КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (31), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.31.996
УДК 004.42:004.43:004.45:004.77

Аронов Андрій Олексійович
кандидат технічних наук, доцент кафедри технологій цифрового розвитку,
Державний університет інформаційно-комунікаційних технологій, Київ, Україна
ORCID: 0009-0000-7868-8341
a.aronov@duikt.edu.ua

Гавор Артур Станіславович
викладач кафедри Технологій цифрового розвитку,
Державний університет інформаційно-комунікаційних технологій, Київ, Україна
ORCID: 0009-0002-9705-1666
a.havor@duikt.edu.ua

Герцюк Микола Модестович
доктор філософії, доцент кафедри Технологій цифрового розвитку,
Державний університет інформаційно-комунікаційних технологій, Київ, Україна
ORCID: 0000-0003-2946-9673
m.gertsiuk@duikt.edu.ua

Гордієнко Катерина Олександрівна
старший викладач кафедри Технологій цифрового розвитку,
Державний університет інформаційно-комунікаційних технологій, Київ, Україна
ORCID: 0009-0002-5997-9682
k.hordienko@duikt.edu.ua

Ніщеменко Дмитро Олександрович
викладач кафедри Технологій цифрового розвитку,
Державний університет інформаційно-комунікаційних технологій, Київ, Україна
ORCID: 0009-0006-1781-4109
d.nishchemenko@duikt.edu.ua

АНАЛІЗ ЕФЕКТИВНОСТІ СТРУКТУР ДАНИХ ТА БАЗ ДАНИХ У РОЗПОДІЛЕНИХ СИСТЕМАХ НА ОСНОВІ PYTHON, JAVA ТА БЛОКЧЕЙН-ТЕХНОЛОГІЙ

Анотація. У статті проведено порівняльний аналіз лінійних та нелінійних структур даних із використанням мов програмування Python і Java у контексті розподілених систем. Розглянуто вплив вибору структури на продуктивність, масштабованість та узгодженість даних при роботі з розподіленими базами даних і блокчейн-технологіями. Наведено порівняння ключових характеристик структур даних та сховищ, оцінено їх переваги та обмеження в різних сценаріях обробки інформації. Дослідження дозволяє визначити оптимальні підходи до побудови ефективних систем зберігання та обробки даних. Python відзначається простотою та розвинутими вбудованими колекціями, але обмежується інтерпретатором у високопродуктивних сценаріях. Java ж завдяки віртуальній машині JVM і підтримці багатопоточності, краще підходить для систем з високими навантаженнями. У контексті розподілених баз даних ці відмінності проявляються в роботі реляційних і NoSQL-рішень, які використовують деревоподібні та геш-структури для індексації, тоді як блокчейн базується на геш-деревах, забезпечуючи незмінність, але знижуючи швидкодію. Отже, порівняльний аналіз лінійних та нелінійних структур даних із використанням Python і Java, а також дослідження їх застосування у розподілених базах даних і блокчейні, є актуальним завданням. Це дослідження дає змогу визначити сильні та слабкі сторони різних підходів, а також сформулювати практичні рекомендації для побудови ефективних розподілених систем.

Ключові слова: розподілені системи; структури даних; розподілені бази даних; Python; Java; блокчейн; продуктивність; масштабованість.

ВСТУП

Сучасні інформаційні системи працюють у середовищі, де обсяг та швидкість обробки даних зростають експоненційно. Для забезпечення надійності, масштабованості та продуктивності особливого значення набуває правильний вибір структур даних, технологій зберігання а також їх взаємодії. Лінійні структури, такі як: масиви, списки, стеки, черги, забезпечують простоту організації та швидкий доступ до даних у послідовному порядку, тоді як нелінійні (дерева, графи, геш-таблиці) дозволяють реалізувати складніші моделі пошуку, маршрутизації та оптимізації ресурсів.

Розподілені системи потребують ретельного вибору структури даних і системи зберігання: від цього залежить продуктивність, доступність і узгодженість даних. Мови реалізації — зокрема Python і Java — та архітектура збереження, такі як: розподілені NoSQL, блокчейн, суттєво впливають на поведінку системи під навантаженням.

Таким чином метою дослідження є систематизація знань про те, як вибір структури даних і платформи Python чи Java відображається на продуктивності та комплексний аналіз який допоможе обрати найбільш ефективну комбінацію.

ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

Аналіз ефективності структур даних є ключовим напрямом у проєктуванні сучасних програмних і розподілених систем. Лінійні структури даних — масиви, списки, черги, стеки — забезпечують послідовну організацію даних і добре підходять для задач, що потребують упорядкованого доступу [1]. Нелінійні структури, такі як дерева, графи та геш-таблиці, дозволяють ефективно вирішувати задачі індексації, маршрутизації та оптимізації зв’язків між елементами [2].

У контексті сучасних мов програмування особливу увагу приділяють реалізації цих структур у Python та Java. У Python вбудовані типи даних такі як list, dict, set, tuple базуються на динамічних масивах і геш-таблицях, що забезпечує гнучкість і швидку розробку, проте продуктивність часто обмежується інтерпретатором CPython і наявністю Global Interpreter Lock [3]. У Java реалізація структур ArrayList, LinkedList, HashMap, TreeMap оптимізована для багатопоточності та стабільної роботи під навантаженням завдяки JVM та JIT-компіляції [4]. Порівняльні дослідження підтверджують, що Java демонструє вищу пропускну здатність у CPU-інтенсивних сценаріях [5], тоді як Python переважає в задачах асинхронної обробки даних і швидкого прототипування [6].

Значна увага у науковій літературі приділяється використанню структур даних у розподілених системах зберігання. Реляційні СУБД такі як: PostgreSQL, MySQL, використовують B-дерева для індексації даних [7, 15], тоді як NoSQL-рішення, наприклад MongoDB, ґрунтуються на LSM-деревах, які оптимізують роботу з великими потоками записів і забезпечують масштабування [8].

Окремий напрям досліджень стосується блокчейн-технологій, які базуються на геш-функціях і Merkle-деревах [9]. Блокчейн-бази даних демонструють високу безпеку, гарантуючи цілісність та незмінність даних, проте поступаються традиційним СУБД за швидкодією через необхідність криптографічної перевірки кожного блоку та масштабованістю [10]. Тому все частіше досліджуються гібридні рішення, що поєднують швидкодію розподілених NoSQL-сховищ [11] із прозорістю блокчейн-записів [12-13].

Узагальнюючи, більшість робіт зосереджується на окремих аспектах — алгоритмах, мовах програмування або типах баз даних [14]. Проте комплексне

дослідження, яке поєднує порівняльний аналіз лінійних і нелінійних структур даних у Python і Java в умовах розподілених баз даних та блокчейн-сховищ, залишається недостатньо опрацьованим. Це обґрунтовує актуальність даного дослідження та визначає його наукову новизну.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Метою дослідження було оцінити ефективність лінійних і нелінійних структур даних у Python та Java в умовах розподілених систем зберігання — традиційних (SQL/NoSQL) і блокчейн-орієнтованих. Для цього було проведено серію тестів, спрямованих на вимірювання швидкодії та масштабованості при виконанні типових операцій над даними. Тестування проведено на персональному комп'ютері Intel Core i5 3.0 GHz, 16Gb RAM DDR4, x64 Windows 11 Pro. Для кожної структури даних виконувались 10000000 операцій. Використовувалось середнє значення часу за 10 виконань алгоритму.

У тестах на Java використані ArrayList для списків, HashMap для хеш-таблиць та власна реалізація бінарного дерева. На Python відповідні структури реалізовані через вбудовані list, dict та власну реалізацію бінарного дерева.

Реалізація додавання, пошуку, видалення елементу бінарного дерева на Java:

```
static class Node {
        int key;
        Node left, right;
        Node(int k) { key = k; left = right = null; }     }
    static Node insert(Node root, int key) {
        if (root == null) return new Node(key);
        if (key < root.key) root.left = insert(root.left, key);
        else root.right = insert(root.right, key);
        return root;     }
    static Node search(Node root, int key) {
        if (root == null || root.key == key) return root;
        if (key < root.key) return search(root.left, key);
        return search(root.right, key);     }
    static Node delete(Node root, int key) {
        if (root == null) return null;
        if (key < root.key) root.left = delete(root.left, key);
        else if (key > root.key) root.right = delete(root.right, key);
        else {
            if (root.left == null) return root.right;
            if (root.right == null) return root.left;
            Node temp = root.right;
            while (temp.left != null) temp = temp.left;
            root.key = temp.key;
            root.right = delete(root.right, temp.key);         }
        return root;    }
```

Реалізація додавання, пошуку, видалення елементу бінарного дерева на Python:

```
class Node:
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None
def bst_insert(root, key):
    if root is None:
        return Node(key)
    if key < root.key:
```

ISSN 2663 - 4023

```
            root.left = bst_insert(root.left, key)
        else:
            root.right = bst_insert(root.right, key)
        return root
    def bst_search(root, key):
        if root is None or root.key == key:
            return root
        if key < root.key:
            return bst_search(root.left, key)
        return bst_search(root.right, key)
    def bst_delete(root, key):
        if root is None:
            return None
        if key < root.key:
            root.left = bst_delete(root.left, key)
        elif key > root.key:
            root.right = bst_delete(root.right, key)
        else:
            if root.left is None:
                return root.right
            if root.right is None:
                return root.left
            temp = root.right
            while temp.left:
                temp = temp.left
            root.key = temp.key
            root.right = bst_delete(root.right, temp.key)
        return root
```

Для кожної структури проводилися операції вставки, пошуку та видалення. вибір Python та Java дозволяє порівняти легкість реалізації та продуктивність інтерпретованої і компільованої мови, а різні структури даних дають змогу оцінити ефективність лінійних і нелінійних підходів.

Частина коду алгоритму мовою Java:

```
import java.util.*;
public class DataStructBench {
    // …
    public static void main(String[] args) {
        int N = 10000000;

        long start = System.nanoTime();
        List<Integer> list = new ArrayList<>();
            for (int i = 0; i < N; i++) list.add(i);
            for (int i = 0; i < 1000; i++) list.get(new Random().nextInt(N));
            for (int i = 0; i < 1000; i++) list.remove(list.size() - 1);
        long end = System.nanoTime();
        System.out.println("ArrayList time: " + (end - start) / 1e6 + " ms");

        start = System.nanoTime();
        Map<Integer, Integer> map = new HashMap<>();
            for (int i = 0; i < N; i++) map.put(i, i);
            for (int i = 0; i < 1000; i++) map.get(new Random().nextInt(N));
            for (int i = 0; i < 1000; i++) map.remove(new Random().nextInt(N));
        end = System.nanoTime();
        System.out.println("HashMap time: " + (end - start) / 1e6 + " ms");

        start = System.nanoTime();
        Node root = null;
        for (int i = 0; i < N; i++) root = insert(root, i);   // вставка
        for (int i = 0; i < 1000; i++) search(root, rand.nextInt(N)); // пошук
        for (int i = 0; i < 1000; i++) root = delete(root, rand.nextInt(N));
        end = System.nanoTime();
```

```
        System.out.println("BTree time: " + (end-start)/1e6 + " ms");
 } }
```

Частина коду алгоритму на мові Python:

```
import timeit, random
def list_operations(n=10000000):
    data = []
    for i in range(n):
        data.append(i)
    for k in range(1000):
        k = random.choice(data)
    for k in range(1000):
        data.pop()
print("List:", timeit.timeit(list_operations, number=1))

def dict_operations(n=10000000):
    data = {}
    for i in range(n):
        data[i] = i
    for k in range(1000):
        k = data.get(random.randint(0, n - 1))
    for k in range(1000):
        data.pop(random.randint(0, n - 1), None)
print("Dict:", timeit.timeit(dict_operations, number=1))

// …
def bst_operations(n=1000000):
    root = None
    for i in range(n):
        root = bst_insert(root, i)
    for _ in range(1000):
        _ = bst_search(root, random.randint(0, n-1))
    for _ in range(1000):
        root = bst_delete(root, random.randint(0, n-1))
  print("BTree:", timeit.timeit(bst_operations, number=1))
```

Середні результати витраченого часу на виконання алгоритмів на Python і Java при 10 мільйонах елементів в структурах даних наведені у Таблиці 1.

Таблиця 1

Результати тестування продуктивності алгоритмів лінійних та нелінійних структур даних у Python і Java

№	Операція	Час виконання Python	Час виконання Java	Ефективність, %
1.	Список, вставка	11798 мс.	8415 мс.	40%
2.	Список, пошук	390 мс.	250 мс.	56%
3.	Список, видалення	1151 мс.	880 мс.	31%
4.	Геш-таблиця, вставка	9259 мс.	6112 мс.	51%
5.	Геш-таблиця, пошук	130 мс.	82 мс.	59%
6.	Геш-таблиця, видалення	170 мс.	110 мс.	55%
7.	Бінарне дерево, вставка	812 мс.	547 мс.	48%
8.	Бінарне дерево, пошук	750 мс.	520 мс.	44%
9.	Бінарне дерево, видалення	860 мс.	585 мс.	47%

У Python вставка у списки повільніша на 40% порівняно з Java, що пов’язано з інтерпретацією циклів у Python. Пошук та видалення теж демонструють перевагу Java через оптимізацію та швидший доступ до пам’яті.

При роботі з геш-таблицями Python dict показує гідну продуктивність, однак Java HashMap перевищує її у середньому на 50%, що пояснюється внутрішньою реалізацією геш-функцій.

Робота з бінарними деревами демонструє ефективну реалізацію у обох мовах у порівнянні з іншими структурами даних та має найменшу різницю у швидкості вставки між Java та Python.

За результатами витраченого часу на виконання алгоритмів під час локальних тестів Java демонструє стабільно вищу продуктивність для великих обсягів даних, проте Python зручний для швидкої розробки та прототипування.

Проте, незважаючи на нижчу швидкодію, Python показав значно вищу ефективність на етапі прототипування. Створення і тестування алгоритму пошуку у дереві займало близько 20 хвилин у Python проти понад 1 години у Java через необхідність компіляції, визначення типів і реалізації допоміжних класів. (див. Таб. 2).

Таблиця 2

Порівняння зручності процесу розробки та тестування

№	Назва	Python	Java
1.	Час реалізації	20 хв.	70 хв.
2.	Складність коду	низька	висока
3.	Підтримка сторонніх бібліотек для аналізу та тестування	широка	обмежена

Це свідчить про перевагу Python для етапів моделювання та дослідницької розробки, коли швидкість перевірки гіпотези важливіша за продуктивність виконання. Таким чином, Python ефективніший у створенні експериментальних моделей структур даних і сценаріїв розподіленої обробки, тоді як Java доцільно використовувати у фінальних високопродуктивних реалізаціях.

Окрім локального тестування, було змодельовано роботу структур даних у розподілених сховищах

Для моделювання розподілених сценаріїв було використано три типи систем зберігання:

- MySQL – приклад реляційної SQL-бази даних;
- MongoDB – представник NoSQL-сховищ документного типу;
- Hyperledger Fabric – платформа блокчейн-орієнтованого зберігання.

Обидві мови програмування і Python і Java використовувалися для підключення до цих систем за допомогою відповідних бібліотек:

- у Java взаємодія з MySQL реалізовувалася через JDBC, у Python для MySQL застосовувався модуль mysql.connector;
- для роботи з MongoDB Java-реалізація базувалася на офіційному MongoDB Java Driver, у Python за допомогою бібліотеки PyMongo;
- у випадку Hyperledger Fabric, використовувався Fabric Java SDK у Java для формування і підтвердження транзакцій у блоках, тоді як у Python застосовувався Fabric Python SDK.

Такий підхід дозволив оцінити вплив вибору мови на затримку підтвердження транзакцій та обробку геш-структур у блокчейні.

Результати порівняння середніх показників часу транзакцій та пропускної здатності при 100 000 запитах, наведені у Таблиці 3.

Таблиця 3

Порівняння продуктивності та узгодженості у розподілених системах зберігання (MySQL, MongoDB, Hyperledger Fabric)

№	Система, тип структури	Пропускна здатність	Затримка
1.	MySQL (SQL)	6200 транз./с.	3.5 мс.
2.	MongoDB (NoSQL)	8300 транз./с.	2.6 мс.
3.	Hyperledger Fabric (Блокчейн)	2100 транз./с.	8.9 мс.

MySQL забезпечує високу узгодженість транзакцій, але має обмежену горизонтальну масштабованість через реплікацію. Найкращу пропускну здатність та швидкість обробки завдяки LSM-структурам та Sharding демонструє MongoDB. Hyperledger Fabric забезпечує максимальну узгодженість і безпеку через механізм консенсусу, але має найнижчу продуктивність і затримку, обмежену складністю перевірки Merkle-гешів.

Таким чином вибір сховища в першу чергу залежить від пріоритетів системи продуктивності та масштабованості (у випадку з MongoDB) або більшої узгодженості та безпеки (у випадку з Hyperledger Fabric).

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Отримані результати свідчать, що ефективність структур даних суттєво залежить як від мови програмування, так і від середовища зберігання інформації. Порівняльний аналіз показав, що Java демонструє стабільнішу продуктивність при виконанні великої кількості операцій над великими наборами даних завдяки ефективній реалізації багатопоточності та оптимізації пам’яті у JVM. Водночас Python виявив перевагу у швидкому прототипуванні та простоті реалізації алгоритмів, що робить його зручним для експериментального моделювання та навчальних цілей.

Під час тестування структур даних Java забезпечила швидше виконання операцій, тоді як Python продемонстрував кращу читабельність, зручність та швидкість реалізації. В залежності від завдання різниця у швидкодії між мовами може компенсуватись загальною швидкістю реалізації необхідного алгоритму.

У розподілених системах спостерігались різні результати залежно від типу сховища. MySQL виявив найвищу стабільність транзакцій, проте поступався MongoDB за пропускною здатністю. MongoDB краще масштабувалася при зростанні кількості запитів, тоді як Hyperledger Fabric характеризувався більшою затримкою через механізм консенсусу, проте забезпечує найвищий рівень узгодженості та безпеки даних.

Таким чином, використання Python доцільне для швидкого прототипування та аналітики, тоді як Java — для побудови масштабованих і високопродуктивних компонентів у розподілених системах зберігання даних, SQL-сховища доцільно використовувати у транзакційних системах, NoSQL — у високонавантажених додатках з неструктурованими даними, а блокчейн — у середовищах, де критичними є довіра та незмінність інформації..

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. QuickCodingExplanation. (2024, February 20). *Data structures overview: Array, stack, queue, linked-list, hash table, heap, binary tree*. Medium. https://quickcodingexplanation.medium.com/data-structures-overview-arrays-stack-queue-linked-list-hash-table-heap-binary-tree-7b88a5711a0b
2. Massachusetts Institute of Technology. (2020). *6.006 Introduction to algorithms — Lecture notes (binary trees, graphs, hashing)*. MIT OpenCourseWare. https://ocw.mit.edu/courses/6-006-introduction-to-algorithms-spring-2020/pages/lecture-notes/
3. Python Software Foundation. (n.d.). *Python 3 — Tutorial: Data structures*. Python.org. https://docs.python.org/3/tutorial/datastructures.html
4. Imaginary Cloud. (2025). *Java vs. Python: Key differences, performance, and use cases*. https://www.imaginarycloud.com/blog/python-vs-java
5. Ozcay, D. (2025, September 22). *Java vs Python 2025: Complete performance guide (with real benchmark results)*. Medium. https://medium.com/codeelevation/java-vs-python-2025-complete-performance-guide-with-real-benchmark-results-a122e50edfd5
6. Python Software Foundation. (n.d.). *asyncio — Asynchronous I/O*. Python.org. https://docs.python.org/3/library/asyncio.html
7. Kostenko, O. B., & Havrylenko, I. O. (2021). *Organization of databases and knowledge: Lecture notes*. Kharkiv National University of Municipal Economy.
8. Kozub, V. (2024). *Technology for improving storage efficiency in No-SQL databases*. *Information Technology and Society*, 14–22. https://doi.org/10.32689/maup.it.2024.3.2
9. Gate.io. (n.d.). *Merkle tree and Merkle root in blockchain*. https://www.gate.com/uk/learn/articles/merkle-tree-and-merkle-root-in-blockchain/166
10. McConaghy, T., et al. (2016). *BigchainDB: A scalable blockchain database — Whitepaper*. https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf
11. Zinchenko, O. V., Ishcheryakov, S. M., Prokopov, S. V., Serykh, S. O., & Vasylenko, V. V. (2020). *Cloud technologies*. State University of Telecommunications.
12. Shevchenko, S., Zhdanova Y., Skladannyi, P., & Ishchuk, M. (2025). Creation of a Blockchain Platform for Electronic Voting. *Electronic Professional Scientific Journal «Cybersecurity: Education, Science, Technique»*, *4*(28), 701–714. https://doi.org/10.28925/2663-4023.2025.28.860
13. Zhebka, V., et al. (2024). *Methodology for choosing a consensus algorithm for blockchain technology*. In Workshop on Digital Economy Concepts and Technologies (DECaT) (Vol. 3665, pp. 106–113).
14. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhyltsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science. https://doi.org/10.28925/9720213284km
15. Rzaieva, S. L., Skladannyi, P. M., Mashkina, I. V., & Kostiuk, Yu. V. (2025). A model for implementing role-based access control (RBAC) in a tiered data warehouse architecture**.** *Modern Information Protection*, (3), 137–149.

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (31), 2025
ISSN 2663 - 4023

Andrii Aronov
PhD in Engineering, Associate Professor, Department of Digital Development Technologies,
State University of Information and Communication Technologies, Kyiv, Ukraine
ORCID: 0009-0000-7868-8341
a.aronov@duikt.edu.ua

Artur Havor
Lecturer, Department of Digital Development Technologies,
State University of Information and Communication Technologies, Kyiv, Ukraine
ORCID: 0009-0002-9705-1666
a.havor@duikt.edu.ua

Mykola Hertsiuk
Doctor of Philosophy, Associate Professor, Department of Digital Development Technologies,
State University of Information and Communication Technologies, Kyiv, Ukraine
ORCID: 0000-0003-2946-9673
m.gertsiuk@duikt.edu.ua

Kateryna Hordiienko
Senior Lecturer, Department of Digital Development Technologies,
State University of Information and Communication Technologies, Kyiv, Ukraine
ORCID: 0009-0002-5997-9682
k.hordienko@duikt.edu.ua

Dmytro Nishchemenko
Lecturer, Department of Digital Development Technologies,
State University of Information and Communication Technologies, Kyiv, Ukraine
ORCID: 0009-0006-1781-4109
d.nishchemenko@duikt.edu.ua

ANALYSIS OF THE EFFICIENCY OF DATA STRUCTURES AND DATABASES IN DISTRIBUTED SYSTEMS BASED ON PYTHON, JAVA AND BLOCKCHAIN TECHNOLOGIES

Abstract. The article presents a comparative analysis of linear and nonlinear data structures using the Python and Java programming languages in the context of distributed systems. The impact of the choice of structure on performance, scalability, and data consistency when working with distributed databases and blockchain technologies is considered. The key characteristics of data structures and storages are compared, and their advantages and limitations in various information processing scenarios are assessed. The study allows us to determine the optimal approaches to building effective data storage and processing systems. Python is characterized by simplicity and developed built-in collections, but is limited to an interpreter in high-performance scenarios. Java, thanks to the JVM virtual machine and multithreading support, is better suited for systems with high loads. In the context of distributed databases, these differences are manifested in the work of relational and NoSQL solutions that use tree and hash structures for indexing, while blockchain is based on hash-trees, ensuring immutability, but reducing performance. The research allows us to formulate practical recommendations for building effective distributed systems, to identify the strengths and weaknesses of different approaches. Therefore, a comparative analysis of linear and nonlinear data structures using Python and Java, as well as research into their application in distributed databases and blockchain, is a relevant task.

Keywords: distributed systems, data structures, distributed databases, Python, Java, blockchain, performance, scalability.

REFERENCES

1. QuickCodingExplanation. (2024, February 20). *Data structures overview: Array, stack, queue, linked-list, hash table, heap, binary tree*. Medium. https://quickcodingexplanation.medium.com/data-structures-overview-arrays-stack-queue-linked-list-hash-table-heap-binary-tree-7b88a5711a0b
2. Massachusetts Institute of Technology. (2020). *6.006 Introduction to algorithms — Lecture notes (binary trees, graphs, hashing)*. MIT OpenCourseWare. https://ocw.mit.edu/courses/6-006-introduction-to-algorithms-spring-2020/pages/lecture-notes/
3. Python Software Foundation. (n.d.). *Python 3 — Tutorial: Data structures*. Python.org. https://docs.python.org/3/tutorial/datastructures.html
4. Imaginary Cloud. (2025). *Java vs. Python: Key differences, performance, and use cases*. https://www.imaginarycloud.com/blog/python-vs-java
5. Ozcay, D. (2025, September 22). *Java vs Python 2025: Complete performance guide (with real benchmark results)*. Medium. https://medium.com/codeelevation/java-vs-python-2025-complete-performance-guide-with-real-benchmark-results-a122e50edfd5
6. Python Software Foundation. (n.d.). *asyncio — Asynchronous I/O*. Python.org. https://docs.python.org/3/library/asyncio.html
7. Kostenko, O. B., & Havrylenko, I. O. (2021). *Organization of databases and knowledge: Lecture notes*. Kharkiv National University of Municipal Economy.
8. Kozub, V. (2024). *Technology for improving storage efficiency in No-SQL databases. Information Technology and Society*, 14–22. https://doi.org/10.32689/maup.it.2024.3.2
9. Gate.io. (n.d.). *Merkle tree and Merkle root in blockchain*. https://www.gate.com/uk/learn/articles/merkle-tree-and-merkle-root-in-blockchain/166
10. McConaghy, T., et al. (2016). *BigchainDB: A scalable blockchain database — Whitepaper*. https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf
11. Zinchenko, O. V., Ishcheryakov, S. M., Prokopov, S. V., Serykh, S. O., & Vasylenko, V. V. (2020). *Cloud technologies*. State University of Telecommunications.
12. Shevchenko, S., Zhdanova Y., Skladannyi, P., & Ishchuk, M. (2025). Creation of a Blockchain Platform for Electronic Voting. *Electronic Professional Scientific Journal «Cybersecurity: Education, Science, Technique»*, *4*(28), 701–714. https://doi.org/10.28925/2663-4023.2025.28.860
13. Zhebka, V., et al. (2024). *Methodology for choosing a consensus algorithm for blockchain technology*. In Workshop on Digital Economy Concepts and Technologies (DECaT) (Vol. 3665, pp. 106–113).
14. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhyltsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science. https://doi.org/10.28925/9720213284km
15. Rzaieva, S. L., Skladannyi, P. M., Mashkina, I. V., & Kostiuk, Yu. V. (2025). A model for implementing role-based access control (RBAC) in a tiered data warehouse architecture**.** *Modern Information Protection*, (3), 137–149.