



Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №4
з дисципліни
Сучасні методи обробки масивів даних

Виконав(ла):

студент(ка) групи ІМ-42мп:
Бардін В. Д.

Перевірила:

ст. викладач
Тимофєєва Ю.С.

Київ 2024

1 КОД ЗАСТОСУНКУ

```
public class Lab4Streams : IStreamTopologyBuilder
{
    private const string InputTopic = "int.k-connect.csv.plastic-pollution";

    private static JsonSerializerSettings GetJsonSerializerSettings()
    {
        return new JsonSerializerSettings
        {
            Converters = { new NaIntConverter() },
            ContractResolver = new DefaultContractResolver
            {
                NamingStrategy = new SnakeCaseNamingStrategy(),
            }
        };
    }

    public StreamBuilder BuildTopology(StreamBuilder streamBuilder)
    {
        JsonConvert.DefaultSettings = GetJsonSerializerSettings();

        var records = streamBuilder.Stream(InputTopic, new StringSerDes(), new
        JsonSerializerSettings<PlasticPollutionInfo>())
            .Filter((_, v, _) => v is not null);

        BuildVolunteersCountTopology(records);
        BuildUkraineTotalCountTopology(records);

        return streamBuilder;
    }

    private static void BuildVolunteersCountTopology(IKStream<string,
    PlasticPollutionInfo> records)
    {
        const int minEventsThreshold = 10;
        var volunteersCountStream = records
            .Filter((_, v, _) => v.NumEvents < minEventsThreshold)
            .Map<string, int>((_, v, _) => KeyValuePair.Create("passed",
            v.Volunteers))
            .GroupByKey<StringSerDes, Int32SerDes>()
            .Aggregate<int, Int32SerDes>(
                () => 0,
                (_, v, agg) => agg + v
            )
            .ToStream();

        volunteersCountStream.Print(Printed<string, int>.ToOut());
    }

    private static void BuildUkraineTotalCountTopology(IKStream<string,
    PlasticPollutionInfo> records)
    {
        const string ukraineEventsKey = "Ukraine";
        var totalCollectedInUkraine = records.MapValues<long>((_, v, _) =>
            v.NumEvents)
    }
```

```
        .Filter((k, _, _) => k is ukraineEventsKey)
        .GroupByKey()
        .Aggregate<long, Int64SerDes>(
            () => 0,
            (_, v, agg) => agg + v
        )
        .ToStream();

    totalCollectedInUkraine.Print(Printed<string, long>.ToOut());
}
```

2 СКРІНШОТИ З РЕЗУЛЬТАТАМИ РОБОТИ

The screenshot shows the Visual Studio IDE with the 'KafkaStreamsProcessor' project open. The 'Lab4Streams.cs' file is selected in the Solution Explorer and is open in the editor. The code defines a `Lab4Streams` class that implements `IStreamTopologyBuilder`. It includes a `BuildTopology` method that filters records with `NumEvents < 10`, maps them to a `KeyValuePair` with the value `"passed"`, groups them by key, and aggregates them using `Aggregate<int, Int32SerDes>` to sum the values. The output console shows a series of 'passed' messages with IDs ranging from 6210783 to 6210972, followed by an info message indicating that 19216 total records were processed.

```
public class Lab4Streams : IStreamTopologyBuilder
{
    public StreamBuilder BuildTopology(StreamBuilder streamBuilder)
    {
        // .Print(Printed<string, long>.ToOut());

        records
            .Filter((_, v, _) => v.NumEvents < 10)
            .Map<string, int>((_, v, _) => KeyValuePair.Create("passed", v.Volunteers))
            .GroupByKey<StringSerDes, Int32SerDes>()
            .Aggregate<int, Int32SerDes>((
                () => 0,
                (_, v, agg) => agg + v
            ))
            .ToStream()
            .Print(Printed<string, int>.ToOut());

        return streamBuilder;
    }
}
```

Run: KafkaStreamsProcessor

Output:

```
[ ]: passed 6210783
[ ]: passed 6210810
[ ]: passed 6210837
[ ]: passed 6210864
[ ]: passed 6210891
[ ]: passed 6210918
[ ]: passed 6210945
[ ]: passed 6210972
Info: Streamiz.Kafka.Net.Processors.StreamThread[0]
      Processed 19216 total records, ran 0 punctuators and committed 6 total tasks since the last update
```

The screenshot shows the Visual Studio IDE with the 'KafkaStreamsProcessor' project open. The 'Lab4Streams.cs' file is selected in the Solution Explorer and is open in the editor. The code defines a `Lab4Streams` class that implements `IStreamTopologyBuilder`. It includes a `BuildTopology` method that filters records with `k is "Ukraine"`, maps them to a `KeyValuePair` with the value `"Ukraine"`, groups them by key, and aggregates them using `Aggregate<long, Int64SerDes>` to sum the values. The output console shows a series of 'Ukraine' messages with IDs ranging from 17499 to 17635, followed by an info message indicating that 26760 total records were processed.

```
public class Lab4Streams : IStreamTopologyBuilder
{
    public StreamBuilder BuildTopology(StreamBuilder streamBuilder)
    {
        var records = streamBuilder.Stream(inputTopic, new StringSerDes(), new JsonSerializer<PlasticPollutionInfo>());

        records
            .MapValues<long>((_, v, _) => v.NumEvents)
            .Filter((k, _, _) => k is "Ukraine")
            .GroupByKey()
            .Aggregate<long, Int64SerDes>((
                () => 0L,
                (_, v, agg) => agg + v
            ))
            .ToStream()
            .Print(Printed<string, long>.ToOut());

        return streamBuilder;
    }
}
```

Run: KafkaStreamsProcessor

Output:

```
[ ]: Ukraine 17499
[ ]: Ukraine 17516
[ ]: Ukraine 17533
[ ]: Ukraine 17550
[ ]: Ukraine 17567
[ ]: Ukraine 17584
[ ]: Ukraine 17601
[ ]: Ukraine 17618
[ ]: Ukraine 17635
Info: Streamiz.Kafka.Net.Processors.StreamThread[0]
      Processed 26760 total records, ran 0 punctuators and committed 5 total tasks since the last update
```

3 ВИСНОВКИ

У цій роботі було використано Kafka Streams для обробки вхідного потоку даних та їх агрегації за допомогою використання операцій зі збереженням стану.