Supporting Data Set Joins in BIRT

Design Specification
Draft 1: Feb 13, 2006

Abstract

This is the design specification of the BIRT Data Set Join feature.

Document Revisions

Version	Date	Description of Changes
Draft 1	Feb 13, 2005	First draft
Draft 2	Feb 22, 2005	Second draft
Draft 3	Mar 23, 2005	Third draft

Contents

1. Introduction

It is a commonly requested feature that a BIRT table/list be able to consume data from two or more data sets joined together with a set of join conditions. This feature is very similar to SQL table joins. In fact SQL terminologies are used throughout this specification to define the data set join feature, and the reader of this document is assumed to be familiar with SQL. In comparing BIRT's data set join to SQL joins, a BIRT data set is equivalent to a SQL table, while a data set column is equivalent to a SQL column.

Data set joins in BIRT 2.1 is achieved via the introduction of a new type of data set called *Joint Data Set*. A Joint Data Set is a derived data set whose data comes from joining two data sets.

2. Features and Requirements

2.1 Supported Features

2.1.1 Heterogeneous and Homogeneous Data Set Joins

Data sets involved in a join can be of any data source type. For example, a JDBC data set may be joined to another JDBC data set, or a Scripted Data Set may be joined to an XML data set. A joint data set may also be joined to other data sets.

A data set may be joined to itself (self-join).

2.1.2 Inner and Outer Joins

Inner join and outer join are supported as join types.

2.1.3 Using Joint Data Set

A Joint Data Set may be used wherever a data set may be used, including list, table, data elements etc.

A Joint Data Set may have its own computed columns and filters. A Joint Data Set cannot have its own parameters (instead it automatically inherits parameters from the data sets used to define it).

2.2 Limitations in 2.1

The follow limitations apply to the data set joins in BIRT 2.1

2.2.1 Joint Data Set Limited to Two tables

A Joint Data Set may only join two tables. It is however still possible to produce multi-data set joins by chaining joint data sets, since a joint data set may be used to define another joint data set

2.2.2 Single Column Equality Joint Condition

A Joint Data Set may only define a single joint condition between the two tables. Such a joint condition can only be an equality comparison of two data set columns, one from

each data set. To illustrate, suppose that data set D1 (which has result set column c1, a1) is joined to data set D2 (which has result set column c2, a2), then

row[“c1”] == row[“c2”] is a valid join condition

row[“c1”] >= row[“c2”] is not a valid join condition (not an equality comparison)

row[“c1”] *2 == row[“c2”] is not a valid join condition (must join on columns, not expressions; consider creating a computed column on D1 if such a join condition must be used)

row[“c1”] == row[“a1”] is not a valid join condition (both columns are from one data set)

row[“c1”] == row[“c2”] && row[“a1”] == row[“a2”] is not a valid join condition (more than one comparison is included; if such a join condition must be used, consider create computed columns to combine the values of multiple columns in the join key to a single value)

2.3 Features for Future Consideration

2.3.1 Data Set Unions

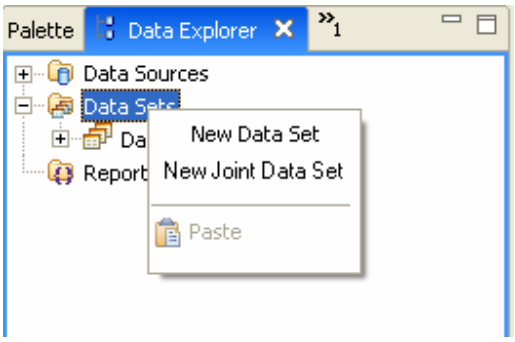
Joint data set may be expanded to produce a *union* of two or more data sets.

3. Creating Joint Data Set in BIRT Designer

3.1 Menu Option to Create New Joint Data Set

In BIRT Designer’s “Data” menu, a new item, “New Joint Data Set” will be added. This menu item will also be added to the menu when the user right clicks on the “Data Sets” node in the Data Explorer view (next to the existing menu item “New Data Set”).

Selecting “New Joint Data Set” menu item will cause the Joint Data Set Wizard to be brought up.



3.2 Joint Data Set Wizard

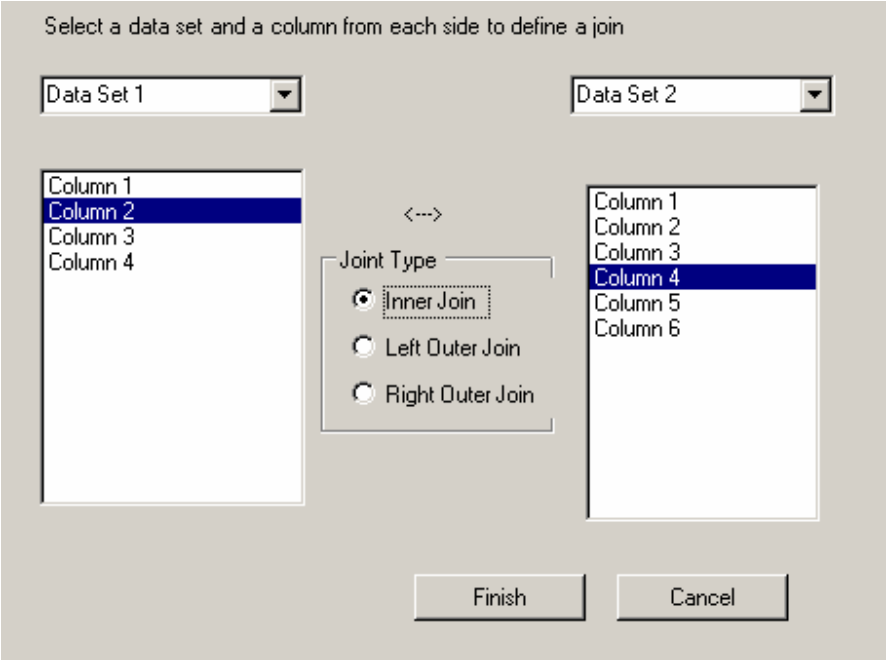
The Joint Data Set Wizard helps a user create a new joint data set. In this wizard the user chooses

(1) Two data sets (note that they can be the same data set) to join

(2) One column from each data set to define the equality join condition

(3) A join type (inner, left outer or right outer; inner join is the default)

The next diagram is a rough mockup of the wizard dialog. The wizard may choose to implement a series of dialogs (e.g., the first dialog picks the table, the 2nd dialog picks the join columns and join type etc.)



Upon finishing the Joint Data Set wizard, the regular Data Set Editor dialog is brought up so that the user can define filters and computed columns on the joint data set.

3.3 Joint Data Set Editor

When the “Edit” action is selected for a joint data set, the Joint Data Set Editor is brought up. This editor dialog is similar to the editor of a regular ODA data set, with these exceptions:

(1) There is no Query or Data Source tab (A joint data set is not associated with a data source)

(2) There is no Property Binding tab (joint data set does not have properties)

(3) There is no Cache Preference tab (we do not plan to cache a joint data set result)

4. Using a Joint Data Set in BIRT Designer

A joint data set can be used wherever a data set is accepted. However a joint data set has some special characteristics due to the derivative nature of its data and metadata.

4.1 Data Set Parameters

A joint data set combines the parameters (both input and output) of the two data sets that it joins.

In the Joint Data Set Editor, the user may view the list of combined parameters. However the user may not add, remove or modify any parameter.

If the two joined data sets have parameters with identical names, a name clash will occur. BIRT designer will report this as an error. The user should rename the ambiguous parameter in one of the originating data sets to resolve the ambiguity.

4.2 Output Columns

A joint data set combines the output columns of the two data sets that it joins.

In the Joint Data Set Editor, the user may view the list of combined output columns. However the user may not modify the definition of any column.

If the two joined data sets have columns with identical name, a ROM expression which refers to a column by that name will result in ambiguity. In order to resolve such ambiguity, a result set column of a joint data set may be identified by name "*DataSetName*::*ColumnName*". For example, if data sets D1 and D2 are joined to produce a joint data set J, a table which binds to data set J may contain the following expressions:

row["col"]

row["D1::col"]

Both expressions are valid references to the value of column *col* in D1. However if data set D2 also has a column named *col*, expression *row["col"]* is an ambiguous reference and may produce undefined results.

4.3 Scripts

Joint data set does not support event scripts.

5. ROM and Model API Enhancements

5.1 Rules to support joint data set

- The source data sets are input by the sequence they are used.
- Joint conditions are between the adjacent data sets and the columns of the join condition are from the adjacent data sets.
- Joint conditions are single conditions, the comparison operation between the columns can be equality or others. (Multiple conditions can be supported by multiple single conditions, which won't be supported in 2.1 M5)

5.2 Snippet of design file

<joint-dataset>

```
        <property-list name=”source-datasets”>
            <property name=”dataset”>dataset1</property>
          <property name=”dataset”>dataset2</property>
          ……..
      </property-list >

      < list-property name=”joint-conditions”>

          <structure name=”join-condition”>
              <property name=”jointType”>inner</property>
              <property name=”jointOperator”>eq</property>
              <property name=”leftDataset”>leftDataset</property>
              <property name=”rightDataset”>rightDataset</property>
              <property name=”leftExpression”>leftExpression </property>
              <property name=”rightExpression”>rightExpression </property>
           </structure>
           ……
       </list-property>

    </joint-dataSet>
```

5.3 ROM Definition

```
<ChoiceType name="jointType">
  <Choice displayNameID="Choices.jointType.inner" name="inner"/>
  <Choice displayNameID="Choices.jointType.leftOut" name="left-out"/>
  <Choice displayNameID="Choices.JointType.rightOut" name="right-out"/>
 </ChoiceType>


 <ChoiceType name="jointOperator">
  <Choice displayNameID="Choices.JointOperator.equal" name="eq"/>
 </ChoiceType>


 <Element allowsUserProperties="true" canExtend="true"
displayNameID="Element.JointDataset" extends="ReportElement" hasStyle="false"
isAbstract="false" isNameRequired="true"
javaClass="org.eclipse.birt.report.model.elements.JointDataSet" name="jointDataset"
nameSpace="dataset" since="2.1" xmlName="joint-dataset">
  <Property subType="element" detailType="Dataset"
displayNameID="Element.JointDataset.sourceDatasets" name="sourceDatasets"
runtimeSettable="true" since="2.1" type="list"/>
```

```
    <Property detailType="JointCondition"
displayNameID="Element.JointDataset.JointCondition" isList="true"
name="jointConditions" runtimeSettable="true" since="2.1" type="structure"/>
  </Element>

  <Structure displayNameID="Structure.JointCondition" name="jointCondition"
since="2.1">
    <Member detailType="JointType"
displayNameID="Structure.JointCondition.jointType" isIntrinsic="true" name="jointType"
runtimeSettable="true" since="2.1" type="choice">
      <Default>inner</Default>
    </Member>
    <Member detailType="JointOperator"
displayNameID="Structure.JoinyCondition.jointOperator" isIntrinsic="true"
name="jointOperator" runtimeSettable="true" since="2.1" type="choice">
      <Default>eq</Default>
    </Member>
    <Member displayNameID="Structure.JointCondition.leftDataset" isIntrinsic="true"
name="leftDataset" runtimeSettable="true" since="2.1" type="string"
valueRequired="true"/>
    <Member displayNameID="Structure.JointCondition.rightDataset" isIntrinsic="true"
name="rightDataset" runtimeSettable="true" since="2.1" type="string"
valueRequired="true"/>
    <Member displayNameID="Structure.JointCondition.leftExpression" isIntrinsic="true"
name="leftExpression" returnType="any" runtimeSettable="true" since="2.1"
type="expression" valueRequired="true"/>
    <Member displayNameID="Structure.JointCondition.rightExpression" isIntrinsic="true"
name="rightExpression" returnType="any" runtimeSettable="true" since="2.1"
type="expression" valueRequired="true"/>
  </Structure>
```

5.4 Model API

JointDataset and JointCondition are main elements need to be added into model. As before a respective handle will be provided for each element, which is the public API for other modules to use these elements. In this case, the handles are JointDatasetHandle and JointConditionHandle. Followed are the detailed APIs for the new added components in Model.

5.4.1 JointDataset (org.eclipse.report.element):

No special API in JointDataSet.

5.4.2 JointCondition:

Methods:

String getJointType()

String getOperator();

String getLeftDataset();

String getRightDataset();

String getLeftExpression();

String getRightExpression();

void setJointType(String jointType);

void setOperator(String operator);

void setLeftDataset(String datasetName);

void setRightDataset(String datasetName);

void setLeftExpression(String expression);

void setRightExpression(String expression);

5.4.3 JointDatasetHandle

List getDataSetNames();

void addDataset(String datasetName);

Iterator jointConditionsIterator( )

5.4.4 JointConditionHandle:

Interfaces of JointCondition are same as those of JointConditon.

6. Data Engine API Enhancements

The following interfaces/classes are to be added to Data Engine API to accommodate new requirements.

6.1 Package org.eclipse.birt.data.engine.api

6.1.1 Interface IJointDataSetDesign.

This interface describes the static design of a Joint Data Set. It extends interface org.eclipse.birt.data.engine.api.IBaseDataSetDesign.

The following table lists the static constants of this interface.

Constant Signature	Description

public static int LEFT_OUTER_JOIN = 1	The integer value stands for a left outer join operator.
public static int RIGHT_OUTER_JOIN = 2	The integer value stands for a right outer join operator.
public static int INNER_JOIN = 0	The integer value stands for an inner join operation.

The following table lists the method of this interface.

Method Signature	Description
public String getLeftDataSetDesign()	This method returns the name of data set design which servers as left operand of a joint.
public String getRightDataSetDesign()	This method returns the name of data set design which servers as right operand of a joint.
public int getJointType()	This method returns the Joint Type. There are three types of Joints. They are inner joints, left outer join, right outer join. Their corresponding integer values are defined in this interface as well.
public List getJointConditions()	This method returns the Joint conditions. Only rows which can make these IJointConditionalExpression instance evaluate to true will be jointed.

6.1.2 Interface IJoinCondition.

This interface describes a specified conditional expression which returns a Boolean value, and used for Joint Data Set only.

The following table lists the static constants of this interface.

Constant Signature	Description
public static int OP_EQ = 1	The integer value stands for an equality operator. It is evaluated to true if the result of first expression is equal to that of second expression.

The following table lists the method of this interface.

Method Signature	Description
public IScriptExpression getLeftExpression()	This method returns the IScriptExpression instance which will be evaluated against the data set which servers as left operand of a joint.
public IScriptExpression	This method returns the IScriptExpression instance

getRightExpression()	which will be evaluated against the data set which servers as right operand of a joint.
public int getOperator()	This method returns the integer standing for a compare operator. All supported compare operators are defined in this interface.

6.2 Package org.eclipse.birt.data.engine.api.queryDefn

6.2.1 Class JointDataSetDesign.

This class extends org.eclispe.birt.data.engine.queryDefn.BaseDataSetDesign, and implements org.eclipse.birt.data.engine.api.IJointDataSetDesign. It is a default implementation of latter.