Flink复杂事件处理 (CEP)教程

link作为新一代大数据流式计算引擎，是目前最火爆的技术之一，想从事大数据相关专业的同事，未来的趋势一定是flink流式计算。

其中flink的复杂事件处理（CEP）语句MATCH_RECOGNIZE用于从输入流中识别符合指定规则的事件，并按照指定的方式输出。

CEP语法
SELECT [ALL | DISTINCT]
{ * | variable [,variable]* }
FROM tableExpression
MATCH_RECOGNIZE (
[PARTITION BY { variable [,variable]*}]
ORDER BY {orderItem [, orderItem]*}
MEASURES {measureItem AS col [,variable AS col]*}
ONE ROW PER MATCH
[AFTER MATCH SKIP]
PATTERN (patternVariable[quantifier] [patternVariable[quantifier]]*)
[WITHIN intervalExpression]
DEFINE {patternVariable AS patternDefinationExpression [, patternVar iable AS patternDefinationExpression]*})];

每个 MATCH_RECOGNIZE 查询都包含以下子句：

参数	说明
PARTITION BY(可选)	定义表的逻辑分区；类似于 GROUP BY 操作。 注意：强烈建议对传入数据进行分区，否则 MATCH_RECOGNIZE 子句将被转换为非并行 operator 以确保全局排序。
ORDER BY(必选)	指定如何排序行，这是必不可少的，因为模式取决于排序。但必须以 EVENT TIME 列或者 PROCESS TIME 列作为排序的首列。
MEASURES(必选)	定义如何根据匹配成功的输入事件构造输出事件。类似于 SELECT 子句。
ONE ROW PER MATCH (可选)	对于每一次成功的匹配，产生一行输出事件。
AFTER MATCH SKIP （可选）	指定下一个匹配的开始位置；这是一种控制单个事件属于多少个不同匹配的方法。
PATTERN(必选)	定义待识别的事件序列需要满足的规 则，需要定义在()中，由一系列自定义的 patternVariable 构成。允许使用类似正则表达式的语法构造将要搜索的模式。
DEFINE(必选)	此部分定义（匹配）模式变量必须满足的条件。

CEP语句详解

(1)Partitioning(分区)

可以在分区数据中查找模式，例如，单个股票或特定用户的趋势。 这可以使用PARTITION BY子句表示。 该子句类似于使用GROUP BY进行聚合。

注意：强烈建议对传入数据进行分区，否则MATCH_RECOGNIZE子句将被转换为非并行operator以确保全局排序。且只支持字段，不支持函数。

(2)Order of Events(事件排序)

允许根据时间搜索模式; 处理时间或事件时间。

事件时间的情况下，事件会被排序，在传入内部匹配状态机之前。因此，无论行追加到表的顺序如何，生成的输出都是正确的。按照每行中包含的时间指定的顺序的评估模式。

MATCH_RECOGNIZE子句假定时间属性具有升序排序作为ORDER BY子句的第一个参数。
(3)Defining a Pattern(定义模式)
MATCH_RECOGNIZE子句允许用户使用功能强大且富有表现力的语法搜索事件流中的模式，该语法与广泛使用的正则表达式语法类似。
每个模式都是由基本构建块构建的，称为模式变量，可以应用运算符（量词和其他修饰符）。整个模式必须括在括号中。
示例模式可能如下所示：
PATTERN (A B+ C* D)
可以使用以下运算符：
连接 – 像（A B）这样的模式意味着A和B之间的连续性是严格的。因此，不存在未映射到A或B之间的行。
(4)Greedy & Reluctant Quantifiers(贪婪量词和非贪婪量词)
每个量词可以是贪婪的（默认行为）或非贪婪的。贪婪量词试图匹配尽可能多的行，而非贪婪的量词试图尽可能少地匹配。
量词：用于指定符合pattern中定义的事件的出现次数。
(5)Time constraint(时间限制)
特别是对于流式使用情况，通常要求模式在给定的时间段内完成。这允许限制Flink必须在内部维护的整体状态大小，即使在贪婪量词的情况下也是如此。
因此，Flink SQL支持WITHIN用于定义模式的时间约束的附加（非标准SQL）子句。该子句可以在该PATTERN子句之后定义，并采用毫秒分辨率的间隔。
如果潜在匹配的第一个和最后一个事件之间的时间长于给定值，则此匹配不会附加到结果表。
注意：通常鼓励使用该WITHIN子句，因为它有助于Flink进行有效的内存管理。(6)Aggregations(聚合)
聚合可以用在DEFINE和MEASURES子句中。支持内置和自定义用户定义的功能。
(7)After Match Strategy(跳过策略)
AFTER MATCH SKIP子句指定在找到完整匹配后开始新匹配过程的位置。
有四种不同的策略：
若不写默认模式是: SKIP TO NEXT ROW
SKIP PAST LAST ROW –匹配成功之后，从匹配成功的事件序列中的最后一个事件的下一个事件开始进行下一次匹配。.
SKIP TO NEXT ROW –匹配成功之后，从匹配成功的事件序列中的第一个事件的下一个事件开始进行下一次匹配。(默认模式).
SKIP TO LAST variable –匹配成功之后，从匹配成功的事件序列中最后一个对应于变量的事件开始进行下一次匹 配。
SKIP TO FIRST variable –匹配成功之后，从匹配成功的事件序列中第一个对应于变量的事件开始进行下一次匹配。

案例描述：
当相同的card_id在十分钟内，从两个不同的location发生刷卡现象，就会触发报警机制，以便于监测信用卡盗刷等现象。
数据源如下图：
定义计算逻辑：
select

```
start_timestamp,
end_timestamp,
card_id,
event
from stream_CEP_pyy01
MATCH_RECOGNIZE (
PARTITION BY card_id        -- 按card_id分区,将相同卡号的数据分到同一个计算节点上。
ORDER BY timestamp          -- 在窗口内，对事件时间进行排序。
MEASURES                    --定义如何根据匹配成功的输入事件构造输出事件。
e2.name as  e2name,                 --事件e2的name称作event
e1.timestamp as start_timestamp,
LAST(e2.timestamp) as end_timestamp         --最新的事件时间为 end_timestamp。
ONE ROW PER MATCH                           --匹配成功输出一条。
AFTER MATCH SKIP TO NEXT ROW                --匹配后跳转到下一行。
PATTERN (e1 e2+?) WITHIN INTERVAL '10' MINUTE     -- 定义两个事件e1和e2。
DEFINE                      --定义在PATTERN中出现的变量的具体含义。
e1 as e1. name = 'Tom',                     --事件一的name标记为Tom。
e2 as e2. name = 'Tom' and e2.location1 <> e1.location1 --事件二的name标记为Tom,且事件一和事件二的位置不一样。
);
```

测试结果如图所示：

测试结果：

start_timestamp	end_timestamp	card_id	event
2018-04-13 20:00:00.0	2018-04-13 20:05:00.0	1	Tom
2018-04-13 20:05:00.0	2018-04-13 20:10:00.0	1	Tom

timestamp (TIMESTAMP)	card_id(VARCHAR)	Location1(VARCHAR)	name (VARCHAR)
2018-04-13 12:00:00	1	BJ	Tom
2018-04-13 12:05:00	1	SH	Tom
2018-04-13 12:10:00	1	GZ	Tom
2018-04-13 12:20:00	1	BJ	Tom