Data Engineering

Elasticsearch #4

Aggregations

Aggregations

- Aggregationen fassen Daten als Metriken, Statistiken oder andere Analysen zusammen
- Ermöglichen das Beantworten von Fragen wie:
  - Was ist die durchschnittliche Ladezeit der Website?
  - Wer sind die wertvollsten Kunden basierend auf dem Transaktionsvolumen?
  - Wie viele Produkte gibt es in jeder Produktkategorie?

Kategorien von Aggregationen

- **Metric Aggregations**: Berechnen Metriken wie Summe oder Durchschnitt aus Feldwerten.
- **Bucket Aggregations**: Gruppieren Dokumente in Buckets basierend auf Feldwerten, Ranges oder anderen Kriterien
- **Pipeline Aggregations**: Nutzen die Ausgabe anderer Aggregationen als Input, anstatt direkt auf Dokumente oder Felder zuzugreifen.

Aggregationen ausführen

- Aggregationen werden über den „Search“-Endpunkz ausgeführt, indem der Parameter **"aggs"** angegeben wird
- Aggregations können zusammen mit “normalen“ Queries verwendet werden, um den Datensatz, auf dem die Aggregation ausgeführt wird einzuschränken
- Es können mehrere Aggregations in einem Request durchgeführt werden
- Aggregations sind immer benannt

Einfache Aggregation

```
GET /my-index-000001/_search
{
  "aggs": {
    "my-agg-name": {
      "terms": {
        "field": "my-field"
      }
    }
  }
}
```

Aggregation – Response


```
{
  "took": 78,
  "timed_out": false,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 5,
      "relation": "eq"
    },
    "max_score": 1.0,
    "hits": [...]
  },
  "aggregations": {
    "my-agg-name": {
      "doc_count_error_upper_bound": 0,
      "sum_other_doc_count": 0,
      "buckets": []
    }
  }
}
```

Kombination Aggregation & Query

```
GET /my-index-000001/_search
{
  "query": {
    "range": {
      "@timestamp": {
        "gte": "now-1d/d",
        "lt": "now/d"
      }
    }
  },
  "aggs": {
    "my-agg-name": {
      "terms": {
        "field": "my-field"
      }
    }
  }
}
```

Mehrere Aggregations

```
GET /my-index-000001/_search
{
  "aggs": {
    "my-first-agg-name": {
      "terms": {
        "field": "my-field"
      }
    },
    "my-second-agg-name": {
      "avg": {
        "field": "my-other-field"
      }
    }
  }
}
```

Size

- Aggregations liefern neben dem Aggregationsergebnis auch ein Set der Dokumente aus, auf denen die Aggregation ausgeführt wurde
- In vielen Fällen ist dies nicht notwendig und kostet unnötige Ressourcen
- Wird der „size“ parameter auf 0 gesetzt, liefert die Reqeust ausschließlich das Ergebnis der Aggregation zurück

```
GET /my-index-000001/_search
{
  "size": 0,
  "aggs": {
    "my-agg-name": {
      "terms": {
        "field": "my-field"
      }
    }
  }
}
```

Bucket Aggregations

Bucket Aggregations

- Erstellen Buckets (Gruppen) von Dokumenten basierend auf bestimmten Kriterien, anstatt Metriken über Felder zu berechnen wie Metrics Aggregationen (ähnlich wie „group by“)
- Jedes Bucket ist mit einem Kriterium verknüpft, das bestimmt, ob ein Dokument hineinfällt
- Buckets definieren effektiv Mengen von Dokumenten.
- Zusätzlich wird die Anzahl der Dokumente in jedem Bucket berechnet und zurückgegeben.

Bucket Aggregations

- Können Sub-Aggregationen enthalten, die für die Dokumente in ihren Buckets ausgeführt werden
- Arten von Bucket Aggregatoren:
  - Einige definieren ein einzelnes Bucket
  - Andere definieren eine feste Anzahl von Buckets
  - Manche erstellen Buckets dynamisch während des Aggregationsprozesses

```
GET /my-index-000001/_search
{
  "aggs": {
    "my-agg-name": {
      "terms": {
        "field": "my-field"
      },
      "aggs": {
        "my-sub-agg-name": {
          "avg": {
            "field": "my-other-field"
          }
        }
      }
    }
  }
}
```

Terms-Aggregation

- Bucket Aggregation die auf einem Feld der Dokumente im Index ausgeführt wird
- Gruppiert nach dem Wert des Feldes
- Beispiel: Feld Produktkategorie: Gruppiert alle Dokumente, die die gleiche Produktkategorie enthalten
- Field **muss** vom Typ Keyword sein – nicht auf Freitext ausführbar

Terms-Aggregation

```
GET /_search
{
  "aggs": {
    "genres": {
      "terms": { "field": "genre" }
    }
  }
}
```

```
{
  ...
  "aggregations": {
    "genres": {
      "doc_count_error_upper_bound": 0,
      "sum_other_doc_count": 0,
      "buckets": [
        {
          "key": "electronic",
          "doc_count": 6
        },
        {
          "key": "rock",
          "doc_count": 3
        },
        {
          "key": "jazz",
          "doc_count": 2
        }
      ]
    }
  }
}
```

Terms-Aggregation

- Terms-Aggregation gibt die top 10 Terms eines Felds zurück
- Size Parameter der Aggregation kann die Anzahl der Terms erhöhen resultiert in höherer Memory Consumption
- Terms werden pro Shard berechnet: **Bei vielen Terms kann das zu ungenauen Ergebnissen führen!**
- **Warum?**

Berechnung von Terms

- Terms werden auf Shards berechnet
- Die Size gibt an wie viele Terms maximal berechnet werden
- Ist ein Term in einem Index oft vorhanden, in anderen aber nur sehr selten kann es dazu kommen, dass:
  - Die Dokumente in Shards mit wenig Vorkommnissen des Terms nicht über das Size-Limit kommen
  - Der Shard erachtet den Term als nicht relevant und fügt die Dokumente nicht zur Ergebnismenge hinzu

Berechnung von Terms – Shard Size

- Um das Problem zu umgehen gibt es neben der Size auch eine sogenannte „Shard-Size“
- Die Shard-Size ist die Menge an Terms die auf einem Shard berechnet werden
- Die Shard-Size wird wie folgt berechnet: size * 1.5 + 10
- Damit wird das Problem **entschärft,** es kann jedoch weiterhin zu Ungenauigkeiten kommen

Histogram

- Histogram-Aggregationen bilden Buckets auf Basis von numerischen Werten eines Felds
- Beispiel Webshop: Gruppierung von Produkten nach Preis-Ranges
- Histogram-Aggregation benötigt ein Intervall auf dessen Basis Buckets gebildet werden

Histogram

- Dokumente werden diesen Bucket hinzugefügt in dem auf- bzw. abgerundet wird
- Beispiel:
  - Intervall: 50
  - Produkt 1: 60€, Produkt 2: 90€
  - Buckets:
    - 50: [Produkt 1, ...]
    - 100: [Produkt 2, ...]

Histogram

```
POST /sales/_search?size=0
{
  "aggs": {
    "prices": {
      "histogram": {
        "field": "price",
        "interval": 50
      }
    }
  }
}
```

```
{
  ...
  "aggregations": {
    "prices": {
      "buckets": [
        {
          "key": 0.0,
          "doc_count": 1
        },
        {
          "key": 50.0,
          "doc_count": 1
        },
        {
          "key": 100.0,
          "doc_count": 0
        },
        {
          "key": 150.0,
          "doc_count": 2
        },
        {
          "key": 200.0,
          "doc_count": 3
        }
      ]
    }
  }
}
```

Date Histogram

- Gleiche Funktion wie Histogram nur auf Basis von Daten
- Auch das Date Histogram rundet: bucket_key = Math.floor(value / interval) * interval
- Interval kann als Zeiteinheit dargestellt werden

Date Histogram

```
POST /sales/_search?size=0
{
  "aggs": {
    "sales_over_time": {
      "date_histogram": {
        "field": "date",
        "calendar_interval": "month"
      }
    }
  }
}
```

Range

- Bucketing nach definierten Ranges
- Alle Dokumente werden dem definierten Intervall zugeordnet
- Keine Rundung sondern eine genaue Zuordnung

Range Aggregation

```
{
  ...
  "aggregations": {
    "price_ranges": {
      "buckets": [
        {
          "key": "*-100.0",
          "to": 100.0,
          "doc_count": 2
        },
        {
          "key": "100.0-200.0",
          "from": 100.0,
          "to": 200.0,
          "doc_count": 2
        },
        {
          "key": "200.0-*",
          "from": 200.0,
          "doc_count": 3
        }
      ]
    }
  }
}
```

```
GET sales/_search
{
  "aggs": {
    "price_ranges": {
      "range": {
        "field": "price",
        "ranges": [
          { "to": 100.0 },
          { "from": 100.0, "to": 200.0 },
          { "from": 200.0 }
        ]
      }
    }
  }
}
```

Date Range

```
POST /sales/_search?size=0
{
  "aggs": {
    "range": {
      "date_range": {
        "field": "date",
        "format": "MM-yyyy",
        "ranges": [
          { "to": "now-10M/M" },
          { "from": "now-10M/M" }
        ]
      }
    }
  }
}
```

Categorize Text

- Bei semistrukturiertem Text kann eine Kategorisierung auf Basis von Text sinnvoll sein
- Beispiel:
  - Applikationen loggen abhängig von events bspw. User-Login, HTTP Request inbound, HTTP Request Outgoing, Nutzer Registrierung, etc.
  - Manchmal besitzen die Dokumente kein „Typ“ Feld auf dessen Basis eine Aggregation über die verschiedenen Typen von Events passieren können
- Elasticsearch bietet dafür die „Categorize Text“-Aggregation, die Felder auf Basis der ersten 100 Keywords in einem Textfeld kategorisiert

Categorize Text

- Achtung: Kategorisierung basiert auf übereinstimmenden Token (Worten), keine inhaltliche Kategorisierung
- Funktion ansonsten weitestgehend wie Terms-Aggregation
- Analyzer **oder** sog. Categorization-Filter sind einstellbar
  - Categorization-Filter sind eine Menge an Regex, die Token filtern, die das Ergebnis verfälschen

Metric Aggregation

Metric Aggregations

- Aggregationen die einen oder mehrere Werte aus einer Menge an Dokumenten berechnen
- Basieren in der Regel auf den Daten aus den Feldern der Dokumente
- Numerische Aggregationen kommen in zwei Formen vor
  - „*Single value numeric aggregation*“ – Aggregation erzeugt einen numerischen Wert (z.B. average)
  - „*multi-value numeric metrics aggregation*“ – Aggregation erzeugt mehrere numerische Werte (z.B. stats)

Avg

- Berechnet den Durchschnitt auf Basis eines Felds
- Kann auf ein Set an Dokumenten angewendet werden oder auf Buckets (zb. Histogram oder Term)
- In Elastic gibt es sogenannte „*Histogram Fields*“ diese können ebenfalls mit Hilfe der Avg-Aggregation verarbeitet werden

Avg

```
POST /exams/_search?size=0
{
  "aggs": {
    "avg_grade": { "avg": { "field": "grade" } }
  }
}
```

Avg

```
PUT metrics_index/_doc/2
{
  "network.name" : "net-2",
  "latency_histo" : {
      "values" :  [0.1, 0.2, 0.3, 0.4, 0.5], 1
      "counts" : [8, 17, 8, 7, 6] 2
   }
}

POST /metrics_index/_search?size=0
{
  "aggs": {
    "avg_latency":
      { "avg": { "field": "latency_histo" }
    }
  }
}
```

Max / Min

- Die Max bzw. die Min Aggregation gibt das Maximum bzw das Minimum eines Felds in einer Menge an Dokumenten zurück
- Auch Max / Min unterstützen Histogramm-Felder

Max / Min

```
{
  ...
  "aggregations": {
      "max_price": {
          "value": 200.0
      }
  }
}
```

```
POST /sales/_search?size=0
{
  "aggs": {
    "max_price": { "max": { "field": "price" } }
  }
}
```

Percentils

- Berechnet die statistischer Verteilung der Werte eines Felds in einer Menge von Dokumenten
- Die Aggregation weißt den definierten Percentilen einen konkreten, numerischen Wert zu
- Sehr gut zur Suche von Anomalien und Outlier
- Percentils ist eine multi-value Metrik

Percentils

```
{
  ...

 "aggregations": {
    "load_time_outlier": {
      "values": {
        "1.0": 10.0,
        "5.0": 30.0,
        "25.0": 170.0,
        "50.0": 445.0,
        "75.0": 720.0,
        "95.0": 940.0,
        "99.0": 980.0
      }
    }
  }
}
```

```
GET latency/_search
{
  "size": 0,
  "aggs": {
    "load_time_outlier": {
      "percentiles": {
        "field": "load_time" (1)
      }
    }
  }
}
```

Stats

- Multi-Value Aggregation die eine Reihe von Statistiken für ein Feld in einem Set von Dokumenten berechnet
- Enthält: min, max, sum, count & average

Stats

```
{
  ...

  "aggregations": {
    "grades_stats": {
      "count": 2,
      "min": 50.0,
      "max": 100.0,
      "avg": 75.0,
      "sum": 150.0
    }
  }
}
```

```
POST /exams/_search?size=0
{
  "aggs": {
    "grades_stats": { "stats": { "field": "grade" } }
  }
}
```

Pipeline Aggregationen

Pipeline Aggregation

- Pipeline Aggregationen arbeiten auf dem Output anderer Aggregationen und nicht auf einer Dokumentenmenge
- Nicht zu verwechseln mit genesteten Aggregationen
- Häufig genutzt um Aggregationen über Buckets hinweg zu machen
- Zwei Arten:
  - Parent: Bekommt den Output der von übergeordnete Aggregationen
  - Sibling: Bekommt den Output einer Aggregation auf der gleichen Ebene

Pipeline Aggregation – Zugriff auf Werte

- Pipeline Aggregationen besitzen einen Parameter, der sich ähnlich wie ein Feld verhält
- Dadurch kann auf den Output anderer Aggregationen zugegriffen werden
- Dieser parameter nennt sich buckets_path und besitzt eine eigene Zugriffssyntax

Pipeline Aggregation – Zugriff auf Werte

```
AGG_SEPARATOR       =  `>` ;
METRIC_SEPARATOR    =  `.` ;
AGG_NAME            =  <the name of the aggregation> ;
METRIC              =  <the name of the metric (in case of multi-value metrics aggregation)> ;
MULTIBUCKET_KEY     =  `[<KEY_NAME>]`
PATH                =  <AGG_NAME><MULTIBUCKET_KEY>? (<AGG_SEPARATOR>, <AGG_NAME> )* ( <METRIC_SEPARATOR>, <METRIC> ) ;
```

Pipeline Aggregation – Zugriff auf Werte

```
POST /_search
{
  "aggs": {
    "my_date_histo": {
      "date_histogram": {
        "field": "timestamp",
        "calendar_interval": "day"
      },
      "aggs": {
        "the_sum": {
          "sum": { "field": "lemmings" }
        },
        "the_deriv": {
          "derivative": { "buckets_path": "the_sum" }
        }
      }
    }
  }
}
```

Pipeline Aggregation – Zugriff auf Werte

```
POST /_search
{
  "aggs": {
    "sales_per_month": {
      "date_histogram": {
        "field": "date",
        "calendar_interval": "month"
      },
      "aggs": {
        "sales": {
          "sum": {
            "field": "price"
          }
        }
      }
    },
    "max_monthly_sales": {
      "max_bucket": {
        "buckets_path": "sales_per_month>sales"
      }
    }
  }
}
```

Average Bucket

- Berechnung des Average über alle Buckets einer Bucket Aggregation
- **Eine Sibling-Aggregation**: Steht also auf der selben ebene wie die Bucket-Aggregation
- Muss zwingend nach einer Bucket Aggregation kommen
- Analog dazu gibt es auch min/max/sum –Bucket Aggregationen

Average Bucket

```
POST _search
{
  "size": 0,
  "aggs": {
    "sales_per_month": {
      "date_histogram": {
        "field": "date",
        "calendar_interval": "month"
      },
      "aggs": {
        "sales": {
          "sum": {
            "field": "price"
          }
        }
      }
    },
    "avg_monthly_sales": {
// tag::avg-bucket-agg-syntax[]                    (1)
      "avg_bucket": {
        "buckets_path": "sales_per_month>sales",
        "gap_policy": "skip",
        "format": "#,##0.00;(#,##0.00)"
      }
// end::avg-bucket-agg-syntax[]                    (2)
    }
  }
}
```

Bucket Selector

- Kann Buckets mit Hilfe eines Scripts was einen Boolean-Wert emittiert aus der Bucket-Aggregation filtern
- **Eine Parent-Aggregation**: Ist unter Bucket-Aggregation genested

Bucket Selector

```
POST /sales/_search
{
  "size": 0,
  "aggs": {
    "sales_per_month": {
      "date_histogram": {
        "field": "date",
        "calendar_interval": "month"
      },
      "aggs": {
        "total_sales": {
          "sum": {
            "field": "price"
          }
        },
        "sales_bucket_filter": {
          "bucket_selector": {
            "buckets_path": {
              "totalSales": "total_sales"
            },
            "script": "params.totalSales > 200"
          }
        }
      }
    }
  }
}
```

Caching

- Elasticsearch erlaubt Caching auf **Shard–Level**
- Neben „normalen“ Suchanfragen werden auch Aggregations gecached
- Hier ist der “size“-Parameter wichtig: Dokumente aus der Ergebnismenge des Requests zu exkludieren, verringert den Footprint des Caches