Executing Streaming Queries

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Prefix integrity and implications

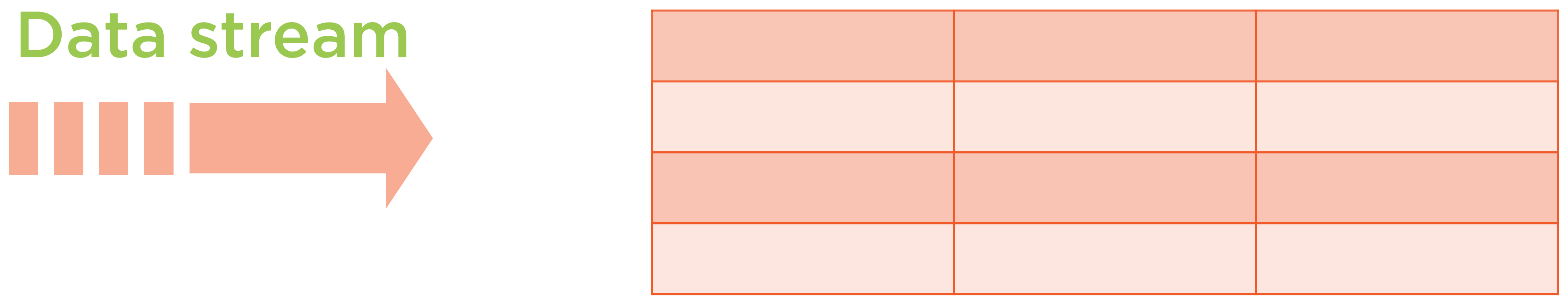
Output modes - Append, Complete, and Update

Executing streaming queries in Spark

Schema and schema auto-detection

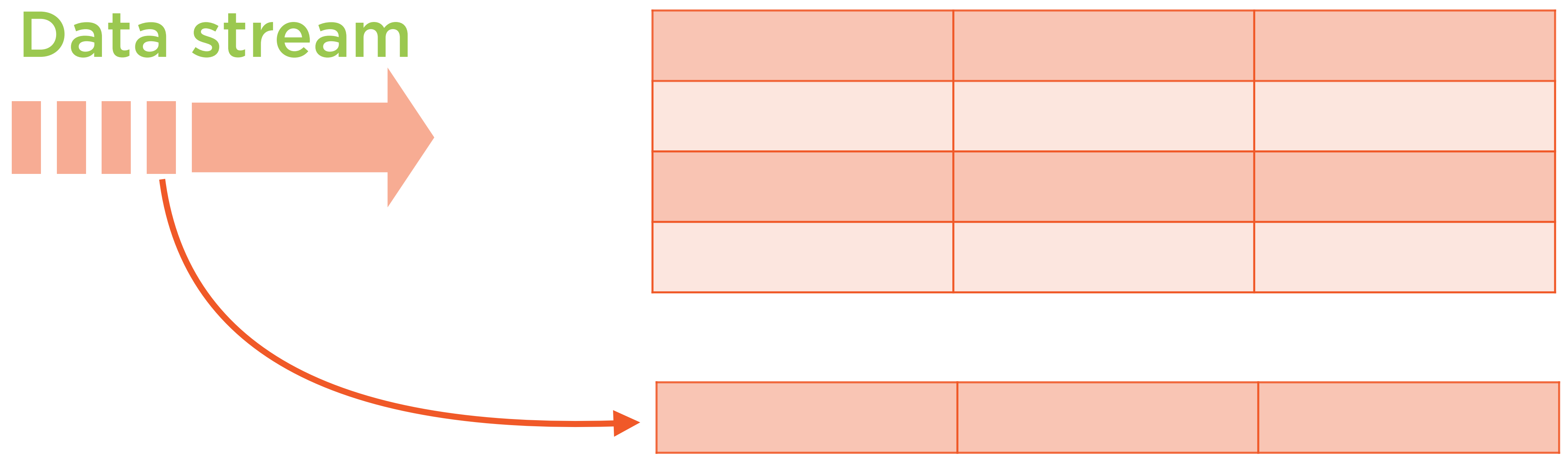
Prefix Integrity

Streaming Data Spark 2.x



Every data item that is arriving on the stream is like a new row being appended to the input table

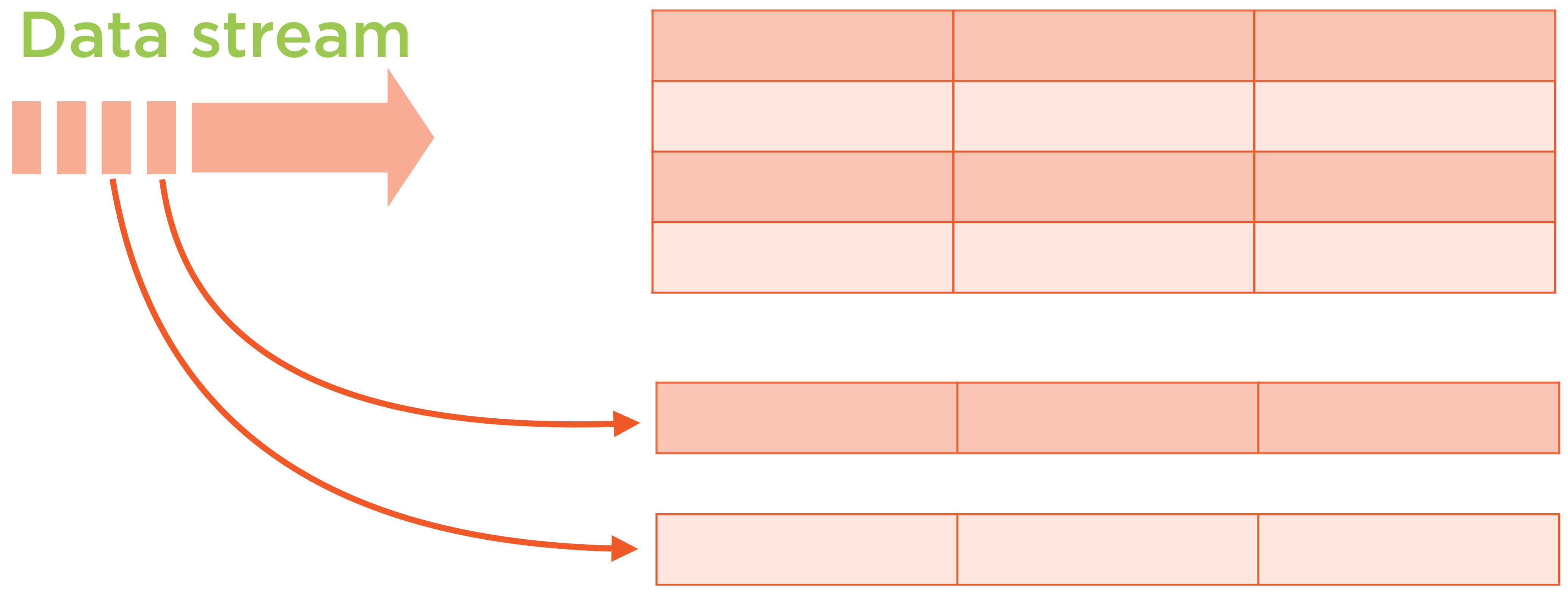
Streaming Data Spark 2.x



Every data item that is arriving on the stream is like a new row being appended to the input table

Data stream as an unbounded input table

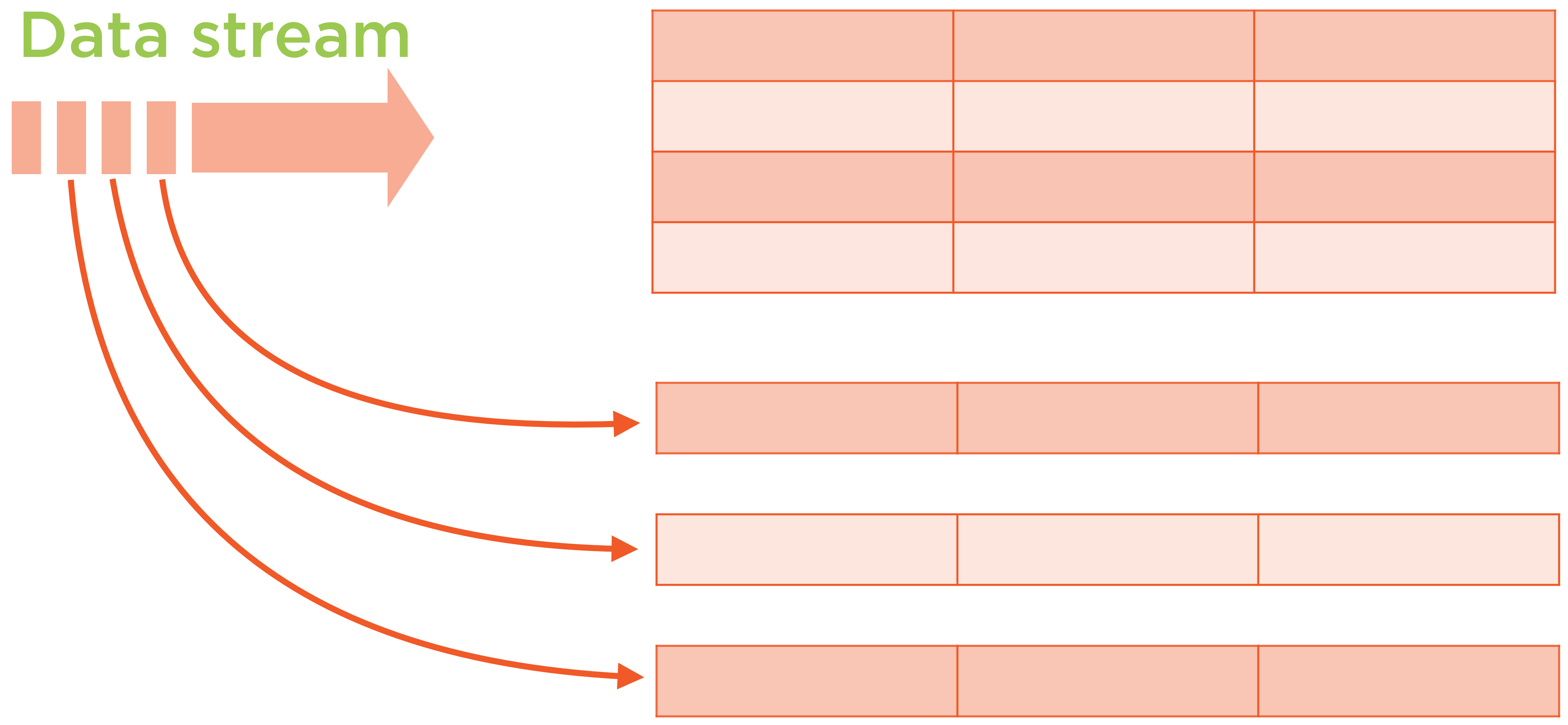
Streaming Data Spark 2.x



Every data item that is arriving on the stream is like a new row being appended to the input table

Data stream as an unbounded input table

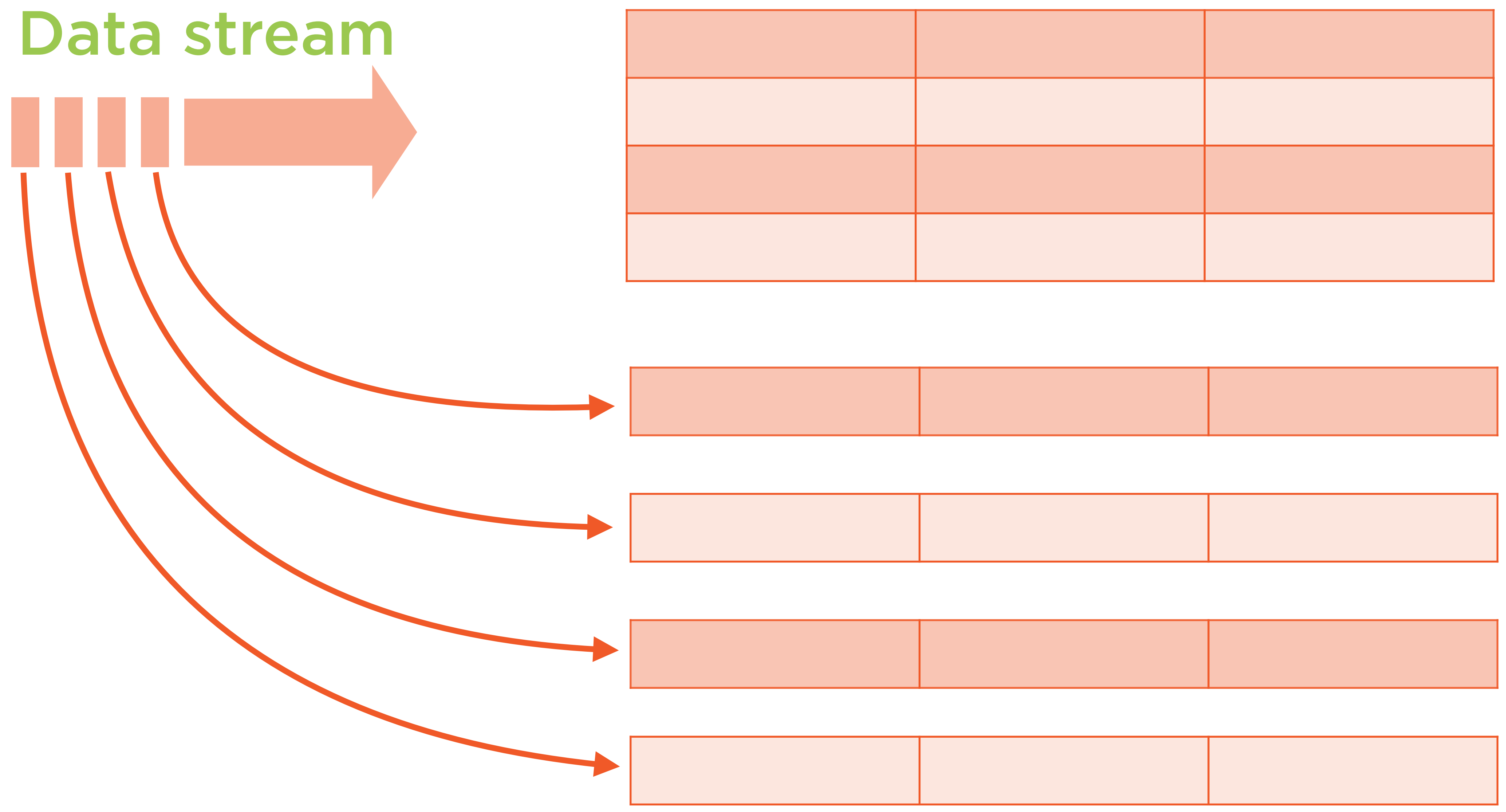
Streaming Data Spark 2.x



Every data item that is arriving on the stream is like a new row being appended to the input table

Data stream as an unbounded input table

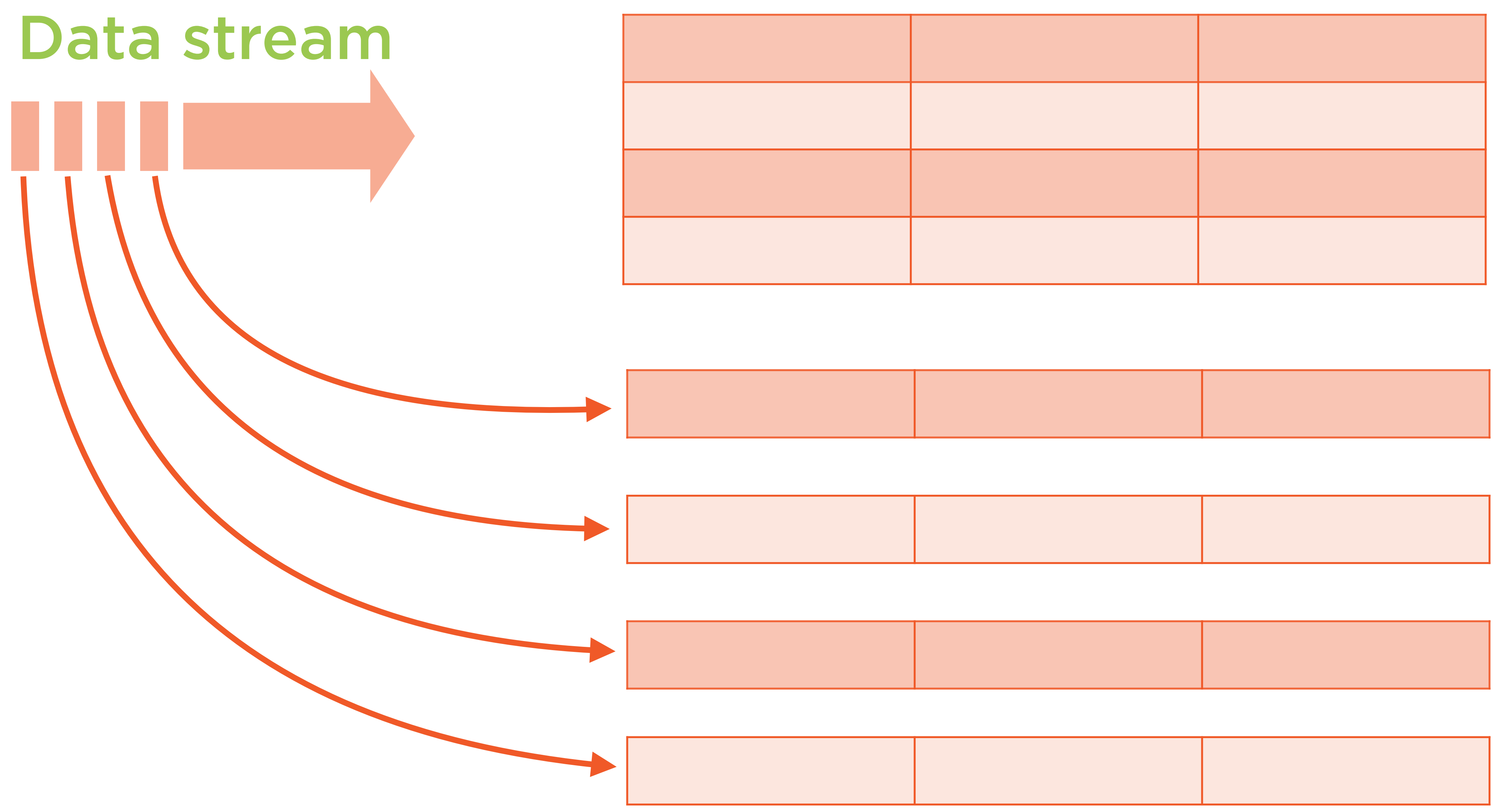
Streaming Data Spark 2.x



Every data item that is arriving on the stream is like a new row being appended to the input table

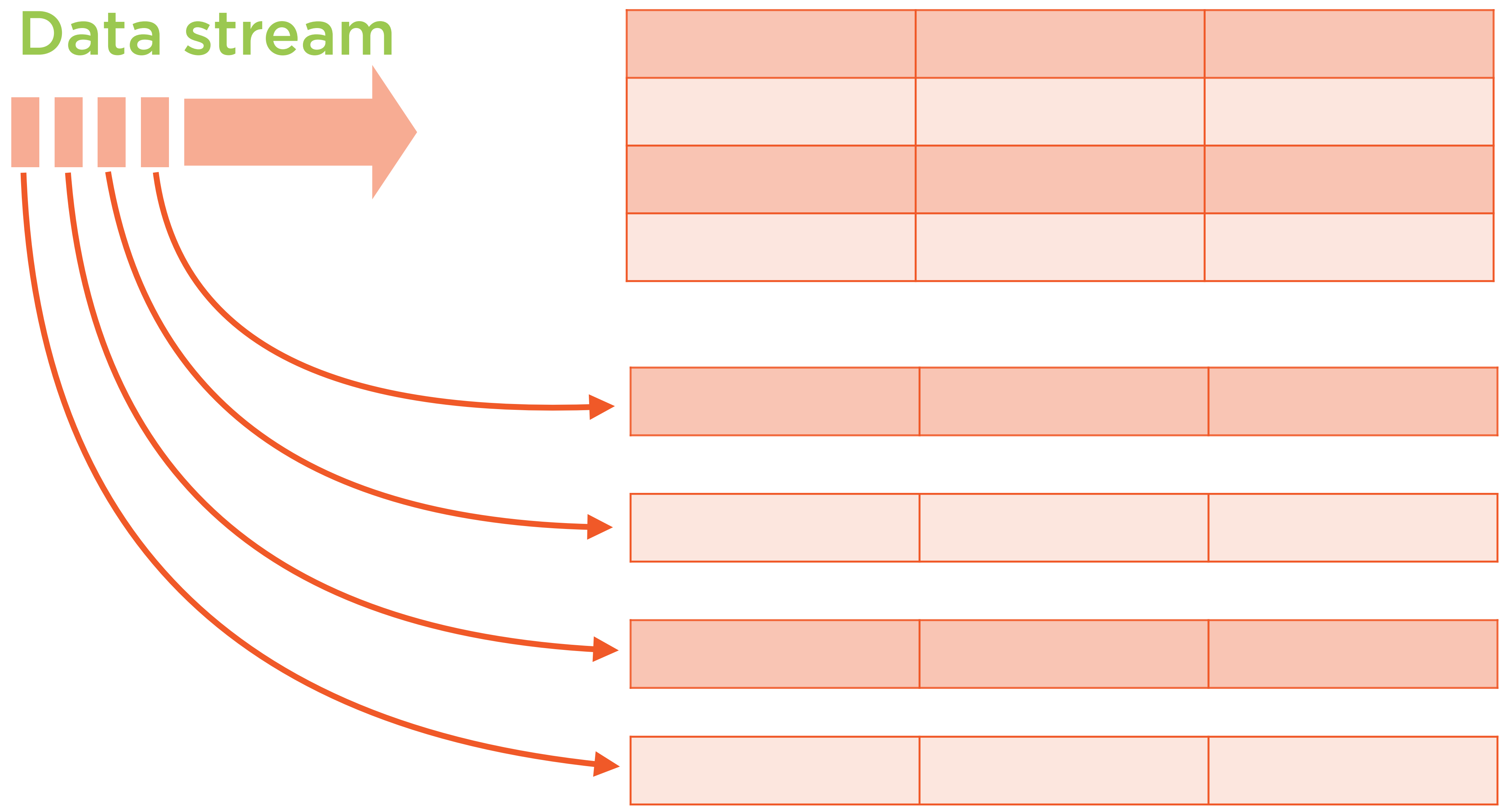
Data stream as an unbounded input table

Batch is Simply A Prefix of Stream



In other words, the input table (batch) is simply a prefix of the stream

Batch is Simply A Prefix of Stream



All operations that can be performed on data frames can be performed on the stream

Prefix Integrity

Running job on continuous data yields same result as running job on batch data (where the batch is a prefix or snapshot of continuous data).

Prefix Integrity

Running job on continuous data yields **same result** as running job on batch data (where the batch is a prefix or snapshot of continuous data).

Prefix Integrity

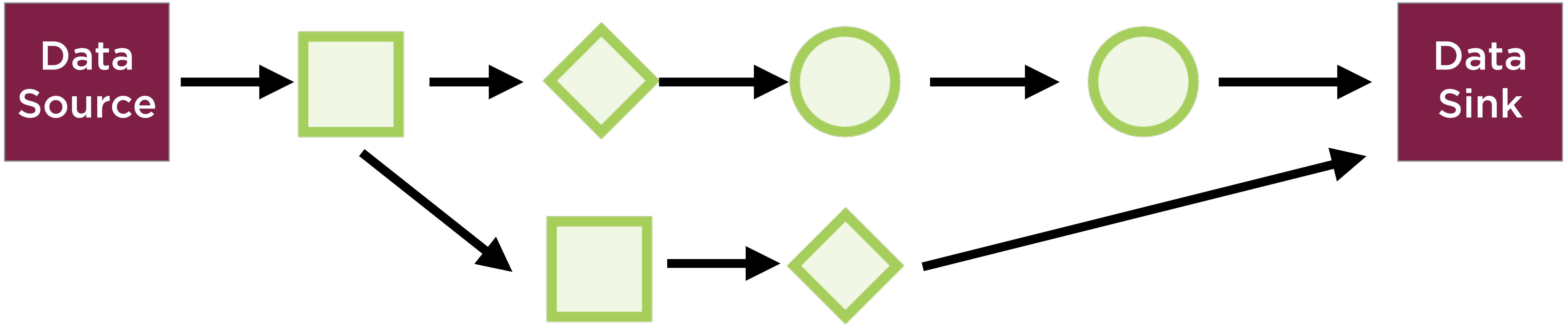
Running job on continuous data yields same result as running job on batch data (where the batch is a prefix or snapshot of continuous data).

Structured Streaming treats a live data stream as a table that is being continuously appended

Burden of stream-processing shifts from user to system

Triggers and Output Modes

Stream Processing Model



Data Source

File source

Kafka source

Socket source

Rate source

Data Sink

File sink

Kafka sink

Foreach sink

Console sink

Memory sink

Data Sink

Structured streaming reads data from source

- Processes incrementally
- Updates result
- Discards source

Only minimal intermediate state maintained

Trigger

Events that determine when transformations on accumulated input data need to be re-performed. Each trigger event emits new data into the Result Table.

Trigger

Events that determine when transformations on accumulated input data need to be re-performed. Each trigger event emits new data into the Result Table.

Trigger

Events that determine when transformations on accumulated input data need to be re-performed. Each trigger event emits new data into the Result Table.

Trigger

Events that determine when transformations on accumulated input data need to be re-performed. Each trigger event emits new data into the Result Table.

Result Table

Executing a query on input data generates the Result Table. Rows in the Result Table are written out to an external data sink.

Types of Triggers

Default

Fixed interval micro-batch

One-time micro-batch

Continuous with fixed checkpoint interval

Micro-batch Processing Mode

Default

Fixed interval micro-batch

One-time micro-batch

Continuous with fixed checkpoint interval

Continuous Processing Mode

Default

Fixed interval micro-batch

One-time micro-batch

Continuous with fixed checkpoint interval

Default

Used when no trigger setting specified

Query executed in micro-batch mode

Each new micro-batch generated when previous one completes processing

Fixed Interval Micro-batch

Micro-batch kicked off at user-specified intervals

If no data available no processing

Fixed Interval Micro-batch

If previous micro-batch completes within the interval:

- engine waits till interval is over

If previous micro-batch takes longer than specified interval:

- next micro-batch starts as soon as data arrives

One-time Micro-batch

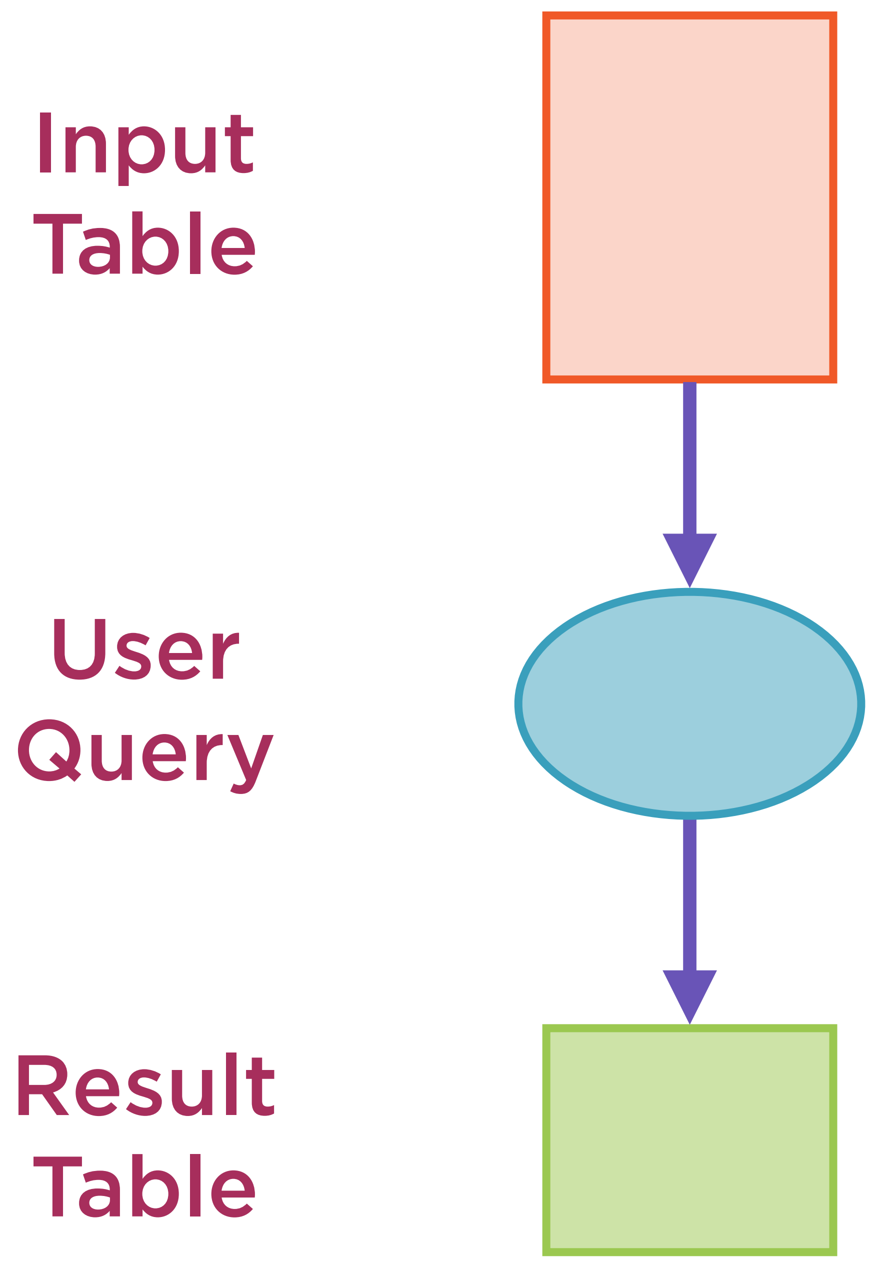
Execute only one micro-batch to process all available data

Once processed query will stop

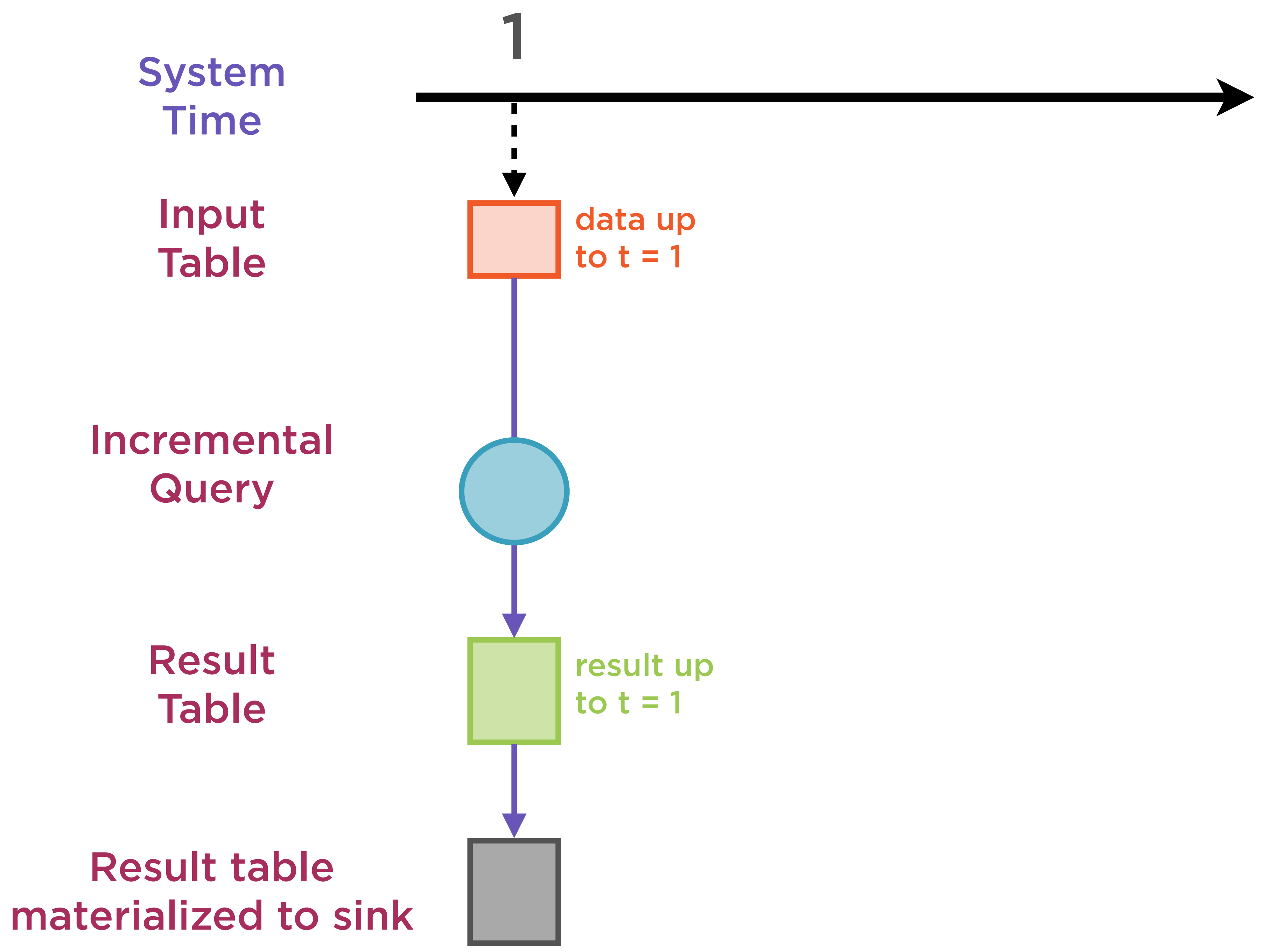
Used when cluster periodically spun up to process data since last period

May result in significant cost savings

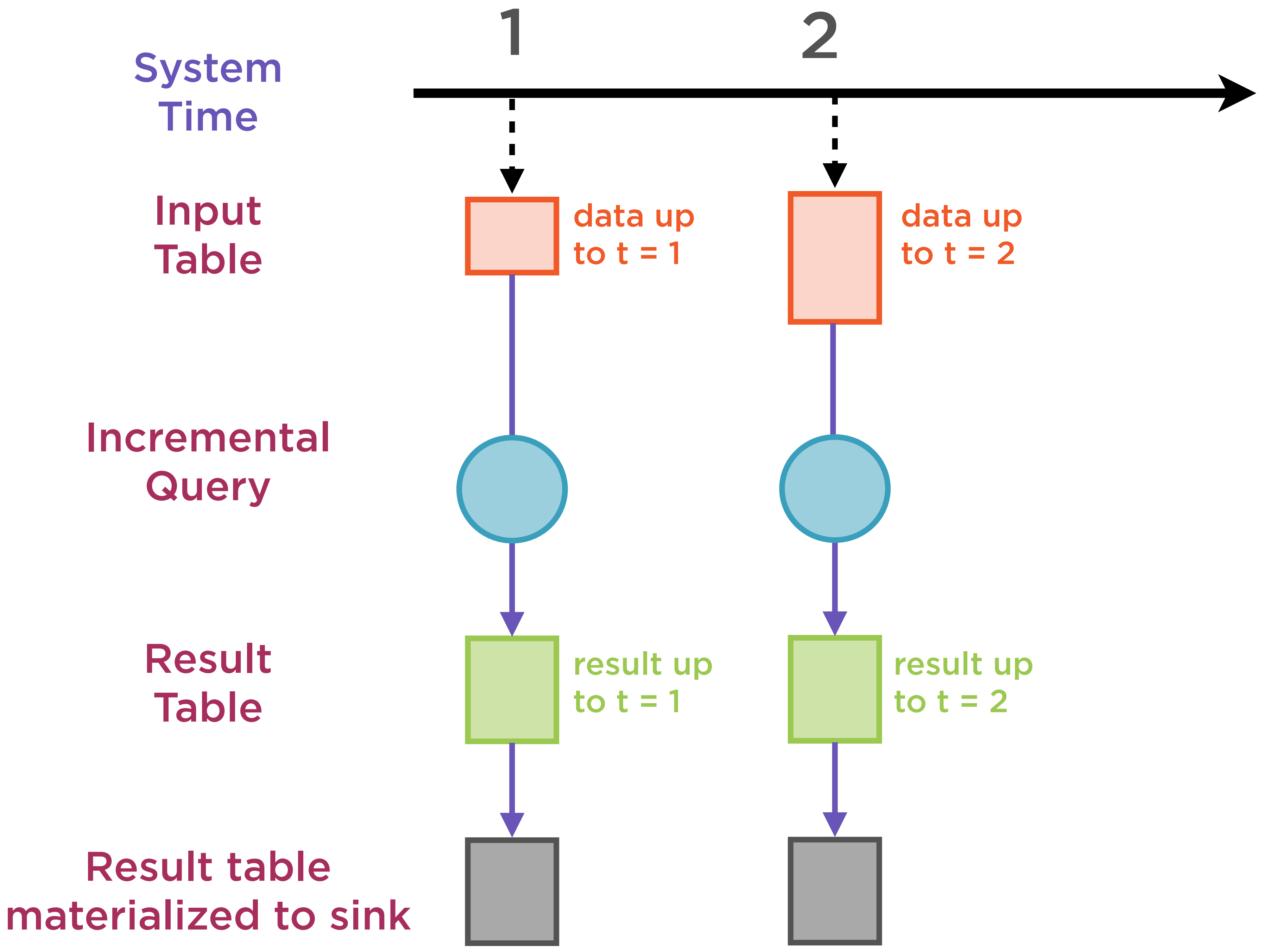
Result Table



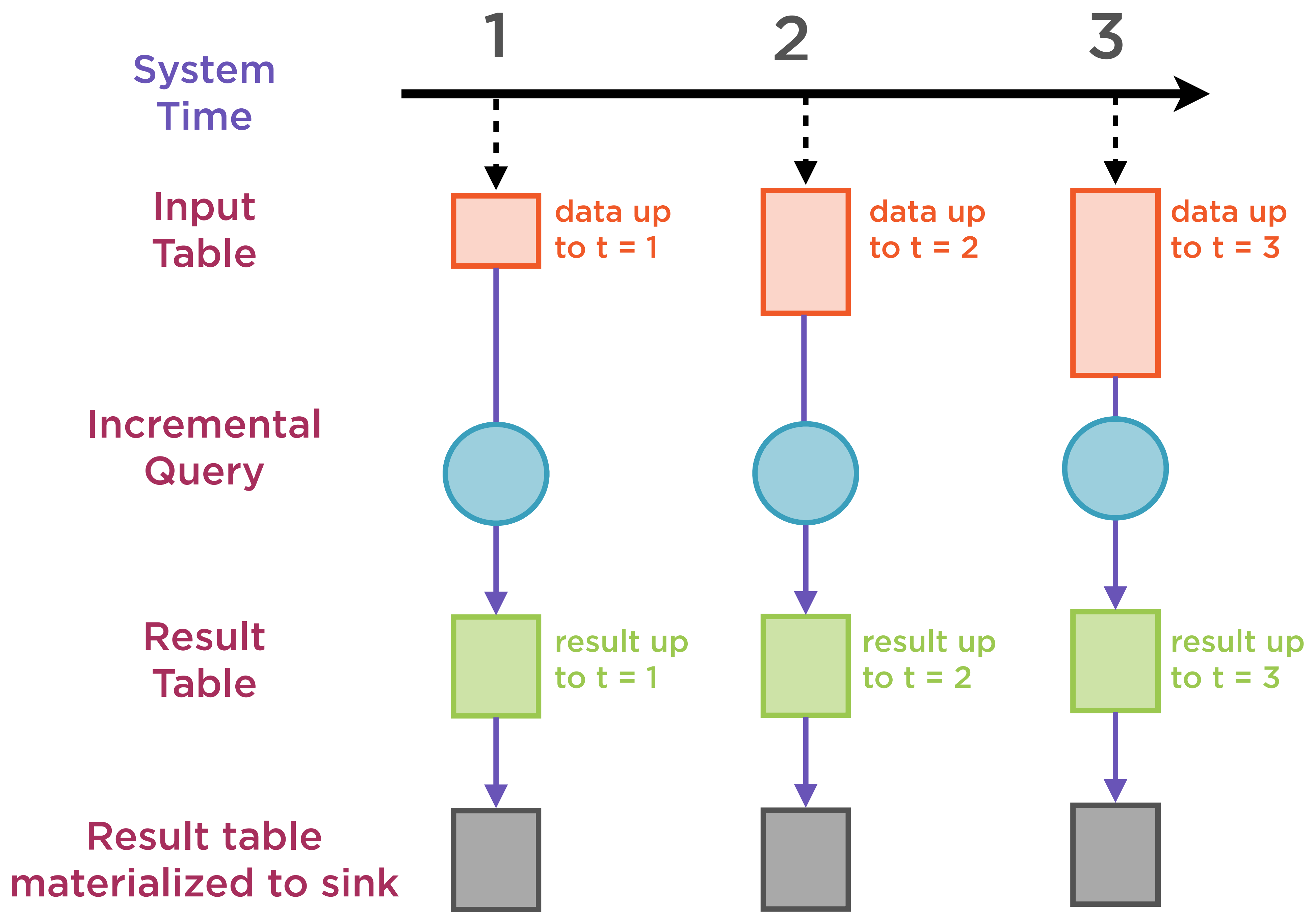
Result Table



Result Table



Result Table



When writing to the sink the entire Result Table is not materialized

What is written out depends on the **mode**

Output Modes

Determines what Result Table rows get sent to storage

- Update mode
- Append mode
- Complete mode

Output Modes

Update mode - only Result Table rows updated since last trigger

Even previous results will be updated in case of aggregations

Append mode - only Result Table rows appended since last trigger

Previous (existing) output rows cannot change

Complete mode - entire updated Result Table is sent across

Storage connector must decide how to use all that data

Output Modes

Update mode

Selections, projections, and aggregations

Append mode

Selections, projections, aggregations not supported

Complete mode

Selections, projections, aggregations, ordering

Demo

Projections and filtering in append mode using DataFrames

Demo

Projections and filtering in append mode using Spark SQL

Demo

Aggregations and grouping in complete mode using DataFrames

Demo

Aggregations and grouping in complete mode using Spark SQL

Demo

Aggregations and grouping in update mode using DataFrames

Demo

Aggregations and grouping in update mode using Spark SQL

Schema Inference

Schema Inference

Streaming DataFrames can be untyped

- Schemas will not be checked at compile-time

However some operations need schema information at compile-time

- map, flatMap

Untyped Streaming DataFrames can be converted to typed in such cases

Schema Inference

Structured Streaming from file sources requires schema to be specified

- Done by default to ensure consistent schema even in case of failures

For ad-hoc cases, can turn on schema inference

- spark.sql.streaming.schemaInference = true

Schema Inference

Perform schema inference on a small subset of the input stream

Process as batch data

Once schema known, use with streaming sources

Demo

Adhoc schema inference using batch data

Summary

Prefix integrity and implications

Output modes - Append, Complete, and Update

Executing streaming queries in Spark

Schema and schema auto-detection

Up Next:

Understanding Scheduling and Checkpointing