엘라스틱서치 실무 가이드

자바카페 스터디 / 23.09.19 ~ 23.12.12

컴공선배	제리	소프트스퀘어드	정우현

1장. 검색 시스템 이해하기

- **검색 엔진 : 정보를 수집해 검색 결과를 제공하는 프로그램**
- **검색 시스템 : 검색 엔진을 기반으로 구축된 시스템**
- **검색 서비스 : 검색 시스템을 활용해 검색 결과를 서비스로 제공하는 것**

검색 시스템의 구성 요소

1. 수집기 : 크롤러, 스파이더, 웜, 웹로봇 등으로 불리는 웹에서 필요한 정보를 수집하는 프로그램
2. 스토리지 : 색인한 데이터를 저장하는 물리적 저장소
3. 색인기 : 다양한 형태소 분석기를 조합해 정보에서 의미 있는 용어를 추출하고 검색에 유리한 역색인 구조로 데이터를 저장하는 것
4. 검색기 : 형태소 분석기를 이용하여 색인기에서 저장한 역색인 구조에서 일치하는 문서를 찾아 결과로 반환하는 것

엘라스틱서치와 관계형 데이터베이스 비교

엘라스틱서치	관계형 데이터베이스
인덱스 Index	데이터베이스 Database
샤드 Shard	파티션 Partition
타입 Type	테이블 Table
문서 Document	행 Row
필드 Field	열 Column
매핑 Mapping	스키마 Schema
Query DSL	SQL

엘라스틱서치 HTTP Method / 기능 / 데이터베이스 질의 문법

GET / 데이터 조회 / SELECT

PUT / 데이터 생성 / INSERT

POST / 인덱스 업데이트, 데이터 조회 / UPDATE, SELECT

DELETE / 데이터 삭제 / DELETE

HEAD / 인덱스의 정보 확인 / -

엘라스틱서치 사용 예시 **curl -X (메서드) http://host:port/(인덱스)/(타입)/(문서 id) -d '{json 데이터}'**

엘라스틱서치의 장점

1. 오픈소스 검색엔진 : Apache Lucene 기반 오픈소스 검색엔진이다.

2. 전문 검색 : RDB 와 달리 전문 검색(Full Text) 이 가능하다.
   *** 내용 전체를 색인해서 특정 단어가 포함된 문서를 검색하는 것**

3. 통계 분석 : 비정형 로그 데이터를 수집하고 한 곳에 모아 통계 분석을 할 수 있다.

4. 스키마리스 : 정형화되지 않은 다양한 형태의 문서도 자동으로 색인하고 검색 할 수 있다.

5. RESTful API : HTTP 기반의 RESTful API 를 지원하고 JSON 형식을 사용 할 수 있다.

6. 멀티테넌시 (Multi-tenancy) : 서로 상이한 인덱스일지라도 검색 할 필드명만 같으면 여러 개의 인덱스를 한번에 조회 할 수 있다.

7. Document-Oriented : 여러 계층의 데이터를 JSON 형식의 구조화된 문서로 인덱스에 저장 할 수 있다.

8. 역색인 (Inverted Index) : 일반적인 NoSQL 과 달리 역색인을 제공한다.
   *** 종이책 마지막 페이지에 제공하는 색인 페이지와 비슷하게 용어와 문서 위치를 매핑해둔 것**

9. 확장성과 가용성 : 분산 구성이 가능하여 대량의 문서를 효율적으로 처리 할 수 있다.

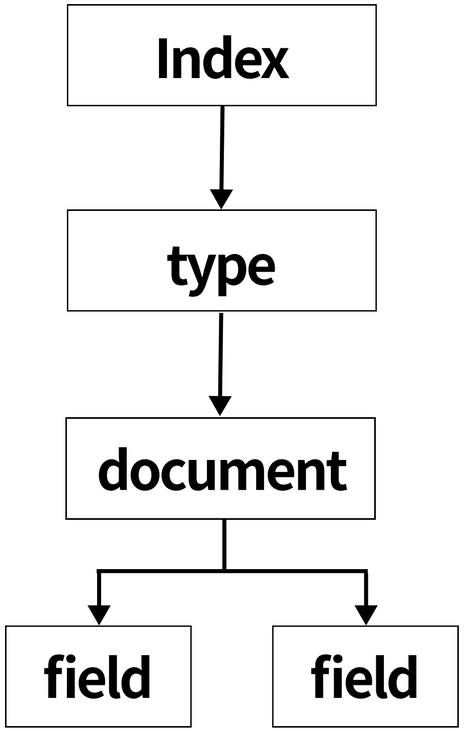
엘라스틱서치의 약점

1. 준 실시간 (Near Realtime) : 색인된 데이터는 Commit 과 Flush 같은 복잡한 과정을 거치며 통상적으로 1초 뒤에나 검색이 가능해진다.
   때문에 실시간이 아니다.

2. 트랜잭션(Transaction) 과 롤백(Rollback) 미지원 : 분산 시스템으로 구성하여기 때문에 클러스터의 성능 향상을 위해 시스템적 비용 소모가 큰 트랜잭션과 롤백을 지원하지 않는다. 최악의 경우 데이터 손실의 위험이 있다.

3. 데이터 업데이트 미제공 : 업데이트 요청이 오더라도 기존 문서를 삭제하고 새로운 문서를 생성하는 방식이다.
   비용적으로 비효율적으로 보이지만 불변적(Immutable) 이라는 이점을 취할 수 있다는 장점도 있다.

2장. 엘라스틱서치 살펴보기

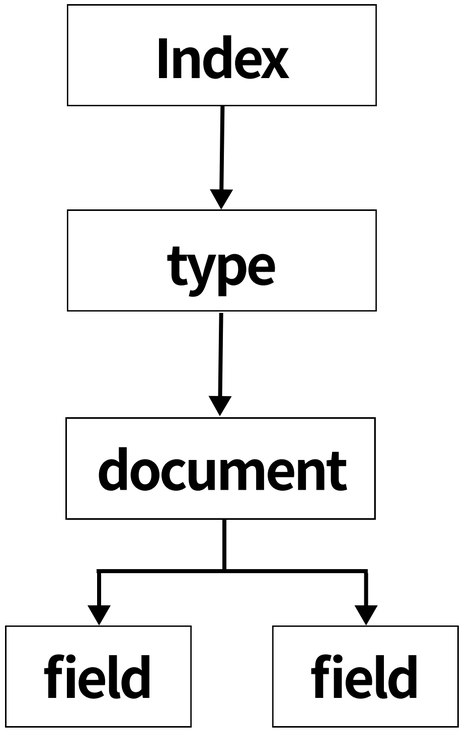


인덱스 (Index)

- 데이터 저장 공간
- 1개의 인덱스는 1개의 타입만 가진다.
- 하나의 물리적인 노드에 여러 개의 논리적 인덱스를 생성 할 수 있다.
- 검색 시 인덱스 이름으로 문서 데이터를 검색한다.
- 여러 개의 인덱스를 동시에 검색하는 것이 가능하다.
- 분산 환경으로 구성시 하나의 인덱스가 여러 노드에 분산 저장되어 관리된다.
- 인덱스 생성시 5개의 Primary 샤드와 1개의 Replica 샤드 세트를 생성한다.

샤드 (Shard)

- 색인된 문서는 하나의 인덱스에 담긴다.
- 인덱스 내부에 색인된 데이터는 물리적인 공간에 여러 개의 파티션으로 나뉘어 구성되는데, 엘라스틱서치에서는 파티션을 샤드라고 부른다.
- 다수의 샤드로 문서를 분산 저장하고 있어 데이터 손실 위험을 최소화 할 수 있다.



타입 (type)

- 인덱스의 논리적 구조를 의미한다.
- 인덱스 속성에 따라 분류하기도 한다.
- 1개의 인덱스에 1개의 타입을 설정 할 수 있다.

문서 (Document)

- 데이터가 저장되는 최소 단위
- JSON 포맷으로 저장된다.
- 하나의 문서는 여러 개의 필드로 구성되어있다.

필드 (Field)

- 문서를 구성하기 위한 속성
- 하나의 필드는 목적에 따라 다수의 데이터 타입을 가질 수 있다.

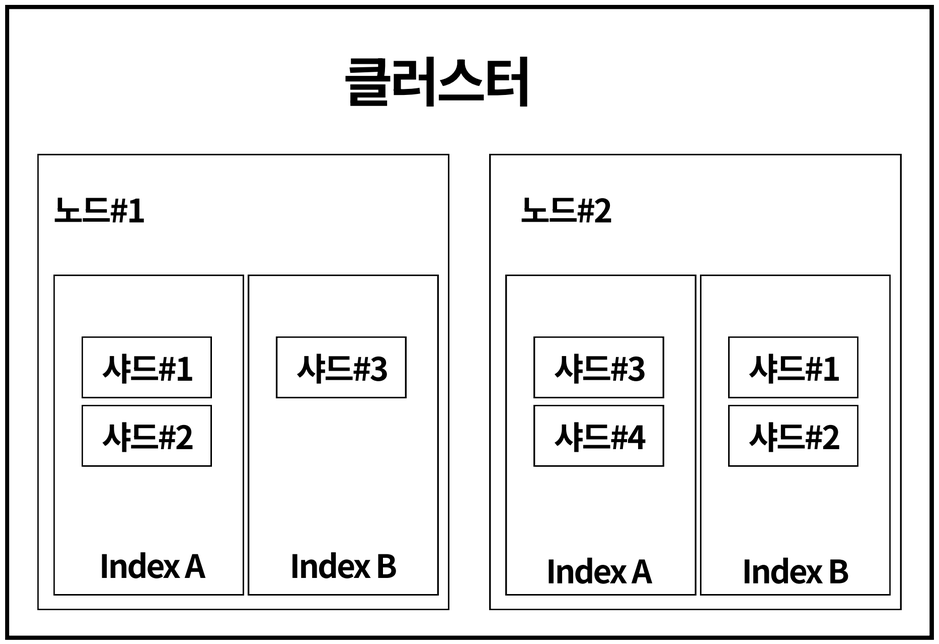
매핑 (Mapping)

- 문서의 필드와 속성을 정의하고, 그에 따른 색인 방법을 정의하는 프로세스를 의미한다.
- 인덱스의 매핑 정보에는 여러 가지 데이터 타입을 지정 할 수 있지만, 필드명은 중복해서 사용 할 수 없다.

노드의 종류

클러스터 (Cluster)

- 물리적인 노드 인스턴스들의 모임
- 모든 노드의 검색과 색인 작업을 관장하는 논리적 개념



마스터 노드 (Master Node)

- 클러스터를 관리한다.
- 노드 추가 및 제거 같은 클러스터 전반적 관리를 담당한다.

데이터 노드 (Data Node)

- 실질적인 데이터를 저장한다.
- 검색과 통계 같은 데이터 관련 작업을 수행한다.

코디네이팅 노드 (Coordinationg Node)

- 사용자의 요청만 받아서 처리한다.
- 클러스터 관련 요청은 마스터 노드에 전달하고,
  데이터 관련 요청은 데이터 노드에 전달한다.

인제스트 노드 (Ingest Node)

- 문서의 전처리 작업을 담당한다.
- 인덱스 생성 전 문서의 형식을 다양하게 변경 할 수 있다.

Index vs Indices

- Index = 색인 이란, 데이터가 토큰화되어 저장된 자료구조
- 하지만 엘라스틱서치에서는 인덱스라는 용어를 다르게 쓴다.

- Index : 색인 데이터
- Indexing : 색인하는 과정
- Indices : 매핑 정보를 저장하는 논리적인 데이터 공간

*** 스키마리스 기능은 성능 이슈가 발생 할 수 있으니 사용을 지양하자.**

인덱스 관리 API (Indices API)

- 인덱스 생성시 매핑 정보 정의하여 필드에 데이터 타입 지정 가능
  - 데이터 타입 : keyword(단순 문자열), text(형태소 분석용 문자열), integer(숫자), date(날짜) ...

문서 관리 API (Document API)

- 문서 추가/수정/삭제
- Single Document API 라고도 부른다.
  - Index API : 한 건의 문서를 색인
  - Get API : 한 건의 문서를 조회
  - Delete API : 한 건의 문서를 삭제
  - Update API : 한 건의 문서를 업데이트

- Multi-Document API 도 제공한다.
  - Multi Get API : 다수의 문서를 조회
  - Bulk API : 대량의 문서를 색인
  - Delete By Query API : 다수의 문서를 삭제
  - Update By Query API : 다수의 문서를 업데이트
  - Reindex API : 인덱스의 문서를 다시 색인

검색 API (Search API)

HTTP URI 형태의 파라미터를 URI 에 추가해 검색하는 방법

- GET /movies/_doc/{VALUE}
- POST /movies/_search?q={VALUE}
- POST /movies/_search?q={FIELD}:{VALUE}

RESTful API 방식인 QueryDSL 을 사용해 Request Body 에 질의 내용을 추가해 검색하는 방법

- POST /{index 명}/_search
  { // JSON 쿼리 구문
    size : 몇 개의 결과를 반환할지 결정 (Default 10)
    from : 0 일 경우 상위 0~10건을 반환 (Default 0)
    _source : 특정 필드만 결과로 반환하고 싶을때 정의
    sort : 특정 필드 기준 asc, desc 가능
    query : 검색될 조건 정의
    filter : 검색 결과 중 특정 값을 필터링 가능,
        결과 내 재검색 이라고 보면 된다.
        필터를 사용하면 자동 score 정렬은 안된다.
  }

집계 API (Aggregation API)

```
POST /movies/_search?size=10
{
  aggs: {
    집계명: {
      terms: {
        filed : 필드명
      }
    }
  }
}
```

- 집계 내에 집계를 넣을 수 있다. = 버킷 안에 버킷을 구성 할 수 있다.
- 버킷(Bucket) 집계 : 필드를 기준으로 버킷을 집계
- 메트릭(Matric) 집계 : 추출된 값을 가지고 Sum, Max, Min, Avg 계산
- 매트릭스(Matrix) 집계 : 행렬의 값을 합하거나 곱한다.
- 파이프라인(Pipeline) 집계 : 버킷에서 도출된 결과 문서를 다른 필드 값으로 재분류한다.
  - 다른 집계에 의해 생성된 출력 결과를 다시 한번 집계한다.

3장. 데이터 모델링

데이터 유형에 따라 필드에 적절한 데이터 타입을 지정하는 것
= 매핑 (Mapping) = 데이터 모델링

매핑 시 고려 할 사항

1. 문자열을 분석 할 것인가?
2. _source 에 어떤 필드를 정의 할 것인가?
3. 날짜 필드를 가지는 필드는 무엇인가?
4. 매핑에 정의되지 않고 유입되는 필드는 어떻게 처리 할 것인가?

인덱스 생성 및 매핑 확인

PUT movie_search

GET movie_search/_mapping

매핑 파라미터

- analyzer : 형태소 분석할 분석기 지정. Default 는 Standard Analyzer
- normalizer : term query 에 분석기를 사용하기 위해 사용.
  cafe, Cafe 와 같은 서로 다른 문자열을 normalizer 에 asciifolding 필터를 사용하면 같은 데이터로 인식 가능
- boost : 필드에 가중치를 부여하는 것
  * 가중치(Weight)에 따라 유사도 점수(_score) 가 달라지기 때문에 boost 설정시 검색 결과의 노출 순서에 영향을 준다.
  * 색인 시점에 boost 설정을 하면 재색인 하지 않는 이상 가중치 변경이 불가능하기 때문에 주의해야한다.
- coerce : 색인시 자동 변환을 허용할 것인지 설정. “10” 과 같은 숫자 형태의 문자열이 integer 필드에 들어올 경우 자동 형변환을 수행 할 것인가
- copy_to : 하나 혹은 여러개의 필드를 1개의 필드로 복사 할 수 있다. 복사 할 때 타입도 변경 할 수 있다.
- fielddata : 엘라스틱서치가 힙 공간에 생성하는 메모리 캐시. 최신 엘라스틱서치는 doc_values 라는 캐시를 제공하고있음. filedata 는 잘 안쓰
- doc_values : 엘라스틱서치에서 사용하는 기본 캐시. text 타입을 제외한 모든 타입에서 사용.
- dynamic : 매핑에 필드를 추가 할 때 동적으로 생성할지, 생성하지 않을지 결정. true / false / strict (새로운 필드가 들어오려고하면 예외 발생시킴)
- enabled : 검색 결과에는 포함하지만 색인은 하고 싶지 않을 경우 사용

매핑 파라미터

- format : 날짜/시간을 문자열로 표시하기 때문에 포맷팅 가능.
- ignore_above : 필드에 저장되는 문자열이 지정한 크기를 넘어서면 빈 값으로 색인한다.
- ignore_malformed : 잘못된 데이터 타입을 색인하려고하면 예외가 발생하고 해당 문서 전체가 색인되지 않는다.
  해당 필드의 에러를 무시하고 색인을 하려면 ignore_malformed 옵션을 활성화해준다.
- index : 필드 값을 색인 할것인가 아닌가. Default 값은 true
- fields : 다중 필트를 설정 할 수 있는 옵션. 필드안에 필드 가능
- norms : _score 값 계산에 필요한 정규화 인수 사용 여부. Default 값은 true. _score 계산이 필요없는 필드라면 비활성화하기
- null_value : 문서에 필드가 없거나 null 인 경우 색인 시 필드를 생성하지 않는다. 이 경우 null_value 를 설정하면 null 이어도 필드를 생성한다.
- position_increment_gap : 필드 데이터 중 단어와 단어 사이에 간격(slop) 을 허용할지 말지 설정하는 것.
  배열 형태 데이터를 색인 할 때 정확도를 높일 수 있다.
- properties : 오브젝트 타입이나 중첩(Nested) 타입의 스키마를 정의 할 때 사용하는 옵션.

매핑 파라미터

- similarity : 유사도 측정 알고리즘을 지정. 기본 알고리즘인 BM25 에서 다른 알고리즘으로 변경 가능 (BM25, classic, boolean 등)

- store : 필드의 값을 저장해 검색 결과에 값을 포함하기 위한 매핑 파라미터.
  엘라스틱 서치에서는 색인된 문서가 _source 에 저장되지만, store 옵션을 사용하면 해당 필드를 자체적으로 저장 할 수 있다.

- term_vector : 루씬에서 분석된 용어의 정보를 포함할지 말지. (no / yes / with_positions / with_offsets / with_positions_offsets)

메타 필드 : 엘라스틱서치에서 생성한 문서에서 제공하는 메타데이터 저장하는 특수 목적 필드

- _index : 문서가 속한 인덱스의 이름
- _type : 문서가 속한 매핑의 타입 정보
- _id : 문서를 식별하는 유일 키
- _uid : 특수한 목적의 식별 키. “#” 태그를 사용해 _type 과 _id 값을 조합해서 사용한다.
- _source : 문서의 원본 데이터를 제공.
  - _reindex API 나 스크립트를 사용해 _source 메타 필드를 변경 할 수 있다.
- _all : 색인에 사용된 모든 필드의 정보를 가짐. 모든 필드의 내용이 하나의 텍스트로 합쳐져서 제공.
  6.0 ver 부터 deprecated 되어서 copy_to 파라미터를 사용해야한다.
- _routing : 특정 문서를 특정 샤드에 저장하기 위해 사용.

필드 데이터 타입

- keyword, text 같은 문자열 타입
- date, long, double, integer, boolean, ip 같은 일반적인 데이터 타입
- 객체 또는 중첩문과 같은 JSON 계층 특성의 데이터 타입
- geo_point, geo_shape 같은 특수한 데이터 타입

- keyword : 형태소 분석을 하지 않을 텍스트 타입. 아래의 목적일때 사용
  - 검색 시 필터링 되는 항목
  - 정렬이 필요한 항목
  - 집계해야하는 항목

- text : 형태소 분석을 할 텍스트 타입. 색인시 지정된 분석기가 데이터를 문자열로 인식하고 분석한다.
  - 검색 뿐만 아니라 정렬이나 집계 연산을 사용 해야 할 경우 text 와 keyword 타입을 동시에 갖도록 멀티 필드로 설정 가능하다.

- array : 문자열 배열, 숫자 배열, 객체 배열 생성 할 수 있다. array 내의 타입은 지정이 불가능하며 배열 내의 값은 모두 똑같은 타입이어야만한다.

- numeric : 숫자 데이터 타입
  - long, integer, short, byte, double, float, half_float

- date : 문자열로 표현되는 날짜 값. 문자열로 인식하기 때문에 format 을 명시하도록 하자.

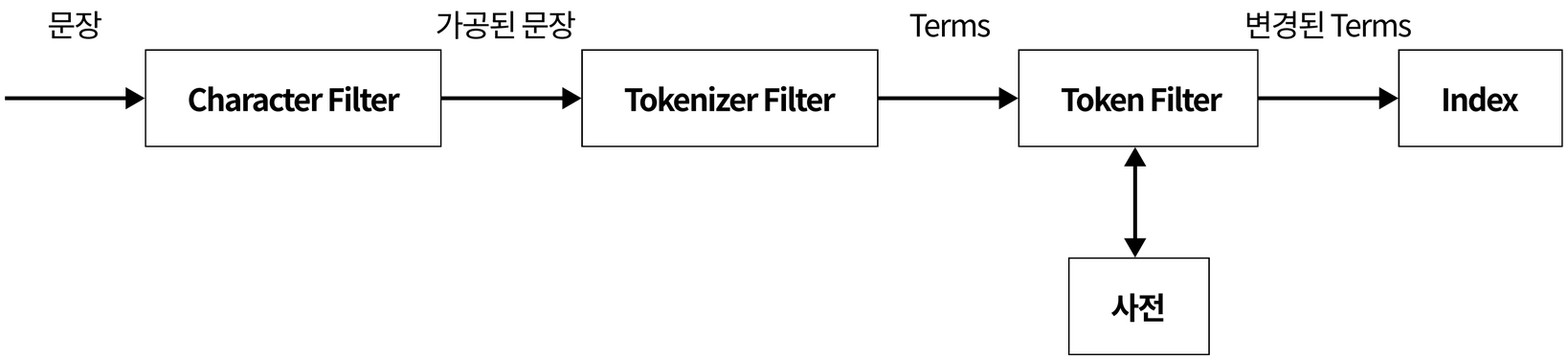
필드 데이터 타입

- range : 범위가 있는 데이터를 저장 할 때 사용
  - integer_range
  - float_range
  - long_range
  - double_range
  - date_range
  - ip_range
- boolean : 참과 거짓을 가지는 데이터 타입
- geo-point : 위도, 경도 등 위치 정보를 담은 데이터 타입
- ip : IPv4 와 IPv6 모두 지정 할 수 있는 데이터 타입
- object : 계층 구조로 데이터를 저장 할 수 있는 데이터 타입
- nested : array 타입은 OR 조건으로 검색이 된다. nested 를 사용하면 AND 조건처럼 검색 할 수 있다.

엘라스틱서치 분석기

- 역색인 구조
  - 모든 문서가 가지는 단어의 고유 단어 목록
  - 해당 단어가 어떤 문서에 속해 있는지에 대한 정보
  - 전체 문서에 각 단어가 몇 개 들어있는지에 대한 정보
  - 하나의 문서에 단어가 몇 번씩 출현했는지에 대한 빈도

- 분석기의 구조
  1. 문장을 특정한 규칙에 의해 수정한다. = CHARACTER FILTER
  2. 수정한 문장을 개별 토큰으로 분리한다. = TOKENIZER FILTER
  3. 개별 토큰을 특정한 규칙에 의해 변경한다. = TOKEN FILTER



분석기 사용법

- 색인시 사용 할 분석기와 검색시 사용할 분석기를 따로 정의 할 수 있다.

대표적인 분석기

- Standard Analyzer : 공백 혹은 특수 기호를 기준으로 토큰을 분리하고 모든 문자를 소문자로 변경하는 토큰 필터를 사용하는 분석기
- Whitespace Analyzer : 공백 문자열을 기준으로 토큰을 분리하는 분석기
- Keyword Analyzer : 전체 입력 문자열을 하나의 키워드처럼 처리하는 분석기

전처리 필터 (Character Filter)

- HTML strip char filter : 문장에서 HTML 을 제거하는 필터

토크나이저 필터 (Tokenizer Filter)

- Standard 토크나이저 : 일반적으로 사용하는 토크나이저로써 대부분의 기호를 만나면 토큰으로 나눈다.
- Whitespace 토크나이저 : 공백을 만나면 텍스트를 토큰화한다.
- Ngram 토크나이저 : 한 글자씩 토큰화한다. 특정 문자를 지정 할 수도 있으며, 다른 옵션들을 사용해서 자동완성을 만들때 유용하게 활용이 가능하다.
- Edge Ngram 토크나이저 : 지정된 문자의 목록 중 하나를 만날 때마다 시작 부분을 고정시켜 단어를 자르는 방식. 자동완성을 만들때 유용하게 활용이 가능하다.
- Keyword 토크나이저 : 텍스트를 하나의 토큰으로 만든다.

토큰 필터 (Token Filter)

- Ascii Folding 토큰 필터 : 아스키 코드에 해당하는 127개의 알파벳, 숫자, 기호에 해당하지 않는 경우 문자를 ASCII 요소로 변경한다.
- Lowercase 토큰 필터 : 토큰을 구성하는 전체 문자열을 소문자로 변환한다.
- Uppercase 토큰 필터 : 토큰을 구성하는 전체 문자열을 대문자로 변경한다.
- Stop 토큰 필터 : 불용어로 등록할 사전을 구축해서 사용하는 필터이다.
- Stemmer 토큰 필터 : Stemming 알고리즘을 사용해 토큰을 변형하는 필터이다.
- Synonym 토큰 필터 : 동의어를 처리 할 수 있는 필터이다.
- Trim 토큰 필터 : 앞뒤 공백을 제거하는 필터이다.

동의어 사전

- 매핑 설정 정보에 파라미터로 등록하는 방식 < **특정 파일을 별도로 생성해서 관리하는 방식**

- 동의어 추가는 쉼표(,) 로 분리해서 등록한다.

- 동의어 치환은 화살표(=>) 로 표시한다.

- 동의어 사전은 실시간으로 적용되지 않는다.
  - 색인 시점에 인덱스를 Reload 하는 방식
  - 검색 시점에 인덱스를 Reload 하는 방식
- 색인 시점에 동의어 사전이 사용되었다면, 동의어 사전이 바뀌더라도 재색인 하지 않으면 반영이 안된다.
- 검색 시점에 동의어 사전이 사용되었다면, 검색 할 때마다 재색인 된다.
  - 동의어 사전이 빈번하게 수정되는 인덱스의 경우 색인 시점이 아니라 검색 시점에 동의러 사전을 사용하는 것이 좋다.

Document API 이해하기

- 엘라스틱서치에서 제공하는 대표적인 Document API
  - Index API : 문서를 생성 = PUT Method
  - Get API : 문서를 조회 = GET Method
  - Delete API : 문서를 삭제 = DELETE Method
    - Delete By Query API : 검색 결과로 나온 문서만 삭제하고 싶을때 사용
  - Update API : 문서를 수정 = POST Method
  - Bulk API : 대량의 문서를 처리 = POST Method
  - Reindex API : 문서를 복사 = POST Method

문서 파라미터

- _id : 문서의 고유한 키
- _version : 도큐먼트 갱신/삭제시 마다 자동 증가
- op_type : 데이터가 존재 할 경우 update 하지 않고 색인이 실패하길 원할때 사용
- timeout : 색인 작업의 최대 시간 지정
- 인덱스 매핑 정보 자동 생성 : action.auto_create_index 와 index.mapper.dynamic 옵션을 사용하여 인덱스의 자동 생성 여부와 동적 매핑 사용 여부를 true / false 설정 할 수 있다.