尚硅谷大数据技术之 Griffin

(作者：尚硅谷大数据研发部)

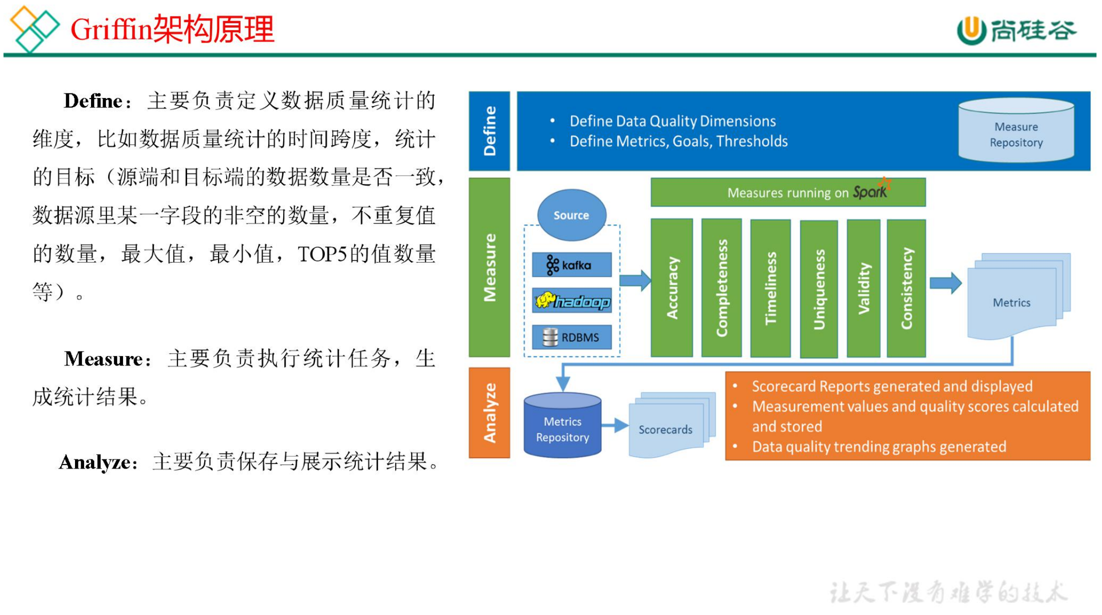
版本：V2.0

第 1 章 Griffin 入门

1.1 Griffin 概述

Apache Griffin 是一个开源的大数据数据质量解决方案，它支持批处理和流模式两种数据质量检测方式，可以从不同维度度量数据资产，从而提升数据的准确度和可信度。例如：离线任务执行完毕后检查源端和目标端的数据数量是否一致，源表的数据空值等。

1.2 Griffin 架构原理



Griffin架构原理

Define：主要负责定义数据质量统计的维度，比如数据质量统计的时间跨度，统计的目标（源端和目标端的数据数量是否一致，数据源里某一字段的非空的数量，不重复值的数量，最大值，最小值，TOP5的值数量等）。

Measure：主要负责执行统计任务，生成统计结果。

Analyze：主要负责保存与展示统计结果。

Define
Define Data Quality Dimensions
Define Metrics, Goals, Thresholds
Measure Repository
Measure
Measures running on Spark
Source
kafka
hadoop
RDBMS
Accuracy
Completeness
Timeliness
Uniqueness
Validity
Consistency
Metrics
Analyze
Metrics Repository
Scorecards
Scorecard Reports generated and displayed
Measurement values and quality scores calculated and stored
Data quality trending graphs generated
让天下没有难学的技术

第 2 章 Griffin 安装及使用

2.1 安装前环境准备

2.1.1 安装 ES5.2

1）上传 elasticsearch-5.2.2.tar.gz 到 hadoop102 的/opt/software 目录，并解压到/opt/module 目录

```
[atguigu@hadoop102 software]$ tar -zxvf elasticsearch-5.2.2.tar.gz -C /opt/module/
```

2）修改/opt/module/elasticsearch-5.2.2/config/elasticsearch.yml 配置文件

```
[atguigu@hadoop102 config]$ vim elasticsearch.yml

network.host: hadoop102
http.port: 9200
http.cors.enabled: true
http.cors.allow-origin: "*"
bootstrap.memory_lock: false
bootstrap.system_call_filter: false
```

3）修改 Linux 系统配置文件/etc/security/limits.conf

```
[atguigu@hadoop102      elasticsearch-5.2.2]$      sudo      vim
/etc/security/limits.conf

#添加如下内容
* soft nproc 65536
* hard nproc 65536
* soft nofile 65536
* hard nofile 65536
```

```
[atguigu@hadoop102      elasticsearch-5.2.2]$      sudo      vim
/etc/sysctl.conf

#添加
vm.max_map_count=655360
```

```
[atguigu@hadoop102      elasticsearch-5.2.2]$      sudo      vim
/etc/security/limits.d/90-nproc.conf
#修改配置
*          soft    nproc     2048
```

```
[atguigu@hadoop102 elasticsearch-5.2.2]$ sudo sysctl -p
```

4）需要重新启动虚拟机

```
[atguigu@hadoop102 elasticsearch-5.2.2]$ su root
root@hadoop102 elasticsearch-5.2.2]# reboot
```

5）在/opt/module/elasticsearch-5.2.2 路径上，启动 ES

```
[atguigu@hadoop102         elasticsearch-5.2.2]$         nohup
/opt/module/elasticsearch-5.2.2/bin/elasticsearch &
```

6）在 ES 里创建 griffin 索引

```
[atguigu@hadoop102 ~]$ curl -XPUT http://hadoop102:9200/griffin
-d '
{
   "aliases": {},
   "mappings": {
       "accuracy": {
           "properties": {
               "name": {
                   "fields": {
                       "keyword": {
                           "ignore_above": 256,
                           "type": "keyword"
                       }
```

```
                    },
                    "type": "text"
                },
                "tmst": {
                    "type": "date"
                }
            }
        }
    },
    "settings": {
        "index": {
            "number_of_replicas": "2",
            "number_of_shards": "5"
        }
    }
}
'
```

2.1.2 启动 HDFS & Yarn 服务

```
[atguigu@hadoop102 hadoop-2.7.2]$ sbin/start-dfs.sh
[atguigu@hadoop103 hadoop-2.7.2]$ sbin/start-yarn.sh
```

2.1.3 修改 Hive 配置

注意：Hive 版本至少 2.2 及以上

3）将 Mysql 的 mysql-connector-java-5.1.27-bin.jar 拷贝到/opt/module/hive/lib/

```
[atguigu@hadoop102                    module]$                 cp
/opt/software/mysql-libs/mysql-connector-java-5.1.27/mysql-con
nector-java-5.1.27-bin.jar /opt/module/hive/lib/
```

4）在/opt/module/hive/conf 路径上，修改 hive-site.xml 文件，添加红色部分。（注意 mysql 的密码要正确，否则元数据连接不上）

```
[atguigu@hadoop102 conf]$ vim hive-site.xml

#添加如下内容
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
        <name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://hadoop102:3306/metastore?createDatabaseIfNotExist=tru
e</value>
        <description>JDBC connect string for a JDBC metastore</description>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.jdbc.Driver</value>
        <description>Driver class name for a JDBC metastore</description>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>root</value>
                  <description>username    to    use    against    metastore
```

```
database</description>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>123456</value>
                <description>password   to   use   against   metastore
database</description>
    </property>

   <property>
        <name>hive.metastore.warehouse.dir</name>
        <value>/user/hive/warehouse</value>
        <description>location    of    default    database    for    the
warehouse</description>
   </property>

   <property>
       <name>hive.cli.print.header</name>
       <value>true</value>
   </property>

   <property>
       <name>hive.cli.print.current.db</name>
       <value>true</value>
   </property>

   <property>
       <name>hive.metastore.schema.verification</name>
       <value>false</value>
   </property>

   <property>
       <name>datanucleus.schema.autoCreateAll</name>
       <value>true</value>
   </property>

   <property>
       <name>hive.metastore.uris</name>
       <value>thrift://hadoop102:9083</value>
   </property>
</configuration>
```

3）启动服务

```
[atguigu@hadoop102   hive]$   nohup   /opt/module/hive/bin/hive
--service metastore &
[atguigu@hadoop102   hive]$   nohup   /opt/module/hive/bin/hive
--service hiveserver2 &
```

注意：hive2.x 版本需要启动两个服务 metastore 和 hiveserver2，否则会报错 Exception in thread "main" java.lang.RuntimeException: org.apache.hadoop.hive.ql.metadata.HiveException: java.lang.RuntimeException: Unable to instantiate org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClient

4）服务启动完毕后在启动 Hive

```
[atguigu@hadoop102 hive]$ /opt/module/hive/bin/hive
```

2.1.4 安装 Spark2.4.6

注意：Spark 版本至少 2.2.1 及以上

1）把 spark-2.4.6-bin-hadoop2.7.tgz 上传到/opt/software 目录，并解压到/opt/module

```
[atguigu@hadoop102 software]$ tar -zxvf spark-2.4.6-bin-hadoop2.7.tgz -C /opt/module/
```

2）修改名称/opt/module/spark-2.4.6-bin-hadoop2.7 名称为 spark

```
[atguigu@hadoop102 module]$ mv spark-2.4.3-bin-hadoop2.7/ spark
```

3）修改/opt/module/spark/conf/spark-defaults.conf.template 名称为 spark-defaults.conf

```
[atguigu@hadoop102 conf]$ mv spark-defaults.conf.template spark-defaults.conf
```

4）在 spark-default.conf 文件中配置 Spark 日志路径

```
[atguigu@hadoop102 conf]$ vim spark-defaults.conf

#添加如下配置
spark.eventLog.enabled          true
spark.eventLog.dir
hdfs://hadoop102:9000/spark_directory
```

5）修改配置文件 slaves 名称

```
[atguigu@hadoop102 conf]$ mv slaves.template slaves
```

6）修改 slave 文件，添加 work 节点：

```
[atguigu@hadoop102 conf]$ vim slaves

hadoop102
hadoop103
hadoop104
```

7）修改/opt/module/spark/conf/spark-env.sh.template 名称为 spark-env.sh

```
[atguigu@hadoop102 conf]$ mv spark-env.sh.template spark-env.sh
```

8）在/opt/module/spark/conf/spark-env.sh 文件中配置 YARN 配置文件路径、配置历史服务器相关参数

```
[atguigu@hadoop102 conf]$ vim spark-env.sh

#添加如下参数
YARN_CONF_DIR=/opt/module/hadoop-2.7.2/etc/hadoop
export SPARK_HISTORY_OPTS="-Dspark.history.ui.port=18080
-Dspark.history.retainedApplications=30
-Dspark.history.fs.logDirectory=hdfs://hadoop102:9000/spark_directory"
SPARK_MASTER_HOST=hadoop102
SPARK_MASTER_PORT=7077
```

9）在 hadoop 集群上提前创建 spark_directory 日志路径

```
[atguigu@hadoop102 spark]$ hadoop fs -mkdir /spark_directory
```

10）把 Hive 中/opt/module/hive/lib/datanucleus-*.jar 包拷贝到 Spark 的/opt/module/spark/jars 路径

```
[atguigu@hadoop102 lib]$ cp /opt/module/hive/lib/datanucleus-*.jar /opt/module/spark/jars/
```

9）把 Hive 中/opt/module/hive/conf/hive-site.xml 包拷贝到 Spark 的/opt/module/spark/conf

路径

```
[atguigu@hadoop102 conf]$ cp /opt/module/hive/conf/hive-site.xml
/opt/module/spark/conf/
```

10）修改 hadoop 配置文件 yarn-site.xml,添加如下内容：

```
[atguigu@hadoop102 hadoop]$ vi yarn-site.xml
        <!--是否启动一个线程检查每个任务正使用的物理内存量，如果任务超出分配值，
则直接将其杀掉，默认是 true -->
        <property>
                <name>yarn.nodemanager.pmem-check-enabled</name>
                <value>false</value>
        </property>
        <!--是否启动一个线程检查每个任务正使用的虚拟内存量，如果任务超出分配值，
则直接将其杀掉，默认是 true -->
        <property>
                <name>yarn.nodemanager.vmem-check-enabled</name>
                <value>false</value>
        </property>
```

11）分发 spark & yarn-site.xml

```
[atguigu@hadoop102                      conf]$                      xsync
/opt/module/hadoop-2.7.2/etc/hadoop/yarn-site.xml
[atguigu@hadoop102 conf]$ xsync /opt/module/spark
```

10）测试环境

```
[atguigu@hadoop102 spark]$ bin/spark-shell

scala>spark.sql("show databases").show
```

2.1.5 安装 Livy0.3

1）上传 livy-server-0.3.0.zip 到 hadoop102 的/opt/software 目录下，并解压到/opt/module

```
[atguigu@hadoop102  software]$  unzip  livy-server-0.3.0.zip  -d
/opt/module/
```

2）修改/opt/module/livy-server-0.3.0 文件名称为 livy

```
[atguigu@hadoop102 module]$ mv livy-server-0.3.0/ livy
```

3）修改/opt/module/livy/conf/livy-env.sh 文件，添加 livy 环境相关参数

```
export HADOOP_CONF_DIR=/opt/module/hadoop-2.7.2/etc/hadoop/
export SPARK_HOME=/opt/module/spark/
```

3）修改/opt/module/livy/conf/livy.conf 文件，配置 livy 与 spark 相关参数

```
livy.server.host = hadoop102
livy.spark.master =yarn
livy.spark.deployMode = client
livy.repl.enableHiveContext = true
livy.server.port = 8998
```

4）配置需要的环境变量

```
[atguigu@hadoop102 conf]$ sudo vim /etc/profile

#SPARK_HOME
export SPARK_HOME=/opt/module/spark
export PATH=$PATH:$SPARK_HOME/bin
```

```
[atguigu@hadoop102 conf]$ source /etc/profile
```

5）在/opt/module/livy/路径上，启动 livy 服务

```
[atguigu@hadoop102 livy]$ bin/livy-server start
```

2.1.6 初始化 MySQL 数据库

1）上传 Init_quartz_mysql_innodb.sql 到 hadoop102 的/opt/software 目录

2）使用 mysql 创建 quartz 库，执行脚本初始化表信息

```
[atguigu@hadoop102 ~]$ mysql -uroot -p123456
mysql> create database quartz;
mysql> use quartz;
mysql> source /opt/software/Init_quartz_mysql_innodb.sql
mysql> show tables;
```

2.2 编译 Griffin（不选择）

2.2.1 安装 Maven

1）Maven 下载：https://maven.apache.org/download.cgi

2）把 apache-maven-3.6.1-bin.tar.gz 上传到 linux 的/opt/software 目录下

3）解压 apache-maven-3.6.1-bin.tar.gz 到/opt/module/目录下面

```
[atguigu@hadoop102 software]$ tar -zxvf
apache-maven-3.6.1-bin.tar.gz -C /opt/module/
```

4）修改 apache-maven-3.6.1 的名称为 maven

```
[atguigu@hadoop102 module]$ mv apache-maven-3.6.1/ maven
```

5）添加环境变量到/etc/profile 中

```
[atguigu@hadoop102 module]$ sudo vim /etc/profile
#MAVEN_HOME
export MAVEN_HOME=/opt/module/maven
export PATH=$PATH:$MAVEN_HOME/bin
```

6）测试安装结果

```
[atguigu@hadoop102 module]$ source /etc/profile
[atguigu@hadoop102 module]$ mvn -v
```

7）修改 setting.xml，指定为阿里云

```
[atguigu@hadoop102 maven]$ cd conf
[atguigu@hadoop102 maven]$ vim settings.xml

<!-- 添加阿里云镜像-->
  <mirror>
      <id>nexus-aliyun</id>
      <mirrorOf>central</mirrorOf>
      <name>Nexus aliyun</name>
  <url>http://maven.aliyun.com/nexus/content/groups/public</ur
  l>
  </mirror>
  <mirror>
      <id>UK</id>
```

```
    <name>UK Central</name>
    <url>http://uk.maven.org/maven2</url>
    <mirrorOf>central</mirrorOf>
</mirror>
<mirror>
    <id>repo1</id>
    <mirrorOf>central</mirrorOf>
    <name>Human Readable Name for this Mirror.</name>
    <url>http://repo1.maven.org/maven2/</url>
</mirror>
<mirror>
    <id>repo2</id>
    <mirrorOf>central</mirrorOf>
    <name>Human Readable Name for this Mirror.</name>
    <url>http://repo2.maven.org/maven2/</url>
</mirror>
```

8）在/home/atguigu 目录下创建.m2 文件夹

```
[atguigu@hadoop102 ~]$ mkdir .m2
```

2.2.2 修改配置文件:

1）上传 griffin-master.zip 到 hadoop102 的/opt/software 目录，并解压 tar.gz 包到/opt/module

```
[atguigu@hadoop102 software]$ unzip griffin-master.zip -d /opt/module/
```

2）修改/opt/module/griffin-master/ui/pom.xml 文件，添加 node 和 npm 源。

```
[atguigu@hadoop102 ui]$ vim pom.xml
<!-- It will install nodejs and npm -->
<execution>
    <id>install node and npm</id>
    <goals>
        <goal>install-node-and-npm</goal>
    </goals>
    <configuration>
        <nodeVersion>${node.version}</nodeVersion>
        <npmVersion>${npm.version}</npmVersion>
        <nodeDownloadRoot>http://nodejs.org/dist/</nodeDownloadRoot>
        <npmDownloadRoot>http://registry.npmjs.org/npm/-/</npmDownloadRoot>
    </configuration>
</execution>
```

2）修改/opt/module/griffin-master/service/pom.xml 文件，注释掉 org.postgresql，添加 mysql 依赖。

```
[atguigu@hadoop102 service]$ vim pom.xml

<!--
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>${postgresql.version}</version>
</dependency>
-->
<dependency>
```

```
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>
注意:版本号删除掉
```

3）修改/opt/module/griffin-master/service/src/main/resources/application.properties 文件

```
[atguigu@hadoop102                    service]$                    vim
/opt/module/griffin-master/service/src/main/resources/applicat
ion.properties

# Apache Griffin 应用名称
spring.application.name=griffin_service
# MySQL 数据库配置信息
spring.datasource.url=jdbc:mysql://hadoop102:3306/quartz?autoR
econnect=true&useSSL=false
spring.datasource.username=root
spring.datasource.password=000000
spring.jpa.generate-ddl=true
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.jpa.show-sql=true
# Hive metastore 配置信息
hive.metastore.uris=thrift://hadoop102:9083
hive.metastore.dbname=default
hive.hmshandler.retry.attempts=15
hive.hmshandler.retry.interval=2000ms
# Hive cache time
cache.evict.hive.fixedRate.in.milliseconds=900000
# Kafka schema registry 按需配置
kafka.schema.registry.url=http://hadoop102:8081
# Update job instance state at regular intervals
jobInstance.fixedDelay.in.milliseconds=60000
# Expired time of job instance which is 7 days that is 604800000
milliseconds.Time unit only supports milliseconds
jobInstance.expired.milliseconds=604800000
# schedule predicate job every 5 minutes and repeat 12 times at
most
#interval time unit s:second m:minute h:hour d:day,only support
these four units
predicate.job.interval=5m
predicate.job.repeat.count=12
# external properties directory location
external.config.location=
# external BATCH or STREAMING env
external.env.location=
# login strategy ("default" or "ldap")
login.strategy=default
# ldap
ldap.url=ldap://hostname:port
ldap.email=@example.com
ldap.searchBase=DC=org,DC=example
ldap.searchPattern=(sAMAccountName={0})
# hdfs default name
fs.defaultFS=
# elasticsearch
elasticsearch.host=hadoop102
elasticsearch.port=9200
elasticsearch.scheme=http
```

```
# elasticsearch.user = user
# elasticsearch.password = password
# livy
livy.uri=http://hadoop102:8998/batches
# yarn url
yarn.uri=http://hadoop103:8088
# griffin event listener
internal.event.listeners=GriffinJobEventHook
```

4）修改/opt/module/griffin-master/service/src/main/resources/sparkProperties.json 文件

```
[atguigu@hadoop102                    service]$                  vim
/opt/module/griffin-master/service/src/main/resources/sparkPro
perties.json

{
  "file": "hdfs://hadoop102:9000/griffin/griffin-measure.jar",
  "className": "org.apache.griffin.measure.Application",
  "name": "griffin",
  "queue": "default",
  "numExecutors": 2,
  "executorCores": 1,
  "driverMemory": "1g",
  "executorMemory": "1g",
  "conf": {
    "spark.yarn.dist.files":
"hdfs://hadoop102:9000/home/spark_conf/hive-site.xml"
  },
  "files": [
  ]
}
```

5）修改/opt/module/griffin-master/service/src/main/resources/env/env_batch.json 文件

```
[atguigu@hadoop102                    service]$                  vim
/opt/module/griffin-master/service/src/main/resources/env/env_
batch.json

{
  "spark": {
    "log.level": "INFO"
  },
  "sinks": [
    {
      "type": "CONSOLE",
      "config": {
        "max.log.lines": 10
      }
    },
    {
      "type": "HDFS",
      "config": {
        "path": "hdfs://hadoop102:9000/griffin/persist",
        "max.persist.lines": 10000,
        "max.lines.per.file": 10000
      }
    },
    {
      "type": "ELASTICSEARCH",
      "config": {
        "method": "post",
        "api": "http://hadoop102:9200/griffin/accuracy",
        "connection.timeout": "1m",
        "retry": 10
      }
    }
  ],
  "griffin.checkpoint": []
}
```

6）修改/opt/module/griffin-master/service/src/main/resources/env/env_streaming.json 文件

```
[atguigu@hadoop102 service]$ vim /opt/module/griffin-master/service/src/main/resources/env/env_streaming.json

{
  "spark": {
    "log.level": "WARN",
    "checkpoint.dir": "hdfs:///griffin/checkpoint/${JOB_NAME}",
    "init.clear": true,
    "batch.interval": "1m",
    "process.interval": "5m",
    "config": {
      "spark.default.parallelism": 4,
      "spark.task.maxFailures": 5,
      "spark.streaming.kafkaMaxRatePerPartition": 1000,
      "spark.streaming.concurrentJobs": 4,
      "spark.yarn.maxAppAttempts": 5,
      "spark.yarn.am.attemptFailuresValidityInterval": "1h",
      "spark.yarn.max.executor.failures": 120,
      "spark.yarn.executor.failuresValidityInterval": "1h",
      "spark.hadoop.fs.hdfs.impl.disable.cache": true
    }
  },
  "sinks": [
    {
      "type": "CONSOLE",
      "config": {
        "max.log.lines": 100
      }
    },
    {
      "type": "HDFS",
      "config": {
        "path": "hdfs://hadoop102:9000/griffin/persist",
        "max.persist.lines": 10000,
        "max.lines.per.file": 10000
      }
    },
    {
      "type": "ELASTICSEARCH",
      "config": {
        "method": "post",
        "api": "http://hadoop102:9200/griffin/accuracy"
      }
    }
  ],
  "griffin.checkpoint": [
    {
      "type": "zk",
      "config": {
        "hosts": "zk:2181",
        "namespace": "griffin/infocache",
        "lock.path": "lock",
        "mode": "persist",
        "init.clear": true,
        "close.clear": false
      }
    }
  ]
}
```

7）修改/opt/module/griffin-master/service/src/main/resources/quartz.properties 文件

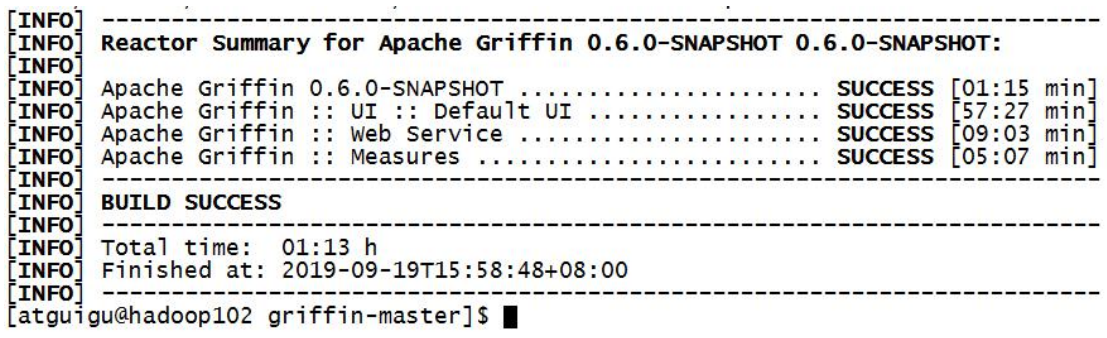
```
[atguigu@hadoop102                    service]$                vim
/opt/module/griffin-master/service/src/main/resources/quartz.p
roperties

org.quartz.scheduler.instanceName=spring-boot-quartz
org.quartz.scheduler.instanceId=AUTO
org.quartz.threadPool.threadCount=5
org.quartz.jobStore.class=org.quartz.impl.jdbcjobstore.JobStor
eTX
# If you use postgresql as your database,set this property value
to org.quartz.impl.jdbcjobstore.PostgreSQLDelegate
# If you use mysql as your database,set this property value to
org.quartz.impl.jdbcjobstore.StdJDBCDelegate
# If you use h2 as your database, it's ok to set this property value
to StdJDBCDelegate, PostgreSQLDelegate or others
org.quartz.jobStore.driverDelegateClass=org.quartz.impl.jdbcjo
bstore.StdJDBCDelegate
org.quartz.jobStore.useProperties=true
org.quartz.jobStore.misfireThreshold=60000
org.quartz.jobStore.tablePrefix=QRTZ_
org.quartz.jobStore.isClustered=true
org.quartz.jobStore.clusterCheckinInterval=20000
```

2.2.3 执行编译

1）在/opt/module/griffin-master 路径执行 maven 命令，开始编译 Griffin 源码

```
[atguigu@hadoop102 griffin-master]$ mvn -Dmaven.test.skip=true
clean install
```

2）见到如下页面，表示编译成功。（大约需要 1 个小时左右）



```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for Apache Griffin 0.6.0-SNAPSHOT 0.6.0-SNAPSHOT:
[INFO]
[INFO] Apache Griffin 0.6.0-SNAPSHOT ...................... SUCCESS [01:15 min]
[INFO] Apache Griffin :: UI :: Default UI ................. SUCCESS [57:27 min]
[INFO] Apache Griffin :: Web Service ...................... SUCCESS [09:03 min]
[INFO] Apache Griffin :: Measures ......................... SUCCESS [05:07 min]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:13 h
[INFO] Finished at: 2019-09-19T15:58:48+08:00
[INFO] ------------------------------------------------------------------------
[atguigu@hadoop102 griffin-master]$
```

2.3 直接使用编译好的 Griffin 包（选择）

2.3.1 修改 jar 配置文件

Griffin 编译完成后，会在 Service 和 Measure 模块的 target 目录下分别看到 service-0.6.0.jar 和 measure-0.6.0.jar 两个 jar 包。因为我们使用的是直接编译好的 jar 包，所以需要将 service-0.6.0.jar 中的配置文件修改成与环境一致。

1）使用 WinRaR 等解压工具打开 service-0.6.0.jar（注意：是打开不是解压）

2）修改 BOOT-INF/classes/application.properties

```
# Apache Griffin 应用名称
spring.application.name=griffin_service
# MySQL 数据库配置信息
spring.datasource.url=jdbc:mysql://hadoop102:3306/quartz?autoR
econnect=true&useSSL=false
spring.datasource.username=root
spring.datasource.password=123456
spring.jpa.generate-ddl=true
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.jpa.show-sql=true
# Hive metastore 配置信息
hive.metastore.uris=thrift://hadoop102:9083
hive.metastore.dbname=default
hive.hmshandler.retry.attempts=15
hive.hmshandler.retry.interval=2000ms
# Hive cache time
cache.evict.hive.fixedRate.in.milliseconds=900000
# Kafka schema registry 按需配置
kafka.schema.registry.url=http://hadoop102:8081
# Update job instance state at regular intervals
jobInstance.fixedDelay.in.milliseconds=60000
# Expired time of job instance which is 7 days that is 604800000
milliseconds.Time unit only supports milliseconds
jobInstance.expired.milliseconds=604800000
# schedule predicate job every 5 minutes and repeat 12 times at
most
#interval time unit s:second m:minute h:hour d:day,only support
these four units
predicate.job.interval=5m
predicate.job.repeat.count=12
# external properties directory location
external.config.location=
# external BATCH or STREAMING env
external.env.location=
# login strategy ("default" or "ldap")
login.strategy=default
# ldap
ldap.url=ldap://hostname:port
ldap.email=@example.com
ldap.searchBase=DC=org,DC=example
ldap.searchPattern=(sAMAccountName={0})
# hdfs default name
```

```
fs.defaultFS=
# elasticsearch
elasticsearch.host=hadoop102
elasticsearch.port=9200
elasticsearch.scheme=http
# elasticsearch.user = user
# elasticsearch.password = password
# livy
livy.uri=http://hadoop102:8998/batches
# yarn url
yarn.uri=http://hadoop103:8088
# griffin event listener
internal.event.listeners=GriffinJobEventHook
```

2）修改 BOOT-INF/classes/sparkProperties.json

```
{
  "file": "hdfs://hadoop102:9000/griffin/griffin-measure.jar",
  "className": "org.apache.griffin.measure.Application",
  "name": "griffin",
  "queue": "default",
  "numExecutors": 2,
  "executorCores": 1,
  "driverMemory": "1g",
  "executorMemory": "1g",
  "conf": {
    "spark.yarn.dist.files":
"hdfs://hadoop102:9000/home/spark_conf/hive-site.xml"
  },
  "files": [
  ]
}
```

3）修改 BOOT-INF/classes/hive-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
        <name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://hadoop102:3306/metastore?createDatabaseIfN
otExist=true</value>
               <description>JDBC   connect   string   for   a   JDBC
metastore</description>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.jdbc.Driver</value>
                <description>Driver   class   name   for   a   JDBC
metastore</description>
    </property>

    <property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>root</value>
              <description>username   to   use   against   metastore
database</description>
    </property>
    <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>123456</value>
             <description>password  to  use  against  metastore
database</description>
    </property>

    <property>
         <name>hive.metastore.warehouse.dir</name>
         <value>/user/hive/warehouse</value>
         <description>location  of  default  database  for  the
warehouse</description>
    </property>

    <property>
        <name>hive.cli.print.header</name>
        <value>true</value>
    </property>

    <property>
        <name>hive.cli.print.current.db</name>
        <value>true</value>
    </property>

    <property>
        <name>hive.metastore.schema.verification</name>
        <value>false</value>
    </property>

    <property>
        <name>datanucleus.schema.autoCreateAll</name>
        <value>true</value>
    </property>
<!--
<property>
    <name>hive.execution.engine</name>
    <value>tez</value>
</property>
-->
 <property>
        <name>hive.metastore.uris</name>
        <value>thrift://hadoop102:9083</value>
 </property>

</configuration>
```

4）修改 BOOT-INF/classes/application-mysql.properties

```
#Data Access Properties
spring.datasource.url=jdbc:mysql://192.168.1.102:3306/quartz?a
utoReconnect=true&useSSL=false
spring.datasource.username=root
spring.datasource.password=123456
spring.jpa.generate-ddl=true
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

5）修改 BOOT-INF/classes/env/env_batch.json

```
{
  "spark": {
    "log.level": "INFO"
  },
  "sinks": [
    {
      "type": "CONSOLE",
      "config": {
        "max.log.lines": 10
      }
    },
    {
      "type": "HDFS",
      "config": {
        "path": "hdfs://hadoop102:9000/griffin/persist",
        "max.persist.lines": 10000,
        "max.lines.per.file": 10000
      }
    },
    {
      "type": "ELASTICSEARCH",
      "config": {
        "method": "post",
        "api": "http://hadoop102:9200/griffin/accuracy",
        "connection.timeout": "1m",
        "retry": 10
      }
    }
  ],
  "griffin.checkpoint": []
}
```

6）修改 BOOT-INF/classes/env/env_streaming.json

```
{
  "spark": {
    "log.level": "WARN",
    "checkpoint.dir": "hdfs:///griffin/checkpoint/${JOB_NAME}",
    "init.clear": true,
    "batch.interval": "1m",
    "process.interval": "5m",
    "config": {
      "spark.default.parallelism": 4,
      "spark.task.maxFailures": 5,
      "spark.streaming.kafkaMaxRatePerPartition": 1000,
      "spark.streaming.concurrentJobs": 4,
      "spark.yarn.maxAppAttempts": 5,
      "spark.yarn.am.attemptFailuresValidityInterval": "1h",
      "spark.yarn.max.executor.failures": 120,
      "spark.yarn.executor.failuresValidityInterval": "1h",
      "spark.hadoop.fs.hdfs.impl.disable.cache": true
    }
  },
  "sinks": [
    {
      "type": "CONSOLE",
      "config": {
        "max.log.lines": 100
      }
    },
    {
      "type": "HDFS",
      "config": {
        "path": "hdfs://hadoop102:9000/griffin/persist",
        "max.persist.lines": 10000,
        "max.lines.per.file": 10000
      }
    },
    {
      "type": "ELASTICSEARCH",
      "config": {
        "method": "post",
        "api": "http://hadoop102:9200/griffin/accuracy"
      }
    }
  ],
  "griffin.checkpoint": [
    {
      "type": "zk",
      "config": {
        "hosts": "zk:2181",
        "namespace": "griffin/infocache",
        "lock.path": "lock",
        "mode": "persist",
        "init.clear": true,
        "close.clear": false
      }
    }
  ]
}
```

2.4 上传执行 Griffin

2.4.1 修改名称并上传 HDFS

命令执行完成后，会在 Service 和 Measure 模块的 target 目录下分别看到 service-0.6.0.jar 和 measure-0.6.0.jar 两个 jar 包。

1）修改/opt/module/griffin-master/measure/target/measure-0.6.0-SNAPSHOT.jar 名称

```
[atguigu@hadoop102  measure]$  mv  measure-0.6.0-SNAPSHOT.jar
griffin-measure.jar
```

2）上传 griffin-measure.jar 到 HDFS 文件目录里

```
[atguigu@hadoop102 measure]$ hadoop fs -mkdir /griffin/
[atguigu@hadoop102 measure]$ hadoop fs -put griffin-measure.jar
/griffin/
```

注意：这样做的目的主要是因为 Spark 在 YARN 集群上执行任务时，需要到 HDFS 的/griffin 目录下加载 griffin-measure.jar，避免发生类 org.apache.griffin.measure.Application 找不到的错误。

3）上传 hive-site.xml 文件到 HDFS 的/home/spark_conf/路径

```
[atguigu@hadoop102 ~]$ hadoop fs -mkdir -p /home/spark_conf/
[atguigu@hadoop102 ~]$ hadoop fs -put /opt/module/hive/conf/hive-site.xml /home/spark_conf/
```

2.4.2 执行 Griffin

1）确保其他服务已经启动

① 启动 HDFS & YARN ：

```
[atguigu@hadoop102 module]$ /opt/module/hadoop-2.7.2/sbin/start-dfs.sh
[atguigu@hadoop103 module]$ /opt/module/hadoop-2.7.2/sbin/start-yarn.sh
```

② 启动 elasticsearch 服务：

```
[atguigu@hadoop102 module]$ nohup /opt/module/elasticsearch-5.2.2/bin/elasticsearch &
```

③ 启动 hive 服务：

```
[atguigu@hadoop102 hive]$ nohup /opt/module/hive/bin/hive --service metastore &
[atguigu@hadoop102 hive]$ nohup /opt/module/hive/bin/hive --service hiveserver2 &
```

④ 启动 livy 服务：

```
[atguigu@hadoop102 livy]$ /opt/module/livy/bin/livy-server start
```

2）进入到/opt/module/griffin-master/service/target/路径，运行 service-0.6.0-SNAPSHOT.jar

控制台启动：控制台打印信息

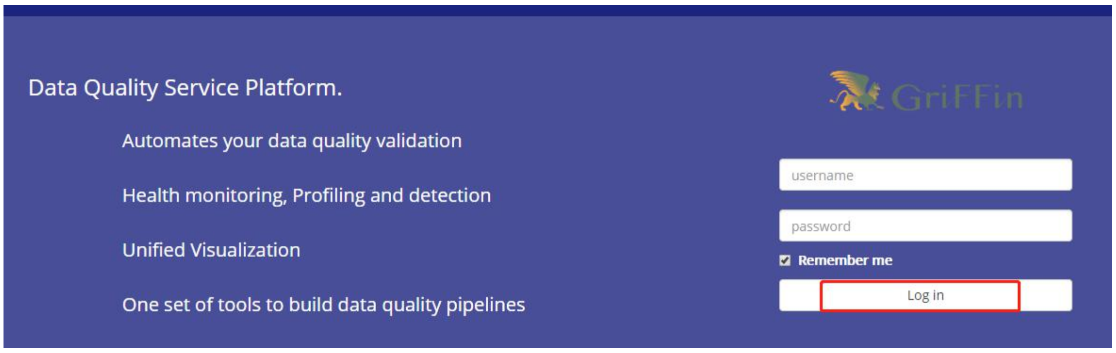
```
[atguigu@hadoop102 target]$ java -jar /opt/module/griffin/service-0.6.0-SNAPSHOT.jar
```

后台启动：启动后台并把日志归写倒 service.out

```
[atguigu@hadoop102 ~]$ nohup java -jar service-0.6.0-SNAPSHOT.jar>service.out 2>&1 &
```

2.4.3 浏览器访问

http://hadoop102:8080 默认账户和密码都是无


Data Quality Service Platform.

Automates your data quality validation

Health monitoring, Profiling and detection

Unified Visualization

One set of tools to build data quality pipelines

第 3 章 案例实操

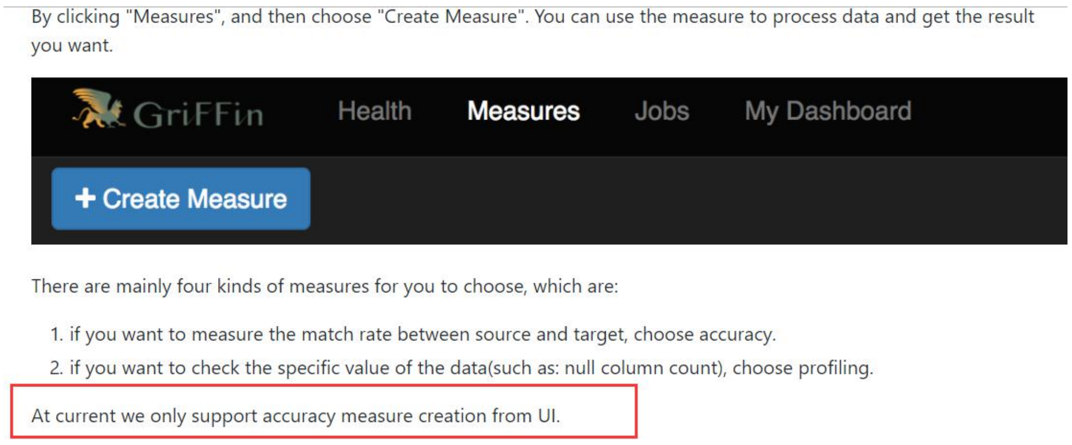
3.1 生产测试数据

获取官网测试数据。在/opt/module/目录下创建 data 文件夹，并下载相关测试数据

```
[atguigu@hadoop102 moudle]$ mkdir data
[atguigu@hadoop102 data]$
wget http://griffin.apache.org/data/batch/gen_demo_data.sh
wget http://griffin.apache.org/data/batch/gen_delta_src.sh
wget http://griffin.apache.org/data/batch/demo_basic
wget http://griffin.apache.org/data/batch/delta_tgt
wget
http://griffin.apache.org/data/batch/insert-data.hql.template
wget http://griffin.apache.org/data/batch/gen-hive-data.sh
wget http://griffin.apache.org/data/batch/create-table.hql
wget http://griffin.apache.org/data/batch/delta_src
wget http://griffin.apache.org/data/batch/delta_tgt

[atguigu@hadoop102 data]$ chmod 777 ../data -R
#生成临时文件
[atguigu@hadoop102 data]$ ./gen_demo_data.sh
#生产测试数据
[atguigu@hadoop102 data]$ ./gen-hive-data.sh
```

3.2 UI 创建 Measure

注意根据官网描述，目前 UI 创建 Measure 只支持 Accuracy 的 Measure，UI 界面上虽然有其他选项但是无法运行 job。

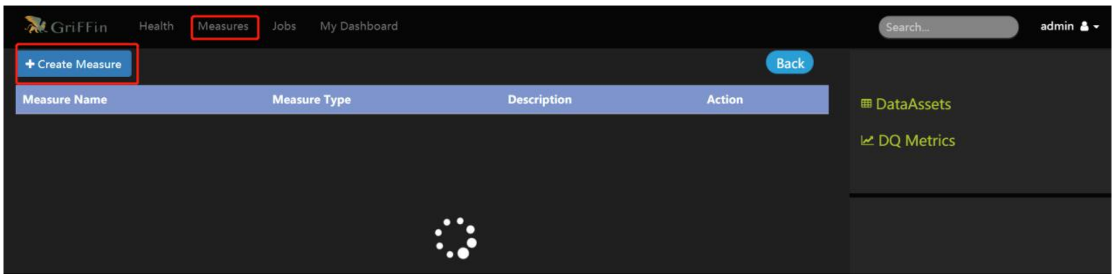

By clicking "Measures", and then choose "Create Measure". You can use the measure to process data and get the result you want.

There are mainly four kinds of measures for you to choose, which are:

1. if you want to measure the match rate between source and target, choose accuracy.
2. if you want to check the specific value of the data(such as: null column count), choose profiling.

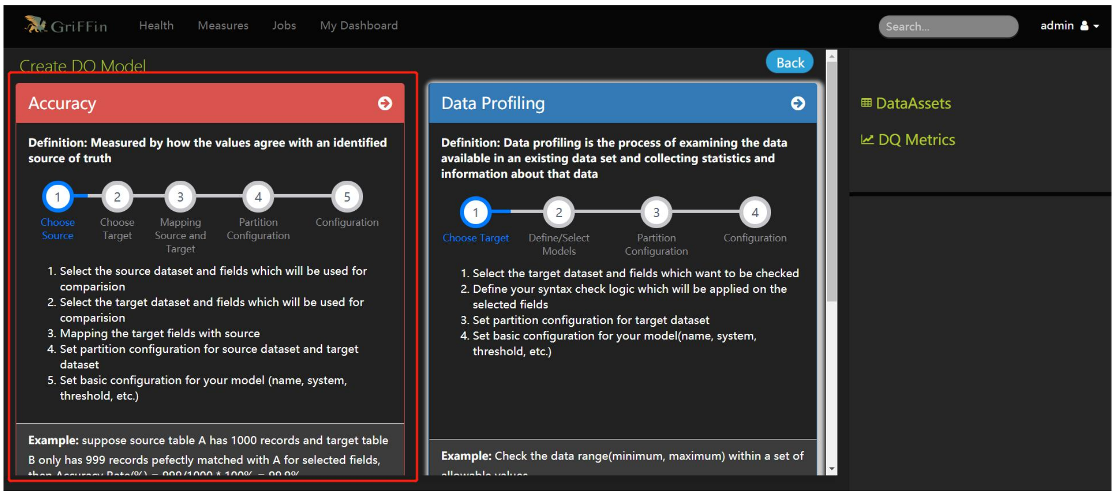
At current we only support accuracy measure creation from UI.

3.2.1 添加一个新的 Measure



Measure Name	Measure Type	Description	Action

3.2.2 选择准确度 Accuracy



Accuracy

Definition: Measured by how the values agree with an identified source of truth

1 Choose Source — 2 Choose Target — 3 Mapping Source and Target — 4 Partition Configuration — 5 Configuration

1. Select the source dataset and fields which will be used for comparision
2. Select the target dataset and fields which will be used for comparision
3. Mapping the target fields with source
4. Set partition configuration for source dataset and target dataset
5. Set basic configuration for your model (name, system, threshold, etc.)

Example: suppose source table A has 1000 records and target table B only has 999 records pefectly matched with A for selected fields,

Data Profiling

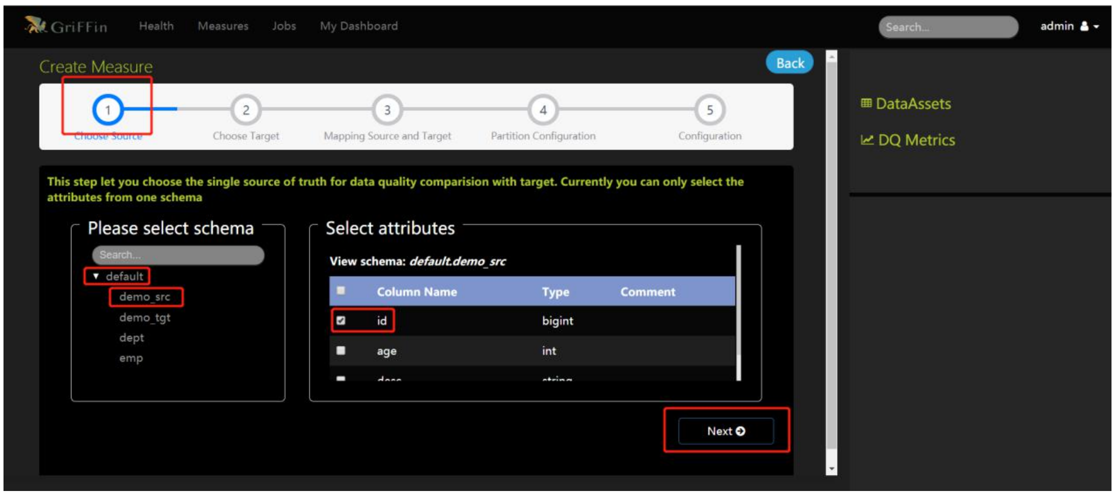
Definition: Data profiling is the process of examining the data available in an existing data set and collecting statistics and information about that data

1 Choose Target — 2 Define/Select Models — 3 Partition Configuration — 4 Configuration

1. Select the target dataset and fields which want to be checked
2. Define your syntax check logic which will be applied on the selected fields
3. Set partition configuration for target dataset
4. Set basic configuration for your model(name, system, threshold, etc.)

Example: Check the data range(minimum, maximum) within a set of

3.2.3 选择数据源的字段



This step let you choose the single source of truth for data quality comparision with target. Currently you can only select the attributes from one schema

Please select schema

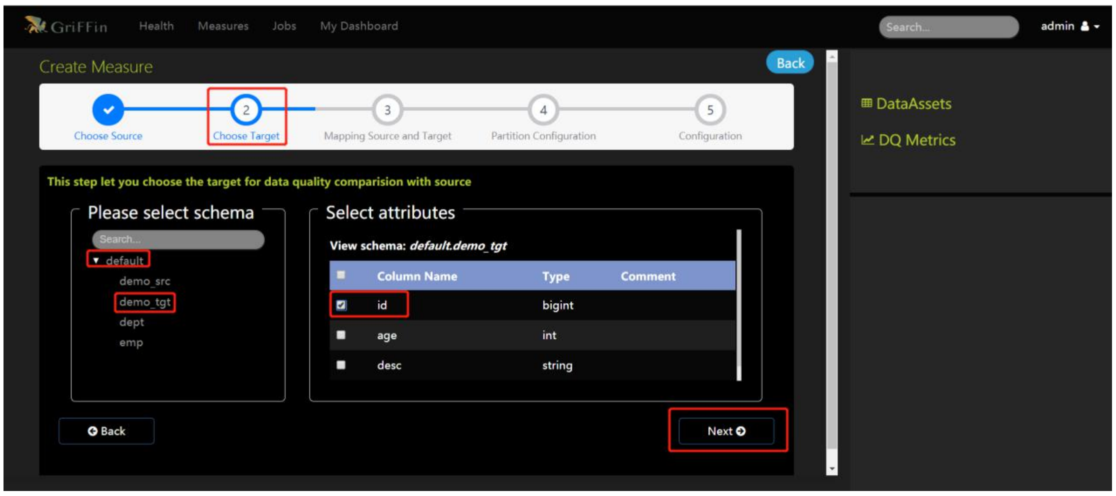
- default
  - demo_src
  - demo_tgt
  - dept
  - emp

Select attributes

View schema: *default.demo_src*

	Column Name	Type	Comment
☑	id	bigint	
☐	age	int	

3.2.4 选择目标表的字段



This step let you choose the target for data quality comparision with source

Please select schema

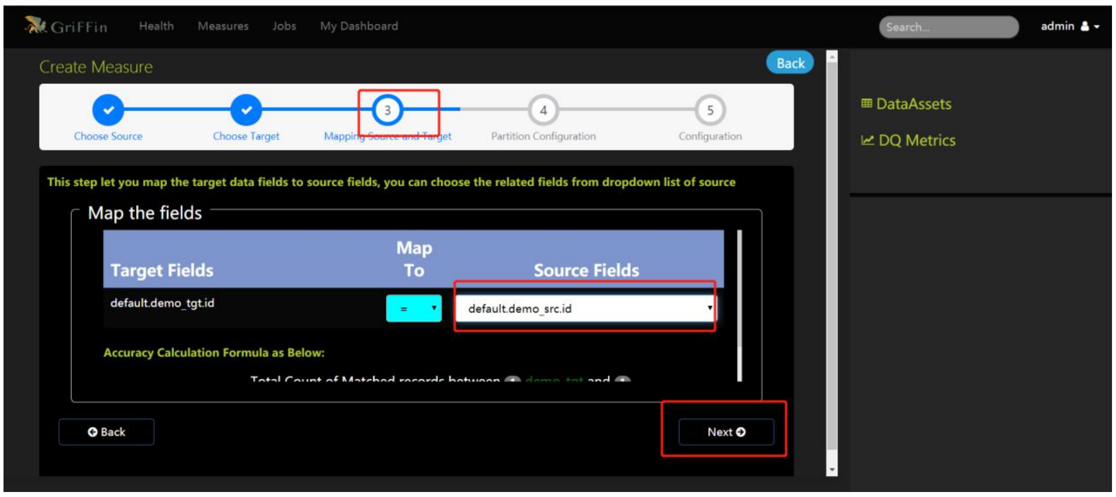
- default
  - demo_src
  - demo_tgt
  - dept
  - emp

Select attributes

View schema: *default.demo_tgt*

	Column Name	Type	Comment
✓	id	bigint	
	age	int	
	desc	string	

3.2.5 选择条件



This step let you map the target data fields to source fields, you can choose the related fields from dropdown list of source

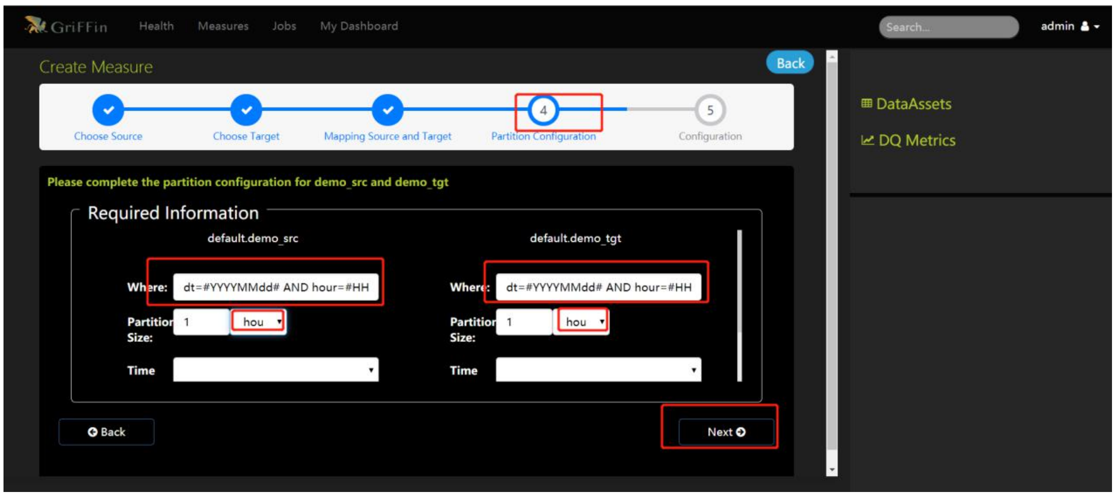
Map the fields

Target Fields	Map To	Source Fields
default.demo_tgt.id	=	default.demo_src.id

Accuracy Calculation Formula as Below:

Total Count of Matched records between demo_tgt and

3.2.6 选择时间格式和分区尺度

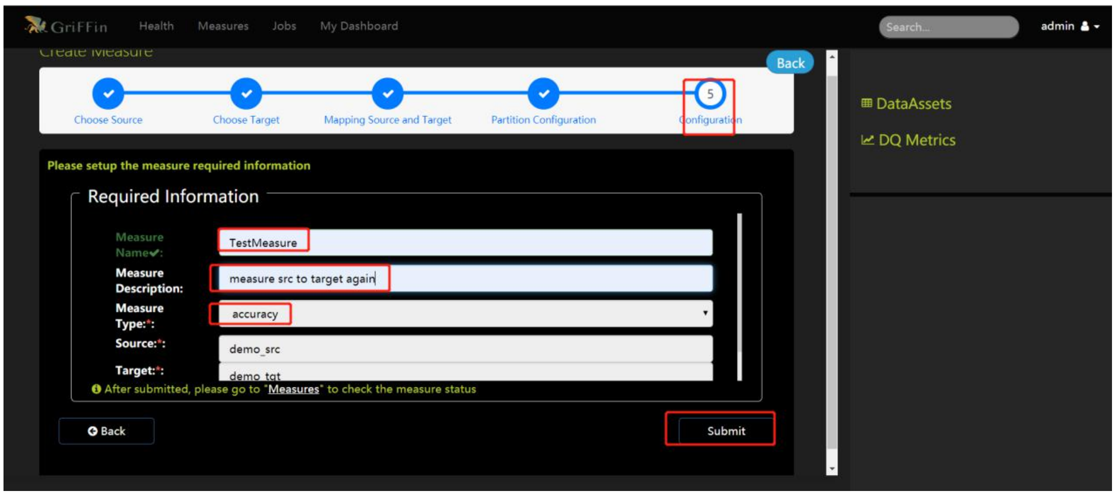


Please complete the partition configuration for demo_src and demo_tgt

Required Information

	default.demo_src	default.demo_tgt
Where:	dt=#YYYYMMdd# AND hour=#HH	dt=#YYYYMMdd# AND hour=#HH
Partition Size:	1 hou	1 hou
Time		

3.2.7 添加 Measure 名称和描述



Please setup the measure required information

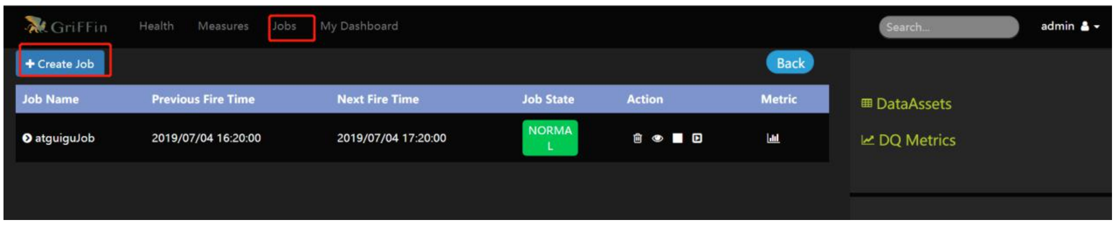
Required Information

Measure Name✔: TestMeasure
Measure Description: measure src to target again
Measure Type*: accuracy
Source*: demo_src
Target*: demo_tgt

After submitted, please go to "Measures" to check the measure status

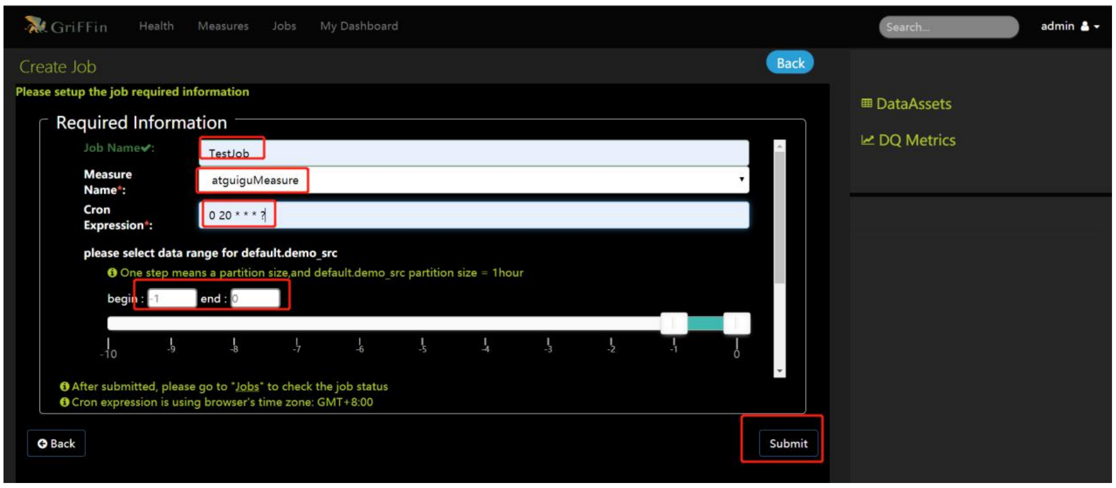
3.3 UI 创建 Job

3.3.1 新建一个 Job



Job Name	Previous Fire Time	Next Fire Time	Job State	Action	Metric
atguiguJob	2019/07/04 16:20:00	2019/07/04 17:20:00	NORMAL		

3.3.2 与 Measure 结合并调度任务执行



Please setup the job required information

Required Information

Job Name✔: TestJob
Measure Name*: atguiguMeasure
Cron Expression*: 0 20 * * * ?

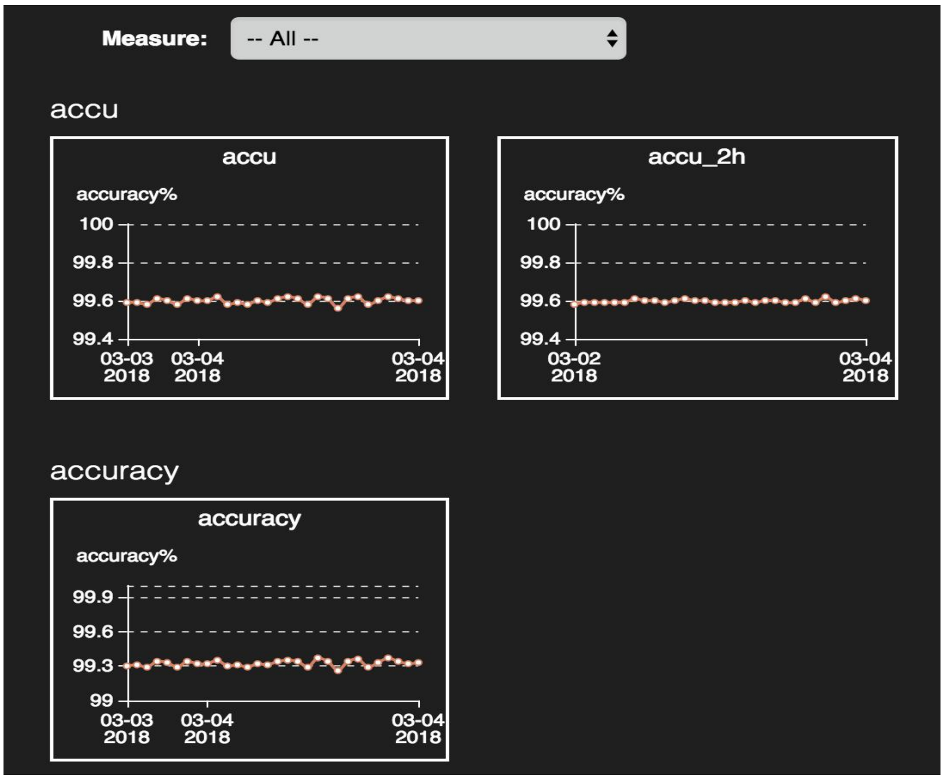
please select data range for default.demo_src

One step means a partition size,and default.demo_src partition size = 1hour

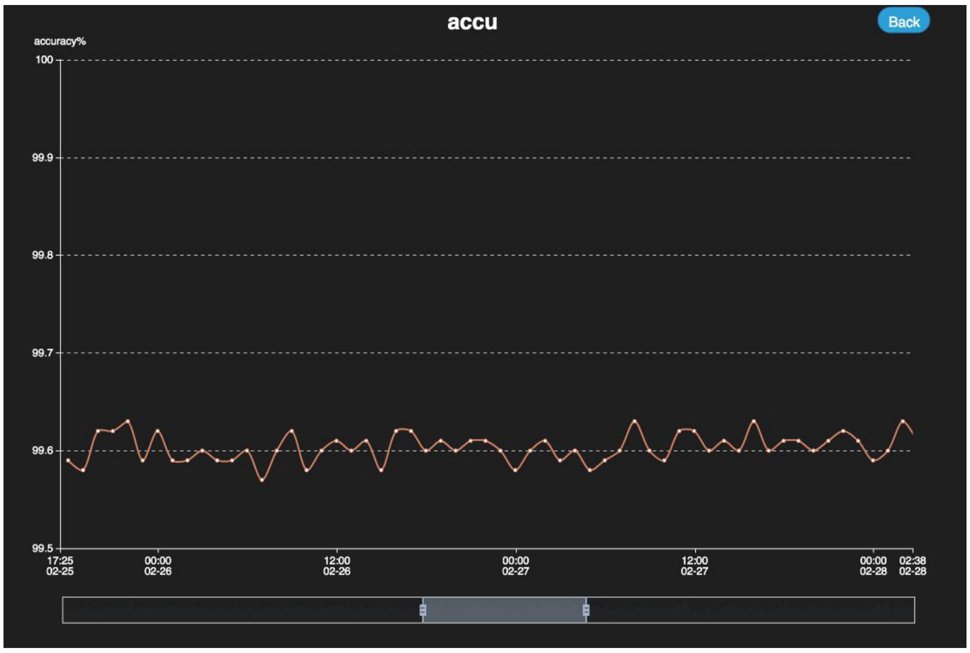
begin : -1 end : 0

After submitted, please go to "Jobs" to check the job status
Cron expression is using browser's time zone: GMT+8:00

3.3.3 查看运行结果单击“DQ 指标”



单击放大图片



第 4 章 数据校验

4.1 ODS 层校验

4.1.1 数据校验通用脚本

通过 shell 脚本调用 hive，检验当日分区增加的记录数量是否在合理的范围之内，同时检验关键字段为空的记录的记录数量，根据生成的指标结合 Griffin 进行数据质量监控与管理。

1）创建数据检查脚本文件夹，用于存放数据校验 shell 脚本和 azkaban 的 job 文件

```
[atguigu@hadoop102 module]$ mkdir -p data_check/sh
[atguigu@hadoop102 module]$ mkdir -p data_check/job
[atguigu@ hadoop102 sh]$ pwd
/opt/module/data_check/sh
[atguigu@ hadoop102 sh]$ pwd
/opt/module/data_check/job
```

2）在 Hive 中创建表数据质量校验记录表，记录数据校验的各个指标：

```
[atguigu@atguigu data_check]$ hive
```

创建数据库：

```
hive (default)> create database datacheck;
```

创建数据表：

```
hive (default)> create table datacheck.table_count_add_standard(
    data_date string comment '数据时间分区 dt',
    database_name string comment '库名',
    table_name string comment '表名',
    table_type string comment '表类型（全量/增量）',
    add_count bigint comment '当日增量数据的记录数',
    null_count bigint comment '表空值记录数',
    total_count bigint comment '全表记录数'
);
```

3）在路径/opt/module/data_check/sh 下创建数据检验增量表通用 shell 脚本

```
[atguigu@hadoop102 sh]$ vim increment_data_check_public.sh
```

在脚本中编写如下内容：

```
#!/bin/bash
do_date=$1
table_name=$2
null_column=$3
null_where_sql_str=''
array=(${null_column//,/ })

for(( i=0;i<${#array[@]};i++)) do
   if [ $i -eq 0 ];then
        null_where_sql_str=" where ${array[i]} is null "
   else
        null_where_sql_str="$null_where_sql_str or ${array[i]} is
```

```
null "
   fi
done;

add_count_query_result=`hive -e "select count(*) from
gmall.$table_name where dt='$do_date'"`

add_count=${add_count_query_result:3}

total_count_query_result=`hive -e "select count(*) from
gmall.$table_name"`

total_count=${total_count_query_result:3}

table_null_query_result=`hive -e "select count(*) from
gmall.$table_name $null_where_sql_str"`

null_count=${table_null_query_result:3}

hive -e "insert into datacheck.table_count_add_standard
values('$do_date','gmall','$table_name','increment_table',$add
_count,$null_count,'$total_count')"
```

脚本参数注释：

第一个参数：传入时间分区参数(dt)

第二个参数：需要进行数据校验的表名(table_name)

第三个参数：需要判断是否为空值的字段名称用逗号‘ , ’隔开，例如：col1,col2,col3

脚本执行示例：

```
[atguigu@hadoop102 data_check]$
./increment_data_check_public.sh 2020-03-10 ods_activity_rule
id,activity_id
```

4）在路径/opt/module/data_check/sh 下创建数据检验全量表通用 shell 脚本

```
[atguigu@hadoop102 sh]$ vim total_data_check_public.sh
```

在脚本中编写如下内容：

```
#!/bin/bash

do_date=$1
table_name=$2
null_column=$3
null_where_sql_str=''
array=(${null_column//,/ })

for(( i=0;i<${#array[@]};i++)) do
   if [ $i -eq 0 ];then
       null_where_sql_str=" where ${array[i]} is null "
   else
       null_where_sql_str="$null_where_sql_str or ${array[i]} is
null "
   fi
done;

table_count_query_result=`hive -e "select count(*) from
```

```
gmall.$table_name"`

table_count=${table_count_query_result:3}

table_null_query_result=`hive -e "select count(*) from 
gmall.$table_name $null_where_sql_str"`

null_count=${table_null_query_result:3}

hive -e "insert into datacheck.table_count_add_standard 
values('$do_date','gmall','$table_name','total_table',null,$nu
ll_count,'$table_count')"
```

脚本参数注释：

第一个参数：传入数据校验日期(dt)

第二个参数：需要进行数据校验的表名(table_name)

第三个参数：需要判断是否为空值的字段名称用逗号’,‘隔开，例如：col1,col2,col3

脚本执行示例：

```
[atguigu@hadoop102 data_check]$ ./ total_data_check_public.sh 
2020-03-10 ods_activity_rule id,activity_id
```

4.1.2 ODS 层各表检验

1. 涉及表

增量检查

（1）订单详情表（ods_order_detail）

（2）用户表（ods_user_info）

（3）支付流水表（ods_payment_info）

（4）订单状态表（ods_order_status_log）

（5）商品评论表（ods_comment_info）

（6）退单表（ods_order_refund_info）

（7）活动订单关联表（ods_activity_order）

全量检查

（1）订单表（ods_order_info）

（2）SKU 商品表（ods_sku_info）

（3）商品一级分类表（ods_base_category1）

（4）商品二级分类表（ods_base_category2）

（5）商品三级分类表（ods_base_category3）

（6）品牌表（ods_base_trademark）

（7）SPU 商品表（ods_spu_info）

（8）加购表（ods_cart_info）

（9）商品收藏表（ods_favor_info）

（10）优惠券领用表（ods_coupon_use）

（11）优惠券表（ods_coupon_info）

（12）活动表（ods_activity_info）

（13）优惠规则表（ods_activity_rule）

（14）编码字典表（ods_base_dic）

2. ODS 层数据检查脚本

1）在路径/opt/module/data_check/sh 下创建 ODS 层数据检查脚本

```
[atguigu@atguigu sh]$ pwd
/opt/module/data_check/sh
[atguigu@atguigu sh]$ vim ods_data_check.sh
```

在脚本中编写如下内容：

```
#!/bin/bash
data_date=$1

# 增量检查
# 订单详情表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_order_detail
id,order_id,user_id,sku_id,sku_name,order_price,sku_num,create
_time
# 用户表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_user_info
id,name,birthday,gender,email,user_level,create_time,operate_t
ime
# 支付流水表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_payment_info
id,out_trade_no,order_id,user_id,alipay_trade_no,total_amount,
subject,payment_type,payment_time
# 订单状态表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_order_status_log
id,order_id,order_status,operate_time
# 商品评论表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_comment_info
id,user_id,sku_id,spu_id,order_id,appraise,create_time
# 退单表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_order_refund_info
id,user_id,order_id,sku_id,refund_type,refund_num,refund_amoun
t,refund_reason_type,create_time
# 活动订单关联表
/opt/module/data_check/sh/increment_data_check_public.sh
$data_date ods_activity_order
id,activity_id,order_id,create_time
# 全量检查
# 订单表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_order_info
id,final_total_amount,order_status,user_id,out_trade_no,create
_time,operate_time,province_id,benefit_reduce_amount,original_
total_amount,feight_fee
# SKU 商品表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_sku_info
id,spu_id,price,sku_name,sku_desc,weight,tm_id,category3_id,cr
```

```
eate_time
# 商品一级分类表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_base_category1 id,name
# 商品二级分类表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_base_category2 id,name,category1_id
# 商品三级分类表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_base_category3 id,name,category2_id
# 品牌表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_base_trademark tm_id,tm_name
# SPU 商品表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_spu_info id,spu_name,category3_id,tm_id
# 加购表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_cart_info
id,user_id,sku_id,cart_price,sku_num,sku_name,create_time,oper
ate_time,is_ordered,order_time
# 商品收藏表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_favor_info
id,user_id,sku_id,spu_id,is_cancel,create_time,cancel_time
# 优惠券领用表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_coupon_use
id,coupon_id,user_id,order_id,coupon_status,get_time,using_tim
e,used_time
# 优惠券表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_coupon_info
id,coupon_name,coupon_type,condition_amount,condition_num,acti
vity_id,benefit_amount,benefit_discount,create_time,range_type,
spu_id,tm_id,category3_id,limit_num,operate_time,expire_time
# 活动表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_activity_info
id,activity_name,activity_type,start_time,end_time,create_time
# 优惠规则表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_activity_rule
id,activity_id,condition_amount,condition_num,benefit_amount,b
enefit_discount,benefit_level
# 编码字典表
/opt/module/data_check/sh/total_data_check_public.sh $data_date
ods_base_dic
dic_code,dic_name,parent_code,create_time,operate_time
```

4.2 DWD 层校验

4.2.1 使用 Griffin 进行数据质量监控管理

1. 浏览器访问 Griffin 的 Web 页面

```
http://hadoop102:8080/
```

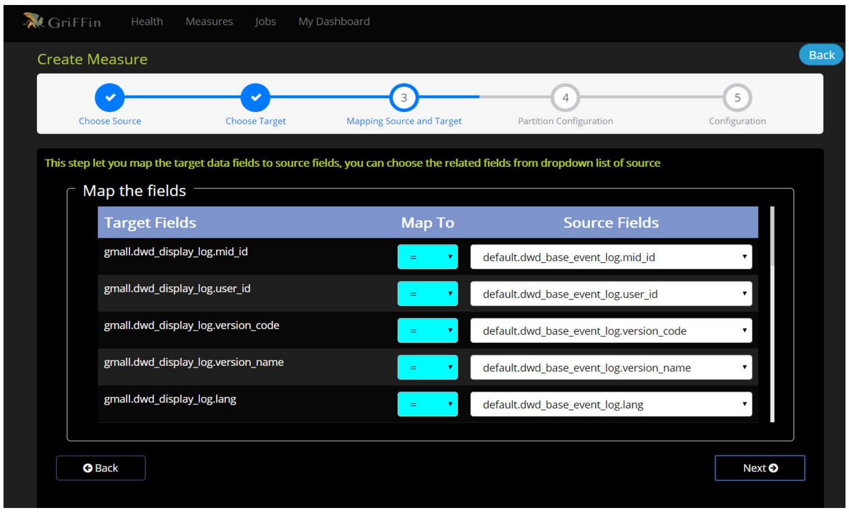
2. 创建 DWD 层的数据校验规则 Measure 和定时任务 Job

1）用户行为事件表

① 商品点击表

```
数据源表：dwd_base_event_log
源表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
数据目标表：dwd_display_log
目标表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
分区：dt='2020-03-10'
```

1）设置 Measure 的数据对比规则



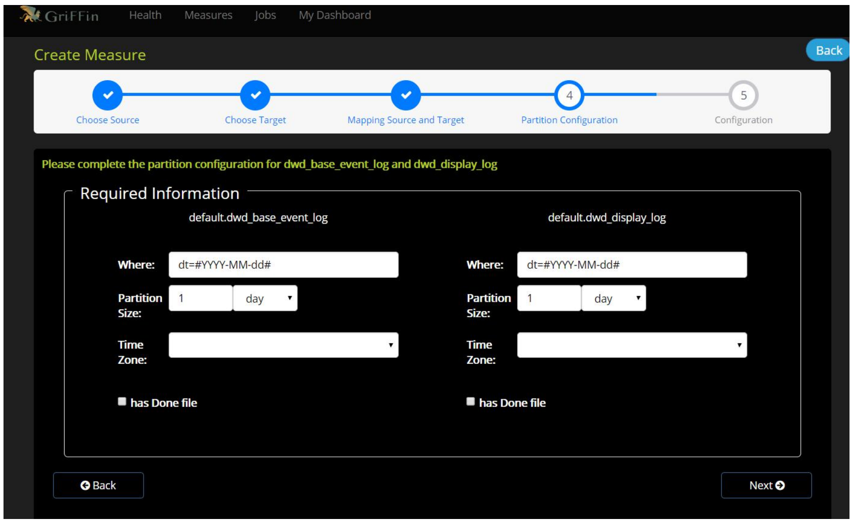
Create Measure

This step let you map the target data fields to source fields, you can choose the related fields from dropdown list of source

Map the fields

Target Fields	Map To	Source Fields
gmall.dwd_display_log.mid_id	=	default.dwd_base_event_log.mid_id
gmall.dwd_display_log.user_id	=	default.dwd_base_event_log.user_id
gmall.dwd_display_log.version_code	=	default.dwd_base_event_log.version_code
gmall.dwd_display_log.version_name	=	default.dwd_base_event_log.version_name
gmall.dwd_display_log.lang	=	default.dwd_base_event_log.lang

2）设置分区



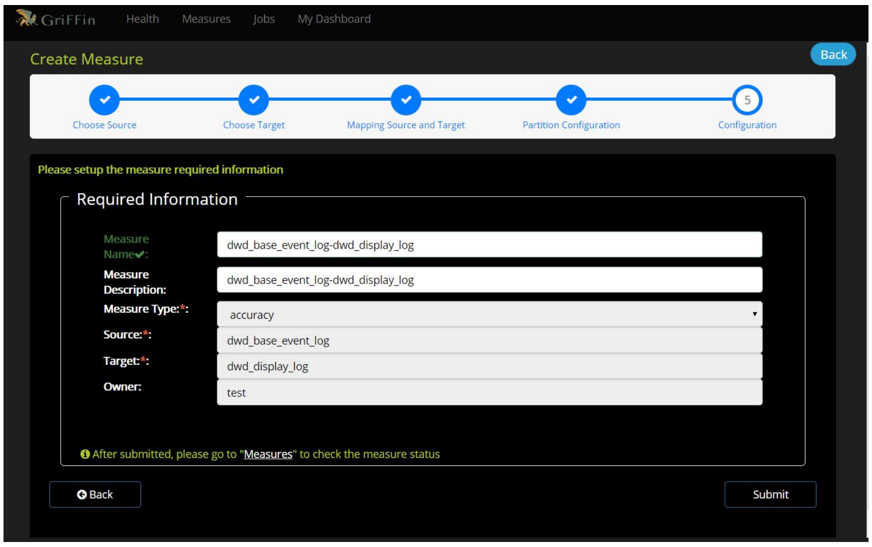
Create Measure

Please complete the partition configuration for dwd_base_event_log and dwd_display_log

Required Information

	default.dwd_base_event_log	default.dwd_display_log
Where:	dt=#YYYY-MM-dd#	dt=#YYYY-MM-dd#
Partition Size:	1 day	1 day
Time Zone:		
has Done file	☐	☐

3）设置 Measure 名称和描述



Create Measure

Please setup the measure required information

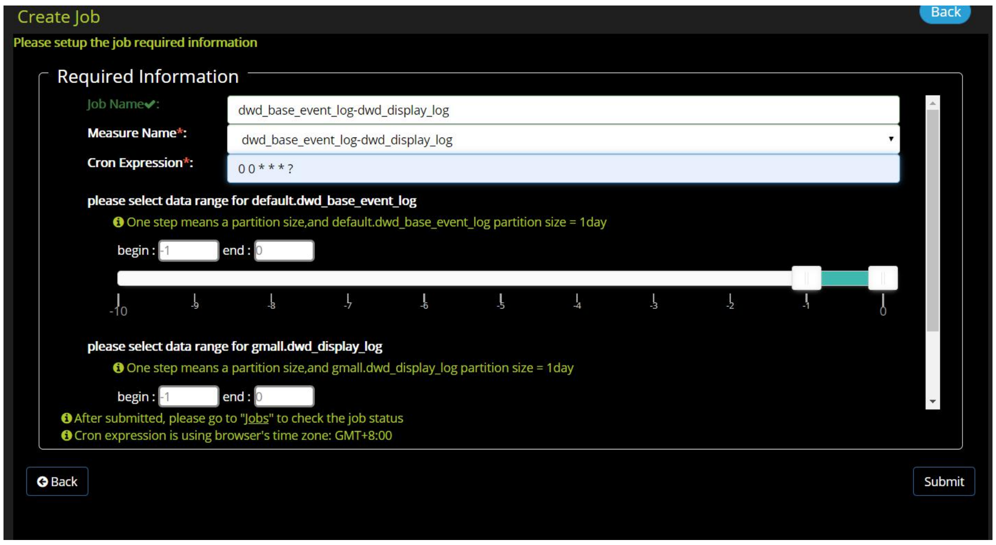
Required Information

Measure Name✔:	dwd_base_event_log-dwd_display_log
Measure Description:	dwd_base_event_log-dwd_display_log
Measure Type:*:	accuracy
Source:*:	dwd_base_event_log
Target:*:	dwd_display_log
Owner:	test

After submitted, please go to "Measures" to check the measure status

Back Submit

4）设置定时数据校验任务 job



Create Job

Please setup the job required information

Required Information

Job Name✔:	dwd_base_event_log-dwd_display_log
Measure Name*:	dwd_base_event_log-dwd_display_log
Cron Expression*:	0 0 * * * ?

please select data range for default.dwd_base_event_log

One step means a partition size,and default.dwd_base_event_log partition size = 1day

begin : -1 end : 0

please select data range for gmall.dwd_display_log

One step means a partition size,and gmall.dwd_display_log partition size = 1day

begin : -1 end : 0

After submitted, please go to "Jobs" to check the job status

Cron expression is using browser's time zone: GMT+8:00

ps：以下表均按上述操作。

② 商品详情页表

数据源表：dwd_base_event_log
源表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
数据目标表：dwd_newsdetail_log
目标表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
分区：dt='2020-03-10'

③ 商品列表页表

数据源表：dwd_base_event_log
源表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
数据目标表：dwd_loading_log
目标表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time

```
分区：dt='2020-03-10'
```

④ 广告表

```
数据源表：dwd_base_event_log
源表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
数据目标表：dwd_ad_log
目标表字段:mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
分区：dt='2020-03-10'
```

⑤ 消息通知表

```
数据源表：dwd_base_event_log
源表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
数据目标表：dwd_notification_log
目标表字段:mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
分区：dt='2020-03-10'
```

⑥ 用户后台活跃表

```
数据源表：dwd_base_event_log
源表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
数据目标表：dwd_active_background_log
目标表字段:mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
分区：dt='2020-03-10'
```

⑦ 评论表

```
数据源表：dwd_base_event_log
源表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
数据目标表：dwd_comment_log
目标表字段:mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
分区：dt='2020-03-10'
```

⑧ 收藏表

```
数据源表：dwd_base_event_log
源表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
数据目标表：dwd_favorites_log
目标表字段:mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
分区：dt='2020-03-10'
```

⑨ 点赞表

数据源表：dwd_base_event_log
源表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
数据目标表：dwd_praise_log
目标表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
分区：dt='2020-03-10'

⑩ 错误日志表

数据源表：dwd_base_event_log
源表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
数据目标表：dwd_error_log
目标表字段：mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network,lng,lat,server_time
分区：dt='2020-03-10'

2）业务数据表

① 优惠券信息表

数据源表：ods_coupon_info
源表字段：id,coupon_name,coupon_type,condition_amount,condition_num,activity_id,benefit_amount,benefit_discount,create_time,range_type,spu_id,tm_id,category3_id,limit_num,operate_time,expire_time
数据目标表：dwd_dim_coupon_info
目标表字段：id,coupon_name,coupon_type,condition_amount,condition_num,activity_id,benefit_amount,benefit_discount,create_time,range_type,spu_id,tm_id,category3_id,limit_num,operate_time,expire_time
分区：dt='2020-03-10'

② 订单明细事实表

数据源表：ods_order_detail
源表字段：id,order_id,user_id,sku_id,sku_name,order_price,sku_num,create_time
数据目标表：dwd_fact_order_detail
目标表字段：id,order_id,user_id,sku_id,sku_name,order_price,sku_num,create_time
分区：dt='2020-03-10'

③ 支付事实表

数据源表：ods_payment_info
源表字段：id,out_trade_no,order_id,user_id,alipay_trade_no,total_amount,subject,payment_type,payment_time
数据目标表：dwd_fact_payment_info
目标表字段：id,out_trade_no,order_id,user_id,alipay_trade_no,payment_amount,subject,payment_type,payment_time
分区：dt='2020-03-10'

④ 退款事实表

数据源表：ods_order_refund_info
源表字段：id,user_id,order_id,sku_id,refund_type,refund_num,refund_amount,refund_reason_type,create_time

数据目标表：dwd_fact_order_refund_info
目标表字段：id,user_id,order_id,sku_id,refund_type,refund_num,refund_amount,refund_reason_type,create_time
分区：dt='2020-03-10'

⑤ 评价事实表

数据源表：ods_comment_info
源表字段：id,user_id,sku_id,spu_id,order_id,appraise,create_time
数据目标表：dwd_fact_comment_info
目标表字段：id,user_id,sku_id,spu_id,order_id,appraise,create_time
分区：dt='2020-03-10'

⑥ 加购事实表

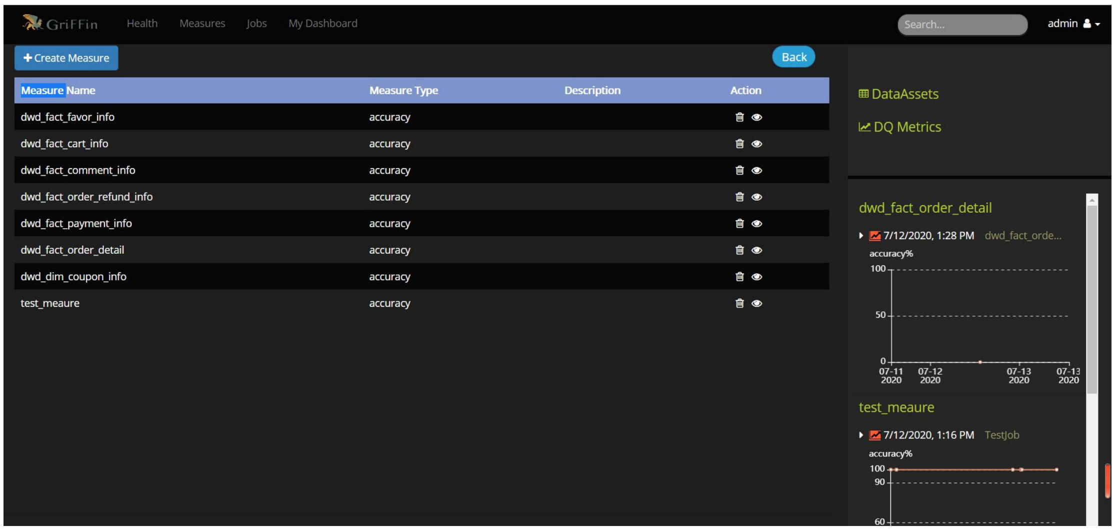
数据源表：ods_cart_info
源表字段：id,user_id,sku_id,cart_price,sku_num,sku_name,create_time,operate_time,is_ordered,order_time
数据目标表：dwd_fact_cart_info
目标表字段：id,user_id,sku_id,cart_price,sku_num,sku_name,create_time,operate_time,is_ordered,order_time
分区：dt='2020-03-10'

⑦ 收藏事实表

数据源表：ods_favor_info
源表字段：id,user_id,sku_id,spu_id,is_cancel,create_time,cancel_time
数据目标表：dwd_fact_favor_info
目标表字段：id,user_id,sku_id,spu_id,is_cancel,create_time,cancel_time
分区：dt='2020-03-10'

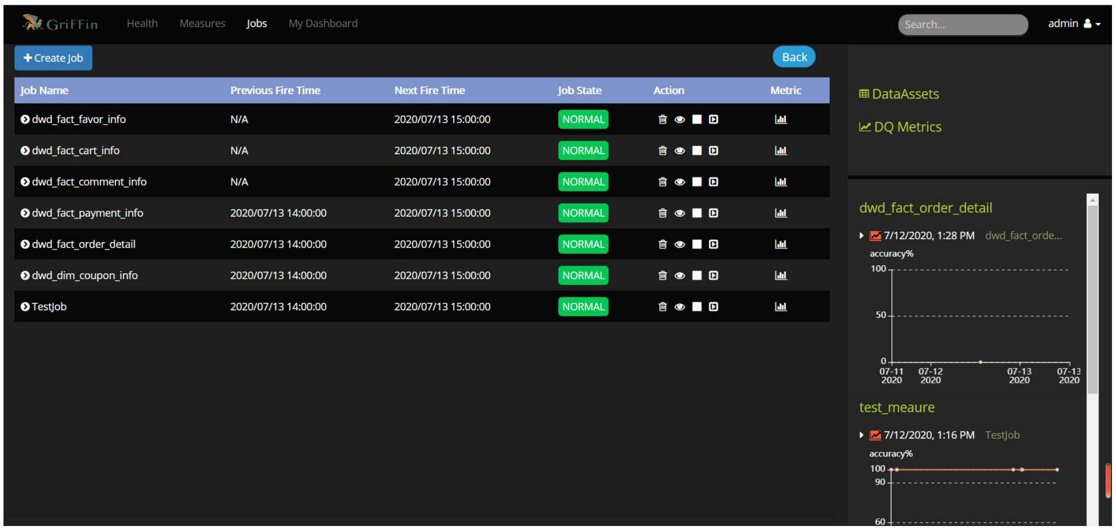
3. 部分任务截图

1）Measure 列表



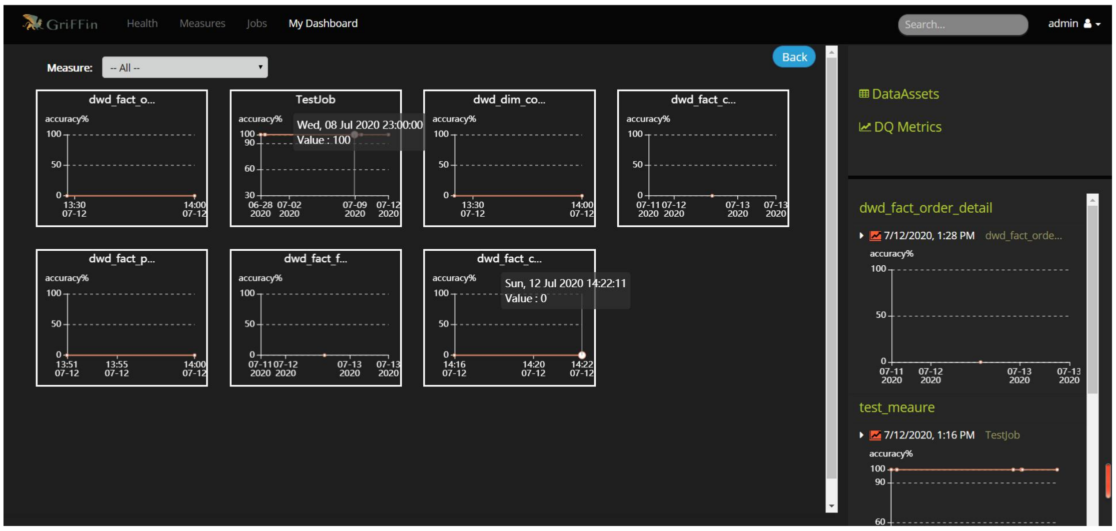
Measure Name	Measure Type	Description	Action
dwd_fact_favor_info	accuracy		
dwd_fact_cart_info	accuracy		
dwd_fact_comment_info	accuracy		
dwd_fact_order_refund_info	accuracy		
dwd_fact_payment_info	accuracy		
dwd_fact_order_detail	accuracy		
dwd_dim_coupon_info	accuracy		
test_meaure	accuracy		

2） Job 列表



Job Name	Previous Fire Time	Next Fire Time	Job State	Action	Metric
dwd_fact_favor_info	N/A	2020/07/13 15:00:00	NORMAL		
dwd_fact_cart_info	N/A	2020/07/13 15:00:00	NORMAL		
dwd_fact_comment_info	N/A	2020/07/13 15:00:00	NORMAL		
dwd_fact_payment_info	2020/07/13 14:00:00	2020/07/13 15:00:00	NORMAL		
dwd_fact_order_detail	2020/07/13 14:00:00	2020/07/13 15:00:00	NORMAL		
dwd_dim_coupon_info	2020/07/13 14:00:00	2020/07/13 15:00:00	NORMAL		
TestJob	2020/07/13 14:00:00	2020/07/13 15:00:00	NORMAL		

3） 监控仪表盘面板


GriFFin
Health
Measures
Jobs
My Dashboard
Search...
admin
Back
Measure:
-- All --
dwd_fact_o...
TestJob
Wed, 08 Jul 2020 23:00:00
Value : 100
dwd_dim_co...
dwd_fact_c...
dwd_fact_p...
dwd_fact_f...
dwd_fact_c...
Sun, 12 Jul 2020 14:22:11
Value : 0
accuracy%
DataAssets
DQ Metrics
dwd_fact_order_detail
7/12/2020, 1:28 PM dwd_fact_orde...
test_meaure
7/12/2020, 1:16 PM TestJob