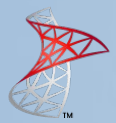


Microsoft® SQL Server®

Microsoft® Research

An Extensible Test Framework for Microsoft StreamInsight

- Alex Raizman
- Asvin Ananthanarayan
- Anton Kirilov
- Badrish Chandramouli
- Mohamed Ali

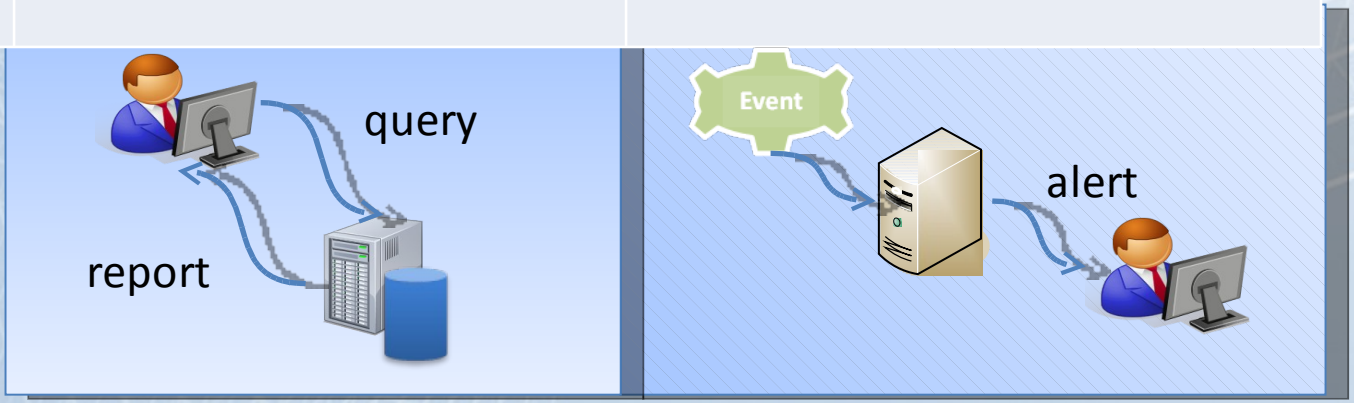
Agenda

- Introduction to Microsoft StreamInsight
- Testing Challenges
- End to End test framework
  - Intent based testing (Declarative testing)
  - Event generation
  - Functional Verification
- Case study – Top K
- Conclusions

What is Stream Processing?

Stream Processing is the continuous and incremental processing of event streams from multiple sources based on declarative query and pattern specifications **with near-zero latency**.

	RELATIONAL	STREAMING
Query Paradigm	Ad-hoc queries or requests	Continuous standing queries
Latency	Seconds, hours, days	Milliseconds or less
Data Rate	Hundreds of events/sec	Tens of thousands of events/sec or more
Temporal semantics	Optional	Integral to streaming queries



Events and Event Streams

- An event is a collection of fields
- Streaming engine provisioned timestamp fields capture all the different temporal event characteristics
- Event sources populate timestamp fields

Start Time	End Time	Long pumpID	String Location	Double flow	Double pressure
	…	…	…	…	…

- A stream is a sequence of events
  - Possibly infinite
- Stream characteristics:
  - Event/data arrival patterns (steady, bursty)
  - Out of order events: Order of arrival of events does not match the order of their application timestamps
  - Events with varying lifetimes

Relational Semantics Versus Streaming Semantics

Relational

- A join B on A.C1 = B.C1

A

C1	C2
1	2
2	3
4	5
8	10

B

C1	C2
1	7
12	15
4	10
8	12

A Join B

A.C1	A.C2	B.C2
1	2	7
4	5	10
8	10	12

- Top 2 rows ORDER BY C1 DESC

C1	C2
1	2
4	5
8	10
15	20
2	3
40	50

→

C1	C2
40	50
15	20

Streaming

- A join B on A.C1 = B.C1

A

S	E	C1	C2
20	30	1	2
30	40	2	3
40	50	4	5
50	60	8	10

B

S	E	C1	C2
15	25	1	7
30	40	12	15
35	38	4	10
45	70	8	12

A Join B

S	E	A.C1	A.C2	B.C2
20	25	1	2	7
50	60	8	10	12

← Predicate matches but events do not overlap in time

- Top 2 rows order by C1 DESC on a time window of size 10

S	E	C1	C2
1	5	1	2
2	8	4	5
7	12	8	10
18	30	2	3
25	35	40	50
9	15	15	20

Out of order event →

→

S	E	C1	C2
1	10	15	20
1	10	8	10
10	20	15	20
10	20	8	10
20	30	40	50
20	30	2	3

StreamInsight Query Examples

Example – JOIN, PROJECT, FILTER:

```
from e1 in MyStream1
join e2 in MyStream2
on e1.ID equals e2.ID
where e1.f2 == "foo"
select new { e1.f1, e2.f4 };
```

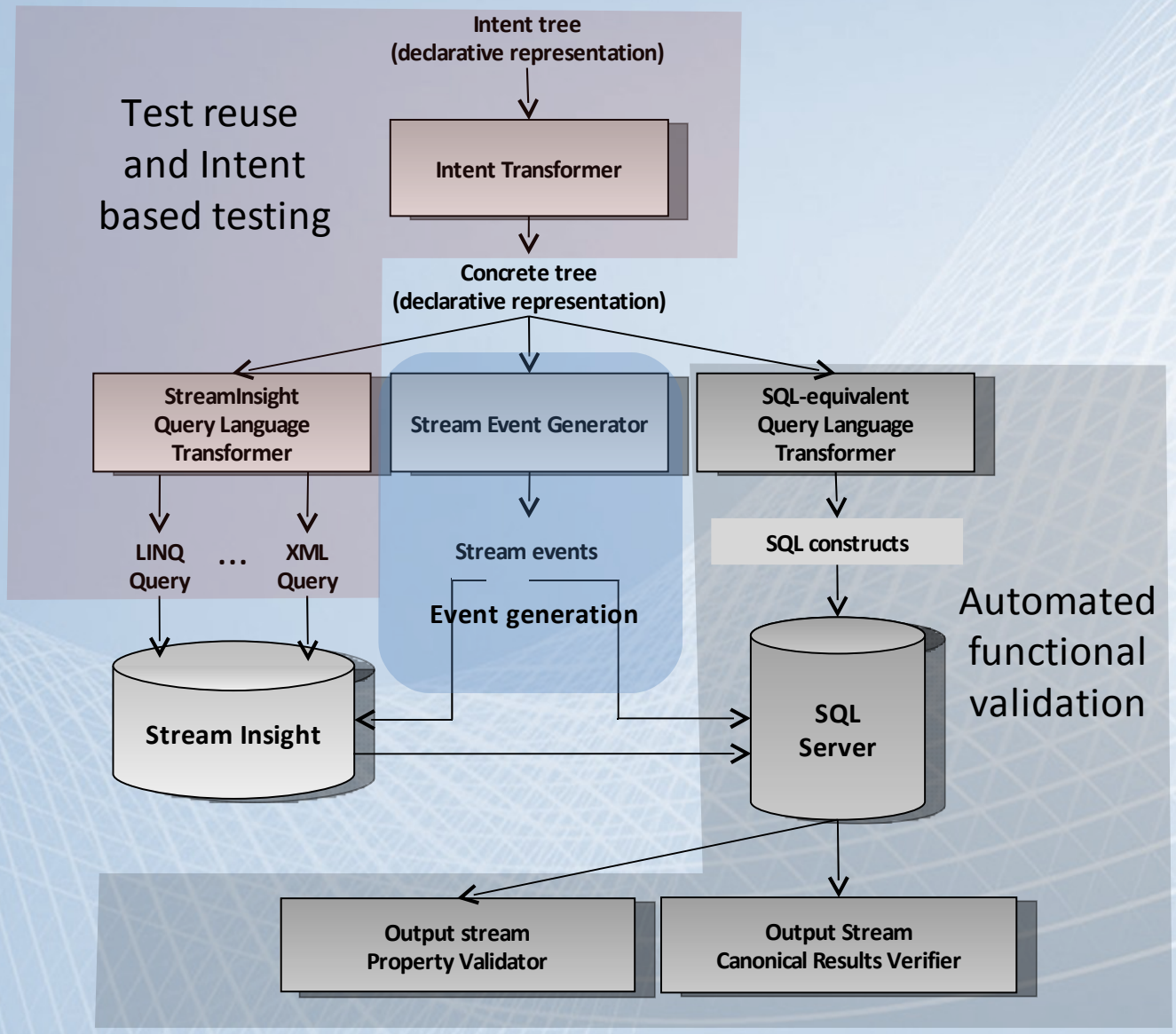
Example – GROUP&APPLY, WINDOW:

```
from e3 in MyStream3
group e3 by e3.i into SubStream
from win in SubStream.HoppingWindow(
                    TenSeconds)
            order by win.f).Take(10);
```

Testing Challenges

- Enable reuse of tests for different product languages
  - Multiple entry points to the query engine
- Event generation
  - Temporal characteristic.
  - More than payload generation
- Automated functional validation.
  - We have become very good at generating tests, but programmatic verification is still a challenge

Test Framework



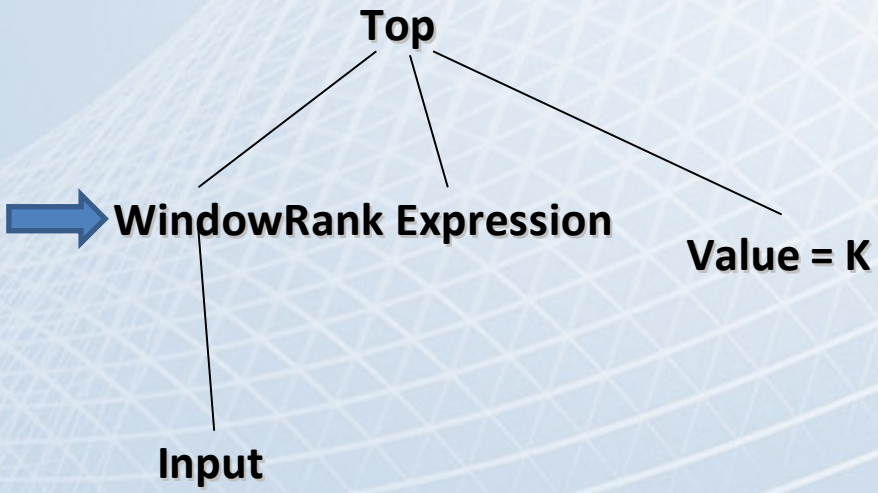
Intent based testing

- Top K events on a window with an descending rank expression
  - Must have an operator as input to Top K
  - Must have a rank expression.

- Builds an intent tree.

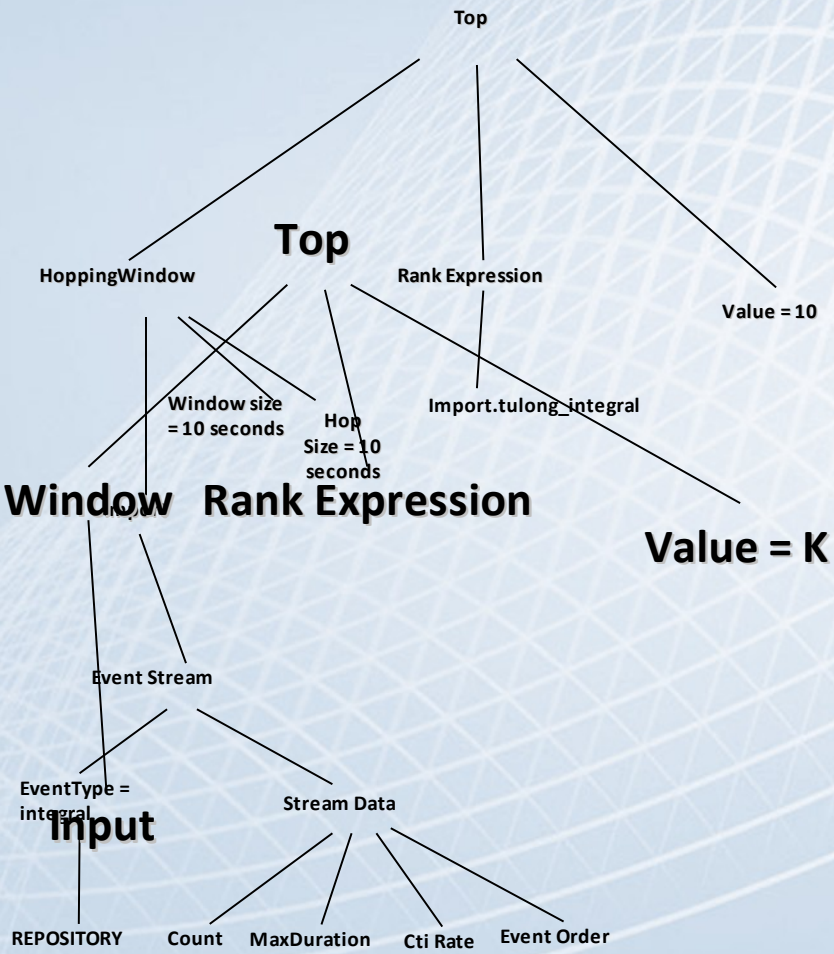
```
var rank = input.CreateExpression();

var topK = input
            .Window()
            .OrderBy(rank.Desc())
            .Top(K);
```



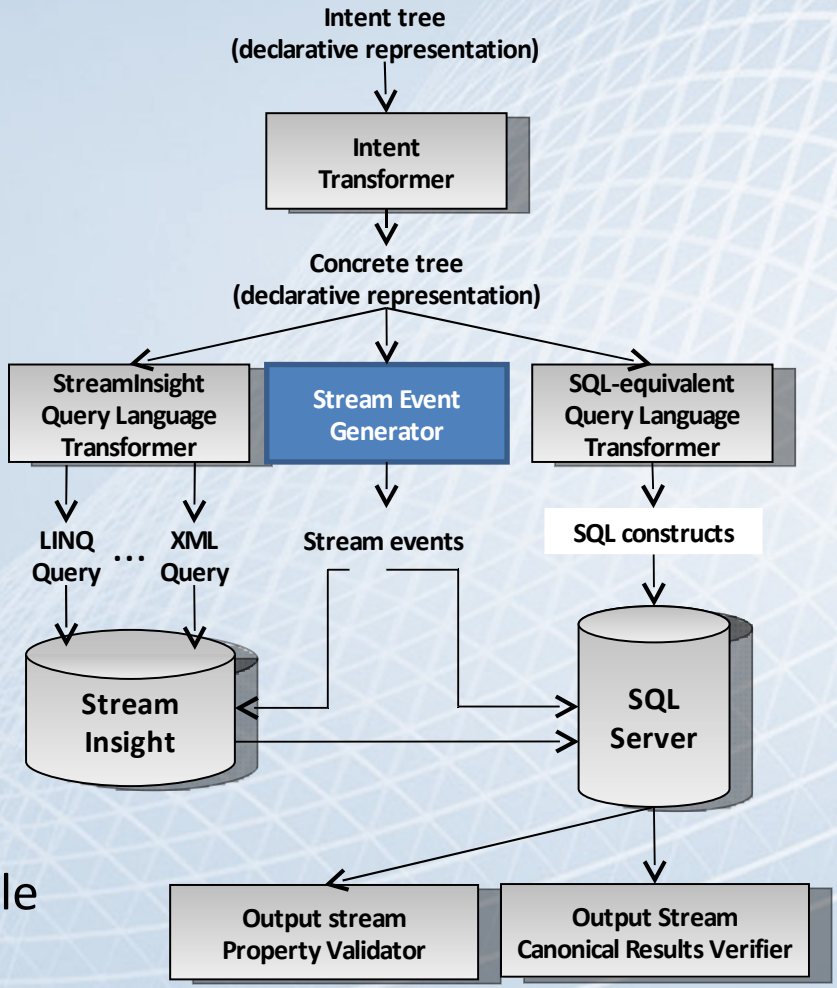
Intent tree to Concrete tree

- Building out the rest of the tree.
  - Finalize tree by binding to stream
  - Extend tree with another operator
  - Run model to generate subtree
- Repository of event types
- Expression builder
- Concrete tree = test
- Test suite = Multiple concrete trees



Stream Event Generator

- Two components
  - Temporal generation
  - Payload generation
- Supported temporal facets
  - event count
  - out of order
  - overlap
  - maximum duration of event
  - boundary cases
  - repository of pre defined patterns
- Payload generation
  - leveraged existing payload generators
- Tester specifies as many facets or as little as they want.



Query Language Transformer

- Concrete tree to XML transform generates xml language for the test

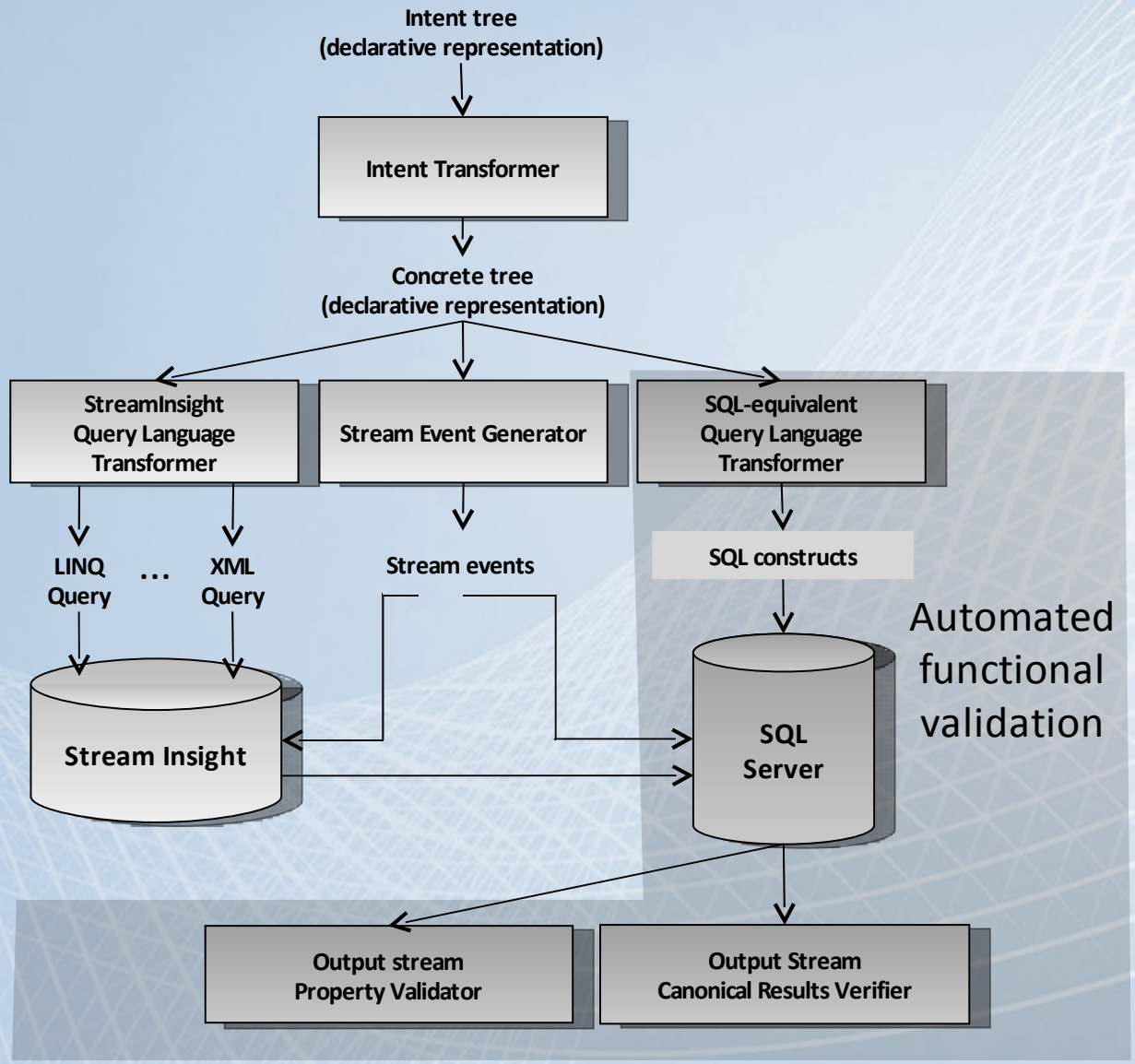
```
<QueryTemplate Name="cep:/Server/Application/CTP1DefaultApp/QueryTemplate/QT"
               xmlns="http://schemas.microsoft.com/ComplexEventProcessing/2010/01/Meta
  <Import Name="Import" Type="cep:/Server/Application/CTP1DefaultApp/EventType/integra
    <OutputStream Name="Import_stream" />
  </Import>
  <Export Name="Export">
    <!--Export result type is integral-->
    <InputStream Name="TopK_stream" />
  </Export>
  <TopK Name="TopK" RankDepth="10">
    <InputStream Name="Import_stream" />
    <OutputStream Name="TopK_stream" />
    <HoppingWindow>
      <WindowDefinition>
        <Size>PT10S</Size>
        <HopSize>PT10S</HopSize>
        <Alignment>0001-01-01T00:00:00Z</Alignment>
      </WindowDefinition>
      <InputPolicy>
        <Clip Left="true" Right="true" />
      </InputPolicy>
      <OutputPolicy>
        <Clip Type="WindowEnd" />
      </OutputPolicy>
    </HoppingWindow>
    <RankExpression Order="Descending">
      <InputField Name="tulong_integral" StreamName="Import_stream" />
    </RankExpression>
  </TopK>
</QueryTemplate>
```

- Concrete tree to LINQ transform generates C# code for the same test

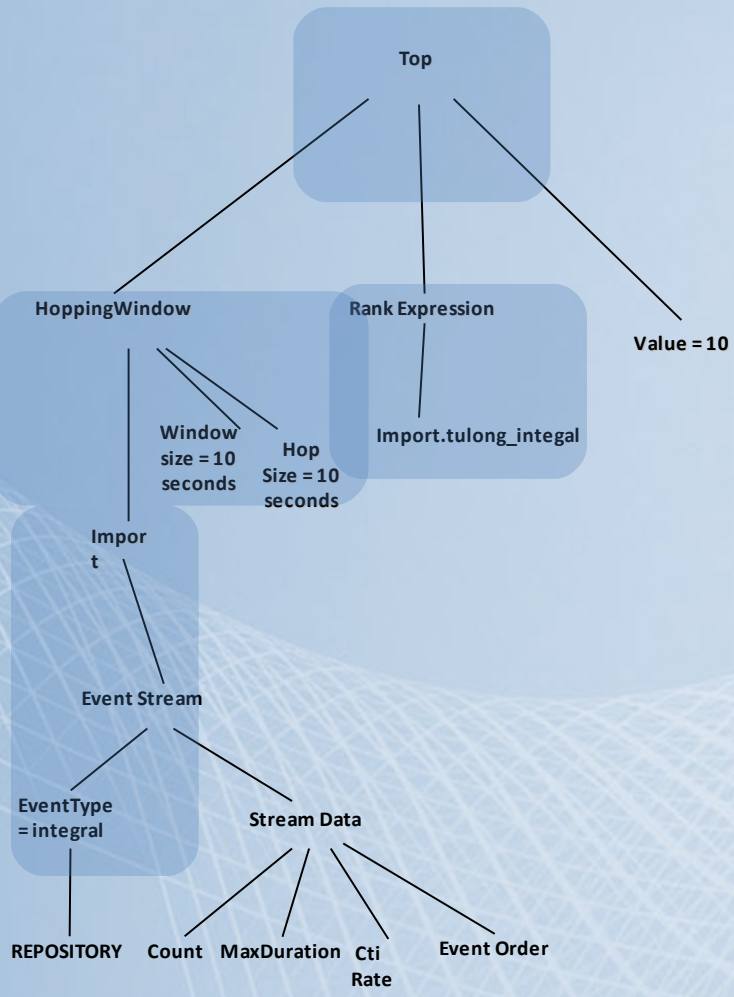
```
public static class QueryBuilder
{
    public static object CreateStream()
    {
        CepStream<integral>stream1 = CepStream<integral>.Create("Import");
        return (from window1 in stream1.HoppingWindow(
                    new TimeSpan(100000000),
                    new TimeSpan(100000000),
                    HoppingWindowOutputPolicy.ClipToWindowEnd
                    )
                from event1 in window1
                orderby event1.tulong_integral descending
                select event1).Take(10);
    }
}

public class integral
{
    public int tint_integral { get; set; }
    public long? tlong_integral { get; set; }
    public short? tshort_integral { get; set; }
    public uint tuint_integral { get; set; }
    public ulong? tulong_integral { get; set; }
    public ushort tushort_integral { get; set; }
}
```

Test Framework



Functional validation using SQL




```
using Microsoft.SqlServer.Test.ComplexEventProcessing.Common.StreamToSql.SqlClrTypes;
CREATE VIEW [QT_TopK_stream_4]
AS SELECT
CAExpr.EventType,
CAExpr.ValidStart as ValidStart,
ValidEnd = CASE
WHEN D.endSqlTime > CAExpr.ValidEnd THEN CAExpr.ValidEnd
WHEN D.endSqlTime < CAExpr.ValidEnd THEN D.endSqlTime
ELSE D.endSqlTime
END,NewValidEnd =
CASE
WHEN D.endSqlTime > CAExpr.ValidEnd THEN CAExpr.ValidEnd
WHEN D.endSqlTime < CAExpr.ValidEnd THEN D.endSqlTime
ELSE D.endSqlTime
END,tint_integral,tlong_integral,tshort_integral,
tuint_integral,tulong_integral,tushort_integral FROM [HoppingWindow]()
AS D
CROSS APPLY
(
SELECT TOP (10) WITH TIES
EventType,
startSqlTime AS ValidStart,
endSqlTime AS ValidEnd,
endSqlTime AS NewValidEnd,
tint_integral,tlong_integral,tshort_integral,tuint_integral,
tulong_integral,tushort_integral
FROM [Import_stream]  WHERE ValidStart < endSqlTime AND ValidEnd > startSqlTime
ORDER BY
[RankExpr]([Import_stream].tulong_integral).Data DESC
)
as CAExpr
```

```
    insert into @periodWindows values (@Vs, dbo.DateTime2AddTimeSpan(@Vs, '00:00:10').Data)
    SET @Vs = dbo.DateTime2AddTimeSpan(@Vs, '00:00:10').Data
end
return
end
```

Functional validation using SQL

- SQL input table is populated with the event stream from the generator
- Equivalent SQL Queries are run over the input table
- Output of the StreamInsight query is also piped into SQL Server using output adapter
- Compares SQL output with output from StreamInsight for equality.
- If comparison fails → BUG!

Conclusions

- Intent based testing allows testers to think about test scenarios leading to richer, more interesting tests.
- A relational database can work as a test oracle for validating streaming queries.
- Most of the bugs were due to interesting temporal aspects of the event stream.
- Not everything can / should be tested using intent based techniques.
  - Negative tests, boundary cases

QUESTIONS ?

For More Information

- StreamInsight main page & download : http://www.microsoft.com/sqlserver/2008/en/us/R2-comp
- StreamInsight blog: http://blogs.msdn.com/streaminsight/
- StreamInsight MSDN documentation: http://msdn.microsoft.com/en-us/library/ee362541(SQL.1(
- StreamInsight E-clinics on Microsoft e-learning https://www.microsoftelearning.com/eLearning