Flink Table API编程

程鹤群（军长） · 阿里巴巴 / 高级研发工程师

Aache Flink 在线教程

自我介绍

CONTENT

目录 >>

01

什么是Table API

Apache Flink

1.什么是Table API

1.1 Flink API总览

1.2 Table API的特性

1.1 Flink API总览

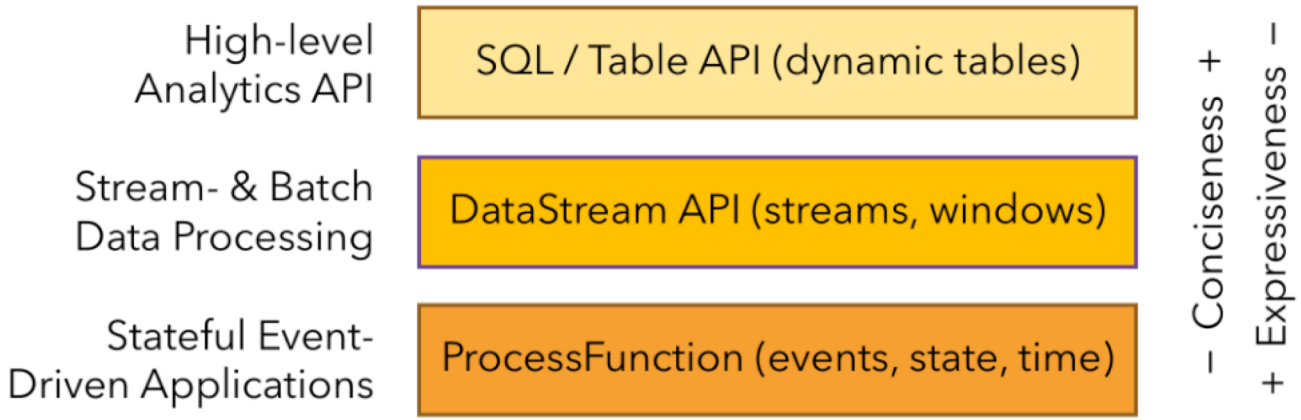


Table API & SQL

- 声明式 – 用户只关心做什么，不用关心怎么做
- 高性能 – 支持查询优化，可以获取更好的执行性能
- 流批统一 – 相同的统计逻辑，既可以流模式运行，也可以批模式运行
- 标准稳定 – 语义遵循SQL标准，不易变动
- 易理解 – 语义明确，所见即所得

Table API:

```
tab.groupBy("word")
   .select("word, count(1) as count")
```

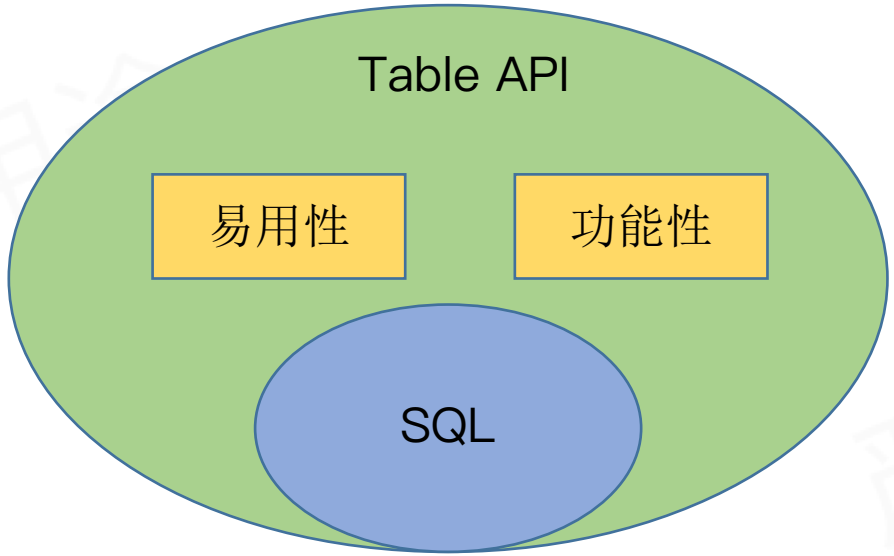
SQL:

```
SELECT word, COUNT(*) AS cnt
FROM MyTable
GROUP BY word
```

Table API 的特点

Table API 特点

- Table API 使得多声明的数据处理写起来比较容易
- Table API 使得扩展标准SQL更为容易（当且仅当需要的时候）



02

Table API编程

Apache Flink

2. Table API 编程

2.1 WordCount 示例

2.2 Table API 操作

- 2.2.1 how to get a Table
- 2.2.2 how to emit a Table
- 2.2.3 how to query a Table

WordCount 示例

https://github.com/hequn8128/TableApiDemo

```
3   import org.apache.flink.api.common.typeinfo.Types;
4   import org.apache.flink.api.java.ExecutionEnvironment;
5   import org.apache.flink.table.api.Table;
6   import org.apache.flink.table.api.java.BatchTableEnvironment;
7   import org.apache.flink.table.descriptors.FileSystem;
8   import org.apache.flink.table.descriptors.OldCsv;
9   import org.apache.flink.table.descriptors.Schema;
10  import org.apache.flink.types.Row;
11
12  public class JavaBatchWordCount {
13
14      public static void main(String[] args) throws Exception {
15          ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
16          BatchTableEnvironment tEnv = BatchTableEnvironment.create(env);
17
18          String path = JavaBatchWordCount.class.getClassLoader().getResource( name: "words.txt").getPath();
19          tEnv.connect(new FileSystem().path(path))
20              .withFormat(new OldCsv().field( fieldName: "word", Types.STRING).lineDelimiter("\n"))
21              .withSchema(new Schema().field( fieldName: "word", Types.STRING))
22              .registerTableSource( name: "fileSource");
23
24          Table result = tEnv.scan( ...strings: "fileSource")
25              .groupBy( s: "word")
26              .select( s: "word, count(1) as count");
27
28          tEnv.toDataSet(result, Row.class).print();
29      }
30  }
```

Environment

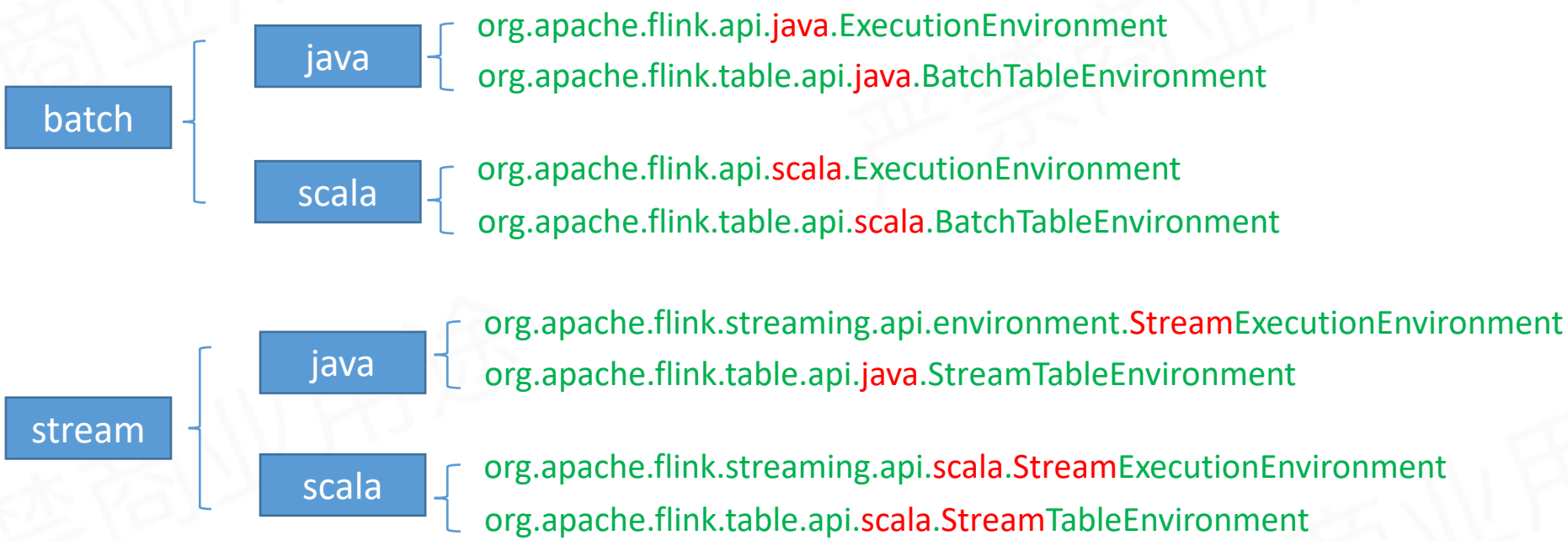


Table Environment使用优化
https://mail.google.com/mail/u/0/?tab=wm#search/label%3Aflink-dev+table+environment/FMfcgxvzMBlhTWVjxlzCnVZsLvvDkmph
https://cwiki.apache.org/confluence/display/FLINK/FLIP-32%3A+Restructure+flink-table+for+future+contributions

How to get a Table ?

How to get a table?

How to register a table?

```
Table myTable = tableEnvironment.scan(“MyTable”);
```

1. Table descriptor

```
tEnv
  .connect(
    new FileSystem()
      .path(path))
  .withFormat(
    new OldCsv()
      .field("word", Types.STRING)
      .lineDelimiter("\n"))
  .withSchema(
    new Schema()
      .field("word", Types.STRING))
  .registerTableSource("sourceTable");
```

2. User defined table source

```
TableSource csvSource = new CsvTableSource(
  path,
  new String[]{"word"},
  new TypeInformation[]{Types.STRING});

tEnv.registerTableSource("sourceTable2", csvSource);
```

3. Register a DataStream

```
DataStream<String> stream = ...

// register the DataStream as table " myTable3" with
// fields "word"
tableEnv
    .registerDataStream("myTable3", stream, "word");
```

How to emit a Table ?

How to emit a table?

resultTable.insertInto(“TargetTable”);

1. Table descriptor

```
tEnv
  .connect(
    new FileSystem()
      .path(path))
  .withFormat(
    new OldCsv()
      .field("word", Types.STRING)
      .lineDelimiter("\n"))
  .withSchema(
    new Schema()
      .field("word", Types.STRING))
  .registerTableSink("targetTable");
```

2. User defined table sink

```
TableSink csvSink = new CsvTableSink(
  path,
  new String[]{"word"},
  new TypeInformation[]{Types.STRING});

tEnv.registerTableSink("sinkTable2", csvSink);
```

3. emit to a DataStream

```
// emit the result table to a DataStream
DataStream<Tuple2<Boolean, Row>> stream =
  tableEnv
    .toRetractStream(resultTable, Row.class);
```

How to query a Table ?

Table API操作

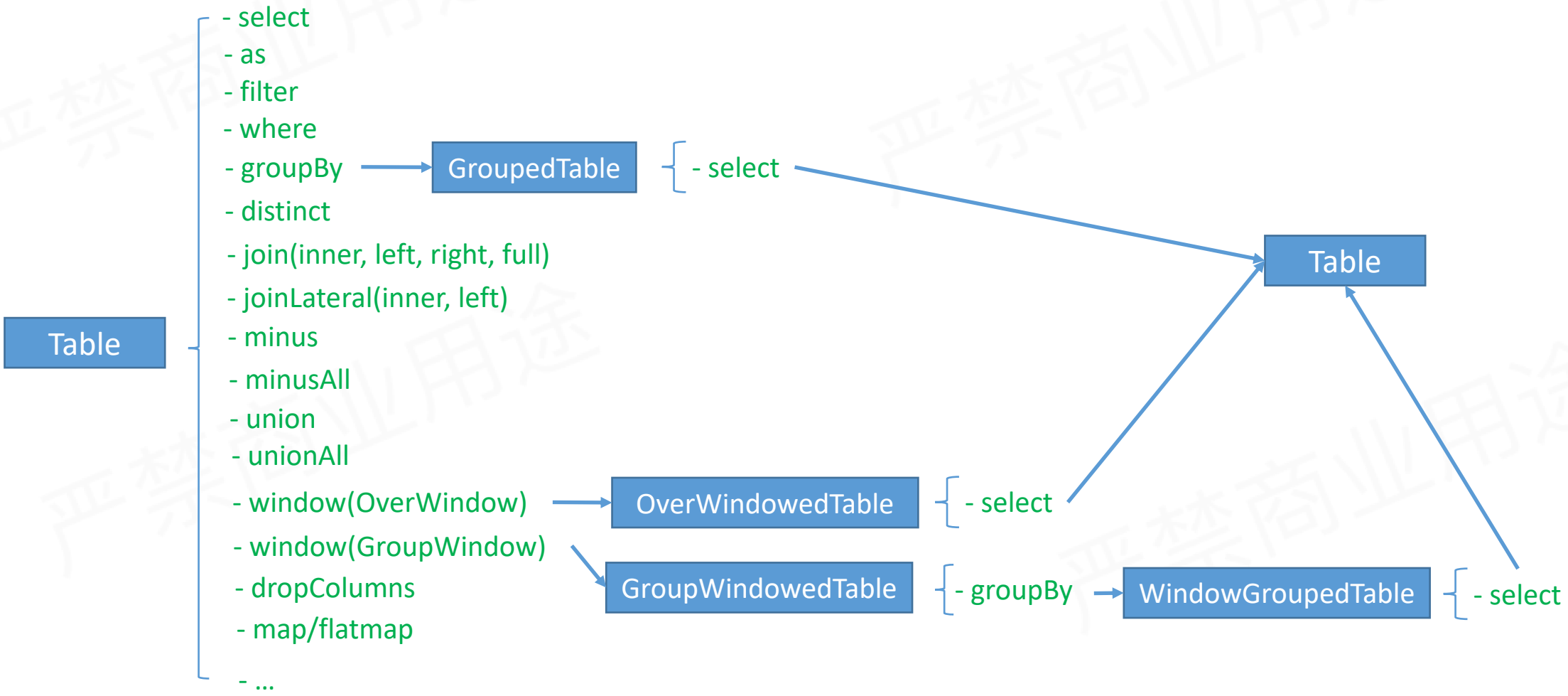
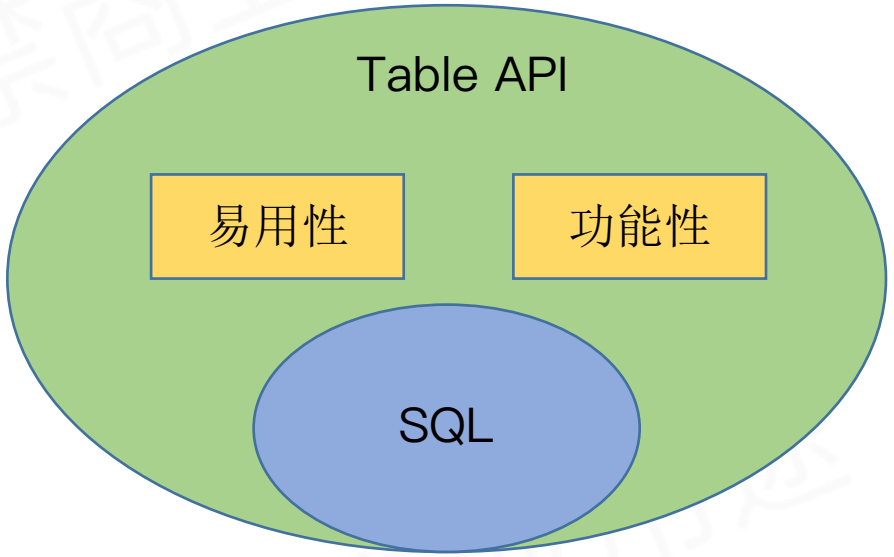


Table API操作

Table API操作

1. 和Sql对齐的操作
2. 提升Table API 易用性的操作
3. 增强Table API 功能性的操作



Columns Operation & Function

Columns Operation – 易用性

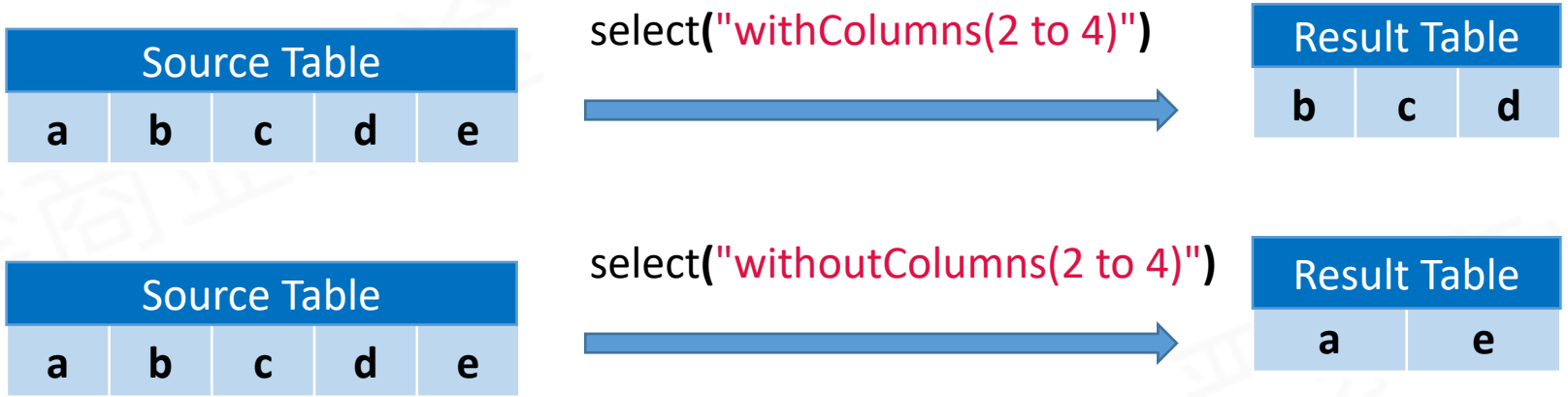
假设有一张100列的表，我们需要去掉一列，需要怎么操作？

Operators	Examples
AddColumns	Table orders = tableEnv.scan("Orders"); Table result = orders.addColumns(“concat(c, ‘sunny‘) as desc");
AddOrReplaceColumns	Table orders = tableEnv.scan("Orders"); Table result = orders.addOrReplaceColumns("concat(c, 'sunny') as desc");
DropColumns	Table orders = tableEnv.scan("Orders"); Table result = orders.dropColumns("b, c");
RenameColumns	Table orders = tableEnv.scan("Orders"); Table result = orders.renameColumns("b as b2, c as c2");

假设有一张100列的表，我们需要选择第20-第80列，需要怎么操作？

Columns Function – 易用性

SYNTAX	DESC
withColumns(...)	select the specified columns
withoutColumns(...)	deselect the columns specified



Columns Function – 易用性

Syntax

The proposed column operation syntax is as follows:

columnOperation:
withColumns(columnExprs) / withoutColumns(columnExprs)

columnExprs:
columnExpr [, columnExpr]*

columnExpr:
columnRef | columnIndex to columnIndex | columnName to columnName

columnRef:
columnName(The field name that exists in the table) | columnIndex(a positive integer starting at 1)

Example: withColumns(a, b, 2 to 10, w to z)

总结

API	Example
Columns Operation	table.dropColumns('a, 'b)
Columns Function	table.select(withColumns('a, 1 to 10))

Row-based operation

Map operation – 易用性

Method signature
方法签名

```
def map(scalarFunction: Expression): Table
```

Usage
用法

```
val res = tab
  .map(fun('e)).as('a, 'b, 'c)
  .select('a, 'c)
```

Benefit
好处

```
table.select(udf1(), udf2(), udf3()….)
VS
table.map(udf())
```

```
class MyMap extends ScalarFunction {
  var param : String = ""

  def eval([user defined inputs]): Row = {
    val result = new Row(3)
    // Business processing based on data and parameters
    // 根据数据和参数进行业务处理
    result
  }

  override def getResultType(signature: Array[Class[_]]):
TypeInformation[_] = {
    Types.ROW(Types.STRING, Types.INT, Types.LONG)
  }
}
```

INPUT(Row)	OUTPUT(Row)
1	1

FlatMap operation – 易用性

Method signature
方法签名

```
def flatMap(tableFunction: Expression): Table
```

Usage
用法

```
val res = tab
  .flatMap(fun('e,'f)).as('name, 'age)
  .select('name, 'age)
```

Benefit
好处

table.joinLateral(udtf) VS table.flatMap(udtf())

```
case class User(name: String, age: Int)

class MyFlatMap extends TableFunction[User] {
  def eval([user defined inputs]): Unit = {
    for(..){
      collect(User(name, age))
    }
  }
}
```

INPUT(Row)	OUTPUT(Row)
1	N(N>=0)

Aggregate operation – 易用性

Method signature
方法签名

```
def aggregate(aggregateFunction: Expression): AggregatedTable
class AggregatedTable(table: Table, groupKeys: Seq[Expression], aggFunction: Expression)
```

Usage
用法

```
val res = tab
   .groupBy(‘a)
   .aggregate(agg(‘e,’f) as (‘a, ‘b, ‘c))
   .select(‘a, ‘c)
```

Benefit
好处

```
table.select(agg1(), agg2(), agg3()….)
 VS
table.aggregate(agg())
```

```
class CountAccumulator extends JTuple1[Long] {
  f0 = 0L //count
}
class CountAgg extends AggregateFunction[JLong, CountAccumulator]
{
  def accumulate(acc: CountAccumulator): Unit = {
    acc.f0 += 1L
  }


  override def getValue(acc: CountAccumulator): JLong = {
    acc.f0
  }
  ... retract()/merge()
}
```

INPUT(Row)	OUTPUT(Row)
N(N>=0)	1

FlatAggregate operation – 功能性

Method signature
方法签名

```
def flatAggregate(tableAggregateFunction: Expression): FlatAggregateTable
class FlatAggregateTable(table: Table, groupKey: Seq[Expression], tableAggFun: Expression)
```

Usage
用法

```
val res = tab
   .groupBy('a)
   .flatAggregate(
      flatAggFunc('e,'f) as ('a, 'b, 'c))
   .select('a, 'c)
```

Benefit
好处

新增了一种agg，能输出多行

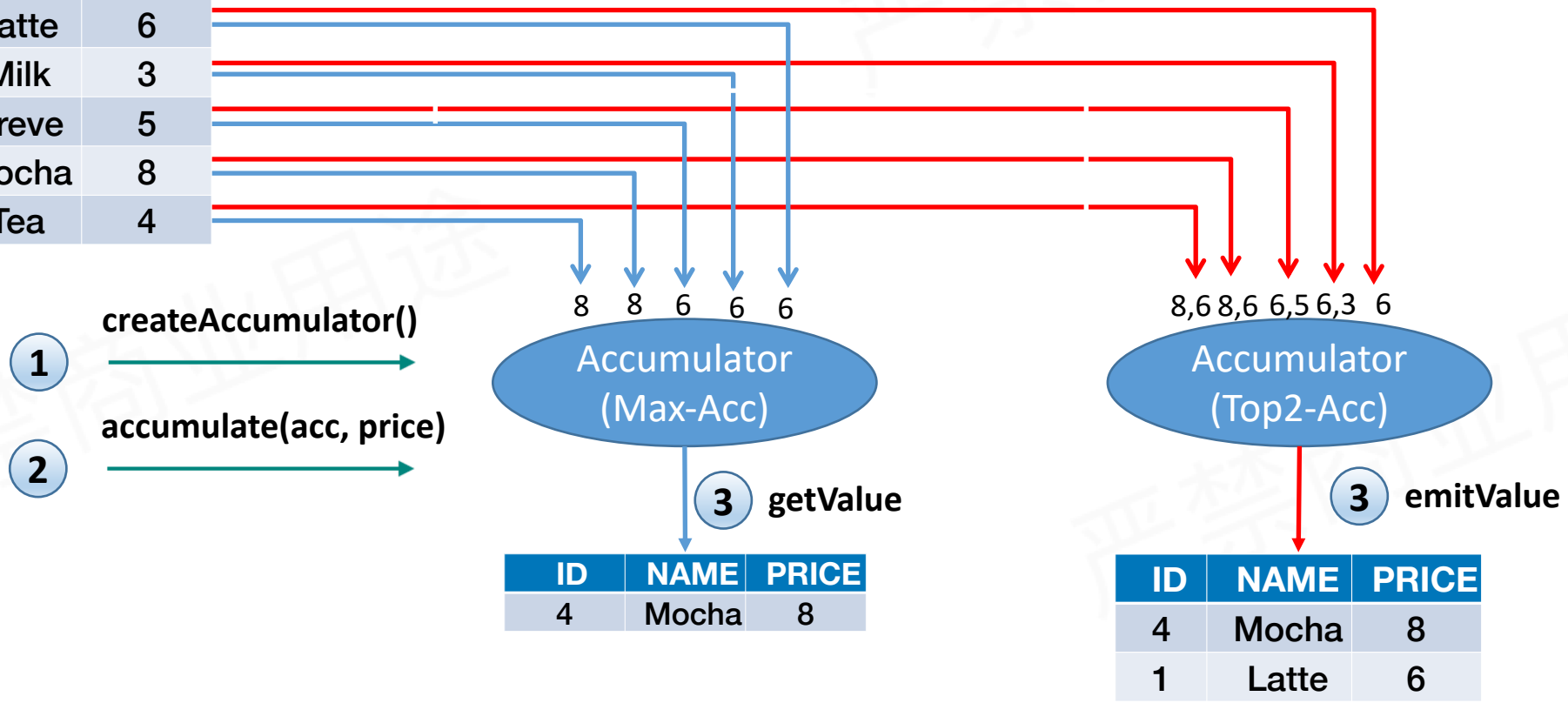
```
class TopNAcc {
  var data: MapView[JInt, JLong] = _  // (rank -> value)
  ...
}
class TopN(n: Int) extends TableAggregateFunction[(Int, Long),
TopNAccum] {

  def accumulate(acc: TopNAcc, [user defined inputs]) {
    ...
  }
  def emitValue(acc: TopNAcc, out: Collector[(Int, Long)]): Unit = {
    ...
  }
  ...retract/merge
}
```

INPUT(Row)	OUTPUT(Row)
N(N>=0)	M(M>=0)

Aggregate VS TableAggregate

使用 Max 和 Top2 的场景比较 Aggregate 和 FlatAggregate 之间的差异

ID	NAME	PRICE
1	Latte	6
2	Milk	3
3	Breve	5
4	Mocha	8
5	Tea	4

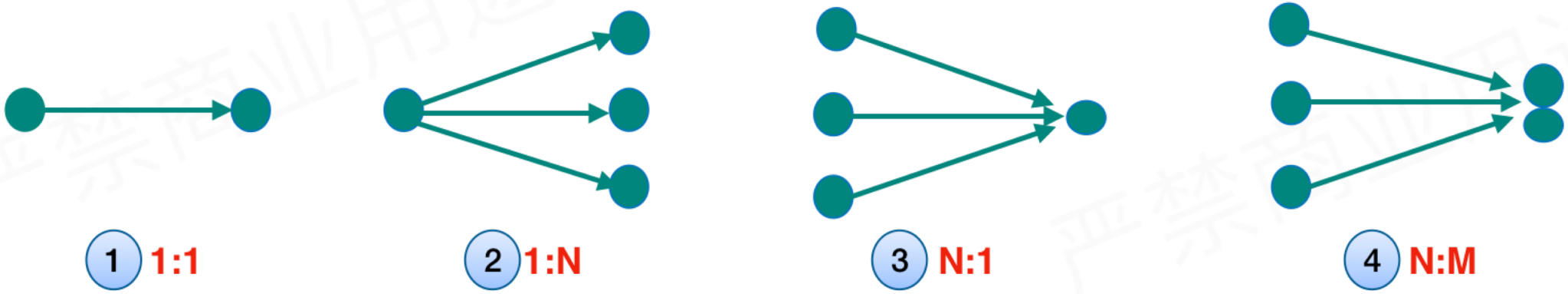


ID	NAME	PRICE
4	Mocha	8

ID	NAME	PRICE
4	Mocha	8
1	Latte	6

总结

	Single Row Input 单行输入	Multiple Row Input 多行输入
Single Row Output 单行输出	ScalarFunction (select/map)	AggregateFunction (select/aggregate)
Multiple Row Output 多行输出	TableFunction (joinLateral/flatmap)	TableAggregateFunction (flatAggregate)



03

Table API动态

3.1 Table API 动态

3.1 Flip29

https://issues.apache.org/jira/browse/FLINK-10972

3.2 Python Table API

https://issues.apache.org/jira/browse/FLINK-12308

3.3 Interactive Programming(交互式编程)

https://issues.apache.org/jira/browse/FLINK-11199

3.4 Iterative processing(迭代计算)

https://issues.apache.org/jira/browse/FLINK-11199

THANKS

Flink China社区大群

扫一扫群二维码，立刻加入该群。