[**Flume-自定义 Source 读取 MySQL 数据**](https://www.cnblogs.com/jhxxb/p/11589851.html)

<https://www.cnblogs.com/jhxxb/p/11589851.html>

开源实现：<https://github.com/keedio/flume-ng-sql-source>

这里记录的是自己手动实现。

测试中要读取的表

CREATE TABLE `student` (

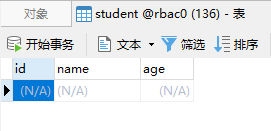
`id` int(**11**) NOT NULL AUTO\_INCREMENT,

`name` varchar(**255**) COLLATE utf8\_bin DEFAULT NULL,

`age` int(**11**) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=**3** DEFAULT CHARSET=utf8 COLLATE=utf8\_bin;



记录表（必须），告诉 Flume 每次从哪开始读取

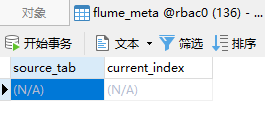
CREATE TABLE `flume\_meta` (

`source\_tab` varchar(**255**) COLLATE utf8\_bin NOT NULL,

`current\_index` bigint(**255**) DEFAULT NULL,

PRIMARY KEY (`source\_tab`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin;



## 一、编写自定义 Source

### 1.添加 pom 依赖

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com</groupId>

<artifactId>flume</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.apache.flume</groupId>

<artifactId>flume-ng-core</artifactId>

<version>1.9.0</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.27</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.8</source>

<target>1.8</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

</plugins>

</build>

</project>

[复制代码](javascript:void(0);)

### 2.编写类

MySQLSourceHelper，JDBC 工具类，主要是读取数据表和更新读取记录

[复制代码](javascript:void(0);)

package source;

import org.apache.flume.Context;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.math.BigInteger;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class MySQLSourceHelper {

private static final Logger LOG = LoggerFactory.getLogger(MySQLSourceHelper.class);

// 开始 id

private String startFrom;

private static final String DEFAULT\_START\_VALUE = "0";

// 表名

private String table;

// 用户传入的查询的列

private String columnsToSelect;

private static final String DEFAULT\_Columns\_To\_Select = "\*";

private static String dbUrl, dbUser, dbPassword, dbDriver;

private static Connection conn = null;

private static PreparedStatement ps = null;

// 获取 JDBC 连接

private static Connection getConnection() {

try {

Class.forName(dbDriver);

return DriverManager.getConnection(dbUrl, dbUser, dbPassword);

} catch (SQLException | ClassNotFoundException e) {

e.printStackTrace();

}

return null;

}

// 构造方法

MySQLSourceHelper(Context context) {

// 有默认值参数：获取 flume 任务配置文件中的参数，读不到的采用默认值

this.startFrom = context.getString("start.from", DEFAULT\_START\_VALUE);

this.columnsToSelect = context.getString("columns.to.select", DEFAULT\_Columns\_To\_Select);

// 无默认值参数：获取 flume 任务配置文件中的参数

this.table = context.getString("table");

dbUrl = context.getString("db.url");

dbUser = context.getString("db.user");

dbPassword = context.getString("db.password");

dbDriver = context.getString("db.driver");

conn = getConnection();

}

// 构建 sql 语句，以 id 作为 offset

private String buildQuery() {

StringBuilder execSql = new StringBuilder("select " + columnsToSelect + " from " + table);

return execSql.append(" where id ").append("> ").append(getStatusDBIndex(startFrom)).toString();

}

// 执行查询

List<List<Object>> executeQuery() {

try {

// 每次执行查询时都要重新生成 sql，因为 id 不同

String customQuery = buildQuery();

// 存放结果的集合

List<List<Object>> results = new ArrayList<>();

ps = conn.prepareStatement(customQuery);

ResultSet result = ps.executeQuery(customQuery);

while (result.next()) {

// 存放一条数据的集合（多个列）

List<Object> row = new ArrayList<>();

// 将返回结果放入集合

for (int i = 1; i <= result.getMetaData().getColumnCount(); i++) {

row.add(result.getObject(i));

}

results.add(row);

}

LOG.info("execSql:" + customQuery + "\tresultSize:" + results.size());

return results;

} catch (SQLException e) {

LOG.error(e.toString());

// 重新连接

conn = getConnection();

}

return null;

}

// 将结果集转化为字符串，每一条数据是一个 list 集合，将每一个小的 list 集合转化为字符串

List<String> getAllRows(List<List<Object>> queryResult) {

List<String> allRows = new ArrayList<>();

StringBuilder row = new StringBuilder();

for (List<Object> rawRow : queryResult) {

for (Object aRawRow : rawRow) {

if (aRawRow == null) {

row.append(",");

} else {

row.append(aRawRow.toString()).append(",");

}

}

allRows.add(row.toString());

row = new StringBuilder();

}

return allRows;

}

// 更新 offset 元数据状态，每次返回结果集后调用。必须记录每次查询的 offset 值，为程序中断续跑数据时使用，以 id 为 offset

void updateOffset2DB(BigInteger size) {

try {

// 以 source\_tab 做为 KEY，如果不存在则插入，存在则更新（每个源表对应一条记录）

String sql = "insert into flume\_meta VALUES('" + table + "','" + size + "') on DUPLICATE key update current\_index='" + size + "'";

LOG.info("updateStatus Sql:" + sql);

ps = conn.prepareStatement(sql);

ps.execute();

} catch (SQLException e) {

e.printStackTrace();

}

}

// 从 flume\_meta 表中查询出当前的 id 是多少

private BigInteger getStatusDBIndex(String startFrom) {

BigInteger dbIndex = new BigInteger(startFrom);

try {

ps = conn.prepareStatement("select current\_index from flume\_meta where source\_tab='" + table + "'");

ResultSet result = ps.executeQuery();

if (result.next()) {

String id = result.getString(1);

if (id != null) {

dbIndex = new BigInteger(id);

}

}

} catch (SQLException e) {

e.printStackTrace();

}

// 如果没有数据，则说明是第一次查询或者数据表中还没有存入数据，返回最初传入的值

return dbIndex;

}

// 关闭相关资源

void close() {

try {

ps.close();

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

public String getTable() {

return table;

}

}

[复制代码](javascript:void(0);)

MySQLSource，自定义 Source 类

[复制代码](javascript:void(0);)

package source;

import org.apache.flume.Context;

import org.apache.flume.Event;

import org.apache.flume.EventDeliveryException;

import org.apache.flume.PollableSource;

import org.apache.flume.conf.Configurable;

import org.apache.flume.event.SimpleEvent;

import org.apache.flume.source.AbstractSource;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.math.BigInteger;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

public class MySQLSource extends AbstractSource implements Configurable, PollableSource {

// 打印日志

private static final Logger LOG = LoggerFactory.getLogger(MySQLSource.class);

// sqlHelper

private MySQLSourceHelper sqlSourceHelper;

// 两次查询的时间间隔

private int queryDelay;

private static final int DEFAULT\_QUERY\_DELAY = 10000;

@Override

public long getBackOffSleepIncrement() {

return 0;

}

@Override

public long getMaxBackOffSleepInterval() {

return 0;

}

@Override

public void configure(Context context) {

// 初始化

sqlSourceHelper = new MySQLSourceHelper(context);

queryDelay = context.getInteger("query.delay", DEFAULT\_QUERY\_DELAY);

}

@Override

public Status process() throws EventDeliveryException {

try {

// 存放 event 的集合

List<Event> events = new ArrayList<>();

// 存放 event 头集合

HashMap<String, String> header = new HashMap<>();

header.put("table", sqlSourceHelper.getTable());

// 查询数据表

List<List<Object>> result = sqlSourceHelper.executeQuery();

// 如果有返回数据，则将数据封装为 event

if (!result.isEmpty()) {

List<String> allRows = sqlSourceHelper.getAllRows(result);

Event event = null;

for (String row : allRows) {

event = new SimpleEvent();

event.setHeaders(header);

event.setBody(row.getBytes());

events.add(event);

}

// 将 event 写入 channel

getChannelProcessor().processEventBatch(events);

// 更新数据表中的 offset 信息，取最后一条数据的第一列（id 列）

sqlSourceHelper.updateOffset2DB(new BigInteger(result.get(result.size()-1).get(0).toString()));

}

// 等待时长

Thread.sleep(queryDelay);

return Status.READY;

} catch (InterruptedException e) {

LOG.error("Error procesing row", e);

return Status.BACKOFF;

}

}

@Override

public synchronized void stop() {

LOG.info("Stopping sql source {} ...", getName());

try {

sqlSourceHelper.close();

} finally {

super.stop();

}

}

}

[复制代码](javascript:void(0);)

## 二、打包测试

### 1.打包上传

记得把 pom 依赖中的 MySQL 的 jar 包也传上去。

参考：<https://www.cnblogs.com/jhxxb/p/11582804.html>

### 2.编写 flume 配置文件

**mysql.conf**

[复制代码](javascript:void(0);)

# Name the components on this agent

a1.sources = r1

a1.sinks = k1

a1.channels = c1

# Describe/configure the source

a1.sources.r1.type = source.MySQLSource

a1.sources.r1.db.driver = com.mysql.jdbc.Driver

a1.sources.r1.db.url = jdbc:mysql://192.168.8.136:3306/rbac0

a1.sources.r1.db.user = root

a1.sources.r1.db.password = root

a1.sources.r1.table = student

# a1.sources.r1.columns.to.select = \*

# a1.sources.r1.start.from = 0

# Describe the sink

a1.sinks.k1.type = logger

# Describe the channel

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

[复制代码](javascript:void(0);)

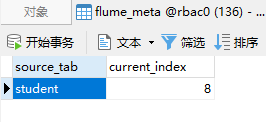
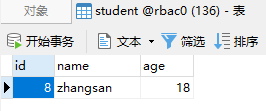
启动

cd /opt/apache-flume-1.9.0-bin

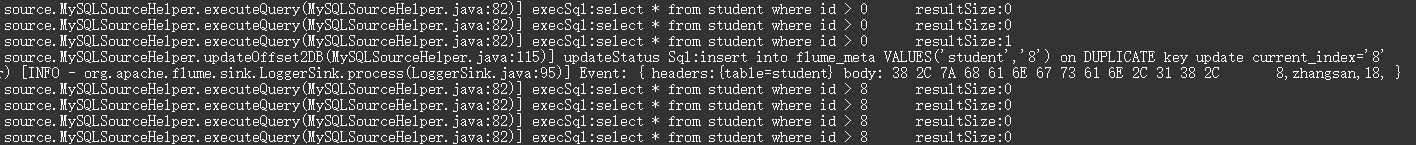
bin/flume-ng agent --conf conf/ --name a1 --conf-file /tmp/flume-job/mysql.conf -Dflume.root.logger=INFO,console

向监控表插入数据

INSERT student VALUES(NULL,'zhangsan',**18**);



Flume 的控制台日志



cd /opt/apache-flume-1.9.0-bin

bin/flume-ng agent --conf conf/ --name a1 --conf-file /data/cona\_tbds/cona3/env/flume/conf/mysql.conf -Dflume.root.logger=INFO,console

INSERT student VALUES(NULL,'zhangsan',18);