Blink Streaming Table API Syntax

Blink is a modified Flink in Alibaba, which is more adapted to our business and more stable. A lot of business want to use Table API which is easier to use. But the Flink Streaming Table API very limited. So we started to extend its features (such as: groupBy,agg, window, join, UDF…) based on Flink 1.0 branch half a year ago. By now, we want to join the community and contribute to the new Flink Table API & SQL. Below is our Table API looks like

Simple Operators

The following operators are identical to operations currently supported by Flink batch and streaming table API.

Operator	Table API
Projection (no aggregation)	val res = tab.select(‘a, ‘b * 2, ‘c as ‘d, ‘t)
Selection	val res = tab.where(‘a < 10) // or alternatively val res = tab.filter(‘a < 10)
Union All	val res = tab1.union(tab2)

Group Operators

Group window operators define row groupings on which aggregation functions can be applied. The following operators’ definition are a little different from Flink’s.

Operator	Table API
grobal aggregation	val res = tab .groupBy(‘a) .select(‘a, ‘b.sum, ‘c.min)
time window aggregation	// tumbling window val res = tab .groupBy(‘a) .time_window(5.Seconds) .select(‘a, ‘b.sum, ‘c.min)

	// sliding window,evaluate the last 10 seconds data every 5 seconds val res = tab .groupBy(‘a) .time_window(10.Seconds, 5.Seconds) .select(‘a, ‘b.sum, ‘c.min)

count window aggregation	// tumbling window
	val res = tab
	.groupBy(‘a)
	.count_window(5)
	.select(‘a, ‘b.sum, ‘c.min)
	// sliding window,evaluate last 10 rows every 5 rows
	val res = tab
	.groupBy(‘a)
	.count_window(10, 5)
	.select(‘a, ‘b.sum, ‘c.min)

- groupBy() is used to group the events of the stream (as in SQL), similar to keyBy() in the DataStream API. It could be renamed to keyBy() to bring Table API closer to the DataStream API. If no groupBy() clause is defined, the whole stream moves over a single window operator, i.e., an allWindow with DOP = 1.
- Aggregation functions are specified in the select() clause. It is not necessary to define a window before aggregation. If we haven’t define window, it will evaluate and emit result for each incoming row like datastream.keyBy(f1).sum(f2). It seems that all aggregation functions are scoped to the defined window in Flink Stream Table API design and Calcite StreamSQL desgin.
- time_window() and count_window() functions are similar to timeWindow and countWindow in DataStream.

Join Operators

Join operator defines how to join two stream table on the specific keys and a window. Currently, we only implement inner join.

Operator	Table API
Join	val res = tab1

	.join(tab2) .where(‘f1 === ‘f2) .time_window(5.Seconds)

Join two data streams on a given key and a common window.

User Defined Function Operators

User can define own functions to apply on fields. Build-in UDFs includes md5, length, concatWs, hash and so on.

To defind your own user defined function, you need to extend UDF, and define a evel function. The following is a example of md5:

```
object md5 extends UDF {
  def eval(field: String): String = {
    if (UDFUtils.isNull(field)) {
      null
    } else {
      DigestUtils.md5Hex(field)
    }
  }
}
```

And in Table API , we can use md5 like this :

```
val res = tab.select(‘a, md5(‘b), ‘c)
```