InterSystems CACHÉ®

Using the DeepSee Architect

Version 2011.1
26 May 2011

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Using the DeepSee Architect
Caché Version 2011.1 26 May 2011
Copyright © 2011 InterSystems Corporation
All rights reserved.

This book was assembled and formatted in Adobe Page Description Format (PDF) using tools and information from the following sources: Sun Microsystems, RenderX, Inc., Adobe Systems, and the World Wide Web Consortium at www.w3c.org. The primary document development tools were special-purpose XML-processing applications built by InterSystems using Caché and Java.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Customer Support

Tel:	+1 617 621-0700
Fax:	+1 617 374-9391
Email:	support@InterSystems.com

Table of Contents

List of Tables

About This Book

Important: This book is for DeepSee I. For information on DeepSee II, see *Getting Started with DeepSee II*, which also lists the other books for DeepSee II.

This book describes how an implementer can use DeepSee Architect to create a DeepSee model for use in pivot tables. It contains the following sections:

- Introduction to the DeepSee Architect
- Core Modeling Principles
- Enabling Classes for Use in DeepSee
- Defining Dimensions
- Defining Dynamic Dimensions
- Defining Compound Members
- Defining Relationships between Dimensions
- Defining Measures
- Defining Subject Areas
- Defining Listing Fields
- Using Other Tables to Define Dimensions, Measures, and Listing Fields
- Defining Detail Listings
- Using Replacements, Ranges, and Transformations
- Validating Your DeepSee Data Model
- When to Recompile and Rebuild

For a detailed outline, see the table of contents.

For more information, see the following books:

- *Overview of DeepSee*, an introductory guide for all users who are interested in learning about DeepSee.
- *DeepSee Model Design Guide*, an introductory guide for implementers and business users.
- *DeepSee Developer Tutorial*, a tutorial for implementers who are creating DeepSee models, pivot tables, and dashboards.
- *Using the DeepSee Connector*, a guide for implementers who are using the DeepSee Connector to import externally stored data. Note that the DeepSee Connector is available only with Ensemble.
- *Using the DeepSee Analyzer*, a guide for implementers and advanced users who want to create pivot tables to embed in applications — or who simply want to explore their data.
- *Using the DeepSee Dashboard Designer*, a guide for implementers who are using the Dashboard Designer to create dashboards.
- *Expressions and Scripts in DeepSee*, an implementer guide that describes the syntax and options for all formulas, expressions, and scripts supported in DeepSee. This book also lists all the locations where you can use these expressions and scripts.

- *DeepSee Site Configuration and Maintenance Guide*, a guide for implementers and system administrators. This book describes how to configure and maintain a DeepSee site. It also includes a chapter that lists common problems and their solutions.
- *DeepSee User Guide*, a user manual for your end users. This book describes how to work with deployed dashboards and pivot tables.

For general information, see the *InterSystems Documentation Guide*.

1
Introduction to the DeepSee Architect

Important: This book is for DeepSee I. For information on DeepSee II, see *Getting Started with DeepSee II*, which also lists the other books for DeepSee II.

This chapter introduces the DeepSee Architect. It discusses the following topics:

- Purpose of the Architect
- How to log in to DeepSee
- How to access the DeepSee Architect
- How to switch to another namespace

Be sure to consult *InterSystems Supported Platforms* for information on system requirements.

Also, this book assumes that you are familiar with the concepts described in the book *Overview of DeepSee*.

1.1 Purpose of the DeepSee Architect

You use the DeepSee Architect to create DeepSee models, which in turn are needed for creating pivot tables and for using the Analyzer for analysis. When you are done using the Architect, you will have created all the following elements:

- BI-enabled classes.
- Dimensions based on properties or source expressions using properties of those classes.
- Measures based on properties and source expressions. You can add more measures with the DeepSee Analyzer, which is discussed in another book.
- Detail listing fields based on properties and source expressions.
- Detail listings associated with different user roles, as needed.
- Subject areas associated with different user roles, as needed.

The Architect also provides options for exporting and importing model elements; these options are documented in the *DeepSee Site Configuration and Maintenance Guide*.

1.2 Logging In to DeepSee

To log in to DeepSee:

1. In Internet Explorer, go to the following URL:

   ```
   http://localhost:57772/csp/sys/bi/default.htm
   ```

   Where *localhost* is the server on which Caché is running, and *57772* is the port used by the web server.

   If you have not yet specified a namespace, the system displays a page that prompts you for a namespace. Otherwise, the system displays the DeepSee login page.

2. If you are prompted for a namespace, type the name of the namespace you want to work in and then click **Logon to DeepSee**.

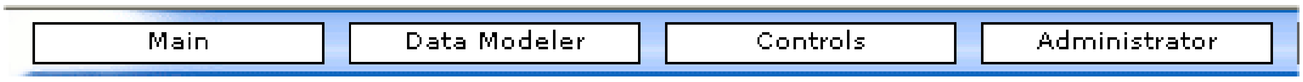
   The system then displays the DeepSee login page.

3. If you are prompted for a namespace, type the name of the namespace you want to work in and then click **Logon to DeepSee**.

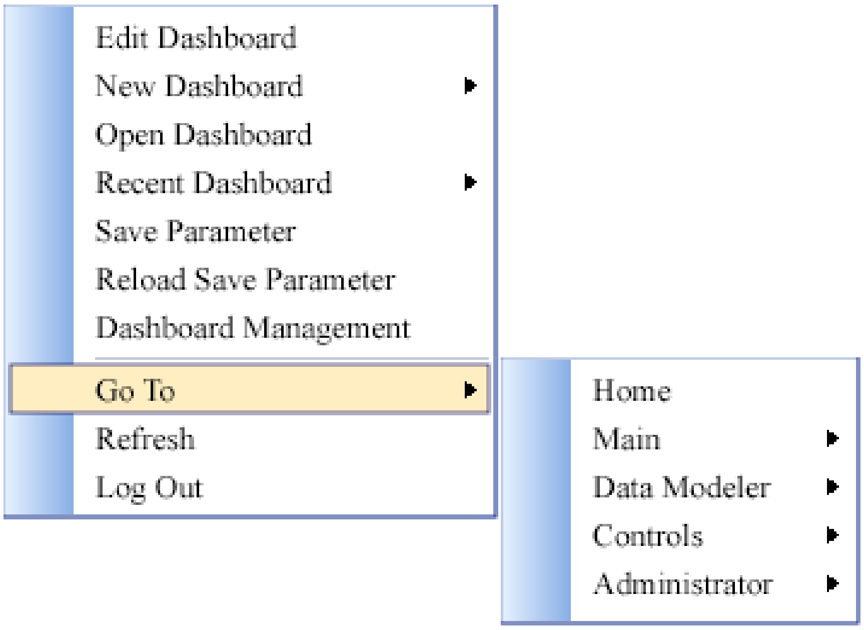
   The system then displays the DeepSee login page.

4. On the DeepSee login page, enter a DeepSee username and password. For example, you can use the username `demo` with the password `demo`.
5. For **Role**, select `demo`.
6. Click **Login**.

DeepSee displays the home page, which depends upon the user ID you used to log in. The home page is either a DeepSee module or a dashboard. If the page is a DeepSee module, it has a row of buttons at the top as follows:



If the page is a dashboard, the context menu provides access to all the same options provided by these buttons, in addition to options that apply to dashboards:

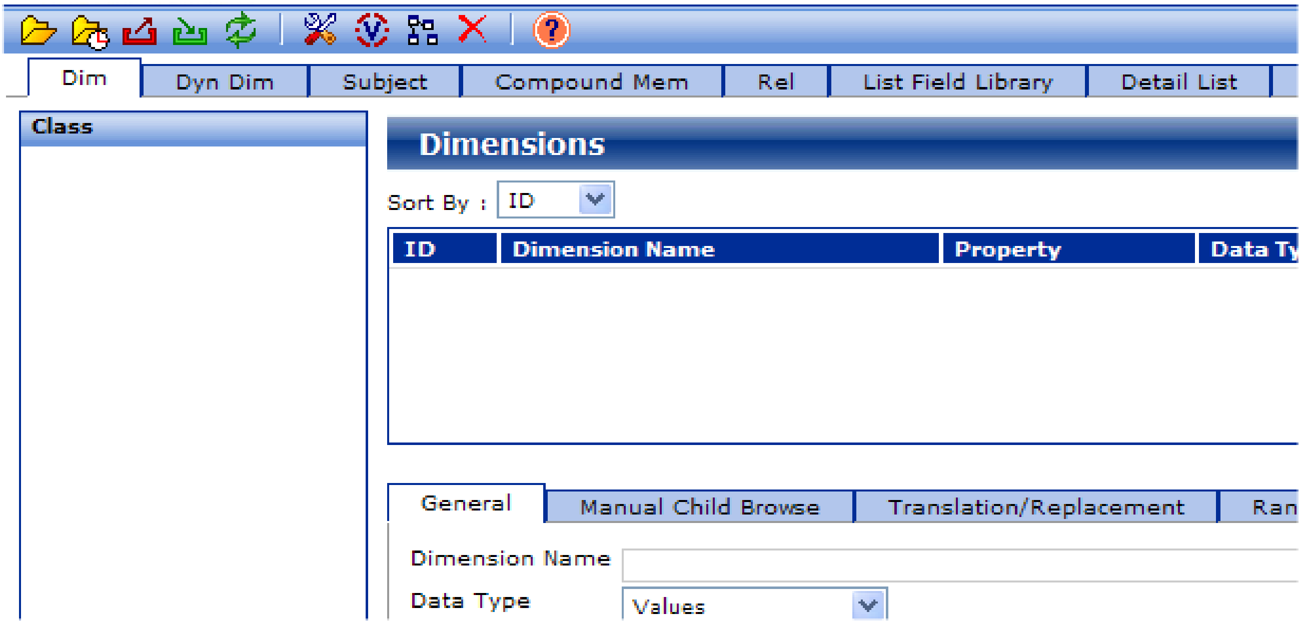


1.3 Accessing the DeepSee Architect

To access the Architect:

- If you are currently viewing a DeepSee module, click **Data Modeler > Architect**.
- If you are currently viewing a dashboard, right-click and then click **Go To > Data Modeler > Architect**.

The Architect looks like this:



1.4 Switching to Another Namespace

To switch to a different namespace:

1. Log out. To do so, do one of the following, depending on what you are currently viewing:

    - If you are currently viewing a DeepSee module, click **Log Off** in the upper right.
    - If you are currently viewing a DeepSee dashboard, right-click and then click the **Log Out** option.

    The system logs you out of DeepSee.

2. Click **Switch Namespace**.
3. For **Namespace**, type the name of the namespace you want to work in.
4. Click **Logon to DeepSee**.

    This displays a login page.

5. Log in as usual.

2

Core Modeling Principles

This chapter discusses the core principles that affect your DeepSee models:

- A measure value must have a one-to-one relationship with the base table
- Be careful with collection-based dimensions
- Make sure to handle null values appropriately

This chapter assumes that you are familiar with the concepts of DeepSee, as discussed in the *Overview of DeepSee*.

2.1 Measures Must Have a One-to-one Relationship with the Base Table

The value on which a measure is based must be one-to-one with the records in the base table. Otherwise, DeepSee would not aggregate that measure in the way that you intend it to. This section demonstrates this principle.

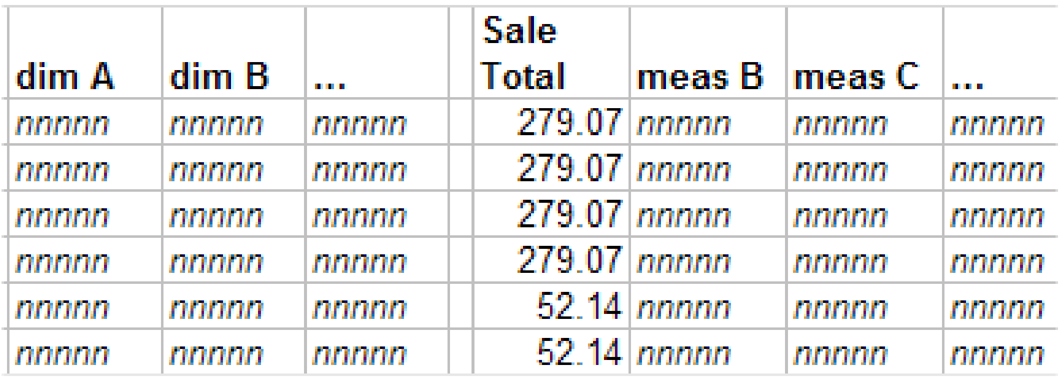
2.1.1 Measures from Parent Tables

Do not base a measure on a field in a parent table. For example, consider the following two tables:

- `Order` — Each row represents an order submitted by a customer. The field `SaleTotal` represents the total monetary value of the order.
- `OrderItem` — Each row represents an item in that order. In this table, the field `OrderItemSubtotal` represents the monetary value of this part of the order.

Suppose that we use `OrderItem` as the base table. Also suppose that we define the measure `Sale Total`, based on the parent's `SaleTotal` field. The goal for this measure is to display the total sale amount for all the sales of the selected order items.

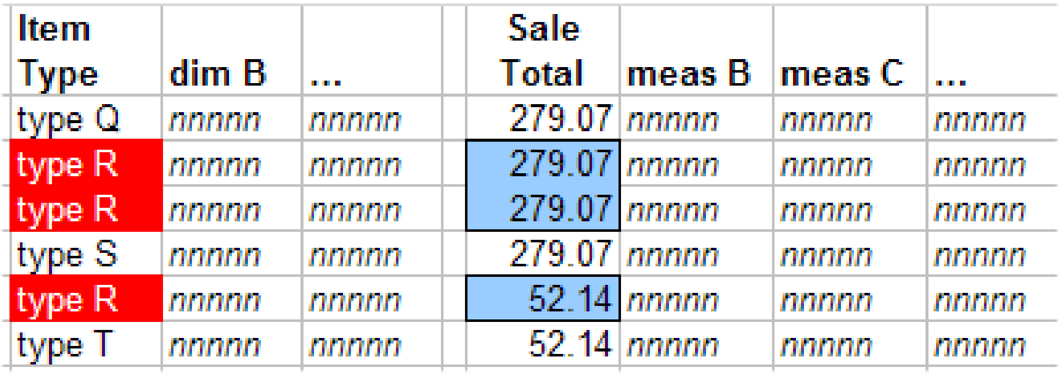
Let us consider the contents of the fact table. The following shows an example:



dim A	dim B	...		Sale Total	meas B	meas C	...
nnnnn	*nnnnn*	*nnnnn*		279.07	*nnnnn*	*nnnnn*	*nnnnn*
nnnnn	*nnnnn*	*nnnnn*		279.07	*nnnnn*	*nnnnn*	*nnnnn*
nnnnn	*nnnnn*	*nnnnn*		279.07	*nnnnn*	*nnnnn*	*nnnnn*
nnnnn	*nnnnn*	*nnnnn*		279.07	*nnnnn*	*nnnnn*	*nnnnn*
nnnnn	*nnnnn*	*nnnnn*		52.14	*nnnnn*	*nnnnn*	*nnnnn*
nnnnn	*nnnnn*	*nnnnn*		52.14	*nnnnn*	*nnnnn*	*nnnnn*

The first four rows represent the items in the same order. The next two rows represent the items of another order, and so on.

Suppose that this model has a dimension called `Item Type`. Let us examine what happens when DeepSee retrieves records for all items of type R:



Item Type	dim B	...		Sale Total	meas B	meas C	...
type Q	*nnnnn*	*nnnnn*		279.07	*nnnnn*	*nnnnn*	*nnnnn*
type R	*nnnnn*	*nnnnn*		279.07	*nnnnn*	*nnnnn*	*nnnnn*
type R	*nnnnn*	*nnnnn*		279.07	*nnnnn*	*nnnnn*	*nnnnn*
type S	*nnnnn*	*nnnnn*		279.07	*nnnnn*	*nnnnn*	*nnnnn*
type R	*nnnnn*	*nnnnn*		52.14	*nnnnn*	*nnnnn*	*nnnnn*
type T	*nnnnn*	*nnnnn*		52.14	*nnnnn*	*nnnnn*	*nnnnn*

To compute the value of the `Sale Total` measure for type R, DeepSee adds together the three values shown here: 279.07, 279.07, and 52.14. But this action double-counts one of the orders.

Depending on the case, the `Sale Total` measure might aggregate correctly; that is, it might show the correct total sales figure for the selected order items. But you cannot ensure that the measure does this, because you cannot prevent double-counting as shown as in this example.

2.1.2 Measures from Child Tables

You can use a value in a child table as the basis of a measure, but to do so, you must aggregate that value across the relevant rows of the child table.

Consider the following two tables:

- `Customer` — Each row represents a customer.
- `Order` — Each row represents a customer order. The field `SaleTotal` represents the total monetary value of the order.

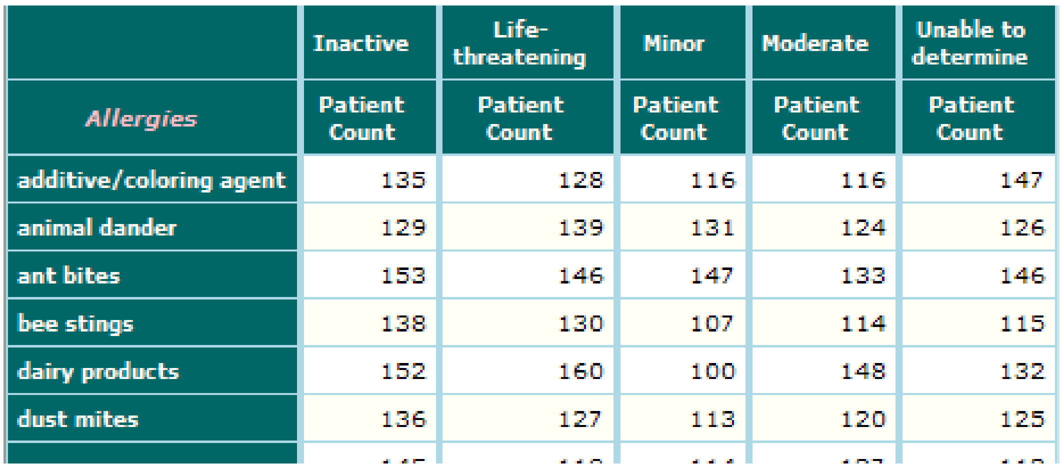
Suppose that we use `Customer` as the base table, and that we want to create a measure based on the `SaleTotal` field.

Because a customer potentially has multiple orders, there are multiple values for `SaleTotal` for a given customer. To use this field as a measure, we must aggregate those values together. The most likely options are to add the values or to average them, depending on the purpose of this measure.

2.2 Be Careful With Collection-based Dimensions

In DeepSee, unlike many other BI tools, you can base a dimension upon a list value. Such dimensions are useful, but it is important to make sure that the users understand the behavior of the dimension.

For example, a patient can have multiple allergies. Each allergy has an allergen and a severity. Suppose that the base table is `Patients` and the model includes the `Allergen` and `Allergy Severity` dimensions. We could create a pivot table that looks like this:



	Inactive	Life-threatening	Minor	Moderate	Unable to determine
Allergies	Patient Count	Patient Count	Patient Count	Patient Count	Patient Count
additive/coloring agent	135	128	116	116	147
animal dander	129	139	131	124	126
ant bites	153	146	147	133	146
bee stings	138	130	107	114	115
dairy products	152	160	100	148	132
dust mites	136	127	113	120	125

Upon first seeing this pivot table, the user might think that this pivot table shows correlations between different sets of patient allergies. It does not.

This pivot table, as with all other pivot tables in this subject area, shows sets of patients. For example, the `ant bites` row represents patients who have an allergy to ant bites. The `Minor` column represents patients who have at least one allergy that is marked as minor. There are 147 patients who have an allergy to ant bites and at least one allergy that is marked as minor. This does not mean that there are 147 patients with minor allergies to ant bites.

It is possible to create a pivot table that does show correlations between different sets of patient allergies. To do so, however, you would have to define a model based on the patient allergy, rather than the patient.

2.3 Make Sure to Handle Null Values Correctly

For any measure or dimension, the value you use as a basis could potentially have no value in some cases. It is important to understand how DeepSee handles nulls and to adjust your model according to your business needs and usability requirements.

2.3.1 Null Values in a Measure

For a measure, if the source value is missing, DeepSee treats the null value as zero. This can adversely affect any measure that aggregates by averaging or by using the minimum value.

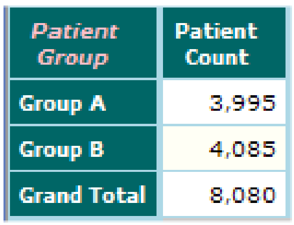
For example, in the Patients sample, you can create an `Average Test Score` measure based on the TestScore property of the patient. Some patients do have a null value for the TestScore property, and the null scores are treated as 0. However, the lowest possible score on the “test” is 50. If you compute the average using all patients, the results are incorrect.

To properly compute the average test score, you should use only patients that have a value for the TestScore property. In this case, you would create a filtered measure that considers only those patients. For details, see *Using the DeepSee Analyzer* (you create this kind of measure in the Analyzer). Also see the *DeepSee Developer Tutorial*.

2.3.2 Null Values in a Dimension

For a dimension, if the source value is missing for a given record in the base class, DeepSee omits that record from the dimension, by default. That is, DeepSee does not automatically create a member to contain the null values. Then when you use this dimension, DeepSee filters out the records that have null values.

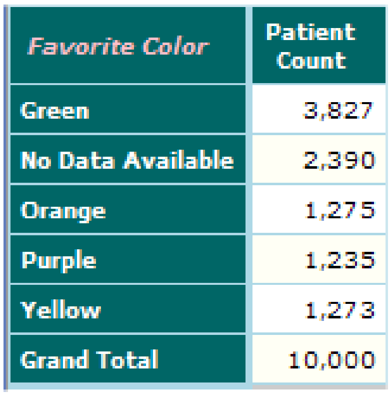
For example, in the Patients sample, you can create a `Patient Group` dimension based on the PatientGroup property of the patient. For some patients, this property is not specified. If you display this dimension in a pivot table, you might see something like this:



Patient Group	Patient Count
Group A	3,995
Group B	4,085
Grand Total	8,080

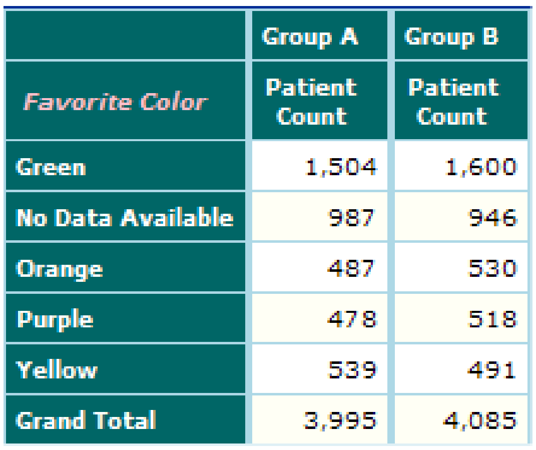
In this case, however, there are 10000 patients, which is greater than the total shown here. This pivot table includes only the patients that have a value for the PatientGroup property.

Also consider the following pivot table:



Favorite Color	Patient Count
Green	3,827
No Data Available	2,390
Orange	1,275
Purple	1,235
Yellow	1,273
Grand Total	10,000

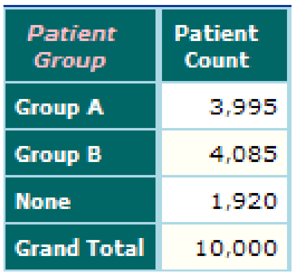
Suppose that you redefine this by adding the `Patient Group` dimension as columns:



	Group A	Group B
Favorite Color	Patient Count	Patient Count
Green	1,504	1,600
No Data Available	987	946
Orange	487	530
Purple	478	518
Yellow	539	491
Grand Total	3,995	4,085

By adding the `Patient Group` dimension to this pivot table, we automatically filtered out the patients with no value for the PatientGroup property.

In these cases, your users might prefer to see a placeholder member. For a given dimension, the placeholder member would be used to collect all records that have a null value for this dimension, and the dimension would not omit any records. For example:



Patient Group	Patient Count
Group A	3,995
Group B	4,085
None	1,920
Grand Total	10,000

Depending on how you create the dimension, there are different options for handling null values. For details, see the chapter "Defining Dimensions." Also see the *DeepSee Developer Tutorial*.

3
Enabling Classes for Use in DeepSee

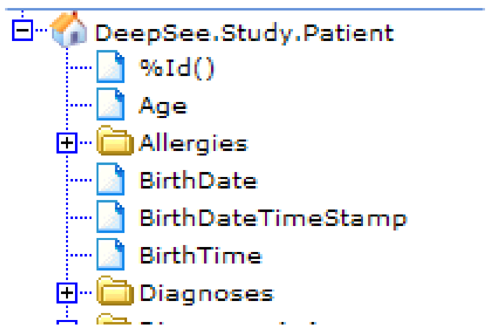
Within the Architect, you can perform most activities in nearly any order. In all cases, however, you start by enabling Caché class definitions for use with DeepSee. This process is known as BI-enabling the base class; you may need more than one base class. This chapter discusses the following topics:

- What it means to BI-enable a class
- How to BI-enable a class
- How to open a class definition from the Architect
- How to open a BI-enabled class in the Architect
- How the Architect displays the properties and methods of the BI-enabled class
- How to recompile the class and rebuild the indices
- How to selectively rebuild the indices
- How to specify the number of processes used when DeepSee rebuilds the indices
- How to clear unsaved work in the Architect
- How to create SQL shortcuts to manage confusing property names
- How to remove %BI.Adaptor from a class

3.1 Overview

A BI-enabled class is a class that inherits from %BI.Adaptor. When you BI-enable a class, you can create dimensions, measures, and listing fields that use data in this class or in any related classes. You can either directly use properties or you can write Caché ObjectScript expressions that retrieve the data.

To enable you to access properties directly, the DeepSee Architect displays the BI-enabled class and most related classes in a convenient treelike display; the other classes do not need to be BI-enabled. For example:



You can expand the folders, which contain more properties and possibly more class definitions.

The root item is special; this class is the *base class* in this cube. The base class determines the meaning of the predefined `Count` measure, as we will see later in this chapter.

3.2 BI-enabling Your Classes

The first step depends on where your data is now:

- If you have Caché classes, you can open them from within Architect. This automatically adds the DeepSee superclass (%BI.Adaptor) to the superclass list. See the following section.

  In some cases, you cannot modify the class or you have an existing complex query that you want to use. DeepSee supports an alternative approach in which you create and BI-enable a class query; see "Using a Class Query," later in this book.

- If you are creating new classes, create them in Studio. For any class that needs it, include the DeepSee superclass (%BI.Adaptor) in the superclass list.

- If you are starting with an existing Caché application that does not use Caché class definitions, map the existing globals to class definitions and then add the DeepSee superclass (%BI.Adaptor) to the superclass list.

- If your data is stored outside of Caché, use the DeepSee Connector to generate BI-enabled class definitions and to load the data into Caché.

In all cases, the Architect automatically has access to any classes in this namespace that extend %BI.Adaptor, as well as to all object-valued properties of those classes, recursively.

The Architect accesses and modifies the same class definitions that you see directly in Studio. This means that if you make a change in one location and save it, that change is immediately available in the other.

For convenience, you can recompile and rebuild the base class as needed from Architect.

3.3 Opening a Class Definition

When you open a class definition in the Architect, the Architect automatically adds %BI.Adaptor to its superclass list, if that is not yet included.

To open a class definition in the Architect:

1. First, do any of the following:

- Click the Open Class button in the menu bar. The Architect displays a dialog box. Click **Add** in the lower left of this dialog box. Then click **Search**.
- Click the **Base Class** link in the lower left. The Architect displays a dialog box. Click **Add** in the lower left of this dialog box. Then click **Search**.
- Click **Add** in the lower left.

In all cases, the Architect displays a search dialog box. The appearance of this dialog box depends on the path you used.

2. In the search dialog box, optionally type the name of the package that contains the class. If you do not specify a package name, the Architect searches all packages in this namespace.
3. Click **Search**.

   The dialog box then displays all classes in this namespace or all classes in this package.
4. Click the class name and then click **OK**.

3.4 Opening a BI-enabled Class

If a given class already extends %BI.Adaptor, you can open it in fewer steps (no search is required).

To open a BI-enabled class definition in the Architect:

1. First, do either of the following:
   - Click the Open Class button in the menu bar.
   - Click the Open Recent Class button in the menu bar.
2. Click the class name and then click **OK**.

3.5 Display of Properties and Methods of a BI-enabled Class

The following rules govern the display of a class in the Architect:

- For a BI-enabled class, all properties are shown except for the parent side of a parent-child relationship.
- This display is recursive; that is, properties of properties are shown.
- If a property is a collection (a list or an array), it is shown as a folder that contains the properties and methods of the collection.
- If a property is of type %List (which is the object equivalent of **$LISTBUILD**), it is not shown as a folder.
- If a property is a relationship, it is not shown as a folder.
- If a class is not accessible from the base class via Caché dot syntax, it is not shown by default.

- The Architect does not display properties and methods inherited from superclasses. You can use these properties and methods anyway.

This view of the class provides a useful starting point for creating dimensions, measures, and listing fields. In many cases, you can simply create an element by double-clicking a property within this hierarchy.

But it is important to know that although this view provides a convenient way to access some properties, you can in fact access any property, and you can access data in any table by using **Complex Code**.

3.6 Recompiling and Rebuilding

To recompile, rebuild, or both:

1. Click the Process button in the menu bar.

   DeepSee displays a dialog box that shows the results from the last time you recompiled or rebuilt.

2. In this dialog box, do one of the following:

   - To recompile the base class (the BI-enabled class), click **Compile**.
   - To recompile the base class and rebuild the indices, click **Rebuild**.

   Tip: This dialog box is not always automatically refreshed when the compilation or rebuild status changes. To refresh the display and see the current status, click **Refresh**.

3. When you are done, click **Cancel**.

You can also recompile in Studio, in the same way that you recompile any other class definitions. In case of compilation errors, Studio sometimes provides more detailed messages than the Architect does.

To rebuild all the indices programmatically, use the Caché Terminal, go to the appropriate namespace, and enter the following command, where *classname* is the package and class name of the BI-enabled class:

```
 Do ##class("classname").HiRebuild(0)
```

3.6.1 Incremental Rebuilds

After initial development, it is not usually desirable to rebuild all the indices when the data changes, because that can be time-consuming. Instead, your application should detect data changes and update the indices for only the affected objects. There are three parts to this approach:

- You must enable DeepSee to rebuild the indices incrementally. To do this, enable the site option **ETL > Incremental Index Update** and then recompile the base class. See the *DeepSee Site Configuration and Maintenance Guide*.

  Despite the fact that this option is in the **ETL** group, it does apply to all BI-enabled classes in this namespace (it is not specific to classes created via the DeepSee Connector).

- If a data change affects a record in the base class, DeepSee automatically detects that change and updates the indices for that record.

  This occurs because the base class inherits from %BI.Adaptor, which provides the methods %OnAfterSave() and %OnDelete() and the triggers **HyperDelete**, **HyperInsert**, and **HyperUpdate**, all of which automatically update the DeepSee indices.

For example, when an object is saved, Caché runs the method %OnAfterSave(), which updates the DeepSee indices for this object. Similarly, when an record is deleted, inserted, or updated via SQL, Caché fires the corresponding trigger, which updates the DeepSee indices for this record.

- If a data change affects a record in another class to which the base class refers, it is the responsibility of your application to do the following:
  1. Determine the IDs of the affected objects in the base class.
  2. Update the DeepSee indices for those objects by using the **zzBuildOne()** method of the base class (inherited from %BI.Adaptor). The first argument for this method is the ID of the object whose indices need to be updated.

  For an example, see the *DeepSee Developer Tutorial* and the DeepSee.Study sample classes in the SAMPLES database.

3.7 Rebuilding Selectively

To rebuild a subset of dimensions and listing fields:

1. Click the Selective Rebuild button in the menu bar.
2. You are prompted to confirm this action. Click **OK**.

Also see "Controlling Whether a Dimension Is Selectively Rebuilt" and "Controlling Whether a Listing Field Is Selectively Rebuilt."

3.8 Specifying the Number of Processes

By default, DeepSee uses one process when it builds the indices. If you are rebuilding the DeepSee indices on a machine that has multiple CPUs, you can increase the number of processes that DeepSee uses. To do so:

1. Click the **Sys** tab.
2. For **Number of Processors**, specify an integer. Typically you use a number that is the same as or slightly larger than the number of processors.
3. Click **Save**.

You can also specify this option system-wide. See the chapter "Configuring the DeepSee Site" in the book *DeepSee Site Configuration and Maintenance Guide*.

3.9 Clearing Unsaved Work

To clear any unsaved work in the Architect, click the Refresh button . The unsaved changes are immediately cleared.

3.10 Defining SQL Shortcuts

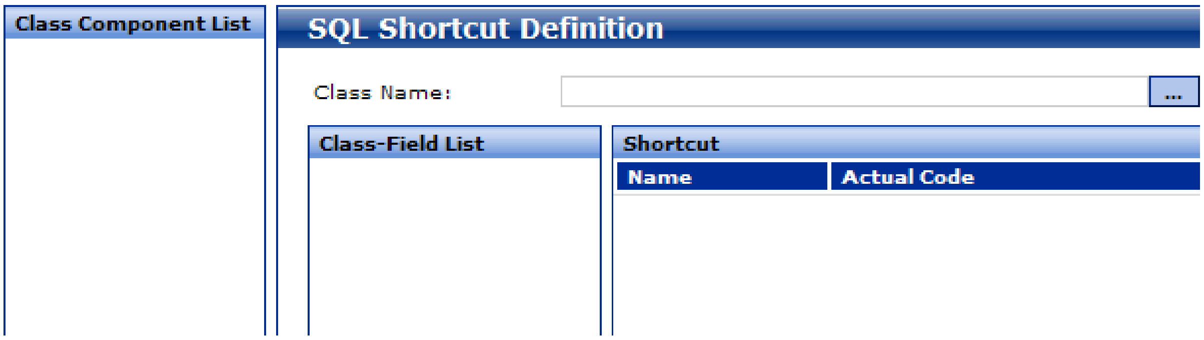
If the property names in your classes are confusing and if you have many properties, you might find it convenient to define *SQL shortcuts* before you start to create dimensions, measures, and listing fields.

In DeepSee, an SQL shortcut is an alias that provides a new name for a class property (or property of a property), to make it easier to identify the properties you want to use in DeepSee.

To define SQL shortcuts:

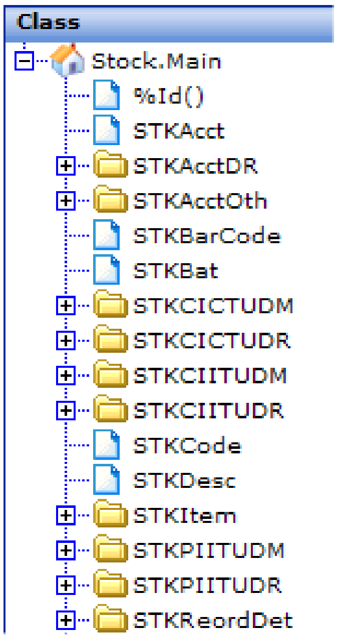
1. At the top of the Architect page, click **Data Modeler > SQL Field Shortcuts**.

   The system then displays a page like the following:

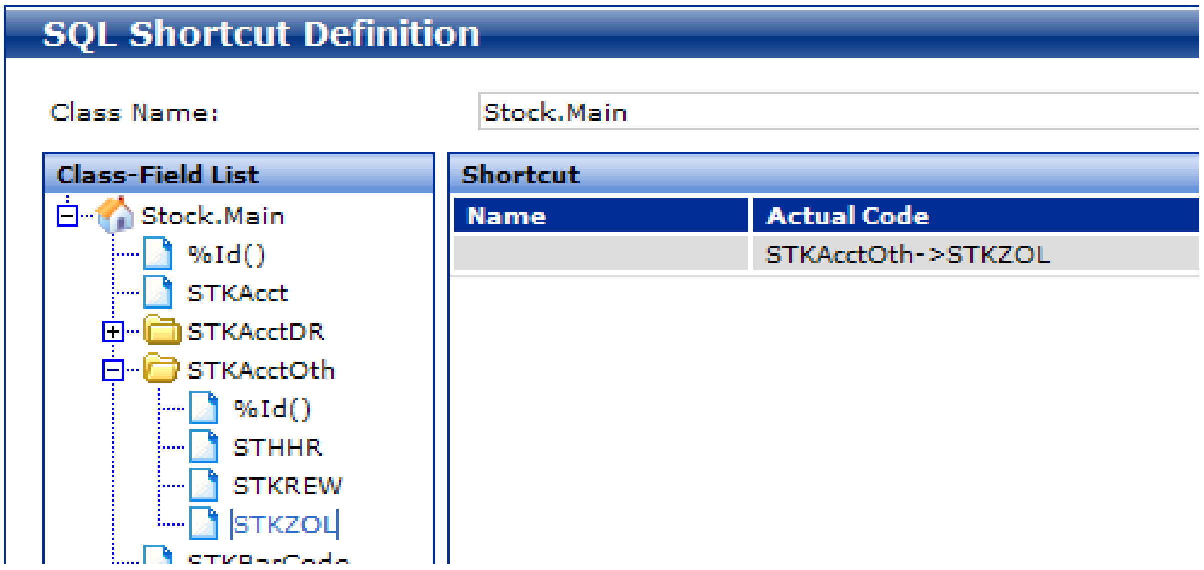


2. Click the Browse button **...** next to **Class Name**.
3. Click a class name and then click **OK**.

   The class is displayed in **Class-Field List**.
4. Expand the class folder and subfolders in **Class-Field List**, as needed. As the following example shows, not all class definitions contain only simple, easy-to-read property names:



5. To create an SQL shortcut for any property, double-click that property. When you do, the system adds a row to the **Shortcut** table. For example, if you double-clicked the property STKZOL within the property STKAcctOth, you would see the following:


SQL Shortcut Definition

Class Name: Stock.Main

Class-Field List
- Stock.Main
  - %Id()
  - STKAcct
  - STKAcctDR
  - STKAcctOth
    - %Id()
    - STHHR
    - STKREW
    - STKZOL
  - STKBarCode

Shortcut

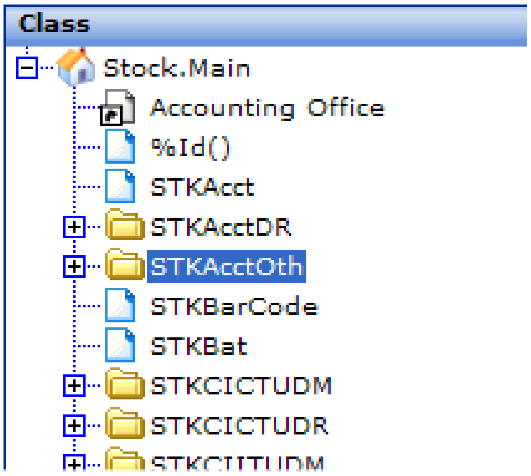
Name	Actual Code
	STKAcctOth->STKZOL

6. Type a string into **Name** in the same row. Typically, this string is easier to remember and understand than the field name. For example: `Accounting Office`.

7. Optionally type a tooltip into **Tooltip**.

8. Click **Save**.

When you next view the class in the Architect, the shortcuts are shown at the top, as follows:


Class
- Stock.Main
  - Accounting Office
  - %Id()
  - STKAcct
  - STKAcctDR
  - STKAcctOth
  - STKBarCode
  - STKBat
  - STKCICTUDM
  - STKCICTUDR
  - STKCIITUDM

If you hover the pointer over a shortcut, the Architect displays the tooltip for the shortcut. You can use these shortcuts when you define dimensions and listing fields.

3.11 Removing %BI.Adaptor from a Class

In some cases, you might find that you want to remove %BI.Adapter from the superclass list of a class.

Before you do this, do the following:

1. Open the class in the Architect.
2. Delete every dimension, measure, listing field, and listing.

4 Defining Dimensions

This chapter describes how to create most kinds of dimensions. It discusses the following topics:

- An overview of the Dim tab
- How to define a dimension
- How to specify the value on which a dimension is based
- How to choose a dimension data type
- How to define a dimension based on a date
- How to define a dimension based on a time
- How to define a dimension based on a collection
- How to modify a dimension
- How to control whether a dimension is included in the subset that can be selectively rebuilt
- How to activate or deactivate a dimension
- How to delete a dimension
- How to make your changes visible in the Analyzer

Also see the chapters "Defining Special Dimensions" and "Using Other Tables to Define Dimensions, Measures, and Listing Fields."

For general guidelines, see the chapter "Core Modeling Principles."

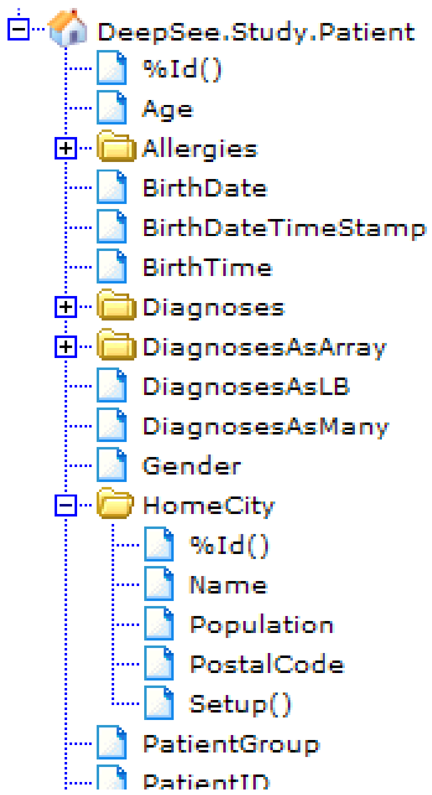
For a tutorial, see the *DeepSee Developer Tutorial*.

4.1 Overview of the Dim Tab

You use the **Dim** tab to add, modify, and delete dimensions. This tab has three main areas.

4.1.1 Class Hierarchy

The class hierarchy on the left shows the *base class* (the BI-enabled class) and most of the classes available from it via property relationships:



Here you can do the following:

- Expand a collapsed folder by clicking the Expand button ⊞ to its left.
- Collapse an expanded folder by clicking the Collapse button ⊟ to its left.
- Double-click a property to create a new dimension definition based on it.
- Drag and drop a property, instance method, or SQL shortcut to the **Dimensions** table on the right, to create a new dimension definition based on it.

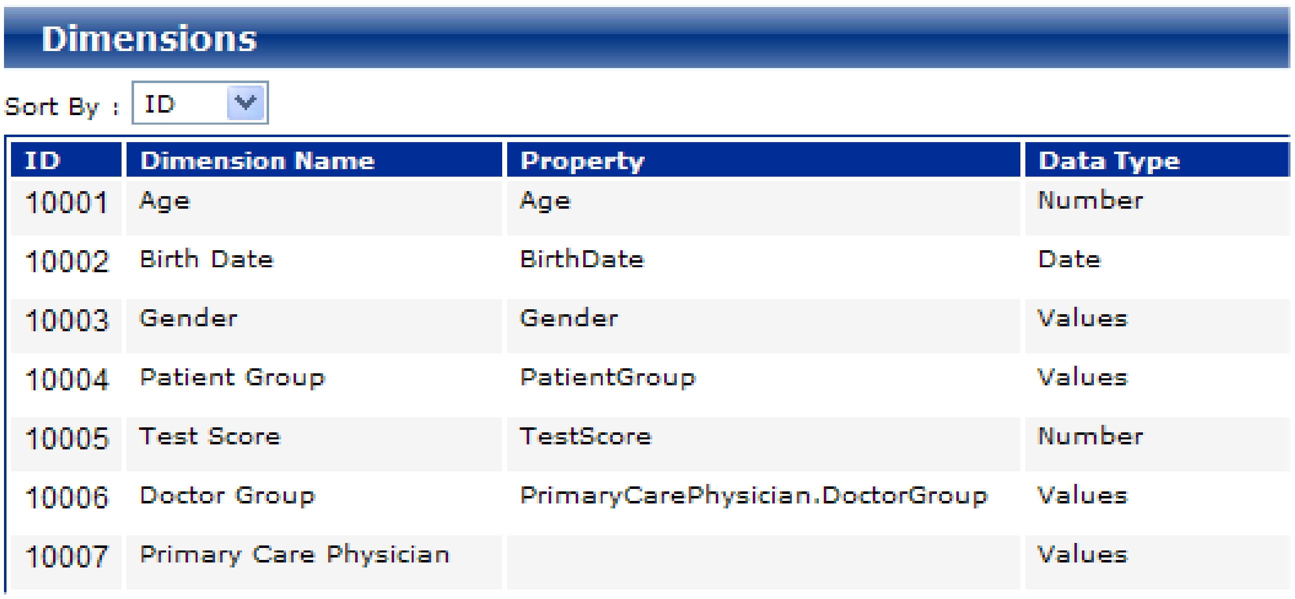
  Any instance methods are shown alphabetically after the properties.

Although this view provides a convenient way to access some properties, you can in fact access any property, and you can access data in any table by using **Complex Code**.

For details on what DeepSee displays in this hierarchy, see "Available Properties and Methods of a BI-enabled Class," in the previous chapter.

4.1.2 Dimensions Table

The **Dimensions** table on the right summarizes the currently defined dimensions:



ID	Dimension Name	Property	Data Type
10001	Age	Age	Number
10002	Birth Date	BirthDate	Date
10003	Gender	Gender	Values
10004	Patient Group	PatientGroup	Values
10005	Test Score	TestScore	Number
10006	Doctor Group	PrimaryCarePhysician.DoctorGroup	Values
10007	Primary Care Physician		Values

Each row corresponds to a dimension. For **Sort By**, click either **ID** or **Name**, to control how the rows are sorted. In this table:

- **Dimension Name** is the name of the dimension, as seen in the Analyzer and in any pivot tables that include this dimension. When you first create a dimension, this is the same as the name of the property, instance method, or SQL shortcut.

  Note: You can include a question mark (?) or a single quote (') in the name, but avoid other non-alphanumeric characters. In particular, do not use a period.

  Also, dimension names must be unique, and dimensions and measures cannot have the same name.

- **Property** is the class property or instance method on which the dimension is based.

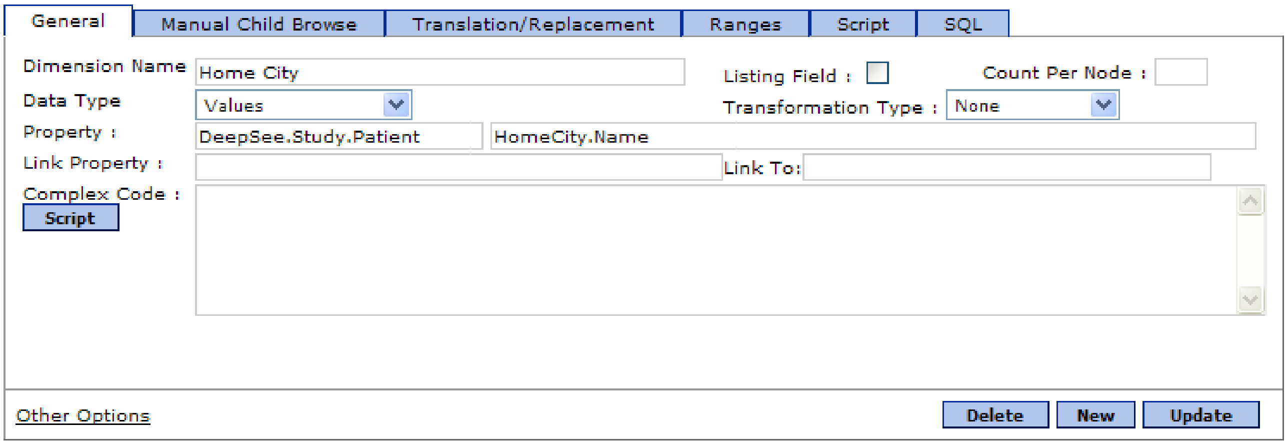
  This field is ignored if the dimension is based on complex code.

- **Data Type** affects how members are defined, as well as their default names. The data type is set automatically based on the data type of the field you are using, but you can change it. See "Specifying the Dimension Data Types."
- **Active** controls whether this dimension is active (visible in the Analyzer and in pivot tables). See "Deactivating or Activating a Dimension."
- **Complex** indicates whether this dimension is based on a Caché ObjectScript expression, instead of on a single class property. See "Basing a Dimension on a Caché ObjectScript Expression."
- **M.C.B.** indicates whether the **Manual Child Browse** tab contains information for this dimension. You use this tab to describe how to index the values in dimension that is based on a list-type property. See "Defining a Dimension Based on a Collection."
- **Trns/Rng** indicates whether you have entered information on the **Translation/Replacement** or **Range** tabs. See the chapter "Using Replacements, Ranges, and Transformations."
- **Listing** indicates whether this dimension definition also defines a listing field. See the chapter "Defining Listing Fields."
- **Select** controls whether this dimension is included in the subset that can be selectively rebuilt. See "Controlling Whether a Dimension Is Selectively Rebuilt."

In this table, when you click the dimension that you want to work on, the tabs in the lower right display details for that dimension.

4.1.3 Tabs

The tabs in the lower right display details about the currently selected dimension:



Dimension Name	Home City	Listing Field : ☐	Count Per Node :
Data Type	Values	Transformation Type :	None
Property :	DeepSee.Study.Patient	HomeCity.Name	
Link Property :		Link To:	
Complex Code :			

The following table summarizes the options on these tabs. Unless otherwise noted, the sections listed here are in this chapter.

Table 4–1: Options on the Dimension Tabs

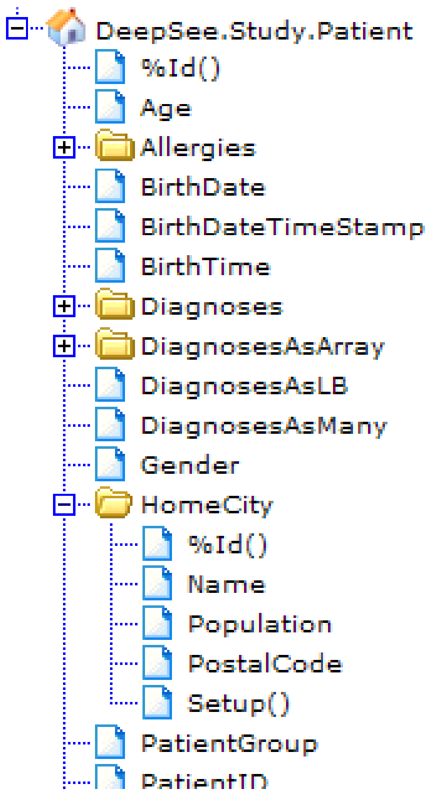
Option	Purpose
General tab: **Listing Field** option	Reuses this definition as a listing field as well. See the chapter "Defining Listing Fields."
General tab: **Count Per Node** option	Used only if this dimension is also used as a listing field. See the section "Creating a Dimension-type Listing Field," later in this book.
General tab: **Data Type** option	Controls how members are defined, as well as their default names. The data type is set automatically based on the data type of the field you are using, but you can change it. Several of the following sections discuss the types.
General tab: **Transformation Type** option	Applies a transformation that affects how the dimension members are defined and displayed. See the chapter "Using Replacements, Ranges, and Transformations." If you apply a transformation, do not use the **Translation/Replacement** and **Ranges** tabs.
General tab: **Property** options and **Complex Code** option	Specifies the data on which this dimension is based. See "Defining a Dimension."
General tab: **Link Property** and **Link To** options	Used to access data in an external class. See "Using the Link Options," later in this book.
Manual Child Browse	Applies only to dimensions based on list-type properties. This tab specifies how to index the values in the list. See "Defining a Dimension Based on a Collection."
Translation/Replacement tab	Defines a set of replacements that alter the names of the dimension members. See the chapter "Using Replacements, Ranges, and Transformations."

Option	Purpose
Ranges tab	Defines the dimension members as a set of ranges of values (applies to numeric-valued and time-valued dimensions). See the chapter "Using Replacements, Ranges, and Transformations."

4.2 Defining a Dimension

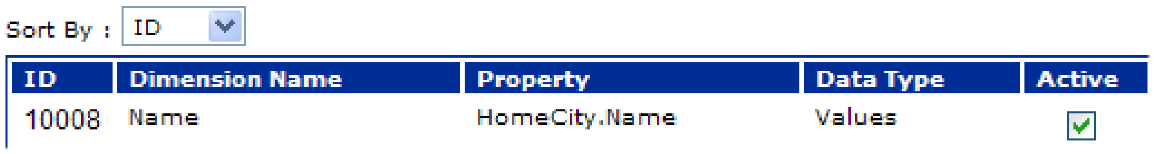
This section describes the basic method for defining dimensions:

1. Open a BI-enabled class as described in the previous chapter.
2. Click the **Dim** tab.
3. Expand the class hierarchy in the left area as needed.



4. Drag and drop a property, instance method, or SQL shortcut from the class hierarchy to the **Dimensions** table on the right. Or double-click the property, instance method, or SQL shortcut.

   In either case, the **Dimensions** table now includes a dimension that is based on your selection. For example:



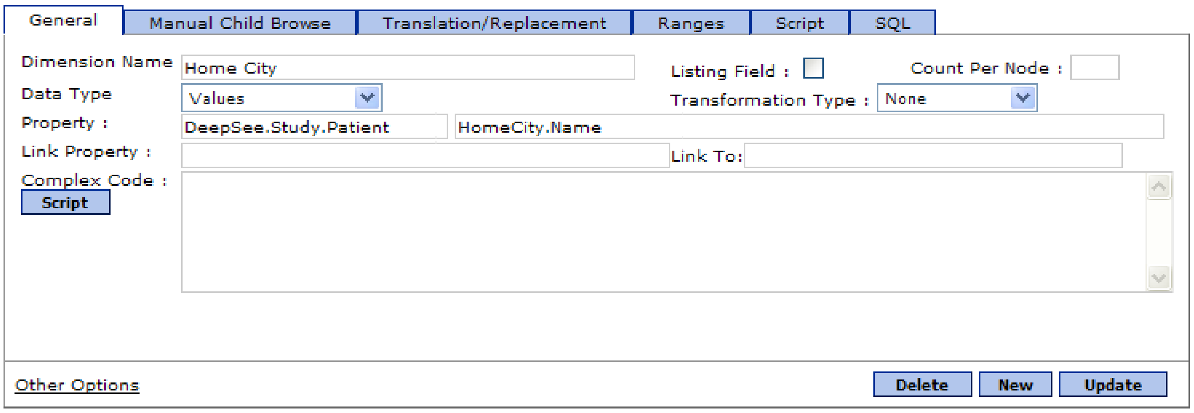
Sort By : ID

ID	Dimension Name	Property	Data Type	Active
10008	Name	HomeCity.Name	Values	✓

5. Click this new row so that it is selected. The bottom right area of the page then shows details for this dimension.

6. Optionally rename the dimension by changing the value in **Dimension Name** on the **General** tab. This name is shown in the Analyzer and is also used as the default column or row label in any pivot tables that include this dimension.

   The dimension name should make sense on its own; notice that it does not indicate the table that owns this property. In the example here, `Name` might be ambiguous, so you might rename this dimension to `Home City`. Also, the dimension name can use only alphanumeric characters.

   Dimension Name: Home City
   Listing Field :
   Count Per Node :
   Data Type: Values
   Transformation Type : None
   Property : DeepSee.Study.Patient HomeCity.Name
   Link Property :
   Link To:
   Complex Code :

   In the **General** tab, the editable two **Property** fields contain the following data:

   a. The first field, which is required, shows the package and class name of the BI-enabled class from which you started, in the form *package*.*class*. For example: `DeepSee.Study.Patient`

   b. The second field, which is optional, shows the name of the property or instance method on which this dimension is based, relative to this class. This is expressed in the form *property* or *property*.*subproperty* and so on, in exactly the same way you would refer to that property in Caché ObjectScript. For example: `HomeCity.Name`

      This field is optional because there are other ways to specify the data on which the dimension is based; see "Specifying the Data on Which a Dimension Is Based," later in this chapter. Also see the chapter "Using Other Tables to Define Dimensions, Measures, and Listing Fields."

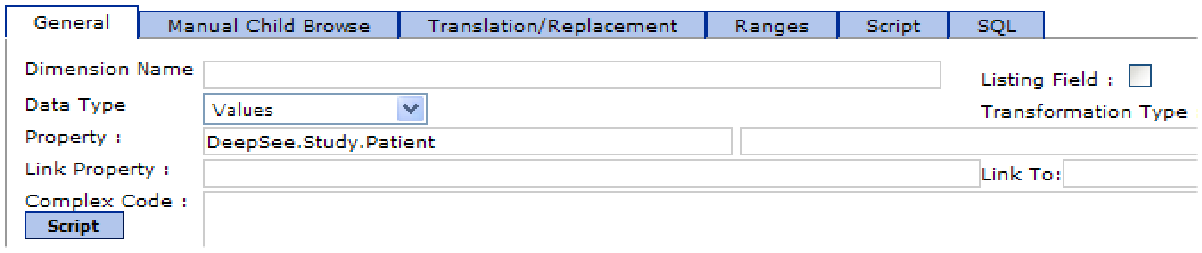
7. Click **Update** to save your changes.

4.2.1 Using the New Button

You can also create a dimension as follows:

1. Click **New** in the lower right.

   DeepSee initializes the screen as follows:

   Dimension Name
   Listing Field :
   Data Type: Values
   Transformation Type
   Property : DeepSee.Study.Patient
   Link Property :
   Link To:
   Complex Code :

2. Type a name into **Dimension Name**.

3. Optionally type a property name into the second **Property** field. For other options, see "Specifying the Data on Which a Dimension Is Based," later in this chapter.

4. Change other settings as needed.
5. Click **Add** in the lower right to add the dimension. DeepSee then adds the dimension to the **Dimensions** table.
6. Click the dimension in the table to reselect it.
7. Click the **Active** check box for the dimension and save it again.

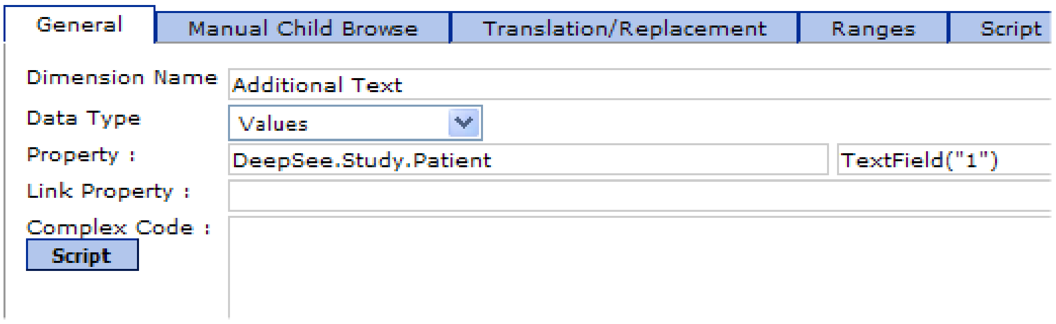
4.2.2 Using ChkField(), DateField(), and TextField()

The Architect provides options you can use to create dimensions of specific types (**Values**, **Boolean**, and **Date**) slightly more quickly than using the **New** button. To use these options:

1. Double-click **ChkField()**, **DateField()**, or **TextField()**.

   Or drag and drop this item to the **Dimensions** table.

   DeepSee initializes the screen as follows:

   Dimension Name: Additional Text
   Data Type: Values
   Property : DeepSee.Study.Patient TextField("1")
   Link Property :
   Complex Code :

   The example shown here applies to **TextField()**.

   For **ChkField()**, the initial dimension name is `Additional Boolean` and the initial value for **Data Type** is **Boolean.**

   For **DateField()**, the initial dimension name is `Additional Date` and the initial value for **Data Type** is **Date.**

   The second **Property** field is also initialized, but the initial name is probably is not suitable in any case. Ignore the suggested syntax presented in this field.
2. Edit the name of the dimension as needed.
3. Edit the property name in the second **Property** field. For other options, see the next section.
4. Change other settings as needed.
5. Click **Update** to save your changes.

4.3 Specifying the Data on Which a Dimension Is Based

There are three options for specifying the value on which the dimension is based:

- Base it directly on a single class property, as in the preceding section.
- Use **Complex Code** and base the dimension on a Caché ObjectScript expression, possibly using multiple class properties, as described in the following subsection.

  This option takes precedence over the preceding option.

- Use the **Manual Child Browse** option to generate a set of values, typically from a collection property. See "Defining a Dimension Based on a Collection," later in this chapter. When you do this, DeepSee generates an expression that returns multiple values; the dimension is based on these values.

  This option takes precedence over the preceding two options.

If you use **Complex Code** or **Manual Child Browse** option, DeepSee ignores the setting of **Null Field Replacement** (on the **Translation/Replacement** tab).

Each unique value (case-sensitive) of the property or expression corresponds to a separate member of this dimension.

4.3.1 Basing a Dimension on a Caché ObjectScript Expression

To redefine a dimension so that it is based on a Caché ObjectScript expression:

1. In the **Complex Code** field, type a Caché ObjectScript expression that returns a different value for each dimension member. The expression can use the variable *%this*, which refers an instance of the BI-enabled class you started from.

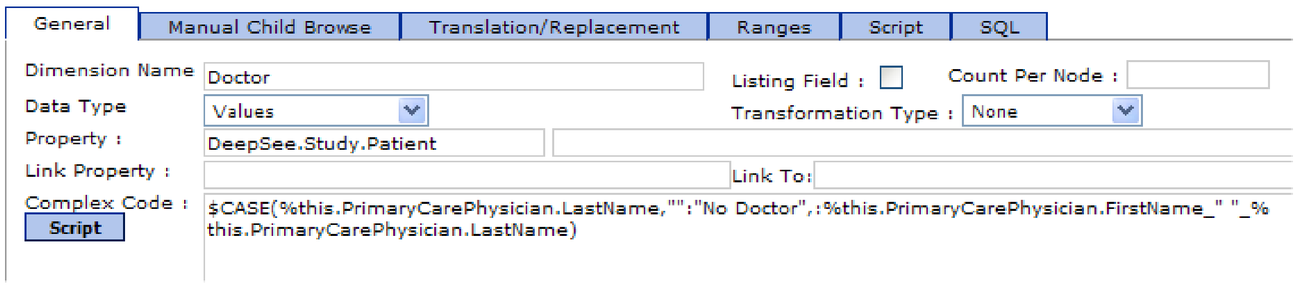
   Or click **Script**, which displays an editor where you can build an expression from existing dimensions, data fields, and Caché ObjectScript functions.

   You can use any Caché ObjectScript expression, which means that you can also invoke methods or routines in this namespace.

2. Delete the contents of the second **Property** field.

   This step is optional, because this field is ignored if there is data in **Complex Code**.

Consider the following example:



Dimension Name Doctor
Listing Field :
Count Per Node :
Data Type Values
Transformation Type : None
Property : DeepSee.Study.Patient
Link Property :
Link To:
Complex Code : $CASE(%this.PrimaryCarePhysician.LastName,"":"No Doctor",:%this.PrimaryCarePhysician.FirstName_" "_%this.PrimaryCarePhysician.LastName)

This dimension (`Primary Care Physician`) groups the source records into buckets based on the following string, which is based on two properties:

```
$CASE(%this.PrimaryCarePhysician.LastName,"":"No Doctor",
:%this.PrimaryCarePhysician.FirstName_" "_%this.PrimaryCarePhysician.LastName)
```

Each first name/last name combination that is present in the data receives a separate bucket. Each member of this dimension corresponds to one of these buckets. If a patient has no recorded primary care physician, the patient is assigned to the `No Doctor` member.

Also notice that the **Complex** check box is checked in the table at the top of the page; this happens automatically when you enter data into the **Complex Code** field.

When you use **Complex Code**, DeepSee ignores the setting of **Null Field Replacement** (on the **Translation/Replacement** tab). This means that if you want to replace null values with a string such as `"No Assigned Staff"`, you must handle this yourself within **Complex Code**.

4.4 Specifying the Dimension Data Type

The **Data Type** option of a dimension affects how members are defined. The following table summarizes the options.

Table 4–2: Data Types for Dimensions

Dimension Data Type	Purpose and Comments
Values	Creates a dimension that has a member for each unique value of the dimension data. By default, these are collated as strings. Can be used with any type of data. This is the most commonly used option.
Reference	Creates a dimension that has a member for each unique internal identifier. Use this type of dimension if you expect the dimension data values to change and if you do not want to rebuild the indices when that occurs. For example, if you use the customer name as a dimension, you might use the **Reference** type for this dimension. Then if a customer company changes its name, it would not be necessary to rebuild the indices to accommodate that change. Can be used with any type of data. When a dimension is defined this way, however, it cannot also be used as a listing field.
Date	See the section "Defining a Dimension Based on a Date," later in this chapter.
Number	Creates a series object dimension and a measure. Appropriate only for numeric data. Note that you can use **Null Field Replacement** with this type, as you can with the other types.
Boolean	Creates a dimension that has two members: • `No` groups records for which the dimension data equals 0, `"N"`, `"NO"`, `"F"`, or `"FALSE"` (not case-sensitive). • `Yes` groups records for which the dimension data equals 1, `"Y"`, `"YES"`, `"T"`, or `"TRUE"` (not case-sensitive).
Words	Treats the property as a space-separated collection of words and creates a dimension for each word. DeepSee considers case; for example, it treats `AllergyCode1` and `allergycode1` as different words. See "Defining a Dimension Based on a Collection," later in this chapter.
Date and format	See the section "Defining a Dimension Based on a Date," later in this chapter.
Time	Use this only if you also specify a **Range Dimension Suffix.** Note that you can also use **Values** to create dimensions based on time. See the section "Defining a Dimension Based on a Time," later in this chapter.

In this table, the phrase "dimension data" refers to the data on which the dimension is based (whether that is a single field or an expression).

The drop-down list also includes any custom types that have been defined.

4.4.1 Creating a Custom Data Type

If the standard types do not meet your needs, you can create and use custom data types, as follows:

1. At the top of the Architect page, click **Data Modeler > Abstract Data Type**.

   The **Date Type Name** list shows any custom types that have been defined. The right area shows the details for the currently selected type, if any.

2. Type a name into **Type Name**.

3. Click the **Base Type** drop-down list and then choose a type upon which to base the new type.

4. Specify details for this type. The details depend upon the base type.

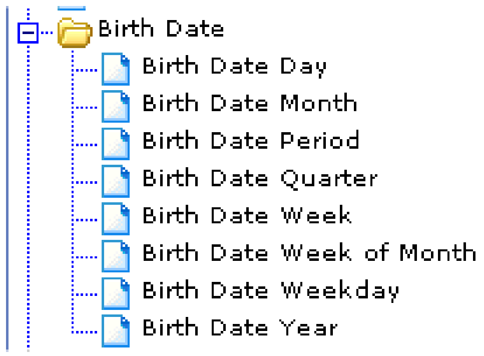
5. Click **Add**.

For example, the standard type **Values** uses string collation, which is often unsuitable for numeric data. For example, if you create a dimension based on age, you would usually prefer to sort the ages in numeric order rather than in string order. For such a dimension, you would create a custom type based on **Values**, but you would set **Collating Sequence** equal to **Numeric**. You would use this type as the type for your dimension.

Listing field definitions, described later in this book, are similar in many ways to dimension definitions. However, custom data types are available only for dimensions.

4.5 Defining a Dimension Based on a Date

Date values are special; when you create a dimension that uses a date, DeepSee automatically generates a set of dimension variations that group data by weekday, week, month, quarter, and so on. For example, if you define the `Birth Date` dimension, DeepSee creates the following dimensions (as seen in the Analyzer):



DeepSee automatically recognizes dates of type %Date and %TimeStamp, but also accepts several string forms, and you can convert from any other format. This section describes how to use date-valued properties as dimensions and gives details on the resulting dimension variations.

4.5.1 Recognized Date Formats

If you start with a property of type %Date or %TimeStamp and you create a dimension based on the property, then DeepSee automatically sets **Data Type** equal to **Date**, the default date format. If the property is a string that contains date values, you must set **Data Type** appropriately as follows:

Property Type	Format of Date Property, if String Type*	Appropriate Setting for Data Type of Dimension
%Date		**Date**
%TimeStamp		**Date** or **Date YYYY-MM-DD**
%String	*YYYY-MM-DD* with optional trailing time data	**Date** or **Date YYYY-MM-DD**
	YYYY/MM/DD with optional trailing time data	**Date YYYY/MM/DD**
	DD-MM-YYYY with optional trailing time data	**Date DD-MM-YYYY**
	DD/MM/YYYY with optional trailing time data	**Date DD/MM/YYYY**

*Here, *DD* is a two-digit day, *MM* is a two-digit month, and *YYYY* is a four-digit year.

The logical value for a property of type %TimeStamp includes the time, which the predefined dimension data types ignore. For a property of type %String, if the date piece is followed by a space and then a time piece, the time piece is ignored.

4.5.2 Converting Other Date Formats

If the date dimension is not in one of the preceding formats, you must use the **Complex Code** option to convert its format.

For example, suppose the property BirthDateVar5 is in the format *YYYYMMDD*, where *YYYY* is a four-digit year, *MM* is a two-digit month, and *DD* is a two-digit day. To use this property as a date dimension, you could do the following:

- Specify the following for **Complex Code**:

  ```
  $ZDATEH(%this.BirthDateVar5,8)
  ```

- Set **Data Type** equal to **Date**.

For information on the **$ZDATEH** function, see the book *Caché ObjectScript Reference*.

4.5.3 Date Dimension Details

This section describes the behavior of the dimension variations that DeepSee automatically generates, using the `Birth Date` dimension as an example:

- The `Birth Date Day` dimension has one member for each day of the month, 1 through 31. Any given member uses data for the corresponding day of the month, ignoring the year and month used in the data. For example, the member 2 uses data for all records where the birth date is the second day of the month, for any month and year.
- The `Birth Date Month` dimension has one member for month, January through December. Any given member uses data for the corresponding month. For example, the member March uses data for all records where the birth date is in March, for any year.
- The `Birth Date Period` dimension has one member for year-month combination. For example, the member 2005-11 uses data for all records where the birth date is in November 2005.
- The `Birth Date Quarter` dimension has four members: Q1, Q2, Q3, and Q4. Any given member uses data for the corresponding quarter. For example, the member Q1 uses data for all records where the birth date is in Q1, for any year.
- The `Birth Date Week` dimension has one member for each week number of the year. Any given member uses data for the corresponding week. For example, the member 7 uses data for all records where the birth date is in the seventh week of the year, for any year.

- The `Birth Date Week of Month` dimension has one member for week number of the month. Any given member uses data for the corresponding week of the month. For example, the member 3 uses data for all records where the birth date is in the third week of the month, for any month and year.
- The `Birth Date Weekday` dimension has one member for each day of the week, Sunday through Saturday. Any given member uses data for the corresponding weekday, ignoring the year and month used in the data. For example, the member Wednesday uses data for all records where the birth date is on a Wednesday, for any month and year.
- The `Birth Date Year` dimension has one member for each year.

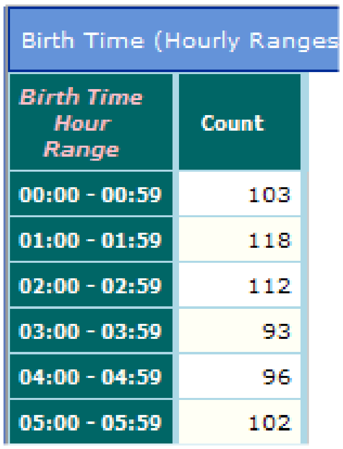
4.6 Defining a Dimension Based on a Time

You can define dimensions based on time data. Typically you would do this to look at periodicity over the course of the day, and you would group data by hour or perhaps into larger buckets. This section describes three possible techniques:

- Creating hourly buckets
- Creating buckets by parsing the time as a string
- Creating custom ranges

4.6.1 Creating Hourly Buckets

If you use the **Time** dimension data type and specify a **Range Dimension Suffix**, DeepSee generates a set of hourly buckets as follows:



Birth Time (Hourly Ranges

Birth Time Hour Range	Count
00:00 - 00:59	103
01:00 - 01:59	118
02:00 - 02:59	112
03:00 - 03:59	93
04:00 - 04:59	96
05:00 - 05:59	102

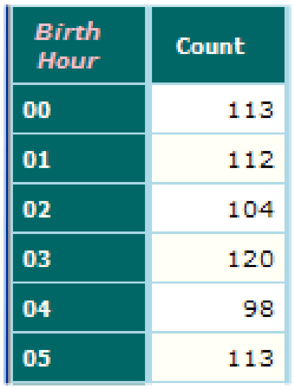
To create a dimension with hourly buckets as shown in this example:

1. Set **Data Type** to **Time**.

   For this dimension data type, DeepSee correctly handles properties of type %TimeStamp, %Time, or even a string with the format *hh:mm:ss* (where *hh* is the hour, *mm* is the minutes, and *ss* is the seconds). You do not need to specify any conversion of format.
2. Click the **Ranges** tab.
3. For **Range Dimension Suffix**, type a string such as `Hours`.
4. Click **Update** to save your changes.

4.6.2 Creating Time Buckets by Parsing the Time as a String

You can also create the dimension data as string data and pick out the part to use as the dimension. For example:



Birth Hour	Count
00	113
01	112
02	104
03	120
04	98
05	113

To create such a dimension:

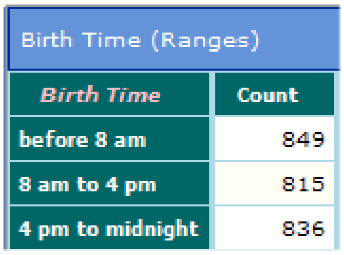
1. Set **Data Type** to **Values**.
2. For **Complex Code**, use an expression that extracts the part that you want to use as the dimension. For example, the following extracts the hour piece from a date in ODBC format:

   ```
   $PIECE($PIECE(%this.BirthDateTimeStamp," ",2),":",1)
   ```

3. Click **Update** to save your changes.

4.6.3 Defining Time Ranges

You can also create custom time ranges as in the following example:



Birth Time (Ranges)

Birth Time	Count
before 8 am	849
8 am to 4 pm	815
4 pm to midnight	836

To create custom time ranges:

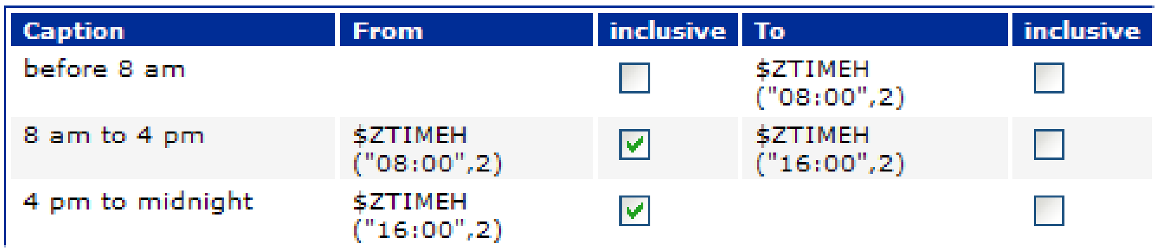
1. Set **Data Type** to **Values**.
2. Optionally use **Complex Code** to convert the data to a format that is easier to work with. For example, suppose that the property BirthDate contains dates in ODBC format (*YYYY-MM-DD hh:mm:ss*, where *YYYY* is a four-digit year, *MM* is a two-digit month, *DD* is a two-digit day, *hh* is the hour, *mm* is the minutes, and *ss* is the seconds).

   To extract from this property the number of seconds since midnight, you would use the following for **Complex Code**:

   ```
   $ZTIMEH($PIECE(%this.BirthDate," ",2),3)
   ```

   DeepSee evaluates this expression before comparing to any ranges.

3. Click the **Ranges** tab and define ranges. You can use Caché ObjectScript expressions for the **From** and **To** values. For example:



Caption	From	inclusive	To	inclusive
before 8 am		☐	$ZTIMEH ("08:00",2)	☐
8 am to 4 pm	$ZTIMEH ("08:00",2)	✓	$ZTIMEH ("16:00",2)	☐
4 pm to midnight	$ZTIMEH ("16:00",2)	✓		☐

The dimension has one member for each range that you define.

4. Click **Update** to save your changes.

Note: By default, the members of this dimension are sorted alphabetically, but when you use this dimension in a given pivot table, you can easily modify the sort order; see the chapter "Creating and Customizing Dimensions" in the book *Using the DeepSee Analyzer*.

4.7 Defining a Dimension Based on a Collection

In many cases, the BI-enabled class includes properties that contain collections. You can define dimensions based on collections. This section describes the following:

- How to use the Words dimension data type
- How to use the Words dimension data type with custom code
- How to use the M.C.B. feature with a list property
- How to use the M.C.B. feature with an array property
- How to force M.C.B. to return at least one value (null handling)
- How to interpret such dimensions

4.7.1 Using the Words Dimension Data Type

The simplest kind of collection is a space-separated list. The **Words** dimension data type removes non-alphanumeric characters (other than spaces) from the dimension data, parses the data into space-delimited words, and creates a member for each unique word. DeepSee considers case; for example, it treats `AllergyCode1` and `allergycode1` as different words. For example, suppose that your BI-enabled class included the property FavoriteColors, which might have values like the following:

- `Red`
- `Green Red`
- `Blue`
- `Pink Red`

If you use this property as a dimension and you set **Data Type** to **Words**, you create the members `Red`, `Green`, `Blue`, and `Pink`.

4.7.2 Using the Words Dimension Data Type with Complex Code

You may be able to convert a collection property to a space-separated list so that you can use the **Words** dimension data type. You might find it convenient to create class methods or routines to perform the conversion.

For example, suppose that your BI-enabled class represents patients and that it has a property called Diagnoses, which is an array. To create a dimension based on diagnoses, you could do the following:

1. Click the **Dim** tab.
2. Click **New** in the lower right area.

   DeepSee automatically sets **Property** to the name of the BI-enabled class, and you should not change this.
3. For **Dimension Name**, type the dimension name.
4. For **Data Type**, click **Words**.
5. For **Complex Code**, type an expression that returns a space-separated list. For example:

   ```
   ##class(Dev.Utils).ArrayToString(%this.DiagnosesAsArray)
   ```

   This class method iterates through the array, extracts the values, and returns a string consisting of a space-separated list of the values.
6. Click **Add** in the lower right area.

The class method used here is defined as follows:

```
ClassMethod ArrayToString(input, delimiter As %String)
{
   If '$Data(delimiter) set delimiter=" "
   set string=""
   Set i=""
   for
      {
      Set i=input.Next(i)
      Quit:(i="")
      If i=1 {Set string=input.GetAt(i)}
      Else {Set string=string_delimiter_input.GetAt(i)}
      }

    If string="" Set string="None"
    Quit string
}
```

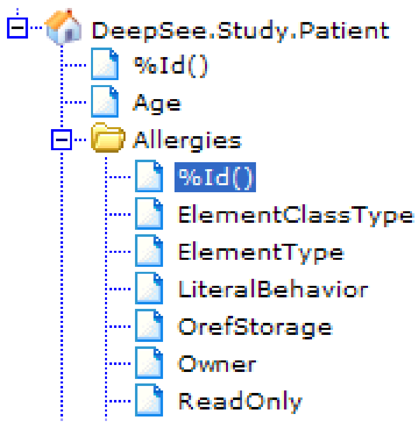
This technique is not suitable if the resulting string is longer than 32 KB (unless long strings are enabled). Also remember that non-alphabetic characters are stripped from the string and are not used in the member names.

4.7.3 Using the M.C.B. Feature with a List-type Property

To see how to use the **M. C. B. (Manual Child Browse)** feature, it is best to consider an example. The DeepSee.Study.Patient class contains the Allergies property, which is defined as follows:

```
Property Allergies As list Of DeepSee.Study.PatientAllergy;
```

Within the Architect, this property is shown as a folder:



For this example, to use this list property as a dimension:

1. Drag and drop the entire property folder from the **Class** list to the **Dimensions** table. This adds a dimension whose default name is the folder name.

2. Set **Data Type** to **Values**.

3. Click the **Manual Child Browse** tab. Here you specify how to iterate through the elements of the list and how to get the dimension data that corresponds to each element.

   You can access all elements of the list or a subset according to your own logic.

4. For **Loop Coding**, specify a fragment of Caché ObjectScript code that iterates through the list (or click **Sample** to insert an editable sample). For example:

   ```
   for i=1:1:%this.Allergies.Count()  do
   ```

   Important: You must include two spaces before **Do** and one space after **Do**. There must not be any empty lines after that.

   To refer to the list property, use the syntax *%this.PropertyName* as shown.

   You can include complete statements before the fragment, but do not include manual line breaks. (That is, type on a single line within **Loop Coding**. The text wraps automatically so that you can see all of it.)

5. For **Each Fetch**, specify one or more Caché ObjectScript statements that assign a value to the variable *val* (or click **Sample** to insert an editable sample). For example:

   ```
   set val=%this.Allergies.GetAt(i).Allergen.Description
   ```

   Your code can refer to the variables used in **Loop Coding**. If you use multiple statements, do not include manual line breaks. (That is, type on a single line within **Each Fetch**. The text wraps automatically so that you can see all of it.)

6. Click the **General** tab and then delete the contents of the second **Property** field.

   This step is optional, because this field is ignored if there is data on the **Manual Child Browse** tab.

Notice that in the **Dimensions** table, the **M.C.B.** check box becomes selected automatically for this dimension. This indicates that this dimension has information specified on the **Manual Child Browse** tab.

4.7.4 Using the M.C.B. Feature with an Array-type Property

For an array-type property, you would use a similar technique.

For **Loop Coding**, you would use something like the following:

```
Set wI="" For  Set wI=%this.Categories.Next(wI) Quit:wI=""  Do
```

Important: You must include two spaces before **Set**. Also, there must be two spaces before **Do** and one space after **Do**. There must not be any empty lines after that.

To refer to the array property, use the syntax **%this.***PropertyName* as shown.

For **Each Fetch**, you would use something like the following:

```
Set val=%this.Categories.GetAt(wI)
```

4.7.5 Forcing M.C.B. to Return at Least One Value

You can write the **Loop Coding** and **Each Fetch** options so that DeepSee returns at least one value. For example, a patient might not have any allergies. In this case, you might prefer for the **M.C.B.** feature to return the string `No Allergies`. If Allergies is a list, you could use the following for **Loop Coding**:

```
Set key="" Set count=%this.Allergies.Count() Set count=$Case(count,0:1,:count) For i=1:1:count DO
```

And use the following for **Each Fetch**:

```
Set allergy=%this.Allergies.GetNext(.key) Set val=$Case(allergy,"":"No
Allergies",:allergy.Allergen.Description)
```

For alternative approaches, see the *DeepSee Developer Tutorial*.

4.7.6 Understanding Dimensions Based on Collections

When you use dimensions that are based on collections, it is important to remember that, as always, the items being counted are the items in the base class, not the items of the collection.

For example, a company has multiple employees, and each employee has a single home town. Suppose that your base class is Company, and suppose that you use **Manual Child Browse** to create a dimension called Employee Home Town.

If you performed a breakout by Employee Home Town, you might see a pivot table like the following:

Employee Home Town	Count
Beverly	4
Cambridge	59
Dedham	16
...	...

Because Company is the base class, this pivot table is counting companies. The row for Beverly indicates that 4 companies have at least one employee whose home town is Beverly. It does *not* indicate the number of employees who live in Beverly.

It is also important to remember that, as always, the items being correlated are the items in the base class, not the items of the collection. Let us expand the previous example by adding the Employee Gender dimension. If we add this dimension to the preceding pivot table, we might see this:

	Female	Male
Employee Home Town	**Count**	**Count**
Beverly	2	4
Cambridge	34	30
Dedham	8	7
...	...	

Let us examine the cells in the Beverly row. Notice that the count for Female and the count for Male combine to more than the count for Beverly in the preceding example.

The Beverly, Female=2 cell means that there are two companies that have at least one employee living in Beverly and at least one female employee. There is no guarantee that these employees are the same people. If Teradyne Corp. has 1296 male employees who live in Beverly and one female employee who lives on Martha's Vineyard, that company is counted once for the Beverly, Female cell.

Tip: As you can see, the default measure name Count can be misleadingly generic. When you use this measure in a pivot table, it is good practice to rename it as appropriate for your base class (for example, Company Count or Companies). See "Defining Measures" in the book *Using the DeepSee Analyzer*.

4.8 Modifying a Dimension

To modify the definition of a dimension:

1. Open a BI-enabled class as described in the previous chapter.
2. Click the **Dim** tab.
3. Click the dimension in the **Dimensions** table.
4. Make the changes.
5. Click **Update** to save your changes.

4.9 Controlling Whether a Dimension Is Selectively Rebuilt

The Architect provides an option to rebuild only a subset of the dimensions. Within the definition of each dimension, you specify whether that dimension is included within the selective rebuild. To do so:

1. Open a BI-enabled class as described in the previous chapter.
2. Click the **Dim** tab.
3. To enable this dimension to be rebuilt selectively, click the **Select** check box for the dimension in the **Dimensions** table.
4. Click **Update** to save your changes.

See "Rebuilding Selectively," earlier in this book.

4.10 Activating or Deactivating a Dimension

When you create a dimension, it is automatically activated. You can deactivate it so that it is no longer available in the Analyzer.

To activate or deactivate a dimension:

1. Open a BI-enabled class as described in the previous chapter.
2. Click the **Dim** tab.
3. Click the dimension in the **Dimensions** table.
4. To deactivate the dimension, clear the **Active** check box on the **General** tab.

   Or to activate the dimension, click the **Active** check box.
5. Click **Update** to save your changes.

4.11 Deleting a Dimension

To delete a dimension, do the following:

1. Open a BI-enabled class as described in the previous chapter.
2. Click the **Dim** tab or **Compound Mem** tab, as appropriate.

   (If you create a dimension on the **Compound Mem** tab, that is where you can delete it. For information on the purpose of this tab, see the chapter "Defining Compound Members.")
3. Click the dimension in the **Dimensions** table.
4. Click **Delete**. You are prompted to confirm this action.

4.12 Next Steps

After you define, modify, or delete a dimension, do the following to make your changes visible in the Analyzer:

1. Recompile and rebuild, as described in "Recompiling and Rebuilding" earlier in this book.
2. Define a subject area, as described later in this book.

   If you have added a dimension, make sure the subject area includes the dimension.

   When you define the subject area, make sure to give access to all suitable roles.
3. If the subject area is already open in the Analyzer, you may need to reopen it.

5
Defining Dynamic Dimensions

This chapter describes how to define dynamic dimensions. It discusses the following topics:

- Overview of dynamic dimensions
- How to create dynamic dimensions for use in filters
- A basic demonstration
- How to make your changes visible in the Analyzer

Also see the chapter "Defining Dimensions."

5.1 Overview of Dynamic Dimensions

A dynamic dimension is a special kind of dimension that can be used only in filter expressions. At runtime, it converts the given filter value to the ID or IDs of the records to use.

This type of dimension works as follows:

- The dimension can have any name, as long as that name follows the normal naming rules and does not conflict with an existing dimension.
- The dimension is defined by Caché ObjectScript statements.
- These statements specify the ID or IDs of the records to use:
  - To specify a single record of the base table, set the variable *val* equal to the ID of the record.
  - To specify multiple records of the base table, set nodes of the multidimensional array variable *OLIST*.
- These statements can use the variable *PARAM*, which contains the value specified on the right side of a filter expression that uses this dimension. This value can either be hardcoded or can be chosen by the user at runtime.

Dynamic dimensions cannot be used for rows or columns.

5.2 Creating a Dynamic Dimension

To define a dynamic dimension:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Dyn Dim** tab.

   The tab lists any dynamic dimensions that are already defined.
3. Right-click and then click **Add Line**.

   DeepSee adds a new, blank line to the table in this tab.
4. For **Dimension Name**, type the name of the dimension. For example: `MyDim`
5. For **Script**, type one or more Caché ObjectScript statements that specify the records of the base table to use. You typically call your custom routine or class method. In your custom code, do the following:
   - Optionally use the variable named *PARAM*, which will contain the value specified by a filter that uses this dimension.
   - To use one record of the base table, set the variable *val* equal to an ID of that record.
   - To use multiple records of the base table, set *OLIST* equal to the number of records. Then set the nodes *OLIST(n)* where *n* is an integer. Set these nodes equal to the IDs of the desired records.

     For example, to use three records:

     ```
     Set OLIST=3
     Set OLIST(1)=id1
     Set OLIST(2)=id2
     Set OLIST(3)=id3
     ```

     Where *objectid1*, *objectid2*, and *objectid3* are IDs of the three records.
   - If no ID is appropriate, set *val* or *OLIST* equal to null.
6. Click **Save**.

5.3 Basic Demonstration

As a simple example, suppose that we want a dynamic dimension that behaves as follows:

- If the user selects `A`, DeepSee should access the record with ID equal to 1.
- If the user selects `B`, DeepSee should access the records with IDs equal to 2, 3, and 4.
- If the user selects `C`, DeepSee should access no records.

To see the basic capability, do the following using the DeepSee.Study sample:

1. In Studio, define a routine, `demodyndim`, as follows:

   ```
    if PARAM="A" {set val=1}
    if PARAM="B" {set OLIST=3,OLIST("1")=2,OLIST("2")=3,OLIST("3")=4}
    if PARAM="C" {set val=""}
   ```
2. Define a dynamic dimension with the following details:
   - **Dimension Name** — `DemoDynDim`
   - **Script** — `Do ^demodyndim`
3. Compile the DeepSee.Study.Patient class.

4. In the Analyzer, create a pivot table that uses this class. Apply the following filter expression:

   ```
   [DemoDynDim = A]
   ```

   Run the pivot table and display a detail listing. You should see that only this pivot table uses only one record, the first patient.

   Try again with the following filter expression:

   ```
   [DemoDynDim = B]
   ```

   In this case, the pivot table should use three records.

   Last, try the following filter expression:

   ```
   [DemoDynDim = C]
   ```

   In this case, the pivot table should be empty.

5.4 Next Steps

After you define, modify, or delete a dynamic dimension, do the following to make your changes visible in the Analyzer:

1. Recompile, as described in "Recompiling and Rebuilding" earlier in this book.
2. Define a subject area, as described later in this book.

   If you have added a dimension, make sure the subject area includes those dimensions.

   When you define the subject area, make sure to give access to all suitable roles.
3. If the subject area is already open in the Analyzer, you may need to reopen it.

6
Defining Compound Members

This chapter describes how to define compound members in the Architect. It discusses the following topics

- Overview of compound members
- How to create compound members based on filter expressions
- How to make your changes visible in the Analyzer

Also see the chapter "Defining Dimensions."

You can also create compound members in the Analyzer; see the book *Using the DeepSee Analyzer*.

6.1 Overview of Compound Members

A compound member is defined by a DeepSee filter expression. When you create a compound member, you assign it to a dimension, which can be an existing dimension or a new one.

You use compound members in two different cases:

- If you want to group data at a higher level than is directly supported by the object model. In such a case, you can define a dimension whose members are groups of lower-level data. For example, you can define the continent dimension. The North American member would be defined by a filter that selected the North American countries. For example:

  ```
  [Country IN Canada,Mexico,USA]
  ```

- If you want to define a dimension whose members are determined by complex logic. For example, one member might be defined as follows:

  ```
  [Item Group = ABC] AND [Item Status IN Planned,Executed]
  ```

In either case, you define each member separately.

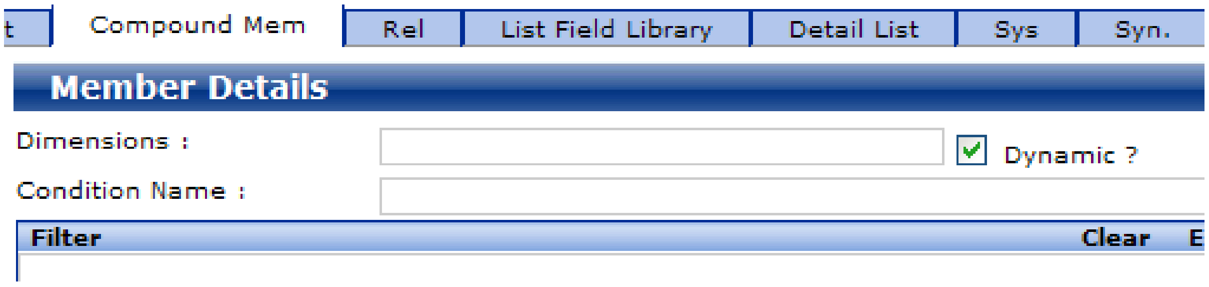
A compound member can be indexed (and thus available for use as columns or rows), but does not have to be.

6.2 Creating a Compound Member

You use the **Compound Mem** tab to define compound members. To define a compound member:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Compound Mem** tab.

   The left side of the page lists any dimensions that contain compound members, along with their members.
3. Click **New** in the lower right, to ensure that any existing data is cleared.



4. For **Dimensions**, type the name of the dimension that owns this compound member. For example: `Color Groups`
5. Optionally clear the **Dynamic?** check box if you want this member to be indexed (dynamic members are not indexed).
6. For **Condition Name**, type the name of a member. For example: `Primary Colors`
7. In the **Filter** box, type a filter expression that selects the data for this member. For example:

```
[Favorite Color = Blue] OR [Favorite Color = Red] OR [Favorite Color = Yellow]
```

   Or click the **Edit** option and define the filter expression interactively. For syntax and other details, see the book *Expressions and Scripts in DeepSee*.

   Note: The **Edit Filter** dialog box does not let you enter a filter expression until you have defined at least one subject area. (Before that, the dialog box is displayed as a blank box.) Also, it shows only the dimensions whose definitions were active the last time the DeepSee indices were built.

8. Click **Add**.

   Repeat as needed until you have defined all members.

6.3 Next Steps

After you define, modify, or delete a compound member, do the following to make your changes visible in the Analyzer:

1. If the member is index, rebuild, as described in "Recompiling and Rebuilding" earlier in this book.
2. Define a subject area, as described later in this book.

   Make sure the subject area includes the dimension that contains this member.

   When you define the subject area, make sure to give access to all suitable roles.
3. If the subject area is already open in the Analyzer, you may need to reopen it.

7

Defining Relationships between Dimensions

Where appropriate, you use the **Rel** tab to define relationships between dimensions, which improves performance when you use the dimensions together.

For example, suppose that you create one dimension based on a unique item code and another dimension based on the corresponding unique item description. In this case, you should also define a one-to-one relationship between these dimensions.

This chapter describes how to define relationships between dimensions.

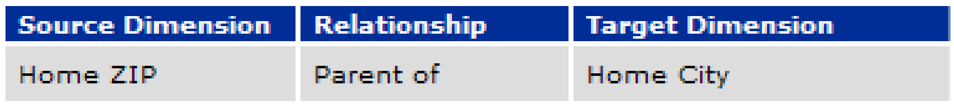
7.1 Defining a Relationship

To define a relationship between dimensions, do the following:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Rel** tab.

   The tab lists any relationships that are already defined.
3. Right-click and then click **Add Line**.
4. Double-click in the row, within the column for **Source Dimension**. DeepSee then displays a drop-down list of the dimensions (not including any dynamic dimensions).
5. Click a dimension in the column for **Source Dimension**.
6. Double-click in the row, within the column for **Relationship**. DeepSee then displays a drop-down list of the available types of relationships: **Parent of**, **One to One**, and **Many to Many**.
7. Click a relationship type.
8. Double-click in the row, within the column for **Target Dimension**. DeepSee then displays a drop-down list of the dimensions (not including any dynamic dimensions).
9. Click a dimension in the column for **Target Dimension**.
10. Click **Save**.

For example, the following relationship states that the `Home ZIP` dimension is the parent of the `Home City` dimension:



Source Dimension	Relationship	Target Dimension
Home ZIP	Parent of	Home City

7.2 Next Steps

After you define, modify, or delete a relationship, do the following to make your changes visible in the Analyzer:

1. Recompile and rebuild, as described in "Recompiling and Rebuilding" earlier in this book.
2. Define a subject area, as described later in this book.

   When you define the subject area, make sure to give access to all suitable roles.
3. If the subject area is already open in the Analyzer, you may need to reopen it.

8

Defining Measures

This chapter describes how to define measures in the Architect. You can define additional measures within the Analyzer, using these measures as building blocks.

8.1 About Measure Definitions

For this discussion, it is useful to distinguish two kinds of measures: *base measures* and *calculated measures*.

- Base measures are defined in the Architect. The formula for a base measure has the following form: `[MeasureName]`

  Every subject area includes the default measure, `Count`, which counts the records in the base class. Each base measure is either numeric or date-typed. Other than `Count`, each base measure is based on values in the raw data, is calculated at index build time, and stored in the fact table.

  When you define a numeric or date dimension in the Architect, DeepSee creates a base measure that uses that data. Also see "Defining Custom Aggregates for Use as Base Measures," later in this book.

  You can use `Count` in pivot tables. You can also use any numeric base measure in pivot tables; the values are summed. You cannot use date-type base measures in pivot tables, but they are available for use in calculated measures.

- Calculated measures are defined in the Analyzer. The formula for a calculated measure can include references to base measures, and it can include information on how to aggregate values, for example: `[Test Score.Average]`

  The values for calculated measures can be numbers, dates, or strings. Numeric values are summed by default. There is no default aggregation for dates or strings.

  Calculated measures are calculated at runtime, and they are not stored in the fact table.

  It is good practice to create a calculated measure based on the default measure but with a more informative name. For example, for the `Patients` subject area, you could create a calculated measure named `Patient Count` whose definition is `[Count]`.

8.2 Defining a Base Measure

To define a base measure, define a numeric or date dimension as described in "Defining a Dimension: The Basics." When you do so, make sure of the following:

- Ensure that the measure value is one-to-one with the records in the base class. For an explanation, see the For general guidelines, see the chapter “Core Modeling Principles.”
- Use a numeric or date-valued field or expression as the dimension data.

  Also see the chapter “Using Other Tables to Define Dimensions, Measures, and Listing Fields.”
- For **Data Type**, choose **Number**, **Date**, or one of the date variations.
- Ensure that the measure name uses only alphanumeric characters and is not the same as the name of any dimension. Otherwise, the measure is not available in the Analyzer.
- When you define a numeric measure, DeepSee truncates any data after two decimal places, so that the measure can also be used as a dimension. If this is undesirable, use the following workaround: Within the base measure definition, apply a multiplicative factor to it, such as 100. Then in the Analyzer, create a calculated measure that divides this base measure by that same factor.

Also see “Defining Custom Aggregates for Use as Base Measures,” later in this book.

8.3 Next Steps

After you define, modify, or delete a base measure, do the following to make your changes visible in the Analyzer:

1. Recompile and rebuild, as described in “Recompiling and Rebuilding” earlier in this book.
2. If you have not yet done so, define a subject area, as described later in this book. Be sure to give access to all suitable roles.
3. If the subject area is already open in the Analyzer, you may need to reopen it.
4. In the Analyzer, optionally create calculated measures based on this base measure.

   For information, see *Using the DeepSee Analyzer* or *Expressions and Formulas in DeepSee*.

9

Defining Subject Areas

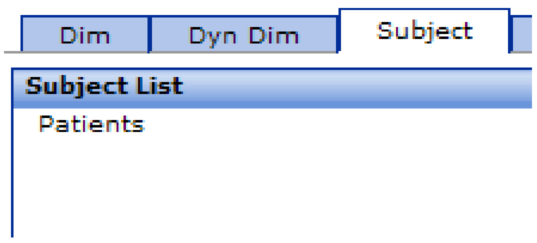
This chapter describes how to create subject areas. It discusses the following topics:

9.1 Overview of the Subject Area Tab

You use the **Subject** tab to define, redefine, and delete subject areas. This screen has two main areas.

9.1.1 Subject List

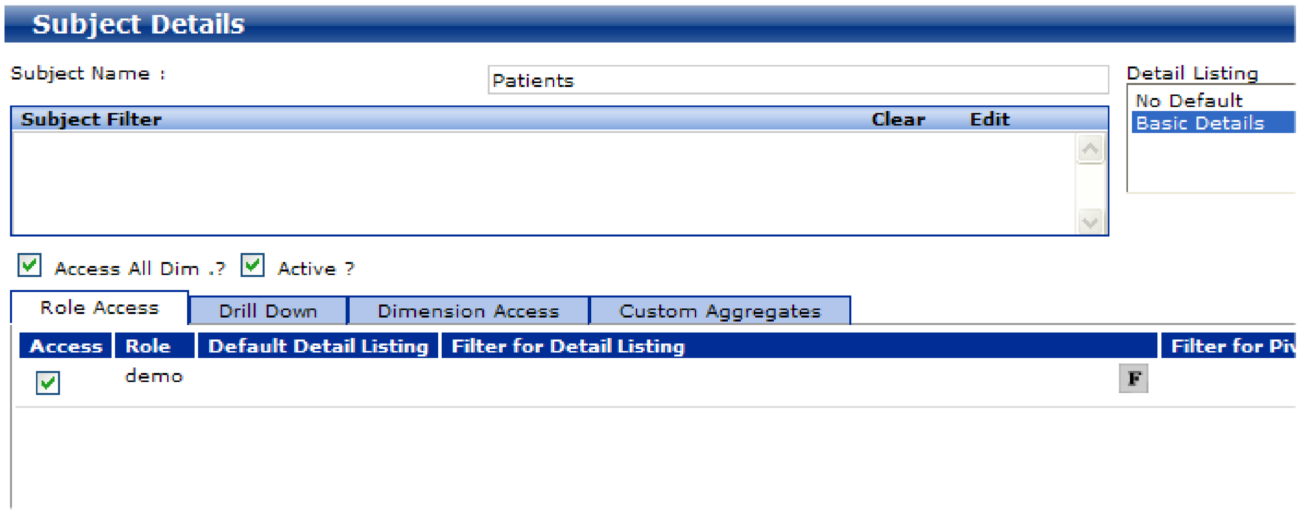
The left area lists the subject areas that have already been defined, if any.



Here you click the subject area that you want to work on.

9.1.2 Subject Details

The right area displays details for the currently selected subject area.



The following table summarizes the options. Unless otherwise noted, the sections listed here are in this chapter.

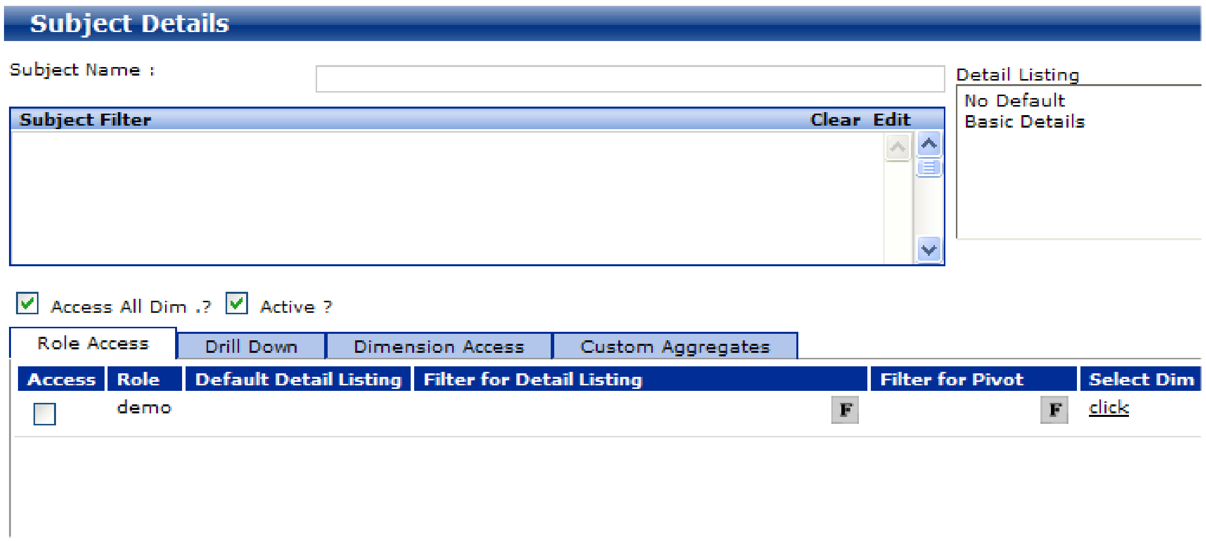
Table 9–1: Options for Subject Areas

Option	Purpose
Subject Name	Name of currently selected subject area; you can change this name.
Detail Listing	This list shows all the available detail listings. The highlighted detail listing is the default detail listing, if any, for this subject area.
Subject Filter options	Optionally specifies how this subject area is filtered. See "Filtering a Subject Area."
Access All Dim. check box	Controls whether the subject area includes all dimensions. See "Specifying the Available Dimensions in a Subject Area."
Active? check box	Controls whether this subject area is active (visible in the Analyzer). When you create a subject area, it is active by default. See "Deactivating or Activating a Subject Area."
Role Access tab	Lists the roles that have access to this subject area. See "Specifying Role Access to a Detail Listing."
Drill Down tab	Optionally defines the dimension-to-dimension (double-click) drill options available in this subject area. See "Defining Dimension-to-dimension Drill Options."
Dimension Access tab	Optionally specifies the subset of dimensions included in this subject area. See "Specifying the Available Dimensions in a Subject Area."
Custom Aggregates tab	Optionally specifies custom aggregates, for use in defining measures. See "Defining Custom Aggregates for Use in Base Measures."

9.2 Defining a Subject Area: The Basics

This section describes the basic method for defining a subject area:

1. Open a BI-enabled class as described in “Opening a BI-enabled Class,” earlier in this book.
2. Click the **Subject** tab.
3. Click **New** in the lower right, to ensure that the screen is clear.



Access	Role	Default Detail Listing	Filter for Detail Listing	Filter for Pivot	Select Dim
☐	demo		F	F	click

4. In **Subject Area**, type the name of this subject area.
5. In **Detail Listing**, optionally click the detail listing to use as the default within this subject area.
6. On the **Role Access** tab, click the check box next to each role that should have access to this subject area.

   By default, no roles are selected.
7. Click **Add**.

This new subject area has access to all the dimensions associated with the BI-enabled class and all the data that is currently in the database. If you have already built the DeepSee indices, you can immediately use the subject area in the Analyzer.

9.3 Modifying a Subject Area

To modify the definition of a subject area:

1. Open a BI-enabled class as described in “Opening a BI-enabled Class,” earlier in this book.
2. Click the **Subject** tab.
3. Click the subject area in the **Subject List** area.
4. Make the changes.
5. Click **Update** to save your changes.

9.4 Filtering a Subject Area

By default, a subject area provides access to all records in the associated tables. You can filter a subject area to restrict the number of records that it can access, for security or for performance.

To filter a subject area:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Subject** tab.
3. Click the subject area in the **Subject List** area.
4. In the **Filter** box, type a filter expression that selects the data for this subject area. For example:

   ```
   [Gender = M]
   ```

   Or click the **Edit** option and define the filter expression interactively. For syntax and other details, see the book *Expressions and Scripts in DeepSee*.

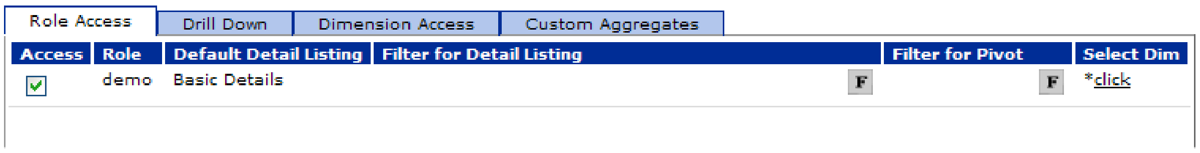
   Note: The **Edit Filter** dialog box does not let you enter a filter expression until you have defined at least one subject area. (Before that, the dialog box is displayed as a blank box.) Also, it shows only the dimensions whose definitions were active the last time the DeepSee indices were built.

Also see "Applying a Runtime Filter to Specify Role Access," later in this chapter.

9.5 Specifying Role Access to a Subject Area

To specify the roles that have access to a subject area:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Subject** tab.
3. Click the subject area in the **Subject List** area.
4. Click the **Role Access** tab.
5. Click the check box next to each role that should have access to this subject area. Clear the check box for any roles that should not have access.



Access	Role	Default Detail Listing	Filter for Detail Listing	Filter for Pivot	Select Dim
✓	demo	Basic Details	F	F	*click

6. Click **Save**.

9.5.1 Applying a Runtime Filter to Specify Role Access

Rather than creating multiple subject areas with different filters, you may be able to define a single subject area gives access to all roles on the **Role Access** tab but that uses the value *%session.Data("CurrRole")*, which contains the DeepSee role with which the user logged in.

For example:

```
[Region Name = {$CASE(%session.Data("CurrRole"),"role1":"North","role2":"South","role3":"West",:"East")
 }]
```

9.5.2 Advanced Role Options

You can also make the following additional changes in the row for a given role:

- In the **Default Detail Listing** field, double-click to display the drop-down list. Then select a detail listing to use as the default for users of this role. This option overrides the default detail listing for the subject area.
- In the **Filter for Detail Listing** field, type a filter expression to be applied to any detail listing that is displayed by users of this role. Or click the **F** option and define the filter expression interactively.
- In the **Filter for Pivot** field, type a filter expression to be applied to any pivot table that is displayed by users of this role. Or click the **F** option and define the filter expression interactively.

For filter syntax and other details, see the book *Expressions and Scripts in DeepSee*.

You can also filter the subject area; see "Filtering a Subject Area."

9.6 Defining Dimension-to-dimension Drill Options

DeepSee provides support for dimension-to-dimension drill through. For example, a user might double-click on an age group in a pivot table and immediately see a drill through by age bucket, for that age group.

Or, if the user is viewing the pivot table in graph mode, the user might right-click an age group and select **Drill Down**. DeepSee would then display a graph showing the age buckets for that age group.

In either case, if more than one drill option is available, DeepSee displays a list of the available options, and the user can choose one.

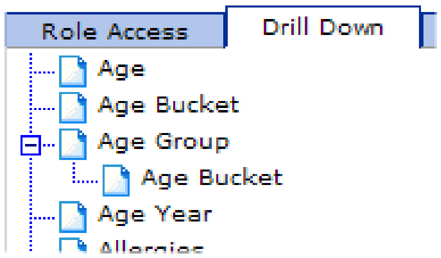
This type of drilling is configured separately for each dimension; that is, the available behavior is potentially different for each dimension.

To define dimension-to-dimension drill options:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Subject** tab.
3. Click the subject area in the **Subject List** area.
4. Click the **Drill Down** tab. This tab displays all dimensions in the subject area.

   Note: Despite the tab name, here you are defining drill through, not drill down.

5. To enable drill options in a given dimension, drag and drop other dimensions to that dimension. For example, drag and drop `Age Bucket` to `Age Group`. Then you see the dimension that you dragged shown as a node beneath the dimension you dragged it to; for example:



6. Click **Update** to save your changes.

To define other drill options, you use the Analyzer. See *Using the DeepSee Analyzer*.

9.7 Specifying the Available Dimensions in a Subject Area

By default, when you create a subject area, it includes all dimensions that are defined in the model. You can define the subject area so that it includes only a subset of those. To do so:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Subject** tab.
3. Click the subject area in the **Subject List** area.
4. Clear the **Access All Dim.** check box.
5. Click the **Dimension Access** tab. This tab displays all dimensions that are defined for the model.
6. Click the check box for each dimension to include in this subject area.
7. Click **Update** to save your changes.

9.8 Defining Custom Aggregates for Use as Base Measures

A subject area can include *custom aggregates*. A custom aggregate uses a Caché ObjectScript expression to retrieve data directly from the globals that store data for your application. When you define custom aggregates, they are available as additional base measures within the measures editor in the DeepSee Analyzer (in addition to `Count`, for example).

To define a custom aggregate:

1. Make sure you understand the structure of the globals that store your data.
2. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
3. Click the **Subject** tab.
4. Click the subject area in the **Subject List** area.
5. Click the **Custom Aggregates** tab.
6. Right-click and then click **Add Line**. A new, empty line is added to the table on this tab.

   Click within this line and type.

7. For **Aggregate Name**, specify a user-friendly name (for example `FreightFromGlobal`). Any user who is going to define measures in the Analyzer should be able to understand what this custom aggregate is.

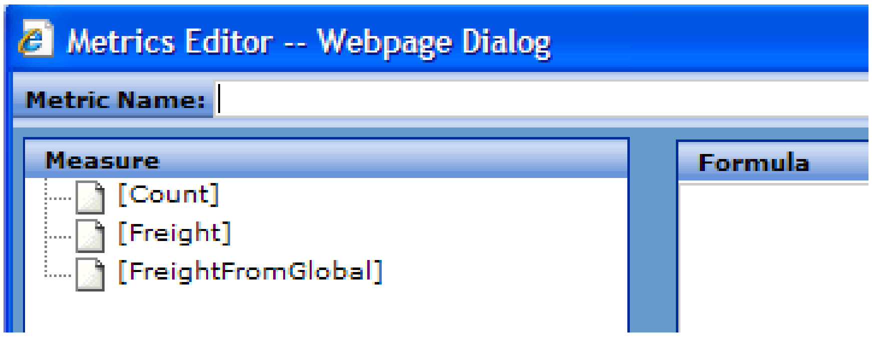
   Note: Do not use spaces in the name. (If you do, the custom aggregate cannot be used.)

8. For **Aggregate Access Code**, specify a Caché ObjectScript expression that retrieves data from the appropriate global. In this expression, use *Id* to refer to the object ID. For example:

   ```
   $LG(^Sample.OrdersD(Id),4)
   ```

9. Click **Update** to save your changes.

The new custom aggregate is shown in the measures editor in the DeepSee Analyzer as follows:



9.9 Activating or Deactivating a Subject Area

When you create a subject area, it is automatically activated. You can deactivate it so that it is no longer available in the Analyzer.

To activate or deactivate a subject area:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Subject** tab.
3. Click the subject area in the **Subject List** area.
4. To deactivate the subject area, clear the **Active?** check box.

   Or to activate the subject area, click the **Active?** check box.
5. Click **Update** to save your changes.

9.10 Deleting a Subject Area

To delete a subject area:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Subject** tab.
3. Click the subject area in the **Subject List** area.
4. Click **Delete**.

9.11 Next Steps

After you define, modify, or delete a subject area, your changes are usually immediately visible in the Analyzer.

If the subject area is already open in the Analyzer, in some cases, you may need to reopen it.

10
Defining Listing Fields

This chapter describes how to create listing fields. It discusses the following topics:

- An overview of listing fields
- An overview of the List Field Library tab
- The basic process to define a listing field
- How to modify a listing field
- How to specify the data on which a listing field is based, particularly if you do not want to base it on a single class property as in the basic technique
- How to choose a data type for a listing field
- How to control whether a listing field is included in the subset that can be selectively rebuilt
- How to activate or deactivate a listing field
- How to delete a listing field
- How to make your changes visible in the Analyzer

Also see the chapter "Using Other Tables to Define Dimensions, Measures, and Listing Fields."

10.1 Overview of Listing Fields

There are two kinds of listing fields:

- A *dimension-type listing field* is a listing field whose definition is owned by a dimension definition. If you redefine the dimension, the listing field is automatically redefined as well. (Similarly, if you delete the dimension, the listing field is automatically deleted.)

  The data used by the listing field is stored in a DeepSee global, which means that its performance is fast. This kind of listing field, however, requires more disk space than the other kind. The number of different values of the listing field affects the size of the global; more values make the global larger.

- An *independent listing field* is a listing field that has an independent definition (not dependent on the definition of any dimension).

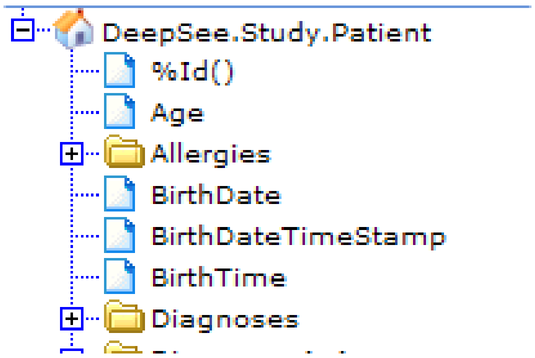
  For this kind of listing field, DeepSee does not access the data until the user displays the detail listing; then DeepSee directly accesses the source data.

10.2 Overview of the List Field Library Tab

You use the **List Field Library** tab to add, modify, and delete the independent listing fields; it does not display the dimension-type listing fields. This tab has three main areas.

10.2.1 Class Hierarchy

The class hierarchy on the left shows the *base class* (the BI-enabled class) and most of the classes available from it via property relationships:



Here you can do the following:

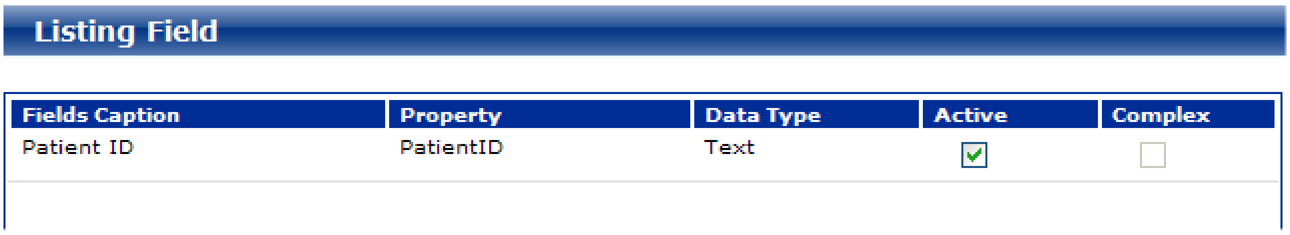
- Expand a collapsed folder by clicking the Expand button ⊞ to its left.
- Collapse an expanded folder by clicking the Collapse button ⊟ to its left.
- Double-click a property to create a new listing field definition based on it.
- Drag and drop a property, instance method, or SQL shortcut to the **Listing Field** table on the right, to create a new listing field definition based on it.

  Any instance methods are shown alphabetically after the properties.

For details on what DeepSee displays in this hierarchy, see "Available Properties and Methods of a BI-enabled Class," earlier in this book.

10.2.2 Listing Field Table

The **Listing Field** table on the right summarizes the currently defined independent listing fields:


Listing Field

Fields Caption	Property	Data Type	Active	Complex
Patient ID	PatientID	Text	✓	☐

Each row corresponds to a listing field. Here you click the listing field that you want to work on.

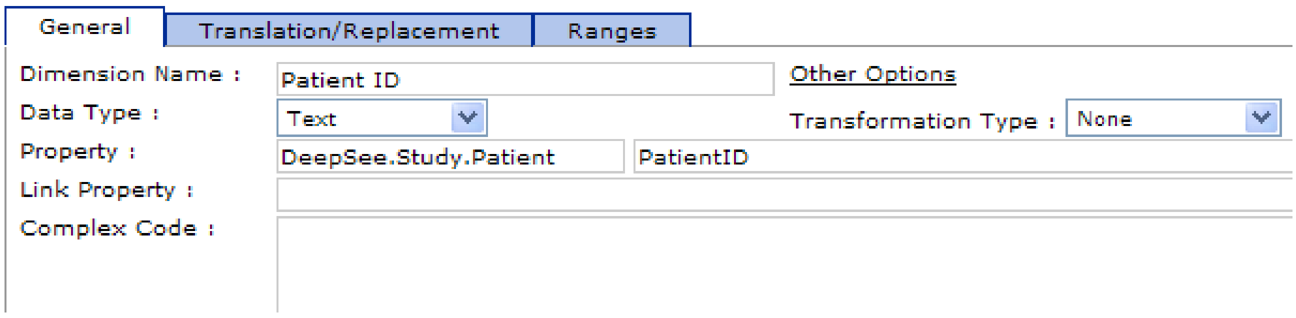
- **Fields Caption** is the name of the listing field, as seen in any detail listings that include it. When you first create a listing field, this is the same as the name of the property, instance method, or SQL shortcut.

- **Property** is the property or instance method on which it is based. This field is empty if the listing field is based on complex code.

  If you use an SQL shortcut, **Property** shows the property used in that shortcut.
- **Data Type** controls how the data is displayed. The data type is set automatically based on the data type of the field you are using, but you can change it. See "Specifying the Data Type for a Listing Field."
- **Active** controls whether this listing field is active (visible in the Analyzer and in pivot tables). When you create a listing field, it is active by default. See "Deactivating or Activating a Listing Field" later in this chapter.
- **Complex** indicates whether this dimension is based on a Caché ObjectScript expression. See "Specifying the Data on Which a Listing Field Is Based" later in this chapter.

10.2.3 Tabs

The tabs in the lower right display details about the currently selected listing field:



Dimension Name : Patient ID
Data Type : Text
Transformation Type : None
Property : DeepSee.Study.Patient PatientID
Link Property :
Complex Code :

The following table summarizes the options on these tabs. Unless otherwise noted, the sections listed here are in this chapter.

Table 10–1: Options on the List Field Library Tabs

Option	Purpose
General tab: **Other Options** link	Provides miscellaneous options. See "Deactivating or Activating a Listing Field" and "Controlling Whether a Listing Field Is Selectively Rebuilt."
General tab: **Data Type** option	Controls how the data is displayed. The data type is set automatically based on the data type of the field you are using, but you can change it. See "Specifying the Data Type for a Listing Field."
General tab: **Transformation Type** option	Applies a transformation that affects how the listing field data is defined and displayed. See the chapter "Using Replacements, Ranges, and Transformations." If you apply a transformation, do not use the **Translation/Replacement** and **Ranges** tabs.
General tab: **Property** field and **Complex Code** field	Specifies the data on which this listing field is based. See "Specifying the Data on Which a Listing Field Is Based."
Translation/Replacement tab	Defines a set of replacements that alter the listing field data. See the chapter "Using Replacements, Ranges, and Transformations."
Ranges tab	Defines a set of ranges to display instead of the listing field data (applies only to numeric-valued listing fields). See the chapter "Using Replacements, Ranges, and Transformations."

10.3 Defining a Listing Field: The Basics

Listing fields and dimensions are defined in similar ways; the difference is that a dimension definition includes more information than the listing field definition. Sometimes you want to define both a dimension and a listing field using the same basic definition; the Architect provides a couple of ways to do this.

As a result, there are two kinds of listing fields:

- To define a dimension-type listing field, click the **Listing Field** check box when you define the dimension. See the subsection "Creating a Dimension-type Listing Field."
- You can define an independent listing field in a couple of equivalent ways:
  - You can copy a dimension definition and use it as the basis of a listing field definition, which you can then modify. See the subsection "Creating an Independent Listing Field by Copying a Dimension." Then, if you redefine or delete either the dimension or the listing field, the other is unaffected.
  - You can create a listing field definition from scratch. See the subsection "Creating an Independent Listing Field from Scratch."

10.3.1 Creating a Dimension-type Listing Field

For any type of dimension other than **Reference**, you can define the dimension for use as a listing field as well. (For information on this type, see "Specifying the Dimension Data Types.")

To use a dimension as a listing field:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Dim** tab.
3. Click the dimension in the **Dimensions** table to select it.
4. In the **General** tab, click the **Listing Field** check box.

   Notice that at the top of the page, the **Listing** check box is checked; this indicates that this definition specifies both a dimension and a listing field.
5. Click **Update** to save your changes. The listing field is now created.

Dimension-type listing fields are stored within a global, with 500 records stored by default in each node of the global. If the listing field data contains many characters, DeepSee can exceed the limit permitted for a global node, which causes the following error:

```
Error: <MAXSTRING>dlrTR2+4^%bi.SWr.WebMain2
ErrorNo: 5002
CSP Page:
/csp/sys/bi/%CSP.Broker.cls
Namespace: NZZ
Class: %CSP.Broker
Routine: %bi.SWr.WebMain2
Location: dlrTR2+4
```

If you receive this error when you build the DeepSee indices, change the definition of the dimension and set the value of **Count Per Node** to a number that is smaller than 500. (This setting is on the **General** tab below the table.)

Or create an independent listing field instead, as described later in this section.

10.3.2 Creating an Independent Listing Field by Copying a Dimension

To create independent listing fields by copying one or more dimension definitions:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Dim** tab.
3. Then do one of the following:
   - To copy the dimension definition, right-click the dimension in the **Dimensions** table and click **Add To Listing Fields**.
   - To copy all the dimension definitions, right-click anywhere in the **Dimensions** table and click **Add All To Listing Fields**.

     If you already have a dimension-type listing field that uses a given dimension, that listing field is not added a second time.

   The definition or definitions are copied immediately.
4. Click the **List Field Library** tab and make edits as needed.

10.3.3 Creating an Independent Listing Field from Scratch

To define an independent listing field manually, do the following:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **List Field Library** tab.
3. Expand the class hierarchy in the left area so that you can see the class property that you want to use.
4. Drag and drop the property, instance method, or SQL shortcut from the class hierarchy to the **Listing Field** table on the right. Or double-click the property, instance method, or SQL shortcut.

   In either case, the **Listing Field** table now includes a listing field that is based on your selection.
5. Click this new row so that it is selected. The bottom right area of the page then shows details.
6. Optionally rename the listing field by changing the value in **Dimension Name** on the **General** tab. This name is shown in the Analyzer and is also used as the default column header in the detail listings. The name should make sense on its own; notice that it does not indicate the table that owns this property.
7. Optionally make additional changes as described in the rest of this chapter.
8. Click **Update** to save your changes.

Also see the subsections in "Defining a Dimension," earlier in this book.

10.4 Modifying a Listing Field

The way to modify a listing field depends on its type:

- If the listing field is a dimension-type listing field, modify the dimension definition. See "Modifying a Dimension," earlier in this book.
- If the listing field is an independent listing field, modify its definition on the **List Field Library** tab, as described in the following steps.

To modify the definition of an independent listing field:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **List Field Library** tab.
3. Click the listing field in the **Listing Fields** table.
4. Make the changes.
5. Click **Update** to save your changes.

To modify the definition of a dimension-type listing field, modify the dimension. See the chapter "Defining Dimensions."

10.5 Specifying the Data on Which a Listing Field Is Based

A listing field of any kind is based on data in the same way that a dimension is. There are two cases:

- A listing field is based on a single class property, instance method, or SQL shortcut.
- A listing field is based on a Caché ObjectScript expression, possibly using multiple class properties.

The techniques are the same as they are for dimensions. See the section "Basing a Dimension on a Caché ObjectScript Expression," earlier in this book.

10.6 Specifying the Data Type for an Independent Listing Field

The data type of a listing field affects how its data is displayed. To specify the data type for an independent listing field, choose a value for **Data Type** on the **General** tab. Depending on your choice, the page then shows another field that you can use to specify an appropriate formatting option.

The following table describes the options.

In the following table, the phrase "dimension data" refers to the data on which the dimension is based (whether that is a single property or an expression).

Table 10–2: Data Types for Listing Fields

Listing Field Data Type	Purpose and Comments
Text	Displays the data as text. This is recommended for date values other than Caché dates.
Check Box	Display a check box instead of the data: • The check box is cleared if the data equals 0 or false. • The check box is selected if the data equals 1 or true.

Listing Field Data Type	Purpose and Comments
CacheDate	Displays the data as a date. If you select this, the system displays a drop-down list that you can use to select a display style. This is suitable for use with Caché date-type data (dates in **$HOROLOG** format).
CacheTime	Displays the data as time. If you select this, the system displays a drop-down list that you can use to select a display style.
Number	Displays the data as a number.
Currency	Displays the data as currency. If you select this, the system displays another field in which you can type the number of decimal places to display; the default is 1.

10.7 Controlling Whether a Listing Field Is Selectively Rebuilt

The Architect provides an option to rebuild only a subset of the listing fields. Within the definition of each listing field, you specify whether that listing field is included within the selective rebuild. To modify this for an independent listing field:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **List Field Library** tab.
3. Click the listing field.
4. On the **General** tab, click the **Other Options** link, which displays a dialog box.
5. To enable this listing field to be rebuilt selectively, click **Selective Rebuild** and click **OK**.
6. Click **Update** to save your changes.

For a dimension-type listing field, the listing field is rebuilt selectively if the dimension is rebuilt selectively. See the chapter "Defining Dimensions."

See "Rebuilding Selectively," earlier in this book.

10.8 Activating or Deactivating a Listing Field

When you create a listing field, it is automatically activated. You can deactivate it so that it is no longer available in the Analyzer.

To activate or deactivate an independent listing field:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **List Field Library** tab.
3. Click the listing field.
4. On the **General** tab, click the **Other Options** link, which displays a dialog box.
5. To deactivate the listing field, clear the **Active?** check box.

Or to activate the listing field, click the **Active?** check box.

6. Click **OK** to exit the dialog box.
7. Click **Update** to save your changes.

To activate or deactivate a dimension-type listing field, activate or deactivate the dimension. See the chapter "Defining Dimensions."

10.9 Deleting a Listing Field

To delete an independent listing field, do the following:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **List Field Library** tab.
3. In the upper right area, click the listing field.
4. Click **Delete** to save your changes. You are prompted to confirm this action.

To delete a dimension-type listing field, do one of the following:

- Delete the dimension.
- Modify the dimension and clear the **List Field** check box on the **General** tab.

See the chapter "Defining Dimensions."

10.10 Next Steps

After you define, modify, or delete any kind of listing field, do the following to make your changes visible in the Analyzer:

1. If you have added a listing field, add it to a detail listing, as described in the chapter "Defining Detail Listings."
2. If you have deleted a listing field, remove it from all detail listings, as described in the chapter "Defining Detail Listings."
3. Recompile and rebuild, as described in "Recompiling and Rebuilding" earlier in this book.
4. If you have not yet done so, define a subject area, as described in the chapter "Defining Subject Areas."

   When you define the subject area, make sure to give access to all suitable roles.
5. If the subject area is already open in the Analyzer, you may need to reopen it.

11

Using Other Tables to Define Dimensions, Measures, and Listing Fields

The preceding chapters describe how to define dimensions, measures, and listing fields that are based on properties that are in the base class or in classes that are related to it by property relationships.

Sometimes, however, it is necessary to access properties in other tables, properties that you cannot access via dot syntax. This chapter describes the techniques for doing so:

- Using complex code and SQL
- Using the Link options
- Using a class query

11.1 Using Complex Code and SQL

You can use the following technique to define a dimension, measure, or listing field on a property in a table that is external to the base class:

1. Write a class method or a routine that uses dynamic SQL or embedded SQL to access the external property.
2. Use **Complex Code** to invoke that method or routine.

For example, suppose that the base class is DeepSee.Study.Patient, and suppose that another class DeepSee.Study.PatientDetails contains properties that you want to use as a dimension. There is no property that connects these classes; that is, you cannot access properties of DeepSee.Study.PatientDetails from DeepSee.Study.Patient via cascading dot syntax.

Instead, in both classes, the property PatientID, which is not the primary key, identifies the patient.

You could write a method that uses dynamic SQL like the following:

```
ClassMethod GetFavoriteColor(patientID As %String) As %String
{
    Set ReturnValue="No Data Available"
    Set myquery="SELECT FavoriteColor AS ReturnValue "
    Set myquery=myquery_"FROM DeepSee_Study.PatientDetails "
    Set myquery=myquery_"WHERE PatientDetails.PatientID=?"
    Set rset=##class(%ResultSet).%New("%DynamicQuery:SQL")
    Set status=rset.Prepare(myquery)
    If $$$ISERR(status) {
        Do $System.Status.DisplayError(status) Write "Prepare error"}
    Set status=rset.Execute(patientID)
    If $$$ISERR(status) {
        Do $System.Status.DisplayError(status) Write "Execute error"}
    While rset.Next() {
     Set ReturnValue=rset.Data("ReturnValue")
     }
    Quit ReturnValue
}
```

Then you could define a dimension as follows:

- **Dimension Name**: `Favorite Color`
- **Property** (first field): `DeepSee.Study.Patient`
- **Property** (second field): empty
- **Complex Code**:

  ```
  ##class(DeepSee.Study.Utils.ForModel).GetFavoriteColor(%this.PatientID)
  ```

The method is executed when the indices are generated.

Tip: To make this method run more quickly, be sure to add an index to the child table, on the field that you are using in the SELECT statement. In the example here, add an index to PatientID in the PatientDetails table.

11.2 Using the Link Options

The Architect provides options (**Link Property** and **Link To**) that you can use to access properties of classes that are external to the base class. You can use these options to define dimensions, but not measures or listing fields. You cannot use the dimension for filtering.

You can use the Link options only if the child table points to the ID property in the parent table. This is because the Link options internally use object relationships. If the child table does not point to the ID property of the parent table, use **Complex Code** and embedded SQL as described earlier in this chapter.

11.2.1 Using the Link Options

This method is best illustrated via an example. Suppose that Sample.Orders is the base class that you are using. The class Sample.OrderDetails contains useful information such as the product name and product category, which you want to use as dimensions. Sample.Orders does not have any property that gives us access to Sample.OrderDetails.

To access properties of Sample.OrderDetails from within Sample.Orders, it is necessary to indicate how the tables are associated. To do so:

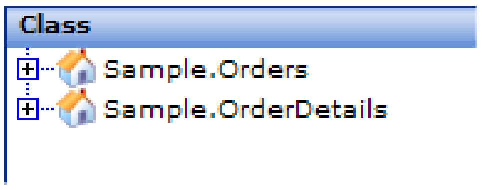
1. Open the base class that you are working with (for example Sample.Orders), as described earlier in this book.
2. Click **Add** in the bottom left area.

   This displays a search dialog box.
3. Optionally type the package name or complete package and class name into **Search Class** and then click **Search**.

The system displays a list of classes that match your search criteria. (Note that the class you are now adding does not have to be BI-enabled, but must have some relationship to the primary class.)

4. Click the class name and then click **OK**.

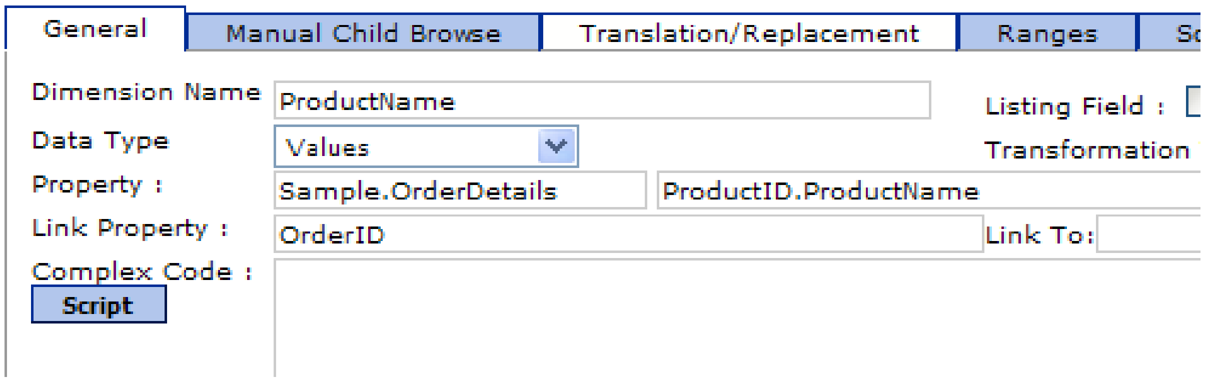
   Now the **Class** list displays both classes:



5. Expand the secondary class as needed, so that you can see the property or instance method that you want to use.

6. Drag and drop the property or instance method from the secondary class to the **Dimension** table; this adds a new dimension.

   For example, if you dragged and dropped the ProductName property. You would then see the following new dimension:

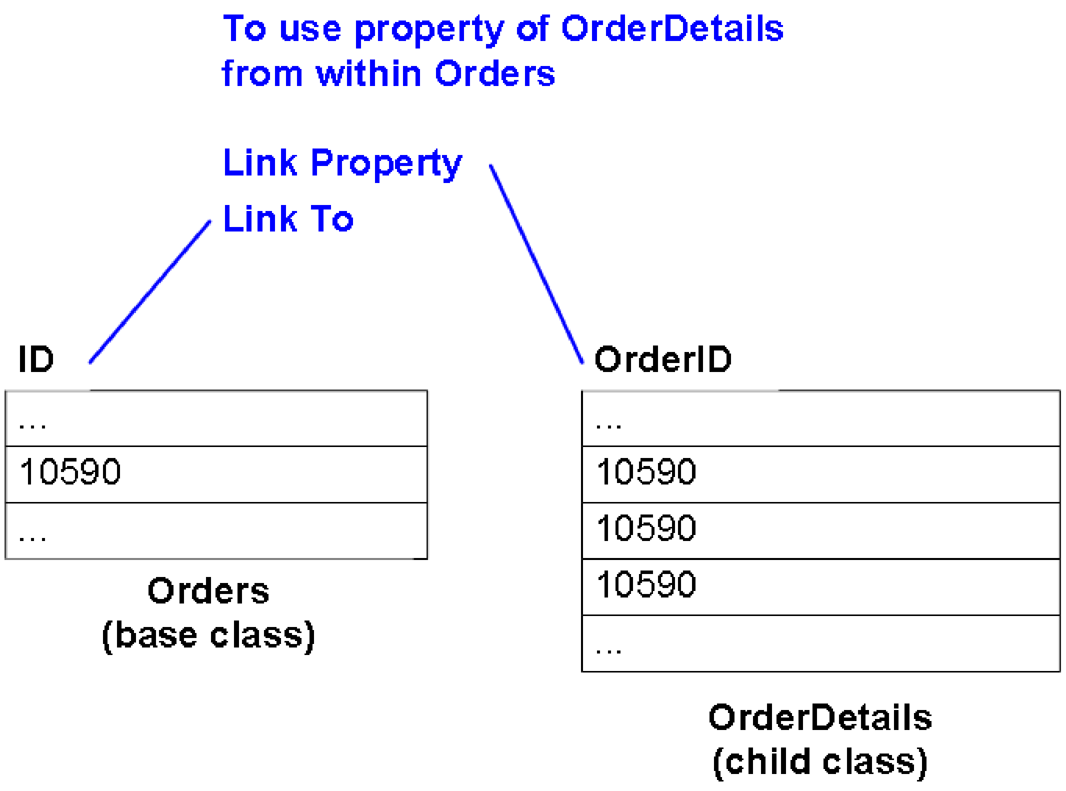


Dimension Name: ProductName
Data Type: Values
Property : Sample.OrderDetails ProductID.ProductName
Link Property : OrderID
Link To:
Complex Code :

In general, you need to specify one or both of the following values:

- **Link Property** indicates the property in the child class that refers to the base class. In this case, this value was set automatically because of the drag-and-drop action.
- **Link To** indicates the ID property in the base class to which the child class points. The default is *%this.ID*. In this example, the default is used.

The following figure shows these two properties.



11.2.2 A Look at the Internals

Internally, when you use the Link options, DeepSee uses that information to determine which IDs of the child table to use, as follows:

```
select ID into :ChildId from TEST.OrderDetails where OrderID=:Id
```

In this example *OrderID* and *ID* are the values we provided for in the **Link Property** and **Link To** fields.

When the DeepSee indices are generated, DeepSee uses the preceding information as follows. For each row in the parent table, the system finds the IDs for all the associated rows in the child table. DeepSee then iterates through those rows and extracts the source data as specified.

11.3 Using a Class Query

In some cases, it is simpler to create a class query that combines multiple tables as needed and then to use that as the basis of the DeepSee model. The following steps describe the technique:

1. Create a class that inherits from %RegisteredObject and %BI.Adaptor.
2. In this class, create a query that selects the desired fields. This query must specify which field is the ID field.

   The class definition should include only one query. DeepSee ignores additional queries.
3. In the Architect, use those fields as dimensions or measures.

   Note: You cannot use the fields as listing fields.

This technique is also useful if you cannot modify the base class, if you have an existing complex query that you want to use, or if you want to combine multiple classes via an SQL UNION.

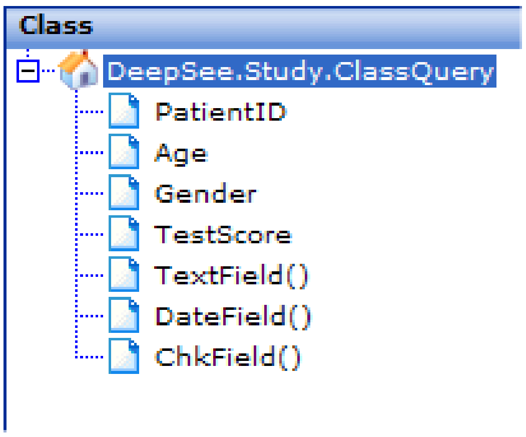
For example, suppose that DeepSee.Study.Patient contains unique patient records, and that DeepSee.Study.PatientSet2 contains an additional set of unique patient records (perhaps that come from a different source). To perform analysis on the combined set of patients, you could create an additional class like the following:

```
Class DeepSee.Study.PatientClassQuery Extends (%RegisteredObject, %BI.Adaptor)
{

Query Patients() As %SQLQuery(CONTAINID = 1)
{
 SELECT PatientID,Age,Gender,TestScore FROM DeepSee_Study.Patient
 UNION ALL
 SELECT PatientID,Age,Gender,TestScore FROM DeepSee_Study.PatientSet2
}

}
```

DeepSee executes the class query when it builds the indices.

In this example, the `PatientID` field is the ID field, which we indicate in the query by including `CONTAINID = 1`.

When you open this class in the Architect, you see the following:



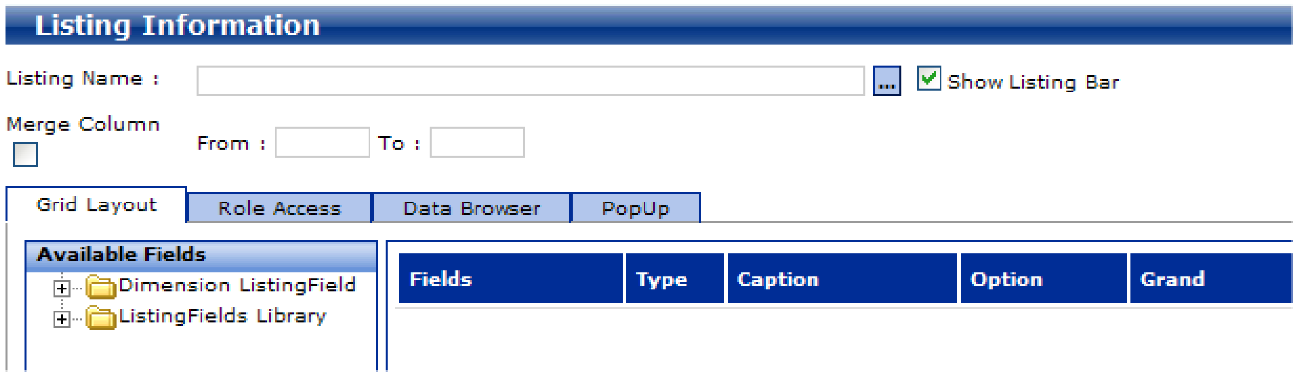
Each field in the query is available for use as a dimension or (if numeric) a measure. You cannot, however, use these fields as listing fields.

12
Defining Detail Listings

This chapter describes how to create detail listings. It discusses the following topics:

12.1 Overview of the Detail List Library Tab

You use the **Detail List** tab to define, redefine, and delete detail listings.



The **Listing Name** field shows the name of currently selected detail listing. To select a detail listing, click the Browse button **...**, click the detail listing, and then click **OK**.

The following table summarizes the other options. Unless otherwise noted, the sections listed here are in this chapter.

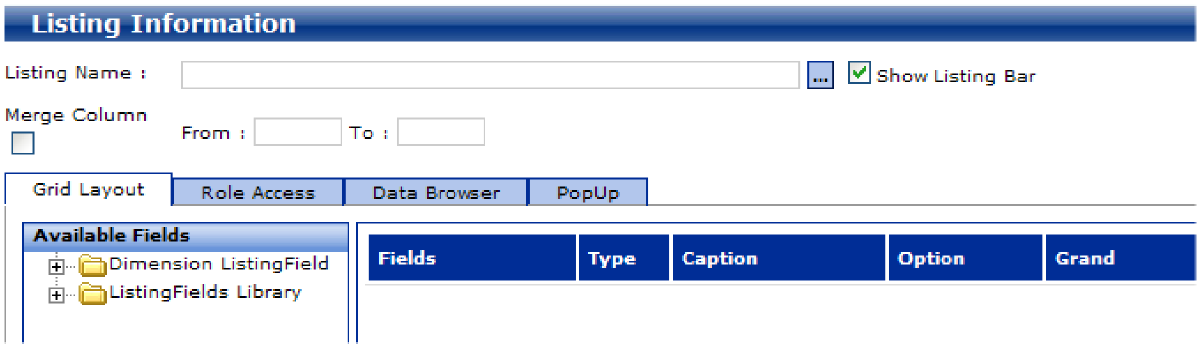
Table 12–1: Options for Detail Listings

Option	Purpose
Show Listing Bar option	Controls whether the title bar is shown within this detail listing.
Grid Layout tab	Lists the listing fields in this detail listing, in order. See "Modifying the Layout of a Detail Listing" and "Adding Summary Lines to a Detail Listing."
Role Access tab	Lists the roles that have access to this detail listing. See "Specifying Role Access to a Detail Listing."
Data Browser tab	Lists the detail listings to which the user can jump, from within this detail listing. See "Adding a Link to Another Detail Listing."

12.2 Defining a Detail Listing: The Basics

You define detail listings on the **Detail List** tab. Although it is useful to become acquainted with all the options on this tab, in many cases you can define detail listings very simply, without knowing these options in detail. This section describes the basic method, which is a suitable start in all cases:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Detail List** tab. The right area of the Architect displays the following:



3. In **Listing Name**, type the name of this detail listing.
4. Optionally clear **Show Listing Bar** if you do not want to include the title bar within this detail listing.
5. In **Available Fields**, expand the two folders so that you can see all the available listing fields.

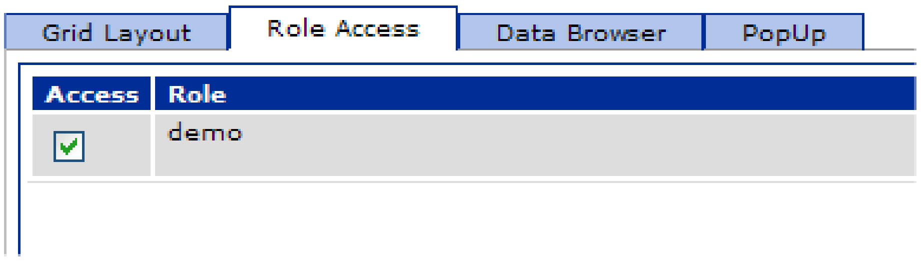
   The **Dimension ListingField** folder lists the listing fields that are based directly on dimension definitions. The **ListingFields Library** folder lists the listing fields that you created manually on the **List Field Library** tab.

   Tip:
   If you do not see all the listing fields you expect to see, click the Refresh button in the menu bar.

6. Double-click each listing field that you want to include.

   Note: The listing field is then copied to the detail list. This means that if you later redefine the listing field, the changes are not reflected in this detail list.

7. After you add a listing field, optionally click it in the table and edit the **Caption** field, which is the column header for this listing field.
8. Click the **Role Access** tab.
9. Click the check box next to each role that should have access to this detail listing. By default, no roles are selected.



Access	Role
✓	demo

10. Click **Add**.

Note: DeepSee does not provide a way to specify the sort order of the detail listing.

12.3 Modifying a Detail Listing

To modify the definition of a detail listing:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Detail List** tab.
3. Click the Browse button ... next to **Listing Name**.
4. Click the detail listing and then click **OK**.
5. Make the changes.
6. Click **Update** to save your changes.

12.4 Modifying the Layout of a Detail Listing

To modify the layout of a detail listing:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Detail List** tab.
3. Click the Browse button ... next to **Listing Name**, click the detail listing, and then click **OK**.
4. Now you can do the following on the **Grid Layout** tab:
   - Move a row up. To do so, right-click it and click **Move Up**. This action moves the corresponding listing field to the left within the detail listing.
   - Move a row down. To do so, right-click it and click **Move Down**. This action moves the corresponding listing field to the right within the detail listing.

- Remove a row. To do so, right-click it and click **DeleteLine**.
- Add a listing field. To do so, in **Available Fields**, expand the two folders so that you can see all the available listing fields.

  Tip:
  If you do not see all the listing fields you expect to see, click the Refresh button in the menu bar.

  Double-click each listing field that you want to include. The additions are added at the bottom of the table.
- Edit the column header for any listing field. To do so, click it in the table and edit the **Caption** field.

5. Click **Save**.

12.5 Adding Summary Lines to a Detail Listing

A detail listing can include either or both of the following kinds of summary lines:

- A summary line for the current page of the detail listing
- A summary line for the entire detail listing

You define these summary lines separately. To add such summary lines:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Detail List** tab.
3. Click the Browse button ... next to **Listing Name**, click the detail listing, and then click **OK**.
4. To add a summary line for the entire detail listing:
   a. Decide which detail listing fields you want to summarize in this line.
   b. Double-click in the first **Option** column, in the row for one of those detail listing fields.
   c. In this column, click an option that specifies the kind of syntax to use. For example, **System** refers to system functions. The options are explained after these steps.
   d. Double-click in the **Grand** column, in the same row.
   e. In this column, either select an appropriate option or type the appropriate syntax. For example, **Total** displays a numeric total; this option is available if you selected **System** in the previous column. The options are explained after these steps.

   Repeat as necessary for all detail listings that you want to summarize in this summary line.
5. To add a page summary line, follow the preceding steps but use the second **Option** column and the **Page** column.
6. Click **Save**.

In either **Option** column, the value you select controls the value to use in the column to its right, as follows:

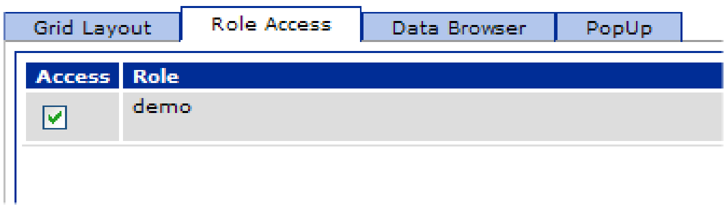
Table 12–2: Options for the Detail Listing Summary Lines

Value of **Option**	Value to Use for **Grand** or **Page**
Label	Type a string to use as a label.
System (available only for dimension-type listing fields)	Select one of the following numeric aggregation functions: • **Total** • **Average** • **Max** • **Min** • **Count** • **Std. Dev** • **Variance** • **% Per** (Percentage of the total value) • **Median**

12.6 Specifying Role Access to a Detail Listing

To specify the roles that have access to a detail listing:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Detail List** tab.
3. Click the Browse button **...** next to **Listing Name**, click the detail listing, and then click **OK**.
4. Click the **Role Access** tab.
5. Click the check box next to each role that should have access to this detail listing. Clear the check box for any roles that should not have access.



Access	Role
✓	demo

6. Click **Save**.

12.7 Adding a Link to Another Detail Listing

Within a detail listing, a user can display another detail listing, in any subject area to which the user has access. Specifically, the user can right-click a row and then select the name of another detail listing; DeepSee then displays that other detail listing with details for the selected row.

To add a link to another detail listing:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Detail List** tab.
3. Click the Browse button ... next to **Listing Name**, click the detail listing, and then click **OK**.
4. Click the **Data Browser** tab.

   This table displays one row for each detail listing that you can access from within this detail listing.
5. For each detail listing to which you want to link, right-click and then click **Add Line**. This adds a new, empty row. Then enter the following information:
   - **Listing ID** — Specifies the detail listing to display. Double-click this field. DeepSee displays a list of all detail listings available to you, given your role access. Click a detail listing and then click **OK**.
   - **Menu Caption** — Specifies a string to display in the context menu. (For example, type the name of the detail listing).
   - **Query** — Specifies the filter expression that indicates which records to select when displaying this associated detail listing.

     If the associated detail listing is in a different subject area, use the following guidelines:

     – The dimension name you use should be the name of the dimension as defined in the other subject area. Use this dimension name on the left side of the filter expression.

     – To select members of that dimension, you can use the syntax *%this.propertyname* to refer to a property in the current subject area. Use this syntax on the right side of the filter expression.

     For example, suppose that we want to link from detail listing 1 (in the subject Orders) to detail listing 2 (in the subject area Order Details). Also suppose that:

     – The Order Details subject area has an `Order ID` dimension.

     – The Orders subject area has an `OrderID` property.

     We would use the following filter expression:

     ```
     [Order ID = %this.OrderID]
     ```

     The user right-clicks in detail listing 1, perhaps on order 10366, and selects Order Details. Then DeepSee evaluates *%this.OrderID*. Next, when it retrieves records from the Order Details subject area to display in detail listing 2, DeepSee applies the filter expression `[Order ID = 10366]`.
6. Right-click in the empty area and select **Save Data Browser**.
7. Click **Save**.

On the **Data Browser** tab, you can also do the following:

- Move a line up or down, to control the order of the options in the detail listing. To do so, right-click a line and then click **Move Up** or **Move Down**.

- Remove a line from the table. To do so, right-click it and then click **Delete Line**. The line is immediately removed.
- Remove all lines from the table. To do so, right-click in the empty area and then click **Clear**. DeepSee prompts you for confirmation.

Note that your changes on this tab are not saved unless you right-click in the empty area and select **Save Data Browser**.

12.8 Adding a Link to a URL

Within a detail listing, a user can right-click, select an option, and go to a URL in a new browser window. To add a right-click option that opens a URL:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Detail List** tab.
3. Click the Browse button ... next to **Listing Name**, click the detail listing, and then click **OK**.
4. Click the **Popup** tab.

   This table displays one row for each URL that you can access from within this detail listing.
5. For each URL to which you want to link, right-click and then click **Add Line**. This adds a new, empty row. Then enter the following information:
   - **Menu Caption** — String to display in the context menu.
   - **Pre Script** — Caché ObjectScript statement to execute before opening the URL.
   - **URL Action** — URL to open in a new browser window.
   - **Post Script** — Caché ObjectScript statement to execute after opening the URL.
6. Click **Save**.

Tip: To remove a line from the table, right-click it and then click **Delete Line**. DeepSee prompts you for confirmation.

12.9 Deleting a Detail Listing

To delete a detail listing:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Click the **Detail List** tab.
3. Click the Browse button ..., click the detail listing, and then click **OK**.
4. Click **Delete**.

12.10 Next Steps

After you define, modify, or delete a detail listing, do the following to make your changes visible in the Analyzer:

1. Recompile, as described in "Recompiling and Rebuilding" earlier in this book.
2. If you defined a new detail listing, make sure to give access to the detail listing to all suitable roles, as described earlier in this chapter.
3. If you have not yet done so, define a subject area, as described later in this book.

   When you define the subject area, make sure to give access to it to all suitable roles.
4. If the subject area is already open in the Analyzer, you may need to reopen it.

13
Using Replacements, Ranges, and Transformations

Replacements, ranges, and transformations allow you to modify the data shown in the Analyzer and in pivot tables, for dimensions and for listing fields. The Architect provides these tools in several contexts. This chapter discusses the following topics:

- An overview
- Predefined transformations provided by DeepSee
- How to replace null values for a dimension, measure, or listing field
- How to define a set of replacements for a dimension or listing field
- How to define a set of ranges for a dimension or listing field
- How to define a transformation to use in a dimension or a listing field
- How to apply a transformation to a dimension or a listing field

13.1 Overview of Replacements, Ranges, and Transformations

Replacements, ranges, and transformations allow you to modify the data shown in the Analyzer and in pivot tables, for dimensions and for listing fields.

A *replacement* substitutes one string for another. If you use a replacement within a dimension, you are effectively renaming a dimension member. If you use a replacement within a listing field, you are replacing any occurrences of the old string in any detail listing that uses this field. The comparison is case-sensitive and it considers the entire string (there is no partial matching option). With one exception, replacements do not affect measures; that is, DeepSee ignores them if the dimension data type is **Number**.

A special case of replacement is the **Null field replacement** option, which replaces null values with the string or number you specify. You can use this option with all dimension data types (including **Number**, which is the type used by measures).

Ranges examine numeric source data and place it into named buckets. For a dimension, when you define ranges, you are effectively defining the members of that dimension. Each member of the dimension corresponds to a range of values. For a listing field, when you define ranges, then in each row of a detail listing, the name of the appropriate range is displayed instead of the listing field data.

Transformations are reusable definitions that include replacements, ranges, and other options such as case conversion and field justification. You define them separately and apply them to dimensions or listing fields where needed.

Transformations have a lower priority than replacements and ranges that are included directly within dimension or listing field definitions; for example, if a dimension includes a set of replacements and has a transformation applied to it, the transformation is ignored.

13.2 Predefined Transformations

For convenience, DeepSee provides the following transformations:

- **Convert to Flag** — Given a set of country names, displays an image file instead of that country name.
- **External Value** — Causes DeepSee to display the external value (the display value) instead of the internal value, for a field that has both a value list and a corresponding display list.

  For example, consider the following property definition:

  ```
  Property Gender As %String(DISPLAYLIST = ",Female,Male", VALUELIST = ",F,M");
  ```

  If you base a dimension on this property, by default, the member names are `F` and `M`. If instead you use the **External Value** transformation, the member names are `Female` and `Male`.

13.3 Replacing Null Values for a Dimension, Measure, or Listing Field

To replace null values for a dimension, measure, or a listing field:

1. Open a BI-enabled class as described in “Opening a BI-enabled Class,” earlier in this book.
2. Select the dimension, measure, or a listing field. Either:
   - Click the **Dim** tab and then click the dimension or measure.
   - Click the **Listing Field** tab and then click the listing field.
3. Click the **Translation/Replacement** tab.
4. Type a value into **Null field replacement**. For example: `No region`
5. Click **Update** to save your changes.

If you edited a dimension or measure, notice that the **Trns/Repl** check box is checked in the table at the top of the page; this happens automatically when you enter information on the **Translation/Replacement** or the **Range** tab.

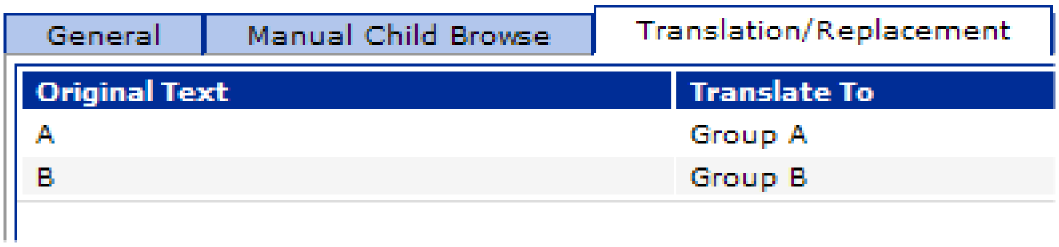
Important: DeepSee ignores the **Null field replacement** option if you have entered an expression into **Complex Code**, if you are using the **Words** dimension data type, or if you are using the **Manual Child Browse** feature. In such cases, write an expression in **Complex Code** that detects the nulls and replaces them appropriately. For an example, see “Force M.C.B. to Return at Least One Value,” earlier in this book.

13.4 Defining Replacements for a Dimension or Listing Field

To define a set of text replacements for a dimension or a listing field:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Select the dimension or a listing field. Either:
   - Click the **Dim** tab and then click the dimension.
   - Click the **Listing Field** tab and then click the listing field.
3. Click the **Translation/Replacement** tab.
4. To add a row to the table:
   a. Right-click in the empty area and select **Add Line**.
   b. Click the blank line, beneath the **Original Text** heading.
   c. Type the exact text that you are going to replace. Matching is exact and case-sensitive.
   d. Press **Tab** or click the blank line beneath the **Translate to** heading.
   e. Type the replacement text.

   Repeat as necessary. For example:



Original Text	Translate To
A	Group A
B	Group B

5. Click **Update** to save your changes.

If you edited a dimension, notice that the **Trns/Repl** check box is checked in the table at the top of the page; this happens automatically when you enter information on the **Translation/Replacement** or the **Range** tab.

13.4.1 Other Options

You can also do the following on the **Translation/Replacement** tab:

- To delete a single line, right-click it and select **Delete Line**. The line is immediately deleted.
- To delete all lines, right-click and select **Clear**. You are prompted to confirm this action.
- To copy all lines of this table to the system clipboard, right-click and select **Copy to Clipboard**.
- To paste the contents of the system clipboard, right-click and select **Paste from Clipboard**.

  CAUTION: If the clipboard does not currently contain lines from a replacement table like this (perhaps from a different dimension or listing field), the current contents of your replacement table are cleared and nothing is pasted.

13.5 Defining Ranges for a Dimension or Listing Field

You can define ranges for dimensions and for listing fields that use numeric data:

- If you define ranges for a dimension, you are defining the members of the dimension. Each range creates one dimension member; the name of the range is the name of the member.

  Also see "Creating Hourly Buckets, "earlier in this book.

- If you define ranges for a listing field, then in each row of a detail listing, the name of the appropriate range is displayed instead of the listing field data.

13.5.1 Basics

To define a set of ranges for a dimension or a listing field:

1. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.
2. Select the dimension or a listing field. Either:
   - Click the **Dim** tab and then click the dimension.
   - Click the **Listing Field** tab and then click the listing field.
3. Click the **Ranges** tab.

   In this tab, you define a set of ranges, each with a name or caption.
4. To add a row to the table:
   a. Right-click in the empty area and select **Add Line**.

      The system then adds a blank line.

   b. Click the blank line, beneath the **Caption** heading.

      Then the cursor blinks to indicate that you can type here.

   c. Type the name of this member.

      To move to other fields in the row, press **Tab** or click the blank line beneath the appropriate heading.

   d. For **From**, type the lower end of the range, if any.

      This can be a Caché ObjectScript expression or a constant. If you use an expression, note that DeepSee evaluates it at compile time.

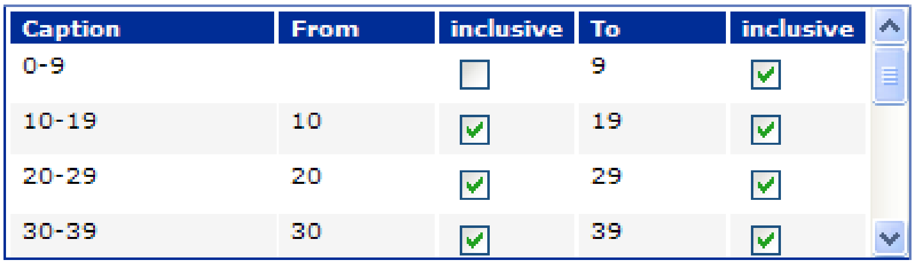
   e. If this range includes this value, click the check box in the first **inclusive** column.

   f. For **To**, type the upper end of the range, if any.

      This can be a Caché ObjectScript expression or a constant. If you use an expression, note that DeepSee evaluates it at compile time.

   g. If this range includes this value, click the check box in the second **inclusive** column.

   Repeat as necessary. For example:



Caption	From	inclusive	To	inclusive
0-9		☐	9	✓
10-19	10	✓	19	✓
20-29	20	✓	29	✓
30-39	30	✓	39	✓

5. If you are editing a dimension, optionally type a string into **Range Dimension Suffix**. This string is appended to the dimension name.
6. Click **Update** to save your changes.

If you edited a dimension, notice that the **Trns/Repl** check box is checked in the table at the top of the page; this happens automatically when you enter information on the **Translation/Replacement** or the **Range** tab.

13.5.2 Other Options

You can also do the following on the **Range** tab:

- To delete a single line, right-click it and select **Delete Line**. The line is immediately deleted.
- To delete all lines, right-click and select **Clear**. You are prompted to confirm this action.
- To copy all lines of this table to the system clipboard, right-click and select **Copy to Clipboard**.
- To paste the contents of the system clipboard, right-click and select **Paste from Clipboard**.

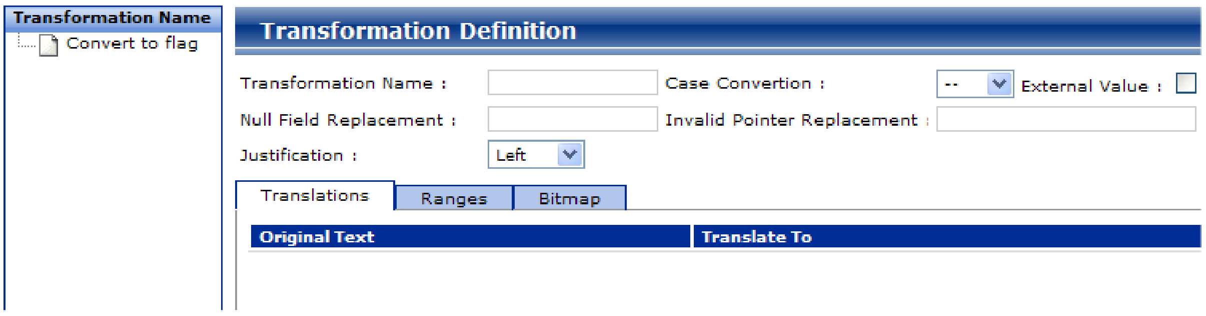
  CAUTION: If the clipboard does not currently contain lines from a range table like this (perhaps from a different dimension or listing field), the current contents of your range table are cleared and nothing is pasted.

13.6 Defining and Modifying Transformations

To define or redefine a transformation:

1. At the top of the Architect page, click **Data Modeler > Transformation**.

   The system then displays a page like the following:



Transformation Name
Convert to flag

Transformation Definition

Transformation Name :
Case Convertion : --
External Value : ☐
Null Field Replacement :
Invalid Pointer Replacement :
Justification : Left

Translations | Ranges | Bitmap

Original Text	Translate To

2. Either:
   - Type a new transformation name into **Transformation Name**.
   - Click a transformation name in the list on the left.

3. Make edits as needed. The options are described after these steps.
4. Click **Add** or **Update** in the lower right.

Table 13–1: Options in a Named Transformation

Option	Purpose
Transformation Name	Name of this transformation, as you want to see in the Architect.
Case Conversion	Optionally specifies how to convert the case of any strings. Choose **Upper Case** or **Lower Case**.
External Value	Click this option if you want DeepSee to use the external value for this field, rather than the internal value. This option takes effect only if the field has both internal and external values.
Null Field Replacement	Optionally specifies a string to use as replacement for any null values.
Invalid Pointer Replacement	Optionally specifies a string to use as replacement for any invalid pointers.
Justification	Specifies how to convert the horizontal alignment of any strings. Optionally choose **Left**, **Center**, or **Right**.
Translations tab	Optionally defines a set of text replacements. Follow the instructions in "Defining Replacements for a Dimension or a Listing Field," earlier in this chapter.
Ranges tab	Optionally defines a set of ranges. Follow the instructions in "Defining Ranges for a Dimension or a Listing Field," earlier in this chapter.
Bitmaps tab	Optionally defines a set of text-to-image replacements. This affects only independent listing fields. See the following subheading.

13.6.1 Defining Text-to-image Replacements

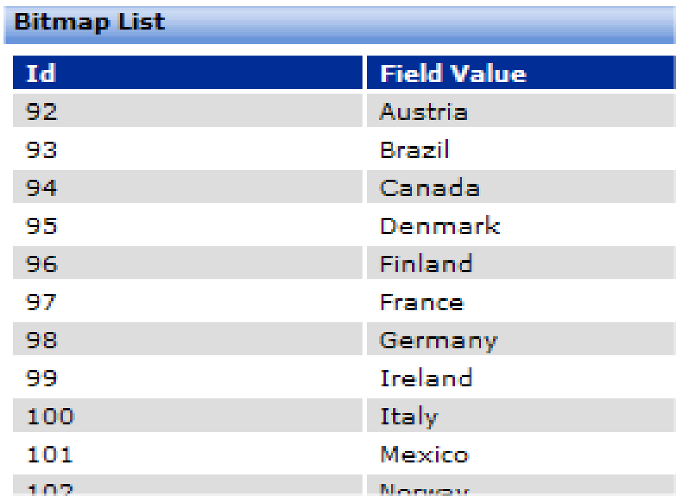
Within a transformation, you can define a set of replacements that substitute images in the place of text. You specify the text to replace, and a bitmap image to display in its place. This replacement affects only independent listing fields; dimensions and dimension-type listing fields ignore any text-to-image replacements.

To define text-to-image replacements within a transformation:

1. Load the image files into the DeepSee library. For information, see the *DeepSee Site Configuration and Maintenance Guide*.
2. Make sure the **Translations** tab is empty. Otherwise, the replacements defined there take precedence over the text-to-image replacements that you are adding.
3. Click the **Bitmaps** tab.
4. Right-click and click **Add Line**.
5. In the right column (**Field Value**), type the exact text that you are going to replace. Matching is exact and case-sensitive.
6. Make sure the new row is still selected.
7. Click **Select Image**.

   This brings up a dialog box.
8. Click the image that you want to use and click **OK**. The internal ID of this image is automatically written to the **Id** column.
9. In the **Remark**, type an optional remark.
10. Repeat as needed.

For example:



Bitmap List

Id	Field Value
92	Austria
93	Brazil
94	Canada
95	Denmark
96	Finland
97	France
98	Germany
99	Ireland
100	Italy
101	Mexico
102	Norway

On this tab, you can also do the following:

- Delete a line. To do so, right-click the line and click **Delete Line**.
- Delete all lines. To do so, right-click and click **Clear**. You are prompted to confirm this action.
- Preview the image used in a given row. To do so, click the line. The image is displayed in the **Bitmap Preview** box.
- Remove an image for a given line. To do so, click the line and click **Clr**.

If you make any change, click **Update** to save this change.

13.6.2 Deleting a Transformation

To delete a transformation:

1. At the top of the Architect page, click **Data Modeler > Transformation**.
2. Click a transformation name in the list on the left.
3. Click **Delete** in the lower right. The transformation is immediately deleted.

13.6.3 Next Steps

After you add, modify, or delete a transformation, recompile and rebuild, as described in "Recompiling and Rebuilding" earlier in this book.

13.7 Applying a Transformation to a Dimension or Listing Field

To apply a transformation to a dimension or a listing field:

1. Define the transformation if it is not yet defined. See the previous section.
2. Open a BI-enabled class as described in "Opening a BI-enabled Class," earlier in this book.

3. Select the dimension or a listing field. Either:
   - Click the **Dim** tab and then click the dimension.
   - Click the **Listing Field** tab and then click the listing field.
4. In the **General** tab, select the transformation from the **Transformation Type** drop-down list.
5. Click **Update** to save your changes.

14
Validating Your DeepSee Data Model

This chapter briefly describes some useful tools and techniques to help you validate your dimensions, measures, and listing fields.

14.1 Useful Tools

To assist you during testing, make sure that the **Listing Field** option is selected for each measure.

Also make sure that you have the following tools:

- A listing field based on an identifier or other unique value in the source data (for example, `OrderID` or `PatientID`. This enables you to easily identify the source records when viewing detail listings.
- A detail listing that displays the preceding listing field and all measures. Even if you do not intend to provide this to users, you will find this tool useful during testing.
- Other detail listings that contain all the listing fields that you intend to provide to users.
- A low-level dimension (so that any member uses a small number of records that you can easily look at). For example, if the base class is Sample.Orders, you might base a dimension on OrderID. Even if you do not intend to provide this dimension to the users, you will find it useful during testing.
- The Management Portal, especially its SQL tools.
- The DeepSee Analyzer.

Before you test your model, be sure that the DeepSee indices are current.

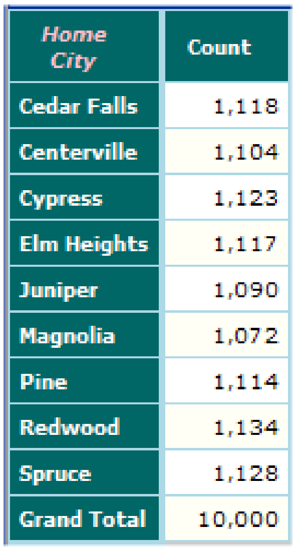
14.2 Validating a Dimension

For a dimension, it is useful to consider the following questions:

- Do the members access the expected records and no others?
- Does the dimension handle nulls in the way that you want it to?

The following procedure is a good starting point:

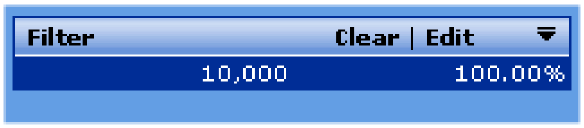
1. In the Analyzer, place the dimension on the row axis and display a grand total. Use the default measure (Count). The following example shows the home city of the patients as a dimension, using patients as the base class:



Home City	Count
Cedar Falls	1,118
Centerville	1,104
Cypress	1,123
Elm Heights	1,117
Juniper	1,090
Magnolia	1,072
Pine	1,114
Redwood	1,134
Spruce	1,128
Grand Total	10,000

DeepSee computes the grand total by adding the Count for each member.

2. Compare the grand total to the record count for this subject area. (You can see the record count in the **Filter** area on the left.)



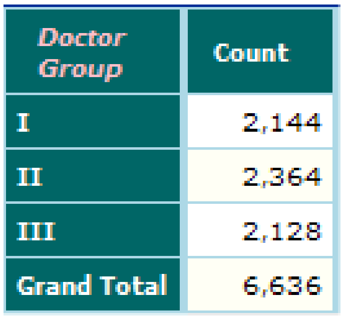
10,000 100.00%

If these numbers are not the same, make sure that you understand why they do not match.

There are two reasons why these numbers can be different:

- In some of the source records, the data used this dimension might contain null values. If you do not convert those null values to a string, this dimension essentially ignores those records. This outcome may or may not be desirable depending on your business needs.

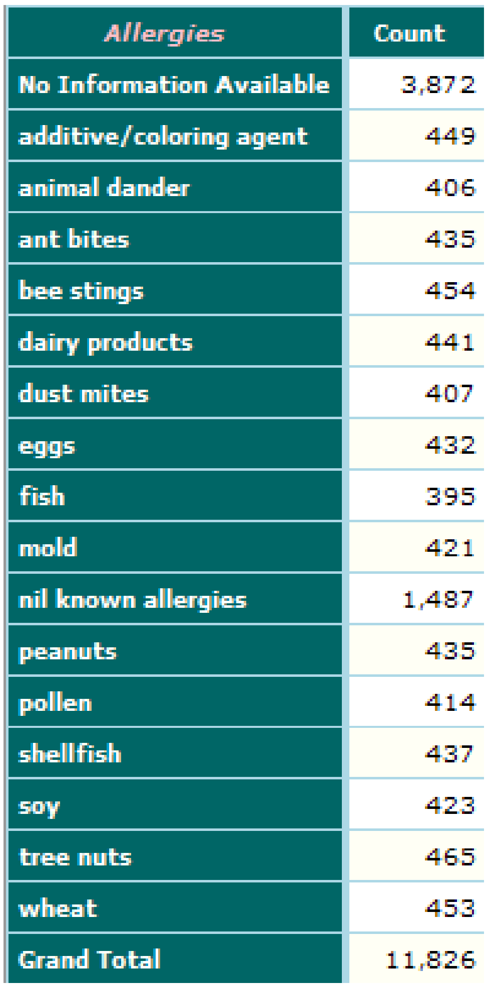
  For example, consider the following dimension, which is based on the doctor group assigned to the primary care physician that is recorded for each patient:



Doctor Group	Count
I	2,144
II	2,364
III	2,128
Grand Total	6,636

  By comparing the total shown here to the record count, we can tell that the dimension probably does not handle nulls correctly.

- If the dimension is based on a collection, any given source record can be included in multiple dimensions. The following example shows a dimension based on patient allergies, using patients as the base class:



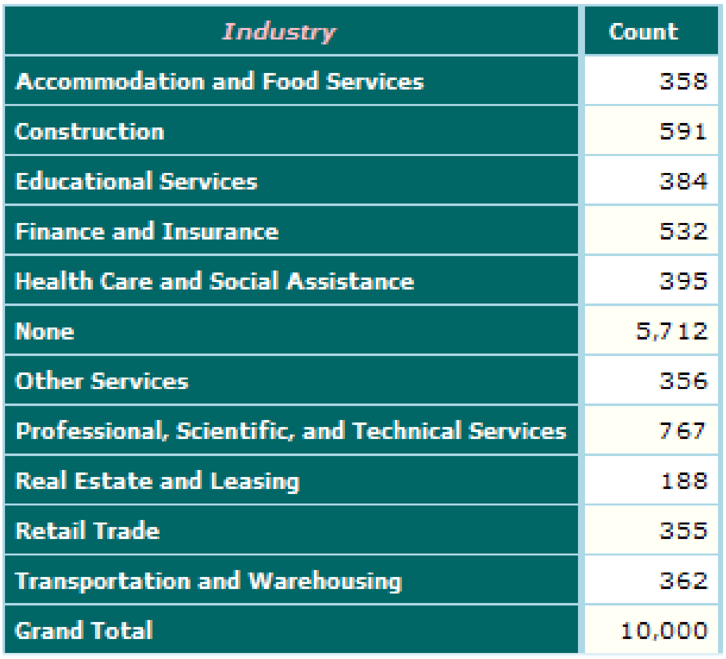
Allergies	Count
No Information Available	3,872
additive/coloring agent	449
animal dander	406
ant bites	435
bee stings	454
dairy products	441
dust mites	407
eggs	432
fish	395
mold	421
nil known allergies	1,487
peanuts	435
pollen	414
shellfish	437
soy	423
tree nuts	465
wheat	453
Grand Total	11,826

Because a patient can have multiple allergies, the total here (11826) is different from the number of patients (10000), as expected.

3. Double-check that the dimension is accessing the records that you expect.

   Look at the count for a given member. In the Management Portal, execute a suitable query to determine how many records this particular member should include. Compare that to the count shown in the pivot table.

   For example, consider the `Real Estate and Leasing` member in the following pivot table:

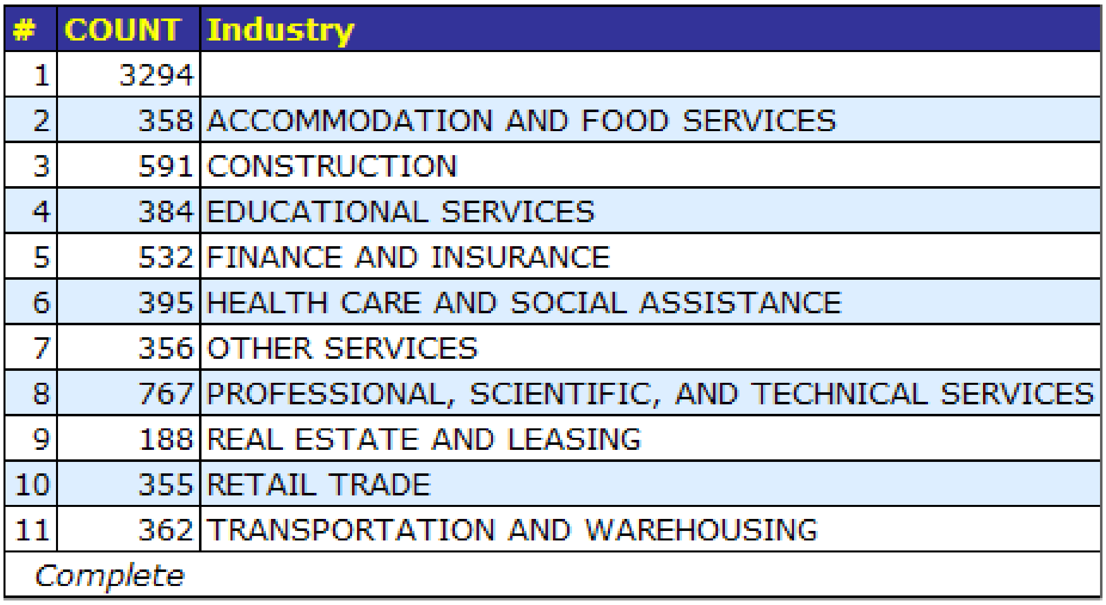


Industry	Count
Accommodation and Food Services	358
Construction	591
Educational Services	384
Finance and Insurance	532
Health Care and Social Assistance	395
None	5,712
Other Services	356
Professional, Scientific, and Technical Services	767
Real Estate and Leasing	188
Retail Trade	355
Transportation and Warehousing	362
Grand Total	10,000

In the Management Portal, you could enter an SQL query like the following:

```
SELECT Count(*) as Count, Profession->Industry FROM DeepSee_Study.PatientDetails
GROUP BY Profession->Industry
```

The results might be as follows:



#	COUNT	Industry
1	3294	
2	358	ACCOMMODATION AND FOOD SERVICES
3	591	CONSTRUCTION
4	384	EDUCATIONAL SERVICES
5	532	FINANCE AND INSURANCE
6	395	HEALTH CARE AND SOCIAL ASSISTANCE
7	356	OTHER SERVICES
8	767	PROFESSIONAL, SCIENTIFIC, AND TECHNICAL SERVICES
9	188	REAL ESTATE AND LEASING
10	355	RETAIL TRADE
11	362	TRANSPORTATION AND WAREHOUSING

Complete

There are indeed 188 patients employed in this industry.

If the counts are not the same, display a detail listing in the Analyzer and determine which records DeepSee is using. Compare this to the records in the source table.

Note: The query shows that the industry is null for 3294 patients in this table, which is fewer than the patients shown for the `None` member previously (5712). This discrepancy occurs because not all patients have records in the PatientDetails table.

14.3 Validating a Measure

For a measure, particularly a measure based on **Complex Code**, it is useful to consider the following questions:

- Does the measure return the correct value at the lowest level?
- Is the measure aggregated as expected to higher levels?

The following procedure is a good starting point:

1. In the Analyzer, place the lowest-level dimension on the row axis. For **Metrics**, use the measure you are testing and the Count measure.
2. Find a member that has a fairly low value for Count.
3. Display the detail listing for this member.
4. For each measure, look at a single record and check that the value shown in the detail listing is as expected.
5. If the lowest-level values for a measure are as expected, aggregate those values for all the records in this detail listing. Use the aggregation type specified in the measure. Compare the resulting number to the number shown for this measure in the pivot table.

14.4 Validating a Listing Field

For a listing field, it is useful to consider the following questions:

- Does the listing field display the correct value?

  Are the appropriate transformations, ranges, or replacements applied?
- Does the listing field display the correct value when the source data is null?

Because listing fields are directly associated with source records and because they are not aggregated in any way, it is a simpler process to validate them.

1. In the Analyzer, use any dimensions and measures, and display a detail listing.
2. Examine each listing field and consider the preceding questions.

A

When to Recompile and Rebuild

For your convenience, this appendix summarizes when to recompile and when to rebuild indices, in order to make the changes visible in the Analyzer and in pivot tables.

Item Added, Modified (Including Renaming), or Deleted	Need to Recompile?	Need to Rebuild Indices?
Dimension	*Yes*	*Yes*
Compound dimension or member (if indexed)	*Yes*	*Yes*
Compound dimension or member (if dynamic)	No	No
Dynamic dimension	*Yes*	No
Relationship between dimensions	*Yes*	*Yes*
Subject area	No	No
Dimension-type listing field	*Yes*	*Yes*
Independent listing field	*Yes*	No
Detail listing, any change other than role access	*Yes*	No
Detail listing, changing role access	No	No
Transform used by dimension or dimension-type listing field	*Yes*	*Yes*
Transform used by independent listing field	*Yes*	No

For information on recompiling and rebuilding, see the section "Recompiling and Rebuilding," earlier in this book.

Index