Navigating Dynamic Databases

Version 0.8
April 18, 2006

Dubberly Design Office
2501 Harrison Street, #7
San Francisco, CA 94110

415 648 9799

Contents

Introduction

Purpose

This document introduces several issues related to the problem of navigating multi-dimensional data spaces—large databases.

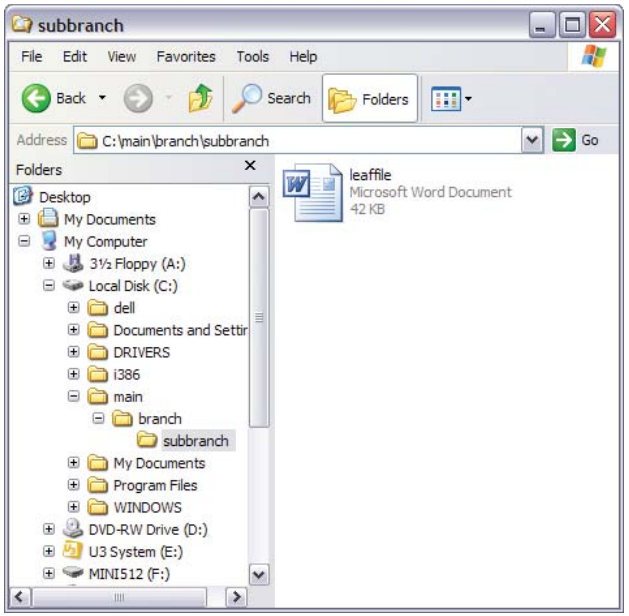
It examines problems with trying to conform data to a single taxonomy and the limits of tree structures as navigational devices.

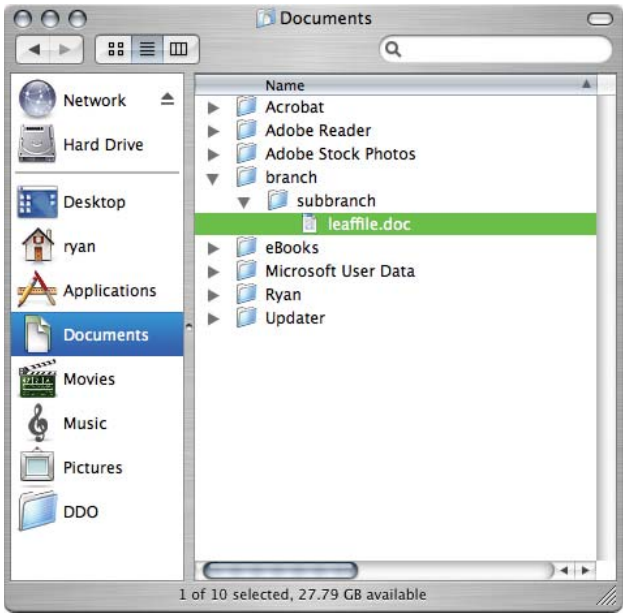
It offers several alternative devices, and it notes the need to enable random, multi-variate filtering so that users may narrow and expand at will. It also introduces the concept of pivoting: narrowing along one path and turning (or pivoting) to expand along another path.

Context

Over the past 25 years we have come to expect digital information to be organized in hierarchies. File systems are tree structures:

C:\main\branch\subbranch\leaffile.doc

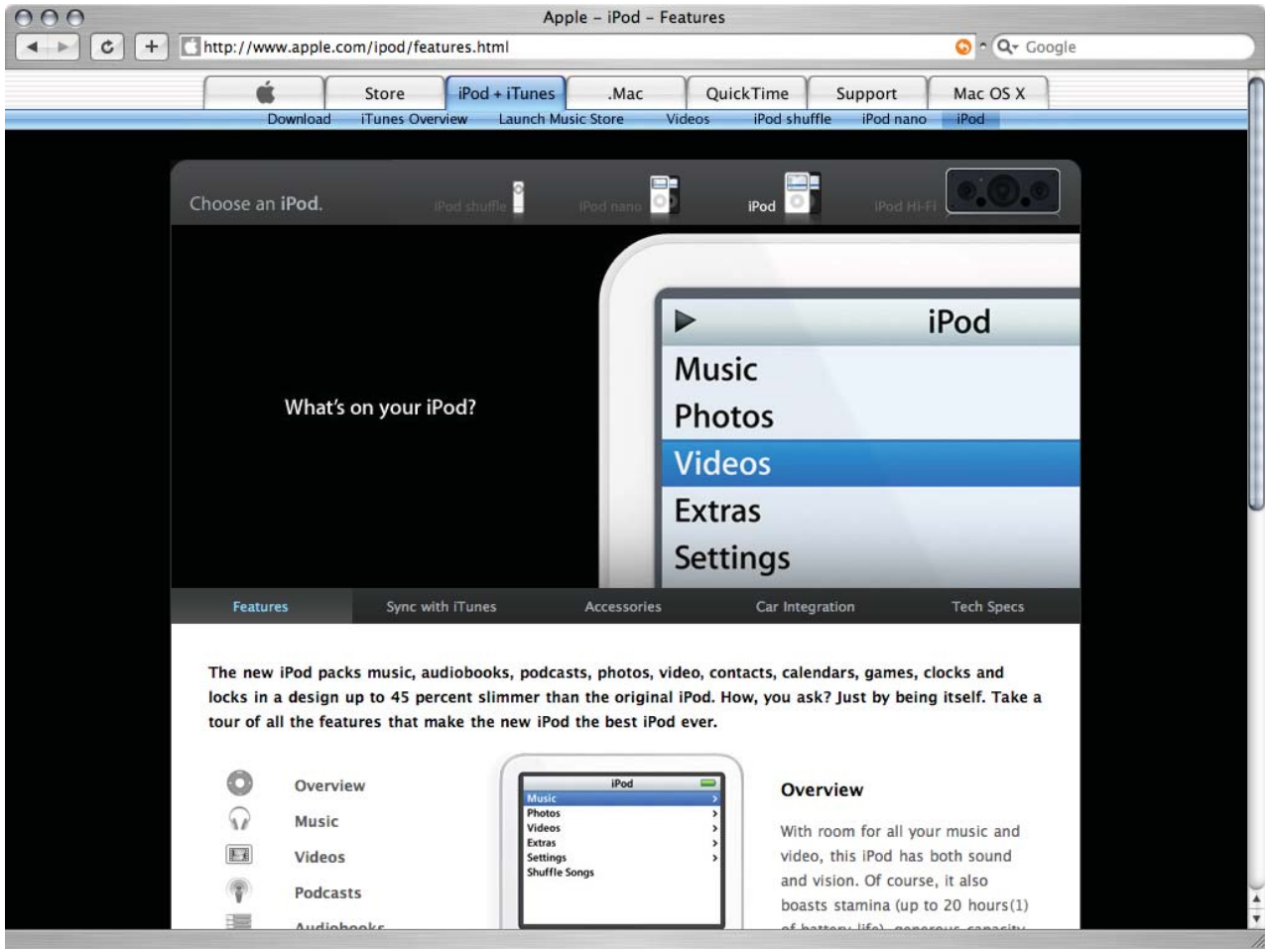




Context

The web, built on top of networked file systems, reinforced this form of navigation:

http://www.apple.com/ipod/features.html

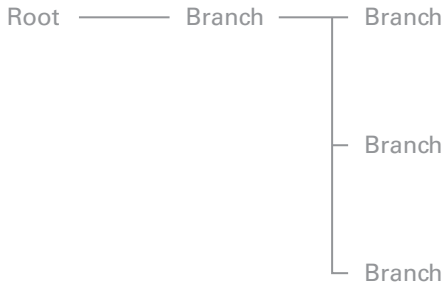


Introduction to Tree Structures

Tree structures begin with a root.

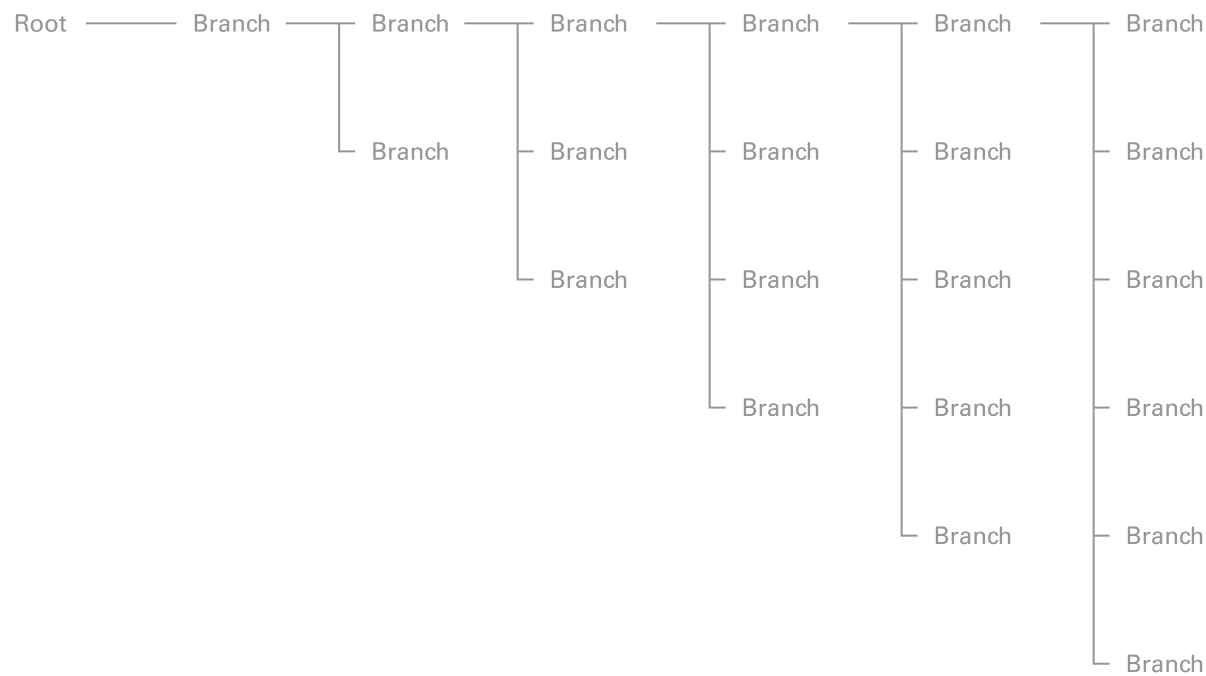
Introduction to Tree Structures

They also have branches.



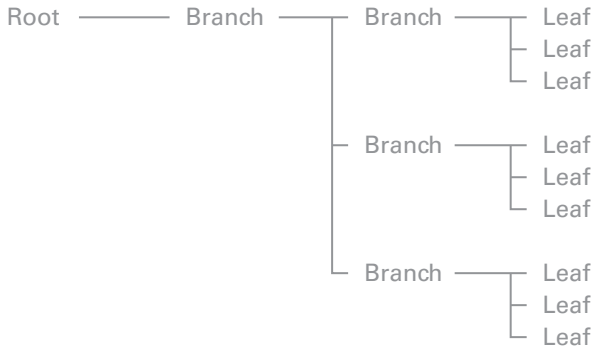
Introduction to Tree Structures

Imagine this process repeated. A tree can be as deep (or tall) as you want. For our purposes, we will restrict our tree to only three levels.



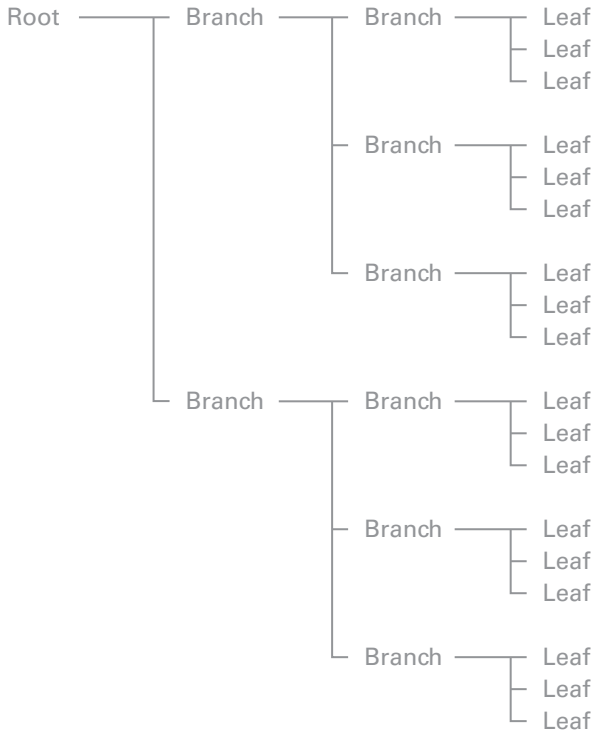
Introduction to Tree Structures

At the end of the branches are leaves.



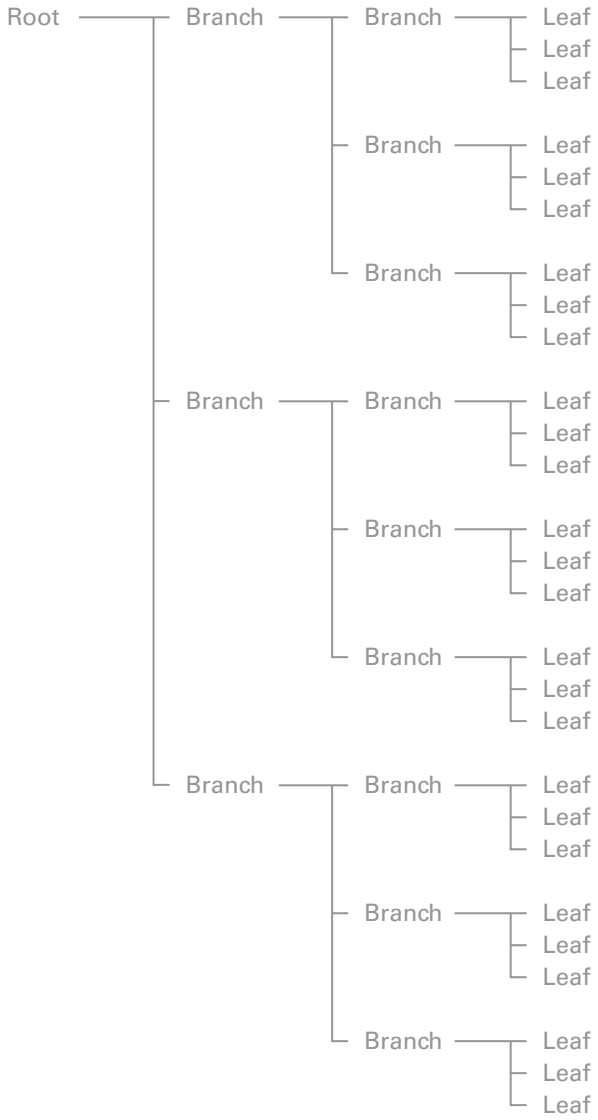
Introduction to Tree Structures

Imagine this process repeated
any number of times.



Introduction to Tree Structures

As wide (or broad) as you want.



Limitations of Tree Structures

Earlier, we saw that trees can be arbitrarily deep or wide. In practice, human perception has limits. Faced with many choices we may have difficulty comprehending them all. For example, many items at the same level in a menu may be difficult to parse.

George Miller's famous paper, *The Magic Number 7, Plus or Minus 2: Limits on Our Capacity for Processing Information* and research on short-term memory suggest a rule-of-thumb for designers. It may be best to limit the number of choices in a list to 7. If you have more than 7, it may be time to create groups—to nest categories.

That suggests the following corollaries:

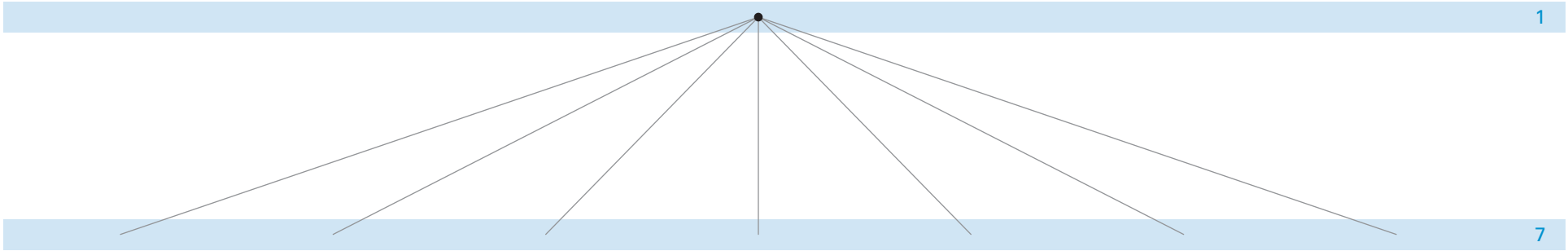
- No more than 7 tabs
- No more than 7 main sections in a web site
- No more than 7 main menu items
- No more than 7 items in between "spacers" in a menu.

Limitations of Tree Structures

The rule of seven has other consequences for site design. Imagine a root, a home page.

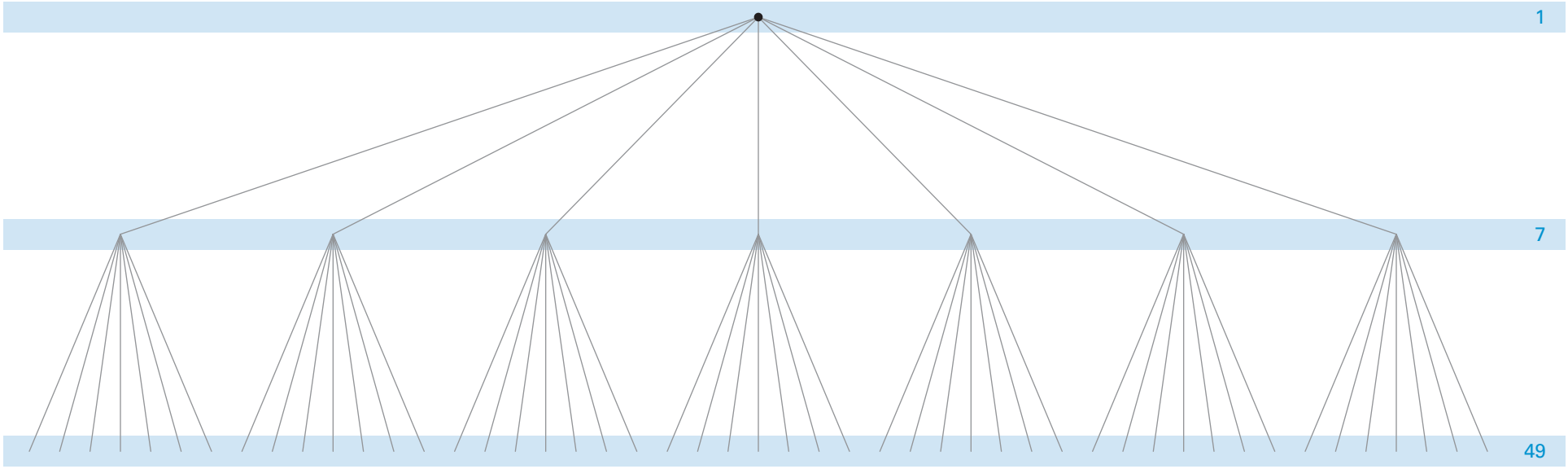
Limitations of Tree Structures

The site has seven main sections.
The second level of the tree has seven navigation paths.



Limitations of Tree Structures

If each section has seven sub-sections,
then the third level has 49 navigation paths.

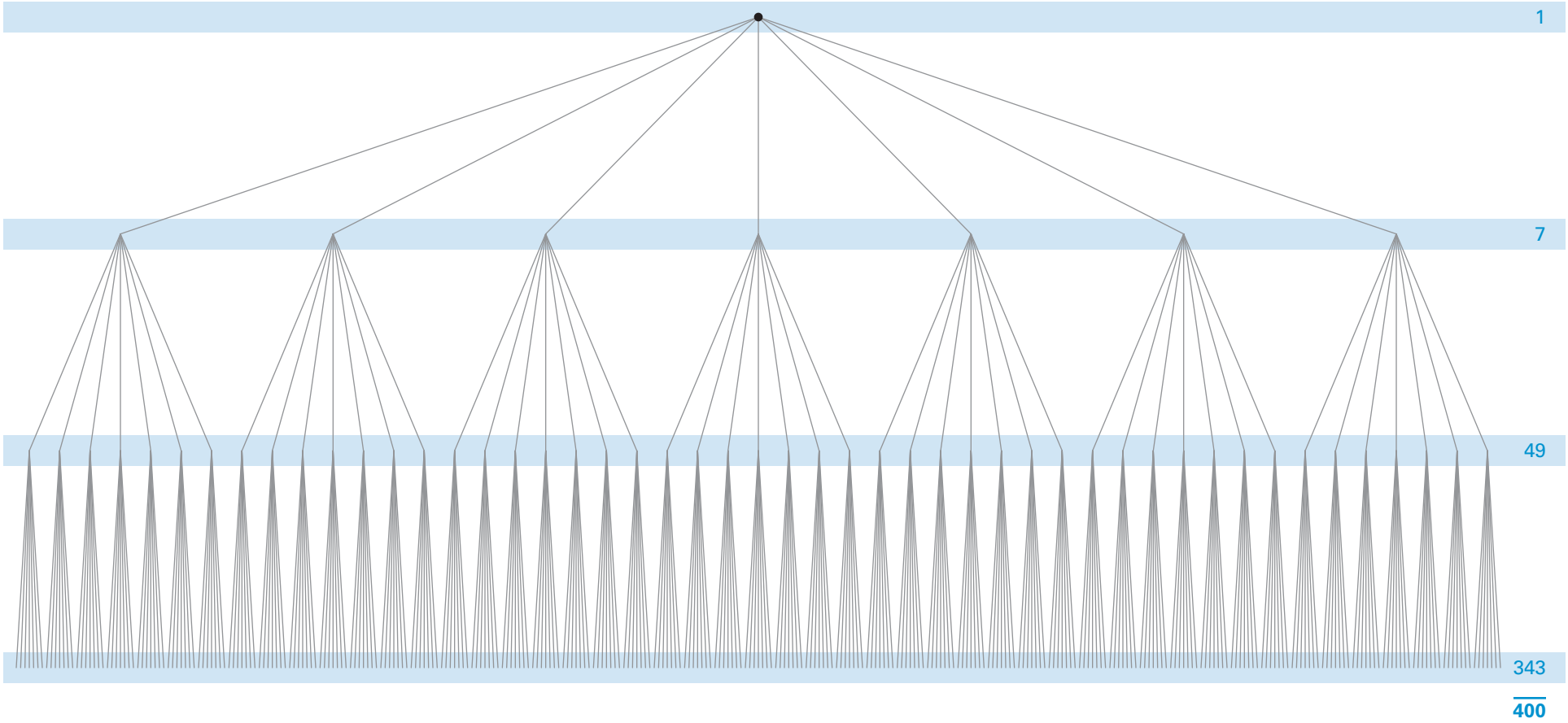


Limitations of Tree Structures

If each sub-section has seven pages, the site has 343 navigation paths.

Add up the series: 1 + 7 + 49 + 343 = 400. A conveniently 'round' number. As a rule-of-thumb, it provides a loose distinction between small and large sites. We contend tree-based navigation systems begin to fail on sites much larger than 400 categories.

Large sites, larger data collections, need other navigation structures, such as searching and filtering.



Wine Tree

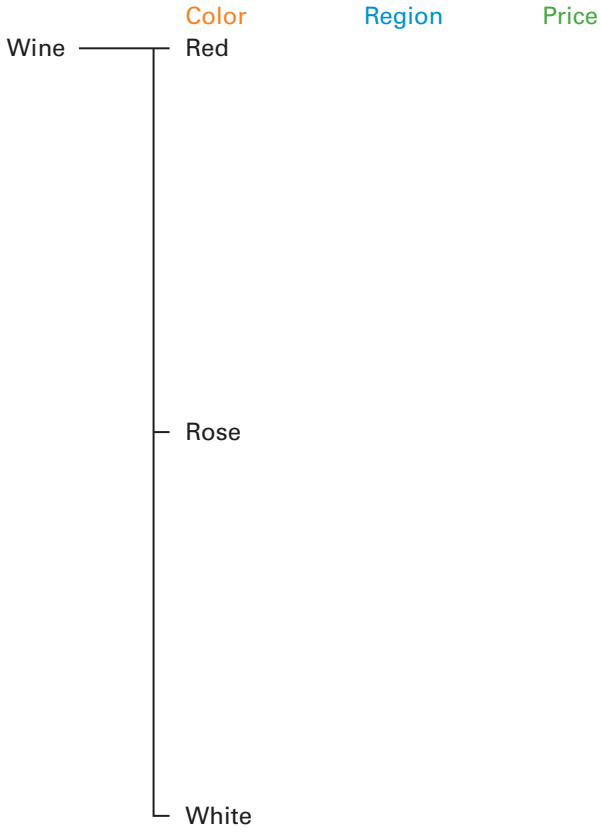
Let's look at an example. Say, for instance, that we have a collection of wine, and that we organize it by Color, Region, and Price.

A database could be of anything. It could be cars, for example, organized by Year, Type, and Make. It could be poems . . .

Wine Tree

Each category has its own sub-categories:

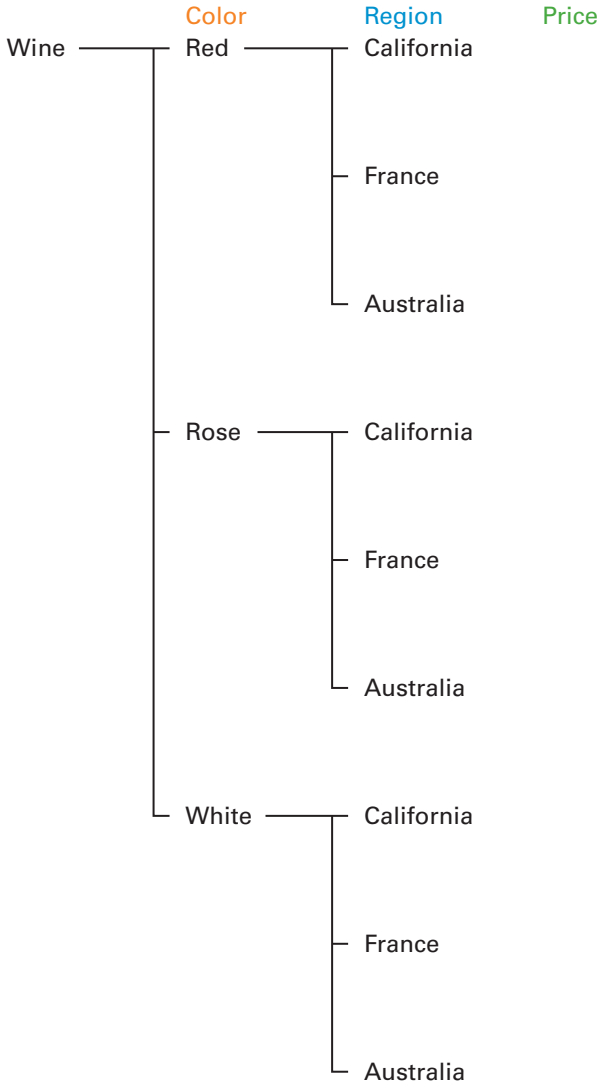
Color has Red, Rose, and White.



Wine Tree

Each category has its own sub-categories:

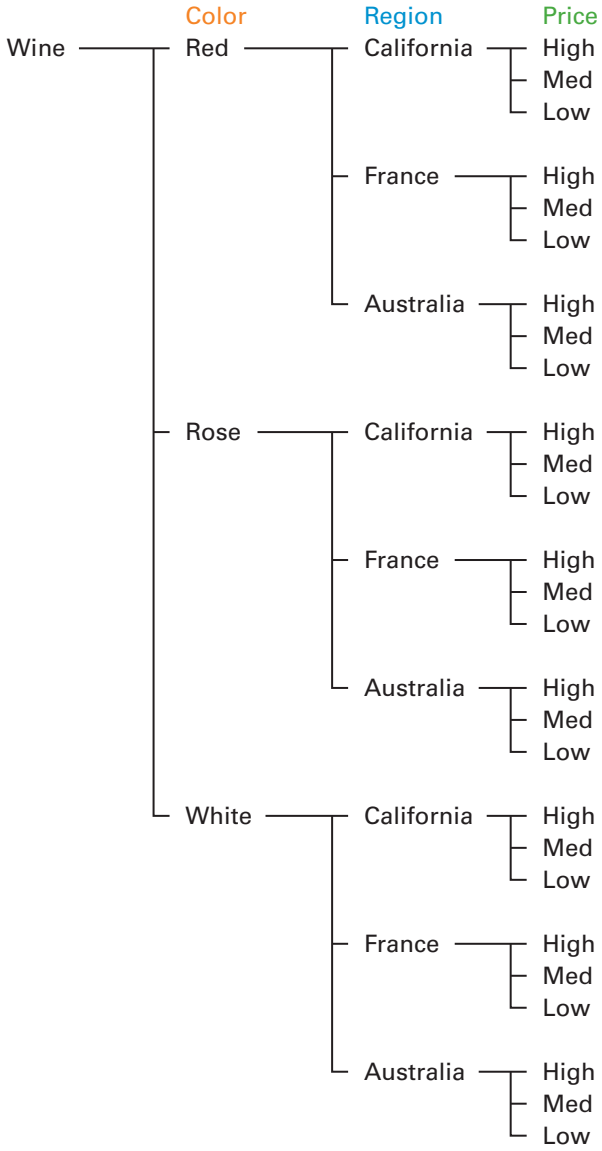
Region has California, France, and Australia.



Wine Tree

Each category has its own sub-categories:

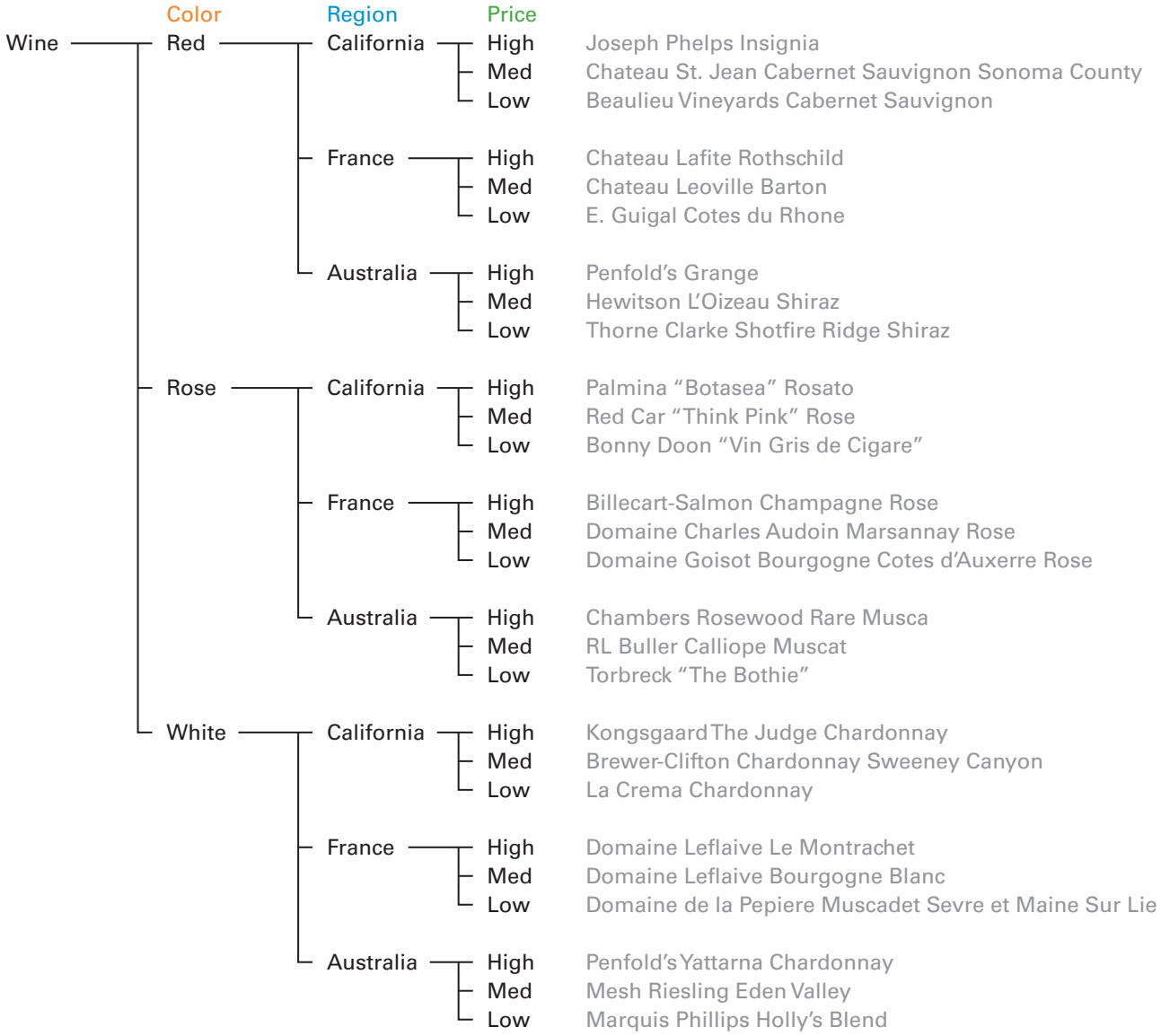
Price has High, Medium, and Low.



Wine Tree

In our example, the final result is a single bottle of wine that matches each of the categories.

In a wine store, each category might have multiple bottles, e.g., from several vineyards and multiple years. Of course, vineyard and years are fourth and fifth dimensions. Variety of grape is another dimension. We're keeping things simple for our illustration.



Wine Tree Combinations

When we created our directory tree, we randomly started with Color, then chose Region, and finally Price. That order might make sense if we're beginning our search with a goal of matching a wine to a food. However, we may want to begin with price or even region. The point is: No single taxonomy (tree structure) is best. Taxonomies are useful within a context—for a particular user, with a particular goal, at a particular time.

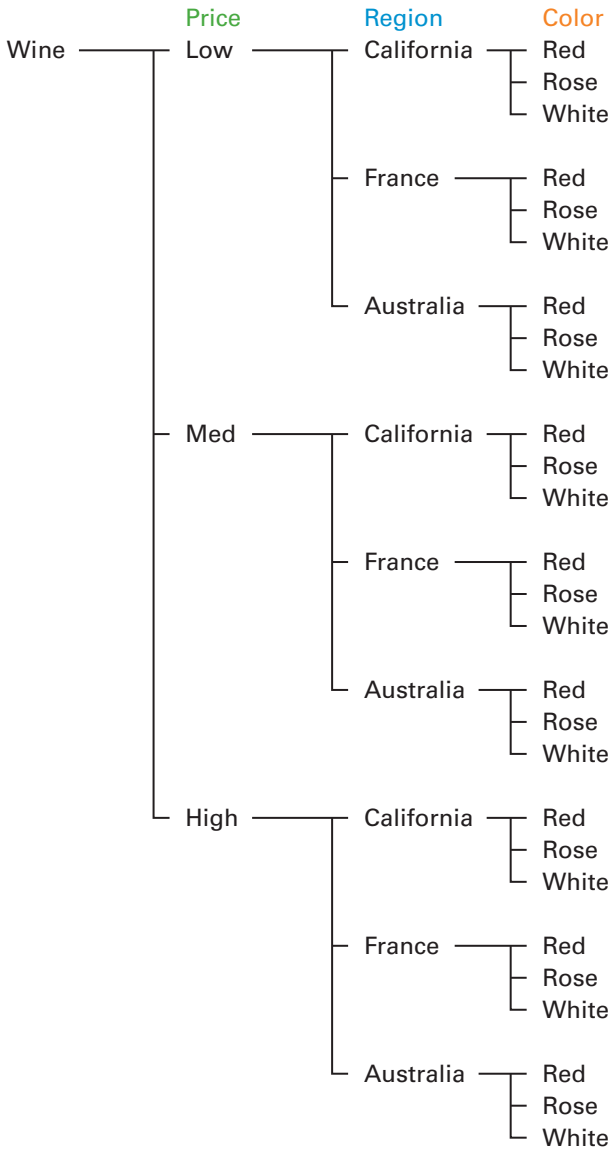
This set of information can be ordered in six ways (three unique elements in three positions yield six combinations).

1.	ABC	Price	Region	Color
2.	ACB	Price	Color	Region
3.	BCA	Region	Color	Price
4.	BAC	Region	Price	Color
5.	CBA	Color	Region	Price
6.	CAB	Color	Price	Region

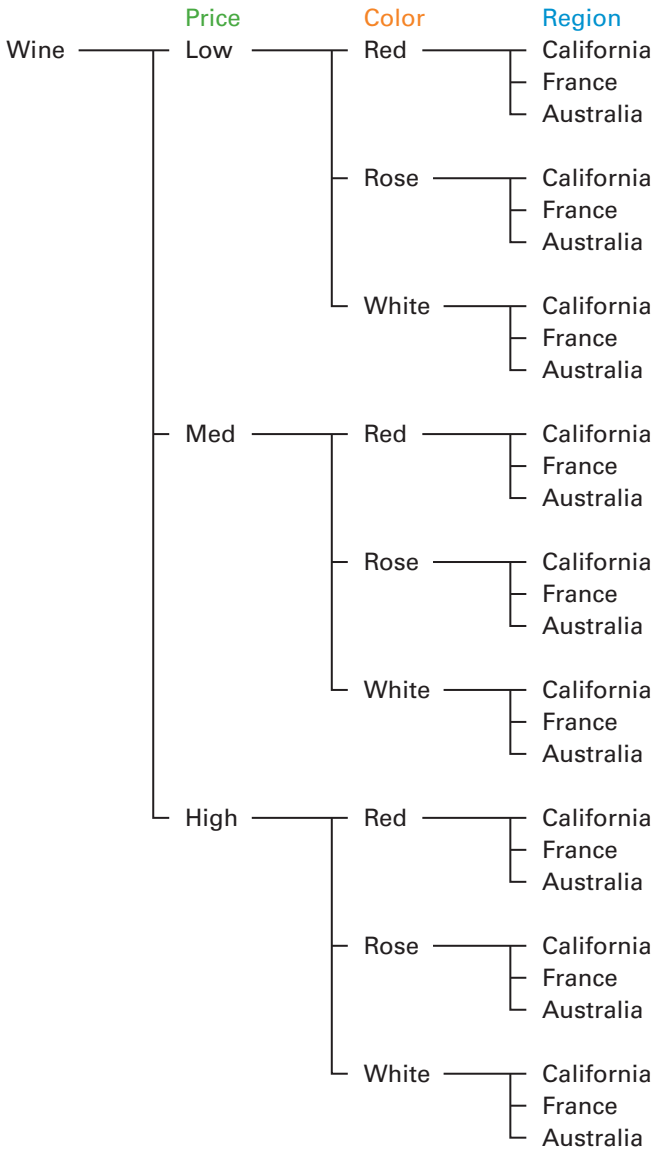
Real data is often much richer, allowing even more combinations.

Wine Tree Combinations

1. Price, Region, Color

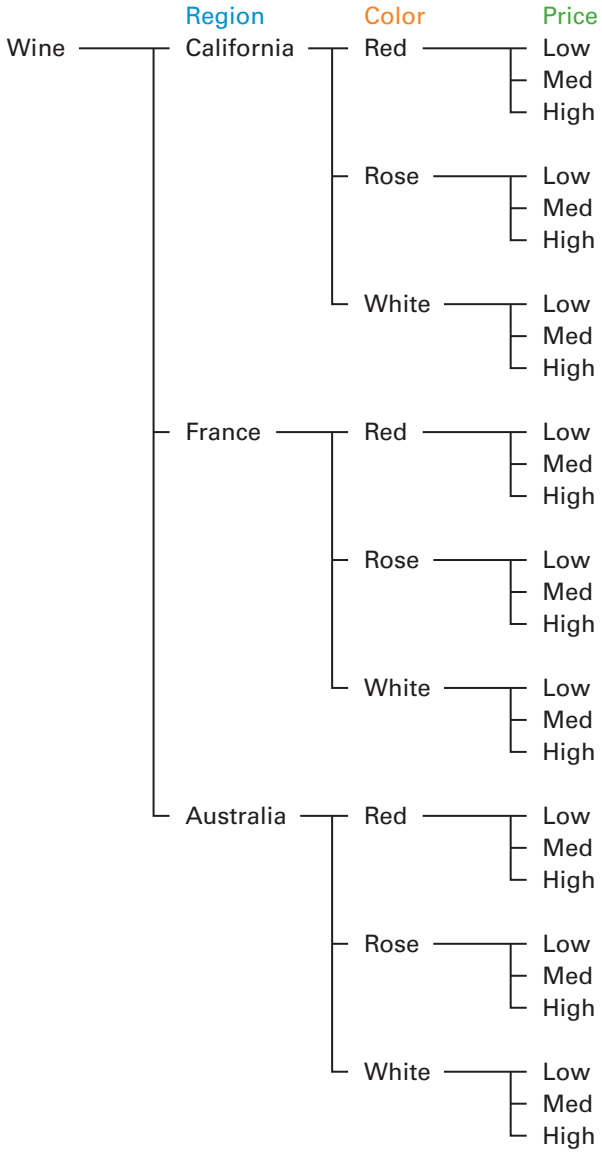


2. Price, Color, Region

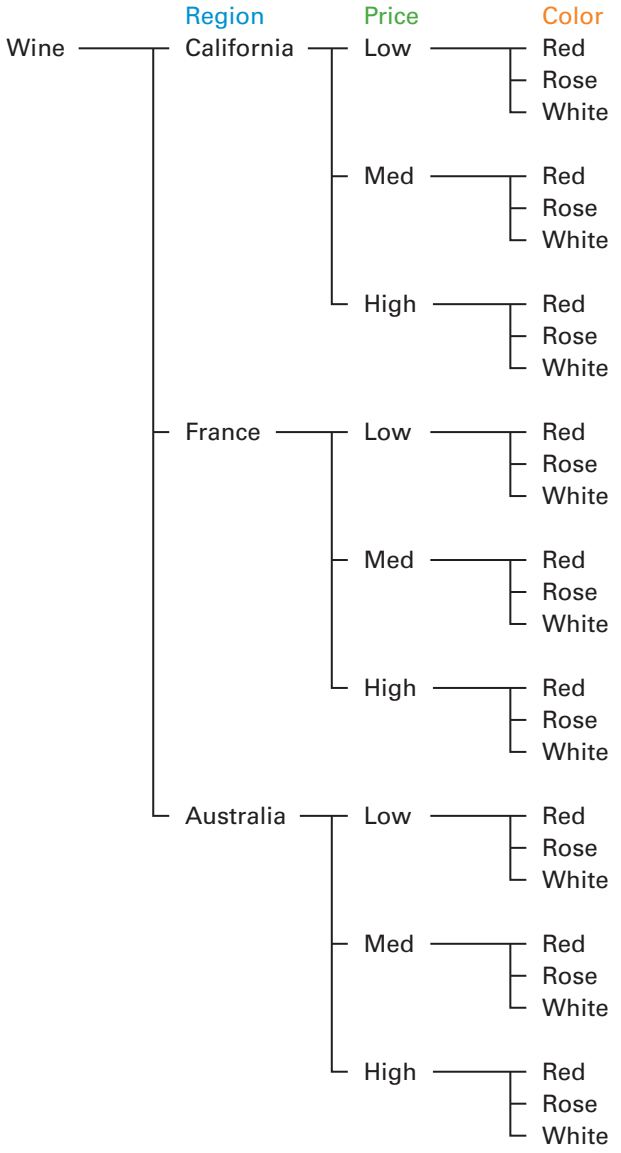


Wine Tree Combinations

3. Region, Color, Price

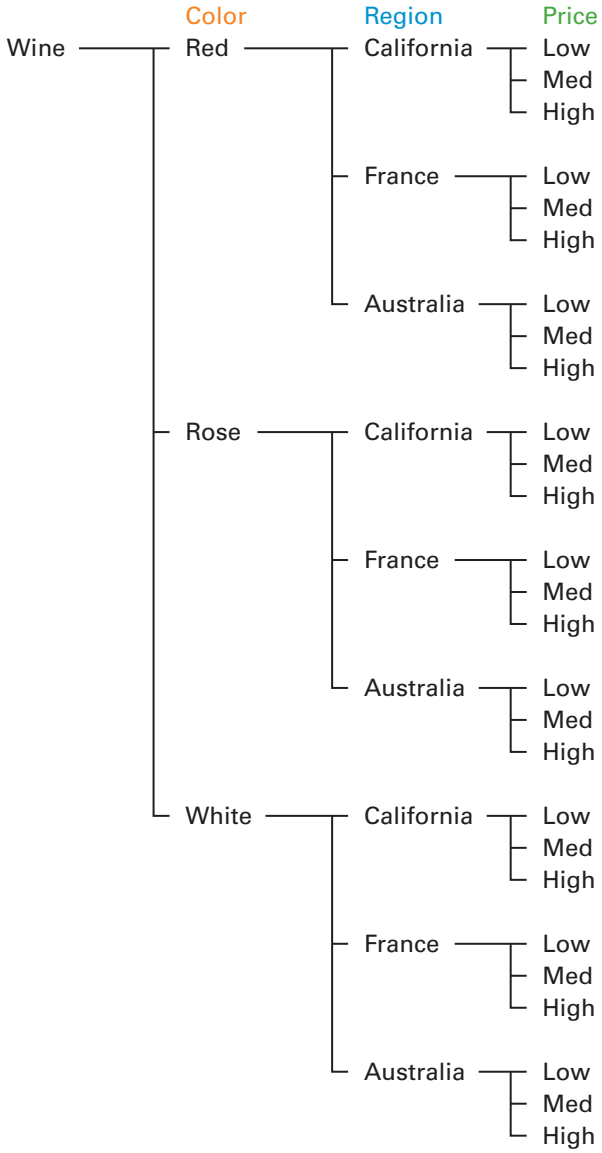


4. Region, Price, Color

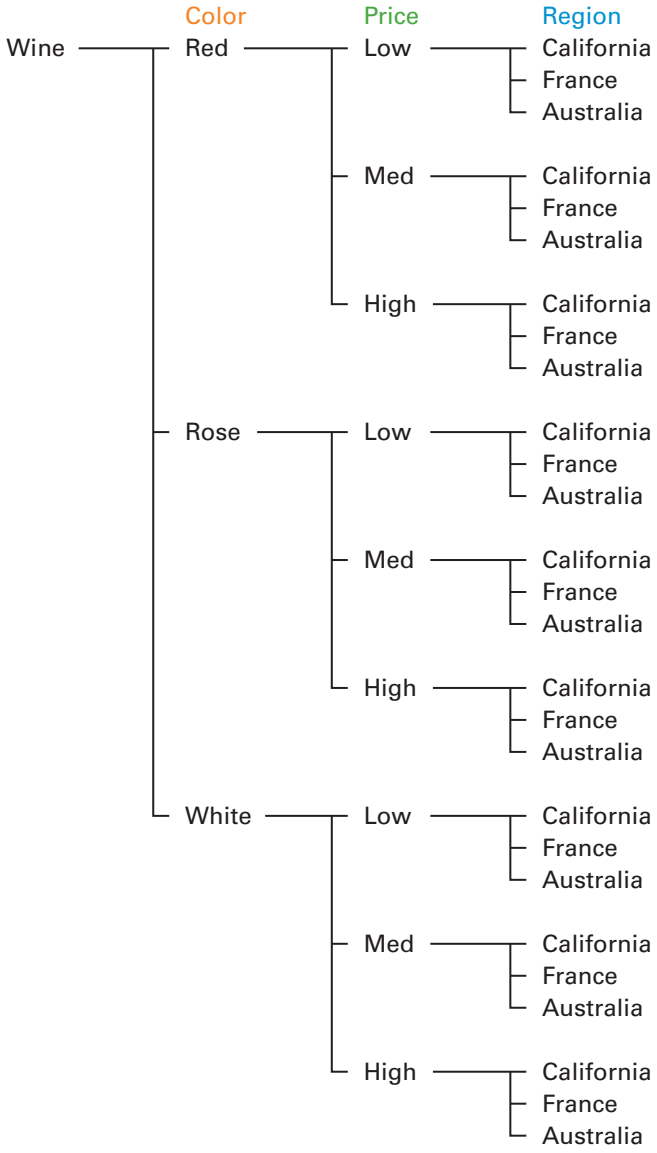


Wine Tree Combinations

5. Color, Region, Price



6. Color, Price, Region



Cube Structures

Any one of the 6 trees is a valid representation of the data—and any one might be useful. How should we think about the data? Is there a more "natural" form?

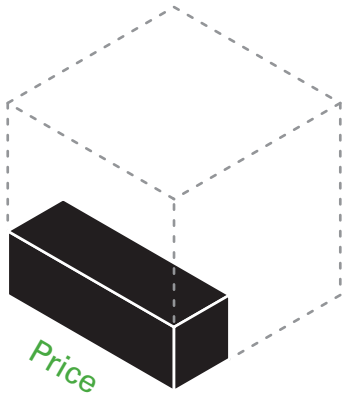
In this case, the data has three dimensions. The data suggests a cube. We might say that its natural shape is a cube. Of course, a real database might have many more dimensions. While 4 or 5 or 6 or more dimensional spaces are difficult to represent, we can describe data spaces as N-dimensional—with the ability to be sliced or filtered along each dimension.

The next section shows how that might work.

Wine Cube

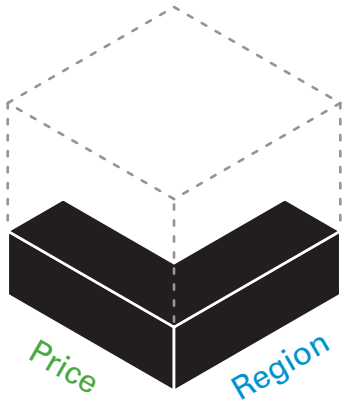
Using the same set of information as in our tree structure, we create the dimensions of the cube:

Price is the first dimension.



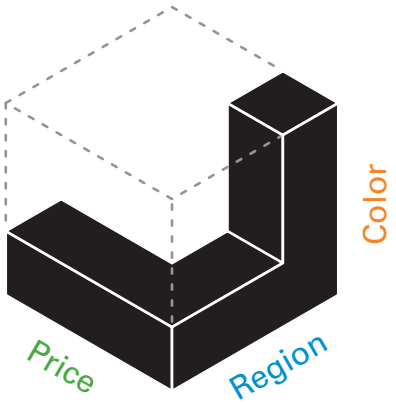
Wine Cube

Region is the second dimension.



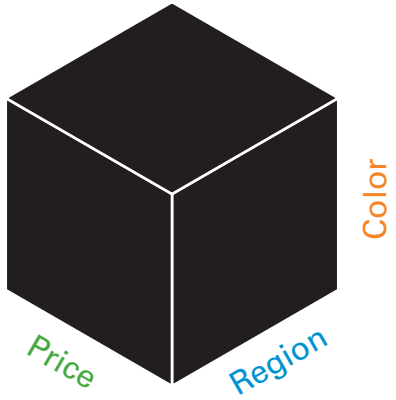
Wine Cube

Color is the third, and final dimension.



Wine Cube

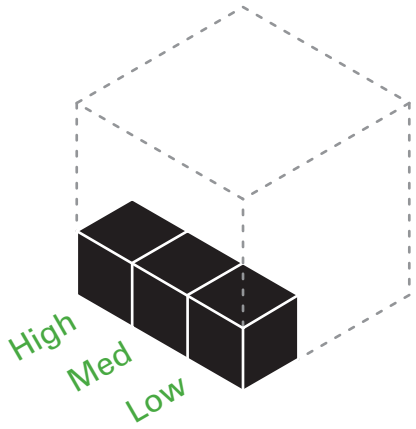
This completes our wine collection.



Wine Cube

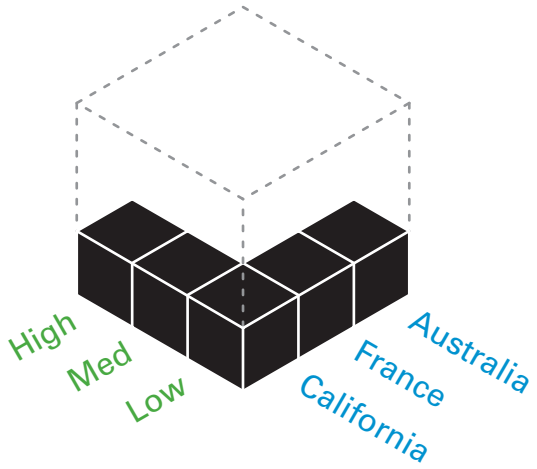
We also know that each of our three dimensions has its own sub-categories:

Price has High, Medium, and Low.



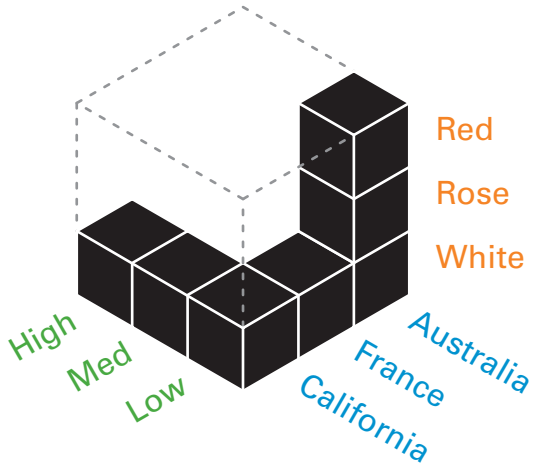
Wine Cube

Region has California, France, and Australia.



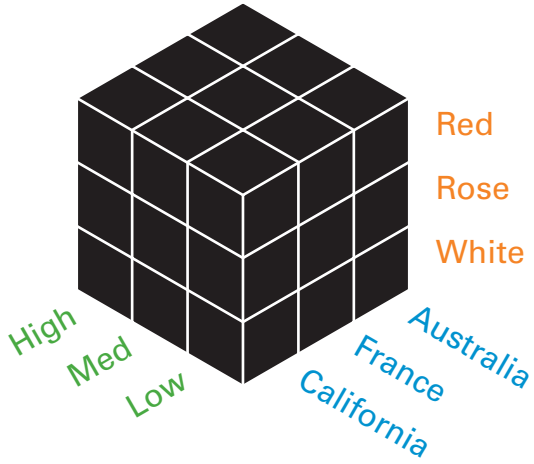
Wine Cube

Color has Red, Rose, and White.



Wine Cube

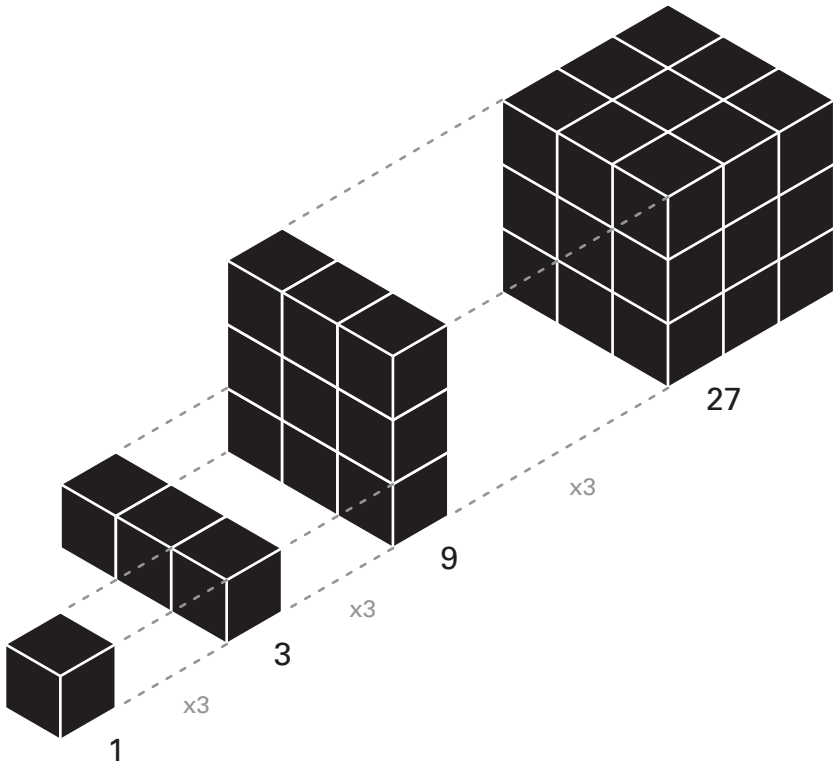
Adding these sub-categories divides our finished wine cube into 27 cells.



Wine Cube

Three cells make a row.
Three rows make a block.
Three blocks make up the cube.

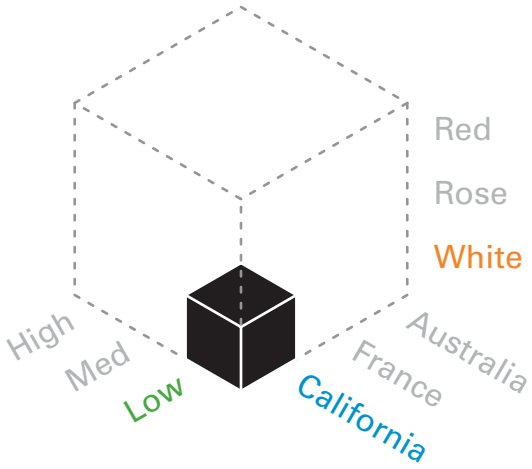
Below is a tally of single cells.



Wine Cube

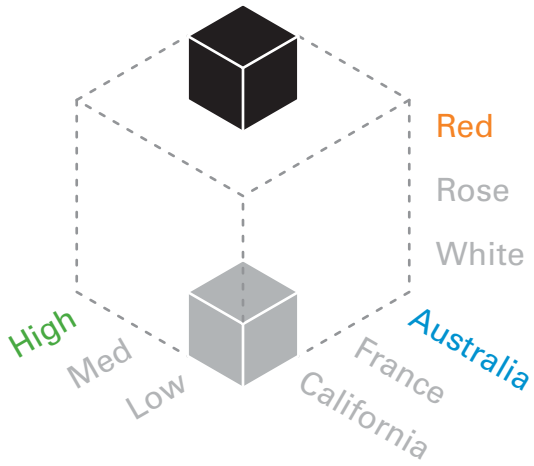
Each leaf in the tree corresponds to a cell in the cube. Each cell has coordinates: x, y, z.

1,1,1 = Low, California, White.



Wine Cube

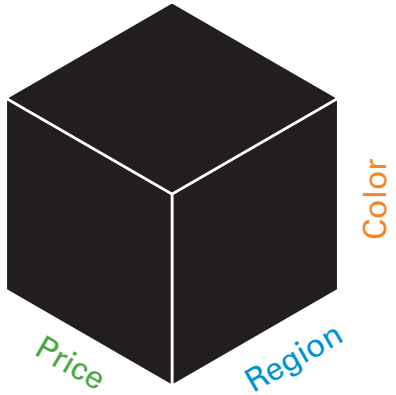
3,3,3 = High, Australia, Red.



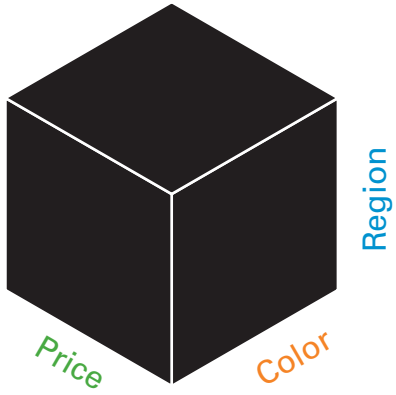
Wine Cube Combinations

Re-ordering the dimensions provides the same six combinations as the tree structure.

1. Price, Region, Color

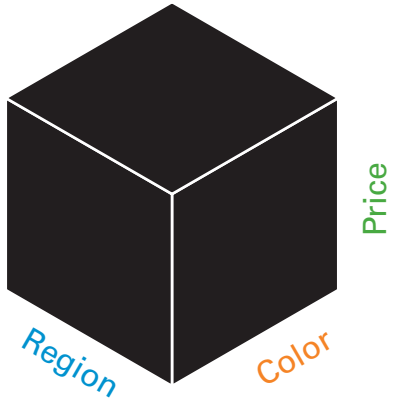


2. Price, Color, Region

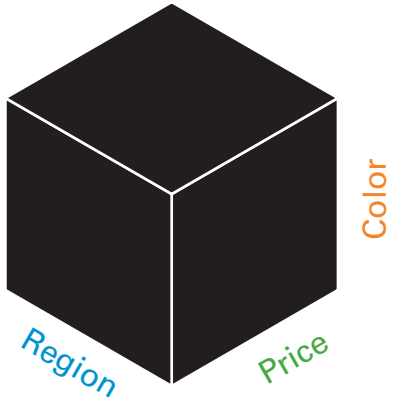


Wine Cube Combinations

3. Region, Color, Price

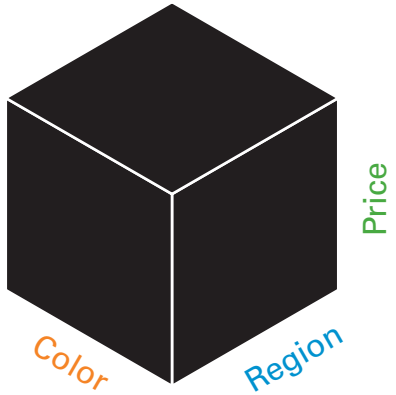


4. Region, Price, Color

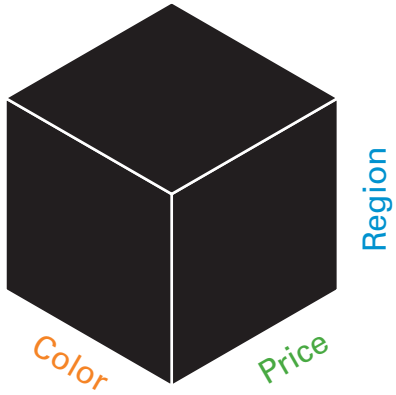


Wine Cube Combinations

5. Color, Region, Price



6. Color, Price, Region



Possible Interfaces

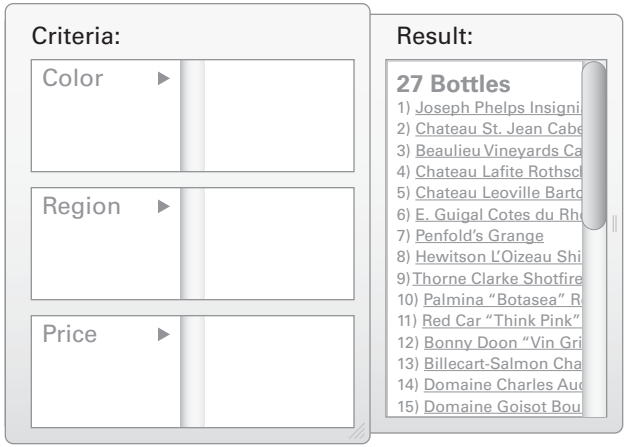
Now how would a user interact and navigate within this set of information? One could imagine a variety of interfaces to narrow down choices:
- Column List
- Pull-down Menus
- Sliders
- Checkboxes

The following pages show some examples of possible interfaces.

Column List

This interface is divided into our three categories but one could imagine it being very flexible to accommodate any number of categories and sub-categories.

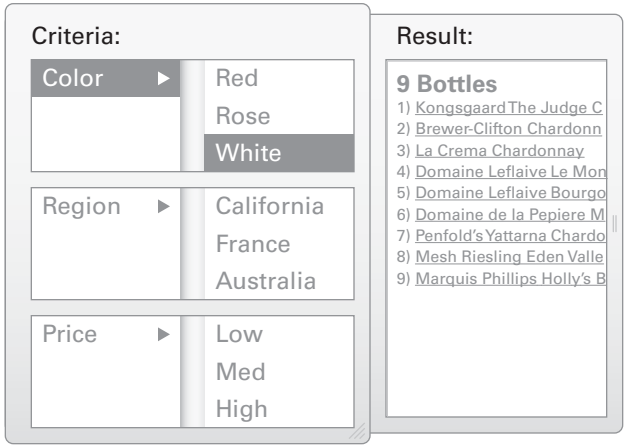
Notice that nothing is selected and the user receives the maximum amount of 27 results (real cases would have many more).



Column List

The user selects the first category of Color, then selects its sub-category of White.

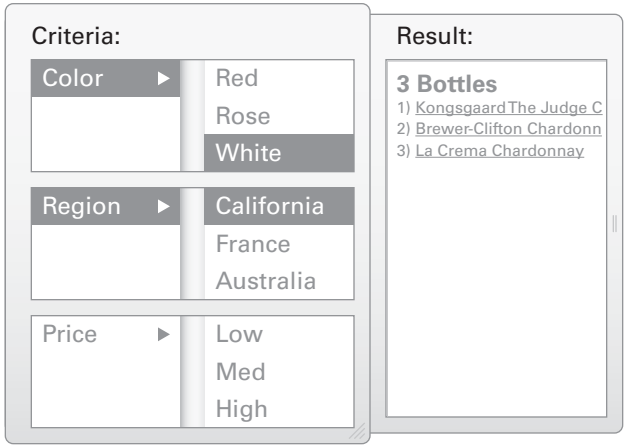
The results returned are narrowed to 9.



Column List

The user selects the second category of Region, then selects its sub-category of California.

The results returned are now narrowed to 3.

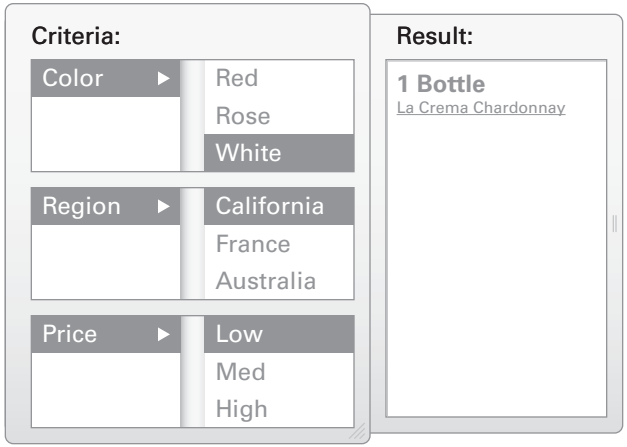


Column List

The user selects the third category of Price, then selects its sub-category of Low.

The results returned are now narrowed to a single bottle.

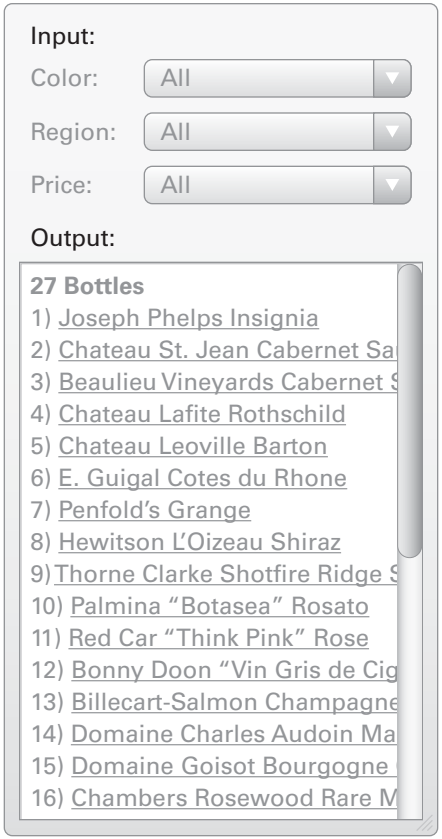
Of course, users could begin with Region or Price.



Pull-down Menus

This interface utilizes pull-down menus to organize our three categories. The user could begin with any pull-down menu.

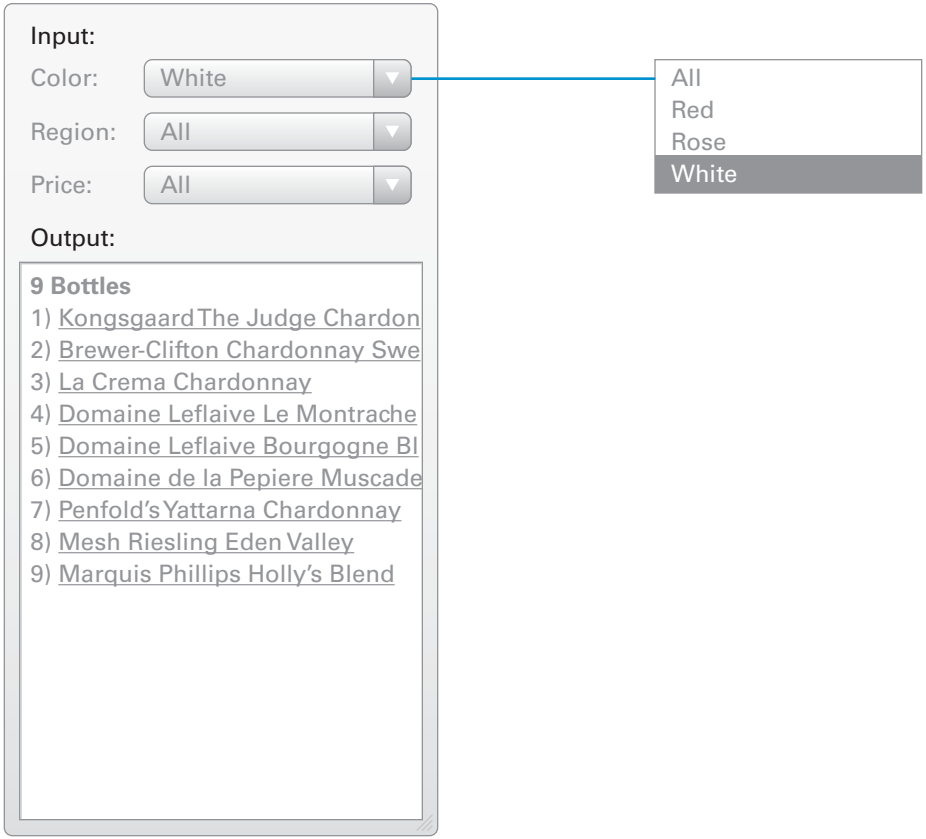
Again, nothing is selected, and the maximum result of 27 bottles is returned.



Pull-down Menus

The user selects the first category of Color, then selects its sub-category of White.

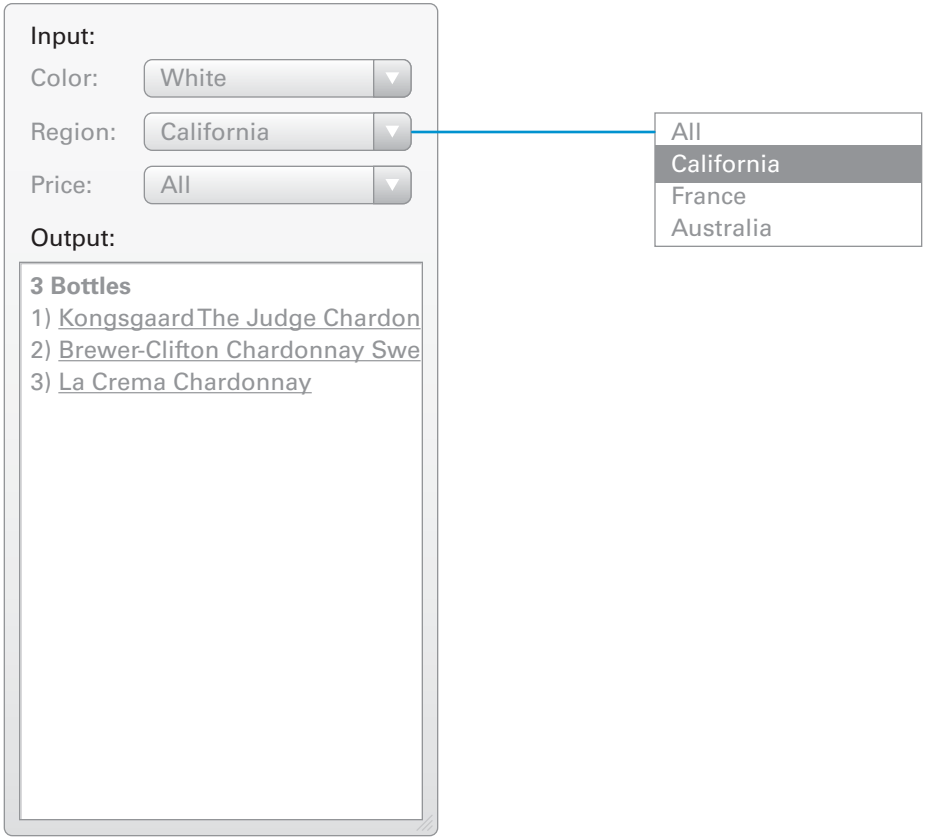
The results returned are narrowed to 9.



Pull-down Menus

The user selects the second category of Region, then selects its sub-category of California.

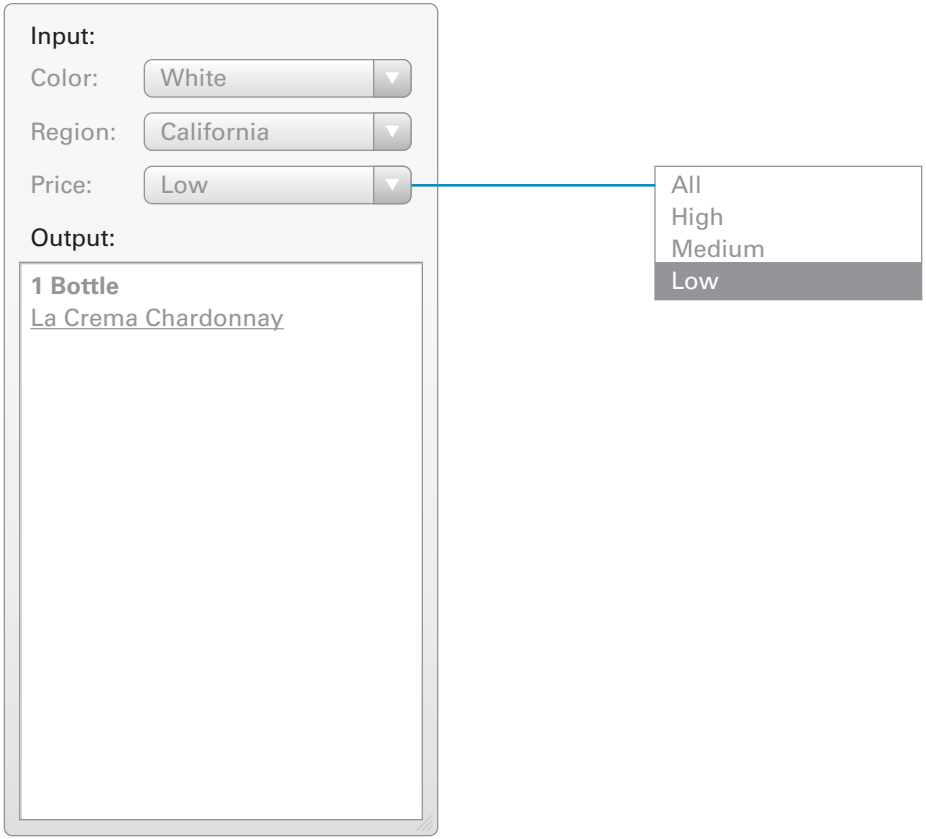
The results returned are now narrowed to 3.



Pull-down Menus

The user selects the third category of Price, then selects its sub-category of Low.

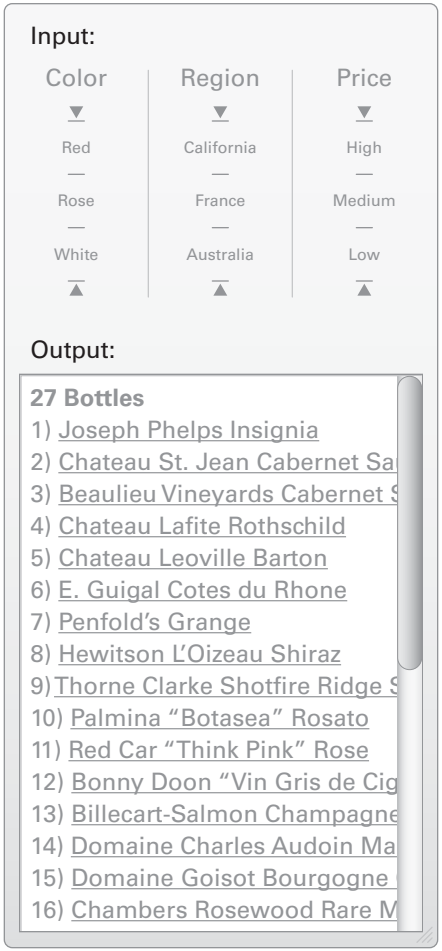
The results returned are now narrowed to a single bottle.



Sliders

This interface utilizes sliders to narrow down the choices of each category. Sliders are important when dealing with a large amount of data because they allow for continuous or nearly continuous ranges, i.e., 0 – 255.

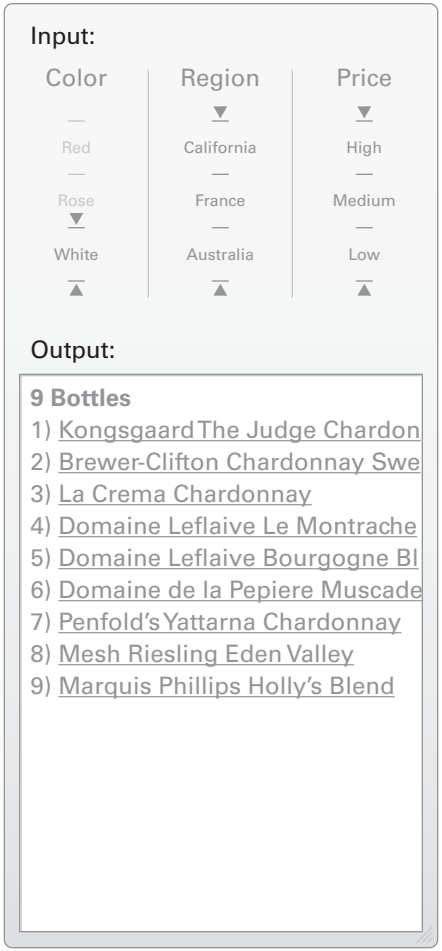
The sliders specify that all sub-categories are being viewed, the maximum result of 27 bottles is returned.



Sliders

The user selects the first category of Color, then moves the slider up to the single sub-category of White.

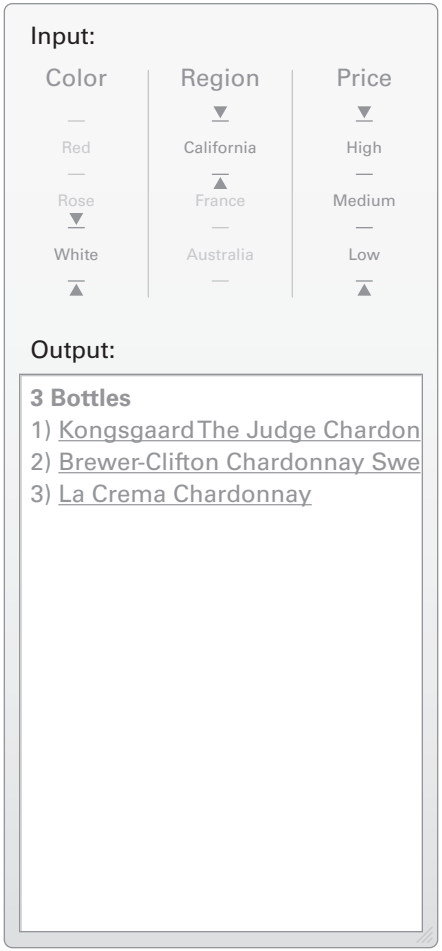
The results returned are narrowed to 9.



Sliders

The user selects the second category of Region, then moves the slider up to the single sub-category of California.

The results returned are now narrowed to 3.

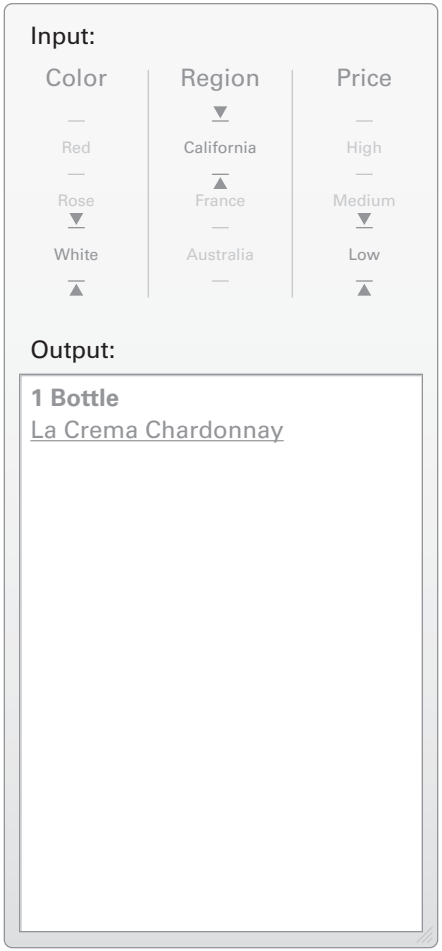


Sliders

The user selects the final category of Price, then moves the slider down to the single sub-category of Low.

The results returned are now narrowed to a single bottle.

One advantage of sliders is that they enable users to quickly expand a category that is already been narrowed.



Checkboxes

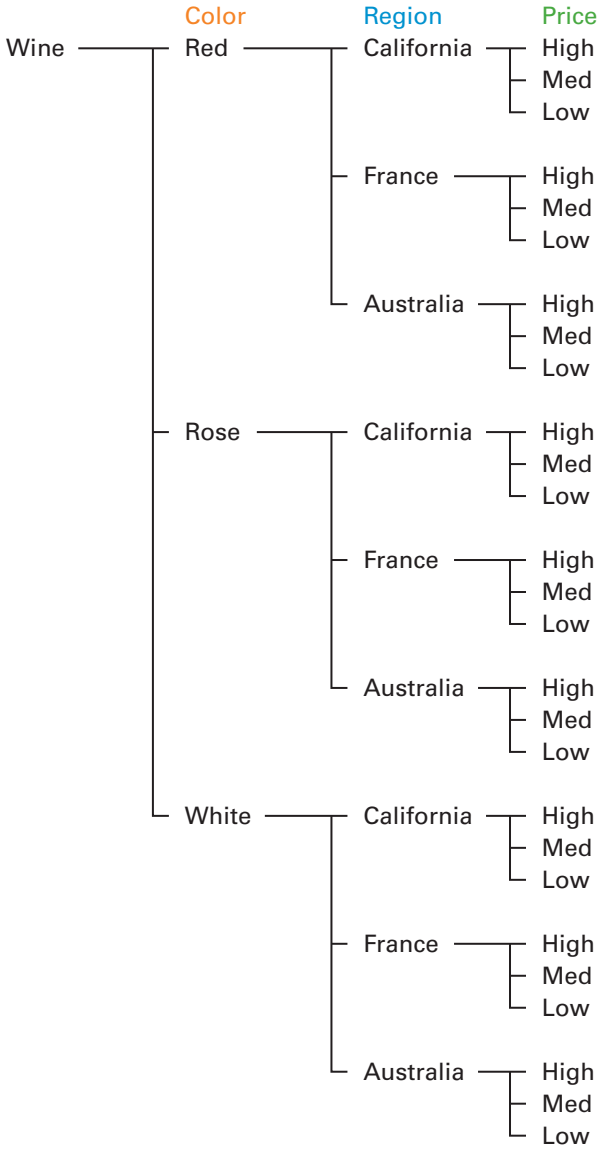
This interface reveals the sub-categories of our three categories at first look. Another strength of the checkboxes is that the user is able to choose whichever box they want, in any order they want to. This can also be done in the column list and pull-down menus, but is less intuitive as they are placed on top of one another, whereas here, the checkboxes are not as hierarchical.

We will utilize this checkbox interface in the following flow example.

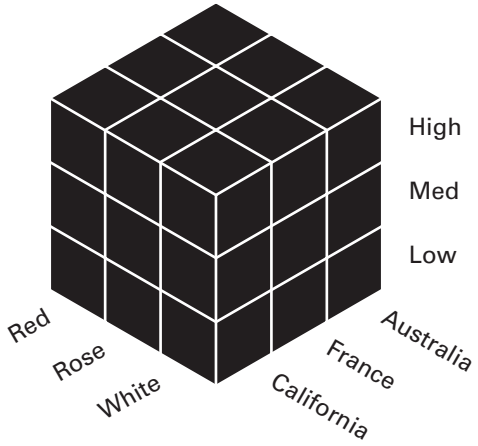
Flow Example

All Wine

Tree



Cube



Interface

Criteria:

☑ Red	☑ California	☑ High
☑ Rose	☑ France	☑ Med
☑ White	☑ Australia	☑ Low

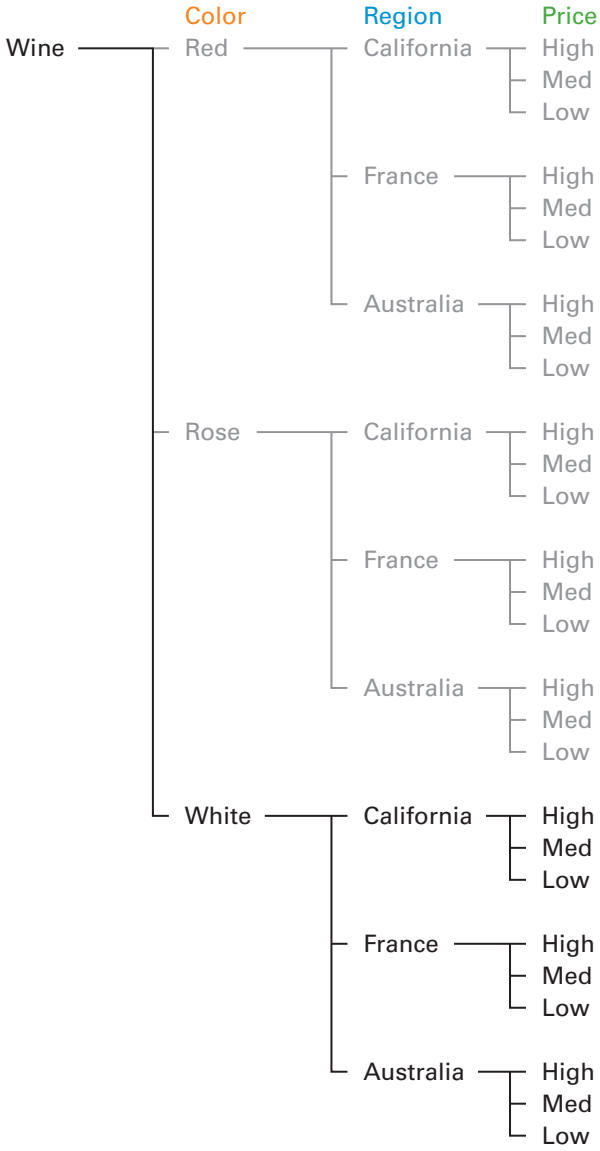
Result:

27 Bottles

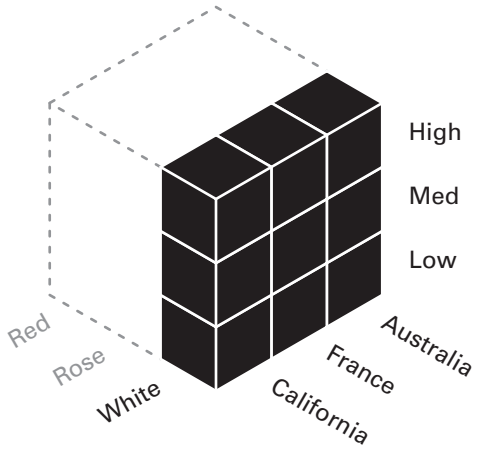
1) Joseph Phelps Insignia
2) Chateau St. Jean Cabernet Sauvignon Sonoma
3) Beaulieu Vineyards Cabernet Sauvignon
4) Chateau Lafite Rothschild
5) Chateau Leoville Barton
6) E. Guigal Cotes du Rhone
7) Penfold's Grange
8) Hewitson L'Oizeau Shiraz
9) Thorne Clarke Shotfire Ridge Shiraz
10) Palmina "Botasea" Rosato

Color

Tree



Cube



Interface

Criteria:

☐ Red	☑ California	☑ High
☐ Rose	☑ France	☑ Med
☑ White	☑ Australia	☑ Low

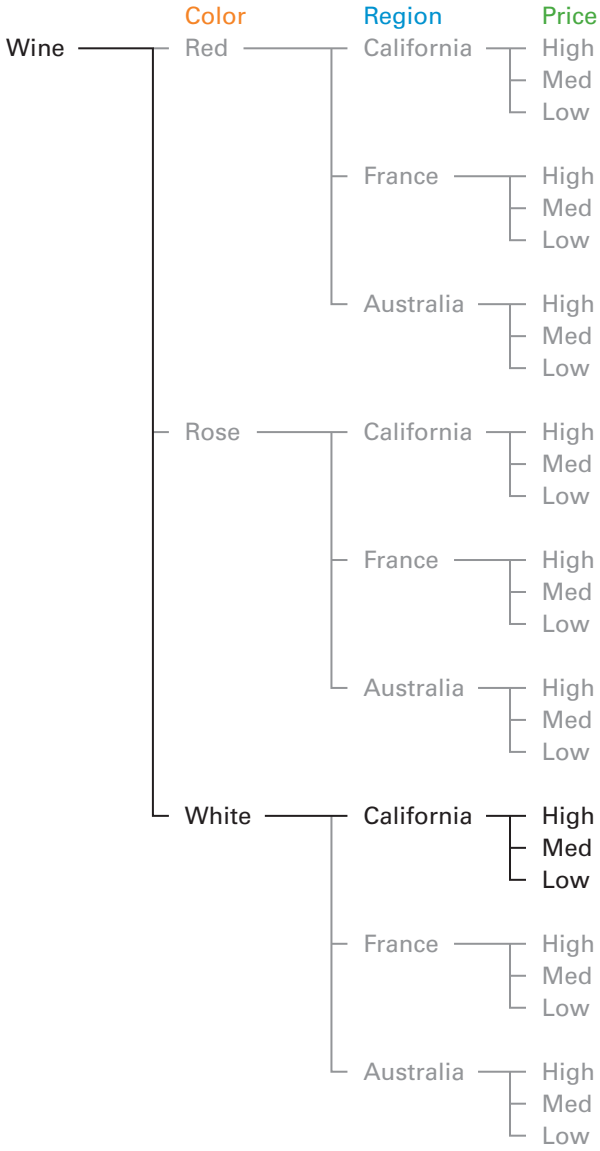
Result:

9 Bottles

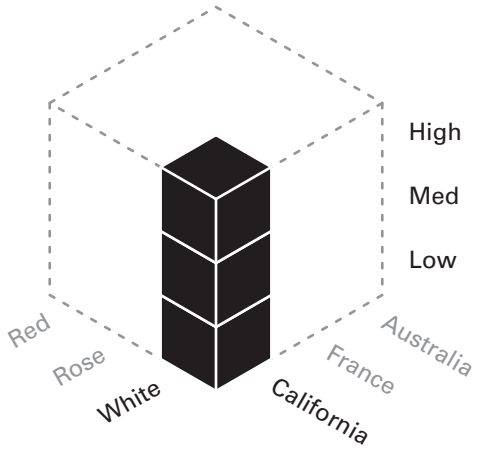
1) Kongsgaard The Judge Chardonnay
2) Brewer-Clifton Chardonnay Sweeney Canyon
3) La Crema Chardonnay
4) Domaine Leflaive Le Montrachet
5) Domaine Leflaive Bourgogne Blanc
6) Domaine de la Pepiere Muscadet Sevre et Maine
7) Penfold's Yattarna Chardonnay
8) Mesh Riesling Eden Valley
9) Marquis Phillips Holly's Blend

Color/Region

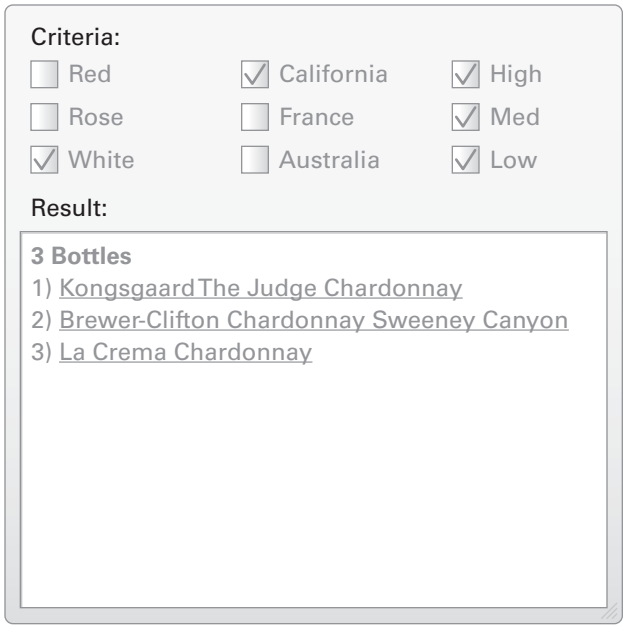
Tree



Cube

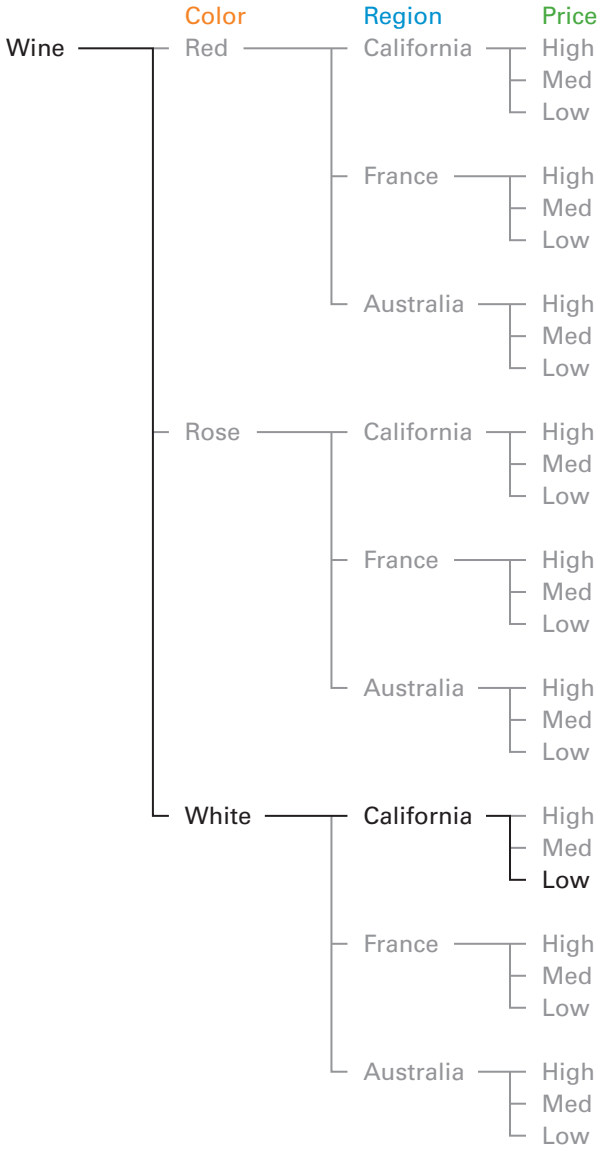


Interface

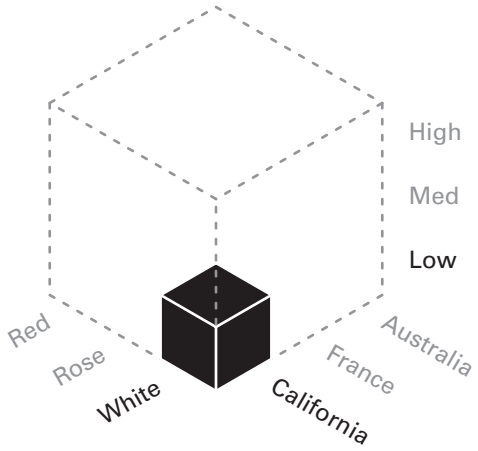


Color/Region/Price

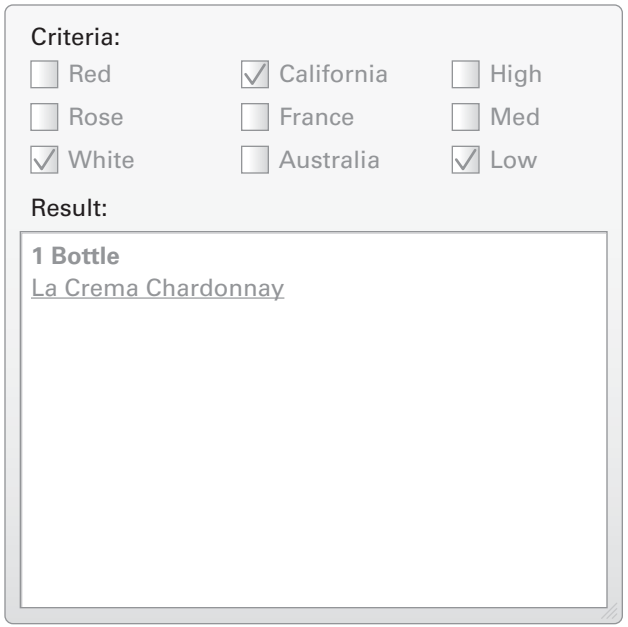
Tree



Cube



Interface



Pivoting

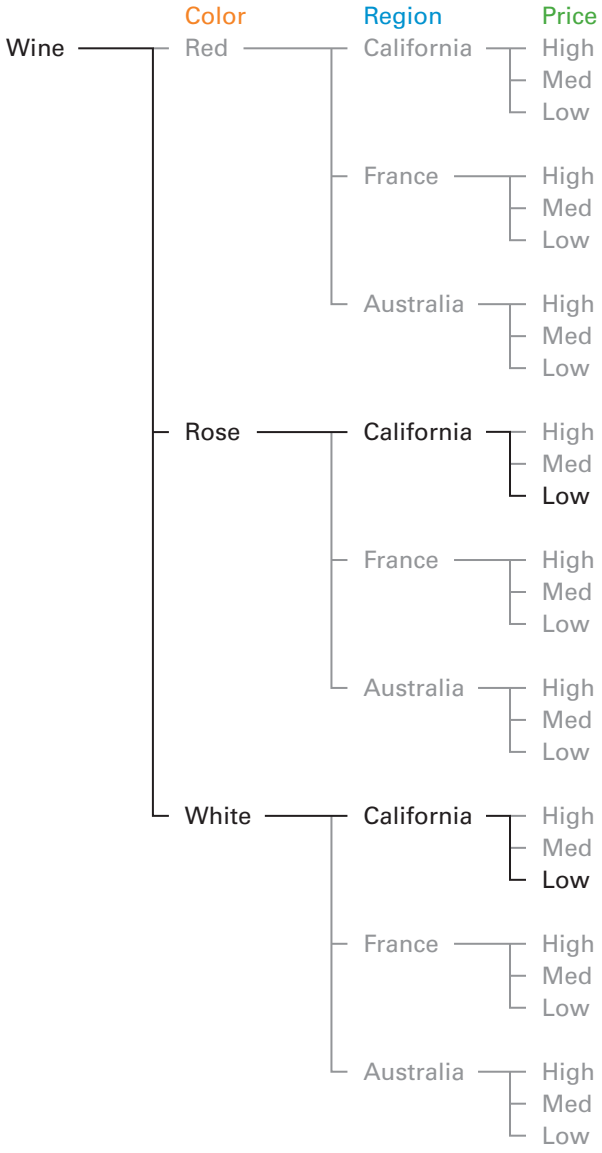
Here's where things get really interesting.

The user has narrowed down to White, California, Low. But suppose she doesn't like the results or she wants to explore other options. In a tree structure she would have to back out to the root and then travel back down the tree. Finding all the options for low cost would require an awful lot of climbing up and down the tree. The result would be a frustrated user.

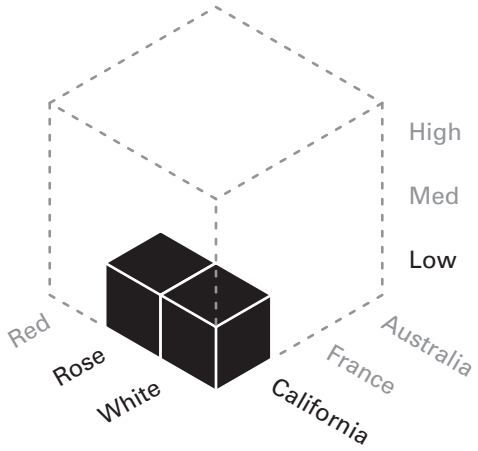
The navigation solution involves two things. First, conceiving the data as an N-dimensional matrix and second recognizing that users may narrow along one path and then turn and "pivot" to expand back up another path—before narrowing again.

California/Low/More Colors

Tree



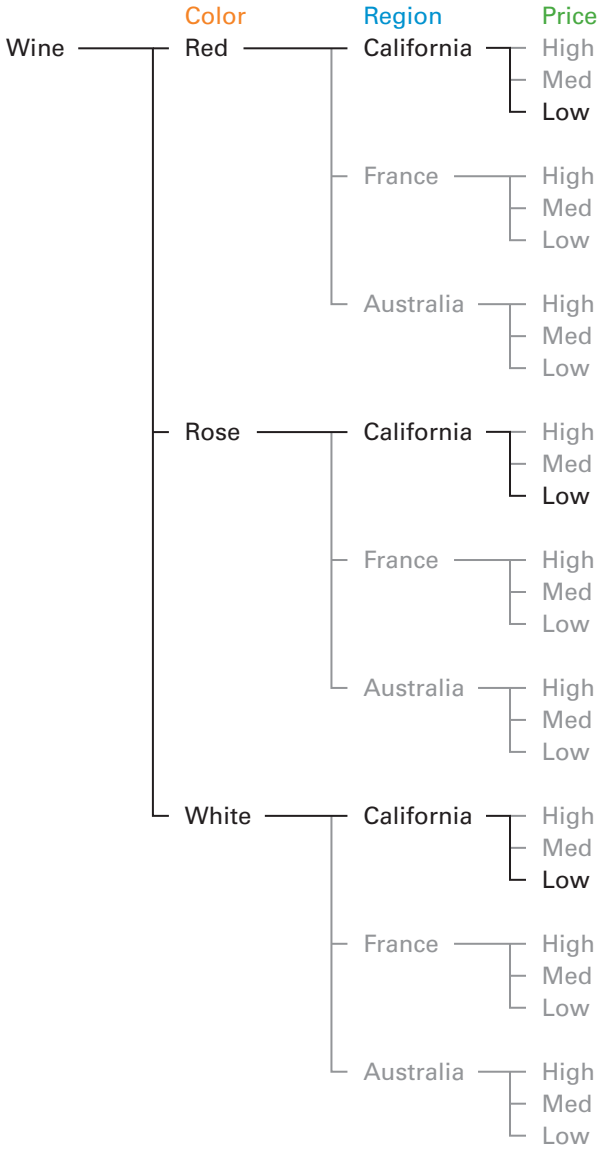
Cube



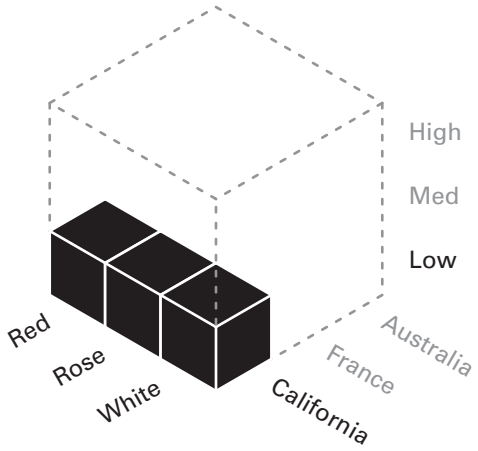
Interface

California/Low/All Colors

Tree



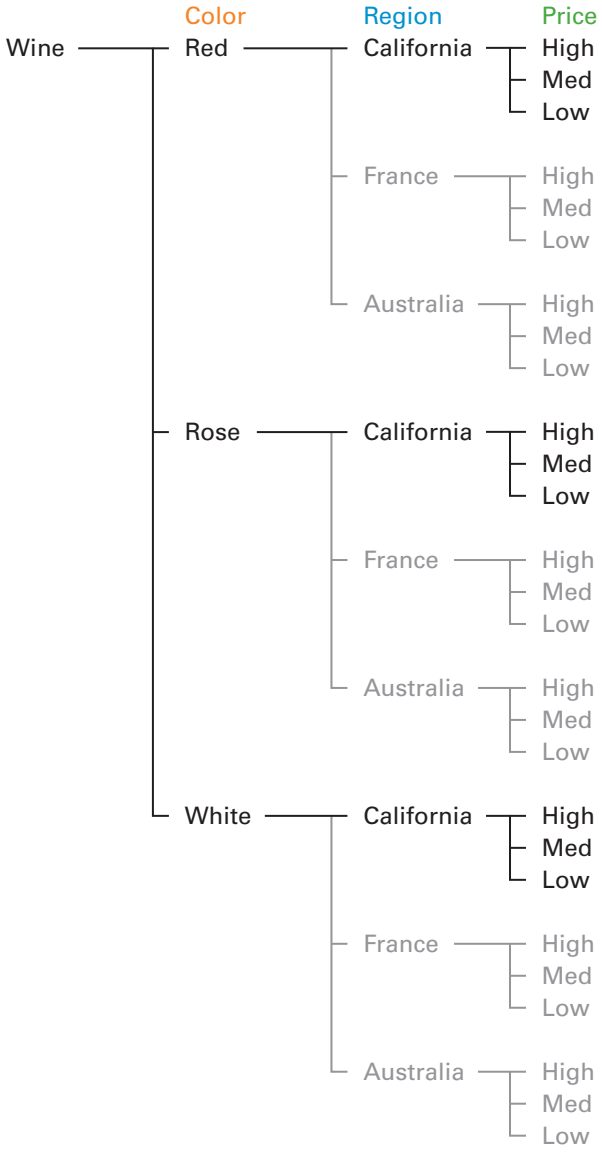
Cube



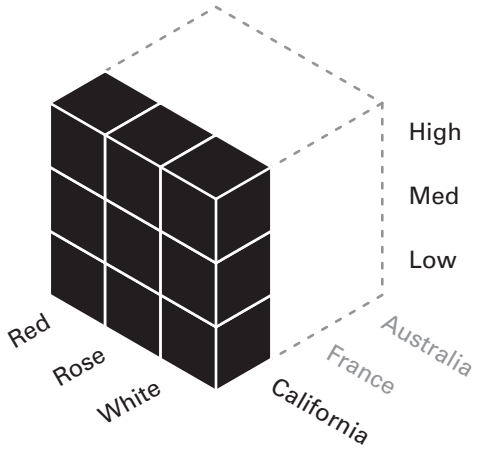
Interface

California/All Colors/All Prices

Tree



Cube



Interface

Criteria:

☑ Red	☑ California	☑ High
☑ Rose	☐ France	☑ Med
☑ White	☐ Australia	☑ Low

Result:

9 Bottles

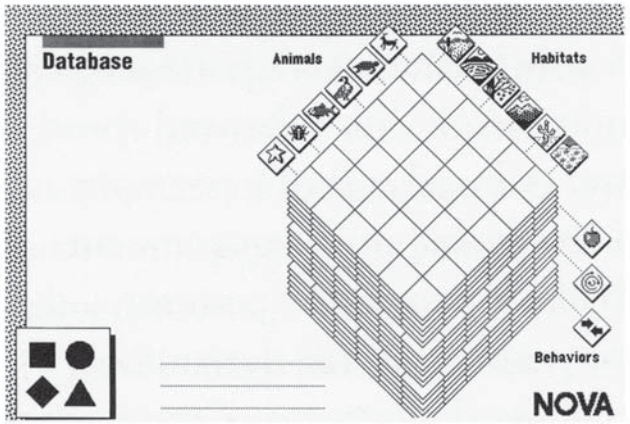
1) La Crema Chardonnay
2) Bonny Doon "Vin Gris de Cigare"
3) Beaulieu Vineyards Cabernet Sauvignon
4) Brewer-Clifton Chardonnay Sweeney Canyon
5) Red Car "Think Pink" Rose
6) Chateau St. Jean Cabernet Sauvignon Sonoma
7) Kongsgaard The Judge Chardonnay
8) Palmina "Botasea" Rosato
9) Joseph Phelps Insignia

Practical Applications

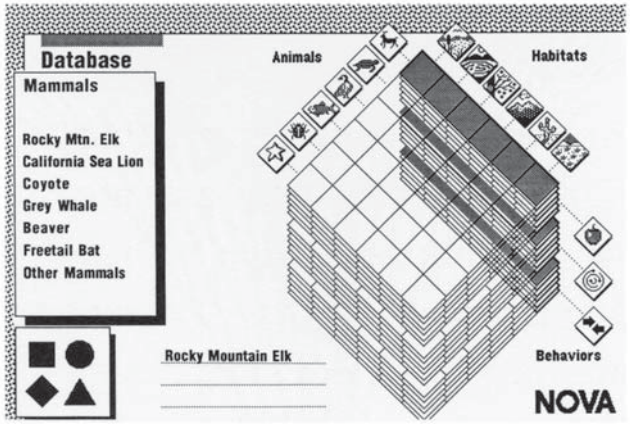
Interactive Nova

In 1988, Paul Souza designed this interface to a database of animal information, their habitats, and behavior. Clicking the icon for mammals displays a list of sub-categories. Clicking on the icon for mountains displays a list of sub-categories. Clicking on the icon for sensing narrows the options further. After all the parameters have been set, clicking the 'Go To' diamond takes the user to the desired screen.

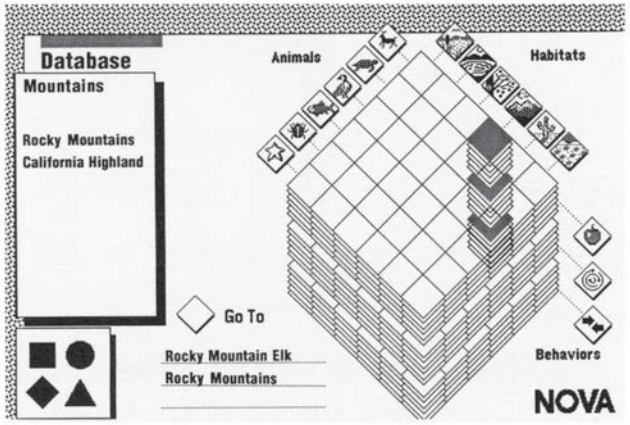
1. All Dimensions



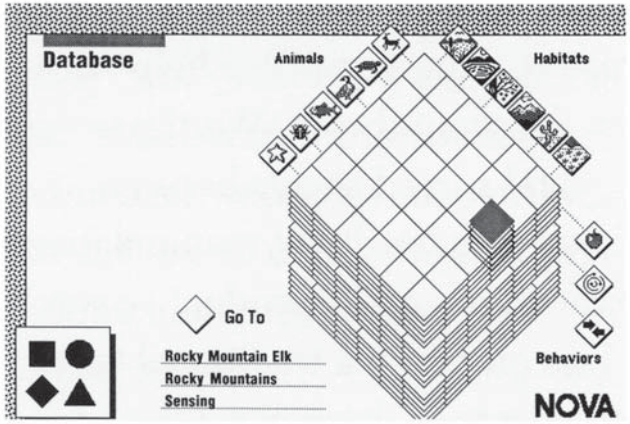
2. Animal



3. Animal/Habitat

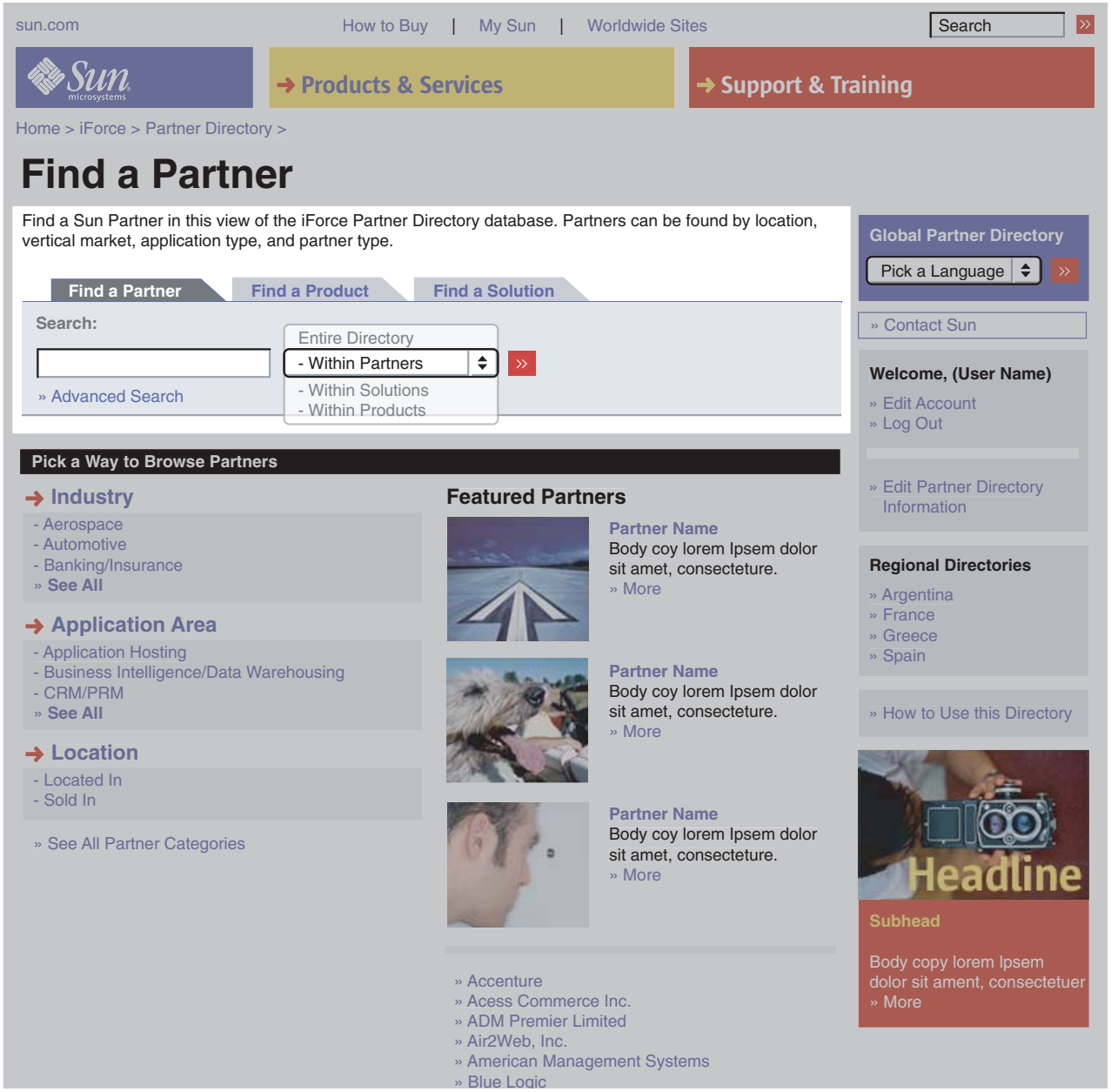


4. Animal/Habitat/Behavior



Sun Microsystems

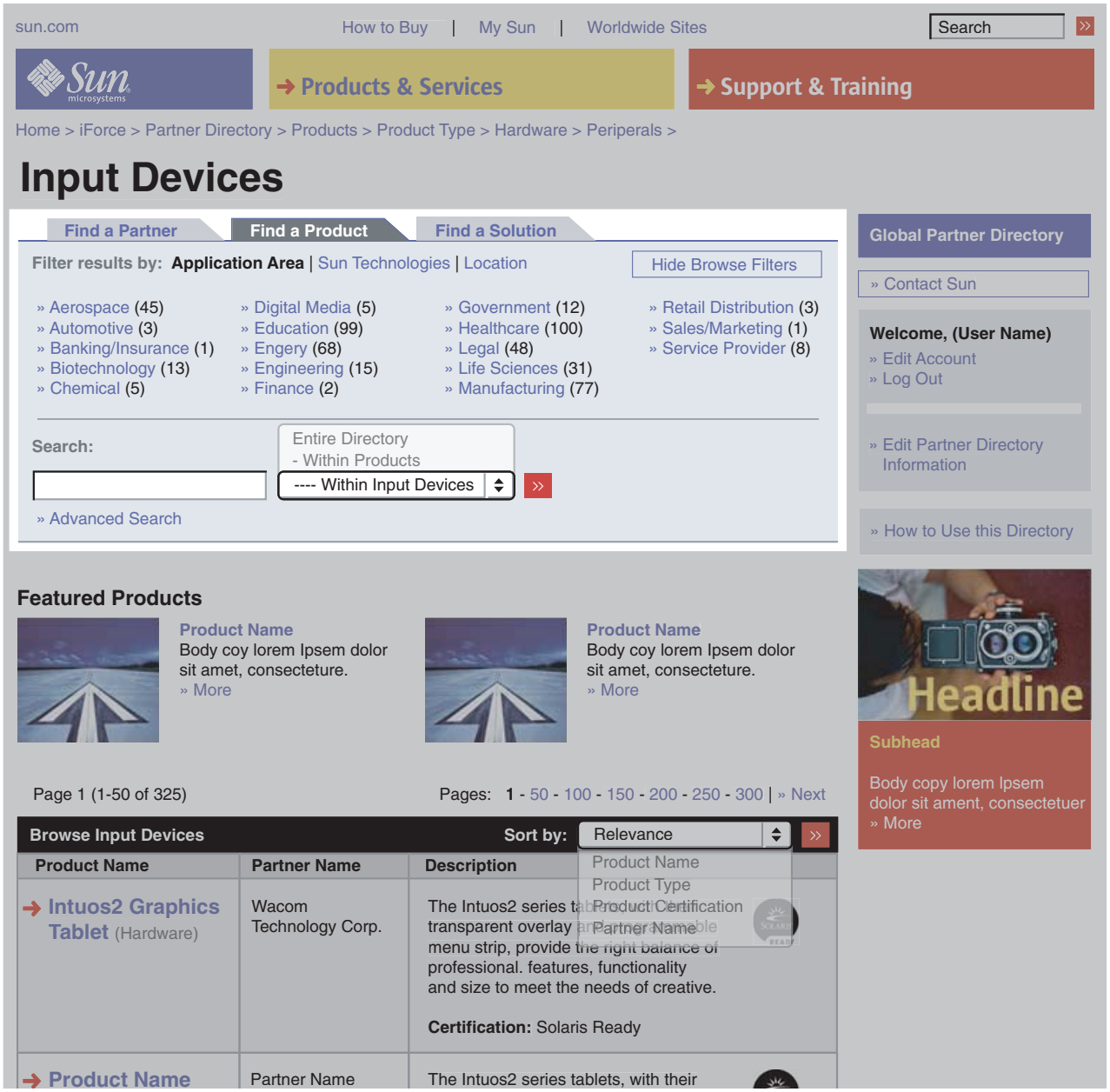
In 2003, Greg Baker designed a search interface for Sun Microsystems. It is organized by three main tabs: Find a Partner, Find a Product, and Find a Solution. Each tab contains a different set of information, what is unique is that the interface provides three ways to browse the same set of information. The following pages show an example of performing a search and filtering the results with different filters.



Sun Microsystems

The user switches from the Find a Partner tab to the Find a Product tab. Notice the addition of the three browse filters mentioned on the previous page: Filter results by: Application Area, Sun Technologies, or Location.

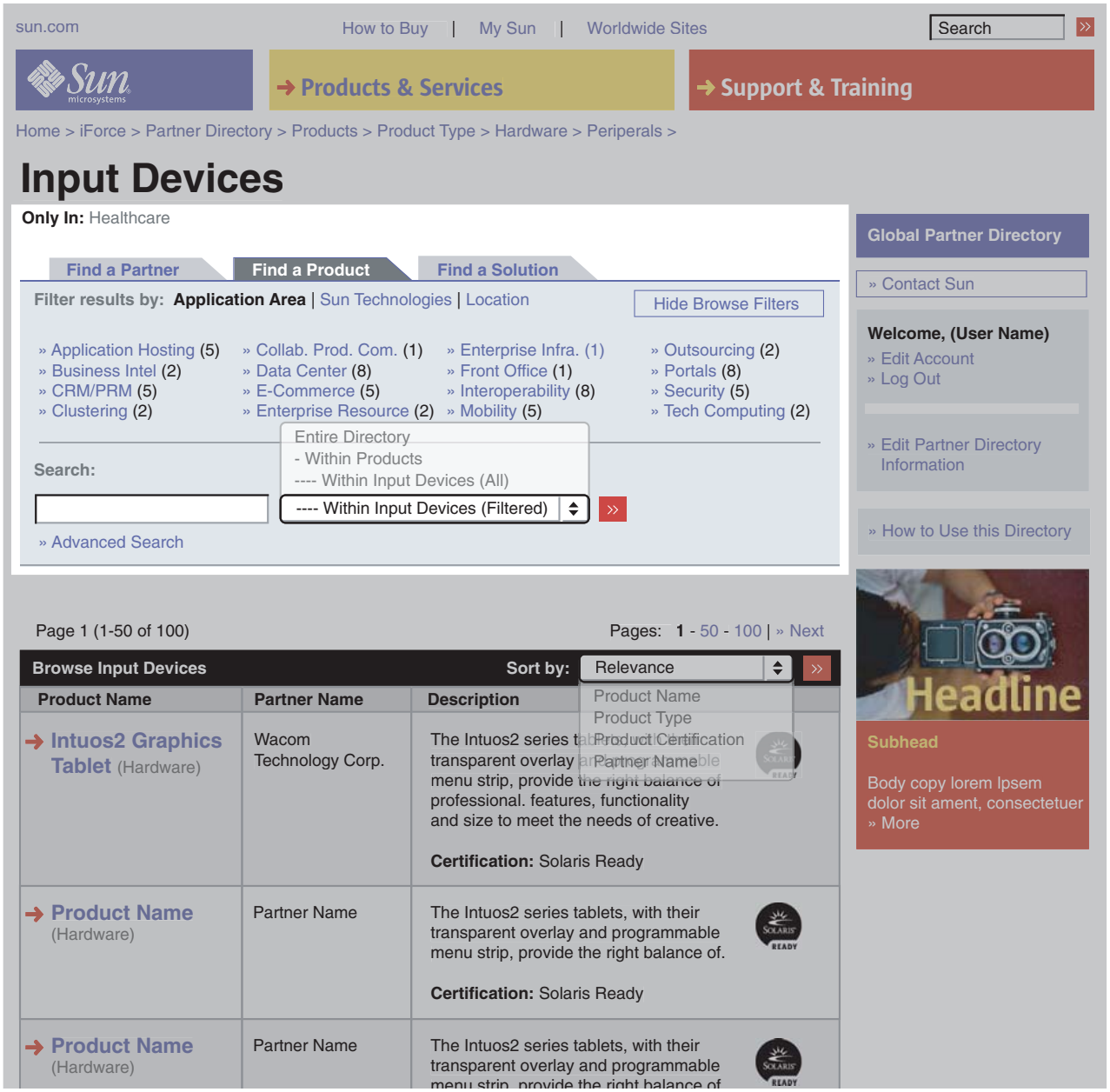
The user chooses to filter their results by Application Area, and then selects Healthcare.



Sun Microsystems

After the user selects Healthcare, the browse filters change out depending on the current choice and provide a new set of options.

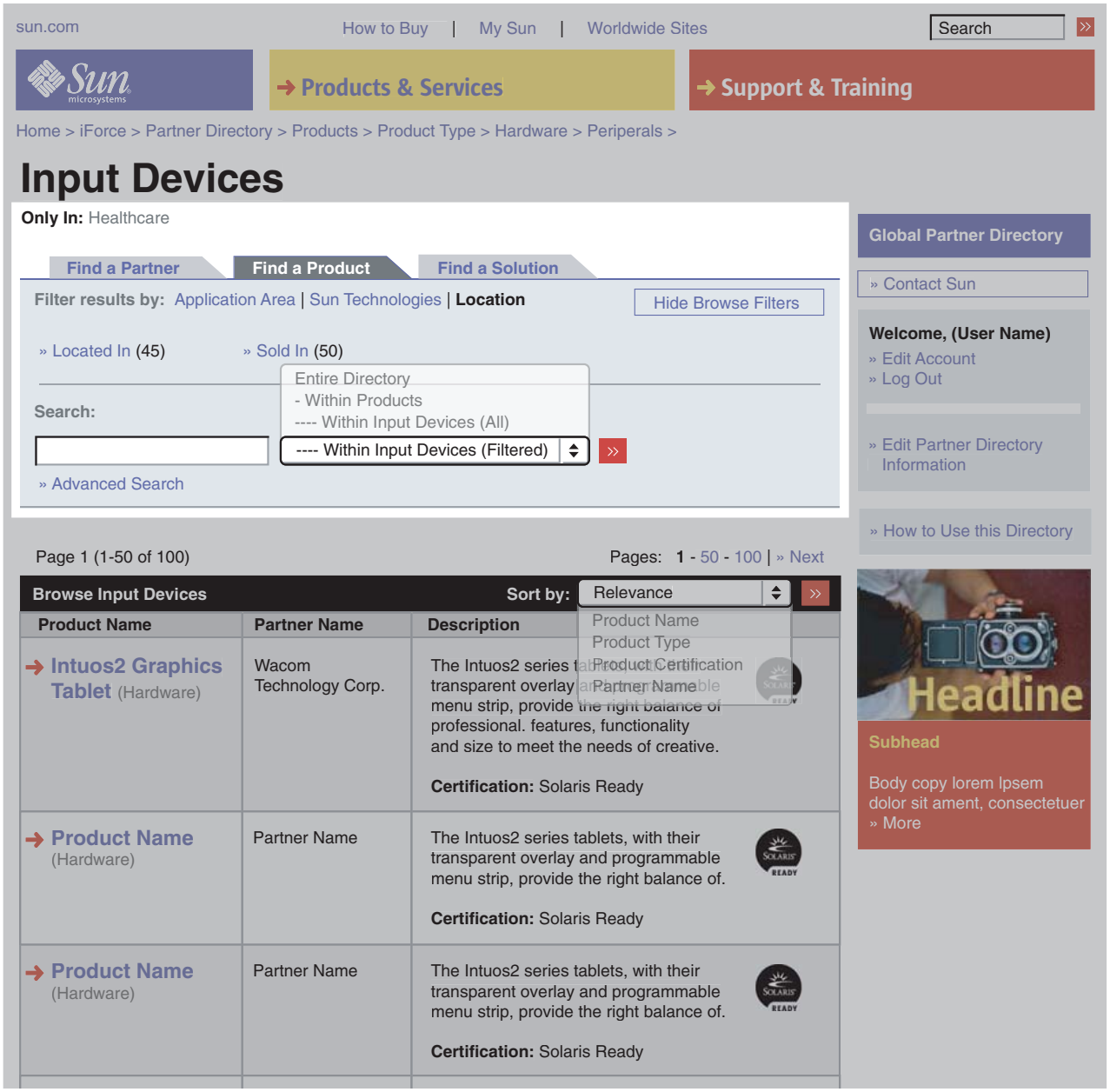
The user now decides to “pivot” and filter their results by Location instead of Application Area.



Sun Microsystems

Now that the user has switched to filter by Location, the browse filters change out again and provide a new set of options. The user selects Sold In.

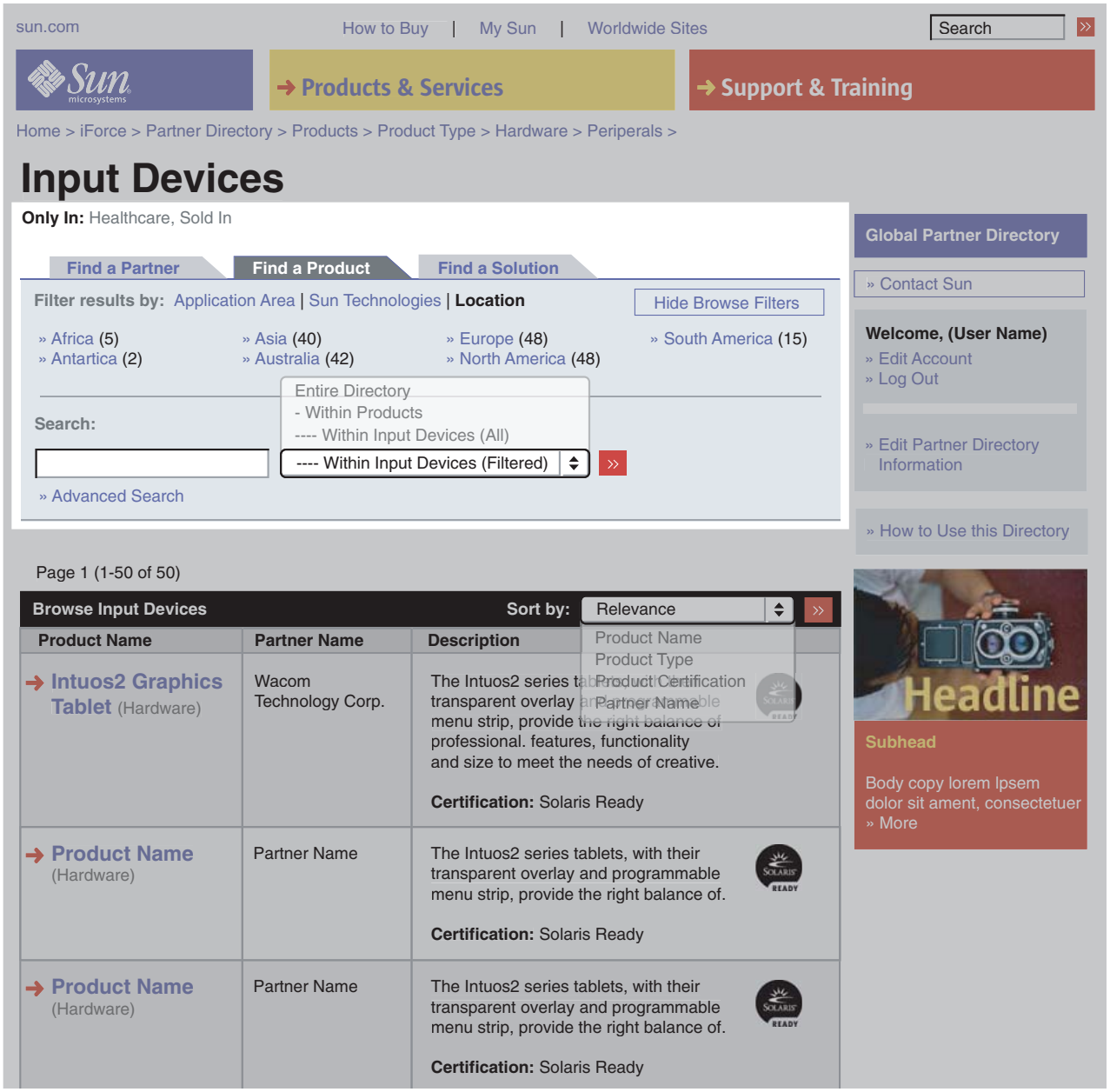
Notice that the breadcrumb has not changed, effectively providing a point of reference for the user.



Sun Microsystems

The browse filters continue to change out and provide a new set of options. This process continues until the user reaches a “leaf” results page.

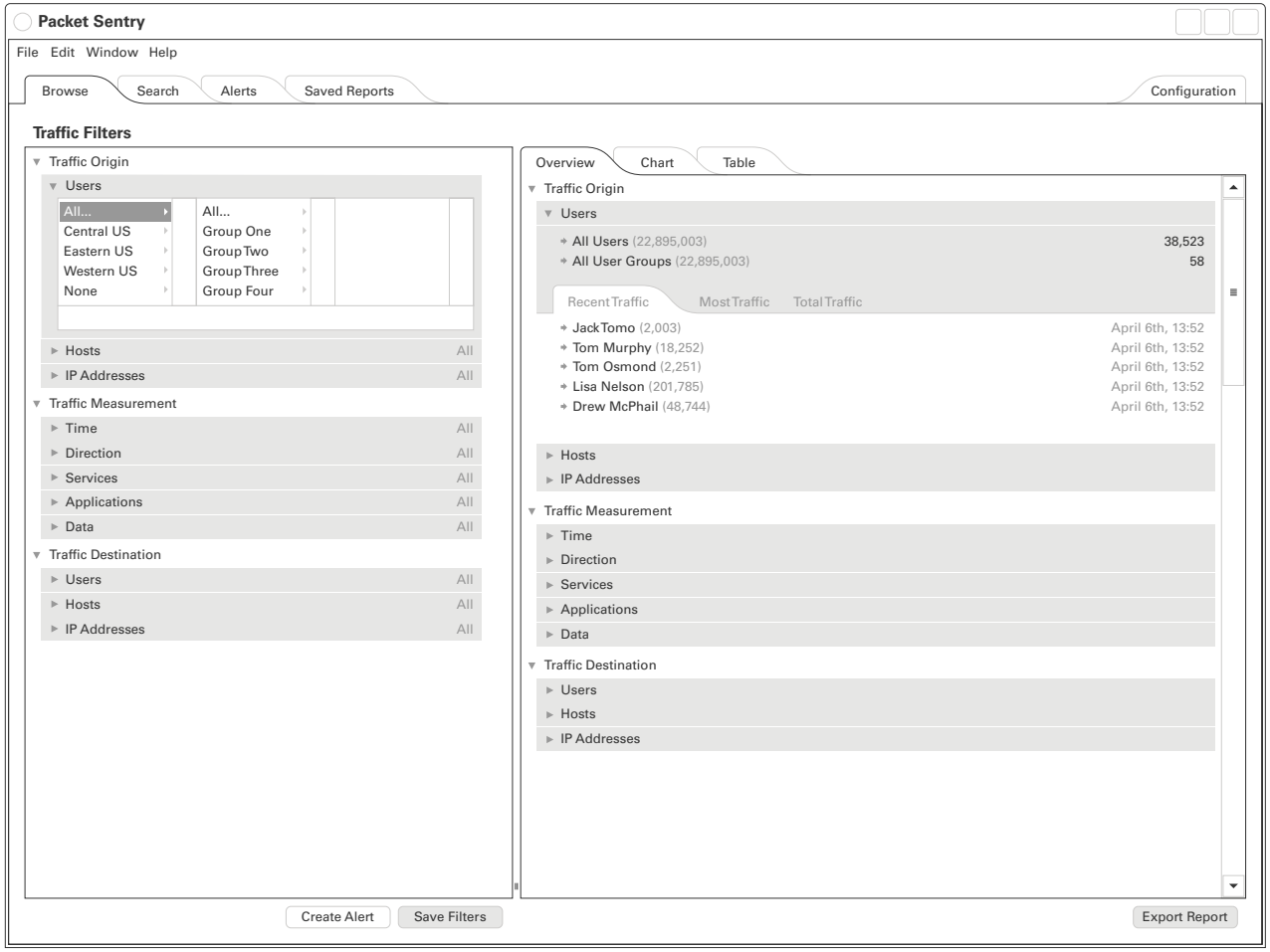
Notice that the breadcrumb grows, and will continue to grow as the search continues.



PacketMotion

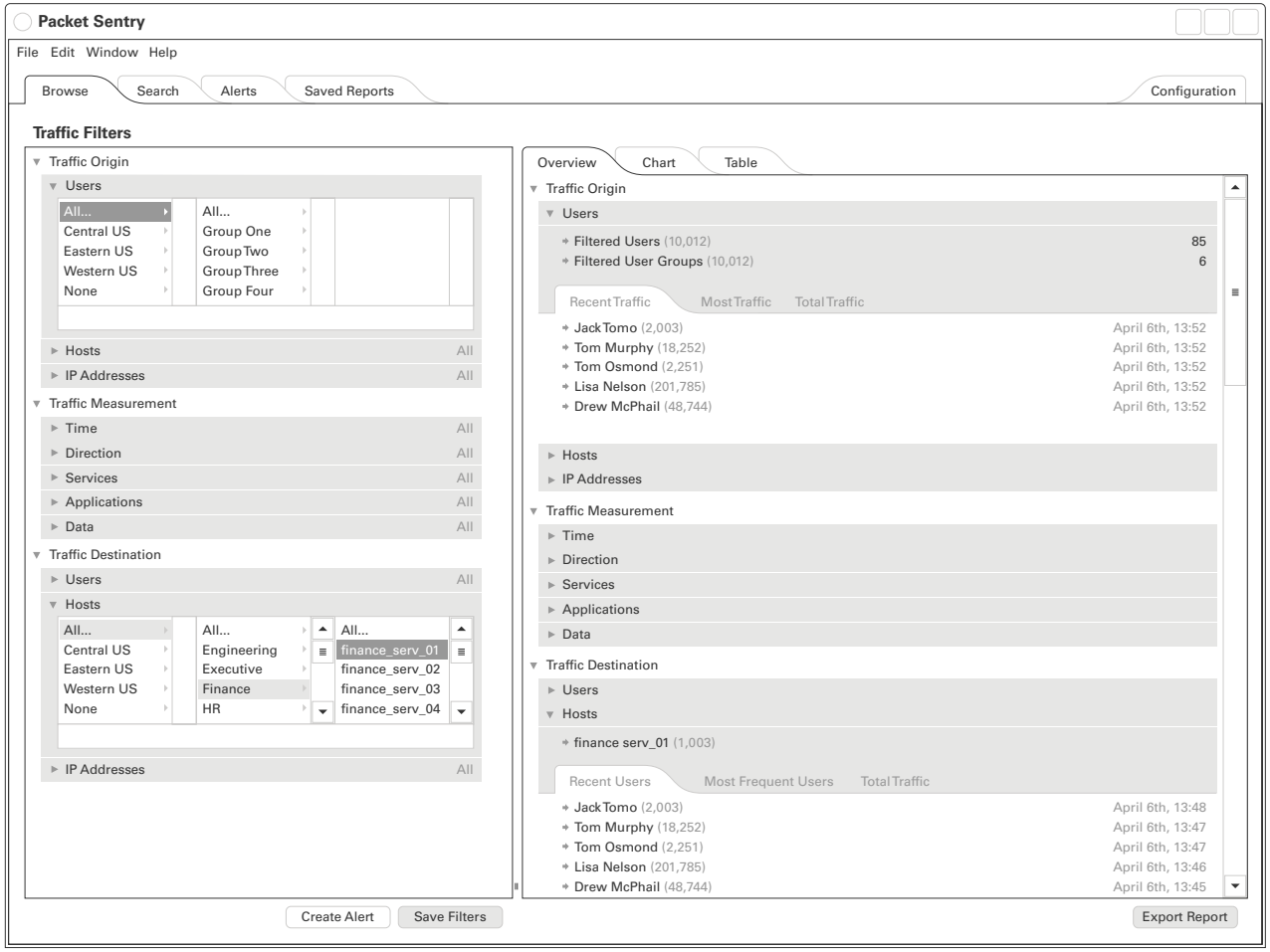
In 2006, Greg Baker designed an interface that for PacketMotion, a company which provides multi-dimensional visibility into network activity to allow for better control and security over an enterprise. The interface covers 11 dimensions, which are grouped under 3 sections: Traffic Origin, Traffic Measurement, and Traffic Destination.

First, the user selects the Traffic Origin section, then the Users dimension, and stops at All users. The results section, on the right, returns more detailed information.



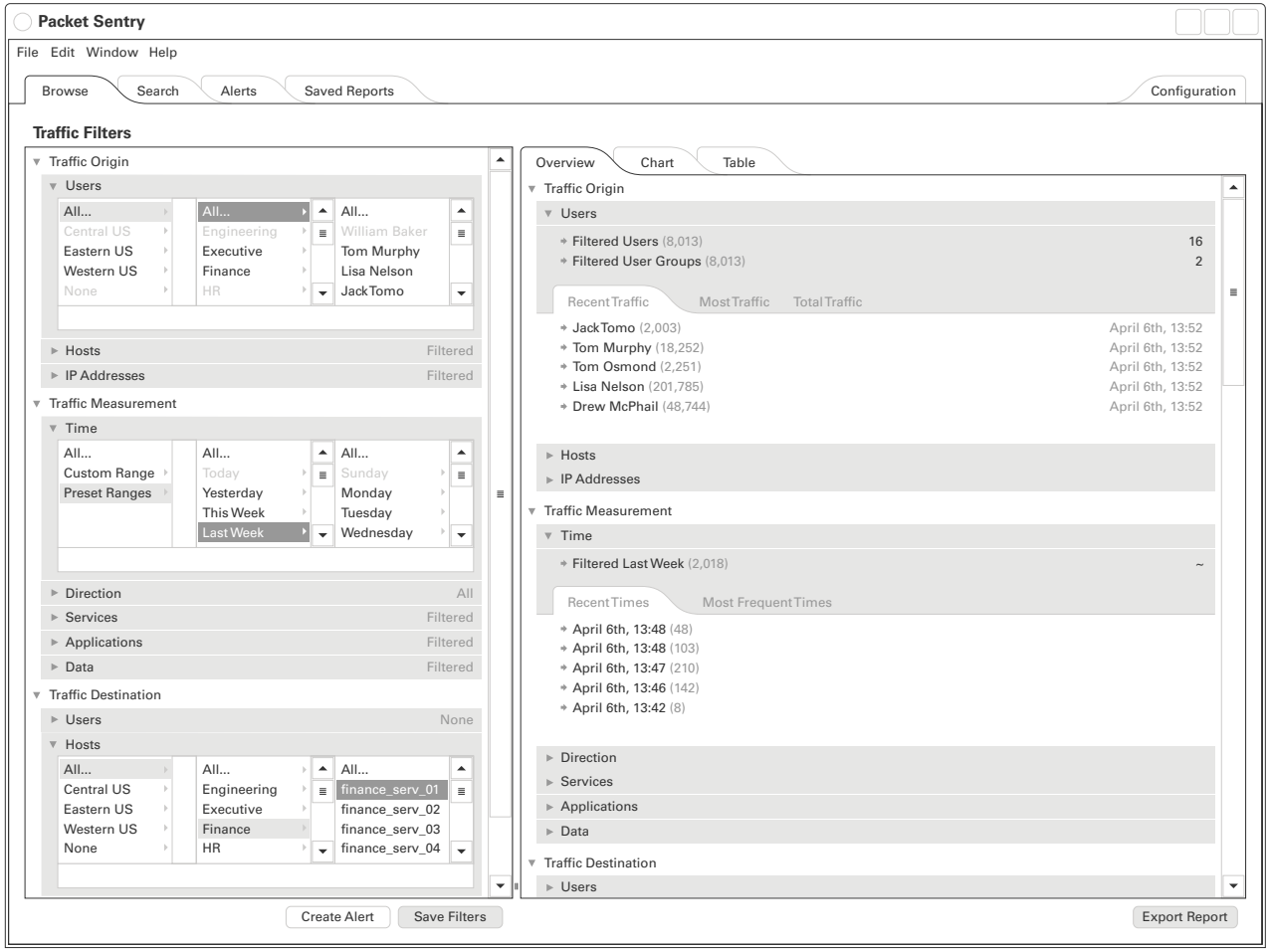
PacketMotion

Next, the user selects the Traffic Destination section, then the Host dimension, and continues refining their input down to a specific host machine. Again, The overview section updates to reveal more details.



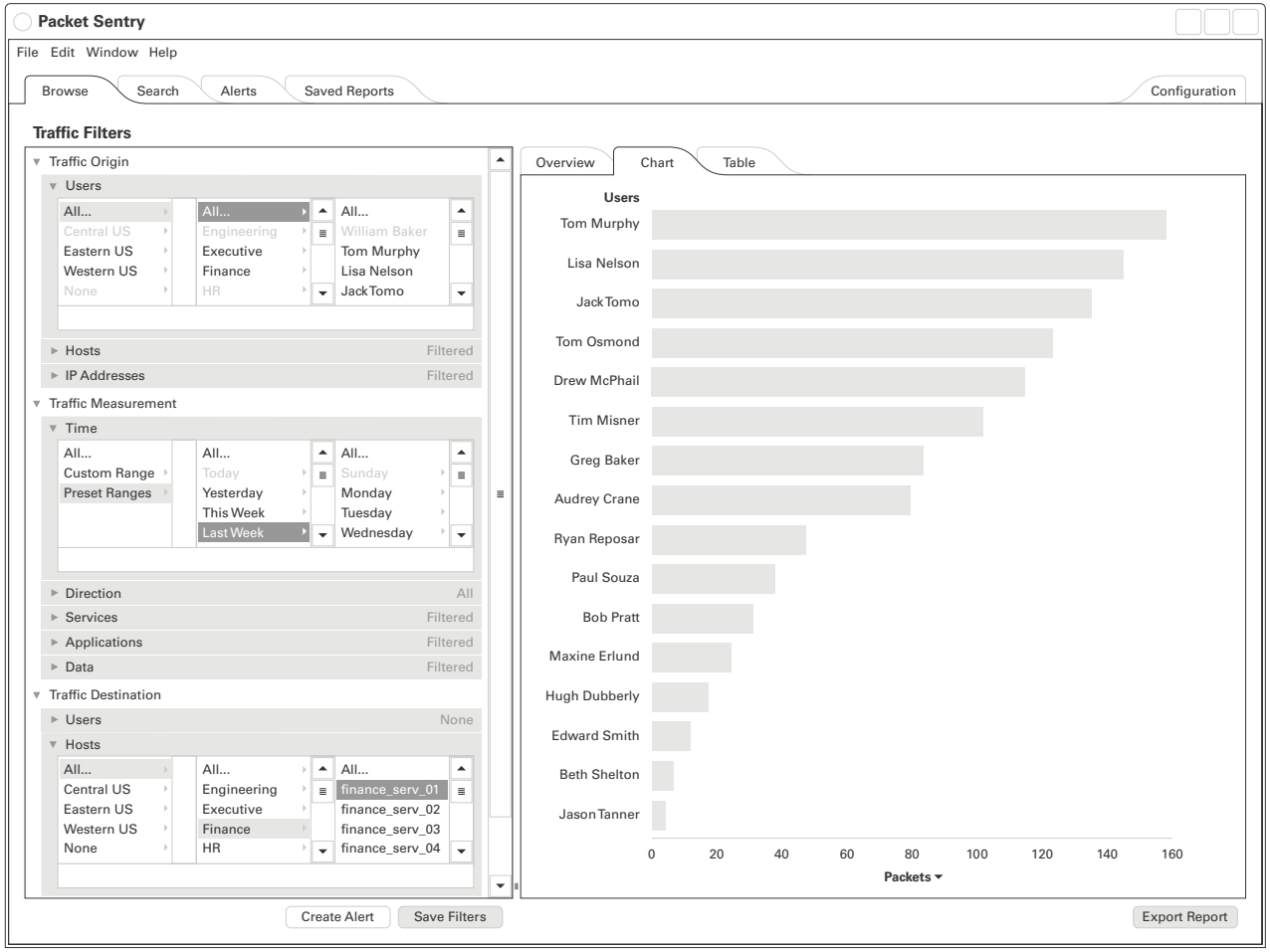
PacketMotion

Then, the user selects the last section of Traffic Measurement, then the time dimension, and specifies Last Week. Notice that the total results returned (in the top right of the users overview) have been narrowed down from 38,523 to 16.



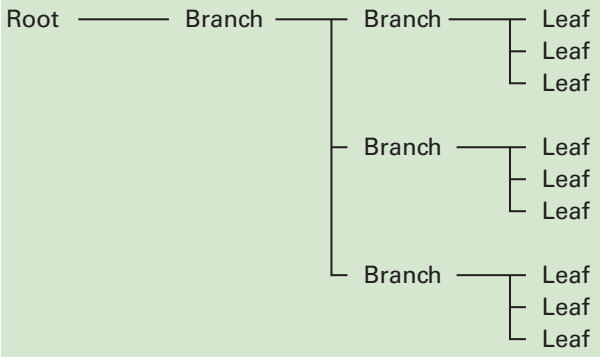
PacketMotion

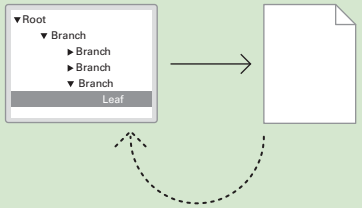
Finally, with all the traffic filters specified, the user switches from the overview tab to the chart tab to see a different view of their desired information.



Summary of Navigation Structures

Browse a Tree





Pros:
- Direct control
- May show full data set
- User sees scope and location

Cons:
- Doesn't scale
- Becomes unwieldy over ~400 items

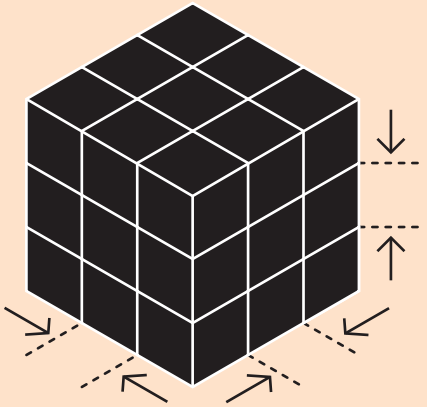
Search a Database

Can be like looking at a database through a keyhole.

Pros:
- Works better for large data sets
- Great if you know the data set
- Can be fast, direct

Cons:
- Can't see full scope of data
- May swamp users with too many results

Filter a Database



Pros:
- Allows exploration of large data sets
- Can be combined with browse and search

Cons:
- Less familiar to many users
- Requires highly structured data