作业二实验报告

团队成员分工：

- 祝溢泽218352001：可视化展示
- 宣伟康218352002：环境搭建、代码编写
- 陈绘新218352003：数据接收
- 顾城218352004：环境搭建、代码编写

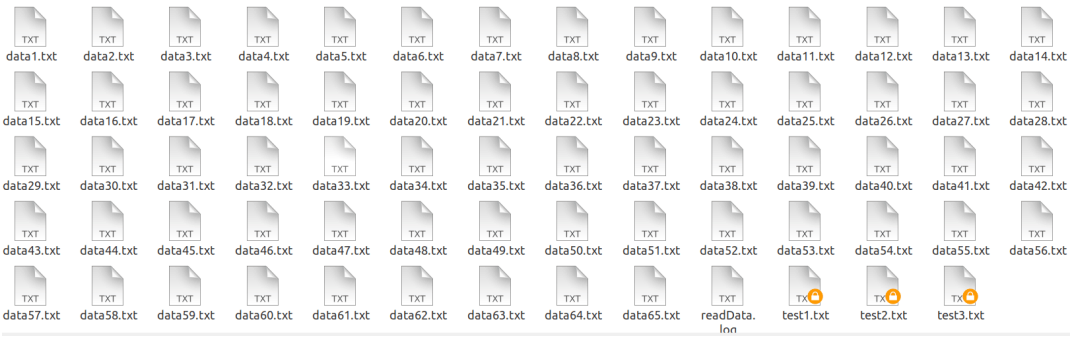
实验流程

数据获取

- 静态数据获取

```
    val reader = session.read.format("jdbc")
      .option("url", "jdbc:hive2://172.29.4.17:10000/default")
      .option("user", "student")
      .option("password", "nju2022")
      .option("driver", "org.apache.hive.jdbc.HiveDriver")
    val registerHiveDqlDialect = new RegisterHiveSqlDialect()
    registerHiveDqlDialect.register()
```

- 动态数据获取
  - 搭建消费者获取流数据，暂存到txt中



  - 在分析数据时，再创建生产者，将这些数据通过生产者发送

实验环境的搭建

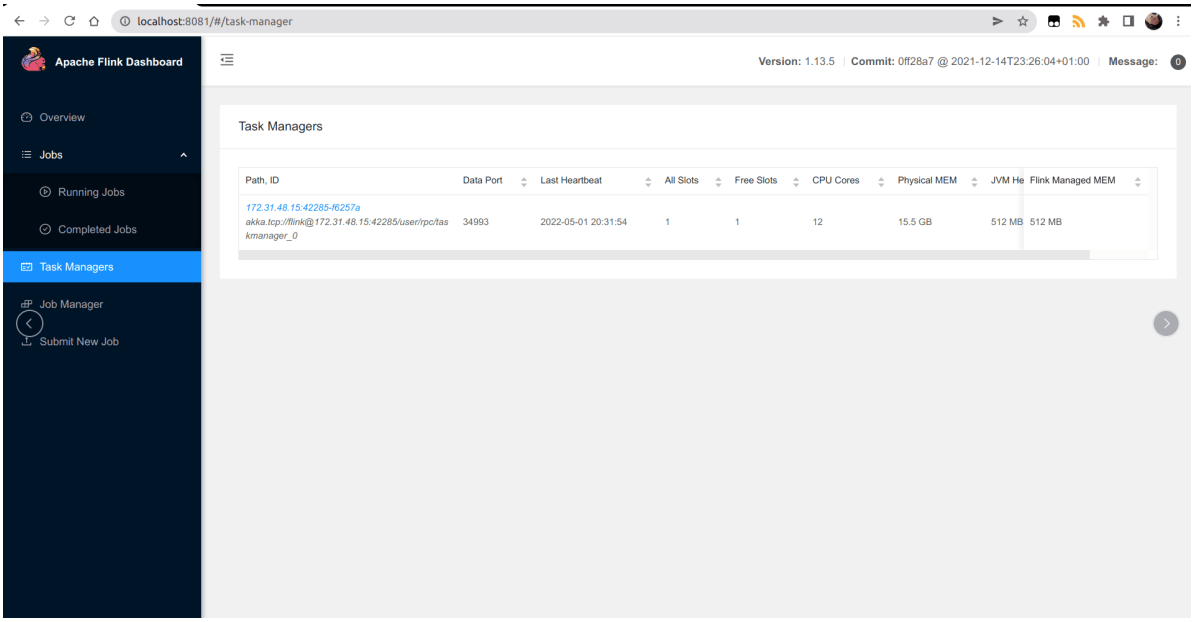
- 搭建Hadoop、Spark

```
jackson@jackson-Lenovo-XiaoXinPro-13API-2019:/usr/local/spark/sbin$ jps
12209 ResourceManager
12052 SecondaryNameNode
13077 Jps
11653 NameNode
10007 NailgunRunner
11831 DataNode
12824 Master
10041 RemoteMavenServer36
13002 Worker
12527 NodeManager
9455 Main
```

- 创建Kafka

```
etl
mytopic
testAll
transaction
=====创建kafka主题命令：=====
/usr/local/kafka/bin/kafka-topics.sh --create --zookeeper 172.31.48.15:2181 --replication-factor 1 --partitions 1 --topic 主题名
=====等待3秒...=====
=====Jps=====
18817 Main
323142 Worker
322958 Master
322542 ResourceManager
31148 QuorumPeerMain
31149 Kafka
57614 Launcher
45331 StandaloneSessionClusterEntrypoint
306608 org.eclipse.equinox.launcher_1.6.300.v20210813-1054.jar
45622 TaskManagerRunner
322879 NodeManager
57661 MyFlinkSinkClickhouse
```

- Flink的搭建



数据转换

```
SingleOutputStreamOperator<Object> dataStream = source.map((value) -> {
    if (value == null) {
        return null;
    } else {
        JSONObject jsonObject = JSONObject.parseObject(value);
        String eventType = jsonObject.getString("eventType");
        if (eventType == null) {
            return null;
        }
    }
}
```

根据不同的eventtype转化为不同的java对象(部分举例)

```
if ("sa".equals(eventType)) {
            return jsonObject.getObject("eventBody", dm_v_tr_sa_mx.class);
        } else if("shop".equals(eventType)){
            return jsonObject.getObject("eventBody", dm_hlw_shop_info.class);
        }else if("djk_info".equals(eventType)){
            return jsonObject.getObject("eventBody", dm_v_as_djk_info.class);
        }else if("djkfq".equals(eventType)){
            return jsonObject.getObject("eventBody", dm_v_as_djkfq_info.class);
        }
```

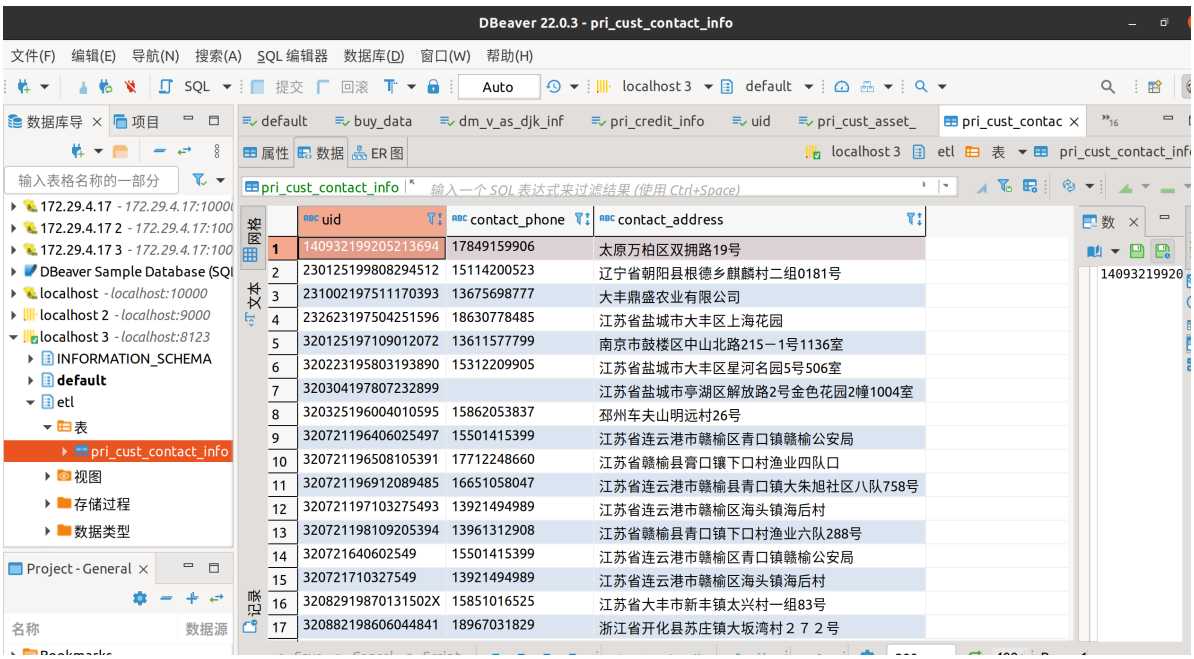
数据存储

- 数据写入clickhouse

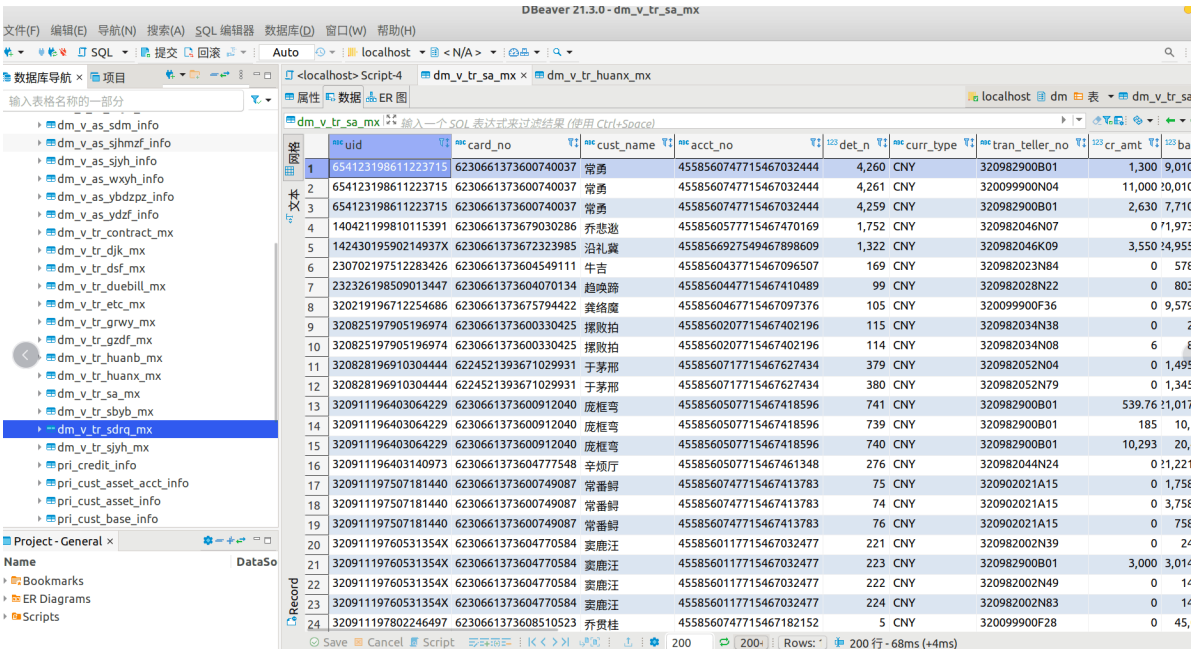
```
 StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
        String topic = "etl";
        Properties props = new Properties();
        props.setProperty("bootstrap.servers", "172.31.48.15:9092");
        props.setProperty("group.id", "consumer-group");
        props.setProperty("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        props.setProperty("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        FlinkKafkaConsumer010<String> consumer = new FlinkKafkaConsumer010(topic,
new SimpleStringSchema(), props);
        consumer.setStartFromGroupOffsets();
        consumer.setStartFromEarliest();
        DataStreamSource<String> source = env.addSource(consumer);
```

实现对pri_cust_contact_info表的ETL

```
      rdf = session.sql("select * from UseLessData where contact not in('无',
'null', '', '-')")

      rdf = rdf.withColumn("contact_phone",

        when(rdf.col("con_type") === "TEL" || rdf.col("con_type") === "OTH"
          || rdf.col("con_type") === "MOB",
          col("contact")))

      rdf = rdf.withColumn("contact_address",
        when(rdf.col("con_type").notEqual("TEL") &&
rdf.col("con_type").notEqual("OTH")
          && rdf.col("con_type").notEqual("MOB"),
          col("contact")))
      //    删去其他字段
      rdf = rdf.drop("con_type", "contact", "sys_source", "create_date",
"update_date")

      rdf = rdf.dropDuplicates()
      rdf.createTempView("tempView")

      val tempView = session.sql("select a.uid, " +
        "concat_ws(',',collect_list(a.contact_phone)) as contact_phone,\n " +
        "concat_ws(',',collect_list(a.contact_address)) as contact_address\n " +
        "from tempView a\n" +
        "group by a.uid"
      )
      rdf = tempView.drop("con_rn")
      res = rdf
      println(res.count())
    }
```



实现对流式数据的存储



可视化展示

我们选取了shop该表，用python做了一个可视化展示。该表展示了日期与销售额之间的关系。