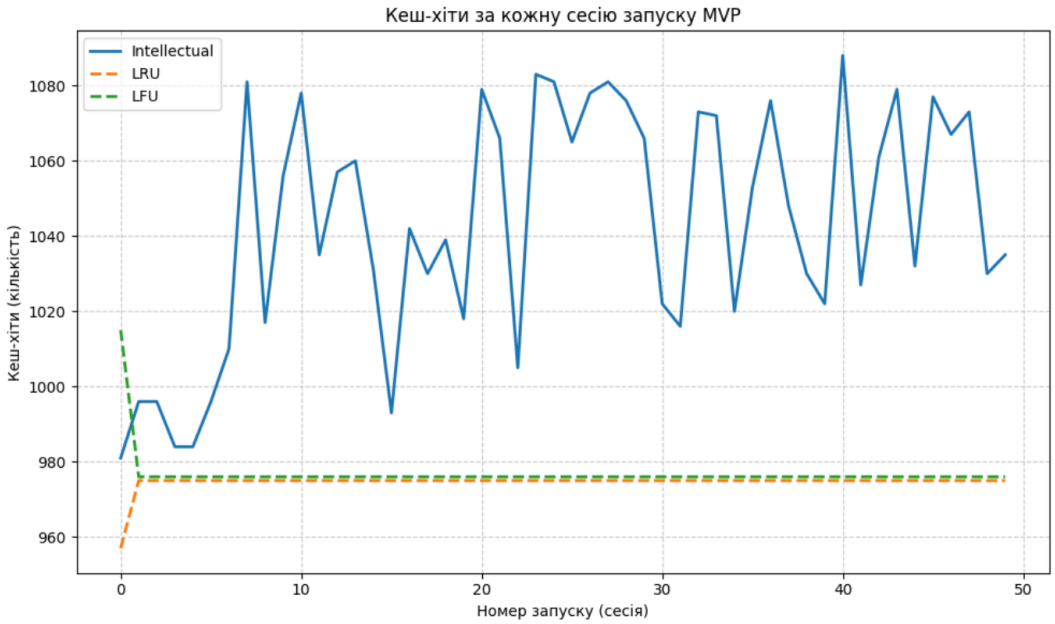

Рисунок 4.9. Графік кількості кеш-хітів для трьох методів управління кешем після 50 запусків системи

Цей підхід дозволив оцінити не лише поточну ефективність, а й здатність системи до покрокового покращення через накопичення досвіду. Існуючі методи кешування демонструють сталий рівень кількості кеш-хітів у всіх сесіях, тоді як гібридний метод інтелектуального управління кешем поступово покращує результати, адаптуючись до нових варіантів запитів. Це підкреслює перевагу запропонованого підхожу не лише в окремому запуску, а й у тривалому застосуванні. У той час як класичних методах кешування LRU та LFU демонструють стабільний, але обмежений рівень кількості кеш-хітів, гібридний метод за кілька запусків суттєво покращує результати. Це свідчить про здатність моделі адаптуватися до специфіки запитів через механізми відкладеного навчання.

Особливо помітна перевага інтелектуального кешу в умовах обмеженого розміру кешу (рис. 4.10.): моделі з машинним навчанням поступово накопичують знання про шаблони запитів, що дозволяє витісняти менш релевантні об’єкти та зберігати в кеші ті, що мають вищу ймовірність повторного використання. У результаті досягається вищий відсоток кеш-хітів навіть у порівнянні з існуючими методами кешування.

```
📊 Запускаємо 50 сесій кешування...
--- Конфігурація експерименту ---
Кількість унікальних об’єктів: 100
Об’єм кешу: 10
Загальна кількість запитів: 10000


=== 📊 ПІДСУМКИ ЗА ВСІ СЕСІЇ ===
Середній кеш-хіт (Intellectual): 1043.30
Середній кеш-хіт (LRU):          974.64
Середній кеш-хіт (LFU):          976.78

Покращення над LRU:              7.04%
Покращення над LFU:              6.81%
```

Рисунок 4.10. Підсумкові показники кількості кеш-хітів для трьох методів управління кешем після 50 запусків системи

Графіки та числові дані дозволяють оцінити динаміку ефективності кешування, але для більш повного аналізу доцільно порівняти ключові властивості кожного з підходів у цілому. В табл. 4.2. наведено порівняльний аналіз ефективності методів кешування основними критеріями: швидкістю роботи, складністю реалізації, здатністю до адаптації та точністю прийняття рішень у змінних умовах.

Таблиця 4.2.

Порівняльний аналіз ефективності та складності методів кешування

Метод кешування	Час прийняття рішення	Час навчання або оновлення	Складність реалізації	Адаптивність до змін	Точність прийняття рішень
LRU	Дуже швидкий (O(1))	Не потребує	Низька	Немає	Низька
LFU	Швидкий (O(log (n)), де: n – кількість об’єктів у	Не потребує	Середня	Немає	Помірна

	кеші)				
Q-Learning (табличний)	Швидкий (O(1))	Помірний	Помірна	Є	Висока
Логістична регресія	Помірний (O(m*n), де: m – кількість об’єктів в кеші, n – кількість ознак об’єкта)	Високий	Висока	Є	Висока

Таким чином, отримані результати підтверджують доцільність використання моделей машинного навчання для задач кешування у складних динамічних середовищах. Інтелектуальні підходи, зокрема табличний Q-Learning і логістична регресія, демонструють вищу адаптивність і точність прийняття рішень порівняно з існуючими методами кешування, що базуються на простих евристиками. Водночас це супроводжується деякими обмеженнями у складності та часі навчання, що вимагає подальшого балансування між ефективністю та обсягом використаних ресурсів в реальних умовах застосування.

Висновки до розділу 4

1. Описано структуру та логіку роботи системи експериментального тестування ефективності гібридного методу інтелектуального кешування, яка вирішує ключові задачі кешування. Показано взаємодію між основними компонентами: MongoDB, Redis, моделями машинного навчання та модулем керування, який координує обробку запитів та реалізовує алгоритм роботи гібридного методу інтелектуального управління кешем.

2. Описано програмну реалізацію системи експериментального тестування ефективності гібридного методу інтелектуального кешування, що вирішує основні задачі кешування з використання моделей машинного навчання – табличного Q-Learning та логістичної регресії. У розробленій архітектурі системи передбачено генерацію вхідних даних, побудову ознак, прийняття рішень, оновлення моделей в асинхронному режимі та збереження всіх необхідних даних у базі даних MongoDB та кеші Redis.

3. Проведено аналіз результатів роботи системи експериментального тестування ефективності гібридного методу інтелектуального кешування, який показав, що за умови наявності 100 об’єктів у базі даних MongoDB та розміром кешу Redis 10:

- Після 1 запуску системи з 100000 випадково згенерованими запитами покращення становить 3.01% порівняно з LRU і 3.09% порівняно з LFU;
- Після 50 запусків системи з 10000 випадково згенерованими запитами покращення становить 7.04% порівняно з LRU і 6.81% порівняно з LFU.

4. Отримані результати підтверджують здатність гібридного методу інтелектуального управління кешем адаптуватися до характеру та динаміки запитів, поступово підвищуючи ефективність кешування.

ВИСНОВКИ

1. На основі проведеного аналізу проблем, що виникають при роботі з даними у процесі будівельно-технічної експертизи та у високонавантажених інформаційних системах, встановлено доцільність використання нереляційних баз даних і технологій кешування як ефективних інструментів подолання таких викликів, як різноманітність форматів даних, відсутність єдиного джерела правди, дублювання та повільний доступ до часто запитуваної інформації.

2. З урахуванням структурних та функціональних вимог до бази даних системи підтримки процесу відновлення пошкоджених об’єктів нерухомості обґрунтовано вибір MongoDB як бази даних системи; визначено її ключові переваги – гнучка документоорієнтована модель, масштабованість, підтримка геопросторових функцій та ACID-транзакцій; спроєктовано структурну модель бази даних у вигляді чотирьох колекцій документів, яка враховує потреби користувачів з різними рівнями доступу та закладає основу для ефективної інтеграції з механізмами інтелектуального кешування.

3. У межах реалізації гібридного методу інтелектуального управління кешем у системі підтримки процесу відновлення пошкоджених об’єктів нерухомості формалізовано три ключові задачі: прийняття рішення про доцільність кешування нового об’єкта, вибір об’єкта для видалення у разі заповнення кешу та періодичну інвалідацію неактуальних об’єктів. Для вирішення першої задачі застосовано модель табличного Q-Learning, яка інтерпретує прийняття рішення як марковський процес і враховує вектор ознак об’єкта, що описує його стан у системі на момент обробки запиту. Для другої та третьої задач використано модель логістичної регресії як бінарного класифікатора, що дозволяє оцінювати ймовірність повторного використання об’єкта на момент видалення із заповненого кешу або його інвалідації. Визначено набори суттєвих ознак, механізми асинхронної генерації навчальних прикладів та періодичного оновлення параметрів логістичної регресії.

Спроєктовано колекції документів «Q-таблиця», «Навчальні приклади» та «Параметри моделі», що забезпечують збереження даних для машинного навчання. Описано повний алгоритм роботи гібридного методу, який координує взаємодію моделей машинного навчання, бази даних MongoDB та кешу Redis. Обґрунтовано доцільність використання Redis як технології кешування з огляду на її продуктивність та масштабованість. Показано, що інтеграція модуля інтелектуального кешування, що реалізовує гібридний метод інтелектуального управління кешем у загальну модель системи є ключовим чинником підвищення ефективності роботи з даними.

5. Розроблено систему експериментального тестування ефективності гібридного методу інтелектуального кешування і проведено аналіз результатів роботи цієї системи, який показав, що за умови наявності 100 об’єктів у базі даних MongoDB та розміром кешу Redis 10:

- Після 1 запуску системи з 100000 випадково згенерованими запитами покращення становить 3.01% порівняно з LRU і 3.09% порівняно з LFU;
- Після 50 запусків системи з 10000 випадково згенерованими запитами покращення становить 7.04% порівняно з LRU і 6.81% порівняно з LFU.

6. Отримані результати підтверджують здатність гібридного методу інтелектуального управління кешем адаптуватися до характеру та динаміки запитів, поступово підвищуючи ефективність кешування. Таким чином, реалізована система експериментального тестування не лише підтверджує працездатність запропонованого методу, а й демонструє його переваги над існуючими методами кешування у динамічних умовах роботи за обмежених розмірів вхідних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pasko R., Panko O., Terenchuk S. DIGITAL TECHNOLOGIES INTO PROCESS INSPECTION OF BUILDINGS, PROPERTY AND INFRASTRUCTURE OBJECTS INTRODUCING. *Management of Development of Complex Systems*. 2022. № 49. С. 74–80. URL: https://doi.org/10.32347/2412-9933.2022.49.74-80

2. KSE, Київська школа економіки, 2023, URL: https://kse.ua/ua/about-the-school/news/ponad-54-mlrd-zbitki-zhitlovogo-fondu-ukrayini-vnaslidok-povnomasshtabnoyi-viyni-na-kinets-travnya-2023-roku/

3. Проблеми будівельної галузі України, 2023, URL: https://dbn.co.ua/publ/problemy_budivelnoji_galuzi_ukrajini/2-1-0-975

4. Методика обстеження будівель та споруд, пошкоджених внаслідок надзвичайних ситуацій, бойових дій та терористичних актів: Затверджено наказом Міністерства розвитку громад та територій України від 28 квітня 2022 року № 65 (скасовано). – Київ: Міністерство розвитку громад та територій України, 2022

5. Інструкція про призначення та проведення судових експертиз та експертних досліджень, затверджена наказом Міністерства юстиції України 08.10.1998 №53/5, із змінами, внесеними згідно з наказами Міністерства юстиції №144/5 від 30.12.2004, №126/5 від 29.12.2006, №1198/5 від 15.07.2008, №965/5 від 01.06.2009, №1950/5 від 26.12.2012 [Електронний ресурс]. URL: https://zakon.rada.gov.ua/laws/show/z0705-98#Text.

6. Про затвердження: Змін до Інструкції про призначення та проведення судових експертиз та експертних досліджень. Міністерства юстиції України від 22.02.2019 р. №563/5. – [Електронний ресурс]. URL: http://search.ligazakon.ua/l_doc2.nsf/link1/RE33158.html.

7. Loza T. V. Problems of the organization of forensic expert activity in Ukraine and ways to solve them. *Analytical and Comparative Jurisprudence*. 2023. No. 6. P. 633–641. URL: https://doi.org/10.24144/2788-6018.2023.06.112

8. Data Life UA. Архітектура даних: короткий огляд. URL: https://data-life-ua.com/uncategorized/архітектура-даних-короткий-огляд

9. PRAVOKATOR. Особливості проведення судових експертиз для відшкодування збитків, завданих внаслідок бойових дій. URL: https://pravokator.club/news/osoblyvosti-provedennya-sudovyh-ekpertyh-dlya-vidshkoduvannya-zbytkiv-zavdanyh-vnaslidok-bojovyh-dij

10. Martynenko N. Digitalisation of forensic expert activity in Ukraine: Organisational and legal framework. *Forensic Science International: Synergy*. 2025. Vol. 10. P. 100578. URL: https://doi.org/10.1016/j.fsisyn.2025.100578

11. Босенко І. В. Моделі і методи штучного інтелекту в процесі виконання будівельно-технічної експертизи. Управління розвитком складних систем. Київ, 2025. № 61. С. 180 – 186, dx.doi.org\10.32347/2412-9933.2025.61.180-186

12. SGS. Increase Throughput, Reduce Error and Stabilize Circuits with Expert Systems. SGS White Paper. July 2012. URL: https://www.sgs.com/-/media/sgscorp/documents/corporate/brochures/sgs-min-wp-increase-throughput-reduce-error-and-stabilize-circuits-with-expert-systems-en-12.cdn.en-CA.pdf

13. Bailis P., Gan E., Rong K., Suri S. *Prioritizing Attention in Fast Data: Principles and Promise*. 8th Biennial Conference on Innovative Data Systems Research (CIDR ’17), 2017. URL: https://dawnd9.sites.stanford.edu/publications/macrobase/prioritizing-attention-fast-data-principles-and-promise

14. LexisNexis. *The Cost of Low-Quality Data for Decision Intelligence*. LexisNexis Industry Insights. 2023. URL: https://www.lexisnexis.com/community/insights/professional/b/industry-insights/posts/the-cost-of-low-quality-data-for-decision-intelligence

15. Metaplane. *Stale Data Leads to Bad Business Decisions*. Metaplane Blog. 23.05.2023. URL: https://www.metaplane.dev/blog/stale-data-leads-to-bad-business-decisions

16. Intersystems. *NoSQL Databases Explained: Advantages, Types, and Use Cases*. Intersystems. URL: https://www.intersystems.com/resources/nosql-databases-explained-advantages-types-and-use-cases/

17. Integrate.io. *SQL vs NoSQL: 5 Critical Differences*. Integrate.io. URL: https://www.integrate.io/blog/the-sql-vs-nosql-difference

18. MongoDB. *Why NoSQL? Learn about the Benefits and Best Use Cases*. MongoDB. URL: https://www.mongodb.com/nosql-explained

19. MyCounter. *Що таке кеш? Все, що потрібно знати про кешування інформації.* URL: https://mycounter.com.ua/scho-take-kesh-vse-scho-potribno-znaty-pro-keshuvannya-informatsii/

20. Prisma. *Database caching: Overview, types, strategies and their benefits.* URL: https://www.prisma.io/dataguide/managing-databases/introduction-database-caching

21. HOSTiQ.ua. *Кеш (cache): що це таке і навіщо потрібно.* URL: https://hostiq.ua/wiki/ukr/cache/

22. Hazelcast. *What Is Caching? How Caching Works and Its Limitations.* URL: https://hazelcast.com/foundations/caching/caching/

23. Akash Rajpurohit Blog. *Caching Strategies: Understand Write-Through, Write-Behind, Read-Through, and Cache-Aside.* URL: https://akashrajpurohit.com/blog/caching-strategies-understand-writethrough-writebehind-readthrough-and-cacheaside

24. S. Terenchuk, R. Pasko, A. Buhrov, V. Ploskyi, O. Panko and V. Zapryvoda, "Computerization of the process of reconstruction of damaged or destroyed real estate," 2022 IEEE 3rd KhPI Week on Advanced Technology (KhPIWeek), Kharkiv, Ukraine, 2022, pp. 1-6, https://doi.org/10.1109/KhPIWeek57572.2022.9916470

25. Modeling the Process of Assessing the Technical Condition of Damaged Real Estate Objects / B. Volokh et al. 2023 IEEE International Conference on Smart Information Systems and Technologies (SIST), Astana, Kazakhstan, 4–6 May 2023. 2023. URL: https://doi.org/10.1109/sist58284.2023.10223547

26. Бугров А. А., Волох Б. Ю., Босенко І. В., Теренчук С. А. Система підтримки процесу відновлення об’єктів нерухомості: обробка і збереження даних. Управління розвитком складних систем. Київ, 2024. № 60. С. 136 – 145, dx.doi.org\10.32347/2412-9933.2024.60.136-145.

27. W3Schools. SQL JOIN. URL: https://www.w3schools.com/sql/sql_join.asp

28. Amazon Web Services. Amazon Simple Storage Service (S3). Офіційна документація. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

29. Amazon Web Services. What is Database Sharding? URL: https://aws.amazon.com/what-is/database-sharding/

30. TechTarget. What is Database Replication and How Does it Work? URL: https://www.techtarget.com/searchdatamanagement/definition/database-replication

31. Microsoft Learn. Failover modes for availability groups - SQL Server Always On. URL: https://learn.microsoft.com/en-us/sql/database-engine/availability-groups/windows/failover-and-failover-modes-always-on-availability-groups

32. Digma AI. How Indexing Enhances Query Performance. URL: https://digma.ai/how-indexing-enhances-query-performance/

33. Databricks. ACID Transactions in Databases. URL: https://www.databricks.com/glossary/acid-transactions

34. Пошкоджене майно. Портал Дія. URL: https://diia.gov.ua/services/poshkodzhene-majno

35. Галінський О.М., Молодід О.С., Шарикіна Н.В., Плохута Р.О. Дослідження технологій відновлення бетонного захисного шару залізобетонних конструкцій при реконструкції будівель і споруд. *IOP Conference Series: Materials Science and Engineering.* 2020. Вип. 907. С. 012056. URL: https://doi.org/10.1088/1757-899X/907/1/012056

36. The CAP Theorem in DBMS. GeeksforGeeks. URL: https://www.geeksforgeeks.org/the-cap-theorem-in-dbms/

37. Гілберт С., Лінч Н.А. Перспективи теореми CAP. *Computer*. 2012. Т. 45. № 2. С. 30–36. URL: https://doi.org/10.1109/MC.2011.389

38. Стецик О.А., Теренчук С.А. Порівняльний аналіз архітектур нереляційних баз даних. *Управління розвитком складних систем*. 2021. № 47. С. 78–82. URL: https://doi.org/10.32347/2412-9933.2021.47.78-82

39. Strategies for improving database performance in high-traffic environments – New Relic. URL: https://newrelic.com/blog/how-to-relic/strategies-for-improving-database-performance-in-high-traffic-environments

40. Database Scalability: Horizontal & Vertical Scaling Explained. Couchbase. URL: https://www.couchbase.com/resources/concepts/database-scalability/

41. MongoDB. Geospatial Queries. URL: https://www.mongodb.com/docs/manual/geospatial-queries/

42. Тудоріка Б.-Д., Букур К. Порівняння кількох баз даних NoSQL з коментарями та примітками. *Міжнародна конференція RoEduNet: Мережі в освіті та дослідженнях*. 2011. Вип. 10. С. 1–5. URL: https://doi.org/10.1109/RoEduNet.2011.5993686

43. SQL vs NoSQL: 5 Critical Differences. Integrate.io. URL: https://www.integrate.io/blog/the-sql-vs-nosql-difference/#:~:text=SQL%20databases%20are%20vertically%20scalable,data%20like%20documents%20or%20JSON.

44. Дьорьоді К., Дьорьоді Р., Печерле Г., Олах А. Порівняльне дослідження: MongoDB проти MySQL. *13-та Міжнародна конференція з інженерії сучасних електричних систем (EMES)*. 2015. С. 1–5. URL: https://www.researchgate.net/publication/278302676_A_Comparative_Study_MongoDB_vs_MySQL

45. Мішра О., Лодхі П., Мехта Ш. Документально-орієнтовані NoSQL бази даних: емпіричне дослідження. *Data Science and Analytics*. Сінгапур: Springer Singapore, 2018. Вип. 799. С. 126–136. URL: https://doi.org/10.1007/978-981-10-8527-7_12

46. Араужу Ж. М. А., де Моура А. К. Е., да Сілва С. Л. Б., Холанда М., Рібейро Е. Д. О., да Сілва Г. Л. Порівняльний аналіз продуктивності баз даних NoSQL Cassandra та MongoDB. *IEEE 16-та Іберійська конференція з інформаційних систем і технологій (CISTI)*. Шавеш, Португалія, 23–26 червня 2021 р. С. 1–6. URL: https://doi.org/10.23919/CISTI52073.2021.9476319

47. Що таке MongoDB – робота та можливості. GeeksforGeeks. URL: https://www.geeksforgeeks.org/what-is-mongodb-working-and-features/

48. Хосе Б., Абрахам С. Аналіз продуктивності NoSQL та реляційних баз даних за допомогою MongoDB та MySQL. *Materials Today: Proceedings*. 2020. Т. 24. С. 2036–2043. URL: https://doi.org/10.1016/j.matpr.2020.03.634

49. Why Use MongoDB? MongoDB. URL: https://www.mongodb.com/resources/products/fundamentals/why-use-mongodb

50. Index Types. MongoDB Documentation. URL: https://www.mongodb.com/docs/manual/core/indexes/index-types/

51. Sharding. MongoDB Documentation. URL: https://www.mongodb.com/docs/manual/sharding/

52. Replication. MongoDB Documentation. URL: https://www.mongodb.com/docs/manual/replication/

53. Replica Set Secondary Members. MongoDB Documentation. URL: https://www.mongodb.com/docs/manual/core/replica-set-secondary/

54. JSON and BSON. MongoDB Documentation. URL: https://www.mongodb.com/resources/basics/json-and-bson

55. Performance Best Practices: Hardware and OS Configuration. MongoDB Blog. URL: https://www.mongodb.com/blog/post/performance-best-practices-hardware-and-os-configuration

56. Concurrency and Locking. MongoDB Documentation. URL: https://www.mongodb.com/docs/manual/faq/concurrency/

57. Multi-Document ACID Transactions on MongoDB. MongoDB Documentation. URL: https://www.mongodb.com/resources/products/capabilities/mongodb-multi-document-acid-transactions

58. ACID Properties in DBMS. GeeksforGeeks. URL: https://www.geeksforgeeks.org/acid-properties-in-dbms/

59. Transactions. MongoDB Documentation. URL: https://www.mongodb.com/docs/manual/core/transactions/

60. 2dsphere Indexes. MongoDB Documentation. URL: https://www.mongodb.com/docs/manual/core/indexes/index-types/geospatial/2dsphere/

61. Geospatial Queries. MongoDB Documentation. URL: https://www.mongodb.com/docs/manual/geospatial-queries/

62. Cloud Providers. MongoDB Atlas Documentation. URL: https://www.mongodb.com/docs/atlas/reference/cloud-providers/

63. Configure Security Features for Clusters. MongoDB Atlas Documentation. URL: https://www.mongodb.com/docs/atlas/setup-cluster-security/

64. Li X., Wang Y., Hu Y. A Distributed Storage and Access Approach for Massive Remote Sensing Data Based on MongoDB. *ISPRS International Journal of Geo-Information*. 2019. Vol. 8(12). P. 533. URL: https://www.mdpi.com/2220-9964/8/12/533

65. Drones4Safety. Implementation of the platform with data flow and visualization modules. 2022. URL: https://drones4safety.eu/wp-content/uploads/2022/06/D6.3-Implementation-of-the-platform-with-data-flow-and-visualization-modules.pdf

66. Data Modeling Introduction. MongoDB Documentation. URL: https://www.mongodb.com/docs/manual/data-modeling/

67. Li C., Yang W. The distributed storage strategy research of remote sensing image based on MongoDB. *Proceedings of the 2014 3rd International Workshop on Earth Observation and Remote Sensing Applications (EORSA)*, Changsha, China, 11–14 June 2014. IEEE. P. 101–104. URL: https://doi.org/10.1109/EORSA.2014.6927858

68. Волох Б. Ю. Інтелектуальне кешування у високонавантажених системах як відповідь на обмеження класичних методів керування кешем; *Шляхи підвищення ефективності будівництва. 2023.*

69. CacheFly. *The Power of Machine Learning for Advanced CDN Caching Strategies*. 2022, URL: https://www.cachefly.com/news/the-power-of-machine-learning-for-advanced-cdn-caching-strategies

70. Distributed Caching. Redis. URL: https://redis.io/glossary/distributed-caching/

71. Design Distributed Cache | System Design. GeeksforGeeks. URL: https://www.geeksforgeeks.org/design-distributed-cache-system-design/

72. Detailed strategies for mastering distributed caching in design. DesignGurus.io. URL: https://www.designgurus.io/answers/detail/detailed-strategies-for-mastering-distributed-caching-in-design

73. Cache Eviction Strategies Every Redis Developer Should Know. Redis. URL: https://redis.io/blog/cache-eviction-strategies/

74. Wikipedia contributors. Cache replacement policies. Wikipedia. URL: https://en.wikipedia.org/wiki/Cache_replacement_policies

75. GeeksforGeeks. Cache Eviction Policies | System Design. URL: https://www.geeksforgeeks.org/cache-eviction-policies-system-design/

76. Megiddo N., Modha D. *ARC: A Self-Tuning, Low Overhead Replacement Cache*, USENIX Conference on File and Storage Technologies (FAST), 2003. URL: https://www.usenix.org/legacy/event/fast03/tech/full_papers/megiddo/megiddo.pdf

77. Rotem-Gal-Oz A. *Fallacies of Distributed Computing Explained*, 2006. URL: https://arnon.me/wp-content/uploads/Files/fallacies.pdf

78. Swain D., Paikaray B., Swain D. AWRP: Adaptive Weight Ranking Policy for Improving Cache Performance. *Journal of Computing*. 2011. Vol. 3(2). URL: https://doi.org/10.48550/arXiv.1107.4851

79. Scully T., Jiang S., Zhang X. Learning-based Cache Replacement for High Performance and Dynamic Workloads. *Proceedings of the 2020 IEEE International Conference on Big Data*. 2020. P. 4042–4051. URL: https://doi.org/10.1109/BigData50022.2020.9378040

80. M Liu E., Hashemi M., Swersky K., Ranganathan P., Ahn J. An Imitation Learning Approach for Cache Replacement, *International Conference on Machine Learning (ICML)*, 2020. URL: https://doi.org/10.48550/arXiv.2006.16239

81. Wikipedia contributors. Markov decision process. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Markov_decision_process

82. Wikipedia contributors. Least Recently Used. Wikipedia. URL: https://en.wikipedia.org/wiki/Least_recently_used

83. Wikipedia contributors. Least Frequently Used. Wikipedia. URL: https://en.wikipedia.org/wiki/Least_frequently_used

84. Jain A., Lin C. An Imitation Learning Approach for Cache Replacement. *arXiv preprint* arXiv:2006.16239. 2020. URL: https://arxiv.org/pdf/2006.16239

85. Li Y. Deep Reinforcement Learning: An Overview. *arXiv preprint* arXiv:1701.07274. 2017. URL: https://arxiv.org/abs/1701.07274

86. Ghasemi M., Ebrahimi D. Introduction to Reinforcement Learning. *arXiv preprint* arXiv:2408.07712. 2024. URL: https://arxiv.org/abs/2408.07712

87. Bellman Equation in Reinforcement Learning. GeeksforGeeks. URL: https://www.geeksforgeeks.org/bellman-equation-in-reinforcement-learning/

88. Epsilon-Greedy Algorithm in Reinforcement Learning. GeeksforGeeks. URL: https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/

89. Binary classification. Wikipedia. URL: https://en.wikipedia.org/wiki/Binary_classification

90. Logistic regression - Wikipedia. Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Logistic_regression

91. Biau G., Scornet E. A Random Forest Guided Tour. *TEST*. 2016. Vol. 25(2). P. 197–227. URL: https://doi.org/10.1007/s11749-016-0481-7

92. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. 2016. P. 785–794. URL: https://doi.org/10.1145/2939672.2939785

93. Geng Y., Li Q., Yang G., Qiu W. Logistic Regression. In: Practical Machine Learning Illustrated with KNIME. Singapore: Springer Nature, 2024. P. 99–132. URL: https://doi.org/10.1007/978-981-97-3954-7_4

94. Baeldung. Gradient Descent Equation in Logistic Regression. 2025. URL: https://www.baeldung.com/cs/gradient-descent-logistic-regression

95. Jurafsky D., Martin J. H. *Speech and Language Processing*. 3rd ed. Draft. Stanford University. 2023. Chapter 5: Logistic Regression. URL: https://web.stanford.edu/~jurafsky/slp3/5.pdf

96. Memcached vs Redis: Choose Your In-Memory Cache. Kinsta. URL: https://kinsta.com/blog/memcached-vs-redis/

97. What is a Distributed Cache? Hazelcast. URL: https://hazelcast.com/foundations/caching/distributed-cache/

98. Redis. Official Documentation. URL: https://redis.io/docs/latest/

99. Memcached. Official Documentation. URL: https://docs.memcached.org/

100. Hazelcast. What is Hazelcast. Hazelcast Documentation. URL: https://docs.hazelcast.com/hazelcast/latest/what-is-hazelcast

101. Redis. Cluster Specification. Redis Documentation. URL: https://redis.io/docs/latest/operate/oss_and_stack/reference/cluster-spec/

102. Redis. Data Types. Redis Documentation. URL: https://redis.io/docs/latest/develop/data-types/

103. Redis. Eviction Policy. Redis Documentation. URL: https://redis.io/docs/latest/develop/reference/eviction/

104. Redis. Redis Data Integration. Redis Documentation. URL: https://redis.io/docs/latest/integrate/redis-data-integration/

105. Scale with Redis Cluster. Redis Documentation. URL: https://redis.io/docs/latest/operate/oss_and_stack/management/scaling/

106. Redis replication. Redis Documentation. URL: https://redis.io/docs/latest/operate/oss_and_stack/management/replication/

107. Redis geospatial. Redis Documentation. URL: https://redis.io/docs/latest/develop/data-types/geospatial/

108. Yehorov O., Volokh B., Bosenko I., Yeremenko B., Terenchuk S., Modeling of highly loaded distributed system supporting the process of assessing the technical condition of objects, 4th International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2024, Ternopil, Ukraine, Opole, Poland, 23-25 October 2024, Vol. 3896, P. 173 – 185. URL: https://ceur-ws.org/Vol-3896/paper10.pdf

109. Pasko, R., Klochko, A., Petrochenko, O., Ploskyi, V., **Volokh B.,** Intelligent Supporting System to Building-Technical Expertise Process, *CEUR Workshop Proceedings, 3rd International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2023,* Ternopil, Ukraine, 22-24 November 2023, Vol. 3628, P. 702 – 708. URL: https://ceur-ws.org/Vol-3628/short9.pdf

110. Why Is The Python Programming Language So Popular? Quantum Zeitgeist. URL: https://quantumzeitgeist.com/why-is-the-python-programming-language-so-popular/

111. Python Language Advantages and Applications. GeeksforGeeks. URL: https://www.geeksforgeeks.org/python-language-advantages-applications/

112. What's New In Python 3.9. Python Documentation. URL: https://docs.python.org/3/whatsnew/3.9.html

113. What is an IDE? Integrated Development Environment Explained. Amazon Web Services. URL: https://aws.amazon.com/what-is/ide/

114. JetBrains. PyCharm: the Python IDE for data science and web development. JetBrains. URL: https://www.jetbrains.com/pycharm/

115. MongoDB Python | Insert and Update Data. GeeksforGeeks. URL: https://www.geeksforgeeks.org/mongodb-python-insert-update-data/

116. What is MongoDB Compass? MongoDB Documentation. URL: https://www.mongodb.com/docs/compass/current/

117. MongoDB Compass – Explained in Detail. The Knowledge Academy. URL: https://www.theknowledgeacademy.com/blog/mongodb-compass/

118. Redis-py – Python Client for Redis. Redis Documentation. URL: https://redis.io/docs/latest/develop/clients/redis-py/

119. Redis Commands – redis-py dev documentation. URL: https://redis-py.readthedocs.io/en/stable/commands.html

120. JSON Docs - Redis. Redis Documentation. URL: https://redis.io/docs/latest/develop/data-types/json/

121. Another Redis Desktop Manager – A faster, better and more stable Redis desktop manager. GitHub. URL: https://github.com/qishibo/AnotherRedisDesktopManager

ДОДАТКИ

Додаток А

Переваги MongoDB та пропозиції їх застосування в СППВПОН

Перевага	Опис	Застосування
Індексація	Створення додаткових структур для прискорення виконання запитів.	Створення індексів для ключових полів. Використання комбінованих індекси для складних запитів.
Масштабованість	Розподіл даних між кількома серверами (шардинг).	Розподіл даних за регіонами (шард-ключ: місце розташування). Додавання серверів при збільшенні обсягу даних.
Реплікація	Резервування даних для підвищення стійкості.	Резервні копії даних для запобігання втратам. Розподілене читання для зменшення навантаження.
Документоорієнтована модель	Гнучке зберігання даних без фіксованої схеми.	Зберігання різнорідних даних в одній колекції. Легке додавання нових полів при появі нових типів пошкоджень.
Багатопоточність	Підтримка одночасного виконання кількох запитів.	Одночасна робота кількох спеціалістів. Паралельне завантаження даних із дронів чи супутників.
ACID-транзакції	Гарантія узгодженості даних навіть у складних операціях.	Одночасне оновлення бази даних і кешу для забезпечення узгодженості. Оновлення кількох пов’язаних документів у межах однієї транзакції.
Геопросторові функції	Запити та аналіз даних із географічними координатами.	Швидкий пошук будівель у певному регіоні. Візуалізація пошкоджень на карті.
Інтеграція з хмарними платформами	Легка інтеграція з хмарними сервісами (AWS, Google Cloud).	Зберігання великих файлів (зображень, відео) у Amazon S3. Аналіз супутникових знімків за допомогою хмарних обчислень.

Додаток Б

Структура документа в колекції «Пошкоджені будівлі»

Поле	Опис	Тип даних (укр / англ)
ID	Обов'язковий унікальний ідентифікатор	Рядок / ObjectId
Розташування	Координати, місто, вулиця	Об'єкт / Object (GeoJSON + String)
Тип	Тип об’єкта (лікарня, школа, житловий будинок тощо)	Рядок / String
Дата пошкодження	Дата виникнення шкоди	Дата / Date
Медіафайли	Шляхи до медіафайлів, що описують пошкодження	Масив рядків / Array of Strings
Джерело інформації	Джерело даних про пошкодження	Рядок / String
Ім'я контакту	Ім’я особи, яка надала інформацію	Рядок / String
Контактна роль	Роль контактної особи	Рядок / String
Контактна адреса електронної пошти	Електронна адреса контактної особи	Рядок / String
Контактний телефон	Номер телефону контактної особи	Рядок / String
Пошкодження	Опис пошкоджень	Рядок / String
Ступінь тяжкості пошкоджень	Класифікація: немає, несуттєві, суттєві, критичні	Рядок (перелік) / Enum (String)
Причини пошкоджень	Причини (напр., бомбардування, пожежа тощо)	Рядок / String
Технічний стан	Оцінка стану конструктивних елементів	Рядок / String
Оцінений	Чи технічний стан вже оцінений	Булеве / Boolean

Додаток В

Основні існуючі методи кешування [74], [75]

Назва методу	Принцип роботи	Переваги	Недоліки
LRU (Least Recently Used)	Видаляє найдавніше використаний об’єкт.	Простий у реалізації; добре працює при локальності посилань.	Не враховує частоту звернень; не адаптується до змін.
LFU (Least Frequently Used)	Видаляє об’єкт із найменшою частотою звернень.	Враховує історію використання; корисний для стабільного потоку запитів.	Утримує «застарілі» популярні об’єкти; складна реалізація при великій кількості об’єктів.
FIFO (First In First Out)	Видаляє об’єкти у порядку їх надходження.	Надзвичайно простий у реалізації.	Ігнорує корисність та частоту; може витіснити важливі об’єкти.
MRU (Most Recently Used)	Видаляє об’єкт, який використовувався останнім.	Ефективний при одноразових запитах.	У більшості випадків менш ефективний, ніж LRU.
Random Replacement	Видаляє випадковий об’єкт незалежно від частоти чи часу.	Дуже проста реалізація; непогано працює у хаотичних запитах.	Нестабільна ефективність; можлива втрата важливих об’єктів.
2Q (дві черги)	Використовує дві черги для об’єктів з одноразовим і повторним доступом.	Зменшує «забруднення» кешу; адаптується до короткострокових/довгострокових шаблонів.	Ускладнена реалізація.
ARC (Adaptive Replacement Cache)	Динамічно балансує між LRU та LFU на основі поточного навантаження.	Адаптивність до зміни шаблонів; не потребує налаштування.	Висока складність реалізації.
TTL (Time-To-Live)	Кожному об’єкту призначається термін життя; після завершення об’єкт видаляється автоматично.	Добрий для кешування тимчасових даних; простий у налаштуванні.	Не враховує реальну потребу в об’єкті; можливі передчасні видалення

Додаток Г

Результати порівняльного аналізу основних технологій кешування

Критерій	Redis	Memcached	Hazelcast
Тип даних, що підтримуються	Складні структури (списки, множини, хеші)	Прості пари ‘ключ-значення’	Складні структури, підтримка транзакцій
Масштабованість	Висока (кластеризація)	Помірна (без кластеризації)	Висока (розподілена архітектура)
Стійкість до збоїв	Так (реплікація)	Ні	Так
Продуктивність	Дуже висока	Висока	Висока
Дані зберігаються після перезапуску сервісу	Так	Ні	Так (з інтеграцією)
Можливість автоматичного видалення застарілих даних	Так	Ні	Так
Простота використання	Відносно проста	Дуже проста	Складніше, ніж Redis або Memcached
Ефективність використання пам’яті	Висока	Висока	Помірна
Сумісність із розподіленими системами	Відмінна	Обмежена	Відмінна
Типи сценаріїв використання	Сесії, транзакції, черги, аналітика	Сесії, кеш запитів, тимчасові дані	Сесії, транзакції, розподілені обчислення

Додаток Ґ

ДОВІДКА ПРО ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОГО ДОСЛІДЖЕННЯ В НАВЧАЛЬНИЙ ПРОЦЕС

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

ФАКУЛЬТЕТ АВТОМАТИЗАЦІЇ І ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

«23» травня 2025 р.

ДОВІДКА
про впровадження результатів дослідження
на здобуття наукового ступеня доктора філософії
зі спеціальності 126 Інформаційні системи та технології

Запропонований в дисертаційній роботі Волоха Богдана Юрійовича гібридний метод інтелектуального управління кешем та структура модель документів для нереляційної бази даних MongoDB є прикладом інноваційного проекту, що навчає студентів працювати з передовими технологіями нереляційних баз даних.

Використані в роботі алгоритми машинного навчання впроваджені в курс «Інформаційні технології представлення обробки та розпізнавання зображень» освітньої програми спеціальності 126 – «Інформаційні системи і технології», курс «Теорія алгоритмів» освітньої програми спеціальностей 126 – «Інформаційні системи і технології» і 126 – «Інформаційні системи і технології. Штучний Інтелект. Когнітивні технології», курс «Прикладна теорія графів» освітньої програми спеціальності 174 – «Автоматизація та комп'ютерно-інтегровані технології» факультету автоматизації і інформаційних технологій Київського національного університету будівництва і архітектури в 2022-2025 роках.

Це надає можливість впроваджувати необхідні теоретичні знання та практичні навички фахівцям галузі інформаційних технологій при вирішенні задач цифрової трансформації і підтверджує актуальність і вагоме теоретичне і практичне значення у процесі підготовки майбутніх спеціалістів галузі.

декан факультету АІТ,
доктор технічних наук, професор Олександр ТЕРЕНТЬЄВ

Додаток Е

ДОВІДКИ
ПРО АПРОБАЦІЇ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОГО ДОСЛІДЖЕННЯ

МІНІСТЕРСТВО ЮСТИЦІЇ УКРАЇНИ
КИЇВСЬКИЙ НАУКОВО-ДОСЛІДНИЙ ІНСТИТУТ СУДОВИХ ЕКСПЕРТИЗ МІНІСТЕРСТВА ЮСТИЦІЇ УКРАЇНИ (КНДІСЕ)
вул. Сім'ї Бродських, 6, м. Київ, 03057
тел. (044) 200-29-11, (044) 200-29-29
e-mail: info@kndise.gov.ua
https://kndise.gov.ua
код згідно з ЄДРПОУ 02883096

Паперова копія оригіналу електронного документа

Ректору
Київського національного університету будівництва і архітектури

Олексію ДНІПРОВУ
просп. Повітряних сил, 31, м. Київ, 03037

_____ № _____
На № 1/02/796/25 від 08.06.2025

Про направлення довідки про апробацію

09.06.2025 до Київського науково-дослідного інституту судових експертиз Міністерства юстиції України надійшов запит на апробацію аналітичних матеріалів дисертаційного дослідження Волоха Богдана Юрійовича за темою «Інтелектуальне кешування даних в системі підтримки процесу відновлення пошкоджених об'єктів нерухомості».

За результатами проведеної апробації в умовах Лабораторії інженерно-технічних видів досліджень Київського науково-дослідного інституту судових експертиз Міністерства юстиції України направляємо Довідку про апробацію результатів дисертаційної роботи.

Додаток. Довідка про апробацію результатів дисертаційної роботи на 1 арк.

Директор **Максим КИСЕЛЬОВ**

Олексій КОМАНДИРОВ 200 29 23

КНДІСЕ № 12699/10347-12-25/40 від 18.06.2025
Сертифікат 2E0C180800000000000000000000000000000001
Підписувач КИСЕЛЬОВ МАКСИМ ЄВГЕНОВИЧ
Дійсний з 04.03.2025 11:35:01 по 14.01.2027 23:59:59

Київський науково-дослідний інститут судових експертиз Міністерства юстиції України
ДЛЯ ДОКУМЕНТІВ №1

ЗГІДНО З ОРИГІНАЛОМ
ПРОВІДНИЙ ФАХІВЕЦЬ
БОЛЮК К.О.

КАНЦЕЛЯРІЯ №1
ВХ. № 875/05/25
"24" 06 25 р.

МІНІСТЕРСТВО ЮСТИЦІЇ УКРАЇНИ
КИЇВСЬКИЙ НАУКОВО-ДОСЛІДНИЙ ІНСТИТУТ СУДОВИХ ЕКСПЕРТИЗ МІНІСТЕРСТВА ЮСТИЦІЇ УКРАЇНИ (КНДІСЕ)

вул. Сім'ї Бродських, 6, м. Київ, 03057, тел. (044) 200-29-11, (044) 200-29-29, код згідно з ЄДРПОУ 02883096
номер електронного кабінету в підсистемі Електронний суд ЄСІТС **02883096**

ВІДОМОСТІ ПРО АТЕСТАЦІЮ ТА АКРЕДИТАЦІЮ
ENFSI (Європейська мережа судово-експертних установ) – сертифікат від 18.05.2017
НААУ – атестат від 29.10.2024 № 201141 (ISO-17025), атестат від 20.12.2022 №70530 (ISO-17020), атестат від 19.08.2024 №10436 (ISO -17065)
ДП «Укрметртестстандарт» – сертифікат від 18.11.2020 № ПТ-428/20
ФДМУ – сертифікат від 15.08.2022 № 414/2022

ДОВІДКА
про апробацію результатів дисертаційної роботи «Інтелектуальне кешування даних в системі підтримки процесу відновлення пошкоджених об'єктів нерухомості» Волоха Богдана Юрійовича

за спеціальністю 126 «Інформаційні системи та технології»

Лабораторією інженерно-технічних видів досліджень Київського науково-дослідного інституту судових експертиз Міністерства юстиції України засвідчується актуальність впровадження результатів дисертаційного дослідження Волоха Богдана Юрійовича «Інтелектуальне кешування даних в системі підтримки процесу відновлення пошкоджених об'єктів нерухомості».

Особливу практичну значимість має запропонована в роботі структурна модель документів для нереляційних баз даних, оптимізована для зберігання інформації про пошкоджені об'єкти нерухомості, історію запитів до них від користувачів і статистику пошкоджень, що надасть можливість зменшити навантаження на експертів і підвищити зручність та ефективність їх роботи.

Завідувач Лабораторії інженерно-технічних видів досліджень, к.т.н., Заслужений будівельник України Роман ПАСЬКО

Завідувач Відділу дослідження обсягів, якості та вартості будівельних робіт Лабораторії інженерно-технічних видів досліджень, к.т.н., Заслужений будівельник України Олексій КОМАНДИРОВ