Flink sql 完成CEP操作地址

https://help.aliyun.com/document_detail/73845.html

实时计算 > Flink SQL开发指南 > Flink SQL > QUERY语句 > 复杂事件处理（CEP）语句

搜本产品 ▾ 请输入关键词

复杂事件处理（CEP）语句

更新时间：2020-03-21 17:36:18

编辑 · 我的收藏

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。

语法

```
SELECT [ ALL | DISTINCT ]
{ * | projectItem [, projectItem ]* }
FROM tableExpression
[MATCH_RECOGNIZE (
[PARTITION BY {partitionItem [, partitionItem]*}]
[ORDER BY {orderItem [, orderItem]*}]
[MEASURES {measureItem AS col [, measureItem AS col]*}]
[ONE ROW PER MATCH|ALL ROWS PER MATCH|ONE ROW PER MATCH WITH TIMEOUT ROWS|ALL ROWS PER MATCH WITH T
[AFTER MATCH SKIP]
PATTERN (patternVariable[quantifier] [ patternVariable[quantifier]]*) WITHIN intervalExpression
DEFINE {patternVariable AS patternDefinationExpression [, patternVariable AS patternDefinationExpression]
)];
```

PARTITION BY	分区的列，可选项。
ORDER BY	可指定多列，但是必须以 `EVENT TIME` 列或者 `PROCESS TIME` 列作为排序的首列，可选项。
MEASURES	定义如何根据匹配成功的输入事件构造输出事件。
ONE ROW PER MATCH	对于每一次成功的匹配，只会产生一个输出事件。
ONE ROW PER MATCH WITH TIMEOUT ROWS	除了匹配成功的时候产生输出外，超时的时候也会产生输出。超时时间由**PATTERN**语句中的**WITHIN**语句定义。
ALL ROWS PER MATCH	对于每一次成功的匹配，对应于每一个输入事件，都会产生一个输出事件。
ALL ROWS PER MATCH WITH TIMEOUT ROWS	除了匹配成功的时候产生输出外，超时的时候也会产生输出。超时时间由**PATTERN**语句中的**WITHIN**语句定义。
[ONE ROW PER MATCH|ALL ROWS PER MATCH|ONE ROW PER MATCH WITH TIMEOUT ROWS|ALL ROWS PER MATCH WITH TIMEOUT ROWS]	为可选项，默认为 `ONE ROW PER MATCH` 。
AFTER MATCH 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。
AFTER MATCH 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。
AFTER MATCH SKIP TO FIRST	匹配成功之后，从匹配成功的事件序列中第一个对应于patternItem的事件开

示例

- 示例语法

```
SELECT *
FROM Ticker MATCH_RECOGNIZE (
PARTITION BY symbol
ORDER BY tstamp
MEASURES STRT.tstamp AS start_tstamp,
LAST(DOWN.tstamp) AS bottom_tstamp,
LAST(UP.tstamp) AS end_tstamp
ONE ROW PER MATCH
AFTER MATCH SKIP TO NEXT ROW
PATTERN (STRT DOWN+ UP+) WITHIN INTERVAL '10' SECOND
DEFINE
```

```
    DOWN AS DOWN.price < PREV(DOWN.price),
UP AS UP.price > PREV(UP.price)    );
```

- 测试数据

timestamp（TIMESTAMP）	card_id(VARCHAR)	location（VARCHAR）
2018-04-13 12:00:00	1	Beijing
2018-04-13 12:05:00	1	Shanghai
2018-04-13 12:10:00	1	Shenzhen
2018-04-13 12:20:00	1	Beijing

- 测试案例语法

```
CREATE TABLE datahub_stream (
    `timestamp`               TIMESTAMP,
    card_id                   VARCHAR,
    location                  VARCHAR,
    `action`                  VARCHAR,
    WATERMARK wf FOR `timestamp` AS withOffset(`timestamp`, 1000)
) WITH (
    type = 'datahub'
    ...
);
CREATE TABLE rds_out (
    start_timestamp              TIMESTAMP,
    end_timestamp                TIMESTAMP,
    card_id                      VARCHAR,
    event                        VARCHAR
) WITH (
    type= 'rds'
    ...
);

--案例描述：当相同的card_id在十分钟内，在两个不同的location发生刷卡现象，就会触发报警

--定义计算逻辑。
insert into rds_out
select
`start_timestamp`,
`end_timestamp`,
card_id, `event`
from datahub_stream
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
    MEASURES               --定义如何根据匹配成功的输入事件构造输出事件。
        e2.`action` as `event`,
        e1.`timestamp` as `start_timestamp`,   --第一次的事件时间为start_tim
        LAST(e2.`timestamp`) as `end_timestamp`--最新的事件时间为end_timesta
    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patternVariable的具体含
        e1 as e1.action = 'Consumption',    --事件一的action标记为Consumpti
        e2 as e2.action = 'Consumption' and e2.location <> e1.location --
);
```

- 测试结果

start_timestamp（TIMESTAMP）	end_timestamp（TIMESTAMP）	card_id（VARCI
2018-04-13 12:00:00.0	2018-04-13 12:05:00.0	1
2018-04-13 12:05:00.0	2018-04-13 12:10:00.0	1