Oracle® Real-Time Decisions

New Features Guide

Version 2.2.1

E12098-01

February 2008

ORACLE®

Oracle Real-Time Decisions New Features Guide, Version 2.2.1

E12098-01

Primary Author: Mike Meditzky

Contributors: Michel Adar, Lalitha Balachandra, Nicolas Bonnet, Paul Huang, Don Madison, Jason Wu, Ron Yang

Contents

Preface

This document describes the new features in Oracle Real-Time Decisions (Oracle RTD) Version 2.2.1.

Audience

This document is intended for the following Oracle RTD users:

- Technical users configuring Inline Services using Decision Studio
- Business users of Decision Center
- Administrators

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, 7 days a week. For TTY support, call 800.446.2398. Outside the United States, call +1.407.458.2479.

Related Documents

For more information, see the following documents in the Oracle Real-Time Decisions Version 2.2 documentation set:

- *Oracle Real-Time Decisions Installation and Administration of Oracle RTD*
- *Oracle Real-Time Decisions Decision Studio Reference Guide*
- *Oracle Real-Time Decisions Decision Center User Guide*
- *Oracle Real-Time Decisions Release Notes*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 General Features

This chapter describes the following new general features in Oracle Real-Time Decisions (Oracle RTD), Version 2.2.1:

1.1 IE7 Compliance

Oracle RTD 2.2.1 is fully compatible with Microsoft Internet Explorer 7.

For more information about the browsers supported by Oracle RTD, see *System Requirements and Supported Platforms*.

1.2 Quick Zip Installation

Oracle RTD Version 2.2.1 provides a fast way to set up and configure the Oracle RTD system. Oracle RTD Version 2.2.1 includes a zip file, **2.2.1_RTD_OC4J_QuickZip.zip**, that includes all the components required (except the database) to install Oracle RTD on a computer running Microsoft Windows.

The zip file includes a fully documented set of instructions, which describes the steps required to perform the installation. Make sure that you do not install into a path that contains spaces.

Caution: Because the quick zip installation installs standalone OC4J, which cannot be clustered, it is not recommended for production systems.

2

Dynamic Choices

This chapter describes Dynamic Choices, as implemented in Oracle Real-Time Decisions (Oracle RTD), Version 2.2.1. It contains the following topics:

2.1 Dynamic Choices Overview

In Oracle RTD, Choices represent the universe of alternatives, from which Oracle RTD can select its recommendations, such as the best offer in a cross selling application.

In previous releases of Oracle RTD, the only type of Choices available were Static Choices, where the Choice values to present to the requesting application or self-learning model were completely defined within Oracle RTD.

Static Choices are useful in cases where the number and values of the Choice options are known in advance, and are constant over a period of time.

In version Oracle RTD Version 2.2.1, Choices can now be either Static or Dynamic.

Dynamic Choices take their values from data retrieved from external Data Sources, typically database tables or views. This allows for the management of Choices to be done at the source system, such as Choices based on offers defined in an offer management system.

Each time a Dynamic Choice needs to be evaluated by Oracle RTD, the data is retrieved from the external Data Source or Sources. Therefore, the Choices to be presented to an application may vary over time, but always reflect the up-to-date state of the application data. They are not restricted to predefined, static values, changes to which would require a redeployment of the Oracle RTD Inline Service.

Note: While this section focuses on Dynamic Choices, in Oracle RTD Version 2.2.1, a Choice Group can contain a combination of Static and Dynamic Choices.

A Decision, as in previous releases, can be associated with one or more Choice Groups, no matter what types of Choice they contain.

2.2 Simple Example of Dynamic Choices

As a simple example, take the case of an **Insurance_Proposals** table, as shown in Figure 2–1. This table contains rows for different insurance products, as identified by the common value **InsuranceProducts** in the **ChoiceGroupId** column. The column that categorizes or groups the Dynamic Choices is an important required key identifier for setting up Dynamic Choices.

Each row in the group shows a different type of insurance product being offered, such as **AutoInsurance**, and **DisabilityInsurance**. Each row represents a Dynamic Choice.

One column serves to identify the particular Dynamic Choice within the group. In this example, the **ChoiceID** column is the Dynamic Choice identifier column.

Other columns in the table, such as **ProfitMargin**, can be used by Oracle RTD in the evaluation process. These columns can also be sent back to the application as part of the Dynamic Choice recommendation, as a value for a defined Choice attribute.

Figure 2–1 Insurance Products in the Insurance_Proposals Table

ChoiceId	ChoiceGroupId	ProfitMargin
AutoInsurance	InsuranceProducts	0.5
DisabilityInsurance	InsuranceProducts	0.5
HealthInsurance	InsuranceProducts	0.5
HomeownersInsurance	InsuranceProducts	0.5
LifeInsurance	InsuranceProducts	0.5

In short, the setup process is that, in Oracle RTD, you set up a Choice Group for Dynamic Choices, and associate this Choice Group with the required external Data Source or Sources. The Dynamic Choices are then available to be recommended by Oracle RTD.

After sufficient recommendations have been made and models have been updated for the corresponding Choice Group, you can analyze the performance of the various Dynamic Choices through the Oracle RTD Decision Center, as shown in Figure 2–2.

Figure 2–2 Decision Center Analysis of Dynamically Chosen Insurance Products

Distribution of Insurance Products

Insurance Products	Outcome	Count	%
AutoInsurance	Delivered	15707	100%
	Interested	11364	72%
	Purchased	4343	28%
DisabilityInsurance	Delivered	15714	100%
	Interested	11430	73%
	Purchased	4284	27%
HealthInsurance	Delivered	15722	100%
	Interested	11547	73%
	Purchased	4175	27%
HomeownersInsurance	Delivered	15724	100%
	Interested	11515	73%
	Purchased	4209	27%
LifeInsurance	Delivered	15735	100%
	Interested	11425	73%
	Purchased	4310	27%

2.2.1 Basic Dynamic Choice Design Implications

The basic design process for Dynamic Choices is similar to that for Static Choices. You must first set up a Choice Group, then define the required elements and parameters

for Dynamic Choices in the Choice Group. For more detailed information on how to perform the setups, see Section 2.5, "Setting up Dynamic Choices in Oracle RTD Decision Studio."

Using the Insurance_Proposals example, this section acts as an overview of the design process. It also introduces key terms used in the design process, as follows:

- The *set of all the Dynamic Choices* is identified as all the rows that have a common value in a "grouping" or categorizing column. In the Insurance_Proposals example, the categorizing column (or set identifier) is the **ChoiceGroupId** column.
- Each row in the database set represents a *single Dynamic Choice*. In the Insurance_Proposals example, the Dynamic Choice itself is identified by the unique value in the **ChoiceId** column.
- When you define the *Choice Group for Dynamic Choices* in Oracle RTD, you must link the Group to the set of rows that contain the Dynamic Choices.
- When you define the *Dynamic Choices in the Choice Group* in Oracle RTD, you must link each Dynamic Choice in the Group to the corresponding single Dynamic Choice row in the Data Source.

2.3 Multiple Category Dynamic Choices from a Single Data Source

In the simplest Dynamic Choice case, all the rows of the database table belong to the same category, that is, have the same value in a categorizing column.

You can provide different Dynamic Choices from either the same database table or a variety of data sources. The following example, as illustrated in Figure 2–3, shows the case where the Insurance_Proposals table is extended to provide Choices for both Insurance Products and Insurance Services.

Figure 2–3 Insurance Products and Insurance Services in the Insurance_Proposals Table

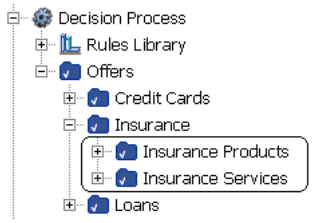
ChoiceId	ChoiceGroupId	ProfitMargin
AutoInsurance	InsuranceProducts	0.5
DisabilityInsurance	InsuranceProducts	0.5
HealthInsurance	InsuranceProducts	0.5
HomeownersInsurance	InsuranceProducts	0.5
LifeInsurance	InsuranceProducts	0.5
AutoServices	InsuranceServices	0.5
DisabilityServices	InsuranceServices	0.5
HealthServices	InsuranceServices	0.5
HomeownersServices	InsuranceServices	0.5
LifeServices	InsuranceServices	0.5

For this situation, you set up two Choice Groups in Oracle RTD, making both sets of data available for recommendations in the application.

After sufficient recommendations have been made and models have been updated for the corresponding Choice Group, you can analyze the performance of either or both of the Insurance Products and Insurance Services Dynamic Choices.

For example, the Choice Groups could have been set up as two groups in a group hierarchy, and available for analysis in the Oracle RTD Decision Center as shown in Figure 2–4.

Figure 2–4 Choice Groups in the Decision Center



Analyzing the Insurance Products provides the same results as shown in Figure 2–2. Figure 2–5 shows an equivalent analysis report for Insurance Services.

Figure 2–5 Decision Center Analysis of Dynamically Chosen Insurance Services

Distribution of Insurance Services

Insurance Services	Outcome	Count	%
AutoServices	Delivered	104	100%
	Interested	75	72%
	Purchased	29	28%
DisabilityServices	Delivered	103	100%
	Interested	71	69%
	Purchased	32	31%
HealthServices	Delivered	104	100%
	Interested	78	75%
	Purchased	26	25%
HomeownersServices	Delivered	104	100%
	Interested	74	71%
	Purchased	30	29%
LifeServices	Delivered	103	100%
	Interested	74	72%
	Purchased	29	28%

2.3.1 Considerations for Different Dynamic Choice Categories in the Same Data Source

For each real-world category, you must set up a separate Choice Group, usually, but not necessarily, in a Choice Group hierarchy.

The design considerations for and components of each Choice Group are the same as described in Section 2.2.1, "Basic Dynamic Choice Design Implications."

For general information on how to set up Choice Groups, see Section 2.5, "Setting up Dynamic Choices in Oracle RTD Decision Studio."

For specific information on how to set up different Choice Groups from the same Data Source, see Section 2.5.7, "Creating a Multi-Category Choice Group."

2.4 Prerequisite External Data Source

The data required for Dynamic Choices exists in an external Data Source.

For the sake of simplicity, the following description assumes that the external Data Source is a database table or view in the calling application.

To be useful for Dynamic Choices, the data must include:

- One column to be used for categorizing and extracting the data.

For a single Dynamic Choice, the rows to be extracted will all have the same value in the categorizing column, and this column is used to control the extraction.

For example:

- The database table **Special_Events** has a column **Event_Type**.
- There are three distinct values of **Event_Type** across all the rows, namely **Promotion**, **Product_Launch**, and **Mailshot**.

In this example, **Event_Type** is the categorizing column, and for a single Dynamic Choice, Oracle RTD will extract all the rows of one event type, such as all the **Promotion** rows.

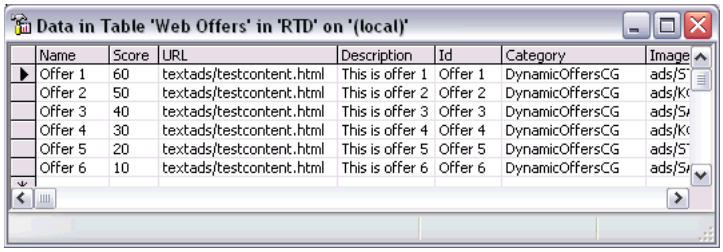
- One column that uniquely identifies the rows extracted for a particular Dynamic Choice.

  The column does not need to have unique values across all the rows, just within the extracted data set.

  Any column that provides a unique identifier within the extracted data is sufficient. Oracle recommends that the column values include some textual component. These values appear as headers for some Oracle RTD Decision Center reports, and an identifier that is meaningful in the real world sense is more useful than a strictly numeric identifier.

Figure 2–6 is an example of a database table Web Offers, that could be used as the external data source for a Dynamic Choice data source. This table is not released with Oracle RTD.

Figure 2–6 Example of an External DatabaseTable



Name	Score	URL	Description	Id	Category	Image
Offer 1	60	textads/testcontent.html	This is offer 1	Offer 1	DynamicOffersCG	ads/5
Offer 2	50	textads/testcontent.html	This is offer 2	Offer 2	DynamicOffersCG	ads/K
Offer 3	40	textads/testcontent.html	This is offer 3	Offer 3	DynamicOffersCG	ads/S
Offer 4	30	textads/testcontent.html	This is offer 4	Offer 4	DynamicOffersCG	ads/K
Offer 5	20	textads/testcontent.html	This is offer 5	Offer 5	DynamicOffersCG	ads/5
Offer 6	10	textads/testcontent.html	This is offer 6	Offer 6	DynamicOffersCG	ads/S

The table illustrates the following features:

- The categorizing column is **Category**, and the common value in all the **Category** columns is **DynamicOffersCG**.
- You could select either **Name** or **ID** as the Dynamic Choice identifier column for the DynamicOffersCG category.

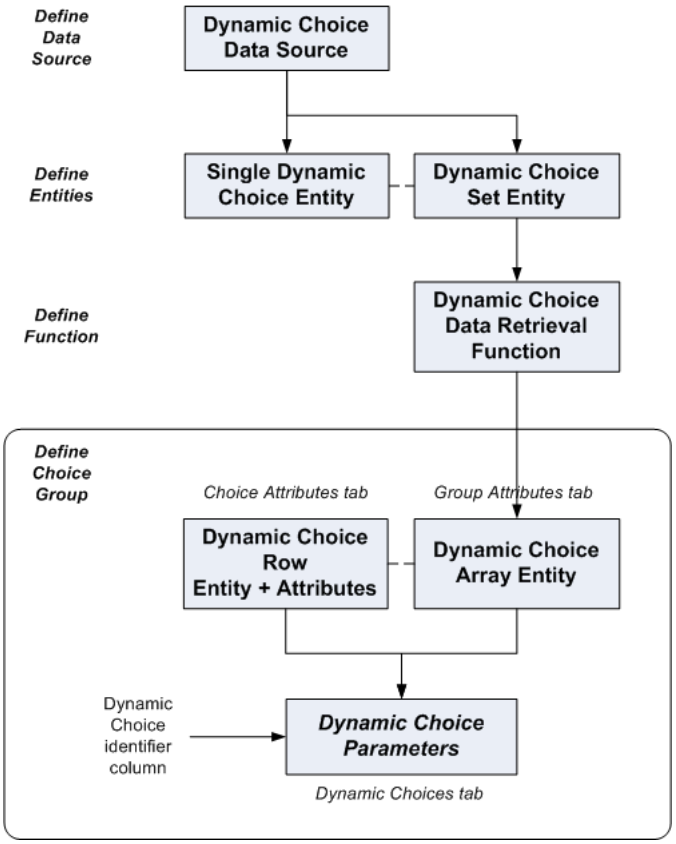
2.5 Setting up Dynamic Choices in Oracle RTD Decision Studio

This section consists of the following topics:

- Section 2.5.6, "Creating a Single Category Choice Group"
- Section 2.5.7, "Creating a Multi-Category Choice Group"

Figure 2–7 shows an overview of setting up a simple, single category Choice Group for Dynamic Choices. The elements in the diagram are referred to in the more detailed process descriptions that appear later in this section.

Figure 2–7 Overview of Setup Process for Single Category Dynamic Choices



Note: The diagrams and Oracle RTD Decision Studio screenshots illustrating the setup process, which appear later in this section, are based on the DC_Demo Inline Service that is released with Oracle RTD Version 2.2.1.

2.5.1 Creating the Dynamic Choice Data Source

To create the Dynamic Choice Data Source:

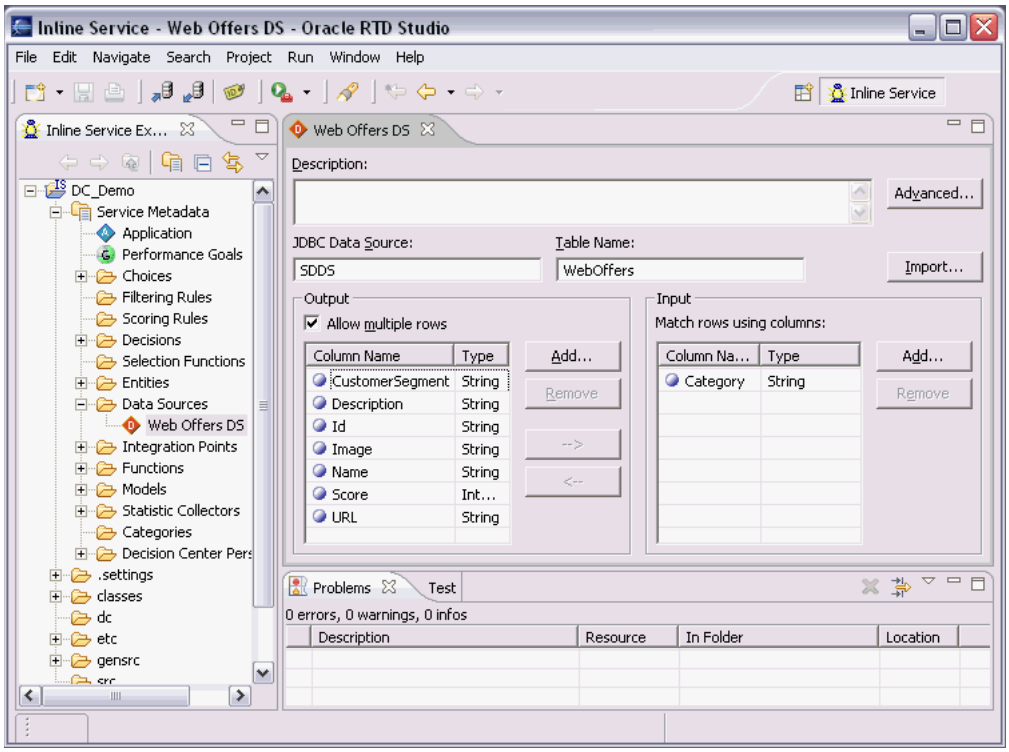
1. Create a new Data Source that maps to the table described in Section 2.4, "Prerequisite External Data Source," using the **Import** button to point to the external data source.
2. In the Output column area, check **Allow multiple rows**, and select all the columns that you require for a Dynamic Choice.

   In the Input column area, select the column that contains the common value that categorizes and groups the Dynamic Choice rows.

Note: You do not have to select the Dynamic Choice identifier column from among the Output columns at this stage.

Figure 2–8 shows how the Data Source **Web Offers DS** is set up from the table **SDDS.WEBOFFERS**, with **Category** as the Input identifier, and a number of other columns that represent attributes of the Dynamic Choice itself.

Figure 2–8 Defining the Web Offers DS Data Source



2.5.2 Creating the Single Dynamic Choice Entity

The Dynamic Choice data exists in the Data Source. You must create a Single Dynamic Choice Entity in Oracle RTD that consists of all the information associated with a particular category, but not the category itself.

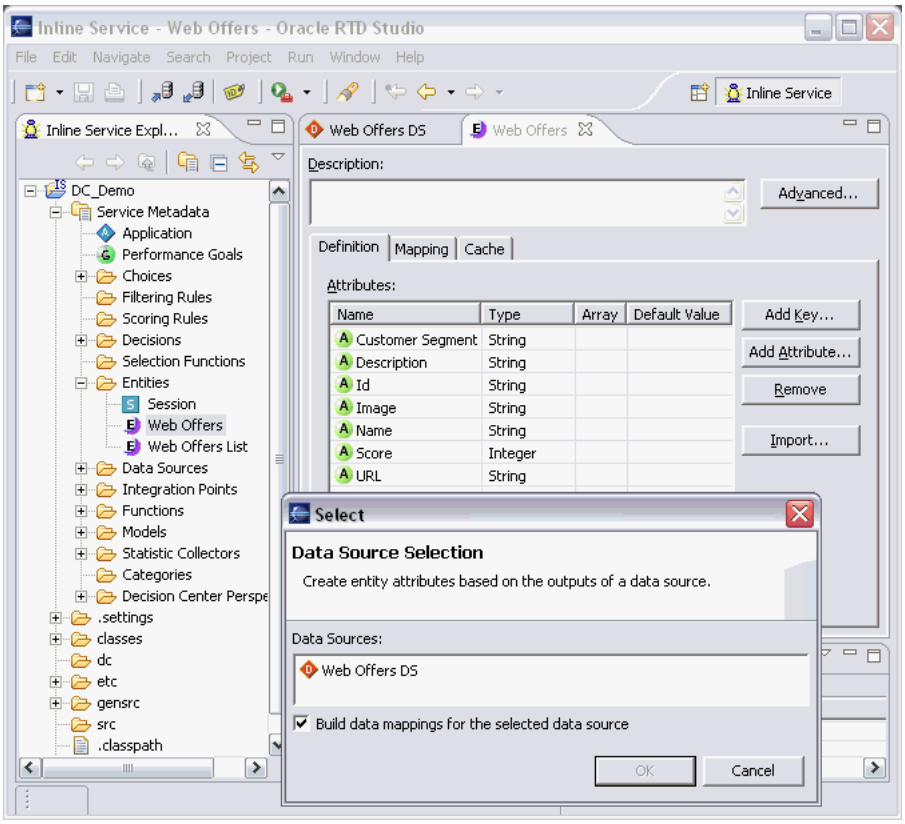
In terms of the Data Source that you created, the Entity attributes for the Single Dynamic Choice Entity are the Output attributes of the Data Source.

To create the Single Dynamic Choice Entity:

1. Create an Entity for the Dynamic Choice data, using the **Import** functionality to bring in all the Output columns from the Data Source described in Section 2.4, "Prerequisite External Data Source."
2. When selecting the Data Source, be sure to uncheck the **Build data mappings for the selected data source** option found in the Select window that appears when you import.

Figure 2–9 shows the Definition tab for the setup of the Single Dynamic Choice Entity **Web Offers**. The attributes are the Output columns of the Data Source **Web Offers DS**.

Figure 2–9 Defining the Web Offers Entity



2.5.3 Creating the Dynamic Choice Set Entity

In addition to the Single Dynamic Choice Entity, you must create a Dynamic Choice Set Entity, that includes the following Attributes:

- A Key Attribute, which is the input, categorizing column of the Data Source that contains the Dynamic Choice data
- An array Attribute that stores the Single Dynamic Choice Entity data

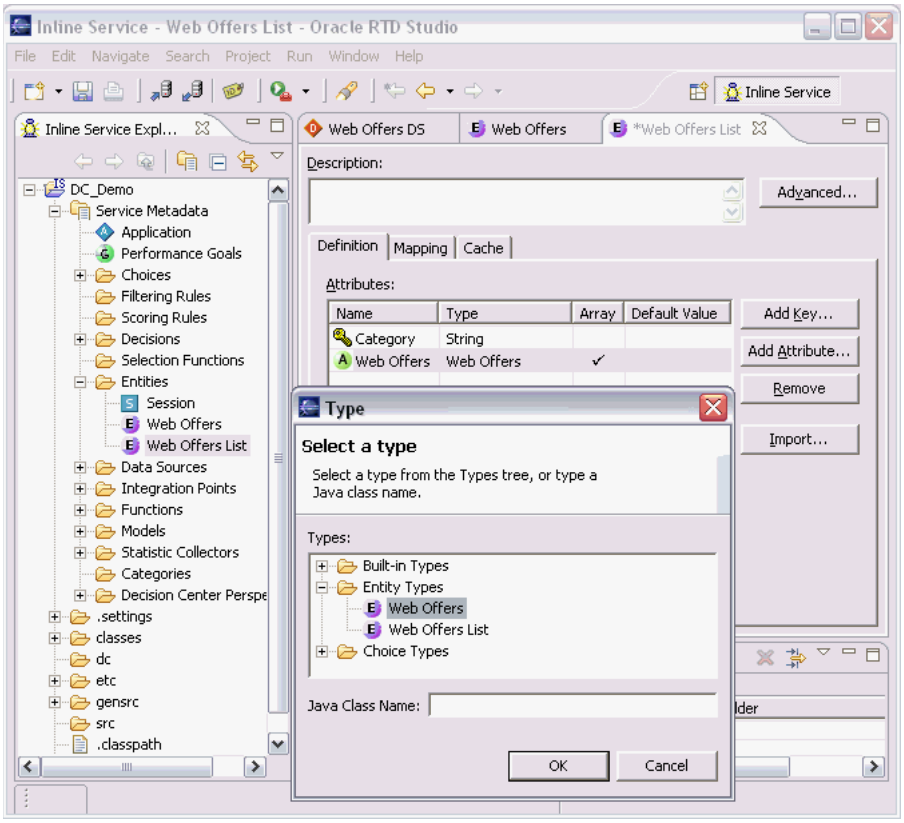
  This array Attribute must be of the same Entity type as the Entity that you created in Section 2.5.2, "Creating the Single Dynamic Choice Entity." This array is the container for all the Attributes of the data to be extracted from the Data Source required for the Dynamic Choice except for the categorizing Attribute.

To create the Dynamic Choice Set Entity:

1. Create an Entity in Oracle RTD.
2. For the key Attribute, click **Add Key**, and select the Dynamic Choice categorizing Attribute from the Data Source.
3. Create an Attribute whose type is the name of the Entity created in Section 2.5.2, "Creating the Single Dynamic Choice Entity."
4. Mark this entity-type Attribute as an **Array**.

   Figure 2–10 shows the Definition tab for the setup of the Dynamic Choice Set Entity **Web Offers List**. The Key Attribute is the Input column **Category** of the Data Source **Web Offers DS**, and the second Attribute is an array Attribute of type **Web Offers**.

Figure 2–10 Defining the Dynamic Choice Set Entity Web Offers List

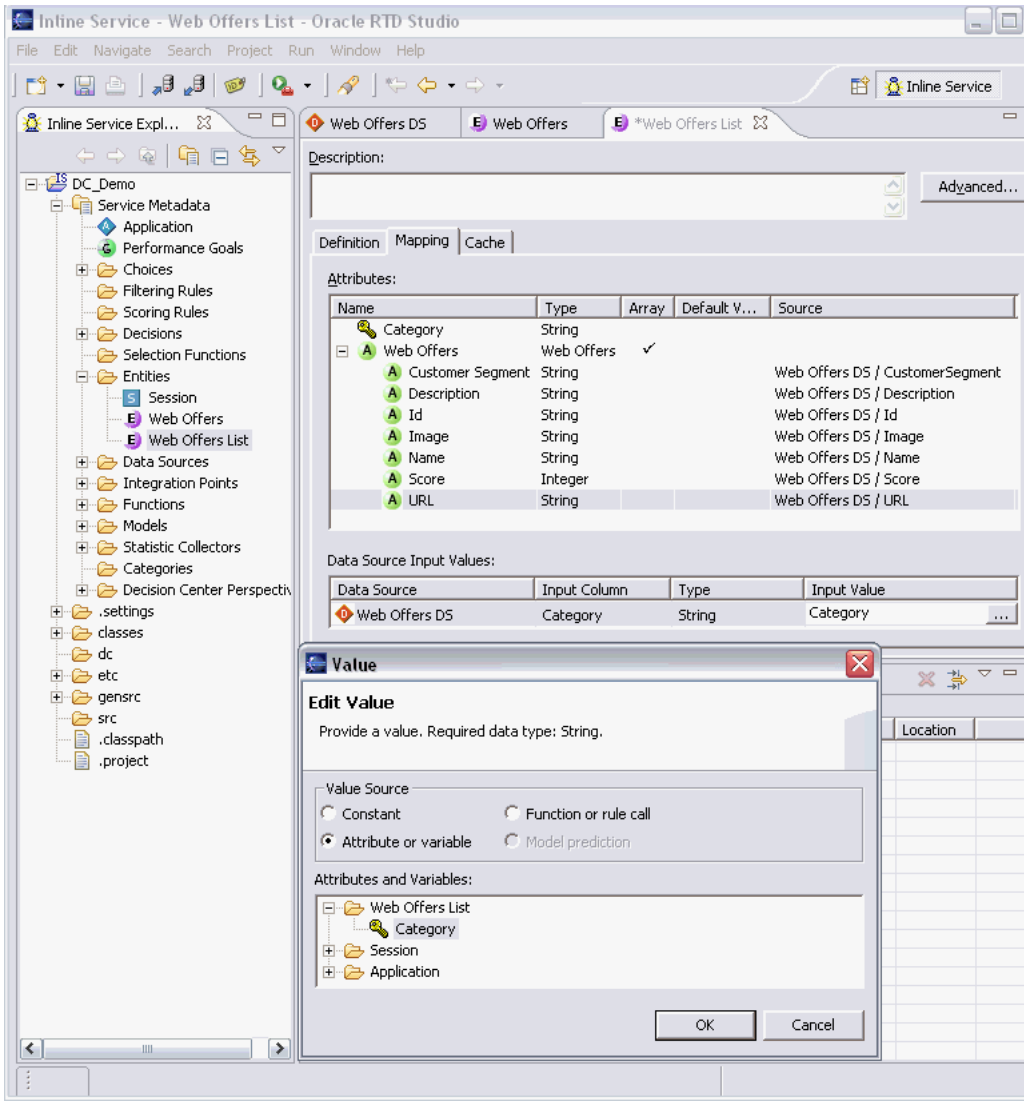


5. Click the Mapping tab, and map each Attribute within the entity-type Attribute to the appropriate column in the original Data Source.

6. In the Data Source Input Values region, for the Input Value of the Data Source, select the Dynamic Choice categorizing Attribute that you created in step 2.

   Figure 2–11 shows the Mapping tab for the setup of the Dynamic Choice Set Entity **Web Offers List**. Each of the Attributes within the array Attribute is mapped to the corresponding column in the **Web Offers DS** Data Source. In the Data Source Input Values region, the Attribute selected for the Input Value is the key Attribute **Category**.

Figure 2–11 Mapping the Web Offers Attributes in the Web Offers List Entity



7. Click the Cache tab.
8. Select the check box to **Enable caching for this entity type**.

> **Note:** It is important to enable caching on the Dynamic Choice Set Entity. Enabling caching will keep the Oracle RTD Decision Server from repeatedly pulling the Dynamic Choices from the data source with each new session.

2.5.4 Creating the Dynamic Choice Data Retrieval Function

To extract the Dynamic Choice data from the database, you must create a Function that will perform the data retrieval. This function will be called by the Choice Group that you will create in the steps that follow. The properties of the Function are as follows:

- The Function returns a value.

- The return value is of type Array.
- The Data Type of the array elements is the Single Dynamic Choice entity that you created previously
- The Function has a Parameter that is the same as the Data Source Input Value of the Dynamic Choice Set Entity that you created previously.
- The logic of the Function instantiates a new occurrence of the Dynamic Choice Set Entity, and uses the Parameter to retrieve the Dynamic Choice data into the array.

To create the Dynamic Choice Data Retrieval Function:

1. Create the Function, and select the **Return value** check box.
2. Select the **Array** option, to ensure that the return value is of type Array.
3. For the **Data Type**, select the name of the entity that you created in Section 2.5.2, "Creating the Single Dynamic Choice Entity."
4. In the Parameters area, add the **Name** and **Type** of the Key attribute that you created in step 2 of Section 2.5.3, "Creating the Dynamic Choice Set Entity."
5. In the **Logic** field, enter code similar to the following, adapting it as required for the names of your entities and attributes:

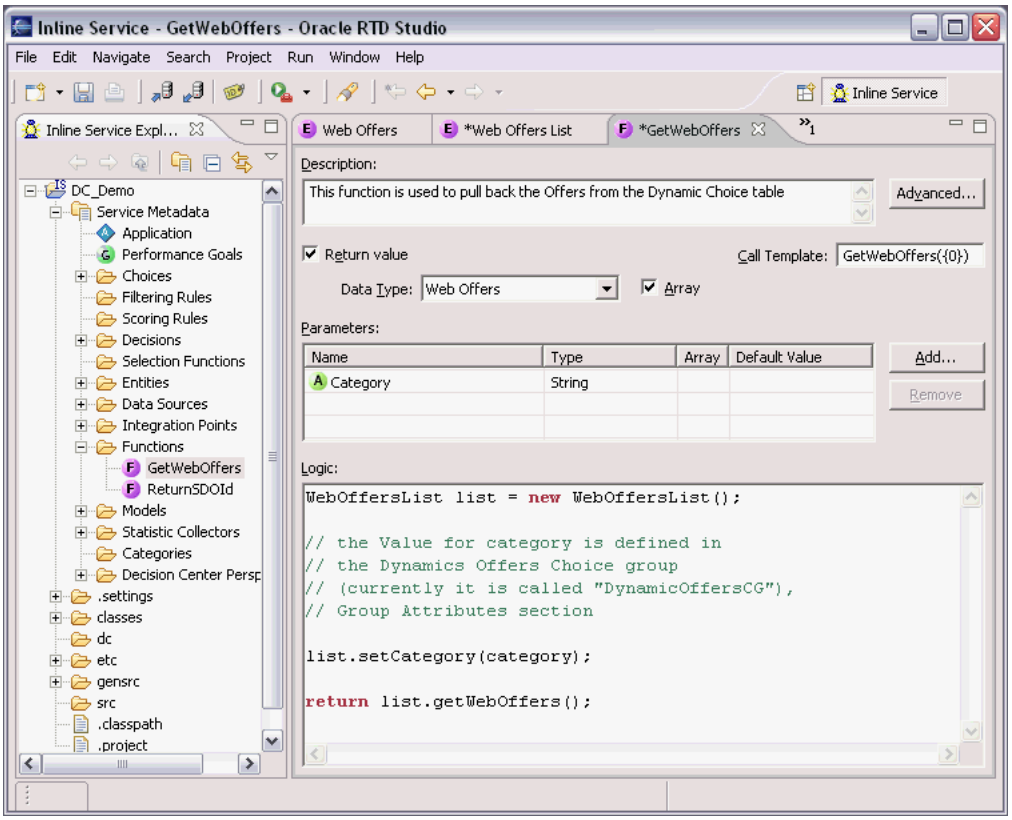
```
WebOffersList list = new WebOffersList();
list.setCategory(category);
return list.getWebOffers();
```

   where:

   - **WebOffersList** is the object name, with internal spaces deleted, for the Entity created in Section 2.5.3, "Creating the Dynamic Choice Set Entity."
   - **list.setCategory** references the Entity key that you created in Section 2.5.3, "Creating the Dynamic Choice Set Entity," step 2.
   - **getWebOffers()** refers to the Entity created in Section 2.5.2, "Creating the Single Dynamic Choice Entity," that is mapped inside the Dynamic Choice Set Entity.

     Figure 2–12 shows the definition of the **GetWebOffers** Function.

Figure 2–12 Defining the GetWebOffers Function



2.5.5 Considerations for Choice Group Design

Dynamic Choices enable application data administrators to control the choices that Oracle RTD recommends to the application. Unlike Static Choices, Dynamic Choices may be added, edited, and deleted in the application tables without requiring any changes in the interfacing Oracle RTD Inline Service.

If there is a requirement to have both type of Choice in a single Inline Service, Oracle recommends that Static Choices and Dynamic Choices are clearly separated in the designing of the Choice Groups. This section concentrates on the design of Choice Groups for Dynamic Choices.

The main real world conditions that affect the design of Choice Groups for Dynamic Choices are:

- Single Dynamic Choice Category in a database table
- Multiple Dynamic Choice Categories in a database table

Single Choice Group

Where there is a single Dynamic Choice category in a database table (such as Promotion in a Special_Events table), then the recommended design strategy is:

1. Design a single Choice Group.
2. Enter and select the required parameters in each of the following tabs for the Choice Group: Group Attributes tab, Choice Attributes tab, Dynamic Choices tab.

In Oracle RTD Decision Studio, this Choice Group has no subgroups.

Choice Group Hierarchy

For multiple categories in a database table (such as Insurance Products and Insurance Services in an Insurance_Proposals table), design a Choice Group hierarchy. Typically, all that is required is a two-level hierarchy, as follows:

1. For the top-level Choice Group, enter and select the required parameters in the Choice Attributes tab, but not the Group Attributes tab, nor the Dynamic Choices tab.
2. For each separate Dynamic Choice category, specify one lower-level Choice Group. In each of the lower-level Choice Groups, enter and select the required parameters in the Group Attributes tab and the Dynamic Choices tab, but not in the Choice Attributes tab.

> **Note:** You only need to fill in Dynamic Choices tab parameters in the lowest-level Choice Groups of a multi-level Choice Group hierarchy.

In Oracle RTD Decision Studio, the lower-level Choice Groups have no subgroups.

2.5.6 Creating a Single Category Choice Group

To use Dynamic Choices, you must create one or more Choice Groups. Where the Dynamic Choices refer to data that belongs to one type or category, create a single category Choice Group.

> **Note:** In Oracle RTD Decision Studio, when you create a Choice Group for Dynamic Choices, the individual Dynamic Choices do not appear in any of the RTD Decision Studio windows.
>
> In Oracle RTD Decision Center reports, you can see all the Dynamic Choices which satisfy the following conditions:
>
> - They have been returned by Decisions called by the front-end applications.
> - They have had RTD models updated for those Choices.

In Oracle RTD Decision Studio, the Choice Group is configured to be able to extract the Choices dynamically at runtime through options that you set up in the following tabs:

- Group Attributes tab
- Choice Attributes tab
- Dynamic Choices tab

These are the main tabs where you configure Dynamic Choices.

> **Note:** You can also use the Choice Eligibility tab, to filter the Dynamic Choice data as it is extracted from the Data Source.
>
> Eligibility rules created for a Dynamic Choice are shared across all Choices retrieved for the same Dynamic Choice Group.

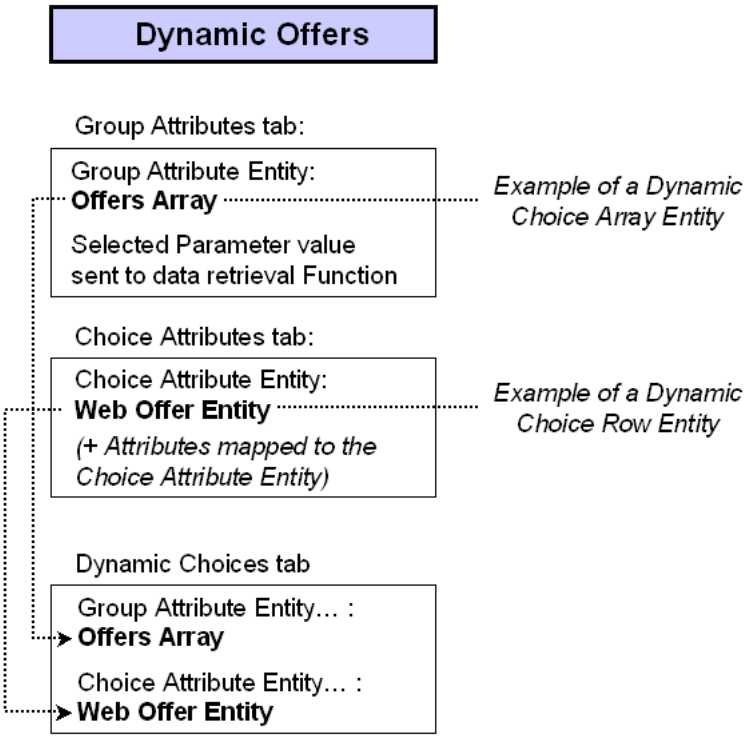
Figure 2–13 shows an example of the main elements required to set up a single category Choice Group, **Dynamic Offers**.

The Group Attribute setup indicates that all the data to be retrieved for the Dynamic Choices will be of one category only, and you must specify the exact category here.

The Choice Attribute setup describes the individual attributes that will be retrieved.

The Group and Choice Attributes are then referenced in the Dynamic Choices tab for this single category Choice Group.

Figure 2–13 Defining the Choice Group Dynamic Offers



2.5.6.1 Group Attributes Tab

In the Group Attributes tab, you specify an array Attribute of the same Entity type as that which you created in Section 2.5.2, "Creating the Single Dynamic Choice Entity." This Attribute is referred to as the Dynamic Choice Array Entity in Figure 2–7, which shows an overview of the single category Dynamic Choice setup process.

At this level, you also specify the Function that retrieves the Dynamic Choice data. You must choose a value for the Function parameter. This enables the function to retrieve just the Dynamic Choice data relevant for one particular real world type or category.

To create a Choice Group and specify the Group Attributes:

1. Create a Choice Group.
2. Click the Group Attributes tab.
3. Create a new entity-type Group Attribute (the Dynamic Choice Array entity), whose type is the name of the entity that you created in Section 2.5.2, "Creating the Single Dynamic Choice Entity."

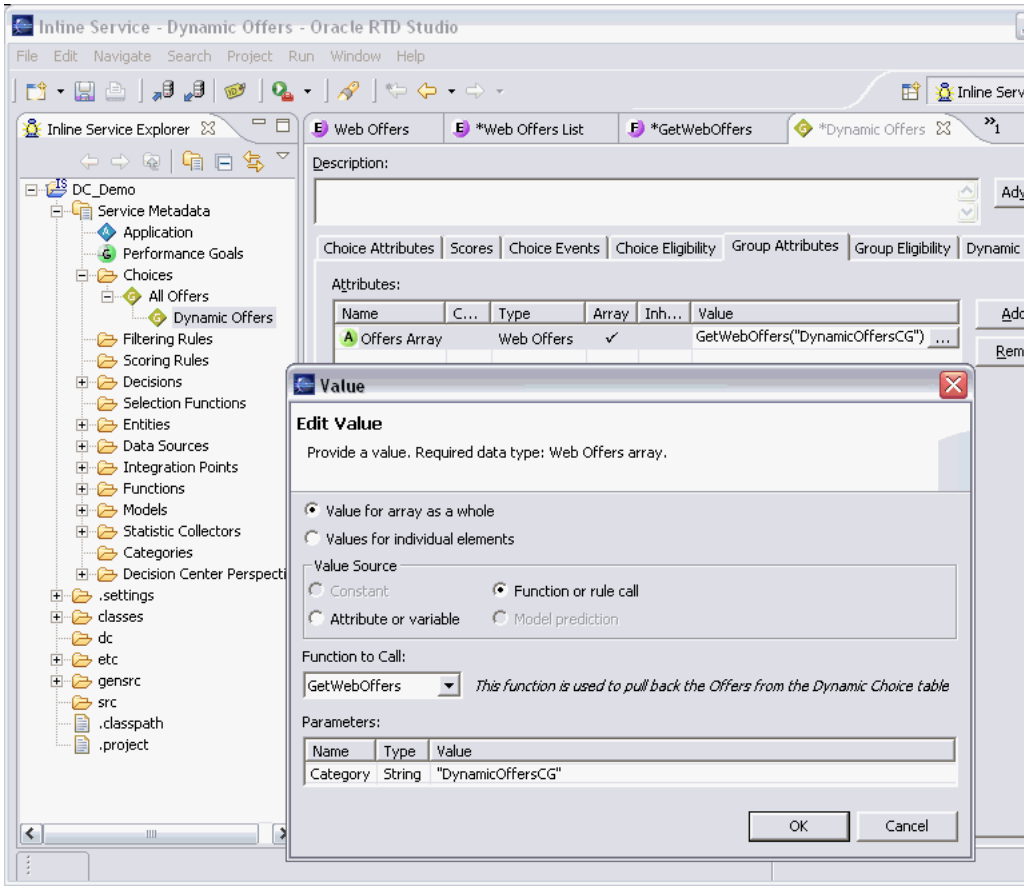
4. Specify that this Attribute is an **Array**.
5. Click the right-hand end of the **Value** box to expose the ellipsis (...) button, then click the ellipsis button to open the Value window.
6. In the Value window, select the option **Value for array as a whole**.
7. For Value Source, select **Function or rule call**, then select the Function that you created in Section 2.5.4, "Creating the Dynamic Choice Data Retrieval Function."
8. In the Parameters area, choose the **Value** of the parameter that will retrieve the corresponding rows in the Data Source whose Input Attribute contains that value.

Note: This string Value is the exact value in the database that categorizes all the Dynamic Choice rows for a Choice Group.

For example, for a Choice Group set up for the Insurance_Proposals table as described in Section 2.2, "Simple Example of Dynamic Choices," the Value is **InsuranceProducts**.

Figure 2–14 shows the Group Attributes tab for the Choice Group **Dynamic Offers**. The Function to call is **GetWebOffers**. The Value in the Parameters area is the string **DynamicOffersCG**.

Figure 2–14 Defining the Group Attributes for the Choice Group Dynamic Offers



2.5.6.2 Choice Attributes Tab

In the Choice Attributes tab, you must:

- Specify an entity-type Attribute of the same type as the Entity that you created in Section 2.5.2, "Creating the Single Dynamic Choice Entity."

  This Attribute is referred to as the Dynamic Choice Row entity in Figure 2–7, which shows an overview of the single category Dynamic Choice setup process.

- For each of the component attributes within this Dynamic Choice Row Entity, create a separate Choice Attribute, which you must then map to the corresponding attribute within the Dynamic Choice Row entity that you just created.

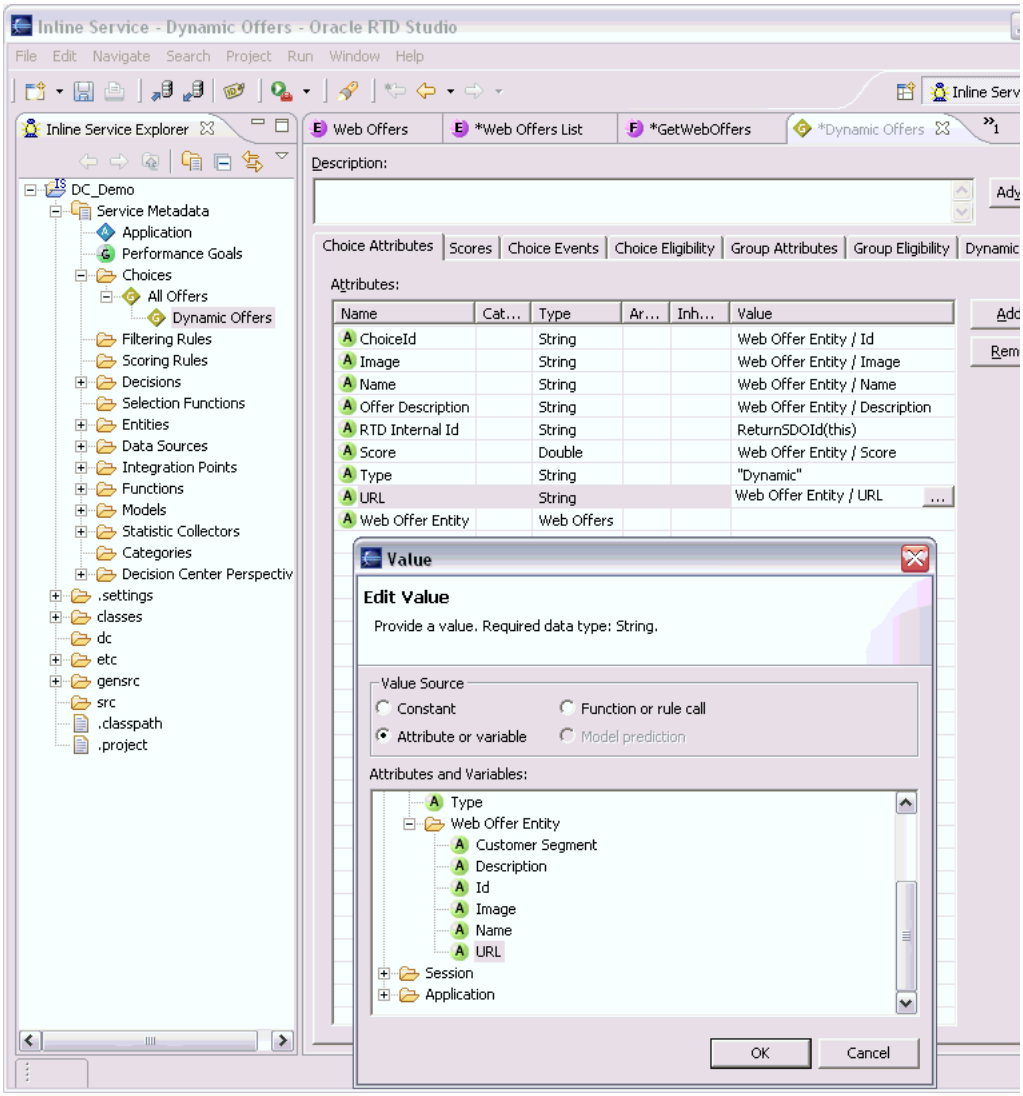
To specify the Choice Attributes of the Choice Group:

1. Click the Choice Attributes tab.
2. Create a new entity-type Attribute (the Dynamic Choice Row entity), whose type is the name of the entity that you created in Section 2.5.2, "Creating the Single Dynamic Choice Entity."
3. Ensure that the **Array** check box is not selected.
4. For each attribute of the new Dynamic Choice Row entity, create a corresponding Choice Attribute.
5. For each Choice Attribute created in the previous step, map its Value to the corresponding attribute within the Dynamic Choice Row entity that you created in step 2.

   Figure 2–15 shows the Choice Attributes tab for the Choice Group **Dynamic Offers**. The Choice Attributes are the following:

   - One Dynamic Choice Row Entity **Web Offer Entity**
   - Several other Attributes, each of whose Values derives from the corresponding Attribute of the Dynamic Choice Row Entity **Web Offer Entity**

Figure 2–15 Defining the Choice Attributes for the Choice Group Dynamic Offers



2.5.6.3 Dynamic Choices Tab

In the Dynamic Choices tab, you provide the following information:

- You explicitly select this Choice Group to be for Dynamic Choices.
- You specify the Group and Choice Attributes that you set up in the corresponding Group Attributes and Choice Attributes tabs.
- You select the Attribute that identifies each Dynamic Choice.
- You describe how you wish the Dynamic Choices to appear in Oracle RTD Decision Center reports. Because the number of Dynamic Choices could be considerable, you can choose to break up a potentially long list of Dynamic Choices into smaller units or "folders", and you indicate how you want the data grouped in the folders.

To specify the Dynamic Choice parameters:

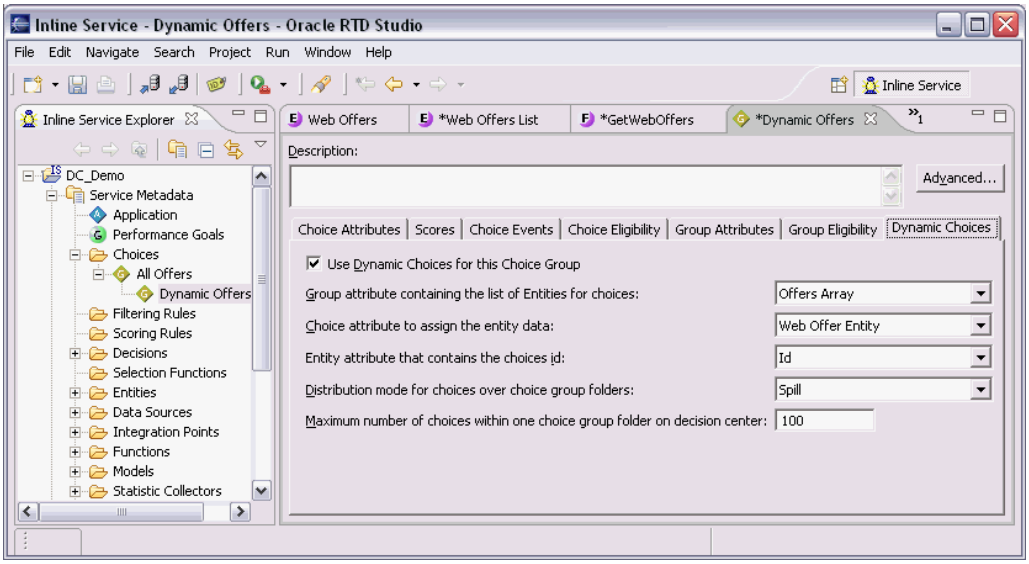
1. Click the Dynamic Choices tab.

2. Select the check box option to **Use Dynamic Choices for this Choice Group**.
3. For the **Group attribute containing the list of Entities for choices**, select the Dynamic Choice Array attribute that you created in Section 2.5.6.1, "Group Attributes Tab."
4. For the **Choice attribute to assign the entity data**, select the Dynamic Choice Row attribute that you created in Section 2.5.6.2, "Choice Attributes Tab."
5. For the **Entity attribute that contains the choices id**, select the Attribute that serves as the unique identifier for each of the extracted Dynamic Choice rows.
6. For the **Distribution mode for choices over choice group folders**, select Spill or Even.

> **Note:** For more information about this parameter, and the parameter in the following step, see Section 2.6.2, "Distribution of Choices Across Decision Center Folders."

7. Select the **Maximum number of choices within one choice group folder on decision center**.

Figure 2–16 Defining the Dynamic Choice Parameters for the Choice Group



2.5.7 Creating a Multi-Category Choice Group

To use Dynamic Choices, you must create one or more Choice Groups. Where you want to be able to select different groups of data from the same data source, create a multi-category Choice Group. This section describes the standard way to set up a multi-category Choice Group.

> **Note:** In Oracle RTD Decision Studio, when you create a Choice Group for Dynamic Choices, the individual Dynamic Choices do not appear.
>
> In Oracle RTD Decision Center reports, you can see all the Dynamic Choices that have been returned by Decisions to calling applications.

In RTD Decision Studio, a Choice Group is configured to be able to extract the Choices dynamically at run time through options that you set up in the following tabs:

- Group Attributes tab
- Choice Attributes tab
- Dynamic Choices tab

These are the main tabs where you configure Dynamic Choices.

> **Note:** You can also use the Choice Eligibility tab, to filter the Dynamic Choice data as it is extracted from the data source.
>
> Eligibility rules created for a Dynamic Choice are shared across all Choices retrieved for the same Dynamic Choice Group.

To allow for multiple Dynamic Choice categories, you must create a hierarchy of Choice Groups, and set up the Choice Group elements at different levels.

Figure 2–17 shows an example of the main elements required to set up a two-category Choice Group, **Incentive Choices**.

The Choice Group **Incentive Choices** is the parent Choice Group, with two child Choice Groups, **Discounts** and **Gifts**.

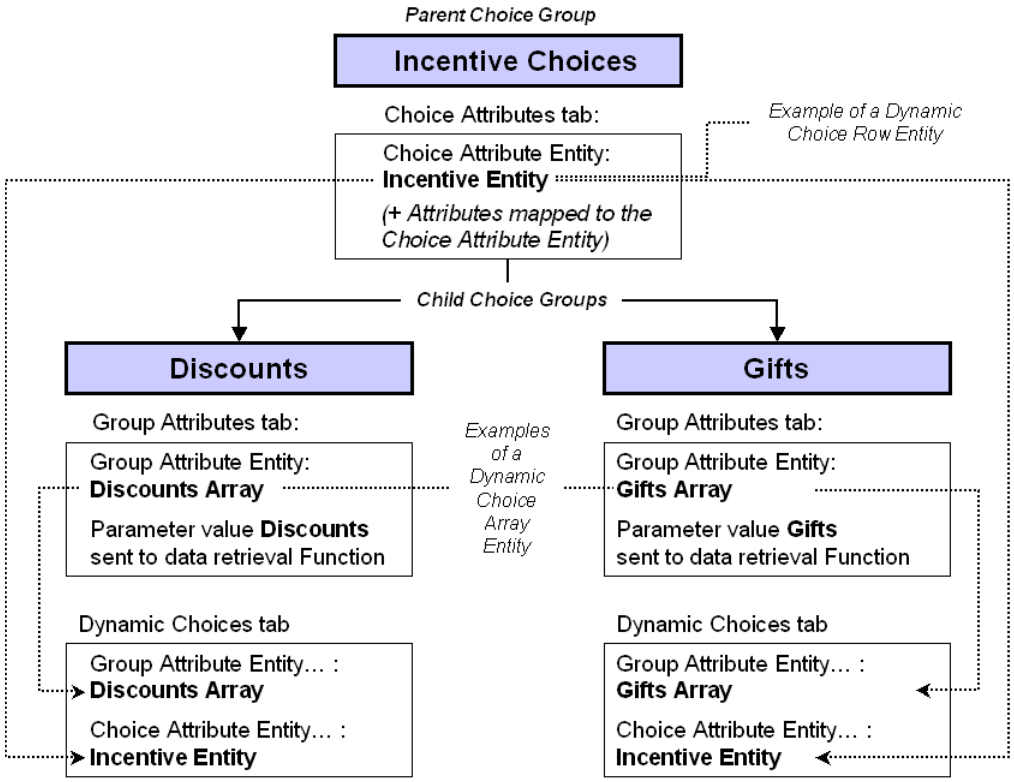
You specify the Choice Attributes at the top level, in the parent Choice Group. These Choice Attributes are then inherited by the two child Choice Groups.

> **Note:** In the parent Choice Group, or in any higher level Groups of a multi-level Choice Group hierarchy, you do not enter or select any values in the Dynamic Choices tab. Dynamic Choice parameters are only specified in the lowest level Group of any Choice Group hierarchy.

Each child Choice Group enables a different category set of data to be retrieved, through the Group Attributes setup. The Group and Choice Attributes are then referenced in the Dynamic Choices tab for both of the child Choice Groups.

To compare this process with the equivalent single category Choice Group setup, see Figure 2–7.

Figure 2–17 Example of Defining a Choice Group Hierarchy



2.5.7.1 Choice Attributes Tab in the Parent Choice Group

In the Choice Attributes tab, you specify an entity-type Choice Attribute of the same type as that which you created in Section 2.5.2, "Creating the Single Dynamic Choice Entity."

This Choice Attribute is also known as the Dynamic Choice Row Entity, as in the equivalent single category Dynamic Choice setup process shown in Figure 2–7.

For each of the attributes within this Dynamic Choice Row entity, create a separate Choice Attribute, which you must map to the corresponding attribute in the Dynamic Choice Row entity that you just created.

To create the Parent Choice Group and Choice Attributes:

1. Create the parent Choice Group.
2. Click the Choice Attributes tab.
3. Create a new entity-type Choice Attribute, whose type is the name of the entity that you created in Section 2.5.2, "Creating the Single Dynamic Choice Entity."
4. Ensure that you do not specify that this is an **Array**.
5. For each of the attributes of the new entity-type Choice Attribute, create a corresponding Choice Attribute.
6. For each of the attributes created in the previous step, map its Value to the corresponding attribute within the Choice Attribute that you created in step 3.

2.5.7.2 Group Attributes Tab in the Child Choice Groups

For each child Choice Group, in the Group Attributes tab, you specify an entity-type array Attribute of the same type as that which you created in Section 2.5.2, "Creating the Single Dynamic Choice Entity."

This Group Attribute is also known as the Dynamic Choice Array Entity, as in the equivalent single category Dynamic Choice setup process shown in Figure 2–7.

At this level, you also specify the function that retrieves the Dynamic Choice data. You must choose a value for the Function parameter. This enables the Function to retrieve just the Dynamic Choice data relevant for one particular real world type or category.

First, you need to create the child Choice Groups under the previously created parent Choice Group, then enter the required elements in the Group Attributes tab.

To create the Child Choice Groups and Group Attributes:

1. Create the first child Choice Group.
2. Create extra child Choice Groups as required, one for each separate Dynamic Choice category.

   Within each child Choice Group, you must now set up the required elements and parameters in the Group Attributes tab and the Dynamic Choices tab.

   The steps following in this section describe the actions required in the Group Attributes tab for each child Choice Group. Section 2.5.7.3 describes the actions required in the Dynamic Choices tab for each child Choice Group.
3. Click the Group Attributes tab of the child Choice Group.
4. Create a new entity-type Group Attribute, whose type is the name of the Entity that you created in Section 2.5.2, "Creating the Single Dynamic Choice Entity."
5. Specify that this Attribute is an **Array**.
6. Click the right-hand end of the **Value** box to expose the ellipsis (...) button, then click the ellipsis button to open the Value window.
7. In the Value window, select the option **Value for array as a whole**.
8. For Value Source, select **Function or rule call**, then select the Function that you created in Section 2.5.4, "Creating the Dynamic Choice Data Retrieval Function."
9. In the Parameters area, choose the **Value** of the parameter that will retrieve the corresponding rows in the Data Source whose Input attribute contains that value.

2.5.7.3 Dynamic Choices Tab in the Child Choice Groups

For each child Choice Group, in the Dynamic Choices tab, you must provide the following information:

- You explicitly select this Choice Group to be a Choice Group for Dynamic Choices.
- You specify the Group and Choice Attributes that you set up in the corresponding Group Attributes and Choice Attributes tabs.
- You select the Attribute that identifies each Dynamic Choice.
- You describe how you wish the Dynamic Choices to appear in Oracle RTD Decision Center reports. Because the number of Dynamic Choices could be considerable, you can choose to break up a potentially long list of Dynamic Choices into smaller units or "folders", and you indicate how you want the data grouped in the folders.

To specify the Dynamic Choice parameters:

1. Click the Dynamic Choices tab.
2. Select the check box option to **Use Dynamic Choices for this Choice Group**.
3. For the **Group attribute containing the list of Entities for choices**, select the attribute that you created in Section 2.5.7.2, "Group Attributes Tab in the Child Choice Groups."
4. For the **Choice attribute to assign the entity data** select the attribute that you created in Section 2.5.7.1, "Choice Attributes Tab in the Parent Choice Group."
5. For the **Entity attribute that contains the choices id**, select the Attribute that serves as the unique identifier for each of the extracted Dynamic Choice rows.
6. For the **Distribution mode for choices over choice group folders**, select Spill or Even.

> **Note:** For more information about this parameter, and the parameter in the following step, see Section 2.6.2, "Distribution of Choices Across Decision Center Folders."

7. Select the **Maximum number of choices within one choice group folder on decision center**.

2.6 Dynamic Choice Reporting in the Decision Center

This section consists of the following topics:

- Section 2.6.1, "Dynamic Choice Reporting Overview"
- Section 2.6.2, "Distribution of Choices Across Decision Center Folders"

2.6.1 Dynamic Choice Reporting Overview

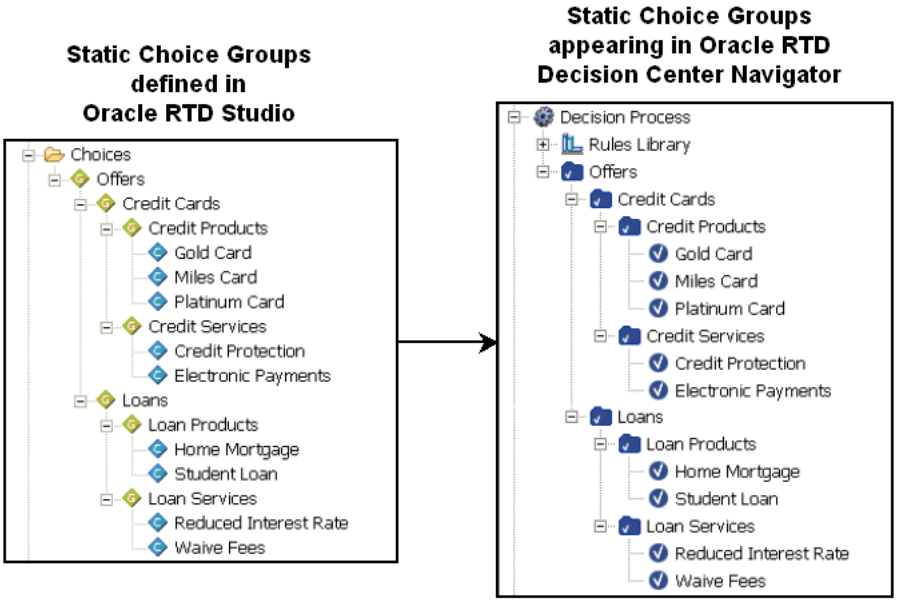
This section consists of the following topics:

- Section 2.6.1.1, "No Impact for Applications with Static Choices Only"
- Section 2.6.1.2, "Dynamic Choice Visibility"
- Section 2.6.1.4, "System-Created Range Folders"
- Section 2.6.1.3, "Dynamic Choice and Range Folder Appearance in the Decision Center"
- Section 2.6.1.5, "Range Folder for Presentation Only"
- Section 2.6.1.6, "Dynamic Choice Reporting"

2.6.1.1 No Impact for Applications with Static Choices Only

If your application has been configured to use only Static Choices, there is no impact on Decision Center reporting. The Choice Groups, subgroups, and Static Choices that you defined in Oracle RTD Decision Studio will appear in the same hierarchical layout in the Oracle RTD Decision Center Navigator, as shown in the example in Figure 2–18.

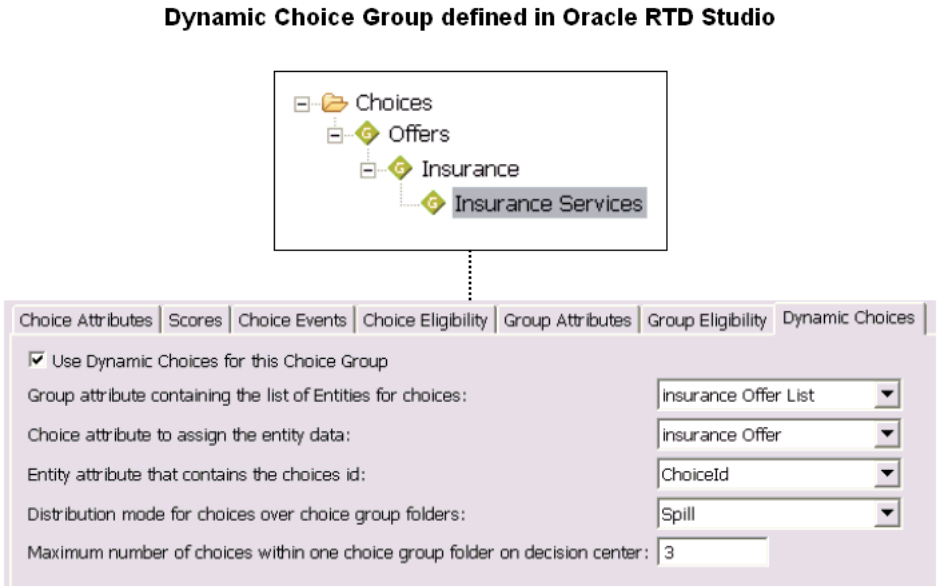
Figure 2–18 Example of Definition and Reporting with Static Choices Only



2.6.1.2 Dynamic Choice Visibility

Dynamic Choices, by their very nature, cannot be predefined in Oracle RTD Decision Studio. A Choice Group can be configured to hold dynamically-extracted external data, from which Dynamic Choices can be recommended. Figure 2–19 shows an example of a Choice Group set up to display Dynamic Choices for insurance services.

Figure 2–19 Example of Dynamic Choice Group Definition



In Oracle RTD Decision Center, only those Dynamic Choices that have actually been recommended and added to model learning data appear in the Decision Center Navigator, and have Performance and Analysis reports.

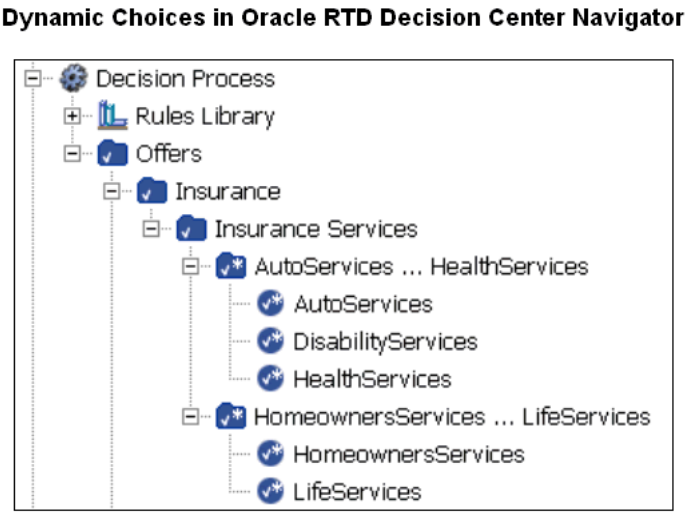
The other factor that influences how the Dynamic Choices appear in the Decision Center is the parameter **Maximum number of choices within one choice group folder on decision center**, which you specify when you define the Dynamic Choice Group. If the number of choices exceeds this maximum, the choices appear under system-created range folders, otherwise they appear directly under the Choice Group name.

For more information on range folders, see Section 2.6.1.4, "System-Created Range Folders."

The example Decision Support Navigator menu in Figure 2–20 shows the following:

- Five Dynamic Choices were recommended and added to model learning data.
- The maximum number of choices per choice group is 3.
- Each Dynamic Choice appears under one of the two system-created folder names.

Figure 2–20 Example of Dynamic Choice Layout in Oracle RTD Decision Center



2.6.1.3 Dynamic Choice and Range Folder Appearance in the Decision Center

Each Dynamic Choice is indicated in the Decision Center Navigator by a circle icon, that also contains a check mark and an asterisk.

Unlike the standard folder, which represents a Choice Group defined in Oracle RTD Decision Studio, the icon representing the range folder contains a check mark and an asterisk.

2.6.1.4 System-Created Range Folders

The name of each system-created folder is made up of the names of the first and last Choices in the folder, with the string "..." separating the two Choices. System-created folders are also known as *range folders*.

Note: The Choices within a range folder can be a mixture of Static and Dynamic Choices. Both components of the range folder name can therefore be either a Static or a Dynamic Choice.

In general, Oracle recommends that applications keep Static and Dynamic Choices in separate Choice Groups or separate Choice Group hierarchies.

If the total number of (Static choices + Dynamic Choices recommended and added to model learning data) exceeds the maximum defined for the Choice Group folder, the choices appear in system-created "groups" or subfolders, otherwise they appear directly under the Choice Group name.

2.6.1.5 Range Folder for Presentation Only

There is an important difference in how you use standard and range folders in the Decision Center.

Whereas you can click the name of a standard folder to open reports for Choices within the folder, range folders are for presentation only.

In other words, clicking the name of a range folder will not open any reports. To see reports for a Dynamic Choice, you must click the Dynamic Choice itself.

2.6.1.6 Dynamic Choice Reporting

When you click a Dynamic Choice in the Navigator, the two tabs that appear are the Performance and Analysis tabs.

Static Choice reporting includes a third option, namely Definition Reports. Because Dynamic Choices are not defined in Oracle RTD Decision Studio, there are no Definition Reports for Dynamic Choices.

The Performance and Analysis reports for Dynamic Choices show exactly the same kind of data and statistics as for Static Choices.

Note: Static Choices always appear in the Decision Center, Dynamic Choices only appear if they were recommended within the time interval of the report.

2.6.2 Distribution of Choices Across Decision Center Folders

When configuring a Choice Group for Dynamic Choices in Oracle RTD Decision Studio, there are two parameters that affect how choices appear in Oracle RTD Decision Center.

Both parameters are in the Dynamic Choices tab, and they are only enabled if the Choice Group is selected to be used for Dynamic Choices. The parameters are:

- **Distribution mode for choices over choice group folders**
- **Maximum number of choices within one choice group folder on decision center**

For simplicity, these parameters are referred to as **Distribution mode** and **Maximum number of choices** in this section.

The **Maximum number of choices** parameter determines how choices appear in the Decision Center directly under the Choice Group name or under a system-created

range folder. For more information on range folders, see Section 2.6.1.4, "System-Created Range Folders."

> **Note:** In Oracle RTD Decision Center reports, range folders are not dedicated to Static or Dynamic Choices, that is, both Static and Dynamic Choices may appear together in the same range folder.

The **Maximum number of choices** parameter limits the number of Choices, regardless of whether they are Static or Dynamic, in each range folder.

The **Distribution mode** parameter specifies how the range folders are populated:

- In **Spill** mode, each range folder is filled up to the maximum, and the final range folder typically has less values than the maximum.
- In **Even** mode, the aim is to distribute the Choices evenly across the range folders.

For example, if there is a total of 106 Static or Dynamic Choices to display in the Decision Center, and the maximum per range folder is 25:

- In **Spill** Mode, the distribution across the range folders is 25,25,25,25,6.
- In **Even** Mode, the distribution across the range folders is 22,21,21,21,21.

3

Model Snapshots

This chapter describes model snapshots, as implemented in Oracle Real-Time Decisions (Oracle RTD), Version 2.2.1. It contains the following topics:

3.1 Model Snapshots Overview

In Oracle RTD Version 2.2.1, you can copy the results of Oracle RTD's learning process to database tables. These results include counts of events, predictiveness values, and correlations. From these tables, you can generate reports using simple SQL commands. This feature is known as Model Snapshots.

Each Inline Service is associated with a Study name, and each Inline Service may have one or more Models. The learning process data applies to each Model in a Study. Using the Model Snapshot feature, you dump all the Model results for a Study to the database.

Note: Strictly speaking, the correct term for the data saved is *study snapshot* rather than *model snapshot*. However, for simplicity, the two terms are used interchangeably in this section.

Overview of Setting Up and Using Model Snapshots

There are four main stages in the process of setting up and using model snapshots, as follows:

1. Configuring the model snapshot tables.
2. Running one or more applications which add Oracle RTD learning data to their associated models.

   This stage does not explicitly use the model snapshot tables, but it supplies the data to be used in the following stage.

3. Populating the model snapshot tables from the learned data, and clearing the tables as required.
4. Creating reports from the model snapshot tables.

Two new system parameters related to model snapshots have been added in Oracle RTD Version 2.2.1. For more information, see Section 3.8, "Tuning the Model Snapshot Process."

3.2 Model Snapshot Tables Schema

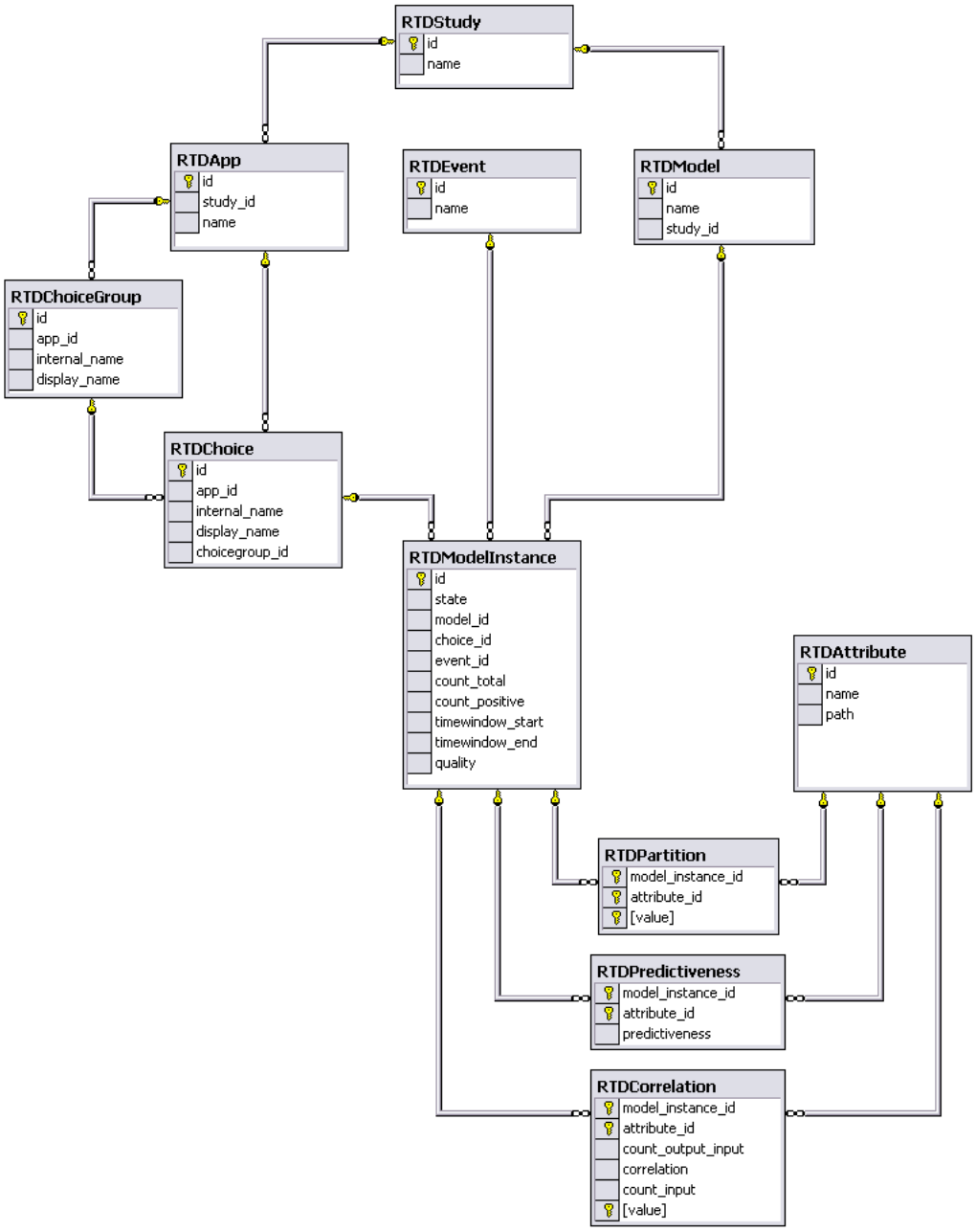
Figure 3–1 shows the model snapshot tables, their columns, and the inter-table relationships, as represented by the connector lines.

Each connector represents a one-to-many relationship, from the table at the top end of the connector to the table at the lower end. In the reverse direction, the relationship from the lower table to the top table is one-to-one.

For example, the relationship from RTDStudy to RTDApp is one-to-many. From RTDApp to RTDStudy, the relationship is one-to-one. This corresponds to the real-world situation that each Study can have one or more Applications, and each Application must have one Study only. For production work, Oracle recommends that you use only one Application per Study.

Note: In this section, the term *Application* is synonymous with the term *Inline Service*. For example, the table RTDApp stores information about Oracle RTD Applications, which are also known as Inline Services.

Figure 3–1 Model Snapshot Tables Schema



The one-to-many relationships between the elements represented by the tables are as follows (each one-to-many relationship implies the corresponding one-to-one relationship in the reverse direction):

- Each Study can have one or more Applications, and one or more Models.
- Each Application can have one or more Choice Groups, and one or more Choices.
- Each Choice Group can have one or more Choices.
- Each Choice, Event, and Model can have one or more Model Instances.

There are also three many-to-many relationships between Model Instances and Attributes, namely Partition, Predictiveness, and Correlation.

The table and column names appear in the following list.

- **RTDApp**

Column	Description
id	Primary key.
study_id	Foreign key to RTDStudy.
name	Application name.

- **RTDAttribute**

Column	Description
id	Primary key.
name	Attribute name.
path	Delimited attribute names showing how this attribute was reached from the root of the session.

- **RTDChoice**

Column	Description
id	Primary key.
app_id	Foreign key to RTDApp.
internal_name	Internal name for Choice.
display_name	Name displayed for Choice.
choicegroup_id	Foreign key to RTDChoiceGroup.

- **RTDChoiceGroup**

Column	Description
id	Primary key.
app_id	Foreign key to RTDApp.
internal_name	Internal name for Choice Group.
display_name	Name displayed for Choice Group.

- **RTDCorrelation**

Column	Description
model_instance_id	Foreign key to RTDModelInstance.
attribute_id	Foreign key to RTDAttribute.
count_output_input	Count of those in the population where the value of this output attribute was found given the input.
correlation	Value between -1 and 1. A positive correlation indicates the degree to which the input attribute's value is associated with the value of the output. A negative correlation indicates a negative association.

Column	Description
count_input	Count of those in the population where this value of this input attribute was found.
value	Value for the input attribute.

- **RTDEvent**

Column	Description
id	Primary key.
name	Event name.

- **RTDModel**

Column	Description
id	Primary key.
name	Model name.
study_id	Foreign key to RTDStudy.

- **RTDModelInstance**

Column	Description
id	Primary key.
state	Model state. Values can be: c: Completed.The model is completed, such as for previous time windows, and is no longer subject to change. It will not be rewritten unless you perform a total dump, or delete the dumps and then perform an incremental dump. d: Combined. The model is for the current and previous time window. s: Split. The model is for the current time window. w: Written. The model is currently being written. Results may be inconsistent.
model_id	Foreign key to RTDModel.
choice_id	Foreign key to RTDChoice.
event_id	Foreign key to RTDEvent.
count_total	Total number of base events.
count_positive	Size of the subset of the population where this non-base event was recorded.
time_window_start	Start of time window.
time_window_end	End of time window.
quality	Quality of the model, value vary from 0 to 1. Higher values are better, 0 means nothing was learned.

- **RTDPartition**

Column	Description
model_instance_id	Foreign key to RTDModelInstance.
attribute_id	Foreign key to RTDAttribute.
value	Attribute value for the partition.

- **RTDPredictiveness**

Column	Description
model_instance_id	Foreign key to RTDModelInstance.
attribute_id	Foreign key to RTDAttribute.
predictiveness	Explanatory score of an input attribute for a particular model instance. Values vary from 0 to 1, higher values are better.

- **RTDStudy**

Column	Description
id	Primary key.
name	Study name.

3.3 Configuring the Model Snapshot Tables

The main objectives of this stage are to create the model snapshot tables, and to register them with the application server and the JMX MBeans.

To configure the model snapshot tables:

1. Select the database where your model snapshot tables will be stored.

   Note: You may choose to store the tables in the SDDB database, but this is not recommended for a production system.

2. From the RTD_HOME\scripts directory, run the command that creates the model snapshot tables:

   ```
   sdexec com.sigmadynamics.tools.SDDBTool.SDDBTool -f -i -I InitSnapshotDb.ctl
   db_type db_host db_port db_name db_runtime_user db_admin_user db_admin_password
   ```

   The following table describes the parameters for the sdexec script.

Parameter	Description
db_type	The database type. Select one of the following: oracle, sqlserver, db2.
db_host	The name of the computer hosting the database server.
db_port	The database port number.
db_name	The name of the database or, for Oracle Database, the SID.
db_runtime_user[1]	The user name of the run-time user for the system.

Parameter	Description
db_admin_user	The name of a user that has rights on the database to create tables and stored procedures.
db_admin_password	The password of the administrative user.

[1] For Oracle Database, the *db_runtime_user* and *db_admin_user* are the same user.

3. Create a new Data Source in your application server, that references the database where your model snapshot tables are stored.

   For details of how to create a Data Source in an application server, see the chapter Configuring Data Access for Oracle Real-Time Decisions in the *Oracle Real-Time Decisions Installation and Administration of Oracle RTD*.

   As part of the operation of creating a new Data Source, you provide a new JNDI name, that is used in the steps following.

4. Add the new Data Source to Oracle RTD.

   The actions required to add the new Data Source to Oracle RTD depend on which application server you are using.

   Perform the steps appropriate to your application server, as follows:

   - If your application server is OC4J, continue at step 5.
   - If your application server is WebSphere, continue at step 11.
   - If your application server is WebLogic, continue at step 14.

   Adding the Data Source to Oracle RTD in OC4J

5. If Oracle RTD is running on standalone OC4J, go to the directory *OC4J_HOME*/j2ee/home/applications/OracleRTD. If Oracle RTD is running on Oracle Application Server, go to *ORACLE_AS_HOME*/j2ee/*oc4j_instance*/applications/OracleRTD.

   This directory is the location where the RTD.ear file was expanded when you deployed it as an application.

6. Locate the file ./ls/WEB-INF/web.xml and open it for editing. Scroll to the bottom of the file. Copy the section for the definition of the resource reference of SDDS and paste it after the existing section. In the copied section, replace the string SDDS with the JNDI name (*jndi_name*) that you entered in step 3.

   For example:

   ```
   <resource-ref id="jndi_name_LS">
   <res-ref-name>jndi_name</res-ref-name>
   <res-type>javax.sql.DataSource</res-type>
   <res-auth>Container</res-auth>
   <res-sharing-scope>Unshareable</res-sharing-scope>
   </resource-ref>
   ```

   Then, save the changes and close the file.

7. Go to the directory *OC4J_HOME*/j2ee/home/application-deployments/OracleRTD. Make sure to go to the application-deployments directory, not the applications directory.

8. Locate the file ./ls/WEB-INF/orion-web.xml and open it for editing. Scroll to the bottom of the file. Copy the line for the definition of the resource reference mapping of SDDS and paste it after the existing line. In the copied line, replace the

string SDDS with the JNDI name (*jndi_name*) that you entered in step 3. For example:

```
<resource-ref-mapping name="jndi_name" location="jndi_name"/>
```

Then, save the changes and close the file.

9. Start OC4J.
10. Continue at step 16.

Adding the Data Source to Oracle RTD in WebSphere

11. Open the following file for editing:

    WEBSPHERE_HOME/AppServer/profiles/*profile_name*/config/cells/*server_name*/applications/OracleRTD.ear/deployments/OracleRTD/ls.war/WEB-INF/web.xml

12. Create a new <resource-ref> entry, as follows:
    a. Copy the existing <resource-ref> entry for the Oracle RTD Database (SDDS) and paste it directly below.
    b. Modify the id attribute by entering a descriptive value followed by _LS, similar to the SDDS entry. The id must have a unique value within the file.
    c. Modify the <res-ref-name> tag by entering the JNDI name (*jndi_name*) that you provided in step 3.

    For example:

```
<resource-ref id="jndi_name_LS">
  <res-ref-name>jndi_name</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Unshareable</res-sharing-scope>
</resource-ref>
```

    Save the changes and close the file.

13. Continue at step 16.

Adding the Data Source to Oracle RTD in WebLogic

14. Go to the directory where you expanded the RTD.ear file during installation (*RTD_HOME*/package/expanded).
15. Open the ls.war archive, extract web.xml, then open web.xml for editing. Scroll to the bottom of the file. Copy the section for the definition of the resource reference of SDDS_LS and paste it after the existing section. In the copied section, replace the string SDDS with the JNDI name (*jndi_name*) that you provided in step 3.

    For example:

```
<resource-ref id="jndi_name_LS">
  <res-ref-name>jndi_name</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Unshareable</res-sharing-scope>
</resource-ref>
```

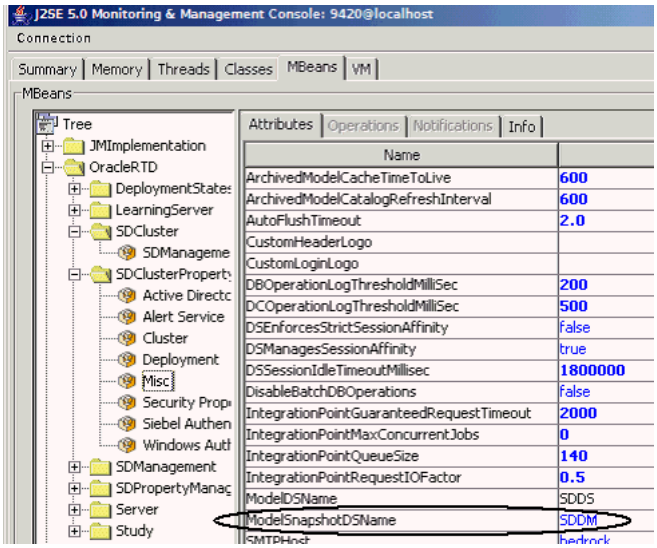
    Save the changes and close the file, then re-archive the file back in ls.war.

    Continue at step 16.

16. In the J2SE Console, register the name of the new Data Source in the OracleRTD MBeans:
    - Navigate to MBeans > OracleRTD > SDClusterPropertyManager > Misc.
    - For the **ModelSnapshotDSName** attribute, specify the name of the Data Source that you created for the model snapshots.

    Figure 3–2 shows the model snapshot Data Source name **SDDM** specified for the **ModelSnapshotDSName** attribute in the J2SE Console.

Figure 3–2 Data Source Name in J2SE Console



3.4 Activating Model Learning

Applications that use Oracle RTD are typically configured to activate model learning, as part of their normal operating procedures.

The process of activating Oracle RTD model learning is simply to execute the calling application. The appropriate application operations then trigger the procedures that send data to Oracle RTD's model learning process. By default, all models learn at session close time.

3.5 Populating and Clearing the Model Snapshot Tables

You can take snapshots of the model learning data at any time, even if not enough data has been accumulated for prediction purposes.

Each model is defined to have a time window, such as week, month, and quarter. This determines how much data to gather in the learning process, and also influences how much data to write to the model snapshot tables.

You can select the amount of data to write from the following options:

- All the model learning data for a Study
- All the model learning data for a Study for the current time window

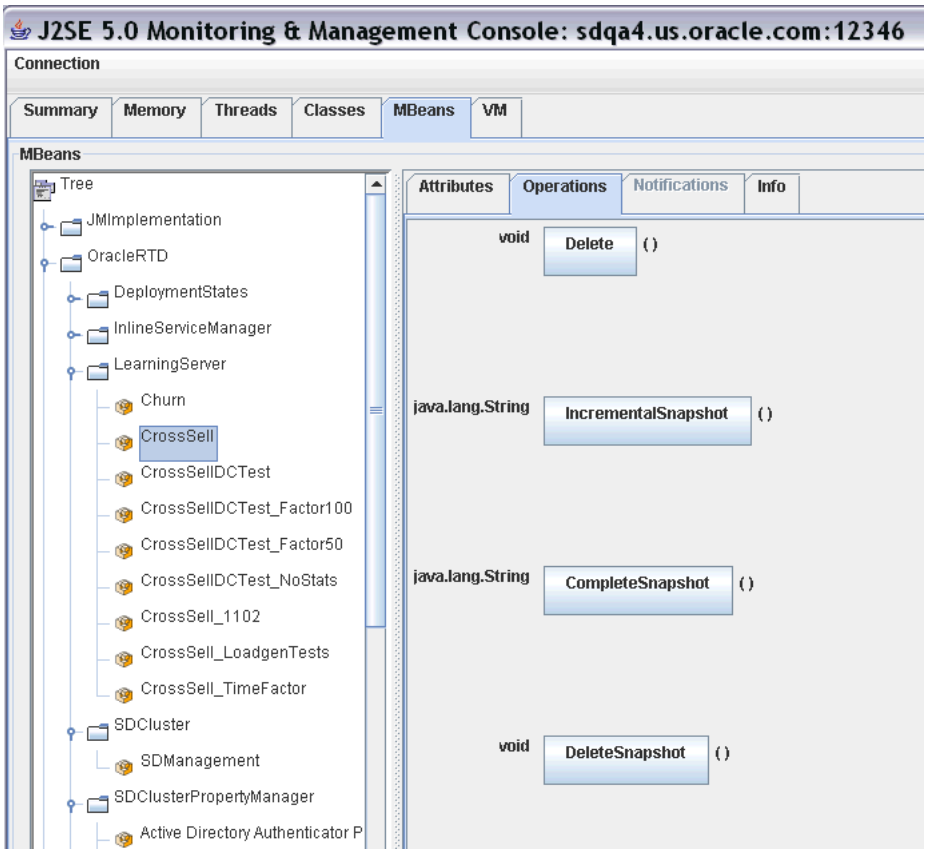
You can also delete all model learning data for a Study from the model snapshot tables.

To populate or clear the model snapshot tables:

1. In the J2SE Console, navigate to MBeans > OracleRTD > Learning Server > *your study name*.
2. Click the Operations tab.

   Figure 3–3 shows the model snapshot operations specified for the **CrossSell** Study in the J2SE Console.

Figure 3–3 MBeans Model Snapshot Operations in J2SE Console



3. Click the appropriate snapshot option:
   - **CompleteSnapshot**

     Deletes all previously snapped data for the Study, and rewrites all Study data, up to the current time.
   - **IncrementalSnapshot**

     Deletes Study data for any incomplete time window, and rewrites the current time window's data, up to the current time.
   - **DeleteSnapshot**

     Deletes all snapped data for the Study.

3.6 Creating Reports from the Model Snapshot Data

You can create reports from the model snapshot tables, typically by using standard SQL Select commands.

This section provides examples of scripts that extract information from several of the model snapshot tables. Each script example is followed by sample output. Notes are provided for some of the examples to help you to interpret the results.

The Inline Service used for these examples was a CrossSell application, and the data was generated by running the Oracle RTD Load Generator script to completion, simulating 400,000 user sessions.

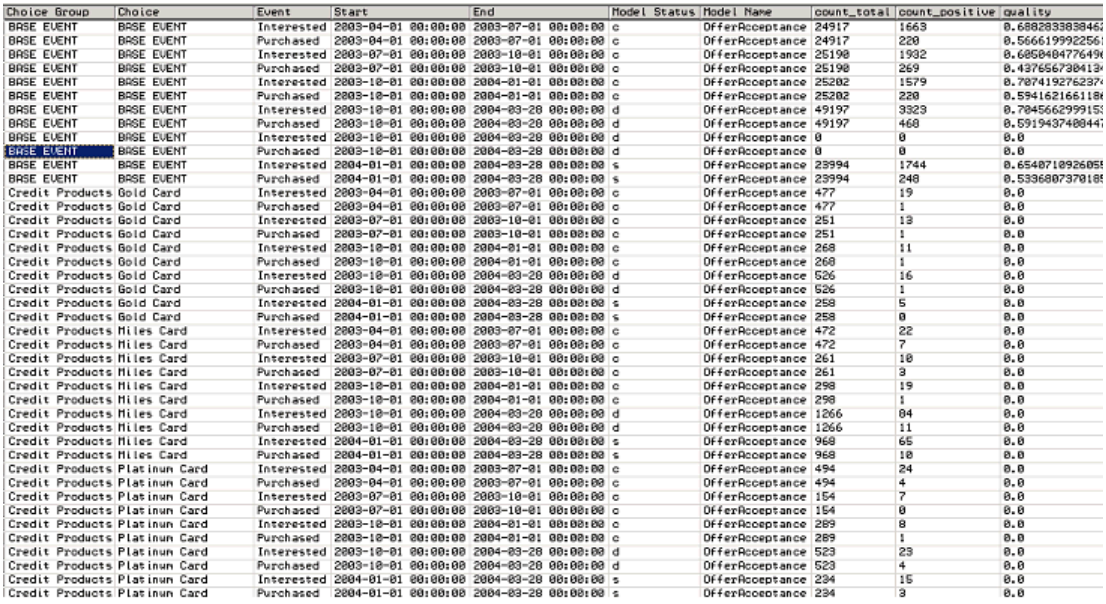
3.6.1 Counts by Choice Query

The following query gets the counts for every Choice, for all time windows:

```
select g.display_name     as 'Choice Group',
       c.display_name     as 'Choice',
       e.name             as 'Event',
      mi.timewindow_start as 'Start',
      mi.timewindow_end   as 'End',
      mi.state            as 'Model Status',
       m.name             as 'Model Name',
      mi.count_total,
      mi.count_positive,
      mi.quality
from RTDApp a
     inner join RTDStudy s          on s.id=a.study_id
     inner join RTDModel m          on m.study_id=s.id
     inner join RTDModelInstance mi on mi.model_id=m.id
     inner join RTDEvent e          on mi.event_id=e.id
     inner join RTDChoice c         on c.id=mi.choice_id
     inner join RTDChoiceGroup g    on c.choicegroup_id=g.id
where a.name='CrossSell'
order by m.name,
         g.display_name,
         c.display_name,
       mi.timewindow_start
```

Figure 3–4 shows the results of the Counts by Choice query.

Figure 3–4 Counts by Choice Query Results



Choice Group	Choice	Event	Start	End	Model Status	Model Name	count_total	count_positive	quality
BASE EVENT	BASE EVENT	Interested	2003-04-01 00:00:00	2003-07-01 00:00:00	c	OfferAcceptance	24917	1663	0.688283383846
BASE EVENT	BASE EVENT	Purchased	2003-04-01 00:00:00	2003-07-01 00:00:00	c	OfferAcceptance	24917	220	0.566619992256
BASE EVENT	BASE EVENT	Interested	2003-07-01 00:00:00	2003-10-01 00:00:00	c	OfferAcceptance	25198	1932	0.605040477649
BASE EVENT	BASE EVENT	Purchased	2003-07-01 00:00:00	2003-10-01 00:00:00	c	OfferAcceptance	25198	269	0.437656738413
BASE EVENT	BASE EVENT	Interested	2003-10-01 00:00:00	2004-01-01 00:00:00	c	OfferAcceptance	25202	1579	0.707419276237
BASE EVENT	BASE EVENT	Purchased	2003-10-01 00:00:00	2004-01-01 00:00:00	c	OfferAcceptance	25202	220	0.594162166118
BASE EVENT	BASE EVENT	Interested	2003-10-01 00:00:00	2004-03-28 00:00:00	d	OfferAcceptance	49197	3323	0.704566299915
BASE EVENT	BASE EVENT	Purchased	2003-10-01 00:00:00	2004-03-28 00:00:00	d	OfferAcceptance	49197	468	0.591943748044
BASE EVENT	BASE EVENT	Interested	2003-10-01 00:00:00	2004-03-28 00:00:00	d	OfferAcceptance	0	0	0.0
BASE EVENT	BASE EVENT	Purchased	2003-10-01 00:00:00	2004-03-28 00:00:00	d	OfferAcceptance	0	0	0.0
BASE EVENT	BASE EVENT	Interested	2004-01-01 00:00:00	2004-03-28 00:00:00	s	OfferAcceptance	23994	1744	0.654071092605
BASE EVENT	BASE EVENT	Purchased	2004-01-01 00:00:00	2004-03-28 00:00:00	s	OfferAcceptance	23994	248	0.533680737018
Credit Products	Gold Card	Interested	2003-04-01 00:00:00	2003-07-01 00:00:00	c	OfferAcceptance	477	19	0.0
Credit Products	Gold Card	Purchased	2003-04-01 00:00:00	2003-07-01 00:00:00	c	OfferAcceptance	477	1	0.0
Credit Products	Gold Card	Interested	2003-07-01 00:00:00	2003-10-01 00:00:00	c	OfferAcceptance	251	13	0.0
Credit Products	Gold Card	Purchased	2003-07-01 00:00:00	2003-10-01 00:00:00	c	OfferAcceptance	251	1	0.0
Credit Products	Gold Card	Interested	2003-10-01 00:00:00	2004-01-01 00:00:00	c	OfferAcceptance	268	11	0.0
Credit Products	Gold Card	Purchased	2003-10-01 00:00:00	2004-01-01 00:00:00	c	OfferAcceptance	268	1	0.0
Credit Products	Gold Card	Interested	2003-10-01 00:00:00	2004-03-28 00:00:00	d	OfferAcceptance	526	16	0.0
Credit Products	Gold Card	Purchased	2003-10-01 00:00:00	2004-03-28 00:00:00	d	OfferAcceptance	526	1	0.0
Credit Products	Gold Card	Interested	2004-01-01 00:00:00	2004-03-28 00:00:00	s	OfferAcceptance	258	5	0.0
Credit Products	Gold Card	Purchased	2004-01-01 00:00:00	2004-03-28 00:00:00	s	OfferAcceptance	258	0	0.0
Credit Products	Miles Card	Interested	2003-04-01 00:00:00	2003-07-01 00:00:00	c	OfferAcceptance	472	22	0.0
Credit Products	Miles Card	Purchased	2003-04-01 00:00:00	2003-07-01 00:00:00	c	OfferAcceptance	472	7	0.0
Credit Products	Miles Card	Interested	2003-07-01 00:00:00	2003-10-01 00:00:00	c	OfferAcceptance	261	10	0.0
Credit Products	Miles Card	Purchased	2003-07-01 00:00:00	2003-10-01 00:00:00	c	OfferAcceptance	261	3	0.0
Credit Products	Miles Card	Interested	2003-10-01 00:00:00	2004-01-01 00:00:00	c	OfferAcceptance	298	19	0.0
Credit Products	Miles Card	Purchased	2003-10-01 00:00:00	2004-01-01 00:00:00	c	OfferAcceptance	298	1	0.0
Credit Products	Miles Card	Interested	2003-10-01 00:00:00	2004-03-28 00:00:00	d	OfferAcceptance	1266	84	0.0
Credit Products	Miles Card	Purchased	2003-10-01 00:00:00	2004-03-28 00:00:00	d	OfferAcceptance	1266	11	0.0
Credit Products	Miles Card	Interested	2004-01-01 00:00:00	2004-03-28 00:00:00	s	OfferAcceptance	968	65	0.0
Credit Products	Miles Card	Purchased	2004-01-01 00:00:00	2004-03-28 00:00:00	s	OfferAcceptance	968	10	0.0
Credit Products	Platinum Card	Interested	2003-04-01 00:00:00	2003-07-01 00:00:00	c	OfferAcceptance	494	24	0.0
Credit Products	Platinum Card	Purchased	2003-04-01 00:00:00	2003-07-01 00:00:00	c	OfferAcceptance	494	4	0.0
Credit Products	Platinum Card	Interested	2003-07-01 00:00:00	2003-10-01 00:00:00	c	OfferAcceptance	154	7	0.0
Credit Products	Platinum Card	Purchased	2003-07-01 00:00:00	2003-10-01 00:00:00	c	OfferAcceptance	154	0	0.0
Credit Products	Platinum Card	Interested	2003-10-01 00:00:00	2004-01-01 00:00:00	c	OfferAcceptance	289	8	0.0
Credit Products	Platinum Card	Purchased	2003-10-01 00:00:00	2004-01-01 00:00:00	c	OfferAcceptance	289	1	0.0
Credit Products	Platinum Card	Interested	2003-10-01 00:00:00	2004-03-28 00:00:00	d	OfferAcceptance	523	23	0.0
Credit Products	Platinum Card	Purchased	2003-10-01 00:00:00	2004-03-28 00:00:00	d	OfferAcceptance	523	4	0.0
Credit Products	Platinum Card	Interested	2004-01-01 00:00:00	2004-03-28 00:00:00	s	OfferAcceptance	234	15	0.0
Credit Products	Platinum Card	Purchased	2004-01-01 00:00:00	2004-03-28 00:00:00	s	OfferAcceptance	234	3	0.0

Notes on the Counts by Choice Query Results

1. In row 13, Choice=Gold Card, Event=Interested, count_total=477, count_positive=19, and quality=0.0. This shows that, out of 477 users that were presented with the Gold Card offer, 19 were interested. The counts are small and the model quality with respect to the Gold Card Choice and the Interested Event is 0.

   In row 14, Choice=Gold Card, Event=Purchased, count_total=477, count_positive=1, and quality=0.0. This shows that one user purchased this offer. The model quality with respect to the Gold Card Choice and the Purchased Event is 0.

   Both row 13 and row 14 apply to the period of time from April 1, 2003 to July 1, 2003.

2. In the columns Choice Group and Choice, the value BASE EVENT means "in general" or "overall."

   For example, in row 1, Choice=BASE EVENT, Event=Interested, count_total=24917, count_positive=1663, and quality is approximately 0.6882. This means that, in the period between the Start and End dates, a grand total of 24917 users were presented offers, and 1663 of these were interested. The overall model quality for this period of time was about 0.69.

   In row 2, Choice=BASE EVENT, Event=Purchased, count_total=24917, count_positive=220, and quality is approximately 0.5666. This means that, for the same period of time as for row 1, there were 220 Purchased events across all Choices, and the model quality was about 0.57.

3.6.2 Top Six Predictive Attributes Query

The following query selects the top six predictive attributes, for each time window, for the Credit Protection Choice resulting in the Purchased Event.

```
select a.name                'Attribute Name',
       p.predictiveness      'Predictiveness',
       c.display_name        'Choice Name',
      mi.timewindow_start as 'Start',
      mi.timewindow_end   as 'End',
      mi.state            as 'Model Status'
from RTDApp app
     inner join RTDChoice c         on c.app_id=app.id
     inner join RTDStudy s          on s.id=app.study_id
     inner join RTDModel m          on m.study_id=s.id
     inner join RTDModelInstance mi on mi.model_id=m.id and mi.choice_id=c.id
     inner join RTDEvent e          on mi.event_id=e.id
     inner join RTDPredictiveness p on p.model_instance_id=mi.id
     inner join RTDAttribute a      on a.id=p.attribute_id
where app.name         = 'CrossSell'
  and   c.display_name = 'Credit Protection'
  and   e.name         = 'Purchased'
  and   m.name         = 'OfferAcceptance'
  and   7 > (select count(*)
             from RTDPredictiveness p2
             where p2.model_instance_id = p.model_instance_id
               and p2.predictiveness > p.predictiveness)
order by mi.timewindow_end desc,
         p.predictiveness desc
```

Figure 3–5 shows the results of the Top Six Predictive Attributes query.

Figure 3–5 Top Six Predictive Attributes Query Results

	Attribute Name	Predictiveness	Choice Name	Start	End
1	customer CreditLineAmount	1.3216953724622726E-2	Credit Protection	2003-07-01 00:00:00	2003-10-01 00:0
2	customer MaritalStatus	0.01082124374806881	Credit Protection	2003-07-01 00:00:00	2003-10-01 00:0
3	customer AvailableCreditAsPercentOfCreditLine	6.3693146221339703E-3	Credit Protection	2003-07-01 00:00:00	2003-10-01 00:0
4	customer Age	6.1343498528003693E-3	Credit Protection	2003-07-01 00:00:00	2003-10-01 00:0
5	customer Amount Of Pending Transactions	3.1012502165002823E-3	Credit Protection	2003-07-01 00:00:00	2003-10-01 00:0
6	customer MinimumAmountDue	1.8700361251831055E-3	Credit Protection	2003-07-01 00:00:00	2003-10-01 00:0
7	customer CreditLineAmount	1.3111146166920662E-2	Credit Protection	2003-04-01 00:00:00	2003-07-01 00:0
8	customer AvailableCreditAsPercentOfCreditLine	1.0825838893651962E-2	Credit Protection	2003-04-01 00:00:00	2003-07-01 00:0
9	customer Age	6.3290330581367016E-3	Credit Protection	2003-04-01 00:00:00	2003-07-01 00:0
10	customer MaritalStatus	5.4984893649816513E-3	Credit Protection	2003-04-01 00:00:00	2003-07-01 00:0
11	customer Amount Of Pending Transactions	2.6986880693584681E-3	Credit Protection	2003-04-01 00:00:00	2003-07-01 00:0
12	customer Occupation	2.452038461342454E-3	Credit Protection	2003-04-01 00:00:00	2003-07-01 00:0

3.6.3 Difference Between Expected and Actual Counts Query

The following query shows, for people of different marital statuses, the number who actually purchased credit protection and the number who were expected to do so. The report also shows the difference between the two values, and the importance of the correlation for the offer acceptance.

```
select cor.value               'customer MaritalStatus',
       cor.count_output_input  'Actual Count',
       mi.count_positive*cor.count_input/mi.count_total 'Expected Count',
 (100*(mi.count_positive*cor.count_input/mi.count_total
     - cor.count_output_input))/cor.count_output_input  'Percent Difference',
       cor.correlation         'Importance',
         c.display_name        'Choice Name',
       mi.timewindow_start as 'Start',
       mi.timewindow_end   as 'End',
       mi.state            as 'Model Status'
from RTDApp app
     inner join RTDChoice c         on c.app_id=app.id
     inner join RTDStudy s          on s.id=app.study_id
```

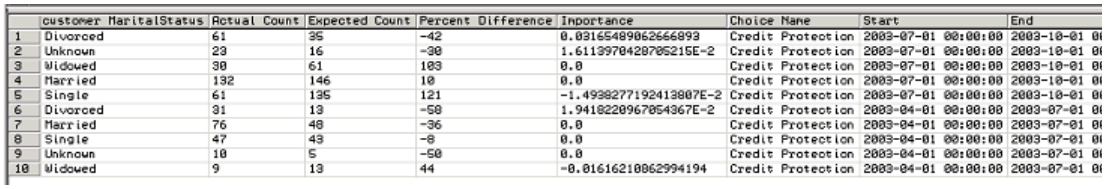
```
     inner join RTDModel m          on m.study_id=s.id
     inner join RTDModelInstance mi on mi.model_id=m.id and mi.choice_id=c.id
     inner join RTDEvent e          on mi.event_id=e.id
     inner join RTDCorrelation cor  on cor.model_instance_id=mi.id
     inner join RTDAttribute a      on a.id = cor.attribute_id
where app.name         = 'CrossSell'
    and c.display_name = 'Credit Protection'
    and e.name         = 'Purchased'
    and m.name         = 'OfferAcceptance'
    and a.name         = 'customer MaritalStatus'
order by mi.timewindow_end desc,
        cor.correlation desc
```

Notes on the Difference Between Expected and Actual Counts Query

1. The Actual Count, `cor.count_output_input`, is the actual number of people who purchased credit protection, for each marital status.
2. The Expected Count is a simple linear projection of the total count of each marital status, `cor.count_input`, to those that purchased credit protection, as expressed by `mi.count_positive/mi.count_total`.
3. The Percent Difference is 100 * (Expected Count - Actual Count) / Actual Count.

Figure 3–6 shows the results of the Difference Between Expected and Actual Counts query.

Figure 3–6 Difference Between Expected and Actual Counts Query Results



	customer MaritalStatus	Actual Count	Expected Count	Percent Difference	Importance	Choice Name	Start	End
1	Divorced	61	35	-42	0.031655489062566893	Credit Protection	2003-07-01 00:00:00	2003-10-01 00
2	Unknown	23	16	-30	1.6113970428705215E-2	Credit Protection	2003-07-01 00:00:00	2003-10-01 00
3	Widowed	30	61	103	0.0	Credit Protection	2003-07-01 00:00:00	2003-10-01 00
4	Married	132	146	10	0.0	Credit Protection	2003-07-01 00:00:00	2003-10-01 00
5	Single	61	135	121	-1.4938277192413807E-2	Credit Protection	2003-07-01 00:00:00	2003-10-01 00
6	Divorced	31	13	-58	1.9418220967054367E-2	Credit Protection	2003-04-01 00:00:00	2003-07-01 00
7	Married	76	48	-36	0.0	Credit Protection	2003-04-01 00:00:00	2003-07-01 00
8	Single	47	43	-8	0.0	Credit Protection	2003-04-01 00:00:00	2003-07-01 00
9	Unknown	10	5	-50	0.0	Credit Protection	2003-04-01 00:00:00	2003-07-01 00
10	Widowed	9	13	44	-0.01616210862994194	Credit Protection	2003-04-01 00:00:00	2003-07-01 00

Notes on the Difference Between Expected and Actual Counts Query Results

1. There are two rows for each marital status, each corresponding to one of the two time periods, April 1, 2003 - July 1, 2003 and July 1, 2003 - October 1, 2003.

3.7 Handling Partitions

The RTDPartition table holds values for partitioning attributes. If a Model is not partitioned, the Model has one Model Instance per time window, and there are no associated rows in the RTDPartition table.

A Model that is split along one or more of its dimensions is a partitioned Model.

As an example, a Model M can be partitioned by two attributes, Marital Status and Favorite Beverage. If there are 3 values for Marital Status (Married, Single, Divorced) and 2 for Favorite Beverage (coffee, tea), then this model has 6 model instances.

In this case, each Model Instance has two associated RTDPartition rows. For example, the Model Instance for the combination (Marital Status=Married and Favorite Beverage=Coffee) would be associated with two RTDPartition rows, containing the following information:

- RTDPartition row 1: Attribute=Marital Status, Value=Married

- RTDPartition row 2: Attribute=Favorite Beverage, Value=Coffee

Whether or not a Model is partitioned can influence the results of queries on the model snapshot tables. To avoid repetitions in your results, include RTDPartition and RTDAttribute join conditions in your query.

The following example modifies and extends the Difference Between Expected and Actual Counts query to cover the case of a Model partitioned on two attributes, Diabetic, which has "yes" and "no" values, and Marital Status.

```
select a.name,
       p.value,
      subquery.*
from (select cor.value               'Favorite Sports',
             cor.count_output_input  'Actual Count',
             mi.count_positive*cor.count_input/mi.count_total 'Expected Count',
 (100*(mi.count_positive*cor.count_input/mi.count_total
     - cor.count_output_input))/cor.count_output_input  'Percent Difference',
       cor.correlation         'Importance',
         c.display_name        'Choice Name',
        mi.timewindow_start as 'Start',
        mi.timewindow_end   as 'End',
        mi.state            as 'Model Status'
        mi.id                   model_instance_id
  from RTDApp app
       inner join RTDChoice c         on c.app_id=app.id
       inner join RTDStudy s          on s.id=app.study_id
       inner join RTDModel m          on m.study_id=s.id
       inner join RTDModelInstance mi on mi.model_id=m.id and mi.choice_id=c.id
       inner join RTDEvent e          on mi.event_id=e.id
       inner join RTDCorrelation cor  on cor.model_instance_id=mi.id
       inner join RTDAttribute a      on a.id = cor.attribute_id
  where app.name       = 'HighlyPartitionedDataset'
    and c.display_name = 'Fanta'
    and e.name = 'loved'
    and m.name = 'SatisfactionModel'
    and a.name = 'Favorite Sports') as subquery
inner join RTDPartition p on subquery.model_instance_id = p.model_instance_id
inner join RTDAttribute a on p.attribute_id = a.id
order by subquery.[End] desc,
         subquery.model_instance_id,
         a.name,
         p.value
         subquery.[Importance] desc
```

Figure 3–7 shows the results of the Partitioned Expected and Actual Counts query.

Figure 3–7 Partitioned Expected and Actual Counts Query Results

	name	value	Favorite Sports	Actual Count	Expected Count	Percent Differenct	Importance	Choice Name	Star
1	Diabetic	no	baseball	89	92	3	0.0	Fanta	2007
2	Diabetic	no	basketball	87	91	4	0.0	Fanta	2007
3	Diabetic	no	football	93	92	-1	0.0	Fanta	2007
4	Diabetic	no	golf	88	90	2	0.0	Fanta	2007
5	Marital Status	married	baseball	89	92	3	0.0	Fanta	2007
6	Marital Status	married	basketball	87	91	4	0.0	Fanta	2007
7	Marital Status	married	football	93	92	-1	0.0	Fanta	2007
8	Marital Status	married	golf	88	90	2	0.0	Fanta	2007
9	Diabetic	yes	baseball	71	73	2	0.0	Fanta	2007
10	Diabetic	yes	basketball	83	83	0	0.0	Fanta	2007
11	Diabetic	yes	football	71	77	8	0.0	Fanta	2007
12	Diabetic	yes	golf	64	70	9	0.0	Fanta	2007
13	Marital Status	married	baseball	71	73	2	0.0	Fanta	2007
14	Marital Status	married	basketball	83	83	0	0.0	Fanta	2007
15	Marital Status	married	football	71	77	8	0.0	Fanta	2007
16	Marital Status	married	golf	64	70	9	0.0	Fanta	2007

3.8 Tuning the Model Snapshot Process

Two new system properties are available in Oracle RTD Version 2.2.1 for tuning the model snapshot process:

- **ModelSnapshotMinAbsCorrelation** controls whether to dump all correlation rows or to set a minimum correlation value for dumping. The default value of 0.000001 prevents the dumping of very small value correlation rows. Set the value to 0 to dump all correlation rows.
- **ModelSnapshotNumberOfBins** controls the number of bins for model snapshots. Numeric attribute values are automatically binned, or assigned to numeric ranges. The default number of bins is 5. To achieve greater resolution of your numeric data, increase the number of bins.

> **Caution:** For the same numeric attribute, Oracle RTD creates different bins in different time windows. Therefore, it is unlikely that you will be able to join numeric attribute values across time windows.

If you want to change the value of a system property, set the system properties as server properties for your application server. See *Oracle Real-Time Decisions Installation and Administration of Oracle RTD* for details of how to configure server properties for your application server.

For WebSphere and OC4J, edit the JVM arguments. For WebLogic, edit the Java options. For example, add the parameter `-DModelSnapshotNumberOfBins=10` to the JVM arguments for WebSphere and OC4J, or to the Java options for WebLogic.