elastic

엘라스틱서치
검색 기본과 매핑

순천향대학교 컴퓨터공학과
이 상 정

학습 내용

1. 기본 검색 툴
   - 빈 검색, 다중 인덱스 및 타입 검색, URI 질의
2. 매핑과 분석
   - 매핑 자료형, 역색인, 분석과 분석기, 동적/정적 매핑, 객체 자료형

1. 기본 검색 툴

엘라스틱서치 검색 (Searching)

- 다큐먼트의 모든 필드는 색인되어 질의(query, 쿼리)될 수 있음
  - 검색 대상은 구조화된 필드나 전문(full-text) 또는 이들 둘의 조합
- 검색과 관련된 주요 주제
  - 매핑(mapping): 각 필드의 데이터 접근 및 처리 방법
  - 분석(analysis): 전문이 검색 가능하도록 처리하는 방법
  - Query DSL: 엘라스틱서치에서 사용하는 질의 언어

빈 검색 (Empty Search) (1)

- 어떤 질의도 기술하지 않으면 단순히 클러스터의 모든 인덱스를 리턴

```
GET /_search
```


```
{
   "hits" : {
      "total" :       14,
      "hits" : [
        {
          "_index":   "us",
          "_type":    "tweet",
          "_id":      "7",
          "_score":   1,
          "_source": {
             "date":    "2014-09-17",
             "name":    "John Smith",
             "tweet":   "The Query DSL is really powerful and flexible",
             "user_id": 2
          }
        },
         ... 9 RESULTS REMOVED ...
      ],
      "max_score" :   1
   },
   "took" :           4,
   "_shards" : {
      "failed" :      0,
      "successful" :  10,
      "total" :       10
   },
   "timed_out" :      false
}
```


빈 검색 (1)

- hits 섹션
  - total 필드: 질의와 매치되는 다큐먼트들의 수
  - hits 필드 배열 : 매치된 10개의 다큐먼트 표시
    - 다큐먼트 메타 데이터: _index, _type, _id
    - _score는 질의에 매치된 정도를 보여 주는 연관성 점수(relevance score)로 빈 질의에는 최대 점수(max_score) 1
    - _source
- took: 검색 실행 시간을 ms로 표시
- _shards
  - 질의에 포함된 전체 샤드들의 수
  - 검색 성공(successful), 실패(failed) 샤드 수
  - timeout: 만료 시간, 디폴트로 만료 시간은 없고 파라미터로 지정
    - 만료 시간을 기술해도 응답만 정해진 시간 내로 하고, 질의 수행은 계속 진행

```
GET /_search?timeout=10ms
```

다중 인덱스 및 타입 검색 (Multi-index, Multi-type Searching)

- URI(URL)에 인덱스들과 타입들을 기술하여 다중 인덱스 및 타입 검색
  - /_search # 모든 인덱스 내의 모든 타입을 검색
  - /gb/_search # gb 인덱스 내의 모든 타입을 검색
  - /gb,us/_search # gb,us 인덱스 내의 모든 타입을 검색
  - /g*,u*/_search # g,u로 시작하는 인덱스 내의 모든 타입을 검색
  - /gb/user/_search # 인덱스 gb, 타입 user 내의 모든 다큐먼트 조회
  - /gb,us/user,tweet/_search # gb, us 인덱스 내의 user, tweet 타입 내 조회
  - /_all/user,tweet/_search # 모든 인덱스 내의 user와 tweet 타입 내 조회

빈 검색 실행 예 – 테스트 데이터 색인

- 실습을 위해 다음 테스트 데이터를 사용
  - gb, us 인덱스와 user, tweet 타입
  - 데이터 다운로드
    - https://gist.github.com/clintongormley/8579281
    - 강의 홈 페이지 load_test_data.sh
  - 테스트 데이터 색인

```
$ vi load_test_data.sh
$
$ sh load_test_data.sh
{
  "_index" : "us",
  "_type" : "user",
  "_id" : "1",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "created" : true
}
{
  "_index" : "gb",
```

빈 검색 실행 예 – 테스트데이터

```
$ cat load_test_data.sh
curl -XPUT 'http://localhost:9200/us/user/1?pretty=1' -d '
{
    "email" : "john@smith.com",
    "name" : "John Smith",
    "username" : "@john"
}
'

curl -XPUT 'http://localhost:9200/gb/user/2?pretty=1' -d '
{
    "email" : "mary@jones.com",
    "name" : "Mary Jones",
    "username" : "@mary"
}
'

curl -XPUT 'http://localhost:9200/gb/tweet/3?pretty=1' -d '
{
    "date" : "2014-09-13",
    "name" : "Mary Jones",
    "tweet" : "Elasticsearch means full text search has never been so easy",
    "user_id" : 2
}
'

curl -XPUT 'http://localhost:9200/us/tweet/4?pretty=1' -d '
{
    "date" : "2014-09-14",
    "name" : "John Smith",
    "tweet" : "@mary it is not just text, it does everything",
    "user_id" : 1
}
'

curl -XPUT 'http://localhost:9200/gb/tweet/5?pretty=1' -d '
{
    "date" : "2014-09-15",
```


빈 검색 실행 예

```
$
$ curl -XGET 'localhost:9200/us/_search?pretty'
{
  "took" : 18,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 7,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "us",
        "_type" : "tweet",
        "_id" : "14",
        "_score" : 1.0,
        "_source" : {
          "date" : "2014-09-24",
          "name" : "John Smith",
          "tweet" : "How many more cheesy tweets do I have to write?",
          "user_id" : 1
        }
      },
      {
        "_index" : "us",
        "_type" : "tweet",
        "_id" : "8",
        "_score" : 1.0,
        "_source" : {
          "date" : "2014-09-18",
```

출력 페이지 범위 지정 (Pagination)

- from, size 파라미터로 출력 페이지 결과 범위지정
  - size: 출력되는 결과(다큐먼트)의 수 지정, 디폴트는 10
  - from: 출력 시작 결과(다큐먼트) 지정, 디폴트 0

```
GET /_search?size=5
GET /_search?size=5&from=5
GET /_search?size=5&from=10
```

  - URI 파라미터는 & 로 구분

URI 질의 – q 파라미터

- 질의 명령어의 2가지 방식
  - q 파라미터를 이용한 URI 질의
    - "Lite" 질의 문자열 버전
  - HTTP 요청 몸체(request body) 질의
    - DSL(Domain Specific Language)

- URI 질의
  - _search ?q= 값 또는 필드명:질의어을 하나 또는 여러 개를 입력하고 검색

```
GET /_all/tweet/_search?q=tweet:elasticsearch
```

  - tweet 필드에 elasticsearch 값을 갖는 모든 다큐먼트 검색

URI 질의 – URI 파라미터 구분 '&'

- & 사용하여 파라미터 구분

- 파라미터 예
  - pretty: 출력 결과를 정돈
  - _source: 다큐먼트 내용 표시 여부 지정, false이면 메타 데이터만 출력
  - 예
    q = name:mary&_source=false&pretty

URI 질의 실행 예

```
$ curl -XGET 'localhost:9200/_all/tweet/_search?q=tweet:elasticsearch&pretty'
{
  "took" : 218,
  "timed_out" : false,
  "_shards" : {
    "total" : 41,
    "successful" : 41,
    "failed" : 0
  },
  "hits" : {
    "total" : 7,
    "max_score" : 0.7081689,
    "hits" : [
      {
        "_index" : "gb",
        "_type" : "tweet",
        "_id" : "13",
        "_score" : 0.7081689,
        "_source" : {
          "date" : "2014-09-23",
          "name" : "Mary Jones",
          "tweet" : "So yes, I am an Elasticsearch fanboy",
          "user_id" : 2
        }
      },
      {
        "_index" : "gb",
        "_type" : "tweet",
        "_id" : "5",
```

URI 질의 – 조건 나열 '+'

- + 를 사용하여 질의 조건을 나열

- 예
  - q=name:mary+tweet:DSL
    - 두 필드 조건을 만족하는 다큐먼트 모두 검색
  - q=date:>2014-09-20+tweet:DSL&pretty
    - date 필드가 2014년 9월20일 이 후 이거나
    - tweet 필드에 DSL을 포함하는 다큐먼트 모두 검색


```
$ curl -XGET 'localhost:9200/gb,us/_search?q=date:>2014-09-20+tweet:DSL&pretty'
{
  "took" : 12,
  "timed_out" : false,
  "_shards" : {
    "total" : 10,
    "successful" : 10,
    "failed" : 0
  },
  "hits" : {
    "total" : 5,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "us",
        "_type" : "tweet",
        "_id" : "14",
        "_score" : 1.0,
        "_source" : {
          "date" : "2014-09-24",
          "name" : "John Smith",
          "tweet" : "How many more cheesy tweets do I have to write?",
          "user_id" : 1
        }
      ...........

      {
        "_index" : "gb",
        "_type" : "tweet",
        "_id" : "7",
        "_score" : 0.6030227,
        "_source" : {
          "date" : "2014-09-17",
          "name" : "Mary Jones",
          "tweet" : "The Query DSL is really powerful and flexible",
          "user_id" : 2
        }
      }
    ]
  }
}
$
```

URI 질의 – 내장 '()'

- ()를 사용하여 여러 값들 포함
- 예
  - q=tweet:(API, DSL)
    - tweet 필드에 API나 DSL을 포함하는 모든 다큐먼트 검색
  - q=name:(mary, john)+date:>2014-09-20+(aggregations, geo)
    - 검색 조건
      - name 필드의 값이 mary 또는 john
      - date 필드 값이 2014-09-10 보다 큼
      - aggregations과 geo 단어를 포함하는 모든 필드


```
$ curl -XGET 'localhost:9200/gb,us/tweet/_search?q=tweet:(API,DSL)&pretty'
{
  "took" : 49,
  "timed_out" : false,
  "_shards" : {
    "total" : 10,
    "successful" : 10,
    "failed" : 0
  },
  "hits" : {
    "total" : 2,
    "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" : "gb",
        "_type" : "tweet",
        "_id" : "7",
        "_score" : 0.6030227,
        "_source" : {
          "date" : "2014-09-17",
          "name" : "Mary Jones",
          "tweet" : "The Query DSL is really powerful and flexible",
          "user_id" : 2
        }
      }
    ]
  }
}
```

_all 필드

- 질의에서 검색 필드를 명시하지 않으면 엘라스틱서치는 _all 필드 라는 특수 필드 상에서 검색
  - _all 필드는 모든 필드들을 연결한 문자열

```
{
    "tweet":    "However did I manage before Elasticsearch?",
    "date":     "2014-09-14",
    "name":     "Mary Jones",
    "user_id":  1
}
```

=> `"However did I manage before Elasticsearch? 2014-09-14 Mary Jones 1"`

  - 다음 질의를 하면 다큐먼트의 필드 중 하나에 mary가 포함되면 검색

```
GET /_search?q=mary
```


2. 매핑과 분석

매핑 (Mapping) – 자료형

- 매핑은 데이터의 저장 형태(자료형 속성)의 정보를 기술

- 아래 질의 예에서 서로 다른 결과가 나오는 이유?
  - 인덱스에 12개의 2014년 트윗이 있다고 가정
  - _all 필드 검색 시에는 2014 string 형 검색으로 2014년 모두 리턴
  - 정확한 date 형 기술 시에만 올바른 검색

```
GET /_search?q=2014              # 12 results
GET /_search?q=2014-09-15        # 12 results !
GET /_search?q=date:2014-09-15   # 1  result
GET /_search?q=date:2014         # 0  results !
```


tweet 타입 매핑 확인 예

```
GET /gb/_mapping/tweet
```

```
{
   "gb": {
      "mappings": {
         "tweet": {
            "properties": {
               "date": {
                  "type": "date",
                  "format": "strict_date_optional_time||epoch_millis"
               },
               "name": {
                  "type": "string"
               },
               "tweet": {
                  "type": "string"
               },
               "user_id": {
                  "type": "long"
               }
            }
         }
      }
   }
}
```

매핑 타입 확인 실행 예

```
$ curl -XGET 'localhost:9200/gb/_mapping/tweet?pretty'
{
  "gb" : {
    "mappings" : {
      "tweet" : {
        "properties" : {
          "date" : {
            "type" : "date"
          },
          "name" : {
            "type" : "text",
            "fields" : {
              "keyword" : {
                "type" : "keyword",
                "ignore_above" : 256
              }
            }
          },
          "tweet" : {
            "type" : "text",
            "fields" : {
              "keyword" : {
                "type" : "keyword",
                "ignore_above" : 256
              }
            }
          },
          "user_id" : {
            "type" : "long"
          }
        }
      }
    }
  }
}
$
```

멀티필드

키워드 자료형

256개 문자 이상은 무시

- name, tweet 필드는 멀티필드 (multi-fields)로 지정
  - 한 필드에 여러 자료형을 지정
  - 전문 검색인 경우 text로 매핑
  - 정렬, 어그리게이션인 경우 keyword 자료형
    - 메일주소, 호스트이름, 상태코드 , zip 코드, 태그, ...

정확한 값과 전문 (Exact Values vs. Full Text)

- 2가지 종류의 엘라스틱서치의 데이터

- 정확한 값 (exact value)
  - 자료형을 갖는 구조적인 데이터
  - 날짜, 사용자 ID 등
  - 정확한 값을 사용한 질의의 결과는 일치 여부로 단순
- 전문 (full text)
  - 텍스트 데이터와 같은 비구조적인 데이터
  - 전문 검색의 결과는 정확한 일치 여부 보다 연관성 정도
  - 엘라스틱서치는 먼저 전문 검색 필드를 분석하여 역파일 색인 (inverted index)를 구축

역 색인 (Inverted Index) (1)

- 엘라스틱서치는 빠른 전문 검색을 위해 역 색인을 사용
  - 역파일 색인에는 다큐먼트의 모든 워드(텀 term 또는 토큰 token)들이 나열
  - 각 워드가 다큐먼트들에 존재 여부 표시
  - content 필드에 다음 내용을 갖는 두 다큐먼트의 역파일 색인 예
    1. The quick brown fox jumped over the lazy dog
    2. Quick brown foxes leap over lazy dogs in summer
  - quick brown 검색 예
    - 두 다큐먼트에 매치
    - 텀들의 수를 카운트하는 유사성 알고리즘 수행하면 첫 번째 다큐먼트가 두 번째 보다 더 큰 연관성을 가짐

Term	Doc_1	Doc_2
brown	X	X
quick	X	
Total	2	1

Term	Doc_1	Doc_2
Quick		X
The	X	
brown	X	X
dog	X	
dogs		X
fox	X	
foxes		X
in		X
jumped	X	
lazy	X	X
leap		X
over	X	X
quick	X	
summer		X
the	X	

역파일 색인 (2)

- 앞의 역파일 색인은 다음을 구분 못하는 문제점
  - +Quick +fox 질의 시 매치되는 다큐먼트 찾지 못함
    - 워드 앞의 + 연산자는 다큐먼트에 반드시 있어야 매치됨
    - Quick, quick가 같은 워드임에도 텀이 서로 분리
    - fox와 foxes, dog과 dogs 는 서로 같은 워드에서 분리된 유사한 텀
    - jumped와 leap은 같은 워드는 아니지만 비슷한 의미를 가짐
  - 만약 텀을 다음과 같이 정규화(normalization)할 수 있으면 매치되는 다큐먼트 검색
    - Quick을 quick, foxes를 fox, dogs를 dog
    - jumped와 leap은 단일 텀 jump로 인덱스
- 이러한 토큰화(tokenization), 정규화(normalization)과정을 분석(analysis)이라 함

Term	Doc_1	Doc_2
brown	X	X
dog	X	X
fox	X	X
in		X
jump	X	X
lazy	X	X
over	X	X
quick	X	X
summer		X
the	X	X

분석과 분석기 (Analysis and Analyzer)

- 분석(analysis) 과정
  - 텍스트의 블록을 역파일 색인에 적합한 개별 텀으로 토큰화 (tokenizing)
  - 검색 능력 향상을 위해 텀들을 표준 형식으로 정규화(normalizing)
- 분석기(analyzer)는 다음 3개의 함수들을 순서대로 실행
  - 문자 필터 (character filter)
    - 토큰화 전에 문자열을 정돈
    - 예를 들어 HTML을 제거, &를 and로 변환 등등
  - 토크나이저 (tokenizer)
    - 정돈된 문자열을 개별 텀들로 토큰화
    - 공백이나 구두점(punctuation,문장부호),마침표 등을 만나면 텀으로 분리
  - 토큰 필터 (token filter)
    - 텀들을 정규화된 텀으로 변환
    - 예를들어 Quck을 quick으로 변환, a and the 등 제거, 텀들을 추가 (jump,leap 동의어)

내장 분석기 (Built-in Analyzer) (1)

- 엘라스틱서치는 많은 별도의 분석기들을 제공
  - 여기서는 다음 예에 대해 미리 설치된 내장 분석기 변환 예 소개

```
"Set the shape to semi-transparent by calling set_trans(5)"
```

- 표준 분석기(standard analyzer)
  - 엘라스틱서치가 사용하는 디폴트 분석기
  - 유니코드 기준으로 워드를 분리, 문장 부호 제거, 모든 텀을 소문자로 변환

```
set, the, shape, to, semi, transparent, by, calling, set_trans, 5
```

- 단순 분석기 (simple analyzer)
  - 문자가 아닌 것은 제외하고 소문자로 변환

```
set, the, shape, to, semi, transparent, by, calling, set, trans
```

내장 분석기 (2)

- 공백문자 분석기 (whitespace analyzer)
  - 공백문자를 기준으로 분리, 소문자로 변환 하지 않음

```
Set, the, shape, to, semi-transparent, by, calling, set_trans(5)
```

- 다국어 분석기 (language analyzer)
  - 특정 언어의 분석기 (한국어 미포함)
  - 영어(english) 분석기의 경우 스톱워드(stopword, and the … 등)를 제거하고 어원으로 변환

```
set, shape, semi, transpar, call, set_tran, 5
```


한글 분석기

- 엘라스틱서치에서는 한글을 위한 별도의 분석기를 제공하지 않음
- 아래와 같은 별도의 플러그인을 설치해야 함
  - http://eunjeon.blogspot.kr/

분석기 적용 시점

- 분석기는 전문 필드(full-text field) 질의 시에만 적용
- 앞의 예 적용

```
GET /_search?q=2014              # 12 results
GET /_search?q=2014-09-15        # 12 results !
GET /_search?q=date:2014-09-15   # 1  result
GET /_search?q=date:2014         # 0  results !
```

  - 필드가 생략된 _all 필드 검색은 전문 문자열 검색
    - 따라서 2014, 09, 15 3개의 텀 검색
    - 위의 첫 2개 질의에서 모두 매치
  - date 필드 질의 시에는 정확한 date 형 값 검색
    - 3번째 질의 매치
    - 4번째 질의는 매치된 값 없음

분석기 테스트

- _analyze API를 사용하여 텍스트 데이터의 분석 결과를 출력

```
GET /_analyze
{
  "analyzer": "standard",
  "text": "Text to analyze"
}
```

```
{
   "tokens": [
      {
         "token":        "text",
         "start_offset": 0,
         "end_offset":   4,
         "type":         "<ALPHANUM>",
         "position":     1
      },
      {
         "token":        "to",
         "start_offset": 5,
         "end_offset":   7,
         "type":         "<ALPHANUM>",
         "position":     2
      },
      {
         "token":        "analyze",
         "start_offset": 8,
         "end_offset":   15,
         "type":         "<ALPHANUM>",
         "position":     3
      }
   ]
}
```

  - start_offset, end_offset은 문자의 위치
  - position은 텀의 순서

분석기 테스트 실행 예

- position은 0 부터 시작

```
$ curl -XGET 'localhost:9200/_analyze?pretty' -H 'Content-Type: application/json' -d'
> {
>   "analyzer": "standard",
>   "text": "Text to analyze"
> }
> '
{
  "tokens" : [
    {
      "token" : "text",
      "start_offset" : 0,
      "end_offset" : 4,
      "type" : "<ALPHANUM>",
      "position" : 0
    },
    {
      "token" : "to",
      "start_offset" : 5,
      "end_offset" : 7,
      "type" : "<ALPHANUM>",
      "position" : 1
    },
    {
      "token" : "analyze",
      "start_offset" : 8,
      "end_offset" : 15,
      "type" : "<ALPHANUM>",
      "position" : 2
    }
  ]
}
$
```


매핑 (Mapping)

- 매핑은 데이터의 저장 형태(자료형 속성)를 기술하고, 검색 엔진에서 해당 데이터를 접근하고 처리하는 방법에 대한 명세
  - 관계형 데이터베이스의 스키마에 해당

- 주요 자료형
  - 문자열: string
  - 정수: byte, short, integer, long
  - 실수: float, double
  - 날짜: date

동적 매핑 (Dynamic Mapping)

❑ 동적 매핑

- 새로운 필드의 다큐먼트를 색인할 때 JSON의 기본 자료형으로 부터 아래와 같이 필드의 자료형을 추정

JSON type	Field type
Boolean: `true` or `false`	`boolean`
Whole number: `123`	`long`
Floating point: `123.45`	`double`
String, valid date: `2014-09-15`	`date`
String: `foo bar`	`string`

매핑의 확인

❑ _mapping API를 사용하여 매핑을 확인

```
GET /gb/_mapping/tweet
```

```
{
   "gb": {
      "mappings": {
         "tweet": {
            "properties": {
               "date": {
                  "type": "date",
                  "format": "strict_date_optional_time||epoch_millis"
               },
               "name": {
                  "type": "string"
               },
               "tweet": {
                  "type": "string"
               },
               "user_id": {
                  "type": "long"
               }
            }
         }
      }
   }
}
```

정적 매핑

❑ 정적 매핑

- 필드의 자료형 속성 등을 지정
- 전문 문자열과 정확한 값의 문자열 구분, 다국어 분석기, 날짜 형식, 분석기 등을 지정

- type 속성: 필드의 자료형 지정

```
{
    "number_of_clicks": {
        "type": "integer"
    }
}
```

- index 속성: string 형의 색인 방식을 지정
  - analyzed : 전문으로 취급하여 분석 후 색인 디폴트로 지정
  - not_analyzed: 정확한 값의 문자열로 취급하여 분석하지 않고 색인
  - no: 색인을 하지 않음. 검색할 수 없음

```
{
    "tag": {
        "type":     "string",
        "index":    "not_analyzed"
    }
}
```

- analyzer 속성: 다국어 분석기 지정

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


매핑의 갱신

❑ 정적 매핑의 2가지 방법

- 인덱스를 생성할 때 매핑의 속성을 지정
- 인덱스 생성 후 매핑의 새로운 속성을 추가
  - 기존의 매핑에 추가는 가능하지만 기존의 속성의 변경은 안됨

❑ 매핑 갱신 예

- 기존의 gb 인덱스 삭제
- tweet 필드에 english 분석기 지정한 새로운 인덱스 생성
- tag 필드에 not_analyzed 지정 추가

```
DELETE /gb
```

매핑 갱신 예

```
PUT /gb 1
{
  "mappings": {
    "tweet" : {
      "properties" : {
        "tweet" : {
          "type" :    "string",
          "analyzer": "english"
        },
        "date" : {
          "type" :    "date"
        },
        "name" : {
          "type" :    "string"
        },
        "user_id" : {
          "type" :    "long"
        }
      }
    }
  }
}
```

```
PUT /gb/_mapping/tweet
{
  "properties" : {
    "tag" : {
      "type" :    "string",
      "index":    "not_analyzed"
    }
  }
}
```


매핑 테스트

- analyze API를 사용하여 다음의 두 요청의 결과를 비교

```
GET /gb/_analyze
{
  "field": "tweet",
  "text": "Black-cats" 1
}

GET /gb/_analyze
{
  "field": "tag",
  "text": "Black-cats" 2
}
```

- tweet 필드는 black, cat 의 두 개 텀 생성
- *tag 필드는 Black-cats 의 한 개 텀 생성*
  => 엘라스틱서치 5.1.1 버전에서는 분석기 지정 하지 않아 에러

매핑 갱신 실행 예 (1)

```
$ curl -XDELETE 'localhost:9200/gb?pretty'
{
  "acknowledged" : true
}
$
$ curl -XPUT 'localhost:9200/gb?pretty' -H 'Content-Type: application/json' -d'
> {
>   "mappings": {
>     "tweet" : {
>       "properties" : {
>         "tweet" : {
>           "type" :    "string",
>           "analyzer": "english"
>         },
>         "date" : {
>           "type" :   "date"
>         },
>         "name" : {
>           "type" :   "string"
>         },
>         "user_id" : {
>           "type" :   "long"
>         }
>       }
>     }
>   }
> }
> '
{
  "acknowledged" : true,
  "shards_acknowledged" : true
}
$
```


매핑 갱신 실행 예 (2)

```
$ curl -XPUT 'localhost:9200/gb/_mapping/tweet?pretty' -H 'Content-Type: application/json' -d'
> {
>   "properties" : {
>     "tag" : {
>       "type" :    "string",
>       "index":    "not_analyzed"
>     }
>   }
> }
> '
{
  "acknowledged" : true
}
$
```

```
$ curl -XGET 'localhost:9200/gb/_mapping?pretty'
{
  "gb" : {
    "mappings" : {
      "tweet" : {
        "properties" : {
          "date" : {
            "type" : "date"
          },
          "name" : {
            "type" : "text"
          },
          "tag" : {
            "type" : "keyword"
          },
          "tweet" : {
            "type" : "text",
            "analyzer" : "english"
          },
          "user_id" : {
            "type" : "long"
          }
        }
      }
    }
  }
}
```

매핑 테스트 실행 예

```
$ curl -XGET 'localhost:9200/gb/_analyze?pretty' -H 'Content-Type: application/json' -d'
> {
>   "field": "tweet",
>   "text": "Black-cats"
> }
> '
{
  "tokens" : [
    {
      "token" : "black",
      "start_offset" : 0,
      "end_offset" : 5,
      "type" : "<ALPHANUM>",
      "position" : 0
    },
    {
      "token" : "cat",
      "start_offset" : 6,
      "end_offset" : 10,
      "type" : "<ALPHANUM>",
      "position" : 1
    }
  ]
}
```

```
$ curl -XGET 'localhost:9200/gb/_analyze?pretty' -H 'Content-Type: application/json' -d'
> {
>   "field": "tag",
>   "text": "black-cats"
> }
> '
{
  "error" : {
    "root_cause" : [
      {
        "type" : "remote_transport_exception",
        "reason" : "[esVB-1][192.168.56.101:9300][indices:admin/analyze[s]]"
      }
    ],
    "type" : "illegal_argument_exception",
    "reason" : "Can't process field [tag], Analysis requests are only supported on tokenized fields"
  },
  "status" : 400
}
```


복잡한 자료형

❑ 배열 자료형

- 다 수의 같은 자료형을 기술
- 색인 후에 배열 원소의 순서는 유지하지 않음

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

❑ 빈 필드

- null 값은 필드의 값이 없음을 표시
- 다음 예는 모두 빈 필드임

```
"null_value":                null,
"empty_array":               [],
"array_with_null_value":     [ null ]
```

복잡한 자료형 – 객체 (Object)

- 필드에 객체를 저장
  - user 필드의 객체 예
    - 내부에 name 필드도 객체 자료형

```
{
    "tweet":            "Elasticsearch is very flexible",
    "user": {
        "id":           "@johnsmith",
        "gender":       "male",
        "age":          26,
        "name": {
            "full":     "John Smith",
            "first":    "John",
            "last":     "Smith"
        }
    }
}
```


객체의 매핑

```
{
  "gb": {
    "tweet": { ❶
      "properties": {
        "tweet":            { "type": "string" },
        "user": { ❷
          "type":           "object",
          "properties": {
            "id":           { "type": "string" },
            "gender":       { "type": "string" },
            "age":          { "type": "long"   },
            "name":   { ❸
              "type":       "object",
              "properties": {
                "full":     { "type": "string" },
                "first":    { "type": "string" },
                "last":     { "type": "string" }
              }
            }
          }
        }
      }
    }
  }
}
```

- 객체는 object 자료형을 갖고, properties 속성 으로 지정

내부 객체의 색인

- 루씬(Lucene)은 객체를 지정하지 못해서, 엘라스틱서치는 다큐먼트의 색인 시에 객체를 아래와 같이 변환

```
{
    "tweet":            [elasticsearch, flexible, very],
    "user.id":          [@johnsmith],
    "user.gender":      [male],
    "user.age":         [26],
    "user.name.full":   [john, smith],
    "user.name.first":  [john],
    "user.name.last":   [smith]
}
```


객체의 배열

- 내부 객체들의 배열과 색인 예

```
{
    "followers": [
        { "age": 35, "name": "Mary White"},
        { "age": 26, "name": "Alex Jones"},
        { "age": 19, "name": "Lisa Smith"}
    ]
}
```

```
{
    "followers.age":    [19, 26, 35],
    "followers.name":   [alex, jones, lisa, smith, mary, white]
}
```

- {age:35} 와 {name: Mary White}의 연관성을 손실
  - "35세인 Mary White가 있는가? " 라는 질의에 검색 못함

=> 별도의 nested 필드를 사용하여 객체를 기술해야 함

과 제

- 자신만의 데이터에 대한 검색, 매핑과 분석 적용 및 실행 예

참고 자료

- Elasticsearch: The Definitive Guide, Getting Started
  - Searching – The Basic Tools
    - https://www.elastic.co/guide/en/elasticsearch/guide/current/search.html
  - Mapping and Analysis
    - https://www.elastic.co/guide/en/elasticsearch/guide/current/mapping-analysis.html