IBM Business Analytics Proven Practices: IBM Cognos Cube Designer - Model Aggregate Cubes Workshop

Product(s): IBM Cognos Cube Designer; Area of Interest: Modeling

Introduction

Purpose of Document
This document will explain how to prepare and model aggregate tables in IBM Cognos Cube Designer as well as show step-by-step instructions on the technique using the samples shipped with the product.

Applicability
This document was written against IBM Cognos BI 10.2.1 (FP1).

Exclusions and Exceptions
This document will not discuss or illustrate in-memory aggregates.

Assumptions
This document assumes knowledge of modeling in IBM Cognos Cube Designer.

IBM Cognos Dynamic Cubes Overview
As part of the IBM Cognos BI platform, IBM Cognos Dynamic Cubes is an intelligent, in-memory relational OLAP (ROLAP) technology that enables IT to model and deploy cubes with a modern architecture to maximize query performance over terabytes of data.

IBM Cognos Dynamic Cubes can leverage existing aggregates or can be configured to create new in-memory aggregates for improved performance. The technology can analyze query workload and cube model structure to make recommendations for improvement, which can be immediately submitted for implementation without re-authoring the cube. As data values are added into
memory, performance will continue to increase over time. This high performance environment enables end users to quickly analyze and report on volumes of data for faster decisions.

Using IBM Cognos Cube Designer, IT can easily model and deploy cubes and even combine cubes using the virtual cube technology to present consolidated data.

**IBM Cognos Cube Designer Overview**

IBM Cognos Dynamic Cubes uses IBM Cognos Cube Designer to import and model metadata. The data source that the metadata is imported from must be in a Star or Snowflake Schema format, which is an industry standard reporting database design as seen in Figure 1.

**Figure 1: Sales Fact table surrounded by related dimension tables Geo, Product, Time, and Customer**

The relationships defined in the data source are used to relate the dimensional information, such as time, products, geography, etc., to the measures in the fact table.

Each cube designed in IBM Cognos Cube Designer can only be based on one fact table and its related dimensions. To combine multiple fact tables, a virtual cube can be created that combines multiple cubes.

The model design is based on re-usable metadata. For example, a dimension is created, levels are added to the dimension, and then those levels are added to one or more hierarchies in that dimension. In this sense, the levels are re-usable.
Figure 2: Project Explorer tree showing levels defined for the Time dimension used in two different hierarchies

The same concept applies for dimensions. Dimensions can be added to one or more cubes.

Figure 3: Project Explorer tree showing the Product and Time dimensions used in a cube called GO Sales by Month

Security can be applied to the model and it can also be enhanced through the use of calculations.

Aggregate Awareness

You can also leverage aggregate tables in the database by modeling aggregate cubes within an existing cube. This is beneficial for initial measure queries when the values are not already stored in the in-memory data cache. The aggregate cube can be leveraged at the aggregate level, but also for any additive measures for any levels above the aggregate level.
When designing and creating your aggregate table in the database, it is recommended to add the keys that represent the levels the measures are rolled up to as well as all their parent keys. This makes for more efficient queries since the query only involves the one aggregate table at run time. If the parent keys are not included in the aggregate table, then extra modeling in Cube Designer is required to include the dimension tables, which in turn involves more tables in the aggregate queries at run time.

You can leverage Aggregate Advisor, found in IBM Cognos Dynamic Query Analyzer, which can recommend database aggregates that are structured in a way that makes for easy modeling in Cube Designer and efficient usage in Dynamic Cubes at run time.

During the modeling process, you provide information that tells IBM Cognos in which scenarios the aggregate tables should be used by mapping the appropriate measures and keys to be used in the generated SQL at run time and which cube dimension levels trigger the use of the aggregate cube. For example, as seen in Figure 4, the Editor pane shows the configuration of an aggregate cube and highlights which aggregate levels are selected for the Product dimension, in this case, Product line and Product type.

Figure 4: Cube Designer showing the Editor pane for an aggregate cube

You can model several aggregate cubes for various scenarios. You can even implement partitioned aggregate tables. For example, sales measures are partitioned by year. Using the Slicers configuration tab, you would indicate with the appropriate member from the Time dimension, which year applies to the partitioned aggregate table. You would create an aggregate cube for each year partition with the appropriate slicer member configuration.

Once the cube is designed, it can be published to IBM Cognos BI as a package and used as a data source for dimensional reporting.

Workshop Objectives

You have been given an overview of some of the key capabilities of Dynamic Cubes and Cube Designer. This workshop will focus on modeling a simple aggregate table into the cube to give you a better understanding of how it works and the best practices.
You will start with an existing model provided with this document found in the Download section at the bottom of the page. Download the zip file and extract the **Cube Designer Model.fmd** file contained within and place it in a location that you will access later in this workshop.

This model is based on the GOSALES DW samples database shipped with the product. You will augment it by creating an aggregate cube to leverage an existing aggregate table in the database to improve performance. The aggregate table has Quantity and Sale Total values rolled up to the Quarter level from the Time dimension, the Product Type level from the Products dimension, and the Order Method level for the Order Method dimension.

Once you have finished modeling, you will publish the cube and create test reports to test the aggregate cube you just created. To analyze the SQL generated at run time, you will use Dynamic Query Analyzer, which is a tool used to analyze and troubleshoot Dynamic Query Mode (DQM) reports as well as make aggregate recommendations for dynamic cubes. You can also view cube metrics about aggregate table usage in the IBM Cognos Administration console.

### Connect to an IBM Cognos BI Data Source

To complete this workshop, you will need to have installed and setup the IBM Cognos BI samples that ship with the product. In particular, you will need to have setup the GOSALES DW database and created a data source connection to the database in IBM Cognos BI called **great_outdoors_warehouse**. Please refer to the “Samples for IBM Cognos Business Intelligence” user guide on how to setup the samples.

1. In IBM Cognos Cube Designer, click **Open Existing**.

   **Figure 5: IBM Cognos Cube Designer welcome screen with cursor pointing to Open Existing option**

2. In the Open dialog, navigate to the location you saved the **Cube Designer Model.fmd** file provided with this document, and then click **Open**.

3. In the Source pane on the left, expand **gosalesdw > gosalesdw > tables**.
Figure 6: Source pane showing Tables folder expanded

This is a data source made available by the IBM Cognos BI administrator. Notice the presence of a table called AGGR_TIME_PROD_OM_FACT. This is an aggregate table created by the database administrator which contains rollups for quantity and sale total values at the quarter level for time, product type for products and order method type for order method.

4. Expand AGGR_TIME_PROD_OM_FACT.

Figure 7: Source pane showing AGGR_TIME_PROD_OM_FACT table expanded

Notice the structure of the table. It contains the aggregated measures, QUANTITY and SALE_TOTAL, as well as the keys for the levels the measures are aggregated to, QUARTER_KEY, PRODUCT_TYPE_KEY, and ORDER_METHOD_KEY. And beyond that, there are the parent keys for the hierarchies that have higher levels as seen with CURRENT_YEAR and PRODUCT_LINE_CODE. Both the time dimension and product dimension have levels above the aggregate level. By including these parent keys, IBM Cognos can further aggregate the measures to the higher levels by a single query to this one aggregate table. No other tables are required for the rollups since all the information required is present in this one table.
Automatically Create an Aggregate Cube

You will now use drag and drop functionality to automatically create an aggregate cube. When doing so, the aggregate cube that is generated should be examined to ensure items have been mapped correctly and meet your requirements. In some cases, Cube Designer may not be able to auto generate the cube and you will have to create it manually.

1. In the Project Explorer pane, expand **Model > GO Sales by Month – Dynamic Cube**.

   **Figure 8: Project Explorer pane showing GO Sales by Month – Dynamic Cube expanded**

Under GO Sales by Month – Dynamic Cube, you see a folder called Aggregate Cubes. Here is where you will define a cube subset within the larger cube, called an aggregate cube. This aggregate cube defines where and how to retrieve aggregate values from the database upon initial measure value queries. Once these values are retrieved, they are stored in memory in the data cache and leveraged where possible for future queries unless the data cache is cleared or the cube is restarted.

You will now use a simple drag and drop method to create an aggregate cube and explore the results. You will then recreate the same aggregate cube manually to understand each of the components used to build the aggregate cube.

2. Double-click **Aggregate Cubes**, and then from the Source pane, drag **AGGR_TIME_PROD_OM_FACT** to the **Aggregate** pane.
3. In the Aggregate pane, double-click **AGGR_TIME_PROD_OM_FACT**. The Editor pane for the new aggregate cube appears.

**Figure 10: Editor pane for the new aggregate cube**

Here is where all the dimensions to be used in the aggregate cube are configured. In this case the dimensions are Product, Time, and Order Method.

4. Under the Dimension column, click on **Product**. A Product hierarchy column appears to the right displaying the available levels, in this case Product line, Product type, and product.

**Figure 11: Editor pane showing the Product hierarchy column**

Notice that only the first two levels are selected as being in scope for the aggregate cube. This was automatically
configured when the table was dropped in the drop zone for the aggregate cube because the key names could be matched to the levels in the existing Product dimension in the model. If you click on the Time dimension, you will see that only the Year and Quarter levels are in scope, but not the Month level.

**Figure 12: Editor pane showing the Time hierarchy column**

5. Click the **Measures** button.
   The Measures pane appears showing Sale Total under the Measures column and SALE_TOTAL under the Mapping column as seen in Figure 13.

**Figure 13: Measures pane showing the measure configuration just implemented**

In this case, Sale Total was automatically configured. Quantity was not brought in. This is why it is recommended to fully review all the settings for the aggregate cube after an automatic configuration and correct any issues. You will now quickly add and map the Quantity measure.

6. Click on the **Add Measure** button on the right side of the Measures pane, click the checkbox left of the Quantity measure, and then click **OK**.
7. In the Source pane, from the AGGR_TIME_PROD_OM_FACT table, drag **QUANTITY** to the Mapping column beside the Quantity measure in the Measures pane.
Figure 14: Measures pane now showing Quantity under the Measures column and QUANTITY mapped from the aggregate table under the Mapping column

8. At the top of the pane, click the AGGR_TIME_PROD_OM_FACT link to navigate back to the previous screen, and then click on the Key Mappings button.

Figure 15: Aggregate pane with Key Mappings displayed

Here the keys from the aggregate table have been automatically mapped to the level keys defined in the model's Dimensions, in this case, Order Method, Time, and Product dimensions.

You will now delete this aggregate cube and then recreate it from scratch manually in order to learn the procedure thoroughly.

9. In the Project Explorer, expand Aggregate Cubes, select the AGGR_TIME_PROD_OM_FACT aggregate cube, and then press Delete on your keyboard.

Manually Create an Aggregate Cube

In this section you will manually create an aggregate cube to learn all the required pieces for a successful implementation.

1. In the Project Explorer, double-click Aggregate Cubes.
2. Click the New Aggregate button to the right of the Aggregate pane, and then in the New Aggregate Cube dialog, select Quantity and Sales Total under Measures and all dimensions.
3. Click **OK**.
   You will give the aggregate cube a more meaningful name.

4. In the Project Explorer, right-click the **GO Sales by Month – Dynamic Cube2** aggregate cube, click **Rename**, type **Aggr Time Prod OM Fact**, and then press **Enter**.

5. In the Aggregates pane, select **Aggr Time Prod OM Fact**, and then click on the **Edit** button to the right of the pane.

6. Under the Dimension column, select **Product**, and then to the right, select the **Product Type** level. This will automatically select the Product line level as well. Parent levels are automatically included.

   **Figure 17: The Editor pane showing both Product type and Product line levels selected for the Product dimension**

7. Select **Time**, and then select the **Quarter** level. Again, the Year level is automatically selected.
Figure 18: The Editor pane showing both Quarter and Year levels selected for the Time dimension

8. Click **Order Method**, and then select the **Order Method** level.

Figure 19: The Editor pane showing the Order Method level selected for the Order Method dimension

Next, you will map the keys from the aggregate table to the unique keys defined in the dimensions. This will let IBM Cognos BI know which keys will trigger the use of the aggregate table.

9. Click the **Key Mappings** button, and then from the Source pane, from the AGGR_TIME_PROD_OMP_FACT table, drag the following items to their target Mapping column in the aggregate cube.

- **PRODUCT_LINE_CODE** to **Product Line Code** from Product
- **PRODUCT_TYPE_KEY** to **Product Type Key** from Product
- **CURRENT_YEAR** to **Current Year** from Time
- **QUARTER_KEY** to **Quarter Key** from Time
- **ORDER_METHOD_KEY** to **Order Method Key** from Order Method
Figure 20: Key Mappings pane showing the keys mapped from the aggregate table in the database to the Level Unique Key items in the model’s dimensions

10. Click the Editor button, and then click the Measures button to the left of the Editor pane.

11. From the Source pane, from the AGGR_TIME_PROD_OM_FACT table, drag QUANTITY to the Quantity mapping column and SALE_TOTAL to the Sale Total mapping column.

Figure 21: Measures pane showing measures from the aggregate table in the database mapped to the appropriate measures defined in the model

Now, when either of these measures are used at a level defined in the aggregate cube or any of their parent levels, IBM Cognos BI will generate the appropriate SQL to pull the values from the aggregate table rather than the detail fact table modelled in the cube, in this case SLS_SALES_FACT.

12. Save the model.

Publish Cube and Create Test Reports

In this section you will publish the GO Sales by Month – Dynamic Cube which now contains an aggregate cube and then create test reports which you will use to see how the queries perform.

1. In the Project Explorer pane, right-click GO Sales by - Month Dynamic Cube, and then click Publish.

2. In the Publish dialog, click on Additional Options.

3. Select the Select all options check box. If you are using Anonymous in a test environment, you can deselect the Associate my account and signon with the cube datasource check box.
By default, the package is published to the root of Public Folders, but you can browse to another location if you wish.

4. Click **OK**, and then click **OK** in the Deploy Results dialog.

A data source and data store called GO Sales by Month – Dynamic Cube was just created in IBM Cognos BI and a package called GO Sales by Month – Dynamic Cube was published to the root of Public Folders.

5. Open a supported web browser, in the address bar, type the URL to your Cognos BI instance, for example **http://<server name>/ibmcognos**, press **Enter**, and then click **IBM Cognos Content**. You will now create three test reports.

6. Click the **GO Sales by Month - Dynamic Cube** package, and then from the Launch menu, click **Report Studio**.

7. Click **Create New**, click **Crosstab**, and then click **OK**.

8. In the Source pane, expand **Product > Product**, and then drag **Product line** to the Rows drop zone on the crosstab.

9. Expand **Time > Time**, and then drag **Year** to the Columns drop zone.

10. Expand **Measures**, and then drag **Sale Total** to the Measures drop zone.
11. Save the report and name it **Product Line and Year Aggregate Test**.
12. From the Source Pane, nest **Product type** to the right of Product line and nest **Quarter** below Year in the crosstab. A flashing cursor between two lines will appear when you are over the appropriate drop zone for the nesting action.

**Figure 24: Crosstab report showing Product type and Quarter nested under Product line and Year respectively**

13. From the File menu, click **Save As**, and then save the report as **Product Type and Quarter Aggregate Test**.
14. From the Source pane, nest **Month** below Quarter.
Figure 25: Crosstab report with Month nested below Quarter

This report does not match a pattern that would allow the aggregate table to be used, because at least one dimension in the report goes to a level lower than that found in the aggregate table.

15. Save this third report as Product Type and Month Aggregate Test.

Test Aggregate Table Using Dynamic Query Analyzer

In this section, you will use Dynamic Query Analyzer to run the three test reports you just created to determine if the aggregate table is being used as expected. The test is only valid on the first run of the report as the results will be cached in the cube for any subsequent runs. If you wish to retest, you would need to clear the cache.

You will have needed to configure Dynamic Query Analyzer to point to your IBM Cognos BI instance as well as create a virtual directory in your web server that points to the XQE logs location. Please refer to the Dynamic Query Analyzer Installation and Configuration Guide for details.

1. Press the Windows key on the keyboard, type Dynamic Query Analyzer, and then click on IBM Cognos Dynamic Query Analyzer under Programs.
2. Ensure you are in the Analyze Logs window. To check, under the Windows menu you should see the Aggregate Advisor option. If you do, you are in the correct view. If you do not, then you should see the Analyze Logs option. If this is the case, click Analyze Logs and the correct view for the next tasks will appear.
3. If necessary, from the File menu, click Open IBM Cognos Portal.
4. In the Content Store pane, expand GO Sales by Month – Dynamic Cube, and then double-click Product Line and Year Aggregate Test. The report runs and once finished, you are asked if you would like to open the newest log.
5. Click Yes.
   If Dynamic Query Analyzer is configured properly, all the nodes for the execution trace appear.
The XMdxSelect node indicates that there is a sub query for this part of the execution. This sub query is the SQL used to retrieve the measure values on this initial run of the report. Subsequent runs would use the cached values.

6. Right-click the XMdxSelect node, and then click Open sub queries.
   In the Query pane, you will see the SQL used to retrieve the measures.

**Figure 27: Query pane showing SQL used to retrieve measure values from the database**

Notice that the table used is the AGGR_TIME_PROD_OM_FACT table you modelled as the aggregate cube. The query is behaving as expected and using the more efficient query against the aggregate table and grouping on the PRODUCT_LINE_CODE and CURRENT_YEAR.

Also notice that the query is requesting both measures, SALES_TOTAL as well as QUANTITY. This is an optimization feature. Since we are already querying the database, IBM Cognos BI will retrieve the other measure as well to prevent another future query to the database. An internal algorithm determines when this behaviour is optimal.
7. Click on the **Content Store** tab, and then double-click the **Product Type and Quarter Aggregate Test** report.

8. Click **Yes** to view the newest log, right-click the **XMdxSelect** node, and then click **Open sub queries**.

   **Figure 28: Query pane showing SQL used to retrieve measure values from the database**

   ```sql
   SELECT "AGGR_TIME_PROD_OM_FACT"."PRODUCT_TYPE_KEY" AS "Product Type Key",
   "AGGR_TIME_PROD_OM_FACT"."QUARTER" AS "Quarter",
   SUM("AGGR_TIME_PROD_OM_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT") AS "SLS_SALES_FACT"
   GROUP BY "AGGR_TIME_PROD_OM_FACT"."PRODUCT_TYPE_KEY";
   ```

   Again, the **AGGR_TIME_PROD_OM_FACT** table is used to retrieve the measure values at the Product Type and Quarter levels because these still match the aggregate cube scope. You will now run the report that is out of scope for the aggregate cube.

9. Click on the **Content Store** tab, and then double-click the **Product Type and Month Aggregate Test** report.

10. Click **Yes** to view the newest log, right-click the **XMdxSelect** node, and then click **Open sub queries**.

    **Figure 29: Query pane showing SQL used to retrieve measure values from the database**

    ```sql
    SELECT "SLS_SALES_FACT"."SLS_SALES_FACT" AS "SLS_SALES_FACT",
    "SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"."SLS_SALES_FACT"
    ```

    Because the Month level is out of scope for the aggregate cube you defined, the SQL generated goes against the detailed fact table, in this case **SLS_SALES_FACT**.

Each of the three test reports behaved as expected. The first two reports, which were in scope of the aggregate cube, generated SQL to retrieve measures from the aggregate table.

The one report that was not in scope used the main implementation of the cube which was to use the detail fact table.

**View Aggregate Table Usage with Cube Metrics**

You can also view metrics on aggregate table use in the Admin console by going to IBM Cognos Administration > Data Stores, click on the cube name, click on the server group, and then click on the QueryService for the dispatcher. If you have cubes running on more than one dispatcher, you will have to go to each one individually to view the metrics for that cube.

You can also view cube metrics by going to IBM Cognos Administration > System, click on the server name, click on the dispatcher running the cube you are interested in, click on QueryService, and then click on the cube name.

In either location, you can see the metrics for the cube in the Metrics pane as seen in Figure 30.
Figure 30: IBM Cognos Administration Metrics pane for cube

![IBM Cognos Administration Metrics pane for cube](image)

Metrics to note include **Aggregate table hit rate**, **Aggregate table hit rate in last hour**, **In-memory aggregate cache hit rate**, and **In-memory aggregate cache hit rate in last hour**. Notice the Aggregate table hit rate is 66.67%. In our example this means 2 out of the 3 reports we ran generated queries to the database that were routed to the in-database aggregate table. Since this workshop does not use in-memory aggregates, the In-memory aggregate cache hit rate is 0%.

Appendix A – Model an Aggregate Dimension

There may be situations where you prefer to normalize your aggregate data to reduce data redundancy and storage space in your aggregate fact table.

To do that you can create aggregate dimension tables as well. These aggregate dimension tables would simply include the key for the aggregate level found in the aggregate fact table and all its parent keys.

Let’s use this document’s simple Product dimension example. The original product dimension table has the following levels, Product line, Product type, and Product. Instead of including the `PRODUCT_LINE_CODE` and `PRODUCT_TYPE_KEY` in the aggregate fact table as was seen in our example, you could simply have the aggregate fact table contain just the `PRODUCT_TYPE_KEY`, and then create an aggregate Product dimension that contains both the `PRODUCT_LINE_CODE` and `PRODUCT_TYPE_KEY`.

What this does is reduce the redundancy of the `PRODUCT_LINE_CODE` in the aggregate fact table. Rather than the code repeat for every aggregate measure in the aggregate fact table, it would only repeat once for every `PRODUCT_TYPE_KEY` instance that roles up to the `PRODUCT_LINE_CODE` in the aggregate dimension. In short, perform normalization as seen in Figure 31 where the `AGGR_PROD_DIM` table contains `PRODUCT_LINE_CODE` and `PRODUCT_TYPE_KEY` and relates to `AGGR_TIME_PROD_OM_FACT` on `PRODUCT_TYPE_KEY`. 
Figure 31: AGGR_PROD_DIM related to AGGR_TIME_PROD_OM_FACT on PRODUCT_TYPE_KEY

Note: The AGGR_PROD_DIM table is not part of the sample that ship with the product.

To model this type of aggregate dimension table in IBM Cognos Cube Designer, you need to map the keys in this table to the appropriate Unique Keys for the model dimension it represents and then create a relationship to the aggregate fact table on the appropriate key.

In our example, you would edit the Aggr Time Prod OM Fact aggregate cube, you would click on the **Key Mappings** button and then map the PRODUCT_LINE_CODE and PRODUCT_TYPE_KEY keys from the AGGR_PROD_DIM table to the Product Line Code and Product Type Key in the Level Unique Key column of the model’s Product dimension as shown in Figure 32.

Figure 32: AGGR_PROD_DIM keys mapped to the Product dimension

Next, you would click on the **Editor** button, and then click on the **Edit** link in the Relationship column for the Product dimension as seen in Figure 33.
You then create a relationship between AGGR_PROD_DIM and AGGR_TIME_PROD_OM_FACT on PRODUCT_TYPE_KEY as seen in Figure 34.

You would leave the **Join is at the lowest level of detail for the dimension** selected. In this case it refers to the key from the aggregate dimension table that is joined to the fact table. In this case it is PRODUCT_TYPE_KEY, which is actually unique and the lowest level in this table.

If you click on the **Implementation** button, you will see the two aggregate tables with a relationship between them with 1 to n cardinality from the dimension table to the fact table as seen in figure 35.
Figure 35: Implementation pane showing relationship between AGGR_PROD_DIM and AGGR_TIME_PROD_OM_FACT

You can now publish the cube and if you run a query that matches the pattern for this aggregate cube, for example, Year from Time, Product Line from Product and Sale Total from the measures, the SQL would query both aggregate tables.
Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cube Designer Model.fmd file</td>
<td>IBMCognosCubeDesigner-ModelAggregateCubes.zip</td>
<td>5KB</td>
</tr>
</tbody>
</table>

© Copyright IBM Corporation 2014

Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)