Oracle Endeca Commerce

Concepts Guide

Version 11.0 • January 2014

ORACLE

COMMERCE

Contents

Copyright and disclaimer

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Preface

Oracle Endeca Commerce is the most effective way for your customers to dynamically explore your storefront and find relevant and desired items quickly. An industry-leading faceted search and Guided Navigation solution, Oracle Endeca Commerce enables businesses to help guide and influence customers in each step of their search experience. At the core of Oracle Endeca Commerce is the MDEX Engine™, a hybrid search-analytical database specifically designed for high-performance exploration and discovery. The Endeca Content Acquisition System provides a set of extensible mechanisms to bring both structured data and unstructured content into the MDEX Engine from a variety of source systems. Endeca Assembler dynamically assembles content from any resource and seamlessly combines it into results that can be rendered for display.

Oracle Endeca Experience Manager is a single, flexible solution that enables you to create, deliver, and manage content-rich, cross-channel customer experiences. It also enables non-technical business users to deliver targeted, user-centric online experiences in a scalable way — creating always-relevant customer interactions that increase conversion rates and accelerate cross-channel sales. Non-technical users can determine the conditions for displaying content in response to any search, category selection, or facet refinement.

About this guide

This guide introduces Oracle Endeca Commerce and provides a walkthrough of the key concepts of Endeca applications including basic data structures, query syntax, and comparisons of difference search types.

Who should use this guide

This guide is intended for developers who are new to Oracle Endeca Commerce, as well as individuals who want to understand the core concepts underlying an Endeca application.

Conventions used in this guide

This guide uses the following typographical conventions:

Code examples, inline references to code elements, file names, and user input are set in `monospace` font. In the case of long lines of code, or when inline monospace text occurs at the end of a line, the following symbol is used to show that the content continues on to the next line: ¬

When copying and pasting such examples, ensure that any occurrences of the symbol and the corresponding line break are deleted and any remaining space is closed up.

Contacting Oracle Support

Oracle Support provides registered users with important information regarding Oracle Endeca software, implementation questions, product and solution help, as well as overall news and updates.

You can contact Oracle Support through Oracle's Support portal, My Oracle Support at *https://support.oracle.com*.

Chapter 1

Welcome to Oracle Endeca Commerce

This section introduces Oracle Endeca Commerce.

Introduction

Endeca was founded to provide people with a way to explore data interactively in real time, through an interface that remains easy to understand and use regardless of the scale and complexity of the underlying data – which is often stored in numerous large data sources.

People need to search, navigate, and analyze all of their data, sometimes in fine detail, and at other times in broad aggregate views. They need an application that responds intelligently to their current navigation state, guiding them along valid paths and eliminating invalid paths that do not lead to data. They should experience simple, intuitive navigation even as they perform the equivalent of extremely complex, multi-dimensional database intersection queries.

Oracle Endeca Commerce, based on the Endeca MDEX Engine, is a powerful technology designed to help you build Guided Navigation applications that fulfill all of these requirements. Guided Navigation not only tells users the results of their query, it also tells them all the valid "next-step questions" that they can ask to refine and explore further, while eliminating the frustrating reply of "No Results Found". These next-steps are re-ranked and re-organized with each click, creating a much more productive and satisfying navigation experience.

Market solutions

The Oracle Endeca product line offers a series of market solutions built on top of Oracle Endeca Commerce.

These solutions, such as Oracle Endeca Experience Manager, Oracle Endeca Commerce Business Intelligence, and Oracle Endeca Workbench, are packaged to meet specific information access business needs.

As each solution incorporates a different set of modules, not all of the features described in the documentation may be available as part of your Oracle Endeca license. However, because these solutions are based on the same platform, much of the functionality is identical across solutions.

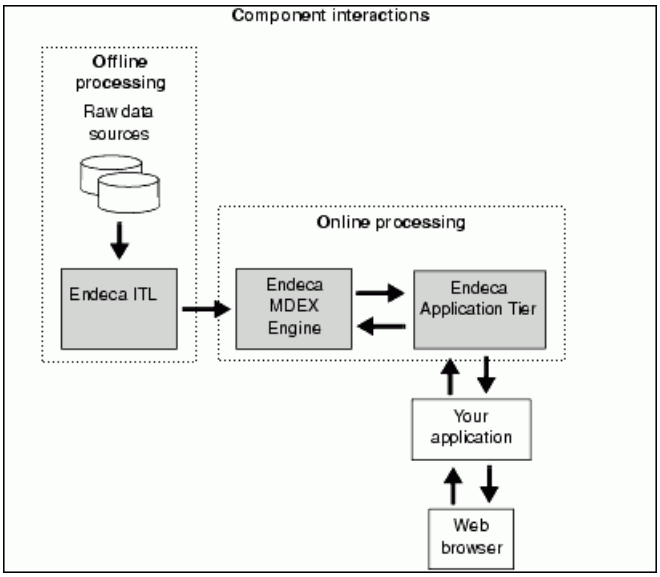
Oracle Endeca Commerce components

Oracle Endeca Commerce comprises three major components.

These components are:

- Endeca Information Transformation Layer (ITL)
- Endeca MDEX Engine
- Endeca Application Tier

The components interact with your data sources and application as shown in the following figure:



The Endeca Information Transformation Layer (ITL) reads your raw source data and manipulates it into a set of Oracle Endeca MDEX Engine indexes. The ITL consists of the Content Acquisition System (which includes the Endeca CAS Server and Console, the CAS API and the Endeca Web Crawler), and the Data Foundry (which includes data-manipulation programs such as Forge).

The Oracle Endeca MDEX Engine is the query engine that is the core of Oracle Endeca Commerce. The MDEX Engine consists of the Indexer (Dgidx), and the Dgraph. The MDEX Engine loads the indexes generated by the indexing component of the Endeca Information Transformation Layer. Although the Indexer (also known as Dgidx) is installed as part of the MDEX Engine package, in effect it is part of the ITL process.

After the indexes are loaded, the MDEX Engine receives queries from the Endeca Application Tier, executes them against the loaded indexes, and returns the results to the client application. The Application Tier provides an interface to the MDEX Engine via the Endeca Assembler. The Assembler aggregates queries, sends them to the MDEX Engine, and performs any necessary post-processing on the results.

Note: The Endeca ITL components are run offline at intervals that are appropriate for your business requirements. The Endeca MDEX Engine and Endeca Application Tier are both online processes – , that is, they must remain running as long as you want clients to have access to your data set.

Installation packages for Endeca components

The Endeca components are installed in several packages.

This section provides general information about the installation packages. For detailed information on all installation packages for Oracle Endeca Commerce and their related documentation, see the *Oracle Endeca Commerce Getting Started Guide*.

MDEX Engine

The MDEX Engine core installation package installs the Endeca MDEX Engine. The server on which the MDEX Engine will run requires the EAC Agent from the Platform Services installation package. For detailed information on installation requirements, see the *Oracle Endeca MDEX Engine Installation Guide*.

Oracle Endeca Tools and Frameworks

The Oracle Endeca Tools and Frameworks enable the development of Web applications that present dynamic and contextual content to end customers across multiple channels. The main components of Tools and Frameworks are:

- Oracle Endeca Workbench with Experience Manager, a Web-based tool that enables business users to create and manage dynamic content configurations across multiple channels or applications.
- The Endeca Assembler, a system for collecting and sending queries to the Endeca MDEX Engine and returning the resulting data to an application as ready-to-render content items.
- The Discover Electronics store, a reference application that demonstrates best practices for Oracle Endeca Commerce applications.

For detailed information on installation requirements, see the *Tools and Frameworks Installation Guide*.

Platform Services

The Platform Services core installation package includes Forge, the Oracle Endeca Application Controller (EAC), and the Endeca Logging and Reporting System. For detailed information, see the *Oracle Endeca Platform Services Installation Guide*.

Endeca MDEX Engine query results

All query results returned from the Endeca MDEX Engine contain two types of information.

These information types are:

- The appropriate results for the query (for example, a record set or an individual record)
- The supporting information for building follow-on queries

The follow-on query information allows users to refine or broaden their query and, correspondingly, their query results. The method the MDEX Engine uses to compute this information eliminates invalid follow-on queries (*dead ends*). Eliminating dead ends and providing relevant next-step refinement choices are two of the primary features that distinguish Endeca solutions from other types of search implementations.

Two types of queries

Oracle Endeca Commerce uses two types of queries: navigation queries and keyword search queries.

- *Navigation queries* return a set of records based on application-defined record characteristics (such as wine type or region in an online wine store), plus any follow-on query information.
- *Keyword search queries* return a set of records or dimensions based on a user-defined keyword, plus any follow-on query information. For more information, see "Using Keyword Search."

Navigation queries and keyword search queries are complementary. In fact, a keyword search query is a specialized form of navigation query, and the data structures for the results of the two queries are identical: a set of records and follow-on query information.

Users can execute a combination of navigation queries and keyword search queries to navigate to their desired record set in the way that works best for them. For example, users can execute a keyword search query to retrieve a set of records, then use a follow-on navigation query to refine that set of records. The reverse situation is also valid.

Related Links

Using Keyword Search on page 31
This section describes the two types of keyword search queries: record search and dimension search.

Logical versus physical structure

The data in an Endeca application has both a physical structure and a logical structure that support your MDEX Engine queries.

To understand the difference between physical structure and logical structure, consider a relational database. A relational database has a set of tables, each of which contains its own data. Relationships exist among the tables that allow you to create logical records from data spread across multiple tables. The database's physical structure is a set of individual tables, and its logical structure is a set of records whose data is drawn from those tables.

An Endeca implementation also imposes both a physical structure and a logical structure on your data. "Understanding Records, Dimensions, and Properties" describes these structures and how the MDEX Engine uses them to respond to queries.

Reference implementations

In addition to the three core Oracle Endeca Commerce components, your Endeca distribution also contains reference implementations that implement many of Endeca's features.

You can use these optional reference implementations as a starting point or guide when building your own application. The reference implementations incorporate two types of information:

- *Data reference implementations* show you how to use the Information Transformation Layer features to convert your source data into MDEX Engine indexes.
- *UI reference implementations* show you how to add Endeca features to your application's user interface.

The reference implementations are located in the `reference` directories of your Platform Services and Tools and Frameworks installations. For more information about the reference implementations, see the *Oracle Endeca Commerce Getting Started Guide* and the *Endeca MDEX Engine Development Guide*.

Chapter 2

Understanding Records, Dimensions, and Properties

This section describes Endeca records, dimensions, and properties.

About Endeca records, dimensions, and properties

Endeca records, dimensions, and properties store and organize product information, making it accessbile to customers through your Endeca applications.

Endeca records

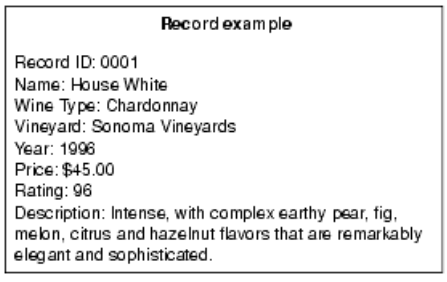
Endeca records are the elements of your data set that users navigate to or search for.

Endeca records are based on traditional records in a source database. Source database records typically contain information such as the bottles of wine in a wine store, the customer records in a CRM application, or the mutual funds in a fund evaluator.

Source database records store this information in one or more key/value pairs, known as properties. This information becomes available to your Endeca application when you transform the source database records into Endeca records. To transform the source database records into Endeca records, you must map the source record properties to properties or dimensions within Endeca records.

Thus, Endeca records are collections of Endeca properties and dimensions that correspond to the properties of source database records. Like source record properties, Endeca properties and dimensions are key/value pairs.

The following figure illustrates key/value pairs in a simple Endeca record:



A single Endeca record can correspond to any number of source records. For example, suppose that four different source records refer to the same book in different formats: hardcover, paperback, large print, and

audio. You can configure your Endeca application to combine the information in these four source records into a single Endeca record.

Dimensions and dimension values

Dimensions are logical categories that make it possible to organize your Endeca records into a hierarchical structure that customers can search for information.

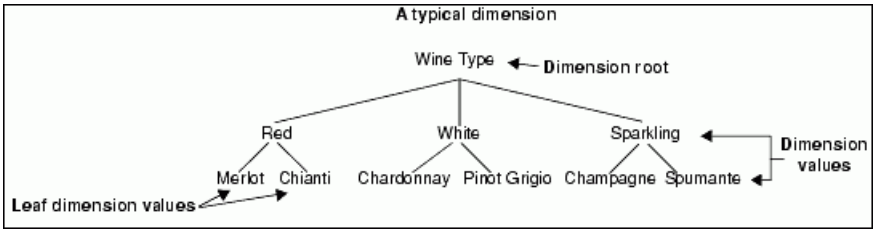
Records can be organized into searchable hierarchies by tagging them with dimension values that are themselves organized into hierarchies.

A hierarchy of dimension values is known as a dimension. The top-most dimension value in a dimension is known as the *dimension root*. A dimension root is always the same as the name of the dimension.

Each dimension value can have one or more *child* dimension values; a dimension value with child dimension values is known as a *parent* dimension value. A child dimension value can have only one parent dimension value. Dimension values that are children of the same parent dimension value are known as *sibling* dimension values.

Sibling dimension values cannot be identical. However, dimension values that are not siblings can be identical, even within the same dimension.

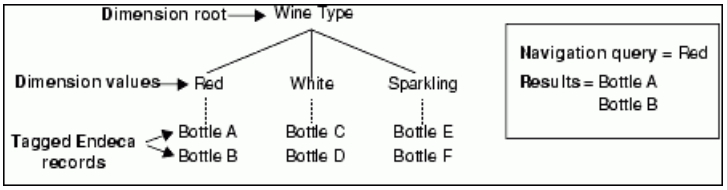
The bottom-most dimension values in a dimension are known as *leaf* dimension values. For most purposes, it is best to tag Endeca records only with leaf dimension values.



Tagging a record with a dimension value does the following things:

- It specifies the location of the record within the associated dimension. In the example below, the Endeca records for Bottles A and B are tagged with the Red dimension value in the Wine Type dimension, while the Endeca records for Bottles C and D are tagged with the White dimension value, and so on.
- It identifies the record as a valid result when that dimension value is selected in a navigation query.

In the example below, a navigation query on the Red dimension value produces a result set that contains Bottles A and B.



You can think of a navigation query as "return all the records that are organized under the xxx dimension value in the yyy dimension." Or, you can think of it as "return all the records that have been tagged with the xxx dimension value in the yyy dimension." In either case, the result is the same.

A record can be tagged with any number of dimension values from any number of dimensions. This enables users to navigate to records through any dimension or combination of dimensions that they choose. Using multiple dimensions is described in more detail in the "Multiple dimensions" topic.

Related Links

Multiple dimensions on page 17
Each record in your Endeca data set can be tagged with zero, one, or more dimension values within each dimension.

Transforming source records to Endeca records

Before you transform a source record into an Endeca record, you must do one of three things to each of the source record's properties:

- Map the source property to an Endeca dimension.

  Mapping a source property to a dimension tags the Endeca record with a dimension value from that dimension. As described in the section above, dimensions and dimension values are the basis of Endeca's navigation functionality.

- Map the source property to an Endeca property.

  The values of Endeca properties are descriptive information that can be displayed with a record when a user has accessed the record though a search or by navigation. Note that Endeca properties *cannot be used to classify records or to navigate to them*.

- Decide to ignore the source property (for using it in navigation).

  If a source property does not contain information suitable for navigation, search, or display, you can instruct your Endeca implementation to ignore it during the data transformation process.

When all the properties of a source record have been mapped, the source record can be transformed into an Endeca record consisting of Endeca properties and Endeca dimensions.

Comparing Endeca properties and dimensions

Both Endeca properties and dimensions:

- Are usually generated from a source record's properties, to which they have been mapped.
- Consist of key/value pairs (property name/property value, dimension name/dimension value).
- Can be searched for and displayed.

Although Endeca properties and Endeca dimensions share many characteristics, only dimensions can be used for guided navigation. The MDEX engine generates indexes based on your mappings, and only indexes for dimensions can be used for navigation.

Thus, when you map a source property to an Endeca property or dimension, consider whether the value of that property is suitable for use as a navigation term. If it is suitable, map it to an Endeca dimension. If it is not suitable, map it to an Endeca property. For example, a source property whose value is "Wine Type" is useful for navigation because it enables you to identify a set of wines according to type: red, white, sparkling, and so forth. Source properties that contain this type of information should be mapped to Endeca dimensions.

On the other hand, a property whose value is a long description of a wine is not suitable for use as a navigation term, and for this reason should be mapped to an Endeca property rather than to an Endeca dimension. For example, end users do not query for wines that are "*intense, with complex earthy pear, fig, melon, citrus and hazelnut flavors that are remarkably elegant and sophisticated.*" This type of information is useful to display when the user has navigated to the record, and source properties that contain this type of information should be mapped to Endeca properties.

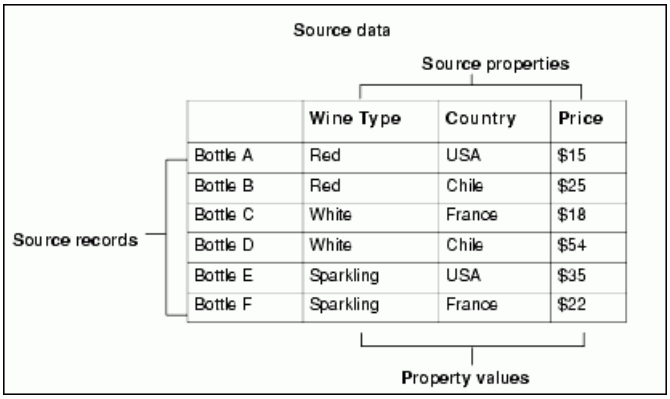
Endeca properties often contain more specific information about a record than dimensions. For example, a Price Range dimension is useful for navigation—give me all the bottles of wine that cost between $10 and $20 dollars—but it is the exact price of each bottle that you want to see when looking at the individual records. A common implementation for this type of application uses a Price Range *dimension* for navigation and a Price *property* that is displayed when a bottle's record has been located.

Note: Property, dimension, and dimension value names are case sensitive.

Automatically deriving dimensions from source data

You can choose to generate dimensions in your Endeca records automatically, basing the dimensions on properties in your source database records.

The following two-dimensional table gives an example of what your source data might look like:



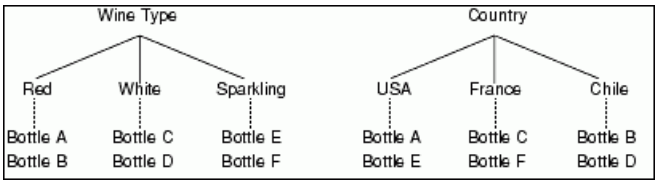
	Wine Type	Country	Price
Bottle A	Red	USA	$15
Bottle B	Red	Chile	$25
Bottle C	White	France	$18
Bottle D	White	Chile	$54
Bottle E	Sparkling	USA	$35
Bottle F	Sparkling	France	$22

Note: While it is most convenient to use a tabular format for your source data, other formats are also acceptable. For details, see the *Endeca Platform Services Forge Guide*.

In this source data table, the following relationships exist:

- Each row corresponds to a source record. Each source record can be transformed into an Endeca record.
- Each column represents a property. Each property can be mapped to a dimension, an Endeca property, or both.
- The column entries represent the property values that are assigned to each record for the given property. These values map to either dimension values (if the source property is mapped to a dimension), or Endeca property values (if the source property is mapped to an Endeca property).

With automatic dimension generation, source properties become dimensions and source property values become dimension values. If we mapped the Wine Type and Country source properties to dimensions, and then automatically generated those dimensions, the logical representation of the generated dimensions would look like this:



The Wine Type and Country source properties become the Wine Type and Country dimensions. Each source property value becomes a dimension value within its respective dimension: Red, White, and Sparkling for the

Wine Type dimension and USA, France, and Chile for the Country dimension. The individual records are organized beneath their respective dimension values accordingly.

With automatically-generated dimensions, it is important to remember that changes to your source data's properties and property values directly influence those dimensions and their dimension values.

Note: The Endeca Information Transformation Layer (ITL) enables you to map source record properties to Endeca dimensions and properties in any way that you require. For example, you can map only some of your source properties to Endeca properties and dimensions, and you can create Endeca properties and dimensions that do not exist in your source data. You can also configure ITL to resolve property name conflicts – for example, you can configure ITL to create synonyms for dimension values that would be identical to their siblings, which is not allowed.

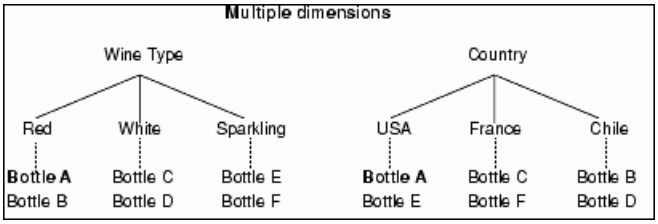
More about dimensions

This section provides additional information about dimensions and dimension values.

Multiple dimensions

Each record in your Endeca data set can be tagged with zero, one, or more dimension values within each dimension.

In the illustration below, Bottle A is tagged with the Red dimension value in the Wine Type dimension, and with the USA dimension value in the Country dimension.



By using multiple dimensions to classify your Endeca records, you make it possible to perform navigation queries that include multiple dimensions, like those in the following example:

Navigation query 1 = Red	Navigation query 2 = USA	Combined query = Red + USA
Results = Bottle A Bottle B	Results = Bottle A Bottle E	Results = Bottle A

In this example, a *navigation query* was used. A navigation query is a query that returns a set of records based on user-selected characteristics along with any follow-on information.

If an Endeca record is tagged with dimension values from multiple dimensions, a user can navigate to the record using any of the dimension values. Thus, tagging a record with multiple dimension values creates multiple navigation paths to that record. The navigation queries in both of the following examples produce the same results:

Initial navigation query = Red	Initial navigation query = USA
Results = Bottle A Bottle B	Results = Bottle A Bottle E
Refinement query = Red + USA	Refinement query = USA + Red
Results = Bottle A	Results = Bottle A

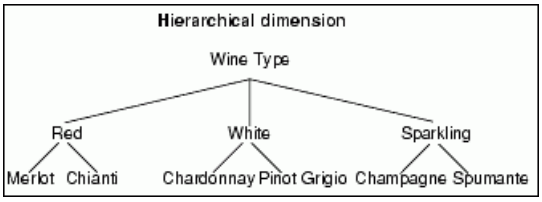
Dimension hierarchy

A dimension hierarchy is a logical structure that organizes your Endeca records.

The dimension values in a dimension are related to each other as *parent* and *child* dimension values.

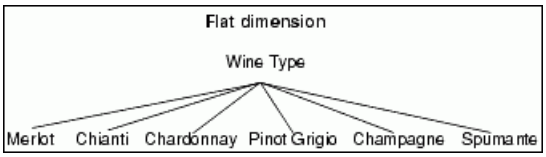
A dimension value that has sub-dimension values is the *parent* of those sub-dimension values. The sub-dimension values are *children* or *child dimension values* of their parent. Child dimension values that have the same parent are *dimension value siblings*.

The following figure illustrates a typical dimension:



Child dimension values enable users to refine their navigation queries and, consequently, the resulting record sets. Thus, when you configure dimensions, make sure that child dimension values contain more specialized data than their parents. (See the topic "Refining a navigation query in a hierarchical dimension" for a more detailed explanation of query refinement.)

Dimensions that have only one level of hierarchy beneath the dimension root are called *flat* dimensions. The following figure illustrates a flat version of the Wine Type dimension.



Automatically generated dimensions are flat unless you configure their dimension values to be of type Sift. The Sift type is an extension of automatic dimension generation that makes it possible to place automatically generated dimension values (of type Sift) in an existing hierarchical dimension. For information about how to create and use dimensions with Sift dimension values, refer to the *Endeca Developer Studio Help*.

Related Links

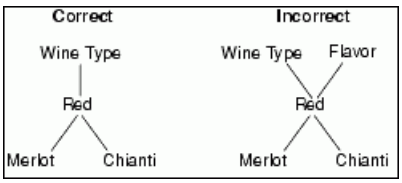
Refining a navigation query in a hierarchical dimension on page 19

The MDEX Engine returns not only the records tagged with the dimension value to which you navigated, but also those records that are tagged with any of that dimension value's children.

The one parent rule

A dimension value can have only one parent dimension value but any number of child dimension values.

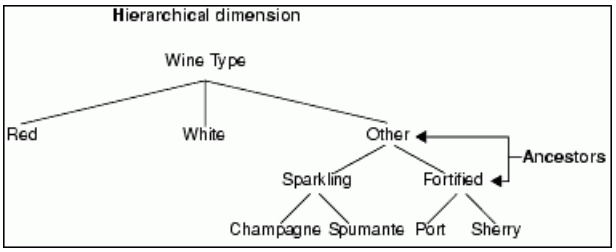
In the example on the left below, Red is the child of the Wine Type dimension value and the parent of the Merlot and Chianti dimension values. The example on the right is incorrect, because Red is specified as the child of both the Wine Type and Flavor dimension values. An Endeca dimension value can have only one parent.



Ancestors

The ancestors of a given dimension value are all the dimension values between the given dimension value and the dimension root. The parent of the given dimension value is also one of its ancestors.

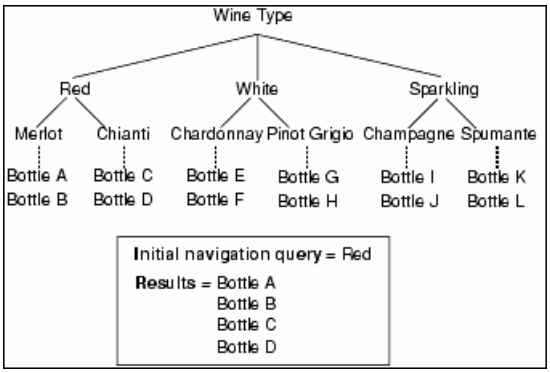
In the example below, Other and Fortified represent the ancestors for the Sherry dimension value.



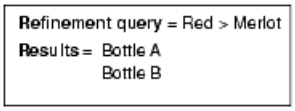
Refining a navigation query in a hierarchical dimension

The MDEX Engine returns not only the records tagged with the dimension value to which you navigated, but also those records that are tagged with any of that dimension value's children.

In the following example, a navigation query on Red would produce a result set of four bottles, Bottles A through D.



If you refine the query by navigating to one of Red's children (Merlot, for example), the result set is reduced in size and now contains only Bottles A and B.



Navigating from a dimension value to one of its children refines the navigation query and typically reduces the size of the result set. Conversely, navigating from a dimension value to the value's parent typically broadens the query and increases the size of the result set. The information required to create these refining and broadening queries makes up the bulk of the follow-on query information contained in the MDEX Engine's query results.

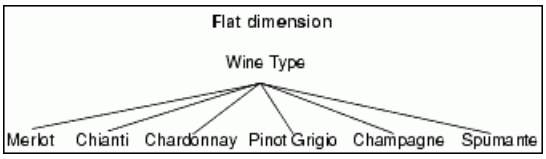
Note: Child dimension values are often referred to as refinement values or refinements.

Because navigation queries can cross multiple dimensions, the MDEX Engine's query results contain follow-on information for every dimension. This means that you can refine or broaden your query in one or more dimensions while making no change to the query parameters in the other dimensions.

Advantages of dimension hierarchy

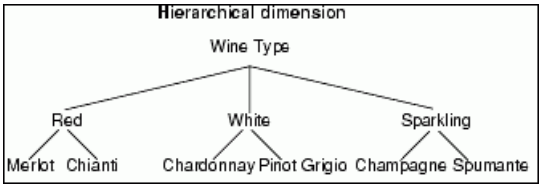
You can design dimension hierarchies in ways that reduce the number of follow-on queries that are presented to users as they navigate.

For example, in the flat dimension below, a navigation query on the Wine Type dimension value would return six possible refinement queries, one for each child.



A simple flat dimension such as the one shown in the preceding figure is small enough to navigate through quickly. Larger flat dimensions, however, present too many choices for follow-on queries to be easily navigable.

Organizing data as dimensions reduces this information overload and provides for an easier, more intuitive navigation experience. In the hierarchical example below, the Wine Type dimension value has only three possible refinement queries: Red, White, or Sparkling.



A second reason to use dimension hierarchy is that, by limiting the number of refinement queries, you decrease the amount of time it takes for the MDEX Engine to return its results. Returning query results that contain follow-on information for 20 refinement queries is faster than returning query results that contain information for 2000 refinement queries.

Note: The time that the MDEX Engine takes to process a large flat dimension can be lessened by requesting that the MDEX Engine return only the most important refinements.

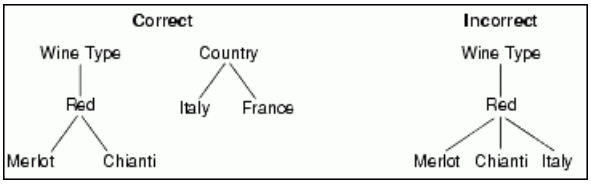
Additional dimension hierarchy recommendations

Creating easy to understand dimension hierarchies makes it easier for your end users to navigate through your data.

This section provides information about how to make dimensions as easy to understand and navigate through as possible.

Using a consistent theme

All the dimension values in a dimension should be part of the same theme.



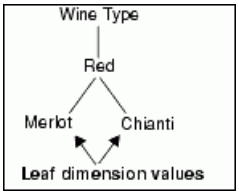
Because navigating on a dimension value implicitly navigates all of the dimension value's children, a child dimension value that does not have the same theme as its parent can produce a result set that contains Endeca records with multiple unrelated themes.

Making a theme increasingly specialized

Child dimension values should be specializations of their parent's theme. Child values that are not more specialized than their parents can create nonsensical navigation paths. In the incorrect example above, it makes good sense to refine the Wine Type dimension value by navigating to the Red dimension value. Refining the Red dimension value by navigating to the Italy dimension value, however, does not make sense, because "Italy" is not a type of red wine.

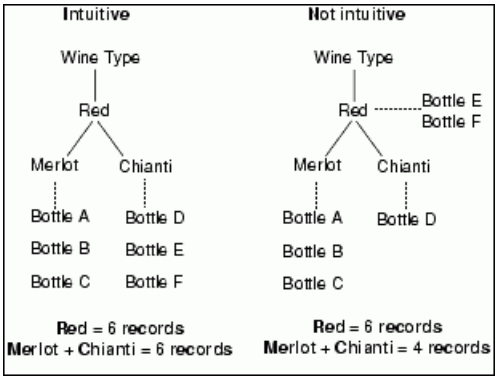
Tagging Endeca records with leaf dimension values

While it is technically possible to tag your Endeca records with any dimension value that exists in a dimension, it is highly recommended that you tag your Endeca records only with leaf dimension values. The following figure illustrates leaf dimension values in a simple dimension.



Tagging your Endeca records with leaf dimension values creates a more easily understood navigation experience, because the sum of the records in the leaf dimension values equals the number of records returned for the parent dimension value.

The following figure illustrates the difference between tagging all records with leaf dimension values and tagging some records with dimension values that are not leaves:



As the preceding figure shows, tagging all records with either of the two leaf values Merlot or Chianti produces the same result set for the query on "Red" and the query on "Merlot + Chianti". However, tagging two of the records (Bottle E and Bottle F) with a non-leaf dimension value ("Red") produces different result sets for the query on "Red" and the query on "Merlot + Chianti".

Creating dimensions and tagging records

You build dimensions and their hierarchies in Endeca Developer Studio.

Records are tagged with dimension values when you process your source data into a set of MDEX Engine indexes, using the Endeca Information Transformation Layer.

Alternatively, you can set up your Endeca application to derive dimensions automatically from the source data's properties. Remember, however, that automatically derived dimensions are flat, unless you specify Sift as the type of the dimension values in the dimension. In this case, the automatically generated dimension is a hierarchy

of Sift dimension values. Otherwise, however, you must create hierarchical dimensions manually, in Developer Studio.

Chapter 3

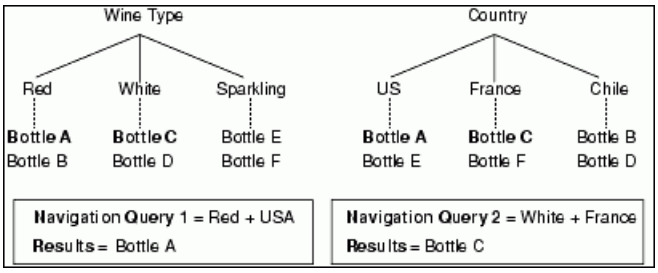
Understanding Guided Navigation

This section describes how Endeca's Guided Navigation provides a powerful and intuitive experience to users as they navigate through a data set.

The anatomy of a navigation query

In its most basic form, a navigation query is a combination of one or more dimension values.

These dimension values are referred to as the *navigation descriptors*. A navigation query instructs the Endeca MDEX Engine to return the set of records that represents the intersection of all the dimension values that it contains. For example, in the illustration below, Bottle A represents the intersection between the Red and USA dimension values. Bottle C represents the intersection between the White and France dimension values.



When an intersection does not exist between all of the dimension values in a navigation query, that query is considered a dead end. For example, in the illustration above, the Sparkling and Chile dimension values have no bottles in common and, therefore, no intersection. (In other words, there are no sparkling wines from Chile.)

The MDEX Engine automatically removes the possibility of such dead-end queries by the way it structures the follow-on query information that it returns in its query results. This is the essence of Guided Navigation.

Note: A default navigation query uses all of the dimension values together (that is, it implies an AND operation). This is the type of query we are describing here. Oracle Endeca Commerce has features that allow you to implement other, more advanced types of queries. See the "Default and advanced navigation queries" topic.

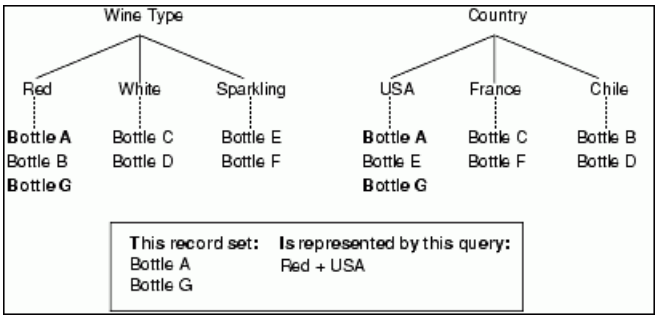
Related Links

Default and advanced navigation queries on page 24
Default navigation queries allow only one dimension value per dimension.

Identifying a record set using a navigation query

To extend the concept introduced in this section one step further, a specific record set can be represented by a combination of dimension values.

The example below adds another bottle of wine to our data set, Bottle G. Bottle G is tagged with the same dimension values as Bottle A (Red and USA). Executing a query on Red + USA would produce a record set including both Bottles A and G. Therefore, the record set of Bottles A and G can be represented by the navigation query "Red + USA".



Default and advanced navigation queries

Default navigation queries allow only one dimension value per dimension.

The following are examples of default navigation queries:

```
Red
Red + France
Red + France + 1996
```

In order for a record to be returned for a default query, it must be tagged with all of the dimension values in the query. In other words, a default navigation query is an AND query.

The Endeca MDEX Engine also supports more advanced queries that contain multiple dimension values from a single dimension. These values can be used in AND queries or OR queries. For example:

```
(Oak AND Berry) + 1996
Red + (France OR Chile)
```

The Endeca MDEX Engine does not support the use of OR queries across dimensions. In other words, a query for *France OR Red* is not allowed.

See the *Endeca MDEX Engine Development Guide* for more information on Endeca's advanced multi-select query options.

About Guided Navigation

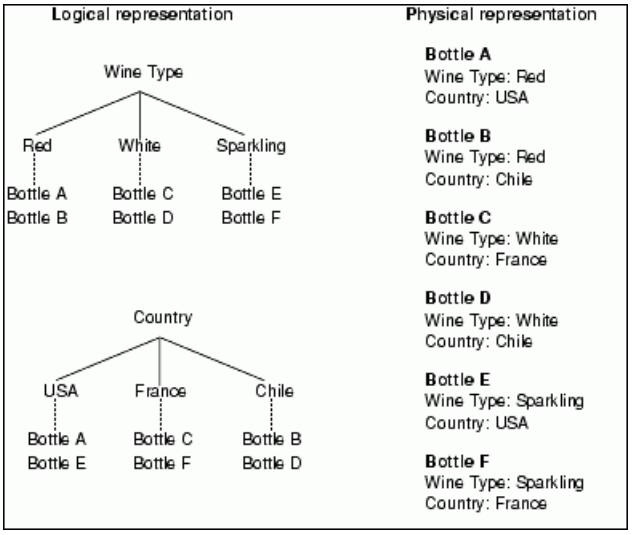
Guided Navigation is the presentation of valid follow-on refinement queries to the user.

You can think of Guided Navigation as the elimination of invalid refinement queries, or dead ends. To understand how the MDEX Engine accomplishes Guided Navigation, we'll look at the refinement information that the MDEX Engine returns in response to a query.

Note: Unless otherwise stated, everything said in this section pertains to both navigation queries and record search queries (a type of keyword search query described in the "Record search" topic).

The refinement information that is returned for a query is *specific to the record set that is returned for the query*. For example, if a returned record set contains only red wines, refinements associated with white or sparkling wines are not returned. It is the record data that is returned for a query, therefore, that drives which refinements are also returned with the query.

A dimension is a tree of dimension values. The tree structure makes it easy to imagine a specific location within a dimension. It also makes it easier to see the intersections among dimension values. However, the tagging of records with dimension values happens at the record level. Thus, in the following illustration the data in the column on the right more accurately represents the data in the MDEX Engine than the trees structures.



Every query returns a set of records that are tagged with dimension values, much like the physical representation above. From this representation, the MDEX Engine can quickly determine which dimension values are tagged to the returned records and which are not. This dimension value tagging determines what follow-on query information is returned along with the set of records:

- Any dimension value that is tagged to at least one record in the returned record set is considered a valid refinement (because querying the dimension value would return at least one record). The follow-on refinement queries for these dimension values are returned in the query results.
- Dimension values that are not tagged to any records in the returned record set are considered invalid. No follow-on query information is returned for these dimension values.

  To understand this concept better, remember that querying a dimension value returns the set of records that have been tagged with that dimension value. If none of the records in a record set have been tagged with a particular dimension value, then querying that dimension value would produce no results. This makes the dimension value an invalid refinement option.

When data is tagged with dimension values and follow-on queries are refined as described above, the MDEX Engine can dynamically build navigation paths among the Endeca records in a data set. Users can navigate along any path supported by the current record set, and the set of paths is updated as the record set changes. These features are the essence of Guided Navigation.

Related Links

Record search on page 31
A record search query is Endeca's equivalent to full-text search.

Dimensions and dimension values on page 14
Dimensions are logical categories that make it possible to organize your Endeca records into a hierarchical structure that customers can search for information.

Guided Navigation example

This section illustrates the Guided Navigation concept in the context of a typical Endeca application.

In the example, we use a simplified version of the wine UI reference implementation you used to test your Endeca installation. It also illustrates the use of dimension hierarchy.

Note: The example in this section deals solely with navigation queries. See the "Using Keyword Search" section for more information on record search queries, and using navigation queries and record search queries together.

Related Links

Using Keyword Search on page 31
This section describes the two types of keyword search queries: record search and dimension search.

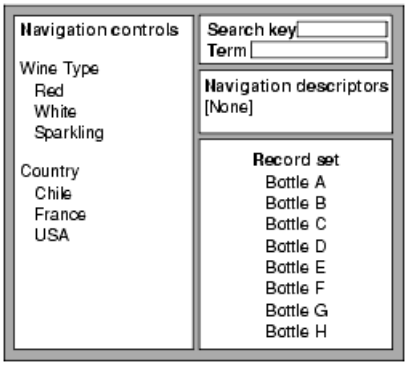
A typical Endeca application

The Endeca Assembler supports a wide variety of application features and user interface styles.

Commonly, however, the primary page of an Endeca application has these four components:

- Some type of navigation controls that contain the dimension values the user can select to navigate around the data set.
- A record set that is made up of all the Endeca records that are valid results for the current query.
- A set of navigation descriptors that tells the user which dimension values, if any, were used in the query that returned the current record set.
- Some type of search controls that allow users to perform keyword searches.

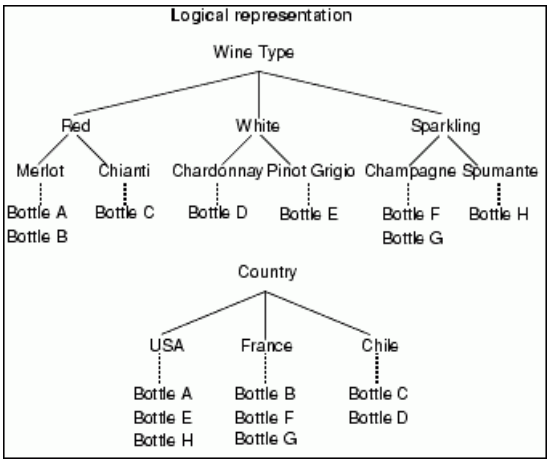
The following illustration shows a typical primary page for an Endeca application.



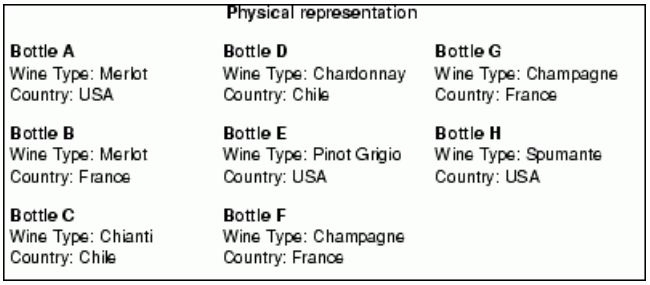
Source data

This topic shows the logical and physical representations of the wine store data.

The logical representation of the wine store data looks like the following:



The physical representation of the wine store data looks like this:

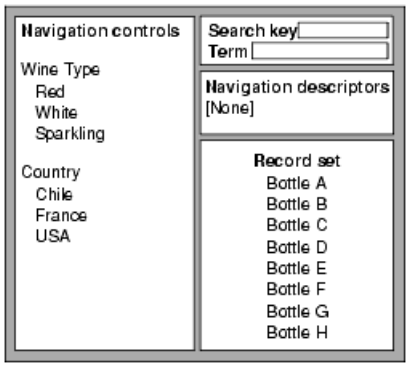


Working through the example

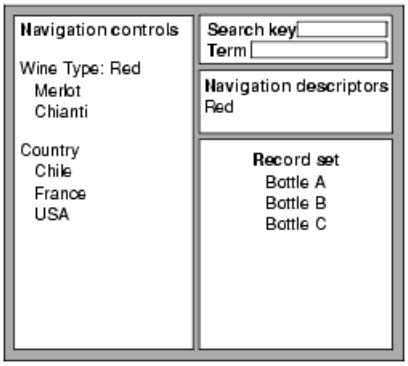
In the UI reference implementations, you start by looking at the entire data set (this is also known as starting at the root of the data set).

Starting at the root means that all of your Endeca records are displayed, and all of your refinement values are displayed, organized by dimension.

In the case of our wine store example, Bottles A through H are displayed in the Endeca record set, and the refinements in the Wine Type and Country dimensions are displayed in the navigation controls.



Step 1: We begin refining the list of bottles by selecting the Red dimension value from the Wine Type dimension. The Endeca Presentation API sends a navigation query containing the Red dimension value to the MDEX Engine which returns the query results.



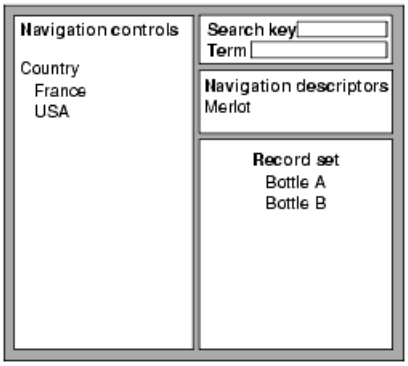
The sample application displays the following behavior:

- The record set is refined to include only bottles tagged with the Red dimension value or any of its children.

  Bottles A, B, and C are tagged with children of the Red dimension value so they remain in the record set. Bottles D through H are excluded.

- Red is added to the navigation descriptors area, indicating that the navigation query used to retrieve the current record set contained the Red dimension value.
- The navigation controls change to reflect the current record set, according to the rules of Guided Navigation:
  - Bottles A and B are tagged with Merlot, and Bottle C is tagged with Chianti, so those dimension values remain in the navigation controls.
  - None of the bottles are tagged with White or Sparkling, so those dimension values are omitted from the Wine Type dimension hierarchy. (This makes sense because a bottle of wine cannot be red and white, or red and sparkling.)
  - The Country dimension remains unchanged because Bottles A, B, and C are tagged with USA, France, and Chile, respectively.

Step 2: Next, we can refine our navigation query by selecting the Merlot dimension value.



The sample application displays the following behavior:

- The record set is refined to include only bottles tagged with the Merlot dimension value.

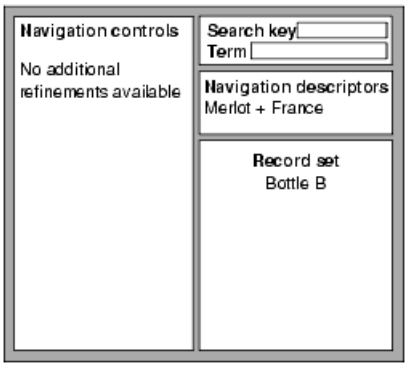
  Bottles A and B are tagged with Merlot, so they remain in the record set. Bottle C is excluded.

- The Red dimension value is refined to Merlot in the navigation descriptors area, indicating that the navigation query used to retrieve the current record set contained the Merlot dimension value.

  Note: Default queries only allow one dimension value per dimension, so Merlot replaces Red in the query.

- The navigation controls change to reflect the current record set:
  - The Wine Type dimension is omitted from the navigation controls because all of the records in the current record set are tagged with Merlot, and Merlot is a leaf dimension value that cannot be further refined. In other words, the Wine Type dimension no longer needs to be displayed because there are no refinement values left to pick from it.
  - The Country dimension changes to reflect the current record set:
    - Bottles A and B are tagged with USA and France, respectively, so those dimension values remain in the navigation controls.
    - Neither bottle is tagged with Chile, so the Chile dimension value is omitted from the Country dimension hierarchy.

Step 3: Next, we can select France from the Country dimension to see the effects of navigating across multiple dimensions.



The sample application displays the following behavior:

- The record set is refined to include only bottles tagged with both the Merlot and France dimension values.

  Bottles B is tagged with Merlot and France so it remains in the record set. Bottle A is tagged with Merlot and USA, so it is excluded.

- France is added to Merlot in the navigation descriptors area.
- The navigation controls change to reflect the current record set. In this case, the Country dimension is removed from the navigation controls because France is a leaf dimension value that cannot be further refined. This leaves our example with no more dimensions to navigate.

Chapter 4

Using Keyword Search

This section describes the two types of keyword search queries: record search and dimension search.

Record search

A record search query is Endeca's equivalent to full-text search.

Record searches return the following:

- A set of records based on a user-defined keyword(s).
- Follow-on query information, based on the returned record set.

Record search queries are performed against a particular property or dimension, also known as the *search key*. In order to perform a record search, the application's user:

1. Chooses an Endeca property or dimension to act as the search key.
2. Specifies a term, or terms, to search for within the key.

The set of records that a record search returns is made up of all records whose value for the search key contains the specified term(s). By default, if the record search query specifies multiple terms, a value must contain all of the terms in order for its record to be returned (the query is treated as an AND query).

Note: See the *Endeca MDEX Engine Development Guide* for information on advanced search queries using OR and Boolean logic.

Record search queries apply to a defined record set only (for example, the current record set). So a record search query is made up of two parts:

- A set of dimension values that identify the currently defined record set (in other words, a navigation query).
- The search key and term(s).

In essence, a record search query is a navigation query that is modified by a search key and terms. This is why the two queries, navigation and search, have identical data structures for their results: a set of records and follow-on query information.

Only properties or dimensions that have been configured for record search in Endeca Developer Studio can be used as search keys.

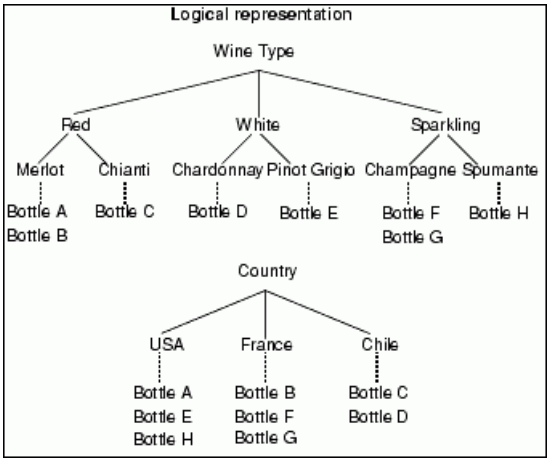
Using search interfaces

Endeca properties and dimensions that have been specified as searchable may also be combined into searchable groups called *search interfaces*. Search interfaces allow users to search across multiple properties/dimensions simultaneously.

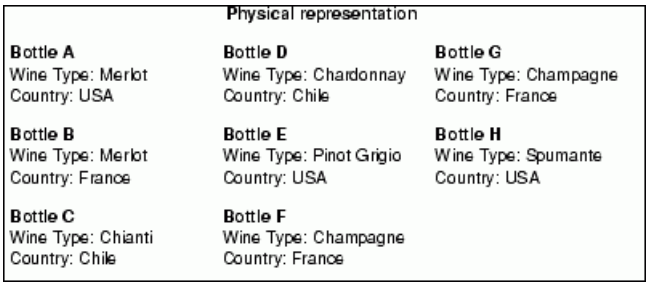
Integrated navigation and record search example

This section illustrates integrating record search queries with navigation queries to navigate to a specific record set.

It uses the same sample application and source data from the "Guided Navigation example" topic. As a reminder, the logical representation of the wine store data from the Guided Navigation example looks like this:



The physical representation of the wine store data looks like this:



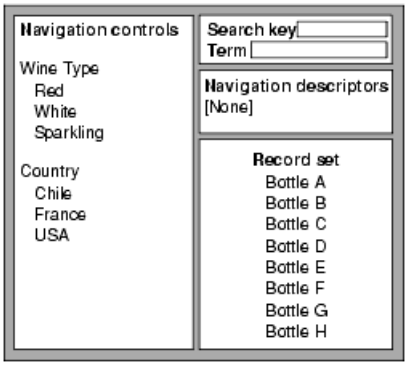
Related Links

Guided Navigation example on page 26
This section illustrates the Guided Navigation concept in the context of a typical Endeca application.

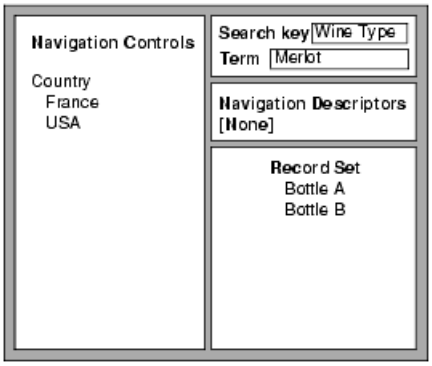
Working through the example

Again, in the UI reference implementations, we start by looking at the entire data set.

The figure below displays all of the Endeca records and all of the refinement values, organized by dimension.



Step 1: We begin by performing a record search query for *merlot* in the Wine Type dimension. A record search query containing the search key and search term (Wine Type and Merlot, respectively) is sent to the MDEX Engine and the query results are returned.



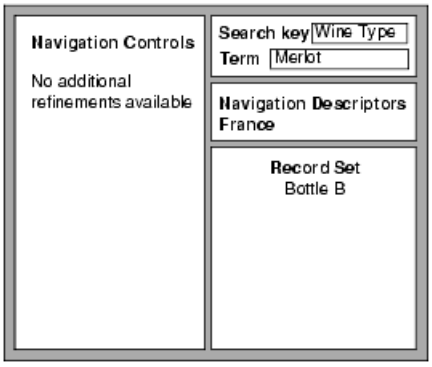
The sample application displays the following behavior:

- The record set is refined to include only bottles whose value for Wine Type includes (anywhere) the term *merlot*.

  Bottles A and B fit this description so they remain in the record set. Bottles C through H are excluded.

- The navigation controls change to reflect the current record set, according to the rules of Guided Navigation:
  - The Wine Type dimension is omitted from the navigation controls because all of the records in the current record set are tagged with Merlot, and Merlot is a leaf dimension value that cannot be further refined. In other words, the Wine Type dimension no longer needs to be displayed because there are no refinement values left to pick from it.
  - The Country dimension changes to reflect the current record set:
    - Bottles A and B are tagged with USA and France, respectively, so those dimension values remain in the navigation controls.
    - Neither bottle is tagged with Chile, so the Chile dimension value is omitted from the Country dimension hierarchy.

Step 2: Next, we can select France from the Country dimension to see the effects of navigating after performing a record search. A navigation query for France, that has been modified with the Merlot search term, is sent to the MDEX Engine.



The sample application displays the following behavior:

- The record set is refined to include only bottles that are:
  - Tagged with the France dimension value
  - Have a Wine Type value that includes the text *merlot*

  Bottle B fits this description so it remains in the record set. Bottle A is excluded.
- The navigation controls change to reflect the current record set. In this case, the Country dimension is removed from the navigation controls because France is a leaf dimension value that cannot be further refined. This leaves our example with no more dimensions to navigate.

Dimension search

In addition to record search, the Endeca MDEX Engine supports a second type of keyword search called dimension search.

Dimension search queries return dimension values that have names that contain search term(s) the end user has specified. Unlike record search, dimension search does not require a search key. Dimension search always searches any dimension values that have been identified as searchable for the terms provided. You identify a dimension as searchable in Endeca Developer Studio.

The dimension values that are returned for a dimension search are, by definition, navigable. This means that you can use the information in the dimension search results to build navigation queries, providing your users a seamless transition from a dimension search paradigm to a navigation paradigm.

Default and compound dimension searches

Dimension search has two modes: default and compound.

The difference between the two dimension search modes is:

- Default mode returns single dimension values, organized by dimension.
- Compound mode returns sets of dimension values.

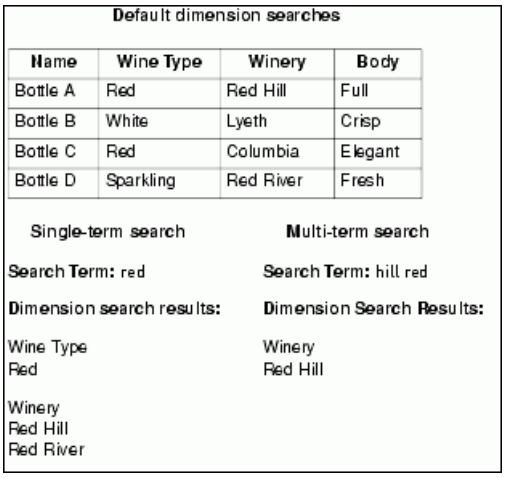
Default dimension search

Default dimension searches return individual dimension values, organized by dimension, for both single and multi-term queries.

If the user provides multiple terms, then a dimension value must contain *all* of the terms to be included in the results.

The example below illustrates both single term and multi-term default dimension searches. For the search term *red*, the MDEX Engine returns three dimension values: Red (from the Wine Type dimension), and Red Hill and Red River (from the Winery dimension). The dimension values are organized by dimension.

The multi-term default dimension search returns a single dimension value, Red Hill, because it is the only dimension value that contains both search terms.



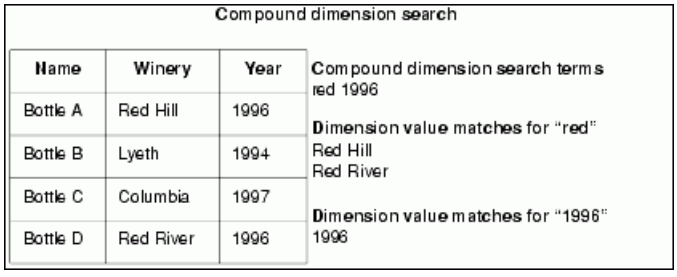
Compound dimension search

Compound dimension search extends the functionality of default dimension search by returning not single dimension values, but combinations of dimension values called navigation references.

A navigation reference is essentially a navigation query waiting to happen.

To start a compound dimension search, you provide the search terms. After the terms are provided, the following happens:

1. The MDEX Engine looks at each term individually, and searches for any dimension values that match the term. The result of this process is a list of dimension values.



2. The MDEX Engine combines the dimension values it found in Step 1 into navigation references, and tests the validity of those references.

   A navigation reference is valid if executing a query with it returns at least one record. In the example above, two navigation references are valid:

```
Red Hill + 1996
Red River + 1996
```

The MDEX Engine continues evaluating navigation references until it has assessed all possible combinations.

3. The MDEX Engine returns any valid navigation references it has found in Step 2.

Combining search queries

You can combine record search and dimension search queries into one consolidated MDEX Engine query.

You can combine search queries to give your users additional information as they navigate through your site.

Combining search queries enables you to retrieve not only a refined record set with follow-on query information, but also a separate set of navigable dimension values based on the same keyword(s). You can present this separate set of dimension values to the end user as a more targeted list of potential follow-on queries (in addition to the standard follow-on queries that accompany a record search).

When you pass multiple types of search to the MDEX Engine in one query, the exact handling of the queries is determined by your application tier components:

- The Endeca Assembler attempts to aggregate queries prior to sending them as a request to the MDEX Engine. The Assembler then processes the returned result objects into ready-to-render content items.
- The Endeca Presentation API treats each search as an individual request to the MDEX Engine. The search queries are completely independent of each other, and the MDEX Engine returns a result object for each search type. The Presentation API then combines these multiple result objects into one query result object.

Note: When combining record search and dimension search, keep in mind that record searches require a search key and one or more search terms. Dimension searches use only search terms.

Comparing dimension search and record search

Dimension search and record search each have their own strengths.

In general, you should:

- Use dimension search when the search terms are included in your dimension hierarchy.
- Use record search when you want to search unstructured data that is not part of the dimension hierarchy.

Dimension search is more appropriate than record search when the value for the search key does not consistently contain the search terms across all records. For example, not all red wines have "red" in their name, so a record search for *red* in the Name key would not return a complete set of red wine records. Dimension search, however, would return a Red dimension value that allows the user to navigate to a red wine record set (assuming you have tagged all red wines with a Red dimension value).

On the other hand, it is inappropriate to create dimension values out of certain types of property values that are long, wordy, or otherwise unsuitable (for example, descriptions or reviews). Because these property values are not included in the dimension hierarchy, they cannot be searched with dimension search. They must be searched using record search.

Additional search features

There are other search features that you can incorporate into your application.

This chapter gave you a brief overview of the types of search that you can implement in your Endeca application. There are many more search features that you can take advantage of, some of which are:

- Spelling functionality enables search queries to return expected results even though the user has misspelled the search term.
- Did You Mean functionality allows you to provide suggestions for further record searches to your users. Did You Mean is very useful when dealing with misspelled words or words that have exact matches that may not be as appropriate as other more popular alternatives.
- Stemming and thesaurus allow your Endeca application to consider alternate forms of individual words as equivalent for the purpose of search querying. For example, in many applications it is desirable for singular nouns to match their plural equivalents in the searchable text, and vice versa; this is an example of stemming. The Thesaurus feature allows the system to return matches for related concepts to words or phrases contained in user queries. For example, one might configure a thesaurus entry to allow searches for *Mark Twain* to match text containing the phrase *Samuel Clemens*.
- Relevance Ranking allows you to control the order in which results are returned. In particular, it is typically desirable to return results for the actual user query ahead of results for stemming and/or thesaurus transformed versions of the query.

See the *MDEX Engine Development Guide* for more information about the search feature.

Chapter 5

Understanding Cartridges

This section describes how you can use cartridges to expose features such as guided navigation, search, and spotlighting in your Web applications.

About Experience Manager

Experience Manager is a component of Oracle Endeca Workbench that enables business users to create pages and Web experiences within faceted search and navigation.

Experience Manager provides a flexible platform for creating and delivering content-rich customer experiences. It enables business users to deliver targeted, user-centric experiences that accelerate sales.

Using Experience Manager, business users can combine data-driven dynamic content with static content. They can target relevant content to the right customer at the right time, as well as guide and influence customers with advanced merchandising and recommendations capabilities.

Application pages

With Experience Manager, business users can rapidly create rich, data-driven application pages targeted specifically to a given context. Experience Manager pages can consist of a variety of promotional strategies:

- Product promotions
- Low-performing or lucrative categories
- Seasonal landing pages
- Targeted customer segments

About cartridges

Application pages consist of modular components known as cartridges, which business users create and configure to contain the content that they want to display.

For example, product spotlights, global navigation menus, faceted navigation menus, search results, slide shows, and image maps all can be implemented as cartridges. Each cartridge might have several fields to configure, or might be a container cartridge that contains additional nested cartridges within it.

Depending on the cartridge selected, business users might be asked to specify a path to an image or flash file, the default sort order for a results list, or the sequence of product records for spotlighting. The simplest

cartridges might need no configuration at all. Business users can also move cartridges around on pages -- within the constraints laid out by their application development teams.

A key aspect of cartridges is that they include a visual rendition for multiple channels like mobile devices or browsers. When a business user plugs a cartridge into a page, the cartridge renders itself. The cartridge is ready to render, though it might need further configuration.

Cartridge templates

Experience Manager uses cartridges based on cartridge templates created by application development teams. These teams can create cartridge templates depending on the specific needs of your business. Cartridge templates define the data structure for a cartridge along with the initial data values for that data structure. Experience Manager copies the data from a cartridge template when creating a cartridge instance on a page.

Cartridge templates also specify the form-based authoring interface for the cartridge that business users see in Experience Manager when they configure the cartridge.

Container cartridges

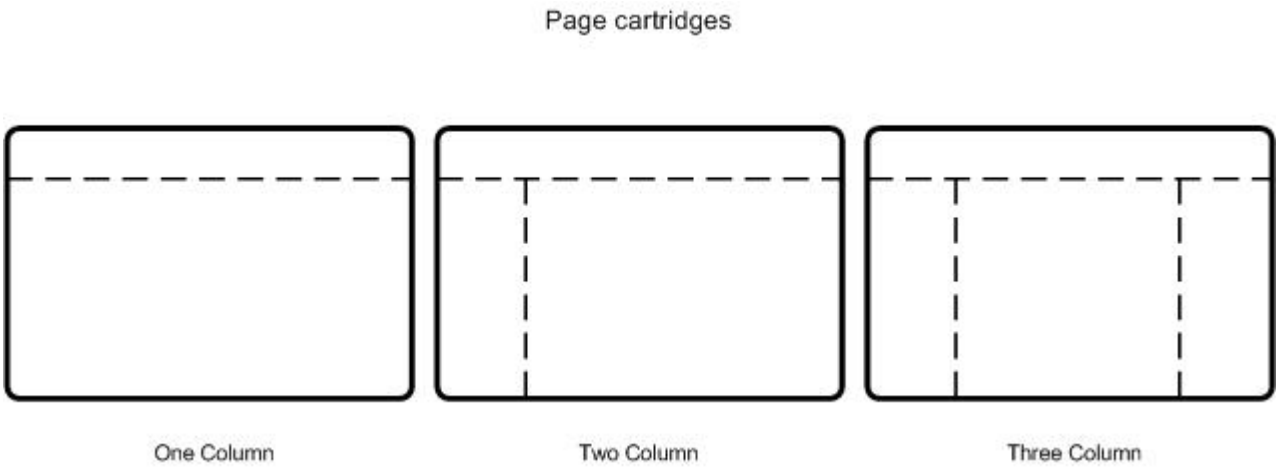
Container cartridges are cartridges that contain child cartridges.

A key consideration for container cartridges is that the system needs to provide a mechanism to restrict the types of children that can be placed into them. For example, it does not make sense to allow a business user to place a page cartridge inside of a banner cartridge. Thus, cartridges have a specific type. This type allows developers to control restriction rules for cartridge nesting.

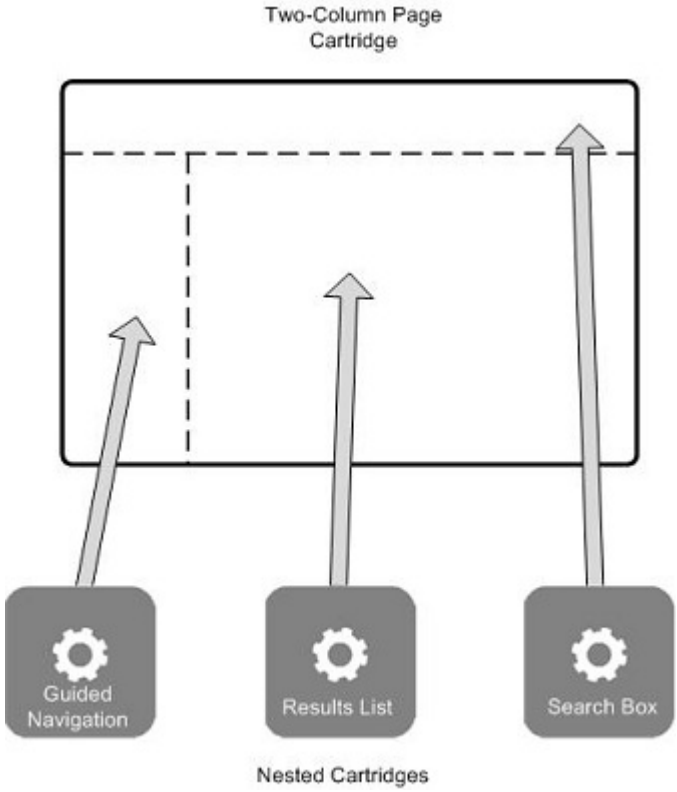
A page cartridge is an example of container cartridge. Pages share all of the characteristics of cartridges, with the following additions:

- Pages typically represent the "root" node in a response structure
- Pages contain child cartridges
- Pages can typically be referenced by external URLs

The following illustration shows the layouts of three page cartridges in the Discover Electronics store reference application.



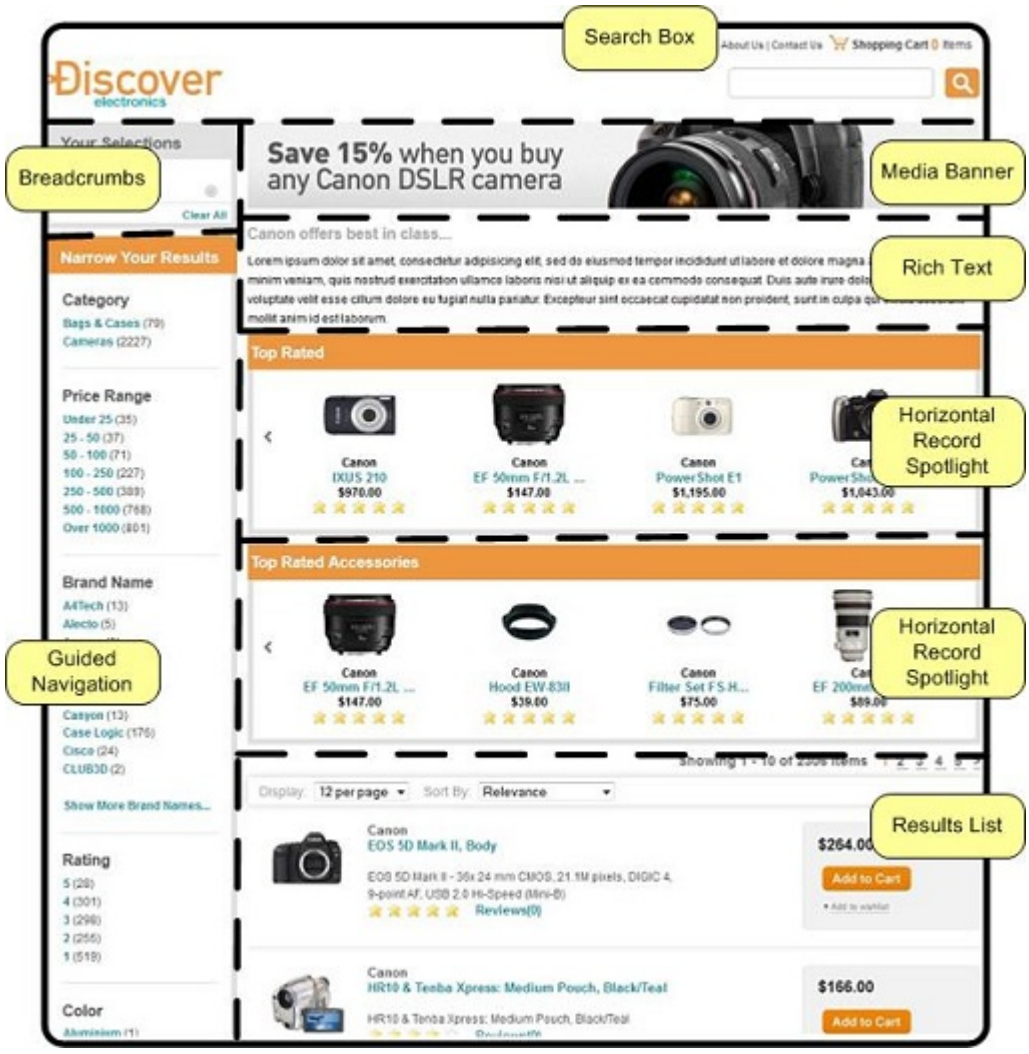
The following illustration shows how cartridges are plugged into a two-column page cartridge. The cartridges are nested in the container page cartridge.



Cartridge example

This section illustrates the cartridges concept in the context of a typical application using the Discover Electronics store reference application that Oracle Endeca Commerce provides.

This example shows where cartridges are plugged into the two-column page. The header contains the search box. The left column contains the breadcrumbs and guided navigation, while the right column contains the media banner, rich text, spotlights, and the results list.



The following table describes each of the cartridges plugged in to the page.

Cartridge	Description
Search Box	Provides site visitors with an interface for using keyword search in an application. You can also configure the cartridge to display auto-suggestion results.
Breadcrumbs	Displays navigation breadcrumbs in the sidebar of a page, so that the current navigation state is clearly visible to the site visitor. Visitors can clear refinements from the breadcrumbs individually or all at once.
Guided Navigation	Displays refinements for navigation. You can configure it to boost or bury selected refinements.
Media Banner	Displays images or video to the site visitor.
Rich Text	Include and format text.
Horizontal Record Spotlight	Dynamically showcases products to the site visitor in a banner format.

Cartridge	Description
Results List	Displays search and navigation results to site visitors. You can configure it to boost or bury selected records.

Index

O

P

Q

R

S

T