Overview of DeepSee

Version 2011.1
26 May 2011

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

Overview of DeepSee
Caché Version 2011.1 26 May 2011

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.

InterSystems CACHÉ and InterSystems ENSEMBLE

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

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 introduces InterSystems DeepSee, which enables you to embed business intelligence (BI) into your applications. The purpose of this book is to describe the capabilities that you can add to your applications and to give a detailed look at the results that you can provide to your users. It defines the key concepts, introduces the DeepSee components that you use during different phases of development, and outlines how you use these components together.

This book is written primarily for implementers who are responsible for using InterSystems DeepSee. The first few chapters, however, may also be useful for anyone else who wants to become acquainted with InterSystems DeepSee.

This book contains the following chapters:

- Introduction to InterSystems DeepSee
- Concepts
- How DeepSee Works
- Introduction to the Toolkit
- Available Options

For a detailed outline, see the table of contents.

For more information, see the following books:

- *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 Architect*, a guide for implementers who are setting up a DeepSee model for use in the Analyzer.
- *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 InterSystems DeepSee

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 DeepSee. It discusses the following topics:

- Purpose of DeepSee
- Architecture of applications that use DeepSee
- Pivot tables
- Charts
- Detail listings
- Dashboards
- Runtime values passed to DeepSee
- Custom code in DeepSee
- Security
- Programmatic access to data

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

Because it is important to understand the options available to your users, this chapter summarizes the key options. The *DeepSee User Guide* discusses the options more thoroughly; this book is provided to supplement any material you might prepare for your users.

1.1 Purpose

The purpose of InterSystems DeepSee is to enable you to embed business intelligence (BI) into your applications so that your users can ask and answer sophisticated questions of their data. Your application can include *dashboards*, which can include any or all of the following:

- DeepSee *pivot tables*, which are interactive, drillable displays of data, designed for specific user roles or for specific areas of your user interface.
- Controls such as date choosers that affect what the dashboard displays.
- *Speedometers*, which display *KPIs* (key performance indicators).

- Buttons and drop-down menus that enable the user to access other dashboards.
- DeepSee tools such as the DeepSee Analyzer for advanced end users, to enable them to explore their data in an open-ended manner.

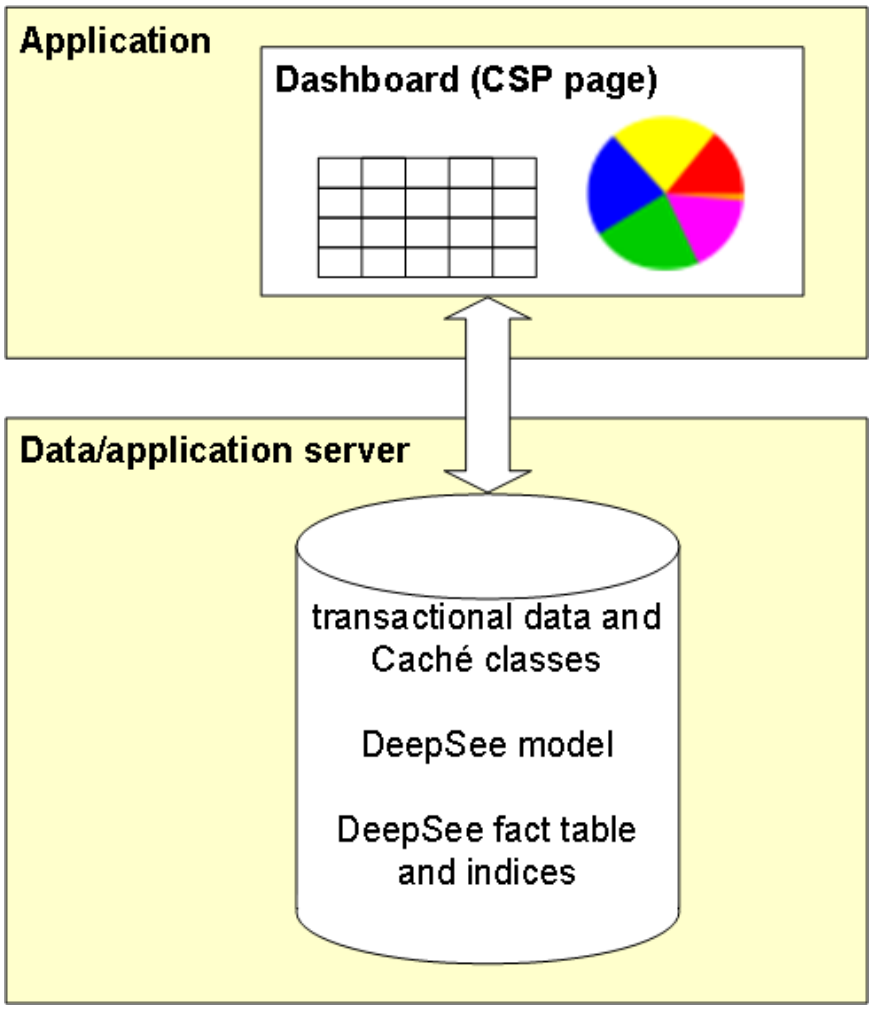
In contrast to traditional BI systems that use static data warehouses, DeepSee uses the live transactional data.

1.2 Architecture

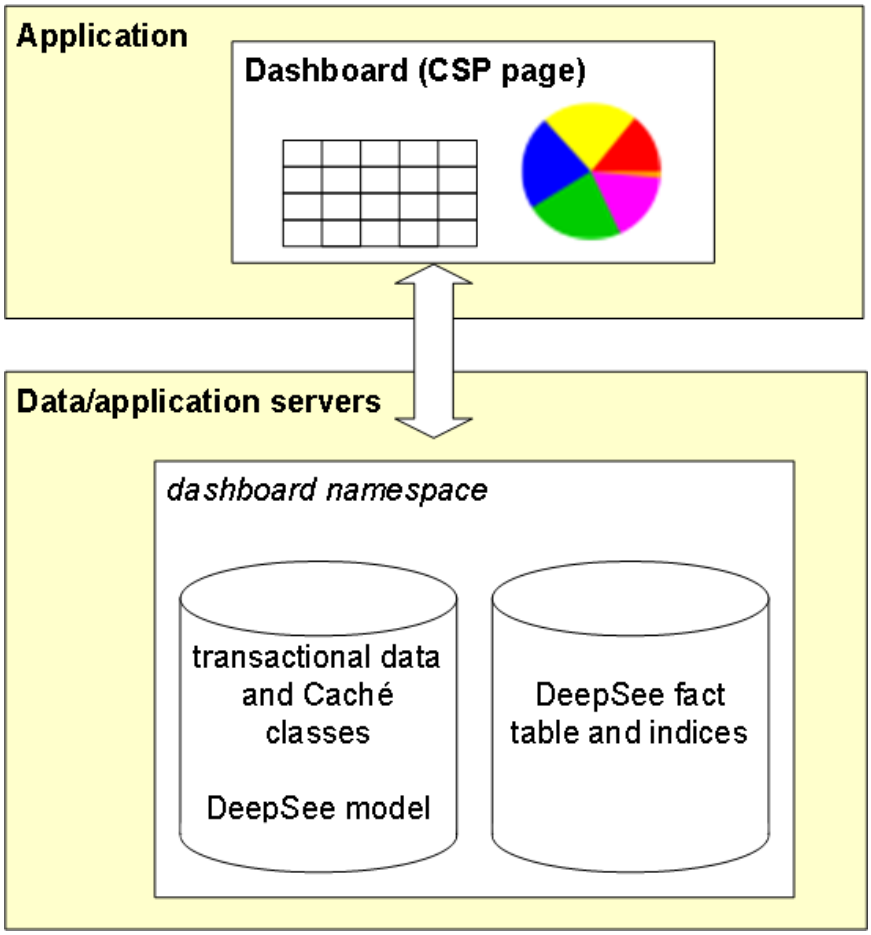
In the typical scenario, dashboards are added to an existing application and make use of the same data that the rest of the application uses. (It is also possible for the dashboards to execute against a shadow of the data, if that is necessary for some reason.)

Most dashboards contain at least one pivot table or other DeepSee data element. DeepSee data elements work by accessing the DeepSee indices and fact table. The *fact table* is an internal structure defined by a DeepSee model. When the DeepSee indices are built, DeepSee updates both the fact table and indices, making them available for runtime use. Similarly, when data changes, DeepSee detects the change and updates the corresponding parts of the fact table and indices. (For information on how DeepSee uses the model, fact table, and indices, see the chapter "How DeepSee Works.")

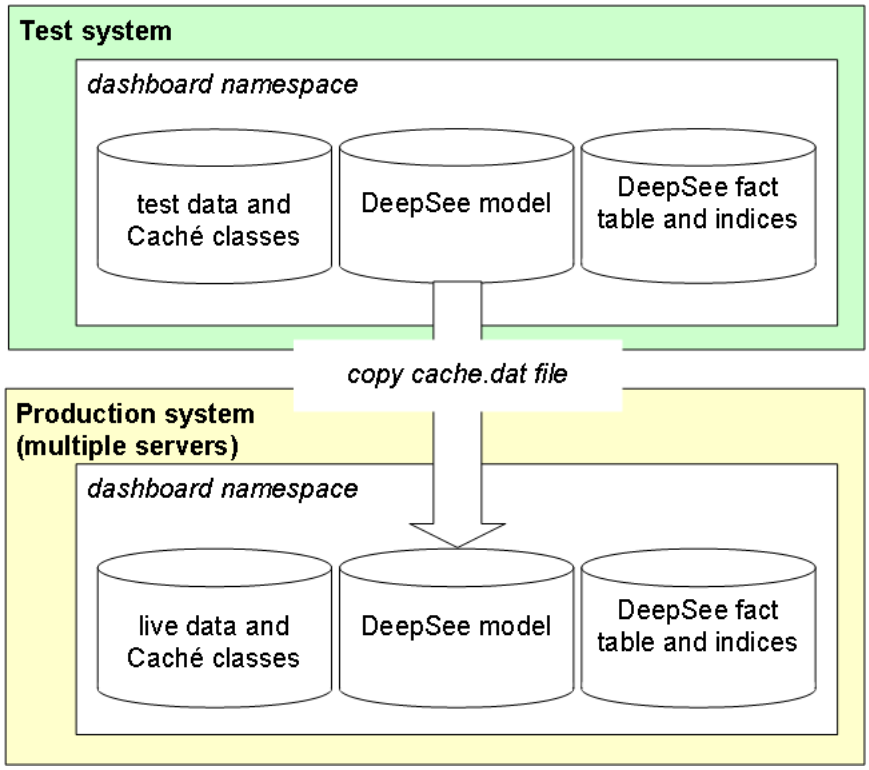
In the simplest scenario, the Caché classes and their data, the DeepSee model, and the DeepSee fact table and indices reside within one Caché database on a server. The application runs on another server, and the dashboards communicate with the DeepSee data as needed. The following shows a generic example:



When the volume of data is large, it is often desirable to store the DeepSee indices in a separate database, possibly on another server:



In the most general case, the DeepSee model is stored in another database:



This simplifies deployment — to move the model from a test system to a production system, you can copy the cache.dat file (and then recompile the base class). It is not necessary to configure DeepSee in this way, however, because DeepSee does provide options for exporting and importing the model and other metadata.

Finally, in another variation, DeepSee can use data that is imported periodically from an external system. DeepSee provides a tool (the Connector) that enables you to easily create Caché classes that model external data and that define data loading and cleaning scripts that you can execute as needed. Another tool (the Scheduler) enables you to automate data loading and index building.

1.3 Pivot Tables

Pivot tables are central to InterSystems DeepSee; they select and aggregate data and display it in an interactive format.

In InterSystems DeepSee, the phrase *pivot table* is a generic phrase that can mean a tabular display of data, a graph of the same data, or a combination format. We start by looking at pivot tables in table format.

The following figure shows an example pivot table, in table format. It shows the number of patients and the average allergy count per patient, grouped by age and gender.

		Female		Male	
Age Group	*Age Bucket*	Patient Count	Average Allergy Count	Patient Count	Average Allergy Count
0-29	0-9	661	0.79	688	0.79
	10-19	705	0.85	728	0.80
	20-29	692	0.83	682	0.82
30-59	30-39	800	0.76	776	0.82
	40-49	728	0.81	757	0.76
	50-59	568	0.77	535	0.80
60+	60-69	389	0.89	308	0.84
	70-79	368	0.77	261	0.77
	80+	235	0.86	119	0.80

Because the DeepSee concepts are interrelated, making it impossible to discuss each concept without reference to the others, it is useful for us to start with preliminary definitions:

- A *dimension* is a construct that DeepSee uses when it groups the source data. A dimension has *members*. Each member, in turn, corresponds to a specific set of records in the source data.

  For example, the `Age Group` dimension has the members `0-29`, `30-39`, and `60+`. The `Age Bucket` dimension has the members `0-9`, `10-19`, `20-29`, and so on. The `Gender` dimension has the members `Female` and `Male`.

- A *measure* is a value displayed in the body of the pivot table; it is based on values in the source data. For a given context, a measure aggregates the values for all applicable source records and represents them with a single value.

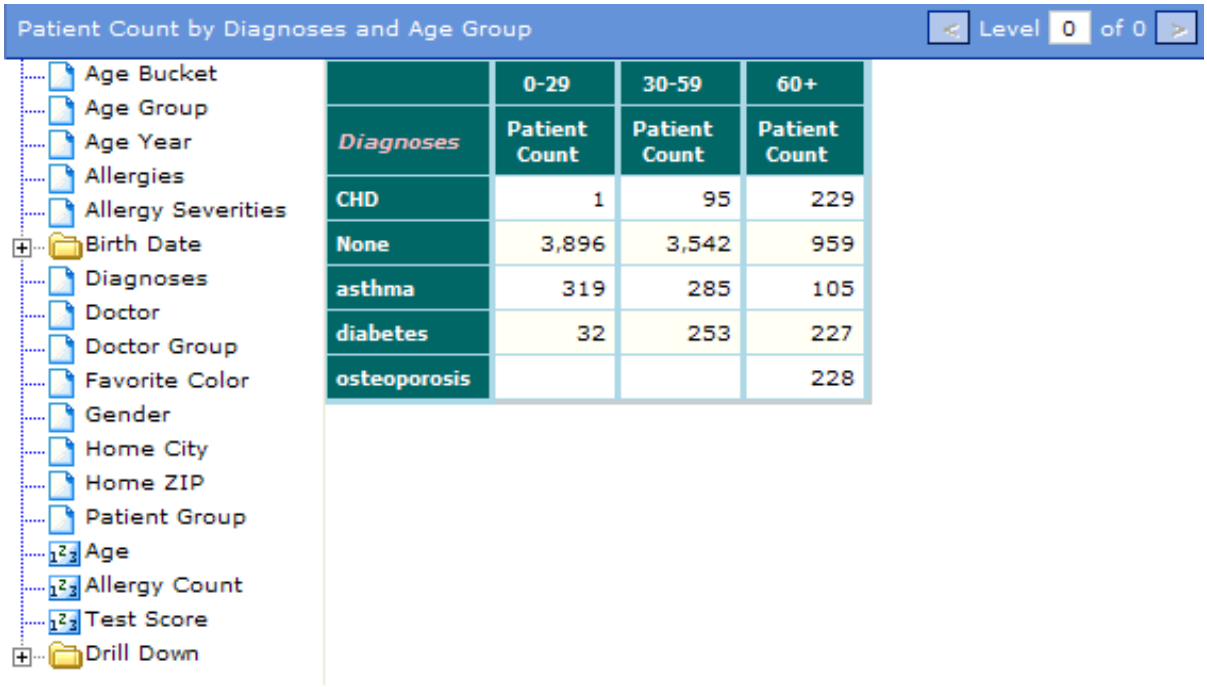
  For example, the measure `Patient Count` is the number of patients, and the measure `Average Allergy Count` is the average number of allergies per patient.

1.3.1 Key User Options

A pivot table is an interactive display of data. At runtime, users can analyze, sort, and export the data; many of these tasks are similar to what can be performed in spreadsheets and are not discussed here.

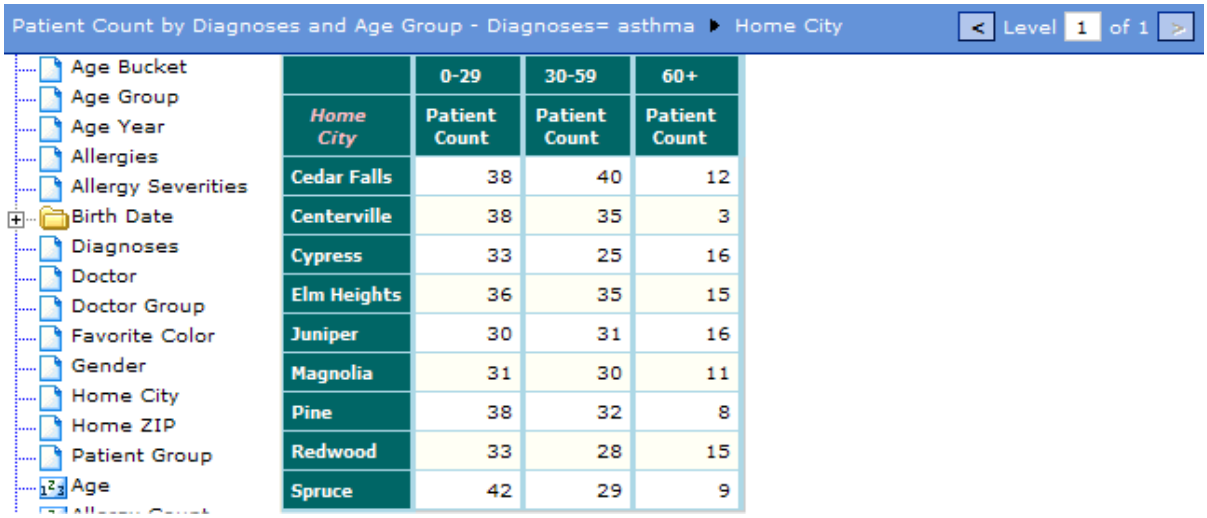
The analysis options, however, are unique to DeepSee. This section summarizes these options briefly. Users can do the following:

- *Drill through* to see a selected row with a breakout by a different dimension. First, the user clicks a button to display the drill options list, which appears on the left side of the pivot table:



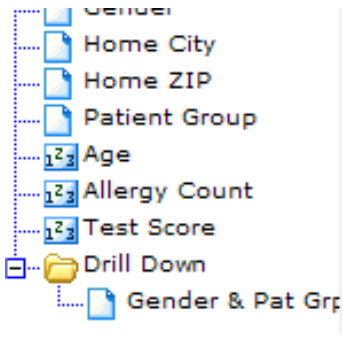
	0-29	30-59	60+
Diagnoses	Patient Count	Patient Count	Patient Count
CHD	1	95	229
None	3,896	3,542	959
asthma	319	285	105
diabetes	32	253	227
osteoporosis			228

The drill options list shows the dimensions that are available for drill through. The user drags and drops a dimension (for example, `Home City`) from this list onto a row (for example, `asthma`) of the pivot table. DeepSee then displays a closer look at that row, broken out as requested. For example:

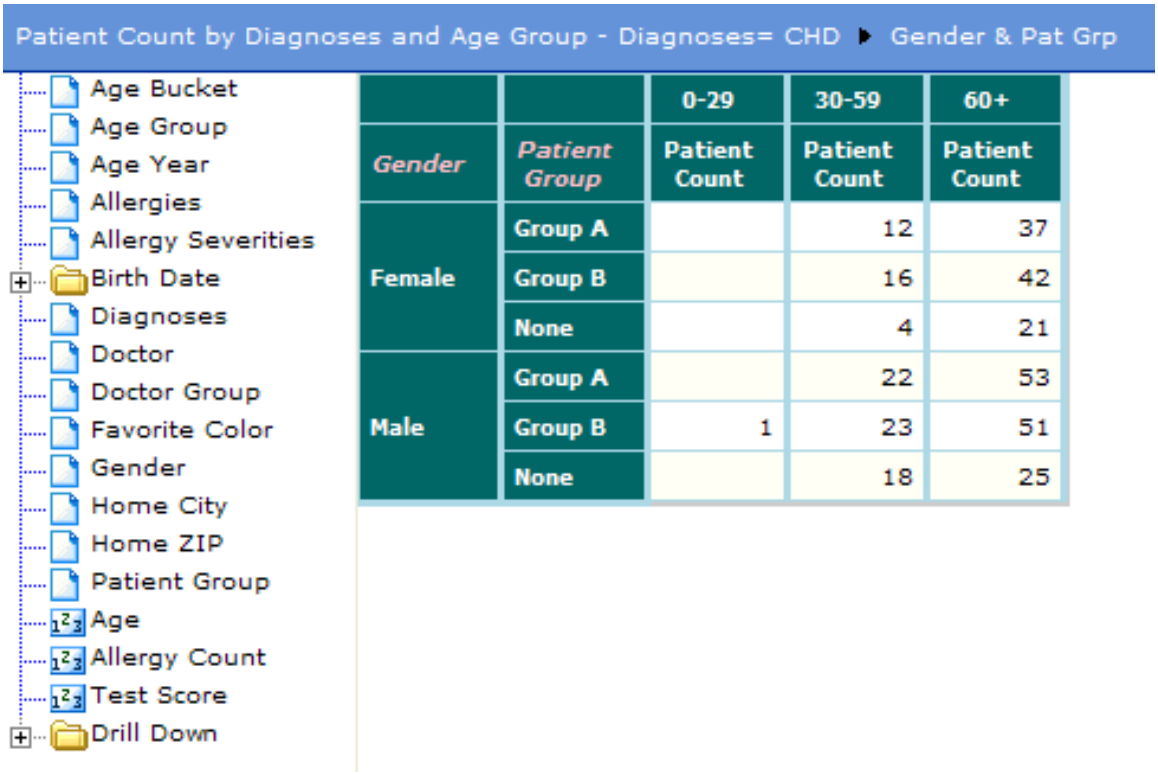


	0-29	30-59	60+
Home City	Patient Count	Patient Count	Patient Count
Cedar Falls	38	40	12
Centerville	38	35	3
Cypress	33	25	16
Elm Heights	36	35	15
Juniper	30	31	16
Magnolia	31	30	11
Pine	38	32	8
Redwood	33	28	15
Spruce	42	29	9

- *Drill down*. This task is similar to the preceding task, except that the user drags and drops a drill-down rather than a single dimension. A drill-down is a named combination of multiple dimensions. To see the drill-downs, the user expands the **Drill Down** folder at the bottom of the drill options list. For example:



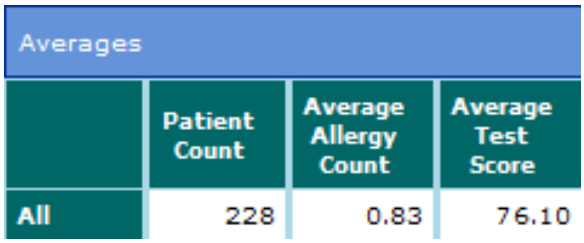
Suppose that the user drags the `Gender & Pat Grp` drill-down and drops it into the `CHD` row. The following shows a possible result:



		0-29	30-59	60+
Gender	*Patient Group*	Patient Count	Patient Count	Patient Count
Female	Group A		12	37
	Group B		16	42
	None		4	21
Male	Group A		22	53
	Group B	1	23	51
	None		18	25

In contrast to drilling through, DeepSee displays a breakout by multiple dimensions, in this case `Gender` and then `Patient Group`.

- Right-click a row in the pivot table and then open a *related pivot table*, which is filtered by the context from which the user started. For example, if the user right-clicks the `osteoporosis` row in the original table and then clicks `Drill Down to Averages (By Row)`, DeepSee displays the following in another window in front of the original pivot table:



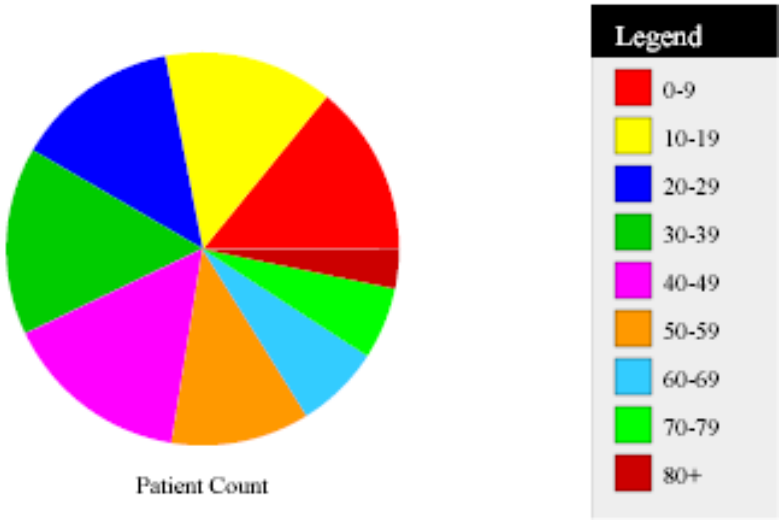
	Patient Count	Average Allergy Count	Average Test Score
All	228	0.83	76.10

Notice that in this case, DeepSee displays a different set of measures than the original pivot table. In contrast, the drill-through and drill-down actions both displayed the same measures as the original pivot table.

- Display a *detail listing* that shows the underlying source records, which is valuable especially when trying to understand an unexpected result. A user right-clicks and selects **Detail Listing**, and DeepSee then displays the detail listing. See the section "Detail Listings," later in this chapter.

1.4 Pivot Tables in Chart Format

A pivot table can be displayed as a chart, in a wide variety of styles:



1.4.1 Key User Options

In a chart, users can perform many tasks, including the following:

- Display details in hover text.
- Change the chart type and modify properties of the chart.
- Display detail listings.
- Switch the display to table format.
- Drill down by double-clicking (depending on how the system is configured).

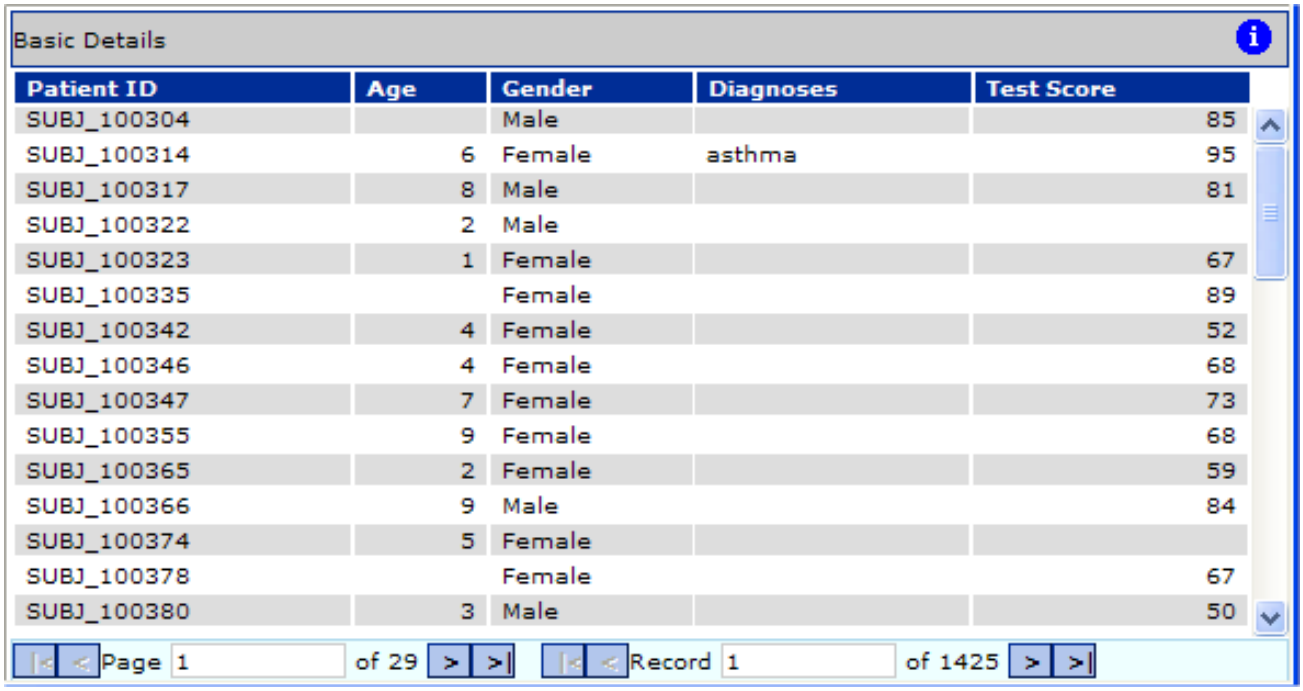
1.5 Detail Listings

Detail listings display selected fields for the underlying source records in the current context. Users can see detail listings in the following ways:

- By right-clicking in a pivot table and selecting **Detail Listing**. DeepSee then displays the detail listing, which is filtered by the context in which the user clicked.

- By right-clicking in a speedometer in a dashboard and selecting **Detail Listing**. DeepSee then displays the detail listing, which is filtered by the context in which the user clicked.
- By accessing a dashboard that displays a detail listing directly. Depending on the design of the dashboard, the detail listing may be filtered by other elements on that dashboard, may have a filter of its own, or both.

Detail listings are valuable especially when trying to understand an unexpected result. The following shows an example:

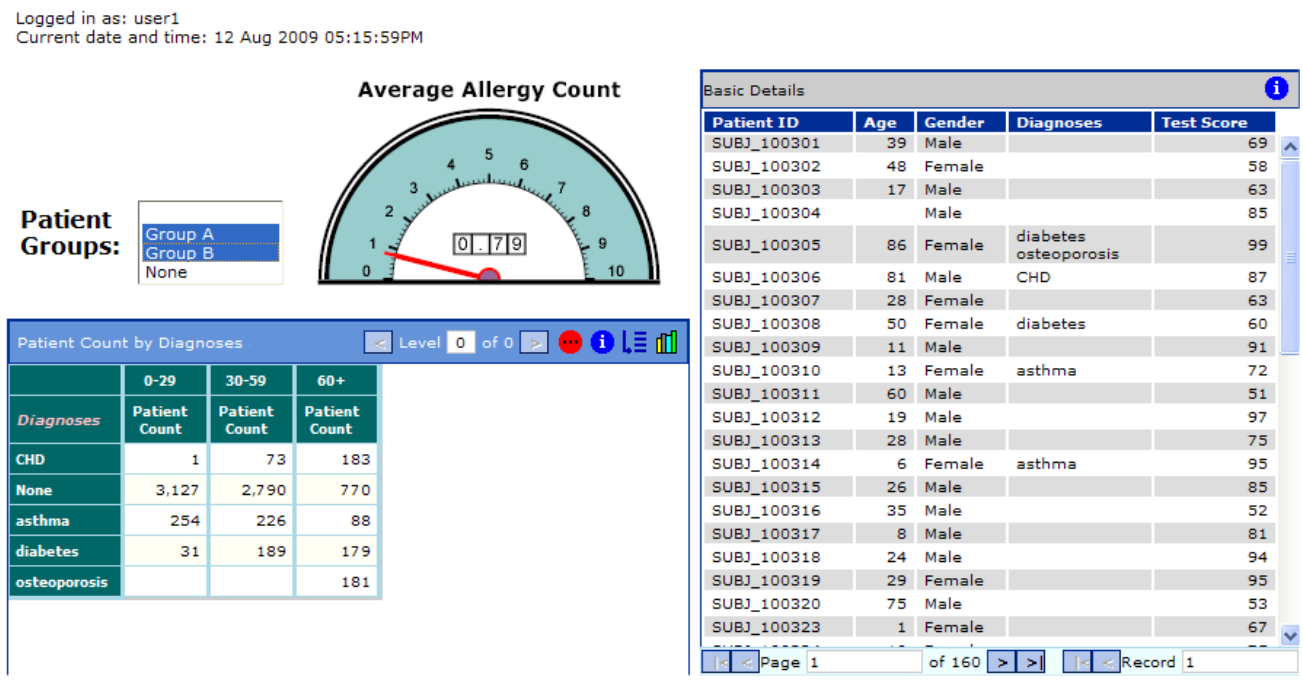


Basic Details

Patient ID	Age	Gender	Diagnoses	Test Score
SUBJ_100304		Male		85
SUBJ_100314	6	Female	asthma	95
SUBJ_100317	8	Male		81
SUBJ_100322	2	Male		
SUBJ_100323	1	Female		67
SUBJ_100335		Female		89
SUBJ_100342	4	Female		52
SUBJ_100346	4	Female		68
SUBJ_100347	7	Female		73
SUBJ_100355	9	Female		68
SUBJ_100365	2	Female		59
SUBJ_100366	9	Male		84
SUBJ_100374	5	Female		
SUBJ_100378		Female		67
SUBJ_100380	3	Male		50

Page 1 of 29 > >| Record 1 of 1425 > >|

For another example, the following dashboard includes a directly embedded detail listing:



Logged in as: user1
Current date and time: 12 Aug 2009 05:15:59PM

Patient Groups:

Group A
Group B
None

Average Allergy Count

Patient Count by Diagnoses — Level 0 of 0

Diagnoses	0-29 Patient Count	30-59 Patient Count	60+ Patient Count
CHD	1	73	183
None	3,127	2,790	770
asthma	254	226	88
diabetes	31	189	179
osteoporosis			181

Basic Details

Patient ID	Age	Gender	Diagnoses	Test Score
SUBJ_100301	39	Male		69
SUBJ_100302	48	Female		58
SUBJ_100303	17	Male		63
SUBJ_100304		Male		85
SUBJ_100305	86	Female	diabetes osteoporosis	99
SUBJ_100306	81	Male	CHD	87
SUBJ_100307	28	Female		63
SUBJ_100308	50	Female	diabetes	60
SUBJ_100309	11	Male		91
SUBJ_100310	13	Female	asthma	72
SUBJ_100311	60	Male		51
SUBJ_100312	19	Male		97
SUBJ_100313	28	Male		75
SUBJ_100314	6	Female	asthma	95
SUBJ_100315	26	Male		85
SUBJ_100316	35	Male		52
SUBJ_100317	8	Male		81
SUBJ_100318	24	Male		94
SUBJ_100319	29	Female		95
SUBJ_100320	75	Male		53
SUBJ_100323	1	Female		67

Page 1 of 160 > >| Record 1

The next section discusses dashboards in more detail.

1.5.1 Key User Options

In a detail listing, users can perform tasks such as the following:

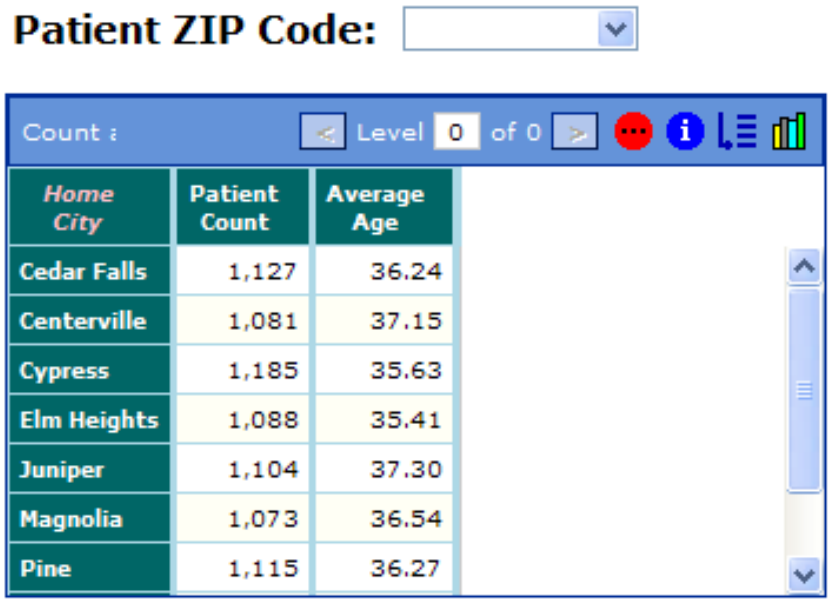
- Navigate through the pages, if there are multiple pages.
- Display details about the detail listing and about a selected cell.
- Right-click and export the data.

1.6 Dashboards

To include a pivot table in your application, you package it up in a dashboard, which is a CSP page. Dashboards can also include detail listings, speedometers, images, and text.

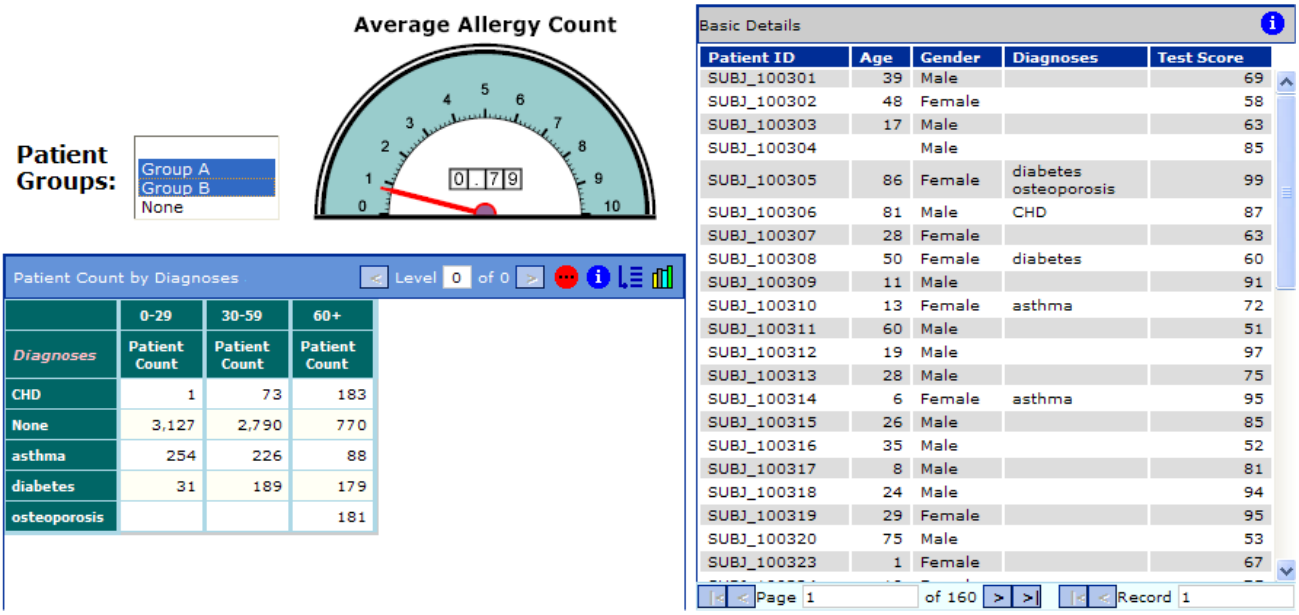
You then include these CSP pages in your application.

A dashboard can be as simple or as complex as needed. First let us consider a rather simple one:



To use this dashboard, a user can click a ZIP code, which instantly filters the pivot table to show only patients in that ZIP code.

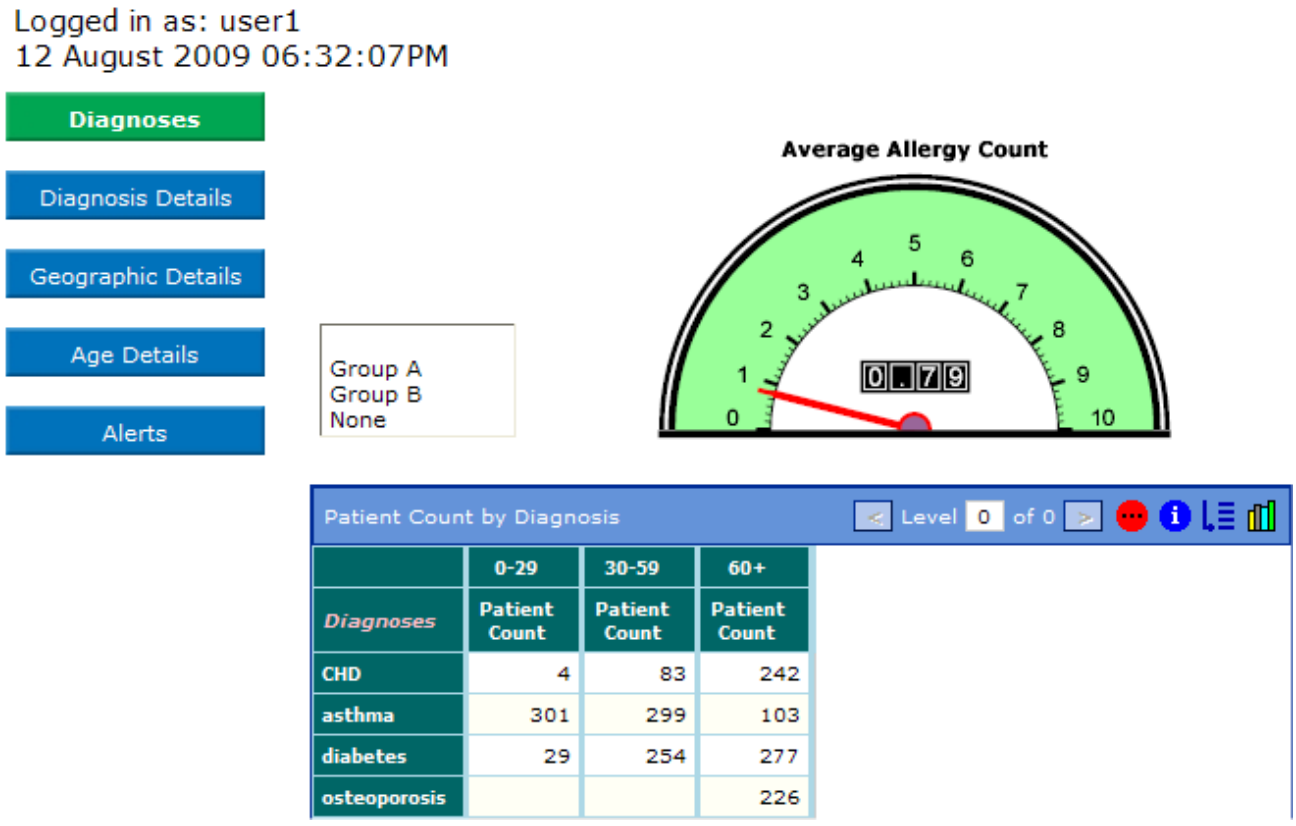
For another example:



In this case, the user has selected two patient groups, and the rest of the dashboard displays data for those patients.

1.6.1 Frames

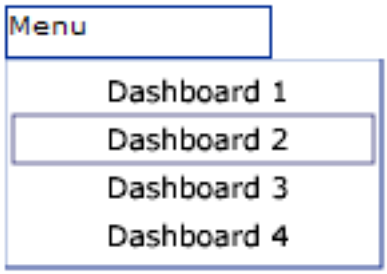
A dashboard can be configured to include frames, which can display Web pages or other dashboards. In this case, the main dashboard typically has buttons or other controls that specify what to display in the frame. For example:



Diagnoses	0-29 Patient Count	30-59 Patient Count	60+ Patient Count
CHD	4	83	242
asthma	301	299	103
diabetes	29	254	277
osteoporosis			226

In this example, users can click the buttons on the left, and this controls which dashboard is displayed on the right.

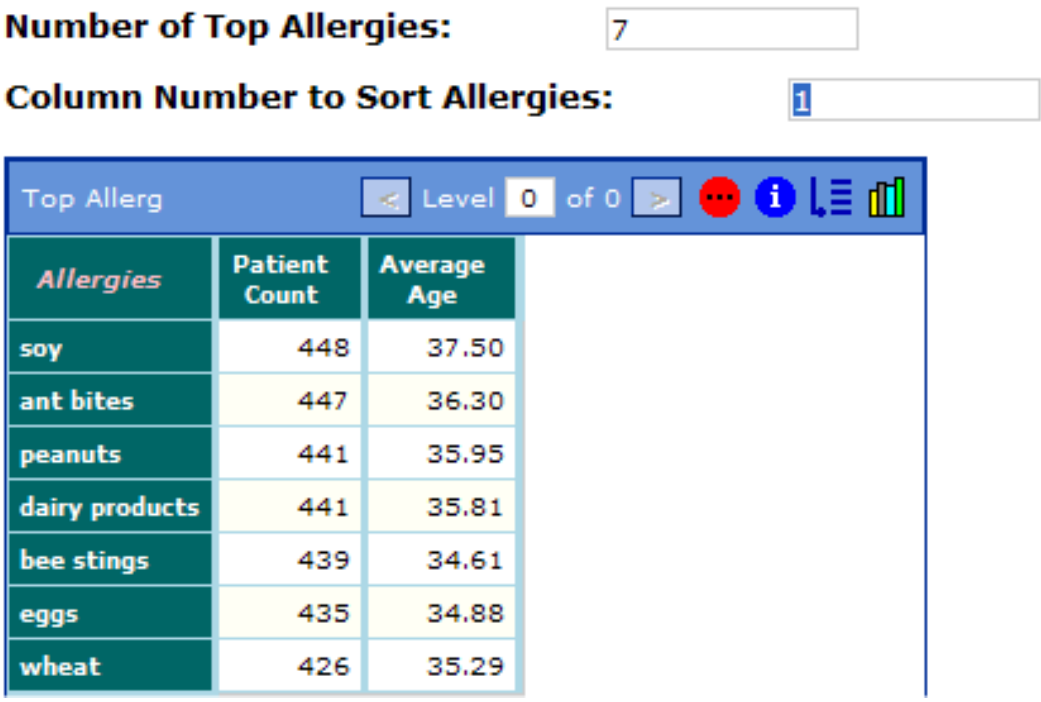
In other cases, the dashboard may include drop-down menus (buttons that contain menu choices), which control the contents of the frame. The following shows an example of a drop-down menu:



1.6.2 Key User Options

Depending on the configuration of a dashboard, a user can perform many tasks, including the following:

- Select values from or enter values into controls that affect the details of a pivot table, such as how many items are shown. For example:



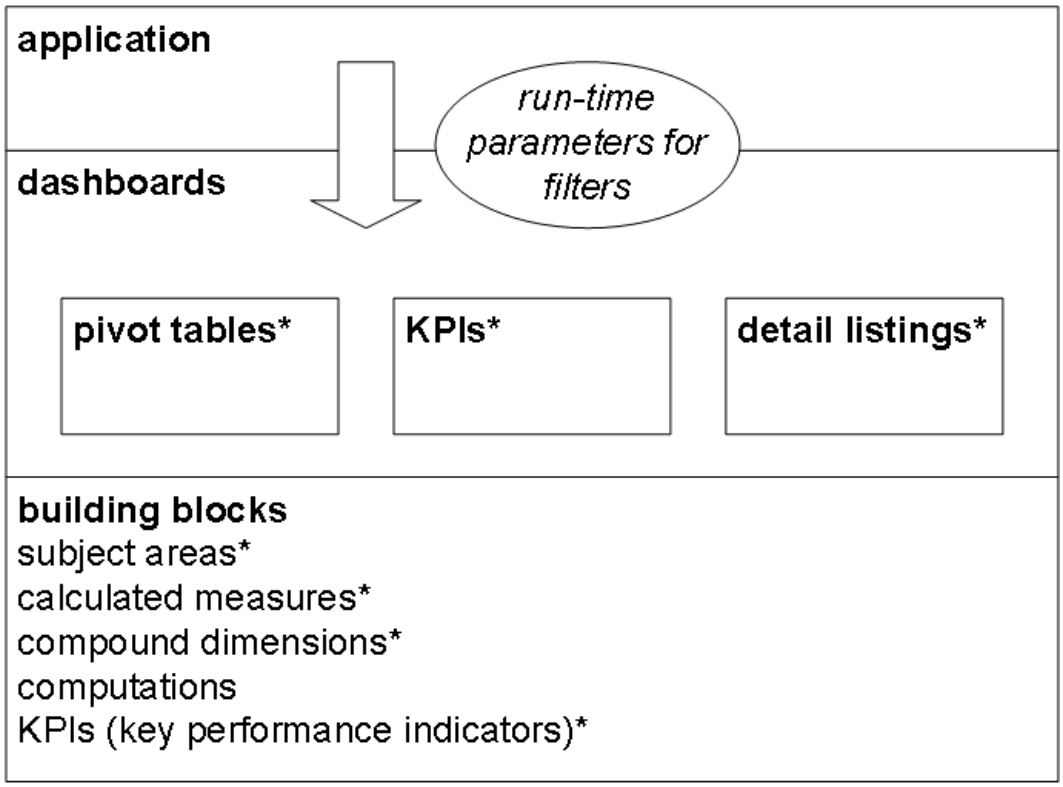
Allergies	Patient Count	Average Age
soy	448	37.50
ant bites	447	36.30
peanuts	441	35.95
dairy products	441	35.81
bee stings	439	34.61
eggs	435	34.88
wheat	426	35.29

- Right-click and select options for the dashboard. These options include the ability to open other dashboards, save and reload selections in the current dashboard, refresh the data, and access DeepSee tools. (The roles to which a user belongs determine the options that the user can actually execute.)
- Right-click in any pivot table, chart, or any detail listing and then select options appropriate for that element, as described earlier.
- Click an element and launch a post action. A post action can do any of the following:
  - Display a dashboard, possibly in a new window.
  - Display a Web page in another browser window.

- Display a small child window that displays the value of a KPI.
- Execute a custom script.

1.7 Runtime Parameters

DeepSee provides the ability to pass runtime values from your application into the dashboards. The values you pass in can be used in filters throughout DeepSee. The following figure shows the areas that you can affect at runtime by this mechanism:



*can use filters

This picture is intended to give you an idea of the pervasiveness of filter expressions and the ability to pass runtime values to them. These terms are defined in the next chapter.

1.8 Custom Code

You can include Caché ObjectScript expressions in many places throughout DeepSee. Some expressions are evaluated when the indices and fact table are built or updated. Other expressions are evaluated at runtime. In particular, all filters can include custom expressions, and these are evaluated at runtime.

Even if you are not going to perform this programming yourself, it is useful to be aware of the options that the language provides. These include the following:

- Mathematical, string, and date operations

- Pattern matching
- Logical operations, in particular, operations that evaluate a condition and select a value
- Flow control commands and functions such as **If-Then**, **$CASE**, **$SELECT**

1.9 Security

In DeepSee (as in the rest of Caché and Ensemble), access to data and to functions is secured by roles, although you define DeepSee users and roles within a DeepSee configuration tool rather than within the Management Portal.

The roles to which a user belongs determine the user's access or ability as follows:

- Roles control whether the user can use pivot tables at all.
- Roles determine the subject areas that the user can see. A subject area is a subset of data that has access to some or all of the dimensions defined for that area. For example, a subject area could consist of all lab tests, all lab tests for a specific hospital, or all lab tests of a given type.
- Roles determine the detail listings that the user can see.
- Roles control whether the user can use the different components of DeepSee, and if so, which options are available. For information, see the chapter "Introduction to the Toolkit."

1.10 Programmatic Access to Data

Via the %BI.Utils class, you can run pivot tables programmatically and access data in them. This enables you to perform background processing to check on values and react accordingly, for example.

With the same class, you can access the value of any *KPI* (key performance indicator), including any filtering that you specify.

2

Concepts

This chapter defines the terms and key concepts in DeepSee. It includes the following topics:

- Introduction
- Core model
- Filter expressions
- External building blocks
- Using multiple models together

The following chapter ("How DeepSee Works") describes how DeepSee uses the elements described here. A later chapter ("Available Options") provides details on the options related to the elements.

2.1 Introduction

In DeepSee, you create and then use a model of your data. In most of the DeepSee documentation, the term *model* is used in a generic way, meaning "representation of your data." In this chapter and in the rest of the book, however, it is important to make a distinction:

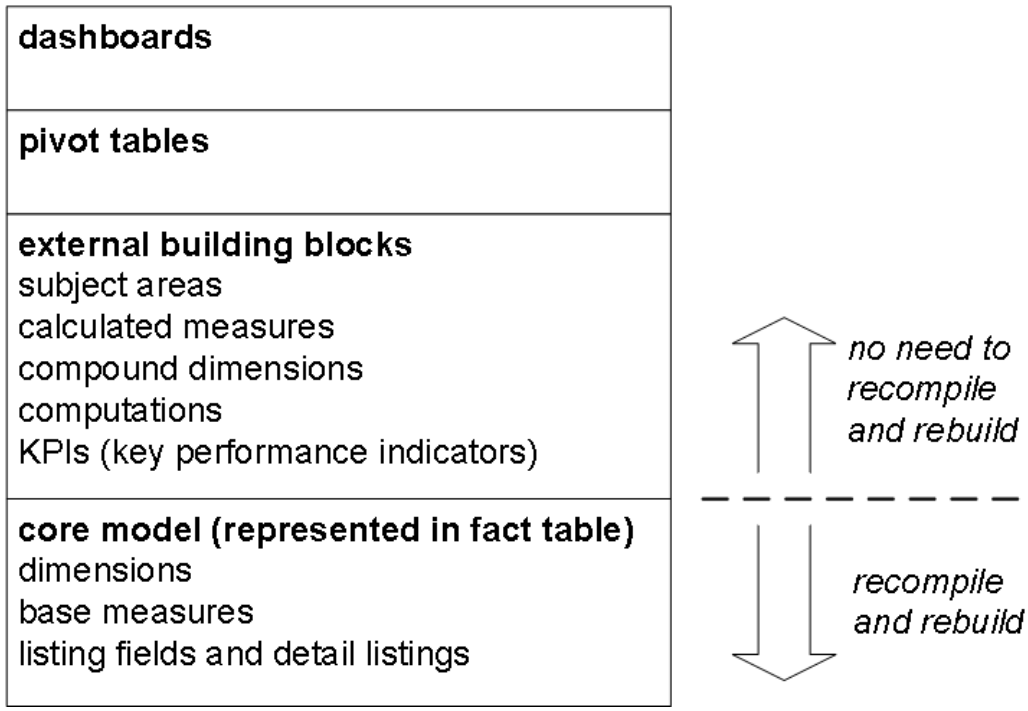
- The *core model* is represented in the fact table and indices. After making any change to this, it is necessary to recompile and rebuild, which is typically time-consuming.
- Outside of the core model, users can create *building blocks* that extend the core model.

  You can consider the combined core model and building blocks as the DeepSee "model."

Typically, a superuser or developer creates the core model and some building blocks during a model development phase. In most cases, it is necessary to recompile and rebuild fairly frequently during this phase.

Then other users extend the model by adding building blocks as needed for their pivot tables and dashboards, which they also create.

The following figure summarizes the kinds of changes that can be made:



The following sections explain these elements.

2.2 Core Model

A DeepSee core model is based on a base table and consists the following elements:

- *Dimensions*, which enable you to group and select data at different levels.
- *Measures*, which contain data that can be aggregated to any level.
- *Detail listings*, which display selected fields of the lowest-level data.
- *Listing fields*, which are the fields used in detail listings.

These elements are based, directly or indirectly, on fields of your tables.

This section discusses a single core model, but in practice you might define multiple models, which you can use in combination.

2.2.1 Base Table

Each DeepSee model is based on a *base table*. (DeepSee assumes that your data is stored in tables and is modeled with equivalent Caché classes. These tables can be your live transaction tables.) Your choice of base table affects the meaning of every measure and every pivot table.

A DeepSee model analyzes a set of entities. The basic entity corresponds to a row in the base table that you choose for the model. The base table affects what you see in all pivot tables. This concept is best explained with an example. Consider the following pivot table:

	Female		Male	
Allergies	Patient Count	Average Age	Patient Count	Average Age
additive/coloring agent	201	38.22	211	33.81
animal dander	165	37.19	200	34.35
ant bites	198	37.74	213	30.92
bee stings	231	36.08	213	32.42
dairy products	203	37.78	230	33.91
dust mites	207	35.72	204	33.96
eggs	217	36.94	215	37.78
fish	221	35.91	215	35.80

This pivot table uses a DeepSee model that uses `Patients` as the base table. Thus, each data cell in this pivot table represents a set of patients. For example, the row labeled `ant bites` shows data for patients who are allergic to ant bites. In this row, the first two cells show data for female patients who are allergic to ant bites, and the last two cells show data for male patients who are allergic to ant bites.

In any given data cell, the value shown is aggregated across the relevant set. In the `ant bites` row, the cell `Female>Patient Count` shows the total count of female patients who are allergic to ant bites. In the same row, the cell `Female>Average Age` shows the average age of that set of patients.

2.2.2 Dimensions and Members

In DeepSee, a dimension consists of a set of members. Each record in the base table is associated with zero, one, or more members of the dimension. The members enable you to select different sets of records.

The pivot table earlier in this chapter used two dimensions: `Gender` and `Allergies`.

- The `Gender` dimension has two members: `Female` and `Male`. Each patient is associated with (or belongs to) one of these members.
- The `Allergies` dimension has members such as `additive/coloring agent`, `animal dander`, and so on. For this dimension, any patient can belong to multiple members.

Each dimension is based on source data. For example, the `Gender` dimension is based on a property of the patient.

When you use a dimension as a row or a column of a pivot table, DeepSee displays a breakout that shows each member of that dimension. In the previous example pivot table, the `Allergies` dimension is used as rows. This dimension has one member for each allergen, and these are shown as rows. Similarly, the `Gender` dimension is used as columns. This dimension has two members, and these are shown as columns.

2.2.3 Date Dimensions

Dimensions based on dates are special, because in addition to the dimension you create, DeepSee creates a set of variants that enable you to analyze in multiple ways by date.

For example, if you create a date-type dimension called `Birth Date`, DeepSee creates that dimension and the following variants:

- `Birth Date Day`
- `Birth Date Month`

- Birth Date Period
- Birth Date Quarter
- Birth Date Week
- Birth Date Week of Month
- Birth Date Weekday
- Birth Date Year

2.2.3.1 Plain Date Dimension

The plain Birth Date dimension is special. It cannot be used for rows or columns as ordinary dimensions are. You can, however, use it in a filter expression. See "Filter Expressions," later in this chapter.

2.2.3.2 Date Dimension Variants

Most of the date dimension variants use only a single part of the date. For example, the Birth Date Year variant uses only the year part of the date:

Birth Date Year	Patient Count
1910	3
1911	8
1912	5
1913	4

Here, the 1910 member displays all patients born in 1910.

Similarly, Birth Date Quarter uses only the quarter number of the date:

Birth Date Quarter	Patient Count
Q1	2,492
Q2	2,501
Q3	2,525
Q4	2,481

Here, the Q1 member displays all patients born in the first quarter of the year, in any year.

The Birth Date Period dimension uses both the year and the month parts of the date. For example:

Birth Date Period	Patient Count
1910-03	1
1910-05	1
1910-12	1
1911-03	2
1911-06	1

2.2.3.3 Date Dimensions Used in Combination

You can use these dimensions in combination, in the same way that you do with any other dimensions:

	Q1	Q2	Q3	Q4
Birth Date Year	Patient Count	Patient Count	Patient Count	Patient Count
1910	1	1		1
1911	2	1	4	1
1912	1	2	1	1
1913	3	1		

Or:

Birth Date Year	Birth Date Quarter	Patient Count
1910	Q1	1
	Q2	1
	Q4	1
1911	Q1	2
	Q2	1
	Q3	4

2.2.4 Base Measures

In most cases, some of the data cells in a pivot table display measures. There are two types of measures: *base measures* (defined within the core model) and calculated measures (discussed in "Calculated Measures," later in this chapter).

When you choose a base table, DeepSee provides a default (base) measure, `Count`, which is a count of the records in the base table. Other base measures are based on numeric or date source values.

You can use `Count` in pivot tables. You can also use any other 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.

2.2.5 Detail Listings and Listing Fields

A detail listing shows the lowest-level data relevant to the current context of the user, specifically the row of the pivot table that the user is examining. An example follows:

Basic Details

PatientID	Age	Birth Date	Test Score	Gender	Favorite Color	Home ZIP
SUBJ_100368	7	21 Mar 2002	89	Female	Blue	34577
SUBJ_100371	8	23 Dec 2001	95	Female	Green	36711
SUBJ_100380	24	16 Feb 85	92	Female	Yellow	32006
SUBJ_100392	65	02 Jul 44	97	Female	Yellow	38928
SUBJ_100417	13	24 Sep 96	90	Female	No Data Available	34577
SUBJ_100419	35	17 Dec 74	81	Female	Orange	36711
SUBJ_100424	2	07 Jun 2007	91	Female	Orange	34577
SUBJ_100437	34	06 Mar 75	54	Female	Yellow	32006
SUBJ_100449	50	30 May 59		Female	Purple	32007
SUBJ_100450	4	23 Apr 2005		Female	No Data Available	38928
SUBJ_100460	44	05 Mar 65	74	Female	Green	34577
SUBJ_100462	29	10 Aug 80	76	Female	Yellow	38928

Each field in the detail listing is a listing field. A detail listing can have many listing fields. For usability, it is best to keep the number of fields small enough so that users do not have to scroll horizontally.

A user can display a detail listing on demand, or a dashboard can display it directly as shown earlier in "Detail Listings."

Also, there are two types of listing fields:

- *Dimension-type listing fields*, which are stored in the fact table. A listing field of this type is a dimension that has also been configured for use as a listing field.
- *Independent listing fields*, which are not stored in the fact table and which are defined separately from any dimensions. The data for these is retrieved at runtime.

2.3 Filter Expressions

Filter expressions (also called *filters*) are extremely important in DeepSee. When you create a building block, pivot table, or dashboards, you can specify a filter expression as part of the configuration of that element. So before we look at building blocks, it is useful to understand these expressions.

A filter expression is a boolean expression that specifies which source records to select. It uses a simple syntax that is specific to DeepSee. For example, the following expression selects only male patients.

```
[Gender = Male]
```

In most applicable locations, DeepSee provides a user interface to create these expressions.

You can apply filter expressions to pivot tables, to KPIs (key performance indicators), and to many intermediate elements in DeepSee. Filters can be driven by user choices in your dashboards as well as by runtime values passed in from your application.

You can create more advanced filter expressions that use logical operators, other comparison operators, and even embedded Caché ObjectScript expressions. Within a filter expression, you can refer to members of different dimensions. This means that you can build arbitrarily complex selection expressions.

In all cases, DeepSee selects specific rows of the fact table. The chapter "How DeepSee Works" shows a conceptual example in a simple case.

2.4 External Building Blocks

Depending on their permissions, developers, superusers, and ordinary users can all create *building blocks* that extend the core model. These building blocks are as follows:

- Subject areas, which partition the data by role and provide access to model elements.
- Compound members, which are custom members defined by filter expressions.
- Calculated measures, which are custom measures based on base measures.
- KPIs (key performance indicators), which are based on measures or on Caché ObjectScript expressions (which can refer to other KPIs).
- Computations, which typically calculate data in a pivot table in terms of other data shown in the pivot table.

2.4.1 Subject Areas

A *subject area* is a handle by which users can access selected dimensions, measures, and detail listings. Specifically, when users create pivot tables, they must start by selecting a subject area. Then the users can access some or all of the dimensions, measures, and detail listings of that subject area.

As noted earlier, subject areas also partition the data. This mechanism works as follows:

- A subject area includes a filter expression, which determines which records can be accessed.
- Access to a subject area is determined by role. Multiple roles can access a subject area.
- Users are assigned to roles. Each user can be assigned to multiple roles but chooses one when logging in to DeepSee.

2.4.2 Compound Members

Compound members are custom members based on existing members. A compound member is defined by a filter expression, and there is no limit to the complexity of that expression. For example, compound members can do the following:

- Select all patients with birth dates between X and Y.
- Select all patients with last names that start with R.
- Select all customers of class D who have traded assets of type T.
- Select all orders except for those relating to a specific order type.
- Select all transactions between dates X and Y that apply to customers in a particular income bracket.
- Select all transactions of a given status that involve assets of vendor V.

When you define a compound member, you assign it to a dimension, which is either an existing dimension or a new dimension.

The compound members can be used in the same way as any other members, which means that the users can drag and drop it into pivot tables, without having to formulate a complex query.

2.4.3 Calculated Measures

You can define *calculated measures*, which can refer to base measures and which can specify other types of aggregation. There are two parts to the definition of a calculated measure:

1. Its basic formula, which determines the value of that measure for a given record of the fact table.

   A calculated measure is not stored in the fact table but is evaluated separately for each applicable record of the fact table.

   The expression is a Caché ObjectScript expression and can include references to the values of any base measures for the same record.

2. Its optional aggregation function. After determining the separate values for each applicable record of the fact table, DeepSee aggregates those values into a single value.

   For numeric values, you can use the following functions: `.Sum`, `.Average`, `.Distinct`, `.Max`, `.Min`, `.Median`, `.StdDev`, and `.Variance`. The default is `.Sum`.

   For date values, you must use the `.FirstDate` and `.LastDate` functions. There is no default.

2.4.4 KPIs

A *KPI* (*key performance indicator*) is essentially a measure that has been given official recognition for stand-alone use. If the measure is redefined, the KPI is updated accordingly.

More generally, you can define a KPI as a Caché ObjectScript expression, which can refer to measures, other KPIs, or any other value that is needed.

You can display KPIs in labels or in speedometers in DeepSee dashboards, in addition to including them in pivot tables.

Unlike a measure, a KPI can have ranges that control font and color.

2.4.5 Computations

Computations are analogous to spreadsheet formulas. You can, for example, subtract one column from another column. Computations are written in Caché ObjectScript, which provides options and features that support any computation that you would perform in a spreadsheet, as well as others that are not supported in spreadsheet applications.

2.4.6 Comparison of Measures, Computations, and KPIs

The following table compares these elements:

Detail	Base Measures	Calculated Measures	Computations	KPIs
Definition can refer to source data	Yes	No	No	No
Definition can refer to a base measure	No	Yes	No	No
Definition can refer to a calculated measure	No	No	No	Yes
Definition can refer to a KPI	No	Yes	Yes	Yes
Definition can refer to other cells in a pivot table	No	No	Yes	No
Definition can include a filter	No	Yes	No	Yes
Can be used as a column in a pivot table	Yes (if numeric)	Yes	Yes	Yes*
Can be used as a row in a pivot table	No	No	Yes	No
Can be used directly in a dashboard	No	No	No	Yes
Can include display rules based on value	No	No	No	Yes

*A KPI can be used as a column of a special type of pivot table called a *scorecard*. For information, see *Using the DeepSee Analyzer*.

2.5 Using Multiple Models in Combination

In many cases, it is appropriate to create multiple models (and their added building blocks). You can use multiple models together in two ways:

- You can create linked pivot tables that use multiple core models (and any added building blocks).

  Such a pivot table either has columns from different models or has rows from different models.

  In the first case, the rows in the pivot table correspond to members of a dimension that is defined identically in both models.

  Similarly, in the second case, the columns correspond to members of a dimension that is defined identically in both models.

- You can create dashboards that use elements from multiple core models (and any added building blocks).

  Such a dashboard typically has filters that use dimensions that are defined identically in both models.

Typically, dimensions based on time and location are common, even for unrelated subject areas. Other dimensions might or might not be common depending on whether the models are related to each other in any meaningful sense. For example, suppose that one model represents patients and another represents doctors. These two subject areas might have common dimensions such as doctor, doctor group, and so on.

For examples, see the *DeepSee Developer Tutorial*.

3

How DeepSee Works

This chapter describes how DeepSee works, at a high level. It includes the following topics:

3.1 Introduction

DeepSee works by creating and then using a fact table and associated indices. When a user runs a dashboard or pivot table, DeepSee uses these elements rather than accessing the raw data directly. Then, when the user displays a detail listing, DeepSee might or might not access the raw data, depending on how the listing fields are defined.

To understand how DeepSee works, it is useful to follow the steps in a possible scenario. In this example, the DeepSee model is based on a table that contains one row for each patient. This model includes dimensions called `Age Bucket` and `Allergies`. It also includes measures called `Patient Count`, `Test Score`, and `Average Test Score`.

The following dashboard includes a pivot table and a drop-down list that filters the pivot table. The user selects an allergy from the drop-down list, and DeepSee filters the pivot table to show only data for that allergy.

Select an allergy: pollen

Stats for Selected Allergy   Level 0 of 0

Age Bucket	Patient Count	Cumulative Patient Count	Test Score	Average Test Score
0-9	63	63	3,871	75.90
10-19	60	123	3,400	73.91
20-29	54	177	3,495	77.67
30-39	80	257	4,693	72.20
40-49	70	327	4,008	74.22
50-59	55	382	2,836	74.63
60-69	24	406	1,577	71.68
70-79	25	431	1,755	76.30
80+	10	441	760	76.00

In addition to the measures, this pivot table includes a column that is labeled `Cumulative Patient Count`. This column displays a computation, which is a cumulative count of the patients in the current age bucket and the preceding age buckets.

The following sections demonstrate how DeepSee determines the values shown in this sample dashboard.

3.2 Introduction to the Fact Table and Indices

When you build the indices for DeepSee, DeepSee also creates a *fact table*. This table is for DeepSee internal use and does not support direct queries. However, it is very useful to visualize this table and to know how it is built and used.

The fact table has one record for each record of the base table. The table contains one field for each dimension and one field for each measure. The following shows a sketch:

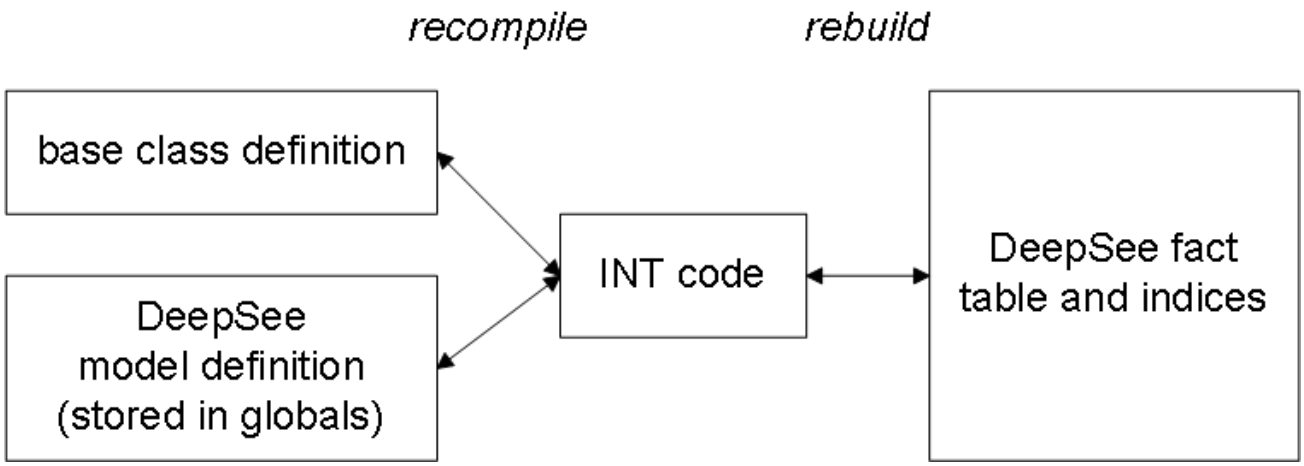
dim A	dim B	dim C	...	meas A	meas B	meas C	...
nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn

The field for a given dimension might contain no values, a single value, or multiple values. Each distinct value corresponds to a member of this dimension. The field for any given measure contains either null or a single value.

When DeepSee builds this fact table, it also generates indices for it. DeepSee uses its own bitmap indices (which are ignored by Caché SQL). Null values are not indexed.

3.3 Relationship of the DeepSee Model and the Base Class

When you define a DeepSee model, the model definition is stored in globals and the details are not visible in Studio as part of the base class. When you compile the class, however, Caché uses the model definition and the class definition together and compiles both into the same INT code, as the following figure shows:



When you build the DeepSee fact table and indices, Caché uses this INT code.

3.4 Building the Fact Table and Indices

When you build the DeepSee fact table and indices, Caché iterates through the records of the base table. For each record, Caché does the following:

- Examines the definition of each dimension and obtains either no value, a single value, or multiple values.

  In this step, Caché determines how to categorize the record.
- Examines the definition of each measure and obtains either no value or a single value.

Caché then writes this data to the corresponding row in the fact table and updates the indices appropriately.

3.4.1 Determining Values for a Dimension

In a DeepSee model, each dimension is specified as either a source property or a source expression. For a source expression, you either specify a Caché ObjectScript expression directly or you specify information that is compiled into an expression. Most source expressions return a single value for given record, but DeepSee provides an option (called **Manual Child Browse**) that generates an expression that can return multiple values.

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

In any case, for a given record in the base table, DeepSee evaluates that property or expression at build time, stores the corresponding value or values in the fact table, and updates the indices appropriately.

For example, the `Age Bucket` dimension is defined as an expression that returns one of the following strings: `0-9`, `10-19`, `20-29`, and so on. The value returned depends upon the patient's age. DeepSee stores the returned value in the fact table, within the field that corresponds to the `Age Bucket` dimension. The following figure shows an example of what the fact table could look like:

Age Bucket	Allergies	dim C	...	meas A	meas B	meas C	...
40-49	dairy products,eggs	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
0-9		nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
50-59		nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
10-19	ant bites,bee stings,soy	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
0-9		nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
10-19	dairy	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
30-39	ant bites,dairy products	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
20-29	ant bites,bee stings,eggs	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
80+	ant bites	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
20-29	soy	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn	nnnnn
...							

Similarly, the `Allergy` dimension uses the **Manual Child Browse** option to return multiple values, one for each allergy of the patient.

3.4.2 Determining the Value for a Measure

DeepSee has two types of measures:

- Base measures, which are stored in the fact table. In a DeepSee model, each base measure is specified as either a source property or a Caché ObjectScript source expression.
- Calculated measures, which are not stored in the fact table. These measures are instead based on the base measures and are computed when requested. Calculated measures can be aggregated in many different ways.

When DeepSee builds the fact table, it determines and stores values for the base measures. For a given row in the base table, DeepSee looks at the measure definition, evaluates it, and stores that value (if any) in the appropriate measure field.

For example, the `Test Score` measure is based on the `TestScore` property of the patients. The following figure shows an example of what the fact table could look like:

Age Bucket	Allergies	dim C	...	Test Score	meas B	meas C	...
40-49	dairy products,eggs	nnnnn	nnnnn	78	nnnnn	nnnnn	nnnnn
0-9		nnnnn	nnnnn	53	nnnnn	nnnnn	nnnnn
50-59		nnnnn	nnnnn	89	nnnnn	nnnnn	nnnnn
10-19	ant bites,bee stings,soy	nnnnn	nnnnn	61	nnnnn	nnnnn	nnnnn
0-9		nnnnn	nnnnn	90	nnnnn	nnnnn	nnnnn
10-19	dairy	nnnnn	nnnnn	67	nnnnn	nnnnn	nnnnn
30-39	ant bites,dairy products	nnnnn	nnnnn	74	nnnnn	nnnnn	nnnnn
20-29	ant bites,bee stings,eggs	nnnnn	nnnnn	89	nnnnn	nnnnn	nnnnn
80+	ant bites	nnnnn	nnnnn	84	nnnnn	nnnnn	nnnnn
20-29	soy	nnnnn	nnnnn	59	nnnnn	nnnnn	nnnnn
...							

3.5 Responding to Real-time Data Changes

Suppose that for patient 100–000–000, a user updates the list of allergies and the test score, within your application. Also suppose that we have enabled incremental updates for DeepSee.

DeepSee detects the data change and automatically updates the fact table and indices. The fact table might now look like this, assuming that patient 100–000–000 is represented in the first row:

Age Bucket	Allergies	dim C	...	Test Score	meas B	meas C	...
40-49	eggs	*nnnnn*	*nnnnn*	84	*nnnnn*	*nnnnn*	*nnnnn*
0-9		*nnnnn*	*nnnnn*	53	*nnnnn*	*nnnnn*	*nnnnn*
50-59		*nnnnn*	*nnnnn*	89	*nnnnn*	*nnnnn*	*nnnnn*
10-19	ant bites,bee stings,soy	*nnnnn*	*nnnnn*	61	*nnnnn*	*nnnnn*	*nnnnn*
0-9		*nnnnn*	*nnnnn*	90	*nnnnn*	*nnnnn*	*nnnnn*
10-19	dairy	*nnnnn*	*nnnnn*	67	*nnnnn*	*nnnnn*	*nnnnn*
30-39	ant bites,dairy products	*nnnnn*	*nnnnn*	74	*nnnnn*	*nnnnn*	*nnnnn*
20-29	ant bites,bee stings,eggs	*nnnnn*	*nnnnn*	89	*nnnnn*	*nnnnn*	*nnnnn*
80+	ant bites	*nnnnn*	*nnnnn*	84	*nnnnn*	*nnnnn*	*nnnnn*
20-29	soy	*nnnnn*	*nnnnn*	59	*nnnnn*	*nnnnn*	*nnnnn*
...							

Similarly, if a user used your application to add or delete a patient record, DeepSee would detect that data change and update the fact table and indices correspondingly.

3.6 Displaying a Dashboard and Pivot Table

Now let us consider what occurs when a user displays the dashboard.

1. First, consider the pivot table shown earlier. The first row in the pivot table refers to the `0-9` member of the `Age Bucket` dimension.

   DeepSee uses the indices to find all the relevant patients (shown here with red highlighting) in the fact table:

Age Bucket	Allergies	dim C	...	Test Score	meas B	meas C	...
40-49	eggs	*nnnnn*	*nnnnn*	84	*nnnnn*	*nnnnn*	*nnnnn*
0-9		*nnnnn*	*nnnnn*	53	*nnnnn*	*nnnnn*	*nnnnn*
50-59		*nnnnn*	*nnnnn*	89	*nnnnn*	*nnnnn*	*nnnnn*
10-19	ant bites,bee stings,soy	*nnnnn*	*nnnnn*	61	*nnnnn*	*nnnnn*	*nnnnn*
0-9		*nnnnn*	*nnnnn*	90	*nnnnn*	*nnnnn*	*nnnnn*
10-19	dairy	*nnnnn*	*nnnnn*	67	*nnnnn*	*nnnnn*	*nnnnn*
30-39	ant bites,dairy products	*nnnnn*	*nnnnn*	74	*nnnnn*	*nnnnn*	*nnnnn*
20-29	ant bites,bee stings,eggs	*nnnnn*	*nnnnn*	89	*nnnnn*	*nnnnn*	*nnnnn*
80+	ant bites	*nnnnn*	*nnnnn*	84	*nnnnn*	*nnnnn*	*nnnnn*
20-29	soy	*nnnnn*	*nnnnn*	59	*nnnnn*	*nnnnn*	*nnnnn*
...							

2. In the pivot table, the `Patient Count` column is meant to contain the count of patients used in a given context.

   For the first row in the pivot table, to determine the value for the `Patient Count` column, DeepSee counts the number of records in the fact table that it has found for the `0-9` member, in this case 849 patients.

3. The `Test Score` column is meant to display the cumulative test score for the patients in a given context.

   For the first row in the pivot table, to determine the value for this measure, DeepSee first finds the values for the `Test Score` in the fact table that it has found for the `0-9` member:

Age Bucket	Allergies	dim C	...	Test Score	meas B	meas C	...
40-49	eggs	*nnnnn*	*nnnnn*	84	*nnnnn*	*nnnnn*	*nnnnn*
0-9		*nnnnn*	*nnnnn*	53	*nnnnn*	*nnnnn*	*nnnnn*
50-59		*nnnnn*	*nnnnn*	89	*nnnnn*	*nnnnn*	*nnnnn*
10-19	ant bites,bee stings,soy	*nnnnn*	*nnnnn*	61	*nnnnn*	*nnnnn*	*nnnnn*
0-9		*nnnnn*	*nnnnn*	90	*nnnnn*	*nnnnn*	*nnnnn*
10-19	dairy	*nnnnn*	*nnnnn*	67	*nnnnn*	*nnnnn*	*nnnnn*
30-39	ant bites,dairy products	*nnnnn*	*nnnnn*	74	*nnnnn*	*nnnnn*	*nnnnn*
20-29	ant bites,bee stings,eggs	*nnnnn*	*nnnnn*	89	*nnnnn*	*nnnnn*	*nnnnn*
80+	ant bites	*nnnnn*	*nnnnn*	84	*nnnnn*	*nnnnn*	*nnnnn*
20-29	soy	*nnnnn*	*nnnnn*	59	*nnnnn*	*nnnnn*	*nnnnn*
...							

   Then it aggregates those numbers together, in this case by adding them to the value 49167.

4. The `Average Test Score` column is meant to display the average test score for the patients in a given context.

   This measure is based on the `Test Score` base measure but is aggregated by averaging the values. DeepSee adds up all the values for `Test Score` for the patients in the `0–9` year bucket and then divides that sum by the number of patients in that bucket. The resulting value is 74.84.

5. DeepSee repeats the preceding for all rows and columns, ignoring any computations for now. In memory, the result are as follows:

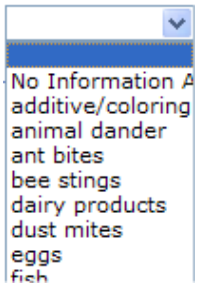
Age Bucket	Patient Count	Cumulative Patient Count	Test Score	Average Test Score
0-9	849	???	49,167	74.84
10-19	887	???	53,567	74.50
20-29	825	???	48,502	75.20
30-39	925	???	53,011	73.83
40-49	959	???	55,788	75.08
50-59	702	???	41,489	74.62
60-69	428	???	25,803	73.93
70-79	368	???	21,955	74.68
80+	189	???	10,626	72.29

6. DeepSee evaluates any computations for this pivot table and then displays the pivot table.

   This pivot table includes one computation, `Cumulative Patient Count`. This computation is defined as the sum of the cell above and the cell to the immediate left. After evaluating this computation for each cell in that column, DeepSee displays the result:

Age Bucket	Patient Count	Cumulative Patient Count	Test Score	Average Test Score
0-9	1,378	1,378	80,784	74.32
10-19	1,448	2,826	87,025	74.89
20-29	1,372	4,198	81,846	74.95
30-39	1,519	5,717	89,363	74.22
40-49	1,515	7,232	88,676	74.96
50-59	1,123	8,355	66,852	74.61
60-69	727	9,082	43,382	73.78
70-79	597	9,679	35,785	75.34
80+	321	10,000	18,538	72.70

7. The dashboard shown earlier also has a drop-down list from which the user can select an allergy. This drop-down list has been configured to filter the pivot table. The drop-down list displays all the allergies:



   To build this list, DeepSee finds all distinct values for the `Allergy` dimension.

8. When the user selects an allergy, DeepSee uses the indices to find the appropriate rows in the fact table. For example, the user selects `ant bites`:

Age Bucket	Allergies	dim C	...	Test Score	meas B	meas C	...
40-49	eggs	*nnnnn*	*nnnnn*	84	*nnnnn*	*nnnnn*	*nnnnn*
0-9		*nnnnn*	*nnnnn*	53	*nnnnn*	*nnnnn*	*nnnnn*
50-59		*nnnnn*	*nnnnn*	89	*nnnnn*	*nnnnn*	*nnnnn*
10-19	ant bites,bee stings,soy	*nnnnn*	*nnnnn*	61	*nnnnn*	*nnnnn*	*nnnnn*
0-9		*nnnnn*	*nnnnn*	90	*nnnnn*	*nnnnn*	*nnnnn*
10-19	dairy	*nnnnn*	*nnnnn*	67	*nnnnn*	*nnnnn*	*nnnnn*
30-39	ant bites,dairy products	*nnnnn*	*nnnnn*	74	*nnnnn*	*nnnnn*	*nnnnn*
20-29	ant bites,bee stings,eggs	*nnnnn*	*nnnnn*	89	*nnnnn*	*nnnnn*	*nnnnn*
80+	ant bites	*nnnnn*	*nnnnn*	84	*nnnnn*	*nnnnn*	*nnnnn*
20-29	soy	*nnnnn*	*nnnnn*	59	*nnnnn*	*nnnnn*	*nnnnn*
...							

   These rows are the only rows that DeepSee will use in the pivot table. It ignores all other rows of the fact table.

9. DeepSee then constructs each pivot table row as it did before, but using only the subset of rows from the fact table.

3.7 Displaying a Detail Listing

When a user requests a detail listing, DeepSee accesses the data as follows:

- For each dimension-type listing field, DeepSee retrieves the data from the fact table. This data is thus determined at build time.
- For each independent listing field, DeepSee accesses the source tables, as specified in the DeepSee model. This data is thus determined at runtime.

4

Introduction to the Toolkit

This chapter introduces the tools provided by InterSystems DeepSee. It discusses the following topics:

- How to log in to DeepSee.
- DeepSee Architect, which you use to define the DeepSee model for use in the DeepSee Analyzer.
- DeepSee Analyzer, which you use to define pivot tables or to simply explore the data.
- DeepSee Designer, which you use to define dashboards, in particular dashboards that embed pivot tables.
- DeepSee Connector, which you use to import data from external database or flat files, in order to generate classes (and data-loading methods) that serve as the basis of the DeepSee model. This component is available only with Ensemble, and it is not needed in all cases.
- Additional, supporting modules.
- An overview of how you use these tools in combination.

4.1 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**.

4.2 DeepSee Architect

The purpose of DeepSee Architect is to define the core model and some of the building blocks needed to create pivot tables and dashboards. In the Architect, you define the following:

- Dimensions
- Base measures
- Detail listing fields
- Detail listings
- Subject areas
- Compound members (which can also be defined in the Analyzer)

4.2.1 Users

The Architect is meant for use by implementers, primarily application partners of InterSystems. In order to use the Architect most effectively, it is important to have a good understanding of your data classes.

4.2.2 Use Options

The Architect lets you work with and modify any BI-enabled classes, which are any classes that extend the DeepSee superclass (%BI.Adaptor).

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.
- If you are creating a new application, you can create it in Studio in the same way you create any other Caché application. For any class that needs it, you 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, you can first use the DeepSee Connector to browse the data and generate class definitions and their corresponding data-loading methods. This module is discussed later in this chapter.

In all cases, the Architect automatically has access to any classes that extend %BI.Adaptor.

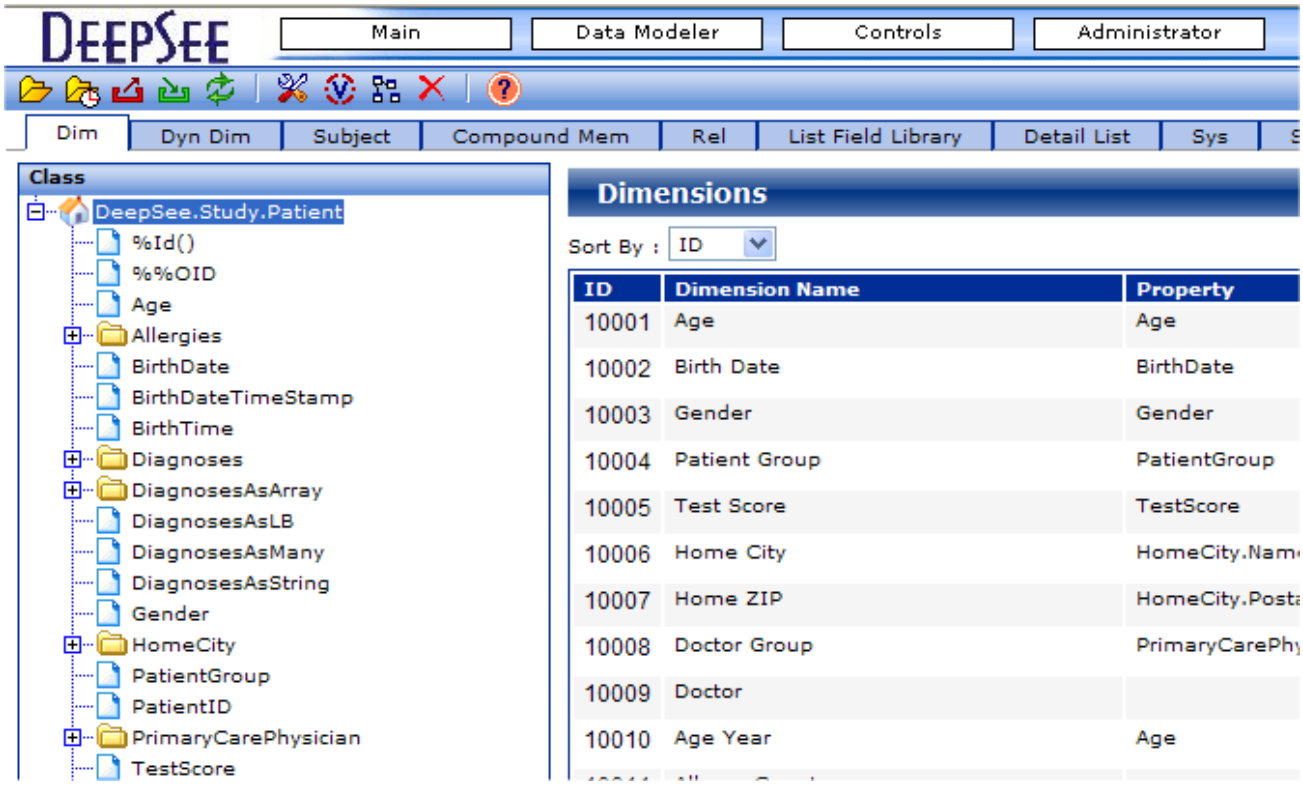
Once you have a BI-enabled class, you use the Architect to create dimensions, measures, and the other model elements based on this class. Some changes directly affect your class definition, but other changes are written to globals and are not visible in the class definition.

The Architect accesses and modifies the same class definition 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 classes as needed from Architect.

4.2.3 A Quick Look

The Architect looks like this:



In this example, we are looking at the BI-enabled class `DeepSee.Study.Patient` and its related classes. We can use all properties of these classes when we define the DeepSee model. As you can see, many dimensions have already been created.

To use the Architect, you first open a BI-enabled class. The Architect displays this class and related classes as a hierarchy, as you can see in the previous example. Tabs in the Architect display different model elements related to that class and its related classes:

- The **Dim** tab displays the dimensions defined within these classes.
- The **Dyn. Dim** tab displays the dynamic (non-indexed) dimensions defined within these classes.
- The **Subject** tab displays the subject areas defined for these classes and specifies the DeepSee roles that can access them.
- The **Compound Member** tab displays the compound members. These members may or may not be indexed, as you choose.
- The **List Field Library** tab displays the independent listing fields.
- The **Detail List** tab displays the detail listings and specifies the DeepSee roles that can access them.

The other tabs are for special cases and are not described here.

The toolbar provides options for recompiling and rebuilding, as well as options for exporting or importing elements of the base model.

4.3 DeepSee Analyzer

The primary purpose of DeepSee Analyzer is to define pivot tables and to support analysis. In the Analyzer, you define the following:

- Pivot tables
- Calculated measures
- Computations
- Compound members (which can also be defined in the Architect)

4.3.1 Users

The Analyzer is intended for implementers and advanced users.

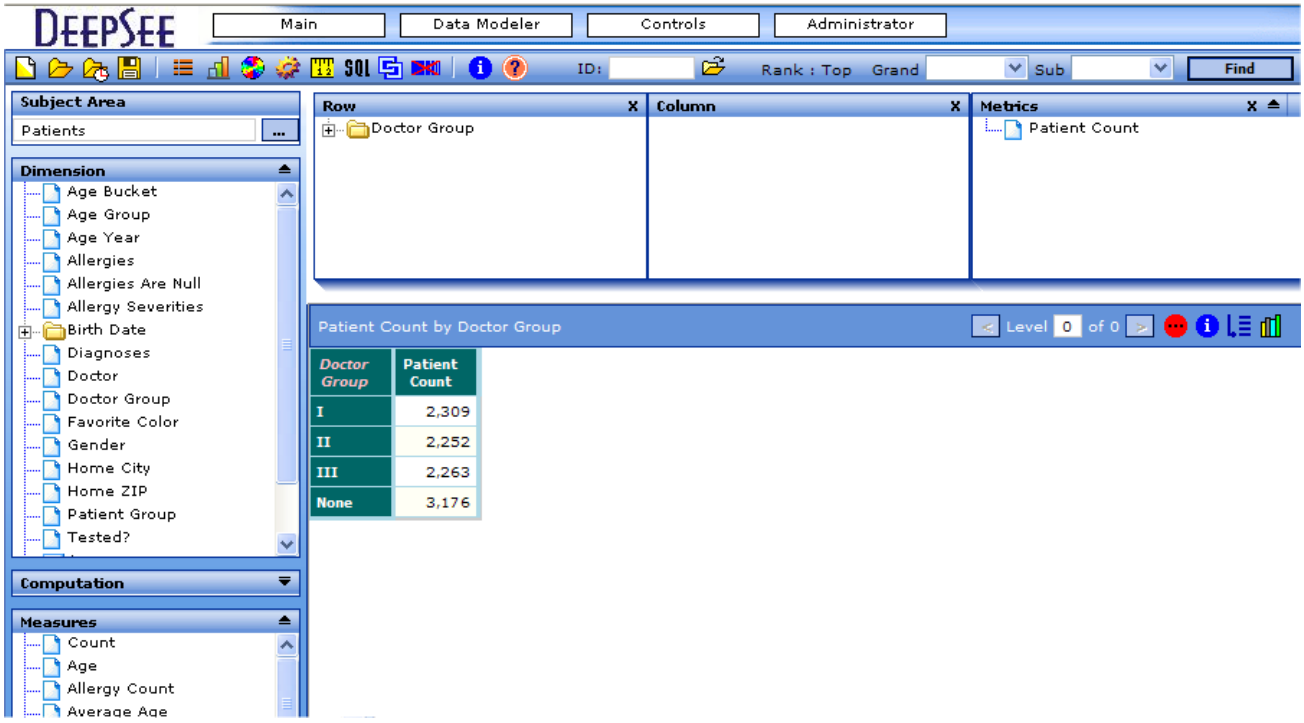
4.3.2 Use Options

In most cases, the Analyzer is used to create pivot tables, but you can also use it simply to explore the data, without ever saving a pivot table.

Also, the Analyzer can be packaged within a dashboard and can thus be provided as an embedded user interface within your application, for the appropriate set of users.

4.3.3 A Quick Look

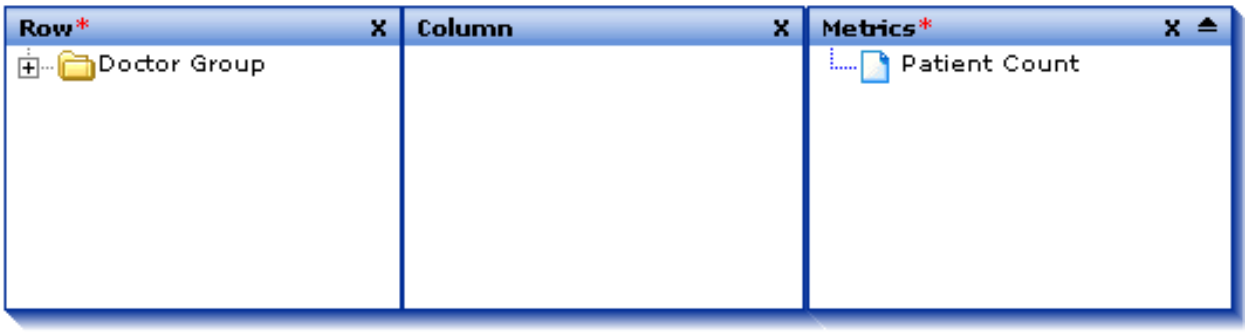
The Analyzer looks like this:



The left area displays the building blocks available within a subject area:

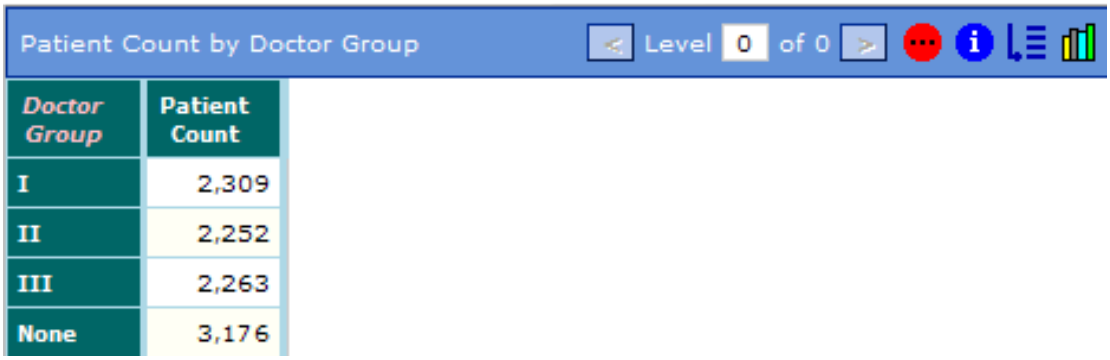
- The **Subject Area** panel shows the currently selected subject area.
- The **Dimension** panel shows the dimensions in that subject area that are available to your roles.
- The **Computation** panel shows the computations defined in this subject area. This includes predefined computations and any computations you have added.
- The **Measures** panel shows the measures defined in this subject area. This includes predefined measures and any measures you have added.
- The **Drill Down** panel (not shown) shows the drill-downs, if any, that you have defined in this subject area or for this pivot table.
- The **Others** panel (not shown) lets you add time series, scorecard rows, and labels to the currently viewed pivot table.
- The **Filter** panel (not shown) lets you define and see the filter applied to the currently viewed the pivot table.

The top right panels define the currently displayed pivot table. For example:



- The **Row** panel displays the dimensions used to group the data rows, as well as any labels, computations, or other elements used as rows.
- The **Column** panel displays the dimensions used to group the data columns, as well as any other elements used as columns.
- The **Metrics** panel displays the measures or computations shown in the data cells.

The bottom right area displays a fully functioning preview of the pivot table that you are currently creating or modifying, as in the following example:



Doctor Group	Patient Count
I	2,309
II	2,252
III	2,263
None	3,176

4.4 DeepSee Designer

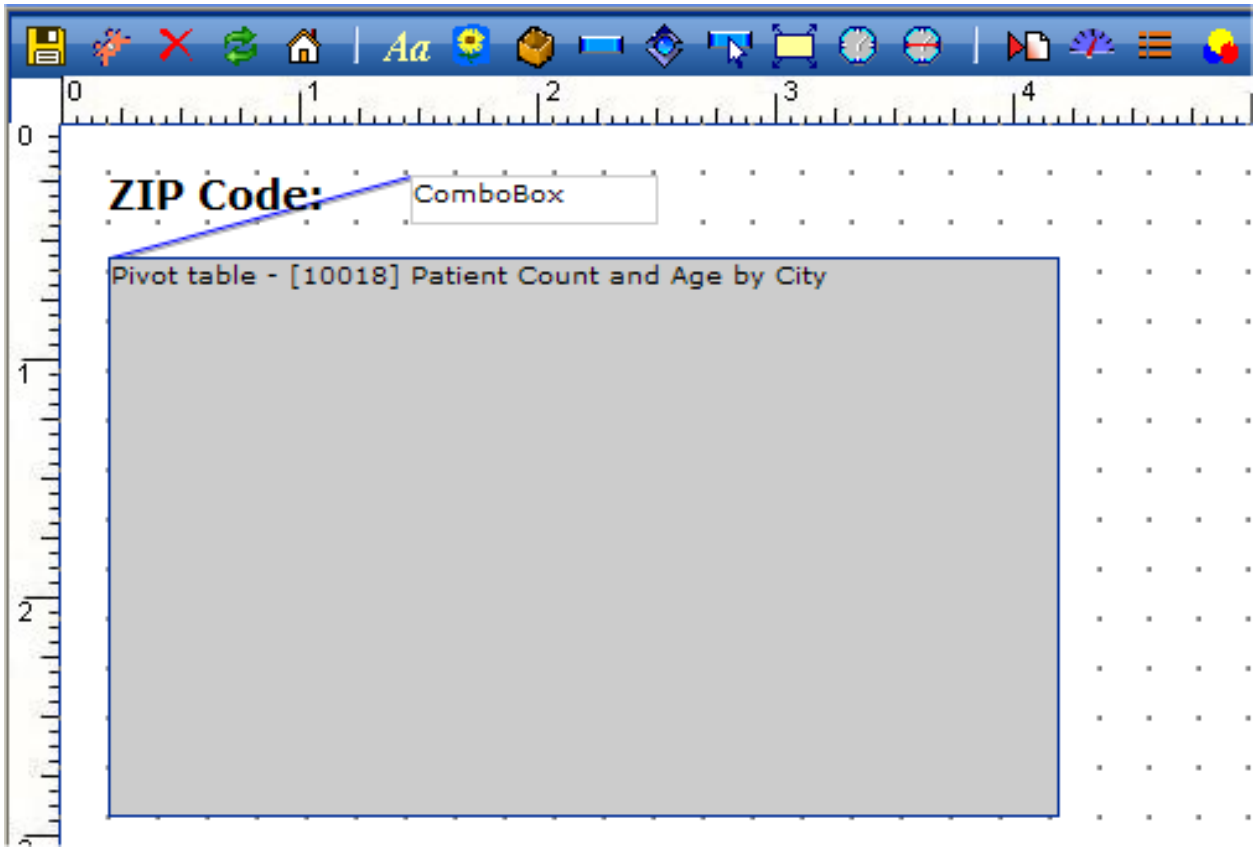
The purpose of the DeepSee Dashboard Designer is to define dashboards.

4.4.1 Users

The Designer is intended for implementers and advanced users.

4.4.2 A Quick Look

The Designer displays an editable version of a dashboard, along with tools you can use to make changes. For example:



The toolbar contains elements that you can add to the dashboard and options that control its appearance and behavior.

Within the grid of dots, the white and gray boxes represent elements displayed on the dashboard, and the blue and gray lines represent different kinds of connections between the elements. In this case, the combo box element is connected to each speedometer and to the dashboard in two ways:

- The filter specified by the combo box is applied to the associated elements.
- When the combo box is used, the associated elements are refreshed.

4.5 DeepSee Connector

The purpose of the DeepSee Connector is to import externally stored data (in external database or in flat files) and to generate classes that model the data, along with data-loading methods for those classes. This component is available only with Ensemble.

4.5.1 Users

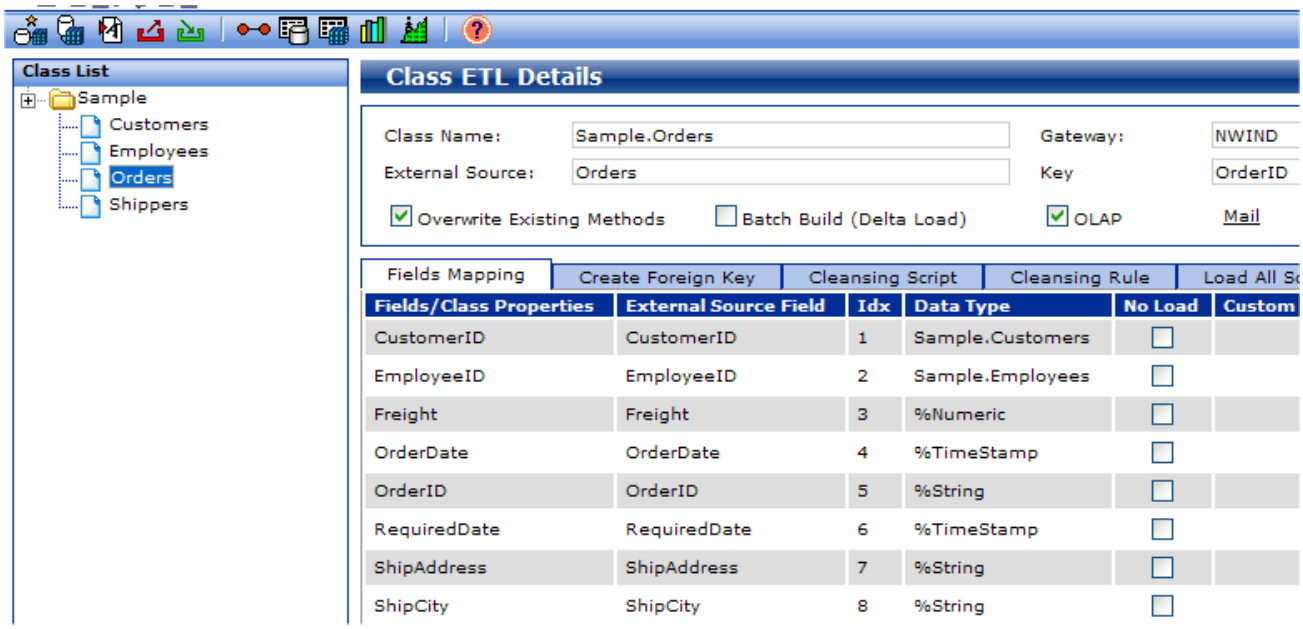
The Connector is intended primarily for implementers.

4.5.2 Use Options

You use the Connector in two phases. First you use it to import the external data and specify its basic structure (primary keys, data types, and so on). In this step, you create the initial classes and methods that load the data for them from the external sources. Then you use it to examine the classes, fine-tune the structure, and specify additional options such as data cleaning rules.

4.5.3 A Quick Look

The Connector looks like the following:



Class List (on the left) shows the classes that you have created with the Connector. The right area shows details for the selected class. The buttons in the toolbar launch additional wizards.

4.6 Additional Modules

DeepSee provides additional, supporting modules which include the following:

Module	Purpose	Relevant Book
Transformations	Defines reusable transformations that you can use within the Architect and the Connector.	*Using the DeepSee Architect*
KPI Setup	Defines KPIs (key performance indicators) that you can use within the Analyzer and the Dashboard Designer.	*Using the DeepSee Analyzer*
Shortcut Management	Define shortcuts to use in dashboard drop-down menus.	*Using the DeepSee Dashboard Designer*
Venn Diagram Setup	Defines Venn diagrams that you can use within the Dashboard Designer.	
Image List	Load image files into the DeepSee library, for use in dashboards, alerts, and KPI display rules	*DeepSee Site Configuration and Maintenance Guide*
Scheduler	Schedule the tasks of rebuilding indices, loading data, and executing custom Caché ObjectScript.	
Site Configuration	Specify site-wide configuration options.	
Role Maintenance	Define DeepSee roles.	
User Maintenance	Define DeepSee users.	
User Preferences	Configure the DeepSee user interface as seen by each user.	
Folder Management	Export and import DeepSee pivot tables, KPIs, and dashboards.	
Multi Sites Setup	Enables you to define site groups. A site group consists of a set of databases from which you intend to extract data in a common form to use as a single source for use in the Connector.	*Using the DeepSee Connector*

Important: DeepSee includes a utility to set default values for query variables; this is called the Query Variable module. InterSystems recommends that you ignore this module, which does not provide sufficient functionality.

4.7 Overall Sequence

You use these tools together as in a sequence of overlapping phases, as follows:

1. Use the Site Configuration module to define basic site options such as the locations of log files, the date format, and so on.
2. Optionally use the DeepSee Connector to access external data and generate classes.

   Also use the Transformations module to define transformations for use in the Connector.

   Otherwise, use Studio to create new classes or to BI-enable existing classes.

3. Use the DeepSee Architect to define the core model and some additional building blocks.

   Also use the Transformations module to define transformations for use in the Connector and the Architect.

   If necessary, return to the Connector and make changes there.

   Both the Architect and the Connector can potentially make changes to the class definitions; both components provide a convenient option to recompile.

4. Use the DeepSee Analyzer to define pivot tables and any building blocks as needed.

   Also, use the KPI Setup module to define KPIs for use in pivot tables and in dashboards.

   Also, use the Image List module to load images for use in alerts and in KPI override rules.

   If necessary, return to the Architect (or less often, the Connector), make changes to the core model, and recompile or rebuild if necessary.

5. Use the DeepSee Designer to create dashboards.

   Also, use the KPI Setup module and the Venn Diagram Setup module to define KPIs and Venn diagrams for use in dashboards.

   Also, use the Image List module to load images for dashboards.

   If necessary, return to the Analyzer and change or add pivot tables.

6. Use the Scheduler module to schedule the tasks of data loading (if applicable) and full index rebuilding (if necessary).

5

Available Options

This chapter summarizes most of the options available in DeepSee, grouped into the following categories:

- Options for core models
- Options for external building blocks
- Options for pivot tables
- Options for charts
- Options for dashboards
- Runtime parameters passed by your application
- Custom code in DeepSee

For options related to the Connector, see *Using the DeepSee Connector*.

For options related to site setup and administration, see the *DeepSee Site Configuration and Maintenance Guide*.

5.1 Options for Core Models

This section summarizes options for a core model; these options affect how the fact table and indices are built. This means that if you modify these options, it is necessary to recompile and (in nearly all cases) rebuild. These options are all available in the Architect.

5.1.1 Dimensions

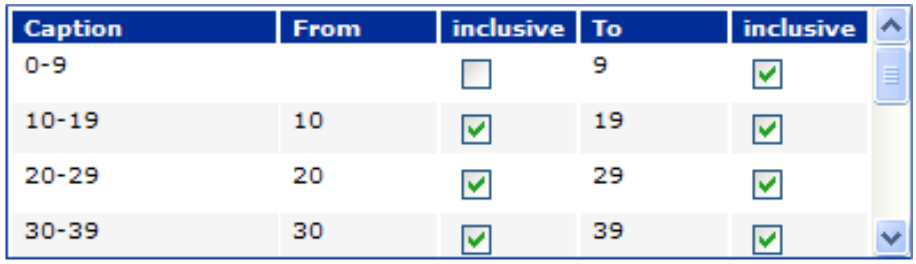
You can define dimensions in all the following ways:

- You can create a dimension based directly on a property. DeepSee generates a separate member for each different value of this property.

  In this case, the property must be a simple-valued property (that is, not an object).
- Similarly, you can create a dimension based on a property of a property (using cascading dot syntax).
- You can define a dimension that has multiple values for a given record in the base table, as noted earlier. For example, a patient can have multiple allergies.

  This type of dimension is typically based on a collection property.

- You can create a dimension based on bins that are applied to a numeric property. Each bin becomes a separate member of this dimension. For example:

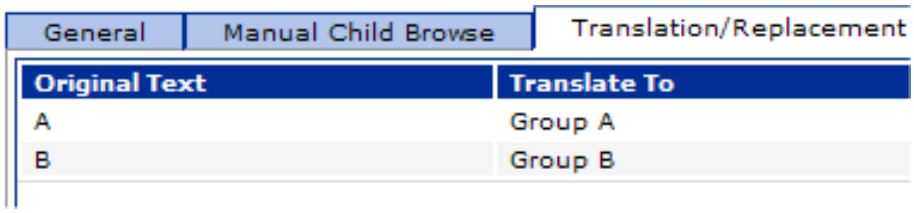


Caption	From	inclusive	To	inclusive
0-9		☐	9	☑
10-19	10	☑	19	☑
20-29	20	☑	29	☑
30-39	30	☑	39	☑

  The bins are assigned when the fact table and indices are built.

  Note that you are not required to place numeric values into bins; you can directly use a numeric property as a dimension.

- You can create a dimension that uses a set of translations, so that DeepSee stores and uses different strings than the raw data. For example:



General | Manual Child Browse | Translation/Replacement

Original Text	Translate To
A	Group A
B	Group B

  The translations are performed when the fact table and indices are built.

- If you base a dimension on a property and that property is defined to have both internal and external values, you can define the dimension to use the external values. You would use this with a property like the following:

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

- You can specify a string to use as the member name when the dimension value is null.

  For example, if a patient has no assigned patient group, by default, DeepSee does not write any data into the `Patient Group` dimension field of the fact table for that patient. You can specify a string for DeepSee to use in this scenario, such as `No Group`. In this case, DeepSee writes `No Group` into the fact table and indexes that value.

- You can create a dimension based on a date property. When you do so, DeepSee actually creates a set of related dimensions that enable you to analyze by date in multiple ways, as described in "Date Dimensions," earlier in this book.

- You can create a dimension based on time (that is, at a lower level than by day). You can create hourly bins or any other bins that are appropriate for your needs.

- You can create a dimension based on an arbitrary Caché ObjectScript expression. This expression could do any of the following, for example:

  – Evaluate a condition and return a string

  – Use dynamic or embedded SQL to retrieve a value from another table

  – Combine fields of the base table into a single string

  The expression can, and typically does, refer to the current record of the base table. This expression is evaluated when DeepSee iterates through the base table, as it builds the fact table and indices.

- If you expect the member names to change and do not want to rebuild the fact table when that occurs, you can define the dimension as type **Reference**.

When you define dimensions, you specify their names. Within the Analyzer, the users can customize these names and specify display properties. Users can also modify any member names.

5.1.2 Base Measures

Base measures are the measures that are stored in the fact table. Base measures must have numeric or date values. You can define them in all the following ways:

- You can create a measure based directly on a property.
- You can create a measure based on an arbitrary Caché ObjectScript expression. This expression could do any of the following, for example:
  - Evaluate a condition and return a number
  - Use dynamic or embedded SQL to retrieve a value from another table
  - Combine numeric or date fields of the base table into a single value

  The expression is evaluated when DeepSee builds the fact table and indices.

When you define measures, you specify their names. Within the Analyzer, the users can customize these names and specify display properties.

5.1.3 Detail Listings and Listing Fields

When defining a listing field, you choose to define it as either a dimension-type listing field or an independent listing field. In either case, you have all the options listed earlier for dimensions, with the following exceptions:

- A listing field cannot have multiple values for a record in the base table. It can, however, contain a string that displays multiple items.
- If you create a dimension of type **Reference**, you cannot use that as a listing field.

You can define as many detail listings as you need. The definition of a detail listing indicates which listing fields to use, what captions to use, and the order to display them. The definition also specifies which DeepSee roles can see the detail listing.

5.2 Options for External Building Blocks

After a DeepSee model has been defined, compiled, and built, users can extend the model in several ways that do not require recompiling or rebuilding. This section summarizes those options.

5.2.1 Subject Areas

Subject areas are defined in the Architect. The definition of a subject area consists of the following:

- Its name.
- A set of user roles that have access to this subject area.
- A subset (or all) of the dimensions defined for these BI-enabled classes.

For example, if a given user role has access to data from only one country, that user does not need the Country and Continent dimensions. For another example, there might be some breakdowns that only certain users should be able to view; only those users would have access to the associated dimensions.

- An optional filter expression.

  When a pivot table runs, if the pivot table is unfiltered, it retrieves all the data in the subject area. It is therefore important to filter the subject areas appropriately.

5.2.2 Compound Members

Compound members are defined in the Architect or in the Analyzer; they are visible in either location.

The definition of a compound member consists of the following:

- Its name.
- The filter expression that defines it.
- Name of the dimension to which this member belongs. This can be an existing dimension or a new one.

For example, if the model already has a Country dimension, a user could create a Continent dimension. The definition of, for example, the Europe member would be an expression that selects records for Austria, Belgium, Denmark, and so on.

5.2.3 Calculated Measures

Calculated measures are defined in the Analyzer. The definition of a calculated measure consists of the following:

- Its name.
- Its basic formula, which determines the value of that measure for a given record of the fact table.

  The expression is a Caché ObjectScript expression and can include references to the values of any base measures for the same record.
- Its aggregation function. After determining the separate values for each applicable record of the fact table, DeepSee aggregates those values into a single value.

  For numeric values, you can use the following functions: `.Sum`, `.Average`, `.Distinct`, `.Max`, `.Min`, `.Median`, `.StdDev`, and `.Variance`. The default is `.Sum`.

  For date values, you can use the `.FirstDate` and `.LastDate` functions.
- Display options including format and (if applicable) the number of decimal places to show.
- Where it can be seen. A calculated measure is defined either in the subject area or locally in a single pivot table.
- An optional filter expression.

The following list suggests some possible calculated measures:

- Backlog calculations.
- Average number of days since a particular event.
- Number of associated items. For example, a custom measure could indicate the number of lab tests or the number of types of tests.

5.2.4 KPIs

KPIs are defined in the KPI Setup module. The definition of a KPI consists of the following details:

- Its name.
- Name of the measure on which this KPI is based. (If the measure is redefined, the KPI is updated accordingly.)

  Or a KPI can be based on a Caché ObjectScript expression, which can combine values for KPIs.
- An optional filter expression.
- A set of rules that specify font and color overrides, based on ranges.

5.2.5 Computations

Computations are defined in the Analyzer. The definition of a computation consists of the following set of information:

- Its name.
- Its formula, which is a Caché ObjectScript expression that can refer to data in other cells of the pivot table.

  A large selection of functions, macros, and special variables is available, and you can use any Caché ObjectScript expression.
- Display options including format and (if applicable) the number of decimal places to show.
- Where it can be seen. A calculated measure is defined either in the subject area or locally in a single pivot table.

The following list suggests some possible computations:

- Compute the difference between two columns or two rows
- Compute a cumulative total
- Divide one column by another and display as a percentage
- Use If-Then-Else logic to determine which number to display
- Compute the total value shown in another column or row (or use this total within a more complex calculation)

5.3 Options for Pivot Tables

Pivot tables are defined in the Analyzer. When you create pivot tables, you have the options described in this section.

5.3.1 Basic Content

You can control the content of pivot tables in all the following ways:

- Select the dimensions for the rows and columns.
- Select the measures to display as columns.
- Select the computations to show as rows or columns.
- Customize all titles.
- Control whether null rows and null columns are included and specify how null values are replaced, if wanted.

- Specify any filters to apply to the pivot table.

5.3.2 Display Options

In addition to controlling the content, you can control the appearance of any given pivot table as follows:

- Specify the default sorting, maximum number of rows and columns, and maximum number of rows per page.
- Rank the items, control how ranking is performed (top or bottom, number of items), specify which value is used for ranking (for example, the patient count or the average test score), and control whether to display the remainder or not.
- Control the use of captions. You have control over every caption.
- Specify the default font and colors used in the table.
- Resize the columns, and you can specify the row height.
- Hide the pivot header (which contains buttons for the users).
- Hide columns. This is useful if the pivot table includes columns as intermediate calculations that you do not want to display.
- Add labels (empty, labelled rows or columns) to group data visually or for other reasons.
- Define a pivot table that links multiple pivot tables, possibly from different subject areas. The ability to link tables visually gives you greater control over layout.

  The linked pivot table can combine the tables horizontally, vertically, or by nesting them.

5.3.3 Alerts

A pivot table can have alerts. For example:

	Female	Male	Male/Female Ratio
Age Bucket	Patient Count	Patient Count	
0-9	695	719	1.03
10-19	691	702	1.02
20-29	686	670	0.98
30-39	759	789	1.04
40-49	794	742	0.93
50-59	583	559	0.96
60-69	344	351	1.02
70-79	349	243	0.70
80+	221	103	0.47

The definition of an alert includes the following parts:

- The cells to consider

- A value range, which can be open-ended
- An alert display style, which can include color and font information, as well as an image to display rather than the value

An alert examines each data cell in the pivot table, checks whether the value in the cell falls into the alert range, and if so, applies the alert display style to that cell.

When a pivot table is displayed, the system uses each alert, one at a time, starting with the first alert. For a given cell, if multiple alerts apply to the cell, the last alert takes precedence.

5.3.4 User Options

For a pivot table, you can specify the following user options:

- Default detail listing
- Drill options and their order
- Drill-down options
- Associated pivot tables that are available via right-click
- Options to launch when a user clicks a cell that has an alert

5.4 Options for Charts

For a pivot table shown in chart format, you can control the relevant properties from the previous sections as well as features such as the default chart type, and the color and fill patterns used in the chart.

You can also specify which dimensions are available as drill-downs via double-click or the context menu (right-click menu).

5.5 Options for Dashboards

Dashboards are created in the DeepSee Designer. This section summarizes the features of DeepSee dashboards.

5.5.1 Available Dashboard Elements

A dashboard can include some or all of the following elements:

- Pivot tables
- Detail listings
- Speedometers, which display KPIs (key performance indicators) that you have defined
- Labels, which can display static text, values of KPIs, or values returned by Caché ObjectScript expressions
- Images
- Buttons, drop-down lists, text boxes, check boxes, and other user interface controls to use as filters for the pivot tables or detail listings
- Frames that display Web pages or other dashboards

- DeepSee components such as the Analyzer

You control the size and position of all these elements.

5.5.2 Other Dashboard Properties

Also, you can specify properties of the dashboard such as the following:

- Background color or background image
- Dimensions of the dashboard
- Fonts used in the dashboard
- Password needed to display the dashboard, for added security
- Refresh timer so that DeepSee periodically reruns the underlying queries and refreshes one or more elements displayed in the dashboard

5.5.3 User Options

You specify any post actions on a dashboard. A post action can do any of the following:

- Display another dashboard, possibly in a new window
- Display a Web page in another browser window
- Display a small child window that displays the value of a KPI
- Execute a custom script

5.6 Runtime Parameters

The DeepSee *query variable* mechanism enables you to pass runtime values from your application into the dashboards. This mechanism works as follows:

- You can refer to variables within any DeepSee filters, as well as within several kinds of labels in the dashboards.
- When your application opens a dashboard, it does so by accessing the URL for the dashboard. The URL string can include values for any variables used in that dashboard.

It is not necessary to declare these variables; they are ordinary Caché ObjectScript variables.

Important: DeepSee includes a utility to set default values for query variables; this is called the Query Variable module. InterSystems recommends that you ignore this module, which does not provide sufficient functionality.

5.7 Custom Code

You can include Caché ObjectScript expressions within many modules throughout DeepSee.

Some expressions are evaluated at build time, some at runtime, and some when data is loaded. The following subsections list some of the places where you can use Caché ObjectScript. For a complete list, see *Expressions and Scripts in DeepSee*.

5.7.1 Build Time

You can use Caché ObjectScript expressions that are evaluated when the indices and fact table are built or updated. These can affect many model elements, including the following:

- Definitions of dimensions, measures, and listing fields
- Values of range definitions for a dimension

5.7.2 Runtime

You can include Caché ObjectScript expressions to be evaluated at runtime in locations such as the following:

- Within filters (for a reminder of the places in which you can use filters, see the section "Runtime Parameters," earlier in this book)
- Within the definitions of calculated measures and computations
- Within the definitions of KPIs
- As the default values for controls in dashboards
- As text in dashboards

5.7.3 Load Time

If you use the Connector, you can include Caché ObjectScript expressions in the following locations:

- To use as the value for a property when loading the data
- To determine which rows to reject while loading the data

Index

L

M

N

P

Q

R

S

T

U

V