|  |
| --- |
| FICS |
| Hadoop |
| MapReduce |

|  |
| --- |
| SeokRae Kim  2017-7-31 |

목 차

[**I.** **MapReduce Application 개발** 1](#_Toc489994352)

[**1.** **항공 데이터 분석** 1](#_Toc489994353)

[**2.** **공통 클래스(AirlinePerformanceParser) 구현** 5](#_Toc489994354)

[**3.** **항공 출발 지연 데이터 분석** 5](#_Toc489994355)

[**4.** **항공 도착 지연 데이터 분석** 9](#_Toc489994356)

[**5.** **사용자 정의 옵션 사용** 12](#_Toc489994357)

[**6.** **카운터 사용** 15](#_Toc489994358)

[**7.** **다수의 파일 출력** 17](#_Toc489994359)

[**II.** **정렬** 20](#_Toc489994360)

[**1.** **보조 정렬** 20](#_Toc489994361)

[**2.** **부분 정렬** 24](#_Toc489994362)

[**3.** **전체 정렬** 29](#_Toc489994363)

[**#SequenceFileTotalSort 작성** 29](#_Toc489994364)

1. **MapReduce Application 개발**
   1. **항공 데이터 분석**

* 데이터 다운로드를 위한 셸스크립트 작성
* CSV 파일 HDFS에 업로드
* 공통 클래스(AirlinePerformanceParser) 작성
* 항공 출발 지연 데이터 분석을 위한 매퍼(DepartureDelayCountMapper) 작성
* 매퍼에서 출력한 데이터를 처리하는 리듀서(DelayCountReducer) 작성
* 매퍼와 리듀서를 실행하는 드라이버(DepartureDelayCount) 작성
* 드라이버 실행

|  |
| --- |
| **#데이터 내려 받기** |
| **http://stat-computing.org/dataexpo/2009/**    **#1987년 ~ 2008년까지 총 12GB의 데이터로 하둡을 테스트하기로 한다.** |

|  |
| --- |
| **#경로 만들기** |
| **[root@wikibooks01 hadoop]# mkdir dataexpo** |

|  |
| --- |
| **#다운로드 받기 위한 셸 스크립트 작성** |
| **[root@wikibooks01 dataexpo]# vi downloads.sh**  **#!/bin/sh**  **for((i = 1987; i <= 2008; i++)) ; do**  **wget http://stat-computing.org/dataexpo/2009/$i.csv.bz2**  **bzip2 -d $i.csv.bz2**  **sed -e '1d' $i.csv > $i\_temp.csv**  **mv $i\_temp.csv $i.csv**  **done** |

|  |
| --- |
| **#셸스크립트 실행 (다운 받는데 시간이 오래 걸림)** |
| **[root@wikibooks01 dataexpo]# chmod 755 downloads.sh**  **[root@wikibooks01 dataexpo]# ./downloads.sh** |

|  |
| --- |
| **#다운로드 확인 및 용량 확인** |
| **[root@wikibooks01 dataexpo]# ls –al**  **[root@wikibooks01 dataexpo]# du -h** |

|  |
| --- |
| **#CSV 파일을 HDFS에 업로드 및 확인 – 오래 걸림** |
| **[root@wikibooks01 dataexpo]# hadoop dfs –mkdir airdata**  **[root@wikibooks01 dataexpo]# hadoop dfs –put /home/hadoop/dataexpo/\*.csv airdata**  **[root@wikibooks01 dataexpo]# hadoop dfs –ls** |

지금까지 항공 데이터 분석을 위한 데이터를 모두 다운로드 받았다.

이 데이터를 통해 앞으로 “항공 출발 지연 데이터 분석”과 “항공 도착 지연 데이터 분석”을 실행한다.

이제 데이터 분석을 위해 필요한 프로그램 개발을 설명한다.

* 1. **공통 클래스(AirlinePerformanceParser) 구현**

|  |
| --- |
| **#프로그램 구조** |
| **chapter05**  - 매퍼(Mapper)  - 리듀서(Reducer)  - 드라이버(Driver) **common**  - Parser |

|  |
| --- |
| **#항공 데이터를 파싱 할 클래스(AirlinePerformanceParser) 구현** |
| <https://github.com/SeokRae/hadoop_beginner/tree/SeokRae-branch/src/main/java/wikibooks/hadoop/common> |

* 1. **항공 출발 지연 데이터 분석**
     1. **매퍼(DepartureDelayCountMapper) 구현**

|  |
| --- |
| **#DepartureDelayCountMapper 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DepartureDelayCountMapper.java> |

* + 1. **리듀서(DelayCountReducer) 구현**

|  |
| --- |
| **#DelatCountReducer 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DelayCountReducer.java> |

* + 1. **드라이버(DepartureDelayCount) 클래스 구현**

|  |
| --- |
| **#DepartureDelayCount 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DepartureDelayCount.java> |

* + 1. **드라이버 클래스 실행**

|  |
| --- |
| **#맵리듀스 잡 실행 (10분정도 소모)** |
| 실행 전 Maven build로 jar파일 생성을 잊지 말 것.  **[root@wikibooks01 hadoop]# hadoop jar hadoop-beginner-airportdata-0.0.1-SNAPSHOT.jar wikibooks.hadoop.chapter05.DepartureDelayCount airdata dep\_delay\_count**    매퍼에서 데이터를 받아 리듀서에서 최종적으로 생성한 데이터를 보여주는 핵심 로그  **17/07/31 21:31:22 INFO mapred.JobClient: Map input records=123534969**  **17/07/31 21:31:22 INFO mapred.JobClient: Map output records=50018329**  **17/07/31 21:31:22 INFO mapred.JobClient: Reduce input records=50018329**  **17/07/31 21:31:22 INFO mapred.JobClient: Reduce input groups=255** |

|  |
| --- |
| **#HDFS에 저장된 출력 데이터확인** |
| **[root@wikibooks01 hadoop]# hadoop fs –ls**  **[root@wikibooks01 hadoop]# hadoop fs -cat dep-delay-count/part-r-00000 | head -10**  **[root@wikibooks01 hadoop]# hadoop fs -cat dep-delay-count/part-r-00000 | tail -10** |

|  |
| --- |
| **#엑셀로 시각화하기 위해 CSV 파일로 만들기** |
| **[root@wikibooks01 hadoop]# hadoop fs -get dep\_delay\_count/part-r-00000 dep\_delay\_count.csv**  dep\_delay\_count.csv 파일이 만들어졌는지 확인한다. |

|  |
| --- |
| **#엑셀 피벗차트를 이용해서 데이터 확인** |
| **CentOS에 있는 LibreOffice Calc를 이용해서 데이터를 읽은 후,**  **엑셀프로그램으로 피벗차트를 만들어 보았다.** |

* 1. **항공 도착 지연 데이터 분석**
     1. **매퍼(ArrivalDelayCountMapper) 구현**

|  |
| --- |
| **# ArrivalDelayCountMapper 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/ArrivalDelayCount.java> |

* + 1. **리듀서(DelayCountReducer) 구현**

|  |
| --- |
| **# DelayCountReducer 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DelayCountReducer.java> |

* + 1. **드라이버 클래스 구현**

|  |
| --- |
| **# ArrivalDelayCount 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/ArrivalDelayCount.java> |

* + 1. **드라이버 클래스 실행**

|  |
| --- |
| **#맵 리듀스 실행 (10분정도 소요)** |
| **[root@wikibooks01 hadoop]# hadoop jar hadoop-beginner-airportdata-0.0.1-SNAPSHOT.jar wikibooks.hadoop.chapter05.ArrivalDelayCount airdata arr\_delay\_count**    **17/08/01 21:29:33 INFO mapred.JobClient: Map input records=123534969**  **17/08/01 21:29:33 INFO mapred.JobClient: Map output records=57839171**  **17/08/01 21:29:33 INFO mapred.JobClient: Reduce input records=57839171**  **17/08/01 21:29:33 INFO mapred.JobClient: Reduce output records=255** |

|  |
| --- |
| **#head 와 tail로 데이터 확인** |
| **[root@wikibooks01 hadoop]# hadoop fs -cat arr\_delay\_count/part-r-00000 | head -10**  **[root@wikibooks01 hadoop]# hadoop fs -cat arr\_delay\_count/part-r-00000 | tail -10** |

* + 1. **항공 도착 지연 데이터 차트 만들기**

|  |
| --- |
| **# CSV 파일 만들기** |
| **[root@wikibooks01 hadoop]# hadoop fs -get arr\_delay\_count/part-r-00000 arr \_delay\_count.csv** |

|  |
| --- |
| **#엑셀 피벗차트를 이용해서 데이터 확인** |
| **CentOS에 있는 LibreOffice Calc를 이용해서 데이터를 읽은 후,**  **엑셀프로그램으로 피벗차트를 만들어 보았다.** |

* 1. **사용자 정의 옵션 사용**
* 사용자가 정의한 파라미터를 통해 **도착** 혹은 **출발** 지연 데이터를 분석
  + 1. **매퍼(DelayCountMapper) 구현**

|  |
| --- |
| **# DelayCountMapper 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DelayCountMapper.java> |

* + 1. **리듀서 구현**

|  |
| --- |
| **#** **DelayCountReducer 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DelayCountReducer.java> |

* + 1. **드라이버 클래스 구현**

|  |
| --- |
| **# DelayCount 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DelayCount.java> |

* + 1. **드라이버 클래스 실행**
       1. **workType = departure**

|  |
| --- |
| **#departure 파라미터를 이용해 출발 데이터를 분석** |
| **[root@wikibooks01 hadoop]# hadoop jar hadoop-beginner-airportdata-0.0.1-SNAPSHOT.jar wikibooks.hadoop.chapter05.DelayCount -D workType=departure airdata departure\_delay\_count** |

|  |
| --- |
| **# 결과물 확인** |
| **3. 항공 출발 지연 데이터 분석과 같은 결과 인지를 확인**  **17/08/02 15:54:50 INFO mapred.JobClient: Map input records=123534969**  **17/08/02 15:54:50 INFO mapred.JobClient: Map output records=50018329**  **17/08/02 15:54:50 INFO mapred.JobClient: Reduce input records=50018329**  **17/08/02 15:54:50 INFO mapred.JobClient: Reduce output records=255** |

|  |
| --- |
| **#HDFS에 저장된 데이터 확인** |
| **[root@wikibooks01 hadoop]# hadoop fs -cat departure\_delay\_count/part-r-00000 | head -10**  **[root@wikibooks01 hadoop]# hadoop fs -cat departure\_delay\_count/part-r-00000 | tail -10** |

* + - 1. **workType = arrival**

|  |
| --- |
| **#arrival 파라미터를 이용해 도착 지연 데이터 분석** |
| **[root@wikibooks01 hadoop]# hadoop jar hadoop-beginner-airportdata-0.0.1-SNAPSHOT.jar wikibooks.hadoop.chapter05.DelayCount -D workType=arrival airdata arrival\_delay\_count**    **4. 항공 도착 지연 데이터 분석의 결과와 동일한지 확인**  **17/08/02 16:17:59 INFO mapred.JobClient: Map input records=123534969**  **17/08/02 16:17:59 INFO mapred.JobClient: Map output records=57839171**  **17/08/02 16:17:59 INFO mapred.JobClient: Reduce input records=57839171**  **17/08/02 16:17:59 INFO mapred.JobClient: Reduce output records=255** |

* 1. **카운터 사용**
* **Job 실행 결과 메시지와 웹 인터페이스에서 간편하게 데이터를 확인하기 위한 작업**
  + 1. **사용자 정의 카운터 구현**

|  |
| --- |
| **# DelayCounters 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DelayCounters.java> |

* + 1. **매퍼 구현**

|  |
| --- |
| **# DelayCountMapperWithCounter 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DelayCountMapperWithCounter.java> |

* + 1. **드라이버 클래스 구현**

|  |
| --- |
| **# DelayCountWithCounter 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DelayCountWithCounter.java> |

* + 1. **드라이버 클래스 실행**

|  |
| --- |
| **# 맵리듀스 실행하기** |
| **[root@wikibooks01 hadoop]# hadoop jar hadoop-beginner-airportdata-0.0.1-SNAPSHOT.jar wikibooks.hadoop.chapter05.DelayCountWithCounter -D workType=departure airdata departure\_delay\_count\_counter**    **도착 지연을 확인할 수 없는 경우 2,302, 136건**  **스케줄 대로 도착한 경우 26,416,798건**  **스케줄 보다 일찍 도착한 경우 44,797,706건** |

|  |
| --- |
| **#맵리듀스 잡 관리자용 웹 화면** |
| **DelayCounters 확인** |

* 1. **다수의 파일 출력**
     1. **MultipleOutputs 이해**

MultipleOutputs는 여러 개의 출력 데이터를 쉽게 생성하도록 돕는 기능을 제공합니다. MultipleOutputs는 여러 개의 OutputCollectors를 만들고 각 OutputCollectors에 대한 출력 경로, 출력 포맷, 키와 값 유형을 설정합니다.

* + 1. **매퍼 구현**

|  |
| --- |
| **#** **DelayCountMapperWithMultipleOutputs 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DelayCountMapperWithMultipleOutputs.java> |

* + 1. **리듀서 구현**

|  |
| --- |
| **# DelayCountReducerWithMultipleOutputs 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DelayCountReducerWithMultipleOutputs.java> |

* + 1. **드라이버 클래스 구현**

|  |
| --- |
| **# DelayCountWithMultipleOutputs 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter05/DelayCountWithMultipleOutputs.java> |

* + 1. **드라이버 클래스 실행**

|  |
| --- |
| **#전체 카운터 정보를 출력** |
| **[root@wikibooks01 hadoop]# hadoop jar hadoop-beginner-airportdata-0.0.1-SNAPSHOT.jar wikibooks.hadoop.chapter05.DelayCountWithMultipleOutputs airdata delay\_count\_mos** |

|  |
| --- |
| **#출력 데이터 확인** |
| **[root@wikibooks01 hadoop]# hadoop fs -ls delay\_count\_mos**  **arrival-r-00000, departure-r-00000 파일 확인**    **\*\* part-r-00000 파일의 크기가 0인 이유는 reduce 메서드에서 context 객체에 write를 하지 않았기 때문이다.** |

|  |
| --- |
| **#출발 지연 데이터 확인** |
| **[root@wikibooks01 hadoop]# hadoop fs –cat delay\_count\_mos/departure-r-00000 | head -10**  **[root@wikibooks01 hadoop]# hadoop fs –cat delay\_count\_mos/departure-r-00000 | tail-10** |

|  |
| --- |
| **#도착 지연 데이터 확인** |
| **[root@wikibooks01 hadoop]# hadoop fs –cat delay\_count\_mos/arrival-r-00000 | head -10**  **[root@wikibooks01 hadoop]# hadoop fs –cat delay\_count\_mos/arrival-r-00000 | tail-10** |

1. **정렬**

* 항공 데이터 분석 프로그램에 보조 정렬을 적용
* 부분 정렬과 전체 정렬을 이용해 단순하게 입력 데이터만 정렬하는 맵리듀스 프로그램 개발
  1. **보조 정렬**
* 키의 값들을 그룹핑하고, 그룹핑된 레코드에 순서를 부여하는 방식

|  |
| --- |
| **#보조 정렬의 알고리즘** |
| 1. 기존 키의 값들을 조합한 복합 키(Composite Key)를 정의, 이때 키의 값 중에서 어떤 키를 그룹핑 키로 사용할지 결정 2. 복합키의 레코드를 정렬하기 위한 비교기(Comparator)를 정의 3. 그룹핑 키를 파티셔닝할 파티셔너(Partitioner)를 정의 4. 그룹핑 키를 정렬하기 위한 비교기(Comparator)를 정의 |

* + 1. **복합키 구현**

|  |
| --- |
| **# DataKey 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter06/DateKey.java> |

* + 1. **복합키 비교기 구현**

|  |
| --- |
| **# DataKeyComparator 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter06/DateKeyComparator.java> |

* + 1. **그룹키 파티셔너 구현**

|  |
| --- |
| **# GroupKeyPartitioner 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter06/GroupKeyPartitioner.java> |

* + 1. **그룹키 비교기 구현**

|  |
| --- |
| **# GroupKeyComparator 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter06/GroupKeyComparator.java> |

* + 1. **매퍼 구현**

|  |
| --- |
| **# DelayCountMapperWithDateKey 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter06/DelayCountMapperWithDateKey.java> |

* + 1. **리듀서 구현**

|  |
| --- |
| **# DelayCountReducerWithDateKey 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter06/DelayCountReducerWithDateKey.java> |

* + 1. **드라이버 구현**

|  |
| --- |
| **# DelayCountWithDateKey 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter06/DelayCountWithDateKey.java> |

* + 1. **드라이버 실행**

|  |
| --- |
| **# 드라이버 클래스 실행** |
| **[root@wikibooks01 hadoop]# hadoop jar hadoop-beginner-sort-0.0.1-SNAPSHOT.jar wikibooks.hadoop.chapter06.DelayCountWithDateKey airdata delay\_count\_sort**    **17/08/08 20:00:49 INFO mapred.JobClient: Reduce input groups=44**  **리듀서의 입력 데이터 그룹이 510개였는데, 44개만 출력되어 있음을 확인**  **왜냐하면 보조 정렬에서는 연도만 그룹 처리를 했기 때문에 1987 ~ 2008년까지 총 22개월로 출발 지연과 도착 지연에 대한 두 개의 데이터 셋을 만들어서 44개 되었다.** |

|  |
| --- |
| **# 출력 데이터 조회** |
| **[root@wikibooks01 hadoop]# hadoop fs –ls delay\_count\_sort**    **HDFS에 저장된 출력데이터 조회 시  출발 지연(departure-r-00000)과 도착 지연(arrival-r-00000)에 대한  처리 데이터가 모두 생성 되었음을 확인** |

|  |
| --- |
| **# 도착 지연 데이터 head, tail 데이터 조회** |
| **[root@wikibooks01 hadoop]# hadoop fs –cat delay\_count\_sort/arrival-r-00000 | head -10**  **[root@wikibooks01 hadoop]# hadoop fs –cat delay\_count\_sort/arrival-r-00000 | tail -10** |

|  |
| --- |
| **# 출발 지연 데이터 head, tail 데이터 조회** |
| **[root@wikibooks01 hadoop]# hadoop fs –cat delay\_count\_sort/departure-r-00000 | head -10**  **[root@wikibooks01 hadoop]# hadoop fs –cat delay\_count\_sort/ departure-r-00000 | tail -10** |

* 1. **부분 정렬**
* 매퍼의 출력 데이터를 맵파일(MapFile)로 변경해 데이터를 검색하는 방법
* 데이터 검색 시 주로 사용 됨

|  |
| --- |
| **#항공 지연 통계 데이터를 항공 운항 거리 순서대로 정렬하는 부분정렬 프로그램** |
| 1. 입력 데이터를 시퀀스 파일로 생성 2. 시퀀스 파일을 맵파일로 변경 3. 맵파일에서 데이터를 검색 |

* org.apache.hadoop.mapred 패키지를 이용해 맵리듀스 프로그램을 개발해야 한다.
  + 1. **시퀀스파일 생성**

|  |
| --- |
| **# SequenceFileCreator 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter06/SequenceFileCreator.java> |

|  |
| --- |
| **#2008년 통계 데이터로 맵리듀스 잡을 실행** |
| **[root@wikibooks01 hadoop]# hadoop jar hadoop-beginner-sort-0.0.1-SNAPSHOT.jar wikibooks.hadoop.chapter06.SequenceFileCreator airdata/2008.csv 2008\_sequencefile**    **매퍼는 입력 데이터를 연산하지 않기 때문에 리듀서는 필요하지 않다.** |

|  |
| --- |
| **# 출력 폴더에 시퀀스파일이 만들어져 있는지 확인** |
| **[root@wikibooks01 hadoop]# hadoop fs -ls 2008\_sequencefile** |

|  |
| --- |
| **#text 옵션을 이용해 시퀀스파일의 10개 라인을 확인** |
| **[root@wikibooks01 hadoop]# hadoop fs -text 2008\_sequencefile/part-00000 | head -10**     |  | | --- | | **810 2008,1,3,4,2003,1955,2211,2225,WN,335,N712SW,128,150,116,-14,8,IAD,TPA,810,4,8,0,,0,NA,NA,NA,NA,NA**  **810 2008,1,3,4,754,735,1002,1000,WN,3231,N772SW,128,145,113,2,19,IAD,TPA,810,5,10,0,,0,NA,NA,NA,NA,NA**  **515 2008,1,3,4,628,620,804,750,WN,448,N428WN,96,90,76,14,8,IND,BWI,515,3,17,0,,0,NA,NA,NA,NA,NA**  **515 2008,1,3,4,926,930,1054,1100,WN,1746,N612SW,88,90,78,-6,-4,IND,BWI,515,3,7,0,,0,NA,NA,NA,NA,NA**  **515 2008,1,3,4,1829,1755,1959,1925,WN,3920,N464WN,90,90,77,34,34,IND,BWI,515,3,10,0,,0,2,0,0,0,32**  **688 2008,1,3,4,1940,1915,2121,2110,WN,378,N726SW,101,115,87,11,25,IND,JAX,688,4,10,0,,0,NA,NA,NA,NA,NA**  **1591 2008,1,3,4,1937,1830,2037,1940,WN,509,N763SW,240,250,230,57,67,IND,LAS,1591,3,7,0,,0,10,0,0,0,47**  **1591 2008,1,3,4,1039,1040,1132,1150,WN,535,N428WN,233,250,219,-18,-1,IND,LAS,1591,7,7,0,,0,NA,NA,NA,NA,NA**  **451 2008,1,3,4,617,615,652,650,WN,11,N689SW,95,95,70,2,2,IND,MCI,451,6,19,0,,0,NA,NA,NA,NA,NA**   1. **2008,1,3,4,1620,1620,1639,1655,WN,810,N648SW,79,95,70,-16,0,IND,MCI,451,3,6,0,,0,NA,NA,NA,NA,NA** |   **데이터의 키 순서가 810 -> 515 -> 688 -> 1591로 표시되었는지 확인**  **-> 맵리듀스는 입력 데이터의 키를 기준으로 정렬되기 때문에 줄 번호를 기준으로 출력** |

* + 1. **맵파일 생성**
* **맵파일은 키 값을 검색할 수 있게 색인과 함께 정렬된 시퀀스 파일이다.**

|  |
| --- |
| **# MapFileCreator 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter06/MapFileCreator.java> |

|  |
| --- |
| **#시퀀스파일의 압축 포맷과 입출력 데이터 경로를 설정하고 잡을 실행** |
| **[root@wikibooks01 hadoop]# hadoop jar hadoop-beginner-sort-0.0.1-SNAPSHOT.jar wikibooks.hadoop.chapter06.MapFileCreator 2008\_sequencefile 2008\_mapfile** |

|  |
| --- |
| **#2008\_map\_file 폴더 조회** |
| **part-00000 디렉터리가 생성되어 있음을 확인**  **[root@wikibooks01 hadoop]# hadoop fs -ls 2008\_mapfile** |

|  |
| --- |
| **#part-00000 폴더 조회** |
| **맵파일의 규격에 맞게 data와 index 파일이 생성되어 있음을 확인**  **[root@wikibooks01 hadoop]# hadoop fs -ls 2008\_mapfile/part-00000** |

|  |
| --- |
| **#data 파일에 출력한 키와 값이 저장되어 있는지 확인** |
| **[root@wikibooks01 hadoop]# hadoop fs -text 2008\_mapfile/part-00000/data | head -10**     |  | | --- | | **11 2008,8,10,7,1315,1220,1415,1320,OH,5572,N819CA,60,60,14,55,55,JFK,LGA,11,8,38,0,,0,55,0,0,0,0**  **11 2008,5,15,4,2037,1800,2125,1900,OH,4988,N806CA,48,60,31,145,157,JFK,LGA,11,10,7,0,,0,145,0,0,0,0**  **17 2008,3,8,6,NA,1105,NA,1128,AA,1368,,NA,23,NA,NA,NA,EWR,LGA,17,NA,NA,1,B,0,NA,NA,NA,NA,NA**  **21 2008,5,9,5,48,100,117,130,AA,588,N061AA,29,30,11,-13,-12,MIA,FLL,21,6,12,0,,0,NA,NA,NA,NA,NA**  **21 2008,2,8,5,NA,1910,NA,1931,AA,1668,,NA,21,NA,NA,NA,FLL,MIA,21,NA,NA,1,A,0,NA,NA,NA,NA,NA**  **24 2008,3,12,3,955,931,1021,948,9E,2009,91619E,26,17,10,33,24,IAH,HOU,24,7,9,0,,0,0,0,9,0,24**  **24 2008,1,2,3,1245,1025,1340,1125,OH,5610,N806CA,55,60,11,135,140,IAD,DCA,24,5,39,0,,0,135,0,0,0,0**  **24 2008,11,27,4,943,940,1014,956,9E,5816,91469E,31,16,9,18,3,IAH,HOU,24,5,17,0,,0,0,0,18,0,0**  **28 2008,2,22,5,2046,2050,NA,2156,OO,3698,N298SW,NA,66,NA,NA,-4,SLC,OGD,28,NA,12,0,,1,NA,NA,NA,NA,NA**  **30 2008,12,31,3,NA,1800,NA,1820,OO,5613,N579SW,NA,20,NA,NA,NA,SFO,SJC,30,NA,NA,1,B,0,NA,NA,NA,NA,NA** |   **운항거리가 11 -> 17 -> 21 -> 24 -> 28 -> 30의 순서대로 정렬되어 있음을 확인** |

* + 1. **검색 프로그램 구현**
* 맵파일에서 원하는 키에 해당하는 값을 검색하는 방법
* 검색하고자 하는 키가 속하는 파티션 번호를 조회한 후, 파티션 번호로 맵파일에 접근해 데이터를 검색

|  |
| --- |
| **#SearchValueList 작성** |
| <https://github.com/SeokRae/hadoop_beginner/blob/SeokRae-branch/src/main/java/wikibooks/hadoop/chapter06/SearchValueList.java> |

|  |
| --- |
| **#운항 거리가 100마일인 데이터 목록을 조회** |
| **[root@wikibooks01 hadoop]# hadoop jar hadoop-beginner-sort-0.0.1-SNAPSHOT.jar wikibooks.hadoop.chapter06.SearchValueList 2008\_mapfile 100**    **파일을 찾을 수 없다는 에러가 발생하는 것이 정상**  **이유는 MapFileOutputFormat의 메서드는 파라미터로 전달받은 폴더에 있는 모든 폴더에서 맵파일이 존재하는지 체크를 한다.**  **2008\_mapfile 폴더에는 맵리듀스 잡에서 생성한 로그 파일 폴더가 있는데, MapFileOutputFormat이 로그 폴더를 체크해서 맵파일이 없다는 오류를 발생시킨 것이다.**  **검색 프로그램이 제대로 동작하기 위해서는 로그 폴더를 모두 삭제 한다.** |

|  |
| --- |
| **#로그 폴더 삭제** |
| **[root@wikibooks01 hadoop]# hadoop fs -rmr 2008\_mapfile/\_\*** |

|  |
| --- |
| **#검색 폴더 실행** |
| **[root@wikibooks01 hadoop]# hadoop jar hadoop-beginner-sort-0.0.1-SNAPSHOT.jar wikibooks.hadoop.chapter06.SearchValueList 2008\_mapfile 100 | head -10**    **운항 거리가 100마일인 데이터가 정상적으로 출력**  **특정 키에 해당하는 데이터만 검색해서 사용하는 경우 부분 정렬을 활용하는 예제 결과 확인** |

* 1. **전체 정렬**

|  |
| --- |
| **#SequenceFileTotalSort 작성** |
|  |

|  |
| --- |
| **#** |
|  |