Scope Tutorial > Combining Rowsets with JOINs with JOINs

Author: Saveen Reddy

Date: 3/15/2015

# Overview

**JOIN** operators combine rowsets into a new rowset based on matching a field in one RowSet to a field in another RowSet.

Before we delve into the syntax, we will spend some time making sure we are clear on the intended behavior of joins. To make this very simple we will illustrate using some basic datasets.

# INNER JOINs and OUTER Joins

## Example Dataset 1

Consider the following rowsets:



* An **INNER JOIN** of these two rowsets will return the combined rows where the **DepID** field matches.
* A **LEFT OUTER JOIN** is a superset of the **INNER JOIN** that includes ALL the rows from the left rowset. Because D3 is not present in the right rowset, NULL values are used for that output row.
* A **RIGHT OUTER JOIN** is a superset of the **INNER JOIN** that includes ALL the rows from the right rowset. Because D2 is not present in the right rowset, NULL values are used for that output row.
* A **FULL OUTER JOIN** is a combination of the **LEFT OUTER** and **RIGHT OUTER JOIN**
* Using **OUTER JOIN** without specifying **LEFT**, **RIGHT**, or **FULL** indicates a **LEFT OUTER JOIN**.



## Example Dataset 2

Let's add an extra employee to the original data. In this case Smith is also assigned to D1. See how this changed the results:



## Example Dataset 3

Now let's add another department – but notice that this department is also assigned to D1.



# CROSS JOIN

CROSS JOIN is basically an INNER JOIN without a join condition – it returns all the possible combinations of records. This is the *Cartesian product* of the records in both rowsets.



## SEMIJOIN

It is easiest to think of **SEMIJOIN** as a way to filter a rowset.

There are two variants: **LEFT SEMIJOIN** and **RIGHT SEMIJOIN**.

LEFT SEMIJOIN -> Give only those rows in the **left rowset** that have a matching row in the **right rowset**.

RIGHT SEMIJOIN -> Give only those rows in the **right rowset** that have a matching row in the **left rowset**.

NOTE: If you leave out **LEFT** or **RIGHT**, and instead simply write **SEMIJOIN** then what you get is **LEFT SEMIJOIN**. Do not leave out **LEFT** or **RIGHT** always explicitly it..

Example: Find all employees that are in valid departments (find all the employees in the left rowset that have a departmentid that is listed in the right rowset)

rs0 =

SELECT employees.DepID,

employees.EmpName

FROM employees

LEFT SEMIJOIN departments

ON employees.DepID == departments.DepID;

Example: Find all departments with at least one employee

rs1 =

SELECT departments.DepID,

departments.DepName

FROM departments

LEFT SEMIJOIN employees

ON departments.DepID == employees.DepID;

## Simulating an ANTIJOIN

SQL has an **ANTIJOIN** (<http://en.wikipedia.org/wiki/Relational_algebra#Antijoin>) operator, but Scope does not.

You can achieve **ANTIJOIN** using this method:

FilterRowSet = SELECT DISTINCT JoinKey FROM FilterRowSet;

Result =

SELECT \*

FROM DataSet

LEFT OUTER JOIN FilterRowSet ON DateSet.JoinKey == FilterRowSet.JoinKey

WHERE FilterRowSet.JoinKey == null;

# Don’t Use Implicit JOINs

You will at some point come across a JOIN that looks like this:

rs0 =

SELECT

employees.DepartmentID AS EmpDepartmentId,

departments.DepartmentID AS DepDepartmentID,

employees.LastName,

departments.DepartmentName

FROM **employees , departments**

**WHERE employees.DepartmentID == departments.DepartmentID;**

Some things to notice:

* There’s a JOIN operation specified.
* The FROM clause contains a comma-separated list of rowsets.
* There’s a WHERE clause instead of an ON clause.

This example uses an **Implicit INNER JOIN**. We want you to **always be explicit** about your joins. **Don’t use Implicit INNER JOIN, instead make it explicit.**

ProTip: in some SQL variants – but not Scope - the WHERE condition is optional. When it doesn’t appear, this implicit join behaves as a CROSS JOIN. As you can see, it’s easy to accidently do an (expensive) CROSS JOIN when using such a syntax. It’s for this reason Implicit Join should be avoided even in languages that support it.

# JOIN Examples

To help make sure JOINs are understood, the samples from the Wikipedia article on joins is included:

<http://en.wikipedia.org/wiki/Join_(SQL)>

employees =

EXTRACT EmpName:string, DepID:string

FROM @"my/ScopeTutorial/SampleInputs/Employees.txt"

USING DefaultTextExtractor();

departments =

EXTRACT DepID:string, DepName:string

FROM @"my/ScopeTutorial/SampleInputs/departments.txt"

USING DefaultTextExtractor();

### INNER JOIN

rs\_inner =

SELECT employees.DepID AS EmpDepId, departments.DepID , employees.EmpName, departments.DepName

FROM employees

INNER JOIN departments ON employees.DepID == departments.DepID;

OUTPUT rs\_inner TO @"Output.txt";

NOTE: If you leave out **INNER** or **OUTERR**, and instead simply write **JOIN** then what you get is **INNER JOIN**. Do not leave out **INNER** or **OUTER.** Always explicitly it..

### LEFT OUTER JOIN

rs0 =

SELECT employees.DepID AS EmpDepId, departments.DepID , employees.EmpName, departments.DepName

FROM employees

LEFT OUTER JOIN departments ON employees.DepID == departments.DepID;

### RIGHT OUTER JOIN

rs0 =

SELECT employees.DepID AS EmpDepId, departments.DepID , employees.EmpName, departments.DepName

FROM employees

RIGHT OUTER JOIN departments ON employees.DepID == departments.DepID;

NOTE: If you leave out **LEFT** or **RIGHT**, and instead simply write **OUTER JOIN** then what you get is **LEFT OUTER JOIN**. Do not leave out **LEFT** or **RIGHT**. Always explicitly it..

### FULL OUTER JOIN

rs0 =

SELECT employees.DepID AS EmpDepId, departments.DepID , employees.EmpName, departments.DepName

FROM employees

FULL OUTER JOIN departments ON employees.DepID == departments.DepID;

### LEFT SEMIJOIN

rs0 =

SELECT employees.DepID AS EmpDepId, employees.EmpName

FROM employees

LEFT SEMIJOIN departments ON employees.DepID == departments.DepID;

### RIGHT SEMIJOIN

rs0 =

SELECT departments.DepID , departments.DepName

FROM employees

RIGHT SEMIJOIN departments ON employees.DepID == departments.DepID;

### CROSS JOIN

rs0 =

SELECT employees.DepID AS EmpDepId, departments.DepID , employees.EmpName, departments.DepName

FROM employees

CROSS JOIN departments;

PROTIP: Using **RIGHT OUTER JOIN**, **RIGHT SEMIJOIN** and **FULL OUTER JOIN** is quite rare. INNER JOIN, LEFT OUTER JOIN, AND LEFT SEMIJOIN are fairly common. Aside from certain useful scenarios, CROSS JOIN should be avoided since it typically causes large data explosion.

# Custom JOINs via Combiners

You can implement your own JOINs with Combiner UDOs.

This section will not go into how to create one, but we will illustrate the usage in a Scope Script.

RS2 =

COMBINE left WITH right

ON left.X == right.X

USING MyCombiner()

PRODUCE A,B,C,D;

As you can see:

* Multiple rowset inputs are identified
* Rows are combined ON a specific column from each rowset
* The USING clause specifies the combiner
* The PRODUCE clause identifies the schema that is returned from the operation