

IBM Cognos Dynamic Cubes
Version 11.1.0

User Guide



©

Product Information

This document applies to IBM Cognos Analytics version 11.1.0 and may also apply to subsequent releases.

Copyright

Licensed Materials - Property of IBM

© Copyright IBM Corp. 2012, 2021.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

- Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft product screen shot(s) used with permission from Microsoft.

Contents

- Introduction..... vii**

- Chapter 1. What's new.....1**
 - New features in 10.2.2 FP1..... 1
 - New features in 10.2.2.....1
 - New features in 10.2.1.1..... 5
 - New features in 10.2.1.....5

- Chapter 2. Cognos Dynamic Cubes overview.....7**

- Chapter 3. Cognos Dynamic Cubes workflow..... 9**

- Chapter 4. Dimensional metadata and dynamic cubes..... 15**
 - Dimensional metadata.....15
 - Dimensions..... 15
 - Hierarchies..... 15
 - Parent-child hierarchies..... 21
 - Levels..... 22
 - Joins.....23
 - Attributes..... 24
 - Dynamic cubes.....24
 - Measures..... 26
 - Regular aggregates.....27
 - Aggregation rules..... 28
 - Virtual cubes..... 30
 - Virtual cube scenarios..... 33
 - In-database aggregates..... 33

- Chapter 5. Getting started with Cognos Cube Designer..... 37**
 - Introduction to Cognos Cube Designer..... 37
 - Import metadata.....39
 - Importing metadata from a Content Manager data source..... 39
 - Importing metadata from a Framework Manager package..... 40
 - Import InfoSphere Warehouse Cubing Services cube metadata..... 41
 - Managing a project.....43
 - Validate a project and individual objects..... 44

- Chapter 6. Dimensional metadata modeling..... 45**
 - Model dimensions.....45
 - Defining a dimension.....45
 - Defining a dimension based on a relational table..... 46
 - Defining a shared member cache..... 47
 - Model hierarchies..... 48
 - Defining a hierarchy.....49
 - Model levels..... 49
 - Defining a level 51
 - Defining a level unique key..... 51
 - Defining the member sort order.....52
 - Model parent-child hierarchies..... 52

| | |
|--|------------|
| Defining a parent-child hierarchy..... | 54 |
| Browsing members..... | 55 |
| Dimension filters..... | 55 |
| Defining a dimension filter..... | 56 |
| Defining named sets..... | 56 |
| Parameter maps..... | 57 |
| Creating parameter maps manually..... | 58 |
| Creating parameter maps by importing entries..... | 58 |
| Creating parameter maps from existing query items..... | 58 |
| Chapter 7. Dynamic cube modeling..... | 61 |
| Creating an IBM Cognos Framework Manager project for a ROLAP model..... | 61 |
| Model a dynamic cube..... | 61 |
| Defining a dynamic cube based on a relational table..... | 62 |
| Defining a dynamic cube manually..... | 63 |
| Model measures..... | 63 |
| Defining a measure-to-dimension join..... | 67 |
| Measure dimension filters..... | 67 |
| Measure folders..... | 68 |
| Sort measures and folders..... | 68 |
| Deploying and publishing dynamic cubes..... | 69 |
| Creating and publishing packages..... | 70 |
| Publishing packages based on ROLAP data sources..... | 70 |
| Estimating hardware requirements..... | 70 |
| Chapter 8. Advanced dynamic cube modeling..... | 73 |
| Calculated members..... | 73 |
| Calculated member and measure examples..... | 74 |
| Defining a calculated member..... | 77 |
| Model relative time dimensions..... | 77 |
| Next period relative time members..... | 80 |
| Custom relative time members..... | 81 |
| Defining a relative time dimension..... | 86 |
| Examples of level current period expressions..... | 88 |
| Multiple locales..... | 89 |
| Selecting the design language and supported locales..... | 89 |
| Adding multiple locale names to metadata objects and dynamic cube objects..... | 90 |
| Adding support for multiple locales to members and attributes..... | 90 |
| Chapter 9. Aggregate modeling..... | 93 |
| Modeling in-database aggregates..... | 93 |
| Defining an in-database aggregate automatically..... | 94 |
| Defining an in-database aggregate manually..... | 95 |
| Defining an in-database aggregate with a parent-child dimension..... | 96 |
| Filtering data using an aggregate slicer..... | 97 |
| Creating user-defined in-memory aggregates..... | 97 |
| Chapter 10. Virtual cube modeling..... | 99 |
| Defining a virtual cube | 99 |
| Model virtual dimensions..... | 100 |
| Model virtual hierarchies..... | 101 |
| Viewing virtual levels..... | 102 |
| Model virtual members..... | 103 |
| Model virtual measures..... | 104 |
| Chapter 11. Define security..... | 107 |
| Security filters for hierarchy members..... | 107 |

| | |
|---|------------|
| Default members..... | 110 |
| Secure calculated members..... | 111 |
| Security filters based on a lookup table..... | 111 |
| Defining a role-based security filter..... | 113 |
| Security views..... | 113 |
| Tuple security..... | 114 |
| Defining a security view..... | 115 |
| Chapter 12. Cognos dynamic cubes administration..... | 117 |
| Access permissions and capabilities for dynamic cubes..... | 117 |
| Creating a Dynamic Cubes Developer role..... | 121 |
| Assigning data access accounts for dynamic cubes | 123 |
| Creating trusted credentials..... | 124 |
| Creating a signon..... | 124 |
| Configure dynamic cubes for the query service..... | 125 |
| Adding dynamic cubes to the query service..... | 126 |
| Starting and managing dynamic cubes..... | 127 |
| Setting query service properties for dynamic cubes..... | 129 |
| Starting and stopping the query service..... | 131 |
| Setting dynamic cube properties..... | 132 |
| Setting general properties for a dynamic cube | 137 |
| Creating and scheduling query service administration tasks..... | 138 |
| Setting access permissions for security views..... | 139 |
| Memory monitoring in the dynamic query mode server..... | 141 |
| Configuring the dynamic query mode server monitoring settings..... | 141 |
| Enabling IPF logging for Cognos Cube Designer..... | 144 |
| Chapter 13. Near real-time updates of dynamic cubes data | 147 |
| Enabling near real-time updates for dynamic cubes..... | 147 |
| Loading incremental updates to dynamic cubes..... | 148 |
| Incremental updates of aggregate tables | 150 |
| Pausing a dynamic cube to update aggregate tables | 151 |
| Chapter 14. Relational and DMR modeling in Cognos Cube Designer..... | 153 |
| Enabling relational modeling..... | 154 |
| Creating a relational model | 154 |
| Defining query subjects..... | 155 |
| Query items..... | 156 |
| Defining query item sets..... | 157 |
| Determinants..... | 158 |
| Relationships..... | 160 |
| Creating a DMR model..... | 163 |
| Dimensions..... | 164 |
| Measure dimensions and measures..... | 166 |
| Relationships between dimensions and measure dimensions..... | 167 |
| Filters..... | 168 |
| Defining a stand-alone filter..... | 168 |
| Defining an embedded filter..... | 169 |
| Calculations..... | 169 |
| Defining a stand-alone calculation..... | 170 |
| Creating and publishing packages..... | 170 |
| Governors..... | 171 |
| Securing packages..... | 177 |
| Appendix A. Accessibility features..... | 179 |
| Accessibility features in Cognos Cube Designer..... | 179 |
| Keyboard shortcuts for Cognos Cube Designer..... | 179 |

| | |
|--|------------|
| Appendix B. Report considerations..... | 183 |
| Calculated members in reports..... | 183 |
| Relative time calculated members in reports..... | 185 |
| Removal of padding members from reports..... | 186 |
| Appendix C. DCAdmin command-line tool..... | 187 |
| Appendix D. Troubleshooting..... | 189 |
| Possible overflow in measure attributes..... | 189 |
| In-memory aggregates fail to load..... | 189 |
| Issues with dynamic cubes that contain members with duplicate level keys..... | 189 |
| Issues with starting a published dynamic cube in a multi-server environment..... | 190 |
| Notices..... | 191 |
| Index..... | 193 |

Introduction

This document is intended for use with IBM® Cognos® Dynamic Cubes. It describes the processes required to model dimensional metadata and to create dynamic cubes to use as data sources in the Content Manager.

Audience

The following knowledge and experience can help you to use the product.

- Knowledge of OLAP concepts.
- Knowledge of your business requirements.
- An understanding of the structure of your data sources.
- Experience of installing and configuring applications.

Finding information

To find product documentation on the web, including all translated documentation, access [IBM Knowledge Center](http://www.ibm.com/support/knowledgecenter) (<http://www.ibm.com/support/knowledgecenter>).

Accessibility features

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products. IBM Cognos Dynamic Cubes has accessibility features. For information about these features, see the [accessibility section](#) in this document.

IBM Cognos HTML documentation has accessibility features. PDF documents are supplemental and, as such, include no added accessibility features.

Forward-looking statements

This documentation describes the current functionality of the product. References to items that are not currently available may be included. No implication of any future availability should be inferred. Any such references are not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of features or functionality remain at the sole discretion of IBM.

Samples disclaimer

The Sample Outdoors Company, Great Outdoors Company, GO Sales, any variation of the Sample Outdoors or Great Outdoors names, and Planning Sample depict fictitious business operations with sample data used to develop sample applications for IBM and IBM customers. These fictitious records include sample data for sales transactions, product distribution, finance, and human resources. Any resemblance to actual names, addresses, contact numbers, or transaction values is coincidental. Other sample files may contain fictional data manually or machine generated, factual data compiled from academic or public sources, or data used with permission of the copyright holder, for use as sample data to develop sample applications. Product names referenced may be the trademarks of their respective owners. Unauthorized duplication is prohibited.

Chapter 1. What's new

This information will help you plan your upgrade, deployment strategies, and training requirements for IBM Cognos Analytics.

For information about upgrading, see the *IBM Cognos Analytics Installation and Configuration Guide*.

For information about new features for IBM Cognos Analytics, see the *IBM Cognos Analytics New Features Guide*.

To review an up-to-date list of environments that are supported by IBM Cognos Analytics products, including information on operating systems, patches, browsers, web servers, directory servers, database servers, and application servers, see the [IBM Software Product Compatibility Reports page \(www.ibm.com/support/pages/node/735235\)](http://www.ibm.com/support/pages/node/735235).

New features in 10.2.2 FP1

New features for IBM Cognos Dynamic Cubes 10.2.2 Fix pack 1 include enhancements for server stability functionality.

Server stability functionality updates

The following updates have been made to server stability.

- Memory monitoring for Oracle Java™ virtual machines has been added.
- Memory monitoring is available when using balanced garbage collection.
- If a query running on a server is canceled due to insufficient memory, the original report or analysis is now routed to another server within the server group.
- If a cube is refreshing its member cache or if it is restarting, notably on a system with another cube that is actively processing queries, loading members might push the query service beyond its available memory. If this situation occurs, the query service cancels the queries to protect the availability of the cubes that are already active.

For more information, see [“Memory monitoring in the dynamic query mode server” on page 141](#).

New features in 10.2.2

New features for IBM Cognos Dynamic Cubes 10.2.2 include enhancements for in-memory aggregates, memory utilization improvements, near real-time cube updates, hardware sizing calculator, and a new administration utility.

Naming changes

Some naming changes were introduced in IBM Cognos Cube Designer and IBM Cognos Administration.

- The **Data Stores** page on the **Status** tab in IBM Cognos Administration is now named **Dynamic Cubes**. For more information, see [Chapter 12, “Cognos dynamic cubes administration,” on page 117](#).
- Aggregate cubes are now named in-database aggregates.

New options to manage dynamic cubes

You can now pause a dynamic cube and incrementally update the cube data. The new options are available in IBM Cognos Administration from the dynamic cube right-click menu.

The new options are:

- **Pause**

You can pause a dynamic cube to maintain aggregate tables for near real time updates, or to make database configuration changes, such as recycling a database or increasing buffer pools, while a dynamic cube is active.

- **Incrementally update data**

You can use this option to update the aggregate cache and data cache to reflect the newly-added fact rows.

For more information, see [“Starting and managing dynamic cubes”](#) on page 127.

You can also use the DCAdmin command-line tool to pause a dynamic cube and perform incremental updates of the cube data. For more information, see [Appendix C, “DCAdmin command-line tool,”](#) on page 187.

New dynamic cube properties

The following advanced settings that were available in previous versions of the product are now replaced by dynamic cube properties:

- `qsMaxCubeLoadThreads` - replaced by **Maximum hierarchies to load in parallel** property
- `qsMaxAggregateLoadThreads` - replaced by **Maximum in-memory aggregates to load in parallel** property
- `qsMeasuresThreshold` - replaced by **Measures threshold** property.

There is also a new dynamic cube property named **Post in-memory trigger name**.

For more information on the new properties, see [“Setting dynamic cube properties”](#) on page 132.

Updated roles for managing dynamic cubes

The role names used in previous versions of IBM Cognos Dynamic Cubes have been changed so that they are more consistent with the predefined role names within IBM Cognos Analytics. For more information, see [“Access permissions and capabilities for dynamic cubes”](#) on page 117.

Memory monitoring in the dynamic query mode server

The dynamic query server now monitors memory to avoid failing if there is not enough memory available. If available memory is low, it takes action to cancel queries. For more information, see [“Memory monitoring in the dynamic query mode server”](#) on page 141.

Updates to relative time

You can now add relative time members for these periods:

- Future time periods
- Single period custom members, for example, Same Month, Prior Year
- Period-to-date custom members, for example, Quarter to Date, Prior Year
- N-period running total members, for example, 6 month rolling window

You can also control which relative time members to add to a time hierarchy. For more information, see [“Model relative time dimensions”](#) on page 77.

Packages

You can now publish a package containing more than one cube. A package can contain dynamic cubes, virtual cubes, namespaces and folders. For more information, see [“Creating and publishing packages”](#) on page 70.

Import of Framework Manager packages into Cognos Cube Designer

You can import the Framework Manager packages that contain your dimensionally modeled relational (DMR) and relational models into Cognos Cube Designer and use the metadata in the packages to create dynamic cubes. Regardless of what is contained within the Framework Manager model, the metadata from the model that is used to create a dynamic cube must represent a star or snowflake schema.

You can import only packages that are published to the IBM Cognos Analytics content store. Packages that are saved on a disk cannot be imported.

This functionality allows you to leverage your existing IBM Cognos Analytics modeling investment when you implement dynamic cubes. Additional modeling is required after the Framework Manager package is imported into Cognos Cube Designer. The reports that are based on a DMR model are not migrated to the dynamic cube model that is based on the DMR model.

For more information, see [“Importing metadata from a Framework Manager package” on page 40.](#)

Parameter maps

Use parameter maps to substitute settings when a report is run. You can create parameter maps manually, import them from a file, or use an existing query item in the model as the key-value pair for the parameter map.

In dynamic cubes, the parameterization is resolved at cube start-up time.

For more information, see [“Parameter maps” on page 57.](#)

User-defined in-memory aggregates

User-defined in-memory aggregates provide the dynamic cube modelers with the ability to suggest specific in-memory aggregates to be included in the Aggregate Advisor recommendations.

This new type of in-memory aggregates is created in IBM Cognos Cube Designer without recommendations from Aggregate Advisor. However, Aggregate Advisor must be used to generate recommendations with the user-defined aggregates and apply the aggregates to the dynamic cube for usage. For more information, see [“Creating user-defined in-memory aggregates” on page 97.](#)

Automatic optimization of in-memory aggregates

The in-memory aggregates are automatically improved in response to the report queries.

The functionality minimizes the number of manual Aggregate Advisor runs, reduces the need to generate comprehensive workload logs, and improves report performance by adjusting the set of in-memory aggregates over time to better match the query activity.

For more information, see [“Automatic optimization of in-memory aggregates” on page 135.](#)

No need to restart cubes when enabling or disabling workload logging

The dynamic cube property **Enable workload logging** enables or disables workload logging.

When this property is enabled, the workload log file captures the information that represents user workload usage, such as running reports. This log file allows Aggregate Advisor to suggest aggregates, in-database or in-memory, that correspond directly to the reports contained in the log file. You do not need to restart the dynamic cube for this property change to take effect.

For more information, see [“Workload log for Aggregate Advisor ” on page 134.](#)

Hardware sizing calculator

This calculator facilitates quick, initial estimation of hardware resources that are needed to support a dynamic cube. The estimates are based on the two largest dimensions in the cube and the number

of concurrent queries per report. The calculated output gives the required memory size, number of processor cores, and the hard disk space that is needed to support the cube.

For more information, see [“Estimating hardware requirements” on page 70.](#)

Memory utilization improvements for dynamic cubes

The following memory utilization improvements were implemented:

- Dimension members can now be optionally shared between dynamic cubes and virtual cubes that reside on the same server to reduce the overall memory footprint. For more information, see [“Defining a shared member cache” on page 47.](#)
- The size of the member cache has been reduced, and requires approximately 550 bytes per member.
- The query service now recognizes when the JVM heap is nearly full and attempts to release resources from in-memory caches to avoid out of memory exceptions.

Near real-time updates of dynamic cubes data

With near real-time updates, data can be inserted into fact and aggregate tables in the data warehouse without stopping dynamic cubes. Dynamic cubes can consume the inserted new records immediately and IBM Cognos Analytics queries return consistent data. The data caches are updated and not rebuilt.

In previous versions of IBM Cognos Analytics, to maintain consistent data values between the fact tables, aggregate tables, and data caches in a running dynamic cube, it was necessary to stop the cube before you could apply any changes to the data warehouse. This step was required to change the data values in the data warehouse tables during the execution of end user analytics queries. When the update was complete, it was necessary to restart the cube in order to rebuild the data caches to reflect the new table values.

For more information, see [Chapter 13, “Near real-time updates of dynamic cubes data,” on page 147.](#)

Named sets

A named set is an expression that defines a set of members. You can define named sets in Cognos Cube Designer in the context of a dynamic cube or a virtual cube.

Named set expressions can be any valid member set expressions. They can also include macro expressions. Named sets are accessible in the authoring interfaces of IBM Cognos Analytics, including Reporting and Cognos Workspace Advanced.

For more information, see [“Defining named sets” on page 56.](#)

Relational and DMR modeling

You can now create relational and dynamically modeled relational (DMR) models in Cognos Cube Designer.

The relational and DMR experience in Cognos Cube Designer 10.2.2 is in its early stage and does not currently offer all of the usability features that IBM Cognos Framework Manager does. For more information, see [Chapter 14, “Relational and DMR modeling in Cognos Cube Designer,” on page 153.](#)

DCAdmin command-line tool

A new command-line tool is now available with IBM Cognos Analytics server. You can use this tool to run various administrative commands on dynamic cubes.

For more information, see [Appendix C, “DCAdmin command-line tool,” on page 187.](#)

New features in 10.2.1.1

In IBM Cognos Dynamic Cubes 10.2.1.1, new features include dimension filters and measure dimension filters, measure folders, sorting measures, and embedded prompts and macros.

Dimension filters and measure dimension filters

You can now create dimension filters to restrict the members available in a published dynamic cube. For more information, see [“Dimension filters” on page 55](#).

You can also create measure dimension filters to restrict the fact data available in a published dynamic cube. For more information, see [“Measure dimension filters” on page 67](#).

Measure folders and sorting

You can now create folders in a measure dimension to contain regular measures and calculated measures. For more information, see [“Creating a measure folder” on page 68](#).

You can also change the order in which measures and folders are sorted. For more information, see [“Changing the sort order of measures and folders” on page 69](#).

Embedded prompts and macros

You can now embed prompts and macros in a calculated member or calculated measure expression. For more information about using prompts and macros, see the *IBM Cognos Framework Manager User Guide*.

New features in 10.2.1

See the following topics for new features since the last release. Links to directly related topics are included.

Importing InfoSphere Warehouse Cubing Services cube metadata

You can now import cube metadata from an IBM InfoSphere® Warehouse Cubing Services model.

For more information, see [“Import InfoSphere Warehouse Cubing Services cube metadata” on page 41](#).

Generating cubes and dimensions

From the Data Source Explorer in IBM Cognos Cube Designer, two new options are available to help reduce the overall time to build a cube. **Generate, Cube with dimensions using data sampling** creates a set of dimensions that are based on a selected fact table and the tables it joins. Each dimension is generated with one or more levels. **Generate, dimension using data sampling** creates a dimension with one or more levels that are based on the selected table.

For more information, see [“Defining a dynamic cube based on a relational table” on page 62](#) and [“Defining a dimension based on a relational table” on page 46](#).

The **Generate, Cube** option from the previous release has been renamed to **Generate, Cube with basic dimensions**. The functionality remains unchanged.

Aggregation rules

Three aggregation rules for measures have been added for this release. From the **Aggregation Rules** tab, you can access the **First**, **Last**, and **Current Period** options from the **Aggregation Rule** drop down list.

For more information, see [“Aggregation rules”](#) on page 28.

Aggregation Advisor

The Aggregation Advisor now suggests summary tables to assist in the loading of in-memory aggregates.

Improved security

Security features have been enhanced in the following areas for this release:

- Member security

Security rules can now be stored in relational database lookup tables, better enabling the automation of security definitions for dynamic cubes.

- Dimension security

It is now possible to secure user access to entire dimensions within a dynamic cube.

- Attribute security

It is now possible to restrict user access to specific member attributes in a hierarchy. Member security definitions stored in database tables.

- Refresh security

It is now possible to refresh security without having to restart a dynamic cube as long as there no significant changes to the modeled cube. If there are changes to dimensions, hierarchies, levels, or attributes, you must restart the dynamic cube.

For more information, see [Chapter 11, “Define security,”](#) on page 107.

Performance issues

In Cognos Cube Designer, there is a new **Performance Issues** tab that shows a list of all the performance issues for objects. These are issues that affect how well a dynamic cube performs when it is published and started.

For more information, see [“Validate a project and individual objects”](#) on page 44.

Centralized administration interface for dynamic cubes

A new page named **Data Stores** was added on the **Status** tab in IBM Cognos Administration. On this page, administrators can view, configure, manage, and monitor all dynamic cubes available in the IBM Cognos environment.

For more information, see [Chapter 12, “Cognos dynamic cubes administration,”](#) on page 117.

Chapter 2. Cognos Dynamic Cubes overview

In a dimensional data warehouse, you model relational database tables using a star or snowflake schema. This type of data warehouse differs from a traditional OLAP model in the following ways:

- It stores information about the data in fact and dimension tables rather than in proprietary OLAP data structures.
- It describes the relationships within the data using joins between the dimension and fact tables, the collection of dimension keys in a fact table, and the different attribute columns in a dimension table.

IBM Cognos Dynamic Cubes adds an in-memory relational OLAP component to the dynamic query mode server to provide a multidimensional view of a relational data warehouse with accelerated performance. You can then perform OLAP analysis using the Cognos Dynamic Cubes server.

Cognos Dynamic Cubes differs from Cognos dimensionally-modeled relational (DMR) data sources for the following reasons:

- It provides increased scalability and the ability to share data caches between users for better performance.
- It allows you to create a dynamic cube data source that is pre-loaded with dimensions.
- It allows for a richer set of dimensional modeling options and the explicit management of the member and data caches of a dynamic cube.

The benefits of Cognos Dynamic Cubes can be achieved only when using a dynamic cube as a data source. To use a dynamic cube as a data source, you must use the dynamic query mode.

Cognos Dynamic Cubes introduces a performance layer in the Cognos query stack to allow low-latency, high-performance OLAP analytics over large relational data warehouses. By using the power and scale of a relational database, Cognos Dynamic Cubes can provide OLAP analytics over terabytes of warehouse data.

Cognos Dynamic Cubes uses the database and data cache for scalability, and also uses a combination of caching, optimized aggregates (in-memory and in-database), and optimized SQL to achieve performance. The Cognos Dynamic Cubes solution includes the following characteristics:

- It uses simple, multi-pass SQL that is optimized for the relational database.
- It is able to minimize the movement of data between the relational database and the Cognos Dynamic Cubes engine.

This data control is achieved by caching only the data that is required and by moving appropriate calculations and filtering operations to the database. At run time, only fact data is retrieved on demand.

- It is aggregate-aware, and able to identify and use both in-memory and in-database aggregates to achieve optimal performance.

Aggregate awareness (aggregates tables that are created in the database and modeled into a dynamic cube) uses specialized log files to allow the dynamic query mode server to decompose queries to take advantage of the aggregate tables.

- It optimizes aggregates (in-memory and in-database) using workload-specific analysis.

Aggregate Advisor, part of IBM Cognos Dynamic Query Analyzer, analyzes the performance of dynamic cubes using log files and provides suggestions for improving cube performance.

- It can achieve low latency over large data volumes, such as billions of rows or more of fact data and millions of members in a dimension.

By using virtual cubes, companies can still present the complete view of the data, but need to refresh only smaller sets of data, leaving pre-cached query results for larger static sets. Users experience better performance for queries run against pre-cached results.

Evaluating your data

Before starting to model a cube, it is important to understand how your data affects the processing in IBM Cognos Cube Designer.

Referential integrity in data warehouses

Most databases today support referential integrity. However, it is typically turned off or is made declarative and instead is enforced during extract, transform, and load (ETL) processing. Erroneous modifications made to the data during or outside of the ETL process can create cases where a fact table has no matching dimension records.

Each data point in a dynamic cube is defined by a member from each dimension in the cube. If a value is required for some data point, then the SQL generated by Cognos Dynamic Cubes does not specify a filter on the table associated with a particular dimension if the member of that dimension is the All member. This allows for smaller SQL queries and also faster executing queries.

When a dimension is in scope, the join between the fact and dimension table is specified in the SQL query and the dimension is filtered by an explicit set of dimension key values. When the member of a dimension is the All member, dynamic cubes will not specify a filter for that dimension. All records are included, even records with invalid or missing dimension key values. This difference causes a discrepancy between values, depending upon which dimensions are involved in a query.

Even if your fact records have invalid or unknown dimension key values, you should validate your records before implementing Cognos Dynamic Cubes. Run an SQL query similar to the following for each dimension in a dynamic cube. This determines if there are any fact records with invalid dimension key values. Any returned data is the set of invalid dimension key values. If no data is returned, there are no referential integrity errors.

```
select distinct FACT.Key
from FactTable FACT
where not exists
(select *
 from DimensionTable DIM
 where DIM.Key = FACT.Key)
```

The SQL query can also be used as a subquery, to obtain the full set of records from the fact table.

If your fact table might contain records with invalid or unknown dimension key values, a common practice is to create a row in the dimension table to represent these dimension keys. New fact rows with invalid or unknown dimension key values can be assigned this dimension key value until the fact records and the dimension table can be updated with correct information. With this practice, records with problematic dimension key values are visible, regardless of which dimensions are involved in a report or analysis.

You should also validate snowflake dimensions.

You may have a situation where tables in a snowflake dimension are joined on a column for which the outer table did not contain values for rows in the inner table. In this case, the inner dimension table joins to the fact table, but the outer dimension table does not join to the inner dimension table.

To ensure that snowflake dimensions do not have this type of referential integrity error, run an SQL query similar to the following. In this example, the dimension is built from two tables, D1_outer and D2_inner. D2_inner is joined to the fact table. Key is the column on which the two dimension tables are joined.

```
select distinct INNER.Key
from D2_inner INNER
where not exists
(select *
 from D1_outer OUTER
 where OUTER.Key = INNER.Key)
```


Chapter 3. Cognos Dynamic Cubes workflow

IBM Cognos Dynamic Cubes brings faster, more powerful, cube performance into the IBM Cognos reporting environment. Cognos Dynamic Cubes is used to improve access to large sets of data.

The following diagram illustrates the relationship between the main activities performed using IBM Cognos Dynamic Cubes and the corresponding tools. IBM Cognos Cube Designer provides dynamic cube design and modeling capability. The Administration Console is used to deploy and manage the cube data. The dynamic query mode (DQM) server maintains the cube data. Studio applications use the data in reporting environments. In addition, various tools, such as Dynamic Query Analyzer, are used to analyze and optimize the data as necessary.

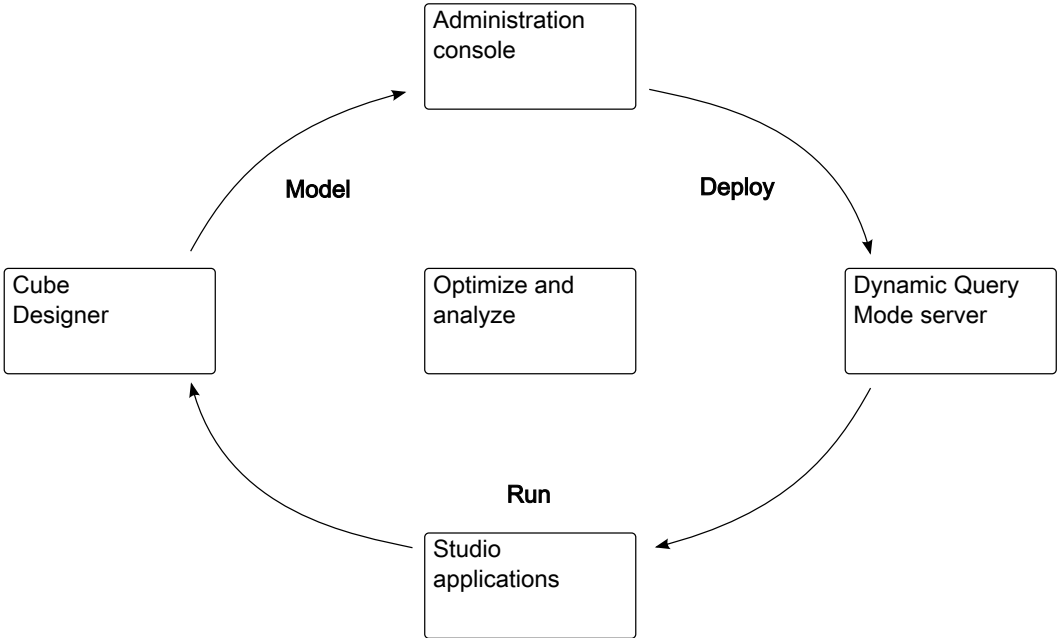


Figure 1. Relationships between Cognos Dynamic Cubes activities and tools

The following diagram shows the five major steps in a typical process flow, showing the users who are involved at each step.

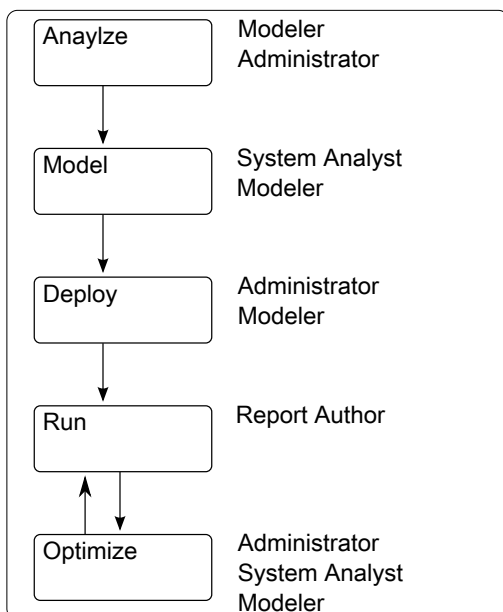


Figure 2. Cognos Dynamic Cubes typical process flow

Analyze the data

Before installing IBM Cognos Dynamic Cubes, the modeler and relational database administrator prepare for project implementation by completing the following tasks:

- Determining whether the data is a good candidate for Cognos Dynamic Cubes.
- Reviewing prerequisites to ensure correct implementation.

For more information about assessing your data and understanding prerequisites, see [Chapter 2, “Cognos Dynamic Cubes overview,”](#) on page 7.

Design and model a dynamic cube

The system analyst determines high-level business requirements and evaluates cube design against reporting requirements.

The modeler creates a basic dynamic cube, adds features to satisfy the business requirements and ensures that the cube is available to IBM Cognos Administration. Within IBM Cognos Cube Designer, the modeler performs tasks such as:

- Importing relational metadata to use as the basis for dynamic cube design.
- Designing dynamic, aggregate, and virtual cubes.
- Setting cube-level security for hierarchies and measures.
- Publishing the dynamic cube.

For more information about designing and modeling dynamic cubes, see the following topics:

- [“Import metadata”](#) on page 39
- [“Model a dynamic cube”](#) on page 61
- [“Calculated members”](#) on page 73
- [“Modeling in-database aggregates”](#) on page 93
- [Chapter 10, “Virtual cube modeling,”](#) on page 99
- [Chapter 11, “Define security,”](#) on page 107
- [“Deploying and publishing dynamic cubes”](#) on page 69

Optionally, the modeler runs *Aggregate Advisor* for recommendations regarding the dynamic cube design. For information about *Aggregate Advisor*, see the *IBM Cognos Dynamic Query Analyzer User Guide*.

Deploy and manage a dynamic cube

After dynamic cubes are published to Content Manager, the Administrator handles the initial configuration and subsequent management. Within IBM Cognos Administration, the administrators perform tasks such as:

- Setting the **Access Account** property in the Administration Console.
- Assigning users, groups and roles to security views.
- Assigning a server group to the dispatcher.
- Assigning a routing set to all packages associated with a dynamic cube.
- Creating a routing rule to route queries for the routing set to the server group.
- Configuring the query service and the dynamic cube for a dispatcher.
- Starting the dynamic cube for initial use.
- Refreshing the dynamic cube, as necessary.
- Stopping the dynamic cube (soft or hard stop) while the data warehouse is being updated.
- Optionally, turning on logging. Log files are required to optimize the cube.
- Clearing workload logs.

For more information about deploying and managing dynamic cubes, see [Chapter 12, “Cognos dynamic cubes administration,”](#) on page 117 and the *IBM Cognos Analytics Administration and Security Guide*.

Run reports using dynamic cube data

The report author uses the dynamic cube as a data source in reporting applications.

Optimize a dynamic cube

To optimize individual cube performance, the administrator can monitor the metrics of the dynamic cubes, and make changes, if necessary, to the cube configuration.

To further optimize performance, the system analyst can run a series of reports that are a representative workload against the dynamic cube. The resulting workload logs are used by *Aggregate Advisor* to return recommendations for additional in-memory and in-database aggregates. The analyst can also examine request execution log files in the *Dynamic Query Analyzer*. The log files help the analyst understand where time is spent within the dynamic cube engine, the type of SQL queries that are posed, how much time is spent executing the queries, and how many rows of data are returned. For information about *Aggregate Advisor*, see the *IBM Cognos Dynamic Query Analyzer User Guide*.

When you save in-memory aggregate recommendations to the content store, they are loaded automatically the next time the dynamic cube is started.

For in-database aggregate recommendations, the database administrator creates the aggregate tables in the database and the modeler uses *IBM Cognos Cube Designer* to model and publish the dynamic cube. For more information, see [“Modeling in-database aggregates”](#) on page 93.

After new aggregates are published by the modeler, the administrator sets the in-memory aggregate size and restarts the dynamic cube to use new aggregates.

For detailed information, see [Chapter 12, “Cognos dynamic cubes administration,”](#) on page 117.

Workflow summary

To prepare for and manage project implementation, there are tasks external to the IBM Cognos software and tasks performed using IBM Cognos software. The following table shows a summary of responsibilities in each step of the workflow.

Table 1. Workflow responsibilities by role

| Workflow | Responsibilities | Tools | Role |
|--------------------|---|--|-------------------------------------|
| Analyze, configure | Gather requirements and best practices. Determine best practices. Prepare an overall design. Perform hardware assessments. | | Solutions architect |
| Configure | Determine operating system administration changes. Perform middleware installation and maintenance. | O/S command tools, system administration console | System administrator |
| Analyze, model | Design the database physical model. Design the multi-dimensional model. | Modeling tools, document/presentation software | Data architect |
| Analyze, model | Gather business requirements. Design the logical model. Prepare the security definition. | Modeling tools, document/presentation software | Business/ Application consultant |
| Model, optimize | Design dynamic cubes. Define security rules and views. | IBM Cognos Cube Designer, IBM Cognos Dynamic Query Analyzer | |
| Manage, deploy | Configure and manage dynamic cubes. | Cognos Administration Console, Cognos Dynamic Query Analyzer | |
| Manage, deploy | | Cognos Administration Console | Cognos administrator (security) |
| Manage, deploy | Manage IBM Cognos data sources. Assign users to security views. | Cognos Administration Console | |
| Optimize, Model | Evaluate overall performance. Run Aggregate Advisor. | Cognos Cube Designer, Cognos Dynamic Query Analyzer | |

Table 1. Workflow responsibilities by role (continued)

| Workflow | Responsibilities | Tools | Role |
|----------------------------|--|--|------------------------|
| Run | Author reports, analyses or dashboards for use by collection of users | Cognos Analytics client applications | Cognos report author |
| Configure, Model, Optimize | Implement database updates Perform database maintenance such as extract, transform and load (ETL) processes, backup and recovery. | Database administration console, ETL tools | Database administrator |

Chapter 4. Dimensional metadata and dynamic cubes

Understanding concepts relating to dimensional metadata and dynamic cubes helps you to plan and create effective dynamic cubes.

Dimensional metadata

In IBM Cognos Dynamic Cubes, dimensional metadata refers to dimensions and hierarchies. You can create commonly used dimensional metadata independent of any dynamic cubes in a project. The appropriate dimensional metadata can then be shared by one or more cubes in a project.

You can also create dimensional metadata that is connected to a specific dynamic cube.

Dimensions

In IBM Cognos Dynamic Cubes, you can create two types of dimensions: regular and parent-child.

A regular dimension is a collection of hierarchies and levels that describe one aspect of a measure, such as Customer or Product. This type of dimension can contain one or more hierarchies. A hierarchy uses levels to describe the relationship and order of dimension attributes. Related attributes and the joins that are required to group these attributes are defined in the dimension. For more information, see [“Hierarchies” on page 15](#).

A parent-child dimension contains dimension data based on a recursive relationship and is not level-based. This type of dimension can contain only a single parent-child hierarchy. For more information, see [“Parent-child hierarchies” on page 21](#).

Data for regular dimensions and parent-child dimensions is typically stored in dimension tables.

Cognos Dynamic Cubes also supports degenerate dimensions. A degenerate dimension is a regular dimension for which dimension data is stored in a fact table. When modeling a dynamic cube based on a degenerate dimension, you do not need to specify a measure-to-dimension join.

Hierarchies

A hierarchy uses levels to describe the relationship and order of dimension attributes. For example, a Customer dimension might contain a Region hierarchy.

For more information about attributes and levels, see [“Attributes” on page 24](#) and [“Levels” on page 22](#).

IBM Cognos Dynamic Cubes supports balanced, unbalanced, and ragged hierarchies. Padding members are used to balance unbalanced and ragged hierarchies, so they appear as balanced hierarchies in the IBM Cognos studios. For more information, see [“Padding members” on page 18](#).

Multiple hierarchies

Multiple hierarchies can be defined for dimensions containing level-based hierarchies.

You create multiple hierarchies for a dimension when you want to organize dimension members in different ways. For example, in a Time dimension, you can create hierarchies for Calendar year and Fiscal year.

Because dimension members in separate hierarchies can be used to represent the same entity, each hierarchy should contain the same lowest level members. For example, in a Time dimension, the Calendar hierarchy might have Year, Month, and Day levels. The Fiscal hierarchy might have Year, Quarter, and Day levels. The lowest level in both dimensions is the Day level.

Hierarchies that are modeled using a shared level can be optimized during query execution to remove non-intersecting values. To do this, you must ensure the **Remove non-existent tuples** property is set in a dynamic cube. For more information, see [“Model a dynamic cube” on page 61](#).

Related concepts

[Balanced hierarchies](#)

[Unbalanced hierarchies](#)

[Ragged hierarchies](#)

[Padding members](#)

[Extraneous padding members](#)

Balanced hierarchies

In a balanced hierarchy, the branches of the hierarchy all descend to the same level. The parent of every member comes from the next highest level.

A balanced hierarchy can be used to represent time where the meaning and depth of each level, such as Year, Quarter, and Month, is consistent. They are consistent because each level represents the same type of information, and each level is logically equivalent. The following diagram shows an example of a balanced time hierarchy.

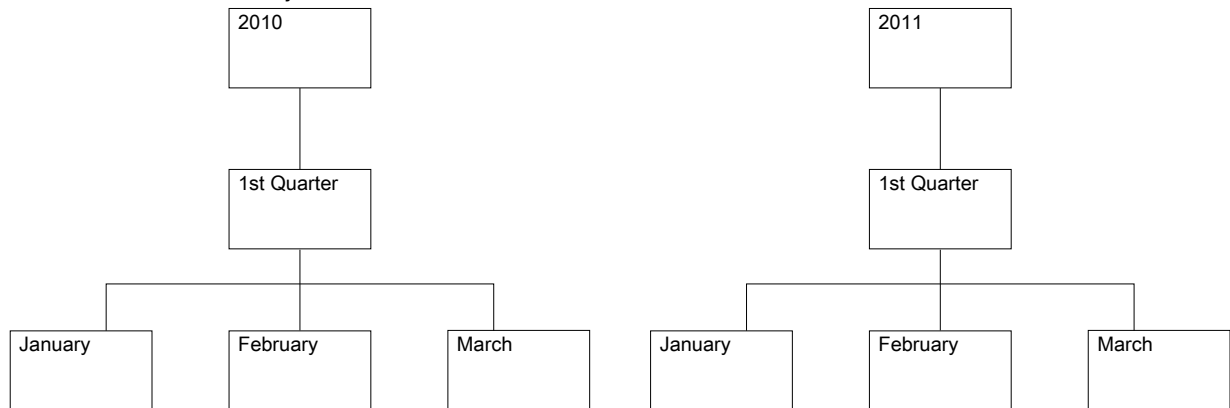


Figure 3. Example of a balanced hierarchy

Related concepts

[Multiple hierarchies](#)

[Unbalanced hierarchies](#)

[Ragged hierarchies](#)

[Padding members](#)

[Extraneous padding members](#)

Unbalanced hierarchies

Unbalanced hierarchies include levels that are logically equivalent, but each branch of the hierarchy can descend to a different level. In other words, an unbalanced hierarchy contains leaf members at more than one level. The parent of every member comes from the level immediately above.

An example of an unbalanced hierarchy is the following organization chart, which shows reporting relationships between employees in an organization. The levels within the organizational structure are unbalanced, with some branches in the hierarchy having more levels than others.

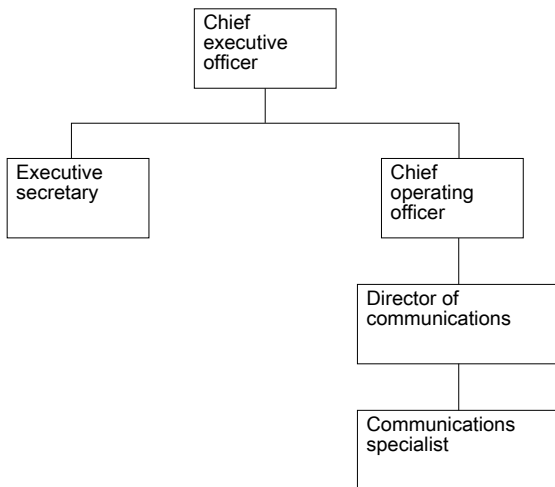


Figure 4. Example of an unbalanced hierarchy

IBM Cognos Dynamic Cubes inserts padding members to balance such hierarchies. For more information, see [“Padding members”](#) on page 18.

Related concepts

[Multiple hierarchies](#)

[Balanced hierarchies](#)

[Ragged hierarchies](#)

[Padding members](#)

[Extraneous padding members](#)

Ragged hierarchies

In a ragged hierarchy, the parent of at least one member does not come from the level immediately above, but a level higher up.

The following diagram shows a Geography hierarchy with Continent, Region, State, and City levels defined. One branch has North America as the continent, Canada as the region, Manitoba as the state, and Winnipeg as the city. Another branch has Europe as the continent, Greece as the region, and Athens as the city, but has no entry for the state level because this level is not applicable. The parent of Athens is at the region level rather than the state level, creating a ragged hierarchy.

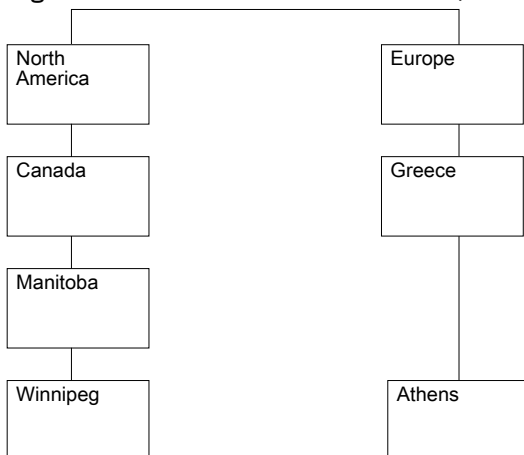


Figure 5. Example of a ragged hierarchy

IBM Cognos Dynamic Cubes inserts padding members to balance such hierarchies. For more information, see [“Padding members”](#) on page 18.

Related concepts

[Multiple hierarchies](#)

[Balanced hierarchies](#)

[Unbalanced hierarchies](#)

[Padding members](#)

[Extraneous padding members](#)

Padding members

IBM Cognos Dynamic Cubes inserts padding members to balance unbalanced and ragged hierarchies. Padding members do not represent actual dimension members; they are visible only for navigational and performance reasons.

You can reference a padding member in an expression in the same way as any other hierarchy member.

Padding members can include a blank caption or the same caption as the parent. The following diagram illustrates a ragged hierarchy with a padding member included in the Europe branch. A blank caption was used as the caption for the padding member.

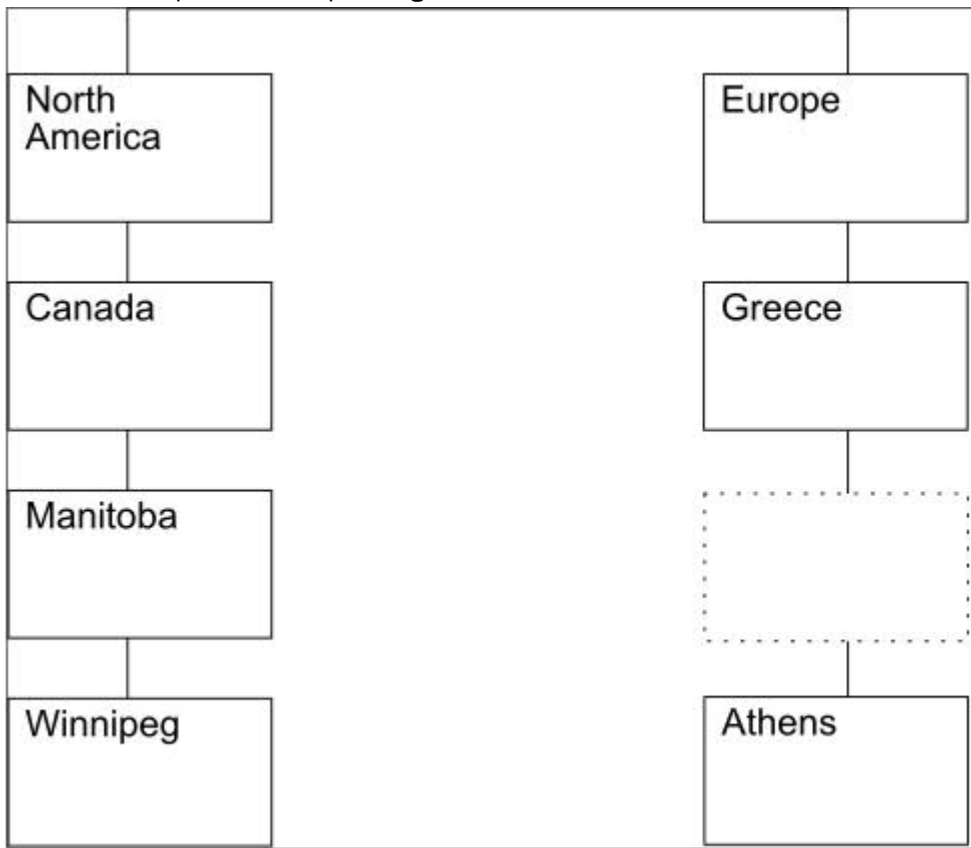


Figure 6. Example of a ragged hierarchy with blank padding member

In the IBM Cognos studios, the metadata for this hierarchy with blank captions would show a level without a caption, as in the following example:

```
North America
|--Canada
    |--Manitoba
        |--Winnipeg
Europe
|--Greece
    |--
        |--Athens
```

Figure 7. Example metadata showing blank padding member

The metadata for the same hierarchy using parent captions would show a level that uses the same caption as the parent, as in the following example:

```
North America
|--Canada
    |--Manitoba
        |--Winnipeg
Europe
|--Greece
    |--Greece
        |--Athens
```

Figure 8. Example metadata showing parent padding member

A dimensional member can have only one child padding member.

The use of padding members can result in skewed calculations related to the members of a hierarchy level. For information about removing skewed data from reports, see [“Removal of padding members from reports” on page 186](#).

Related concepts

[Multiple hierarchies](#)

[Balanced hierarchies](#)

[Unbalanced hierarchies](#)

[Ragged hierarchies](#)

[Extraneous padding members](#)

Extraneous padding members

With a level-based hierarchy, you can assign values from a dimension table to any member within the hierarchy; in other words to leaf members and non-leaf members. Data for non-leaf members can also be obtained by rolling up (aggregating) data from leaf members.

Tip: To roll up data for non-leaf members, the table that is used to model a level-based hierarchy must join to the fact table by using surrogate keys.

For example, a sales manager can also be a sales person with their own sales values. To assign sales values to the sales manager, the dimension table must contain a row in which the level key values for all levels below the manager level are Null.

For example, a sales manager can also be a sales person with their own sales values. The following example dimension table shows data for two sales people (Mark and Fred), and their sales manager (James). James is a non-leaf member which has a separate data value (100).

| Manager | Sales Person | Sales Total |
|---------|--------------|-------------|
| James | Mark | 15 |
| James | Fred | 20 |
| James | <null> | 100 |

Using IBM Cognos Dynamic Cubes, you can construct this hierarchy in one of the following ways:

- Create a path of extraneous padding members.

This option creates a full path of padding members from the non-leaf member to the leaf level to ensure that the hierarchy is balanced. It also provides a value at the lowest level so that the data can be rolled up. This is known as a rollup hierarchy.

The caption of these members can be blank or the same as the non-leaf member. If a non-leaf member has a value associated with it, this value is assigned to the padding member, which allows the contribution of a non-leaf member to its own rollup value.

- Remove the path of extraneous padding members.

Depending on the number of hierarchy levels, and the number of non-leaf member values, adding a path of extraneous padding members can result in a large hierarchy. To allow easier navigation of such a hierarchy, you can remove these paths.

To ensure that a hierarchy is balanced, you can remove a path of extraneous padding members only where a non-leaf member includes other leaf members.

If paths are removed for any hierarchy, the entire dimension is identified as a non-rollup hierarchy. This prevents the query engine from assuming that the value of a parent is the rollup of its children. In addition, extraneous padding members are assigned a value of Null for all measures. This typically occurs when a detail filter is applied at a level below the lowest projected level in a report, or if the context filter (slicer) in a report contains multiple members from a single hierarchy.

The following example illustrates a data from a hierarchy with a path of extraneous padding members.

| Manager | Sales Person | Sales Total |
|---------|--------------|-------------|
| James | Mark | 15 |
| James | Fred | 20 |
| James | James | 100 |

By default, the path of extraneous padding members is removed in a level-based hierarchy. To show or remove the path, you must set the **Show Extraneous Padding Members** property. For more information about setting this property, see [“Model hierarchies”](#) on page 48.

Related concepts

[Multiple hierarchies](#)

[Balanced hierarchies](#)

[Unbalanced hierarchies](#)

[Ragged hierarchies](#)

[Padding members](#)

Parent-child hierarchies

A parent-child hierarchy contains relational dimension tables based on a recursive relationship for which there are no predefined levels. For example, an Employee parent-child hierarchy may specify Supervisor as the parent member, and Employee as the child member. The relationships within the data determine what is visible to report users in the IBM Cognos studios, and you can drill down from member to member according to the defined relationships.

IBM Cognos Dynamic Cubes supports parent-child hierarchies.

Data members

With a parent-child hierarchy, you can assign values from a dimension table to any member within the hierarchy; in other words to leaf members and non-leaf members. Data for non-leaf members can also be obtained by rolling up (aggregating) data from leaf members.

For example, a sales manager can also be a sales person with their own sales values. The following example dimension table shows data for two sales people (Mark and Fred), and their sales manager (James). In this example, Mark and Fred are leaf members, and James is a non-leaf member.

Table 4. Example dimension table for a parent-child hierarchy

| Sales person | Sales |
|--------------|-------|
| Mark | 15 |
| Fred | 20 |
| James | 100 |

In the corresponding hierarchy structure, sales person values roll up to the sales manager. This is known as a rollup hierarchy.

The following example illustrates the report data for a rollup hierarchy with non-leaf members shown. The report includes two values for non-leaf member James - the child value that is assigned from the dimension table (100) and the total rolled up sales value that includes this child value (135).

Table 5. Example report data with a non-leaf member shown

| Sales person | Sales |
|--------------|------------|
| Mark | 15 |
| Fred | 20 |
| James | 100 |
| James | 135 |

The following example illustrates the same report data using a non-rollup hierarchy, where non-leaf members hidden.

| <i>Table 6. Example report data with a non-leaf member hidden</i> | |
|---|--------------|
| Sales person | Sales |
| Mark | 15 |
| Fred | 20 |
| James | 135 |

Rolling up report data in a non-rollup hierarchy causes two problems:

- Data for non-leaf members is not explicitly shown because it is already rolled up.
To work out the individual value of a non-leaf member, you must extrapolate the data.
- If a parent-child hierarchy contains hidden non-leaf members, the entire dimension is identified as a non-rollup hierarchy.

This prevents the query engine from assuming that the value of a parent is the rollup of its children. You must set the data members to be visible to allow a hierarchy to be identified as a rollup hierarchy.

When you model a dynamic cube, it is important to consider the presentation of a hierarchy against the effect on reports/analyses posed against the hierarchy, the parent dimension, and related hierarchies.

By default, non-leaf members are hidden in a parent-child hierarchy. To show or hide non-leaf members, you must set the **Show Data Members** property. For more information about setting this property, see [“Model parent-child hierarchies”](#) on page 52.

If the **Show Data Members** property is set to true, a child member is added to each non-leaf member in a parent-child hierarchy. The caption of these members can be blank or the same as the non-leaf member. If a non-leaf member has a value associated with it, this value is assigned to the child data member, which allows the contribution of a non-leaf member to its own rollup value.

Levels

A level is a collection of attributes related to one aspect of a hierarchy. For example, a Region hierarchy can contain States and City levels.

For more information about attributes, see [“Attributes”](#) on page 24.

You can define an All level at the highest level of a hierarchy. An All level contains a single member that aggregates data from all members in the child levels of the hierarchy. For example, you can include an All level in a Region hierarchy that aggregates data for all cities, in all states, in all regions.

Important: There are many ways to model a hierarchy using levels. Whether you follow best practice or different modeling techniques, it is important that you define each level so that the level key attributes uniquely identify the values in that level.

Best practice modeling

Both star and snowflake schemas can be used to implement best practice modeling. For example, in a star schema the relational data for each dimension is stored in a single dimension table that contains ID columns for each of the levels in the dimension, and each ID column uniquely identifies the values in the level. You might have a single dimension table for the Region dimension that contains the following columns:

| <i>Table 7. Example of a single dimension table using best practice modeling</i> |
|--|
| Columns in a best practice Region dimension table |
| City ID (Primary key) |
| City name |
| City mayor |

Table 7. Example of a single dimension table using best practice modeling (continued)

| Columns in a best practice Region dimension table |
|---|
| State ID |
| State name |
| State governor |
| Region ID |
| Region name |

Alternative modeling

If you do not have unique ID data columns for each level in your hierarchy, you must be careful when you define the level key attributes for each level. For example, you might have a single dimension table for the Region dimension that contains the following columns:

Table 8. Example of a single dimension table using alternative modeling

| Columns in an alternative Region dimension table |
|--|
| City ID (Primary key) |
| City name |
| City mayor |
| State name |
| State governor |
| Region name |

You can create a hierarchy that contains Region, State, and City levels, like in the best practice modeling example. However, you must carefully define the level key attributes to ensure that each row in the level can be uniquely defined. For example, City name does not uniquely define the City level because there are cities with the same name in the United States and in England. The only way to uniquely define the City level is with the combination of the Region name, State name, and City name attributes, as shown in the following table.

Table 9. Example of unique level key attributes using multiple columns

| Level | Level key attributes | Level related attributes |
|--------|------------------------------------|--------------------------|
| Region | Region name | |
| State | Region name, State name | State governor |
| City | Region name, State name, City name | City mayor |

Joins

A join combines columns from two relational tables using an operator to compare the columns. A join uses attributes that reference columns in the tables being joined.

The simplest form of a join uses two attributes: one that maps to a column in the first table and one that maps to a column in the second table. You also specify an operator to indicate how the columns are compared. For example, "Time ID = time_id".

A join can also model composite joins where two or more columns from the first table are joined to the same number of columns in the second table. A composite join uses pairs of attributes to map

corresponding columns together. Each pair of attributes has an operator that indicates how that pair of columns is compared. For example, "Customer Number = customer_number AND Store Number = store_number".

A join also has a type and cardinality. The join types map to relational join types. Joins are primarily used to join the cube dimensions to the relational tables. Joins can also be used to join dimension tables together in snowflake schema.

The most common type of join is the one-to-many equality join.

Attributes

An attribute is an item used to describe part of a level. For example, a Product level can have a Color attribute. An attribute contains an expression that can be a simple mapping to a data source column or a more complex expression. Complex expressions can combine multiple columns or attributes. They can use functions that are supported against a relational data source, including user-defined functions, if necessary.

When modeling levels in IBM Cognos Cube Designer, there are some special attributes you can define:

- **Member caption** does not appear as a separate attribute of a level; it is used only as the caption for hierarchy members.
- **Member description** appears as a separate attribute with the name *level name* description.
- **Level unique key** appears as a separate attribute with the name *level name* key.

When additional attributes are used in an expression, they cannot form attribute reference loops. For example, if Attribute A references Attribute B, then Attribute B cannot reference Attribute A.

Attribute names must be unique from the names of all other attributes in a dimension.

Dynamic cubes

A dynamic cube represents a dimensional view of a star or snowflake schema. It is based on a single fact table and defines the relationships between dimensions and measures.

To model a basic dynamic cube, you must ensure that it contains the following items:

- A measure dimension that contains at least one measure
- At least one dimension
- At least one hierarchy and associated levels defined for each dimension
- Mappings between the measures and dimensions
- Attributes that reference table columns either directly, by expressions, or by an expression that is a constant value

Measures are used to aggregate data from a fact table using specified dimensions. They describe data calculations using columns in a relational table. The following diagram shows how measures relate to relational data.

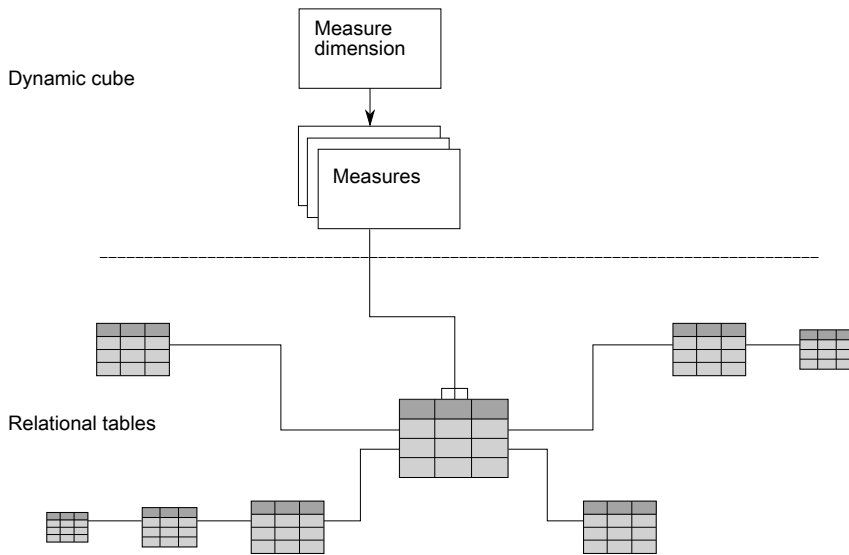


Figure 9. Relationship between measures and relational data

Dimensions are connected to a measure using joins. A hierarchy provides a way to calculate and navigate a dimension. It stores information about how the levels within a dimension are related to each other, and how they are structured. Each dimension has one or more hierarchies that contain levels with sets of related attributes. The following diagram shows how dimensions are built from relational tables.

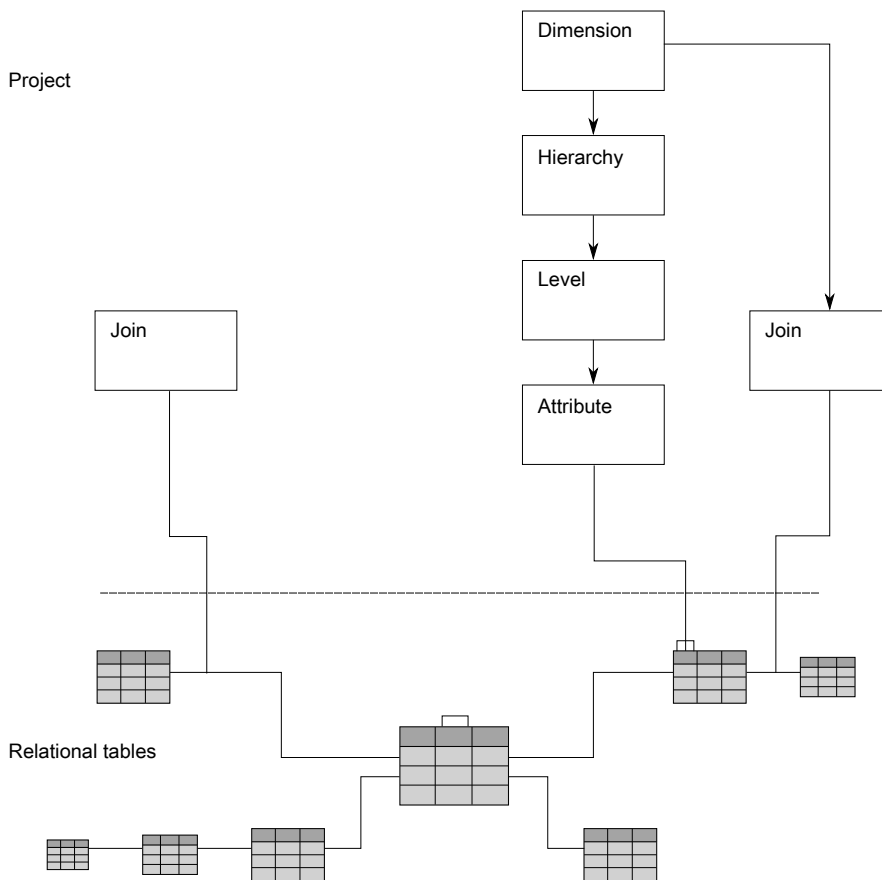


Figure 10. Relationship between dimensions in a project and the source relational tables

In a star schema, joins are used to connect tables to create a dimension or a measure. Joins can also connect a measure dimension to specific dimensions. The dimensions reference their corresponding hierarchies, levels, attributes, and related joins. A measure dimension references its measures, attributes,

and related joins. In a snowflake schema, joins can also connect tables between dimensions. The following diagram shows how the items fit together in a dynamic cube and map to a relational snowflake schema.

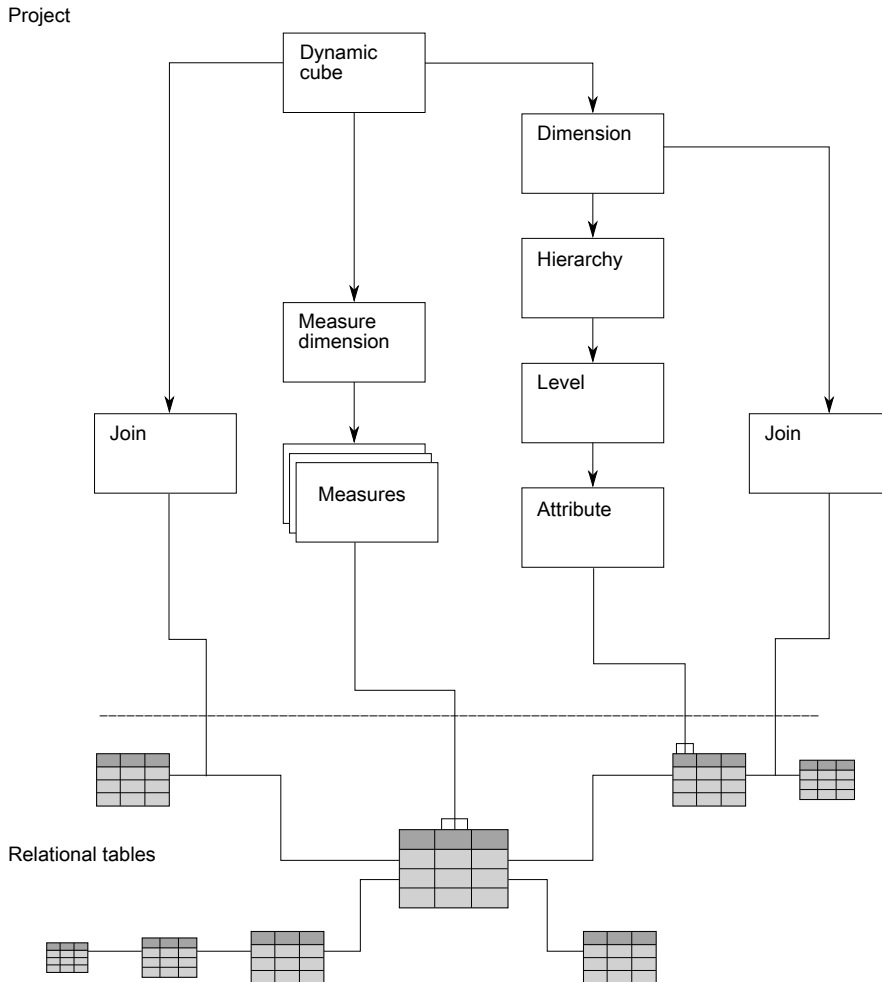


Figure 11. Mapping of items in a dynamic cube to the relational snowflake schema

Measures

In IBM Cognos Dynamic Cubes, you can define regular measures and calculated measures.

Regular measures are mapped directly to a database column of numerical data or defined by an expression. If defined by an expression, the expression is constructed from relational metadata and cannot include dimensional constructs and functions.

Calculated measures are computed in the context of a dynamic cube and computed in the dynamic query server. The expression is constructed from cube metadata and uses dimensional constructs and functions. Dimensional expressions are required when it is necessary to traverse hierarchical relationships or compute complex calculations which are difficult or impossible with relational expressions. With dimensional expressions, you have the ability to access parent/child relationships, to calculate parallel periods, to use set operations and to define an expressions which are evaluated based on its context within a query.

There are some behavior similarities between calculated measures and calculated members. For information about calculated members, see [“Calculated members”](#) on page 73.

In Cognos Dynamic Cubes, a measure dimension, containing a set of measures, is used in a dynamic cube as the center of a star schema. The physical grouping of measures into a single fact table implies that they share one area of interest. Each measure references the attributes that are used in measure-to-dimension

joins. Each measure also references the attributes and joins that are used to map the additional measures across multiple database tables. The value of a measure is meaningful only within the context of the dimensions in a cube. For example, a revenue of 300 has no meaning on its own, but does have meaning in the context of dimensions, such as Region and Time. For example, the revenue for New York in January is 300. Common examples of measures are Revenue, Cost, and Profit.

Simple arithmetic expressions can often be evaluated either by the relational database or in the context of the cube. If a measure expression can be evaluated in either context, it may be preferable to choose a relational expression. Relational databases usually have access to wider range of functions and may be more efficient. If a database is constrained in terms of resources, an alternative is to use calculated measures.

Regular aggregates

Each measure has a regular aggregate. Aggregation rules can be used in addition to the regular aggregate. Aggregation rules define how a measure is aggregated in relation to one or more dimensions. A measure is aggregated by first applying the regular aggregate to all dimensions not specified by aggregation rules, then applying aggregation rules in the order they are listed.

A semi-aggregate measure is a measure that may aggregate differently relative to one or more dimensions within a cube. For example, relative to warehouses, inventory levels are additive. Relative to time, inventory levels are computed as of a point in time. This is typically the first or last occurrence within a time period (first or last day of the month). Therefore, an inventory level measure would have a Regular Aggregate of Sum, and an Aggregation Rule of First or Last relative to the Time dimension.

The **Regular Aggregate** property can have values of Average, Calculated, Count, Count Distinct, Count Non Zero, Custom, Maximum, Median, Minimum, Standard Deviation, Sum or Variance.

The Custom value indicates that the value of the measure is computed by an external business process. Custom measures are a specialized form of non-distributive measure that do not roll up. Values must exist in the measure or aggregate tables at the precise level of aggregation required for a query, otherwise the values are shown as Null. You can customize measure values using advanced business logic and make those values available in IBM Cognos Analytics.

The Calculated value controls the order of operations for calculations. When you use a **Calculated** Regular Aggregate, IBM Cognos Dynamic Cubes first aggregates each measure in the expression using its Regular Aggregate property. Then, it uses the values of the aggregated measures to calculate the expression.

Use Sum and Count aggregates rather than Average where possible. You can also use simple calculations by selecting a measure and assigning a rule, such as Average.

Table 10. Sample data for calculated Regular Aggregate example

| Location | Time | Sales | Average Returns |
|----------|------|-------|-----------------|
| USA | Q1 | 10 | 2 |
| USA | Q2 | 30 | 4 |
| USA | Q3 | 50 | 6 |

Sales is defined with a Regular Aggregate of Sum. Average Returns are defined with a Regular Aggregate of Average.

In this example, the calculated measure, Measure A, is defined by the expression (Sales - Average Returns).

If Measure A is assigned a Regular Aggregate value of Sum, its value is computed as follows if grouping by distinct values of Location.

10 - 2 = 8
 30 - 4 = 26
 50 - 6 = 44

```
-----  
Measure A  8+ 26 + 44 = 78
```

If Measure A is assigned a Regular Aggregate value of Calculated, its value is computed as follows if grouping by distinct values of Location.

```
Sales          10 + 30 + 50 = 90  
Average Returns (2 + 4 + 6) / 3 = 4  
-----  
Measure A  90 - 4 = 86
```

Aggregation rules

Each measure has a regular aggregate. Aggregation rules can be used in addition to the regular aggregate. Aggregation rules define how a measure is aggregated in relation to one or more dimensions. A measure is aggregated by first applying the regular aggregate to all dimensions not specified by aggregation rules, then applying aggregation rules in the order they are listed.

Aggregation rules can be

- Distributive (Count, Sum, Maximum, Minimum)
- Non-distributive (Average, Standard Deviation, Variance)
- Time state (First, Last, Current Period)

Distributive measures can be aggregated from one level to the next. Existing aggregate values can be used to compute higher level aggregates. Non-distributive measures must be calculated from the base fact table data. They cannot be aggregated from one level to the next.

Non-distributive measures

Non-distributive measures must always be aggregated from the detail fact table grain and cannot be aggregated from one summary level to the next.

A non-distributive measure is a measure that is defined with a non-distributive aggregation rule such as:

- Count Distinct
- Average
- Standard Deviation
- Variance

Aggregate tables can be used only if they are calculated from the exact group of levels of the SQL query. If none of the aggregate tables exactly match the required roll ups, the aggregate value must be computed from the fact table. As a result, higher level aggregations of non-distributive measures against a large fact table can take longer to calculate than measures that can take advantage of external aggregate tables

A dynamic cube stores the values of non-distributive measures in its data cache for later use.

When calculating summary values in a query, non-distributive measures require a separate SQL query for each summary. These summary values are query-specific and are not stored in the data cache.

For a cross tab report with row/column summaries, each summary requires a separate SQL query which, depending upon the underlying database, can have an impact on query performance.

Unlike non-distributive measures, distributive measures can always be aggregated from one level to the next. For example, the sum of Sales for a quarter can be calculated by summing monthly sales data.

Time state aggregation rules

The **First**, **Last** and **Current Period** aggregation rules represent the state of a measure at specific times. They are commonly used in inventory or account balances. There are several points to consider when using time state aggregation rules:

- Time state aggregation rules are computed at the granularity of the fact table. If the fact table is at a lower level of granularity than the dimension associated with the measure with a **First**, **Last**, or **Current Period** aggregation rule, IBM Cognos Cube Designer issues a warning.
- Aggregation cannot be computed correctly across multiple cubes. If a virtual cube contains a measure in which the underlying base measures have aggregation rules, Cognos Cube Designer issues a warning. The warning is issued only if the base cubes containing the aggregation rules exists in the project model.
- If there is no value associated with the appropriate leaf level member of the aggregation rule, the value of the measure is NULL.
- Time state aggregation rules are not affected by member security.
- Time state aggregation rules are not affected by attribute security.
- Time state aggregation rules are not supported for parent child hierarchies.
- If the dimension upon which the time state aggregation rule is based is secured for a user, the value for the measure is computed as that for the default member of the dimension as per the rules established for dimension security.

Errors must be corrected to be able to publish the cube. Warnings are informational, and do not prevent the cube from being published.

First

The **First** aggregation rule provides the measure value associated with the first leaf level descendant of the current member of the dimension for which the semi-aggregation rule is defined. For example, a time hierarchy contains years, quarters, and months, and you are examining data at the quarter level. For each quarter, the First rule reports the measure value from the first month of the quarter. When you examine data at the year level, the rule reports the first value from the first month in the first quarter of each year.

Last

The **Last** aggregation rule provides the measure value associated with the last leaf level descendant of the current member of the dimension for which the semi-aggregation rule is defined. For example, if a time hierarchy contains years, quarters, and months, and you are examining data at the quarter level, for each quarter, the Last Period rule reports the measure value from the last month of each quarter. When you examine data at the year level, it reports the value from the last month in the last quarter of each year.

Current Period

The **Current Period** aggregation rule provides the measure value associated with the leaf level descendant of the current member of the Time dimension that corresponds to the Current Period relative time member. If the current period is not a descendant of the current member, it provides the value of the Last leaf level descendant. For example, a time dimension contains years, quarters, and months, and Quarter 1 starts in January. The current period is set to April 2007. At the year level, the **Current Period** option reports the measure value for April 2007. At the quarter level, the option reports the measure value for April in Quarter 2 because April is the current period, but it shows the value of the last active month in every other quarter; that is, March for Quarter 1, September for Quarter 3, and December for Quarter 4.

The **Current Period** aggregation is only supported when it is defined relative to a dimension identified as a time dimension. The associated dimension must be a time dimension and each of the hierarchies in the time dimension must have the relative time property enabled.

Relative time and security cannot be enabled on a hierarchy at the same time. Therefore, **Current Period** is not supported on a secured time hierarchy.

Time state aggregation rules with multi-hierarchy dimensions

For a measure with a time state aggregation rule relative to a multi-hierarchy dimension, tuple values are computed according to the following rules:

Rule 1:

If a tuple has a non-ALL member from any hierarchy of a multi-hierarchy dimension, only the non-ALL members are resolved to the corresponding leaf level member for the time state aggregation rule.

For example, the Time dimension has two hierarchies Time.Actual and Time.Fiscal. Both have ALL members. The Closing Inventory measure has aggregation rule of Last.

The tuple (Closing Inventory, Time.Actual.ALL, Time.Fiscal.2012) resolves to: (Closing Inventory, Time.Actual.ALL, Time.Fiscal.2013Jan). The result is Closing Inventory for 2013Jan because the aggregate rule is Last and 2013Jan is the last month of fiscal year 2012.

The tuple (Closing Inventory, Time.Actual.2012, Time.Fiscal.2012) resolves to (Closing Inventory, Time.Actual.2012Dec, Time.Actual.2013Jan). The result is null because the Time members resolve to different months which preclude any fact data.

Rule 2

If a tuple only projects ALL members from a multi-hierarchy dimension, only the default member of default hierarchy is resolved for the time state aggregation rule.

For example, hierarchies Time.Actual and Time.Fiscal, both have ALL members. Time.Actual.ALL is the default member of default hierarchy.

The tuple (Closing Inventory, Time.Actual.ALL, Time.Fiscal.ALL) resolves to (Closing Inventory, Time.Actual.2012Dec, Time.Fiscal.ALL). The Result is Closing Inventory for 2012Dec

Virtual cubes

In IBM Cognos Dynamic Cubes, a virtual cube consists of two merged cubes. You can merge cubes by using the following combinations:

- Merge two source cubes.
- Merge two virtual cubes.
- Merge one source cube with one virtual cube.

By combining two virtual cubes, or one source cube with a virtual cube, you can merge more than two cubes into a single virtual cube.

Some advantages of using virtual cubes include the following points:

- Virtual cubes use less memory than physical cubes.
- There is reduced cube refresh latency.
- You can add volatile information to a lookup cube.
- You can join cubes to present consolidated data and provide more sophisticated calculations.
- Each source cube can be derived from a separate data source.

A virtual cube must contain the following objects:

- A virtual measure dimension that contains one or more virtual measures.
- At least one virtual dimension that contains one or more virtual hierarchies.

It can also contain virtual calculated measures, and virtual calculated members.

When you create a virtual cube, the following objects are added, if they exist in at least one source cube:

- Dimensions
- Hierarchies
- Measures
- Levels
- Members

Virtual dimensions and hierarchies

Any dimensions and hierarchies with identical names in the source cubes are known as conformed dimensions and conformed hierarchies. These objects are added to the virtual cube as merged virtual dimensions and virtual hierarchies.

For example, two source cubes with a Time dimension are merged into a virtual dimension also named Time.

Any dimensions and hierarchies that do not have identical names, or that exist in only one of the source cubes, are known as non-conformed dimensions and non-conformed hierarchies. These objects are added to the virtual cube as new virtual dimensions and virtual hierarchies.

For example, if source cube 1 contains a Sales Q3 hierarchy, and source cube 2 contains a Sales Q4 hierarchy, the dimensions are not merged because the names do not match. Instead, two virtual hierarchies, Sales Q3 and Sales Q4, are added to the virtual cube.

If a virtual cube contains a non-conformed hierarchy, the virtual cube queries both source cubes to retrieve data only if one of the following conditions is met:

- The non-conformed hierarchy is deleted from the virtual cube.
- The virtual hierarchy includes an All member and the query includes this member.

This can occur if the All member is referenced explicitly in the query or if the All member is the default member.

If neither of these conditions is met, the virtual cube queries only the source cube with the non-conformed hierarchy, and never the second source cube.

Virtual measures

Any measures with identical names in the source cubes are added to the virtual cube as merged virtual measures. Any measures that do not have identical names, or that exist in only one of the source cubes, are added to the virtual cube as new virtual measures.

Important: It is possible to merge measures only when the regular aggregate is one of the following: Sum, Maximum, Minimum, or Count. It is not possible to merge non-distributive measures or a distributive measure with an aggregation rule applied.

When merging measures from two source cubes, if there is a conflict between the data format of each measure, the data format of the merged virtual measure is set to * or unknown. For example, if a measure in source cube 1 has a US currency data format, and a measure in source cube 2 has a UK currency data format, the data format cannot be merged.

Virtual levels

Source cubes containing identical levels in a hierarchy (same number of levels and identical names), are merged as virtual levels. If levels in the source cubes are not identical, level names from the first source cube are used as the names of the virtual levels. If one source cube contains more hierarchy levels than the second source cube, the extra levels are added as the lowest levels of the virtual hierarchy.

For example, source cube 1 contains a Time hierarchy with Year, Quarter, and Month levels. Source cube 2 also has a Time hierarchy with Year, Month, Day, and Time levels. When they are merged, a Time virtual hierarchy is created with Year, Quarter, and Month, and Time virtual levels with the following members:

- The Quarter virtual level contains Quarter members from source cube 1 and Month members from source cube 2.
- The Month virtual level contains Month members from source cube 1 and Day members of source cube 2.
- The Time virtual level contains Time members from source cube 2.

Virtual members

For a virtual hierarchy that is merged from two conformed dimensions, all hierarchy members from the source cubes are available as virtual members. If the level key for each source member is identical, members are added to the virtual cube as merged virtual members. Any members that do not have matching level keys are added to the virtual cube as new virtual members.

Tip: To browse virtual members, ensure that each source cube is deployed as data source to the content store and started.

Calculated measures and calculated members

Calculated measures and calculated members from source cubes are not added to a virtual cube. To use calculated measures or members from source cubes, you must manually define them in the virtual cube.

For more information, see [“Calculated members” on page 73](#).

In-database aggregates

In-database aggregates are unavailable in a virtual cube because a virtual cube can retrieve data only from source cubes, not by querying a data source.

Support for multiple locales

If source cubes include support for multiple locales, a virtual cube also has multiple locale support.

A virtual cube automatically supports all locales defined in the source cubes. For example, in source cube 1, English and French are defined as supported locales. In source cube 2, English and Japanese are defined as supported locales. In the virtual cube, English, French and Japanese are included as supported locales.

A virtual cube also supports the use of multilingual names and captions for a virtual cube, virtual dimensions, virtual hierarchies, virtual levels, and virtual measures. However, with the exception of the All member caption, multilingual names and captions from source cubes are not automatically added to a virtual cube. To use multilingual names and captions from source cubes, you must manually define them in the virtual cube.

Manual merging of source objects

It is possible to manually merge objects in a virtual cube that could not be merged automatically. For example, source cube 1 contains a Time dimension and source cube 2 contains a Fiscal Time dimension. They are not merged, so two virtual dimensions Time and Fiscal Time are added to the virtual cube. If both dimensions contain the same structure and data, you could manually merge them into one virtual dimension named Time. You could then delete the redundant Fiscal Time virtual dimension.

You cannot reference a source object more than once in a virtual cube. For example, if the Time source hierarchy is used in the Time virtual hierarchy, it cannot also be used in the Fiscal Time virtual dimension.

Virtual cube scenarios

Common scenarios for using virtual cubes are described here. You can combine these scenarios based on your specific needs.

Cubes with partitioned data

Sales information for a large region is stored in two cubes. Fact data for each cube can originate from a single fact table or two separate fact tables. One cube, `WestSales`, stores sales information for the west region, and the other cube, `EastSales`, stores sales information for the east region. `WestSales` and `EastSales` have the same structure. To provide a combined view of the sales data, you can define a virtual cube `AllSales` to merge the two regional cubes.

Cubes with pre-cached historical data and current data

Sales information is stored in a single cube called `AllSales`. The cache of this large cube must be rebuilt frequently to reflect the updates in the database. The rebuilding process usually takes a long time.

To address this problem, you can split `AllSales` into two cubes: one to record the historical sales information (`HistoricSales`), and another to record the daily sales information for the current month (`CurrentMonthSales`). You can then define a virtual cube called `VirtualSales` to join these two cubes. By reorganizing the cubes this way, performance is improved in the following ways:

- Because you refresh data only for `CurrentMonthSales`, cube refreshing performance is improved.
- Because query results from `HistoricSales` are pre-cached, and `CurrentMonthSales` is small in size, performance for queries run against the sales data of the entire time period is improved.
- Because of the smaller size of `CurrentMonthSales`, performance for queries run against the sales data of the current month is improved.

Cubes with shared dimensions

Sales information is stored in a single cube called `GlobalSales`. You need to convert some sales figures into other currencies. You could add exchange rates to this cube, but the cube might contain redundant data and would be hard to maintain.

Instead, you can create a cube called `ExchangeCurrency` to store the exchange rates, and define a virtual cube `SalesConversion` to perform currency conversion for the sales data. `GlobalSales` and `ExchangeCurrency` share some dimensions but do not have the same structure.

In-database aggregates

In IBM Cognos Cube Designer, you can model in-database aggregates within a dynamic cube when the imported data source for a dynamic cube contains fact tables with pre-aggregated data.

IBM Cognos Dynamic Cubes supports the use of in-database aggregates created in a dynamic cube and rewrites queries to use the underlying aggregate tables whenever possible. For information about in-database aggregate modeling, see [“Modeling in-database aggregates” on page 93](#).

Aggregate tables

Although it is a best practice to store the lowest level of data in a detail fact table in a data warehouse, selected data can be summarized in a separate table known as an aggregate table. An aggregate table contains detail fact data that is aggregated at a higher level relative to one or more of the dimensions associated with the data.

Using aggregates is critical to achieving performance over large scales for the following reasons:

- It allows you to use pre-calculated data from a data warehouse.
- It decreases the amount of data required to be accessed from the data warehouse.

Some database vendors use special table types for aggregate tables. For example, IBM DB2® uses Materialized Query Tables (MQTs) and Oracle uses Materialized Views. The relational database understands that these special tables are aggregates and routes to them for performance if the database can determine they are applicable and faster. The aggregate awareness feature in Cognos Dynamic Cubes can also use these tables so that a dynamic cube routes to these aggregates tables rather than relying on the database to do the routing.

To increase performance, more than one aggregate table can be necessary in a schema. However, if an aggregate table summarizes data at too high a level within one or more hierarchies, the aggregates can be applicable to only a few queries. In addition, if many dimensions are used, it can be difficult to design frequently used aggregate tables.

When creating aggregate tables, refer to the documentation for your database for information about creating a data warehouse, in particular indexing your data, and co-locating fact and dimension tables. Cognos Dynamic Cubes supports these concepts:

- Sharing common dimension tables if fact and aggregate tables are co-located in the same storage space.
- Use of separate dimension tables for aggregate tables (co-locating dimension and fact data).
- Inclusion of dimension level keys entirely within an aggregate table to avoid joins to dimension tables.
- Partitioning of data.

In-database aggregates

In-database aggregates are aggregate tables that a database administrator can create and apply to the database. After the database has been updated, a modeler must model an in-database aggregate for each created aggregate table in the database and redeploy the dynamic cube to the content store.

In-memory aggregates

In-memory aggregates are aggregate tables that can be applied by the IBM Cognos Analytics server the next time the cube is started. These aggregates are stored in the content store.

Aggregate Advisor

Aggregate Advisor is an external tool, available with IBM Cognos Dynamic Query Analyzer, that can analyze the underlying model in a dynamic cube data source and recommend which aggregates to create. These aggregates can be created both in-database and in-memory.

Aggregate Advisor can also reference a workload log file that allows it to suggest aggregate tables (in-database or in-memory) that correspond directly to the reports contained in the log file.

The Aggregate Advisor does not include recommendations for the following types of measures:

- Calculated measures

The Aggregate Advisor recommends aggregates to accelerate queries that are processed by the underlying database. Because calculated measure expressions are processed in the dynamic query engine, there are no corresponding aggregate recommendations for these types of expressions.

- Semi-aggregate measures

Semi-aggregate measures are not supported by the aggregate cache. However, you can model an in-database aggregate to an existing in-database aggregate with a semi-aggregate measure. If there is an exact match between a query and an in-database aggregate with a semi-aggregate measure, the dynamic query engine routes the query to the corresponding in-database aggregate.

- Measures with **Regular Aggregate** type of Standard Deviation, Median, Variance, or Unknown.

Because these aggregate types are processed by the dynamic query engine, there are no corresponding aggregate recommendations for these types of measures

For more information on using Aggregate Advisor, see the *IBM Cognos Dynamic Query Analyzer User Guide*.

Chapter 5. Getting started with Cognos Cube Designer

IBM Cognos Cube Designer is the modeling tool provided with IBM Cognos Dynamic Cubes. You use it to build dynamic cubes and publish them for use in the IBM Cognos studios.

To get started, you import metadata from a relational database. Using the metadata, you model dynamic cubes and save the cube definitions in a project. After you publish the cubes, they are listed as data sources in Content Manager and their related packages are available to report authors.

Note: Administrator privileges are required for the account that is used to run Cognos Cube Designer.

Introduction to Cognos Cube Designer

IBM Cognos Cube Designer is the application that you use to model dimensional metadata and dynamic cubes. The **Data Source Explorer** tree, the **Project Explorer** tree, the object editors, and the **Properties** pane are the main parts of the Cognos Cube Designer user interface.

Getting Started page

The **Getting Started** page is shown when you start Cognos Cube Designer. You can also display this page at any time by clicking **Show the Getting Started Page** from the **Help** menu.

You can perform the following tasks:

- Click **Create New from Metadata** to import metadata into a new project.
For more information, see [“Import metadata” on page 39](#).
- Click **Create New Blank Project** to create a project.
For more information, see [“Managing a project” on page 43](#).
- Click **Open Existing** to open a project.
For more information, see [“Managing a project” on page 43](#).

Data Source Explorer

The **Data Source Explorer** shows the metadata that is imported from relational data sources. You can view the columns, keys, and joins by expanding a table in the **Data Source Explorer** tree.

You can perform the following tasks:

- Right-click a table and select **Explore Metadata** to view a graphical representation of the metadata in the **Relational Explorer Diagram** tab.
You can view the columns in a table, the primary key and foreign keys, and its joins to other tables.
- Right-click a table and select **View Data** to view sample data from the data source in the **Tabular Data** tab.
Data is retrieved from the data source and shown in IBM Cognos Viewer.
- Right-click a fact table and select **Generate, cube with basic dimensions** or **Generate, cube with dimensions using data sampling** to create a dynamic cube.

Use one of these options to create a dynamic cube that is based on a fact table in the data source. The cube, including all required dimensional metadata, is added to the project in the **Project Explorer**. For more information about creating cubes, see [“Model a dynamic cube” on page 61](#).

Relational Explorer

The **Relational Explorer Diagram** shows a graphical view of your data source metadata. Use the **Relational Explorer Diagram** to explore your metadata and view the relationships between objects.

Tip: When this tab is visible, you can drag tables from the **Data Source Explorer** tree to explore them.

Project Explorer

The **Project Explorer** shows all the dimensional metadata definitions and dynamic cube definitions included in a project. Use the **Project Explorer** tree to add objects to your dynamic cubes, access the object editors, and publish your cubes.

You can perform the following tasks:

- Model dimensions and hierarchies.

For more information, see [Chapter 6, “Dimensional metadata modeling,” on page 45.](#)

- Model dynamic cubes

For more information, see [“Model a dynamic cube” on page 61.](#)

- Right-click and select **Validate** to validate an entire project or an individual object.


For more information about validation, see [“Validate a project and individual objects” on page 44.](#)

- Right-click a cube and select **Publish** to deploy the cube and, optionally, publish a package to be used by report authors.

For more information about publishing, see [“Deploying and publishing dynamic cubes” on page 69.](#)

Tip: When you add a dynamic cube to a project, the data source on which it is based is added to the **Data Sources** folder in the **Project Explorer** tree. You can view the database catalog and schema that is referenced by the data source in the **Properties** tab.

Functions tab

From the **Functions** tab , you have access to the operators, summaries, constants, and functions that you use in expressions.

Object editors

There is an editor available for each object. When an editor tab is visible, you can also access other functionality that is related to the object. For example, when viewing the cube editor, you have access to the **Aggregates**, **Security**, and **Implementation** tabs.

To access an editor and its related tabs, right-click the object in the **Project Explorer** tree and select **Open Editor**.

Tip: To keep multiple editor tabs accessible, right-click the tab and select **Pin**. Because some of the editor windows are similar in appearance, verify your edit location on the tab.

Implementation tab

The **Implementation** tab shows a physical diagram of the current object. For example, to view the implementation of an entire cube, right-click the cube in the **Project Explorer** tree tab, select **Open Editor**, and then select the **Implementation** tab. For some objects, you can also add or edit relationships between the cube objects. Select an object and click to use menus to explore the diagram.

Object properties

On the **Properties** tab, you can view and edit the properties of an object.

To access the properties of an object, select the object in the **Project Explorer** tree. For more information about object properties, see [Chapter 6, “Dimensional metadata modeling,” on page 45](#) and [“Model a dynamic cube” on page 61](#).

Validation issues

The **Issues** tab shows modeling errors and warnings for objects that must be fixed to validate them.

The **Performance Issues** tab shows a list of all the performance issues for objects. These issues affect how well a dynamic cube performs when it is published and started.

You can view validation issues for all objects in a project or for an individual object. Select the project or object in the **Project Explorer** tree and then click the **Issues** tab. For more information about validating objects, see [“Validate a project and individual objects” on page 44](#).

Import metadata

You import metadata to use it as the basis for modeling dimensional metadata and dynamic cubes.

Remember: You must ensure that the data source from which you import metadata supports the dynamic query mode.

You can import metadata from the following sources:

- A Content Manager data source.

Select this option to import metadata from a relational data source that is defined in IBM Cognos Analytics. For more information, see [“Importing metadata from a Content Manager data source” on page 39](#).

- A Framework Manager package.

Select this option to import metadata from an IBM Cognos Framework Manager package that is published to the IBM Cognos Analytics content store. For more information, see [“Importing metadata from a Framework Manager package” on page 40](#).

- A Cubing Services model.

Select this option to import cube metadata from an IBM InfoSphere Warehouse Cubing Services model. IBM Cognos Cube Designer creates a separate dynamic cube definition for each cube that is contained in the InfoSphere Warehouse Cubing Services cube model. For more information, see [“Import InfoSphere Warehouse Cubing Services cube metadata” on page 41](#).

Tip: If you want to browse hierarchy members in a data source when modeling dynamic cubes, before you import the metadata, check whether a data source connection exists that contains a subset of the metadata. Using smaller volumes of metadata can speed up the modeling process.

Importing metadata from a Content Manager data source

If you want to model dimensional metadata and dynamic cubes based on a relational database, you import the metadata from a Content Manager data source.

You import metadata from one schema at a time. You must perform a separate import for each schema you want to use.

A separate file is created for each data source from which you import metadata. These files are stored in the *installation_location\data* directory.

A dynamic cube is modeled using a single data source only. A project can contain many dynamic cubes, and if you have imported multiple data sources, each dynamic cube can be derived from a separate data source.

Important: The following data sources are not supported as sources of metadata for dynamic cubes:

- MySQL
- MemSQL

- Google Cloud SQL MySQL
- Amazon Aurora MySQL
- Microsoft Azure MySQL
- Denodo
- MariaDB

Before you begin

Check the following prerequisites:

- The data source contains a star or snowflake schema.
- The data source connection to the database uses a Java Database Connectivity (JDBC) driver. This is required by the dynamic query mode.
- The data source is created in IBM Cognos Analytics. If not, you must create it first. For more information, see the *Managing IBM Cognos Analytics* guide or the *IBM Cognos Analytics Administration and Security Guide*.

Procedure

1. From the **Start** menu programs, click **IBM Cognos Cube Designer**.

You can also start Cognos Cube Designer from IBM Cognos Framework Manager. From the **Tools** menu, select **Run Cube Designer**.

2. From the toolbar, click **Get Metadata**.
3. Click **Browse Content Manager Datasource**.
4. Select the database schema from which to import data, and then click **OK**.

The imported metadata is shown as a list of database tables in the **Data Source Explorer** tree.

Tip: If your project contains more than one imported data source, each data source is shown in a separate panel.

You can now model dimensional metadata and dynamic cubes.

5. When you finish working, click **Save** .

Importing metadata from a Framework Manager package

You can import Framework Manager packages into IBM Cognos Cube Designer to use the metadata in the dimensionally modeled relational (DMR) and relational models in the packages to create dynamic cubes. Regardless of what is contained within the Framework Manager model, the metadata from the model that is used to create a dynamic cube must represent a star or snowflake schema.

Important: The reports that are based on a DMR model are not migrated to the dynamic cube model that is based on the DMR model.

Before you begin

The packages that you want to import must be published the Cognos Analytics content store. You cannot import packages that are saved on a disk.

About this task

The dynamic cube model that is created in Cognos Cube Designer is based on the physical metadata from the original Framework Manager model. If the cube model is closed and then re-opened, the association between the Framework Manager model and the cube model is lost. In this situation, you might need to re-open the Framework Manager model and re-import the metadata. In the **File** menu, you can see the

imported packages that were used recently. The packages, unlike the Cognos Cube Designer models that are also shown here, do not include the directory path and the .fmd extension.

Procedure

1. From the **Start** menu programs, click **IBM Cognos Cube Designer**.
2. From the toolbar, click **Get Metadata > Select Framework Manager Package**.
3. Select the package from which to import data, and click **OK**.

The package metadata is displayed in the **Source** tree. It includes the Framework Manager model metadata and the data sources that are referenced in the model. The Framework Manager metadata includes all objects in the model that are imported, such as measure dimensions, dimensions, shortcuts, query subjects, filters, calculations, and parameter maps. Hidden objects are also imported. This view corresponds closely to the view of the model in Framework Manager.

4. Import objects from the **Source** pane to the project area by using the import menu options. In the **Source** pane, right-click the object that you want to move and choose one of the following import options:
 - For query subjects, click **Import > As a Dimension** to import the object as a regular dimension, or **Import > As the Measure Dimension of the New Cube** to import the object as a measure dimension.
 - For dimensions, click **Import**. Depending on their type, the dimensions are automatically imported as regular or measure dimensions. The hierarchies, levels, and level attributes of the time dimension are also automatically imported.
 - For namespaces and folders, click **Import as Cubes**.

A cube is created for each imported measure dimension. Dimensions that have scope relationships on the measure dimension are discovered and a dimension is created for each of those dimensions. Every dimension that has a scope relationship to a measure dimension is included in the cube. A conformed dimension is included into each cube that it belongs to. Parameter maps are imported automatically.

Tip: You cannot model a cube that is based on the model query subjects.

5. In the **Project Explorer** pane, right-click the model name and click **Validate**.
Very likely, errors are reported. Resolve the errors using the dynamic cube modeling practices.
6. When you finish working, click **Save**.

Import InfoSphere Warehouse Cubing Services cube metadata

You can import cube metadata from an IBM InfoSphere Warehouse Cubing Services model. IBM Cognos Cube Designer creates a project with a separate dynamic cube for each cube that is contained in the imported model.

Cognos Cube Designer retains the basic structure of imported cubes and dimensions when you import cube metadata, but there are some differences in underlying InfoSphere Warehouse Cubing Services models that can cause issues during import. The following table describes these issues and suggestions to work around them.

| <i>Table 11. Import issues and suggested solutions</i> | |
|---|--|
| Issue | Workaround |
| InfoSphere Warehouse Cubing Services models use name-based member unique names (MUNs) to identify members whereas IBM Cognos Dynamic Cubes uses key-based MUNs. | In Cognos Dynamic Cubes, create the MUN expressions by using the supported Cognos expression syntax. |

Table 11. Import issues and suggested solutions (continued)

| Issue | Workaround |
|---|--|
| In InfoSphere Warehouse Cubing Services models, it is possible to create a dimension with multiple hierarchies and reference one single hierarchy in a cube. Cognos Dynamic Cubes does not support hierarchy selection, so all hierarchies are included by each cube that references the dimension. | In Cognos Dynamic Cubes, make a copy of the dimension and delete the unwanted hierarchies. You can then reference the new dimension in a dynamic cube. |
| Cognos Dynamic Cubes does not support shared attributes. As a result, only the first level that references the attributes contains the attributes. The other levels remain empty. | Delete the empty levels and, where appropriate, create the required attributes by dragging columns to the required levels. |
| In InfoSphere Warehouse Cubing Services models, expressions are created by using SQL. Cognos Dynamic Cubes converts the attribute references, but not the expression, to the dynamic query mode. | In Cognos Dynamic Cubes, create the expressions by using the supported Cognos expression syntax. |
| In InfoSphere Warehouse Cubing Services models, if a dimension has an attribute defined that is not part of any level, and the attribute is used to join to a fact table, Cognos Dynamic Cubes incorrectly imports the attribute, adds it to the lowest level, and marks the attribute as hidden. | In Cognos Dynamic Cubes, delete the incorrect attributes manually. |
| In InfoSphere Warehouse Cubing Services models, if the showMembers property is defined for a hierarchy, this property is lost during import. | In Cognos Dynamic Cubes, manually set the Show Extraneous Padding Members property. |
| In Cognos Dynamic Cubes, the default member of a hierarchy is not migrated during import. | Manually set the default member property. |
| Cognos Dynamic Cubes imposes restrictions on certain special characters that are used for cube names and other object names. If an unsupported special character is encountered, an error is shown. | Rename any InfoSphere Warehouse Cubing Services models to remove any unsupported characters before you import them. |
| In InfoSphere Cubing Services models, it is possible to define attributes under a measure dimension that can be used in expressions. Cognos Dynamic Cubes does not support this feature. Attributes are imported as query items in a measure dimension but are flagged as invalid. | Delete the query items in the measure dimension after you import an InfoSphere Warehouse Cubing Services model. |
| If security defined for InfoSphere Cubing Services models, it is lost during import. | In Cognos Dynamic Cubes, set up the required security definitions. |

A log file is also created during the import process that includes details of any objects that cannot be fully imported.

Performing an InfoSphere Warehouse Cubing Services model import

You import cube metadata from an IBM InfoSphere Warehouse Cubing Services model into a project.

Before you begin

Check that the following tasks are complete:

- Ensure that the model is exported from the Design Studio in InfoSphere Warehouse Cubing Services.
- Ensure that the data source that is associated with the InfoSphere Warehouse Cubing Services model has a Java Database Connectivity (JDBC) data source connection defined. This data source connection is required by the dynamic query mode.
- Ensure that the associated data source is defined in IBM Cognos Analytics.

Procedure

1. From the **Start** menu programs, click **IBM Cognos Cube Designer**.

You can also start the Cognos Cube Designer from IBM Cognos Framework Manager. From the **Tools** menu, select **Run IBM Cognos Cube Designer**.

2. From the **File** menu, click **Import Cubing Services Model**.
3. Select the model from which you want to import metadata, and then click **OK**.
4. Select the data source connection that is associated with the InfoSphere Warehouse Cubing Services cube model, and then click **OK**.

Cognos Cube Designer creates a project that contains one or more cubes based on the imported metadata.

If there any issues with the imported metadata, a log file is created and a confirmation message is shown.

5. Click **OK** to acknowledge the message. You can then investigate the issues in the log file.

By default, the log file is stored in `cognos_analytics_location\logs`

You can now continue working in the project.


6. Click **Save**  to save the project.

Managing a project

Dynamic cube definitions are saved in a project. This section describes how to open, edit, and save an existing project.

Tip: It is good practice to save a project at regular intervals.

Procedure

1. From the toolbar, click **Open** .
2. Select the project file (.fmd).
3. Click **OK**.
4. Edit individual objects as required.

For more information, see [Chapter 6, “Dimensional metadata modeling,” on page 45](#) and [“Model a dynamic cube” on page 61](#).

5. When you finish, click **Save** .

Validate a project and individual objects

IBM Cognos Cube Designer automatically validates individual objects as you design them. Modeling issues are identified in the **Project Explorer** with icons shown next to objects that are causing the issues:

- Errors are indicated by a white cross on a red circle.
- Warnings are indicated by a yellow triangle.
- Performance issues are indicated by a gauge.

The **Issues** tab shows a list of all the modeling issues that are related to a selected object. You can click an issue for more details. If a solution is provided, you can resolve the issue by selecting the solution and clicking **OK**. You can also click **Invoke Editor** to access the object editor. Modeling issues affect the validity of a dynamic cube and prevent you from deploying it.

The **Performance Issues** tab shows a list of all the performance issues that are related to a selected object. These issues affect how well a dynamic cube performs when it is published and started. They do not affect the validity of a dynamic cube.

You can validate an entire project or an individual object at any time. Validate frequently and resolve issues as they are reported. If you attempt to model a large cube without validating as you go, you can have a long list of issues to resolve.

You can validate each object as you create it by right-clicking it in the **Project Explorer** and selecting **Validate**.

You cannot deploy a dynamic cube that contains modeling errors. It is possible to deploy a valid cube when the project contains unrelated objects that are not valid.

Chapter 6. Dimensional metadata modeling

You use IBM Cognos Cube Designer to model dimensions, hierarchies and levels.

Model dimensions

With IBM Cognos Cube Designer, you can model commonly used dimensions at the project level and reference them in one or more dynamic cubes. You can also model dimensions within a specific cube.

The following table lists the properties that you can set when modeling a dimension.

| Property | Description |
|----------------------------------|--|
| Name | The dimension name shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. For more information about multiple locales, see “Multiple locales” on page 89 |
| Comment | A comment or description of the dimension. Comments are not visible in the IBM Cognos studios. |
| Default Hierarchy | The hierarchy to use when no hierarchy been specified for a dimension used in an expression. Applies only when multiple hierarchies are defined for a dimension. |
| Multilingual Support | Disabled (default) - Specifies that members do not have multiple locale support. By Column - Specifies that members support multiple locals. For more information about multiple locales, see “Multiple locales” on page 89 . |
| Share member cache for all cubes | If enabled, specifies that shared dimensions can have a shared member cache. Creating a shared member cache improves performance by reducing the amount of memory consumed when cubes are published. Default: Disabled (false) For more information, see “Defining a shared member cache” on page 47 . |
| Dimension Type | Regular (default) - Identifies a regular dimension. Time - Identifies a time dimension. For more information about relative time dimensions, see “Defining a relative time dimension” on page 86 . Important: Relative time dimensions are not supported for DMR modeling. |

Defining a dimension

In IBM Cognos Cube Designer, you can model commonly used dimensions at the project level and reference them in one or more dynamic cubes. You can also model dimensions within a specific cube.

When you add a dimension, it contains an initial set of objects that you need to complete the dimension. When you validate the dimension, you can use information from the **Issues** tab to help you complete the dimension definition.

Procedure

1. Select the location from which you want create the dimension:

- To create a shared dimension at the project level, select **Model** from the **Project Explorer** tree.
- To create a dimension that this automatically linked to a dynamic cube, select the cube from the **Project Explorer** tree.

The dimension is also shared at the project level.

Tip: Use folders and namespaces to organize objects. Using folders and namespaces makes it easier for you to locate objects and view the structure of a project in the **Project Explorer**.

2. Click **New Dimension** .

The dimension contains a set of initial objects you can use to complete the dimension.

3. To create additional hierarchies, click **New Hierarchy** .

4. To create additional levels, click **New Level** .

5. On the **Properties** pane, set the default hierarchy.

6. To access the dimension editor, right-click a dimension from the **Project Explorer** tree and select **Open Editor**.

7. Change the order of the levels by clicking **Move Up**  and **Move Down** .

What to do next

To complete the dimension, you must complete the definition of each hierarchy and level that belongs to the dimension. For more information, see [“Defining a hierarchy” on page 49](#) and [“Defining a level ” on page 51](#).

Tip: Right-click a relational table and select **Explore Metadata**. You can use the **Relational Explorer Diagram** to help you understand the structure of the metadata used to design the hierarchies and levels.

When you finish modeling a dimension, you can perform the following tasks:

- Browse members from the data source. For more information, see [“Browsing members” on page 55](#).
- Add a shared dimension to a dynamic cube by dragging and dropping it onto the dynamic cube in the **Project Explorer** tree.

Related tasks

[Defining a hierarchy](#)

[Defining a level](#)

[Defining a parent-child hierarchy](#)

Defining a dimension based on a relational table

In IBM Cognos Cube Designer, you can generate commonly used dimensions at the project level and reference them in one or more dynamic cubes. You can also generate dimensions within a specific cube.

Generate, dimension using data sampling applies a heuristic algorithm that interprets relationships among the data to identify levels. Based on the data in the selected table, a hierarchy of levels is generated, based on the cardinality of the data and the column names.

If your data is clean and complete, the generated levels are more accurate. The algorithm does not detect multiple hierarchies.

Procedure

1. Select the location from which you want create the dimension:

- To create a shared dimension at the project level, select **Model** from the **Project Explorer** tree.
- To create a dimension that this automatically linked to a dynamic cube, select the cube from the **Project Explorer** tree.

The dimension is also shared at the project level.

Tip: Use folders and namespaces to organize objects. Using folders and namespaces makes it easier for you to locate objects and view the structure of a project in the **Project Explorer**.

2. Click **Generate, dimension with data sampling**.

What to do next

Review the generated dimension definition and, if required, manually modify it to reflect how you want to view your data.

Tip: Right-click a relational table and select **Explore Metadata**. You can use the **Relational Explorer Diagram** to help you understand the structure of the metadata that is used to design the hierarchies and levels.

When you finish modeling a dimension, you can perform the following tasks:

- Browse members from the data source. For more information, see [“Browsing members”](#) on page 55.
- Add a shared dimension to a dynamic cube by dragging and dropping it onto the dynamic cube in the **Project Explorer** tree.

Defining a shared member cache

If a project contains dimensions that are referenced by more than one cube or virtual cube, you can create a shared member cache. This means that each shared dimension is only published once regardless of the number of cubes that reference it. Creating a shared member cache improves performance by reducing the amount of memory consumed when cubes are published.

A shared dimension can include calculated members and relative time members. You can add a shared dimension to security views and security filters defined for a cube. You cannot share a measure dimension.

Procedure

1. From the **Project Explorer** tree, select the required dimension.
2. In the **Properties** tab, set the **Share member cache for all cubes** property to **true**.

Results

When you validate a shared dimension in a virtual cube, IBM Cognos Cube Designer checks whether a dimension can be shared between the source cube and virtual cube. You can check for warnings on the **Issues** tab.

After publishing cubes with shared dimensions, the dimension members are not automatically updated when a member cache is refreshed. This is to prevent all the cubes sharing a dimension from refreshing. If you want to update the dimension members, you must stop all cubes to remove the dimension from the shared dimension cache. You can then publish the cubes again.

Model hierarchies

IBM Cognos Dynamic Cubes supports level-based hierarchies and parent-child hierarchies. A single level-based hierarchy is automatically added when you create a dimension. You can also create multiple level-based hierarchies in a dimension.

For more information, see [“Dimensions”](#) on page 15 and [“Hierarchies”](#) on page 15.

Complete the hierarchy definition using the properties listed in the following table:


| Property | Description |
|---------------------------------|---|
| Name | The hierarchy name shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Comment | A comment or description of the hierarchy. Comments are not visible in the IBM Cognos studios. |
| Multiple root members | <p>False (default) - the hierarchy uses a single root member at the top of the hierarchy. Selecting this option creates the All level at the top of the hierarchy. You can change the default caption of the top level by editing the Root Caption property.</p> <p>True - the hierarchy contains multiple root members. Selecting this option deletes the All level that is automatically created at the top of the hierarchy.</p> <p>If a hierarchy is single root, Cognos Cube Designer generates the root member. Because all members must belong to a level, the root member is in the All level.</p> |
| Add Relative Time Members | <p>False (default) - If the hierarchy belongs to a Time dimension, relative time members are not added to the hierarchy.</p> <p>True - If the hierarchy belongs to a Time dimension, relative time members are added to the hierarchy.</p> <p>For more information, see “Defining a relative time dimension” on page 86.</p> |
| Default Member | <p>The member value to use when evaluating member expressions, where no value is specified for a hierarchy.</p> <p>If the default member is empty, the root member of the hierarchy is used.</p> <p>To set a default member, drag the required member from the Members folder in the Project Explorer tree.</p> |
| Root Caption | The caption of the root member at the top of the hierarchy shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the caption in all supported languages. |
| Parent-Child | <p>False - Indicates that the hierarchy does not use a parent-child structure.</p> <p>This property cannot be edited.</p> |
| Show Extraneous Padding Members | <p>False (default) - collapse multiple paths of padding members under a single member into a single path.</p> <p>True - show multiple paths of padding members for a single member.</p> <p>For more information, see “Extraneous padding members” on page 20.</p> <p>Important: Padding members are not supported for DMR modeling.</p> |

| <i>Table 13. Properties of a hierarchy (continued)</i> | |
|--|---|
| Property | Description |
| Caption of Padding Members | <p>The caption to use for padding members in the hierarchy.</p> <p>Empty (default) - use a Null caption.</p> <p>Parent's caption - use the caption of the parent.</p> <p>For more information, see “Padding members” on page 18.</p> <p>Important: Padding members are not supported for DMR modeling.</p> |

Defining a hierarchy

In IBM Cognos Cube Designer, a single level-based hierarchy is automatically added when you create a dimension. You can also create multiple level-based hierarchies in a dimension.

Procedure

- From the **Project Explorer** tree, select the dimension you want to work with.
 - To create a new hierarchy, click **New Hierarchy** .
 - To access the hierarchy editor, right-click a hierarchy that belongs to the dimension and select **Open Editor**.
- Complete or modify the hierarchy definition using the **Properties** tab. Identify the **Default Member** and **Root Caption** if required.
- Set the **Show Extraneous Padding Members** and **Caption of Padding Members** if required.

For more information, see [“Padding members” on page 18](#).
- If an **All** level is not required, set the **Multiple Root Members** property to **true**.
- To add levels to the hierarchy, drag levels from the **Levels** folder to the hierarchy.

Model levels

In IBM Cognos Cube Designer, each level in a dimension is defined by creating attributes, mapping those attributes to the relational database source and identifying which attributes are level keys.

When you create a hierarchy, an All level is created at the top of the hierarchy. An All level contains a single member that aggregates data from all members in the lower levels of the hierarchy. For example, an All level in a Region hierarchy aggregates data for all cities, in all states, in all regions.

Complete the level definition using the properties listed in the following table:

| <i>Table 14. Properties of a level</i> | |
|--|---|
| Property | Description |
| Name | The level name shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Comment | A comment or description of the level. Comments are not visible in the IBM Cognos studios. |
| Level Type | <p>Identifies whether the level is regular or time-based.</p> <p>Default: Regular</p> |

Table 14. Properties of a level (continued)

| Property | Description |
|----------------|--|
| Current Period | An expression used to define the current period in a time-based level. The value of the expression is compared to the value of the level key attribute at the level. |

Complete the definition of the level attributes using the properties listed in the following table:

Table 15. Properties of an attribute

| Property | Description |
|--------------|--|
| Name | The attribute name shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Comment | A comment or description of the attribute. Comments are not visible in the IBM Cognos studios. |
| Expression | This property is available only for attributes created in the Cognos Cube Designer. |
| Column Name | The name of the associated column in the relational database. If the Multilingual property is true, this value can be set. For more information, see “Adding support for multiple locales to members and attributes” on page 90. |
| Visible | Controls whether the object is visible in the published package. Non-visible objects are typically used to represent intermediate values. These objects are not intended to be used for direct reporting. However a non-visible object is always present in the published package because the object might be needed by other objects in a dynamic cube. Non-visible objects are not displayed in the metadata browser and are removed from the output of reports which contain references to them. For example, a report that references a non-visible object does not include output from that measure. Default: True |
| Data Type | The data type of the associated column in the relational database. This property cannot be edited. |
| Precision | The precision of the associated column in the relational database. This property cannot be edited. |
| Scale | The scale of the associated column in the relational database This property cannot be edited. |
| Multilingual | This property appears only if support has been enabled for multiple locales in the dimension. For more information, see “Multiple locales” on page 89. False (default) - This attribute does not support multiple locales. True - This attribute supports multiple locales. |

The **Level Unique Key** consists of one or more attributes whose values uniquely identify each instance of a level. For more information, see [“Defining a level unique key” on page 51](#).




Member Sort consists of one or more attributes that provide information about the ordering of the members within a level. For more information, see [“Defining the member sort order” on page 52](#).

Defining a level

In IBM Cognos Cube Designer, you define levels to model the relationships in a hierarchy.


For each level, you assign or create attributes, map them to the relational data source, identify level keys and, optionally, define a sort order. You can also hide the attribute in the published package if required.

Procedure

1. From the **Project Explorer** tree, select a dimension, and click **New level** .
2. To access the level editor, right-click the level in the **Project Explorer** tree, and select **Open Editor**.
3. To create an attribute, click **New Attribute** .
Tip: To give the new attribute a more meaningful name, right-click it and select **Rename**.
4. To map a table column to the new attribute, select the required column from the **Data Source Explorer** tree and drop it onto the **Mapping** column.
Tip: You can also create attributes by dropping table columns to the **Attribute** column.
5. Select the attributes assigned to **Member Caption** and, if required, **Member Description**.
For more information about these special attributes, see [“Attributes” on page 24](#).
6. You can define the **Level Unique Key** one of two ways:
 - If the level unique key is a single attribute, select the **Level Unique Key** check box for the attribute.
 - If the level unique key is a composite key, click **Level Key** . For more information, see [“Defining a level unique key” on page 51](#).
7. If required, specify the member sort order. For more information, see [“Defining the member sort order” on page 52](#).
8. To hide an attribute in the published package, change the **Visible** property to false.
9. To assign the level to a hierarchy, select the level and drop it onto the hierarchy in the **Project Explorer** tree.
Tip: You can also assign levels by dropping them into the hierarchy editor.
10. Expand the hierarchy in the **Project Explorer** tree and, if necessary, modify the order of the levels as they appear under the hierarchy.

Defining a level unique key

The **Level Unique Key** consists of one or more attributes whose values uniquely identify each instance of the level.

A level key is meant to uniquely identify each of the members within a level. The first level key shown in the **Level Key** window is the business key and is denoted with the business key icon . The business key is significant as it generates the members. If a level key does not uniquely identify members within a level, then attributes from the current level or parent levels must be used to uniquely identify the members within the level

For example, a City level can use a unique ID as its level key attribute. City names are not unique, so you cannot use the city name attribute as a level unique key. You can include the set of Region name, State




name, and City name attributes as a composite level unique key because the three attributes together uniquely define a city.

Level keys in SQL statements retrieve data values from the database and the corresponding columns are used as the basis for grouping, joining, and filtering. For optimal performance, use an attribute with an integer data type as the level key. Avoid character and text fields. There can be a performance difference between an integer level key and any other numeric type depending on the database system in use. For more information, see “Levels” on page 22.

If the level unique key is a single attribute, select the **Level Unique Key** check box for the attribute.

If there are multiple level key attributes, the first attribute must be the level key for the level. You may have to reorder the attributes to ensure the appropriate attribute is defined as the level key.

Procedure





1. To define a composite level unique key, right-click a level in the **Project Explorer** tree, and select **Open Editor**.
 2. Click **Level Key** .
 3. Select the attributes that together uniquely identify the level.
 4. Change the order of the attributes by clicking **Move Up**  and **Move Down** .
- The first attribute shown in the **Level Key** window must be the level key for the level.

Defining the member sort order

By default, hierarchy members are shown in the order in which they are loaded into a dynamic cube.

You can select one or more attributes that define the sort order of the members within a level. For example, a Month level might have Month ID as a key attribute, Month Name as the caption attribute, and Month Number as an ordering attribute. Month Number is specified as the ordering attribute because Month Number sorts the months by calendar order whereas Month Name sorts the months alphabetically.

Procedure

1. Right-click a level in the **Project Explorer** tree, and select **Open Editor**.
 2. Click **Member Sort** .
 3. Select the required attributes from the **Attribute** column and click **Add**  to add them to the **Sorting** column.
- You can change the sort order by selecting an attribute and clicking **Move Up**  and **Move Down** .
4. To change the direction of sorting for an attribute, click the **Direction** column and select the required option.
 5. Click **OK**.

Model parent-child hierarchies

In IBM Cognos Cube Designer, you model a parent-child hierarchy when the dimension data is based on a recursive relationship and it is not level-based.

For more information, see “Parent-child hierarchies” on page 21.

To model a parent-child hierarchy, you create attributes, map them to the relational data source and identify which attributes represent the parent key and child key. The child key also acts as the member key.

The top level member in a parent-child hierarchy is determined as the member whose parent is Null.

You define a parent-child hierarchy within a parent-child dimension. Be aware of the following constraints:

- A dimension containing a parent-child hierarchy cannot include any other hierarchies.
- The attributes used for the parent key and member key cannot be composite keys.
- A parent-child hierarchy member cannot contain multiple parents.

If the imported data source contains hierarchy members with multiple parents, you can use surrogate keys in the data source to overcome this issue.

To access the parent-child dimension properties, double-click a parent-child dimension from the **Project Explorer** tree.

Complete the parent-child dimension definition using the properties listed in the following table:

| <i>Table 16. Properties of a parent-child dimension</i> | |
|---|---|
| Property | Description |
| Name | The dimension name shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Comment | A comment or description of the dimension. Comments are not visible in the IBM Cognos studios. |
| Default Hierarchy | The parent-child hierarchy defined within the dimension. This property cannot be edited. |
| Multilingual Support | Disabled (default) - Specifies that members do not have multiple locale support. By Column - Specifies that members support multiple locales. For more information about multiple locales, see “Multiple locales” on page 89. |

To access the parent-child hierarchy properties, double-click a parent-child hierarchy from the **Project Explorer** tree.

Complete the parent-child hierarchy definition using the properties listed in the following table:

| <i>Table 17. Properties of a parent-child hierarchy</i> | |
|---|--|
| Property | Description |
| Name | The parent-child hierarchy name shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Comment | A comment or description of the parent-child hierarchy. Comments are not visible in the IBM Cognos studios. |
| Default Member | The member value to use when evaluating member expressions, where no value is specified for a hierarchy. If the default member is empty, the root member of the hierarchy is used. To set a default member, drag the required member from the Members folder in the Project Explorer tree. |

Table 17. Properties of a parent-child hierarchy (continued)

| Property | Description |
|-------------------------|--|
| Root Caption | The caption of the root member shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the caption in all supported languages. |
| Parent-Child | True - Indicates that the hierarchy uses a parent-child structure. This property cannot be edited. |
| Show Data Members | True (default) - show data members for non-leaf members in the hierarchy. False - hide data members for non-leaf members in the hierarchy. For more information, see “Data members” on page 21 . |
| Caption of Data Members | The caption to use for data members in the hierarchy. Empty (default) - use a Null caption. Parent's caption - use the caption of the parent. |

To access properties of an attribute, select the attribute in the **Attribute** column in the parent-child hierarchy editor. For more information about attribute properties, see [“Model levels” on page 49](#).

Defining a parent-child hierarchy

In IBM Cognos Cube Designer, you can model commonly used parent-child hierarchies at the project level and reference them in one or more dynamic cubes. You can also model parent-child hierarchies within a specific dynamic cube.

Procedure

1. Select the location from which you want create the parent-child hierarchy:
 - To create a shared parent-child hierarchy at the project level, select **Model** from the **Project Explorer** tree.
 - To create a parent-child hierarchy that this automatically linked to a dynamic cube, select the cube from the **Project Explorer** tree.

The parent-child hierarchy is also shared at the project level.

2. Click **New Parent-Child Dimension** .

A new parent-child dimension is created with a parent-child hierarchy.

3. Edit the dimension properties in the parent-child **Properties** pane.
4. Open the parent-child hierarchy editor.
5. From the **Project Explorer** tree, drag table columns to the **Attribute** column to create the hierarchy attributes.
6. Select the attributes assigned to the Parent key and Child key.
These attributes are mandatory.
7. Select the attributes assigned to the Member Caption, and Member Description.
The Member Caption attribute is mandatory.
8. If required, specify the member sort order. For more information, see [“Defining the member sort order” on page 52](#).
9. Complete the parent-child hierarchy definition using parent-child hierarchy editor **Properties** pane.

10. If required, edit the properties of the attributes using the attribute editor **Properties** pane.

Browsing members

When you finish modeling a dimension containing a regular hierarchy or parent-child hierarchy, you can browse the dimension members from the data source.

Tip: A dimension must be valid before you can browse its members. If the dimension that you want to browse is contained in a dynamic cube, the cube must also be valid.

When viewing members in the Cognos Cube Designer, relative time members do not reflect the current period expressions defined in a project, but the members can be used in other expressions if desired. Current period expressions are used when the cube is started.

Procedure

1. From the **Project Explorer** tree, select the hierarchy for which you want to browse members.
2. Expand the **Members** folder.

The parent level dimension members are shown.

Tip: Depending on the volume of metadata included in the data source, it can be time consuming to browse the full list of members. You can cancel browsing by pressing the Escape key.

3. Expand a member to view its child members.

Repeat this step to view further child members.

4. If you make changes to a dimension or hierarchy, you must refresh the list of members to browse.

- To refresh members for all hierarchies in a dimension, right-click the dimension and select **Refresh Members**.
- To refresh members in a specific hierarchy, right-click the **Members** folder and select **Refresh**.

Dimension filters

In IBM Cognos Dynamic Cubes, you can create dimension filters to restrict the members available in a published dynamic cube when a dimension contains more data than is required by the cube.

For example, a Time dimension might contain data for the previous 10 years, but a dynamic cube might reference data for a single year only.

You can also use dimension filters to restrict data to only those members that contain a corresponding record in the fact table. For example, if a product has no sales figures because it is new, you can exclude it from the Product dimension. This example requires a filter expression such as `Fact.productId = Dim.employeeId`. You must also set the **Exclude facts without corresponding dimension keys** property to False.

If a dimension is large, filtering dimension data can also improve performance of a published dynamic cube.

Important: When you create a dimension filter, it is automatically applied to all dynamic cubes that reference the dimension. If you do not want to apply a dimension filter to a dynamic cube, you must duplicate the dimension, delete the dimension filter, and reference the duplicated dimension.

The following table lists the properties that you can set when you define a dimension filter.

| Property | Description |
|----------|--|
| Name | The name of the dimension filter. Filters are not visible in the IBM Cognos studios. |

Table 18. Properties of a dimension filter (continued)

| Property | Description |
|--|--|
| Expression | Defines the value of the filter by using attributes or measures from the dimension. |
| Exclude facts without corresponding dimension keys | <p>Indicates whether to also filter fact data to ensure consistency in the summary data in a published dynamic cube.</p> <p>Default: True</p> <p>Important: Setting this option to True might reduce the performance.</p> <p>For example, you have dimension filter for the Time dimension to include data for year 2013 only. If the sales fact table also contains sales data for other years, and you do not set this property report to True, users can see the sales data for all years in the summary data.</p> |

Dimension filters in in-database aggregates

If your project contains an in-database aggregate that references a dimension filter, there might be issues if the in-database aggregate does not include the same attributes or measures that are specified in the filter expression. You must ensure that the data is valid for the in-database aggregate.

Defining a dimension filter

In IBM Cognos Cube Designer, you define dimension filters within a dimension at the project level.

Procedure

1. From the **Project Explorer** tree, select the dimension for which you want to define a filter.
2. Select the **Filters** tab.

3. Click the **New Filter** icon .

4. Select the filter, and then complete the dimension filter properties.

Defining named sets

A named set allows you to create an expression that defines a set of members. When you run a report containing a named set, the corresponding expression is evaluated and the resulting set of members is rendered in the report.

A named set is defined by a dimensional set expression that evaluates to a set of members from a single hierarchy. For example, `topcount(Customers, 5, Sales)`.

After you publish dynamic cubes, the named sets are available as data items in the **Named Sets** folder within the metadata tree in IBM Cognos studios.

Tip: Named sets can also be defined at the query level. However, the named sets that are defined for dynamic cubes can be authored once and reused multiple times in different reports.

IBM Cognos Cube Designer validates the syntax of named set expressions. After a cube is started, the dynamic cube server validates the semantics of the expressions by using the cube default member context and the security of the access account. Any expression that is not successfully validated while the cube is starting is removed from the cube and is not available in the studios. If an expression is removed, an error message is recorded in a `cognos_analytics_location/logs/XQE` log file.

A named set is dynamic. It is evaluated at report execution time using the query context and the security of the currently authenticated user. For example, a named set nested under a set of years is evaluated independently for each year.

You can use named sets within other named set expressions or within a calculated member or measure expression. Named sets can include parameters and macros.

If you are using member or attribute security, the security is also applied to the named set members.

Named sets from source cubes are not inherited in a virtual cube. If you want to use the named sets in a virtual cube, you must define them for the virtual cube.

You create named sets at the cube level, for dynamic cubes and virtual cubes. Named sets are stored within the **Named Sets** folder. You can organize named sets by creating sub-folders within the **Named Sets** folder.

Procedure

Use the following steps to create a new folder in the **Named Sets** folder and to define a named set expression.

1. Create a new folder in the **Named Sets** folder using the following steps:
 - a) From the **Project Explorer** tree, expand your cube.
 - b) Right-click the **Named Sets** folder, and then click **New > Named Set Folder**.
 - c) The new named set is created with a working name **New Named Set Folder**. Rename the folder as required.
2. Right-click the named set folder where you want to store the named set expression, and click **New > Named Set**.

Tip: You can create the named set expression in the folder that you created in step 1, or in a different folder in the **Named Sets** folder.

3. The new named set is created with a working name **New Named Set**. Rename the named set as required.
4. Double-click the name set to open the expression editor.
5. Define the named set expression using members and a valid set of multidimensional operators and functions.
6. Right-click the name set in **Project Explorer**, and click **Validate** to validate the expression syntax.

The validation errors are displayed on the **Issues** tab.

Tip: If a named set contains circular references (a reference to itself), a validation error occurs at cube start time and the named set is removed from the cube.

Parameter maps

Use parameter maps to substitute settings when a report is run. Parameter maps are objects that store key-value pairs.

Each parameter map has two columns, one for the key and one for the value that the key represents. You can manually enter the keys and values, import them from a file, or base them on existing query items in the model.

You can also export parameter maps to a file. To modify the parameter map, you can export the map values to a file, do additions or modifications, and then import it back into IBM Cognos Cube Designer. This is especially useful for manipulating large, complex parameter maps.

All parameter map keys must be unique so that IBM Cognos Dynamic Cubes can consistently retrieve the correct value. Do not place quotation marks around a parameter value. You can use quotation marks in the expression in which you use the parameter.

The value of a parameter can be another parameter. However, you must enclose the entire value in number signs (#). The limit when you nest parameters as values is five levels.

When you use a parameter map as an argument to a function, you must use a percentage sign (%) instead of a dollar sign (\$). Assign an alias to a query item that uses a parameter map as part of its name and to add the multilingual names to the object in the Language tab (Properties pane).

You create parameter maps at the project level. They are stored within the **Parameter Maps** folder.

Creating parameter maps manually

You can manually enter the keys and values into the parameter map.

Procedure

1. Right-click the **Parameter Maps** folder in the **Project Explorer** pane and select **New Parameter Map With Manual Entries**.

A new parameter map is added in the **Parameter Maps** folder under a working name **New Parameter Map**.

2. Rename the map as required and double-click it to open the editor.
3. Do one of the following:

- To manually enter values, click the **New parameter map entry** icon and type the values.
- To import keys and values, click **Import Parameter Map entries** icon and identify the location of the appropriate .csv or .txt file. For a .txt file to be used for import, the values must be separated by tabs and the file must be saved as UTF8 or Unicode format. ANSI text files are not supported.
- To export the parameter map, click the **Export parameter map entries** icon and save the map as a .csv or .txt file.

4. Optional: In the properties pane, specify the **Default Value** property.

The default value is used if the key that is used in an expression is not mapped. If no default is provided, an unmapped key could produce an error.

Creating parameter maps by importing entries

You can import an existing parameter map.

Procedure

1. Right-click the **Parameter Maps** folder in the **Project Explorer** pane and select **New Parameter Map With Imported Entries**.

2. Browse to the location of the appropriate .csv or .txt file and select the file.

For a .txt file to be used for import, the values must be separated by tabs and the file must be saved as UTF8 or Unicode format. ANSI text files are not supported.

A new parameter map is added in the **Parameter Maps** folder with the working name **New Parameter Map**.

3. Rename the map as required.
4. Optional: In the properties pane, specify the **Default Value** property.

The default value is used if the key that is used in an expression is not mapped. If no default is provided, an unmapped key could produce an error.

Creating parameter maps from existing query items

You can create a parameter map that is based on existing query items.

Procedure

1. Right-click the **Parameter Maps** folder in the **Project Explorer** pane and select **New Parameter Map Based on Query Items**.

A new parameter map is added in the **Parameter Maps** folder with the working name **New Parameter Map**.

2. Rename the map as required and double-click the map to open the editor.
3. Click the New Query item icon.
4. Click the query item to use as the key, and then click the query item to use as the value. Both query items must be from the same query subject.
5. Optional: In the properties pane, specify the **Default Value** property.
The default value is used if the key that is used in an expression is not mapped. If no default is provided, an unmapped key could produce an error.

Chapter 7. Dynamic cube modeling

With IBM Cognos Dynamic Cubes, you design and prepare dynamic cubes for use as data sources in the IBM Cognos studios.

The process to create dynamic cubes includes the following tasks:

- In IBM Cognos Administration, create a Java Database Connectivity (JDBC) data source connection to your relational database.

For more information, see the "Create a data source" topic in the *IBM Cognos Analytics Administration and Security Guide*.

- In Cognos Cube Designer, import the metadata to use for modeling dynamic cubes.
- In Cognos Cube Designer, model the dimensional metadata.
- In Cognos Cube Designer, model the dynamic cubes.
- In Cognos Cube Designer, deploy individual dynamic cubes as OLAP data sources to Content Manager in IBM Cognos Analytics.
- In Cognos Cube Designer, publish a package that includes a deployed cube.

It is also possible to manually publish a package using IBM Cognos Framework Manager. You might publish manually, for example, if you want to create a package containing more than one dynamic cube. For more information about creating and publishing packages, see the *IBM Cognos Framework Manager User Guide*.

- In IBM Cognos Administration, configure the deployed cube for use as a data source by the Query Service.
- In IBM Cognos Administration, start the dynamic cube.

Creating an IBM Cognos Framework Manager project for a ROLAP model

Before you can start designing cube models using the ROLAP Cube Designer, you must create a project using IBM Cognos Framework Manager.

Procedure

1. From the **Welcome** page of Cognos Framework Manager, click **Create a new project**.
2. In the **New Project** page, specify a name and location for the project, and click **OK**.
3. In the **Select Language** page, click the design language for the project, and click **OK**.
4. In the **Metadata Wizard**, click **Cancel**.

Cognos Framework Manager creates a project containing an empty model.

5. From the **Tools** menu, select **Run ROLAP Cube Designer**.

You are now ready to import the metadata for a cube. For more information, see [“Import metadata” on page 39](#).

6. When you finish, click **Save**  to save the project.

Model a dynamic cube

In IBM Cognos Cube Designer, you can define a dynamic cube manually or generate a dynamic cube based on a table in your relational database.

A basic dynamic cube contains the following items:

- A measure dimension that contains at least one measure
- At least one dimension
- At least one hierarchy and associated levels defined for each dimension
- Mappings between the measure and the dimensions
- Attributes that reference table columns either directly, by expressions, or by an expression that is a constant value

For more information, see [“Dynamic cubes”](#) on page 24.

When modeling a dynamic cube, the relationship between a measure and a dimension must be defined for each dimension in the cube. This relationship is defined by a measure-to-dimension join. For more information, see [“Defining a measure-to-dimension join”](#) on page 67.

| <i>Table 19. Properties of a dynamic cube</i> | |
|---|--|
| Property | Description |
| Name | The name of the dynamic cube. This is also used as the name of the data source that represents the cube. If the project supports multiple locales, there can be versions of the name in all supported languages. Tip: When creating a Framework Manager package for the dynamic cube, select this name from the list of data sources. |
| Comment | A comment or description of the dynamic cube. Comments are not visible in the IBM Cognos studios. |
| Remove non-existent tuples | True (default) - remove tuples from the cross join set that cannot contain any data. False - do not remove tuples from the cross join set. Applies when a dimension has multiple hierarchies and a report contains the cross join of two or more of those hierarchies. With this feature enabled, only those tuples for which data may exist are retained from the cross join, improving report efficiency. A cross join of hierarchies from the same dimension may contain tuples for which no data can possibly exist. For example, in a Time dimension with two hierarchies, the cross join of [2011 Q1] and [2011 Aug] would be removed since [2011 Q1] and [2011 Aug] do not share a common month. |

Related tasks

[Defining a dynamic cube manually](#)

[Defining a dynamic cube based on a relational table](#)

Defining a dynamic cube based on a relational table

When you generate a dynamic cube, IBM Cognos Cube Designer creates a basic cube structure. The structure includes a measure dimension with measures, a set of dimensions, and appropriate mappings to the tables and columns in the database. To complete the dynamic cube definition, you resolve any issues and manually adjust the definition to meet your requirement.

Before you begin

By selecting a fact table, you can use one of two options to generate a dynamic cube.

- **Generate, cube with basic dimensions**

This option generates one or more levels per dimension. Dimension tables are located by using the foreign primary key relationship and dimensions are created based on these dimension tables. If a

single dimension table is detected, a single level is created by using the table columns as attributes of the level. If more levels are required, create them manually and move the attributes to the new levels. If a snowflake dimension is detected, a level is created for each table in the snowflake. Measures in the measure dimension are generated by using numeric columns that are not foreign keys in the selected fact table.

- **Generate, cube with dimensions using data sampling**

This option generates one or more levels per dimension. It applies a heuristic algorithm to interpret relationships among the data to identify levels. A hierarchy of levels is generated, based on the cardinality of the data and the column names. If your data is clean and complete, the generated levels are more accurate. The algorithm does not detect multiple hierarchies.

Tip: If the table that you select has no relationship to other tables, Cognos Cube Designer provides the option to create a cube by using the selected table as a measure dimension, with any numeric columns added as measures.

Because Cognos Cube Designer needs foreign keys to determine relationships, only fact tables with foreign keys can be used to generate a dynamic cube. If your database does not use referential integrity, you can manually define a dynamic cube to meet your requirements. For more information, see [“Defining a dynamic cube manually”](#) on page 63.

Procedure

1. Select a fact table in the **Data Source Explorer**.
2. Right-click and select a **Generate** option.
 - **Generate, cube with basic dimensions.**
 - **Generate, cube with dimensions using data sampling.**

What to do next


Review the generated cube definition and, if required, manually modify it to reflect how you want to view your data. Any object that causes a modeling issue or requires further design is identified in the **Project Explorer** and an icon appears next to the object. On the **Issues** tab, you might be presented with actions required to resolve these issues and validate the dynamic cube.

Defining a dynamic cube manually

Because IBM Cognos Cube Designer needs the information provided by foreign keys to determine relationships, only fact tables with foreign keys can be used to generate a dynamic cube. If your database does not use referential integrity, you can manually define a dynamic cube to meet your requirements.

Any object causing a modeling issue or requiring further design is identified in the **Project Explorer** and an icon appears next to the object. You can validate an entire project or an individual object at any time. An effective practice is to validate each object as you create it. Right-click an object in the **Project Explorer** tree and select **Validate**.

Procedure

1. Select a namespace in the **Project Explorer** tree.
2. Click **New Cube** .

What to do next

A measure dimension is automatically created. To complete your dynamic cube, define your measures, dimensions, hierarchies, levels and joins.

Model measures

Using IBM Cognos Cube Designer, you can define a measure manually or you can generate a measure based on a column in your relational database. A dynamic cube contains one measure dimension.

For more information, see [“Measures”](#) on page 26.

| Property | Description |
|-----------------|--|
| Name | The measure dimension name shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Comment | A comment or description of the measure dimension. Comments are not available to the studio users. |
| Default Measure | <p>During report processing, if no measure is defined for evaluation of a value expression, the default measure is used. The default measure can be a regular or calculated measure.</p> <p>Important: Calculated measures are not supported for DMR modeling.</p> <p>Default: the first measure added to the dynamic cube.</p> |

| Property | Description |
|-------------|---|
| Name | The measure name shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Comment | A comment or description of the measure. Comments are not available to the studio users. |
| Expression | <p>The expression can refer to measures in the dynamic cube.</p> <p>The expression cannot contain multidimensional dynamic query mode constructs.</p> <p>This property is only available for measure items that were created in Cognos Cube Designer.</p> |
| Column Name | <p>The name of the associated column in the relational database.</p> <p>This property cannot be edited.</p> |

Table 21. Properties of a measure item (continued)


| Property | Description |
|-------------------|---|
| Visible | <p>Controls whether the object is visible in the published package.</p> <p>Non-visible measures are typically used to represent intermediate values in the construction of a complex calculated measure. These measures are not intended to be used for direct reporting. However, a non-visible measure is always present in the published package because the measure might be needed by other objects in a dynamic cube.</p> <p>Important: Calculated measures are not supported for DMR modeling.</p> <p>Non-visible measures do not display in the metadata browser and are removed from the output of reports which contain references to them. For example, a report that references a non-visible measure does not include output from that measure.</p> <p>The default measure cannot be hidden.</p> <p>Default: True</p> |
| Data Type | <p>The data type of the associated column in the relational database.</p> <p>This property cannot be edited.</p> |
| Precision | <p>The precision of the associated column in the relational database.</p> <p>This property cannot be edited.</p> |
| Scale | <p>The scale of the associated column in the relational database.</p> <p>This property cannot be edited.</p> |
| Regular Aggregate | <p>The primary method used to aggregate data for the measure.</p> <p>Default: Sum</p> |
| Data Format | <p>Set the default format properties for corresponding data type (number, currency, percentage) for the measure.</p> |

Defining a measure based on a relational column

In IBM Cognos Cube Designer, you can define a measure based on a relational column. To create measures, add a cube and then create measures in the measure dimension folder under the cube.

For information about creating calculated measures, see [“Calculated members”](#) on page 73.

Procedure

1. From the **Project Explorer** tree, expand your cube.
2. Right-click the measure dimension  and select **Open Editor**.
3. From the **Data Source Explorer**, drop a table column onto the **Editor** pane.



The mapping to the associated column is automatically created. The **Property** fields are initialized from the table column values.

Defining a measure manually

In IBM Cognos Cube Designer, you can define a measure manually by creating a mapping to a database column or to an expression. To create measures, add a cube and then create measures in the measure dimension folder under the cube.

For information about creating calculated measures, see [“Calculated members”](#) on page 73.

Procedure

1. From the **Project Explorer** tree, expand your cube.
2. Right-click the measure dimension  and select **Open Editor**.
3. Click **New Measure**  to add a blank measure.
4. To give the new measure a more meaningful name, right-click the new measure and select **Rename**.
5. You can complete the measure one of two ways:
 - To map the measure to a table column, drag a table column from the **Data Source Explorer** onto the **Mapping** field.
 - To map the measure to an expression, define an expression in **Expression** property on the **Properties** pane.

Defining aggregation rules

Each measure has a regular aggregation type. The **Regular Aggregate** property identifies the type of aggregation that is applied to the measure. Aggregation rules can be used in addition to the regular aggregate. They specify how semi-aggregate measures are aggregated with respect to information from the dimension.

When you import metadata, IBM Cognos Cube Designer assigns values to the **Data Type, Precision, Scale** and **Regular Aggregate** properties based on the relational object. For cube measures, you can define aggregation rules for each related dimension.

Aggregate rules are applied in this order:

1. The **Regular Aggregate** property is applied to dimensions that are included in the query but do not have assigned **Aggregation Rules**.
2. The **Aggregation Rules** are then applied to their specified dimensions, in the order that you specified the rules.
3. The report-level aggregation that is specified in the query.

For more information about measures and aggregation rules, see [“Measures”](#) on page 26.

Procedure

1. Select the **Aggregation Rules** tab.
2. Select a measure in the **Measures** pane.
3. Select a related dimension from the **Dimension** column.
4. Click **Include** to activate the aggregation rule for the dimension.
5. From the **Aggregation Rule** drop-down list, select the aggregation rule to be used for the selected dimension.
6. When you have finished adding aggregation rules for the dimension, use **Up, Down, Top** and **Bottom** to specify the order to apply the aggregation rules.

Defining a measure-to-dimension join

You can define a measure-to-dimension join in a dynamic cube when the level of a join does not match the level of the fact table. You must define the correct measure-to-dimension join to avoid double counting data from the fact table.

For example, a fact table can contain data at the Day level, but it can be joined to the Time hierarchy at the Week level. If the measure-to-dimension join is not defined, measure data equates to actual counts multiplied by the number of days in a week.

Before you begin

You must add the required dimension and measures to a dynamic cube before you can define a measure-to-dimension join. For more information, see [“Model dimensions” on page 45](#) and [“Model measures” on page 63](#).

Procedure

1. From the **Project Explorer** tree, right-click the cube and select **Open Editor**.
2. For each dimension, select **Edit**.
3. Specify the join by relating columns in the dimension to columns in the measure.
4. Specify the relationship operator.
5. If the join is at a higher grain than the lowest level of the dimension, clear the **Join is at the lowest level of detail for the dimension** check box.



Warning: IBM Cognos Cube Designer cannot automatically detect that a join is at a higher grain than the lowest level of a dimension.

Measure dimension filters

In IBM Cognos Dynamic Cubes, you can create measure dimension filters to restrict the fact data available in a published dynamic cube when measures contains more data than is required by the cube.

If a dimension is large, filtering dimension data can also improve the performance of a published dynamic cube.

The following table lists the properties that you can set when you define a measure dimension filter.

| Property | Description |
|------------|--|
| Name | The name of the measure dimension filter. Filters are not visible in the IBM Cognos studios. |
| Expression | Defines the value of the filter by using attributes and measures from the dynamic cube. |

Measure dimension filters in in-database aggregates

If your project contains an in-database aggregate that references a measure dimension filter, there might be issues if the in-database aggregate does not include the same attributes or measures that are specified in the filter expression. You must ensure that the data is valid for the in-database aggregate.

Defining a measure dimension filter

In IBM Cognos Cube Designer, you define measure dimension filters within a dynamic cube.

Procedure

1. From the **Project Explorer** tree, select the measure dimension for which you want to define a filter.
2. Select the **Filters** tab.

3. Click the **New Filter** icon .

4. Select the filter, and then complete the measure dimension filter properties.

Measure folders

In IBM Cognos Cube Designer, you can create measure folders within a measure dimension to organize measures and calculated measures. You can also create sub folders within a folder.

A measure folder does not have any value and cannot be included in expressions or calculations.

When you publish a dynamic cube, empty folders are not visible to report users in the IBM Cognos studios. A folder that contains only hidden measures or secured measures is treated as an empty folder.

Measure folders in virtual cubes

You can create measure folders in a virtual cube. If a source cube includes a measure folder, it is not included in the virtual cube, but any measures within the folder are added.

Creating a measure folder

You create measure folders at the cube level.

Procedure

1. From the **Project Explorer** tree, select a measure dimension, and click the **New measure folder** icon



2. If required, create sub folders by selecting the measure folder, and clicking **New measure folder**.
3. Drag objects as required into the measure folders.

What to do next

You can change the sort order of objects in measure folders. For more information, see [“Changing the sort order of measures and folders” on page 69](#).

Sort measures and folders

In IBM Cognos Cube Designer, you can change the order in which measures, calculated measures, and folders are sorted within a measure dimension. You can also sort objects within a specific folder.

The default sort order for dynamic cubes is the order that is shown in the measure dimension. You can change the order by manually moving objects to any position that you require. You can also sort objects in ascending or descending alphanumeric order. Sorting applies within a single level of nesting only. If a folder contains sub folders, they are not included in an alphanumeric sort.

Important: The default sort order for dynamic cubes that are published with previous versions of Cognos Cube Designer is ascending alphanumeric order. If you update or republish the cube with version 10.2.1.1, it overrides the previous sort order with the new default order.

Cognos Cube Designer sorts measures according to the project design language, not the locale that is defined for measures and folders or the server content language.

Sorting in virtual cubes

You can sort measures, calculated measures, and measure folders in a virtual cube. If objects are sorted in a source cube, the sort order is not included in the virtual cube.

Changing the sort order of measures and folders

You sort measures, calculated measures, and measure folders at the cube level.

Procedure

1. To manually sort measure objects, drag them to the required position in a measure dimension in the **Project Explorer** tree.
2. To sort items alphanumerically, from the **Project Explorer** tree, right-click a measure dimension or folder in which to sort items, and click one of the following options:
 - **Sort, Ascending**
 - **Sort, Descending**

Deploying and publishing dynamic cubes

When you finish modeling a dynamic cube in IBM Cognos Cube Designer, you can deploy it as an OLAP data source to Content Manager. To work with a deployed cube in the IBM Cognos studios, you must also publish a Framework Manager package for it, configure the cube as a data source, and start the cube.

Important: You must validate a dynamic cube before you can deploy it.

You use the **Publish** option to deploy a dynamic cube. You can also perform the additional tasks required to publish a cube in one step.

- **Select all options**

This option publishes a Framework Manager package for the deployed dynamic cube, then configures and starts the cube.

- **Publish the package in: My Folders**

By default, the cube name is used as the Framework Manager package name. You can specify a different package name in the **Package Name** box.

Tip: You can move the location of published packages by using IBM Cognos Administration.

- **Add the dynamic cube to the default dispatcher**

This option configures the deployed dynamic cube as a data source.

- **Start the dynamic cube**

This option starts the dynamic cube, if you also configure the cube as a data source.

Note: The **Dispatcher URI** specified when configuring Cognos Cube Designer must be identical to the **Dispatcher URI** specified when configuring the IBM Cognos Analytics server. However, the **Dispatcher URI** specified when configuring Cognos Cube Designer must be in lowercase even if the **Dispatcher URI** specified when configuring the IBM Cognos Analytics server is in uppercase.

- **Associate my account and signon with the cube datasource**

This option allows you to use credentials to access the data source in the IBM Cognos Studios.

Select if anonymous access is disabled. Your account must use associated credentials. Go to the **Personal** tab in the **Set preferences** dialog of the IBM Cognos Portal, and create your credentials.

Important: Because these options use default settings, they are intended for deploying and testing a dynamic cube in a development environment rather than a production environment.

Procedure

1. Open the project that contains the dynamic cube that you want to deploy and publish.
2. In the **Project Explorer** tree, right-click the required cube, and then select **Publish**.
3. Select the additional options required to publish the cube.
4. Click **OK**.

Results

When the deployment and publish process is complete, a confirmation message is shown.

Creating and publishing packages

You can now publish a package containing more than one cube. A package can contain dynamic cubes, virtual cubes, namespaces and folders.

You create packages at the project level.

Procedure

1. Select the **Packages** folder from the **Project Explorer** tree.

2. Click **New Package** .

If you include a namespace or folder, all cubes within it are included by default.

By default, packages are published to the Public Folders location.

3. If required, change the **Publish location** on the **Properties** tab.
4. When you are ready to publish the package, right-click it, and then select **Publish**.

Publishing packages based on ROLAP data sources

You can use IBM Cognos Framework Manager to select a ROLAP data source and create a package based on a cube. You can then publish the package directly to IBM Cognos Analytics, making it available for use in the IBM Cognos reporting, dashboarding, or the legacy studios.

By default, each package contains a connection to only one cube. If you want to create a package containing multiple cubes, run the Metadata Wizard and create a package for each cube. Then create a package that includes individual packages as required.

Before creating a package containing multiple cubes, consider the potential performance impacts. In IBM Cognos Analytics, each time a package is used, a connection is made to each of the data sources defined in the package. Creating large packages with multiple cubes can have a negative impact on performance. To offset the potential performance impact of creating one large package containing many cubes, create one package per cube and then create smaller combinations of packages as required.

Estimating hardware requirements

You can use the hardware sizing calculator in Cognos Cube Designer to estimate the minimum hardware requirements to provide the optimal level of query performance and product stability with Cognos dynamic cubes.

This calculator is applicable to base cubes only. It does not consider multiple locales or shared dimensions in its estimates.

Use the estimated values when you configure your dynamic cube in IBM Cognos Administration. For example, use the estimated memory size when you configure the query service properties **Initial JVM heap size for the query service (MB)** and **JVM heap size limit for the query service (MB)**. Use the

estimates for data cache, aggregate cache, and hard disk space when you configure the dynamic cube properties **Data cache size limit (MB)**, **Maximum space for in-memory aggregates (MB)** and **Maximum amount of disk space to use for result set cache (MB)**.

For information about configuring query service properties, see [“Setting query service properties for dynamic cubes”](#) on page 129. For information about configuring dynamic cube properties, see [“Setting dynamic cube properties”](#) on page 132.

In the calculator help pane, you can find information about the parameters that you need to specify to obtain your estimates.

Procedure

1. In the **Project Explorer** tree, locate the dynamic cube that you want to configure.
2. From the cube right-click menu, click **Estimate Hardware Requirements**.

The calculator is displayed.

3. Enter values for the different parameters.

As you type the value, the values for **Memory**, **CPU cores**, and **Hard disk space** are calculated.

Tip: When you enter a value for a parameter, the help topic associated with the parameter is displayed in the help pane. You can also click the parameter value box to display the help topic.

To see the estimates that are based on the current cube values, click the **Retrieve Values from Cube** button.

4. Experiment with different values and write down the numbers.
5. To close the calculator, click **OK**.

Results

The calculator provides estimates for an individual dynamic cube. To make an estimate for the whole environment, add the estimated values for this cube and the estimated values for other cubes to the estimated hardware requirements for the report server.

Chapter 8. Advanced dynamic cube modeling

After you create a basic dynamic cube in IBM Cognos Cube Designer, there are numerous ways to enhance the functionality of the cube.

You can perform the following tasks:

- add calculated members and measures
- model relative time dimensions
- use multi-locales and associated formatting

Calculated members

Calculated members add business logic into dimensions by introducing members whose value is computed from the values present within the underlying data.

The new members are available for use without being added to the underlying relational data source. A calculated member is defined by a dimensional expression.

A calculated measure is a calculated member that belongs to the measure dimension. There are no behavior differences between calculated members and calculated measures.

For more information, see [“Calculated members in reports”](#) on page 183.

For information about relative time calculated members, see [“Model relative time dimensions”](#) on page 77.

Table 23. Properties of a calculated member

| Property | Description |
|---------------|--|
| Name | The name that appears in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Parent Member | Specifies the parent of the calculated member in the member tree. |
| Expression | Defines the value of the calculated member using other members and a valid set of multidimensional operators and functions. |

Table 24. Properties of a calculated measure

| Property | Description |
|-------------|--|
| Name | The name that appears in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Expression | Defines the value of the calculated measure using other members and a valid set of multidimensional operators and functions. |
| Data Format | Set the default data properties for each type of data. |

Table 24. Properties of a calculated measure (continued)

| Property | Description |
|-------------------|--|
| Visible | <p>Controls whether the object is visible in the published package.</p> <p>Non-visible measures are typically used to represent intermediate values in the construction of a complex calculated measure. These measures are not intended to be used for direct reporting. However, a non-visible measure is always present in the published package because the measure might be needed by other objects in a dynamic cube.</p> <p>Non-visible measures are not displayed in the metadata browser and are removed from the output of reports which contain references to them. For example, a report that references a non-visible measure does not include output from that measure.</p> <p>Default: True</p> |
| Regular Aggregate | <p>The primary method used to aggregate data for the measure.</p> <p>Default: Sum</p> |

Author calculated member expressions

IBM Cognos Cube Designer validates the syntax of expressions. After a cube is started, the dynamic cube engine validates the semantics of the calculated member and calculated measure expressions. Any calculated member or expression which is not successfully validated during cube start is removed from the cube and is not available in the studios.

The expression editor does not limit functions to the valid ones for a specific context.

There are some restrictions that apply to IBM Cognos Dynamic Cubes calculated members.

Do not use the following relational constructs in expressions used to define calculated members:

- Value summary functions (Not Member Summary functions)
- Value Analytic functions (rank, first, last, percentile, percentage, quantile, quartile, distinct clause, prefilter clause) - (Summaries/Member Summaries)
- Value Summary functions (standard-deviation-pop, variance-pop, distinct clause, prefilter clause)
- All running- or moving- summary functions (Summaries)
- All FOR clauses in aggregate functions (Summaries/Member Summaries)
- Date/time constants (Constants)
- All business date/time functions (Business Date/Time functions)
- Like, lookup, string concat '||', trim, coalesce, cast (Common Functions)
- MOD function (Common Functions)

Calculated member and measure examples

IBM Cognos Cube Designer allows for the definition of dimensional calculated members and measures. Such expressions were previously defined only in the reporting environment. When defined in a dynamic cube, the calculated members are accessible in all of the IBM Cognos Analytics studios. You can use calculated measures to determine constant or weighted values. You can create calculated members that represent an N period rolling window of data relative to a current period member.

Constant and weighted allocation

Measures in base dynamic cubes must have the same grain since each base cube is constructed from a single fact table. In a virtual cube, it is possible for a measure from one base cube to be valid only for a subset of the levels of a virtual hierarchy.

In this example, the virtual cube Sales Inventory is built from two base cubes: Sales and Inventory. The Sales cube has the measure Sales Amount and its Time hierarchy contains Year and Quarter levels. The Time hierarchy in the Inventory cube contains a Year, Quarter, and Month levels. When the Sales Inventory cube is created, the virtual Time hierarchy contains the Year, Quarter, and Month levels.

In this situation, any Sales Amount value that is computed in the virtual cube at the Day level is null since there is no value in the Sales cube at the Month level.

In the following diagram, the Sales Amount measure has no values at the Month level but the Stock measure, from the Inventory cube, does. Only partial data is used to show the hierarchy.

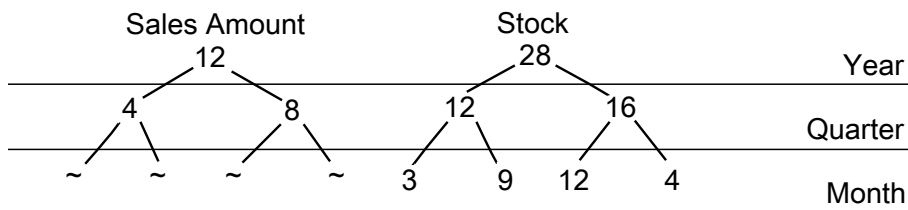


Figure 12. Example of differences in the time hierarchy for two cubes

You can use calculated measures to compute constant or weighted values for a measure such as Sales Amount. A constant allocation allocates a measure value from a higher level evenly across all of its descendants at each level below the in scope level. The in scope level is typically the lowest at which the measure is valid.

Using constant allocation, the following diagram shows the Sales Amount values. The values from the Quarter level are evenly distributed across the descendants at the Month level. Only partial data is used to show the allocation.

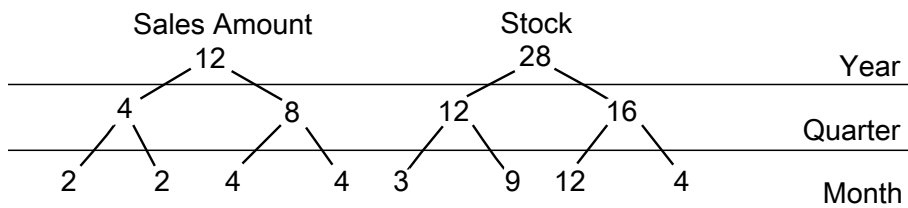


Figure 13. Example of the use of constant allocation

A weighted allocation allocates values to the descendants relative to the values of another measure that is in scope, and that is correlated with the measure being allocated so the allocation is reasonable.

For example, the Sales Amount values are allocated based on the weights of the Stock measure from the Inventory cube.

Using weighted allocation, the following diagram shows the Sales Amount values. The values from the Quarter level are distributed using the same weighting as the Stock measure. Only partial data is used to show the allocation.

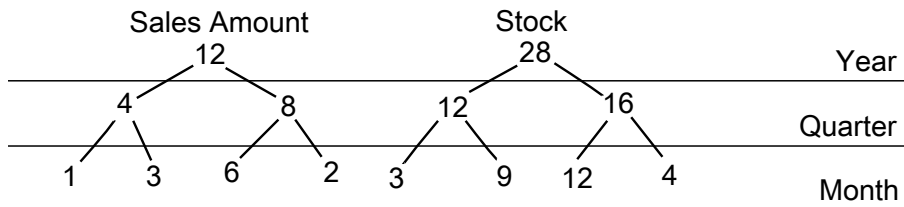


Figure 14. Example of the use of weighted allocation

Constant and weighted allocation expressions

Note: To create the expression for a calculated measure, the database objects must be dragged from the **Project Explorer** into the **Editor**. In the example code, the bold text represents metadata objects such as hierarchies, levels, and measures that are dragged and dropped into the expression editor. The code is visible in the **Expression** property but cannot be entered as text.

The following expressions can be used to create calculated measures in the sample virtual cube `gosldw_sales_and_target`. Because the Sales Target data at the month level exists in the sample cube, these expressions are not necessary but are shown to illustrate how the expressions are constructed.

In this constant allocation example, the Sales Target measure is used.

```

if (roleValue
 ('_levelNumber', currentmember
 ( [gosldw_sales_and_target].[Time].[Time])) > 2 )
then
 (
 tuple( [gosldw_sales_and_target].[Measures].[Sales target],
 ancestor(currentmember( [gosldw_sales_and_target].[Time].[Time]),
 [gosldw_sales_and_target].[Time].[Time].[Quarter]))
 /
 count(1 within set descendants
 (ancestor(currentmember( [gosldw_sales_and_target].[Time].[Time]),
 [gosldw_sales_and_target].[Time].[Time].[Quarter])
 ),
 roleValue('_levelNumber', currentmember
 ( [gosldw_sales_and_target].[Time].[Time])) - 2, self ) ) )
 else
 (
 [gosldw_sales_and_target].[Measures].[Sales target]
 )
 )

```

In this weighted allocation example, the Sales Target values are allocated based on the weights of the Revenue measure.

```

if (roleValue
 ('_levelNumber', currentmember( [gosldw_sales_and_target].[Time].[Time])) > 2 )
then
 (
 tuple( [gosldw_sales_and_target].[Measures].[Sales target],
 ancestor(currentmember( [gosldw_sales_and_target].[Time].[Time]),
 [gosldw_sales_and_target].[Time].[Time].[Quarter]))
 ) *
 tuple( [gosldw_sales_and_target].[Measures].[Revenue],
 currentmember( [gosldw_sales_and_target].[Time].[Time])
 ) /
 tuple( [gosldw_sales_and_target].[Measures].[Revenue],
 ancestor(currentmember( [gosldw_sales_and_target].[Time].[Time]),
 [gosldw_sales_and_target].[Time].[Time].[Quarter]))
 )
 )
 else
 (
 [gosldw_sales_and_target].[Measures].[Sales target]
 )
 )

```

Defining a calculated member



You define calculated members in the expression editor using dimensional constructs and functions. You can define a calculated member based on a calculated member.

Calculated members are added to the member tree as a children of the parent member. You identify the parent member by selecting a member from the member tree under the **Members** folder of a hierarchy.

If there is no ALL member, the calculated member does not have to have a parent defined. The calculated member then becomes a member of the root level. If there is an ALL member, the calculated member must have a named parent and if one is not specified, the calculated member will fail to load. The failure is recorded in the log file.

It is a good practice to use a naming convention so that you and your report users can easily identify calculated members.

Procedure

1. From the **Project Explorer**, click a dimension and expand it.
2. Right-click a hierarchy that belongs to the dimension and select **Open Editor**.
3. Expand the hierarchy to access the **Members** folder.
4. Expand the member tree until you can view the member you wish to define as the parent of your new calculated member.
5. Select the **Calculated Members** tab.
6. Click **New Calculated Member** .
7. Select the new calculated member.
8. To set the **Parent Member** on the **Properties** pane, drag a member from the member tree in the **Project Explorer**.
This property specifies the position of the calculated member in the member tree.
9. From the **Properties** pane, define the calculated member in the **Expression** property.
 - To use an object from the project, drag the item from the **Project Explorer** into the expression.
 - To use a calculated member, drag the calculated member from the member tree.
 - To add functions, summaries and operators, select the **Functions** tab,  to access the required elements.
10. Right-click the **Members** folder of the hierarchy and select **Refresh**.

Results

The new calculated member is displayed under the **Calculated Members** folder for the hierarchy. The calculated member can also be seen under the parent member in the **Members** folder of the hierarchy.

Model relative time dimensions

In IBM Cognos Dynamic Cubes, relative time members are specialized calculated members that are added to a time hierarchy when a cube is started.

You use IBM Cognos Cube Designer to create a fixed set of relative time members in a time hierarchy, and create custom relative time calculated members (if required). A report author can then create reports relative to the current period. These reports can be run at any time and remain valid based on the value of the current period at the report execution time.

When you model a relative time dimension, you can include the following predefined relative time members:

- Current Period
- Prior Period
- Current Period to Date
- Prior Period to Date
- Current Period to Date Change
- Current Period to Date % Growth
- Next Period
- Next Period to Date
- Next Period to Date Change
- Next Period to Date % Growth

For more information about next period members, see [“Next period relative time members” on page 80](#).

You can also create the custom relative time members. For more information, see [“Custom relative time members” on page 81](#).

You can create further calculated members that are based on predefined or custom members. A report author can then create expressions that are based on these members. Calculated member expressions are resolved when a cube is started or member cache refreshed.

Levels

Levels must appear in order in a hierarchy. The order in which they can be used is reflected in the level type list in Cognos Cube Designer.

The level type is used to construct the name of predefined relative time members. For example, Current Semester (1/2 of a time interval) .

Special rules apply when you use the following level types: semesters (1/2 of a time interval), trimesters (1/3 of a time interval), holidays, and seasons.

- Holidays and seasons can be assigned in any order to any level, and can be used multiple times in the same hierarchy.
- Semesters (1/2 of a time interval) and trimesters (1/3 of a time interval) cannot be used in the same hierarchy.
- Trimesters (1/3 of a time interval) and quarters cannot be used in the same hierarchy.
- Semesters (1/2 of a time interval) and quarters can be used in the same hierarchy.

Current period

Each level includes a **Current Period** property. The current period property of a level is used to filter members by their level key value to identify the single leaf member that is the current period member in the hierarchy. This is the basis for defining the current member at each level in the hierarchy. If a current period expression is defined, it is used to filter members at that level by the value of the level key for that level. The current period value should map to the business key value of the member you want to be the current period member. The expression can be static, based on a current date/time value, or based on a value in the relational database that is typically populated by the ETL process.

When defining the current period based on a value in a database that is created during the ETL process, use this approach. Create a single-row table with one or more columns containing the key values that correspond to the levels of the time dimension at which you want to define the current period. The table must also contain a column with a single arbitrary value, such as an integer with the value of 1. The time dimension table must contain a corresponding column with the same single value that can be used in the

dimension implementation editor to define a join between the single-row table and the time dimension table. The one or more columns in the single-row table must be added as hidden attributes of one of the levels of the hierarchy, typically the highest level. These attributes can now be referenced in the current period expression to define the current period. During ETL, the values required for the current period are applied to the single-row table so that when a dynamic cube is started, the current period value is obtained from the values in the single-row table.

The levels in a relative time dimension are not required to have a current period. If no current period expressions are defined, the current period that is used is the rightmost, most recent leaf level member of the hierarchy.

The combination of level current period expressions is used to identify a specific leaf member. You can determine which member is used as the current period by examining the levels of the hierarchy in a top-down manner. If there are levels with no current period expression defined, the member that is chosen at each level is the rightmost, most recent child of the member that is selected from the previous, higher level. As soon as a level is encountered where a current period expression is defined, the default selection of members at the higher levels is ignored and the member at that level which determines the path to the leaf level current period begins with the member defined by the expression. It is possible to define the current period of a hierarchy by providing a current period at the leaf level.

Where current period expressions are defined for all levels in a relative time dimension, the member captions shown in the hierarchy reflect these expressions. Where there are no current period expressions defined, the captions use the rightmost, recent member as the current period for that level.

Calculated member behavior

The following relative time members have the same behavioral characteristics:

- Current Period
- Prior Period
- Current Period to Date
- Prior Period to Date
- Next Period
- Next Period to Date
- Custom single period
- Custom period-to-date

The following relative time members have the same behavioral characteristics:

- Period to Date Change
- Period to Date % Growth
- Next Period to Date Change
- Next Period to Date % Growth
- Custom N-period running total

For more information, see [“Relative time calculated members in reports” on page 185](#).

Virtual cubes

All relative time definitions (members and auto-generation options) are inherited from the single source cube that supplies the current period.

Security

A relative time dimension cannot include members with non-default security rules.

Next period relative time members

The following next period members are available to add to a relative time dimension:

- Next Period
- Next Period to Date
- Next Period to Date Change

This member is derived from Next Period to Date - Period to Date

- Next Period to Date % Growth

This member is derived from Next Period to Date Change / Period to Date * 100

In all cases, Period is the level type that is defined for the hierarchy, for example Year or Semester.

These members have a fixed offset from the current period of +1. For example, if the current month is November, the next month is December.

Consider the following Time dimension and Sales fact tables. The current quarter is 201303.

Table 25. Time dimension

| Year | Quarter |
|------|---------|
| 2012 | 201201 |
| 2012 | 201202 |
| 2012 | 201203 |
| 2012 | 201204 |
| 2013 | 201301 |
| 2013 | 201302 |
| 2013 | 201303 |
| 2013 | 201304 |
| 2014 | 201401 |
| 2014 | 201402 |
| 2014 | 201403 |
| 2014 | 201404 |

Table 26. Sales fact table

| Quarter | Sales |
|---------|-------|
| 201201 | 3 |
| 201202 | 4 |
| 201203 | 5 |
| 201204 | 6 |
| 201301 | 7 |
| 201302 | 8 |
| 201303 | 9 |
| 201304 | 10 |

| <i>Table 26. Sales fact table (continued)</i> | |
|---|--------------|
| Quarter | Sales |
| 201401 | 11 |
| 201402 | 12 |
| 201403 | 13 |
| 201404 | 14 |

The value of 'Year to Date (2013)' is 24. This value is derived from 'aggregate(currentMeasure within set periodsToDate(Year, 201303))'.

The value of 'Prior Year to Date (2012)' is 12. This value is derived from 'aggregate(currentMeasure within set periodsToDate(Year, parallelPeriod(Year,1, 201303)))'.

The value of 'Next Year to Date (2014)' is 36. This value is derived from 'aggregate(currentMeasure within set periodsToDate(Year, parallelPeriod(Year,-1, 201303)))'.

The value of 'Year to Date Change' is 12. This value is derived from 'Year to Date' - 'Prior Year to Date'.

The value of 'Next Year to Date Change' is 12. This value is derived from 'Next Year to Date' - 'Year to Date'.

The value of 'Next Year to Date % Growth' is 50%. This value is derived from 'Next Year to Date Change' / 'Year to Date' * 100.

Custom relative time members

You can add the following types of custom relative time members to a relative time dimension:

- [“Custom single period” on page 82](#)
- [“Custom period-to-date” on page 83](#)
- [“Custom N period running total” on page 85](#)

For more information on creating a custom relative time member, see [“Creating a custom relative time member” on page 88](#).

IBM Cognos Cube Designer validates custom member property values as follows:

- Offsets must be an integer value (-n, 0, +n).
- The context period must be higher than the target period. If the target period is set to the highest level, the context period must be blank.
- For period-to-date, if life-to-date is false, the target period must be lower than the to-date period. If life-to-date is true, target period can be the highest level.
- For n-period running total, the target period cannot be the highest level, and number of periods must be an integer greater than or equal to 1.

The target and context properties operate in a similar way to the parallel period function, allowing selection of any member relative to a current period member. Start with the current period member at the target level. Find the ancestor at the context level, and then find the sibling of ancestor using the context offset. Locate the member parallel to current period among the descendants of sibling at target level. Then, apply the target offset.

If the target and context properties are set so that the corresponding member is outside the bounds of the hierarchy, the custom member is dropped at cube start time, and an event is logged in the file `cognos_analytics_location/logs/XQE/xqe.log.xml`

The parent of a custom relative time member in a hierarchy is assigned automatically by the server as follows:

- Custom single period - the parent is the predefined current period member at the level above the target period. For example, Current Quarter is the parent of 'Same Month Last Year'.
- Custom period-to-date - the parent is the predefined period-to-date member at the level above the to-date period. For example, Year to Date is the parent of 'Quarter to Date Last Year'.
- Life-to-date - the parent is the All member for a single root hierarchy, or the member is at the root level for a multi-root hierarchy.
- N-period running total - the parent is the All member for a single root hierarchy, or the member is at the root level for a multi-root hierarchy.

Limitations

Custom relative time members have the following limitations:

- When you browse relative time members in Cognos Cube Designer, the endpoint member (normally shown in parenthesis) is not displayed for custom members.
This applies only to source cube, not virtual cubes.
- When you browse relative time members in Cube Designer, the sub tree of reference members is not displayed for custom members.
This applies only to source cube, not virtual cubes.
- For the life-to-date custom member, there is no sub tree of relative time members available in IBM Cognos Cube Designer or the IBM Cognos components, such as reporting, dashboarding, or the legacy studios.
- Values that are returned by custom and predefined relative time members for a retail calendar or Gregorian calendar with a week level differ from values that are returned by IBM Cognos Transformer or PowerPlay®.

Custom single period

Use custom single period to define a relative time member that corresponds to a single member at the same level as a current period member, but offset by a defined period. The relative position is specified by a target period with offset, and a context period with offset.

For example, to define a relative time member "Same month, last quarter" you specify:

- target period: month
- target period offset: 0
- context period: quarter
- context offset: -1

This example is illustrated in the following diagram.

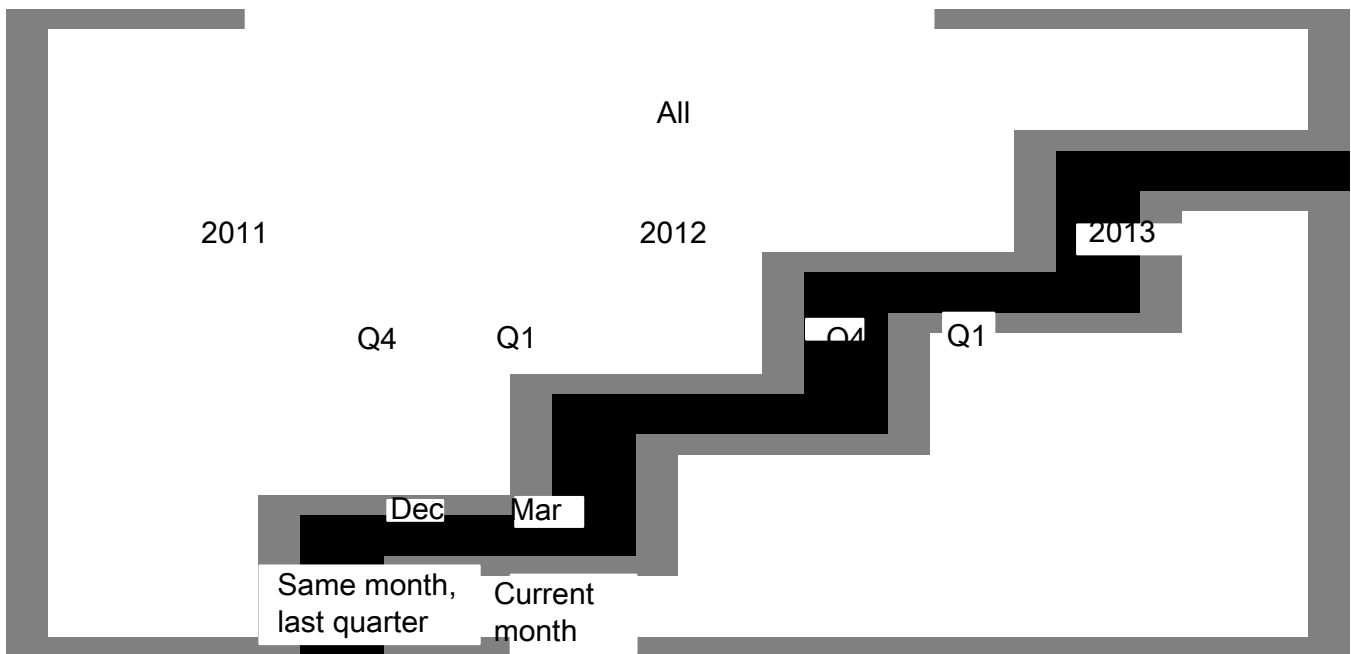


Figure 15. Illustration of single period example

Use a positive offset for a future period. For example, to define relative time member "Next month, next year" you specify:

- target period: month
- target period offset: 1
- context period: year
- context offset: 1

Custom period-to-date

Use custom period-to-date to define a relative time member that is an aggregation from the beginning of a time period to an endpoint within the period.

You must specify whether the period is life-to-date or for a specific to-date period. You then specify target period with offset, and context period with offset.

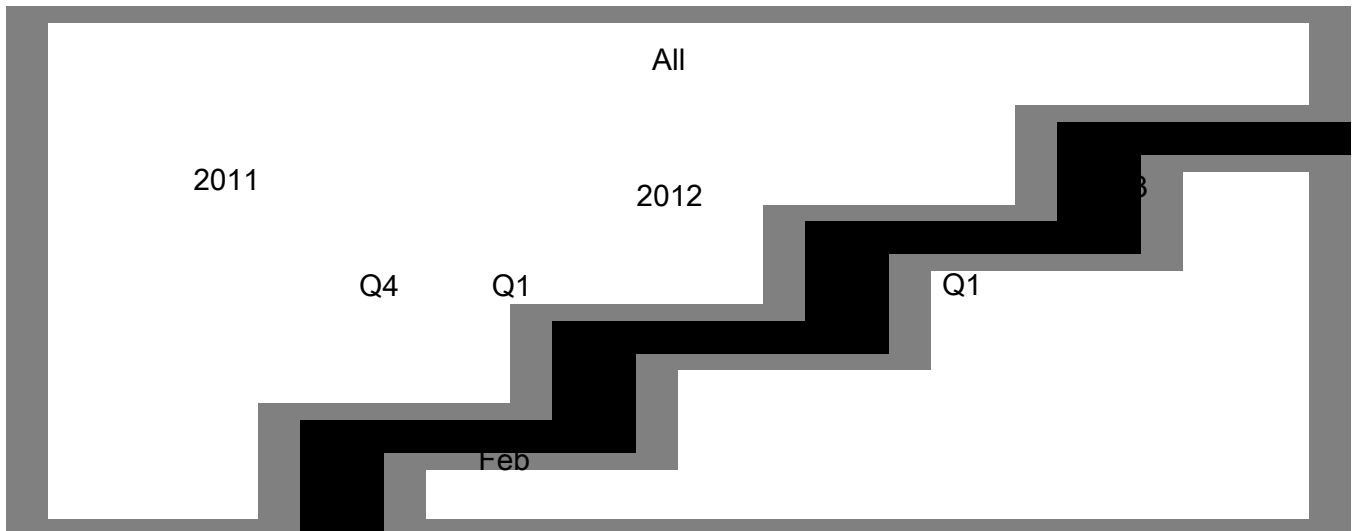
Life-to-date aggregates data for all time periods to a defined endpoint. The endpoint is defined by the target and context properties.

The target period that you specify affects the granularity of the period-to-date calculation. The calculation ends at the 'close' of the target period, where the close is the last sibling among the descendants. For example, if the current day is January 10, and day is the leaf level, quarter-to-date aggregates January 1 to January 10, if target period is day. If target period is month, quarter-to-date includes all days in the month, January 1 to January 31.

For example, assume a hierarchy with All, Year, Quarter, and Month levels. To define a relative time member "Quarter to date, last year" you specify:

- life-to-date: false
- to-date period: quarter
- target period: month
- target period offset: 0
- context period: year
- context offset: -1

This example is illustrated in the following diagram.



Quarter to date, last year

Figure 16. Illustration of custom period-to-date example

In this example, if the current month is February, and the quarter ends in March, the defined endpoint is February because the target period is month.

To define a relative time member "life to date (target = quarter)", you specify:

- life-to-date: true
- to-date period: n/a
- target period: quarter
- target period offset: 0
- context period: year
- context offset: 0

This example is illustrated in the following diagram.

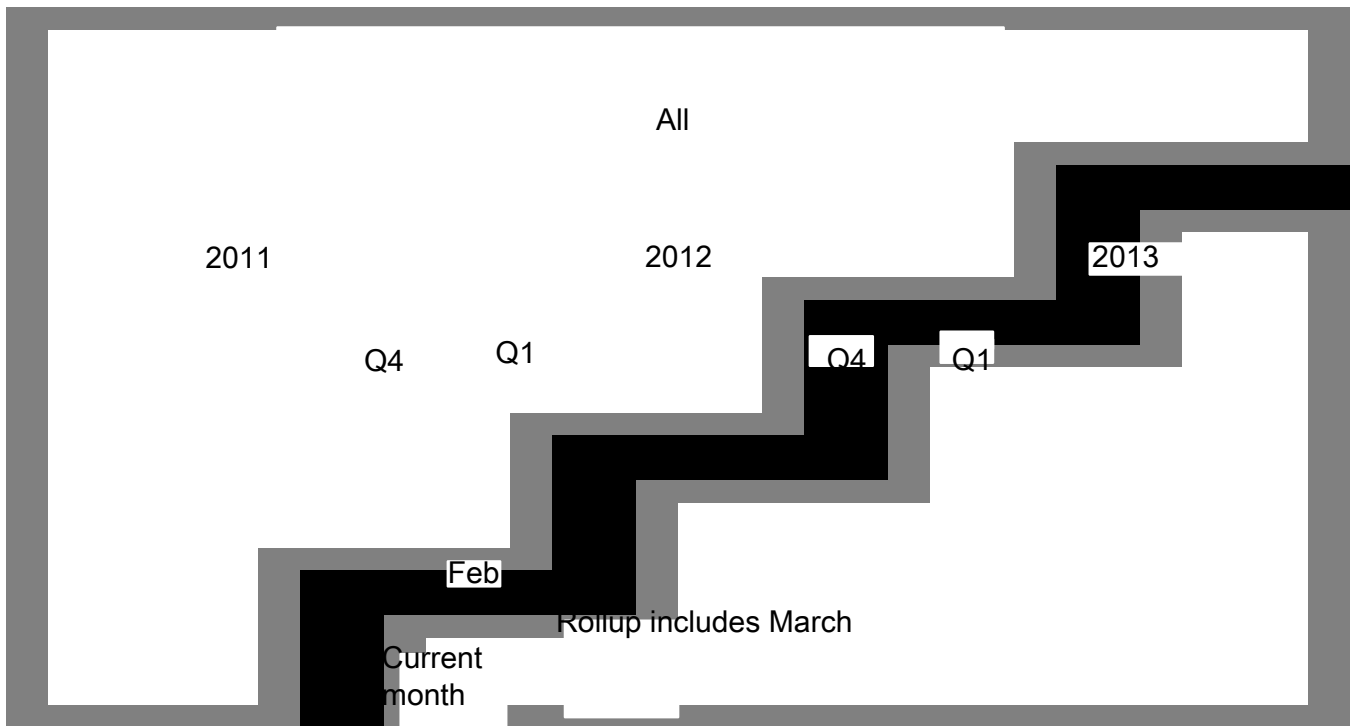


Figure 17. Illustration of life-to-date example

For life-to-date members, the sub tree of reference members is not generated in the member browser or in the IBM Cognos studios.

Custom N period running total

Use custom N-period running total to define a relative time member that is an aggregation of a defined number of consecutive periods.

You must specify the number of periods, target period with offset, and context period with offset. The endpoint is defined by the target and context properties.

For example, to define a relative time member for "Trailing 6 months, next year" you specify:

- number of periods: 6
- target period: months
- target period offset: -1
- context period: year
- context offset: 1

This example is illustrated in the following diagram.

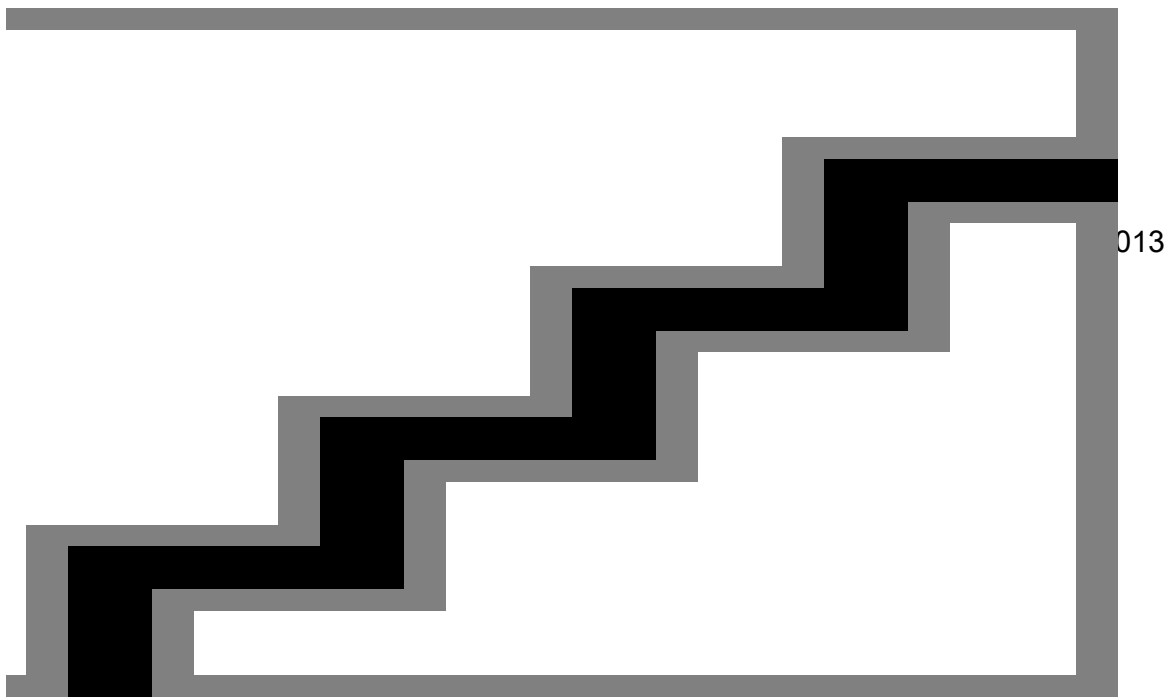


Figure 18. Illustration of custom N period running total example

You cannot select the highest level. For example, if levels are All, Year, Quarter, and Month, you cannot select Year as the target period.

Defining a relative time dimension

To use relative time, you define a dimension as a time dimension, modify time properties for the level and generate relative time members on a hierarchy by hierarchy basis.

Procedure

1. Select the location from which you want to create the dimension:

- To create a shared dimension at the project level, select **Model** from the **Project Explorer** tree.
- To create a dimension that is automatically linked to a dynamic cube, select the cube from the **Project Explorer** tree.

The dimension is also shared at the project level.

Tip: Create one relative time dimension and use it in all your dimensions to avoid conflicts between multiple time dimensions.

2. Click **New Dimension** .

The dimension contains a set of initial objects you need to complete the dimension.

3. On the **Properties** pane of the dimension, set **Dimension Type** to **Time**.

4. On the **Properties** pane of a hierarchy that belongs the dimension, set **Add relative time members** to **True**.

This enables generation of the predefined relative time members.

5. Build your desired level structure.

For more information about creating levels, see [“Defining a level ” on page 51.](#)

6. For each time level, select a **Level Type** .

The levels must appear in order in the hierarchy. For example, the Year, Month, Day levels cannot appear as Year, Day, Month. Use the **Periods** level type when the level does not conform to one of the predefined level types.

7. For each time level, enter an expression in the **Current Period** property.

For some examples of current period expressions, see [“Examples of level current period expressions” on page 88.](#)

8. With the time dimension selected, right-click, and select **Refresh Members**.

Predefined calculated members for relative time are added to the member tree.

Control the automatic generation of predefined relative time members

You can control the automatic generation of the following relative time members:

- prior period members
- next period members
- the sub tree of reference members for all relative time members

Reference members are relative time members that refer to regular members within a time hierarchy. They have the same caption and business key value as the members to which they refer.

The purpose of the reference members is to show the sub tree of a hierarchy to which a relative time member corresponds.

In the following example, you can see the first level of reference members that are highlighted in blue.

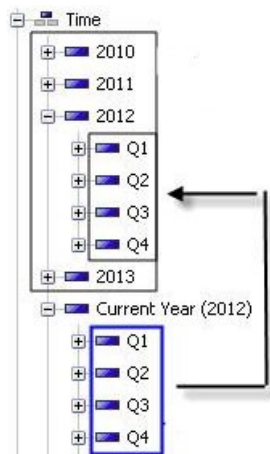


Figure 19. Example of first-level reference members

By default, the sub tree of reference members is generated. Depending on the hierarchy structure, there might be many reference members, and you can now exclude them from being auto-generated.

Procedure

1. From the **Project Explorer** tree, right click the hierarchy you want to work with, and then select **Open Editor**.
2. Select the **Relative Time** tab.
3. Select one of the following options for prior period members:
 - **Auto-generate members** to include predefined members (default).
 - **Do not auto generate members** to exclude predefined members.
4. Select one of the following options for next period members:
 - **Auto-generate members** to include predefined members.

- **Do not auto generate members** to exclude predefined members (default).
5. Select one of the following options for the **Reference relative time members subtree**:
- **Include** to include a sub tree of members (default).
 - **Exclude** to exclude a sub tree of members.

Creating a custom relative time member

You can create custom members in a relative time dimension.

Procedure

1. From the **Project Explorer** tree, right-click the hierarchy that you want to work with, and then select **Open Editor**.
2. Select the **Relative Time** tab.
3. Click one of the following options to create a custom relative time member:
 - **New Custom Single Period Definition**
 - **New Custom Period To Date Definition**
 - **New Custom N-Period Running Total Definition**
4. Complete the definition using the **Properties** tab.

Examples of level current period expressions

Some common examples of level current period expressions are defined in the following list.

The expressions resolve to the value of the business key of the member that you want to be the current member.

Year

```
extract( year, localtimestamp)
```

Half-Year

```
if(extract(month, localtimestamp) < 7) then
  (1)
else
  (2)
```

Quarter

```
'Q' || cast(
  if (extract(month, localtimestamp) <= 3) then (1)
  else ( if (extract(month, localtimestamp) <= 6) then (2)
  else ( if (extract(month, localtimestamp) <= 9) then (3)
  else (4) ) ) , varchar(1))
```

The `curent_timestamp` function returns Greenwich Mean Time while the `localtimestamp` function returns local time.

Month

```
extract(month, localtimestamp)
```

Week of Year

```
cast(extract(year, localtimestamp), varchar(4))
|| 'W' || cast(_week_of_year(localtimestamp), varchar(2))
```


Day of Year

```
cast(extract(year, localtime), varchar(4))  
|| 'W' || cast(_week_of_year(localtimestamp), varchar(2))
```

Day of Week

```
_day_of_week(localtimestamp, 7)
```

Day of Month

```
_days_between(localtimestamp, _first_of_month(localtimestamp)) + 1
```

Hour

```
extract(hour, localtime)
```

Week of Month

```
if( (_days_between( localtime , _first_of_month  
(localtimestamp)) + 1) >  
day_of_week(_first_of_month(localtimestamp), 7) )  
then (1)  
else (0)  
+  
if (((_days_between( localtime , _first_of_month  
(localtimestamp)) + 1)  
day_of_week(_first_of_month(localtimestamp), 7)) > 21)  
then (4)  
else(if (((_days_between( localtime , _first_of_month  
(localtimestamp)) + 1)  
- _day_of_week(_first_of_month(localtimestamp), 7)) > 14)  
then (3)  
else (if (((_days_between( localtime , _first_of_month  
(localtimestamp)) + 1)  
- _day_of_week(_first_of_month(localtimestamp), 7)) > 7)  
then (2)  
else (1)))
```

Multiple locales

You can add support for multiple locales to IBM Cognos dynamic cubes. Metadata object names and captions, dynamic cube object names, and member attribute names can be assigned different values in different locales. Then when a user switches between different content languages in IBM Cognos Analytics, names and captions are displayed in the appropriate language.

You use IBM Cognos Cube Designer to add support for multiple locales to a project, and you can then add metadata object names and member attribute names and captions in multiple languages. After adding multiple language support, you publish the dynamic cube in the normal fashion.

Selecting the design language and supported locales

When creating a project in IBM Cognos Cube Designer, the design language of the project defaults to the locale setting of the computer. You can change the default design language. Normally the default design language is the locale or language of the data in the database. After the design language is set, you can add other supported locales to the project.

Procedure

1. To change the design language, in the **Properties** tab of a project, click the value of the **Design Language** and select the design language from the drop-down list.
2. To add locales, in the **Properties** tab of a project, click **Add Locale(s)** and check the boxes next to the required locales.

Adding multiple locale names to metadata objects and dynamic cube objects

You can add names in multiple languages to metadata objects for supported locales.

Procedure

1. In **Project Explorer**, click a metadata object, such as a dimension, or a dynamic cube object, such as a measure.
2. In the **Properties** tab, click the value of the **Name** property.
The supported locales for the project are displayed.
3. For each supported locale, enter a name for the object in that language.
4. You can add additional locales to the project by clicking the **Add Language** button.
This adds locales to the project, not just to the selected object.
5. If the metadata object is a hierarchy, you can add language versions for the **Root Caption** property using the same steps.

Adding support for multiple locales to members and attributes

You add support for multiple locales for members and attributes by dimension. It is not necessary that all dimensions in a dynamic cube support multiple locales. IBM Cognos Dynamic Cubes supports dynamic cube definitions in which only some dimensions have members with multiple locales.

Before you begin

If you are adding multiple locales to attributes, the data source must contain a column for each locale associated with the attribute. For example, the **Great Outdoors Warehouse** data source has a **Product line** attribute in the **Products** dimension. This attribute has columns named PRODUCT_LINE_EN, PRODUCT_LINE_FR, and so on, for each of the supported locales in the database.

Procedure

1. In **Project Explorer**, click a dimension for which you want to add support for multiple locales.
2. In the **Properties** tab, click the value for **Multilingual Support** and select **By Column**.
You can now provide multilingual names for members and attributes.
3. Perform the following steps for each member in the dimension that you want to give names in multiple languages.
 - a) In **Project Explorer**, click a member in the dimension.
 - b) In the **Properties** tab, click the value of the **Name** property.
The supported locales for the project are displayed.
 - c) For each supported locale, enter a name for the member in that language.
 - d) You can add additional locales to the project by clicking the **Add Language** button.
4. Perform the following steps for each attribute in the dimension that you want to give names in multiple languages.
 - a) In **Project Explorer**, click an attribute in the dimension.
 - b) In the **Properties** tab, click the value of the **Name** property.
The supported locales for the project are displayed.
 - c) For each supported locale, enter a name for the attribute in that language.
 - d) You can add additional locales to the project by clicking the **Add Language** button.
 - e) In the **Properties** tab, change the value of the **Multilingual** property to **true**.
 - f) In the **Properties** tab, click the value of the **Column Name** property.
The supported locales for the project are displayed.

- g) Expand the data source in **Data Source explorer**, and drag the column associated with each locale into the respective **Column Name** value.

For example, the **Great Outdoors Warehouse** data source has a **Product line** attribute in the **Products** dimension. This attribute has columns named PRODUCT_LINE_EN, PRODUCT_LINE_FR, and so on, for each of the supported locales in the database. If you are enabling multilingual support for a dynamic cube that uses the **Product line** attribute in this database, you would drag the PRODUCT_LINE_EN column into the **Column Name** value for English, the PRODUCT_LINE_FR column into the **Column Name** value for French, and so on.

Chapter 9. Aggregate modeling

In IBM Cognos Cube Designer, you can model in-database aggregates within a dynamic cube when the imported data source for a dynamic cube contains fact tables with pre-aggregated data. You can also create user-defined in-memory aggregates that can be included in Aggregate Advisor recommendations.

Modeling in-database aggregates

You can model in-database aggregates within a dynamic cube when the imported data source for a dynamic cube contains fact tables with pre-aggregated data.

For more information about pre-defined aggregate fact tables, see [“Aggregate tables”](#) on page 33.

After publishing a dynamic cube that contains in-database aggregates, when you run queries on the cube data source, IBM Cognos Dynamic Cubes analyses these queries and redirects them to the appropriate aggregate table in the data source.

You must be familiar with the fact data in the data source to model an in-database aggregate. Understand which fact tables are set up as aggregates, and which detail tables the fact tables relate to.

Tip: It is good practice to prefix aggregate table names in the relational database with "Aggregate" so that you can easily identify them. You can also use the Relational Explorer to check the relationships between fact tables.

Before you can start modeling an in-database aggregate, you must set up the dynamic cube and aggregate tables by performing the following tasks:

1. For level-based hierarchies only, create the hierarchy levels required for aggregation if they do not exist in the dimension. For example, if an aggregate table in the data source summarizes data by quarter, the Date dimension must include a Quarter level.
2. For each aggregation level in the dimension, ensure that the required attributes and level unique keys are defined.
3. Aggregate tables must contain data at the highest level of aggregation used by the in-database aggregate so that you can roll up dimensions to the required level.

For example, if a Time dimension contains Year, Quarter and Month levels, and you want to roll up data up to the Year level in an in-database aggregate, the aggregate table usually contains data at the Year level.

If Cognos Dynamic Cubes cannot match a rollup level to an aggregate table, it uses an aggregate table defined at a particular level of aggregation to satisfy higher level aggregate requirements. For example, if you want to roll up the Time dimension to the Year level, and the aggregate table only contains data at the Quarter level, it uses this aggregate table and rolls it up to the higher levels.

The way in which you model an in-database aggregate depends on the data it contains:

- Simple aggregate table

With a simple aggregate table, all fact data and levels keys are contained in a single table, so no joins to dimension data are required.

The aggregate table can be joined to the same dimension tables as the detailed fact table or joined to aggregate dimension tables. Aggregate tables do not contain the same level of detail as the non-aggregate dimension tables.

- In-database aggregate with a parent-child dimension

A parent-child dimension does not have hierarchy levels. You create the relationships by mapping a single column in the aggregate table to the child key in the parent-child dimension.

You can partition data in an in-database aggregate by using aggregate slicers. Partitioning is possible where the data source contains a set of aggregate tables, each providing a subset of the data set available. For example, an aggregate table can contain sales data for specific dates.

The following table lists the properties that you can set when you model an in-database aggregate.

| <i>Table 27. Properties of an in-database aggregate</i> | |
|---|--|
| Property | Description |
| Name | The name of the in-database aggregate. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Comment | A comment or description of the in-database aggregate. |
| Remove non-existent tuples | This property is applicable to the dynamic cube only and must not be edited. |
| Ordinal | <p>The order in which the dynamic query mode server redirects queries to an in-database aggregate.</p> <p>If there is only a single in-database aggregate that can satisfy a query, the in-database aggregate is used.</p> <p>If there are multiple in-database aggregates that can satisfy a query, then the in-database aggregate with the lowest cardinality at the lowest level of aggregation (ordinal value) is selected.</p> <p>If there are multiple in-database aggregates with the same lowest ordinal value, then the in-database aggregate that is defined higher in the list in IBM Cognos Cube Designer is selected.</p> <p>For example, suppose you have the following in-database aggregates:</p> <ul style="list-style-type: none"> • in-database aggregate 1, cardinality of 100, ordinal value 1 • in-database aggregate 2, cardinality of 100, ordinal value 2 • in-database aggregate 3, cardinality of 50, ordinal value 3 • in-database aggregate 4, cardinality of 200, ordinal value 4 • in-database aggregate 5, cardinality of 100, ordinal value 1 <p>If a query can be satisfied by in-database aggregates 1, 2 or 3, then in-database aggregate 3 is selected because it has the lowest cardinal value.</p> <p>If a query can be satisfied by in-database aggregates 1, 2 or 4, then in-database aggregate 1 is selected because it has a lower ordinal value than in-database aggregate 2.</p> <p>If a query can be satisfied by in-database aggregates 1 or 5, then in-database aggregate 1 is selected because it is defined higher in the list in Cognos Cube Designer.</p> |

Defining an in-database aggregate automatically

You can automatically define an in-database aggregate when primary keys in the aggregate table match the level keys in dimensions of a dynamic cube. This allows you to create relationships between the dimensions and the aggregate table.

IBM Cognos Cube Designer can create these relationships automatically if the aggregate table contains the following:

- Measures that match the measures in the in-database aggregate.
- Dimensions that match the dimensions in the in-database aggregate.
- Data at the highest level of aggregation that is required by the in-database aggregate.

Procedure

1. Open the Cube editor for the dynamic cube in which you want to define an in-database aggregate.
2. Click the **Aggregates** tab.
3. Drag the required aggregate table from the **Data Source Explorer** to the **Aggregates** tab.

An in-database aggregation is created in the **Aggregates** tab. The cube also appears under the **In-Database Aggregates** folder in the **Project Explorer** tree. Where matching measures and dimensions are found in the in-database aggregate, Cognos Cube Designer maps each of these items to the aggregate table. Where possible, it also attempts to identify the highest level of aggregation that is required and roll up dimensions.

The ability to automatically map is dependent on how the aggregate tables are set up.

Results

The in-database aggregate is now complete. You can fine-tune the mapping by following step 4 onwards in the topic “Defining an in-database aggregate manually” on page 95. When you finish, you can test the validity of the in-database aggregate. For more information, see [“Validate a project and individual objects” on page 44.](#)

Defining an in-database aggregate manually

You define a in-database aggregate manually when an aggregate table uses level keys or is joined to a separate dimension that contains the required levels for aggregation. For example, to improve query performance, if a dimension table contains many records, you decide to create a dimension table that does not contain the lowest level members and contains only the level keys of its members. In this instance, you must map the relevant dimension in the in-database aggregate to a separate dimension aggregate table.

Procedure

1. Select the dynamic cube in which you want to define an in-database aggregate from the **Project Explorer** tree.

2. Click **New In-Database Aggregate** .

3. Select the measures and dimensions to include in the in-database aggregate, then click **OK**.

An in-database aggregate is created, which also appears under the **In-Database Aggregates** folder in the **Project Explorer** tree.

By default, each dimension is mapped to the lowest dimension level defined in the detail fact table. If aggregation occurs at a higher level in the aggregate table, you must roll up dimensions in the in-database aggregation to the correct level.

4. In the **Project Explorer** tree, double-click the in-database aggregate in the **In-Database Aggregates** folder.

The In-Database Aggregate editor is shown.

5. Click the dimension to roll up, and select the required level from the list of levels shown.

Repeat this step for each dimension you want to roll up.


For dimensions that are mapped to a separate dimension aggregate table, you must now map the level unique keys in the dimensions to columns in the required aggregate table.

6. In the In-Database Aggregate editor, click the **Key Mappings** tab.

7. For each level unique key, drag a column from the required aggregate table in the **Data Source Explorer** to the **Mapping** field.

Tip: If you drag a whole aggregate table, IBM Cognos Cube Designer attempts to automatically map all level unique keys.

Now you must map measures in the in-database aggregate to columns in the aggregate table.

8. In the In-Database Aggregate editor, click **Measures** .

The Measures editor is shown.

9. Map each measure to a column in the aggregate table by dragging a column from the required aggregate table in the **Data Source Explorer** onto the **Mapping** field.

For those dimensions where primary keys in the aggregate table match the level unique keys in dimensions of the dynamic cube, you can now create the relationships between dimensions and measures in the in-database aggregate.

10. In the **Project Explorer** tree, double-click the in-database aggregate in the **In-Database Aggregates** folder.

The In-Database Aggregate editor is shown.

11. For each dimension, click **Edit**, then select the dimension primary key and measure key to which it is joined.
12. If required, define the measure-to-dimension join in the **Join is at the lowest level of detail for the dimension** check box.

For more information about this check box, see [“Defining a measure-to-dimension join” on page 67.](#)

13. Click **OK**.

Results

The in-database aggregate is complete. You can now test the validity of the in-database aggregate. For more information, see [“Validate a project and individual objects” on page 44.](#)

Defining an in-database aggregate with a parent-child dimension

An in-database aggregate can contain a parent-child dimension. Because the dimension does not have hierarchy levels, you create the relationships by mapping a single column in the aggregate table to the child key in the parent-child dimension.

The in-database aggregate can also contain dimensions with level-based hierarchies. For more information about adding these dimensions, see [“Defining an in-database aggregate manually” on page 95.](#)

Procedure

1. Select the dynamic cube in which you want to define an in-database aggregate from the **Project Explorer** tree.

2. Click **New In-database Aggregate** .

3. Select the measures and parent-child dimension to include in the in-database aggregate, then click **OK**.

An in-database aggregate is created, which also appears under the **In-Database Aggregates** folder in the **Project Explorer** tree.

Now map a single column in the aggregate table to the child key in the parent-child dimension.

4. In the **Project Explorer** tree, double-click the in-database aggregate in the **In-Database Aggregates** folder.

The In-Database Aggregate editor is shown.

5. Select the parent-child dimension, then select the **I want to remap the columns for this dimension, as they are included in my aggregate** check box.
6. Click the **Key Mappings** tab.
7. Drag a column from the required aggregate table in the **Data Source Explorer** to the **Mapping** field for the child key.

Next, you must map measures in the in-database aggregate to columns in the aggregate table.

8. In the In-Database Aggregate editor, click **Measures** .

The Measures editor is shown.

9. Map each measure to a column in the aggregate table by dragging a column from the required aggregate table in the **Data Source Explorer** onto the **Mapping** field.

Results

The in-database aggregate is complete. You now test the validity of the in-database aggregate. For more information, see [“Validate a project and individual objects” on page 44.](#)

Filtering data using an aggregate slicer

You can filter the data in an in-database aggregate using aggregate slicers. Filtering is possible where the data source contains a set of aggregate tables, each one providing a subset of the data set available. For example, a data warehouse might contain five years of sales data, and also contain aggregate tables with sales data summarized for each quarter.

Procedure

1. Define the in-database aggregate you require.

For more information, see [“Defining an in-database aggregate automatically” on page 94](#), [“Defining an in-database aggregate manually” on page 95](#), and [“Defining an in-database aggregate with a parent-child dimension” on page 96](#).

2. Double-click the in-database aggregate in the **Project Explorer** tree, then click the **Slicers** tab.
3. Select the data to include in the filter by dragging and dropping members from the **Members** folder in the **Project Explorer** tree to the **Member Slicers** field.

Note: All selected members must come from a single hierarchy level.

Results

The in-database aggregate is complete. You can now test the validity of the in-database aggregate. For more information, see [“Validate a project and individual objects” on page 44.](#)

Creating user-defined in-memory aggregates

User-defined in-memory aggregates provide the dynamic cube modelers with the ability to suggest in-memory aggregates to be included in the Aggregate Advisor recommendations.


User-defined in-memory aggregates might reduce the time to optimize dynamic cubes. However, the modelers should understand how these aggregates affect the performance and memory utilization of a dynamic cube. As a best practice, the modelers should continue running Aggregate Advisor to obtain recommendations for defining in-memory aggregates.

User-defined in-memory aggregates do not have a limit on the size of the dimensional space that they can encompass and they can span any dimensional space. This results in an increased memory footprint per value stored in the aggregates as the size of the dimensional space grows, which is independent of the actual number of values in the dimensional space.

User-defined in-memory aggregates support all measure types except for semi-additive measures. Non-distributive measures, such as Average, can also be included in the aggregate. However, these types of measures can only be used if the query is an exact match of the aggregate.

After you publish a cube with the user-defined in-memory aggregate to IBM Cognos Analytics, use Aggregate Advisor to generate a recommendation with the user-defined aggregate and apply the aggregate to the dynamic cube for usage. You can generate the recommendations for the user-defined in-memory aggregate together with the recommendations for other types of in-memory aggregates.

Procedure

1. In the **Project Explorer** tree, locate the dynamic cube for which you want to create the user-defined in-memory aggregate.
2. Double-click the cube to open its editor.
3. Click the **Aggregates** tab.
4. In the **User-Defined In-Memory Aggregates** section, click the **New User-Defined In-Memory Aggregates**  icon.
5. Select the measures and dimensions that you want to include in the aggregate, and click **OK**.

The new aggregate appears in the **User-Defined In-Memory Aggregates** section as **New User-Defined In-Memory Aggregate**.

6. Click the **New User-Defined In-Memory Aggregate**, and in the properties box, rename it as required.
7. Double-click the new aggregate, and for each dimension in the aggregate, select the levels in the hierarchies to include in the aggregate. Do this in the following way:
 - a) On the **Dimensions** tab, click on a dimension to see the hierarchies for that dimension.
 - b) In the hierarchies, select the levels to which you want to apply this aggregate. If you select a lower level, all higher levels are automatically selected.

If you do not select at least one level from one of the hierarchies in each dimension, validation errors occur when you try to publish the cube.

8. To quickly go back to the cube editor view, click the **Aggregates** breadcrumb in the **Project** navigation bar. At this point, you can create another user-defined in-memory aggregate, or edit or delete an existing aggregate.
9. Save the dynamic cube and then publish it to IBM Cognos Analytics. For more information, see [“Deploying and publishing dynamic cubes” on page 69](#).
10. Run Aggregate Advisor to obtain recommendations for the user-defined in-memory aggregates.

In the general options screen, select the **In-memory aggregates** option. For **Query Workload Information**, you can select any option. However, if you need recommendations only for the user-defined in-memory aggregates, the **User Defined Only** option returns them faster than the other options. For more detailed information about using Aggregate Advisor, see the *IBM Cognos Dynamic Query Analyzer User Guide*.

11. In IBM Cognos Administration, set the property **Maximum space for in-memory aggregates (MB)** to a value larger than the estimated size of the recommendations. For more information about this property, see [“Setting dynamic cube properties” on page 132](#).
12. Using Aggregate Advisor, apply the user-defined in-memory aggregates to the dynamic cube.

Results

The user-defined in-memory aggregates will be available for report queries after the in-memory aggregates are loaded. The aggregates start loading after the dynamic cube is started and they might take some time to load.

Chapter 10. Virtual cube modeling

Using IBM Cognos Cube Designer, you can model virtual cubes in a project.

For information about using virtual cubes, see “Virtual cubes” on page 30.




The following table lists the properties that you can set when modeling a virtual cube.


| Property | Description |
|----------------|---|
| Name | The name of the virtual cube. This is also used as the name of the data source that represents the cube. If the project supports multiple locales, there can be versions of the name in all supported languages. Tip: When creating a Framework Manager package for the virtual cube, select this name from the list of data sources. |
| Comment | A comment or description of the virtual cube. Comments are not visible in the IBM Cognos studios. |
| Merge Operator | The method used to aggregate data in the source cubes. Default: Sum The cube merge operator is the default merge operator for all virtual measures and virtual members. You can also define a merge operator for a specific virtual measure or virtual member which overrides the cube merge operator. |

Defining a virtual cube

You define a virtual cube at the project level.

Procedure

1. Select a namespace in the **Project Explorer** tree.
2. Click **New Virtual Cube** .
3. Select a maximum of two source cubes to merge into a virtual cube. You can include dynamic cubes from the current project, and dynamic cubes or virtual cubes deployed as data sources to the content store:
 - To include a dynamic cube from the project, select the cube from the list.
 - To include a dynamic cube or virtual cube from the content store, click **Add Content Store Cube**, select the required data source, then click **OK**.
4. Click **OK**.
5. Complete the virtual cube definition by using the **Properties** tab.
You can view the source cubes from which the virtual cube is derived.
6. From the **Project Explorer** tree, right-click the virtual cube and select **Open Editor**. You can perform the following tasks from here:
 - To add a source cube, click **Add Source Cube** .
 - To delete a source cube, select the cube name, and click **Delete** .

- To view the virtual measure dimension, click **Measures** .

What to do next

You can now fine-tune virtual objects and define further objects as required. For more information, see [“Model virtual dimensions” on page 100](#), [“Model virtual hierarchies” on page 101](#), [“Viewing virtual levels” on page 102](#), [“Model virtual members” on page 103](#), and [“Model virtual measures” on page 104](#).

You can also add calculated measures or calculated members to a virtual cube. For more information, see [“Calculated members” on page 73](#).

When you finish, you can test the validity of the virtual cube to check for errors, and then deploy and publish the virtual cube. For more information, see [“Validate a project and individual objects” on page 44](#) and [“Deploying and publishing dynamic cubes” on page 69](#).

Tip: If a virtual cube contains a source cube deployed as a data source to the content store, the data source must be started before you can deploy the virtual cube.

Model virtual dimensions

When you create a virtual cube, IBM Cognos Cube Designer adds dimensions from the source cubes to the virtual cube.

Dimensions with identical names in the source cubes (conformed dimensions) are added to a virtual cube as merged virtual dimensions. Non-conformed dimensions are added to a virtual cube as new virtual dimensions. For examples of the merging process, see [“Virtual cubes” on page 30](#).

If a virtual dimension is not merged correctly, or could not be automatically merged, you can manually merge two source dimensions. You can also delete redundant virtual dimensions.

When merging dimensions in a virtual cube, it is not possible to map a source dimension to more than one virtual dimension.

The following table lists the properties that you can set when modeling a virtual dimension.


| Property | Description |
|-------------------|---|
| Name | The name shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Comment | A comment or description of the virtual dimension. Comments are not visible in the IBM Cognos studios. |
| Default Hierarchy | The hierarchy to use when no hierarchy been specified for a dimension used in an expression. Applies only when multiple hierarchies are defined for a dimension. |
| Dimension Type | Regular (default) - Identifies a regular dimension. Time - Identifies a time dimension. For more information about relative time dimensions, see “Defining a relative time dimension” on page 86 . |

Defining a virtual dimension

Using IBM Cognos Cube Designer, you can define virtual dimensions within a virtual cube.

Procedure

1. From the **Project Explorer** tree, right-click the virtual cube and select **Open Editor**. The editor tab shows the following columns:
 - Virtual dimensions - the virtual dimensions added to the virtual cube.
 - Dimensions - the dimensions in the source cubes to which the virtual dimension is mapped.
2. To manually merge source dimensions into a new virtual dimension, follow these steps:
 - a) Click **Add Virtual Dimension**.
 - b) Click **Editor** for the source dimension column related to the new virtual dimension, then select a source dimension, and click **OK**.

Tip: If you cannot select a source dimension because it is already mapped to a different virtual dimension, you must first delete the source dimension from the other virtual dimension.
 - c) Repeat the step b for the second blank source dimension.
3. You can also perform the following tasks from here:
 - To delete a source dimension from a virtual dimension, select the source dimension, and click **Delete** .
 - To delete a virtual dimension from a virtual cube, select the virtual dimension, and click **Delete**.
4. To complete the definition of a virtual dimension, select the virtual dimension in the **Project Explorer** tree to display the **Properties** tab.

Model virtual hierarchies

When you create a virtual cube, IBM Cognos Cube Designer adds hierarchies from the source cubes to the virtual cube.

Hierarchies with identical names in the source cubes (conformed hierarchies) are added to a virtual cube as merged virtual hierarchies. Non-conformed hierarchies are added to a virtual cube as new virtual hierarchies. For examples of the merging process, see [“Virtual cubes” on page 30](#).

If a virtual hierarchy is not merged correctly, or could not be automatically merged, you can manually merge two source hierarchies. You can also delete redundant virtual hierarchies.

When merging hierarchies in a virtual cube, it is not possible to map a source hierarchy to more than one virtual hierarchy.

The following table lists the properties that you can set when modeling a virtual hierarchy.

| Property | Description |
|-----------------|--|
| Name | The name shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Comment | A comment or description of the virtual dimension. Comments are not visible in the IBM Cognos studios. |
| Default Member | The member value to use when evaluating member expressions, where no value is specified for a hierarchy. If the default member is empty, the root member of the hierarchy is used. To set a default member, drag the required member from the Members folder in the Project Explorer tree. |

Table 30. Properties of a virtual hierarchy (continued)


| Property | Description |
|---------------------------|---|
| Parent-Child | True - Indicates that the hierarchy uses a parent-child structure. False - Indicates that the hierarchy does not use a parent-child structure. This property cannot be edited. |
| Add Relative Time Members | False (default) - The hierarchy does not belong to a Time dimension. True - The hierarchy belongs to a Time dimension. For more information, see “Defining a relative time dimension” on page 86. |


Defining a virtual hierarchy

Using IBM Cognos Cube Designer, you can define virtual hierarchies within a virtual cube.

Procedure

- From the **Project Explorer** tree, right-click the virtual dimension for which you want to define virtual hierarchies and select **Open Editor**. The editor tab shows the following columns:
 - Virtual hierarchies - the virtual hierarchies added to the virtual dimension.
 - Hierarchies - the source hierarchies in the source cubes to which the virtual hierarchy is mapped.

Tip: If the virtual dimension was created from one source dimension only (not merged), only one source hierarchy column is shown.
- To manually merge source hierarchies into a new virtual hierarchy, follow these steps:
 - Click **Add Virtual Hierarchy** .
 - Click **Editor** for the source hierarchy column related to the new virtual dimension, then select a source hierarchy, and click **OK**.

Tip: If you cannot select a source hierarchy because it is already mapped to a different virtual hierarchy, you must first delete the source hierarchy from the other virtual hierarchy.
 - Repeat the step b for the second blank source hierarchy.
- You can also perform the following tasks from here:
 - To delete a source hierarchy from a virtual hierarchy, select the source hierarchy, and click **Delete** .
 - To delete a virtual hierarchy from a virtual cube, select the virtual hierarchy, and click **Delete**.
- To complete the definition of a virtual hierarchy, select the virtual hierarchy in the **Project Explorer** tree to display the **Properties** tab.

Viewing virtual levels

When you create a virtual cube, IBM Cognos Cube Designer adds levels from the source cubes to the virtual cube.

Source cubes containing identical levels in a hierarchy are merged as virtual levels. If levels in the source cubes are not identical, level names from the first source cube are used as the names of the virtual levels. If one source cube contains more hierarchy levels than the second source cube, the extra levels are added as the lowest levels of the virtual hierarchy. For examples of the merging process, see [“Virtual cubes”](#) on page 30.

Procedure

From the **Project Explorer** tree, right-click the virtual hierarchy for which you want to view virtual levels and select **Open Editor**. The editor tab shows the following columns:

- Virtual levels - the virtual levels added to the virtual hierarchy.
- Levels - the source levels in the source cubes to which the virtual level is mapped.

Tip: If the virtual hierarchy was created from one source hierarchy only (not merged), only one source level column is shown.

Model virtual members

When you create a virtual cube, IBM Cognos Cube Designer adds members from the source cubes to the virtual cube.

For a virtual hierarchy that is merged from two conformed dimensions, all hierarchy members from the source cubes are available as virtual members. If the level key for each source member is identical, members are added to the virtual cube as merged virtual members. Any members that do not have matching level keys are added to the virtual cube as new virtual members. For examples of the merging process, see [“Virtual cubes” on page 30](#).

If a virtual member is not merged correctly, or could not be automatically merged, you can manually merge two source members. You can also delete redundant virtual members.

When manually merging virtual members, if member names do not match, a new virtual member is created using this format: <source member 1?source member 2>. For example, two source cubes contain a Time hierarchy. Source cube 1 contains one member: All. Source cube 2 contains one member: All_Time. The virtual member created is All?All_Time.

Tip: Merged virtual names are required for internal member unique names (MUNs) only, and are not visible to report users.

The following table lists the properties that you can set when modeling a virtual member.

| Property | Description |
|----------------|---|
| Name | The name that appears in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Comment | A comment or description of the virtual member. Comments are not visible in the IBM Cognos studios. |
| Merge Operator | The method used to aggregate virtual members in the source cubes. By default, the merge operator is set to the same method that is defined for the virtual cube. |
| Precedence | The merge operator to use if a tuple contains virtual members with different merge operators. The merge operator with the highest precedence is used. If there are two or more merge operators with the same precedence, the merge operator for the first virtual member in the tuple is used. Default: 0 |

The following table lists the properties that you can set when working with a source member.

Table 32. Properties of a source member

| Header | Header |
|---------|--|
| Name | The name that appears in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Include | <p>Controls whether the source member is included in the virtual cube.</p> <p>If the same member exists in two source cubes, and you exclude the member from both source cubes, the member is excluded from the virtual cube.</p> <p>If the member is excluded from one source cube only, the member is included in the virtual cube.</p> <p>Default: True</p> |

Defining a virtual member


Using IBM Cognos Cube Designer, you can model virtual members within a virtual cube.

Procedure

1. From the **Project Explorer** tree, right-click the virtual hierarchy for which you want to define virtual members and select **Open Editor**.
2. Select the **Members** tab. The editor tab shows the following columns:
 - Virtual members - the virtual members added to the virtual hierarchy.
 - Members - the source members in the source cubes to which the virtual level is mapped.

Tip: If the virtual hierarchy was created from one source hierarchy only (not merged), only one source member column is shown.
3. To manually merge source members into a new virtual member, follow these steps:
 - a) Click **Add Virtual Member**.
 - b) Click **Editor** for the source member column related to the new virtual member, then select a source member, and click **OK**.

Important: To see the list of source members in a hierarchy, the source cube must be deployed as data source to the content store and started.

Tip: If you cannot select a source member because it is already mapped to a different virtual member, you must first delete the source member from the other virtual member.
 - c) Repeat the step b for the second blank source dimension.
4. You can also perform the following tasks from here:
 - To delete a source member from a virtual member, select the source member, and click **Delete** .
 - To delete a virtual member from a virtual cube, select the virtual member, and click **Delete**.
5. To complete the definition of a virtual member, select the virtual member to display the **Properties** tab.

Model virtual measures

When you create a virtual cube, IBM Cognos Cube Designer adds measures from the source cubes to the virtual cube.

Measures with identical names in the source cubes are added to a virtual cube as merged virtual measures. Measures that do not have identical names or that exist in only one of the source cubes

are added to a virtual cube as new virtual measures. For examples of the merging process, see [“Virtual cubes”](#) on page 30.

If a virtual measure is not merged correctly, or could not be automatically merged, you can manually merge two source measures. You can also delete redundant virtual measures.

When merging measures in a virtual cube, it is not possible to map a source measure to more than one virtual measure.

The following table lists the properties that you can set when modeling a virtual measure.

| <i>Table 33. Properties of a virtual measure</i> | |
|--|--|
| Property | Description |
| Name | The name shown in the IBM Cognos studios. If the project supports multiple locales, there can be versions of the name in all supported languages. |
| Comment | A comment or description of the virtual dimension. Comments are not visible in the IBM Cognos studios. |
| Visible | <p>Controls whether the measure is visible in the published package.</p> <p>Non-visible measures are typically used to represent intermediate values. These members are not intended to be used for direct reporting. However a non-visible measure is always present in the published package because the measure might be needed by other objects in a virtual cube.</p> <p>Non-visible measures are not displayed in the metadata browser and are removed from the output of reports which contain references to them. For example, a report that references a non-visible measure does not include output from that object.</p> <p>Default: True</p> |
| Merge Operator | <p>The method used to aggregate virtual measures in the source cubes.</p> <p>By default, the merge operator is set to the same method that is defined for the virtual cube, but you can override it.</p> |
| Precedence | <p>The merge operator to use if a tuple contains virtual measures with different merge operators.</p> <p>The merge operator with the highest precedence is used. If there are two or more merge operators with the same precedence, the merge operator for the first virtual measure in the tuple is used.</p> <p>Default: 0</p> |
| Data Format | Set the default data properties for each type of data. |

Defining a virtual measure

Using IBM Cognos Cube Designer, you can define virtual measures within a virtual cube.

Procedure

1. From the **Project Explorer** tree, right-click the virtual measure dimension and select **Open Editor**.

The editor tab shows the following columns:

- Virtual measures - the virtual measures added to the virtual dimension.
- Measures - the source measures in the source cubes to which the virtual measure is mapped.

2. To manually merge source measures into a new virtual measure, follow these steps:


a) Click **Add Virtual Measure** .

b) Click **Editor** for the source measure column related to the virtual measure, then select a source measure, and click **OK**.

Tip: If you cannot select a source measure because it is already mapped to a different virtual measure, you must first delete the source measure from the other virtual measure.

c) Repeat the step b for the second blank source measure.

3. You can also perform the following tasks from here:

- To a source measure from a virtual measure, select the source measure, and click **Delete** .

- To delete a source measure dimension (including all measures) from a virtual cube, select the source measure dimension, and click **Delete**.

- To delete a virtual measure from a virtual cube, select the virtual measure, and click **Delete**.

4. To complete the definition of a virtual measure dimension or a virtual measure, select the required object in the **Project Explorer** tree to display the **Properties** tab.

Chapter 11. Define security

You can define security on hierarchy basis for a dynamic cube. Security is used to control the metadata that is available to specific users or user groups in the IBM Cognos studios. For example, if a dynamic cube includes a Geography hierarchy with two members, such as Canada and Europe, you can secure all members for Europe so that it is only accessible to certain users.

To define security, complete the following tasks as required:

- Define which members to secure in your hierarchies by creating one or more security filters for them.
You can add security rules after modeling hierarchies in a project. They are independent of any dynamic cube.
- Apply security filters to a dynamic cube by creating one or more security views for them.
- Define which dimensions, attributes, and measures to secure in a dynamic cube by adding them to the security views.
- Publish a dynamic cube to the content store.

Tip: IBM Cognos Cube Designer validates security definitions when you validate or publish a dynamic cube.

After you publish a dynamic cube to the content store, you must complete the following tasks in IBM Cognos Administration:

- Assign users, groups, and roles to security views.

This step is required if you are using role-based security filters.

If you are using lookup filter-based security filters only, access rights are defined in the lookup tables, so it is only necessary to assign Read access to the user group named Everyone.

For more information on role-based security filters and lookup table-based security filters, see [“Security filters for hierarchy members”](#) on page 107.

- If you make further changes to security to a dynamic cube that is already started, refresh the security settings of the dynamic cube on the query service.

For more information on performing administration tasks, see [Chapter 12, “Cognos dynamic cubes administration,”](#) on page 117.

Security for virtual cubes

Define security in source cubes as required. Virtual cubes automatically inherit the security settings that are defined in source cubes to maintain consistent security rules.

Security filters for hierarchy members

Secure members in a hierarchy by using a security filter. A security filter specifies whether you are granting or denying users access to one or more members.

You can add security rules after modeling hierarchies in a project. They are independent of any dynamic cube.

Every hierarchy in IBM Cognos Cube Designer contains a default security filter that is named **All Members Granted**. This option explicitly grants access to all hierarchy members. You can define further security filters as required.

There are two types of security filter you can create:

- Lookup table-based security filter

If security rules for users are stored in a relational database table, you can import the data source and use the lookup table in a security filter.

- Role-based security filter

You can also manually define security rules, for example, where no suitable lookup tables exist.

You can also combine lookup table-based and role-based security filters. For example, you can restrict access to sales data to the Sales Employees user group by using a security view, and then use IBM Cognos Administration to further restrict access for each salesperson in the lookup table.

For each filter, you must specify the scope to indicate whether you are explicitly granting or denying access to hierarchy members. You then complete the filter as follows:

- If you are defining a role-based security filter, you use a dynamic query mode expression to specify the required hierarchy members to include in the filter.
- If you are defining a lookup table-based filter, you specify which lookup table columns contain keys for hierarchy members from each level. You then use an expression to select the rows from the lookup table that are relevant to the user executing the query.

You can include macro expressions to match the user information in the lookup table to the current user information. An example is shown here:

```
( User Name = #sq($account.personalInfo.userName)#) and ( Security Type = 'grant') and ( Security Scope = 'self_and_descendant')
```

Important: A valid expression must return a set of hierarchy members.

In IBM Cognos Dynamic Cubes security, the deny scope has precedence over the grant scope. After a member is explicitly denied, it cannot be accessed. You can use a combination of deny filters to further restrict user access to the members in a hierarchy.

The following table describes the scope options that you can use when you are defining a security filter.

| <i>Table 34. Security filter scope options</i> | |
|---|--|
| Scope | Description |
| Grant Members | Explicitly grant access to specified hierarchy members. Report users can see only the specified hierarchy members and associated values. Using a grant scope without the Ancestors option can lead to visible ancestors. |
| Grant Members and Descendants | Explicitly grant access to hierarchy members and all their descendants. Report users can see only the specified hierarchy members and associated values. Using a grant scope without the Ancestors option can lead to visible ancestors. |
| Grant Members and Ancestors | Explicitly grant access to hierarchy members and all their ancestors. Report users can see only the specified hierarchy members and associated values. |
| Grant Members, Descendants and Ancestors | Explicitly grant access to hierarchy members together with all their descendants and ancestors. Report users can see only the specified hierarchy members and associated values. |
| Deny Members and Descendants | Explicitly deny access to hierarchy members and all their descendants. Report users cannot see the specified hierarchy members and associated values. |

When you set up a security filter, you must consider the following points:

- When you explicitly grant access to a hierarchy member, report users can see only that member and its associated values. Users are denied access to all other hierarchy members.

For example, the Geography hierarchy contains the following members: All, Canada, and Europe. If you grant access to the All member only, users cannot see Canada or Europe.

- When you explicitly grant access to a hierarchy member by using the **Grant Members** option or the **Grant Members and Descendants** option, report users can also see the ancestor members, but not their values.

The values of these visible ancestor members are shown as ERR to differentiate them from a true Null value. The use of visible ancestors ensures that there is a path from a root member of the hierarchy to any granted members. Without a path from a root member to granted members, the IBM Cognos Studios cannot properly display members. Because Cognos Dynamic Cubes does not support visual totals, visible ancestors ensure that rollup values do not reveal information about secured descendants.

- When you explicitly deny access to a hierarchy member, access to all other members in the hierarchy, except descendant members, is implicitly granted.
- When you explicitly deny access to a hierarchy member, access to all descendant members is also denied.

If the result of this option is an unbalanced or ragged hierarchy, padding members are used to balance the hierarchy. For more information, see [“Padding members” on page 18](#).

- If a security filter is set up with a grant or deny scope option, but does not contain an expression, then no members are granted or denied.
- If a security filter contains references to a member that cannot be resolved, the member reference is ignored.

If a member reference cannot be resolved because the member does not exist in a hierarchy, the security filter is still valid.

If it cannot be resolved because the filter contains an invalid expression, an error occurs and access to the full hierarchy is denied.

- If an error occurs as a result of applying a security filter, when a user opens a package or runs a report, an error message is shown because access to the whole hierarchy is automatically denied.

Secured padding members

The use of secured padding members ensures that the hierarchies remain balanced. Balanced, non-ragged hierarchies have better performance in the studios. Secured padding members are inserted into a secured hierarchy member tree when a granted member has all its child members restricted. This scenario is most common with the **Grant Members** option, when descendants are not included in the scope. However, it can also occur with deny filters or with a combination of grant and deny filters.

Consider the following points:

- If all descendants of a non-leaf member are restricted, then secured padding members are inserted into all the levels below the non-leaf member.
- If all leaf members are restricted, padding members are inserted; the leaf level is not removed.
- The caption of secured padding members is either empty or blank or has the name of the parent. This is the same configuration setting for caption of padding member in ragged and unbalanced hierarchies.
- Secured padding members are secured similar to visible ancestors.
- Intrinsic properties of secured padding members are accurate, however member properties are null.
- There is at most one secured padding member for each level under a parent member.

Aggregated data in a secured dynamic cube

When you grant access to hierarchy members, it is possible that report users might inadvertently infer member values to which they are denied.

For example, suppose that you have a Geography hierarchy with these members and values: All (100), Canada (30), Europe (70). Using the **Grant Members and Ancestors** option, access is explicitly granted

to Canada and its parent (All). Report users can see All (100), and Canada (30). If report users are aware that Europe is the only one other hierarchy member, they can infer that its value is 70.

Default members

When a hierarchy is secured, a new default member on the hierarchy can be specified for the user. For example, if a single member and descendants is granted access, the default member can be modified. In this scenario, the single member is used as the new root of the hierarchy, although the member might not be at the root level.

The following steps determine the correct default member for a secured hierarchy:

- The original default member is checked to ensure that it is not restricted and not a visible ancestor. If the original default member is unsecured, then it remains the default member.
- A breadth first search of the hierarchy is done to find the first level with an unsecured member.
 - If the first level with an unsecured member has only the one unsecured member, then the unsecured member is the new default member.
 - If the first level with an unsecured member has more than one unsecured member, or also has a visible ancestor on the level, then their common ancestor is the new default member. In some cases, this common ancestor can be a visible ancestor. In the case of a visible ancestor as a default member, any time a non-visible ancestor member is not the context in the report, the visible ancestor, whose value is always ERR, will be the context.

Any time a hierarchy with a visible ancestor as the default member is not explicitly included in the report, the default member is used in the context, and ERR is the cell values.

Data caching using default members

The same report run by a user with all access and a user with security policies will normally hit the same cache. In general, the secured user needs only a subset of the members that the unsecured user used, because security limits access to the members. However, when the default member differs between the two users, the slice of the cube differs, and a different section of the cache might be required.

The following example shows a cross-tab report of All Product against All Time on Quantity. The security views have the Branches hierarchy secured, but the Branches hierarchy is not included in the report. The default member for the Branches hierarchy is the slicer for the report.

In the case of the unsecured user, with a built-in scope of Grant All Members, the report uses the default member, All Branches, for the context of the Branches hierarchy. The tuple value searched for in the data cache is All Time, All Products, All Branches, Quantity.

| <i>Table 35. Example of a cross-tab report using a default member of All Branches</i> | |
|---|---------------------|
| Quantity | All Products |
| All Time | 89,237,091 |

For the secured user that is assigned to a security view with a scope of Grant United States and descendants, the report uses the default member of United States, for the context of the Branches hierarchy. The tuple searched for in the data cache is (All Time, All Products, United States, Quantity). This differs from the unsecured user's tuple.

| <i>Table 36. Example of a cross-tab report using a default member of United States</i> | |
|--|---------------------|
| Quantity | All Products |
| All Time | 10,444,575 |

Because the tuples are not the same, reports that are run by one user would not populate the tuple value in the data cache of the other. Also, because the Branches context is at different levels in the two tuples, the query structure to access the values in the underlying data source differs.

Secure calculated members

To secure calculated members, the members must be explicitly included in the dynamic query expression. Drag the calculated members into the expression editor to create a set expression that resolves to a set of members to be secured. For example, if you want to secure calculated members A1 and A2, drag them into the editor and create an expression such as SET(A1, A2). Functions such as MEMBERS do not return calculated members that are present.

A calculated member is not accessible unless its parent member is accessible.

It might be possible that a calculated member definition references a secured member or measure. If a calculated member references a secured measure, a query with the calculated member returns the following exception: XQE-V5-0005 Identifier not found '[gosales_dw].[Measures].[Unit Sales]'.

If the calculated member references a secured member, the value of the secured member is treated as null in the calculation.

Security filters based on a lookup table

If security rules for users are stored in a relational database lookup table, you can reference the lookup table in a security filter.

When you define a security filter, you specify the hierarchy levels on which to secure members. To specify the hierarchy levels, you map the level keys to one or more query items. You are not required to map all levels in a hierarchy. You map only those levels that you want to secure, and for which there is data in the lookup table. For each level with a multi-part level key, for example YearMonth for the Month level, you must map query items to all parts of the key.

The required combination of mapped query items depends on whether the level keys are unique or non-unique.

For example, suppose you have a Dates hierarchy with Year, Month, and Day levels, and you want to filter members at the Month level.

The following table illustrates that level keys are unique for each level.

| Hierarchy Level | Level Key | Example Member Value |
|-----------------|-----------|----------------------|
| Year | YearCode | 2013 |
| Month | MonthCode | 201301 |
| Day | DayCode | 20130104 |

Because the level key uniquely identifies members at each level, you map only the level key for the Month level.

Consider the same hierarchy, but with non-unique level keys.

| Hierarchy Level | Level Key | Example Member Value |
|-----------------|--------------|----------------------|
| Year | Year | 2013 |
| Month | YearMonth | January |
| Day | YearMonthDay | Friday |

For each level, the unique level key is made up of the level key and parent level key. In this example, you must map the level keys for Year and Month.

You can define security for members at one or more levels of a hierarchy by using a single lookup table with null values. The lookup table must contain columns that correspond to the level keys for the levels that you want to secure.

For example, suppose a lookup table contains the columns Year, Quarter, and Month. The level keys are Year, YearQuarter, and YearQuarterMonth. If you reference the lookup table in a security filter, it can be used to identify members at any one of those levels. The following rows identify members from different levels:

- 2013, Null, Null identifies a year member.
- 2013, Q1, Null identifies a quarter member.
- 2013, Q1, Jan identifies a month member.

Each row in a lookup table corresponds to a one member at a single level. It should contain correct member key values in the columns corresponding to required level, and Null values in all other key columns. Incorrectly encoded keys are ignored

Tip: The All member in a hierarchy does not have an associated level key value. To include the All member item, you must use Null values in all the lookup table key columns.

Before you can create the security filter, you must complete the following tasks:

- Import the metadata for the lookup table from the data source.
For more information, see [“Importing metadata from a Content Manager data source” on page 39.](#)
- Model the lookup table by creating a query subject and adding query items to it.


Each query item maps to a column in the lookup table.

For more information, see [“Modeling a lookup table” on page 112.](#)

Modeling a lookup table

In IBM Cognos Cube Designer, you model a lookup table by creating a query subject at the project level.

Procedure


1. Select **Model** from the Project Explorer tree, and then click **New Query Subject** .
2. Right-click the query subject and select **Open Editor**.
3. Drag the required lookup table, or specific columns in the lookup table, from the **Data Source Explorer** onto the **Editor** pane.

A query item is created for each column in the lookup table.

Defining a security filter based on a lookup table

After you have finished modeling a lookup table, you can define a security filter that is based on it.

Procedure

1. Select the hierarchy for which you want to define a security filter from the **Project Explorer** tree.
2. From the **Security** tab, click **Add Lookup Table Based Security Filter** .
3. Select the security filter, and then select the required option from the **Scope** list.

For information on scope options, see [“Security filters for hierarchy members” on page 107.](#)

4. Select the query subject that you defined for the lookup table from the **Query Subject** list.
5. Define the hierarchy level on which to filter by mapping the level keys to one or more query items in the **Level Key Filters** list.

6. Click **Edit** to define an expression to filter data in the lookup table.

For example, you can define an expression that restricts a report user to their own data only.

7. Define the filter expression. You can use the following methods to create the expression:

- Select query items to include in the filter by dragging and dropping them from the query subject in the **Project Explorer** tree.
- Type the expression manually, by using functions available from the **Functions** tab in the **Project Explorer** tree as required.

For more information on using an expression in a security filter, see [“Security filters for hierarchy members” on page 107](#).

Tip: Query item references cannot be typed; they must be dragged and dropped.

8. Click **Validate** to check the expression is valid.

9. Click **OK**.


What to do next

To apply a security filter to a dynamic cube, you must now add the filter to a security view.

Defining a role-based security filter

You can manually define security rules for users where no lookup tables exist.

Procedure

1. Select the hierarchy for which you want to define a security filter from the **Project Explorer** tree.
2. Select the **Security** tab.
3. Click **Add Role Based Security Filter** .
4. Select the security filter, and then select the required option from the **Scope** list.
5. Click **Edit** to define an expression to add members to the security filter.

For example, you can define an expression that restricts a report user to their own data only.

6. Define the filter expression. You can use the following methods to create the expression:

- Select members to include in the filter by dragging and dropping them from the **Members** folder in the **Project Explorer** tree.
- Type the expression manually, by using functions available from the **Functions** tab in the **Project Explorer** tree as required.

7. Click **Validate** to check the expression is valid.

8. Click **OK**.

What to do next

To apply a security filter to a dynamic cube, you must now add the filter to a security view.

Security views

You apply security to a dynamic cube by defining a security view.

You can apply the following types of security to a view:

- hierarchy member security

To apply hierarchy member security, you add one or more security filters to a security view.

One view that contains a set of filters and a group of views that collectively contain the same set of filters should have the same view of a cube. The only difference is if tuples are not possible in an underlying view.

- measure, dimension, and attribute security

To apply measure, dimension, and attribute security, you grant or deny access to the required objects in a dynamic cube.

There are several points to consider when you are setting up a security view:

- A security view that contains any explicit grant, including the built-in Grant All Members filter, takes precedence over a view that has no grant filters. A security rule might not have any grant filters if one of the following scenarios exists: if there are deny filters only for the hierarchy or if there are no filters that are defined for the hierarchy.
- If a security view contains a security filter that explicitly denies access to a hierarchy member, it is not possible for another security filter (in the same view or a separate view) to grant access to the same member.
- When you add multiple security filters to a security view, each filter is processed independently. If a security view does not include any security filters, users have access to all hierarchy members.

If a security view contains multiple security filters, the resulting list of granted members is derived from merging all granted members minus all denied members.

If there are no explicitly granted members, "all granted members" is replaced by all members in the hierarchy.

Report users are granted access to an individual member only if that member is granted access in all individual security filters.

- When you merge security views by using IBM Cognos Administration, the resulting list of granted members is derived from merging all granted members minus all denied members.

If there are no explicitly granted members, "all granted members" is replaced by all members in the hierarchy.

Report users are granted access to an individual member only if that member is granted access in all individual security views.

- When a security view includes security filters that contain both grant and deny expressions, the resulting list of granted members is derived from merging all granted members minus all denied members.
- If report users are not assigned to any security view where security is defined, they are denied access to all hierarchy members.

Tuple security

IBM Cognos Dynamic Cubes dimensional security supports defining only which member users have access. There is no support for defining security on specific tuples or cells. However, if a user is in multiple views, it is possible that the combination of views will expose tuples that were not visible in any of the underlying views. If the tuple value is not visible in at least one of the underlying views, the tuple value will be ERR in the final view.

For a tuple value to be visible, the tuple must be visible in at least one of the underlying views.

Security view 1 contains granted United States, Outdoor Protection, and their descendants.

The table shows the tuple value.

| <i>Table 39. Example of a tuple value in a security view</i> | | |
|--|---------------|---------------------------|
| Quantity | | Outdoor Protection |
| Americas | United States | 2,033,754 |

Security view 2 contains granted Brazil, Camping Equipment, and their descendants.

The table shows the tuple value.

| Table 40. Example of a tuple value in a security view | | |
|---|--------|-------------------|
| Quantity | | Camping Equipment |
| Americas | Brazil | 752,338 |

Because the tuples (Brazil, Outdoor Protection) and (United States, Camping Equipment) are not visible in either of the underlying views, the tuples are indicated as errors in the final views.


The table shows the tuple value for the combined security views 1 and 2.

| Table 41. Example of a tuple value for a combined security view | | | |
|---|---------------|-------------------|--------------------|
| Quantity | | Camping Equipment | Outdoor Protection |
| Americas | United States | --- | 2,033,754 |
| | Brazil | 752, 338 | --- |

Defining a security view

You use IBM Cognos Cube Designer to define a security view for a dynamic cube.

Procedure

1. From the **Project Explorer** tree, right-click the required dynamic cube and select **Open Editor**.
2. Select the **Security** tab.
3. Click **Add Security View** .


What to do next

You can now add the required security filters and define which measures, dimensions, and attributes to secure.

Adding a security filter to a security view

You secure hierarchy members in a dynamic cube by adding the required security filters to a security view.

Procedure


1. From the **Project Explorer** tree, right-click the required dynamic cube and select **Open Editor**.
2. Select the **Security** tab.
3. Select the security view to which you want to add a security filter.
4. Select the **Members** tab.
5. Click **Add Secured Member** .
6. Select the security filters for each hierarchy that you want to secure, then click **OK**.

Defining secured measures

You secure measures in a dynamic cube by granting or denying access in a security view.

Procedure


1. From the **Project Explorer** tree, right-click the required dynamic cube and select **Open Editor**.
2. Select the **Security** tab.
3. Select the security view to which you want to add secured measures.
4. Select the **Measures** tab.

5. Click **Add Secured Measures** .
6. Select the measures for which you want to grant or deny access, and then click **OK**.
7. Select **Grant** or **Deny** as required for each measure that is listed in the **Measures** tab.

Defining secured dimensions

You secure dimensions in a dynamic cube by granting or denying access in a security view.


Procedure

1. From the **Project Explorer** tree, right-click the required dynamic cube and select **Open Editor**.
2. Select the **Security** tab.
3. Select the security view to which you want to add secured dimensions.
4. Select the **Dimensions** tab.
5. Click **Add Secured Dimensions** .
6. Select the dimensions for which you want to grant or deny access, and then click **OK**.
7. Select **Grant** or **Deny** as required for each dimension that is listed in the **Dimensions** tab.

Defining secured attributes

You secure attributes in a dynamic cube by granting or denying access in a security view.

Procedure

1. From the **Project Explorer** tree, right-click the required dynamic cube and select **Open Editor**.
2. Select the **Security** tab.
3. Select the security view to which you want to add secured attributes.
4. Select the **Dimensions** tab.
5. Click **Add Secured Attributes** .
6. Select the attributes for which you want to grant or deny access, and then click **OK**.
7. Select **Grant** or **Deny** as required for each attribute that is listed in the **Dimensions** tab.

Chapter 12. Cognos dynamic cubes administration

Dynamic cubes are published as OLAP data sources to IBM Cognos Content Manager. Administrators perform a number of tasks before dynamic cubes can be used by the IBM Cognos studios to create reports and analyses, and can perform additional tasks to manage or optimize the performance of dynamic cubes.

After the dynamic cubes are published as data sources, they can be accessed and configured in IBM Cognos Administration on the **Status** tab, in the **Dynamic Cubes** page. They can also be accessed from different areas in Cognos Administration; however, the **Dynamic Cubes** page is the central location where you can administer all instances of dynamic cube data sources in the IBM Cognos Analytics environment.

If you need information on publishing dynamic cubes, see [“Deploying and publishing dynamic cubes” on page 69](#)

Administration tasks

Before you can work with published dynamic cube data sources, you must perform the following tasks:

- Assign an account in IBM Cognos to access the relational database that contains the data for the dynamic cubes.
- If you are using multiple dispatchers, define routing rules to ensure that reports are directed to the dynamic query server.
- Specify access permissions and capabilities required for modeling, configuring, managing, and optimizing dynamic cubes.
- Add dynamic cubes to the query service.
- Start dynamic cubes in the query service.

You can perform the following tasks to manage dynamic cubes or to optimize the performance of dynamic cubes:

- Assign users and groups to security views.
- Manage dynamic cubes. For example, you might refresh caches or security settings.
- Edit the query service configuration parameters for dynamic cubes. For example, you might need to edit the JVM (Java Virtual Machine) heap size.
- Edit dynamic cube properties. For example, you might change the default value for the data cache size limit.
- Create and schedule query service tasks.

After the dynamic cubes are used in reports and log files are analyzed, you can perform the following tasks:

- Use Aggregate Advisor to view aggregate recommendations.
- Monitor the metrics of the dynamic cubes added to the query service. For information about system performance metrics, see the *IBM Cognos Analytics Administration and Security Guide*.

Access permissions and capabilities for dynamic cubes

Use the IBM Cognos groups, roles, and capabilities to define access permissions that are required for modeling, configuring, managing, and optimizing dynamic cubes.

User permissions and capabilities can differ from one environment to another. For example, in a development environment, a user might be granted the capabilities to assign cubes to a dispatcher and to start the cubes. In a production environment, the same user might not be granted access to publish a cube to the content store.

The user roles from previous versions of IBM Cognos Dynamic Cubes were renamed so that they are more consistent with the predefined role names in the **Cognos** namespace in IBM Cognos Analytics. The following table shows the mapping between the new and old role names.

| New role name | Old role name |
|--|---------------------------|
| Dynamic Cubes Modelers | Model cubes |
| Dynamic Cubes Security Administrators | Secure cubes |
| Dynamic Cubes Configuration Administrators | Configure cubes |
| Dynamic Cubes Managers | Manage cubes |
| Dynamic Cubes Optimizers | Optimize cubes |
| Dynamic Cubes Administrators | <i>No equivalent role</i> |
| <i>No equivalent role</i> | Prime cubes |

The following table describes the user roles that are associated with managing dynamic cubes and the typical tasks that these roles perform. Administrators must ensure that these roles are created in the **Cognos** namespace in IBM Cognos Administration.

| Role | Tasks |
|--|---|
| Dynamic Cubes Modelers | Model and publish cubes, assign cubes to dispatchers, and start cubes. If needed, this role can be further broken down to limit the capabilities of individual users (as outlined in Table 44 on page 119). |
| Dynamic Cubes Security Administrators | Assign users, groups, or roles to dynamic cubes security views. |
| Dynamic Cubes Configuration Administrators | Assign cubes to server groups and dispatchers, and configure the query service and individual cubes. |
| Dynamic Cubes Managers | Perform interactive administrative tasks on cubes, and create and schedule query service administrative tasks. |
| Dynamic Cubes Optimizers | Save in-memory aggregate recommendations from Aggregate Advisor to the content store. To perform other Aggregate Advisor tasks, an administrator only needs access to IBM Cognos Dynamic Query Analyzer and an account in IBM Cognos Analytics. |
| Dynamic Cubes Administrators | Perform all of the operations described in this table. This role is assigned to all the roles described in Table 44 on page 119 or Table 45 on page 120 . |

Each role requires an associated IBM Cognos Analytics capability to perform specific tasks on dynamic cubes. To grant access to a capability, you must grant the correct permissions for it to the appropriate

roles. For example, dynamic cube modelers who create models need execute and traverse permissions for the **Import relational metadata** capability.

The following table lists the roles and the capabilities that these roles require for managing dynamic cubes.

| <i>Table 44. Roles and their capabilities</i> | | |
|--|---|------------------------------------|
| Role | Capability | Required access permissions |
| Dynamic Cube Modelers (creating new models) | Import relational metadata | Execute, Traverse |
| Dynamic Cube Modelers (starting cubes) | Administration Administration > Configure and manage the system | Execute, Traverse |
| Dynamic Cube Modelers (generating cubes or dimensions with data samples) | Specification Execution | Execute, Traverse |
| Dynamic Cubes Security Administrators | Administration Administration > Data Source Connections | Execute, Traverse |
| Dynamic Cubes Configuration Administrators | Administration Administration > Administration tasks Administration > Configure and manage the system Administration > Data Source Connections | Execute, Traverse |
| Dynamic Cubes Managers | Administration Administration > Administration tasks Administration > Configure and manage the system Administration > Query Service Administration Administration > Run activities and schedules Scheduling Cognos Viewer Cognos Viewer > Run With Options | Execute, Traverse |
| Dynamic Cubes Optimizers (saving in-memory recommendations) | Administration > Configure and manage the system | Execute, Traverse |

Tip: Capabilities are also referred to as secured functions and features. This distinction is helpful when dealing with two-level capabilities, such as the **Administration** capabilities. In this case, capabilities such as **Configure and manage the system**, **Data Source Connections**, or **Query Service Administration** are secured features of the **Administration** capability, which itself is a secured function. For more information about the IBM Cognos Analytics capabilities, see the security sections in the *IBM Cognos Analytics Administration and Security Guide*.

In addition to capabilities, dynamic cube administrators need the correct combination of access permissions for the content store objects. The following table specifies the objects and the permissions required for specific roles.

| <i>Table 45. Content store objects permissions for roles</i> | | |
|--|--|--|
| Role | Content store object | Required access permissions |
| Dynamic Cubes Modelers (publishing cube to a server) | Configuration, Data Source Connections, Directory, Cognos | Read, Write, Execute, Traverse |
| Dynamic Cubes Modelers (publishing a package) | My Folders, Public Folders | Read, Write, Traverse |
| Dynamic Cubes Modelers (assigning cube to a dispatcher) | Query Service (on one or more dispatchers), Configuration, Dispatchers and Services | Read, Write, Execute, Traverse |
| Dynamic Cubes Security Administrators | Configuration, Data Source Connections, Directory, Cognos | Read, Write, Execute, Traverse, Set policy |
| Dynamic Cubes Configuration Administrators | Configuration, Query Service (on all dispatchers on which cubes are managed), Dispatchers and Services | Read, Write, Execute, Traverse |
| Dynamic Cubes Managers | Query Service (on all dispatchers on which cubes are managed) Configuration, Content Administration | Read, Write, Execute, Traverse |
| Dynamic Cubes Optimizers (saving in-memory recommendations) | Configuration, Data Source Connections, Directory, Cognos | Read, Write, Execute, Traverse |

Securing cube data

Each source dynamic cube is assigned a single data access account. The dynamic query mode server that hosts dynamic cubes is a trusted process that uses the connection and signon for the specified account to access the underlying relational data source for the dynamic cube when the account user creates trusted credentials.

A Cognos system administrator has access to all data within a dynamic cube. However, a dynamic cube does not necessarily expose all the data accessible through the relational data source connection. In such cases, it may be required to ensure that the system administrator does not have the ability to access the relational data source using the data access account assigned to a dynamic cube.

The configuration of the relational data source is no different than the configuration of any other relational data source in IBM Cognos Analytics. If an explicit signon that consists of a user ID and password is used to access the relational data source, the system administrators can grant themselves access to this signon and use it to connect to the relational data source.

If users are granted the capability to manage their own data source signons, these users can create and save a signon for a particular data source. This self-managed signon can be used to secure a dynamic cube, assuming that the users have also created trusted credentials. The system administrators can assign the data access account of a specific user to secure the dynamic cube, but they cannot use the signon to access the relational data source.

If an external namespace is used for authentication to an external data source, there is no signon that the system administrator can use to access the relational data source. In this case, the trusted dynamic query mode server impersonates the user of the data access account to sign on to the relational database.

Creating a Dynamic Cubes Developer role

Developers of applications that are based on dynamic cubes need a specific set of access permissions to be able to model, deploy, and manage a dynamic cube without asking for assistance from the Cognos system administrator each time they want to perform a specific administration task on the cube, for example, when they want to deploy or restart the cube.

To address this need, it might be useful to create the role of a Dynamic Cube Developer, in addition to the standard dynamic cube roles that are documented in the section [“Access permissions and capabilities for dynamic cubes”](#) on page 117.

As a system administrator who creates the Dynamic Cubes Developer role, you must carefully consider which types of access permissions and capabilities you must grant to this role to enable the developers to do their job without compromising the security of the system.

The following table specifies the tasks that a member of the Dynamic Cubes Developer role performs, and the restrictions that the system administrator would likely impose when granting access permissions for the developers in the context of these tasks.

| Task | Restrictions on the associated access permissions |
|--|---|
| Import relational metadata into Cognos Cube Designer | Grant access only to a specific set of relational data sources that the developers need to import metadata. |
| Publish a cube to the content store | Grant permissions to either create new dynamic cube data sources or update the existing dynamic cube data sources. |
| Create a package in the content store | Grant access only to specific folders where the developers can create packages. |
| Assign a user account to the data access account of a cube | Grant access only to those accounts that can be assigned to access the relational data source. Developers should not have the permission to edit data source connections or signons. |
| Assign a cube to a dispatcher and modify the cube configuration. | Restrict the dispatchers to which the developers can assign a cube and modify the cube configuration. |
| Perform administration tasks on a cube | Restrict the dispatchers on which the developers can manage a cube. Do not allow the developers to perform any other administration tasks, such as stopping or starting the query service. |
| Create and execute administration tasks for dynamic cubes | The ability to create and edit administrative tasks cannot be restricted to dynamic cubes only. Allowing a user to create administration tasks for dynamic cubes allows them to create and execute these types of tasks for the whole system. |

Procedure

The following steps are performed by the Cognos system administrator.

1. Create the Dynamic Cubes Developer role in the **Cognos** namespace in IBM Cognos Administration.

2. Specify access permissions for the Dynamic Cubes Developer role.

The following list specifies which access permissions are required for each task that the members of the Dynamic Cubes Developer role perform.

Tip: If a user owns a specific cube, denying access permissions on this cube has no effect for that user.

Import relational metadata into Cube Designer

Deny all access permissions for relational data sources that cannot be imported. This does not allow to use the data source in Framework Manager as well.

Grant read and execute permissions, and deny write permissions for the relational data sources that can be imported.

Publish a cube to the content store

Grant read, write, execute, and traverse permissions for existing dynamic cube data sources that the user can update.

Grant read, execute, and traverse permissions, and deny write permissions for existing dynamic cube data sources that the user cannot update.

Create a package in the content store

Deny all permissions for folders to which the users are denied access. The users cannot see these folders in Cognos Administration, but they might see them in Cognos Cube Designer. The users cannot publish packages to these folders.

Grant read, execute, and traverse permissions, and deny write permissions for folders that the users can view but cannot update.

Grant read, write, execute, and traverse permissions for folders that the users can update.

Grant read, execute, and traverse permissions, and deny write permissions for packages that the users can view but cannot update.

Grant read, write, execute, and traverse permissions for packages that the users can update.

Grant read, write, execute, and traverse permissions for the dynamic cube data sources.

Assign a user account to the data access account of a cube

Deny all permissions for the connection and signon objects of the relational data source that the dynamic cube is based on.

Provide a limited number of signons for the relational data source connection to control the data access because the developer can assign any user account to a dynamic cube. Grant read, execute, and traverse permissions on selected connection and signon objects.

Assign a cube to a dispatcher and edit the cube configuration

Grant read, write, execute, and traverse permissions for the query service on the dispatchers that the users can access.

Grant execute and traverse permissions, and deny read and write permissions for the query services on the dispatchers that the users cannot access.

Deny all permissions for the dynamic cube data sources that might not be configured.

Perform administrative tasks on a cube

Grant read, write, execute, and traverse permissions for query service on the dispatchers that the users can access.

Create and edit only dynamic cubes administrative tasks

If a developer who can manage dynamic cubes has no need to create administration tasks for dynamic cubes, you might choose not to assign the following capabilities to the developers:

- **Administration > Run activities and schedules**
- **Scheduling**
- **Cognos Viewer**
- **Cognos Viewer > Run With Options**

3. Add users to the Dynamic Cubes Developer role.

4. Add this role to one or more of the dynamic cube administrative roles that are documented in the section [“Access permissions and capabilities for dynamic cubes”](#) on page 117.

Assigning data access accounts for dynamic cubes

Assign a single IBM Cognos account as a data access account for each dynamic cube. The account that you assign must have access to the relational database that the source dynamic cube is based on.

The query service that hosts dynamic cubes uses the account database connection and signon to access the relational database that the dynamic cube is based on when trusted credentials are created for this account. The account is used to log on to IBM Cognos Analytics, to load data and metadata from the relational database, and to execute startup triggers in virtual cubes.

Before you begin

Before you assign the data access accounts for your dynamic cubes, perform the following tasks:

- Create trusted credentials for the user who will access the relational database that contains the source dynamic cube.

For more information, see [“Creating trusted credentials”](#) on page 124.

- Create a data source signon for the user who will access the relational database that contains the source dynamic cube.

The user ID and password that make up the signon must already be defined in the relational database.

You can use multiple data source connections or multiple data source signons for dynamic cube data sources. In this situation, however, one of the connections and one of the signons must be defined using the name **DynamicCubes**.

For more information, see [“Creating a signon”](#) on page 124.

For more information about creating data source connections and data source signons, see the *IBM Cognos Analytics Administration and Security Guide*.

About this task

Virtual cubes do not require an access account because they obtain data from other source or virtual cubes. However, if a virtual cube has a startup trigger, it needs an access account. In this situation, the virtual cube uses the access account of the first source cube in the cube definition.

If a virtual cube is built by using two virtual cubes, it uses the access account that belongs to the first source cube of the first virtual cube.

Procedure


1. In **IBM Cognos Administration**, on the **Status** tab, click **Dynamic Cubes**.
In the **Scorecard** section, you can see all published dynamic cube data sources.
2. For the dynamic cube for which you want to specify the access account, click the **Actions** drop-down menu, and then click **Set properties**.
3. On the **General** tab of the properties page, in the **Access Account** section, click **Select the access account**.
4. Browse the directory and select the user who will own the access account.
5. Click **OK**. The user name appears in the **Access Account** section.

Creating trusted credentials

You can create trusted credentials when you want to authorize other users to use your credentials because those users do not have sufficient access permissions to perform specific tasks.

For users to use trusted credentials, traverse permissions must be granted for the namespace.

Procedure

1. Click the my area options button , **My Preferences**.
2. On the **Personal** tab, under **Credentials**, if you have not created credentials before, click **Create the Credentials**.

Tip: If your trusted credentials are already created, you might only need to renew them by clicking **Renew the credentials**.

3. Select the users, groups, or roles that you want to authorize to use your credentials.

If you are prompted for your credentials, provide your user ID and password.

4. If you want to add entries, click **Add** and choose how to select entries:

- To choose from listed entries, click the appropriate namespace, and then select the check boxes next to the users, groups, or roles.
- To search for entries, click **Search** and in the **Search string** box, type the phrase you want to search for. For search options, click **Edit**. Find and click the entry you want.
- To type the name of entries you want to add, click **Type** and type the names of groups, roles, or users using the following format, where a semicolon (;) separates each entry:

namespace/group_name;namespace/role_name;namespace/user_name;

Here is an example:

Cognos/Authors;LDAP/scarter;

5. If you want to remove an entry from the list, select the check box next to it and click **Remove**.

Results

The users, groups, or roles that can use your credentials are now listed in the **Credentials** section.


Creating a signon

The data source connection signon must be defined so that the query service can automatically access the data required for loading dynamic cubes.

About this task

A data source connection must have at least one signon that the query service can use to connect to the data source. If the data source connection has two or more signons, one of the signons must be named `Dynamic Cubes`. This signon will be used by the query service to connect to the data source.

Procedure

1. In **IBM Cognos Administration**, on the **Configuration** tab, click **Data Source Connections**.
2. Click the data source, and then click the connection to which you want to add a new signon.
3. Click the new signon button .
4. In the name and description page, type a unique name for the data source signon and, if you want, a description and screen tip, and then click **Next**.
5. Type the **User ID** and **Password** to connect to the database, and click **Next**.

The **Select the users** page appears.

6. To add users and groups that can use the signon, and click **Add**.

- To choose from listed entries, click the appropriate namespace, and then select the check boxes next to the users, groups, or roles.
- To search for entries, click **Search** and in the **Search string** box, type the phrase you want to search for. For search options, click **Edit**. Find and click the entry you want.
- To type the name of entries you want to add, click **Type** and type the names of groups, roles, or users using the following format, where a semicolon (;) separates each entry:

```
namespace/group_name;namespace/role_name;namespace/user_name;
```

Here is an example:

```
Cognos/Authors;LDAP/scarter;
```

7. Click the right-arrow button and when the entries you want appear in the **Selected entries** box, click **OK**.

Tip: To remove entries from the **Selected entries** list, select them and click **Remove**. To select all entries in a list, in the title bar for the **Name** list, select the check box. To make the user entries visible, click **Show users in the list**.

8. Click **Finish**.

The new data source signon appears under the connection.

Configure dynamic cubes for the query service

The query service manages dynamic query requests and returns the results to the batch or report service that sent the request. You can configure one or more instances of the query service to run an instance of a dynamic cube.

You can perform most configuration and management actions for dynamic cubes from the **Status** tab, **Dynamic Cubes** page. On the **Dynamic Cubes** page, in the Scorecard section there are different views available: **Dynamic Cubes - (All)**, **Dynamic Cubes - Base Cubes**, **Dynamic Cubes - Virtual Cubes**, and **All server groups**. To change the view, click the drop-down menu of the current view.

In the **Dynamic Cubes - (All)** view, you see a list of all dynamic cube data sources in the IBM Cognos Analytics environment, and in the Scorecard section you can see status information about cubes.

Cubes that are published to IBM Cognos Content Manager, but not configured show the status **Unknown**.

Cubes that are configured appear hyperlinked and show the status **Unavailable**. Note that cubes can appear hyperlinked, but Unknown for up to 30 seconds while the configuration process is being completed.

Cubes that are started show the status **Available**.

In the event that the query service is down, or communication between the dispatcher and the query service is unavailable, then the query service displays the status Unavailable and all cubes display the status Unknown.

Use the drop-down action menus for each data source to perform different actions on cubes. The actions that are available, depend on the status of cubes. The status and the action menus can be stale so you might need to click the **Refresh** icon to update the view.

You can drill down on each configured data source to the server groups for the cube and you can drill down again to the dispatchers. When you drill down to the dispatcher level, the **Metrics** section is populated with metrics for individual dynamic cubes. You can hover your cursor over each of the metrics to view descriptions of the metrics.

In the **All server groups** view you see a list of the query service groups to which cubes have been assigned. You can drill down on server groups to the dispatchers, and you can drill down again on the

dispatchers for a list of all data sources served by a dispatcher. Use the drop-down action menus at each level to perform actions on cubes.

On occasion, when an action, such as changing query service properties for dynamic cubes, requires you to start or restart the query service, you must access the query service through the **System** page on the **Status** tab. The start and stop actions in the **Dynamic Cubes** page are only used to perform actions on cubes.

Using multiple dispatchers for the query service

If you plan to use multiple dispatchers for the query service, you must define routing rules to ensure that reports are directed to the dynamic query server for execution. To ensure that your server processes dynamic cube requests, you need to complete the following tasks:

- Assign a server group to the dispatcher.

Tip: To specify a server group name, on the **Status** tab in IBM Cognos Administration, click **System**. In the Scorecard section, choose the **All dispatchers** view. For each dispatcher, on its set properties page, click the **Settings** tab and choose **Tuning** under **Category**. For the Server group property, type a name of your choice in the **Value** box.

- Assign a routing set to all packages that are associated with a dynamic cube.
- Create a routing rule to send queries for the routing set to the server group.

Set routing rules in either IBM Cognos Administration or the IBM Cognos Software Development Kit. For more information, see the *IBM Cognos Analytics Administration and Security Guide* or the *IBM Cognos Software Development Kit Developer Guide*.

Adding dynamic cubes to the query service

Before you can start dynamic cubes, you must add them to the query service. You can add dynamic cubes to the query service individually or in groups.

Before you begin

You can add dynamic cubes to the query service by selecting the default server group.

If you are allocating dispatchers to dynamic cubes and are routing reports to select dispatchers within your Cognos Analytics environment, you need to create named server groups. For information on assigning dispatchers to server groups see [“Configure dynamic cubes for the query service” on page 125](#)

If a set of virtual cubes and source cubes are part of the same hierarchy, you must add all cubes in the set to the same query service. For more information about hierarchies, see [“Hierarchies” on page 15](#).


Procedure

1. In **IBM Cognos Administration**, on the **Status** tab, click **Dynamic Cubes**.
2. In the **Scorecard** section, select the **Dynamic Cubes - (All)** view.
Tip: To change the view, click the drop-down menu for the current view.
3. Decide whether you want to add one or more dynamic cubes to a server group.
 - To add one dynamic cube, click its **Actions** drop-down menu and click **Add data store to server group**.
 - To add multiple dynamic cubes, select the check boxes for the applicable dynamic cubes. From the **Group actions** drop-down menu, click **Add data store to server group**.
4. In the window that is displayed, select the available server group or **All**.

Tip: If the dynamic cubes that you are configuring are associated with dispatchers that share the same server group, add the cubes to this server group now. This helps to avoid problems with load balancing when you run reports based on these cubes.

5. View the results of your action in the response window.

In the Scorecard section, the dynamic cube now appears hyperlinked.

6. In the Scorecard section, click the **Refresh** icon  occasionally until the status of the cube changes to **Unavailable**. Configuration can take 30 seconds.

When the cube is configured and has the status **Unavailable**, the drop-down action menu of the cube displays the **Start** action.

Tip: The cube status and its action menu can be stale. To update the view, click the **Refresh** icon.

Results

When a dynamic cube is added to the query service, it is assigned the default configuration settings. You can change the default dynamic cube properties and query service properties. For more information, see [“Setting dynamic cube properties” on page 132](#) and [“Setting query service properties for dynamic cubes” on page 129](#).

After dynamic cubes are added to the query service, they must be started before they can be used by the IBM Cognos studios. For more information about starting cubes, see [“Starting and managing dynamic cubes” on page 127](#).

If you need to remove dynamic cubes from the query service, use the **Remove data store from server group** action. The specified dynamic cube data sources will no longer be hyperlinked and the status will change to **Unknown**.

Starting and managing dynamic cubes

The query service runs and creates an instance of a dynamic cube which is based on the model stored in Content Manager. Administrators can start, stop, refresh, and perform other actions to manage instances of dynamic cubes.

Before you begin

Because virtual cubes are composed of source cubes there are several things to consider before you start, stop, and refresh the cubes:

- Virtual cubes and their source cubes must be available on the same dispatcher.
- Source cubes that are a part of a virtual cube must be started first.
- If source cubes are part of a virtual cube, the virtual cube must be stopped before the source cubes are stopped.
- When you refresh the data and member cache of a source cube, the data and member cache of any associated virtual cubes is also refreshed.
- You cannot start a virtual cube if its source cube is paused. You also cannot pause a source cube or virtual cube when a dependent virtual cube is running.
- You can only perform the following actions on virtual cubes: **Start**, **Stop after active tasks complete**, and **View recent messages**.

About this task

You can perform most actions on individual or on multiple dynamic cubes. The actions that are available depend on the status of the cubes. Adding and removing cubes from server groups is described in the topic [“Adding dynamic cubes to the query service” on page 126](#). The following list describes other actions that are associated with managing dynamic cubes in the query service.

Start

This action starts dynamic cubes in the query service. You must start dynamic cubes in the query service to use them in IBM Cognos studios. When you start a cube, the hierarchy members are loaded in the cache.

Cubes started in the query service display the **Available** status in the **Scorecard** section of the **Dynamic Cubes** view. In some cases, when a cube is starting it will display the status **Partially Available**. The parent dynamic cube status reflects the consolidated status of child cubes.

Set Properties

This action lets you set a number of general properties for dynamic cubes including hiding entries, and selecting the access account for the entry. For more information see [“Setting general properties for a dynamic cube”](#) on page 137.

Stop after active tasks complete

This action stops cubes after the queries that are currently running are complete. Typically, you stop a cube if it does not need to be online and accessible.

Stop immediately

Stopping immediately stops the cube and cancels any queries that are currently running. This action is useful if you want to restart cubes to apply changes made to the model without waiting for long-running queries to complete.

Restart

This action stops and then starts a cube. For example, you might restart to reset a cube after a failure, or after a successful ETL (Extract, Transform, Load) run. Restarting a cube is a different action than restarting the query service. When following procedures, note whether the cube or the query service must be restarted.

Pause

This action cancels all existing queries and does not allow any new queries against the cube. The query service waits for all queries to clear before changing the cube state. If the request times out, an error occurs and the state reverts to **Running**. When paused, the dynamic cube continues to run so that the data caches remain valid. While the cube is paused, its status is shown as **Partially available**.

You can pause a dynamic cube to maintain aggregate tables for near real time updates, or to make database configuration changes, such as recycling a database or increasing buffer pools, while a dynamic cube is active. For more information, see [“Pausing a dynamic cube to update aggregate tables”](#) on page 151. You must pause each dynamic cube separately.

Use the **Start** command to resume the cube to **Running** state and allow new queries against the cube.

Refresh member cache

If the dimension tables have been updated while the cube is running, you can refresh the member cache which will allow the cube to remain accessible to users while the dimension tables are reloaded from the back-end data source.

An update of the member cache, builds a new set of members in the background. This new set becomes available when the refresh is complete. This refresh requires additional memory to store two copies of the member cache in memory while the new cache is built.

Once the new member cache is available, the data cache is refreshed. This is because the data in the cache is tied to the structure of the members in the member cache.

For more information see [“Types of caches used by dynamic cubes”](#) on page 134.

Refresh data cache

Refreshing the data cache picks up changes to the fact table and re-syncs the data caches with the fact table. Data caches are refreshed dynamically while queries are still running so cubes remain accessible to users. When the member cache is brought online, a new corresponding data cache is also created. Even though a new data cache starts as empty, some additional space is required while the new cache is introduced and while queries are using the previous version of the data cache.

For more information see [“Types of caches used by dynamic cubes”](#) on page 134.

Refresh security settings

While the cube is still running, this action reloads the access permissions on the security views, and clears cached information that was loaded from the security lookup table.

This action also attempts to reload security rules from the model of a published cube. Rule reload succeeds only if the rest of the model was not significantly changed; for example, no levels, hierarchies or dimensions were added, changed, or removed. If these types of changes were done

in the model, the rule reload is not executed and a corresponding message is written to the recent message log for the cube.

Edit security view permissions

Administrators can access the security view models for cubes, override the default group permissions, and add the appropriate users and groups to the model views. For more information see [“Setting access permissions for security views”](#) on page 139.

Clear workload log

This action removes all log entries for a dynamic cube. This is useful if you want to capture new information about report usage. For more information, see [“Workload log for Aggregate Advisor ”](#) on page 134.

Incrementally update data

This action invokes an incremental load that updates the aggregate cache and data cache to reflect the newly-added fact rows.

For more information, see [“Loading incremental updates to dynamic cubes”](#) on page 148.

Delete

This action deletes a published cube from Content Manager.

View recent messages

This action allows the administrator to view recent log messages to diagnose problems with dynamic cubes. The time zone displayed is the time zone of the administrator that is viewing the log messages.

Procedure

1. In **IBM Cognos Administration**, on the **Status** tab, click **Dynamic Cubes**.
2. In the **Scorecard** section, click the **Dynamic Cubes - (All)** view.
 - To perform an action on one dynamic cube, click the chosen action from the cube **Actions** drop-down menu.
 - To perform an action on a group of dynamic cubes, select the check boxes that are associated with the chosen cubes. Then from the **Group actions** drop-down menu, select the action that you want to perform.
3. View the results of your action in the **View the results** window.

Tip: The cube status and its action menu can be stale. To update the view, click the **Refresh** icon



Results

For information about scheduling query service administration tasks, see [“Creating and scheduling query service administration tasks”](#) on page 138.

Setting query service properties for dynamic cubes

The query service uses a number of environment, logging, and tuning configuration settings.

About this task

When a dynamic cube is added to the query service, the default query service configuration values are assigned to the cube. You can change the values to meet the requirements of your IBM Cognos Analytics system.

Procedure

1. In **IBM Cognos Administration**, on the **Status** tab, select **System**.
2. In the **Scorecard** section, select the **All servers groups** view.

Tip: To select a different view, in the **Scorecard** section, click the drop-down menu for the current view.

3. Click the server group under **System**.
4. From the **Actions** menu for the **QueryService - dispatcher_name**, click **Set properties**
5. Click the **Settings** tab.
6. In the **Value** column, type or select the values for the properties that you want to change. The following list describes the properties that you can set for the query service.

Advanced settings

Click **Edit** to specify advanced configuration settings. Because an entry acquires advanced settings from a parent, editing these settings overrides the acquired advanced settings. For information about types of advanced settings, see the *IBM Cognos Analytics Administration and Security Guide*.

Dynamic cube configurations

Click **Edit** to add dynamic cubes to the query service.

Important: Starting with version 10.2.1 of IBM Cognos Analytics, the preferred way of adding dynamic cubes to the query service is documented in the topic [“Adding dynamic cubes to the query service”](#) on page 126.

Audit logging level for query service

Select the level of logging that you want to use for the query service.

Enable query execution trace

A query execution trace (run tree trace) shows queries that run against a data source. You use the trace to troubleshoot query-related issues.

You can find execution trace logs in the following location: *install_location/logs/XQE/reportName/runtreeLog.xml*

You can view and analyze these log files using IBM Cognos Dynamic Query Analyzer. For more information, see the *IBM Cognos Dynamic Query Analyzer User Guide*.

Enable query planning trace

Query plan tracing (plan tree) captures the transformation process of a query. You can use this information to gain an advanced understanding of the decisions and rules that are executed to produce an execution tree.

The query planning trace is logged for every query that runs using dynamic query mode. You can find planning trace logs in the following location: *install_location/logs/XQE/reportName/plantreeLog.xml*

Since planning logs are large, there is an impact on query performance when this setting is enabled.

Generate comments in native SQL

Specifies which reports are generating the SQL queries in the database.

Write model to file

Specifies whether the query service will write the model to a file when a query runs. The file is used only for troubleshooting purposes. Modify this property only with the guidance of IBM Software Support.

You can find the file in the following location:

install_location\logs\XQE\model\packageName.txt

Idle connection timeout

Specifies the number of seconds to maintain an idle data source connection for reuse.

The default setting is 300. Valid entries are 0 to 65535.

Lower settings reduce the number of connections at the expense of performance. Higher settings might improve performance but raise the number of connections to the data source.

Do not start dynamic cubes when service starts

Prevents the dynamic cubes from starting when the query service starts.

Dynamic cube administration command timeout

Specify the amount of time to wait for a resource to be available for a dynamic cubes administration action. This action is canceled if the time period is exceeded.

Tip: Setting this value to zero causes the command to wait indefinitely.

Minimum query execution time before a result set is considered for caching

Specify the minimum amount of time to wait for a query before caching the results.

This setting only applies to dynamic cubes.

Initial JVM heap size for the query service

Specifies the initial size, in MB, of the Java Virtual Machine (JVM) heap.

JVM heap size limit for the query service

Specifies the maximum size, in MB, of the JVM heap.

Initial JVM nursery size

Specifies the initial size, in MB, that the JVM allocates to new objects. The nursery size is automatically calculated. You do not need to change the setting unless IBM Cognos customer support recommends a change.

JVM nursery size limit

Specifies the maximum size, in MB, that the JVM allocates to new objects. The nursery size is automatically calculated. You do not need to change the setting unless IBM Cognos customer support recommends a change.

JVM garbage collection policy

Specifies the garbage collection policy used by the JVM. You do not need to change the setting unless IBM Cognos customer support recommends a change.

Additional JVM arguments for the query service

Specifies other arguments that control the Java Virtual Machine (JVM). The arguments may vary depending on the JVM.

Number of garbage collection cycles output to the verbose log

Specifies the number of garbage collection cycles to be included in the verbose garbage collection. This controls the maximum size of the log file. Consult with IBM Cognos customer support to increase the setting and collect more logs.

Disable JVM verbose garbage collection logging

Controls JVM verbose garbage collection logging. You do not need to change the setting unless IBM Cognos customer support recommends a specialized change.

7. Start or restart the query service.

For more information, see [“Starting and stopping the query service” on page 131](#).

Results

A summary of the query service properties is displayed in the **Settings - Query Service** pane.

Starting and stopping the query service

When you change the query service configuration settings for dynamic cubes, you need to start or restart the query service for the changes to take effect.

Procedure

1. In IBM Cognos Administration, on the **Status** tab, select **System**.
2. In the **Scorecard** section, click the **All Servers** drop-down menu, point to **Services** and then click **Query**.
3. From the **QueryService** drop-down menu, click the required action.

Setting dynamic cube properties

Dynamic cubes are assigned default property values when they are added to the query service, but these values can be changed.

The default values are often the best choice, with the exception of the data cache size limit.

After setting the properties, before you restart a dynamic cube, you should allow about one minute for updates to refresh the Content Store and the query service. If you start the dynamic cube immediately after saving your changes, the updates may not be available.

You can set the following dynamic cube properties:

Disabled

Disables the cube. This means that the cube is configured for a server, but is not running on that server.

Startup trigger name

Type the name of the trigger event to send after this cube starts.

When a cube is available for query processing, the event is triggered for execution against the server which triggered the event. The purpose of the event is to run reports to populate the cube cache with data.

Post in-memory trigger name

Type the name of the trigger event to send after in-memory aggregates have loaded for a dynamic cube. When in-memory aggregates have finished loading, the event is triggered for execution against the server that triggered the event. The purpose of the event is to run reports to populate the cube cache with data.

Disable result set cache

Disabling the cache is useful during the development or testing phase of a cube because it allows you to test the performance of the data cache. For more information, see [“Types of caches used by dynamic cubes”](#) on page 134.

Data cache size limit (MB)

Type the maximum size of the data cache for the cubes.

The default value is 1024 MB. The result of each query is written to disk. If the maximum size is exceeded, the older report sets are removed from the cache. For more information, see [“Types of caches used by dynamic cubes”](#) on page 134.

Maximum amount of disk space to use for result set cache (MB)

Type the maximum size of disk space.

The result of each query is written to disk. If the maximum amount of disk space is exceeded, the older report sets are removed from the cache. For more information, see [“Types of caches used by dynamic cubes”](#) on page 134.

Enable workload logging

Workload logging is used to capture information about queries that are sent to the dynamic query engine processes. This workload information is used by Aggregate Advisor to determine aggregate recommendations. You do not need to restart the dynamic cube for this property change to take effect. For more information, see [“Workload log for Aggregate Advisor”](#) on page 134.

Maximum space for in-memory aggregates (MB)

Type the maximum size of the memory to use for the in-memory aggregates. In-memory aggregates are loaded when cubes are started and restarted, and when data cache is refreshed. The size of the aggregate cache is used in determining the total JVM heap size of the query service.

In-memory aggregates are loaded on a first come, first served basis. This means that if the aggregate cache is full, no more in-memory aggregates can be loaded. Additionally, an in-memory aggregate may fail to load if the in-memory aggregate cache size limit would be exceeded if it did load.

The default value is 0, which specifies that in-memory aggregates cannot be loaded even if they are defined.

Disable external aggregates

Disabling and enabling external aggregates is useful during the cube and application development phase to measure the impact of external aggregates.

To measure the impact of external aggregates you need to gather output twice. First, you gather output when external aggregates are enabled and then you gather output again when external aggregates are disabled. You use these two sets of output to determine the impact of external aggregates.

Percentage of members in a level referenced in a filter predicate

If no limit is required, type 0.

This value must be between 0 and 100.

This parameter applies to retrieving data associated with a set of members. If there is a greater percentage retrieved than what you specify in this field, then the SQL query that is generated retrieves measure values for all of the members at the level (a speculative pre-fetch of data).

Maximum hierarchies to load in parallel

This parameter specifies the maximum number of hierarchies to load in parallel for cube start and member cache refresh.

The default value is 0, which automatically calculates the number of parallel loads as double the number of CPU cores.

Maximum in-memory aggregates to load in parallel

This parameter specifies the maximum number of in-memory aggregates to load in parallel.

The default value is 0, which automatically calculates the number of parallel loads as double the number of CPU cores.

Measures threshold

This parameter specifies the percentage of measures to retrieve from a dynamic cube. Calculated measures, non-visible measures, and semi-aggregate measures are not included. If the percentage of measures retrieved is greater than specified here, the generated SQL query retrieves all measures.

The default value is 30.

This value must be between 0 and 100. Specify 0 if all measures for a set of levels are required for queries. Specify 100 if only a few measures are required for queries. For example, when using only predefined reports.

Enable automatic optimization of in-memory aggregates

This property enables the automatic optimization of in-memory aggregates that are based on report queries. With this property enabled, the system continually analyzes the workload activity and automatically optimizes the set of in-memory aggregates based on the report queries. If this property is enabled, set the **Maximum space for in-memory aggregates (MB)** property to a value greater than 0.

This property is disabled by default. For more information, see [“Automatic optimization of in-memory aggregates”](#) on page 135.

Procedure

1. In **IBM Cognos Administration**, on the **Status** tab, click **Dynamic Cubes**.
2. In the **Scorecard** section, select the **Dynamic Cubes - (All)** view.
3. Click the dynamic cube that you want to modify, and then click the server group under the cube name.
4. For the **QueryServicedispatcher_name**, click the actions drop-down menu and click **Set properties**.
5. Change the property values as required.
6. Restart the dynamic cube to apply your changes.

Some properties do not require a cube restart. For more information, see descriptions of the properties in this topic.

Types of caches used by dynamic cubes

Several types of caches are available for dynamic cubes to allow improvements to query response times.

Result set cache

Result set cache is an intermediate storage of multidimensional expression language (MDX) query results. This cache is stored on disk in a binary format. The in-memory portion of the result set cache stores the queries and the associated security profile. If an MDX query from the dynamic query mode server to the IBM Cognos Dynamic Cubes engine matches an entry in the result set cache and matches the security profile for the cache, then the result is read from the disk and the query is not run.

Expression cache

The MDX engine caches the results of various intermediate MDX set expressions that are keyed by the expression, its query context, and the security profile of the user. If the MDX engine encounters a set expression that was previously executed then it will retrieve the result set from the expression cache instead of calculating the set expression.

The expression cache helps to relieve the costs associated with the time and memory it takes to run set expressions.

Data cache

The MDX engine sends data queries to the Cognos Dynamic Cubes engine. The result of each query that is retrieved from the database (fact table), database aggregate tables, and in-memory aggregate cache is stored in the data cache.

Before sending any query to the database, the Cognos Dynamic Cubes engine scans the data cache for all the entries that are able to provide some or all of the required data without querying the database.

The data cache is also known as the query cache.

Member cache

This cache contains cube members that are loaded from the source relational data source. The member cache can be refreshed when appropriate, such as when the source data is changed. Refreshing the member cache updates the cube with the latest metadata.

Aggregate cache

Aggregate Advisor analyzes dynamic cubes and suggests aggregates that can improve cube performance. The aggregate cache contains pre-calculated values for aggregations that are suggested by Aggregate Advisor. The pre-calculated values are results of queries to the database.

Aggregate tables

Data can be summarized in a table known as an aggregate table. An aggregate table contains detail fact data that is aggregated at a higher level relative to one or more of the dimensions associated with the data. Using an aggregate table allows the use of pre-calculated data from a data warehouse and decreases the amount of data that is accessed from the data warehouse.

Workload log for Aggregate Advisor

Aggregate Advisor can analyze the underlying model in a dynamic cube data source and recommend which aggregates to create. Aggregate Advisor runs on the query service and can reference a workload log file.

If you want Aggregate Advisor to consider information from workload logs when making recommendations, you must enable the workload log file on the dynamic cube. Then, run a representative

set of reports and queries so that a comprehensive workload is captured in the workload log before running Aggregate Advisor.

When enabled, the workload log file captures the information that represents user workload usage, such as running reports. This log file allows Aggregate Advisor to suggest aggregates, in-database or in-memory, that correspond directly to the reports contained in the log file.

To enable the dynamic cube workload log file, use the **Enable workload logging** cube property. When you enable or disable workload logging in IBM Cognos Administration, you do not need to restart the dynamic cube. However, you might need to wait a few seconds, no longer than a minute, for the property change to take effect.

Workload logging starts or stops as a result of changing the **Enable workload logging** property if the cube is in a running or paused state. If the cube is in another state, for example, the cube is starting, the property change does not take effect when the cube reaches the running state, but when the cube is restarted the next time. To avoid waiting for the next cube restart, you can re-save the property after the cube starts running.

For more information about specifying dynamic cube properties, see [“Setting dynamic cube properties” on page 132](#).

For more information about using Aggregate Advisor, see *IBM Cognos Dynamic Query Analyzer User Guide*.

Clearing the workload log

Clearing the workload log removes all entries for a dynamic cube from this log. This is useful if you want to capture new information about report usage.

This action clears only the workload log entries that are captured in conjunction with the dynamic cube property **Enable workload logging**. This action does not clear the workload activity information that is captured by a cube that has the **Enable automatic optimization of in-memory aggregates** property enabled.

You can create and schedule query service tasks for clearing the workload. For more information see [“Creating and scheduling query service administration tasks” on page 138](#).

You can also clear the workload manually. For more information, see [“Starting and managing dynamic cubes” on page 127](#).

Automatic optimization of in-memory aggregates

When you enable this feature, the system continually analyzes the workload activity and automatically optimizes the set of in-memory aggregates in response to the report queries.

Automatic optimization of in-memory aggregates has the following benefits:

- It minimizes the number of manual Aggregate Advisor runs and reduces the need to generate comprehensive workload logs.
- It improves report performance by adjusting the set of in-memory aggregates over time to better match query activity.
- It complements the near real-time updates of dynamic cubes.
- It reduces the cost of dynamic cube ownership.

When automatic optimization of in-memory aggregates is enabled, the workload is logged automatically. Aggregate Advisor runs automatically in the background, quickly analyzes the workload, recommends new and more effective in-memory aggregates, and applies them to the content store. The server then automatically, one in-memory aggregate at a time, loads or removes the in-memory aggregates from the running instance of the cube. If this feature is enabled for multiple cubes on a server, the automatic optimization steps are performed sequentially, one cube at a time. This helps to minimize the impact on the live system, which includes the query service and the database server.

Because the workload activity is logged automatically and not filtered, you do not need to manually enable workload logging and capture a comprehensive workload ahead of time. The system adjusts the

set of in-memory aggregates over time, in a conservative manner. For example, the system creates additional in-memory aggregates if it estimates that there is enough memory space. If there is a possibility that the space might be exceeded, the system tries to make an intelligent compromise between the previously- and newly-recommended in-memory aggregates. The system is especially cautious when it recommends to remove aggregates. This approach results in minimal changes to the set of in-memory aggregates, and when the in-memory aggregates are loaded one at a time, minimizes the impact on the system.

User-defined in-memory aggregates, if they are in the model, are always included in the set of in-memory aggregates, regardless of the memory space estimates or matching to in-database aggregates.

Automatic optimization of in-memory aggregates works best in the following situations:

- In-database aggregates are either not required in the model, or are stable and do not include slicers.
- Additive measures are used in the model.

Non-additive measures cannot rollup from in-database aggregates and can result in a number of in-memory aggregates to provide a direct match for queries.

If you are using multiple dispatchers for the query service, only the server with the dynamic cube property enabled has its in-memory aggregates automatically and continually optimized. The cube on other servers synchronizes and loads in-memory aggregates when it starts.

To enable automatic optimization of in-memory aggregates, turn on the dynamic cube property **Enable automatic optimization of in-memory aggregates**, and set the dynamic cube property **Maximum space for in-memory aggregates (MB)** to a value greater than 0. For more information, see [“Starting and managing dynamic cubes”](#) on page 127.

When automatic optimization of in-memory aggregates is enabled, you can use the following, optional, query service **Advanced settings** to configure this functionality:

qsAutomaticAggregateOptimizationMatchInDatabaseAggregates

By default, Aggregate Advisor recommends in-memory aggregates that are based only on the workload, which means that the in-memory aggregates are loaded either from the in-database aggregates or from the fact table. Loading an in-memory aggregate that is based on a large fact table takes a long time.

Load performance is improved if the in-memory aggregates load from in-database aggregates that are smaller than the fact table. To ensure that your in-memory aggregates load from in-database aggregates, change this setting to `True`. As a result, Aggregate Advisor recommends only in-memory aggregates that match the in-database aggregates. Because the in-memory aggregates do not have slicers, Aggregate Advisor ignores any in-database aggregates with slicers when assessing whether an in-memory aggregate would match.

Value: True or False

Default: False

Tip: When you enable automatic optimization of in-memory aggregates, Aggregate Advisor does not recommend in-database aggregates. Users must create and model the in-database aggregates in the cube. The users can either create the in-database aggregates themselves, or manually run Aggregate Advisor to obtain in-database recommendations.

qsAutomaticAggregateOptimizationStartTime

By default, the system determines when to run Aggregate Advisor and load in-memory aggregates. Use this property if you prefer to start this activity at a specific time.

Value: 00:00 to 23:59. This value is based on a 24-hour clock. For example, if you specify 23:00, the automatic optimization of the dynamic cubes on the server occurs nightly, starting at 11:00 PM.

Default: Empty string

qsAutomaticAggregateOptimizationMaxConcurrentCubeTasks

By default, the system performs automatic optimization of one cube at a time. For example, if there are three cubes on a server that is enabled for automatic optimization of in-memory aggregates, the system automatically runs Aggregate Advisor and loads any recommended aggregates for the first

cube. Once complete, this action is repeated for the second cube, and then for the third cube. This type of processing minimizes the load on the query service and database servers.

You can change this setting to specify the number of cubes to be optimized concurrently. This is typically done when the cubes are configured for automatic optimization at a specified time (the `qsAutomaticAggregateOptimizationStartTime` advanced setting is configured to a non-default value), preferably during a maintenance window when the system is lightly used. However, if optimization occurs throughout the day, which is the default behavior (the `qsAutomaticAggregateOptimizationStartTimeset` setting is configured to use the default value), you must be cautious about changing this setting.

Value: Positive integer starting with 1

Default: 1

Tip: This advanced setting does not require a query service restart.

For information about configuring the query service **Advanced settings**, see [“Setting query service properties for dynamic cubes”](#) on page 129.

Setting general properties for a dynamic cube

You can view and edit general properties of an individual dynamic cube data source.

Procedure

1. In **IBM Cognos Administration**, on the **Status** tab, click **Dynamic Cubes**.
2. In the **Scorecard** section, select the **Dynamic Cubes - (All)** view.
3. For the dynamic cube that you want to modify, click the **Actions** drop-down menu and click **Set properties**.
4. On the **General** tab, view or change the following properties as required:

Type

The type of property. For example, a Dynamic Cubes database, a Dispatcher, or a Namespace are all a type of property.

Owner

The owner of the entry. By default, the owner is the person who created the entry. When the owner no longer exists in the namespace, or is from a different namespace than the current user, the owner shows as **Unknown**.

If you have Set policy permissions, then you can click **Make me the owner** to become the owner of the entry.

Contact

The person responsible for the entry. Click **Set the contact** and then click **Select the contact** to set the contact for the entry or click **Enter an email address** to enter the contact email address.

Location

The location of the entry in the portal and its ID. Click **View the search path, ID and URL** to view the fully qualified location and the ID of the entry in the content store.

Entries are assigned a unique identification (ID) number.

Created

The date the entry was created.

Modified

The most recent date that the entry was modified.

Icon

The icon for the entry. Click **Edit** to specify a different icon.


Disable this entry

When selected, users that do not have write permission for this entry cannot access it. The entry is no longer visible in the portal.

If an entry is disabled and you have write permission to it, the disabled icon is displayed next to the entry.

Hide this entry

Select this property to hide reports, packages, pages, folders, jobs, and other entries. Hide an entry to prevent it from unnecessary use, or to organize your view. The hidden entry is still accessible to other entries. For example, a hidden report is accessible as a drill-through target.

A hidden entry remains visible, but its icon is faded. If you clear the **Show hidden entries** check box in my area options , **My Preferences**, the entry disappears from your view.

You must have access to the **Hide Entries** capability granted by your administrator to see this property.

Language

A list of languages that are available for the entry name, screen tip, and description according to the configuration that was set up by your administrator.

Name

The name of the entry for the selected language.

Note: Renaming a dynamic query cube can cause problems for objects that reference this cube. For this reason, you should not change the name of the dynamic cube data source.

Screen tip

An optional description of the entry. The screen tip displays when you pause your pointer over the icon for the entry in the portal. Up to 100 characters can be used for a screen tip.

Description

An optional description of the entry, which displays in the portal when you set your preferences to use the details view.

Details view is displayed only in Public Folders and My Folders.

Access Account

The access account is used by the dynamic cube data source to access the relational database. The dynamic cube uses the data source signon credentials to access the relational database that contains the data warehouse of a dynamic cube. You can select which Cognos account to use based on its credentials. You must create the credentials before you define the access account.

For more information about defining the access account, see [“Assigning data access accounts for dynamic cubes”](#) on page 123.

Creating and scheduling query service administration tasks

Administrators can create and schedule query service tasks for dynamic cube data sources. For example, you can schedule cache clearing, and clear the cache to control memory usage by a specific data source or cube.

The following query service tasks can be scheduled for one or more cubes:

- Clear workload log.
- Refresh data cache.
- Refresh member cache.
- Refresh security settings.
- Restart.
- Start.
- Start cube and source cubes.
- Stop after active tasks complete.
- Stop immediately.

You can create query service administration tasks and run them on demand. You can run the administration tasks at a scheduled time or based on a trigger, such as a database refresh or an email.

You can schedule them as part of a job. You can also view the run history of query service administration tasks. For more information, see the *IBM Cognos Analytics Administration and Security Guide*.


Before you begin

When you create and schedule tasks for dynamic cubes, you must schedule start and stop tasks for source cubes and virtual cubes separately. Consider the following factors when scheduling start and stop tasks for dynamic cubes:

- Source cubes that are part of a virtual cube must be scheduled to start first.
- If source cubes are part of a virtual cube, the virtual cube must be scheduled to stop before the source cubes.
- You need to provide enough time for source cubes to start before scheduling a virtual cube to start. The same condition applies when you schedule virtual and source cubes to stop.

To start virtual cubes, you can use the **Start cube and source cubes** action.

Procedure

1. In IBM Cognos Administration, on the **Configuration** tab, click **Content Administration**.
2. In the page toolbar, click the **New Query service administration task** icon , and then click **Dynamic cube**.
3. Specify a name, description, screen tip, and location for the new task, and click **Next**.
4. Select an operation.

For detailed information about the different actions, see [“Starting and managing dynamic cubes” on page 127](#).

5. Select the required **Server Group**, **Dispatcher**, and **Cubes**, and click **Next**.
6. Choose how to run the task:
 - To run the task now or later, click **Save and run once** and click **Finish**. Specify a time and date for the run, and then click **Run**.
 - To schedule the task at a recurring time, click **Save and schedule** and click **Finish**. Then, select frequency and start and end dates.
Tip: To temporarily disable the schedule, select the **Disable the schedule** check box.
 - To save the task without scheduling or running, click **Save only** and click **Finish**.

Results

After they are saved, the query service administration tasks appear on the **Configuration** tab, in **Content Administration**.

What to do next

You must delete a scheduled task if you delete the associated cube from the query service. Otherwise, your scheduled tasks will point to a cube that no longer exists.

Setting access permissions for security views

The model contains the security views that were defined for the dynamic cube in IBM Cognos Cube Designer. Administrators set access permissions for the security views.

About this task

Security views can be accessed from the model within a dynamic cube data source. A model view in IBM Cognos Administration is equivalent to a security view in Cognos Cube Designer.

By default, when a dynamic cube is published to the content store, the group **Everyone** has access to the model view. Administrators must override the access permissions to remove **Everyone** and add the appropriate users, groups, or roles to the model view.

Only read permissions are required to give the users, groups, or roles access to the metadata in a dynamic cube.

Procedure

1. In **IBM Cognos Administration**, on the **Status** tab, click **Dynamic Cubes**.

In the **Scorecard** section, you see a list of all published dynamic cube data sources in the IBM Cognos Analytics environment.

2. Point to the data source that you want to edit, and from the **Actions** drop-down menu, click **Edit security view permissions**.

The available security views are listed in the model.

3. For the selected security view, in the **Actions** column, click the **Set properties** icon.

4. Choose whether to use the permissions of the parent entry or specify permissions specifically for the entry:

- To use the permissions of the parent entry, clear the **Override the access permissions acquired from the parent entry** check box, then click **OK** if you are prompted to use the parent permissions.
- To set access permissions for the entry, select the **Override the access permissions acquired from the parent entry** check box, and proceed to step 5.

5. Optional: If you want to remove an entry from the list, select its check box and click **Remove**.

Tip: If you want to select all entries, select the check box at the top of the list. Clear the check box to deselect all entries.

6. To specify the entries for which you want to grant or deny access, click **Add**, and choose how to select entries:

- To choose from listed entries, click the appropriate namespace, and then select the check boxes next to the users, groups, or roles.
- To search for entries, click **Search** and in the **Search string** box, type the phrase you want to search for. For search options, click **Edit**. Find and click the entry you want.
- To type the name of entries that you want to add, click **Type** and type the names of groups, roles, or users using the following format, where a semicolon (;) separates each entry:

```
namespace/group_name;namespace/role_name;namespace/user_name;
```

Here is an example: Cognos/Authors;LDAP/scarter;

7. Click the arrow icon to move the selected entry to the **Selected entries** box, and when all required entries are in this box, click **OK**.

Tip: To remove entries from the **Selected entries** box, select them and click **Remove**. To select all entries in a list, click the check box in the upper-left corner of the list. To make the user entries visible, click **Show users in the list**.

8. Grant read permissions for each entry in the list, and click **OK**.

Tip: In the **Permissions** column, an icon appears next to the user, group, or role. This icon represents the type of access granted or denied to the entry.

9. If you want to remove access permissions that were previously set for the child entries so that the child entries can acquire permissions set for this entry, in the **Option** section, select the **Delete the access permissions of all child entries** check box.

This option appears only with entries that are containers. You can use it to restrict access to a hierarchy of entries. Select this option only when you are certain that changing access permissions of the child entries is safe.

Memory monitoring in the dynamic query mode server

By default, the dynamic query mode server monitors its use of the IBM or Oracle Java virtual machine (JVM) heaps.

If the dynamic query mode server detects that the amount of available memory is 10% or less, it becomes overloaded, and performs the following actions to avoid running out of memory:

- It rejects any subsequent queries from starting.
- It cancels selected queries until available memory is 10% or more of the JVM heap.

If a cube is refreshing its member cache or if it is restarting, notably on a system with another cube that is actively processing queries, loading members might push the query service beyond its available memory. If this situation occurs, the query service cancels the queries to protect the availability of the cubes that are already active.

The cancellation of the member loading queries in turn causes the cube start or refresh to fail. Therefore, a subsequent refresh or start of the cube is required, ideally when more memory is available.

In the presence of a query being canceled on one server due to insufficient memory, the original report or analysis is now routed to another server within the server group. This process continues until either the report or query runs successfully or all servers cancel the query due to insufficient memory. At this point, an error is returned to the user.

Queries are selected for cancellation by analyzing the running time and size of each query in progress, and ranking them according to their impact on the server. The impact is determined by these factors (listed in order of importance):

- The largest set created during the request.
- The number of data points added to the data cache.
- The request running time.

After determining impact, the dynamic query mode server cancels queries in this order:

1. It cancels the query with the highest rank.
2. If step 1 does not resolve the issue, it cancels next 30% of highest ranked queries.
3. If step 2 does not resolve the issue, it cancels all remaining queries.

If these actions do not resolve the available memory issue, the dynamic query mode server stops and restarts after a delay of 5 minutes to allow any orphaned database queries to be cancelled on the database server.

When the dynamic query mode server rejects new incoming queries, the following error is shown:

```
The query was canceled because the server is low on memory.  
You can try to execute the query later. If the  
problem persists, contact your system administrator.
```

When the dynamic query mode server cancels queries in progress, the following error is shown:

```
Your request could not be completed now because the system is busy.  
Please try again later.
```

Errors that are related to low available memory are also saved to a log file. You can view and analyze them in the Resources.Monitor category of the *cognos_analytics_location/logs/XQE* log file.

You can modify various memory monitoring settings to suit your own circumstances. For more information, see [“Configuring the dynamic query mode server monitoring settings”](#) on page 141.

Configuring the dynamic query mode server monitoring settings

You can change the default behaviour of the memory monitoring feature (the Resource Monitor) within the dynamic query mode server when the server becomes overloaded.

To configure the Resource Monitor, add the Java command line argument **-D** to the **Additional JVM arguments for the query service** property in the query service settings, then append the appropriate Resource Monitor setting. For example, to change the maximum IBM Java virtual machine (JVM) heap in use that is considered normal to 95, add the following string:
-DresourceMonitor.overloadedPercent=95

For more information about configuring query service settings, see [“Setting query service properties for dynamic cubes” on page 129](#).

You can also configure the Resource Monitor by copying the file *cognos_analytics_location/configuration/xqe.config.xml*, and renaming it to *xqe.config.custom.xml* for editing. You can then add the appropriate Resource Monitor setting to this file. For example, to change the maximum IBM JVM heap in use that is considered normal to 95, add the following section:

```
<resourceMonitor>
  <overloadedPercent>95</overloadedPercent>
</resourceMonitor>
```

Tip: If a command line argument differs from the corresponding setting in the *xqe.config.xml* file, the command line argument takes precedent.

The following table lists the different settings available for the Resource Monitor.

| Setting name | Description | Values |
|--|--|--------------------------------|
| <code>resourceMonitor.enabled</code> | Enables or disables the Resource Monitor. | true (default) false |
| <code>resourceMonitor.overloadedPercent</code> | The maximum percentage of IBM JVM heap in use that is considered normal. When memory use exceeds this level, the dynamic query mode server is overloaded. Increasing this value provides the dynamic query mode server with more memory, but the risk of running out of memory is also higher. | 90 (default) 75-100 |
| <code>resourceMonitor.maxQueries</code> | Limits the maximum number of internal SQL queries that are generated by a single report in the dynamic query mode server. When this limit is reached, the report is canceled. Other reports are not affected. Setting this value to 100 impacts the server. Lowering this value allows the dynamic query mode server to cancel less complex queries. | 100000 (default) 1-10000000 |

Table 47. Resource Monitor settings (continued)

| Setting name | Description | Values |
|---|--|-------------------------------------|
| <code>resourceMonitor.cancelDelay</code> | <p>The number of seconds that the dynamic query mode server must wait between successive sets of query cancelations until memory use returns to normal.</p> <p>After the dynamic query mode server cancels the highest impact query, it waits for the specified period before it continues with subsequent cancelations.</p> <p>Increasing this value results in fewer cancelations, and allows a longer time for memory to clear after the initial cancellation. However, it also increases the risk of a memory error if memory is not cleared in the specified time period.</p> | <p>10 (default)</p> <p>1-600</p> |
| <code>resourceMonitor.cancelRampupPercentage</code> | <p>The percentage of queries to cancel after the initial, highest rank query is canceled, but before attempts to cancel all remaining queries.</p> <p>Increasing this value results in a higher number of canceled queries after the initial cancellation, which might adversely affect users.</p> | <p>30 (default)</p> <p>1-100</p> |
| <code>resourceMonitor.ballastEnabled</code> | <p>Enables a block of memory (ballast) to be reserved that is later released on low memory. The ballast allows the dynamic query mode server to manage queries that need to be canceled.</p> <p>You can disable this setting if the ballast is destabilizing your system or using too much memory.</p> | <p>true (default)</p> <p>false</p> |
| <code>resourceMonitor.ballastPercentage</code> | <p>The percentage of memory to use as a ballast.</p> | <p>2 (default)</p> <p>1-10</p> |
| <code>resourceMonitor.gcEnabled</code> | <p>Enables periodic Java garbage collection requests when the dynamic query mode server is overloaded. This option encourages Java to free memory more frequently.</p> | <p>true (default)</p> <p>false</p> |
| <code>resourceMonitor.gcRetryPeriod</code> | <p>The number of seconds that the dynamic query mode server must wait before it attempts a specific garbage collection request again.</p> | <p>120 (default)</p> <p>10-3600</p> |

Table 47. Resource Monitor settings (continued)

| Setting name | Description | Values |
|------------------------------|--|---------------------|
| resourceMonitor.gcIterations | The number of times to request garbage collection from Java in each period. Increasing this value results in a higher number of requests, but might also result in longer pauses between garbage collection requests. | 1 (default) 1-10 |

You can also configure the following related settings in the **Advanced settings** property in the query service.

Table 48. Advanced settings

| Setting name | Description | Values |
|--------------------------------|---|-------------------------|
| qsMaxCrossjoinOrderOfMagnitude | The maximum size of a cross join in the MDX engine. The value is defined as an order of magnitude, for example, $\log_{10}(\text{value})$. For example, $\log_{10}(1000) = 3$. To disable the limit, set the value to 0. | 8 (default) 0-10 |
| qsCubeStartDelayOnRecovery | The number of seconds to delay starting a dynamic cube when the dynamic query mode server restarts due to a critical failure. Increasing this value lowers the impact on the database if the dynamic query mode server continuously fails and restarts, and dynamic cubes load, while previous queries are still running and not canceled. | 300 (default) 0-3600 |

For more information about configuring query service advanced settings, see [“Setting query service properties for dynamic cubes”](#) on page 129.

Enabling IPF logging for Cognos Cube Designer

You can record activities and debugging information for IBM Cognos Cube Designer using the IBM Cognos Analytics logging mechanism called Indication Processing Facility (IPF).

To enable IPF logging for any Cognos Analytics component, the file `ipfclientconfig.xml` must be present in the `cognos_analytics_location/configuration` directory. The same directory contains the `ipfCubeDesignerclientconfig.xml.sample` file that defines all the logging categories available for Cognos Cube Designer. To enable logging for Cognos Cube Designer, you only need to rename the `ipfCubeDesignerclientconfig.xml.sample` file to `ipfclientconfig.xml`.

The following logging categories are defined in the `ipfCubeDesignerclientconfig.xml.sample` file.

Trace.fmeng.memory

Logs memory information, such as the amount of used, available, and free memory, for the whole process.

Trace.fmeng.platform

Logs platform-related messages, such as information about session management. All exceptions are logged to this category.

Trace.fmeng.metadata

Logs messages related to fetching metadata.

Trace.fmeng.import.cubingServices

Logs messages related to importing IBM InfoSphere Warehouse Cubing Services cubes. These are the same messages as in the log text file that is generated after importing a cube.

Trace.fmeng.import.frameworkManager

Logs messages related to importing objects from the classic Framework Manager model.

Trace.fmeng.publish

Logs messages related to publishing models, starting cubes, and so on.

Trace.fmeng.error

Logs exceptions.

The Cognos Cube Designer events are logged in the `fmeng_trace.log` file in the `cognos_analytics_location/logs` directory. If a logging database is defined in IBM Cognos Configuration under **Environment > Logging**, the Cognos Cube Designer events are also logged in this database. The level of details logged in the `fmeng_trace.log` file depends on the logging level defined for each category in the `ipfCubeDesignerclientconfig.xml` file. The debug logging level allows to log all events.

You do not need to restart the IBM Cognos service after you enable or disable logging.

Procedure

1. In the `cognos_analytics_location/configuration` directory, make a copy of the `ipfCubeDesignerclientconfig.xml.sample` file and save it as `ipfclientconfig.xml`.

Important: The `ipfclientconfig.xml` file is used for logging by different Cognos Analytics components. If this file is already present in the `cognos_analytics_location/configuration` directory, contact your Cognos administrator to ensure that you can overwrite this file.

2. Open the `ipfclientconfig.xml` file, uncomment the logging categories that you want to use, and save the file.
3. If you later need to disable logging for Cognos Cube Designer, rename the `ipfclientconfig.xml` file to any name.

Results

The Cognos Cube Designer events are logged in the `fmeng_trace.log` file.

Chapter 13. Near real-time updates of dynamic cubes data

With near-real time updates, data can be inserted into fact and aggregate tables in the data warehouse without stopping a dynamic cube.

New data records added to a fact table can be applied to a dynamic cube incrementally, on demand. The data caches are updated, and not rebuilt.

The advantages of using near real-time updates are:

- Data and caches are loaded to a dynamic cube only once. The cube is available for queries at any time after the initial load.
- Aggregate table data is updated separately from the fact table data. You can choose when to run maintenance updates on aggregate tables.
- Low latency is achieved without negative impact on performance.
- Databases can be changed while the cube runs.
- Queries to the cube are consistent to a point in time, even if they require multiple database or cache accesses.

You can load fact table updates outside of your regular maintenance window. This reduces the amount of time that is needed for regular maintenance. The only time you must stop a dynamic cube for updates is when either update or delete changes are made to a fact table.

Limitations

Currently, near real-time updates are limited to new fact rows only. It is not possible to apply near real-time updates to the following items:

- Updated or deleted rows in the fact table
- New, updated, or deleted rows in dimension tables
- Measures with Custom (Unknown) aggregate type
- Virtual cubes with the data cache and result set cache enabled

Enabling near real-time updates for dynamic cubes

To enable near real-time updates, you must add a nullable transaction ID (TID) column to each fact table.

Before you begin

Before you load fact data to a dynamic cube, you insert new rows into the fact tables in the data source. New rows for near real-time updates must adhere to the following rules:

- Each insert transaction must use a TID value greater than any previous transaction.
- All rows can use the same TID value within a single transaction.

Procedure

1. Add a nullable transaction ID (TID) column to each fact table.

The data type for this column can be set to any type that supports SQL comparison operators and MAX SQL functions. You can use the BIGINT, INTEGER, or TIMESTAMP data type.

To improve query performance, create an index on the TID column.

For the initial fact data, set the TID column to Null. Any other TID value implies an incremental update to the fact data. This is illustrated in the following example.

| <i>Table 49. Fact table</i> | | |
|-----------------------------|--------------|-----------------------------|
| Product | Sales | Transaction ID (TID) |
| Paper | 50 | |
| Pen | 75 | |
| Paper | 45 | |
| Paper | 5 | |
| Paper | 20 | 1 |
| Paper | 5 | 1 |
| Paper | 25 | 2 |

Create the aggregate tables from the initial fact data only (rows with a null TID column), as illustrated in this table.

| <i>Table 50. Aggregate table for initial fact data</i> | |
|--|------------------------|
| Product | Aggregate sales |
| Paper | 100 |
| Pen | 75 |

2. Identify the TID column in the dynamic cube using Cognos Cube Designer:
 - a) From the **Project Explorer** tree, expand your cube.
 - b) Select the measure dimension folder.
 - c) In the **Properties** pane, select the TID column from the **Transaction ID** drop-down list.
3. Publish the dynamic cube.

Results

When you start a published dynamic cube, the query service performs the following tasks:

- Checks for the highest TID value and uses it for the initial load.
 In the example fact table in step 1, there are rows of initial fact data with null TID values, and rows of updates for two increments with TID values 1 and 2. In this case, the query service uses the TID value 2 for the initial load.
- Loads the aggregate cache and query data cache with aggregate table and fact data based on the initial load state.

Loading incremental updates to dynamic cubes

After you load initial fact data, you can add new fact rows to the fact table at any time. After you start a dynamic cube and update the fact table with new rows, make the updates visible in a dynamic cube by using an incremental load.

To identify the new rows to dynamic cubes, you must use a non-null TID value for the rows that is higher than the TID value for the previously inserted rows. For example, if a previous update to the fact data used a TID value 2, the next update must use a TID value of 3 or higher.

You can load more than one increment at a time. For example, if you have updates for TID values 3, 4 and 5, you can load them all at once. Alternatively, you can specify that you want to load only incremental updates up to TID value 4.

Important: Assign the same TID value to all fact rows that are loaded together.

While incremental updates to a dynamic cube are in progress, queries against the cube return values that are based on the current completed update. When the update is complete, and data caches are updated, new queries return values that are based on the latest incremental update.

An incremental update is a memory-intensive process that requires additional memory for the duration of the incremental load and during the dynamic cube operation. The following examples illustrate how to estimate the amount of memory that is needed in both situations:

- Additional memory requirements during an incremental load.

You should plan for 500 bytes of memory for each new tuple. A tuple is defined as the number of tuples processed in the incremental update. For example, 10M tuples requires 5 GB of extra memory to load. This is calculated using the following formula: number of unique rows at the grain of the cube times the number of additive measures.

In this formula, the number of unique rows at the grain of the cube is the number of unique rows that are affected at the grain of the cube. This value is used by a query to fetch the incremental values. It must always be equal to or less than the number of inserted rows. It might be less than the number of inserted rows if the grain of the cube is higher. For example, the cube is modeled to hour, but rows are inserted to the minute.

The number of inserted rows in this formula is the number of rows in the fact table.

- Additional memory requirements during dynamic cube operation.

Tuples for the latest increment are saved after the **incrementallyLoadCubes** command finishes, at a cost of 100 bytes per tuple. For example, a 10M increment requires an extra 1 GB of memory. This extra memory is required while the cube runs and applies to the last set of loaded tuples. For example, if you incrementally load a cube ten times, when the load commands are finished the required extra memory is: 100 bytes times the number of tuples in the last load.

You can load updates to aggregate tables separately, and you can choose when to run these updates. For more information, see [“Incremental updates of aggregate tables” on page 150](#).

You can load incremental updates to dynamic cubes by using the **Incrementally update data** action in IBM Cognos Administration. This method allows you to run these commands by schedule and by trigger. For more information see [“Starting and managing dynamic cubes” on page 127](#).

You can also load incremental updates to dynamic cubes by using the DCAdmin command-line tool, as shown in the following procedure.

Procedure

Perform the following steps in the DCAdmin command-line tool to load an incremental update:

1. Open the DCAdmin command-line tool. For information about running the tool, see [Appendix C, “DCAdmin command-line tool,” on page 187](#).
2. Run the **getCubeMetrics** command to check the following metrics:
 - The metric **timeLastNearRealTimeUpdateAvailable** returns the date and time when the latest increment was loaded.
 - The metric **timeToApplyLastNearRealTimeUpdates** returns the time used to build the latest increment.
 - The metric **valueOfLastNearRealTimeTID** returns the TID value of the latest increment.

By checking these metrics, you can determine which TID value was loaded last and decide how frequently the dynamic cube should be updated.

3. Run the **incrementallyLoadCubes** command.

This command includes a `transactionID` parameter that you can use to specify the transaction ID (TID) value to which to load fact data updates. If you do not specify this parameter, the command runs a MAX query to determine the latest TID value. Use the `transactionID` parameter for non-indexed

databases, such as Netezza® and IBM Db2 BLU, where performance might be adversely affected by using a MAX query. For indexed databases, such as Db2 and Oracle, running a MAX query has no adverse effect, therefore, it is not necessary to use this parameter.

4. Optional: Run the **getCubeMetrics** command again to check if the updates were successful.

Results

When this update and the data caches updates are complete, new queries return values that are based on the latest incremental update.

Incremental updates of aggregate tables

It is not necessary to update fact tables and aggregate tables simultaneously when you use a dynamic cube that is configured for near real-time updates. This reduces the need for frequent aggregate table maintenance cycles. You might choose to never update aggregate tables if they are accessed only by a dynamic cube.

When querying an aggregate table for a dynamic cube with near real-time updates, to determine the correct aggregate value, the query engine checks the fact table for the latest update and combines this result with the aggregate table to obtain a combined value. Over time, as new rows are added to the fact table that are not reflected in the aggregate tables, the SQL queries might slow down. To restore the performance, you need to update the aggregate tables.

Aggregate table updates must adhere to the following rules:

- Any row with a null TID value must be included in all aggregate tables.
- Any row with a non-null TID value must not be included in any aggregate table.
- An aggregate table cannot be updated to past the point of the latest incremental load if the dynamic cube is still running.

For example, table 3 in this section has three non-null TIDs, 1, 2, and 3. If the last incremental load went only to TID 2, the aggregate table can be built to include only TID 1 data, or TID 1 and TID 2 data, but not TID 3 data.

Before you update an aggregate table, run an incremental load of fact data up to a specific TID value. For more information, see [“Loading incremental updates to dynamic cubes”](#) on page 148. Then, update the aggregate table up to the same TID value. This ensures that the maintenance of fact and aggregate tables does not reset to null the TID value for rows that have not yet been processed. This also ensures consistency between the fact and aggregate tables.

The following example illustrates how to reset the TID values for the fact table, resume a dynamic cube with the correct TID value, and set the TID value for the future fact table updates when the last incremental load of fact table was for the TID value 3.

| Product | Sales | TID |
|---------|-------|-----|
| Paper | 50 | |
| Pen | 75 | |
| Paper | 45 | |
| Paper | 5 | |
| Paper | 20 | 1 |
| Paper | 5 | 1 |
| Paper | 25 | 2 |
| Pen | 25 | 3 |

For consistency, you must also update the aggregate table up to TID value 3.

| <i>Table 52. Aggregate table before an incremental update</i> | |
|---|-----------------|
| Product | Aggregate sales |
| Paper | 100 |
| Pen | 75 |

The following tables show the same fact table and aggregate table after the update when all TID values less than 3 were reset to null.

| <i>Table 53. Fact table after the update</i> | | |
|--|-------|-----|
| Product | Sales | TID |
| Paper | 50 | |
| Pen | 75 | |
| Paper | 45 | |
| Paper | 5 | |
| Paper | 20 | |
| Paper | 5 | |
| Paper | 25 | |
| Pen | 25 | |

| <i>Table 54. Aggregate table after the update</i> | |
|---|-----------------|
| Product | Aggregate sales |
| Paper | 150 |
| Pen | 100 |

When you add rows to the fact table next time, specify the TID value of 4 to maintain consistency.

Important: All future TID values for fact table updates must be higher than all previous TID values even if all previous TID values were updated in the aggregate table.

If you choose to update aggregate tables, you have the following options:

- Stop the cube.

You can then build the aggregates, reset the transaction ID (TID) column value in the fact table to null, and restart the cube. This option requires a reload of data caches and should be used when you stop a dynamic cube for other updates that require a cube restart, such as cube model updates, in-memory aggregate definition updates, and cube property updates.

- Pause the cube.

When paused, a dynamic cube continues to run so that the data caches remain valid, but is unavailable for queries by report users. For more information, see [“Pausing a dynamic cube to update aggregate tables”](#) on page 151.

You can then build the aggregates, reset the transaction ID (TID) column value in the fact table to null, and resume the cube with the last TID value. There is no need to reload data caches.

Pausing a dynamic cube to update aggregate tables

You can pause a dynamic cube that uses near real-time updates to update the aggregate tables. This is a less memory-intensive method of updating aggregate tables because it does not require a reload of data caches.

You can pause a dynamic cube by using the **Pause** action in IBM Cognos Administration. For more information see [“Starting and managing dynamic cubes” on page 127](#).

You can also pause a dynamic cube by using the DCAdmin command-line tool, as shown in the following procedure.

Procedure

Refer to the following steps when you use the DCAdmin command-line tool to pause a dynamic cube.

1. Pause the dynamic cube by using the **pauseCube** command that is available from the DCAdmin tool.

For information about the DCAdmin commands, see [Appendix C, “DCAdmin command-line tool,” on page 187](#).

2. Check that the cube is paused by using the **getCubeState** command.
3. Update the aggregate tables up to a specified TID value.

This value can be less than or equal to the TID value that was used for the last incremental load of fact data. You should update to the latest TID value to achieve optimal future performance.

4. Reset the TID values to null in the fact table for all rows that are rolled into the aggregate table.
5. Resume the cube by using the **startCube** command.
6. Check that the cube is resumed by using the **getCubeState** command.

Chapter 14. Relational and DMR modeling in Cognos Cube Designer

Although the primary function of IBM Cognos Cube Designer is to create dynamic cubes, you can also use it to create relational and dimensionally modeled relational (DMR) models, as in IBM Cognos Framework Manager.

Use Cognos Cube Designer for relational and DMR modeling when Framework Manager does not satisfy your needs in certain areas. For example, users with limited vision might prefer to use Cognos Cube Designer because of accessibility features that Framework Manager does not offer.

Some objects that are used when modeling dynamic cubes are not compatible with relational and DMR metadata. Such objects include: virtual cubes, parent-child dimensions, relative time dimensions, named sets, calculated measures, and calculated members.

Important: You cannot use Cognos Cube Designer to work with relational or DMR models that were created using Framework Manager.

Differences when modeling with Cognos Cube Designer and Framework Manager

Framework Manager is a mature product that has benefited from many years of iterative improvements to its user experience. The relational and DMR experience in Cognos Cube Designer 10.2.2 is in its early stage and does not currently offer all of the usability features that Framework Manager does, including the ability to:

- Set properties for multiple items at once.
- View context diagram for modeling relationships.
- View data for query items from different query subjects.
- View generated SQL for a selection of objects.
- Copy and paste query subjects.

The fundamental approach to relational and DMR modeling in Cognos Cube Designer is different from that of Framework Manager.

Modeling with Framework Manager is an iterative process of refining different views, or layers, of your metadata, starting with the data source view, then the business view, and finally the presentation view that your users consume.

The modeling approach with Cognos Cube Designer is streamlined to focus entirely on the reporting application and ensure that a quality model is defined and deployed. Cognos Cube Designer guides your modeling activities to allow you to succeed in your reporting requirements.

In Framework Manager, you create data source query subjects by importing tables and views from your data source into your model. You also create model query subjects that are not generated directly from a data source, but are based on query items in other query subjects, including other model query subjects.

A query subject in Cognos Cube Designer has similarities to the model query subject in Framework Manager. In Cognos Cube Designer, you first create a query subject and then add query items from your data source to it. Similarly to how model query subjects in Framework Manager provide an insulating layer from schematic changes to the data source, query subjects in Cognos Cube Designer remain static from a reporting perspective while their underlying implementation is updated to reflect the new database structure. In Cognos Cube Designer, you can also create a query item set to obtain an abstract, business-oriented collection of query items.

The security paradigms in Cognos Cube Designer and Framework Manager are different. In Framework Manager, you associate security filters with users, groups, and roles whose unique identifiers are stored within the model. In Cognos Cube Designer, you create security filters within named security views. You then associate the security views with users, groups and roles; these associations are stored in your

Content Manager instances. This implementation allows for greater portability of the model between different Cognos Analytics environments. This type of security is similar to the security paradigm used by IBM Cognos Transformer.

Enabling relational modeling

To model relational and DMR metadata in IBM Cognos Cube Designer, you must enable relational modeling. Only then can you access the product features that allow you to work with relational and DMR objects.

Enabling this functionality has no impact on dynamic cubes. You can work with your dynamic cubes with relational modeling enabled or disabled.

Procedure

From the **File** menu in Cognos Cube Designer, select the **Enable Relational Modeling** check box.

You can select or clear this check box at any time during your modeling activities. If you clear this check box while modeling a relational or DMR model, you will not lose your work. However, to continue your relational or DMR modeling, you must select this check box again.

Creating a relational model

To create relational models in Cognos Cube Designer, you must import metadata and define the required objects. Query subjects are the basic objects in a relational model.

You can import metadata only from a Content Manager data source. You must perform a separate import for each schema that you want to use. A separate file is created for each data source from which you import metadata. The metadata, one file per schema, is cached on your computer in the *cognos_analytics_location\data* directory to improve performance.

For more information, see [Chapter 5, “Getting started with Cognos Cube Designer,” on page 37.](#)

Before you begin

Check the following prerequisites:

- The data source connection to the database uses a Java Database Connectivity (JDBC) driver. This is required by dynamic query mode.
- The data source is defined in the administration component of IBM Cognos Analytics. If a data source does not exist, you must first create it. For more information, see the *IBM Cognos Analytics Administration and Security Guide*.

Procedure

1. Start Cognos Cube Designer and select one of the following options from the Welcome page:
 - **Create New from Metadata** to import metadata into a new project.
 - **Create New Blank Project** to create a project.
2. From the toolbar, click **Get Metadata > Browse Content Manager Datasource**.
3. Select the database schema from which to import data, and then click **OK**. Repeat this step for each schema that you want to import.

The imported metadata is shown as a list of database tables in the **Source** explorer tree. If your project contains more than one imported data source, each data source is shown in a separate panel. To view the contents of the data source, expand it.

4. If you want to start building your model now, the first task is to define a query subject. For more information, see [“Defining query subjects” on page 155.](#)
5. To save your project, from the **File** menu, click **Save as**. The project is saved as .fmd file.

What to do next

Continue defining the relational objects that include query subjects, query items, query item sets, determinants, and relationships.

You can also add filters and calculations to a relational model.

Defining query subjects

A query subject is a set of query items that have an inherent relationship. By using query subjects, you can create a more abstract, business-oriented view of a data source for your report authors and consumers. The concept of a query subject is fundamental to relational modeling in IBM Cognos software.

In Cognos Cube Designer, you first create a query subject and then add query items from your data source to it. When you make changes to the underlying database structure, the query subjects remain static from a reporting perspective.

You can modify query subjects to optimize and customize the data that they retrieve by adding filters, determinants, or relationships.

Procedure

1. In **Project Explorer**, right-click a namespace, and click **New > Query Subject**.

A new query subject is added to the namespace under a working name **New Query Subject**.


2. Rename the query subject as required, and double-click it to open the editor.

Tip: You can also rename the query subject later. Renaming it in one view, automatically changes the query subject name in all other views.

The query subject editor contains the following tabs: **Editor**, **Filters**, **Determinants**, **Implementation**, and **Relationships**.

3. On the **Editor** tab, add query items to the query subject by using one of the following methods:

- Drag the selected items from the tables in the **Source** explorer to the **Editor** tab.
- In the **Source** explorer, right-click a table or a table column, and click **Drop on > Attributes**.

- Click the **New Query Item**  icon in the query subject editor. Click the query item on the **Editor** tab, and on the **Properties** tab, click in the value field for the query item **Expression** property to open the expression editor. Define the expression. To include another query item in the expression, right-click the item in **Project Explorer**, and click **Drop on > Expression Editor**. For more information, see [“Calculations” on page 169](#).

- In **Project Explorer**, right-click the query subject, and click **New > Query Item**. The new query item is added to the query subject in the **Project Explorer** tree. Double-click the query item to open the expression editor, and define the expression.

You can add columns from multiple tables to the query subject.

4. In the properties pane, specify the properties for the query items as required. For more information, see [“Query items” on page 156](#).
5. On the **Determinants** tab, define the determinants as required. For more information, see [“Determinants” on page 158](#).
6. On the **Implementation** tab, define the relationships between tables in the query subject. For more information, see [“Defining table joins for a query subject” on page 162](#).
7. On the **Relationships** tab, define the relationships with the corresponding query subjects. For more information, see [“Relationships” on page 160](#).
8. Add filters or calculations as required. For more information, see [“Filters” on page 168](#) and [“Calculations” on page 169](#).
9. To see how the query subject data will be displayed to the report authors, right-click the query subject in **Project Explorer**, and select **View Data**.

10. To validate the query subject and resolve potential issues, right-click the query subject in **Project Explorer**, and select **Validate**. For more information, see [“Validate a project and individual objects”](#) on page 44.

Query items

A query item is the smallest object in a relational model that can be placed in a report and a basic building block of a query subject.

Because reports can contain different query items from one or more objects in the model, query item properties control many aspects of the final report. You can modify the properties for individual query items only.

You can specify the following properties for query items:

Name

Specifies the name of the query item. You can rename the query item here.

Description

Specifies a description of the query item.

Expression

This property is used to create embedded calculations that provide the users with calculated values that they regularly use.

Column name

Specifies the name of the column in the database table.

Visible

Specifies whether the query item should be visible to the report authors in the IBM Cognos studios. The values are true or false.

Data type

Specifies the data type for the query item. This property is set in the data source and can only be viewed in Cube Designer.

Precision

Specifies the total number of digits. This property is set in the data source and can only be viewed in Cube Designer.

Scale

Specifies how many digits are represented in the scale. For example, you can show numbers in thousands so that 100,000 means 100,000,000. This property is set in the data source and can only be viewed in Cube Designer.

Regular Aggregate

Specifies the type of aggregation that is associated with the query item in a published package. The property can have values of **Automatic**, **Average**, **Calculated**, **Count**, **Count Distinct**, **Count Non-Zero**, **Custom**, **Maximum**, **Median**, **Minimum**, **Sum**, **Unsupported**, or **Variance**. The default is value is **Unsupported**. For more information, see [“The Regular Aggregate property”](#) on page 157.

Usage

Specifies the intended use for the data represented by the query item. The property can have values of **Identifier**, **Fact**, **Attribute**. For more information, see [“The Usage property”](#) on page 157.

You can rename a query item in the model tree in **Project Explorer**, on the **Editor** tab of the query subject editor, or on the **Properties** tab. Renaming the query item in one place updates all references to this query item in the model.

You create query items within query subjects. For more information, see [“Defining query subjects”](#) on page 155.

The Regular Aggregate property

The **Regular Aggregate** property identifies the type of aggregation for the query item when you publish the query item. Users can either use this default setting to perform calculations on groups of data, or apply a different type of aggregation.

For example, if the **Regular Aggregate** property value for the Quantity query item is sum, and the query items are grouped by Product Name in a report, the Quantity column in the report shows the total quantity of each product.

When modifying this property, you must understand what the data represents to know which aggregate value is required. For example, if you aggregate a part number, the only aggregate values that apply are count, count distinct, maximum, and minimum.

For related information, see [“Regular aggregates” on page 27](#).

The Usage property

The **Usage** property identifies the intended use for the data represented by each query item. During the metadata import, this property is set according to the type of data that the query items represent in the data source.

You need to verify that this property is set correctly. For example, if you import a numeric column that participates in a relationship, this property is set to identifier. You can change the property.

For relational query items, the value of the **Usage** property depends on the type of database object that the query item is based on. You can specify the following values for this property:

Identifier

Database object: key, index, date, datetime

Represents a column that is used to group or summarize the data in a fact column with which the column has a relationship. It also represents an indexed column, and a column that is based on date or time.

Fact

Database object: numeric, timeinterval

Represents a column that contains numeric data that can be grouped or summarized, such as Product Cost.

Attribute

Database object: string

Represents a column, such as Description, that is neither an Identifier nor a Fact.

Unknown

The value is not specified.

Defining query item sets

A query item set represents a business-oriented collection of query items.

Query item sets can contain query items from different query subjects. The query item sets can be included in a package and made available to the report authors in the IBM Cognos studios.

For example, you can create query item sets for different reports, and in each query item set include only those query items that are needed for a specific report. The report authors use the query item sets in the Cognos studios to quickly find all the query items that they need for their reports.


Procedure

1. Right-click a namespace in **Project Explorer**, and click **New > Query Item Set**.
A new query item set is added to the namespace under a working name **New Query Item Set**.
2. Rename the query item set as required, and double-click it to open the editor.

Tip: You can also rename the query item set later. Renaming it in one view, automatically changes the name in all other views.

3. On the **Editor** tab, add query items to the query item set by using one of the following methods:

- Drag the selected query items from the query subjects in **Project Explorer** to the **Editor** tab.

- Click the **New Query Item**  icon, and in the expression editor that is displayed specify the expression for the query item that you want to add to the query item set.

You can include query items from multiple query subjects.

4. To validate the query item set and resolve potential issues, right-click the query item set in **Project Explorer**, and click **Validate**. For more information, see [“Validate a project and individual objects” on page 44](#).

Determinants

You use determinants to control the SQL that provides the granularity of query subjects. Determinants are most closely related to the concept of keys and indexes in a data source. By adding determinants, you can represent groups of repeated data that are relevant for your application. You can also override the index and key information in your data source, replacing it with information that is better aligned with your reporting and analysis needs.

Note: You cannot use determinants with user-defined SQL that is part of a query defined in IBM Cognos Analytics - Reporting.

An example of a unique determinant is Day in the Time example below. An example of a non-unique determinant is Month; the key in Month is repeated for the number of days in a particular month.

When you define a non-unique determinant, you should specify **Group By**. This indicates to IBM Cognos software that when the keys or attributes associated with that determinant are repeated in the data, it should apply aggregate functions and grouping to avoid double-counting. It is not recommended that you specify determinants that have both **Uniquely Identified** and **Group By** selected or have neither selected.

| Year Key | Month Key | Month Name | Day Key | Day Name |
|----------|-----------|------------|----------|-------------------------|
| 2006 | 200601 | January 06 | 20060101 | Sunday, January 1, 2006 |
| 2006 | 200601 | January 06 | 20060102 | Monday, January 2, 2006 |

You can define three determinants for this data set as follows - two **Group By** determinants (Year and Month) and one unique determinant (Day). The concept is similar but not identical to the concept of levels and hierarchies.

| Name of the Determinant | Key | Attributes | Uniquely Identified | Group By |
|-------------------------|-----------|---|---------------------|----------|
| Year | Year Key | None | No | Yes |
| Month | Month Key | Month Name | No | Yes |
| Day | Day Key | Day Name Month Key Month Name Year Key | Yes | No |

In this case, we use only one key for each determinant because each key contains enough information to identify a group within the data. Often Month is a challenge if the key does not contain enough information

to clarify which year the month belongs to. If the Month key cannot uniquely identify the month to a specific year, then include the Year key in the key definition for the Month determinant.

Note: While you can create a determinant that groups months without the context of years, this is a less common choice for reporting because all data for February of all years would be grouped together instead of all data for February 2006 being grouped together.

When to use determinants

While determinants can be used to solve a variety of problems related to data granularity, you should always use them in the following primary cases:

- A query subject that behaves as a dimension has multiple levels of granularity and will be joined on different sets of keys to fact data.

For example, Time has multiple levels, and it is joined to Inventory on the Month Key and to Sales on the Day Key.

- There is a need to count or perform other aggregate functions on a key or attribute that is repeated.

For example, Time has a Month Key and an attribute, Days in the month, that is repeated for each day. If you want to use Days in the month in a report, you do not want the sum of Days in the month for each day in the month. Instead, you want the unique value of Days in the month for the chosen Month Key. In SQL, that is `XMIN(Days in the month for Month_Key)`. There is also a `Group by` clause in the Cognos SQL.

- To avoid double-counting when a measure comes from a dimension table (the measure is on the 1-side of a 1-N join), or when a measure comes from a denormalized fact table.

Aggregating repeating values is referred to as "double-counting" values. To avoid double-counting, the query planning code can generate a specific SQL pattern based on the determinants that are specified in the query subject where the measure is defined.

The key section of the determinants defines the columns which uniquely identify a row of data.

Selecting the **Group By** property on the determinant instructs the query planning process to generate SQL that will not repeat the values for the measure, and therefore aggregate only non-repeating values.

There are less common cases when you need to use determinants:

- You want to uniquely identify the row of data when retrieving text BLOB data from the data source.

Querying blobs requires additional key or index type information. If this information is not present in the data source, you can add it using determinants. Override the determinants imported from the data source that conflict with relationships created for reporting.

You cannot use multiple-segment keys when the query subject accesses blob data. With summary queries, blob data must be retrieved separately from the summary portion of the query. To do this, you need a key that uniquely identifies the row and the key must not have multiple segments.

- A join is specified that uses fewer keys than a unique determinant that is specified for a query subject.

If your join is built on a subset of the columns that are referenced by the keys of a unique determinant on the 0..1 or 1..1 side of the relationships, there will be a conflict. Resolve this conflict by modifying the relationship to fully agree with the determinant or by modifying the determinant to support the relationship.

- You want to override the determinants imported from the data source that conflict with relationships created for reporting.

For example, there are determinants on two query subjects for multiple columns but the relationship between the query subjects uses only a subset of these columns. Modify the determinant information of the query subject if it is not appropriate to use the additional columns in the relationship.

If a query subject contains query items from more than one table, typically you would only add determinants to the table that is joined to the fact tables. This usually applies to tables where the data is denormalized and has different levels of granularity. If the schema is a snowflake (normalized data), for example one-to-many relationships from Product Line to Product Type to Product, then determinants


are not required. In this situation, IBM Cognos Analytics uses the cardinality of the relationships to understand the granularity of the items.

Adding a determinant

You add determinants to control the SQL that provides the granularity of a query subject.

Where a query subject contains determinants, each query item must be included in at least one of the determinants.

Procedure



1. From the **Project Explorer** tree, select the query subject to which you want to add a determinant.
2. Select the **Determinants** tab.
3. From the **Table** drop-down list in the left pane, select the table for which to add a determinant.
4. In the right pane, click **Add Determinant** .
5. Select the determinant, and select the following options as required:

- To specify that the determinant should be used as the unique identifier, select the **Uniquely Identified** check box.

You should select this option only if the data in this item is unique for every row in the underlying data source.

You can specify more than one unique determinant if they are truly unique. At query time, the relationship determines which unique determinant to apply.

Tip: When you select this option, Cognos Cube Designer automatically adds all query items that are not defined as part the key to the **Attributes** box in the lower pane. If some of these query items should be defined as the key, you can move them to the **Key** box.

- Select the **Group By** check box to indicate that when keys or attributes associated with the determinant are repeated in the data, IBM Cognos Analytics should apply aggregate functions and grouping to avoid double-counting.
6. To define a key, select the query items from the **Column** box in the left pane, then click **Add Key**  in the lower pane.
 7. To identify the query items to associate with the determinant, select the query items from the **Column** box in the left pane, then click **Add Attribute**  in the lower pane.

Defining attributes is optional. The query engine uses a determinant with no attributes to indicate which query items are indexed.

8. Repeat steps 4 to 7 to add further determinants if required.
9. To change the order of the determinants, use the arrow buttons.

Determinants are processed in the order in which they are specified. If a query subject contains more than one determinant, the first one that covers all the requested items is used. Determinants are evaluated in the context of each required join as well as the context of requested items.

Relationships

A relationship describes how to create a relational query for multiple objects in the model. Without relationships, these objects are isolated sets of data.

Relationships work in both directions. You often must examine both directions to fully understand the relationship.

Cognos Cube Designer supports the following types of relationships:

- One-to-one

One-to-one relationships occur when one instance of data in a query subject relates to exactly one instance of another. For example, each student has one student number.

- One-to-many or zero-to-many

One-to-many or zero-to-many relationships occur when one instance of data in a query subject relates to many instances of another. For example, each teacher has many students.

- Many-to-many

Many-to-many relationships occur when many instances of data in a query subject relate to many instances of another. For example, many students have many teachers.

When importing metadata, IBM Cognos Cube Designer creates relationships between objects in the model based on the primary and foreign keys in the data source. You can create or remove relationships in the model so that the model better represents the logical structure of your business.

After you import metadata, verify that the relationships that you need exist in the project and that the cardinality is set correctly. The data source might be designed without using referential integrity. Often, many primary and unique key constraints are not specified. Without these constraints, the necessary relationships between fact tables and dimension tables cannot be generated.

Cardinality

Relationships exist between two query subjects or between tables within a query subject. The cardinality of a relationship is the number of related rows for each of the two objects in the relationship. The rows are related by the expression of the relationship; this expression usually refers to the primary and foreign keys of the underlying tables.

IBM Cognos software uses the cardinality of a relationship in the following ways:

- To avoid double-counting fact data.
- To optimize access to the underlying data source system.
- To identify query subjects that behave as facts or dimensions.

You must ensure that all relationships and cardinality correctly reflect your users' reporting requirements.

Notation

By default, Cognos Cube Designer uses Merise notation. Merise notation marks each end of the relationship with the minimum and maximum cardinality of that end.

When you interpret cardinality, you must consider the notation that displays at both ends of the relationship. Possible end labels are shown in the following list:

- 0..1 (zero or one match)
- 1..1 (exactly one match)
- 0..n (zero or more matches)
- 1..n (one or more matches)

The first part of the notation specifies the type of join for this relationship:

- An inner join (1)
- An outer join (0)

An inner join shows all matching rows from both objects. An outer join can be qualified as full, left, or right. Left and right outer joins take everything from the left or right side of the relationship respectively and only what matches from the other side.

Your users see a different report depending on whether you use an inner or outer join. For example, your users want a report that lists salespeople and orders. If you use an outer join to connect salespeople and

orders, the report shows all salespeople, regardless of whether they have any orders. If you use an inner join, the report shows only the salespeople who have placed orders.

Data in one object might have no match in the other object. However, if the relationship has a minimum cardinality of 1, an inner join is always used. Conversely, if all the items match, but the relationship in the model has a minimum cardinality of 0, an outer join is always used, although the results are the same with an inner join. For example, the underlying table for one object contains a mandatory (non-NULLable) foreign key for the other. Ensure that the data and cardinalities match.

The second part of the notation defines the relationship of query items between the objects.

Cardinality in generated queries

IBM Cognos software supports both minimum-maximum cardinality and optional cardinality.

In 0:1, 0 is the minimum cardinality, and 1 is the maximum cardinality.

In 1:n, 1 is the minimum cardinality, and n is the maximum cardinality.

A relationship with cardinality specified as 1:1 to 1:n is commonly referred to as 1 to n when focusing on the maximum cardinalities.

A minimum cardinality of 0 indicates that the relationship is optional. You specify a minimum cardinality of 0 if you want the query to retain the information on the other side of the relationship in the absence of a match. For example, a relationship between customer and actual sales might be specified as 1:1 to 0:n. This indicates that reports will show the requested customer information even though there might not be any sales data present.

Therefore, a 1 to n relationship can also be specified as shown in the following list:

- 0:1 to 0:n
- 0:1 to 1:n
- 1:1 to 0:n
- 1:1 to 1:n

It is important to ensure that the cardinality is correctly captured in the model because it determines the detection of fact query subjects and it is used to avoid double-counting factual data.

When generating queries, IBM Cognos software follows these basic rules to apply cardinality:

- Cardinality is applied in the context of a query.
- 1 to n cardinality implies fact data on the n side and implies dimension data on the 1 side.
- A query subject may behave as a fact query subject or as a dimensional query subject, depending on the relationships that are required to answer a particular query.

Defining table joins for a query subject

You can join logically related tables in a query subject so that the model properly represents the logical structure of your business.

About this task

When a query subject contains query items from multiple tables, Cognos Cube Designer automatically creates relationships, also referred to as joins, between the tables. The relationships are based on the primary and foreign keys in the data source, or on identically-named columns if there are no primary and foreign keys. You can change or remove these relationships, or create new ones.

You can create only one join between two tables.


Procedure

1. In **Project Explorer**, double-click the query subject for which you want to define the table relationships.
2. In the query subject editor, click the **Implementation** tab.

This tab shows a diagram of the tables and joins between them in the query subject.

3. Right-click anywhere in the diagram and use the available menu options to change the level of detail shown in the diagram, select a different view, or change the table layout. You can also use the slider in the toolbar to change the level of detail shown in the diagram.
4. To view or edit a specific join, double-click the line that represents the join. In the **Edit join** window that is displayed you can see the current definition of the join. You can change, delete, or add a new join definition here.



5. To create a new join, click the **Create Join** icon in the toolbar. In the window that is displayed, specify the relationships between the tables, and then click the **Add to Join Expression**  icon to define the relationships between the columns in the tables.
6. On the **Issues** tab, view and resolve potential issues in the diagram.

Defining relationships between query subjects

To join logically-related query subjects that your users want to combine in a single report, you create a relationship between the query subjects.

This is useful when objects were not selected during metadata import, were not joined in the data source, or are from multiple data sources.

Procedure

1. In **Project Explorer**, double-click the query subject for which you want to define a relationship.
2. In the query subject editor, click the **Relationships** tab.
3. Click the **New Relationship** icon.
4. In the **Add Relationships dialog**, select the query subject that you would like to join with the query subject selected in step 1, and click **OK**.
5. Click the **Edit** button for the query subject that you added.
The relationship editor is displayed.
6. In the relationship editor, click the **Add** button to add a new row of query items. Then, click the query item in the query subject on one side, and select the matching query item in the query subject on the opposite side. Repeat this action for all query items that you want to match.
7. Specify the cardinality between the matching query items, as documented in the [“Cardinality”](#) on page 161 topic.
8. Click a different object in **Project Explorer** to exit the relationship editor.

Creating a DMR model

To create a DMR model in IBM Cognos Cube Designer, you must import metadata and define the required objects.

You can import metadata only from a Content Manager data source. You must perform a separate import for each schema that you want to use. A separate file is created for each data source from which you import metadata. These files are stored in the *cognos_analytics_location\data* directory to improve performance.

For more information, see [Chapter 5, “Getting started with Cognos Cube Designer,”](#) on page 37.

Before you begin

Check the following prerequisites:

- The data source connection to the database uses a Java Database Connectivity (JDBC) driver. This is required by dynamic query mode.
- The data source is defined in the administration component of IBM Cognos Analytics. If a data source does not exist, you must first create it. For more information, see the *IBM Cognos Analytics Administration and Security Guide*.

Procedure

1. Start Cognos Cube Designer and select one of the following options from the Welcome page:
 - **Create New from Metadata** to import metadata into a new project.
 - **Create New Blank Project** to create a project.
2. From the toolbar, click **Get Metadata > Browse Content Manager Datasource**.
3. Select the database schema from which to import data, and then click **OK**. Repeat this step for each schema that you want to import.

The imported metadata is shown as a list of database tables in the **Source** explorer tree. If your project contains more than one imported data source, each data source is shown in a separate panel. To view the contents of the data source, expand it.

4. From the File menu, click **Save as** to save the project as a .fmd file.

What to do next

You must now define the dimensions and measure to include in the model. For more information, see [“Defining a dimension” on page 164](#) and [“Defining a measure” on page 166](#).

You can also add filters and calculations to a DMR model.

Dimensions

You can add regular dimensions to a DMR model.

A regular dimension is a collection of hierarchies and levels that describes one aspect of a measure, such as Customer or Product. For more information about hierarchies and levels, see [“Dimensional metadata” on page 15](#).

Important: Parent-child dimensions, relative time dimensions, and padding members are not supported for DMR modeling.


When you have added the dimensions you require, you must identify the relationships between the dimensions and measures. For more information, see [“Relationships between dimensions and measure dimensions” on page 167](#).

Defining a dimension

Using IBM Cognos Cube Designer, you can define a dimension manually or you can generate a dimension based on a table in your relational database. When you validate the dimension, you can use information from the **Issues** tab to help you complete the dimension definition.

Procedure

1. Select **Model** from the **Project Explorer** tree.
 - To create a new dimension based on a relational table, in **Data Source Explorer** tree, right-click the dimension table that you want to add to the model, and click **Generate > Dimension using data sampling**.

- To create a new dimension manually, click **New Dimension** . The dimension contains a set of initial objects you can use to complete the dimension.
- To access the dimension editor, right-click a dimension from the **Project Explorer** tree and select **Open Editor**.

Tip: Use folders and namespaces to organize objects. Using folders and namespaces makes it easier for you to locate objects and view the structure of a project in the **Project Explorer**.

2. Set the **Default Hierarchy**, and complete the dimension definition using the **Properties** tab.

For more information about dimension properties, see [“Model dimensions” on page 45](#).

What to do next


To complete the dimension, you must complete the definition of each hierarchy and level that belongs to the dimension.

Tip: Right-click a relational table and select **Explore Metadata**. You can use the **Relational Explorer Diagram** to help you understand the structure of the metadata used to design the hierarchies and levels.

Defining a hierarchy

In IBM Cognos Cube Designer, a single level-based hierarchy is automatically added when you create a dimension. You can also create multiple level-based hierarchies in a dimension.

Procedure

1. From the **Project Explorer** tree, select the dimension you want to work with.
 - To create a new hierarchy, click **New Hierarchy** .
 - To access the hierarchy editor, right-click a hierarchy that belongs to the dimension and select **Open Editor**.
2. Complete or modify the hierarchy definition using the **Properties** tab. Identify the **Default Member** and **Root Caption** if required.


For more information about hierarchy properties, see [“Model hierarchies” on page 48](#).
3. If an **All** level is not required, set the **Multiple Root Members** property to **true**.
4. To add levels to the hierarchy, drag levels from the **Levels** folder to the hierarchy.


Defining a level

In IBM Cognos Cube Designer, you define levels to model the relationships in a hierarchy.

For each level, you assign or create attributes, map them to the relational data source, identify level keys and, optionally, define a sort order. You can also hide attributes in the published package if required.

Procedure

1. From the **Project Explorer** tree, select the dimension you want to work with.
 - To create a new level, click **New level** .
 - To access the level editor, right-click the level in the **Project Explorer** tree, and select **Open Editor**.
2. Complete or modify the level definition using the **Properties** tab.

For more information about level properties, see [“Model levels” on page 49](#).
3. To create an attribute, click **New Attribute** .


4. To map a table column to the new attribute, select the required column from the **Data Source Explorer** tree and drop it onto the **Mapping** column.

Tip: You can also create attributes by dropping table columns to the **Attribute** column.

5. Select the attributes assigned to **Member Caption** and, if required, **Member Description**.

For more information about these special attributes, see [“Attributes” on page 24](#).

6. You can define the **Level Unique Key** one of two ways:

- If the level unique key is a single attribute, select the **Level Unique Key** check box for the attribute.
- If the level unique key is a composite key, click **Level Key** . For more information, see [“Defining a level unique key” on page 51](#).

7. If required, specify the member sort order. For more information, see [“Defining the member sort order” on page 52](#).

8. To hide an attribute in the published package, change the **Visible** property to false.

9. To assign the level to a hierarchy, select the level and drop it onto the hierarchy in the **Project Explorer** tree.

Tip: You can also assign levels by dropping them into the hierarchy editor.

10. Expand the hierarchy in the **Project Explorer** tree and, if necessary, modify the order of the levels as they appear under the hierarchy.

Measure dimensions and measures

You can add regular measures to a measure dimension in a DMR model.

A measure dimension is a container for a set of measures. For more information about measures, see [“Measures” on page 26](#).

Important: Calculated measures are not supported for DMR modeling.



When you have added the measures you require, you must identify the relationships between the dimensions and measures. For more information, see [“Relationships between dimensions and measure dimensions” on page 167](#).

Defining a measure

Using IBM Cognos Cube Designer, you can define a measure by using one of the following methods:

- Generate a measure based on a column in a relational database. The mapping to the associated column is automatically created.
- Manually define a measure by creating a mapping to a database column or to an expression.

Procedure

1. Select **Model** from the **Project Explorer** tree.
2. Click **New Measure Dimension**  to create a container for the measures.
3. Right-click the measure from the **Project Explorer** tree and select **Open Editor**.
 - To create a new measure based on a column in a relational table, from the **Data Source Explorer** tree, drop the column onto the **Editor** pane.
 - To create a new measure manually, click **New Measure**  to add a blank measure. You can complete the measure one of two ways:
 - To map the measure to a table column, drag a table column from the **Data Source Explorer** onto the **Mapping** field.

- To map the measure to an expression, define an expression in **Expression** property on the **Properties** pane.
4. Complete the measure definition using the **Properties** tab.

For more information about measure properties, see [“Model measures” on page 63](#).

Relationships between dimensions and measure dimensions

For DMR modeling, you can define joins and edit the scope relationship for the dimensions and measure dimensions in a model.

Joins

You can define a join between a dimension and a measure dimension using common key(s) in the underlying relational tables. If the join is at a higher grain than the lowest level of the dimension, you must ensure the **Join is at the lowest level of detail for the dimension** option is not selected. This ensures that measures that roll up to the specified level are not double counted.

A join combines columns from two relational tables using an operator to compare the columns. A join uses attributes that reference columns in the tables being joined. The simplest form of a join uses two attributes: one that maps to a column in the first table and one that maps to a column in the second table. You also specify an operator to indicate how the columns are compared. For example, "Time ID = time_id". A join can also model composite joins where two or more columns from the first table are joined to the same number of columns in the second table. A composite join uses pairs of attributes to map corresponding columns together. Each pair of attributes has an operator that indicates how that pair of columns is compared. For example, "Customer Number = customer_number AND Store Number = store_number".

For information on defining a join, see [“Defining a join between a dimension and measure dimension” on page 167](#).

Scope relationship

You define a scope relationship between a dimension and measure dimension to identify the level at which the measures are available for reporting. A scope relationship is not the same as a join and does not affect the Where clause. There are no conditions or criteria set in a scope relationship to govern how a query is formed, it specifies only if a dimension can be queried with a particular fact.


When you create a measure dimension, IBM Cognos Dynamic Cubes creates a scope relationship between the each measure and dimension. The scope is automatically set to the lowest level in the dimension for each measure in the measure dimension. If data is reported at a different level for the measures, you can set the scope for a measure. You can also specify the lowest level at which data can be reported.

For information on defining a scope relationship, see [“Defining a scope relationship” on page 168](#).

Defining a join between a dimension and measure dimension

You can define a join between a dimension and a measure dimension when the level of a join does not match the level of the fact table. You must define the correct join to avoid double counting data from the fact table.

Procedure

1. From the **Project Explorer** tree, select the dimension or measure dimension for which you want to define a join.
2. Select the **Relationships** tab.
3. Click **New Relationship** , select the dimensions or measure dimensions that you want to join, then click **OK**.

4. For each dimension or measure dimension, click **Edit**, and select the **Joins** tab.
5. Specify the join by relating columns in the dimension to columns in the measure dimension.
6. Specify the relationship operator.
7. If the join is at a higher grain than the lowest level of the dimension, clear the **Join is at the lowest level of detail for the dimension** check box.

Important: Note: IBM Cognos Cube Designer cannot automatically detect that a join is at a higher grain than the lowest level of a dimension.


Defining a scope relationship

If the scope relationship created by IBM Cognos Dynamic Cubes between each measure and dimension is not correct, you can edit it.

Before you begin

You must create a join relationship before you can set a scope relationship. For more information, see [“Defining a join between a dimension and measure dimension”](#) on page 167.

Procedure

1. From the **Project Explorer** tree, select the dimension or measure dimension for which you want to set the scope relationship.
2. Select the **Relationships** tab.
3. For each dimension or measure dimension, click **Edit**, and select the **Scope Relationship** tab.
4. Select the hierarchy level at which you want to set the scope.
5. Select the measure for which you are setting the scope, and then click **Set Scope** .

Filters

A filter is an expression that specifies the conditions that rows must meet to be retrieved for a dimension, query subject, calculation, or report to which the filter is applied. Cognos Cube Designer supports stand-alone and embedded filters.

A filter returns a boolean value so that you can limit the rows returned by a dimension or query subject.

For example, you can use the `in_range` function to create a filter that retrieves data for products introduced in a specific time frame. The syntax for this example looks like this:

```
[gosales_goretailers].[Products].[Introduction date] in_range  
{Feb 14, 2002 : July 14, 2010}
```

Note: When using a date or time function, you must use a 24-hour clock. For example, use 20:00 to signify 8 p.m.

You can restrict the data represented by dimensions or query subjects in a project by creating a security filter for these objects. The security filter controls the data that your users can see when they set up their reports. This filter can be used in a security view.

To restrict the data that the queries in a package retrieve, you can use governors.

Defining a stand-alone filter

Create a stand-alone filter when you want to reuse the filter expression. You can include a stand-alone filter in a package to make the filter available to users.

Procedure

1. In **Project Explorer**, right-click the namespace or folder where you want to define the filter, and click **New > Filter**.

The filter is created in the specified location. You can rename the filter now or later.

2. Double-click the filter to open its expression editor.
3. Specify the filter expression on the **Expression** tab.

If the expression is complex, you can use an external editor to edit the expression and then copy it to the **Specification** tab.

4. Right-click the filter name in **Project Explorer**, and click **Validate**. Resolve possible errors in the expression. For more information, see [“Validate a project and individual objects” on page 44](#).

Defining an embedded filter

Create an embedded filter when you want to apply it to only one dimension or query subject.

Embedded filters that have the **Usage** property set to **Security** are used to secure data in the model. These filters are used with security views. For more information, see [“Securing packages” on page 177](#).

Procedure

1. In **Project Explorer**, right-click the dimension or query subject for which you want to define the filter, and click **Open Editor**.



2. On the **Filters** tab, click the **New Filter** icon.

The filter appears in the project window. You can rename the filter now or later.

3. Specify the **Usage** property for the filter.

- When you select **Always**, the filter is applied to all objects in the dimension or query subject.
- When you select **Security**, the filter is used to define data security.

4. In the project pane, double-click the filter to open the expression editor and specify the filter expression.

If the expression is complex, you can use an external editor to edit the expression and then copy it to the **Specification** tab.

5. On the **Issues** tab, check for possible errors in the expression, and resolve the errors.

Calculations

You can create calculations to provide your users with calculated values that they regularly use. Calculations can use query items, parameters, variables, expressions, and expression components, such as functions.

Punctuation characters, such as the question mark (?), must be in 7-bit ASCII character code. If you type a punctuation character from a multi-byte enabled keyboard, ensure that you type the 7-bit ASCII representation of the character. For example, type Alt+063 for the question mark.

Avoid using characters that are used for expression operators in the name of the calculation. Syntax errors may occur when the expression is evaluated. For example, a calculation named Margin * 10 causes errors when used in an expression such as [Margin * 10] < 20.

In expressions, an operator or function may require operands to be of a particular dimensional type. When an operand is not of the required type, one or more coercion rules may be applied to coerce the operand to the appropriate type. Because coercion rules are not applied to expressions in query subjects, ensure that those expressions are valid without relying on coercion rules. For more information about coercion rules, see the *IBM Cognos Analytics - Reporting User Guide*.

You can create the following types of calculations:

- Stand-alone calculations

Use a stand-alone calculation when you want to reuse an expression. You can include the calculation in a package to make it available to the users. For more information, see [“Defining a stand-alone calculation” on page 170](#).

- Embedded calculations

Use an embedded calculation when you want to use a calculation with only one dimension or query subject. You can create an embedded calculation when modifying a query subject (for more information, see [“Defining query subjects” on page 155](#)), or a dimension (for more information, see [“Defining a dimension” on page 164](#)).

Defining a stand-alone calculation

Define a stand-alone calculation when you want to reuse an expression. You can include the calculation in a package to make it available to the users.

A stand-alone calculation can reference an embedded calculation.

Procedure

1. Right-click a namespace in **Project Explorer**, and click **New > Calculation**.
The calculation is added to the namespace under a working name **New Calculation**.
2. Rename the calculation as required, and double-click it to open the editor.
3. Define the expression for the calculation. To include a query item or an attribute in the expression, right-click the item in **Project Explorer**, and click **Drop on > Expression Editor**.
4. To validate the calculation and resolve potential issues, right-click the calculation in **Project Explorer**, and click **Validate**. For more information, see [“Validate a project and individual objects” on page 44](#).
5. Click a different object in **Project Explorer** to exit the calculation editor.

Creating and publishing packages

You publish a package to make the relational and DMR metadata available to the report authors in IBM Cognos reports and dashboards. Packages must contain all the information that a specific user or group of users needs to create reports.

When creating a package, you can set governors to restrict the data that the queries in a package retrieve, and apply security by using security views.

Before you begin

Validate the model and resolve any issues.

About this task

The selectable items in a package can include: query subjects, dimensions, query item sets, filters, calculations, and parameter maps.

Procedure

1. In **Project Explorer**, right-click the **Packages** folder, and click **New > Package**.
A new package is added in the **Packages** folder under a working name **New Package**.
2. Rename the package as required, and double-click it to open the editor.
Tip: You can also rename the package later. Renaming it in one view, automatically changes the name in all other views.

3. In **Project Explorer**, right-click the object that you want to add to the package, and click **Drop on > Package Editor**. Repeat this step for each object that you want to add to the package. You can also drag-and-drop the selected objects into the package.

The objects that you added appear on the **Editor** tab.

4. Define governors and apply security. For more information, see [“Governors” on page 171](#) and [“Securing packages” on page 177](#).
5. Validate the package and resolve potential problems that are reported on the **Issues** tab. For more information, see [“Validate a project and individual objects” on page 44](#).
6. On the **Properties** tab, browse to the **Publish Location** to which you would like to publish the package. The location is either in **Public Folders** or **My Folders** in IBM Cognos Connection.
7. In **Project Explorer**, right-click the package name, and click **Publish**.

A message about successful package creation should appear.

Results

The package is now available in Cognos Analytics, and can be used by the reporting and dashboarding components.

Governors

You set governors in a package to ensure that the metadata contains the specified limits. Governors also reduce system resource requirements and improve performance. Default governor settings are applied to a package unless you change them. Because governors are set at the package level, it is possible for individual packages to use different governor values.

You can also set governors in IBM Cognos Analytics - Reporting. The governor settings in Cognos Reporting override the governor settings in a package.

Important:

For governors that affect caching, you must enable caching in one of the following ways:

- Enable the **Allow usage of local cache** governor in Cognos Cube Designer.
- Enable the **Use Local Cache** query property for a report in Cognos Reporting.

Maximum number of retrieved rows

You can set data retrieval limits by controlling the number of rows that are returned in a query or report. Rows are counted as they are retrieved.

When you run a report and the data retrieval limit is exceeded, an error message displays and the query or report is shown with no data.

A setting of zero means that no limit is set.

Cross-product joins

You can control whether cross-product joins can be used in a query or report. A cross-product join retrieves data from tables without joins. This type of join can take a long time to retrieve data.

The default value for this governor is **Deny**. Select **Allow** to allow cross-product joins.

SQL join syntax

You can control how SQL is generated for inner joins by selecting one of the following settings:

- If the governor is set to **Server determined**, the IBM Cognos Analytics server determines the behavior at run time.
- The **Implicit** setting uses the where clause.

For example,

```
SELECT publishers.name, publishers.id,
books.title FROM publishers, books WHERE publishers.id
= books.publisher_id ORDER BY publishers.name, books.title;
```

- The **Explicit** setting uses the from clause with the keywords inner join in an on predicate.

For example,

```
SELECT
publishers.name, publishers.id,
books.title FROM publishers INNER JOIN books ON publishers.id
= books.publisher_id ORDER BY publishers.name, books.title;
```

You can set the join type on the query property in Cognos Reporting to override the value of this governor.

Regardless of the setting you use for this governor, the **Explicit** setting is used for left outer joins, right outer joins, and full outer joins.

This governor has no impact on user-defined SQL.

SQL generation for level attributes

You can control the use of the minimum aggregate in SQL generated for attributes of a level (member caption).

If the governor is set to **Server determined**, the IBM Cognos Analytics server determines the behavior at run time.

The **Minimum** setting generates the minimum aggregate for the attribute. This setting ensures data integrity if there is a possibility of duplicate records. For example,

```
select XMIN(Product.Product_line
for Product.Product_line_code) as Product_line, //level attribute
Product.Product_line_code as Product_line_code
from
(...) Product
```

The **Group By** setting adds the attributes of the level in the group by clause with no aggregation for the attribute. The distinct clause indicates a group by on all items in the projection list. The **Group By** setting is used if the data has no duplicate records. It can enhance the use of materialized views and may result in improved performance. For example,

```
select distinct
Product.Product_line as Product_line, //level attribute
,Product.Product_line_code
as Product_line_code
from (...) Product
```

SQL generation for determinant attributes

You can control the use of the minimum aggregate in SQL generated for attributes of a determinant with the group by property enabled.

If the governor is set to **Server determined**, the IBM Cognos Analytics server determines the behavior at run time.

The **Minimum** setting generates the minimum aggregate for the attribute. This setting ensures data integrity if there is a possibility of duplicate records. For example:

```
select PRODUCT_LINE.PRODUCT_LINE_CODE
as Product_line_code,
XMIN(PRODUCT_LINE.PRODUCT_LINE_EN
for PRODUCT_LINE.PRODUCT_LINE_CODE)
as Product_line //attribute
from
great_outdoors_sales..GOSALES.PRODUCT_LINE PRODUCT_LINE
group by
PRODUCT_LINE.PRODUCT_LINE_CODE //key
```

The **Group By** setting adds the attributes of the determinants in the group by clause with no aggregation for the attribute. This setting is used if the data has no duplicate records. It can enhance the use of materialized views and can result in improved performance. For example:

```
select
PRODUCT_LINE.PRODUCT_LINE_CODE as Product_line_code,
PRODUCT_LINE.PRODUCT_LINE_EN as Product_line //attribute
from
great_outdoors_sales..GOSALES.PRODUCT_LINE PRODUCT_LINE
group by
PRODUCT_LINE.PRODUCT_LINE_CODE //key
PRODUCT_LINE.PRODUCT_LINE_EN //attribute
```

SQL parameter syntax

This governor specifies whether generated SQL uses parameter markers or literal values.

If the governor is set to **Server determined**, the IBM Cognos Analytics server determines the behavior at run time.

You can override the value of this governor in Cognos Reporting.

Dynamic SQL applications can prepare statements that include markers in the text that denote that the value is provided later. This is most efficient when the same query is used many times with different values. The technique reduces the number of times a database must hard parse an SQL statement and it increases the reuse of cached statements. However, when queries navigate larger amounts of data with more complex statements, they have a lower chance of matching other queries. In this case, the use of literal values instead of markers can result in improved performance.

Allow usage of local cache

Select this governor to specify that all reports based on this package should use cached data. This governor is enabled by default.

This setting affects all reports that use a package. Use Cognos Reporting if you want a report to use a different setting than the package.

Use WITH clause when generating SQL

If your data source supports the WITH clause, you can use it in a Cognos SQL query.

(DQM) Adjust SQL generation for exact numeric division

This governor controls how calculations with division are adjusted to ensure that the division results contain information that is significant for the reports.

The **Cast to Double** setting converts the calculation as follows:

- $[item1] / [item2]$ becomes $cast([item1] \text{ as double precision}) / [item2]$
- $cast([item1] \text{ as decimal}(9,2)) / [item2]$ becomes $cast(cast([item1] \text{ as decimal}(9,2)) \text{ as double precision}) / [item2]$

The **Cast to Double conditional** setting converts the calculation in the following ways. Use this setting if the numerator is not a cast operation

- $[item1] / [item2]$ becomes $cast([item1] \text{ as double precision}) / [item2]$
- $cast([item1] \text{ as decimal}(9,2)) / [item2]$ becomes $cast([item1] \text{ as double precision}) / [item2]$

The **Do not adjust** setting does not convert the calculation.

The default setting is **Cast to Double**.

(DQM) Cache is sensitive to connection command blocks

This governor specifies whether the key to the cache includes the expanded value of the connection command blocks. If the connection command block evaluates to different values for different users, you likely want the key to the cache to include this information.

For example, you create a connection command block that contains a macro with a reference to the session parameter for a user name. As a result, the expanded value of the command block is different for each user. However, this difference is not significant if the user name is used only for logging. In this case, the cache likely can be shared and you can turn off this governor. However, if the user name controls data retrieval, the cache likely cannot be shared and you must select this governor.

If this governor is selected, the cache is shared only by users that share the version of the expanded connection command blocks used to load data into the cache.

If this governor is not selected, differences in connection command blocks are ignored.

The governor is selected by default.

For more information about using command blocks, see the *IBM Cognos Analytics Administration and Security Guide*.

(DQM) Cache is sensitive to DB info

This governor controls the sensitivity of the cache that is associated with a package that is shared by users of the connection. It also specifies what database information is used to restrict sharing in that cache. The information is originally specified in IBM Content Manager and is provided on the request to the cache.

The **DB + Connection + Signon** setting specifies that the cache is shared only if users specify the same data sources, connection strings, and signon information.

The **DB + Connection** setting specifies that the cache is shared only if users specify the same data sources and connection strings.

The **DB** setting specifies that the cache is shared only if users specify the same data sources.

The **None** setting specifies that none of the data sources, connection strings, or signon information is shared.

The default setting is **DB + Connection + Signon**.

(DQM) Cache is sensitive to model security

This governor controls the security that is used to access the cache.

The **Automatic** setting specifies that the IBM Cognos user and user classes are used to confirm access to all the security filters in the model. The union of the security objects with the model security filters is used to identify the cache.

The **User** setting specifies that the IBM Cognos user identity is used to identify the cache. The cache is reusable for the current user only. No sharing with other users takes place.

The **UserClass** setting specifies that the IBM Cognos user classes are used to identify the cache.

The **None** setting disables checking for model security filters, even if there are such filters in the model.

The default setting is **Automatic**.

(DQM) Local cache policy

Use this governor to control the level of queries for which reusable cursors are created.

The **Lowest summary sub-query** setting specifies that the cache is created only on lowest summary subqueries of the request. This behavior is also the behavior in the compatible query mode.

The **Query referenced by layout** setting specifies that the cache is created only on queries using the dynamic query mode that are referenced by the layout. The cursor that is created in this option does not contain nesting cursors.

The **Explicitly per query** setting specifies that the cache is created on every query that has a local cache that is enabled. The cursor that is created in this option contains nesting cursors if required.

The default setting is **Lowest summary sub-query**.

The **Allow usage of local cache** governor specifies that all reports based on the package use cached data. By default, if the **Allow usage of local cache** governor is enabled, reusable cursors on the lowest summary subqueries are automatically created. However, if a query has query references such as join or union queries, the join or union process is not pushed to the database. If the join or union process can be better handled by the database when the database server has more resources, choose either the **Query referenced by the layout** or **Explicitly per query** setting.

(DQM) Cursor mode

Use this governor to control how long the resources required by a query are retained before they are released.

The query engine loads data from a data source into a data set with a cursor. The cursor can be read completely or partially. As soon as the query engine reads the last record, the result set is complete and the database resources are released.

The **Automatic** setting specifies that the query engine stops reading data after the requested number of records. It leaves the resources active in anticipation of further requests for data. The stopped query retains the database connection and the cursor for future data retrieval requests. Stopped queries are released after a specified amount of idle time. As a result, database resources are released either after all data is rendered or after a certain amount of idle time or maximum age time. During this time, these resources cannot be used by other queries.

The **Query Per Page** setting specifies that the query engine releases resources as soon as the current report page is rendered to the user. Every subsequent page request, including those requests that previously loaded the complete result set, requires that the database connection and the cursor to be reestablished. This setting releases data source resources the fastest, but requires the most use of time and resources to re-execute a query.

The **Load In Background** setting specifies that the query engine returns the requested portion of the data and then starts a background thread to load the rest of the data into a cache. The background thread runs at a lesser priority. Further requests return the data that is loaded by the background thread from the cache. If more data is required before the background thread loads sufficient data, the new

request takes priority. This setting provides a fast first page response and improved response time for subsequent pages. The resources are released as soon as all the data is loaded into the cache. However, more memory is used for the cached data than with the other settings.

The default setting is **Automatic**.

(DQM) Summary query join operator

Use this governor to control the syntax for joining summary queries.

The **Is Not Distinct From** setting specifies that **Is Not Distinct From** should always be used to join.

The **Equal operator** setting specifies that the **Equal operator** should always be used to join. You should not use this setting unless you are certain that a column you are joining contains no null values.

The **Automatic** setting specifies that if a column is nullable, then **Is Not Distinct From** is used to join, otherwise the **Equal operator** is used to join.

The default setting is **Is Not Distinct From**.

(DQM) Multi fact join operator

Use this governor to control the syntax of the full outer join in Cognos SQL that is used to join multi-fact queries.

The **Is Not Distinct From** setting specifies that **Is Not Distinct From** should always be used to join.

The **Equal operator** setting specifies that the **Equal operator** should always be used to join. You should not use this setting unless you are certain that a column you are joining contains no null values.

The **Automatic** setting specifies that if a column is nullable, then **Is Not Distinct From** is used to join, otherwise the **Equal operator** is used to join.

The default setting is **Is Not Distinct From**.

In the following example, the join between FS1 and FS2 is applied to two columns, Item_Code (not nullable), and Customer_Number (nullable). The governor is set to **Is Not Distinct From**.

```
SELECT
  COALESCE(
    FS1.Item_Code,
    FS2.Item_Code) AS Item_Code,
  COALESCE(
    FS1.Customer_Number,
    FS2.Customer_Number) AS Customer_Number,
  FS1.Order_Quantity AS Order_Quantity,
  FS2.Plan_Sales_Quantity AS Plan_Sales_Quantity
FROM
  FS1
  FULL OUTER JOIN FS2
  ON
    FS1.Item_Code IS NOT DISTINCT FROM FS2.Item_Code AND
    (FS1.Customer_Number IS NOT DISTINCT FROM FS2.Customer_Number)
```

The following example shows the same join where the governor is set to **Equal operator**.

```
SELECT
  COALESCE(
    FS1.Item_Code,
    FS2.Item_Code) AS Item_Code,
  COALESCE(
    FS1.Customer_Number,
    FS2.Customer_Number) AS Customer_Number,
  FS1.Order_Quantity AS Order_Quantity,
  FS2.Plan_Sales_Quantity AS Plan_Sales_Quantity
FROM
  FS1
  FULL OUTER JOIN FS2
  ON
```



```
FS1.Item_Code = FS2.Item_Code AND
(FS1.Customer_Number = FS2.Customer_Number)
```

Because `Customer_Number` is nullable, the output can show results where the stitch is not applied properly if `Customer_Number` has null values.

In the following example, the governor is set to **Automatic** for the same join:

```
SELECT
  COALESCE(
    FS1.Item_Code,
    FS2.Item_Code) AS Item_Code,
  COALESCE(
    FS1.Customer_Number,
    FS2.Customer_Number) AS Customer_Number,
  FS1.Order_Quantity AS Order_Quantity,
  FS2.Plan_Sales_Quantity AS Plan_Sales_Quantity
FROM
  FS1
  FULL OUTER JOIN FS2
    ON
      FS1.Item_Code = FS2.Item_Code AND
      (FS1.Customer_Number IS NOT DISTINCT FROM FS2.Customer_Number)
```

In this instance, **Is Not Distinct From** is used only when a column is nullable. This provides correct results and better performance than the default governor setting.


Setting governors

You set governors to reduce system resource requirements and improve the performance of a published package.

You can specify different governor values for different packages.

Procedure

1. From the **Project Explorer** tree, select the package for which you want to set governors.
2. Select the **Governors** tab.
3. Update the properties of each governor as required.

Tip: To reset the properties of all governors to their default values, click **Reset** .

Securing packages

You apply security to packages by defining security views and assigning access permissions for the views.

Before you begin

Security views can include security filters that are used to secure data in the query subjects and dimensions. The security filters must already be defined. For more information, see [“Defining an embedded filter” on page 169](#).

About this task


You can define multiple security views for one package. Each security view should include the objects that are required for certain groups of users or for certain reporting purposes.

After the security view is defined, you use the users, groups and roles from the **Cognos** namespace and from the namespaces that are configured for your Cognos Analytics environment to assign access permissions for the security views.

Procedure

1. In **Project Explorer**, double-click the package name to open the package editor.
2. Click the **Security** tab in the editor.
3. In the **Security Views** section, click the **Add Security View** icon.

A new security view is added under a working name **New Security View**.

4. Rename the security view as required. As a best practice, do not use the package name as the security view name.
5. With the security view selected, click the **Objects** tab in the adjacent section.
6. Select the objects that you would like to include in this security view by clicking the **Grant** button for them. Click the **Deny** button for objects that you want to exclude from the security view.
7. Click the **Data** tab to include existing security filters in the security view. Click the **Add Security Filter**  icon to see the filters, and use the **Grant** and **Deny** buttons to include or exclude each filter.
8. Repeat steps 3 to 7 to define as many security views as you need.
9. After publishing the package to Cognos Analytics, from the **File** menu in Cognos Cube Designer, click **Assign Users, Groups, and Roles Dialog**.
10. In the window that is opened, select the published package in the **Package** field. The **Security View** drop-down list should contain the security views that you created for the package.
11. For each security view on this list, select the users, groups, or roles from the **Directory** list that need to access the view.

Move the entries between the two sections in the window by using the arrow icons.

Tip: The **Directory** list contains the **Cognos** namespace and the namespaces specific to your environment. Click on each namespace to expand the directory structure.

Results

In the Cognos studios, the users can access only the objects and data that are included in the security view for which they have access permissions.

Appendix A. Accessibility features

Accessibility features help users who have a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

The major accessibility features for IBM Cognos Cube Designer are described in the following list. You can

- customize the display to enhance accessibility. For example, you can enable a focus ring that emphasizes the selected element.
- use shortcut keys to navigate and trigger actions.
- apply operating system display settings, such as high-contrast display.

For more information about the commitment that IBM has to accessibility, see the [IBM Accessibility Center](http://www.ibm.com/able) (<http://www.ibm.com/able>).

Accessibility features in Cognos Cube Designer

You can customize the IBM Cognos Cube Designer display to enhance accessibility.

The **View** menu includes the following display controls.

| View menu item | Description |
|-----------------------------|--|
| Show Access Keys | Adds a numeric identifier to each pane. To navigate to a different pane, press Alt+Shift+pane number . The navigation control works when Show Access Keys is disabled. |
| Show Focus Rectangle | Displays a dotted rectangle around the object that has the current keyboard focus. |

Keyboard shortcuts for Cognos Cube Designer

You can use keyboard shortcuts to navigate through and perform some tasks in IBM Cognos Cube Designer.

| Applies to | Description | Keyboard shortcut |
|------------------|---|-------------------|
| General | Perform the default action for an active command button. | Enter or Spacebar |
| General controls | Move forward to the next control at the same level. | Tab |
| General controls | Move backward to the previous control at the same level. | Shift+Tab |
| Check boxes | Toggle a check box from selected to cleared or cleared to selected. Tip: This shortcut also applies to other settings that can have an on or off state. | Spacebar |

Table 56. Keyboard shortcuts for Cognos Cube Designer (continued)

| Applies to | Description | Keyboard shortcut |
|---------------------------------------|--|---------------------------|
| Radio buttons that are not in a group | Move to the next radio button and select it. | Tab |
| Radio button groups | Move to the next radio button in the group and select it. | Right arrow Down arrow |
| Radio button groups | Move to the previous radio button in the group and select it. | Up arrow Left arrow |
| Drop-down lists | Open and display the drop-down list contents. | Alt+Down arrow |
| Drop-down lists | Close an open drop-down list. | Alt+Up arrow |
| Tree controls | Move to the first selectable node below, or, if the node below has child nodes and the node is expanded, move to the first child node. | Down arrow |
| Tree controls | Move to the first selectable node above. | Up arrow |
| Tree controls | Expand the selected node or move to the first selectable child node. | Right arrow |
| Tree controls | Collapse the selected node, move to the parent node, or move to the first selectable node above. | Left arrow |
| Tree controls | Move to the first node in a tree control. | Home |
| Tree controls | Move to the last node in a tree control. | End |
| Menus | Move to the next available menu item. | Down arrow |
| Menus | Move to the previous available menu item. | Up arrow |
| Menus | Expand the child menu items. | Right arrow |
| Menus | Collapse the child menu items. | Left arrow |
| Context menus | Open the context menu for the selected item. | Shift+F10 |

Table 56. Keyboard shortcuts for Cognos Cube Designer (continued)

| Applies to | Description | Keyboard shortcut |
|-------------------|-----------------------------|------------------------------|
| Context menus | Close an open context menu. | Esc |
| Scrolling | Scroll down. | Down arrow Page down |
| Scrolling | Scroll up. | Up arrow Page up |
| Columns | Change the width. | Ctrl+Shift+► Ctrl+Shift+◄ |

Appendix B. Report considerations

There are a number of points to consider when you view report data based on a published dynamic cube.

Calculated members in reports

For most reports, IBM Cognos Dynamic Cubes calculated members are used the same way as regular members. However, because of some different constraints and capabilities, the report user may encounter unexpected results. In these cases, you must consider the required type and behavior of the members to obtain the desired output. In reporting environments, calculated members appear to be identical to regular members. It is a good practice to use a naming convention so that report users can easily identify calculated members.

The values of calculated members and measures are not retained within a dynamic cube. The values are computed at every occurrence within reports and analyses when executed.

You create Cognos Dynamic Cubes calculated members manually. Cognos Dynamic Cubes relative time calculated members are specialized calculated members automatically added to a relative time hierarchy and cannot be modified.

Calculated members that you manually create have the following characteristics:

- Each occurrence of a single calculated member in a report or analysis is considered unique. (SET operations, Filtering calculated members)
- They do not have siblings or children.
- They should not be nested.
- Their rank value in IBM Cognos Analysis Studio is always Null.

Relative time calculated members

The relative time feature generates three types of calculated members.

The Period to Date Change and Period to Date Growth relative time calculated members share the following characteristics with Cognos Dynamic Cubes calculated members.

- Each occurrence of a single calculated member in a report or analysis is considered unique. (SET operations, Filtering calculated members)
- They do not have siblings or children.
- They should not be nested
- Their rank value in IBM Cognos Analysis Studio is always Null.

The Current Period, Prior Period, Current Period to Date and Prior Period to Date members may have children. Therefore, the functions CHILDREN, DESCENDANT, FIRSTCHILD and LASTCHILD can return results. These relative time calculated members share the following characteristics with Cognos Dynamic Cubes calculated members:

- Each occurrence of a single calculated member in a report or analysis is considered unique. (SET operations, Filtering calculated members)
- They should not be nested
- Their rank value in IBM Cognos Analysis Studio is always Null.

Reference relative time members refer to other members within the time hierarchy and have the same caption and member key values as the members to which they refer. Within the context of other reference members, these members behave the same as Cognos Dynamic Cubes calculated members. Unlike Cognos Dynamic Cubes calculated members, these members are not considered unique, they can have

children and they can be nested. Reference members at the same level are siblings of other reference members. When applied to a reference member, functions such as FIRSTSIBLING or NEXTMEMBER will return a reference member. Their rank value in IBM Cognos Analysis Studio is always Null.

SET operations

Because a calculated member is considered to be unique from all other calculated members, the UNION, EXCEPT, UNIQUE, and INTERSECT functions may give results that appear incorrect.

In the following examples, [USA] and [Canada] are regular members and [CM1] and [CM2] are calculated members.

| <i>Table 57. Examples of SET operations with calculated members</i> | |
|---|--|
| Example | Result Set |
| UNION (SET([USA], [CM1], [CM2]), SET([USA], [Canada], [CM1])) | SET ([USA], [CM1], [CM2], [Canada], [CM1]) The member [CM1] appears twice in the result. |
| EXCEPT (SET([USA], [CM1], [CM2]), SET([USA], [Canada], [CM1])) | SET ([CM1], [CM2], [Canada], [CM1]) The member [USA] is removed, but the member [CM1] appears twice in the result. |
| UNIQUE (SET([USA], [CM1], [USA], [CM1], [Canada])) | SET ([USA], [CM1], [CM1], [Canada]) The member [CM1] appears twice in the result. |
| INTERSECT (SET([USA], [CM1], [CM2]), SET([USA], [Canada], [CM1])) | SET([USA]) Calculated members do not appear in the intersection of two sets. |

Filtering calculated members

Since calculated members are considered to be unique from all other calculated members, a filter will not remove the members.

If a report contains a filter based on IBM Cognos Dynamic Cubes calculated members and the same hierarchy is visible in the report, the data values in the report will be correct. However, the filter will not remove visible members from the report. If the same hierarchy is not visible in the report, the report output will be as expected.

Nesting calculated members

IBM Cognos Dynamic Cubes calculated members should not be nested. Because all calculated members are considered unique, the dynamic query mode query planner resolves the intersection to an empty set. The rows remain in the reports but the values are Null.

Calculated members siblings and children

IBM Cognos Dynamic Cubes calculated members do not have siblings or children. Functions that require a member sibling or child as a result will always be Null.

- NEXTMEMBER([CM1]) = NULL
- PREVMEMBER([CM2]) = NULL
- LEAD([CM1], 0) = NULL
- LAG(([CM2], 0) = NULL

Cognos Analysis Studio rank

In IBM Cognos Analysis Studio, the rank of an IBM Cognos Dynamic Cubes calculated member is always Null. The context in which the rank is computed and the context used to compute the values visible in the cross tab are not the same. Because, the computed rank values could contradict the visible values, the rank is always set to Null.

Relative time calculated members in reports

IBM Cognos Dynamic Cubes relative time members are specialized calculated members that are added to a time hierarchy.

The relative time feature generates three types of calculated members

Period to Date Change, Period to Date Growth

These relative time calculated members share the following characteristics with Cognos Dynamic Cubes calculated members.

- They are considered unique.
- They do not have siblings or children
- They should not be nested
- Their rank value in IBM Cognos Analysis Studio is always null.

Current Period, Prior Period, Current Period to Date, Prior Period to Date

These members behave the same as Cognos Dynamic Cubes calculated members with one exception. These members may have children. Therefore, the functions CHILDREN, DESCENDANT, FIRSTCHILD and LASTCHILD can return results.

These relative time calculated members share the following characteristics with Cognos Dynamic Cubes calculated members:

- They are considered unique.
- They should not be nested
- Their rank value in IBM Cognos Analysis Studio is always null.

Reference relative time members

These members refer to other members within the time hierarchy and have the same caption and member key values as the members to which they refer. Within the context of other reference members, these members behave the same as Cognos Dynamic Cubes calculated members with one exception. These members can have children. Reference members at the same level are siblings of other reference members. When applied to a reference member, functions such as FIRSTSIBLING or NEXTMEMBER will return a reference member.

These relative time calculated members share the following characteristics with Cognos Dynamic Cubes calculated members:

- They are considered unique.

- They can be nested
- Their rank value in IBM Cognos Analysis Studio is always null.

Removal of padding members from reports

The use of padding members can result in skewed calculations related to the members of a hierarchy level. If a level contains padding members, they are included in the count of members. In addition, because padding members can have associated fact data values, this can skew the value of aggregates computed on a level-basis.

For example, in a State/City hierarchy, if the state of California has no city level members, a padding member at the city level is created as a child of California to balance the hierarchy. If the Sales measure value for California is 100, then the child padding member also has a value of 100. The number of city entries across all states is now inflated by 1 and that the sum of all Sales values across all cities is inflated by 100.

To remove skewed data from a report, you can define a filter for a set of members based on a dynamic cube.

Hierarchies with padding members are not displayed as ragged or unbalanced in the IBM Cognos studios. A report user can identify ragged and unbalanced hierarchies by looking for members with a blank caption or the same caption as their parent. These members have a NULL business key because they do not represent real members. Filtering members with a NULL business key removes all the padding members. A report filter such as `FILTER(MEMBERS([My Level]), [My Level].[My Level - Key] = NULL)` removes padding members from the report.

Appendix C. DCAdmin command-line tool

This tool is available with IBM Cognos Analytics server. You can use this tool to run various administrative commands on dynamic cubes.

You access the DCAdmin command-line tool from the *cognos_analytics_location*\bin directory for 32-bit installations or from *cognos_analytics_location*\bin64 directory for 64-bit installations, by clicking on one of the following files:

- `dcadmin.bat` (Microsoft Windows)
- `dcadmin.sh` (UNIX)

You can run the following commands by using the DCAdmin command-line tool.

| Command | Description |
|-------------------------------------|--|
| <code>getCubeState</code> | Checks whether a dynamic cube is started, paused, or stopped. |
| <code>getCubeMetrics</code> | Checks the metrics of a dynamic cube before or after incremental load updates. For more information, see “Loading incremental updates to dynamic cubes” on page 148. |
| <code>startCubes</code> | Starts or resumes a published dynamic cube. |
| <code>forceStartCubes</code> | Starts a published dynamic cube by using the parameter startROLAPCubesAndSourceCubes . |
| <code>stopCubes</code> | Stops a dynamic cube. |
| <code>forceStopCubes</code> | Stops a dynamic cube by using the parameter stopROLAPCubesImmediately . |
| <code>restartCubes</code> | Restarts a dynamic cube. |
| <code>pauseCubes</code> | Pauses a dynamic cube to make it unavailable to report users. |
| <code>incrementallyLoadCubes</code> | Loads incremental updates to dynamic cube data caches. For more information, see “Loading incremental updates to dynamic cubes” on page 148. |
| <code>refreshCubeDataCache</code> | Refreshes the data cache for a dynamic cube. |
| <code>refreshCubeMemberCache</code> | Refreshes the member cache for a dynamic cube. |
| <code>refreshCubeSecurity</code> | Refreshes the security settings for a dynamic cube. |
| <code>clearCubeWorkloadLog</code> | Removes the workload logs for a dynamic cube. |

Instead of using the `pauseCubes` and `incrementallyLoadCubes` commands, you can perform relevant actions on the query service in IBM Cognos Administration. This method allows you to run these commands by schedule and by trigger. For more information, see [“Starting and managing dynamic cubes”](#) on page 127.

Syntax

Use the following syntax to run DCAdmin commands. Any parameter that contains a comma or a space must be included in double quotation marks. For example: "param1,param2".

```
dcadmin[.bat|.sh] [-p port] [-s server] [-x output_file]
  [-l "namespace,userid,password"]
  [-arg argName argValue] command [cube0 cube1 ...]
```

The following table describes the parameters used by the DCAdmin commands.

| <i>Table 59. DCAdmin command-line syntax</i> | |
|--|---|
| Parameter | Description |
| <i>-p port</i> | Specifies the port to use. Default: 9300. |
| <i>-s server</i> | Specifies the server name to use. Default: localhost |
| <i>-x output_file</i> | Specifies the name of the output file to which to write the structured command results. |
| <i>-l "namespace,userid,password"</i> | Specifies the Cognos Analytics server logon parameters. For example: <i>-l "LDAP,admin,secret123"</i> |
| <i>-arg argName argValue</i> | Specifies command arguments. transactionID is an optional argument for the incrementallyLoadCubes command. |

When you run the DCAdmin command-line tool, the output script is shown on the screen. You can also save the output to xml file for parsing by specifying *-x output_file*.

When a command is complete, the output script returns an exit code 0 if the command was successful. If there was an error, it returns an exit code 1.

Appendix D. Troubleshooting

This section provides solutions for problems you may encounter when using IBM Cognos Dynamic Cubes.

Possible overflow in measure attributes

Measure attributes in a dynamic cube may be too small to hold aggregate values of the measures.

The Measure properties of **Data type**, **Precision** and **Scale** are inherited from the relational database metadata and cannot be modified. If the aggregate value of a measure exceeds the size of the attribute, you see an error indicating that an overflow has occurred. For example, a Quantity Measure defined as Int(4) overflows when summed in a dynamic cube.

To avoid overflow errors, first evaluate the database columns you want to use as measures. If the resulting data type will not accommodate the aggregation value of the measure, do the following:

- Create a default measure for the database column you want to use as a measure.
- Evaluate the measure to determine an appropriate aggregate size.
- Hide the original measure that you determined to cause an overflow.
- Create a new measure.
- Define the measure using the expression property. The expression must be an explicit cast of the original measure into a larger data type.

The syntax for the CAST function is CAST (<expression>, <datatype>)

For example:

```
CAST( [MyDataItem], varchar(10))
```

If casting to a data type that accepts size, precision, or scale, those parameters appear in parentheses after the data type. For example

```
CAST( [MyDataItem], decimal(10,2))
```

In-memory aggregates fail to load

If in-memory aggregates fail to load when a dynamic cube starts, additional memory may be required for the aggregate cache.

In-memory aggregates are defined by running Aggregate Advisor in IBM Cognos Dynamic Query Analyzer and saving the in-memory aggregate definitions. When a dynamic cube is restarted, the in-memory aggregates are loaded. If they fail to load, check the dynamic cube error log for the following message:

```
"Loading of in-memory aggregates was skipped because the value for the  
'Maximum amount of memory to use for aggregate cache'' property is zero.  
To enable loading in-memory aggregates, update the property to a value greater  
than zero to be the amount of memory to allocate for the aggregate cache."
```

In IBM Cognos Administration, open the properties for the cube and set the **Maximum amount of memory to use for aggregate cache** to a value greater than or equal to the one used when creating the recommendations in Aggregate Advisor.

Issues with dynamic cubes that contain members with duplicate level keys

In the previous release, it was possible to model a hierarchy level with members that contain duplicate level keys.

In IBM Cognos Dynamic Cubes 10.2.1 and later versions, when you browse members with duplicate level keys in IBM Cognos Cube Designer, the member tree is constructed as you expand each member, and no check is done for members with duplicate level keys. However, when you start a dynamic cube that contains members with duplicate level keys, the cube might now fail with an error. This behavior is a result of improved validation in this release.

To avoid this error, you can update the hierarchy level that contains duplicate level key members by specifying more columns to ensure that the level key is unique.

If you don't want to update the dynamic cube, you can disable the new validation check. To do that, for every dispatcher that owns a query service, set the query service property **Additional JVM arguments for the query service** to **-DdisableDuplicateLevelCheck=true**, and restart the query service. For more information, see [“Setting query service properties for dynamic cubes”](#) on page 129.

Issues with starting a published dynamic cube in a multi-server environment

IBM Cognos Cube Designer supports starting published cubes only in a single-server environment.

When you publish a cube to a multi-server configuration and select all options to start the cube, after publishing, you might see an error message similar to this one:

```
XQE-ROL-0002 Cannot find cube cube_name,  
verify if cube exists in Cognos Content Manager.
```

This error can occur when the cube start command is not sent to the correct server.

In multi-server environments, you can start and stop cubes in IBM Cognos Administration.

Notices

This information was developed for products and services offered worldwide.

This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. This document may describe products, services, or features that are not included in the Program or license entitlement that you have purchased.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group
Attention: Licensing

3755 Riverside Dr.
Ottawa, ON
K1V 1B7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's

- name
- user name
- password

for purposes of

- session management
- authentication
- enhanced user usability
- single sign-on configuration
- usage tracking or functional purposes other than session management, authentication, enhanced user usability and single sign-on configuration

These cookies cannot be disabled.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at <https://www.ibm.com/privacy/us/en/>.

Index

A

- access permissions
 - dynamic cubes [117](#)
- accessibility features
 - keyboard shortcuts [179](#)
- adding to query service
 - dynamic cubes [126](#)
- administration
 - dynamic cubes [117](#)
- Aggregate Advisor
 - in-database [34](#)
 - in-memory [34](#)
 - user-defined in-memory aggregates [97](#)
- aggregate awareness [33, 93](#)
- aggregate slicer [97](#)
- aggregate tables
 - incremental updates [150](#)
 - pausing a cube [151](#)
- aggregates
 - user-defined in-memory aggregates [97](#)
- aggregation rules
 - calculated [27, 28](#)
 - defining [66](#)
 - time state [28](#)
- attributes
 - level key [22, 24](#)
 - level unique key [24](#)
 - member caption [24](#)
 - member description [24](#)
 - overview [24](#)
 - relational mapping [25](#)
 - support multiple locales [90](#)
- automatic optimization in-memory aggregates
 - properties [135](#)

B

- balanced hierarchies [16](#)
- browsing
 - members in a hierarchy [55](#)
- business keys, *See* level keys

C

- calculated measures [73, 183](#)
- calculated members
 - defining [77](#)
 - relative time [183, 185](#)
 - security [111](#)
 - unique [183](#)
- calculations
 - embedded [169](#)
 - stand-alone [170](#)
- capabilities
 - dynamic cubes [117](#)
- cardinality

- cardinality (*continued*)
 - Merise notation [161](#)
 - relationships [161](#)
- Cognos Cube Designer
 - overview [37](#)
 - user interface [37](#)
- Cognos Dynamic Cubes
 - overview [7](#)
 - workflow [9](#)
- configuring
 - dynamic cubes [69, 125](#)
 - memory monitoring [141](#)
- controlling the automatic generation of members
 - next period [87](#)
 - prior period [87](#)
 - sub tree of reference members [87](#)
- creating
 - measure folders [68](#)
 - packages [70](#)
- creating a signon
 - dynamic cubes [124](#)
- cube model
 - creating a Framework Manager project for [61](#)
- current data [33](#)
- current period
 - expression examples [88](#)
- current period examples
 - expressions [88](#)
- current period expressions
 - examples [74, 88](#)
- custom N period running total [81, 85](#)
- custom period-to-date [81, 83](#)
- custom single period [81, 82](#)

D

- data members
 - hiding [21](#)
 - leaf [21](#)
 - non-leaf [21](#)
 - overview [21](#)
 - parent-child hierarchies [21](#)
 - showing [21](#)
- data source connections
 - signons [124](#)
- data sources
 - importing metadata [39](#)
- DCAAdmin command-line tool [187](#)
- defining
 - dimension filters [56](#)
 - dimensions [45, 46](#)
 - filters [168](#)
 - level unique keys [51](#)
 - measure dimension filters [67](#)
 - security filters [113](#)
 - security views [115](#)
 - sort order of members [52](#)

- defining (*continued*)
 - virtual dimensions [100](#)
- denying
 - security [107](#), [113](#)
- deploying
 - dynamic cubes [69](#)
- design language [89](#)
- determinants
 - adding [160](#)
 - changing the order of [160](#)
 - specifying [158](#)
 - uniquely identified [158](#)
- dimension filters
 - defining [56](#)
- dimensional metadata
 - attributes [24](#)
 - dimensions [15](#)
 - hierarchies [15](#)
 - joins [23](#)
 - levels [22](#)
 - modeling [45](#), [153](#)
 - overview [15](#)
 - parent-child hierarchies [21](#)
- dimensions
 - defining [45](#), [46](#)
 - degenerate [15](#)
 - DMR models [164](#)
 - filter [55](#)
 - modeling [45](#)
 - parent-child [15](#)
 - regular [15](#)
 - relational mapping [25](#)
 - relationships [167](#)
 - relative time [77](#), [86](#)
- disabling
 - memory monitoring [141](#)
- DMR modeling
 - enabling [154](#)
- DMR models
 - creating [163](#)
 - dimensions [164](#)
 - hierarchies [164](#), [165](#)
 - levels [164](#), [165](#)
 - measure dimensions [166](#)
 - measures [166](#)
 - relationships [167](#)
- double-counting [158](#)
- dynamic cubes
 - access permissions and capabilities [117](#)
 - adding to query service [126](#)
 - administration [117](#)
 - advanced modeling [73](#)
 - based on a relational table [62](#)
 - configuring [69](#), [125](#)
 - creating a signon [124](#)
 - creating from InfoSphere Warehouse Cubing Services model [41](#)
 - defining manually [61](#)
 - deploying [69](#)
 - editing [43](#)
 - errors [44](#)
 - failure to start a published cube [190](#)
 - generating from a relational table [61](#)
 - importing metadata [40](#)
- dynamic cubes (*continued*)
 - issues [44](#)
 - measure dimension [24](#)
 - measures [26](#)
 - modeling [61](#)
 - multiple dispatchers [125](#)
 - opening [43](#)
 - overview [24](#)
 - properties [132](#)
 - publishing [69](#)
 - removing from query service [126](#)
 - saving [43](#)
 - security [107](#)
 - specify a server group name [125](#)
 - starting [69](#)
 - starting and managing [127](#)
 - validating [44](#)
 - warnings [44](#)
- dynamic cubes developer role
 - access permissions and capabilities [121](#)
- dynamic query mode server
 - memory monitoring
 - dynamic query mode server [141](#)

E

- editing
 - dynamic cubes [43](#)
 - projects [43](#)
- errors [44](#)
- estimating hardware requirements
 - calculator [70](#)
- examples
 - calculated members [74](#)
 - level current period expressions [88](#)
- execution trace
 - query service [129](#)
- expression editor
 - defining calculated members [77](#)
- extraneous padding members
 - hiding [20](#)
 - showing [20](#)

F

- failure to start dynamic cubes
 - browse member issue [189](#)
- features
 - new [1](#)
- filtering data
 - in-database aggregates [97](#)
- filters
 - dimension [55](#), [56](#)
 - embedded [169](#)
 - measure dimension [67](#)
 - stand-alone [168](#)
- folders
 - measure [68](#)
- Framework Manager
 - relational and DMR models [153](#)
- Framework Manager package
 - importing [40](#)

G

governors

- (DQM) Adjust SQL generation for exact numeric division [173](#)
- (DQM) Cache is sensitive to DB info [174](#)
- (DQM) Cache is sensitive to model security [174](#)
- (DQM) Cursor mode [175](#)
- (DQM) Local cache policy [175](#)
- (DQM) Multi fact join operator [176](#)
- (DQM) Summary query join operator [176](#)
- Allow usage of local cache [173](#)
- Cache is sensitive to connection command blocks [174](#)
- Cross-product Joins [171](#)
- Maximum number of retrieved rows [171](#)
setting [171](#), [177](#)
- SQL generation for determinant attributes [172](#)
- SQL generation for level attributes [172](#)
- SQL join syntax [171](#)
- SQL parameter syntax [173](#)
- Use WITH clause when generating SQL [173](#)

granting

- security [107](#), [113](#)

H

hiding

- data members [21](#)
- extraneous padding members [20](#)
- measures [65](#), [73](#)

hierarchies

- balanced [16](#)
- browsing members [55](#)
- default [110](#)
- defining [49](#)
- defining security for [107](#)
- DMR models [164](#), [165](#)
- modeling [48](#)
- multiple [15](#)
- overview [15](#)
- padding members [18](#)
- ragged [17](#)
- relational mapping [25](#)
- security [113](#)
- sort order of members [52](#)
- unbalanced [16](#)

holidays [78](#)

I

identifiers

- unique [158](#)

idle connection timeout

- query service [129](#)

importing

- Framework Manager package [40](#)
- InfoSphere Warehouse Cubing Services model [41](#)
metadata for a relational model [154](#)

importing metadata

- Content Manager data source metadata [39](#)

importing metadata from data sources [39](#)

in-database [34](#)

in-database aggregates

in-database aggregates (*continued*)

- defining [96](#)
- defining automatically [94](#)
- defining manually [95](#)
- filtering data for [97](#)
- in-memory aggregates [33](#)
- modeling [93](#)
- overview [33](#)
- parent-child dimension [96](#)
- properties [93](#)

in-memory [34](#)

in-memory aggregates

- automatic optimization [135](#)
- defined by users [97](#)
- fail to load error [189](#)

incremental updates

- aggregate tables [150](#)

InfoSphere Warehouse Cubing Services

- importing cube metadata [41](#)

IPF logging

- ipfclientconfig.xml [144](#)
- ipfCubeDesignerclientconfig.xml.sample [144](#)

issues [44](#)

J

joins

- measure-to-dimension [62](#), [67](#)
- overview [23](#)
- query subjects [162](#)
- relational mapping [25](#)
- relationships [167](#)

K

keyboard shortcuts [179](#)

L

languages

- adding to metadata objects [90](#)
- design language [89](#)
- support multiple locales [90](#)
- supporting different locales [89](#)

level keys

- attribute [24](#)

level unique keys

- attribute [24](#)
- defining [51](#)

levels

- alternative modeling [22](#)
- best practice modeling [22](#)
- DMR models [164](#), [165](#)
- modeling [49](#)
- overview [22](#)
- relational mapping [25](#)

locales

- adding languages to metadata objects [90](#)
- adding to members and attributes [90](#)
- design language [89](#)
- supporting [89](#)

M

- measure dimension
 - filters [67](#)
- measure dimension filters
 - defining [67](#)
- measure dimensions
 - DMR models [166](#)
 - relationships [167](#)
- measure folders
 - creating [68](#)
 - sorting [68](#)
- measure-to-dimension joins
 - defining [67](#)
- measures
 - aggregation rules [27](#), [28](#)
 - calculated [73](#), [183](#)
 - calculated measures [26](#)
 - DMR models [166](#)
 - dynamic calculated measures [26](#)
 - folders [68](#)
 - hidden [65](#), [73](#)
 - modeling [63](#), [65](#)
 - overview [26](#)
 - regular aggregation [27](#), [28](#)
 - security [113](#)
 - sorting [68](#)
 - visible [65](#), [73](#)
- member caption
 - attribute [24](#)
- member description
 - attribute [24](#)
- members
 - browsing in hierarchies [55](#)
 - calculated [73](#), [183](#), [185](#)
 - defining security for [107](#)
 - defining sort order [52](#)
 - leaf [21](#)
 - named sets [56](#)
 - non-leaf [21](#)
 - predefined relative time [77](#)
 - support multiple locales [90](#)
- memory monitoring
 - configuring [141](#)
- metadata
 - adding languages [90](#)
 - importing from a Content Manager data source [39](#)
 - importing from a data source [39](#)
 - importing metadata for [41](#)
- modeling
 - advanced [73](#)
 - dimensional metadata [45](#), [153](#)
 - dimensions [45](#)
 - dynamic cubes [61](#)
 - hierarchies [48](#)
 - in-database aggregates [93](#)
 - levels [49](#)
 - measures [63](#)
 - parent-child hierarchies [52](#)
 - relative time dimensions [77](#)
 - virtual cubes [99](#)
 - virtual dimensions [100](#)
 - virtual hierarchies [101](#)
 - virtual measures [104](#)

- modeling (*continued*)
 - virtual members [103](#)
- multiple hierarchies [15](#)

N

- named sets
 - defining [56](#)
- near real-time updates
 - incremental updates [148](#)
- near real-time updates of cubes data
 - enabling [147](#)
- new features
 - administration interface [6](#)
 - Aggregation Advisor [6](#)
 - aggregation rules [5](#)
 - dimension filters [5](#)
 - embedded prompts and macros [5](#)
 - generate cube [5](#)
 - generate dimension [5](#)
 - measure dimension filters [5](#)
 - measure folders [5](#)
 - measure sorting [5](#)
 - metadata import [5](#)
 - performance issues [6](#)
 - security [6](#)
 - version 10.2.2 [1](#)
 - version 10.2.2 FP1 [1](#)
- next period [80](#)
- next period to date [80](#)
- next period to date % growth [80](#)
- next period to date change [80](#)
- non-rollup hierarchy [20](#), [21](#)

O

- objects
 - errors [44](#)
 - issues [44](#)
 - validating [44](#)
 - warnings [44](#)
- opening
 - dynamic cubes [43](#)
 - projects [43](#)
- overview
 - Cognos Cube Designer [37](#)
 - Cognos Dynamic Cubes [7](#)
 - dimensional metadata [15](#)
 - dimensions [15](#)
 - dynamic cubes [24](#)
 - hierarchies [15](#)
 - in-database aggregates [33](#)
 - parent-child hierarchies [21](#)
 - user interface [37](#)
 - virtual cubes [30](#)

P

- packages
 - creating [70](#), [170](#)
 - publishing [70](#), [170](#)
 - securing [177](#)
- padding members

- padding members (*continued*)
 - removing from reports [186](#)
- parameter maps
 - based on existing query items [58](#)
 - importing entries [58](#)
 - manually entered keys and values [58](#)
- parent member [73](#)
- parent-child hierarchies
 - data members [21](#)
 - defining [54](#)
 - modeling [52](#)
 - overview [21](#)
- pausing a dynamic cube
 - aggregate tables [151](#)
- performance issues [6](#)
- pre-cached historical data [33](#)
- projects
 - adding support for locales [89](#)
 - creating for a cube [61](#)
 - defining virtual cubes [99](#)
 - design language [89](#)
 - editing [43](#)
 - errors [44](#)
 - importing metadata [40](#)
 - importing metadata for [41](#)
 - issues [44](#)
 - opening [43](#)
 - saving [43](#)
 - validating [44](#)
 - warnings [44](#)
- properties
 - query service [129](#)
- propertiesdynamic cubes
 - properties [132](#)
- publishing
 - dynamic cubes [69](#)
 - packages [70](#)

Q

- query item sets [157](#)
- query items
 - creating [155](#)
 - properties [156](#)
 - query item sets [157](#)
- query planning trace
 - query service [129](#)
- query service
 - creating and scheduling administration tasks [138](#)
 - execution trace [129](#)
 - idle connection timeout [129](#)
 - properties [129](#)
 - query planning trace [129](#)
 - run tree trace [129](#)
 - starting [131](#)
 - stopping [131](#)
 - write model to file [129](#)
- query subjects
 - defining query items [155](#)
 - relationships [160](#), [163](#)
- query subjects for relational metadata
 - determinants [158](#)

R

- ragged hierarchies [17](#)
- regular aggregation [27](#), [28](#)
- relational model
 - creating [154](#)
 - importing metadata [154](#)
- relational modeling
 - enabling [154](#)
- relational tables
 - using for dynamic cubes [62](#)
- relationships
 - cardinality [161](#)
 - dimensions [167](#)
 - joins [167](#)
 - joins between tables [162](#)
 - query subjects [160](#), [163](#)
 - scope [168](#)
- relative time
 - calculated members [183](#), [185](#)
- relative time dimensions
 - controlling the automatic generation of members [87](#)
 - creating a custom member [88](#)
 - custom N period running total [81](#), [85](#)
 - custom period-to-date [81](#), [83](#)
 - custom single period [81](#), [82](#)
 - defining [86](#)
 - holidays [78](#)
 - modeling [77](#)
 - next period [80](#)
 - next period to date [80](#)
 - next period to date % growth [80](#)
 - next period to date change [80](#)
 - seasons [78](#)
 - semesters [78](#)
 - trimesters [78](#)
- renewing
 - trusted credentials [124](#)
- ROLAP packages [70](#)
- role-based [113](#)
- rollup hierarchy [20](#), [21](#)
- run tree trace
 - query service [129](#)

S

- saving
 - dynamic cubes [43](#)
 - projects [43](#)
- scenarios
 - virtual cube [33](#)
- schedules
 - query service administration tasks [138](#)
- schemas
 - snowflake [24](#)
 - star [24](#)
- scope
 - relationships [168](#)
- seasons [78](#)
- security
 - calculated members [111](#)
 - denying access [107](#), [113](#)
 - dynamic cubes [107](#)
 - granting access [107](#), [113](#)

- security (*continued*)
 - hierarchies [107](#), [113](#)
 - measures [113](#)
 - members [107](#)
 - tuple [114](#)
 - virtual cubes [107](#)
- security filters
 - defining manually [113](#)
- security views
 - creating [177](#)
 - defining [115](#)
 - securing packages [177](#)
- semesters [78](#)
- setting
 - governors [171](#)
- shared dimensions [33](#), [47](#)
- shared member cache [47](#)
- showing
 - data members [21](#)
 - extraneous padding members [20](#)
- simple aggregate table [94](#)
- snowflake schemas [24](#)
- sort order
 - defining for members [52](#)
- sorting
 - measure folders [68](#)
 - measures [68](#)
- stand-alone
 - calculations [170](#)
- star schemas [24](#)
- starting
 - dynamic cubes [69](#)
 - query service [131](#)
- starting and managing
 - dynamic cubes [127](#)
 - virtual cubes [127](#)
- stopping
 - query service [131](#)

T

- trimesters [78](#)
- troubleshooting
 - in-memory aggregates [189](#)
 - members with duplicate level keys [189](#)
 - starting a published dynamic cube [190](#)
- trusted credentials
 - renewing [124](#)
- tuple
 - security [114](#)

U

- unbalanced hierarchies [16](#)
- unique identifiers [158](#)
- uniquely identified determinants [158](#)
- updating cube data in near real-time [147](#)
- user interface [37](#)
- user-defined in-memory aggregates [97](#)

V

- viewing

- viewing (*continued*)
 - members in a hierarchy [55](#)
- virtual cubes
 - current data [33](#)
 - defining in a project [99](#)
 - defining virtual dimensions [100](#)
 - modeling [99](#)
 - objects [30](#)
 - overview [30](#)
 - pre-cached historical data [33](#)
 - scenarios [33](#)
 - security [107](#)
 - shared dimensions [33](#)
 - starting and managing [127](#)
- virtual dimensions
 - defining [100](#)
 - modeling [100](#)
- virtual hierarchies
 - modeling [101](#)
- virtual measures
 - modeling [104](#)
- virtual members
 - modeling [103](#)
- visible [73](#)

W

- warnings [44](#)
- workflow
 - Cognos Dynamic Cubes [9](#)
- workload logging [134](#)
- workload logs [134](#)
- write model to file
 - dynamic cubes
 - query service properties [129](#)
 - query service [129](#)

