IBM Cognos Proven Practices: IBM Cognos BI – Improve Performance for Interactive Relational Data Access

Nature of Document: Guideline; Product(s): IBM Cognos 8 Framework Manager, IBM Cognos 8 Studios; Area of Interest: Modeling, Report Design

Business Analytics Proven Practices Team
April 16, 2013
(First published April 30, 2010)

Guidelines and techniques that can help to improve performance when authors and analysts deal interactively with relational data sources within the IBM Cognos BI studios (Query Studio, Analysis Studio, Report Studio Express).

View more content in this series

Introduction

Purpose
This document will provide some guidelines and techniques that can help to improve performance when authors and analysts deal interactively with relational data sources within the IBM Cognos BI studios.

Applicability
The guidelines and techniques in this document were validated using:

- IBM Cognos BI version 8.4.x, and version 10.1
- the GS_DB DB2 sample database

Exclusions
This document does not cover OLAP data sources.

Overview
In IBM Cognos BI, authors and analysts have the opportunity to interactively work with data sources in studios such as IBM Cognos BI Query Studio, IBM Cognos BI Analysis Studio, IBM Cognos BI Report Studio Express, and IBM Cognos 10’s new Business Insight Advanced studio
(which replaced Report Studio Express in IBM Cognos 10). These studios are referred to as interactive studios which allow the user to drag query items onto the report, one at a time or several at once, to dynamically generate a report. For each one of these gestures, IBM Cognos BI resubmits the query to the data source. When dealing with relational data sources, this may cause concerns about response times and loads on the data source.

For example, in Query Studio, it is quite common for authors to slowly build up their report by dragging on one item at a time. Consider the model below.

**Framework Manager diagram of a simple star schema**

![Framework Manager diagram of a simple star schema](image)

The diagram shows a simple star schema design with one fact table in the middle and its related dimensions, in this case, promotions, products, and time.

Now consider the contents of the package as they appear in Query Studio.
Query Studio UI with a blank report

Authors may slowly build up their ad hoc query by adding Year from the time dimension, which produces one query to the database.

Query Studio Report with Year column added

Second, the author may add promotion name from the promotions dimension.
Query Studio report with additional column added

At this point, two queries have been issued to the database and authors may notice a performance delay for the results. These two dimensions are joined through the sales fact table. Therefore the query may take longer than expected as it is also incorporating the fact table in the query to produce results. In cases where the fact table is large in volume, millions or even billions of rows, the wait time for what is perceived to be a simple query may take longer than expected.

Authors may then continue to build up their queries one item at a time, which produces a database query per addition.

This document will discuss some techniques that authors should be aware of when dealing with the data sources interactively, as well as provide some solutions which can be implemented by metadata modellers or report administrators to help provide better response times for potentially slow queries.

Filter First

One option for authors and analysts to improve interactive query response time is to filter their reports or analyses first. This can be accomplished by the user creating the filters themselves, or imposed on the authors through implicit filters created in the model.

In Query Studio, filters can be added prior to adding query items to the report. Simply right-click on the item you wish to base your filter on and select Filter for report.
Apply all the filters intended for the report first and then add the query items to the report. This can drastically reduce the records retrieved from the database once items are added to the report.

For the other studios, context filters can be applied before adding items to the report or analyses. This method applies to dimensional data sources only such as OLAP sources or dimensionally modelled relational (DMR) sources. DMR is the concept of adding dimensional information in a Framework Manager model to allow for OLAP-Style querying on the underlying relational source. Since this document is focused on relational sources, the following techniques imply DMR packages, but also apply to OLAP data sources.

In Analysis Studio, right-click on the item to be filtered and click **Filter as Context** (left screen capture below), or drag the item to the **Context filter** area (right screen capture below) of the analysis.

In Report Studio Express or Business Insight Advanced, items can be dragged to the **Context filter** area to achieve the same results. The screen capture below is of Report Studio Express.
While Report Studio Express only supports dimensional data sources, Business Insight Advanced supports both dimensional and relational sources. Therefore, with respect to Business Insight Advanced, this technique only applies to DMR packages where relational data sources are concerned.

**Select Multiple Items at One Time**
The more items that are added to a report or analysis at the same time will reduce the amount of queries sent to the data source.

For list reports in **Query Studio**, all items required for the report can be multi-selected and dragged to the report at the same time.

If one is not sure of all the items required for the report, adding more initially and then removing items from the report is recommended as queries are cached. The cached results can be reused, which will reduce the response time as the report is edited. Adding new items, however, will require a new query to the data source.
For **Analysis Studio** and **Report Studio Express**, multiple items can be added to the rows or columns at once, which will help to reduce queries to the data source. However, unlike list reports, all report items cannot be added at once since rows, columns and measures have their own drop zones.

**Analysis Studio showing default query drop zones**

![Analysis Studio showing default query drop zones](image)

In **Business Insight Advanced**, you have the option of creating lists, crosstabs, and charts. The type of report object you are working with will determine the scope of items you can add to the report object at once. For charts, data is only retrieved after a measure has been added to the report.

**Design with No Data or Limited Data**

Depending on the interactive studio, users can author without data or limited data. Limited data in Query Studio is made possible by metadata modellers implementing Design Filters, which limits the rows retrieved from the data source based on the filters. Design Filters are covered in the next section.

Designing reports without data allows authors to add all the items required for the report before submitting a query to the data source. Just like adding all the items at once in the previous section, this can drastically reduce queries to the data source while allowing authors the freedom to add items individually without worrying about wait times between each addition. Once satisfied with the design of the report, authors can view the report with all the data.

In **Query Studio** the options are found in the **Run Report** menu.
Query Studio UI with Preview with No Data option turned on

The report can be previewed with no data or limited data. Again, limited data will be data based on Design Filters implemented by the metadata modeller in Framework Manager. Once the design of the report satisfies the requirements, the report can be run with all data.

When previewing with limited data, authors will notice blank summaries. The screen capture below illustrates this behavior.

Query Studio with Preview with Limited Data turned on

This is done to ensure users don't mistakenly view these totals as true business figures. The figures are based on a reduced record set constrained by model filters beyond the user's control.

Note: The default behavior for working interactively with the data source can be changed for Query Studio. Preview with no data or with limited data can be set as the default behavior by following the steps below.

1. Stop the IBM Cognos BI server.
2. On the IBM Cognos BI server, navigate to `<IBM Cognos BI Install Directory>/templates/ps/async`.
3. Copy the existing `system.xml.sample` file to `system.xml`.
   The following line defines the default behavior:
   ```xml
   <param name="limitedDataMode">nodata</param>
   ``
   When set to nodata, the default behavior will be to preview with no data. If set to partial, as shown below, the default behavior will be to use the Design Filters.
   ```xml
   <param name="limitedDataMode">partial</param>
   ``
4. Modify the entry as required, save the file and restart the server.
To revert to the original behavior, stop the server, remove or rename the system.xml file, and then restart the server.

In **Report Studio Express** and **Business Insight Advanced**, authors can work in design mode under the **View** menu by selecting **Page Design**.

**Note:** The default behavior for working interactively with the data source can be changed for Report Studio Express and Business Insight Advanced. Preview mode or design mode can be set as the default behavior by following the steps below.

**Report Studio Express:**

1. On the IBM Cognos BI server, navigate to `<IBM Cognos BI Install Directory>\webcontent\pat\profiles`.
2. Open the `profile_express.xml` file in an editor and change the following XML element from:
   `<defaultPageView view="PagePreview"/>`
   to
   `<defaultPageView view="PageDesign"/>`
3. Save the file.

**Business Insight Advanced:**

1. On the IBM Cognos BI server, navigate to `<IBM Cognos BI Install Directory>\webcontent\pat\profiles`.
2. Open the `Profile_bua_standalone.xml` file in an editor and change the following XML element from:
   `<setting name="StartPageView" defaultValue="PagePreview"/>`
   to
   `<setting name="StartPageView" defaultValue="PageDesign"/>`
3. Save the file.

In **Analysis Studio**, authors can choose to get fact data later by selecting **Get Data Later** under the **Settings** menu.

**Analysis Studio showing the Get Data Later option**
Dimensional members are still displayed on the edges, in this case product lines and years, but the fact data is not retrieved. To retrieve the fact data, simply click the Get data link in the fact area of the analysis, or uncheck the Get Data Later option under the Settings menu.

This option, however, still retrieves the dimension members on the edges through joins to a common fact table. If performance is still not acceptable using the Get Data Later feature, then consider the technique described in section 8, Create Fake Fact Query Subjects.

**Prompts, Design Filters, and Common Calculations**

Modellers can implement different types of prompt filters to guide authors towards more efficient queries as well as create Design Filters to allow authors to work with a smaller subset of data when designing their reports. Both types of filters can be useful with interactive data access. Calculations can also be added to the model to reduce gestures in the interactive studios.

**Prompts**

Modellers can implement prompts by using a simple prompt syntax such as ?Prompt Name?, or by using prompt macros. Prompt macros provide a greater degree of control over the prompts but are beyond the scope of this document. Please refer to the IBM Cognos BI Framework Manager User Guide for detailed information about prompt macros.

One example of implementing a prompt in a model is shown below in which the time dimension query subject has a filter added with prompt syntax.

**Framework Manager showing a filter expression**

Users will be required to filter their report on one or more years.

To add a filter:

1. In the Project Viewer, double-click on the query subject to open its Definition dialog.
2. Click on the Filters tab, and then click on the Add link in the bottom right corner.
3. Enter the desired filter expression and then click OK.

The filter appears as shown below. If the ellipsis under Usage is clicked (also shown below), there are three options to choose from. The default is Always in which the filter will always be applied. Optional allows users to bypass the prompt, and Design Mode Only affects testing in Framework Manager and affects the rows returned when previewing limited data in Query Studio. It also has
an effect on reports run in Report Studio should they be set to run with limited data, but that is beyond the scope of this document as it does not relate to interactive data access.

**Framework Manager showing filter Usage options on the filter tab of a query subject definition dialog**

![Filter Usage Options](image)

In Query Studio, for example, the prompt appears as soon as an item from the time dimension is added to the report.

**Query Studio showing an IBM Cognos BI generated prompt for Year**

![Prompt](image)

Authors are automatically required to focus their reports by providing one or more years of data rather than returning all by default.

**Design Filters**

In Framework Manager, modellers can also apply **Design Filters**. For example, the promotions dimension can be limited to a handful of dimensions by filtering on the PROMOTION_KEY field.

**Framework Manager showing a filter expression for Promotion Key**

![Filter Expression](image)
The usage property of the filter will be set to **Design Mode Only**.

**Framework Manager showing a filter's Usage property set to Design Mode Only**

Now, if Preview with Limited Data is enabled in Query Studio, as seen below, authors can begin adding items to the report with faster response times since the data is filtered.

**Query Studio with Preview with Limited Data option turned on**

Promotions in the report are filtered by the Design Filter applied in the model.

**Query Studio report showing limited data based on Design filters**

If **Preview All Data** is enabled, then all promotions types are returned in the report.
Query Studio showing report returning all data after Preview with Limited Data option is turned off

<table>
<thead>
<tr>
<th>YEAR</th>
<th>PROMOTION_NAME_EN</th>
<th>SALE_TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>Centennial 2007 Promotion</td>
<td>798,359.56</td>
</tr>
<tr>
<td>2007</td>
<td>Clinber 2007 Bundle</td>
<td>1,366,609.84</td>
</tr>
<tr>
<td>2007</td>
<td>Compact Relief 2007 Promotion</td>
<td>54,901.02</td>
</tr>
<tr>
<td>2007</td>
<td>Cooler 2007 Promotion</td>
<td>2,625,827.02</td>
</tr>
<tr>
<td>2007</td>
<td>Course Pro 2007 Bundle</td>
<td>548,963.85</td>
</tr>
<tr>
<td>2007</td>
<td>Deluxe Relief 2007 Promotion</td>
<td>79,702.50</td>
</tr>
<tr>
<td>2007</td>
<td>EverGlow Lamp 2007 Promotion</td>
<td>2,205,857.02</td>
</tr>
<tr>
<td>2007</td>
<td>Everglow Single 2007 Promotion</td>
<td>1,836,990.96</td>
</tr>
<tr>
<td>2007</td>
<td>Extreme 2007 Bundle</td>
<td>3,760,058.24</td>
</tr>
<tr>
<td>2007</td>
<td>Firefly 2007 Bundle</td>
<td>1,029,234.35</td>
</tr>
<tr>
<td>2007</td>
<td>Granita Mist 2007 Promotion</td>
<td>715,401.31</td>
</tr>
<tr>
<td>2007</td>
<td>Hallstorm 2007 Bundle</td>
<td>6,915,389.02</td>
</tr>
<tr>
<td>2007</td>
<td>Hibernator 2007 Bundle</td>
<td>4,921,918.02</td>
</tr>
<tr>
<td>2007</td>
<td>In the Rain 2007 Bundle</td>
<td>6,721,516.63</td>
</tr>
</tbody>
</table>

Tips for applying Design Filters

Some dimension tables can be quite large and are good candidates for Design Filters. However, it is typically fact tables that are the largest and will benefit from Design Filters. The fact tables can be filtered on one or more of their foreign keys.

If Design Filters are applied to both a fact query subject and any relating dimension query subjects, ensure that the filters are the same for both the fact and the related dimension. If not, no data will be returned in the reports when previewing with limited data.

Recall the PROMOTION_KEY example shown earlier for the promotions dimension. Since the sales fact table is quite large, the same filter should be applied there.

Framework Manager showing a filter expression on the fact query subject

The same keys must be used otherwise there will be no record matches between the two underlying tables when they are filtered and no data will be returned.

Common Calculations

Modellers can add commonly used calculations to the model to help authors be more productive while at the same time reducing the amount queries sent to the database. Adding the calculations to the model can reduce the number of gestures an author needs to do when building up a calculation in the interactive studios.

For example, in Query Studio, an author may want to multiply column A by column B. Each of these columns needs to be added to the report before that calculation can take place which has the potential to send three separate queries to the database, one query for column A, one for
column B, and one for the calculation of column A multiplied by column B. A model calculation can reduce this to one query.

**Provide Starting Point Queries**

Report administrators may take the approach of providing commonly used starting point reports or analyses for the interactive studios. These starting point reports/analyses can be appropriately filtered to ensure acceptable response times. This alone, or in conjunction with prompts and Design Filters, can drastically improve wait times as authors design their reports.

**Create Fake Fact Query Subjects**

As mentioned earlier, in some cases authors may build up their reports in an interactive studio one item at a time. They may add items from two or more dimensions prior to adding any items from a common fact table between the dimensions.

**Framework Manager diagram showing a simple star schema**

For example, they may add Year from the time dimension and Promotion Name from the promotion dimension. The result is that the two dimensions will be joined through the sales fact table. If the fact table is large in volume, then the response time may be longer than expected. Authors may find waiting for fact data from a large table acceptable, but may not understand why a simple query from two dimension tables does not return immediately.

Another side effect to this scenario is that any form of materialization (pre-calculated aggregates) in the database may be ignored since no fact is included in the query. If the query was written by adding a dimension followed by a fact and then another dimension, then the database optimization would include the materialization as necessary.
Beyond some of the guidelines provided earlier in this document, metadata modelers can employ a technique in which they can implement a fake fact table that can be an alternate join for the dimensions of a large fact table. The fake fact table will be used in the joins instead of the actual fact table, thereby improving response times when querying solely from the dimensions.

The net result of the fake fact query subject will be to create a Cartesian Product for the dimensions, which returns in a shorter period of time. While the results may not be what is expected from a data perspective, this technique allows authors to work quickly with some form of result. The results become meaningful once items from the fact table are added to the report. If this is acceptable to users and the other techniques mentioned in this document are not viable options or need to be augmented, then this technique may be considered.

Like any technique and implementation, it must be thoroughly tested and fine tuned until it meets the requirements.

To implement a fake fact query subject, either create a one row table in the database or use a virtual table such as SYSDUMMY1 for DB2 or Dual for Oracle. In this example SYSDUMMY1 will be used.

1. In Framework Manager, create a data source query subject and provide it a name that comes first alphabetically over the true fact table it is meant to replace in the dimension to dimension queries. The reason for this is that the join path will use the fact query subject that comes first alphabetically. The alphabetically first relationship path is determined based on the ASCII codes of the query subject names where all lower case letters go after capital letters. In this example the name "A Fake Sales Fact" will be used with the following expression:

   **Framework Manager showing the SQL for a fake sales fact data source query subject**

   ![Image of SQL query](image)

   Select 1 as Key from SYSIBM.SYSDUMMY1 returns one row with a value of 1.
Framework Manager showing query results on the Test tab of a query subject definition dialog which shows 1 row with a value of 1

Sample syntax for Oracle would be `Select 1 as key from dual`. Sample syntax for SQL Server would be `Select 1 as 'Key'`. Each vendor will have slightly different syntax.
For some vendors, such as with Oracle and SQL Server, the SQL Type option will need to be changed to `Native`. This option is found by selecting the Test tab in the query subject definition, clicking on Option in the bottom right corner, clicking the SQL Settings tab, and then selecting `Native` from the SQL Type drop down list.

Framework Manager showing the SQL Setting tab of the Options dialog for a data source query subject

2. Next, create joins from the fake fact query subject to the required dimensions. Initially create a join from the dimension's primary key, to the fake fact key.
3. Click the ellipsis for the relationship expression and add the following logic to the relationship definition.

\[ 1=1 \]

or

\[ \text{[DB2 GOSALESDW].[GO_TIME_DIM].[DAY_KEY]}=\text{[DB2GOSALESDW].[A Fake Sales Fact].[Key]} \]

4. Click **OK**.

Notice the change in the relationship dialog.
Framework Manager showing the Relationship Definition dialog after the relationship has been edited

The link line is no longer there between the keys since the relationship is now a complex expression that goes beyond linking items from one query subject to another. The 1=1 one logic is in the expression to ensure the join will always be true and allows for the Cartesian Product result set. Joining the keys will never be true in this case, but is required since IBM Cognos BI requires at least one item per query subject to create a valid join. Create this type of join between the fake fact and each required dimension. In this example, the results appear as shown below.
"Framework Manager diagram showing the addition of the fake fact table query subject and its relationships to the related dimension query subjects

Test Fake Facts

Before publishing the package to IBM Cognos BI, ensure that the fake fact is included in the package as a hidden object. This way it will be available for the joins, but not visible to users, which will avoid confusion and clutter.
Framework Manager Package Definition dialog showing the inclusion of the new fake fact query subject as a hidden item

Publish the package to IBM Cognos BI and open it in Query Studio. In this case an item from the time dimension and the promotion dimension will be added to the report.

Query Studio showing a report with Year and Promotion Name added to it illustrating the Cartesian Product results

A Cartesian Product is returned. These results may not be meaningful to the author until a measure from the fact is added as it is simply a combination of each item in one table with every item in the other table. However, the response time will allow authors to work more quickly.

Once a fact is added, the data provides expected results.
Query Studio report showing a fact added to it

![Query Studio report showing a fact added to it](image)

Data may seem to disappear at this point since the joins through the fact table is likely more sparse in its combinations than the Cartesian Product results. This may confuse some authors and would require some education.

Fake Fact Considerations

If the Framework Manager model does not apply a form of filtering on the dimension query subjects, large Cartesian joins may be attempted. This operation may potentially fail or cause significant overhead at either the database level, or for IBM Cognos BI operations at run time.

The following are cases where this may occur.

- Large dimension tables can produce resource intensive Cartesian joins. For example 10,000 rows in one dimension multiplied by 10,000 rows in another would produce a Cartesian join of 100,000,000 rows.
- Combining three or more dimensions in a query without fact data, can also cause large Cartesian joins. Even if the individual dimensions themselves are relatively small, the Cartesian join grows exponentially with each addition of a dimension. For example, if dimension 1 has 500 rows, dimension 2 has 600 rows and dimension 3 has 1,000 rows, the Cartesian Product would equal 300,000,000 rows.
- In certain scenarios, IBM Cognos BI builds dynamic cubes that are used as a source for reports. Another potential side effect with a Cartesian join is that the amount of data that is transferred to the dynamic cube may cause a build governor to throw an exception (too much memory used) or exceed cube limits (number of members, members per level, etc.) that may not normally occur when fact data is included since it may be sparser in nature.

In either case, consider adding additional logic in the join expressions to the fake fact table that will reduce the range of join key values that can be returned from the dimension tables.

For example, in the relationship expression between the time dimension and the fake fact, a filter can be applied to reduce the records from the time dimension to just a few days.

Framework Manager showing a relationship Expression definition with additional logic added to it to reduce the record set even further

![Framework Manager showing a relationship Expression definition with additional logic added to it to reduce the record set even further](image)
The filter can be tailored to whatever makes the most sense for the business requirement. Perhaps the filter is for the current year, current quarter or current month. Filtering can be applied to any of the related dimensions relationships where necessary to improve performance.

This type of reduced result set can be compared to viewing reports with limited data. The difference in this case is that the filtering is done automatically for the author until they add fact data to the report.

Using the sample model in this document, adding items in Query Studio from all three dimensions returns a much reduced record set.

**Query Studio showing a report with a much reduced Cartesian product based on additional relationship logic**

The Cartesian join is reduced by the filters found in the join expressions from each dimension to the fake fact.

As soon as fact data from the sales fact query subject is added to the report, the join relationship goes through the sales fact table rather than the fake fact, and the data returns as expected.

**Query Studio showing report results once a fact is added to it**
Alternative to the Fake Fact Method
As an alternative to the fake fact method implemented in the model, and providing your DBA does not mind, you could request that a factless fact table be created in the database that only contains foreign key combinations for all dimensions you wish to relate together in the absence of going through a fact table.

The key combinations included in this table would be ones that make sense for dimension data returned in a report when not using actual fact data.

Unlike the fake fact method, you will import this factless fact table into the model rather than create it. However, like the fake fact method, you will ensure the query subject is named so that it comes alphabetically before the actual fact table and that it is related to all required dimensions. You will also include the new query subject in the package and configure it as a hidden object.

CQEConfig.xml - Disable Going Through Fact Query Subjects
IBM Cognos BI can be configured to not go through fact query subjects in dimension only queries. This configuration is done in the CQEConfig.xml file located in the <IBM Cognos BI install location>/configuration folder. By default, the file is only a sample file called CQEConfig.xml.sample. The configurations in this file are not picked up when IBM Cognos BI is started until the file is renamed to CQEConfig.xml.

To globally configure IBM Cognos BI to not go through a fact query subject for dimension only queries, follow these steps on each IBM Cognos BI report server in your environment.

1. In the <IBM Cognos BI install location>/configuration folder, copy CQEConfig.xml.sample to CQEConfig.xml.
2. Add the following Transformations section element under the <component name="CQE"> element as shown below.

   ```
   <component name="CQE">
     <section name="Transformations">
       <!--Default: "useAlphabeticalFirstFact"-->
       <entry name="dimOnlyQuery" value="useNoFacts" />
     </section>
   </component>
   ```

   The commented section (between the <!-- --> tags) provides the default value setting, useAlphabeticalFirstFact, for the dimOnlyQuery entry. This setting tells IBM Cognos BI to use the alphabetically first common fact table between the dimensions in a dimension only query. The useNoFacts value tells IBM Cognos BI to not go through a common fact table for dimension only queries.

3. Save the file and restart the IBM Cognos BI service.

useNoFacts Example
Before configuring the CQEConfig.xml file to useNoFacts, the SQL of a dimension only query that selects Year from the Time dimension query subject and Order method from the Order method query subject appears as shown below.

Cognos SQL
select distinct 
  GO_TIME_DIM.CURRENT_YEAR as Year1,
  SLS_ORDER_METHOD_DIM.ORDER_METHOD_EN as Order_method 
from 
great_outdoors_warehouse..GOSALESDW.GO_TIME_DIM GO_TIME_DIM,
great_outdoors_warehouse..GOSALESDW.SLS_ORDER_METHOD_DIM SLS_ORDER_METHOD_DIM,
great_outdoors_warehouse..GOSALESDW.SLS_SALES_FACT SLS_SALES_FACT 
where 
(SLS_SALES_FACT.ORDER_DAY_KEY = GO_TIME_DIM.DAY_KEY) and 
(SLS_SALES_FACT.ORDER_METHOD_KEY = SLS_ORDER_METHOD_DIM.ORDER_METHOD_KEY)

Native SQL

select distinct "GO_TIME_DIM"."CURRENT_YEAR" AS "Year1",
"SLS_ORDER_METHOD_DIM"."ORDER_METHOD_EN" AS
"Order_method"
from "GOSALESDW"."GO_TIME_DIM", "GOSALESDW"."SLS_ORDER_METHOD_DIM"
"SLS_SALES_FACT" "SLS_SALES_FACT"
where "SLS_SALES_FACT"."ORDER_DAY_KEY" = "GO_TIME_DIM"."DAY_KEY" and
"SLS_SALES_FACT"."ORDER_METHOD_KEY" = "SLS_ORDER_METHOD_DIM"."ORDER_METHOD_KEY"

In both the Cognos SQL and the Native SQL, Year and Order method are joined through the SLS_SALES_FACT table.

After configuring the CQEResultConfig.xml file to useNoFacts in the Transformations section, the SQL appears as shown below.

Cognos SQL

select D2.Year1 as Year1, 
  D3.Order_method as Order_method 
from 
(select 
  GO_TIME_DIM.CURRENT_YEAR as Year1,
  RSUM(1 at GO_TIME_DIM.CURRENT_YEAR
order by GO_TIME_DIM.CURRENT_YEAR asc local) as sc 
from 
  (select
    GO_TIME_DIM.CURRENT_YEAR as CURRENT_YEAR 
  from 
    great_outdoors_warehouse..GOSALESDW.GO_TIME_DIM GO_TIME_DIM
  group by
    GO_TIME_DIM.CURRENT_YEAR
  ) GO_TIME_DIM
group by
  GO_TIME_DIM.CURRENT_YEAR
order by
  Year1 asc
) D2 
full outer join 
(select
  SLS_ORDER_METHOD_DIM.ORDER_METHOD_EN as Order_method,
  RSUM(1 at SLS_ORDER_METHOD_DIM.ORDER_METHOD_EN order by
SLS_ORDER_METHOD_DIM.ORDER_METHOD_EN asc local) as sc 
from 
  great_outdoors_warehouse..GOSALESDW.SLS_ORDER_METHOD_DIM SLS_ORDER_METHOD_DIM
group by
  SLS_ORDER_METHOD_DIM.ORDER_METHOD_EN
order by
  Order_method asc
Native SQL

```
select "GO_TIME_DIM"."CURRENT_YEAR" AS "CURRENT_YEAR"
from "GOSALES Dw"."GO_TIME_DIM" "GO_TIME_DIM"
group by "GO_TIME_DIM"."CURRENT_YEAR"
order by 1 asc

select "SLS_ORDER_METHOD_DIM"."ORDER_METHOD_EN" AS "C0"
from "GOSALES Dw"."SLS_ORDER_METHOD_DIM" "SLS_ORDER_METHOD_DIM"
```

You will notice in the Cognos SQL that there are two sub queries, one for Year and one for Order method. There is also a full outer join between these two sub queries based on a column called sc. The sc column is an alias for a column that is used to stitch the two sub queries together. In other words, it is a stitch column (sc). This allows the complete record set to be retrieved for both Year and Order method and then stitched together on the sc column which is simply a running count (1, 2, 3, 4, etc.). If one record set has more rows than the other, data in that column will continue while nulls will be displayed for the other as seen below.

**List report displaying full outer join results**

<table>
<thead>
<tr>
<th>Year</th>
<th>Order method</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>E-mail</td>
</tr>
<tr>
<td>2005</td>
<td>Fax</td>
</tr>
<tr>
<td>2006</td>
<td>Mail</td>
</tr>
<tr>
<td>2007</td>
<td>Sales visit</td>
</tr>
<tr>
<td></td>
<td>Special</td>
</tr>
<tr>
<td></td>
<td>Telephone</td>
</tr>
<tr>
<td></td>
<td>Web</td>
</tr>
</tbody>
</table>

This stitching is done locally in this case and can be determined by examining the Native SQL. The Native SQL is simply two separate select statements, one for Year (CURRENT_YEAR in the actual table) and one for Order method (ORDER_METHOD_EN in the actual table).

While this global configuration may avoid going through fairly large fact tables and improve performance in many cases, you may encounter scenarios where the dimension tables are also fairly large. In these cases, keep in mind that the local processing of a full outer join may also impact performance. Again, some form of filtering on the dimension tables (discussed earlier in this document) can help alleviate performance concerns.

**Conclusion**

For users experiencing slow response times in the IBM Cognos BI interactive studios with relational data sources, using any or a combination of the techniques and guidelines described in this document may improve the end user experience. As with any implementation, the requirements should be scoped before deciding on a strategy, followed by thorough testing. Testing should be conducted in all studios that apply to the environment.