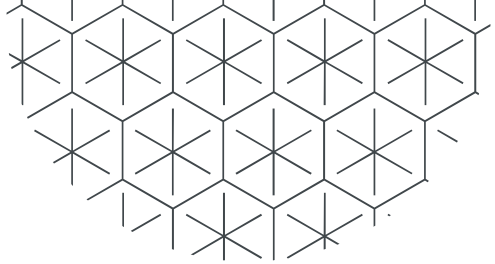
Tecnologías de datos masivos

Doble Grado en Ingeniería en Tecnologías de Telecomunicación y Business Analytics



Structured Streaming

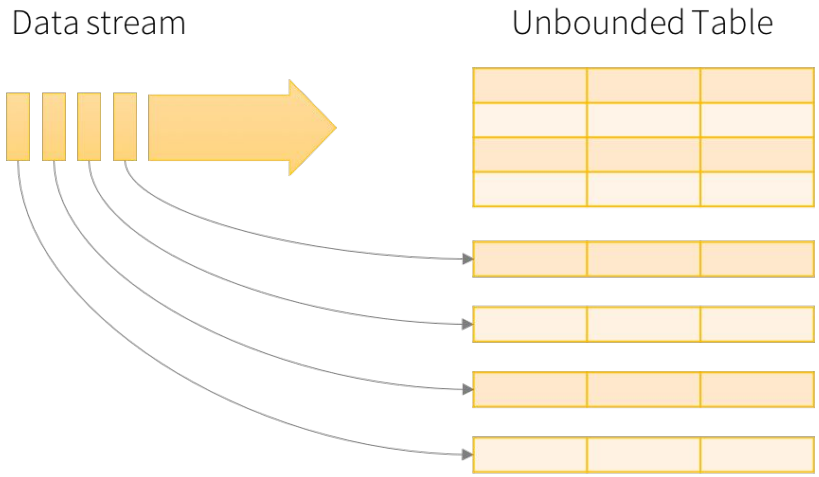
StructuredStreaming - Introducción

- A partir de la versión 2.0 el objetivo de Spark es migrar la mayor parte de sus procesos a SQL
- En el caso de la casuística de streaming adaptar el flujo continuo a una tabla de base de datos
- El enfoque cambia ya nos pregunta cada cuanto cogemos generar DStream si no cada cuanto queremos volver a guardar en la tabla
- Se hace mediante el método Trigger y el tiempo por defecto es “lo antes posible” llegando a un máximo de 1 milisegundo entre queries

StructuredStreaming - Introducción

- Otro de los objetivos de SparkSQL es que la forma de operar con los datos sea prácticamente similar independientemente del caso de uso
- La forma de interactuar con los datos en StructuredStreaming es muy similiar a la forma de interactuar con **SparkSQL**
- Al igual que en **SparkStreaming**, Structured Streaming seguirá lanzando un conjunto infinito de queries
- Dichas queries irán alimentando una tabla infinita que será la consultada continuamente
- Las acciones de **StructuredStreaming** usarán un determinado número de registros dependiendo del paramentro **outputMode**

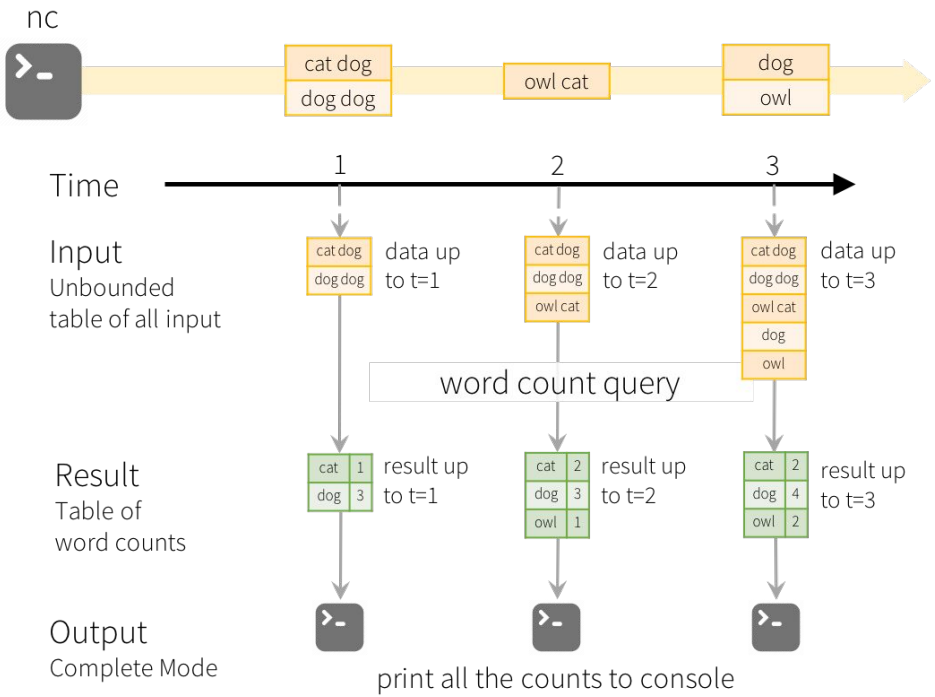
StructuredStreaming - Introducción



new data in the
data stream
=
new rows appended
to a unbounded table

Data stream as an unbounded table

StructuredStreaming - Introducción



Model of the Quick Example

StructuredStreaming - Introducción

Leer un directorio CSV desde SparkSQL

```
>>> clients = spark.read.option(“delimiter”, “;”).option(“header”, “true”).csv(“clients”)
```

StructuredStreaming - Introducción

Leer un directorio CSV desde StructuredStreaming

```
>>> clients = spark.read.option(“delimiter”, “;”).option(“header”, “true”).csv(“clients”)

>>> from pyspark.sql.types import StringType, StructField, StructType

>>> schema = StructType([ StructField("userId", StringType(), True),StructField("Origin",
StringType(), True),StructField("Destination", StringType(), True),StructField("Date",
StringType(), True)])

>>> flights_in_streaming = spark.readStream.option("maxFilesPerTrigger",
1).option("header", True).schema(schema).csv(“flights”)
```

StructuredStreaming - Introducción

Leer un topic de Kafka desde StructuredStreaming

```
>>> clients = spark.read.option(“delimiter”, “;”).option(“header”, “true”).csv(“clients”)

>>> from pyspark.sql.types import StringType, StructField, StructType

>>> schema = StructType([ StructField("userId", StringType(), True),StructField("Origin",
StringType(), True),StructField("Destination", StringType(), True),StructField("Date",
StringType(), True)])

>>> flights_in_streaming = spark.readStream.option("maxFilesPerTrigger",
1).option("header", True).schema(schema).csv(“flights”)

>>> kafka_df = spark.readStream.format("kafka").option("kafka.bootstrap.servers",
"host1:port1").option("subscribe", "topic1").load()
```

StructuredStreaming - Introducción

Seleccionar dos campos en un **Dataframe** de **SparkSQL**

```
>>> clients_id_age = clients.select(“customerId”,”age”)
```

StructuredStreaming - Introducción

Seleccionar dos campos en un **Dataframe** de **StructuredStreaming**

```
>>> clients_id_age = clients.select(“customerId”,”age”)

>>> flights_customer_origin = flights_in_streaming.select(“customerId”,”origin”)
```

StructuredStreaming - Introducción

Contar el número de registros por una clave en un **Dataframe** de **SparkSQL**

```
>>> clients_id_count = clients.groupBy(“customerId”).count()
```

StructuredStreaming - Introducción

Contar el número de registros por una clave en un **Dataframe** de **StructuredStreaming**

```
>>> clients_id_count = clients.groupBy(“customerId”).count()

>>> flights_customer_id_count = flights_in_streaming.groupBy(“customerId”).count()
```

StructuredStreaming - Introducción

Mostrar los 5 primeros registros de un **Dataframe** de **SparkSQL**

```
>>> clients.show(5)

+----------+-----+
|customerId|count|
+----------+-----+
|     00022|    1|
|     00001|    1|
|     00044|    1|
|     00055|    1|
|     00066|    1|
+----------+-----+
```

StructuredStreaming - Introducción

Mostrar los 5 primeros registros cada 5 segundos de un **Dataframe** de **StrucutedStreaming**

```
>>>flights_customer_id_count.writeStream.format("console").queryName("counts_N").option("nu
mRows",5).trigger(processingTime='5 seconds').outputMode("complete").start()

-------------------------------------------
Batch: 0
-------------------------------------------
+----------+-----+
|customerId|count|
+----------+-----+
|     00022|    2|
|     00001|    1|
|     00044|    5|
|     00055|    7|
|     00066|    2|
+----------+-----+

only showing top 5 row
```

StructuredStreaming - Introducción

Mostrar los 5 primeros registros cada 5 segundos de un **Dataframe** de **StrucutedStreaming**

```
>>>flights_customer_id_count.writeStream.format("console").queryName("counts_N").option("nu
mRows",1).trigger(processingTime='2 seconds').outputMode("complete").start()
-------------------------------------------
Batch: 1
-------------------------------------------
+----------+-----+
|customerId|count|
+----------+-----+
|     00033|    2|
|     00045|    4|
|     00034|  150|
|     00055|    7|
|     00066|    2|
+----------+-----+
only showing top 5 row
WARN ProcessingTimeExecutor: Current batch is falling behind. The trigger interval is 5000
milliseconds, but spent 10970 milliseconds
```

StructuredStreaming - Introducción

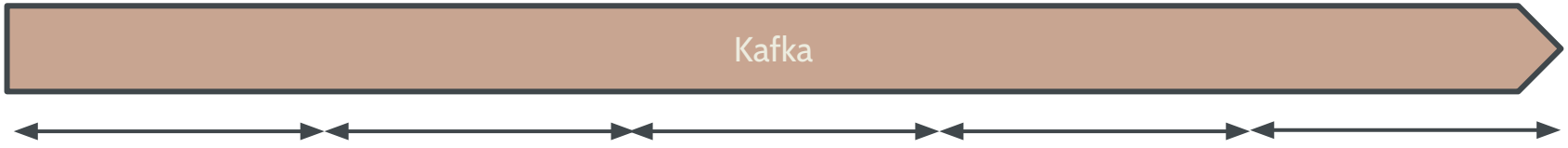
Mostrar los 5 primeros registros cada 5 segundos de un join entre un **Dataframe** de **StrucutedStreaming y** otro de **SparkSQL**

```
>>>flights_customer_id_count.join(client.witColumnRenamed(“customerId”,“customerId_client”)
col(“customerId”)==col(“customerId_client”)).groupBy(“age”).count().writeStream.format("con
sole").queryName("counts_N").option("numRows",2).trigger(processingTime='5
seconds').outputMode("complete").start()

-------------------------------------------
Batch: 0
-------------------------------------------
+----------+-----+
|customerId|count|
+----------+-----+
|        21|    2|
|        35|    4|
+----------+-----+

only showing top 2 row
```

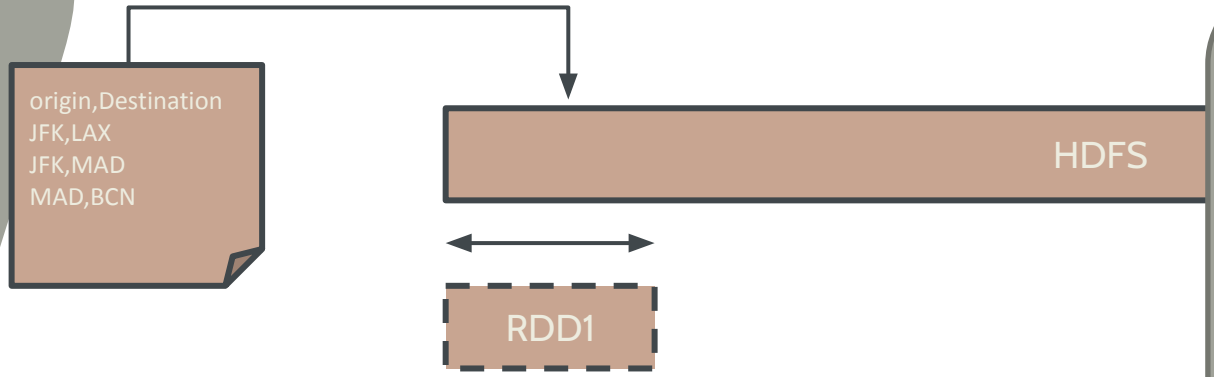
StructuredStreaming - outputMode

- **StrucutedStreaming** puede seleccionar los datos que van a terminar siendo procesados
- Se determina mediante la propiedad **ouputMode**
- Hay tres posibles valores:
    - **Append:** Sólo usará los registros recibidos en la última query(micro-batch)
    - **Update:** Sólo usará los registros con datos actualizados desde la última query(si no hay una agregación tiene el mismo comportamiento que el modo **append)**
    - **Complete:** Siempre se mostrarán todos los registros de la tabla infinita(sólo disponible si hay una agregación)

StructuredStreaming - outputModes

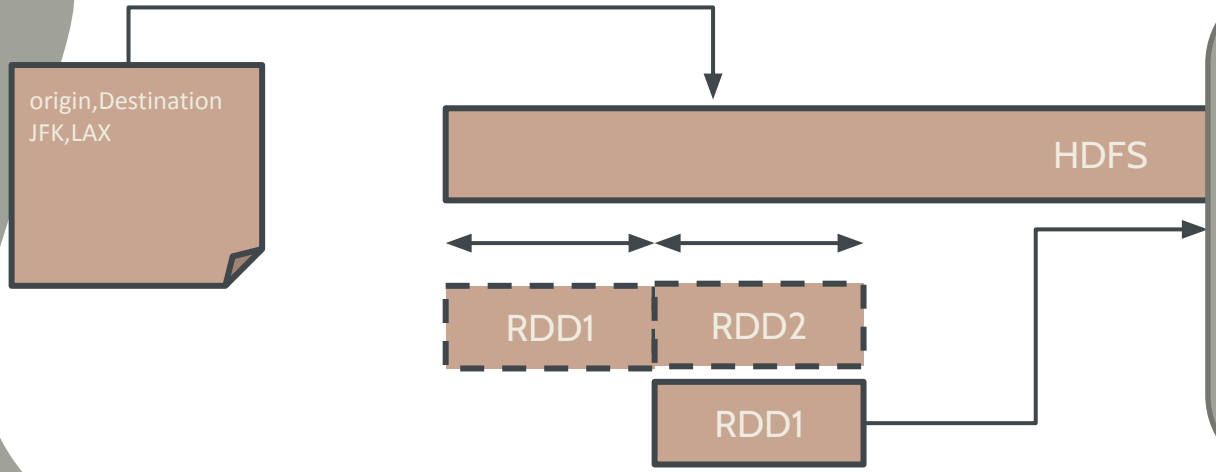


StructuredStreaming - outputModes



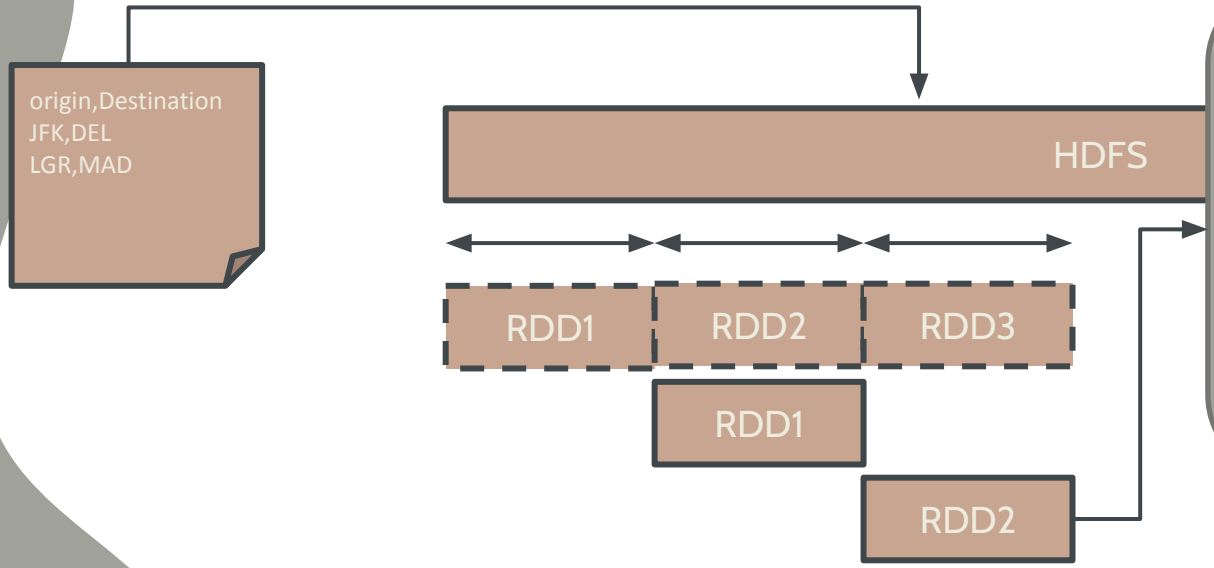

```
>>>flights.writeStream.format("console").query
Name("fligths").trigger(processingTime='2
seconds').outputMode("append").start()
```

StructuredStreaming - outputModes



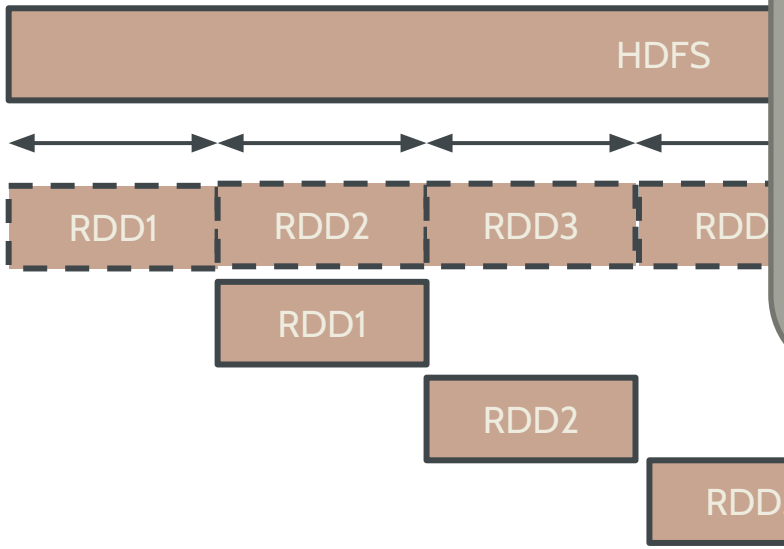

```
>>>flights.writeStream.format("console").query
Name("fligths").trigger(processingTime='2
seconds').outputMode("append").start()
-------------------------------------------
Batch: 0
-------------------------------------------
+------+-----------+
|Origin|Destination|
+------+-----------+
|   JFK|        LAX|
|   JFK|        MAD|
|   MAD|        BCN|
+------+-----------+
```

StructuredStreaming - outputModes



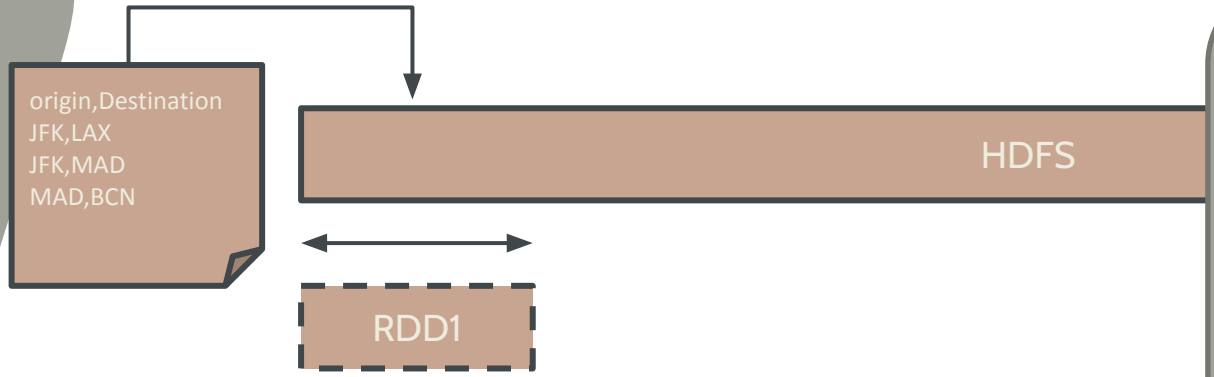

```
>>>flights.writeStream.format("console").query
Name("fligths").trigger(processingTime='2
seconds').outputMode("append").start()
-------------------------------------------
Batch: 2
-------------------------------------------
+------+-----------+
|Origin|Destination|
+------+-----------+
|   JFK|        LAX|
+------+-----------+
```

StructuredStreaming - outputModes



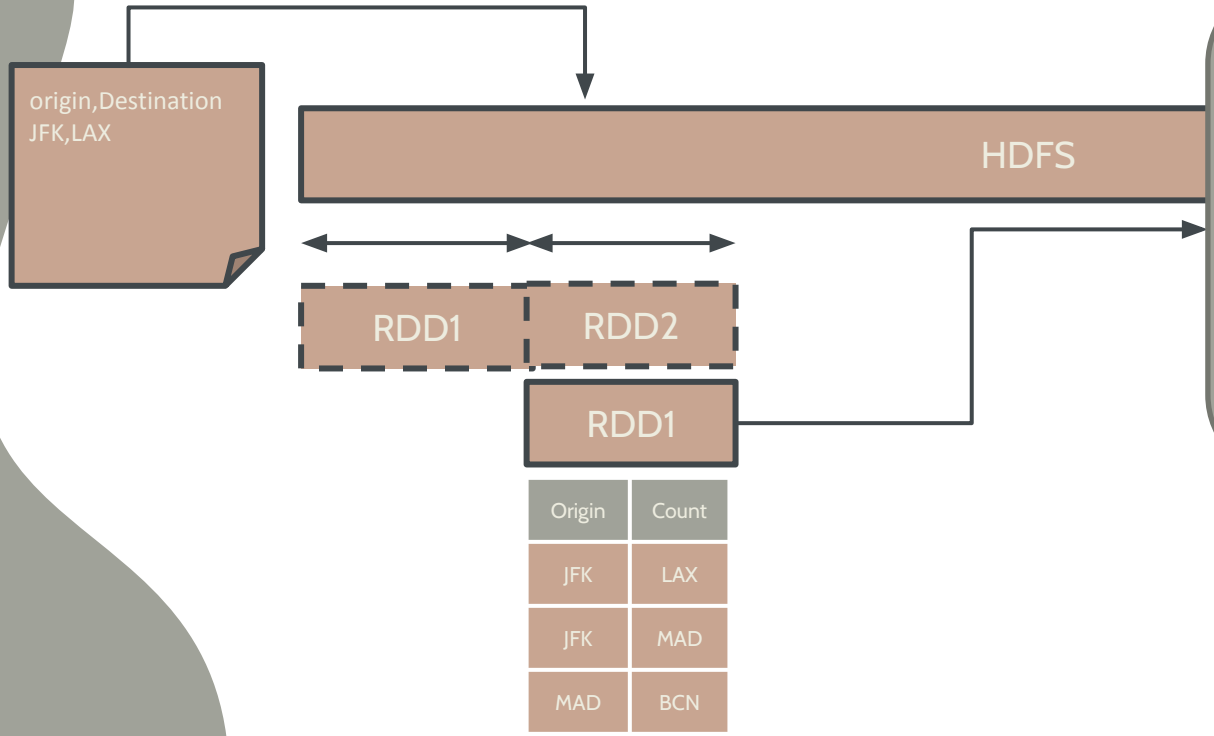

```
>>>flights.writeStream.format("console").query
Name("fligths").trigger(processingTime='2
seconds').outputMode("append").start()
-------------------------------------------
Batch: 2
-------------------------------------------
+------+-----------+
|Origin|Destination|
+------+-----------+
|   JFK|        DEL|
|   LGR|        MAD|
+------+-----------+
```

StructuredStreaming - outputModes



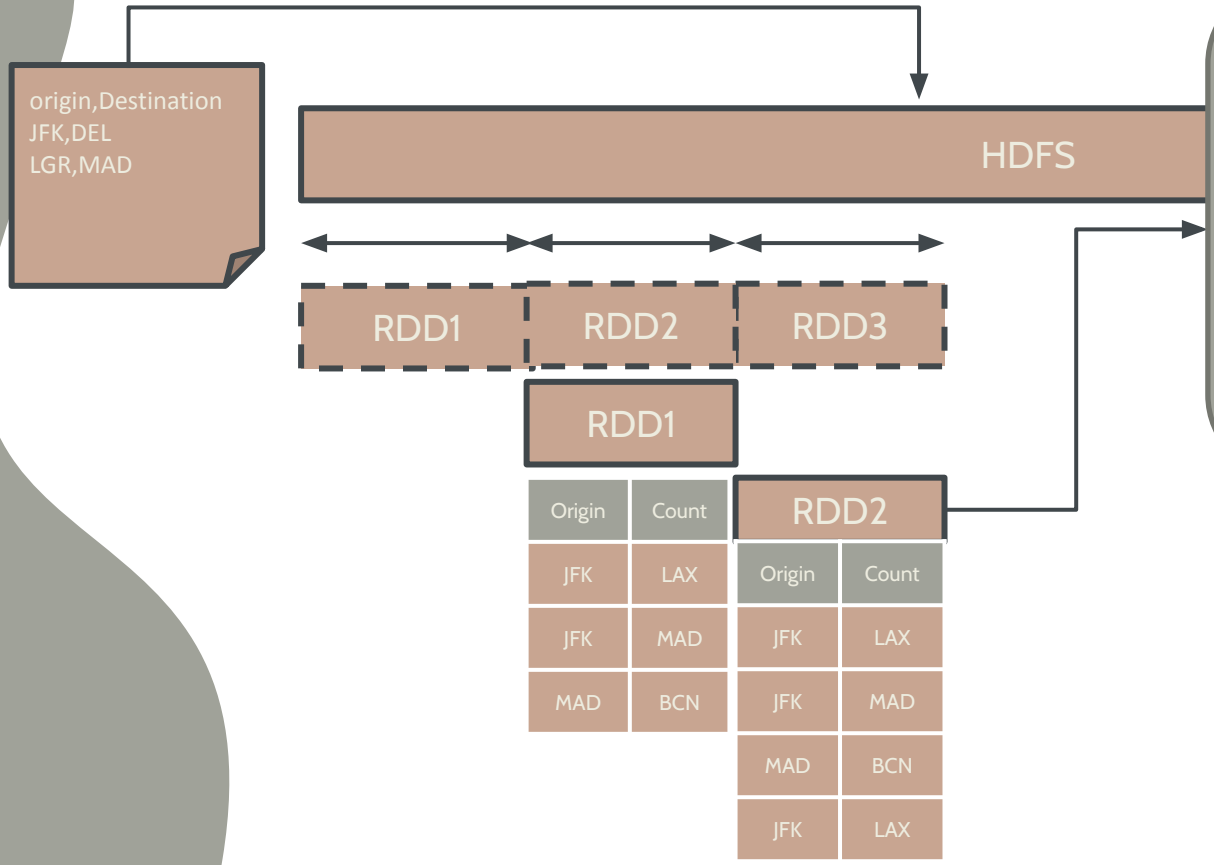

```
>>>flights.groupBy(“origin”).count().writeStream.format("console").queryName("fligths_count").trigger(processingTime='2 seconds').outputMode("complete").start()
```

StructuredStreaming - outputModes

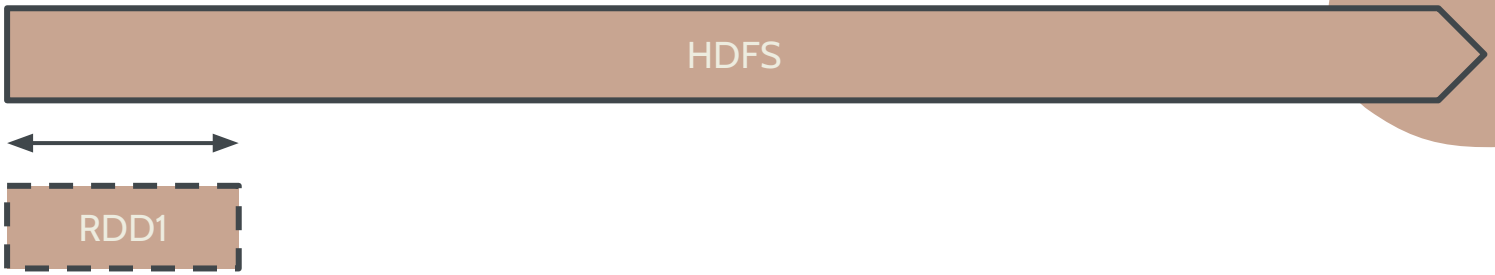



```
>>>flights.groupBy("origin").count().writeStre
am.format("console").queryName("fligths_count"
).trigger(processingTime='2
seconds').outputMode("complete").start()
-------------------------------------------
Batch: 0
-------------------------------------------
+------+-----+
|Origin|count|
+------+-----+
|   JFK|    2|
|   MAD|    1|
+------+-----+
```

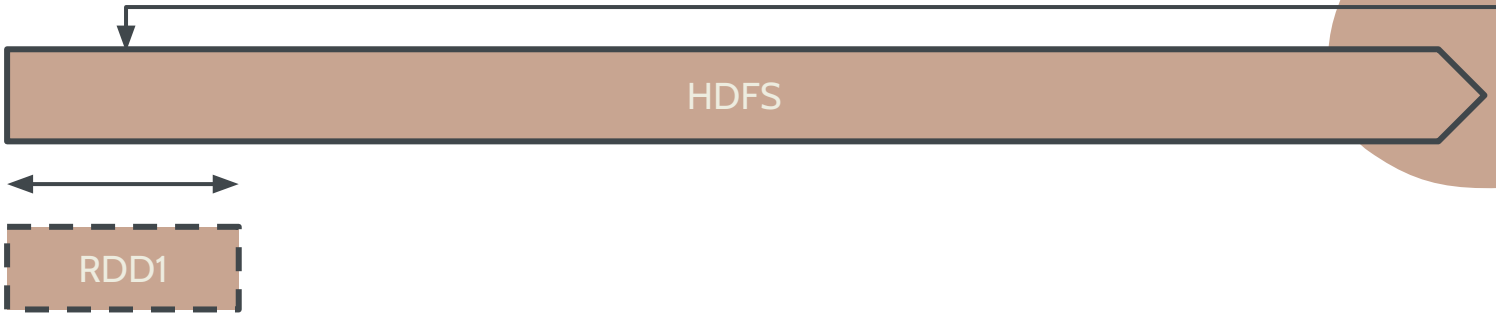
StructuredStreaming - outputModes



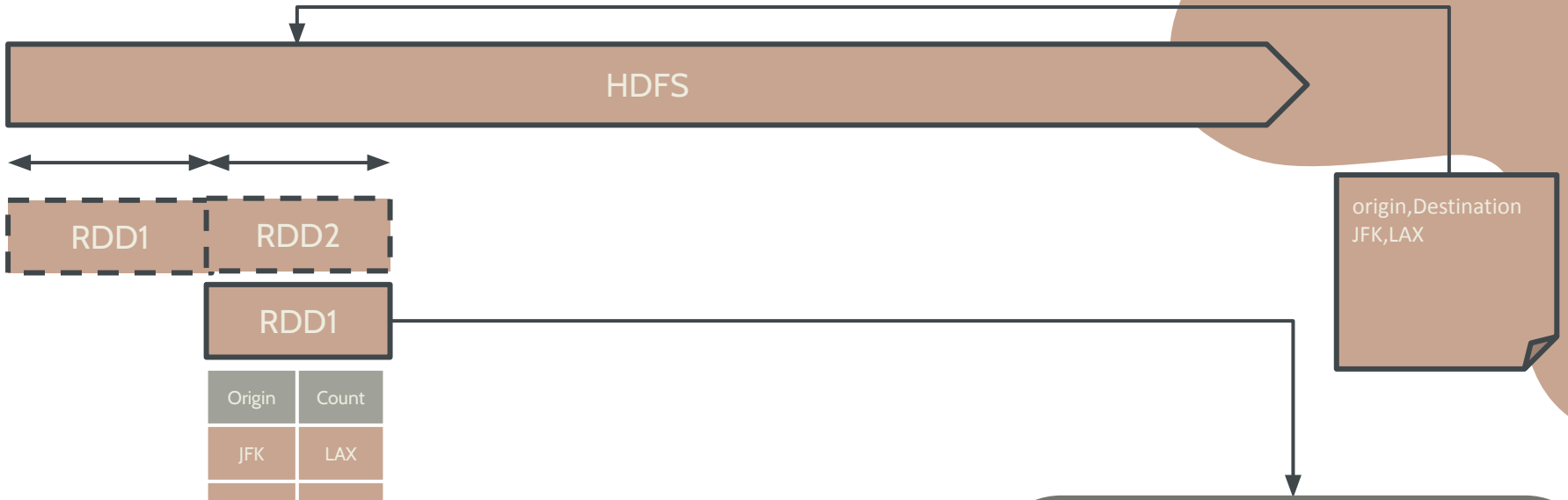

```
>>>flights.groupBy("origin").count().writeStre
am.format("console").queryName("fligths_count"
).trigger(processingTime='2
seconds').outputMode("complete").start()
-------------------------------------------
Batch: 1
-------------------------------------------
+------+-----+
|Origin|count|
+------+-----+
|   JFK|    3|
|   MAD|    1|
+------+-----+
```

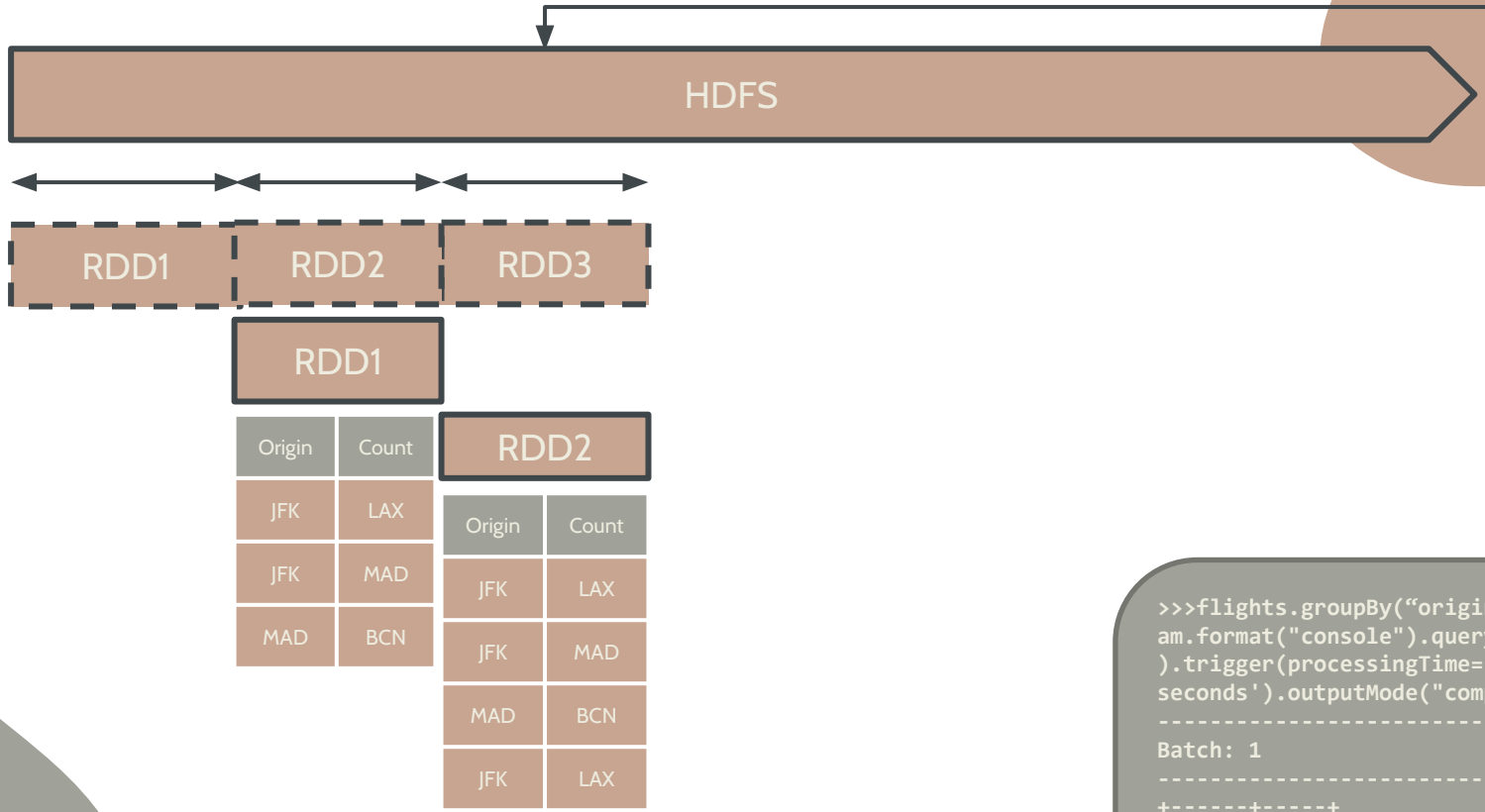



```
>>>flights.groupBy(“origin”).count().writeStream.format("console").queryName("fligths_count").trigger(processingTime='2 seconds').outputMode("complete").start()
```



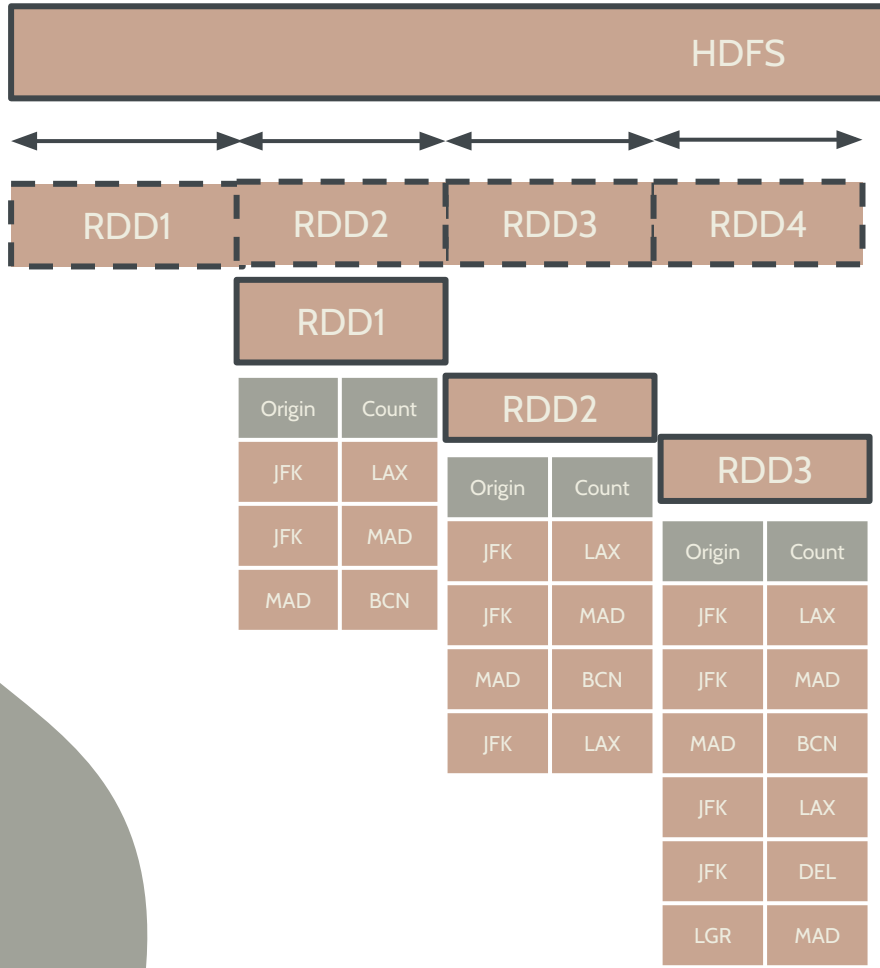

```
>>>flights.groupBy(“origin”).count().writeStre
am.format("console").queryName("fligths_count"
).trigger(processingTime='2
seconds').outputMode("complete").start()
```


HDFS
RDD1
RDD2
RDD1
Origin
Count
JFK
LAX
JFK
MAD
MAD
BCN
origin,Destination
JFK,LAX
>>>flights.groupBy("origin").count().writeStream.format("console").queryName("fligths_count").trigger(processingTime='2 seconds').outputMode("complete").start()
Batch:0
+------+-----+
|Origin|count|
+------+-----+
| JFK| 2|
| MAD| 1|
+------+-----+

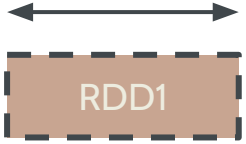


origin,Destination
JFK,DEL
LGR,MAD

```
>>>flights.groupBy("origin").count().writeStre
am.format("console").queryName("fligths_count"
).trigger(processingTime='2
seconds').outputMode("complete").start()
-------------------------------------------
Batch: 1
-------------------------------------------
+------+-----+
|Origin|count|
+------+-----+
|   JFK|    3|
|   MAD|    1|
+------+-----+
```

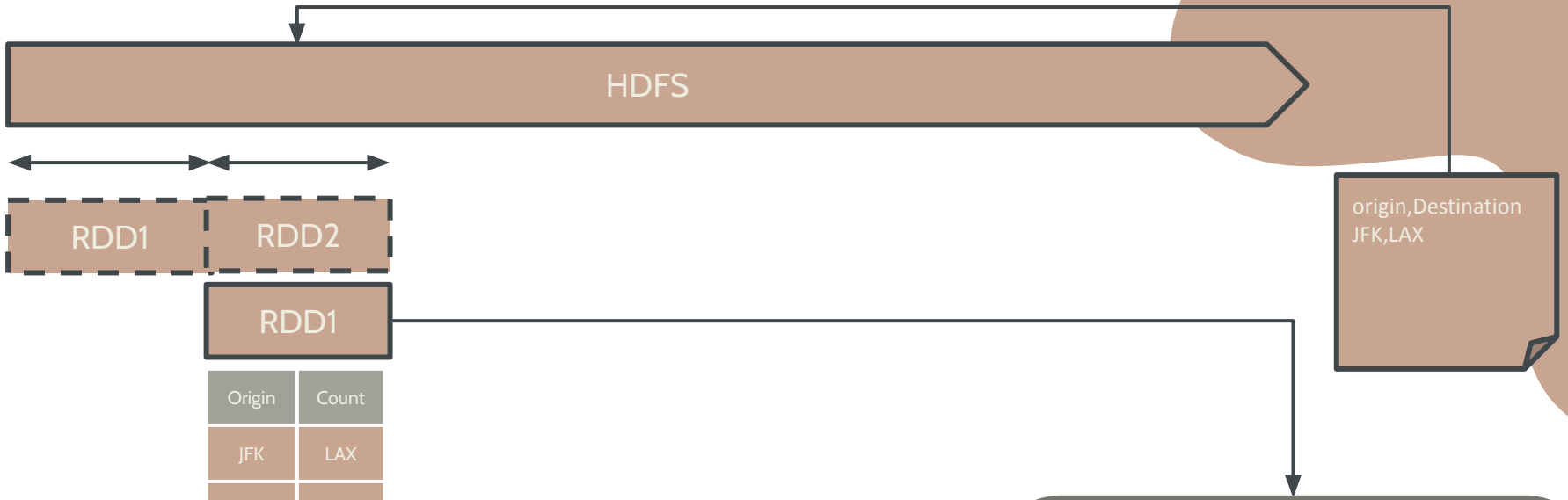



```
>>>flights.groupBy(“origin”).count().writeStre
am.format("console").queryName("fligths_count"
).trigger(processingTime='2
seconds').outputMode("complete").start()
-------------------------------------------
Batch: 2
-------------------------------------------
+------+-----+
|Origin|count|
+------+-----+
|   JFK|    4|
|   MAD|    1|
|   LGR|    1|
+------+-----+
```



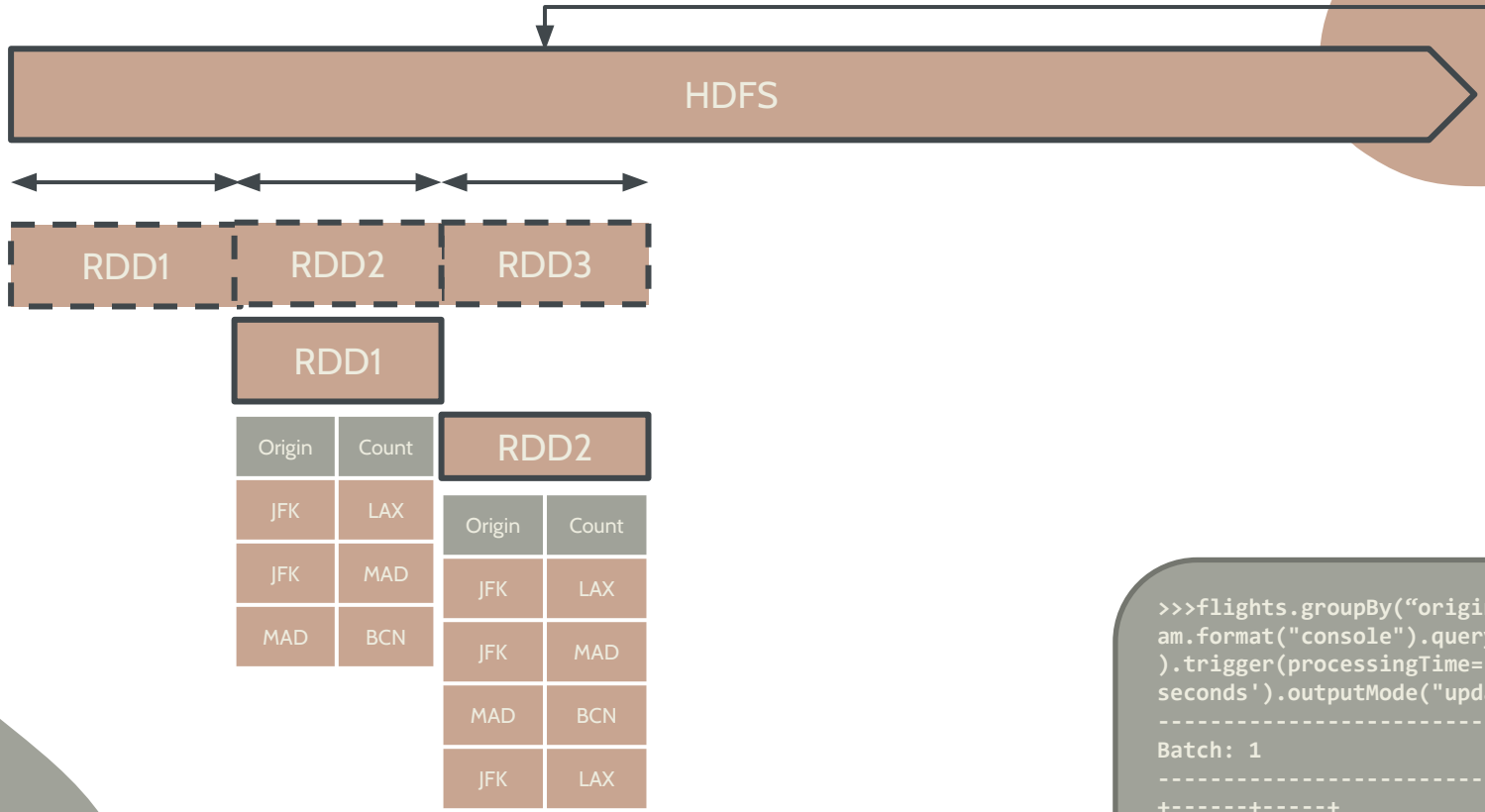
origin,Destination
JFK,LAX
JFK,MAD
MAD,BCN

```
>>>flights.groupBy(“origin”).count().writeStre
am.format("console").queryName("fligths_count"
).trigger(processingTime='2
seconds').outputMode("update").start()
```

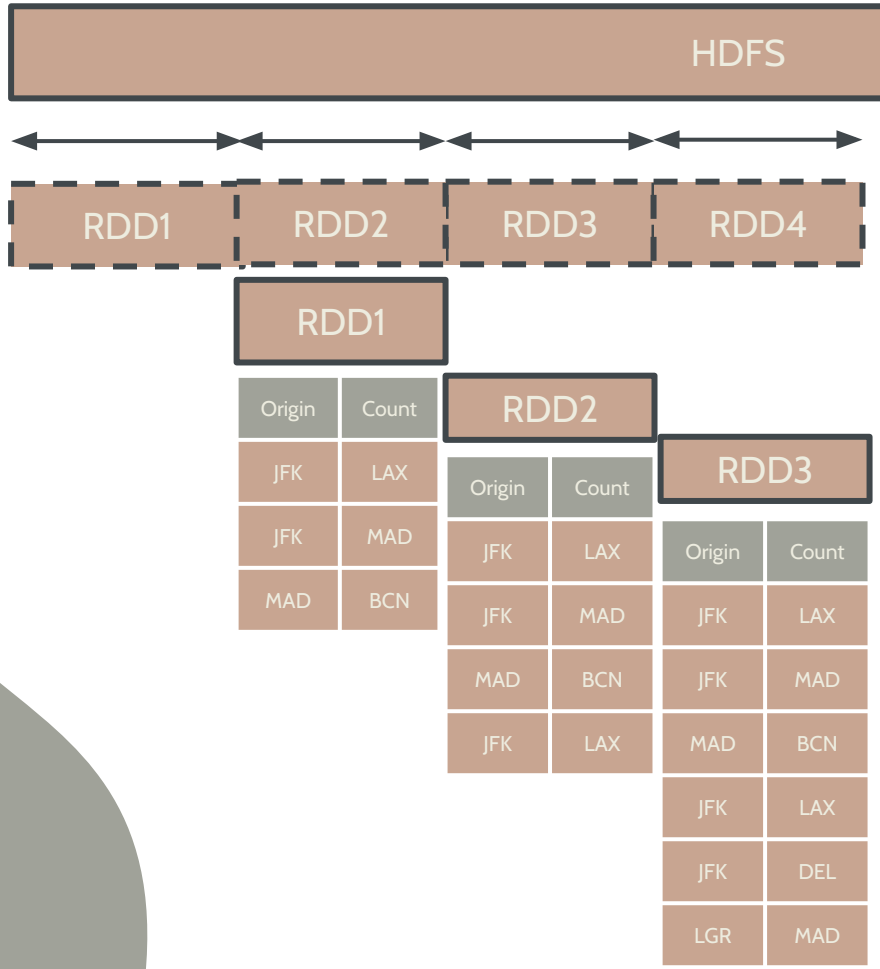

HDFS
RDD1
RDD2
RDD1
Origin
Count
JFK
LAX
JFK
MAD
MAD
BCN
origin,Destination
JFK,LAX
>>>flights.groupBy("origin").count().writeStream.format("console").queryName("fligths_count").trigger(processingTime='2 seconds').outputMode("update").start()

Batch:0

+------+-----+
|Origin|count|
+------+-----+
| JFK| 2|
| MAD| 1|
+------+-----+




```
>>>flights.groupBy("origin").count().writeStre
am.format("console").queryName("fligths_count"
).trigger(processingTime='2
seconds').outputMode("update").start()
-------------------------------------------
Batch: 1
-------------------------------------------
+------+-----+
|Origin|count|
+------+-----+
|   JFK|    3|
+------+-----+
```




```
>>>flights.groupBy("origin").count().writeStre
am.format("console").queryName("fligths_count"
).trigger(processingTime='2
seconds').outputMode("update").start()
-------------------------------------------
Batch: 2
-------------------------------------------
+------+-----+
|Origin|count|
+------+-----+
|   JFK|    4|
|   LGR|    1|
+------+-----+
```

StructuredStreaming - windowAggregation

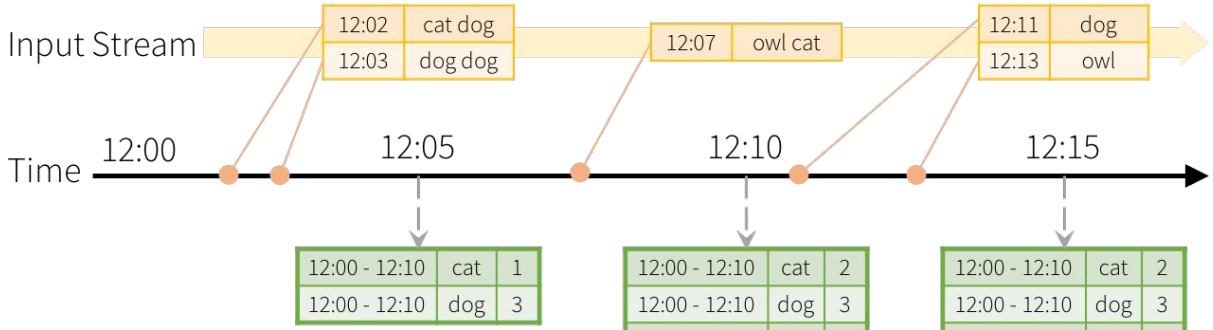
- En muchos casos de uso la fecha de recepción del evento no forma parte del resultado final
- **StructuredStreaming** nos permite definir una columna como variable temporal y decir que sólo se usen los últimos X segundos para agregación
- Muy útil para casos de uso en los que el tiempo de recepción y el de envio no son el mismo

StructuredStreaming - windowAggregation

Contar las palabras repetidas en los últimos 10 minutos cada 5 minutos

```
>>> words = ...  # streaming DataFrame of schema { timestamp: Timestamp, word: String }

# Group the data by window and word and compute the count of each group
windowedCounts = words.groupBy(
    window(words.timestamp, "10 minutes", "5 minutes"),
    words.word
).count()
```

StructuredStreaming - windowAggregation



Result Tables
after 5 minute triggers

counts incremented for windows
12:00 - 12:10 and 12:05 - 12:15

counts incremented for windows
12:05 - 12:15 and 12:10 - 12:20

Windowed Grouped Aggregation
with 10 min windows, sliding every 5 mins

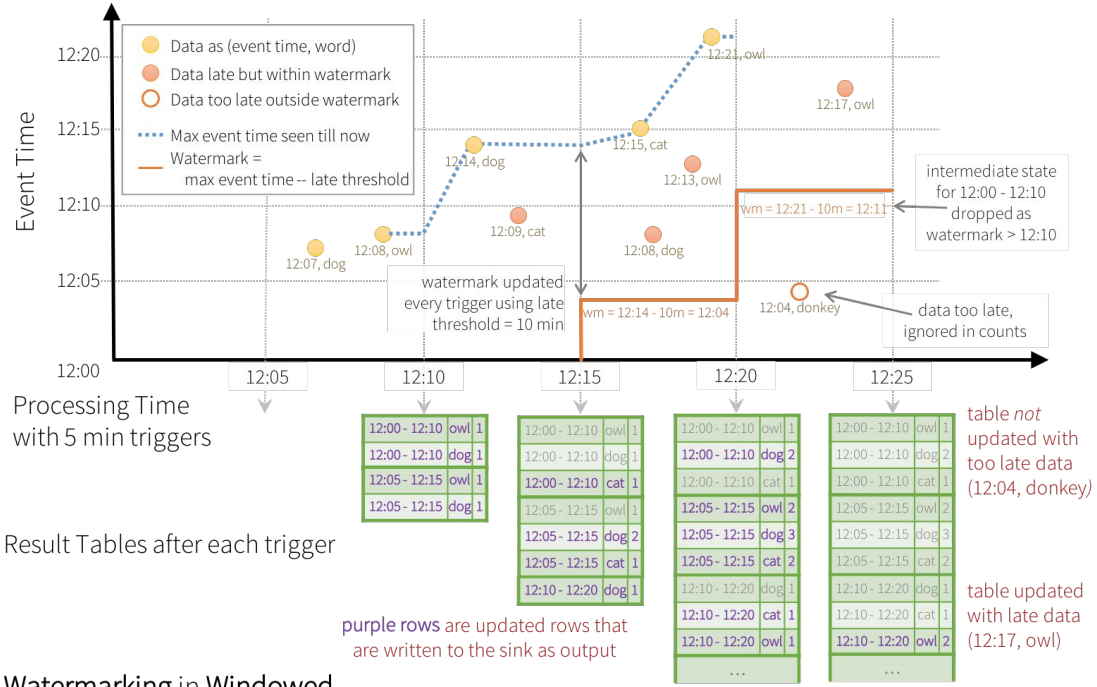
StructuredStreaming - watermarking

- Uno de los mayores problemas del streaming es la diferencia entre tiempo del evento y tiempo de emisión del mismo
    - **Tiempo del evento**: instante de tiempo en el que evento es generado
    - **Tiempo de emisión del evento:** instante de tiempo en el que evento es emitido
- Muchos sistemas de emisión de eventos tardan un tiempo en emitir cada evento que generan
- En los sistemas de streaming muchas veces es mejor descartar un evento que procesarlo en un tiempo erróneo
- Dado que Spark resuelve la casuística de Streaming usando micro-batch este concepto se hace fundamental

StructuredStreaming - watermarking

- En Spark Streaming estos conceptos de recepción y emisión de evento tenían que ser gestionados por el desarrollador
- Uno de los objetivos de StructuredStreaming es facilitar el desarrollo de aplicaciones con casuística streaming se tiene que facilitar está gestión
- En StructuredStreaming esto se gestiona mediante *watermarking* (marcas de agua)
- En cada Dataframe se tiene que asignar un campo de tipo fecha que será el denominado “campo de marca de agua”
- Además hay que asignar un tiempo máximo de recepción del evento, si un evento llega más tarde que este tiempo se descarta

StructuredStreaming - watermarking



Watermarking in **Windowed Grouped Aggregation** with **Update Mode**

Structured Streaming