УДК [004.4:004.62]

ELASTICSEARCH. ПЕРЕХОД К NOSQL РЕШЕНИЯМ ДЛЯ ОРГАНИЗАЦИИ ЭФФЕКТИВНОГО ПОИСКА В БОЛЬШИХ ДАННЫХ

Бабич А.С.

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

Научный руководитель: Давыдович К.И. – ассистент кафедры ИПиЭ, магистр тех. наук

Аннотация. Рассмотрено применение Elasticsearch в качестве средства полнотекстового поиска и NoSQl-хранилища. Представлено описание работы инверсированного индекса и алгоритма ранжирования BM25. Освещены механизмы управления релевантностью поисковой выдачи и повышения возможностей: использование синонимов, корректировка опечаток, автодополнение поискового запроса. Приведено сравнение Elasticsearch по критериям производительности, масштабируемости и согласованности данных с классическими SQL решениями на примере PostgreSQL

Ключевые слова: полнотекстовый поиск, большие данные, скорость поиска, анализ логов, NoSQL, Elasticsearch.

Введение. В связи с ростом темпов цифровизации, а, следовательно и постоянным увеличением количества информации, представленной в электронном виде, появилась проблема её потребления, связанная с невозможностью обработки стремительного потока данных человеком. Для решения этой проблемы необходимо использовать специальные средства хранения и обработки, которые позволят с гибкостью управлять поиском и агрегацией информации для её анализа. В качестве такого универсального инструмента можно использовать Elasticsearch.

Основная часть. Сердцем работы эффективного поиска Elasticsearch является механизм индексации. Индекс – это основная единица хранения в Elasticsearch. Он представляет собой набор документов, однозначно идентифицированных именем или псевдонимом [1]. Его можно сравнить с таблицей из реляционных баз данных, но в свою очередь реализован куда сложнее, хотя и скрытно от пользователя. Индекс по факту является логической сущностью, с которой работает пользователь, объединяющей несколько его частей под одним именем. Каждая такая часть называется шардом и представляет собой отдельный поисковой движок. Запрос к индексу транслируется к каждой его части и после происходит конкатенация результатов. Шарды делятся на два вида первичные и реплики, служащие для отказоустойчивости и распределения read-нагрузки. Документ, так называется единица данных, попадая в индекс проходит процесс индексации, каждое текстовое поле проходит два шага [2]:

1 Tokenization – разбиение текста на более мелкие фрагменты (токены).

2 Nomalization – нормализует токены к стандартному виду, что позволяет сопоставлять похожие токены с поисковым запросом.

В результате образуется обратный индекс, каждый токен сопоставляется со списком документов, в которых он содержится.

Чем гибче инструмент поиска, тем более расплывчатые данные он может дать, в связи с этим необходимо разбивать данные по их релевантности. Elasticsearch использует для этого алгоритм BM25, использующий в своей основе следующие параметры: частота токена в документе, обратная частота токена во всех документах, отношение среднего размера документа к текущему.

Рассказав об основах строения и работы Elasticsearch можно перейти к его основной функции, а именно гибкий поиск. В качестве инструментов для гибкого поиска можно выделить: синонимы, автодополнение, коррекция опечаток, а так семантический поиск.

Синонимы позволяют связать различные токены друг с другом, в результате чего оба токена ведут на документ, содержащий хотя бы одного из них. Используется два

взаимозаменяемых принципа построения синонимов: на этапе индексации (предъявляет увеличенные требования к размеру индекса) и на этапе поискового запроса (позволяет более гибко работать с синонимами, но увеличивает время обработки запроса).

Автодополнение можно разделить на два вида: использование префикса (поиск по началу токена), основывается на дереве префиксов, и предложения, то есть Elasticsearch предлагает пользователю возможное продолжение слова в зависимости от хранящихся токенов в индексе.

Включение исправления опечаток (неточный поиск) при поиске по полям документа, чей формат поддерживает данную функцию, позволяет находить токены незначительно отличающиеся (на пару букв в зависимости от длинны слова).

Семантический поиск, позволяет по средством подключения моделей искусственного интеллекта преобразовывать текст в вектора и искать максимально схожий по смыслу текст, который значительно отличается в лексико-грамматическом плане.

Данные механизмы значительно упрощают поисковые операции человеку, который теперь может найти результаты среди большого количества данных, зная лишь общую направленность необходимой ему информации.

Пользователь может захотеть отдать приоритет тому или иному документу, по определенным признакам. Для решения этой проблемы используются механизмы управления релевантностью. Самый простой из них это boosting, позволяющий повысить ценность отдельной части поискового запроса, и function scope, повещающий scope (оценка релевантности) у документов, соответствующих некому условию.

Рассмотрим причины перехода к решениям NoSQL формата для поиска и работы с большими данными:

1 Гибкость хранения. Не смотря на более высокую сложность организации Elasticsearch не выдвигает строгих требований к постоянству форматов документов. Что особенно важно при работе с логированием, когда формат логов может меняется в зависимости тот источника и времени. В том числе для работы с логированием имеются удобные функции для агрегирования.

2 Дружелюбность управления. Классические реляционные базы данных требуют знания sql языка, что повышает требования к администраторам. Elasticsearch предлагает использование Kibana – интерфейса для запроса, анализа, визуализацией и управления данными, хранящимися в Elasticsearch [3].

3 Высокая производительность. На относительно небольшой выборке данных реляционные системы не отстают, а иногда и обгоняют по производительности NoSQL решения, но если говорить о больших данных, то различия становятся существенными. Приведем сравнительную характеристику Elasticsearch и реляционной базой данных PostgreSQL (см. таблицу 1).

Таблица 1 – Сравнительная характеристика Elasticsearch и PostgreSQL.

	Elasticsearch	**PostgreSQL**
Количество записей	100M+	<10M
Время полнотекстового поиска (10M записей)	80-200 мс	20-40 мс
Время агрегаций (10M записей)	300-800 мс	80-150 мс

Однако Elasticsearch имеет свои минусы, вызванные более сложной архитектурой, и не может стать основным хранилищем информации. Он требует более сложной стартовой настройки, не поддерживает механизм транзакций, что плохо сказывается на согласованности данных, а данные попадают в индекс лишь спустя определенное время.

На практике применяется следующий вариант совместного использования реляционных баз данных, таких как PostgreSQL, и Elasticsearch. Реляционная база данных выполняет функции источника истины сохраняя ACID-свойства, а Elasticsearch используется как поисковый движок, снимая нагрузку с транзакционной базы данных. Для такой схемы необходимо разработать механизм синхронизации данных между основной базой данных и

Elasticsearch. Данный механизм возможен через Dual Write схеме, когда приложение одновременно пишет данные в обе системы, главным недостатком данного варианта является что работоспособность системы одновременно зависит и от работы реляционной и от NoSQL базы данных. Более предпочтительной схемой является Change Data Capture паттерн (см. рисунок 1). Приложение заполняет данные только в реляционной базе данных, и его работоспособность зависит только от её. Приложение прослушивает основную базу и в случае её изменения, асинхронно обновляет данные в Elasticsearch, для этого могут использоваться Debezium или Kafka Connect. Единственным минусом этого варианта является задержка между сохранением данных в основную базу и появлении их в Elasticsearch, что является допустимым на фоне значительного повышения уровня надежности.

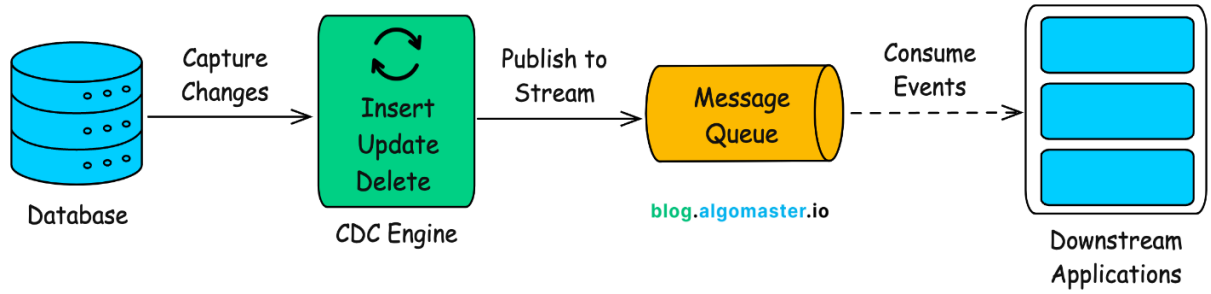

Рисунок 1 – Паттерн CDC

Заключение. Elasticsearch закрывает потребность в быстром и интеллектуальном поиске среди больших данных. Архитектура на основе шардирования и обратных индексов обеспечивает масштабируемость и отказоустойчивость. Рекомендуется использовать Elasticsearch в связке с реляционными базами данных для достижения баланса между гибкостью поиска и надежностью хранения данных.

Список литературы

1. *The Elasticsearh data store [Электронный ресурс]. – Режим доступа: htpps://elastic.co/docs/manage-data/data-store/. Дата доступа: 23.02.2026*

2. *Text analysis [Электронный ресурс]. – Режим доступа htpps://elastic.co/docs/manage-data/data-store/text-analysis. Дата доступа: 23.02.2026*

3. *Kibana [Электронный ресурс]. – Режим доступа htpps://elastic.co/kibana. Дата доступа: 23.02.2026*

UDC [004.4:004.62]

ELASTICSEARCH.TRANSITION TO NOSQL SOLUTIONS FOR ORGANIZING EFFICIENT SEARCH IN BIG DATA

Babich A.S.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Davydovich K.I. – Master. of Sci., Assistant at the Department of Engineering Psychology and Ergonomics

Annotation. The article examines the application of Elasticsearch as a tool for full-text search and NoSQL storage. A description of the inverted index operations and the BM25 ranking algorithm is presented. Mechanisms for managing search result relevance and enhancing capabilities are highlighted: the use of synonyms, typo correction and search query autocomplete. A comparison of Elasticsearch with classical SQL solutions using PostgreSQL as an example is provided based on performance? scalability and data consistency criteria.

Keywords: full-text search, big data, search speed, log analysis, NoSQL, Elasticsearch.