**利用canal实现mysql数据库数据与redis同步**



[Mr\_LGZ](https://blog.csdn.net/lyl0724) 2018-05-31 18:25:02 https://csdnimg.cn/release/blogv2/dist/pc/img/articleReadEyes.png 10619 https://csdnimg.cn/release/blogv2/dist/pc/img/tobarCollect.png 收藏 26

分类专栏： [canal](https://blog.csdn.net/lyl0724/category_7704501.html) 文章标签： [redis](https://www.csdn.net/tags/NtjacgwsODIzOC1ibG9n.html" \t "_blank) [mysql](https://www.csdn.net/tags/MtTaEg5sOTYwNC1ibG9n.html" \t "_blank) [canal](https://www.csdn.net/tags/MtTaEg0sNDAyNTItYmxvZwO0O0OO0O0O.html) 版权

**一、MySQL配置**

使用的MySQL版本：mysql 5.6.40（其他自行测试）   
查看数据版本：mysql -V   
这里写图片描述

修改mysql的配置文件my.cnf: vim /etc/my.cnf   
在[mysqld]下添加：

# 开启mysql的binlog模块

log-bin=mysql-bin

binlog-format=ROW

# server\_id需保证唯一，不能和canal的slaveId重复

server\_id=12

# 需要同步的数据库名称

binlog-do-db=test\_canal

# 忽略的数据库，建议填写

binlog-ignore-db=mysql

# 启动mysql时不启动grant-tables授权表

skip-grant-tables

配置完后重启数据库   
登陆数据库：mysql -uroot -p   
登陆mysql后，创建一个MySQL用户canal并授予权限：

CREATE USER canal IDENTIFIED BY 'canal';

GRANT ALL PRIVILEGES ON test\_canal.user TO 'canal'@'%'

FLUSH PRIVILEGES;

至此，MySQL配置完成

**二、canal部署与配置**

下载canal：<https://github.com/alibaba/canal/releases/>   
这里以1.0.25为例   
   
下载完后解压到canal目录（canal目录实现先创建好）：tar -zxvf canal.deployer-1.0.25.tar.gz /usr/local/canal   
解压后得到四个目录bin、conf、lib、logs   
这里写图片描述   
有两个重要的配置文件：

canal/conf/example/instance.properties

canal/conf/canal.properties

canal.properties可保持不变，默认的端口时五个1：11111   
instance.properties需要配置：



启动canal: ./bin/startup.sh   
查看是否正常启动成功，需要查看两个日志文件：logs/canal/canal.log 和logs/example/example.log。

logs/canal/canal.log文件中有如下内容：the canal server is running now ......

[main] INFO com.alibaba.otter.canal.deployer.CanalLauncher - ## start the canal server.

[main] INFO com.alibaba.otter.canal.deployer.CanalController - ## start the canal

[main] INFO com.alibaba.otter.canal.deployer.CanalLauncher - ## the canal server is running now ......

logs/example/example.log文件中有如下内容：start successful....

[main] INFO c.a.o.c.i.spring.support.PropertyPlaceholderConfigurer - Loading properties file from class path resource [canal.properties]

[main] INFO c.a.o.c.i.spring.support.PropertyPlaceholderConfigurer - Loading properties file from class path resource [example/instance.properties]

[main] INFO c.a.otter.canal.instance.spring.CanalInstanceWithSpring - start CannalInstance for 1-example

[main] INFO c.a.otter.canal.instance.core.AbstractCanalInstance - start successful....

证明启动成功

**三、Java的canal客户端和jedis**

在项目maven中添加redis和canal依赖

<dependency>

<groupId>com.alibaba.otter</groupId>

<artifactId>canal.client</artifactId>

<version>1.0.25</version>

</dependency>

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

<version>2.9.0</version>

</dependency>

**redis工具类**：

package com.test.canal;

import redis.clients.jedis.Jedis;

import redis.clients.jedis.JedisPool;

import redis.clients.jedis.JedisPoolConfig;

/\*\*

\* @author lgz

\*/

public class RedisUtil {

private static Jedis jedis = null;

public static synchronized Jedis getJedis() {

if (jedis == null) {

jedis = new Jedis("192.168.188.128", 6379);

jedis.auth("redis1234");

}

return jedis;

}

public static boolean existKey(String key) {

return getJedis().exists(key);

}

public static void delKey(String key) {

getJedis().del(key);

}

public static String stringGet(String key) {

return getJedis().get(key);

}

public static String stringSet(String key, String value) {

return getJedis().set(key, value);

}

public static void hashSet(String key, String field, String value) {

getJedis().hset(key, field, value);

}

}

**canal客户端**：

package com.test.canal*;*

import com.alibaba.fastjson.JSONObject*;*

import com.alibaba.otter.canal.client.CanalConnector*;*

import com.alibaba.otter.canal.client.CanalConnectors*;*

import com.alibaba.otter.canal.protocol.CanalEntry.\**;*

import com.alibaba.otter.canal.protocol.Message*;*

import java.net.InetSocketAddress*;*

import java.util.List*;*

*/\*\**

*\* @author lgz*

*\*/*

public class CanalClient {

public static void main(String args[]) {

CanalConnector connector = CanalConnectors.newSingleConnector(new InetSocketAddress("192.168.188.128",

11111), "example", "", "")*;*

int batchSize = 100*;*

try {

connector.connect()*;*

connector.subscribe(".\*\\..\*")*;*

connector.rollback()*;*

while (true) {

// 获取指定数量的数据

Message message = connector.getWithoutAck(batchSize)*;*

long batchId = message.getId()*;*

int size = message.getEntries().size()*;*

System.out.println("batchId = " + batchId)*;*

System.out.println("size = " + size)*;*

if (batchId == -1 || size == 0) {

try {

Thread.sleep(1000)*;*

} catch (InterruptedException e) {

e.printStackTrace()*;*

}

} else {

printEntry(message.getEntries())*;*

}

// 提交确认

connector.ack(batchId)*;*

// connector.rollback(batchId)*; // 处理失败, 回滚数据*

}

} finally {

connector.disconnect()*;*

}

}

private static void printEntry(List<Entry> entrys) {

for (Entry entry : entrys) {

if (entry.getEntryType() == EntryType.TRANSACTIONBEGIN || entry.getEntryType() == EntryType.TRANSACTIONEND) {

continue*;*

}

RowChange rowChage = null*;*

try {

rowChage = RowChange.parseFrom(entry.getStoreValue())*;*

} catch (Exception e) {

throw new RuntimeException("ERROR ## parser of eromanga-event has an error , data:" + entry.toString(),

e)*;*

}

EventType eventType = rowChage.getEventType()*;*

System.out.println(String.format("================> binlog[%s:%s] , name[%s,%s] , eventType : %s",

entry.getHeader().getLogfileName(), entry.getHeader().getLogfileOffset(),

entry.getHeader().getSchemaName(), entry.getHeader().getTableName(),

eventType))*;*

for (RowData rowData : rowChage.getRowDatasList()) {

if (eventType == EventType.DELETE) {

redisDelete(rowData.getBeforeColumnsList())*;*

} else if (eventType == EventType.INSERT) {

redisInsert(rowData.getAfterColumnsList())*;*

} else {

System.out.println("-------> before")*;*

printColumn(rowData.getBeforeColumnsList())*;*

System.out.println("-------> after")*;*

redisUpdate(rowData.getAfterColumnsList())*;*

}

}

}

}

private static void printColumn(List<Column> columns) {

for (Column column : columns) {

System.out.println(column.getName() + " : " + column.getValue() + " update=" + column.getUpdated())*;*

}

}

private static void redisInsert(List<Column> columns) {

JSONObject json = new JSONObject()*;*

for (Column column : columns) {

json.put(column.getName(), column.getValue())*;*

}

if (columns.size() > 0) {

RedisUtil.stringSet("user:" + columns.get(0).getValue(), json.toJSONString())*;*

}

}

private static void redisUpdate(List<Column> columns) {

JSONObject json = new JSONObject()*;*

for (Column column : columns) {

json.put(column.getName(), column.getValue())*;*

}

if (columns.size() > 0) {

RedisUtil.stringSet("user:" + columns.get(0).getValue(), json.toJSONString())*;*

}

}

private static void redisDelete(List<Column> columns) {

JSONObject json = new JSONObject()*;*

for (Column column : columns) {

json.put(column.getName(), column.getValue())*;*

}

if (columns.size() > 0) {

RedisUtil.delKey("user:" + columns.get(0).getValue())*;*

}

}

}

**四、同步测试**

在test\_canal数据库新建一个表：

CREATE TABLE `user` (

`id` int(11) NOT NULL,

`name` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

打印：

size = 1

================> binlog[mysql-bin.000005:4635] , name[test\_canal,user] , eventType : CREATE

在user表中添加一条数据：

INSERT INTO `user` (id,name) VALUES (1,'zhangsan');

打印：

size = 3

================> binlog[mysql-bin.000005:5001] , name[test\_canal,user] , eventType : INSERT

从redis过去数据：

String key = "user:1";

String value = null;

if (RedisUtil.existKey(key)) {

value = RedisUtil.stringGet(key);

}

System.out.println(value);

打印：

{"name":"zhangsan","id":"1"}

* 1

至此，证明redis和mysql同步已经成功   
本人测试增删改都可以成功。