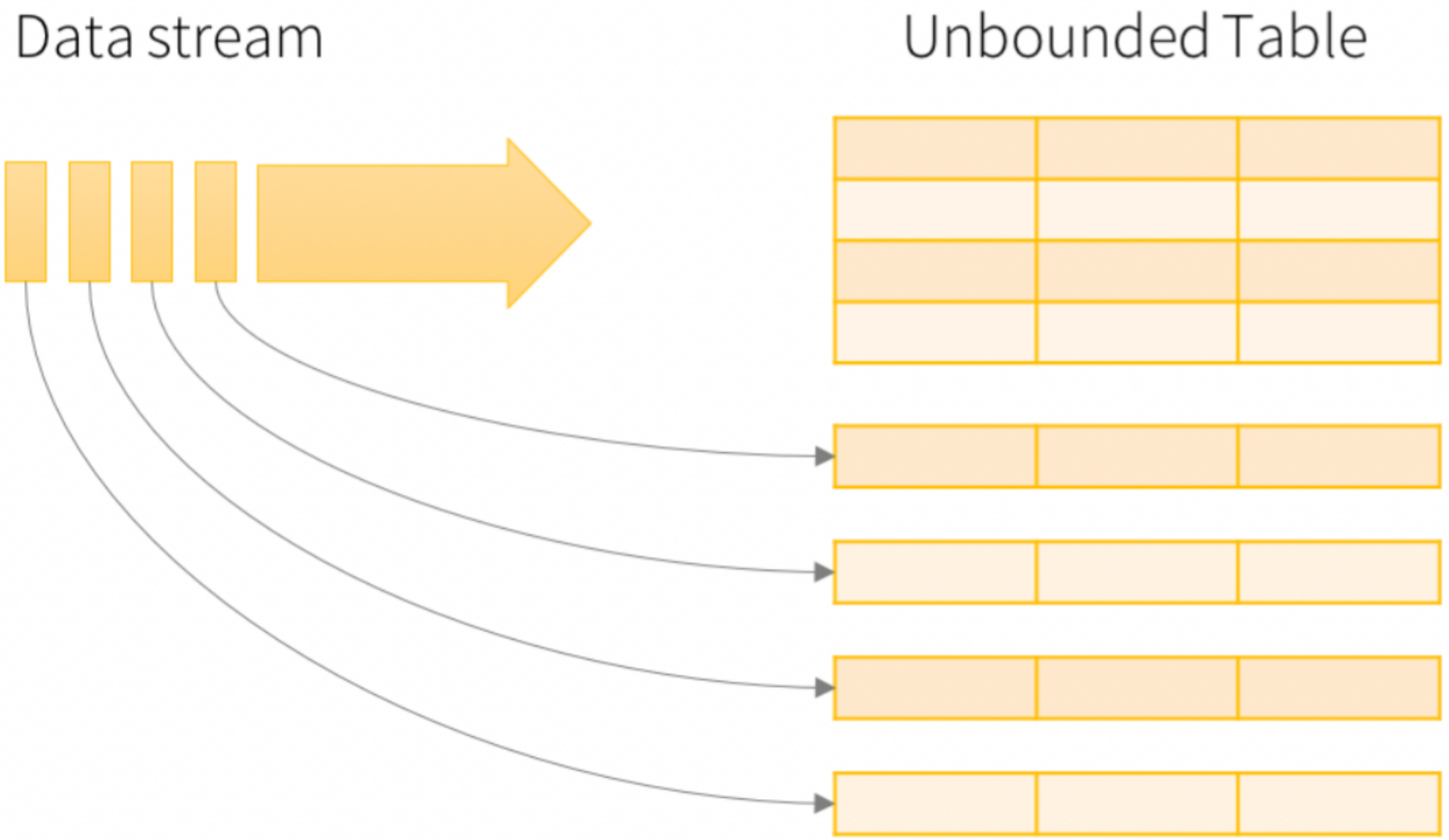
Apache Spark

Andrés Gómez Ferrer
andresgomezfrr@gmail.com

Spark Structured Streaming

Spark Structured Streaming: Basics

- Structured Streaming es un motor de streaming processing construido sobre el motor de SparkSQL.
- Permite expresar el procesamiento en streaming, igual que se hace con el procesamiento en batch.
- El sistema da garantías de procesamiento end-to-end exactly-once con tolerancia a fallos utilizando mecanismos de checkpointin.
- Structured Streaming propociona un sistema de procesamiento en stream rápido, escalable, tolerance a fallos y con garantías de exactly-one, sin que el usuario tenga que saber que se encuentra trabajando sobre un stream de datos.
- Los datos son procesados usando motor de procesamiento en micro-batch:
  Procesa pequeños lotes de datos, con latencias end-to-end de 100 milisegundos y proporcionando garantías exactly-one.
- Spark2.3 añade un modo llamado **Continuos Processing,** permitiendo latencias de 1 milisegundos y proporcionando garatías at-least-once.

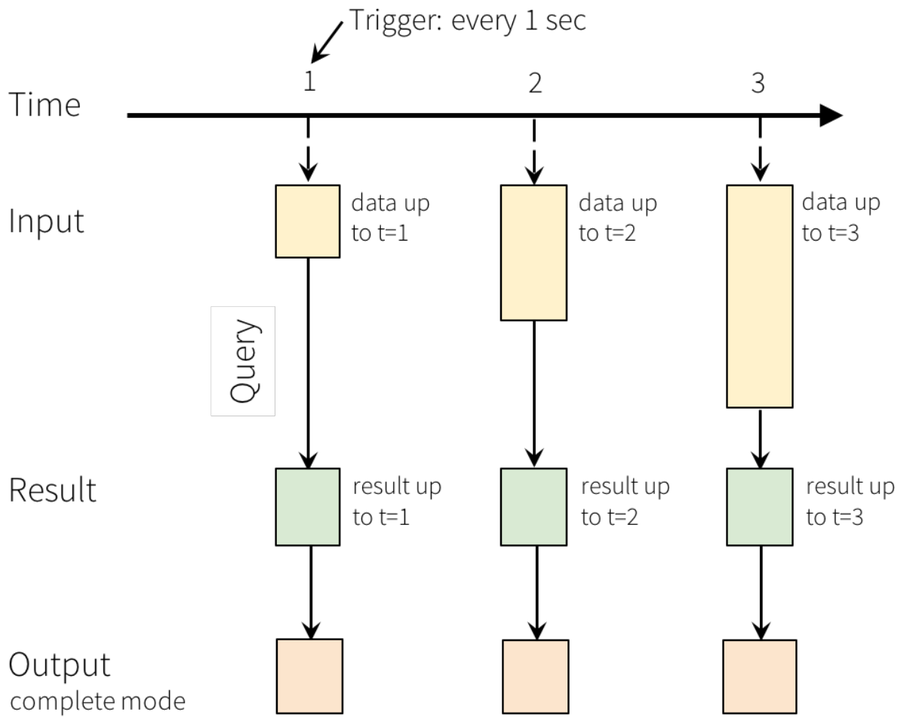


new
d

new r
to a u

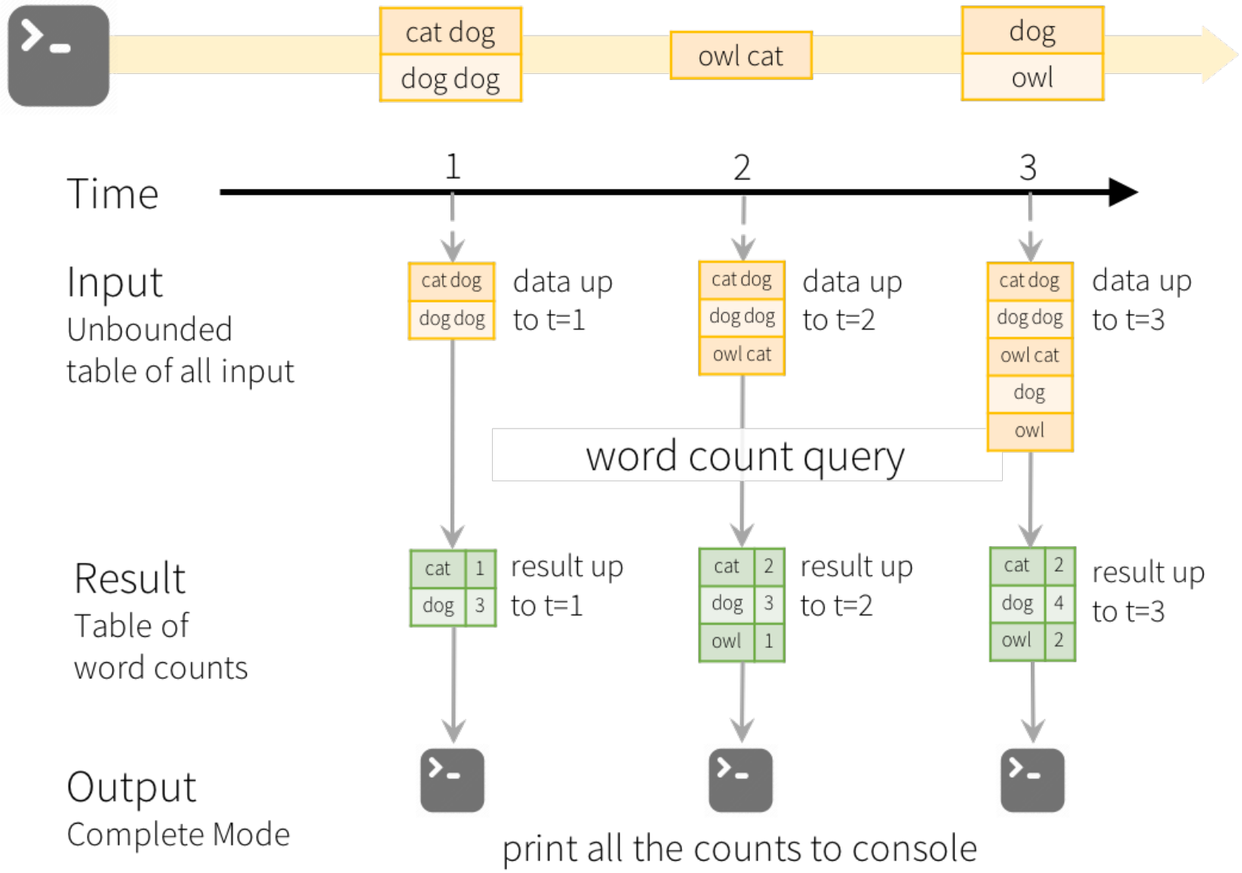
Spark Structured Streaming: Concepts

- Una query en la entrada generará la tabla de resultados.
- Cada vez que se dispara el trigger (1 sec), nuevas rows se añaden a la tabla de Input. Eventualmente se va generando la tabla de resultados.
- Cuando la tabla de resultados es actualizada, podríamos enviar estos datos a un almacenamiento externo.
- La salida puede ser definida en distintos modos:
  - **Complete Mode:** Es enviado toda la tabla de resultados al completo.
  - **Append Mode:** Únicamente son enviadas las rows nuevas añadidas desde el ultimo trigger. Asumimos que las rows antiguas no se van a modificar.
  - **Update Mode:** Son enviadas todas las rows que hayan tenido alguna modificación desde el último trigger.



Programming Model for Structured Streaming

Spark Structured Streaming: Concepts



Model of the Quick Example

```
import org.apache.spark.sql.functions._
import org.apache.spark.sql.SparkSession

val spark = SparkSession.builder
        .appName("StructuredNetworkWordCount").getOrCreate()

import spark.implicits._

val lines = spark.readStream
        .format("socket")
        .option("host", "localhost")
        .option("port", 9999)
        .load()

val query = lines.as[String]
                .flatMap(_.split(" "))
                .groupBy("value")
                .count()
                .writeStream
                .outputMode("complete")
                .format("console")
                 start()

query.awaitTermination()
```

Exercise 1

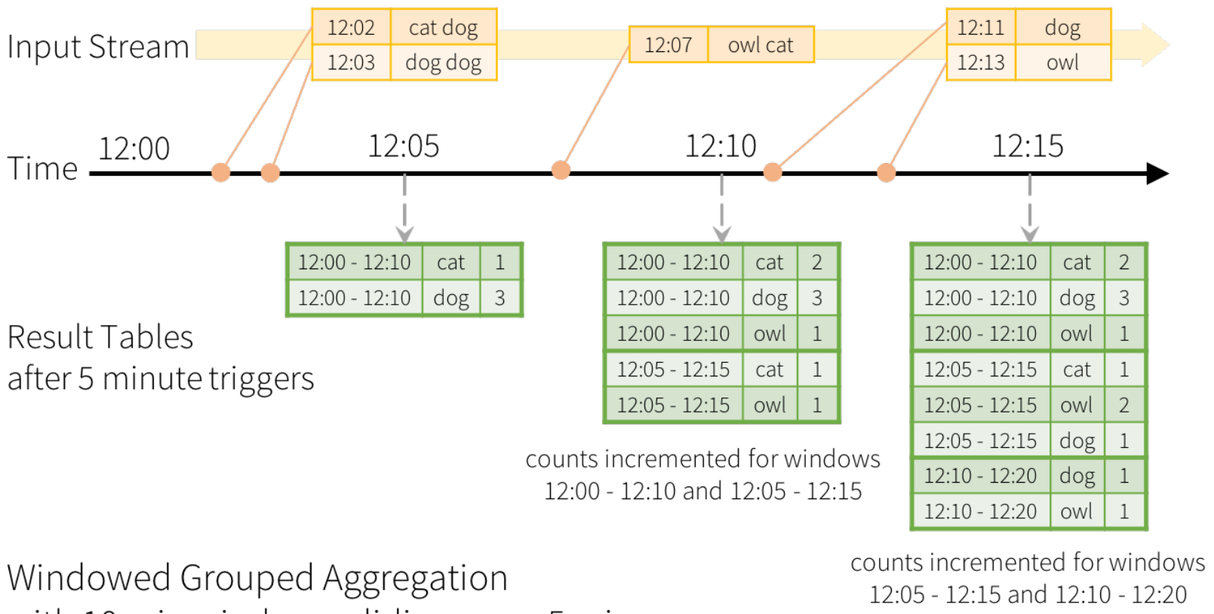
Basic Example / Output Modes

Spark Structured Streaming: Windows

```
import spark.implicits._

val words = ... // streaming DataFrame of schema
{ timestamp: Timestamp, word: String }

val windowedCounts = words
    .groupBy(
        window($"timestamp", "10 minutes", "5 minutes"),
        $"word"
    )
    .count()
```



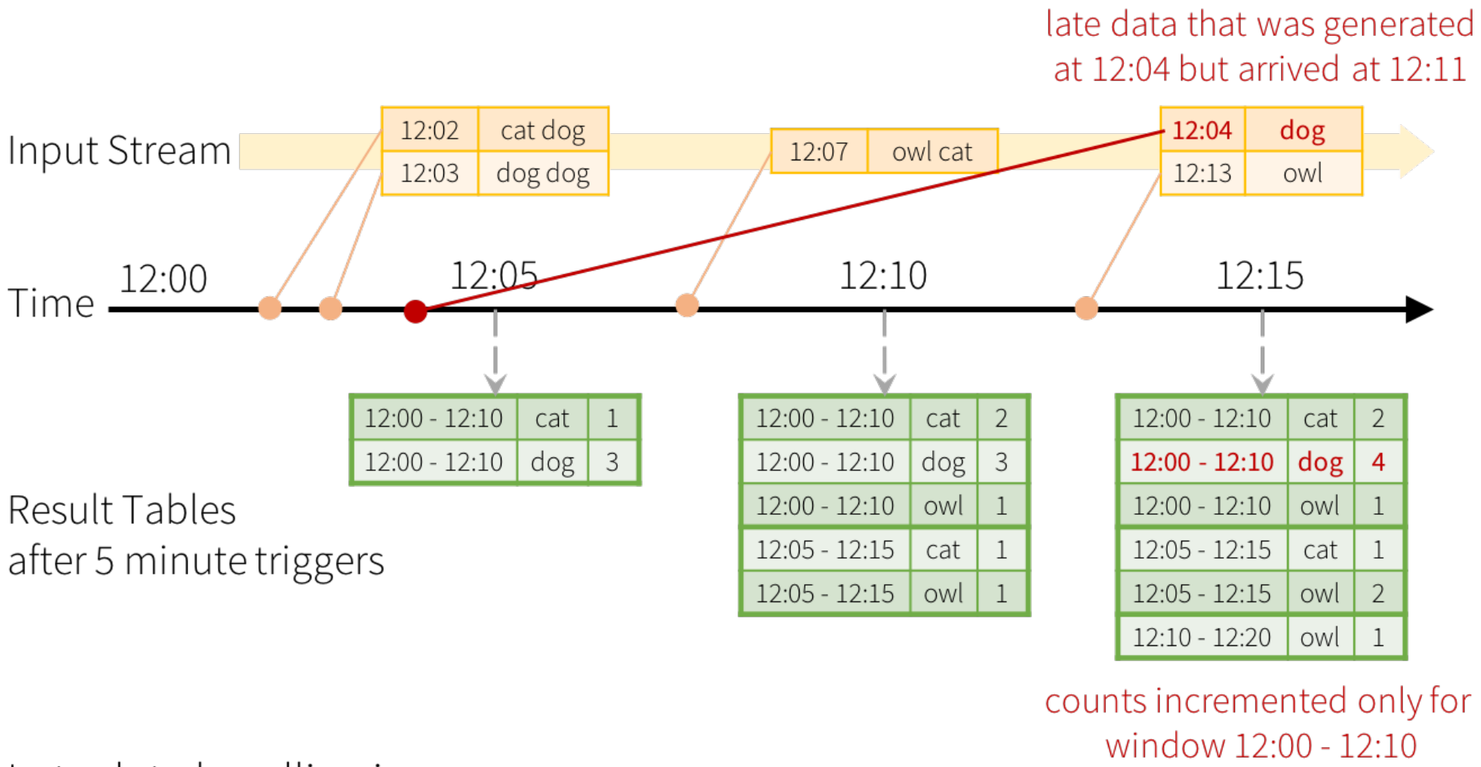
- Agrega los datos en una ventana de 10 minutos que se desliza cada 5 minutos.
- El resultado de las tablas es enviado en base el modo que se utilice.

Spark Structured Streaming: Windows

- En ocasiones se pueden recibir datos antiguos.
- Spark puede mantener en memoria un estado parcial de las agregaciones durante un gran periodo de tiempo, por lo que se puede agregar eventos que llegan con retrasos.

HAY UN LIMITE

- Ejecutar esta consulta durante días puede alcanzar limite de los datos que se pueden almacenar.
- El sistema necesita saber cuando poder descartar datos antiguos, **watermarking**



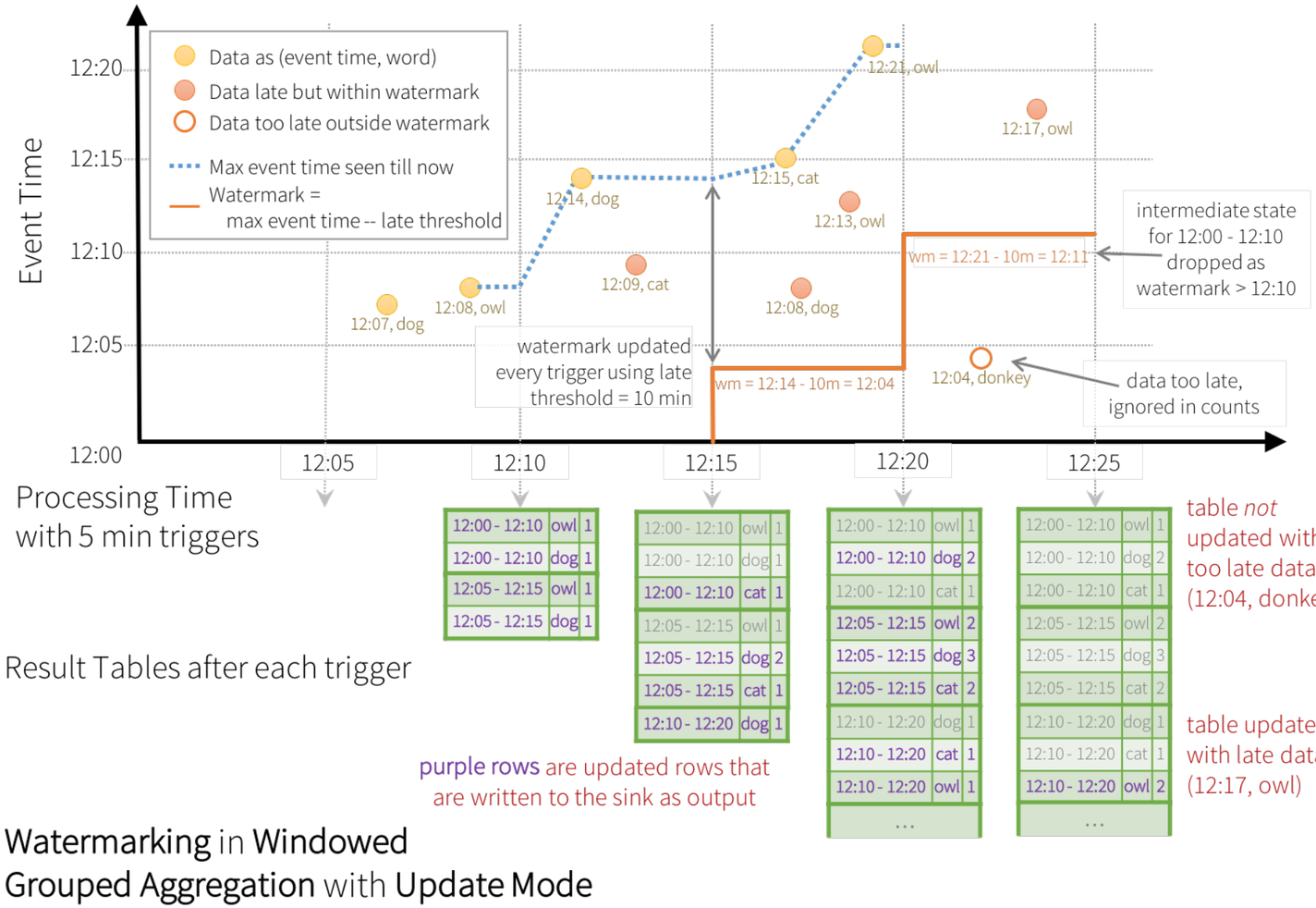
Spark Structured Streaming: Windows

- En este ejemplo se permite recibir datos con una antigüedad máxima de 10 minutos

```
val windowedCounts = words
        .withWatermark("timestamp", "10 minutes")
        .groupBy(
                window($"timestamp", "10 minutes", "5 minutes"),
                $"word")
        .count()
```

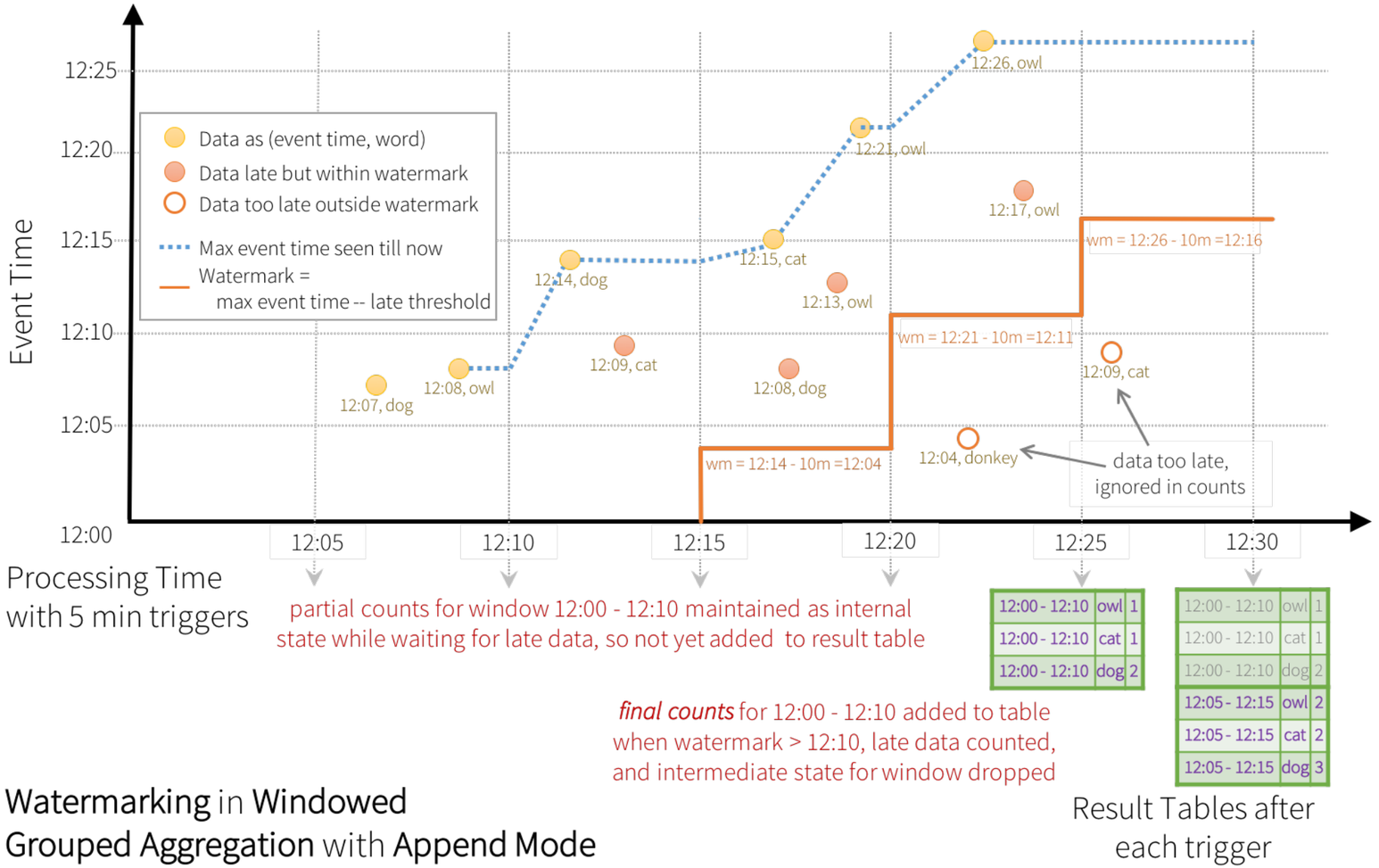
Condiciones

1. Modo Append / Update
2. Agregación con tiempo o usando window.
3. withWatermark, utilizado en la misma columna de tiempo que la agregación.
4. Configurar el withWatermark, antes que la agregación.



Spark Structured Streaming: Windows

- En el modo Append, los datos solamente son reportados cuando expira el watermark.



Watermarking in Windowed Grouped Aggregation with Append Mode

Exercise 2

Windows

Spark Structured Streaming: Joins

- Structured Streaming soporte hacer joins con dataset estáticos y con otros streams.

- **Stream-Static**: No necesita mantener estado, el stream de datos hace join con el dataset estatico previamente cargado en memoria.

```
val staticDf = spark.read. ...
val streamingDf = spark.readStream. ...

streamingDf.join(staticDf, "type")
streamingDf.join(staticDf, "type", "right_join")
```

UNICAMENTE SOPORTADOS EN MODO APPEND

- **Stream-Stream:** Necesita mantener estado de los streams, para asegurar que eventos que llegan en diferentes momentos en el tiempo hace join.

Importante:

- Definir watermarks para gestionar la memoria.

- Definir condiciones de join adicionales, para indentificar cuando las entradas dejaran de hacer match con entradas futuras.

Spark Structured Streaming: Joins

- Condiciones de joins adicionales:
  - Condiciones de rango de tiempo:

```
JOIN ON
 leftTime BETWEEN
 rightTime AND rightTime + INTERVAL 1 HOUR
```

  - Condiciones en ventana de tiempo:

```
JOIN ON leftTimeWindow = rightTimeWindow
```

- Las condiciones ayudan a que Spark pueda descartar datos a la hora de ejecutar los joins entre los dos streams.

```
import org.apache.spark.sql.functions.expr

val impressions = spark.readStream. ...
val clicks = spark.readStream. ...

// Apply watermarks on event-time columns
val impressionsWithWatermark = impressions.withWatermark(
                                    "impressionTime", "2 hours”
                                )
val clicksWithWatermark = clicks.withWatermark("clickTime", "3 hours")

// Join with event-time constraints
impressionsWithWatermark.join(
      clicksWithWatermark,
      expr("""
            clickAdId = impressionAdId AND
            clickTime >= impressionTime AND
            clickTime <= impressionTime + interval 1 hour
            """)
)
```

Exercise 3

stream-static join
stream-stream join

Spark Structured Streaming: Deduplication

- Spark soporta deduplicados de eventos, para descartar eventos que se consideran repetidos.

- **Con watermark**: La query usa el watermark para borrar datos antiguos del estado, donde no se esperan más duplicados.
- **Sin watermark**: No limites para definir cuando un record duplicado se recibe.

Guarda todos los records pasado en el estado.

```
val streamingDf = spark.readStream. ... // columns: guid, eventTime, ...

// Without watermark using guid column
streamingDf.dropDuplicates("guid")

// With watermark using guid and eventTime columns
streamingDf.withWatermark("eventTime", "10 seconds")
           .dropDuplicates("guid", "eventTime")
```

Exercise 4

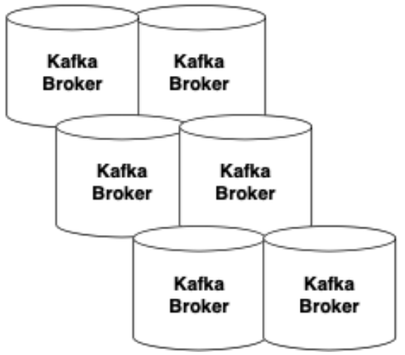
Deduplication

Spark Structured Streaming: Sinks

Sink	Supported Output Modes	Options	Fault-tolerant	Notes
File Sink	Append	`path`: path to the output directory, must be specified. For file-format-specific options, see the related methods in DataFrameWriter (Scala/Java/Python/R). E.g. for "parquet" format options see `DataFrameWriter.parquet()`	Yes (exactly-once)	Supports writes to partitioned tables. Partitioning by time may be useful.
Kafka Sink	Append, Update, Complete	See the Kafka Integration Guide	Yes (at-least-once)	More details in the Kafka Integration Guide
Foreach Sink	Append, Update, Complete	None	Yes (at-least-once)	More details in the next section
ForeachBatch Sink	Append, Update, Complete	None	Depends on the implementation	More details in the next section
Console Sink	Append, Update, Complete	`numRows`: Number of rows to print every trigger (default: 20) `truncate`: Whether to truncate the output if too long (default: true)	No	
Memory Sink	Append, Complete	None	No. But in Complete Mode, restarted query will recreate the full table.	Table name is the query name.

Apache Kafka

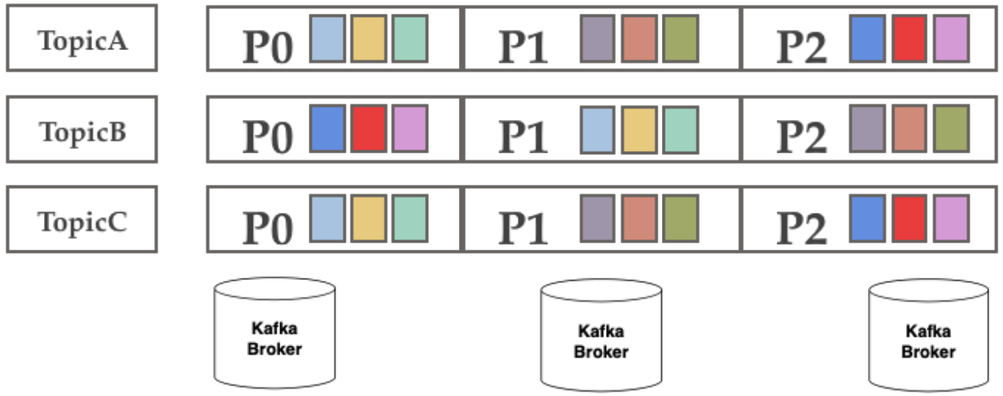
- Sistema distribuido de paso de mensajes.
- Desarrollado por LinkedIn y adoptado por Apache.



- Servicio principal de Kafka: Kafka Broker
- Almacena colas de mensaje: Topics
- Crea cluster, para distribuir los mensajes.
- Sincronización usando Apache Zookeeper como dependencia.

- Topics divididos en particiones: almacenan mensajes.
- Las particiones tienen replicas.

- + particiones = + procesamiento
- + replicas = + tolerancia de fallos



https://kafka.apache.org/

Exercise 5

Kafka Source

https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html

Exercise 6

Monitoring

Exercise 7

DataProc:
Sensor Data Realtime Aggregation Job