Apache Pulsar 实战篇

传智教育大数据平台

Stream Native

黑马程序员 www.itheima.com 传智教育旗下 高端IT教育品牌

PULSAR

目录
Contents

学习目标

Learning Objectives

1. 理解项目的架构实施
2. 掌握Canal集成Pulsar方案
3. 掌握Pulsar与Flink集成
4. 完成HBase对接Phoenix与Hive
5. 完成基于CK的实时数仓分析
6. 能够基于FineBI完成报表处理

01 传智教育大数据平台架构介绍

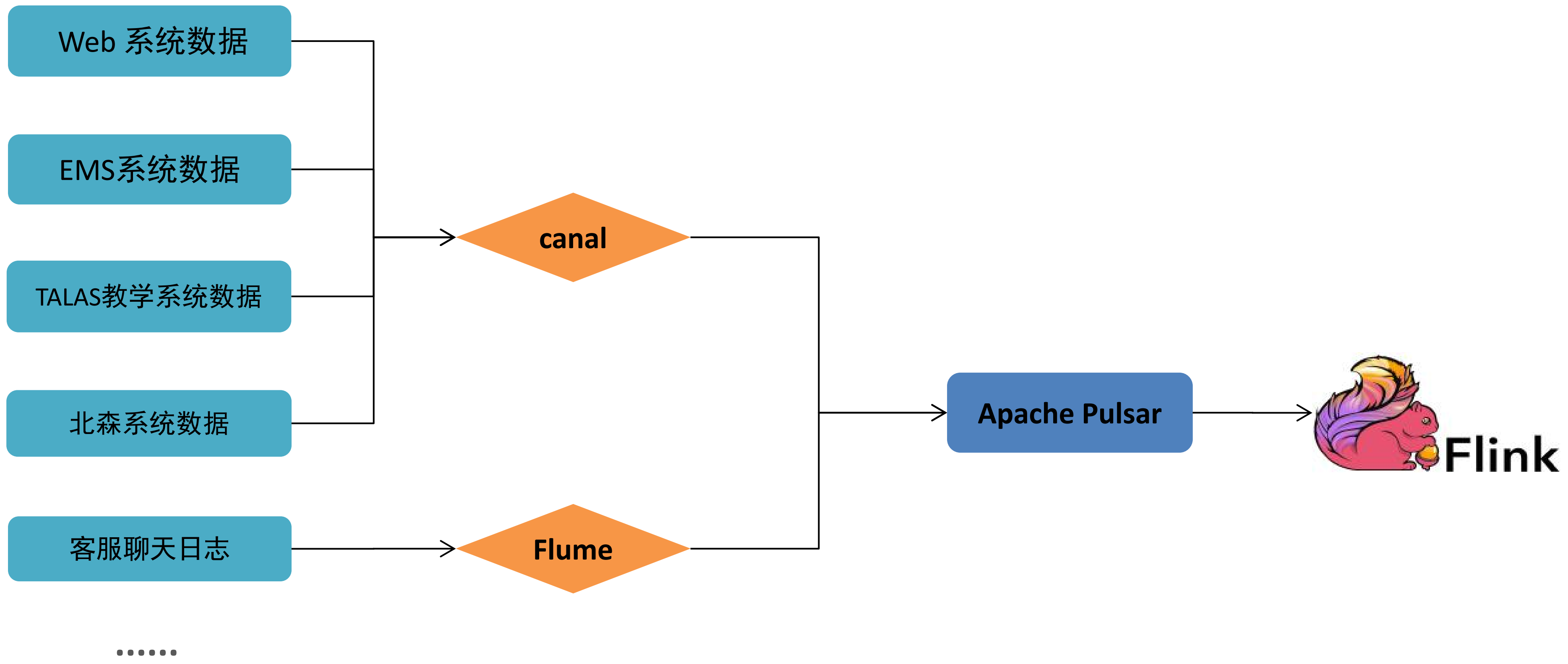
传智教育大数据平台基本介绍

大数据平台是传智教育在2016年初开始构建，最初始主要是进行离线的数仓平台构建,力争将公司核心数据(访问咨询数据,意向用户,报名数据以及学员考勤数据等)进行整合,对这些过往数据以天为单位进行挖掘分析，从而能够更加了解学员的相关的指标，能够更好的为学员服务

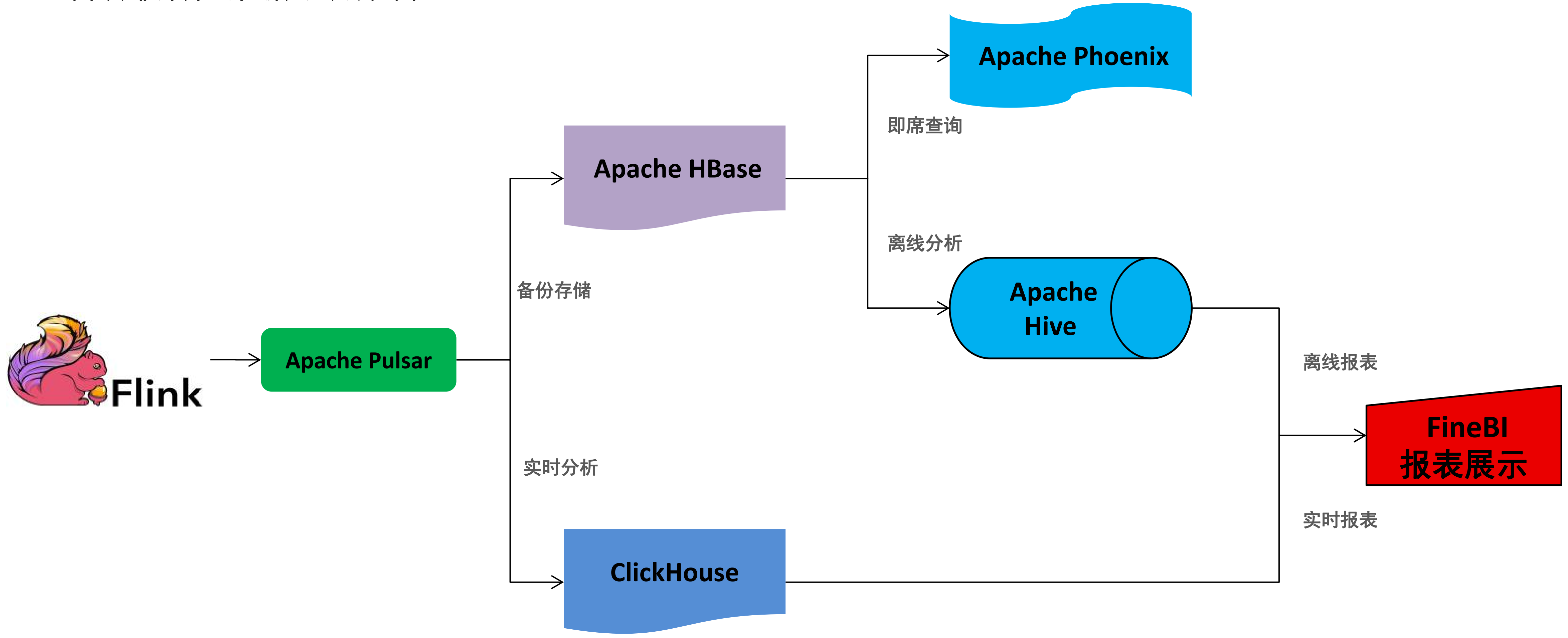
在2021年初，大数据平台开始引入流式的处理，主要采用Pulsar完成实时数据的传输，基于Flink进行实时数据预处理以及转换操作，最终基于CK完成实时指标统计，构建实时数仓

同时公司高层要求能够快速便捷的查询过去历史数据集，为了满足此需要，我们对离线平台做了重新设计,将其纳入流式平台中，构建基于HBase的实时查询系统以及离线分析平台，对整个大数据平台进行重构

传智教育大数据平台架构



传智教育大数据平台架构



02

传智教育实战项目介绍

Stream Native

黑马程序员 www.itheima.com 传智教育旗下 高端IT教育品牌

PULSAR

传智教育实战项目介绍

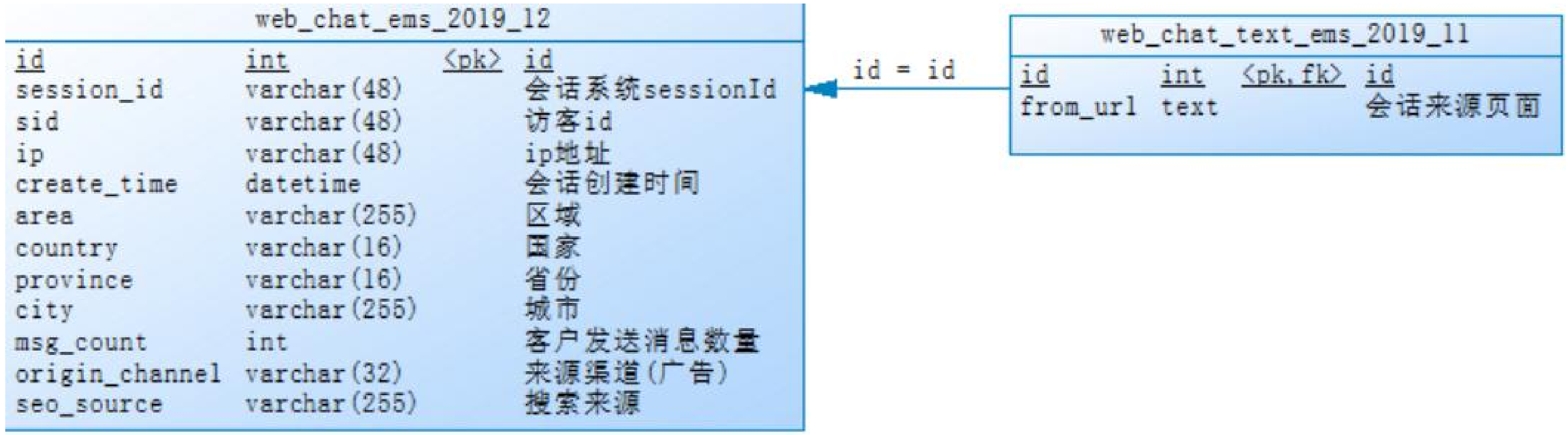
传智教育整个大数据平台大致设计了有以下多个主题的流批开发：访问咨询，意向用户主题，线索主题，报名用户主题，学生考勤主题，学员就业主题,标签画像等.

本次课程主要以访问咨询主题为代表，讲解传智教育整个项目开发流程以及实施方案

数据源说明：

本主题主要涉及到核心表有二个 web_chat_ems 表 和 web_chat_text_ems表

核心字段介绍：



web_chat_ems_2019_12

id	int	<pk>	id
session_id	varchar(48)		会话系统sessionId
sid	varchar(48)		访客id
ip	varchar(48)		ip地址
create_time	datetime		会话创建时间
area	varchar(255)		区域
country	varchar(16)		国家
province	varchar(16)		省份
city	varchar(255)		城市
msg_count	int		客户发送消息数量
origin_channel	varchar(32)		来源渠道(广告)
seo_source	varchar(255)		搜索来源

web_chat_text_ems_2019_11

id	int	<pk,fk>	id
from_url	text		会话来源页面

传智教育实战项目介绍

离线业务需求： 1. 总访问客户量 2. 地区独立访客 3. 访客咨询率：咨询率=发起咨询的人数/访问客户量 4. 客户访问量和访客咨询率趋势 5. 各时间段访问客户量 6. 各来源渠道/各搜索来源访问量 7. 活跃页面TOP10 指标： 访问量，咨询量 维度： 日期：年、月、天、小时 地区、来源渠道、搜索来源、受访页面	实时业务需求： 1. 总访问客户量 2. 总访客咨询量 3. 访客咨询率：咨询率=发起咨询的人数/访问客户量 指标： 1. 总访问量 2. 总咨询量 3. 咨询率 维度： 日期：当天，小时

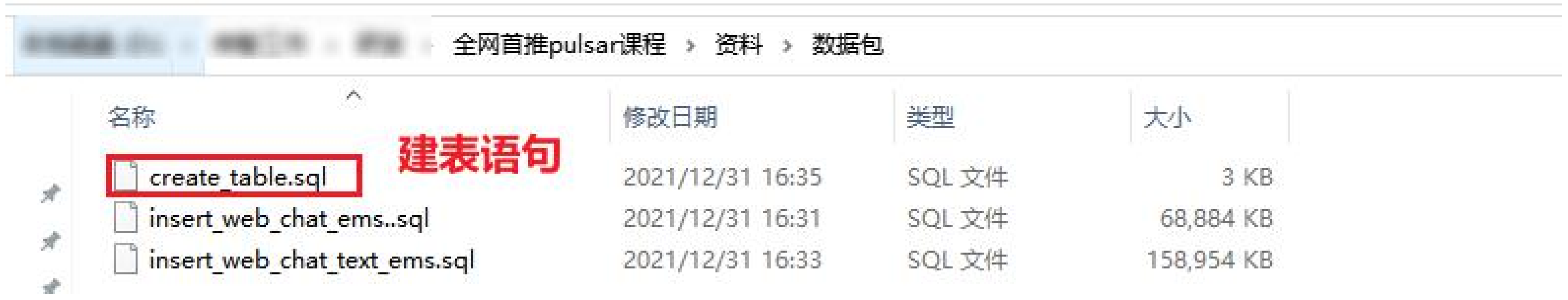
03 初始化数据源

初始化数据源：在Mysql中建库建表

- 1- 创建库：

```
CREATE DATABASE itcast_ems CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci ;
```

- 2- 构建表：构建 web_chat_ems 和 web_chat_text_ems


全网首推pulsar课程 › 资料 › 数据包

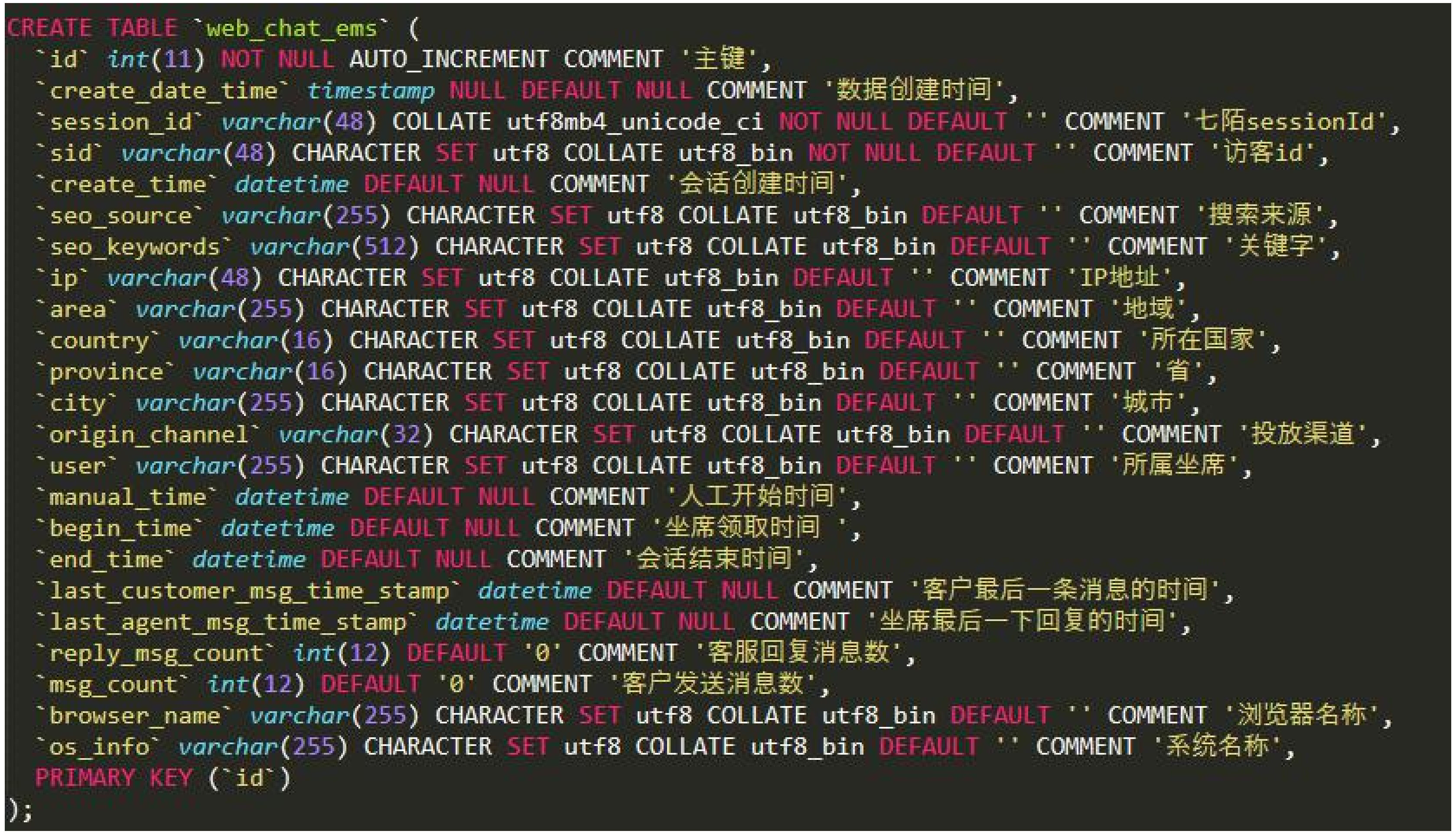
名称	修改日期	类型	大小
create_table.sql	2021/12/31 16:35	SQL 文件	3 KB
insert_web_chat_ems..sql	2021/12/31 16:31	SQL 文件	68,884 KB
insert_web_chat_text_ems.sql	2021/12/31 16:33	SQL 文件	158,954 KB

说明：

将文件中建表语句复制出来，进行执行，即可构建相关表

初始化数据源：在Mysql中建库建表

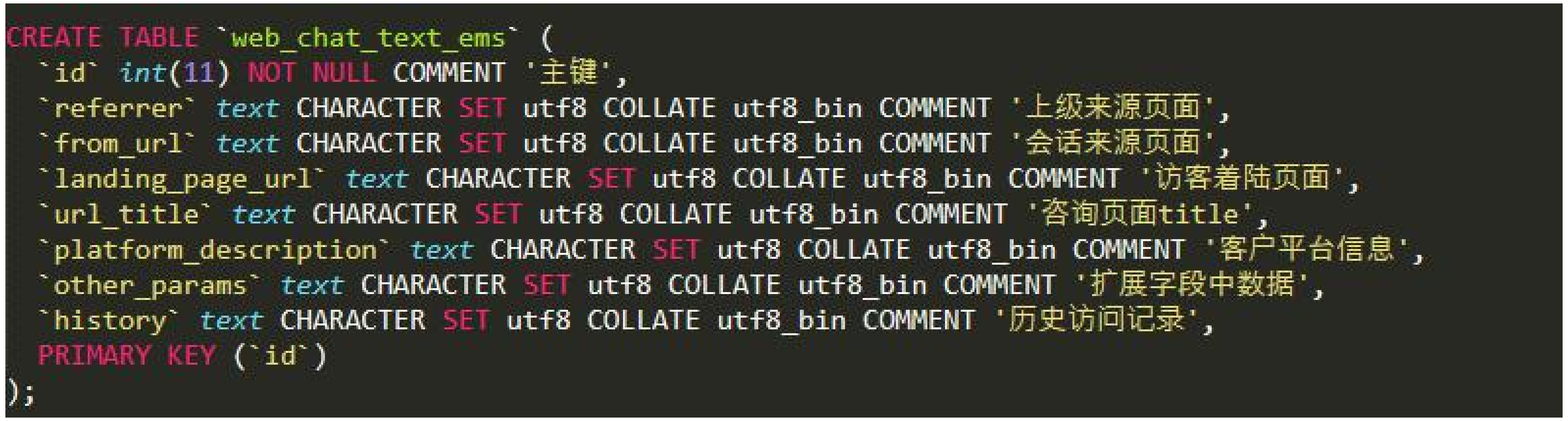
- 表字段完整说明：



```sql
CREATE TABLE `web_chat_ems` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT '主键',
  `create_date_time` timestamp NULL DEFAULT NULL COMMENT '数据创建时间',
  `session_id` varchar(48) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT '' COMMENT '七陌sessionId',
  `sid` varchar(48) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '' COMMENT '访客id',
  `create_time` datetime DEFAULT NULL COMMENT '会话创建时间',
  `seo_source` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT '' COMMENT '搜索来源',
  `seo_keywords` varchar(512) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT '' COMMENT '关键字',
  `ip` varchar(48) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT '' COMMENT 'IP地址',
  `area` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT '' COMMENT '地域',
  `country` varchar(16) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT '' COMMENT '所在国家',
  `province` varchar(16) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT '' COMMENT '省',
  `city` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT '' COMMENT '城市',
  `origin_channel` varchar(32) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT '' COMMENT '投放渠道',
  `user` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT '' COMMENT '所属坐席',
  `manual_time` datetime DEFAULT NULL COMMENT '人工开始时间',
  `begin_time` datetime DEFAULT NULL COMMENT '坐席领取时间 ',
  `end_time` datetime DEFAULT NULL COMMENT '会话结束时间',
  `last_customer_msg_time_stamp` datetime DEFAULT NULL COMMENT '客户最后一条消息的时间',
  `last_agent_msg_time_stamp` datetime DEFAULT NULL COMMENT '坐席最后一下回复的时间',
  `reply_msg_count` int(12) DEFAULT '0' COMMENT '客服回复消息数',
  `msg_count` int(12) DEFAULT '0' COMMENT '客户发送消息数',
  `browser_name` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT '' COMMENT '浏览器名称',
  `os_info` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT '' COMMENT '系统名称',
  PRIMARY KEY (`id`)
);
```

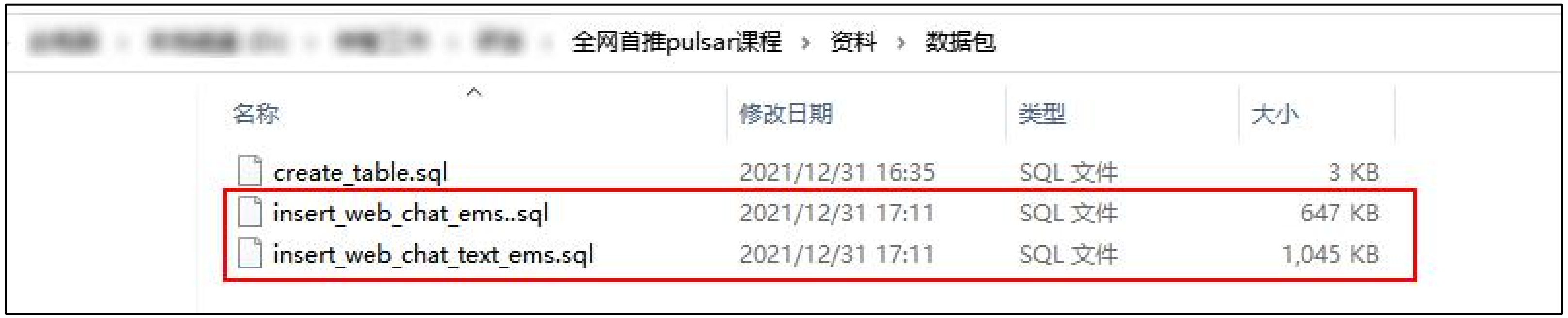
初始化数据源：在Mysql中建库建表

● 表字段完整说明：



```
CREATE TABLE `web_chat_text_ems` (
  `id` int(11) NOT NULL COMMENT '主键',
  `referrer` text CHARACTER SET utf8 COLLATE utf8_bin COMMENT '上级来源页面',
  `from_url` text CHARACTER SET utf8 COLLATE utf8_bin COMMENT '会话来源页面',
  `landing_page_url` text CHARACTER SET utf8 COLLATE utf8_bin COMMENT '访客着陆页面',
  `url_title` text CHARACTER SET utf8 COLLATE utf8_bin COMMENT '咨询页面title',
  `platform_description` text CHARACTER SET utf8 COLLATE utf8_bin COMMENT '客户平台信息',
  `other_params` text CHARACTER SET utf8 COLLATE utf8_bin COMMENT '扩展字段中数据',
  `history` text CHARACTER SET utf8 COLLATE utf8_bin COMMENT '历史访问记录',
  PRIMARY KEY (`id`)
);
```

本次两个表各提供了1000条数据,进行后续的测试操作,目前暂时不导入



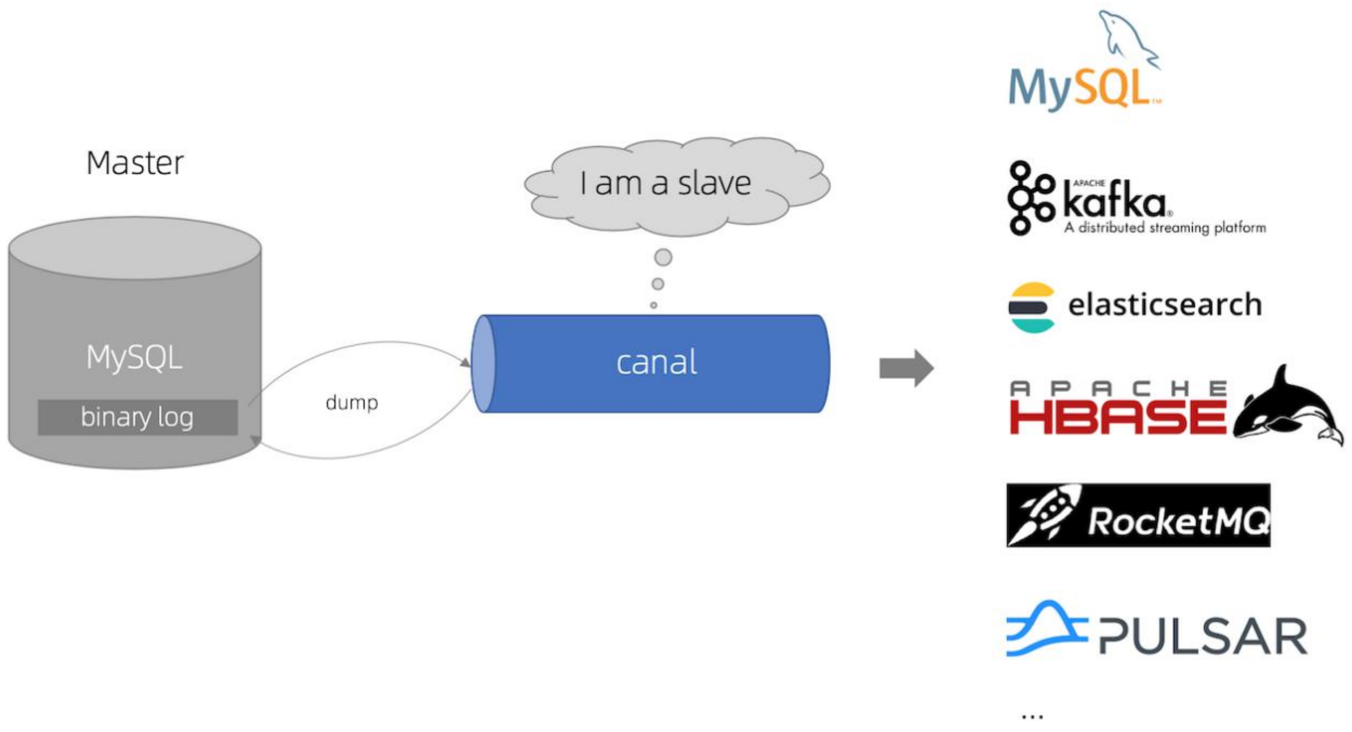
[redacted] › 全网首推pulsar课程 › 资料 › 数据包

名称	修改日期	类型	大小
create_table.sql	2021/12/31 16:35	SQL 文件	3 KB
insert_web_chat_ems..sql	2021/12/31 17:11	SQL 文件	647 KB
insert_web_chat_text_ems.sql	2021/12/31 17:11	SQL 文件	1,045 KB

04

基于Canal采集数据到Pulsar

Canal基本介绍



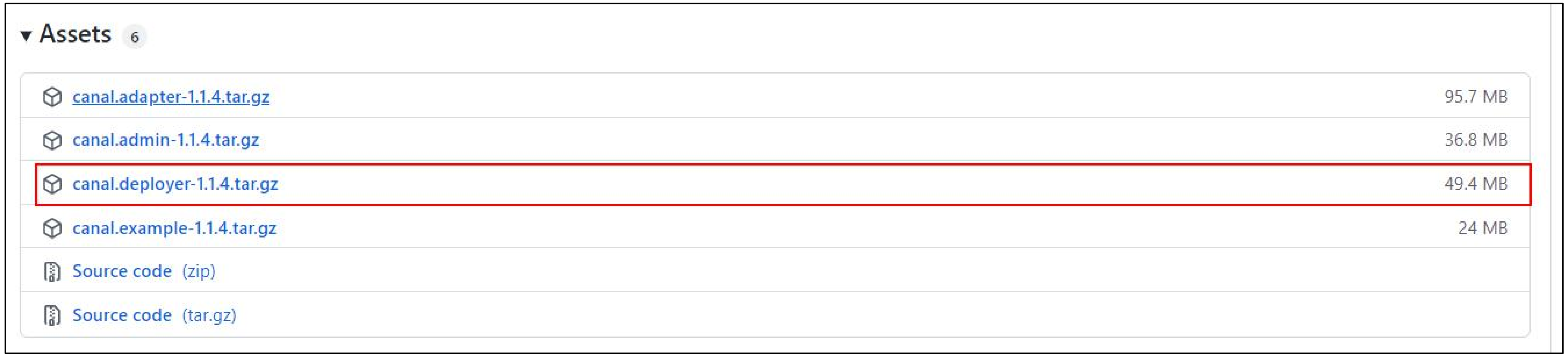
- 基于 MySQL 数据库增量日志解析，提供增量数据订阅和消费
- 早期阿里巴巴因为杭州和美国双机房部署，存在跨机房同步的业务需求，实现方式主要是基于业务 trigger（触发器） 获取增量变更

Canal基本介绍

- 从 2010 年开始，业务逐步尝试数据库日志解析获取增量变更进行同步，由此衍生出了大量的数据库增量订阅和消费业务，基于日志增量订阅和消费的业务包括
  - 数据库镜像
  - 数据库实时备份
  - 索引构建和实时维护(拆分异构索引、倒排索引等)
  - 业务 cache 刷新
  - 带业务逻辑的增量数据处理
- 当前的 canal 支持源端 MySQL 版本包括 5.1.x , 5.5.x , 5.6.x , 5.7.x , 8.0.x
- github地址：https://github.com/alibaba/canal

Canal安装

下载地址：https://github.com/alibaba/canal/releases/tag/canal-1.1.4/



▾ Assets 6

canal.adapter-1.1.4.tar.gz	95.7 MB
canal.admin-1.1.4.tar.gz	36.8 MB
canal.deployer-1.1.4.tar.gz	49.4 MB
canal.example-1.1.4.tar.gz	24 MB
Source code (zip)	
Source code (tar.gz)	

或：大家可以直接选择资料中提供好的下载包

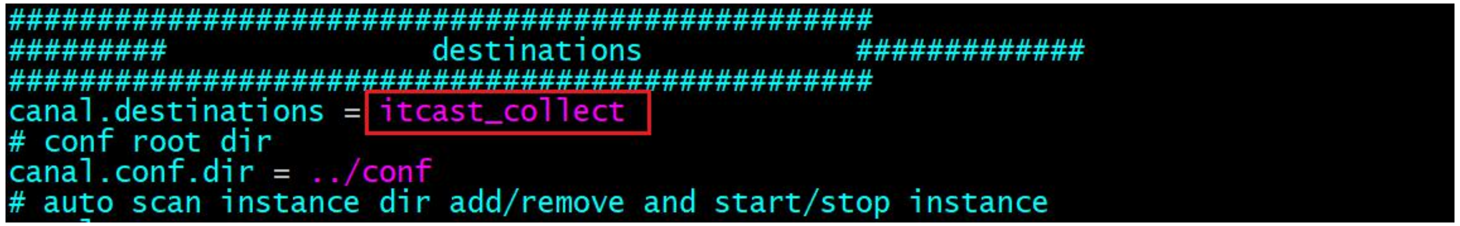
- 1- 将下载好的canal安装包 上传到/export/software 目录下
- 2- 在/export/servers/创建文件夹canal，将canal.deployer-1.1.4.tar.gz 解压到该目录

```
mkdir /export/servers/canal
tar -zxvf canal.deployer-1.1.4.tar.gz -C /export/server/canal
```

基于canal采集MySQL数据到Pulsar

- 1- 修改canal下的conf目录中canal.properties

```
cd /export/server/canal/conf
vim canal.properties
内容如下:
修改96行: 指定采集目的地的配置文件路径所在的目录名字
canal.destinations = itcast_collect

注: 可以配置多个, 配置几个, 后续就得按照这个名字配置几个目录, 构建目的地配置信息
```



```
#################################################
#########               destinations            ##############
#################################################
canal.destinations = itcast_collect
# conf root dir
canal.conf.dir = ../conf
# auto scan instance dir add/remove and start/stop instance
```

- 2- 在conf目录下，创建 itcast_collect 目录

```
cd  /export/server/canal/conf
mkdir -p itcast_collect
```

基于canal采集MySQL数据到Pulsar

- 3- 将conf下的example目录中的instance.properties 拷贝到刚刚创建的 itcast_collect 中

```
cd /export/server/canal/conf
cp example/instance.properties itcast_collect/
```

- 4- 进入itcast_collect, 编辑instance.properties文件

```
cd  /export/server/canal/conf/itcast_collect
vim instance.properties

修改如下:
# 第9行
canal.instance.master.address=node1.itcast.cn:3306
# 第 33和34行:
canal.instance.dbUsername=root
canal.instance.dbPassword=123456
# 第 41行:
canal.instance.filter.regex=itcast_ems\\..*
#50行: 删除topic名称
canal.mq.topic=
```

基于canal采集MySQL数据到Pulsar

- 5- 在Pulsar配置与canal对接的配置信息

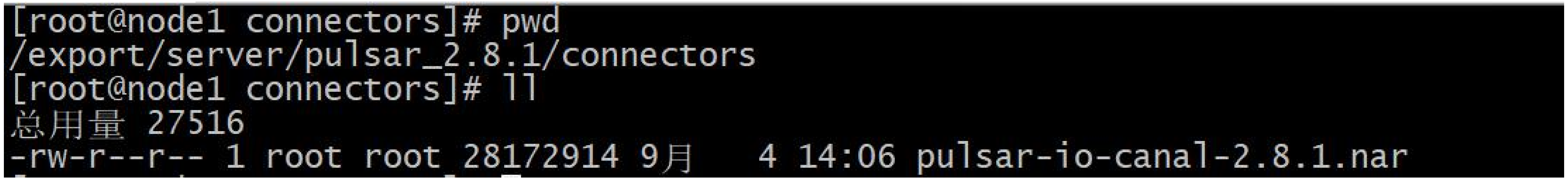
```
cd /export/server/pulsar_2.8.1/conf/
vim canal-mysql-source-config.yaml

内容如下:
configs:
   zkServers: ""
   batchSize: "1"
   destination: "itcast_collect"
   username: "canal"
   password: "canal"
   cluster: false
   singleHostname: "node1.itcast.cn"
   singlePort: "11111"
```

基于canal采集MySQL数据到Pulsar

- 6-下载Pulsar与Canal集成的connector IO 依赖包

```
cd /export/server/pulsar_2.8.1
mkdir -p connectors
进入connectors目录:
cd /export/server/brokers/connectors

执行下载
wget https://archive.apache.org/dist/pulsar/pulsar-2.8.1/connectors/pulsar-io-canal-2.8.1.nar
```

或者，可以选择直接将资料中提供好的io包进行上传即可

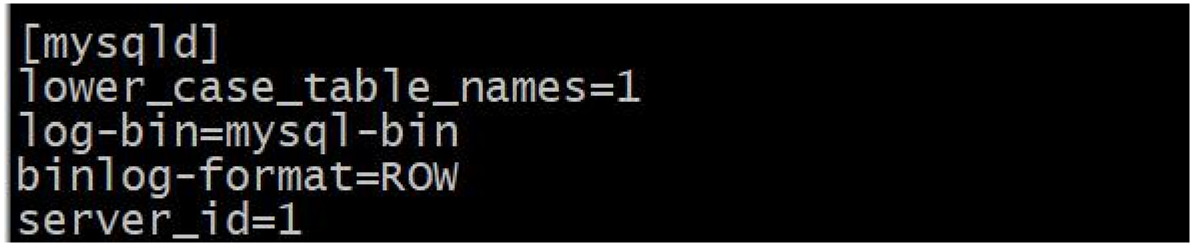


```
[root@node1 connectors]# pwd
/export/server/pulsar_2.8.1/connectors
[root@node1 connectors]# ll
总用量 27516
-rw-r--r-- 1 root root 28172914 9月   4 14:06 pulsar-io-canal-2.8.1.nar
```

开启Mysql BinLog日志

- 修改mysql的配置文件(本次课程 MySQL安装在node1节点上)

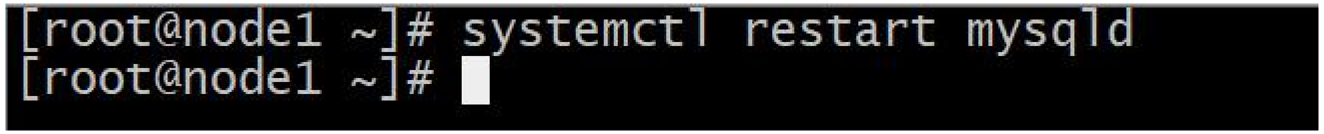
```
vim /etc/m.cnf

在[mysqld]标签下，新增如下代码：
lower_case_table_names=1
log-bin=mysql-bin # 开启 binlog
binlog-format=ROW # 选择 ROW 模式,可选值[STATEMENT(记录SQL) |ROW(记录数据) | MIXED(混合使用) ]
server_id=1 # 配置 MySQL 主从复制,需要定义，不要和 canal 的 slaveId 重复
```



```
[mysqld]
lower_case_table_names=1
log-bin=mysql-bin
binlog-format=ROW
server_id=1
```

- 重启mysql服务

```
systemctl restart mysqld
```



```
[root@node1 ~]# systemctl restart mysqld
[root@node1 ~]# 
```

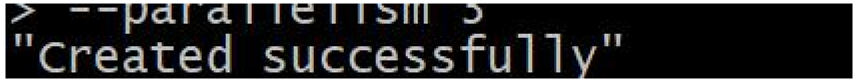
启动 Pulsar Connectors

- 1- 在Pulsar中创建Topic

```
bin/pulsar-admin topics create-partitioned-topic persistent://public/default/itcast_canal_collect  --partitions 3
```

- 2- 创建并启动connector

```
./bin/pulsar-admin source create \
--archive ./connectors/pulsar-io-canal-2.8.1.nar \
--classname org.apache.pulsar.io.canal.CanalStringSource \
--tenant public \
--namespace default \
--name canal_collect \
--destination-topic-name itcast_canal_collect \
--source-config-file /export/server/pulsar_2.8.1/conf/canal-mysql-source-config.yaml \
--parallelism 3
```

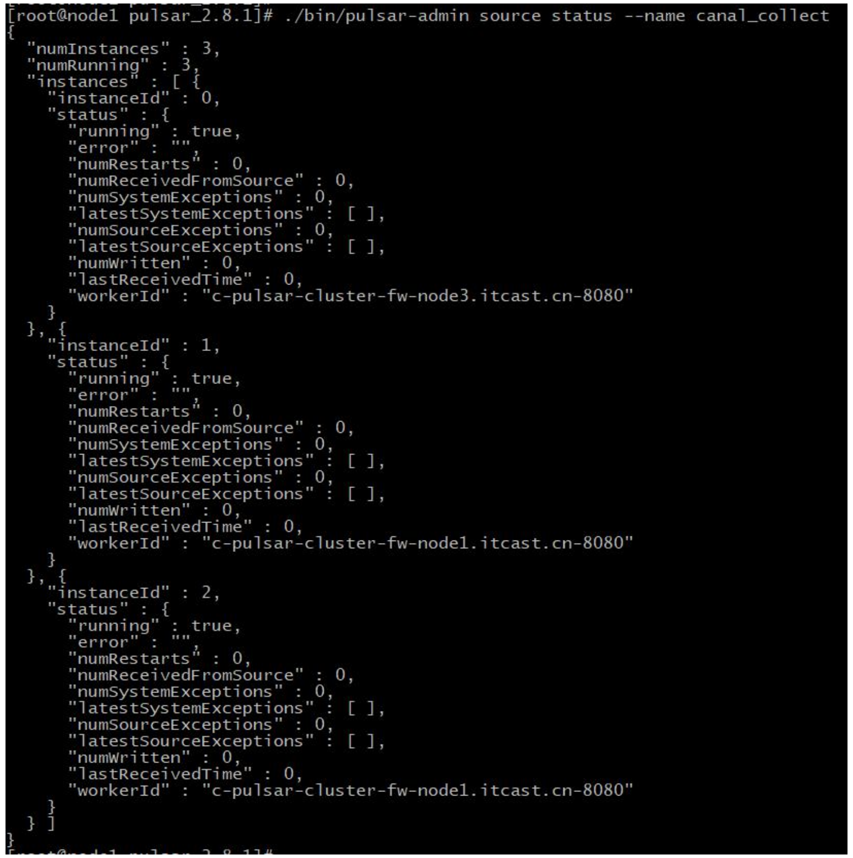


```
> --parallelism 3
"Created successfully"
```

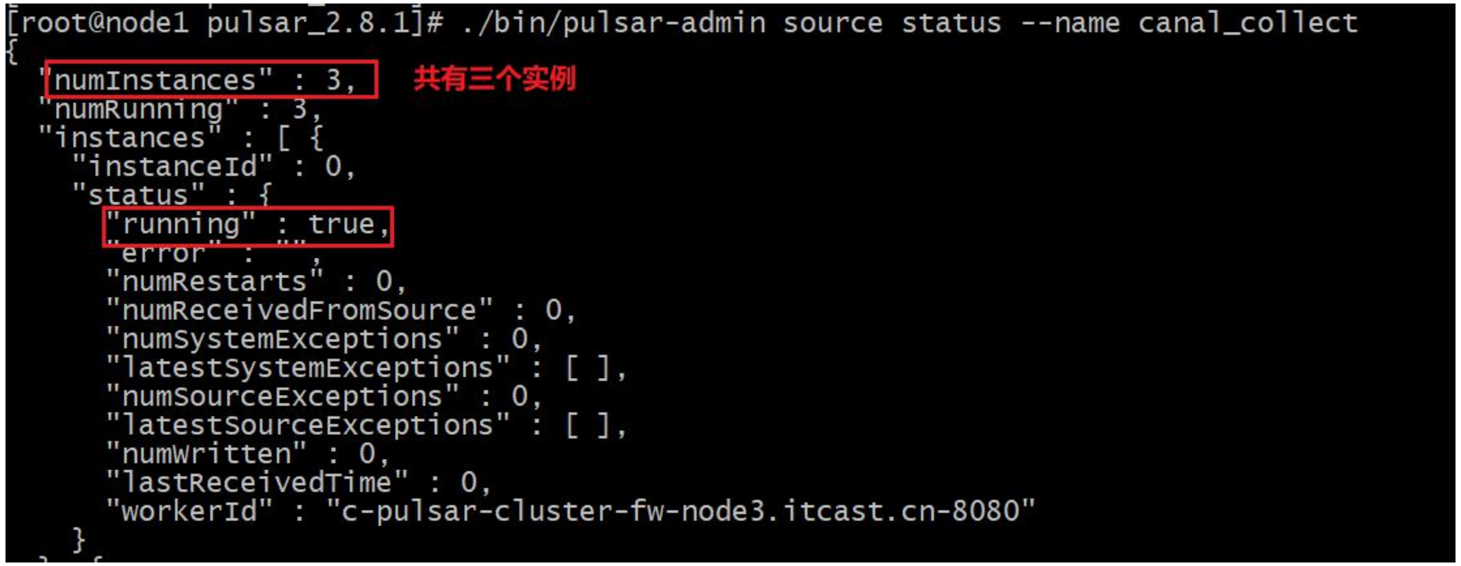
启动 Pulsar Connectors

- 3- 查看状态信息

```
./bin/pulsar-admin source status --name canal_collect
```



```
[root@node1 pulsar_2.8.1]# ./bin/pulsar-admin source status --name canal_collect
{
  "numInstances" : 3,
  "numRunning" : 3,
  "instances" : [ {
    "instanceId" : 0,
    "status" : {
      "running" : true,
      "error" : "",
      "numRestarts" : 0,
      "numReceivedFromSource" : 0,
      "numSystemExceptions" : 0,
      "latestSystemExceptions" : [ ],
      "numSourceExceptions" : 0,
      "latestSourceExceptions" : [ ],
      "numWritten" : 0,
      "lastReceivedTime" : 0,
      "workerId" : "c-pulsar-cluster-fw-node3.itcast.cn-8080"
    }
  }, {
    "instanceId" : 1,
    "status" : {
      "running" : true,
      "error" : "",
      "numRestarts" : 0,
      "numReceivedFromSource" : 0,
      "numSystemExceptions" : 0,
      "latestSystemExceptions" : [ ],
      "numSourceExceptions" : 0,
      "latestSourceExceptions" : [ ],
      "numWritten" : 0,
      "lastReceivedTime" : 0,
      "workerId" : "c-pulsar-cluster-fw-node1.itcast.cn-8080"
    }
  }, {
    "instanceId" : 2,
    "status" : {
      "running" : true,
      "error" : "",
      "numRestarts" : 0,
      "numReceivedFromSource" : 0,
      "numSystemExceptions" : 0,
      "latestSystemExceptions" : [ ],
      "numSourceExceptions" : 0,
      "latestSourceExceptions" : [ ],
      "numWritten" : 0,
      "lastReceivedTime" : 0,
      "workerId" : "c-pulsar-cluster-fw-node1.itcast.cn-8080"
    }
  } ]
}
```



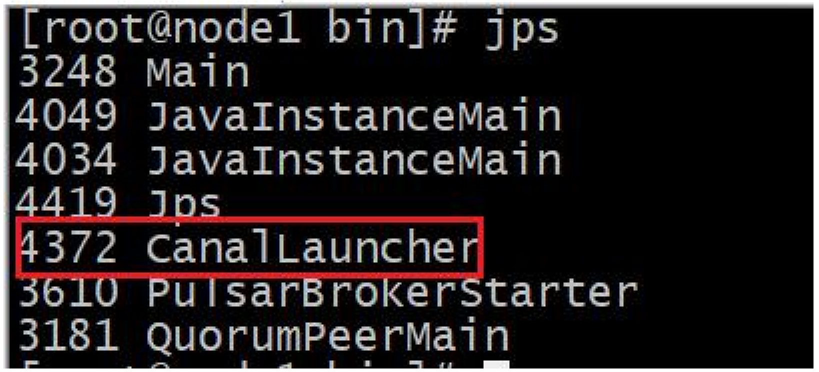
```
[root@node1 pulsar_2.8.1]# ./bin/pulsar-admin source status --name canal_collect
{
  "numInstances" : 3,
  "numRunning" : 3,
  "instances" : [ {
    "instanceId" : 0,
    "status" : {
      "running" : true,
      "error" : "",
      "numRestarts" : 0,
      "numReceivedFromSource" : 0,
      "numSystemExceptions" : 0,
      "latestSystemExceptions" : [ ],
      "numSourceExceptions" : 0,
      "latestSourceExceptions" : [ ],
      "numWritten" : 0,
      "lastReceivedTime" : 0,
      "workerId" : "c-pulsar-cluster-fw-node3.itcast.cn-8080"
    }
```

共有三个实例

启动 Canal开始进行采集数据

- 1- 查看状态信息

```
cd /export/servers/canal/bin
sh startup.sh
```

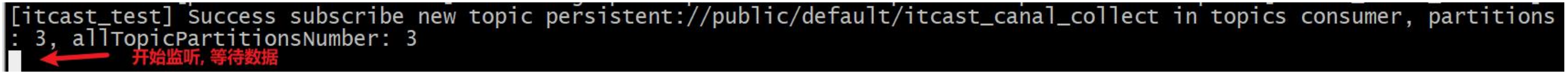


```
[root@node1 bin]# jps
3248 Main
4049 JavaInstanceMain
4034 JavaInstanceMain
4419 Jps
4372 CanalLauncher
3610 PulsarBrokerStarter
3181 QuorumPeerMain
```

测试:

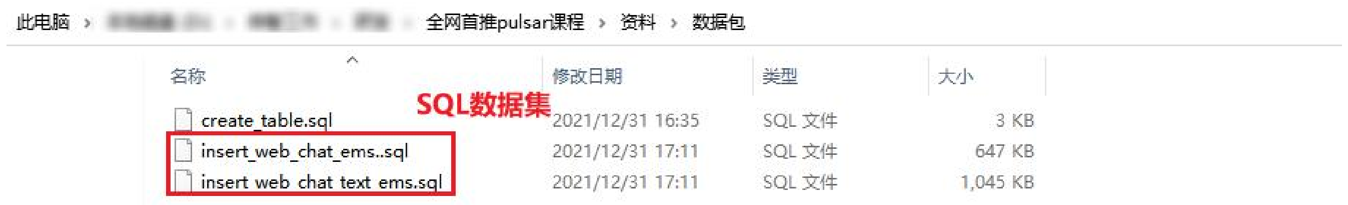
- 1- 在Pulsar中启动一个消费者,用于监听Canal是否可以将数据采集到Pulsar中

```
./bin/pulsar-client consume -s 'itcast_test' -n 0 itcast_canal_collect
```



```
[itcast_test] Success subscribe new topic persistent://public/default/itcast_canal_collect in topics consumer, partitions
: 3, allTopicPartitionsNumber: 3
```

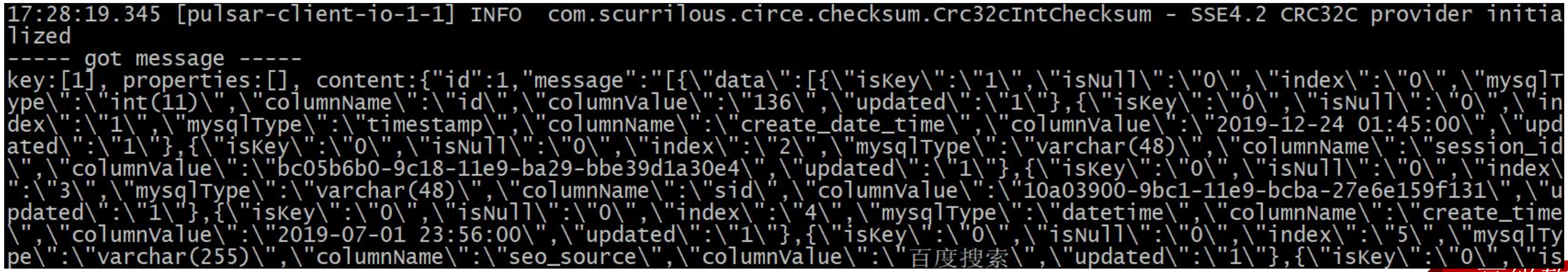
- 2- 将项目中对两个表中添加数据的SQL语句各拿出一条， 添加到mysql中， 观察是否可以采集到



此电脑 › [redacted] › [redacted] › [redacted] › 全网首推pulsar课程 › 资料 › 数据包

名称	修改日期	类型	大小
create_table.sql	2021/12/31 16:35	SQL 文件	3 KB
insert_web_chat_ems..sql	2021/12/31 17:11	SQL 文件	647 KB
insert_web_chat_text_ems.sql	2021/12/31 17:11	SQL 文件	1,045 KB

- 消费者成功收到消息



```
17:28:19.345 [pulsar-client-io-1-1] INFO  com.scurrilous.circe.checksum.Crc32cIntChecksum - SSE4.2 CRC32C provider initia
lized
----- got message -----
key:[1], properties:[], content:{"id":1,"message":"[{\"data\":[{\"isKey\":\"1\",\"isNull\":\"0\",\"index\":\"0\",\"mysqlT
ype\":\"int(11)\",\"columnName\":\"id\",\"columnValue\":\"136\",\"updated\":\"1\"},{\"isKey\":\"0\",\"isNull\":\"0\",\"in
dex\":\"1\",\"mysqlType\":\"timestamp\",\"columnName\":\"create_date_time\",\"columnValue\":\"2019-12-24 01:45:00\",\"upd
ated\":\"1\"},{\"isKey\":\"0\",\"isNull\":\"0\",\"index\":\"2\",\"mysqlType\":\"varchar(48)\",\"columnName\":\"session_id
\",\"columnValue\":\"bc05b6b0-9c18-11e9-ba29-bbe39d1a30e4\",\"updated\":\"1\"},{\"isKey\":\"0\",\"isNull\":\"0\",\"index\
":\"3\",\"mysqlType\":\"varchar(48)\",\"columnName\":\"sid\",\"columnValue\":\"10a03900-9bc1-11e9-bcba-27e6e159f131\",\"u
pdated\":\"1\"},{\"isKey\":\"0\",\"isNull\":\"0\",\"index\":\"4\",\"mysqlType\":\"datetime\",\"columnName\":\"create_time
\",\"columnValue\":\"2019-07-01 23:56:00\",\"updated\":\"1\"},{\"isKey\":\"0\",\"isNull\":\"0\",\"index\":\"5\",\"mysqlTy
pe\":\"varchar(255)\",\"columnName\":\"seo_source\",\"columnValue\":\"百度搜索\",\"updated\":\"1\"},{\"isKey\":\"0\",\"is
```

05

Pulsar对接Flink完成数据预处理

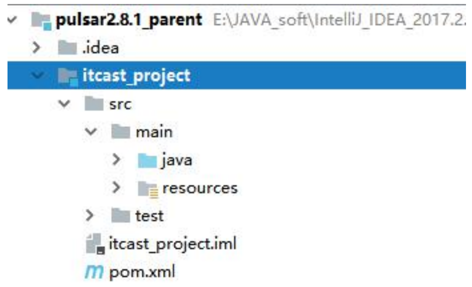
预处理需求说明

- 1- 两个表需要进行Join合并，抽取需求核心相关字段，形成宽表数据
- 2- 对表进行拉宽操作：create_time字段 拉宽为 年、月、天、小时

核心字段:
- web_chat_em:
  - join字段: id
  - 时间维度: create_time
  - 地区维度: area
  - 来源渠道: origin_channel
  - 搜索来源: seo_source
  - 指标字段: sid ,session_id,ip
- web_chat_text_ems_2019_12:
  - 受访页面维度: from_url
  - join字段: id

构建Maven项目,加入依赖

- 1- 创建maven项目： itcast_project



```
pulsar2.8.1_parent  E:\JAVA_soft\IntelliJ_IDEA_2017.2.
  .idea
  itcast_project
    src
      main
        java
        resources
      test
    itcast_project.iml
    pom.xml
```

- 2- 加入相关依赖

```
<repositories><!--代码库-->
    <repository>
        <id>aliyun</id>
        <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
        <releases><enabled>true</enabled></releases>
        <snapshots>
            <enabled>false</enabled>
            <updatePolicy>never</updatePolicy>
        </snapshots>
    </repository>
</repositories>
```

构建Maven项目,加入依赖

```
<dependencies>
  <dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-table-api-java-bridge_2.11</artifactId>
    <version>1.13.1</version>
  </dependency>
  <dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-table-planner-blink_2.11</artifactId>
    <version>1.13.1</version>
  </dependency>
  <dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-streaming-scala_2.11</artifactId>
    <version>1.13.1</version>
  </dependency>
  <dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-table-common</artifactId>
    <version>1.13.1</version>
  </dependency>
```

构建Maven项目,加入依赖

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-clients_2.11</artifactId>
  <version>1.13.1</version>
</dependency>
<dependency>
  <groupId>io.streamnative.connectors</groupId>
  <artifactId>pulsar-flink-connector_2.11</artifactId>
  <version>1.13.1.5-rc1</version>

  <exclusions>
    <exclusion>
      <groupId>org.apache.pulsar</groupId>
      <artifactId>pulsar-client-all</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>org.apache.pulsar</groupId>
  <artifactId>pulsar-client-all</artifactId>
  <version>2.8.1</version>
</dependency>
```

构建Maven项目,加入依赖

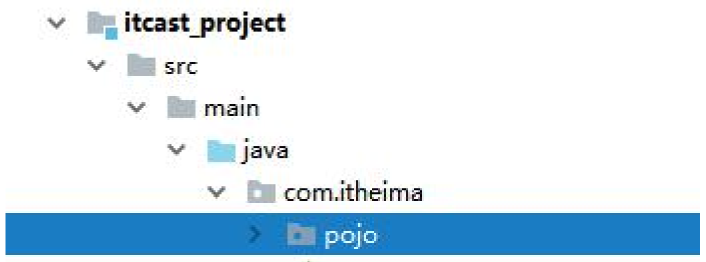
```
 <dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.62</version>
 </dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-connector-hbase-2.2_2.11</artifactId>
    <version>1.13.1</version>
 </dependency>

 <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-common</artifactId>
    <version>3.0.0</version>
 </dependency>
 <dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-jdbc_2.11</artifactId>
    <version>1.10.1</version>
 </dependency>
```

构建Maven项目,加入依赖

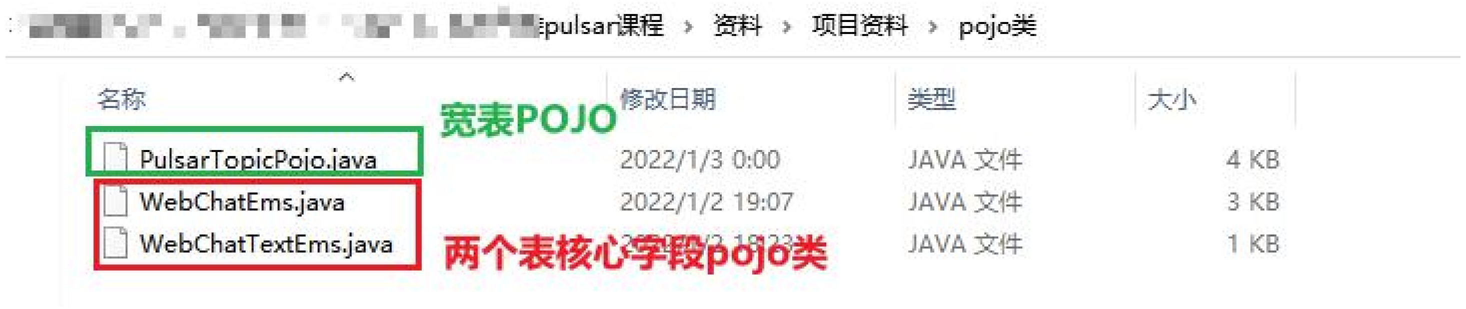
```
        <dependency>
            <groupId>ru.yandex.clickhouse</groupId>
            <artifactId>clickhouse-jdbc</artifactId>
            <version>0.3.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpcore</artifactId>
            <version>4.4.13</version>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>
                    <target>1.8</target>
                    <source>1.8</source>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

添加相关POJO类

- 1- 在项目中创建: com.itheima.pojo



- 2- 加入资料中Pojo类到此包下



名称	修改日期	类型	大小
PulsarTopicPojo.java	2022/1/3 0:00	JAVA 文件	4 KB
WebChatEms.java	2022/1/2 19:07	JAVA 文件	3 KB
WebChatTextEms.java	2022/1/2 18:23	JAVA 文件	1 KB

编写Flink代码，对数据进行处理操作

● 1- Flink与Pulsar对接, 完成数据消息：Pulsar Connector Flink Source

```
public class ItcastEmsFlink {
    private static PulsarTopicPojo pulsarTopicPojo = new PulsarTopicPojo();
    public static void main(String[] args) throws Exception {
        //1. 创建环境对象
        StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
        StreamTableEnvironment tableEnv = StreamTableEnvironment.create(env);

        //2. 设置 Source
        Properties props = new Properties();
        props.setProperty("topic", "persistent://public/default/itcast_canal_collect");
        props.setProperty("partition.discovery.interval-millis", "5000");
        FlinkPulsarSource<String> pulsarSource = new FlinkPulsarSource<String>(
                "pulsar://node1:6650,node2:6650,node3:6650", "http://node1:8080,node2:8080,node3:8080",
                PulsarDeserializationSchema.valueOnly(new SimpleStringSchema()), props);

        pulsarSource.setStartFromLatest();
        DataStreamSource<String> source = env.addSource(pulsarSource);
```

编写Flink代码，对数据进行处理操作

- 2- 对数据进行转换处理 -- 过滤无用消息数据

```
//3.1: 过滤掉无消息的数据
SingleOutputStreamOperator<String> filterDataStream = source.filter(new FilterFunction<String>() {
  @Override
  public boolean filter(String msg) throws Exception {
    Map<String, Object> msgMap = JSONObject.parseObject(msg, Map.class);
    return !"[]".equals(msgMap.get("message"));
  }
});
```

- 3- 获取其中Data数据: 先获取webChatEms核心字段数据

```
    SingleOutputStreamOperator<WebChatEms> webChatEmsDataStream = filterDataStream.flatMap(new
FlatMapFunction<String, WebChatEms>() {
      @Override
      public void flatMap(String canalJson, Collector<WebChatEms> collector) throws Exception {

        Map<String, Object> msgMap = JSONObject.parseObject(canalJson, Map.class);
        String message = (String) msgMap.get("message");

        List<Map<String, Object>> canalMsgPojos = (List<Map<String, Object>>) JSON.parse(message);
```

编写Flink代码，对数据进行处理操作

```
        for (Map<String, Object> canalMsg : canalMsgPojos) {
            String type = (String) canalMsg.get("type");
            if ("INSERT".equals(type)) {
                String tableName = (String) canalMsg.get("table");
                if("web_chat_ems".equals(tableName)){
                    List<Map<String, String>> rowMap = (List<Map<String, String>>) canalMsg.get("data");
                    WebChatEms webChatEms = new WebChatEms();
                    for (Map<String, String> colAndVal : rowMap) {
                        String columnName = colAndVal.get("columnName");
                        String columnValue = colAndVal.get("columnValue");
                        if("id".equals(columnName))  webChatEms.setId(Integer.parseInt(columnValue));
                        if("create_time".equals(columnName)) webChatEms.setCreate_time(columnValue);
                        if("area".equals(columnName)) webChatEms.setArea(columnValue);
                        if("origin_channel".equals(columnName)) webChatEms.setOrigin_channel(columnValue);
                        if("seo_source".equals(columnName)) webChatEms.setSeo_source(columnValue);
                        if("sid".equals(columnName)) webChatEms.setSid(columnValue);
                        if("session_id".equals(columnName)) webChatEms.setSession_id(columnValue);
                        if("ip".equals(columnName))webChatEms.setIp(columnValue);
                        if("msg_count".equals(columnName) && columnValue != null){
                            webChatEms.setMsg_count(Integer.parseInt(columnValue));
                        }
                    }
                    collector.collect(webChatEms);
                }
            }
        } }
    });
```

编写Flink代码，对数据进行处理操作

- 4- 获取其中Data数据: 先获取webChatTextEms核心字段数据

```
    SingleOutputStreamOperator<WebChatTextEms> webChatTextEmsDataStream = filterDataStream.flatMap(new FlatMapFunction<String,
WebChatTextEms>() {
        @Override
        public void flatMap(String canalJson, Collector<WebChatTextEms> collector) throws Exception {
            Map<String, Object> msgMap = JSONObject.parseObject(canalJson, Map.class);
            String message = (String) msgMap.get("message");
            List<Map<String, Object>> canalMsgPojos = (List<Map<String, Object>>) JSON.parse(message);
            for (Map<String, Object> canalMsg : canalMsgPojos) {
                String type = (String) canalMsg.get("type");
                if ("INSERT".equals(type)) {
                    String tableName = (String) canalMsg.get("table");
                    if("web_chat_text_ems".equals(tableName)){
                        List<Map<String, String>> rowMap = (List<Map<String, String>>) canalMsg.get("data");
                        WebChatTextEms webChatTextEms = new WebChatTextEms();
                        for (Map<String, String> colAndVal : rowMap) {
                            String columnName = colAndVal.get("columnName");
                            String columnValue = colAndVal.get("columnValue");
                            if("id".equals(columnName))webChatTextEms.setId(Integer.parseInt(columnValue));
                            if("from_url".equals(columnName))webChatTextEms.setFrom_url(columnValue);
                        }
                        collector.collect(webChatTextEms);
                    }
            }}}
    });
```

编写Flink代码，对数据进行处理操作

- 5- 转换为Flink Sql API准备对数据处理

```
Schema webChatEmsSchema = Schema.newBuilder()
        .column("id", DataTypes.INT())
        .column("session_id", DataTypes.STRING())
        .column("sid", DataTypes.STRING())
        .column("create_time", DataTypes.STRING())
        .column("seo_source", DataTypes.STRING())
        .column("ip", DataTypes.STRING())
        .column("area", DataTypes.STRING())
        .column("origin_channel", DataTypes.STRING())
        .column("msg_count", DataTypes.INT())
        .build();

tableEnv.createTemporaryView("web_chat_ems",webChatEmsDataStream,webChatEmsSchema);

Schema webChatTextEmsSchema = Schema.newBuilder()
        .column("id", DataTypes.INT())
        .column("from_url", DataTypes.STRING())
        .build();

tableEnv.createTemporaryView("web_chat_text_ems",webChatTextEmsDataStream,webChatTextEmsSchema);
```

编写Flink代码，对数据进行处理操作

● 6- 编写SQL进行数据转换合并拉宽数据

```
        //3.4: 进行数据合并拉宽转换操作
        Table table = tableEnv.sqlQuery("select wce.id,wce.sid,wce.ip,wce.session_id,wce.create_time,year(wce.create_time) as
yearInfo,month(wce.create_time) as monthInfo,day(wce.create_time) as dayInfo,hour(wce.create_time) as
hourInfo,wce.seo_source,wce.area,wce.origin_channel,wce.msg_count,wcte.from_url from web_chat_ems wce join web_chat_text_ems wcte on
wce.id = wcte.id");
```

● 7- 将Table转换DS, 对数据进行处理, 准备进行写回Pulsar处理

```
//3.5: 转换为DS 准备输出操作
        DataStream<Row> rowDataStream = tableEnv.toDataStream(table);
        SingleOutputStreamOperator<PulsarTopicPojo> pulsarTopicDataStream = rowDataStream.map(new MapFunction<Row, PulsarTopicPojo>() {
            @Override
            public PulsarTopicPojo map(Row row) throws Exception {
                Integer id = (Integer) row.getField("id");
                String sid = (String) row.getField("sid");
                String ip = (String) row.getField("ip");
                String session_id = (String) row.getField("session_id");
                String create_time = (String) row.getField("create_time");
                String yearInfo = (String) row.getField("yearInfo");
                String monthInfo = (String) row.getField("monthInfo");
                String dayInfo = (String) row.getField("dayInfo");
                String hourInfo = (String) row.getField("hourInfo");
```

编写Flink代码，对数据进行处理操作

```
            String seo_source = (String) row.getField("seo_source");
            String area = (String) row.getField("area");
            String origin_channel = (String) row.getField("origin_channel");
            Integer msg_count = (Integer) row.getField("msg_count");
            String from_url = (String) row.getField("from_url");
            pulsarTopicPojo.setData(id, sid, ip, session_id, create_time,yearInfo, monthInfo, dayInfo, hourInfo, seo_source, area, origin_channel,
msg_count, from_url);
            return pulsarTopicPojo;
        }
    });
```

- 8- 设置sink, 将数据写出到Pulsar: Pulsar Connector Flink Sink

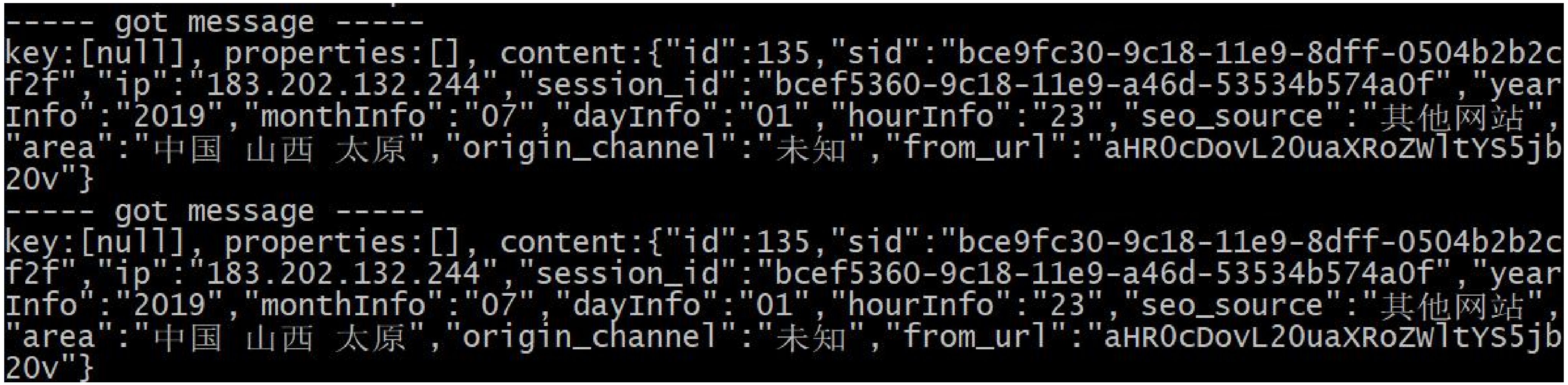
```
    //3.6: 设置sink 进行数据输出操作
    PulsarSerializationSchemaWrapper<PulsarTopicPojo> pulsarSerialization = new
PulsarSerializationSchemaWrapper.Builder<>(JsonSer.of(PulsarTopicPojo.class))
            .usePojoMode(PulsarTopicPojo.class, RecordSchemaType.JSON) .build();
    FlinkPulsarSink<PulsarTopicPojo> pulsarSink = new FlinkPulsarSink<>(
            "pulsar://node1:6650,node2:6650,node3:6650", "http://node1:8080,node2:8080,node3:8080",
            Optional.of("persistent://public/default/itcast_ems_tab"), new Properties(),
            pulsarSerialization
    );
    pulsarTopicDataStream.addSink(pulsarSink);
    //4. 执行启动
    env.execute("itcast_ems");
  }
}
```

测试数据是否可以正常写回到Pulsar

- 1- 启动一个消费者, 用于接收消息

```
cd /export/server/pulsar_2.8.1/
./bin/pulsar-client consume -s 'itcast_test01' -n 0 itcast_ems_tab
```

- 2- 启动Flink程序: 并通过mysql向两个表写入数据, 观察消费者是否可以正常输出数据



```
----- got message -----
key:[null], properties:[], content:{"id":135,"sid":"bce9fc30-9c18-11e9-8dff-0504b2b2c
f2f","ip":"183.202.132.244","session_id":"bcef5360-9c18-11e9-a46d-53534b574a0f","year
Info":"2019","monthInfo":"07","dayInfo":"01","hourInfo":"23","seo_source":"其他网站",
"area":"中国 山西 太原","origin_channel":"未知","from_url":"aHR0cDovL20uaXRoZWltYS5jb
20v"}
----- got message -----
key:[null], properties:[], content:{"id":135,"sid":"bce9fc30-9c18-11e9-8dff-0504b2b2c
f2f","ip":"183.202.132.244","session_id":"bcef5360-9c18-11e9-a46d-53534b574a0f","year
Info":"2019","monthInfo":"07","dayInfo":"01","hourInfo":"23","seo_source":"其他网站",
"area":"中国 山西 太原","origin_channel":"未知","from_url":"aHR0cDovL20uaXRoZWltYS5jb
20v"}
```

06 基于Flink将数据写入到HBase

编写Flink完成数据写入到Hbase操作，完成数据备份，便于后续进行即席查询和离线分析

HBase基本介绍

hbase是基于Google发布bigTable论文产生一款软件，是一款noSQL型数据，不支持SQL. 不支持join的操作，没有表关系，不支持事务(多行事务),hbase是基于 HDFS的采用java 语言编写

查询hbase数据一般有三种方案(主键(row key)查询，主键的范围检索,查询全部数据)

都是以字节类型存储，存储结构化和半结构化数据

hbase表的特点：大   面向列的存储方案 稀疏性

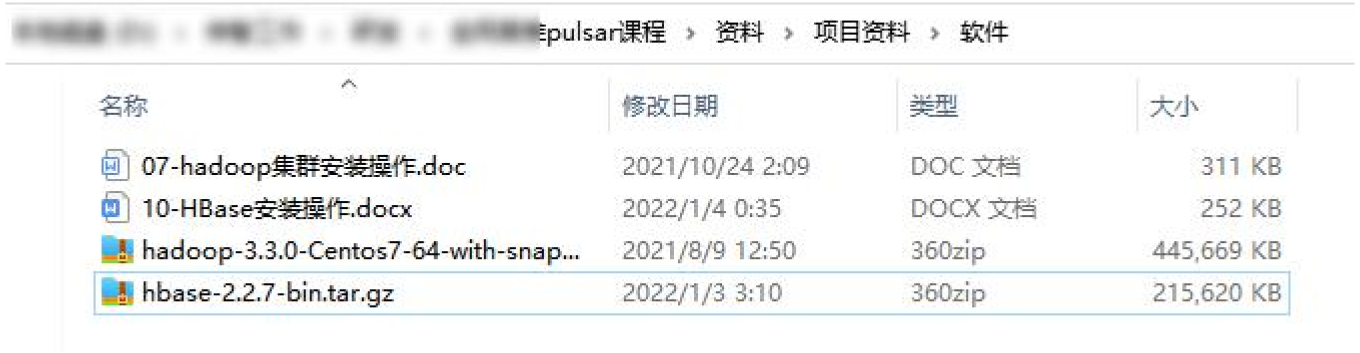
应用场景

- 1）需要进行随机读写的操作
- 2）数据量比较大
- 3）数据比较稀疏

编写Flink完成数据写入到Hbase操作，完成数据备份，便于后续进行即席查询和离线分析

HBase安装操作

本次安装的HBase为2.2.7,详细的安装手册大家可以参考资料，还需要大家注意,HBase的启动需要依赖于zookeeper和HDFS的，顾需要先安装 HADOOP与zookeeper



pulsar课程 › 资料 › 项目资料 › 软件

名称	修改日期	类型	大小
07-hadoop集群安装操作.doc	2021/10/24 2:09	DOC 文档	311 KB
10-HBase安装操作.docx	2022/1/4 0:35	DOCX 文档	252 KB
hadoop-3.3.0-Centos7-64-with-snap...	2021/8/9 12:50	360zip	445,669 KB
hbase-2.2.7-bin.tar.gz	2022/1/3 3:10	360zip	215,620 KB

编写Flink完成数据写入到Hbase操作，完成数据备份，便于后续进行即席查询和离线分析

- 1- 在Hbase中创建目标表

```
create 'itcast_h_ems, {NAME=>'f1',COMPRESSION=>'GZ'},{NUMREGIONS=>6, SPLITALGO=>'HexStringSplit'}
```

- 2- 编写Flink代码完成写入Hbase操作

```
// 将数据写入到Hbase中
public class ItcastFlinkToHbase {
    public static void main(String[] args) {
        //1. 创建Flink流式处理的核心环境类对象
        StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
        StreamTableEnvironment tableEnv = StreamTableEnvironment.create(env);
        //2. 添加source组件: 从Pulsar中读取数据
        Properties props = new Properties();
        props.setProperty("topic","persistent://public/default/itcast_ems_tab");
        FlinkPulsarSource<PulsarTopicPojo> pulsarSource = new FlinkPulsarSource<>(
                "pulsar://node1:6650,node2:6650,node3:6650",
                "http://node1:8080,node2:8080,node3:8080",
                JsonDeser.of(PulsarTopicPojo.class),
                props
        );
        DataStreamSource<PulsarTopicPojo> streamSource = env.addSource(pulsarSource);
```

编写Flink完成数据写入到Hbase操作，完成数据备份，便于后续进行即席查询和离线分析

```
        //3. 转换为Flink Table
        Schema schema = Schema.newBuilder()
                .column("id", DataTypes.INT()).column("sid", DataTypes.STRING())
                .column("ip", DataTypes.STRING()) .column("session_id", DataTypes.STRING())
                .column("create_time", DataTypes.STRING()).column("yearInfo", DataTypes.STRING())
                .column("monthInfo", DataTypes.STRING()).column("dayInfo", DataTypes.STRING())
                .column("hourInfo", DataTypes.STRING()).column("seo_source", DataTypes.STRING())
                .column("area", DataTypes.STRING()).column("origin_channel", DataTypes.STRING())
                .column("msg_count", DataTypes.INT()).column("from_url", DataTypes.STRING())
                .build();
        tableEnv.createTemporaryView("itcast_ems",streamSource,schema);
        //4, 定义Hbase的表
        String hTable = "CREATE  TABLE itcast_h_ems (" +
                " rowkey INT," +
                " f1 ROW<sid STRING,ip STRING,session_id STRING,create_time STRING,yearInfo STRING,monthInfo STRING,dayInfo STRING,hourInfo 
STRING,seo_source STRING,area STRING,origin_channel STRING,msg_count INT,from_url STRING>," +
                " PRIMARY KEY (rowkey) NOT ENFORCED" +
                ") WITH (" +
                " 'connector' = 'hbase-2.2'," +
                " 'table-name' = 'itcast_h_ems'," +
                " 'zookeeper.quorum' = 'node1:2181,node2:2181,node3:2181'" +
                ")";
        tableEnv.executeSql(hTable);
        tableEnv.executeSql("insert into itcast_h_ems select 
id,ROW(sid,ip,session_id,create_time,yearInfo,monthInfo,dayInfo,hourInfo,seo_source,area,origin_channel,msg_count,from_url) from itcast_ems");
    }
}
```

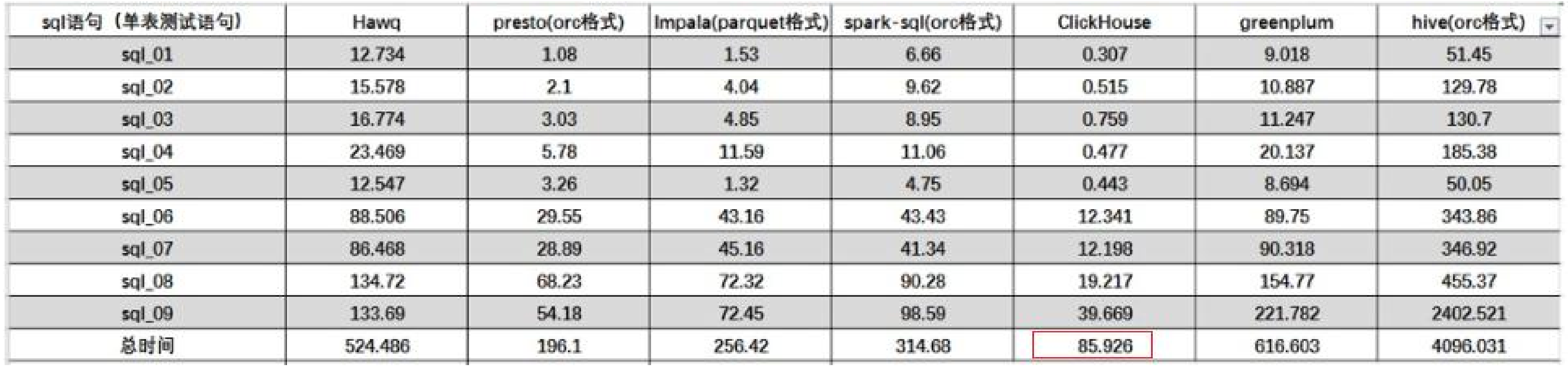
07

基于Flink将数据写入到ClickHouse

编写Flink完成数据写入到ClickHouse操作，后续基于CK完成指标统计操作

ClickHouse基本介绍

ClickHouse 是俄罗斯的Yandex于2016年开源的列式存储数据库（DBMS），使用C++语言编写，主要用于在线分析处理查询（OLAP），能够使用SQL查询实时生成分析数据报告。



sql语句（单表测试语句）	Hawq	presto(orc格式)	Impala(parquet格式)	spark-sql(orc格式)	ClickHouse	greenplum	hive(orc格式)
sql_01	12.734	1.08	1.53	6.66	0.307	9.018	51.45
sql_02	15.578	2.1	4.04	9.62	0.515	10.887	129.78
sql_03	16.774	3.03	4.85	8.95	0.759	11.247	130.7
sql_04	23.469	5.78	11.59	11.06	0.477	20.137	185.38
sql_05	12.547	3.26	1.32	4.75	0.443	8.694	50.05
sql_06	88.506	29.55	43.16	43.43	12.341	89.75	343.86
sql_07	86.468	28.89	45.16	41.34	12.198	90.318	346.92
sql_08	134.72	68.23	72.32	90.28	19.217	154.77	455.37
sql_09	133.69	54.18	72.45	98.59	39.669	221.782	2402.521
总时间	524.486	196.1	256.42	314.68	85.926	616.603	4096.031

结论：ClickHouse像很多OLAP数据库一样，单表查询速度由于关联查询，而且ClickHouse的两者差距更为明显。

编写Flink完成数据写入到ClickHouse操作，后续基于CK完成指标统计操作

ClickHouse安装步骤

本项目中,我们仅需要安装单机测试版本即可使用(node2安装)，在实际生产中，大家可以直接将分布式集群版本

- 1- 设置yum源

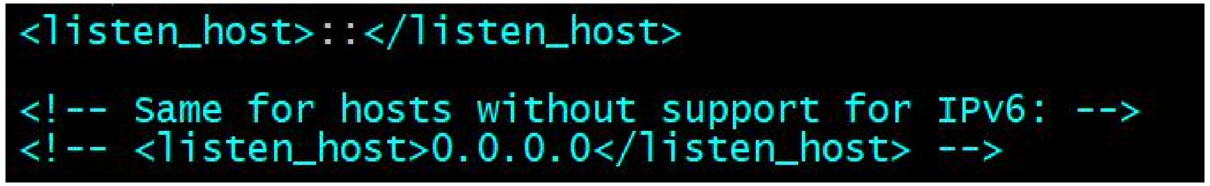
```
sudo yum install yum-utils
sudo rpm --import https://repo.clickhouse.com/CLICKHOUSE-KEY.GPG
sudo yum-config-manager --add-repo https://repo.clickhouse.com/rpm/stable/x86_64
```

- 2- 直接基于yum安装即可

```
sudo yum install clickhouse-server clickhouse-client
```

- 3- 修改配置文件

```
vim /etc/clickhouse-server/config.xml

修改178行: 打开这一行的注释
<listen_host>::</listen_host>
```



```
<listen_host>::</listen_host>

<!-- Same for hosts without support for IPv6: -->
<!-- <listen_host>0.0.0.0</listen_host> -->
```

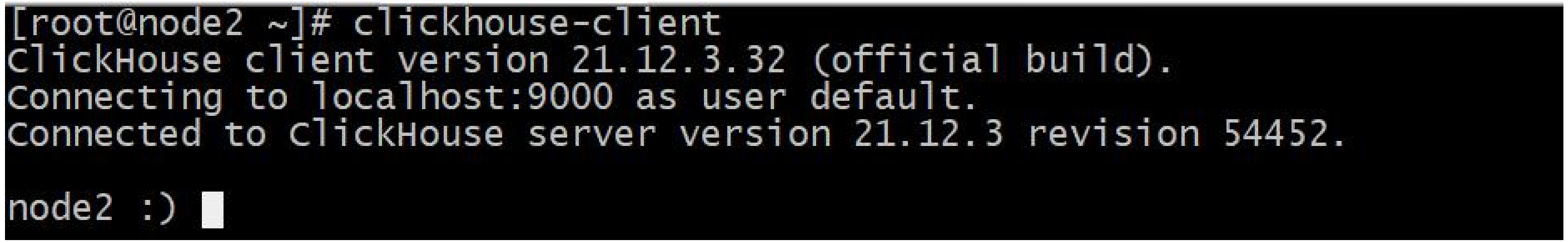
编写Flink完成数据写入到ClickHouse操作，后续基于CK完成指标统计操作

ClickHouse安装步骤

- 4- 启动clickhouse的server

```
启动服务
    systemctl start clickhouse-server

停止:
    systemctl stop clickhouse-server

重启
   systemctl restart clickhouse-server
```

- 5- 进入客户端



```
[root@node2 ~]# clickhouse-client
ClickHouse client version 21.12.3.32 (official build).
Connecting to localhost:9000 as user default.
Connected to ClickHouse server version 21.12.3 revision 54452.

node2 :) 
```

编写Flink完成数据写入到ClickHouse操作，后续基于CK完成指标统计操作

- 1- 在ClickHouse中创建目标表

```
create database itcast_ck;
use  itcast_ck;
create table itcast_ck.itcast_ck_ems(
   id int,
   sid varchar(128),
   ip varchar(128),
   create_time varchar(128),
   session_id varchar(128),
   yearInfo varchar(128),
   monthInfo varchar(128),
   dayInfo varchar(128),
   hourInfo varchar(128),
   seo_source varchar(128),
   area varchar(128),
   origin_channel varchar(128),
   msg_count int(128),
   from_url varchar(128),
   PRIMARY KEY (`id`)
) ENGINE=ReplacingMergeTree() ;
```

编写Flink完成数据写入到ClickHouse操作，后续基于CK完成指标统计操作

- 2- 编写Flink代码完成写入到CK操作

```
// 将数据写入到CK中
public class ItcastFlinkToCK {

    public static void main(String[] args) throws Exception {

        //1. 创建Flink流式处理的核心环境类对象
        StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
        //2. 添加source组件: 从Pulsar中读取数据
        Properties props = new Properties();
        props.setProperty("topic","persistent://public/default/itcast_ems_tab");
        FlinkPulsarSource<PulsarTopicPojo> pulsarSource = new FlinkPulsarSource<>(
                "pulsar://node1:6650,node2:6650,node3:6650",
                "http://node1:8080,node2:8080,node3:8080",
                JsonDeser.of(PulsarTopicPojo.class),
                props
        );
        DataStreamSource<PulsarTopicPojo> streamSource = env.addSource(pulsarSource);
```

编写Flink完成数据写入到ClickHouse操作，后续基于CK完成指标统计操作

```
    //3. 转换数据
    SingleOutputStreamOperator<Row> rowDS = streamSource.map(new MapFunction<PulsarTopicPojo, Row>() {
      @Override
      public Row map(PulsarTopicPojo pulsarTopicPojo) throws Exception {
        return Row.of(
                pulsarTopicPojo.getId(), pulsarTopicPojo.getSid(),pulsarTopicPojo.getIp(),
pulsarTopicPojo.getSession_id(),pulsarTopicPojo.getCreate_time(),pulsarTopicPojo.getYearInfo(),
pulsarTopicPojo.getMonthInfo(),pulsarTopicPojo.getDayInfo(), pulsarTopicPojo.getHourInfo(),pulsarTopicPojo.getSeo_source(),
pulsarTopicPojo.getArea(),pulsarTopicPojo.getOrigin_channel(), pulsarTopicPojo.getMsg_count(),pulsarTopicPojo.getFrom_url());
      }
    });
    //4.执行写入CK
    String insertSql = "INSERT INTO itcast_ck.itcast_ck_ems
(id,sid,ip,session_id,yearInfo,monthInfo,dayInfo,hourInfo,seo_source,area,origin_channel,msg_count,from_url) values(?,?,?,?,?,?,?,?,?,?,?,?,?)";
    JDBCAppendTableSink tableSink = JDBCAppendTableSink.builder()
        .setDrivername("ru.yandex.clickhouse.ClickHouseDriver")
        .setDBUrl("jdbc:clickhouse://node2:8123/itcast_ck").setQuery(insertSql).setBatchSize(1)
        .setParameterTypes(Types.INTEGER,Types.VARCHAR,Types.VARCHAR,Types.VARCHAR,Types.VARCHAR,Types.VARCHAR,
Types.VARCHAR,Types.VARCHAR,Types.VARCHAR,Types.VARCHAR,Types.VARCHAR,Types.INTEGER,Types.VARCHAR).build();
    tableSink.emitDataStream(rowDS);
    env.execute("itcast_to_ck");
  }
}
```

08 HBase对接Phoenix实现即席查询

Phoenix安装操作

Phoenix是属于apache旗下的一款基于hbase的工具， 此工具提供一种全新的方式来操作hbase中数据(SQL)，同时Phoenix对hbase进行大量的优化工作，能够让我们更加有效的操作hbase

整个安装操作，大家可以参考资料中安装手册，进行安装即可

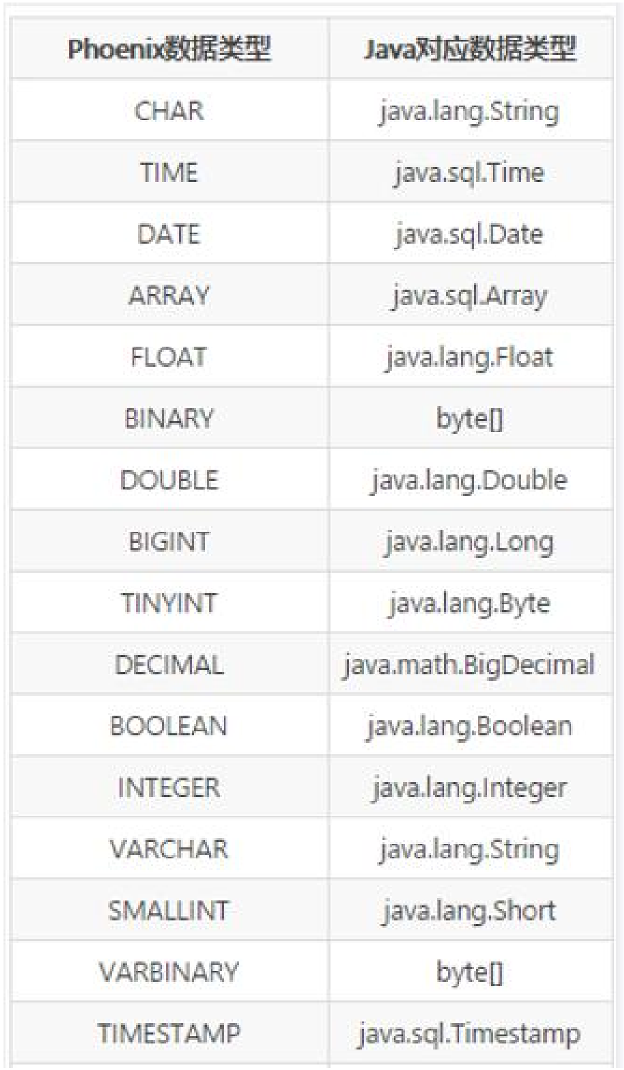
Phoenix对接HBase完成即席查询

- 1- 在Phoenix中创建表

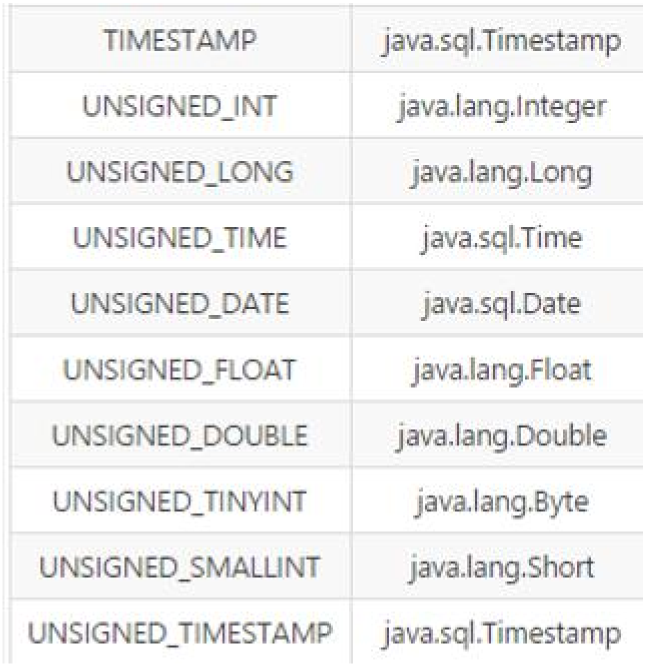
```
create view "itcast_h_ems" (
   "id" integer primary key,
   "f1"."sid" varchar,
   "f1"."ip" varchar,
   "f1"."create_time" varchar,
   "f1"."session_id" varchar,
   "f1"."yearInfo" varchar,
   "f1"."monthInfo" varchar,
   "f1"."dayInfo" varchar,
   "f1"."hourInfo" varchar,
   "f1"."seo_source" varchar,
   "f1"."area" varchar,
   "f1"."origin_channel" varchar,
   "f1"."msg_count" integer,
   "f1"."from_url" varchar
  );
```

Phoenix对接HBase完成即席查询

- 在Phoenix中类型说明



Phoenix数据类型	Java对应数据类型
CHAR	java.lang.String
TIME	java.sql.Time
DATE	java.sql.Date
ARRAY	java.sql.Array
FLOAT	java.lang.Float
BINARY	byte[]
DOUBLE	java.lang.Double
BIGINT	java.lang.Long
TINYINT	java.lang.Byte
DECIMAL	java.math.BigDecimal
BOOLEAN	java.lang.Boolean
INTEGER	java.lang.Integer
VARCHAR	java.lang.String
SMALLINT	java.lang.Short
VARBINARY	byte[]
TIMESTAMP	java.sql.Timestamp



TIMESTAMP	java.sql.Timestamp
UNSIGNED_INT	java.lang.Integer
UNSIGNED_LONG	java.lang.Long
UNSIGNED_TIME	java.sql.Time
UNSIGNED_DATE	java.sql.Date
UNSIGNED_FLOAT	java.lang.Float
UNSIGNED_DOUBLE	java.lang.Double
UNSIGNED_TINYINT	java.lang.Byte
UNSIGNED_SMALLINT	java.lang.Short
UNSIGNED_TIMESTAMP	java.sql.Timestamp

09 HBase对接HIVE完成离线统计分析

Stream Native

黑马程序员 www.itheima.com 传智教育旗下 高端IT教育品牌

PULSAR

HIVE基本介绍

hive是一款基于hadoop的数据仓库工具, 那么也就意味如果要启动hive, 必须先启动好hadoop, 最初由Facebook开发, 后期贡献给apache. 成为了apache的顶级项目

hive的作用 ， 可以将结构化的数据文件映射为一张数据库表，并提供类SQL查询功能。

hive本质上就是一款翻译软件, 主要用于将用户输入的SQL, 编译为MapReduce, 运行在yarn平台之上, 数据来源于HDFS

HIVE安装操作

HIVE安装操作大家可参考资料中安装手册即可

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- HIVE中创建表

```
create database itcast_edu;
create external table itcast_edu.itcast_h_ems_ods (
   id int,sid string,ip string,session_id string,create_time string,
   yearInfo string,monthInfo string,dayInfo string,hourInfo string,
   seo_source string,area string,origin_channel string,msg_count int,from_url string
)
stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties('hbase.columns.mapping'=':key#b,
f1:sid,
f1:ip,
f1:session_id,
f1:create_time,
f1:yearInfo,
f1:monthInfo,
f1:dayInfo,
f1:hourInfo,
f1:seo_source,
f1:area,
f1:origin_channel,
f1:msg_count#b,
f1:from_url') tblproperties('hbase.table.name'='itcast_h_ems');
```

HIVE集成HBase， 完成与HIVE对接，基于HIVE进行离线分析

业务需求:

1. 总访问客户量
2. 地区独立访客
3. 访客咨询率 ： 咨询率=发起咨询的人数/访问客户量
4. 客户访问量和访客咨询率趋势
5. 各时间段访问客户量
6. 各来源渠道/各搜索来源访问量
7. 活跃页面TOP10

维度:

日期: 截止到上一天的每小时,每天,每月, 每年

地区、来源渠道、搜索来源、受访页面

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

流程分析：

1- 构建DW层表，用于存储上一天及以前的数据

2- 构建RPT层，用于存储分析的结果

3- 对DW层的表进行统计分析，将统计分析的结果保存到RPT层

需求分析：

1- 抽取对接Phoenix的HIVE表上一天的数据到DW层，如果是第一次抽取，抽取截止上一天的全部

此表与对接Phoenix的HIVE表的字段保持一致即可

2- RPT层构建为两个表，分别用于保存访问量和咨询量

访问量的RPT表：total_visit,yearinfo,monthinfo,dayinfo,hourinfo,area,origin_channel,seo_source,from_url,time_type,group_type

咨询量的RPT表：total_consult,yearinfo,monthinfo,dayinfo,hourinfo,area,origin_channel,time_type,group_type

3- 在DW层基于各个维度完成统计操作

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 1-构建DW层表

```
create table itcast_edu.itcast_h_ems_dw(
   sid string,
   ip string,
   session_id string,
   hourInfo string,
   seo_source string,
   area string,
   origin_channel string,
   msg_count int,
   from_url string
)
COMMENT 'DW宽表'
PARTITIONED BY (yearinfo string, monthinfo STRING, dayinfo string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS ORC
LOCATION '/user/hive/warehouse/itcast_edu.db/itcast_h_ems_dw'
TBLPROPERTIES ('orc.compress'='SNAPPY');
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 2-构建RPT层表 -- 访问量RPT表

```
CREATE TABLE IF NOT EXISTS  itcast_edu.itcast_h_ems_visit_rpt (
  total_visit INT COMMENT '根据sid去重求count',
  area STRING COMMENT '区域信息',
  seo_source STRING COMMENT '搜索来源',
  origin_channel STRING COMMENT '来源渠道',
  hourinfo STRING COMMENT '创建时间，统计至小时',
  time_str STRING COMMENT '时间明细',
  from_url STRING comment '会话来源页面',
  groupType STRING COMMENT '产品属性类型：1.地区；2.搜索来源；3.来源渠道；4.会话来源页面；5.总访问量',
  time_type STRING COMMENT '时间聚合类型：1、按小时聚合；2、按天聚合；3、按月聚合；4、按年聚合；')
comment 'EMS访客日志RPT表'
PARTITIONED BY(yearinfo STRING,monthinfo STRING,dayinfo STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
stored as orc
location '/user/hive/warehouse/itcast_edu.db/itcast_h_ems_visit_rpt'
TBLPROPERTIES ('orc.compress'='SNAPPY');
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 3-构建RPT层表 -- 咨询量RPT表

```
CREATE TABLE IF NOT EXISTS itcast_edu.itcast_h_ems_consult_rpt (
  total_consult INT COMMENT '根据sid去重求count',
  area STRING COMMENT '区域信息',
  origin_channel STRING COMMENT '来源渠道',
  hourinfo STRING COMMENT '创建时间，统计至小时',
  time_str STRING COMMENT '时间明细',
  groupType STRING COMMENT '产品属性类型：1.地区；2.来源渠道',3.总咨询量
  time_type STRING COMMENT '时间聚合类型：1、按小时聚合；2、按天聚合；3、按月聚合；4、按年聚合；'
)
COMMENT '咨询量RPT宽表'
PARTITIONED BY (yearinfo string, monthinfo STRING, dayinfo string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS ORC
LOCATION '/user/hive/warehouse/itcast_dws.db/consult_dws'
TBLPROPERTIES ('orc.compress'='SNAPPY');
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 4-DW层:全量采集

```
--分区
SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=nonstrict;
--hive压缩
set hive.exec.compress.intermediate=true;
set hive.exec.compress.output=true;
--写入时压缩生效
set hive.exec.orc.compression.strategy=COMPRESSION;
-- 全量采集: 截止上一天的数据
insert into table itcast_edu.itcast_h_ems_dw partition(yearinfo,monthinfo,dayinfo)
select
    sid,
    ip,
    session_id,
    hourinfo,
    seo_source,
    area,
    origin_channel,
    msg_count,
    from_url ,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_ods where substr(create_time ,1,10) < '2021-01-03' ;
```

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

- 4-DW层:增量抽取上一天的数据

```
-- 全量采集: 截止上一天的数据
insert into table itcast_edu.itcast_h_ems_dw  partition(yearinfo,monthinfo,dayinfo)
select
    sid,
    ip,
    session_id,
    hourinfo,
    seo_source,
    area,
    origin_channel,
    msg_count,
    from_url ,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_ods where substr(create_time ,1,10) = ’2021-01-04’ ;
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期 统计总访问量

```
-- 小时 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    '-1' as seo_source,
    '-1' as origin_channel,
    hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time_str,
    '-1' as from_url,
    '5' as grouptype,
    '1' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,dayinfo,hourinfo
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

● 5-RPT层:基于 日期 统计总访问量

```
-- 天 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    '-1' as seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo) as time_str,
    '-1' as from_url,
    '5' as grouptype,
    '2' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,dayinfo
```

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

- 5-RPT层:基于 日期 统计总访问量

```
-- 月 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    '-1' as seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo) as time_str,
    '-1' as from_url,
    '5' as grouptype,
    '3' as time_type,
    yearinfo,
    monthinfo,
    '-1' as dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期 统计总访问量

```
-- 年 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    ‘-1’ as area,
    ‘-1’ as seo_source,
    ‘-1’ as origin_channel,
    ‘-1’ as hourinfo,
    yearinfo as time_str,
    ‘-1’ as from_url,
    ‘5’ as grouptype,
    ‘4’ as time_type,
    yearinfo,
    ‘-1’ as monthinfo,
    ‘-1’ as dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo
```

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

- 5-RPT层:基于 日期+地区 统计总访问量

```
-- 小时 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    area,
    '-1' as seo_source,
    '-1' as origin_channel,
    hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time_str,
    '-1' as from_url,
    '1' as grouptype,
    '1' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,dayinfo,hourinfo,area
```

HIVE集成HBase, 完成与HIVE对接, 基于HIVE进行离线分析

- 5-RPT层:基于 日期+地区 统计总访问量

```
-- 天 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    area,
    '-1' as seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo) as time_str,
    '-1' as from_url,
    '1' as grouptype,
    '2' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,dayinfo,area
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期+地区 统计总访问量

```
-- 月 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    area,
    '-1' as seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo) as time_str,
    '-1' as from_url,
    '1' as grouptype,
    '3' as time_type,
    yearinfo,
    monthinfo,
    '-1' as dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,area
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期+地区 统计总访问量

```
-- 年 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    area,
    ‘-1’ as seo_source,
    ‘-1’ as origin_channel,
    ‘-1’ as hourinfo,
    yearinfo as time_str,
    ‘-1’ as from_url,
    ‘1’ as grouptype,
    ‘4’ as time_type,
    yearinfo,
    ‘-1’ as monthinfo,
    ‘-1’ as dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,area
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期+搜索来源 统计总访问量

```
-- 小时 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    seo_source,
    '-1' as origin_channel,
    hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time_str,
    '-1' as from_url,
    '2' as grouptype,
    '1' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,dayinfo,hourinfo,seo_source
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期+搜索来源 统计总访问量

```
-- 天 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo) as time_str,
    '-1' as from_url,
    '2' as grouptype,
    '2' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,dayinfo,seo_source
```

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

- 5-RPT层:基于 日期+搜索来源 统计总访问量

```
-- 月 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo) as time_str,
    '-1' as from_url,
    '2' as grouptype,
    '3' as time_type,
    yearinfo,
    monthinfo,
    '-1' as dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,seo_source
```

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

- 5-RPT层:基于 日期+搜索来源 统计总访问量

```
-- 年 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    yearinfo as time_str,
    '-1' as from_url,
    '2' as grouptype,
    '4' as time_type,
    yearinfo,
    '-1' as monthinfo,
    '-1' as dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,seo_source
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期+来源渠道 统计总访问量

```
-- 小时 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    '-1' as seo_source,
    origin_channel,
    hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time_str,
    '-1' as from_url,
    '3' as grouptype,
    '1' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,dayinfo,hourinfo,origin_channel
```

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

- 5-RPT层:基于 日期+来源渠道 统计总访问量

```
-- 天 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    '-1' as seo_source,
    origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo) as time_str,
    '-1' as from_url,
    '3' as grouptype,
    '2' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,dayinfo,origin_channel
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期+来源渠道 统计总访问量

```
-- 月 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    ‘-1’ as area,
    ‘-1’ as seo_source,
    origin_channel,
    ‘-1’ as hourinfo,
    concat(yearinfo,’-’,monthinfo) as time_str,
    ‘-1’ as from_url,
    ‘3’ as grouptype,
    ‘3’ as time_type,
    yearinfo,
    monthinfo,
    ‘-1’ as dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,origin_channel
```

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

● 5-RPT层:基于 日期+来源渠道 统计总访问量

```
-- 年 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    '-1' as seo_source,
    origin_channel,
    '-1' as hourinfo,
    yearinfo as time_str,
    '-1' as from_url,
    '3' as grouptype,
    '4' as time_type,
    yearinfo,
    '-1' as monthinfo,
    '-1' as dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,origin_channel
```

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

- 5-RPT层:基于 日期+受访页面 统计总访问量

```
-- 小时 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    ‘-1’ as area,
    ‘-1’ as seo_source,
    ‘-1’ as origin_channel,
    hourinfo,
    concat(yearinfo,’-’,monthinfo,’-’,dayinfo,’ ’,hourinfo) as time_str,
    from_url,
    ‘4’ as grouptype,
    ‘1’ as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,dayinfo,hourinfo,from_url
```

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

● 5-RPT层:基于 日期+受访页面 统计总访问量

```
-- 天 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    '-1' as seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo) as time_str,
    from_url,
    '4' as grouptype,
    '2' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,dayinfo,from_url
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期+受访页面 统计总访问量

```
-- 月 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    '-1' as seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo) as time_str,
    from_url,
    '4' as grouptype,
    '3' as time_type,
    yearinfo,
    monthinfo,
    '-1' as dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,monthinfo,from_url
```

HIVE集成HBase， 完成与HIVE对接，基于HIVE进行离线分析

- 5-RPT层:基于 日期+受访页面 统计总访问量

```
-- 年 统计
insert into table itcast_edu.itcast_h_ems_visit_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
    '-1' as area,
    '-1' as seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    yearinfo as time_str,
    from_url,
    '4' as grouptype,
    '4' as time_type,
    yearinfo,
    '-1' as monthinfo,
    '-1' as dayinfo
from itcast_edu.itcast_h_ems_dw group by yearinfo,from_url
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 6-RPT层:基于 日期 统计咨询量

```
-- 小时 统计
insert into table itcast_edu.itcast_h_ems_consult_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_consult,
    ‘-1’ as area,
    ‘-1’ as seo_source,
    ‘-1’ as origin_channel,
    hourinfo,
    concat(yearinfo,’-’,monthinfo,’-’,dayinfo,’ ’,hourinfo) as time_str,
    ‘-1’ as from_url,
    ‘3’ as grouptype,
    ‘1’ as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw where msg_count>0  group by yearinfo,monthinfo,dayinfo,hourinfo
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 6-RPT层:基于 日期 统计咨询量

```
-- 天 统计
insert into table itcast_edu.itcast_h_ems_consult_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_consult,
    '-1' as area,
    '-1' as seo_source,
    '-1' as origin_channel,
     '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo) as time_str,
    '-1' as from_url,
    '3' as grouptype,
    '2' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw where msg_count>0  group by yearinfo,monthinfo,dayinfo
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 6-RPT层:基于 日期 统计咨询量

```
-- 月 统计
insert into table itcast_edu.itcast_h_ems_consult_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_consult,
    ‘-1’ as area,
    ‘-1’ as seo_source,
    ‘-1’ as origin_channel,
     ‘-1’ as hourinfo,
    concat(yearinfo,’-’,monthinfo) as time_str,
    ‘-1’ as from_url,
    ‘3’ as grouptype,
    ‘3’ as time_type,
    yearinfo,
    monthinfo,
    ‘-1’ as dayinfo
from itcast_edu.itcast_h_ems_dw where msg_count>0  group by yearinfo,monthinfo
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 6-RPT层:基于 日期 统计咨询量

```
-- 年 统计
insert into table itcast_edu.itcast_h_ems_consult_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_consult,
    '-1' as area,
    '-1' as seo_source,
    '-1' as origin_channel,
     '-1' as hourinfo,
    concat(yearinfo) as time_str,
    '-1' as from_url,
    '3' as grouptype,
    '4' as time_type,
    yearinfo,
    '-1' as monthinfo,
    '-1' as dayinfo
from itcast_edu.itcast_h_ems_dw where msg_count>0  group by yearinfo
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期+地区 统计总咨询量

```
-- 小时 统计
insert into table itcast_edu.itcast_h_ems_consult_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_consult,
    area,
    '-1' as seo_source,
    '-1' as origin_channel,
    hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time_str,
    '-1' as from_url,
    '1' as grouptype,
    '1' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw where msg_count>0  group by yearinfo,monthinfo,dayinfo,hourinfo,area
```

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

- 5-RPT层:基于 日期+地区 统计总咨询量

```
-- 天 统计
insert into table itcast_edu.itcast_h_ems_consult_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_consult,
    area,
    '-1' as seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo) as time_str,
    '-1' as from_url,
    '1' as grouptype,
    '2' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw where msg_count>0  group by yearinfo,monthinfo,dayinfo,area
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期+地区 统计总咨询量

```
-- 月 统计
insert into table itcast_edu.itcast_h_ems_consult_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_consult,
    area,
    '-1' as seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo) as time_str,
    '-1' as from_url,
    '1' as grouptype,
    '3' as time_type,
    yearinfo,
    monthinfo,
    '-1' as dayinfo
from itcast_edu.itcast_h_ems_dw where msg_count>0  group by yearinfo,monthinfo,area
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期+地区 统计总咨询量

```
-- 年 统计
insert into table itcast_edu.itcast_h_ems_consult_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_consult,
    area,
    '-1' as seo_source,
    '-1' as origin_channel,
    '-1' as hourinfo,
    concat(yearinfo) as time_str,
    '-1' as from_url,
    '1' as grouptype,
    '4' as time_type,
    yearinfo,
   '-1' as monthinfo,
    '-1' as dayinfo
from itcast_edu.itcast_h_ems_dw where msg_count>0  group by yearinfo,area
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期+来源渠道 统计总咨询量

```
-- 小时 统计
insert into table itcast_edu.itcast_h_ems_consult_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_consult,
    '-1' as area,
    origin_channel,
    hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo,' ',hourinfo) as time_str,
    '2' as grouptype,
    '1' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw where msg_count>0  group by yearinfo,monthinfo,dayinfo,hourinfo,origin_channel
```

HIVE集成HBase， 完成与HIVE对接， 基于HIVE进行离线分析

- 5-RPT层:基于 日期+来源渠道 统计总咨询量

```
-- 天 统计
insert into table itcast_edu.itcast_h_ems_consult_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_consult,
    '-1' as area,
    origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo,'-',dayinfo) as time_str,
    '2' as grouptype,
    '2' as time_type,
    yearinfo,
    monthinfo,
    dayinfo
from itcast_edu.itcast_h_ems_dw where msg_count>0  group by yearinfo,monthinfo,dayinfo,origin_channel
```

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

- 5-RPT层:基于 日期+来源渠道 统计总咨询量

```
-- 月 统计
insert into table itcast_edu.itcast_h_ems_consult_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_consult,
    '-1' as area,
    origin_channel,
    '-1' as hourinfo,
    concat(yearinfo,'-',monthinfo) as time_str,
    '2' as grouptype,
    '3' as time_type,
    yearinfo,
    monthinfo,
    '-1' as dayinfo
from itcast_edu.itcast_h_ems_dw where msg_count>0  group by yearinfo,monthinfo,origin_channel
```

HIVE集成HBase，完成与HIVE对接，基于HIVE进行离线分析

- 5-RPT层:基于 日期+来源渠道 统计总咨询量

```
-- 年 统计
insert into table itcast_edu.itcast_h_ems_consult_rpt partition(yearinfo,monthinfo,dayinfo)
select
    round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_consult,
    '-1' as area,
    origin_channel,
    '-1' as hourinfo,
    concat(yearinfo) as time_str,
    '2' as grouptype,
    '4' as time_type,
    yearinfo,
    '-1' asmonthinfo,
    '-1' as dayinfo
from itcast_edu.itcast_h_ems_dw where msg_count>0  group by yearinfo,origin_channel
```

10 基于clickhouse完成实时数仓分析

基于clickhouse完成实时数仓分析

业务需求:

1. 总访问量
2. 总咨询量
3. 咨询率

维度:

日期: 当天,小时

基于clickhouse完成实时数仓分析

- 1- 统计当天总访问量、总咨询量、咨询率

```
with t1 as (select \
yearInfo,monthInfo,dayInfo,
round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) / 3 ,2) as total_visit, \
round((count(distinct if( msg_count >0,sid,null)) + count(distinct if( msg_count >0,session_id,null)) + count(distinct if( msg_count
>0,ip,null))) / 3 ,2) as total_consult \
from itcast_ck.itcast_ck_ems
where substr(create_time,1,10) = '2022-01-06') \
select total_visit,total_consult ,  round(total_consult/total_visit *100,2)  from t1;
```

- 2- 统计当天，每个小时的总访问量、总咨询量、咨询率

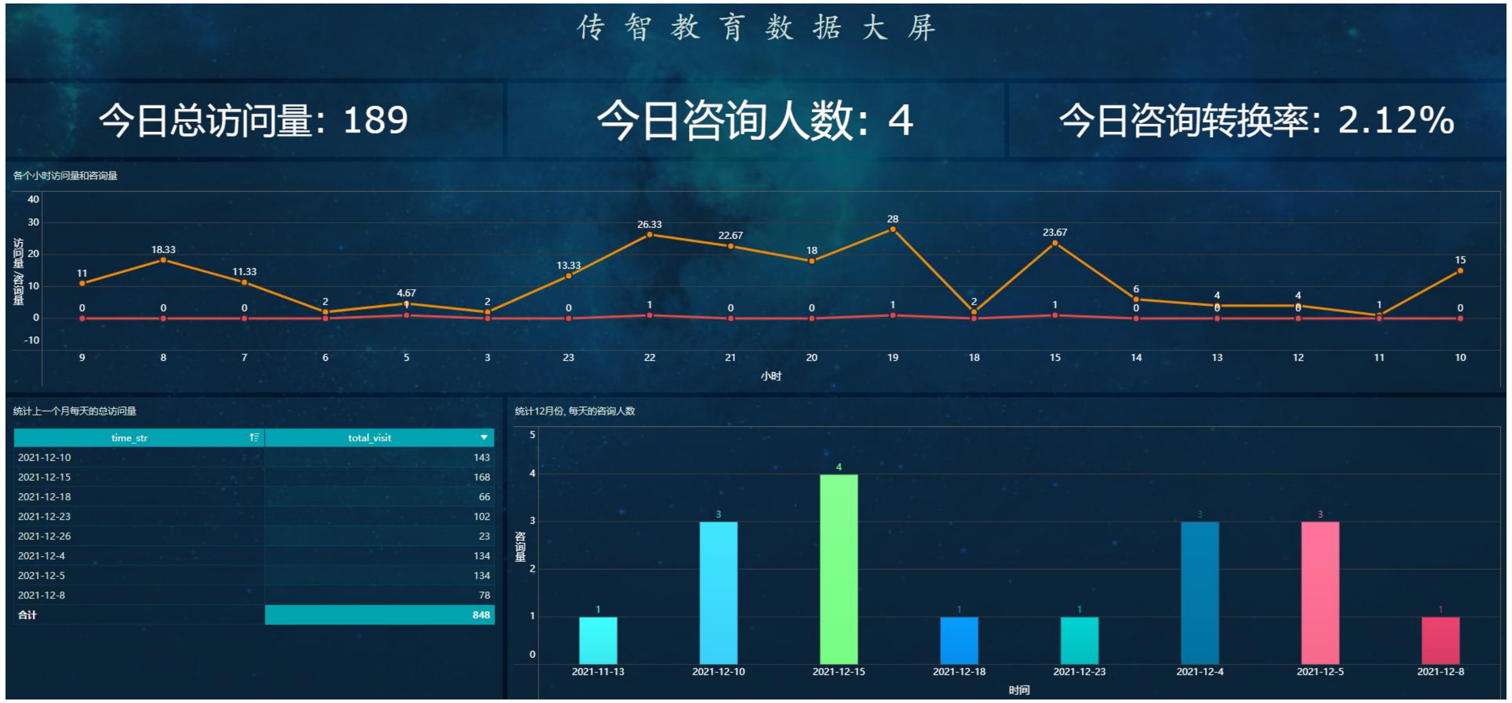
```
with t1 as (
select
        yearinfo,monthinfo,dayinfo,hourinfo
        round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) /3,2) as total_visit,
        round((count(distinct if(msg_count >0 ,sid,null)) + count(distinct if(msg_count >0 ,session_id,null)) + count(distinct
if(msg_count >0 ,ip,null))) /3,2) as total_consult
from itcast_ck.itcast_ck_ems where substr(create_time,1,10) = date(NOW())
group by yearinfo,monthinfo,dayinfo,hourinfo)
select yearinfo, total_visit, total_consult, round(total_consult/total_visit*100,2) from t1
```

11 基于FineBI实现实时报表

基于FineBI实现实时报表

本次我们主要通过帆软公司提供FineBI实现整个实时图表的实现操作，帆软也是国内比较大型一家专门做商业智能BI公司

FineBI如何安装以及如何激活，大家可以直接参考资料提供的安装文档即可，或者也可以直接到帆软官方进行下载安装即可，本次我们就直接使用FineBI即可

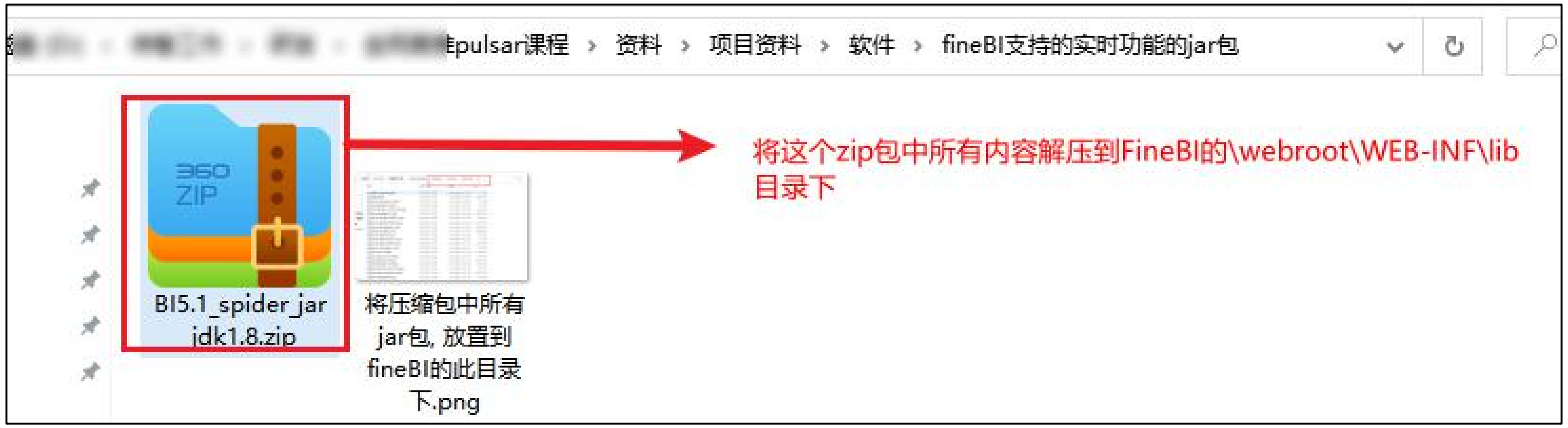


统计上一个月每天的总访问量

time_str	total_visit
2021-12-10	143
2021-12-15	168
2021-12-18	66
2021-12-23	102
2021-12-26	23
2021-12-4	134
2021-12-5	134
2021-12-8	78
合计	848

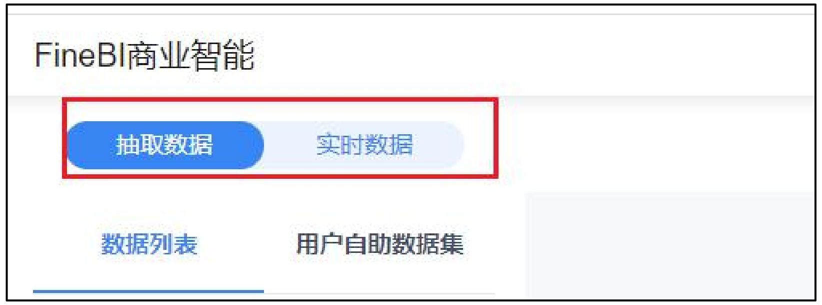
FineBI集成实时功能组件

- 1- 关闭FineBI服务
- 2- 将资料中提供的实时组件的包放置到fineBI的\webroot\WEB-INF\lib目录下


pulsar课程 › 资料 › 项目资料 › 软件 › fineBI支持的实时功能的jar包
BI5.1_spider_jar_jdk1.8.zip
将压缩包中所有jar包, 放置到fineBI的此目录下.png

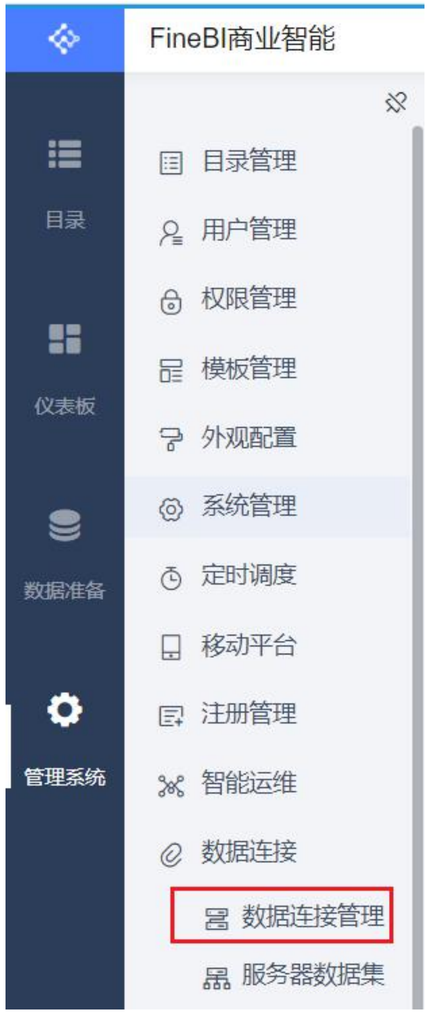
将这个zip包中所有内容解压到FineBI的\webroot\WEB-INF\lib目录下

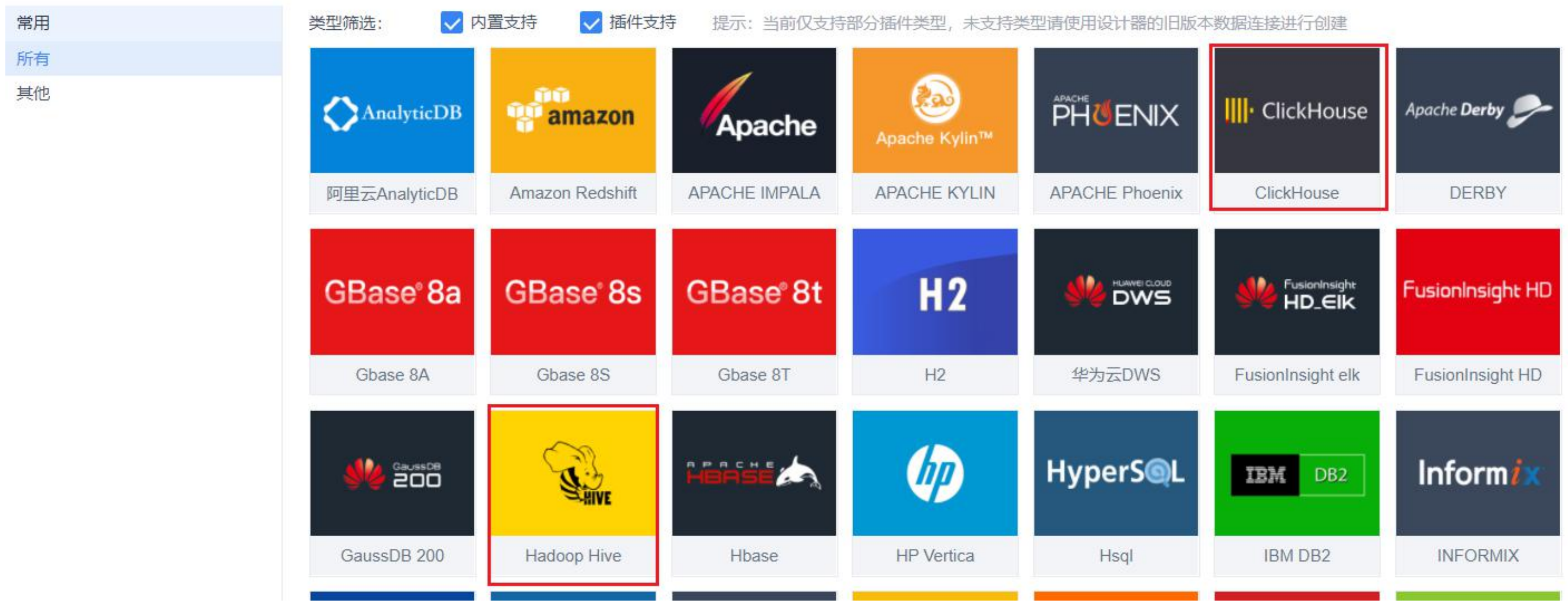
- 3- 启动FineBI即可：在数据准备窗口中查看



FineBI集成数据源准备工作

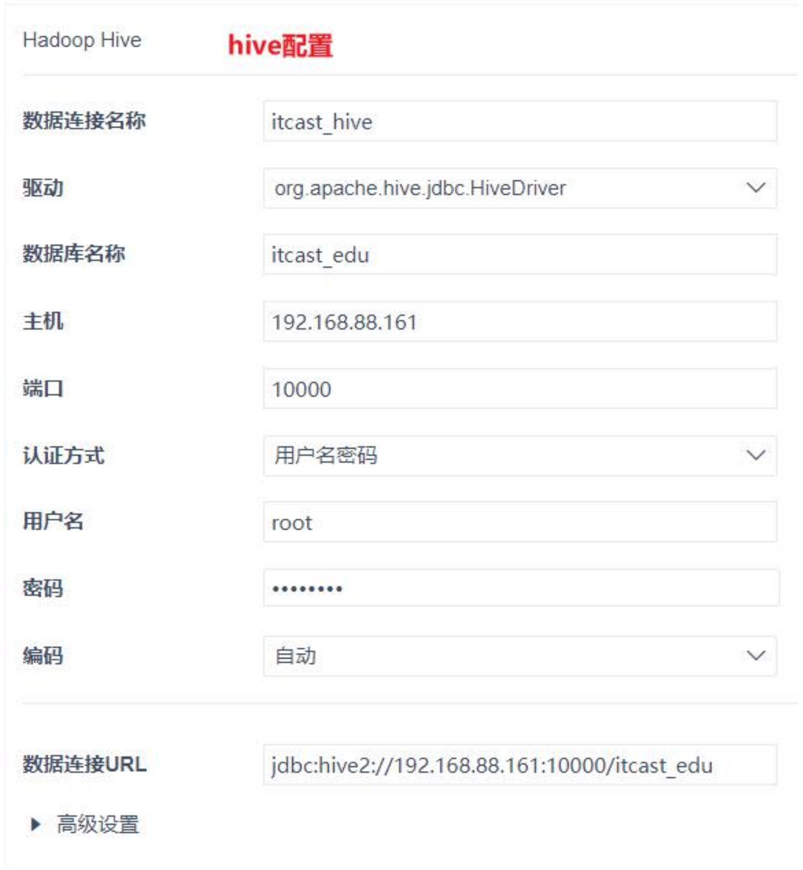
● 1- 基于FineBi连接ck和hive





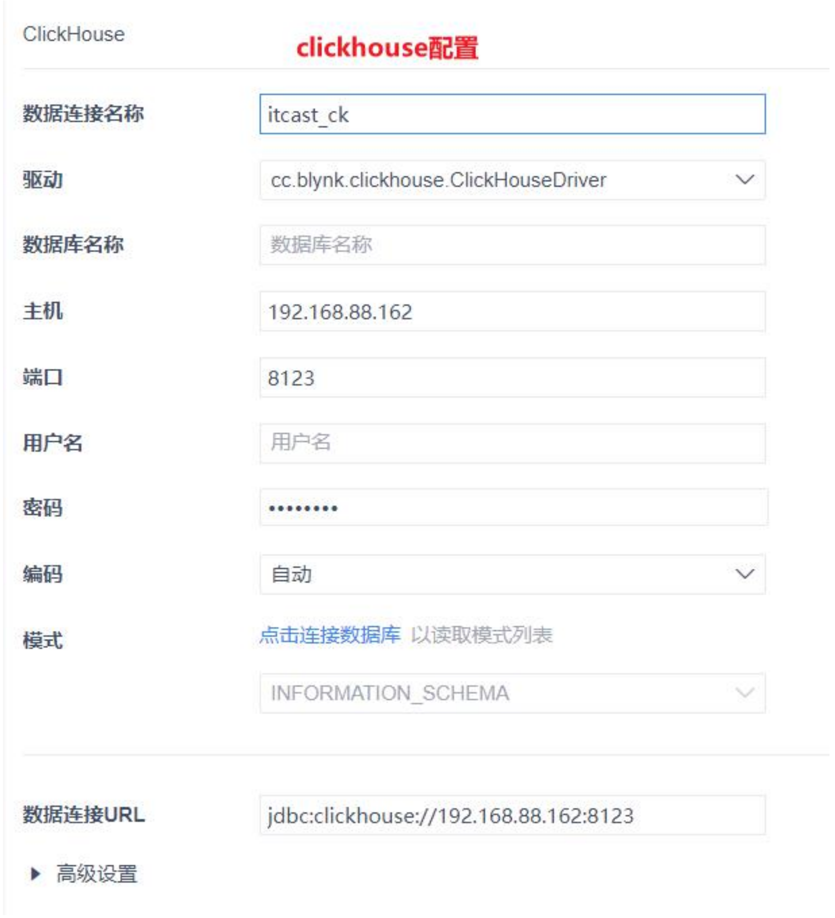
FineBI集成数据源准备工作

- 1- 基于FineBi连接ck和hive


Hadoop Hive hive配置

数据连接名称	itcast_hive
驱动	org.apache.hive.jdbc.HiveDriver
数据库名称	itcast_edu
主机	192.168.88.161
端口	10000
认证方式	用户名密码
用户名	root
密码	••••••••
编码	自动
数据连接URL	jdbc:hive2://192.168.88.161:10000/itcast_edu

▸ 高级设置

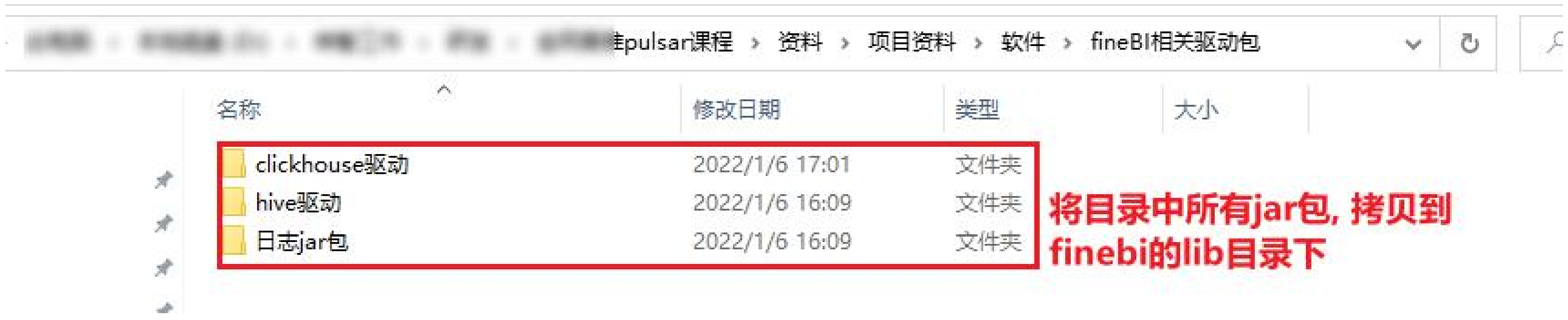

ClickHouse clickhouse配置

数据连接名称	itcast_ck
驱动	cc.blynk.clickhouse.ClickHouseDriver
数据库名称	数据库名称
主机	192.168.88.162
端口	8123
用户名	用户名
密码	••••••••
编码	自动
模式	点击连接数据库 以读取模式列表
	INFORMATION_SCHEMA
数据连接URL	jdbc:clickhouse://192.168.88.162:8123

▸ 高级设置

FineBI集成数据源准备工作

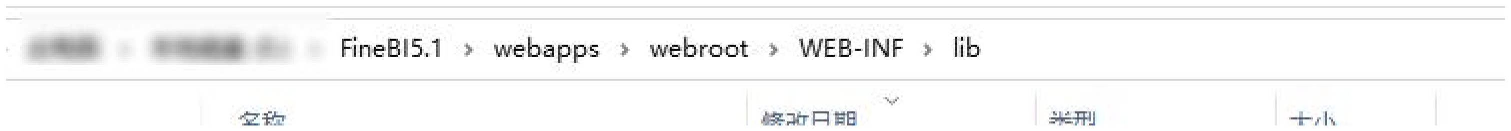
- 2- 驱动说明： 默认FineBi并没有hive驱动和ck的驱动， 需要下载驱动， 并放置到 finebi的lib目录下



[redacted] pulsar课程 › 资料 › 项目资料 › 软件 › fineBI相关驱动包

名称	修改日期	类型	大小
clickhouse驱动	2022/1/6 17:01	文件夹	
hive驱动	2022/1/6 16:09	文件夹	
日志jar包	2022/1/6 16:09	文件夹	

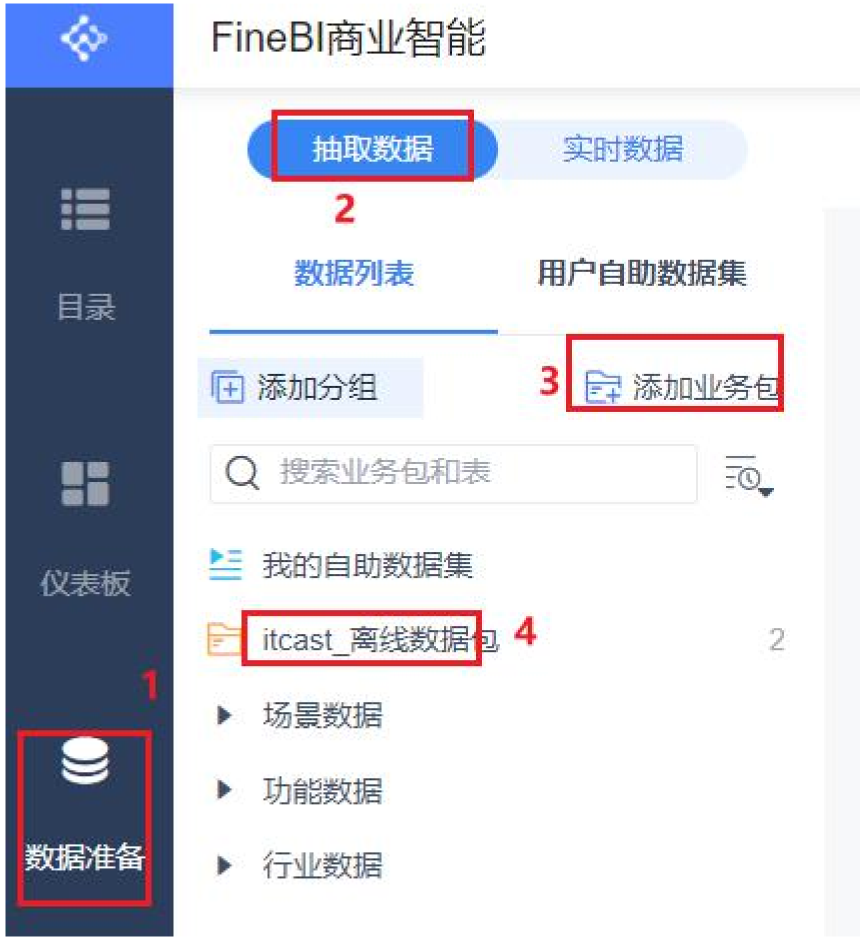
将目录中所有jar包，拷贝到 finebi的lib目录下

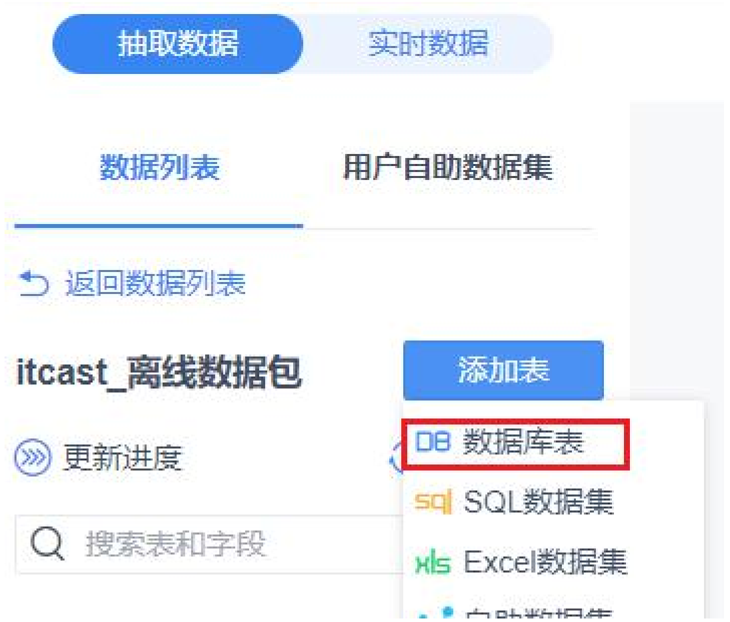


[redacted] › FineBI5.1 › webapps › webroot › WEB-INF › lib

FineBI集成数据源准备工作

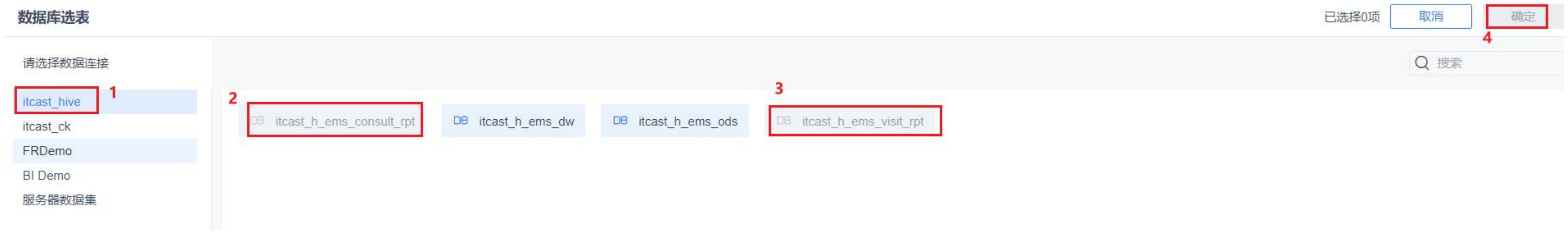
- 3- 数据准备：离线数据集准备





FineBI集成数据源准备工作

● 3- 数据准备：离线数据集准备



数据库选表

已选择0项 取消 确定 4

请选择数据连接

itcast_hive 1

itcast_ck

FRDemo

BI Demo

服务器数据集

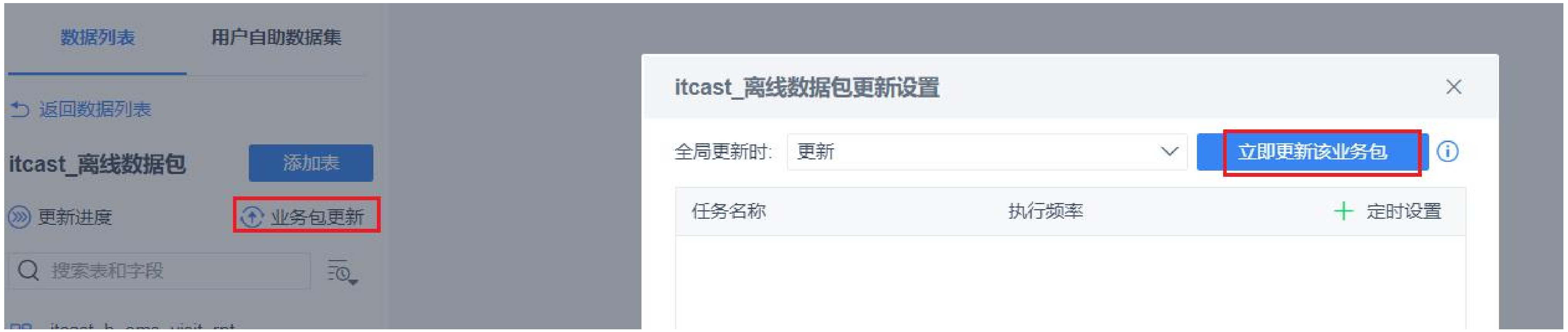
搜索

2 itcast_h_ems_consult_rpt

itcast_h_ems_dw

itcast_h_ems_ods

3 itcast_h_ems_visit_rpt



数据列表 用户自助数据集

返回数据列表

itcast_离线数据包 添加表

更新进度 业务包更新

搜索表和字段

itcast_离线数据包更新设置

全局更新时：更新

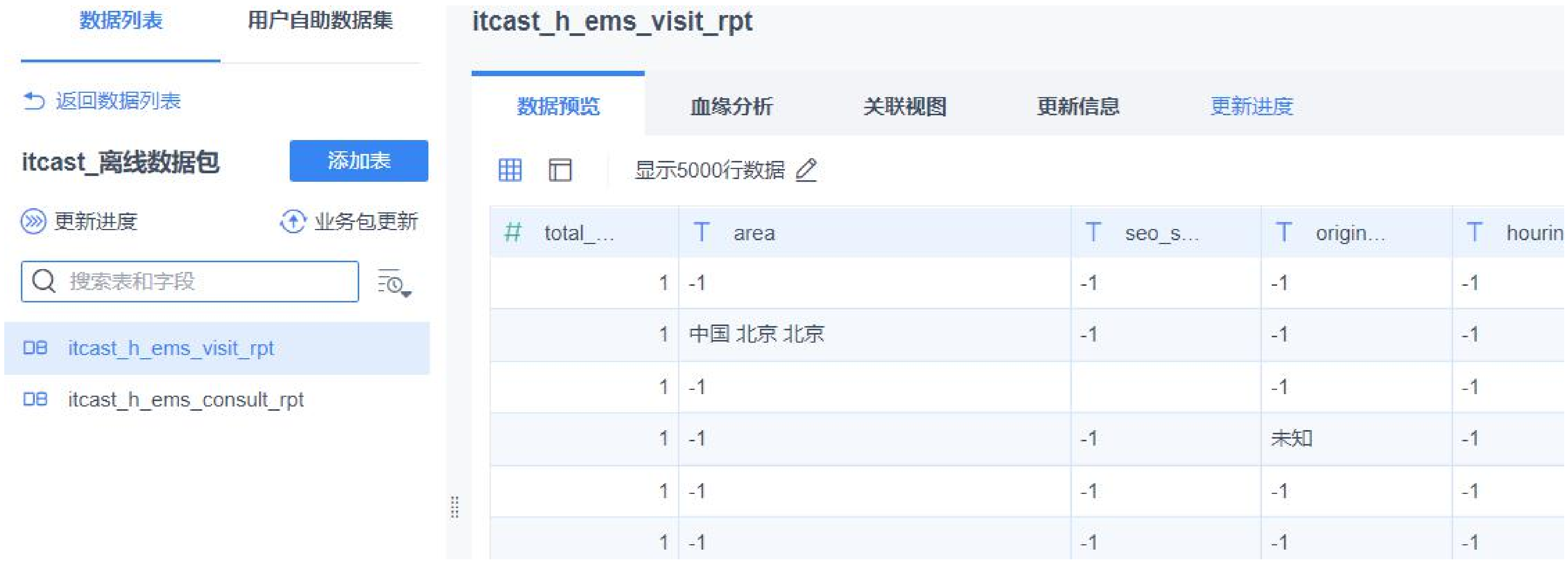
立即更新该业务包

任务名称	执行频率	定时设置

FineBI集成数据源准备工作

● 3- 数据准备：离线数据集准备

在连接hive的时候，如果遇到字段是乱码的，可以直接手动重命名调整即可

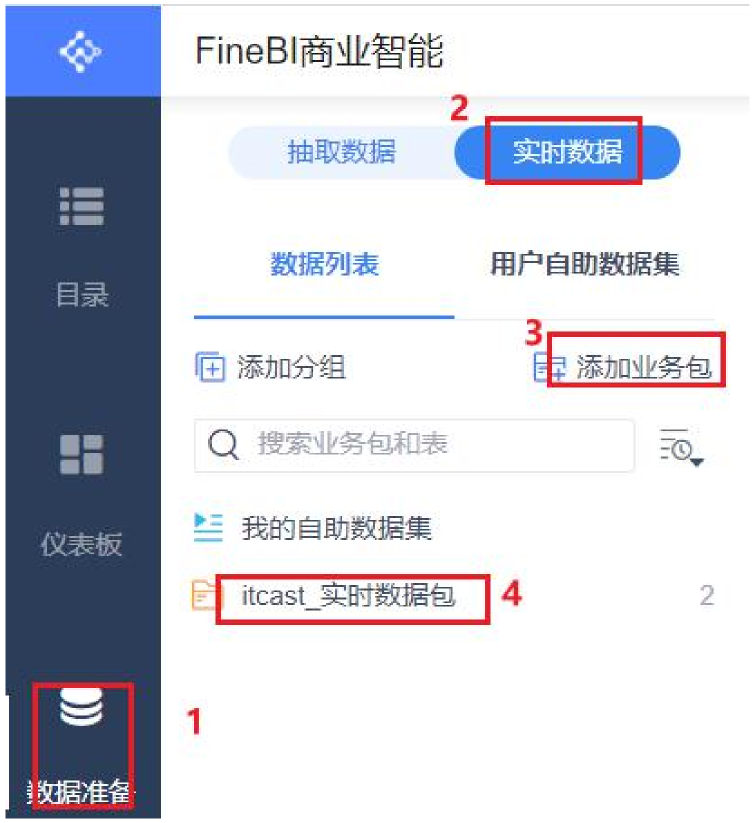


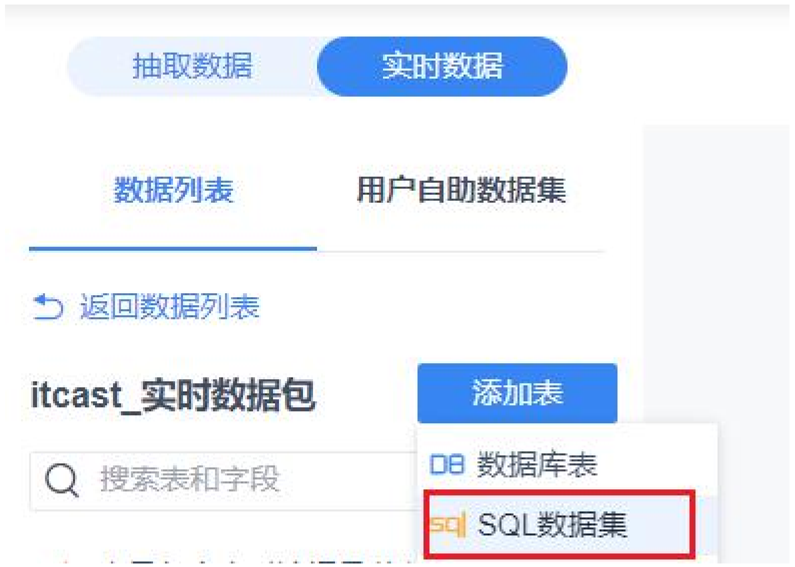
itcast_h_ems_visit_rpt

# total_...	T area	T seo_s...	T origin...	T hourin
1	-1	-1	-1	-1
1	中国 北京 北京	-1	-1	-1
1	-1		-1	-1
1	-1	-1	未知	-1
1	-1	-1	-1	-1
1	-1	-1	-1	-1

FineBI集成数据源准备工作

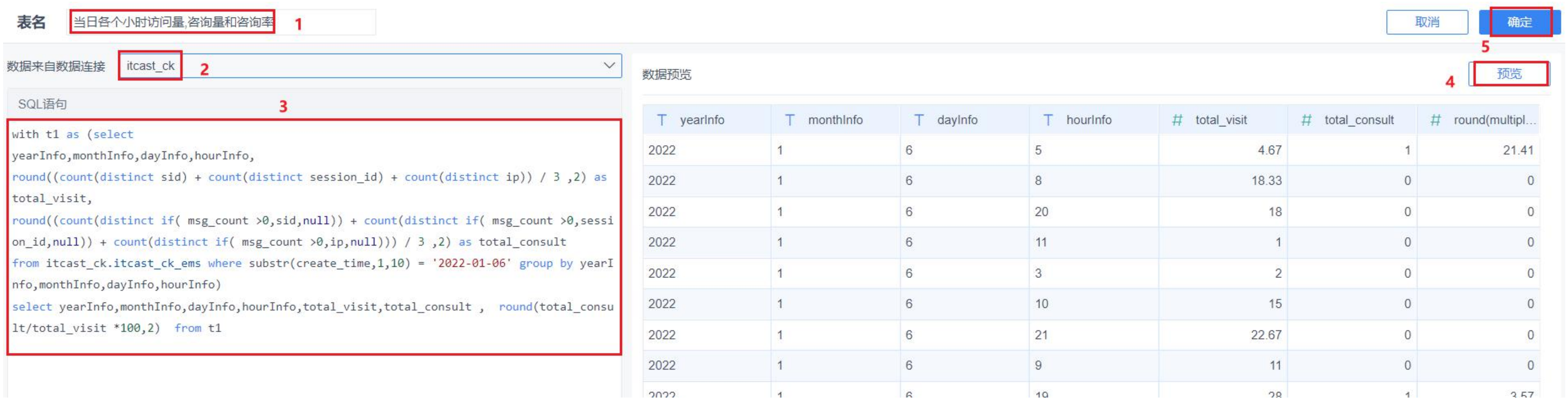
- 4- 数据准备：实时数据集准备





FineBI集成数据源准备工作

- 4- 数据准备：实时数据集准备

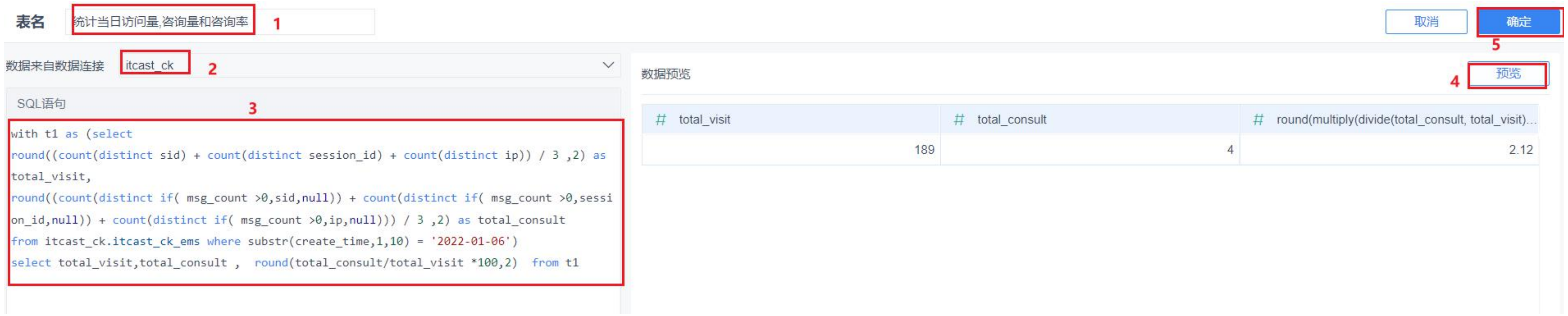


```
with t1 as (select
yearInfo,monthInfo,dayInfo,hourInfo,
round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) / 3 ,2) as
total_visit,
round((count(distinct if( msg_count >0,sid,null)) + count(distinct if( msg_count >0,sessi
on_id,null)) + count(distinct if( msg_count >0,ip,null))) / 3 ,2) as total_consult
from itcast_ck.itcast_ck_ems where substr(create_time,1,10) = '2022-01-06' group by yearI
nfo,monthInfo,dayInfo,hourInfo)
select yearInfo,monthInfo,dayInfo,hourInfo,total_visit,total_consult ,  round(total_consu
lt/total_visit *100,2)  from t1
```

T yearInfo	T monthInfo	T dayInfo	T hourInfo	# total_visit	# total_consult	# round(multipl...
2022	1	6	5	4.67	1	21.41
2022	1	6	8	18.33	0	0
2022	1	6	20	18	0	0
2022	1	6	11	1	0	0
2022	1	6	3	2	0	0
2022	1	6	10	15	0	0
2022	1	6	21	22.67	0	0
2022	1	6	9	11	0	0
2022	1	6	19	28	1	3.57

FineBI集成数据源准备工作

● 4- 数据准备：实时数据集准备

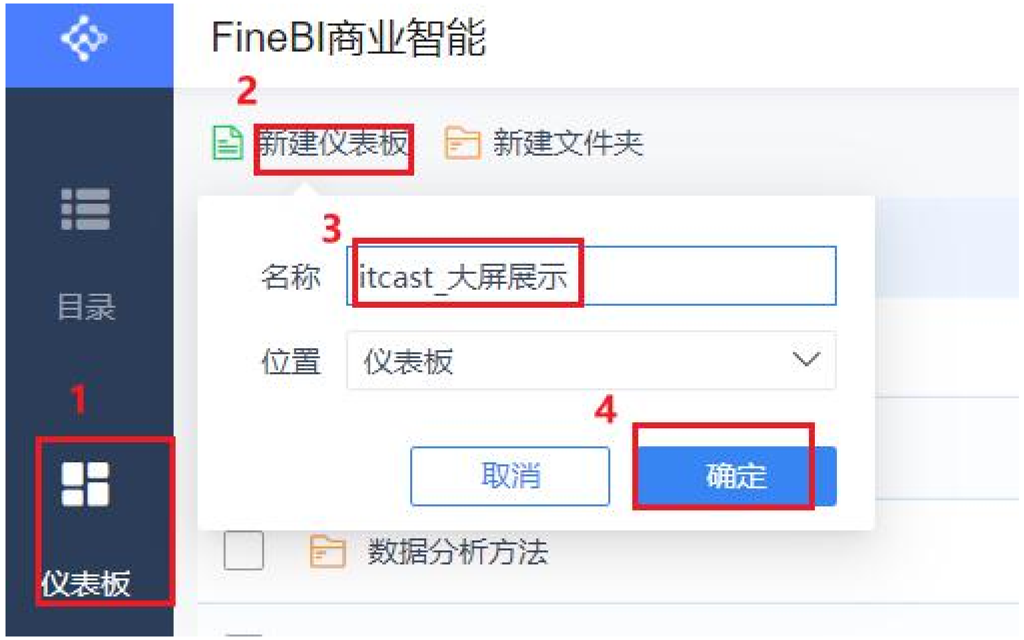


```
with t1 as (select
round((count(distinct sid) + count(distinct session_id) + count(distinct ip)) / 3 ,2) as
total_visit,
round((count(distinct if( msg_count >0,sid,null)) + count(distinct if( msg_count >0,sessi
on_id,null)) + count(distinct if( msg_count >0,ip,null))) / 3 ,2) as total_consult
from itcast_ck.itcast_ck_ems where substr(create_time,1,10) = '2022-01-06')
select total_visit,total_consult ,  round(total_consult/total_visit *100,2)  from t1
```

# total_visit	# total_consult	# round(multiply(divide(total_consult, total_visit)...
189	4	2.12

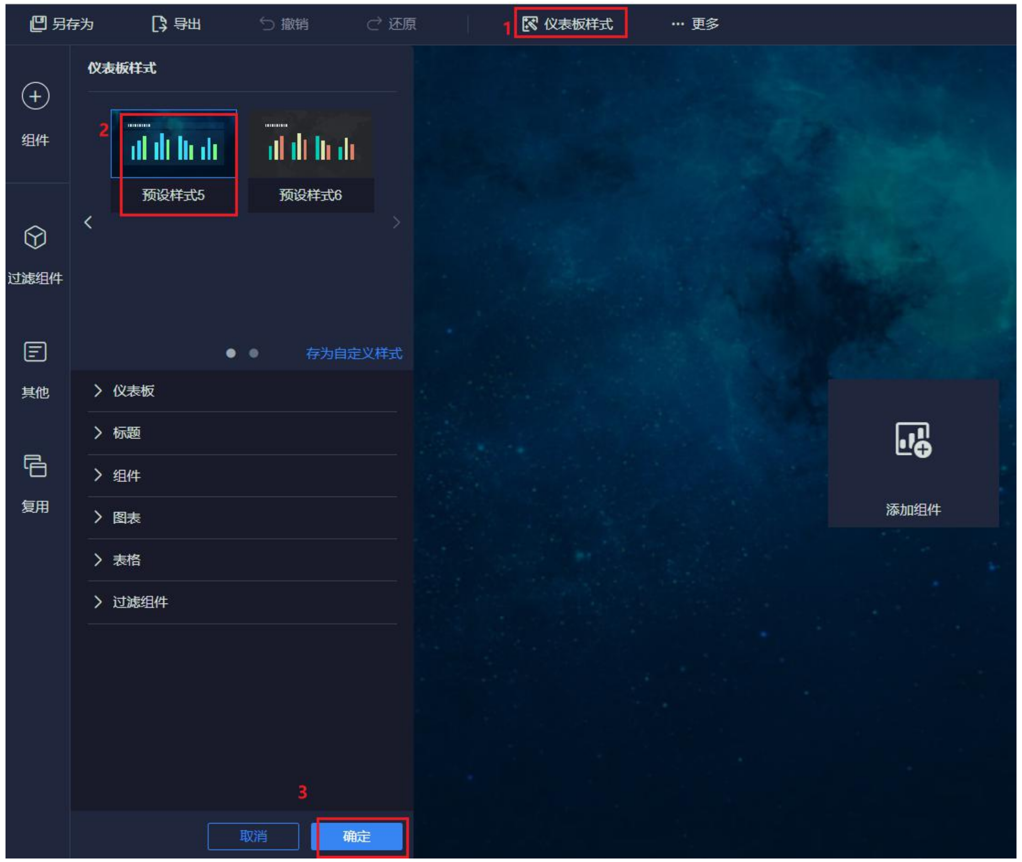
FineBI实现实时大屏

- 1- 创建仪表盘



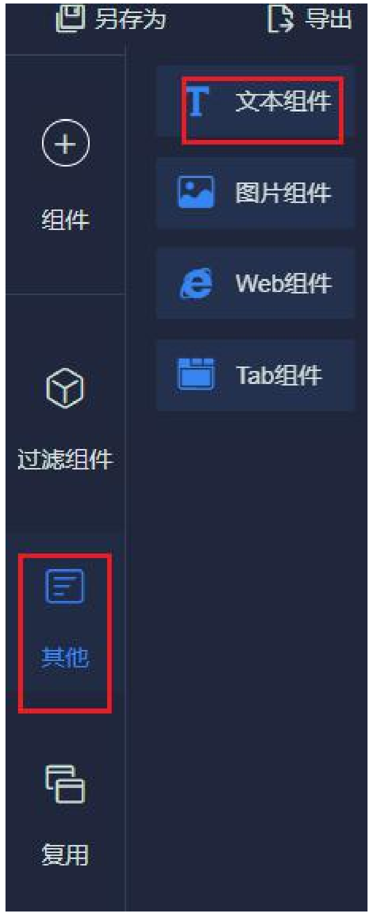
FineBI实现实时大屏

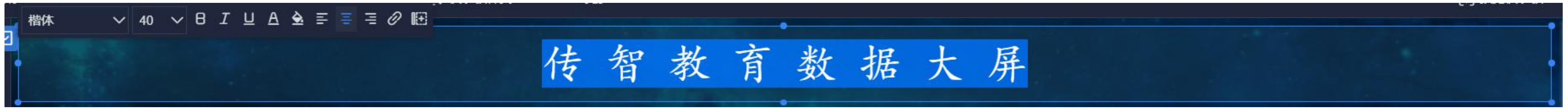
- 2- 修改仪表盘样式



FineBI实现实时大屏

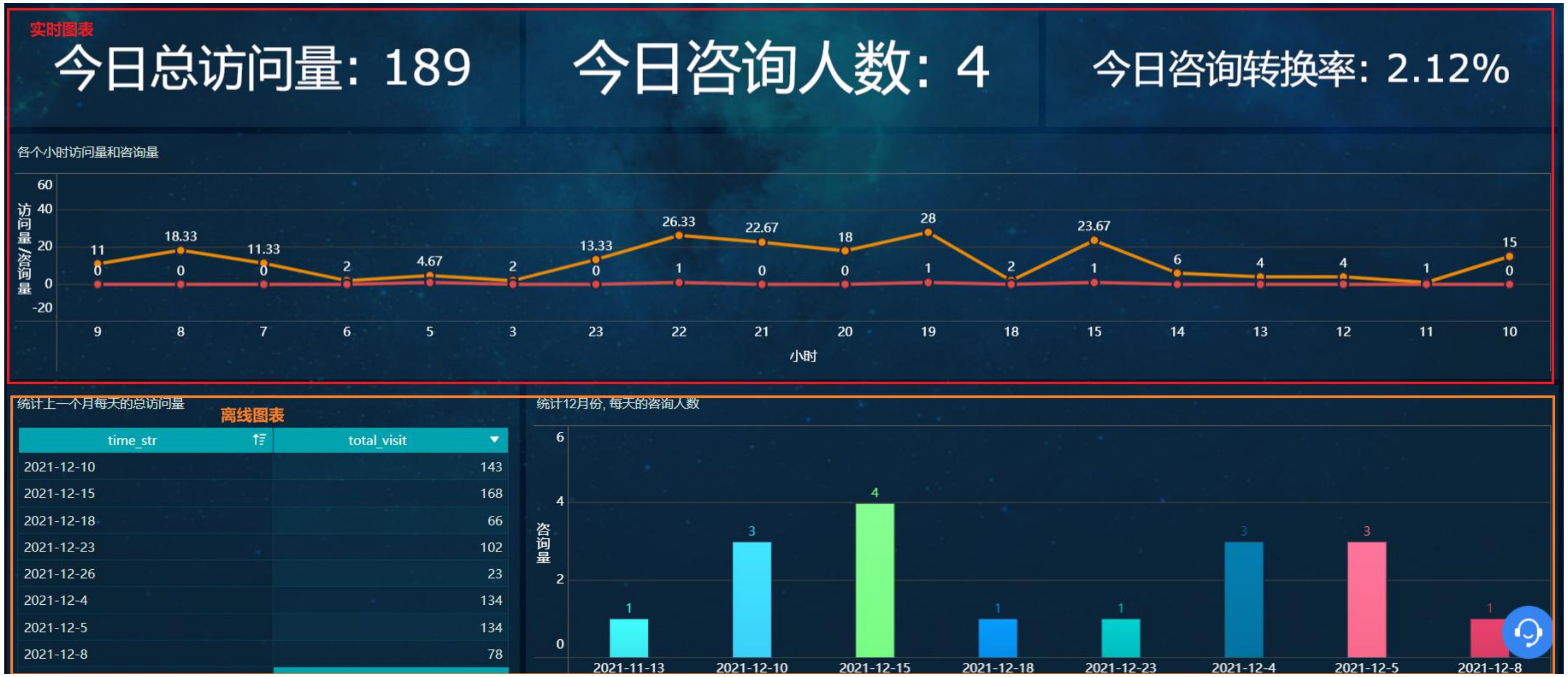
- 3- 添加标题





FineBI实现实时大屏

- 4- 进行后续的图表制作： 此部分细节较多，大家可根据视频或者官网说明来操作，难度系数不高



今日总访问量：189

今日咨询人数：4

今日咨询转换率：2.12%

统计上一个月每天的总访问量

time_str	total_visit
2021-12-10	143
2021-12-15	168
2021-12-18	66
2021-12-23	102
2021-12-26	23
2021-12-4	134
2021-12-5	134
2021-12-8	78