Schweizer Taschenmesser für Daten

Serverlose Suchmaschinen

Peter Soth

Dieser Artikel zeigt, dass sich Suchmaschinentechnologien für die unterschiedlichsten Anwendungsfälle – vergleichbar mit einem Schweizer Taschenmesser – eignen, auch wenn diese zurzeit fast ausschließlich für die Suche in unstrukturierten Daten genutzt werden. Ob sich Suchmaschinen als disruptive Technologie etablieren und existierende, wie Data-Warehouses (DWH), ablösen können, wird sich in den nächsten Jahren erst noch zeigen.

Suchmaschinen werden täglich genutzt, um in unstrukturierten Informationen, wie Dokumenten oder Webseiten, zu recherchieren. Dass man diese Technologie noch weitaus universeller einsetzen kann, zeigen die Autoren Gregory Grefenstette und Laura Wilber in ihrem Buch „Search Based Applications“ [SBA2010] auf. Die Suchmaschine bildet hierbei die Kernkomponente für den Informationszugriff und das Reporting innerhalb einer „Search Based Application“ (SBA).

Solch eine Suchlösung ist gleichermaßen für die Verarbeitung von gewaltigen Mengen an unstrukturierten sowie strukturierten Daten geeignet und ermöglicht neuartige Lösungsansätze nicht nur bei der Big-Data-Thematik. So ist es beispielsweise denkbar, in Drittsystemen – wie ERP oder CRM – gespeicherte strukturierte Informationen mit unstrukturierten (Dokumente, Social Media usw.) zu verknüpfen. Auf diese Weise kann man Analysen durchführen, die mit den klassischen Business Intelligence (BI) Tools nicht ohne Umwege zu bewerkstelligen sind.

Kommerzielle versus Open-Source-Lösungen

Es gibt unzählige Anbieter im Bereich Suchmaschinen. Zu den namhaften zählen Firmen wie Oracle mit Endeca, Dassault Systems mit Exalead oder Hewlett Packard mit IDOL. Im Open-Source-Bereich haben sich Apache Solr und Elasticsearch durchgesetzt, beide basieren auf der Such-Bibliothek Apache Lucene. Alle weiteren Erläuterungen in diesem Artikel stützen sich auf mit Elasticsearch gesammelte Erfahrungen.

Von NoSQL und dem invertierten Index profitieren

Für das Verständnis des Unterschieds von NoSQL und SQL kann als Analogie das Parken eines Autos in einem Parkhaus herangezogen werden. Bei einer relationalen Datenbank (RDBMS) wird der PKW komplett in seine Komponenten zerlegt und diese werden einzeln abgelegt oder „geparkt“, mit dem Vorteil, dass weniger Raum beansprucht wird. Bei NoSQL wird hingegen das Auto als Ganzes „geparkt“ beziehungsweise gespeichert.

Da sich unstrukturierte Daten nur unzureichend in einem relationalen Datenmodell persistieren lassen, verwenden die meisten Suchmaschinen analog zu NoSQL-Datenbanken JSON-Dokumente, die schemafrei verwendbar sind. Ein Dokument durchläuft auf dem Weg in den Suchindex einen Analyseprozess. Hierbei wird unter anderem der Tokenizer involviert, der eine Aufspaltung in einzelne Terme – beispielsweise durch Leerzeichen getrennt – vornimmt und diese in den invertierten Index einfügt. Diesen kann man sich wie das Stichwortverzeichnis am Ende eines Buches vorstellen, mit dessen Hilfe ein Suchbegriff in den JSON-Dokumenten rasch wiederauffindbar ist.

Abgrenzung von Suchmaschinentechnologien zu NoSQL und SQL

Es stellt sich des Öfteren die Frage, ob eine Suchlösung nicht auch ein klassisches RDBMS ersetzen kann. Dies hängt von den Anforderungen ab. Im nächsten Abschnitt werden einige Einsatzszenarien aufgezeigt. Ein RDBMS ist im Vergleich zu NoSQL oder einer Suchmaschine noch immer die erste Wahl, wenn Zuverlässigkeitsgarantien (ACID) bei der Datenverarbeitung unablässig sind. Bei NoSQL gibt es mit OrientDB [ORI2016] einen vielversprechenden Anbieter, der gleichfalls ACID-Transaktionen unterstützt. Falls sichere Transaktionen benötigt werden, ist ein hybrider Ansatz vorzuziehen. Hierbei wird eine ACID-konforme Datenbank (NoSQL oder SQL) für die transaktionssichere Persistierung der Daten benutzt. Die Abfrage kann hingegen über eine Suchmaschine erfolgen.

Wo lassen sich Suchtechnologien einsetzen?

Dieser Abschnitt beschreibt mögliche Einsatzszenarien von Suchtechnologien und ihre möglichen Mehrwerte gegenüber klassischen relationalen Datenbanken.

- Suchen und Finden von Informationen: Dies ist die klassische Disziplin einer Suchmaschine. Es kann mit ihr sowohl in unstrukturierten als auch strukturierten Daten (Tabellen) gesucht werden. Damit ist auch eine unternehmensübergreifende Suche denkbar, die in verschiedenen Drittsystemen versteckte Informationen durchsuchbar macht. Im Vergleich zu SQL ist hier noch hinzuzufügen, dass eine Textsuche innerhalb einer Datenbanktabelle mit dem Like-Operator in SQL nicht so flexibel ist. Des Weiteren ist eine facettierte Suche, wie sie aus Online-Shops bekannt ist, viel einfacher als mit traditionellem SQL umzusetzen.
- Query-Turbo für SQL-Datenbanken: Komplexe SQL-Statements – beispielsweise hierarchische oder tief kaskadierte – können auf einer relationalen Datenbank äußerst träge sein. Dieser Umstand ist besonders häufig bei Web-Portalen mit deren vielfältigem Rollenkonzept anzutreffen. Aus diesem

Grund bietet es sich hier an, eine Suchmaschine wie Elasticsearch für die Beschleunigung von SQL-Abfragen einzusetzen. Als prominentes Beispiel kann an dieser Stelle das Liferay-Portal erwähnt werden. Die vielen Datenbankabfragen des Asset-Publisher-Portlets bedingen die mangelhafte Performanz des Liferay-Portals in der Version 6.1. Ab 6.2 wird deshalb ein Lucene-Index anstatt der DB-Abfragen für das Asset-Publisher-Portlet innerhalb von Liferay verwendet [LIF2013].

- Verarbeitung von GEO-Daten: Die Verarbeitung von GEO-Daten unterstützen die Anbieter von RDBMS durch Spatial-Erweiterungen. Suchmaschinen ermöglichen ebenfalls das Suchen und Filtern von raumbezogenen Informationen und dies mit einer unkomplizierteren Syntax, die noch dazu eine geringere Einarbeitungszeit erfordert, bei der Abfrage. Hierbei sei jedoch anzumerken, dass komplexe Fragestellungen, die bei einem Geoinformationssystem (GIS) machbar sind, nicht durchgeführt werden können, da der Sprachumfang nur einfache Abfragen ermöglicht (die GeoDistanceQuery von Elasticsearch unterstützt nur die Distanz zu einem Punkt, nicht jedoch zu einem Polygon). Des Weiteren fehlt die Möglichkeit einer Routenplanung, da die hierfür benötigte Unterstützung von Graphen noch nicht existiert. Es ergeben sich trotzdem äußerst interessante Einsatzszenarien. Beispiele sind Postleitzahlenumkreissuchen oder man indiziert OpenStreetMap [EXB2013] und kann damit in einem Online-Shop beziehungsweise CRM-System die vorhandenen Adressen überprüfen. Für ein Immobilien-Portal ist es weiterhin denkbar, im Umkreis einer Immobile alle öffentlichen Einrichtungen, wie etwa Schulen, zu finden.
- Business Intelligence: Suchmaschinen eignen sich ebenfalls für die Analyse von multidimensionalen Informationen. Dabei entfällt die von einem klassischen Data-Warehouse (DWH) bekannte nächtliche Voraggregation für die Auflösung komplexer SQL-Joins; man erhält quasi ein Realtime-DWH. Für eine Auswertung wird zum Beispiel eine Datenbank, die die zu analysierenden Zahlen enthält, mit Elasticsearch indiziert. Dieser Extraktions-, Transformations- und Lade-Prozess (ETL) beansprucht in üblichen DWH-Projekten fast 80 Prozent des Projektaufwands und kann in manchen Fällen mit Synonymlisten (Straße == Strasse) vereinfacht werden. Die Abbildung von Dimensionen erfolgt im herkömmlichen DWH unter Zuhilfenahme des Star-Schemas. In Elasticsearch geschieht dies dynamisch über Aggregationen, mit deren Hilfe gefiltert wird, analog zum „Drill-Down“ und „Roll-Up“ in Pivot-Tabellen. Zusätzlich könnte man eine Volltextsuche nutzen, um unstrukturierte Daten zu durchsuchen. Mit dieser kann man darüber hinaus die Navigation in einem umfassenden DWH mit zahlreichen Dimensionen verbessern.
- Internet of Things (IoT): RDBMS sind nicht für die Speicherung immenser Datenmengen geeignet, Elasticsearch hingegen ist speziell hierfür ausgelegt worden. Neben der Datenhaltung und den Standardabfragemöglichkeiten werden auch verschiedene Analysemöglichkeiten angeboten. So kann man etwa Sensordaten mit Hilfe eines Zeit-Histogramms auf Stundenbasis aggregieren. Mit einer weiteren Sub-Aggregation wird anschließend der Durchschnittswert der Messdaten berechnet, und das fast wiederum in Echtzeit. Über das Percolator-API von Elasticsearch besteht zusätzlich die Möglichkeit Alarme zu setzen, falls beispielsweise eine Temperaturgrenze überschritten wird [EXS2014].

Bei allen etwaigen Vorteilen gilt es jedoch zu beachten, dass sich die hohe Geschwindigkeit von Elasticsearch – verglichen mit Hadoop – bei der Auswertungsgenauigkeit umgekehrt proportional verhält. Allerdings nur bei umfangreichen Datenmengen, die eine Verteilung auf viele Server innerhalb eines Clusters erfordern.

So ermittelt beispielsweise jeder Knoten für sich die Liste der TOP-5-Kunden, diese TOP-5-Klienten stimmen als Einzelergebnis, aber nicht zwingend über alle Knoten des Clusters. Elasticsearch konsolidiert demzufolge nicht die geteilten Analyse-Ergebnisse. Diese Ausnahme gilt nicht, falls nur ein Knoten im Einsatz ist. Bei Big-Data-Projekten zählt jedoch meistens mehr der Gesamtzusammenhang der Informationen als der einzelne Datensatz. Oder anders formuliert: Mit einer Suchmaschine wird man vermutlich kein SOX-konformes Konzernreporting für die Jahresbilanz aufsetzen wollen.

Codebeispiel

Als Musterbeispiel dient eine einfache Pivottabelle aus dem Buch „Excel 2010 im Controlling“ [EXL2010]. Hierbei sollen die Umsatzzahlen pro Kunde für die einzelnen Monate aggregiert werden. Die Umsätze stehen als CSV-Datei zur Verfügung. Der Import erfolgt mit dem Tool Logstash, das Skript findet man auf dem exensio Blog [EXB2016]. Listing 2 zeigt, wie ein Zugriff auf Elasticsearch mit Java umzusetzen ist, und Listing 1 stellt das Ergebnis in Form einer ASCII-Tabelle dar. Das hier gewählte Beispiel ist absolut schlicht gehalten, um eine erste Idee zu bekommen, wie unkompliziert mit dem Elasticsearch-API programmiert werden kann, und ist natürlich ebenso mit folgender trivialen SQL-Abfrage lösbar:

```
select Kunde, sum(Januar) as Januar, sum(Februar) as Februar,
             sum(März) as März
from umsatz group by Kunde order by Kunde;
```

Die Summen auf Zeilenebene könnte man gleichermaßen innerhalb von Elasticsearch mit einem Skript ermitteln, jedoch sind diese aus Sicherheitsgründen seit Version 1.4.3 nicht länger standardmäßig eingeschaltet. Die Endsummen auf Spaltenebene könnten mit Hilfe einer zweiten Suchabfrage implementiert werden. Neben dem Java-API gibt es auch die Möglichkeit, direkt von Java über das RESTful-API mit Elasticsearch zu kommunizieren.

Wie oben erwähnt, dient dieses Beispiel nur als Einstieg in die Thematik. Elasticsearch kann seine Stärken in Vollkommenheit erst ausspielen, wenn komplexe SQL-Joins für die Datenanalyse benötigt und zusätzlich gewaltige Datenmengen verarbeitet werden müssen.

```
|                KUNDE|       JAN|       FEB|       MAE|    GESAMT|
|         Beispiel GmbH|  12710,00|  10428,80|  15203,10|  38341,90|
|              Dummy AG| 106392,00| 109992,50| 114392,00| 330776,50|
|         Felix Test AG|  13250,00|  11550,00|  15450,00|  40250,00|
|        Muster & Söhne|     79,90|     95,88|     79,90|    255,68|
|             Muster AG|  23406,50|  14112,50|  17476,70|  54995,70|
|           No Name GbR|    125,00|    262,50|    362,50|    750,00|
|           P. Robe GbR|   1117,50|   1188,00|   1240,00|   3545,50|
|            Probe GmbH|   9879,65|  16000,00|  14495,88|  40375,53|
|        Test & Partner|    719,10|    998,75|    958,80|   2676,65|
|             Test GmbH|   8000,00|   9600,00|   9600,00|  27200,00|
|              Übung AG|  29730,00|  36645,00|  35172,00| 101547,00|
| Übungsgesellschft mbH|   2625,00|   3750,00|   4250,00|  10625,00|
|               GESAMT| 208034,65| 214623,93| 228680,88| 651339,46|
```

Listing 1: Ergebnis der Suche in Form einer ASCII-Tabelle

```
package de.javaspektrum;

import org.elasticsearch.action.search.SearchRequestBuilder;
import org.elasticsearch.action.search.SearchResponse;
import org.elasticsearch.client.Client;
import org.elasticsearch.client.transport.TransportClient;
import org.elasticsearch.common.transport.InetSocketTransportAddress;
import org.elasticsearch.index.query.QueryBuilders;
import org.elasticsearch.search.aggregations.AggregationBuilders;
import org.elasticsearch.search.aggregations.bucket.terms.Terms;
import org.elasticsearch.search.aggregations.metrics.sum.Sum;

import java.net.InetAddress;
import java.net.UnknownHostException;

public class elastic {
  private static final String HEADER_FORMAT =
    "|%25s|%10s|%10s|%10s|%10s|\n";
  private static final String ROW_FORMAT =
    "|%25s|%10.2f|%10.2f|%10.2f|%10.2f|\n";

  private static Client getESClient() throws UnknownHostException {
    return TransportClient.builder().build()
      .addTransportAddress(new InetSocketTransportAddress(
        InetAddress.getByName("localhost"), 9300));
  }

  private static SearchResponse getPivotData()
      throws UnknownHostException {
    Client client = getESClient();
    SearchRequestBuilder requestBuilder = client.prepareSearch();
    requestBuilder
      .setSize(0)
      .setQuery(QueryBuilders.matchAllQuery())
      .addAggregation(
      AggregationBuilders
        .terms("Umsatz")
        .field("Kunde.raw")
        .size(0)
        .order(Terms.Order.term(true))
        .subAggregation(AggregationBuilders.sum(
          "Januar").field("Januar"))
        .subAggregation(AggregationBuilders.sum(
          "Februar").field("Februar"))
        .subAggregation(AggregationBuilders.sum(
          "März").field("März"))
    );
    SearchResponse response = requestBuilder.execute().actionGet();
    client.close();
    return response;
  }

  private static void printPivot() throws UnknownHostException {
    SearchResponse response = getPivotData();
    Terms revenues = response.getAggregations().get("Umsatz");
    final Double[] sum_col = {0.0, 0.0, 0.0, 0.0};
    System.out.printf(HEADER_FORMAT, "KUNDE", "JAN",
      "FEB", "MAE", "GESAMT");
    revenues.getBuckets().forEach(
      bucket -> {
        Double jan = ((Sum) bucket.getAggregations().get(
          "Januar")).getValue();
        sum_col[0] += jan;
        Double feb = ((Sum) bucket.getAggregations().get(
          "Februar")).getValue();
        sum_col[1] += feb;
        Double mar = ((Sum) bucket.getAggregations().get(
          "März")).getValue();
        sum_col[2] += mar;
        sum_col[3] += jan + feb + mar;
        System.out.printf(ROW_FORMAT, (String) bucket.getKey(),
            jan, feb, mar, jan + feb + mar);
      });
      System.out.printf(ROW_FORMAT, "GESAMT",
        sum_col[0], sum_col[1], sum_col[2], sum_col[3]);
    }

    public static void main(String[] args) throws UnknownHostException {
      printPivot();
    }
  }
```

Listing 2: Zugriff auf Elasticsearch mit Java

Fazit

Suchmaschinen eignen sich für weit mehr als nur die Suche in unstrukturierten Daten. Sie können vorhandene Softwareapplikationen um zusätzliche interessante Funktionen erweitern, die mit bestehenden Technologien vollumfänglicher, aber auch kostspieliger umzusetzen wären. In vielen Situationen wird eine „kleinere" Lösung höchstwahrscheinlich vollkommen ausreichen. Suchtechnologien kann man auf jeden Fall ausgesprochen universell einsetzen und sollten demzufolge im Projektgeschäft häufiger in Betracht gezogen werden.

Literatur und Links

[ELA2016] Scripting, **https://www.elastic.co/guide/en/elasticsearch/reference/2.2/modules-scripting.html**

[EXB2013] Geocoding (Teil 2) mit Elasticsearch und OpenStreetMap, exensio IT blog, 14.12.1013, **http://blog.exensio.de/2013/12/geocoding-mit-elasticsearch-und.html**

[EXB2016] Pivot-Tabellen mit Elasticsearch und Kibana anstatt Excel geht das?, exensio IT blog, 14.2.2016, **http://blog.exensio.de/2016/02/pivot-tabellen-mit-elasticsearch-und.html**

[EXL2010] St. Nelles, Excel 2010 im Controlling, Rheinwerk-Verlag, 2010

[EXS2014] T. Kraft, Elasticsearch und IoT, Search Meetup Karlsruhe, 16.10.2016, **https://speakerdeck.com/exensio/elasticsearch-und-iot**

[LIF2013] Asset Publisher now queries by search index instead of database, to increase performance, LRDOCS-486, **https://issues.liferay.com/browse/LRDOCS-486**

[ORI2016] OrientDB, **http://orientdb.com/orientdb**

[SBA2010] G. Grefenstette, L. Wilber, Search-Based Applications: At the Confluence of Search and Database Technologies, Morgan & Claypool Publishers, 2010

Peter Soth arbeitete nach seinem Studium der Elektrotechnik und Informatik mehrere Jahre als Software-Engineer und Senior Consultant bei international tätigen Unternehmen, wie Hewlett Packard, SAS Institute und BEA Systems (nun Oracle). Im Jahre 2006 gründete er die exensio GmbH mit. Seine Tätigkeitsschwerpunkte umfassen die Architekturberatung und Realisierung von Enterprise-Projekten mittels Java-Technologien.
E-Mail: peter.soth@exensio.de