DOI: https://doi.org/10.33216/1998-7927-2025-287-1-5-11

УДК 004.41:004.65

АНАЛІЗ МЕТОДІВ ПРЕДСТАВЛЕННЯ ІЄРАРХІЧНОЇ ІНФОРМАЦІЇ В ДОКУМЕНТО-ОРІЄНТОВАНИХ БАЗАХ ДАНИХ

Дьомін М.К., Полупан Ю.В.

ANALYSIS OF METHODS FOR REPRESENTING HIERARCHICAL INFORMATION IN DOCUMENT-ORIENTED DATABASES

Domin M.K.,Polupan Yu.V.

Вибір ефективного методу представлення ієрархічної інформації у базах даних є важливим завданням при проєктуванні програмних систем, оскільки ієрархічні структури даних використовуються в багатьох сферах, зокрема у системах рекомендацій, управлінні ланцюгами поставок та аналітичних рішеннях. У статті аналізуються різні методи зберігання ієрархічних структур у документо-орієнтованих базах даних на прикладі MongoDB та порівнюється їх ефективність.

При роботі з невеликою кількістю ієрархічних даних майже будь-який метод буде функціонувати з прийнятною швидкістю. Однак у випадках, коли система працює з сотнями і навіть тисячами ієрархій, неправильний вибір способу їх зберігання може суттєво вплинути на продуктивність системи, збільшити операційні витрати та ускладнити обробку даних.

У рамках дослідження проаналізувані методи представлення ієрархічних структур у документо-орієнтованих базах даних, виявлені їх переваги, недоліки та визначена їх швидкодія у реальних сценаріях використання.

Документо-орієнтовані бази даних, такі як MongoDB, надають гнучкі можливості для зберігання ієрархічних даних. У MongoDB можна використовувати кілька підходів до представлення ієрархій: метод збереження матриці суміжності, метод матеріалізованих маршрутів та метод вкладених множин.

У статі наведені результати тестування швидкодії зазначених методів для двох сценаріїв, а саме при використанні бази даних, встановленої на власний сервер, а також при використанні бази даних хмарного сервісу Microsoft Azure.

Вибір методу залежить від конкретних вимог до швидкості вибірки, задіяних операцій з ієрархіями та складності реалізації.

Метод збереження матриці суміжності показав себе набагато краще, ніж при роботі з реляційними базами даних і може використовуватися в багатьох сценаріях, оскільки він є найпростішим в реалізації. Але якщо для програмної системи характерні часті вибірки ієрархії або їх частин, то можна рекомендувати використання методу матеріалізованого маршруту оскільки він працює значно швидше, хоча і є більш складним у реалізації, та потребує більше пам'яті для збереження ієрархічної інформації. Метод вкладених множин не надає відчутних переваг для найбільш типових сценаріїв роботи. Дослідження є корисним для розробників, які проєктують системи, що працюють з великими обсягами ієрархічної інформації.

Ключові слова: *документо-орієнтовані бази даних, MongoDB, ієрархічна інформація, тестування швидкодії, Microsoft Azure, одиниці запиту.*

Вступ

Вибір ефективного методу представлення ієрархічної інформації у базах даних є надзвичайно важливим завданням у проектування програмних систем. Необхідність зберігання ієрархічної інформації виникає доволі часто. Але якщо при роботі з відносно невеликої кількістю ієрархічної інформації будь який метод представлення ієрархій буде працювати з задовільної швидкістю, то є завдання які потребують роботи з сотнями різних ієрархій і у цьому випадку вибір

найкращого методу може суттєво поліпшити швидкодію системи та знизити операційні витрати.

У якості прикладів таких завдань наведемо системи рекомендації контенту або управління ланцюгами поставок. Ієрархія рекомендацій відноситься до структурованого способу, у якому рекомендації генеруються та вдосконалюються, часто на основі кількох рівнів даних. Це зазвичай використовується в системах рекомендацій вмісту, таких як YouTube, Netflix і платформах електронної комерції, таких як Amazon. Ієрархія рекомендацій побудована з використанням кількох рівнів фільтрації, агрегації та персоналізації для динамічного вдосконалення пропозицій. Наприклад, для стримінгово сервісу можуть бути такі рівні ієрархії: популярні фільми у всьому світі, популярні фільми у країні, жанрові переваги користувача, відповідність історії переглядів, коригування рекомендацій на основі останніх дій користувача. І майже при кожному запиті користувача система повинна отримувати дані з ієрархії чи оновлювати деякі частини у існуючих ієрархічних структурах [1]. При управлінні ланцюгом поставок кожен продукт має специфікацію матеріалів (BillofMaterials), можливо із тисячами підкомпонентів. Компанії відстежують мільйони продуктів, що переміщуються через фабрики, склади та роздрібні торгові мережі [2]. Аналітика в режимі реального часу повинна обробляти тисячі дерев специфікації, щоб виявити проблеми з постачанням.

У власній практиці ми зіткнулися з необхідністю обрання найкращого методу представлення ієрархічної інформації при роботі над створенням бази даних інформаційно управлінських архітектур. Результати роботи по аналізу методів представлення ієрархічної інформації для реляційних баз даних опубліковані у роботі [3]. Також можна відзначити, що для завдання створення бази даних інформаційно-управлінських архітектур доволі добре підходять документо-орієнтовані системи керування базами даних, наприклад, MongoDB. Вимоги, що визначені при проєктуванні системи моніторингу інформаційно-управлінських архітектур до баз даних, що використовуватимуться системою, зазначені у [4]. Наприклад, для такої бази даних не важлива транзакційність, а гнучкість структури даних є великою перевагою. Першим етапом у визначенні доцільності використання документно-орієнтованої бази даних для системи моніторингу інформаційно-управлінських архітектур є визначення можливості ефективно зберігати і обробляти ієрархічну інформацію при використані такої бази даних.

Хоча задача виникла при роботі над створенням системи моніторингу інформаційно-управлінських архітектур, але дослідження має самостійну цінність і його результати можуть використовуватись для проєктування будь-яких систем, що потребують роботи з великою кількістю ієрархічної інформації, приклади яких вже були наведені у статті.

Загальний порівняльний аналіз швидкодії MongoDB у порівнянні з реляційними системами керування базами даних наведений у [5].

Можливі методи представлення ієрархічної інформації наведені у документації до системи керування базами даних MongoDB [6]. Там зазначені такі методи: збереження посилання на батьківський вузол, збереження посилання на дочірні вузли, збереження масиву батьківських вузлів, збереження матеріалізованих маршрутів, метод вкладених множин. Перші два методи – це варіанти зберігання інформації про графи, які неявно використовують таку структуру даних як матриця суміжності. Вони мають схожі методи роботи та швидкодію. Для цілей дослідження буде достатньо розглянути один з них, а саме метод, заснований на збереженні посилання на батьківський вузол. Збереження масиву батьківських вузлів є варіацією методу матеріалізованих маршрутів. Ці два методи так само мають схожу (майже ідентичну) реалізацію та схожу швидкодію. Але метод матеріалізованих маршрутів виявляється трошки швидшим, адже MongoDB ефективніше обробляє окремі строки, а не масиви строкових даних. Для даного дослідження нам достатньо розглянути один з цих методів, а саме метод матеріалізованих маршрутів.

Наведемо основи обраних методів. Наприклад, необхідно зберегти дані про ієрархію, що представлена на рисунку 1.

Найбільш простим для реалізації є метод збереження посилання на батьківський елемент. При використанні цього методу документи, що моделюють ієрархію виглядатимуть, як вказано у лістингу 1.

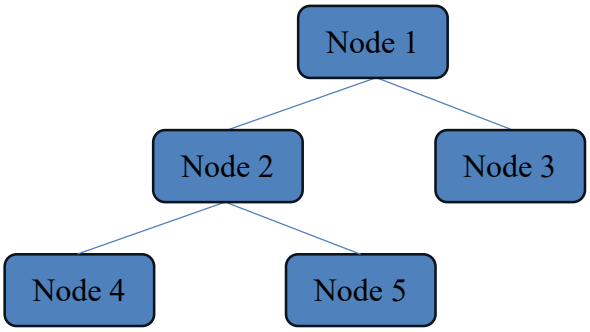

Рис. 1. Приклад ієрархії для збереження у базі даних.

```
{
 "key": 1,
 "name": "Node 1"
},
{
 "key": 2,
 "parent": 1,
 "name": "Node 2"
},
{
 "key": 3,
 "parent": 1,
 "name": "Node 3"
},
{
 "key": 4,
 "parent": 2,
 "name": "Node 4"
},
{
 "key": 5,
 "parent": 2,
 "name": "Node 5"
}
```

Лістинг 1. Структура документів, що моделює ієрархію методом збереження посилання на батьківський вузол

Метод збереження матеріалізованого маршруту полягає у збереженні спеціальної строки, що містить ідентифікатор вузла, а також ідентифікатори усіх батьківських вузлів, розділених за допомогою спеціального символу. При використанні цього методу документи, що моделюють ієрархію виглядатимуть, як вказано у лістингу 2.

Метод вкладених множин був вперше запропонований Джо Селко в [7]. Він заснований на обході ієрархії у прямому порядку. Вважається, що кожен вузол відвідується двічі (на спуску та на поверненні). Починається обхід при значенні лічильника 1. При кожному відвідуванні вузла лічильник збільшується на 1. При першому відвідуванні вузла значення лічильника зберігається у полі"left", а при другому – у полі "right". Можливість роботи ієрархічної інформації забезпечується тим, що для всіх дочірніх вузлів деякого елемента ієрархії виконуються такі умови:
$"left"_{дочірнього} > "left"_{батьківського}$ та $"right"_{дочірнього} < "right"_{батьківського}$. При використанні цього методу документи, що моделюють ієрархію виглядатимуть, як вказано на лістингу 3.

```
{
  "key": 1,
"path": "1.",
  "name": "Node 1"
},
{
  "key": 2,
  "path": "1.2.",
  "name": "Node 2"
},
{
  "key": 3,
  "path": "1.3.",
  "name": "Node 3"
},
{
  "key": 4,
  "path": "1.2.4.",
  "name": "Node 4"
},
{
  "key": 5,
  "path": "1.2.5.",
  "name": "Node 5"
}
```

Лістинг 2. Структура документів, що моделює ієрархію методом збереження матеріалізованого маршруту

Метою роботи є порівняння ефективності різних методів представлення ієрархічної інформації в докумнто-орієнтованій СКБД MongoDB за результатами тестування швидкодії зазначених методів та визначення доцільності використання цих методів у типових сценаріях.

```
{
 "key": 1,
 "left": 1,
 "right": 10
 "name": "Node 1"
},
{
 "key": 2,
 "left": 2,
 "right": 7
 "name": "Node 2"
},
{
 "key": 3,
 "left": 3,
 "right": 4
 "name": "Node 3"
},
{
 "key": 4,
 "left": 5,
 "right": 6
 "name": "Node 4"
},
{
 "key": 5,
 "left": 7,
 "right": 8
 "name": "Node 5"
}
```

Лістинг 3. Структура документів, що моделює ієрархію методом вкладених множин

Виклад основного матеріалу

Спочатку оцінимо складність реалізації типових операцій роботи з ієрархіями при використанні зазначених методів. Найпростішим для реалізації є метод збереження матриці суміжності. Операції додавання елементів, видалення окремих елементів або цілого піддерева та модифікації ієрархії (зміна батьківського вузла для піддерева) реалізуються тривіально. Єдина операція, яка могла б спричинити труднощі в реалізації, це операція вибірки всієї ієрархії або її частини. При роботі з реляційними системами керування базами даних для таких операцій потрібна підтримка рекурсивних збережених процедур і реалізація не є тривіальної. При використанні MongoDB для реалізації цієї операції можна використати aggregatepipeline [8]. Реалізація цього запиту для структури документів з лістингу 2 представлена у лістингу 4.

Бачимо, що MongoDB має спеціальну операцію «graphLookup», що безпосередньо може використовуватися при роботі з ієрархічними даними. Зазначимо, що документ, що повертається цим запитом має дещо незвичну структуру. Документ верхнього рівня описує вузол, що представляє корінь дерева, до якого додане поле «childNodes», що містить всі інші вузли дерева. Ця структура потребує спеціальної обробки у клієнтському коді, але ускладнення реалізації є незначними. Також, всі дочірні вузли відсортовані по рівнях ієрархії, що робить таку структуру зручною для конструювання на її основі структури даних «дерево» у клієнтському коді.

```
db.Nodes.aggregate([
  {
    $match: {
"ɯdentifier": 1
    }
  },
  {
"$graphLookup": {
"from": "Nodes",
"startWith": "$identifier",
"connectFromField": "identifier",
"connectToField": "parentIdentifier",
"as": "childNodes",
"depthField": "string"
    }
  }
])
```

Лістинг 4. Запит для отримання ієрархії при використанні методу збереження матриці суміжності

Метод збереження матриці суміжності є найпростішим в реалізації.

При використанні методу матеріалізованого маршруту операції додавання вузла у ієрархії та операції видалення вузла чи піддерева реалізуються дуже просто. А ось операція зміни батьківського вузла вже реалізується дещо складніше. Хоча безпосередньо реалізація не є складної, бо потрібно лише оновити значення маршруту для кожного вузла піддерева, що змінює позицію у ієрархії. Але, якщо можливе одночасне виконання подібних операцій, то необхідну синхронізацію потрібно буде реалізовувати у клієнтському коді, бо СКБД не підтримує транзакціонність, а атомарність операцій підтримується лише на рівні окремих документів. Це значно ускладнює реалізацію. Але реалізація вибірки ієрархії чи її частини є простішою за метод збереження матриці суміжності, хоча вона і потребує використання регулярних виразів. Запит для отримання ієрархії представлений у лістингу 5.

```
db.Nodes.find({path: {$regex: /^1006\./}})
.sort({path: 1})
```

Лістинг 5. Запит для отримання ієрархії при використанні методу збереження матеріалізованого маршруту

Сортування необхідно для того, щоб отримати дані у вигляді, зручному для відновлення ієрархії на стороні клієнта. У такому разі після інформації про елемент ієрархії відразу ж слідують дані про дочірні вузли. Насправді цей порядок робить конструювання дерева на стороні клієнта ще простішим ніж порядок з методу збереження матриці суміжності, але і той порядок є достатньо ефективним.

Метод вкладених множин є найскладнішим в реалізації. Тут при реалізації будь-якої операції може знадобитися оновлення значної частини вузлів ієрархії, навіть тих, що не мають відношення до тієї частини ієрархії, що оновлюється. Ці оновлення реалізувати доволі складно і вони мають бути реалізовані у клієнтському коді бо mongoDB не підтримує тріггерів. Виключенням є реалізація MongoDBAtlas, але вона доступна лише на єдиному хостінгу. Одже при реалізації будь яких операцій оновлення ієрархії розробники будуть стикатися з викликами синхронізації, такими самими що і при реалізації зміни батьківського вузла для методу матеріалізованого маршруту. Реалізація запиту для отримання ієрархії при використанні цього методу наведена у лістингу 6.

```
db.Nodes.find({$and:[{left: {$gte: 0}},
{left: {{$lte: 124}}, {right: {$lte: 124}}]})
.sort({left: 1})
```

Лістинг 6. Запит для отримання ієрархії при використанні методу вкладених множин

У цьому прикладі значення полів left та right є 0 та 124 відповідно. Зазначимо, що перевірка на те, що значення left має також бути меншим за 124 не є необхідним, бо ця нерівність і так завжди виконується. Однак додавання цієї перевірки необхідно для ефективного задіяння індексів, бо без неї виконується сканування всіх документів для яких значення left більше за 0. Для цього прикладу це взагалі всі документи. При використанні двох перевірок для поля leftMongoDB ефективно визначає документи, що представляють вузли дерева за допомогою використання індексів.

Перейдемо до порівняння ефективності методів у завданнях вибірки даних. Було проведено експеримент із тестування обраних методів. Далі наводиться опис цього експерименту та його результати.

Для аналізу ефективності була створена тестова база даних, у якій ієрархічна інформація представлена відразу трьома методами. БД була заповнена тестовою інформацією, що включає 10000 ієрархій. Розміри ієрархій варіюються від 40 до 5500 елементів, середній розмір ієрархій – 300 елементів, максимальна глибина – 6, максимальна кількість дочірніх елементів – 9 (середня кількість 5). Всі вузли ієрархії зберігалися у колекції «Nodes» що була проіндексована по всім полям, що використовуються будь-яким з трьох методів.

Для тестування швидкодії методів був розроблений додаток, який виконував вибірку 10 ієрархій одного розміру 50 разів для кожного методу з заміром часу виконання всіх операцій. Після цього обчислювався середній час виконання однієї операції вибірки для кожного методу. Умови для тестування в усіх експериментах були однакові (ОС, поточне навантаження на систему тощо). Ми не зазначаємо конфігурації комп'ютера, що використовувався для тестування, бо у результаті експерименту нас цікавить виключно порівняння ефективності методів, а не абсолютні значення часу виконання. Результати цього експерименту слід застосовувати при використанні традиційного хостингу, наприклад, при встановленні СКБД MongoDB на виділений чи на віртуальний сервер. Результати для ієрархії двох різних розмірів наведені у таблиці 1.

Таблиця 1

Швидкодія вибірки даних для різних методів представлення ієрархічної інформації (MongoDB)

Метод	Вибірка ієрархії розміром 300 елементів (глибина 4 рівня), мсек	Вибірка ієрархії розміром 5200 елементів (глибина 6 рівнів), мсек
Матриця суміжності	4,773	61
Матеріалізований маршрут	2,16	27,2
Вкладені множини	2,36	32,3

При розробці сучасних інформаційних систем часто використовуються хмарні сервіси. База даних MongoDB також може бути розміщена «у хмарі». Для запитів, що виконуються при хостінгу у хмарних сервісів буває важливіше знати вартість запитів у одиницях запиту (requestunits - RU). Використання одиниць запиту (RU) замість простого часу виконання (мілісекунди) є важливим, оскільки RU забезпечують узгоджену, незалежну від робочого навантаження міру продуктивності бази даних, тоді як час виконання може сильно відрізнятися. RU є кращим показником для розуміння того наскільки алгоритм є економічно ефективним. Наприклад простий індексований пошук може тривати 5 мс і використовувати 2 RU, а повне сканування таблиці також може тривати 5 мс, але споживати 200 RU. Лише RU показують фактичну вартість запиту [10].

Для розуміння відносної ефективності методів було проведене додаткове тестування з використанням хмарного хостінгу Microsoft Azure. У цьому разі не знадобилося багаторазове виконання запитів, бо сервіс дозволяє отримати вартість запиту у RU. Результати додаткового експерименту наведені у таблиці 2.

Таблиця 2

Швидкодія вибірки даних для різних методів представлення ієрархічної інформації (AzureMongoDB)

Метод	Вибірка ієрархії розміром 300 елементів (глибина 4 рівня), RU	Вибірка ієрархії розміром 5200 елементів (глибина 6 рівнів), RU
Матриця суміжності	54,73	703
Матеріалізований маршрут	15,4	190
Вкладені множини	18,4	250

Висновки. У статті були проаналізовані можливі методи представлення ієрархічної інформації в документо-орієнтованих базах даних, на прикладі СКБД MongoDB. Був проведений експеримент по порівняльному тестуванню обраних методів.

Зазначимо, що на нашу думку, використання методу збереження матриці суміжності доцільно у разі, якщо часті вибірки ієрархій не є типовими для інформаційної системи. Насправді для великої кількості випадків цей метод можна обрати, як такий, що забезпечує достатню швидкодію та є найпростішим у реалізації.

Досить несподіваним виявилося те, що швидкодія методу вкладених множин є трохи меншої за швидкодію методу збереження матеріалізованого маршруту. При таких результатах ми не бачимо доцільності використання цього методу. Єдине, що слід зазначити, це те, що він використовує менше постійної пам'яті, ніж метод збереження матеріалізованого маршруту. Але зазвичай вузли ієрархій містять доволі багато інших даних, окрім даних про положення в ієрархії, тобто загальний виграш буде незначним.

Метод зберігання матеріалізованого маршруту виявився найшвидшим. Його доцільно використовувати у випадках, коли характерні часті операції вибірки ієрархій. Зазначимо також, що його економічна ефективність при використанні хмарних сервісів виявилася навіть більшою, ніж ефективність стосовно часу виконання.

Література

1. Dionysius: A Framework for Modeling Hierarchical User Interactions in Recommender Systems / J. Wang, K. Kenthapadi, K. Rangadurai, D. Hardtke. // arXiv preprint, 2017. DOI: https://doi.org/10.48550/arXiv.1706.03849
2. Miller T. Hierarchical Operations and Supply Chain Planning. Springer London, London, 2001. DOI: http://dx.doi.org/10.1007/978-1-4471-0305-9
3. Дьомін М.К. Методи представлення ієрархічної інформації в реляційних базах даних інформаційно-управлінських архітектур. // Радіоелектроніка. Інформатика. Управління, Запоріжжя: ЗНТУ, 2007, №1(17). С. 56-62.
4. Danich V.N. Information-administrative architecture conception and principles of their modeling. / V.N. Danich, M.K. Domin// TEKA. Commision of motorization and energetics in agriculture. – Lublin University of Technology, Volodymyr Dahl East-Ukrainian National University in Lugansk, 2012. Vol. 12. No 4. P. 23-30.
5. Benymol J. Performance analysis of NoSQL and relational data bases with MongoDB and MySQL./ J. Benymol, A. Sajimon // Materials-Today-ProceedingsVolume 24, Part 3, 2020, p. 2036-2043. DOI: https://doi.org/10.1016/j.matpr.2020.03.634
6. Model Tree Structures. [Електронний ресурс]. – Режим доступу: //https://www.mongodb.com/docs/manual/applications/data-models-tree-structures/
7. Celko J. A Lookat SQL Trees / JoeCelko // DBMS online [Electronic resource] / Miller Freeman, Inc– Electronicmag. [USA]: *DBM S*and *Internet Systems, 1996*. March. DB [Електронний ресурс]. – Режим доступу: http://www.dbmsmag.com/9603d06.html, free. – Title from screen

8. Aggregation Pipeline. MongoDB Manual. [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/docs/manual/core/aggregation-pipeline/
9. What is MongoDB Atlas? [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/docs/atlas/
10. Request Unitsin Azure Cosmos DB [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/azure/cosmos-db/request-units?utm_source=chatgpt.com

References

1. Dionysius: A Framework for Modeling Hierarchical User Interactions in Recommender Systems / J. Wang, K. Kenthapadi, K. Rangadurai, D. Hardtke. // arXiv preprint, 2017. DOI: https://doi.org/10.48550/arXiv.1706.03849
2. Miller T. Hierarchical Operations and Supply Chain Planning. Springer London, London, 2001. DOI: http://dx.doi.org/10.1007/978-1-4471-0305-9
3. Domin M.R. Methods for representation of hierarchical information in relational databases of information-administrative architechtures. // Radioelectronics. Informatics. Management, Zaporizhia: ZNTU, 2007, №1(17) P. 56-62.
4. Danich V.N. Information-administrative architecture conception and principles of their modeling. / V.N. Danich, M.K. Domin// TEKA. Commision of motorization and energetics in agriculture. Lublin University of Technology, Volodymyr Dahl East-Ukrainian National University in Lugansk, 2012. Vol. 12. No 4. P. 23-30.
5. Benymol J. Performance analysis of NoSQL and relational databases with MongoDB and MySQL. / J. Benymol, A. Sajimon // Materials-Today-Proceedings Volume 24, Part 3, 2020, p. 2036-2043. DOI: https://doi.org/10.1016/j.matpr.2020.03.634
6. Model Tree Structures. [Electronic resource]. – URL: https://www.mongodb.com/docs/manual/applications/data-models-tree-structures/
7. Celko J. A Lookat SQL Trees / Joe Celko // DBMS online / Miller Freeman, Inc– Electronicmag. – [USA]: *DBM S*and *Internet Systems, 1996*. – March. [Electronic resource]. – URL: http://www.dbmsmag.com/9603d06.html,free. – Title from screen
8. Aggregation Pipeline. MongoDB Manual. [Electronic resource]. – URL: https://www.mongodb.com/docs/manual/core/aggregation-pipeline/
9. What is MongoDB Atlas? [Electronic resource]. - URL: https://www.mongodb.com/docs/atlas/
10. Request Units in Azure Cosmos DB [Electronic resource]. –URL: https://learn.microsoft.com/en-us/azure/cosmos-db/request-units?utm_source=chatgpt.com

Domin M.K., PolupanYu.V.Analysis of Methods for Representing Hierarchical Information in Document-Oriented Databases

The choice of an effective method for representing hierarchical information in databases is an important task in software system design since hierarchical data structures are widely used in various fields, including recommendation systems, supply chain management and analytical solutions. The article analyzes various methods of storing hierarchical structures in document-oriented databases using MongoDB as an example and compares their efficiency.

When working with a small amount of hierarchical data, almost any method will function at an acceptable speed. However, in cases where the system operates with hundreds or even thousands of hierarchies, an incorrect choice of representation method can significantly impact system performance, increase operational costs, and complicate data processing.

As part of the research, the authors analyzed methods for representing hierarchical structures in document-oriented databases, examined their advantages, disadvantages, and performance in real-world usage scenarios.

Document-oriented databases such as MongoDB provide flexible options for storing hierarchical data. In MongoDB, several approaches can be used: the adjacency matrix method, the materialized path method, and the nested sets method.

The article presents the results of performance testing of these methods in two scenarios: using a database installed on a dedicated server and using a cloud database service on Microsoft Azure. The choice of method depends on specific requirements for query speed, update flexibility, and implementation complexity.

The method of storingthe adjacency matrixperformed significantly better than in relational databases and can be used in many scenarios due to its simplicity of implementation. However, if frequent hierarchy queries or partial hierarchy retrievals are required in the software system, the materialized path method is recommended, as it operates much faster. Despite its higher implementation complexity and increased memory consumption, it provides better query performance. The nested sets method does not offer significant advantages for the most common use cases.

This research is useful for developers working on systems that process large volumes of hierarchical information.

Keywords: *document-oriented databases, MondoDB, hierarchical information, performance testing, Microsoft Azure, request units.*

Дьомін Максим Костянтинович – к. т. н., доц., доц. СНУ ім. В. Даля, факультет інформаційних технологій та електроніки, кафедра інформаційних технологій та програмування, maksymdmn@snu.edu.ua.
Полупан Юлія Вікторівна – к. т. н., доц., доц., КПІ ім. Ігоря Сікорського, факультет інформатики та обчислювальної техніки, кафедра інформатики та програмної інженерії, juliy_polupan@i.ua

Стаття подана 10.02.2025.