КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (33), 2026
ISSN 2663 – 4023
DOI 10.28925/2663-4023.2026.33.1248
УДК 004.8: 004.65

Смірнов Олексій Анатолійович
д.т.н., професор, завідувач кафедри кібербезпеки та програмного забезпечення
Центральноукраїнський національний технічний університет, Кропивницький, Україна
ORCID: 0000-0001-9543-874X
dr.smirnovoa@gmail.com

Ткачук Роман Олегович
Старший системний інженер ЕПАМ Україна,
асистент кафедри кібербезпеки та програмного забезпечення
Центральноукраїнський національний технічний університет, Кропивницький, Україна
ORCID: 0000-0002-1984-0419
tkachukroman64@gmail.com

Козірова Наталія Леонідівна
викладач кафедри кібербезпеки та програмного забезпечення
Центральноукраїнський національний технічний університет, Кропивницький, Україна
ORCID: 0009-0005-8753-5132
natalidonchenko23@gmail.com

Константинова Лілія Володимирівна
викладач кафедри кібербезпеки та програмного забезпечення
Центральноукраїнський національний технічний університет, Кропивницький, Україна
ORCID: 0000-0002-3305-2427
liliyashel1976@gmail.com

Конопліцька-Слободенюк Оксана Костянтинівна
викладач кафедри кібербезпеки та програмного забезпечення
Центральноукраїнський національний технічний університет, Кропивницький, Україна
ORCID: 0000-0001-9981-5194
ksuha80@gmail.com

Якименко Наталія Миколаївна
кандидат фізико-математичних наук, доцент, доцент кафедри кібербезпеки та програмного забезпечення
Центральноукраїнський національний технічний університет, Кропивницький, Україна
ORCID: 0000-0002-4498-0093
yakimenko_n_m@ukr.net

Смірнов Сергій Анатолійович
кандидат технічних наук, доцент, доцент кафедри кібербезпеки та програмного забезпечення
Центральноукраїнський національний технічний університет, Кропивницький, Україна
ORCID: 0000-0002-7649-7442
smirnov.ser.81@gmail.com

ДОСЛІДЖЕННЯ ЗАСТОСУВАННЯ ВЕКТОРНИХ БАЗ ДАНИХ У ГЕНЕРАТИВНОМУ ШТУЧНОМУ ІНТЕЛЕКТІ

Анотація. У даній роботі проведено дослідження застосування векторних баз даних у генеративному штучному інтелекті. Метою даної статті є дослідження можливості використання векторних баз даних як фундаменту сучасної AI-інфраструктури та їхня роль у розширенні когнітивних можливостей GenAI. Об’єктом дослідження є процес застосування баз даних у штучному інтелекті. Предметом дослідження є застосування векторних баз даних у генеративному штучному інтелекті. У даному дослідженні були розв’язані наступні завдання: досліджено роль векторних сховищ у розширенні когнітивних можливостей генеративного ШІ; здійснено порівняльний аналіз традиційних реляційних систем та векторних баз даних; проаналізовано механіку та верифікаційний потенціал архітектурних парадигм RAG та RAFT. Визначені перспективи подальших досліджень, які заключаються у наступному: розробці ефективних механізмів моніторингу та автоматичної

корекції семантичного дрейфу вбудовувань в умовах постійного оновлення корпоративних знань; дослідженню адаптивних стратегій управління пам'яттю для усунення ефектів порогової продуктивності при масштабуванні векторних індексів; дослідженню можливостей об'єднання гібридних архітектур пошуку, що поєднують семантичне та повнотекстове індексування в єдиному технологічному середовищі; стандартизації метрик оцінки якості векторного пошуку в контексті конкретних вимог промислових RAG-систем, що забезпечить більш обґрунтований вибір інфраструктурних рішень для конкретних прикладних завдань.

Ключові слова: бази даних, штучний інтелект, векторні бази даних (ВБД), генеративний штучний інтелект (GenAI), великі мовні моделі (LLM), Retrieval-Augmented Generation (RAG), семантичний пошук, векторні вбудовування (embeddings), HNSW, RAFT, масштабованість, когнітивна інфраструктура, ANN, квантування.

ВСТУП

Постановка завдання дослідження. Перехід від розробки прототипів GenAI до їх промислової експлуатації виявив інфраструктурний розрив, що обмежує продуктивність сучасних AI-агентів. Практичне впровадження AI-агентів виявило низку критичних вразливостей самих сховищ: від деградації точності через семантичний дрейф ембедингів до виникнення «ресурсного обриву» при дефіциті RAM під час масштабування.

У межах даного дослідження детально розглядаються методи подолання цих бар'єрів, зокрема через аналіз ефективності квантування (PQ) та алгоритмів наближеного пошуку (ANN), які мають забезпечити стабільність системи за умов зростання обсягів даних. Важливим аспектом аналізу є систематизація техніко-економічних параметрів провідних рішень (Pinecone, Milvus, Weaviate тощо), що дозволяє визначити межі їхньої експлуатаційної придатності та вплив архітектурних особливостей бази на роботу надбудов типу RAG та RAFT.

Такий аналітичний підхід необхідний для обґрунтування переходу від сприйняття векторних баз як пасивних сховищ математичних об’єктів до їх використання як динамічної когнітивної пам’яті. Дослідження доводить, що саме вирішення інфраструктурних проблем VDB є критичною умовою для створення релевантних, автономних та економічно ефективних інтелектуальних систем, здатних стабільно функціонувати в межах реального бізнес-середовища.

Постановка проблеми. Незважаючи на високий потенціал генеративного ШІ, основною перешкодою для його впровадження в реальний сектор економіки залишається інфраструктурна неготовність традиційних методів зберігання та обробки знань. Головна суперечність полягає в тому, що сучасні LLM мають величезний інтелектуальний потенціал, але не мають надійної, актуальної та масштабованої пам’яті.

Векторні бази даних, які мають стати такою пам’яттю, стикаються з трьома критичними бар’єрами в умовах промислової експлуатації:

- Економічна та ресурсна неефективність: «ресурсний розрив» під час роботи з оперативною пам’яттю (RAM) робить підтримку великих векторних масивів занадто дорогою для бізнесу.
- Технологічна нестабільність: відсутність перевірених механізмів боротьби з семантичним дрейфом і складність синхронізації векторів із динамічними реляційними даними призводять до прихованої деградації всієї AI-системи.
- Архітектурна невизначеність: відсутність єдиної методології порівняння VDB змушує компанії обирати рішення «всліпу», що часто призводить до проблем із затримками (latency) та безпекою на пізніх етапах впровадження.

Таким чином, існує потреба в системних дослідженнях, які дозволять подолати ці виклики та трансформувати векторні сховища з пасивних «сховищ чисел» у надійну основу когнітивної пам’яті інтелектуальних агентів.

Аналіз останніх досліджень і публікацій. Сучасні дослідження переважно зосереджені на переході від локальних бібліотек (FAISS) до повноцінних керованих систем (VDBaaS), таких як Pinecone, Milvus або Weaviateоскільки за останні п’ять років спостерігається стрімкий зсув від розробки алгоритмів до створення комплексних систем управління векторними даними [1]. Наукова спільнота детально описує архітектуру RAG, у якій векторні бази даних виступають як зовнішня пам’ять для LLM,що дозволяє динамічно оновлювати базу знань та мінімізувати «галюцинації» моделей [2], а також досліджує метод RAFT, що допомагає моделям краще відфільтровувати нерелевантний контекст [2].

Однак проведений аналіз останніх досліджень і публікацій показує, що більшість публікацій фокусується на функціональності моделей, залишаючи поза увагою практичні проблеми експлуатації інфраструктури:

- Залежність від ресурсів: наявні дослідження рідко розглядають, як саме дефіцит оперативної пам'яті (RAM) впливає на швидкість пошуку та чи існує критичний поріг, після якого продуктивність системи різко падає [3].
- Цілісність і актуальність даних: недостатньо досліджено проблему семантичного дрейфу (коли зміна моделі ембедінгів робить старі вектори неточними) та складнощі синхронізації даних між традиційними базами даних і векторними сховищами.
- Відсутність критеріїв вибору: бракує чітких порівнянь спеціалізованих систем (Milvus, Qdrant) із розширеннями для SQL-баз даних (pgvector). Питання вартості володіння (TCO) та складності підтримки в реальних проєктах залишаються відкритими.

Саме необхідність заповнення цих практичних прогалин і визначила напрям цього дослідження.

Метою статті є дослідження можливості використання векторних баз даних як фундаменту сучасної AI-інфраструктури та їхня роль у розширенні когнітивних можливостей GenAI. В межах цієї мети необхідно вирішити наступні завдання:

- Дослідити роль векторних сховищ у розширенні когнітивних можливостей генеративного ШІ.
- Здійснити порівняльний аналіз традиційних реляційних систем та векторних баз даних.
- Проаналізувати механіку та верифікаційний потенціал архітектурних парадигм RAG та RAFT.

Об'єктом дослідження є процес застосування баз даних у штучному інтелекті.

Предметом дослідження є застосування векторних баз даних у генеративному штучному інтелекті.

ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

Використання векторних баз даних для розширення когнітивних можливостей генеративного ШІ. У сучасній архітектурі GenAI векторні бази даних (VDB) виконують роль зовнішньої пам'яті, що дозволяє подолати головне обмеження великих мовних моделей – статичність їхніх знань (knowledge cut-4]. Замість дорогого повторного навчання VDB дають змогу моделям отримувати доступ до динамічних наборів даних у режимі реального часу. Це перетворює ШІ із закритої системи на відкриту екосистему, здатну працювати з актуальною корпоративною та аналітичною інформацією [5, 6].

VDB – це спеціалізовані системи, що зберігають дані у вигляді точок у багатовимірному просторі. Це дозволяє перейти від класичного пошуку за ключовими словами до семантичного аналізу. Завдяки механізмам пошуку за схожістю (наприклад, косинусна схожість), система визначає об'єкти на основі їхнього змісту та контексту, а не просто збігу символів [7, 8]. Такий підхід працює як інтелектуальний фільтр: він відбирає лише релевантні фрагменти знань, запобігаючи перевантаженню контекстного вікна моделі та підвищуючи точність генерації [7, 9].

Окрім обробки тексту, векторна парадигма є незамінною у мультимодальних сценаріях:

- Пошук у медіа: ідентифікація зображень і відео за візуальними дескрипторами.
- Кібербезпека: виявлення аномалій у поведінкових патернах для запобігання шахрайству.
- Рекомендаційні системи та біометрія: персоналізація контенту та миттєве розпізнавання обличчя або голосу [6, 7].

Таким чином, VDB стають стратегічним компонентом інтелектуальної інфраструктури, забезпечуючи масштабованість і точну обробку складних даних у режимі реального часу.

Робота векторної бази даних (VBD) базується на двоетапному конвеєрі: індексації (завантаження) та пошуку (retrieval). На початковому етапі дані проходять через моделі ембедінгів, які перетворюють текст, зображення або аудіо у вектори – цифрові представлення об'єктів у багатовимірному просторі. Ці числові структури організовуються у спеціальні індекси (графові або кластерні), що дозволяє системі миттєво обробляти великі обсяги даних без необхідності повного їх перегляду [7-9].

В основі цього процесу лежить геометрична близькість. На відміну від класичних систем, які шукають прямі збіги символів, NLP-моделі вимірюють відстані між точками у латентному просторі. Наприклад, вектори для слів «мікропроцесор» і «контролер» будуть розташовані близько один до одного через їхню семантичну схожість, навіть якщо вони не мають спільних літер.

Як бачимо, координати документів A і B майже однакові, що дозволяє системі миттєво визначати їхню релевантність до одного запиту. Щоб підтримувати таку точність на промисловому рівні, бази даних використовують квантування продукту (product quantization) – методи стиснення, які зменшують використання пам'яті [8, 9].

Рис 1 Архітектура роботи векторних баз даних

Таблиця 1

Приклад перетворення даних у векторному просторі

Об’єкт (сирі дані)	Латентні ознаки (зміст)	Приклад вектора (ембедінгів)
Документ А: «Інструкція процесора»	Технології, обчислення, апаратне забезпечення	[0.12, 0.85, -0.44]
Документ В: «Опис мікроконтролера»	Технології, обчислення, електроніка	[0.10, 0.82, -0.40]
Документ С: «Рецепт яблучного пирога»	Кулінарія, їжа, десерт	[-0.91, 0.05, 0.77]

Порівняльний аналіз традиційних реляційних систем та векторних баз даних: архітектурні та алгоритмічні відмінності.

Головна відмінність між традиційними реляційними системами (RDBMS) та векторними базами (VDB) полягає в способі обробки даних. Класичні БД використовують B-Trees та Exact Match (точний збіг), що ідеально працює для чисел та рядків. Проте при спробі знайти схожі об’єкти у високих розмірностях ($d > 100$) виникає «прокляття розмірності»: точний пошук стає занадто повільним, а математична чіткість відстаней між точками втрачається [7, 9].

Векторні бази долають ці бар'єри завдяки трьом механізмам:

– Імовірнісні графи (HNSW): Замість перебору таблиць, ВБД будують ієрархічні шари для швидкої навігації між кластерами значень [7, 9].
– Алгоритми ANN (Наближені найближчі сусіди): Замість пошуку ідентичності, система шукає семантичну близькість, балансуючи між швидкістю та повнотою видачі [9, 10].
– Квантування (PQ): Стиснення мільярдів векторів для їх зберігання в оперативній пам'яті (RAM), що гарантує відгук у мілісекундах [7, 10].

Таблиця 2

Порівняльна характеристика систем

Критерій	Традиційні СКБД (SQL)	Векторні БД (VDB)
Об'єкт	Скалярні дані (текст, числа)	Ембединги (вектори)
Логіка пошуку	Точний збіг (Boolean logic)	Схожість (Cosine, L2)
Мета	Цілісність транзакцій (ACID)	Семантична релевантність
Критерій	Традиційні СКБД (SQL)	Векторні БД (VDB)

У підсумку, якщо SQL-бази залишаються еталоном для транзакцій (банкінг, логістика), то ВБД є інфраструктурною необхідністю для когнітивних систем. Традиційна база знайде «ноутбук» лише за словом, тоді як векторна – зрозуміє запит через координату в латентному просторі, автоматично залучивши до відповіді «лептопи» та «портативні ПК» на основі їхнього змісту [4-6].

Архітектурна парадигма Retrieval-Augmented Generation (RAG): Механіка та верифікація даних. Ключовою технологічною проблемою великих мовних моделей (LLM) залишається ризик генерації фактично недостовірного контенту (галюцинацій), що зумовлено обмеженнями етапу попереднього навчання [11, 12]. На відміну від традиційного підходу Zero-shot prompting, архітектурна парадигма Retrieval-Augmented Generation (RAG) вирішує цю проблему шляхом інтеграції векторних баз даних (VDB) як динамічного джерела істини (Ground Truth), що радикально підвищує фактичну точність відповідей [11, 13].

Функціонування системи RAG базується на конвеєрі, який починається з попередньої обробки корпусу знань: гетерогенні тексти сегментуються, перетворюються на дескриптори за допомогою моделей ембедінгів і зберігаються у VDB [11, 14]. Під час обробки користувацького запиту система виконує його векторизацію та знаходить релевантні фрагменти інформації за допомогою метрик подібності (наприклад, косинусної відстані) [13, 14]. Отримані дані інтегруються у вхідне вікно LLM (етап аугментації), створюючи контекст, який спрямовує процес логічного висновування моделі [11]. На етапі генерації модель синтезує відповідь, надаючи пріоритет зовнішньому контексту, що мінімізує галюцинації та запобігає апроксимації знань із застарілих внутрішніх параметрів [11, 12].

Реалізація RAG забезпечує стратегічні переваги: динамічне оновлення бази знань у реальному часі без необхідності перенавчання моделі та забезпечення конфіденційності (можливість роботи з закритими корпоративними архівами) [13, 15]. Ефективність системи безпосередньо залежить від якості векторних представлень і характеристик обраного сховища (Pinecone, Milvus, Weaviate) [14, 15].

Традиційне використання LLM без додаткового контексту або контрольованого донавчання має обмеження: моделі залишаються вразливими до появи нових даних, які не входили до навчальної вибірки [12]. Підхід RAG дозволяє системі працювати з актуальною інформацією, знаходячи її у зовнішньому джерелі безпосередньо в момент запиту [11, 13].

Розвиток методів контекстного навчання призвів до появи гібридних стратегій, зокрема підходу RAFT (Retrieval-Augmented Fine-Tuning). Цей метод спрямований на подолання вразливості класичного RAG до нерелевантного контексту або помилок під час пошуку [16]. На відміну від стандартного fine-tuning, у RAFT модель навчається на наборах даних, що містять запитання, релевантні документи («oracle») та документи-відволікання («distractors») [16]. Це змушує модель розвивати навичку відокремлення корисної інформації від шуму та формувати розгорнуті ланцюжки міркувань (Chain-of-Thought), спираючись на першоджерела [16].

Інтеграція RAFT в архітектуру векторних систем підвищує стійкість до маніпуляцій контекстом і ефективність у вузькоспеціалізованих галузях (медицина, право), де помилки є критичними [16]. RAFT виступає як надбудова над RAG, перетворюючи модель на активного аналітика контексту.

Підсумовуючи, RAG усуває критичний розрив між даними та інтелектом. Без цієї архітектури векторна база даних функціонує лише як пасивне сховище. Технологія RAG перетворює статичний пошук на активний когнітивний процес, виступаючи «технологічним мостом», що трансформує набір векторів у VDB у перевірені, релевантні та зрозумілі знання [11, 13].

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Попри стрімкий розвиток векторних баз даних як фундаменту для GenAI-рішень, процес їх впровадження у промислову експлуатацію (production) супроводжується низкою критичних викликів. Як показує аналіз практичного застосування, реальна продуктивність систем часто дивергує з показниками, заявленими у технічних специфікаціях. Це зумовлено не лише математичною складністю алгоритмів наближеного пошуку (ANN), а й інфраструктурними обмеженнями сучасних обчислювальних середовищ.

Основна проблематика роботи з векторними базами даних.

Проблема «пам’яті»: Чому векторні бази не вміють економити RAM. Складність оперування векторними даними полягає в низькому коефіцієнті їх стиснення порівняно з текстовою інформацією. Оскільки кожен вектор представлений масивом із сотень чисел, які мають бути доступними для миттєвих обчислень, виникає інфраструктурне протиріччя: висока вартість RAM обмежує масштабування до мільярдних масивів, тоді як стандартні SSD не забезпечують необхідної швидкості доступу [18, 22].

Аналіз сучасних рішень свідчить, що навіть «диско-орієнтовані» архітектури зберігають критичну залежність від обсягу оперативної пам'яті [20, 22]. Системи типу DiskANN намагаються знайти компроміс між затримкою (latency) та точністю, проте їхній практичний мінімум ресурсів жорстко обмежений специфікою навантаження [20].

Важливо зауважити, що деградація продуктивності в таких умовах має характер «обриву», а не лінійної регресії. Дослідження алгоритмів масштабованого пошуку на дисках (зокрема, аналіз SPANN [19]) демонструють існування критичних порогів життєздатності:

– Системи PipeANN та SPANN втрачають працездатність, якщо обсяг доступної пам’яті падає нижче 30% [21].

– DiskANN виявляє критичне падіння пропускної здатності при дефіциті RAM понад 20% [20].

Таким чином, у векторних базах даних зменшення обсягу пам’яті не призводить до поступового уповільнення – при досягненні певного ліміту система зазнає раптового колапсу продуктивності, фактично припиняючи ефективне виконання операцій [18, 20]. Для нівелювання цього ефекту та

віддалення точки «обриву» сучасні архітектури впроваджують методи багаторівневого стиснення та гібридної індексації. Зокрема, використання продуктового квантування (Product Quantization) дозволяє зменшити обсяг вектору в пам'яті у десятки разів, зберігаючи при цьому достатню точність для навігації [22]. Паралельно з цим, перехід до ієрархічних графів (HNSW), адаптованих для роботи з SSD, дозволяє винести основну вагу даних на диск, залишаючи в RAM лише критично важливі «маршрути» для пошуку [20, 22]. Такий синергетичний підхід дозволяє системам балансувати на межі ресурсного порогу, не допускаючи фатального колапсу продуктивності при зростанні обсягів даних.

Дрейф ембедингів погіршує якість пошуку. На відміну від традиційних збоїв баз даних, які генерують помилки, якість векторного пошуку може погіршуватися непомітно. У міру зміни ваших даних і оновлення моделей, зсув розподілу означає, що нові проіндексовані дані можуть мати інший розподіл, ніж початкові тренувальні дані [18, 23]. Вектори змінюються, але запити все ще повертають результати, просто гірші [18].

Це особливо підступно, оскільки перевірка «істинного результату» є складною. Щоб визначити, чи ваш топ-10 справді найкращий, потрібно обчислити справжній топ-10 по всьому набору даних. Тобто виконати ту саму дорогу операцію, якої ви намагалися уникнути, використовуючи наближений пошук [23].

Коли дрейф стає критичним, повна реконструкція передбачає моніторинг продуктивності індексації та запуск повної перебудови індексу [11]. Але в масштабі перебудова всього індексу стає ресурсним вузьким місцем, що порушує роботу залежних сервісів [18, 11]. Підходи на основі адаптерів, такі як Drift-Adapter, можуть відновити 95-99% продуктивності пошуку без повної перебудови ембедингів у протестованих сценаріях [24], але це все ще активна область досліджень.

Складнощі гібридного пошуку. У реальних застосунках використання чистого векторного пошуку трапляється вкрай рідко. Запити на кшталт «знайти подібні товари вартістю до 50 доларів США, які є в наявності» поєднують обчислення векторної схожості з фільтрацією за метаданими – саме на цьому етапі виникають основні труднощі [18].

Чистий векторний пошук має низку обмежень. По-перше, він не забезпечує точного зіставлення термінів і не підтримує інтерпретацію булевих виразів, внаслідок чого результати можуть бути надмірно узагальненими або нерелевантними. По-друге, відсутній вбудований механізм фільтрації за структурованими атрибутами, такими як ціна чи доступність товару.

Поширеним підходом до розв’язання цієї проблеми є гібридний пошук, який передбачає розподіл запитів між окремими механізмами пошуку за ключовими словами та векторною схожістю з подальшим об’єднанням результатів за допомогою алгоритмів, зокрема Reciprocal Rank Fusion [25]. Водночас логіка оцінювання та ранжування результатів часто реалізується поза межами основної бази даних, що призводить до неузгодженості ранжування, обмежує можливості оптимізації та збільшує інфраструктурні витрати.

Інший підхід реалізовано в системі Redis через механізм Redis Query Engine, який забезпечує підтримку векторної схожості разом із фільтрацією за географічними, числовими, теговими та текстовими метаданими в межах єдиного запиту без необхідності окремих етапів повторного ранжування.

Підтримка синхронізації векторних ембедингів із вихідними даними. Ваші вихідні дані постійно змінюються: документи оновлюються, об’єкти вилучаються зі сховища, профілі користувачів зазнають змін. Кожна така зміна потенційно потребує обчислення нових ембедінгів, а операції вставки або оновлення (upsert) можуть бути ресурсоємними, оскільки основні витрати припадають на повторне векторне представлення та індексацію – особливо за умов частих змін документів [18].

Багато команд переходять до пакетних оновлень, свідомо обмінюючи миттєву синхронізацію на зниження операційних витрат. Однак це призводить до виникнення часових інтервалів, у межах яких векторне сховище та джерело істини перебувають у асинхронному стані [18].

Розподілені архітектури додатково ускладнюють ситуацію. Коли прикладні дані зберігаються в одній базі даних, а векторні представлення – в іншій, забезпечення атомарних транзакцій між ними стає складним завданням і часто потребує спеціалізованих рішень [18]. Наслідком цього можуть бути так звані «примарні документи», коли векторний пошук повертає посилання на об’єкт, який уже відсутній в основній базі даних [18].

Системи класу RAG додають ще один рівень складності. Додавання нового факту до бази знань RAG не призводить до видалення або заміни попередніх тверджень, внаслідок чого одночасно можуть повертатися як застарілі, так і актуальні значення [11, 26]. Згідно з одним із досліджень оновлення фактів у діалогових системах на основі RAG, точність становила лише 83,3% навіть після повної реіндексації, оскільки невідповідності між формулюванням тверджень і запитів продовжували спричиняти помилки під час пошуку [26].

Горизонтальне масштабування стикається з обмеженнями комунікації. Додавання нових вузлів (node) не дає лінійного масштабування. Пропускна здатність обчислень зростає швидше, ніж пропускна здатність мережі, тому розподілений векторний пошук часто стає обмеженим мережею. Розрив між тим, як швидко вузли можуть обчислювати, і тим, як швидко вони можуть обмінюватися даними, створює верхню межу масштабування [7, 22].

Розподіл запитів між кількома вузлами може прискорити роботу, але виграш зазвичай менший за лінійний через накладні витрати на комунікацію та конкуренцію за ресурси [18, 22]. Ретельне розбиття даних допомагає, але мінімізація міжвузлової комунікації залишається відкритою проблемою [23, 22].

Операційні інструменти та моніторинг. Багатьом командам досі бракує зрілих інструментів моніторингу ML, а спеціалізовані метрики векторних баз даних, такі як затримка запитів, якість recall, час побудови індексу та використання пам'яті, часто з'являються із запізненням або додаються як другорядні елементи [27, 18].

У багатьох компаній немає достатнього досвіду, щоб налаштувати векторні бази даних, для оптимального використання ресурсів [18]. Це дуже сильно впливає на створення агентних AI-систем у корпоративному середовищі, де векторні бази даних є ключовою залежністю, але операційна зрілість ще не досягає необхідного рівня [27].

Порівняльний аналіз провідних векторних рішень у сучасній екосистемі GenAI. Для об'єктивної оцінки наявного інструментарію та вибору оптимального рішення для конкретних бізнес-задач, необхідно провести детальний аналіз популярних векторних баз даних, враховуючи їхню продуктивність, тип розгортання та специфіку обробки даних.

Критерії архітектурного вибору векторних систем у проектах GenAI. Процес вибору оптимальної векторної бази даних (ВБД) для систем генеративного штучного інтелекту вимагає комплексного оцінювання низки факторів. Вибір інфраструктурного рішення безпосередньо корелює з операційною стабільністю майбутньої системи та її здатністю до ефективної обробки багатовимірних даних. Ключові критерії вибору в межах даного дослідження класифіковано за наступними характеристиками:

– Масштабованість та архітектурна еластичність. Здатність системи до горизонтального та вертикального масштабування є критичною для проєктів із динамічним зростанням обсягів знань. ВБД повинна підтримувати високу продуктивність індексації та пошуку при переході від прототипу до промислової експлуатації, зберігаючи лінійну залежність між обчислювальними ресурсами та пропускною здатністю [28].

– Показники латентності та інтенсивність інференсу. Ефективність системи визначається швидкістю виконання запитів на пошук подібності (Similarity Search). Для додатків реального часу пріоритетними є показники низької затримки (latency) при високій щільності запитів (QPS), що забезпечує безшовну інтеграцію векторного сховища в генеративні цикли [28].

– Екосистемна сумісність. Важливим чинником є рівень інтеграції з існуючим технологічним стеком, зокрема з фреймворками оркестрації агентів (LangChain, LlamaIndex) та бібліотеками обробки вбудовувань. Наявність зрілих SDK та нативних конекторів суттєво знижує складність розгортання та пришвидшує життєвий цикл розробки ПЗ [13].

– Спеціалізація функціональних можливостей. Оцінювання має враховувати підтримку специфічних механізмів, таких як гібридний пошук (поєднання векторного та ключового), фільтрація за метаданими та оптимізація під архітектуру RAG. Наявність вбудованих інструментів семантичного ранжування дозволяє суттєво підвищити точність контекстуальної вибірки [13].

– Інформаційна безпека та комплаєнс. При оперуванні конфіденційними даними критичного значення набувають механізми захисту, шифрування у стані спокою (at rest) та під час передачі (in transit), а також відповідність нормативним вимогам щодо локалізації. Можливість розгортання в ізольованих контурах (on-premise або private cloud) є визначальною для корпоративного сектору [29].

– Економічна доцільність та TCO (Total Cost of Ownership). Аналіз передбачає оцінювання загальної вартості володіння, що включає витрати на хмарну інфраструктуру, ліцензування або витрати на інженерну підтримку у випадку рішень із відкритим кодом [30, 31].

Таким чином, ретельна верифікація доступних рішень за наведеними критеріями дозволяє обрати векторну систему, яка не лише задовольняє поточні технічні вимоги, а й забезпечує потенціал для подальшої еволюції інтелектуальних сервісів.

Таблиця 3

Експрес-порівняння провідних векторних рішень для GenAI

Параметр	Pinecone [34]	Milvus [35]	Weaviate [36]	Qdrant [37]	Pgvector [38]	Elasticsearch [39]
Спеціалізація [33]	Managed vector DB для production AI	High-scale AI/Big Data	Semantic search + knowledge graph	Real-time search + filtering	SQL + embeddings	Hybrid search (text + vector)
Архітектура	Fully managed, cloud-native, serverless	Distributed (CPU/GPU), сегменто-вана	Graph-based + modular	Rust-based, distributed HNSW	PostgreSQL extension	Distributed search engine (Lucene + ANN)
Системні обмеження	Vendor lock-in, платний	Складність деплою, ресурсо-ємність	Високе споживання RAM	Менше mature scaling ніж Milvus	Обмежена масштабованість	Не спеціалізована під vectors
Ціна [32]	$$$ (≈ $200-400+/міс)	$$-$$$ (cloud $300-600)	$$ (≈ $150-300)	$-$$ (≈ $120-250 або self-host)	$ (open-source)	$$ (infra + storage)
p95 latency [33]	~15-40 ms	~5-20 ms	~20-60 ms	~10-40 ms	~30-150 ms	~40-150 ms
QPS [33]	~200-1000	~500-2000	~100-600	~200-1200	~50-300	~100-500
Recall@10	0.90-0.97	0.92-0.99	0.88-0.95	0.90-0.97	0.80-0.92	0.85-0.93

Згідно з результатами експериментальних досліджень (зокрема VectorDBBench та vendor benchmark-звітів), сучасні векторні бази даних демонструють затримку відповіді в межах 5-60 мс для оптимізованих конфігурацій, тоді як показники Recall@10 можуть досягати 0.95-0.99 за рахунок використання алгоритмів наближеного пошуку найближчих сусідів (ANN) [21], таких як HNSW. Водночас системи, інтегровані у традиційні СУБД (наприклад, pgvector), характеризуються нижчою пропускною здатністю та більшою затримкою через обмеження транзакційної моделі.

Представлена вище порівняльна характеристика (таблиця 3) демонструє значну диференціацію ринку векторних систем за критеріями масштабованості, вартості та складності експлуатації. Однак для прийняття обґрунтованого архітектурного рішення недостатньо лише загальних показників; необхідно детально проаналізувати внутрішню логіку функціонування кожного з провідних рішень. Кожна з розглянутих систем пропонує унікальний компроміс між продуктивністю та операційними витратами, що безпосередньо впливає на стабільність RAG-пайплайнів.

Розпочнемо детальний огляд із найбільш зрілого представника сегмента Managed-рішень, який задав стандарти простоти розгортання векторного пошуку в хмарних середовищах.

Pinecone представляє сегмент проприєтарних хмарних платформ, реалізованих за моделлю fully managed serverless. Основна архітектурна ідея системи полягає в повній абстракції інфраструктурного рівня: завдання моніторингу, шардингу та динамічного розподілу ресурсів делегуються провайдеру. Це забезпечує високу еластичність обчислювальних потужностей, дозволяючи системі автоматично адаптуватися до навантажень у мільярди векторів. Завдяки нативній інтеграції з фреймворками LangChain та LlamaIndex, Pinecone мінімізує показник Time-to-Market, стаючи де-факто стандартом для швидкого прототипування RAG-додатків [34].

Milvus – це cloud-native система з відкритим кодом, розроблена для роботи з масивами екза-масштабу на основі розподілу рівнів зберігання та обчислень (вузли Query, Index, Data). Така архітектура дозволяє незалежно масштабувати ресурси та використовувати GPU-прискорення, що забезпечує

мілісекундну латентність навіть у високовимірних просторах великих корпорацій. Проте екстремальна продуктивність Milvus вимагає складної підтримки: на відміну від Weaviate, система фокусується суто на пошуку, тому розробникам потрібно самостійно налаштовувати векторизацію та керувати зовнішніми залежностями. Його впровадження є стратегічно доцільним для проєктів обсягом понад 10 млн об’єктів, де потужність критичніша за простоту адміністрування, а компромісом між цими параметрами виступає керована платформа Zilliz Cloud [35].

Weaviate займає унікальну нішу завдяки поєднанню векторної близькості та класичного пошуку за ключовими словами (BM25) у межах одного запиту. Такий гібридний підхід критично важливий для RAG-систем, оскільки він запобігає «семантичному розмиттю» термінів і забезпечує високу точність контексту. На відміну від Milvus, Weaviate має модульну структуру, що дозволяє векторизувати дані прямо «всередині» бази, значно спрощуючи інженерний конвеєр. Використання GraphQL робить його зручним для побудови складних знаннєвих графів, хоча за високу релевантність доводиться платити підвищеним споживанням RAM та CPU. Weaviate є оптимальним вибором для проєктів середнього масштабу (до 100 млн об’єктів), де якість пошуку та швидкість розробки є пріоритетнішими за граничну пропускну здатність системи [36].

На відміну від важкої інфраструктури Milvus, Qdrant робить ставку на ресурсну ефективність завдяки мові Rust. Це забезпечує високу продуктивність при мінімальному споживанні пам'яті, що ідеально підходить для IoT-пристроїв або edge-розгортань. Головною перевагою системи є гнучка фільтрація за метаданими через JSON: у RAG-додатках це дозволяє миттєво поєднувати семантичний пошук із жорсткими бізнес-правилами (права доступу, дати) без втрати швидкості.

Qdrant є надзвичайно лояльним до розробників на етапі прототипування та MVP, забезпечуючи стабільну затримку в 1 мс на малих масивах даних. Проте при масштабуванні понад 50 млн векторів система демонструє деградацію пропускної здатності порівняно з Milvus і гірше справляється з інтенсивним одночасним записом. Таким чином, Qdrant – це стратегічний вибір для стартапів та проєктів середнього масштабу, де бюджетна ефективність та гнучка робота з метаданими важливіші за екза-масштабованість [37].

На відміну від спеціалізованих систем, pgvector інтегрує векторні функції безпосередньо в екосистему PostgreSQL, дозволяючи виконувати семантичний пошук поруч із реляційними запитами без складної синхронізації даних. Завдяки розширенню pgvectorscale та алгоритмам DiskANN, система демонструє високу продуктивність – до 471 QPS (при 99% точності) на масивах у 50 млн векторів, випереджаючи за латентністю навіть Pinecone. Експлуатація pgvector зазвичай на 75% дешевша за Managed-сервіси, оскільки дозволяє використовувати вже наявну інфраструктуру та експертизу команди. Проте при масштабуванні понад 100 млн об’єктів реляційна модель може створювати ресурсну конкуренцію з основними транзакціями, поступаючись гнучкістю розподіленим системам типу Milvus. Таким чином, pgvector є оптимальним вибором для компаній, які прагнуть додати AI-можливості у свій SQL-стек без ускладнення архітектури на середніх масштабах даних [38].

Заключним об’єктом аналізу є Elasticsearch, який представляє категорію зрілих корпоративних систем, адаптованих до вимог GenAI. Його головна перевага – експлуатаційна стабільність та найкращий у класі механізм гібридного пошуку на основі методу Reciprocal Rank Fusion (RRF), що дозволяє поєднувати алгоритм BM25 із семантичною схожістю для максимальної релевантності в RAG-системах. Впровадження алгоритму HNSW із 8-бітним квантуванням у версіях 8.14+ суттєво скоротило розрив зі спеціалізованими базами, хоча за латентністю система все ще поступається Pinecone. Основним недоліком є висока ресурсна інтенсивність (RAM/CPU) та складність Query DSL, що робить Elasticsearch ідеальним вибором лише для зрілих екосистем, де вже використовується Elastic Stack. Для нових проєктів, орієнтованих виключно на вектори, платформа може бути надлишковою, проте вона залишається безальтернативною для змішаних навантажень, що вимагають одночасного поєднання аналітики, повнотекстового та семантичного пошуку [39].

Комплексний аналіз свідчить про перехід від експериментів до промислової експлуатації векторної інфраструктури. Сучасний ринок чітко сегментувався за трьома напрямами: спеціалізована потужність (Milvus, Pinecone) для екза-масштабів, гібридизація (Weaviate, Elasticsearch) для максимальної релевантності контексту та архітектурна конвергенція (pgvector, Qdrant) для спрощення стека. У 2026 році вибір ВБД став стратегічним рішенням, що визначає вартість володіння (TCO) та життєздатність інтелектуальних агентів. Ефективність наступного покоління когнітивних систем залежить не стільки від параметрів LLM, скільки від здатності інфраструктури забезпечити швидкість і консистентність знань, що і слугуватиме фундаментом для проектування складних мультиагентних середовищ.

Синтез результатів та рекомендації щодо вибору. На основі проведеного аналізу інфраструктурних викликів та характеристик сучасних VDB нижче наведено рекомендації щодо вибору технологічного стеку для вирішення конкретних проблем промислової експлуатації:

– Для подолання проблеми RAM: Найбільш ефективним рішенням є pgvector з розширенням pgvectorscale та алгоритмом DiskANN. Це дозволяє винести значну частину індексів на SSD, запобігаючи «ресурсному обриву» та зберігаючи високу швидкість при менших витратах на оперативну пам'ять.

– Для мінімізації дрейфу ембедингів та покращення якості: Доцільно використовувати Pinecone, який пропонує найбільш зрілі інструменти моніторингу релевантності та стабільності векторного простору «з коробки».

– Для реалізації складного гібридного пошуку: Безальтернативними лідерами є Weaviate та Elasticsearch. Weaviate забезпечує кращу нативну інтеграцію векторів і ключових слів (BM25), тоді як Elasticsearch пропонує найпотужніші алгоритми ранжування (RRF) для корпоративних знаннєвих графів.

– Для синхронізації ембедингів із джерелами: Найкращу цілісність даних забезпечує pgvector, оскільки він зберігає вектори безпосередньо в реляційній базі, усуваючи потребу в складних ETL-процесах синхронізації.

– Для горизонтального масштабування без втрат: Оптимальним вибором для масивів понад 100 млн векторів залишається Milvus завдяки повній дезагрегації рівнів обчислень та зберігання, що мінімізує комунікаційні затримки між вузлами.

– Для спрощення операційного моніторингу: У проєктах, де критичною є швидкість розгортання та легкість підтримки («footprint»), найкраще себе проявляє Qdrant (завдяки Rust-архітектурі) або керована платформа Zilliz Cloud.

Такий синтез доводить, що універсальної системи не існує: вибір архітектури когнітивної пам'яті повинен базуватися на пріоритетності конкретних інфраструктурних обмежень проєкту.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Проведені дослідження архітектурних та техніко-економічних характеристик векторних баз даних дозволяють зробити наступні висновки.

Векторні бази даних є критичним компонентом когнітивної інфраструктури сучасних генеративних систем штучного інтелекту, виконуючи функцію зовнішньої непараметричної пам'яті для великих мовних моделей. Їх інтеграція через архітектурні шаблони RAG та RAFT дозволяє подолати статичний характер знань та значно знизити ризик генерування недостовірної інформації, перетворюючи закриті системи ШІ на відкриті екосистеми, здатні працювати з актуальними корпоративними даними в режимі реального часу.

Промислова експлуатація векторних баз даних пов'язана зі значними інфраструктурними викликами, центральною з яких є раптовий обвал продуктивності у разі нестачі оперативної пам'яті, що відбувається, коли її доступний обсяг падає нижче порогового значення, що залежить від використовуваного алгоритму індексації. Окремою прихованою загрозою є семантичний дрейф вбудовувань, який спричиняє поступове зниження якості пошуку без видимих технічних збоїв.

Ринок векторних рішень характеризується динамічною сегментацією та поляризацією підходів. Спеціалізовані системи забезпечують максимальну продуктивність у масштабі, гібридні рішення підвищують бізнес-релевантність, поєднуючи векторний та повнотекстовий пошук, тоді як конвергентні архітектури задовольняють потребу у спрощенні технологічного стеку, інтегруючи семантичний пошук в існуючі середовища.

Перспективи подальших досліджень пов'язані з кількома ключовими напрямками. Розробка ефективних механізмів моніторингу та автоматичної корекції семантичного дрейфу вбудовувань в умовах постійного оновлення корпоративних знань потребує пріоритетної уваги. Актуальним залишається дослідження адаптивних стратегій управління пам'яттю для усунення ефектів порогової продуктивності при масштабуванні векторних індексів. Окремим перспективним напрямком є дослідження можливостей об'єднання гібридних архітектур пошуку, що поєднують семантичне та повнотекстове індексування в єдиному технологічному середовищі. Нарешті, питання стандартизації метрик оцінки якості векторного пошуку в контексті конкретних вимог промислових RAG-систем потребує подальшого розвитку, що забезпечить більш обґрунтований вибір інфраструктурних рішень для конкретних прикладних завдань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. FAISS. (n.d.). *Documentation*. https://faiss.ai/index.html

2. Taipalus, T. (2024). Vector database management systems: Fundamental concepts, use-cases, and current challenges. *Cognitive Systems Research, 85*. https://doi.org/10.1016/j.cogsys.2024.101216
3. Zhang, T., et al. (2024). *RAFT: Adapting language model to domain specific RAG*. arXiv. https://doi.org/10.48550/arXiv.2403.10131
4. Gao, L., et al. (2023). *Retrieval-augmented generation for large language models: A survey*. arXiv. https://doi.org/10.48550/arXiv.2312.10997
5. Pan, X., et al. (2023). *Survey of vector database management systems*. arXiv. https://doi.org/10.48550/arXiv.2310.14021
6. Shi, Y., et al. (2024). *Enhancing retrieval and managing retrieval: A four-module synergy for improved quality and efficiency in RAG systems*. arXiv. https://doi.org/10.48550/arXiv.2407.10670
7. Ma, L., et al. (2023). *A comprehensive survey on vector database: Storage and retrieval technique, challenge*. arXiv. https://doi.org/10.48550/arXiv.2310.11703
8. Wang, M., et al. (2023). *Embedding in recommender systems: A survey*. arXiv. https://doi.org/10.48550/arXiv.2310.18608
9. Milvus. (n.d.). *Vector database documentation*. https://milvus.io
10. Pinecone. (n.d.). *Vector database guide*. https://www.pinecone.io/learn/vector-database/
11. Gao, Y., et al. (2023). *Retrieval-augmented generation for large language models: A survey*. arXiv. https://doi.org/10.48550/arXiv.2312.10997
12. Huang, L., et al. (2023). *A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions*. arXiv. https://doi.org/10.48550/arXiv.2311.05232
13. Gupta, S., Ranjan, R., & Singh, S. N. (2024). *A comprehensive survey of retrieval-augmented generation (RAG): Evolution, current landscape and future directions*. arXiv. https://doi.org/10.48550/arXiv.2410.12837
14. Pan, J. J., Wang, J., & Li, G. (2024). *Survey of vector database management systems*. https://dbgroup.cs.tsinghua.edu.cn/ligl/papers/vldbj2024-vectordb.pdf
15. Schwaber-Cohen, R. (n.d.). *What is a vector database & how does it work? Use cases + examples*. Pinecone. https://www.pinecone.io/learn/vector-database/
16. Tu, Y., Su, W., Zhou, Y., Liu, Y., & Ai, Q. (2025). *RbFT: Robust fine-tuning for retrieval-augmented generation against retrieval defects*. arXiv. https://doi.org/10.48550/arXiv.2501.18365
17. Gu, J. (2024). A research of challenges and solutions in retrieval augmented generation (RAG) systems. *Humanities and Social Sciences Communications / DRPress (HSET)*. https://doi.org/10.54097/364hex16
18. Wallace, J. A. (n.d.). *What are the most common vector database challenges?* Redis. https://redis.io/blog/common-challenges-working-with-vector-databases/
19. Kang, D., Jiang, D., Yang, H., Liu, H., & Li, B. (2025). *Scalable disk-based approximate nearest neighbor search with page-aligned graph*. arXiv. https://doi.org/10.48550/arXiv.2509.25487
20. Subramanya, S. J., Devvrit, Kadekodi, R., Krishaswamy, R., & Simhadri, H. V. (2019). DiskANN: Fast accurate billion-point nearest neighbor search on a single node. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (pp. 13766-13776). Curran Associates. https://dl.acm.org/doi/abs/10.5555/3454287.3455520
21. Chen, Q., Zhao, B., Wang, H., Li, M., Liu, C., Li, Z., Yang, M., & Wang, J. (2021). *SPANN: Highly-efficient billion-scale approximate nearest neighbor search*. arXiv. https://doi.org/10.48550/arXiv.2111.08566
22. Microsoft Research. (n.d.). *DiskANN: Vector search for all*. https://www.microsoft.com/en-us/research/project/project-akupara-approximate-nearest-neighbor-search/
23. Jiang, P., Ouyang, S., Jiao, Y., Zhong, M., Tian, R., & Han, J. (2025). *A survey on retrieval and structuring augmented generation with large language models*. arXiv. https://arxiv.org/html/2509.10697v1
24. Vejendla, H. (2025). *Drift-adapter: A practical approach to near zero-downtime embedding model upgrades in vector databases*. arXiv. https://arxiv.org/pdf/2509.23471
25. Milvus. (n.d.). *RRF ranker*. https://milvus.io/docs/rrf-ranker.md
26. Li, Z., Wang, Z., Wang, W., Hung, K., Xie, H., & Wang, F. (2025). Retrieval-augmented generation for educational application: A systematic survey. *Computers and Education: Artificial Intelligence, 8*, 100417. https://doi.org/10.1016/j.caeai.2025.100417
27. Brown, A., Roman, M., & Devereux, B. (2025). *A systematic literature review of retrieval-augmented generation: Techniques, metrics, and challenges*. arXiv. https://doi.org/10.48550/arXiv.2508.06401
28. Johnson, J., Douze, M., & Jégou, H. (2017). *Billion-scale similarity search with GPUs*. arXiv. https://doi.org/10.48550/arXiv.1702.08734

29. Chakraborty, A., Dahal, C., & Gupta, V. (2025). *Federated retrieval-augmented generation: A systematic mapping study*. arXiv. https://arxiv.org/pdf/2505.18906
30. Amazon Web Services. (n.d.). *Cost considerations for vector databases*. https://docs.aws.amazon.com/prescriptive-guidance/latest/choosing-an-aws-vector-database-for-rag-use-cases/cost.html
31. Microsoft Azure. (n.d.). *Pricing for search services*. https://azure.microsoft.com/en-us/pricing/details/search/
32. TensorBlue Blog. (2025). *Best vector database 2025: Pinecone vs Weaviate vs Qdrant vs Milvus*. https://tensorblue.com/blog/vector-database-comparison-pinecone-weaviate-qdrant-milvus-2025
33. Theodo. (n.d.). *How to choose your vector database*. https://www.theodo.com/en-fr/blog/how-to-choose-your-vector-database
34. Pinecone. (n.d.). *Official website*. https://www.pinecone.io/
35. Milvus. (n.d.). *Official website*. https://milvus.io/
36. Weaviate. (n.d.). *Official website*. https://weaviate.io/
37. Qdrant. (n.d.). *Official website*. https://qdrant.tech/
38. pgvector. (n.d.). *GitHub repository*. GitHub. https://github.com/pgvector/pgvector
39. Amazon Web Services. (n.d.). *What is Elasticsearch?* https://aws.amazon.com/what-is/elasticsearch/
40. Kuznetsov, O., Smirnov, O., Akhmetov, B., Alimseitova, Z., & Imoize, A. L. (2025). Deep learning frontiers in copy-move forgery detection: Advances, challenges, and future directions. In *Advancements in Cybersecurity Next Generation Systems and Applications* (pp. 202-229). https://doi.org/10.1201/9781003546153
41. Smirnov, O., Fedorov, E., Neskorodieva, A., & Neskorodieva, T. (2024). Intellectual classification method of gymnastic elements based on combinations of descriptive and generative approaches. *CEUR Workshop Proceedings, 3664*, 11-23.
42. Al-Mudhafar, A. A., Smirnova, T., Buravchenko, K., & Smirnov, O. (2023). The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning. *Advanced Information Systems, 7*(2), 49-56.
43. Smirnov, O., Karapetyan, A., & Fedorov, E. (2022). Creating neural network and single solution human-based metaheuristic methods of solving the traveling salesman problem. *CEUR Workshop Proceedings, 3312*, 47-58.
44. Lysenko, I., Mynailenko, R., Smirnov, S., Buravchenko, K., Yakymenko, N., & Smirnov, O. (2025). Research of artificial intelligence tools for intelligent data analysis. *Cybersecurity: Education, Science, Technique, 3*(31), 227-241. https://doi.org/10.28925/2663-4023.2025.31.1022
45. Usik, P. S., Smirnova, T. V., Buravchenko, K. O., Smirnov, O. A., Ulichev, O. S., & Smirnov, S. A. (2025). Research of cybersecurity technologies for banking systems using artificial intelligence. *Cybersecurity: Education, Science, Technique, 1*(29), 704-716. https://doi.org/10.28925/2663-4023.2025.29.930
46. Smirnov, O. A., Konstantynova, L. V., Konoplitska-Slobodeniuk, O. K., Kozirova, N. V., Yakymenko, N. M., Dorenskyi, O. P., & Buravchenko, K. O. (2025). Research of artificial intelligence tools for working with databases and data analysis. *Cybersecurity: Education, Science, Technique, 3*(27), 429-448. https://doi.org/10.28925/2663-4023.2025.27.763
47. Al-Mudhafar, A. A. A., Smirnova, T. V., Buravchenko, K. O., & Smirnov, O. A. (2023). Method of assessment and improvement of subscriber user experience in software-defined networks based on machine learning. *Advanced Information Systems, 7*(2), 49-56. https://doi.org/10.20998/2522-9052.2023.2.07

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (33), 2026
ISSN 2663 – 4023

Oleksii Smirnov
Doctor of Technical Sciences, Professor,
Head of Cybersecurity & Software Academic Department
Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine
ORCID: 0000-0001-9543-874X
dr.smirnovoa@gmail.com

Roman Tkachuk
Senior Systems Engieer at EPAM Ukraine,
assistant of Cybersecurity & Software Academic Department
Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine
ORCID: 0000-0002-1984-0419
tkachukroman64@gmail.com

Nataliia Kozirova
lecturer of Cybersecurity & Software Academic Department
Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine
ORCID: 0009-0005-8753-5132
natalidonchenko23@gmail.com

Liliia Konstantynova
Lecturer of Cybersecurity & Software Academic Department
Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine
ORCID: 0000-0002-3305-2427
liliyashel1976@gmail.com

Oksana Konoplitska-Slobodeniuk
Lecturer
Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine Центральноукраїнський національний технічний університет
ORCID: 0000-0001-9981-5194
ksuha80@gmail.com

Nataliia Yakymenko
candidate of Physical and Mathematical Sciences, associate professor, associate professor of Cybersecurity & Software Academic Department
Central Ukrainian National Technical University, Kropivnitskiy, Ukraine
ORCID: 0000-0002-4498-0093
yakimenko_n_m@ukr.net

Serhii Smirnov
Candidate of Science (Engineering), associate professor, associate professor of Cybersecurity & Software Academic Department
Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine
ORCID: 0000-0002-7649-7442
smirnov.ser.81@gmail.com

RESEARCH INTO THE APPLICATION OF VECTOR DATABASES IN GENERATIVE ARTIFICIAL INTELLIGENCE

Abstract. This paper studies the use of vector databases in generative artificial intelligence. The purpose of this article is to study the possibility of using vector databases as the foundation of modern AI infrastructure and their role in expanding the cognitive capabilities of GenAI. The object of the study is the process of using databases in artificial intelligence. The subject of the study is the use of vector databases in generative artificial intelligence. The following tasks were solved in this study: the role of vector repositories in expanding the cognitive capabilities of generative AI was investigated; a comparative analysis of traditional relational systems and vector databases was carried out; the mechanics and verification potential of the RAG and RAFT architectural paradigms were analyzed. Prospects for further research are identified, which consist of the following: development

of effective mechanisms for monitoring and automatic correction of semantic drift of embeddings in conditions of constant updating of corporate knowledge; research into adaptive memory management strategies to eliminate threshold performance effects when scaling vector indexes; research into the possibilities of combining hybrid search architectures that combine semantic and full-text indexing in a single technological environment; standardization of metrics for assessing the quality of vector search in the context of specific requirements of industrial RAG systems, which will ensure a more informed choice of infrastructure solutions for specific application tasks.

Keywords: databases, artificial intelligence, vector databases (VDB), generative artificial intelligence (GenAI), large language models (LLM), Retrieval-Augmented Generation (RAG), semantic search, vector embeddings, HNSW, RAFT, scalability, cognitive infrastructure, ANN, quantization.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. FAISS. (n.d.). *Documentation*. https://faiss.ai/index.html
2. Taipalus, T. (2024). Vector database management systems: Fundamental concepts, use-cases, and current challenges. *Cognitive Systems Research, 85*. https://doi.org/10.1016/j.cogsys.2024.101216
3. Zhang, T., et al. (2024). *RAFT: Adapting language model to domain specific RAG*. arXiv. https://doi.org/10.48550/arXiv.2403.10131
4. Gao, L., et al. (2023). *Retrieval-augmented generation for large language models: A survey*. arXiv. https://doi.org/10.48550/arXiv.2312.10997
5. Pan, X., et al. (2023). *Survey of vector database management systems*. arXiv. https://doi.org/10.48550/arXiv.2310.14021
6. Shi, Y., et al. (2024). *Enhancing retrieval and managing retrieval: A four-module synergy for improved quality and efficiency in RAG systems*. arXiv. https://doi.org/10.48550/arXiv.2407.10670
7. Ma, L., et al. (2023). *A comprehensive survey on vector database: Storage and retrieval technique, challenge*. arXiv. https://doi.org/10.48550/arXiv.2310.11703
8. Wang, M., et al. (2023). *Embedding in recommender systems: A survey*. arXiv. https://doi.org/10.48550/arXiv.2310.18608
9. Milvus. (n.d.). *Vector database documentation*. https://milvus.io
10. Pinecone. (n.d.). *Vector database guide*. https://www.pinecone.io/learn/vector-database/
11. Gao, Y., et al. (2023). *Retrieval-augmented generation for large language models: A survey*. arXiv. https://doi.org/10.48550/arXiv.2312.10997
12. Huang, L., et al. (2023). *A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions*. arXiv. https://doi.org/10.48550/arXiv.2311.05232
13. Gupta, S., Ranjan, R., & Singh, S. N. (2024). *A comprehensive survey of retrieval-augmented generation (RAG): Evolution, current landscape and future directions*. arXiv. https://doi.org/10.48550/arXiv.2410.12837
14. Pan, J. J., Wang, J., & Li, G. (2024). *Survey of vector database management systems*. https://dbgroup.cs.tsinghua.edu.cn/ligl/papers/vldbj2024-vectordb.pdf
15. Schwaber-Cohen, R. (n.d.). *What is a vector database & how does it work? Use cases + examples*. Pinecone. https://www.pinecone.io/learn/vector-database/
16. Tu, Y., Su, W., Zhou, Y., Liu, Y., & Ai, Q. (2025). *RbFT: Robust fine-tuning for retrieval-augmented generation against retrieval defects*. arXiv. https://doi.org/10.48550/arXiv.2501.18365
17. Gu, J. (2024). A research of challenges and solutions in retrieval augmented generation (RAG) systems. *Humanities and Social Sciences Communications / DRPress (HSET)*. https://doi.org/10.54097/364hex16
18. Wallace, J. A. (n.d.). *What are the most common vector database challenges?* Redis. https://redis.io/blog/common-challenges-working-with-vector-databases/
19. Kang, D., Jiang, D., Yang, H., Liu, H., & Li, B. (2025). *Scalable disk-based approximate nearest neighbor search with page-aligned graph*. arXiv. https://doi.org/10.48550/arXiv.2509.25487
20. Subramanya, S. J., Devvrit, Kadekodi, R., Krishaswamy, R., & Simhadri, H. V. (2019). DiskANN: Fast accurate billion-point nearest neighbor search on a single node. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (pp. 13766-13776). Curran Associates. https://dl.acm.org/doi/abs/10.5555/3454287.3455520

21. Chen, Q., Zhao, B., Wang, H., Li, M., Liu, C., Li, Z., Yang, M., & Wang, J. (2021). *SPANN: Highly-efficient billion-scale approximate nearest neighbor search*. arXiv. https://doi.org/10.48550/arXiv.2111.08566
22. Microsoft Research. (n.d.). *DiskANN: Vector search for all*. https://www.microsoft.com/en-us/research/project/project-akupara-approximate-nearest-neighbor-search/
23. Jiang, P., Ouyang, S., Jiao, Y., Zhong, M., Tian, R., & Han, J. (2025). *A survey on retrieval and structuring augmented generation with large language models*. arXiv. https://arxiv.org/html/2509.10697v1
24. Vejendla, H. (2025). *Drift-adapter: A practical approach to near zero-downtime embedding model upgrades in vector databases*. arXiv. https://arxiv.org/pdf/2509.23471
25. Milvus. (n.d.). *RRF ranker*. https://milvus.io/docs/rrf-ranker.md
26. Li, Z., Wang, Z., Wang, W., Hung, K., Xie, H., & Wang, F. (2025). Retrieval-augmented generation for educational application: A systematic survey. *Computers and Education: Artificial Intelligence, 8*, 100417. https://doi.org/10.1016/j.caeai.2025.100417
27. Brown, A., Roman, M., & Devereux, B. (2025). *A systematic literature review of retrieval-augmented generation: Techniques, metrics, and challenges*. arXiv. https://doi.org/10.48550/arXiv.2508.06401
28. Johnson, J., Douze, M., & Jégou, H. (2017). *Billion-scale similarity search with GPUs*. arXiv. https://doi.org/10.48550/arXiv.1702.08734
29. Chakraborty, A., Dahal, C., & Gupta, V. (2025). *Federated retrieval-augmented generation: A systematic mapping study*. arXiv. https://arxiv.org/pdf/2505.18906
30. Amazon Web Services. (n.d.). *Cost considerations for vector databases*. https://docs.aws.amazon.com/prescriptive-guidance/latest/choosing-an-aws-vector-database-for-rag-use-cases/cost.html
31. Microsoft Azure. (n.d.). *Pricing for search services*. https://azure.microsoft.com/en-us/pricing/details/search/
32. TensorBlue Blog. (2025). *Best vector database 2025: Pinecone vs Weaviate vs Qdrant vs Milvus*. https://tensorblue.com/blog/vector-database-comparison-pinecone-weaviate-qdrant-milvus-2025
33. Theodo. (n.d.). *How to choose your vector database*. https://www.theodo.com/en-fr/blog/how-to-choose-your-vector-database
34. Pinecone. (n.d.). *Official website*. https://www.pinecone.io/
35. Milvus. (n.d.). *Official website*. https://milvus.io/
36. Weaviate. (n.d.). *Official website*. https://weaviate.io/
37. Qdrant. (n.d.). *Official website*. https://qdrant.tech/
38. pgvector. (n.d.). *GitHub repository*. GitHub. https://github.com/pgvector/pgvector
39. Amazon Web Services. (n.d.). *What is Elasticsearch?* https://aws.amazon.com/what-is/elasticsearch/
40. Kuznetsov, O., Smirnov, O., Akhmetov, B., Alimseitova, Z., & Imoize, A. L. (2025). Deep learning frontiers in copy-move forgery detection: Advances, challenges, and future directions. In *Advancements in Cybersecurity Next Generation Systems and Applications* (pp. 202-229). https://doi.org/10.1201/9781003546153
41. Smirnov, O., Fedorov, E., Neskorodieva, A., & Neskorodieva, T. (2024). Intellectual classification method of gymnastic elements based on combinations of descriptive and generative approaches. *CEUR Workshop Proceedings, 3664*, 11-23.
42. Al-Mudhafar, A. A., Smirnova, T., Buravchenko, K., & Smirnov, O. (2023). The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning. *Advanced Information Systems, 7*(2), 49-56.
43. Smirnov, O., Karapetyan, A., & Fedorov, E. (2022). Creating neural network and single solution human-based metaheuristic methods of solving the traveling salesman problem. *CEUR Workshop Proceedings, 3312*, 47-58.
44. Lysenko, I., Mynailenko, R., Smirnov, S., Buravchenko, K., Yakymenko, N., & Smirnov, O. (2025). Research of artificial intelligence tools for intelligent data analysis. *Cybersecurity: Education, Science, Technique, 3*(31), 227-241. https://doi.org/10.28925/2663-4023.2025.31.1022
45. Usik, P. S., Smirnova, T. V., Buravchenko, K. O., Smirnov, O. A., Ulichev, O. S., & Smirnov, S. A. (2025). Research of cybersecurity technologies for banking systems using artificial intelligence. *Cybersecurity: Education, Science, Technique, 1*(29), 704-716. https://doi.org/10.28925/2663-4023.2025.29.930
46. Smirnov, O. A., Konstantynova, L. V., Konoplitska-Slobodeniuk, O. K., Kozirova, N. V., Yakymenko, N. M., Dorenskyi, O. P., & Buravchenko, K. O. (2025). Research of artificial intelligence tools for

working with databases and data analysis. *Cybersecurity: Education, Science, Technique*, *3*(27), 429-448. https://doi.org/10.28925/2663-4023.2025.27.763

47. Al-Mudhafar, A. A. A., Smirnova, T. V., Buravchenko, K. O., & Smirnov, O. A. (2023). Method of assessment and improvement of subscriber user experience in software-defined networks based on machine learning. *Advanced Information Systems*, *7*(2), 49-56. https://doi.org/10.20998/2522-9052.2023.2.07

Отримано редакцією журналу / Received: 27.02.26
Прорецензовано / Revised: 10.03.26
Схвалено до друку / Accepted: 25.06.26