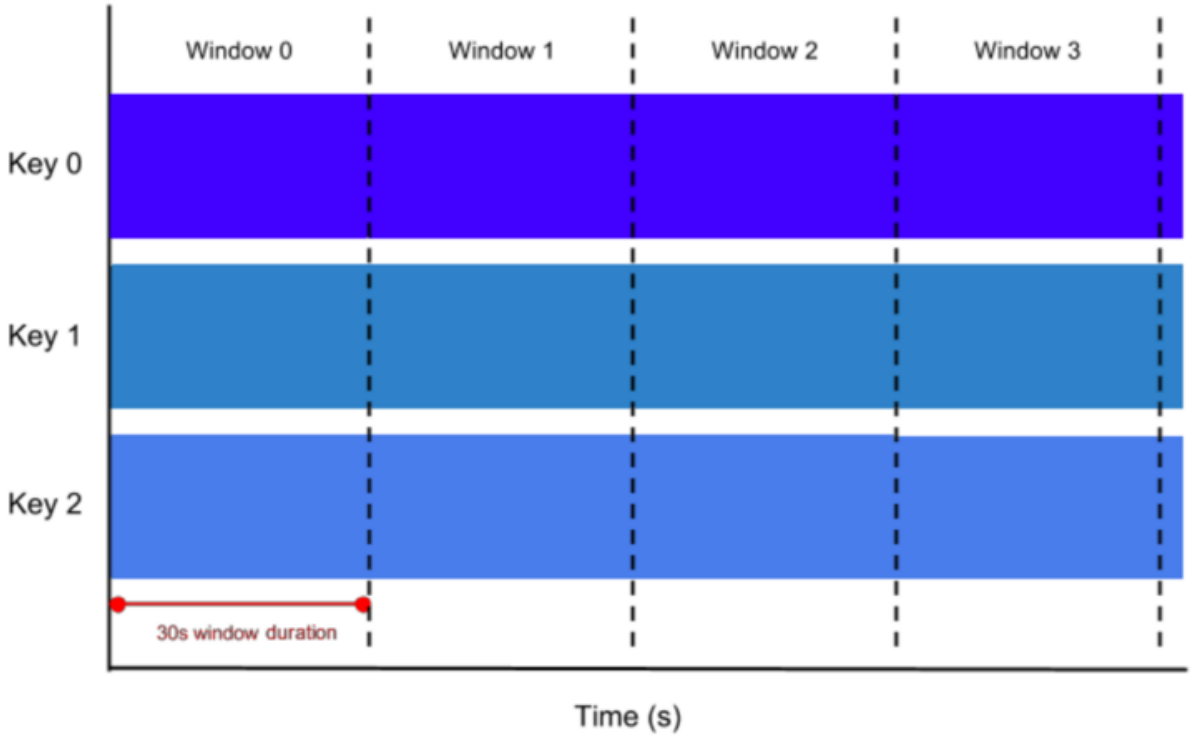
Flink学习--Flink SQL 窗口函数

flink窗口函数包含滚动窗口、滑动窗口、会话窗口和OVER窗口

滚动窗口

滚动窗口（TUMBLE）将每个元素分配到一个指定大小的窗口中。通常，滚动窗口有一个固定的大小，并且不会出现重叠。例如，如果指定了一个5分钟大小的滚动窗口，无限流的数据会根据时间划分为`[0:00 - 0:05)`、`[0:05, 0:10)`、`[0:10, 0:15)`等窗口。下图展示了一个30秒的滚动窗口。



使用标识函数选出窗口的起始时间或者结束时间，窗口的时间属性用于下级Window的聚合。

窗口标识函数	返回类型	描述
`TUMBLE_START(time-attr, size-interval)`	TIMESTAMP	返回窗口的起始时间（包含边界）。例如`[00:10, 00:15)`窗口，返回`00:10`。
`TUMBLE_END(time-attr, size-interval)`	TIMESTAMP	返回窗口的结束时间（包含边界）。例如`[00:00, 00:15]`窗口，返回`00:15`。
`TUMBLE_ROWTIME(time-attr,`		返回窗口的结束时间（不包含边界）。例如`[00:00, 00:15]`窗口，返回`00:14:59.999`。返回值是一个rowtime

`size-interval)`	TIMESTAMP(rowtime-attr)	attribute，即可以基于该字段做时间属性的操作，例如，级联窗口只能用在基于Event Time的Window上
`TUMBLE_PROCTIME(time-attr, size-interval)`	TIMESTAMP(rowtime-attr)	返回窗口的结束时间（不包含边界）。例如`[00:00, 00:15]`窗口，返回`00:14:59.999`。返回值是一个proctime attribute，即可以基于该字段做时间属性的操作，例如，级联窗口只能用在基于Processing Time的Window上

TUMBLE window示例

```
import org.apache.flink.api.common.typeinfo.TypeHint;
import org.apache.flink.api.common.typeinfo.TypeInformation;
import org.apache.flink.api.java.tuple.Tuple3;
import org.apache.flink.streaming.api.TimeCharacteristic;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.datastream.SingleOutputStreamOperator;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.api.functions.timestamps.AscendingTimestampExtr
import org.apache.flink.table.api.EnvironmentSettings;
import org.apache.flink.table.api.Table;
import org.apache.flink.table.api.bridge.java.StreamTableEnvironment;


import java.sql.Timestamp;
import java.util.Arrays;

public class TumbleWindowExample {

    public static void main(String[] args) throws Exception {

        /**
         * 1 注册环境
         */
        EnvironmentSettings mySetting = EnvironmentSettings
                .newInstance()
//                .useOldPlanner()
```

sql逻辑，每十秒钟聚合
执行结果：
(2019-11-01 06:53:00.0,2019-11-01 06:53:10.0,603)
(2019-11-01 06:53:20.0,2019-11-01 06:53:30.0,208)
(2019-11-01 06:53:10.0,2019-11-01 06:53:20.0,204)

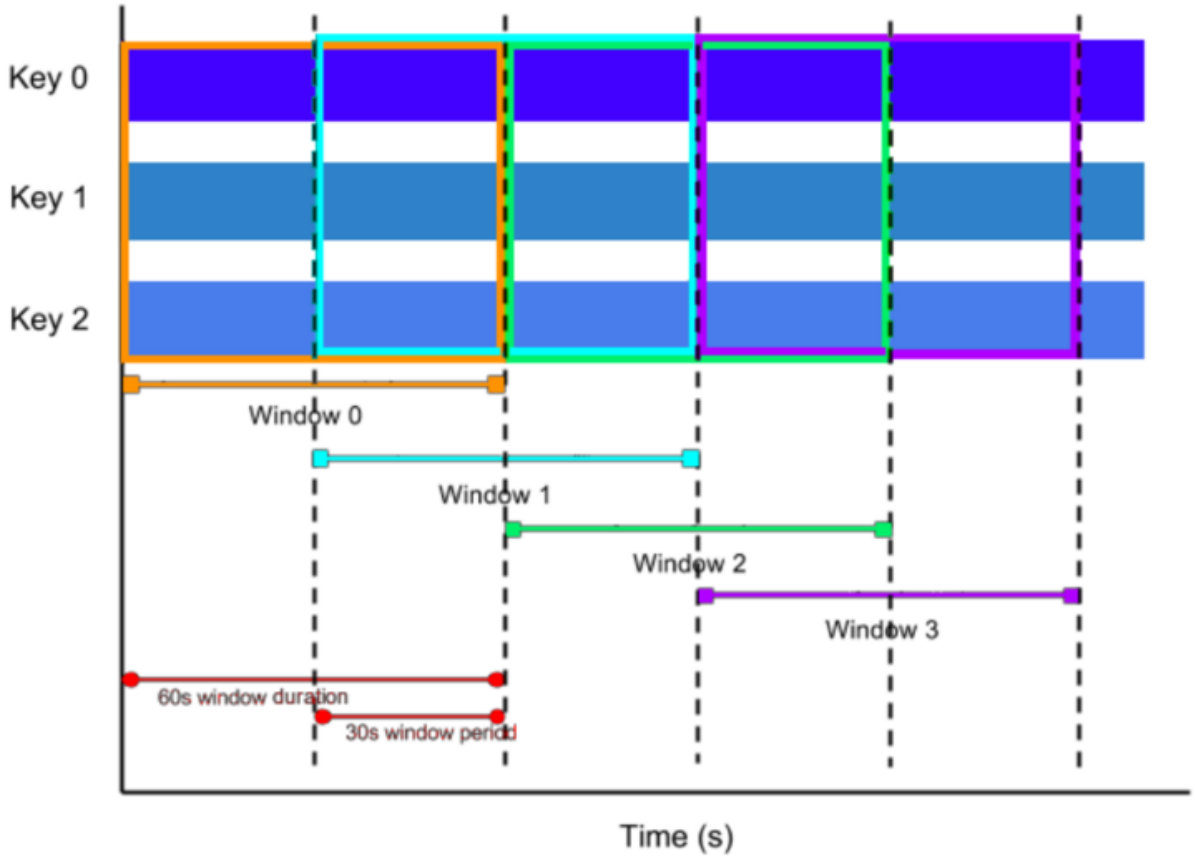
滑动窗口

滑动窗口（HOP），也被称作Sliding Window。不同于滚动窗口，滑动窗口的窗口可以重叠。

滑动窗口有两个参数：slide和size。slide为每次滑动的步长，size为窗口的大小。

- slide < size，则窗口会重叠，每个元素会被分配到多个窗口。
- slide = size，则等同于滚动窗口（TUMBLE）。
- slide > size，则为跳跃窗口，窗口之间不重叠且有间隙。

通常，大部分元素符合多个窗口情景，窗口是重叠的。因此，滑动窗口在计算移动平均数（moving averages）时很实用。例如，计算过去5分钟数据的平均值，每10秒钟更新一次，可以设置slide为10秒，size为5分钟。下图为您展示间隔为30秒，窗口大小为1分钟的滑动窗口。



使用滑动窗口标识函数选出窗口的起始时间或者结束时间，窗口的时间属性用于下级Window的聚合。

窗口标识函数	返回类型	描述
`HOP_START (<time-attr>, <slide-interval>, <size-interval>)`	TIMESTAMP	返回窗口的起始时间（包含边界）。例如`[00:10, 00:15)`窗口，返回`00:10`。
`HOP_END (<time-attr>, <slide-interval>, <size-interval>)`	TIMESTAMP	返回窗口的结束时间（包含边界）。例如`[00:00, 00:15)`窗口，返回`00:15`。
`HOP_ROWTIME (<time-attr>, <slide-interval>, <size-interval>)`	TIMESTAMP（rowtime-attr）	返回窗口的结束时间（不包含边界）。例如`[00:00, 00:15)`窗口，返回`00:14:59.999`。返回值是一个rowtime attribute，即可以基于该字段做时间类型的操作，

		只能用在基于event time的window上。
`HOP_PROCTIME (<time-attr>, <slide-interval>, <size-interval>)`	TIMESTAMP（rowtime-attr）	返回窗口的结束时间（不包含边界）。例如`[00:00, 00:15)`窗口，返回`00:14:59.999`。返回值是一个proctime attribute

滑动窗口实例：
java代码同上，sql语句改为：

```
SELECT HOP_START(t, INTERVAL '5' SECOND, INTERVAL '10' SECOND) AS window_start," +
                "HOP_END(t, INTERVAL '5' SECOND, INTERVAL '10' SECOND) AS window_
                + logT + " GROUP BY HOP(t, INTERVAL '5' SECOND, INTERVAL '10' SEC
```

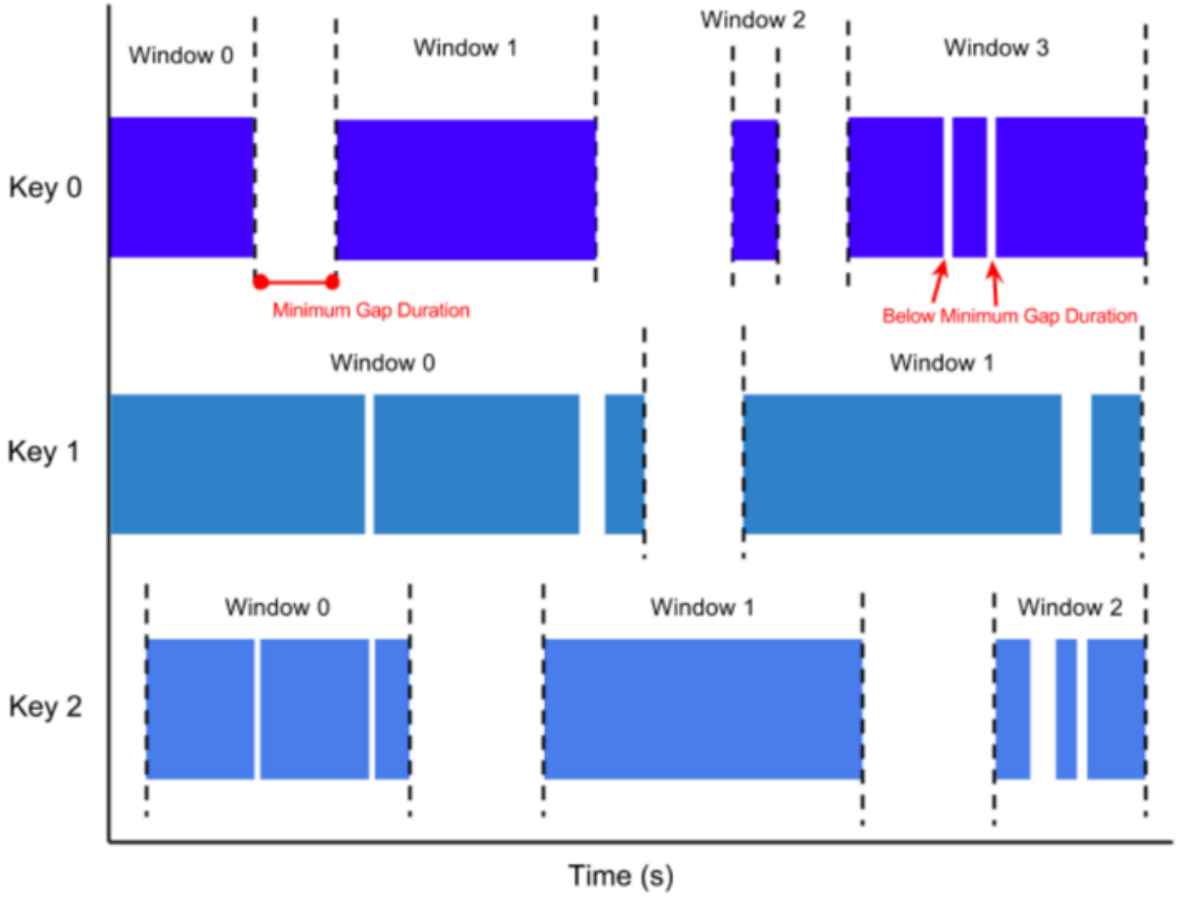
每间隔5秒统计10秒内的数据
sql结果如下：
(2019-11-01 06:53:15.0,2019-11-01 06:53:25.0,208)
(2019-11-01 06:53:10.0,2019-11-01 06:53:20.0,204)
(2019-11-01 06:53:05.0,2019-11-01 06:53:15.0,507)
(2019-11-01 06:53:20.0,2019-11-01 06:53:30.0,208)
(2019-11-01 06:53:00.0,2019-11-01 06:53:10.0,603)
(2019-11-01 06:52:55.0,2019-11-01 06:53:05.0,300)

会话窗口

会话窗口（SESSION）通过Session活动来对元素进行分组。会话窗口与滚动窗口和滑动窗口相比，没有窗口重叠，没有固定窗口大小。相反，当它在一个固定的时间周期内不再收到元素，即会话断开时，这个窗口就会关闭。

会话窗口通过一个间隔时间（Gap）来配置，这个间隔定义了非活跃周期的长度。例如，一个表示鼠标点击活动的数据流可能具有长时间的空闲时间，并在两段空闲之间散布着高浓度的点击。 如果数据在指定的间隔（Gap）之后到达，则会开始一个新的窗口。

会话窗口示例如下图。每个Key由于不同的数据分布，形成了不同的Window。



使用标识函数选出窗口的起始时间或者结束时间，窗口的时间属性用于下级Window的聚合。

窗口标识函数	返回类型	描述
`SESSION_START (<time-attr>, <gap-interval>)`	Timestamp	返回窗口的起始时间（包含边界）。如`[00:10, 00:15)`的窗口，返回`00:10`，即为此会话窗口内第一条记录的时间。
`SESSION_END (<time-attr>, <gap-interval>)`	Timestamp	返回窗口的结束时间（包含边界）。如`[00:00, 00:15)`的窗口，返回`00:15`，即为此会话窗口内最后一条记录的时间+`<gap-interval>`。
`SESSION_ROWTIME (<time-attr>, <gap-interval>)`	Timestamp（rowtime-attr）	返回窗口的结束时间（不包含边界）。如`[00:00, 00:15)`的窗口，返回`00:14:59.999`。返回值是一个rowtime attribute，也就是可以基于该字段进行时间类型的操作。该参数只能用于基于event time的window。
		返回窗口的结束时间（不包含边界）。如`[00:00,`

`SESSION_PROCTIME (<time-attr>, <gap-interval>)`	Timestamp（rowtime-attr）	`00:15)` 的窗口，返回 `00:14:59.999` 。返回值是一个 proctime attribute，也就是可以基于该字段进行时间类型的操作。该参数只能用于基于processing time的window 。

会话窗口实例：
java代码同上
sql语句如下：
每隔5秒聚合

```
"SELECT SESSION_START(t, INTERVAL '5' SECOND) AS window_start," +
                "SESSION_END(t, INTERVAL '5' SECOND) AS window_end, SUM(v) FROM "
                + logT + " GROUP BY SESSION(t, INTERVAL '5' SECOND)"
```

sql结果：

(2019-11-01 06:53:21.0,2019-11-01 06:53:26.0,208)
(2019-11-01 06:53:00.0,2019-11-01 06:53:05.0,300)
(2019-11-01 06:53:09.0,2019-11-01 06:53:17.0,507)

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窗口中每1个元素都对应1个窗口。窗口内的元素是当前元素往前多少个或往前多长时间的元素集合，因此流数据元素分布在多个窗口中。

在应用OVER窗口的流式数据中，每1个元素都对应1个OVER窗口。每1个元素都触发1次数据计算，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后1行。在实时计算的底层实现中，OVER窗口的数据进行全局统一管理（数据只存储1份），逻辑上为每1个元素维护1个OVER窗口，为每1个元素进行窗口计算，完成计算后会清除过期的数据。

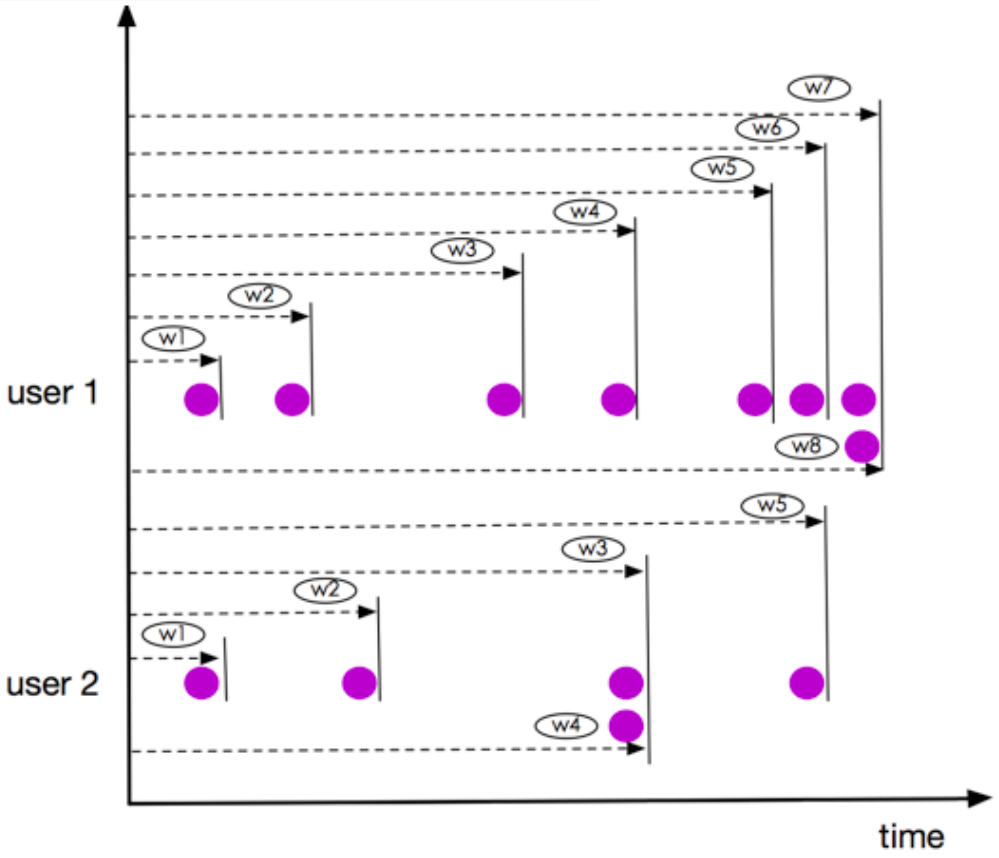
Flink SQL中对OVER窗口的定义遵循标准SQL的定义语法，传统OVER窗口没有对其进行更细粒度的窗口类型命名划分。按照计算行的定义方式，OVER Window可以分为以下两类：

- 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。
- 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。

Rows OVER Window语义

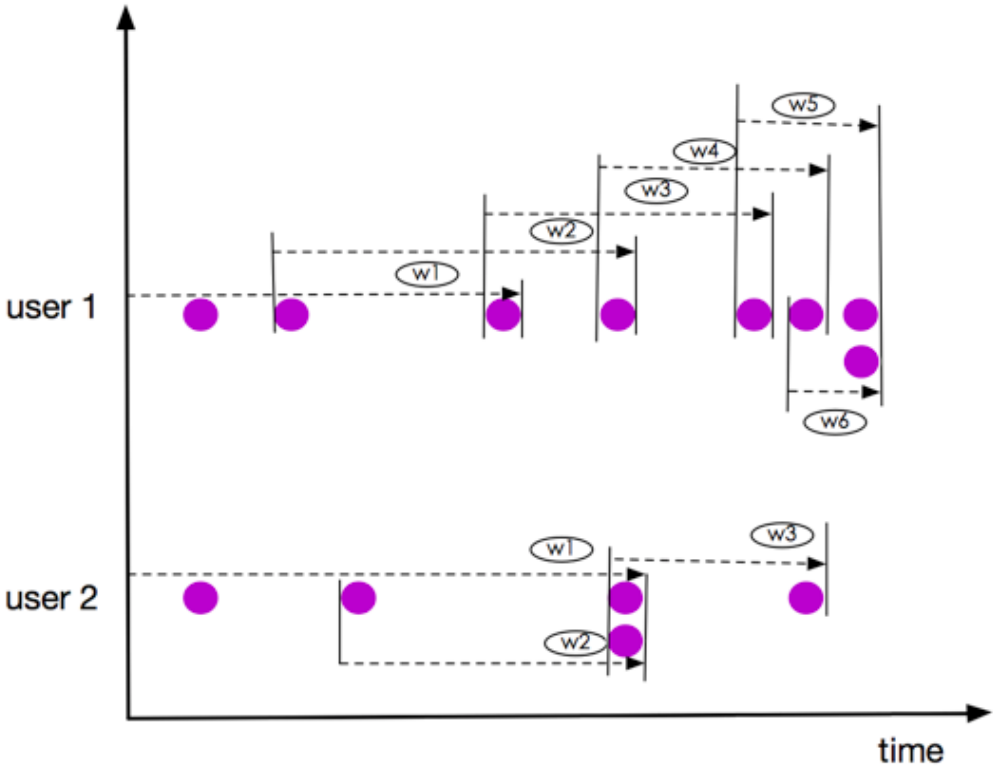
窗口数据

ROWS OVER Window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。



虽然上图所示窗口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（往前2个元素）的窗口为例，如下图所示。

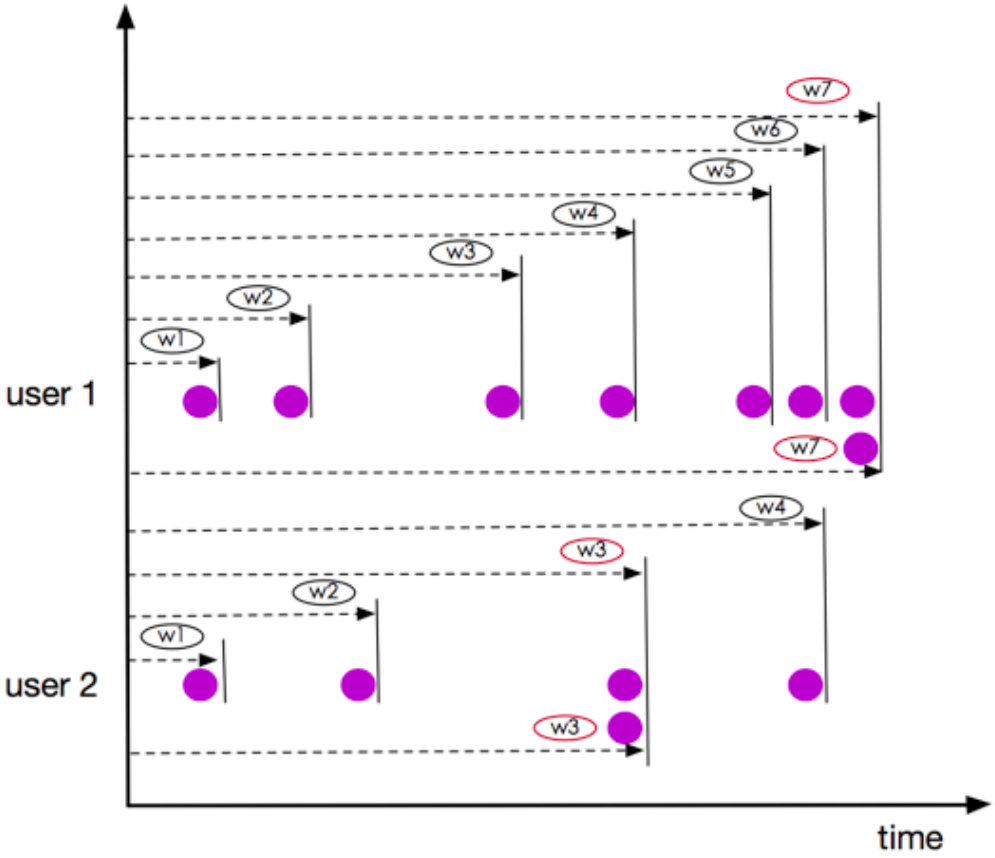


虽然上图所示窗口user1的w5、w6及user2的窗口w1、w2都是同一时刻到达，但它们仍然在不同的窗口，这一点与RANGE OVER Window不同。

RANGE OVER Window语义

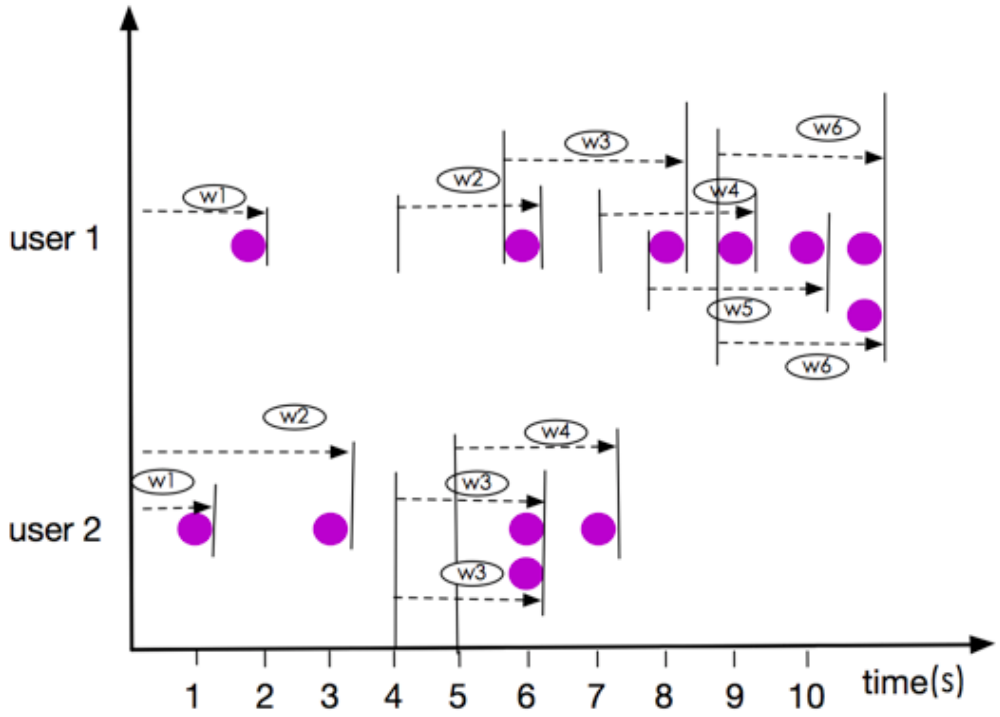
窗口数据

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。

OVER窗口实例：
java代码同上
初始数据如下：

```
// 初始数据
        DataStream<Tuple3<Long, String,Integer>> log = env.fromCollection(Arrays.
                //时间 14:53:00
                new Tuple3<>(1572591180_000L,"xiao_ming",999),
                //时间 14:53:09
                new Tuple3<>(1572591189_000L,"zhang_san",303),
                //时间 14:53:12
                new Tuple3<>(1572591192_000L, "xiao_li",888),
                //时间 14:53:21
                new Tuple3<>(1572591201_000L,"li_si", 908),
                //2019-11-01 14:53:31
                new Tuple3<>(1572591211_000L,"li_si", 555),
                //2019-11-01 14:53:41
                new Tuple3<>(1572591221_000L,"zhang_san", 666),
                //2019-11-01 14:53:51
                new Tuple3<>(1572591231_000L,"xiao_ming", 777),
                //2019-11-01 14:54:01
                new Tuple3<>(1572591241_000L,"xiao_ming", 213),
                //2019-11-01 14:54:11
                new Tuple3<>(1572591251_000L,"zhang_san", 300),
                //2019-11-01 14:54:21
                new Tuple3<>(1572591261_000L,"li_si", 112)
        ));
```

ROWS over Windown sql语句如下：

```
"SELECT name,v,MAX(v) OVER(\n" +
                "PARTITION BY name \n" +
                "ORDER BY t \n" +
                "ROWS BETWEEN 2 PRECEDING AND CURRENT ROW\n" +
                ") FROM " + logT
```

sql结果如下：
(zhang_san,303,303)
(xiao_li,888,888)
(li_si,908,908)
(xiao_ming,999,999)
(zhang_san,666,666)
(li_si,555,908)
(xiao_ming,777,999)
(li_si,112,908)
(zhang_san,300,666)
(xiao_ming,213,999)

RANGE OVER Window sql 语句如下：

```
"SELECT name,v,MAX(v) OVER(\n" +
                "PARTITION BY name \n" +
                "ORDER BY t \n" +
                "RANGE BETWEEN INTERVAL '15' SECOND PRECEDING AND CURRENT ROW\n" +
                ") FROM "+ logT
```

sql结果如下：
(xiao_ming,999,999)
(xiao_li,888,888)
(zhang_san,303,303)
(li_si,908,908)
(li_si,555,908)
(xiao_ming,777,777)
(zhang_san,666,666)
(li_si,112,112)
(xiao_ming,213,777)
(zhang_san,300,300)

本文的java代码来自：
https://github.com/CheckChe08...