Apache Solr (아파치 솔라)

- 오픈소스 정보 검색 라이브러리
- 엔터프라이즈 검색 서버인만큼 다양한 기능을 지원
    - 텍스트 검색. 다면적 검색, 실시간 인덱싱, 클러스터링. DB 통합
    - 다양한 문서 처리 및 검색, 솔라 분산 인덱싱 등

- 루씬 검색 라이브러리 기반의 파일을 인덱스하는 검색 엔진 사용
- HTTP 프로토콜 상에서 XML을 통해 색인
- 검색은 HTTP GET으로 쿼리를 보낼 수 있으며 XML의 형태로 값을 얻음
- 솔라은 인덱스 복제라는 기술을 활용
    - Solr는 대규모 검색 볼륨에 대한 쿼리에 적절한 대응력을 제공
- 솔라 검색 서버 URL을 사용하기에 인터넷을 통해 파일을 질의, 인덱스하는 어느 곳에서나 접근 가능

사용 방법

Downloads

- `https://solr.apache.org/downloads.html` 에서 최신 버전의 Binary Release를 다운
    - 8.11.2 버전의 .zip 파일을 다운 받은다.

Unzip

- 원하는 폴더에 압축을 풀어준다.

SolrCloud 모드로 Solr 실행

- Solr를 시작하기 위해 명령어를 입력해야한다.
    - bin 폴더에서 `solr.cmd start -e cloud` 를 cmd에 입력
- 초기에 몇개의 Solr node를 실행하고 싶은지 물어본다.
    - 마지막에 [2] 라고 써있는 것 처럼 기본값이 이미 2로 되어 있으니, 간단하게 엔터 입력
- node를 설정하면 각각의 노드에 할당할 포트 번호를 설정한다.
    - 각각의 포트를 사용하는 곳이 딱히 없으면 엔터 입력
    - 각각 8983/ 7574 포트를 사용하게 해준다.
    - Solr의 기본 포트는 8983 이다.
- 실행이 잘 되었고, 이제 색인을 위한 Collection을 생성해야 하므로 이름을 입력해야한다.
    - 기본적으로 Solr에 포함되어 있는 샘플 데이터인 techproducts를 가져올 수 있다.
    - 따라서 초기에는 `techproducts`를 입력하고 엔터
- 그러면 몇개의 shard를 만들어 techproduct를 나눌지 물어본다.

    - 기본 값이 2로 되어 있다.
    - 기본 값이 2인 것은 두 개의 노드에 균일하게 색인을 나누게 된다.
    - 따라서 그냥 엔터를 입력
- 이번에는 replica를 몇개 만들지 물어본다.
    - 레플리카는 Failover(장애 극복 기능 : 네트워크에 장애가 생겼을 때 준비한 다른 시스템으로 자동 변환) 기능을 위해 사용
    - 엔터를 입력해 기본 값인 2를 선택
- 컬렉션에 적용 할 설정 파일을 선택
    - Solr는 기본적으로 configset이라고 불리는 두 개의 샘플 설정 파일이 있다.
    - 모든 컬렉션은 반드시 두 개의 메인 설정 파일을 포함한 하나의 configset을 갖는다.
        - `Schema File` (managed-schema OR schema.xml)
        - `solrconfig.xml`
    - 입력해야할 설정은 `_default`와 `sample_techproducts_configs` 중 어떤 configset으로 시작할지 선택하는 것
        - 이때, `sample_techproducts_config`를 입력
- 위 과정을 모두 완료하면, Solr를 사용할 준비가 완료된 것이다.
- 만약 Solr를 종료하고 싶으면 bin/solr stop -all을 입력
- 종료한 Solr를 다시 키고 싶으면 아래 명령어를 연속으로 입력
    - `./bin/solr start -c -p 8983 -s example/cloud/node1/solr`를 입력 (첫번째 노드 실행)
    - `/bin/solr start -c -p 7574 -s example/cloud/node2/solr -z localhost:9983`

Solr Admin UI

- 웹 브라우저를 통해 `http://localhost:8983/solr/` 페이지에 방문하면
    - Solr 관리자 모드에 진입 가능
- Solr는 현재 각각 8983과 7574 포트에서 두 개의 노드로 실행되고 있다.
    - techproducts라는 이름으로 자동으로 생성된 하나의 컬렉션이 각각 2개의 레플리카를 가진 두개의 샤드로 나누어져있다.
    - 관리자 화면에서 cloud 탭을 누르면 표로 확인이 가능함

Techproducts 데이터 색인

- 현재 아무런 데이터도 포함하고 있지 않기 때문에 쿼리를 실행해 볼 수 없다.
- Solr는 다양한 타입의 문서를 손쉽게 색인할 수 있도록 /bin/post 경로에 툴을 포함하고 있기에 예제를 색인할 수 있다.
    - Windows에서는 자바 프로그램을 사용해야 예제 툴을 사용할 수 있다.
    - `java -jar -Dc=techproducts -Dauto example\exampledocs\post.jar example\exampledocs\*`

- 이러면 Solr에 데이터가 포함되어 검색할 수 있다.

기본 검색

- Solr는 REST 클라이언트, curl, 포스트맨 등등 다양한 언어로 작성된 클라이언트를 통해 쿼리가 가능
- Solr Admin UI는 쿼리탭을 통한 쿼리 빌더 인터페이스를 포함 하고 있기 때문에 Execute Query 버튼을 클릭해 JSON 포맷의 쿼리 결과를 확인 할 수 있다.
    - UI에서 Collection select를 누르고 `techproducts` 컬렉션을 선택
    - 이후 좌측 하단의 Query 버튼을 누르면 쿼리를 보낼 수 있는 인터페이스가 등장
    - 아무런 수정 없이 `Execute Query`를 누른다.
- 파라미터 q 값인 `*:*`는 인덱스의 모든 document를 검색하라는 뜻
    - rows 파라미터를 설정해주면 해당 값 만큼 n개의 document를 검색해준다.
        - default는 10이기 때문에 설정을 하지 않으면 10개의 document를 검색

단어 검색

- `curl "http://localhost:8983/solr/techproducts/select?q=foundation"`을 입력하면
    - foundation을 포함하는 모든 document를 보여준다.
- fl (Field List)
    - 검색 결과로 받은 document들은 모든 field를 포함하고 있다.
    - 기본적은 동작은 이렇지만 응답에서 보여줄 필드들을 제한하고 싶으면 fl 파라미터에 값을 넣어 요청하면 된다.
    - 각각의 필드명은 `,`(콤마)를 통해 구분
    - fl을 `id`로 제한하면 검색 결과의 필드가 모두 id 만으로 제한할 수 있다.

필드 검색

- solr에서의 쿼리는 일부 필드들을 사용해 검색을 수행
- 보통의 경우에는 여러가지 필드를 동시에 이용해서 검색 하는게 필요
- 하지만 쿼리를 하나의 필드로 제한하여 검색할 수 있다.
- q 파라미터에 `electronics`를 입력하여 query를 수행하면
    - 색인된 모든 필드에서 electronics라는 단어를 검색
- q 파라미터에 `cat:electronics`를 입력하면
    - cat 필드에서 electronics를 찾아 검색해준다.

문장 검색

- 여러개의 단어로 이루어진 문장을 검색하려면 쌍따옴표로 양쪽을 감싸서 쿼리를 작성
  - `CAS latency`를 검색 하려면 Solr Admin UI의 q 박스에 `"CAS latency"`라고 작성
  - curl을 사용한다면 띄어쓰기가 URL 인코딩 된다.
    - `curl "http://localhost:8983/solr/techproducts/select?q=\"CAS+latency\""`

복합 검색

- 기본적으로 단일 쿼리에서 여러개의 단어나 문장들을 검색하면 Solr는 그중 하나만 존재하더라도 검색 결과에 해당하는 document로 인식
- 쿼리에서 요청한 단어를 여러개 포함하면 할 수록 결과 목록에서는 더 높은 순위가 되어 상위에 노출
- 이 때, `+` prefix를 활용하면 반드시 포함 하도록 검색 옵션을 줄 수 있고 그 반대로 `-` prefix를 붙여 포함하지 않는 조건을 거는 것 또한 가능
- 예를 들어 `electronics`와 `music`을 모두 포함하는 검색 결과를 위해 q박스에 `+ electronics +music` 을 입력
- curl을 사용한다면 `+`를 `%2B`로 인코딩
  - `curl "http://localhost:8983/solr/techproducts/select?q=%2Belectronics%20%2Bmusic"`
- electronics를 포함 하지만 music을 포함 하지 않는 검색 결과를 위해 `+electonics -music`을 쿼리
- 인코딩이 필요한 `+`와 다르게 `-`는 인코딩할 필요가 없다.
  - `curl "http://localhost:8983/solr/techproducts/select?q=%2Belectronics+-music"`

컬렉션 삭제

- `bin/solr delete -c techproducts`을 입력하여 techproducts 컬렉션 삭제 가능
- Solr 를 종료하려면 아래의 명령어를 입력
  - `bin/solr stop -all`

새로운 컬렉션 생성

- 새로운 컬렉션이란 새로운 데이터 셋을 사용하는 것
- Solr는 색인하는 과정에서 필드에 어떤 데이터 타입이 있을지 추측하는 field gussing 기능을 제공
  - 또한 field guessing은 새로 들어오는 documents의 새로운 field들을 스키마에 새로운 필드로 자동 추가
  - 이를 Schemaless라고 한다.
- Solr에서의 스키마는 Field와 Field Type에 대해 Solr가 예측하고 이해할 수 있도록 하는 XML 파일이다.
- Schema는 필드나 필드 타입명 뿐만 아니라 색인되기 전 필드에 수행되어햐할 수정사항도 정의

films 컬렉션 생성

- `films` 라는 이름의 컬렉션을 `_default` configset을 사용해 생성
  - `bin/solr create -c films -s 2 -rf 2`
- configset을 따로 설정 하지않으면, 자동으로 `_default` configset이 선택
- `-s`와 `-rf` 옵션은 각각 생성할 Shard와 Replica의 수를 의미
- `_default` configset을 프로덕션(상업) 목적에서 사용하기에는 추천하지 않는다고 경고문이 출력
- Admin UI에 접속하면 films 컬렉션이 생성된 것을 볼 수 있다.

Films 데이터를 위한 Schemaless

- `_default` configset 설정시 두가지 일이 발생
  - Solr Schema API를 통해서만 변경되도록 설정된 managed schema를 사용하게 된다.
    - 즉, 직접 수정 할 수 없기 때문에 어떤 소스로부터 어떤 수정이 왔는지 혼동할 필요가 없다.
    - Solr Schema API가 필드나 필드타입 혹은 다른 스키마 룰을 수정 할 수 있게 해준다.
  - `solrconfig.xml` 파일에 설정된 `field guessing`을 사용하게 된다.
    - 필드 추측은 Document를 색인 하기 전에 document에 있을 것으로 생각되는 모든 필드를 직접 정의하지 않고도 Solr를 바로 시작 할 수 있도록 해준다.
    - 색인할 문서가 있을때 곧바로 Solr를 실행만 하면 필드들을 만들어 내기 때문에 `Schemaless` 라고 한다.
- 한계
  - 무작위 대입으로 추측하기 때문에 만약 추측이 잘못되었다면 문서가 일단 색인 된 후에는 재색인을 하지 않는 이상은 수정이 꽤나 제한적
  - 문서가 몇천건 정도만 있다면 사용 할 만 하지만 수백만건이 넘어간다거나 심지어는 더이상 원래의 데이터에 접근 할 수 없는 상황이라면 상황이 굉장히 심각해진다.
  - 따라서 Solr 커뮤니티에서는 schema를 직접 정의하지 않고 사용하는걸 권장하지 않는다.
  - 자동으로 생성된 스키마가 기대한 것과 일치하는지 그리고 쿼리가 원하는 대로 작동하는지를 항상 확인
- 하지만, Schemaless 기능을 미리 정의해둔 Schema와 섞어서 사용 할 수 있다
  - Schema API를 사용해서 직접 컨트롤 하고 싶은 필드를 정의
  - 덜 중요하거나 테스트를 통해 잘 작동한다는 확신이 있는 필드들에 대해서는 Solr에 맡길 수 있다.

names 필드 생성

- films 데이터는 각각의 영화가 몇개 안되는 필드만을 포함
  - ID
  - director name
  - film name

    - release data
    - genre
- `head --line 16 example/films/films.xml` 을 통해 films.xml을 열어본다.
    - 예제 파일을 확인 해 보면 첫번째 필름의 이름이 `.45` 이며 2006년에 개봉했다는 사실을 확인 할 수있다.
- Solr는 record 상의 데이터를 토대로 field type을 추측하게 되는데 만약 이 데이터를 그냥 색인 한다면 Solr는 첫번째 영화 제목을 보고 필드 타입을 `float` 으로 결정
- 그래서 name이라는 필드가 `FloatPointField`라는 타입으로 자동 생성 되는데 이후로 들어오는 모든 데이터들은 이로인해 name에 float 값이 들어올 것을 기대
- 따라서 'A mighty Wind' 나 'Chicken Run' 같은 문자열로 된 제목들이 있으면
    - Solr가 name 필드를 float로 추측하기 때문에 제목을 색인 하는 과정에서 인덱싱 실패 에러를 맞이
- Solr가 항상 제목을 문자열로 해석 할 수 있도록 name 필드를 색인 전에 미리 설정
    - `curl -X POST -H 'Content-type:application/json' --data-binary '{"add-field": {"name":"name", "type":"text_general", "multiValued":false, "stored":true}}' http://localhost:8983/solr/films/schema`
- 위의 명령은 Schema API를 활용해 `name`이라는 필드 명이 `text_general` 필드 타입을 갖도록 명시

catchall Copy Field 생성

- 색인을 시작하기 전 모든 쿼리를 대상으로 검색 할 필드를 선언 해 주어야한다.
    - 모든 필드의 모든 데이터를 가져와 `_text`라는 필드에 인덱싱 하는 copy field를 정의해 `catchall field`를 설정
    - `curl -X POST -H 'Content-type:application/json' --data-binary '{"add-copy-field" : {"source":"*","dest":"_text_"}}' http://localhost:8983/solr/films/schema`
- 이 설정을 통해 모든 필드의 데이터가 `_text` 필드에 복사
- 이 경우에는 Solr가 모든 색인을 두번씩 하기 때문에 색인이 느려지고 색인 크기가 커지게 된다.
- 프로덕션에서 이렇게 설정하는 건 무리가 있으므로 그때는 복사가 꼭 필요하다고 여겨지는 필드만을 카피

Sample Film 데이터 색인

- films 데이터는 solr 폴더의 `example/films` 경로에 있다.
    - JSON, XML, CSV 세가지 포맷으로 제공
    - 하나의 포맷을 선택 해 films 컬렉션에 색인
    - `java -jar -Dc=films -Dauto example\exampledocs\post.jar example\films\*.json`
- 앞서 설정한 `catachall` 필드가 잘 동작하는지 확인하기 위해 파라미터 q에 `comedy`를 입력해 검색

Faceting

- Solr의 가장 유명한 기능 중 하나
- Faceting을 사용하면 검색 결과를 몇개의 subsets(혹은 buckets or categories)로 분류하거나 각각 subset별 카운트 제공 등이 가능
- faceting의 타입들의 예
    - 필드 값
    - 숫자 혹은 날짜 범위
    - 피봇
    - 임의 쿼리 패싯

Field Facets

- 검색 결과를 제공 할 뿐 아니라 Solr 쿼리는 모든 결과에서 특정 value를 포함하는 document들의 수를 반환할 수도 있다.
- admin UI 쿼리탭에서 facet 체크박스를 클릭 하면 facet과 관련된 옵션들이 보인다.
- 모든 documents `q=*:*` 에서 facet 카운트를 확인 해 보기 위해 `facet=true`로 설정 후 facet 하고 싶은 필드를 `facet.field` 파라미터를 통해 설정
- 만약 도큐먼트 컨텐츠가 필요 없고 facets 만을 원한다면 `rows=0` 으로 설정
    - `curl "http://localhost:8983/solr/films/select?q=*:*&rows=0&facet=true&facet.field=genre_str"`
- `facet.mincount` 파라미터를 설정 하면 특정 갯수 이상이 포함된 facet만을 조회
    - `http://localhost:8983/solr/films/select?facet.field=genre_str&facet.mincount=100&facet=true&indent=true&q.op=OR&q=*:*&rows=0`

Range Facets

- 날짜나 숫자는 각각 하나씩 값을 나누는 것 보다는 특정 범위별로 파티션을 나누는게 도움
- Films 데이터에는 영화의 개봉일이 있기 때문에 날짜 범위별로 facet 할 수 있다.
- Solr Admin UI 자체적으로는 range facet 옵션을 제공하지 않는다.
    - curl 'http://localhost:8983/solr/films/select?q=:&rows=0'\
      '&facet=true'\
      '&facet.range=initial_release_date'\
      '&facet.range.start=NOW-20YEAR'\
      '&facet.range.end=NOW'\
      '&facet.range.gap=%2B1YEAR'`

Pivot Facets

- 결정 트리(Decision trees)라고 알려진 Pivot facets
- Pivot facets은 2개 혹은 그 이상의 필드들을 가능한 모든 조합에 대해 중첩
- Pivot facets은 Films 데이터를 활용해서 Drama 카테고리에 있는 영화 중 특정 감독에 의해 촬영된 영화가 몇개인지를 알아낼 수 있다.
    - `curl "http://localhost:8983/solr/films/select?q=*:*&rows=0&facet=on&facet.pivot=genre_str,directed_by_str"`

나만의 컬렉션 생성

- 초기에 새로운 컬렉션을 생성해야한다.
- 원하는 이름으로 지으면 된다.
- `./bin/solr create -c localDocs -s 2 -rf 2`
    - localDocs 대신 원하는 이름으로 변경하면 된다.
- 위 명령으로 컬렉션을 생성하면 `_default` configset을 사용하게 되며 모든 schemaless 기능들이 제공
- 적절한 schema를 만들어 내기 위해서는 인덱싱 작업을 여러번 반복

색인하는 다양한 방법

- Solr는 데이터 인덱싱을 위해 다양한 방법들을 제공
- 적절한 방법이라고 판단되는걸 골라 접근
- 로컬 파일을 bin/post 를 활용해 색인
    - 파일데이터들을 가지고 있다면, `/bin/post`에 있는 Post Tool 을 사용해서 특정 폴더의 파일들을 색인 할 수 있다.
    - Post Tool은 JSON, XML, CSV 파일 뿐만 아니라 HTML, PDF 뿐만 아니라 MS Office 포맷, Plain text 등 다양한 파일들을 다룰 수 있다.
    - Documents 폴더에 색인할 파일들이 있다고 가정하고, 방금 위에서 만든 localDocs 컬렉션에 색인한다면
        - `./bin/post -c localDocs ~/Documents`
        - 위 명령어를 통해 색인을 진행할 수 있다.
    - 로컬 파일들의 색인을 시도 할 때 오류가 제법 발생
    - field guessing 기능이나 혹은 지원되지 않는 파일 타입을 사용하기 때문일 가능성이 높다.
    - 컨텐츠를 색인 하기 위해서는 데이터를 충분히 이해 하고 시행착오와 에러들을 경험 하며 Solr 색인에 대한 계획을 세울 수 있게 되어야 한다.

DataImportHandler

- Solr는 DataImportHandler(DIH) 라고 불리는 툴을 내장
- 이를 활용하면 데이터베이스나(jdbc driver 필요) 메일 서버, 혹은 다른 데이터 소스들에 접근 가능
- `example/example-DIH` 폴더의 README.txt 파일에 이 도구를 사용 하는 방법이 자세히 작성

Solrj

- Solrj는 자바 기반의 Solr와의 통신을 지원하는 클라이언트
- JVM 기반의 언어를 사용한다면 Solr를 활용
- 다른 프로그램 언어를 이용해 Solr를 사용한다면
  - `https://solr.apache.org/guide/8_11/client-apis.html` 에서 찾아봐야한다.
  - python도 제공하기에 프로젝트에서 사용할 수 있다.

Document Screen

- Admin UI의 Document 탭에 색인하고자 하는 문서를 복사해 붙여 넣거나 혹은 Document Type에서 `Document Builder` 를 선택하면 한번에 하나씩의 필드를 직접 색인 할 수도 있다.

데이터 업데이트

- Solr에서 컨텐츠를 한번 이상 색인하면, 어떠한 중복 데이터도 생기지 않는다.
  - Solr Schema(managed-schema 혹은 schema.xml파일) 가 id라는 특정한 유니크키를 가지고 있기 때문
  - 이미 존재하는 문서와 같은 유니크키로 여러번 POST 추가 요청을 보낸다면 매번 추가 대신 기존 문서를 업데이트
- 이런 사실은 Solr Admin UI의 Core Admin 탭에 작성된 numDocs와 maxDocs를 통해 쉽게 확인 가능
- `numDocs` 는 색인 된 검색 가능한 document의 수를 나타낸다.
  - 몇몇 파일은 하나 이상의 document를 포함 할 수 있기 때문에 파일의 수보다는 보통 많다.
- `maxDoc` 는 numDocs에 이미 삭제 되었지만 아직 물리적으로 삭제되지 않은 색인들을 포함한 값
- 같은 파일을 계속 해서 추가 하다고 해도 numDocs는 늘어나지 않는다.
  - 새로운 document가 계속 기존의 document를 대신하기 때문

데이터 삭제

- document를 삭제하는건 필드 선언에 어떠한 변경도 주지 않는다.
- 그렇기 때문에 필요에 의해 몇몇 필드를 변경 했다면 데이터도 재 색인 할 필요가 있다.
- 특정 문서의 유니크 키 필드 값을 특정 하거나 혹은 여러개의 document 를 선택하는 쿼리로 삭제 할 문서를 정하고, update URL에 삭제 커맨드를 POST 요청해 삭제

- 요청을 적절하게 잘 구성한다면 `bin/post`를 사용해서도 문서를 삭제 할 수 있다.
    - `bin/post -c localDocs -d "<delete><id>SP2514N</id></delete>"`
- 모든 document를 삭제하려면 `delete-by-query` 커맨드를 사용 할 수 있다.
    - `bin/post -c localDocs -d "<delete><query>*:*</query></delete>"`
- 위의 쿼리를 살짝 수정해서, 조건에 맞는 문서들만 삭제 할 수도 있다.

공간 쿼리

- Solr는 정교한 지형 검색도 제공
- 특정 위치로 부터 특정 거리 내의 지역을 검색하거나, 거리 순으로 정렬하거나 혹은 심지어 거리에 따른 검색 결과 Boosting(상위 노출) 옵션도 제공
- techproducts 문서에 위치 정보가 포함 되어 있기 때문에 샌프란시스코에서 10킬로 이내에 있는 아이팟을 검색하는게 가능