Scope Tutorial > Combining Rowsets with Set Operators

Author: Saveen Reddy

Date: 3/15/2015

# Introduction

Sometimes you must construct a rowset from two or more other rowsets. Scope gives you three ways of doing this:

* JOINs
* Set-Theoretic operations
* Combiner UDOs

# Combining Rowsets with Set Operations

To start, let's assume we have two rowsets as shown below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | A | | | id:int | Name | | 1 | Smith | | 1 | Smith | | 2 | Brown | | 3 | Case | | |  |  | | --- | --- | | B | | | id:int | Name | | 1 | Smith | | 1 | Smith | | 1 | Smith | | 2 | Brown | | 4 | Dey | | 4 | Dey | |

Let's load these datasets from the samples:

a = EXTRACT

Id:int, Name:string

FROM @"/my/SampleInputs/SetOps\_A.txt"

USING DefaultTextExtractor();

b = EXTRACT

Id:int, Name:string

FROM @"/my/SampleInputs/SetOps\_B.txt"

USING DefaultTextExtractor();

# Merging Rows with UNION

Combining two rowsets is done with the **UNION** operator.

**UNION ALL** will preserve any duplicates while **UNION** will remove them.

union\_distinct = SELECT \* FROM a

UNION DISTINCT

SELECT \* FROM b;

union\_all = SELECT \* FROM a

UNION ALL

SELECT \* FROM b;

OUTPUT union\_distinct

TO @"/my/Outputs/union\_distinct.txt";

OUTPUT union\_all

TO @"/my/Outputs/union\_all.txt";

|  |  |
| --- | --- |
| UNION DISTINCT | UNION ALL |
| |  |  | | --- | --- | | id:int | Name | | 1 | Smith | | 2 | Brown | | 3 | Case | | 4 | Dey | | |  |  | | --- | --- | | id:int | Name | | 1 | Smith | | 1 | Smith | | 2 | Brown | | 3 | Case | | 1 | Smith | | 1 | Smith | | 1 | Smith | | 2 | Brown | | 4 | Dey | | 4 | Dey | |

# Finding Common Rows with INTERSECT

Sometimes, we only care about the rows both rowsets have *in common*. We use the **INTERSECT** operator to accomplish this. **INTERSECT ALL** preserves duplicates while **INTERSECT** removes duplicates.

rs1 = SELECT \* FROM a

INTERSECT DISTINCT

SELECT \* FROM b;

rs2 = SELECT \* FROM a

INTERSECT ALL

SELECT \* FROM b;

OUTPUT rs1

TO @"/my/Outputs/intersect.txt";

OUTPUT rs2

TO @"/my/Outputs/intersect-all.txt";

|  |  |
| --- | --- |
| INTERSECT ALL | INTERSECT DISTINCT |
| |  |  | | --- | --- | | id:int | name | | 1 | Smith | | 1 | Smith | | 2 | Brown | | |  |  | | --- | --- | | id:int | name | | 1 | Smith | | 2 | Brown | |

# Finding Rows That Are NOT in the Other RowSet with EXCEPT

The **EXCEPT** operator returns all the rows in the left RowSet that *are not* in the right RowSet.

rs0 = SELECT \* FROM a

EXCEPT DISTINCT

SELECT \* FROM b;

rs1 = SELECT \* FROM a

EXCEPT ALL

SELECT \* FROM b;

rs2 = SELECT \* FROM b

EXCEPT DISTINCT

SELECT \* FROM a;

rs3 = SELECT \* FROM b

EXCEPT ALL

SELECT \* FROM a;

OUTPUT rs0 TO @"/my/Outputs/except\_distinct\_a\_b.txt";

OUTPUT rs1 TO @"/my/Outputs/except-all\_a\_b.txt";

OUTPUT rs2 TO @"/my/Outputs/except\_distinct\_b\_a.txt";

OUTPUT rs3 TO @"/my/Outputs/except-all\_b\_a.txt";

|  |  |  |  |
| --- | --- | --- | --- |
| EXCEPT ALL (A,B) | EXCEPT DISTINCT (A,B) | EXCEPT ALL (B,A) | EXCEPT DISTINCT (B,A) |
| |  |  | | --- | --- | | id:int | name | | 3 | Case | | |  |  | | --- | --- | | id:int | name | | 3 | Case | | |  |  | | --- | --- | | id:int | name | | 1 | Smith | | 4 | Dey | | 4 | Dey | | |  |  | | --- | --- | | id:int | name | | 4 | Dey | |