데이터 엔지니어링 테이블 수집

관계형 데이터베이스 수집 도구 스쿱

Park Suhyuk

psyoblade

psyoblade

apache **sqoop** 1.4.7
apache **hadoop** 2.9.0
docker 19.03.8 CE
ubuntu 20.04 LTS

NCSOFT®

목차

About Apache Sqoop

관계형 데이터베이스 수집 도구 아파치 스쿱

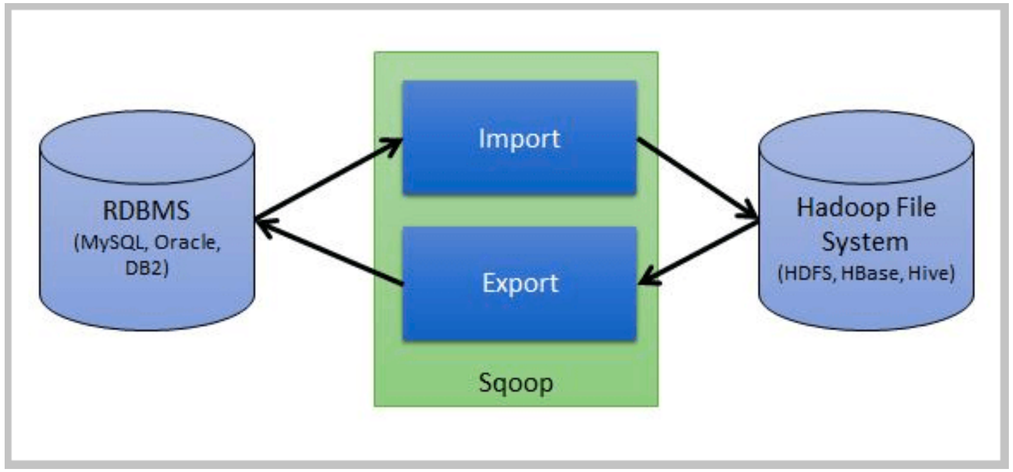
아파치 스쿱 제공 기능

What is 'Apache Sqoop'?

아파치 스쿱은 관계형 데이터베이스로 부터 하둡 클러스터(HDFS) 데이터를 전송하거나, 반대로 하둡 클러스터에 존재하는 데이터를 관계형 데이터베이스로전송하는 데에 필요한 오픈 소스 도구입니다. 가장 널리 사용 되고 있는 테이블 수집과 적재를 위해 고안된 도구입니다. 스쿱의 경우 Hadoop 분산 처리 프레임워크인 맵 리듀스(MapReduce) 기법을 사용하여 수행 되므로, 하둡 클러스터가 존재한다면 추가적인 운영비용이 발생하지 않으며, 하둡 클러스터가 제공하는 내결함성 및 병렬 수행 등의 이점을 얻을 수 있습니다. 현재 Sqoop Version 1 과 Version 2 로 분리된 프로젝트로 제공되지만 수집 기능을 담당하는 코어는 동일하며 해당 서비스를 제공하는 외부 시스템 아키텍처에서 차이가 있습니다.

주요 기능으로는 테이블을 수집 및 적재하는 기능과 전체 테이블을 가져오거나 혹은 증분 데이터만 가져올 수 있으며, 대부분의 상용 데이터베이스의 커넥터를 제공하며, (대상 JDBC Jar 파일은 별도로 추가 해야만 합니다) 질의문 혹은 테이블 명으로 수집할 수 있는 다양한 기능을 제공합니다.

https://www.hdfstutorial.com/sqoop-architecture/



APACHE SQOOP

아파치 스쿱 제공 기능

Sqoop Tools

```
$ sqoop help
usage: sqoop COMMAND [ARGS]
$ sqoop eval -e "select my_column from my_table"
```

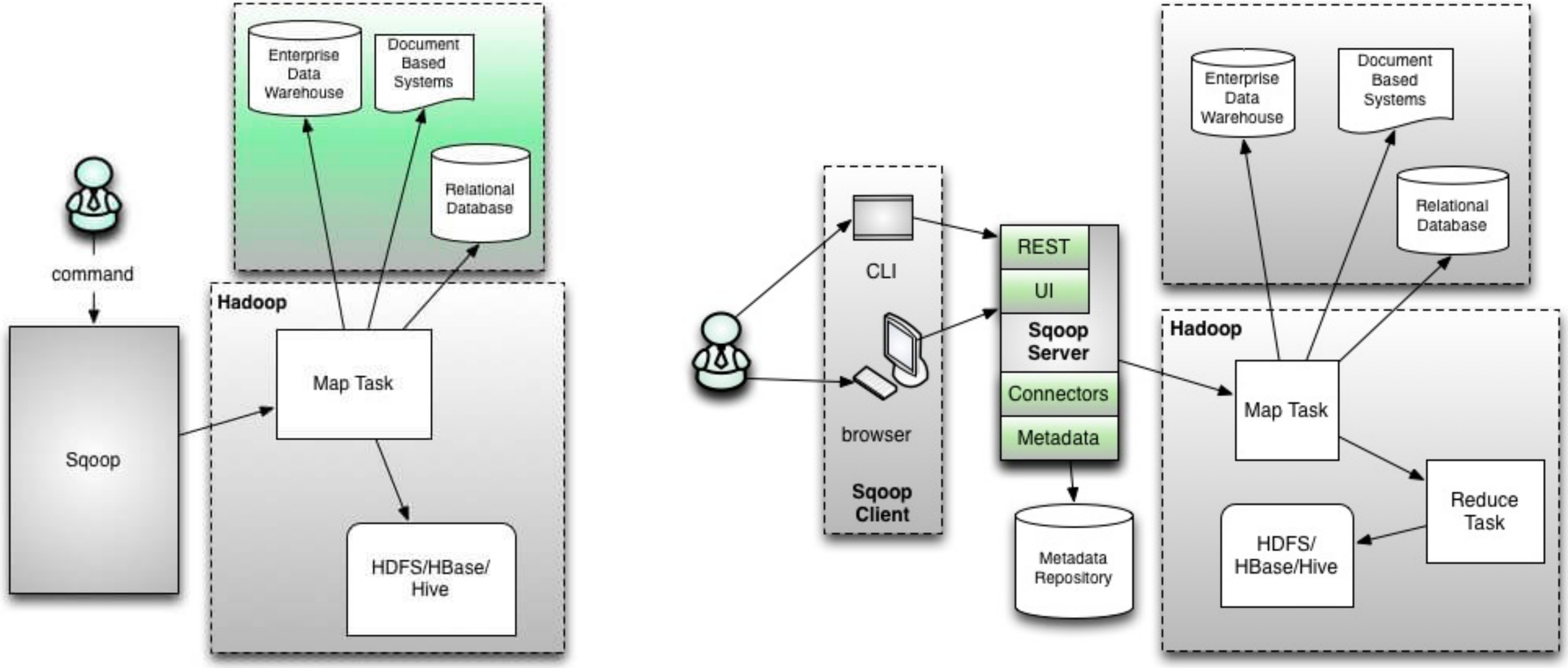
https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html#_sqoop_tools

커맨드	설명
codegen	데이터베이스 레코드를 가져오기 위한 MapReduce 수행을 위한 Java 코드 및 Jar 파일을 생성합니다
eval	SQL 구문을 수행하고 결과를 출력합니다
export	HDFS 경로에 저장되어 있는 데이터를 데이터베이스 테이블로 익스포트 합니다
help	도움말을 출력합니다
import	데이터베이스 테이블을 HDFS 에 임포트 합니다
import-all-tables	데이터베이스의 대상 테이블 들을 HDFS 에 임포트 합니다
list-databases	데이터베이스 목록을 출력합니다
list-tables	테이블 목록을 출력합니다

아파치 스쿱 1 vs. 2

Apache Sqoop version 1 vs. 2

아파치 스쿱은 2가지 버전으로 개발되고 있으나 코어 엔진은 거의 동일하지만, Sqoop 1 에서의 메타데이터 관리의 번거로움 그리고 외부 서비스와의 연동이 어려운 점을 개선하여 메타데이터 관리를 위해 추가된 메타데이터 저장소와 외부 연동을 위한 API 서버가 추가 되었다



아파치 스쿱 1 vs. 2

Apache Sqoop version 1 vs. 2

별도의 메타데이터 저장소를 가지는 것이 경우에 따라서 레거시 시스템과의 연동이나 통합에 어려움을 줄 수도 있으므로, 항상 바람직하다고 보기는 어려우며, 다만 Oozie 혹은 REST API 를 통해 독립적인 서비스를 빠르게 연동하고자 하는 경우 Sqoop v2 를 선택하는 것이 적절하며, 별도의 서버 운영이 부담스럽거나 충분히 작은 수준의 테이블을 관리하는 경우에는 클라이언트 모듈 설치로 바로 적용이 가능한 Sqoop v1 이 더 나은 선택으로 판단됩니다

구분	Sqoop v1	Sqoop v2
작업 제출	Sqoop Client	Sqoop Server
클라이언트	작업 수행을 위해 항상 Sqoop Client 설치를 해야만 한다	REST API 호출을 위한 Light-weight Client 만 필요하다 (Web UI)
직접 호출	라이브러리 수준에서 연동 및 Sqoop Runner 통한 제출이 가능	REST API 를 통해서만 연동이 가능
메타데이터	별도로 관리되는 메타데이터 정보는 없음	별도의 메타데이터 레포지토리 관리가 됩니다 (일장일단)
서버 운영	클라이언트 라이브러리 통한 수행으로 별도의 서버가 없음	메타데이터 저장소 및 API 서버 운영이 필요합니다
외부 연동	Java 로 직접 개발하거나 Command Line 연동만 가능	REST API 통해 연동이 유연하고, Oozie 등과 쉽게 연동이 가능

How to run apache sqoop

도커 컴포즈를 이용한 아파치 스쿱 실행

아파치 스쿱 실행

git clone https://github.com/psyoblade/data-engineer-intermediate-training.git

```
$ git clone https://github.com/psyoblade/data-engineer-intermediate-training.git
$ cd data-engineer-intermediate-training/day2
$ docker-compose pull
$ docker-compose up -d
$ docker-compose exec sqoop bash
$ sqoop import -jt local -m 1 --connect jdbc:mysql://mysql:3306/testdb \
  --table seoul_popular_trip --target-dir file:///tmp/seoul_popular_trip \
  --username sqoop --password sqoop
```

1. 도커 컴포즈를 통해 MySQL 과 Sqoop 이 설치된 컨테이너가 기동됩니다.
2. MySQL 서버 기동 시에 testdb.seoul_popular_trip 테이블이 자동으로 생성됩니다
3. compose exec 명령을 통해 컨테이너로 접속하여 sqoop 명령어을 통해 테이블 수집을 합니다

```
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=498416
21/12/11 13:20:27 INFO mapreduce.ImportJobBase: Transferred 0 bytes in 2.2492 seconds (0 bytes/sec)
21/12/11 13:20:27 INFO mapreduce.ImportJobBase: Retrieved 1956 records.
```

```yaml
version: "3"

services:
  mysql:
    container_name: day1_mysql
    image: psyoblade/data-engineer-intermediate-day1-mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: testdb
      MYSQL_USER: sqoop
      MYSQL_PASSWORD: sqoop
    ports:
      - '3306:3306'
    volumes:
      - $PROJECT_HOME/mysql/custom:/etc/mysql/conf.d
  sqoop:
    container_name: day1_sqoop
    image: psyoblade/sqoop-hive:2.3.3
    tty: true
    ports:
      - '8088:8088'
      - '10020:10020'
      - '19888:19888'
      - '50070:50070'
      - '50075:50075'
    volumes:
      - $PROJECT_HOME/jars:/jdbc
```

테이블 임포트 - 리모트 + 리소스 매니저

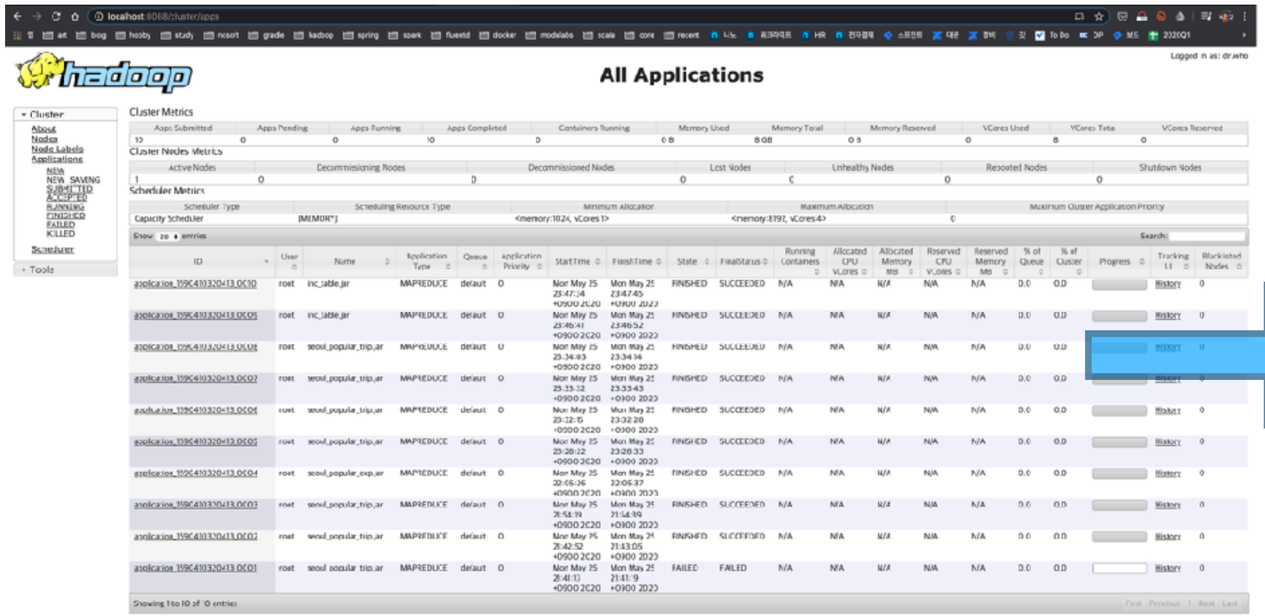
sqoop import + http://resource-manager:8088

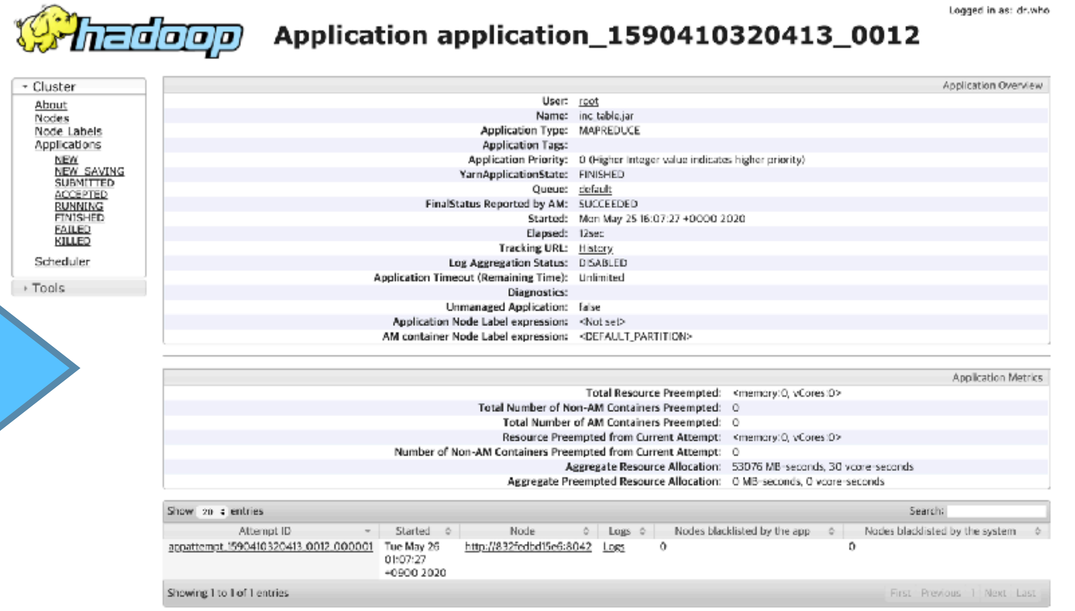
하둡 리소스 매니저를 통해 분산 저장소에 테이블 수집을 수행합니다

$ docker exec -it day1_sqoop sqoop import -m 1 --connect jdbc:mysql://day1_mysql:3306/testdb --table seoul_popular_trip --delete-target-dir --target-dir /user/sqoop/target/seoul_popular_trip --fields-terminated-by '\t' --verbose --username sqoop --password sqoop

원격지에 수집되고 있는 작업을 Hadoop Resource Manager 화면을 통해 확인합니다

http://localhost:8088/cluster/apps



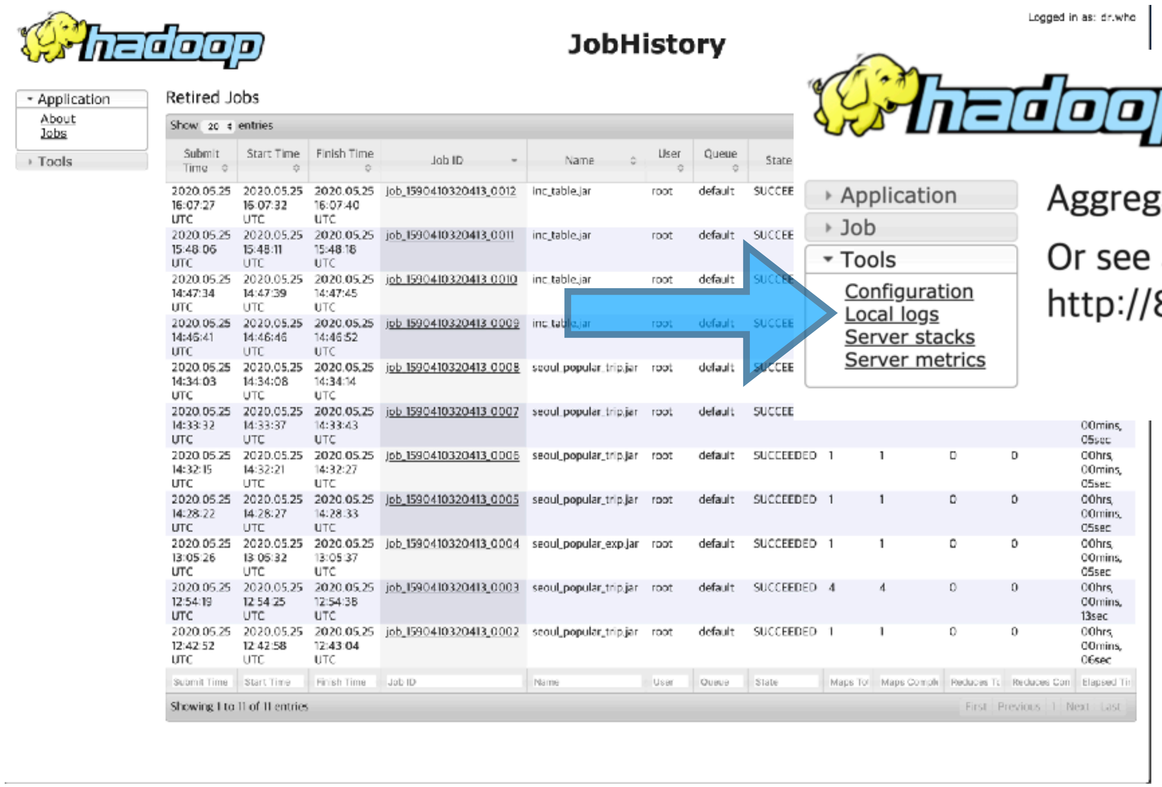


테이블 임포트 – 리모트 + 히스토리 서버

sqoop import + http://history-server:19888

하둡의 경우 작업이 RUNNING 중인 경우에는 Resource Manager 에서 확인이 가능하지만, 작업이 완료되면 History Server 로 이동 됩니다.

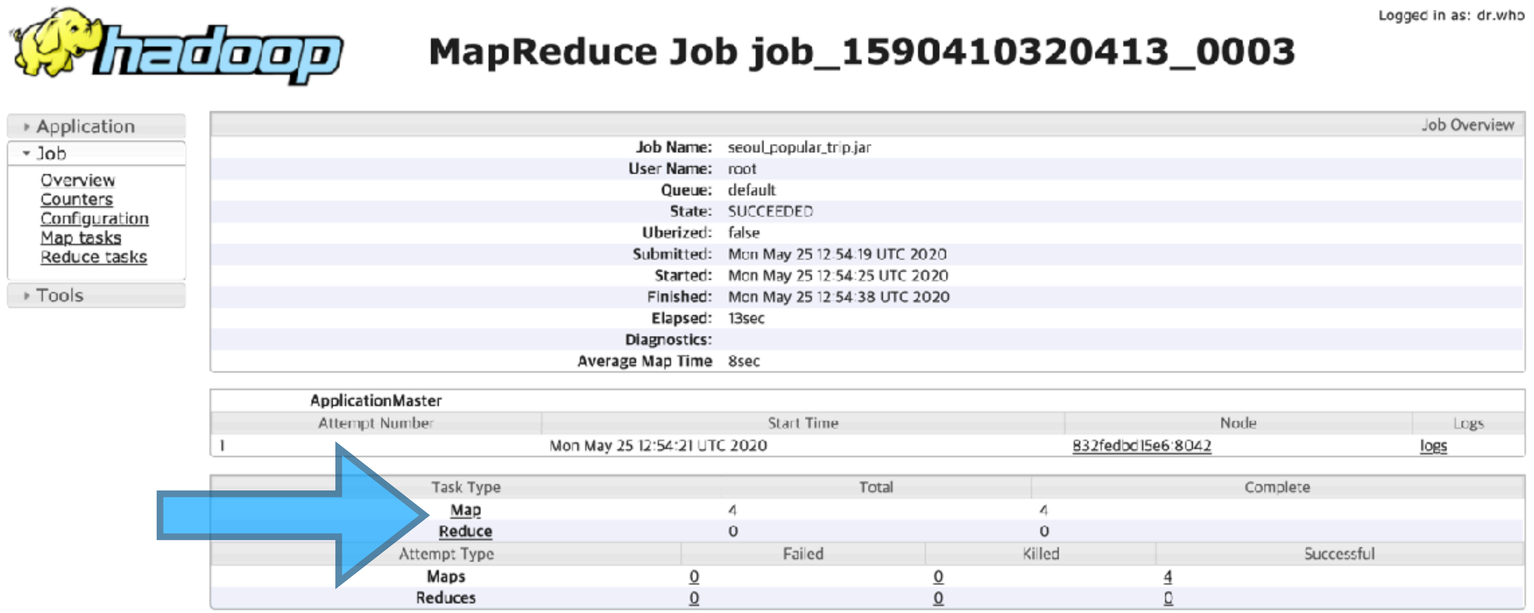
http://localhost:19888/jobhistory



테이블 임포트 – 병렬 수행

sqoop import -m <num-mappers> --split-by <column>

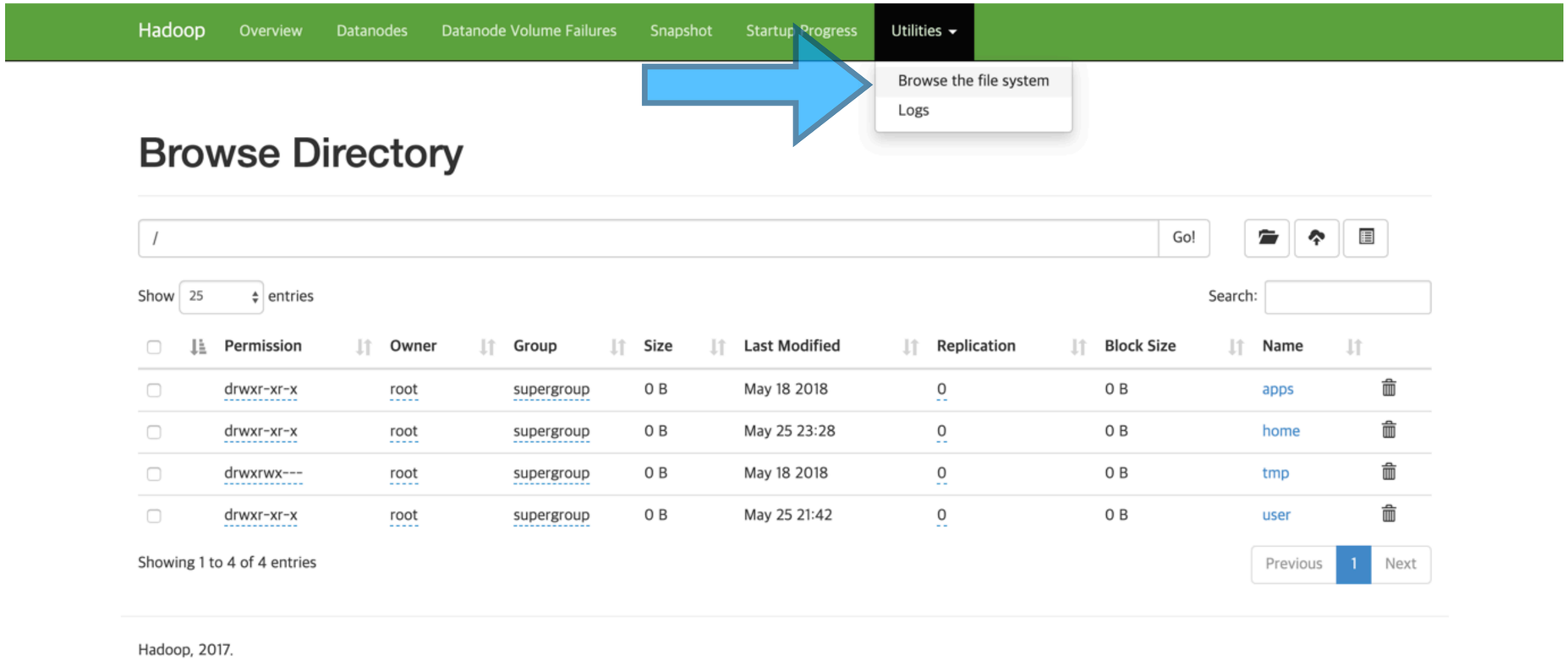
```
# 4개의 Task 가 병렬로 테이블을 수집하며, 특정 필드의 값을 기준으로 min(id), max(id) 를 통해 4개의 mapper 가 수행됩니다
$ docker exec -it day1_sqoop sqoop import -m 4 --split-by id --connect jdbc:mysql://day1_mysql:3306/testdb --table seoul_popular_trip --target-dir /home/sqoop/target/seoul_popular_trip --fields-terminated-by '\t' --verbose --username sqoop --password sqoop
# 수행된 작업이 Mapper 가 4개가 떠서 수행 되었는지, 그리고 어떻게 조회 되었는지 확인합니다
```



테이블 임포트 – 리모트 + 네임노드

sqoop import + http://namenode:50070

```
# 적재가 완료된 파일을 터미널 뿐만 아니라 웹 UI 를 통해서 확인할 수 있습니다
http://localhost:50070/explorer.html
```



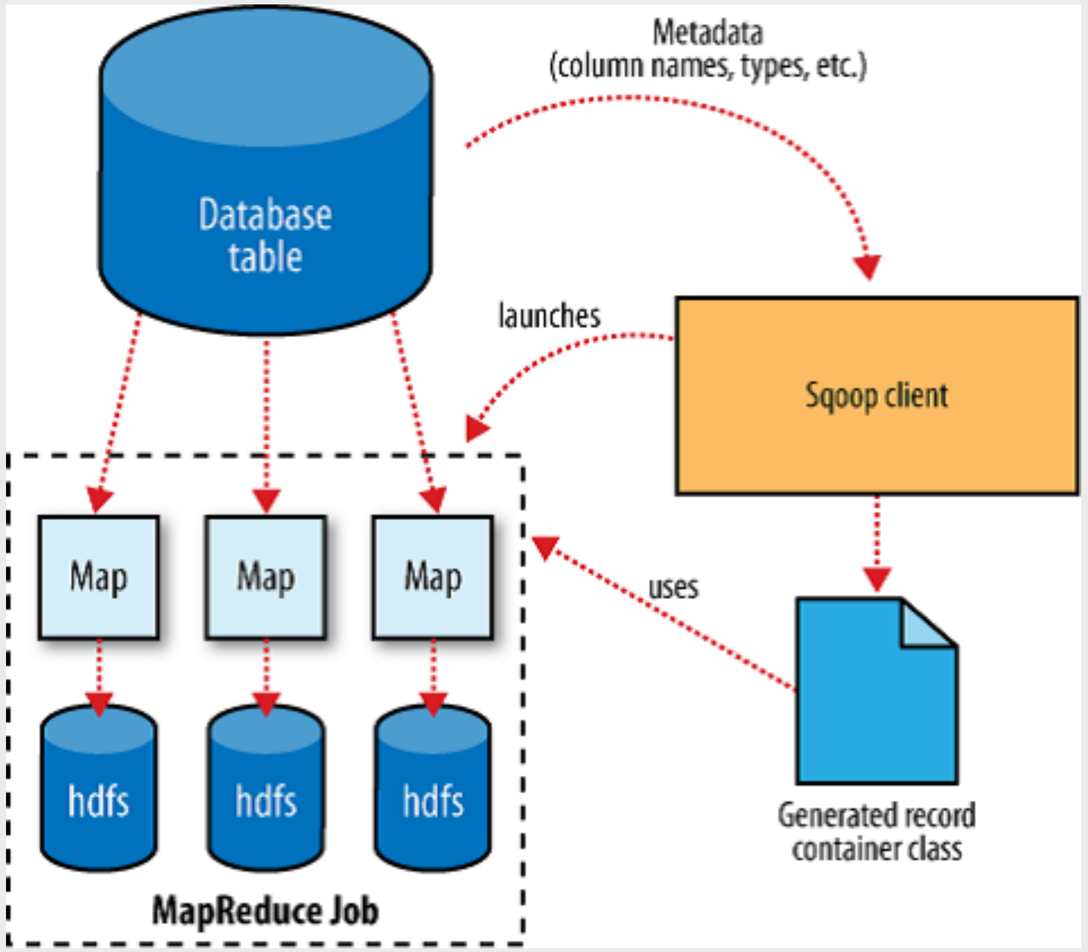
Apache Sqoop Internal

아파치 스쿱 내부구조 및 동작 방식의 이해

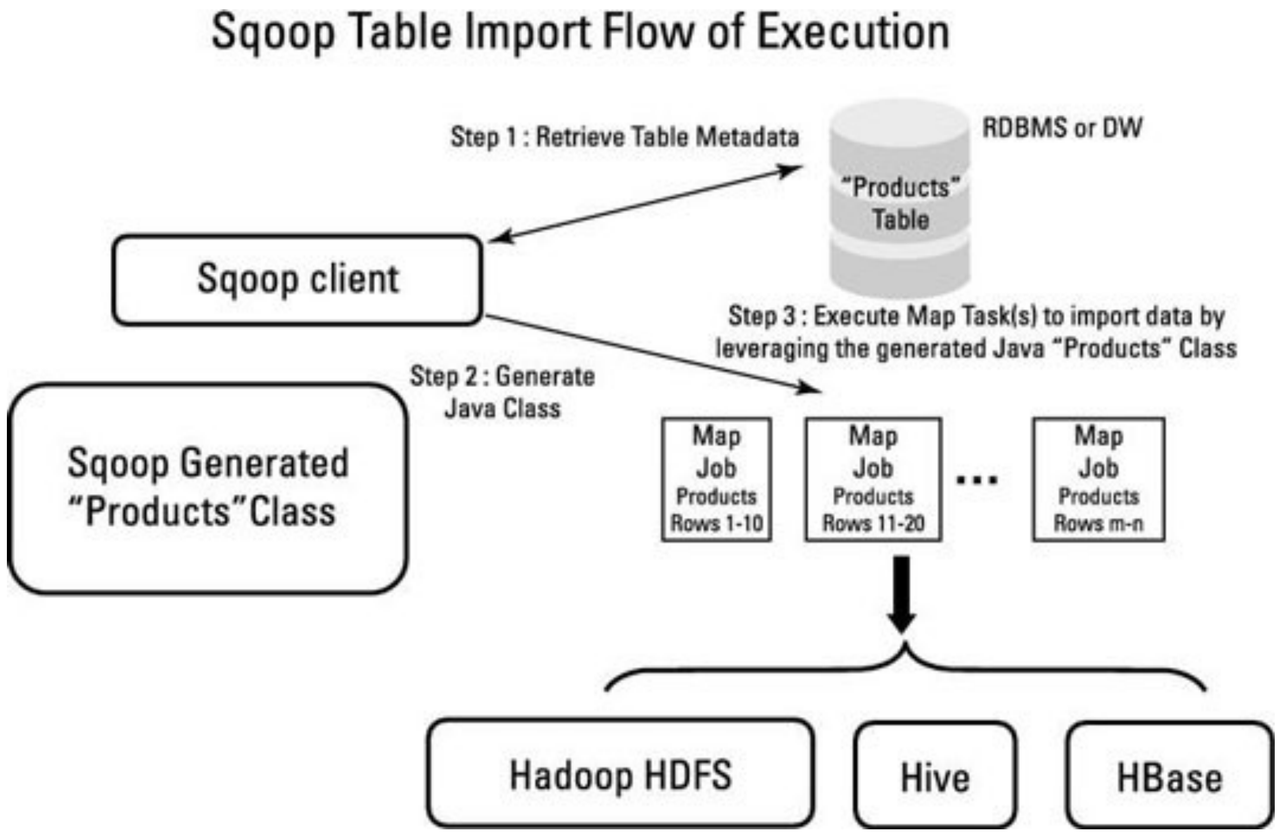
아파치 스쿱 동작 방식 – Code Generation

Sqoop Import Code Generator

```
$ sqoop codegen \
    -- connect jdbc:mysql://${host-ip}:${port}/${db-name} \
    -- username ${username} \
    -- password ${password} \
    -- table ${table-name}
$ docker exec -it day1_sqoop sqoop codegen -fs local -jt local \
    --connect jdbc:mysql://day1_mysql:3306/testdb \
    --table seoul_popular_trip \
    --fields-terminated-by '\t' \
    --username sqoop \
    --password sqoop
# 생성된 자바 코드를 vim 을 통해 확인합니다
$ docker exec -it day1_sqoop cat \
    /tmp/sqoop-root/compile/.../seoul_popular_trip.java | vi -
```



Sqoop Import Internal



Step 1. Sqoop Import

1-1. 스쿱 클라이언트가 입력 파라미터를 해석 및 검증 수행

1-2. 접속정보를 통해 대상 데이터베이스로 부터 메타 정보를 수집

Step 2. Code Generation

2-1. 메타정보를 통해 M/R 수행을 위한 코드를 생성합니다

2-2. 이 때에 POJO, ORM 및 DBWritable 구현 코드가 생성됩니다

Step 3. Execute Map Task(s)

3-1. 2 단계에서 생성된 Jar 파일을 통해 M/R Job 을 제출합니다

3-2. 해당 Task 는 개별 컨테이너에서 DB 접속 및 수집을 합니다

Step 4. Complete

4-1. 수집된 데이터는 로컬 혹은 분산 저장소에 저장됩니다

4-2. 작업이 성공하면 _SUCCESS 파일과 함께 part 파일이 생성됩니다

https://www.freecodecamp.org/news/an-in-depth-introduction-to-sqoop-architecture-ad4ae0532583/

About commands part I

hadoop common options
sqoop import command

테이블 임포트 – 공통 매개변수

Common arguments

모든 명령어에 포함되는 공통 매개변수 혹은 인수

$ sqoop import --connect <jdbc-uri>

https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html

인수	설명
--connect <jdbc-uri>	JDBC 접속 문자열 (예: jdbc:mysql://hostname:port/dbname)
--driver <class-name>	JDBC 드라이버 명 (예: com.mysql.jdbc.Driver)
--hadoop-mapred-home <dir>	환경변수 $HADOOP_MAPRED_HOME 값을 덮어씁니다
--password-file	패스워드 파일을 포함한 경로를 지정합니다 (예: --password-file hdfs://user/sqoop/.password)
-P	패스워드를 콘솔에서 입력합니다
--password <password>	패스워드를 암호화되지 않은 상태로 입력합니다
--username <username>	유저이름을 입력합니다
--verbose	보다 상세한 설명을 출력합니다
--relaxed-isolation	격리 수준을 낮게하여 커밋되지 않은 데이터도 가져옵니다 (Read or Shared Lock 을 잡지 않고 조회합니다)

테이블 임포트 – 하둡 일반 매개변수

Generic Hadoop arguments

하둡 관련 일반 매개 변수는 명령어 다음에 나와야 하며, --connect 와 같은 도구 명령어 보다는 앞에 명시 되어야만 합니다

$ sqoop import -conf <config-file> -D <property=value> -fs <local> -jt <local> --connect jdbc:mysql://localhost:3306 ...

https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html

인수	설명
-conf <configuration file>	어플리케이션 Configuration 파일을 지정합니다
-D <property=value>	어플리케이션 Property 와 값을 지정합니다 (예: D mapred.job.name=<job_name>)
-fs <local | namenode:port>	로컬 파일 시스템 혹은 분산 저장소 네임노드를 지정합니다
-jt <local | jobtracker:port>	로컬 잡 실행 혹은 리모트 잡 트래커를 지정합니다
-files <comma separated files>	맵 리듀스 작업시에 복사되어야 할 로컬 파일들을 콤마로 구분하여 지정합니다
-libjars <comma separated jars>	클래스패스에 포함할 로컬 jar 파일들을 콤마로 구분하여 지정합니다
-archives <comma separated archives)	머신에서 압축 해제되어야 할 로컬 압축 파일 목록을 콤마로 구분하여 지정합니다

테이블 임포트 – 수집 조정 매개변수

Import control arguments

테이블 수집에 필요한 조정 매개변수 입니다

$ sqoop import --num-mappers <n> --table <table-name> ...

https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html

인수	설명
--append	대상 경로에 파일을 추가합니다
--as-textfile or --as-parquetfile	저장 포맷을 텍스트 혹은 파케이 파일로 저장합니다 (예: --as-avrodatafile, --as-sequencefile 등)
--columns <col, col, col ...>	임포트 할 컬럼 이름 목록을 지정합니다
--delete-target-dir	임포트 전에 결과 경로를 삭제합니다 (M/R 특성상 대상 경로가 있으면 수행되지 않습니다)
--direct	데이터베이스에서 다이렉트 커넥터가 지원하는 경우 적용합니다
--fetch-size <n>	데이터베이스에서 한 번에 패치 해 올 레코드 수를 지정합니다 (default : 1000)
-m, --num-mappers <n>	동시에 가져오는 작업 수를 지정합니다 (단, --split-by 옵션이 지정되어야 합니다)
-e, --query <statement>	가져올 질의를 직접 지정합니다 (단, 전체 데이터를 가져올 때에도 WHERE $CONDITIONS 구문을 반드시 지정해야 합니다)
--split-by <column-name>	병렬로 수행하기 위한 필터 컬럼 지정을 합니다 (숫자, 날짜 등의 min, max 함수를 통해 분할 가능한 컬럼을 지정 합니다)
--split-limit <n>	병렬로 수행될 값의 최대값을 지정합니다

테이블 임포트 – 수집 조정 매개변수

Import control arguments

테이블 수집에 필요한 조정 매개변수 입니다

$ sqoop import --num-mappers <n> --table <table-name> ...

https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html

인수	설명
--table <table-name>	수집 대상 테이블의 이름을 지정합니다
--target-dir <dir>	저장될 대상 경로를 지정합니다
--warehouse-dir <dir>	수집될 대상 경로의 최상위 경로를 지정합니다 (--import-all-tables 의 경우 필수)
--where <where clause>	조건 절을 지정합니다
-z, --compress	압축 여부를 결정합니다
--compression-codec <c>	압축 코덱을 지정합니다 (default : gzip)
--null-string <null-string>	문자열 컬럼인 경우의 널 값을 지정합니다 (varchar 와 같은 문자열 컬럼)
--null-non-string <null-string>	문자열 컬럼이 아닌 경우 널 값을 지정합니다 (숫자, 날짜와 같은 컬럼) (하이브의 경우 default 널 값이 \\N 으로 지정해야 is null 과 같은 구문이 동작합니다)

테이블 임포트 – 수집 조정 입력 포맷

Input line formatting arguments

출력되는 라인의 포맷 설정에 대한 매개변수 입니다

https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html

인수	설명
--input-enclosed-by <char>	필드를 구분 캐릭터에 무관하게 묶을 때에 사용하는 캐릭터를 지정합니다 (쌍따옴표: --enclosed-by '\"')
--input-escaped-by <char>	특수문자를 이스케이핑 할 때 사용하는 캐릭터를 지정합니다 (역슬래시: --escaped-by \\ )
--input-fields-terminated-by <char>	텍스트 출력 시 필드의 구분 캐릭터를 지정합니다 (default: ,)
--input-lines-terminated-by <char>	텍스트 출력 시 라인의 구분 캐릭터를 지정합니다 (default: \n)

```
\b (backspace)
\n (newline)
\r (carriage return)
\t (tab)
\" (double-quote)
\\' (single-quote)
\\ (backslash)
\0 (NUL)
```

테이블 임포트 – 수집 조정 출력 포맷

Output line formatting arguments

출력되는 라인의 포맷 설정에 대한 매개변수 입니다

https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html

인수	설명
--enclosed-by <char>	필드를 구분 캐릭터에 무관하게 묶을 때에 사용하는 캐릭터를 지정합니다 (쌍따옴표: --enclosed-by '\"')
--escaped-by <char>	특수문자를 이스케이핑 할 때 사용하는 캐릭터를 지정합니다 (역슬래시: --escaped-by \\ )
--fields-terminated-by <char>	텍스트 출력 시 필드의 구분 캐릭터를 지정합니다 (default: ,)
--lines-terminated-by <char>	텍스트 출력 시 라인의 구분 캐릭터를 지정합니다 (default: \n)

	`\b (backspace)`
	`\n (newline)`
	`\r (carriage return)`
	`\t (tab)`
	`\" (double-quote)`
	`\\' (single-quote)`
	`\\ (backslash)`
	`\0 (NUL)`

--optionally-enclosed-by <char>	필드에 구분자가 포함된 경우에만 대상 캐릭터로 묶습니다

아파치 스쿱 제공 기능 – 하이브 매개변수

Hive arguments

하이브 테이블 임포트 시에 필요한 매개변수 입니다

https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html

인수	설명
--hive-home <dir>	$HIVE_HOME 값을 덮어씁니다
--hive-import	하이브로 테이블을 임포트 합니다
--hive-overwrite	기존 하이브 테이블을 덮어씁니다
--create-hive-table	하이브 테이블을 새로 생성합니다 (default: false)
--hive-table <table-name>	하이브 테이블 별도의 이름으로 지정합니다
--hive-drop-import-delims	하이브로 임포트 시에 문자열 필드의 \n, \r, \01 문자열을 제거합니다
--hive-delims-replacement <c>	하이브로 임포트 시에 문자열 필드의 \n, \r, \01 문자열을 지정한 문자열로 교체합니다
--hive-partition-key <k>	파티셔닝 된 하이브 테이블의 파티션 키 이름을 지정합니다
--hive-partition-value <v>	하이브로 임포트시에 파티션 키에 대한 값을 지정합니다
--map-column-hive <map>	SQL 유형을 Hive 유형으로 변경 시에 별도로 맵핑을 지정할 때에 사용합니다

테이블 임포트 - 증분 수집

Incremental Imports

이전에 수집했던 마지막 특정 컬럼 값을 알고 있어야 하고, 해당 값이 이후에 증가한 데이터에 대해서만 증분 수집이 가능하므로 아래의 3가지 옵션을 활용하여 제한적인 상황에서만 수집이 가능합니다

https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html

커맨드	설명
--check-column <col>	대소 비교를 통해 범위 조건을 지정할 수 있는 컬럼 (예: auto_increment 유형의 number 혹은 modified date 컬럼)
--incremental <mode>	append 모드는 항상 새로운 값들이 추가만 되는 상황에 적용할 수 있으며, check-column 지정으로 특정 시점을 찾습니다 lastmodified 모드는 임의의 필드 값들이 업데이트 되는 상황에서 적용하며, last_modified 같은 date 유형에 활용합니다
--last-value <value>	이전 import 시에 적용되었던 check-column 의 마지막 값이며, import 완료 시에 해당 값이 화면에 출력됩니다 출력 예: 20/05/25 14:47:47 INFO tool.ImportTool: --incremental append 20/05/25 14:47:47 INFO tool.ImportTool: --check-column id 20/05/25 14:47:47 INFO tool.ImportTool: --last-value 2

테이블 임포트 – 로컬

sqoop import -fs local -jt local

```
# 컨테이너 로컬 경로에 테이블 수집을 수행합니다
$ docker exec -it day1_sqoop sqoop import -fs local -jt local -m 1 --connect jdbc:mysql://day1_mysql:3306/testdb --table seoul_popular_trip --delete-target-dir --target-dir /home/sqoop/target/seoul_popular_trip --fields-terminated-by '\t' --verbose --username sqoop --password sqoop
# 로컬로 복사해서 첫 줄을 읽어 정상적으로 수집이 되었는지 확인합니다
$ rm -rf part-m-00000 ; docker cp day1_sqoop:/home/sqoop/target/seoul_popular_trip/part-m-00000 . ; head -1 part-m-00000
# 콤마 구분자로 저장하되 필드를 쌍따옴표로 묶습니다
$ docker exec -it day1_sqoop sqoop import -fs local -jt local -m 1 --connect jdbc:mysql://day1_mysql:3306/testdb --table seoul_popular_trip --delete-target-dir --target-dir /home/sqoop/target/seoul_popular_trip --fields-terminated-by ',' --enclosed-by '\"' --verbose --username sqoop --password sqoop
# 콤마가 포함된 필드만 쌍따옴표로 묶습니다
$ docker exec -it day1_sqoop sqoop import -fs local -jt local -m 1 --connect jdbc:mysql://day1_mysql:3306/testdb --table seoul_popular_trip --delete-target-dir --target-dir /home/sqoop/target/seoul_popular_trip --fields-terminated-by ',' --optionally-enclosed-by '\"' --verbose --username sqoop --password sqoop
```

테이블 임포트 – 파일 추가 수집

sqoop import --append

```
# 이미 적재된 테이블에 추가로 part-m-0000? 파일을 추가해야 하므로 id 값의 범위를 확인합니다
$ docker exec -it day1_sqoop sqoop eval --connect jdbc:mysql://day1_mysql:3306/testdb --username sqoop --password sqoop -e "select min(id), max(id) from seoul_popular_trip"
| min(id)  | max(id)    |
| 34       | 29578      |
# 처음에는 15000 미만의 id 값을 수집하고 두 번째는 15000 이상의 값을 수집하여 append 적재 합니다
$ docker exec -it day1_sqoop sqoop import -m 1 --connect jdbc:mysql://day1_mysql:3306/testdb --table seoul_popular_trip --where "id < 15000" --target-dir /user/sqoop/target/seoul_popular_append --fields-terminated-by '\t' --verbose --username sqoop --password sqoop
$ docker exec -it day1_sqoop sqoop import -m 1 --connect jdbc:mysql://day1_mysql:3306/testdb --table seoul_popular_trip --where "id >= 15000" --target-dir /user/sqoop/target/seoul_popular_append --fields-terminated-by '\t' --verbose --username sqoop --password sqoop --append
# 네임노드에 적재된 파일이 총 2개의 파트 파일로 적재 되었는지 확인합니다
http://localhost:50070/explorer.html#/user/sqoop/target/seoul_popular_append
```

테이블 임포트 – 증분 파일 수집

sqoop import --incremental {mode} --check-column {column} --last-value {value}

```
# 증분 테이블의 경우 항상 증가하는 키가 필요하므로 auto_increment id 를 가진 테이블을 생성후 데이터를 입력합니다 (편의상 쿼리문은 -e 로만 표현)
$ docker exec -it day1_sqoop sqoop eval --connect jdbc:mysql://day1_mysql:3306/testdb --username sqoop --password sqoop \
    -e "create table inc_table (id int not null auto_increment, name varchar(30), salary int, primary key (id))"
    -e "insert into inc_table (name, salary) values ('suhyuk', 10000)"
    -e "select * from inc_table "
# 1건이 입력된 테이블을 수집하고, part-m-00000 파일이 생성되었는지 namenode 에서 확인합니다 (초기 수집이라 --last-value 0 은 넣지 않아도 됩니다)
$ docker exec -it day1_sqoop sqoop import -m 1 --connect jdbc:mysql://day1_mysql:3306/testdb --table inc_table --incremental append --check-column id --last-value 0 --target-dir /user/sqoop/target/seoul_popular_incremental --fields-terminated-by '\t' --verbose --username sqoop --password sqoop
# 두 번째 값을 입력 후, 동일한 방식으로 part-m-00001 파일이 생성되었는지 확인합니다
    -e "insert into inc_table (name, salary) values ('psyoblade', 1000000)"
$ docker exec -it day1_sqoop sqoop import -m 1 --connect jdbc:mysql://day1_mysql:3306/testdb --table inc_table --incremental append --check-column id --last-value 1 --target-dir /user/sqoop/target/seoul_popular_incremental --fields-terminated-by '\t' --verbose --username sqoop --password sqoop
```

테이블 임포트 – 모든 테이블 수집

sqoop-import-all-tables

```
$ sqoop import-all-tables (generic-args) (import-args)
# 지정한 데이터베이스의 모든 테이블을 수집합니다
```

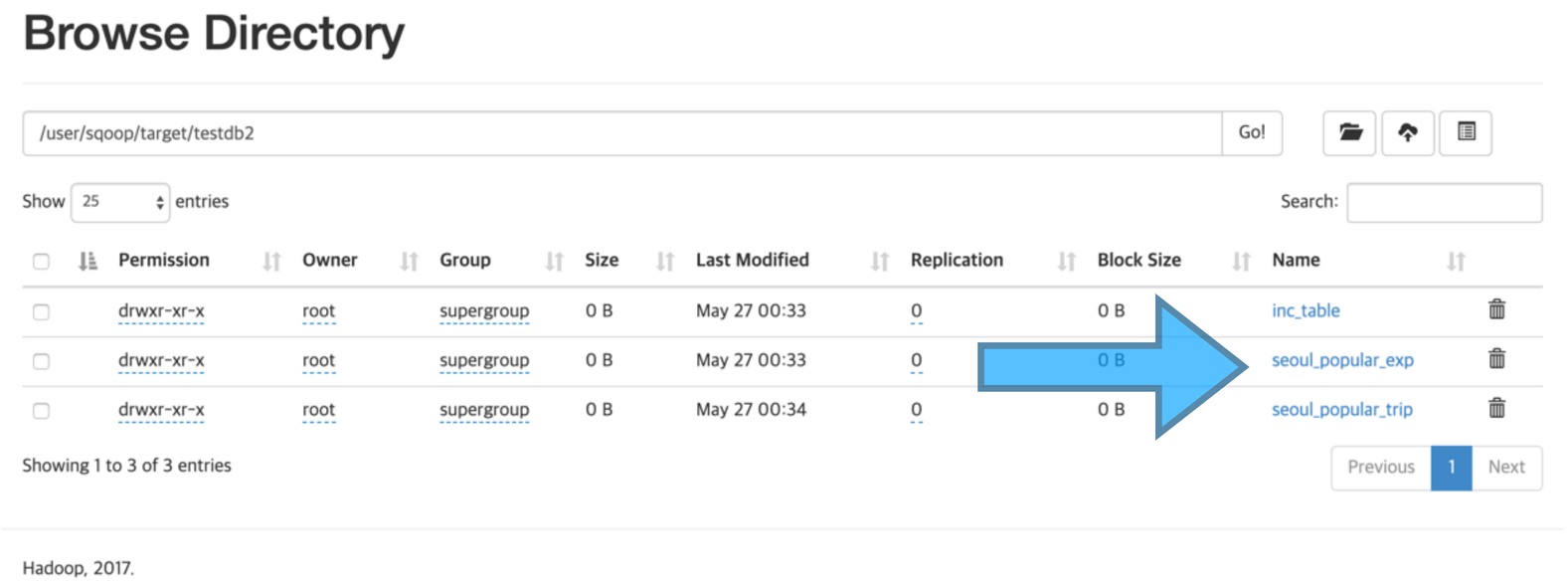
https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html

커맨드	설명
--warehouse-dir	적재 테이블의 최상위 경로를 지정합니다. (하위에 테이블 이름 별로 경로가 생성됩니다)
	1. 수집 대상 테이블은 반드시 primary-key 가 존재해야 하거나, --autoreset-to-one-mapper 옵션이 지정되어야 합니다
	2. 수집 되는 테이블의 필드를 지정할 수 없기 때문에 모든 필드가 수집 됩니다
	3. 수집 되는 테이블의 WHERE 절을 지정할 수 없습니다

테이블 임포트 – 모든 테이블 수집

sqoop-import-all-tables

```
# 지정한 데이터베이스의 모든 테이블을 수집합니다
$ docker exec -it day1_sqoop sqoop import-all-tables  --connect jdbc:mysql://day1_mysql:3306/testdb --warehouse-dir /user/sqoop/target/testdb
--fields-terminated-by '\t' --verbose --username sqoop --password sqoop
```



테이블 임포트 – 패스워드 저장파일 활용

sqoop import --password-file {password-file}

```
# 보안성 패스워드 파일을 400 으로 설정하고 아래와 같이 패스워드 입력 없이 수행이 가능합니다.
(단 파일 생성 시에 공백이 없도록 echo -n "password" > .password 로 생성해야 안전합니다
$ docker exec -it day1_sqoop bash
$ cd /home/sqoop ; echo -n "sqoop" > .password
$ hadoop fs -put .password /user/sqoop/
$ hadoop fs -chmod 400 /user/sqoop/.password
$ docker exec -it day1_sqoop sqoop eval --connect jdbc:mysql://day1_mysql:3306/testdb --username sqoop --password-file /user/sqoop/.password
-e "select * from inc_table "
```

About commands part II

sqoop export command

테이블 익스포트 – 적재 조정 매개변수

Export control arguments

테이블 적재에 필요한 조정 매개변수 입니다

$ sqoop export --num-mappers <n> --table <table-name> ...

https://sqoop.apache.org/docs/1.4.7/SqoopUserGuide.html

인수	설명
--columns <col, col, col...>	익스포트 할 테이블의 컬럼을 지정합니다
--direct	데이터베이스에서 다이렉트 커넥터가 지원하는 경우 적용합니다
--export-dir	적재 대상 데이터가 저장되어 있는 경로를 지정합니다
-m, --num-mappers <n>	동시에 가져오는 작업 수를 지정합니다 (단, --split-by 옵션이 지정되어야 합니다)
--table <table-name>	데이터가 적재될 테이블 이름을 지정합니다
--update-key <col-name>	업데이트 모드에서 업데이트를 위한 키를 지정합니다
--update-mode <mode>	updateonly (default) 는 업데이트만 수행하고, allowinsert 는 upsert 로 동작합니다
--input-null-string <null-string>	저장된 문자열 컬럼에서 널값으로 적재할 문자열을 지정합니다
--input-null-non-string <null-string>	저장된 문자열이 아닌 컬럼에서 널값으로 적재할 문자열을 지정합니다
--staging-table <staging-table-name>	적재 실패를 대비하여, 입력하기 전에 일차 스테이징이 이루어질 테이블이름을 지정합니다 (중복 키 혹은 일부 task 실패 등)
--clear-staging-table	스테이징 테이블을 삭제합니다

테이블 익스포트 – 테이블 적재

sqoop export --export-dir <dir>

```
# 적재 테스트를 위해 임포트 했던 데이터와 동일한 스키마를 가진 테이블을 생성하고 확인 합니다.
$ docker exec -it day1_sqoop sqoop eval --connect jdbc:mysql://day1_mysql:3306/testdb --username sqoop --password sqoop -e "create table testdb.seoul_popular_exp (category int not null, id int not null, name varchar(100), address varchar(100), naddress varchar(100), tel varchar(20), tag varchar(500)) character set utf8 collate utf8_general_ci;"
$ docker exec -it day1_sqoop sqoop list-tables --connect jdbc:mysql://day1_mysql:3306/testdb --username sqoop --password sqoop
# 익스포트 명령을 통해서 임포트 한 데이터를 그대로 적재합니다
$ docker exec -it day1_sqoop sqoop export -m 1 --connect jdbc:mysql://day1_mysql:3306/testdb --table seoul_popular_exp --export-dir /user/sqoop/target/seoul_popular_trip --fields-terminated-by '\t' --verbose --username sqoop --password sqoop
# 원본 테이블의 레코드 수와 적재된 테이블의 레코드 수를 비교합니다 (eval 명령은 편의상 질의문만 기술합니다)
    -e "select count(1) from seoul_popular_trip"
    -e "select count(1) from seoul_popular_exp"
| count(1)          |
| 1956              |
```

About commands part III

codegen, eval, list-databases, list-tables

명령어 도구 – 코드 생성

sqoop-codegen

```
$ sqoop codegen (generic-args) (codegen-args)
$ docker exec -it day1_sqoop sqoop codegen -fs local -jt local --connect jdbc:mysql://day1_mysql:3306/testdb --table seoul_popular_trip --fields-terminated-by '\t' --username sqoop --password sqoop
```

커맨드	설명
--bindir <dir>	컴파일된 오브젝트 저장위치를 지정
--class-name <name>	생성될 클래스 명을 지정합니다 (--package-name 옵션보다 우선합니다)
--jar-file <file>	코드 생성은 하지 않고, jar 만 생성합니다
--outdir <dir>	생성된 코드의 위치를 지정합니다
--package-name <name>	자동으로 생성된 클래스의 패키지를 지정합니다
--map-column-java <m>	기본 SQL type 과 Java type 데이터 유형 맵핑을 다시 구성합니다

명령어 도구 – 질의 수행

sqoop-eval

```
$ sqoop eval (generic-args) (codegen-args)
$ docker exec -it day1_sqoop sqoop eval --connect jdbc:mysql://day1_mysql:3306/testdb --query "show databases"
```

커맨드	설명
-e, --query <statement>	SQL 문을 실행합니다

명령어 도구 – 질의 수행

sqoop eval, list-databases, list-tables

```
# Join Table
$ docker exec -it day1_sqoop sqoop eval --connect jdbc:mysql://day1_mysql:3306/testdb --username sqoop --password sqoop \
    -e "select a.id as a_id, b.id as b_id from seoul_popular_trip a join seoul_popular_trip b on (a.id = b.id) limit 10"

# Create, Insert and Select Table - 여기서부터 편의상 앞의 명령어는 생략했습니다
    -e "create table tbl_salary (id int not null auto_increment, name varchar(30), salary int, primary key (id))"
    -e "insert into tbl_salary (name, salary) values ('suhyuk', 10000)"
    -e "select * from tbl_salary "

# Show Database, Tables - list-database, list-tables 와 거의 동일합니다
    -e "show databases"
    -e "show tables"
```