/01

MARCH 29th

# EPL Reference : Match recognize

Mathilde Rieussec, Tania Ramos, Léa Charbonnier

# Introduction

The MATCH RECOGNIZE clause, represents a powerful tool for pattern recognition within relational data streams. Unlike conventional SQL queries that operate on static datasets stored in tables, MATCH RECOGNIZE allows for the analysis of streaming data, making it particularly valuable for real-time analytics and stream processing applications. By enabling the detection of complex patterns within dynamic data streams, MATCH RECOGNIZE offers a versatile solution across various domains, including finance, telecommunications, and more.

## Introduction

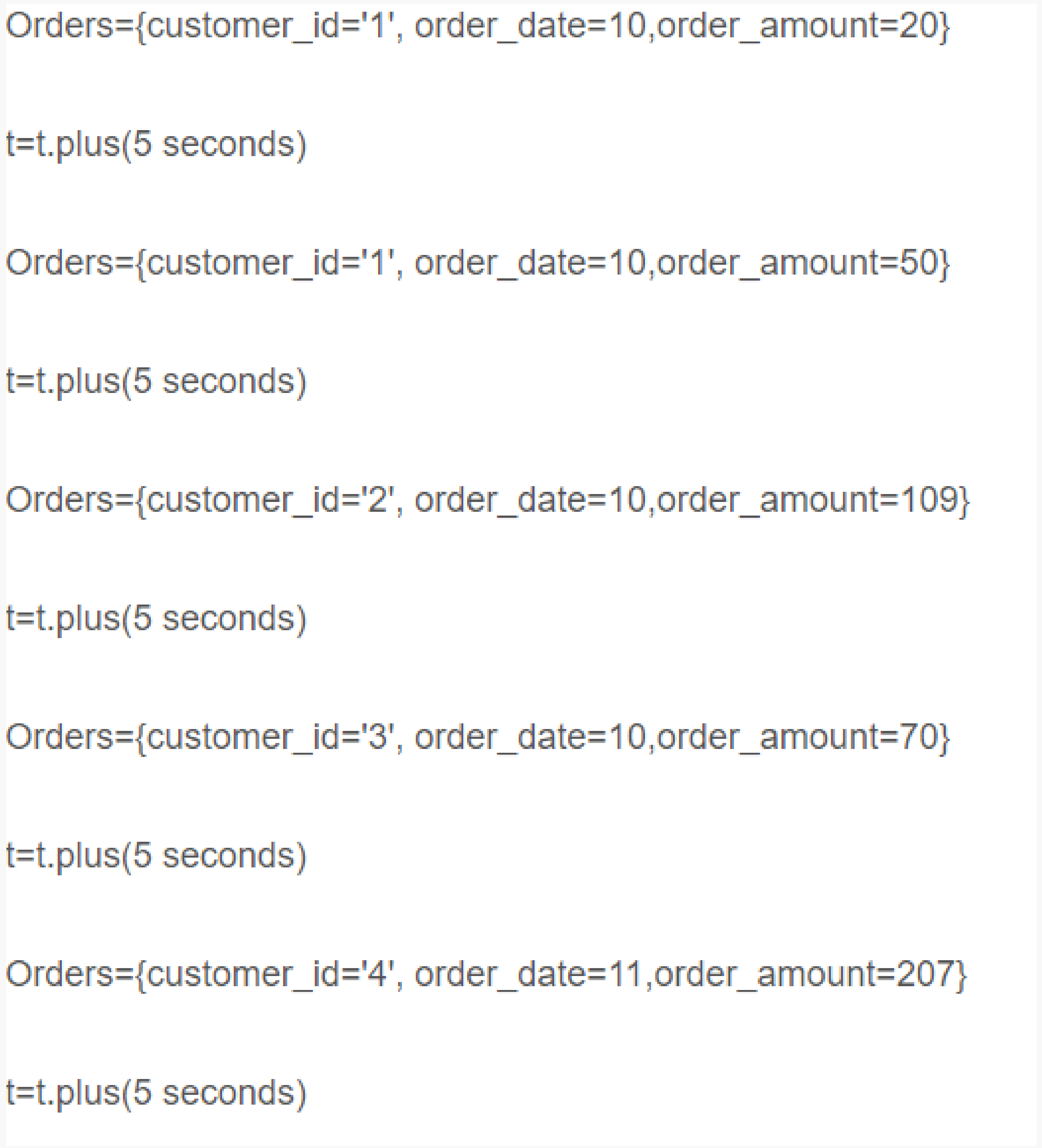
Consider a retail company that operates an online marketplace. With thousands of transactions occurring every minute, the company needs to monitor its platform for fraudulent activities continuously.

By utilizing the MATCH RECOGNIZE clause, they can define patterns indicative of fraudulent behavior, such as multiple purchases from the same IP address within a short time frame. Upon detection, the system can automatically flag these transactions for review, allowing the company to promptly respond to potential threats and protect its customers from fraud.

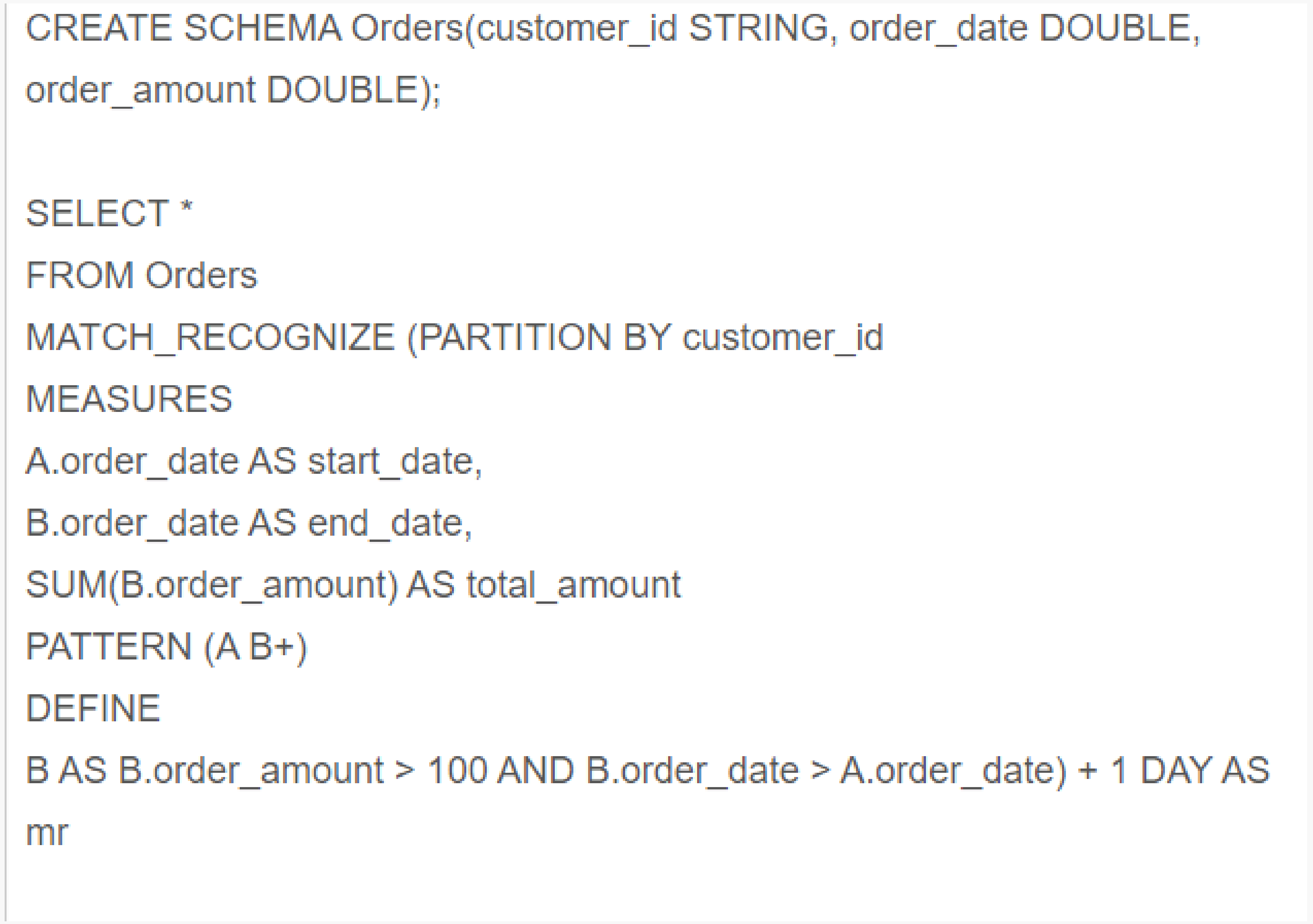
/04

## Illustrative example

The following example illustrates how the Match Recognize works.



```
Orders={customer_id='1', order_date=10,order_amount=20}

t=t.plus(5 seconds)

Orders={customer_id='1', order_date=10,order_amount=50}

t=t.plus(5 seconds)

Orders={customer_id='2', order_date=10,order_amount=109}

t=t.plus(5 seconds)

Orders={customer_id='3', order_date=10,order_amount=70}

t=t.plus(5 seconds)

Orders={customer_id='4', order_date=11,order_amount=207}

t=t.plus(5 seconds)
```



```
CREATE SCHEMA Orders(customer_id STRING, order_date DOUBLE,
order_amount DOUBLE);

SELECT *
FROM Orders
MATCH_RECOGNIZE (PARTITION BY customer_id
MEASURES
A.order_date AS start_date,
B.order_date AS end_date,
SUM(B.order_amount) AS total_amount
PATTERN (A B+)
DEFINE
B AS B.order_amount > 100 AND B.order_date > A.order_date) + 1 DAY AS
mr
```

## Illustrative example

# Syntax summary

1. **SELECT :** It is the keyword that indicates the selection of data from the pattern recognition result.
2. **FROM :** This is the keyword that indicates the source of the events to analyze.
3. **PARTITION BY** : This clause divides the data into partitions based on one or more columns. Data within each partition is processed independently of each other.
4. **ORDER BY** : This clause specifies the order in which events should be processed inside each partition. This is important because match patterns are defined based on the order of events.
5. **MEASURES** : This clause specifies the measurements to be calculated for each detected pattern. Metrics can be aggregations over corresponding events or user-defined functions.
6. **PATTERN** : This clause defines the pattern that the system should look for in the event stream. The pattern is specified using declarative syntax that indicates a sequence of events and the relationships between them.
7. **DEFINE** : This clause defines the variables and conditions used in the pattern defined in the `PATTERN` clause. These variables and conditions can be used to specify additional constraints on the pattern or to perform complex calculations.

## Additional details

**CONCATÉNATION** : In data analysis, "concatenation" refers to the action of combining strings or sequences of data into a single entity. For example, in the context of a football match, concatenation could be used to group different information such as team names, match date and final score into a single string of text. This allows this data to be manipulated and processed more efficiently in analysis or visualization applications.

**ALTERNANCE** : In data analysis, "alternation" refers to the regular change or variation between two different values or states in a data set. This can be observed through sequences of data where values fluctuate between high and low levels, or between distinct categories. An example of alternation in data could be the fluctuation in temperature recorded every hour over the course of a day. You might see an alternation between higher temperatures during the day and lower temperatures at night.