1.

(1)scan : 扫描一个注册过的表。

(2)select : 类似于SQL中的SELECT语句。 执行select 操作。

(3)as ：字段别名

(4)where/filter : 类似于SQL的Where。 过滤掉不符合条件的行。

(5)groupby :分组。

(6)distinct :返回去重的结果记录

(7)join:连接两张表。两张表的字段名不能有相同的,通过 join 算子使用where或filter定义至少有一个等值连接条件谓词

(8)leftOuterJoin

(9)rightOuterJoin

(10)Intersect : 返回两张表的交集。如果某个记录在一张或两张表中出现多次，也只返回一条记录，即结果是去重的。两张表的字段类型必须完全一致。

(11)order by :返回全局排序过的记录

(12)union :合并两张表，两张表的字段类型必须完全一致

### 开发

**import** org.apache.flink.api.java.tuple.Tuple1;  
**import** org.apache.flink.api.java.tuple.Tuple2;  
**import** org.apache.flink.streaming.api.datastream.DataStream;  
**import** org.apache.flink.streaming.api.datastream.DataStreamSource;  
**import** org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;  
**import** org.apache.flink.table.api.StreamTableEnvironment;  
**import** org.apache.flink.table.api.Table;  
  
**public class** h1 {  
 **public static void** main(String[] args)**throws** Exception {  
 StreamExecutionEnvironment env = StreamExecutionEnvironment.*getExecutionEnvironment*();  
 org.apache.flink.table.api.java.StreamTableEnvironment tenv = StreamTableEnvironment.*getTableEnvironment*(env);  
 DataStreamSource<Tuple1<String>> source1 = env.fromElements(Tuple1.*of*(**"aa"**),Tuple1.*of*(**"bb"**),Tuple1.*of*(**"aa"**),Tuple1.*of*(**""**));  
 tenv.registerDataStream(**"t1"**,source1,**"name"**);  
  
 Table f1 = tenv.scan(**"t1"**).filter(**"name.isNotNull"**).groupBy(**"name"**).select(**"name as a1,name.count as a2"**);  
 DataStream<Tuple2<Boolean, tat1>> tat1d = tenv.toRetractStream(f1, tat1.**class**);  
 tat1d.print();  
 env.execute(**"a"**);  
 }  
}

**package** com.sql;  
  
**public class** tat1 {  
 **public** String **a1**;  
 **public** Long **a2**;  
 **public** tat1(){  
  
 }  
 @Override  
 **public** String toString(){  
 **return a1**+**" "**+**a2**;  
 }  
}

### 例子2： join

**package** com.sql;  
  
**import** org.apache.flink.api.java.tuple.Tuple2;  
**import** org.apache.flink.streaming.api.datastream.DataStream;  
**import** org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;  
**import** org.apache.flink.table.api.Table;  
**import** org.apache.flink.table.api.TableEnvironment;  
**import** org.apache.flink.table.api.java.StreamTableEnvironment;  
  
**public class** h2 {  
 **public static void** main(String[] args)**throws** Exception{  
 StreamExecutionEnvironment env = StreamExecutionEnvironment.*getExecutionEnvironment*();  
 StreamTableEnvironment tEnv = TableEnvironment.*getTableEnvironment*(env);  
  
 DataStream<Tuple2<Integer, String>> stream1 = env.fromElements(**new** Tuple2<>(1, **"hello"**)  
 ,**new** Tuple2<>(2, **"nihao"**));  
 DataStream<Tuple2<Integer, String>> stream2 = env.fromElements(**new** Tuple2<>(1, **"beijing"**)  
 ,**new** Tuple2<>(3, **"shanghai"**));  
  
 Table table1 = tEnv.fromDataStream(stream1, **"id1, word1"**);  
 Table table2 = tEnv.fromDataStream(stream2, **"id2, word2"**);  
 Table table = table1.join(table2).where(**"id1=id2"**).select(**"word1,word2"**);  
 Table table11 = table1.leftOuterJoin(table2,**"id1=id2"**).select(**"word1,word2"**);  
  
 DataStream<Tuple2<Boolean, tat2>> d1 = tEnv.toRetractStream(table, tat2.**class**);  
 DataStream<Tuple2<Boolean, tat2>> d2 = tEnv.toRetractStream(table11, tat2.**class**);  
 d1.print();  
 d2.print();  
 env.execute(**"a"**);  
 }  
}

### 例子3：intersect（交集）

**package** com.sql;  
  
**import** org.apache.flink.api.java.DataSet;  
**import** org.apache.flink.api.java.ExecutionEnvironment;  
**import** org.apache.flink.api.java.operators.DataSource;  
**import** org.apache.flink.api.java.tuple.Tuple2;  
**import** org.apache.flink.table.api.Table;  
**import** org.apache.flink.table.api.TableEnvironment;  
**import** org.apache.flink.table.api.java.BatchTableEnvironment;  
  
**public class** h3 {  
 **public static void** main(String[] args)**throws** Exception{  
 ExecutionEnvironment env = ExecutionEnvironment.*getExecutionEnvironment*();  
 BatchTableEnvironment tenv = TableEnvironment.*getTableEnvironment*(env);  
  
 DataSource<Tuple2<Integer, String>> table1 = env.fromElements(**new** Tuple2<>(1, **"zs"**),  
 **new** Tuple2<>(2, **"ls"**),  
 **new** Tuple2<>(3, **"ww"**)  
 );  
 DataSource<Tuple2<Integer, String>> table2 = env.fromElements(**new** Tuple2<>(1, **"zs"**),  
 **new** Tuple2<>(12, **"haha"**),  
 **new** Tuple2<>(13, **"hehe"**)  
 );  
  
 Table t11 = tenv.fromDataSet(table1, **"id1,name1"**);  
 Table t22 = tenv.fromDataSet(table2, **"id2,name2"**);  
  
 Table t3 = t11.intersect(t22);  
 DataSet<tat3> t23 = tenv.toDataSet(t3, tat3.**class**);  
 t23.print();  
 }  
}

**package** com.sql;  
  
**public class** tat3 {  
 **public** Integer **id1**;  
 **public** String **name1**;  
  
 **public** tat3(){  
  
 }  
 @Override  
 **public** String toString(){  
 **return id1**+**" "**+**name1**;  
 }  
}