**23. Sqoop数据导入导出（Mysql，HDFS，Hive，HBase）**

**实验目的**

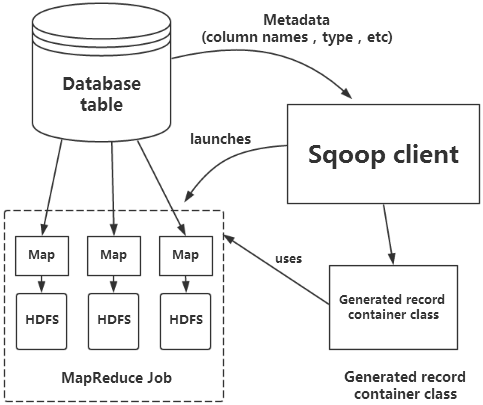
1. 掌握Sqoop的各种导入操作

2. 掌握Sqoop的各种导出操作

**实验原理**

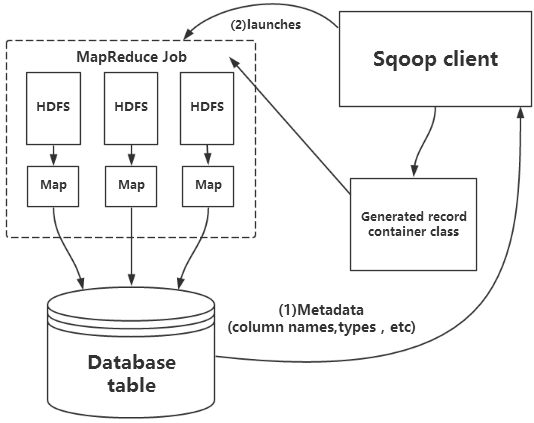
在导入开始之前，Sqoop使用JDBC来检查将要导入的表。检索出表中所有的列以及列的SQL数据类型。这些SQL类型（varchar、integer）被映射到Java数据类型（String、Integer等），在MapReduce应用中将使用这些对应的Java类型来保存字段的值。Sqoop的代码生成器使用这些信息来创建对应表的类，用于保存从表中抽取的记录。Sqoop启动的MapReduce作业用到一个InputFormat，他可以通过JDBC从一个数据库表中读取部分内容。

Hadoop提供的DataDriverDB InputFormat能为查询结果进行划分传给指定个数的map任务。为了获取更好的导入性能，查询会根据一个“划分列”来进行划分。Sqoop会选择一个合适的列作为划分列（通常是表的主键）。在生成反序列化代码和配置InputFormat之后，Sqoop将作业发送到MapReduce集群。Map任务将执行查询并将ResultSet中的数据反序列化到生成类的实例，这些数据要么直接保存在SequenceFile文件中，要么在写到HDFS之前被转换成分割的文本。Sqoop不需要每次都导入整张表，用户也可以在查询中加入到where子句，以此来限定需要导入的记录。

[](http://192.168.1.241/doc/exper/39/img/01.png)

（Sqoop数据导入原理图）

Sqoop导出功能的架构与其导入功能非常相似，在执行导出操作之前，Sqoop会根据数据库连接字符串来选择一个导出方法。一般为JDBC。然后，Sqoop会根据目标表的定义生成一个Java类。这个生成的类能够从文本文件中解析记录，并能够向表中插入合适类型的值。接着会启动一个MapReduce作业，从HDFS中读取源数据文件，使用生成的类解析记录，并且执行选定的导出方法。

[](http://192.168.1.241/doc/exper/39/img/02.png)

（Sqoop 数据导出原理图）

**实验内容**

1. 练习使用Sqoop将数据在Mysql与HDFS之间的导入导出。

2. 练习使用Sqoop将数据从Mysql中导入到HBase中。

3. 练习使用Sqoop将数据在Mysql与Hive之间的导入导出。

**实验步骤**

现有buyer\_log表，存储了买家行为日志，包含四个字段（ID，用户ID，时间，IP，操作类型）。

ID 用户ID 时间 IP 操作类型

461 10181 2010-03-26 19:45:07 123.127.164.252 1

462 10262 2010-03-26 19:55:10 123.127.164.252 1

463 20001 2010-03-29 14:28:02 221.208.129.117 2

464 20001 2010-03-29 14:28:02 221.208.129.117 1

465 20002 2010-03-30 10:56:35 222.44.94.235 2

466 20002 2010-03-30 10:56:35 222.44.94.235 1

481 10181 2010-03-31 16:48:43 123.127.164.252 1

482 10181 2010-04-01 17:35:05 123.127.164.252 1

483 10181 2010-04-02 10:34:20 123.127.164.252 1

484 20001 2010-04-04 16:38:22 221.208.129.38 1

**一、使用Sqoop将数据从MySQL导出到HDFS**

1. 数据准备

获取buyer\_log，存放在/home/hfut/sqoop-data中

2. 系统环境准备

启动Hadoop、启动MySQL

3. MySQL上的操作

创建u\_sqoop用户，密码为sqoop

mysql> grant all on \*.\* to u\_sqoop@'%' identified by 'sqoop';

mysql> grant all on \*.\* to u\_sqoop@'localhost' identified by 'sqoop';

mysql> grant all on \*.\* to u\_sqoop@'master' identified by 'sqoop';

登录MySQL

[hfut@master ~]$ mysql -u u\_sqoop -p

创建数据库mydb

mysql> create database mydb;

切换到mydb数据库，创建数据表record，用于存储buyer\_log中的内容。

mysql> use mydb;

mysql> create table record

(

id varchar(100),

buyer\_id varchar(100),

dt varchar(100),

ip varchar(100),

opt\_type varchar(100)

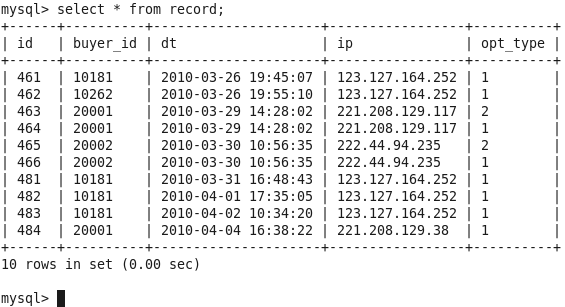
);

将Linux本地/home/hfut/sqoop-data/buyer\_log里的内容，导入的mydb数据库record表中。

mysql> load data local infile '/home/hfut/sqoop-data/buyer\_log' into table record fields terminated by '\t';

查看record表中内容。

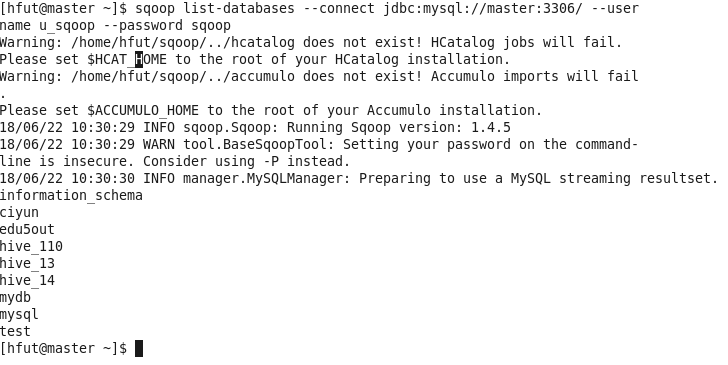
mysql> select \* from record;



4. Sqoop上的操作举例

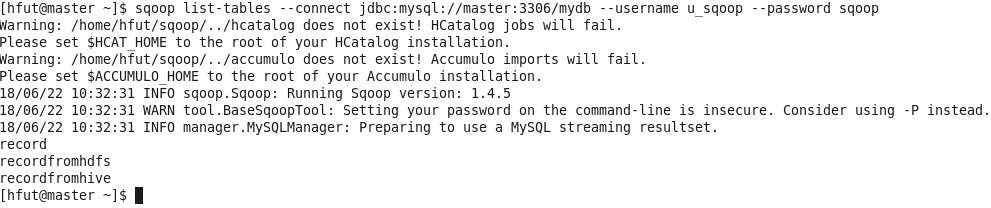
另开一个终端窗口，使用Sqoop查看Mysql中的数据库。此步目的是检查Sqoop以及Mysql是否可以正常使用。

[hfut@master ~]$ sqoop list-databases --connect jdbc:mysql://master:3306/ --username u\_sqoop --password sqoop



使用Sqoop查看Mysql中的表 (在jdbc连接字符串中添加了数据库的名称。用于直接访问数据库实例)。

[hfut@master ~]$ sqoop list-tables --connect jdbc:mysql://master:3306/mydb --username u\_sqoop --password sqoop



5. 使用Sqoop将MySQL中的数据导入到HDFS

使用Sqoop将Mysql中mydb数据库record表里的数据导入到HDFS的/mysqoop目录里。（HDFS上的/mysqoop目录，不需要提前创建）。

其中-m 1 参数代表的含义是使用多少个并行，这个参数的值是1，说明没有开启并行功能。

将m参数的数值调为5或者更大，Sqoop就会开启5个进程，同时进行数据的导入操作。

[hfut@master ~]$ sqoop import --connect jdbc:mysql://master:3306/mydb --username u\_sqoop --password sqoop --table record --target-dir /mysqoop -m 1

导入过程：

Warning: /home/hfut/sqoop/../hcatalog does not exist! HCatalog jobs will fail.

Please set $HCAT\_HOME to the root of your HCatalog installation.

Warning: /home/hfut/sqoop/../accumulo does not exist! Accumulo imports will fail.

Please set $ACCUMULO\_HOME to the root of your Accumulo installation.

18/06/22 10:38:54 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5

18/06/22 10:38:54 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.

18/06/22 10:38:55 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.

18/06/22 10:38:55 INFO tool.CodeGenTool: Beginning code generation

18/06/22 10:38:55 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `record` AS t LIMIT 1

18/06/22 10:38:55 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `record` AS t LIMIT 1

18/06/22 10:38:55 INFO orm.CompilationManager: HADOOP\_MAPRED\_HOME is /home/hfut/hadoop-2.5.2

Note: /tmp/sqoop-hfut/compile/35bfdcab7def4a96c1481afcdc0caf06/record.java uses or overrides a deprecated API.

Note: Recompile with -Xlint:deprecation for details.

18/06/22 10:38:59 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-hfut/compile/35bfdcab7def4a96c1481afcdc0caf06/record.jar

18/06/22 10:38:59 WARN manager.MySQLManager: It looks like you are importing from mysql.

18/06/22 10:38:59 WARN manager.MySQLManager: This transfer can be faster! Use the --direct

18/06/22 10:38:59 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.

18/06/22 10:38:59 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)

18/06/22 10:38:59 INFO mapreduce.ImportJobBase: Beginning import of record

SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/home/hfut/hadoop-2.5.2/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/home/hfut/hbase-1.1.2/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.

SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

18/06/22 10:39:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

18/06/22 10:39:00 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar

18/06/22 10:39:01 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps

18/06/22 10:39:02 INFO client.RMProxy: Connecting to ResourceManager at master/192.168.79.129:18040

18/06/22 10:39:05 INFO db.DBInputFormat: Using read commited transaction isolation

18/06/22 10:39:05 INFO mapreduce.JobSubmitter: number of splits:1

18/06/22 10:39:05 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_1529634989134\_0002

18/06/22 10:39:05 INFO impl.YarnClientImpl: Submitted application application\_1529634989134\_0002

18/06/22 10:39:05 INFO mapreduce.Job: The url to track the job: http://master:18088/proxy/application\_1529634989134\_0002/

18/06/22 10:39:05 INFO mapreduce.Job: Running job: job\_1529634989134\_0002

18/06/22 10:39:18 INFO mapreduce.Job: Job job\_1529634989134\_0002 running in uber mode : false

18/06/22 10:39:18 INFO mapreduce.Job: map 0% reduce 0%

18/06/22 10:39:27 INFO mapreduce.Job: map 100% reduce 0%

18/06/22 10:39:29 INFO mapreduce.Job: Job job\_1529634989134\_0002 completed successfully

18/06/22 10:39:29 INFO mapreduce.Job: Counters: 30

File System Counters

FILE: Number of bytes read=0

FILE: Number of bytes written=105333

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=87

HDFS: Number of bytes written=475

HDFS: Number of read operations=4

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

Job Counters

Launched map tasks=1

Other local map tasks=1

Total time spent by all maps in occupied slots (ms)=7695

Total time spent by all reduces in occupied slots (ms)=0

Total time spent by all map tasks (ms)=7695

Total vcore-seconds taken by all map tasks=7695

Total megabyte-seconds taken by all map tasks=7879680

Map-Reduce Framework

Map input records=10

Map output records=10

Input split bytes=87

Spilled Records=0

Failed Shuffles=0

Merged Map outputs=0

GC time elapsed (ms)=61

CPU time spent (ms)=1230

Physical memory (bytes) snapshot=109727744

Virtual memory (bytes) snapshot=840323072

Total committed heap usage (bytes)=29687808

File Input Format Counters

Bytes Read=0

File Output Format Counters

Bytes Written=475

18/06/22 10:39:29 INFO mapreduce.ImportJobBase: Transferred 475 bytes in 27.5658 seconds (17.2315 bytes/sec)

18/06/22 10:39:29 INFO mapreduce.ImportJobBase: Retrieved 10 records.

查看HDFS上/mysqoop目录下的文件内容。

[hfut@master ~]$ hadoop fs -cat /mysqoop/part-m-00000



**二、使用Sqoop将数据从HDFS导出到MySQL**

在Mysql窗口，mydb数据库下，新建一张表recordfromhdfs，表结构与record表相同。将HDFS上/mysqoop/part-m-00000里的数据，导出到MySQL中的recordfromhdfs表中。

1. MySQL上的操作

在Mysql窗口，mydb数据库下，新建一张表recordfromhdfs，表结构与record表相同。

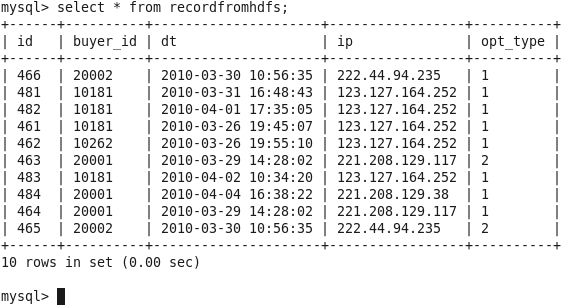
mysql> use mydb;

mysql> create table recordfromhdfs like record;

2. Sqoop上的操作

[hfut@master ~]$ sqoop export --connect jdbc:mysql://master:3306/mydb?characterEncoding=UTF-8 --username u\_sqoop --password sqoop --table recordfromhdfs --export-dir hdfs://master:9000/mysqoop/part-m-00000

3. 查看Mysql中recordfromhdfs表中内容：



**三、将MySQL中数据导入到HBase中**

1. 新开一个终端窗口，启动HBase，进入HBase Shell

[hfut@master ~]$ start-hbase.sh

[hfut@master ~]$ hbase shell

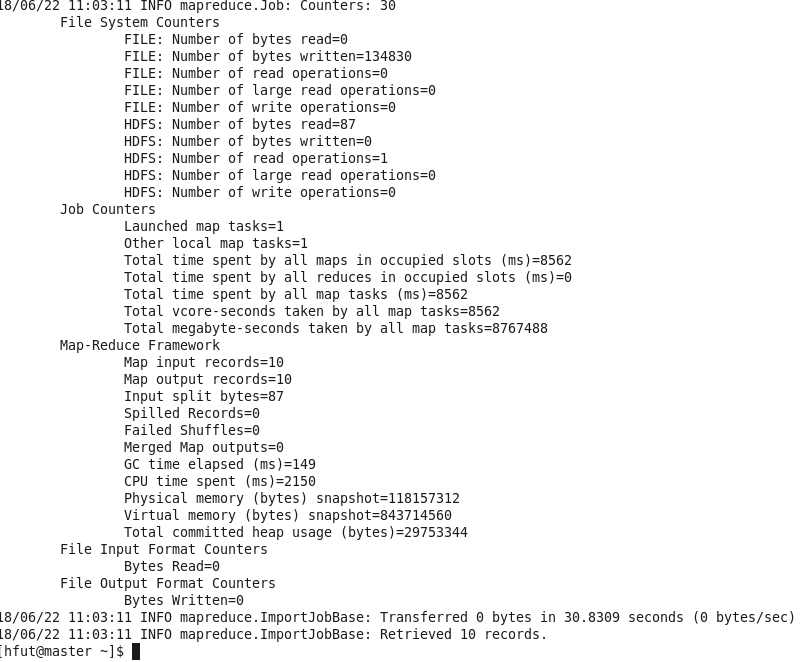
2. HBase上的操作

hbase(main):009:0> create 'hbaserecord', 'mycf'

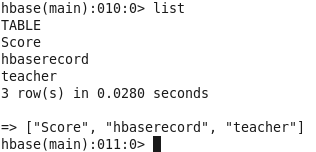
创建hbaserecord表，有一个列族mycf

3. Sqoop上的操作

[hfut@master ~]$ sqoop import --connect jdbc:mysql://master:3306/mydb?characterEncoding=UTF-8 --username u\_sqoop --password sqoop --table record --columns "id,buyer\_id,dt,ip, opt\_type" --hbase-table hbaserecord --column-family mycf --hbase-row-key dt -m 1



4. 查看HBase中，都有哪些表



5. 查看HBase表hbaserecord里的内容。

hbase(main):011:0> scan 'hbaserecord'



**四、将HBase中数据导出到MySQL中**

使用Sqoop将HBase中数据导出到Mysql中，暂时无法直接接口实现，需要借助其他途径去处理，比如：HBase->HDFS->MySQL或 HDFS->Hive->MySQL

**五、将MySQL中的数据导入到Hive中**

使用Sqoop将Hive表hiverecord表中的数据，导出到MySQL中的recordfromhive表中。

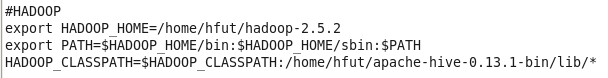
1. 修改$HADOOP\_CLASSPATH

[hfut@master ~]$ gedit ~/.bash\_profile

[hfut@master ~]$ source ~/.bash\_profile

将以下内容追加到~/.bash\_profile的#hadoop下：

HADOOP\_CLASSPATH=$HADOOP\_CLASSPATH:/home/hfut/apache-hive-0.13.1-bin/lib/\*



2. Hive上的操作

新开一个终端窗口，开启Hive，在Hive中创建hiverecord表，包含（id,buyer\_id,dt,ip,opt\_type）五个字段，字符类型均为varchar(100)，分隔符为‘\,’。

[hfut@master ~]$ hive

在Hive中创建hiverecord表：

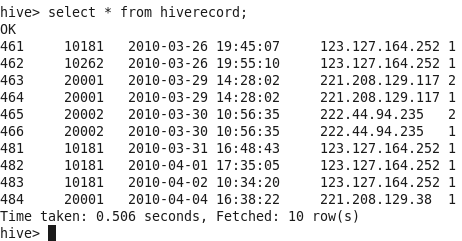
hive> create table hiverecord (id varchar(100),buyer\_id varchar(100),dt varchar(100), ip varchar(100), opt\_type varchar(100)) row format delimited fields terminated by ',' stored as textfile;

3. Sqoop上的操作

使用Sqoop将Mysql中record表导入Hive中。

[hfut@master ~]$ sqoop import --connect jdbc:mysql://master:3306/mydb?characterEncoding=UTF-8 --username u\_sqoop --password sqoop --table record --hive-import --hive-table hiverecord --fields-terminated-by ',' -m 1

4. 在hive下,查看Hive中hiverecord表



**六、将Hive中的数据导出到MySQL中**

使用Sqoop将Hive表hiverecord表中的数据，导出到MySQL中的recordfromhive表中。

1. MySQL上的操作

在Mysql中创建表recordfromhive。

mysql> create table recordfromhive like record;

2. Sqoop上的操作

[hfut@master ~]$ sqoop export --connect jdbc:mysql://master:3306/mydb?characterEncoding=UTF-8 --username u\_sqoop --password sqoop --table recordfromhive --export-dir /user/hive/warehouse/hiverecord/part-m-00000 --input-fields-terminated-by ','

3. 导入完成，查看Mysql中recordfromhive表。

mysql> select \* from recordfromhive;

