УДК 681.512

[1]**Ірина Миколаївна Доманецька**
Кандидат технічних наук, доцент, доцент кафедри інформаційних технологій
[2]**Ярослав Володимирович Матейко**
Розробник програмного забезпечення
[1]**Олена Володимирівна Федусенко**
Кандидат технічних наук, доцент, доцент кафедри інформаційних технологій
[1]**Володимир Миколайович Хроленко**
Кандидат технічних наук, доцент, доцент кафедри інформаційних технологій
[1]**Анатолій Олександрович Федусенко**
Аспірант кафедри інформаційних технологій

[1]*Київський національний університет будівництва і архітектури, Київ*
[2]*Науково-освітній центр «Інноваційні рішення», Київ*

ДОСЛІДЖЕННЯ ВПЛИВУ МОДЕЛІ ДАНИХ НА ЕФЕКТИВНІСТЬ РОБОТИ ВИСОКОНАВАНТАЖЕНИХ СИСТЕМ

Розглянуто архітектурні особливості високонавантажених інтернет-проектів та шаблони проектування їх інтерфейсів. Визначено, відповідні до інтерфейсних рішень, типові запити до баз даних. На прикладі СУБД MySQL проведено тестування та визначені залежності часу виконання запиту від об'ємів даних та схем формування результуючих наборів даних. Показана неспроможність реляційної моделі даних для забезпечення зростаючих вимог до WEB-додатків.

Ключові слова: високонавантажені системи, реляційна модель даних, масштабування систем, SQL-запит, сервер бази даних

Рассмотрены архитектурные особенности высоконагруженных интернет-проектов и шаблоны проектирования их интерфейсов. Определены, соответствующие интерфейсным решениям, типичные запросы к базам данных. На примере СУБД MySQL проведено тестирование и определены зависимости времени выполнения запроса от объемов данных и схем формирования результирующих наборов данных. Показана несостоятельность реляционной модели данных для обеспечения все возрастающих требований к WEB-приложениям.

Ключевые слова: высоконагруженные системы, реляционная модель данных, масштабирование систем, SQL-запрос, сервер базы данных

Modern HighLoad applications have changed the requirements for the database. The systems are simply not able to process such a large number of users requests simultaneously, causing malfunction of Internet resources. Actual effective technologies of customized solutions with guaranteed response time when processing large data sets are creating. HighLoad Internet projects architectural features and design patterns of interfaces was considered. The sample queries to databases were determined according to the front-end solutions. The query execution time dependtnces on the large volume of data and circuit formation resulting datasets were tested and analyzed. The MySQL database was used to testing these dependtnces. Relational data model failure for the ever-increasing requirements for WEB- application was shown. As shown by the tests increase query execution time with increasing amounts of information inherent in the nature of the relational model. Instead, the urgent task is to develop this architecture, which in the future with minimal time and resources can be scaled horizontally.

Keywords: HighLoad systems, the relational data model, scale systems SQL-query, database server

Постановка проблеми

Типовий випадок розвитку інтернет-проекту: він стає популярним і з кожним днем його аудиторія зростає все швидше і швидше. Зі зростанням навантаження часто спостерігається істотний спад продуктивності системи, що супроводжується численними скаргами щодо неприйнятної швидкості роботи. Користувачі протягом робочого дня

отримують повідомлення про помилки типу «Конфлікт блокувань при виконанні транзакції», «Час очікування блокування ресурсу минув» і т.п. Як правило, усвідомлення проблеми, пов'язаної з великими навантаженнями на сервіс, приходить еволюційним шляхом. Тобто той функціонал, який ще вчора працював добре, сьогодні «раптом» починає працювати з незадовільною відмовостійкістю, повільно, а деякі сервіси або весь інтернет-проект стає недоступним. Проблематика цієї статті полягає в аналізі глибинних причин недієздатності традиційної моделі масштабування системи збереження та опрацювання даних в умовах великих об'ємів даних, які необхідно структурувати та аналізувати, що зумовлені особливостями механізмів роботи реляційної моделі даних. Стрімко зростаючі об'єми інформації породжують нові складні завдання щодо організації її зберігання і обробки [11].

Аналіз останніх досліджень і публікацій

Реляційна модель даних стала одним з найпоширеніших стандартів представлення інформації в комп'ютерних системах забезпечивши розробників прикладних систем зручним підходом до подання та маніпулювання даними. Однак реляційна модель, опрацьовуючи тільки атомарні дані, не забезпечує обробку неструктурованої інформації, і, більше того, не надає можливості зберігання мультимедійних даних. До того ж персоналізація послуг в рамках транзакційних баз даних за умови зростання об'ємів інформації все більш ускладнює опрацювання даних засобами реляційних СУБД [1].

Оптимізація запитів до баз даних має широку бібліографію. Над цією тематикою працювали С.Д. Кузнєцов, М. Ярк (M. Jarke), Ю. Кох (J. Koch), Т.Сейджер (T. Sager), Грейф (G. Graefe), К. Дейт (C. Date), Дж. Герке (J. Gehrcke) та багато інших [1; 2; 4; 8]. На ряду з академічними дослідженнями друкуються публікації компаній, присвячені розв'язанню цієї задачі в рамках окремих СУБД (наприклад, MySQL, Oracle, PostgreSQL). Крім того, за даною тематикою проведена велика кількість конференцій, таких як HighLoad++ (2007, 2008, 2009, 2010, 2011 і 2012, 2013). З появою великої кількості високонавантажених інтернет-проектів, які оперують великими обсягами даних і мають велику кількість відвідувачів, для яких найважливішим завданням є забезпечення постійної високої пропускної здатності сховища при необмеженому збільшенні обсягів даних почав розвиватися напрям NoSQL. На сьогодні створено велику кількість NoSQL рішень, написано багато робіт та проведено спеціалізованих конференцій (NoSQL matters Conference (2013), NoSQL NOW (2013)) [4; 7; 6; 9; 10; 11].

Мета статті

Мета даної публікації полягає в аналізі найбільш поширених шаблонів проектування інтерфейсів WEB-додатків та визначення відповідних їм типових запитів до реляційної бази, що є потенційними чинниками зменшення продуктивності високонавантажених систем, а особливо, збільшення часу очікування відповіді користувачами.

Виклад основного матеріалу

Сучасні WEB-системи висувають все більш зростаючі вимоги до системних ресурсів, таких як: процесорний час, оперативна пам'ять, дисковий простір, завантаженість дискової підсистеми кількістю операцій на секунду. Для них типовою стає ситуація, коли зростання аудиторії випереджає нарощування потужностей системи і настає момент, коли вже досягнута верхня межа можливостей архітектури і додавання нового комп'ютера нічого не вирішує, або вирішує на дуже короткий час [5].

Архітектура інтернет-додатків, як правило, складається з трьох рівнів доступу.

Перший рівень: HTTP сервер, наприклад Apache HTTPD або NginX. Виконує роль маршрутизатора запитів до серверів бізнес-логіки.

Другий рівень: сервери бізнес-логіки, можуть бути виконані на будь-якій мові програмування. Виконують основні операції з базою даних, опрацьовують всі запити, формують відповіді на запити.

Третій рівень: сервери баз даних. Один з найважливіших рівнів. Виконує авторизацію запитів до джерел даних, їх обробку та видачу результатів на запити.

Дуже часто вся інфраструктура побудована на базі реляційної моделі роботи з даними. Отже всі бізнес-процеси залежать від особливостей реалізації реляційної бази даних. Традиційний шлях масштабування – збільшення системних ресурсів системи, перестає бути ефективним на підході до експлуатаційних лімітів системи, при цьому, зростання вартості має експоненціальний характер [3].

Під час проектування інтерфейсів інтернет-застосувань, дизайнери інтерфейсів зазвичай використовують певні правила, шаблони та методики проектування. Інтерфейс, який використовує типові, звичні користувачам рішення, має більше шансів отримати симпатії користувачів. В роботі [3] Я.В. Матейко виділив шаблони проектування інтерфейсів WEB-додатків типу інтернет-магазин.

До шаблонів проектування інтерфейсів можна віднести такі:

- динамічно доповнювана сторінка результатів;

- сторінка видачі результатів з розподіленням (пагінацією) даних на сегменти;
- сторінка видачі результатів з розподіленням (пагінацією) даних на сторінки та фільтрацією за критеріями (каталоги, фільтри).

Динамічно доповнювана сторінка результатів (рис. 1) зазвичай використовується для відображення результуючого набору даних, який незручно переглядати повністю з міркувань зручності користувача.

Цей набір даних доповнюється у міру того, як користувач проглядає його. Наприклад: стіна повідомлень соціальних мереж, деякі пошукові системи.

Сторінка видачі результатів з розподіленням (пагінацією) даних на сегменти (рис. 2), зазвичай використовується в пошукових та інших системах, де необхідно розділити результуючий набір даних на сегменти для більш зручної навігації та зменшення швидкості завантаження сторінки. Наприклад: пошукові системи, портали новин, каталоги товарів.

Сторінка видачі результатів з розподіленням (пагінацією) даних на сегменти та фільтрацією за критеріями (каталоги, фільтри) (рис. 3), доповнює звичайну сторінку з пагінацією фільтрами. Фільтри дозволяють виділити певну множину даних з результуючого набору цих даних, за певним критерієм.

На прикладі роботи наявного та діючого на даний момент часу інтернет-магазину з продажу автозапчастин Drozd.kiev.ua розглянемо, за якими шаблонами побудовані його функціональні блоки.

Спочатку охарактеризуємо особливості функціонування цього інтернет-магазину. Вже на сьогодні маємо справу з високонавантаженою системою, яка характеризується недостатньою швидкістю виконання операцій з пошуку, відбору, сортування товарів та великою завантаженістю з боку відвідувачів сайту та пошукових роботів, які виконують регулярну індексацію сайту. База товарів, яка наявна на сайті, складає близько 6,000 автозапчастин. Актуальна повна база товарів, складає понад 100,000 одиниць товарів. Крім того на поточний момент спостерігається тенденція до збільшення номенклатури товарів, що призведе до ще більшого зростання об'ємів інформації, якою необхідно оперувати і як наслідок, до збільшення часових витрат на обробку запитів.

Система має такі основні функціональні блоки:

- блок новин;
  - перегляд останніх 10 новин;
  - перегляд всіх новин;
- каталог товарів;
  - відбір товарів за групою;
- відбір товарів за виробником
- можливість сортування товарів за ціною;пошук;
- швидкий доступ до каталогу;
- швидкий доступ до конкретної одиниці товару;
- рекламні посилання товарів;
- здійснення замовлень в магазині через форму замовлень.

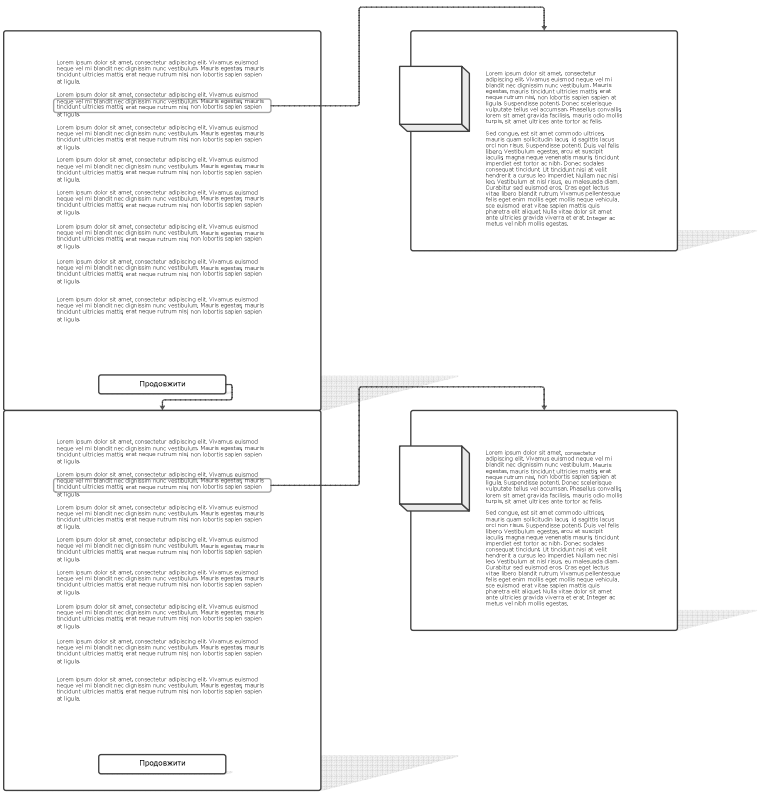

Рис. 1. Динамічно доповнювана сторінка результатів

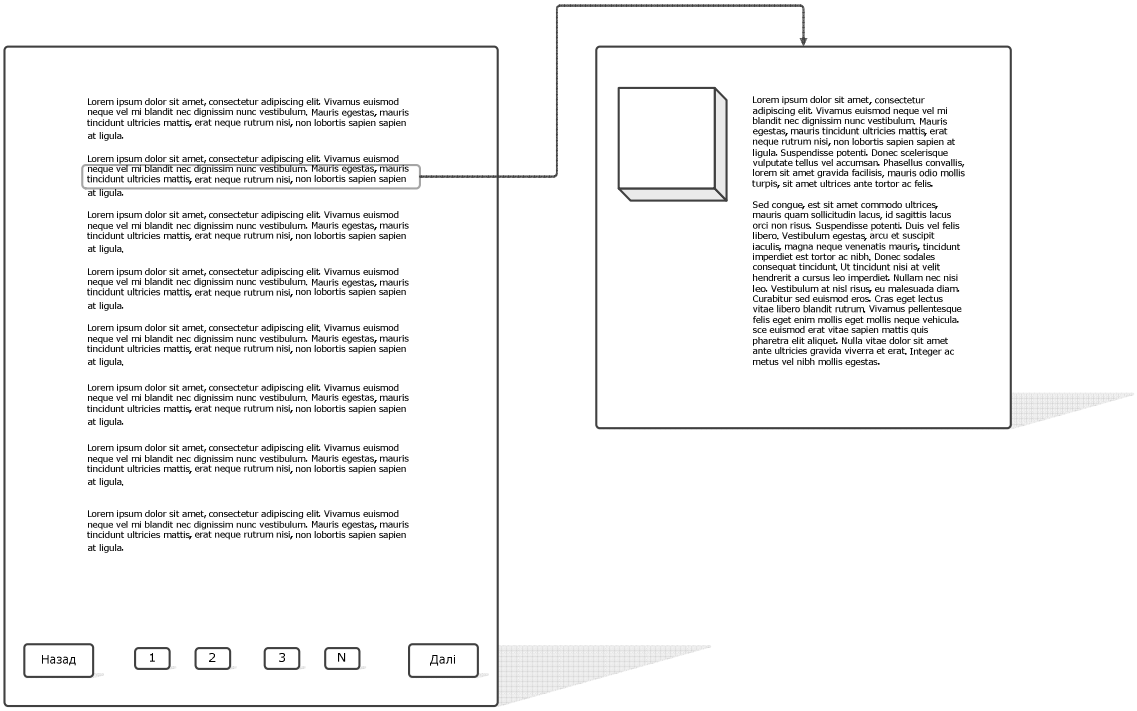

Рис. 2. Сторінка видачі результатів з розподіленням (пагінацією) даних на сегменти

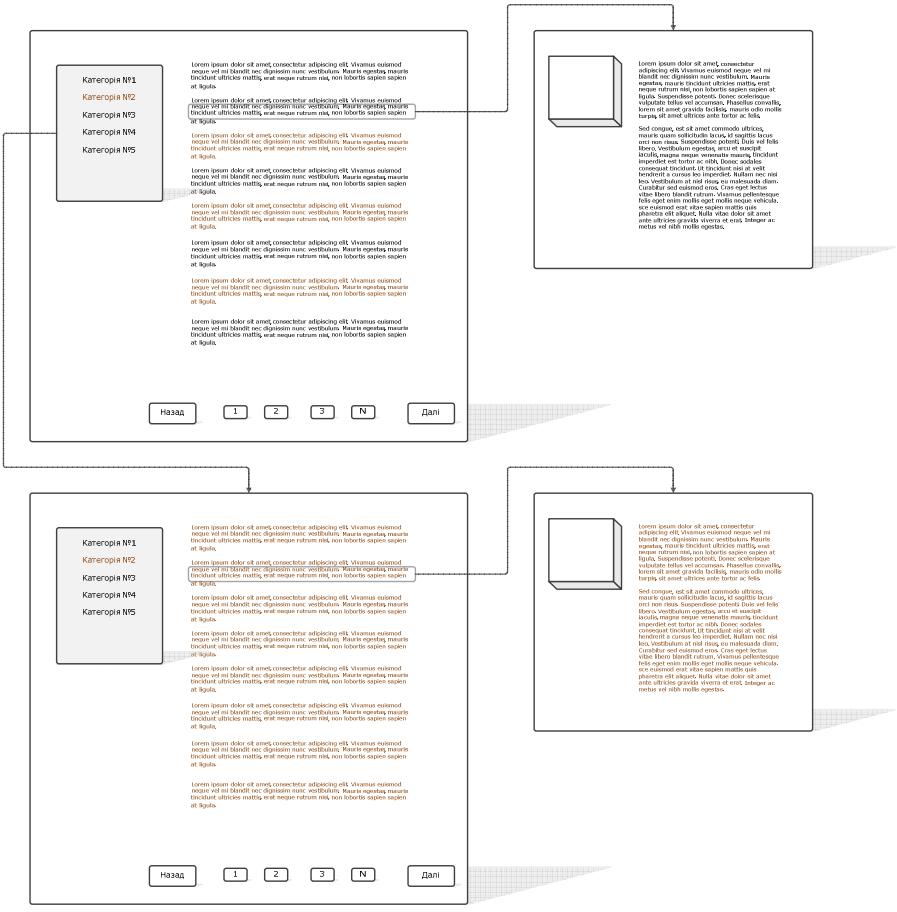

Рис. 3. Сторінка видачі результатів з розподіленням (пагінацією) даних на сегменти та фільтрацією за критеріями (каталоги, фільтри)

Основні функціональні блоки інтернет-магазину побудовані з врахуванням основних шаблонів проектування інтерфейсів.

Так, блок новин та пошук можуть бути реалізовані як динамічно доповнювана сторінка. Доступ до каталогу може бути реалізований в свою чергу, з врахуванням особливостей сторінки видачі результатів з розподіленням (пагінацією) даних на сегменти та фільтрацією за критеріями.

Блоки рекламних посилань та форми замовлень не розглядаються в даній роботі, оскільки вони не зачіпають основну проблематику даної роботи.

Розглянемо IDEF0 діаграми для основних функціональних блоків інтернет-магазину.

IDEF0 діаграма функції «пошук по сайту» має такі параметри:

- вхід: таблиці та індекси;
- керуючий елемент: запит (після декомпозиції розділяється на умову виходу та ключові слова);
- виконавець: SQL сервер;
- вихід: результат роботи функціонального блоку у вигляді результуючого набору даних.

IDEF0 діаграма функції «пошук по сайту» після декомпозиції розподіляється на два функціональних блоки: агрегатор та сканер таблиць (рис. 4).

Розглянемо їх більш детально. Агрегатор це підсистема серверу баз даних, яка виконує ініціалізацію процесу, передачу вхідних даних дочірнім підсистемам, збирання результуючої інформації, передавання вихідних даних та фіналізацію процесу. Сканер таблиць виконує повнотекстове сканування вхідної таблиці з фільтрацією з ключовими словами та повертає результат агрегатору.

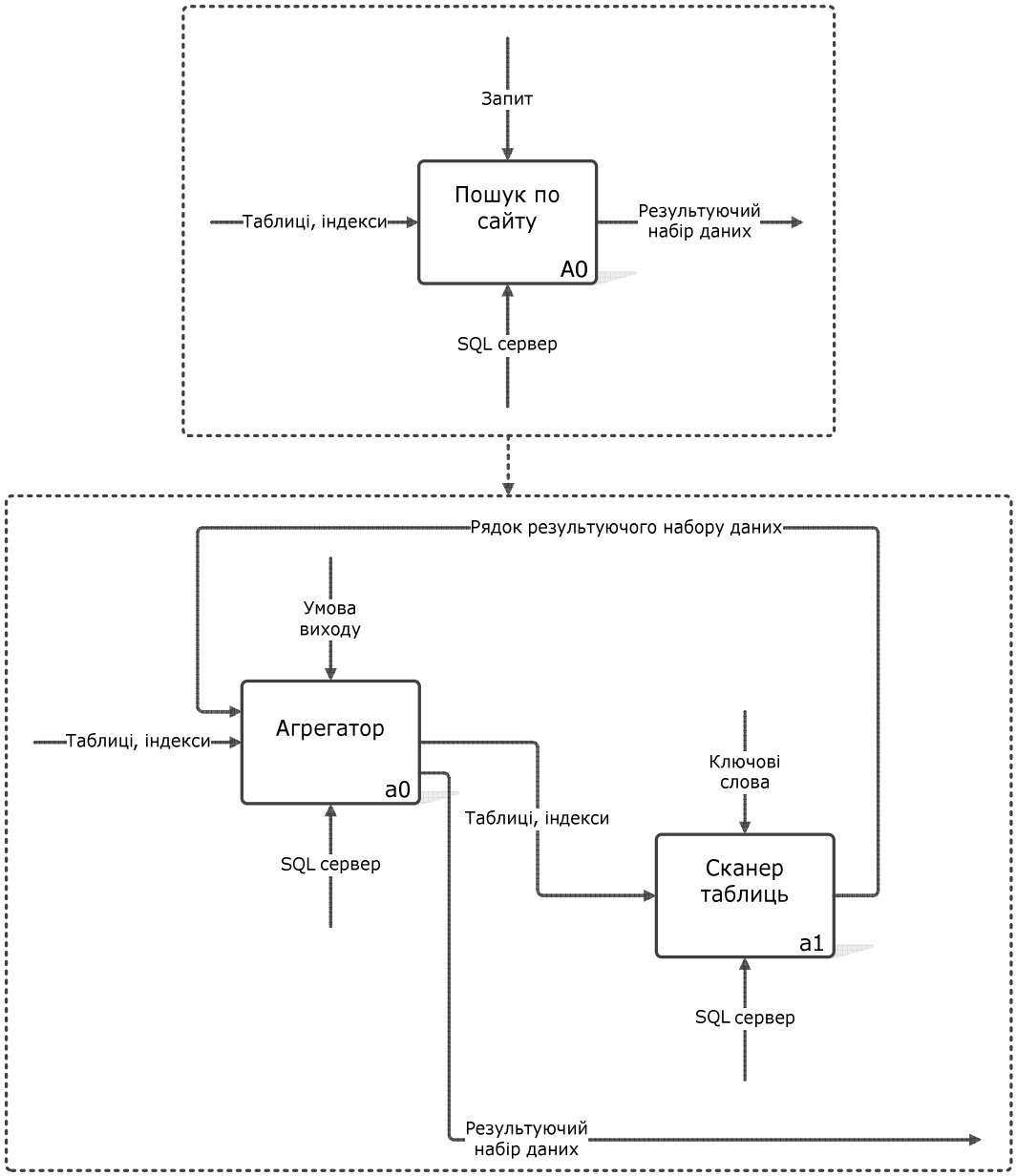

Рис.4. IDEF0 діаграма функції «пошук по сайту»

Оскільки функції «перегляд новин» та «перегляд каталогу» мають однаковий алгоритм роботи, їх IDEF0 діаграми однакові та мають такі параметри:

- вхід: таблиці та індекси;
- керуючий елемент: запит (після декомпозиції розділяється на умову виходу та критерії фільтрації);
- виконавець: SQL сервер;
- вихід: результат роботи функціонального блоку у вигляді результуючого набору даних.

IDEF0 діаграми функцій «перегляд новин» та «перегляд каталогу» після декомпозиції розподіляється на три функціональних блоки: агрегатор, сканер індексу та сканер таблиць. Сканер індексу виконує сканування вхідних індексів з фільтрацією за критеріями фільтрації та повертає результат (адреса рядку в таблиці) сканеру таблиць у вигляді керуючого елементу.

Щоб не дублювати діаграму нижнього рівня двічі зобразимо всі три діаграми на єдиному (рис. 5).

IDEF0 діаграма функції «перегляд конкретної одиниці каталогу» має такі параметри:

- вхід: таблиці та індекси;
- керуючий елемент: запит (після декомпозиції розділяється на ідентифікатор товару та критерії фільтрації);
- виконавець: SQL сервер;
- вихід: результат роботи функціонального блоку у вигляді результуючого набору даних.

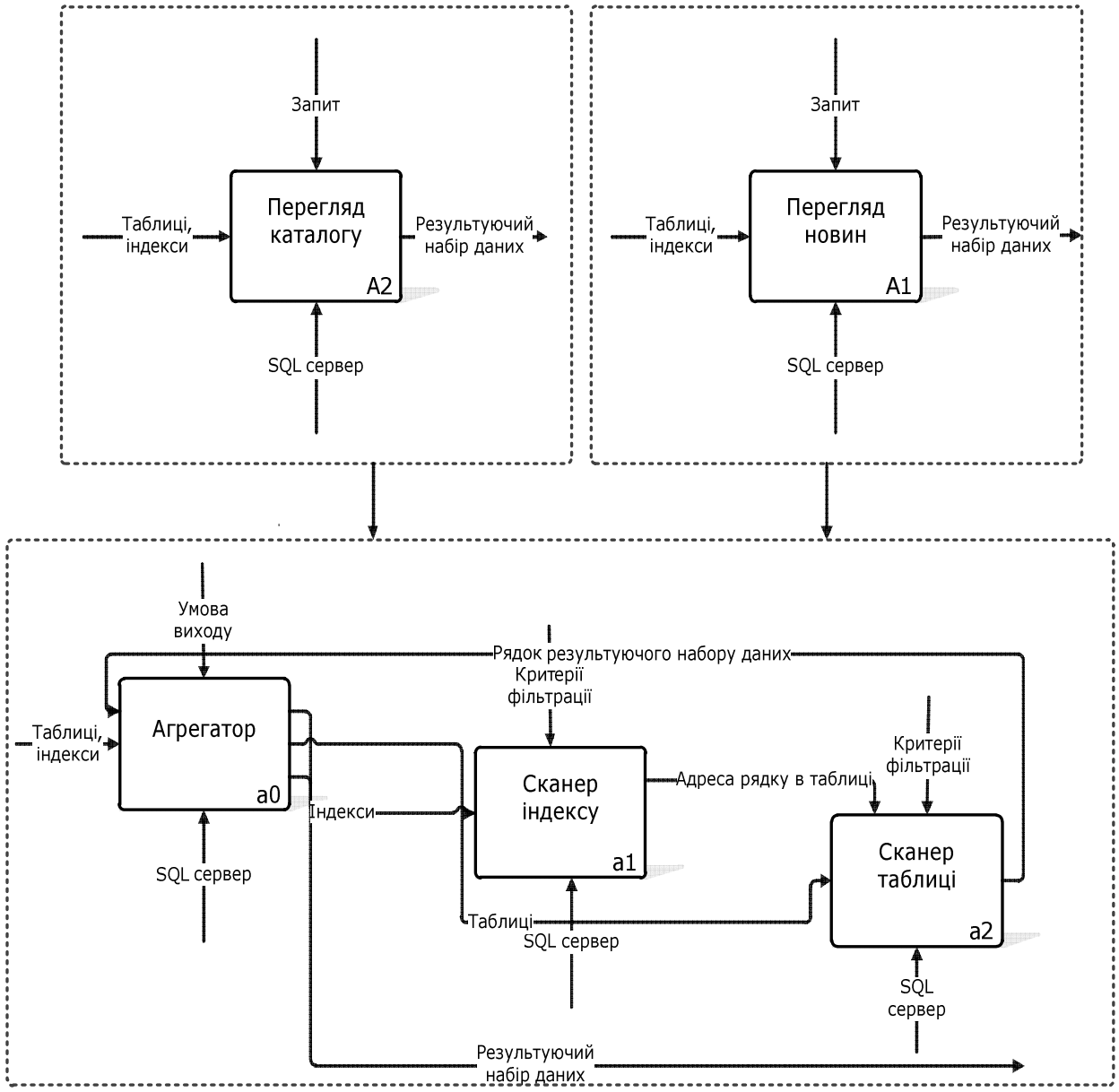

Рис. 5. IDEF0 діаграма функції «перегляд конкретної одиниці каталогу»

IDEF0 діаграма функції «перегляд конкретної одиниці каталогу» після декомпозиції розподіляється на два функціональних блоки: сканер індексу та сканер таблиць (рис. 6).

Як показано на діаграмах виконавцем усіх дій над даними є сервер бази даних.

Розглянемо архітектурні особливості системи управління базами даних на прикладі СУБД MySQL. Вибір цієї системи як еталонної, пояснюється такими фактами:

- найбільш поширена в промисловості серед постачальників хостингових послуг;
- лояльна схема ліцензування та використання;
- має найбільшу та найактивнішу спільноту в світі, отримати допомогу набагато легше аніж з іншими, менш розповсюдженими системами.

В більшості задач, які стоять перед архітектором програмного забезпечення, переважають певні шаблони проектування. Це зумовлено шаблонністю вимог, задач та їх рішень [3].

До таких задач можна віднести такі:

- вибірку перших N-елементів;
- вибірку останніх N-елементів;
- пошук в масиві даних;
- фільтрування даних;
- пошук елементу за його ідентифікатором.

Для визначення часових витрат при типових сценаріях роботи інтернет-застосувань, було проведено тестування щодо витрат часу на виконання типових вибірок на великих обсягах даних.

Конфігурація стенду:

- Processor: Intel(R) Core(TM)2 Quad CPU Q8300 @ 2.50GHz (4 CPUs), ~3.0GHz;
- Memory: 8192MB RAM;
- Fixed hard disk: SAMSUNG SP2004C 7200 rpm 200 gb.

Програмне забезпечення:

- Microsoft Windows 7 Enterprise Version 6.1.7601 Service Pack 1;
- mysql-5.5.15-win32.

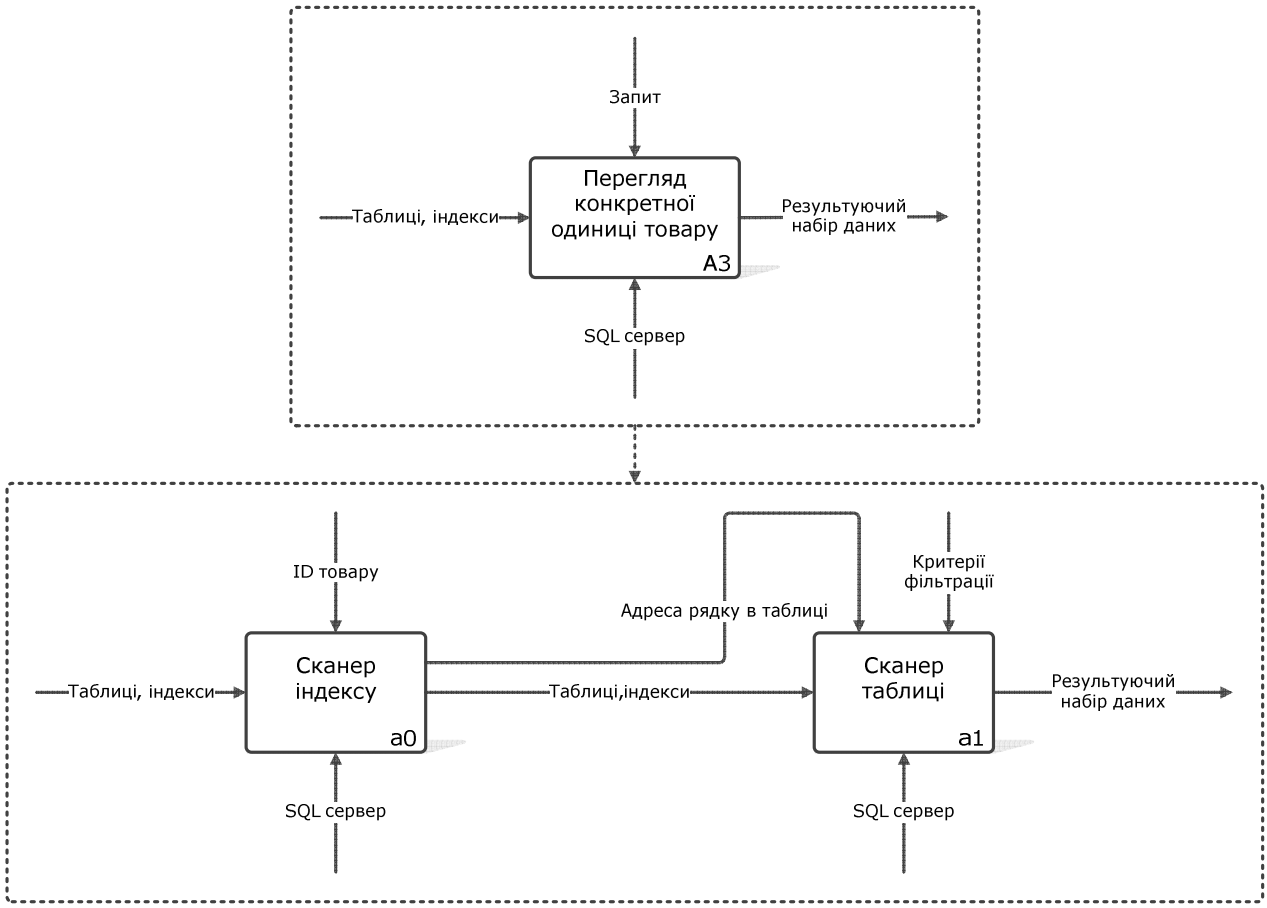

Рис. 6. IDEF0 діаграма функції «перегляд конкретної одиниці каталогу»

Для тестування використовується набір даних у формі наведеній в таблиці.

Таблиця

Фізична модель даних таблиці для тестування

c1 (int)	c2 (CHAR(20))
1000	"obj.0001000"
1001	"obj.0001001"
…	…
99999999	"obj.99999999"

Загальний об'єм тестових даних складає повних 1,000,000 та 10,000,000 рядків зазначеного вище формату.

Необхідно провести тести за найбільш розповсюдженими сценаріями використання ПЗ.

На першому етапі тестування проводилось на наборі даних без застосування індексів.

На другому етапі тестування проводилось з наступними налаштуваннями індексу:

- create index i1 using btree on d1{0}m (c1 asc,c2 asc);
- create index i2 using btree on d1{0}m (c1 desc,c2 desc).

Кожен етап містить два різновиди тестів:

- вибірка перших n-елементів;
- вибірка останніх n-елементів.

Виконувались такі запити:

- select * from d1{0}m limit 10 offset n;
- select * from d1{0}m order by c1 asc limit 10 offset n;
- select * from d1{0}m order by c2 asc limit 10 offset n.

Аналізуючи результати тесту «вибірка перших n-елементів», можна зробити певні висновки, а саме:

- швидкість вибірок на масиві даних деградує прямо пропорційно об'єму цих даних (загальна тенденція);
- без індексів, швидкість вибірок лежить в одному діапазоні при використанні order by.

Деградація в швидкості напряму впливає на такі функціональні блоки, де необхідно отримувати послідовний доступ до даних. Це можуть бути новини, товари, замовлення і т.д.

Для тесту «вибірка останніх n-елементів» використовувались такі запити:

- select * from d1{0}m order by c1 desc limit 10 offset n;
- select * from d1{0}m order by c2 desc limit 10 offset n.

Результати цього тесту корелюють з результатами попереднього тесту: швидкість вибірок без використання індексів лежить в одному діапазоні; при використанні order by система страждає від надмірного навантаження, прямо пропорційно об'єму. Вплив на функціональні блоки аналогічний впливу при вибірках перших n –елементів [3].

Таким чином, дослід демонструє, що для деяких випадків реляційна модель не забезпечує прийнятних значень часових параметрів.

Висновки

Однією з особливостей сучасних систем є зростаючі вимоги до системних ресурсів, таких як: процесорний час, оперативна пам'ять, дисковий простір, завантаженість дискової підсистеми кількістю операцій на секунду.

Традиційний шлях масштабування – збільшення системних ресурсів системи, перестає бути ефективним на підході до експлуатаційних лімітів системи, при цьому, зростання вартості має експоненціальний характер.

У даній роботі було проаналізовано особливості впливу реляційної моделі даних на значення часових характеристик реагування системи на запити користувачів. Як показали проведені тести, збільшення часу виконання запитів зі зростанням об'ємів інформації закладено в самій природі реляційної моделі. Натомість, актуальною задачею є розробка такої архітектури, яка в майбутньому з мінімальними витратами часу та ресурсів, зможе масштабуватись горизонтально, при цьому каталізатором до таких змін буде виступати потреба в збільшені паралельно виконаних операцій на секунду.

Список літератури

1. *Волков Д. Открытые системы СУБД – М. – 2012. – № 02. С. 1028-7493.*
2. *Клеменков П.А. Большие данные: современные подходы к хранению и обработке / П.А. Клеменков, С.Д. Кузнецов / Труды Института системного программирования, т. 23. – М.: ИСП РАН, 2012. – С. 143-158.*
3. *Матейко Я.В. Розробка ефективних механізмів індексації для високонавантажених систем в умовах обмежених ресурсів на прикладі інтернет-магазину автозапчастин / Дипломна робота на здобуття кваліфікації спеціаліст з напряму 7.05010101 Інформаційні управляючі системи та технології. – К.: КНУБА, 2012. – 123 с.*
4. *Мендкович Н. А. Обзор развития методов лексической оптимизации запросов / Н. А. Мендкович, С. Д. Кузнецов / Труды Института системного программирования т. 23. – М.: ИСП РАН, 2012. – С. 195-214.*
5. *Martin L. Abbott, Michael T. Fisher. The art of scalability. – Addison-Wesley, 2009.*
6. *Mark A. Beyer, Douglas Laney. The Importance of 'Big Data': A Definition, Mark Beyer, Douglas Laney, G00235055*
7. *Rick Cattell Scalable SQL and NoSQL Data Stores / Cattell Rick / SIGMOD Record, December 2010 (Vol. 39, No. 4).*

8. Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber. Bigtable: a distributed storage system for structured data. Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, vol. 7, p. 15-15, USENIX Association Berkeley, CA, USA, 2006

9. P. Hunt, M. Konar, F. P. Junqueira, and B. Reed. ZooKeeper: wait-free coordination for internet-scale systems. USENIXATC'10: Proceedings of the 2010 USENIX conference on USENIX annual technical conference. Berkeley, CA, USA: USENIX Association, 2010, pp. 11–11.

10. Konstantin Shvachko, Hairong Kuang, Sanjai Radia, Robert Chansler. The Hadoop Distributed File System. MSST '10 Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010, pp. 1-10.

11. Tom White Hadoop: The Definitive Guide, 3rd Edition / White Tom /O'Reilly Media, 2012, 688 p.

References

1. Volkov D. Open Systems DBMS / M. 2012, № 02, pp.1028-7493.

2. Klemenkov, P.A. Big Data: current approaches to storage and processing / P.A. Klemenkov, S.D. Kuznetsov / Proceedings of the Institute for System Programming, Vol 23, Moscow, ISP RAS, 2012, pp. 143-158.

3. Mateiko Y.V. Development of effective indexing for high loaded systems under limited resources on the example of an online store of auto parts / Thesis for obtaining specialist qualification in the specialty 7.05010101 Information Control Systems and Technologies / Kyiv, KNUBA, 2012, 123c.

4. Mendkovich, N.A. Review of lexical query optimization techniques / N.A. Mendkovich, S.D. Kuznetsov / Proceedings of the Institute for System Programming, Vol 23, Moscow, ISP RAS, 2012, pp. 195-214.

5. Martin L. Abbott, Michael T. Fisher. The art of scalability. – Addison-Wesley, 2009Tom White Hadoop: The Definitive Guide, 3rd Edition / White Tom /O'Reilly Media, 2012, 688 p.

6. Mark A. Beyer, Douglas Laney. The Importance of 'Big Data': A Definition, Mark Beyer, Douglas Laney, G00235055

7. Rick Cattell Scalable SQL and NoSQL Data Stores / Cattell Rick / SIGMOD Record, December 2010 (Vol. 39, No. 4)

8. Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber. Bigtable: a distributed storage system for structured data. Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, vol. 7, USENIX Association Berkeley, CA, USA, 2006, p. 15.

9. P. Hunt, M. Konar, F. P. Junqueira, and B. Reed. ZooKeeper: wait-free coordination for internet-scale systems. USENIXATC'10: Proceedings of the 2010 USENIX conference on USENIX annual technical conference. Berkeley, CA, USA: USENIX Association, 2010, pp. 11–11.

10. Konstantin Shvachko, Hairong Kuang, Sanjai Radia, Robert Chansler. The Hadoop Distributed File System. MSST '10 Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010, pp. 1-10.

11. Tom White Hadoop: The Definitive Guide, 3rd Edition / White Tom /O'Reilly Media, 2012, 688 p

Стаття надійшла до редколегії 12.02.2014

Рецензент: д-р техн. наук, проф. С.В. Цюцюра, Київський національний університет будівництва і архітектури, Київ.