def￼Scope Tutorial

Author: Saveen Reddy

Date: 1/31/2015

# Table of Contents

[Table of Contents 1](#_Toc422908609)

[Structured Streams 1](#_Toc422908610)

[Extracting and Extractors 2](#_Toc422908611)

[Outputting and Outputters 3](#_Toc422908612)

[StreamSets 3](#_Toc422908613)

# Structured Streams

Most of the examples in the Tutorial have focused on *unstructured* streams. Structured streams have additional metadata and indexing in them.

## Creating a Structured Stream

Let's take the **searchlog** unstructured stream and save it as a structured stream.

searchlog = VIEW @"Views/SearchLog.view";

OUTPUT searchlog TO SSTREAM "searchlog.ss"

CLUSTERED BY Region

SORTED BY Region;

PROTIP: The CLUSTERED BY and SORTED BY clauses are optional but we HIGHLY RECOMMEND that you ALWAYS use them when creating a structured stream. Their use is discussed in the [Scope Performance Guide](https://microsoft.sharepoint.com/teams/Cosmos/_layouts/15/WopiFrame.aspx?sourcedoc=%7b20525669-7A51-403E-BB84-44130A090112%7d&file=ScopePerformanceGuide.docx&action=default).

## Reading from a structured Stream

Now that we have a structured stream to use, let's read from it:

searchlog\_ss = SSTREAM "searchlog.ss";

searchlog =

SELECT \*

FROM searchlog\_ss

WHERE Region == "en-us";

OUTPUT searchlog TO "/my/Outputs/output.txt";

# Extracting and Extractors

## Filtering During Extraction

Filtering can be performed at the time of extraction using the **HAVING** clause. You don't need to create an intermediate rowset to do additional filtering. The **WHERE** clause is NOT supported for the **EXTRACT** operator.

rs0 =

EXTRACT

FirstName : string,

LastName : string,

Age : int

FROM

"/test\_input.tsv"

USING DefaultTextExtractor()

**HAVING Age > 40;**

However, in general we'd prefer you to write this as two statements.

rs0 =

EXTRACT

FirstName : string,

LastName : string,

Age : int

FROM

"/test\_input.tsv"

USING DefaultTextExtractor();

rs1 = SELECT \*

FROM rs0

**HAVING Age > 40;**

## Extracting CSV Files

**DefaultTextExtractor** by default handles TSV files, but it has optional arguments that let you specify the delimiter character.

departments =

EXTRACT

DepID : string,   
 DepName : string

FROM "/my/SampleData/departments.txt"

USING DefaultTextExtractor( delimiter: ',');

PROTIP: DefaultTextExtractor is **NOT** a full-fledged CSV file parser, since the CSV format permits special values with quotes. If you need to handle the full syntax, you will need a dedicated CSV extractor.

Learn more about **DefaultTextExtractor** here: <https://microsoft.sharepoint.com/teams/Cosmos/Wiki/DefaultTextExtractor.aspx>

## Extracting From Multiple Streams

If you have multiple streams with the same layout, you can **EXTRACT** from all of them at once by naming each stream in the **FROM** clause.

rs1 =

EXTRACT

A:string,

B:string,

C:string

FROM   
 "stream1.tsv",

"stream2.tsv",

"stream3.tsv"

USING DefaultTextExtractor();

# Outputting and Outputters

## Controlling DateTime format for DefaultTextOutputter

**DefaultTextOutputter** uses the "G" DateTime format by default (same as C# does). You can control the format used by using the "-datetime" argument.

OUTPUT TO "/my/test.txt"

USING DefaultTextOutputter( datetime: "o" )

# StreamSets

## The Motivation

Often streams are created so that their names have some implicit structure. For example, the streams may be numbered as shown below:

* log1.txt
* log2.txt
* log3.txt

In other cases, the streams may be named according to the date they were created:

* sales\_2013-03-31.txt
* sales\_2013-04-01.txt
* sales\_2013-04-02.txt

Going further in the sales example, people often create folders to organize their files so that not all the logs are in one folder.

* sales/2013/03/sales-03-31.txt
* sales/2013/04/sales2013-04-01.txt
* sales/2013/04/sales2013-04-02.txt

As shown below it is possible to use multiple streams in the FROM statement.

data = EXTRACT a:string, b:string

FROM "/my/SampleData/StreamSets/log1.txt" ,

"/my/SampleData/StreamSets/log2.txt" ,

"/my/SampleData/StreamSets/log3.txt"

USING DefaultTextExtractor();

This works, but there are several issues:

* What happens when there is a large number of files?
* What happens if you want to get a different range?
* The syntax above does not work with the SSTREAM keyword.

## The Solution

Cosmos has an even more powerful feature to help you deal with streams whose names follow a fixed pattern: **StreamSets**.

Before we explain the mechanics of StreamSets, let us see the code above transformed to use the StreamSets feature.

data =

EXTRACT

a:int,

b:string

FROM **STREAMSET "/my/SampleData/StreamSets/"**

**PATTERN "log%n.txt"**

**RANGE \_\_serialnum=["1", "3"]**;

StreamSets are a big topic. Here is a quick list of some of the features.

* StreamSets work on both structured and unstructured streams
* StreamSets can be **SPARSE** allowing them to skip over missing streams
* They support ranges such as
  + Integer ranges (via **\_\_serialnum**)
  + Hour ranges
  + Date Ranges
  + DateTime Ranges

To learn more read the [StreamSet manual](https://microsoft.sharepoint.com/teams/Cosmos/Documents/Scope/StreamSets.docx?d=we0f81827777c4127806d869291c75b06) or see the StreamSet content at <http://aka.ms/CosmosPresentations>