УДК 004.6

АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ SQL И NOSQL ПРИ РАЗЛИЧНЫХ ПРОФИЛЯХ НАГРУЗКИ (OLTP И OLAP)

Рыженкова Е.И., студент гр.382571, Терешёнок М.Н., студент гр.382571

Белорусский государственный университет информатики и радиоэлектроники,
Институт информационных технологий,
г. Минск, Республика Беларусь

Бакунов А.М. – старший преподаватель кафедры ИСиТ

Аннотация. В статье был проведен сравнительный анализ эффективности MySQL (реляционная СУБД) и MongoDB (документно-ориентированная СУБД). Оценка производительности осуществлялась при моделировании различных сценариев нагрузки: OLTP (операции с транзакциями) и OLAP (аналитические запросы). Для эксперимента использовались дата сеты объемом от 100 записей до 10 миллионов, на которых выполнялись операции: INSERT, SELECT, UPDATE, OLAP и JOIN. Основная задача исследования заключалась в выявлении особенностей поведения каждой системы при масштабировании данных и интенсивности запросов.

Ключевые слова. SQL, NoSQL, MySQL, MongoDB, OLTP, OLAP, производительность, базы данных, нагрузка.

Введение.

Информационные системы сталкиваются с необходимостью высокоэффективной обработки значительных объёмов данных в условиях разнообразных нагрузок. В качестве фундаментальных подходов к архитектуре хранения данных выступают реляционные (SQL) и нереляционные (NoSQL) системы управления базами данных [1]. Выбор между ними зависит от характера операций и объёма данных.

Целью данной работы является сравнительный анализ производительности MySQL и MongoDB при выполнении операций, характерных для транзакционных (OLTP) и аналитических (OLAP) нагрузок [2, 3]. OLTP-нагрузка включает операции вставки, выборки, обновления, тогда как OLAP-нагрузка связана с агрегированием и анализом больших массивов данных.

Основная часть.

Анализ производился на персональном компьютере с 16 ГБ оперативной памяти и SSD-накопителем. Для получения достоверных результатов был разработан программный модуль на языке JavaScript (Node.js), позволяющий автоматизировать выполнение операций над базами данных. Перед началом базы данных очищались, что обеспечивало одинаковое начальное состояние.

Загрузка данных выполнялась пакетами. Размер пакета адаптировался в зависимости от объёма данных, что позволило снизить расходы и повысить стабильность выполнения. После вставки формировался набор существующих идентификаторов, используемый в последующих операциях что исключало возможность ошибок, связанных с обращением к несуществующим записям.

Для обеспечения корректности сравнения в MongoDB были созданы по полям поиска, а в MySQL использовались встроенные механизмы индексирования. Замеры времени выполнения операций производились с помощью системного времени до и после выполнения серии запросов. Каждая операция выполнялась многократно, что позволило получить стабильные значения времени.

Исследование включало операции:

– INSERT – массовая вставка данных;

– SELECT – случайные выборки пользователей;

– UPDATE – обновление баланса пользователей;

– OLAP-запросы – агрегирование данных по странам;

– JOIN – соединение таблиц и агрегация заказов на пользователя.

Для каждого объёма операции выполнялись отдельно для каждой СУБД, а результаты сохранялись для последующего анализа и визуализации, позволяя сравнить производительность MySQL и MongoDB при транзакционных и аналитических нагрузках.

Анализ результатов.

Операции вставки (INSERT). Результаты показывают, что при малых объёмах данных MySQL и MongoDB показывают схожее время выполнения (рисунок 1). Но с увеличением объёма данных преимущество постепенно переходит к MongoDB, так при 10 м записей MySQL потребовалось 411 168 мс, а MongoDB – 85 667 мс. Это объясняется более гибкой моделью хранения и отсутствием необходимости поддерживать сложные связи.

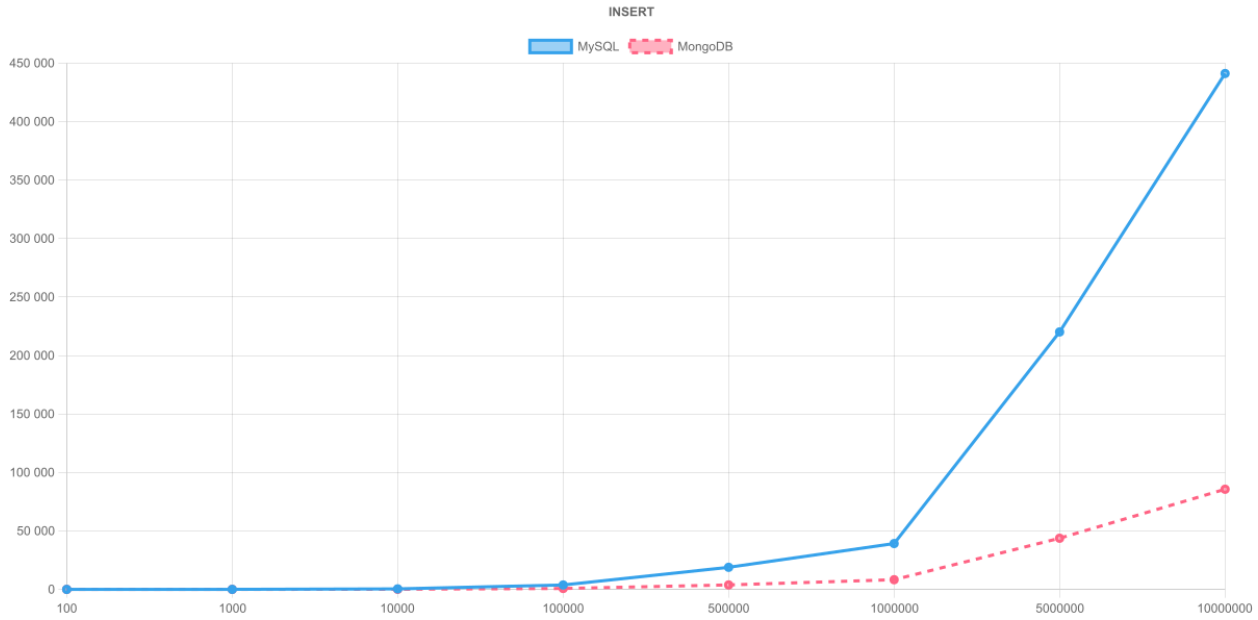

Рисунок 1 – Результаты операции INSERT

При массовой вставке данных MongoDB оказывала более высокую нагрузку на оперативную память и процессор, это связано с обработкой вложенных документов и дополнительными операциями подготовки структуры документа перед записью. MySQL использовал ресурсы более равномерно, и, как правило, не создавал высокой пиковой нагрузки на CPU, но при больших объёмах также была заметна нагрузка на диск и оперативную памяти из-за пакетной записи и журналов.

Операции SELECT. Выборки данных продемонстрировали заметную разницу в поведении СУБД. MySQL показывала почти ровную линию на графике, так как поиск по индексированному полю оставался быстрым даже при росте объёма данных (рисунок 2). В MongoDB время выборки увеличивалось с ростом объёма данных. Основные причины происходящего - это структура документов, каждый пользователь содержит вложенный массив и при поиске обрабатывается весь документ, также база возвращает весь документ, а не отдельные поля, что повышает нагрузку на процессор и память.

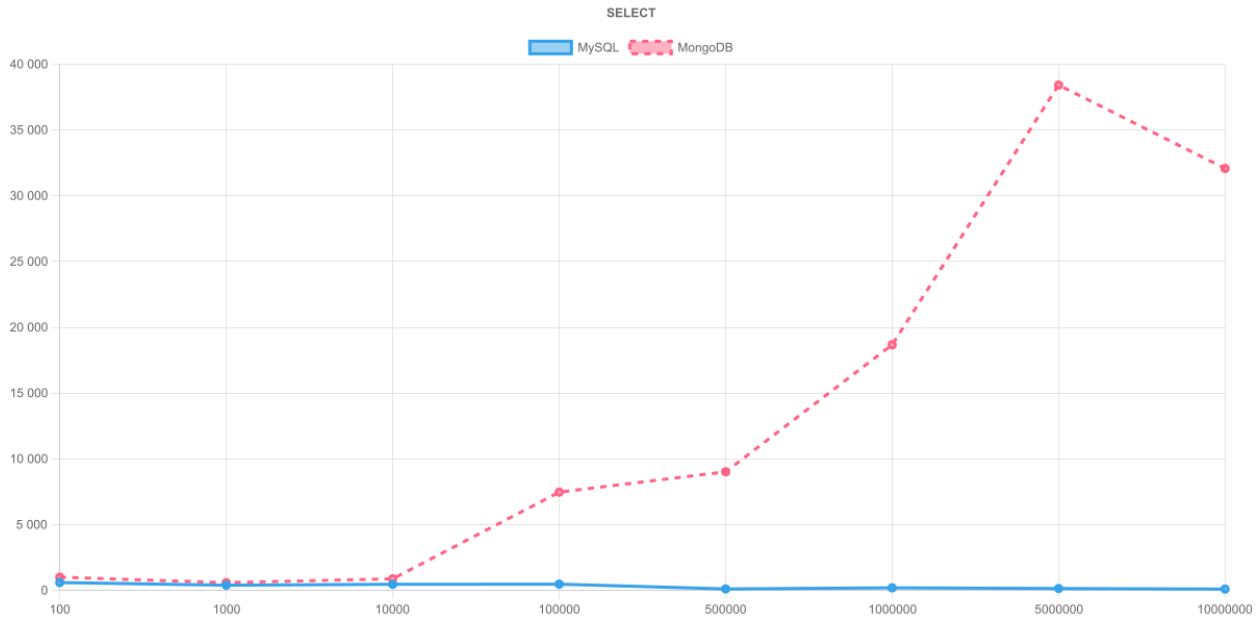

Рисунок 2 – График операции SELECT

Операции UPDATE. При операциях обновления график показывает интересную динамику: на малых и средних объёмах MongoDB была быстрее, так как обновляла отдельные поля документа без пересчёта индексов и связей (рисунок 3). Одна при росте объёма данных MySQL опережает MongoDB. Из-за увеличения числа документов и параллельных операций MongoDB начинает потреблять больше

ресурсов, что замедляет выполнение, также вложенные массивы и объём документов заставляют перерабатывать больше данных, даже при обновлении одного поля. MySQL более стабильно использует ресурсы, даже не смотря на пересчёт индексов, увеличение объёма данных сказывается меньше, и при больших наборах данных MySQL становится более эффективной для обновлений.

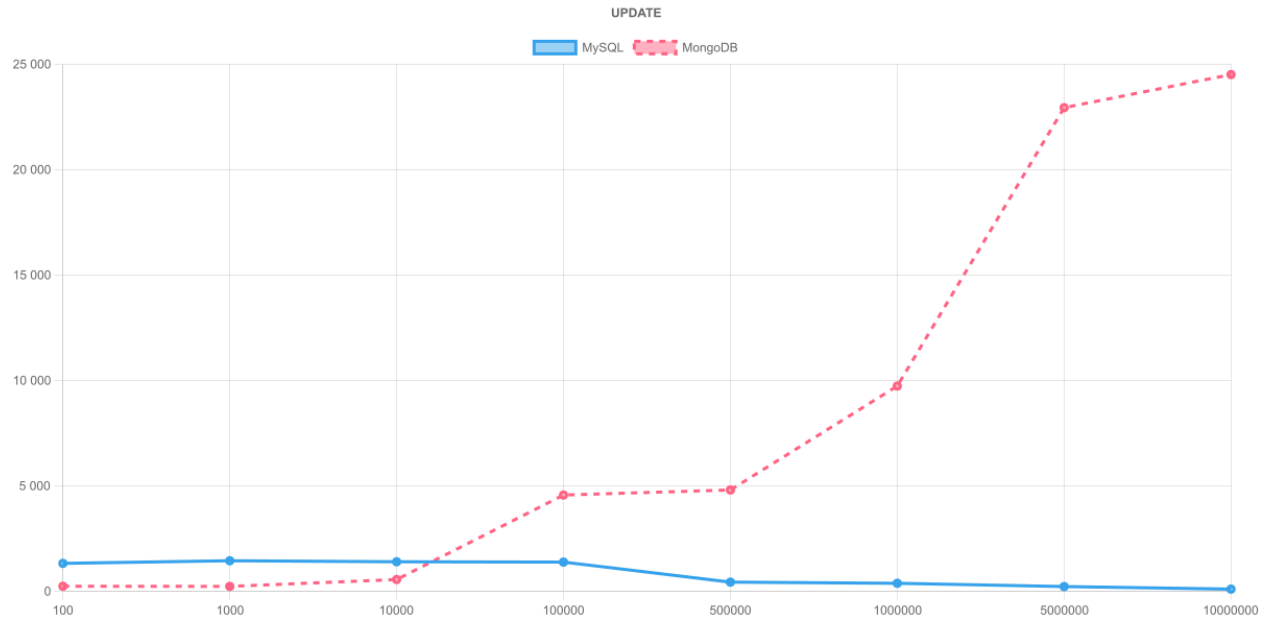

Рисунок 3 – График операции UPDATE

Операции OLAP. Для аналитических запросов наблюдается обратная тенденция: MongoDB показывает более высокую производительность при больших объёмах данных (Рисунок 4). MongoDB агрегирует данные без необходимости сканирования всех таблиц, кроме того, встроенная оптимизация работы с массивами и вложенными структурами хорошо справляется с массивами, ускоряя вычисления на средних и максимальных значений. MySQL же требует пересчёта индексов и последовательного сканирования таблиц для агрегаций, что делает её медленнее при больших наборах данных, но потребление ресурсов при этом остаётся умеренным.

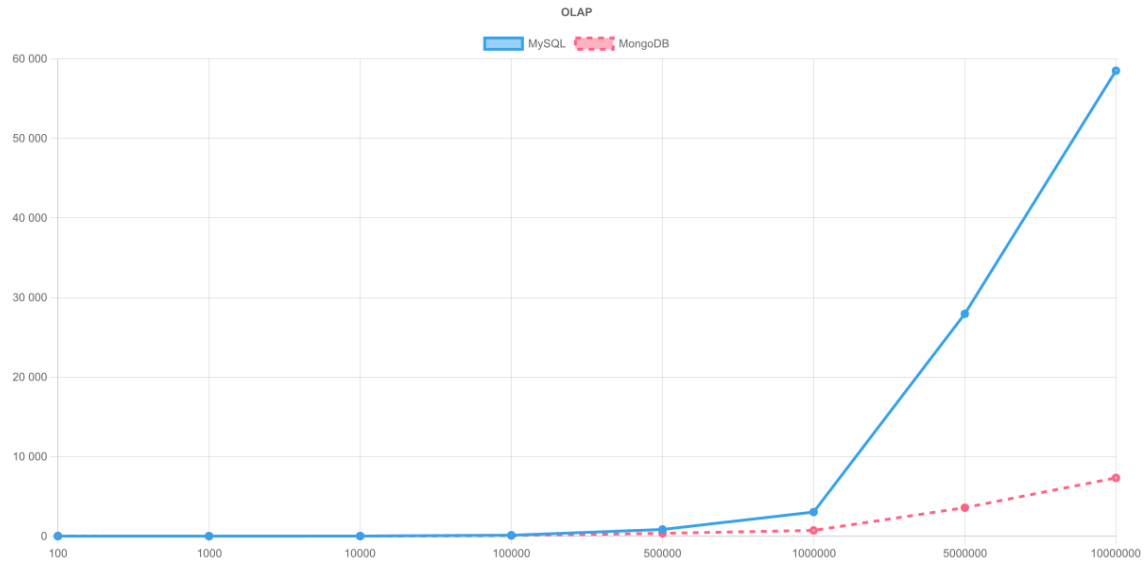

Рисунок 4 – График операции OLAP

Операции JOIN. Соединение таблиц в MySQL выполняется быстро и стабильно даже на больших объёмах: при 10 000 000 записей JOIN занимает 17 мс. MongoDB аналогичная операция требует значительно больше ресурсов CPU и памяти, а время выполнения достигает 89 634 мс (рисунок 5). Это подтверждает, что реляционная модель лучше подходит для сложных связей между таблицами.

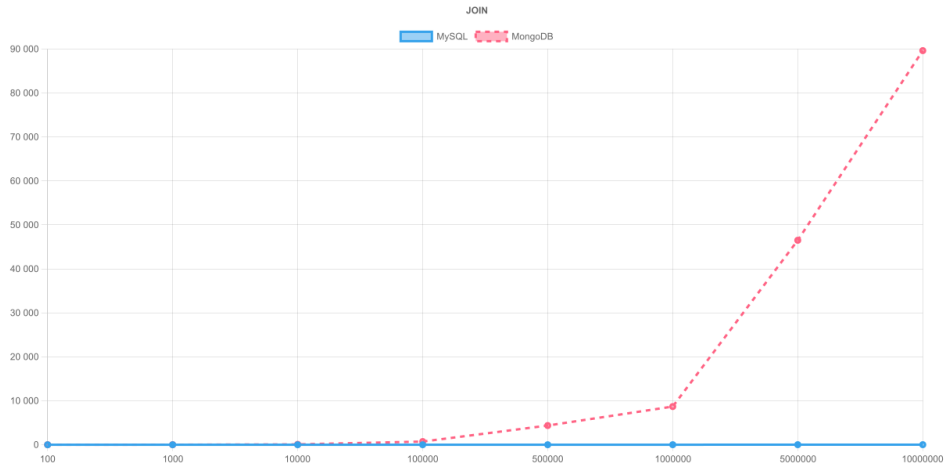

Рисунок 5 – График операции JOIN

Данные, приведенные в таблице 1, легли в основу графиков, визуализирующих масштабируемость исследуемых систем.

Таблица 1 – Результаты анализа

Операция	100	1000	10000	100000	500000	1 млн	5 млн	10 млн
MySQL_INSERT	15	47	453	3796	18914	39225	220178	441168
Mongo_INSERT	17	27	118	801	3838	8303	43755	85667
MySQL_SELECT	598	395	464	475	112	197	145	102
Mongo_SELECT	1005	590	883	7470	9015	18671	38412	32070
MySQL_UPDATE	1328	1453	1404	1388	438	383	228	105
Mongo_UPDATE	243	236	562	4569	4809	9739	22942	24512
MySQL_OLAP	2	2	11	116	843	3032	27937	58496
Mongo_OLAP	21	3	11	73	359	721	3578	7325
MySQL_JOIN	2	5	4	5	4	11	15	17
Mongo_JOIN	3	15	94	727	4369	8689	46503	89634

Заключение.

В ходе анализа показано, что MySQL эффективнее для транзакционных операций и сложных связей между таблицами, обеспечивая стабильность и быструю обработку выборок и обновлений. MongoDB превосходит при массовой вставке и аналитических запросах на больших объёмах данных благодаря гибкой структуре документов и встроенным механизмам работы с массивами. Выбор СУБД должен основываться на характере нагрузки, объёме данных и требованиях к масштабируемости.

Список использованных источников:

1. Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. / Eric Redmond, Jim R. Wilson. // Pragmatic Bookshelf, 2018.

2. MySQL [Электронный ресурс]. – Режим доступа: https://www.mysql.com. – Дата доступа: 24.03.2026.

3. MongoDB [Электронный ресурс]. – Режим доступа: https://www.mongodb.com. – Дата доступа: 24.03.2026.

UDC 004.6

SQL AND NOSQL PERFORMANCE ANALYSIS UNDER DIFFERENT LOAD PROFILES (OLTP AND OLAP)

Ryzhenkova E.I., Tserashonak M.N.

Institute of Information Technologies of the Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Bakunov A.M. – Senior Lecturer, Department of IS&T

Annotation. This article provides a comparative analysis of the performance of MySQL (a relational DBMS) and MongoDB (a document-oriented DBMS). Performance was assessed using simulated workload scenarios: OLTP (transactional operations) and OLAP (analytical queries). The experiment utilized data sets ranging in size from 100 to 10 million records, executing the following operations: INSERT, SELECT, UPDATE, OLAP, and JOIN. The primary objective of the study was to identify the specific behavior of each system when scaling data and query intensity.

Keywords. SQL, NoSQL, MySQL, MongoDB, OLTP, OLAP, performance, databases, load.