尚硅谷大数据技术之 Maxwell

（作者：尚硅谷大数据研发部）

版本：V3.0

第 1 章 Maxwell 概述

1.1 Maxwell 定义

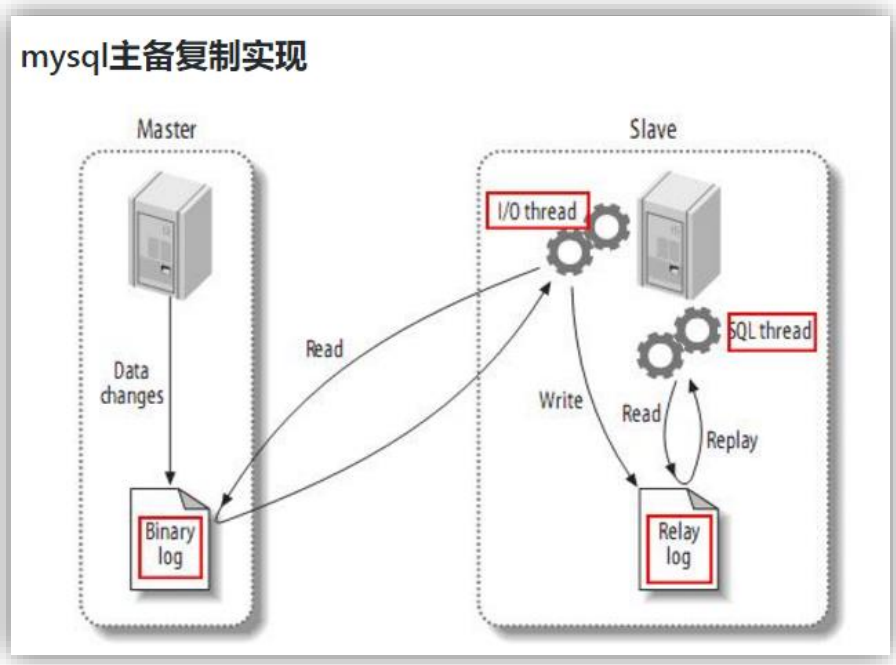
Maxwell 是由美国 Zendesk 开源，用 Java 编写的 MySQL 实时抓取软件。 实时读取 MySQL 二进制日志 Binlog，并生成 JSON 格式的消息，作为生产者发送给 Kafka，Kinesis、RabbitMQ、Redis、Google Cloud Pub/Sub、文件或其它平台的应用程序。

官网地址：http://maxwells-daemon.io/

1.2 Maxwell 工作原理

1.2.1 MySQL 主从复制过程

- Master 主库将改变记录，写到二进制日志(binary log)中
- Slave 从库向 mysql master 发送 dump 协议，将 master 主库的 binary log events 拷贝到它的中继日志(relay log)；
- Slave 从库读取并重做中继日志中的事件，将改变的数据同步到自己的数据库。



1.2.2 Maxwell 的工作原理

Maxwell 的工作原理很简单，就是**把自己伪装成 MySQL 的一个 slave，然后以 slave 的身份假装从 MySQL(master)复制数据。**

1.2.3 MySQL 的 binlog

(1) 什么是 binlog

MySQL 的二进制日志可以说 MySQL 最重要的日志了，它记录了所有的 DDL 和 DML(除了数据查询语句)语句，以事件形式记录，还包含语句所执行的消耗的时间，MySQL 的二进制日志是事务安全型的。

一般来说开启二进制日志大概会有 1%的性能损耗。二进制有两个最重要的使用场景:

- 其一：MySQL Replication 在 Master 端开启 binlog，Master 把它的二进制日志传递给 slaves 来达到 master-slave 数据一致的目的。
- 其二：自然就是数据恢复了，通过使用 mysqlbinlog 工具来使恢复数据。

二进制日志包括两类文件：二进制日志索引文件（文件名后缀为.index）用于记录所有的二进制文件，二进制日志文件（文件名后缀为.00000*）记录数据库所有的 DDL 和 DML(除了数据查询语句)语句事件。

(2) binlog 的开启

- 找到 MySQL 配置文件的位置
- Linux: /etc/my.cnf

  如果/etc 目录下没有，可以通过 locate my.cnf 查找位置
- Windows: \my.ini
- 在 mysql 的配置文件下,修改配置

  在[mysqld] 区块，设置/添加 **log-bin=mysql-bin**

  这个表示 binlog 日志的前缀是 mysql-bin，以后生成的日志文件就是 mysql-bin.000001 的文件后面的数字按顺序生成，每次 mysql 重启或者到达单个文件大小的阈值时，新生一个文件，按顺序编号。

(3) binlog 的分类设置

mysql binlog 的格式有三种，分别是 STATEMENT,MIXED,ROW。

在配置文件中可以选择配置 **binlog_format= statement|mixed|row**

- 三种格式的区别:
  - **statement**

    语句级，binlog 会记录每次一执行写操作的语句。

    相对 row 模式节省空间，但是可能产生不一致性，比如

    update test set create_date=now();

    如果用 binlog 日志进行恢复，由于执行时间不同可能产生的数据就不同。

    优点： 节省空间

    缺点： 有可能造成数据不一致。
  - **row**

    行级， binlog 会记录每次操作后每行记录的变化。

    优点：保持数据的绝对一致性。因为不管 sql 是什么，引用了什么函数，他只记录执行后的效果。

    缺点：占用较大空间。
  - **mixed**

    混合级别，statement 的升级版，一定程度上解决了 statement 模式因为一些情况而造成的数据不一致问题。

默认还是 statement，在某些情况下，譬如：

当函数中包含 UUID() 时；

包含 AUTO_INCREMENT 字段的表被更新时；

执行 INSERT DELAYED 语句时；

用 UDF 时；

会按照 ROW 的方式进行处理

优点：节省空间，同时兼顾了一定的一致性。

缺点：还有些极个别情况依旧会造成不一致，另外 statement 和 mixed 对于需要对 binlog 监控的情况都不方便。

综合上面对比，Maxwell 想做监控分析，选择 row 格式比较合适

1.3 Maxwell 与 Canal 的对比

对比	Canal	Maxwell
语言	java	java
数据格式	格式自由	json
采集数据模式	增量	全量/增量
数据落地	定制	支持 kafka 等多种平台
HA	支持	支持

第 2 章 Maxwell 使用

2.1 Maxwell 安装部署

2.1.1 安装地址

（1）Maxwell 官网地址：http://maxwells-daemon.io/

（2）文档查看地址：http://maxwells-daemon.io/quickstart/

2.1.2 安装部署

（1）软件基础，读者需要提前安装好 kafka 和 MySQL，此文档不再赘述。

（2）上传 maxwell-1.29.2.tar.gz 到/opt/software 下

（3）解压 maxwell-1.29.2.tar.gz 的安装包到/opt/module 下

```
[atguigu@hadoop102 software]$ tar -zxvf maxwell-1.29.2.tar.gz -
C /opt/module/
```

2.1.3 MySQL 环境准备

（1）修改 mysql 的配置文件，开启 MySQL Binlog 设置

```
atguigu@hadoop102 software]$ sudo vim /etc/my.cnf
在[mysqld]模块下添加一下内容
[mysqld]
server_id=1
log-bin=mysql-bin
binlog_format=row
#binlog-do-db=test_maxwell

并重启 Mysql 服务
[atguigu@hadoop102 software]$ sudo systemctl restart mysqld

登录 mysql 并查看是否修改完成
[atguigu@hadoop102 ~]$ mysql -uroot -p123456
mysql> show variables like '%binlog%';
查看下列属性
binlog_format                           | ROW
```

（2）进入/var/lib/mysql 目录，查看 MySQL 生成的 binlog 文件

```
[atguigu@hadoop102 ~]$ cd /var/lib/mysql
[atguigu@hadoop102 mysql]$ sudo ls -l
总用量 188500
-rw-r-----. 1 mysql mysql     154 11 月 17 16:30 mysql-
bin.000001
-rw-r-----. 1 mysql mysql      19 11 月 17 16:30 mysql-
bin.index
```

注：MySQL 生成的 binlog 文件初始大小一定是 154 字节，然后前缀是 log-bin 参数配置的，后缀是默认从.000001，然后依次递增。除了 binlog 文件文件以外，MySQL 还会额外生产一个.index 索引文件用来记录当前使用的 binlog 文件。

2.1.4 初始化 Maxwell 元数据库

（1）在 MySQL 中建立一个 maxwell 库用于存储 Maxwell 的元数据

```
[atguigu@hadoop102 module]$ mysql -uroot -p123456
mysql> CREATE DATABASE maxwell;
```

（2）设置 mysql 用户密码安全级别

```
mysql> set global validate_password_length=4;
mysql> set global validate_password_policy=0;
```

（3）分配一个账号可以操作该数据库

```
mysql> GRANT ALL  ON maxwell.* TO 'maxwell'@'%' IDENTIFIED BY
```

```
'123456';
```

（4）分配这个账号可以监控其他数据库的权限

```
mysql> GRANT  SELECT ,REPLICATION SLAVE , REPLICATION CLIENT  ON
*.* TO maxwell@'%';
```

（5）刷新 mysql 表权限

```
mysql> flush privileges;
```

2.1.5 Maxwell 进程启动

Maxwell 进程启动方式有如下两种：

（1）使用命令行参数启动 Maxwell 进程

```
[atguigu@hadoop102      maxwell-1.29.2]$      bin/maxwell      --
user='maxwell'   --password='123456'   --host='hadoop102'      --
producer=stdout
```

--user 连接 mysql 的用户

--password 连接 mysql 的用户的密码

--host mysql 安装的主机名

--producer 生产者模式(stdout：控制台　kafka：kafka 集群)

（2）修改配置文件，定制化启动 Maxwell 进程

```
[atguigu@hadoop102              maxwell-1.29.2]$              cp
config.properties.example config.properties
[atguigu@hadoop102 maxwell-1.29.2]$ vim config.properties
[atguigu@hadoop102      maxwell-1.29.2]$      bin/maxwell      --
config ./config.properties
```

2.2 Maxwell 入门案例

2.2.1 监控 Mysql 数据并在控制台打印

1）实现步骤：

（1）运行 maxwell 来监控 mysql 数据更新

```
 [atguigu@hadoop102      maxwell-1.29.2]$      bin/maxwell      --
user='maxwell'    --password='123456'    --host='hadoop102'    --
producer=stdout
```

（2）向 mysql 的 test_maxwell 库的 test 表插入一条数据，查看 maxwell 的控制台输出

```
mysql> insert into test values(1,'aaa');

{
    "database": "test_maxwell",   --库名
    "table": "test",    --表名
    "type": "insert",    --数据更新类型
    "ts": 1637244821,   --操作时间
```

```
    "xid": 8714,          --操作 id
    "commit": true,       --提交成功
    "data": {             --数据
        "id": 1,
        "name": "aaa"
    }
}
```

（3）向 mysql 的 test_maxwell 库的 test 表同时插入 3 条数据，控制台出现了 3 条 json 日志，说明 maxwell 是以数据行为单位进行日志的采集的。

```
mysql> INSERT INTO test VALUES(2,'bbb'),(3,'ccc'),(4,'ddd');

{"database":"test_maxwell","table":"test","type":"insert","ts"
:1637245127,"xid":9129,"xoffset":0,"data":{"id":2,"name":"bbb"
}}
{"database":"test_maxwell","table":"test","type":"insert","ts"
:1637245127,"xid":9129,"xoffset":1,"data":{"id":3,"name":"ccc"
}}
{"database":"test_maxwell","table":"test","type":"insert","ts"
:1637245127,"xid":9129,"commit":true,"data":{"id":4,"name":"dd
d"}}

mysql> update test set name='zaijian' where id =1;

{"database":"test_maxwell","table":"test","type":"update","ts"
:1631618614,"xid":535,"commit":true,"data":{"id":1,"name":"zai
jian"},"old":{"name":"nihao"}}
```

（4）修改 test_maxwell 库的 test 表的一条数据，查看 maxwell 的控制台输出

```
mysql> update test set name='abc' where id =1;

{
    "database": "test_maxwell",
    "table": "test",
    "type": "update",
    "ts": 1637245338,
    "xid": 9418,
    "commit": true,
    "data": {               --修改后的数据
        "id": 1,
        "name": "abc"
    },
    "old": {                --修改前的数据
        "name": "aaa"
    }
}
```

（5）删除 test_maxwell 库的 test 表的一条数据，查看 maxwell 的控制台输出

```
mysql> DELETE FROM test WHERE id =1;

{
    "database": "test_maxwell",
    "table": "test",
    "type": "delete",
    "ts": 1637245630,
    "xid": 9816,
    "commit": true,
```

```
    "data": {
        "id": 1,
        "name": "abc"
    }
}
```

2.2.2 监控 Mysql 数据输出到 kafka

1）实现步骤：

（1）启动 zookeeper 和 kafka

```
[atguigu@hadoop102 bin]$ jpsall
=============== hadoop102 ===============
3511 QuorumPeerMain
4127 Kafka
=============== hadoop103 ===============
1885 Kafka
1342 QuorumPeerMain
=============== hadoop104 ===============
1345 QuorumPeerMain
1886 Kafka
```

（2）启动 Maxwell 监控 binlog

```
atguigu@hadoop102 maxwell-1.29.2]$ bin/maxwell --user='maxwell' --password='123456'  --host='hadoop102'  --producer=kafka  --kafka.bootstrap.servers=hadoop102:9092 --kafka_topic=maxwell
```

（3）打开 kafka 的控制台的消费者消费 maxwell 主题

```
[atguigu@hadoop102 ~]$ kafka-console-consumer.sh --bootstrap-server hadoop102:9092 --topic maxwell
```

（4）向 test_maxwell 库的 test 表再次插入一条数据

```
mysql> insert into test values (5,'eee');
```

（5）通过 kafka 消费者来查看到了数据，说明数据成功传入 kafka

```
{"database":"test_maxwell","table":"test","type":"insert","ts":1637245889,"xid":10155,"commit":true,"data":{"id":5,"name":"eee"}}
```

2）kafka 主题数据的分区控制

在公司生产环境中，我们一般都会用 maxwell 监控多个 mysql 库的数据，然后将这些数据发往 kafka 的一个主题 Topic，并且这个主题也肯定是多分区的，为了提高并发度。那么如何控制这些数据的分区问题，就变得至关重要，实现步骤如下：

（1）修改 maxwell 的配置文件，定制化启动 maxwell 进程

```
[atguigu@hadoop102 maxwell-1.29.2]$ vim config.properties

# tl;dr config
log_level=info

producer=kafka
kafka.bootstrap.servers=hadoop102:9092
```

```
# mysql login info
host=hadoop102
user=maxwell
password=123456



#      *** kafka ***

# list of kafka brokers
#kafka.bootstrap.servers=hosta:9092,hostb:9092

# kafka topic to write to
# this can be static, e.g. 'maxwell', or dynamic, e.g.
namespace_%{database}_%{table}
# in the latter case 'database' and 'table' will be replaced
with the values for the row being processed
kafka_topic=maxwell3


#         *** partitioning ***

# What part of the data do we partition by?
#producer_partition_by=database # [database, table,
primary_key, transaction_id, column]
producer_partition_by=database

控制数据分区模式，可选模式有 库名，表名，主键，列名
# specify what fields to partition by when using
producer_partition_by=column
# column separated list.
#producer_partition_columns=name

# when using producer_partition_by=column, partition by this
when
# the specified column(s) don't exist.
#producer_partition_by_fallback=database
```

（2）手动创建一个 3 个分区的 topic，名字就叫做 maxwell3

```
[atguigu@hadoop102 maxwell-1.29.2]$ kafka-topics.sh --zookeeper
hadoop102:2181,hadoop103:2181,hadoop104:2181/kafka  --create --
replication-factor 2 --partitions 3 --topic maxwell3
```

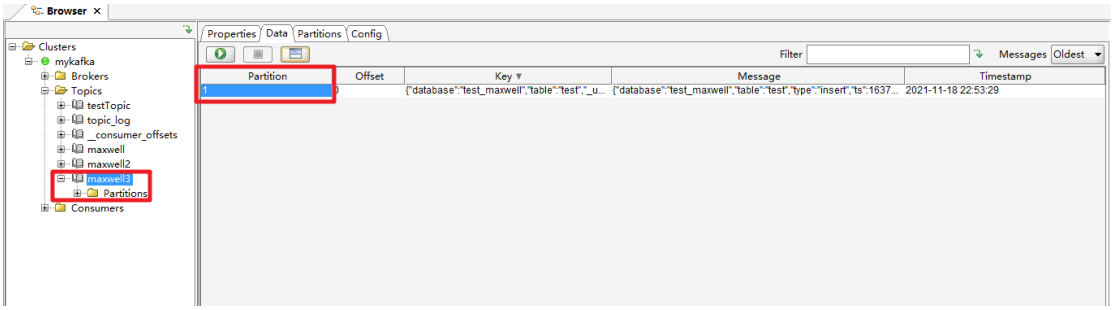
（3）利用配置文件启动 Maxwell 进程

```
[atguigu@hadoop102      maxwell-1.29.2]$      bin/maxwell      --
config ./config.properties
```

（4）向 test_maxwell 库的 test 表再次插入一条数据

```
mysql> insert into test_maxwell.test values (6,'fff');
```

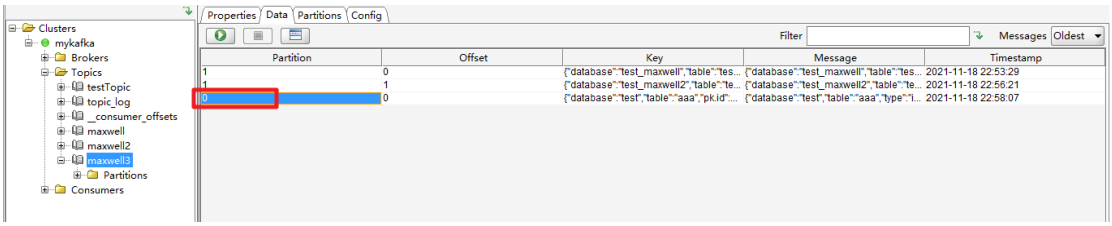
（5）通过 kafka tool 工具查看，此条数据进入了 maxwell3 主题的 1 号分区



（6）向 test 库的 aaa 表插入一条数据

```
mysql> insert into test_maxwell2.test values (23,'dd');
```

（7）通过 kafka tool 工具查看，此条数据进入了 maxwell3 主题的 0 号分区，说明库名会对数据进入的分区造成影响。



2.2.3 监控 Mysql 指定表数据输出控制台

（1）运行 maxwell 来监控 mysql 指定表数据更新

```
[atguigu@hadoop102 maxwell-1.29.2]$ bin/maxwell --user='maxwell' --password='123456' --host='hadoop102' --filter 'exclude: *.*, include:test_maxwell.test' --producer=stdout
```

（2）向 test_maxwell.test 表插入一条数据，查看 maxwell 的监控

```
mysql> insert into test_maxwell.test values(7,'ggg');

{"database":"test_maxwell","table":"test","type":"insert","ts":1637247760,"xid":11818,"commit":true,"data":{"id":7,"name":"ggg"}}
```

（3）向 test_maxwell.test2 表插入一条数据，查看 maxwell 的监控

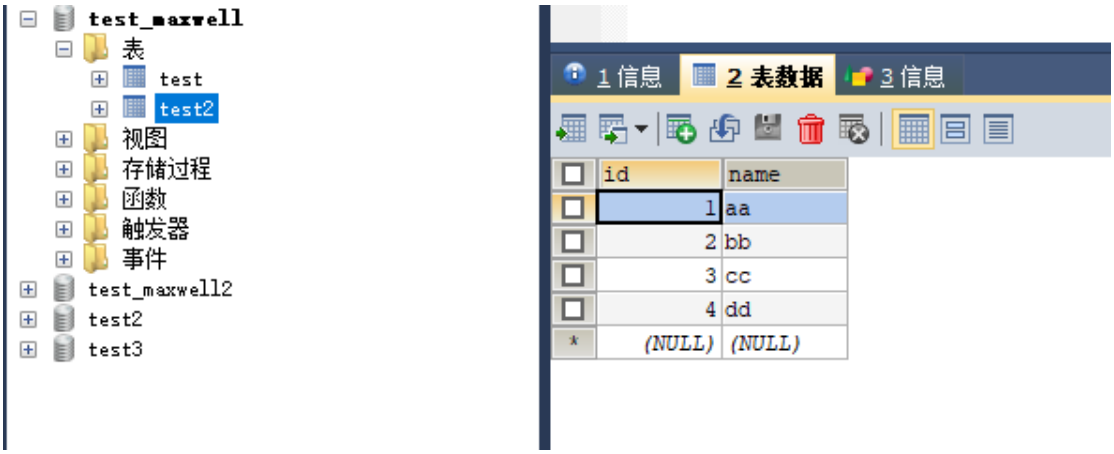
```
mysql> insert into test1 values(1,'nihao');
本次没有收到任何信息
说明 include 参数生效，只能监控指定的 mysql 表的信息
```

注：还可以设置 include:test_maxwell.*，通过此种方式来监控 mysql 某个库的所有表，也就是说过滤整个库。读者可以自行测试。

2.2.4 监控 Mysql 指定表全量数据输出控制台，数据初始化

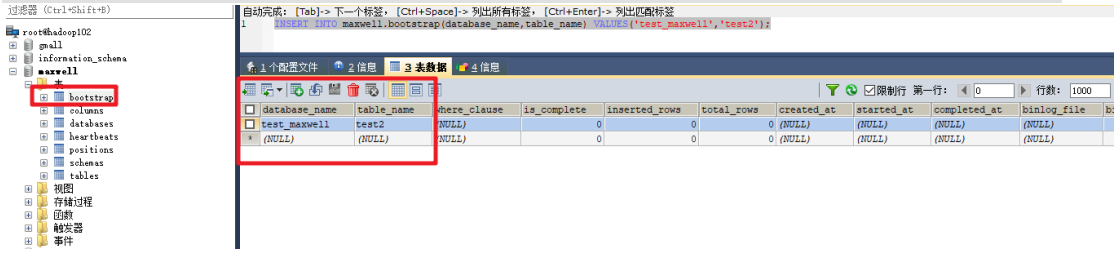
Maxwell 进程默认只能监控 mysql 的 binlog 日志的新增及变化的数据，但是 Maxwell 是支持数据初始化的，可以通过修改 Maxwell 的元数据，来对 MySQL 的某张表进行数据初始化，也就是我们常说的全量同步。具体操作步骤如下：

需求：将 test_maxwell 库下的 test2 表的四条数据，全量导入到 maxwell 控制台进行打印。



（1）修改 Maxwell 的元数据，触发数据初始化机制，在 mysql 的 maxwell 库中 bootstrap 表中插入一条数据，写明需要全量数据的库名和表名。

```
mysql> insert into maxwell.bootstrap(database_name,table_name)
values('test_maxwell','test2');
```



（2）启动 maxwell 进程，此时初始化程序会直接打印 test2 表的所有数据

```
[atguigu@hadoop102 maxwell-1.29.2]$ bin/maxwell --
user='maxwell' --password='123456' --host='hadoop102' --
producer=stdout

Using kafka version: 1.0.0
23:15:38,841 WARN  MaxwellMetrics - Metrics will not be
exposed: metricsReportingType not configured.
23:15:39,110 INFO  Maxwell - Maxwell v1.22.0 is booting
(StdoutProducer), starting at Position[BinlogPosition[mysql-
bin.000004:611096], lastHeartbeat=1637248429242]
23:15:39,194 INFO  MysqlSavedSchema - Restoring schema id 6
(last modified at Position[BinlogPosition[mysql-
bin.000004:517625], lastHeartbeat=1637246435111])
```

```
23:15:39,299 INFO  MysqlSavedSchema - Restoring schema id 1
(last modified at Position[BinlogPosition[mysql-
bin.000004:158612], lastHeartbeat=0])
23:15:39,342 INFO  MysqlSavedSchema - beginning to play
deltas...
23:15:39,343 INFO  MysqlSavedSchema - played 5 deltas in 1ms
{"database":"test_maxwell","table":"test2","type":"bootstrap-
start","ts":1637248539,"data":{}}
23:15:39,367 INFO  SynchronousBootstrapper - bootstrapping
started for test_maxwell.test2
23:15:39,369 INFO  BinlogConnectorReplicator - Setting initial
binlog pos to: mysql-bin.000004:611096
{"database":"test_maxwell","table":"test2","type":"bootstrap-
insert","ts":1637248539,"data":{"id":1,"name":"aa"}}
{"database":"test_maxwell","table":"test2","type":"bootstrap-
insert","ts":1637248539,"data":{"id":2,"name":"bb"}}
{"database":"test_maxwell","table":"test2","type":"bootstrap-
insert","ts":1637248539,"data":{"id":3,"name":"cc"}}
{"database":"test_maxwell","table":"test2","type":"bootstrap-
insert","ts":1637248539,"data":{"id":4,"name":"dd"}}
{"database":"test_maxwell","table":"test2","type":"bootstrap-
complete","ts":1637248539,"data":{}}
23:15:39,387 INFO  SynchronousBootstrapper - bootstrapping
ended for #8 test_maxwell.test2
23:15:39,465 INFO  BinaryLogClient - Connected to
hadoop102:3306 at mysql-bin.000004/611096 (sid:6379, cid:108)
23:15:39,465 INFO  BinlogConnectorLifecycleListener - Binlog
connected.
```

（3） 当数据全部初始化完成以后，Maxwell 的元数据会变化

is_complete 字段从 0 变为 1

start_at 字段从 null 变为具体时间(数据同步开始时间)

complete_at 字段从 null 变为具体时间(数据同步结束时间)