Article
Peter Steiwer · Jan 6, 2020 4m read

Filtering MDX results with SQL using %SQLRESTRICT

What is %SQLRESTRICT

%SQLRESTRICT is a special %FILTER clause for use in MDX queries in InterSystems IRIS Business Intelligence. Since this function begins with %, it means this is a special MDX extension created by InterSystems. It allows users to insert an SQL statement that will be used to restrict the returned records in the MDX Result Set. This SQL statement must return a set of Source Record IDs to limit the results by. Please see the documentation for more information.

Why is this useful?

This is useful because there are often times users want to restrict the results in their MDX Result Set based on information that is not in their cubes. It may be the case that this information may not make sense to be in the cube. Other times this can be useful when there is a large set of values you want to restrict. As mentioned before, this is not a standard MDX function, it was created by InterSystems to handle cases were queries were not performing well or cases that were not easily solved by existing functions.

Does %SQLRESTRICT replace any existing features?

%SQLRESTRICT will work alongside a couple existing features, but it opens the door for new possibilities.

1) Currently Build Restrictions allow users to only include data into their cube that meets a certain criteria. Build Restrictions are also SQL statements but the main difference is that Build Restrictions happen at build time, %SQLRESTRICT happens at run time. Both of these have their pros and cons. For example, a Build Restriction will directly limit the number of records in your cube. This means that queries will never even bother looking at other records to see if they meet the given criteria. However, if you want to use multiple Build Restrictions against the same set of data, you will need to have multiple cubes (one for each Build Restriction). %SQLRESTRICT needs to generate the index for which records to include on the fly. This has the benefit of allowing you to test out different restrictions without possibly needing to wait hours for your cube to build. Since multiple %SQLRESTRICT clauses can be included in your Cube, these different filters can be toggled on and off. You can easily switch between different restrictions or no restrictions at all. This can all be done from a single cube definition.

2) Searchable measures function in a similar way to %SQLRESTRICT since they can help efficiently query dynamic conditions without needing to build the exact condition into your model. Both of these will create a new index at runtime to use within the query. %SQLRESTICT will query against your Source Data, which means it does not require an index on the data being used to restrict to exist in the cube like searchable measures do.

3) Large sets of members do not always work well in MDX. It is possible that this can cause poor query performance. This is because each time context of a new member is used within an MDX statement, the engine needs to do additional work to either include or exclude the results for the given member. Using %SQLRESTRICT can force the SQL engine to generate an index to be used within the DeepSee engine without the DeepSee engine needing to do a lot of the necessary calculations.

4) Perhaps you already have an existing optimized SQL query that defines a cohort. This query can easily be plugged into %SQLRESTRICT and used within MDX to perform additional analysis on this specific group. If you have multiple cohorts, you can easily toggle between the different cohorts if they are used dynamically as a filter.

Example

In the HoleFoods Sample (included in the Samples-BI GitHub Repository), our cube contains grocery store sales. Each record has an associated location in which it was sold. This location contains latitude and longitude information. We do not currently have this information built into our cube and we do not think it would be useful to build into the cube since we already have a dimension that contains "Region", "Country", and "City". A new dried fruit product is being introduced and will be marketed towards colder climates. We would like to monitor how fruit sales in these areas are changing. Instead of going through by hand and picking a list different cities that we think may meet the target criteria, we can simply create a new calculated member that uses %SQLRESTRICT to get a list of Sales Records that meet our criteria. My Calculated Member can have the following expression:

```
%SQLRESTRICT.&[SELECT ID FROM HoleFoods.SalesTransaction WHERE Outlet->Latitude>45 OR
 Outlet->Latitude<-45]
```

As the documentation states, this expression can also start at the WHERE clause:

```
%SQLRESTRICT.&[WHERE Outlet->Latitude>45 OR Outlet->Latitude<-45]
```

I can now use this Calculated Member as a FILTER in my MDX query, or I can use it on Rows/Columns. This can be used in addition to regular filters so that we can filter on both this new Latitude restriction AND Product Category=Fruit.

#Analytics #Analyzer #Dashboards #MDX #Performance #SQL #Tips & Tricks #InterSystems IRIS #InterSystems IRIS BI (DeepSee)