elastic

엘라스틱서치
요청 몸체 질의와 정렬

순천향대학교 컴퓨터공학과
이 상 정

학습 내용

1. 요청 몸체 질의
   - Query DSL
   - 결합 항목 질의
   - 질의와 필터
2. 정렬과 연관성
   - 정렬, 다단계 정렬, 다중 값 정렬
   - 연관성, 점수

1. 요청 몸체 질의 (Request Body Query)

요청 몸체 질의 (Request Body Query)

- URI 질의는 간편하지만 강력한 검색 능력 활용을 위해서는 요청 몸체 질의 사용
  - 리턴 되는 결과에 하이라이트(highlight) 표시
  - 결과에 대한 집계 분석(aggregate analysis)
  - ……
- 요청 몸체 질의에서는 질의 파라미터들이 질의 문자열이 아닌 HTTP 요청 몸체로 전달
  - 일반적으로 Query DSL을 사용하여 검색

빈 검색 (Empty Search)

- 인덱스들 안의 모든 다큐먼트들을 검색 예

```
GET /_search
{}
```

- 특정 인덱스들에 특정 타입들에 대한 검색 예

```
GET /index_2014*/type1,type2/_search
{}
```

- 출력 페이지 범위 지정 예

```
GET /_search
{
  "from": 30,
  "size": 10
}
```

- POST 메서드를 사용도 가능

```
POST /_search
{
  "from": 30,
  "size": 10
}
```


Query DSL

- Query DSL (Domain Specific Language)
  - JSON 형식의 엘라스틱서치의 질의 언어
  - query 파라미터로 질의를 전달

```
GET /_search
{
    "query": YOUR_QUERY_HERE
}
```

  - match_all 질의 항목(query clause) 을 사용한 빈 검색

```
GET /_search
{
    "query": {
        "match_all": {}
    }
}
```

질의 유형 구조

- 질의 항목(Query Clause) 구조에 질의 값, 특정 필드의 질의 값 등을 기술

```
{
    QUERY_NAME: {
        ARGUMENT: VALUE,
        ARGUMENT: VALUE,...
    }
}
```

```
{
    QUERY_NAME: {
        FIELD_NAME: {
            ARGUMENT: VALUE,
            ARGUMENT: VALUE,...
        }
    }
}
```

- tweet 필드에 elasticsearch를 검색하는 match 질의 항목 예

```
GET /_search
{
    "query": {
        "match": {
            "tweet": "elasticsearch"
        }
    }
}
```


질의 유형 구조 실행 예

```
$ curl -XGET 'localhost:9200/_search?pretty' -H 'Content-Type: application/json' -d'
> {
>     "query": {
>         "match": {
>             "tweet": "elasticsearch"
>         }
>     }
> }
> '
{
  "took" : 52,
  "timed_out" : false,
  "_shards" : {
    "total" : 27,
    "successful" : 27,
    "failed" : 0
  },
  "hits" : {
    "total" : 3,
    "max_score" : 0.6395861,
    "hits" : [
      {
        "_index" : "us",
        "_type" : "tweet",
        "_id" : "6",
        "_score" : 0.6395861,
        "_source" : {
          "date" : "2014-09-16",
          "name" : "John Smith",
          "tweet" : "The Elasticsearch API is really easy to use",
          "user_id" : 1
        }
      },
      {
        "_index" : "us",
```

다 수의 질의 항목들의 결합 (Combining Multiple Clauses)

- 여러 질의 항목들을 결합하여 복잡한 질의를 구성

- leaf 항목
  - match 질의 등과 같이 질의 문자열과 필드의 값을 비교
- 결합 항목 (compound clause)
  - 다른 질의 항목들을 결합
  - bool 결합 항목: 아래 조건 항목을 사용하여 결합
    - must 조건: 반드시 만족해야 하는 조건, AND 조건
    - must_not 조건: 반드시 만족하지 말아야 하는 조건, NOT 조건
    - shoud 조건: 반드시 만족해야 할 필요성은 없지만 만족하면 높은 점수
    - filter 조건: 바이너리 조건 검색 (Yes/No)

결합 항목 질의 예 (1)

- 다음은 다음 조건을 만족하는 bool 질의 예
  - tweet 필드에 elasticsearch 문자열이 반드시 있어야 하고,
  - name 필드에 mary는 없어야만 하고,
  - tweet 필드에 full text가 있으면서,
  - age 필드 값이 30 보다 커야 함

```
{
    "bool": {
        "must":     { "match": { "tweet": "elasticsearch" }},
        "must_not": { "match": { "name":  "mary" }},
        "should":   { "match": { "tweet": "full text" }},
        "filter":   { "range": { "age" : { "gt" : 30 }} }
    }
}
```

결합 항목 질의 예 (2)

```
{
    "bool": {
        "must": { "match":   { "email": "business opportunity" }},
        "should": [
            { "match":       { "starred": true }},
            { "bool": {
                "must":      { "match": { "folder": "inbox" }},
                "must_not":  { "match": { "spam": true }}
            }}
        ],
        "minimum_should_match": 1
    }
}
```


결합 항목 질의 실행 예 (1)

```
$ cat compound.dsl
{
    "query": {
      "bool": {
        "must":     { "match": { "tweet": "elasticsearch" }},
        "must_not": { "match": { "name":  "mary" }},
        "should":   { "match": { "tweet": "full text" }},
        "filter":   { "range": { "age" : { "gt" : 30 }} }
      }
    }
}
$
$ curl -XGET 'localhost:9200/_search?pretty' -d @compound.dsl
{
  "took" : 77,
  "timed_out" : false,
  "_shards" : {
    "total" : 27,
    "successful" : 27,
    "failed" : 0
  },
  "hits" : {
    "total" : 0,
    "max_score" : null,
    "hits" : [ ]
  }
}
```

결합 항목 질의 실행 예 (2)

```
$ cat compound1.dsl
{
    "query": {
      "bool": {
        "must":     { "match": { "tweet": "elasticsearch" }},
        "must_not": { "match": { "name":  "mary" }},
        "should":   { "match": { "tweet": "full text" }},
        "filter":   { "range": { "date": { "gt" : "2014-09-20" }} }
      }
    }
}
$
$ curl -XGET 'localhost:9200/gb,us/_search?pretty' -d @compound1.dsl
{
  "took" : 50,
  "timed_out" : false,
  "_shards" : {
    "total" : 10,
    "successful" : 10,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 0.15716305,
    "hits" : [
      {
        "_index" : "us",
        "_type" : "tweet",
        "_id" : "12",
        "_score" : 0.15716305,
        "_source" : {
          "date" : "2014-09-22",
          "name" : "John Smith",
          "tweet" : "Elasticsearch and I have left the honeymoon stage, and I still love her.",
          "user_id" : 1
        }
      }
    ]
  }
}
```


질의와 필터 (Queries and Filters)

- 엘라스틱서치의 질의 기능은 질의(query)와 필터링(filtering)으로 구분
  - 질의는 전문(full-text) 검색에 사용되며, 연관성 관련 점수를 계산
    - 아래 예와 같은 조건의 다큐먼트들을 검색
      - "full text search" 단어들에 가장 일치
      - "run" 단어를 포함, "runs" "running“ "sprint" 도 해당할 수 있음
      - "quick", "brown", "fox" 단어들을 포함
  - 필터링은 연관성 점수를 계산하지 않고, Yes/No와 같은 이진 조건 검색
    - 아래 예와 같이 다큐먼트가 이진 조건 만족여부 검색
      - "created" date가 2014-2016 범위에 있는가?
      - "staus" 필드가 텀 "published" 를 포함하는가?
      - "lat_lon" 필드가 특정 지점의 10km 이내에 있는가?

질의와 필터 비교

쿼리와 필터의 비교

	쿼리 (Query)	필터 (Filter)
검색 대상 (일반적)	전문검색 (Full Text)	바이너리 구분 (Y/N)
점수 계산	O	X
캐싱	X	O
응답 속도 (상대적)	느림	빠름

주요 질의 유형 – match

❑ match 질의

- 주어진 필드에 대해 전문 또는 정확한 값의 검색을 질의
- 전문인 경우 검색 전에 질의 문자열을 분석기가 분석

```
{ "match": { "tweet": "About Search" }}
```

- 숫자, date, 부울값, not_analyzed 문자열과 같은 정확한 값의 필드는 분석하지 않고 바로 검색

```
{ "match": { "age":    26           }}
{ "match": { "date":   "2014-09-01" }}
{ "match": { "public": true         }}
{ "match": { "tag":    "full_text"  }}
```

주요 질의 유형 – multi_match, match_all, range

- multi_match 질의는 match와 유사하며 여러 필드들에 질의를 적용

```
{
    "multi_match": {
        "query":    "full text search",
        "fields":   [ "title", "body" ]
    }
}
```

- match_all 질의는 모든 다큐먼트들을 조회

```
{ "match_all": {}}
```

- range 질의는 주어진 범위에 해당하는 다큐먼트를 검색
  - 연산자: gt, gte, lt, lte

```
{
    "range": {
        "age": {
            "gte":  20,
            "lt":   30
        }
    }
}
```


multi_match 실행 예

```
$ cat multimatch.dsl
{
    "query": {
      "multi_match": {
         "query": "post location",
         "fields": ["title", "tweet"]
      }
    }
}
$
$ curl -XGET 'localhost:9200/_search?pretty' -d @multimatch.dsl
{
  "took" : 164,
  "timed_out" : false,
  "_shards" : {
    "total" : 27,
    "successful" : 27,
    "failed" : 0
  },
  "hits" : {
    "total" : 3,
    "max_score" : 0.7076487,
    "hits" : [
      {
        "_index" : "gb",
        "_type" : "tweet",
        "_id" : "9",
        "_score" : 0.7076487,
        "_source" : {
          "date" : "2014-09-19",
          "name" : "Mary Jones",
          "tweet" : "Geo-location aggregations are really cool",
          "user_id" : 2
        }
      },
      {
        "_index" : "website",
        "_type" : "blog",
        "_id" : "AVpJ-xYI8NG4cVdMNO6L",
        "_score" : 0.18232156,
        "_source" : {
          "title" : "My second blog post"
        }
      },
      {
        "_index" : "website",
        "_type" : "blog",
        "_id" : "AVpfk3OM8NG4cVdMNWtX",
        "_score" : 0.18232156,
        "_source" : {
          "title" : "My second blog post"
        }
      }
    ]
  }
}
$
```

주요 질의 유형 – term, terms

- term 질의
  - 숫자, date, 부울값, not_analyzed 문자열과 같은 정확한 값의 질의
  - 입력 필드에 대해 분석을 하지 않고 검색

```
{ "term": { "age":    26           }}
{ "term": { "date":   "2014-09-01" }}
{ "term": { "public": true         }}
{ "term": { "tag":    "full_text"  }}
```

- terms 질의는 term과 유사하며 여러개 값(배열)들 질의를 적용

```
{ "terms": { "tag": [ "search", "full_text", "nosql" ] }}
```


term 질의 실행 예

```
$ curl -XGET 'localhost:9200/gb,us/_search?pretty' -d '
{
  "query": {
    "term": { "date": "2014-09-15" }
  }
}'
{
  "took" : 31,
  "timed_out" : false,
  "_shards" : {
    "total" : 10,
    "successful" : 10,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "gb",
        "_type" : "tweet",
        "_id" : "5",
        "_score" : 1.0,
        "_source" : {
          "date" : "2014-09-15",
          "name" : "Mary Jones",
          "tweet" : "However did I manage before Elasticsearch?",
          "user_id" : 2
        }
      }
    ]
  }
}
```

주요 질의 유형 – exists, missing

- 다큐먼트 내에 필드의 값이 있는지 (exists), 없는지 (missing)를 확인

```
{
    "exists":   {
        "field":    "title"
    }
}
```


질의의 결합

- 여러 개의 질의가 결합 시 연관성 점수 계산
  - 하부 질의에 대해 각각 독립적으로 연관성 점수 계산
  - bool 질의가 각 독립된 점수들을 통합하여 단일 점수로 리턴
  - 예
    - title 필드가 how to make millions에 매치되고, tag가 spam이 아닌 다큐먼트 검색
    - tag가 starred 이거나 또는 2014년 이 후(또는 둘 다 만족)이면 높은 연관성 점수

```
{
    "bool": {
        "must":     { "match": { "title": "how to make millions" }},
        "must_not": { "match": { "tag":   "spam" }},
        "should": [
            { "match": { "tag": "starred" }},
            { "range": { "date": { "gte": "2014-01-01" }}}
        ]
    }
}
```

필터링 질의 추가 (1)

- 앞의 질의에서 range 질의를 filter 항목으로 이동하면 더 이상 다큐먼트의 연관성 점수에 기여하지 않음

```
{
    "bool": {
        "must":     { "match": { "title": "how to make millions" }},
        "must_not": { "match": { "tag":   "spam" }},
        "should": [
            { "match": { "tag": "starred" }}
        ],
        "filter": {
          "range": { "date": { "gte": "2014-01-01" }} ❶
        }
    }
}
```


필터링 질의 추가 (2)

- 필터링 조건에 부울 논리를 추가한 예

```
{
    "bool": {
        "must":     { "match": { "title": "how to make millions" }},
        "must_not": { "match": { "tag":   "spam" }},
        "should": [
            { "match": { "tag": "starred" }}
        ],
        "filter": {
          "bool": { ❶
              "must": [
                  { "range": { "date": { "gte": "2014-01-01" }}},
                  { "range": { "price": { "lte": 29.99 }}}
              ],
              "must_not": [
                  { "term": { "category": "ebooks" }}
              ]
          }
        }
    }
}
```

질의의 검증

- _validate-query API를 사용하여 질의의 유효성 검증

```
GET /gb/tweet/_validate/query
{
   "query": {
      "tweet" : {
         "match" : "really powerful"
      }
   }
}
```

```
{
  "valid" :         false,
  "_shards" : {
    "total" :       1,
    "successful" :  1,
    "failed" :      0
  }
}
```


에러의 이해

- explain 파라미터를 사용하여 에러의 원인을 조사

```
GET /gb/tweet/_validate/query?explain
{
   "query": {
      "tweet" : {
         "match" : "really powerful"
      }
   }
}
```

```
{
  "valid" :     false,
  "_shards" :   { ... },
  "explanations" : [ {
    "index" :   "gb",
    "valid" :   false,
    "error" :   "org.elasticsearch.index.query.QueryParsingException:
                 [gb] No query registered for [tweet]"
  } ]
}
```

질의의 이해

- explain 파라미터는 유효한 질의의 설명이 리턴
  - 아래 예에서 2개의 단일 텀으로 검색
  - gb 인덱스는 english 분석기를 적용하여 텀이 변환

```
GET /_validate/query?explain
{
   "query": {
      "match" : {
         "tweet" : "really powerful"
      }
   }
}
```

```
{
  "valid" :         true,
  "_shards" :       { ... },
  "explanations" : [ {
    "index" :       "us",
    "valid" :       true,
    "explanation" : "tweet:really tweet:powerful"
  }, {
    "index" :       "gb",
    "valid" :       true,
    "explanation" : "tweet:realli tweet:power"
  } ]
}
```


질의의 이해 실행 예

```
$ curl -XGET 'localhost:9200/gb,us/_validate/query?explain&pretty' -d'
{
   "query": {
      "match" : {
         "tweet" : "really powerful"
      }
   }
}'
{
  "valid" : true,
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "failed" : 0
  },
  "explanations" : [
    {
      "index" : "gb",
      "valid" : true,
      "explanation" : "tweet:realli tweet:power"
    },
    {
      "index" : "us",
      "valid" : true,
      "explanation" : "tweet:really tweet:powerful"
    }
  ]
}
```

2. 정렬과 연관성 (Sorting and Relevance)

정렬

- 검색된 결과는 디폴트로 연관성 점수(relevance score)로 정렬되어 리턴
  - 점수는 실수로 메타데이터 _score 로 표시
  - 디폴트 정렬은 _score의 내림차순
  - 필터를 사용하여 연관성 점수가 의미가 없는 경우에는 임의의 순서로 결과 리턴
  - sort 파라미터를 사용하여 필드 값에 따라 정렬할 수 있음

```
GET /_search
{
    "query" : {
        "bool" : {
            "filter" : { "term" : { "user_id" : 1 }}
        }
    },
    "sort": { "date": { "order": "desc" }}
}
```

필드 값 정렬 결과

```
"hits" : {
    "total" :           6,
    "max_score" :       null, ❶
    "hits" : [ {
        "_index" :      "us",
        "_type" :       "tweet",
        "_id" :         "14",
        "_score" :      null, ❷
        "_source" :     {
             "date":    "2014-09-24",
             ...
        },
        "sort" :        [ 1411516800000 ] ❸
    },
    ...
}
```

① 정렬에 사용되지 않았기 때문에 _score 값이 null

②

③ UNIX epoch 시간으로 표시
- 1970년 1월1일 이후 부터 ms 으로 표시

- 아래와 같이 간단하게 정렬되는 필드의 이름을 기술하여 정렬할 수도 있음
  - 디폴트로 올림차순으로 정렬

```
"sort": "number_of_children"
```


정렬 실행 예

```
$ curl -XGET 'localhost:9200/gb,us/_search?pretty' -d'
{
    "query" : {
        "bool" : {
            "filter" : { "term" : { "user_id" : 1 }}
        }
    },
    "sort": { "date": { "order": "desc" }}
}
{
  "took" : 46,
  "timed_out" : false,
  "_shards" : {
    "total" : 10,
    "successful" : 10,
    "failed" : 0
  },
  "hits" : {
    "total" : 6,
    "max_score" : null,
    "hits" : [
      {
        "_index" : "us",
        "_type" : "tweet",
        "_id" : "14",
        "_score" : null,
        "_source" : {
          "date" : "2014-09-24",
          "name" : "John Smith",
          "tweet" : "How many more cheesy tweets do I have to
          "user_id" : 1
        },
        "sort" : [
          1411516800000
        ]
      },
      {
        "_index" : "us",
        "_type" : "tweet",
        "_id" : "12",
        "_score" : null,
        "_source" : {
          "date" : "2014-09-22",
          "name" : "John Smith",
          "tweet" : "Elasticsearch and I have le
          "user_id" : 1
        },
        "sort" : [
          1411344000000
        ]
      },
      {
        "_index" : "us",
        "_type" : "tweet",
        "_id" : "10",
        "_score" : null,
        "_source" : {
          "date" : "2014-09-20",
          "name" : "John Smith",
          "tweet" : "Elasticsearch surely is one
          "user_id" : 1
        },
        "sort" : [
          1411171200000
        ]
      },
```

다단계 정렬 (Multilevel Sorting)

- date 필드 정렬 하고, date가 같은 경우 _score로 정렬하는 예
  - sort 파라미터에 기술된 순서대로 정렬 적용

```
GET /_search
{
    "query" : {
        "bool" : {
            "must":   { "match": { "tweet": "manage text search" }},
            "filter" : { "term" : { "user_id" : 2 }}
        }
    },
    "sort": [
        { "date":   { "order": "desc" }},
        { "_score": { "order": "desc" }}
    ]
}
```


다중 값 필드의 정렬

- 필드의 값들이 여러 개인 경우 정렬 모드를 지정
  - 모드 값: min, max, avg, sum

```
"sort": {
    "dates": {
        "order": "asc",
        "mode":  "min"
    }
}
```

다중 값 문자열 정렬 (1)

- 다중 값을 갖는 문자열 정렬 시에는 각 텀들은 분리하여 정렬할 필요가 있음
  - "fine old art" 의 경우 3개의 텀으로 분리
  - 엘라스틱서치에서는 다중값으로 정렬 시 3개의 텀을 선택 순서는 무작위
    - fine을 먼저 선택한다는 보장 없음
- 다중 값을 갖는 문자열을 정렬 시 다큐먼트를 2개의 필드로 구분
  - 검색 시에는 분석기를 사용하는 analyzed 모드
  - 정렬 시에는 not_analyzed 모드로 동작

다중 값 문자열 정렬 (2)

```
"tweet": {
    "type":     "string",
    "analyzer": "english"
}
```

- 위 필드 타입을 아래와 같이 2개의 필드로 구분

```
"tweet": { ❶
    "type":     "string",
    "analyzer": "english",
    "fields": {
        "raw": { ❷
            "type":  "string",
            "index": "not_analyzed"
        }
    }
}
```

① tweet 필드는 analyzed 전문 필드

② tweet.raw 서브 필드는 not_analyzed

다중 값 문자열 정렬 (3)

- 검색 시에는 tweet 필드를 적용하고, 정렬 시에는 tweet.raw 필드를 적용

```
GET /_search
{
    "query": {
        "match": {
            "tweet": "elasticsearch"
        }
    },
    "sort": "tweet.raw"
}
```


연관성 (Relevance)

- 연관성(relevance)은 질의 문자열과 전문 필드의 내용이 얼마나 유사한가를 계산하는 알고리즘

- 엘라스틱서치의 표준 유사성 알고리즘은 TF/IDF (term frequency/inverse document frequency)로 다음을 고려
  - 텀 빈도(term frequency)
    - 다큐먼트의 필드에서 해당하는 텀의 빈도수로 클수록 연관성 증가
  - 역 다큐먼트 빈도 (Inverse document frequency)
    - 인덱스에서 해당 텀의 빈도 수로 클수록 연관성 감소
    - 다큐먼트에 많이 존재하면 적은 텀들보다 가중치가 감소
  - 필드 길이 표준 (Field-length norm)
    - 필드의 내용의 길이가 클수록 필드 내의 단어들의 연관성이 감소
    - tilte 필드의 짧은 텀들이 content 필드의 같은 텀 보다 더 큰 가중치

연관성 점수 이해

- 연관성 점수 _score 계산 방식을 이해하기 위해 검색 시 explain 파라미터를 기술하여 디버깅

```
GET /_search?explain ❶
{
   "query"   : { "match" : { "tweet" : "honeymoon" }}
}
```

- 먼저, 응답에 일반 메타데이터를 리턴

```
{
    "_index" :      "us",
    "_type" :       "tweet",
    "_id" :         "12",
    "_score" :      0.076713204,
    "_source" :     { ... trimmed ... },
```

- 다큐먼트가 저장된 노드와 샤드 정보 응답
  - 텀과 다큐먼트는 인덱스가 아닌 샤드 단위로 계산함으로 이 정보 중요

```
    "_shard" :      1,
    "_node" :       "mzIVYCsqSWCG_M_ZffSs9Q",
```


- 연관성 점수 계산 과정 _explanation 응답

```
"_explanation": { ❶
   "description": "weight(tweet:honeymoon in 0)
                   [PerFieldSimilarity], result of:",
   "value":       0.076713204,
   "details": [
      {
         "description": "fieldWeight in 0, product of:",
         "value":       0.076713204,
         "details": [
            { ❷
               "description": "tf(freq=1.0), with freq of:",
               "value":       1,
               "details": [
                  {
                     "description": "termFreq=1.0",
                     "value":       1
                  }
               ]
            },
            { ❸
               "description": "idf(docFreq=1, maxDocs=1)",
               "value":       0.30685282
            },
            { ❹
               "description": "fieldNorm(doc=0)",
               "value":        0.25,
            }
         ]
      }
   ]
}
```

① honeymoon의 점수 계산 요약

② 텀 빈도 수
-> 이 다큐먼트에서 tweet 필드의 honeymoon 텀 빈도 수

③ 역 다큐먼트 빈도 수
-> 인덱스 내의 모든 다큐먼트에서 tweet 필드의 honeymoon 텀 빈도 수

④ 필드 길이 표준
-> 이 다큐먼트 내에서 tweet 필드의 길이, 필드의 길이가 길수록 이 값이 작아짐

다큐먼트 매치 여부 이해

- explain API를 사용하여 매치된 다큐먼트와 매치되지 않는 다큐먼트에 대한 디버깅

```
GET /us/tweet/12/_explain
{
   "query" : {
      "bool" : {
         "filter" : { "term" :  { "user_id" : 2           }},
         "must" :  { "match" : { "tweet" :   "honeymoon" }}
      }
   }
}
```

```
"failure to match filter: cache(user_id:[2 TO 2])"
```

  - user_id 필터링으로 매치되지 않았음을 표시

과 제

- 자신만의 데이터에 대한 요청 몸체 질의, 정렬과 연관성 적용 및 실행 예

참고 자료

- Elasticsearch: The Definitive Guide, Getting Started
  - Full-Body Search
    - https://www.elastic.co/guide/en/elasticsearch/guide/current/full-body-search.html
  - Sorting and Relevance
    - https://www.elastic.co/guide/en/elasticsearch/guide/current/sorting.html