대용량 데이터

작 성 일 : 20.04.01
작 성 자 : 도 원 진

1. 대량 데이터 준비

1.1 대량 데이터 설계 기준

- 기준
  - INSERT할 총 게시글 수 = 1,000,000건(100만건)
  - 사용자 수 = 100명

1.1.1 뉴스기사 크롤링

네이버 뉴스기사를 크롤링 해옴. (JSOUP 라이브러리 설치 - maven 저장소이용)

```
<dependency>
            <groupId>org.jsoup</groupId>
            <artifactId>jsoup</artifactId>
            <version>1.13.1</version>
        </dependency>

        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi</artifactId>
            <version>3.17</version>
        </dependency>
```

```
WebCrawling wc = new WebCrawling();
        // 1. URL ����
        String connUrl = "https://news.naver.com/main/ranking/read.nhn?
rankingType=popular_day&oid=081&aid=";
        for(int i=95; i<=100; i++) {
            try {
                Document doc = Jsoup.connect(connUrl+String.format("%010d",
i)).get();
                System.out.println(connUrl+String.format("%010d", i));

                Element title = doc.getElementById("articleTitle");
                Element article = doc.getElementById("articleBodyContents");

                String refinedArticle = article.toString().replace(" <br>",
"\n").replace("\n\n\n", "\n").replace("\n\n", "\n");
                refinedArticle = getText(refinedArticle);
                int idx = refinedArticle.indexOf("\n");
                refinedArticle = refinedArticle.substring(idx+1);
                for(int j=0;j<3;j++) {
                    idx = refinedArticle.lastIndexOf("\n");
```

```
                    if(idx != -1)refinedArticle = refinedArticle.substring(0,
idx);
                }

                //파일 객체 생성
                File file1 = new
File("C:/Users/Bizspring/Desktop/largeData/title"+ i +".txt");
                File file2 = new
File("C:/Users/Bizspring/Desktop/largeData/article"+ i +".txt");

                try {
                  FileWriter fwTitle = new FileWriter(file1);
                  FileWriter fwArticle = new FileWriter(file2);
                  //fwArticle.write(article.toString());
                  fwTitle.write(title.text());
                  fwArticle.write(refinedArticle);
                  fwTitle.close();
                  fwArticle.close();

                } catch (IOException e) {
                  e.printStackTrace();
                }

            } catch (IOException e) {
                // Exp : Connection Fail
                e.printStackTrace();
            }
        }
```

1.1.2 뉴스기사 문장 자르기

- 대용량 데이터를 만들기 위해 뉴스기사 String을 문단별로 분리하고 이후 , 문장별로 분리한다
- 분리 완료된 모든 문장을 List에 보관한다.

```
for(String news : allNews) {
    String [] paragraphs = news.split("\n");
    for(String paragraph : paragraphs) {
        String [] sentences = paragraph.split("\\.");
        for(String sentence : sentences) {
            allSentence.add(sentence);
        }
    }
}
```

1.2 사용자 속성

1.2.1 게시글 작성이 활발한 정도(활동도)

- 100건/1년을 기준으로
- 정규분포확률표의 X축 값을 리턴하는 Random 클래스의 gaussian 메소드 활용

- 최소 -5 ~ 최대 5 의 x값을 리턴
- 빈도수는 정규분포확률에 따라 리턴된다.

1.2.2 사용자 ID

- UUID를 통해 중복없는 ID부여
- java.util.UUID 클래스
    - randomUUID() 메소드를 통해 UUID객체 리턴

1.3 게시글 속성

- Shuffle된 사용자배열에서 게시글 점유율 만큼 게시글 배열의 ArticlVO의 writer필드에 기록
- 월, 일, 시간 순서의 `다중 반복문`을 통해 ArticleVO의 date필드 기록

2. 게시글 INSERT

2.1 INSERT결과

Insert할 게시글 수	묶음 단위	걸린시간	비고
1,000,000	1,000	71.781 s	
1,000,000	200	80.765 s	
1,000,000	100	83.765 s	

2.1.1 게시글 리스트 조회하는데 걸리는 시간

- 셋팅 사항
    - 조회할 게시글 수 : **1,000,000개**

```
select  *
from article
limit #{start}, #{amount}
```

    - 결과 : **3.276 s**

2.1.2 MyISAM vs InnoDB

	MyISAM	InnoDB	비고
SELECT 명령	빠름(READ ONLY)	느림(INSERT, UPDATE 에 빠름)	
트랜젝션	트랜잭션이나 복구등이 필요 없을 경우, 데이터 무결성에 대한 보장이 되지 않음	트랜젝션을 허용, 무결성 보장	
속도	Table-level Lock을 사용하기 때문에 쓰기 작업(INSERT, UPDATE) 속도가 느림	Row-level Lock (행 단위 Lock) 을 사용하기 때문에 변경 작업(INSERT, UPDATE, DELETE)에 대한 속도가 빠름	
구현난이도	별다른 기능이 없으므로 데이터 모델 디자인이 단순	데이터 모델 디자인에는 많은 시간이 필요	
Full-text인덱싱	Full-text 인덱싱이 가능하여 검색하고자 하는 내용에 대한 복합검색이 가능.	Full-text 인덱싱이 불가능	

3. 게시판 통계페이지

3.1 POI 라이브러리

엑셀 다운로드 기능 제공

```
<dependency>
  <groupId>egovframework.rte</groupId>
  <artifactId>egovframework.rte.fdl.excel</artifactId>
  <version>${egovframework.rte.version}</version>
</dependency>

<!-- XML스키마를 자바클래스로 생성하는 라이브러리 -->
<dependency>
    <groupId>org.apache.xmlbeans</groupId>
    <artifactId>xmlbeans</artifactId>
    <version>2.4.0</version>
</dependency>
```

https://offbyone.tistory.com/250 상세코드 참고

3.2 통계차트

3.2.1 c3.js 라이브러리를 통한 차트그리기

- css와 js 라이브러리 받아오기

```
<head>     <!-- stylesheet -->     <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/c3/0.4.11/c3.min.css"/> </head>
<body>     <!-- javascript -->     <script src="https://d3js.org/d3.v3.min.js">
</script>     <script
src="https://cdnjs.cloudflare.com/ajax/libs/c3/0.4.11/c3.min.js"></script>
</body>
```

- 차트가 선택할 데이터 설정

```
var chart = c3.generate({
  bindto: "#linechart",
  data: {
    columns: [
      ['data1', 30, 200, 100, 400, 150, 250],
      ['data2', 50, 20, 10, 40, 15, 25]
    ]
  }
});
```

- html 태그

```
<div id="linechart"></div>
```

- 결과