IBM Cognos Dynamic Cubes

Discover how Cognos drives performance with flexible in-memory acceleration

Learn how the Aggregate Advisor makes optimizing performance easier

Maximize the value of your large data warehouse

ibm.com/redbooks
Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (October 2012)
This edition applies to Version 10, Release 2, Modification 0 of IBM Cognos Business Intelligence (product number 5724-W12).
# Contents

## Notices
- ix

## Trademarks
- x

## Preface
- xi
  - The team who wrote this book: xii
  - Now you can become a published author, too: xv
  - Comments welcome: xv
  - Stay connected to IBM Redbooks: xv

## Chapter 1. Overview of Cognos Dynamic Cubes
- 1
  - 1.1 Introduction to IBM Cognos Dynamic Cubes: 2
    - 1.1.1 High performance interactive analysis: 3
    - 1.1.2 Enterprise data warehouses for analytics: 4
    - 1.1.3 Data volumes and context in the Enterprise Data Warehouse: 4
    - 1.1.4 Architecture summary: 4
  - 1.2 Skillset requirements for IBM Cognos Dynamic Cubes: 6
    - 1.2.1 Modeling: 6
    - 1.2.2 Cognos administration: 6
    - 1.2.3 Infrastructure design: 6
  - 1.3 When to use Cognos Dynamic Cubes: 7
  - 1.4 System requirements: 7
  - 1.5 Comparison to other IBM cube technologies: 8

## Chapter 2. IBM Cognos Dynamic Cubes architecture
- 9
  - 2.1 Dynamic cubes in the IBM Cognos environment: 10
    - 2.1.1 Dynamic cubes and the data warehouse: 11
    - 2.1.2 Modeling dynamic cubes: 13
    - 2.1.3 Dynamic cubes as data sources: 16
    - 2.1.4 Configuring dynamic cubes: 17
    - 2.1.5 Dynamic cube management: 19
    - 2.1.6 Virtual cubes: 20
    - 2.1.7 Aggregate Advisor: 22
  - 2.2 Dynamic cubes and the Dynamic Query Mode server: 24
    - 2.2.1 Dynamic cubes and report server: 24
    - 2.2.2 Dynamic cubes within the Dynamic Query Mode server: 25
  - 2.3 Cognos Dynamic Cubes caching: 27
    - 2.3.1 Result set cache: 28
    - 2.3.2 Expression cache: 29
    - 2.3.3 Member cache: 30
    - 2.3.4 Aggregate cache: 30
    - 2.3.5 Data cache: 31
  - 2.4 Dynamic cubes query processing: 32
    - 2.4.1 Metadata query processing in the Dynamic Query Mode server: 32
    - 2.4.2 OLAP query processing in the Dynamic Query Mode server: 32
  - 2.5 Using the database to evaluate set expressions: 34
  - 2.6 Management of Cognos Dynamic Cubes: 34
Chapter 3. Installation and configuration of Cognos Cube Designer and Dynamic Query Analyzer ........................................... 37
3.1 Introduction ............................................................................. 38
3.2 Cognos Cube Designer operating requirements ....................... 38
   3.2.1 IBM Cognos 10.2 Business Intelligence Server .................. 38
   3.2.2 IBM Cognos 10.2 Report Server ...................................... 39
   3.2.3 IBM Cognos 10.2 Framework Manager ............................... 39
   3.2.4 Java Runtime Environment Libraries ................................. 39
   3.2.5 Supported relational databases ....................................... 40
   3.2.6 Cognos Cube Designer supported operating systems .......... 40
3.3 Cognos Dynamic Cubes hardware requirements for the BI Server ................................................................. 40
3.4 Installing IBM Cognos Cube Designer ...................................... 40
3.5 Installing Dynamic Query Analyzer ........................................... 47
   3.5.1 Dynamic Query Analyzer installation requirements ............ 47
   3.5.2 Installation procedure ................................................... 48
   3.5.3 Configuring Dynamic Query Analyzer ............................... 55

Chapter 4. Modeling dynamic cubes .............................................. 59
4.1 Introduction to IBM Cognos Cube Designer .............................. 60
4.2 Working with Cognos Cube Designer ........................................ 60
   4.2.1 Launching Cognos Cube Designer ..................................... 60
   4.2.2 Creating a new project ................................................... 61
   4.2.3 The Cognos Cube Designer workspace .............................. 61
   4.2.4 Creating calculated measures ........................................ 64
   4.2.5 Importing metadata ..................................................... 66
   4.2.6 Exploring data using the Relational Explorer Diagram ........ 66
   4.2.7 Exploring data using the tabular data view ....................... 68
   4.2.8 Creating a dynamic cube .............................................. 68
4.3 Modeling dimensions .............................................................. 70
   4.3.1 Relationships .................................................................. 71
   4.3.2 Modeling levels ............................................................ 71
   4.3.3 Member tree organization options ..................................... 73
   4.3.4 Modeling level keys ....................................................... 75
   4.3.5 Parent-child dimensions .............................................. 77
4.4 Calculated members ................................................................. 78
4.5 Modeling time dimensions for relative time ............................ 78
   4.5.1 Using relative time ....................................................... 79
4.6 Slowly changing dimensions .................................................. 79
   4.6.1 Modeling slowly changing dimensions ............................ 79
   4.6.2 Metadata discovery ..................................................... 80
   4.6.3 Slowly changing dimension as a parent-child dimension .... 80
   4.6.4 Modeling a slowly changing dimension in a regular dimension ................................................................. 81
4.7 Role-playing dimensions ........................................................ 82
4.8 Multi-grain fact scenarios ......................................................... 84
4.9 Aggregate cubes ................................................................. 85
   4.9.1 Modeling aggregate cubes ............................................ 85
   4.9.2 Degenerate dimensions with matching level keys ............... 87
   4.9.3 Degenerate dimensions with matching join keys ............... 87
   4.9.4 Rolled-up dimensions .................................................. 87
   4.9.5 Parent-child dimensions .............................................. 87
   4.9.6 Custom aggregation ................................................. 87
   4.9.7 Slicers .................................................................... 88
   4.9.8 Other modeling considerations .................................... 88
10.3.5 Verifying changes to the BranchProfitByYear Report .................. 275
10.4 Virtual cube query execution tracing ........................................... 277
10.5 Dynamic cube QueryService logging ............................................. 278
10.6 Dynamic cube query plan tracing ................................................ 279
10.7 Aggregate matching .................................................................. 279
10.8 Common problems ...................................................................... 279
  10.8.1 Configuring Cognos Dynamic Cubes logging ......................... 280
  10.8.2 Cube does not start ............................................................... 282
  10.8.3 Unexpected data values returned ............................................. 285
  10.8.4 Cube loaded but relative time members are missing .................. 288

Related publications .......................................................................... 293
IBM Redbooks ............................................................................... 293
Online resources ............................................................................ 293
Help from IBM ............................................................................... 293
Notices

This information was developed for products and services offered in the U.S.A.

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.

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.

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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

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.

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.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (© or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

- Cognos®
- Cube Views®
- DB2 OLAP Server™
- DB2®
- IBM®
- Impromptu®
- Informix®
- InfoSphere®
- PowerPlay®
- Redbooks (logo) ®
- ReportNet®
- TM1®

The following terms are trademarks of other companies:

Netezza, and N logo are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.
IBM® Cognos® Business Intelligence (BI) provides a proven enterprise BI platform with an open data strategy, providing customers with the ability to leverage data from any source, package it into a business model, and make it available to consumers in various interfaces that are tailored to the task.

IBM Cognos Dynamic Cubes complements the existing Cognos BI capabilities and continues the tradition of an open data model. It focuses on extending the scalability of the IBM Cognos platform to enable speed-of-thought analytics over terabytes of enterprise data, without having to invest in a new data warehouse appliance. This capability adds a new level of query intelligence so you can unleash the power of your enterprise data warehouse.

This IBM Redbooks® publication addresses IBM Cognos Business Intelligence V10.2 and specifically, the IBM Cognos Dynamic Cubes capabilities. This book can help you in the following ways:

- Understand core features of the Cognos Dynamic Cubes capabilities of Cognos BI V10.2
- Learn by example with practical scenarios by using the IBM Cognos samples

This book uses fictional business scenarios to demonstrate the power and capabilities of IBM Cognos Dynamic Cubes. It primarily focuses on the roles of the modeler, administrator, and IT architect.

Information about IBM Cognos Business Intelligence 10.2.0 is in the information center:

http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/index.jsp

Also see the *Dynamic Cubes User Guide*, v10.2.0:

http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/nav/5_6
The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Dmitry Beryozahas over 20 years of experience in software design and implementation. He has been working at IBM since 2001, focusing on data access technologies (including Dynamic Query and Dynamic Cubes) in several generations of IBM Cognos products. Before joining IBM, Dmitry worked on software projects for a number of companies and research labs as an Architect, Developer, and Consultant. He holds a Ph.D. in Computer Science and has extensive experience in Java and C++ programming, and web and systems software development.

Shilpa Bhimavarapu joined IBM at the Silicon Valley Laboratory in 2008 and was a member of the IBM InfoSphere® Cubing Services and Alphablox quality assurance teams before joining the Dynamic Cube quality engineering group as a Functional Verification Tester and an Integration Test Engineer. Shilpa has a Bachelor's degree in Computer Science and Engineering from the Chaitanya Bharathi Institute of Technology in India, and a Master's degree in Software Engineering from San Jose State University in the US.

Alan Blake is a Technical Solutions Architect and is a member of IBM North America IOT Advanced POC Team based in New York and New Jersey. He has over 15 years in developing, implementing, and supporting business analytic and data warehouse solutions in the finance, healthcare, and local government sectors. His current role is to demonstrate the suitability of IBM Business Analytic solutions within the customer’s current infrastructure. Alan holds a Bachelor’s degree in Technology and Business Studies from the University of Strathclyde in Scotland and a Master’s degree in Information Systems Management from Stevens Institute of Technology, New Jersey, US.

David Cushing is the Lead Architect for IBM Cognos Dynamic Cubes, based in Ottawa, Canada. David was a member of the Cognos data access team for 23 years and joined IBM as a result of the acquisition of Cognos. He has been involved with the development of relational and multi-dimensional data access in various products, including IBM Impromptu®, IBM PowerPlay®, IBM Cognos ReportNet®, Cognos 8 and Cognos 10. David was part of the team that produced the Dynamic Query Mode server before becoming the Lead Architect for IBM InfoSphere Cubing Services. David has a Master’s of Computer Science degree from the Technical University of Nova Scotia (now a part of Dalhousie University), Canada, and holds several patents in the area of query generation and processing.
Vlaunir Da Silva is a Technology Solutions Architect on the IBM Business Analytics Advanced Proof of Concept Team in Miami, Florida, US. He is originally from Brazil and holds a degree in Computer Sciences. Vlaunir has over 15 years of experience in business analytics, database support, and application development. His expertise is primarily in the areas of business analytics applications, databases, data management, data warehousing, business intelligence, and software support. He has written several technical articles about these subjects, taught classes, and presented at international conferences. Vlaunir is a certified in these areas: Online III Engineer (Dial-up Engineer), IBM DB2® Advanced Database Administrator, IBM Informix® System Administrator, Netezza® Technical Professional, and Informix 4GL Developer.

Mario Daigle is a Senior Product Manager in the IBM Cognos BI Platform with over 12 years of experience in the software industry, having led initiatives in globalization, extensibility, cloud. More recently, his focus is on IBM Cognos query technology.

Avery Hagleitner is an Advisory Software Engineer in business analytics at the IBM Silicon Valley Laboratory. She has over 11 years of software development experience at IBM, with interests that range from high-performance JavaServer applications to engaging client user interfaces. Her areas of expertise include data warehousing, online analytical processing (OLAP), business intelligence, and enterprise content management. Avery holds a Master’s degree in Software Engineering from San Jose State University, in the US, and a Bachelor of Science in Computer Science and minor in Psychology from the University of California, San Diego, US.

Ian Henderson is the QE Technical Team Lead for IBM Cognos Business Analytics Modeling applications. He has nearly 25 years of experience in the area of business intelligence, modeling, and reporting.

Beth Jubera has been a Software Engineer at IBM Silicon Valley Laboratory, San Jose, California for 10 years. She was a member of the development team for InfoSphere Warehouse Cubing Services and is currently a Senior Developer for IBM Cognos Dynamic Cubes. She holds a Bachelor’s degree in Computer Engineering from the University of Michigan and a Master’s degree in Software Engineering from San Jose State University, both in the US.

Skyla Loomis is a Senior Development Manager in the IBM Cognos Business Intelligence product area of the Business Analytics group. She leads the development of the IBM Cognos Dynamic Cubes solution, which is a new capability introduced in Cognos 10.2. Skyla has over 10 years of experience in the BI and warehousing space, previously leading areas of development and quality assurance in InfoSphere Warehouse.
Thanks to the following people for their contributions to this project:

Ralf Vierich
Architect, Framework Manager and Cube Designer, Business Analytics

Greg Harris
Software Engineer, SVT Quality Engineering, Business Analytics

Sarah Jeong
Business Unit Executive, Business Intelligence Market Strategy, Business Analytics

Marcela Adan
IBM Redbooks Project Leader, International Technical Support Organization

Stephen Smith
IBM Redbooks Technical Writer, International Technical Support Organization

---

Chris McPherson is a Senior Product Manager on the Cognos Platform team in the IBM Canada Ottawa Lab. He is responsible for relational modeling and metadata for IBM Cognos BI and has more than eight years of experience within the IBM Business Analytics organization. Chris is a graduate of the University of Western Ontario and Information Technology Institute.

Steven Rogers is an IBM Business Analytics Solution Architect and is a senior member of the IBM Business Analytics European Products and Technology Experts team. He has more than eight years of experience within the IBM Business Analytics organization and is a specialist in IBM Cognos Business Intelligence and TM1® implementations. Based in London, his current role is focused on ensuring that IBM customers achieve the target business benefit from their proposed Business Analytics solutions. Steven's prior experience includes specialization in RDBMS systems and application development using various programming languages and technologies. Steven is an IBM Certified IT Specialist.

Carlos Ruiz has been working in business reporting and business analytics since 1989. He has worked for several business intelligence companies, and is now working at the Business Analytics division of IBM. He is presently part of the Subject Matter Experts team at IBM where he specializes in business analytics on SAP and Teradata.

Christopher Yao is an Advisory Software Engineer at the IBM Silicon Valley Laboratory in San Jose, California, US, and is currently the team lead of the IBM Cognos Dynamic Cubes QA team. Since joining IBM in 2000, he worked on a number of other IBM OLAP products including DB2 OLAP Server™, IBM DB2 Cube Views®, and InfoSphere Warehouse Cubing Services. He holds a Bachelor of Science degree in Computer Science and Engineering with a minor in Managerial Economics from the University of California, Davis, US.
Now you can become a published author, too

Here’s an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

► Use the online Contact us review Redbooks form found at:
  ibm.com/redbooks

► Send your comments in an email to:
  redbooks@us.ibm.com

► Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

► Find us on Facebook:
  http://www.facebook.com/IBMRedbooks

► Follow us on Twitter:
  http://twitter.com/ibmredbooks

► Look for us on LinkedIn:
  http://www.linkedin.com/groups?home=&gid=2130806

► Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

► Stay current on recent Redbooks publications with RSS Feeds:
  http://www.redbooks.ibm.com/rss.html
Overview of Cognos Dynamic Cubes

This chapter provides an overview of IBM Cognos Dynamic Cubes: how the solution extends the scalability of the IBM Cognos Business Intelligence platform and uses the core strengths of an enterprise data warehouse. This chapter also describes the architecture of Dynamic Cubes: the IBM Cognos Cube Designer, the IBM Cognos Dynamic Cubes Server, and the Aggregate Advisor, a part of IBM Cognos Dynamic Query Analyzer.

This chapter contains the following sections:

- **1.1, “Introduction to IBM Cognos Dynamic Cubes” on page 2**
- **1.2, “Skillset requirements for IBM Cognos Dynamic Cubes” on page 6**
- **1.3, “When to use Cognos Dynamic Cubes” on page 7**
- **1.4, “System requirements” on page 7**
- **1.5, “Comparison to other IBM cube technologies” on page 8**
1.1 Introduction to IBM Cognos Dynamic Cubes

With the advent of new data types and data sources, data warehouses are more critical than ever. The information in a well-designed data warehouse contains the corporate memory of an organization and the context to make sense of other data. For example, the contextual value of your business products and locations will make social data from systems such as Apache Hadoop much more relevant.

In addition, when new data is collected that provides insight, either from new sources or from instrumented devices, the ability to store it for later reference is critical. Recording analyses and point-in-time reports in an organized system provides this ability.

It is no surprise, then, that the data warehousing market continues to increase in popularity, growing at double digits year after year. The volume of data being generated from multiplying sources makes these technologies a must-have and must-leverage for virtually all organizations.

IBM Cognos Business Intelligence provides a proven enterprise BI platform with an open-data access strategy. It provides customers with the ability to pull data from various data sources, package it into a business model, and make it available to consumers in a variety of interfaces that are suited to the task, using business language that is relevant to consumers.

Cognos Dynamic Cubes complements the existing query engine, and extends Cognos scalability to enable speed-of-thought analytics over terabytes of enterprise data, without being forced to rely on a new data-warehousing appliance. This capability adds a new level of query intelligence so you can unleash the power of your large enterprise data warehouse.
Figure 1-1 on page 3 shows how Cognos Dynamic Cubes is tightly integrated into the Cognos BI stack, and its data can be surfaced through any of the Cognos interfaces. This way allows existing customers to integrate this technology into their application environment without affecting existing users, who are already familiar with interfaces such as Report Studio, Business Workspace, and Business Workspace Advanced (previously named Business Insight and Business Insight Advanced).

1.1.1 High performance interactive analysis

Cognos Dynamic Cubes is a technology meant to solve a specific but growing business problem: enabling high-performance interactive analysis over terabytes of data contained in an enterprise data warehouse (EDW).

As data volumes grow, analyzing that data with speed-of-thought performance can be challenging. Even with modern data warehouse technology, some operations require significant computation or data movement. This computation or movement creates delays and reduces the satisfaction of business users who want to perform these analyses.

Various ways exist to accomplish performance over large volumes of data. From self-contained cubes to large in-memory appliances, different vendors are employing variations of similar methodologies to give business users timely response times.

The Cognos Dynamic Cubes technology aims to give maximum flexibility in how memory is leveraged to accelerate interactive analysis over terabytes of data, giving you the ability to evolve your deployments over time.
1.1.2 Enterprise data warehouses for analytics

Data warehouses are the recognized foundation for enterprise analytics. They enable an organization to bring together cleansed data from separate sources of input, both internal and external, such as from partners or suppliers. Instead of garbage in, garbage out information to support decision-making, a consistent and consolidated enterprise-wide view of data from a business provides the foundation to improve your business. Building upon a trusted information platform for analytics is a key contributor to long-term business health. Not only do data warehouses enable higher quality information, they are designed to enable high-performance data access for analytic style applications.

Cognos Dynamic Cubes technology helps to leverage the core strengths of an EDW and take the EDW to the next level of performance for analytics, while making the deploying and tuning easier and faster.

1.1.3 Data volumes and context in the Enterprise Data Warehouse

With social data generating petabytes per day, and instrumented devices becoming the norm, data volume growth is accelerating at an unprecedented pace.

Big data is a growing business trend, with data from unconventional sources having the potential to be business disrupters. However, before the power of these new sources can be fully used, we must understand what is happening within our own business. Understanding your own business is added value of a data warehouse, and why taking full advantage of these data holdings is a critical first step to using these new sources of data.

In addition, any organization that relies on instrumented infrastructures has an opportunity to maximize the efficiency of its operations. Analytics is key to accomplishing this type of optimization, leading to concrete business results.

With data volumes exploding, and growing urgency to make sense of this sea of data, using a data warehouse technology, and maintaining a connection to the data warehouse from your analysis tools helps to rationalize investments.

Easy visibility into enterprise data is a critical step on the journey to insight from various sources, traditional and emerging. Semi-structured or unstructured data will only make sense within the context of your business. Your data warehouse contains this context.

1.1.4 Architecture summary

The Cognos Dynamic Cubes technology is part of the IBM Cognos BI query stack, and is available with existing IBM Cognos entitlements. It provides a new and powerful tool to enable high performance analytics over large data warehouses.

Cognos Dynamic Cubes solution consists of a modeling tool (IBM Cognos Cube Designer), a dynamic cube object in the administration environment, which becomes the data source, and a wizard (Aggregate Advisor), launched from Dynamic Query Analyzer:

- IBM Cognos Cube Designer

  Cognos Cube Designer is a modeling tool that brings together best modeling principles from past successful modeling technology, with a modern and extensible architecture. The first step to deploying Cognos Dynamic Cubes is to model with the Cognos Cube Designer.
IBM Cognos Dynamic Cubes Server

After a dynamic cube is designed and deployed, it becomes available in the Cognos environment and acts as the data source to the interface layer for dynamic cube applications. It manages all aspects of data retrieval, and leverages memory to maximize responsiveness while giving you full flexibility to manage what is contained in memory and when you want to refresh in-memory data. You manage dynamic cubes in the Cognos administration environment.

A dynamic cube contains a number of in-memory elements to drive performance:

- **Metadata members**
  When you start a cube, all members are loaded in memory. This approach assures a fast experience as users explore the metadata tree. This approach also helps the server to maximize its ability to run the most efficient queries possible.

- **Aggregates**
  With Cognos Dynamic Cubes, you have the option to create in-memory aggregates. This way allows summaries or aggregates to be returned immediately when users navigate down from top-level summaries, such as sales per country, and start to drill down to lower levels, such as states. Each level requires a new total to be returned. When these totals or other aggregates are loaded in memory, they are returned instantly.

- **Data**
  When users run the system and run queries, individual data items are loaded in memory. Because security is applied on top of the dynamic cube, the reuse of data is maximized, allowing as many users and reports as possible to benefit from previous queries.

- **Results sets**
  When a report is run, it is saved in memory for later reuse. For commonly used reports, this way allows the engine to benefit from previous runs, giving users the fastest response time possible.

- **Expressions**
  To accelerate query planning, the engine saves expressions that are generated when queries are run. In this way, query planning can be accelerated for future queries when expressions can be reused.

With in-memory assets that can include aggregates, expressions, results, and data, Cognos Dynamic Cubes technology provides the tools to minimize user wait-times, ensuring the best possible user experience.

Aggregate Advisor (part of IBM Cognos Dynamic Query Analyzer)

After a dynamic cube is deployed in the environment, it becomes available for reporting and analysis. The server maintains logs to understand how the system is being used, and can use these logs to help optimize aggregates, both in-memory and in-database.

To create aggregates, you use the Aggregate Advisor, launched from Dynamic Query Analyzer (DQA), which is a diagnostic tool to help you better understand performance for Dynamic Query and Cognos Dynamic Cubes. The Aggregate Advisor scans cube definitions and usage logs, and recommends aggregates to improve performance. This approach helps to more easily address specific performance problems.
Figure 1-2 shows the lifecycle of a dynamic cube.

1.2 Skillset requirements for IBM Cognos Dynamic Cubes

To take advantage of Cognos Dynamic Cubes, three specific sets of skills are required: modeling, Cognos administration, and infrastructure. A strong partnership with your database administrator (DBA) can also contribute to a successful deployment.

1.2.1 Modeling

Cognos Dynamic Cubes is based on dimensionally modeled relational data that is structured in a star or snowflake schema. As such, the key skill that is required to realize the full potential of Cognos Dynamic Cubes is data modeling, including knowledge of star or snowflake relational structures, and dimensional modeling.

1.2.2 Cognos administration

Because the dynamic cube technology is included with IBM Cognos BI, knowing how to administer the Cognos platform and its multitiered architecture is required so you can properly architect and deploy a Cognos solution that includes Cognos Dynamic Cubes. For example, to set up a large Cognos Dynamic Cubes application along with other large relational applications, advanced routing rules must be set up, which requires advanced knowledge of the Cognos platform.

1.2.3 Infrastructure design

Cognos Dynamic Cubes uses modern hardware capabilities to maximize performance over large volumes of data. Specifically, it relies on large memory spaces, and careful
management of Java heap sizes for the Dynamic Query service that is used for Cognos Dynamic Cubes.

Other considerations in large deployments require careful infrastructure planning. To share a Cognos deployment across many demanding applications, routing rules across multiple report servers are required to optimize each query service so that all users have a positive experience.

1.3 When to use Cognos Dynamic Cubes

Cognos Dynamic Cubes gives customers an additional tool in the query stack to handle a growing business problem: high-performance analytics over large data warehouses.

Certain conditions make Cognos Dynamic Cubes the correct choice, acknowledging that other variables, which are not listed here, might influence your choice:

- Data warehouse with star or snowflake schema
  A properly designed data warehouse is a recognized best practice to provide a well performing experience for users. As such, a properly designed data warehouse is a requirement to use Cognos Dynamic Cubes.

- A server with adequate memory
  Cognos Dynamic Cubes relies on memory to maximize its potential. Therefore, a server with enough memory to support the application can help to realize the full potential of Cognos Dynamic Cubes. A smaller server can be quite sufficient for small applications such as proof-of-concept (POC), but when you want to query a data warehouse, a proper server is required. For more information about hardware requirements, see *Understanding Hardware Requirements for Dynamic Cubes* from the business analytics proven practices website:

- A database with approximately 25 million or more fact table rows
  For small databases with fewer than 25 million rows of fact table data, there are easier ways to accomplish the task of presenting relational data dimensionally. For example, an OLAP Over Relational (OOR) model provides the ability to explore data dimensionally, and uses an existing Framework Manager model. However, OOR models cannot effectively handle data volumes beyond this size, while providing speed-of-thought interactive analysis performance.
  Cognos Dynamic Cubes is optimized to effectively scale up to terabytes of data, using aggregate-awareness and in-memory technology to accelerate performance.

1.4 System requirements

Cognos Dynamic Cubes includes IBM Cognos Business Intelligence entitlements, as of release 10.2. The standard system requirements that are described in the 10.2 documentation and conformance web page apply. However, similar to any large application deployment, more CPU power and memory are required to serve more than a small volume of data to a small number of users.

Because Cognos Dynamic Cubes relies heavily on in-memory technology to accelerate performance, consider using a powerful server with a significant amount of memory to realize the full potential of Cognos Dynamic Cubes.
A simple way to start thinking about memory that is required for your application is to apply the 10:1 ratio from a data warehouse footprint to a data warehouse summary table footprint, and then applying the same ratio to in-memory aggregate memory size. For example, a 100 GB warehouse translates to 10 GB of in-database summary tables, and 1 GB of in-memory aggregates. This information translates to approximately 4 GB for your Java virtual machine (JVM) heap to support this application. Although this is a good starting point, your specific application might have different requirements.

For more information about hardware requirements, see Understanding Hardware Requirements for Dynamic Cubes from the business analytics proven practices website: http://www.ibm.com/developerworks/analytics/practices.html

1.5 Comparison to other IBM cube technologies

Different data requirements require different data solutions. One data path cannot be proficient at solving widely different data problems. Therefore, IBM Cognos has technologies that are built to suit specific application requirements. Table 1-1 is intended to help you better understand the primary use case for each technology. However, carefully consider your individual application requirements when you make a decision.

Table 1-1 Use cases for Cognos Dynamic Cubes

<table>
<thead>
<tr>
<th>Cube technology</th>
<th>Primary use cases</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TM1</strong> In-memory cube technology with write-back support</td>
<td>▶ Is optimal for write-back, what-if analysis, planning and budgeting, or other specialized applications.  &lt;br&gt;▶ Can handle medium data volumes. Cube is run 100% in memory.  &lt;br&gt;▶ Aggregation occurs on demand, which can affect performance with high data and high user volumes.</td>
</tr>
<tr>
<td><strong>Dynamic Cubes</strong> In-memory accelerator for dimensional analysis</td>
<td>▶ Is optimal for read-only reporting and analytics over large data volumes.  &lt;br&gt;▶ Provides extensive in-memory caching for performance, backed by aggregate awareness to use the power and scalability of a relational database.  &lt;br&gt;▶ Star or snowflake schema is required in underlying database (use to maximize performance).</td>
</tr>
<tr>
<td><strong>PowerCubes</strong> File-based cube with pre-aggregation</td>
<td>▶ Is optimal to provide consistent interactive analysis experience to large number of users when the data source is an operational or transactional system, and a star or snowflake data structure cannot be achieved.  &lt;br&gt;▶ Pre-aggregated cube architecture requires careful management using cube groups to achieve scalability.  &lt;br&gt;▶ Data latency is inherent with a pre-aggregated cube technology, where data movement into the cube is required.</td>
</tr>
<tr>
<td><strong>OLAP Over Relational (OOR)</strong> Dimensional view of a relational database</td>
<td>▶ Is optimal to easily create a dimensional data exploration experience over low data volumes in an operational or transactional system, and where latency must be carefully managed.  &lt;br&gt;▶ Caching on the Dynamic Query server helps performance.  &lt;br&gt;▶ Processing associated with operational or transactional system affects performance.</td>
</tr>
</tbody>
</table>
IBM Cognos Dynamic Cubes architecture

IBM Cognos Dynamic Cubes adds relational online analytic processing (ROLAP) capabilities to the IBM Cognos Dynamic Query Mode (DQM) server in version 10.2 of IBM Cognos, enabling reporting and ad-hoc analysis on multi-terabyte data volumes.

Cognos Dynamic Cubes uses its shared in-memory member and data caches, and its awareness of in-memory and in-database aggregates, to provide fast query performance for a large number of users.

This chapter contains the following sections:
- 2.1, “Dynamic cubes in the IBM Cognos environment” on page 10
- 2.2, “Dynamic cubes and the Dynamic Query Mode server” on page 24
- 2.3, “Cognos Dynamic Cubes caching” on page 27
- 2.4, “Dynamic cubes query processing” on page 32
- 2.5, “Using the database to evaluate set expressions” on page 34
- 2.6, “Management of Cognos Dynamic Cubes” on page 34
2.1 Dynamic cubes in the IBM Cognos environment

Dynamic cubes are in-memory OLAP containers that reside within a DQM server. Dynamic cubes are sourced from a star or snowflake data warehouse, stored in a relational database that is contained in any one of the supported relational databases. Figure 2-1 shows multiple computing nodes that host Cognos report and DQM servers, which support a dynamic cube that accesses a relational data warehouse.

![Figure 2-1 A high level representation of the Cognos Dynamic Cubes architecture](image)

Dynamic cubes are modeled in the IBM Cognos Cube Designer. After modeling is complete, cube models are then published to the IBM Cognos Content Manager (CM) as a Cognos Dynamic Cubes data source. From within the IBM Cognos Administration console, an administrator can assign a dynamic cube to one or more dispatchers and configure the properties of the cube on each dispatcher, for example, defining the amount of space that is allotted for in-memory aggregates.

After a cube is assigned to a dispatcher and configured, it can be started, at which time the DQM server loads the cube model from CM and loads all dimensional members into its member cache. Then at this time, it is made available for processing of reports and analyses.
Dynamic cubes have many important features and capabilities that are best obtained through the coordination of database administrators and designers, business intelligence (BI) modelers, Cognos administrators, and report authors. This section provides an overview of the various aspects of IBM Cognos Dynamic Cubes in an attempt to show how dynamic cubes are designed and optimized across a set of coordinated activities.

### 2.1.1 Dynamic cubes and the data warehouse

A dynamic cube provides OLAP access to a star or snowflake data warehouse to Cognos BI users. IBM Cognos Dynamic Cubes does not provide support for data warehouses that do not conform to the star or snowflake topology, nor do they provide access to complex transactional databases.

The goal of IBM Cognos Dynamic Cubes is to provide quick response to reports and analyses on large volumes of data. To accomplish that goal, it is important that the queries that are posed to a relational database conform to a simple set of known patterns that are easily optimized by relational databases. Star and snowflake topologies are well documented as the best approach for modeling data warehouses, and are consequently supported by specific query optimization by nearly all major database vendors.

As with any other relational data source, the database must be defined in Cognos Administration as a data source, specifically with JDBC connectivity (required for all relational data sources that are accessed by the DQM server).
Figure 2-3 shows the data source connection with JDBC connectivity.

![Data source connection with JDBC connectivity](image)

In addition to a star or snowflake data warehouse, Cognos Dynamic Cubes is able to take advantage of any relational tables that contain pre-aggregated data that represents the value of measures from the fact table, summarized by members from one or more dimensions at a level of aggregation higher than that represented by the fact table. Cognos Dynamic Cubes is able to route queries for measure values, if appropriate, to these aggregate tables to improve query performance.
Figure 2-4 shows routing of a query to a summary table.

![Figure 2-4 Routing of a query to a summary table](image)

### 2.1.2 Modeling dynamic cubes

Cognos Cube Designer is used to model dynamic cubes on star or snowflake data warehouses that are stored in a relational database. Cognos Cube Designer analyzes a cube model as it is being constructed to ensure the underlying data warehouse is truly a star or snowflake, and raises exceptions if it recognizes relationships in the data that contradict constraints of either data topology.

A Cognos Cube Designer project can contain one or more namespaces. A namespace can contain a collection of dimension definitions which can then be used to model one or more cubes within the namespace. A dynamic cube is associated with a single fact table in the underlying data warehouse and contains dimensions defined either in the encompassing namespace, or within the cube itself, that is, dimensions which are specific to a particular cube.
Figure 2-5 shows metadata objects in Cognos Cube Designer.

Another possibility in Cognos Cube Designer is to model virtual cubes, which represent a merging of the members and data of two other cubes, either of which may be a Dynamic Cube or another virtual cube. With virtual cubes, users can analyze data that encompasses multiple fact tables, such as sales and inventory data.

Cognos Dynamic Cubes supports the following features, all of which are modeled in the Cognos Cube Designer:

- Dimensions
- Multiple hierarchies per dimension
- Level attributes
- Parent-child (recursive) hierarchies
- Time hierarchies, including automatic creation of relative time members
- Calculated (dimensional) members and measures
- Member security
- Aggregate cubes
- Virtual cubes

Cognos Cube Designer also provides a member browser, which can be used to validate the structure of a hierarchy as it is being modeled.
Figure 2-6 shows viewing members in the Cognos Cube Designer member browser.

Cognos Cube Designer also allows a modeler to validate all or part of a cube model to identify modeling problems as they occur. Figure 2-7 shows how to validate an object.

After a cube is modeled in Cognos Cube Designer, the modeler can then publish it to CM so that it can be configured and used. In a development environment where the expected (and
repeated) cycle is to model, publish, test, and remodel, Cognos Cube Designer provides an extended publishing feature that will publish, configure, and start a cube, all from a single click.

Figure 2-9 shows extended Cognos Cube Designer publish options.

![Extended Cognos Cube Designer publish options](image)

**Figure 2-9  Extended Cognos Cube Designer publish options**

### 2.1.3 Dynamic cubes as data sources

When a dynamic cube is published to CM from Cognos Cube Designer, it appears in the list of data sources as either a dynamic or virtual cube, each of which has its own specific icon to distinguish it from other data sources that are defined in CM.

Figure 2-10 shows the Dynamic Cube data source among other connections.

![Dynamic Cube data source among other connections](image)

**Figure 2-10  Dynamic Cube data source among other connections**

Dynamic and virtual cube data sources not only visually differ from other CM data sources, they also have their own specific set of properties. Of the dynamic cube properties, the most important is the access account, which is the single account that is used to access the underlying relational database of the cube.
Figure 2-11 shows dynamic cube data access property.

![Figure 2-11 Dynamic cube data access property](image)

A dynamic cube also has a child model element that, if security is defined for a cube, contains a collection of security views to which users, groups, or roles can be assigned.

Figure 2-12 shows dynamic cube security views.

![Figure 2-12 Dynamic Cube security views](image)

By being presented in CM as a data source, a dynamic cube can be included in a Framework Manager model as an OLAP data source, allowing it to be part of multidata source package.

### 2.1.4 Configuring dynamic cubes

Publishing a dynamic cube to CM as a data source does not make a cube suddenly accessible to users. At this point, all that happened is that the metadata definition of a cube was published so that it is accessible to other parts of Cognos. The next step to making a cube available to users is to configure a dynamic cube for one or more dispatchers within a Cognos installation. Configuring a dynamic cube for a dispatcher accomplishes two steps:

- It identifies a specific dispatcher on which a cube can be active.
- It defines the operational characteristics of a cube on that dispatcher.
In most circumstances, dynamic cubes are assigned to specific dispatchers that are running on hardware specifically allocated to house dynamic cubes because of their memory and CPU requirements, which are in excess of what is normally required for a DQM server. One consequence of this approach is the necessity to use advanced routing rules to route requests for dynamic cubes to the appropriate dispatchers.

Figure 2-13 shows multiple cubes deployed across multiple servers.

Each cube that is assigned to a dispatcher has its own set of properties. Several of the properties, such as data and aggregate cache sizes, must be assigned non-default values based on an evaluation of the data warehouse and the expected use of a cube. Other properties may be assigned values during the course of the development and production lifecycle of a cube, for example, the enabling of workload logging.

Because dynamic cubes are created within the DQM server, there are also QueryService properties, notably those related to the JVM heap, that must be assigned values that take into account the presence of dynamic cubes.
Chapter 2. IBM Cognos Dynamic Cubes architecture

Figure 2-14 shows query service settings for dynamic cubes.

![Query service settings for dynamic cubes](image)

After a dynamic cube is configured, it is still not accessible to users. Dynamic cubes must be either explicitly or implicitly started before they can be accessible to users, and there must also be at least one package available to users to access a cube, as with any other data source defined within CM.

### 2.1.5 Dynamic cube management

A cube that was configured for a dispatcher can be started in one of four ways:

- **Starting the QueryService.**
  - By default, cubes will start automatically when the QueryService starts. This feature can be disabled on a per-cube basis.
- **Starting the cube from IBM Cognos Administration.**
  - In the Status section of Cognos Administration, an administrator can view the cubes that are assigned to the QueryService under a single dispatcher, along with the status of the cubes. From this dialog, performing a series of actions against a cube, including starting a cube, is possible.
- **Creating and scheduling a ROLAP administrative task.**
- **Creating an SDK application that starts a cube.**

After a cube is started, it immediately begins to retrieve all of the data from the dimension tables in the data warehouse that are represented in a cube model, and stores this information as an in-memory representation of all of the hierarchies of a cube. After this information is loaded, a cube is available for access from the various BI studios.

A cube may also have in-memory aggregates defined. These are loaded after a cube is available for access and the status of this activity is visible in the metrics for a cube, in Cognos Administration.

If a cube has an associated startup trigger, this too executes after a cube is available for use.
Figure 2-15 shows assigning a trigger to a cube.

![Figure 2-15 Assigning a trigger to a cube](image1)

The intent of a trigger is to run one or more reports, analyses, or both, to preload data cache of a cube, ensuring that the data is available in the cache for subsequent users.

Several operations can be done against a cube after it is running:

- Stop
- Restart (stop, start)
- Refresh member cache
- Refresh data cache
- Refresh security
- Clear workload log
- View recent messages

Figure 2-16 shows the menu for administering a cube.

![Figure 2-16 Menu for administering a cube](image2)

### 2.1.6 Virtual cubes

Cognos Dynamic Cubes supports the definition of virtual cubes, which are defined in Cognos Cube Designer as the merge of two cubes, either of which may be a dynamic (base) cube or another virtual cube. In this manner it is possible to model multi-fact cubes that span multiple fact tables and business areas of interest.
Figure 2-17 shows the organization of virtual cubes.

With virtual cubes, dimensions, hierarchies, levels, members, properties, and measures from two cubes can be merged into a single cube, either automatically or manually on an object-by-object basis.

During query execution, queries that are posed to virtual cubes are decomposed into queries that are posed to each underlying cube and their results are merged by using a specified merge rule. Queries for data are posed to cubes only as appropriate. For example, if a measure is present in only a single cube, only that cube is queried for its value.
2.1.7 Aggregate Advisor

A key feature of Cognos Dynamic Cubes is its ability to take advantage of both in-database and in-memory pre-computed summaries. These pre-computed summaries can improve the performance of queries by orders of magnitude, providing the type of performance required for interactive reporting and analysis.

If you have pre-existing summary tables in your data warehouse, you can model them as aggregate cubes in the Cognos Cube Designer. When a cube is published and the cube is restarted, it will automatically route SQL queries to the summary tables when possible. For distributive measures (those whose aggregation rule is SUM, COUNT, MAX, or MIN), summary tables can be employed to compute summary values at higher levels of aggregation than that at which an aggregate cube is defined.

As useful as this capability is, one of the most difficult tasks when of pre-computed summaries is trying to determine what it is that should be pre-aggregated, especially in a large, multi-user environment that might involve hundreds or thousands of reports and analyses. The Cognos Dynamic Cubes Aggregate Advisor, available as part of the Dynamic Query Analyzer, performs this task.

The Aggregate Advisor can be used to suggest database aggregate tables, in-memory aggregate cubes, or both. The Aggregate Advisor makes use of a cube's model and statistics it gathers from the underlying data warehouse to determine which summary tables to suggest. However, it can also make use of workload log files that are generated from the execution of reports and analyses to make more accurate suggestions of what will optimize the performance of an application workload.
Figure 2-19 shows the Aggregate Advisor selection of workload information.

The workload log that is generated by the DQM server contains detailed information that can be used as filters to isolate a subset of workload to be optimized. In addition, the information in the workload log allows the Aggregate Advisor to determine which values to pre-aggregate that can be used across a wide range of reports and analyses, for example, aggregating data at the quarterly level and allowing this to be used to compute annual totals without pre-computing the annual levels.

In-database aggregate suggestions contain a detailed description of each aggregation it suggests, and a generic set of column descriptions and the database-specific SQL that can be used to populate the table. These suggestions can be used by a DBA to construct the necessary tables and the corresponding extract, transform, and load (ETL) scripts to build and maintain the tables.

In-memory recommendations are published to CM. After a cube is restarted, the aggregates execute the necessary SQL statements to retrieve the summarized values and place the values in its aggregate cache for subsequent use during query processing.

Virtual cubes cannot have an aggregate cache because they have no underlying database from which to obtain aggregate values.
2.2 Dynamic cubes and the Dynamic Query Mode server

Dynamic cubes are created within a DQM server. As such, an important concept to understand is how dynamic cubes exist within a DQM server and how administrative commands, metadata, and data requests are directed to a dynamic cube. This section describes the interaction between a report server and a DQM server as it pertains to Cognos Dynamic Cubes, the components of a dynamic cube within a DQM server, and finally, how commands and requests are directed within the DQM server to a dynamic cube.

2.2.1 Dynamic cubes and report server

A dispatcher may host, at most, only a single DQM server, which appears as the QueryService in the list of services associated with a dispatcher. All of the report servers that run under a dispatcher communicate with the single DQM instance through a single port. Thus, any dynamic cubes that are assigned to a dispatcher are all hosted within the single instance of the DQM server.

It is possible to configure a cube on more than a single dispatcher, which can provide user scalability. Note, however, that each instance of the DQM server hosts a separate version of a cube. The interaction of a cube with the underlying relational database, and the management of its caches, is distinct from that of any other instance of the cube on other dispatchers. Consequently, distributing a cube, though it does allow for more users to access a single, logical cube does incur some additional overhead:

- Each computing node to which a dynamic cube is configured must have the memory and CPU cores necessary to support the cube and its expected user volume, taking load balancing into account.
- If a dynamic cube is deployed across multiple nodes, the load on the underlying relational database can be multiplied notably during the loading of in-memory aggregates if start, restart, or data cache refresh operations are performed at the same time across the multiple instances of the cube.
The data caches of the different instances of a cube are not synchronized, so if a user executes a query on one machine and another, query response can fluctuate.

The underlying database might encounter additional load because each instance of a cube is required to load data independently of the other node.

The dispatcher is not knowledgeable of dynamic cubes, therefore it does not account for whether a cube is available under a particular dispatcher when dispatching a request for execution. Consequently, extra steps are required to ensure cubes are taken on and off line on individual dispatchers to ensure users do not encounter error messages to the effect that a cube is unavailable.

### 2.2.2 Dynamic cubes within the Dynamic Query Mode server

Dynamic cubes exist in memory within the DQM server. A dynamic cube, when running within DQM, involves the following cache:

- **An in-memory member cache**
  
  The members of each hierarchy are retained in memory that allows for the traversal of parent-child relationships, and level relationships. This cache is used during DQM query planning to validate member-unique names, and during query execution to perform member and set operations, which do not require the evaluation of measure values, for example, `MEMBERS([Level Name])`.

- **An in-memory aggregate cache**
  
  If in-memory aggregate recommendations from the Aggregate Advisor are published to CM and the cube that is configured to assign memory for in-memory aggregates, the cube will retain the aggregate values in separate *cubelets* in a separate cache, up to the amount of space specified in the configuration for the cube.

- **An in-memory data cache**
  
  Any queries that retrieve data from the underlying database, or further aggregate data from the aggregate cache, are stored as cubelets in a separate cache.

- **An in-memory security cache**
  
  Every unique combination of security views is retained in memory to provide efficient application of security filters to queries.
Figure 2-21 shows the various caches in a Dynamic Cube.

These structures are retained in a cube construct within, but separate from, the remainder of DQM. The DQM server interacts with a Dynamic Cube in three ways:

- Administrative commands
- Metadata requests
- Query execution

Administrative commands may be received either through Java Management Extensions (JMX) or BI bus commands (typically from Cognos SDK applications). These administrative commands are directed by DQM to the Cognos Dynamic Cubes management interface, which in turn directs the requests to the specified cube to perform such operations as stop, start, and so on.

Metadata requests typically are posed by either BI client applications to populate the metadata browser, or by the DQM query planner, which interacts with a cube to obtain various pieces of metadata to validate a metadata references of a request.

The DQM server converts queries that it receives into MDX queries, which are then run by DQM's own MDX engine. During the running of an MDX query, the MDX engine might need to perform member operations such as obtaining the children of a member, and does so through the metadata interface of a cube. The MDX engine might also require measure values, which it will request through an interface called the query strategy.
Both metadata and data requests apply any security defined in the model before returning member metadata and before retrieving measure values.

Data requests, through the query strategy of a cube, make use aggregate and data caches of a cube.

Figure 2-22 shows the command dispatch within the DQM Server.

![Figure 2-22 Command dispatch within the DQM Server](image)

A single DQM server may host one or more dynamic cubes at one time. If a DMQ server supports a virtual cube, all of the cubes upon which the cube is based must also reside on the same server. Because a dispatcher can be assigned to only a single server group, all cubes that reside within a DQM server are implicitly part of the same server group, and all packages that reference any of the cubes on the server must all route their requests to this single server group.

### 2.3 Cognos Dynamic Cubes caching

The basis for the performance of Cognos Dynamic Cubes is its various in-memory caches and its use of database summary tables. The DQM server supports 64-bit machine architectures, and, as such, provides Cognos Dynamic Cubes access to large amounts of memory.

The aggregate and data caches in Cognos Dynamic Cubes not only provide fast query response, but also help to alleviate the processing load on the underlying relational database, because Cognos Dynamic Cubes does not have to re-retrieve data that is held in its various caches.
The result set and expression caches, in turn, reduce the amount of processing that occurs within the MDX engine within the DQM server, helping to reduce spikes in resource usage.

Cognos Dynamic Cubes also makes use of available processing cores to do costly operations in parallel, such as loading hierarchy tables, or processing query result sets.

Cognos Dynamic Cubes uses five caches, each with a separate purpose. The following sections describe these cache types.

### 2.3.1 Result set cache

The result set cache exists at a layer directly above the MDX engine with the DQM server. The result set of each MDX query executed by the engine is stored within the on-disk result set cache. Each MDX query that is planned for execution is first searched for in the result set cache. So, if the query was previously executed, and the combination of security views of the user who executed the query matches that of the current user, then the result set of the query is obtained from the result set cache. Thus, the query is not executed other than the first time.

Figure 2-23 shows the result set cache.

![Diagram of the result set cache]

The cache is aware of queries that are currently executing, so it is not possible for a query to be executed twice. If necessary, a second request will wait for the first to complete execution. Also, the result set cache is flushed when the data cache is refreshed to ensure the data and result set caches are synchronized.

Although stored on disk, the result set cache is capable of providing extremely fast report execution because, in many cases, the output of a report is small compared to the amount of data that is processed to produce the final result set.

The result set cache, besides sharing report execution results between users, also speeds drill up operations, because the drill up involves the re-execution of a query.
2.3.2 Expression cache

A set within the context of MDX contains one or more tuples. A tuple contains a member from one or more dimensions, where one of those dimensions can be the measures dimension. Tuples within a set in MDX each contain a member from the same set of dimensions and in the same order.

The expression cache is used by the MDX engine to cache the result of set expressions that operate on large sets and output a much smaller set for subsequent reuse, either within the same query or within subsequent MDX queries processed by the MDX engine. The savings from the use of cache expression results can be significant, given that in some cases a set can contain millions of tuples and only output five or ten tuples.

As with the result set cache, the intermediate results that are stored in the expression cache are security-aware and are flushed when the data cache is refreshed.

Many MDX functions can operate on large sets, which can be characterized as those that process a large set of tuples as input and, do the following tasks:

- Perform a simple operation to return a smaller set of tuples (for example, HEAD).
- Fetch the corresponding values, and return a single value (for example, AGGREGATE).
- Fetch the corresponding values, and return a smaller set of tuples.

The expression cache feature focuses on those MDX function which process a large input set of tuples and return a smaller set of tuples. Specifically, the following MDX functions are supported by the expression cache:

- TOPCOUNT
- BOTTOMCOUNT
- TOPPERCENT
- BOTTOMPERCENT
- TOPSUM
- BOTTOMSUM
- FILTER

In all, but the following, circumstances are the results of the functions (in the previous list) that are cached:

- If any of the inputs to a function contains a calculated member, named set or inline named set.
- If the report from which the MDX statement was generated has a prompt variable defined and it is referenced within the expression.
- Expressions cached by the engine with an internal, default value. The use of a default value is an optimization within the MDX engine that can conflict with the use of the expression cache.

Each entry in the expression cache has three components that uniquely identifies it:

- The text of the MDX expression.
- A security key that identifies a combination of security views.
- The context in which the expression was evaluated, expressed as a tuple.

Because the expression cache is stored in memory and is so closely associated with the data cache (see 2.3.5, “Data cache” on page 31), the expression cache is actually stored within the space allotted to the data cache.
2.3.3 Member cache

The hierarchies defined in a cube model are all loaded into memory when the cube starts. After all hierarchies are loaded, the following actions occur simultaneously:

- The cube state is changed to available and it is able to respond to metadata and data queries.
- If a cube has a startup trigger defined, it is executed.
- If a cube has in-memory aggregates defined, and space allotted for them, the queries to retrieve the data are executed.

Members of each hierarchy are retrieved by running a SQL statement to retrieve all of the attributes for all levels within a hierarchy, including multilingual property values, and are stored in memory. The parent-child and sibling relationships inherited in the data are used to construct the hierarchical structure in memory.

All metadata requests by the client application, the DMQ query planning engine, and all member functions (for example, PARENT, CHILDREN) are serviced from the in-memory cache. Cognos Dynamic Cubes will never pose a query to retrieve member attribute values after the member cache is loaded.

Queries that are posed by Cognos Dynamic Cubes to retrieve data to populate its data cache make use of the level key values stored in the member cache as the basis for constructing filters in the SQL queries that are posed to the underlying relational database.

2.3.4 Aggregate cache

IBM Cognos Dynamic Cubes supports two types of pre-computed aggregate values: those stored in database tables and those stored in its in-memory aggregate cache. Though database tables that contain aggregate values (referred to as summary or aggregate tables) may pre-exist within a data warehouse, the Aggregate Advisor within the IBM Cognos Dynamic Query Analyzer (DQA) can be used to suggest additional aggregate tables to improve query performance.

In addition to suggesting database aggregate tables, the Aggregate Advisor can also suggest a collection of in-memory aggregates that will also improve query performance. One primary difference between the two types of aggregates is that the in-memory aggregates do not require the involvement of the relational database DBA; the Aggregate Advisor recommendations can be stored in CM and take effect the next time a cube is started.

As with in-database aggregate tables, in-memory aggregates contain measure values that are aggregated by the members at the level of one or more hierarchies within the cube. These values can be used to provide values at precisely the same level of aggregation. In the case of measures with distributive aggregation rules (SUM, MAX, MIN, and COUNT), in-memory aggregates can be used to compute values at a higher level of aggregation.

**Difference between aggregate types:** *In-database* aggregates are usually built during a predefined ETL processing “window.” *In-memory* aggregates are built during cube-start, which suggests that cubes should be started during, or immediately after, the ETL process. The number of queries posed concurrently to populate the in-memory aggregate cache can be controlled by an advanced property of Cognos Dynamic Cubes to ensure the underlying relational database is not saturated with concurrent requests computing summary values.
Figure 2-24 shows the loading of in-memory aggregates.

Queries that are issued to retrieve aggregate cache values are planned by Cognos Dynamic Cubes in the same manner as when data is required to populate the data cache. As a consequence, they are able to take advantage of any available in-database aggregate tables.

The size of the aggregate cache that is specified in the properties of a dynamic cube is simply a maximum. The aggregate cache, however, is loaded on a first-come basis; if an aggregate result will not fit into the cache, it is discarded. If the cache is full, no more queries are issued to retrieve aggregate values. An aggregate cache size of zero disables the aggregate cache.

2.3.5 Data cache

The data cache contains the result of queries posed by the MDX engine to a dynamic cube for data. Each such query is posed as a set of tuples, each of which specifies a single measure for which a value is required. As described elsewhere, a request for data from the MDX engine can be satisfied by data that exists in the data or aggregate cache. However, if data is not present in either cache to satisfy the request for data, or if only part of it can be retrieved from the cache, dynamic cubes will obtain the data either by aggregating data in the aggregate cache, or retrieving data from the underlying relational database. In either case, the data obtained is stored in the data cache as a cubelet, a multi-dimensional container of data. Thus, the data cache is a collection of such cubelets that are continuously ordered in such a manner as to reduce the time that is required to search for data in the cache.

Data within the data cache is shared among all users. Security is applied in a layer above the data cache and ensures that users are provided access only to data to which they have been granted, or to which they have not been denied, access.

Both dynamic (or base) cubes and virtual cubes may be assigned a data cache. If a cube is not directly accessible by users, that is, it has no packages associated with it, a data cache is not necessary. However, if a cube is being used to retain historic data as part of a virtual cube, then a data cache on the base cube is useful because it ensures that the historic data,
which is not updated regularly, remains in memory even when the latest data in another base cube is updated.

2.4 Dynamic cubes query processing

This section describes the types of query processing that can be used: metadata and OLAP.

2.4.1 Metadata query processing in the Dynamic Query Mode server

A dynamic cube maintains all of its metadata in memory, including information about the following objects:
- Cube
- Dimensions
- Hierarchies
- Levels
- Members
- Properties
- Measures (including calculated measures)

The majority of metadata requests, either those posed by a client application (to populate a metadata browser) or those issued by the DQM query planner (to validate metadata references within a query) are managed by DQM's metadata framework (MFW) component, which in turn obtains metadata from various dynamic cubes.

When searches are made for members within a hierarchy or level within a metadata browser, these searches are performed by DQM's MDX engine, which in turn obtains the member information directly from the underlying dynamic cube.

2.4.2 OLAP query processing in the Dynamic Query Mode server

Reports and analyses are processed by the report server, which in turn issues one or more corresponding queries to the DQM server for processing. These requests are posed in XML. The DQM server, upon receiving a command for execution, first reads the command type to determine how to dispatch the command within the server. A query is first dispatched to the query planner, and if that is successful, is then dispatched to the query execution engine for execution of the final query plan.

As described earlier, the query planner interacts with the DQM metadata framework component during query planning to validate (bind) metadata references within a query, including items such as dimension and member references. The MFW4J component in turn interacts with the Dynamic Cubes OLAP data provider (ODP) metadata interface to obtain metadata. This component in turn retrieves the necessary metadata from the cube itself.

The outcome of successful query planning against any OLAP data source in DQM is an MDX query. In the case of Cognos Dynamic Cubes, the query is executed within DQM by using its own MDX execution engine. As mentioned before, during query execution, the MDX engine typically interacts with both the metadata and query strategy interfaces of Cognos Dynamic Cubes to obtain metadata and data from a dynamic cube.

The final output of the execution of the MDX query is a cross tabular result set, which is then processed by the result set component of DQM, which converts the query results into a format amenable to the report server.
The process is described in the following steps:

1. The query is searched for in the DQM plan cache. If the query is found, the run tree that is stored in the cache is used and no query planning occurs.

2. Otherwise, query planning is performed on the query, the result of which is an MDX run tree.

3. Cognos Dynamic Cubes searches the result set cache for the MDX query. If it is found and the security views of the result and the current user match, the result set is returned to DQM for further processing.

4. Otherwise, the MDX engine executes the run tree, executing nodes as it walks the tree.

5. Members and member metadata is retrieved from the metadata cache of the cube to satisfy particular run tree nodes.

6. For particular nodes such as FILTER and TOPCOUNT, the expression cache is checked for previously computed results. If none are found, the expression is evaluated and if applicable, the result set is added to the expression cache.

7. When a run tree node requires data from a Dynamic Cube, a request is made to the cube.

8. Cognos Dynamic Cubes first checks the data cache for cubelets, which contain the data required to satisfy the request for data.

9. If all of the data required is found, a result set is returned to the MDX engine containing all of the requested data.

10. Otherwise, Cognos Dynamic Cubes examines the aggregate cache. The cubelets within the aggregate cache are used only if they are able to provide values for all of the tuples within the remaining data request, that is, the tuples must contain members in each hierarchy which are from the same level.

   A cubelet from the aggregate cache can be used if it matches the levels of the tuples for which data is requested, or if the data can be rolled up to the level requested. If aggregation is required, the resulting values are stored in the data cache.

11. If there are still tuples for which data is not retrieved, Cognos Dynamic Cubes queries the underlying relational database for the values. The tuples will be organized into groups of identical levels on a per hierarchy basis.

12. Cognos Dynamic Cubes examines its metadata to determine if there is an aggregate table in the database that can be used to obtain the required values for each query group and will direct queries to aggregate tables or the original fact table, as required.

13. Data retrieved from the database is stored in data cache and then added to the result set that is returned to the MDX engine.

The flow from step 5 through step 12 is repeated recursively, depending on the structure of the run tree, until the final result is computed.

Each cache within a cube attempts to provide data values for the unresolved portion of a query after the previous cache finishes attempting to provide data. The caches at the beginning of this processing flow are coarse grained, smaller, and provide less reuse and sharing. For example, an entry in the result set cache contains only the output of an MDX query and can be shared only by users with the same combination of security views as the user who first executed the query. Those at the end of the flow are fine grained, larger, and provide more reuse and sharing. For example, the aggregate cache might contain gigabytes of data, but it can be shared among all users regardless of their assigned security privileges.

Because the aggregate cache is preloaded into memory, its contents are (typically) based upon prior workload. Because its data can be aggregated, the aggregate cache can be
thought of as the primary cache of a dynamic cube, but not of virtual cubes, which do not support aggregate caches.

If a cube is not assigned an aggregate cache, the data cache becomes the primary cache. Otherwise, it plays a secondary role, caching the values that do not directly hit the aggregate cache.

### 2.5 Using the database to evaluate set expressions

For particular MDX functions, given a sufficiently large input set, Cognos Dynamic Cubes pushes the evaluation of the expression to the underlying database for execution. Without this feature, those MDX functions, such as TOPCOUNT, retrieve large sets of data from the database and perform the function in the MDX engine. The transfer of the large data sets, which can be in the range of millions of rows of data, can degrade query performance.

The intent of query pushdown is to exploit the proximity of RDBMS to the data, allowing the operation to be performed without requiring the transfer of the data to the MDX engine. The effect is similar to a cache, in that it provides faster response time and has lower and more consistent resource requirements.

Query pushdown is supported for the FILTER, TOPCOUNT, and TOPSUM functions, but is not supported in the following cases:

- Sets of members from different levels of the same hierarchy
- Parent-child hierarchies
- Semi-aggregate measures
- Measures with an UNKNOWN aggregation rule
- Complex numeric expression in the 'count' parameter of the TOPCOUNT function
- Calculated members/measures in the numeric expression
- Sets containing both negative values and nulls

Pushdown is not supported for queries that are posed to virtual cubes.

### 2.6 Management of Cognos Dynamic Cubes

DQM accepts most of its commands from the report servers by using an XML protocol via a single port open for all such requests, including report execution and metadata retrieval. DQM, as a Cognos service, also accepts BI bus requests, notably those that are used to manage dynamic cubes. In addition, DQM also supports JMX, which is used to implement portions of Cognos Administration interface for managing requests.

When any of the cube management requests are received by DQM, they are directed dynamic cubes, which then dispatches the request (command) to the specified cube. The following commands, to name several, are included:

- Start cube
- Stop cube
- Restart cube
- Refresh metadata cache
- Refresh data cache
- Refresh security
- Clear workload log
- View messages
Cognos Dynamic Cubes ensures that concurrent user report executions behave correctly when any of these commands occur, and ensures that concurrent administrative commands are handled in a logical fashion. For example, a request to refresh the metadata cache immediately fails if a cube is in the process of being stopped.

When member cache of a cube is refreshed, queries continue to execute by using the existing member cache while a new member cache is constructed in the background. When the new cache is finally constructed, all new queries reference the new member cache. Queries that are in the midst of running when the new cache is made available continue to use the old cache. The old cache is discarded after all queries that reference it have completed running.

Though this approach does require enough memory to store two versions of the member cache in memory at one time, the effect of this is that user queries continue to operate normally and with consistent performance while the member cache is being refreshed.

Refreshing the data cache is slightly simpler. Because the data cache is constructed on demand, a request to refresh the data cache causes a new cache to be constructed. All new queries use (and populate) the new data cache; queries that were executing at the time of the refresh continue to use the old cache. The old data cache is discarded when all queries which reference it have completed execution.

Because the values in the data cache are aggregated relative to the members in the member cache, a refresh of the member cache implicitly causes a refresh of the data cache.

Similarly, when the data cache is refreshed, the expression cache (which is actually contained within the data cache) and the result set cache are implicitly flushed also, because they must be synchronized with the data cache. Also, the aggregate cache is flushed and the queries that are used to populate the aggregate cache are re-executed. While the queries that load the aggregate cache are executing, any user queries that are posed during that time are unable to take advantage of the in-memory aggregates, although the in-memory aggregates are made available after the data for a particular in-memory aggregate has been loaded.
Installation and configuration of Cognos Cube Designer and Dynamic Query Analyzer

This chapter helps you prepare your environment and install the IBM Cognos Cube Designer and Dynamic Query Analyzer products to allow using Cognos Dynamic Cubes in your Cognos BI environment.

For the most recent information about IBM Cognos Dynamic Cubes installation and configuration, see Dynamic Cubes Installation and Configuration Guide 10.2.0:
http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/nav/0_9

For the most recent information about Dynamic Query Analyzer installation and configuration see, Dynamic Query Analyzer Installation and Configuration Guide 10.2.0:
http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/nav/0_7

This chapter contains the following sections:
- 3.1, “Introduction” on page 38
- 3.2, “Cognos Cube Designer operating requirements” on page 38
- 3.3, “Cognos Dynamic Cubes hardware requirements for the BI Server” on page 40
- 3.4, “Installing IBM Cognos Cube Designer” on page 40
- 3.5, “Installing Dynamic Query Analyzer” on page 47
3.1 Introduction

Fully understanding the various components of IBM Cognos 10.2 and its architecture is important. All the components are described in the documentation that is included with the software. Also important is to review Chapter 2, “IBM Cognos Dynamic Cubes architecture” on page 9, which explains detail of the architecture of IBM Cognos Dynamic Cubes.

The remaining chapters in this book refer to Cognos Dynamic Cubes sample data. This sample data is available in the IBM Cognos 10.2 installation files and should be installed by the Cognos Administrator.

For information about installing the IBM Cognos Business Intelligence Samples, see “Install the IBM Cognos Business Intelligence Samples” at:

http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/topic/com.ibm.swg.ba.cognos.inst_cr_winux.10.2.0.doc/c_inst_installc8samples.html?path=0_12_23_1#inst_InstallC8Samples

For information about setting up the samples, see “Setting up the samples” at:

http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/topic/com.ibm.swg.ba.cognos.ug_cra.10.2.0.doc/c_settingupsamplesbi.html?path=2_2_4_5#SettingUpSamples

For the latest information regarding installation and configuration of components that are related to IBM Cognos Dynamic Cube, use this link and go to the release notes that are included for your version.

3.2 Cognos Cube Designer operating requirements

Before you use Cognos Dynamic Cubes, prerequisite components must be installed and running to ensure a successful implementation.

This section outlines the key components that are necessary before you install Cognos Cube Designer.

3.2.1 IBM Cognos 10.2 Business Intelligence Server

Cognos Dynamic Cubes is a component of IBM Cognos 10.2 Business Intelligence; therefore, it requires having an existing IBM Cognos 10.2 Business Intelligence Server in the network. Minimally, the Business Intelligence Server should have a gateway, an application tier, and a content store readily available.

The workstation where Cognos Cube Designer will be installed must have network access to the Business Intelligence Server. Authenticating in IBM Cognos Connection through a web browser is a good way to test the connectivity.

IBM Cognos 10.2 Business Intelligence Server is available in a 32-bit and 64-bit version. To be able to use Cognos Dynamic Cubes, the 64-bit version of the server must be installed.

Also, ensure JDBC connectivity is configured to your data warehouse from the BI server. Consult the Cognos BI Server Installation Guide.
3.2.2 IBM Cognos 10.2 Report Server

IBM Cognos 10.2 Business Intelligence Server provides a 32-bit and 64-bit version of the Report Server. By default, the Report Server will be set to use the 32-bit version only.

Because Cognos Dynamic Cubes is based on the Dynamic Query Mode, an important task is to enable the 64-bit version of the Report Server (as shown in Figure 3-1).

Complete the following steps to enable the 64-bit Report Server:

1. Go to the IBM Cognos Configuration.
2. In the Explorer panel, click Environment.
3. For the Report Server execution mode, select the 64-bit option, as shown in Figure 3-1.
4. Save the Cognos Configuration and restart the IBM Cognos Service.

![Figure 3-1 IBM Cognos Configuration Report Server execution mode (64-bit)](image)

3.2.3 IBM Cognos 10.2 Framework Manager

IBM Cognos 10.2 Framework Manager must be installed as a prerequisite before the installation of Cognos Cube Designer. Both software components are installed on the intended designer workstation in a shared common root directory. Cognos Cube Designer will use the configuration and connectivity of Framework Manager.

3.2.4 Java Runtime Environment Libraries

Cognos Dynamic Cubes requires IBM Java 6 SR 10 FP1 or newer version of the Java Runtime Environment (JRE) libraries. IBM Cognos 10.2 Business Intelligence Server on Microsoft Windows automatically installs an IBM version of the JRE.
3.2.5 Supported relational databases

For the most recent information regarding data warehouse platforms (relational databases) that are supported for use with Dynamic Cube Designer, go to:


Select your Cognos Business Intelligence version, for example, 10.2. Locate the table with heading “Dynamic Query Mode.” Under this table, the “Relational” section lists the relational databases supported.

3.2.6 Cognos Cube Designer supported operating systems

For current version information of supported operating systems for Cognos Cube Designer, see the IBM Support website that describes supported environments. Select your Cognos Business Intelligence version and review the section for Framework Manager and Data Manager Designer


3.3 Cognos Dynamic Cubes hardware requirements for the BI Server

For more information about hardware requirements, see Understanding Hardware Requirements for Dynamic Cubes in the Business Analytics proven practices web location at:


3.4 Installing IBM Cognos Cube Designer

The most common method of installation for IBM Cognos Cube Designer is called attended installation. This method involves the use of the Cognos Cube Designer Installation Wizard which guides you through all of the necessary steps to correctly install the application on the workstation.

This book covers only the attended installation method. For instructions to use the unattended installation method, see the installation guide in IBM Cognos 10.2.
Complete the following steps to install Cognos Cube Designer.

1. Start the installation by going to the location of the installation files in the root folder where the iso tar file is extracted. Right-click Autorun.inf and select Install (Figure 3-2).

![Figure 3-2  Run Autorun.inf](image)

2. Depending on the security level that is set at the workstation, you might be prompted to confirm the correct file (Figure 3-3). Click Run.

![Figure 3-3  Security warning](image)
An IBM welcome window opens (Figure 3-4), followed by the first option of the Installation Wizard (Figure 3-5).

Figure 3-4   IBM Cognos InstallStream welcome window

3. Select a language, and then click **Next** (Figure 3-5).

Notice on this window that you can click the **Installation Guide and Release Notes** link to view the this manual, as mentioned at the earlier in this chapter.

Figure 3-5   Language selection
4. The IBM License Agreement window opens (Figure 3-6). Read the terms of the license agreement. When you are sure you fully understand and agree with these terms, select I Agree, and then click Next.

![Figure 3-6 IBM License Agreement](image)

The first window of the Installation Wizard opens (Figure 3-7); it lists the installation location.

![Figure 3-7 Installation location](image)

5. In this window, you may change the installation location. By default, the installation directory is `C:\Program Files (x86)\ibm\cognos\c10`. Some business requirements might require a different changed. For example, some organizations allow only the operating system to reside on the C drive; other applications, such as Cognos Dynamic Cubes, must be installed on a secondary drive. Use this window to change the default location if necessary. However, because Cognos Dynamic Cubes requires Framework
Manager 10.2, the installation directory must be the same directory where Framework Manager 10.2 is installed.

If you select a directory other than where Framework Manager is currently installed, an error window opens (Figure 3-8).

After you select the correct location for the installation, click **Next**.

6. The Component Selection window opens (Figure 3-9). Cognos Dynamic Cubes is one single component, which is named IBM Cognos Cube Designer. This option is selected by default. Click **Next**.

![Figure 3-8 Installation location error message](image)

![Figure 3-9 Component selection](image)
7. The Shortcut Folder window opens (Figure 3-10). In this window, you can select the Start Menu folder where you want to install the Cognos Cube Designer application. By default, the Cognos Cube Designer shortcut will be placed on the IBM Cognos 10 folder.

In this window, you may also make available the Cognos Cube Designer shortcut to all users for that workstation, or you can limit the shortcut to only the user account that we are using to install the application. Security is the main reason why some people might want limit the shortcut to only the installation user, especially when multiple users have access to the same workstation.

When you finish with your selections, click Next.

![Figure 3-10  Shortcut folder](image)
8. The Installation Summary window opens (Figure 3-11). It lists all the options you selected during the installation. Carefully review all the settings. If you want to change a setting, click **Back** to go through all the options of the Installation Wizard. Make any necessary changes and return to the installation summary. When you are satisfied with your selections, click **Next**.

![Figure 3-11 Installation summary](image1)

The Installation Wizard begins the installation on the workstation. A progress window opens (Figure 3-12).

![Figure 3-12 Installation progress](image2)
9. When the installation is complete, a Finish window opens (Figure 3-13). You may view the transfer log and summary-error log. Click Finish to complete the installation.

![Figure 3-13   Finish window](image)

### 3.5 Installing Dynamic Query Analyzer

Dynamic Query Analyzer (DQA) is a troubleshooting application that analyzes the query logs generated by dynamic cube query requests. It provides a straightforward graphical interface to help you understand and analyze queries for further performance tuning.

Dynamic Query Analyzer can be installed on Microsoft Windows or on Linux. It also is available in 32-bit and 64-bit versions. For the purpose of this book, only the Microsoft Windows 64-bit version is described. The Microsoft Windows 32-bit version uses exactly the same installation method, except that the default location is different.

If you want to install the Linux version of the Dynamic Query Analyzer, see the installation guide that is included with IBM Cognos 10.2.

#### 3.5.1 Dynamic Query Analyzer installation requirements

DQA can be installed locally on, or remotely from, your IBM Cognos BI Server. The installation folder can be independent or shared with other IBM Cognos software components; there are no inherent dependencies like Framework Manager and Cognos Cube Designer. Consult the [Dynamic Query Analyzer Installation and Configuration Guide 10.2.0](http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/nav/0_7) for available configuration options:
3.5.2 Installation procedure

Complete the following steps to install Dynamic Query Analyzer:

1. Go to the location of the installation files in the root folder where the iso.tar file is extracted. Right-click Autorun.inf and select Install (Figure 3-14).

![Autorun.inf](image)

*Figure 3-14  Autorun.inf*

2. Depending on the security level set at the workstation, you might be prompted to confirm the correct installation file (Figure 3-15). Click Open.

![Security warning](image)

*Figure 3-15  Security warning*
Chapter 3. Installation and configuration of Cognos Cube Designer and Dynamic Query Analyzer

An IBM welcome window opens (Figure 3-16), followed by the first option of the Installation Wizard (Figure 3-17).

3. Select a language to use during the installation and click **Next** (Figure 3-17).

Notice on this window that you can click the **Installation Guide and Release Notes** link to view the this manual, as mentioned at the earlier in this chapter.
4. The IBM License Agreement window opens (Figure 3-18). Read the terms of the license agreement. When you fully understand and agree with these terms, select **I Agree**, and click **Next**.

![Image of License Agreement Window]

**Figure 3-18** License agreement

The first window of the Installation Wizard opens (Figure 3-19); it shows the installation location.

![Image of Installation Location Window]

**Figure 3-19** Installation location
5. In this window, you may change the installation. By default, when you install the 64-bit version, the installation directory is `C:\Program Files\ibm\cognos\c10_64`. A different default location might be necessary for some business requirements. For example, some organizations allow only the operating system to reside on the C drive; other applications, such as Dynamic Query Analyzer, must be installed on a secondary drive. Use this window to change the default location.

After you select the correct location for the installation, click **Next**.

6. The Component Selection window opens (Figure 3-20). Dynamic Query Analyzer is one single component. This option is selected by default. Click **Next**.

![Component selection](image-url)
7. The Shortcut Folder window opens (Figure 3-21). In this window, you may select the Start Menu folder where you want to install the Dynamic Query Analyzer application. By default, the Dynamic Query Analyzer shortcut will be placed on the IBM Cognos 10 folder.

In this window, you have the option to make available the Dynamic Query Analyzer shortcut to all users for that workstation, or you can limit the shortcut to only the user account that we are using to install the application. Security is the main reason why some people might want limit the shortcut to only the installation user, especially when multiple users have access to the same workstation.

When you finish making your selections, click **Next**.
8. The Installation Summary window opens (Figure 3-22). This window lists all the options you selected during the installation. Carefully review all the settings. If you want to change a setting, click Back to go through all the options of the Installation Wizard. Make any necessary changes and return to the Installation Summary. When you are satisfied with your selections, click Next.

![Figure 3-22  Installation summary](image)

The installation wizard starts installing the Dynamic Query Analyzer on the workstation. A progress window opens (Figure 3-23).

![Figure 3-23  Installation progress](image)
9. When the installation is complete, a Finish window opens (Figure 3-24). You may view the transfer log and summary-error log. Click **Finish** to complete the installation.

![Finish window](image)
3.5.3 Configuring Dynamic Query Analyzer

To be able to use Dynamic Query Analyzer, you must properly configure it to connect to the IBM Cognos 10.2 Business Intelligence Server. Use the following procedure to configure Dynamic Query Analyzer.

1. Start Dynamic Query Analyzer. Select **Start menu → Programs → IBM Cognos 10**, and then click **IBM Cognos Dynamic Query Analyzer**.
2. After Dynamic Query Analyzer is loaded, select **Window → Preferences**, as shown in Figure 3-25.

![Preferences](image)

*Figure 3-25 Preferences*
3. Select **Cognos Server** from the navigation pane (Figure 3-26).

![Figure 3-26  Cognos server preferences](image)

Update the following information:

- **Dispatcher URI and Gateway URI**
  
  By default, the Dispatcher URI and Gateway URI are set to *Localhost*. If the IBM Cognos 10.2 Server is on a different computer, then update both settings. If you do not know the location of the IBM Cognos 10.2 Server, contact your Cognos Administrator.

- **Name, password, and namespace of the user**
  
  If security is implemented on the IBM Cognos 10.2 Business Intelligence Server, then type your user name, password, and select the namespace. If you do not know your user name, password, or namespace, contact your Cognos Administrator.
4. If you want to use Dynamic Query Analyzer to analyze query log files, select **Logs** in the navigation pane (Figure 3-27).

![Figure 3-27  Dynamic Query Analyzer logs preferences](image)

Update the following information:

- **Virtual directory path**
  
  In the Logs directory URL field, enter the virtual directory where IBM Cognos 10.2 Business Intelligence Server is saving the query log files.

- **Name and password of the user**
  
  If security was implemented on the virtual directory, enter your user name and password. If you do not know your user name or password, contact your Cognos Administrator.
Chapter 4. Modeling dynamic cubes

This chapter describes how to use the IBM Cognos Cube Designer to design and deploy dynamic cubes, how to model cube dimensions, and how to validate and analyze the models you create. Also included is information about troubleshooting the models and creating a sample cube.

This chapter contains the following sections:

- 4.1, “Introduction to IBM Cognos Cube Designer” on page 60
- 4.2, “Working with Cognos Cube Designer” on page 60
- 4.3, “Modeling dimensions” on page 70
- 4.4, “Calculated members” on page 78
- 4.5, “Modeling time dimensions for relative time” on page 78
- 4.6, “Slowly changing dimensions” on page 79
- 4.7, “Role-playing dimensions” on page 82
- 4.8, “Multi-grain fact scenarios” on page 84
- 4.9, “Aggregate cubes” on page 85
- 4.10, “Validation” on page 90
- 4.11, “Deploying dynamic cubes for reporting and analysis” on page 91
- 4.12, “Troubleshooting” on page 92
- 4.13, “Creating a sample cube model by using GOSLDW” on page 95
4.1 Introduction to IBM Cognos Cube Designer

IBM Cognos Cube Designer is a new modeling environment that is built for the design and deployment of dynamic cubes. Cognos Cube Designer contains a number of features to streamline the modeling process and help users ensure that a quality model is defined and deployed.

Users can explore the structure of their data and examine the relationships between the tables in their data warehouse to determine the objects to be used in their cube model. Auto-design features in Cognos Cube Designer use primary-foreign keys, if they exist, in the data source of the users to create a cube model as a starting point for users to further refine.

Cognos Cube Designer modeling workflow
To design and deploy a dynamic cube application to your IBM Cognos BI environment, you must first model your cube definition in Cognos Cube Designer. Here, you select the database schema that contains the data against which you want to report or perform analysis, import the required metadata, and then define the dimensions and measures that will be included in your dynamic cube. After the cube is defined, you may deploy your dynamic cube to your server and then perform additional operations, such as use Dynamic Query Analyzer to recommend aggregates, or apply security in Cognos Connection.

The following sections describe these steps in greater detail and provide step-by-step examples of how to perform these tasks.

4.2 Working with Cognos Cube Designer

We will be working with the gosldw_sales data source that is included in the IBM Cognos BI 10.2 Samples. A model is used to illustrate points and to allow you to get experience with Cognos Cube Designer. This model is similar to the sample Cube Designer model that is included with the IBM Cognos BI samples. If you want to follow along with the activities in this chapter, you can use the Cognos Cube Designer sample model. The assumption is that you have access to the gosldw_sales data source as you work through this chapter.

The end of this chapter has a collection of tasks, in tutorial format, to guide you through building the entire Cognos Cube Designer model that is used in this chapter.

4.2.1 Launching Cognos Cube Designer

Cognos Cube Designer can be launched from the Programs Menu in the Start Menu or from within IBM Cognos Framework Manager, by selecting the Run IBM Cognos Cube Designer option from the Tools menu. If you installed Cognos Cube Designer correctly, you see this additional menu option available in the Tools menu. If it is not present, reinstall Cognos Cube Designer on top of your Framework Manager. After starting Cognos Cube Designer, the Welcome window opens and you have the option of creating a new project or opening an existing project from the file system.
Figure 4-1 shows the Welcome window of IBM Cognos Cube Designer.

4.2.2 Creating a new project

To create a new project, launch IBM Cognos Cube Designer and either select the **Create New from Metadata** option on the Welcome window, or choose the **Create New Blank Project** option. Selecting to create a new project from metadata will open the Select a database schema dialog, which will display the Cognos BI data sources that you have access to. All configured data sources from your Cognos BI Server environment to which you have access, are displayed. From here, you can select the schema you want to use and move forward with the metadata import step.

If you choose to create a new project without importing metadata, the Cognos Cube Designer workspace starts and you can either begin to build the framework of your cube or you can choose to import metadata at that point.

4.2.3 The Cognos Cube Designer workspace

The Cognos Cube Designer workspace is organized into three main areas, from left to right in your window: Data Source Explorer, Project Explorer, and Model Explorer. Each area can be expanded or collapsed to free additional real estate in the window for the area you are using at a given time.
Figure 4-2 shows the workspace of Cognos Cube Designer.

The Data Source Explorer is where you can view tables, views, and other objects from a reporting data warehouse. These objects are created during the metadata-import step that we describe in a later section. This area contains the source objects that you will use to create your dimensions and measure dimensions for your dynamic cube.

The Project Explorer is where the dynamic cubes, dimensions, measure dimensions, and other objects will be created as you build your application. This area is organized in a tree view, which is similar to the organization of objects you see in the metadata tree in the IBM Cognos reporting studios. When you created your new project in 4.2.2, “Creating a new project” on page 61, a model object was created for you at the root of your project. A dynamic cube project will have a single model, but you may have multiple dynamic cubes and other objects such as dimensions, measure dimensions, calculations, and so on in your model.

You can create your dynamic cube objects in the Project Explorer or you can create them in the Model Explorer. In either case, the objects themselves will be added to the Project Explorer.

The Model Explorer is where the main object edit windows are located. Although you can create dimensions and cubes directly in the Project Explorer, the main object definition...
editors are available only in the Model Explorer window. The default state of the Model Explorer is a list of all of the objects in your model. Double-clicking any of the objects in this list will open the appropriate editor for the selected object.

Immediately below the Model Explorer are the Properties and Issues tabs:

- The Properties tab contains the various properties and expression definition of the item that is currently selected and in focus. Many of these properties are editable.
- The Issues tab contains a list of any unresolved issues with the object currently selected. These issues must be resolved before the object you are working on is valid. A cube cannot be deployed to Cognos Connection if there are outstanding issues to be resolved.

Cubes
Dynamic cubes consist of a collection of dimensions, representing dimension tables in the data source, and a single measure dimension, representing the fact table. There can be only one measure dimension in a dynamic cube, but there can be any number of regular dimensions in a cube. The term regular dimension is used to distinguish between dimensions that contain member data versus those that contain fact data or measures.

The modeling exercise is to create the cube by deciding the fact table you want to use in the measure dimension of the cube, identify the dimensions you want, their grain to the fact table, and establish relationships between them and the cube’s measure dimension.

Dimensions
Dimensions are metadata constructs. They guide the relational queries and the specification and creation of the hierarchies and members of the dimensions.

A measure dimension is the container for the facts in the cube. These facts might exist in the data warehouse or can be created as expressions. They can be values or created from counts of events, states, and other attributes that your consumer might be interested in tracking. The latter are commonly known as fact-less fact tables.

Three types of dimensions exist in addition to the measure dimension: regular dimensions, time dimensions, and parent-child dimensions.

Hierarchies
A hierarchy is a structure that organizes and relates the attributes of a dimension into layers of increasing level of detail from the most abstract to the most concrete.

For example, a time dimension hierarchy can consist of a year level, a month level, and a day level. The hierarchy’s highest level, year, has the most abstract information classifying time. If you were examining something from the level of years, all the data would be aggregated at that level. Months are more detailed and days even more so. A product dimension hierarchy can consist of a product line level, product type level, and a products level.

It is possible to have multiple hierarchies in a dimension. For example, a time hierarchy can describe the time elements in a calendar year. Another hierarchy might organize time in a fiscal year. Each hierarchy is a way to classify data in an organizing structure.

Levels can consist of attributes of a dimension or expressions that operate on attributes. For example, a store dimension might have a hierarchy that organizes stores by the attribute of size. A level might consist of an expression that takes individual store sizes and groups stores with similar-sized stores.
A member could exist in multiple locations in a level but each instance would be qualified by the level unique key so that the context that makes each instance unique would be identified.

**Measure dimensions**

Measure dimensions contain the fact data, or measures, that you will use in your reporting and analysis. A measure dimension is based on a single fact table in the data source. A dynamic cube can have any number of regular dimensions, but only a single measure dimension based on a single fact table in the data source. If your application requires using measures from multiple fact tables, see Chapter 6, “Virtual cubes” on page 133 to learn more about virtual cubes.

### 4.2.4 Creating calculated measures

You can extend the usefulness of cubes by creating calculated measures, which have custom expressions that your consumers will want to use. The expressions can use the rich function library, available to create calculated measures. With the calculated measures, you can build, into your cube, measures that are complex and take considerable time to generate results. By building them into the cube, they need to be created only once. It is not necessary to re-create the expression for each report that your consumers need to use. The administration of each measure is easier, too. Because the measure is built into the cube, the performance of reports that use it will be faster.

**Report objects**

Cognos reporting applications use two types of metadata in their packages that are available to report authors. They provide unique features that can help you fulfill your reporting requirements.

The first type is the metadata that you create when you model in Cognos Cube Designer. These are the dimensions, hierarchies, levels, attributes, and measures. For example, using a level in a report will always get the members that belong to that level. The report does not need to be updated as the dimension changed.
Figure 4-3 shows the metadata that is created when you model in Cognos Cube Designer.

Figure 4-3 Creating metadata

The second type is the members that are generated in the cube as a result of modeling, as shown in Figure 4-4.

Figure 4-4 Members generated in the cube

How you want to represent and model your dimensions is your choice. A regular dimension can have many hierarchies. Using multiple hierarchies within a dimension may help organize similar hierarchies. Depending on what your dimension is like, you could have many possible hierarchies.

There is no effect of the presence of a hierarchy on the other hierarchies in the dimension. The query that generates the members for each hierarchy is separate. If, however, objects from multiple hierarchies are included in a report, the tuple combinations that do not exist in the fact will be removed from the set during query processing. The effect in the report is to filter it.
### 4.2.5 Importing metadata

When you create a new project in Cognos Cube Designer, one of the steps you are asked to do is to select a data source. The list of data sources you see are those Dynamic Query Mode (DQM) data sources that exist in the Cognos BI Server environment that you have access to. When you created the project, you were asked to authenticate with the Cognos BI Server environment. The list of data sources you see is filtered to show only those DQM data sources you have access to. If you do not see a data source that you believe you should have access to, contact your Cognos system administrator to ensure that the correct level of access is granted.

Generally, you should import the metadata from your reporting data warehouse before you begin building the dimensions and cubes that will form your application. Although this is the suggested workflow, you may begin building the framework for your cubes, dimensions, and measure dimensions, and then import the metadata and perform a mapping exercise to associate the level attributes with columns in your tables and views. If you are unfamiliar with your data source, this workflow will be difficult to do; therefore, the suggested workflow is to import metadata first.

When you select a data source for import, you are presented with a list of the schemas or databases that you have access to. You may select the schema or database that you want to use. After you select the schema or database, all objects that you have permission to access will be imported. In many cases, importing mean the entire contents of the schema or database. The permissions granted are determined by the credentials that were used when configuring the data source in Cognos BI Server. If you notice that tables or views that you were expecting to see are not imported, a permissions issue for the data source might be the cause. Work with your Cognos system administrator and your DBA to resolve these issues.

After the import is complete, a tree view opens that contains all objects that were imported. You can expand the tables and views, and select the individual columns to see their properties in the Properties tab. The Data Source Explorer is a read-only control, so you may not edit any object in that area.

You may import metadata from more than one data source into your project, however each cube might contain objects that are sourced from a single data source. The ability to use multiple cubes from multiple data sources is described in Chapter 6, “Virtual cubes” on page 133. To import additional metadata from another data source or schema, click Get Metadata on the toolbar. The list of available data sources, which you saw when creating the project, are displayed. Repeat the import steps for the data source of your choice. An accordion control is displayed in the Data Source Explorer so that the individual collections of metadata are kept separate. You can toggle between these data sources by clicking the tab of your choice in the Data Source Explorer.

### 4.2.6 Exploring data using the Relational Explorer Diagram

During the process of building your application, you might want to explore the data source you are using, to either see what tables are related with primary-foreign key relationships, or how a dimension table joins to the fact table. Alternatively, you might want to see the contents of your data source in a more flexible view than the standard tree view in the Data Source Explorer.

Cognos Cube Designer features an explorer diagram that you can use to see the tables and views in your data source, and detect how the tables relate to one another. To open the diagram, right-click a table in the Data Source Explorer control and select Explore Metadata. The diagram opens and adds the selected table to the default view.
After the explorer diagram is open, you can drag additional tables onto the diagram from the Data Source Explorer control. Any primary-foreign key relationships that exist are automatically visible. You can also choose to show any related tables to those already visible on the diagram. The show joined tables buttons on the toolbar automatically add these tables to the canvas. Buttons show all related tables or tables that are joined in a 1..n or n..1 relationship to the selected table.

Use the slider control on the toolbar to control the level of detail visible on the canvas. You can choose to see a low level of detail, such as the table names, or higher levels of detail, such as the individual columns and keys for the object on the canvas.

Figure 4-5 shows a Relational Explorer Diagram.

Diagram is read-only: The diagram is read-only; its purpose is to enable the user to examine the objects in the data source and how they relate to one another. There is no ability to edit any of the objects in the diagram, or create new objects. These operations must be done in either the Project Explorer or the Model Explorer.
4.2.7 Exploring data using the tabular data view

You can use the tabular data view to preview the data that exists in a table. For many modeling operations, such as modeling level keys and identifying hierarchies in dimensions, knowing the data that you are working with can be useful. You can pin the tabular data view so you may consult it as you do your modeling. You can click the Top, Page up, Page down, and Bottom buttons to view additional data in the table.

4.2.8 Creating a dynamic cube

When you are satisfied that you are ready to begin modeling your application, you can begin working in any one of a number of ways. You can begin building dimensions and a measure dimension and then insert them into a cube, or you can start with a cube and then add dimensions and a measure dimension. If starting with a cube, you have the option of creating the cube and the other objects manually, or, if there are primary-foreign key relationships present in your data source, you can use some of the auto-design functionality available in Cognos Cube Designer. This functionality is similar to the functionality available in IBM Cognos Transformer and in the Model Design Accelerator feature in IBM Cognos Framework Manager.

Auto-generate a dynamic cube

The auto-design functionality in Cognos Cube Designer uses the primary-foreign key relationships between tables in the data source to detect and extrapolate the structure of the dimensions. In a star schema design warehouse, it creates a dimension for each dimension table that is joined to the fact table. It creates a default hierarchy with a single level. In the case of a snowflake warehouse design, it creates a dimension with a default hierarchy and a level for each table in the snowflake structure of the dimension, moving from the furthest from the fact table inwards. Columns in each table will be attributes of the level corresponding to its parent table.

To use the auto-generate cube feature, right-click the fact table you want to use from the Data Source Explorer, and select Generate cube.
Chapter 4. Modeling dynamic cubes

Figure 4-6 shows auto-generation of a dynamic cube.

Figure 4-6  Auto-generating a dynamic cube

Cognos Cube Designer generates a cube in the project explorer with the fact table you selected as the measure dimension and a dimension for each dimension table joined to the fact table.

By default, the cube inherits the name of the fact table because a dynamic cube may have only a single fact table. The dimensions inherit the name of the source tables by default. You can then go through the process of making any modifications necessary, because the auto-designed cube may not perfectly match your reporting requirements. This feature is intended to provide a head start for the modeling process and save you time. Click the Issues tab for a list of items that need to be addressed for each of the dimensions and the measure dimension.

Note: This feature uses primary-foreign key joins between the dimension tables and the fact table. The auto-generate function does not work if no joins are present in the data source.
Manually create a dynamic cube

If you want to build your dynamic cube manually or need to do so because there are no primary-foreign key joins between the tables in your data source, you can create a new empty cube by clicking the cube button on the toolbar in the Project Explorer window. This steps adds a new empty cube to your project. You can then rename the cube and begin adding dimensions and a measure dimension.

There are no limits to the number of cubes that you can have in your Cognos Cube Designer project, but the cubes must be published one at a time, and a single, separate data source and package will be created for each dynamic cube. If your application requires more than a single cube, see Chapter 6, “Virtual cubes” on page 133 to learn more about virtual cubes.

4.3 Modeling dimensions

Dimensions provide context to the facts and delimit the spaces in which the facts have meaning. They have metadata information that you specify to indicate how to organize the data into dimensions and how to classify the data from higher to lower to degrees of abstraction.

The most common dimension is the regular dimension. Its defining characteristic is that it consists of one or more defined hierarchies. Each hierarchy is a set of ordered levels that relate the attributes of the dimension to each other and organize the data in them.

For example, a store dimension could be organized in a hierarchy that consists of levels of groups of countries, countries, sub-national groupings such as states, provinces or regions (plus handling for entities that do not have such groupings), cities, and stores. The branch dimension has a hierarchy of that nature.

You can have a hierarchy that organizes stores by some other attribute, such as store size.

Time dimensions are regular dimensions with the addition of a setting to allow for the generation of relative-time calculated members.

In parent-child dimensions, the organizational information is contained in the data. You need to specify which member is a parent to another member. From that information, the tree of members is assembled when the dynamic cube is generated. You must specify that a member has one and only one parent. For more information, see 4.9.5, “Parent-child dimensions” on page 87 and 4.6.3, “Slowly changing dimension as a parent-child dimension” on page 80.

In a data warehouse, dimensions are contained in either dimension tables or snowflakes.

- If the dimension is in a dimension table, the Cognos Cube Designer model will need to have the levels of the dimension defined by a process of renormalization. You usually able to identify a hierarchy through the redundancy of data in some columns. There might be certain analytical scenarios that you are being asked to model in which the data could be less redundant but a valid hierarchy could exist.

- If the dimension is in a snowflake, the tables might or might not match the levels that you want to model. You must identify these cases and model accordingly. The goal of the application can be useful guidance.

In some cases, what looks like a snowflake is a dimension table with associated lookup tables.
4.3.1 Relationships

You must specify a relationship between the measure dimension and its participating dimensions. These relationships form the basis of the relational query join statements.

In addition, you must specify a relationship between objects within a dimension if they do not have one. For example, when you create the Product dimension in 4.13, "Creating a sample cube model by using GOSLDW" on page 95, you will need to create a join between VIEW_PRODUCT_NAME and SLS_PRODUCT_DIM. There are two instances:

- One is creating a relationship within a level, such as with a lookup table. Lookup tables are usually one to one.
- The other is between one level and another. If your database does not have primary key-foreign key constraints you need to do this instance.

Relationships between levels are implicitly one-to-many. In a snowflake schema, this cardinality usually indicates that the tables that are related in that way are part of a dimension. Although it is not necessarily the case, each table in the snowflake can be a level in the dimension. The flow of relationship cardinality should flow down: from the highest level in a hierarchy to the lowest level.

In a dimension table, you need to renormalize the attributes that you want in a hierarchy.

The cardinality flow from each dimension to the measure dimension should be one-to-many.

A relationship does not need to be used with a column that has referential integrity. In the gosldw_target cube, which we create in 4.8, “Multi-grain fact scenarios” on page 84, the relationship between the target measure dimension and the Time.Month dimension would use MONTH_KEY. If you look at GO_TIME_DIM in the Data Source Explorer, you will see that MONTH_KEY is neither a primary key (PK) nor foreign key (FK). For more information, see 4.15, “Modeling for multi-grain scenarios” on page 112.

The grain of the fact can be higher than the grain of the dimension. For example, the grain of the Time dimension in gosldw_target is month. The reason is because the target facts are abstracted to the month level. The Time dimension’s grain is day. To model this, create a copy of the Time dimension, terminating at the month level, and put it into the target cube. For more information see 4.8, “Multi-grain fact scenarios” on page 84.

You can create a relationship by using many pairs of attributes. The set of attributes that define a relationship must be confined to defining only one relationship. If your measure dimension has more than one possible relationship to a dimension, you must create role-playing dimensions to have unambiguous queries. If the dimension has more than one relationship between it and a fact table, you need to verify what key is being used for any relationship. That key defines part of the relational query and is important.

Because dimensions can have PK-FK constraints with the fact table in the database, relationships can automatically be created by Cognos Cube Designer.

4.3.2 Modeling levels

The level editor contains the list of attributes that define the level, its characteristics, and the attributes that are associated with members in the level.

With the level key editor, you can define the level key. The level key editor has a list of attributes that you use to define the level key. The first attribute in the level key list is deemed to be the business key. The level key editor has a list of attributes that are used in the other
levels in the dimension. You can bring them into the level key. They are not added to the level itself but to the level key. For example, in a slowly changing dimension, you need to have the business keys of the higher levels included in the level key to uniquely identify the member. For more information, see 4.6, “Slowly changing dimensions” on page 79.

The level editor requires you to set an attribute to be the member caption. This value will be displayed in the member tree and reports. The member caption does not need to be unique. In a slowly changing dimension, you can have many instances of an entity each with the same member caption but defined by a different level unique key. The uniqueness for the instances is defined in the level key. If you want each member to have a caption that you or your consumers can use to distinguish a member from another without consulting its Member Unique Name (MUN), you can create an attribute with an expression that contains the values that you want to use. An expression concatenating attributes usually does this task.

Figure 4-7 shows the list of level keys.

![Figure 4-7 Defining a level key](image)

You can also sort members. With the member sort editor, you can define which attributes to sort the members and by what order. You can have many sorts. For example, you may sort an employee level by the employee last name and then the employee start date. Each sort is performed on the members, based on the preceding sort order.
Figure 4-8 shows an example of sorting members.

![Figure 4-8 Sorting members](image)

You can define a member description. It is an optional role. In the Cognos studios this attribute will be displayed in the metadata tree.

An attribute is a property of an entity that provides information about the entity. A simple attribute might be a column from a table. You can create attributes that have expressions in them. For example, you can create an expression that concatenates the values of two columns. Some attributes are necessary for defining the level key and member caption.

Attributes are displayed in the metadata trees of those Cognos Studios that support them. They appear as children of the level. All attributes appear except for the member caption and level key attributes. Two attributes that do not appear in the level are displayed also:

- One is the member description. If you have set it in Cognos Cube Designer, this attribute displays the member description text. If not, it displays empty cells.
- The second generated attribute will have the business key value. Its name is in the following form:
  
  `{Level name} – Key`

You can use the attributes to help filter reports by values that are in the attributes. For example, you may use a product color attribute to filter a report to return values only for those products that are red.

**4.3.3 Member tree organization options**

The (A11) level is a container for a member that acts as a parent to the members of the highest level of a hierarchy. It exists in a hierarchy that has the multi-root property value set as false. If the multi-root value is set as `true`, no (A11) level exists in the hierarchy. The highest level members display as the parents in the member tree or metadata tree member folder. The (A11) level has a default value in English of (A11). It can be named anything and can have any name for any locale.

The name of the member that is in the (A11) level is set in the root member property. It is sometimes referred to as the ALL member.
The effect of the existence of the (A11) member is illustrated here. Time (ship date), for the purposes of illustration, had its multiple root members property value set to true. Time retains the (A11) level. As you can see, the member tree for Time will have the root member generated.

Figure 4-9 shows the member tree organization.

Figure 4-9  Member tree organization

Figure 4-10 shows ALL member tree.

Figure 4-10  ALL member tree
4.3.4 Modeling level keys

The level key is the mechanism to define the members of a level. It is a set of attributes and attribute expressions. The level key must uniquely identify the member within the level. The level key can consist of one or more attributes. The combination of these attributes will be the level unique key.

A member is uniquely identified when the value of the level key for the member does not match the value of the level key for any other member in the level. This uniqueness is defined partly by the data contained in the key and partly by the objects of the key. If the data in an attribute is sufficiently detailed, it can uniquely identify a member. If not, you must add more attributes to the level key.

These keys guide the SQL queries that are made to the databases to retrieve the data, governing, among other things, GROUP BY.

A possibility is for a single column to uniquely identify a member, even if the column was not a key in a table. The data in the column might be sufficient to identify the member. For example, in the sample database GOSLDW, the column QUARTER_KEY in GO_TIME_DIM has values in a format that uniquely identifies things. The values in QUARTER_KEY, such as 20131, 20132, 20133, and 20134 have sufficient information to identify a quarter within the context of a calendar year. The modeler must be careful, otherwise additional data causes the key to no longer be unique. It is also possible that the sample that the modeler examined contains uniquely identifying data, but other records do not.

An example of a column that does not uniquely identify a member by its data is be the MONTH_NUMBER column in GO_TIME_DIM. The values, such as 1, 2, and 3 do not identify what the month is in the context of any particular year. GO_TIME_DIM has a column that serves as a month level key, MONTH_KEY. In many cases you might not have that luxury. You might need to create a key where the business key is insufficient to identify the member.

It is necessary to distinguish uniquely identifying data from unique data. Data is unique when a datum in a column occurs only in one record in the table. Uniquely identifying data might exist multiple times. In a time dimension table in a data warehouse, the datum for any particular quarter could be duplicated for each record at the lowest grain of the dimension. If the dimension is at the day grain, there could be over 90 records in which each quarter value would exist. In GO_TIME_DIM you will see examples of such redundancy.

In the level key, the business key is the value that governs the generation of members. Each distinct value in the business key will be used to create a member. In the level key editor, it is identified by the key icon. The order of the keys in the level key is important. The first key attribute is assigned the business key role. You must re-order the attributes to make the attribute that you want to be business key. A business key does not need to uniquely identify an entity. That is the purpose of the level key. What is necessary is that the combination of the business key, and the business keys of its ancestors, uniquely identify an entity.

The additional keys provide context. Usually they can be the business keys of the higher levels in the hierarchy. For example, assume that the values identifying the quarters in a time dimension are not unique but are in the form of 1,2,3,4. You need to include additional keys to make each quarter unique.

You might think that you need to include the business keys of higher levels in the level key but that can have performance and other issues. It should not be necessary. The metadata should contain sufficient information, either in the business key itself or in conjunction with the other keys that you use in the level key. For a slowly changing dimension, however, the inclusion of the higher-level business keys are necessary.
Knowing the data and what it is being used for is important. Something that seems conceptually sound can produce results that are incorrect. If the application is meaningless to you, it can be difficult to know if you are implementing it correctly.

Some reporting requirements can specify alternate hierarchies which will organize the members based on particular attributes that are of interest to users. You need to be aware that the levels must identify the members as being unique within the context of the hierarchy. It is possible that these hierarchies can produce non-unique members although the level key can produce uniquely identified members in other contexts.

For example, assume that a dimension organizes stores by geography in one hierarchy. An alternate hierarchy that organizes the stores by some attribute, store size class for instance, can be problematic if you attempt to reuse some levels of the geography level in the alternate hierarchy, but some levels can be reused without a problem. For example, the city level can be uniquely identified in the geography hierarchy but, because stores of different sizes exist in a city, the city can repeat in the city level of the store size hierarchy. The city level in the store size hierarchy will need additional keys to identify each instance of that city in the level.

Composite key expressions are common in some applications but the nature of the level key does not require their use. Having the attributes that uniquely identify the members in the level key is sufficient.

In modeling for multilingual models, it is important to use keys that do not vary from one locale to another.

Be aware of the potential number of members that are generated for any node in the member tree. A large number can be quite difficult to navigate through and a very large number can impede performance. Discuss with your consumers what sort of reports they will write. With this information, you can create hierarchies that categorize members into smaller, more exact sets.

You can use that discussion as an opportunity to refine the model to meet the needs of the consumers. They might have requirements that they did not tell you about or did not request because they might not believe that model it is possible. This information can assist your consumers in generating reports and designing the functionality that might otherwise need to be specified in the report. Additional functionality, because it would be built into the cube, can speed up report processing time. It can have practical benefits of easing communications and other aspects of office politics.

An expression can have a null result if one element of the expression returns a null. For example, some records in EMPLOYEE_DIM have null values for the column Address2. An expression that used Address2 will return nulls for each record where the value for Address2 was null. If you have data with nulls, you must incorporate tests for nulls to handle them appropriately:

```
Address1|| ' ' || Address2
```

An expression that tested for nulls might look like this:

```
Address1|| ' ' || if (Address2 is null) then ( ' ' ) else (Address2)
```

If the null happens for an attribute that is being used in the level key, some members might not be generated. If the null happens in a member caption, the member caption displays NULL as its caption, even if other elements of the expression that is used to define the member caption are not null. Expressions of this type usually need the data type to be the same.
4.3.5 Parent-child dimensions

Parent-child dimensions do not have defined levels. The data of the parent and child attributes determine how the members are generated and assembled into a member tree. They can have only one hierarchy.

Functions that reference a member in the context of a level or descendants return each child member that is that many member nodes down the tree. If you have an unbalanced hierarchy, the member tree will generate without any additional modeling required.

If the parent-child dimension is a slowly changing dimension, the parent and child attributes need to be surrogate keys.

Data members are members that contain values that are associated with the parent member itself. The data member allows you to see those values. If the data member does not get specified, the values will still be aggregated into the parent member but there will be no indication of the source of that value. For aggregations such as sum, it would be fairly readily apparent that the parent member had values that are associated with it. For others, such as average, it is would be less so. A good idea is to include them by setting the show data member property value to true and setting the caption of data members property value to be Parent’s caption.

Root member will contain the ultimate parent, enabling you to model so that the ultimate parent member is displayed in the member tree. If you do not set a value for the root member property on the root member of the dimension for the member tree will be the ultimate parent. The caption for the member as displayed in the member tree in the Cognos studios will be the dimension’s name. The actual member name will be used in reports.

Figure 4-11 shows a parent-child dimension where the show data member property was set to true and the caption of data members value was set to the parent’s caption.

![Parent-child dimension](image)

Figure 4-11 Parent-child dimension

An example of a parent-child dimension is in 4.6, “Slowly changing dimensions” on page 79.
4.4 Calculated members

Calculated members are members that the modeler can create by using expressions. They are appended to the end of the list of members in the level of the children of their parent.

Cognos Cube Designer validates the syntax of expressions. It does not check semantic, data type, and binding errors. After a cube is started, the dynamic cube engine validates the semantics of the calculated member and calculated measure expressions. The expression editor does not limit functions to the valid ones for a specific context.

Several restrictions apply to Cognos Dynamic Cubes calculated members. Do not use the following relational constructs in expressions 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 and time constants (Constants)
- All business date and time functions (Business Date and Time functions)
- Like, lookup, string concat '||', trim, coalesce, cast (Common Functions)
- MOD function (Common Functions)

4.5 Modeling time dimensions for relative time

Time awareness is an important modeling and reporting requirement. Many queries and functions require some point in time to know when the current period is and to perform relative time calculations, such as a rolling time window. With relative time capability, report authors can easily write a report, which contains calculations that depend on a current period. The report can be run at any time and return the correct results for the updated period without the report author having to update the report. The cube can be updated.

Built-in relative time members can also save time for report authors and consumers.

In addition, you might have data, such as planning data, that extends into the future, or the period of interest passed but the actual time period you are in has not closed. The last time period value is not necessarily the current period for your reports. For example, assuming your fiscal months correspond with calendar months, on August 14 your current period is probably still July. You need a method to specify this current period.

The current period property exists for each level in a Time dimension. You can seed it with a set value or with an expression. If you use an expression, it must be a scalar function, that is, it must resolve to a single value. The value for the current period must match the business key value for the period that you want as the current period.
4.5.1 Using relative time

Cognos Dynamic Cubes generate built-in relative time members:
- Current period
- Prior period
- Period to date
- Prior period to date
- Period to date change
- Period to date growth

If you do not set the current period value for a level, the last member in the level will be deemed to be the current period.

Instructions for creating a time dimension are in 4.14, “Creating a time dimension” on page 112.

4.6 Slowly changing dimensions

Slowly changing dimensions (SCD) are a type of dimension in which the data describing the entities in a dimension will change over time. This section describes the slowly changing dimension types and use cases that Dynamic Cubes supports.

4.6.1 Modeling slowly changing dimensions

An example of a slowly changing dimension that is easy to understand is an employee dimension. Employees are hired, promoted, given raises, change positions, change workplace locations, resign, retire, and are let go. The effect of these changes can alter the structure of a tree that describes the hierarchical relationship between members of the organization. For example, a change in position might result in an employee becoming a manager and acquiring subordinates who then appear under the employee in the member tree. The change can result in an employee exchanging one set of subordinates for another set.

How the data changes are handled classify the types of slowly changing dimensions. Although the types and classifications of slowly changing dimension types vary, the definition of Type I and Type II SCDs seem to be universally recognized.
- Type I SCDs are characterized by the overwriting of the attribute state by the new data. This type, obviously, has drawbacks.
- Type II SCDs are characterized by a new record being created in the table for each change in the attributes of an entity tracked by the table. Slowly changing dimensions of this type need some way to identify each record in the dimension table to distinguish one member instance from another. The natural key of an entity will be repeated but a surrogate key will be created to uniquely identify each record.

Cognos Dynamic Cubes supports Type II slowly changing dimensions.
4.6.2 Metadata discovery

Be aware of the nature of your dimension and whether it is or will be a slowly changing dimension.

In the Cognos sample database GOSLDW, the table EMPLOYEE_DIM is structured so that we can explore both the concept of the differences between parent-child hierarchies and regular hierarchies and the concept of slowly changing dimensions.

Examine the EMPLOYEE_DIM table in GOSLDW. It is clear that MANAGER_CODE and EMPLOYEE_CODE, which are the natural keys that identify these entities in the dimension, are normally sufficient to be respectively the parent and child keys of this dimension if you are modeling it as a parent-child dimension. Because of changes, however, new records that document the changes are made and those keys no longer are adequate to identify members.

Use the following steps:

1. Select the table EMPLOYEE_DIM in the data source explorer.
2. Right-click View data. The Tabular data tab opens. The first 30 records of the table are displayed. If you want to see more data, click either the Next page or Bottom button.
3. Right-click the Tabular data tab and select Pin. Pinning causes the tab to remain open until you close it.
4. Examine several of the records. The EMPLOYEE_NAME column is near the end of the columns in the table so you must scroll over to it. The employee named Björn Winkler has two records.

Examine the EMPLOYEE_KEY, EMPLOYEE_CODE, MANAGER_CODE, and MANAGER_KEY column values for the two records from Björn.

As you examine the records, you see that Björn changed managers, as Table 4-1 shows.

Table 4-1 Employee values

<table>
<thead>
<tr>
<th>EMPLOYEE_CODE</th>
<th>EMPLOYEE_KEY</th>
<th>MANAGER</th>
<th>MANAGER_CODE</th>
<th>MANAGER_KEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>10016</td>
<td>4006</td>
<td>Gretchen Goetschy</td>
<td>10701</td>
<td>4403</td>
</tr>
<tr>
<td>10016</td>
<td>4007</td>
<td>Fritz Hirsch</td>
<td>10017</td>
<td>4161</td>
</tr>
</tbody>
</table>

Other records in this data source document employees who change workplaces (Antoine Dubois; his natural key is 10680). Enrico Marino, who is located in the member path of Maximilian Saltzman/Bernard Simon/Nicolas Bichot/Tresa Seefelder, gets a promotion. Daniel Hart and Isaac Brule become managers and acquire subordinates.

4.6.3 Slowly changing dimension as a parent-child dimension

To model a slowly changing dimension in a parent-child dimension, both the parent and child need surrogate keys to uniquely identify the member. A member can have multiple parents so a surrogate key must exist: not only for the key that defines the member, but for its position in the hierarchy. Without that information, the hierarchy attempts to form as a network hierarchy and thus fail.
Creating a slowly changing dimension as a parent-child dimension
Complete the following steps:
1. Select the gosldw_sales cube.
2. Right-click create parent-child dimension.
3. Expand EMPLOYEE_DIM in the Data Source Explorer tree.
4. Drag EMPLOYEE_NAME, EMPLOYEE_KEY, EMPLOYEE_CODE, MANAGER_CODE, and MANAGER_KEY into the dimension.
5. Set EMPLOYEE_NAME as the member caption.
6. Set MANAGER_KEY as the parent.
7. Set EMPLOYEE_KEY as the child.
8. Set the relationship between the dimension and the measure dimension to use EMPLOYEE_KEY for the dimension and EMPLOYEE_KEY for the measure dimension.
9. Publish the cube.
10. Expand the member tree for the dimension in Report Studio.

4.6.4 Modeling a slowly changing dimension in a regular dimension

For a regular dimension, the business keys of all the higher levels in the hierarchy must be included in a level. This approach has some performance implications, but, without it, the members will not be uniquely identified. In certain cases, it is possible for the cube to start and generate members, but it is an incorrect set.

The leaf level should have a unique key, which would exist in the database table. More information for handling slowly changing dimensions in a data warehouse would be contained in the standard reference sources for data warehouses, and extract, transform, and load (ETL).

The surrogate key will need to be joined to the fact table. In this way, each record can track the data for each instance of an entity that has changed.

Creating a slowly changing dimension regular dimension
Complete the following steps:
1. Create a regular dimension.
2. Create a hierarchy that has seven levels in it.
3. Select the first level and rename it to: Manager level 1
4. Drag MANAGER_CODE1 and MANAGER1 into the level.
5. Set MANAGER_CODE1 to be the business key and MANAGER1 to be the member caption.
6. Select the second level and rename it to: Manager level 2
7. Drag MANAGER_CODE2 and MANAGER2 into the level.
8. Set MANAGER_CODE2 to be the business key and MANAGER2 to be the member caption.
9. Select the third level and rename it to: Manager level 3
10. Drag MANAGER_CODE3 and MANAGER3 into the level.
11. Set MANAGER_CODE3 to be the business key and MANAGER3 to be the member caption.
12. Select the fourth level and rename it to: Manager level 4
13. Drag MANAGER_CODE4 and MANAGER4 into the level.
14. Set MANAGER_CODE4 to be the business key and MANAGER4 to be the member caption.
15. Select the fifth level and rename it to: Manager level 5
16. Drag MANAGER_CODE5 and MANAGER5 into the level.
17. Set MANAGER_CODE5 to be the business key and MANAGER5 to be the member caption.
18. Select the sixth level and rename it to: Manager level 6
19. Drag MANAGER_CODE6 and MANAGER6 into the level.
20. Set MANAGER_CODE6 to be the business key and MANAGER6 to be the member caption.
21. Select the lowest level and rename it to: Employee
22. Drag EMPLOYEE_KEY, EMPLOYEE_CODE, and EMPLOYEE_NAME into the level.
23. Set EMPLOYEE_KEY as the level unique key.
24. Set EMPLOYEE_NAME as the member caption.
25. Open each level in turn and add the level keys of each level above it.

For example, Manager level 2 will have MANAGER_CODE1 added to it from Manager level 1. Manager level 3 will have MANAGER_CODE1 added to it from Manager level 1, and MANAGER_CODE2 added from Manager level 2.

### 4.7 Role-playing dimensions

A role-playing dimension is a dimension that has more than one possible join between it and a fact table. Each join represents an aspect of the relationship between the dimension and the fact table and how the aspects, or roles, give context to the fact. The goal for modeling role-playing dimensions is to make the relational queries be generated in a predictable fashion.

To do that task, you must separate each of the relationships so that only one relationship exists between an entity and the fact table. You will need to make a duplicate of the dimension for each additional relationship that you want to have. Each of these entities is referred to as a role-playing dimension. They exist as aliases to the dimension table. Each role-playing dimension uses one of the keys to the fact table.

For example, a Sales fact table can have a relationship to the time dimension based on both the sales date and the product shipping date. You need a way to plan the query so that, if you want to know the quantity that was shipped in a particular month, the intended keys are used in the query.

With a role-playing dimension, you know that the query will generate in the way that you want it to. A role-playing dimension enables a query to be generated in a deterministic way. If you use an object from the Sales date dimension in a report, you can be assured that the query will be using the sales date key. If you use an object from the Product shipping date time dimension, you can be assured that the query will be using the shipping date key.
The sample Cognos Cube Designer model has examples of role-playing dimensions. The following steps are the procedure for discovering them. You see three time dimensions. Each has a relationship to the fact table in the measure dimension. The key in the time dimensions is the same, but each relationship uses a different key in the fact table.

1. Select the gos1dw_sales cube.
2. Right-click **Open Editor**. A list of the dimensions that exist in the cube is displayed.
3. Click the **Relationship Edit** link for the **Time** dimension.
   The Join to Measure dialog has a table with three columns:
   - One displays the name of the dimension, **Time**. The key **DAY_KEY** displays under it.
   - The middle column displays the relationship operator.
   - Column three has the key being used by the measure dimension, **ORDER_DAY_KEY**.
4. Click the **DAY_KEY** field and examine the list that is displayed. It contains the columns of the GO_TIME_DIM table. If your dimension was sourced from tables in a snowflake structure, you see all the columns from each of the tables in the snowflake.
5. Click the **ORDER_DAY_KEY** field and examine the list that is displayed. It has the columns of the SLS_SALES_FACT table.
6. Click **Cancel**.
7. Click the **Edit Relationship** link for the **Time** (close date) dimension.
   The Join to Measure dialog has a table with three columns:
   - One displays the **Time** (close date) dimension name. The key **DAY_KEY** displays under it.
   - The middle column displays the relationship operator.
   - Column three has the key being used by the measure dimension, **CLOSE_DAY_KEY**.
8. Click the **DAY_KEY** field and examine the list that is displayed. It contains the columns of the GO_TIME_DIM table.
9. Click the **CLOSE_DAY_KEY** field and examine the list that is displayed. It has the columns of the SLS_SALES_FACT table.
10. Click **Cancel**.
11. Click the **Edit Relationship** link for the **Time** (ship date) dimension.
    The Join to Measure dialog has a table with three columns:
    - One displays the **Time** (ship date) dimension name. The key **DAY_KEY** displays under it.
    - The middle column displays the relationship operator.
    - Column three has the key being used by the measure dimension, **SHIP_DAY_KEY**.
12. Click the **DAY_KEY** field and examine the list that is displayed. It contains the columns of the GO_TIME_DIM table.
13. Click the **SHIP_DAY_KEY** field and examine the list that is displayed. It has the columns of the SLS_SALES_FACT table.
14. Click **Cancel**.
4.8 Multi-grain fact scenarios

A common modeling problem is multi-granularity, which occurs when levels of dimensional detail for facts differ. In your data source, the level of information in a dimension can be more precise than fact data of some fact tables in which the dimension takes part.

For example, a time dimension can have dimension information for the levels of year, quarter, month, and day. For a Sales fact table, the facts exist at the day level. For a fact table with planned sales values, the fact grain or level is probably at a higher level of detail such as month.

This difference in fact grain can make it difficult to plan queries correctly if a report user included a level below the fact grain in the report.

The method for handling this scenario is to create role-playing dimensions for each instance of a dimension that has varying levels of granularity to different fact tables.

In this example, you need to model a time dimension down to the day level for the Sales cube and model a time dimension down to the month level for the sales target cube. For the former, the relationship is formed between the appropriate keys in the fact table and the day level. For the latter, the relationship is formed between the month level and the fact table keys.

If you want to create reports that used data from both cubes, you can create a virtual cube. The virtual cube can have the two Time dimensions merged. The levels of both source cubes will be merged. The virtual cube might have levels that are below the grain of measures from one of the source cubes. Queries that are made in the virtual cube that use dimension levels below the grain of a measure will return null values, ensuring that the consumers of the cube do not have double counting. For more information about virtual cubes, see Chapter 7, “Dimensional security” on page 147.

Perhaps, as in the example in 4.7, “Role-playing dimensions” on page 82, you have the following cubes:

- A sales fact cube where the fact grain for the time dimension is at the day level
- A planned sales cube where the fact grain for the time dimension is at the month level,

A virtual cube that uses both of these cubes as its source enables you to make a query with day level objects (either the level itself or a member of that level, depending on the studio that you are using) against the sales facts and get results and get the expected nulls for the planned sales facts. If you used a Time dimension grain that was common to both fact tables you get non-null results for measures from both fact tables.

Instructions that illustrate this are in the 4.15, “Modeling for multi-grain scenarios” on page 112.

Multiple fact tables

If you want to allow the package users to query against multiple cubes, use virtual cubes.

With virtual cubes you can build a cube that shares dimensions between cubes and build reports that use facts from multiple cubes. One example of this is to have a virtual cube that includes a cube with actual values and a cube with target or budget values.

To the virtual cube, you can build in calculated measures that reference measures from both source cubes. One example of those is a measure that calculates the variance from plan by subtracting the plan values from the actual values. If the remainder is a positive number, then the variance will exceed the plan. This can be beneficial or not, depending on what was
tracked. Another example is a measure that takes that variance and compares it to the plan value to produce a percent of plan measure.

By being built-in, your report authors do not need to create these expressions. This can save their time and the time of users who might want to create these expressions. Also, because they are already there, the report authors do not need to re-create a measure expression that they use in multiple reports.

Because it exists in only one place, the risk of an expression being incorrectly created in one report is removed. There is only one risk point. Enforcing commonality can reduce misunderstanding and misinterpretation of information. In addition, because it is built-in, the performance of the calculation can be faster than a calculation made in a report.

Virtual cubes are covered in greater detail in Chapter 6, “Virtual cubes” on page 133.

4.9 Aggregate cubes

Cognos Dynamic Cubes support aggregate awareness. This awareness is accomplished through aggregate cubes. Aggregate cubes define the measures, dimensions, and dimension grain by which queries can be routed to aggregate tables rather than to the detail fact table. Because aggregate tables store fact data at a higher-than-detail level of granularity, the time necessary to aggregate values during the query can be lessened, thus improving performance. A query can be routed to the aggregate table if all the measures and dimension hierarchies of the query exist in the aggregate cube definition. Not all of the dimensions and measures in the aggregate cube need to be in the query.

The objective of modeling an aggregate cube is to establish rules by which the dynamic cube can know when it can route a query to an aggregate table. This task is done by specifying a mapping from the identifiers in the dimensions and measures in the cube that have scope to the aggregate table, to the identifiers in the aggregate table, and, if necessary, its related tables in a rolled-up dimension schema.

This aggregate cube routing will direct a query only to the aggregate table for a query that uses objects from a dimension grain at or above the grain of the mapping between it and the aggregate table. Therefore, using objects from a grain below the mapping grain does not cause double-counting, because that query continues to route to the detail fact table.

4.9.1 Modeling aggregate cubes

You can use the samples to explore and learn about aggregate cube modeling. The sample database GOSLDW contains one aggregate table. Its type of aggregate table is a degenerate dimension. The name of the aggregate table is AGGR_TIME_PROD_OM_FACT. The fact and dimension information is contained in the one table. The sample Cognos Cube Designer model contains an aggregate cube named gosldw_sales2, which is stored in the gosldw_sales cube.

What you need to know before proceeding

The modeler needs to be aware of the nature of the aggregate table. This information determines what you need to do to model the aggregate cube.

The primary aggregate table scenarios are degenerate dimensions, rolled-up dimensions, parent-child dimensions, custom aggregation, and slicers.
If the aggregate table is a degenerate dimension you need to know the nature of the key in the aggregate table. The nature of the key determines your modeling actions. It is possible that the key allows you to map the aggregate cube to the level keys of dimensions in your cube. It is possible that the key requires you to create a relationship between the aggregate table and a dimension.

If a key in the aggregate table matches the key of a level then you can map the appropriate levels to the aggregate table. For example, if the keys in the aggregate table contain the level keys of a Time dimension's Year and Quarter levels then you can choose to map the Time dimension to the aggregate cube. A query that used Year or Quarter is routed to use the aggregate table. The aggregate table, in relational query terms, is essentially (if not in fact) a view. AGGR_TIME_PROD_OM_FACT is this type of aggregate table.

If a key in the aggregate table matches the key in a level such that the key can be used to identify a relationship between the dimension and the aggregate cube, you can create relationships between the aggregate table and the dimension table objects. The aggregate table becomes an alternate fact table with relationships that are defined between it and its participating dimensions.

It is possible that the aggregate table has associated rolled-up dimension tables. You then need to model the aggregate cube to include them. The aggregate table and its rolled-up dimensions will be a relational schema separate from the schema of the detail fact table and its relational schema.

It is possible that one dimension in the aggregate cube is a parent-child dimension. Parent-child dimensions do not have defined levels. The data of the parent and child attributes determine how the members are generated and assembled into a member tree. The aggregate table has a record for each member. The mapping allows the query to route to the aggregate table.

You need to know if a measure in the cube has custom aggregation. If it does, then your aggregate table needs to support it too. The mapping is similar to a parent-child dimension. Custom aggregation is a method to predefine the aggregation of member values outside of the aggregation functionality of Cognos Dynamic Cubes. For more information about custom aggregation, consult the Cognos Dynamic Cube user guides.

You also need to identify if the aggregate tables use slicers, which are partitioned levels. If you do not include a member that is a slicer in the slicer definition, a query that involves that member will be routed to the detail fact table unnecessarily. If you include a member that is not a slicer in the slicer definition, you can cause the query to route to the aggregate cube when it should not. The query will return a NULL value.

What you need to do for each case
For any particular aggregate table scenario, you need to do several tasks to correctly model the aggregate cube. Some are common to all cases. Some are specific to one case. The following sections describe each of these tasks.
4.9.2 Degenerate dimensions with matching level keys

For the case of degenerate dimensions that have keys that match the level keys of their analogue dimensions, you need to do the following tasks:

- Map the appropriate facts in the aggregate table to those measures in the measure dimension.
- Specify which dimensions participate in the aggregate cube.
- Set the level grain for those hierarchies in each dimension that are involved in the aggregate table.
- Map columns in the aggregate table to the level keys.

4.9.3 Degenerate dimensions with matching join keys

For the case of degenerate dimensions that have keys that match the join keys, you need to do the following tasks:

- Map to the measures.
- Specify which dimensions participate in the aggregate cube.
- Create relationships between the aggregate table and the dimensions.

You do not need to set the level grain for the dimensions. The routing uses the relationship.

4.9.4 Rolled-up dimensions

For rolled-up dimensions, you need to do the following tasks:

- Map to the measures.
- Specify which dimensions participate in the aggregate cube.
- Set the level grain for those hierarchies in each dimension that are involved in the aggregate table.
- Map the key mappings to the keys of the rolled up dimensions.
- Define the relationship between the rolled-up dimension tables and aggregate fact table.

4.9.5 Parent-child dimensions

Parent-child dimensions are treated differently. If you choose to remap the dimension, the members will be mapped to a row in the aggregate table if the key value that is specified in the level unique key value matches a value for a record in the table. In this way, you map the dimension to a table with custom aggregation. You must map a key mapping between the parent-child dimension and aggregate table. The dimension role that you use is the child role.

If you do not choose to remap the dimension, you must define a relationship between the dimension and the aggregate table.

4.9.6 Custom aggregation

The treatment of custom aggregation is similar to that of parent-child dimensions. You need to specify a mapping so that the members in the query can derive their associated measure value for a measure that has custom aggregation from the appropriate records in the aggregate table.
4.9.7 Slicers

You must identify the members that are slicers and add them to the aggregate cube definition by adding them into the slicers tab of the aggregate cube editor.

4.9.8 Other modeling considerations

The objective is to choose the aggregate grain that will match the grain of the dimensions that are contained the aggregate table. If you choose to map the aggregate grain of a dimension to a level above the fact grain of the aggregate table, the query will still route to the aggregate cube but then some aggregation will still have to be performed by the query engine. For example, AGGR_TIME_PROD_OM_FACT has two columns that map to level keys of the time dimension. They map to the level keys of the year and quarter levels of that dimension. Although you could set the aggregate grain of the time dimension to be years, this might not be the best approach because it would preclude some queries from routing to an aggregate cube, defeating the purpose of the aggregate cube.

If you set the aggregate grain to below the grain of the aggregate table, you need to map level keys to objects that do not exist in the aggregate table, which is quite difficult to do. If your application needs to map that grain, the aggregate table must be modified to include that grain.

Be sure to understand the concept of level keys. The keys of all the levels that you set as included in the aggregate grain must be defined in the aggregate table. That is why AGGR_TIME_PROD_OM_FACT, which is the aggregate table that is included in the Cognos sample relational data base GOSLDW, does not have only the quarter level keys but also the year level keys. If your level key needs to include keys from higher levels to uniquely identify the members in the level there is no need to map the key twice. Although the object is used more than once, there will be only one reference of the object to map.

Exploring the aggregate cube

Use the following steps to explore the aggregate cube:

1. Select the gosldw_sales cube.
2. Expand the Project Explorer tree node for the cube or right-click Open Editor and click the Aggregates tab. You see an aggregate cube named gosldw_sales2.
3. Select gosldw_sales2 and right-click Open Editor. You see three dimensions: Time, Order method, and Products.
4. Select Time.
   To the side, you see a list of hierarchies and levels. In the case of Time, only one hierarchy is listed, because it has only one hierarchy. If a dimension has more than one hierarchy, probably only one actually participates in the aggregate table, because having more than one hierarchy can add to the tuple, which in turn has an effect. Several levels show check boxes that are selected: Year and Quarter levels. This selection indicates that the aggregate level grain for the Time dimension is the Quarter level.
5. Deselect the Year check box.
   The Quarter check box is cleared also, which indicates that you must include the keys of higher dimension grains in the aggregate cube to correctly route the query.
6. Click the Quarter check box. The Year check box also becomes selected.
7. Examine the other two dimensions.
   Notice that Order Method has only one level. It might seem pointless to include Order
   Method in the aggregate table, but actually it allows for a richer, more versatile aggregate
   table. The set of queries that can be routed to the aggregate table is enlarged.

8. Click the Measures button.
   You see two measures, Quantity and Revenue. In the mapping column, you see the
   columns in the aggregate table that map to these measures. They are QUANTITY and
   SALE_TOTAL.

9. Hover the mouse on one of the cells in the mapping column.
   The aggregate table name and the column name appear in a tool tip. This step enables
   you to trace the object mapping back to the source. For example, the tool tip for the
   mapping for the measure Quantity displays the following text:
   AGGR_TIME_PROD_OM_FACT.QUANTITY.

10. Return to the Aggregate cube editor and click the Level mapping tab. You see a list of
    level keys, their source dimensions, and the aggregate table mapping.

11. Hover the mouse on one of the cells in the mapping column.
    The aggregate table name and the column name appear in a tool tip. This step enables
    you to trace the object mapping back to the source. Because
    AGGR_TIME_PROD_OM_FACT is a degenerate dimension with level keys, it is sufficient
    to map the cube to gosldw_sales2 with the level key mapping.

12. Return to the Aggregates tab.

13. Click the relationship edit links for any of the dimensions. The Relationship Editor opens. A
    message indicates that no joins are necessary.

14. Click Cancel to close the editor.

15. Deselect the Dimension Grain check boxes for the dimension.

16. Click the Relationship edit links for that dimension.
    You see that the message is gone. If you need to create a mapping to an aggregate table
    through matching join keys, click the columns for the dimension and the aggregate table
    and choose the keys to form a relationship. You do not need to specify level grains.

17. Click Cancel to close the Relationship Editor.

18. Click Undo to restore the level grain or manually click the level grain that was being used.

19. Click the New Dimension button. A dialog opens, which lists the dimensions that exist in
    the cube but do not yet exist in the aggregate cube.

20. Click any of them and click OK. The dimension is added to the aggregate cube.

21. Select the new dimension and click Delete. The dimension is removed from the aggregate
    cube.

22. Click the Slicers tab.
    If you needed to add slicers, expand the member browser for the dimension, select them,
    and add them to the slicer list.

23. Click the Implementation tab.

    The Implementation tab shows a diagram representation of the aggregate cube. Because
    AGGR_TIME_PROD_OM_FACT is a degenerate dimension, the diagram is fairly simple:
    there is only one table in the aggregate cube. If the aggregate cube contained rolled-up
    dimensions, they would be presented in the implementation diagram also.
4.9.9 Automatic aggregate cube creation

You can use the aggregate cube auto-matching functionality to create an aggregate cube automatically. You can do this by dragging the aggregate table onto the aggregates field of the aggregates tab of the cube that you are working on. Where matching measures and dimensions are found in the cube, Cognos Cube Designer maps each of these items to the aggregate table. Where possible, it also attempts to identify the highest level of aggregation required and roll up dimensions. Inspect the aggregate cube and refine it to ensure that it is modeled exactly as you need it to be.

The ability to automatically map depends on how the aggregate tables are set up. A naming convention for aggregate cube columns that conforms them to their analogue source objects is helpful. Complete the following steps:

1. Select the gosldw_sales cube.
2. Expand the Project Explorer tree node for the cube or right-click Open Editor and click the Aggregates tab.
   
   You see an aggregate cube called gosldw_sales2.
3. Expand the metadata tree in the Data Source Explorer and select AGGR_TIME_PROD_OM_FACT.
4. Drag AGGR_TIME_PROD_OM_FACT onto the Aggregate Cube Editor.
5. A new aggregate cube is created. Examine the dimensions, measures, levels, and level key mappings that are created in the aggregate cube.

Cognos Dynamic Query Analyzer has an Aggregate Advisor, which can identify, from the model or from log files, possible candidates for the creation of aggregate tables. For more information about the Aggregate Advisor, see Chapter 8, “Optimization and performance tuning” on page 201 and the Cognos Dynamic Query Analyzer documentation.

4.10 Validation

Cognos Cube Designer can monitor you, as you model and analyze the model. The validation checks for model integrity and attempts to assist the modeler to follow modeling best practices. The results of this validation are displayed in the Issues tab in the property view. If issues are identified, an icon opens on the object that has the issue and also all parent objects. This feature helps you ensure that your chances of seeing that an issue has been generated are good.

There are issues with a severity of error and issues with a severity of warning. The former prevents the modeler from being successful with the model. The latter is not necessarily a problem but, as the modeler, you must attend to it and verify whether you must make a modeling change.

Some examples of the model integrity validation checks include identifying if levels do not have attributes in them, identifying if levels are not used in a hierarchy, identifying if an attribute does not have an expression or object defining it. Other tests include checking to see if the relationship between a dimension and the measure dimension are defined.

One test of modeling best practices is to ensure that the relationship cardinality flows down to the fact table. If an object that has a relationship to another object in the dimension goes against this flow, Cognos Cube Designer flags it as an issue.
4.11 Deploying dynamic cubes for reporting and analysis

This section explains how to quickly deploy or manually deploy Cognos Dynamic Cubes.

4.11.1 Quick-deploy options in Cognos Cube Designer

After you finish modeling, you can publish your cube to Cognos Connection in your IBM Cognos BI Server environment so that report authors and others can begin their reporting and analysis. To do this, right-click the cube you want to deploy from the Project Explorer and select **Publish**.

The resulting dialog, the Publish dialog (Figure 4-12) gives you the option of publishing the cube as a data source, or leveraging some of the quick-deploy options. To perform all of the required steps in Cognos Cube Designer, click **Select all options** in the Publish dialog. Selecting all of the available options will automatically perform the various actions required to enable your cube for reporting. The operation attempts to deploy the cube to content store as a Dynamic Cube data source, start the cube, and then publish a package based on the cube to your My Folders location in Cognos Connection.

![Publish dialog](image)

You may select as many, or as few, of the quick-deploy options you want from the Publish dialog. All of the steps are required, so any option that is deselected here must be accomplished manually in Framework Manager or in Cognos Connection.

After this process completes, begin your reporting and analysis on the dynamic cube package. If you then want to make this dynamic cube package available to other report authors or users, you may copy the package and paste it into the location of your choice in Public Folders in Cognos Connection. The package contains a reference to the underlying cube and data source, so copying or moving the package will not affect the link between the package and the cube.

4.11.2 Manually deploying a dynamic cube

Another way of deploying your dynamic cube to is to manually deploy the cube from Cognos Cube Designer and create and publish a dynamic cube package using Cognos Framework Manager.
Although a number of steps are required to manually deploy your cube, this approach might be the correct approach to use if you are publishing to an environment other than a development or test environment. Acceptance testing or production environments are tightly controlled by administrators, so using the quick-deploy options might not be permitted.

The first required step is to publish the cube from Cognos Cube Designer without the additional options selected. This approach creates a cube data source in Cognos Connection and publishes your cube model to the Cognos content store. The Publish dialog contains an option that must be set if your environment has security defined. Most environments, other than a sandbox environment, have defined some kind of security. The cube you publish to Cognos Connection requires an account to be specified if security is defined. If you do not set it, the cube will not start. You can have Cognos Cube Designer associate your account with the cube by clicking the Associate my account and signon with the cube datasource check box. You must do an additional step of generating credentials. You do that by clicking the First, you must create the credentials link in the Personal tab of the preferences dialog in IBM Cognos Connection.

If a cube of the same name exists in the portal, you are prompted to confirm that you want to continue to deploy the cube and overwrite the existing cube.

The remaining steps to deploy your dynamic cube are done in the IBM Cognos Administration in Cognos Connection, so you might need to work with your system administrator. Detailed steps to add your cube to a query service and start the cube are in Chapter 5, “Administering dynamic cubes” on page 115.

After the cube is started, you then create a package in IBM Cognos Framework Manager to publish to the content store to make the cube available for reporting and analysis. To do this, create a new Framework Manager project and when prompted for a data source to use, select the dynamic cube data source that was created during the publish step in Cognos Cube Designer. The data source will inherit the name of the cube you published. The cube will be imported into Framework Manager as a cube object, so you cannot expand the cube to see the dimensions and levels in the cube. Create a package and add the cube object and then publish the cube to the location of your choice in Cognos Connection. This process is identical to creating a Framework Manager package using any other supported OLAP source such as IBM Cognos TM1 or IBM Cognos Transformer PowerCubes.

4.12 Troubleshooting

This section provides information for troubleshooting metadata, publishing errors, and members.

4.12.1 Importing metadata

Cognos Cube Designer models consist of a model file plus one or more metadata files. For each schema that is imported into the model and used, a metadata file is created in the data directory. The data directory is located in the directory where Cognos Framework Manager and Cognos Cube Designer are installed.

When you open a model, Cognos Cube Designer attempts to load the metadata that is specified in the data sources that have been used in the model. If it cannot load, it displays the following message in the metadata viewer:

Select sources to include in the package
If you expand the data sources, you might see something similar to Figure 4-13.

![Figure 4-13 Data source connection](Image)

The four properties of a data source connection are the Name, the Content Manager Data Source, the Catalog, and the Schema. The Catalog and Schema might be familiar to you. They are properties derived from the databases that you are working on. Depending on your database, you might have a Catalog. If not, this property has no value. Content Manager Data Source property is the name of the data source in Content Manager. Name is the name of the data source given to it by Cognos Cube Designer when the metadata was imported. It is derived from the Content Manager Data Source name.

You can change the data source Name property to any name you want as long as no other data source in the model has that name. You can change the Content Manager Data Source value to be any value you want. You might want to change the value if you want to switch your model from a test environment to a production environment.

There will be a data source for each schema that you are using. Cognos Cube Designer appends a number to each additional data source if the name already exists as a data source. You can change these names to anything you want.

In the Issues tab you see a message similar to the following message:

The data source great_outdoors_warehouse does not have physical metadata associated with it. The data source needs to be refreshed.

You need to change the values of the data source properties to match your connection. Then, select the data source in the model and choose refresh metadata in the context menu. If everything is successful, the Warning overlay icon no longer draws on the Data source icon.

In your data directory, you see a created relational metadata file. The name has the following structure:

```
relmd_{whatever data source connection name you have defined}_{catalog}_{schema}
```

If you have physical metadata schemas in the model but not all that are used in the model, you see the following message in the Issues tab:

The data source {} does not have physical metadata associated with it. The data source needs to be refreshed.

You also see messages in the Issues tab, identifying objects that do not have valid references. The method for handling that case is the same as the general metadata retrieval case.
4.12.2 Publishing errors

Sometimes, you might see the following message after you publish a cube with all the publish options set to On. This message indicates that the cube startup failed to respond before a timeout.

BMT-MD-6587 The Query Service has not acknowledged that the cube configuration has been received. Please verify the cube configuration and status in the IBM Cognos Administration portal.

The reason for the message might be because of a problem with the cube. For example, if you have a level where members are not unique, the cube will not start. Another reason might be a transient timeout problem.

4.12.3 Missing metadata

If you imported metadata in a previous session of Cognos Cube Designer and cannot find it in the data sources list, a possibility is that the model does not have anything in it that uses that metadata.

4.12.4 Refreshing metadata that changed in the database

One of the databases that you are using for your model might contain changed metadata. Changes can include added or deleted tables and columns and changed table and column properties, such as column data types.

Select the data source and right-click Refresh metadata. Confirm that the changed metadata displays in the metadata viewer. Validate the model. If something that is used in the model was removed, correct the object references so that they point to the replacement objects.

4.12.5 Members

You must refresh the members every time you make a change to a dimension to view the effect of the change. To refresh members, either select the members folder and right-click Refresh members or select the dimension and right-click Refresh members.

If you made a change to any hierarchy, you must refresh members for the members folder for any other hierarchy in the dimension.

4.12.6 Missing members

If a null exists in a source column for an attribute that is being used in the level key, some members cannot be generated. If the null happens in a member caption, the member caption displays NULL as its caption, even if other elements of the expression that are used to define the member caption are not null.

To correct this problem, change the attribute expression to handle nulls.

4.12.7 Ellipses in the project viewer tree

Sometimes the project viewer tree might display ellipses for some parts of the project. To force the tree to generate correctly, resize the project viewer tree.
4.12.8 The No Protocol message while running Cognos Cube Designer

If you launch Cognos Cube Designer from the .EXE file directly from the Framework Manager install directory, you can encounter problems. Error messages such as No Protocol are displayed. You must run Cognos Cube Designer from the shortcut that is created during the installation. The shortcut has the Start in value set to run in the BIN directory. Always start Cognos Cube Designer from the shortcut in the Start menu or from the Tools menu in Framework Manager.

4.13 Creating a sample cube model by using GOSLDW

Throughout this book, a model is used to illustrate points and to help you to gain experience with Cognos Cube Designer. This model is similar to the sample Cognos Cube Designer model that is included with the IBM Cognos BI Samples. If you want to follow along with the activities in this chapter, you can also use the Cognos Cube Designer sample model.

Use the following steps to create a sample cube model:
1. Run Cognos Cube Designer.
2. Choose to create a new model from metadata.
3. Import the GOSLDW schema from the sample GOSLDW.
4. Select the model folder.
5. Right-click New cube.
6. Rename the cube to: gosldw_sales
7. Select the cube and right-click Open Editor.
8. Click Measures.

Figure 4-14 shows the dialog for creating a sample cube model.

10. Drag Quantity, Unit cost, Unit price, Unit sale price, Revenue, and Gross profit to the Measures dimension editor. Figure 4-15 on page 96 shows the metadata viewer.
11. Select each measure and format them according to the Measures table for gosldw_sales, as shown in Figure 4-16 on page 97.
4.13.1 Measures of gosldw_sales

Table 4-2 provides the measures for gosldw_sales.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Source column</th>
<th>Regular aggregate</th>
<th>Format</th>
<th>Format value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>SLS_SALES_FACT.QUANTITY</td>
<td>Sum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit cost</td>
<td>SLS_SALES_FACT.UNIT_COST</td>
<td>Calculated</td>
<td>Number of decimal places</td>
<td>2</td>
</tr>
<tr>
<td>Unit sale price</td>
<td>SLS_SALES_FACT.UNIT_PRICE</td>
<td>Calculated</td>
<td>Number of decimal places</td>
<td>2</td>
</tr>
<tr>
<td>Revenue</td>
<td>SLS_SALES_FACT.SALE_TOTAL</td>
<td>Sum</td>
<td>Number of decimal places</td>
<td>2</td>
</tr>
<tr>
<td>Gross profit</td>
<td>SLS_SALES_FACT.GROSS_PROFIT</td>
<td>Sum</td>
<td>Number of decimal places</td>
<td>2</td>
</tr>
</tbody>
</table>
How to create a dimension

Use the following steps to create a dimension:

1. Select the model folder.
2. Right-click **New dimension**.
3. Rename it to: **Time**
4. Expand the Project Explorer tree. Rename the hierarchy to: **Time**
5. Select the hierarchy and right-click **Open Editor**. The Hierarchy Editor opens (Figure 4-17). Note that there is a already a level in the hierarchy.

![Creating a dimension](image)

*Figure 4-17  Creating a dimension*

6. Click **New Level** three times.
7. Rename the levels in the order that the levels appear in the table for specifying the metadata for the Time dimension.
8. Select the **Year** level and right-click **Open editor**.
9. Drag the columns listed in the table for the Year level into the Year level. These items are the attributes of the Time dimension Year level. In this case, there is only one, as shown in Figure 4-18.

![Figure 4-18 Renaming the levels](image)

10. Click the **Member caption** radio button for the attribute that is listed as the member caption.

11. Click the level **Unique key** check box for the attributes that are listed in the level key.
12. Click **Level Key**. The Level Key editor opens (Figure 4-19). For this initial set of dimensions, only one attribute is in the level key. Notice that it has a **key** icon beside it. That icon indicates that this attribute is the business key. For more information about modeling level keys, see 4.3.4, "Modeling level keys" on page 75.

![Level Key editor](image)

**Figure 4-19  Level Key editor**

13. Confirm that the level key is defined.
14. Click **OK**.
15. Click member sort dialog.
16. Drag the attribute listed in the Member sort row of the dimension specification into the member sort control.
17. Set the member sort direction value to be **Ascending**.
18. Click **OK** to save the member sorting.
19. Repeat the process for each other level in the dimension. Figure 4-20 shows the completed dialog. Note that the dimension type property is Regular.

4.14, “Creating a time dimension” on page 112 shows how to convert Time to a Time dimension.

![Figure 4-20 Creating dimension levels](image)

20. Repeat the process for each other dimension in the table.

21. Select the gosldw_sales cube.

22. Right-click Open Editor.

23. Drag each of the dimensions you created into the cube.

### 4.13.2 Dimensions in the gosldw_sales cube

This following sections provide the attributes and source columns for each of the dimensions in the gosldw_sales cube.
## 4.13.3 Time

Table 4-3 provides the attributes and columns for Year.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM_CURRENT_YEAR</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM_CURRENT_YEAR</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td>CURRENT_YEAR, Ascending</td>
</tr>
</tbody>
</table>

Table 4-4 provides the attributes and columns for Quarter.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM_QUARTER_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM_QUARTER_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td>GO_TIME_DIM_CURRENT_QUARTER</td>
</tr>
<tr>
<td>Member sort</td>
<td>QUARTER_KEY, Ascending</td>
</tr>
</tbody>
</table>

Table 4-5 provides the attributes and columns for Month.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM_MONTH_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM_MONTH_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td>GO_TIME_DIM_CURRENT_MONTH,</td>
</tr>
<tr>
<td></td>
<td>GO_TIME_DIM_MONTH_NUMBER,</td>
</tr>
<tr>
<td></td>
<td>GO_TIME_DIM_DAYS_IN_MONTH</td>
</tr>
<tr>
<td>Member sort</td>
<td>MONTH_KEY, Ascending</td>
</tr>
</tbody>
</table>

Table 4-6 provides the attributes and columns for Day.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM_DAY_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM_DAY_KEY</td>
</tr>
<tr>
<td>Other attributes</td>
<td>GO_TIME_DIM_DAY_OF_WEEK, GO_TIME_DIM_DAY_OF_MONTH,</td>
</tr>
<tr>
<td></td>
<td>GO_TIME_DIM_DAY_OF_YEAR, GO_TIME_DIM_WEEKDAY_EN</td>
</tr>
<tr>
<td>Member sort</td>
<td>DAY_KEY, ASCENDING</td>
</tr>
</tbody>
</table>
4.13.4 Employee by region

Table 4-7 provides the attributes and columns for Region.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_REGION_DIM.REGION_CODE</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_REGION_DIM.REGION_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td>REGION_CODE, ASCENDING</td>
</tr>
</tbody>
</table>

Table 4-8 provides the attributes and columns for Country.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_REGION_DIM.COUNTRY_CODE</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_REGION_DIM.COUNTRY_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td>COUNTRY_CODE, ASCENDING</td>
</tr>
</tbody>
</table>

Table 4-9 provides the attributes and columns for Employee.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>EMP_EMPLOYEE_DIM.EMPLOYEE_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>EMP_EMPLOYEE_DIM.EMPLOYEE_NAME</td>
</tr>
<tr>
<td>Other attributes</td>
<td>All other columns from EMP_EMPLOYEE_DIM</td>
</tr>
<tr>
<td>Member sort</td>
<td>EMPLOYEE_KEY, ASCENDING</td>
</tr>
</tbody>
</table>

4.13.5 Promotions

Table 4-10 provides the attributes and columns for Campaign.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>MRK_PROMOTION_DIM.CAMPAIGN_CODE</td>
</tr>
<tr>
<td>Member Caption</td>
<td>MRK_CAMPAIGN_LOOKUP.CAMPAIGN_NAME_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td>CAMPAIGN_CODE, Ascending</td>
</tr>
</tbody>
</table>
Table 4-11 provides the attributes and columns for Promotion.

Table 4-11  Promotion

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>MRK_PROMOTION_DIM.PROMOTION_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>MRK_PROMOTION_DIM.PROMOTION_NAME_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td>MRK_PROMOTION_DIM.PROMOTION_CODE</td>
</tr>
<tr>
<td>Member sort</td>
<td>PROMOTION_KEY, Ascending</td>
</tr>
</tbody>
</table>

4.13.6 Order method

Table 4-12 provides the attributes and columns for Order method.

Table 4-12  Order method

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>SLS_ORDER_METHOD.ORDER_METHOD_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>SLS_ORDER_METHOD.ORDER_METHOD_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td>ORDER_METHOD_KEY, Ascending</td>
</tr>
</tbody>
</table>

4.13.7 Retailers

Table 4-13 provides the attributes and columns for Region.

Table 4-13  Region

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_REGION_DIM REGION_CODE</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_REGION_DIM.REGION_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td>REGION_CODE, Ascending</td>
</tr>
</tbody>
</table>

Table 4-14 provides the attributes and columns for Retailer country.

Table 4-14  Retailer country

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_REGION_DIM.COUNTRY_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_REGION_DIM.COUNTRY_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td>COUNTRY_KEY, Ascending</td>
</tr>
</tbody>
</table>
Table 4-15 provides the attributes and columns for Retailer name.

**Table 4-15  Retailer name**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>SLS_RTL_DIM.RETAILER_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>SLS_RTL_DIM.RETAILER_NAME</td>
</tr>
<tr>
<td>Other attributes</td>
<td>SLS_RTL_DIM.RETAILER_NAME_MB, SLS_RTL_DIM.RETAILER_CODE</td>
</tr>
<tr>
<td>Member sort</td>
<td>RETAILER_KEY, Ascending</td>
</tr>
</tbody>
</table>

Table 4-16 provides the attributes and columns for Retailer site.

**Table 4-16  Retailer site**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>SLS_RTL_DIM.RETAILER_SITE_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>SLS_RTL_DIM.RTL_CITY</td>
</tr>
<tr>
<td>Other attributes</td>
<td>All other unused columns in SLS_RTL_DIM</td>
</tr>
<tr>
<td>Member sort</td>
<td>RETAILER_SITE_KEY, Ascending</td>
</tr>
</tbody>
</table>

**4.13.8 Time (close date)**

Table 4-17 provides the attributes and columns for Year (close date).

**Table 4-17  Year (close date)**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM.CURRENT_YEAR</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM.CURRENT_YEAR</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td></td>
</tr>
</tbody>
</table>

Table 4-18 provides the attributes and columns for Quarter (close date).

**Table 4-18  Quarter (close date)**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM.QUARTER_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM.QUARTER_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td>GO_TIME_DIM.CURRENT_QUARTER</td>
</tr>
<tr>
<td>Member sort</td>
<td>QUARTER_KEY, Ascending</td>
</tr>
</tbody>
</table>
Table 4-19 provides the attributes and columns for Month (close date).

**Table 4-19   Month (close date)**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM.MONTH_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM.MONTH_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td>GO_TIME_DIM.CURRENT_MONTH,</td>
</tr>
<tr>
<td></td>
<td>GO_TIME_DIM.MONTH_NUMBER,</td>
</tr>
<tr>
<td></td>
<td>GO_TIME_DIM.DAYS_IN_MONTH</td>
</tr>
<tr>
<td>Member sort</td>
<td>MONTH_KEY, Ascending</td>
</tr>
</tbody>
</table>

Table 4-20 provides the attributes and columns for Day (close date).

**Table 4-20   Day (close date)**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM.DAY_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM.DAY_KEY</td>
</tr>
<tr>
<td>Other attributes</td>
<td>GO_TIME_DIM.DAY_OF_WEEK,</td>
</tr>
<tr>
<td></td>
<td>GO_TIME_DIM.DAY_OF_MONTH,</td>
</tr>
<tr>
<td></td>
<td>GO_TIME_DIM.DAY_OF_YEAR,</td>
</tr>
<tr>
<td></td>
<td>GO_TIME_DIM.WEEKDAY_EN</td>
</tr>
<tr>
<td>Member sort</td>
<td>DAY_KEY, ASCENDING</td>
</tr>
</tbody>
</table>

### 4.13.9 Time (ship date)

Table 4-21 provides the attributes and columns for Year (ship date).

**Table 4-21   Year (ship date)**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM.CURRENT_YEAR</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM.CURRENT_YEAR</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td></td>
</tr>
</tbody>
</table>

Table 4-22 provides the attributes and columns for Quarter (ship date).

**Table 4-22   Quarter (ship date)**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM.QUARTER_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM.QUARTER_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td>GO_TIME_DIM.CURRENT_QUARTER</td>
</tr>
<tr>
<td>Member sort</td>
<td>QUARTER_KEY, Ascending</td>
</tr>
</tbody>
</table>
Table 4-23 provides the attributes and columns for Month (ship date).

\textit{Table 4-23}  \textit{Month (ship date)}

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM_MONTH_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM_MONTH_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td>GO_TIME_DIM_CURRENT_MONTH, GO_TIME_DIM_MONTH_NUMBER, GO_TIME_DIM_DAYS_IN_MONTH</td>
</tr>
<tr>
<td>Member sort</td>
<td>MONTH_KEY, Ascending</td>
</tr>
</tbody>
</table>

Table 4-24 provides the attributes and columns for Day (ship date).

\textit{Table 4-24}  \textit{Day (ship date)}

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level Key</td>
<td>GO_TIME_DIM_DAY_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM_DAY_KEY</td>
</tr>
<tr>
<td>Other attributes</td>
<td>GO_TIME_DIM_DAY_OF_WEEK, GO_TIME_DIM_DAY_OF_MONTH, GO_TIME_DIM_DAY_OF_YEAR, GO_TIME_DIM_WEEKDAY_EN</td>
</tr>
<tr>
<td>Member sort</td>
<td>DAY_KEY, ASCENDING</td>
</tr>
</tbody>
</table>

4.13.10  Branch

Table 4-25 provides the attributes and columns for Region.

\textit{Table 4-25}  \textit{Region}

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_REGION_DIM_REGION_CODE</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_REGION_DIM_REGION_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td>REGION_CODE, Ascending</td>
</tr>
</tbody>
</table>

Table 4-26 provides the attributes and columns for Country.

\textit{Table 4-26}  \textit{Country}

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_REGION_DIM_COUNTRY_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_REGION_DIM_COUNTRY_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td>COUNTRY_KEY, Ascending</td>
</tr>
</tbody>
</table>
Table 4-27 provides the attributes and columns for Branch.

**Table 4-27  Branch**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>GO_BRANCH_DIM.BRANCH_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_BRANCH_DIM.CITY</td>
</tr>
<tr>
<td>Other attributes</td>
<td>GO_BRANCH_DIM.ADDRESS1, GO_BRANCH_DIM.ADDRESS2, GO_BRANCH_DIM.PROV_STATE, GO_BRANCH_DIM.ADDRESS1_MB, GO_BRANCH_DIM.ADDRESS2_MB, GO_BRANCH_DIM.CITY_MB, GO_BRANCH_DIM.PROV_STATE_MB, GO_BRANCH_DIM.POSTAL_ZONE, GO_BRANCH_DIM.WAREHOUSE_BRANCH_CODE, GO_BRANCH_DIM.BRANCH_CODE</td>
</tr>
<tr>
<td>Member sort</td>
<td>BRANCH_KEY, Ascending</td>
</tr>
</tbody>
</table>

**4.13.11 Product brand**

Table 4-28 provides the attributes and columns for Product brand.

**Table 4-28  Product brand**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>PRODUCT_BRAND_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>SLS_PRODUCT_BRAND_LOOKUP.PRODUCT_BRAND_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td>PRODUCT_BRAND_CODE</td>
</tr>
<tr>
<td>Member sort</td>
<td>PRODUCT_BRAND_KEY, Ascending</td>
</tr>
</tbody>
</table>

**4.13.12 Product name**

Table 4-29 provides the attributes and columns for Product name.

**Table 4-29  Product name**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>BASE_PRODUCT_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>PRODUCT_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td>BASE_PRODUCT_KEY, Ascending</td>
</tr>
</tbody>
</table>
4.13.13 Products

Table 4-30 provides the attributes and columns for Product line.

**Table 4-30  Product line**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>SLS_PRODUCT_DIM.PRODUCT_LINE_CODE</td>
</tr>
<tr>
<td>Member Caption</td>
<td>SLS_PRODUCT_LINE_LOOKUP PRODUCT_LINE</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td>PRODUCT_LINE_CODE, ASCENDING</td>
</tr>
</tbody>
</table>

Table 4-31 provides the attributes and columns for Product line.

**Table 4-31  Product type**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>SLS_PRODUCT_DIM.PRODUCT_TYPE_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>SLS_PRODUCT_TYPE_LOOKUP.PRODUCT_TYPE</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td>PRODUCT_TYPE_KEY, ASCENDING</td>
</tr>
</tbody>
</table>

Table 4-32 provides the attributes and columns for Product.

**Table 4-32  Product**

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level key</td>
<td>SLS_PRODUCT_DIM.BASE_PRODUCT_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>VIEW_PRODUCT_NAME.PRODUCT_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td>BASE_PRODUCT_NUMBER</td>
</tr>
<tr>
<td>Member sort</td>
<td>BASE_PRODUCT_KEY, Ascending</td>
</tr>
</tbody>
</table>

You now need to create a join between VIEW_PRODUCT_NAME and SLS_PRODUCT_DIM. Use the following steps.

1. Open the dimension editor for Products.
2. Click the **Implementation** tab.
3. In the implementation diagram, press Ctrl and select the title bars of the VIEW_PRODUCT_NAME and SLS_PRODUCT_DIM tables.
4. Click the **Create Join** icon. See Figure 4-21.

![Figure 4-21 Creating a join between VIEW_PRODUCT_NAME and SLS_PRODUCT_DIM](image)

5. Choose **PRODUCT_NUMBER** to be the key on both sides of the join (Figure 4-22).

![Figure 4-22 PRODUCT_NUMBER is a key](image)

6. Set the cardinality to: **One to One**

7. Click **OK**.
8. Select the relationship that was created, and verify that the cardinality shows as 1 on both ends of the relationship and that PRODUCT_NUMBER is highlighted in both tables, as shown in Figure 4-23.

![Figure 4-23 Verifying the cardinality](image)

Table 4-33 describes the relationships between the keys.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Dimension key</th>
<th>Measure dimension key</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>GO_TIME_DIM.DAY_KEY</td>
<td>DAY_KEY</td>
</tr>
<tr>
<td>Employee by region</td>
<td>EMPLOYEE_DIM.EMPLOYEE_KEY</td>
<td>EMPLOYEE_KEY</td>
</tr>
<tr>
<td>Products</td>
<td>PRODUCT_KEY</td>
<td>PRODUCT_KEY</td>
</tr>
<tr>
<td>Sales Order</td>
<td>SALES_ORDER_KEY</td>
<td>SALES_ORDER_KEY</td>
</tr>
<tr>
<td>Promotions</td>
<td>PROMOTION_KEY</td>
<td>PROMOTION_KEY</td>
</tr>
<tr>
<td>Order method</td>
<td>ORDER_METHOD_KEY</td>
<td>ORDER_METHOD_KEY</td>
</tr>
<tr>
<td>Retailers</td>
<td>RETAILER_SITE_KEY</td>
<td>RETAILER_SITE_KEY</td>
</tr>
<tr>
<td>TIME (close date)</td>
<td>DAY_KEY</td>
<td>CLOSE_DAY_KEY</td>
</tr>
<tr>
<td>Time (ship date)</td>
<td>DAY_KEY</td>
<td>SHIP_DAY_KEY</td>
</tr>
<tr>
<td>Branch</td>
<td>EMPLOYEE_KEY</td>
<td>EMPLOYEE_KEY</td>
</tr>
<tr>
<td>Product Brand</td>
<td>PRODUCT_KEY</td>
<td>PRODUCT_KEY</td>
</tr>
</tbody>
</table>
4.14 Creating a time dimension

Use the following steps to create a time dimension:
1. Select the TIME dimension in the project explorer.
2. Set the Dimension type value in the Property panel to: Time
3. Select the Time hierarchy.
4. Set the Show relative time members value in the property viewer to be true.
5. Select the GO_TIME_DIM table in the Data Source Explorer.
6. Right-click View Data.
7. View the data in the CURRENT_YEAR column. CURRENT_YEAR is the business key of the Year level of the Time dimension. The values in CURRENT_YEAR will generate members in the Cognos Dynamic Cube.
8. Choose a year to set as the current period.
9. Select the year level in the Time dimension.
10. Set the level type property to: year
    For the other levels, set the level type to match their roles.
11. Click the Current period field in the Property panel.
12. Enter the value that you choose to be the current period. You have now set the current period value for the year level.
13. View the CURRENT_QUARTER column in the Tabular Data tab.
    The data is in the form of 20121, 20122, 20123, and 20124.
14. Decide on the current quarter period that you want.
15. Select the Quarter level in the Time dimension.
16. Click the Current period field in the Property panel.
17. Enter the value that you choose to be the current period.
18. Repeat these steps to determine the current month and current day.

4.15 Modeling for multi-grain scenarios

Use the following steps to model multi-grain scenarios.
1. Launch Cognos Cube Designer.
2. Open the model that you are creating as you progress through this book.
3. Select the TIME dimension in the project explorer.
4. Expand the dimension. It has four levels: Year, Quarter, Month, and day.
5. Select the model namespace.
6. Right-click Create new dimension.
7. Rename it to: Time-Month
8. Model it with the metadata that is listed in Table 4-34 through Table 4-36.

<table>
<thead>
<tr>
<th>Table 4-34   Year</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td></td>
</tr>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM.CURRENT_YEAR</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM.CURRENT_YEAR</td>
</tr>
<tr>
<td>Other attributes</td>
<td></td>
</tr>
<tr>
<td>Member sort</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4-35   Quarter</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td></td>
</tr>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM.QUARTER_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM.QUARTER_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td>GO_TIME_DIM.CURRENT_QUARTER</td>
</tr>
<tr>
<td>Member sort</td>
<td>QUARTER_KEY, Ascending</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4-36   Month</th>
<th>Source column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td></td>
</tr>
<tr>
<td>Level key</td>
<td>GO_TIME_DIM.MONTH_KEY</td>
</tr>
<tr>
<td>Member Caption</td>
<td>GO_TIME_DIM.MONTH_EN</td>
</tr>
<tr>
<td>Other attributes</td>
<td>GO_TIME_DIM.CURRENT_MONTH, GO_TIME_DIM.MONTH_NUMBER, GO_TIME_DIM.DAYS_IN_MONTH</td>
</tr>
<tr>
<td>Member sort</td>
<td>MONTH_KEY, Ascending</td>
</tr>
</tbody>
</table>

9. Create a new cube. Rename it to: gosldw_target

10. Put the SLS_SALES_TARGET_FACT.SALES_TARGET into its measure dimension.

11. Drag Time-Month and Employee by Region into the cube.

12. Open the Cube Editor.

13. Open the relationship between Employee by Region and the measure dimension.

14. Confirm that the relationship uses EMPLOYEE_KEY on both sides of the relationship.

15. Open the relationship between Time-Month and the measure dimension.

16. Confirm that nothing appears on either side of the relationship, because MONTH_KEY is neither a Primary key (PK) nor a Foreign key (FK).

17. Click the drop-down menu for Time-Month and select MONTH_KEY.

18. Click the drop-down menu for Measures and select MONTH_KEY.

19. Click OK.
Administering dynamic cubes

Dynamic cubes must be set up and administered before you can run queries against them. Common administration tasks include assigning the cube to the QueryService instance, starting it, monitoring its health, and refreshing its contents.

This chapter contains the following sections:

- 5.1, “Adding and removing cubes from the QueryService” on page 116
- 5.2, “Starting, stopping, and monitoring cubes” on page 120
- 5.3, “Managing the cache” on page 124
- 5.4, “Setting up routing rules” on page 125
- 5.5, “Setting up redundant cube instances” on page 125
- 5.6, “Scheduling a refresh of the cache” on page 127
5.1 Adding and removing cubes from the QueryService

Dynamic cubes must be assigned to the QueryService before they can be loaded into memory and made available for queries.

5.1.1 Adding a cube to the QueryService

Complete the following steps to use the gosldw_sales cube defined in 4.13.1, “Measures of gosldw_sales” on page 97

1. In IBM Cognos portal, go to the IBM Cognos Administration page. Select the Status tab and click the System item in the list of system commands on the left (Figure 5-1).

![Figure 5-1 System administration tab](image)

2. From the Scorecard view, select All servers → Services → Query (Figure 5-2).

![Figure 5-2 Query menu item](image)
3. The **QueryService** now appears as a link in the Scorecard view. Click it to expose the list of cubes assigned to it. The list is currently empty (Figure 5-3).

![Figure 5-3 QueryService management window](image1)

4. Click the **Properties** icon in the Settings - QueryService view to open the Set properties window (Figure 5-4). To edit cube assignments, click **Edit** in the Value column for Dynamic cube configurations.

![Figure 5-4 QueryService properties](image2)
5. Select the **New configuration** link to open the cube selection dialog (Figure 5-5).

![Figure 5-5 List of assigned dynamic cubes (currently empty)](image)

6. Select the cube that you want to assign to the QueryService, and click the right arrow to make sure the cube appears in the Selected entries list (Figure 5-6).

![Figure 5-6 Cube selection dialog](image)

7. Click **OK** to return to the dynamic cube configurations window. The cube is now in the list of assigned cubes. You can click the pencil icon to change cube-specific properties (Figure 5-7).

![Figure 5-7 List of assigned dynamic cubes (now contains the selected cube)](image)
8. Click **OK** to return to **Set properties – QueryService** view, and then **OK** again to save your settings.

Your cube now appears under the QueryService (Figure 5-8).

**Note:** To see the view update in a timely way, click the **Refresh** icon in the top right corner of the view. It might take several seconds for the cube addition to take effect in the QueryService.

---

5.1.2 Removing a cube from the QueryService

**Note:** To remove the cube from QueryService, stop the cube first.

Removing a cube from the QueryService is similar to the steps for adding a cube:

1. Open the **Set dynamic cube configurations - QueryService** dialog (see 5.1.1, “Adding a cube to the QueryService” on page 116). Select the check box next to the cube that you want to remove and click **Delete** (Figure 5-9).

2. When the cube is deleted from the list, click **OK** until you are back to the QueryService view.
3. Refresh the view by clicking **Refresh** to see that the cube has disappeared from under the QueryService (Figure 5-10).

![Image](image1.png)

*Figure 5-10   After the cube is removed from the QueryService the list becomes empty*

### 5.2 Starting, stopping, and monitoring cubes

This section describes how to start, stop, and monitor cubes.

#### 5.2.1 Starting a cube

Before a cube can be used it has to be started. Use the following steps to start a cube.

1. To start the cube, click the down arrow next to the cube name and select **Start** (Figure 5-11).

![Image](image2.png)

*Figure 5-11   Pop-up menu with available cube administration tasks*
2. When the command is running, a dialog opens (Figure 5-12).

**Note:** This dialog indicates that the command was successfully submitted, but does not mean that it has finished successfully.

![Figure 5-12 Cube started notification dialog](image)

3. When the cube starts successfully, its status changes to **Available**. If the cube start is unsuccessful, or additional information is needed about the command state, use the **View recent messages** command to review details about its state (Figure 5-13). Select the command from the drop-down menu by clicking the down arrow next to the cube name.

![Figure 5-13 Selected “View recent messages” command](image)
A log of recent messages is displayed (Figure 5-14).

![Recent messages](image)

**Figure 5-14  Recent administration messages for the cube**

### 5.2.2 Stopping a cube

To stop a cube, select either of the following commands; both are available by clicking the down-arrow icon next to the cube name:

- **Stop after active tasks complete**: The cube will stop after currently executing queries are finished.
- **Stop immediately**: The cube will stop immediately, without waiting for the active queries and commands to complete. Some user queries can fail as a result.
5.2.3 Monitoring cube state through metrics

When managing dynamic cubes, a good practice is to monitor metrics displayed for each cube in the Metrics window (Figure 5-15).

![Dynamic cube metrics values](image)

In addition to overall values, some metrics also have the value collected in the last hour, which helps you keep track of both historic averages and recent changes to the system.

The following metrics are of particular interest in most cube management scenarios:

- **Average successful request time**
  This metric indicates the average time for a report to execute successfully. It is useful when monitoring the performance of the server and back-end database. Slow request times might indicate performance issues that need to be investigated.

- **Cube state**
  This metric indicates the current cube state: Disabled, Stopped, Starting, Running, and Stopping.

- **Data cache hit rate**
  A higher cache-hit rate indicates better utilization of data cache and better query performance.

- **Last metadata load time**
  This metric shows how long the previous load took for metadata to be loaded or refreshed. Disproportionately long times may indicate modeling or configuration problems.

- **Last response time**
  This metric shows how long the most recent time took to execute any request against this cube. Useful for detecting performance degradation problems.
- Percentage of time spent retrieving data
  This metric indicates the percentage of time data is retrieved from the back end database. Higher percentage means worse performance and fewer cache hits.
- Result set cache hit rate
  A higher hit rate means better result cache use and indicates that the majority of the report results can be reused, improving query times.
- Up time
  This metric indicates the length of time the cube is running.

5.3 Managing the cache

Dynamic Cubes support two types of caches that can be managed by the administrator:

- 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, to update the cube with the latest metadata.

- Data cache
  This cache contains data values that correspond to the current set of cache metadata. This cache can be refreshed when the data values in the source relational data source are changed. In general, data values change more frequently than cube metadata.

Use the following steps to manage the cache:

1. Refresh the member or data cache by clicking the down arrow next to the cube name under QueryService and select Refresh member cache or Refresh data cache (Figure 5-16).

Figure 5-16  Select Refresh member cache
2. Select **View recent messages** to check the progress of the refresh (Figure 5-17). The window indicates either a success message or a failure message, as applicable.

<table>
<thead>
<tr>
<th>Time</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>July 22, 2012 9:12:53 AM</td>
<td>Member cache refresh succeeded</td>
</tr>
<tr>
<td>July 22, 2012 9:12:49 AM</td>
<td>Member cache refresh executing</td>
</tr>
<tr>
<td>July 22, 2012 9:12:49 AM</td>
<td>Cube start succeeded</td>
</tr>
<tr>
<td>July 22, 2012 9:12:47 AM</td>
<td>Cube start executing</td>
</tr>
</tbody>
</table>

*Figure 5-17  Cache refresh status messages*

**Note:** Because data cache is closely associated with the member cache, refreshing the member cache automatically causes the data cache to be refreshed.

### 5.4 Setting up routing rules

If you are setting up your IBM Cognos BI deployment to address applications that are sourced from Dynamic Cubes and other sources in a mixed environment, you must create server groups and dispatcher routing rules to ensure that queries are routed to the correct report server. The reason is because the QueryService that has a dynamic cube enabled is where the metadata and other cached data items are located. If dynamic cube application queries are sent to the wrong report server, an error message is returned rather than the requested data.

For more information about setting up server groups and routing rules, see the “Advanced Dispatcher Routing” topic in the Cognos Business Intelligence Administration and Security Guide at the Cognos 10 Information Center:


### 5.5 Setting up redundant cube instances

To enable load balancing and optimize query performance in high-load environments, you can set up multiple instances of the dispatcher with identical cube configurations.

When the query is submitted by the user, it will be routed to the next available dispatcher, making sure all dispatchers are load-balanced. As your requirements changes, you can add or remove dispatchers from the server group to match your performance needs.

Because queries are routed to different dispatchers based on load, you must ensure that all instances of the dynamic cube on all dispatchers are operating correctly. If a query is routed...
to a cube instance that is down, the query will fail. In such cases, the queries might seem like they are failing randomly, complicating problem diagnostics.

To avoid disrupting query execution, adding and removing cube instances must be done in a controlled way. Before making the instance of the cube in a particular dispatcher available to the server group, ensure that this dispatcher is not a member of the server group. This way, no queries can reach this dispatcher during addition or removal, which will eliminate the risk of disrupting query execution.

After the cube instance is configured and is running, use the following steps to add the dispatcher instance to the server group:

1. **In IBM Cognos Administration**, navigate to the System window (Figure 5-18).

![Figure 5-18 System administration tab](image)

2. In the Scorecard view, click the name of the server that holds the dispatcher instance that you are configuring. Click the down arrow next to the dispatcher address and select **Set properties**, (Figure 5-19).

![Figure 5-19 Dispatcher instance menu](image)
3. In the Properties window, click the **Settings** tab, and scroll to the Server group field (Figure 5-20).

![Dispatcher properties](image)

**Figure 5-20**  Dispatcher properties

4. Enter the name of the server group you want this dispatcher to be a part of, and click **OK**. The dispatcher will now be able to process queries that are routed to the server group.

**Note:** Propagation of information about dispatcher membership in server groups to all affected dispatchers is not instantaneous. You might have to wait up to a minute for the changes to take effect.

The process of removing a dispatcher from the server group is the reverse of the procedure that is described.

Clearing the server group name field in the dispatcher configuration prevents queries for the cubes in that server group from being routed to it. The cube or the dispatcher itself can then be administered or taken down for maintenance.

For more information about server groups and routing tables, see the “Advanced Dispatcher Routing” topic in the Cognos Business Intelligence *Administration and Security Guide* in the Cognos 10 Information Center at the following address:


### 5.6 Scheduling a refresh of the cache

Administration commands often need to be run at predefined times and intervals. For example, the data source from which the dynamic cube is built might be continuously updated, so the cube member and data caches must be updated also.
As an example, we schedule a refresh of a cube metadata to run once a day:

1. In IBM Cognos Administration, click the **Configuration** tab, and click the **New Query service administration task** icon to invoke a drop-down menu. From this menu, you can manage a variety of tasks. Select the **Dynamic cube** menu item to schedule a dynamic cube command (Figure 5-20 on page 127).

![Figure 5-21 Dynamic cube task-scheduling command](image)

2. In the dialog that opens, type in the name for the cache update task that we are creating, for example, **Cache refresh task** (Figure 5-22).

![Figure 5-22 Task name dialog](image)

3. Click **Next**.
4. In the next dialog, select **Refresh member cache** as the command, and select the check box next to the name of the cube that you want to refresh (Figure 5-23).

5. Click **Next**.

6. In the next window, select **Save and schedule** and click **Finish** to create the task (Figure 5-24).
7. In the new dialog that opens, define the schedule of when to run this task (Figure 5-25):
   Click the **By Day** tab and select **Every 1 Hour(s)**. Enter a time period during which you want this task to run. Change the **Start** field to specify a time in the future, for example, the beginning time of the task.

   **Note:** Because administrative commands can increase load on the dynamic cube, a common practice is to schedule maintenance tasks during a time when the cube is not busy, such as late at night.

   ![Schedule definition dialog](image)

8. Click **OK**.

9. The next window shows, in the list, the update task that you created (Figure 5-26).

   ![Content administration window that includes currently scheduled tasks](image)
10. Wait for the time of the scheduled task to pass, and click the More link next to the task. A window that lists available task actions opens (Figure 5-27).

![Figure 5-27 Task actions window](image)

11. From the list of available commands, select View run history. The record of when the task was last run opens and shows the status (Figure 5-28).

![Figure 5-28 Task execution history](image)

12. Click Close.
Virtual cubes

Virtual cubes can be used to drastically improve the response time of the cube server queries by using efficient data partitioning for optimum cache utilization. Virtual cubes offer a solution for combining results by merging different regional cubes into a country cube. They also enable merging sales numbers with currency exchange rates to provide a global view of the business. This chapter explains how to create virtual cubes, how they work, and how to manage them.

This chapter contains the following sections:

- 6.1, “Overview of virtual cubes” on page 134
- 6.2, “Creating virtual cubes” on page 134
- 6.3, “Common use cases for virtual cubes” on page 140
- 6.4, “Managing virtual cubes” on page 141
- 6.5, “Virtual cube considerations” on page 144
### 6.1 Overview of virtual cubes

A virtual cube is a logical cube that is defined by merging two existing cubes. You can use virtual cubes to join two cubes with the same structure to facilitate data management, or to join two cubes that share only some dimensions to make calculations on the shared data. You can merge two source cubes, two virtual cubes, or one virtual cube and one source cube. The resulting virtual cubes can also be merged with other source cubes or virtual cubes.

**Note:** The two cubes that are merged can belong to separate data sources, so they might have completely different internal structures.

For example, Figure 6-1 shows cubes that are combined to form virtual cubes. Cube A and Cube B are merged to form the Virtual Cube AB. Then, Virtual Cube AB and Cube C are merged to form Virtual Cube ABC.

![Sample virtual cube](image)

### 6.2 Creating virtual cubes

This section describes how to create virtual cubes. Use the following steps to create a new virtual cube:

1. Open IBM Cognos Cube Designer and select a namespace in the Project Explorer tree.
2. Right-click the namespace and select **New → Virtual Cube**. The New Virtual Cube wizard starts (Figure 6-2 on page 135). This wizard can also be launched by clicking the **New Virtual Cube** icon in the Project Explorer panel.
3. In the Select Base Cubes window (Figure 6-3), 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, and then click OK.
4. Click **OK**.

5. To open the definitions of the new cube in the **Properties** tab, double-click **New Virtual Cube**, or from the Project Explorer tree, right-click the virtual cube and select **Open Editor**. Figure 6-4 shows the Virtual Cube Editor.

![Figure 6-4 Virtual Cube Editor](image)

Merging of the cubes is done based on dimension names. Dimensions with the same name in both cubes are merged to become virtual dimensions. The dimensions from one cube that do not have a corresponding dimension with the same name in the other cube are not merged together, but instead are added to the virtual cube.

Hierarchies, levels, members, and measures (source objects) all follow these same described rules regarding merging. For all of these, when the two source object names are identical, the two objects will be merged together. When the names differ, two separate objects will be added to the virtual cube.

Complete the following steps to see the measures included in the virtual cube:

1. Open Cognos Cube Designer and go to the Project Explorer tree.
2. Locate the virtual cube in the project.
3. Double-click the virtual cube’s Measure folder or click the **Measure** icon in the cube editor (Figure 6-5).

![Figure 6-5 Open Virtual Measures](image)

When a query is run against a virtual cube, it is routed to the dependent cubes, which produces two intermediary results that are aggregated according to the virtual cube’s merge operator. Figure 6-4 on page 136 shows the virtual cube **Properties** tab. In that section, you can change the virtual cube merge operator. The following merge operators are available:

- Sum
- Subtract
- Multiply
- Divide
- Maximum
- Minimum
- None

### 6.2.1 Deploying virtual cubes

After you finish modeling, you can publish your cube to Cognos Connection in your IBM Cognos BI Server environment so that report authors and others can begin their reporting and analysis. To publish, right-click the cube you want to deploy, from the Project Explorer, and select **Publish**.

When you finish modeling a dynamic cube in Cognos Cube Designer, you can deploy it as an OLAP data source to Content Manager. Right-click the virtual cube you want to deploy from the Project Explorer and select **Publish**. Additional options to available to deploy a cube are described in 4.11, “Deploying dynamic cubes for reporting and analysis” on page 91.
6.2.2 Manually merging source cube objects

It is possible to manually merge objects in a virtual cube that could not be merged automatically. The following sections describe how to manually define these objects.

Defining a virtual dimension

Figure 6-4 on page 136 shows source cube gosldw_sales that has a dimension named Time, and source cube gosldw_target has a dimension named Time-month-target. Because these dimensions have different names, they were not merged automatically, so two virtual dimensions Time and Time-month-target 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.

Use the following steps to merge Time and Time-month-target dimensions.

1. In the Project Explorer, select the Time-month-target dimension.
2. Right-click and select Delete.

3. Select the Time dimension. In the column for the gosldw_target cube, click the Dimension Editor button. Figure 6-6 shows its location.

![Image of Virtual Dimension Editor]

4. In the Select Source Object window, select the Time-month-target dimension.
5. Click OK.

Defining a virtual hierarchy

Use the following steps to create a virtual hierarchy:

1. To open the Hierarchy Editor, in the Virtual Dimensions column, double-click the row that has the Time dimension.

2. Because both Time and Time-month-target dimensions have a hierarchy named Time in the source cubes, a virtual hierarchy named Time was automatically created, merging both source hierarchies, as shown in Figure 6-7 on page 139. If these dimensions had hierarchies with different names, two separate hierarchies would be created. Similar to the

---

Note: You cannot reference a source object more than once in a virtual cube.

---
dimensions, you can manually merge two hierarchies that have the same structure and data.

![Virtual Hierarchy Editor](image)

**Figure 6-7  Virtual Hierarchy Editor**

**Viewing virtual levels**
When you create a virtual cube, Cognos Cube Designer adds levels from the source cubes to the virtual cube. Source cubes that contain 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.

To see the virtual levels, in Project Explorer, double-click the hierarchy for which you want to see the levels. Figure 6-8 shows how levels in the Time hierarchy were merged.

![Viewing virtual levels](image)

**Figure 6-8  Viewing virtual levels**

**Defining virtual members**
Any members that do not have matching level keys are added to the virtual cube as new virtual members.

If a virtual member is not merged correctly, or could not be automatically merged, you may manually merge two source members. You may also delete redundant virtual members. When you manually merge a member, it is removed from the list of members that are automatically merged.
For example, assume that you have a member with a business key of 1066 in one cube and a business key of 1415 in another. You want to merge them because they represent the same entity.

**Defining 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.

If a virtual measure is not merged correctly, or cannot be automatically merged, you may manually merge two source measures. You may 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.

## 6.3 Common use cases for virtual cubes

This section describes several scenarios of when virtual cubes might be useful for your application.

### 6.3.1 Time partitioning

In this scenario, all sales information was originally stored in one cube named AllSales. Fact data is added nightly, which requires rebuilding the data cache and refreshing the summary tables. For large cubes, this task requires a significant amount of time. To improve performance, you can partition the data for the AllSales cube based on time to form the basis for two cubes: HistoricSales and CurrentQuarterSales. And, you may define a virtual cube, named VirtualSales, to join the two cubes. The HistoricSales cube is used to record the historical sales information that does not change frequently. The smaller CurrentQuarterSales cube is used to record the daily sales information for the current quarter. The technique is referred to as the *delta cube*.

### 6.3.2 Currency conversion

Assuming that all sales information from the company is stored in one cube named GlobalSales, and there is a business need to convert some of the sales figures that are in US dollars into other currencies. If both the sales and exchange rate information are in the GlobalSales cube, the cube contains much redundant data and is difficult to maintain. To address this problem, you can create a cube named CurrencyExchange to store the exchange rates to keep the Global Sales cube unchanged, and you can then define a virtual cube named SalesConversion to handle the currency conversion for the sales data. The GlobalSales and CurrencyExchange cubes share some dimensions, such as Time, but they generally have different structures.

The SalesConversion virtual cube needs to define a calculated member that handles the actual currency conversion. Because the actual currency conversion might not occur daily but instead only at preset days or at intervals such as monthly or weekly, the ClosingPeriod and OpeningPeriod functions can be used in this scenario.
6.3.3 Multiple fact tables

With virtual cubes, you can build a cube that will share dimensions between cubes, and build reports that use facts from multiple cubes. One example is to have a virtual cube that includes a cube with actual values and a cube with target or budget values.

Into the virtual cube you can build calculated measures that reference measures from both source cubes. One example is a measure that calculates the variance from plan by subtracting the budget or the target plan values from the actual values. If the remainder is a positive number, then the variance exceeded the budget or target plan. Another example is a measure that takes that variance and compares it to the budget of the target plan value to produce a percent of plan measure.

6.4 Managing virtual cubes

Chapter 5, “Administering dynamic cubes” on page 115 describes tasks involved in Dynamic Cubes administration. Virtual cubes administration follows the same procedures described in that chapter. This section describes additional administration tasks required by virtual cubes.

6.4.1 Verifying virtual cube dependency

Use the following steps to verify the list of source cubes that are associated to a virtual cube:

1. Go to IBM Cognos Administration.
2. Select Status.
3. Select System.
4. Select the server.
5. Select the dispatcher.
7. Select the virtual cube.
In the metrics section, a metric named Source cubes lists the source cubes that are associated to the selected virtual cube, as shown in Figure 6-9.

Figure 6-9  Source cubes associated to virtual cube

To see the cubes that other cubes depend on for a selected source cube, select the source cube; in the metric section, a metric named Dependent cubes lists the cubes that depend on the selected cube, as shown in Figure 6-10.

Figure 6-10  Dependent cubes
6.4.2 Starting the virtual cube

Use the following steps to start a virtual cube.

1. Go to IBM Cognos Administration.
2. Select Status.
3. Select System.
4. Select the server.
5. Select the dispatcher.
7. Right-click the virtual cube you want to start, and select Start.

When you start a virtual cube, if any of the source cubes are not started, a dialog opens. It asks if you want to start the source cubes (Figure 6-11).

![Figure 6-11 Starting the virtual cube](image)

If you click OK, the source cubes that are listed in the dialog will start, then the virtual cube will start after the source cubes are running.

6.4.3 Stopping the virtual cube

You can stop the virtual cube from the IBM Cognos Administration page. Go to QueryService, right-click the virtual cube you want to stop, and select Stop.

6.4.4 Stopping the source cubes

Before you stop a source cube, you must stop the virtual cube. If the virtual cube is running and you try to stop one of the source cubes, a message is displayed (Figure 6-12).

![Figure 6-12 Stopping the source cube](image)
6.5 Virtual cube considerations

This section describes modeling, caching, and performance considerations for virtual cubes.

6.5.1 Modeling considerations for virtual cubes

Modeling considerations for virtual cubes include calculated measures and members, aggregate cubes, and support for multiple locales.

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.

Aggregate cubes
Aggregate cubes are unavailable in a virtual cube because a virtual cube can retrieve data only from source cubes. If the source cube has aggregate cubes, they will be used through virtual cube queries.

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 Portuguese are defined as supported locales. In source cube 2, English and Spanish are defined as supported locales. In the virtual cube, English, Portuguese, and Spanish 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.

Note: The All member caption, multilingual names and captions from source cubes are not automatically added to a virtual cube. You must manually define them in the virtual cube.

6.5.2 Caching and performance considerations

This section describes caching and performance considerations for virtual cubes.

Refreshing virtual cube caches
Cache updates in virtual cubes are different from base cube cache updates. You cannot directly refresh data, a member, or a security cache of a virtual cube. However, when a cache is updated on a source cube, the update is propagated up the stack of its dependent cubes. How to list dependent cubes for a source cube is described in 6.4.1, “Verifying virtual cube dependency” on page 141. The update is designed to work as an atomic transaction: all caches must update successfully, or the entire update fails and caches are rolled back to their previous versions.

In the delta cube scenario, you basically refresh the smaller delta cube's data cache and the data cache update propagates to the virtual cube. The member cache refresh for a virtual cube is almost as straightforward. Consider that if you have an explicit member merge defined, a possibility is that one or more of the source members that were being used as a source member reference for an explicitly defined virtual member might no longer exist after a member refresh. For example, maybe that row for that member was removed from the...
dimension table. If a source member cannot be found it will be ignored resulting in a virtual member with only one source member reference. Default member merging does not have this problem because source members are not explicitly defined. However, you might also end up with a virtual member that has only one source member with default merging if the other source cube does not have the corresponding source member.

**Note:** As with ordinary cubes, a virtual cube’s definition gets updated only when the cube is restarted. Refreshing the member cache updates only the set of members for the cube, but does not update the definition of the cube. For example, if you add a newly explicitly defined virtual member, it will not be picked up after a member refresh. You must restart the virtual cube first.

**Adjusting data cache size for virtual cubes**

As described in 8.8.1, “Cases for adjusting cache sizes” on page 242, if a cube is used only as a virtual cube and it is not accessible from any reporting packages, it might not be necessary to assign a data cache to the cube because data will be cached in the virtual cube that is being accessed. Only caching data for the cubes that are accessible by users is important. Note that any cube, base or virtual, that is not accessible through a reporting package, might have little need for a data cache.

If a virtual cube is built to combine history and recent data into a single cube, a step that might make sense is to assign a data cache to the base cubes, because doing so can ensure fast query performance when the recent data cube is updated.
Dimensional security

This chapter describes the concepts of Dynamic Cubes dimensional security, its hierarchies, security models, members, and filters.

This chapter contains the following sections:

- 7.1, “Overview of dimensional security” on page 148
- 7.3, “Grant versus deny approaches to security” on page 150
- 7.4, “Visible ancestors” on page 150
- 7.5, “Secured padding members” on page 159
- 7.6, “Default members” on page 162
- 7.7, “Calculated members” on page 164
- 7.8, “Merging of security views or security filters” on page 164
- 7.9, “Defining security in Cognos Cube Designer” on page 181
- 7.10, “Publish and start the cube” on page 194
- 7.11, “Applying security views” on page 195
- 7.12, “Verifying the security model” on page 197
7.1 Overview of dimensional security

Dynamic Cubes dimensional security restricts the members in a hierarchy that users are allowed to access. The hierarchy members that a user has access to is defined through dynamic query expressions.

Restricted members are either those members explicitly denied or, when there are members explicitly granted, those members that are not explicitly granted.

With Dynamic Cubes, level-based ragged and unbalanced hierarchies in dynamic cubes are converted into balanced, non-ragged hierarchies through the use of padding members. To remain consistent with the manner in which hierarchies are presented, the application of dimensional security to a level-based hierarchy retains the non-ragged, balanced nature of the hierarchy through the use of secured padding members. Secured padding members are inserted in secured hierarchies where the restriction of members would otherwise have created an unbalanced hierarchy.

When applying member security to parent-child (recursive) hierarchies, secured padding members are not needed.

In addition to secured padding members, Dynamic Cubes also supports visible ancestors, where a member is visible, but any tuple value it is a part of will have a secured value.

Dimensional security does not change the structure of the cube:
- Levels are not removed.
- Members remain on their original levels.
- Members keep their ancestry trees.

7.2 Security model

The security model consists of security filters and security views that are defined in the IBM Cognos Cube Designer.

7.2.1 Security filters

Zero or more security filters can be defined for each hierarchy within a cube. Each security filter has a name that is unique within the scope of the hierarchy in which it is defined.

Each hierarchy has a built-in security filter named Grant All Members that allows access to all members in the hierarchy.

Except for the built-in security filter, all security filters have a member expression and scope. The member expression can be any dynamic query expression that returns a member or a set of members in the given hierarchy. Use the member browser to drag members into the Cognos Cube Designer security filter expression editor.

Five scope options exist: four Grant options and one Deny option.
Grant options
Grant options have the following characteristics:

- The member expression defines the members that users, if granted access to this filter, are allowed access.
- Ancestors of granted members, if not explicitly granted, are treated as visible ancestors. All other members are implicitly denied.

Scope options for Grant:
- Grant members: Only the members that are specified in the member expression are allowed.
- Grant members and descendants: The members that are specified in the member expression and all their descendants are allowed.
- Grant members and ancestors: The members that are specified in the member expression and all their ancestors are allowed.
- Grant members, descendants, and ancestors: The members that are specified in the member expression, all their ancestors, and all their descendants are allowed.

Deny option
Deny option have the following characteristics:

- The member expression defines the members that users, if granted access to this filter, are denied access. All other members are implicitly allowed.
- To ensure hierarchies do not become ragged, denying a member also denies all its descendants.
- Scope option for Deny is deny members and descendants.

Notes:
- Most dynamic query set expressions do not include calculated members. However, the descendants and ascendants flags scope flags include calculated members.
- If errors exist in your dynamic query expression, the hierarchy will have all members denied and you will be unable to query or explore the metadata of the cube.
- Measures are not secured through security filters. Instead, they are secured on the security views.

7.2.2 Security views
A security view defines a combination of security filters from one or more hierarchies into a single view of a cube. Measures can also be secured on a security view. If a view contains two or more filters for a single hierarchy, the set of members to which a user is allowed access is the difference of the union of the allowed sets and the union of the denied sets.

Consider the following guidelines for views:

- If a user is granted access to multiple views, the consolidated view of the cube is the union of the allowed sets and the union of denied sets to which each view provides access.
- After a member is explicitly denied through a security filter, accessing it is not possible, even if the member is explicitly granted access in another filter.
- If security views are defined on the cube, any users who are not assigned to a security view have no access to the cube.
If there are no security filters for a hierarchy, all members are visible in the hierarchy. However, absence of security filters on a hierarchy differs from the built-in Grant All Members security filter.

If there are security filters, but none of the filters have a grant scope, all members that are not explicitly denied will be granted.

### 7.3 Grant versus deny approaches to security

In Dynamic Cubes dimensional security, deny has precedence over grant. After a member is explicitly denied, it cannot be accessed. Using a combination of deny filters further restricts a user’s access to the members in a hierarchy. Conversely, using a combination of grant filters builds out a user’s access to the members in a hierarchy.

Using a grant scope without the ascendants option can lead to visible ancestors.

The possibility of hitting the tuple security, described in 7.8.1, “Secured tuples” on page 164, is found only with grant security when more than one hierarchy is secured, therefore using deny security avoids the validation of that scenario.

### 7.4 Visible ancestors

If ancestors of a granted member are not explicitly granted, the ancestors are visible ancestors. Visible ancestors are visible in the metadata tree and in reports, but their value will always be secured.

Visible ancestors ensure 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 Cognos Studios cannot properly display members.

Because Dynamic Cubes does not support Visual Totals, visible ancestors ensure roll-up values do not reveal information about secured descendants.

Visible ancestors occur only when using grant filters.

#### 7.4.1 Example: Visible ancestors unnecessary

In this example, security view 1 grants access to United States, its ancestors, and its descendants. For all the granted members, their ancestor tree to the root is granted. There is no need for visible ancestors.
Security view 1: Grant United States ascendants and descendants

Figure 7-1 shows granted United States ascendants and descendants.

Figure 7-2 shows the result.

Because Dynamic Cubes does not support visual totals, in this example the values for All and Americas, while unsecured, will not represent the roll up of their visible children, as seen in Figure 7-3.

Notice the values for All and Americas do not correspond to the value in the only visible descendant, United States.

7.4.2 Example: Basic visible ancestor

In this example, security view 2 grants access to United States and its descendants. The All member and Americas are not explicitly granted, but they are treated as visible ancestors.
Security view 2: Grant United States and descendants

Figure 7-4 shows granted United States and descendants.

United States and descendants are granted, so they will have value. The All member and Americas are visible ancestors so have their values secured, as the simple report shows (Figure 7-6). The values for any tuple that contains All or Americas will be secured, as the figure shows.

7.4.3 Example: Combination of visible and granted ancestors

Security view 3 illustrates the possibility of having a combination of granted and visible ancestors in a path to the hierarchy.

Security view 3: Grant Miami and descendants, grant Americas

Figure 7-7 on page 153 shows the granted Miami and descendants and the Americas, for example:

- Grant Miami and descendants
Grant Americas

![Diagram: Granted Miami and descendants, the Americas (1)]

Figure 7-7  Granted Miami and descendants, the Americas (1)

Figure 7-8 shows the result.

![Diagram: Granted Miami and descendants, the Americas (2)]

Figure 7-8  Granted Miami and descendants, the Americas (2)

For security view 3, Miami is granted, United States is a visible ancestor, Americas is granted, and the All member is a visible ancestor.

7.4.4 Examples: View merge impact on visible ancestors

Visible ancestors are based on the final view of the cube, based on security views of a user. If the combination of security views and filters results in a granted member being denied, a visible ancestor will not be included for that now-denied member. The following examples merge views with security view 4 to show the resulting visible ancestors. For more information about the consolidate view from merging views, see 7.8, “Merging of security views or security filters” on page 164.

Security view 4: Grant Miami and descendants and grant Asia Pacific and descendants

Figure 7-9 on page 154 shows the granted Miami and descendants and granted Asia Pacific and descendants, for example:

- Grant Miami and descendants
- Grant Asia Pacific and descendants
Figure 7-9  Granted Miami and descendants and granted Asia Pacific and descendants (1)

Figure 7-10 shows the result.

Figure 7-10  Granted Miami and descendants and granted Asia Pacific and descendants (2)

Security view 4 has visible ancestors in the All member, Americas, and United States.

Security view 5: Deny Americas
Figure 7-11 shows the Americas as denied.

Figure 7-11  Denied Americas (1)
Figure 7-12 shows the result.

Security view 5 has no visible ancestors.

**Consolidated view from merging security view 4 and security view 5**
The visible ancestors, Americas and United States, that are required in view 4 are not in the consolidated view because the members were denied with view 2. The All member is still a visible ancestor in the consolidated view. Figure 7-13 shows this consolidated view.

Figure 7-13  Consolidated view of security views 4 and 5 (1)

Figure 7-14 shows the result.

Figure 7-14  Consolidated view of security views 4 and 5 (2)
Security view 6: Deny Miami and descendants

Figure 7-15 shows Miami and descendants as denied.

Figure 7-15   Denied Miami and descendants (1)

Figure 7-16 shows the result.

Figure 7-16   Denied Miami and descendants (2)

Security view 6 has no visible ancestors.
Consolidated view from merging security view 4 and security view 6
The visible ancestors, Americas and United States, that were in view 4 are not in the consolidated view because Miami is denied. Figure 7-17 shows this consolidated view.

Figure 7-17 Consolidated view of security views 4 and 6 (1)

Figure 7-18 shows the result.

Figure 7-18 Consolidated view of security views 4 and 6 (2)

Security view 7: Grant Americas and descendants
Figure 7-19 shows the granted Americas and descendants.

Figure 7-19 Granted Americas and descendants (1)
Figure 7-20 shows the result.

![Diagram 1](image1.png)

Figure 7-20  Granted Americas and descendants (2)

Security view 7 has a visible ancestor in the All member.

**Consolidated view from merging security view 4 and security view 7**

The visible ancestors, Americas and United States, that were in view 4 are now granted members in the consolidated view because of view 7. The All member is still a visible ancestor. Figure 7-21 shows this consolidated view.

![Diagram 2](image2.png)

Figure 7-21  Consolidated view of security views 4 and 7 (1)

Figure 7-22 shows the result.

![Diagram 3](image3.png)

Figure 7-22  Consolidated view of security views 4 and 7 (2)
7.5 Secured padding members

Secured padding members ensure 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 children restricted. This way is most common with grant members when granting only members and not including descendants 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 a non-leaf member has all its descendants restricted, then secured padding members will be inserted into all the levels below the non-leaf member.
- If all leaf members are restricted, padding members will be inserted; the leaf level will not be 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 should be accurate, however member properties will be null.
- There is at most one secured padding member for each level under a parent member.

7.5.1 Example: Secured padding member unnecessary

This example describes securing a member and its descendants where a secured padding member would not be needed.

**Security view: Deny Americas and descendants**

This example describes securing a member and its descendants where a secured padding member would not be needed.

Figure 7-23 shows the denied Americas and descendants.
Figure 7-24 shows the result, which denies Americas and descendants.

Securing Americas does not result in a ragged or unbalanced tree, so no secured padding member is needed.

7.5.2 Example: Secured padding member needed at leaf level

This example describes securing all children of a member at the leaf level. This results in an unbalanced hierarchy unless a secured padding member is inserted.

Security view: Deny children of United States
Figure 7-25 shows the denial of children of the United States.

Figure 7-26 shows the result.
Securing the children of the United States can result in an unbalanced hierarchy, so a secured padding member can be added to the member tree.

**Note:** This same security scenario can be created by using the Grant scope.

This security view grants the following items:
- All
- Asia Pacific, Northern Europe, Central Europe, Southern Europe and their descendants
- Americas
- Canada, Mexico, Brazil and their descendants
- United States

### 7.5.3 Example: Secured padding members needed on multiple levels

This example describes securing all children of a member, which requires a path of padding members to the leaf level.

**Security view:** Deny children of Americas and their descendants

Figure 7-27 shows the denial of children of Americas and their descendants.

![Denial of children of Americas and their descendants (1)](image)

Figure 7-27  **Denial of children of Americas and their descendants (1)**

Figure 7-28 shows the result.

![Denial of children of Americas and their descendants (2)](image)

Figure 7-28  **Denial of children of Americas and their descendants (2)**

Securing the children of Americas can result in an unbalanced tree, so secured padding members are inserted to the leaf level.
7.6 Default members

When a hierarchy is secured, a new default member on the hierarchy may be specified for the user. The common scenario for modifying the default member is if a single member and descendants is granted access. In this case, it is as though the single member is the new root of the hierarchy, even through the member might not beat the root level.

The following steps determine the correct default member for a secured hierarchy:

1. The original default member is checked to be sure it is not restricted and not a visible ancestor. If the original default member is unsecured, then it remains the default member.
2. 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 will be 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 will be 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.

The same report run by a user with all access and a user with security policies will normally hit the same cache. It varies, depending on the report structure, but in general, the secured user will need only a subset of the members that the unsecured user used, because security will limit access to the members. However, when the default member differs between the two users, the slice of the cube will differ, and a different section of the cache might be required.

7.6.1 Default member examples

Take a simple crosstab 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 will be the slicer for the report Security view 1: Grant all members
The default member for the Branches hierarchy remains All Branches.

The tuple value for security view 1 corresponds to All Time, All Products, All Branches, Quantity. See Figure 7-29.

![Figure 7-29 Tuple value for security view 1](image-url)
Security view 2: Grant United States and descendants
The default value for the Branches hierarchy is United States.

The tuple value for security view 2 corresponds to All Time, All Products, United States, Quantity. See Figure 7-30.

Security view 3: Grant Brazil and descendants
The default member for the Branches hierarchy is Brazil.

The tuple value for security view 3 corresponds to All Time, All Products, Brazil, Quantity. See Figure 7-31.

Security view 4: Grant USA, Brazil and their descendants
The default member for the Branches hierarchy is the common ancestor of United States and Brazil, the visible ancestor is Americas.

The tuple value for security view 4 corresponds to All Time, All Products, Americas, Quantity. See Figure 7-32.

Because Americas is a visible ancestor, any tuple values it participates in will be treated as error values.

7.6.2 Data caching

In the case of the unsecured user, the report in 7.6.1, “Default member examples” on page 162 will use the default member, All Branches, for the context of the Branches hierarchy. The tuple searched for in the data cache is (All Time, All Products, All Branches, Quantity).

In the case of a secured user assigned to security view 2 (United States and descendants), the report in 7.6.1, “Default member examples” on page 162 will use a different default member, United States, for the context of the Branches hierarchy. The tuple searched for in the data cache (All Time, All Products, United States, Quantity) differs from the unsecured user's tuple.
Because the tuples are not the same, reports that are run by one user would not populate the tuple value in 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 will differ.

### 7.7 Calculated members

To secure a calculated member, the calculated member must be included in the dynamic query expression because most dynamic query expressions do not support calculated members. However, calculated members are included in the various scope options.

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 will return an 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.

### 7.8 Merging of security views or security filters

If a user is granted access to multiple views, the consolidated view of the cube is the union of the allowed sets, less the union of denied sets to which each view provides access.

Merging views where measures are secured follows the same behavior as merging security filters on a hierarchy: the consolidated view of the measures is the union of the granted measures, less the union of denied measures.

A user will never have access to a member if the member is explicitly denied access to the member in one or more of the security views to which the user is assigned.

Any explicit grant (including the built-in Grant All Members filter) takes precedence over a view that has no grant filters. In two scenarios, a security rule might not have any grant filters:

- If there are only deny filters for the hierarchy
- If there are no filters defined for the hierarchy

If there are no filters defined for the hierarchy, the security view likely has filters on other hierarchies.

A user who is assigned to one view with many filters should result in the same resulting view on the cube as one user who is assigned to many views, where the views have different filters. The only difference is if tuples are not possible in an underlying view, as described in 7.8.1, “Secured tuples” on page 164.

#### 7.8.1 Secured tuples

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 may expose tuples that were
not visible in any of the underlying views. If the tuple value is not possible 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**

Security view 1 contains granted United States, Outdoor Protection, and their descendants. Figure 7-33 shows the tuple value.

![Figure 7-33 Tuple value for security view 1](image)

**Security view 2**

Security view 2 contains granted Brazil, Camping Equipment, and their descendants. Figure 7-34 shows the tuple value.

![Figure 7-34 Tuple value for security view 2](image)

**Combined security view 1 and view 2**

Because the tuples (Brazil, Outdoor Protection) and United States, Camping Equipment are not visible in either of the underlying views, the tuples are ERR in the final views. Figure 7-35 shows the tuple values.

![Figure 7-35 Tuple values for combined security view 1 and view 2](image)
7.8.2 Example 1

This example shows merging two views where the second view on its own has access to all but Boston and Calgary. However, because no grant filters are specified in security view 2, when merged with security view 1 that has grant filters, the grant filters take precedence. Also, the denied Calgary in security view 2 still applies in the consolidated view.

Security view 1
Figure 7-36 shows granting Canada, Mexico, Brazil, and their descendants.

![Diagram showing security view 1 granting Canada, Mexico, Brazil, and their descendants](image-url)
Security view 2
Figure 7-37 shows denying Boston and denying Calgary.

![Figure 7-37 Security view 2: Deny Boston, deny Calgary]

Figure 7-38 shows the consolidated view for a user who is assigned to both the views.

![Figure 7-38 Consolidated view for a user assigned to both views]

The default member is Americas, even though it is a visible ancestor, because the first level to have an unsecured member, the Country level, had more than one unsecured member. Their common ancestor, Americas, will be the default member.
7.8.3 Example 2

Similar to Example 1, the security view 2 has access to all but Boston and Calgary. However, because security view 2 was defined by using the Grant All Members filter, granting all members has the same weight as granting rules in security view 1.

Security view 1

Figure 7-39 shows granting Canada, Mexico, Brazil, and descendants.

![Figure 7-39 Security view 1: Grant Canada, Mexico, Brazil, and descendants](image-url)
Security view 2

Figure 7-40 shows granting All Members, denying Boston, and denying Calgary.
Figure 7-41 shows the consolidated view for a user who is assigned to both the views.

The default member continues to be All because it was granted in security view 2. All and Americas are not visible ancestors because they were granted in security view 2.
7.8.4 Example 3

This example shows how, when merging two views with only deny filters, the consolidated view will have access to all member but those consolidated.

Security view 1
Figure 7-42 shows denying Central Europe and descendants.

Figure 7-42 Security view 1: Deny Central Europe and descendants
Security view 2

Figure 7-43 shows denying Americas and descendants.

![Diagram: Security view 2: Deny Americas and descendants]

Figure 7-44 shows the consolidated view for a user who is assigned to both the views.

![Diagram: Consolidated view for a user assigned to both the views]

The default member continues to be All because it was granted in security view 2.
7.8.5 Example 4

This example shows how merging two views can maintain secured padding members and how default members are chosen.

Security view 1

Figure 7-45 shows granting United States and descendants.

Figure 7-45  Security view 1: Grant United States and descendants
Security view 2
Figure 7-46 shows granting Central Europe, and denying children of Central Europe and their descendants.

Security view 2 on its own has padding members under Central Europe. Figure 7-47 shows the consolidated view for a user who is assigned to both the views.

The default member continues to be All because the first level with an unsecured member (Central Europe) also has a visible ancestor (Americas). Their common ancestor would be the default member. Americas and the All member would continue to be visible ancestors. Also, notice the padding members are added under Central Europe because all the descendants were secured.
7.8.6 Example 5

This example shows how merging two views can require secured padding members.

Security view 1

Figure 7-48 shows granting Americas, and granting United States and descendants.

![Security view 1: Grant America, grant United States and descendants](image)
Security view 2

Figure 7-49 shows denying United States and descendants.

![Security view 2: Deny United States and descendants]

Figure 7-50 shows the consolidated view for a user assigned to both the views.

![Consolidated view for a user assigned to both the views]

The default member is Americas, and All member is a visible ancestor. While neither security view 1 or security view 2 had padding members, their consolidated view requires padding members to stay balanced.
7.8.7 Example 6

This example shows how merging two views can restrict an entire hierarchy.

Security view 1

Figure 7-51 shows granting United States and descendants.

Figure 7-51  Security view 1: Grant United States and descendants
Security view 2

Figure 7-52 shows denying United States and descendants.

When the two views are merged, no unsecured member remain. The members granted in view 1 are denied in view 2. There are no members for the user to see and the hierarchy is secured.
7.8.8 Example 7

This example shows the merging of a view that has filters on a hierarchy with a view that has no filters for the hierarchy. We can assume that security view 2 has filters on another hierarchy, but the Branches hierarchy is unsecured.

Security view 1

Figure 7-53 shows granting United States and descendants.

![Diagram showing Security view 1: Grant United States and descendants](image-url)
Security view 2

Figure 7-54 shows that no filters are on the hierarchy.

![Diagram of security view 2]

Figure 7-54  Security view 2: No filters on the hierarchy

Figure 7-55 shows the consolidated view for a user who is assigned to both the views.

![Diagram of consolidated view]

Figure 7-55  consolidated view for a user assigned to both the views

When the two views are merged, because there are no filters on security view 2, only the filters from security view 1 are applied. This behavior differs from view with a Grant All Members filter.
7.9 Defining security in Cognos Cube Designer

To create security views in Cognos Cube Designer, you complete the following tasks:
1. Decide which hierarchies you are interested in having secured.
2. Define the appropriate security filters.
3. Define the security views.
4. Publish the cube.

For this scenario, suppose you are interested in creating two security views which secure the Products hierarchy of the gosldw_sales.

Security view 1

Figure 7-56 shows granting access to Outdoor Protection and its descendants, and denying access to Insect Repellents and its descendants.

![Diagram of security view 1 showing access to Outdoor Protection and its descendants, and denying access to Insect Repellents and its descendants.](image)
Figure 7-57 shows the result.

Figure 7-57   Result of security view 1

Security view 2

Figure 7-58 shows granting access to First Aid and descendants, and denying access to Insect Bite Relief.

Figure 7-58   Security view 2
Figure 7-59 shows the result.

![Diagram showing results of security view 2](image)

Figure 7-59  Result of security view 2

Figure 7-60 shows the resulting behavior when a user is assigned to both security view 1 and security view 2.

![Diagram showing user assignment](image)

Figure 7-60  User is assigned to security view 1 and security view 2
Figure 7-61 shows the result.

Figure 7-61 Result of user assigned to security view 1 and security view 2

7.9.1 Selecting the hierarchy on which to define the security filter

Use the following steps to model security on the Products hierarchy.

1. Open Cognos Cube Designer with the cube model created.
2. In the Project Explorer, expand the Products dimension.
3. Double-click the Products hierarchy to open the level editor pane.
4. In the hierarchy editor pane, select the Security tab.

Figure 7-62 shows modeling security on the products hierarchy.
Note the built-in security All Members Granted line in the Security Filters list (Figure 7-63). Every hierarchy has this built-in filter that grants access to all members in the respective hierarchy.

Figure 7-63  All member granted in the security filters

7.9.2 Creating security filters

Use the steps in this section to define the security filters required to model security view 1 and security view 2.

Grant Outdoor Protection and descendants
Complete the following steps:
1. Click the Add Security Filter icon to add a new security filter (Figure 7-64).

Figure 7-64  Adding a new security filter

2. Select the New Security Filter and press F2 to rename the security filter to: Grant Outdoor Protection and Descendants. Figure 7-65 shows how to rename the Security Filter.

Figure 7-65  Renaming the security filter
3. From the Scope drop-down list (Figure 7-66), select *Grant Members and Descendants* as the appropriate scope.

![Figure 7-66 Scope drop down list](image)

4. Click **Edit** to specify the hierarchy members to secure (Figure 7-67).

   The expression editor should specify a dynamic query mode expression that returns a set of members.

![Figure 7-67 Click Edit](image)

5. Use the member browser to add members to the expression editor.

6. In the **Project Explorer**, expand the **Products** hierarchy.

7. Expand the **Members** folder.

8. Navigate to **Outdoor Protection** by expanding the **All** member.

9. Select the **Outdoor Protection** member and drag it into the expression editor.
Figure 7-68 shows dragging the Outdoor Protection.

10. Click **Validate** to ensure the expression is valid.
11. Click **OK**.

**Deny Insect Repellents and descendants**

Complete the following steps to create the second filter in security view 1:

1. Click the Add Security Filter icon to add a new security filter.
2. Select the New Security Filter and press F2 to rename the security filter to: **Deny Insect Repellents and Descendants**
3. From the Scope drop-down list, select **Deny Members and Descendants** as the appropriate scope.
4. Click **Edit** to specify the hierarchy members to secure.
5. In the Project Explorer, expand the **Products** hierarchy.
6. Expand the **Members** folder.
7. Navigate to **Insect Repellents** by expanding the **All** member, then **Outdoor Protection**.
8. Select the **Insect Repellents** member and drag it into the expression editor.
Figure 7-69 shows creating the second filter in security view 1.

![Diagram showing security filter creation](image)

9. Click **Validate** to ensure the expression is valid.
10. Click **OK**.

**Grant First Aid and Descendants**

Complete the following steps to create the first filter in security view 2:

1. Click the **Add Security Filter** icon to add a new security filter.
2. Select the New Security Filter and press F2 to rename the security filter to: **Grant First Aid and Descendants**
3. From the Scope drop-down list, select **Grant Members and Descendants** as the appropriate scope.
4. Click **Edit** to specify the hierarchy members to secure.
5. In the Project Explorer, expand the **Products** hierarchy.
6. Expand the **Members** folder.
7. Navigate to **First Aid** by expanding the **All** member, then **Outdoor Protection**.
8. Select the **First Aid** member and drag it into the expression editor.
Figure 7-70 shows creating the first filter in security view 2.

9. Click **Validate** to ensure the expression is valid.
10. Click **OK**.

**Deny Insect Bite Relief**

Complete the following steps to create the second filter in security view 2:

1. Click the **Add Security Filter** icon to add a new security filter.
2. Select the New Security Filter and press F2 to rename the security filter to: Deny Insect Bite Relief.
3. From the **Scope** drop-down list, select **Deny Members and Descendants** as the appropriate scope.
4. Click **Edit** to specify the hierarchy members to secure.
5. In the Project Explorer, expand the **Products** hierarchy.
6. Expand the **Members** folder.
7. Navigate to **Insect Bite Relief** by expanding the **All** member, then **Outdoor Protection**, then **First Aid**.
8. Select the **Insect Bite Relief** member and drag it into the expression editor.
Figure 7-71 shows creating the second filter in security view 2.

9. Click **Validate** to ensure the expression is valid.
10. Click **OK**.

### 7.9.3 Creating security views

This section describes how to create security views after creating the necessary security filters.

**Security view 1**

Complete the following steps to create security view 1:

1. In **Project Explorer**, click the **gosldw_sales** cube.
2. Select the **Security** tab.
3. Click the **Add Security Filter** icon to create a new security view. Figure 7-72 shows creating a new security view.
4. Select the New Security View and press F2 to rename the security view to Grant Outdoor Protection and Deny Insect Repellents.

5. Click the **Add Secured Data** icon to add security filters to the security view. Figure 7-73 shows adding security filters to the security view.

![Figure 7-73  Adding security filters to security view](image)

6. In the Add Security Filters dialogs, select the security filters you want to add to the security view. For security view 1, expand the Products hierarchy, select Grant Outdoor protection and Descendants and Deny Insect Repellents and Descendants (Figure 7-74).

![Figure 7-74  Selecting the interested security filters](image)

7. Click **OK**. Figure 7-75 show the selected security filters.

![Figure 7-75  Displays the selected security filters](image)

**Security view 2**

Complete the following steps to create security view 2:

1. Click the **Add Security Filter** icon to create a new security view.

2. Select the New Security View and press F2 to rename the security view to Grant First Aid and Deny Insect Bite Relief.
3. Click the **Add Secured Data** icon to add security filters to the security view.

4. Select the security filters you want to add to the security view from the Add Security Filters dialog box. For security view 2, select **Grant First Aid and Descendants** and **Deny Insect Bite Relief**.

5. Click **OK**. Figure 7-76 shows the selected security filters.

![Figure 7-76 Displays the selected security filters](image)

### 7.9.4 Securing measures

Measures can also be secured, but are modeled in a slightly different way than hierarchy members. The grant and deny rules work the same for measures as they do for hierarchy members.

Complete the following steps to secure the measures in **Grant Outdoor Protection and Deny Insect Repellents**:

1. Select the **Add Secured Measures** icon to the right of the Secured Measures editor (Figure 7-77).

![Figure 7-77 Adding the secured measures](image)

2. In the **Add Measures** dialog box, select the access to the measures you are selecting, either grant or deny. For this example, select **Grant Select Members**.

3. Select the check box on the measures you want to grant access to: Unit Cost, Unit Price, Unit sale Price.

   Figure 7-78 on page 193 shows selecting the Grant Select Members.
4. Click OK. The Secured Measures pane now shows the granted measures. If changes were warranted, the other radio button can be selected or the measure can be removed from the list. Figure 7-79 shows the granted measures.

5. Select the Add Secured Measures icon to the right of the Secured Measures editor.

6. In the Add Measures dialog box, notice that only the measures that were not previously selected are listed. Select Deny Select Members.

7. Select the check box on the measures that you want to grant access to: Gross Profit.
Figure 7-80 shows selecting the Deny Selected Measures.

![Adding Measures Screen](image)

**Figure 7-80 Selecting the Deny Selected Measures access**

8. Click **OK**.

The Secured Measures pane now shows the granted measures and the denied measures for view 1. Revenue and Quantity are not included in the secured measures list, similar to hierarchy members not being included in a grant or deny filter expression.

Figure 7-81 shows granted measures and denied measures of **Grant Outdoor Protection** and **Deny Insect Repellents**.

![Secured Measures Table](image)

**Figure 7-81 Granted Measures and Denied Measures for Grant Outdoor Protection and Deny Insect Repellents**

7.10 **Publish and start the cube**

For a new security model, or for changes to an existing security model to be applied, the cube must be redeployed to the content store and the cube must be restarted. See the following sections:

- For detailed instructions about publishing a cube from the Cognos Cube Designer, see section 4.11, “Deploying dynamic cubes for reporting and analysis” on page 91.

- For detailed instructions about starting a dynamic cube, see 5.2.1, “Starting a cube” on page 120.
7.11 Applying security views

Cognos Dimensional Security requires an active directory. Use the following steps to define an active directory:

1. Log on to IBM Cognos Administration as a user with the privileges to assign users to security views.
2. On the Configuration tab, click Data source Connections.
3. Click the gosldw_sales dynamic cube data source (Figure 7-82).
4. Click model to see the security views that are defined on the cube (Figure 7-83).
5. Users, group, and roles can be assigned to the security views. For this example, we will map both views to the same user. Click the Set Properties icon in the Actions column for Grant Outdoor Protection and Deny Insect Repellents. Figure 7-84 shows Set properties for Grant Outdoor Protection and Deny Insect Repellents.
6. Select the **Override the access permissions acquired from the parent entry** option for the ability to add or remove access permissions (Figure 7-85).

![Figure 7-85](image)

*Figure 7-85  Select Override the access permissions acquired from the parent entry*

7. Click **Add** to add user *Alan White* to Grant Outdoor Protection and Deny Insect Repellents.

8. Select **Directory Administrators** and **Everyone**, and then click **Remove** to remove them from Grant Outdoor Protection and Deny Insect Repellents (Figure 7-86).

![Figure 7-86](image)

*Figure 7-86   Removing Directory Administrators and Everyone*

9. Select *Alan White* and then select the **Read** check box (under Grant) to give Alan White access to the security view. See Figure 7-87.

![Figure 7-87](image)

*Figure 7-87   Giving Alan White access to security view*

10. Click **OK**.
11. Follow step 1 on page 195 through step 10 on page 196 for Grant First Aid and Deny Insect Bite Relief for Alan White to have Read access to both security views.

12. Start the gosldw_sales cube.

13. Select the View recent messages check box to ensure there are no errors with the security views.

### 7.12 Verifying the security model

If the package was queried by an administrator or if the cube had no security, the metadata tree show all the members and all the measures.

Figure 7-88 shows the gosldw_sales_PKG.

![Metadata tree](image)

*Figure 7-88  Metadata tree*
A basic crosstab of Unit Cost and the Products hierarchy shows all the members under Outdoor Protection. (Figure 7-89)

![Figure 7-89 Members under Outdoor Protection]

An administrator can see all the members under Outdoor Protection; it is similar to having no security defined.

If the package is queried by user Alan White, because he is assigned to Grant Outdoor Protection and Deny Insect Repellents and Grant First Aid and Deny Insect Bite Relief, he sees a restricted view of the metadata tree with only Unit Cost, Unit Price, Unit Sales Price measures visible.
Figure 7-90 shows a metadata tree with only Unit Cost, Unit Price, Unit Sales Price measures visible.

When the user queries the same report as the administrator, they see a different result because of dimensional security (Figure 7-91).

The members Camping Equipment, Mountaineering Equipment, Personal Accessories, and Golf Equipment are not visible to Alan White under the All level. Only Outdoor Protection is visible under the All member.

Also, under Outdoor Protection, Insect Repellents and its descendants, and Insect Bite Relief under First Aid, are not visible to the user.
This chapter describes the use of the Aggregate Advisor to generate aggregate recommendations to optimize query performance, and the techniques and scenarios for performance tuning.

This chapter contains the following sections:

- 8.1, “Performance implications of OLAP-style reports” on page 202
- 8.2, “Aggregate awareness in Cognos Dynamic Cubes” on page 202
- 8.3, “Overview of the Aggregate Advisor” on page 203
- 8.4, “In-memory aggregates” on page 221
- 8.5, “Database aggregates” on page 231
- 8.6, “Cache Priming Techniques” on page 240
- 8.7, “Cache size best practices” on page 242
- 8.8, “Scenarios for performance tuning” on page 242
8.1 Performance implications of OLAP-style reports

Query performance is mostly determined by how much data is fetched from the caches and the database, and how much information the report is trying to show the user. In general, when querying against large warehouse data, use context filters to constrain the data that is processed.

An expectation is that a vast majority of OLAP-style reports and analyses that run against Cognos Dynamic Cubes show data at high levels of hierarchies. For example, the report on the Gross Profit for Retailers Region and Products Product Line.

Any deeper analysis or drilling down will have filters to highly constrain the resulting data. For example, showing gross profit for a particular product in a specific retailer site.

A report is unlikely to be at the deepest level of all hierarchies. If it is at the bottom of a large dimension hierarchy, it probably is highly constrained. For example, showing data for one employee by region, or the top 10 products.

Note: Reporting and analysis users need to be aware of the performance implications if reports are not filtered or scoped appropriately. A vast majority of OLAP-style reports and analyses are filtered and scoped to show data at high levels of aggregation. Any deeper analysis should have filters to highly constrain the resulting data. For example, showing data for customers within a region, as opposed to including all leaf-level customers of an entire dimension.

8.2 Aggregate awareness in Cognos Dynamic Cubes

Use of aggregates can significantly improve performance of queries by providing data that is aggregated at levels higher than the grain of the fact.

Database aggregates are tables of precomputed data that can improve query performance by reducing the number of database rows processed for a query. Because not all relational databases have built-in aggregate support, or if they do have built-in support, do not do it well for all types of queries. Cognos Dynamic Cubes brings the aggregate routing logic up into its query engine where the multi-dimensional OLAP context is preserved and can better determine whether to route to aggregates.

Cognos Dynamic Cubes can also use aggregate tables that the database optimizer does not know about or the database optimizer might not route to because of complex OLAP-style SQL.

The aggregate tables can be regular tables that happen to hold aggregated data. The Great Outdoors warehouse has an existing database aggregate table:

AGGR_TIME_PROD_OM_FACT

See 4.9.1, “Modeling aggregate cubes” on page 85 for a description of how to include database aggregates in the model to enable the cube to be database aggregate-aware. By enabling the cube to be aware of database aggregate tables, Cognos Dynamic Cubes guarantees routing to them.

In addition to guaranteed routing to aggregates in the database, a major performance feature of Cognos Dynamic Cubes is its support of aggregates that are in memory. Not only do in-memory aggregates provide precomputed data, they can greatly improve query
performance because the aggregated data is stored in memory in the aggregate cache of the dynamic cube.

For warehouses that do not yet have aggregates, or want to supplement existing database aggregates with in-memory and other in-database aggregates, run the Aggregate Advisor as part of an optimization workflow to get recommendations for aggregates, both in-memory and in-database.

**Note:** The easiest way to get aggregates that will significantly improve query performance is to run the Aggregate Advisor. The Aggregate Advisor provides in-memory aggregates for the aggregate cache, and recommendations for database aggregates.

### 8.3 Overview of the Aggregate Advisor

The Aggregate Advisor offers aggregate recommendations that provide coverage for OLAP queries against Cognos Dynamic Cubes based on cube model analysis and, optionally, a query workload. Before running the Aggregate Advisor, the expectation is that the dynamic cube is published in the Content Store, can be started successfully, and that reports and analysis run and return correct results.

#### 8.3.1 Model-based analysis

At the core of the Aggregate Advisor logic is its theoretical, model-based evaluation. The Aggregate Advisor evaluates the cube model and experiments with potential slices of the cube, and evaluates these candidates on various heuristics. Some examples of heuristics include the balance between coverage in relation to other candidates so that aggregates can be logically derived from one another and number of manageable aggregates, and selectivity based on effective size relative to the underlying fact data.

#### 8.3.2 Workload-based analysis

The aggregate recommendations can be much more relevant to actual user queries if information from a representative query workload is captured. If workload logging is enabled for the cube, any report that is run, drill behavior, and any user action in the Cognos studios that results in a query that is processed by the Cognos Dynamic Cubes engine will be logged in a separate workload log.

Running the Aggregate Advisor to consider workload information in the workload log will significantly weigh the measures and slices that are actually hit by users so that the aggregate recommendations are likely to be more reflective of query usage of the data.

**Note:** The Aggregate Advisor can be run without having any workload information. To get better, more relevant aggregate recommendations, a representative workload should be captured and used by the Aggregate Advisor.
8.3.3 Running the Aggregate Advisor without considering workload information

The Aggregate Advisor can be run without having any workload information. This might be the case during new cube development and a representative query workload is not yet available or determined. In this scenario, the Aggregate Advisor uses only its theoretical, model-based evaluation logic to produce recommendations for the gosldw_sales dynamic cube.

Launch Dynamic Query Analyzer
Dynamic Query Analyzer (DQA) is a tool to analyze queries for optimization, and it is the client tool to run and manage Aggregate Advisor recommendations.

Set user permissions
For a secured Cognos server, the user that will be used in DQA should have write access on the Cognos Dynamic Cubes data source. By having this write access on the Cognos Dynamic Cubes data source the user can save a set of in-memory aggregate recommendations, and clear a set of in-memory aggregate recommendations from the Content Store. By default, the Cognos Dynamic Cubes data source will inherit the permissions from the Cognos namespace with only read, execute, and traverse permissions.

If the specified user does not have write permission on the Cognos Dynamic Cubes data source, the user is still able to invoke a run of the Aggregate Advisor. However, the user will not be allowed to save or clear any in-memory aggregates that are associated with the dynamic cube.

Use the following steps to set write access on the Cognos Dynamic Cubes data source.
1. Go to IBM Cognos Administration.
2. Select the Configuration tab.
3. Select Data Source Connections.
4. For the gosldw_sales Cognos Dynamic Cubes data source, select Set properties.
5. Select the Permissions tab (Figure 8-1 on page 205).
6. If the user is not already granted read, write, execute, and traverse permission, use these steps:
   a. Select the Override the access permissions acquired from the parent entry check box.
   b. Click Add to add the selected entries.
   c. Click OK.
   d. Select the newly added selected entries and grant read, write, execute, and traverse permissions.
7. Click OK.
Chapter 8. Optimization and performance tuning

Set Preferences to Cognos Server

Instances of the Aggregate Advisor run within the Cognos Dynamic Cubes engine that is a subcomponent of the QueryService. So, the first step is to configure DQA to point to the dispatcher with the QueryService where instances of the Aggregate Advisor, initiated from DQA, will be run. Use the following procedure and see Figure 8-2 to set preferences:

1. Launch DQA.
2. Select Window → Preferences.
4. Specify locations for the Dispatcher URI and the Gateway URI for the Cognos server that is configured to use Content Store with the published gosldw_sales cube.
5. For a secured Cognos server, specify the Name, Password, and Namespace to use.
6. Click OK.

Figure 8-1  Set permissions on the Cognos Dynamic Cubes data source

Figure 8-2  Set Cognos Server in Preferences
Run the Aggregate Advisor wizard

**Note:** Before running the Aggregate Advisor for the first time, configure DQA to point to the dispatcher with the DQM server where instances of the Aggregate Advisor will be run.

For a secured Cognos server, the specified user should have write access on the Cognos Dynamic Cubes data source.

To initiate a run of the Aggregate Advisor, select **File → Run Aggregate Advisor**. Then, follow the steps in the remainder of this section.

**Select the cube**
The first page of the IBM Cognos Dynamic Cubes Aggregate Advisor wizard is to select a cube on which to run the advisor. A list of dynamic cubes is retrieved from the Content Store.

Having the dynamic cube assigned to a QueryService is not required, although it is likely already is assigned and configured as a cube under the QueryService. The reason is because having a dynamic cube that was verified to start and successfully answer reports are prerequisites of running the Aggregate Advisor.

The cube does not need to be started or running at the time the Aggregate Advisor is run. The Aggregate Advisor will issue queries to the underlying relational data source during the model-based analysis to determine factors such as slice cardinalities. The Aggregate Advisor should be run during off-peak, non-critical business hours to minimize impact to users who are running reports and analysis.

The list of dynamic cubes available does not include any virtual cubes. To optimize the virtual cube using aggregates, run the Aggregate Advisor against the constituent base cubes of the virtual cube.

To run the Aggregate Advisor against the gosldw_sales cube, select **gosldw_sales** as shown in Figure 8-3, and click **Next**.

![Figure 8-3 Select the cube on which to run the Aggregate Advisor](image)

**Specify general options**
In the next panel, select general options such as whether to include workload information, how much memory to allot for in-memory and in-database aggregates, and maximum time
limit for the run. Figure 8-4 shows the Specify General Options window without using the
query workload information.

![IBM Cognos Dynamic Cubes Aggregate Advisor](image)

Figure 8-4   Specify general options without using query workload information

The following fields are shown Figure 8-4 include:

- **Query Workload Information**
  
  Because there is no workload information, the check box for this field is not selected. If the
  option is selected and there is no workload, the Aggregate Advisor will run and use only
  model-based analysis.

- **In-memory aggregates**
  
  The dialog default for the maximum amount of memory to allot for in-memory aggregates
  is 1 GB.

- **In-database aggregates**
  
  The dialog default for the maximum amount of memory for in-database aggregates is
  10 GB.

Guidelines of how to determine the maximum values for in-memory aggregates and
in-database aggregates, are in *Understanding Hardware Requirements for Cognos
Dynamic Cubes* in the business analytics proven practices website:


For the Great Outdoors warehouse, these defaults are more than sufficient.

Specifying a very large amount of memory to use for in-memory and in-database
aggregates in the wizard does not guarantee that the Aggregate Advisor recommends as
many aggregates as it can to fill the space. The Aggregate Advisor only recommends
aggregates that it confidently determines will be of the greatest benefit. Specifying
workload information is a large factor in that confidence because it is better to have
aggregates in-memory that are useful and will be used. In this scenario, that has no workload information, confidence is factored by dimensional characteristics.

- Advisor runtime limit

This option specifies how long to allow the Aggregate Advisor to run. The dialog default is one hour. For the Great Outdoors warehouse, and in most cases with smaller data sets, this is more than enough time. For larger enterprise warehouses, in the terabytes of uncompressed data, run the Aggregate Advisor for several hours or overnight. You may stop the run to get any recommendations it has so far. However, the longer the Aggregate Advisor is allowed to run, the more time it has to complete its iterations of analysis and potentially have better recommendations, especially for those refined from model-based analysis.

Finish the wizard and run the Aggregate Advisor

To complete the wizard and run the Aggregate Advisor, click Finish.

A request is sent from DQA to the dispatcher and QueryService to initiate a run of the Aggregate Advisor on the Cognos server. When the Aggregate Advisor is running, it does not need to be constantly monitored. The progress monitor dialog displayed has a “Run in background” option. When this option is selected, the progress monitor dialog is minimized and the recommendation results are returned to DQA when complete.

The progress graph shows time in minutes on the bottom, X-axis, and a unitless fuzzy value for the Y-axis. The value that is plotted is not a concrete value, such as progress complete or data coverage. It is intended to give insight into how far along and how confident the Aggregate Advisor thinks its candidates are at a given time. The values are to be taken in context, relative to its values on the same graph. A shallow, gradual slope upward indicates processing of aggregate candidates, and notable jumps up indicate a significant improvement or iteration of the aggregate candidate analysis. Examples of such analysis iterations are completion of workload analysis, determining cardinality of a candidate that is of acceptable size, or consolidation of one or more candidates. The shape of the progress graphs can vary from one cube model to another, fact data skew, and what sort of workload information is available.

The first significant jump in the graph is most likely when the Aggregate Advisor has an initial set of recommendations. At this point, the End button on the progress monitor dialog is enabled. Selecting End stops the Aggregate Advisor run, returning any recommendations it has so far. The initial sets of recommendations, however, have notably rough estimate sizes. The intent with these initial recommendations is to quickly provide a set of recommendations that are most likely to be used. These recommendations are correlated from the query workload information, if that option were selected to be considered. In this scenario, where no workload information is considered, it would be high-level slices from a region of the cube model.

The subsequent iterations of the Aggregate Advisor analysis after determining initial recommendations further refines these candidates by running queries to determine the actual cardinality of the candidates, consolidation of similar candidates, rejecting those that are too large in relation to the fact table, and supplementing with other model-based candidates.
Figure 8-5 shows an example of the progress graph while the Aggregate Advisor runs.

![Example of the progress graph while the Aggregate Advisor runs](image)

**Results populated in the Advisor Results view**

When the Aggregate Advisor completes its analysis, results are returned to DQA and displayed as an entry in the Advisor Results view. The Advisor Results view contains the results from the most recent run, and a history of previous run results.

Collapsible fields enable quick comparison of summary information. Notice that although 1 GB was specified as the maximum for in-memory aggregates, in this run of the Aggregate Advisor, it recommends 22 aggregates, totaling less than an estimated 90 MB.
Figure 8-6 shows the Example Advisor Results view with results from multiple Aggregate Advisor runs.

Figure 8-6   Example Advisor Results view with results from multiple Aggregate Advisor runs

8.3.4 Running the Aggregate Advisor with considering workload information

For better aggregate recommendations that are relevant to actual user queries, a representative workload should be captured and provided to the Aggregate Advisor run to do workload-based analysis to complement its model-based analysis.

**Enable workload logging**

Enable workload logging for gosldw_sales by setting properties of the dynamic cube:

1. Go to IBM Cognos Administration.
2. Select **Status**.
3. Select **System**.
4. Select the server.
5. Select the dispatcher.
7. In the QueryService context menu, select **Set properties**.
8. Select the **Settings** tab.
10. Workload logging is enabled on a per cube basis, so select the pencil icon next to gosldw_sales to edit the configuration properties of the cube.
11. Select the **Enable workload logging** check box in the Value column (Figure 8-7).

12. Click **OK**.

If the gosldw_sales cube is currently running, it must be restarted before any workload is captured.

![Figure 8-7  Select to enable workload logging in the cube properties](image)

**Run a representative query workload**

After the cube is started or restarted, any report run, drill behavior, or any user query action against the cube will be captured in the cube’s workload log.

**Workload log file**

The workload log file for Cognos Dynamic Cubes is separate from the Cognos Dynamic Cubes server trace log files. The workload log is stored locally on the server that processed the query in a single file under a directory with the cube’s name. Relative to the server installation directory, the workload log file is in the following location:

`logs\XQE\ROLAPCubes\gosldw_sales\workload_log.xml`

The only consumer of the workload log file is the Aggregate Advisor. Although the file can be opened in a text editor to verify that workload entries are being populated, it is not intended to be human-readable.

**Clearing the workload log**

When workload logging is enabled, user query actions are appended to the existing workload log file. If none yet exists, one is created.

Workload logs can be cleared by selecting the **Clear workload log** context menu option on the cube listed under the QueryService in IBM Cognos Administration.
Figure 8-8 shows how to manually clear a workload log for a cube.

![Figure 8-8 Manually clear workload log for a cube](image)

Clearing of workload logs can also be scheduled as a QueryService administration task:

1. Go to IBM Cognos Administration.
2. Select **Configuration**.
3. Select **Content Administration**.
4. Select the New Query service administration task action on the toolbar.
5. Select **Dynamic cube** from the context menu.
6. Specify a name for the task. Click **Next**.
7. In the Operation field, select **Clear workload log**.
8. In the Server Group field, select the server group that the QueryService is in, such as **Default Server Group**.
9. In the Dispatcher field, select the dispatcher.
10. Select the cubes for which the operation is to be performed, such as **gosldw_sales**. Click **Next**.
11. Select the action frequency and complete the wizard by clicking **Finish**.

**Managing different workloads**

When the Aggregate Advisor considers workload information, it picks up the workload log entries from its specific workload log location. If there are different workload characteristics at different times of the year, for example at month-end and quarter-end, then one approach to actively manage server performance is to capture these various workloads in separate workload log files.

Before running the Aggregate Advisor, ensure that the specific workload for month-end is in place so that the recommended aggregates are weighted toward the month-end workload. Repeat the procedure, replacing with the quarter-end workload, and run the Aggregate Advisor to obtain a different set of aggregate recommendations.

The history of these results is stored in DQA Advisor Results view, and can be used to apply different sets of aggregates when needed.

If replacing the workload log files on the DQM server cannot be done easily, then an alternative approach is to use the workload filter options in the Aggregate Advisor. Switching between workload log files can be difficult to do if the user who is running the Aggregate Advisor does not have access to the Cognos server to replace the workload log file, or if the Cognos server is already running with workload logging enabled and has a hold on the workload log file.
this approach, the workload log file will contain both the month-end and quarter-end workload, and the Filter Workload Information panel can be used to specify which queries the Aggregate Advisor should consider.

**Disable workload logging**

After a representative set of reports and user actions are complete, disable workload logging. Follow similar steps in IBM Cognos Administration to clear the Enable workload logging property for the cube. The cube must be restarted for the setting update to take effect.

**Launch DQA and run the Aggregate Advisor**

Follow similar steps to launch DQA and run the Aggregate Advisor, making sure to select the general option **Use available query workload information to determine aggregate needs**, as shown in Figure 8-9.

![Figure 8-9  Specify general options including use of available query workload information](image)

To narrow the workload information in the workload log for consideration, select **Next** to specify workload information filters. Otherwise, to run the Aggregate Advisor with all the information in the workload log, select **Finish**.

**Filter Workload Information**

With the “Use available query workload information to determine aggregate needs” option selected on the Specify General Options panel, clicking **Next** will poll the QueryService about its workload information.

If no workload information is available on the server, a message is displayed, containing instructions for how to produce workload information. The Aggregate Advisor can still be run although there is no workload information. It will provide recommendations based on only model-based analysis.
If there is workload information available on the server, the filter workload information panel options for package names, report names, user names, and time period are populated with the actual values from the workload log. If no filter options are selected when you finish the wizard, then all the information in the workload log is considered by the Aggregate Advisor. If filter options are selected and specified, the Aggregate Advisor considers only those queries satisfying these conditions from the workload log.

**Finishing the wizard and running the Aggregate Advisor**

To complete the wizard and run the Aggregate Advisor, click **Finish**.

If the option to use query workload information is selected and there is workload information available, the Aggregate Advisor will do the workload analysis first and heavily weigh the recommendations first, so any initial recommendations available are rough and mostly skewed toward the workload. Then, they are continually refined and supplemented with model-based analysis.

### 8.3.5 Opening Aggregate Advisor results and viewing details

As shown in Figure 8-6 on page 210, when the Aggregate Advisor completes its analysis, results are returned to DQA and displayed as an entry in the Advisor Results view. The Advisor Results view contains the results from the most recent run, and a history of previous run results.

**Open an advisor result to view details**

To open an Aggregate Advisor result and take action, in the Advisor Results view, click the first line of the advisor result entry. The first line of the advisor result entry consists of the cube name and the date and time the run was initiated. A detailed view of the advisor result will open with three tabs: General, In-database, and Options.

**General tab**

The General tab lists summary and details for both in-memory and in-database recommendations in terms of the aggregate name, estimated size, hierarchy levels, and measures, in a scrollable view.
Figure 8-10 shows an example of an advisor result's General tab contents.

**In-database tab**

To view the detailed output for in-database aggregate recommendations, open the In-database tab.

The In-database tab contains information that describes the logical aggregate, such as the measures and hierarchy levels, and instructions for next steps. For each logical aggregate, there is example SQL that illustrates how to aggregate the data. This information, however, is not an executable SQL script to create database aggregate tables. These recommendations are of database aggregates that a DBA can take and then choose which aggregates to actually create.
Figure 8-11 shows an example of an advisor result’s In-database tab contents.

*Figure 8-11   Example of an advisor result’s In-database tab contents*

Options tab

To view the options that are specified in the Aggregate Advisor wizard corresponding to the advisor run, select the Options tab.

Figure 8-12 shows advisor results in the Options tab contents.

*Figure 8-12   Example of advisor results, Options tab contents*
Take action to save aggregate recommendations and next steps

To take action and save aggregate recommendations, open the advisor result that contains the recommendations to apply. With advisor result detailed view open, three actions are now available under the File menu, as shown in Figure 8-13:

- Save In-Database Recommendations
- Save In-Memory Recommendations
- Clear Saved In-Memory Recommendations.

Figure 8-13 shows the File menu options for aggregate recommendations.

![File menu options for aggregate recommendations](image)

**Saving in-database recommendations to file**

To save the database aggregate recommendations shown in the in-database tab to an output file, select **File → Save In-Database Recommendations**.

The output text file contains the same information as what is available in the In-database tab. This file can be given to the DBA and modeler as the basis for creating the database aggregates and modeling their support in the cube.

This in-database recommendations file describes the logical aggregates, and example SQL that illustrates how to aggregate the data. This information, however, is not an executable SQL script to create database aggregate tables. These recommendations are of database aggregates that the DBA can take and then choose which aggregates to actually create. The DBA might even choose to modify the SQL to subset the aggregate to only a certain subset of members instead of including the entire level, such as specific years from the Time dimension. See 8.5, “Database aggregates” on page 231 for more information about the contents of this file and other considerations when working with the DBA to create database aggregates.

After the database aggregates are created by the DBA, the modeler can then take the relevant logical aggregate information shown in this tab to model aggregate cubes in IBM Cognos Cube Designer. See 4.9.1, “Modeling aggregate cubes” on page 85 for a description of how to include database aggregates in the model to enable the cube to be aware of the database aggregate.

**Important:** By considering the in-database recommendations, the DBA can choose which aggregates to create and how to create them. The DBA must communicate any differences from what is described in the in-database recommendations file to the modeler.
Saving in-memory recommendations to the cube

To apply the set of in-memory aggregate recommendations in the advisor result to the dynamic cube, select File → Save In-Memory Recommendations.

The set of in-memory aggregate recommendations are saved in the Content Store and are associated with the dynamic cube data source. Because in-memory aggregates are meant to be a turn-key aggregate management solution, all the in-memory aggregates from a specific advisor run result are stored together as a set. There is no option to select various individual in-memory aggregates from the set.

When storing the in-memory aggregate definitions, DQA determines whether the cache size limit of the runtime aggregate is sufficient to load and hold the aggregates. The runtime aggregate cache size limit is the value for the Maximum amount of memory to use for aggregate cache property in cube configuration in the QueryService. A warning message is displayed if the cube configuration property value is less than the estimated size. Select Accept to proceed with saving the aggregate definitions to the Content Store.

Figure 8-14 shows the warning message that is displayed when in-memory aggregates are saved.

![Figure 8-14 Warning message when saving in-memory aggregates](image)

After the in-memory recommendations are saved to the Content Store and the aggregate cache is enabled, the next time the cube is started, the in-memory aggregates will be loaded. See 8.4, “In-memory aggregates” on page 221 for more information about the loading and processing of the aggregate cache.

Note: To have the newly saved in-memory aggregates loaded in the cube, the cube must have the aggregate cache enabled and be started or restarted.

Opening other advisor results and saving other sets of in-memory aggregate definitions from these other advisor results will overwrite any existing definitions that are associated with the cube. Consider, for example, two advisor results in the Advisor Results view, such as one corresponding to a run with workload for month-end, and the other corresponding to a run with workload for quarter-end. Opening the result from the month-end advisor run and selecting to save in-memory recommendations will apply all in-memory aggregate recommendations that correspond to the month-end run to the cube in the Content Store. Then, opening the results from the quarter-end advisor run and selecting to save in-memory recommendations first will remove all month-end recommendations, and then will store the quarter-end recommendations in their place. Because aggregate cache definitions are retrieved and loaded at cube start, the quarter-end recommendations will be the set of in-memory recommendations that are used when the cube is restarted.
Clearing in-memory recommendations from the cube

To clear all in-memory aggregate definitions associated with the cube, open any advisor result for that cube and select File → Clear Saved In-Memory Recommendations.

An example of when to clear the in-memory aggregate recommendations that are associated with the cube is prior to creating a deployment that includes the Cognos Dynamic Cubes data source and the set of in-memory aggregates are not applicable to the target system. Deployments that contain Cognos Dynamic Cubes data sources include all associated in-memory aggregate definitions in the Content Store. If moving a deployment from one scale factor of the data warehouse to another, the same aggregate recommendation might not be applicable. Instead, rerun the Aggregate Advisor against the differently scaled data warehouse to get better suited recommendations. See 9.3, “Best practices for deploying between environments” on page 253 for more information.

Managing entries in the Advisor Results view

The Advisor Results view contains the results from multiple Aggregate Advisor runs.

Sorting entries in the Advisor Results view

The list of advisor results can be sorted by cube name or by the timestamp of the advisor run. By default, the sort order is by timestamp in ascending order.

To sort, select the view menu of the Advisor Results view, and select Sort.

Figure 8-15 shows the Advisor Results view menu options.
The sort dialog allows sorting by two criteria and two directions. Figure 8-16 shows the Sort the Advisor Results dialog.

![Sort the Advisor Results dialog](image)

**Figure 8-16  Sort the Advisor Results dialog**

**Removing entries from the Advisor Results view**

Advisor result entries can be removed from the list by clicking the X button or selecting Clear Aggregate Advisor Results from the Advisor Results view menu (Figure 8-15 on page 219).

Use the Clear Advisor Results Cache dialog to select specific entries to remove from the list (Figure 8-17).

![Clear Advisor Results Cache dialog](image)

**Figure 8-17  Clear Advisor Results Cache dialog**

### 8.3.6 Rerunning the Aggregate Advisor

*Note:* The Aggregate Advisor can be run multiple times. Use the Advisor Results view to manage various aggregate recommendations based on various run characteristics, criteria, and workloads.

Rerun the Aggregate Advisor if any of the following situations exist:

- If non-trivial changes are made to the cube model or underlying star schema, such as in the following examples:
  - Removal of a dimension, hierarchy, or level that is included in an aggregate that is in use
  - Addition of a dimension, hierarchy, or level that will be queried against and you want it to be included in aggregates
– Addition of a measure that will be queried against and you want it to be included in aggregates. Note the following information:

• Updates for a new measure to in-database aggregates can be made without rerunning the Aggregate Advisor. The DBA can modify the database aggregate creation and refresh scripts to include the measure. Then, using Cognos Cube Designer, the modeler can update the aggregate cube definition, and publish the cube.

• Updates to in-memory aggregates require rerunning the Aggregate Advisor and saving a different set of in-memory aggregates from the new advisor results.

➢ Significant data skew changes are made, such as moving from a small-scale development system to a larger QA or production system.

➢ Query performance becomes unsatisfactory.

➢ Workload characteristic changes significantly. In this case, re-capture workload logs and rerun the Aggregate Advisor to get new relevant recommendations.

8.4 In-memory aggregates

As described in 2.3, “Cognos Dynamic Cubes caching” on page 27, Cognos Dynamic Cubes supports two types of precomputed aggregate values: those stored in database tables and those stored in its in-memory aggregate cache.

This section describes how the set of in-memory aggregates that were recommended by the Aggregate Advisor and saved in the Content Store with the cube in 8.3, “Overview of the Aggregate Advisor” on page 203 are loaded into the aggregate cache and used by the query engine.

8.4.1 Applying in-memory aggregates

In “Results populated in the Advisor Results view” on page 209, the Aggregate Advisor recommended 22 in-memory aggregates with an estimated total size of less than 90 MB. After opening and reviewing the advisor result, the set of in-memory aggregate recommendations were saved to the Content Store and were associated with the gosldw_sales cube.

With the in-memory aggregate definitions saved and associated with the cube, the next time gosldw_sales is started or restarted, the in-memory aggregates are loaded if the aggregate cache is not disabled. An aggregate cache size of zero disables the aggregate cache.

8.4.2 Loading in-memory aggregates into the aggregate cache

This section describes how to enable and load the aggregate cache.

Enabling the aggregate cache

The estimated size of the in-memory aggregates, by the Aggregate Advisor, is an estimate done at the time of the advisor run. The actual amount of memory that is consumed might differ because the Aggregate Advisor is making estimates of the size, and as the data warehouse grows, the amount of memory that is needed to hold the in-memory aggregates will increase.
The size of the aggregate cache that is specified in the properties of a dynamic cube is a maximum. The aggregate cache, though, is loaded on a first-come basis: if an aggregate result will not fit into the cache, it is discarded.

Setting the value to a very large number does not increase or waste memory. Only enough memory required to hold the defined aggregates is used. For example, if it takes 90 MB to hold the aggregates for gosldw_sales, and the aggregate cache size is set to 1 GB, only 90 MB of memory is used. Over time, if the underlying fact tables grow, the aggregates are allowed to grow to the specified maximum 1 GB.

**Note:** Set the Maximum amount of memory to use for aggregate cache cube property to a value that is greater than the advisor-estimated size so that all the in-memory aggregates can be loaded. Remember that, as the data warehouse grows, the amount of memory needed to hold the in-memory aggregates also grows based on a sliding scale that is relative to the size of the member cache.

To verify that the aggregate cache is enabled and has a sufficient amount of memory to load all the in-memory aggregates, set the Maximum amount of memory to use for aggregate cache value in the cube properties:

1. Go to IBM Cognos Administration.
2. Select **Status**.
3. Select **System**.
4. Select the server.
5. Select the dispatcher.
7. In the QueryService context menu, select **Set properties**.
8. Select the **Settings** tab.
10. Each cube has its own aggregate cache, so select the pencil icon next to gosldw_sales to edit the configuration properties of the cube.
11. Enter a value for **Maximum amount of memory to use for the aggregate cache (MB)**. Specify a value greater than the advisor estimated size so that all the in-memory aggregates can be loaded. For gosldw_sales, specifying a value of 200 is more than sufficient for now. See Figure 8-18 on page 223.
12. Click **OK**.
Wait several minutes for the updated cube configuration settings to refresh and take effect in the QueryService. Then, start or restart the cube to initiate loading of the in-memory aggregates.

**Loading the aggregate cache**

The following cube administrative actions result in the loading of in-memory aggregates into the aggregate cache:

- Cube start or cube restart
- Refresh data cache
- Refresh member cache

After the cube is started and available, the in-memory aggregates are loaded. The queries to load the aggregates are run concurrently and asynchronously. As each individual aggregate finishes loading, it becomes available for use by the query engine.

**Underlying database considerations**

The DBA should be aware of the aggregate cache-load activities and consider the impact to the underlying relational database. The queries that are run to populate the aggregate cache go against the underlying relational database, and the retrieval and processing of a large number of cells can be resource-intensive for the underlying relational database.

The aggregate load automatically begins after the cube is running and available. The cube is open to user queries at this time. User queries can still be processed during the aggregate load; however, until each in-memory aggregate completes its loading, the aggregate cannot be used and query performance will not be optimal. Because the in-memory aggregates can take some time, the cube start should be initiated during, or immediately after, the underlying database ETL process to allow enough time for the load and provide optimal query performance to users.

These queries are planned by Cognos Dynamic Cubes in the same manner as data queries, so if there are any in-database aggregate tables available, the engine can take advantage of these database aggregates and can route to them. An approach to increase the speed of the
aggregate cache load is to have supporting database aggregates. These database aggregates can be created by the DBA, based on specific needs.

To help determine which supporting database aggregates can be used to increase the speed of aggregate cache load, view the information in the database recommendation output. If the Aggregate Advisor was run to include both in-memory and in-database recommendations, information in the in-database recommendation textual output indicates how many corresponding in-memory aggregate recommendations can be satisfied by the database aggregate recommendation. For example, the header section for a logical database aggregate might contain a line similar to the following line:

Number of recommended in-memory aggregates that this aggregate can cover: 5

Note: The DBA should be aware of the impact to the underlying relational database resources that the aggregate cache load activities can have. So, the DBA should be aware of the cube administration tasks that result in an aggregate cache load, such as cube start, restart, data cache refresh, and member cache refresh.

Controlling the number of aggregates loaded in parallel

By default, the number of aggregates that are loaded concurrently is determined as being twice the number of processors of the QueryService system. For example, if the QueryService system has four processors, then there will be eight threads to load the aggregates in parallel.

The number of queries posed concurrently to populate the in-memory aggregate cache can be controlled by an advanced property of Cognos Dynamic Cubes to ensure the underlying relational database is not saturated with concurrent requests computing summary values.

To reduce the number of aggregate load queries against the underlying database at one time, reduce the value for the qsMaxAggregateLoadThreads property. Note that fewer threads more require more overall time to load the aggregates.

1. Go to IBM Cognos Administration.
2. Select Status.
3. Select System.
4. Select the server.
5. Select the dispatcher.
7. In the QueryService context menu, select Set properties.
8. Select the Settings tab.
10. Select the Override the settings from the parent entry check box.
11. In an available row, enter qsMaxAggregateLoadThreads under the Parameter field, and a value under the corresponding Value field.
12. Click OK.

The setting is a global setting for the QueryService, where each cube has the specified number of threads to load aggregates in parallel.
Chapter 8. Optimization and performance tuning

Monitoring the aggregate cache load

Cube metrics available in Cognos Administration can be used to monitor and indicate when aggregates have completed loading. Select the cube and view the following metrics:

- Loaded/defined in-memory aggregates metric, which shows the number of completely loaded aggregates to the number of defined in-memory aggregates
- Time spent loading in-memory aggregates metric

Figure 8-19 shows an example of the gosldw_sales metrics, related to aggregate cache load.

![Figure 8-19 Example of the gosldw_sales metrics related to aggregate cache load](image)

Similar information is also available if you select View recent messages from the cube context menu. The message about the number of aggregates loaded is available after the entire loading process is complete. This message can be used to indicate that the aggregate loading process is completed. For in-progress load information, see the cube Metrics in Cognos Administration.

Figure 8-20 shows the View recent messages gosldw_sales list, indicating the number of successfully loaded aggregates.

![Figure 8-20 Recent messages for gosldw_sales showing number of successfully loaded aggregates](image)

Note: The speed of aggregate cache-loading can be increased by having available supporting database aggregates that can satisfy these aggregate cache-load data queries.

8.4.3 Monitoring aggregate cache hits

Cube metrics that are available in Cognos Administration can be used to monitor the aggregate cache hit rate, along with the hit rates of the result set cache, data cache, and database aggregate tables. Select the cube and see the In-memory aggregate cache hit rate value.
8.4.4 In-memory aggregate tips and troubleshooting

This section provides tips and troubleshooting information for in-memory aggregates.

System considerations
You must consider several cube and system settings when you use in-memory aggregates.

Aggregate cache size
Hardware sizing and guidelines for the amount of memory to use for a cube’s in-memory aggregate cache are in the Understanding Hardware Requirements for Cognos Dynamic Cubes document, which you can find on the business analytics proven practices website:
https://www.ibm.com/developerworks/analytics/practices.html

The size of the aggregate cache that is specified in the properties of a dynamic cube is simply a maximum. Setting the value to a very large number does not increase or waste memory. Only enough memory that is required to hold the defined aggregates is used. For example, if 90 MB can hold the aggregates for gosldw_sales, and the aggregate cache size is set to 1 GB, only 90 MB of memory is used. Over time, if the underlying fact tables grow, the aggregates are allowed to grow to the specified maximum of 1 GB.

You should not use more than 30 GB for the aggregate cache. Although the QueryService system might have enough physical memory, the number of objects that are created by the large in-memory aggregates can affect the system resources and result in overall degraded performance.

Data cache size
The data cache is still used, even with the presence of in-memory aggregates. If a query is at a higher level of aggregation than the in-memory aggregate, the values in the in-memory aggregate are used to calculate the requested value. After the calculation is done by using the in-memory aggregate values, it is then stored in the data cache for future use. Therefore, you should provide a sufficient amount of memory for the data cache, even with using the in-memory aggregate cache.

JVM considerations
If the JVM garbage collection options are adjusted, be aware that all the memory that is used by the in-memory aggregate cache will be an object with a long lifespan. This means that for
generational concurrent (gencon) garbage collection policies, the administrator should account for enough tenured space to hold the long-lived memory objects of the in-memory aggregate cache.

**Underlying database resources**

See 8.4.2, “Loading in-memory aggregates into the aggregate cache” on page 221 for underlying database considerations that the DBA should be aware of in terms of database resources when in-memory aggregates are loaded.

**Troubleshooting**

This section describes several common in-memory aggregates issues you might encounter, how to identify the cause, and how to resolve the issue.

**In-memory aggregates are not loading**

The most common reason why in-memory aggregates are not loaded is that cube configuration property for the amount of memory to use for the aggregate cache is set to zero, which disables the aggregate cache. If this is the case, a message that describes the situation will be available in the View recent messages dialog list for the cube.

Figure 8-22 shows the dialog and why in-memory aggregates were not loaded.

![Figure 8-22 View recent messages for gosldw_sales, showing why in-memory aggregates were not loaded](http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/nav/5_6)

Another common reason why the aggregates are not loading is that the cube configuration was updated or the in-memory aggregates were saved immediately before initiating a cube-start or restart. In this case, wait a few minutes for the updates to the Content Store (that is, the cube configuration and the in-memory aggregate definitions) to be refreshed and available to the QueryService before attempting to start the cube.

**Incorrect data values**

The first step in troubleshooting a potential in-memory aggregate query problem is to determine whether the problem is because of the aggregates. Disable the aggregate cache and rerun the scenario without the in-memory aggregates to see if the correct data values are returned. To disable the aggregate cache, set the Maximum amount of memory to use for aggregate cache property in the cube configuration to zero. Restart the cube for the updated setting to take effect. Although in-memory aggregates may be defined and associated with the cube, they will not be loaded.

If the values that are calculated from the in-memory aggregates are smaller than the aggregation of the actual underlying database values, it might be a sign of referential integrity problems in the data. For more information, see *Cognos Dynamic Cubes User Guide*, v10.2.0 at the following website:

http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/nav/5_6
In that guide, see the Cognos Dynamic Cubes overview topic, the section about Referential integrity in data warehouses. It describes a set of queries to run to identify when referential integrity issues occur.

**Using the DQM server log to understand in-memory aggregate processing**

This section describes advanced troubleshooting techniques by using more verbose logging levels and examining the DQM server log.

To generate some insight about what in-memory aggregates are selected and what happens when processing a query against an in-memory aggregate, enable the logging event group specifically for the aggregate cache-related activity:

1. On the QueryService system, edit the DQM logging configuration, which is located relative to the server installation directory:
   
   `configuration\xqe.diagnosticlogging.xml`

2. Edit the aggregate cache event group to log level of `info`:
   
   ```xml
   <eventGroup name="ROLAPCubes.AggregateCache" level="info"/>
   ```

3. Edit the query performance event group to log level of `info`:
   
   ```xml
   <eventGroup name="ROLAPQuery.Performance" level="info"/>
   ```

4. Save the `xqe.diagnosticlogging.xml` file.

5. Restart the QueryService for the logging level changes to take effect.

6. Start the cube.

7. After the in-memory aggregates complete loading, run the problem query.

8. By default, the DQM server log is written to a file, which is located relative to the server installation directory:
   
   `logs\XQE\xqelog-<timestamp>.xml`

   When troubleshooting, by using the DQM server log, is complete, revert the changes to the log levels in the `xqe.diagnosticlogging.xml` file and restart the QueryService for the changes to take effect.

Open the DQM server log to verify that the following steps occur when processing a query in the aggregate cache:

- Score in-memory aggregates and select one.
  
  The requested members of the query are inspected to see whether there is an in-memory aggregate that can be used to solve the query. Each aggregate is internally scored based on how close to the user-requested query it is, and the aggregate with the best score is selected. The closer to the user-requested query the aggregate is, the better the score.
  
  For example, if the query is for `[Retailers].[Americas]`, then an aggregate on `[Retailers].[Retailer Country]` will score better than an aggregate on `[Retailers].[Retailer name]`. This log entry is shown in Example 8-1.

  **Example 8-1 Log entry**

  ```xml
  <event ...><![CDATA[Aggregate [aggregate_memory_1] selected with cost 85596.0]]></event>
  ```

- Determine whether the query is a direct hit for the selected aggregate.
  
  If the levels in the query match all the levels in the aggregate, it is considered a direct hit and it is routed directly to the aggregate cache. A direct hit has a score cost of 1.0.
Fetch values from the selected aggregate for rollups if not a direct hit.

If the query is not a direct hit, then roll ups are needed and processing continues. The query is inspected to see what values are needed from the aggregate. A portion of the query is issued against the aggregate to fetch the needed values.

For example, if the query is for [Retailers].[Americas] and the aggregate on [Retailers].[Retailer Country], then the aggregate query will request values for every country in the Americas. The fetch time is also recorded because many values might be needed. This the log entry is shown in Example 8-2.

Example 8-2  Log entry

```
<event ...><![CDATA[Fetched 24,088 tuples in 549ms.]]></event>
```

Process the aggregate values in multiple threads.

After the values are fetched from the aggregate, they are inspected to determine how many values there are and how much work processing might be. This step determines whether the processing is split into multiple subtasks that will execute in parallel. The number of available threads is also accounted for, so a busy system will result in splitting the processing into fewer subtasks.

Also, a calculation modifier value accounts for how many locations each aggregate cell belongs to. Consider an example query that requests [Retailers].[Americas], [Retailers].[Americas].[United States] and [Retailers].[Americas].[United States].[Outdoor Gear Co-op]. The value for [Outdoor Gear Co-op] can be in, or contribute to, three locations: itself, as part of [United States], and also as part of [Americas]. More locations mean more work and a larger calculation modifier.

Example 8-3 shows the log entry.

Example 8-3  Log entry

```
<event ...><![CDATA[Using 4 tasks to process aggregate.  Total number of aggregate cells accessed: 24088 with a calculation modifier of 4.0]]></event>
```

Combine values as each processing thread completes.

Each subtask works on its set of values independently. Then, as these subtasks finish processing, their results are combined to form the final result set.

For example, if the query is for [Retailers].[Americas] and the aggregate on [Retailers].[Retailer name], there can be a subtask for each country, rolling up the values for all retailers from each country independently. Then, the original task only has to roll up the country values together. An entry for each subtask is shown in the log entry in Example 8-4.

Example 8-4  Log entry

```
<event ...><![CDATA[Aggregate calc thread finished.  Processed 6,022 cells. Rollup time : 162 ms into 72 cells with 0 cells in the overfetchResultSet. Result combination time 2ms, representing 1 calc threads.  Cell cache hit rate: 99% 99% 99% 99% 99% 99% 99%]]></event>
```

Query processing is complete.

After all the subtask results are combined, the final result set is returned. The log record in Example 8-5 on page 230 signifies the end of this processing and captures some metrics.
Example 8-5  Log record

```xml
<event ...><![CDATA[Finished execution of Query Strategy for report unknown
Requested 72 tuples
Found 0 tuples in data cache
Found 0 tuples in aggregate cubes
Fetched 0 tuples from database
0 of the tuples were from pushdown queries
Created 72 tuples from the aggregate cache
   Total time processing aggregates : 774 (ms) using 4 tasks
   24088 aggregate cells rolled up into 8 separate locations
```
Keeping the aggregate cache trace-level logging on for longer than what is required can negatively affect performance. Because the trace log level will generate a lot of data to the DQM server log, it should be enabled only for the problem query, and troubleshooting be done when there is little to no other user activity.

The trace log level for the aggregate cache will dump the intermediate calculations from each of the subtask threads to the log. This information can be used to determine the source of the incorrect values.

**Note:** Do not keep aggregate cache trace-level logging enabled longer than what is required for troubleshooting. The trace-level log will generate a lot of log information and may negatively affect performance.

### 8.5 Database aggregates

As described earlier in this chapter, database aggregates can either be built-in relational database system constructs, or be regular tables that hold aggregated data. Both types can be modeled into the cube, and Cognos Dynamic Cubes can guarantee routing to them.

For warehouses that do not yet have aggregates, or want to supplement existing database aggregates with in-memory and other in-database aggregates, run the Aggregate Advisor as described earlier in this chapter to get aggregate recommendations.

This section describes taking the in-database recommendations from the Aggregate Advisor, giving them to the DBA for creation, and other considerations.

#### 8.5.1 In-database aggregate recommendations

In 8.3, “Overview of the Aggregate Advisor” on page 203, the Aggregate Advisor recommended three in-database aggregates for the gosldw_sales cube. After opening and reviewing the advisor result, the in-database output text file was saved.

The in-database recommendations output file describes each aggregate in terms of what it logically contains, and other relevant information in a header section. This file can be given to the DBA and modeler as the basis for creating the database aggregates and modeling their support in the cube. The remainder of this section describes and has an example of the header section that is available for each aggregate.

**Aggregate name**

Each aggregate is given a name based on the levels included:

* Aggregate: Branch - Order method - Promotion

**List of dimensions, hierarchies, and level of aggregation**

Example 8-6 shows the dimension hierarchies with the level at which they are aggregated.

<table>
<thead>
<tr>
<th>* Dimension</th>
<th>Hierarchy</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Branch</td>
<td>Branch</td>
<td>Branch</td>
</tr>
<tr>
<td>* Employee by region</td>
<td>Employee by region</td>
<td>[All]</td>
</tr>
<tr>
<td>* Order method</td>
<td>Order method</td>
<td>Order method</td>
</tr>
</tbody>
</table>
List of measures
Example 8-7 shows the list of measures.

Example 8-7   Measures
* Measures:
* -------------------
* Quantity
* Revenue
* Gross profit

Description of the columns that must be created for the aggregate table
Example 8-8 shows the columns and data types for the aggregate table.

Example 8-8   Description of columns
* Column              Data Type
* -------------------     ---------------
* Region_code         INTEGER
* Country_code        INTEGER
* Branch_key          INTEGER
* Order_method_key    INTEGER
* Campaign_code       INTEGER
* Promotion_key       INTEGER
* Quantity            BIGINT
* Revenue             DECIMAL(38,2)
* Gross_profit        DECIMAL(38,2)

If the data type of the column is unknown, refer to the definition of the column in the example of the SQL (in “Example of the SQL that illustrates how to aggregate the data” on page 233) and the documentation for the database to determine the precise data type.

List of other database aggregates from which the aggregate can be derived
This information can be used by the DBA to create stacked aggregates, or aggregates that can be derived from another recommended database aggregate. Although the example SQL given is still based from the underlying warehouse tables, if the DBA chooses to derive its values from another aggregate table, the DBA has the flexibility to modify the SQL, as follows:

* This aggregate can be derived from any of the following other aggregates for the database:
  * Branch - Order method - Product name - Product - Promotion
  * Branch - Order method - Product name - Product - Promotion - Retailer site
Number of recommended in-memory aggregates from the same, corresponding advisor result that the aggregate can cover

This information can be used to identify database aggregates that can accelerate the loading of in-memory aggregates, because they can satisfy this number of in-memory aggregates, if the in-memory aggregates from the same advisor result are used.

Database aggregates that cover zero in-memory aggregates can still provide performance benefits because queries against aggregate tables are faster than those against the underlying warehouse tables:

* Number of recommended in-memory aggregates that this aggregate can cover: 0

Example of the SQL that illustrates how to aggregate the data

Example 8-9 is not an executable SQL script to create database aggregate tables. It is an example of SQL that can aggregate the data and be used as guidance. The DBA can take this information and choose which aggregates to actually create. The DBA might even decide to modify the SQL to create a subset of the aggregate to contain only a certain subset of members instead of including the entire level, such as specific years from the Time dimension.

Example 8-9 SQL that illustrates how to aggregate the data

```sql
SELECT
  "GO_REGION_DIM"."REGION_CODE" AS "Region_code",
  "GO_BRANCH_DIM"."COUNTRY_CODE" AS "Country_code",
  "GO_BRANCH_DIM"."BRANCH_KEY" AS "Branch_key",
  "SLS_ORDER_METHOD_DIM"."ORDER_METHOD_KEY" AS "Order_method_key",
  "MRK_CAMPAIN_LOOKUP"."CAMPAIGN_CODE" AS "Campaign_code",
  SUM("SLS_SALES_FACT"."QUANTITY") AS "Quantity",
  SUM("SLS_SALES_FACT"."SALE_TOTAL") AS "Revenue",
  SUM("SLS_SALES_FACT"."GROSS_PROFIT") AS "Gross_profit"
FROM
  "GOSALESDW"."EMP_EMPLOYEE_DIM" "EMP_EMPLOYEE_DIM"
  INNER JOIN "GOSALESDW"."GO_BRANCH_DIM" "GO_BRANCH_DIM"
  ON "EMP_EMPLOYEE_DIM"."BRANCH_CODE" = "GO_BRANCH_DIM"."BRANCH_CODE"
  INNER JOIN "GOSALESDW"."SLS_SALES_FACT" "SLS_SALES_FACT"
  ON "EMP_EMPLOYEE_DIM"."EMPLOYEE_KEY" = "SLS_SALES_FACT"."EMPLOYEE_KEY"
  INNER JOIN "GOSALESDW"."GO_REGION_DIM" "GO_REGION_DIM"
  ON "GO_REGION_DIM"."COUNTRY_CODE" = "GO_BRANCH_DIM"."COUNTRY_CODE"
  INNER JOIN "GOSALESDW"."MRK_PROMOTION_DIM" "MRK_PROMOTION_DIM"
  ON "MRK_CAMPAIN_LOOKUP"."CAMPAIGN_CODE" = "MRK_CAMPAIN_LOOKUP"."CAMPAIGN_CODE"
  INNER JOIN "GOSALESDW"."MRK_CAMPAIN_LOOKUP"
  ON "MRK_CAMPAIN_LOOKUP"."CAMPAIGN_CODE" = "MRK_CAMPAIN_LOOKUP"."CAMPAIGN_CODE"
GROUP BY
  "GO_REGION_DIM"."REGION_CODE",
  "GO_BRANCH_DIM"."COUNTRY_CODE",
  "GO_BRANCH_DIM"."BRANCH_KEY",
```
8.5.2 Creating recommended database aggregates

The DBA has the flexibility to choose which in-database aggregate recommendations to use and how to implement them. The recommendations are intended to be easy to use and relevant guidance based on the Aggregate Advisor workload and cube model analysis. Any changes that the DBA makes to the actual database aggregates that are created should be communicated to the modeler so that the database aggregates are properly modeled into the cube. See 4.9.1, “Modeling aggregate cubes” on page 85. That section describes how to use Cognos Cube Designer to include database aggregates in the model, to enable the cube so it is aware of the database aggregate.

Many approaches to create aggregate tables exist. Each differs across various relational database vendors. Example 8-10 shows how, in a simple manner, the information and example SQL from the recommendation file can be used to create an aggregate table and initially populate a database aggregate table. This example does not describe any of the relational database considerations that are involved in creating and inserting into aggregate tables, which a DBA will likely do, such as tablespaces, database transaction log management, isolation levels, indexes, use of built-in database optimizer, or aggregate support.

Example 8-10 Aggregate table

```sql
CREATE TABLE "GOSALESDW"."AGGR_BRANCH_ORDERMETHOD_PROMO" (  
    Region_code INTEGER,  
    Country_code INTEGER,  
    Branch_key INTEGER,  
    Order_method_key INTEGER,  
    Campaign_code INTEGER,  
    Promotion_key INTEGER,  
    Quantity BIGINT,  
    Revenue DECIMAL(38,2),  
    Gross_profit DECIMAL(38,2)  
));
INSERT INTO "GOSALESDW"."AGGR_BRANCH_ORDERMETHOD_PROMO"  
SELECT  
    "GO_REGION_DIM"."REGION_CODE" AS "Region_code",  
    "GO_BRANCH_DIM"."COUNTRY_CODE" AS "Country_code",  
    "GO_BRANCH_DIM"."BRANCH_KEY" AS "Branch_key",  
    "SLS_ORDER_METHOD_DIM"."ORDER_METHOD_KEY" AS  
        "Order_method_key",  
    "MRK_CAMPAIGN_LOOKUP"."CAMPAIGN_CODE" AS "Campaign_code",  
    "MRK_PROMOTION_DIM"."PROMOTION_KEY" AS "Promotion_key",  
    SUM("SLS_SALES_FACT"."QUANTITY") AS "Quantity",  
    SUM("SLS_SALES_FACT"."SALE_TOTAL") AS "Revenue",  
    SUM("SLS_SALES_FACT"."GROSS_PROFIT") AS "Gross_profit"  
FROM  
    "GOSALESDW"."EMP_EMPLOYEE_DIM" "EMP_EMPLOYEE_DIM"  
    INNER JOIN "GOSALESDW"."GO_BRANCH_DIM" "GO_BRANCH_DIM"  
    ON "EMP_EMPLOYEE_DIM"."BRANCH_CODE" = "GO_BRANCH_DIM"."BRANCH_CODE"  
    INNER JOIN "GOSALESDW"."SLS_SALES_FACT" "SLS_SALES_FACT"  
    ON "EMP_EMPLOYEE_DIM"."EMPLOYEE_KEY" = "SLS_SALES_FACT"."EMPLOYEE_KEY"
8.5.3 Maintaining database aggregates

Just as a DBA has the flexibility to choose which database aggregates to create and how to create them, the DBA is responsible for the maintenance of these database aggregate tables. Database aggregates are usually built during a predefined ETL processing window when the underlying warehouse data is also updated. The DBA must consider the time and processing order that are related to the maintenance of any database aggregates that can be derived from one another. The DBA must also consider when the cube should be started, because in-memory aggregates are loaded at that time and can make use of the data in the database.

8.5.4 Monitoring database aggregate table hits

Cube metrics that are available in Cognos Administration can be used to monitor the database aggregate table hit rate, along with the hit rates of the result set cache, data cache, and aggregate cache. Select the cube and see the Aggregate table hit rate value.
Figure 8-23 is an example of the gosldw_sales metrics showing database aggregate table hit rate.

Figure 8-23 Example of the gosldw_sales metrics showing database aggregate table hit rate

8.5.5 Database aggregates tips and troubleshooting

This section provides troubleshooting information for database aggregates.

Suggested practices for database aggregates and Cognos Dynamic Cubes

The following suggested practices are related to database aggregates and Cognos Dynamic Cubes:

- Use the Aggregate Advisor to get recommendations for aggregates.
- Consider stacked aggregates, that is, having aggregates that are derived from one another. This practice can contribute to the following benefits:
  - Higher level aggregates to be derivable from lower aggregates.
  - Improved maintenance time.
  - Deep aggregates to cover more queries, and upper aggregates to give better performance.
- In Cognos Cube Designer, model the measures as additive when possible.
  If the measure is defined by an expression and the expression can be considered as additive, set the regular aggregate type to SUM. This practice will enable the aggregate routing logic to use the aggregate cube and will enable the Aggregate Advisor to consider including it in its recommendations.
- For measures, use SUM and COUNT aggregates rather than AVERAGE, where possible.
- De-normalize the aggregate table to eliminate the need to join tables.
Suggested practices for modeling aggregate cubes in Cognos Cube Designer

When modeling aggregate cubes in Cognos Cube Designer, model smaller aggregates at higher levels first, so that the aggregate cube is assigned a lower ordinal value. Alternatively, the ordinal property can be updated after the aggregate cubes are created. Aggregate cubes with lower ordinal values will be considered before those with higher ordinal values in the Cognos Dynamic Cubes routing logic.

Aggregate cubes with the following types of traits should be assigned lower ordinal values:

- Highly aggregated tables, that is, those with a small number of rows
- Frequently used aggregate tables

Aggregate cubes with the following types of traits should be assigned higher ordinal values:

- Aggregate tables at lower levels of aggregation, that is, those with a larger number of rows
- Infrequently used aggregate tables

Troubleshooting Cognos Dynamic Cubes database aggregate routing

This section provides troubleshooting information for database aggregate routing.

Disable external database aggregate routing

The first step in troubleshooting a potential in-database aggregate query problem is to determine whether the problems is the result of the database aggregates. Disable the external aggregate support and rerun the scenario without the database aggregate awareness to see if the correct data values are returned:

1. Go to IBM Cognos Administration.
2. Select Status.
3. Select System.
4. Select the server.
5. Select the dispatcher.
7. In the QueryService context menu, select Set properties.
8. Select the Settings tab.
10. Each cube has its own aggregate cache, so select the pencil icon next to gosldw_sales to edit the cube’s configuration properties.
11. Select the Disable external aggregates property.
12. Click OK.

Restart the cube for the updated setting to take effect. Although aggregate cubes may be defined in the cube model, selecting to disable this setting will not consider any of them for routing.

Using the DQM server log to understand database aggregate routing

This section describes advanced troubleshooting techniques by using more verbose logging levels and examining the DQM server log.
To generate some insight about which aggregate cubes are considered and why they are selected, enable the logging event group specifically for the database aggregate routing-related activity:

1. On the QueryService system, edit the DQM logging configuration, which is located relative to the server installation directory:
   `configuration\xqe.diagnostics\logging.xml`

2. Edit the aggregate cache event group to log level of trace:
   ```xml
   <eventGroup name="RolAPQuery.AggregateStrategy" level="trace"/>
   ```

3. Save the `xqe.diagnostics\logging.xml` file.

4. Restart the QueryService for the logging level changes to take effect.

5. Start the cube.

6. Run the problem query.

7. By default, the DQM server log is written to a file, which is located relative to the server installation directory:
   `logs\XQE\xqelog-<timestamp>.xml`

8. When troubleshooting by using the DQM server log is complete, revert the changes to the log level in the `xqe.diagnostics\logging.xml` file, and then restart the QueryService for the changes to take effect.

As queries are processed in Cognos Dynamic Cubes, the analysis of which aggregate cubes are considered, which are rejected, and which one is selected, gets written to the DQM server log. A single user query may be decomposed into smaller queries so that some values that can benefit from database aggregates are routed to an aggregate table and other values are routed to the underlying warehouse tables.

The analyses for each of the smaller queries are found in log entries for the database aggregate routing event group denoted by the `<aggregateAnalysis>` element. The aggregate analysis element consists of the following sections:

- Original query: Lists the measures, levels, and each level’s dimension and hierarchy from the input query for consideration.
- Aggregates considered: Lists all of the matching aggregates that are qualified for routing the input query for consideration. There might be more than one matching aggregate cube that is qualified and considered as a match.
- Aggregates selected: Describes the final aggregate that are selected from one or more qualified aggregates for consideration. It also describes the reason for choosing this aggregate.
- Aggregates not matching: All the aggregates that do not qualify for routing are listed here. The reason for not qualifying is also described for each non-matching aggregate. Typically, reasons are as follows:
  - Measure mismatch: The input query measure aggregation is not listed as part of the aggregate cube, and therefore cannot be used.
  - Level mismatch: The levels at which the aggregate cube is defined do not match with the level of aggregate of the input query.
  - Aggregate slice mismatch: The slice coverage of the aggregate cube is not enough to satisfy the input query even though the measure and levels are matching. For example, if the aggregate cube is defined for slice of Time Year 2012, and the input query is
asking for measures with Time Year 2011 and 2012, the aggregate cube with only 2012 slice cannot be used to satisfy the input query.

- Aggregate cannot be rolled up for non-additive measure: Aggregates with additive measures, such as SUM and COUNT, can be used to satisfy queries of the higher level. For example, an aggregate with a SUM measure and Time levels to Quarter can be used to satisfy a query at the Year level because the values from the Quarter level can be added, or rolled up, to get the Year value with this aggregate. For non-additive measures, such as AVG and STDDEV, the roll up cannot be done for higher level queries. In this case, the aggregate is not selected with this reason.

Consider the cross-tab report (in Figure 8-24) against the gosldw_sales cube for Time Year by the measures, Gross profit, Quantity, Revenue, Unit cost, Unit price, and Unit sale price. There is an aggregate cube, gosldw_sales2, that maps to the AGGR_TIME_PROD_OM_FACT database aggregate table, which aggregates measures Quantity and Revenue at Time Quarter and Products Product type levels. In this example scenario, there are no in-memory aggregates enabled, only the database aggregates.

![Figure 8-24  Cross tab report of Time Year by gosldw_sales measures](image)

The log entries in Example 8-11 show cross-tab report user query, decomposed into two smaller queries and their corresponding aggregate analysis. One query does route to the gosldw_sales2 aggregate because the measures and levels match.

**Example 8-11  Cross-tab report user query (1)**

```
<event component="XQE" group="ROLAPQuery.AggregateStrategy">
<![CDATA[<aggregateAnalysis>
<v5Report name="unknown"/>
<originalQuery>
<measures>
<measure name="Quantity"/>
<measure name="Revenue"/>
</measures>
<levels>
<level name="Year" dimension="Time" hierarchy="Time"/>
</levels>
</originalQuery>
<aggregatesConsidered>
<aggregateCube name="gosldw_sales2" ordinal="1"/>
</aggregatesConsidered>
<aggregateSelected>
<aggregateCube name="gosldw_sales2" reasonForChoosing="Only matching aggregate cube found." SQLExecTimeInMS="-1"/>
</aggregateSelected>
<aggregatesNotMatching/>
</aggregateAnalysis>]]>
```

Chapter 8. Optimization and performance tuning  239
The other query does not route because the measures do not match, as shown in Example 8-12.

Example 8-12  Cross-tab report user query (2)

```
<event component="XQE" group="ROLAPQuery.AggregateStrategy"/>
<![CDATA[
<v5Report name="unknown"/>
<originalQuery>
<measures>
<measure name="Unit sale price"/>
<measure name="Unit price"/>
<measure name="Unit cost"/>
<measure name="Gross profit"/>
</measures>
<levels>
<level name="Year" dimension="Time" hierarchy="Time"/>
</levels>
</originalQuery>
<aggregatesConsidered/>
<aggregatesNotMatching>
<aggregateCube name="gosldw_sales2">
<ordinal value="1"/>
<reason value="measureNotMatching">
<measure name="Unit sale price"/>
</reason>
</aggregateCube>
</aggregatesNotMatching>
</aggregateAnalysis>
]]>\n```

In this example, because one of the two decomposed queries was routed to an aggregate cube, the aggregate table hit rate is 50%.

### 8.6 Cache Priming Techniques

This section describes two approaches for ensuring that relevant data is in the Cognos Dynamic Cubes caches to accelerate query performance: **preloading** and **priming**.

#### 8.6.1 Preloading the caches

Preloaded caches are those that are loaded with data that is based on metadata definitions that are associated with the cube model data source, and are loaded at cube start, cube restart, and cache refreshes. In Cognos Dynamic Cubes, the member cache and the aggregate cache are both considered preloaded caches. After they are built and loaded from definition, the member cache and the aggregate cache are neither affected by nor grow based on query usage. Only changes to the cube model or changes to the set of in-memory aggregates that are saved for the cube will affect these caches at the next cube start.

The member cache is always loaded. The size of the member cache is a key factor in determining overall JVM heap size and needs for the cube. For more information about JVM
heap size and cache size considerations, see Understanding Hardware Requirements for Cognos Dynamic Cubes at the business analytics proven practices website:

https://www.ibm.com/developerworks/analytics/practices.html

To use the aggregate cache, run the Aggregate Advisor to get and save recommendations for in-memory aggregates, and enable the aggregate cache by specifying enough memory for the corresponding cube setting, as described in 8.4.2, “Loading in-memory aggregates into the aggregate cache” on page 221. At the core of the Aggregate Advisor logic is a model-based evaluation that uses heuristics to analyze potential slices of the cube to recommend aggregates that balances coverage with effective size. Running the Aggregate Advisor to consider workload information in the workload log will significantly weigh the measures and slices that are actually hit by users so that the aggregate recommendations are likely to be more reflective of query usage of the data.

As described in 8.4.2, “Loading in-memory aggregates into the aggregate cache” on page 221, in-memory aggregates can take some time to load, so make sure that the cube start is initiated during, or immediately after, the underlying database ETL process to allow enough time for the load and to provide optimal query performance to users.

### 8.6.2 Priming the caches

Cognos Dynamic Cubes cache-priming involves running a set of reports to populate the result set cache, expression cache, and data cache to accelerate query performance for users. This technique can be used to optimize specific reports, especially if they compute lots of values or it is more efficient to cache certain information in a targeted way rather than relying on large aggregates.

The reports that are used for cache-priming can be those from a known set of dashboards and reports that a majority of users will use, or those that process large volumes of data to cache upfront for reuse by multiple users. For example, if there are a set of dashboards or reports that most users use as a starting point, these are good candidates for priming so that all users can benefit from the quick performance that cached values can provide.

The result set cache, expression cache, and data cache are populated in response to incoming queries, so cube administrative commands such as Refresh Data Cache and Refresh Member Cache will clear these caches, and incoming queries will miss the cache until they are repopulated by queries. These caches are gradually populated as user queries are run and can be used after the values are stored. However, query performance is not as optimal as compared to when the caches have the relevant data available. The reason is because there might be longer response times as a result of possible data queries to the underlying data source or the processing of large volumes of data.

After a cache refresh, cache repopulation can occur naturally in response to user queries, or proactively by scheduling priming reports to run immediately after cache refresh. Note that refreshing the data cache also refreshes the aggregate cache. The aggregate cache initiates its preloading automatically. Depending on the nature of the priming reports, running these priming reports after the aggregate cache completes loading might be better so that these reports can also use the aggregate cache and not compete for the same underlying resources.
8.7 Cache size best practices

For more information about suggested best practices when estimating cache sizes for a cube, see Understanding Hardware Requirements for Cognos Dynamic Cubes at the business analytics proven practices website:

https://www.ibm.com/developerworks/analytics/practices.html

The section about detailed sizing recommendations describes how to compute the minimum size requirements for each of the caches and describes other considerations in determining sizes for the caches. In general, having more CPU cores and allotting more memory for the data cache can improve performance of Cognos Dynamic Cubes.

8.8 Scenarios for performance tuning

For certain scenarios, performance tuning can be used in caches.

8.8.1 Cases for adjusting cache sizes

The amount of memory per cube is the total of that required for the following items:
- Member cache
- Data cache
- Aggregate cache
- Temporary query execution space
- Additional space to reduce occurrence and cost of JVM garbage collection

This section describes cases in which the amount of memory that is associated with these elements necessitates adjustment of the cache sizes of the cube and possibly the overall QueryService JVM heap size.

Actual number of members is significantly larger than the estimated number of members

For information about estimating cache sizes for a cube, see Understanding Hardware Requirements for Cognos Dynamic Cubes at the business analytics proven practices website:

https://www.ibm.com/developerworks/analytics/practices.html

One approach to estimating cache sizes is to compute based on the size of the two largest dimensions of the cube.

Typically, the two largest dimensions of a cube are larger than all other dimensions by orders of magnitude. If this case does not apply to your situation, use the actual number of members across all the large dimensions, or even all the dimensions, of the cube to calculate the size of the member cache. Then, use the new member cache value to recompute the other cache values and overall memory for the cube.

Use these same steps to recompute the member cache and other cache sizes if the dimensional space changes or grows significantly.
Number of users increases
The size of the data and aggregate caches is related to the dimensional space queried and user behavior. If the number of users increases, accounting for users who are accessing data that is unique to their session might be necessary. There might be more of the dimensional space queried or the nature of the reports might require processing of different amounts of data. Recompute the data cache size based on the updated number of named users for the user factor.

Nature of reports and usage changes
In a majority of cases, the size of the data cache and aggregate cache can be determined relative to the size of the member cache. The data cache and aggregate cache are dependent on the number of rows in the fact table when, relative to the dimensional space of a cube, there are many rows. Many rows means a dense fact table that requires more space to store non-null values.

The data cache retains data until it is nearly full and then discards data that is least accessed, because the data that users are interested in, or most likely to query, can change over time. If the nature of reports changes in such a way that data that would have been cached was discarded to make room for new data cache values, consider increasing the data cache to accommodate and retain more values.

Presence of the aggregate cache
The data cache size can be reduced by the presence of an aggregate cache. The reason is because an aggregate cache may retain many values, which would have been stored in the data cache. If the aggregate cache contains aggregates that are recommended from analyzing query workload information, the aggregate cache should have more data that is relevant to the user. Consider re-computing the data cache for memory efficiency, accounting for the presence of the aggregate cache.

Virtual cubes are directly accessed but base cubes are not
If a cube exists solely for use within a virtual cube, and it is not accessible from any reporting packages, assigning a data cache to the cube might not be necessary, because data will be cached in the virtual cube that is being accessed. It is only important to cache data for the cubes that are accessible by users. Note that any cube, base or virtual, that is not accessible through a reporting package, might have little need for a data cache.

If a virtual cube is built to combine history and recent data into a single cube, assigning a data cache to the base cubes might make sense, because this way will ensure fast query performance when the recent data cube is updated.

Virtual cubes do not have aggregate caches. Consider retaining the aggregate cache for the base cubes so that it can satisfy queries that are ultimately issued to the underlying base cube.

8.8.2 Cases for aggregates
This section describes how aggregates can enhance data warehouse and query performance.

OLAP-style reports that show data aggregated at levels higher than the grain of the fact
The expectation is that a vast majority of OLAP-style reports and analyses that run against Cognos Dynamic Cubes show data at high levels of hierarchies, for example, report on the
Gross Profit for Retailers Region and Products Product Line. Use of aggregates can significantly improve performance of queries by providing data aggregated at levels higher than the grain of the fact.

By using both in-memory aggregates and database aggregates, data warehouses of all sizes, but especially medium, large, and extra large warehouses, can benefit from the greatly decreased query times.

The expectation is that most data warehouses already have database aggregates in some form: either pre-existing aggregates for use by other applications, or aggregates added by the DBA to accelerate certain queries. The process to identify relevant and useful database aggregates can be difficult and time-consuming. For warehouses that do not yet have aggregates, or want to supplement existing database aggregates with in-memory and other in-database aggregates, run the Aggregate Advisor to get recommendations. The DBA has the flexibility to choose which of the recommended database aggregates to implement, but can rely on output from the Aggregate Advisor that these aggregates are relevant. Also, after the modeler incorporates the aggregate cube information into the model, Cognos Dynamic Cubes can guarantee routing to these database aggregates.

The aggregate cache and the in-memory aggregates that it holds should be enabled, when possible. In-memory aggregates are meant to be a turn-key aggregate management solution. Recommendations that are provided by the Aggregate Advisor can be easily saved and loaded the next time the cube starts.

Small data warehouses might be able to provide improved query performance with only in-memory aggregates cache if the queries to the underlying database have sufficient performance.

**Improving overall query performance**

The intent is to improve the overall query performance across queries in a normal workload against Cognos Dynamic Cubes. Use of aggregates can improve performance for a large majority of user queries. However, not all queries need to be covered by aggregates. There might be some outlier queries that do not hit the aggregate cache or route to a database aggregate, because either there is no aggregate that can satisfy this query or it is not run often to justify having an aggregate to cover it.

In addition to the aggregate cache and database aggregate routing, the result set cache, expression cache, data cache, query pushdown, and other features of Cognos Dynamic Cubes also help to provide the fast query response times users expect.

For new cube deployments, where a known workload of queries might not be known, run the Aggregate Advisor by using only its cube model-based analysis logic to get a set of recommendations to start with and apply to the cube data source. Then, later, after a known or a sample workload is determined and captured, run the Aggregate Advisor again. This time, set it to consider workload information to get a new set of recommendations to use that are more relevant to actual user queries.

Over time, if overall query performance becomes less than satisfactory, it might be because the workload characteristics have since changed. Recapture the workload information and rerun the Aggregate Advisor. See the 8.3.6, “Rerunning the Aggregate Advisor” on page 220 for more scenarios about when to rerun the Aggregate Advisor.
Dynamic cube lifecycle

This chapter describes the five main stages in the dynamic cube lifecycle and how they fit within a three-tiered approach of moving from development to test to production. This chapter also offers best practice approaches for moving a dynamic cube onto separate systems.

This chapter contains the following sections:

- 9.1, “Overview of the dynamic cube lifecycle” on page 246
- 9.2, “Lifecycle across different environments” on page 247
- 9.3, “Best practices for deploying between environments” on page 253
9.1 Overview of the dynamic cube lifecycle

The main objective of customers that use Cognos Dynamics Cubes is to have cubes and reports available to their users for actual business usage. A substantial amount of work must be done for this availability to occur. Figure 9-1 illustrates, at a high-level, the basic Cognos Dynamic Cubes workflow and the primary tools that are used to perform the main activities in the cycle.

This section describes each of the workflow activities.

9.1.1 Analyze

Although the initial stage of the lifecycle is not part of the ongoing cycle, it is still an important part of the process. It is where the architects look at their data, hardware, business requirements, and so on, and prepare for implementing Cognos Dynamic Cubes. In this stage, architects do the following tasks:

- Determine business requirements
- Design the database physical model
- Acquire the appropriate hardware
- Install and configure Cognos Dynamic Cubes

9.1.2 Model

The objective of this stage is to design and create a functional dynamic cube by using IBM Cognos Cube Designer. A system analyst determines the high-level business requirements; a modeler creates a dynamic cube to satisfy those requirements. In this stage, a modeler can do the following tasks:

- Create measures, dimensions, and hierarchies
- Define and model security filters and views
- Model aggregate cubes
- Publish cubes and packages
9.1.3 Deploy

When a dynamic cube is published, an administrator configures the cube in IBM Cognos Administration and makes it available for use. The administrator might perform the following tasks:

- Adjust the memory allocation for the query service and in-memory aggregates
- Assign users, groups, and roles to security views
- Monitor cube metrics
- Start and stop cubes
- Refresh the caches
- Enable logging
- Schedule administrative tasks

9.1.4 Run

After the dynamic cube is configured and started, the report author can create reports by using the various reporting applications, such as Business Insight Advanced, Report Studio, or Analysis Studio. In this stage, users can do the following tasks:

- Validate the design of the cube
- Author reports
- Execute reports
- Evaluate the performance to determine if any optimization is required

9.1.5 Optimize

The optimization stage is an optional stage. Only if the performance of the reports does not meet expectations does an administrator proceed to this stage. In this stage, administrators can do the following tasks:

- Tweak the various performance parameters in Cognos Administration
- Use the aggregate advisor in IBM Cognos Dynamic Query Analyzer to generate various recommendations for creating aggregate
- Create aggregate tables
- Add in-memory aggregates to the cube

9.2 Lifecycle across different environments

The various stages in the lifecycle, from start to finish, of the Cognos Dynamic Cube will occur over multiple environments. The actual number of environments might differ slightly from one customer to next, but a three-tiered approach is the most common setup:

- Development
  This is the primary environment used by cube modelers and report developers to design the cube and to create reports. It is the most flexible environment for allowing changes to be made. The database often contains only a subset of the production data. The objective in this environment is to provide functional objects to the test environment.

- Test
  This is a more rigid environment, geared for receiving deliverables from development and performing the necessary testing. The database scale must contain enough data to be able to validate performance.
- **Production**

  This is a highly controlled environment. Business analysts are actively using this system for real work, so changes in the environment should be made only after verifying the changes in the test environment.

  Figure 9-2 shows a flowchart of the dynamic cube lifecycle across development, test, and production environments.

![Figure 9-2 Dynamic Cube lifecycle across development, test, and production environments](image)

### 9.2.1 Mechanisms for moving dynamic cubes

Dynamic cubes can be moved from one environment into another by using one of two methods:

- Importing the cube by using a deployment archive
- Republishing the cube by using Cognos Cube Designer

**Importing the cube by using a deployment archive**

The simplest approach of the two options is to create a deployment archive in one environment and import it into the new environment. This approach fits in with how Cognos traditionally handles the movement between environments. When you create a deployment archive, selecting the **Include data sources and connections** check box causes all data sources (including dynamic cube data sources) to be included.
Figure 9-3 shows the Directory content page of the New Export wizard in IBM Cognos Administration.

![Directory content page of the New Export wizard in Cognos Administration](image)

The deployment archive will include the dynamic cube data source and its corresponding relational data source. The relational data source will still contain the connection information of the data source from the initial environment. Because each environment will likely have its own version of the relational data source, the administrator should update the connection information to point to the correct database, after the deployment archive is imported.

Although importing a deployment archive that contains a dynamic cube data source will work, in most cases of moving between environments, it cannot handle a couple scenarios. If either of the two following scenarios applies, then you must use the Cognos Cube Designer to publish the cube into the new environment instead of importing a deployment archive:

- The table names or table schemas differ between two environments.
- The dynamic cube in the target environment contains user mappings to dimensional security views that you do not want to be overwritten.

**Republishing the cube by using Cognos Cube Designer**

Republishing a cube by using the Cognos Cube Designer provides more flexibility over importing a deployment archive containing a dynamic cube data source. This approach does not have the limitations that users can encounter with the deployment archive when moving a dynamic cube between environments.
**Reusing a project file across separate Cognos Cube Designer instances**

If each environment has its own installation of Cognos Cube Designer, do the following steps:

1. Transfer the dynamic cube project file from the source environment to the target environment.
2. Launch Cognos Cube Designer and open the project file.
3. Right-click the cube you want, in the Project Explorer, and select **Publish**.
4. Complete the publish wizard.

**Redirecting Cognos Cube Designer to a different server**

If multiple environments share a single Cognos Cube Designer, do the following steps:

1. Run the Cognos Configuration tool and modify the “Gateway URI” and the “Dispatcher URI for external applications” properties to reflect the URI of the target server environment.

   Figure 9-4 shows the Configuration properties to modify when redirecting Cognos Cube Designer to a different environment.

   ![Configuration properties](image)

   **Figure 9-4**  Configuration properties to modify when redirecting Cognos Cube Designer to a different environment

2. Save your configuration settings.
3. Launch the Cognos Cube Designer and open your existing project.
4. Right-click the cube you want in the Project Explorer, and select **Publish**.
5. Complete the publish wizard.
Using Cognos Cube Designer to change data source, table schema, or table names

Cognos Cube Designer provides a simple mechanism for modelers to alter the data source name or table schema name that is used in the dynamic cube project. To handle differences in table names, the modeler needs to remap all measures and query items that rely on tables whose names differ in the new environment. Complete the following steps:

1. As described in the previous sections, either transfer your project file or redirect the Cognos Cube Designer to use the target server.
2. Launch the Cognos Cube Designer and open your existing project.
3. From the toolbar, click Get Metadata → Browse Content Manager Datasource.
4. Select the database schema from which to import data, and then click OK.
5. In the Project Explorer, expand the Data Sources folder and select the relational data source.
6. In the Properties tab, you can globally change the relational data source references and the table schema references. Modify the value in the Content Manager Data Source field to change the relational data source references. Modify the value in the Schema field to change the table schema references.

Figure 9-5 shows the Property in Cognos Cube Designer to modify the table schema.

![Figure 9-5 Property in Cognos Cube Designer to modify table schema](image)
7. If your table names do not match those that are used by the prior environment, remap each measure and query item that uses those table names.

8. Right-click the cube in the Project Explorer, and select **Publish**.

9. Complete the publish wizard.

### 9.2.2 Development

The development environment is the primary environment used by cube modelers and report developers to design the cube and to create reports. The modelers use the defined business requirements to create a cube model to satisfy those requirements. The cube should be modeled in an iterative fashion by defining the cube with a small number of hierarchies and measures, and testing them to make sure that they are correct before gradually expanding the cube to include more dimensions and more complex objects. If necessary, add security filters and views into the model.

When the cube is ready for deployment, the modelers might want to take advantage of the quick-deploy feature of Cognos Cube Designer to publish the cube, add the cube to the dispatcher, set the access account, and even start the cube. In this environment, users generally have more authority, and using the quick deploy feature to handle some of the administrative tasks can speed the iterative workflow.

After the cube starts, report authors create sanity reports to validate that the basic functionality of the cube is correct. If any of the dimensions, hierarchies, measures, or other components do not appear to be correct, the cube model returns to the modeler to investigate. This cycle repeats until the cube is satisfactory and the reports against it return the expected data. When this point is reached, the dynamic cube is transferred to the test environment by using one of the methods described in 9.2.1, “Mechanisms for moving dynamic cubes” on page 248.

If issues exist in the subsequent environments, work shifts back into the development environment for further investigation or updates. Additionally, if there are any in-database aggregate recommendations from the Optimize phase of the subsequent environments, those tables must be created and modeled as aggregate cubes in the development environment.

### 9.2.3 Test

When the dynamic cube is transferred into the test environment, a more comprehensive set of reports gets created and executed. If any of the dimensions, hierarchies, measures, or other components do not appear to be correct, the cube model returns to the modeler to investigate. After the testers in the test environment are satisfied that the cube is modeled correctly, they can start analyzing the performance of the reports.

If the performance of the reports is not acceptable, then the tester might adjust some of the tuning parameters on the cube or server. Another option is to enable workload logging, run the query workload, and run the Aggregate Advisor to obtain aggregate recommendations. In-memory aggregates can be applied immediately; in-database aggregates require some additional work. For any in-database recommendations, a DBA will need to create the tables in both the development and test environments. The development environment needs the in-database aggregates, because the modeler will use those tables to model the aggregate cube in the Cognos Cube Designer. After the aggregate cubes are added to the model in the development environment and the cube model is cycled back to the test environment, more performance testing occurs.
When the reports achieve a satisfactory level of performance, the cube, package, and reports can be deployed into the production environment.

9.2.4 Production

The production environment is likely to be a highly controlled environment. The changes that occur in this environment should be well tested before they are implemented at this level.

The final set of in-database aggregate tables must be created in the production environment. If the dynamic cube is transferred by using a deployment archive from the test environment, it must already contain the in-memory aggregates. If the dynamic cube is transferred by using the Cognos Cube Designer, then an administrator will need to use DQA to apply the in-memory aggregates to the production environment. If security is a requirement for the system, the administrator will also map users and groups to the security views.

Even if the test and production environment have mirror databases, a good idea is to make sure that all reports still run in the expected amount of time before going live.

9.3 Best practices for deploying between environments

This section contains various hints and tips to facilitate movement between environments during the dynamic cube lifecycle.

9.3.1 Keep a copy of the dynamic cube project file under source control

Consider the dynamic cube project file as the original version of the cube model. When the Cognos Cube Designer publishes a dynamic cube as a data source, it transforms the cube model in the dynamic cube project into a metadata format that the Dynamic Cube Server will use at run time. However, this transformation is one-way. There are no means to reverse-engineer a dynamic cube data source back into a dynamic cube project.

While the transformed metadata resides within the content store and can be transferred from machine to machine through a deployment archive, that cube metadata remains static. To make any changes to the design of the cube model, the modeler needs to update the original dynamic cube project from within the Cognos Cube Designer and publish it again. If the project file is ever lost, the modeler will need to re-create the cube from the beginning to modify the cube. Therefore, it is important to keep track of the project file even after a cube goes into production.

Keeping the project file under source control ensures that the file is backed up and helps to manage changes to the model, when multiple modelers are working on the same cube.
9.3.2 Use consistent data source, schema, and table names in all environments

In the cube model, the metadata objects for a dynamic cube contain references to the name of the relational data source, the relational tables, and the table columns. Having the fact and dimension tables defined with identical schema and table names in all environments helps to more easily move a cube between environments.

This approach offers the following benefits:

- You can use deployment archives to move a dynamic cube between environments.
- The same scripts can be used in all environments to create the in-database aggregates.

Inconsistent data source name and table schema names between environments can be addressed with a little extra work on the modeler, however, inconsistent table names require a sizable amount of effort and should be avoided.

9.3.3 Skip the optimize step in the development environment

The aggregate advisor considers many factors when devising the aggregate recommendations. Factors, such as the amount of data in the fact table and the cardinalities of the dimensions, influence the advisor. Therefore, running the advisor against a database that contains a small subset of your data can generate drastically different recommendations from the complete database.

These recommendations can be appropriate for the development environment, but might not provide the expected performance improvement in the test and production environments. With a fair amount of human effort required to create aggregate tables and to model the aggregate cubes, it does not make sense to have them created based on the small database in the development environment only to throw away that work when you move to the larger environment.

Therefore, the optimization stage should occur directly in the test environment, rather than trying to apply the optimizations from the development environment.

9.3.4 Use a separate database for each environment

A separate database should be used for each of the environments.

The database for development can be a smaller subset of the data. Because there will be numerous iterations and experimentation in the development system, having a smaller amount of fact data speeds the workflow.

During the development of reports, report authors might unintentionally create reports, leading to resource-intensive queries and negatively affecting the performance of other consumers of that database. The Aggregate Advisor might also generate resource-intensive queries during its run. Both scenarios are reasons why the development and test environments should not use the production database.
9.3.5 Include complete dimensional data in development environment

The development environment will usually have a substantially smaller subset of the fact table. However, if possible, the development environment should have complete dimension data or at least all dimension data that would be explicitly referenced in reports, calculations, or security.

When modeling dimensional security, the expressions for the security filters require modelers to drag in members from the member browser. The members that are displayed in the member browser are based on the data in the dimension tables. Without all the dimension data, users might not be able to define the security filters they want.

In addition, if there are problematic reports from the production environment, you want to be able to run the report in the development environment. If the development environment does not contain all the dimension data, the report specification might include members that do not exist in the development environment, and will fail to run.

9.3.6 Make the test system similar to the production system

The test system must be sufficiently similar to the production system in scale so that performance testing and optimization is meaningful.

In ideal circumstances, the database that is used in the test system is a mirror copy of the production database. This way allows the optimization and testing to occur in the test environment, and allows the recommended aggregates to remain applicable for the production environment. Even if the test system is not a mirror copy, if the test and production systems are of comparable scales (such as within a factor of two), then the recommended aggregates should still be acceptable. If a greater variance exists between the systems, the recommended aggregates might not work as well.

Running the advisor on the production system is generally undesirable, because the advisor might generate some resource-intensive queries while creating the aggregate recommendations.
Troubleshooting

This chapter describes the troubleshooting tools made available for visualizing, investigating, and resolving problems with Dynamic Cubes.

This chapter contains the following sections:

- 10.1, “Dynamic Query Analyzer” on page 258
- 10.2, “Visualizing dynamic cube startup” on page 263
- 10.3, “Reviewing Cognos reports with Dynamic Query Analyzer” on page 268
- 10.4, “Virtual cube query execution tracing” on page 277
- 10.5, “Dynamic cube QueryService logging” on page 278
- 10.6, “Dynamic cube query plan tracing” on page 279
- 10.7, “Aggregate matching” on page 279
- 10.8, “Common problems” on page 279
10.1 Dynamic Query Analyzer

Dynamic Query Analyzer provides Cognos report developers and system administrators with the ability to visualize and review the composition of queries to a dynamic cube and the relational source data the dynamic cube is established from.

Visualizations provided by Dynamic Query Analyzer assist in the understanding of queries requested to satisfy a report, their cost, and their ability to take advantage of existing cached information. Dynamic Query Analyzer can help identify performance impacts of report design elements.

This chapter focuses on the features of Dynamic Query Analyzer that support troubleshooting and investigation into Cognos Dynamic Cubes and their associated reports.

Dynamic Query Analyzer is the primary query visualization tool for Dynamic Query Mode, and many of the areas discussed in this chapter can be applied to reports that are not based on dynamic cubes.

10.1.1 New Features

The Dynamic Query Analyzer user interface is extended to support visualization and troubleshooting capabilities for Dynamic Cubes. Some generic feature extensions were also added.

Aggregate Advisor
Aggregate Advisor is a performance optimization utility that is embedded in Dynamic Query Analyzer. You can use Aggregate Advisor to provide guidance on aggregate table definition based on dynamic cube workload. Cognos Dynamic Cubes takes advantage of aggregate table and in memory aggregate definitions for aggregated data results, optimizing query performance. Aggregate Advisor is explored in Chapter 8, “Optimization and performance tuning” on page 201.

This section reviews how Dynamic Query Analyzer provides tools to support the report author and system administrator confirming query performance through the expected use of Cognos Dynamic Cubes aggregate cache and the use of aggregate tables during query execution.

Cognos Dynamic Cubes query nodes
Dynamic Query Analyzer is extended to include Dynamic Cubes query nodes, which detail the use of Dynamic Query caches and the SQL that is passed to an underlying data source where a query is not satisfied from available cache.

The cube start process is also reviewed and how, with Dynamic Query Analyzer, we can explore the cube-startup process and dimension member caching.

Chapter 2, “IBM Cognos Dynamic Cubes architecture” on page 9 details the architecture of dynamic cubes, cache optimization, and the caching strategy used during query execution.

Report-based Dynamic Query logging
Dynamic Query Analyzer now supports the ability to apply changes to the logging level of Dynamic Query Mode server component on an individual report run basis. Logging of the server component for Dynamic Query traces server activity and is distinct from query planning tracing and query execution tracing. Previously, the logging level of the server component for Dynamic Query could only be amended on the Cognos 10 Business
Intelligence Server. A report author or administrator can now selectively monitor messages from the dynamic query mode QueryService for a specific report.

### 10.1.2 Configuring Dynamic Query Analyzer

Before using Dynamic Query Analyzer, it must be configured to connect to a target Cognos instance whose reports and logs are to be reviewed.

Details of how to configure the Cognos environment that Dynamic Query Analyzer is to connect to are also outlined in Chapter 8, “Optimization and performance tuning” on page 201, but included here for convenience.

**Set preferences for Cognos Server**

Use the following procedure to set preferences for Cognos Server.

1. Launch Dynamic Query Analyzer.
2. Open Dynamic Query Analyzer preferences and select **Window → Preferences**.
3. Select **Cognos Server** from the Preferences navigation pane (Figure 10-1).
4. Enter the target Cognos environment dispatcher and Gateway URIs.
5. If your Cognos environment requires authorization:
   a. Select **Refresh** to populate the list of available authentication Namespaces.
   b. Select a target Namespace from the available choices.
   c. Enter Name and Password details of an authorized user within the selected Cognos Namespace.
6. Select **Apply** to save the configuration preferences.

![Figure 10-1 Set Dynamic Query Analyzer Cognos Server Preferences](image)
**Set preferences for logging folder URI**

If query execution logs are to be accessed through a web server (rather than the file system) enter the URL details from which logs can be accessed:

1. Select **Logs** from the Preferences navigation pane (Figure 10-2). If the preferences pane is closed, open the Preferences dialog and select **Window → Preferences** from the Dynamic Query Analyzer menu.

2. Enter the log folder URL. Your Cognos system administrator can provide details for how to use and connect remotely to the Cognos Server dynamic query mode logs directory.

3. Enter Name and Password details provided by your administrator if authorization is required to access the logs directory through a web connection.

4. Select **Apply** to save your configuration.

5. Select **OK** to close the Preferences window.

![Figure 10-2 Set Dynamic Query Analyzer logs folder URL preferences](image)

### 10.1.3 Query execution tracing

To gather log and trace information for exploration in Dynamic Query Analyzer, enable query execution tracing.

Query execution tracing can be enabled for all requests to a designated Dynamic Query Mode (QueryService). This trace is configured from within Cognos Connection. Appropriate system administration privileges are required to enable query execution. Query execution tracing for an individual report can also be enabled in Dynamic Query Analyzer.
Enable server query execution tracing

Use the following procedure to enable query execution for a target QueryService, tracing all requests including cube startup:

1. Open Cognos Connection.
2. Click the Launch Menu and select IBM Cognos Administration.
   
   If IBM Cognos Administration is not available as an option from the Launch menu, and you believe you should have access to Cognos system administration functionality, contact your Cognos systems administration team.

3. Select Status.
4. Select System.
5. Select the Cognos Server and dispatcher that is running Cognos Dynamic Query Mode (QueryService) and dynamic cube you are targeting for query execution tracing.
7. Select Set properties from the QueryService context menu.
8. Select the Settings tab.
9. Select the check box in the Value column for Enable query execution trace to establish server based query execution tracing. See Figure 10-3.
10. Click OK to enable tracing and return to Cognos Connection.

Figure 10-3 Enable QueryService query execution tracing
Disable server query execution tracing
Disable server query execution tracing as follows:
1. Deselect the check box for the **Enable query execution trace** property.
2. Click **OK** to disable tracing and return to Cognos Connection.

Enable Dynamic Query Analyzer query execution tracing
To help you with troubleshooting, Dynamic Query Analyzer provides the option to enable query execution tracing for reports that are run in the Dynamic Query Analyzer interface. To enable query execution tracing for a report run from Dynamic Query Analyzer, use the following steps. If server-based query execution tracing is inactive, only those reports that are run from Dynamic Query Analyzer will produce query execution logging.
1. Launch IBM Cognos Dynamic Query Analyzer.
2. Select **Windows → Preferences**.
3. Select **General** from the Preferences navigation pane (Figure 10-4).
4. Select **Query execution trace**.
5. Set the logging level.
6. Click **OK** to apply changes and close the Preferences dialog.

![Figure 10-4 Set DQA query execution tracing](image)

**Note:** Query execution tracing is enabled by default within Dynamic Query Analyzer.

Disable Dynamic Query Analyzer query execution tracing
Disable query execution tracing as follows:
1. Deselect the **Query execution trace** check box.
2. Click **OK** to disable tracing and close the Preferences dialog.
10.2 Visualizing dynamic cube startup

Chapter 2, “IBM Cognos Dynamic Cubes architecture” on page 9 outlines the architecture of dynamic cubes and the caching strategy available to optimize performance, when basing reports on Cognos Dynamic Cubes.

With Dynamic Query Analyzer, you can review the startup sequence and queries that are issued to the source database to retrieve dimension members and their associated attributes.

For each cube start, query execution logs are recorded in the <CognosInstall>/Logs/XQE folder in a designated cube folder with a Cube_<CubeName> format.

A separate trace is produced for each dimension that is defined in the cube. A dimension with multiple hierarchies will produce two query execution traces.

Dimension load traces are logged in the following format:
<Dimension Name>_<Hierarchy> (<Date>, <Timestamp>)

Figure 10-5, the Open Logs dialog for Dynamic Query Analyzer, shows the traces that are created at startup.

Figure 10-5 Dynamic Query Analyzer Open Logs

The gosldw_sales dynamic cube is modeled as 10 dimensions, which are the only ones loaded at startup. Each dimension is cached in the dynamic cube at startup as outlined in Chapter 2, “IBM Cognos Dynamic Cubes architecture” on page 9.

Use the following steps to review the SQL query and number of rows that are retrieved for the Branch dimension at cube startup.

1. “Enable server query execution tracing” on page 261.
2. Start the GoSales Dynamic Cube (gosldw_sales).
3. Click the Launch menu and select IBM Cognos Administration.
4. Select Status.
5. Select System.
6. Select the Cognos Server and dispatcher instance.
7. Select QueryService.
8. Select the context menu for the gosldw_sales dynamic cube.
9. Click Restart from the context menu (Figure 10-6).

![Figure 10-6   Restart a dynamic cube](image)

10. Return to Cognos Connection.
11. Launch Dynamic Query Analyzer.
12. Select File → Open log.
13. Select From URL, override the default URL defined as a DQA preference as appropriate.
14. Look for a log named Branch_Branch (<date>, <time>), where <date> and <time> correspond to the date and time that the cube was started and the Branch dimension started loading. For example, Branch_Branch (Aug 7, 9:30:18).
Figure 10-7 shows the Dynamic Query Analyzer Branch dimension query visualization.

Dynamic Query Analyzer displays an analysis of the log in window-based views:

- The Summary window provides top-level analysis of the logs, separated into three sections:
  - Summary: An overview of the log details being reviewed.
  - Timing: A summary of the total time within the QueryService to satisfy the data request and a breakdown, in descending order, of the Working and Wait times within the execution tree.
  - Analysis: A summary of the execution planning and query hints applied during planning.

A fourth section, Nodes Shapes and Colors, provides a key to Execution Tree Node types displayed in the Graph window.

- The Graph window displays the query execution plan as a tree of execution nodes. The nodes and their role can be determined from the key included in the Nodes, Shapes and Colors section of the Summary window.

- The Properties window displays property values associated with each node in the query execution tree as it is selected.

- The Query window details the query executed against a data source to retrieve data to satisfy the report request.
By reviewing Dynamic Query Analyzer's execution trace for the Branch dimension, you can determine, from the Timings section in the Summary window, that the overall time spent in the QueryService to retrieve data from the source relational database was 5787.3545 milliseconds (approximately 6 seconds).

The XSQL node of the execution tree details the number of rows returned from the source database from which the dimension and hierarchy is populated and the overall time taken for the query on the database to complete. The SQL query used to populate the Branch dimension returns 972 rows. The SQL response from the source database is completed in approximately 4 seconds.

Figure 10-8 shows the dimension member count and SQL execution time.

![Figure 10-8 Dimension member count and SQL execution time](image)

It is common for many dynamic cube models to include one or two dimensions with a very large member count, often a products or customer dimension. When you investigate cube startup, you should consider the largest dimensions in the cube model first and their load profile.

The **Re-run the SQL statement on the server** button (Figure 10-9) becomes available when SQL in the query window is selected. The rerun option enables further analysis of query execution to be undertaken with monitoring tools from the source relational database.

The SQL can also be copied from the Query window and rerun to the source relational database by using a preferred SQL editor.
By reviewing the SQL row count that is returned by the Branch dimension query (927) and the number of leaf level members in the Branch dimension is expected to be in the dimension (29), you see a possible discrepancy, which must be considered further.

In reviewing the SQL in the Query window, you see that the Branch dimension includes a join to the employee dimension table (EMP_EMPLOYEE_DIM). This join might be unnecessary, and might be causing the Branch query to return more rows than necessary, and extend the query execution time, delaying cube start.

The model definition of the Branch dimension can be verified in IBM Cognos Cube Designer. Checking the member caption, key, description, and attribute definitions for the Branch level of the Branch dimension, the **Branch code** attribute is mapped using the BRANCH_CODE query item from the Employee dimension table (EMP_EMPLOYEE_DIM). The inclusion of the Branch code from the employee table is introducing the join to the Employees dimension table, a join which may be unnecessary.

Figure 10-10 shows the Cognos Cube Designer gosldw_sales Branch dimension.

![Figure 10-10](image)

Reviewing the model design further, the sales fact maintains a relationship to employee rather than Branch. Branch is a snowflake dimension of employee. To maintain model integrity, the Branch dimension maintains a reference to employees and fact data held in the measure table.

The level unique key mapping on BRANCH_KEY ensures the Branch dimension is loaded correctly with distinct Branch members.

The model design is correct and the SQL that is generated to support the generation of the Branch dimension is optimal.
10.3 Reviewing Cognos reports with Dynamic Query Analyzer

The Content Store view can be used to browse through content held within a Cognos Enterprise system. A selected content object can be run to produce execution and planning traces for review.

A report query execution trace is stored on the IBM Cognos 10.2 Business Intelligence Server and is found relative to the Cognos installation directory under the logs/XQE folders.

A report query execution trace is recorded with the following folder structure and file profile:
FolderName: <Run Date>_<Timestamp>_<ObjectName>

In Dynamic Query Analyzer, a report query execution trace is shown in the Content Store view as children of the report object. Query Execution traces can also be browsed in the Report Logs view.

Use the following procedure to generate a report:

1. Open Dynamic Query Analyzer.
2. Select Window → Preferences → Content Store → Open the latest log to have Dynamic Query Analyzer automatically open the most recent query execution log associated with a report run. The default preference is to prompt to open the most recent log that follows a report run.
3. Verify the Content Store view is visible. Select File → Open IBM Cognos Portal.

Note: You can also make the Content Store view visible from the Show View dialog:
   a. Select Window -> Show View.
   b. Select Content Store from the Navigation folder in the Show View dialog.

4. In the Content Store view, expand folders Dynamic Cube → Reports to browse the available Cognos content to identify the target report BranchProfileByYear.
5. Right-click the context menu on the BranchProfitByYear report, and select Run Report.
6. The report output is displayed in a new Browser view tab.
Chapter 10. Troubleshooting

Figure 10-11 shows the Dynamic Query Analyzer Report View.

![Dynamic Query Analyzer Report View](image)

Figure 10-11  Dynamic Query Analyzer Report View

7. The report execution tree that is associated with the report opens in a new graph view.

**Note:** A report query execution trace can also be loaded from the Report Logs view. Display the Report Logs view (if it is not visible) and load a report query execution trace:

a. Select **Window → Show View**.

b. Select **Report Logs** from the Navigation folder in Show View dialog.

c. Click **OK** to close the dialog.

d. Browse the Report Logs view by report name and timestamp to identify the report query execution trace folder.

e. Expand the report trace folder and double-click **Profile 0**, which is the execution profile.
Figure 10-12 shows the Dynamic Query Analyzer BranchProfitByYear query execution trace.

10.3.1 Reviewing the query execution trace

Dynamic Query Analyzer introduces additional nodes which map to objects within the dynamic cube caching strategy.

Use of dynamic cube caching and requests to the underlying relational database are represented as execution tree nodes of XQueryStrategy. The XQueryStrategy node summarizes the total time spent in retrieving data values for use in satisfying the MDX request for the report against the dynamic cube.

The following nodes are in the Dynamic Query Analyzer:
- XDataCacheRetrieval
  The XDataCacheRetrieval node represents the total time spent satisfying the data request that is reusing data that is held in the data cache.
- XAggregateCacheRetrieval
  The XAggregateCacheRetrieval node represents the total time spent satisfying the data request that is reusing existing data in the aggregate caches.
Chapter 10. Troubleshooting

- XAggregateCubeRetrieval
  The XAggregateCubeRetrieval node represents the total time spent satisfying the data request that uses external aggregate tables, and caching the data returned.

- XCubeRetrieval
  The XCubeRetrieval node represents the total time taken satisfying the data request through requests to the underlying (base) relational data and caching the data returned.

Chapter 2, “IBM Cognos Dynamic Cubes architecture” on page 9 describes details of the caching strategy that employed for Cognos Dynamic Cubes and the data caches that are available to optimize performance.

**Note:** A query execution trace does not record entries against XQueryStrategy child nodes unless the dynamic cube data retrieval component they represent was used to satisfy the MDX data request for the report.

Because the report request for the BranchProfitByYear report is the first report to be requested, there is no result set cache or data cache available from which the results can be obtained. No aggregate table match is identified.

This is also confirmed by reviewing the gosl_dw_sales dynamic cube monitor which confirms the absence of a data cache hit ratio.

Figure 10-13 shows the gosl_dw_sales dynamic cube metrics.

![Figure 10-13  Gosldw_sales dynamic cube metrics](image)
10.3.2 Subsequent run requests: Taking advantage of the result set cache

As Chapter 2, “IBM Cognos Dynamic Cubes architecture” on page 9 describes, a repeat run of the BranchProfitByYear report can be expected to take advantage of the QueryService result set cache, eliminating the need to request data from the dynamic cube, and minimizing the time that is recorded against the XQueryStrategy and child nodes.

To confirm this behavior, use the following procedure to rerun the BranchProfitByYear report from Dynamic Query Analyzer:

1. Select the Content Store view.
2. Expand the Dynamic Cube → Reports folder to explore the Cognos content view and locate the BranchProfitByYear report.
3. From the content object context, select Run Report.
4. Open the query execution trace generated for the most recent report run.

Figure 10-14 shows the BranchProfitByYear query execution trace.

The XMdxSelect node records a wait time of 0.86 milliseconds, and no values are recorded against the XQueryStrategy (or its child nodes).

The query result was taken from the available results in the result set cache.
10.3.3 Confirming the role of caching

To confirm the role of the result set cache and the presence of data in the data cache or the aggregate cache, the result set cache can be disabled and the report can be rerun. Use the procedure that is described in this section.

The role of the result set cache can be confirmed by disabling the result set cache.

Note: The dynamic cube must be restarted before the updated setting to disable the result set cache takes effect.

Complete the following steps:
1. Go to IBM Cognos Administration.
2. Select Status.
3. Select System.
4. Select the server.
5. Select the dispatcher.
7. In the QueryService context menu, select Set Properties.
8. Select the Settings tab.
9. Select Edit the dynamic cube configurations.
10. Each cube has its own result set cache, so select the pencil icon next to gosldw_sales to edit the cube’s configuration properties.
11. Check the Disable result set cache check box.
12. Click OK to apply the change.
13. Return to the dispatcher services scorecard by clicking OK a further two times.
14. Select the QueryService from the list of available services.
15. From the Dynamic Cube context menu, select Restart.

Note: Wait for a minute for the applied changes to take effect.

Run the BranchProfitByYear report once. Then re-run the BranchProfitByYear report in Dynamic Query Analyzer and open the report’s query execution trace.

Data held in the data cache is re-used to satisfy the report query and the total time taken to obtain the data from the data cache is recorded in XDataCacheRetrieval node (220.572 milliseconds).

The data required for the report could be fully satisfied from the data cache. Therefore the XAggregateCache, XAggregateCubeRetrieval and XCubeRetrieval nodes do not record values.
Figure 10-15 shows the BranchProfitByYear, the result set cache is disabled.

Note: When you explore the role of caching on a report and the visualization in Dynamic Query Analyzer, remember that manually clearing the dynamic cube member cache will implicitly clear dependent caches, such as the data cache and aggregate cache.

10.3.4 Reviewing Report SQL

The SQL that is passed to the relational database can be viewed in Dynamic Query Analyzer in these situations:

- When the data that is required to satisfy a report request is not available from one of the dynamic cube caches
- When, for optimization reasons, the query strategy determines that the request (or part of) should be managed by the relational database that supports the dynamic cube

The presence of working and wait time against the XQueryStrategy node indicates that data was retrieved from the relational database of the dynamic cube.
Use the following steps to view the SQL requests that are passed to the relational database:

1. Right-click the XMdxSelect node to display the context menu of the node. The XMdxSelect node is the parent node of XQueryStrategy.
2. Select **Open sub queries** from the context menu to open all the sub query execution plans that are generated for the report execution. Each sub query opens in its own tab.

**Notes:**
- If the **Open sub queries** context menu option is unavailable, no sub query requests have been recorded.
- Sub query execution plans can also be opened from the Report Logs view. Sub Query execution plans are nested folders under the parent report execution plan folder. Individual sub query execution plans can be opened by selecting the sub query Profile node.

### 10.3.5 Verifying changes to the BranchProfitByYear Report

The BranchProfitByYear report was extended by the report development team to help the business understand employee sales performance and their interaction with the sales channels of the company. However, the revised report is slow and is unexpectedly returning no sales data.

You can use Dynamic Query Analyzer to understand the performance difference and the report elements that are introducing the changed profile.

Use the following steps to review the report query execution plan:

1. Open Dynamic Query Analyzer.
2. Enable query execution tracing.
3. From the Content Store view, locate the report to be traced, and from its context menu select **Run Report**.
4. Open the query execution trace from the Report Logs view if the execution trace does not open automatically.
Figure 10-16 shows the query execution trace for the BranchProfitByYearExtended report.

The report query execution trace highlights that the majority of the query execution is taken within the XMdxIfExpression node. Of the 75883.03 milliseconds taken, the total time taken within the query execution, 71659.2163 milliseconds, is spent determining appropriate values for the Gross profit report measure. From the execution trace, the report design seems to be taking advantage of a report expression to determine the appropriate Gross profit and excluding the value for Gross profit where the gross profit is 900000.

To verify the report design, open the report in Report Studio, and explore the report expression definition for the Gross Profit Data item. The following expression calculates the report's Gross Profit measure:

IF ( [gosIdw_sales].[Measures].[Gross profit] <> 900000 ) THEN ( gosIdw_sales].[Measures].[Gross profit] ) ELSE ( NULL )

It is appropriate to consider the definition of the report expression for Gross profit and whether the expression is erroneously included in the report definition.
By reviewing the MDX trace in the query window further, you can see that Order Method data item is a member set, restricted by a set filter or a data item with a filter expression:

```
FILTER(
  [Order method].[Order method].[Order method].MEMBERS,
  NOT (IIF(VBA!INSTR([Order method].[Order
Method].CURRENTMEMBER.PROPERTIES("description"), "EMAIL" )<>0, VBA!MID([Order method].[Order Method].CURRENTMEMBER.PROPERTIES( "description" ), VBA!INSTR([Order method].[Order method].CURRENTMEMBER.PROPERTIES("description"), "EMAIL" ), VBA!LEN("EMAIL"))= "EMAIL" , 0))
)
```

The set filter is intended to remove the email channel from the set of sales channels for which profitability is being assessed. However, the filter expression does not match the member description value for the email element, and the email sales channel remains in the report output. The filter expression must be adjusted if the email element is to be restricted.

A further filter is defined to restrict the branch-based employees to be included on the report, to employees who have recorded sales figures. The MDX trace in the Query view shows the report is including the filter expression:

```
FILTER([Employee by region].[Employee by region].[Employee].MEMBERS,
  ([Measures].[Gross profit], [Time].[Time].DEFAULTMEMBER,
   [Order method].[Order method].DEFAULTMEMBER,
   [Branch].[Branch].DEFAULTMEMBER)<= 0))
```

However the filter expression (<= 0) is restricting the list of employees to be included in the report, to employees who have no profit values recorded for them (unless a loss is recorded). The report output is therefore not yielding the anticipated results.

To restrict the report to only those employees with sales (profit) values, change the filter expression to greater than or equal to zero (>= 0).

### 10.4 Virtual cube query execution tracing

For a virtual cube, the XQueryStrategy and child nodes represent the total time spent to satisfy the MDX data request from the underlying cubes from which it is comprised. The property values represent the cumulative totals across the dependent cubes.

Correspondingly, the virtual cube will not display any subquery nodes that represent the SQL queries that are passed to the underlying relational databases.

Query execution tracing on the dynamic cubes that comprise the virtual cube should be enabled to explore the query execution behavior for requests that are satisfied for the virtual cube.
10.5 Dynamic cube QueryService logging

Logging for the QueryService managing a dynamic cube and associated report executions can be enabled through Dynamic Query Analyzer. Logging that is enabled in Dynamic Query Analyzer is active only for those reports run from Dynamic Query Analyzer. Dynamic Query logging provides insight into issues during report execution that are not available from the query execution log.

QueryService logs are recorded under the `<CognosInstall>/Logs/XQE` folder in files with the following name format. A new log is created when the QueryService is started.

`xqelog-date-timestamp.xml`

To enable QueryService logging, use the steps described 10.6, “Dynamic cube query plan tracing” on page 279. Then, select the Dynamic Query check box logging option on the Preferences → General dialog, available from the Window menu option.

For QueryService logging, you can set the level of detail that is added to the log file by selecting one of the log level radio buttons. You can choose one of four levels, described in the following list. Logging at more detailed levels can also include log messages from lower levels. Establishing the log level at Info, for example, will include indications that are logged at the Info, Warn, and Error levels. Verbose level indications will not be present in the file.

- **Verbose**: This level is the most detailed tracing level. Extensive and detailed tracing of execution flow through the QueryService components will be logged. Verbose logging can create significant amounts of log information.
- **Info**: This level contains detailed status messages, although the detail is more restricted than Verbose logging. Info level is the default level of dynamic query logging.
- **Warn**: This level provides status indications that the administrator or developer might need to address.
- **Error**: This level details critical incidents and unrecoverable errors that the administrator or developer should be aware of because system behavior can be affected.

Use the following steps to view the QueryService log for a report, in Dynamic Query Analyzer:

1. Ensure dynamic query logging is enabled.
2. Run a report from the Content Store view.
3. Open the report’s query execution trace from the Report Logs view, if the trace is not opened automatically.
4. Select File → Show in Server Log to open the Dynamic Query log.
Figure 10-17 shows the dynamic Query log for Report 2.BranchProfitByYear.

10.6 Dynamic cube query plan tracing

To supplement query execution tracing, you might need to also enable query plan tracing. This step is normally done with guidance from the IBM Support team.

Similar to query execution tracing, query plan tracing can be enabled for individual reports run through Dynamic Query Analyzer. To enable query plan tracing, select the check box next to the Query Planning trace option on the Preferences → General dialog available from the Window menu option.

10.7 Aggregate matching

The steps to follow to troubleshoot issues related to the aggregate caching strategy are listed in 8.4.4, “In-memory aggregate tips and troubleshooting” on page 226.

10.8 Common problems

This section describes several problems you might encounter during troubleshooting.
10.8.1 Configuring Cognos Dynamic Cubes logging

This section describes how to configure logging for dynamic cubes to capture information that can help with diagnosing the issue.

Logging is configured by using the configuration/xqe.diagnosticlogging.xml file. It is enabled by specifying a level of either Info or Trace for an event group, such as in the following example:

```xml
<eventGroup name="ROLAPCubes.Management" level="trace"/>
```

Separate event groups exist for separate parts of the code.

The logging information is written to files by using the following format:
`logs/XQE/xqelog-*.*.xml`.

The following sections describe several common problem areas.

Dynamic Query Server configuration (JVM parameters) or Dynamic Query Server not starting

If a dynamic cube fails to start, you can enable the "InitTerm" event group, which causes the DQM server to write additional information regarding the initialization of a dynamic cube to the log file.

- Event group or groups to change:
  ```xml
  <eventGroup name="InitTerm" level="trace"/>
  ```

- Sample logging information returned:
  ```xml
  <event ...><![CDATA[Using JVM heap sizes: -Xms1024m, -Xmx1024m, -Xmn512m, -Xmns512m]]></event>
  ```

Dynamic Query Server configuration (non-JVM parameters) or Dynamic Cube configuration

When you are diagnosing a problem, a helpful step is to confirm the configuration parameters that are applied to a cube. Among other information, enabling the "ROLAPCubes.Management" event group will write the configuration parameters of the cube to the log file.

- Event group or groups to change:
  ```xml
  <eventGroup name="ROLAPCubes.Management" level="trace"/>
  ```

- Sample logging information returned:
  ```xml
  <event ...><![CDATA[Configured ROLAP advanced settings values: qsMaxCubeLoadThreads=16,...]]></event>
  <event ...><![CDATA[[Updating a ROLAP configuration parameter: rolap.qsMultiDimensionalQuerySizeLimit=0]]></event>
  <event ...><![CDATA[[Before: ROLAP configuration: [ROLAPConfiguration:...,qsMultiDimensionalQuerySizeLimit=0,...]]]]></event>
  <event ...><![CDATA[[After: ROLAP configuration: [ROLAPConfiguration:...,qsMultiDimensionalQuerySizeLimit=0,...]]]]></event>
  ```
Dynamic cube management (start, stop)
Depending on the problem being diagnosed, a useful step is to track the various administrative commands that are issued to a cube, such as start and stop. Enabling the "ROLAPCubes.Management" and "ROLAPCubes.Info" event groups will add this information to the log file.

► Event group or groups to change:
  <eventGroup name="ROLAPCubes.Management" level="trace"/>
  <eventGroup name="ROLAPCubes.Info" level="info"/>

► Sample logging information returned:
  <event ...><![CDATA[Manager task submitted cube start with source cubes task for cube RolapSimpleCube_V5_DB2]]></event>
  <event ... rolapCube="RolapSimpleCube_V5_DB2"><![CDATA[Cube start executing]]></event>
  <event ... rolapCube="RolapSimpleCube_V5_DB2" ...><![CDATA[Cube start succeeded]]></event>
  <event ...><![CDATA[Manager task retrieved the status of the cube start with source cubes task for cube RolapSimpleCube_V5_DB2. The cube start with source cubes task was successful.]]></event>

Dynamic cube hierarchy load
If a cube encounters problems when it starts, a helpful step might be to obtain additional information about the loading of each of the hierarchies in the cube, one of the primary activities during cube startup. Enabling the "ROLAPCubes.Loader" event group will add information about the loading of hierarchies to the log file.

► Event group or groups to change:
  <eventGroup name="ROLAPCubes.Loader" level="trace"/>

► Sample logging information returned:
  <event ... rolapCube="RolapSimpleCube_V5_DB2" rolapDimension="Product" rolapHierarchy="Product" ...><![CDATA[Hierarchy load began.]]></event>
  <event ... rolapCube="RolapSimpleCube_V5_DB2" rolapDimension="Product" rolapHierarchy="Product" ...><![CDATA[Started populating level members for hierarchy [Product]]]></event>
  <event ... rolapCube="RolapSimpleCube_V5_DB2" rolapDimension="Product" rolapHierarchy="Product" ...><![CDATA[[Product] loaded 3 members.]]></event>
  <event ... rolapCube="RolapSimpleCube_V5_DB2" rolapDimension="Product" rolapHierarchy="Product" ...><![CDATA[Finished populating level members for hierarchy [Product]]]></event>
  <event ... rolapCube="RolapSimpleCube_V5_DB2" rolapDimension="Product" rolapHierarchy="Product" ...><![CDATA[Hierarchy load succeeded.]]></event>
**Dynamic cube reservation**
So that certain operations can be performed on a cube, a lock on a cube must be obtained. Locks are obtained by requesting a reservation on a cube, which, when granted, locks the cube. Enabling the "ROLAPCubes.Reservation" event group will add information about cube reservations to the log file.

- Event group or groups to change:
  <eventGroup name="ROLAPCubes.Reservation" level="trace"/>

- Sample logging information returned:
  <event ...><![CDATA[Query nesting count: 1]]></event>
  <event ...><![CDATA[Query nesting count: 0]]></event>
  <event ...><![CDATA[Changed reservation count from 0 to 1]]></event>
  <event ...><![CDATA[Changed reservation count from 1 to 0]]></event>

**Dynamic cube locking**
Enabling the "LockManagement" event group will add information about cube locks to the log file.

- Event group or groups to change:
  <eventGroup name="LockManagement" level="trace"/>

- Sample logging information returned:
  <event ...><![CDATA[The lock on the resource [RolapSimpleCube_V5_DB2].[state] was requested in exclusive mode.]]></event>
  <event ...><![CDATA[The lock on the resource [RolapSimpleCube_V5_DB2].[state] was acquired in exclusive mode.]]></event>
  <event ...><![CDATA[The thread pool-3-thread-1 (current) holds locks on the resources: [ [RolapSimpleCube_V5_DB2].[state]-X ] ]]]></event>

**10.8.2 Cube does not start**
The following message indicates a cube that does not start:
XQE-ROL-0051 Unable to logo to the access account specified for the dynamic cube: [cube name].
Figure 10-18 shows the cube that does not start.

Assuming a third-party namespace is in use and dynamic cube data source (or sources) exist, the following steps must be followed to add a dynamic cube configuration to the QueryService and start a cube:

1. Add an access account to the dynamic cube data source.
2. Go to IBM Cognos Administration → Configuration → Data Source Connections.
3. Identify the dynamic cube data source and access its properties by selecting More and then selecting Set properties. Figure 10-19 shows the data source image.
4. From the Set properties page, click **Select the access account**. See Figure 10-20.

![Figure 10-20](image_url)

**Figure 10-20**   Select the access account

5. Select a user from the third-party namespace. See Figure 10-21.

![Figure 10-21](image_url)

**Figure 10-21**   Select User
6. Log in as the user who holds the access account, and go to My Preferences → Personal. The Credentials section shows the following text (Figure 10-22):

Specify the users, groups or roles that can use the credentials to run activities.

- If you do not have a valid credential, then after the text, a hyperlink shows First, you must create the credentials.

- If you see the following text, then you already have a valid credential:
  You can also renew the credentials. Renew the credentials.

![Figure 10-22 Set user credentials](image)

7. Click the hyperlink; the page refreshes and your credentials are displayed.

10.8.3 Unexpected data values returned

Because a requested value might be from various caches, it is often necessary to determine where the incorrect value comes from. The technique to determine from where the value is retrieved involves disabling and then enabling the various caches.

Disabling all dynamic cube caches

Figure 10-23 on page 286 shows a cube with all the caches disabled. A cube with all the caches disabled causes a query to access the database fact table to retrieve the data.
Enabling in-database summary tables
Enabling allows the query to access in-database summary tables to retrieve the data. To enable this cache, clear the check mark. See Figure 10-24.

Enabling the in-memory aggregate cache
This will allow query to access the in-memory aggregates created as part of the Aggregate Advisor process. These aggregates are typically used by queries performing high-level analysis.

To enable this cache, set the value of the **Maximum amount of memory to use for aggregate cache (MB)** to the value that is used when running the Aggregate Advisor. See Figure 10-25.
Enabling the in-memory query (data) cache
This cache maintains the results of the most recently executed queries. To enable this cache, remove the check mark. See Figure 10-26.

![Figure 10-26 Enable result set cache](image)

**Note:** When making any changes to configuration settings of a cube, the cube must be restarted for the changes to take effect.

Wrong numbers from in-memory aggregate
See 8.4.4, “In-memory aggregate tips and troubleshooting” on page 226.

Incorrect auto summary value
An incorrect value might be because of security restrictions. This subject is too large to cover in a troubleshooting section. If security was applied to the cube and incorrect summary values are being returned, you should thoroughly review Chapter 7, “Dimensional security” on page 147 to ensure that this security is not the cause of the issue.

Query returns empty cell for a calculated member
This issue is likely caused by the reference of a secured member by a calculated member (not a measure). The reason is because the value of the secured member is treated as null in the calculation. Further details of calculated members are in 7.7, “Calculated members” on page 164.

Wrong numbers involving multigrain dimensions
This problem might be the result of incorrect modeling. This problem occurs when facts exist at different levels of the same dimension, for example monthly sales target and daily sales totals. If modeled incorrectly, a query might be generated that incorrectly displays values, which is sometimes referred to as *double counting.*
10.8.4 Cube loaded but relative time members are missing

In Cognos Cube Designer, under Time hierarchy, make sure to set the value to true for the Add Relative Time Members property. See Figure 10-27.

Redeploy the cube model and restart cube again.

Another cause of the relative time members not being displayed correctly is a bad expression in a current period expression, for example, a typographical error, missing quotation marks around a string, syntax error in a V5 expression, and so on.
In Figure 10-28, a Current Period expression is selected as valid within the cube modeler.

![Figure 10-28  Valid Current Month](image)

However, when the metadata tree is expanded in Cognos Workspace, none of the relative time members is listed. See Figure 10-29.

![Figure 10-29  No relative time members](image)
To resolve this issue, check the XQE server log to see the error or warning about the current period. Enable query execution trace in QueryService to see even more detail, and stack trace in XQE server log.

In this case, the errors shown in Example 10-1 are in the log.

Example 10-1  XQE server log

```
<event component="XQE" group="ROLAPCubes.Loader" level="ERROR" thread="66"
timestamp="2012-08-13 17:22:39.123" contextId="3"
requestId="-1d0e9e07:13921dd320d:-7f89" rolapCube="gosldw_sales"
rolapDimension="Relative Time" rolapHierarchy="Relative Time"
sessionId="-1d0e9e07:13921dd320d:-7f88"><![CDATA[Current period expression failed to execute. XQE-V5-0005 Identifier not found 'gosldw_sales].[Relative Time].[Relative Time].[Month].
<event component="XQE" group="ROLAPCubes.Loader" level="ERROR" thread="63"
timestamp="2012-08-13 17:22:39.889" contextId="3"
requestId="-1d0e9e07:13921dd320d:-7fe9" rolapCube="gosldw_sales"
rolapHierarchy="Relative Time"
sessionId="-1d0e9e07:13921dd320d:-7fe8"><![CDATA[XQE-ROL-0118 Unable to assign an invalid member '([Relative Time].[Relative Time].[All].[_Current_Year].[_Current_Quarter].[_Current_Month])' as the default member of the hierarchy '([Relative Time].[Relative Time])'. The default member is reverted to its default setting.XQE-GEN-0005 Found an internal error.
```

The expression in the currentMeasure is attempting to find the latest month that contains a measure value (July 2013), rather than the last member of the Time hierarchy (December 2013):

```
item(tail(filter(members([gosldw_sales].[Relative Time].[Relative Time].[Month]),tuple(currentMeasure, currentMember([gosldw_sales].[Relative Time].[Relative Time]))) is not null), 1), 0)
```
However, the inclusion of currentMeasure is invalid in the context of calculating the Relative Time Current Period. After the incorrect expression is removed, the correct Relative Time Hierarchy can be seen in the metadata tree. See Figure 10-30.
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publication provides additional information about the topic in this document:

- IBM Cognos Business Intelligence V10.1 Handbook, SG24-7912

You can search for, view, download or order this document and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

  ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- IBM Cognos Business Intelligence 10.2.0 Information Center
  http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/index.jsp
- Dynamic Cubes Installation and Configuration Guide 10.2.0
  http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/nav/0_9
- Dynamic Cubes User Guide 10.2.0
  http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/nav/5_6
- Dynamic Query Analyzer Installation and Configuration Guide 10.2.0
  http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/nav/0_7
- Dynamic Query Analyzer User Guide 10.2.0
  http://pic.dhe.ibm.com/infocenter/cbi/v10r2m0/nav/5_8

Help from IBM

IBM Support and downloads

  ibm.com/support

IBM Global Services

  ibm.com/services
IBM Cognos Dynamic Cubes

IBM Cognos Business Intelligence (BI) provides a proven enterprise BI platform with an open data strategy, providing customers with the ability to leverage data from any source, package it into a business model, and make it available to consumers in various interfaces that are tailored to the task.

IBM Cognos Dynamic Cubes complements the existing Cognos BI capabilities and continues the tradition of an open data model. It focuses on extending the scalability of the IBM Cognos platform to enable speed-of-thought analytics over terabytes of enterprise data, without having to invest in a new data warehouse appliance. This capability adds a new level of query intelligence so you can unleash the power of your enterprise data warehouse.

This IBM Redbooks publication addresses IBM Cognos Business Intelligence V10.2 and specifically, the IBM Cognos Dynamic Cubes capabilities. This book can help you in the following ways:

- Understand core features of the Dynamic Cubes capabilities of IBM Cognos BI V10.2
- Learn by example with practical scenarios using the IBM Cognos samples

This book uses fictional business scenarios to demonstrate the power and capabilities of IBM Cognos Dynamic Cubes. It primarily focuses on the roles of modeler, administrator, and IT architect.

For more information: ibm.com/redbooks

INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION

BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.