IBM DB2 Cube Views

Setup and User's Guide

Version 8

SC18-7298-00

IBM DB2 Cube Views

Setup and User’s Guide

Version 8

SC18-7298-00

Note

Note: Before using this information and the product it supports, read the information in "Notices" on page 185.

First Edition (June 2003)

Contents

About this book

You might see the phrase *DB2 Multidimensional Metadata Management* in the product. That phrase refers to DB2 Cube Views.

This book provides information about the following DB2 Cube Views topics:

- How to get started with DB2 Cube Views
- The OLAP Center (graphical user interface), which you can use to import and export metadata, create cube models and cubes
- Optimization, which helps you improve the performance of OLAP queries
- Metadata objects that can be stored in the DB2 Universal Database™ (DB2®) catalogs
- Application programming interface (API), with which you can create applications that use SQL to access data
- Examples of how to build dimensions and complex measures from metadata objects; these dimensions and measures can be used to model typical business scenarios

Who should read this book

With DB2 Cube Views, you can capture multidimensional metadata from OLAP and database tools and store that metadata in the DB2 catalogs. You can then use that metadata to create OLAP (online analytical processing) cube models and cubes. (Cubes are subsets of cube models.)

You can use Office Connect Analytic Edition to browse the data in cube models and cubes. Office Connect Analytic is a tool that displays OLAP data in spreadsheets.

DB2 Cube Views also provides an Optimization Advisor that will help you to improve the performance of queries that are issued to the cube models by providing SQL scripts to build summary tables.

Read this book if you are a database administrator who works with OLAP metadata and DB2 Universal Database (DB2). You should be familiar with:

- The DB2 catalogs and automatic summary tables (ASTs)
- OLAP concepts, such as cubes, dimensions, hierarchies, and measures
- API concepts and CLI, ODBC, JDBC, XML and DB2 stored procedures

Online information

This section provides Web addresses related to this product.

http://www.ibm.com/redbooks
IBM® Redbooks™ Web site

Search for, view, download, or order hardcopy/CD-ROM versions of the following Redbooks from the Redbooks Web site:

- DB2 UDB's High Function Business Intelligence in e-business SG24-6546-00
- Up and Running with DB2 UDB ESE Partitioning for Performance in an e-Business Intelligence World SG24-6917-00
- Database Performance Tuning on AIX® SG24-5511-01
- DB2 UDB V7.1 Performance Tuning Guide SG24-6012-00

http://www.ibm.com/software/data/
IBM Data Management Web site

http://www.ibm.com/software/data/db2/udb/winos2unix/support/
DB2 Universal Database and DB2 Connect™ Online Support Web site

http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/download.d2w/report
DB2 Maintenance - Fixpaks for DB2 UDB Web site

http://www.ibm.com/software/data/developer
The DB2 Developer Domain Web site

http://www.ibm.com/software/data/db2/library
DB2 Product and Service Technical Library Web site

http://www.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v8pubs.d2w/en_main
DB2 Publications Web site

Chapter 1. Getting started with DB2 Cube Views

This chapter describes the following topics:

- **What is DB2 Cube Views**

  This section describes the benefits of using DB2 Cube Views and the components that are provided with it.
- **Setting up DB2 Cube Views**

  This section describes how to install DB2 Cube Views and how to configure the sample data.
- **Starting the OLAP Center**

  This section describes how to start the OLAP Center so that you can begin managing your metadata.
- **Browsing cubes by using DB2 Office Connect™**

  This section describes how to browse the sample DB2 Cube Views cube in DB2 Office Connect.
- **The db2mdapiclient utility for importing, exporting, and optimizing metadata**

  This section describes the **db2mdapiclient** command and how to manipulate metadata with it.

What is DB2 Cube Views

DB2 Cube Views is an add-on feature of DB2 Universal Database that improves the ability of DB2 to perform OLAP processing. You can use DB2 Cube Views to streamline the deployment and management of OLAP solutions, and improve the performance of OLAP tools and applications. With DB2 Cube Views, you can describe the dimensional structure of your relational tables and create OLAP constructs. You can store the structural information and the OLAP constructs as multidimensional metadata in the DB2 database.

Using the new multidimensional metadata in DB2 provides two major benefits:

Improve the flow of the multidimensional metadata between business intelligence tools and applications

Using the OLAP Center, a graphical interface that is provided, users of warehousing and business intelligence tools can store the multidimensional metadata as part of the DB2 database, and make it available for all tools and applications.

Enhance the performance of OLAP-style queries

Based on the multidimensional metadata, you can create DB2 summary tables using the recommendations from the Optimization Advisor in the OLAP Center. The summary tables contain precalculated data that maps to your OLAP structures. Queries that are generated from the warehousing or business intelligence application with the same OLAP structure will gain performance improvement.

DB2 Cube Views exploits DB2 features such as summary tables, different index schemes, OLAP-style operators, and aggregation functions. The following components are provided:

Multidimensional metadata objects

You can create a set of metadata objects to dimensionally model your relational data and OLAP structures. DB2 Cube Views stores each of the metadata objects that you create in the DB2 catalog. For more information on metadata objects and how they map to relational data, see "About DB2 Cube Views metadata" on page 11.

OLAP Center

With the OLAP Center, you can create, manipulate, import, or export cube models, cubes, and other metadata objects to be used in OLAP tools. The OLAP Center provides easy-to-use wizards and windows to help you work with your metadata. For example, the Optimization Advisor analyzes your metadata and recommends how to build summary tables that store and index aggregated data for your OLAP-style SQL queries. To start the OLAP Center, see "Starting the OLAP Center" on page 6. After you start the OLAP Center, see "Optimizing a cube model" on page 57 to use the Optimization Advisor wizard.

DB2 Office Connect™ Analytic Edition

DB2 Office Connect Analytic Edition is an easy-to-use spreadsheet add-in tool for querying OLAP data in DB2. With DB2 Office Connect Analytic Edition, you can connect to a DB2 database, select a DB2 Cube Views cube, and explore the data in Microsoft® Excel™.

Multidimensional Services

DB2 Cube Views provides a SQL-based and XML-based application programming interface (API) for OLAP tools and application developers. Through CLI, ODBC, or JDBC connections or by using embedded SQL to DB2, applications and tools can use a single stored procedure to create, modify, and retrieve metadata objects.

Sample data

A sample application and database are available to help you learn how to use the product. See Appendix C, “Sample files”, on page 115 for more information on the sample data that is provided.

You can also exchange metadata objects between the DB2 catalog and OLAP tools. To import or export metadata objects to or from the DB2 catalog, utilities called metadata bridges are available for specific OLAP and database tools. See the documentation for your particular OLAP or database tool to find out if a metadata bridge is provided.

Setting up DB2 Cube Views

DB2 Cube Views expands the DB2 catalog to include multidimensional metadata or OLAP metadata. You can use OLAP Center to create, modify, and browse the OLAP metadata. You can also use the DB2 Office Connect Analytics Edition, an add-in for Microsoft Excel, to browse OLAP data.

Installation requirements

For additional installation information, including disk and memory requirements, see the DB2 Cube Views Installation Notes for your operating system. This document is available from the **Installation Prerequisites** button on the Launch Pad after you start the installation program.

Prerequisites:

Before you install the DB2 Cube Views, ensure that you have the required software.

- Server component:

  On Microsoft® Windows®
  : Windows NT® 4 or Windows 2000® 32-bit, Windows XP Professional

  On AIX
  : AIX Version 4.3.3 32-bit, AIX 5L 32-bit, or AIX 5L 64-bit

  On Linux®:
  : Linux Red Hat™ 8 (kernel 2.4.18/ glibc 2.2.93-5) 32-bit, or Linux SuSE 8.0 (kernel 2.4.18/ glibc 2.2.5) 32–bit

    For the latest information on distribution and kernel levels supported by DB2, go to: `http://www.ibm.com/db2/linux/validate`

  On Sun Solaris™ Operating System
  : Solaris 8 32-bit, or Solaris 9 32-bit

- Client component: Windows NT 4, Windows 2000, or Windows XP 32–bit
- DB2 Universal Database Version 8.1 FixPak 2
- Optional: DB2 Office Connect Analytics Edition 4.0

To use DB2 Office Connect Analytics Edition, you need Microsoft Excel 2000 (Office 2000® with service pack 1 or later) or Microsoft Excel XP (Office XP® with service pack 1 or later).

Installing DB2 Cube Views

Procedure:

To install DB2 Cube Views on Windows:

1. Insert the DB2 Cube Views CD-ROM. The installation program starts automatically.
2. On the Launchpad, check the Installation Notes for the latest installation, disk, and memory requirements. Additionally, check the readme.txt file in the root of the DB2 Cube Views CD-ROM for any additional instructions.
3. Click **Install Products** to begin the installation, and follow the prompts.
4. Optional: You can Install DB2 Office Connect Analytics Edition by inserting the DB2 Office Connect CD-ROM and following the instructions in the Installation Notes, which are located in the `install.html` file on the CD.

To set up DB2 Cube Views on AIX, Linux, or Solaris Operating System, follow these steps:

1. Insert the DB2 Cube Views CD-ROM.
2. Switch to the directory for your UNIX® operating system and launch the `db2setup` file.
3. On the Launchpad, check the Installation Notes for the latest installation, disk, and memory requirements. Additionally, check the readme.txt file in the root of the DB2 Cube Views CD-ROM for any additional instructions.

Setting up a database for DB2 Cube Views

To prepare a database, you must set up the database to be used with DB2 Cube Views. This includes:

- Registering the DB2 Cube Views stored procedure with the database
- Creating metadata catalog tables for DB2 Cube Views.

When you first log on to a database that is not configured for DB2 Cube Views by using the OLAP Center, the OLAP Center sets up the database for you. Alternatively, you can set up the database using the `db2mdapi.sql` script file. Do not alter the `db2mdapi.sql` file or your results will be unpredictable.

Procedure:

To set up a database using the `db2mdapi.sql` script file:

1. Open the DB2 Command window and connect to your database.

2. Change to the `SQLLIB\misc` directory and enter the following command in the DB2 Command window:

   ```
   db2 -tvf db2mdapi.sql
   ```

Setting up the MDSAMPLE sample database

DB2 Cube Views provides sample data that you can use to create a sample database called MDSAMPLE. The sample data includes a set of tables containing data about a fictional company that sells beverages. A set of OLAP metadata objects that describe the sample data tables are also included.

Procedure:

To create and populate the sample MDSAMPLE database, open the DB2 Command window and enter the following commands:

1. Create a sample database called MDSAMPLE:

   ```
   db2 create db mdsample
   ```

2. Connect to the database:

   ```
   db2 connect to mdsample
   ```

3. Run the `db2mdapi.sql` script to set up the database for DB2 Cube Views. Change to the `SQLLIB\misc` directory and enter the following command:

   ```
   db2 -tvf db2mdapi.sql
   ```

4. Change to the `SQLLIB\samples\olap\mdsample` directory. Then, create the MDSAMPLE tables by entering the following DB2 command:

   ```
   db2 -tvf MDSampleTables.sql
   ```

You can create the DB2 Cube Views multidimensional metadata by importing the definitions from an XML file exported from a business intelligence application.

For example, the following procedure populates the DB2 Cube Views catalog tables with a complete description of the MDSAMPLE database.

Procedure:

To import the MDSAMPLE metadata:

1. Start the OLAP Center and connect to the MDSAMPLE database.
2. Click **OLAP Center —> Import**.
3. Browse for the `MDSampleMetadata.xml` file in the `SQLLIB/samples/olap/mdsample` directory. Click **Next**.
4. Browse the metadata in the OLAP Center. For information about using the OLAP Center, see the OLAP Center online help.

Starting the OLAP Center

Procedure:

To start the OLAP Center:

1. Click **Start –> Programs –> IBM DB2 –> Business Intelligence Tools –> OLAP Center**. A database connection window opens.
2. In the database connection window, enter the following information:
   - In the **Database name** field, enter or select the name of the database to which you want to connect.
   - In the **User name** field, enter the user ID for the database you specified.
   - In the **Password** field, enter the password for the user ID you specified.
   - Click **OK**.

Refreshing the objects in the OLAP Center:

The OLAP Center shows a snapshot in time of the OLAP metadata objects in the database. Although DB2 Cube Views always ensures the integrity of the metadata objects that it manages, the contents of the OLAP Center window are not automatically updated when objects are created in the database by another OLAP Center user or by an API application. If another user or an API application changes the metadata, you can select **View —> Refresh** to see the new state of the database. Multiple users who work on the same metadata objects at the same time can experience errors because they might not see the most recent data in the database. Multiple users should not work on the same set of objects at the same time.

Browsing cubes by using DB2 Office Connect

You can use DB2 Office Connect to browse the data in DB2 Cube Views cubes.

Procedures:

To browse the sample DB2 Cube Views cube in DB2 Office Connect:

1. Create an ODBC database for the MDSAMPLE database using the ODBC Data Source Administration tool in Microsoft Windows.
2. Start Excel.
3. Open the Project Manager. Click **DB2 Office Connect —> Project Manager**. Use the Project Manager to connect to the MDSAMPLE database and open the sample cube.
4. In the Cube Wizard - Introduction window, click **Next**. The Cube Wizard - Select Cube window opens.

5. Expand the SalesModel cube model, then select the Sales cube. Click **Finish**. There are new items under Data sources in the Project Manager window. Expand each item to see the MDSAMPLE data.
6. Drag and drop the Sales cube to the Worksheets folder to see the cube data in an Excel spreadsheet.

The db2mdapiclient utility for importing, exporting, and optimizing metadata

The db2mdapiclient utility is provided as sample source code for coding an application for Multidimensional Services.

Overview of the db2mdapiclient utility

This utility is a thin wrapper to the Multidimensional Services stored procedure interface. The utility is provided as sample source code to show how to code an application against the API. The source code is located in `\SQLLIB\samples\olap\client\db2mdapiclient.cpp`.

You can use the db2mdapiclient utility to perform any of the operations that are supported by the DB2 Cube Views stored procedure, MD_MESSAGE(), which are described in the following table:

Table 1. Multidimensional Services operations that the db2mdapiclient utility can perform

Task	Operation
Export metadata to a file	DESCRIBE
Import metadata from a file	CREATE or IMPORT
Change existing metadata	ALTER or RENAME
Delete existing metadata	DROP
Verify that the existing metadata is valid	VALIDATE

The db2mdapiclient utility uses files to hold the XML that is passed to and received from the MD_MESSAGE() stored procedure.

For importing, the db2mdapiclient utility typically uses an XML file that is produced by a DB2 Cube Views bridge or that was exported from the OLAP Center. For exporting, the db2mdapiclient utility produces an XML file that a DB2 Cube Views bridge utility can use to add metadata to a database or OLAP tool. The character encoding used for the input and output XML files is important. For more information on character encoding, see Appendix B, “Code page support”, on page 111.

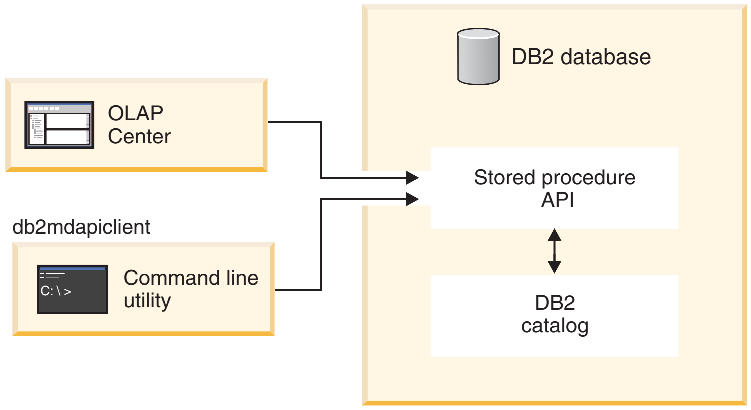


Figure 1. The db2mdapiclient utility and the OLAP Center transfer metadata through Multidimensional Services

The db2mdapiclient command — manipulating metadata

To see a list of parameters for the db2mdapiclient command, you can enter `db2mdapiclient.exe` at a command line. The typical syntax for the **db2mdapiclient** command is:

```
 db2mdapiclient -d dbname -u user -p password -i request.xml -o response.xml
-m inputmetadata.xml -n outputmetadata.xml
```

The minimal syntax for the **db2mdapiclient** command is:

```
db2mdapiclient -d dbname -i request.xml -o response.xml
```

You can use the command line flags to indicate the role of the specified files and the size of the files that can be processed. Each time the db2mdapiclient utility is invoked, it uses a minimum of two or a maximum of four of the following files:

`request.xml`
: This required input file contains the operation to perform.

`response.xml`
: This required output file contains the response XML from the MD_MESSAGE() stored procedure. The third argument in the MD_MESSAGE() stored procedure returns the response XML.

`inputmetadata.xml`
: This optional input file contains DB2 Cube Views metadata.

`outputmetadata.xml`
: This optional output file contains the response metadata XML, if applicable, from the second argument of the MD_MESSAGE() stored procedure.

The following diagram shows how the MD_MESSAGE() stored procedure is associated with the two input and two output files:

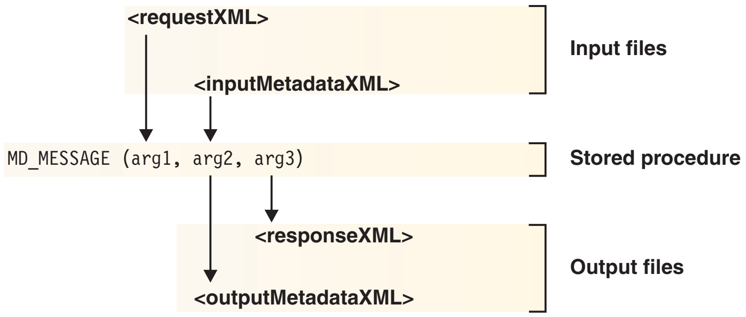


Figure 2. How the stored procedure handles the two input and output files from the db2mdapiclient utility

For example, to import DB2 Cube Views metadata for the MDSAMPLE database, change to the `..\SQLLIB\samples\olap\xml\input` directory and enter the following command:

```
db2mdapiclient -d MDSAMPLE -u db2admin -p mypasswrd -i create.xml
  -o myresponse.xml -m MDSampleMetadata.xml
```

For a description of the sample files that are provided, see Appendix C, “Sample files”, on page 115. For more information on the Multidimensional Services operations, see the Appendix A, “API Reference”, on page 81.

Chapter 2. Multidimensional metadata objects

This chapter describes the following topics:
- Overview of DB2 Cube Views metadata
- General metadata properties
- Cube models
- Facts objects
- Dimensions
- Hierarchies
- Measures
- Attributes
- Attribute relationships
- Joins
- Cubes
- Cube facts
- Cube dimensions
- Cube hierarchies
- Metadata object rules

About DB2 Cube Views metadata

DB2 Cube Views metadata objects describe relational tables as OLAP structures, but these metadata objects are different from traditional OLAP objects. Metadata objects store metadata about the data in the base tables, they describe where pertinent data is located, and they describe relationships within the base data. DB2 Cube Views manages the following 12 metadata objects and stores them in the DB2 catalog:
- Cube models
- Facts objects
- Dimensions
- Hierarchies
- Measures
- Attributes
- Attribute relationships
- Joins
- Cubes

- Cube facts
- Cube dimensions
- Cube hierarchies

Metadata provides a new perspective from which to understand your data. DB2 Cube Views extends the DB2 catalog so that in addition to storing information about tables and columns, the DB2 catalog contains information about how the tables and columns relate to OLAP objects and the relationships between those objects.

Some metadata objects act as a base to directly access relational data by aggregating data or directly corresponding to particular columns in relational tables. Other objects describe relationships between the base metadata objects and link these base metadata objects. All of the objects can be grouped by their relationships to each other into a multidimensional metadata object called a cube model. Essentially, a cube model represents a particular grouping and configuration of relational tables.

Metadata objects that map to relational tables

A cube model can be constructed in many ways, but is often built to represent a relational star schema or snowflake schema. (A star schema has a facts table at the center and one or more dimension tables joined to the fact table, and a snowflake schema is an extension of a star schema such that one or more dimensions are defined by multiple tables.) A cube model that is based on a simple star schema is built around a central facts object. The facts object contains a set of measures that describe how to aggregate data from the fact table across dimensions. Measures describe data calculations from columns in a relational table and are joined to create the facts object. Figure 3 on page 13 shows how measures and a facts object relate to relational data.

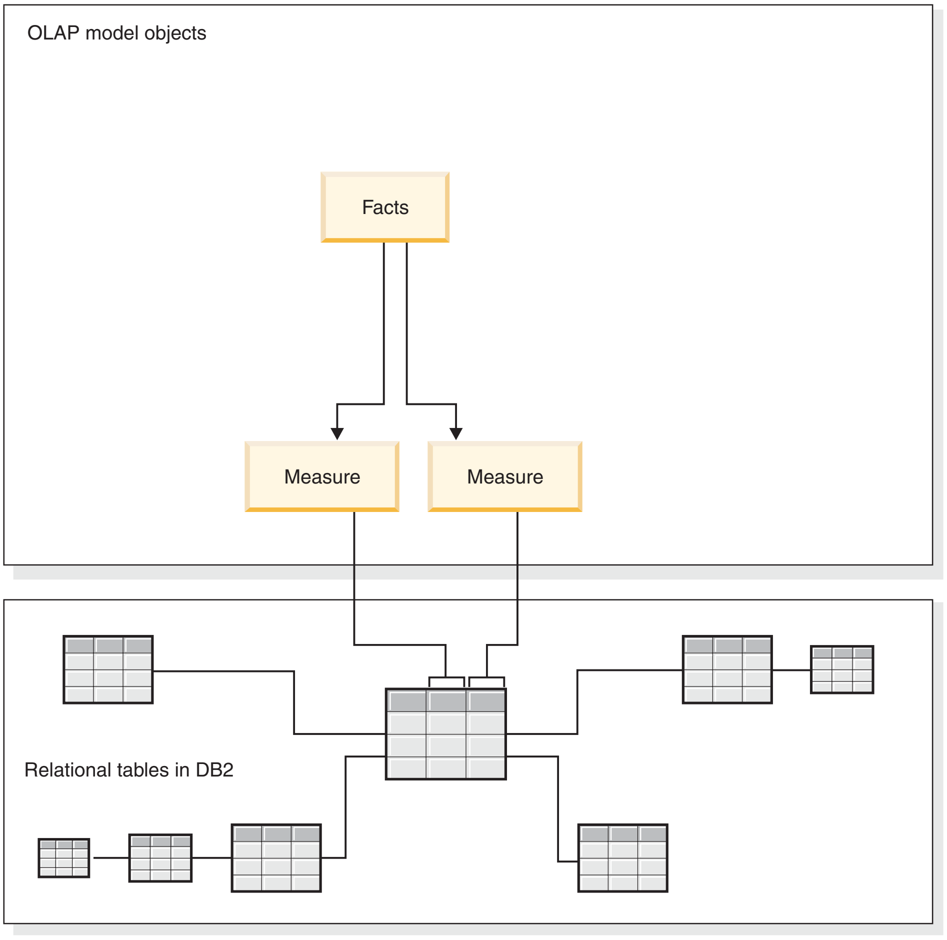


Figure 3. How a facts object and measures relate to relational data

Dimensions are connected to the facts object in a cube model like the dimension tables are connected to the fact table in a star schema. Columns of data from relational tables are represented by attributes that are joined to make a dimension.

Figure 4 on page 14 shows how dimensions are built from relational tables. Hierarchies store information about how the attributes within a dimension are related to each other and structured. A hierarchy provides a way to calculate and navigate the dimension. Each dimension has a corresponding hierarchy with levels that are defined for each member attribute. In a cube model, each dimension can have multiple hierarchies.

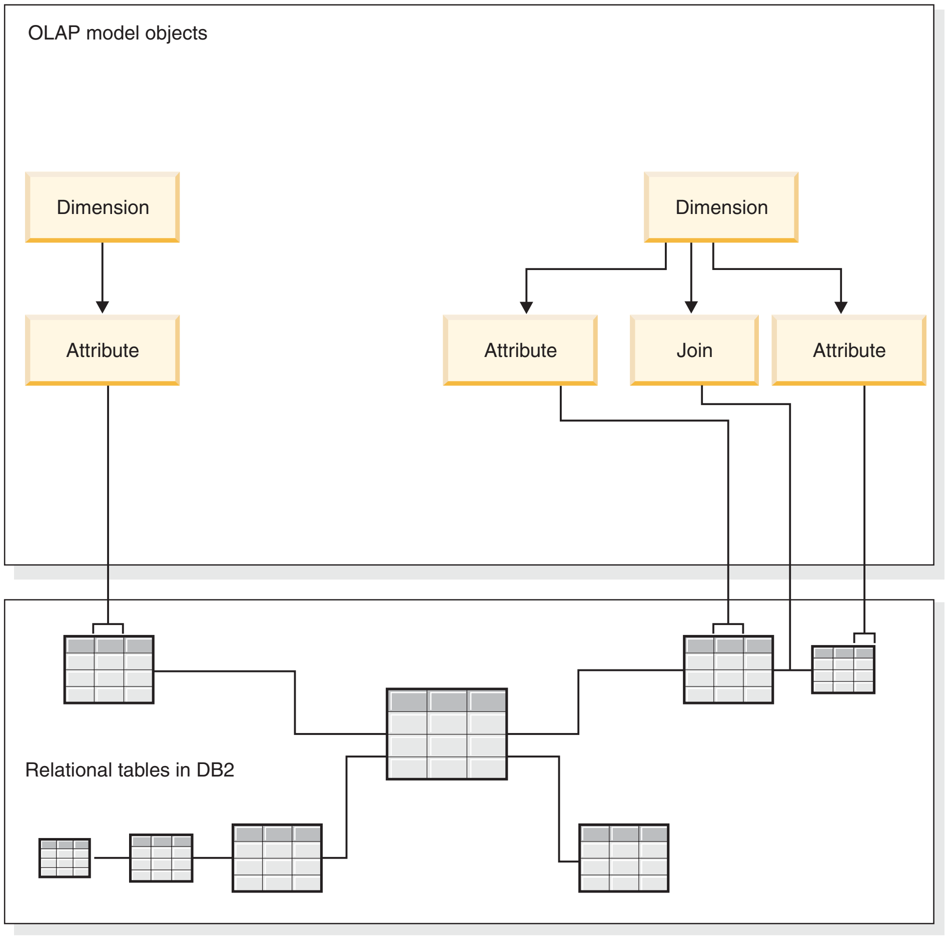


Figure 4. How dimensions are built from relational tables

All of the dimensions are connected to a central facts object to create a cube model based on a star schema. Joins can connect tables to create a facts object or a dimension. In a cube model, joins can connect facts objects to dimensions. The dimensions have information about all of their corresponding hierarchies, attributes, and related joins. Facts objects have information about all of their component measures, attributes, and related joins. Figure 5 on page 15 shows how the metadata objects fit together in a cube model and map to a relational star schema.

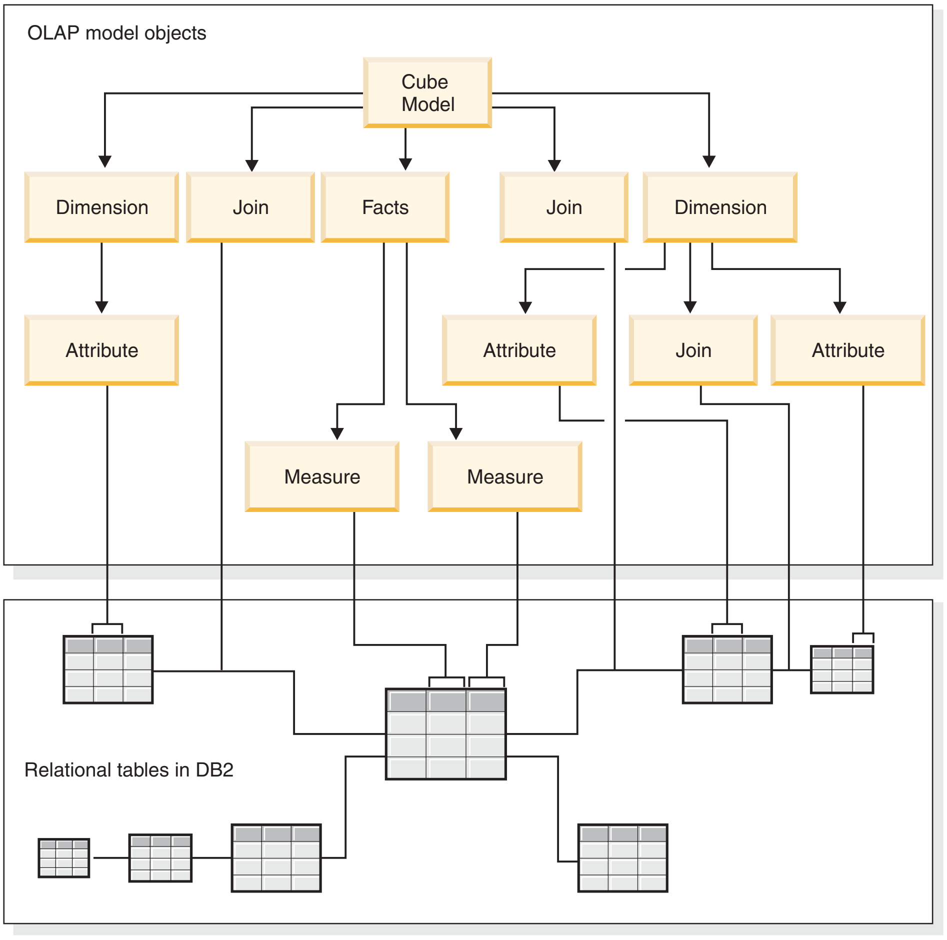


Figure 5. How metadata objects fit together and map to a relational star schema

You can reuse the components of a cube model to create more precise cubes for specific applications. A cube is the most precise metadata object and is the closest object to an OLAP conceptual cube. A cube is a specific instance or subset of a cube model. A cube has a specific set of similar but more restrictive metadata objects derived from the parent cube model: cube dimensions, cube hierarchies, and cube facts. A cube can have only one cube hierarchy defined for each cube dimension, but a dimension can have many hierarchies that are defined for the cube model. Because of this structural difference between a cube and a cube model, you can retrieve most cubes with a single SQL statement.

Example of a cube model that maps to relational tables

A cube model is often built to represent a relational star schema or snowflake schema. Figure 6 on page 16 shows a star schema with a Sales fact table, and

Time, Product and Region dimension tables. The primary key in each dimension table is joined to the corresponding foreign key in the Sales fact table. For example, Time.TimeID = Sales.TimeID, Product.ProductID = Sales.ProductID, and Region.RegionID = Sales.RegionID.

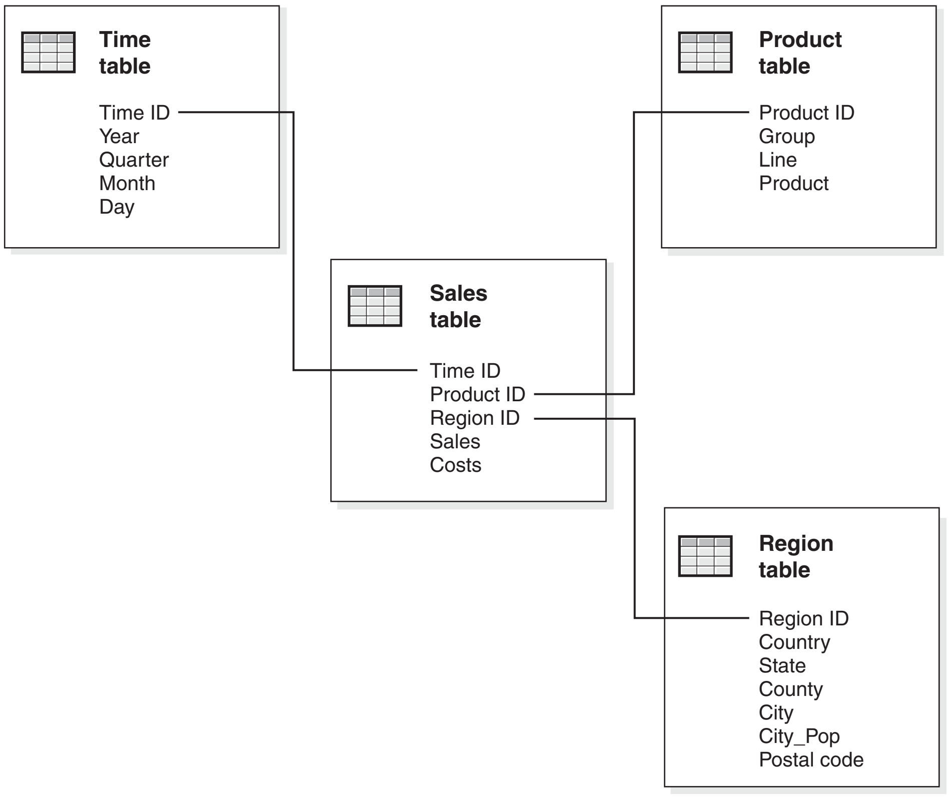


Figure 6. Example of a star schema

The cube model based on this star schema is built around the Sales facts object that describes aggregated relational data from the Sales fact table. Measures describe how to calculate data from columns in the Sales table. The facts object also includes attributes that correspond to the foreign keys in the fact table that are used to join the dimensions to the facts object. In this example, the Sales facts object has two measures: Sales and Costs; it has three attributes: Facts_TimeID, Facts_ProductID, and Facts_RegionID.

Dimensions are connected to the facts object in a cube model like the dimension tables are connected to the fact table in a star schema. Columns of data from relational tables are represented by attribute objects referenced by the dimension. In this example, the Time dimension references five attributes: TimeID, Year, Quarter, Month, and Day. The Product dimension references four attributes: ProductID, Group, Line, and Product. The Region dimension references seven attributes: RegionID, Country, State, County, City, City_Pop, and Postal_code.

A join is created to connect each dimension to the facts object. The three joins in this example are Time, Product, and Region. Figure 7 shows the cube model that is described in this example.

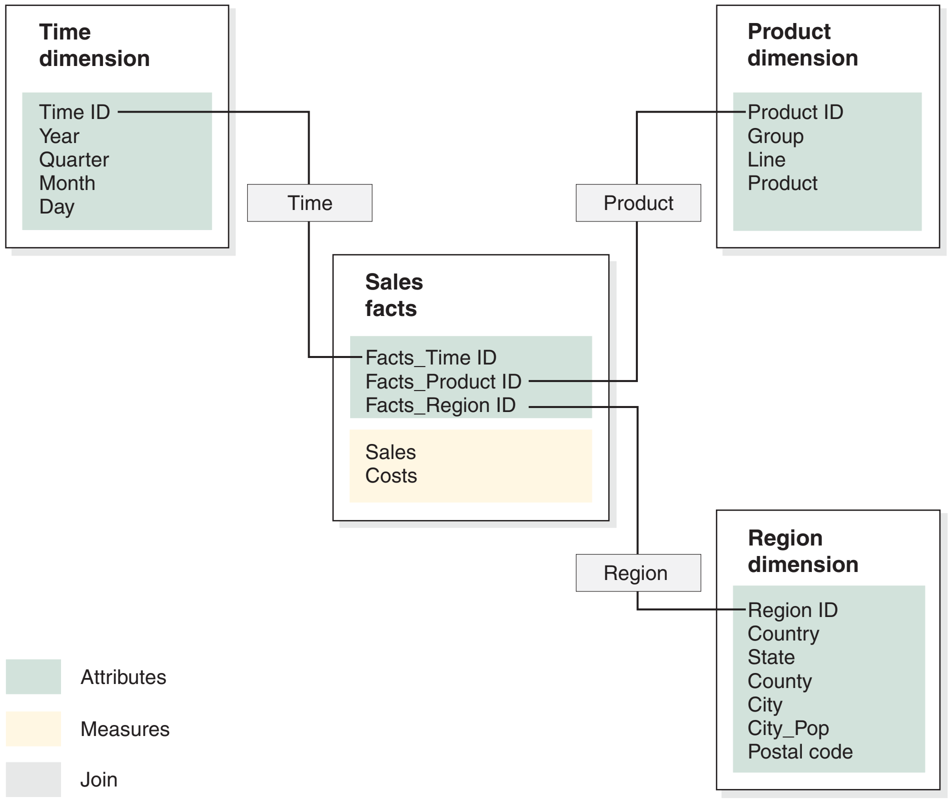


Figure 7. Cube model based on star schema example

Hierarchies store information about how the attributes within a dimension are related to each other and structured. As a metadata object, a hierarchy provides a way to calculate and navigate the dimension. Each dimension has a corresponding hierarchy with each member attribute in a logical level. In a cube model, each dimension can have multiple hierarchies.

In this example, the Region dimension has two hierarchies: the Region overview hierarchy and the Region detail hierarchy, as shown in Figure 8 on page 18. The Region overview hierarchy uses a subset of the attributes in the Region dimension: Country, State, and County. The Region detail hierarchy uses all of the attributes in the Region dimension. City_Pop is an attribute that is used in an attribute relationship with City. The Region detail hierarchy has five levels: Country, State, County, City and Postal Code. You can include City_Pop as a related attribute in the hierarchy because it adds extra information to the hierarchy attribute. For example, when you query data for a specific city, you can also include data about the city's population.

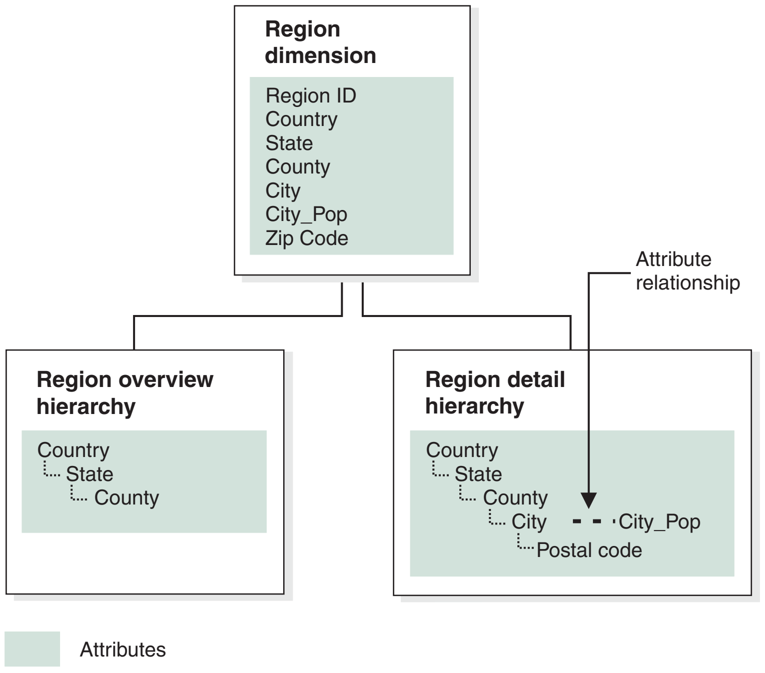


Figure 8. Region dimension with two hierarchies

You can also build one or more cubes for the cube model. For this example, the cube model has one cube, shown in Figure 9 on page 19. The cube facts object references all of the Sales and Costs measures from the cube model facts object. The cube has one cube dimension that references the three dimensions in the cube model. The Region cube dimension has a Region cube hierarchy with a subset of the Country, State, and County attributes. The Time cube dimension has a Time cube hierarchy with a subset of the Year, Quarter, and Month attributes. The Product cube dimension has a Product cube hierarchy with a subset of the Group, Line, and Product attributes. The cube has only one cube hierarchy that is defined per cube dimension. (A cube can have only one cube hierarchy per cube dimension.)

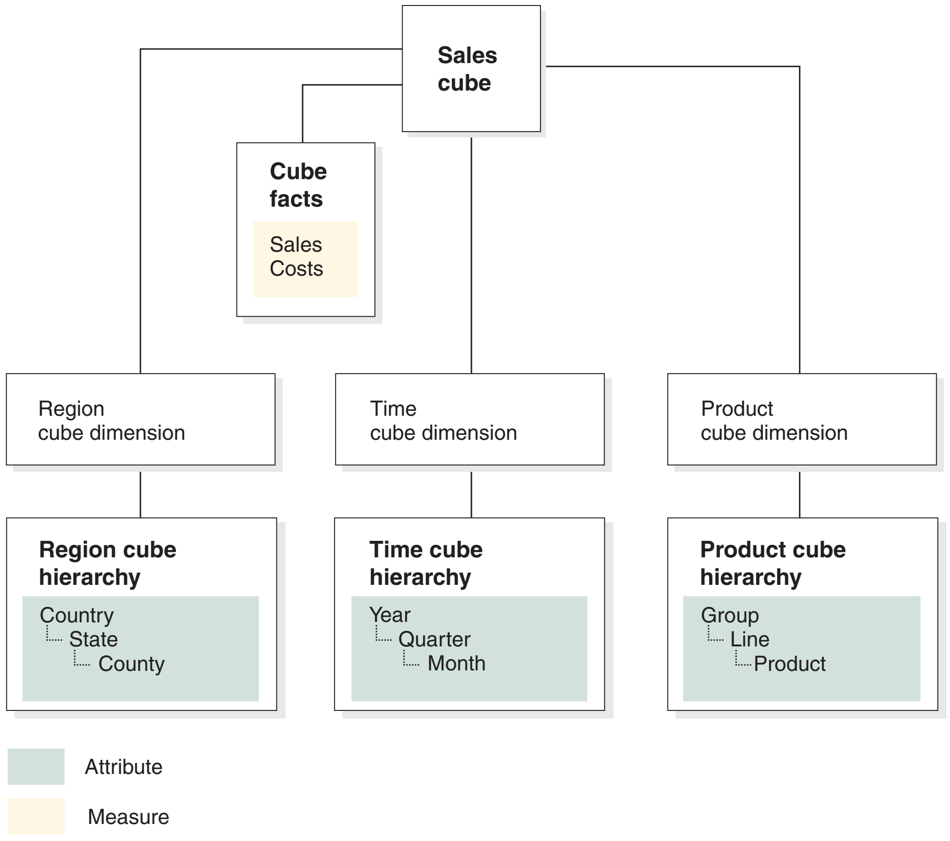


Figure 9. Cube based on cube model example

General metadata properties

Each metadata object has a set of general properties and object-specific properties. The general properties are used to identify the object instances, to describe the usage or role of the object instances, and to track object instance changes. The objects are named by using a schema in the same way that other DB2 objects are named. If you do not want to use the default user name schema for an object, you need to fully qualify the object with the schema name that you want.

The following table describes the general properties that exist for all metadata objects.

Table 2. General metadata object properties

Property	Description
Name	Name of the object.
Schema	Schema that owns the object.
Business name	Name presented to the user. This name can be used in graphic user interfaces as a name more meaningful to the user.
Comments	Textual description or comment on the nature or usage of the object.
Create time	Time that the object was created.
Creator	User (schema) that defined the object.
Modify time	Time the object was last modified.
Modifier	User (schema) that performed the modification.

In addition to a common set of general properties, each metadata object has a set of specific properties. These specific properties describe the components and qualities that define the metadata object. For information about properties that are specific to each metadata object, see the topic for that object.

Metadata object naming conventions

DB2 provides two different naming conventions to name objects: ordinary and delimited. For metadata objects, the delimited convention is used when naming objects and referring to DB2 tables and columns. The delimited convention allows mixed case names, spaces, and special characters, such as national language characters. The complete set of characters is determined by the code page of the database in which the objects are stored.

The following conventions apply to the metadata objects:

Table 3. Naming conventions for DB2 Cube Views metadata objects

Object	Convention
Schema	• Length: 1-30 bytes • Restricted names: schema names must not be *SESSION* or begin with *SYS*. Only the uppercase names are restricted.
Name of object	• Length: 1-128 bytes • No other restrictions.
Business name of object	• Length: 1-128 bytes • No other restrictions

Table 3. Naming conventions for DB2 Cube Views metadata objects (continued)

Object	Convention
Comments for objects	• Length: 0-254 bytes • No other restrictions
Table schema that is used in referencing columns	• Length: 1 to 128 bytes • No other restrictions
Table name that is used in referencing columns	• Length: 1 to 128 bytes • No other restrictions
Column Name that is used in referencing columns	• Length: 1-128 bytes • No other restrictions

Cube models

The DB2 Cube Views cube model is a representation of a logical star schema or snowflake schema. The cube model is a grouping of relevant dimension objects around a central facts object. Each dimension can have multiple hierarchies. The structural information about how to join the tables that are used by the facts and dimension objects is referenced by the cube model. Also stored in the cube model is enough information to retrieve OLAP data. Other reporting and OLAP tools that understand the cube model and can display multiple views of a specific dimension can benefit from its use.

Cube models define a complex set of relationships and can be used to selectively expose relevant facts and dimensions to an application. Each join object that connects a dimension to the central facts object is stored with the corresponding dimension as a set. Subsets of cube model components can be used by many cubes for different analysis purposes.

You can create an empty cube model in the OLAP Center by using the Cube Model wizard. An empty cube model does not have a facts object or any dimensions. With the wizards in the OLAP Center, you can complete the cube model by creating the facts object and one or more dimensions. You can also create a complete cube model using the Quick Start wizard. DB2 Cube Views will validate your cube model by ensuring that you have the mandatory components:

- A facts object
- At least one dimension
- At least one corresponding hierarchy for each dimension
- Joins between the existing facts object and dimensions
- Attributes (if any) that reference valid tables

For more information about creating cube models, see the online help in the OLAP Center.

The properties that are specific to cube models are described in the following table.

Table 4. Cube model properties

Property	Description
Facts object	Facts object that is used in the cube model
Set of (dimension, join)	Dimensions that are used in the cube model and their corresponding joins

Facts objects

The facts object groups related the measures that are interesting to a particular application. Multiple relational fact tables can be joined on specific attributes to map additional related measures. The facts object stores information about the attributes that are used in fact-to-dimension joins, and the attributes and joins that are used to map the additional measures across multiple database tables. Therefore, in addition to a set of measures, a facts object stores a set of attributes and a set of joins. A facts object is used in a cube model as the center of a star schema.

You can use the Facts wizard in the OLAP Center to create a facts object. In the Facts wizard you specify one or more fact tables and any necessary joins, measures, and aggregations for the measures.

The specific properties of a facts object are described in the following table.

Table 5. Facts object properties

Property	Description
Set of measures	Set of all related measures in the facts object
Set of attributes	Set of all attributes that are used in the facts object
Set of joins	Set of all joins that are needed to join all of the specified measures and attributes

Dimensions

Dimensions provide a way to categorize a set of related attributes that together describe one aspect of a measure. Dimensions are used in cube models to organize the data in the facts object according to logical categories such as Region, Product, or Time. Related attributes and the joins that are required to group these attributes are defined in the properties of the dimension.

Dimensions reference one or more hierarchies. Hierarchies describe the relationship and structure of the referenced attributes and can be used in the navigation and calculation of the dimension.

Dimensions also have a type that describes if the dimension is time-oriented. For example, a dimension called Time might contain attributes like Year, Quarter, and Month and is a Time type. Another dimension called Region might contain attributes like Country, State, City, and Population and is a Regular type. Type information can be used by applications to intelligently and appropriately perform time related functions.

You can use the Dimension wizard in the OLAP Center to create a new dimension in the context of a cube model or without a reference to a cube model. You can also add an existing dimension to a cube model by using the Add Dimension wizard.

The specific properties of dimensions are described in the following table.

Table 6. Dimension properties

Property	Description
Set of attributes	Set of all attributes that are used in the dimension.
Set of joins	Set of all joins that are required to join all of the specified attributes. Only the joins that are required to join the dimension tables are specified here.
Set of hierarchies	Set of hierarchies that apply to the dimension.
Type	Dimension type that can be REGULAR or TIME

Hierarchies

A hierarchy defines relationships among a set of one or more attributes in the dimension of a cube model. These relationships provide a navigational and computational means of traversing dimensions. Multiple hierarchies can be defined for a dimension of a cube model. Hierarchies also reference a set of attribute relationships that link attributes in the hierarchy to other related attributes. The attributes that are directly related by an attribute relationship can be queried as part of the hierarchy. For example, a hierarchy for a Region dimension can have a City attribute, and an attribute relationship can link City to a CityPopulation attribute. This hierarchy can include CityPopulation information in a query that includes City.

The hierarchy type describes the relationship among the attributes within the hierarchy. The following four hierarchy types are supported:

Balanced
A hierarchy with meaningful levels and branches that have a consistent depth. Each attribute's logical parent is in the level directly above it. A balanced hierarchy can represent time where the meaning and depth of each level, such as Year, Quarter and Month, is consistent. They are consistent because each level represents the same type of information, and each level is logically equivalent.Figure 10 shows an example of a balanced time hierarchy.

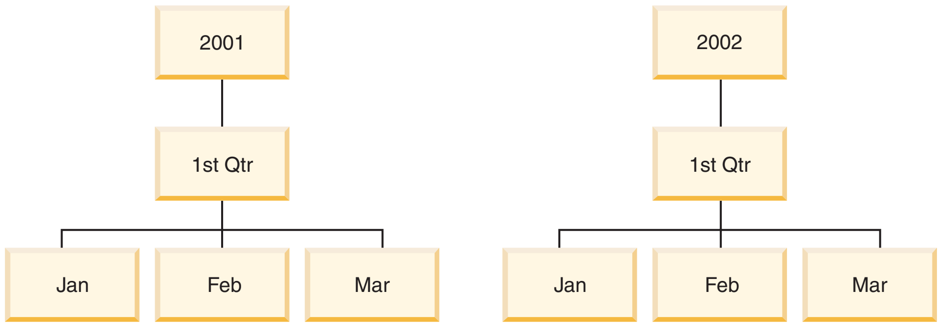


Figure 10. Example of a balanced hierarchy

Unbalanced
A hierarchy with levels that have a consistent parent-child relationship but have a logically inconsistent levels. The hierarchy branches also have inconsistent depths. An unbalanced hierarchy can represent an organization chart. For example, Figure 11 on page 25 shows a CEO on the top level of the hierarchy and at least two of the people that might branch off below including the chief operating officer and the executive secretary. The chief operating officer has more people branching off also, but the executive secretary does not. The parent-child relationships on both branches of the hierarchy are consistent. However, the levels of both branches are not logical equivalents. For example, an executive secretary is not the logical equivalent of a chief operating officer.

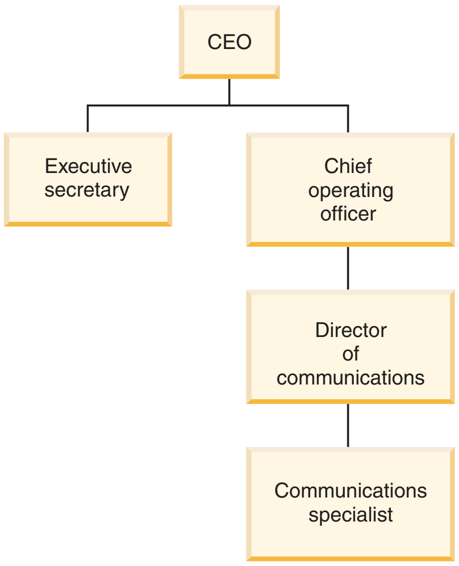


Figure 11. Example of an unbalanced hierarchy

Ragged

A hierarchy in which each level has a consistent meaning, but the branches have inconsistent depths because at least one member attribute in a branch level is unpopulated. A ragged hierarchy can represent a geographic hierarchy in which the meaning of each level such as city or country is used consistently, but the depth of the hierarchy varies. Figure 12 on page 26 shows a geographic hierarchy that has Continent, Country, Province/State, and City levels defined. One branch has North America as the Continent, United States as the Country, California as the Province/State, and San Francisco as the City. However, the hierarchy becomes ragged when one member does not have an entry at all of the levels. For example, another branch has Europe as the Continent, Greece as the Country, and Athens as the City, but has no entry for the Province/State level because this level is not applicable to Greece. In this example, the Greece and United States branches descend to different depths, creating a ragged hierarchy.

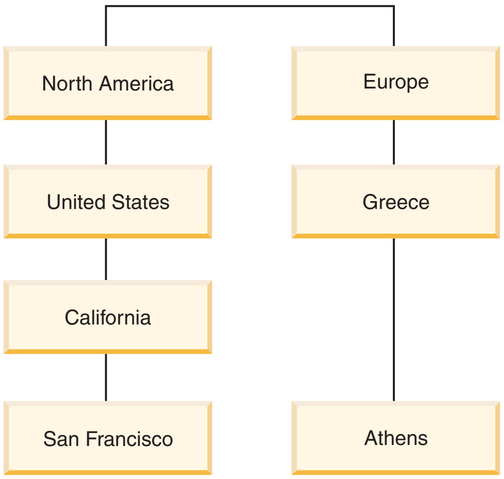


Figure 12. Example of a ragged hierarchy

Network

A hierarchy in which the order of levels is not specified, but in which levels do have semantic meaning. For example, Figure 13 shows a network hierarchy that describes product attributes such as Color, Size, and PackageType. Because the attribute levels do not have an inherent parent-child relationship, the order of the levels is not important. A widget company might have member entries like white for Color, small for Size, and shrink wrap for PackageType. A second member entry might be red for Color, large for Size, and box for PackageType.

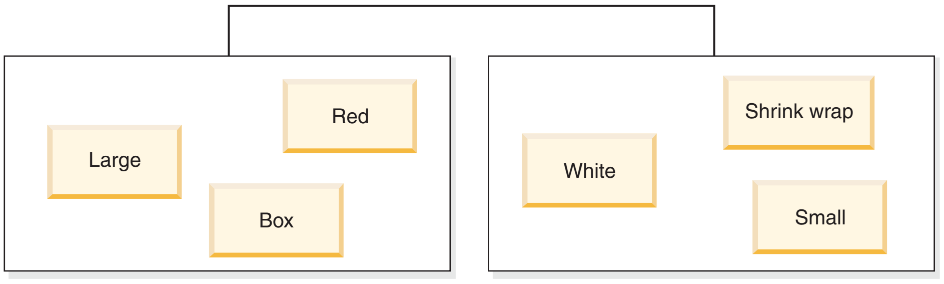


Figure 13. Example of a network hierarchy

A hierarchy also specifies the deployment mechanisms for the hierarchy. A deployment mechanism defines how to interpret the attributes of a hierarchy. The following two deployment mechanisms are supported:

Standard

Uses the level definitions of the hierarchy, where each attribute in the hierarchy defines one level. For example, a balanced hierarchy for a Time dimension would be organized by each defined level including Year, Quarter, and Month. Standard deployment can be used with all four hierarchy types. Table 7 shows how some of the balanced hierarchy attributes for a Time dimension are organized using a standard deployment

Table 7. Standard deployment of a balanced hierarchy for a Time dimension

Year	Quarter	Month
2001	1st Qtr	Jan
2001	1st Qtr	Feb
2001	1st Qtr	Mar
2002	1st Qtr	Jan
2002	1st Qtr	Feb
2002	1st Qtr	Mar

Recursive

Uses the inherent parent—child relationships between the attributes of the hierarchy. An unbalanced hierarchy using a recursive deployment is represented as parent-child attribute pairs. For example, Table 8 shows the attribute pairs for the unbalanced hierarchy describing an organization chart shown in Figure 11 on page 25. The parent—child attribute pairs include: chief executive officer and executive secretary, chief executive officer and chief operating officer, chief operating officer and director of communications, director of communications and communications specialist. Recursive deployment can only be used with an unbalanced hierarchy.

Table 8. Recursive deployment of an unbalanced hierarchy for an Organization dimension

Parent attribute	Child attribute
Chief executive officer	Executive secretary
Chief executive officer	Chief operating officer
Chief operating officer	Director of communications
Director of communications	Communications specialist

You can create a hierarchy in the OLAP Center using the Hierarchy wizard. You can define a hierarchy for a dimension after you have created the dimension.

The properties of a hierarchy object are described in the following table.

Table 9. Hierarchy properties

Property	Description
List of attributes	Ordered list of attributes from the top to the bottom of a hierarchy. In the case of a recursive hierarchy, two attributes are used as parent and child.
Set of attribute relationships	Set of all attribute relationships that link hierarchy attributes to other attributes.
Type	Hierarchy type can be BALANCED, UNBALANCED, RAGGED, NETWORK
Deployment	Hierarchy deployment can be STANDARD, RECURSIVE

Measures

Measures define a measurement entity and is used in facts objects. Measures become meaningful within the context of a dimension. For example, a revenue of 300 is not meaningful by itself. When you put a revenue measure in the context of a dimension, such as Region, the measure becomes meaningful: the revenue for New York is 300. Common examples of measures are Revenue, Cost, and Profit.

A measure is defined by a combination of two properties: an SQL expression list and an aggregation list. Measures are defined by the aggregation of SQL expressions. Table columns, attributes, and measures are mapped to a template to build SQL expressions. The resulting SQL expressions are then used as input for the first aggregation function of the measure. If a measure has more than one aggregation, the aggregation functions are performed in the order that they are listed, with each subsequent aggregation taking the previous aggregation's result as its input. If the SQL expression for the measure only references other measures, the aggregation function is optional. The aggregation function is optional because the referenced measures provide the aggregation.

An SQL expression for the measure is created by the combination of two properties: a template and a list of columns, attributes, and measures. The template uses a token notation where `{$$n}` is the token and `n` references a specific column, attribute, or measure from the list. The list of columns, attributes, and measures is ordered and the position of a column, attribute or measure in the list corresponds to the token `n` value.

SQL expressions are used as input to the first aggregation. Each aggregation specifies a function that is applied to a corresponding list of dimensions. The aggregation function can be any aggregation function that is supported by the underlying database. The following aggregation functions are supported in DB2 Cube Views:

- AVG
- CORRELATION
- COUNT
- COUNT_BIG
- COVARIANCE
- MAX
- MIN
- REGRESSION functions (all 9 types)
- STTDEV
- SUM
- VARIANCE

Each dimension can be aggregated only once by the measure object. A measure must have one aggregation with an empty list of dimensions, and any other aggregations must each have an explicit list of dimensions. The aggregation for an empty list of dimensions is applied to all dimensions in the cube model that are not specifically being used by another aggregation.

An example of a simple measure that directly maps to a column is Revenue. The Revenue measure can be created for a cube model with three dimensions: Product, Market, and Time. Revenue has an SQL expression template `template = "{$$1}"` that represents a simple mapping to the column specified in the one-item list of columns, attributes, and measures where `list = "Column Fact.Rev"`. The aggregation list is `(SUM, <NULL>)` where `SUM` is the aggregation function, and `<NULL>` is an empty list of dimensions. The SQL expression is used as input for the `SUM` aggregation function, which results in the SQL expression: `SUM(Fact.Rev)`.

A more complicated measure, Profit, might have an SQL expression template `template = "{$$1} - {$$2}"`, where the list of attributes, columns, and measures is `list = "Measure Revenue, Column Fact.Cost"`. Replacing the tokens with the correct references, the SQL expression becomes `"Revenue - Fact.Cost"`. Expanding the revenue measure reference to its column reference, the SQL expression becomes: `"Fact.Rev - Fact.Cost"`. The Profit measure's aggregation list is : `(SUM, <NULL>)`. Using the profit SQL expression as input for the `SUM` aggregation function, the Profit measure's SQL is: `SUM(Fact.Rev - Fact.Cost)`.

If the measure has an aggregation function, such as CORRELATION, that requires two or more parameters, the measure will have two or more SQL expressions.

Measures also have a data type that is based on SQL data types. DB2 Cube Views automatically determines the data type of a measure. Each name, when fully qualified by a schema, must be unique among measures and attributes.

The OLAP Center hides much of the complexity of the metadata object definition. In the OLAP Center, you do not have to explicitly define the measure's list of SQL expression or aggregation list. If you want to create a measure that directly maps to a column, attribute, or other measure, you select the source when you create the measure in the Facts wizard or the Facts Properties window. If you want to create a calculated measure, you can use the SQL Expression Builder window to create the source expression. The SQL Expression Builder provides lists of available columns, attributes, and measures, operators, and functions and constants. In the Measure Properties window you can view the data type of the source data for the measure, and the data type of the measure after the source data has been aggregated.

The following table describes the specific properties that define a measure. The OLAP Center defines each of these for you when you create a measure.

Table 10. Measure properties

Property	**Description**
List of SQL expressions (template, [(list of columns, attributes and measures])	List of SQL expressions used as input for the first aggregation function of the measure. Each SQL expression has a template and an ordered list of columns, attributes, and measures.
List of aggregations (function, list of dimensions)	List of aggregations specifying how to calculate the measure. Each aggregation has an SQL aggregation function and an optional list of dimensions to apply the function to.
Data type (schema, name, length, scale)	Determines the data type of the measure. The data type is based on SQL data types, and composed by data type schema, name, length, and scale. The OLAP Center displays the schema only if it is a schema other than SYSIBM.

Attributes

An attribute represents the basic abstraction of the database table columns. An attribute is defined by a SQL expression that can be a simple mapping to a table column, can involve multiple columns and other attributes, and can involve all functionality of the underlying database such as user-defined functions. When other attributes are used in the defining SQL expression, the other attributes cannot form attribute reference loops. For example, if Attribute A references Attribute B, then Attribute B cannot reference Attribute A.

The OLAP Center for DB2 Cube Views hides much of the complexity of the attribute object definition. In the OLAP Center, you do not need to explicitly define the expression template or parameter list of the attribute. If you want to create an attribute that directly maps to a column, you select the source column when you create the attribute in the Dimension wizard or the Dimension Properties window. If you want to create a calculated attribute, you can use the SQL Expression Builder window to create the source expression. The SQL Expression Builder provides lists of available attributes, columns, operators, functions, and constants.

If you want to create an attribute without using the OLAP Center, you must create the attribute's SQL expression definition as a combination of two properties: a template and a list of columns and attributes. The template uses a token notation where `{$$n}` is the token with `n` referencing a specific column or attribute from the list. The list of columns and attributes is ordered and the position of a column or attribute in the list corresponds to the token `n` value. For example, the template `template = "{$$1} || ' ' || {$$2}"` can be used with a corresponding list such as `list = "Column CUSTOMER.FIRSTNAME, Attribute LastName"` to concatenate the first name and last name of customers with a space between the names. Replacing the template tokens with the correct list references, the SQL expression is `"Customer.FirstName || ' ' || LastName"`. The attribute reference is further expanded to a column reference to form the SQL expression: `"Customer.FirstName || ' ' || Customer.LastName"`.

Each name, when fully qualified by a schema, must be unique among attributes and measures.

The following table describes the specific properties that define an attribute. The OLAP Center defines each of these for you when you create an attribute object.

Table 11. Attribute properties

Property	**Description**
SQL expression template	SQL expression that defines the attribute. The template references columns and attributes by using a `{$$n}` notation, where `n` is an ordinal number corresponding to the list of columns and attributes.

Table 11. Attribute properties (continued)

Property	Description
List of columns and attributes for SQL expression	Ordered list of all columns and attributes composing the attribute. These columns and attributes are applied as specified in the SQL expression template.
Data type (schema, name, length, scale)	Determines the data type of the attribute. The data type is based on SQL data types, and composed by data type schema, name, length, and scale. The OLAP Center displays the schema only if it is a schema other than SYSIBM.

Attribute relationships

An attribute relationship describes relationships of attributes in general. The relationships consist of the following properties:

- A left and a right attribute
- A type
- A cardinality
- Determining a possible functional dependency

Type describes what the role of the right attribute is with respect to the left attribute. There are two possible types: Descriptive and Associated.

Descriptive
Specifies that the right attribute is a descriptor of the left attribute. For example, a ProductName right attribute describes a ProductCode left attribute.

Associated
Specifies that the right attribute is associated with the left attribute, but is not a descriptor of the left attribute. For example, a CityPopulation right attribute is associated with but not a descriptor of CityID.

Cardinality describes how the instances of the left and right attributes are related. You can use the following cardinalities for attribute relationships:

1:1 There is at most one left-attribute instance for each right attribute instance, and at most one right attribute instance for each left-attribute instance.

1:Many
: There is at most one left-attribute instance for each right-attribute instance, and any number of right-attribute instances for each left-attribute instance.

Many:1
: There is any number of left-attribute instances for each right-attribute instance, and at most one right-attribute instance for each left-attribute instance.

Many:Many
: There is any number of left-attribute instances for each right-attribute instance, and any number of right-attribute instances for each left-attribute instance.

The functional dependency property tells whether the attribute relationship defines a functional relationship between two attributes. Specifying that an attribute relationship is a functional dependency means that you guarantee that every instance of the left attribute will determine the instance of the right attribute. You cannot define a functional dependency unless the left attribute is unique. For example, if you have a hierarchy with Country, State, and City with City_Pop included in an attribute relationship with City, City can determine City_Pop. However, outside the context of the hierarchy, you might have multiple cities with the same name, such as Concord, California and Concord, New Hampshire. The attribute relationship between City and City_Pop is not a functional dependency. You can define several attribute relationships that are functional dependencies for a hierarchy based on unique IDs, such as CountryID, StateID, and CityID. You can have attribute relationships between CountryID and Country, StateID and State, CityID and City and CityID and City_Pop. Each of attribute relationships is also a functional dependency.

The main use of an attribute relationship is within the context of a hierarchy in a dimension. Attributes that are directly related to the hierarchy attributes can be queried as part of the hierarchy. This allows each level of the hierarchy to define attributes that complement the information of a given level. For example, a hierarchy can have a City attribute. The City attribute can be related to a City_Pop attribute with an attribute relationship. With the attribute relationship information, you can include City_Pop information in a query that includes City.

You can explicitly and implicitly create an attribute relationship object in the OLAP Center. You can explicitly create an attribute relationship using the Attribute Relationship wizard. Open the Attribute Relationship wizard from the Relational objects view and specify all of the object definition properties. You can implicitly create an attribute relationship when you define a hierarchy for a dimension. In the Hierarchy wizard, you can optionally define related attributes for any of the hierarchy attributes. In the Related Attributes Selection window which you can open from the Hierarchy wizard, you can see only the attributes used in attribute relationships with a valid cardinality per the validation rules described in "Metadata object rules" on page 38. For each related attribute that you add, the OLAP Center creates an attribute relationship with the hierarchy attribute as the left attribute and the related attribute as the right attribute. You can view and modify these attribute relationships from the Attribute Relationships folder in the Relational Objects View.

The specific properties that define an attribute relationship object are described in the following table.

Table 12. Attribute relationship properties

Property	Description
Left attribute	Left attribute used in the relationship.
Right attribute	Right attribute used in the relationship.
Type	Type of relationship described by the attribute relationship. The type is used to determine what role an attribute serves: DESCRIPTIVE, ASSOCIATED
Cardinality	Cardinality expected in the join: 1:1, 1:Many, Many:1, Many:Many
Functional dependency	Determines if the attribute relationship is also a functional dependency: YES, NO

Joins

A join object joins two relational tables together. A join references attributes that reference columns in the tables being joined. The simplest form of a join references two attributes: one that maps to a column in the first table and one that maps to a column in the second table. You also specify an operator to indicate how the columns will be compared.

A join can also be used to model composite joins where two or more columns from the first table are joined to the same number of columns in the second table. A composite join uses pairs of attributes to map corresponding columns together. Each pair of attributes has an operator that indicates how that pair of columns will be compared. A join also has a type and cardinality. The join types map to relational join types. Joins can be used in dimensions to join dimension tables or in a cube model to join the dimensions of a cube model to its facts object or within a facts object to join multiple fact tables together. You can use the Join wizard in the OLAP Center to create a join.

The specific properties that define a join are described in the following table.

Table 13. Join properties

Property	Description
List of (left attribute, right attribute, operator)	Left attribute: The attribute on the left side of the join. Right attribute: The attribute on the right side of the join. Operator: Operator expected in the join =, <, >, <>, >=, <=.
Type	Type of join expected: INNER, FULL OUTER, LEFT OUTER, RIGHT OUTER
Cardinality	Cardinality expected in the join: 1:1, 1:Many, Many:1, Many:Many

Cubes

A cube is a precise definition of an OLAP cube that can be delivered using a single SQL statement. A cube, which is derived from a cube model, contains a subset of objects from the cube model. The cube facts and list of cube dimensions are subsets of those in the referenced cube model. Cubes are appropriate for tools and applications that do not use multiple hierarchies because cube dimensions allow only one cube hierarchy per cube dimension.

Cube metadata can be used when optimizing a cube model for extract and drill through queries. You can use the Cube wizard in the OLAP Center to create a cube. You must have a complete cube model to create an associated cube. The properties of a cube object are described in the following table.

Table 14. Cube properties

Property	Description
Cube model	Cube model from which the cube is derived.
Cube facts	Cube facts used in the cube. The cube facts is derived from the facts object in the cube model.
List of cube dimensions	Ordered list of cube dimensions used in the cube. The cube dimension is derived from the dimensions in the cube model. One cube hierarchy is associated with each cube dimension.

Cube facts

A cube facts has a subset of measures in an ordered list from a specific facts object. A cube facts allows a cube the flexibility to scope a cube model's facts. The structural information, such as the joins and attributes, is referenced from the parent facts object.

In the OLAP Center, you create a cube in the context of a cube model, using the Cube wizard. You do not need to explicitly define the cube facts because the OLAP Center knows the cube facts is derived from the facts object in the associated cube model. You select which measures from the cube model facts object that you want to use in the cube.

The specific properties that define a cube facts are described in the following table.

Table 15. Cube facts properties

Property	Description
Facts	Facts from which the cube facts is derived.
List of measures	Ordered list of measures that is used in a cube. All measures must be part of the facts from which the cube facts is derived.

Cube dimensions

A cube dimension is used to scope a dimension for use in a cube. The cube dimension object references the dimension from which it is derived and the relevant cube hierarchy for the given cube. Only one cube hierarchy can be applied to a cube dimension. The joins and attributes that apply to the cube dimension are referenced from the dimension definition.

In the OLAP Center, you create a cube in the context of a cube model, using the Cube wizard. You select which of the cube model dimensions that you want to have in your cube. For each dimension that you include as a cube dimension, you can select which attributes to include in the cube hierarchy.

The specific properties that define a cube dimension object are described in the following table.

Table 16. Cube dimension properties

Property	Description
Dimension	Dimension from which the cube dimension is derived.

Table 16. Cube dimension properties (continued)

Property	Description
Cube hierarchy	Cube hierarchy that applies to the cube dimension.

Cube hierarchies

A cube hierarchy is a subset of a hierarchy, and it is used in a cube. A cube hierarchy references the hierarchy from which it is derived (parent hierarchy), and it can have a subset of the attributes from the parent hierarchy. Additionally, a cube hierarchy object references the attribute relationships that apply to the cube. Only one cube hierarchy can be defined for a cube dimension of a cube. In general, a cube hierarchy has the same hierarchy type and deployment mechanism as the hierarchy from which it is derived. If the hierarchy is the network type, the cube hierarchy is balanced if there are no missing members and ragged if there are missing members.

In the OLAP Center you create a cube in the context of a cube model by using the Cube wizard. You select which of the cube model dimensions you want to have in your cube. For each dimension you include as a cube dimension, you can select which attributes to include in the cube hierarchy. If a selected dimension has related attributes defined, you can include the related attributes in your cube hierarchy. An attribute relationship exists for each related attribute in the hierarchy.

The specific properties that define a cube hierarchy are described in the following table.

Table 17. Cube hierarchy properties

Property	Description
Hierarchy	Hierarchy from which the cube hierarchy is derived.
Lists of attributes	Ordered list of all attributes from the top to the bottom of the cube hierarchy. The order of the attributes should be the same as in the parent hierarchy.
Set of attribute relationships	Set of all attribute relationships that link cube hierarchy attributes to other attributes.

Metadata object rules

Three types of rules apply to metadata objects: base rules, cube model completeness rules, and optimization rules. These rules ensure that each object is valid both in and out of the context of a cube model and that effective SQL queries can be written and optimized.

For information about how to create metadata objects, see the online help in the OLAP Center.

Base rules

Base rules define an object's validity outside the context of its use. Every metadata object has its own set of rules. An instance of a metadata object is valid if it follows all of the object rules. The base rules for each type of metadata object are listed in the following sections.

Cube model base rules:

- A cube model must refer to zero or one facts.
- A cube model must refers to zero or more dimensions.
- Dimension—join pairs must have both a dimension and a join.
- All attributes on one side of a dimension to fact join must exist in the dimension's attribute list and all attributes on the other side of the join must exist in the facts' attribute list.
- A cube model must reference all the explicit dimensions referenced by the aggregations of the measures from the cube model's facts. If a cube model has dimensions, an aggregation with empty list of dimensions must match to at least one dimension from the cube model. Ensure that the dimension is not referenced in other aggregations of the same measure. However, if a cube model has no dimensions, all the measures must have only aggregations with an empty list of dimensions.

Facts objects base rules:

- A facts object must refer to at least one measure.
- All attributes and measures referenced by a facts object must be able to be joined within the facts object. Only the joins of the facts object are considered.
- Exactly one join can be defined between any two tables within the facts object.
- Join loops are not allowed within a facts object.
- Joins referenced by a facts object must refer to the attributes of the facts object.

Dimension base rules:

- A dimension must refer to at least one attribute.

- All attributes referenced by a dimension must be able to be joined within the dimension. Only the joins of the dimension are considered.
- Join loops are not allowed within a dimension.
- Exactly one join can be defined between any two tables within the dimension.
- Hierarchies referenced by a dimension must refer to the attributes of the dimension.
- Attribute relationships that are referenced by the hierarchies of a dimension must refer to the attributes of the dimension.
- Joins referenced by a dimension must refer to the attributes of the dimension.

Hierarchy base rules:
- A hierarchy must refer to at least one attribute.
- Exactly two attributes must exist for a recursive deployment.
- Every attribute relationship within a hierarchy must refer to two attributes where the left attribute is part of the hierarchy and the right attribute is not part of the hierarchy.
- Every attribute relationship within a hierarchy must have a cardinality of 1:1 or Many:1.
- Standard deployment can be used for all types of hierarchies and a recursive deployment can only be used for unbalanced hierarchies.

Measure base rules:
- Each SQL expression template can have zero or more of the following parameters: attributes, columns, and measures.
- Attributes and measures used as parameters for a SQL expression template cannot form a dependency loop.
- The SQL template of a measure cannot be an empty string.
- A SQL template cannot use aggregation functions.
- If at least one measure, and only measures are referenced, defining an aggregation is optional.
- The number of SQL templates must match the number of parameters of the first aggregation function, if an aggregation exists.
- A measure with multiple SQL templates must define at least one step in the aggregation script.
- If a measure refers to a second measure that defines multiple SQL templates, then the referring measure cannot have an aggregation script.
- A multi parameter aggregation function can only be used in the first aggregation.

- If a measure defines one or more aggregations, one aggregation must specify an empty list of dimensions.
- A measure can reference each dimension only once either within an aggregation or across aggregations.
- In a SQL template, token indicators must begin numbering with 1 and must continue numbering consecutively.
- Within a SQL expression, every column, attribute, and measure must be referenced at least once.

Attribute base rules:

- Each SQL template have have zero or more of the following parameters attributes and columns.
- Attributes used as parameters for a SQL expression template cannot form a dependency loop.
- The SQL template of an attribute cannot be an empty string.
- The SQL template cannot have aggregation functions.
- In a SQL template, token indicators must begin numbering with 1 and must continue numbering consecutively.
- Within a SQL expression, every column and attribute must be referenced at least once.

Attribute relationship rules:

- An attribute relationship must refer to two attributes.
- An attribute relationship cannot be defined as a functional dependency with a Many:Many cardinality.

Join rules:

- A join must refer to at least one triplet (left attribute, right attribute, operator).
- A valid operation must be defined for each join triplet. The data types of the left and right attributes should be compatible with each other and with the specified operation.
- All left attributes must resolve into one or more columns of a single table.
- All right attributes must resolve into one or more columns of a single table.

Cube rules:

- A cube must refer to one cube facts.
- A cube must refer to at least one cube dimension.
- Cube facts must be derived from the facts used in the referenced cube model.

- All cube dimensions must be derived from the dimensions used in the referenced cube model.

Cube facts rules:
- A cube facts must refer to one facts.
- A cube facts must refer to at least one measure.
- All measures referenced by a cube facts must also be referenced in the corresponding facts.
-

Cube dimension rules:
- A cube dimension must refer to one dimension.
- A cube dimension must refer to one cube hierarchy.
- The referenced cube hierarchy must be derived from a hierarchy used in the referenced dimension.

Cube hierarchy rules:
- A cube hierarchy must refer to one hierarchy.
- A cube hierarchy must refer to at least one attribute.
- All referenced attributes must be used in the referenced hierarchy.
- Cube hierarchy attributes must be listed in the same order as in the referenced hierarchy. Attribute order does not apply to network hierarchies.
- Every attribute relationship referenced by a cube hierarchy must refer to two attributes where the left attribute is referenced by the cube hierarchy and the right attribute is not referenced by the cube hierarchy.
- All attribute relationships referred to in a cube hierarchy must also be referred to in the corresponding hierarchy.

Completeness rules

Cube model completeness rules extend the base rules to ensure that a cube model links to other objects appropriately and that effective SQL queries can be written.

Cube model completeness rules:
- A cube model must refer to one facts.
- A cube model must refer to at least one dimension.

Optimization rules

Optimization rules further extend the base rules and cube model completeness rules. These rules ensure that the SQL queries created for the metadata can be optimized successfully.

Join optimization rules:

- A constraint must be defined for the columns that participate in a join. If the same set of columns are used on both sides of the equality, a primary key must be defined that matches the set of columns. If different sets of columns are used on each side of the equality, one side must have a matching primary key, and the other side must have a matching foreign key and reference the primary key.
- The join cardinality must be 1:1, Many:1, or 1:Many. In a join with the same set of columns on both sides of the equality, the cardinality must be 1:1. For all other joins, the cardinality must have 1 on the side with the primary key defined, and N on the side with the foreign key defined. If the foreign key side also has a primary key defined, a 1 must be used for the cardinality of that side.
- All attributes used in the join must resolve to nonnullable SQL expressions.
- The join type must be an inner join.

Dimension optimization rule:

- A dimension must have one primary table to which joins attach with a 1:1 or Many:1 cardinality.

Cube model optimization rule:

- The join used to join the facts and dimension must have a cardinality of 1:1 or Many:1 and must join a facts' table to a dimension's primary table.

Chapter 3. Cube model optimization

This chapter describes the following topics:
- Summary tables
- Overview of the optimization process
- Metadata design considerations for optimization
- Constraint definitions for optimization
- Optimizing a cube model
- Parameters for the Optimization Advisor
- Example of an SQL script to create summary tables
- Testing query results
- Troubleshooting summary tables
- Summary table maintenance
- Dropping a summary table

Summary tables

DB2 Cube Views uses DB2 summary tables to improve the performance of queries issued to cube models. A summary table is a special type of a materialized query table (MQT) that specifically includes summary data. Because the Optimization Advisor always recommends MQTs with summarized data, the term summary table is used in the DB2 Cube Views documentation to describe the recommended MQTs.

You can complete expensive calculations and joins for your queries ahead of time and store that data in a summary table. When you run queries that can use the precomputed data, DB2 will reroute the queries to the summary table. A query does not need to match the precomputed calculations exactly. If you use simple analytics like SUM and COUNT, DB2 can dynamically aggregate from the precomputed data. Many different queries can be satisfied by one summary table. Using summary tables can dramatically improve query performance for queries that access commonly used data or that involve aggregated data over one or more dimensions or tables.

Figure 14 on page 44 shows a cube model based on a star schema with a Sales facts object and Time, Region, and Product dimensions. The dimensions each have a set of attributes, the facts object has measures and attributes, and each dimension is joined to the facts by a facts-dimension join.

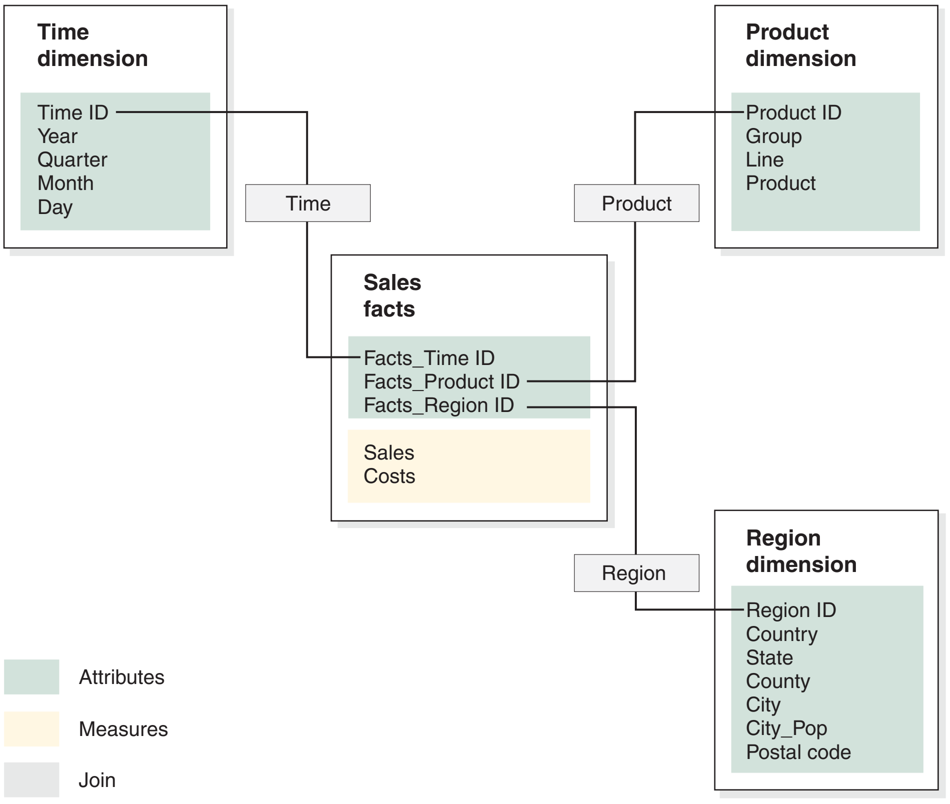


Figure 14. Cube model with a Sales facts object and Time, Product and Region dimensions

The hierarchy for each dimension in the cube model is shown in Figure 15 on page 45. The highlighted boxes connected by the thick dark line represent the data that actually exists in the base tables. Sales data is stored at the Day level, Postal code level, and Product level. Data above the base level in the hierarchy must be aggregated. If you query a base table for sales data from a particular quarter, DB2 must dynamically add the daily sales data to return the quarterly sales figures. For example, you can use the following original query to see the sales data for each country by each quarter in 2002 and each product line in electronics:

```
SELECT QUARTER, COUNTRY, LINE, SUM(SALES)
FROM TIME, REGION, PRODUCT, SALES
WHERE SALES.REGIONID = REGION.REGIONID
  AND SALES.TIMEID = TIME.TIMEID
  AND SALES.PRODUCTID = PRODUCT.PRODUCTID
  AND YEAR = '2002'
  AND GROUP = 'Electronics'
GROUP BY COUNTRY, QUARTER, LINE;
```

The thin line connecting the Quarter-Country-Line slice in Figure 15 represents the slice that the query accesses. Quarter-Country-Line is a slice of the cube model and includes one level from each hierarchy. Summary tables are defined to satisfy queries at or above a particular slice. A summary table can be built for the Quarter-Country-Line slice accessed by the query. Any other queries that access data at or above that slice including All Time, Year, All Regions, All Products, and Group can be satisfied by the summary table with some additional aggregating. However, if you wanted to query more detailed data below the slice, such as Month or City, DB2 cannot use the summary table for that query.

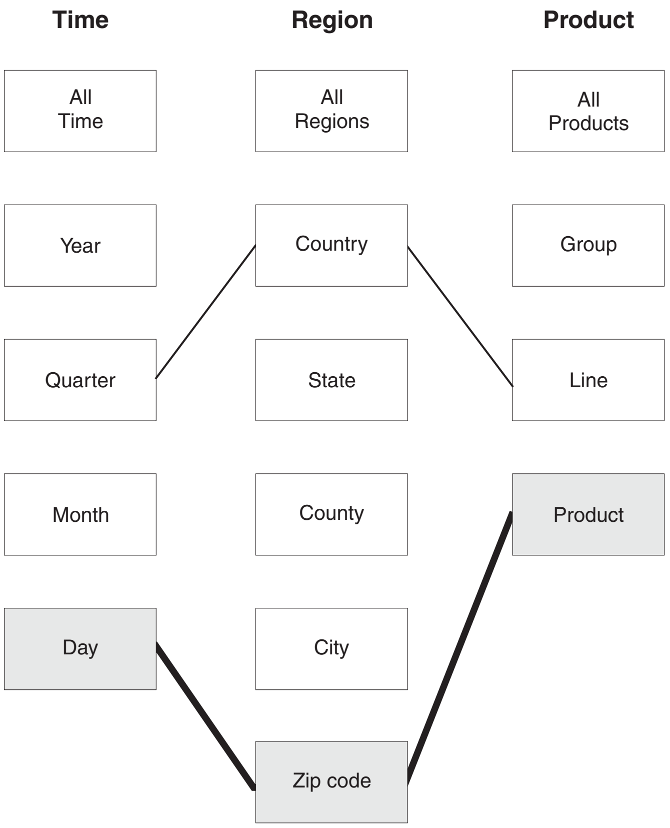


Figure 15. Time, Region, and Product hierarchies. Shows the slice that the previously described query uses.

In Figure 16 on page 46, the dotted line defines the Quarter-City-Line slice. A summary table built for the Quarter-City-Line slice can satisfy any query that accesses data at or above the slice. All of the data that can be satisfied by a summary table built for the Quarter-City-Line slice is included in the top set of highlighted boxes.

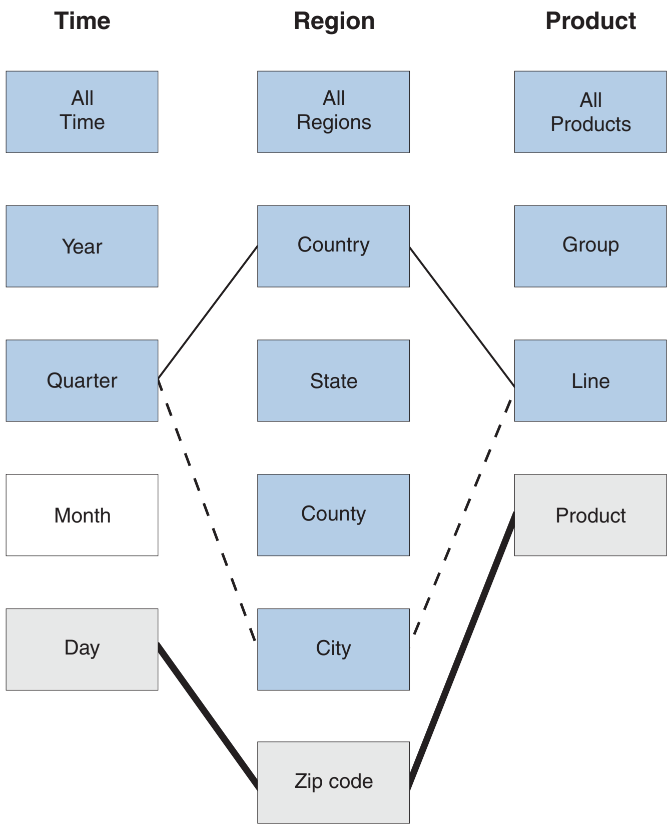


Figure 16. Time, Region, and Product hierarchies. The highlighted data can be satisfied by a summary table built at the Quarter-City-Line slice.

The rewriter in the DB2 SQL compiler knows about the summary tables, and can automatically rewrite queries to read from the summary table instead of the base tables. Rewritten queries are typically much faster because the summary tables are usually much smaller than the base tables and contain preaggregated data. Users continue to write queries against the base tables. DB2 will decide when to use a summary table for a particular query and will rewrite the user's query to go against the summary tables instead, as shown in Figure 17 on page 47. The rewritten query accesses only one table, but the original query accesses multiple tables to return the same results.

You can use the DB2 EXPLAIN facility to see if the query was rerouted, and if applicable, which table it was rerouted to. For more information on using DB2EXPLAIN, see "Testing query results" on page 68.

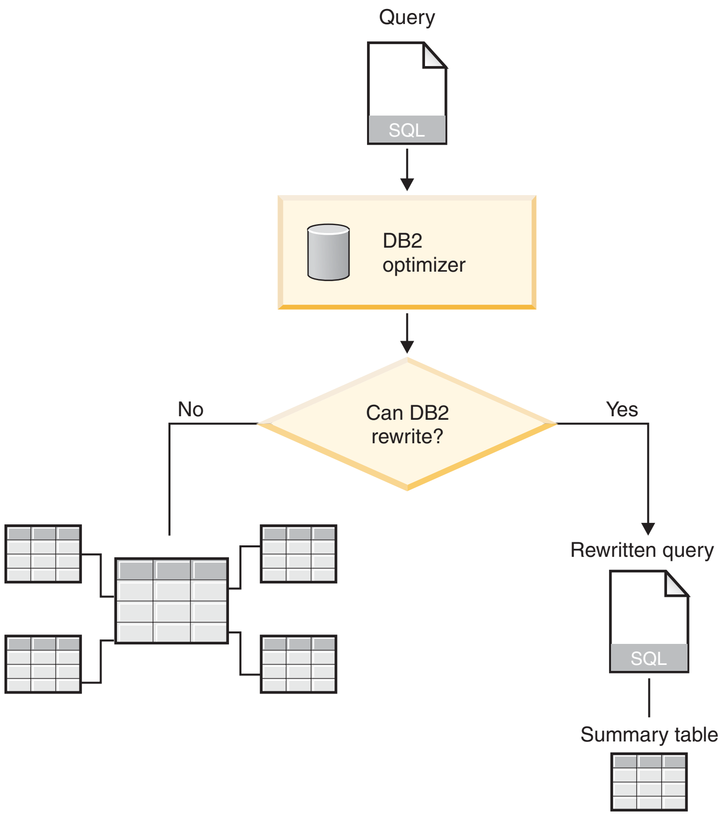


Figure 17. DB2 query rewrite process

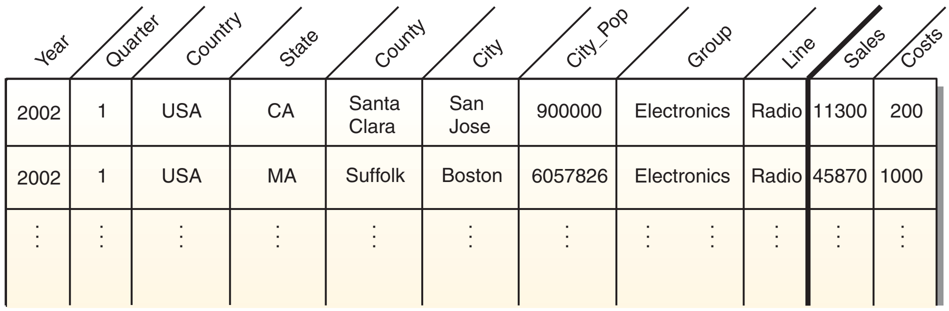
The query to see the sales data for each country by each quarter in 2002 and each product line in electronics, can be rewritten to use the summary table built for the Quarter–City–Line slice. The original query:

```
SELECT QUARTER, COUNTRY, LINE, SUM(SALES)
FROM TIME, REGION, PRODUCT, SALES
WHERE SALES.REGIONID = REGION.REGIONID
  AND SALES.TIMEID = TIME.TIMEID
  AND SALES.PRODUCTID = PRODUCT.PRODUCTID
  AND YEAR = '2002'
  AND GROUP = 'Electronics'
GROUP BY COUNTRY, QUARTER, LINE;
```

can be rewritten as:

```
SELECT QUARTER, COUNTRY, LINE, SUM(SALES)
FROM SUMMARYTABLE1
WHERE YEAR = '2002'
  AND GROUP = 'Electronics'
GROUP BY QUARTER, COUNTRY, LINE;
```

The rewritten query is much simpler and quicker for DB2 to complete because the table joins are precomputed so DB2 accesses one table instead of three. DB2 needs to calculate data for Country from the higher level City instead of from the lower level Postal code, so the summary table has fewer rows than the base tables because there are fewer cities than zip codes. DB2 does not need to perform any additional calculations to return sales data by Quarter and Line because the data is already aggregated at these levels. Because the summary table joins the tables used in the query ahead of time, the joins do not need to be performed at the time the query is issued. For more complex queries, the performance gains can be dramatic.



Year	Quarter	Country	State	County	City	City_Pop	Group	Line	Sales	Costs
2002	1	USA	CA	Santa Clara	San Jose	900000	Electronics	Radio	11300	200
2002	1	USA	MA	Suffolk	Boston	6057826	Electronics	Radio	45870	1000
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮ ⋮	⋮	⋮	⋮

Figure 18. Summary table created for the Quarter–City–Line slice

When the Optimization Advisor recommends a summary table, all of the measures in the cube model are included. Additionally, all attributes and all corresponding attribute relationships defined at or above the slice are included in the table. Figure 18 shows the summary table for the Quarter-City-Line slice. Only two measures, Sales and Costs, are included, and only one attribute is included by an attribute relationship, City_Pop. However, if you define fifty measures for your cube model, all fifty are included in the summary table.

Overview of the optimization process

Optimizing with DB2 Cube Views can improve the performance of OLAP-style SQL queries. The Optimization Advisor wizard can help you optimize your cube models by creating summary tables. DB2 summary tables can improve query performance because they contain precomputed results from one or more tables that can be used in a query. Costly table joins and complex calculations can be computed in advance and stored in a summary table so that future queries using these aggregations can run much faster. For more information on summary tables, see "Summary tables" on page 43.

The Optimization Advisor will analyze your metadata and the information that you provide to the wizard and recommend the appropriate summary tables. After running the Optimization Advisor, you will have an SQL file that can build the set of recommended summary tables. You have the option of modifying the SQL before you run it to create the summary tables.

Running the Optimization Advisor is just one step in the optimization process. Before you begin optimizing, you need to think about several issues including, but not limited to:

- How to effectively use DB2 constraints on the base tables
- What types of queries you want to optimize for
- How much space you want to provide
- How you will maintain the accuracy of your summary tables

Before you can optimize, you must define constraints on your base tables. For more information on the types of constraints that are required, see “Constraint definitions for optimization” on page 53.

Many parts of the optimization process are iterative and might need to be repeated to fine tune and maintain your performance gains. Figure 19 on page 50 shows an overview of the main steps in the optimization process.

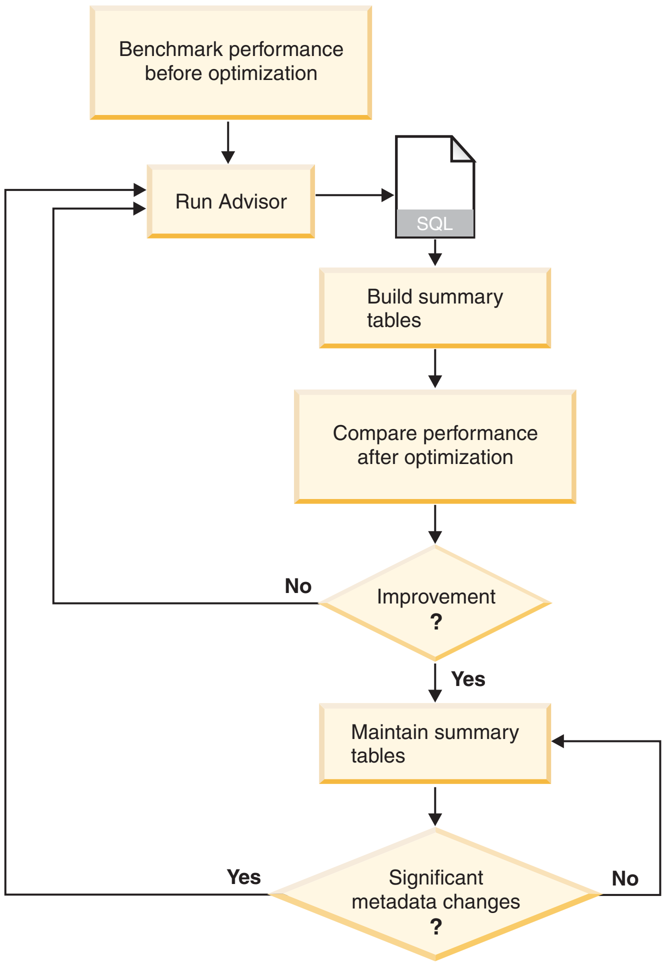

Benchmark performance before optimization
Run Advisor
SQL
Build summary tables
Compare performance after optimization
No
Improvement ?
Yes
Maintain summary tables
Yes
Significant metadata changes ?
No

Figure 19. Overview of the main steps in the optimization process

The optimization process includes these general tasks:

- **Measure performance**

  Before you run the DB2 Cube Views Optimization Advisor, you should measure the current performance for a specific set of typical queries. Performance measurement is an optional step that provides a benchmark so that you can later analyze the success of the optimization. You can use the db2batch Benchmark Tool provided with DB2 Universal Database to create a benchmark. For more information on using db2batch, see "Testing query results" on page 68. You run sample queries to complete the performance benchmark, but the Optimization Advisor does not require sample queries because it is metadata based and makes recommendations without knowing the specific queries that will be issued.

- **Run the Optimization Advisor wizard**

  You provide several important parameters to the wizard including: types of queries that you want to optimize for, disk space and time limitations, update method, and table space locations. For more information on these parameter choices, see “Parameters for the Optimization Advisor” on page 58. The Optimization Advisor creates its recommendations based on the information that you provide, metadata, DB2 statistics, and any data sampling that you allow. The Optimization Advisor considers the parameters that you specify and generates two SQL files. One SQL file contains the SQL commands to build a set of recommended summary tables. The other SQL file contains the SQL commands to update the recommended summary tables.
- **Create the summary tables**

  You can create the summary tables immediately after completing the wizard, or you can add the operation to your normal database maintenance schedule. Creating the summary tables can require significant time and processing resources. After the summary tables are built, verify that the performance of queries against the optimized cube model are improved. Run the same set of sample queries that you ran before optimizing, and compare the performance results. If you do not see significant performance gains, you might need to run the Optimization Advisor wizard again and allocate more disk space, or time, or both, or you might need to change other settings. For more information on how to verify and analyze your performance results, see “Testing query results” on page 68.
- **Maintain the summary tables**

  After your summary tables are created, you need to regularly maintain the tables to ensure that they stay appropriately synchronized with your data. When you run the Optimization Advisor wizard, you choose either a refresh-immediate or refresh-deferred update option. If you choose the immediate update option, DB2 keeps your base tables and summary tables synchronized and incrementally updates the summary tables when the underlying tables are changed. DB2 supports incremental maintenance for simple aggregations such as SUM and COUNT. For other aggregations, the Optimization Advisor recommends the deferred option regardless of which refresh option you select. If you choose the deferred option, you will rebuild your summary tables to update them. You can decide when to perform the summary table update. If you make significant changes throughout your base tables, deferring the update can be more efficient than incremental updates. When choosing between these options, you need to make trade-offs between the resources that you can allocate to the maintenance and how precisely the data must be synchronized. For more information on the immediate and deferred update options, see “Summary table maintenance” on page 71.
- **Periodic reevaluation**

You need to periodically reevaluate the summary tables to ensure that they continue to meet your needs. If you significantly change the metadata by adding or updating a cube model, you might need to run the Optimization Advisor again and build a new set of summary tables. If you add a metadata object such as a new dimension or measure, queries that access data from the new object will not be able to use the existing summary tables. However, queries that do not use the new object will continue to use the summary tables. If you update a metadata object to include data that was not previously optimized for, queries that access the updated object will not be able to use the summary tables. If you delete one or more objects, the effectiveness of the summary tables is not changed, but you are wasting disk space on aggregations that are no longer used.

In addition to significant metadata changes, you might also need to run the wizard again if the regularly performed query types change and are not the type that you optimized for.

Each time you run the wizard and build new tables, you should complete the whole optimization process again including creating a benchmark and analyzing the performance of the summary tables.

If you drop a cube model you can also drop the associated summary tables if they are not used for any other purpose. DB2 Cube Views does not drop summary tables when the associated cube model is dropped. For more information on how to drop a summary table, see "Dropping a summary table" on page 73.

Restriction: DB2 in a federated environment does not use the summary tables that are created by the Optimization Advisor.

Metadata design considerations for optimization

The way that you design your metadata affects the summary tables recommended by the Optimization Advisor wizard. Generally you want to define your facts objects, dimensions, and joins in a particular way according to the structure of your data. You have few choices in the basic structure of these objects, so you can rarely improve the recommended summary tables by altering these objects. However, you do have some flexibility in selecting which cubes, measures, and attribute relationships you want to include for a particular cube model.

Cubes Cubes impact the effectiveness of optimization for extract and drill-through queries. These types of queries optimize for a subset of the cube model based on the cubes that you define. If you expect users to perform extract and drill-through queries, you should build cubes to reflect the data that these queries will likely access. If you perform both extract and drill-through queries for a particular cube model, you should build two cube models: one with cubes designed for extract queries, and a second with cubes designed for drill-through queries. You should not build both types of cubes for one cube model. For more information on extract and drill-through queries, see "Parameters for the Optimization Advisor" on page 58.

Measures
Each measure defined for the cube model is included as a column in the summary table. You cannot choose to optimize for a subset of measures; all measures are automatically included. If you have a large number of measures, your summary table might require a large amount of disk space. If limited disk space is a concern, you might choose to include only critical measures in your cube model and drop any measures that you do not expect to use regularly.

Attribute relationships
Like measures, all attributes involved in attribute relationship in the cube model are included as columns in the summary table. For example, if you define City_Pop and Mayor in an attribute relationship with the City attribute that is defined in your hierarchy, City_Pop and Mayor will also be included in the summary table if City is. Including attribute relationships provides the benefit of being able to group your query results on these items, but at the cost of disk space. If limited disk space is a concern, you might choose to drop all or some of the attribute relationships for your cube model.

Constraint definitions for optimization

You must define constraints on your base tables before you can use the Optimization Advisor wizard. The constraints need to enforce the base, cube model completeness, and optimization rules that are described in "Metadata object rules" on page 38. Ensure that the SQL queries created for the metadata can be optimized successfully. The rules primarily define how to join together the metadata objects of your cube model.

You can use informational constraints for the foreign key constraints that you need to define. Informational constraints are a new type of constraint offered in DB2 Universal Database, Version 8. Informational constraints provide a way to improve query performance without increasing maintenance costs. These constraints can be used by the DB2 SQL compiler but are not enforced by the database manager. This type of constraint allows DB2 to know about the relationships in the data without requiring the relationship to be enforced. See the DB2 Information Center for details about how to define informational constraints. For constraints involving primary keys, you must use the database-enforced constraints provided with DB2.

Each column that participates in a join must have a constraint defined. For example, columns involved in facts-to-dimension joins and if applicable dimension-to-dimension joins used in a snowflake schema need constraints.

To optimize a cube model based on the simple star schema shown in Figure 20, you need to define constraints on each of the facts-to-dimension joins. The three facts-to-dimension joins are:

- between Time.TimeID and Sales.TimeID
- between Product.ProductID and Sales.ProductID
- between Region.RegionID and Sales.RegionID

Several rules apply to each of these joins. You can use informational constraints only for foreign key constraints.

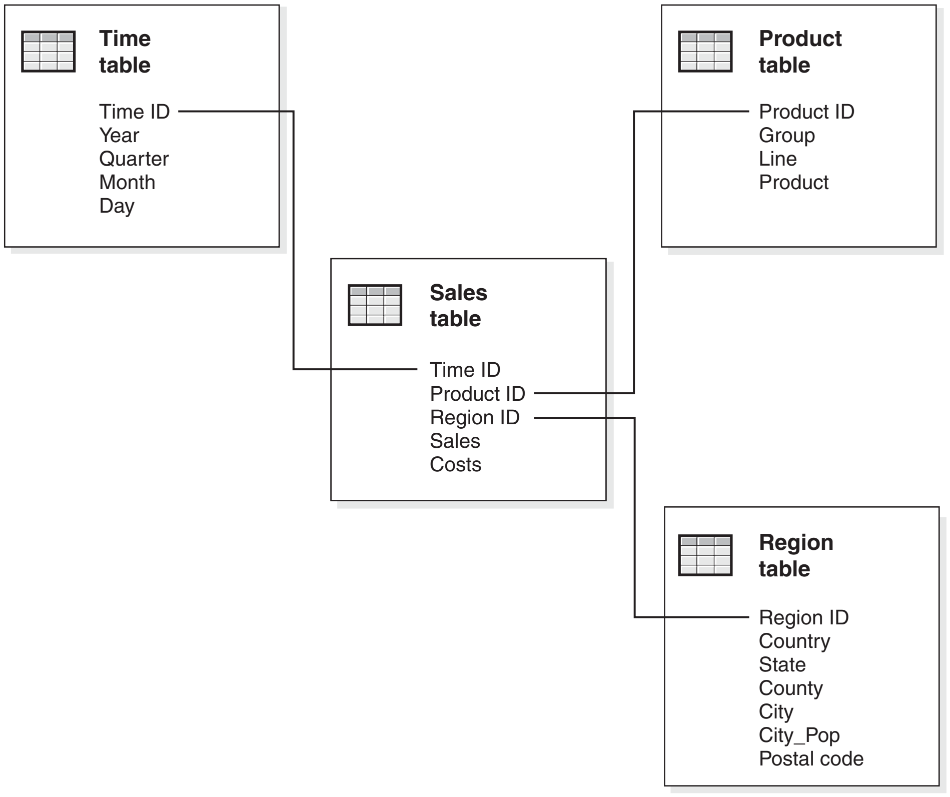


Figure 20. Relational tables used in a simple star schema

For the join between the Time and Sales tables, you must define constraints for the following rules:

- TimeID is the primary key in the Time table.
- Time.TimeID and Sales.TimeID are both nonnullable columns.
- Sales.TimeID is a foreign key referencing Time.TimeID. Foreign key constraints can be defined as informational constraints.

- The join cardinality is 1:Many (Time.TimeID : Sales.TimeID) if Sales.TimeID is not the primary key for the Sales table. If the Sales.TimeID is the primary key for the Sales table, then the join cardinality is 1:1.
- The join type is INNER JOIN.

For the join between the Product and Sales tables, you must define constraints for the following rules:
- ProductID is the primary key in the Product table.
- Product.ProductID and Sales.ProductID are both nonnullable columns.
- Sales.ProductID is a foreign key referencing Product.ProductID. Foreign key constraints can be defined as informational constraints.
- The join cardinality is 1:Many (Product.ProductID : Sales.ProductID) if Sales.ProductID is not the primary key for the Sales table. If Sales.ProductID is the primary key for the Sales table, the the join cardinality is 1:1.
- The join type is INNER JOIN.

For the join between the Region and Sales tables, you must define constraints for the following rules:
- RegionID is the primary key in the Region table.
- Region.RegionID and Sales.RegionID are both nonnullable columns.
- Sales.RegionID is a foreign key referencing Region.RegionID. Foreign key constraints can be defined as informational constraints.
- The join cardinality is 1:Many (Region.RegionID : Sales.RegionID) if Sales.RegionID is not the primary key for the Sales table. If Sales.RegionID is the primary key for the Sales table, the the join cardinality is 1:1.
- The join type is INNER JOIN.

If your cube model is based on a snowflake schema, you must define additional constraints on the joins between the dimension tables. Each dimension has a primary dimension table, to which one or more additional dimensions can join. The primary dimension table is the only table that can join to the fact table. Each of the outrigger tables that join directly to the primary table must have a join cardinality of Many:1 (where Many is on the side of the primary table) or 1:1. The primary dimension table usually has the most detailed level of information of all of the dimension tables because of these join cardinality rules. If a set of dimension tables only uses 1:1 join cardinalities, then all of the tables have the same level of detail.

Figure 21 on page 56 shows a valid set of dimension tables in a snowflake schema dimension. The primary dimension table is the Customer table, with three additional outrigger tables including City and CustomerGroup joined directly to Customer, and CityInfo joined to City. The join cardinalities are

semantically valid because there can be many customers in a city or a customer group, and one set of city information exists per city. This is a valid dimension for optimization because it conforms to the optimization validation rules. The dimension has only one primary table, and the City and CustomerGroup tables joined directly to the primary table are joined with a Many:1 cardinality. The CityInfo table is joined with a 1:1 cardinality which is also valid. The Customer table has the most detailed level of information out of the four dimension tables.
Figure 22 on page 57 shows an invalid set of dimension tables in a snowflake

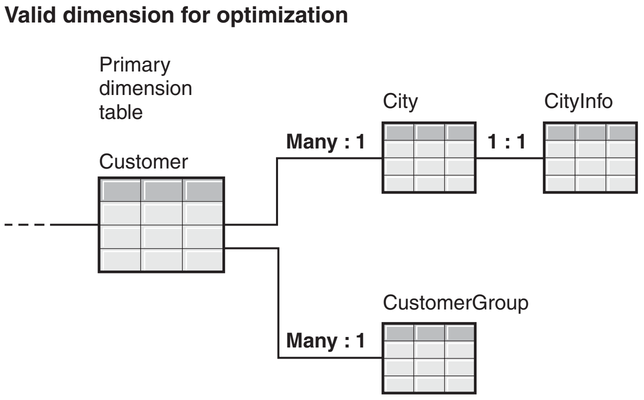


Figure 21. A set of dimension tables used in one dimension that can be optimized

schema dimension. Because of the cardinality relationships that are defined, it is not possible for any of these tables to be the primary dimension table in a cube model that will be optimized. Although the cardinalities are semantically valid, if any of these tables joined with the fact table as the primary dimension table, the data in the fact table would be multiplied, which causes what is known as a fan trap. For example, if Customer is the primary dimension table, the 1:Many join cardinality between Region and SalesRep makes the dimension invalid for optimization. If each region has five sales representatives, then when the SalesRep and Region tables are joined, there are five entries for each region. When these tables are joined with the City and Customer tables, and ultimately with the fact table, an additional five rows are added for each existing row in the City, Customer and facts table. Repeating the same fact row five times causes the measures to be miscalculated. Each of the other tables in the dimension has similar problems. The City table cannot be the primary dimension table because of the 1:Many joins between City and Customer and between Region and SalesRep. The Region table cannot join with the fact table because every join in the dimension is a 1:Many join to the Region table. The SalesRep table cannot be the primary dimension table either because of the 1:Many joins between the Region and City tables and the City and Customer tables.

Invalid dimension for optimization

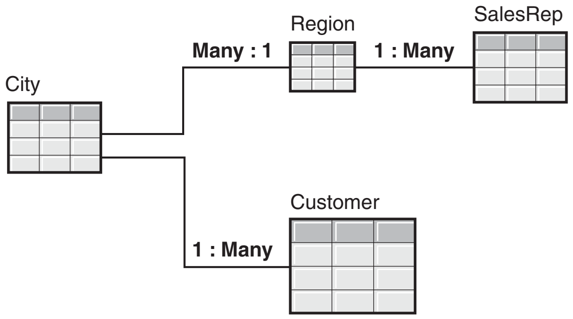


Figure 22. A set of dimension tables used in one dimension that cannot be optimized

Optimizing a cube model

Use the Optimization Advisor wizard to create SQL that can build a set of recommended summary tables for a cube model.

By optimizing for queries performed against a cube model, you can improve the performance of products that issue OLAP-style SQL queries. Summary tables aggregate commonly accessed data to speed up query performance.

Prerequisites:

You must have DB2 constraints specified for your base tables used in the cube model. Constraints must be specified between each fact table and dimension table and between each dimension table in a snowflake schema. The constraints must be specified on nonnullable columns. DB2 Universal Database, Version 8 offers informational constraints a new type of constraint that is less costly because it is not enforced. You can use informational constraints to define foreign key constraints. For more information on defining constraints, see “Constraint definitions for optimization” on page 53.

Procedure:

To optimize a cube model:

1. Open the Optimization Advisor wizard by right-clicking a cube model in the OLAP Center object tree and clicking **Optimization Advisor**.
2. On the Query page, specify the types of queries that you want to optimize for. The information that you provide is used to improve the optimization results. You should select the type or types of queries that you expect to be performed most often. The query types describe how DB2 relational data is typically accessed. The available types of queries are explained in “Parameters for the Optimization Advisor” on page 58.

3. On the Limitations page, specify how much disk space you want to allow for the summary tables and indexes that will be built. Specify if you want to allow data sampling. Also specify the maximum amount of time you want to allow for the Optimization Advisor to determine recommendations. The more space, information, and time that you specify, the more significantly your performance results will improve.
4. On the Summary Tables page, specify if you want immediate or deferred update summary tables. For more information on the update options, see "Summary table maintenance" on page 71. Specify which table space to store the summary tables and summary table indexes in. Click **Next** to have the wizard determine the recommendations for creating and refreshing the summary tables.
5. On the Summary page, enter a unique file name in both the **Create summary tables SQL script** field and the **Refresh summary tables SQL script** field. To view information, errors, or warnings about the recommendations, click the **Details** push button. To view either SQL script, click the corresponding **Show SQL** push button.
6. Click **Finish** to save the recommended SQL scripts into the file names that you specified and close the Optimization Advisor wizard.
7. Run the SQL scripts. If you are creating large summary tables, building the summary tables might require a substantial amount of time to complete. You can use the DB2 Command Center or Command Window to run the SQL scripts. To run the SQL scripts from the DB2 Command Window:
   a. Change to the directory where you saved the SQL scripts.
   b. Connect to the database of the cube model that you optimized. For example, enter: `db2 connect to MDSAMPLE`.
   c. Enter the following command:

      ```
      db2 -tvf filename
      ```

      where *`filename`* is the name of the Create summary table SQL script.

Parameters for the Optimization Advisor

You provide several types of input to the Optimization Advisor wizard. The choices that you provide for each parameter affect the summary tables that the wizard recommends and the performance improvements that you gain. Be sure to supply accurate information and to make careful decisions between cost and performance requirements.

Optimizing for particular query types

You specify the type or types of queries that you want to optimize the cube model for. The query types describe when and how DB2 relational data is typically accessed. This information helps the Optimization Advisor understand which portions of the cube model are queried most frequently.

In the figures for each of the query types in this section, the hierarchies for three dimensions of a cube model are shown. The Time dimension has a hierarchy including All Time, Year, Quarter, Month, and Day. The Region dimension has a hierarchy including All Regions, Country, State, County, City, and Zip code. The Product dimension has a hierarchy including All Products, Group, Line, and Product. The heavy line connecting Day-Zip code-Product levels denotes the slice that corresponds to the base data. The highlighted portions are examples of the parts of a cube model that each query typically accesses.

The four types of queries are explained in this list:

Drill-down

Drill-down queries usually access a subset of data that is focused at the top of a cube model as shown in Figure 23 on page 60. Queries can go to any level in the cube model. When users drill deep into one dimension, they typically stay much higher in the other dimensions. Optimizing for drill-down queries will mostly benefit queries that stay in the upper levels of the cube model. Relational OLAP (ROLAP) spreadsheet applications are usually used to perform drill-down queries. For example, a spreadsheet application user might start by accessing the revenue for all regions and all products for the year 2002. Then the user can move deeper into the data by querying for revenue by quarter in all regions and for each country. Performance is usually very important for these types of queries because they are issued real-time by a user who has to wait for the results to be processed. Optimization is based on the entire cube model and not the cubes defined for the model.

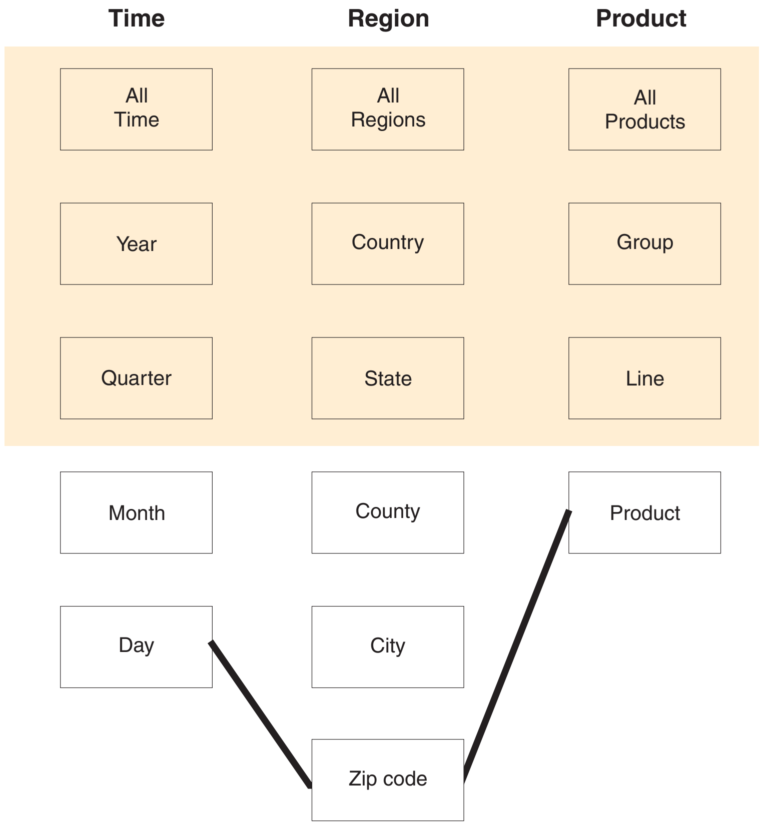


Figure 23. Drill-down style queries access a subset of data at the top of a cube model

Summary tables for drill-down queries aggregate data to the very top of the cube model. These queries are likely to show a large performance improvement. Querying for information at the top of a cube model against the base tables requires the data aggregations to be calculated repeatedly for each query. With summary tables in place, the data is already aggregated so that little to no additional calculations to be are required to satisfy the queries.

Report

Report queries are equally likely to access any part of the cube model, as shown in Figure 24 on page 61. Any cubes defined for the cube model are not significant for this type of query optimization. Report queries are often issued in batches. Query performance is usually not as critical for report queries as for drill-down queries because a user is less likely to be waiting for an immediate response to each individual query.

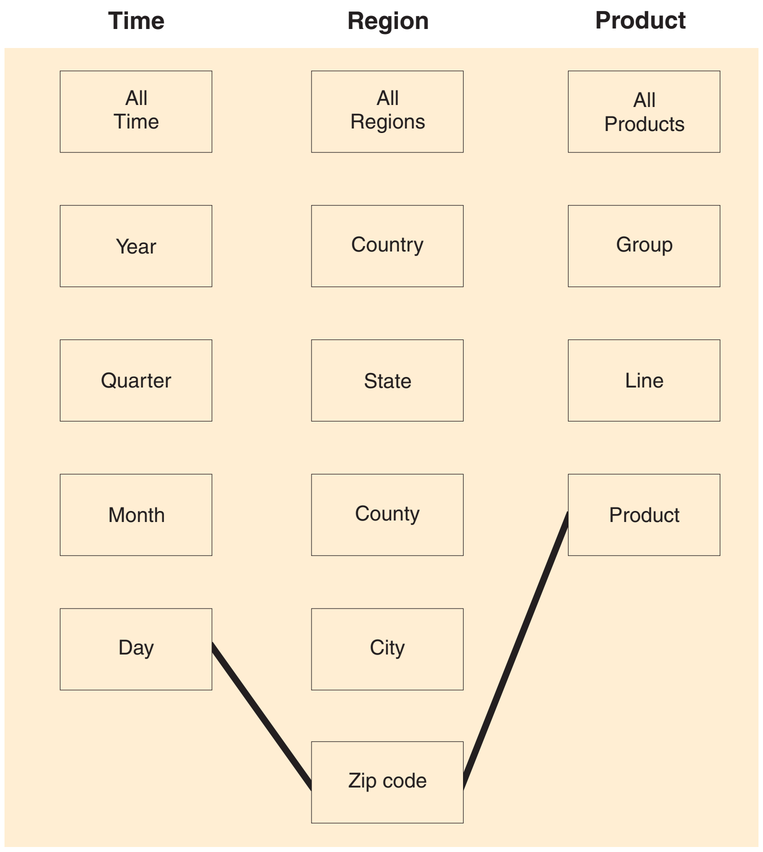


Figure 24. Report style queries are equally likely to access any part of a cube model

Optimization is based on the entire cube model and not the cubes defined for the model.

Extract

Extract queries access only the base level of a cube defined for the cube model and are used to load data into a Multidimensional OLAP (MOLAP) data store. The cubes defined for the cube model logically map to the MOLAP cubes that you want to load the data into. The Quarter-State-Line slice highlighted in Figure 25 on page 62 represents the base level of a cube defined for the cube model. Data aggregated to this level is loaded into a MOLAP application for further processing.

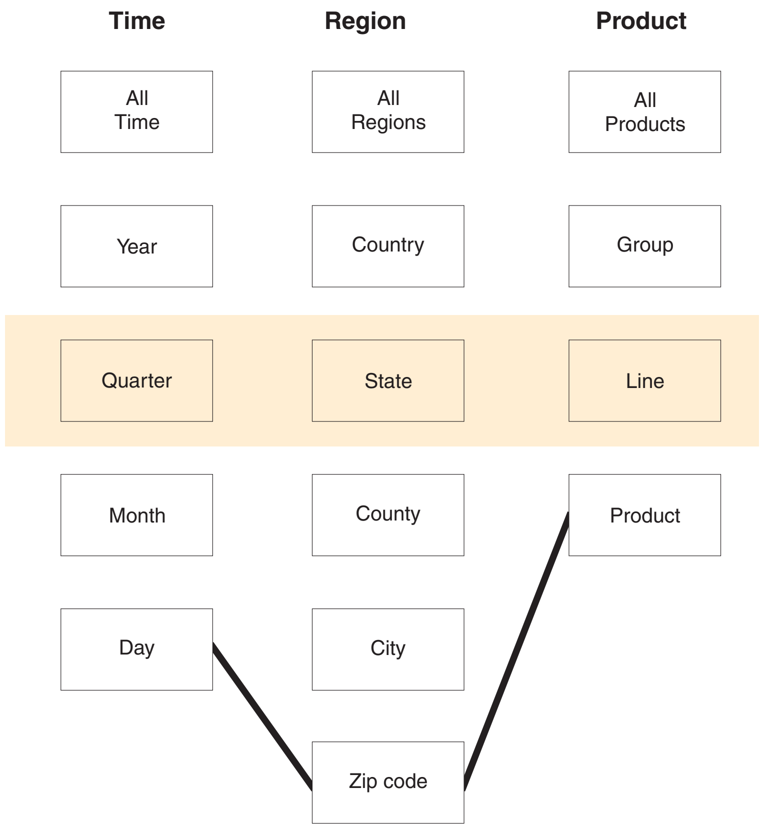


Figure 25. Extract style queries access a subset of data that corresponds to the base level of a cube defined for the cube model

Performance improvements will vary depending on how close the slice of data that corresponds to the base level of the cube is to the bottom of the cube model. The higher the slice is on the cube model, the higher the expected performance improvements are.

Drill-through

Drill-through queries access any part of a cube defined for the cube model. For drill-through queries, the cubes defined for a cube model logically map to hybrid cubes which allow a user to access MOLAP data and the lower-level data that remains in the relational database. The Quarter-State-Line slice that is highlighted in Figure 26 on page 63 represents the bottom slice of a cube that is defined for the cube model. The Quarter-State-Product slice illustrates that the query can drill past the bottom of the cube into relational data.

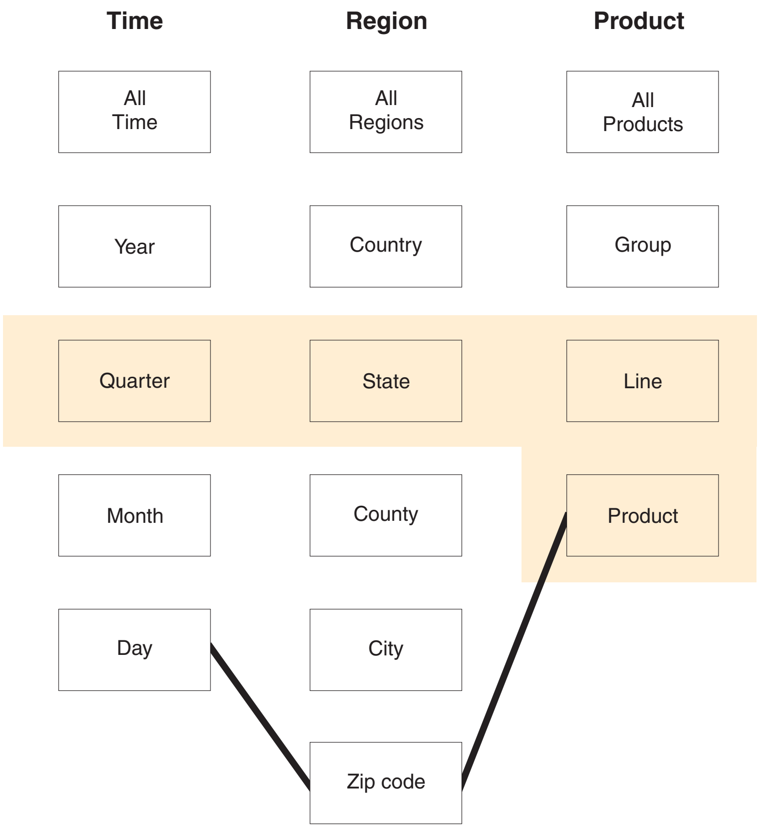


Figure 26. Drill-through style queries access a subset of cube model data that corresponds to any part of a defined cube

Drill through query optimization is based on the cubes defined for the cube model. A subset of the cube model as defined by the cubes will be optimized for.

Disk space limitations

You specify an approximate amount of disk space that can be used for summary tables. The Optimization Advisor cannot know the exact size of the summary tables until they are built, so it recommends summary tables that are as close to the specified amount of disk space as possible. The summary tables that are built might use more or less space than the user specified.

The amount of disk space that you specify is directly related to the optimization results. Increasing the disk space can increase both the number of queries with improved performance and the degree of improvement. You should consider the following factors when choosing the amount of disk space:

- The query performance levels that you want

- The number of cube models that you are optimizing for
- How critical each cube model is
- How frequently each cube model is used
- The availability and cost of the disk space

Typically, you can see significant improvement by allowing a moderate amount of disk space, such as 1% to 10% of the space currently used by the relational tables that are referenced by the cube model. Table 18 shows the relationship between the amount of disk space used for summary tables and the expected query performance improvement. When deciding how much space to provide, consider each cube model in the context of all of your metadata and base tables.

Table 18. Percentages of disk space used and the corresponding expected performance improvements.

Percentage of base tables disk space used for summary tables	Expected improvement for relevant queries
Less than 1%	Low
5%	Medium
50%	High
Unlimited	Highest

Data sampling

Data sampling is a way for the Optimization Advisor to examine the data in your cube model. This provides the Optimization Advisor with more information so that it can create the most effective set of recommendations. Recommendations created with data sampling will match the specified disk space more accurately. Without data sampling, the Optimization Advisor will analyze only the metadata and DB2 statistics to determine the recommendations.

Time limitations

The time that you specify is the maximum amount of time that the Optimization Advisor can use to determine the recommendations. The more time that you allow for the Optimization Advisor to run, the better the results. The following table provides some approximate guidelines for the amount of time you should provide to the Optimization Advisor. Performance results will vary, and you might need to allow more time than is specified in Table 19.

Table 19. Guidelines for how much time to provide to the Optimization Advisor

Database optimization scenario	Approximate time limit
Not performing data sampling	5 to 30 minutes

Table 19. Guidelines for how much time to provide to the Optimization Advisor (continued)

Database optimization scenario	Approximate time limit
Performing data sampling on a small database that is less than 10 gigabytes in size	1 hour or less
Performing data sampling on a large database that is more than 10 gigabytes in size	Several hours

Specifying table spaces

You might want to create a new table space specifically for your summary tables. You can define a tablespace with various page sizes. Summary tables are often very wide, and work best in a table space with a wide page size, such as 32 KB. Make sure that you also have a temporary space with a page size that corresponds with the table space that the summary tables are stored in.

The Optimization Advisor will restrict the row length of the recommended summary table based on the page size of the specified tablespace. Providing a tablespace with a larger page size gives the Optimization Advisor more flexibility in building the summary tables.

Example of an SQL script to create summary tables

The Optimization Advisor wizard provides a create summary tables SQL script that contains the necessary SQL commands to build one or more summary tables. Figure 27 on page 66 shows part of a sample create summary tables SQL script that creates one summary table. In SQL script sample, the summary table is called DB2INFO.MQT0000000021T01, where 21 is the cube model ID and T01 is the summary table ID. The cube model ID can be up to 10 digits long. The summary table ID identifies the summary table within the cube model. The summary table ID allows for up to 99 summary tables in one cube model. Do not change the summary table name that the Optimization Advisor wizard defines. If you change the table name, DB2 Cube Views can not keep track of the summary tables that it creates for the cube model.

```
DROP TABLE DB2INFO.MQT0000000001T01;

UPDATE COMMAND OPTIONS USING c OFF;

CREATE SUMMARY TABLE DB2INFO.MQT0000000001T01 AS
(SELECT

SUM(T2."COGS") AS "COGS",
SUM(T2."MARKETING"+T2."PAYROLL") AS "EXPENSE",
SUM(T2."MARKETING") AS "MARKETING",
SUM(T2."PAYROLL") AS "PAYROLL",
SUM(T2."SALES"-(T2."COGS"+(T2."MARKETING"+T2."PAYROLL"))) AS "PROFIT",
SUM(T2."SALES"-(T2."COGS"+(T2."MARKETING"+T2."PAYROLL")))/SUM(T2."SALES") AS "PROFITMARGIN",
SUM(T2."SALES") AS "SALES",
```

Maps to measures

```
T4."REGION" AS "REGION",
T4."DIRECTOR" AS "DIRECTOR",
T5."FAMILY" AS "FAMILY",
T5."FAMILYNAME" AS "FAMILYNAME"
```

Maps to attributes

```
FROM
"TBC"."MARKET" AS T1,
"TBC"."SALESFACT" AS T2,
"TBC"."PRODUCT" AS T3,
"TBC"."REGION" AS T4,
"TBC"."FAMILY" AS T5
```

Fact and dimension tables

```
WHERE
T1."STATEID"=T2."STATEID" AND
T3."PRODUCTID"=T2."PRODUCTID" AND
T1."REGIONID"=T4."REGIONID" AND
T3."FAMILYID"=T5."FAMILYID"
```

**Joins fact and dimension tables.
If applicable, joins dimension to dimension tables
in a snowflake schema**

```
GROUP BY GROUPING SETS (
(
T4."REGION",
T4."DIRECTOR"
),
```

Maps to one slice of cube model

```
(
T4."REGION",
T4."DIRECTOR",
T5."FAMILY",
T5."FAMILYNAME"
)))
```

Maps to another slice of cube model

```
DATA INITIALLY DEFERRED
REFRESH DEFERRED
NOT LOGGED INITIALLY;
```

Sets summary tables as refresh deferred

```
COMMENT ON TABLE DB2INFO.MQT0000000001T01 IS 'AST created for cube model TBC.TBCSalesModel';

REFRESH TABLE DB2INFO.MQT0000000001T01;

CREATE INDEX DB2INFO.IDX0000000001T0101 ON DB2INFO.MQT0000000001T01("FAMILY");
```

Creates nonclustered index

```
CREATE INDEX DB2INFO.IDX0000000001T0101 ON DB2INFO.MQT0000000001T01("REGION")
CLUSTER;
```

Creates clustered index

```
COMMIT;

REORG TABLE DB2INFO.MQT0000000001T01;

RUNSTATS ON TABLE DB2INFO.MQT0000000001T01 AND INDEXES ALL;
```

Figure 27. Part of a sample create summary tables SQL script

If more than one summary table is recommended for your cube model, your create summary tables SQL script will include a set of these statements for each summary table.

The following sections explain the statements in the sample create summary tables SQL script:

DROP TABLE statement

Each summary table that will be created is first dropped to ensure that a table with that name does not already exist. In Figure 27, the DB2INFO.MQT0000000021T01 table is dropped with the statement: `DROP TABLE DB2INFO.MQT0000000021T01;`.

CREATE TABLE statement

The script creates the summary table using a CREATE TABLE statement. This statement is the largest part of the script and includes the SELECT statement with the SELECT, FROM, WHERE, and GROUP BY clauses, and the update method definition. The summary table is created with aggregated data from the fact and dimension tables, so the CREATE TABLE statement selects the data from these tables to build the summary table.

The table name is defined in the first line of the CREATE TABLE statement: `CREATE SUMMARY TABLE DB2INFO.MQT0000000021T01`.

The SELECT clause shown in Figure 27 on page 66 has seven lines that begin with SUM. Each of these lines maps to one of the cube model's measures. For example, `SUM(T2."MARKETING"+T2."PAYROLL") AS "EXPENSE"` maps to the Expense measure with the aggregation function SUM. The cube model that the summary table is being created for has the following measures: COGS, Expense, Marketing, Payroll, Profit, ProfitMargin, and Sales. The next four lines that select a column without performing any calculations, map to attributes. For example, `T4."REGION" AS "REGION"` maps to the Region attribute. The cube model that the summary table is being created for has the following attributes: Region, Director, Family, and FamilyName.

The tables in the FROM clause are the fact and dimension tables used in the cube model. This example uses the Market, SalesFact, Product, Region, and Family tables.

The WHERE clause defines the joins between the fact and dimension tables, and each join maps to a join object in the cube model. If the cube model is based on a snowflake schema, the dimension-to-dimension joins are also included in the WHERE clause.

The GROUP BY GROUPING SETS clause maps to slices defined for the cube model. Figure 27 on page 66 shows two grouping sets, each mapping to a different slice. The groupings can include three types of metadata to define the slice:

- Level attributes from the hierarchy at the slice level
- Level attributes that are above the slice attributes
- Attributes involved in an attribute relationship with a level attribute

In the example shown, the cube model being optimized has the following hierarchies: Region [Region, State], Product [Family, SKU], and Time [Year, Quarter, Month, Day]. If a level from a hierarchy is not included in the grouping set, then the slice is at the highest level, such as All Time, All Regions, or All Products. The first slice in the GROUP BY clause is the Region-All Products-All Time slice, and includes Region and Director attributes. Region is the highest level in the Region hierarchy, so no additional level attributes are included. Director is used in an attribute relationship with Region, so it is included in the slice definition. The second slice is the Region-Family-All Time slice and includes Region, Director, Family, and FamilyName attributes. Region and Family are level attributes. Director is used in an attribute relationship with Region and FamilyName is used in an attribute relationship with Family, so both Director and FamilyName are included in the slice definition. Each of the level attributes in the slice are attributes that the SELECT clause maps to.

The last part of the CREATE TABLE statement is the update method definition. In Figure 27 on page 66, the last three lines of the CREATE TABLE statement set the summary table as refresh deferred:

```
DATA INITIALLY DEFERRED
REFRESH DEFERRED
NOT LOGGED INITIALLY;
```

If you define a summary table as refresh immediate, the statements would be:

```
DATA INITIALLY DEFERRED
REFRESH IMMEDIATE
NOT LOGGED INITIALLY;
```

CREATE INDEX statements

The Optimization Advisor wizard recommends one or more indexes for your summary table, they will be created after the summary table is created. In Figure 27 on page 66, both a clustered and nonclustered index are created. After the indexes are created, the REORG statement is used to reorganize the table according to the clustering index. In some cases, this can improve read performance on the table.

RUNSTATS statement

After all the recommended aspects of the summary table are created, the RUNSTATS statement updates the DB2 optimizer statistics that the DB2 optimizer uses to consider the summary tables and indexes for query rerouting. The last statement for each summary table is the COMMIT statement so that everything is committed to the database at once.

Testing query results

You can use the db2batch Benchmark Tool in DB2 Universal Database to benchmark your query performance results before and after you create the summary tables with the Optimization Advisor.

Procedure:

To test the performance of your queries:

1. Create an input file with the queries that you want to test separated by semicolons.
2. Enter the following command on a command line:

   ```
   db2batch -d dbname -f file_name -cli
   ```

   where *dbname* specifies the database to run the queries against, *file_name* specifies the input file with your SQL queries, and `-cli` specifies to run in CLI mode. The db2batch tool summarizes performance results and provides both arithmetic and geometric means. For syntax and options, enter `db2batch -h` on a command line. See the DB2 Information Center for more information on the db2batch Benchmark Tool and creating benchmark tests.

If you are satisfied with the performance results after creating the recommended summary tables, you do not need to do any additional performance analysis.

If your queries do not improve as much as you expected, you can run the Optimization Advisor wizard again and allow more disk space and time and enable data sampling if you did not enable it before. Allowing more disk space will most likely have the largest effect on performance. The more space that you provide for the summary tables, the more improvement you will see. If you allow the wizard to perform data sampling, the wizard can make better recommendations. Likewise, the more time that you allow for the wizard to create the recommendations, the better the recommendations are likely to be.

If you are not satisfied with the performance results because your queries do not improve at all or only very little, or if your queries perform satisfactorily for a period of time and then decrease in performance, see "Troubleshooting summary tables".

Troubleshooting summary tables

Prerequisites:

Before you use DB2EXPLAIN to verify that DB2 is using the summary tables, you should:

- Verify that the statistics are up to date on the base tables and the summary tables.
- Identify which queries are performing unacceptably if you do not already know. You can use the DB2 SQL Snapshot Monitor to capture slow queries.

Procedure:

To use DB2EXPLAIN to verify that DB2 is using the summary tables:

1. Create the explain tables. To set up the explain tables for your database, connect to the database and run the following command from your `\SQLLIB\misc` directory:

   ```
   db2 -tvf explain.ddl
   ```

2. Run the explain facility. When explain mode is turned on, the SQL queries do not run and only information requests for the explain command are processed. Run the following series of SQL commands to turn explain mode on, set the refresh age so that DB2 considers summary tables if they are refresh-deferred, run the query, turn explain mode off, and query the explain table to see if the query was rerouted:

   ```
   set current explain mode explain

   set current refresh age any

   SELECT SUM(SALES) FROM MDSAMPLE.SALESFACT

   set current explain mode no

   SELECT EXPLAIN_TIME, EXPLAIN_LEVEL AS "LEV",
       QUERYNO, STATEMENT_TEXT
   FROM EXPLAIN_STATEMENT
   WHERE STATEMENT_TEXT LIKE '%SALESFACT%'
   ORDER BY EXPLAIN_TIME
   ```

3. View the explain information and check that your rewritten query is rerouted to a summary table. For example, you might see a report like the following sample::

   ```
   2002-06-30-23.22.12.325002 O   11 SELECT SUM(SALES)
     FROM MDSAMPLE.SALESFACT
   2002-06-30-23.22.12.325002 P   11 SELECT Q3.$C0
     FROM (SELECT SUM(Q2.$C0) FROM (SELECT Q1.SALESFACT_SALES
     FROM DB2INFO.MQT0000000021T01 AS Q1) AS Q2) AS Q3
   ```

   There are two lines for one execution of the query. The line marked with an `O` is the original query that is sent to DB2. The line marked with a `P` is the query as rewritten by DB2. You can see in the rewritten query from this example that DB2 selected data from the `DB2INFO.MQT0000000021T01` summary table.

4. If the query is rerouted to the summary table you might need to run the Optimization Advisor wizard again with different options.

5. If the query is not rerouted to a summary table, determine the reason and take the appropriate action. The reasons why a query might not be rerouted to a summary table include:

Summary table does not exist

First, make sure that the summary table exists. If it does not exist, run the Optimization Advisor wizard to generate the Create summary tables SQL script. Then run the script to create the summary tables.

Refresh-deferred summary table has expired
If your summary table exists and you set it up to be refresh-deferred, you might need to update the refresh age. You can set the table's refresh age to be as large as possible and session independent by setting `(DFT_REFRESH_AGE) = 99999999999999`.

Query accesses data that is not included in the summary table
If your query is accessing data that is not in your summary table, DB2 will not reroute the query. If you added a new measure after you create your summary tables, that new measure does not exist in your summary tables. If you try to query the new measure, DB2 cannot reroute the query to the summary table because the summary table does not contain all of the data to satisfy the query. Additionally, if you try to query data that is below the cube model slice that the summary table is built for, you cannot use the summary table.

Query contains constructs that cannot be rerouted
DB2 cannot reroute queries that use some complex query constructs. Some complex constructions that inhibit DB2 from rerouting the queries are recursion and physical property functions like:

- NODENUMBER
- Outer joins
- Unions
- XMLAGG
- Window aggregation functions, which are aggregation functions specified with the OVER clause

.

Summary table maintenance

When the data in your base tables changes, you need to update your summary tables. You can update your summary tables in two different ways: refresh immediate or refresh deferred. You choose to create refresh-immediate or refresh-deferred summary tables when you run the Optimization Advisor wizard. The choice that you make affects the update setting for the tables and the refresh summary tables SQL script. For both options, you need to run the refresh summary tables script as part of your normal database maintenance schedule. Running the refresh script can require significant time and processing resources. Make sure you allocate enough time in your maintenance batch window to complete the updates.

Refresh-immediate

Refresh-immediate summary tables are kept closely synchronized with your base tables. DB2 tracks the changes to the base tables so that it can incrementally update the summary tables by changing only the portion of the summary tables that corresponds to the changed portion of the base tables. If it is important to you that the summary table data be kept in unison with your base tables, use the refresh-immediate option. Refresh-immediate might be a good choice if, for example, your base tables are updated with weekly sales data, and users complete weekly reports reflecting the updated sales data.

If you commonly have many changes scattered throughout your base tables, refresh- immediate is probably not the best choice because it can require significant overhead for DB2 to track the changes and individually perform the update statements to aggregate the changes again.

If you update your base tables using regular SQL statements, such an INSERT, UPDATE, and DELETE, DB2 automatically synchronizes the affected summary tables after you change your base tables. However, if you update your base tables using the DB2 LOAD or IMPORT commands, you need to manually trigger the synchronization by running the refresh script after you complete the update.

Immediate update cannot be used in all situations, and the Optimization Advisor wizard might recommend the deferred option if necessary.

Refresh-deferred

Refresh deferred summary tables are usually updated less frequently than refresh immediate because you have to manually synchronize the summary tables with the base tables. The summary tables are based on a snapshot of the data at the time that they are created. Each update re-creates the summary table based on the current data, but has no knowledge of how the data changed since the summary table was last created.

Refresh deferred is a good choice when you are making significant changes throughout the corresponding base tables or if you are updating data more quickly than you need to access it. For example, if your sales data is updated weekly but you need to create reports on a quarterly basis only, you can use the refresh-deferred option and rebuild your summary tables each quarter before running your report.

Dropping a summary table

DB2 Cube Views does not drop the associated summary tables when you drop a cube model. If you do not use the summary tables for any other purpose, you can drop the tables to free up disk space. Summary tables are a type of table, and can be dropped using the normal DB2 procedures using the Control Center or the command line. Any associated indexes are also dropped with the summary table.

The summary tables are defined in the DB2INFO schema. The summary table name includes the cube model ID. For example, a summary table might be named DB2INFO.MQT0000000021T01, where 21 is the cube model ID and T01 uniquely identifies the summary table within the cube model. The cube model ID can be up to 10 digits long. The summary table ID allows for up to 99 summary tables in one cube model.

Procedure:

To drop a summary table from a command line, enter `DROP TABLE` *`table_name`*.

Chapter 4. Troubleshooting the IMPORT or the CREATE and ALTER operations

This chapter describes how to avoid a validation error in two sample situations when using the IMPORT or the CREATE and ALTER operations.

Avoiding a validation error by using the IMPORT or the CREATE and ALTER operation order

In addition to using the Import wizard in OLAP Center to import metadata objects to DB2, you can use the Multidimensional Services IMPORT operation or a combination of the CREATE and ALTER operations. The IMPORT operation internally uses a combination of CREATE and ALTER to import each specified object. OLAP Center uses the IMPORT operation to import metadata objects.

When you use the Multidimensional Services stored procedure, you pass an in-memory XML document to the stored procedure. In the OLAP Center Import wizard, you specify an XML file for the import operation. The XML structure that is contained in the input file to the Import wizard can be the same XML document passed in-memory to the Multidimensional Services stored procedure. You also specify import options. The import options are specified in a second in-memory XML structure for Multidimensional Services, and in the Import wizard in the OLAP Center. The import file or in-memory XML document needs to specify any objects that you want to create or alter. You do not need to specify existing, unchanged objects in the import XML. However, this section includes all of the metadata objects in the descriptions of the import XML to illustrate all of the object relationships that are involved.

A validation error can occur when you use one operation to import or create and alter objects that share referenced objects. Multidimensional Services requires that metadata objects are processed and validated in a particular order, such that referenced objects must appear before referencing objects. Enforcing this requirement can create a situation in which a referenced object knows about a new object before the referencing object does, which is not allowed.

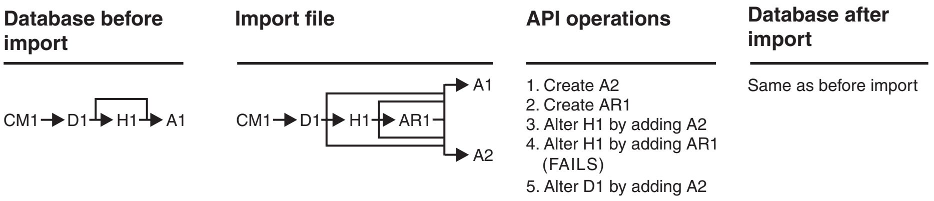
This section describes two sample error situations and how to avoid the validation error.

Shared attribute referenced by a dimension

A dimension references hierarchies, attribute relationships, and attributes. The validation ordering rules require that these objects are processed in the following order for any operation: attribute, attribute relationship, hierarchy, dimension. When you import or alter a dimension, validation errors can occur because of the way that dimensions share attributes with other referenced objects such as hierarchies and attribute relationships. For example, suppose that a dimension (D1) exists in a cube model (CM1), and references a hierarchy (H1) that in turn references an attribute (A1). If the import file contains a dimension (D1) in a cube model (CM1) that refers to a hierarchy (H1), and H1 refers to an attribute relationship (AR1) that refers to two attributes (A1 and A2), you will encounter a validation error when you import. To import the metadata as described in the import file, Multidimensional Services sequentially performs the following operations and validates the state of the metadata after each operation:

1. Create A2
2. Create AR1
3. Alter H1 by adding A2
4. Alter H1 by adding AR1 (Fails)
5. Alter D1 by adding A2

A validation error occurs between steps 4 and 5 because the state of the metadata after step 4 is invalid. The validation rules require that attribute relationships used by a hierarchy in a dimension must refer only to attributes used in the dimension. However, because Multidimensional Services performs operations according to a specific order of metadata objects, it is not possible to complete this import scenario in one operation. Because an error is encountered, the metadata is not changed. The following diagram illustrates this error scenario.



To successfully complete this scenario, you must divide the import operation into two separate operations. For example, you can first use an import file with a dimension (D1), that exists in a cube model (CM1), and references a hierarchy with one attribute (A1). D1 also references a second attribute (A2), and a floating attribute relationship (AR1) references both A1 and A2. To import the metadata as described in the import file, Multidimensional Services sequentially performs the following operations, which can all be successfully validated:

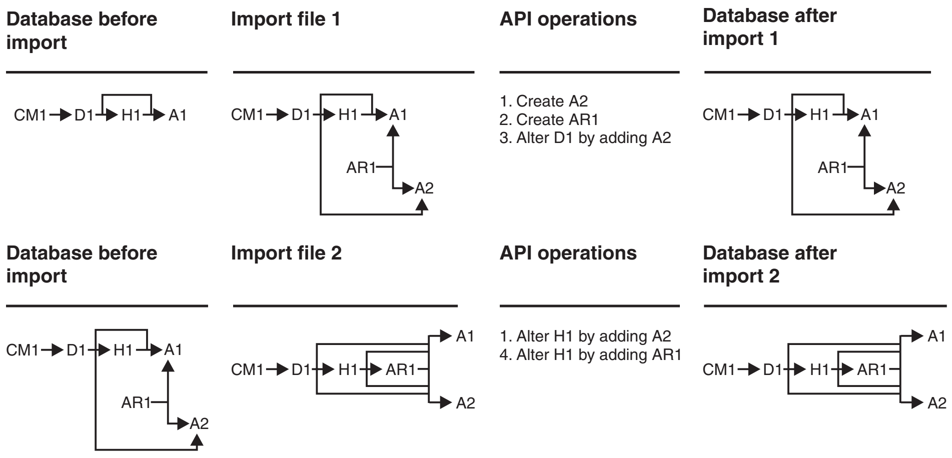
1. Create A2
2. Create AR1

3. Alter D1 by adding A2

This import operation is completed successfully and D1 now references A2 so that in your next import operation you can successfully add A2 and AR1 referencing A1 and A2 to H1. The second import file contains a dimension (D1), in a cube model (CM1) which refers to a hierarchy (H1), and H1 refers to an attribute relationship (AR1) that refers to two attributes (A1 and A2). To import the metadata as described in the import file, Multidimensional Services needs to sequentially perform only the following operations:

1. Alter H1 by adding A2
2. Alter H1 by adding AR1

The following diagram illustrates the two-step approach used in this scenario.



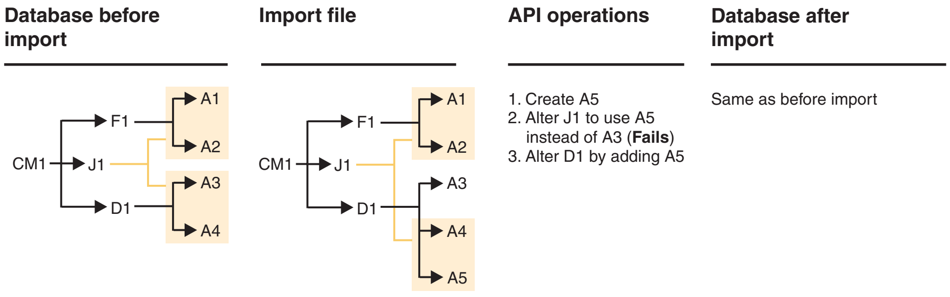
Shared attributes referenced by an altered join

Attributes can also be referenced by facts objects, dimensions, and facts-to-dimension joins. The validation ordering rules require that these objects are processed in the following order for an operation: attribute, join, dimension, facts object. When you import or alter a join object, validation errors can occur because of the way that joins share the attributes in the join pairs with other objects such as facts objects and dimensions.

For example, suppose that a facts object (F1), a dimension (D1), and a facts to dimension join (J1) exist in a cube model (CM1). F1 references two attributes (A1 and A2), D1 references another two attributes (A3 and A4), and J1 joins F1 and D1 together in the set of two attribute pairs: A1 and A3, and A2 and A4. You might want to change the metadata by adding an additional attribute, A5 to D1, and altering J1 so that it joins on the attribute pairs A1 and A5, and A2 and A4. However, if you attempt to complete these changes in a single operation, you will encounter a validation error. To import the metadata as described, Multidimensional Services sequentially attempts the following operations and validates the state of the metadata after each operation:

1. Create A5
2. Alter J1 to use A5 instead of A3 (Fails)
3. Alter D1 by adding A5

A validation error occurs between steps 2 and 3 because the the metadata is in an invalid state after step 2. The validation rules require that all attributes on one side of a facts-to-dimension join must exist in the facts object's attribute list, and all attributes on the other side of the join must exist in the dimension's attribute list. Because Multidimensional Services performs operations individually and sequentially according to a specific order, the A5 attribute on one side of J1 is not yet included in D1's attribute list, which makes the cube model invalid. The metadata is not changed because the operation fails. This import scenario cannot be completed in one operation. The following diagram illustrates this error scenario.



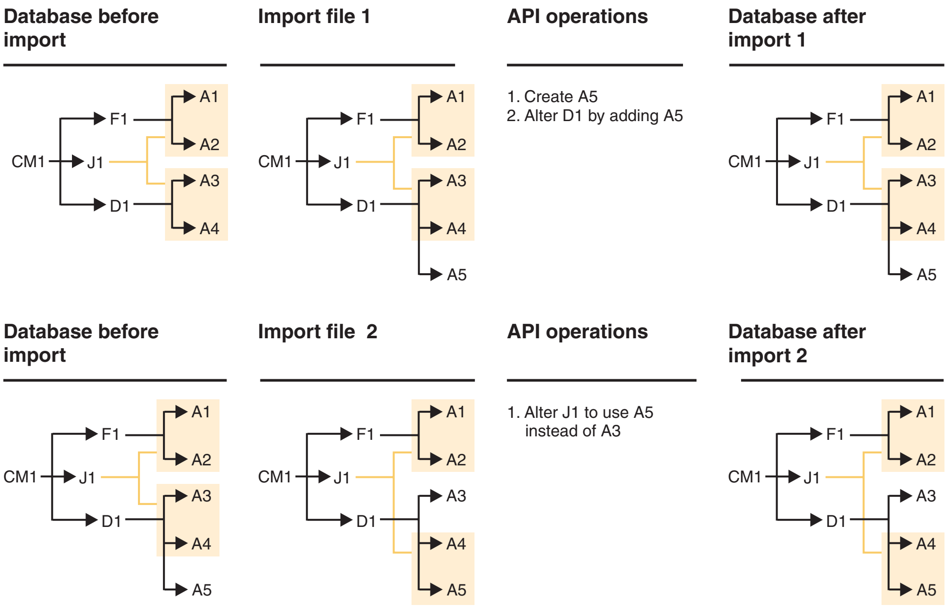
To successfully complete this scenario, you must divide the import operation into two separate operations. For example, you can first use an import file that adds an attribute (A5) to the dimension (D1), but does not alter the join (J1) between D1 and the facts object (F1). To import the metadata as described in the import file, Multidimensional Services sequentially performs the following operations, which can all be successfully validated:

1. Create A5
2. Alter D1 by adding A5

This import operation is completed successfully and D1 now references A5 so that in your next import operation you can successfully alter J1 to join F1 and D1 using A5 instead of A3 in one of the join attribute pairs. Because A5 already exists and is referenced by D1 after the first import operation, Multidimensional Services needs to sequentially perform only the following operation:

1. Alter J1 to use A5 instead of A3

The following diagram illustrates the two-step approach used in this scenario.



Appendix A. API Reference

DB2 Cube Views offers Multidimensional Services, which is an application programming Interface (API) that provides programmatic access to metadata stored in DB2 Cube Views. Using the API, applications can interact with metadata using DB2 Cube Views metadata objects without having to interact with relational tables and joins. Applications using the API can create and modify metadata objects that model multidimensional and OLAP constructs in a data warehouse.

The API uses SQL with ODBC, the DB2 CLI, and JDBC, and the API makes extensive use of XML.

API overview

The API for DB2 Cube Views provides access to the metadata stored in the system catalog tables of a DB2 database. Figure 28 shows how data and metadata is exchanged through the API.

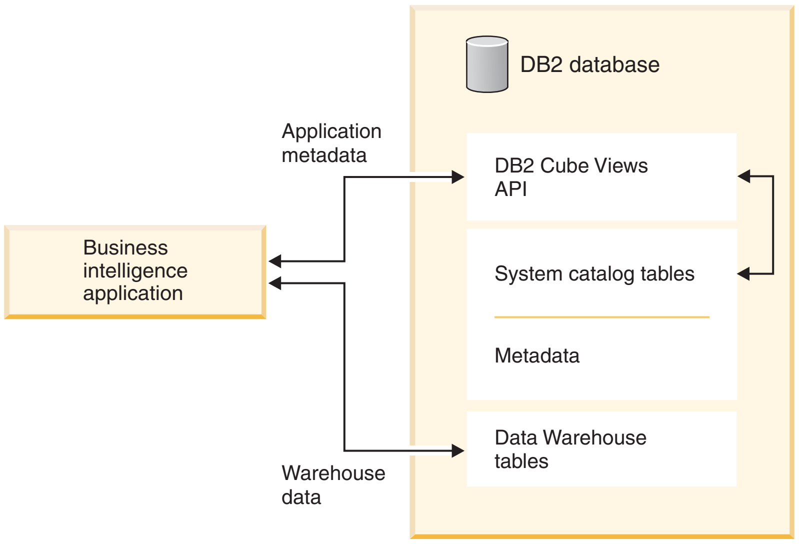


Figure 28. Data exchange though the DB2 Cube Views API

The API is composed of a single stored procedure registered to a DB2 database. This stored procedure accepts input and output parameters in which you can express complex metadata and metadata operations. The API parameter format is defined by an XML schema.

To effectively use the API, you must understand:

1. The DB2 Cube Views metadata objects and their relationships
2. The API parameter format, which is an XML Schema
3. XML, including the parsing and generation of XML, and schemas
4. DB2 application programming involving the calling of stored procedures
5. Error handling in DB2 Cube Views.

DB2 Cube Views stored procedure

The stored procedure is called **md_message** and it processes parameters expressed in the DB2 Cube Views parameter format. The procedure extracts operation and metadata information from the input parameters, and then it performs the requested metadata operations. The procedure generates output parameters that contain the execution status (success or failure) of requested operations, in addition to containing metadata information, depending on the operation.

The DB2 Cube Views stored procedure, is implemented as a DB2 stored procedure. It can be used by any application that makes use of any of DB2's programming interfaces. The name of the stored procedure is case insensitive, while the name and contents of the stored procedure's parameters are case sensitive. The syntax of **md_message** and a prototype are shown in the following example:

```
Syntax:     call md_message (request, metadata, response)
Prototype:  md_message (request  IN   CLOB(1M),
                        metadata INOUT  CLOB(1M),
                        response  OUT   CLOB(1M))
```

The *request*, *metadata*, and *response* parameters are of type CLOB, which is a DB2 data type. An application populates the *request* parameter with a description of the operation to be performed, and it can optionally populate the *metadata* parameter with the metadata that the operation should act upon. After consuming the input parameters, **md_message** returns the status of the operation in the *response* parameter, and returns the requested metadata in the *metadata* parameter. The *metadata* parameter is used for both input and output of metadata. DB2 handles the transfer of parameter structures between business intelligence applications and the **md_message** stored procedure on the database server.

The size of the CLOB arguments can vary. 1M is the default, which is used by the `sqllib/misc/db2mdapi.sql` script. You can recatalog the stored procedure with any size for the CLOB parameters up to 2 GB. When you increase the size of the parameter, more memory is used by the stored procedure at run time because output parameter buffers are preallocated to the cataloged size when the stored procedure is started. If the size is too small, input and output parameters might have their data truncated if they exceed the cataloged sizes.

For a description on how to write programs that use DB2 stored procedures, see the *DB2 Application Development Guide*. For a description of the issues related to programming with the API, refer to section "Application programming notes" on page 105.

The following example shows how to call the DB2 Cube Views stored procedure from an embedded SQL application:

```
// Standard declarations
// ...

// Include the Communication Area to access error details
EXEC SQL INCLUDE SQLCA;

// SQL declarations of host variables that will be used for calling the
// DB2 Cube Views stored procedure
EXEC SQL BEGIN DECLARE SECTION;

// Allocate CLOB for the request parameter
SQL TYPE is CLOB(1M)       request;

// Allocate CLOB for the metadata parameter
SQL TYPE is CLOB(1M)       metadata;

// Allocate CLOB for the response parameter
SQL TYPE is CLOB(1M)       response;

EXEC SQL END DECLARE SECTION;

// Connect to database and other application initializations
// ...

// Populate the request parameter structure with the operation
strcpy(request.data, "<request><describe> ... </describe></request>");

// string length with end-of-string
request.length = strlen(request.data) + 1;

// Populate the metadata parameter structure with the metadata
strcpy(metadata.data, "");
```

Figure 29. Example of an embedded SQL application calling the md_message stored procedure (Part 1 of 2)

```
// string length with EOS
metadata.length = strlen(metadata.data) + 1;

// Call DB2 Cube Views stored procedure
EXEC SQL CALL "DB2INFO.MD_MESSAGE"(:request,:metadata,:response);

// Check that the stored procedure has returned without errors
if (sqlca.sqlcode)
{
// error checking using sqlaintp()
}

// Process response parameter structure to determine success of operation
// ...

// Process metadata parameter structure to extract requested metadata
// ...

// Disconnect from database and other application terminations
// ...
```

Figure 29. Example of an embedded SQL application calling the md_message stored procedure (Part 2 of 2)

Parameters

The API for DB2 Cube Views offers three types of metadata operations: retrieval, modification, and administration. Each type includes one or more operations, and each operation has its own set of parameters. Before understanding the metadata operations, which are described in “Metadata operations” on page 87, it is useful to first understand how the parameters work.

The parameter format defines the standard by which metadata operations and objects are represented and exchanged between business intelligence applications and DB2 Cube Views. The parameter format uses XML to represent DB2 Cube Views metadata operations and objects. The XML Schema defines the parameter format.

Input and output parameters

There are two kinds of input parameters for each metadata operation:

Input parameter	Parameter content
request	Contains the description of one operation being requested of the stored procedure. The operation description contains options that affect its behavior and scope.

metadata	Contains representations of metadata objects that will be used with the operation described in the *request* parameter.

Each metadata operation also has two kinds of output parameters:

Output parameter	**Parameter content**
response	Contains all the results of the operation performed by the stored procedure, except metadata objects.
metadata	Contains representations of metadata objects that were requested by the operation described in the *request* input parameter.

Figure 30 shows how the parameters work.

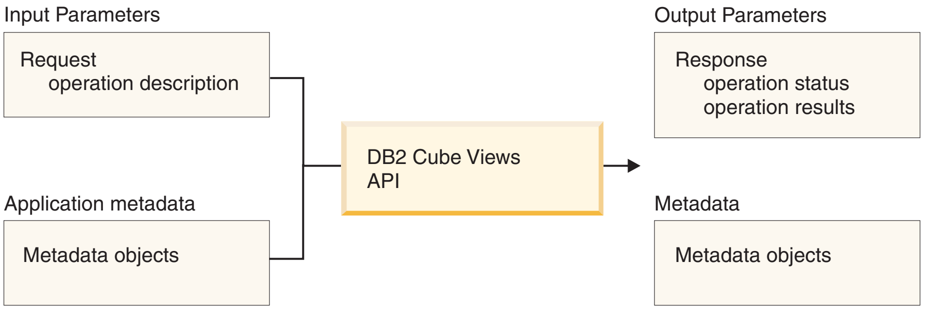


Figure 30. API parameters

Sample parameters

The following examples show how you can structure parameters in the three types of metadata operations. In these examples portions of the XML structures are excluded, but are represented with an ellipsis (...).

Retrieval operation

The following examples show how a retrieval operation called `describe` is structured. See "Retrieval operation" on page 88 for more information about the `describe` operation. The *metadata* parameter is empty on input, but populated on output.

Request:

```
<olap:request xmlns:olap="http://www.ibm.com/olap" ... >
   <describe objectType="cube" recurse="no">
      <restriction>
         <predicate property="schema" operator="=" value ="myschema"/>
      </restriction>
   </describe>
```

```
</olap:request>

<olap:metadata xmlns:olap="http://www.ibm.com/olap" ... />
```

Response:

```
<olap:response xmlns:olap="http://www.ibm.com/olap" ... >
   <describe>
      <status id="0" text="Operation completed
       successfully."type="informational"/>
   </describe>
</olap:response>

<olap:metadata xmlns:olap="http://www.ibm.com/olap" ... >
   <cube name="cube1" schema="myschema" ... > ... </cube>
      ...
   <cube name="cubeN" schema="myschema" ... > ... </cube>"
</olap:metadata>
```

Modification operations

The following examples show how a modification operation called `create` is structured. See "Modification operations" on page 89for more information about the `create` operation and other modification operations. The *metadata* parameter is populated on input, but empty on output.

Request:

```
<olap:request xmlns:olap="http://www.ibm.com/olap" ... >
   <create/>
</olap:request>

<olap:metadata xmlns:olap="http://www.ibm.com/olap" ... >
   <attribute name="LocationID" ... > ... </attribute>
   <attribute name="Country" ... > ... </attribute>
   <attribute name="State" ... > ... </attribute>
   <attribute name="City" ... > ... </attribute>
   <dimension name="Location" ... type="regular">
      <attributeRef name="LocationID" ... </attributeRef>
      <attributeRef name="Country" ... </attributeRef>
      <attributeRef name="State" ... </attributeRef>
      <attributeRef name="City" ... </attributeRef>
        ...
   </dimension>
</olap:metadata>
```

Response:

```
<olap:response xmlns:olap="http://www.ibm.com/olap" ... >
   <create>
      <status id="0" text="Operation completed
       successfully."type="informational"/>
   </create>
</olap:response>

b<olap:metadata xmlns:olap="http://www.ibm.com/olap" ... >
```

Administration operation

The following examples show how an administration operation called `validate` is structured. See "Administration operation" on page 94 for more information about the `validate` operation.

Request:

```
<olap:request xmlns:olap="http://www.ibm.com/olap" ... >
   <validate objectType="cube" mode="base">
      <restriction>
         <predicate property="schema" operator="=" value ="myschema"/>
      </restriction>
   </describe>
</olap:request>

<olap:metadata xmlns:olap="http://www.ibm.com/olap" ... />
```

Response:

```
<olap:response xmlns:olap="http://www.ibm.com/olap" ... >
   <validate>
   <status id="1" text="...Additional information
     returned."type="informational"/>
   <info><message id="6299" text="At least one
     database view was found during validation."
     type="warning"/></info>
   </validate>
</olap:response>

<olap:metadata xmlns:olap="http://www.ibm.com/olap" ... >
```

Metadata operations

Table 20 shows the three types of metadata operations that can be requested and responded to. Each type has one or more operations:

Table 20. List of metadata operations

Types of metadata operation	Metadata operations
Retrieval	"Describe" on page 88
Modification	"Create" on page 90 "Alter" on page 90 "Rename" on page 91 "Drop" on page 92 "Import" on page 93
Administration	"Validate" on page 94

Retrieval Operations

Retrieval operations deal with querying and retrieving metadata from DB2 Cube Views. These operations involve arguments that restrict the scope of a data query and retrieval. The results generated from retrieval operations typically involve metadata object information.

Modification Operations
Modification operations deal with creating, updating, and deleting metadata objects. The operands for these operations involve the metadata being modified. The results generated from modification operations contain operation status indicators describing the success or failure of the requested operations.

Administration Operations
Administration operations deal with validating the integrity of the DB2 Cube Views metadata. These operations involve arguments that restrict the scope of validation actions. The results generated from validation operations contain operation status indicators describing the success or failure of the requested operations.

In the following section, objects that appear in the *metadata* parameter are indicated with an asterisk (*).

Retrieval operation

DB2 Cube Views includes one retrieval operation: Describe.

Describe

This operation performs retrievals of metadata object information. This operation returns information for one or more metadata objects of the specified *objectType* (for example, single dimension object, a set of dimension objects, a set of objects including all object types). In recursive mode, this operation also returns information for all other metadata objects encountered during the traversal of all object-association paths beginning from the set of objects satisfying the *objectType* specification. Table 21 on page 89 lists the components in a Describe request parameter, and Table 22 on page 89 lists the components of the Describe response parameter.

Table 21. Describe Request Components

Name	Type	Data type	Valid Values	Description
objectType	argument	XML string	`all` `cubeModel` `cube` `cubeFacts` `cubeDimension` `cubeHierachy` `dimension` `facts` `hierarchy` `attributeRelationship` `join` `measure` `attribute`	Types of DB2 Cube Views metadata objects being retrieved. One of the valid values can be specified for this parameter.
restriction	argument [optional]	XML string	See "Operation arguments" on page 96.	Limits the scope of a metadata retrieval. This is analogous to predicates in SQL.
recurse	argument	XML string	`yes` `no`	Enables/disables recursive retrieval of metadata objects.

Table 22. Describe Response Components

Name	Type	Data type	Valid Values	Description
status	status message	DB2 Cube Views message structure	See section "Message structure" on page 101.	Message indicating status of requested operation.
object*	retrieval results	XML element	See "Metadata object retrieval results" on page 102.	Requested metadata data objects. This value is the empty string if an error was encountered during the operation.

Modification operations

DB2 Cube Views includes five modification operations:

- Create

- Alter
- Rename
- Drop
- Import

As part of carrying out a modification operation, the stored procedure ensures that objects are complete. They also validate referential integrity between objects. Metadata validations can also be carried out as independent metadata operations; see "Validate" on page 94 for more information.

Create

This operation creates metadata objects. This operation accepts one or more metadata object operands and creates these objects in DB2 Cube Views in the order that they are passed to the operation. Sequences of objects passed to this operation can be composed of objects of different type.

Objects passed to this operation can optionally reference other objects. If references exist between objects, they must be reflected in the ordering of the objects. For example, if `MyObject` references `YourObject`, then `YourObject` must be passed to the operation before `MyOblect`. For more information about how metadata objects can reference each other, see "Metadata object format" on page 102.

This operation validates each object. Errors are returned if the object being created already exists, or if an object referenced by the object being created does not already exist. If an input object specifies a schema that does not exist, the operation creates it, if you have sufficient authority in the database.

Table 23. Create Request Components

Name	Type	Data type	Valid values	Description
object*	operand	XML element	See "Operation operands" on page 100.	Objects being created.

Table 24. Create Response Components

Name	Type	Data type	Valid values	Description
status	status message	DB2 Cube Views message structure	See section "Message structure" on page 101.	Message indicating status of requested operation.

Alter

This operation updates metadata object information. It accepts one or more metadata object operands and updates their object counterparts in the metadata catalog tables. Objects are updated in the order that they are passed to the operation. Sequences of objects passed to this operation can be composed of objects of different type.

Objects passed to this operation can optionally reference other objects. If references exist between objects, they must be reflected in the ordering of the objects. For more information about how metadata objects can reference each other, see “Metadata object format” on page 102.

This operation cannot update the schema or the name of an object. Object names can be changed with the *rename* operation.

This operation performs validate of each object. Errors are returned if the object being updated does not exist.

Table 25. Alter Request Components

Name	Type	Data type	Valid Values	Description
object*	operand	XML element	See “Operation operands” on page 100.	Objects being updated.

Table 26. Alter Response Components

Name	Type	Data type	Valid Values	Description
status	status message	DB2 Cube Views message structure	See section “Message structure” on page 101.	Message indicating status of requested operation.

Rename

This operation renames a single DB2 Cube Views metadata object identified by its current schema and name. Only the name of an object might be changed; the schema of an object cannot be changed. The operation can rename objects even if they are currently being referenced by other metadata objects.

Table 27. Rename Request Components

Name	Type	Data type	Valid Values	Description
objectType	argument	XML string	cubeModel cube cubeFacts cubeDimension cubeHierachy dimension facts hierarchy attributeRelationship join measure attribute	Type of DB2 Cube Views metadata object being renamed. One of the valid values can be specified for this parameter.
currentRef	operand	DB2 MMM object reference	See “Operation operands” on page 100.	Current schema and name of the metadata object being renamed.
newRef	operand	DB2 MMM object reference	See “Operation operands” on page 100.	New schema and name of the metadata object being renamed.

Table 28. Rename Response Components

Name	Type	Data type	Valid Values	Description
status	status message	DB2 Cube Views message structure	See section “Message structure” on page 101.	Message indicating status of requested operation.

Drop

This operation deletes metadata objects from DB2 Cube Views. This operation deletes one or more metadata objects depending on the *objectType* and *restriction* specified. If the object being dropped is currently being referenced by another metadata object, an error is returned.

Table 29. Drop Request Components

Name	Type	Data type	Valid Values	Description
objectType	argument	XML string	See "Operation arguments" on page 96.	Types of metadata object being deleted. One of the valid values can be specified for this parameter.
restriction	argument [optional]	XML string	See "Operation arguments" on page 96.	Limits the scope of a metadata deletion. This is analogous to predicates in SQL.

Table 30. Drop Response Components

Name	Type	Data type	Valid Values	Description
status	status message	DB2 Cube Views message structure	See section "Message structure" on page 101.	Message indicating status of requested operation.

Import

This operation creates metadata objects or reports the existence of metadata objects in the metadata catalog. While creating metadata objects, this operation behaves similar to the *create* operation, except for the manner with which it deals with the presence of preexisting metadata objects. There are various modes of operation for import. What defines these modes are the actions that will be taken against a metadata object upon the determination of whether or not the object already exists in the metadata catalog.

Depending on the mode being run, errors are returned if the object being created already exists, or if an object referenced by the object being created does not already exist. Similar to the *create* operation, if an input object specifies a schema that does not exist, the operation creates it, if you have sufficient authority in the database.

Validation of each object is performed implicitly by this operation.

See "Create" on page 90 for details on metadata object and reference ordering during metadata object creation.

See "Operation arguments" on page 96 for a detailed description of the various operation modes.

Table 31. Import Request Components

Name	Type	Data type	Valid Values	Description

mode	argument	XML string	`create new - ignore collisions`
			`create new - replace collisions`
			`create new - abort on collision`
			`report new - report collisions`
				Defines the actions to be taken for new and existing objects being imported.

object*	operand	XML element	See "Operation operands" on page 100.	Objects being imported.

Table 32. Import Response Components

Name	Type	Data type	Valid Values	Description
status	status message	DB2 Cube Views message structure	See section "Message structure" on page 101.	Message indicating status of requested operation.
newList	reference list	XML element	See the description for *mode* in "Operation arguments" on page 96.	List of name-schema pairs referencing those objects deemed ″new″ by the operation.
collisionList	reference list	XML element	See the description for *mode* in "Operation arguments" on page 96.	List of name-schema pairs referencing those objects deemed ″collisions″ by the operation.

Administration operation

DB2 Cube Views includes one administration operation: Validate.

Validate

This operation checks the validity of one or more metadata objects. Validity is defined as an object's conformance to the DB2 Cube Views object rules. The objects to be acted upon by this operation are specified using the *objectType* and *restriction* arguments . The extent of the validation actions to be performed is specified using the *mode* argument.

The following are examples of the types of check performed as part of the validate operation:

1. Completeness of metadata object information
2. Referential integrity between metadata objects
3. Existence of referenced relational table columns and views
4. Correctness of SQL Expression stored in metadata objects (i.e., attributes and measures)
5. Specialization/subset relationships between various objects (i.e., cube model and cube, dimension and cube dimension, facts and cube facts, hierarchy and cube hierarchy)

This operation terminates upon the first occurrence of an invalid metadata object. Information describing the validation violation is returned by this operation when a violation is encountered. Validation of each object is also performed implicitly by the *create*, *alter*, and *import* operations. See "Operation arguments" on page 96 for a detailed description of the various operation modes.

Table 33. Validate Request Components

Name	Type	Data type	Valid Values	Description
objectType	argument	XML string	See "Operation arguments" on page 96.	Types of DB2 Cube Views metadata object being validated. One of the valid values can be specified for this parameter.
restriction	argument [optional]	XML string	See "Operation arguments" on page 96.	Limits the scope of a metadata validation. This is analogous to predicates in SQL.
mode	argument	XML string	`base` `cubeModel completeness` `optimization`	Defines the extent of the validation actions to be performed.

Table 34. Validate Response Components

Name	Type	Data type	Valid Values	Description
status	status message	DB2 Cube Views message structure	See section "Message structure" on page 101.	Message indicating status of requested operation.

Table 34. Validate Response Components (continued)

Name	Type	Data type	Valid Values	Description
info	message list	List of message structures	See section "Message structure" on page 101.	List of messages describing the warnings and errors generated by the validate operation.

Operation arguments

Various arguments are available for each metadata operation. These arguments tailor an operation's behavior to its particular application. DB2 Cube Views offers five arguments for the metadata operations:

- objectType
- recurse
- restriction
- mode (for the import operation)
- mode (for the validate operation)

objectType

This argument specifies the type of metadata objects involved in the requested operation. The following object types correspond directly to the DB2 Cube Views metadata object model.

- all
- cubeModel
- cube
- cubeDimension
- cubeFacts
- cubeHierarchy
- dimension
- facts
- hierarchy
- attributeRelationship
- join
- measure
- attribute

recurse

This parameter controls whether or not a given operation performs in a recursive manner. In non-recursive mode, a given operation performs its actions on only the metadata objects directly matching the *objectType* and *restriction* parameter specifications. In recursive mode, a given operation starts with the non-recursive mode set of metadata objects and additionally performs its actions on all other metadata objects encountered during the traversal of all object-association paths beginning from the non-recursive mode set of objects. The recurse options available are as follows:

- yes
- no

As an example, a non-recursive operation might return a list of dimensions; whereas, a recursive operation might return not only a list of dimensions, but also all other objects (of different types) referenced by those dimensions and in turn objects referenced by those objects.

restriction

This argument enables a metadata operation to be restricted or limited in its scope. The use of this argument is analogous to the use of predicates in a SQL query. Restrictions are expressed in XML using the <restriction> and <predicate> tags as defined by the DB2 Cube Views XML Schema. A predicate element contains a *property* attribute, an *operator* attribute, and a *value* attribute.

Restrictions can be based upon those object properties common to all of the metadata objects, as well as upon the relationships between metadata objects.

Property-based predicates require that the *property* attribute associated with a predicate element must specify one of the following:

- name
- schema

The *operator* attribute associated with a predicate element must specify the equal sign (=).

The *value* attribute associated with a predicate element is the string representation of the value to be compared against the property specified by the *property* attribute.

Refer to “Sequencing the operation steps” on page 102 for a description of how the *restriction* parameter relates to the overall sequencing of operation steps.

Example: This example restricts the scope of an operation to the objects in the `ABC` schema:

```
<restriction>
      <predicate property="schema" operator="=" value="ABC">
</restriction>
```

mode (for import)

This argument sets the mode for the *import* operation. The following table describes the various modes available.

In this section, the term ″collision″ refers to the situation where an object passed into the *import* operation as input already exists within the metadata catalog.

Table 35. Import modes

Mode	Description	Returned Reference Lists
Create new - ignore collisions	• Input objects that do NOT collide are created. • Input objects that collide are NOT created. • Preexisting objects are NOT altered. • Errors are NOT generated by collisions.	**newList** Contains the name-schema pairs for the objects successfully created. **collisionList** Contains the name-schema pairs for the objects involved in a collision, and therefore not created.
Create new - replace collisions	• Input objects that do NOT collide are created. • Input objects that collide replace preexisting objects. • Preexisting objects are replaced by the input objects. • Errors are NOT generated by collisions.	**newList** Contains the name-schema pairs for the objects successfully created. **collisionList** Contains the name-schema pairs for the objects involved in a collision, and therefore used to replace preexisting objects.

Table 35. Import modes (continued)

Mode	Description	Returned Reference Lists
Create new - abort on collision	• Input objects are created only if no collisions exist for the entire operation. • In the case of a collision, no objects are created as part of the operation. • Preexisting objects are NOT altered. • Errors are generated by collisions.	**newList** Contains the name-schema pairs for the objects either successfully created, or the non-colliding objects, which were not created, in the situation where collisions occurred during the operation. **collisionList** Contains the name-schema pairs for the objects involved in a collision, and therefore not created.
Report new - report collisions	• No objects are created. • Reports on the collision status of the input objects. • Preexisting objects are NOT altered. • Errors are NOT generated by collisions.	**newList** Contains the name-schema pairs for the objects not involved in a collision, and not created. **collisionList** Contains the name-schema pairs for the objects involved in a collision, and not created.

Elements of the newList and collisionList must adhere to a predefined ordering. The following list shows the relative ordering between reference types in addition to the XML tags used.

1. <attributeRef>
2. <joinRef>
3. <attributeRelationshipRef>
4. <hierarchyRef>
5. <cubeHierarchyRef>
6. <dimensionRef>
7. <cubeDimensionRef>
8. <measureRef>
9. <factsRef>

10. <cubeFactsRef>
11. <cubeModelRef>
12. <cubeRef>

mode (for validate)
This argument sets the mode for the *validate* operation. The following table describes the various modes available. The types of rules indicated in the table below refer to categories of the DB2 Cube Views object rules.

Table 36. Validate modes

Mode	Description
base	• Check conformance to the Base Rules
cubeModel completeness	• Check conformance to the Cube Model Completeness Rules • Check conformance to the Base Rules
optimization	• Check conformance to the Optimization Rules • Check conformance to the Cube Model Completeness Rules • Check conformance to the Base Rules

Operation operands

When an operation requires metadata objects or their references to accompany the request, these objects or references are termed the ″operands″ of the operation. The following are descriptions of the various operands passed to metadata operations using either the *request* or *metadata* parameters:

object This operand contains the metadata objects being acted upon. The format used to represent metadata objects is described in "Metadata object format" on page 102.

currentRef
This operand is used during a metadata object renaming operation, and it contains the components of a metadata object reference. These components are the object's schema and name.

newRef
Similar to the *currentRef* operand, this operand is used during a metadata object renaming operation, and it contains the components of a metadata object reference. These components are the object's schema and name.

Message structure

The following table describes the components of a DB2 Cube Views message:

Table 37. Message components

Mode	Description
id	Unique integer identifier for the message.
type	A message can be one of three types: • informational • warning • error
text	The character string representing the text of the message.
tokens	The values substituted into the text string for the message. A message can have any number of tokens. The following XML elements can appear as tokens in a message: • attributeRef • joinRef • attributeRelationshipRef • hierarchyRef • cubeHierarchyRef • dimensionRef • cubeDimensionRef • measureRef • factsRef • cubeFactsRef • cubeModelRef • cubeRef • column • text

Here is an example of a message with no tokens:

```
<status id="0" text="Operation completed successfully."type="informational"/>
```

Here is an example of a message with tokens:

```
<status id="6331" text="The left attribute for
the &quot;MDOBJ_ID_ATTRIBUTERELATIONSHIP.MDSAMPLE.State_PopGroup&quot;
attribute relationship is not a part of the
&quot;MDOBJ_HIERARCHY.MDSAMPLE.RegionState&quot; hierarchy."
type="error">
<tokens>
```

```
<attributeRelationshipRef name="State_PopGroup" schema="MDSAMPLE"/>
<text value="MDOBJ_HIERARCHY.MDSAMPLE.RegionState"/>
</tokens>
</status><
```

Metadata object retrieval results

These results contain the metadata objects requested as part of a retrieval operation. Depending on the operation performed, metadata objects are returned in either the *response* parameter or the *metadata* parameter. The format used to represent retrieved metadata objects is described in “Metadata object format”.

Sequencing the operation steps

Of the arguments outlined in “Operation arguments” on page 96, only three determine the scope of an operation. The three arguments are listed here in the order they are applied to an operation:

1. objectType
2. restriction
3. recurse

The following example shows how you might have objects returned that apparently do not match the restriction you intended, but were returned as part of the ″recurse″ phase of the operation.

Example: Recursively describe the cubes belonging to the schema `myschema`:

Operation Arguments:

```
objectType = "cube"
restriction =  <restriction>
          <predicate property="schema" operator="=" value="myschema"/>
                </restriction>
recurse = "yes"
```

The describe operation begins by first limiting its scope to cube objects. Of these cube objects, only the ones that belong to the `myschema` schema are selected. For each of these selected cube objects, the objects they refer to are selected, and the objects are of different types and potentially different schemas. All the selected objects are then returned as part of a response to the cube request.

Metadata object format

The DB2 Cube Views XML Schema defines the base XML elements that map directly to the objects in the DB2 Cube Views metadata object model. Complex metadata structures are then represented as sequences of these base elements. Associations between objects within complex metadata structures are captured through name references between base elements. An example of a name reference is the way in which a cube element can contain a reference to a dimension element.

An example of the type of data provided for a cube object as defined by the XML Schema follows. In the following example, only text descriptions are shown; in application, XML representations of information are used.

```
cube
->cube model reference
->cube dimension references
->cube facts reference
->view
```

In the case of a cube object, the references to the other types of objects are all contained within the base element representing the cube. With non-recursive retrieval operations, sequences of cube objects (and only cube objects) are presented. With recursive retrieval operations, not only is information on cube objects presented, but information on any other object (of different type) referenced by the identified cubes is also presented.

The ordering of objects is defined by the DB2 Cube Views XML Schema. Within the scope of a single operation, objects of the same type (e.g., cube objects) are grouped together. Within these groups, the order of elements is influenced by the references between objects of the same type. Referenced objects must appear before referencing objects. The ordering between these groups is as follows:

1. attribute
2. join
3. attributeRelationship
4. hierarchy
5. cubeHierarchy
6. dimension
7. cubeDimension
8. measure
9. facts
10. cubeFacts
11. cubeModel
12. cube

Example: The following is the type and order of information that would be returned from a recursive retrieval of a sample cube named *LocationProduct*.

For readability, object schema names have not been included as part of object references, and relational table column names have not been included as part of attribute objects.

```
attribute ("LocationID")
attribute ("LocationID_Facts")
attribute ("Country")
attribute ("State")
attribute ("City")
attribute ("ProductID")
attribute ("ProductID_Facts")
attribute ("GroupName")
attribute ("ProdName")
join ("LocFactsJoin")
       ->attribute references: LocationID, LocationID_Facts
join ("ProdFactsJoin")
       ->attribute references: ProductID, ProductID_Facts
hierarchy ("LocDetail")
       ->attribute references: Country, State, City
hierarchy ("Product")
       ->attribute references: GroupName, ProdName
cubeHierarchy ("LocDetailCH")
       ->attribute references: Country, State
       ->hierarchy reference: LocDetail
cubeHierarchy ("ProductCH")
       ->attribute references: GroupName, ProdName
       ->hierarchy reference: Product
dimension ("Location")
       ->attribute references: LocationID, Country, State, City
       ->join references: ""
       ->function dependency references: ""
       ->hierarchy references: LocDetail
dimension ("Product")
       ->attribute references: ProductID, GroupName, ProdName
       ->join references: ""
       ->function dependency references: ""
       ->hierarchy references: Product
cubeDimension ("LocationCD")
       ->attribute references: LocationID, Country, State, City
       ->dimension reference: Location
       ->cube hierarchy reference: LocDetailCube
cubeDimension ("ProductCD")
       ->attribute references: ProductID, GroupName, ProdName
       ->dimension reference: Product
       ->cube hierarchy reference: ProductCube
measure ("Revenue")
measure ("Profit")
       ->measure references: Revenue
facts ("Facts")
       ->measure references: Revenue, Profit
       ->attribute references: LocationID_Facts, ProductID_Facts
       ->join references: ""
cubeFacts ("FactsCF")
       ->measure references: Revenue, Profit
       ->facts reference: Facts
```

```
cubeModel("LocationProductModel")
       ->facts references: Facts
       ->dimensionInfo:Location
              ->dimension reference:Location, Product
              ->join reference: LocFactsJoin, ProdFactsJoin
       ->dimensionInfo:Product
              ->dimension reference:Product
              ->join reference:ProdFactsJoin
cube ("LocationProduct")
       ->cube model reference: LocationProductModel
       ->cube facts reference: FactsCF
       ->cube dimension references: LocationCD, ProductCD
       ->view: CubeView
```

The order of the object-type groups is independent of the associations between objects. The attribute named *City* is included in the *LocDetail* hierarchy, but it is not included in the *LocDetailCH* cube hierarchy.

The fact that attributes and joins play different roles when associated with different object types does not affect their order within the Association format.

Application programming notes

Most API programming issues are either DB2 stored procedure issues or XML parsing issues. For details on programming with DB2 stored procedures, refer to the *DB2 Application Development Guide*. The following issues should be noted:

Transaction:

Transactional, multi-user metadata access is supported through the use of DB2's traditional transaction mechanisms (Refer to the *DB2 Application Development Guide* for more details). All database actions performed within the API belong to the calling application's database transaction. It is therefore possible for an application using the API to execute COMMIT or ROLLBACK after calling the *md_message* stored procedure as appropriate to create the desired units of database work.

Memory management:

Parameters are exchanged between applications and the *md_message* stored procedure in the form of CLOB structures. Applications calling the *md_message* procedure must preallocate CLOB parameter structures that are the same size as those used to catalog the stored procedure. The API supports the DB2 maximum size for a CLOB, which is 2 GB.

System configuration:

To support the exchange of large parameters, you might have to change the following DB2 settings:

- The database client application that calls *md_message* might have to be linked using larger heap and stack sizes.
- The DB2 query heap size for the database might have to be increased using the **query_heap_sz** setting.

XML parsing:

It is the responsibility of the applications using the API to parse the output parameter returned by the *md_message* stored procedure. A variety of XML parsers are available to developers wanting to use the API. To find IBM Web sites offering XML resources, search for `xml` on www.ibm.com.

Error handling:

Error information is generated in three forms by the API:

1. SQLCODE and SQLSTATE information returned by the stored procedure to the calling application.
2. XML structures delivered to calling applications via the *response* API parameter.
3. Error and run-time log files located on the database server running the API.

If an error occurs due to XML validation, parsing, or tagging, then the *response* parameter will be returned to the calling application with an <error> tag in place of an operation tag. This <error> XML element will contain a return code and return message describing the problem encountered by the API.

If an error occurs within the API unrelated to XML processing, but related still to the execution of a metadata operation, then the contents of the response parameter are returned.

The following example shows the type of information in an <error> tag. Note that within this example, descriptions of the parameter structures use XML tags in a limited fashion. In application, parameters will have more XML tags than are shown here, and parameter contents will be validated using the XML Schema.

```
<olap:response xmlns:olap="http://www.ibm.com/olap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="8.1.2.1.0">
<error>
<status id="3100" text="The system failed to parse XML for
   &quot;INPUT PARAMETER&quot; (line:&quot;3&quot;,
   char:&quot;26&quot;, message:&quot;Unknown element
   'dropa'&quot;). " type="error"/>
</error>
</olap:response>
```

When the DB2 Cube Views stored procedure is called, regardless of whether the stored procedure was actually executed, DB2 returns a SQLCODE and a SQLSTATE to the calling application. If the DB2 Cube Views stored procedure was able to execute, the stored procedure returns a status message as part of the XML data sent to the calling application.

Configuration file

The API can be configured at the level of a DB2 instance. You can change the parameters of a configuration file that is called `db2md_config.xml`. Every installation of DB2 Cube Views has a default configuration file in the `<db2 installation path>/cfg` directory. For example, on Windows, the default configuration file might be in the `c:\sqllib\cfg` directory, and on AIX, the default configuration file might be in the `/usr/opt/db2_08_01/cfg` directory.

Every DB2 instance that runs DB2 Cube Views has a physical copy of the `db2md_config.xml` file in the `<db2 instance path>` directory. For example, on Windows, the physical copy might be in the `c:\sqllib\<my inst>` directory, and on AIX, the physical copy might be in the - `~<my inst>/sqllib` directory.

The db2icrt utility copies the default configuration file to the `<db2 instance path>` directory and it creates a new instance. For DB2 instances that were created before DB2 Cube Views was installed, you can manually copy the configuration file into the instance directory if the installation program did not successfully copy the file. If the API cannot find the configuration file in the instance directory, the API will try to copy the default configuration file to the instance directory.

A configuration file, `db2md_config.xml`, is used to set error logging and run-time tracing. By modifying the contents of the configuration file, an administrator can specify the level of tracing, the severity of errors to log, the buffer size (in bytes) to use when logging.

The content structure of the `db2md_config.xml` configuration file is defined by the `db2md_config.xsd` XML schema file. The following example shows the contents of the configuration file.

```
<olap:config xmlns:olap="http://www.ibm.com/olap"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xsi:schemaLocation="http://www.ibm.com/olap db2md_config.xsd">
      <log>
```

```
        <trace level="none" logFile="mdtrace.log" bufferSize="0"/>
        <error level="medium" logFile="mderror.log" bufferSize="0"/>
    </log>
</olap:config>
```

Run-time tracing

Typically there is no need to run tracing. Tracing might be required if an error occurs within the API and IBM support asks you to provide a trace file.

The API supports three priorities of tracing. Using the configuration file, an administrator can set the level of tracing to log to file. Run-time tracing is turned off by default, and the default trace file name is `mdtrace.log`.

The following table describes the various tracing levels.

Level	Description	Examples
none	• Tracing is off	n/a
high	• Tracks only external and internal API entry and exit points • Tracks inter-component flow	• Begin and end parsing • Begin and end create, describe, drop, etc. • Can include function arguments
medium	• Tracks the flow of control between complex functions within the external and internal API • Tracks the flow between components • Includes high level trace points	• Shows complex function calls made by the create operation
low	• Tracks simple/atomic functions within the internal API • Includes high and medium level trace points • **Most trace points have this level**	• Shows calls to the get/set methods for the metadata objects

When tracing is turned on, with the level set to a value other than `none`, errors that occur in the API might be recorded in both the error log and the trace log, depending on the level and severity setting for these logs.

Log files

The API log files are generated at a DB2 instance level. The name of the error log file is `db2mderror.log`, and the name of the trace log file is `db2mdtrace.log`.

For a given DB2 instance running the DB2 Cube Views API, the log files for the API will be generated in the `<db2 diagnostic data directory path>`, also know as the DB2DIAG path which is:

On Windows
: `<db2 instance path>` directory, such as `c:\sqllib\<myinst>`

On AIX
: `<db2 instance path>/db2dump`, such as `~<my inst>/sqllib/db2dump`

You can change the default DB2DIAG path by using the `DB2DIAG db2 dbm cfg` setting.

The db2idrop utility cleans up the log files associated with a DB2 instance. If the default for DB2DIAG is not used, then the db2idrop utility cannot clean up the log files for the DB2 Cube Views API. The log files that cannot be cleaned up by the db2idrop utility must be manually cleaned up. Errors related to the DB2 Cube Views API loading the configuration file are logged in the `db2mdapi.log` file. The `db2mdapi.log file` is located in the DB2DIAG path similar to the other API logs.

Error logging

The API distinguishes between three severities of errors. The default severity setting is medium, and the default error log file name is `mderror.log`. When an error occurs while reading the configuration file, this error is logged in a file named `mdapi.log`.

The following table describes the error severity levels.

Table 38. Error severity levels

Severity	Description	Examples
none	• Ignore all errors and warnings	n/a
high	• Records only critical, unrecoverable errors • Results in a callstack being dumped to log • **Most errors have this severity**	• Internal coding error
medium	• Records user-recoverable errors • Results in high severity errors being logged also • Results in a callstack being dumped to log	• End-user mistakes, such as attempts to create a duplicate object • Metadata validation errors • Out of memory. You can increase memory or reduce usage.

Table 38. Error severity levels (continued)

Severity	Description	Examples
low	• Records warning situations • Results in high and medium severity errors being logged also • Low severity errors do NOT result in a callstack being dumped • A callstack is dumped for only medium and high severity errors	• Warns of internal error • Informational messages

When the API is configured to `high` or `medium` error logging, and a `high` or `medium` error occurs, the API generates a callstack beginning at the point where the error occurs in the API. This callstack is similar to a medium-level trace, but the data is sent to the error log instead of the trace log.

Examples

Scenario 1 (error severity: high; trace level: medium): When a high-severity error occurs, it appears in both the error and the trace logs.

Scenario 2 (error severity: low; trace level: medium): When a low-severity error occurs, it appears in only the error log because the trace log only allows entries of level medium or high.

Errors related to missing environment variables or to failures accessing log files are returned via the SQLSTATE of the stored procedure call to the database client application. When an error occurs processing the configuration file, this error is logged in the `db2mdapi.log` file. If an error occurs opening any of the user-specified log files, no error is captured.

Appendix B. Code page support

DB2 Cube Views uses two code pages: the DB2 client code page (application code page) and the DB2 database code page. See the DB2 *Application Development Guide: Programming Client Applications*, ″National Language Support″ for information about how to determine the DB2 client code page. The DB2 Cube Views API stored procedure runs in the DB2 database code page. The DB2 database code page is set when the database is created. The DB2 client code page and DB2 database code page can be different. CLI will convert the stored procedure character large object (CLOB) parameters from client code page to database code page for the stored procedure.

The following illustration shows how the client communicates with the server through a call-level interface (CLI). The CLI converts client code pages to the database code page.

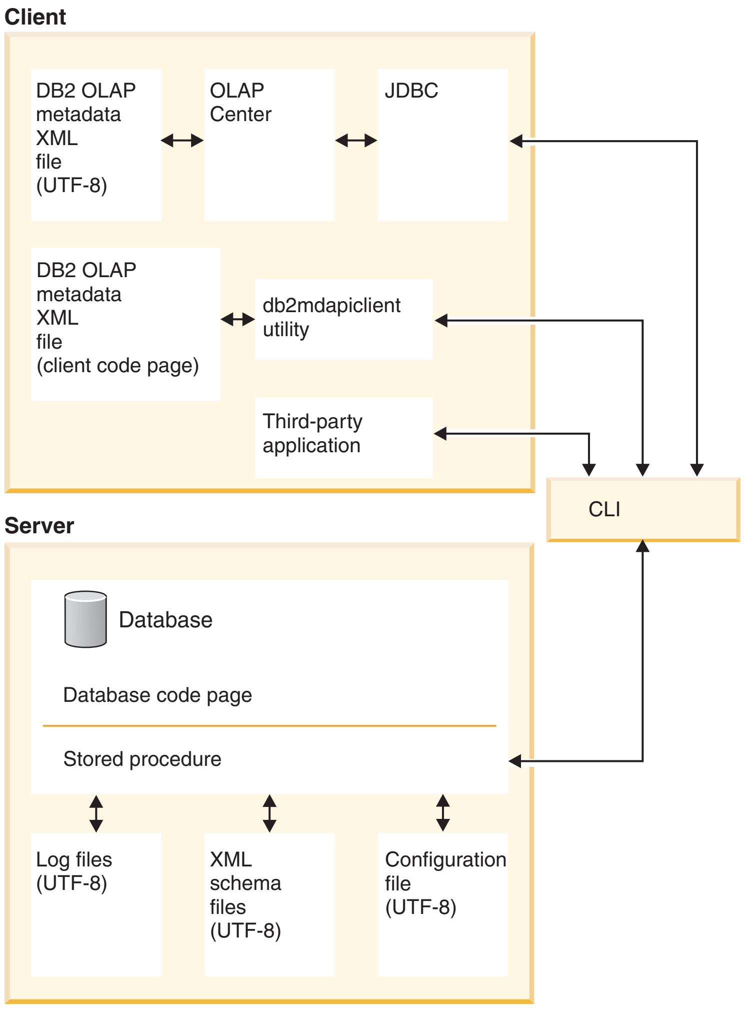


Figure 31. How data flows from different clients that use code pages or UTF-8 format through the DB2 CLI then to the database server

CLI manages the conversion between application code page and database code page. Data that is sent from the DB2 Cube Views client to the API is considered input. Data that is sent from the API to the DB2 Cube Views client is considered output. Input and output data is encoded in the DB2 client code page.

The components of DB2 Cube Views have the following code page specifications. The OLAP Center:

- Accepts and generates only DB2 Cube Views XML files that are encoded in UTF-8

- Returns an error if an input DB2 Cube Views XML file specifies an encoding other than UTF-8
- Interprets the lack of an encoding specification in a file as meaning that the file is encoded in UTF-8
- With the Export function creates DB2 Cube Views XML files with an explicit encoding specification of UTF-8

The db2mdapiclient:
- Interprets input DB2 Cube Views XML files as being encoded in the DB2 client code page, and therefore ignores explicit encoding specifications listed within the files.
- Generates DB2 Cube Views XML files encoded in the DB2 client code page, and does not include explicit encoding specifications within these files.

The stored procedure API:
- Interprets CLOB parameters as being encoded in the DB2 client code page
- Ignores explicit encoding specifications in input DB2 Cube Views XML files
- Generates DB2 Cube Views XML files with no explicit encoding specifications
- Processes input and output XML files using the DB2 database code page
- Generates API log files that are encoded by using UTF-8 including any embedded DB2 messages. See "Reading UTF-8 encoded files" for more information on how to convert files between code pages.
- Does not create XML log files
- Does not create log files explicitly stating an encoding of UTF-8
- Encodes in UTF-8 the XML schema files that are used by the API
- Encodes in UTF-8 the XML API configuration file

For third-party applications, other applications that directly call the DB2 Cube Views API will have to pass and accept as parameters XML files that are encoded in the DB2 client code page.

Reading UTF-8 encoded files

Depending on your operating system, you can directly open a UTF-8 encoded file, or use a utility to convert it to your local code page.

Procedure:

To read a UTF-8 encoded file on Windows, open the file using Notepad.

To read a UTF-8 encoded file on Unix:

1. Use the iconv utility to convert the file to your local code page.
2. Use a text editor to open the converted file.

Code page restrictions

With user definable characters (UDC), you can define new characters and map them to one of the reserved code points.

In the OLAP Center, UDCs might not appear correctly even though importing XML files with UDCs is successful.

Language shortcuts are not removed when DB2 Cube Views is removed. On Windows, when you install DB2 Cube Views in one language and then remove it, the shortcut for the language is not removed. If you install DB2 Cube Views again in another language, the shortcut for the first lanuage remains, and it must be removed manually.

Appendix C. Sample files

This section describes the sample database and application files provided with DB2 Cube Views.

Sample database files

All of the following files, that are related to the MDSAMPLE database, are located in the `\SQLLIB\olap\mdsample\` directory.

create.xml
An XML file with the CREATE operation. Use this file to load the sample with the db2mdapiclient utility.

MDSampleMetadata.xml
An XML file that contains the MDSAMPLE metadata. Use this file to import the MDSAMPLE metadata with the OLAP Center and the db2mdapiclient utility.

MDSampleTables.sql
An SQL script that you use to populate the MDSAMPLE tables.

FAMILY.txt, MARKET.txt, POPULATION.txt, PRODUCT.txt, REGION.txt, SALESFACT.txt, TIME.txt
A set of text files that contain the MDSAMPLE table data.

MDSampleExplain.sql
An SQL script that you can use to determine if DB2 is rerouting a query to a summary table.

Sample application files

Sample application files for the MDSAMPLE database that are provided with DB2 Cube Views. You can use the sample application files to perform sample scenarios. You pass the contents of the sample application files listed for each scenario as parameters to the MD_MESSAGE() stored procedure. Sometimes, you use an empty string, that is noted in the following scenarios as <empty>, for a stored procedure parameter. For more information about using the MD_MESSAGE() stored procedure with the db2mdapiclient utility, see “The db2mdapiclient utility for importing, exporting, and optimizing metadata” on page 7. For more information on using the MD_MESSAGE() stored procedure by itself, see “DB2 Cube Views stored procedure” on page 82. All of the sample application files are located in the `\SQLLIB\samples\olap\xml\` directory. You can use the sample application files to perform the following sample scenarios:

DROP Use these files to drop all of the metadata objects in the metadata catalog. This sample assumes that the metadata catalog is not empty. If the metadata catalogs are empty, you receive a warning message that no objects are found for the operation.

Files that contain input parameter structures:

- **Request:** `input\drop.xml`
- **Metadata:** `<empty>`

Files that contain output parameter structures:

- **Response:** `output\drop.xml`
- **Metadata:** `<empty>`

CREATE

Use these files to create metadata objects in the metadata catalog. This sample assumes that the metadata catalog is empty.

Files that contain input parameter structures:

- **Request:** `input\create.xml`
- **Metadata:** `input\MDSampleMetadata.xml`

Files that contain output parameter structures:

- **Response:** `output\create.xml`
- **Metadata:** `<empty>`

DESCRIBE

Use these files to describe all of the metadata objects in the metadata catalog. This sample assumes that you have already completed the CREATE scenario and that the metadata catalog contains all of the objects created by the CREATE scenario.

Files that contain input parameter structures:

- **Request:** `input\describe.xml`
- **Metadata:** `<empty>`

Files that contain output parameter structures:

- **Response:** `output\describe.xml`
- **Metadata:** `output\MDSampleDescribe.xml`

DESCRIBE (Restricted)

Use these files to recursively describe the MDSAMPLE.Sales cube. This sample assumes that you have already completed the CREATE scenario and that the metadata catalog contains all of the objects created by the CREATE scenario.

Files that contain input parameter structures:

- **Request:** `input\MDSampleDescribe_restricted.xml`
- **Metadata:** `<empty>`

Files that contain output parameter structures:

- **Response:** `output\MDSampleDescribe_restricted.xml`
- **Metadata:** `output\MDSampleDescribe.xml`

ALTER

Use these files to alter metadata objects in the metadata catalog. This sample assumes that you have already completed the CREATE scenario and that the metadata catalog contains all of the objects created by the CREATE scenario.

Files that contain input parameter structures:

- **Request:** `input\alter.xml`
- **Metadata:** `input\MDSampleAlter.xml`

Files that contain output parameter structures:

- **Response:** `output\alter.xml`
- **Metadata:** `<empty>`

RENAME

Use these files to rename the MDSAMPLE.SalesModel cube model. The cube model is renamed to MDSAMPLE.SalesModel(2003). This sample assumes that you have already completed the CREATE scenario and that the metadata catalog contains all of the objects created by the CREATE scenario.

Files that contain input parameter structures:

- **Request:** `input\MDSampleRename.xml`
- **Metadata:** `<empty>`

Files that contain output parameter structures:

- **Response:** `output\MDSampleRename.xml`
- **Metadata:** `<empty>`

VALIDATE

Use these files to validate all of the metadata objects in the metadata catalog using an optimization validation mode. This sample assumes that you have already completed the CREATE scenario and that the metadata catalog contains all of the objects created by the CREATE scenario.

Files that contain input parameter structures:

- **Request:** `input\validate.xml`
- **Metadata:** `<empty>`

Files that contain output parameter structures:

- **Response:** `output\validate.xml`
- **Metadata:** `<empty>`

VALIDATE (Restricted)

Use these files to validate the MDSAMPLE.Sales cube using an optimization validation mode. This sample assumes that you have already completed the CREATE scenario and that the metadata catalog contains all of the objects created by the CREATE scenario.

Files that contain input parameter structures:

- **Request:** `input\MDSampleValidate_restricted.xml`
- **Metadata:** `<empty>`

Files that contain output parameter structures:

- **Response:** `output\MDSampleValidate_restricted.xml`
- **Metadata:** `<empty>`

IMPORT with the *create new - ignore collisions* mode

Use these files to import metadata objects into the metadata catalog using the *create new - ignore collisions* import mode. This sample assumes that the metadata catalog is empty.

Files that contain input parameter structures:

- **Request:** `input\import_mode1.xml`
- **Metadata:** `MDSampleMetadata`

Files that contain output parameter structures:

- **Response:** `output\import_mode1.xml`
- **Metadata:** `<empty>`

IMPORT with the *create new - replace collisions* mode

Use these files to import metadata objects into the metadata catalog using the *create new - replace collisions* import mode. This sample assumes that you have already completed the IMPORT with the *create new - ignore collisions* mode scenario.

Files that contain input parameter structures:

- **Request:** `input\import_mode2.xml`
- **Metadata:** `MDSampleMetadata2`

Files that contain output parameter structures:

- **Response:** `output\import_mode2.xml`
- **Metadata:** `<empty>`

IMPORT with the *create new - abort on collision* **mode**

Use these files to import metadata objects into the metadata catalog using the *create new - abort on collision* import mode. This sample assumes that you have already completed the IMPORT with the *create new - replace collisions* mode scenario.

Files that contain input parameter structures:

- **Request:** `input\import_mode3.xml`
- **Metadata:** `MDSampleMetadata3`

Files that contain output parameter structures:

- **Response:** `output\import_mode3.xml`
- **Metadata:** `<empty>`

IMPORT with the *create new - report collisions* **mode**

Use these files to import metadata objects into the metadata catalog using the *create new - report collisions* import mode. This sample assumes that you have already completed the IMPORT with the *create new - abort on collision* mode scenario.

Files that contain input parameter structures:

- **Request:** `input\import_mode4.xml`
- **Metadata:** `MDSampleMetadata4`

Files that contain output parameter structures:

- **Response:** `output\import_mode4.xml`
- **Metadata:** `<empty>`

Appendix D. Messages

The following messages are from the server, API, and OLAP Center of DB2 Cube Views.

Socket error: Opening and closing a database connection multiple times can cause a socket error. In rare circumstances, a socket error can occur when you run DB2 Cube Views with DB2 Universal Database Enterprise Server Edition, Version 8.1.2 in a partitioned environment on Windows 2000 Advanced Server. The error might occur if you repeat the following steps more than 10 000 times quickly within a single Windows session:

1. Open a connnection to a DB2 database.
2. Call the DB2 Cube Views stored procedure to perform a metadata operation.
3. Close the database connection.

The workaround is to restart the Windows workstation to reactivate the socket.

SQLSTATE, API, and other server messages

01HQ1 See output XML and server logs.

Explanation: The call to the stored procedure completed, but errors were detected during the execution of one of the requested metadata operations.

User Response: Check the contents of the output parameters of the stored procedure for information. You can also check the entries in the server logs for more information.

38Q00 See server logs for more information.

Explanation: The call to the stored procedure failed. The requested metadata operation or operations were not executed. No information was returned from the stored procedure through the output parameters.

User Response: Check the entries in the server logs for more information.

38Q01 Installation path is unknown.

Explanation: The call to the stored procedure failed because the DB2 installation directory cannot be determined by the stored procedure process. The requested metadata operation or operations were not executed. No information was returned from the stored procedure through the output parameters.

User Response: If you use a Windows operating system, ensure that the DB2PATH environment variable is set to the correct value either by default or by user action. Restart the database manager and then reissue the call to the stored procedure. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

38Q02 **Cannot open the server log file.**

Explanation: The call to the stored procedure failed because at least one of the log files that is used by the stored procedure could not be opened for writing by the stored procedure process. The requested metadata operation or operations were not executed. No information was returned from the stored procedure through the output parameters.

User Response: Ensure that the log files specified in the stored procedure configuration file (for example, olap_config.xml) can be created or opened for reading and writing on the appropriate file system. If the log files do not already exist, the stored procedure attempts to create these files. On AIX, ensure that the log files can be read and written to by the fenced database user ID.

38Q03 **Metadata input parameter missing.**

Explanation: The call to the stored procedure failed because the requested metadata operation requires that metadata be passed as input to the stored procedure, but no metadata was provided through the input metadata parameter. No information was returned from the stored procedure through the output parameters.

User Response: Provide the metadata necessary using the input metadata stored procedure parameter for the requested metadata operation the next time you make a call to the stored procedure.

38Q04 "[*error_type*] **ERROR: Response output buffer too small.**

Explanation: The call to the stored procedure failed because the output parameter buffer for the operation response is too small to accomodate the CLOB structure being returned. No information was returned from the stored procedure through the output parameters.

User Response: Recatalog the stored procedure by using a larger size for the output response parameter.

0 **Operation completed successfully. No errors were encountered.**

Explanation: The requested metadata operation completed successfully. No errors were encountered during execution of the operation.

User Response: This information is for your information only. No action is required.

1 **Operation completed. Additional information was returned.**

Explanation: The requested metadata operation completed. The operation has returned additional information that may describe warning or error situations.

User Response: Check the INFO element for the additional information returned.

2 **Operation completed. No changes were made to the metadata.**

Explanation: The requested metadata operation completed. The operation resulted in no changes being made to the metadata in the database catalog.

User Response: Reissue the metadata operation request by using a different mode if you want changes to be made to the metadata in the database catalog.

100 **Failed to allocate memory for** *operation***. Make sure that memory is available.**

Explanation: During execution of the requested metadata operation, the stored procedure failed to allocate required memory segments.

User Response: Increase the memory that is available to the fenced stored procedure process.

101 **An internal error occurred while processing the** *object name***object.**

Explanation: During execution of the requested metadata operation, an unexpected internal error was encountered.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

---

102 The output buffer of size *buffer_size* is too small. Change the buffer size to at least *size*.

Explanation: The output parameter buffer available to the stored procedure is too small to accommodate the CLOB structure that is generated by the stored procedure.

User Response: If possible recatalog the stored procedure using larger sizes for the OUT and INOUT parameters. Otherwise, you must restrict your query so that less information is returned by the stored procedure.

---

103 A valid license for this product does not exist.

Explanation: No metadata operations can be performed because a valid product license does not exist for this installation of the product.

User Response: Install a valid product license on the system, or contact IBM Customer Support or IBM Software Sales to purchase a new product license.

---

104 An internal error occurred. The following tokens were returned: *token0*, *token1*, *token2*, *token3*.

Explanation: During execution of the requested metadata operation, an unexpected internal error was encountered.

User Response: Contact IBM Software Support for further assistance. Provide the status ID and text for the metadata operation that was attempted. If possible, also provide the stored procedure log files from the database server.

---

599 The operation was not executed.

Explanation: An error was encountered prior to executing this operation. As a result, this operation was not executed.

User Response: Check the results of previous metadata operations that were executed during the same stored procedure call. You can also check the entries in the server logs for more information. After you correct the problems that caused the earlier operation to fail, call the stored procedure again and request the same metadata operations.

---

600 The input *parameter_name* parameter is invalid with this message: *message*. Check the parameter and try again.

Explanation: One of the parameters that was passed as input to a method internal to the stored procedure is invalid.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

---

601 The input *parameter_name* parameter is NULL.

Explanation: One of the parameters that was passed as input to a method internal to the stored procedure has an invalid value of NULL.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

---

602 The *parameter_name* parameter with value *value* is not in the valid range of *range_value1*, *range_value2*.

Explanation: One of the parameters that was passed as input to a method internal to the stored procedure has a value outside of the valid range.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

603 **The Unicode String *string* is either bogus or invalid. There might be a memory problem.**

Explanation: A Unicode string in the stored procedure is either incorrect or invalid. This might indicate a memory problem on the system or in the stored procedure. This might also be the result of the wrong version of the ICU libraries being loaded by the stored procedure.

User Response: Ensure that there is adequate memory available to accommodate the volume of data being processed by the stored procedure. Ensure that the version of the ICU libraries that you intend to use with the current version of the stored procedure are being loaded. You might need to check the run-time library search path set in the environment to determine correct setup.

Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

604 **Failed to convert the contents of the string *string* from Unicode to the active code page of *code_page*.**

Explanation: Conversion of a Unicode string object to a string using another encoding failed. This might indicate a memory problem on the system or in the stored procedure. This might also indicate a code page conversion problem on the system, or the wrong version of the ICU libraries was loaded by the stored procedure.

User Response: Ensure that the necessary ICU code page conversion files are installed on the database server system. Ensure that there is adequate memory available to accommodate the volume of data being processed by the stored procedure. Ensure that the version of the ICU libraries intended for use with the current version of the stored procedure are being loaded. You might need to check the run-time library search path set in the environment to determine correct setup.

Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

605 **The allocated memory at *memory_buffer* needs to be freed.**

Explanation: A method internal to the stored procedure has returned a memory buffer that needs to be freed by another internal method.

User Response: A method internal to the stored procedure should free the returned memory buffer. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

606 **Conversion of XMLCh to UChar for *UChar* failed.**

Explanation: Conversion between an XMLCh character and a UChar character failed.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

607 **The input buffer size of *size* is too small. Change the buffer size to at least *new_size*.**

Explanation: A memory buffer internal to the stored procedure is too small to accommodate the text for a required message.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

608 **The type of *stored_procedure_name* is not valid in the current context.**

Explanation: An unexpected type was encountered during stored procedure processing.

User Response: Contact IBM Software Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1000 **Failed to clone object *object_name*.**

Explanation: An error occurred while cloning a class object internal to the stored procedure.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1001 **The list of attributes cannot exceed two elements.**

Explanation: A list of attributes internal to the stored procedure unexpectedly has more than two elements.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1002 **The called function *function_name* is not supported.**

Explanation: A virtual method internal to the stored procedure was not implemented for one of the stored procedure's classes.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1003 **The container is unexpectedly empty.**

Explanation: A container structure internal to the stored procedure is unexpectedly empty.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1004 **The *object_name* object cannot be found in the container.**

Explanation: An object that was searched for in one of the stored procedure's internal container structures is unexpectedly missing.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1005 **A duplicate of the *element_name* element already exists in the container.**

Explanation: An object that should not already exist in one of the stored procedure's internal container structures already exists.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1006 **An exception occurred during a list operation.**

Explanation: An unexpected exception occurred while executing an operation on one of the stored procedure's internal list structures.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1007 An internal error occurred in the container with error code *error and number* **and msg** *message***.**

Explanation: An error occurred while executing an operation on one of the stored procedure's internal container structures.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1008 The copy operation did not copy all properties completely. The copy operation failed for the *property_name* **property with value** *value***.**

Explanation: An error occurred while executing a copy operation on one of the stored procedure's internal objects. One of the internal object's properties failed to be copied.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1009 The object type of *type1* **is not valid.** *Type2* **expected.**

Explanation: An unexpected object type was encountered during stored procedure processing.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1010 The *parameter_name* **parameter does not have a complete ID.**

Explanation: One of the parameters that was passed as input to a method internal to the stored procedure is a metadata object ID that is incomplete.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1011 The *object_name* **object does not have a complete ID.**

Explanation: A metadata object ID is unexpectedly incomplete in the stored procedure.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1012 The *parameter_name* **parameter is the same as the object.**

Explanation: One of the parameters passed as input to an object method internal to the stored procedure is an object that is unexpectedly equal to the object owning the method.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1013 An unexpected NULL pointer was encountered.

Explanation: An unexpected NULL pointer was encountered during stored procedure processing.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1014 The container cursor reached the end of the container.

Explanation: A cursor on one of the container structures internal to the stored procedure has unexpectedly reached the end of the container.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

1015 **The** *object_name* **object is invalid. Reason: ID=***ID***, Message=***message*

Explanation: A metadata object internal to the stored procedure is invalid.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

2000 **The operation failed for the** *object_name* **object with error code** *code_number* **and message** *message***.**

Explanation: An error occurred while executing the requested metadata operation. All changes made to the metadata in the catalog during this call to the stored procedure have been undone.

User Response: Fix the problems outlined in the error message and call the stored procedure again.

2001 **The generated query** *query* **does not contain the required column objectType.**

Explanation: An SQL query generated by the stored procedure is missing a required column.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

3000 **An error occurred in the parser with error code** *code and number* **and msg** *message***.**

Explanation: An error occurred in the stored procedure while parsing the XML passed to the stored procedure.

User Response: Ensure that the XML passed to the stored procedure is well formed and that it is valid for the XML schema that is published for this product. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

3001 **An XML exception was encountered by the parser during** *operation* **with message** *message***.**

Explanation: An unexpected exception was encountered in the stored procedure while parsing the XML passed to the stored procedure.

User Response: Ensure that the XML passed to the stored procedure is well formed and that it is valid for the XML schema that is published for this product. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

3002 **An unexpected parser exception was encountered in** *operation***.**

Explanation: An unexpected exception was encountered in the stored procedure while parsing the XML passed to the stored procedure.

User Response: Ensure that the XML passed to the stored procedure is well formed and that it is valid for the XML schema that is published for this product. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

3003 **SAXParseException encountered by parser during** *operation* **with message** *message***.**

Explanation: An unexpected exception was encountered in the stored procedure while parsing the XML passed to the stored procedure.

User Response: Ensure that the XML passed to the stored procedure is well formed and that it is valid for the XML schema that is published for this product. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

3004 **The system failed to get parser error message for** *operation***.**

Explanation: An unexpected error occurred in the stored procedure while parsing the XML passed to the stored procedure. An error message from the XML parser could not be retrieved.

User Response: Ensure that the XML passed to the stored procedure is well formed and that it is valid for the XML schema that is published for this product. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

3100 **The system failed to parse XML for** *parameter type* **(line:** *line***, char:***character***, message:** *message***).**

Explanation: The stored procedure could not parse the input XML. The input XML might not be well formed or it might be invalid for the XML schema that was published for this product.

User Response: Ensure that the XML passed to the stored procedure is well formed and that it is valid for the XML schema that is published for this product.

3101 **An unknown metadata object was encountered.** *parser_message*

Explanation: An unknown type of metadata object exists in the XML passed to the stored procedure. This input XML cannot be processed by the stored procedure.

User Response: Ensure that the XML passed to the stored procedure is well formed and that it is valid for the XML schema that is published for this product.

3102 **An unknown XML attribute was encountered.** *attribute_name***,** *attribute_value***.**

Explanation: An unknown type of XML attribute exists in the XML passed to the stored procedure. This input XML cannot be processed by the stored procedure.

User Response: Ensure that the XML passed to the stored procedure is well formed and that it is valid against the XML schema that is published for this product.

3103 **An invalid enumeration value was encountered by the handler for the attribute with name** *name* **and value** *value***.**

Explanation: An invalid enumeration value exists in the XML passed to the stored procedure. This input XML cannot be processed by the stored procedure.

User Response: Ensure that the XML passed to the stored procedure is well formed and that it is valid against the XML schema that is published for this product.

3104 **There was an unknown object reference with name** *name***.**

Explanation: The input XML contains a reference to a metadata object that does not exist in the metadata catalog.

User Response: Metadata objects must be created in the metadata catalog before they can be referenced by subsequent metadata operations or by other metadata objects.

3500 **Data is required for the attribute or element name** *name***.**

Explanation: The stored procedure failed to set the value for the indicated XML attribute or element in the XML that is to be returned by the stored procedure.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

3501 Data is required for the attributes *attribute_name1* and *attribute_name2*.

Explanation: The stored procedure failed to set the value for the indicated XML attribute or element in the XML that is to be returned by the stored procedure.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

3502 An invalid enumeration value was encountered by the formatter for the attribute with name *name* and value *value*.

Explanation: An invalid enumeration value was encountered in the stored procedure while formatting the XML that was to be returned by the stored procedure.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4000 The database connection failed. Database name *database_name*, User name *user_name*.

Explanation: The stored procedure failed to establish its own connection to the database.

User Response: Ensure that the user ID that the stored procedure uses has the appropriate privileges to connect to the database.

4001 The database connection was not issued because a connection already exists.

Explanation: The stored procedure did not connect to the database because a connection already exists between the stored procedure and the database.

User Response: This information is for your information only. No action is required.

4002 The database operation failed.

Explanation: An error occurred while executing an SQL statement issued by the stored procedure to the database.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4003 Execution of the CLI call *call_name* failed.

Explanation: An error occurred while executing the indicated CLI call.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4004 The returned data is truncated.

Explanation: Diagnostic information that was returned during the failed database operation was truncated.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4005 **A warning was received from the database. SQLSTATE=*code*, Message=*message*.**

Explanation: Warning information was returned by a CLI call that was issued by the stored procedure.

User Response: Check the database manager log files on the client and on the server.

4008 **An unknown DB2 data type was encountered.**

Explanation: An unknown data type was encountered by the stored procedure while executing a database request.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4009 **No valid savepoint name was generated.**

Explanation: The stored procedure was unable to generate a valid database transaction savepoint name. The stored procedure uses its database application ID to form the savepoint name.

User Response: Reissue the call to the stored procedure. Reissuing the call might generate a new database application ID for the stored procedure, and it might therefore allow for the generation of a valid savepoint name. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4010 **The attempt to set a DB2 savepoint failed.**

Explanation: The stored procedure was unable to set a database transaction savepoint. A savepoint with the same name as the one used by this instance of the stored procedure may already exist in the current transaction.

User Response: If possible, release the savepoints for the current transaction before reissuing the call to the stored procedure. You can also reissue the call to the stored procedure by using a new transaction.

Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4011 **A savepoint was not set prior to this point of execution.**

Explanation: A transaction savepoint is unexpectedly missing at a point in the stored procedure. The missing savepoint was possibly not set by the stored procedure, or the savepoint might was released through database actions that were done outside of the stored procedure.

User Response: Reissue the call to the stored procedure. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4012 **There was an invalid savepoint string storage.**

Explanation: The database transaction savepoint name was not stored correctly in a data structure internal to the stored procedure possibly because not enough memory is available to the stored procedure process.

User Response: Reissue the call to the stored procedure. If the problem persists, then increase the memory available to the fenced stored procedure process. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4013 The savepoint failed to clear.

Explanation: The stored procedure was unable to clear a database transaction savepoint. The stored procedure possibly did not set the missing savepoint, or the savepoint was possibly released through database actions that were done outside of the stored procedure.

User Response: Reissue the call to the stored procedure. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4014 Attempting to determine the DB2 AUTOCOMMIT setting failed.

Explanation: An attempt by the stored procedure to determine whether the DB2 AUTOCOMMIT feature is enabled or disabled failed.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4015 Attempting to set DB2 AUTOCOMMIT OFF failed.

Explanation: An attempt by the stored procedure to disable the DB2 AUTOCOMMIT feature failed.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4016 No data was returned from the CLI call SQLFetch().

Explanation: No data was returned to the stored procedure by the CLI function SQLFetch(). This might be acceptable, but the stored procedure should not have allowed this error to propagate through the stored procedure unchanged.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4017 *Object_name* **object was not properly constructed.**

Explanation: An database object internal to the stored procedure was not initialized properly.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4018 The database disconnect failed.

Explanation: The stored procedure failed to disconnect from the database.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4019 DB2 SQL error - SQLCODE *sqlcode***, SQLSTATE** *sqlstate***, SQLMESG** *sqlmesg***.**

Explanation: An error occurred while executing an SQL statement that was issued by the stored procedure to the database.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4020 DB2 SQL error - No details are available.

Explanation: Diagnostic information is not available for an error that occurred while executing an SQL statement that was issued by the stored procedure to the database.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4021 DB2 SQL error - No details are available.

Explanation: An error occurred while trying to gather diagnostic information for another error that occurred while executing an SQL statement that was issued by the stored procedure to the database.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4022 The allocation of DB2 ***handle_name*** **handle failed.**

Explanation: An error occurred while trying to allocate a DB2 handle in the stored procedure.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4023 Freeing the DB2 ***handle_name*** **handle failed.**

Explanation: An error occurred while trying to free a DB2 handle in the stored procedure.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4028 The transaction was not stopped.

Explanation: An error occurred while trying to end the transaction of the stored procedure.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4029 Duplicate rows found sharing the same name and schema in a main object table.

Explanation: Duplicate rows sharing the same name and schema were unexpectedly found in one of the metadata catalog tables. This sharing is indicative of an internal error in the stored procedure.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4030 The DBINFO structure was not initialized. Make sure that the stored procedure was created in the database by using the DBINFO option.

Explanation: A DBINFO structure was not received by the stored procedure from the database client.

User Response: Ensure that the stored procedure is cataloged in the appropriate database by using the DBINFO option.

4031 Setting the schema as DB2INFO failed.

Explanation: The stored procedure failed to set DB2INFO as the current schema.

User Response: Check the database manager log files on the client and on the server. Reissue the call to the stored procedure.

4032 The operation failed due to a collision between an object in the main object table and the object being inserted.

Explanation: An SQL INSERT statement failed in the stored procedure because it will result in a duplicate metadata object entry in one of the metadata catalog tables.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4033 The operand of the column function includes a column function.

Explanation: A column function nested in another column function was detected in one of the SQL statements issued by the stored procedure. Column functions cannot be nested in SQL statements.

User Response: Modify the SQL expression template for the input attribute or measure object so that nested column functions are no longer present in the SQL statements generated by the stored procedure.

4034 The DB2 ISOLATION LEVEL setting was not determined.

Explanation: An attempt by the stored procedure to determine the database transaction isolation level failed. The isolation level could not be determined.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4035 Setting the DB2 ISOLATION LEVEL to READ STABILITY failed.

Explanation: An attempt by the stored procedure to set the database transaction isolation level failed. The stored procedures requires an isolation level of READ STABILITY.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4036 The DB2 VERSION LEVEL was not found.

Explanation: An attempt by the stored procedure to determine the version level of the database manager failed.

User Response: Check the database manager log files on the client and on the server. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4037 The DB2 VERSION LEVEL is not compatible with this release.

Explanation: The version level of the database manager is not compatible with the version of the stored procedure that was called.

User Response: See the installation and setup documentation for information on database manager and stored procedure compatibility. Ensure that compatible versions of the database manager and the stored procedure are installed on the same server computer.

Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

4038 A SQL statement could not be processed because it is too long or too complex.

Explanation: A statement was issued by the stored procedure that could not be processed because it exceeds a system limit for either length or complexity, or because too many constraints or triggers are involved.

User Response: If possible, increase the size of the statement heap (stmtheap) in the database configuration file. Contact IBM Software Support for further assistance. Provide the status ID and text for the metadata operation that was attempted. If possible, also provide the stored procedure log files from the database server.

5000 The utility failed to parse string *string*.

Explanation: A method internal to the stored procedure encountered an error while parsing an internal string value.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

5001 The utility number format did not initialize successfully. Make sure that enough memory is available.

Explanation: The ICU number formatter was not initialized properly in the stored procedure. This might be the result of inadequate memory resources available to the stored procedure process. This might also be the result of the wrong version of the ICU libraries being loaded by the stored procedure.

User Response: Increase the memory available to the fenced stored procedure process, and reissue the call to the stored procedure. Ensure that the version of the ICU libraries intended for use with the current version of the stored procedure are being loaded. You might need to check the run-time library search path set in the environment to determine correct setup.

Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

5002 The utility resource bundle did not initialize successfully. Error code=*code*. Make sure the bundle exists and it is in the path *path_name*.

Explanation: An ICU resource bundle was not initialized properly in the stored procedure. The improper initialization might be the result of the following problems: adequate memory resources are not available to the stored procedure process; the wrong version of the ICU libraries were loaded by the stored procedure; or the wrong resource bundle was loaded for the stored procedure.

User Response: Increase the memory available to the fenced stored procedure process, and reissue the call to the stored procedure. Ensure that the version of the ICU libraries intended for use with the current version of the stored procedure are being loaded. You might need to check the run-time library search path set in the environment to determine correct setup. Ensure that the correct version of the stored procedure's resource bundle was installed on the database server system.

Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

5003 **The data path from the environment variable** *variable_name* **was not found. Check that the environment variable is set properly.**

Explanation: A DB2 environment variable that is used by the stored procedure is not set.

User Response: Ensure that DB2 was installed correctly on the system. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

5004 **The target stream is closed.**

Explanation: A data stream that is used internally by the stored procedure is unexpectedly closed. There might not be enough filehandles available on the database system.

User Response: Ensure that enough filehandles are available from the operating system. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

5005 **The target is writing the characters by using the default encoding.**

Explanation: The default encoding that is documented for the stored procedure is being used by the stored procedure to write data to files on the database server file system.

User Response: Applications that read the files that are written to by the stored procedure must be able to interpret data encoded in the default encoding of the stored procedure.

5006 **The input log string of** *string* **is not written. The level of the string is** *string_level* **and the level of log is** *log_level***.**

Explanation: The current logging level does not allow the indicated message to be written to one of the log files set for the stored procedure.

User Response: Modify the logging level if the indicated message needs to be written to one of the log files of the stored procedure.

5007 **The message text for the error code** *code* **was not found.**

Explanation: The text for the indicated error code was not found in the resource bundle file of the stored procedure. The wrong version of the resource bundle file might be in use.

User Response: Ensure that the correct version of the resource bundle file of the stored procedure was installed on the database server system. Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

5008 **There was a failure while accessing** *operation* **for the global static MsgBase object.**

Explanation: An error occurred in the stored procedure while trying to access an internal message object.

User Response: Contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

6000 OLAPMSG() failed with error code *code*.

Explanation: The stored procedure failed during execution.

User Response: Based on the return code, either fix the problem and reissue the call to the stored procedure, or contact IBM Customer Support for further assistance. If contacting IBM Customer Support, provide the status ID and text for the metadata operation attempted. If possible, also provide the stored procedure log files from the database server.

6001 The instantiated SQL template(s) for the *object_name* object is invalid with a value of *value*. Reason ID=*ID*, Message *message*.

Explanation: The instantiated SQL template is the SQL statement fragment that can be formed by combining the SQL expression templates for all of the attributes and measures involved in a composite attribute or composite measure. A problem was found with the instantiated SQL template for the specified object.

User Response: See the reason ID and message specified. Reissue the call to the stored procedure after you make any changes that was suggested by the reason message.

6002 The *object1* object references *object2* object, but the *object2* object does not exist in the database.

Explanation: Database objects can reference other objects only if those other objects exist in the database.

User Response: Create the object to be referenced in the database, and then reissue the metadata operation request. Or remove the reference to the missing object, and then reissue the metadata operation request.

6003 The *log_name* log within the specified path could not be opened. Make sure that the specified path exists and that the file has write access.

Explanation: At least one of the log files that was used by the stored procedure could not be opened.

User Response: Ensure that the path specified in the stored procedure configuration file exists. Ensure that user ID running the stored procedure on the database server has the authority to create, read, and write the required log files.

6004 A measure was encountered with *template_names* SQL expression templates and no aggregations.

Explanation: A metadata object rule was violated by the identified measure object. A measure with multiple SQL expression templates must define at least one step in its aggregation script.

User Response: Change the identified measure so that its aggregation script has at least one step. You can also remove one of the measure's SQL expression templates provided the remaining SQL expression template refers to only other measures. See the *Setup and User's Guide* for more information about metadata rules.

6005 The input metadata parameter is unexpectedly empty for this operation. The missing metadata parameter is required for this operation.

Explanation: The requested metadata operation requires that metadata be provided as input. The stored procedure parameter for exchanging metadata is unexpectedly empty.

User Response: Reissue the metadata operation request with the required metadata.

6006 No objects were found matching search criteria: *search_criteria*

Explanation: The metadata operation found no metadata objects that match the specified search criteria. No changes were made to the contents of the metadata catalog.

User Response: Reissue the metadata operation with new search criteria if you want to change the contents of the metadata catalog.

6007 Collision(s) between object(s) in the catalog and object(s) being imported were encountered. No changes were made to the metadata.

Explanation: Collisions were detected between the objects being imported and the objects that already exist in the metadata catalog. Due to the import mode that you specified, no changes were made to the objects in the metadata catalog.

User Response: Reissue the metadata operation by using a different import mode if you want to change the contents of the metadata catalog.

6008 A duplicate *object* **exists within** *metadata_input* **with identity** *ID***.**

Explanation: Duplicate metadata objects were detected in the metadata input for this metadata operation. Duplicate objects are not allowed as input for metadata operations.

User Response: Remove the duplicate metadata object from the input metadata, and reissue the metadata operation.

6009 An object sharing the same identity as the input *object_name* **object already exists in the metadata catalog.**

Explanation: The metadata operation could not be performed because a metadata object with the same identity already exists in the metadata catalog.

User Response: Drop the object from the metadata catalog that shares the same identity as the object being created before reissuing the failed metadata operation. You can also change the existing object to match the properties of the new object being created. Otherwise, you must exclude the new object that is causing this error from the metadata operation being performed.

6010 The reference to the *object_name* **object already exists for the input** *object_name* **object.**

Explanation: A reference between the specified objects is already defined in the metadata catalog. Duplicate references are not allowed.

User Response: Remove one of the duplicate references from the metadata operation request.

6011 The *object_name* **object's schema cannot be changed using the rename operation.**

Explanation: The rename operation cannot be used to change the schema of a metadata object.

User Response: Ensure that the schema that is specified for the object being renamed remains constant, or use the alter operation.

6012 The SQL template *template_name* **is missing token***token_name***.**

Explanation: A metadata object rule was violated by the identified measure object. In the SQL expression template for the measure, token indicators must begin numbering with 1 and must be consecutively numbered.

User Response: Change the identified measure so that the token indicators for its SQL expression templates are consecutively numbered starting with 1. See the *Setup and User's Guide* for more information about metadata rules.

6013 **The version *version1* of the XML schema used by the client is not supported by the API on the server. The API on the server supports version *version2* of the XML schema.**

Explanation: The version of the XML schema that is used by the client and embedded in the input parameter strings is not supported by the version of the stored procedure on the server.

User Response: Ensure that the client application and the stored procedure use the same version of the XML schema published with this product.

6014 **The SQL template(s) for the *object_name* object cannot be formulated. Reason ID *ID*, Message *message*.**

Explanation: The stored procedure formulates the SQL templates for attributes and measures by combining the SQL expression templates for all of the attributes and measures involved in a composite attribute or composite measure. A problem was encountered in the stored procedure during the formulation of a SQL template.

User Response: See the reason ID and message specified. Reissue the call to the stored procedure after you make any changes suggested by the reason message.

6015 **The database user ID does not have the authority to create a database schema in the active database.**

Explanation: The user ID that owns the stored procedure process on the database server does not have the authority to create a database schema in the active database. A database schema is created for each unique metadata object schema.

User Response: Check the database manager log files on the client and on the server. Grant the authority to create a schema in the active database to the user ID that owns the stored procedure process. Reissue the call to the stored procedure.

6016 **The database user ID does not have the authority to perform a required action in the active database. The following error message was returned from the database server: *message*.**

Explanation: The user ID that owns the stored procedure process on the database server does not have the authority to perform a required action in the active database.

User Response: Check the database manager log files on the client and on the server. Grant the required authority to the user ID that owns the stored procedure process. Reissue the call to the stored procedure.

6017 **The *object_name* object does not exist in the metadata catalog.**

Explanation: The requested operation requires that the indicated object exist in the metadata catalog.

User Response: Create the indicated object in the metadata catalog before reissuing the metadata operation request.

6018 **A required table does not exist in the database. The following error message was returned from the database server: *message*.**

Explanation: A table that was required by the requested operation does not exist in the database.

User Response: If the missing table is a user table, then create the table and reissue the metadata operation request. If the missing table is a metadata catalog table or a database system table, then contact IBM Customer Support for further assistance. Provide the status ID and text for the metadata operation that was attempted. If possible, also provide the stored procedure log files from the database server.

6200 **The *object_name* object is not complete. Make sure that the required properties are set.**

Explanation: The specified input object does not have all of its required properties set.

User Response: Set the required properties for the object and reissue the metadata operation request.

6201 **The *hierarchy_name* hierarchy is invalid because it is of type recursive, but is does not have exactly two attributes.**

Explanation: A recursive hierarchy must reference exactly two attributes. The identified hierarchy violates this rule.

User Response: Modify the identified hierarchy to reference exactly two attributes. See the *Setup and User's Guide* for more information about metadata rules.

6202 **The *object_name* object must have at least one SQL template.**

Explanation: Based on the metadata object definitions that is provided in the product documentation, the identified measure must have at least one SQL template defined for it.

User Response: Modify the identified measure so that it has at least one SQL template defined. See the *Setup and User's Guide* for more information about metadata rules.

6203 **The measure has no aggregation, but it contains attr or col ref or no ref at all.**

Explanation: A metadata object rule was violated by the identified measure object. An aggregation is not required for a measure if that measure references at least one other measure and only references measures.

User Response: Change the identified measure by adding an aggregation or ensuring that the identified measure references at least one other measure and references only measures. See the *Setup and User's Guide* for more information about metadata rules.

6204 **The measure has a measure ref with multiple SQL templates, and the measure has an aggregation script (multiple SQL templates or at least one aggregation).**

Explanation: A metadata object rule was violated by the identified measure object. If a measure A refers to a measure B, which defines multiple SQL templates, then measure A must not have an aggregation script. This rule applies for all levels in a measure reference tree.

User Response: Remove the aggregation script from the measure that is causing the problem, or remove one of the SQL expression templates from the measure referenced. See the *Setup and User's Guide* for more information about metadata rules.

6205 **There are two or more aggregations with an empty dimension list.**

Explanation: A metadata object rule was violated by the identified measure object. When a measure defines one or more aggregations, one aggregation must have an empty list of dimensions.

User Response: Change the identified measure so that it has one empty list of dimensions, or change the identified measure so that it defines no aggregations. See the *Setup and User's Guide* for more information about metadata rules.

6206 **The *attribute_name* attribute should have only one SQL template.**

Explanation: Based on the metadata object definitions that are provided in the product documentation, the identified measure must have only one SQL template defined for it.

User Response: Modify the identified measure so that it has only one SQL template defined. See the *Setup and User's Guide* for more information about metadata rules.

6207 **The *attribute_name* attribute is part of a join but has no column reference.**

Explanation: The identified attribute object must refer to a database column for it to be validly referenced by a metadata join object.

User Response: Modify the identified attribute object so that it refers to a database column, or modify the related join object so that it refers to a different attribute object where the different attribute object refers to a database column. See the *Setup and User's Guide* for more information about metadata rules.

6208 **The *attribute_name* attribute is part of a join and it must point to the same table as *table_name*.**

Explanation: The first identified attribute object must refer to the same database table as the other object identified.

User Response: Modify the first identified attribute so that it refers to the same database table as the other object identified, or modify the related join object so that it refers to a different attribute object where the different attribute object refers to the same database table as the other object identified. See the *Setup and User's Guide* for more information about metadata rules.

6209 **The schema of *object_name* object exceeds the maximum length.**

Explanation: The schema of the identified object exceeds the maximum length.

User Response: Shorten the schema for the identified object. See the *Setup and User's Guide* for more information about metadata rules.

6210 **The name of *object_name* object exceeds the maximum length.**

Explanation: The name of the identified object exceeds the maximum length.

User Response: Shorten the name of the identified object. See the *Setup and User's Guide* for more information about metadata rules.

6211 **The table name of *object_name* object exceeds the maximum length.**

Explanation: The table name of the identified object exceeds the maximum length.

User Response: Shorten the name of the table. See the *Setup and User's Guide* for more information about metadata rules.

6212 **The business name of *object_name* object exceeds the maximum length.**

Explanation: The business name of the identified object exceeds the maximum length.

User Response: Shorten the business name. See the *Setup and User's Guide* for more information about metadata rules.

6213 **The comments of *object_name* object exceeds the maximum length.**

Explanation: The comments of the identified object exceeds the maximum length.

User Response: Shorten the comments of the identified objects. See the *Setup and User's Guide* for more information about metadata rules.

6214 **The schema of *object_name* object cannot start with SYS.**

Explanation: The schema for metadata objects cannot begin with the string SYS.

User Response: Use a schema that does not begin with *SYS* for metadata objects. See the *Setup and User's Guide* for more information about metadata rules.

6215 **The schema of *object_name* object cannot be SESSION.**

Explanation: The schema for metadata objects cannot be the string SESSION.

User Response: Use a schema that is not the string SESSION for metadata objects. See the

Setup and User's Guide for more information about metadata rules.

6216 **The name and schema of the** *object_name* **object are not complete. Reason ID=***ID***, Message=***message***.**

Explanation: The name or the schema or both of the identified object is missing or invalid.

User Response: Provide valid strings for both the name and the schema of the identified object. See the *Setup and User's Guide* for more information about metadata rules.

6217 **The** *cube_hierarchy_name* **cube hierarchy is invalid because the** *hierarchy_name* **hierarchy's attribute list is empty, but the cube hierarchy is not.**

Explanation: The identified cube hierarchy refers to attributes that the identified hierarchy does not refer to. These references are invalid.

User Response: Change the identified cube hierarchy to no longer refer to any attributes. Or change the identified hierarchy to refer to the same attributes that the identified cube hierarchy refers to. See the *Setup and User's Guide* for more information about metadata rules.

6299 **At least one database view was found during validation. Constraint-related validation checks were not performed for the joins that involve columns of views. All other validation checks were performed.**

Explanation: Constraint-related validation checks were not performed for those joins that were found to involve columns of views. Constraint-related validation checks were performed on all other joins requested, and all remaining validation checks were performed on all requested objects.

User Response: See the *Setup and User's Guide* for more information about metadata rules, metadata validation, and query optimization.

6300 **The** *model_name* **cube model does not refer to one or more facts.**

Explanation: A metadata object rule was violated by the identified cube model object. A cube model must refer to one or more facts.

User Response: Change the identified cube model so that it refers to one or more facts. See the *Setup and User's Guide* for more information about metadata rules.

6301 **The** *model_name***cube model does not refer to zero or more dimension(s).**

Explanation: A metadata object rule was violated by the identified cube model object. A cube model must refer to zero or more dimensions.

User Response: Change the identified cube model so that it refers to zero or more dimensions. See the *Setup and User's Guide* for more information about metadata rules.

6302 **The** *model_name***cube model is missing a dimension or join or both for one of its dimension-join pairs.**

Explanation: A metadata object rule was violated by the identified cube model object. A dimension-join pair for a cube model must refer to both a dimension and a join.

User Response: Change the identified cube model so that all of its dimension-join pairs refer both a dimension and a join. See the *Setup and User's Guide* for more information about metadata rules.

6303 **The*join_name* join referenced by the *model_name*cube model is not valid. All the attributes on one of its sides must be referenced by the *facts_name*facts, and all of the attributes on the other side must be referenced by one of the cube model's dimensions.**

Explanation: A metadata object rule was violated by the identified cube model object. The joins of a cube model must each refer to the attributes of the cube model's facts on one side, and to the attributes of one of the cube model's dimensions on the other side.

User Response: Change the invalid join for the identified cube model so that all of the attributes on one side of the join come from the cube model's facts, and all of the attributes on the other side of the join come from one of the cube model's dimensions. See the *Setup and User's Guide* for more information about metadata rules.

6304 **One of the aggregations in the *measure_name*measure directly references the *dimension_name* dimension, which is not directly referenced by the *model_name* cube model.**

Explanation: A metadata object rule was violated by the identified cube model object. The aggregations in a measure that are used by a cube model must refer only to those dimensions that are used by the same cube model.

User Response: Change the aggregation for the identified measure so that it refers only to those dimensions used by the identified cube model. See the *Setup and User's Guide* for more information about metadata rules.

6305 **The empty-dimension-list aggregation in the *measure_name* measure does not match to at least one previously unmatched dimension from the *model_name* cube model.**

Explanation: A metadata object rule was violated by the identified cube model object. Empty-dimension-list aggregations in the measures that are used by cube models must match to at least one dimension unmatched otherwise in each cube model.

User Response: Change the aggregation for the identified measure so that its empty-dimension-list matches to at least one previously unmatched dimension in the identified cube model. See the *Setup and User's Guide* for more information about metadata rules.

6306 **The *measure_name* measure must contain only the empty-dimension-list aggregation since the *model_name* cube model does not refer to any dimension objects.**

Explanation: A metadata object rule was violated by the identified cube model object. When a cube model does not refer to any dimensions, the cube model's measure must only contain the empty-dimension-list aggregation.

User Response: Change the identified measure so is contains only the empty-dimension-list aggregation. See the *Setup and User's Guide* for more information about metadata rules.

6307 **The *cube_name* cube does not refer to one cube facts object.**

Explanation: A metadata object rule was violated by the identified cube object. A cube must refer to one cube facts object.

User Response: Change the identified cube so that it refers to one cube facts object. See the *Setup and User's Guide* for more information about metadata rules.

6308 **The *cube_name* cube does not refer to at least one cube dimension object.**

Explanation: A metadata object rule was violated by the identified cube object. A cube must refer to at least one cube dimension object.

User Response: Change the identified cube so that it refers to at least one cube dimension object. See the *Setup and User's Guide* for more information about metadata rules.

6309 **The *cube_facts_name* cube facts referenced by the *cube_name* cube is not derived from the facts object referenced by the *model_name* cube model.**

Explanation: A metadata object rule was violated by the identified cube object. The cube facts that is used by the identified cube must be derived from the facts that is used by the identified cube model.

User Response: Change one or more of the identified objects so that the specified rule is no longer violated. See the *Setup and User's Guide* for more information about metadata rules.

6310 **The *cube_dimension_name* cube dimension referenced by the *cube_name* cube is not derived from one of the dimension objects referenced by the *model_name* cube model.**

Explanation: A metadata object rule was violated by the identified cube object. A cube dimension that is used by the identified cube must be derived from one of the dimensions that is used by the identified cube model.

User Response: Change one or more of the identified objects so that the specified rule is no longer violated. See the *Setup and User's Guide* for more information about metadata rules.

6311 **The *facts_name* facts object does not refer to any measures.**

Explanation: A metadata object rule was violated by the identified facts object. A facts object must refer to at least one measure.

User Response: Change the identified facts so that it refers to at least one measure. See the *Setup and User's Guide* for more information about metadata rules.

6312 **Some of the attributes and measures that are referenced by the facts object *facts_name* cannot be joined using facts object joins.**

Explanation: A metadata object rule was violated by the identified facts object. The attributes and measures of a facts object must all be joinable using the join objects of the facts.

User Response: Make all the attributes and measures referenced by the identified facts object joinable by referencing more join objects from the facts object. Or remove those attributes from the facts object or measures that are not joinable by the facts' current joins. See the *Setup and User's Guide* for more information about metadata rules.

6313 **The *facts_name* facts object has multiple joins between two tables.**

Explanation: A metadata object rule was violated by the identified facts object. A facts object must not have multiple joins between the same two tables.

User Response: Change the identified facts object so that it has only one join between any two given tables. See the *Setup and User's Guide* for more information about metadata rules.

6314 **The *facts_name* facts object contains a join loop.**

Explanation: A metadata object rule was violated by the identified facts object. The joins for the identified facts object form a path loop. This is not allowed.

User Response: Remove one of the joins that is causing the loop from the identified facts object, or change one of the joins that is causing the loop so that a loop no longer exists. See the *Setup and User's Guide* for more information about metadata rules.

6315 **The *join_name* join does not refer to only those attributes in the *facts_name* facts object.**

Explanation: A metadata object rule was violated by the identified facts object. The joins of a facts object must refer only to the attributes of the that facts object.

User Response: Change the identified join so that it refers only to the attributes of the identified facts object, or add to the facts object the missing attributes that the identified join object refers to. See the *Setup and User's Guide* for more information about metadata rules.

6316 **The *cube_facts_name* cube facts object does not refer to a fact object or refers to more than one fact object.**

Explanation: A metadata object rule was violated by the identified cube facts object. A cube facts object must refer to one facts object.

User Response: Change the identified cube facts object so that it refers to one facts object. See the *Setup and User's Guide* for more information about metadata rules.

6317 **The *cube_facts_name* cube facts object does not refer to any measures.**

Explanation: A metadata object rule was violated by the identified cube facts object. A cube facts object must refer to at least one measure.

User Response: Change the identified cube facts object so that it refers at least one measure. See the *Setup and User's Guide* for more information about metadata rules.

6318 **The *measure_name* measure that is referenced by the *cube_facts_name* cube facts object is not a part of the *facts_name* facts object.**

Explanation: A metadata object rule was violated by the identified cube facts object. A cube facts object must refer to measures that are referred to by the facts object from which the cube facts object was derived.

User Response: Add the identified measure to the identified facts object, or remove the identified measure from the identified cube facts object. See the *Setup and User's Guide* for more information about metadata rules.

6319 **The *dimension_name* dimension does not refer any attributes. A dimension must refer to at least one attribute.**

Explanation: A metadata object rule was violated by the identified dimension object. A dimension object must refer to at least one attribute.

User Response: Change the identified dimension object so that it refers to at least one attribute. See the *Setup and User's Guide* for more information about metadata rules.

6320 **Some of the attributes referenced by the *dimension_name* dimension cannot be joined using dimension joins.**

Explanation: A metadata object rule was violated by the identified dimension object. The attributes of a dimension object must all be joinable by using the join objects of the dimension.

User Response: Make all the attributes referenced by the identified dimension object joinable by referencing more join objects from the dimension object. Or remove from the dimension object those attributes that are not joinable by the dimension's current joins. See the *Setup and User's Guide* for more information about metadata rules.

6321 The *dimension_name* dimension contains a join loop.

Explanation: A metadata object rule was violated by the identified dimension object. The joins for the identified dimension object form a path loop. This is not allowed.

User Response: Remove one of the joins that is causing the loop from the identified dimension object, or change one of the joins that is causing the loop so that a loop no longer exists. See the *Setup and User's Guide* for more information about metadata rules.

6322 The *dimension_name* dimension has multiple joins between two tables.

Explanation: A metadata object rule was violated by the identified dimension object. A dimension object must not have multiple joins between the same two tables.

User Response: Change the identified dimension object so that it has only one join between any two tables. See the *Setup and User's Guide* for more information about metadata rules.

6323 The *hierarchy_name* hierarchy that is referenced by the *dimension_name*dimension references attributes that are not also referenced by the same dimension.

Explanation: A metadata object rule was violated by the identified dimension object. The hierarchies of a dimension object must only refer to the attributes of that dimension object.

User Response: Change the identified hierarchy so that it only refers to the attributes of the identified dimension object, or add to the dimension object the missing attributes that the identified hierarchy object refers to. See the *Setup and User's Guide* for more information about metadata rules.

6324 The *relationship_name* attribute relationship, used by the *hierarchy_name* hierarchy, references attributes that are not also referenced by the *dimension_name* dimension.

Explanation: A metadata object rule was violated by the identified dimension object. The attribute relationship of a dimension object must refer only to the attributes of that dimension object.

User Response: Change the identified attribute relationship so that it only refers to the attributes of the identified dimension object, or add to the dimension object the missing attributes that the identified attribute relationship object refers to. See the *Setup and User's Guide* for more information about metadata rules.

6325 The *join_name* join does not refer to only those attributes in the *dimension_name* dimension.

Explanation: A metadata object rule was violated by the identified dimension object. The join of a dimension object must only refer to the attributes of that dimension object.

User Response: Change the identified join so that it only refers to the attributes of the identified dimension object, or add to the dimension object the missing attributes that the identified join object refers to. See the *Setup and User's Guide* for more information about metadata rules.

6326 The *cube_dimension_name* cube dimension does not refer to a dimension.

Explanation: A metadata object rule was violated by the identified cube dimension object. A cube dimension object must refer to a dimension.

User Response: Change the identified cube dimension object so that it refers to a dimension. See the *Setup and User's Guide* for more information about metadata rules.

6327 **The *cube_dimension_name* cube dimension does not refer to a cube hierarchy.**

Explanation: A metadata object rule was violated by the identified cube dimension object. A cube dimension object must refer to a cube hierarchy.

User Response: Change the identified cube dimension object so that it refers to a cube hierarchy. See the *Setup and User's Guide* for more information about metadata rules.

6328 **The *cube_hierarchy_name* cube hierarchy that is referenced by the *cube_dimension_name* cube dimension is not derived from any of the hierarchies that are referenced by the *dimension_name* dimension.**

Explanation: A metadata object rule was violated by the identified cube dimension object. The cube hierarchy used by the identified cube dimension must be derived from one of the hierarchies that is used by the identified dimension.

User Response: Change one or more of the identified objects so that the specified rule is no longer violated. See the *Setup and User's Guide* for more information about metadata rules.

6329 **The *hierarchy_name* hierarchy does not refer to any attributes.**

Explanation: A metadata object rule was violated by the identified hierarchy object. A hierarchy object must refer to at least one attribute.

User Response: Change the identified hierarchy object so that it refers to at least one attribute. See the *Setup and User's Guide* for more information about metadata rules.

6330 **The *hierarchy_name* hierarchy, which uses a recursive deployment, does not reference exactly two attributes.**

Explanation: A metadata object rule was violated by the identified hierarchy object. A hierarchy object that uses recursive deployment must reference two attributes.

User Response: Change the identified hierarchy object so that it refers to two attributes. See the *Setup and User's Guide* for more information about metadata rules.

6331 **The left attribute for the *relationship_name* attribute relationship is not a part of the *hierarchy_name* hierarchy.**

Explanation: A metadata object rule was violated by the identified hierarchy object. When an attribute relationship is referenced by a hierarchy, the left attribute of that attribute relationship must be a part of the referencing hierarchy.

User Response: Change the identified attribute relationship object so that its left attribute is a part of the identified hierarchy. See the *Setup and User's Guide* for more information about metadata rules.

6332 **The type of the *hierarchy_name* hierarchy is not compatible with its deployment.**

Explanation: A metadata object rule was violated by the identified hierarchy object. Compatibility of hierarchy types and deployment are described in the *Setup and User's Guide*.

User Response: Change the identified hierarchy so that its type is compatible with its deployment. See the *Setup and User's Guide* for more information about metadata rules.

6333 **The** *relationship_name* **attribute relationship, which is referenced by the** *hierarchy_name* **hierarchy, does not have a cardinality of either 1:1 or N:1.**

Explanation: A metadata object rule was violated by the identified hierarchy object. When an attribute relationship is referenced by a hierarchy, then the cardinality of the attribute relationship must be either 1:1 or N:1.

User Response: Change the identified attribute relationship so that its cardinality is either 1:1 or N:1, or remove the identified attribute relationship from the identified hierarchy. See the *Setup and User's Guide* for more information about metadata rules.

6334 **The** *cube_hierarchy_name* **cube hierarchy does not refer to just one hierarchy.**

Explanation: A metadata object rule was violated by the identified cube hierarchy object. A cube hierarchy object must refer to one hierarchy.

User Response: Change the identified cube hierarchy object so that it refers to one hierarchy. See the *Setup and User's Guide* for more information about metadata rules.

6335 **The** *cube_hierarchy_name* **cube hierarchy does not refer to at least one attribute.**

Explanation: A metadata object rule was violated by the identified cube hierarchy object. A cube hierarchy object must refer to at least one attribute.

User Response: Change the identified cube hierarchy object so that it refers to at least one attribute. See the *Setup and User's Guide* for more information about metadata rules.

6336 **The** *attribute_name* **attribute, which is referenced by the** *cube_hierarchy_name* **cube hierarchy, is not also referenced by the** *hierarchy_name* **hierarchy.**

Explanation: A metadata object rule was violated by the identified cube hierarchy object. A cube hierarchy object must refer to attributes that are referred to by the hierarchy object from which the cube hierarchy object was derived.

User Response: Add the identified attribute to the identified hierarchy object, or remove the identified attribute from the identified cube hierarchy object. See the *Setup and User's Guide* for more information about metadata rules.

6337 **The order of the attributes in the** *cube_hierarchy_name* **cube hierarchy does not match the order of the same attributes in the** *hierarchy_name* **hierarchy.**

Explanation: A metadata object rule was violated by the identified cube hierarchy object. The relative ordering of the attributes in a cube hierarchy must be the same as the relative ordering of the same attributes in the hierarchy from which the cube hierarchy was derived.

User Response: Change one of the identified objects so that the relative ordering of the attributes in both identified objects is consistent. See the *Setup and User's Guide* for more information about metadata rules.

6338 **The left attribute for** *relationship_name* **attribute relationship is not a part of** *cube_hierarchy_name* **cube hierarchy.**

Explanation: A metadata object rule was violated by the identified cube hierarchy object. When an attribute relationship is referenced by a cube hierarchy, the left attribute of that attribute relationship must be a part of the referencing cube hierarchy.

User Response: Change the identified attribute relationship object so that its left attribute is a part of the identified cube hierarchy. See the *Setup and User's Guide* for more information about metadata rules.

6339 **The** *relationship_name* **attribute relationship, which is referenced by the** *cube_hierarchy_name* **cube hierarchy, is not also referenced by the** *hierarchy_name* **hierarchy.**

Explanation: A metadata object rule was violated by the identified cube hierarchy object. A cube hierarchy object must refer to attribute relationship object that are referred to by the hierarchy object from which the cube hierarchy object was derived.

User Response: Add the identified attribute relationship object to the identified hierarchy object, or remove the identified attribute relationship object from the identified cube hierarchy object. See the *Setup and User's Guide* for more information about metadata rules.

6340 **One of the SQL expression templates for the** *measure_name* **measure uses a parameter that is not an attribute, measure, or column.**

Explanation: A metadata object rule was violated by the identified measure object. The SQL expression templates for measure objects must use parameters that are attributes, measures, or columns.

User Response: Change the identified measure so that its SQL expression templates use attributes, measures, or columns as parameters. See the *Setup and User's Guide* for more information about metadata rules.

6341 **A dependency loop exists among the attributes or measures used as parameters in the SQL expression template for the** *measure_name* **measure.**

Explanation: A metadata object rule was violated by the identified measure object. The attributes and measures that are used as parameters for the SQL expression template of a measure must not form a dependency loop.

User Response: Change the identified measure so that its SQL expression templates do not contain dependency loops that involve their parameters. See the *Setup and User's Guide* for more information about metadata rules.

6342 **The** *measure_name* **measure has an empty string defined for one of its SQL expression templates.**

Explanation: A metadata object rule was violated by the identified measure object. The SQL expression template for a measure cannot be an empty string.

User Response: Change the identified measure so that its SQL expression template is no longer an empty string. See the *Setup and User's Guide* for more information about metadata rules.

6343 **The SQL expression template for the** *measure_name* **measure contains an aggregation function.**

Explanation: A metadata object rule was violated by the identified measure object. The SQL expression template for a measure cannot contain an aggregation function.

User Response: Change the identified measure so that its SQL expression template no longer contains an aggregation function. See the *Setup and User's Guide* for more information about metadata rules.

6344 **The** *measure_name* **measure is missing an aggregation or is incorrectly referencing objects other than measures.**

Explanation: A metadata object rule was violated by the identified measure object. An aggregation is not required for a measure if that measure references at least one other measure and references only measures.

User Response: Change the identified measure by adding an aggregation or ensuring that the identified measure references at least one other measure and references only measures. See the *Setup and User's Guide* for more information about metadata rules.

6345 **The number of SQL expression templates in the *measure_name* measure does not match the number of parameters used with the first aggregation function.**

Explanation: A metadata object rule was violated by the identified measure object. The number of SQL templates in a measure must match the number of parameters for the first aggregation function of that measure if an aggregation is present.

User Response: Change the identified measure so that the number of parameters for its first aggregation function matches the number of SQL expression templates in the measure. See the *Setup and User's Guide* for more information about metadata rules.

6346 **The *measure_name* measure, which has multiple SQL expression templates, does not define at least one step in the aggregation script.**

Explanation: A metadata object rule was violated by the identified measure object. A measure with multiple SQL expression templates must define at least one step in its aggregation script.

User Response: Change the identified measure so that its aggregation script has at least one step. Or remove one of the measure's SQL expression templates provided the remaining SQL expression template refers to only other measures. See the *Setup and User's Guide* for more information about metadata rules.

6347 **The *measure_name1* measure has an aggregation script defined. However, it should not have any aggregation scripts defined because the referenced measure, *measure_name2*, defines multiple templates for SQL expressions.**

Explanation: A metadata object rule was violated by the identified measure object. If measure A refers to measure B, which defines multiple SQL templates, then measure A must not have an aggregation script. This rule applies for all levels in a measure reference tree.

User Response: Remove the aggregation script from the measure causing the problem, or remove one of the SQL expression templates from the measure referenced. See the *Setup and User's Guide* for more information about metadata rules.

6348 **The *measure_name* measure contains a multi-parameter aggregation function that is not used as the first aggregation.**

Explanation: A metadata object rule was violated by the identified measure object. A multi-parameter aggregation function can only be used as the first aggregation for a measure.

User Response: Make the multi-parameter aggregation function the first aggregation that is used by the identified measure, or remove the multi-parameter aggregation function from the identified measure. See the *Setup and User's Guide* for more information about metadata rules.

6349 **The *measure_name* measure does not have exactly one empty-dimension-list aggregation.**

Explanation: A metadata object rule was violated by the identified measure object. When a measure defines one or more aggregations, one aggregation must have an empty list of dimensions.

User Response: Change the identified measure so that it has one empty list of dimensions, or change the identified measure so that it defines no aggregations. See the *Setup and User's Guide* for more information about metadata rules.

6350 **The** *dimension_name* **dimension is referenced multiple times in the** *measure_name* **measure.**

Explanation: A metadata object rule was violated by the identified measure object. In a measure, a dimension cannot be referenced more than once either in an aggregation or across aggregations.

User Response: Change the identified measure so that it references the identified dimension only once. See the *Setup and User's Guide* for more information about metadata rules.

6351 **The SQL expression template for the** *object_name* **object is missing a token indicator with the number** *number***. Token indicators must be consecutively numbered starting with the number 1.**

Explanation: A metadata object rule was violated by the identified measure object. In a measure's SQL expression template, token indicators must begin with 1 and must be consecutively numbered.

User Response: Change the identified measure so that the token indicators for its SQL expression templates are consecutively numbered starting with 1. See the *Setup and User's Guide* for more information about metadata rules.

6352 **The***measure_name* **measure contains an SQL expression template that does not use the provided reference,** *reference***.**

Explanation: A metadata object rule was violated by the identified measure object. The SQL expression template for a measure must make use of every column, attribute, and measure reference that is provided. Each reference can be used more than once.

User Response: Change the SQL expression template for the identified measure so that it makes use of every column, attribute, and measure reference that was provided. Or remove the column, attribute, and measure references that are not used by the identified measure's SQL expression templates. See the *Setup and User's Guide* for more information about metadata rules.

6353 **One of the SQL expression templates for the** *attribute_name* **attribute uses a parameter that is not an attribute or column.**

Explanation: A metadata object rule was violated by the identified attribute object. The SQL expression templates for attribute objects must use parameters that are attributes or columns.

User Response: Change the identified attribute so that its SQL expression templates use attributes or columns as parameters. See the *Setup and User's Guide* for more information about metadata rules.

6354 **A dependency loop exists among the attributes used as parameters in the SQL expression template for the** *attribute_name* **attribute.**

Explanation: A metadata object rule was violated by the identified attribute object. The attributes used as parameters for the SQL expression template of an attribute must not form a dependency loop.

User Response: Change the identified attribute so that its SQL expression templates do not contain dependency loops involving their parameters. See the *Setup and User's Guide* for more information about metadata rules.

6355 **The** *attribute_name* **attribute has an empty string defined in one of its SQL expression templates.**

Explanation: A metadata object rule was violated by the identified attribute object. The SQL expression template for an attribute cannot be an empty string.

User Response: Change the identified attribute so that its SQL expression template is no longer an empty string. See the *Setup and User's Guide* for more information about metadata rules.

6356 The SQL expression template for the *attribute_name* attribute contains an aggregation function.

Explanation: A metadata object rule was violated by the identified attribute object. The SQL expression template for an attribute cannot contain an aggregation function.

User Response: Change the identified attribute so that its SQL expression template no longer contains an aggregation function. See the *Setup and User's Guide* for more information about metadata rules.

6357 The SQL expression template for the *object_name* object is missing a token indicator with a number *number*. Token indicators must be consecutively numbered starting with the number 1.

Explanation: A metadata object rule was violated by the identified attribute object. in an attribute's SQL expression template, token indicators must begin numbering with 1 and must be consecutively numbered.

User Response: Change the identified attribute so that the token indicators for its SQL expression templates are consecutively numbered starting with 1. See the *Setup and User's Guide* for more information about metadata rules.

6358 The *attribute_name* attribute contains an SQL expression template that does not use the provided reference, *reference*.

Explanation: A metadata object rule was violated by the identified attribute object. The SQL expression template for an attribute must make use of every column and attribute reference that is provided. Each reference can be used more than once.

User Response: Change the SQL expression template for the identified attribute so that it makes use of every column and attribute reference that is provided. Or remove the column and attribute references that are not used by the identified attribute's SQL expression templates. See the *Setup and User's Guide* for more information about metadata rules.

6359 The*relationship_name* attribute relationship does not refer to two distinct attributes.

Explanation: A metadata object rule was violated by the identified attribute relationship object. An attribute relationship object must refer to two distinct attributes.

User Response: Change the identified attribute relationship object so that it refers to two distinct attributes. See the *Setup and User's Guide* for more information about metadata rules.

6360 The *relationship_name* attribute relationship is incorrectly defined. The cardinality property is set to N:N, but the functional dependency property is set to YES.

Explanation: A metadata object rule was violated by the identified attribute relationship object. When the functional dependency property of an attribute relationship is set to YES, then the cardinality property of the attribute relationship cannot be set to N:N.

User Response: Change the identified attribute relationship so that its cardinality is not set to N:N, or its functional dependency property is set to NO. See the *Setup and User's Guide* for more information about metadata rules.

6361 The *join_name* join does not refer to at least one triplet. A triplet contains a left attribute, a right attribute, and an operator.

Explanation: A metadata object rule was violated by the identified join object. A join object must refer to at least one triplet that contains a left attribute, a right attribute, and an operator.

User Response: Change the identified join object so that it refers to at least one triplet. See the *Setup and User's Guide* for more information about metadata rules.

6362 **The left attributes in the *join_name* join do not all resolve into a column or columns of a single table.**

Explanation: A metadata object rule was violated by the identified join object. The left attributes of a join must all resolve into a column or the columns of a single database table.

User Response: Change the identified join object so that its left attributes all resolve into a column or the columns of a single table. Or change the left attributes of the identified join object so that they all comply with the metadata rule stated. See the *Setup and User's Guide* for more information about metadata rules.

6363 **The right attributes in the *join_name* join do not all resolve into a column or columns of a single table.**

Explanation: A metadata object rule was violated by the identified join object. The right attributes of a join must all resolve into a column or the columns of a single database table.

User Response: Change the identified join object so that its right attributes all resolve into a column or the columns of a single table. Or change the right attributes of the identified join object so that they all comply with the metadata rule stated. See the *Setup and User's Guide* for more information about metadata rules.

6364 **At least one of the triplets for the *join_name* join does not define a valid operation. The data types of the left and right attributes might not be compatible with each other, or they might not be compatible with the operator.**

Explanation: A metadata object rule was violated by the identified join object. Each triplet of a join object must define a valid operation. The data types for the right and left attributes must be compatible with each other taking into account the operation specified.

User Response: Change the identified join object so that each of its triplets define a valid operation. See the *Setup and User's Guide* for more information about metadata rules.

6365 **The *model_name* cube model does not refer to one and only one facts object.**

Explanation: A metadata object rule was violated by the identified cube model object. A complete cube model must refer to one facts object.

User Response: Change the identified cube model object so that it refers to one facts object. See the *Setup and User's Guide* for more information about metadata rules.

6366 **The *model_name* cube model does not refer to one or more dimensions.**

Explanation: A metadata object rule was violated by the identified cube model object. A complete cube model must refer to at least one dimension object.

User Response: Change the identified cube model object so that it refers to at least one dimension object. Refer to the product documentation for more information on metadata rules.

6367 The cardinality of the*join_name* facts-to-dimension join is not set to either 1:1 or N:1.

Explanation: An optimization rule was violated by the identified join object. The cube model will not benefit from the Optimization Advisor recommendations because the cardinality of the facts-to-dimension join is not 1:1 or N:1. Optimization will not be performed.

User Response: For the cube model to benefit from the Optimization Advisor recommendations, the cardinality for each of the joins that go from the facts to a dimension object must be set to either 1:1 or N:1. The cardinality of the join on the facts attributes must be 1 or N, and the cardinality of the dimension's attributes must be 1. See the *Setup and User's Guide* for more information about optimization rules.

6368 The *join_name* facts-to-dimension join does not join the table for the *facts_name* facts object to a primary table for the *dimension_name* dimension.

Explanation: An optimization rule was violated by the identified join object. Considering the join network formed by the dimension's joins, you must have at least one table (the primary table) in which all joins radiating from this table have a cardinality of N:1 or 1:1. In the cube model, the joins from the facts to the dimension objects must involve this primary table of a dimension.

User Response: In the cube model object, make sure that all of the facts-to-dimension joins are from the facts object to the primary table of each dimension. See the *Setup and User's Guide* for more information about optimization rules.

6369 The *dimension_name* dimension does not have a primary table, as indicated by the join network formed by the joins for the dimension.

Explanation: An optimization rule was violated by the identified dimension object. Considering the join network formed by the dimension's joins, you must have at least one table in which all joins radiating from this table have a cardinality of N:1 or 1:1. Optimization will not be performed if there is no such primary table for a dimension.

User Response: Check the cardinalities of the join objects used in the dimension. For optimization to be performed the dimension must have a primary table as described in the optimization rules. See the *Setup and User's Guide* for more information about optimization rules.

6370 The *join_name* join involves columns on which a referential constraint is not defined.

Explanation: An optimization rule was violated by the identified join object. You must define a constraint on the columns that participate in the join. If the join is a self-join, that is, the same set of columns is used in both sides of the equality, a primary key must be defined that matches the set of columns. In all other cases, when the set of columns of one side are different from the other side of the join, a primary key must match the columns of one side of the join, and a foreign key must match the other set of columns and reference the primary key. Optimization will not be performed due to the missing constraint.

User Response: Create a constraint on the columns that participate in the join. If you do not want the standard constraint because of performance implications, create informational constraints, with query optimization enabled. See the *Setup and User's Guide* for more information about optimization rules.

6371 A primary key is not defined using the columns involved in the *join_name* self-join.

Explanation: An optimization rule was violated by the identified join object. You must define a constraint on the columns that participate in the join. If the join is a self-join, that is, the same set of columns is used in both sides of the equality, a primary key must be defined that matches the set of columns. Optimization will not be performed due to the missing constraint.

User Response: If the table has a primary key defined, set the attributes of the self-join to attributes representing the primary key columns of the table. Otherwise, create a primary key on the columns that participate in the self-join. See the *Setup and User's Guide* for more information about optimization rules.

6372 A primary key is not defined using the columns from one side of the *join_name* join.

Explanation: An optimization rule was violated by the identified join object. You must define a constraint defined on the columns that participate in the join. When the set of columns of one side are different from the other side of the join, a primary key must match the columns of one side of the join, and a foreign key must match the other set of columns and reference the primary key. Optimization will not be performed due to the missing constraint.

User Response: Create a primary key on the columns of one side of the join. See the *Setup and User's Guide* for more information about optimization rules.

6373 A foreign key is not defined using the columns from one side of the *join_name* join.

Explanation: An optimization rule was violated by the identified join object. You must define a constraint defined on the columns that participate in the join. When the set of columns of one side are different from the other side of the join, a primary key must match the columns of one side of the join, and a foreign key must match the other set of columns and reference the primary key. Optimization will not be performed due to the missing constraint.

User Response: Create a foreign key constraint between the primary key columns of the join and the columns of the other side of the join. If you do not want the standard constraint because of performance implications, create informational constraints with query optimization enabled. See the *Setup and User's Guide* for more information about optimization rules.

6374 The foreign key using the columns from one side of the *join_name* join does not reference the primary key using the columns from the other side of the join.

Explanation: An optimization rule was violated by the identified join object. You must define a constraint on the columns that participate in the join. When the set of columns of one side is different from the other side of the join, a primary key must match the columns of one side of the join, and a foreign key must match the other set of columns and reference the primary key. Optimization will not be performed due to the missing constraint.

User Response: Create a foreign key constraint between the primary key columns of the join and the columns of the other side of the join. If you do not want the standard constraint because of performance implications, create informational constraints with query optimization enabled. See the *Setup and User's Guide* for more information about optimization rules.

6375 The cardinality of the *join_name* join is not set to 1:1, N:1, or 1:N.

Explanation: An optimization rule was violated by the identified join object. Optimization cannot be performed if the join cardinality is M:N.

User Response: Set the join cardinality to 1:1, 1:N or N:1, depending on the constraints on which the join is based. See the *Setup and User's Guide* for more information about optimization rules.

6376 The cardinality of the *join_name* self-join is not set to 1:1.

Explanation: An optimization rule was violated by the identified join object. Optimization cannot be performed if the join cardinality of a self-join is not set to 1:1.

User Response: Set the cardinality of the self-join to 1:1. See the *Setup and User's Guide* for more information about optimization rules.

6377 **The cardinality of the *join_name* join is not set to 1 for the side on which the primary key is defined.**

Explanation: An optimization rule was violated by the identified join object. The join cardinality must be 1 on the side in which a primary key is defined and N on the side in which a foreign key is defined. If the foreign key side has also a primary key defined on it, a 1 must be used as cardinality. Optimization cannot be performed if this is not the case.

User Response: The join cardinality should be set to 1 for the side on which the primary key is defined. See the *Setup and User's Guide* for more information about optimization rules.

6378 **The cardinality of the *join_name* join is not set to N for the side on which the foreign key is defined.**

Explanation: An optimization rule was violated by the identified join object. The join cardinality must be 1 on the side which a primary key is defined and N on the side which a foreign key is defined. If the foreign key side has also a primary key defined on it, a 1 must be used as cardinality. Optimization cannot be performed if this is not the case.

User Response: The join cardinality should be set to N for the side on which the foreign key is defined. See the *Setup and User's Guide* for more information about optimization rules.

6379 **The cardinality of the *join_name* join is not set to 1 for the side on which both a primary key and a foreign key are defined.**

Explanation: An optimization rule was violated by the identified join object. The join cardinality must be 1 on the side which a primary key is defined and 1 for the side on which both a primary key and a foreign key are defined. Optimization cannot be performed if this is not the case.

User Response: The join cardinality should be set to 1:1. See the *Setup and User's Guide* for more information about optimization rules.

6380 **The *attribute_name* attribute, which is referenced by the *join_name* join, does not resolve to a nonnullable SQL expression.**

Explanation: An optimization rule was violated by the identified join object. All attributes that are used in the join must resolve to nonnullable SQL expressions. Optimization cannot be performed if a join references an attribute that resolves to a nullable SQL expression.

User Response: Remove the reference to the nullable attribute from the join. See the *Setup and User's Guide* for more information about optimization rules.

6381 **The *join_name* join does not have a type of INNER JOIN.**

Explanation: An optimization rule was violated by the identified join object. The join type must set to INNER JOIN. Optimization cannot be performed.

User Response: Change the join to reference only attributes that resolve to a single column. See the *Setup and User's Guide* for more information about optimization rules.

6382 **The *attribute_name* attribute reference of *join_name* join does not resolve to a single column expression, which is a requirement for it to participate in a constraint.**

Explanation: An optimization rule was violated by the identified join object. DB2 constraints must be applied on the attributes referenced by a join. Constraints can only be applied on columns, so the attributes referenced by a join must resolve to single column in a table. Optimization cannot be performed if this is not the case.

User Response: Change the join to reference only attributes that resolve to a single column. See the *Setup and User's Guide* for more information about optimization rules.

6383 **The right attribute for the *relationship_name* attribute relationship is incorrectly a part of the *hierarchy_name* hierarchy.**

Explanation: A metadata object rule was violated by the identified hierarchy object. When an attribute relationship is referenced by a hierarchy, the right attribute of that attribute relationship cannot be a part of the referencing hierarchy.

User Response: Alter the identified attribute relationship or hierarchy object so that the right attribute of the attribute relationship is no longer a part of the identified hierarchy. See "General metadata properties" on page 19 for more information about metadata rules.

6384 **The right attribute for the *relationship_name* attribute relationship is incorrectly a part of the *cubehierarchy_name* cube hierarchy.**

Explanation: A metadata object rule was violated by the identified cube hierarchy object. When an attribute relationship is referenced by a cube hierarchy, the right attribute of that attribute relationship cannot be a part of the referencing cube hierarchy.

User Response: Alter the identified attribute relationship or cube hierarchy object so that the right attribute of the attribute relationship is no longer a part of the identified cube hierarchy. See "General metadata properties" on page 19 for more information about metadata rules.

6500 **This operation cannot be performed because the SQL template for the *attribute_name* attribute or measure still involves references to other attributes, measures, or columns. These references must be dropped prior to the execution of this operation.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The the SQL expression template for the identified object involves references to other attributes, measures, or columns that must be removed from the identified object prior to the execution of this operation.

User Response: Before you drop the identified object, change the identified object so that its SQL expression template no longer references attributes, measures, or columns. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6501 **The operation cannot be performed because the *attribute_name* attribute or measure is referenced by another attribute or measure.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified attribute or measure is currently referenced by another attribute or measure, so the identified attribute or measure cannot be dropped.

User Response: Before you drop the identified attribute or measure, change the referencing objects so that they no longer reference the identified attribute or measure. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6502 **The operation cannot be performed because the *dimension_name* dimension is referenced by an aggregation that is defined in a measure.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified dimension is currently referenced by an aggregation of a measure, so the identified dimension cannot be dropped.

User Response: Before you drop the identified dimension, change the referencing objects so that they no longer reference the identified dimension. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6503 **The operation cannot be performed for the** *object_name* **object. A cube hierarchy must reference attributes that are already referenced by the hierarchy that was used to derive the cube hierarchy.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The problem occurred because of one of the following situations:

- An attempt to remove an attribute from a hierarchy was made where the attribute being removed is still being used by a related cube hierarchy.
- An attempt to add an attribute to a cube hierarchy was made where the attribute being added is not already being used by a related hierarchy.

User Response: Perform one of the following actions:

- Remove attributes from cube hierarchies before removing the same attributes from related hierarchies.
- Add attributes to hierarchies before adding the same attributes to related cube hierarchies.

See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6504 **The operation cannot be performed for the** *object_name* **object. A cube hierarchy must reference attribute relationships that are already referenced by the hierarchy that was used to derive the cube hierarchy.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The problem occurred because of one of the following situations:

- An attempt to remove an attribute relationship from a hierarchy was made where the attribute relationship being removed is still being used by a related cube hierarchy.
- An attempt to add an attribute relationship to a cube hierarchy was made where the attribute relationship being added is not already being used by a related hierarchy.

User Response: Perform one of the following actions:

- Remove attribute relationships from cube hierarchies before removing the same attribute relationships from related hierarchies.
- Add attribute relationships to hierarchies before adding the same attribute relationships to related cube hierarchies.

See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6505 **The operation cannot be performed because the** *hierarchy_name* **hierarchy is referenced by a cube hierarchy.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified hierarchy is currently referenced by a cube hierarchy, so the identified hierarchy cannot be dropped.

User Response: Before you drop the identified hierarchy, change the referencing objects so that they no longer reference the identified hierarchy.

See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6506 The operation cannot be performed for the *object_name* **object. A cube facts must reference measures that are already referenced by the facts that was used to derive the cube facts.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The problem occurred because of one of the following situations:

- An attempt to remove a measure from a facts was made, where the measure being removed is still being used by a related cube facts.
- An attempt to add a measure to a cube facts was made, where the measure being added is not already being used by a related facts.

User Response: Perform one of the following actions:

- Remove measures from cube facts before removing the same measures from related facts.
- Add measures to facts before adding the same measures to related cube facts.

See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6507 The operation cannot be performed because the *facts_name* **facts object is referenced by a cube facts object.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified facts is currently referenced by a cube facts, so the identified facts cannot be dropped.

User Response: Before you drop the identified facts, change the referencing objects so that they no longer reference the identified facts. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6508 The operation cannot be performed because the *hierarchy_name* **hierarchy is referenced by a dimension.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified hierarchy is currently referenced by a dimension, so the identified hierarchy cannot be dropped.

User Response: Before you drop the identified hierarchy, change the referencing objects so that they no longer reference the identified hierarchy. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6509 The operation cannot be performed because the *join_name* **join is referenced by a facts object.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified join is currently referenced by a facts, so the identified join cannot be dropped.

User Response: Before you drop the identified join, change the referencing objects so that they no longer reference the identified join. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6510 The operation cannot be performed because the *cube_dimension_name***cube dimension is referenced by a cube.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified cube dimension is currently referenced by a cube, so the identified cube dimension cannot be dropped.

User Response: Before you drop the identified cube dimension, change the referencing objects so that they no longer reference the identified cube dimension. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6511 **The operation cannot be performed for the *object_name*object. The cube dimensions of a cube must be derived from the dimensions referenced by the cube model from which the cube was derived.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The problem occurred because of one of the following situations:

- An attempt to remove an dimension from a cube model was made where the dimension being removed is still being used by a related cube's cube dimension.
- An attempt to add a cube dimension to a cube was made, where the dimension for the cube dimension being added is not already being used by a related cube model.

User Response: Perform one of the following actions:

- Remove cube dimensions from cubes before removing related dimensions from related cube models.
- Add dimensions to cube models before adding related cube dimensions to related cubes.

See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6512 **The operation cannot be performed because the dimension *dimension_name* is referenced by a cube dimension.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified dimension is currently referenced by a cube dimension, so the identified dimension cannot be dropped.

User Response: Before you drop the identified dimension, change the referencing objects so that they no longer reference the identified dimension. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6513 **The operation cannot be performed for the *object_name* object. A cube dimension's cube hierarchy must be derived from the hierarchy referenced by the same dimension that was used to derive the cube dimension.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The problem occurred because of one of the following situations:

- An attempt to remove an hierarchy from a dimension was made, where the hierarchy being removed is still being used by a related cube dimension's cube hierarchy.
- An attempt to add a cube hierarchy to a cube dimension was made, where the hierarchy for the cube hierarchy being added is not already being used by a related dimension.

User Response: Perform one of the following actions:

- Remove cube hierarchies from cube dimensions before removing related hierarchies from related dimensions.
- Add hierarchies to dimensions before adding related cube hierarchies to related cube dimensions.

See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6514 **The operation cannot be performed because the** *cube_hierarchy_name* **cube hierarchy is referenced by a cube dimension.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified cube hierarchy is currently referenced by a cube dimension, so the identified cube hierarchy cannot be dropped.

User Response: Before you drop the identified cube hierarchy, change the referencing objects so that they no longer reference the identified cube hierarchy. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6515 **The operation cannot be performed for the** *object_name* **object. A cube dimension's cube hierarchy must be derived from the hierarchy referenced by the same dimension that was used to derive the cube dimension.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The problem occurred because of one of the following situations:

- An attempt to remove an hierarchy from a dimension was made, where the hierarchy being removed is still being used by a related cube dimension's cube hierarchy.
- An attempt to add a cube hierarchy to a cube dimension was made, where the hierarchy for the cube hierarchy being added is not already being used by a related dimension.

User Response: Perform one of the following actions:

- Remove cube hierarchies from cube dimensions before removing related hierarchies from related dimensions.
- Add hierarchies to dimensions before adding related cube hierarchies to related cube dimensions.

See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6516 **The operation cannot be performed because the** *join_name* **join is referenced by a dimension.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified join is currently referenced by a dimension, so the identified join cannot be dropped.

User Response: Before You drop the identified join, change the referencing objects so that they no longer reference the identified join. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6517 **The operation cannot be performed because the** *attribute_name* **attribute is referenced by a dimension.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified attribute is currently referenced by a dimension, so the identified attribute cannot be dropped.

User Response: Before you drop the identified attribute, change the referencing objects so that they no longer reference the identified attribute. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6518 **The operation cannot be performed because the *attribute_name* attribute is referenced by a hierarchy.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified attribute is currently referenced by a hierarchy, so the identified attribute cannot be dropped.

User Response: Before you drop the identified attribute, change the referencing objects so that they no longer reference the identified attribute. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6519 **The operation cannot be performed because the *relationship_name* attribute relationship is referenced by a hierarchy.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified attribute relationship is currently referenced by a hierarchy, so the identified attribute relationship cannot be dropped.

User Response: Before you drop the identified attribute relationship, change the referencing objects so that they no longer reference the identified attribute relationship. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6520 **The operation cannot be performed because the *dimension_name* dimension is referenced by a cube model.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified dimension is currently referenced by a cube model, so the identified dimension relationship cannot be dropped.

User Response: Before you drop the identified dimension, change the referencing objects so that they no longer reference the identified dimension. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6521 **The operation cannot be performed because the *join_name* join is referenced by a cube model.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified join is currently referenced by a cube model, so the identified join cannot be dropped.

User Response: Before you drop the identified join, change the referencing objects so that they no longer reference the identified join. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6522 **The operation cannot be performed because *object_name* is referenced by a facts object.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified object is currently referenced by a facts, so the identified object cannot be dropped.

User Response: Before you drop the identified object, change the referencing objects so that they no longer reference the identified object. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6523 The operation cannot be performed because the *attribute_name* left attribute is referenced by an attribute relationship.

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified attribute is currently referenced by an attribute relationship, so the identified attribute cannot be dropped.

User Response: Before you drop the identified attribute, change the referencing objects so that they no longer reference the identified attribute. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6524 The operation cannot be performed because the *attribute_name* right attribute is referenced by an attribute relationship.

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified attribute is currently referenced by an attribute relationship, so the identified attribute cannot be dropped.

User Response: Before you drop the identified attribute, change the referencing objects so that they no longer reference the identified attribute. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6525 The operation cannot be performed because the *attribute_name* right attribute is referenced by a join.

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified attribute is currently referenced by a join, so the identified attribute cannot be dropped.

User Response: Before you drop the identified attribute, change the referencing objects so that they no longer reference the identified attribute. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6526 The operation cannot be performed because the*attribute_name* left attribute is referenced by a join.

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified attribute is currently referenced by a join, so the identified attribute cannot be dropped.

User Response: Before you drop the identified attribute, change the referencing objects so that they no longer reference the identified attribute. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6527 The operation cannot be performed because the *model_name* cube model is referenced by a cube.

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified cube model is currently referenced by a cube, so the identified cube model cannot be dropped.

User Response: Before you drop the identified cube model, change the referencing objects so that they no longer reference the identified cube model. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6528 **The operation cannot be performed because the *cube_facts_name* cube facts object is referenced by a cube.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified cube facts is currently referenced by a cube, so the identified cube facts cannot be dropped.

User Response: Before you drop the identified cube facts, change the referencing objects so that they no longer reference the identified cube facts. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

6529 **The operation cannot be performed because the *facts_name* facts object is referenced by a cube model.**

Explanation: The requested operation cannot be performed because it will violate a referential constraint that exists between metadata objects in the metadata catalog. The identified facts is currently referenced by a cube model, so the identified facts cannot be dropped.

User Response: Before you drop the identified facts, change the referencing objects so that they no longer reference the identified facts. See the *Setup and User's Guide* for more information about metadata rules and referential constraints between metadata objects.

7001 **There are no cubes defined for the *model_name* cube model.**

Explanation: There are no cubes defined for the cube model.

User Response: If you want to optimize for extract queries that read data from the cube model into a MOLAP cube, you must define cubes that represent your MOLAP cubes. You cannot optimize for extract queries without defining one or more cubes.

7002 **The *model_name* cube model does not exist.**

Explanation: A cube model with the name that you specified is not defined.

User Response: Verify that the correct cube model and schema names are specified. Names and schemas are case sensitive. Use the OLAP Center to view the list of existing cube models.

7003 **Table space *table_space_name* was not found.**

Explanation: A table space with this name is not defined.

User Response: Verify that the correct tablespace name is specified.

7004 **The Optimization Advisor is unable to determine recommendations that fit in the disk space limit.**

Explanation: You specified a certain limit for how much disk space can be used for optimizing this cube model. The advisor could not produce recommendations that use less or the same specified amount of disk space.

User Response: Specify a larger disk space limit and run the Optimization Advisor wizard again.

7005 **Table space *table_space_name* cannot be used to store the summary tables.**

Explanation: The table space does not have the correct data storage type that is required to store table data. The table space must be a REGULAR type tablespace. LONG, USER TEMPORARY, and SYSTEM TEMPORARY table spaces cannot be used to store summary tables.

User Response: Specify a REGULAR table space to store the summary tables.

7006 Table space *table_space_name* cannot be used to store the indexes.

Explanation: The tablespace specified does not have the correct data storage type required to store index data. The tablespace must be a REGULAR or LONG type tablespace. USER TEMPORARY and SYSTEM TEMPORARY table spaces cannot be used to store the indexes.

User Response: Specify a REGULAR or LONG tablespace to store the indexes.

7007 Optimization validation of the *model_name* cube model failed.

Explanation: The cube model and associated metadata objects violate one or more of the metadata object rules required for optimization. Optimization will not be performed.

User Response: Optimization cannot be performed unless the cube model and associated metadata objects conform to the metadata object rules for optimization. See the *Setup and User's Guide* for more information about optimization rules.

7008 The cube model does not have any dimensions that have optimizable hierarchies.

Explanation: Optimization will not be performed because the Optimization Advisor cannot find dimensions with hierarchies that can be optimized.

User Response: Ensure that the cube model has at least one dimension that has a nonrecursive hierarchy.

7200 The recommended summary tables will use deferred refresh because the cube model contains one or more nondistributive measures.

Explanation: The refresh immediate option was selected for the summary tables. However, summary tables cannot be refreshed immediately if there are nondistributive measures defined in the cube model. Distributive measures use simple aggregation functions such as SUM and COUNT that can be aggregated from any intermediate values. Nondistributive measures use more complex aggregation functions, such as STDDEV, and they must be aggregated from the base tables.

User Response: If it is not necessary to maintain the summary tables synchronously with the base tables, no action is required. If the summary tables must be maintained synchronously with the base tables, you need to change the metadata so that only distributive measures are defined.

7201 The *table_name* recommended summary table will use deferred refresh because one or more nullable attributes were found as columns in the fullselect of this recommended summary table.

Explanation: The refresh immediate option was selected for the summary tables. However, the recommended summary table contains one or more attributes that are used as nullable columns in the summary table's fullselect. Using nullable columns in a summary table's fullselect can cause slow immediate refresh performance. The summary table was set to *refresh deferred* to avoid this performance problem.

User Response: To change the nullability of an attribute, you must change the attribute's SQL expression or change the nullability of the DB2 table columns used by the attribute or both. These changes are not usually recommended because they might be difficult to implement.

7202 The *table_name* table does not have statistics.

Explanation: The Optimization Advisor cannot find valid table statistics values for the specified table.

User Response: Use the RUNSTATS command to create statistics for the specified table. Then run the Optimization Advisor wizard again.

7400 The summary tables are defined using the ROLLUP operator because the cube model contains one or more nondistributive measures.

Explanation: Measures are either distributive or nondistributive. Distributive measures use simple aggregation functions such as SUM and COUNT that can be aggregated from any intermediate values. Nondistributive measures use more complex aggregation functions, such as STDDEV, and they must be aggregated from the base tables. To avoid the cost of aggregating nondistributive measures from the base tables, the summary tables are defined using the ROLLUP operator, which pre-aggregates the nondistributive measures.

User Response: No action is required.

7401 The *table_name* summary table is recommended. It is estimated to have *rows* rows, a *n* MB table size, and a *n* MB index size.

Explanation: This message is a description of the recommended summary table, including the estimated row count, estimated disk space, and estimated disk space that is used for indexes.

User Response: No action is required.

7402 There are *n* summary tables that do not fit in the specified disk space limit. They have a cumulative estimated size of *n* MB

Explanation: This message provides information about the recommended summary tables that do not fit in the disk space limit.

User Response: To view these summary tables in the recommendations, run the Optimization Advisor again with a larger specified disk space limit.

7403 The recommendations include optimizations for the *cube_name* cube.

Explanation: Summary tables are recommended for the specified cube. Some queries for this cube will be optimized.

User Response: No action is required.

7404 The recommendations do not include optimizations for the *cube_name* cube.

Explanation: Summary tables are not recommended specifically for this cube. Queries specific to this cube are not likely to show performance improvement.

User Response: If summary tables are not included in the recommendations because of a disk space limitation, run the Optimization Advisor again with a larger specified disk space limit. The recommendations might include one or more summary tables to optimize queries for this cube.

7405 The specified time limit ran out while the Optimization Advisor was determining the recommendations.

Explanation: The Optimization Advisor made a recommendation. If more time is allowed, the Optimization Advisor might be able to make a better recommendation because it can perform additional analysis. Running the Optimization Advisor longer does not guarantee better results.

User Response: You can run the Optimization Advisor again with more time specified, or you can create the recommended summary tables and see if the performance is acceptable.

7406 The *dimension_name* dimension does not have any hierarchies that can be optimized by the Optimization Advisor.

Explanation: The Optimization Advisor cannot optimize for recursive hierarchies. The specified dimension does not contain any hierarchies that can be optimized so the Optimization Advisor ignores this dimension. Queries that refer to attributes from this dimension are not optimized.

User Response: No action is required. Queries that use attributes from this dimension will not have any performance improvement.

7407 **The recommended summary tables optimize for *n* percent of the slices in the cube model. Queries that run against the optimized slices should have improved performance.**

Explanation: SQL queries access particular slices in the cube model. One way to analyze performance improvement is to consider what portion of the slices that can be queried will be improved. If the cube model uses distributive measures, queries that access slices that are logically above the summary table slice will have improved performance.

For example, there are 30 possible slices in a cube model that has a Time dimension with the hierarchy [All-Year-Quarter-Month-Day] and a Region dimension with the hierarchy [All-Country-Region-State-City-Store]. You can calculate the number of possible slices by multiplying the number levels in the dimension hierarchies together. If the recommended summary table optimizes for the Month-City slice, then all slices at or above that slice are optimized. In this example, 20 of the 30 possible slices, or 67% (20/30) of the slices are optimized. There will never be 100% coverage because that will require duplicating the base tables in the summary tables. Typically, the lowest slices are less beneficial to optimize for because they are not very different from the base tables.

User Response: No action is required. If the percentage is low, you can run the Optimization Advisor wizard again with a larger specified disk space limit.

7408 **Reading cube model metadata from the database.**

Explanation: The Optimization Advisor is reading the metadata that describes the cube model. The metadata contains information that significantly affects the optimization recommendations.

User Response: No action is required.

7409 **Selecting which aggregations to include in the summary tables.**

Explanation: The Optimization Advisor is testing potential summary table configurations to determine which configuration is optimal for the specified criteria.

User Response: No action is required.

7410 **Sampling data from cube model.**

Explanation: The Optimization Advisor is reading a subset of data from the fact and dimension tables so that it can estimate the size of the summary table. Sampling might occur multiple times as the Optimization Advisor considers potential summary tables.

User Response: No action is required.

7411 **Defining indexes for recommended summary tables.**

Explanation: The Optimization Advisor determined the recommended summary tables and is selecting indexes to build for the summary tables.

User Response: No action is required.

OLAP Center messages

10000 OLAP Center is unable to retrieve any database names.

Explanation: An error occurred retrieving the list of database names from DB2.

User Response: Check that OLAP Center is installed correctly. If the problem persists, contact IBM Software Support.

10001 Type a user name.

Explanation: The **User name** field is empty.

User Response: Type a user name in the **User name** field.

10002 Type a password.

Explanation: The **Password** field is empty.

User Response: Type a password in the **Password** field.

10003 The attribute *attribute_name* is selected but the relationship type is not specified. Click the current relationship type value and select a new relationship type.

Explanation: With the related attributes window, you can specify which attributes will be included in the levels of the hierarchy. Each attribute that is selected must also have a relationship to describe how the attribute relates to the main hierarchy attribute. This relationship type will be used to build an attribute relationship in the database between the hierarchy attribute and selected attribute.

User Response: Select one of the valid relationship types for the attribute by clicking in the **Relationship type** field next to the selected attribute.

10004 Cannot parse the attribute entered in the SQL expression.

Explanation: The SQL expression that was entered refers to an attribute that is neither valid in the given context nor present in the database.

User Response: Ensure that the SQL expression refers to only those attributes that appear in the **Data** section of the SQL Expression Builder.

10005 The *object_name* object was successfully exported to the *file_name* file.

Explanation: Export was successful.

User Response: No action is required.

10006 Enter the file name to export the metadata objects to.

Explanation: The export utility writes the exported metadata objects in to the file name entered by the user.

User Response: Type a file name in the **File name** field.

10007 Select a cube or a cube model to export.

Explanation: The export window can export a cube or a cube model.

User Response: Select an object to export.

10008 Enter a unique name for the object that you are creating.

Explanation: In the SQL Expression Builder, the Name field of the attribute or measure being created is empty.

User Response: Type an object name in the Name field. The object name must be unique in the name space of the attributes and measures.

10009 Enter an SQL expression for the object.

Explanation: The SQL expression field of the object is empty.

User Response: Enter an SQL expression for the object.

10010 The *column_name* column is not qualified with a table name.

Explanation: The column name entered in the SQL expression is not qualified with a table name.

User Response: Ensure that the column name in the SQL expression is qualified with a table name using '.'as a separator.

10011 The *column_name* column is not qualified with a schema name.

Explanation: Column references in the SQL expression must be qualified with both a table name and a schema name separated by '.'.

User Response: Ensure that the column name in the SQL expression is qualified with both a table name and a schema name separated by '.'.

10012 The first element in an aggregation script cannot be a dimension.

Explanation: An aggregation script was created with a dimension as the first element.

User Response: Use an aggregation function as the first element in the aggregation script.

10013 Select an existing measure or enter an SQL expression as the second parameter for the *function_name* multiparameter function in the aggregation.

Explanation: When you use a multiparameter function in the aggregation script, the first parameter is defined as the measure that the aggregation is associated with. For the second parameter, select an existing measure or enter an SQL expression.

User Response: Enter a measure or an SQL expression as a second parameter.

10014 The *function_name* aggregation function has no matching dimensions.

Explanation: Each aggregation function in the aggregation script must be applied to at least one dimension.

User Response: Ensure that each aggregation function in the aggregation script is applied to at least one dimension.

10015 Closing parenthesis is missing for the *object_name* object.

Explanation: In OLAP Center, attributes, measures or columns specified in an expression should be enclosed by @Attribute(), @Measure() or @Column() respectively.

User Response: Enter a closing parenthesis for the object.

10016 Referring to the *object_name* object in the SQL expression creates an invalid reference loop.

Explanation: The object refers to itself in its SQL expression.

User Response: Ensure that the objects in the SQL expression do not create reference loops.

10017 No errors were found. The SQL expression is valid.

Explanation: The SQL expression is valid.

User Response: No action is required.

10018 No errors were found. The aggregation script is valid.

Explanation: The set of aggregations in the aggregation script is valid.

User Response: No action is required.

10020 Type a name.

Explanation: The **Name** field of the object is empty.

User Response: Type an object name in the **Name** field.

10021 **Type a schema name.**

Explanation: The **Schema** field is empty.

User Response: Type a schema name in the **Schema** field.

10022 **Type a business name.**

Explanation: The **Business name** field is empty.

User Response: The business name can be displayed in business intelligence applications to identify the object to an end user. Type a business name in the **Business name** field.

10023 **Select at least one attribute for the hierarchy.**

Explanation: No attributes are specified for the hierarchy.

User Response: Select at least one attribute for the hierarchy.

10024 **Select at least one measure for the cube facts.**

Explanation: No measures are specified for the cube facts.

User Response: Select at least one measure for the cube facts.

10025 **Select at least one attribute to include in the cube hierarchy.**

Explanation: No attributes are specified in the cube hierarchy.

User Response: Select at least one attribute to include in the cube hierarchy.

10026 **Select at least one dimension in the cube.**

Explanation: No dimensions were specified in the cube.

User Response: Select at least one dimension and then click the [...] button to specify details for the cube dimension.

10027 **One or more dimensions which currently exist in the cube have been cleared. Click Yes to delete the cube dimensions. Click No to return to the window, then click Cancel to close the window without saving the changes.**

Explanation: One or more dimension selections have been cleared. The corresponding cube dimensions will be deleted from the cube.

User Response: Click **Yes** in the window to remove the cube dimensions from the cube. Click **No** to keep the cube dimensions and then click **Cancel** to close the window without saving.

10028 **An attribute relationship cannot be defined with a many:many cardinality if the functional dependency check box is selected.**

Explanation: An attribute relationship cannot be defined with both cardinality *many:many* and functional dependency selected.

User Response: Select a different cardinality for the attribute relationship or clear the functional dependency checkbox.

10029 **Select a left attribute and a right attribute for the attribute relationship.**

Explanation: An attribute relationship cannot be defined if both left and right attributes are not selected.

User Response: Select both left and right attributes.

10030 **The same attribute cannot be selected for both the left and right attribute in an attribute relationship.**

Explanation: An attribute relationship cannot be defined if the left and right attributes are identical.

User Response: Select different left and right attributes.

10031 **An object with the name and schema specified already exists in the database. Type a different name.**

Explanation: An object of the type being created or modified already exists in the database with the same name and schema specified.

User Response: Enter a unique name for the object.

10032 **Specify at least one attribute pair.**

Explanation: A join must have at least one attribute pair.

User Response: Specify at least one attribute pair.

10033 **Duplicate attribute pairs cannot be created.**

Explanation: An attribute pair already exists that matches the new selections.

User Response: Select different left and right attributes.

10034 **Select at least one table.**

Explanation: No tables have been selected.

User Response: Select at least one table in order to proceed.

10035 **Select or create new joins to join all of your selected tables.**

Explanation: No joins were selected.

User Response: Select or create new joins that will join all of your selected tables.

10036 **Select at least one attribute.**

Explanation: No attributes were selected.

User Response: Select at least one attribute.

10037 **Select a join to join the dimension with the facts object.**

Explanation: No joins were selected.

User Response: Select one join which will join your dimension with the facts object.

10038 **Specify only one join between two given tables. The** *join_name1* **join and the** *join_name2* **join both join the same two tables.**

Explanation: More than one join was selected for the same pair of tables.

User Response: Select only one join for each pair of tables.

10039 **All selected tables must be joined. Select a join for the** *table_name* **table.**

Explanation: All selected tables must be joined.

User Response: Select a join for the specified table.

10040 **The number of selected tables does not correspond to the number of selected joins. Verify that there are no join loops and that all of the tables are joined.**

Explanation: All selected tables must be joined.

User Response: Ensure that there are no join loops and that all of the tables are joined.

10042 **Select at least one measure.**

Explanation: No measures were specified.

User Response: Select at least one measure.

10043 **Select a table column.**

Explanation: A table column was not specified.

User Response: Select a column.

10044 **Select an SQL expression.**

Explanation: An SQL expression was not specified.

User Response: Click the **Build Expression** button to build your expression.

10045 **An aggregation script was not specified.**

Explanation: An aggregation script was not specified.

User Response: Click the **Build Script** button to build your aggregation script.

10046 **Select a measure before opening the expression builder.**

Explanation: A measure was not selected.

User Response: Select a measure from the table.

10047 **Select a measure before opening the Aggregation Script Builder.**

Explanation: A measure was not selected.

User Response: Select a measure.

10048 **The Aggregation Script Builder cannot be launched for the** *measure_name* **measure because the** *model_name* **cube model does not have at least one dimension.**

Explanation: An aggregation script cannot be specified if the cube model does not have at least one dimension.

User Response: Add dimensions to the cube model before specifying an aggregation script.

10049 **To edit the expression, specify an attribute.**

Explanation: An attribute is not selected.

User Response: Select an attribute.

10050 **The metadata will be refreshed from the database. Any changes that were being made when the error occurred will be lost.**

Explanation: An error occurred calling the DB2 stored procedure.

User Response: Click **OK** to refresh the metadata displayed byOLAP Center. Any changes that were being made when the error occurred will be lost. The objects displayed in OLAP Center will be refreshed with the corresponding objects in the database allowing the user to continue working.

10051 **The** *model_name* **cube model cannot be validated for optimization. DB2 returned the following message:***message***.**

Explanation: OLAP Center cannot start the Optimization Advisor wizard for the selected cube model because the selected cube model did not pass the validation that was performed by the stored procedure API.

User Response: Check the stored procedure API documentation for the cube model validation rule. Follow the instructions from the return message from DB2.

10052 **Some of the loaded attributes or measures map to** *column_names* **columns that no longer exist in the database. Resolve the problem either by restoring the tables to which the columns belong or by dropping the invalid attributes or measures or both.**

Explanation: This message appears when you start OLAP Center or after you click **View —> Refresh**. It appears because a table that the loaded attributes or measures map to was dropped or renamed.

User Response: Correct the problem in one of the following ways:

- Restore the table that was deleted or renamed.
- Map the attributes/measures to a table that does exist in the database.
- Drop the attribute/measures that map to the columns that do not exist.

10053 **The** *model_name* **cube model optimization validation returned a warning. DB2 returned the following message:***message*

Explanation: OLAP Center attempted to validate the cube model before starting the Optimization Advisor and DB2 returned a warning. The warning might indicate that you have a cube model that cannot be optimized. For example, your cube model might contain views that reference tables that do not have constraints defined between them.

User Response: Check the message returned by DB2 and decide if you want to continue running the Optimization Advisor wizard.

10060 **The cube model is not complete. Before a cube can be created, the cube model must contain a facts object, at least one dimension, and at least one hierarchy for each dimension.**

Explanation: The cube model is not in a valid state for a cube to be created.

User Response: Modify the cube model so that it has a facts object and at least one dimension. Ensure that each dimension has at least one hierarchy.

10061 **When a cube model is dropped, its dimensions are removed and its facts are dropped. The removed dimensions will continue to be available from the All Dimensions folder. Are you sure you want to drop the** *model_name* **cube model?**

Explanation: Drop confirmation message.

User Response: Ensure the selected object is the one you want to drop and click **Yes**. If you do not want to drop the selected object, click **No**.

10062 **When a dimension is dropped, its hierarchies and corresponding cube dimensions are also dropped. Are you sure you want to drop the** *dimension_name* **dimension?**

Explanation: Drop confirmation message.

User Response: Ensure that the selected object is the one you want to drop and click **Yes**. If you do not want to drop the selected object, click **No**.

10063 **When a cube is dropped, its cube dimensions, cube hierarchies, and cube facts are also dropped. Are you sure you want to drop the** *cube_name* **cube?**

Explanation: Drop confirmation message.

User Response: Ensure the selected object is the one you want to drop and click **Yes**. If you do not want to drop the selected object, click **No**.

10064 When a cube dimension is dropped, its cube hierarchies are also dropped. Are you sure you want to drop the *cube_dimension_name* cube dimension?

Explanation: Drop confirmation message.

User Response: Ensure the selected object is the one you want to drop and click **Yes**. If you do not want to drop the selected object, click **No**.

10065 Are you sure you want to drop *object_name*?

Explanation: Drop confirmation message.

User Response: Ensure the selected object is the one you want to drop and click **Yes**. If you do not want to drop the selected object, click **No**.

10066 When a dimension is removed, all of the corresponding cube dimensions are removed from their cubes. Are you sure you want to remove the *dimension_name* dimension from *object_name*?

Explanation: Remove dimension confirmation message.

User Response: Ensure that the selected object is the one that you want to remove and click **Yes**. If you do not want to remove the selected object, click **No**.

10067 The file with the name *file_name* already exists. Do you want to overwrite its contents?

Explanation: Overwrite file confirmation message.

User Response: Ensure that you want to overwrite the contents of the file name that you entered.

10068 Unable to determine the data type for the object with the *object_name* name and *schema_name* schema. The database returned the following information: *message*.

Explanation: For the specified object, OLAP Center cannot determine the source data type or aggregated data type.

User Response: Ensure that the SQL expression for the specified object is correct. If you cannot resolve the problem, contact IBM Software Support.

10069 Unable to determine the source data type for the measure with the *measure_name* name and *schema_name* schema.

Explanation: For the specified measure, OLAP Center cannot determine the source data type because the specified measure has an invalid source expression. A measure can have an invalid source expression when the None aggregation setting is applied to it because the measure is validated with aggregations of referenced measures rather than as a self-contained expression.

User Response: You can perform one of the following actions:

- Alter the source expression of the specified measure so that it validates correctly with the None aggregation setting.
- Do not use the specified measure in your expression.

10070 When a facts object is dropped, its measures are also dropped. Are you sure you want to drop the *facts_name* facts?

Explanation: Drop confirmation message.

User Response: Ensure the selected object is the one you want to drop and click **Yes**. If you do not want to drop the selected object, click **No**.

10071 All the selected objects will be dropped from the database. Do you want to drop these objects?

Explanation: More than one object was selected and the drop option was selected.

User Response: Ensure that the selected objects are the ones that you want to drop and click **Yes**. If you do not want to drop the selected objects, click **No**.

10072 Some of the selected objects cannot be dropped. These objects remain in the database.

Explanation: OLAP Center cannot drop all of the selected objects. This is probably because some of the selected objects are referenced by other objects in the database and dropping the selected object makes the referencing object invalid.

User Response: No action is required.

10080 Object of type *type* not found during the second pass of XML.

Explanation: An object referenced in the XML being read could not be located.

User Response: Ensure that the XML file being imported is correctly formed. If this error occurs while starting OLAP Center, contact IBM Software Support.

10084 An object with the name *object_name* in schema *schema_name* already exists. The object cannot be created. Type a unique name, schema, or both for the new object.

Explanation: OLAP Center attempted to create a new object, but an object of this type with the same name and schema already exists.

User Response: Enter a different name, schema, or both for the object being created.

10085 An object of name *object_name* in schema *schema_name* already exists. The object cannot be renamed. Type a unique name, schema, or both for the object being renamed.

Explanation: OLAP Center attempted to rename an object, but an object of this type with the same name and schema already exists.

User Response: Enter a different name, schema, or both for the object being renamed.

10086 A database connection could not be made. DB2 returned: *message*.

Explanation: OLAP Center could not connect to the database. Some error information provided by DB2 is included in the message.

User Response: Read the text returned by DB2 and correct the problem.

10087 The *object_name1* metadata object cannot be dropped because it is referred to by the *object_name2* object of type *type*.

Explanation: The selected metadata object cannot be dropped because it is in use by at least one other metadata object.

User Response: Remove the object from any other metadata objects it is a part of, then try dropping the object again.

10088 An error occurred registering the DB2 driver with the JDBC Driver Manager. A database connection could not be established. The following information was returned: *message*.

Explanation: Before connecting to a DB2 database, OLAP Center has to register the JDBC driver that it will use with the Driver Manager. An error occurred during the JDBC driver registration.

User Response: Check the DB2 installation to make sure that the `db2java.zip` and `db2jcc.jar` files are installed. Ensure that Java and any JDBC components are installed correctly. Read the information returned in the message to help resolve the problem.

10089 An error occurred while accessing the database. The database returned the following information: \n SQL State: *message***\n SQL Error Code:** *code***\n SQL Message:** *SQL_message*

Explanation: OLAP Center application called DB2 using the API stored procedure. The execute command threw an SQLException that could not be handled by OLAP Center.

User Response: Use the additional error information provided in the message to resolve the problem. If you cannot resolve the problem, contact IBM Software Support.

10090 Executing the DB2 stored procedure caused a false return code. No error information was found in the returned XML document. Contact IBM Software Support.

Explanation: The OLAP Center application called DB2 using the API stored procedure. The execute command returned *false*, but there was no error information in the XML document returned by the stored procedure.

User Response: It is possible that the operation completed successfully, but you should report this problem to IBM Software Support.

10091 An error occurred while processing a database API call. The following information was returned: \n SQL State:*message***\n SQL Error Code:** *code***\n Operation:***operation***\n Status ID:** *ID***\n Status Text:** *text*

Explanation: The OLAP Center stored procedure API call had an error while executing some OLAP Center changes.

User Response: See the information contained in the message. If the problem cannot be resolved, contact IBM Software Support.

10092 An error occurred while parsing the XML returned by the database API call. The following information was returned: *message***.**

Explanation: The OLAP Center stored procedure API call returned XML that was incomplete or badly formed. OLAP Center could not read the returned XML.

User Response: Use the information that is contained in the message to resolve the problem. If the problem cannot be resolved, contact IBM Software Support.

10093 The *file_name* **file does not exist.**

Explanation: The specified file does not exist.

User Response: Specify a file that exists.

10094 An I/O error occurred reading the *file_name* **file. The following system information was returned:** *message***.**

Explanation: An I/O error occurred while reading from a file.

User Response: Check the system information to try to resolve the problem or specify a different file.

10095 An I/O error occurred writing to the *file_name* **file. The following system information was returned:** *message***.**

Explanation: An I/O error occurred while writing to a file.

User Response: Check the system information to try to resolve the problem or specify a different file.

10096 A query to retrieve the database schema failed. The database returned the following information: ***message*****.**

Explanation: A query to retrieve the database schema failed.

User Response: Check the database information to resolve the problem.

10097 A query to retrieve a schema's tables failed. The database returned the following information: ***message*****.**

Explanation: A query to retrieve a schema's tables failed.

User Response: Check the database information to resolve the problem.

10098 A query to retrieve a table's columns failed. The database returned the following information: ***message*****.**

Explanation: A query to retrieve a table's columns failed.

User Response: Check the database information to resolve the problem.

10099 A commit of a DB2 connection failed. The database returned the following information: ***message*****.**

Explanation: A commit of a DB2 connection failed.

User Response: Check the database information to resolve the problem.

10100 A rollback of a DB2 connection failed. The database returned the following information: ***message*****.**

Explanation: A rollback of a DB2 connection failed.

User Response: Check the database information to resolve the problem.

10101 *Object_name* **cannot be dropped because it is the last cube dimension in the** *cube_name* **cube. A cube must have at least one cube dimension to be valid.**

Explanation: OLAP Center attempted to drop the last cube dimension in a cube.

User Response: A cube must have at least one cube dimension to be valid. Do not attempt to drop the last cube dimension from a cube.

10102 Object *object_name1* **of type** *type1* **refers to object** *object_name2* **or type** *type2* **which could not be found.**

Explanation: An object within the XML file being read refers to an object which cannot be found. If the error occurs during import, the object being referred to might not exist within the file being imported.

User Response: If an import is being performed, ensure the file contains all of the objects needed for the import to succeed. If the error occurs while starting OLAP Center, contact IBM Software Support.

10200 The file being imported does not have a UTF-8 encoding. Select a file with UTF-8 encoding.

Explanation: OLAP Center can import files only in the UTF-8 encoding.

User Response: Import a file with the supported encoding.

10201 Enter a file name for the SQL script used to refresh summary tables.

Explanation: The Optimization Advisor wizard creates an SQL script to refresh summary tables when the Deferred update option is selected. This script should be saved in a file and run to refresh the summary tables.

User Response: Enter a file name to save the SQL script to.

10202 Enter a file name for the SQL script used to create summary tables.

Explanation: The Optimization Advisor wizard generates an SQL script to create summary tables. This script should be saved in a file and run to create the summary tables.

User Response: Enter a file name to save the SQL script to.

10203 The selected measure cannot have None as its aggregation setting. Only calculated measures which refer exclusively to other measures in their expressions can specify the None aggregation setting.

Explanation: The None aggregation setting can only be selected for measures that only use expressions which refer exclusively to other measures.

User Response: Select a different aggregation.

10204 No dimensions exist. Create a new dimension to add to the cube model.

Explanation: No dimensions exist. Create a new dimension to add to the cube model.

User Response: Create a new dimension, instead of adding a dimension.

10205 There are no dimensions to add because all of the existing dimensions are already included in the cube model.

Explanation: All existing dimensions have been added to the cube model.

User Response: No action is required.

10206 You have changed your selected options. To see new recommendations from the Optimization Advisor wizard, you must run the Optimization Advisor wizard process again. If you do not run the Optimization Advisor wizard process again, you will see the recommendations created for the earlier options. Do you want to run the Optimization Advisor wizard process again?

Explanation: You have changed the selected options after running the Optimization Advisor wizard process. To view updated recommendations for the summary tables, run the Optimization Advisor wizard process again. If you do not run the Optimization Advisor wizard process again, you will see the recommendations created for the earlier options.

User Response: Click **Yes** to run the Optimization Advisor wizard process. Click **No** if you do not want to run the Optimization Advisor wizard process again.

10207 No dimension table has been detected.

Explanation: No dimension table was detected.

User Response: Ensure that referential integrity constraints are correctly set.

10208 *Object_names* **objects that OLAP Center cannot directly display exist in the database. These objects might cause future problems with OLAP Center. Click Yes to drop the objects or click No to keep the objects in the database.**

Explanation: OLAP Center detected a number of objects (such as hierarchies or facts) in the database that it cannot display directly. These objects might be preexisting or might be created after importing metadata. These objects might cause name clash and reference problems in OLAP Center in the future. Unless you have a good reason to keep these objects, it is recommended that you choose to drop them.

User Response: Click **Yes** to drop the objects or click **No** to keep the objects in the database.

10209 Unexpected error occurred during the import operation. Check the input XML file for errors.

Explanation: During import, the stored procedure API returned a warning with nothing in the output XML.

User Response: Ensure that the input XML metadata complies with the format defined in the OLAP metadata schema and the XML file defines all the metadata objects referred in it.

10210 Import operation failed. The stored procedure API returned the following message: ***message*****.**

Explanation: During import process, the stored procedure API returned an error message.

User Response: Resolve the problem using the information provided in the message. If the problem cannot be resolved, contact IBM Software Support.

10211 The nonnumeric measure *measure_name* **cannot use the** *function_name* **aggregation function because that function expects a numeric argument.**

Explanation: Measures with nonnumeric data types cannot have numeric aggregation functions. You can only only select **MIN**, **MAX**, or **COUNT** as aggregation functions for nonnumeric data.

User Response: Choose a different aggregation function.

10212 Unable to read the objects from the input XML file. Check the input XML file for errors.

Explanation: OLAP Center failed to read the objects from the input XML file.

User Response: Ensure that the input XML metadata complies with the format defined in the OLAP metadata schema and the XML file defines all of the metadata objects referred in it.

10213 The *file_name* **input XML file does not exist in the specified directory.**

Explanation: The input XML file does not exist in the specified directory.

User Response: Ensure that the input XML file exists in the specified directory.

10214 The *object_name* **object contained in the import file refers to the** *column_name* **column that does not exist in the database. Ensure that the tables and columns referred to by the metadata objects in the import file exist before importing the file.**

Explanation: The import XML file contains objects that refer to tables and columns that do not exist in the database.

User Response: Ensure that the tables referred to by the objects in the import XML file exist in the database before importing the file.

10215 OLAP Center cannot run the SQL script recommended by the Optimization Advisor wizard. The database returned the following information: ***message*****.**

Explanation: OLAP Center cannot execute the SQL script recommended by the Optimization Advisor wizard. You might not have sufficient privileges to execute the SQL script.

User Response: Ensure that you have the correct authorities to run the Optimization Advisor recommendations. The required authorities are described in the topic on ″Authorities and privileges″ in the OLAP Center online help. See the DB2 Multidimensional Metadata Management (DB2 Cube Views) *Setup and User's Guide* for information on optimizing a cube model.

10216 The recommendations from the Optimization Advisor were successfully saved in the specified file(s).

Explanation: The recommended create summary tables SQL script and if applicable, the refresh summary tables SQL script, were saved into the specified files.

User Response: No action is required.

10217 The summary tables and their indexes were created successfully.

Explanation: The summary tables and indexes recommended by the Optimization Advisor were successfully created in the database.

User Response: No action is required.

10218 You have selected a view. The Optimization Advisor cannot verify that referential constraints exist for tables referenced by your view.

Explanation: Optimization might not be effective when you create summary tables for cube models using views that reference tables without constraints. The Optimization Advisor cannot detect if constraints exist on the tables referenced by the view.

User Response: If the tables referenced by your view do not have constraints and you want to run the Optimization Advisor you can either: 1. Not use the view in your cube model. 2. Create constraints for the tables before running the Optimization Advisor.

10300 Failed to parse the *measure_name* **measure entered in the SQL expression.**

Explanation: The SQL expression specified refers to a measure that is either invalid in the given context or is not present in the database.

User Response: Ensure that the SQL expression refers to only those measures that appear in the Data list of the SQL Expression Builder.

10301 Failed to parse the *column_name* **column entered in the SQL expression.**

Explanation: The SQL expression specified refers to a column that is either invalid in the given context or is not present in the database.

User Response: Ensure that the SQL expression refers to only those columns that appear in the Data list of the SQL Expression Builder.

10302 The *attribute_name* **attribute is not qualified with a schema name.**

Explanation: References to attributes in the SQL expression must be qualified with a schema name separated by a'.'.

User Response: Ensure that all of the references to attributes in the SQL expression are qualified with a schema name separated by '.'.

10303 The *measure_name* **measure is not qualified with a schema name.**

Explanation: References to measures in the SQL expression must be qualified with a schema name separated by a '.'.

User Response: Ensure that all the references to measures in the SQL expression are qualified with a schema name separated by '.'.

10304 Missing object name inside the *object_name* **object tag.**

Explanation: The SQL expression specified has an empty column tag @Column or an empty attribute tag @Attribute or an empty measure tag @Measure.

User Response: Ensure that the object type tags @Column, @Measure and @Attribute have an enclosing object name.

10305 The expression specified is invalid. The database returned the following information: *message*.

Explanation: There is a syntax error in the SQL expression. This error is also displayed when the

SQL expression references columns, attributes, or measures without enclosing tags. A reference to a column, attribute or measure must be enclosed inside @Column(), @Attribute() or @Measure() tags respectively.

User Response: Correct the syntax error. Ensure that each column, attribute and measure is enclosed in the appropriate tag.

10306 The data type of the expression that you have entered is nonnumeric. Enter a numeric expression as a second parameter.

Explanation: The data type of the second parameter must be numeric.

User Response: Ensure that the data type of the expression entered results in to a numeric data type.

10307 The expression of the measure *measure_name* results to a nonnumeric data type. Select a measure whose expression results to a numeric data type.

Explanation: The data type of the second parameter must be numeric.

User Response: Ensure that the data type of the selected measure's expression is a numeric.

10308 OLAP Center cannot communicate with the specified database. This might be because the database is not correctly configured for Multidimensional Metadata Management. Configuring the database might take some time. Click Yes to configure the specified database for Multidimensional Metadata Management. Click No if you do not want to configure the specified database now.

Explanation: OLAP Center can connect to the database using the user name and password supplied, but it cannot communicate with the stored procedure API. This might be because the Multidimensional Metadata Management (DB2 Cube Views) stored procedure API is not registered for the specified database, or the DB2 catalog does not exist for the specified database.

User Response: Click **Yes** to configure the database for Multidimensional Metadata Management (DB2 Cube Views); otherwise, click **No**.

10309 The *database_name* database was successfully configured for Multidimensional Metadata Management.

Explanation: OLAP Center successfully created the Multidimensional Metadata Management (DB2 Cube Views) catalog and registered the stored procedure API for the specified database.

User Response: No action is required.

10310 OLAP Center cannot configure the database for DB2 Multidimensional Metadata Management. The database returned the following information: *message*.

Explanation: OLAP Center cannot configure the specified database for DB2 Multidimensional Metadata Management (DB2 Cube Views). This might be because OLAP Center cannot register the DB2 Multidimensional Metadata Management (DB2 Cube Views) stored procedure API, or OLAP Center cannot create one or more DB2 Multidimensional Metadata Management (DB2 Cube Views) catalog tables.

User Response: Ensure that you have the correct setup and install authorities that are described in the topic on ″Authorities and privileges″ in the OLAP Center online help. See the DB2 Multidimensional Metadata Management (DB2 Cube Views) *Setup and User's Guide* for information on configuring a database.

10311 The aggregation validation failed. One or more specified aggregation functions are not compatible with the source SQL Expression.

Explanation: One or more specified aggregation functions are not compatible with the source SQL Expression. This might be because the specified aggregation function expects a parameter with a data type that is different from the source SQL expression data type.

User Response: Ensure that the aggregation function is valid for the specified measure's source data type.

10312 The measure's source expression is syntactically correct only with the None aggregation setting. The measure must use the None aggregation setting.

Explanation: The measure expects the None aggregation setting when:

- The SQL Expression is syntactically incorrect when the aggregation functions are not applied to its referred measures but it is syntactically correct when those aggregation functions are applied. For example, char + int is syntactically incorrect but COUNT(char) + SUM(int) is syntactically correct.
- The SQL Expression uses OLAP functions like RANK(), DENSE_RANK() and ROW_NUMBER().

User Response: Ensure that the measure has None aggregation setting applied to it.

10401 The expression cannot include a column function, a scalar fullselect, or a subquery.

Explanation: The SQL expression cannot include a column function, scalar fullselect or a subquery.

User Response: Correct the use of the column function to eliminate the invalid expression.

10501 The schema name cannot start with *prefix*.

Explanation: The schema name cannot start with 'SYS' and 'SESSION'.

User Response: Type different schema name.

10502 The join properties are not valid for cube model performance optimization. Resolve this problem then run the Optimization Advisor wizard again. The database returned the following information: *message*.

Explanation: The join properties are invalid for cube model performance optimization.

User Response: Specify correct settings for your join by applying the optimization validation rules.

10503 The hierarchy cannot be modified because it has an associated cube hierarchy.

Explanation: If a cube hierarchy exists for the hierarchy, the hierarchy cannot be modified.

User Response: Ensure that no cube hierarchy references the hierarchy being modified before making changes to the hierarchy. You can also create a different hierarchy with the required modifications.

10504 This measure must use the None aggregation setting because it refers to the measure that uses a multiparameter aggregation function.

Explanation: Only measures that use the None aggregation setting can refer to measures that use a multiparameter function. You cannot change the aggregation setting from None to another function.

User Response: You can perform one of the following actions:

- Do not alter the measure's aggregation setting.

- Alter the specified measure so that it does not use a multiparameter function.

10505 This measure cannot use a multiparameter function because the *measure_name* measure that uses an aggregation setting other than None, refers to this measures.

Explanation: Only measures that use the None aggregation setting can refer to measures that use a multiparameter function. You cannot change the aggregation script of the measure being edited to include a multiparameter function because the measure being edited is referred to by a measure that does not use the None aggregation setting.

User Response: You can perform one of the following actions:

- Do not alter the measure's aggregation script.
- Alter the specified measure so that it does not refer to the measure being edited.

10506 The existing aggregation setting is invalid with the specified SQL expression. OLAP Center will reset the aggregation setting to *setting*.

Explanation: The existing aggregation setting is invalid with the new SQL expression and was reset to the default aggregation setting. This might be because:

- The data type of the source SQL expression changed.
- The current aggregation setting is expected to be None. It must be None when:
  - The SQL expression is syntactically incorrect when the aggregation functions are not applied to its referred measures, but it is syntactically correct when those aggregation functions are applied. For example, char + int is syntactically incorrect, but COUNT(char) + SUM(int) is syntactically correct.
  - The SQL Expression uses OLAP functions like RANK(), DENSE_RANK() and ROW_NUMBER().

User Response: No action is required.

10507 One or more dimensions in the cube model do not have a hierarchy. They will not be available for inclusion in the cube.

Explanation: For a cube dimension to be created it must be based on a dimension that has at least one hierarchy. You are attempting to create or modify a cube that has one or more dimensions that do not have a hierarchy. These dimensions will be omitted from the selection list used for defining cube dimensions.

User Response: Either create or modify your cube without references to the omitted dimensions or ensure that each dimension in the cube model has a hierarchy.

Appendix E. Status messages from DB2 and DB2 Cube Views

When the DB2 Cube Views stored procedure is called, regardless of whether the stored procedure was executed, DB2 returns an SQLCODE and an SQLSTATE to the calling application. If the DB2 Cube Views stored procedure can execute, the stored procedure returns a status message as part of the XML data that is sent to the calling application.

The following table shows the relationship between the status messages that are returned by metadata operations and the SQLSTATE that is returned by DB2 for the call to the stored procedure.

Table 39. Metadata operation IDs versus SQLSTATE codes

SQLCODE	SQLSTATE	Metadata operation status message IDs	Metadata operation status message types	Metadata operation status messages returned
0	0	0 2	Informational	No
0	0	1	Informational	Yes
0	0	599 6006 6299 7200 7201 7202	Warning	No
462	01HQ1	0 – 7999 (excluding IDs listed in other rows)	Error	No
443	38Q00	Not applicable	Not applicable	Not applicable
443	38Q01	Not applicable	Not applicable	Not applicable
443	38Q02	Not applicable	Not applicable	Not applicable
443	38Q03	Not applicable	Not applicable	Not applicable

Notices

Trademarks

Glossary

This glossary defines terms that are used in this book.

aggregation function. One of the DB2 SQL aggregation functions such as SUM, AVG, MIN, and MAX. The aggregation function is used to control how rollups are performed on measures.

attribute. A DB2 object that maps to either a single column in a table or an expression that is a combination of a set of columns or other attributes or bother. An attribute can perform a number of roles. For example, it can be a reference to data that is in the cube, or it can be a reference to a column that is used by a join or other attribute relationship.

attribute relationship. Describes relationships of attribute objects in general. The relationships are described by a left and a right attribute, a type, a cardinality, and whether they determine a functional dependency. The type describes what the role of the right attribute is with respect to the left attribute. There are two possible types: Descriptive and Associated. The Descriptive type specifies that the right attribute is a descriptor of the left attribute.

balanced hierarchy. A hierarchy with meaningful levels and branches that have a consistent depth. The logical parent of each attribute is in the level directly above it. See *network hierarchy, ragged hierarchy*, and *unbalanced hierarchy*.

calculated measure. Contains built-in calculations that you create by using the Expression Builder in the OLAP Center or with SQL. A calculated measure has an SQL expression that performs calculations and does not map to a single column or attribute.

cube. A DB2 object that is derived from a cube model. The cube facts and cube dimensions are subsets of those that are referenced in the cube model. model. Cubes are appropriate for tools and applications that do not use multiple hierarchies because cube dimensions allow only one cube hierarchy per cube dimension.

cube dimension. A DB2 object that is part of a cube and is derived from a dimension in the cube model that corresponds to the cube. A cube dimension references a subset of the attributes of the dimension from which it is derived. It also references a single cube hierarchy.

cube facts. A DB2 object that is part of a cube and is derived from a dimension in the cube model that corresponds to the cube. A cube facts references a subset of the measures from the facts object from which it is derived.

cube hierarchy. A DB2 object that is part of a cube dimension and is derived from a hierarchy in the dimension that corresponds to the cube dimension. A cube hierarchy references a subset of the attributes of the hierarchy from which it is derived where the order of the attributes must be in the same order as their order in the hierarchy.

cube model. A DB2 object that describes all data related to a collection of measures. Typically, the cube model relates to a star schema or snowflake schema in the database. The cube model references a single facts object and one or more dimensions. Cube models can be optimized to improve the performance of SQL queries issued to the star schema or snowflake schema of the cube model.

dimension. A DB2 object that references a collection of related attributes that describe some aspect of a set of measures. A dimension can reference attributes from one or more dimension tables. However, if attributes from multiple dimension tables are used, the tables must have joins between them and those joins must be referenced by the dimension. A dimension also references one or more hierarchies and can reference relationships between its attributes.

dimension table. A table in a data warehouse whose entries describe data in a fact table. Dimension tables contain the data from which dimensions are created.

facts object. A DB2 object that groups related measures that are interesting to a specific application. The facts object stores information about the attributes that are used in fact to dimension joins, and the attributes and joins that are used to map the additional measures across multiple database tables. Therefore, in addition to a set of measures, a facts object stores a set of attributes and a set of joins. A facts object is used in a cube model as the center of a star schema.

fact table. A central table in a data warehouse schema that contains numerical measures and keys relating facts to dimension tables. Fact tables contain data that describes specific events within a business, such as bank transactions or product sales.

hierarchy. A DB2 object that defines relationships among a set of one or more attributes within a specific dimension of a cube model. DB2 Cube Views supports four types of hierarchies: balanced, unbalanced, ragged, and network. Hierarchies can be deployed as either standard or recursive.

hybrid cube. Contains multidimensional data and references relational data so that you can query lower level data in your base tables.

join. Joins two relational tables. A join references attributes that then reference columns in the tables that are being joined. The simplest form of a join references two attributes: one that maps to a column in the first table and one that maps to a column in the second table. The join also includes an operator to indicate how the columns are compared. A join object can also be used to model composite joins where two or more columns from the first table are joined to the same number of columns in the second table. A composite join uses pairs of attributes to map corresponding columns. Each pair of attributes has an operator that indicates how that pair of columns are compared. A join also has a type and cardinality. Joins can be used in dimensions to join dimension tables together or in a cube model to join the dimensions of the cube model to its facts object or within a facts object to join multiple fact tables.

materialized query table. A table whose definition is based on the result of a query and whose data is in the form of precomputed results that are taken from one or more tables on which the materialized query table definition is based.

measure. A DB2 object that defines a measurement entity and is used in facts objects. Measures become meaningful within the context of a dimension. Common examples of measure objects are Revenue, Cost, and Profit.

metadata. Information about the properties of data, such as the type of data in a column (numeric, text, and so on) or the length of a column. It can also be information about the structure of data or information that specifies the design of objects such as cubes or dimensions.

MQT. See *materialized query table*.

network hierarchy. A hierarchy in which the order of levels is not specified, but in which levels do have semantic meaning. Because the attribute levels do not have an inherent parent-child relationship, the order of the levels is not important. See *balanced hierarchy, ragged hierarchy*, and *unbalanced hierarchy*.

outrigger table. Any dimension table in a snowflake schema that is not the primary dimension table in the dimension.

primary dimension table. In a snowflake schema, the dimension table that joins to the fact table.

ragged hierarchy. A hierarchy in which each level has a consistent meaning but the branches have inconsistent depths because at least one member attribute in a branch level is unpopulated. See *balanced hierarchy, network hierarchy*, and *unbalanced hierarchy*.

recursive deployment. Uses the inherent parent-child relationships between the attributes of the hierarchy. An unbalanced hierarchy that uses a recursive deployment is represented as parent-child attribute pairs.

schema. In the SQL-92 standard, a collection of database objects that are owned by a single user and form a single namespace. A namespace is a set of objects that cannot have duplicate names. For example, two tables can have the same name only if they are in separate schemas, no two tables in the same schema can have the same name.

snowflake schema. An extension of a star schema such that one or more dimensions are defined by multiple tables. In a snowflake schema, only primary dimension tables are joined to the fact table. Additional dimension tables are joined to primary dimension tables.

standard deployment. Uses the level definitions of the hierarchy where each attribute in the hierarchy defines one level. For example, a balanced hierarchy for a Time dimension is usually organized by each defined level including Year, Quarter, and Month. Standard deployment can be used with all four hierarchy types.

star join. A join between a fact table (typically a large fact table) and at least two dimension tables. The fact table is joined with each dimension table on a dimension key.

star schema. A relational database structure in which data is maintained in a single fact table at the center of the schema with additional dimension data stored in dimension tables. Each dimension table is directly related to and usually joined to the fact table by a key column. Star schemas are used in data warehouses.

summary table. Contains aggregated data of the base tables that are used by your cube model. DB2 Cube Views uses DB2 summary tables to improve the performance of queries that are issued to cube models. A summary table is a special type of a materialized query table (MQT) that specifically includes summary data. Because DB2 Cube Views always recommends MQTs that have summarized data, the term summary table is used in the DB2 Cube Views documentation to describe the recommended MQTs. See *materialized query table*.

slice. A region of multidimensional database or cube.

unbalanced hierarchy. A hierarchy with levels that have a consistent parent-child relationship but have an inconsistent semantic meaning for all members in a particular level. Also, the hierarchy branches have inconsistent depths. See *balanced hierarchy, network hierarchy*, and *ragged hierarchy*.

Index

Contacting IBM

If you have a technical problem, please review and carry out the actions suggested by the product documentation before contacting DB2 Cube Views Customer Support. This guide suggests information that you can gather to help DB2 Cube Views Customer Support to serve you better.

For information or to order any of the DB2 Cube Views products, contact an IBM representative at a local branch office or contact any authorized IBM software remarketer.

If you live in the U.S.A., you can call one of the following numbers:

- 1-800-237-5511 for customer support
- 1-888-426-4343 to learn about available service options

Product Information

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-IBM-CALL (1-800-426-2255) or 1-800-3IBM-OS2 (1-800-342-6672) to order products or get general information.
- 1-800-879-2755 to order publications.

http://www.ibm.com/software/data/db2/db2md/
Provides links to information about DB2 Cube Views.

http://www.ibm.com/software/data/db2/udb
The DB2 Universal Database Web pages provide current information about news, product descriptions, education schedules, and more.

http://www.elink.ibmlink.ibm.com/
Click Publications to open the International Publications ordering Web site that provides information about how to order books.

http://www.ibm.com/education/certify/
The Professional Certification Program from the IBM Web site provides certification test information for a variety of IBM products.

Note: In some countries, IBM-authorized dealers should contact their dealer support structure instead of the IBM Support Center.

Comments on the documentation

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 Cube Views documentation. You can use any of the following methods to provide comments:

- Send your comments using the online readers' comment form at www.ibm.com/software/data/rcf.
- Send your comments by electronic mail (e-mail) to comments@us.ibm.com. Be sure to include the name of the product, the version number of the product, and the name and part number of the book (if applicable). If you are commenting on specific text, please include the location of the text (for example, a title, a table number, or a page number).

Program Number: 5724-E15

Printed in U.S.A.

SC18-7298-00

Spine information:

IBM DB2 Cube Views   Setup and User's Guide   Version 8