Flink SQL中的窗口函数

1 OVER窗口

OVER窗口（OVER Window）是传统数据库的标准开窗，不同于GROUP BY Window，OVER Window中的每一个元素都对应一个窗口。窗口元素是与当前元素相邻的元素集合，流数据元素分布在多个窗口中。在Flink SQL Window的实现中，每个触发计算的元素所确定的行，都是该元素所在窗口的最优一行。

在应用OVER Window的流式数据中，每一个元素都对应一个OVER Window。每一个元素都触发一次数据计算。在实时计算的底层实现中，OVER Window的数据进行全局同一管理（数据只存储一份），逻辑上为每一个元素维护一个OVER Window，为每一个元素进行窗口计算，完成计算后会清楚过期的数据。

1.1 语法

```
SELECT
    agg1(col1) OVER (definition1) AS colName,
    ...
    aggN(colN) OVER (definition1) AS colNameN
FROM Tab1;
```

注意：① agg1到aggN所对应的OVER definition1必须相同；② 外层SQL可以通过AS的别名查询数据。

1.2 类型

Flink SQL中对OVER Window的定义遵循标准SQL的定义语法，传统的OVER Window没有对其进行更细粒度的窗口类型命名划分。按照计算行的定义方式，OVER Window可以分为以下两类：

① ROWS OVER Window：每一行元素都被视为新的计算行，即每一行都是一个新的窗口；

② RANGE OVER Window：具有相同时间值的所有元素行视为同一计算行，即具有相同时间值的所有行都是同一个窗口。

1.3 属性

正交属性	说明	P
rows	按照实际元素的行确定窗口	

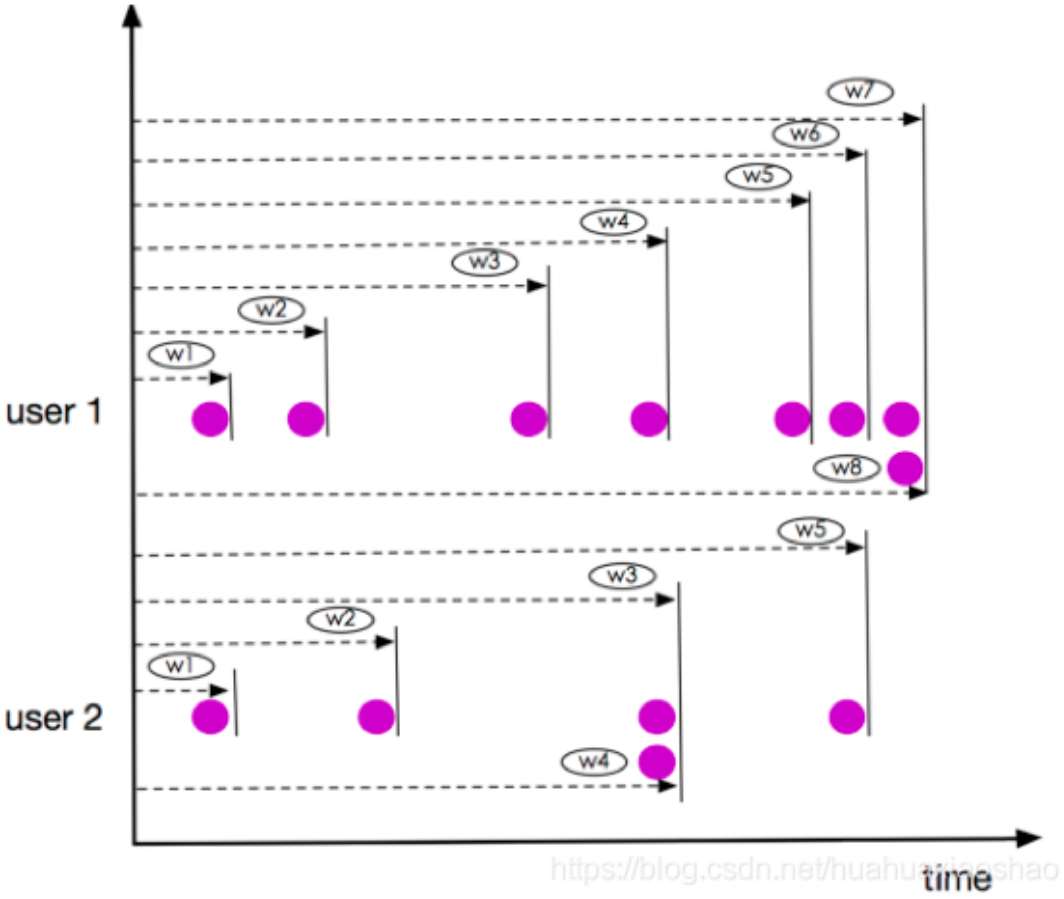
range	按照实际的元素值（时间戳值）确定窗口	

1.4 Rows OVER Window

（1）窗口数据

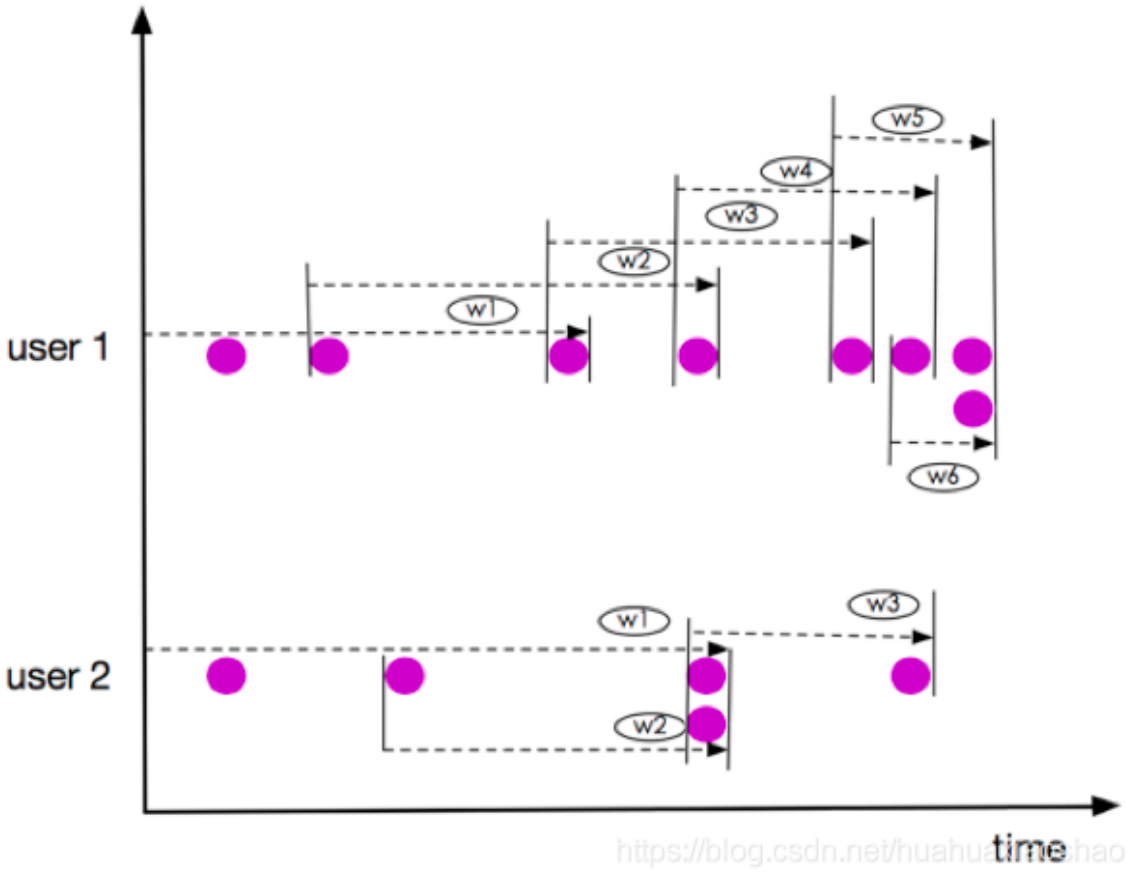
ROWS OVER Windows的每个元素都确定一个窗口。ROWS OVER Window分为Unbounded（无界流）和Bounded（有界流）两种情况。

Unbounded ROWS OVER Window数据示例如下图所示。



虽然上图所示窗user1的w7、w8及user2的窗口w3、w4都是同一时刻到达，但它们仍然在不同的窗口，这一点与RANGE OVER Window不同。

Bounded ROWS OVER Window数据以3个元素（2 PRECEDING）的窗口为例，如下图所示。



虽然上图所示窗口user1的w5、w6及user2的窗口w2、w3都是同一时刻到达，但它们仍然在不同的窗口，这一点与RANGE OVER Window不同。

（2）窗口语法

```
SELECT
    agg1(col1) OVER(
     [PARTITION BY (value_expression1,..., value_expressionN)]
     ORDER BY timeCol
     ROWS
     BETWEEN (UNBOUNDED | rowCount) PRECEDING AND CURRENT ROW) AS colNam
e, ...
FROM Tab1;
```

① value_expression：分区值表达式。
② timeCol：元素排序的时间字段。
③ rowCount：定义根据当前行开始向前追溯几行元素。
输出在当前商品上架之前同类的3个商品中的最高价格。

```
SELECT
    itemID,
    itemType,
    onSellTime,
    price,
    MAX(price) OVER (
```

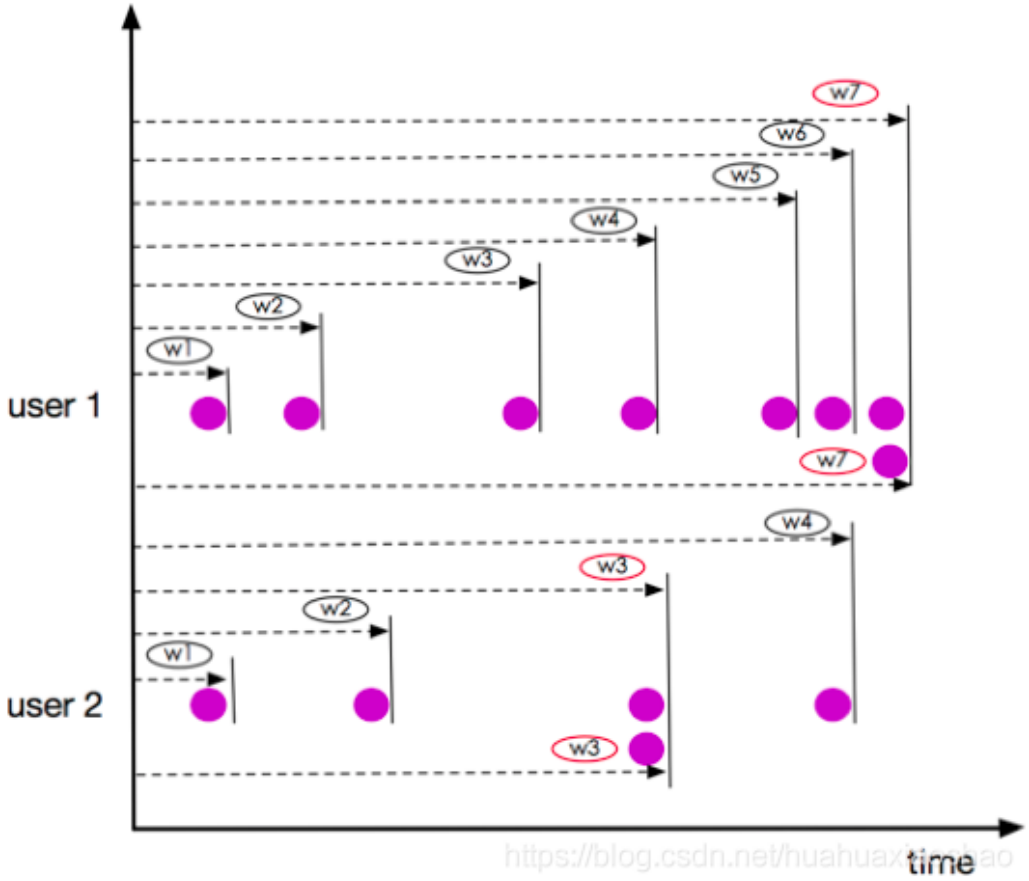
```
        PARTITION BY itemType
        ORDER BY onSellTime
        ROWS BETWEEN 2 preceding AND CURRENT ROW) AS maxPrice
  FROM tmall_item;
```

1.5 RANGE OVER Window

（1）窗口数据

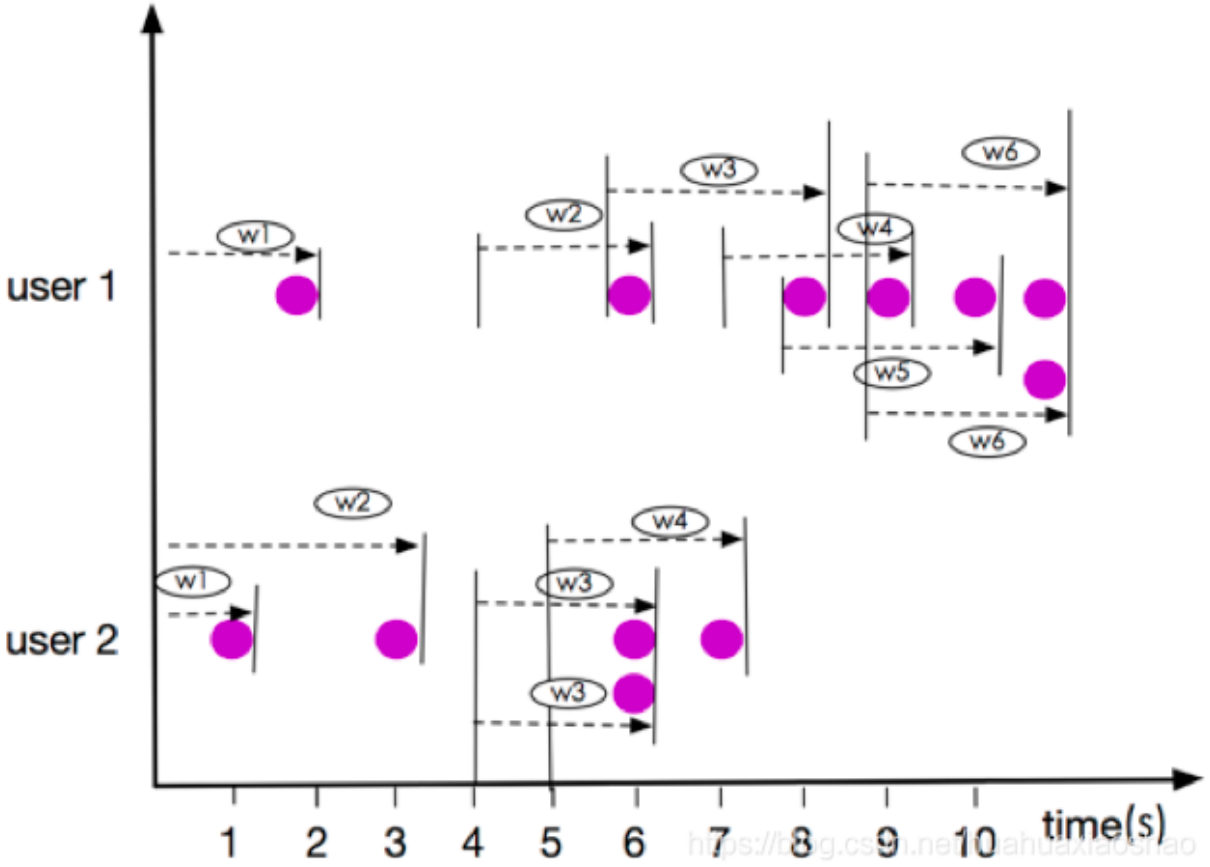
RANGE OVER Window所有具有共同元素值（元素时间戳）的元素行确定一个窗口，RANGE OVER Window分为Unbounded和Bounded的两种情况。

Unbounded RANGE OVER Window数据示例如下图所示。



上图所示窗口user1的w7、user2的窗口w3，两个元素同一时刻到达，属于相同的window，这一点与ROWS OVER Window不同。

Bounded RANGE OVER Window数据，以3秒中数据(INTERVAL ‘2’ SECOND)的窗口为例，如下图所示。



上图所示窗口user1的w6、user2的窗口w3，元素都是同一时刻到达，属于相同的window，这一点与ROWS OVER Window不同。

（2）窗口语法

```
SELECT
    agg1(col1) OVER(
     [PARTITION BY (value_expression1,..., value_expressionN)]
     ORDER BY timeCol
     RANGE
     BETWEEN (UNBOUNDED | timeInterval) PRECEDING AND CURRENT ROW) AS co
lName,
...
FROM Tab1;
```

① value_expression：进行分区的字表达式。

② timeCol：元素排序的时间字段。

③ timeInterval：定义根据当前行开始向前追溯指定时间的元素行。

求比当前商品上架时间早2分钟的同类商品中的最高价格。

```
SELECT
    itemID,
    itemType,
    onSellTime,
    price,
```

```
    MAX(price) OVER (
        PARTITION BY itemType
        ORDER BY onSellTime
        RANGE BETWEEN INTERVAL '2' MINUTE preceding AND CURRENT ROW) AS
maxPrice
  FROM tmall_item;
```

2 分组窗口

2.1 分组窗口的类型

SQL查询的分组窗口是通过GROUP BY子句定义的。类似于使用常规GROUP BY语句的查询，窗口分组语句的GROUP BY子句中带有一个窗口函数为每个分组计算出一个结果。以下是批处理和流出理表支持的分组窗口函数：

（1）TUMBLE(time_attr, interval)

定义一个滚动窗口。滚动窗口把行分配到有固定持续时间（interval）的不重叠的连续窗口。比如，5分钟的滚动窗口以5分钟为间隔对行进行分组。滚动窗口可以定义在事件时间（批处理、流处理）或处理时间（流处理）上。

（2）HOP(time_attr, interval, interval)

定义一个跳跃窗口（在Table API中成为滑动窗口）。滑动窗口有一个固定的持续时间（第二个interval参数）以及一个滑动的间隔（第一个interval参数）。若滑动间隔小于窗口的持续时间，滑动窗口则会出现重叠；因此，行将会被分配到多个窗口中。比如，一个大小为15分钟的滑动窗口，其滑动间隔为5分钟，将会把每一行数据分配到3个15分钟的窗口中。滑动窗口可以定义咋事件时间（批处理、流处理）或处理时间（流处理）上。

（3）SESSION(time_attr, interval)

定义一个会话时间窗口。会话时间窗口没有一个固定的持续时间，但是它们的边界会根据interval所定义的不活跃时间所确定；即一个会话时间窗口在定义的间隔时间内没有新纪录出现，该窗口会被关闭。例如时间窗口的间隔时间是30分钟，当其不活跃的时间达到30分钟后，若观测到新的记录，则会启动一个新的会话时间窗口（否则该行数据会被添加到当前的窗口），且若在30分钟内没有观测到新纪录，这个窗口将会被关闭。会话时间窗口可以使用事件时间（批处理、流处理）或处理时间（流处理）。

2.2 时间属性

在流处理表中的SQL查询中，分组窗口函数的time_attr参数必须引用一个合法的时间属性，且该属性需要指定行的处理时间或事件时间。

对于批处理的SQL查询，分组窗口函数的time_attr参数必须是一个TIMESTAMP类型的属性。

2.3 选择分组窗口的开始和结束时间戳（辅助函数）

（1）返回相对应的滚动、滑动和会话窗口的开始时间（包含边界）

```
TUMBLE_START(time_attr, interval)
HOP_START(time_attr, interval, interval)
SESSION_START(time_attr, interval)
```

（2）返回相对应的滚动、滑动和会话窗口的结束时间（包含边界）

```
TUMBLE_END(time_attr, interval)
HOP_END(time_attr, interval, interval)
SESSION_END(time_attr, interval)
```

注意：返回的间戳不可以在随后基于事件的操作中，作为行时间属性使用，比如基于事件窗口的join以及分组窗口或分组窗口上的聚合。

（3）返回相对应的滚动、滑动和会话窗口的结束时间（不包含边界）

```
TUMBLE_ROWTIME(time_attr, interval)
HOP_ROWTIME(time_attr, interval, interval)
SESSION_ROWTIME(time_attr, interval)
```

返回的是一个可用于后续需要基于时间的操作的时间属性（rowtime attribute），比如基于时间窗口的join以及分组窗口或分组窗口上的聚合。

（4）返回相对应的滚动、滑动和会话窗口的结束时间（不包含边界）

```
TUMBLE_PROCTIME(time_attr, interval)
HOP_PROCTIME(time_attr, interval, interval)
SESSION_PROCTIME(time_attr, interval)
```

返回处理时间参数可用于后续需要基于事件的操作，比如基于时间窗口的join以及分组窗口或分组窗口上的聚合。

注意：辅助函数必须使用与GROUP BY子句中的分组窗口函数完全相同的参数来调用。

2.4 实战

以下的例子展示了如何在流处理表中指定使用分组庄口函数的SQL查询。

（1）计算每日的 SUM(amount)（使用事件时间）

```
SELECT user,
TUMBLE_START(rowtime,INTERVAL ‘1’ DAY) as wStart,
SUM(amount) FROM Orders
GROUP BY TUMBLE(rowtime,INTERVAL ‘1’ DAY),user;
```

（2）计算每日的SUM(amount)（使用处理时间）

```
SELECT user,
SUM(amount) FROM Orders
GROUP BY TUMBLE(proctime,INTERVAL '1' DAY),user;
```

（3）使用事件时间计算过去24小时中每小时的 SUM(amount)

```
SELECT product,SUM(amount)
FROM Orders
GROUP BY HOP(rowtime,INTERVAL '1' HOUR,INTERVAL '1' DAY),product;
```

（4）计算每个以12小时（事件时间）作为不活动时间的会话的SUM(amount)

```
SELECT user,
SESSION_START(rowtime,INTERVAL '12' HOUR) as sStart,
SESSION_ROWTIME(rowtime,INTERVAL '12' HOUR) as snd,
SUM(amount)
FROM Orders
GROUP BY SESSION(rowtime,INTERVAL '12' HOUR),user;
```

3 总结

本章主要是对Flink SQL中的窗口进行了介绍。窗口让我们可以很方便地解决一些实际应用中产生的问题。但是，我觉得我对其掌握的熟练程度不高，有时候，换了一个场景，可能反应不到用哪个类型的窗口会更好一些。后续，我会勤加练习，多了解其在不同场景下的应用。