Guideline

Advanced Multidimensional Reporting

Product(s): IBM Cognos 8 Report Studio
Area of Interest: Report Design
Copyright

Copyright © 2008 Cognos ULC (formerly Cognos Incorporated). Cognos ULC is an IBM Company. While every attempt has been made to ensure that the information in this document is accurate and complete, some typographical errors or technical inaccuracies may exist. Cognos does not accept responsibility for any kind of loss resulting from the use of information contained in this document. This document shows the publication date. The information contained in this document is subject to change without notice. Any improvements or changes to the information contained in this document will be documented in subsequent editions. This document contains proprietary information of Cognos. All rights are reserved. No part of this document may be copied, photocopied, reproduced, stored in a retrieval system, transmitted in any form or by any means, or translated into another language without the prior written consent of Cognos. Cognos and the Cognos logo are trademarks of Cognos ULC (formerly Cognos Incorporated) in the United States and/or other countries. IBM and the IBM logo are trademarks of International Business Machines Corporation in the United States, or other countries, or both. All other names are trademarks or registered trademarks of their respective companies. Information about Cognos products can be found at www.cognos.com

This document is maintained by the Best Practices, Product and Technology team. You can send comments, suggestions, and additions to cscogpp@ca.ibm.com.
1 Introduction

The multidimensional reporting features in ReportStudio provide report authors with powerful tools to create more succinct and meaningful reports. This presentation provides insight into the use of some of these features. Advanced report authors will learn techniques to reduce large sets of data into smaller, more readable reports, as well as advanced filtering techniques that will render more useful data.

As well, there are new techniques to specify how to arrange and organize complex crosstab definitions that will help organize reports and make them more easily understood by report consumers. Report authors can expect to gain knowledge about how to set up a layout specification that will help them design more complicated reports.

With these more advanced features, it also becomes easier to generate data queries that are very execution-time consuming and processor-intensive. There are certain techniques that can be used to limit the amount of processing that is required by the underlying data providers, resulting in a report that can be rendered more quickly without sacrificing usefulness of readability.

It is assumed that attendees will have a good knowledge of Report Studio, as well as some level of familiarity with the expression syntax used for generating complex reports. A basic level of familiarity with OLAP concepts would also be helpful.

2 Segmentation of data

Data segmentation is exactly what the name implies; a way of breaking your data into discrete chunks. This can have a positive effect on the readability of the report, as well as its execution time, but what is the best way of breaking up the data? At a very high level, this can be broken down into two segments… data the user is interested in seeing, and data that the user is not as interested in seeing. This can be broken down somewhat, as follows.

- Visible members
  - In multidimensional models, there are relationships between members that can be leveraged to get more precise information. Objects in a multidimensional model
include dimensions, hierarchies, levels and members. For a more in-depth look at what multidimensional models look like, please refer to Appendix A.

- It is possible for an excessively large number of members to be returned from an expression, causing the report to run very slowly. E.g. “children of Products” could result in many thousands of members to be rendered in the report, making it difficult to extract any useful information.

- Use something like the “head” function to limit the number of children returned. The head function limits the number of members returned in a result set. This can give a sampling of what the members are without listing all of them. There are a few functions that can help limit the amount of data returned to a predictable amount.
  
  Example:
The following example will always return a maximum of 5 members, regardless of how many children 1996 actually has.

  head( children( [1996] ), 5 )

- This is your “visible” members segment. There is more information you can place in the report to make this more useful.

  - Summary
    - Often, users would like to see which member acts as the root of the query. The root is the member that is used in an expression to generate a result set of members. For example, the expression “children of Products” has the member Products as the root of the query. In a typical hierarchy, its measure value is the aggregation of all its child members. In a “members at level” query, the summary’s value would be the aggregation of all the members that belong to that level.

  - Included Subtotal
    - Displaying a limited numbers of members of a given set causes the summary segment to misrepresent the aggregation of the members in the visible set. It could be very useful to be able to see just the aggregation of the visible members. This can be accomplished by using a combination of the “aggregate” and “member” in order to create a calculation.

      Example:
      aggregate( currentMeasure WITHIN SET [visible_members] )
Along with displaying the total of all the visible members, it is possible to display the total of all the members of the expression that are not visible. There are a couple of ways that this can be accomplished. The first is to try to define a set of all the non-visible members, and aggregate that set for the current measure. The problem with this approach is that it can cause significant performance issues, since there could be huge amounts of members that are being aggregated. A somewhat more direct way is to simply subtract the included subtotal value from the summary value.

Example:
\[
\text{[summary]} - (\text{aggregate( currentMeasure WITHIN SET [visible_members] )})
\]

### 3 Filtering

With filtering, a user can start to really see more useful information. Instead of just retrieving the first “n” members of a set of data, the members returned can start to take on more meaning. For instance, if the user wants to see the top 10 earners for a given quarter, a “Top” filter can be applied to the set of all sales people in an organization. Given the correct context, it is quite straightforward to see who the best sales people are.

The different methods of filtering are…

- **Top/Bottom filters**
  - User can choose to see either the top “n” or bottom “n” members of a set of member given a specific context.
  
  Example:
  \[
  \text{TopCount( children of salespeople, 10, tuple([Revenue], [1996 Q1]) )}
  \]

- **Measure and Attribute filters**
  - The user can use a measure filter to filter out all members that do not meet a measure value criteria set out in the filter definition.
  
  Example:
  \[
  \text{filter( children of salespeople , ( tuple([Revenue], [1996 Q1]) > 120 ) )}
  \]

  The result set of members contains all the child nodes of salespeople who have revenue greater than 120 for the first quarter of 1996.

- **Slicer**
The slicer is a way of filtering data values in a crosstab without affecting the sets of members along the edges of the crosstab. An example...

```xml
<slicer>
  <slicerMemberSet>[PC].[Line (Root)].[Dishwashers]</slicerMemberSet>
</slicer>
```

There are some circumstances where the slicer can affect the members displayed on the edge of a crosstab. This can happen if the members along an edge are filtered based on the values on the crosstab. Changing the values can cause a change in which members pass the filter criteria, thereby causing different members to be displayed.

### 3.1 Filter Context

The filter context provides a specification of what values a filter is to be executed against. For example, if the user wishes to see the top ten sales people for a given quarter, it is necessary to specify which measure to use, as well as to specify which quarter. The measure and the quarter are considered to be the context for the filter. A filter’s context can be described using either of two functions, namely...

- **tuple**
  - The tuple function serves to specify the context that a filter will operate against. In the case of a filter such as “Top 10 SalesPeople for Q1 of 2003”, the context would likely be the measure (e.g. revenue) and the member 1996 Q1. Sample syntax would be
    ```sql
    TopCount( children of salespeople, 10, tuple([Revenue], [1996 Q1])
    )
    ```

    If a dimension is not represented in the tuple function for a filter, then the default member of each dimension is used for the context.

- **completeTuple**
  - The completeTuple function is a variant on the tuple function. Ordinarily, the slicer and the evaluation context can dictate a certain amount of context and consequently influence the numbers that are evaluated for a filter. The completeTuple function prevents this from happening. Only the members referenced in the completeTuple expression are used for the context. Any dimension not referenced in it will have its default member used for the evaluation of the filter expression. Why would a user wish to do this? It
provides a greater deal of control over which members appear along the edge of a crosstab. Regardless of what might be placed in the slicer, or what levels of nesting there are, the numbers being evaluated are consistent, and the result set of the filter expression will always be the same. The syntax is similar to the tuple function, as in the following example.

- TopCount( children of salespeople, 10, completeTuple([Revenue], [1996 Q1]) )

### 3.2 Sorting

With the Top and Bottom filter, there is an automatic sorting that is applied to the result set. Measure/Attribute filter, however, do not sort the results. Unfiltered sets of members are unsorted as well. The “order” function is used to sort result data in a crosstab. The method of specifying a sort is quite straightforward, as shown here…

order( <set expression>, tuple(<context>), asc|desc )

The set expression above could correspond to any set of visible members, filtered or not. Keep in mind that if the set is filtered by a Top or Bottom filter, then sorting is redundant. The tuple, as described above, would determine the context that provides the data values used for the sorting. The last parameter states whether the sorted members come back in ascending or descending order.

### 3.3 Ranking

Ranking is a feature that many people have found very useful. Its purpose is, as demonstrated by its name, to rank a specific group of members within a given set. How to define that group of members, or the set within which the members are ranked, can be quite simple with a basic use case, but it can also be quite tricky for more complicated cases.

Syntax:

rank(<measure> <ordinal direction> TUPLE <member being ranked> WITHIN SET <visible members on opposite axis>)

Simple use case:

User wants to see the rank of each Product Line for the year 1993.
rank(currentMeasure DESC tuple [1993] WITHIN SET [members of Product Line])

In this example, we are using the current measure defined in the crosstab. A specific measure reference can also be used. The ordinal direction is a parameter that determines whether the highest values correspond to the lowest ordinal (DESC) or the lowest values correspond to the lowest ordinal (ASC). The tuple describes which member (or intersection of members) is being ranked. The WITHIN SET clause describes the set of members that are used to generate the rank ordinals.

An extension to this test case is to crossjoin the rank calculation within another set of members. As a result, the rank calculation will repeat under each member of the outer nesting group. The rank calculation, however, does not need to change, since the evaluation context will take care of re-evaluating the rank ordinals for each separate instance of the rank calculation. The outer member provides the extra context that will influence the values being ranked.

Another extension to ranking is to rank a member’s value against its sibling members’ values (sibling members are members from the same level who have the same parent). In other words, don’t rank against the opposite axis, but instead use the same axis as the member being ranked. For instance, assume all the children of the Date member are present along the column axis, and they include members 1993, 1994, and 1995. It is possible to create a rank calculation for 1993 along the same axis that ranks the value of 1993 against the values of 1994 and 1995. A different rank ordinal will be generated for each member of the opposite axis. The difference in the expression is to reference the visible members along the same axis as the member being ranked. With our test case above, it would look like …

rank(currentMeasure DESC tuple [1993] WITHIN SET [Years])

There are other variations on the rank calculation, all of which require a modification of the WITHIN SET clause to get different results. A description of other advanced ranking examples is in Appendix B.
4 Complex Axis Definition

4.1 Layout Specification

The layout specification can be used to define a crosstab axis of any level of complexity. The exact syntax of the layout specification is the same for both relational and multidimensional reports, but with multidimensional reporting, the layout serves more as a map of how generated sets of members interact with each other. The ability to group related dataItems together to act as a single entity can be leveraged to allow the definition of more complex crosstabs. There are many ways to create a layout of a variety of dataItems. A single crosstabNode can be created with all segments in it, related or unrelated. Optionally, a different crosstabNode can be created for each segment. The alternative that lends itself more readily to complex axis definition is to have a crosstabNode created for each group of related segments, with each segment represented by a crosstabNodeMember. This approach allows for a logical grouping of segments that can interact with other groups of segments in a meaningful way. That is to say, each crosstabNode can then be combined with other crosstabNodes to create a variety of useful crosstab edges.

There are two primary methods of showing how to combine crosstabNodes (groups of related dataItems). These are Nesting and Unioning.

4.2 Nesting

In order to nest different crosstabNodes, it is necessary to use the crosstabNestedNodes object. This is a sibling object to the crosstabNode, but implies that any crosstabNodes within the crosstabNestedNodes will be nested beneath the sibling crosstabNodes. The following example demonstrates a cross-join between a set of members of the Line dimension and a set of members from the Market dimension.

Example:

```xml
<crosstabRows>
  <crosstabNode>
    <crosstabNodeMembers>
      <crosstabNodeMember class="ml" refDataItem="Line (visible items set)">
        <contents>
          ...
        </contents>
      </crosstabNodeMember>
    </crosstabNodeMembers>
  </crosstabNode>
</crosstabRows>
```
This would create a crosstab with a row axis that looks as follows…

<table>
<thead>
<tr>
<th></th>
<th>Revenue</th>
<th>1993</th>
<th>1994</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dishwashers</strong></td>
<td>Builders</td>
<td>56710</td>
<td>233298</td>
</tr>
<tr>
<td></td>
<td>Furniture</td>
<td>35845</td>
<td>40282</td>
</tr>
<tr>
<td></td>
<td>Home</td>
<td>36189</td>
<td>10852</td>
</tr>
<tr>
<td><strong>Stoves</strong></td>
<td>Builders</td>
<td>63478</td>
<td>80227</td>
</tr>
<tr>
<td></td>
<td>Furniture</td>
<td>50890</td>
<td>27024</td>
</tr>
<tr>
<td></td>
<td>Home</td>
<td>33437</td>
<td>54128</td>
</tr>
<tr>
<td><strong>Microwaves</strong></td>
<td>Builders</td>
<td>71635</td>
<td>43330</td>
</tr>
<tr>
<td></td>
<td>Furniture</td>
<td>23746</td>
<td>33082</td>
</tr>
<tr>
<td></td>
<td>Home</td>
<td>25749</td>
<td>19243</td>
</tr>
</tbody>
</table>

When the inner nesting level is from a different dimension than the outer nesting level, it is known as a “crossjoin”. It is a cross-product of the result set of members from each dimension, causing the inner result members to be repeated for each outer result member.

When the inner and outer nesting levels are from the same dimension, they are referred to as a single-dimension nesting for the purposes of this document. It is a dot-product of the result set of members from each level, implying that there are no repeated members on the inner level.
4.3 Unioning
The concept of unioning sets of members along an axis of a crosstab is quite straightforward. The report author has a couple of options in terms of how to accomplish this. As stated above, the approach that allows easier complex axis definition is to have a crosstabNode created for each group of related segments, with each segment represented by a crosstabNodeMember.

Example:

```xml
<crosstabRows>
  <crosstabNode>
    <crosstabNodeMembers>
      <crosstabNodeMember class="ml" refDataItem="Line (visible items set)">
        <contents>...
        </contents>
      </crosstabNodeMember>
    </crosstabNodeMembers>
  </crosstabNode>
  <crosstabNode>
    <crosstabNodeMembers>
      <crosstabNodeMember class="ml" refDataItem="Market (visible items set)">
        <contents>...
        </contents>
      </crosstabNodeMember>
    </crosstabNodeMembers>
  </crosstabNode>
</crosstabRows>
```

This would render a crosstab that looks as follows…
With this concept of grouping related segments, the author is free to combine these groups with ease in many ways to create very powerful reports that have meaningful content.

For a more advanced example that combines both nesting and unions, please refer to Appendix C.

**Solve Order**

Solve order is a feature used to resolve how calculations intersect. When a report author places a calculation along the edge of both the row axis and the column axis, the order of precedence of these two calculations can influence the value that is returned in the cell that is the intersection of the row and column in question.

An example helps illustrate this concept.

<table>
<thead>
<tr>
<th>Revenue</th>
<th>1993</th>
<th>1994</th>
<th>Sum (1993, 100)</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dishwashers</td>
<td>41</td>
<td>52</td>
<td>141</td>
<td>93</td>
</tr>
<tr>
<td>Stoves</td>
<td>54</td>
<td>40</td>
<td>154</td>
<td>94</td>
</tr>
<tr>
<td>Microwaves</td>
<td>26</td>
<td>34</td>
<td>126</td>
<td>60</td>
</tr>
<tr>
<td><strong>Stoves + Microwaves</strong></td>
<td><strong>80</strong></td>
<td><strong>74</strong></td>
<td><strong>280</strong></td>
<td><strong>154</strong></td>
</tr>
<tr>
<td><strong>Line</strong></td>
<td><strong>121</strong></td>
<td><strong>126</strong></td>
<td><strong>221</strong></td>
<td><strong>247</strong></td>
</tr>
</tbody>
</table>

In this case, the intersection between “Stoves + Microwaves” on the row axis intersects with “Sum(1993, 100)” on the column axis could potentially have two different answers. The resultant value could either be 280 (which is the addition of Stove and Microwaves in the column “Sum(1993, 100)”) or it could be 180 (which is the sum of 1993 and 100 for the “Stoves + Microwaves” row).

Which value is rendered depends on the relative solve order that is assigned to each calculation. If the solve order for the “Stoves + Microwaves” calculation is higher than that of
“Sum(1993, 100)”, then “Stoves + Microwaves” is the last operation performed in the intersection.

In terms of steps, it works as follows...

1. 100 is added to Stoves → result is 154.
2. 100 is added to Microwaves → results is 126.
3. The results of these two calculations are then added (Stoves + Microwaves) → result is 280.

As shown, the “Stoves + Microwaves” is executed last because its solve order is higher. If the relative solve orders were reversed, the steps would be

4. Stoves is added to Microwaves for row 1993 → result is 80.
5. 100 is added to the result of this last calculation → result is 180.

To specify the solve order for a calculation, the solveOrder attribute needs to be set for the crosstabNodeMember object in the layout specification. It looks as follows...

```xml
<crosstabNodeMember class="cl" refDataItem="Line (Custom Calculation Item)" solveOrder="2"/>
```

If the solveOrders of two calculations are the same, then rows will take precedence over column, thereby causing the row calc to have a higher solve order than the column calc.

### 4.4 Performance

- How to avoid slow-running subtotals
  - Avoid placing aggregation subtotals on the inner level of a nested crosstab. Performance is slowed significantly due to the necessity of re-executing the aggregation for each permutation of the evaluation context. Performance is adversely affected against multidimensional data sources. It is worse against relational data sources.
  - Try to avoid running aggregate calculations against an unknown number of members. If there is a large flat dimension, then aggregating most of the members of a level would be very time consuming.
o Redundancy in dataItem expressions
  o If the same expression is repeated several times in one or more dataItem expressions, it improves performance greatly to make a single dataItem with that expression, and reference the single dataItem from the various other expressions.

o How to limit the amount of data shown
  o Design crosstabs or reports that have preset limits, or governors, that limit the amount of members returned from the query. Using functions such as HEAD, TopCount, BottomCount, and FILTER reduce the amount of data returned, as well as improving readability.

o Relational data sources
  o Construct well partitioned dimensions.
  o When building the model over a relational data source, use a star schema as opposed to a snowflake or standard relational setup.
  o Use inner joins in model as opposed to outer joins.
  o Use Materialized Views in the database as much as possible.

5 Summary

There are many methods of extracting useful data from a database to show in a report. OLAP technology and multidimensional modeling is one method that allows a report author to use powerful tools to help them retrieve the data that they are interested in. With experience, the reports that are authored with Report Studio can become even better by taking advantage of the techniques that IBM Cognos 8 provides.
Appendices

6.1 Appendix A - Multidimensional Models

6.1.1 What is a member?
Before describing multidimensional models, it is important to understand what a member is. A member is an identifiable object that has information about it stored in a database. It is analogous to a key or index in a relational table. When certain fact data for a row value in a relational table is aggregated, that row value becomes a member. For instance, if we have a table that has a series of Order Amount values for the month 1997/Jan, we can aggregate all the Order Amounts. The month 1997/Jan becomes a member with the aggregated value of Order Amount being the measure value for that member.

6.1.2 Dimensions and facts
A multidimensional model is a way of exposing a certain view onto a data source that organizes it into facts and dimensions. Numeric values corresponding to a fact table are referred to as measures in a multidimensional model. The areas that a business is interested in are represented by dimensions. For example, there can be a dimension for employees, products that a company builds, regions that a company operates in, etc.

A dimension is further broken down into hierarchies that act as differently organized views on a single dimension. Each hierarchy is organized into a tree of nodes, each of which is a summarization of its child nodes.

6.1.3 Hierarchies and levels
A hierarchy is also organized into levels. For instance, for a time dimension, the top level might be all the years that the data source has information for. All the members that represent individual instances of a year can be said to belong to the “Year” level. If each year in the hierarchy is further broken down into quarters, the each year could have up to 4 children in the tree, one for each quarter. Each node that represents a quarter can be said to belong to the “Quarters” level. Each child of a “Quarter” node could then belong to a “Month” level, and so on.

The following diagram helps illustrate the concepts…
In this example, the root “All Time” folder represents a dimension. Inside the “All Time” dimension, there are several hierarchies represented by the following icon. These hierarchies include All Time, Current Month, Last Month, etc., each of which are different ways of organizing information from the All Time dimension. The All Time hierarchy has one root member with the same name as the hierarchy, and it has three child nodes (or members); 2000, 2001, and 2002. All these members belong to a single level (probably called Years), as denoted by the outline in the diagram above. Each of the members of the Years level has child nodes as well, each of which belong to the Quarters level, and so on.

The objects in a multidimensional model that can be identified for use in an expression are dimensions, hierarchies, measures, levels, and members. This type of model is also referred to as a cube.

While a level is an identifiable object in a multidimensional model, it is not absolutely necessary for a model to contain them. This type of hierarchy is referred to as a parent-child hierarchy. The type of model that contains such hierarchies is sometimes called a Contributor-style cube.
6.1.4 Set Expressions

A set expression (as referred to throughout this document) is an expression using V5 syntax that generates a result set of members. The typical example as seen in the various parts of the document is “children of <member>”. This expression will take a specific member (e.g. “1998”) and return all its child nodes in the hierarchy that 1998 belongs to. These result members would likely be the quarters that 1998 is broken down into (1998Q1, 1998Q2, etc), but it all depends on how the hierarchy is modeled. There are many examples of set expressions, the most basic of which is “set”, which simply returns the members listed in the set expression. Some more examples…


descendants( 1998, Months ) ➔ result set is all members at level Month for 1998; 1998/Jan. 1998/Feb, etc…

parent( 1998Q1 ) ➔ result set is “1998”
6.2 Appendix B – Advanced Ranking Examples

Example 1 – Rank all items across all subgroups of a crossjoin.

This example shows how a rank calculation can be created that ranks all the values across every subgroup of a crossjoin. It makes use of the “nestedSet” function to accomplish this.

<table>
<thead>
<tr>
<th>Revenue</th>
<th>1993</th>
<th>1994</th>
<th>Rank (1993, included items in Line, all items in State)</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dishwashers</td>
<td>CA</td>
<td>285596</td>
<td>363251</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>NY</td>
<td>67120</td>
<td>61380</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>MA</td>
<td>62407</td>
<td>98403</td>
<td>7</td>
</tr>
<tr>
<td>Stoves</td>
<td>CA</td>
<td>352420</td>
<td>236212</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>NY</td>
<td>99487</td>
<td>100870</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>MA</td>
<td>98040</td>
<td>67071</td>
<td>5</td>
</tr>
<tr>
<td>Microwaves</td>
<td>CA</td>
<td>170167</td>
<td>248961</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>NY</td>
<td>50344</td>
<td>69348</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>MA</td>
<td>49274</td>
<td>27651</td>
<td>9</td>
</tr>
</tbody>
</table>

The expression to generate this rank would look something like this.

\[
\text{rank(currentMeasure DESC TUPLE [1993] WITHIN SET nestedSet(children([Line]), children([State]))})
\]

Example 2 – Rank each repeated item of a crossjoin

This example shows how a rank calculation can rank each repeated member in the lower level of a crossjoin. In this case, each value for the row with CA is ranked against the other rows with CA for column 1994. Each row with NY is ranked against the other rows with NY, and so on.

<table>
<thead>
<tr>
<th>Revenue</th>
<th>1993</th>
<th>1994</th>
<th>Rank (1994, each repeated item in State)</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dishwashers</td>
<td>CA</td>
<td>285596</td>
<td>363251</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>NY</td>
<td>67120</td>
<td>61380</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>MA</td>
<td>62407</td>
<td>98403</td>
<td>1</td>
</tr>
<tr>
<td>Stoves</td>
<td>CA</td>
<td>352420</td>
<td>236212</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>NY</td>
<td>99487</td>
<td>100870</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>MA</td>
<td>98040</td>
<td>67071</td>
<td>2</td>
</tr>
<tr>
<td>Microwaves</td>
<td>CA</td>
<td>170167</td>
<td>248961</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>NY</td>
<td>50344</td>
<td>69348</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>MA</td>
<td>49274</td>
<td>27651</td>
<td>3</td>
</tr>
</tbody>
</table>
The expression for this rank would look like this.

\[ \text{rank}(\text{currentMeasure DESC} \ \text{TUPLE \ [1994]} \ \text{WITHIN SET} \ \text{children([Line])}) \]

Note that the set referenced in the WITHIN SET clause is the outer level set of the crossjoin.
6.3 Appendix C – Complex Axis Example

Example – A union between two expressions with a single expression crossjoin.
This example shows two expressions that are unioned, and then the entire union is subsequently crossjoined against another expression.

In this case, members from the Line dimension are unioned with members from the State dimension, and then they are both nested against members from the Market dimension. The crosstab would look as follows…

<table>
<thead>
<tr>
<th>Revenue</th>
<th>1993</th>
<th>1994</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dishwashers</td>
<td>Builders</td>
<td>56710</td>
<td>233298</td>
</tr>
<tr>
<td></td>
<td>Furniture</td>
<td>35845</td>
<td>40282</td>
</tr>
<tr>
<td></td>
<td>Department</td>
<td>243729</td>
<td>232132</td>
</tr>
<tr>
<td>Stoves</td>
<td>Builders</td>
<td>63478</td>
<td>80227</td>
</tr>
<tr>
<td></td>
<td>Furniture</td>
<td>50890</td>
<td>27024</td>
</tr>
<tr>
<td></td>
<td>Department</td>
<td>349352</td>
<td>242774</td>
</tr>
<tr>
<td>Microwaves</td>
<td>Builders</td>
<td>71635</td>
<td>43330</td>
</tr>
<tr>
<td></td>
<td>Furniture</td>
<td>23746</td>
<td>33082</td>
</tr>
<tr>
<td></td>
<td>Department</td>
<td>148655</td>
<td>246840</td>
</tr>
<tr>
<td>CA</td>
<td>Builders</td>
<td>127295</td>
<td>265790</td>
</tr>
<tr>
<td></td>
<td>Furniture</td>
<td>87219</td>
<td>61255</td>
</tr>
<tr>
<td></td>
<td>Department</td>
<td>474880</td>
<td>450631</td>
</tr>
<tr>
<td>NY</td>
<td>Builders</td>
<td>23320</td>
<td>24876</td>
</tr>
<tr>
<td></td>
<td>Furniture</td>
<td>8550</td>
<td>21593</td>
</tr>
<tr>
<td></td>
<td>Department</td>
<td>137877</td>
<td>177366</td>
</tr>
<tr>
<td>MA</td>
<td>Builders</td>
<td>41208</td>
<td>66189</td>
</tr>
<tr>
<td></td>
<td>Furniture</td>
<td>14712</td>
<td>17540</td>
</tr>
<tr>
<td></td>
<td>Department</td>
<td>128979</td>
<td>93749</td>
</tr>
</tbody>
</table>

The layout specification would look as follows. The important thing to note in this example is that the Market expression needs to be repeated as a crosstabNestedNode for each outer unioned expression.

```
<cds crosstabRows>
  <crosstabNode>
    <crosstabNodeMembers>
      <crosstabNodeMember class="ml" refDataItem="Line (visible items set)">
        <contents>
          ...
        </contents>
      </crosstabNodeMember>
    </crosstabNodeMembers>
    <crosstabNestedNodes>
      <crosstabNode>
        <crosstabNodeMembers>
          <crosstabNodeMember class="ml" refDataItem="Market (visible items set)">
            <contents>
...<cds>
```
Advanced Multidimensional Reporting

```xml
... 
</crosstabNodeMember>
</crosstabNodeMembers>
</crosstabNode>
</crosstabNestedNodes>
</crosstabNode>
<crossstabNode>
  <crossstabNodeMembers>
    <crossstabNodeMember class="ml" refDataItem="State (visible items set)">
      <contents>
        ... 
      </contents>
    </crossstabNodeMember>
    <crossstabNodeMembers>
      <crossstabNodeMember class="ml" refDataItem="Market (visible items set)">
        <contents>
          ... 
        </contents>
      </crossstabNodeMember>
      <crossstabNodeMembers>
      </crossstabNode>
      </crossstabNestedNodes>
    </crossstabNode>
  </crossstabNodeMembers>
</crossstabNode>
</crossstabRows>
```