Agenda

- Introduction on Stream Processing Models [done]
- Declarative Language: Opportunities, and Design Principles [done]
- Comparison of Prominent Streaming SQL Dialects for Big Stream Processing Systems
- Conclusion

Our Focus

- Prominent Big Stream Processing Engines that offer a declarative SQL-like interface.
  - Flink,
  - Spark Structured Streaming, and
  - Kafka SQL (KSQL)

Flink SQL

- Available since version 1.3, it builds on Flink Table API (LINQ-style API)
- Uses Apache Calcite of parsing, interpreting and planning, while execution relies on FLINK Runtime.
- **Relevant concepts**: windows as group-by function, temporal tables, match-recognize (not today)

Spark Structured Streaming

- Available since Spark 2.0, it extends Dataframe and Datasets to Streaming Datasets
- SQL-like programming interface that relies on Catalyst for optimization
- **Relevant Concepts**: Complete, Append, and Update modes/

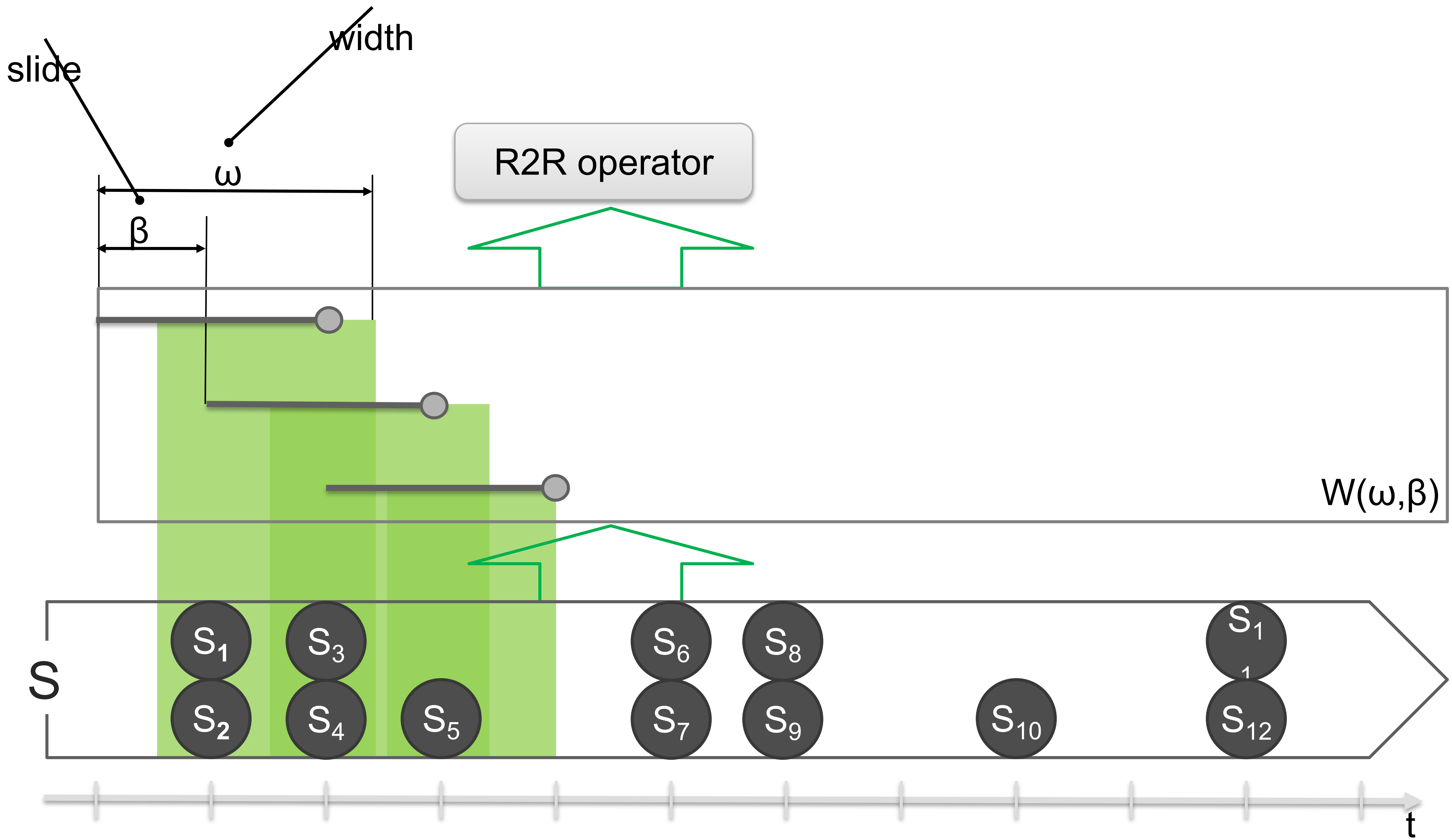
Kafka SQL (KSQL)

- Available since Kafka 1.9/2 (or confluent platform 5)
- builds directly on top of KStreams Library
- **Relevant Concepts:** simplicity is the key, relation (compacted) topic vs table/stream

Time-Window Operators & Aggregates

- Sliding Window
- Tumbling Window
- Session Window
- Aggregations: COUNT, SUM, AVG, MEAN, MAX, MIX

Sliding/Hopping Window



KSQL Hopping Window

DDL Extension

Aggregat
e

Window From Function

CREATE TABLE analysis AS
SELECT nation, **COUNT**(*)
FROM pageviews
WINDOW HOPPING (**SIZE** 30 SECONDS, **ADVANCE BY** 10 SECONDS)
GROUP BY nation;

Results

SELECT * **FROM** analysis

```
1561375069212 | Page_66 : Window{start=1561375050000 end=-} | Page_66 | 1
1561375069311 | Page_11 : Window{start=1561375050000 end=-} | Page_11 | 1
1561375073332 | Page_33 : Window{start=1561375050000 end=-} | Page_33 | 1
1561375077242 | Page_32 : Window{start=1561375050000 end=-} | Page_32 | 1
1561375080706 | Page_55 : Window{start=1561375080000 end=-} | Page_55 | 1
1561375082825 | Page_34 : Window{start=1561375080000 end=-} | Page_34 | 1
1561375085084 | Page_56 : Window{start=1561375080000 end=-} | Page_56 | 1
1561375086275 | Page_85 : Window{start=1561375080000 end=-} | Page_85 | 1
1561375086905 | Page_20 : Window{start=1561375080000 end=-} | Page_20 | 1
1561375094475 | Page_27 : Window{start=1561375080000 end=-} | Page_27 | 1
```

Flink SQL Hopping Window

Window helper functions

Aggregate

Group By Function

SELECT nation, **COUNT**(*) , HOP_START(..) HOP_END(...)
FROM pageviews
GROUP BY **HOP**(rowtime, **INTERVAL** 1H, **INTERVAL** 1M), nation

Results

```
1> (Egypt,2019-06-24 11:38:00.0,2019-06-24 11:38:01.0,1)
1> (Egypt,2019-06-24 11:39:00.0,2019-06-24 11:39:01.0,1)
1> (Egypt,2019-06-24 11:40:00.0,2019-06-24 11:40:01.0,1)
1> (Egypt,2019-06-24 11:41:00.0,2019-06-24 11:41:01.0,1)
2> (Italy,2019-06-24 11:42:00.0,2019-06-24 11:42:01.0,1)
2> (Italy,2019-06-24 11:43:00.0,2019-06-24 11:43:01.0,1)
2> (Italy,2019-06-24 11:44:00.0,2019-06-24 11:44:01.0,1)
2> (Italy,2019-06-24 11:45:00.0,2019-06-24 11:45:01.0,1)
2> (Italy,2019-06-24 11:46:00.0,2019-06-24 11:46:01.0,1)
2> (Italy,2019-06-24 11:47:00.0,2019-06-24 11:47:01.0,1)
2> (Italy,2019-06-24 11:48:00.0,2019-06-24 11:48:01.0,1)
3> (Estonia,2019-06-24 11:49:00.0,2019-06-24 11:49:01.0,1)
```

....

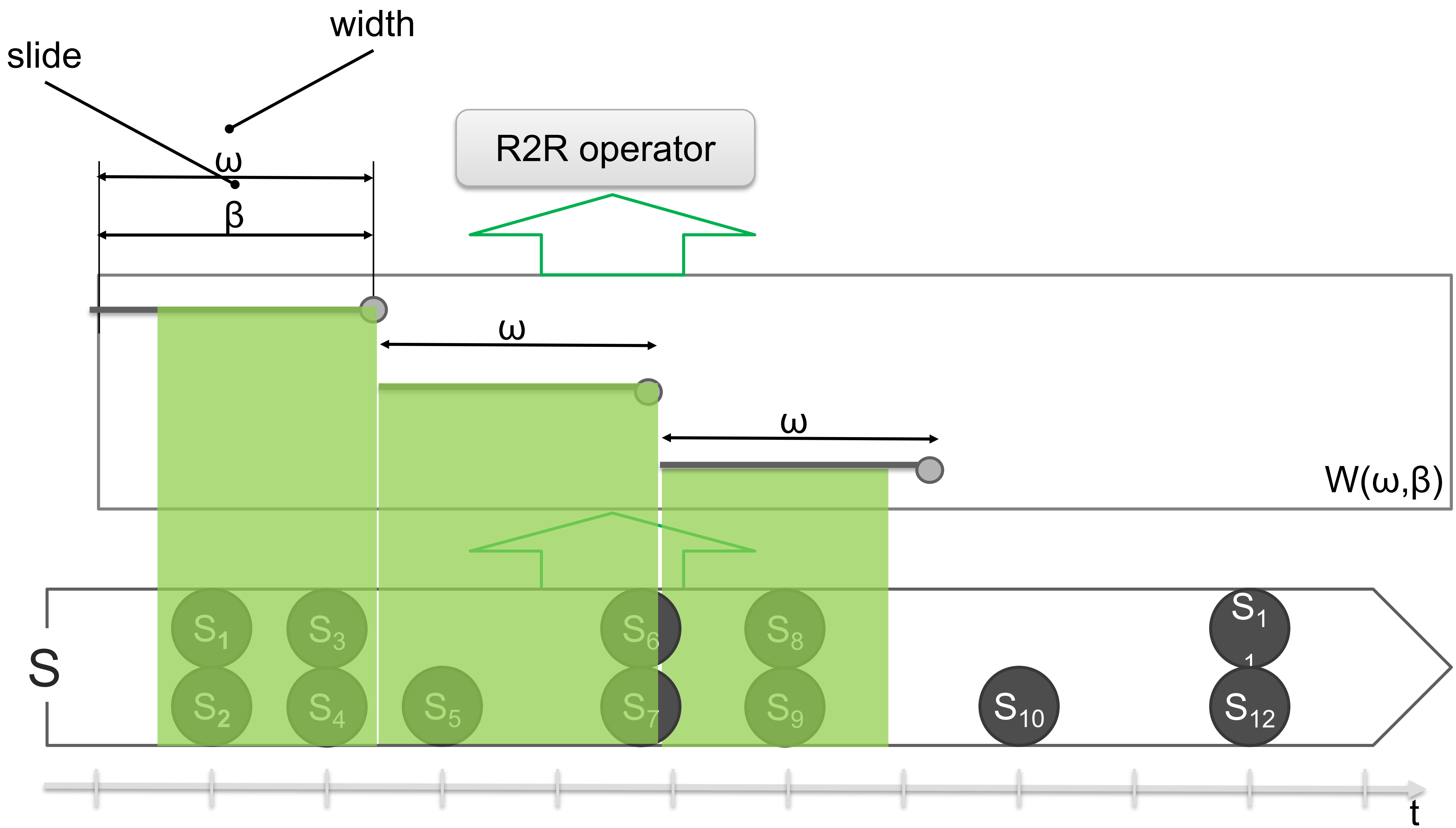
Spark Structured Streaming Hopping Window

Window operator

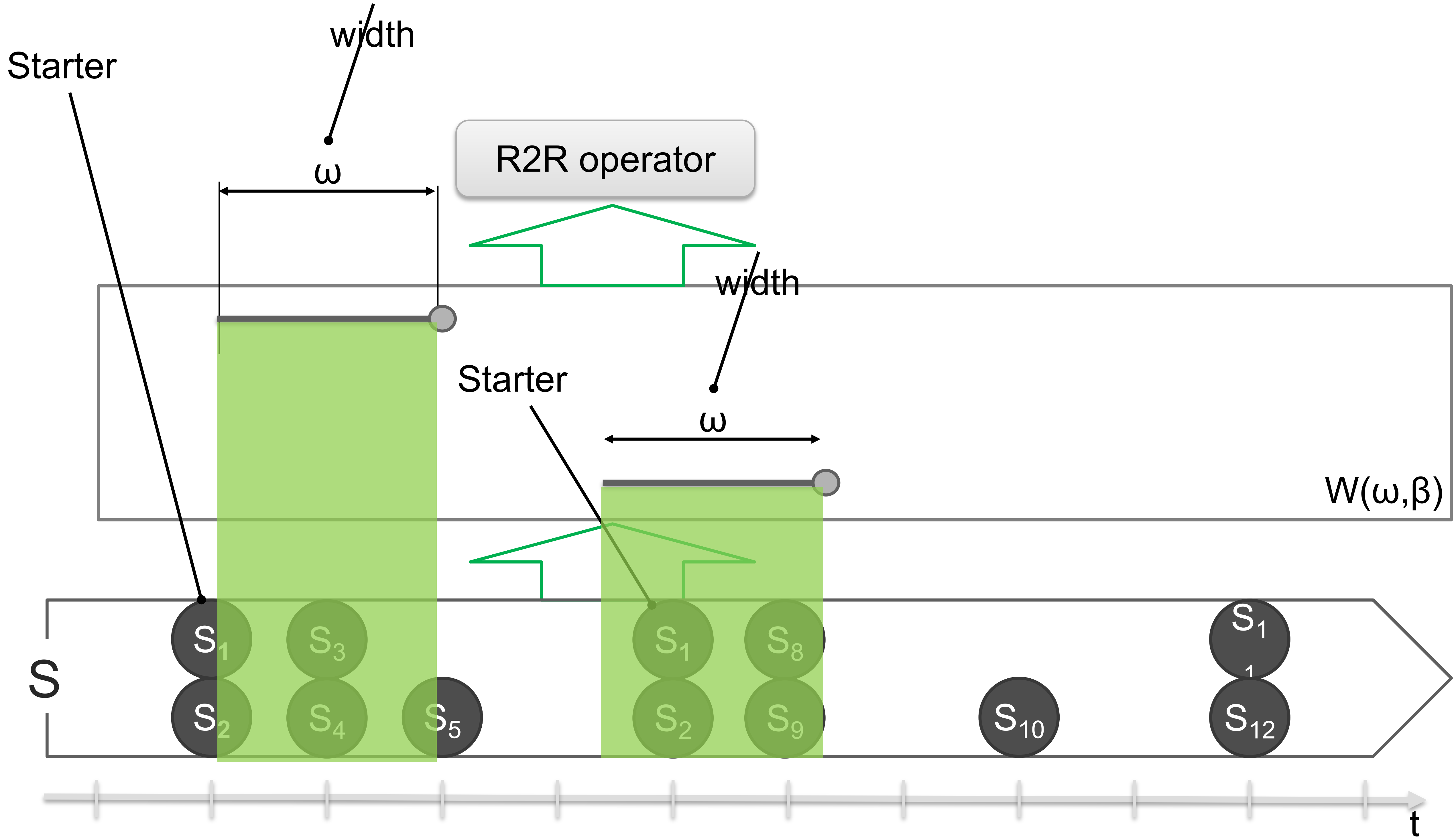
Aggregate

```
val df = pageviews
.groupBy(
window($"timestamp", "1 hour", "1 minute”), $"nation").count()
```

Tumbling Window



Session Window



KSQL Session

DDL Extension

Aggregat
e

CREATE TABLE analysis **AS**
SELECT nation, **COUNT** (*),
TIMESTAMPTOSTRING(windowstart(), 'yyyy-MM-dd HH:mm:ss') **AS** window_start_ts,
TIMESTAMPTOSTRING(windowend(), 'yyyy-MM-dd HH:mm:ss') **AS** window_end_ts

Window From Function

FROM pageviews **WINDOW SESSION** (1 **MINUTE**)
GROUP BY nation;

Results

Page_82 | 2019-06-24 11:47:45 | 2019-06-24 11:47:45 | 1
Page_73 | 2019-06-24 11:47:46 | 2019-06-24 11:47:46 | 1
Page_16 | 2019-06-24 11:47:49 | 2019-06-24 11:47:49 | 1
Page_54 | 2019-06-24 11:47:25 | 2019-06-24 11:47:53 | 2
Page_68 | 2019-06-24 11:47:55 | 2019-06-24 11:47:55 | 1
Page_25 | 2019-06-24 11:47:40 | 2019-06-24 11:47:58 | 2
Page_17 | 2019-06-24 11:47:59 | 2019-06-24 11:47:59 | 1
Page_92 | 2019-06-24 11:48:02 | 2019-06-24 11:48:02 | 1
Page_83 | 2019-06-24 11:48:05 | 2019-06-24 11:48:05 | 1
Page_37 | 2019-06-24 11:48:06 | 2019-06-24 11:48:06 | 1
Page_86 | 2019-06-24 11:48:07 | 2019-06-24 11:48:07 | 1

Flink SQL Session

Aggregat
e

Custom Window Helper Functions

Group By Function

SELECT nation, count(*),
SESSION_START(...), **SESSION_ROWTIME**(...)
FROM pageviews
GROUP BY SESSION(rowtime, **INTERVAL** 1M), nation

Results

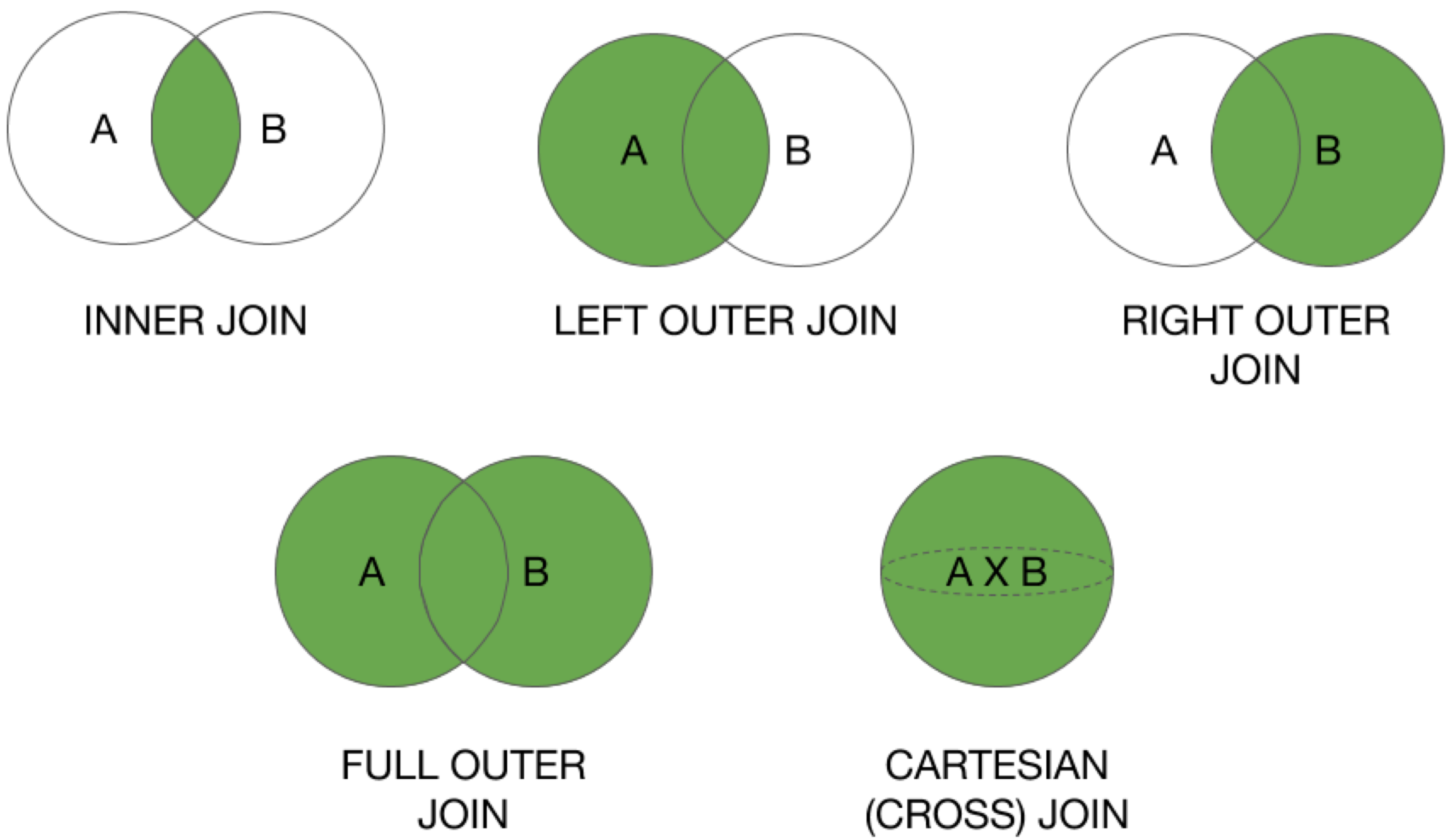
```
3> (Estonia,1,2019-06-24 11:52:55.538,2019-06-24 11:52:56.538,2019-06-24 11:52:56.537)
2> (Italy,1,2019-06-24 11:52:56.132,2019-06-24 11:52:57.132,2019-06-24 11:52:57.131)
1> (Egypt,1,2019-06-24 11:52:56.633,2019-06-24 11:52:57.633,2019-06-24 11:52:57.632)
3> (Estonia,1,2019-06-24 11:52:57.136,2019-06-24 11:52:58.136,2019-06-24 11:52:58.135)
2> (Italy,1,2019-06-24 11:52:57.64,2019-06-24 11:52:58.64,2019-06-24 11:52:58.639)
1> (Egypt,1,2019-06-24 11:52:58.141,2019-06-24 11:52:59.141,2019-06-24 11:52:59.14)
3> (Estonia,1,2019-06-24 11:52:58.643,2019-06-24 11:52:59.643,2019-06-24 11:52:59.642)
2> (Italy,1,2019-06-24 11:52:59.147,2019-06-24 11:53:00.147,2019-06-24 11:53:00.146)
1> (Egypt,1,2019-06-24 11:52:59.648,2019-06-24 11:53:00.648,2019-06-24 11:53:00.647)
3> (Estonia,1,2019-06-24 11:53:00.152,2019-06-24 11:53:01.152,2019-06-24 11:53:01.151)
2> (Italy,1,2019-06-24 11:53:00.653,2019-06-24 11:53:01.653,2019-06-24 11:53:01.652)
1> (Egypt,1,2019-06-24 11:53:01.158,2019-06-24 11:53:02.158,2019-06-24 11:53:02.157)
```

Recap

	Landmark	Tumble	Hop	Session	Aggregates
KSQL	implicit	✓	✓	✓	Standard SQL
Flink SQL	implicit	✓	✓	✓	Standard SQL
SSS	implicit	✓	X	X	Standard SQL

Table 1: Time-Based Window Operators and aggregates across different systems.

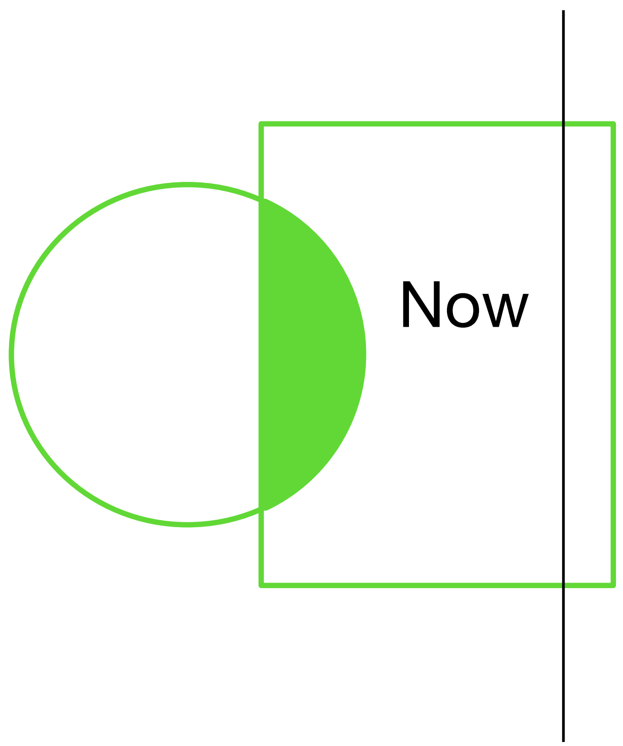
Recap on RA JOINS



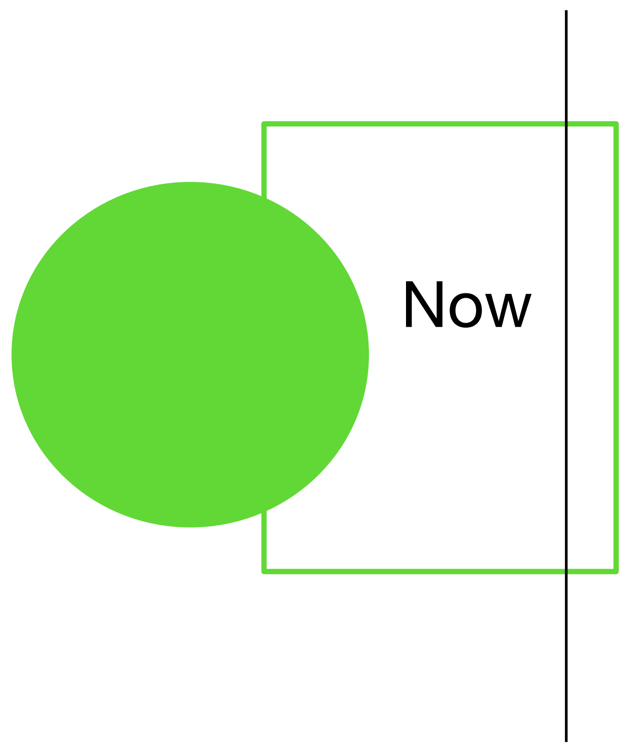
Stream-Table Joins

- Inner Joins
- Left-Outer Join
- Right-Outer Join
- Full-Outer Join

Stream-Table Joins



Inner



Left Outer

KSQL Left-Join

DDL Extension

Stream-Table Join

```
CREATE STREAM SENSOR_ENRICHED AS
SELECT S.SENSOR_ID, S.READING_VALUE, I.ITEM_ID
FROM SENSOR_READINGS S
  LEFT JOIN ITEMS_IN_PRODUCTION I ON
S.LINE_ID=I.LINE_ID;
```

Flink SQL LEFT-JOIN

Stream-Table Join

SELECT S.SENSOR_ID, S.READING_VALUE, I.ITEM_ID
FROM SENSOR_READINGS S **LEFT JOIN**
ITEMS_IN_PRODUCTION I **ON** S.LINE_ID=I.LINE_ID;

Results

```
4> (true,0,10.12666825646483,0)
4> (true,0,10.96399203326454,0)
1> (true,2,10.874856720766067,2)
4> (true,0,10.268731915130621,0)
1> (true,2,10.786008348182463,2)
4> (true,1,10.360322470661394,1)
4> (true,0,10.809087822653261,0)
4> (true,1,10.238883138171406,1)
1> (true,2,10.776781799073452,2)
4> (true,1,10.528528144000497,1)
4> (true,0,10.532966430120872,0)
4> (true,1,10.449756056124912,1)
4> (true,1,10.66021657541424,1)
```

Spark Structured Streaming LEFT-JOIN

Table

```
val itemsInProduction = spark.read. ...

val sensorReadings = spark.readStream. ...

val enrichedSensorReadings =
sensorReadings.join(itemsInProduction, "LINE_ID", "left-join")
```

Stream-Table Join

Recap

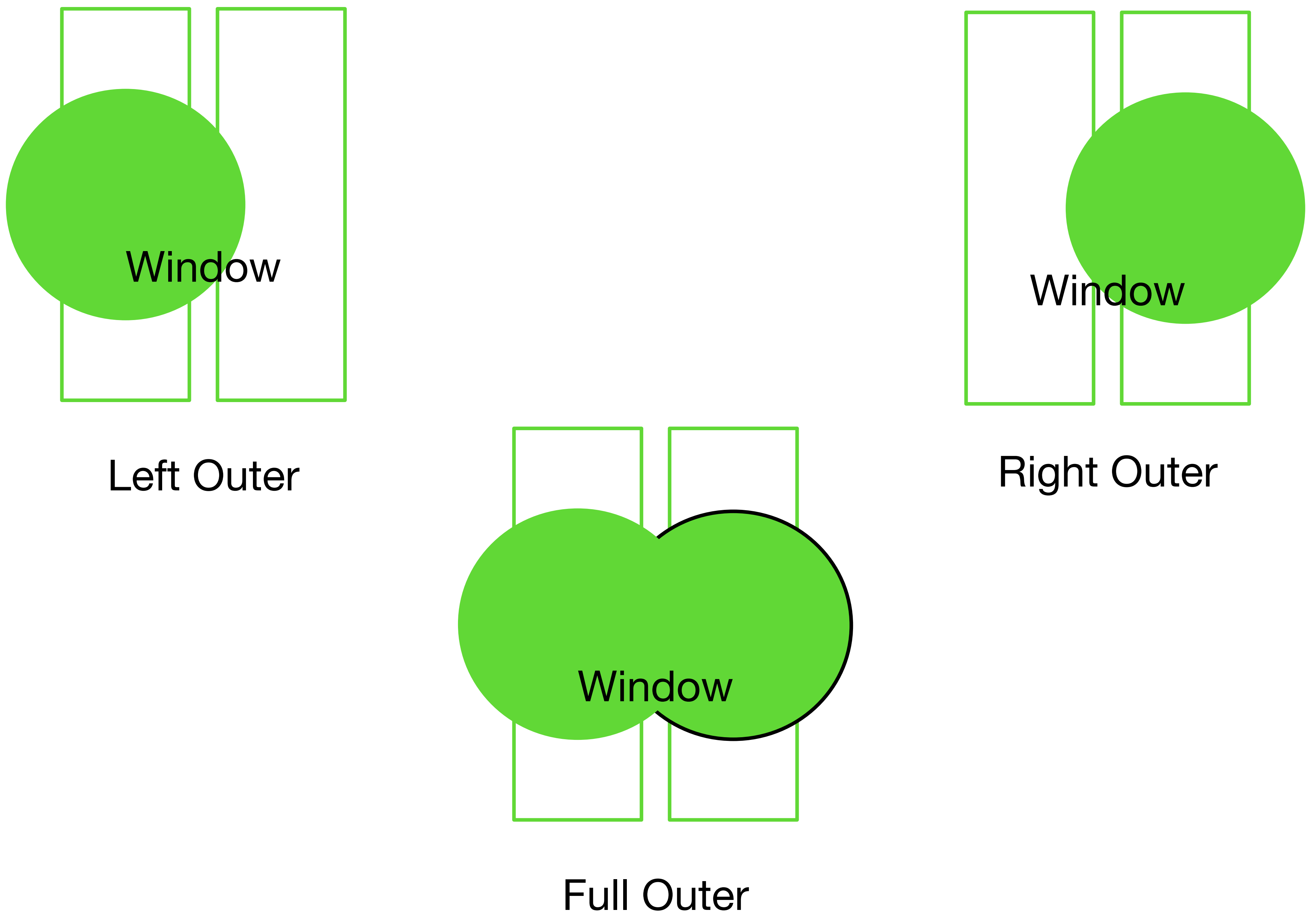
	Inner	Left Outer	Right Outer	Full Outer
KSQL	S	S	NS	NS
Flink SQL	S	S*	S*	S*
SSS	S, Stateless	S, Stateless	NS	NS

Table 2: Stream-Static Joins. [S]upported, [N]ot[S]upported. S*, Flink memory usage might grow indefinitely, Temporal Tables can be used to avoid it.

Stream-Stream Joins

- Inner Joins
- Left-Outer Join
- Right-Outer Join
- Full-Outer Join

Stream-Table Joins



Flink SQL Inner Join

SELECT * FROM IMPRESSIONS, CLICKS
WHERE IMPRESSION_ID = CLICK_ID **AND**
CLICK_TIME **BETWEEN** IMPRESSION_TIME - **INTERVAL**
'1' HOUR **AND** IMPRESSION_TIME

Spark Structured Streaming Inner Join

val impressions = spark.readStream. ...

val clicks = spark.readStream. ...

// Apply watermarks on event-time columns

val imprWithWtmrk =impressions.withWatermark("impressionTime", "2 hours")

val clicksWithWatermark =

clicks.withWatermark("clickTime", "3 hours")

Val imprWithWtmrk.join(clicksWithWatermark, expr(""" clickAdId = impressionAdId AND clickTime >= impressionTime AND clickTime <= impressionTime + interval 1 hour"""))

Recap

	Inner	Left Outer	Right Outer	Full Outer
KSQL	S, win	S, win	S, win	S, win
Flink SQL	S	S*	S*	S*
SSS	S	S + w on left	S + w on right	NS

Table 3: Stream-Stream Joins. [S]upported, [N]ot[S]upported, [W]atermark, [Win]dow. S*, Flink memory usage might grow indefinitely.

Agenda

- Introduction on Stream Processing Models [done]
- Declarative Language: Opportunities, and Design Principles [done]
- Comparison of Prominent Streaming SQL Dialects for Big Stream Processing Systems [done]
- Conclusion

DEMO

KSQL and Flink

survey of spark structured streaming notebook