



大数据技术基础课程实验报告

期末实验二：基于华为云的数据实时处理与展示综合实践

付容天

学号 2020211616

班级 2020211310

计算机学院（国家示范性软件学院）

2023 年 5 月 24 日

1. 基础配置部分

假设平台已经将每个商品的订单信息实时写入 Kafka 中，为了实现对采集到的数据进行实时处理并做关联数据库的查询展示，我们需要先进行一些基础配置，主要包括：

- (1) 购买虚拟私有云 VPC；
- (2) 开通数据接入服务（DMS Kafka）；
- (3) 创建数据库服务（MySQL）；
- (4) 创建数据表；
- (5) 开通数据探索服务（DLI）。

下面我结合得分点给出这部分的试验记录和分析。

根据实验指导书，我创建了符合要求的 VPC，如下所示：



图 1：创建的 VPC 截图

接着，我按照实验指导书创建了 Kafka 实例，截图如下：

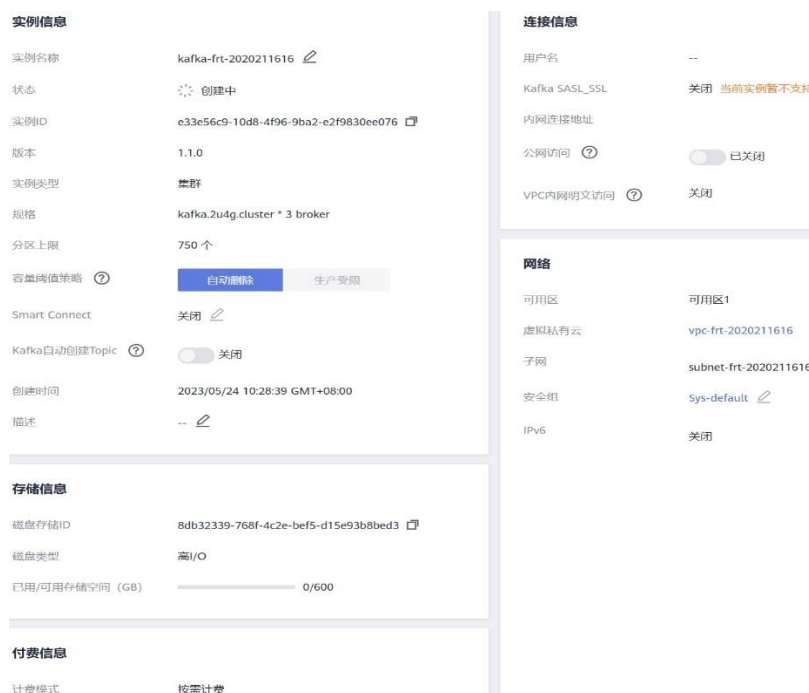


图 2：创建的 Kafka 实例截图

然后，我按照实验指导书要求创建了数据库服务，截图如下：

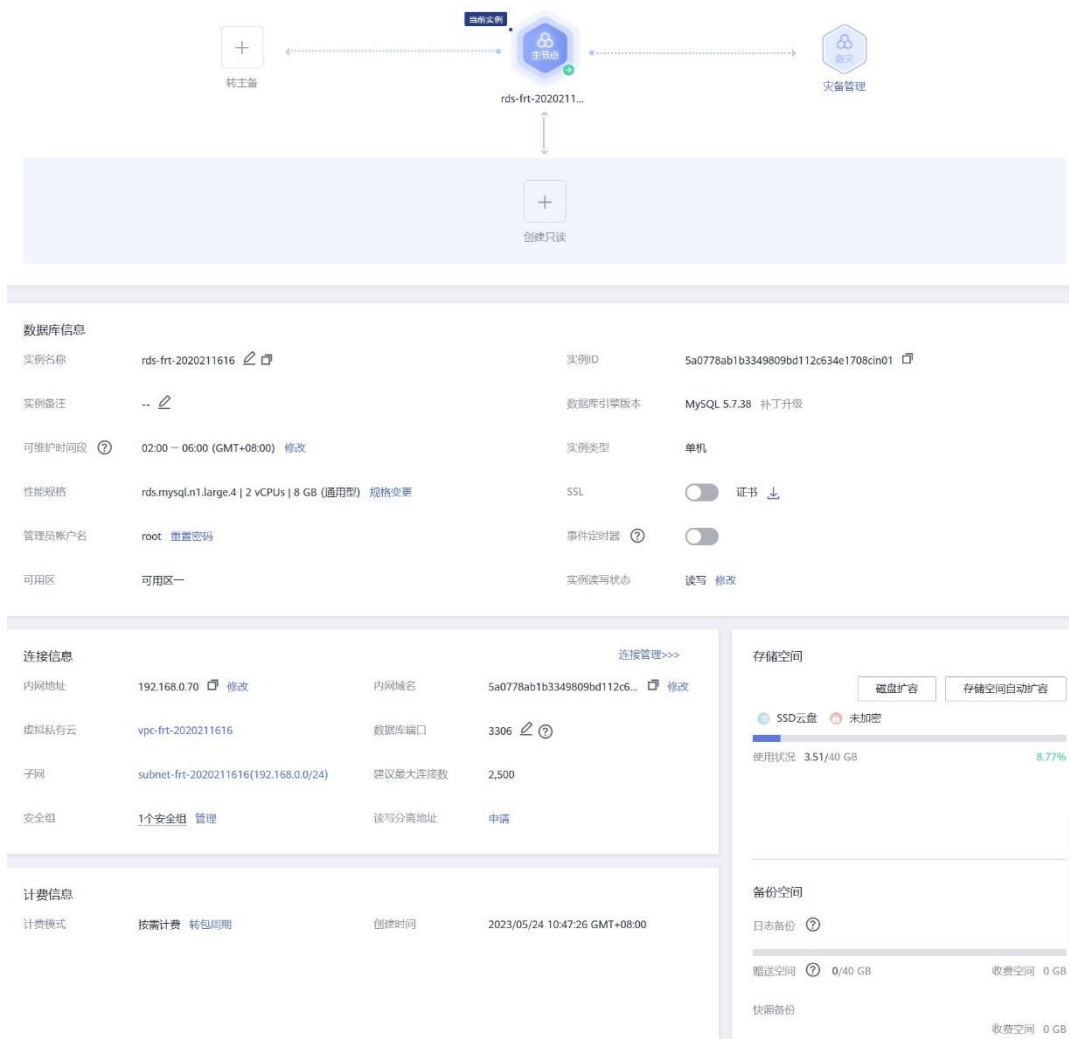


图 3: MySQL 数据库截图

在此基础上，我登录刚刚创建的数据库，创建了如下图所示的数据表：

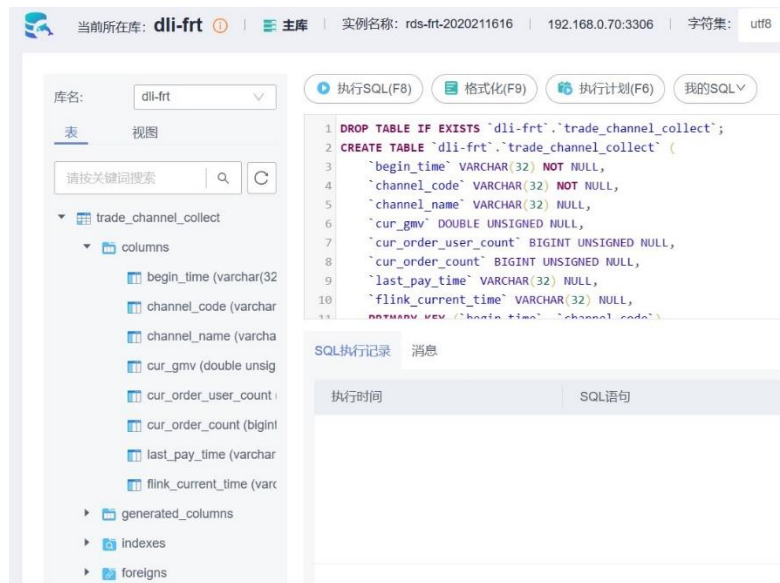


图 4: 新建表格截图

最后，我在“数据湖探索 DLI”页面中开通了数据探索服务，并按照实验指导书的要求新建了相应的队列，命名为 d1l_cce_queue_frt，如下所示：

名称	类型	规格	实际CUs	弹性扩缩容	计费模式	用户名
default	SQL队列	--	--	上限: -- CUs 下限: -- CUs	SQL计算量 2019/03/21 23:40:21 GMT+08:00 创建	DLI
d1l_cce_queue_frt	通用队列	16 CUs	16 CUs	上限: -- CUs 下限: -- CUs	按需计费 2023/05/24 11:00:36 GMT+08:00 创建	rongtianfu-bupt

名称	d1l_cce_queue_frt	队列最大CUs	16
CPU架构	X86	专属资源模式	是
AZ策略	单AZ	网段	172.16.0.0/16
用户名	rongtianfu-bupt	创建时间	2023/05/24 11:00:36 GMT+08:00

图 5：新建队列截图

并且，在数据湖页面配置跨源链接 connect_vpc_frt，并选择创建好的队列 d1l_cce_queue_frt、选择虚拟私有云为 vpc_frt_2020211616，最后得到如下图所示的已激活的跨源连接截图：

增强型跨源	经典型跨源	跨源认证
创建	增强型跨源支持包周期队列和专属资源模式下创建的队列。了解更多DLI队列网络连接操作指导。	
连接名称	连接状态	虚拟私有云
connect_vpc_frt	已激活	vpc-frt-2020211616
		子网
		subnet-frt-2020211616
		创建时间
		2023/05/24 11:04:09 ...
		删除连接 路由信息 更多

图 6：跨源连接截图

紧接着测试 DLI 队列与 RDS 的联通性，下图 7 展示了 RDS 的地址和端口，图 8 展示了测试结果：

转主备

当前实例

主实例

rds-frt-2020211...

创建只读

数据库信息

实例名称	rds-frt-2020211616	实例ID	5a0
实例备机	--	数据库引擎版本	MyS
可维护时间段	02:00 ~ 06:00 (GMT+08:00)	实例类型	单折
性能规格	rds.mysql.n1.large.4 2 vCPUs 8 GB (通用型)	SSL	
管理员帐户名	root	事件定时器	
可用区	可用区一	实例读写状态	读写

连接信息

内网地址	192.168.0.70	内网域名	5a0778ab1b3349809...
虚拟私有云	vpc-frt-2020211616	数据库端口	3306
子网	subnet-frt-2020211616(192.168...	建议最大连接数	2,500
安全组	1个安全组	读写分离地址	申请

基本信息

备份恢复

连接管理

帐号管理

数据库管理

日志管理

SQL审计

参数修改

高级运维

智能DBA助手

标签

云DBA

数据库代理

评价

登录

图 7：RDS 地址与端口信息（图中标红处）

测试地址连通性

测试队列到指定地址是否可达，支持域名和ip，可指定端口。

★ 地址

192.168.0.70:3306

地址192.168.0.70:3306可达。

测试

取消

图 8：RDS 联通性测试

之后测试 DLI 与 Kafka 实例的联通性，下图 9 展示了 Kafka 实例的地址与端口信息，图 10 展示了联通性测试结果：

<input type="checkbox"/>	名称	监控	状态	版本
<input type="checkbox"/>	kafka-frt-2020211616 e33e56c9-10d8-4f96-9ba2-e2f9830ee076		运行中	1.1.0

查看连接地址

×

内网IPv4连接地址 192.168.0.90:9092,192.168.0.91:9092,192.168.0.148:9092

确定

图 9：Kafka 实例地址与端口信息

测试地址连通性

测试队列到指定地址是否可达，支持域名和ip，可指定端口。

★ 地址

192.168.0.90:9092

地址192.168.0.90:9092可达。

测试

取消

图 10：联通性测试结果（可达）

2. 新建作业、运行并查看

在这部分的实验中，我主要是完成了数据统计作业创建和运行，并在完成了必要配置之后输入了相关数据，最后查看结果，具体而言，先后完成了：

- (1) 新建 Flink 作业；
- (2) 新建云主机服务（ECS）并安装 Kafka 客户端；
- (3) 向服务器发送数据；
- (4) 从 MySQL 中查看接收到的数据。

下面我将结合得分点给出这部分实验的记录与分析。

首先，我根据实验指导书在数据湖页面中创建了新的 Flink 作业，在进行相应的配置与编辑后提交并启动 Flink 作业，得到如下所示的结果：

job_frt_2020211616运行中

ID: 246312 | 作业类型: Flink OpenSource SQL

C

作业

作业详情

任务列表

执行计划

提交日志

运行日志

标签

1

-- *****

2

-- 数据源: trade_order_detail_info (订单详情宽表)

3

-- *****

4

create table trade_order_detail (

5

order_id string, -- 订单ID

6

order_channel string, -- 渠道

7

order_time string, -- 订单创建时间

8

pay_amount double, -- 订单金额

9

real_pay double, -- 实际付费金额

10

pay_time string, -- 付费时间

类型

Flink OpenSource SQL

单TM所占CU数

1

名称

job_frt_2020211616

单TM Slot数

0

描述

--

OBS桶

frt-2020211616

状态

运行中

保存作业日志

开启

运行模式

独享

作业异常告警

关闭

所属队列

dll_cce_queue_frt

异常自动重启

关闭

Flink版本

1.12

保存点路径

--

UDF Jar

--

开启Checkpoint

关闭

优化参数

--

空闲状态保留...

1 小时

CU数量

2

脏数据策略

--

实际CU数量

2

创建时间

2023/05/24 11:13:53 GMT+08:00

管理单元

1

更新时间

2023/05/24 11:18:38 GMT+08:00

并行数

1

图 11: Flink 作业提交成功截图

然后，我购买了弹性云服务器 ECS，并在进行相应配置之后启动机器，在本地通过远程连接工具登录到服务器上，在安装完 Java 之后，输入命令查看相应状态，并截图如下：

```
[root@ecs-frt-2020211616 ~]# java -version
openjdk version "1.8.0_372"
OpenJDK Runtime Environment (build 1.8.0_372-b07)
OpenJDK 64-Bit Server VM (build 25.372-b07, mixed mode)
[root@ecs-frt-2020211616 ~]# find / -name profile
/usr/src/kernels/3.10.0-1160.53.1.el7.x86_64/include/config/branch/profile
/usr/src/kernels/3.10.0-957.el7.x86_64/include/config/branch/profile
/etc/profile
[root@ecs-frt-2020211616 ~]# find / -name jre*
/usr/lib/jvm-exports/jre-openjdk
/usr/lib/jvm-exports/jre-1.8.0
/usr/lib/jvm-exports/jre-1.8.0-openjdk-1.8.0.372.b07-1.el7_9.x86_64
/usr/lib/jvm-exports/jre
/usr/lib/jvm/jre-openjdk
```

图 12: Java 安装截图

之后安装 Kafka 客户端并进入其中的 bin 目录，在此目录下启动相应脚本程序，进入数据发送页面，发送指定数据，截图如下所示：

```
[root@ecs-frt-2020211616 bin]# ./kafka-console-producer.sh --broker-list 192.168.0.90:9092,192.168.0.91:9092,192.168.0.148:9092 --topic trade_order_detail_info_frt
{"order_id":"202103241000000001", "order_channel":"webShop", "order_time":"2021-03-24 10:00:00", "pay_amount":"100.00", "real_pay":"100.00", "pay_time":"2021-03-24 10:02:03", "user_id":"0001", "user_name":"Alice", "area_id":"330106"}
{"order_id":"202103241606060001", "order_channel":"appShop", "order_time":"2021-03-24 16:06:06", "pay_amount":"200.00", "real_pay":"180.00", "pay_time":"2021-03-24 16:10:06", "user_id":"0001", "user_name":"Alice", "area_id":"330106"}
{"order_id":"202103251202020001", "order_channel":"miniAppShop", "order_time":"2021-03-25 12:02:02", "pay_amount":"60.00", "real_pay":"60.00", "pay_time":"2021-03-25 12:03:00", "user_id":"0002", "user_name":"Bob", "area_id":"330110"}
{"order_id":"202103251505050001", "order_channel":"qqShop", "order_time":"2021-03-25 15:05:05", "pay_amount":"500.00", "real_pay":"400.00", "pay_time":"2021-03-25 15:10:00", "user_id":"0003", "user_name":"Cindy", "area_id":"330108"}
{"order_id":"202103252020200001", "order_channel":"webShop", "order_time":"2021-03-24 20:20:20", "pay_amount":"600.00", "real_pay":"480.00", "pay_time":"2021-03-25 00:00:00", "user_id":"0004", "user_name":"Daisy", "area_id":"330102"}
{"order_id":"202103260808080001", "order_channel":"webShop", "order_time":"2021-03-25 08:08:08", "pay_amount":"300.00", "real_pay":"240.00", "pay_time":"2021-03-25 08:10:00", "user_id":"0004", "user_name":"Daisy", "area_id":"330102"}
{"order_id":"202103261313130001", "order_channel":"webShop", "order_time":"2021-03-25 13:13:13", "pay_amount":"100.00", "real_pay":"100.00", "pay_time":"2021-03-25 16:16:16", "user_id":"0004", "user_name":"Daisy", "area_id":"330102"}
{"order_id":"202103270606060001", "order_channel":"appShop", "order_time":"2021-03-25 06:06:06", "pay_amount":"50.50", "real_pay":"50.50", "pay_time":"2021-03-25 06:07:00", "user_id":"0001", "user_name":"Alice", "area_id":"330106"}
{"order_id":"202103270606060002", "order_channel":"webShop", "order_time":"2021-03-25 06:06:06", "pay_amount":"66.60", "real_pay":"66.60", "pay_time":"2021-03-25 06:07:00", "user_id":"0002", "user_name":"Bob", "area_id":"330110"}
{"order_id":"202103270606060003", "order_channel":"miniAppShop", "order_time":"2021-03-25 06:06:06", "pay_amount":"88.80", "real_pay":"88.80", "pay_time":"2021-03-25 06:07:00", "user_id":"0003", "user_name":"Cindy", "area_id":"330108"}
{"order_id":"202103270606060004", "order_channel":"webShop", "order_time":"2021-03-25 06:06:06", "pay_amount":"99.90", "real_pay":"99.90", "pay_time":"2021-03-25 06:07:00", "user_id":"0004", "user_name":"Daisy", "area_id":"330102"}
```

图 13: 客户端发送消息至 Kafka 的截图

然后进入华为云中的 Kafka 实例内，点击消息查询，可以看到收到了如下所示的消息：

* Topic 名称

trade_order_detail_info_frt

分区

请输入分区

* 查询方式

按创建时间查询

2023/05/24 00:00:00 — 2023/05/24 23:59:59

📅

搜索

重置

Topic 名称	分区	偏移量	消息大小 (B)	创建时间 ⌵	操作
trade_order_detail_info_frt	0	21	0	2023/05/24 12:07:51 GMT+0...	查看消息正文
trade_order_detail_info_frt	0	20	231	2023/05/24 12:07:51 GMT+0...	查看消息正文
trade_order_detail_info_frt	0	19	0	2023/05/24 12:07:51 GMT+0...	查看消息正文
trade_order_detail_info_frt	0	18	235	2023/05/24 12:07:51 GMT+0...	查看消息正文
trade_order_detail_info_frt	0	17	0	2023/05/24 12:07:51 GMT+0...	查看消息正文
trade_order_detail_info_frt	0	16	229	2023/05/24 12:07:51 GMT+0...	查看消息正文
trade_order_detail_info_frt	0	15	0	2023/05/24 12:07:51 GMT+0...	查看消息正文
trade_order_detail_info_frt	0	14	231	2023/05/24 12:07:51 GMT+0...	查看消息正文
trade_order_detail_info_frt	0	13	0	2023/05/24 12:07:51 GMT+0...	查看消息正文
trade_order_detail_info_frt	0	12	233	2023/05/24 12:07:51 GMT+0...	查看消息正文

10

每页 20

1 2 3 4 5

共 1 页

图 14: Kafka 收到消息的截图

最后，我从 MySQL 中查看接收到的数据，输入 SQL 查询语句，得到如下所示的结果：

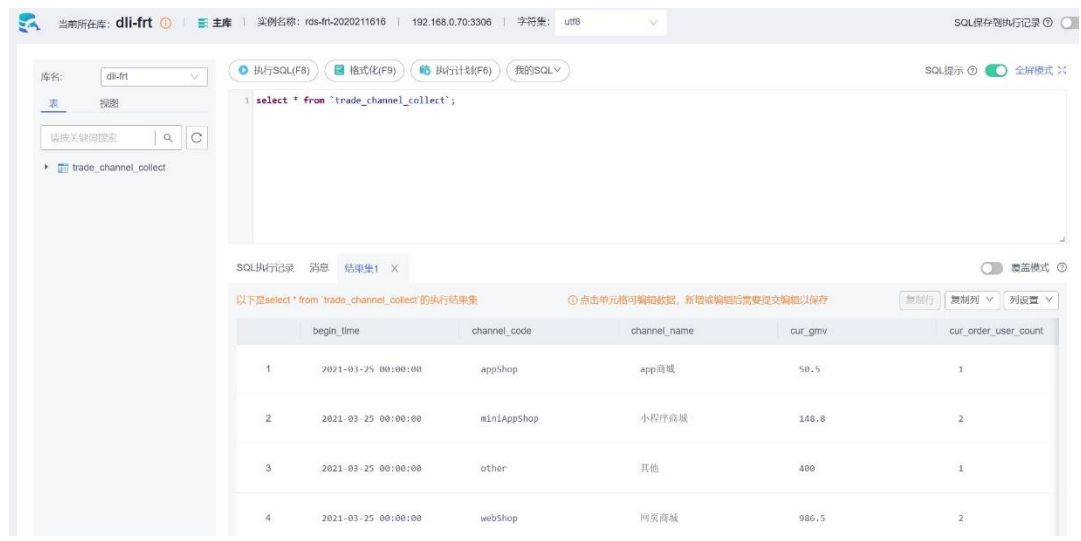


图 15：数据库查询结果截图

3. 实际应用部分

在这部分实验中，我们需要参考上面的实验流程，自行设计一个应用场景，完成数据的实时处理。我选择的应用场景是学生课程分数统计场景。

在学生课程分数统计场景中，数据集的每一行表示一个学生关于一门课的成绩，其数据结构如下所示：

学生 ID	学生班级	所在学院	课程名称	课程分数
stu_id (string)	stu_class (string)	stu_school (string)	course_name (string)	course_score (double)

我们需要实现的效果就是实时地统计出不同班级不同课程的最高分、最低分和平均分，因此结果表的数据结构设计为：

学生班级	课程名称	最高分	最低分	平均分
stu_class (string)	course_name (string)	max_score (double)	min_score (double)	avg_score (double)

我们所用的数据集完整内容在本报告最后的附录中给出。

现在来看我为这个场景和需求编写的 Flink 脚本:

```
--*****
-- 数据源: stu_score_detail (学生课程成绩表)
--*****

create table stu_score_detail (
    stu_id string,      -- 学生ID
    stu_class string,   -- 学生班级
    stu_school string,  -- 学生所在学院
    course_name string, -- 课程名称
    course_score double, -- 课程分数
) with (
    "connector" = "kafka",
    "properties.bootstrap.servers" = "192.168.0.90:9092,192.168.0.91:
9092,192.168.0.148:9092",      -- Kafka连接地址
    "properties.group.id" = "stu_score", -- Kafka groupID
    "topic" = "stu_score_frt",    -- Kafka topic
    "format" = "json",
    "scan.startup.mode" = "latest-offset"
);

--*****
-- 结果表: stu_score_collect (各班级不同课程数据分析表)
--*****

create table stu_score_collect(
    stu_class string,      -- 学生班级
    course_name string,    -- 课程名称
    max_score double,      -- 最高成绩
    min_score double,      -- 最低成绩
    avg_score double,      -- 平均成绩
    primary key (stu_class, course_name) not enforced
) with (
    "connector" = "jdbc",
    "url" = "jdbc:mysql://192.168.0.70:3306/dli-frt",
    "table-name" = "stu_score_collect", -- mysql表名
    "username" = "root",                -- mysql用户名
    "password" = "frtFRT189260",        -- mysql密码
    "sink.buffer-flush.max-rows" = "1000",
    "sink.buffer-flush.interval" = "1s"
);

--*****
-- 具体统计操作
--*****
```

```

insert into stu_score_collect
select
    stu_class
    , course_name
    , MAX(course_score) as max_score
    , MIN(course_score) as min_score
    , AVG(course_score) as avg_score
from stu_score_detail
GROUP BY stu_class, course_name;

```

首先创建数据源表 stu_score_detail，并从 Kafka 中得到相应的数据，并通过 SQL 语法所支持的 insert 语句、select 语句、group by 结构、max 特性等功能得到每个班级每门课程的最高分、最低分和平均分，最后将这些数据输入到结果表 stu_score_collect 中。

我创建了相应的 topic 为 stu_score_frt，并将如上所述的 Flink 脚本提交运行，如下图所示：

The screenshot displays the Flink job management interface. On the left, a sidebar shows a tree view of jobs under 'Flink作业', including 'job_stu_score_frt_2020...' and 'job_frt_2020211616'. The main panel shows details for the job 'job_stu_score_frt_2020211616', which is in a '运行中' (Running) state. The job ID is 246317, and it is a Flink OpenSource SQL job. Below the job details, there is a '作业详情' (Job Details) tab, which is selected. This tab shows the Flink SQL script used for the job, which includes creating a table 'stu_score_detail' and inserting data from a Kafka source. The script is as follows:

```

1 ..=====
2 -- 数据源: stu_score_detail (学生课程成绩表)
3 ..=====
4 create table stu_score_detail (
5   stu_id string,      -- 学生ID
6   stu_class string,   -- 学生班级
7   stu_school string,  -- 学生所在学院
8   course_name string, -- 课程名称
9   course_score double -- 课程分数
10 ) with (
11   "connector" = "kafka",
12   "properties.bootstrap.servers" = "192.168.0.90:9092,192.168.0.91:9092,192.168.0.148:9092", -- Kafka连接地址
13   "properties.group.id" = "stu_score", -- Kafka groupID
14   "topic" = "stu_score_frt", -- Kafka topic
15   "format" = "json",
16   "scan.startup.mode" = "latest-offset"
17 );
18

```

Below the script, there is a table summarizing the job configuration:

类型	Flink OpenSource SQL	单TM所占CU数	1
名称	job_stu_score_frt_2020211616	单TM Slot数	0
描述	--	OBS桶	frt-2020211616
状态	运行中	保存作业日志	开启
运行模式	独享	作业异常告警	关闭
所属队列	dli_cce_queue_frt	异常自动重启	关闭
Flink版本	1.12	保存点路径	--
UDF Jar	--	开启Checkpoint	关闭
优化参数	--	空闲状态保留...	1 小时
CU数量	2	脏数据策略	--
实际CU数量	2	创建时间	2023/05/24 13:10:05 GMT+08:00
管理单元	1	更新时间	2023/05/24 13:12:35 GMT+08:00
并行数	1		

图 16：自编 Flink 脚本正常运行截图

并且，我在 MySQL 中创建了相应的表格，该表格就是前面所述的“结果表”，截图如下所示：

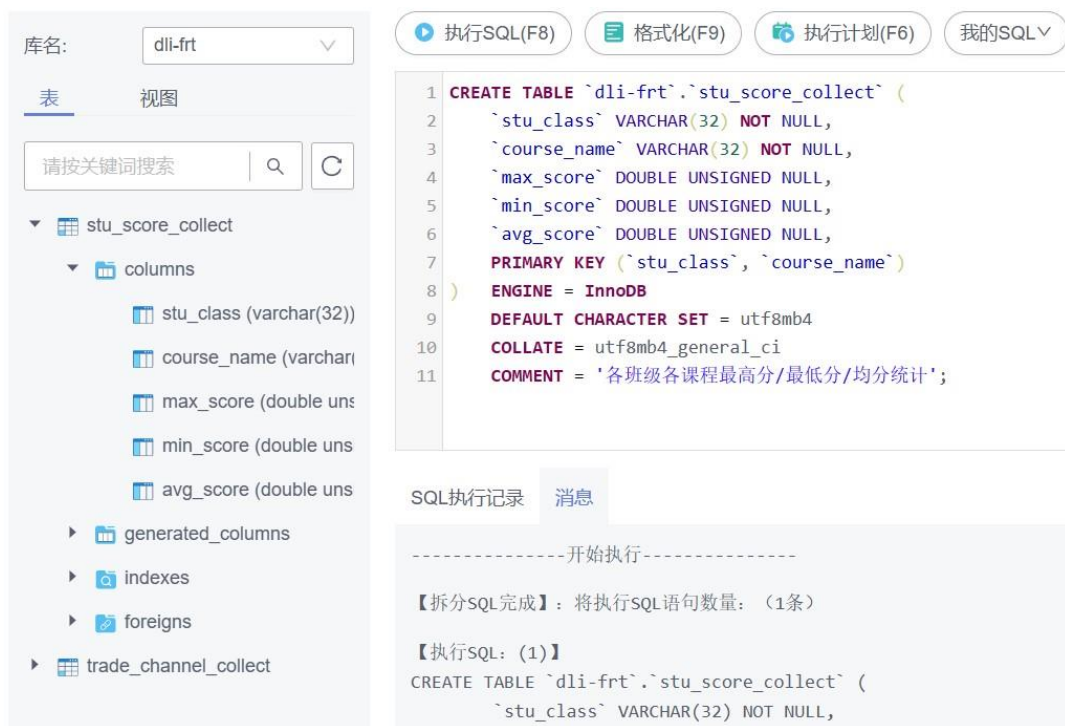


图 17: 创建结果表

然后在本地通过远程连接工具连接到服务器上，并上传如附录所示的待统计数据，截图如下：

```
[root@ecs-frt-2020211616 bin]# ./kafka-console-producer.sh --broker-list 192.168.0.90:9092,192.168.0.91:9092,192.168.0.148:9092 --topic stu_score_frft
{"stu_id":"2020001","stu_class":"class1","stu_school":"计算机学院","course_name":"高等数学","course_score":"90.0"}
{"stu_id":"2020001","stu_class":"class1","stu_school":"计算机学院","course_name":"大学英语","course_score":"88.0"}
{"stu_id":"2020001","stu_class":"class1","stu_school":"计算机学院","course_name":"大数据","course_score":"92.0"}
{"stu_id":"2020002","stu_class":"class1","stu_school":"计算机学院","course_name":"高等数学","course_score":"99.0"}
{"stu_id":"2020002","stu_class":"class1","stu_school":"计算机学院","course_name":"大学英语","course_score":"85.0"}
{"stu_id":"2020002","stu_class":"class1","stu_school":"计算机学院","course_name":"大数据","course_score":"87.0"}
{"stu_id":"2020003","stu_class":"class1","stu_school":"计算机学院","course_name":"高等数学","course_score":"83.0"}
{"stu_id":"2020003","stu_class":"class1","stu_school":"计算机学院","course_name":"大学英语","course_score":"90.0"}
{"stu_id":"2020003","stu_class":"class1","stu_school":"计算机学院","course_name":"大数据","course_score":"96.0"}
{"stu_id":"2020003","stu_class":"class1","stu_school":"计算机学院","course_name":"高等数学","course_score":"95.0"}
{"stu_id":"2020004","stu_class":"class1","stu_school":"计算机学院","course_name":"大学英语","course_score":"86.0"}
{"stu_id":"2020004","stu_class":"class1","stu_school":"计算机学院","course_name":"大数据","course_score":"89.0"}
{"stu_id":"2020005","stu_class":"class2","stu_school":"计算机学院","course_name":"高等数学","course_score":"91.0"}
{"stu_id":"2020005","stu_class":"class2","stu_school":"计算机学院","course_name":"大学英语","course_score":"92.0"}
{"stu_id":"2020005","stu_class":"class2","stu_school":"计算机学院","course_name":"大数据","course_score":"83.0"}
{"stu_id":"2020006","stu_class":"class2","stu_school":"计算机学院","course_name":"高等数学","course_score":"93.0"}
{"stu_id":"2020006","stu_class":"class2","stu_school":"计算机学院","course_name":"大学英语","course_score":"94.0"}
{"stu_id":"2020006","stu_class":"class2","stu_school":"计算机学院","course_name":"大数据","course_score":"88.0"}
{"stu_id":"2020007","stu_class":"class2","stu_school":"计算机学院","course_name":"高等数学","course_score":"89.0"}
{"stu_id":"2020007","stu_class":"class2","stu_school":"计算机学院","course_name":"大学英语","course_score":"90.0"}
{"stu_id":"2020007","stu_class":"class2","stu_school":"计算机学院","course_name":"大数据","course_score":"87.0"}
{"stu_id":"2020008","stu_class":"class2","stu_school":"计算机学院","course_name":"高等数学","course_score":"90.0"}
{"stu_id":"2020008","stu_class":"class2","stu_school":"计算机学院","course_name":"大学英语","course_score":"90.0"}
{"stu_id":"2020008","stu_class":"class2","stu_school":"计算机学院","course_name":"大数据","course_score":"87.0"}
```

图 18: 上传待处理数据

最后，我打开了 MySQL 页面，通过相应查询语句，得到了数据处理结果，每条记录表示了每个班级每门课程的最高分、最低分和平均分，查询结果截图如下所示：

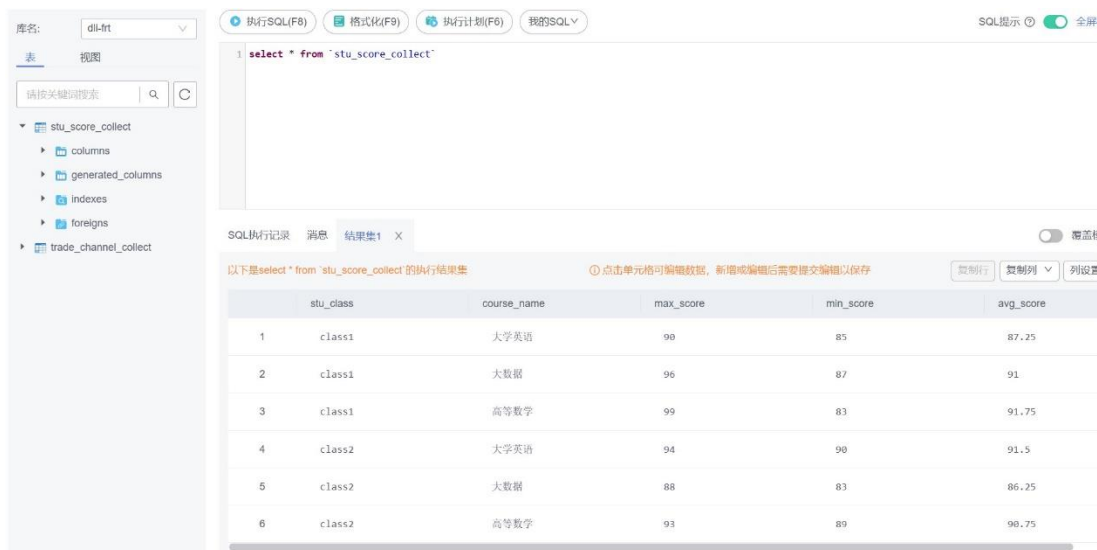


图 19: 查询得到数据处理结果

4. 实验问题与实验总结

在本次实验中，我也遇到了几个问题，现记录如下：

- (1) 联通性测试失败：可能是需要等待一段时间，一开始测试失败，但我耐心地多测试了几次，后面就一直是成功联通的；
- (2) SQL 语句不熟：主要体在编写自己的 Flink 脚本的时候犯基本语法的错误，通过复习数据库课程知识，解决了这个问题。

本次实验是大数据技术基础课程的最后一个实验，我在之前的实验环境下结合实验指导书完成了所有的实验任务。我实现了 Flink 从 Kafka 中实时读取数据、进行统计并输出的功能。我还自行设计了基于实际情况的学生课程成绩统计实验，编写了相应的 Flink 脚本对输入的学生课程成绩进行实时统计并显示。总的来说，我在本次实验中复习了理论知识、进行了工程实践，圆满完成了实验任务，收获颇丰！

在这学期的大数据技术基础课程中，我在鄂海红老师和各位助教、同学的帮助下圆满完成了所有既定的实验任务，对当今热门的大数据技术有了基本的认识。鄂老师耐心细致的讲解、助教和同学们的热心帮助，是我能够完成这门课程的重要原因！感谢鄂老师给我们提供实验环境并进行非常到位的讲解，感谢各位助教以及所有一起参加本课程同学们！

附录：待处理数据

[illegible]