2. Болотов А., Гашков С., Фролов А., Часовских А. Элементарное введение в эллиптическую криптографию // Москва: КомКнига, 2006. – 328 с.

Поліщук О.В.
студентка,
Національний технічний університет України
«Київський політехнічний інститут»

ПЕРСПЕКТИВНИЙ ПОШУКОВИЙ СЕРВЕР ELASTICSEARCH ЯК НОВИЙ ПІДХІД ПОШУКУ ЗБЕРЕЖЕНИХ ДАНИХ

Збереження даних на сьогодні є звичним процесом. Дану функцію виконують, як реляційні бази даних, так і NoSQL. Останні все більше стають популярним, маючи ряд переваг як у об'ємі збереження даних, так і у структурі збереження інформації. Та вони мають певний недолік, який з іншої ж сторони є й перевагою, а саме структура збереження. Більш того ця структура тягне за собою іншу, не менш важливу проблему – пошук по базі даних. Тому все-більш стають популярними різноманітні пошукові системи, такі як Lucene, ElasticSearch, які дають можливості пошуку по різним полям, додавання інформацію в індекси без додаткової переіндексації, заважтаження даних без додаткового блокування таблиць, зниження навантаження на базу при здійсненні join запитів.

Порівняно з найближчими конкурентами, ElasticSearch дає можливість швидкого пошуку, так як здійснює онлайн індексації, підтримує шардинг, реплікацію, витягує із пошуку оригінальні дані. Розподіленість – тренд сьогодення, все частіше розробники намагаються забезпечити розподіленість тим чи іншим чином, що дає можливість розгортання даних у хмарах.

В ElasticSearch не потрібно вказувати схему даних завчасно, достатньо відправити JSON запит, і сервер самостійно виконає операції для визначення типу. Також розробник може задати типи даних, структуру документу, аналізатор, який використовуватиметься, промапити індекси, тощо.

Звісно для розробників із досвідом зручним буде задати схему документів для збереження інформації й така можливість існує. Більш того пошуком можна керувати, починаючи від задання у якому документі здійснювати пошук, закінчуючи вказанням еквівалентності пошуку тої чи іншої фрази.

Пошук можна здійснювати, використовуючи фасет term або ж аналізатор match. Досить часто помилкою розробників є незнання, того що term не є аналізатором, й пошук здійснюється за непростим алгоритмом. Наприклад, term здійснює пошук, лише якщо слово пишеться у нижньому регістрі (у запиті), у самому ж ElasticSearch текст може бути як у нижньому так й у верхньому регістрі. Term має ряд специфічних функцій, й для пошуку який є очікуваний й бажаний користувачу все ж варто використовувати match query. З іншої

сторони term дуже швидкий, так як не використовує аналізаторів. Він швидший у середньому в 10 разів порівняно з phrase query та у 20 разів із proximity querries. Також фільтри швидші, ніж query. З іншої сторони фільтри працюють із точним співпадінням полів. Тому для швидкодії варто використовувати фільтри та term запити.

Term не працює з масивом даних. Правильно висловитися працює, й пошук здійснює по останньому елементу масиву. Відразу таке рішення не є зрозумілим. Справа у тім, що term може здійснювати пошук лише по одному полю й тому береться останній аргумент масиву. Звичайно такий підхід не є добрим, й потрібно використовувати більш наглядний спосіб.

Першим варіантом розв`язання даної задачі є поєднання запитів. Просто, здавалося б, та є ще кращий варіант – terms, який на вхід отримує масив й працює за правилом OR. Тобто достатньо хоча б одного співпадіння, щоб об’єкт був повернений на запит. Останній варіант є правильним, так як є інтуїтивно зрозумілим й пошук здійснюється за один прохід, а не з’єднання декількох результатів пошуку.

Варто звернути увагу на параметри за замовченням. Даний момент досить часто призводить, до того, що сам плагін використовується не продуктивно. Наприклад, створення складних запитів, чи отримання даних із кешу, а потім фільтрування їх у самому проекту. Такий підхід призводить до нераціонального використання всіх можливостей продукту, адже сам продукт має величезні можливості й потужні алгоритми, що здійснюють пошук дуже швидко й писати додатковий код для фільтрації даних просто не має сенсу, адже це вже все зроблено й зроблено оптимально.

Пошук здійснюється з використанням HTTP запитів й передаванні в тілі запиту JSON. За замовченням, пошук здійснюється по всьому сховищу, по всім документам, по всім полям. У самому ж запиті можна вказати назву документа, у якому здійснювати пошук та назви полів, по яким здійснювати пошук. Аналогічно у запиті й вказується операція, яка здійснюється. Це може бути не лише пошук, а й обрахунок кількості даних, що відповідають вимогам, вказаних у JSON; індексація даних, збереження даних у сховищі, агрегація, тощо. Запит повертає 10 записів, якщо користувачу потрібна більша кількість, то для цього необхідно передати значення як GET-параметри або у тілі JSON запиту. Даною властивістю варто користуватися, розуміючи, що результат приходить у відсортованому вигляді, тобто топ 10 елементів, які мають найбільший рівень співпадінь з вхідним параметром.

З іншої сторони, розробник може не лише задати ті поля у яких здійснювати пошук чи важливість знаходження даних у одних полях, порівняно з іншими, а й ті які потрібно повернути у response, що значно пришвидшує опрацювання отриманих даних. Тобто пошук здійснювати по даті шоу, а повернути лише назву шоу (якщо розглядати що у нас є сутність шоу з полями дата, назва, тощо). Як і у багатьох мовах тут існують регулярні вирази, які допомагають зменшити об`єм тексту у request і як наслідок у швидкому

прочитанні й розумінням скрипту іншим розробником. Це можуть бути й вирази на шаблони назв полів, й на вміст інформації.

Фільтр на результат можна встановити й повертати лише ті дані, відповідність яких задовольняє певному відсотку, або ж вказати, що пошук здійснюється із можливістю «префіксу». Тобто розглядається як фраза, по якій здійснюється пошук може мати префікс. А operator вказує як поєднувати фрази, чи розглядати слова по окремості, чи фраза повинна бути представлена вся. Більш того, можна задати чи результат повинен бути обов'язково представлений у відповіді, чи лише один із параметрів пошуку, вказати мінімальну кількість параметрів, які повинні бути у повернутому результаті згідно висунутим вимогам у запиті. Тут навіть можна задавати математичні вирази, комбінації слів, яким повинен відповідати результат, відсоткову відповідність. Цю особливість варто використовувати для статистичної обробки інформації, для аналізу вхідних даних, дослідження того чи іншого питання.

Цікавим є оператор slop, який дозволяє розглядати фразу як не строгу послідовність слів, а варіації з нею. Це є пошук не лише фрази у строгому порядку, а й у оберненому, чи присутність одного, двох й більше слів між ними, між переставленими місцями словами тощо. Дана функція добре підходить для пошуку наявності фрази у тексті з певним відхиленням, присутністю прикметників, непрямого порядку слів, присутність іменника між словами ключової фрази.

Range прекрасно розв'язує проблему пошуку більшого, меншого значення. При поставленні задачі, коли необхідно знайти дані які були створенні після 1 квітня 2015 року до сьогодні мінус 1 година. Даний ресурс має рішення й дозволяє здійснювати не лише по числовим характеристикам дати, чисел, а й за лексикографічним порядком.

Пошук можна здійснювати й за наявність пустих полів. Проблемна зона – одне поле містить масив даних, у такому випадку також можна знайти пусті поля, використавши exists чи missing й здійснювати присутності не повних даних тощо.

Також присутній пошук накшталт like із реляційних баз. Та уваги заслуговує fuzzy_like_this, де подібність визначається на основі співпадіння частини слова, пошук подібних слів, які відрізняються лише декількома словами; написання скриптів, де ще на рівні запиту можна вказати як перетворити вихідні дані, за яких умов, тощо.

Пошуковий сервер повністю виправдовує свою назву – починаючи його удосконалення порівняно із батьківським сервером Lucene (швидкість ElasticSearch у десятки разів перевищує Lucene), закінчуючи еластичним пошуком подібних слів, які не є точними по відношенню до пошукового запиту. Швидкодія даного продукту вражає, і все більше комерційних проекті використовують його, при чому сам продукт є безкоштовним і для досвідчений розробників не є проблемою його локального розгортання. Головною задачею користувачів продукту є вміння правильно використовувати наявні можливості, і лише у такий спосіб пошуковий сервер виправдає свою назву.

Список використаних джерел:

1. Elastic Revealing Insights from Data (Formerly Elasticsearch) [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – U.S., 2015. – Режим доступу: https://www.elastic. co/ (дата звернення 21.04.2015) – Назва з екрана.

Рушанян Г.М.

студент,
Національний технічний університет України
«Київський політехнічний інститут»

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ-ДОДАТКА «РОЗКЛАД»

Серед багатьох можливих напрямів розробок підсистем, орієнтованих на мобільні пристрої, найбільш пріоритетним є створення та впровадження підсистеми «Розклад», функціоналом якої буде переважно перегляд розкладу занять навчальної групи, розкладу занять та консультацій викладачів, сповіщення студентів про перенесення занять на інший час, інтеграція з функціоналом системи «Електронний кампус НТУУ «КПІ»« в задачах перегляду методичного забезпечення кредитного модулю, результатів поточного контролю знань, умінь та навичок студентів.

Дані дослідження існуючих підсистеми

Більшість студентів вже давно зіткнулись з проблемою відсутності зручного електронного розкладу занять з підтримкою мобільних платформ.

Дослідження «ринку» показало наявність вже існуючих додатків розкладу занять, але у них знайдені певні недоліки:

1. Веб-ресус «rozklad.kpi.ua» – офіційний сервіс з функціоналом «розклад», котрим користується більшість студентів НТУУ «КПІ». На жаль, цей ресурс не має підтримки мобільних платформ, саме тому не задовільняє потреби користувачів та не є універсальним.

2. Мобільний додаток «KPI weeks» – на перший погляд повністю реалізована ідея, але основна проблема даного сервісу – це неможливість отримувати оновлений адміністрацією або викладачами розклад занять. Як результат – додаток має дуже обмежений функціонал.

За співробітництвом зі структурним підрозділом НТУУ «КПІ» – Конструкторським бюро інформаційних систем (КБ ІС) отримані сервіси, що базуються на реальних даних та забезпечують можливість розробки такого функціоналу.

Цільовими платформами для додатку обрані Android та IOS, так як більшість користувачів використовує саме ці платформи. В майбутньому також запланована підтримка Windows Mobile.

Додатково новий мобільний додаток повинен задовольняти наступним вимогам: