Conceptual Data Model

CHAPTER 5

IN THIS CHAPTER

- Data Definition Language (DDL)
- Objects in DDL

The conceptual multidimensional data model is the foundation for multidimensional databases. All the components and architecture of a multidimensional database create, control, and provide access to the data in the model. Because it is simple and flexible, not to mention effective, our model has led to widespread adoption of Analysis Services in a very short period of time.

Many people look at the multidimensional data model as simply metadata—data that describes the data stored in the relational database. We're going to look at this from a different angle. We see the conceptual model as an independent specification of the data in the multidimensional system. A relational database might be the source of the data or the place where the data is stored. But the multidimensional database is a completely independent system that can be both the source and the storage place of the data. If the source of the data is external to the multidimensional database, it is defined by the `Data Source` property. Any dependency between multidimensional data and relational data is defined by the `Data Binding` property.

Data Definition Language

We use Data Definition Language (DDL) to define and alter the data models. Extensible Markup Language (XML), which has grown popular among software developers in recent years, turns up in many of the components of our system, including DDL. As the foundation for DDL, XML is easy to use, convenient, and efficient. Throughout this

book we use DDL a lot to describe the data model, so you'll want to be familiar with it. But we're going to focus our discussion here on the semantic properties of the language. You can look for details of the syntax of DDL in Books Online.

DDL is object oriented. It enables you to define a set of objects that are part of the multidimensional model and to define all the properties necessary to those objects.

Objects in DDL

All the objects in DDL are either major objects or minor objects. *Major objects* are objects that the user can manipulate—independently of their parent objects—to create and change the model. *Minor objects* are children of major objects. The root object (which is a major one) of the model is `Database` (sometimes called `Catalog`), which contains a list of all the objects of the model (see Listing 5.1).

Major objects must have two unique identifiers: the `ID` and `Name` properties. A minor object that is part of a major object doesn't need these properties. In addition, each object (major or minor) can have a `Description` property that contains text that describes the purpose of the object (useful for the developer who created the object and the user of the application that uses the object). Objects can also have the `Annotation` property, or lists of annotations, that external applications use to display or manipulate their data.

LISTING 5.1 The DDL Definition of the FoodMart Database

```
<Database
xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
     <ID>FoodMart 2005</ID>
     <Name>FoodMart 2005</Name>
     <CreatedTimestamp>0001-01-01T08:00:00Z</CreatedTimestamp>
     <LastSchemaUpdate>0001-01-01T08:00:00Z</LastSchemaUpdate>
     <LastProcessed>0001-01-01T08:00:00Z</LastProcessed>
     <State>Unprocessed</State>
     <LastUpdate>0001-01-01T08:00:00Z</LastUpdate>
     <DataSourceImpersonationInfo>
          <ImpersonationMode>Default</ImpersonationMode>
          <ImpersonationInfoSecurity>Unchanged</ImpersonationInfoSecurity>
     </DataSourceImpersonationInfo>
     <Dimensions />
     <Cubes />
     <DataSources />
     <DataSourceViews />
  <Translations />
</Database>
```

You can see in the example that the database contains collections of the object, dimensions, cubes, and so forth. (The ending *s* on *Dimensions*, *Cubes*, and so on denotes a collection.) The `Dimension` and `Cube` objects are major objects and can be changed

independently of the database definition. You can find detailed information about dimensions in Chapter 6, "Dimensions in the Conceptual Model," and about cubes in Chapter 7, "Cubes and Multidimensional Analysis."

Figure 5.1 contains the most important objects of our multidimensional model, with major objects in darker gray. Objects that represent the physical model and objects that represent database security aren't included in the figure. These objects will be discussed in later chapters.

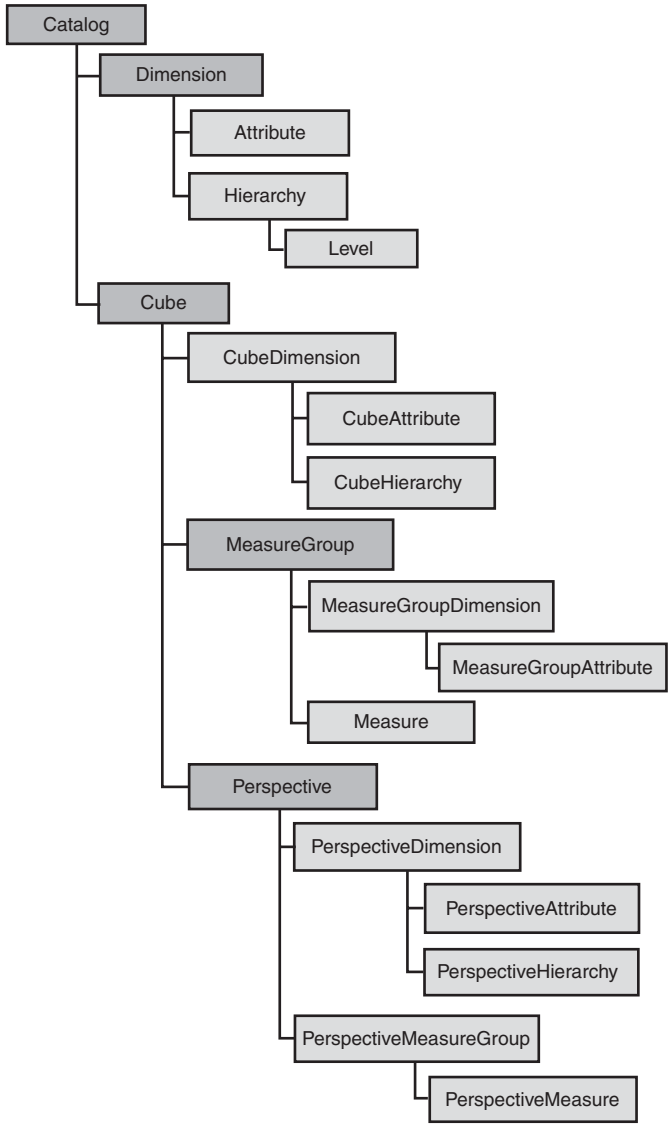

FIGURE 5.1 The major objects of the conceptual model are shown in darker gray.

In the following sections, we'll give you an idea of some of the properties that are commonly used in our conceptual model:

- Multilanguage support
- Ways of ordering your data
- Ways to specify default properties

NOTE

When you specify the identifier, name, and translation of an object, you choose from a limited set of characters; in addition, the strings are limited in length. It's important to pay attention to all these limitations, because usually it takes a long time to figure out what's causing an error or strange behavior that is related to errors in the names. Sometimes the fix of an error like this can require a change in design.

Multilanguage Support

Our model features multilanguage support, which comes in handy for the trend toward internationalization and globalization characteristic of today's enterprises. That support means that the data warehouse can contain data in multiple languages, which, of course, affects data storage requirements. The object's `Language` (sometimes known as *locale*) property is the identifier for any specific language; it is used for both the metadata and in the data itself.

The related `Translation` property specifies what the name of the object will be for the language specified by the `Language` property (see Listing 5.2). The `ID` property of the object, once it's specified, can't be changed. The name of the object and the translation of that name can be easily changed, and because of that can't be used for cross references.

LISTING 5.2 Translating a Database Object Name to Russian

```
<Database xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
     <ID>FoodMart 2005</ID>
     <Name>FoodMart 2005</Name>
     <Translations>
          <Translation>
               <Language>1049</Language>
               <Caption>Сеть Продуктовых Магазинов </Caption>
          </Translation>
     </Translations>
</Database>
```

If you specify this translation in DDL definition, the user application will have access to the caption that is specified in the translation, and can use it in place of the name wherever necessary.

Rules of Ordering

The order of object elements is not all that important, but when possible, Analysis Services preserves the order that you assigned for the elements. However, for some objects, Analysis Services assigns a new order, usually based on the alphabet.

If the order is based on the alphabet, the order can differ from one language to another. Not only does alphabetic order change from one language to another, but there are rules of ordering, defined by the `Collation` property, that add different bases for ordering. `Collation` and its properties, such as `Ignore Case` or the ignoring of special characters, define different ways of ordering for different languages. If you don't specify a collation, Analysis Services uses the default collation, `Case Insensitive`.

Specifying Default Properties

The DDL language has rules for specifying default properties. These rules specify the values that properties take if they're not explicitly specified in the object definition. Usually if there isn't a specific default value, the server assigns a value. This means that it's not always possible to predict what value will be assigned. It also might turn out that in the next version of Analysis Services, the default values will be interpreted differently. It's a good idea to avoid situations where the server would define values for you. However, if you're not interested in the value, you can just go with whatever value the server assigns.

Another rule holds that you don't need to specify a collection of objects if the collection is empty. The server doesn't have default values for empty collections; it assumes that an empty collection doesn't have any objects, and therefore no values. (We would assume that, too.) However, there are some cases in which the server would copy a collection from another object. (For more information about exceptions to these rules, see Chapters 7 and 8, "Measures and Multidimensional Analysis.")

Summary

All the components and architecture of a multidimensional database create, control, and provide access to the data in the conceptual model. The conceptual model is an independent specification of the data in the multidimensional system.

Data Definition Language (DDL), based on XML, is used to define and alter the data models. Objects in DDL are either major objects or minor objects. The DDL object's `Language` property (sometimes known as *locale*) provides multilanguage support. Rules of ordering are defined by the `Collation` property.

This page intentionally left blank