Flink原理实战每日一篇11 ---SQL实例学习

从这里开始讲Flink SQL 听说Flink1.9发布之后 Blink的SQL 会并入到Flink，那时候SQL会更强大，这个时候不到打好基础怎么行呢。。。。。。。。

一，Flink SQL使用

最简单的案例使用：

```
import org.apache.flink.streaming.api.scala.StreamExecutionEnvironment
import org.apache.flink.table.api.{Table, TableEnvironment}
import org.apache.flink.streaming.api.scala._
import org.apache.flink.table.api.scala._

object SqlAPI {
  def main(args: Array[String]): Unit = {

    val env = StreamExecutionEnvironment.getExecutionEnvironment;
    // 创建table对象
    val tableEnv = TableEnvironment.getTableEnvironment(env)

    //Stream 或者 dataSet 与Table的转换

 val dataStream: DataStream[(Int, Int)] = env.fromElements((1, 2), (12, 23)
    tableEnv.registerDataStream("table1", dataStream, 'myLong, 'myString)

 val table: Table = tableEnv.sqlQuery("select myLong,myString from table1")
            //打印输出
 val  rs: DataStream[(Integer, Integer)] = tableEnv.toAppendStream[(Integer
         rs.print()
    env.execute()
  }
}
```

1，执行SQL语句，

 在SQL中引用Table，实际代码开发这样子写会比较好~我个人觉得

要点就是讲sql语句变成 s"........................$table"

```

val table_demo1: Table = tableEnv.fromDataStream(dataStream, 'myLong, 'myStrin
val table: Table = tableEnv.sqlQuery(s"select myLong,myString from $table_demo1
```

基本操作：

```
val table33: Table = tableEnv.sqlQuery(s"select myLong,myString from $table_dem
o1 where myLong > 10")
```

2，Group Windows窗口操作

1）Tumble windows 滚动窗口

窗口是固定的，且窗口和窗口之间的数据不会重合

```
//基于proctime创建滚动窗口，并制定10秒切为一个窗口，

tableEnv.sqlQuery("select id,sum(type) from table1 group by tumble(proctime, in
    基于rowtime创建滚动窗口，并制定5秒切为一个窗口

tableEnv.sqlQuery("select id,sum(type) from table1 group by tumble(rowtime, int
```

上面写的代码不规范，关键字要大写，后续会贴出标准的可以执行的代码

2）HOP Windows 滑动窗口

窗口是固定的，且窗口和窗口之间的数据可以重合 通过tumble(time_attr, interval_01,interval_02)

```
tableEnv.sqlQuery("select id,sum(type) from table1 group by HOP(proctime, int

tableEnv.sqlQuery("select id,sum(type) from table1 group by HOP(rowtime, inte

tableEnv.sqlQuery("select id,sum(type) ,HOP_START(rowtime, interval '5' SECON

注意：  还可以通过 HOP_START ，HOP_END 指定窗口起始，结束 时间。
```

3) Session Windows

```
tableEnv.sqlQuery("select id,sum(type) from table1 group by SESSION(rowtime,
tableEnv.sqlQuery("select id,sum(type) ,SESSION_START(rowtime, interval '5' S
```

注意： 还可以通过 HOP_START，HOP_END 指定窗口起始，结束 时间。

3，数据聚合

1）GroupBy

`tableEnv.sqlQuery("select id,sum(type) from table1 group by id")`

```
/**
  * Flink sql实例API
  */
object Flink_SQL {
  def main(args: Array[String]): Unit = {
    val env = StreamExecutionEnvironment.getExecutionEnvironment


    //todo 创建tableEnv
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    val stream = env.fromElements(("aa", 192L), ("aa", 192L))
    val table: Table = tableEnv.fromDataStream(stream, 'id, 'name)

    tableEnv.registerDataStream("table2", stream, 'myLong, 'myString)
    tableEnv.registerDataStream("table3", stream, 'myLong, 'myString)

    //查询
    tableEnv.sqlQuery("select * from table2")

    //修改

    tableEnv.sqlUpdate("INSERT INTO table3 SELECT * FROM table2")


    //todo 执行SQL ,使用关键字
    val  table2 = stream.toTable(tableEnv,'myLong, 'myString)

    val rs = tableEnv.sqlQuery(s"select * from $table2")

    //todo SQL结果输出
    val  csvTableSink: CsvTableSink = new CsvTableSink("/path/csvFile")
//    tableEnv.registerTable("csvTable",Array("id","name"),Array(Types.STRIN



    //todo 数据查询与过滤
    val rs2 = tableEnv.sqlQuery(s"select * from $table2 where name%2=0")
    //todo Group Window 操作


    tableEnv.sqlQuery(s"SELECT myLong ,SUM(myString) FROM $table2 GROUP BY TU
      s"INTERVAL '5' MINUTE),id")

  }
}
```

2) GroupBy Window

`tableEnv.sqlQuery("select id,sum(type) from table1 group by TUMBLE(rowtime, interval '5' SECOND),id")`

```
object GroupBy_window {
  def main(args: Array[String]): Unit = {
    val env = StreamExecutionEnvironment.getExecutionEnvironment


    //todo 创建tableEnv
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    val stream = env.fromElements(("aa", 192L), ("aa", 192L), ("bb", 112L))

    val table: Table = tableEnv.fromDataStream(stream, 'key, 'time)

    //todo  1，滚动窗口，创建窗口，设置窗口大小 ,基于event time  关键字 on rowtime
    val table2 = table.window(Tumble over 1.minute on 'rowtime as 'window)

    //todo 2,基于process time 关键字  on proctime
    val table3 = table.window(Tumble over 1.minute on 'proctime as 'window)


 //todo 3,基于元素数量 这里的 proctime字段没实际意义  window是重命名，后续使用会用这个
    val table4 = table.window(Tumble over 100.rows on 'proctime as 'window)

    //todo  sliding window

 val sliding_table1 = table.window(Slide over 2.minute every 1.minute on 'ro

 val sliding_table2 = table.window(Slide over 2.minute every 1.minute on 'pr

 val sliding_table3 = table.window(Slide over 100.rows every 1.minute on 'ro

    //todo Session window
  val session_table1 = table.window(Session withGap 2.minute on 'rowtime as

  val session_table2 = table.window(Session withGap 2.minute on 'proctime as



    //todo 普通聚合操作
    table.groupBy('key).select('key, 'time.sum as 'sumValue)

    //todo GroupBy Window 聚合操作（全量聚合）

    val rs = table.window(Tumble over 1.minute on 'proctime as 'window)
      //根据key 很window进行聚合
      .groupBy('key, 'window)
      .select('key, 'window.start, 'window.end, 'window.rowtime, 'key)


    //todo 去重

    val rs2 = table.distinct()
  }
}
```

3) Over Window

tableEnv.sqlQuery("select MAX(var1) OVER (PARTITION BY id ORDER BY proctime ROWS BETWEN 10 PRECEDING AND CURRENT ROW) FROM Sensors)

ROWS BETWEN 10 PRECEDING AND CURRENT ROW ---限定从当前数据向前推10条记录

```
/**
  * over window 是基于当前数据和其周围临近范围内的数据进行聚合统计的，比如
  * 基于当前记录前面的20条数据
  */
object Over_window {
  def main(args: Array[String]): Unit = {

    val env = StreamExecutionEnvironment.getExecutionEnvironment


    //todo 创建tableEnv
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    val stream = env.fromElements(("aa", 192L), ("aa", 192L),("bb", 112L))
    val table: Table = tableEnv.fromDataStream(stream, 'id, 'time)

    //todo 主要四个参数   partitionBy ,  orderBy ,  preceding (following,与prec
    table.window(Over partitionBy 'id orderBy 'rowtime preceding UNBOUNDED_R
      .select('id,'time.sum over 'window,'time.max over 'window)


    //todo Over Window 聚合操作
    val rs: Table =table.window(Over partitionBy 'id orderBy 'rowtime preced
      .select('id,'id.avg over 'window,'time.max over 'window)

    //todo 去重

    val distinctRs: Table = table.window(Over partitionBy 'id orderBy 'rowti
      .select('id,'id.avg over 'window,'time.max over 'window)

    table.distinct()
  }
}
```

4）Distinct 去重

`tableEnv.sqlQuery("select DISTINCT type FROM Sensors)`

5）Grouping sets

统计2个字段的总数。

`tableEnv.sqlQuery("select SUM(id,name) FROM Sensors GTOUP BY  GROUPING SETS((id),(type)))`

6）Hiving

`tableEnv.sqlQuery("select SUM(type) FROM Sensors GTOUP BY id HIVing SUM(type) > 500 )`

7）自定义UDF函数 后续再讲

4，多表关联

```
package com.coder.flink.core.table_sql

import org.apache.flink.streaming.api.scala.{StreamExecutionEnvironment, _}
import org.apache.flink.table.api.scala._
import org.apache.flink.table.api.{Table, TableEnvironment}
```

```
object SQL_join {    def main(args: Array[String]): Unit = {
    val env = StreamExecutionEnvironment.getExecutionEnvironment


    //todo 创建tableEnv
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    val stream = env.fromElements(("aa", 192L), ("aa", 192L))
    val table: Table = tableEnv.fromDataStream(stream, 'id, 'name)

    tableEnv.registerDataStream("table2", stream, 'myLong, 'myString)
    tableEnv.registerDataStream("table3", stream, 'myLong, 'myString)

    // 左外连接
    tableEnv.sqlQuery("select * from table2 LEFT JOIN table3 ON table2.myLong

    //右外连接
    tableEnv.sqlQuery("select * from table2 RIGHT JOIN table3 ON table2.myLon

    // 全外连接
    tableEnv.sqlQuery("select * from table2 FULL OUTER JOIN table3 ON table2.


    // time-window Join  需要指定至少一个关联条件以及坝顶两张表中的关联时间字段，且两个
    tableEnv.sqlQuery("select * from table2 a ,table3 b where  a.myLong = b.r


    // Join with Table Function  sql里面跟Table join
    tableEnv.sqlQuery("select * from table2 , LATERAL TABLE(my_udtf(type)) t




  }
}
```

5，集合操作：

1）UNION 操作

```
tableEnv.sqlQuery("select * from (select * from table2 where myLong > 10) UNION
 (select * from table3 where myLong =  10)" )
```

2）UNION ALL

tableEnv.sqlQuery("select * from (select * from table2 where myLong > 10) UNION ALL (select * from table3 where myLong = 10)")

3）INTERSECT 取交集

tableEnv.sqlQuery("select * from (select * from table2 where myLong > 10) INTERSECT ALL (select * from table3 where myLong = 10)")

4）IN操作

```
tableEnv.sqlQuery("select myLong from  table2 where myLong IN (select myLong from  table2 where myLong > 100)" )
```

5）EXISTS 判断是否存在

```
tableEnv.sqlQuery("select myLong from  table2 where myLong EXISTS(select myLong from  table3)")
```

6）数据输出 用SqlUpdate

```
tableEnv.sqlUpdate("INSERT INTO OutoutTable select myLong from  table2 ")
```

最后总结：如果有不懂的代码 参考 https://github.com/opensourceteams/flink-maven-scala/blob/master/md/sql/flink-sql-dataset-example.md

SELECT

Scan / Select

- 功能描述: 查询一个表中的所有数据
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operation
        import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._
  object Run {



    def main(args: Array[String]): Unit = {


      //得到批环境
      val env = ExecutionEnvironment.getExecutionEnvironment



     val dataSet = env.fromElements(("小明",15,"男"),("小王",45,"男"),("小李",25,"
            //得到Table环境
      val tableEnv = TableEnvironment.getTableEnvironment(env)
      //注册table
      tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex)



      tableEnv.sqlQuery(s"select name,age FROM user1")
        .first(100).print()


      /**
        * 输出结果
        *
        * 小明,15
        * 小王,45
        * 小李,25
        * 小慧,35
        */
    }

  }
```

- 输出结果

```
小明,15
小王,45
小李,25
```

```
小慧,35
```

as (table)

- 功能描述: 给表名取别称
- scala 程序

```scala
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operation
    import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment



  val dataSet = env.fromElements(("小明",15,"男"),("小王",45,"男"),("小李",25,"
      //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex)



    tableEnv.sqlQuery(s"select t1.name,t1.age FROM user1 as t1")
      .first(100).print()


    /**
      * 输出结果
      *
      * 小明,15
      * 小王,45
      * 小李,25
      * 小慧,35
```

```
        */    }

}
```

- 输出结果

```
小明,15
小王,45
小李,25
小慧,35
```

as (column)

- 功能描述: 给表名取别称
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operation
        import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment



   val dataSet = env.fromElements(("小明",15,"男"),("小王",45,"男"),("小李",25,"
         //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex)
    tableEnv.sqlQuery(s"select name a,age as b FROM user1 ")
      .first(100).print()


    /**
      * 输出结果
      *
      * 小明,15
      * 小王,45
      * 小李,25
      * 小慧,35
      */
  }

}
```

- 输出结果

```
小明,15
小王,45
小李,25
小慧,35
```

limit

功能描述:查询一个表的数据，只返回指定的前几行(争对并行度而言,所以并行度不一样，结果不一样)

scala 程序

```
package com.opensourceteams.mo`dule.bigdata.flink.example.sql.dataset.operatio
    import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {
  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment
    env.setParallelism(2)



 val dataSet = env.fromElements(("小明",15,"男"),("小王",45,"男"),("小李",25,"
         //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex)



    /**

  * 先排序，按age的降序排序，输出前100位结果，注意是按同一个并行度中的数据进行排序，也就
      */
 tableEnv.sqlQuery(s"select name,age FROM user1  ORDER BY age desc LIMIT 10
      .first(100).print()

    /**
      * 输出结果 并行度设置为2
      *
      * 小明,15
      * 小王,45
      * 小慧,35
      * 小李,25
      */

    /**
      * 输出结果 并行度设置为1
      *
      * 小王,45
      * 小慧,35
      * 小李,25
      * 小明,15
      */




  }

}
```

- 输出结果

```
小明,15
小王,45
小慧,35
小李,25
```

Where / Filter

- 功能描述:列加条件过滤表中的数据
- scala 程序

```

package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operation
    import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment



 val dataSet = env.fromElements(("小明",15,"男"),("小王",45,"男"),("小李",25,"
        //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex)



    tableEnv.sqlQuery(s"select name,age,sex FROM user1 where sex = '女'")
      .first(100).print()


    /**
      * 输出结果
      *
```

```
        * 小李,25,女
        * 小慧,35,女
        */

  }

}
```

- 输出结果

```
小李,25,女
小慧,35,女
```

between and (where)

- 功能描述: 过滤列中的数据, 开始数据 <= data <= 结束数据
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operation
import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment



    val dataSet = env.fromElements(("小明",15,"男"),("小王",45,"男"),("小李",25,"
    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex)
```

```
     tableEnv.sqlQuery(s"select name,age,sex FROM user1 where age between 20 and
       .first(100).print()

    /**
      * 结果
      *
      * 小李,25,女
      * 小慧,35,女
      */

  }

}
```

- 输出结果

```
小李,25,女
小慧,35,女
```

Sum

- 功能描述: 求和所有数据
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operatio

import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {

    //得到批环境      val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements(("小明",15,"男",1500),("小王",45,"男",4000),

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex,'salary)



    //汇总所有数据
    tableEnv.sqlQuery(s"select sum(salary) FROM user1")
      .first(100).print()


    /**
      * 输出结果
      *
      * 6800
      */


  }

}
```

- 输出结果

```
6800
```

max

- 功能描述: 求最大值
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operati

import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements(("小明",15,"男",1500),("小王",45,"男",4000),

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex,'salary)



    //汇总所有数据
    tableEnv.sqlQuery(s"select max(salary) FROM user1 ")
      .first(100).print()


    /**
      * 输出结果
      *
      * 4000
      */


  }

}
```

- 输出结果

```
4000
```

min

- 功能描述: 求最小值
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operatio

import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements(("小明",15,"男",1500),("小王",45,"男",4000),

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex,'salary)



    tableEnv.sqlQuery(s"select min(salary) FROM user1 ")
      .first(100).print()


    /**
      * 输出结果
      *
      * 500
      */


  }

}
```

- 输出结果

```
500
```

sum (group by)

- 功能描述: 按性别分组求和
- scala 程序

```

package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operatio

import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements(("小明",15,"男",1500),("小王",45,"男",4000),

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex,'salary)



    //汇总所有数据
    tableEnv.sqlQuery(s"select sex,sum(salary) FROM user1 group by sex")
      .first(100).print()

    /**
      * 输出结果
      *
      * 女,1300
      * 男,5500         */

  }

}
```

- 输出结果

```
女,1300
男,5500
```

group by having

- 功能描述:
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operati

import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements(("小明",15,"男",1500),("小王",45,"男",4000),

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex,'salary)


    //分组统计，having是分组条件查询
    tableEnv.sqlQuery(s"select sex,sum(salary) FROM user1 group by sex having
      .first(100).print()

    /**
      * 输出结果
      *
      *
      */



  }

}


```

- 输出结果

```
男,5500

```

distinct

- 功能描述: 去重一列或多列
- scala 程序

```

package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operation
     import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    val env = ExecutionEnvironment.getExecutionEnvironment

  val dataSet = env.fromElements(("a",15,"male"),("a",45,"female"),("d",25,"r
        val tableEnv = TableEnvironment.getTableEnvironment(env)

    tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex)


    /**
      * 对数据去重
      */
    tableEnv.sqlQuery("select distinct name  FROM user1   ")
      .first(100).print()


    /**
      * 输出结果
      *
      * a
      * c
      * d
      */

  }

}
```

- 输出结果

```
a
c
d
```

join

INNER JOIN

- 功能描述: 连接两个表，按指定的列，两列都存在值才输出
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operatic
```

```scala
import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",4
    val dataSetGrade = env.fromElements((1,"语文",100),(2,"数学",80),(1,"外语",

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user",dataSet,'id,'name,'age,'sex,'salary)
    tableEnv.registerDataSet("grade",dataSetGrade,'userId,'name,'fraction)



    //内连接，两个表
   // tableEnv.sqlQuery("select * FROM `user`  INNER JOIN  grade on  `user`.
    tableEnv.sqlQuery("select `user`.*,grade.name,grade.fraction FROM `user`
      .first(100).print()


    /**
      * 输出结果
      * 2,小王,45,男,4000,数学,80
      * 1,小明,15,男,1500,语文,100
      * 1,小明,15,男,1500,外语,50
      */

  }

}
```

- 输出结果

```
2,小王,45,男,4000,数学,80
                           1,小明,15,男,1500,语文,100
1,小明,15,男,1500,外语,50
```

left join

- 功能描述:连接两个表，按指定的列，左表中存在值就一定输出，右表如果不存在，就显示为空
- scala 程序

```

package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operati

import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",4
    val dataSetGrade = env.fromElements((1,"语文",100),(2,"数学",80),(1,"外语",

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user",dataSet,'id,'name,'age,'sex,'salary)
    tableEnv.registerDataSet("grade",dataSetGrade,'userId,'name,'fraction)




  //左连接，拿左边的表中的每一行数据，去关联右边的数据，如果有相同的匹配数据，就都匹配出来，
    tableEnv.sqlQuery("select `user`.*,grade.name,grade.fraction FROM `user`
      .first(100).print()


    /**
```

```
         * 输出结果                  *
         * 1,小明,15,男,1500,语文,100
         * 1,小明,15,男,1500,外语,50
         * 2,小王,45,男,4000,数学,80
         * 4,小慧,35,女,500,null,null
         * 3,小李,25,女,800,null,null
         *
         *
         */

  }

}
```

- 输出结果

```
1,小明,15,男,1500,语文,100
1,小明,15,男,1500,外语,50
2,小王,45,男,4000,数学,80
4,小慧,35,女,500,null,null
3,小李,25,女,800,null,null
```

right join

- 功能描述:连接两个表，按指定的列，右表中存在值就一定输出，左表如果不存在，就显示为空
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operatio

import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {
        //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",4
    val dataSetGrade = env.fromElements((1,"语文",100),(2,"数学",80),(1,"外语",

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user",dataSet,'id,'name,'age,'sex,'salary)
    tableEnv.registerDataSet("grade",dataSetGrade,'userId,'name,'fraction)



  //左连接，拿左边的表中的每一行数据，去关联右边的数据，如果有相同的匹配数据，就都匹配出来，
    tableEnv.sqlQuery("select `user`.*,grade.name,grade.fraction FROM `user`
      .first(100).print()


    /**
      * 输出结果
      *
      * 1,小明,15,男,1500,外语,50
      * 1,小明,15,男,1500,语文,100
      * 2,小王,45,男,4000,数学,80
      * null,null,null,null,null,外语,90
      *


      *
      */

  }

}
```

- 输出结果

```
1,小明,15,男,1500,外语,50
1,小明,15,男,1500,语文,100
2,小王,45,男,4000,数学,80
null,null,null,null,null,外语,90
```

full outer join

- 功能描述: 连接两个表，按指定的列，只要有一表中存在值就一定输出，另一表如果不存在就显示为空
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operatio

import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",40
    val dataSetGrade = env.fromElements((1,"语文",100),(2,"数学",80),(1,"外语",

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user",dataSet,'id,'name,'age,'sex,'salary)
    tableEnv.registerDataSet("grade",dataSetGrade,'userId,'name,'fraction)



  //左，右，全匹配所有数据
    tableEnv.sqlQuery("select `user`.*,grade.name,grade.fraction FROM `user`
      .first(100).print()



    /**
      * 输出结果
      *
      *
      * 3,小李,25,女,800,null,null
      * 1,小明,15,男,1500,外语,50
      * 1,小明,15,男,1500,语文,100
```

```
        * 2,小王,45,男,4000,数学,80
                                      * 4,小慧,35,女,500,null,null
        * null,null,null,null,null,外语,90
        *
        *
        *
        */

  }

}
```

- 输出结果

```
3,小李,25,女,800,null,null
1,小明,15,男,1500,外语,50
1,小明,15,男,1500,语文,100
2,小王,45,男,4000,数学,80
4,小慧,35,女,500,null,null
null,null,null,null,null,外语,90
```

Set Operations

union

- 功能描述: 连接两个表中的数据，会去重
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operatic

import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {

    //得到批环境
        val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",40
    val dataSet2 = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",4

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user",dataSet,'id,'name,'age,'sex,'salary)
    tableEnv.registerDataSet("t2",dataSet2,'id,'name,'age,'sex,'salary)


    /**
      *  union 连接两个表,会去重
      */
    tableEnv.sqlQuery(
      "select * from ("
                +"select t1.* FROM `user` as t1 ) " +
                + " UNION "
                + " ( select t2.* FROM t2 )"



       )
      .first(100).print()


    /**
      * 输出结果
      *
      * 30,小李,25,女,800
      * 40,小慧,35,女,500
      * 2,小王,45,男,4000
      * 4,小慧,35,女,500
      * 3,小李,25,女,800
      * 1,小明,15,男,1500
      *
      */

  }

}
```

- 输出结果

```
30,小李,25,女,800
40,小慧,35,女,500
2,小王,45,男,4000
4,小慧,35,女,500
3,小李,25,女,800
1,小明,15,男,1500
```

unionAll

- 功能描述: 连接两表中的数据，不会去重
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operati

import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",4
    val dataSet2 = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user",dataSet,'id,'name,'age,'sex,'salary)
    tableEnv.registerDataSet("t2",dataSet2,'id,'name,'age,'sex,'salary)


    /**
      *  union 连接两个表,不会去重
      */
    tableEnv.sqlQuery(
      "select * from ("
                    +"select t1.* FROM `user` as t1 ) " +                    + "
                    + " ( select t2.* FROM t2 )"



      )
     .first(100).print()


   /**
     * 输出结果
     *
     * 1,小明,15,男,1500
     * 2,小王,45,男,4000
     * 3,小李,25,女,800
     * 4,小慧,35,女,500
     * 1,小明,15,男,1500
     * 2,小王,45,男,4000
     * 30,小李,25,女,800
     * 40,小慧,35,女,500
     *
     */

  }

}
```

- 输出结果

```
1,小明,15,男,1500
2,小王,45,男,4000
3,小李,25,女,800
4,小慧,35,女,500
1,小明,15,男,1500
2,小王,45,男,4000
30,小李,25,女,800
40,小慧,35,女,500
```

INTERSECT

- 功能描述: INTERSECT 连接两个表,找相同的数据(相交的数据，重叠的数据)
- scala 程序

```scala
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operatio

import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",40
    val dataSet2 = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",4

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user",dataSet,'id,'name,'age,'sex,'salary)
    tableEnv.registerDataSet("t2",dataSet2,'id,'name,'age,'sex,'salary)


    /**
      *  INTERSECT 连接两个表,找相同的数据(相交的数据, 重叠的数据)
      */
    tableEnv.sqlQuery(
      "select * from ("
                +"select t1.* FROM `user` as t1 ) " +
                + " INTERSECT "
                + " ( select t2.* FROM t2 )"




       )
      .first(100).print()


    /**
      * 输出结果
      *
      * 1,小明,15,男,1500
      * 2,小王,45,男,4000
    *    |     */

  }

}
```

- 输出结果

```
1,小明,15,男,1500
2,小王,45,男,4000
```

in

- 功能描述: 子查询
- scala 程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operatio

import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",40
    val dataSet2 = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",4

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user",dataSet,'id,'name,'age,'sex,'salary)
    tableEnv.registerDataSet("t2",dataSet2,'id,'name,'age,'sex,'salary)


    /**
      *  in ,子查询
      */
    tableEnv.sqlQuery(

                "select t1.* FROM `user` t1  where t1.id in " +
                        " (select t2.id from t2) "





       )
      .first(100).print()


    /**
      * 输出结果
      *
      * 1,小明,15,男,1500
      * 2,小王,45,男,4000
      *
      */

  }

}

```

- 输出结果

```
1,小明,15,男,1500
2,小王,45,男,4000

```

EXCEPT

- 功能描述: EXCEPT 连接两个表,找不相同的数据(不相交的数据，不重叠的数据)
- scala 程序

```

package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operatio
```

```scala
    import org.apache.flink.api.scala.{ExecutionEnvironment, _}
import org.apache.flink.table.api.TableEnvironment
import org.apache.flink.table.api.scala._

object Run {



  def main(args: Array[String]): Unit = {


    //得到批环境
    val env = ExecutionEnvironment.getExecutionEnvironment


    val dataSet = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",40
    val dataSet2 = env.fromElements((1,"小明",15,"男",1500),(2,"小王",45,"男",4

    //得到Table环境
    val tableEnv = TableEnvironment.getTableEnvironment(env)
    //注册table
    tableEnv.registerDataSet("user",dataSet,'id,'name,'age,'sex,'salary)
    tableEnv.registerDataSet("t2",dataSet2,'id,'name,'age,'sex,'salary)


    /**
      *  EXCEPT 连接两个表,找不相同的数据(不相交的数据, 不重叠的数据)
      */
    tableEnv.sqlQuery(
      "select * from ("
              +"select t1.* FROM `user` as t1 ) " +
              + " EXCEPT "
              + " ( select t2.* FROM t2 )"




      )
      .first(100).print()



    /**
      * 输出结果
      *
      * 3,小李,25,女,800
      * 4,小慧,35,女,500
      *
      */
    }
  }
```

- 输出结果

```
3,小李,25,女,800
4,小慧,35,女,500
```

DML

insert into

- 功能描述:将一个表中的数据(source)，插入到 csv文件中(sink)
- scala程序

```
package com.opensourceteams.module.bigdata.flink.example.sql.dataset.operation
       import org.apache.flink.api.scala.typeutils.Types
  import org.apache.flink.api.scala.{ExecutionEnvironment, _}
  import org.apache.flink.core.fs.FileSystem.WriteMode
  import org.apache.flink.table.api.TableEnvironment
  import org.apache.flink.table.api.scala._
  import org.apache.flink.api.scala._
  import org.apache.flink.table.sinks.CsvTableSink
  import org.apache.flink.api.common.typeinfo.TypeInformation

  object Run {



    def main(args: Array[String]): Unit = {


      //得到批环境
      val env = ExecutionEnvironment.getExecutionEnvironment



   val dataSet = env.fromElements(("小明",15,"男"),("小王",45,"男"),("小李",25,"
    //得到Table环境

  val tableEnv = TableEnvironment.getTableEnvironment(env)        //注册table
    tableEnv.registerDataSet("user1",dataSet,'name,'age,'sex)




    // create a TableSink

  val csvSink = new CsvTableSink("sink-data/csv/a.csv",",",1,WriteMode.OVERWI
    val fieldNames = Array("name", "age", "sex")
  val fieldTypes: Array[TypeInformation[_]] = Array(Types.STRING, Types.INT,
    tableEnv.registerTableSink("t2",fieldNames,fieldTypes,csvSink)


  tableEnv.sqlUpdate(s" insert into  t2 select name,age,sex FROM user1  ")

    env.execute()


    /**
      * 输出结果
      * a.csv
      *
      * 小明,15,男
      * 小王,45,男
      * 小李,25,女
      * 小慧,35,女
      */






   }

 }
```

- 输出数据 a.csv

```
小明,15,男
小王,45,男
小李,25,女
小慧,35,女
```