НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.041

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
«___»_____2025 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Метод та програмний засіб для формування Skyline-вибірок з багатовимірних даних»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІП-42мп
Кривоносюк Віталій Валерійович _____

Керівник:
Професор кафедри ІПІ, д.т.н., проф.,
Стеценко Інна Вячеславівна, _____

Рецензент:
доцент кафедри ІСТ, к.т.н., доц.,
Лісовиченко Олег Іванович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2025 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Едуард ЖАРІКОВ

«___»_____2025р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Кривоносюку Віталію Валерійовичу

1. Тема дисертації «Метод та програмний засіб формування Skyline-вибірок з багатовимірних даних», науковий керівник дисертації Стеценко Інна Вячеславівна, професор, д.т.н., проф., затверджені наказом по університету від «06» листопада 2025 р. № 4841-с

2. Термін подання студентом дисертації «15» грудня 2025 р.

3. Об’єкт дослідження – алгоритми та програмне забезпечення для формування Skyline-вибірок.

4. Предмет дослідження – підходи, методи та засоби розроблення програмного забезпечення для формування Skyline-вибірок.

5. Перелік завдань, які потрібно розробити – аналіз існуючих методів багатокритеріального прийняття рішень; дослідження проблем масштабованості для великих обсягів даних при формуванні Skyline-вибірок; дослідження методів формування Skyline-вибірок з багатовимірних даних; дослідження кластеризації як способу вирішення проблеми «прокляття розмірності» (CoD); оцінка ефективності запропонованого рішення.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 3 плакати.

7. Орієнтовний перелік публікацій – одна публікація.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розробка удосконаленого методу формування Skyline-вибірок	Стеценко Інна Вячеславівна, професор, д.т.н., проф.		

9.Дата видачі завдання «1» вересня 2025 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Видача завдання	01.09.2025	
2	Збір та аналіз інформації, формулювання вимог	12.09.2025	
3	Огляд та аналіз методів та алгоритмів	18.09.2025	
4	Розробка методів та алгоритмів, проєктування архітектури	02.10.2025	
5	Розробка схем взаємодії компонентів	10.10.2025	
6	Написання програмної реалізації	24.10.2025	
7	Виконання експериментальних досліджень	05.11.2025	
8	Оформлення пояснювальної записки	20.11.2025	
9	Подання дисертації на попередній захист	26.11.2025	
10	Подання дисертації на захист	15.12.2025	

Студент Віталій Кривоносюк

Науковий керівник Інна Стеценко

РЕФЕРАТ

Розмір пояснювальної записки – 105 аркушів, містить 17 ілюстрацій, 38 таблиць, 20 посилань на джерела, 1 додаток.

Актуальність теми. У роботі розглянуто проблеми, пов’язані з формуванням Skyline-вибірок для багатовимірних даних, аналізом існуючих методів багатокритеріального прийняття рішень, дослідженням проблем масштабованості та явища «прокляття розмірності» (CoD), а також особливостями інтеграції Skyline-підходу в інформаційні системи. Показано основні особливості існуючих рішень, їх переваги та недоліки. Виявлено потребу у пошуку методів вирішення проблеми різкого збільшення розміру Skyline-вибірок при роботі з багатовимірними даними, або CoD (curse of dimensionality).

Мета дослідження. Основною метою є вирішення проблем CoD та високої складності інтеграції існуючих рішень для формування Парето-фронтів шляхом розробки та впровадження методу та спеціалізованого програмного засобу формування Skyline-вибірок.

Об’єкт дослідження: алгоритми та програмне забезпечення для формування Skyline-вибірок.

Предмет дослідження: підходи, методи та засоби розроблення програмного забезпечення для формування Skyline-вибірок.

Для реалізації поставленої мети **сформульовані наступні завдання**:

- аналіз існуючих методів багатокритеріального прийняття рішень;
- дослідження проблем масштабованості для великих обсягів даних при формуванні Skyline-вибірок;
- дослідження методів формування Skyline-вибірок з багатовимірних даних;
- дослідження кластеризації як способу вирішення проблеми «прокляття розмірності» (CoD);
- оцінка ефективності запропонованого рішення.

Наукова новизна результатів магістерської дисертації полягає в тому, що запропоновано архітектурне рішення для формування Skyline-вибірок, яке, на відміну від інших, надає користувачеві можливість обмеження розміру Парето-фронту та його інтерактивного дослідження. Результат досягнутий шляхом розробки модернізованого методу формування Skyline-вибірок.

Практичне значення одержаних результатів полягає у розробці програмного засобу Skyline-фільтрації та багатокритеріальної оцінки альтернатив, який може бути інтегрований у прикладні системи для підтримки прийняття рішень у таких галузях, як електронна комерція, фінансовий скоринг, логістика, туристичні сервіси та аналітичні платформи, забезпечуючи підвищення якості рішень, прозорість логіки відбору та ефективне використання великих масивів даних.

Зв'язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

Апробація. Наукові положення дисертації пройшли апробацію на IX Міжнародній науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2025)».

Публікації. Наукові положення дисертації опубліковані в:

1) Кривоносюк В.В., Стеценко І.В. Метод та програмний засіб формування Skyline-вибірок з багатовимірних даних. Матеріали IX Міжнародної науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2025)». Секція кафедри інформатики та програмної інженерії. 26-28 листопада 2025 р. Київ.

2) Кривоносюк В.В., Стеценко І.В. (2026). Метод та програмний засіб формування Skyline-вибірок з багатовимірних даних. Міжвідомчий

науково-технічний збірник «Адаптивні системи автоматичного управління» 1(48) [Прийнята до друку].

Ключові слова: Skyline, Парето-фронт, прокляття розмірності, кластеризація, BNL, SkyCell, Randomized Multi-Pass.

ABSTRACT

Explanatory note size – 105 pages, contains 17 illustrations, 38 tables, 20 references, 1 addition.

Topicality. The work addresses issues related to the formation of Skyline samples for multidimensional data, analysis of existing multi-criteria decision-making methods, investigation of scalability problems and the “curse of dimensionality” (CoD), as well as the specifics of integrating the Skyline approach into information systems. The main characteristics of existing solutions, their advantages, and disadvantages are presented. The need to find methods for solving the problem of the sharp increase in the size of Skyline samples when working with multidimensional data (CoD) is identified.

The aim of the study. The main goal is to solve CoD-related problems and the high complexity of integrating existing solutions for forming Pareto fronts by developing and implementing a method and specialized software tool for generating Skyline samples.

The object of research: algorithms and software for generating Skyline samples.

The subject of research: approaches, methods, and tools for developing software to form Skyline samples.

To achieve this goal, the **following tasks** were formulated:

- Analysis of existing multi-criteria decision-making methods.
- Investigation of scalability issues for large data volumes when generating Skyline samples.
- Study of methods for forming Skyline samples from multidimensional data.
- Investigation of clustering as a way to address the “curse of dimensionality” (CoD).
- Evaluation of the effectiveness of the proposed solution.

The scientific novelty of the master's thesis results lies in the proposed architectural solution for forming Skyline samples which, unlike existing approaches, provides the user with the ability to limit the size of the Pareto front and interactively explore it. The result was achieved by developing a modernized Skyline-sample formation method.

The practical value of the results lies in the development of a software tool for Skyline filtering and multi-criteria evaluation of alternatives, which can be integrated into application systems supporting decision-making in fields such as e-commerce, financial scoring, logistics, tourism services, and analytical platforms. The tool improves decision quality, transparency of selection logic, and efficient use of large data sets.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Computer Science and Software Engineering of the National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute».

Approbation. The scientific findings of the thesis were presented at the IX International Scientific and Practical Conference of Young Scientists and Students "Software Engineering and Advanced Information Technologies (SoftTech-2025)".

Publications. The scientific provisions of the dissertation were published in:

1) Kryvonosiuk V.V., Stetsenko I.V. Method and software tool for forming Skyline samples from multidimensional data. Proceedings of the IX International Scientific and Practical Conference of Young Scientists and Students "Software Engineering and Advanced Information Technologies (SoftTech-2025)". Section of the Department of Computer Science and Software Engineering. November 26–28, 2025. Kyiv.

2) Kryvonosiuk V.V., Stetsenko I.V. Method for forming Skyline samples from multidimensional data. *Adaptive Automatic Control Systems*, Volume 1, No. 48 (2026).

Keywords: Skyline, Pareto front, curse of dimensionality, clustering, BNL, SkyCell, Randomized Multi-Pass.

Пояснювальна записка

до магістерської дисертації

на тему: ***Метод та програмний засіб формування Skyline-вибірок з багатовимірних даних***

виконав студент групи ІП-42мп

Кривоносюк Віталій Валерійович

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

JDBC – прикладний програмний інтерфейс Java, який визначає методи, з допомогою яких програмне забезпечення на Java здійснює доступ до бази даних.

Cache – проміжний буфер з швидким доступом, що містить інформацію, яка може бути запрошена з найбільшою ймовірністю.

БД – база даних.

API (application programming interface) – програмний інтерфейс.

SQL - мова структурованих запитів для роботи з реляційними базами даних, яка дозволяє створювати, оновлювати, видаляти та отримувати дані.

МП – мова програмування.

API (application programming interface) – набір правил або інтерфейс, які дозволяють застосункам комунікувати між собою.

CoD (curse of dimensionality) – проблема різкого збільшення розмірів Skyline-вибірок при роботі з багатовимірними даними.

ПЗ – програмне забезпечення.

ВСТУП

У сучасному світі часто виникає потреба в обробці великих обсягів багатовимірних даних — коли об’єкти характеризуються одночасно кількома критеріями (наприклад: ціна, якість, час доставки, відстань, витрати тощо). При цьому специфіка даних чи домену така, що користувач чи система не може задати чіткі вагові коефіцієнти для кожного критерію, але бажає отримати сукупність «оптимальних» варіантів — таких, які не є однозначно гіршими за інші за всіма критеріями. Саме в цьому контексті значущим стає поняття Парето фронту (або вибірки Skyline) — множини об’єктів, які не є домінованими жодним іншим об’єктом у множині за всіма розглянутими критеріями.

Важливість даного методу зростає разом з об’ємами даних, які користувачу доводиться обробляти самостійно. Наприклад, у домені електронної комерції, де запити часто можуть повертати сотні або тисячі результатів, а звичайної фільтрації не достатньо для формування невеликого набору репрезентативних даних, Skyline вибірка може стати незамінним інструментом, який дозволить користувачеві «відсікати» ті пропозиції, які для нього не є релевантними. Аналогічні ситуації можна часто зустріти і у інших доменах, де людина стикається з необхідністю обрати найкращі варіанти за рівноцінними критеріями: логістика, фінансова аналітика, управління ресурсами, геоінформаційні системи, тощо. У цих системах алгоритм формування Парето-фронту може не тільки спростити задачу для користувача, але й зменшити вірогідність помилок, спричинених «людським фактором».

На сьогодні в науковій літературі є чимало матеріалів, де розглядається тема Skyline-запитів. Зокрема це різні алгоритми їх формування та аналіз

ефективності цих алгоритмів, способи обмеження розмірів кінцевої вибірки при роботі з великою кількістю критеріїв, а також застосування до просторових, невизначених чи потокових даних.

Практичних розробок у даній сфері також немало. Для мов програмування Python та R існують програмні засоби та бібліотеки, що надають функціонал формування Skyline-вибірок з наборів даних. Серед баз даних та систем обробки даних також є ті, що надають такий функціонал: Exasol має спеціальний оператор PREFERRING, D-Sight та Apache Spark мають додаткові інтеграції для цього. Існує також багато некомерційних рішень на C++ з відкритим кодом.

Однак для мови програмування Java — однієї з найпопулярніших технологій для розробки веб і аналітичних систем — не існує універсальної, добре документованої бібліотеки, яка дозволяла б формувати вибірки Парето-фронту (Skyline) із багатовимірних даних просто інтегруючи її у Java-застосунок. Це створює технологічну прогалину: розробники, які працюють із Java-екосистемою, змушені або реалізовувати алгоритми самостійно, або шукати вузькоспеціалізовані рішення, що не враховують прикладних потреб, таких як високонавантажена веб-платформа з інтеграцією з Spring, підтримкою потокової обробки чи кешування.

Більше того, у існуючих рішеннях часто залишається невирішеною проблема обмеження розміру Парето-фронту, отриманого з багатовимірних даних. Специфіка роботи Skyline-запитів полягає у тому, що чим більше критеріїв потрібно брати до уваги, тим більшою стає кінцева вибірка. Це може призвести до випадків, коли алгоритму вдається «відсікти» лише незначну частину кандидатів, що робить його недоцільним.

Отже, тема формування Skyline-вибірок з багатовимірних даних та розроблення бібліотеки з таким функціоналом для Java є доволі актуальною. Саме їй і присвячено дану роботу.

Об’єктом дослідження є алгоритми та програмне забезпечення для формування Skyline-вибірок.

Предметом дослідження виступають підходи, методи та засоби розроблення програмного забезпечення для формування Skyline-вибірок, зокрема їх реалізація у Java-середовищі з оптимізацією продуктивності.

Мета роботи — забезпечити можливість формування вибірки Парето-фронту для великих обсягів багатовимірних даних у мові програмування Java.

Завдання дослідження полягає у розробці універсальної бібліотеки, яку можна інтегрувати у корпоративні додатки для формування Skyline-вибірок та багатокритеріальної фільтрації.

Задачі, що вирішуються:

- аналіз існуючих методів багатокритеріального прийняття рішень;
- дослідження проблем масштабованості для великих обсягів даних при формуванні Skyline-вибірок;
- дослідження методів формування Skyline-вибірок з багатовимірних даних;
- дослідження кластеризації як способу вирішення проблеми «прокляття розмірності» (CoD);
- оцінка ефективності запропонованого рішення.

Наукова новизна даного дослідження полягає у подальшому розвитку методів формування Skyline-вибірок з об’ємних наборів багатовимірних даних, а також розробку першого програмного засобу формування Skyline-вибірок на мові програмування Java.

Практичне значення дослідження полягає у розробці програмного засобу для вирішення задач формування Skyline-вибірок та багатокритеріальної фільтрації, який може бути інтегрований у прикладні системи, розроблені мовою програмування Java. Це забезпечує можливість використання Skyline-відбору в інформаційних системах, де обробляються великі обсяги даних, наприклад у системах рекомендацій, фінансовому скорингу чи пошукових сервісах.

1. ОГЛЯД АЛГОРИТМІВ ТА МЕТОДІВ ФОРМУВАННЯ SKYLINE-ВИБІРОК

1.1 Опис предметної області

У сучасних інформаційних системах користувачі часто стикаються з проблемою прийняття рішень на основі декількох суперечливих критеріїв. Наприклад, при виборі готелю можна враховувати як ціну, так і відстань до центру міста. При цьому, немає єдиної функції оцінювання важливості критеріїв, яка дозволила б звести їх значення до єдиного. У таких випадках рідко буває єдиний варіант, який є оптимальним за всіма параметрами. Замість цього користувачі зазвичай цікавляться підмножиною альтернатив, які представляють найкращий компроміс між розглянутими критеріями.

Запити Skyline надають формальний механізм для вилучення таких підмножин безпосередньо з набору даних. Вони ідентифікують усі об'єкти, над якими не домінує жоден інший об'єкт з урахуванням визначеного набору атрибутів.

Поняття домінантності полягає у наступному: нехай D — множина з n об'єктів даних, де кожен об'єкт $p \in D$ представлений як d-вимірний вектор числових атрибутів: $p=(P_1,P_2,\ldots,P_d)$. Припускаючи, що для кожного атрибуту кращі менші значення, об'єкт p домінує над іншим об'єктом q, що позначається $p \prec q$, якщо і тільки якщо виконуються дві наступні умови:

$$\begin{cases} \forall i \in \{1,2,\ldots,d\} : p_i \leq q_i \\ \exists j \in \{1,2,\ldots,d\} : p_j < q_j \end{cases} \quad (1)$$

Іншими словами, p є принаймні таким же хорошим, як q у всіх вимірах, і строго кращим принаймні в одному вимірі. Відношення домінування визначає частковий порядок над набором даних, оскільки не всі пари об'єктів обов'язково є порівнянними[4].

З огляду на це, Skyline-вибірка набору даних D — це набір усіх об'єктів, які не домінують над жодним іншим об'єктом у D: $Sky(D) = \{p \in D \mid \nexists\, q \in D: q \prec p\}$. Цей набір представляє всі Парето-оптимальні рішення, тобто об'єкти, для яких не існує кращої альтернативи одночасно за всіма атрибутами.

Для прикладу, розглянемо набір даних про готелі, описаний двома атрибутами: відстань до центру міста та ціна за ніч.

Таблиця 1.1 – Приклад набору даних готелів

Готель	Відстань до центру міста (км)	Ціна за ніч (грн)
А	5	1000
Б	4	1200
В	6	900
Г	7	1100

Нижчі значення є кращими як для відстані, так і для ціни. Застосовуючи поняття домінантності стає очевидно, що готель В домінує над готелем Г, оскільки В є ближчим і дешевшим. Жоден інший готель не є домінованим. Таким чином, Skyline-вибірка цього набору даних є наступною: Sky(D)={А,Б,В}. Кожен із цих готелів представляє окремий компроміс між

відстанню та ціною і може вважатися оптимальним залежно від різних уподобань користувачів.

Запити Skyline особливо корисні, коли користувач не може або не бажає визначати конкретну функцію зважування або оцінювання для поєднання декількох критеріїв. Натомість вони надають набір оптимальних варіантів, з яких можна прийняти остаточне рішення вручну або за допомогою додаткових критеріїв ранжування.

Ця властивість робить запити Skyline цінними в таких сферах, як електронна комерція, системи рекомендацій, підтримка фінансових рішень та кредитний рейтинг, де різні зацікавлені сторони можуть присвоювати різні пріоритети таким критеріям, як вартість, ризик або продуктивність. У програмних системах Skyline-вибірки дозволяють здійснювати багатокритеріальну фільтрацію великих наборів даних перед виконанням більш спеціалізованого аналізу або ранжування.

Прямий (наївний) підхід до обчислення Skyline-вибірки передбачає порівняння кожного об'єкта з кожним іншим об'єктом для перевірки домінування, що вимагає $O(n^2)$ порівнянь. Однак цей підхід є непрактичним для великих наборів даних або багатовимірних просторів. Для поліпшення масштабованості було запропоновано різні оптимізовані алгоритми, зокрема:

- Block Nested Loop (BNL) – ітеративний підхід, що підтримує набір Парето-оптимальних кандидатів під час сканування набору даних[8];
- Divide-and-Conquer – розділяє дані на піднабори, обчислює локальні горизонти та об'єднує їх[8];

- Методи на основі індексів – використовують багатовимірні індекси, такі як R-дерева, для обрізання домінуючих областей простору пошуку[8];
- Потокові та розподілені алгоритми горизонту – обробляють запити горизонту над динамічними або розділеними джерелами даних[1].

Основною проблемою в обчисленні горизонту є «прокляття розмірності»: із збільшенням кількості вимірів зменшується ймовірність домінування одного об'єкта над іншим[4]. Як наслідок, розмір горизонту часто зростає експоненціально з розмірністю, що зменшує його практичну корисність у об’ємних наборах даних.

1.2 Аналіз існуючих рішень

Для ефективного аналізу існуючих методів та засобів формування Skyline-вибірок в сфері інженерії програмного забезпечення, необхідно сформулювати ключові дослідницькі питання. Ці питання допоможуть структурувати дослідження та визначити основні аспекти, на яких слід зосередити увагу. Список дослідницьких питань представлено в таблиці 1.2.

Таблиця 1.2 – Дослідницькі питання

RQ1	Які є методи та моделі обчислення Skyline-вибірок в інженерії програмного забезпечення?
RQ2	Які є програмні засоби та інструменти обчислення Skyline-вибірок в інженерії програмного забезпечення?

З дослідницьких питань було сформовано список пошукових запитів, які наведені в таблиці 1.3.

Таблиця 1.3 – Пошукові запити

1	«Skyline query» OR «Skyline selection» OR «Skyline tool» OR «Skyline framework»
2	«R-Skyline» OR «k-Regret» OR «BNL» OR «Restricted Skyline»
3	«Pareto-optimal front» OR «Pareto-optimal selection»
4	(пошуковий запит 1) AND («parallel computing» OR «spatial query»)

Для пошуку було використано різні наукові бази даних та пошукові системи: IEEE Xplore, Google Scholar, Springer Link, Academia.edu.

Щоб уникнути включення нерелевантних досліджень було розроблено критерії виключення публікацій та рішень.

Таблиця 1.4 – Критерії виключення

EC1	Дослідження, до яких немає відкритого доступу.
EC2	Дослідження, опубліковані не англійською або українською мовами.
EC3	Застарілі дослідження (опубліковані до 2014 року).
EC4	Публікація містить резюме семінару, конференції.
EC5	Дослідження описує алгоритми багатокритеріальної оптимізації.
EC6	Публікація містить інструкцію з використання програмного забезпечення без опису функціонування релевантних модулів.

Результати пошуку наведені у таблиці 1.5.

Таблиця 1.5 – Джерело та кількість знайдених робіт

IEEE Xplore	51
Google Scholar	37
Springer Link	16
Academia.edu	22
Не наукові	12
Всього	138

Результат застосування критеріїв виключення наведено у таблиці 1.6.

Таблиця 1.6 – Виключені джерела

EC1	Дослідження, до яких немає відкритого доступу.	72
EC2	Дослідження, опубліковані не англійською або українською мовами.	2
EC3	Застарілі дослідження (опубліковані до 2007 року).	16
EC4	Публікація містить резюме семінару, конференції.	14
EC5	Дослідження описує алгоритми багатокритеріальної оптимізації.	20
EC6	Публікація містить інструкцію з використання програмного забезпечення без опису функціонування релевантних модулів.	3
	Всього	124

Таким чином, залишилось 14 робіт для проведення аналізу.

RQ1. Які є методи та моделі обчислення Skyline-вибірок в інженерії програмного забезпечення?

Block-Nested-Loop (BNL). Класичний алгоритм BNL послідовно зчитує набір даних і зберігає вікно поточних кандидатів горизонту в пам'яті [2]. Для кожної нової точки p він порівнює p з точками у вікні: якщо якийсь кандидат домінує над p, p відкидається; якщо p домінує над деякими точками вікна, вони видаляються; в іншому випадку p додається. BNL є простим і працює в будь-якій розмірності, але його вікно може переповнити пам'ять, якщо накопичиться занадто багато кандидатів. У цьому випадку воно переливається на диск і може вимагати декількох проходів. Його головними перевагами є застосовність і простота, але недоліком є високе використання пам'яті[8].

Sort-First Skyline (SFS): SFS спочатку сортує весь набір даних за монотонною функцією оцінки (наприклад, сумою нормалізованих атрибутів), щоб ймовірні точки горизонту були виявлені на ранній стадії[3, 5]. Обробляючи дані в сортованому порядку, SFS часто може швидко відкинути багато точок, оскільки точка з високою оцінкою може домінувати над багатьма наступними точками. На практиці SFS значно зменшує кількість потенційних точок. Однак він повинен сортувати всі дані, що є дорогим процесом, і в найгіршому випадку навіть найкраща точка горизонту може з'явитися в кінці сортування (якщо функція оцінки є недосконалою), тому все одно необхідне повне сканування.

LESS (лінійне сортування для горизонту): LESS поєднує ідеї SFS і BNL для підвищення ефективності. Він виконує зовнішнє сортування даних, але вставляє фільтр горизонту: під час початкових проходів генерації пробігів він зберігає невелике «вікно виключення» найкращих кандидатів (як вікно BNL), щоб рано відкинути домінуючі записи[6]. Потім він об'єднує ці прогони, ефективно поєднуючи остаточний прохід сортування з одним проходом фільтра Skyline. Фактично, LESS досягає всіх переваг SFS у скороченні без його головного недоліку. Експерименти показують, що LESS перевершує звичайний SFS у більшості випадків, заощаджуючи прохід сортування та обрізаючи записи на льоту. LESS вимагає додаткового буферного простору для вікна елімінації, але загалом дає менший проміжний результат.

SaLSa (Sort and Limit Skyline Algorithm): SaLSa також попередньо сортує дані за монотонною функцією, але обмежує кількість точок, які повністю перевіряються[8]. Після сортування він знаходить точки горизонту в порядку і часто може зупинитися раніше: багато точок в кінці відсортованого списку не можуть увійти в горизонт, оскільки вони вже домінують над попередніми точками. Таким чином, SaLSa обмежує кількість порівнянь домінування і читання даних. Його перевага полягає в тому, що він може значно скоротити роботу над «простими» випадками; він зберігає простоту управління вікнами SFS, але може пропускати перевірку багатьох кортежів. Недоліком є те, що його ефективність залежить від наявності хорошого порядку сортування та розподілу даних.

Бітова карта. Метод бітової карти горизонту кодує кожну точку як бітову карту її атрибутів і використовує бітові операції для перевірки домінування[9]. Він може швидко ідентифікувати і вивести перші кілька точок горизонту (оскільки він, по суті, виконує багато тестів домінування паралельно), але в цілому є дорогим: він перевіряє кожну точку і використовує великі бітові карти, що робить його ресурсоємним і непридатним для високих вимірів або динамічних даних.

Інші розумні методи включають сканування на основі індексів, яке використовує B-дерева або спеціалізовані індекси для кожного виміру (наприклад, R-дерева) для одночасного виключення великих областей[10]. Наприклад, метод «гілка і межа» (BBS) використовує R-дерево для обходу просторових розділів у порядку «найкращий спочатку», обрізаючи цілі гілки, обмежувальні рамки яких домінують. Такі методи на основі індексів можуть бути дуже швидкими та прогресивними (швидко повертаючи точки горизонту), але часто погіршуються із зростанням розмірності.

Існують також розробки, які поєднують методи на основі дерев та паралельне виконання на GPU. Наприклад, алгоритм SkyCell виконує поділ простору рішень на ієрархічні шари сітки[7]. Це, як у випадку з алгоритмами на основі дерев, дозволяє відсікати області рішень, а не окремі рішення. Більше того, даний підхід добре підлаштовується під паралелізацію, що дозволяє значно підвищити його швидкодію.

RQ2. Які є програмні засоби та інструменти обчислення Skyline-вибірок в інженерії програмного забезпечення?

Більшість баз даних SQL не мають вбудованого оператора SKYLINE, тому запити Skyline необхідно переписувати за допомогою підзоду anti-join або власного коду. Наприклад, PostgreSQL не має вбудованого Skyline, хоча використання функціональних індексів може імітувати пошук top-k, подібний до Skyline[14].

Комерційна база даних Exasol пропонує вбудоване розширення skyline: оператор SQL PREFERRING дозволяє користувачам вказати «високі/низькі» переваги для стовпців і повертає набір, оптимальний за Парето[12]. На відміну від цього, основні реляційні бази даних (Oracle, SQL Server, MySQL тощо) не мають вбудованої підтримки Skyline.

Для платформ великих даних дослідні прототипи інтегрують Skyline в такі двигуни, як Spark SQL[11]. На практиці Skyline часто обчислюється в мовах обробки даних. Наприклад, існують бібліотеки Python (наприклад, paretoset на PyPI)[13] і скрипти R, які реалізують Skyline (часто використовуючи вкладені цикли або векторні фільтри). Однак, як зазначають аналітики, «не існує канонічного способу зробити це» в бібліотеках, таких як Pandas або SQL, тому потрібні спеціальні коди або сторонні інструменти. Деякі інструменти підтримки прийняття рішень включають фільтри Skyline/Pareto, а спеціалізовані механізми запитів (або розширення) були створені в рамках академічних проектів.

Сучасні робочі навантаження вимагають масштабованих реалізацій. Алгоритми Skyline були паралелізовані за допомогою MapReduce (наприклад, розділення даних, обчислення локальних горизонтів, а потім об'єднання) і реалізовані на двигунах big data. Наприклад, SkylineMR[11] адаптує BNL/SFS до Hadoop/MapReduce. Аналогічно, були запропоновані алгоритми на основі GPU (наприклад, використання CUDA для паралельного тестування багатьох порівнянь домінування)[7]. Нещодавно такі фреймворки, як Apache Spark, почали підтримувати запити горизонту інтегрували нативний оператор Skyline в Spark SQL, продемонструвавши набагато кращу продуктивність, ніж переписані вручну SQL-коди[11].

1.3 Аналіз літератури

Дослідження показало, що тема Skyline-запитів залишається актуальною, а розробки, пов’язані з нею, продовжують з’являтись як у наукових джерелах, так і у комерційних рішеннях. Також було ідентифіковано основні складнощі та проблеми при побудові Skyline-вибірок: зі збільшенням кількості вимірів (критеріїв) розмір горизонту може різко зрости. Це називають «прокляттям розмірності», або CoD. Воно робить результати занадто великими, щоб бути корисними. Більше того, наївне обчислення горизонту (наприклад, подвійний вкладений цикл, що перевіряє всі пари) дуже повільне для великих даних. Тому були розроблені спеціалізовані алгоритми та структури даних для поліпшення продуктивності, масштабованості та обробки потокових або невизначених данихen.wikipedia.orgajbasweb.com. Переваги та недоліки основних сімейств алгоритмів наведено у таблиці 1.7

Таблиця 1.7 – Переваги та недоліки основних сімейств алгоритмів Skyline

Алгоритм	Переваги	Недоліки
BNL	Простий, здатен працювати без індексації.	Високе використання пам'яті, багаторазові проходження по даним.
На основі сортування	Можуть значно скоротити кількість кандидатів за допомогою попереднього сортування, що часто прискорює час виконання.	Сортування може бути дороговартісним; ефективність значно залежить від функції сортування.
На основі індексу	Використовує просторові індекси для скорочення цілих регіонів	Продуктивність погіршується при великій кількості вимірів або асиметричних даних; створення та підтримка індексів.
На основі прогресивної обробки	Швидко знаходить ранні результати (добре для взаємодії з користувачем); може зупинитися, коли знайдено «достатньо» точок.	Страждає при роботі з багатовимірними даними, надлишковий ввід-вивід, часто вимагає дослідження великого простору пошуку.

Закінчення таблиці 1.7

Алгоритм	Переваги	Недоліки
На основі паралельної обробки	Масштабування та швидкодія.	Накладні витрати на координацію виконання.

Ці алгоритми спрямовані на вирішення проблем масштабованості запитів Skyline: вони зменшують кількість тестів домінування, рано обрізають дані або використовують паралелізм. Але жоден з них не долає комбінаторний вибух розміру Skyline у високих вимірах повністю – це залишається активною проблемою досліджень. Більше того, дані алгоритми не є тривіальними у імплементації та адаптації, а отже розробник, який хоче ввести функціонал формування Skyline-вибірок в свій застосунок, мусить витратити чимало часу на дослідження та написання такого рішення, або ж користуватись існуючими інструментами.

Проте існуючі програмні засоби часто є досить нішевими. Більшість орієнтовані на роботу в домені big data (Apache Spark, Hadoop), вимагають написання складних запитів для роботи (PostgreSQL), або ж є комерційними розробками (Exasol). Існуючі інструменти в МП Python та R є досить ефективними, проте не здатні інтегруватись у застосунки, написані на інших МП. Це залишає досить велику прогалину в можливостях використання Skyline. Більше того, як і у випадку з алгоритмами, багато програмних засобів також не до кінця вирішують проблему CoD.

1.4 Постановка задачі

Метою роботи є забезпечення можливості формування Skyline-вибірки для великих об’ємів багатовимірних даних у Java. Для досягнення мети необхідно:

- провести аналіз існуючих методів формування Skyline-вибірок;
- дослідити проблему масштабованості для великих обсягів даних при формуванні Skyline-вибірок;
- дослідити методи вирішення проблеми CoD;
- на основі проведених досліджень розробити метод формування Skyline-вибірок, який вирішує або мінімізує проблеми CoD та масштабованості;
- створити бібліотеку на основі розробленого методу, що підтримує інтеграцію з рішеннями, розробленими на МП Java, а також вирішує основні проблеми формування Skyline-вибірок: CoD та низьку швидкодію;
- розробити механізм обробки великих об’ємів даних;
- провести оцінку ефективності запропонованого рішення.

Створений програмний засіб повинен відповідати наступним вимогам:

- мати зручний та зрозумілий програмний інтерфейс для інтеграції;
- підтримувати можливість розширення з боку користувача;
- повинен бути надійним та передбачуваним у використанні;

Створений метод повинен відповідати наступним вимогам:

- надавати можливість обробляти різні інтерфейси вводу даних у МП Java;
- вирішує проблему CoD або уникає критичних помилок, спричинених нею;
- швидко та ефективно формує Skyline-вибірки.

Висновки до розділу

У першому розділі було розглянуто предметну область Skyline. Було сформульовано дослідницькі питання, які допомогли структурувати пошук літератури та визначити основні аспекти, на яких слід зосередити увагу. На основі дослідницьких питань розроблено пошукові запити для різних наукових баз даних та пошукових систем. Також було встановлено критерії виключення публікацій для забезпечення релевантності та якості відібраних джерел.

В ході аналізу були описано розвиток основних сімейств алгоритмів та методів формування Skyline-вибірок, а також програмного забезпечення, що надає функціонал Skyline-запитів.

Дослідження літератури показало, що протягом останнього десятиліття обробка запитів до горизонту зосереджувалася на ефективному обрізанні (за допомогою сортування, індексування, розділення) та масштабуванні до великих даних (за допомогою паралельних або апаратних методів прискорення). Популярні алгоритми включають BNL, методи сортування-фільтрування (SFS, LESS, SaLSa), R-tree/BBS та різні адаптації MapReduce. Кожен з них має свої переваги та недоліки: BNL/LSF прості, але дорогі; методи на основі індексації та розділення добре обрізають, але вимагають додаткових витрат. Незважаючи на вдосконалення, висока розмірність і дуже великі набори даних залишаються складними проблемами, однозначного рішення яких досі не було знайдено. Декілька баз даних і фреймворків зараз пропонують вбудовані (PREFERRING від Exasol), або інтегровані можливості Skyline-запитів, що відображає практичну цінність Skyline в аналітиці. Проте ці засоби є нішевими і не покривають потребу в універсалізації даного функціоналу.

Отже, тематика Skyline-запитів залишається актуальною як зі сторони невирішених технічних проблем, так і зі сторони відсутності можливості їх використання у багатьох доменах.

2. РОЗРОБКА УДОСКОНАЛЕНОГО МЕТОДУ ФОРМУВАННЯ SKYLINE-ВИБІРОК

2.1 Алгоритми формування Skyline-вибірок

За результатами проведеного у розділі 1.2 дослідження літератури було виокремлено 3 алгоритми Skyline, які дозволять обробляти дані з різних інтерфейсів вводу даних у МП Java, забезпечуючи при цьому швидкодію, ефективне використання ресурсів та надійність:

- Parallel BNL;
- SkyCell;
- Randomized Multi-Pass Skyline.

У наступних розділах наведено їх опис та особливості.

2.1.1 Алгоритм BNL

BNL працює шляхом виконання парних порівнянь домінування між об'єктами даних, зберігаючи при цьому вікно W поточних кандидатів горизонту в основній пам'яті. Алгоритм послідовно сканує набір даних, розглядаючи кожен кортеж p по черзі та порівнюючи його з тими, які в даний момент зберігаються в W. При порівнянні може бути 3 випадки:

а) W містить хоча б один кортеж $q \in W$ такий, що $q \prec p$;

б) W містить один або кілька кортежів $q \in W$ таких, що $p \prec q$;

в) W не містить жодного кортежу $q \in W$ такого, що $p \prec q$ або $q \prec p$.

У випадку а, якщо існує кортеж $q \in W$, такий, що q домінує над p (тобто $q_i \leq p_i$ для всіх вимірів *i* та $q_j < p_j$ для принаймні одного виміру *j*), то p не може належати до вибірки Skyline. У цьому випадку p негайно відкидається.

У випадку б, якщо p домінує над одним або декількома кортежами в W, ці доміновані кортежі видаляються з W, оскільки вони більше не можуть бути частиною результату Skyline.

У випадку в, якщо жоден кортеж в W не домінує над p, і p не домінує над жодним кортежем в W, він вставляється в W як нова потенційна точка Skyline.

Після завершення сканування набору даних усі кортежі, що залишилися в W, утворюють набір Skyline. Якщо розмір вікна перевищує доступну пам'ять, BNL розділяє набір даних на розділи і виконує кілька проходів, порівнюючи проміжні горизонти, доки не буде отримано кінцевий результат.

Завдяки своїй простоті, алгоритм можна легко пристосувати до роботи як з потоковими, так і зі сталими даними, а також використати багатопоточність для підвищення швидкодії. Псевдокод такої модифікованої версії BNL наведено на рисунку 2.1

Найгірший випадок часової складності класичного BNL — $O(n^2)$, оскільки кожна точка може порівнюватися з усіма іншими. Однак на практиці такий випадок є доволі малоймовірним, особливо для кортежів з малою кількістю ознак або некорельованих наборів даних. Для версії алгоритму, яка використовує багатопоточну обробку, складність залежить від кількості потоків.

Нехай n позначає загальну кількість кортежів, а p — кількість розділів (потоків). Тоді:

- на фазі локального горизонту (computeLocalSkyline) кожна роздільна частина обробляє n/p кортежів незалежно, з часом $O((n/p)^2)$ на роздільну частину в найгіршому випадку.
- на фазі об'єднання (flatten) складність для локальних горизонтів розміром k є $O(k^2)$.

```
Вхідні дані:
    D — набір кортежів.
Вихідні дані:
    SKYLINE(D)

procedure PARALLEL_BNL(D):
    // Крок 1 — обчислити проміжні Skyline паралельно
    LOCAL_SKYLINES ← D.partition().parallelStream()
                      .map(partition → computeLocalSkyline(partition))
                      .collect(toList())

    // Крок 2 — об'єднати проміжні Skyline
    MERGED ← flatten(LOCAL_SKYLINES)

    // Крок 3: обчислити загальну Skyline з проміжних
    GLOBAL_SKYLINE ← computeLocalSkyline(MERGED)

    return GLOBAL_SKYLINE

procedure computeLocalSkyline(S):
    WINDOW ← ∅
    for each tuple p in S:
        dominated ← false
        for each tuple q in WINDOW:
            if dominates(q, p):
                dominated ← true
                break
            else if dominates(p, q):
                remove q from WINDOW
        if not dominated:
            add p to WINDOW
    return WINDOW

procedure dominates(a, b):
    return (∀i: a[i] ≤ b[i]) ∧ (∃j: a[j] < b[j])
```

Рисунок 2.1 – Псевдокод багатопотокового BNL алгоритму

Отже, загальна складність: $O(n^2/p+k^2)$, що наближається до майже лінійного прискорення, коли горизонти невеликі, а обрізка домінування є ефективною.

Вимоги алгоритму до пам'яті залежать від кількості проміжних кандидатів у горизонт, які необхідно зберегти. Коли розмір горизонту невеликий, BNL може ефективно виконуватися за один прохід.

2.1.2 Алгоритм SkyCell

SkyCell відходить від традиційного підходу формування Skyline-вибірок, розділяючи простір даних (а не тільки точки даних) на клітинки сітки, а потім видаляючи цілі клітинки, точки яких не можуть впливати на горизонт. Таке видалення на основі клітинок значно зменшує кількість перевірок домінування, забезпечуючи масштабованість, яка є більш незалежною від кількості точок даних і більш залежною від розмірності даних та деталізації сітки.[16]

Крім того, SkyCell розроблений для паралельної обробки: клітинки-кандидати можуть оброблятися паралельно (наприклад, на GPU), оскільки вони є незалежними.

Алгоритм працює за наступним принципом: нехай $P=\{p_1,p_2,\ldots,p_n\}$ — множина точок у d-вимірному евклідовому просторі. SkyCell вводить додаткові конструкції: простір даних нормалізується до [0,1) і розділяється за допомогою багатошарової сіткової структури. На шарі *i* сітка розділяє кожну розмірність на 2^i сегментів, утворюючи $2^{i,d}$ комірок. Точка даних відображається на конкретну комірку за допомогою її координат.

Для визначення домінування комірок використовуються їх координати, а також вводиться правило строгої (або повної) домінації комірки: $c_i \prec c_j$ якщо $c_i \neq \emptyset$ і в кожному вимірі k: $c_i[k] < c_j[k]$.

Результатом є те, що кількість комірок-кандидатів у заданому шарі обмежена з точки зору гранулярності сітки та розмірності даних і не залежить від кількості точок даних.

Основні поняття

Шар ρ. Простір даних розділений на послідовність шарів (рівнів) із зростаючою грануляцією. Рівень *i* розділяє кожен вимір на 2^i рівних сегментів. Верхній рівень ρ вибирається так, щоб кожна кінцева комірка містила невелику кількість точок (або деякий критерій зупинки).

Ключова клітинка — це непорожня клітинка, яка не є строго чи нестрого домінованою.

Множина клітинок-кандидатів у шарі — це об'єднання всіх ключових клітинок та клітинок, над якими вони частково домінують. Клітинки-кандидати — єдині клітинки, які можуть містити точки горизонту.

SkyCell обчислює клітинки-кандидати зі збільшенням їх грануляції на кожному шарі і зупиняється, коли клітинки-кандидати стають достатньо малими, щоб дозволити звичайне обчислення Skyline на рівні точок всередині них. Покрокове виконання алгоритму виглядає так:

- побудувати шари сітки L_0, L_1, …, L_ρ від великих (див. Layer 0 на рисунку 2.2) до дрібних. Призначити кожній точці клітинку шару ρ; зберегти точки так, щоб точки з однієї клітини займали послідовний сегмент масиву;
- для кожного шару i=0 до ρ−1 обчислити набір R_{i+1} комірок, що представляють інтерес у шарі i+1. Для цього шляхом виконати грануляцію (shrinking) ключових комірок з Ri;
- відсікти клітинки, які є порожніми або не належать до множини клітинок-кандидатів за правилом строгої домінації. Продовжувати грануляцію клітинок;
- на кінцевому шарі ρ зібрати усі клітини-кандидати Cρ. Оскільки

точки в різних клітинах-кандидатах не можуть домінувати одна над одною, обчислити Skyline окремо всередині кожної клітини-кандидата та об’єднати їх.

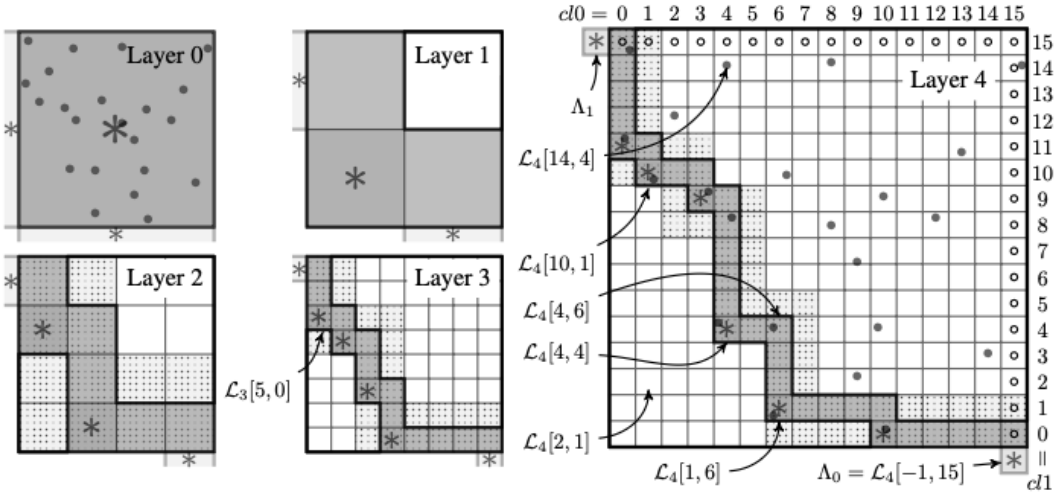


Рисунок 2.2 – Приклад багатошарового поділу простору рішень[16]

Псевдокод алгоритму наведено на рисунку 2.3

```
Вхідні дані:
    D – набір кортежів.
    ρ – кількість рівнів грануляції.
Вихідні дані:
    SKYLINE(D)

Procedure SkyCell(points, ρ):
    // ---------- Крок 1: побудувати рівні грануляції ----------
    (layers, cellToPointIndices) ← buildLayerData(points, ρ, d)
    // layers[i] = множина непорожніх комірок на рівні i
    // рівень 0 має лише одну можливу комірку: (0,0,...,0)

    root ← cell with indices [0,0,...,0]
    KC ← empty set                    // поточні ключові комірки

    if root ∈ layers[0]:
        KC ← { root }
    else:
        return empty set              // немає жодної точки

    // ---------- Крок 2: грануляція ключових комірок між рівнями ----------
    for i from 0 to ρ − 1:
        nextLayerCells ← layers[i + 1]               // непорожні комірки на рівні i+1
        // розширити кожну ключову комірку її нащадками та залишити тільки
        // непорожніх, недомінованих нащадків
        KC ← shrinkKeyCells(d, KC, nextLayerCells)

        if KC is empty:
            return empty set

    candidateCells ← KC // Після циклу KC = множина кандидатних комірок на рівні ρ

    // ---------- Крок 3: зібрати кандидатні точки ----------
    candidatePoints ← all points whose level-ρ cell is in candidateCells
                      (using cellToPointIndices map)

    // ---------- Крок 4: точне обчислення skyline для кандидатів ----------
    skylinePoints ← ExactSkyline(candidatePoints)

    return skylinePoints
```

Рисунок 2.3 – Псевдокод алгоритму SkyCell

Важливою перевагою є те, що SkyCell ніколи не перевіряє точки в клітинах, які, як доведено, домінують на більш грубому рівні, тому кількість перевірок домінування на рівні точок значно зменшується.

У першоджерелі[17] описано можливість паралельного обчислення алгоритму на GPU. У МП Java немає можливості використовувати GPU для обчислювань, проте є можливість максималізувати ефективність обчислень на CPU за допомогою віртуальних потоків[15].

Для SkyCell сортування n точок займає O(nlog n) часу. Побудова шарів сітки Lρ до L0 займає O(2ρ·d) часу, де 2ρ·d — кількість комірок у Lρ. Коли точки розподілені рівномірно, ρ дорівнює щонайбільше (log n)/d, з однією точкою на кожну комірку в Lρ.

Потім процедура зменшення клітинок (ShrinkKeyCells) запускається від шару 0 до шару ρ−1. Виконується обхід підмножини клітинок-кандидатів у шарі *i*, щоб згенерувати ключові клітини у шарі *i*+1. Для кожної клітинки запускається процедура коригування, щоб оновити діапазон індексів стовпців, які будуть перелічені, що займає час O(log d). У шарі *i* кількість клітинок-кандидатів дорівнює

$$\sum_{j=0}^{d-1}\left(2^{i}-1\right)^{j}\ 2^{\,i(d-1-j)} \tag{2}$$

Загальна складність алгоритму[16]:

$$\mathcal{O}!\left(\log d\sum_{i=0}^{\rho-1}\sum_{j=0}^{d-1}\left(2^{i}-1\right)\ j\,2^{\,i(d-1-j)}\right)$$

$$=\mathcal{O}!\left(\log d\sum_{i=0}^{\rho-1}2^{\,i\cdot d}\right)\ =\mathcal{O}!\left(2^{\rho\cdot d}\log d\right) \tag{3}$$

2.1.3 Алгоритм Randomized Multi-Pass Skyline

Даний алгоритм є вдосконаленням BNL алгоритму. Він розглядає вхідні дані як впорядкований потік, який можна сканувати кілька разів (багато проходів). Робоча пам'ять обмежена (набагато менша за n), і між проходами алгоритм може зберігати лише свою робочу пам'ять. Мета полягає в тому, щоб мінімізувати як кількість проходів, так і робочий простір.

Алгоритм працює у вигляді послідовності раундів (проходів). У кожному раунді він підтримує набір M кандидатів у точки Skyline. Він зчитує вхідні дані і намагається відфільтрувати їх відносно поточного набору кандидатів, оновлюючи M за допомогою методів випадкової вибірки, так що після O(logn) проходів M дорівнює справжньому горизонту. Ключові особливості:

- у кожному проході алгоритм зберігає O(h) збережених точок і відкидає інші;
- випадкова вибірка використовується для прискорення видалення домінуючих точок;
- алгоритм гарантує, що тільки точки, які можуть опинитися у Skyline, залишаються в наступних раундах.

Опис роботи алгоритму

Алгоритм працює в два етапи, адаптуючи свою поведінку залежно від кількості точок горизонту *h* та гіперпараметра *b* — визначеного користувачем параметр компромісу, що регулює простір у порівнянні з кількістю проходів. Чим більше значення *b*, тим більше простору займає алгоритм і тим менше проходів виконує, і навпаки.

Етап I: Випадок малого Skyline

Якщо $h \leq log_b n$, достатньо використовувати простий метод вилучення горизонту. Цей метод ідентифікує одну точку горизонту за один прохід, вимагаючи максимум $log_b n$ проходів і O($log_b n$) простору.

Етап II: Випадок великої лінії Skyline

Коли h>$\log_b$n, починається основна процедура рандомізації. Кожна ітерація складається з двох проходів по даних.

Перший прохід – вибірка, коли підмножина R_i вибирається з решти некласифікованих точок P_i за допомогою випадкової вибірки. Кожна точка в P_i включається з імовірністю $c * r_i / |P_i|$, де r_i — поточний розмір вибірки, а c — достатньо велика константа.

Другий прохід – обрізка, коли створюється уточнена множина Ri+. Спочатку $R_i \pm R_j$. Для кожної точки p ∈ P_i: якщо p домінує над будь-якою точкою в R_{i+}, вона додається до R_{i+}. Точки в R_{i+}, що домінують над p, видаляються. Отримана R_{i+} містить тільки точки горизонту[18].

Після проходів точки, що домінують над R_{i+}, видаляються з P_i, утворюючи наступну підмножину P_{i+1}. Якщо $|P_{i+1}| \geq |P_i| / b$, ітерація вважається неефективною, і розмір вибірки r_i+1 збільшується в b разів.

В іншому випадку ітерація є ефективною, що вказує на значне скорочення, і розмір вибірки залишається незмінним. Цей ітераційний процес триває доти, доки всі точки не будуть класифіковані.

Псевдокод алгоритму наведено на рисунку 2.4.

```
Вхідні дані:
    D – набір кортежів.
    ρ – кількість рівнів грануляції.
Вихідні дані:
    SKYLINE(D)

Procedure RandomizedMultipassSkyline(D, b, c):
    candidates ← list(points)
    skyline    ← ∅

    // оцінити log_b(n)
    logb_n  ← ceil( log(max(2, n)) / log(b) )

    // ---------- Крок 1: отримати декілька «зернових» точок skyline ----------
    seedLimit ← max(1, logb_n)
    seeds ← SeedSkylineCandidates(candidates, seedLimit)
    skyline ← skyline ∪ seeds
    RemoveFromCandidatesAllDominatedBy(candidates, seeds)

    // ---------- Крок 2: випадковий багаторазовий прохід ----------
    sampleSize ← max(1, c * logb_n)   // r₀ = c · log_b(n)

    while |candidates| ≥ sampleSize:
        prevCount ← |candidates|
        sample ← RandomSampleWithoutReplacement(candidates, sampleSize)     // 1) взяти випадкову вибірку
        sampleSkyline ← ExpandSampleSkyline(sample, candidates)             // 2) розширити skyline вибірку
        skyline ← skyline ∪ sampleSkyline                                   // 3) додати до глобального skyline
        candidates ← UndominatedPoints(candidates, sampleSkyline)           // 4) відкинути доміновані кандидати
        remaining ← |candidates|

        // 5) адаптувати розмір вибірки для наступного проходу
        if remaining = 0:
            sampleSize ← 1
        else if remaining ≥ ceil(prevCount / b):
            sampleSize ← min(remaining, sampleSize * b)             // дані майже не скоротилися → збільшити вибірку
        else:
            sampleSize ← min(sampleSize, max(1, remaining))         // дані суттєво скоротилися → залишити або зменшити вибірку

    // ---------- Крок 3: точний skyline на залишених точках ----------
    skyline ← skyline ∪ IncrementalSkyline(candidates)
    return IncrementalSkyline(skyline)     // фінальне очищення для видалення домінованих точок у 'skyline'
```

Рисунок 2.4 – Псевдокод алгоритму Randomized Multi-Pass Skyline

Аналіз часової та просторової складності

У кожній ітерації алгоритм зберігає:

- поточний набір виявлених точок горизонту M — розмір O(h);
- поточну вибірку R_i — розмір $O(r_i)$;
- похідний набір R_{i+} — також $O(r_i)$.

Максимальний розмір вибірки r_i зростає до O(b·h), Таким чином, загальна просторова складність становить *O(bh)*.

Нехай P_i — множина некласифікованих точок на ітерації *i*. Кожна ітерація складається з двох повних проходів по набору даних. Складність першого проходу становить O(n). На другому проході для кожної точки $p \in P_i$ виконуються перевірки на домінування відносно поточного набору кандидатів R_{i+}. Це вимагає $O(r_i)$ операцій на точку, що дає $O(nr_i)$ часу на прохід.

Оскільки $r_i = O(b \cdot h)$, це призводить до часової складності проходу $O(n \cdot b \cdot h)$. Алгоритм виконує $O(\log_b n)$ проходів, що дає загальну часову складність: $O(n \cdot b \cdot h \cdot \log_b n)$.

2.2 Методи обробки великих об’ємів даних у Java

Основні труднощі в обробці великих об’ємів даних у Skyline – це робота з великою кількістю вхідних даних та великою результуючою вибіркою. Від способу вирішення цих проблем залежить вибір алгоритму Skyline.

Найпопулярнішим підходом до обробки великих масивів даних є пакетна обробка (batch processing). У ній замість обробки мільйонів записів одночасно, дані обробляються невеликими частинами. Її перевагами є:

- запобігання вичерпанню пам'яті шляхом контролю обсягу даних у кожній партії;
- хороша інтеграція з базами даних: JDBC-сумісні джерела даних підтримують пакетні запити;
- передбачувана продуктивність.

Такий підхід в поєднанні з кешуванням проміжних результатів дозволяє використовувати Skyline алгоритми, яким необхідно мати повний набір вхідних даних. Більше того, обробку пакетів можна легко паралелізувати.

Схему роботи пакетного обробника даних для Skyline наведено на рисунку 2.5.

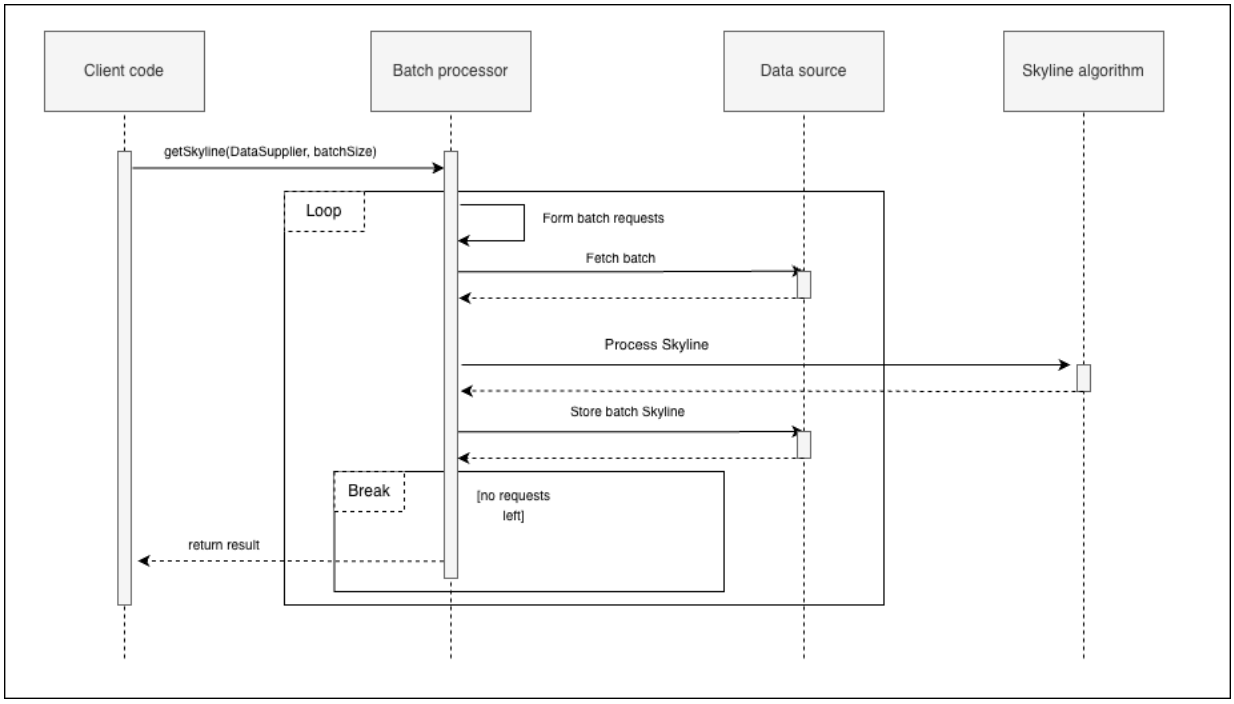


Рисунок 2.5 – Діаграма роботи пакетного обробника даних для Skyline

Як видно з діаграми, клієнтський код відповідає за визначення розміру пакету та запиту до джерела даних. Пакетний обробник ітеративно оброблює дані та зберігає проміжні Skyline-вибірки, після чого виконує з’єднання проміжних Skyline-вибірок таким же способом.

Іншим підходом є обробка вхідних даних через Stream API. Stream API – це інструмент почергової обробки даних у Java, в якому дані вичитуються з джерела у оперативну пам’ять застосунку не одразу, а поступово. Такий підхід «лінивої обробки»[15] даних дозволяє спростити взаємодію з джерелом даних, при цьому не втрачаючи функціонал багатопоточної обробки. Робота з проміжними Skyline-вибірками при цьому залишається пакетною.

Схему роботи обробника даних через Stream API наведено на рисунку 2.6.

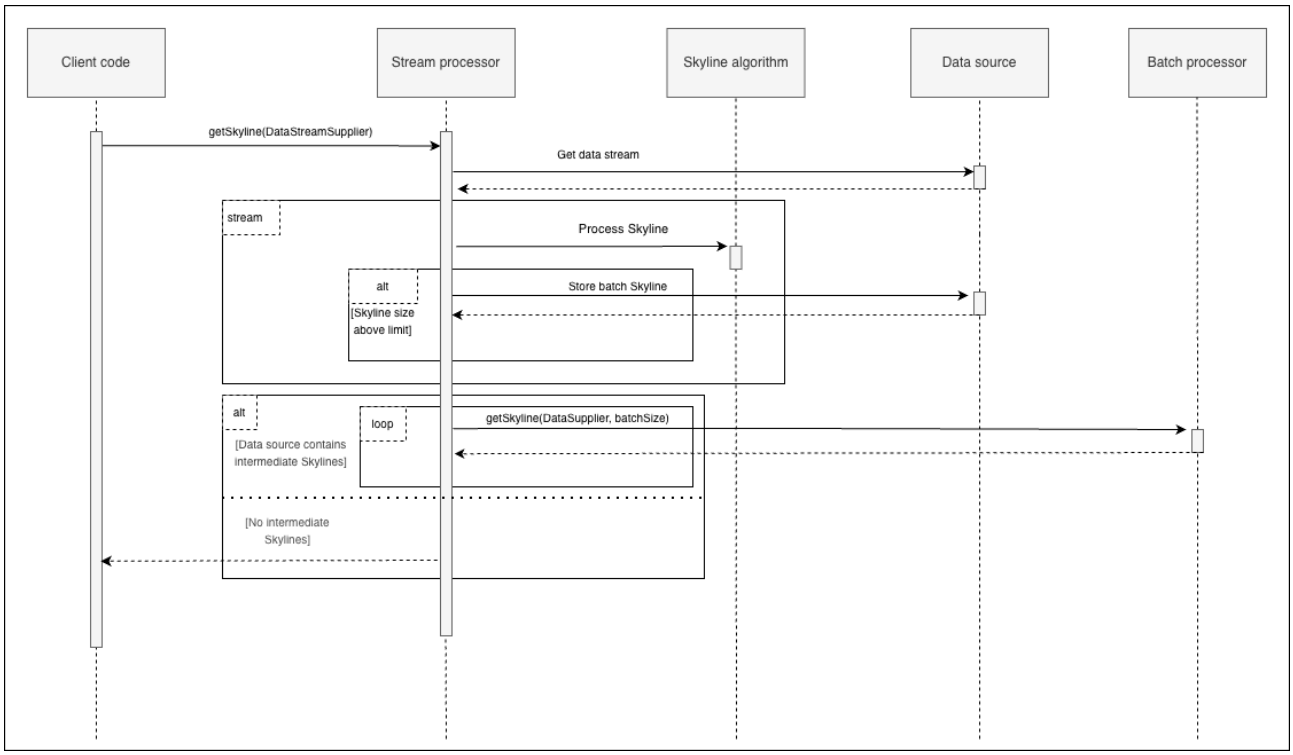


Рисунок 2.6 – Діаграма роботи обробника даних через Stream API

Проте є і недолік – з таким підходом здатен працювати тільки BNL алгоритм. Інші алгоритми можна застосувати лише для з'днання проміжних Skyline-вибірок.

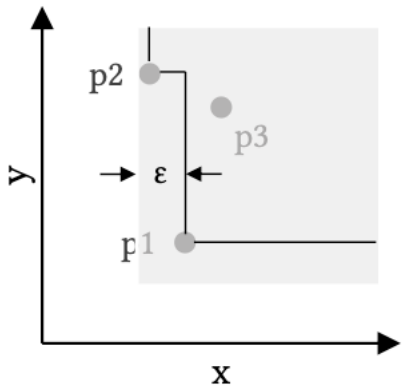
2.3 Методи роботи з багатовимірними даними

При збільшенні кількості критеріїв (вимірів) розміри Skyline-вибірки ростуть за формулою $O((\ln n)^{d-1})$[3]. Для обмеження росту Skyline-вибірки при збільшенні кількості критеріїв (вимірів) існують декілька особливих алгоритмів Skyline. Найпоширеніший з них: ε-Skyline.

2.3.1 ε-Skyline.

У звичайних Skyline алгоритмах кортеж p видаляється з вибірки, якщо вхідні дані містять кортеж q такий, що $q \prec p$. У ε-Skyline є невеликий «допуск» ε в порівнянні — p вважається не домінованим q, якщо жоден вимір q_i не є кращим за p_i більш ніж на ε[3]. Це поняття можна виразити так:

нехай є множина кортежів з d атрибутами і набором ваг $W = \{w! \mid i \in [1, d], 0 < w! \leq 1\}$, і константою $\varepsilon \in [-1,1]$; тоді для двох кортежів q та p з множити $q \prec_\varepsilon p$, якщо $\forall i \in [1, d]$, $q[i] \cdot wi \leq p[i] \cdot wi + \varepsilon$, а також $\exists\, j \in [1, d]$, $q[j] < p[j]$



	x	y
p1	0.4	0.3
p2	0.3	0.8
p3	0.5	0.7

Рисунок 2.7 – Ілюстрація принципу роботи ε-Skyline[16]

На рисунку показано приклад двовимірного набору даних. Темна лінія, що з'єднує точки, зображує оригінальну область домінування горизонту, тоді як світло-сіра область представляє область ε-домінування кортежу p1, коли ε встановлено на 0,1. Цей приклад чітко демонструє, як набір результатів змінюється при зміні області домінування: оригінальний результат горизонту складається із записів p1 і p2, тоді як ε-горизонт повертає тільки запис p1, оскільки p2 ε-домінується p1.

У цього алгоритму є кілька важливих недоліків, що роблять його малоефективним у деяких сценаріях. Перш за все, його можна пристосувати лише до алгоритмів, подібних до SFS-Skyline або Index-Skyline, оскільки оператор ε домінантності не є транзитивним та асиметричним. Іншим недоліком є його неочевидність – розробник не знає точно, який вплив коефіцієнт ε має на розмір кінцевої вибірки, а у середовищах, де дані постійно

змінюються, розробнику доведеться регулярно налаштовувати нові значення ε.

2.3.2 K-Medoids як метод вирішення проблеми CoD

Запропонованим рішенням проблеми CoD є використання кластеризації. У завданнях кластеризації математичні моделі розділяють немарковані дані на групи або кластери на основі схожості між ними. Для використання у задачах формування Skyline найкраще підходить модель K-Medoids. Основна її ідея полягає в тому, щоб представити кожен кластер за допомогою його центроїда — середнього положення всіх точок даних у цьому кластері. Кожна точка даних прив'язується до кластера з найближчим медоїдом, які ітеративно оновлюються, щоб мінімізувати загальну дисперсію всередині кластера. Дисперсія вимірюється за допомогою евклідової відстані для даних, в яких кількість ознак менше 10, та косинусної відстані в іншому випадку. K-Medoids підходить для роботи з великою кількістю ознак в даних, оскільки краще справляється з шумом, може використовувати різні методи обчислення відстаней, а також використовує справжні дані, а не теоретичні точки, як центроїди.

K-Medoids працює за допомогою ітеративної процедури уточнення, що складається з 4 основних етапів.

Ініціалізація: k початкових центроїдів вибираються випадковим чином.

Етап призначення: кожна точка даних призначається кластеру, центроїд якого є найближчим за відстанню. Це створює k груп, де кожне спостереження належить до кластера з мінімальною відстанню до його центроїда.

Етап оновлення: після призначення обчислюються нові центроїди шляхом обчислення середнього значення всіх точок даних у кожному кластері. Ці оновлені центроїди представляють новий центр кожного кластера.

Перевірка збіжності: етапи 2 і 3 повторюються, поки центроїди більше не змінюються істотно або не буде досягнуто заздалегідь визначену кількість ітерацій.

В задачах формування Skyline кластеризація може бути використана для узагальнення результуючої вибірки. Такий підхід дозволяє не тільки зробити розмір Skyline-вибірки регульованим, а й визначати ключові залежності та кореляцію між властивостями кортежів. Коли розмір Skyline перевищує визначений користувачем поріг *x*, знайдений Парето-фронт буде розділено на *x* кластерів. З їх медоїдів буде сформовано новий Skyline, який зберігається у кеш та повертається користувачеві. Такий підхід гарантує, що користувач отримує репрезентативні точки з оригінального простору рішень. Діаграма даного процесу наведена на рисунку 2.8.

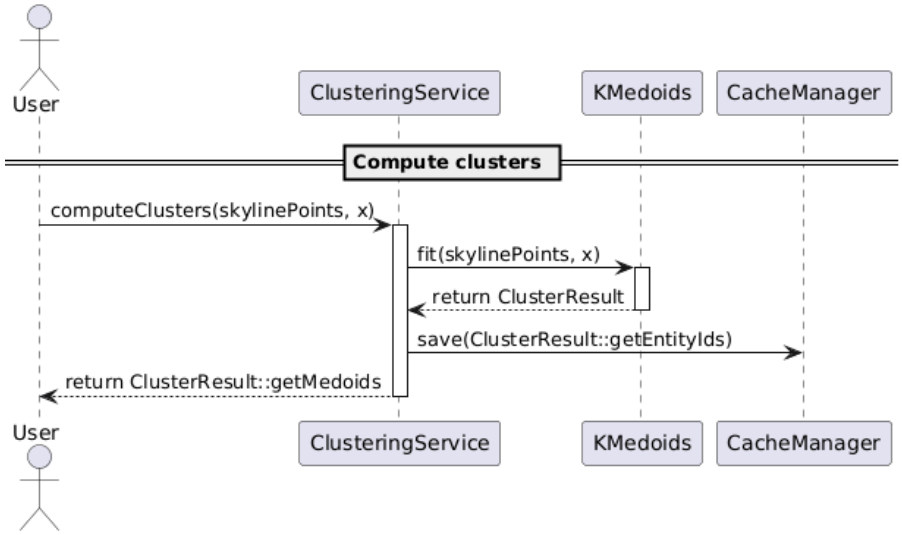


Рисунок 2.8 - Діаграма процесу кластеризації Парето-фронту за допомогою k-Medoids

Крім того, такий підхід відкриває можливість для користувача надсилати запит на отримання *y* подібних до обраного медоїда рішень. Для цього з кешу завантажується кластер, пов’язаний з переданим медоїдом. Після цього, за допомогою тієї ж метрики відстані, яка використовувалась у K-

Medoids при кластеризації, проводиться сортування результатів. З отриманого списку *y «*найближчих» точок і є результатом. Діаграма даного процесу наведена на рисунку 2.9. Ця двоступенева взаємодія дозволяє користувачеві проводити інтерактивне дослідження Парето-фронту: спочатку вивчати медоїди кластерів, а потім проводити аналіз найближчих точок.

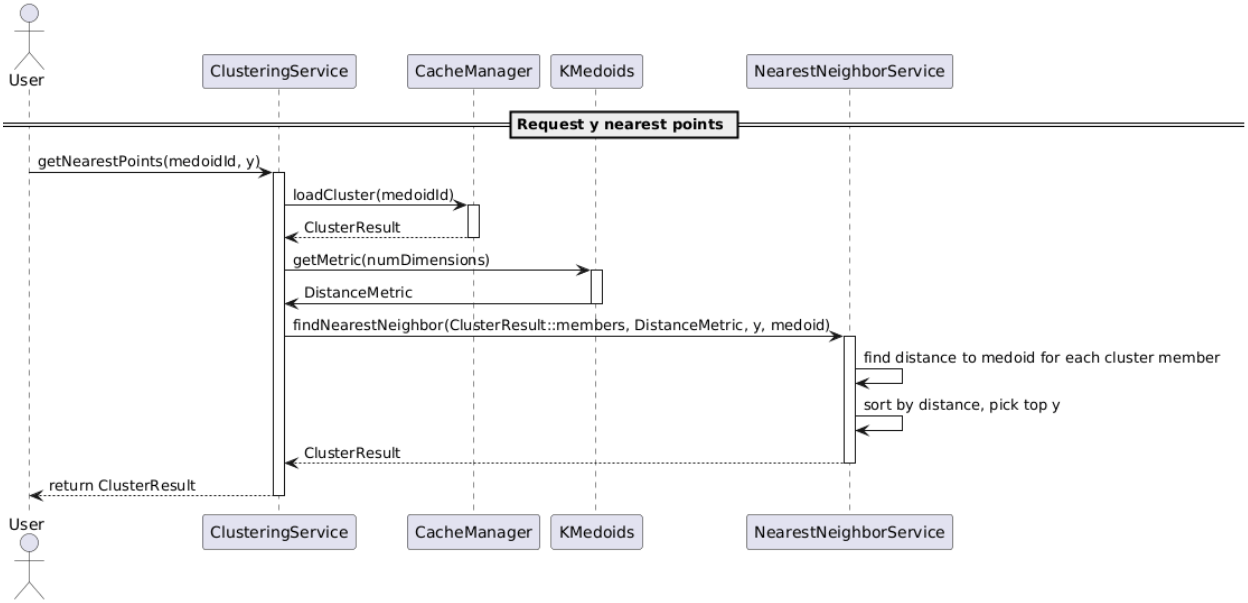


Рисунок 2.9 - Діаграма процесу відбору *y «*найближчих» до медоїда результатів

2.4 Архітектура методу формування Skyline

Запропонований метод формування Парето-фронту з адаптивним вибором алгоритму Skyline та використанням кластеризації для подолання проблеми CoD наведено на рисунку 2.10.

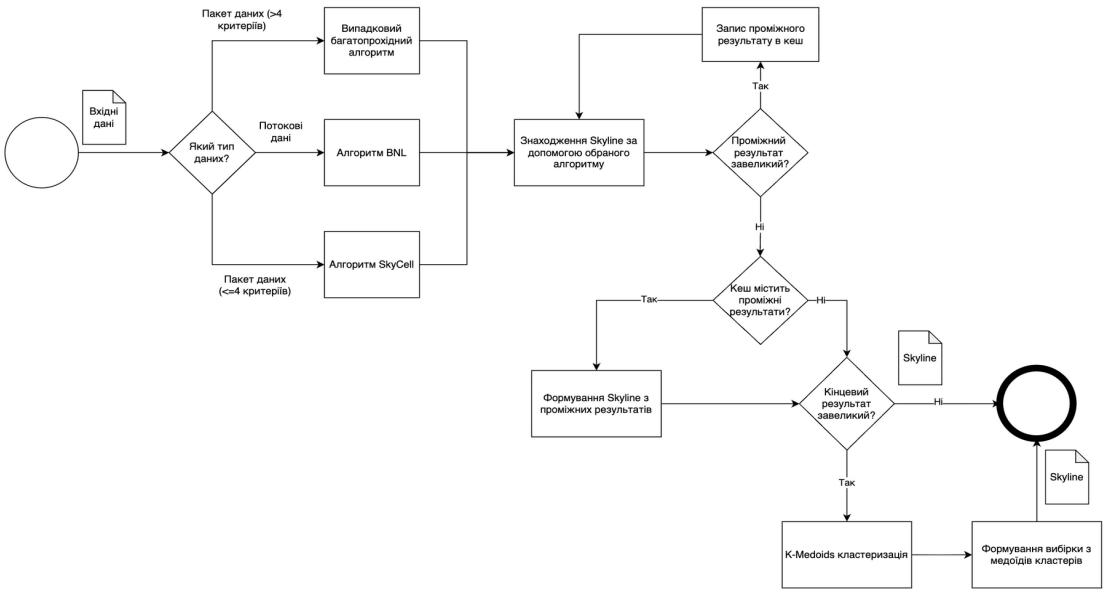


Рисунок 2.10 – Запропонований метод формування Skyline-вибірки

Уся процедура формування Skyline, подана на схемі, являє собою поетапну оркестрацію вибору алгоритму, обчислення проміжних результатів, використання кешу та, за потреби, додаткового етапу кластеризації. На початку виконується класифікація вхідних даних: визначається, чи є вони потоковими, чи представлені у вигляді пакета, а також встановлюється кількість критеріїв. Для потокових даних застосовується спеціалізований випереджувальний багатокритеріальний алгоритм. Для пакетів даних із великою кількістю критеріїв (більше 4) обирається алгоритм BNL. Для пакетів даних із меншою кількістю критеріїв (менше або рівно 4) використовується алгоритм SkyCell. Таким чином відбувається адаптивний вибір найпридатнішого методу, що мінімізує обчислювальні витрати залежно від структури даних.

Після вибору алгоритму виконується процес знаходження Skyline-множини. Якщо під час пошуку розмір проміжний результат виявився завеликим - він зберігається у кеші. Записані у кеш проміжні дані почергово

зчитуються після обробки усього масиву вхідних даних. З них формується фінальна вибірка Skyline. Якщо ж у кеші пусто – це означає що отриманий результат і є Skyline-вибіркою.

Якщо кінцевий результат перевищує обмеження на розмір, встановлене користувачем, використовується кластеризація K-Medoids. Вона поділяє Skyline на кластери, кількість яких дорівнює обмеженню на розмір. З медоїдів отриманих кластерів формується кінцевий результат. Він використовується як підмножина кандидатів для подальшого уточнення Skyline.

Висновок до розділу

У даному розділі було розглянуто алгоритми Skyline, методи обробки великих масивів даних та методи роботи з багатовимірними даними, які було обрано для проведення даного дослідження.

Було виокремлено 3 алгоритми Skyline, які дозволяють обробляти дані з різних інтерфейсів вводу даних у МП Java, забезпечуючи при цьому швидкодію, ефективне використання ресурсів та надійність:

- Parallel BNL;
- SkyCell;
- Randomized Multi-Pass Skyline.

Було наведено опис їх роботи, псевдокод, оцінку швидкодії, просторову складність та особливості реалізації на МП Java.

Було наведено опис двох методів обробки великих масивів даних: пакетну обробку з кешуванням та обробку за допомогою Stream API. Для кожного було наведено переваги та недоліки, а також особливості імплементації.

Було розглянуто способи боротьби з проблемою CoD, наведено опис алгоритму ε-Skyline та його недоліки, а також запропоновано рішення з використанням моделей кластеризації. Було описано модель кластеризації K-

Medoids, яка дозволяє розбивати Skyline-вибірку на кластери та знаходити центроїди кожного з них.

Врешті, було описано запропонований метод формування Skyline-вибірок, використовуючи адаптивний вибір алгоритмів та кластеризацію для вирішення проблеми CoD.

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Для реалізації запропонованого методу формування Skyline-вибірок було обрано форм-фактор програмної бібліотеки на Java. Порівняння з іншими форм-факторами наведено у таблиці 3.1.

Таблиця 3.1 – Порівняння можливих форм-факторів для запропонованого ПЗ

Форм-фактор	**Рівень ізоляції**	**Продуктивність**	**Простота інтеграції**
Програмна бібліотека	Високий	Висока (локальні виклики, без мережевих затримок)	Проста
Вбудований код у застосунку (embedded)	Низький	Висока	Середня (залежить від структури застосунку)
REST / HTTP-мікросервіс (Skyline-as-a-Service)	Середній	Нижча через мережеві витрати	Складна (потрібна інфраструктура)
Плагін до існуючого фреймворку	Середній	Висока	Складна

Як видно з порівняння, форм-фактор бібліотеки є найкращим, оскільки поєднує високий рівень ізоляції внутрішніх компонентів, продуктивність та простоту інтеграції. Такий підхід дозволяє користувачеві (розробнику сервісу чи аналітику):

- легко підлаштовувати метод під свої вимоги, при цьому не втручаючись у деталі внутрішньої реалізації (на противагу вбудованому коду);
- легко контролювати, яку версію програмного забезпечення використовувати (на противагу вбудованому коду та плагіну);
- не перейматись за безпеку даних та мати повний контроль над швидкодією (на противагу мікросервісу);
- уникати лишніх інтеграцій, якщо цікавить тільки функціонал Skyline (на противагу плагіну).

Для побудови бібліотеки для формування Skyline-вибірок за запропонованим методом було обрано архітектурний шаблон «Оркестратор»[19]. У ньому логіка керування процесами та взаємодією між компонентами зосереджена в одному центральному елементі — оркестраторі. Він координує кроки, виклики сервісів та обмін даними між ними. Схема роботи шаблону «Оркестратор» наведена на рисунку 3.1.

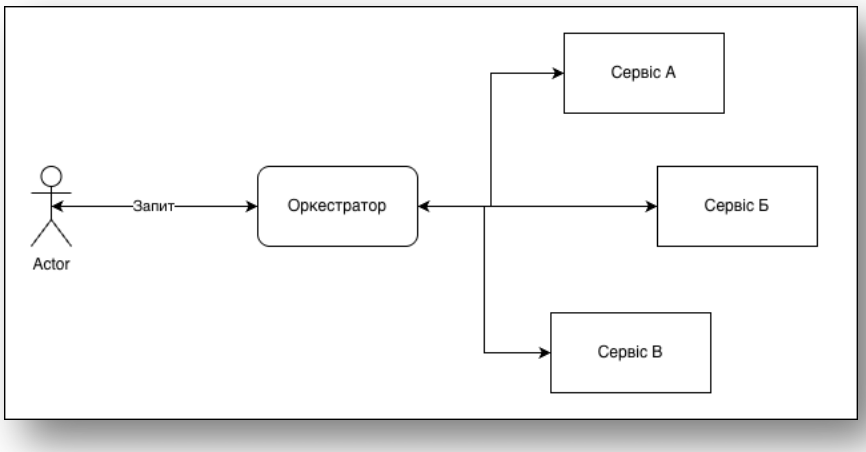


Рисунок 3.1 - Схема роботи архітектурного шаблону «Оркестратор»

В системі існує центральний компонент, який:

- знає повний бізнес-процес;
- викликає інші сервіси або компоненти у потрібній послідовності;
- збирає результати;
- вирішує, що робити далі (branching, retries, compensation);
- контролює життєвий цикл процесу.

Усі інші сервіси (А, Б та В) не знають про загальний контекст процесу й виконують локальні задачі. Вони є «виконавцями», а оркестратор — «диригентом».

«Оркестратор» добре підходить у цій ситуації, тому що бібліотеці потрібне єдине місце для координації кількох складових: обчислення Skyline, кластеризації, запис та зчитування кешу, звернень до репозиторію та конвертації об’єктів, без делегування цієї логіки доменному коду. Більше того, «Оркестратор» зберігає розмежування відповідальностей: алгоритм є окремим компонентом, кеш під’єднується як модуль, доступ до даних абстраговано. Він також пропонує стандартизований API. Це зменшує зв’язність та дозволяє легко робити заміни чи розширення (інше сховище кешу, новий алгоритм кластеризації, альтернативний SolutionConverter/Repository), локалізуючи зміни у налаштуванні оркестратора.

Також він централізує кросс-функціональні задачі — асинхронне очищення кешу, генерацію ключів кешу, перевірку вхідних даних — тож кожен аспект має одного відповідального, а решта системи може залишатися зосередженою на доменних даних.

3.2 Обґрунтування вибору засобів розробки

Для написання бібліотеки для формування Skyline-вибірок було використано МП Java. Дану МП було вибрано через її швидкодію, надійність, широкий функціонал, безпеку та популярність у доменах, де використовуються Skyline-запити. Порівняння з можливими конкурентами наведено у таблиці 3.2.

Таблиця 3.2 - Порівняння Java з можливими конкурентами

Критерій	**Java**	**C#**	**JavaScript**	**Python**
Продуктивність	Висока (компільована, JVM оптимізація)	Висока (компільована, .NET оптимізація)	Середня (інтерпретована, V8 оптимізація)	Середня (інтерпретована, може бути повільною)
Використання в індустрії	Широке	Вужче	Вужче	Вужче
Екосистема та інструменти	Велика (Spring, Hibernate)	Велика (.NET, Visual Studio)	Велика (React, Angular, Node.js)	Велика (NumPy, pandas, TensorFlow)
Масштабованість	Висока	Висока	Середня	Середня
Безпека	Висока (управління пам'яттю, типізація)	Висока (управління пам'яттю, типізація)	Середня	Середня

Java є однією з найпопулярніших мов програмування для розробки серверних додатків і, як видно з таблиці, має ряд переваг, таких як висока швидкість, масштабованість, розвинута екосистема, великий набір інструментів та безпека. Java сервіси користуються високою популярністю у електронній комерції, системах рекомендацій та підтримки рішень, що дає розробленому в рамках даної роботи рішенню велику кількість потенційних користувачів.

Також при розробці функціоналу кластеризації та пошуку «відстані» між рішеннями було використано бібліотеку ELKI [20, с. 142].

ELKI (Environment for Developing KDD-Applications Supported by Index-Structures) — це розвинений open-source фреймворк на Java, орієнтований насамперед на наукові дослідження у галузі data mining, зокрема на неконтрольовані (unsupervised) методи: кластеризацію, виявлення аномалій (outlier detection) та ефективний пошук подібностей / відстаней.

Його архітектура побудована з модульним підходом: алгоритми, типи даних, метрики відстаней, структури індексів, формати вводу/виводу — усе теоретично й практично може бути комбіноване у різних комбінаціях. Це дає можливість гнучкої адаптації під різні задачі, без «жорсткого» зв’язку компонентів.

Хоч в екосистемі Java існують й інші бібліотеки для ML та кластеризації (наприклад, Weka, Java-ML), ELKI вирізняється своєю гнучкістю, масштабованістю та спеціалізацією саме на unsupervised data mining та багатовимірні дані.

ELKI також пропонує механізми індексування (наприклад, R*-tree та інші просторові індекси), що дає значний приріст продуктивності при пошуку kNN, range-запитів та при кластеризації. Це означає, що навіть для великих наборів багатовимірних даних можна реалізувати масштабовані рішення з прийнятною швидкістю.

Для формування Skyline було обрано та реалізовано на МП Java алгоритми Parallel BNL, SkyCell та випадковий багато-прохідний алгоритм (Randomized Multi-Pass). Для реалізації бібліотеки було обрано саме їх

оскільки вони добре доповнюють один одного та дозволяють бібліотеці працювати з різними типами вхідних даних:

- Parallel BNL – єдиний алгоритм, здатний працювати з Java Stream API, завдяки паралельній обробці досягає хороших показників швидкодії;
- SkyCell – оптимізація Parallel BNL, використовується для обробки пакетних даних з кількістю критеріїв менше 4. Його швидкодія менш залежна від розміру вхідних даних, але швидкість обробки багатовимірних даних гірша, ніж у інших;
- Randomized Multi-Pass – оптимізація Parallel BNL, використовується для обробки пакетних даних з кількістю критеріїв більше 4, оптимально працює з багатовимірними даними.

3.3 Конструювання програмного забезпечення

Повну діаграму класів бібліотеки наведено на рисунку 3.2.

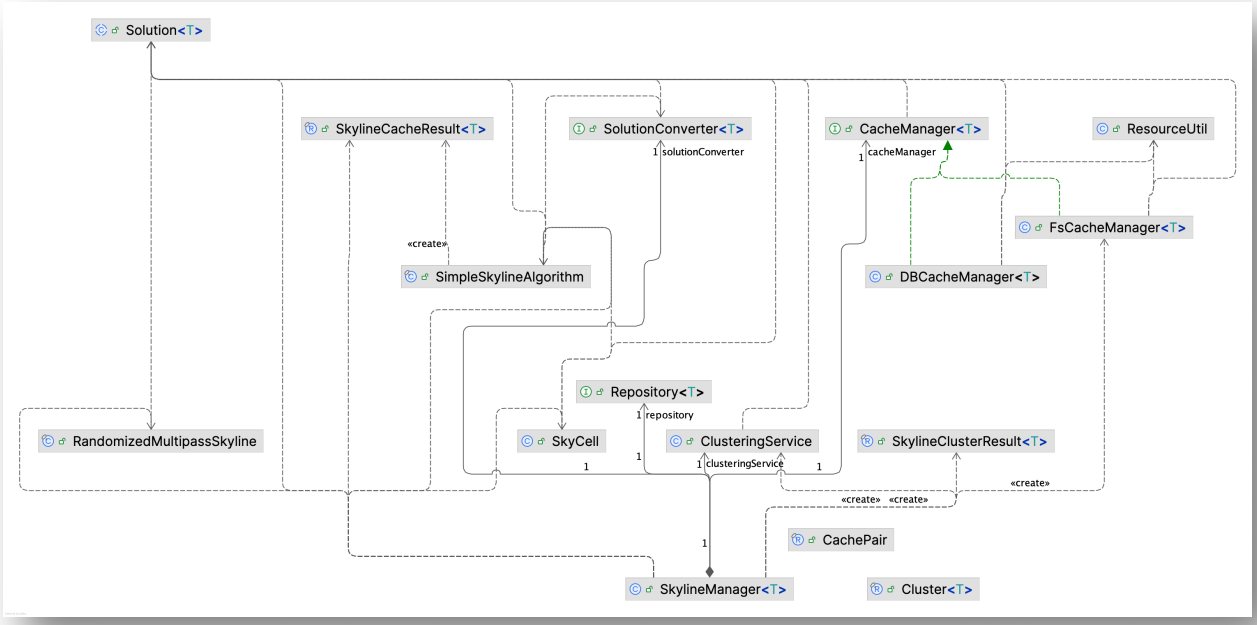


Рисунок 3.2 – Повна діаграма класів

Отже, клас SkylineManager – це «оркестратор» або «диригент», який є основною точкою взаємодії з бібліотекою. Він приймає або авто-конфігурує її параметри, містить логіку та послідовність обробки даних. Користувач повинен передати 2 обов’язкові компоненти: інтерфейс Repository та SolutionConverter.

Repository – функціональний інтерфейс, який користувач повинен передати бібліотеці для взаємодії з об’єктами в сховищі даних.

Рисунок 3.3 – Діаграма класу Repository

Він є типізованим та містить метод findAllByIds(), який дозволяє отримувати об’єкти зі сховища даних за їх ідентифікаторами.

SolutionConverter – функціональний інтерфейс, імплементація якого задає логіку перетворення користувацького об’єкту на модель Solution, з якою взаємодіє бібліотека.

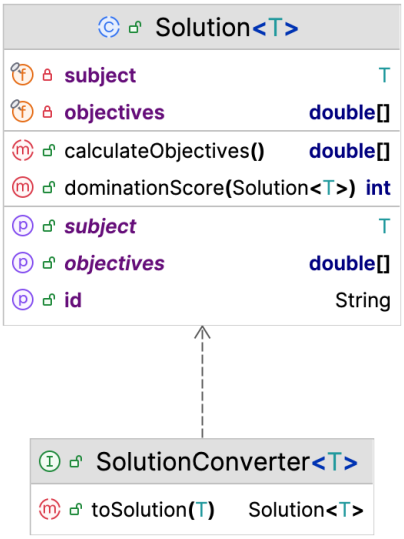


Рисунок 3.4 – Діаграма класів Solution та SolutionConverter

Модель Solution є типізованою та містить в собі методи calculateObjectives() та dominationScore(). Перший відповідає за визначення критеріїв порівняння, а другий – за функцію визначення домінантності.

SkylineManager містить параметри та методи для формування Skyline-вибірок. Його структура наведена на рисунку 3.5.

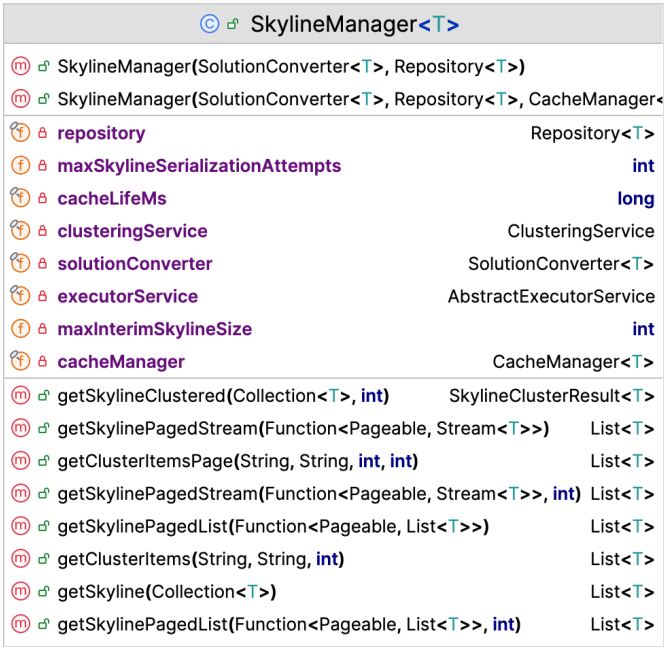


Рисунок 3.5 – Структура класу SkylineManager

Користувач може вказувати наступні параметри у SkylineManagger:

- executorService – абстракція для паралельного формування Skyline-вибірки;
- solutionConverter – реалізація функціонального інтерфейсу SolutionConverter;
- cacheLifeMs – параметр, який регулює, як довго тримати кластери у кеші у мілісекундах;
- repository – реалізація функціонального інтерфейсу Respository;
- maxSkylineSerializationAttempts – ліміт кількості серіалізацій для проміжної вибірки Skyline, при досягненні якого повертається помилка;
- maxInterimSkylineSize – ліміт розміру проміжної вибірки Skyline, при досягненні якого проміжна вибірка записується в кеш;
- clusteringService – реалізація сервісу кластеризації;

- cacheManager – реалізація сервісу кешування.

SkylineManager має такі методи:

- List<T> getSkylinePagedList(Function<Pageable, List<T>> dataFunction) – метод, який використовує пакетну обробку без кластеризації;
- SkylineClusterResult<T> getSkylinePagedList(Function<Pageable, List<T>> dataFunction, int numClusters) – метод, який використовує пакетну обробку з кластеризацією на numClusters кластерів;
- List<T> getSkylinePagedStream(Function<Pageable, Stream<T>> dataFunction) - метод, який використовує потокову обробку без кластеризації;
- SkylineClusterResult<T> getSkylinePagedStream(Function<Pageable, Stream<T>> dataFunction, int numClusters) – метод, який використовує потокову обробку з кластеризацією на numClusters кластерів;
- List<T> getSkyline(Collection<T> items) – метод, який знаходить Skyline для колекції items без кластеризації;
- SkylineClusterResult<T> getSkyline(Collection<T> items, int numClusters) – метод, який знаходить Skyline для колекції items з кластеризацією на numClusters кластерів;
- List<T> getClusterItems(String cacheKey, String medoidId, int k) – метод, який повертає *k* елементів кластера за ідентифікатором його медоїда та ключем кешу;
- List<T> getClusterItemsPage(String cacheKey, String medoidId, int page, int size) - метод, який повертає сторінку елементів кластера за ідентифікатором його медоїда та ключем кешу.

Ці методи та параметри дають користувачеві бібліотеки свободу в виборі та налаштуванні способу формування Skyline-вибірки.

При розробці також було враховано необхідність можливості зберігання кешу як в файловій системі, так і у SQL базах даних. Для цього було розроблено дві імплементації CacheManager: FSCacheManager та DBCacheManager.

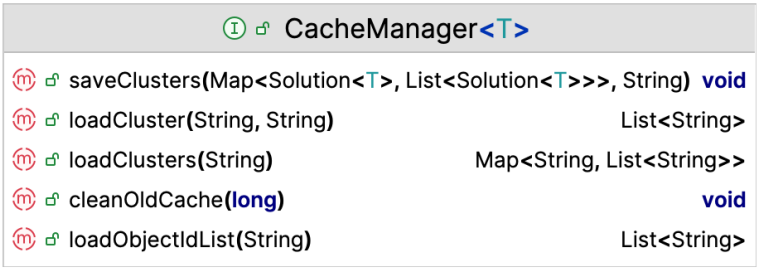


Рисунок 3.6 – Структура класу CacheManager

Методи CacheManager такі:

- void saveClusters(Map<Solution<T>, List<Solution<T>>> clusterMedoidIdMap, String cacheKey) – зберігає мапу кластерів за вказаним ключем;
- Map<String, List<String>> loadClusters(String cacheKey) – завантажує всі кластери за ключем;
- List<String> loadCluster(String cacheKey, String medoidId) – завантажує кластер за ключем і медоїдом;
- List<String> loadObjectIdList(String cacheKey) – завантажує проміжну Skyline-вибірку;

- void cleanOldCache(long cutoffEpochMillis) – видаляє застарілі записи.

FsCacheManager відповідає за запис кешу в файлову систему у вигляді JSON файлів. Він використовує бібліотеку Jackson для серіалізації та десеріалізації даних, зчитування та запису файлів. Для запису проміжних Skyline він створює файл з ідентифікаторами об’єктів. Для запису кластерів використовується мапа, в якій ключ – це ідентифікатор медоїда кластера, а значення – список ідентифікаторів об’єктів кластера. Такий підхід дозволяє економити пам’ять на диску, яку займає кеш. Назви файлів – це ключі, що генеруються відповідно до наступного формату:

skyline | cluster_{{ System.currentTimeMillis()}}_UUID

Префікс skyline або cluster позначає тип кешу, мітка поточного часу дозволяє легко видаляти застарілі записи, суфікс UUID гарантує уникнення колізій в кеші.

DBCacheManager відповідає за збереження кешу у реляційних базах даних. Він використовує бібліотеки HikariCP, Liquibase та JDBI. За допомогою Liquibase ініціалізується структура таблиць, а HikariCP та JDBI дозволяють взаємодіяти з БД. Користувач вказує параметри під’єднання до БД. Структура таблиць бази даних наведена на рисунку 3.7. Опис її таблиць – у таблицях 3.3 – 3.5.

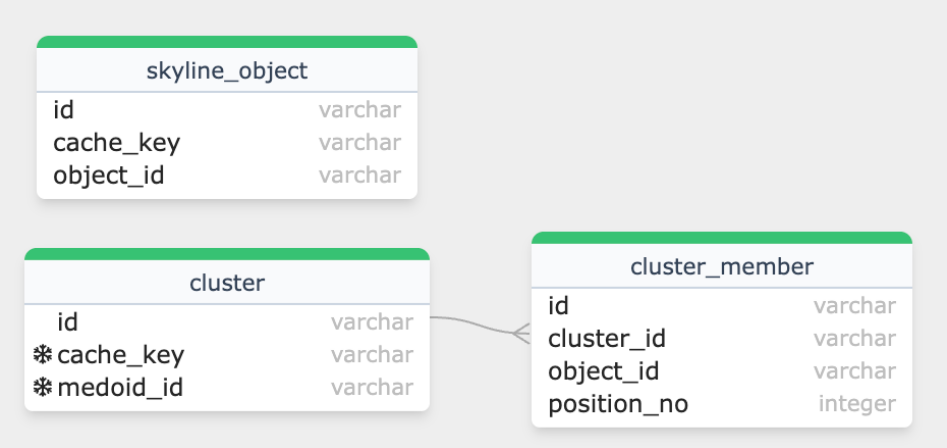


Рисунок 3.7 – Схема таблиць бази даних

Таблиця 3.3 – Опис таблиці cluster

Таблиця	Назва поля	Тип даних	Опис
cluster	id	varchar	унікальний ідентифікатор запису (відповідає одному медоїду кластера). Первинний ключ.
	cache_key	varchar	ключ кешу, який групує кластери в один набір (аналог файлу в FSCacheManager).
	medoid_id	varchar	ідентифікатор медоїда, якому належить кластер.
	(унікальне обмеження)	—	Пара (cache_key, medoid_id) повинна бути унікальною, щоб у межах одного кешу медоїд не повторювався.

Таблиця 3.4 – Опис таблиці cluster_member

Таблиця	Назва поля	Тип даних	Опис
cluster_member	id	varchar	унікальний ідентифікатор запису. Первинний ключ.
	cluster_id	varchar	ідентифікатор кластера (FK → cluster.id). Визначає кластер, якому належить об’єкт.
	position_no	integer	позиція об'єкта в кластері.
	object_id	varchar	ідентифікатор об'єкта, що входить до Skyline-вибірки.
	(зовнішній ключ)	—	cluster_id посилається на cluster(id), при видаленні кластера всі його об’єкти видаляються (ON DELETE CASCADE).

Таблиця 3.5 – Опис таблиці skyline_object

Таблиця	Назва поля	Тип даних	Опис
skyline_object	id	varchar	унікальний ідентифікатор запису (відповідає одному медоїду кластера). Первинний ключ.
	cache_key	varchar	ключ кешу, який групує кластери в один набір (аналог файлу в FSCacheManager).
	object_id	varchar	ідентифікатор об'єкта, що входить до Skyline-вибірки.

3.4 Оцінка якості запропонованого рішення

Для оцінки якості запропонованого рішення складається з таких етапів:

- перевірка правильності роботи алгоритмів;
- оцінка швидкодії рішення.

3.4.1 Перевірка правильності роботи алгоритмів

Забезпечення коректності алгоритмів формування Skyline-вибірок потребує систематичної методики тестування, здатної охопити як функціональні властивості, так і сценарії роботи з різними типами даних. Верифікація має поєднувати формальні, емпіричні та порівняльні підходи.

Першим кроком є використання контрольних наборів даних зі заздалегідь відомими Skyline-результатами для тестування кожного алгоритму

окремо. Такі набори дозволяють формально перевірити коректність реалізації без впливу випадковостей або неоднозначності.

Для повноти покриття необхідно включати:

- малі детерміновані набори (10–20 точок), для яких Skyline можна знайти вручну;
- великі синтетичні набори з контрольованим розподілом значень (лінійні, кластеризовані, гаусівські, рівномірні);
- випадки граничних ситуацій: висококорельовані виміри, повна відсутність Skyline (усі точки рівні), Skyline розміром 1.

Коректність результату повинна оцінюватися не лише за кінцевим списком точок, а й за дотриманням фундаментальних властивостей Skyline:

- антисиметрія домінування: у Skyline не можуть бути присутні точки, що домінують одна одну;
- максимальність результату: жодна точка поза Skyline не повинна домінувати точки з Skyline;
- неповторність представлення: усі точки Skyline є унікальними за домінантними координатами;
- ідемпотентність: повторне виконання алгоритму над уже отриманою Skyline-множиною не змінює її;

Перевірка цих властивостей забезпечує строгий контроль над коректністю незалежно від конкретного алгоритмічного підходу.

Другий крок – диференційне тестування, яке передбачає паралельний запуск декількох незалежних реалізацій одного й того самого алгоритмічного завдання. Будь-яка різниця у вихідних Skyline-вибірках сигналізує про потенційну помилку в одній із реалізацій. У випадку наявності кількох алгоритмів диференційне тестування стає природним інструментом перевірки:

збіг результатів усіх трьох реалізацій при різних тестових наборах є сильною індикцією коректності.

Третій крок - аналіз стабільності та інваріантності до порядку даних. Оскільки порядок записів не повинен впливати на результат Skyline між однаковими точками, алгоритми мають демонструвати інваріантність до перестановки входу. Тестування включає:

- багаторазове виконання алгоритмів над одним і тим самим набором даних, але у випадкових перестановках;
- перевірку детермінованості результату;
- оцінку стабільності у випадках із дубльованими або рівними точками.

Останній крок - стохастичне генеративне тестування. Воно полягає у створенні великої кількості випадкових наборів даних з різними розподілами: рівномірним, гауссівським, експоненційним, мультимодальним, корельованим та анти-корельованим.

Для кожного набору перевіряються властивості Skyline та порівнюються результати диференційним методом. Велика варіативність даних дозволяє виявити складні помилки, які неможливо знайти на ручних прикладах.

3.4.2 Оцінка швидкодії рішення

Для оцінки швидкодії рішення необхідно провести заміри часу виконання алгоритмів при різних об’ємах (від 10^3 до 10^7) та кількостях критеріїв (4 та 5) вхідних даних. Результати цього дослідження наведені у таблицях 3.7-3.8.

Таблиця 3.7 – Дослідження швидкодії при кількості критеріїв 5

Розмір вхідного набору	BNL	SkyCell	Multi-pass
1 000	3 мс	10 мс	8 мс
10 000	28 мс	44 мс	20 мс
100 000	174 мс	245 мс	34 мс
1 000 000	849 мс	1763 мс	141 мс
10 000 000	9804 мс	50687 мс	4006 мс

Таблиця 3.8 – Дослідження швидкодії при кількості критеріїв 4

Розмір вхідного набору	BNL	SkyCell	Multi-pass
1 000	4 мс	8 мс	4 мс
10 000	21 мс	18 мс	10 мс
100 000	72 мс	37 мс	33 мс
1 000 000	263 мс	72 мс	92 мс
10 000 000	4626 мс	967 мс	2587 мс

Дослідження швидкодії алгоритмів підтверджує їх теоретичні сценарії використання. SkyCell оптимальний при малій кількості критеріїв (менше або рівно 4) та великій кількості даних (від 10^5 або 10^6). Randomized Multi-Pass

підходить для обробки вибірок з кількістю критеріїв 5 та більше та розміром від 10^5. BNL підходить для обробки малих об’ємів даних (до 10^4) а також у випадках, коли неможливо застосувати попередні алгоритми.

Висновки до розділу

У даному розділі було розглянуто повний цикл розробки програмного забезпечення, створеного для формування Skyline-вибірок у середовищі Java. На основі проведеного аналізу було обґрунтовано вибір форм-фактора програмної бібліотеки, який забезпечує оптимальне поєднання продуктивності, ізоляції внутрішніх компонентів та простоти інтеграції у різні застосунки.

У межах архітектурного проєктування було визначено ключові компоненти бібліотеки та їхню взаємодію, зокрема менеджер Skyline-процесу, інтерфейси для отримання даних та перетворення користувацьких об’єктів. Побудована модель забезпечує масштабованість, гнучкість та можливість розширення з боку розробника.

Було проведено обґрунтування вибору мови програмування та інструментів для імплементації програмного засобу. На основі порівняльного аналізу було доведено правильність вибору Java як мови реалізації ПЗ. Було також наведено аргументацію вибору алгоритмів Skyline та засобів кластеризації та оцінки відстаней між рішеннями.

У розділі, присвяченому конструюванню програмного забезпечення, було детально описано особливості функціонування бібліотеки та способи взаємодії з нею. Наведено детальний опис способів інтеграції бібліотеки та можливостей її конфігурування. Окрему увагу приділено механізмам роботи з великими обсягами інформації: використання кешу для збереження проміжних результатів, потоковій обробці, а також кластеризації K-Medoids,

яка дозволяє керувати розміром Skyline-вибірки та пом’якшувати вплив проблеми «прокляття розмірності».

Врешті, було проведено опис методів перевірки якості отриманого ПЗ та дослідження його швидкодії.

4. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ

4.1 Опис ідеї проєкту

Таблиця 4.1 — Опис ідеї стартап-проєкту

<table>
<tr><th>Зміст ідеї</th><th>Основні напрямки застосування</th><th>Вигоди для користувачів</th></tr>
<tr><td rowspan="5">Розроблення та комерціалізація програмного продукту у вигляді Java-бібліотеки та хмарного B2B-сервісу для реалізації Skyline/Pareto-фільтрації та багатокритеріального відбору альтернатив. Продукт забезпечує виконання ефективних алгоритмів Skyline (Parallel BNL, SkyCell, Randomized Multi-Pass), підтримує кластеризацію Skyline-множини на основі алгоритму K-Medoids, а також реалізує механізми кешування результатів та обробки даних у пакетному та потоковому режимах.</td><td>E-commerce і маркетплейси</td><td rowspan="2">Зменшення інформаційного перевантаження (представлення лише Pareto-оптимальних альтернатив)</td></tr>
<tr><td>Фінансовий сектор (скоринг клієнтів, відбір інвестиційних і кредитних продуктів)</td></tr>
<tr><td>Логістика (маршрути з урахуванням часу, вартості, ризиків, екопоказників)</td><td>Підвищення прозорості та обґрунтованості рішень</td></tr>
<tr><td>Туристичні сервіси (відбір готелів, авіаквитків, турів)</td><td>Для розробників — скорочення витрат часу на реалізацію Skyline-алгоритмів</td></tr>
<tr><td></td><td>Підвищення конверсії та якості сервісів (e-commerce, фінанси)</td></tr>
</table>

Закінчення таблиці 4.1

Зміст ідеї	**Основні напрямки застосування**	**Вигоди для користувачів**
	Бізнес-аналітика та СППР (Парето-оптимальні набори альтернатив)	Для розробників — скорочення витрат часу на реалізацію Skyline-алгоритмів
		Для бізнесу — швидше впровадження багатокритеріального відбору та зниження операційних ризиків

Для початкової оцінки релевантності ідеї на ринку потрібно провести порівняльний аналіз запропонованого рішення з існуючими альтернативами. На сьогодні таких альтернатив декілька:

- аналітичні СУБД із вбудованою підтримкою Skyline/Pareto (наприклад, Exasol, IBM DB2);
- реалізації Skyline на базі Apache Spark;
- пакети для мов Python/R, що реалізують Skyline/Pareto-фільтрацію.

У таблицях 4.2 – 4. наведено порівняння сильних та слабких сторін за техніко-економічними характеристиками.

Таблиця 4.2 — Порівняння з конкурентами за простотою інтеграції в існуючі інформаційні системи та технологічну інфраструктуру

Проєкт	**W (слабка сторона)**	**N (нейтральна сторона)**	**S (сильна сторона)**
Мій проєкт	Необхідність наявності Java-стеку або додаткового сервісу-шлюзу для інтеграції з не-Java системами.	Потреба у мінімальній адаптації коду з боку команди розробки (налаштування конвертерів, репозиторіїв тощо).	Надання Java-бібліотеки та REST API, що спрощує інтеграцію з типовими застосунками. Вбудоване ефективне вирішення проблеми CoD.
Exasol, IBM DB2	Потреба у впровадженні та підтримці аналітичної СУБД, що вимагає міграції або реплікації даних. Відсутність вбудованого вирішення CoD.	Простота інтеграції для організацій, які вже використовують відповідну СУБД або планують міграцію на неї.	Вбудована підтримка Skyline/Pareto-запитів на рівні СУБД. Вища швидкодія.

Закінчення таблиці 4.2

Проєкт	**W (слабка сторона)**	**N (нейтральна сторона)**	**S (сильна сторона)**
Apache Spark Skyline	Низька доцільність для «легких» серверних застосунків без big-data інфраструктури. Порівняно висока складність налаштування, розгортання та підтримки Spark-кластерів.	Інтеграція є прийнятною у середовищах, де Apache Spark уже є ключовим елементом інфраструктури.	Висока придатність для вже існуючих кластерів обробки даних на базі Spark (особливо для batch-аналітики).
Python/R packages	Обмежена придатність до прямої інтеграції з застосунками на інших МП. Потреба у додаткових сервісах/обгортках для використання в продуктивних B2B-рішеннях.	Можуть використовуватися як допоміжний інструмент для прототипування моделей, паралельно з іншими технологіями.	Зручність для локального аналізу даних аналітиками та data scientists у звичних середовищах (Jupyter, RStudio).

Таблиця 4.3 – Порівняння з конкурентами за загальною вартістю володіння (TCO) та можливістю масштабування

Проєкт	**W (слабка сторона)**	**N (нейтральна сторона)**	**S (сильна сторона)**
Мій проєкт	На початкових етапах розвитку продукту фінансові витрати на підтримку та розвиток можуть бути відносно високими порівняно зі зрілими рішеннями великих вендорів.	Потреба в інвестиціях у впровадження й налаштування на боці замовника (типово для більшості рішень подібного класу).	Можливість гнучкої моделі ліцензування. Масштабування шляхом розгортання у мікросервісній архітектурі, використання кешування та пакетної/потокової обробки даних. Адаптація до роботи з будь-якими джерелами даних.
Exasol, IBM DB2	Значна вартість ліцензій та інфраструктури аналітичної СУБД. Додаткові витрати на міграцію даних та адаптацію існуючих систем.	Економічна доцільність може бути виправдана для великих організацій із масштабними аналітичними потребами.	Висока продуктивність для великих обсягів даних за умови повного використання можливостей платформи.

Закінчення таблиці 4.3

Проєкт	**W (слабка сторона)**	**N (нейтральна сторона)**	**S (сильна сторона)**
Apache Spark Skyline	Висока вартість розгортання та супроводу Spark-кластерів у випадку відсутності відповідної інфраструктури. Потреба у кваліфікованих спеціалістах для підтримки й оптимізації.	Вартість володіння є прийнятною для компаній, що вже мають інфраструктуру big data і компетентні DevOps-/data-інженерні команди.	Високі можливості горизонтального масштабування для великих обсягів даних у розподілених середовищах.
Python/R packages	Високі непрямі витрати у випадку спроб інтеграції в промислове середовище. Обмежені можливості масштабування для великих промислових навантажень без програмного доопрацювання.	Фінансові витрати можуть бути невисокими за умови використання в рамках невеликих аналітичних команд чи академічних досліджень.	Відсутність прямої вартості ліцензування (open-source, безкоштовні пакети).

4.2 Технологічний аудит проєкту

Таблиця 4.3. — Технологічна здійсненність ідеї проєкту

№	Ідея проєкту	Технології реалізації	Наявність технологій	Доступність технологій
1	Java-бібліотека Skyline/Pareto	Публікація артефактів у Maven Central, Spring Boot starter, інтеграція з системами логування та моніторингу (Micrometer, OpenTelemetry).	Усі технології є загальнодоступними.	Висока: Java та Spring Boot є одним із домінуючих стандартів у корпоративному сегменті
2	Хмарний B2B API («Skyline-as-a-Service»)	Spring Boot REST, Docker, Kubernetes, шлюз API, механізми аутентифікації та авторизації (OAuth2/JWT)	Технології широко розповсюджені, підтримуються провідними хмарними провайдерами	Висока: існує велика кількість готових рішень та практик розгортання
3	Інтеграція з BI/аналітичними системами	JDBC/ODBC-конектори, REST-конектори до BI-платформ, експорт результатів Skyline у формати CSV/Parquet	Стандартні механізми інтеграції BI-систем із зовнішніми джерелами даних	Середня/висока: потрібен додатковий час на розроблення та сертифікацію конекторів

Висновок: технологічна реалізація продукту – можлива, вибрана технологія №1.

4.3 Аналіз ринкових можливостей запуску стартап-проєкту

Таблиця 4.4. — Попередня характеристика потенційного ринку

№	**Показник стану ринку**	**Характеристика**
1	Кількість головних гравців, од.	На глобальному ринку спеціалізованих рішень Skyline/Pareto-фільтрації налічується орієнтовно 5–7 прямих гравців (аналітичні БД із Skyline-функціями, окремі реалізації та бібліотеки), а також значна кількість непрямих конкурентів (BI-платформи, ML-рекомендаційні сервіси).
2	Загальний обсяг продаж, грн./ум. од.	Розглянута ніша інтегрується до ринків BI/аналітики та B2B-рекомендацій, сукупний обсяг яких становить десятки мільярдів доларів США на рік.
3	Динаміка ринку	Ринки BI та аналітики демонструють середньорічні темпи зростання близько 8–10%, а ринок B2B-рекомендаційних рішень – понад 30% на рік. Сегмент Skyline/Pareto-фільтрації перебуває на стадії раннього зростання з високим потенціалом розвитку.

Закінчення таблиці 4.4

№	Показник стану ринку	Характеристика
4	Наявність обмежень для входу	Бар'єри входу є середніми: необхідні глибокі знання алгоритмів і обробки даних, відповідність вимогам корпоративних клієнтів (безпека, надійність, підтримка). Водночас відносно невелика кількість прямих конкурентів пом'якшує конкурентний тиск.
5	Специфічні вимоги до стандартизації та сертифікації	Спеціалізовані стандарти для Skyline-технологій відсутні. Однак для роботи з корпоративними даними актуальними є загальноприйняті стандарти інформаційної безпеки (шифрування, захист персональних даних, відповідність GDPR, ISO 27001), а також підтримка стандартних інтерфейсів (REST, SQL, JDBC/ODBC).
6	Середня норма рентабельності, %	Для компаній, що працюють у сегменті програмного забезпечення, типовою є чиста рентабельність у межах 20–40%. За умови успішної комерціалізації очікувана рентабельність запропонованого продукту може становити не менше 20% у середньостроковій перспективі.

Висновок: обчислення Skyline є технологією, що все ще розвивається. Вона має високий потенціал, і, враховуючи невелику кількість конкурентів та середню норму рентабельності можна зробити висновок, що на даний момент, ринок для входження стартап-продукту є привабливим.

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проєкту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності в поведінці цільових груп	Вимоги споживачів до товару
1	Необхідність готового інструменту Skyline/Pareto для вбудовування у програмні системи	Команди розробників Java, технічні фахівці продуктів e-commerce, фінансових та логістичних систем	Орієнтація на якісний програмний інтерфейс, повноту документації, надійність і прогнозовану продуктивність	Наявність стабільного Java SDK, прикладів інтеграції, підтримка Spring Boot та роботи з різними джерелами даних
2	Потреба у B2B-рішенні для автоматизованого багатокритеріального відбору та рекомендацій	Бізнес-клієнти (маркетплейси, банки, страхові компанії, логістичні оператори, туристичні платформи)	Зосередженість на бізнес-результатах (конверсія, зниження ризиків, підвищення ефективності персоналу); висока чутливість до якості сервісу	Наявність готового функціоналу у форматі сервісу (SaaS/API), SLA, можливість інтеграції з існуючими системами, прозора модель ліцензування

Таблиця 4.6. — Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція підприємства
1	Вихід на ринок великих постачальників аналогічною функціональніст ю	Додавання Skyline/Pareto-функцій до продуктів провідних хмарних платформ і СУБД може суттєво зменшити частку ринку для нового гравця	Диференціація за рахунок спеціалізації (Java SDK, on-premise-рішення, галузеві модулі), розвиток партнерських програм, акцент на додаткових можливостях (кластеризація, пояснюваність рішень)
2	Внутрішня розробка рішен клієнтами (insourcing)	Організації з потужними IT-відділами можуть самостійно реалізувати Skyline-функціонал або використати пакети Python/R	Комунікація повної вартості володіння (TCO), демонстрація переваг промислового рішення (підтримка, продуктивність, оновлення), пропозиція змішаної моделі (відкритий базовий код + комерційна підтримка)
3	Низька обізнаність щодо Skyline/Pareto-підходів	Потенційні клієнти можуть не усвідомлювати наявність відповідної технології та її переваг над традиційними фільтрами та рейтинговими моделями	Проведення освітніх заходів (статті, вебінари, приклади використання), демонстрація практичних кейсів та економічного ефекту від впровадження

Таблиця 4.7. — Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція підприємства
1	Зростання ринку BI та рекомендаційних систем	Підвищення ролі аналітики та персоналізованих рекомендацій у бізнес-процесах	Позиціонування продукту як спеціалізованого модуля для реалізації багатокритеріальних рекомендацій, активний розвиток інтеграцій з BI та ETL-інструментами
2	Попит на прозорі та пояснювані рішення	Бізнес-користувачі та регулятори очікують зрозумілих механізмів прийняття рішень	Адаптація продукту до задач explainable AI: надання пояснень щодо включення/невключення альтернативи до Skyline-множини, розробка відповідних інтерфейсів
3	Обмежена кількість спеціалізованих конкурентів	Ніша Skyline/Pareto-фільтрації недостатньо насичена комерційними продуктами	Формування іміджу «стандартного» інструменту для Skyline-фільтрації, активна участь у професійних спільнотах і конференціях

Закінчення таблиці 4.7

№	Фактор	Зміст можливості	Можлива реакція підприємства
4	Можливість використання моделі відкритого ядра (open core)	Часткове відкриття програмного коду дозволяє залучити спільноту користувачів і розробників	Створення відкритої базової версії бібліотеки, доповненої комерційними модулями (розширені функції, підтримка, консалтинг)

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

№	Особливість конкурентного середовища	Прояв характеристики	Вплив на діяльність підприємства
	Обмежена кількість прямих конкурентів	Наявність лише декількох рішень із явною підтримкою Skyline/Pareto	Сприятлива ситуація для входу на ринок; доцільно швидко сформувати пізнаваний бренд і мережу партнерств
2	Потужні непрямі конкуренти	Широке використання BI-платформ та ML-рекомендаційних систем	Необхідність позиціонувати продукт як доповнення до існуючих систем, а не їх заміну; акцент на унікальності Skyline-підходу

Закінчення таблиці 4.8

№	Особливість конкурентного середовища	Прояв характеристики	Вплив на діяльність підприємства
3	Високі вимоги enterprise-клієнтів	Очікування надійності, безпеки, наявності підтримки та SLA	Побудова процесів контролю якості, моніторингу, технічної підтримки, підготовка документації та забезпечення відповідності вимогам безпеки
4	Швидкий технологічний розвиток	З’являються нові алгоритми, архітектури, інструменти обробки даних	Регулярне оновлення продукту, інвестиції в науково-дослідні роботи, співпраця з академічними установами
5	Чутливість ринку до продуктивності	Для великих обсягів даних недостатньо ефективні реалізації стають непридатними	Оптимізація алгоритмів, використання кешування, можливість горизонтального масштабування; проведення бенчмарків і публікація результатів
6	Залежність від інтеграції	Продукт має бути легко інтегрований в існуючу IT-інфраструктуру клієнта	Розвиток SDK, конекторів та шаблонів інтеграції, підтримка декількох сценаріїв розгортання (on-premise, хмара, гібридні моделі)

Таблиця 4.9 — Аналіз конкуренції в галузі за М. Портером

Складова	**Прямі конкуренти**	**Потенційні конкуренти**	**Постачальники**	**Клієнти**	**Товари-замінники**
Зміст	Аналітичні СУБД із Skyline, кастомні реалізації на Apache Spark, Skyline-пакети для Python/R	Великі хмарні платформи та виробники СУБД/BI, які можуть інтегрувати Skyline до наявних продуктів	Постачальники хмарної інфраструктури, серверного обладнання, інструментів розроблення, open-source-компонентів	Клієнти у сферах e-commerce, фінансів, логістики, туризму, консалтингу	ML-рекомендаційні системи без Skyline, традиційні методи фільтрації та ранжування, BI-платформи
Рівень сили впливу	Середній	Середньо-високий	Низький/середній	Високий	Середньо-високий

Висновок: структура галузі передбачає помірний тиск з боку прямих конкурентів, але значні ризики, пов’язані з появою нових рішень від великих постачальників та використанням товарів-замінників. Висока сила впливу клієнтів вимагає від стартапу гнучкої політики ціноутворення та високої якості сервісу.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор	Обґрунтування
1	Якість та повнота Skyline-алгоритмів	Наявність кількох алгоритмів Skyline, оптимізованих під різні обсяги та розмірності даних, забезпечує універсальність застосування продукту
2	Продуктивність та масштабованість	Підтримка пакетної і потокової обробки, кешування результатів та можливість горизонтального масштабування дозволяють обробляти великі набори даних із прийнятним часом відгуку
3	Простота інтеграції	Використання звичних для enterprise-сегмента технологій (Java, Spring Boot, REST, JDBC) знижує вартість інтеграції та скорочує її терміни
4	Гнучкість підключення джерел даних	Абстракція джерел даних (репозиторії, конвертери) дозволяє працювати з різними СУБД, файлами та сервісами без їхньої заміни
5	Документація та зручність для розробників	Наявність якісної документації, прикладів і шаблонів проєктів підвищує прийнятність рішення для розробницьких команд
6	Вартість володіння	Спеціалізований продукт із чітко визначеним функціоналом є економічно доцільною альтернативою впровадженню важких аналітичних СУБД лише заради однієї функції

Закінчення таблиці 4.10

7	Безпека та надійність	Підтримка локального розгортання та інтеграції з корпоративною інфраструктурою безпеки дозволяє задовольнити вимоги регульованих галузей
8	Можливість кастомізації	Відкриті розширювані інтерфейси дають змогу адаптувати продукт до специфіки конкретних предметних областей
9	Сервісна підтримка	Надання послуг консалтингу, інтеграції та технічної підтримки є важливим чинником для корпоративних замовників
10	Інструменти візуалізації	Наявність базових інструментів візуального аналізу Skyline-множин або інтеграція з BI-системами підвищують практичну значущість продукту для аналітиків

Таблиця 4.12. — Порівняльний аналіз сильних та слабких сторін

№	**Фактор конкурентоспроможності**	**Оцінка продукту (1–20)**
1	Якість реалізованих Skyline-алгоритмів	18
2	Продуктивність та масштабованість	17
3	Простота інтеграції	19

Закінчення таблиці 4.12

№	**Фактор конкурентоспроможності**	**Оцінка продукту (1–20)**
4	Гнучкість підключення джерел даних	18
5	Загальна вартість володіння (TCO)	17
6	Рівень документації та підтримки розробників	15
7	Забезпечення безпеки та надійності	16
8	Можливість кастомізації	18
9	Розвиненість сервісної підтримки	14
10	Наявність візуалізацій та аналітичного інтерфейсу	13

SWOT аналіз стартап-проєкту.

Сильні сторони (S)

- спеціалізація на Skyline/Pareto-фільтрації та багатокритеріальному відборі;
- наявність кількох алгоритмів і кластеризації Skyline-множини;
- використання поширеної технологічної платформи (Java, Spring Boot);

- гнучкість інтеграції з різними джерелами даних та системами;
- можливість реалізації як бібліотеки та як сервісу.

Слабкі сторони (W)

- відсутність відомого бренду та портфеля впроваджень;
- потреба у значних зусиллях для розбудови документації, UI та маркетингової присутності;
- обмежені можливості візуалізації у порівнянні з провідними BI-платформами;
- обмежені кадрові й фінансові ресурси на початковому етапі.

Можливості (O)

- зростання попиту на BI, аналітику і персоналізовані рекомендації;
- невелика кількість прямих конкурентів у вузькій Skyline/Pareto-ніші;
- підвищений інтерес до прозорих і пояснюваних моделей прийняття рішень;
- потенціал використання моделі відкритого ядра та формування спільноти користувачів.

Загрози (T)

- можливий вихід на ринок рішень із подібним функціоналом від великих постачальників (хмарних та СУБД-вендорів);
- внутрішня розробка аналогічних рішень силами IT-відділів великих компаній;

- конкуренція з боку ML-рекомендаційних систем і традиційних BI-підходів;
- негативні макроекономічні тенденції, що можуть зменшувати інвестиції у нові IT-рішення.

Таблиця 4.13 — Альтернативи ринкового впровадження стартап-проєкту

№	**Альтернатива (комплекс заходів)**	**Ймовірність отримання ресурсів**	**Орієнтовні строки реалізації**
1	Виведення на ринок open-core Java-бібліотеки (відкритого базового ядра) з комерційними розширеннями	Висока; потребує переважно розробницьких ресурсів та мінімальної інфраструктури	6–12 місяців до отримання перших користувачів та пілотних клієнтів
2	Створення хмарного сервісу (SaaS/API) «Skyline-as-a-Service» із тарифікацією за обсягом використання	Середня; вимагає інвестицій в інфраструктуру, DevOps, організацію служби підтримки	12–18 місяців до запуску стабільної версії та укладання перших комерційних контрактів
3	Зосередження на співпраці з системними інтеграторами та партнерами (SDK-ліцензії)	Середня; потребує додаткових ресурсів для розвитку партнерських відносин і навчання партнерів	12–24 місяці для формування стабільної мережі партнерів

Закінчення таблиці 4.13

№	Альтернатива (комплекс заходів)	Ймовірність отримання ресурсів	Орієнтовні строки реалізації
4	Розроблення галузевих вертикальних рішень (для e-commerce, фінансів, логістики тощо) на основі базового ядра	Низька/середня; потребує поглиблених предметних знань та інвестицій у маркетинг у кожній ніші	18–36 місяців із поетапним виходом на окремі галузеві сегменти

4.4 Розроблення ринкової стратегії проєкту

Таблиця 4.15. — Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Команди розробників Java та архітектори програмних систем у галузях e-commerce, логістики, SaaS	Висока	Середній/високий	Низька/середня	Проста

Закінчення таблиці 4.15

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
2	Корпоративні B2B-клієнти (маркетплейси, банки)	Середня/висока –	Високий –	Середня/висока	Середня
3	Аналітичні підрозділи, BI-команди, data scientists, які виконують багатокритеріальний аналіз даних	Середня	Середній	Середня	Середня

Які цільові групи обрано: Основними цільовими групами на початковому етапі обрано:

- команди розробників Java та архітекторів програмних систем (сегмент 1);
- корпоративних B2B-клієнтів, зацікавлених у впровадженні модулів багатокритеріальних рекомендацій і скорингу (сегмент 2).

Сегмент 3 (аналітичні підрозділи) розглядається як перспективний для подальшого розвитку та розширення продуктового портфеля.

Обрана альтернатива розвитку – розвиток за моделлю спеціалізованого технологічного постачальника з open-core Java-бібліотекою та комерційними розширеннями (B2B API, enterprise-модулі, сервісна підтримка). Стратегія концентричного (сфокусованого) охоплення ринку:

- на першому етапі – орієнтація на вузький сегмент Java-розробників та технічних команд;
- на наступних етапах – поетапний вихід на ширший B2B-сектор (корпоративні замовники в e-commerce, фінансах, логістиці, travel).

Ключові конкурентоспроможні позиції відповідно до обраної альтернативи:

- спеціалізація на Skyline/Pareto-фільтрації та багатокритеріальному відборі;
- технологічна відповідність потребам enterprise-сегмента (Java, Spring, REST, інтеграція з наявними СУБД);
- гнучкі моделі розгортання (on-premise, хмарний сервіс, гібридні сценарії);
- нижча загальна вартість володіння порівняно з важкими аналітичними СУБД;
- можливість подальшої кастомізації під галузеві рішення.

Базовою стратегією розвитку обрано стратегію диференціації у спеціалізованій ніші**:** створення продукту з чітко визначеною унікальною цінністю (готовий Skyline/Pareto-движок), який доповнює наявні в клієнтів BI

та ML-рішення, забезпечуючи їх прозорим та формалізованим багатокритеріальним відбором альтернатив.

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Так, оскільки у сегменті спеціалізованих Java-рішень Skyline/Pareto-фільтрації з орієнтацією на B2B-інтеграції його можна розглядати як одного з перших нішевих гравців.	Так, стратегія передбачає переважно формування нового попиту та перехід частини споживачів від існуючих конкурентів	Ні, компанія не планує пряме копіювання, проте відтворює наявний у конкурентів функціонал	Стратегія заняття конкурентної ніші

Таблиця 4.18. — Визначення стратегії позиціонування

№	**Вимоги до товару цільової аудиторії**	**Базова стратегія розвитку**	**Ключові конкурентоспроможні позиції власного стартап-проєкту**	**Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту**
1	Простота інтеграції в Java-застосунки, повнота документації, стабільний API для розробників	Стратегія диференціації у спеціалізованій ніші (Java SDK + open core)	Використання поширених технологій (Java, Spring), наявність готових прикладів, гнучка архітектура для вбудовування	«Стандартний інструмент Skyline для Java-розробників», «бібліотека, що економить час і зменшує ризики власної реалізації»
2	Надійність, прогнозований економічний ефект, підтримка і сервіс, відповідність вимогам безпеки	Концентричне охоплення ринку з фокусом на B2B-клієнтів	Можливість on-premise-розгортання, підтримка кількох моделей ліцензування, сервісна підтримка та консалтинг	«Надійний модуль багатокритеріального відбору для корпоративних систем», «інструмент підвищення якості бізнес-рішень»

Закінчення таблиці 4.18

№	**Вимоги до товару цільової аудиторії**	**Базова стратегія розвитку**	**Ключові конкурентоспроможні позиції власного стартап-проєкту**	**Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту**
3	Прозорість логіки прийняття рішень, можливість пояснення результатів, зручність аналізу	Розвиток інтеграцій із BI та аналітичними платформами	Формальний Skyline/Pareto-підхід, можливість пояснення, чому та чи інша альтернатива є оптимальною, експорт результатів до BI	«Прозорий механізм формування Парето-оптимальних наборів», «інструмент explainable-аналітики»

Відповідно до проведеного аналізу можна зробити висновок, що початковим позиціонуванням стартап-проєкту має бути представлення його як стандартного спеціалізованого інструменту Skyline/Pareto-фільтрації для Java- та B2B-рішень, що забезпечує просту інтеграцію, прозору логіку багатокритеріального відбору та економічно доцільну альтернативу важким аналітичним платформам.

4.5 Розроблення маркетингової програми стартап-проєкту

Таблиця 4.19. — Визначення ключових переваг концепції потенційного товару

№	**Потреба**	**Вигода, яку пропонує товар**	**Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)**
1	Потреба у формальному багатокритеріальному відборі (Skyline/Pareto)	Автоматизоване формування набору Pareto-оптимальних рішень із великої кількості альтернатив	Спеціалізація продукту на Skyline-функціоналі, наявність кількох оптимізованих алгоритмів, підтримка кластеризації для підвищення зручності аналізу
2	Потреба у швидкому впровадженні функціоналу без значних витрат на розробку	Скорочення термінів і витрат на створення та підтримку власних реалізацій Skyline	Готовий Java SDK і можливий B2B API, приклади інтеграції, можливість використання з існуючими СУБД без їх заміни
3	Потреба у прозорій та пояснюваній логіці прийняття рішень	Чітке обґрунтування, чому альтернатива входить до Skyline-множини або виключається з неї	Використання формальної концепції домінування, інструменти для пояснення та візуалізації результатів; перевага над «чорними скриньками» ML-моделей

Закінчення таблиці 4.19

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
4	Потреба у гнучкому та економічно доцільному рішенні	Оптимальне співвідношення вартості та можливостей за рахунок вузької спеціалізації	Нижча вартість володіння порівняно з аналітичними СУБД; можливість масштабування витрат залежно від фактичного використання (ліцензійні/підпискові моделі)

Таблиця 4.21. — Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни
безкоштовні open-source пакети для мов Python/R та власні реалізації Skyline у компаніях	аналітичні СУБД та комплексні платформи з підтримкою Skyline/Pareto з високою вартістю ліцензій та супроводу	корпоративні клієнти середнього та великого бізнесу	нижня межа – визначається сумою змінних витрат та мінімально прийнятною нормою прибутку;

Закінчення таблиці 4.20

Рівень цін на товари-замінники	**Рівень цін на товари-аналоги**	**Рівень доходів цільової групи споживачів**	**Верхня та нижня межі встановлення ціни**
традиційні BI-рішення, що частково вирішують подібні задачі	спеціалізовані модулі рекомендацій/скорингу в рамках великих ML/AI-платформ.	розробницькі команди, що працюють у проєктах із достатнім фінансуванням, де виправдано придбання ліцензій на інструменти, які знижують ризики та скорочують терміни розроблення.	верхня межа – обмежується рівнем цін на комплексні аналітичні рішення (аналітичні СУБД, ML-платформи) та готовністю клієнтів сплачувати за спеціалізований модуль; доцільно встановлювати ціну суттєво нижче вартості впровадження повноцінної аналітичної СУБД при збереженні відчутної економії для замовника.

Таблиця 4.21 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>1. Товар за задумом</td><td colspan="3">Спеціалізований програмний продукт, призначений для реалізації Skyline/Pareto-фільтрації та багатокритеріального відбору альтернатив у корпоративних інформаційних системах.</td></tr>
<tr><td rowspan="7">2. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>Програмна реалізація у вигляді Java-бібліотеки та, за потреби, окремого мікросервісу</td><td>Нм</td><td>Тх, Тл</td></tr>
<tr><td>Підтримка кількох Skyline-алгоритмів та кластеризації результатів</td><td>Нм</td><td>Тл</td></tr>
<tr><td>Інтерфейси інтеграції (Java API, REST API, адаптери для джерел даних)</td><td>Нм</td><td>Тх, Ор</td></tr>
<tr><td>Сумісність із поширеними стеком і СУБД</td><td>Нм</td><td>Тх, Тл</td></tr>
<tr><td>Документація, приклади використання, шаблони проєктів</td><td>Нм</td><td>Ор</td></tr>
<tr><td>Гнучка ліцензійна політика (перпетуальні ліцензії, підписка, open-core)</td><td>Нм</td><td>Е, Ор</td></tr>
</table>

Закінчення таблиці 4.21

Рівні товару	Сутність та складові
3. Товар із підкріпленням	До продажу: наявна повна документація, акції на придбання декількох ліцензій, знижки для певних сегментів на покупку товару
	Після продажу: додаткова підтримка спеціалістів налаштування, підтримка з боку розробника
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності, патент	

Таблиця 4.22. — Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	**Функції збуту, які має виконувати постачальник товару**	**Глибина каналу збуту**	**Оптимальна система збуту**
тривалий цикл прийняття рішень, участь кількох стейкхолдерів	забезпечення інтеграційної та післяпродажної підтримки	переважно прямі та однорівневі канали збуту	прямі продажі ключовим корпоративним клієнтам

Закінчення таблиці 4.22

Специфіка закупівельної поведінки цільових клієнтів	**Функції збуту, які має виконувати постачальник товару**	**Глибина каналу збуту**	**Оптимальна система збуту**
потреба у пілотних впровадженнях, прототипах, доказі економічного ефекту	проведення демонстрацій	можливе використання онлайн-каналів, для залучення розробників та малих клієнтів	партнерський канал
увага до наявності технічної підтримки, референсів, кейсів	укладення ліцензійних договорів і угод про супровід		онлайн-канали для поширення базової/спільнотної версії продукту та генерації лідів

Таблиця 4.23 – Концепція маркетингових комунікацій

№	**Специфіка поведінки цільових клієнтів**	**Канали комунікацій, якими користуються цільові клієнти**	**Ключові позиції, обрані для позиціону-вання**	**Завдання рекламного повідомлення**	**Концепція рекламного звернення**
1	Розробники та архітектори Java-систем: активно користуються професійним и ресурсами, орієнтуються на технічні характеристи ки, приклади коду, open-source	Професійні форуми та спільноти, GitHub, конференції та мітапи (Java, data engineering), технічні блоги, профільні соцмережі (LinkedIn, Twitter)	«Стандартн ий Skyline-інструмент для Java», «простота інтеграції, економія часу розробки»	Поінформувати про існування спеціалізован ого Skyline-рішення, показати простоту його використання і перевагу над власними реалізаціями	Лаконічні технічні повідомлення зі зрозумілими прикладами коду: «Підключіть Skyline за декілька рядків Java-коду», посилання на репозиторій і документацію

Закінчення таблиці 4.23

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціону-вання	Завдання рекламного повідомлення	Концепція рекламного звернення
2	Бізнес-клієнти (керівники напрямів, продакт-менеджери, IT-директори): орієнтовані на бізнес-результати, ризики, окупність інвестицій	Ділові заходи (галузеві конференції, вебінари), тематичні публікації, персональні презентації, LinkedIn, e-mail-маркетинг, партнерські заходи з інтеграторами	«Надійний модуль багатокритеріального відбору для корпоративних систем», «підвищення якості рішень і прозорість логіки вибору»	Сформувати розуміння економічних та операційних переваг від впровадження Skyline-підходу, стимулювати запуск пілотних проектів	Аналітичні кейси й історії успіху: «Як багатокритеріальний відбір зменшив час прийняття рішення і збільшив конверсію», демонстрація фінансових показників та прозорості рішення

Як результат було створено ринкову (маркетингову) програму, що включає в себе визначення ключових переваг концепції потенційного товару,

опис моделі товару, визначення меж встановлення ціни, формування системи збуту та концепцію маркетингових комунікацій.

Висновки до розділу

В четвертому розділі описано стратегії та підходи з розроблення стартап-проєкту, визначено наявність попиту, динаміку та рентабельність роботи ринку, як висновок було вказано що існує можливість ринкової комерціалізації проєкту. Розглянувши потенційні групи клієнтів, бар'єри входження, стан конкуренції та конкурентоспроможність проєкту було встановлено що проєкт є перспективним. Розглянуто та вибрано альтернативу впровадження стартап-проєкту та доведено доцільність подальшої імплементації проєкту.

ВИСНОВКИ

У ході виконання магістерської дисертації було розглянуто питання, пов’язані з формуванням Skyline-вибірок для багатовимірних даних, аналізом існуючих методів багатокритеріального прийняття рішень, дослідженням проблем масштабованості та явища «прокляття розмірності» (CoD), а також особливостями інтеграції Skyline-підходу в інформаційні системи.

На основі даних, отриманих в процесі аналізу, сформульовано задачу розробки методу формування Skyline-вибірок, який забезпечує ефективну обробку великих обсягів багатовимірних даних, мінімізує вплив CoD та проблем масштабованості, а також може бути реалізований у вигляді універсальної бібліотеки, придатної для інтеграції з Java-застосунками та JDBC-сумісними джерелами даних.

Для розробки методу та програмного забезпечення було використано мову програмування Java, стандартні колекції та засоби обробки потоків даних (Java Stream API), бібліотеку ELKI для реалізації методів кластеризації та вимірювання відстаней між рішеннями, а також інструменти для взаємодії з базами даних на основі JDBC. Все перераховане є широко вживаними та безкоштовними інструментами, що дозволить спростити впровадження розробленого рішення в існуючі інформаційні системи, знизити бар’єри входу для потенційних користувачів та забезпечити подальший розвиток продукту в open-source та комерційних форматах.

Розроблена модель формування Skyline-вибірок поєднує адаптивний вибір алгоритмів Parallel BNL, SkyCell та Randomized Multi-Pass залежно від розмірності та характеру даних, механізми попередньої обробки та кластеризації результатів, а також підтримку пакетного та потокового режимів роботи, що дає змогу отримувати Парето-оптимальні множини з урахуванням вимог продуктивності та масштабованості.

Розроблено програмне забезпечення у форм-факторі Java-бібліотеки Skyline/Pareto-фільтрації, яке містить модулі для роботи з різними джерелами даних, механізми кешування та кластеризації результатів, а також стабільний API для інтеграції з корпоративними інформаційними системами.

Проведений маркетинговий аналіз стартап-проєкту. Проведено опис ідеї проєкту, технологічний аудит ідеї проєкту, аналіз ринкових можливостей запуску стартап-проєкту. Розроблено ринкову стратегію проєкту та маркетингову програму стартап-проєкту з урахуванням цільових сегментів (Java-розробники, корпоративні B2B-клієнти, аналітичні підрозділи), конкурентного середовища та можливостей позиціонування як спеціалізованого технологічного постачальника рішень Skyline/Pareto-фільтрації.

Результати роботи над магістерською дисертацією опубліковані у статті.

Наукова новизна одержаних результатів магістерської дисертації:

вперше:

- запропоновано метод формування Skyline-вибірок для об’ємних багатовимірних даних у середовищі Java, який поєднує адаптивний вибір Skyline-алгоритмів, механізми кластеризації результатів та врахування проблеми CoD;
- надана можливість використовувати спеціалізовану Java-бібліотеку Skyline/Pareto-фільтрації як готовий модуль багатокритеріального відбору для інтеграції в корпоративні інформаційні системи з JDBC-сумісними базами даних;

удосконалено:

- модель програмної реалізації Skyline-вибірок у вигляді ізольованої Java-бібліотеки з чітко визначеним API, підтримкою кількох алгоритмів та сценаріїв використання, що спрощує інтеграцію та супровід;
- підходи та методи обробки пакетної обробки великих масивів даних з використанням кешу в Java;
- інтерактивний підхід до дослідження великих об’ємів даних;

здобуло подальший розвиток:

- алгоритми формування Skyline-вибірок;
- використання алгоритмів кластеризації для характеризування великих вибірок даних.

Практична значимість одержаних результатів полягає у розробці програмного засобу Skyline-фільтрації та багатокритеріальної оцінки альтернатив, який може бути інтегрований у прикладні системи для підтримки прийняття рішень у таких галузях, як електронна комерція, фінансовий скоринг, логістика, туристичні сервіси та аналітичні платформи, забезпечуючи підвищення якості рішень, прозорість логіки відбору та ефективне використання великих масивів даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) P. Ciaccia and D. Martinenghi. Reconciling skyline and ranking queries. PVLDB, 2017. P.1454–1465.

2) K. Mouratidis, K. Li and B. Tang. Marrying Top-k with Skyline Queries: Relaxing the Preference Input while Producing Output of Controllable Size. Proceedings of the 2021 International Conference on Management of Data (SIGMOD'21), June 20–25, 2021. P. 14.

3) Xie, M., Wong, R., Li, J., Long, C., Lall, A.: "Efficient k-regret query algorithm with restriction-free bound for any dimensionality". Proceedings of the 2018 ACM International Conference on Management of Data, 2018.

4) Paolo Ciaccia, Davide Martinenghi. Flexible Skylines: Dominance for Arbitrary Sets of Monotone Functions. ACM Trans, 2020. P. 18-45 .

5) Kyriakos Mouratidis, Bo Tang. Exact Processing of Uncertain Top-k Queries in Multi-criteria Settings. Proc. VLDB Endow, 2018. P. 866-879.

6) Martin Farach-Colton, Meng Li, and Meng-Tsung Tsai. Streaming algorithms for planar convex hulls. In International Symposium on Algorithms and Computation (ISAAC), 2018. P. 47:1–47:13.

7) Kenneth S. Bøgh, Sean Chester, Darius Šidlauskas, and Ira Assent. Template skycube algorithms for heterogeneous parallelism on multicore and GPU architectures. In SIGMOD, 2020. P. 447–462.

8) Christos Kalyvas. A Survey of Skyline Query Processing. Cornell University, 2018. P. 17-21.

9) C. J. Date. An Introduction to Database Systems 8th edition. Pearson, 2003. P.34-60.

10) MySQL // MySQL. URL: https://www.mysql.com/ (дата звернення:

27.07.2025).

11) Lukas Grasmann, Reinhard Pichler, Alexander Selzer. Integration of Skyline Queries into Spark SQL. 6th International Conference on Extending Database Technology (EDBT), 2023.

12) Exasol // Exasol docs. URL: https://docs.exasol.com/ (дата звернення: 27.07.2025).

13) Skyline queries in Python // Max Halford Blog. URL: https://shorturl.at/8bHNI/ (дата звернення: 27.07.2025).

14) Searching for the best compromise: “Skyline” queries // cybertec-postgresql.com. URL: https://shorturl.at/ooqZu/ (дата звернення: 27.07.2025).

15) Spring Batch // Spring Guides. URL: https://spring.io/projects/spring-batch (дата звернення: 29.07.2025).

16) Chuanwen Li, Yu Gu, Jianzhong Qi, Ge Yu. SkyCell: A Space-Pruning Based Parallel Skyline Algorithm. The University of Melbourne, Australia, 2021.

17) Chuanwen Li, Yu Gu, Jianzhong Qi, Ge Yu. SkyCell: Parallel Skyline Processing Using Space Pruning on GPU. The University of Melbourne, Australia, 2021.

18) Timothy M. Chan, Saladi Rahul. Simple Multi-Pass Streaming Algorithms for Skyline Points and Extreme Points. University of Illinois at Urbana-Champaign, IL, USA, 2021.

19) О. Швець. Занурення в Патерни проектування. Київ : Refactoring.Guru, 2021. 264 с.

20) Preeti Arora, Shipra Varshney, Deepali Virmani: Analysis of K-Means and K-Medoids Algorithm For Big Data. Procedia Computer Science. 2016. Vol. 78, No 4(54). P. 2-6. DOI: 10.1016/j.procs.2016.02.095.

ДОДАТОК А

Результати перевірки роботи на співпадіння

StrikePlagiarism.com

Дата звіту 12/5/2025
Дата редагування 12/5/2025

Документ прийнятий

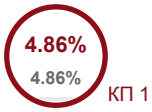
Звіт подібності

Метадані

Назва організації
National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute
Заголовок
ІП-42мп_Кривоносюк_ПЗ
Автор Науковий керівник / Експерт
КривоносюкСтеценко І.В.
підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



9403
Кількість слів

69709
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		7
Інтервали	A→	0
Мікропробіли	·	30
Білі знаки		0
Парафрази (SmartMarks)	a	35

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Метод та програмний засіб гейміфікації навчання в сфері інженерії програмного забезпечення [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	82 0.87 %
2	Метод та програмний засіб гейміфікації навчання в сфері інженерії програмного забезпечення [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	50 0.53 %

3	Архітектурне рішення для забезпечення аналізу і моделювання сигналів від епідермічних сенсорів [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	28 0.30 %
4	Метод та програмний засіб гейміфікації навчання в сфері інженерії програмного забезпечення [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	28 0.30 %
5	Метод та програмний засіб гейміфікації навчання в сфері інженерії програмного забезпечення [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	27 0.29 %
6	Вебсервіс інтелектуальної агрегації погодних даних [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	22 0.23 %
7	Вебсервіс інтелектуальної агрегації погодних даних [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	21 0.22 %
8	Метод та програмний засіб гейміфікації навчання в сфері інженерії програмного забезпечення [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	21 0.22 %
9	Вебсервіс інтелектуальної агрегації погодних даних [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	17 0.18 %
10	Вебсервіс інтелектуальної агрегації погодних даних [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	15 0.16 %

з домашньої бази даних (4.33 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Метод та програмний засіб гейміфікації навчання в сфері інженерії програмного забезпечення [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**254 (11) 2.70 %**
2	Вебсервіс інтелектуальної агрегації погодних даних [redacted] 3/16/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**87 (5) 0.93 %**
3	Архітектурне рішення для забезпечення аналізу і моделювання сигналів від епідермічних сенсорів [redacted] 3/15/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**56 (3) 0.60 %**
4	Програмне забезпечення обробки природної мови для потоків текстових даних в режимі реального часу [redacted] 3/15/2025 **National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)**	**10 (1) 0.11 %**

з програми обміну базами даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з Інтернету (0.53 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
5	**https://ela.kpi.ua/bitstream/123456789/58761/1/Fedor_bakalavr.pdf**	**36 (3) 0.38 %**
6	http://lib.kart.edu.ua/bitstream/123456789/14802/1/%D0%9A%D0%BE%D0%BD%D1%81%D0%BF%D0%B5%D0%BA%D1%82%20%D0%BB%D0%B5%D0%BA%D1%86%D1%96%D0%B9.pdf	14 (2) 0.15 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)