cbr:works

Getting Started

Revision 2.1, May 2000

This document is subject to change without notice

Disclaimer:

THIS DOCUMENT IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.

The information contained in this document represents the current view of tec:inno on the issues discussed as of the date of publication.

INFORMATION PROVIDED IN THIS DOCUMENT IS PROVIDED 'AS IS' WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND FREEDOM FROM INFRINGEMENT.

tec:inno GmbH

Sauerwiesen 2
D-67661 Kaiserslautern
Tel: +49 (0) 6301 606 400
Fax: +49 (0) 6301 606 409

cbrworks@tecinno.com
support@tecinno.com
www.tecinno.com

Content

Content i

1 Introduction 1

2 Used cars Application Model 2

2.1 Our first Application Scenario 2
2.2 Starting CBR- Works 3
2.3 Terminology 5
2.4 Basics on Using CBR-Works 7
How to find the saved application model 21
2.5 Working with the Case Base 23
2.6 Consulting the Case Base 32
2.7 Advanced Queries 36
2.8 Query Wizard 40
2.9 Summary 43

3 Enhanced Similarity Model 44

3.1 The Color Problem 44
3.2 Taxonomy Modeling 45
3.3 Detailing and Similarity 46
3.4 Summary 50

4 Enhancing Usability & Online Interface 51

4.1 The Apartments Demo 51
4.2 Questions 55
4.3 Language Settings 57
4.4 Annotations 60
4.5 Question Ordering 62
4.6 Using the Application Online through the Web 65
4.7 Summary 73

5 Automatic Calculation with Rules 75
5.1 The Vacation Demo 75
5.2 Different Kinds of Rules 77
5.3 Defining Rules 78
5.4 Controlling the Execution of Rules 82
5.5 Summary 83

6 Some Remarks on Application Development 84
6.1 Identify Knowledge 85
6.2 Define Domain Model 85
6.3 Case Import 86
6.4 Build GUI 86

7 From Data Bases to intelligent Web Applications 87
7.1 Identify Database 88
7.2 Describe Model 93
Creating Concepts from an ODBC Source 93
Defining Types from the Data Source 97
7.3 Import Cases 101
7.4 Create Application 104
7.5 Conclusion 108

8 Concluding Remarks 109

1 Introduction

The following Tutorial pages will give you a first guide on how to utilize the CBR-Works tools to produce your own CBR applications. After you have finished these small exercises you will be able to operate the basic functions. The road map is as follows:

- We will start by building a very simple application to explore the main windows and functions of CBR-Works.
- This simple application will be extended to introduce the different sorts of concepts and types you can use to model an application domain.
- You will learn to adapt some of the demo applications for your needs.
- Then, we will show you some more sophisticated modeling techniques like the employment of domain specific similarity measures.
- Based on the demo applications that come with CBR-Works, we will look at some key features for usability and automated calculation.

2 Used cars Application Model

In this chapter we will provide you an overview and the basic concepts of CBR-Works. You will also learn the terminology used throughout this tutorial.

2.1 Our first Application Scenario

As an application scenario we will use the problem of selecting a used car from a dealer database or a newspaper.

Suppose you want to produce a CBR application that eases the search for a used car in the private advertising area of a newspaper. Usually, such advertisements describe a car by a number of features, like age, mileage, color, body etc.

In the example given in figure 2-1 you see an advertisement for an Opel/GM Vectra. The description of the car tells that it has a catalytic converter, it is colored red, it has a sunroof, aluminum wheels, a hitch and a power steering. The car is offered for 5,900 EUR and it has a mileage of 70,000 km.

Vectra, 1.5 GL, Cat., red, hitch, power steering, sunroof, LM-tires, 70,000 km, 5,900 EUR, Touring-Garage Enterprises. Phon [redacted]44

Figure 2-1: A Car Advertisement

From this you will easily come up with an initial set of features to describe a car. For each of these you can think of a meaningful type and range of values. Table 2-1 gives some examples, which we will use during the first modeling session with CBR-Works.

Table 2-1: Some Features of a Car

Feature	Type	Range
Brand	Symbol	Opel, BMW, Ford,...
Mileage	Integer	1 to 500,000 km
Price	Integer	1 to 200,000 DM
Color	Symbol	red, blue, green,...
Sunroof	Boolean	Yes, No
Hitch	Boolean	Yes, No
Catalytic Converter	Boolean	Yes, No
Seller	String	Free Text

2.2 Starting CBR- Works

We will now take a little walk through CBR-Works and build the first model of an application domain.

CBR-Works is started by double-clicking the *CBR-Works.exe* file in the folder where you installed the package. CBR-Works will welcome you with its start-up screen (figure 2-2).

Figure 2-2: CBR-Works Start-up Screen

After selecting the *New* item from the *File* menu the screen switches to the CBR-Works *Concept Manager* (figure 2-3). This is the editing view where you will enter the main information about the features of your *application model*.

Note: The steps described here are available as ScreenCams viewable from the CBR-Works CD

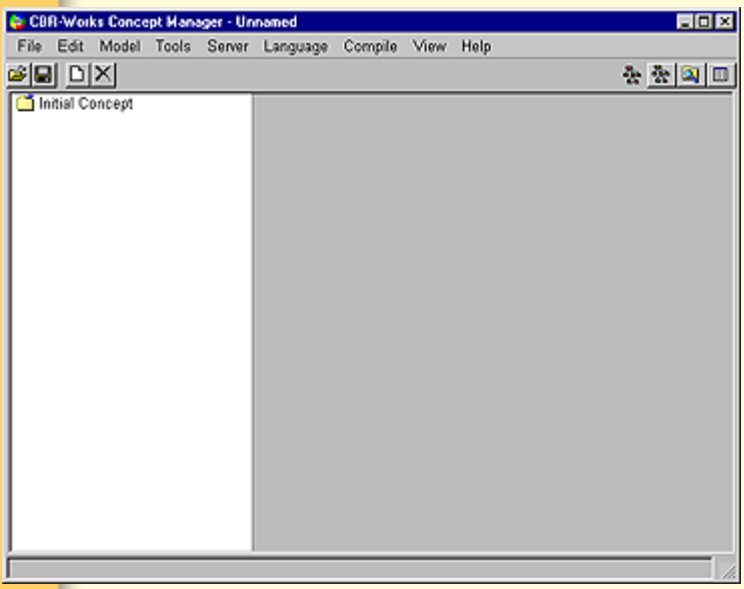


Figure 2-3: CBR-Works Concept Manager

At this stage we want to clarify some parts of the terminology that will be used throughout this document and that you need to know when using CBR-Works.

2.3 Terminology

When we deal with building a CBR-Works application, a number of terms are used to identify elements of the modeling process:

- By *domain* we denote the subject area of an application. In our example, we are working in the domain, or let us say in the world of used cars.
- The *application* is basically the software product that we build using the CBR-Works tools.

In our example, we describe an application that helps in selecting a suitable car from a newspaper advertising database. Such an application consists of a *case base* and a *domain model*, or simply *model.*

- The *case base* is the database where the case data is stored. In our example a single car advertisement will be a *case*.
- The *domain model* contains all information describing the relations among case data and the terminology of a domain. In other words: it adds the meaning to the pure data which is required to compare and select cases. In our example domain, the knowledge on colors provides such information, e.g., which colors exist and what is the similarity between those colors.
- A model will be described in terms of *concepts*, *attributes* and *types*. These correspond to the way we described used cars earlier: a car is the concept of our domain. A feature of a car is represented by an attribute of the concept. The value-range of an attribute is given by a type. This type contains all values valid for the feature which the attribute represents, e.g., for mileage a number restricted to the range between 1 and 500,000.

When building an application the construction of a domain model is the first step. The second step is the case collection. In reality there will be a lot of switching between these two steps, i.e., a prototypical approach. Now, before starting the modeling job, some basics on using CBR-Works.

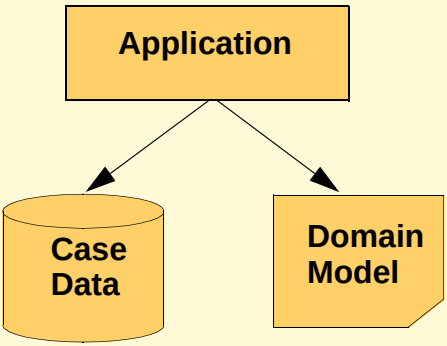


Figure 2-4: Basic Structure of a CBR-Works Application

2.4 Basics on Using CBR-Works

While working with CBR-Works you will navigate through four screens that allow you to access and to manipulate all relevant informations and data of your application. To switch between these views you can use the icon toolbar in the upper right corner of the CBR-Works window. The respective icons and a short description are shown in table 2-2.

Table 2-2: CBR-Works Views

Icon	CBR-Works View
	CBR-Works Concept Manager: in this view you will enter concepts and their properties.
	CBR-Works Type Manager: here you will define the types you want to assign to attributes. Basic types such as Integer, Symbol, etc., are predefined to ease your work.
	CBR-Works Case Explorer: allows you to enter case data and to organize the case base.

Icon	CBR-Works View
	CBR-Works Case Navigator: here you can test the retrieval properties of your application.

On the left hand side of the *Concept Manager*'s view you see a *Tree View*. It displays the structure of the concepts of your application, i. e., the hierarchy of concepts. You can use this view just like the Folder Tree of a standard Explorer window. When starting you will see only one concept: the **Initial Concept**. After selecting this concept its properties will appear on the right hand side of the screen. They are grouped on five card views as listed in table 2-3.

Table 2-3: Properties of a Concept

Card	Description
Properties	Here you can add additional information to a concept like, e.g., its name in different languages and its creator.
Attributes	A table displays the attributes of the selected concept. Here you can create, modify, and delete attributes. An attribute is specified by several properties like, e.g., its name, its type, and its weight.
Similarity	Specifies, how the similarity will be calculated for the selected concept. Similarities will be explained in a later section.
Rules	Here you can define rules for completion and adaptation for the selected concept. Please, refer to the CBR-Works Reference Manual for further details.
Questions	This group lets you define an order on questions. This will be used in the Query Wizard to determine the sequence in which questions are asked.

To model an application domain the attributes are the most important group. These are displayed first

by default. Let's start with our simple example application to learn how to enter attributes. Use the attribute definitions from table 2-1. Remember first the last step you did. It was to click on the initial concept symbol in the upper left of the screen. Now select the *New Attribute* button in the lower middle of the screen. An empty row will appear in the table above the button (figure 2-5).

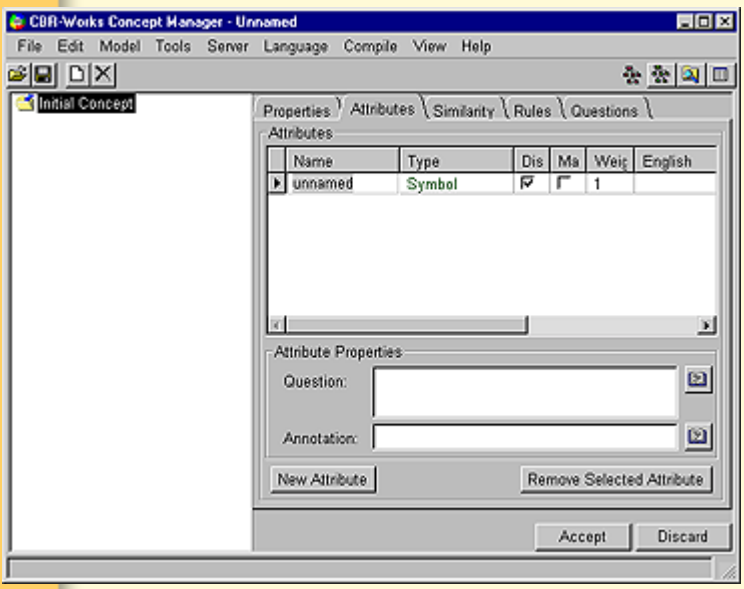


Figure 2-5: Screen before entering the Attributes Names

Enter the attribute's name (**Brand**) in the *Name* column and select the type *Symbol* from the selection list in the *Type* column.

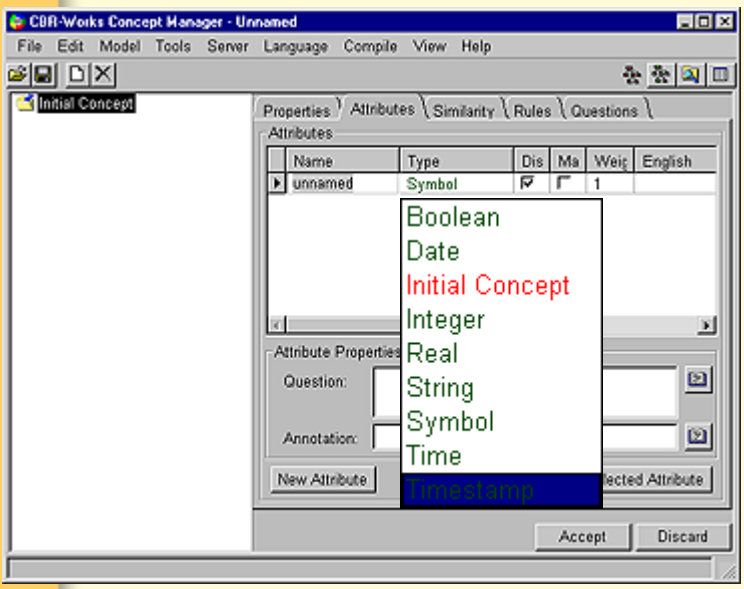


Figure 2-6: Entering Attributes

Repeat these steps for all the features from table 2-1 on page 3. After entering one attribute push the *New Attribute* button to enter the next one. Each time a marked section called unnamed will appear again by default in the name column. Don't worry. Just start typing the attribute's name and the default setting will disappear automatically. At this state of work the screen will appear analog to the following figure 2-7.

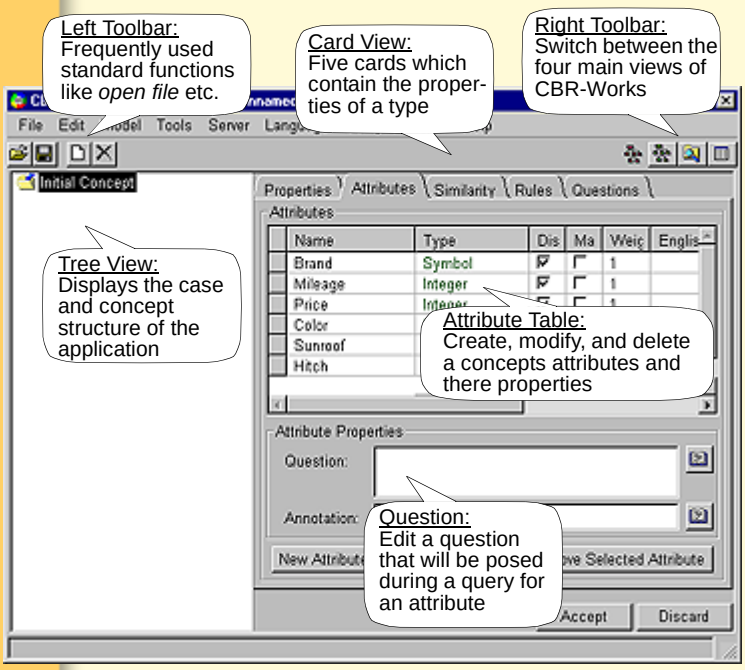


Figure 2-7: CBR-Works Concept Manager

When searching for a used car all these attributes are relevant except that one usually will not search for a specific seller of a used car in private advertisements. To exclude such an attribute from the search process you simply deselect its *Discriminant* check box in the *Discriminant* column of the table view.

In our example we only allow advertisements where **Mileage** and **Price** are given. To do so, we select the *Mandatory* check boxes for both attributes. A new advertisement will be tested on those attributes and it will be added to the searchable cases only, if **Mileage** and **Price** are given.

Name	Type	Dis	Ma
Brand	Symbol	☑	☐
Mileage	Integer	☑	☑
Price	Integer	☑	☑

Figure 2-8: Attributes, Types, Discriminant- and Mandatory Checkboxes

When you have finished all entries you must press the *Accept* button at the bottom of the screen to approve your changes. By selecting *Discard* you will discard all your recent changes.

The resulting attribute list should now look like the one in figure 2-9.

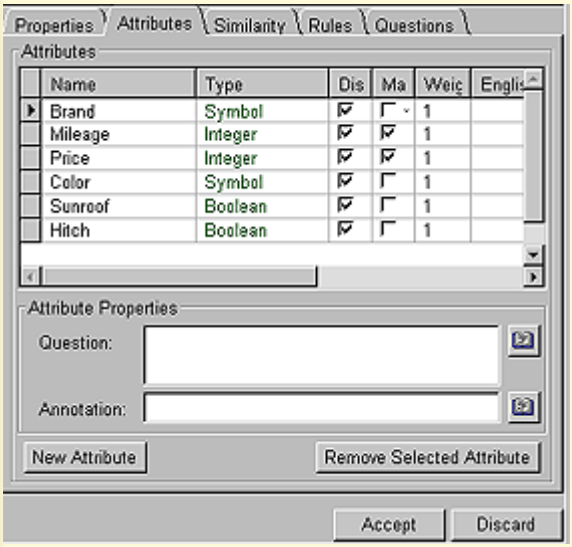


Figure 2-9: First Attribute List

You can also rename the **Initial Concept** to **Used Car** by selecting the Properties Tab. Use the *Accept* button to accept the new name

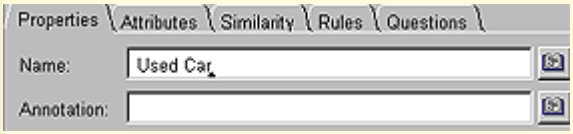


Figure 2-10: Renaming on the Properties Tab

In table 2-1 on page 3 we already specified additional information on the range of values of the used car features. To be able to incorporate such information in a CBR-Works application you have to create your own data types similar to a programming language like C or Pascal.

In our little example a type called **Car Colors** enumerating valid color values for cars would be suitable for the **Color** attribute of the **Used Car** concept. As indicated in table 2-1 on page 3 this should be a subtype of the predefined **Symbol** type.

Application specific types are always subtypes of some other yet defined types, i. e., they are range restrictions of existing types. The **Car Colors** type will be a subtype of **Symbol** that restricts the value range of **Symbol** to an enumeration of valid color symbols like **red**, **green**, **blue**, etc.

To define a new type you have to switch the CBR-Works screen to the *Type Manager*. Select the *Type Manager* button in the right top of the screen.This is very important. If you don't push the green *Type Manager* button the screen will not change to the tree view which is necessary to define new types.

To create the new type select the type that will be its super-type in the tree view on the left hand side of the screen. For **Car Colors** this is **Symbol**. Now select the *New Type* button in the toolbar above the tree view. A dialog box appears where you can enter the name (**Car Colors**) of the new type. Mind pushing the *OK* button.

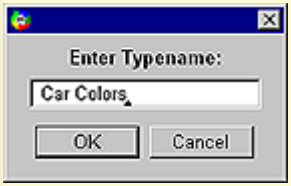


Figure 2-11: Typename Dialog Box

On the right hand side of the display a card view shows up with three groups of type properties (table 2-4). First of all select the card *Range*. The next step is to change the default set up in the lower middle of the screen. Select *Enumeration* instead of *No Range*. Put in the new values in the *New Values Input Field* directly above and push the *Add* button.

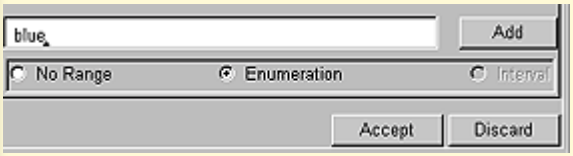


Figure 2-12: New Values Input Field

The following figure shows the displayed screen where a new type **Car Colors** has already been created.

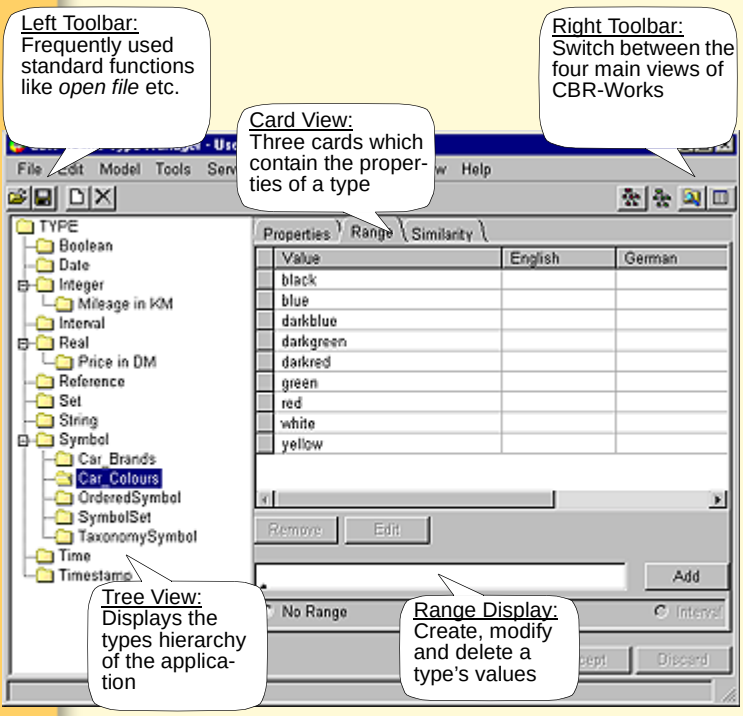


Figure 2-13: CBR-Works Type Manager

Repeat these steps for every new value. Finally use once again the *Accept* button.

Table 2-4: Properties of a Type

Card	Description
Properties	Here you can add additional information to a type like, e.g., its name in different languages and its creator.
Range	Displays the value range of a type depending on the kind of a type.
Similarity	Specifies how the similarity will be calculated for the selected type. Similarities will be explained in a later section.

Enter the remaining types analogous, i. e. a type **Car Brands** as a subtype of **Symbol** and two **Integer** subtypes for **Mileage** in **KM** and **Price** in **DM**.

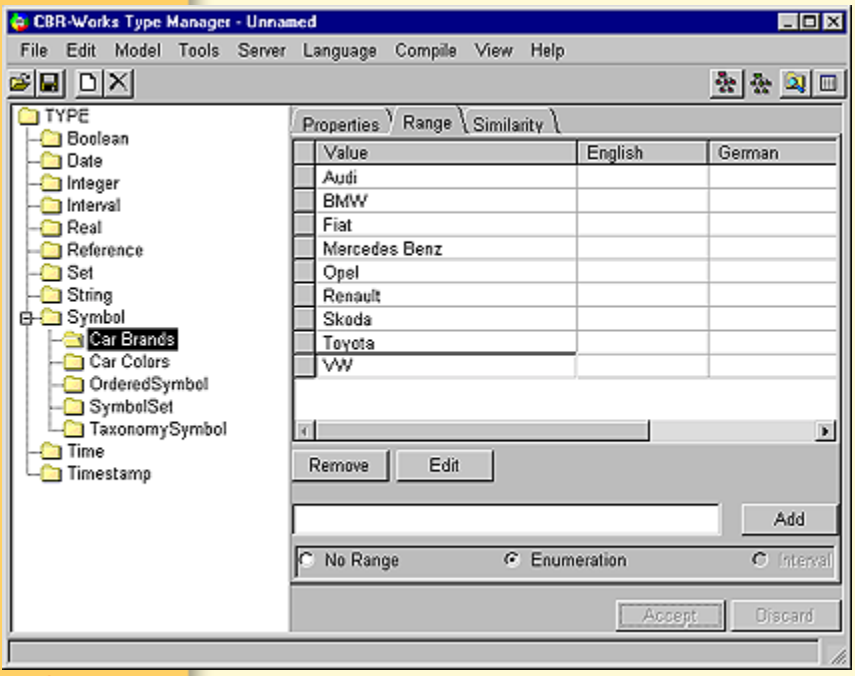


Figure 2-14: Entering the new type Car Brands

When you create the **Integer** subtypes you will notice that you have the choice between defining an interval of values or to enumerate the type's values. For numerical ranges an interval will be the natural choice in most cases. Always choose the bounds of such an interval as narrow as possible because it influences the calculation of the similarity measure. When your bounds are the maximum and the minimum integer values then the calculation is still correct and valid but not very helpful.

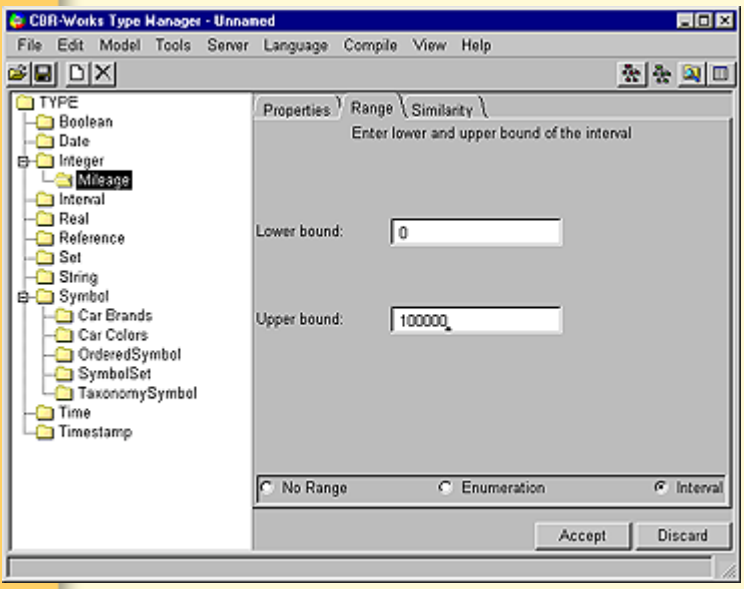


Figure 2-15: Mileage Interval

In table 2-5 a summary of all predefined basic types is shown that are provided with CBR-Works.

Table 2-5: Predefined Types

Type	Range/Description
Boolean	true, false
Integer	[-5e8..+5e8]
Real	[-1.0e30..1.0e30]
String	Arbitrary character strings or a fixed enumeration of strings
Symbol	Arbitrary or fixed enumerations of character sequences
Ordered Symbol	Fixed and ordered enumeration of symbols. The order is given by the sequence of values in the enumeration

Type	Range/Description
Taxonomy Symbol	Fixed and partially ordered enumeration of symbols. The order is given by a hierarchical tree structure
SymbolSet	(Obsolete, use Set and Symbol instead)
Set	All but Boolean. Sets can be defined over any other defined Type
Date	Enumeration or interval of dates in the form day/month/year
Time	Enumeration or interval of time in the form hh:mm:ss
TimeStamp	Enumeration or interval of timestamps in the form day/month

Now switch back to the *Concept Manager* to associate the new types with the respective attributes, i. e., select the **Used Car** concept and the **Color** attribute. When you open the selection list in the types column of the attribute table you will see that the new types appear as a valid type selection.

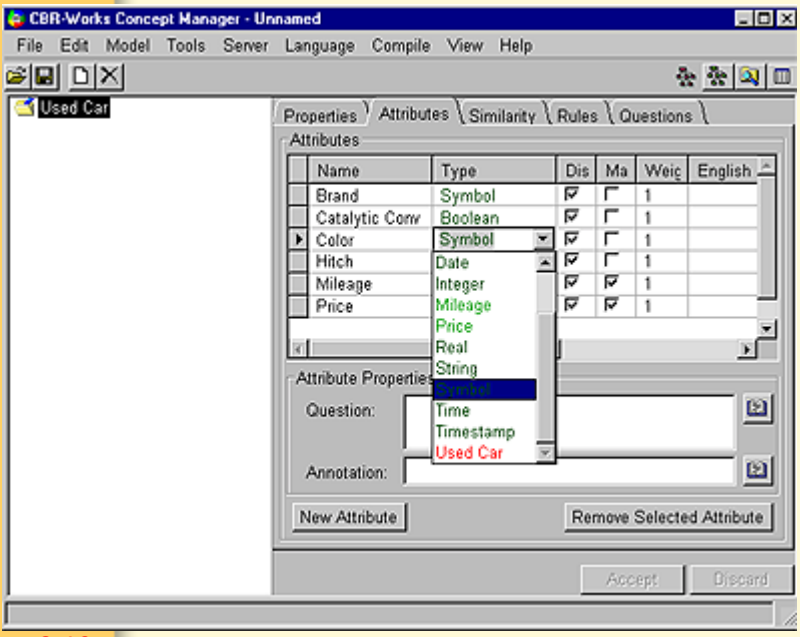


Figure 2-16: New Valid Types

CBR-Works provides a powerful maintenance component to apply all changes in the model to the case base. If an operation, like changing the type of an attribute, might cause data loss in the case base, a notifier pops up telling you so. Because we do not have established a case base at this point, we may ignore this notifier and close it by pushing the *OK* button. For advanced users it is possible to switch off such warnings by changing the according preferences.

Before you now enter your first cases into the case base you should save your first application model. Select *Save as...* from the *File* menu. A dialog appears (figure 2-17). Choose an appropriate volume and folder, type a name for the new application, and select *Save*. CBR-Works stores a binary description

of your domain into a new folder with the application name.

Figure 2-17: Save Dialog

Note: Instead of saving an external application domain description you can save the current state of your actual work. Selecting *Save Workspace* from the *File* menu will open a dialog (figure 2-18). Choose a name to save Workspace to and press the *OK* button. This will store the complete state of CBR-Works into a CBR-Works folder, including domain model, cases and preferences. To restart CBR-Works later first open your Windows Explorer and doubleclick the CBR-Works folder. On the right hand side of your screen you will recognize the name of the Saved Workspace Document. Doubleclick it and you will be back exactly in the state of the software where you saved the workspace last time.

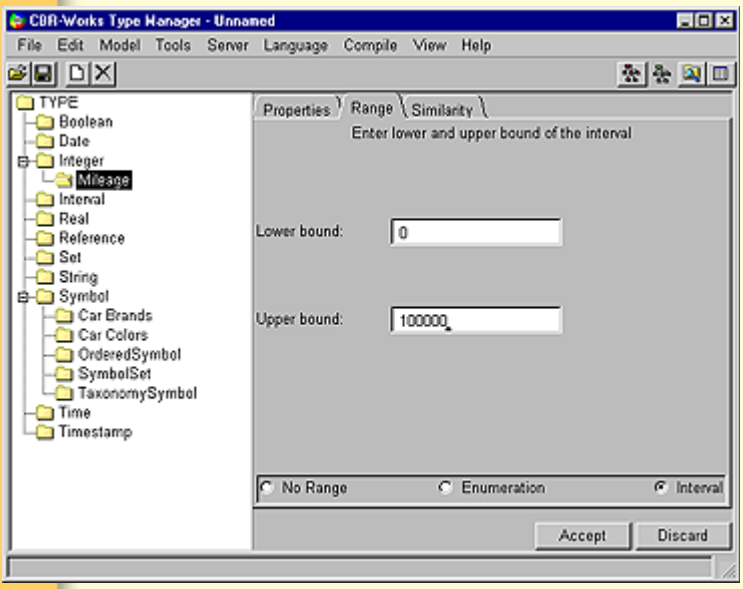


Figure 2-18: Save Workspace Dialog

2.4.1 How to find the saved application model

Remember the last step you did. You saved your first application model. For further modeling steps it is first necessary to open exactly the model you saved. Select *Open* from the *File* menu. A Dialog will open that looks like the Save Dialog we described on the last page.

Figure 2-19: Open Dialog

But there are some important differences. In the lower middle of the screen you will notice that the check boxes have changed from save to load.

At the bottom of the screen instead of the *Save* button you will see an *Open* button. Push this button. The screen will now appear as described by the following figure.

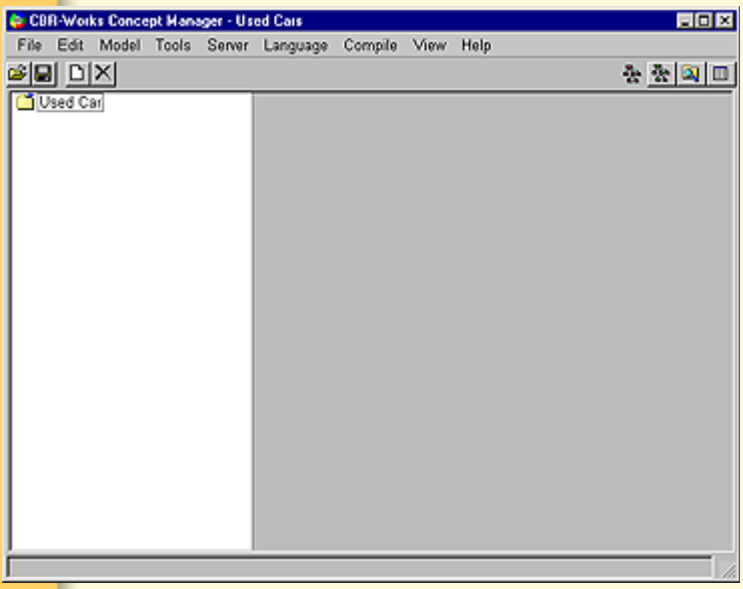


Figure 2-20: After opening the Saved Model

To enter cases into the case base you have to switch to the *Case Explorer*. Select the *Case Explorer* button from the upper right toolbar.

2.5 Working with the Case Base

The CBR-Works Case Explorer is the basic tool to manage your case data. It consists of a tree view to display the case structure as well as the case list and a table view to display and edit case data.

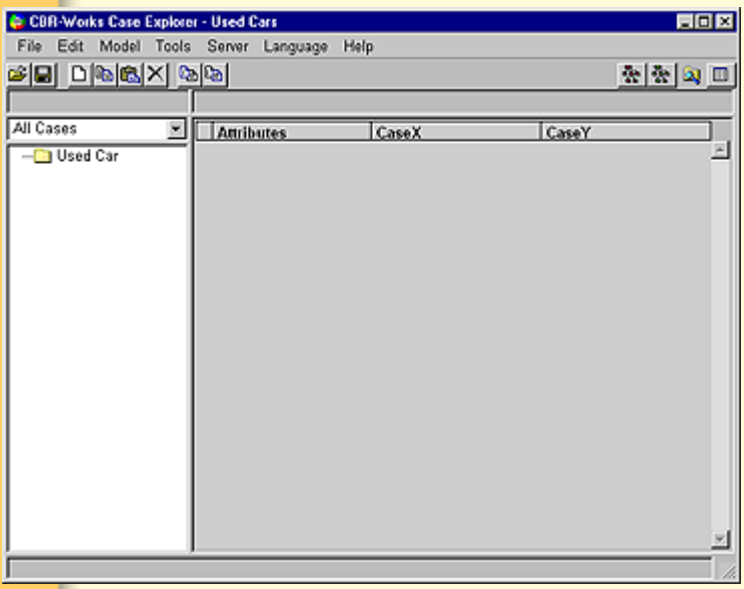


Figure 2-21: After switching to the Case Explorer

Cases are stored in the Case Base. A case can be in one of four modes as described in table 2-6.

Table 2-6: Case Modes

Symbol	Mode	Description
	Unconfirmed	Case is incomplete or not yet validated. Cases of this mode will not be retrieved.
	Confirmed	Case is complete and validated. Cases of this mode are allowed for retrieval.
	Protected	As Confirmed, but protected against unwanted changes to the case.
	Obsolete	Case contains old data, may be interesting for statistics. Cases of this mode will not be retrieved.

The process of case data management usually needs to be embedded into the workflow where the CBR-Works application is deployed. In the used car example a new advertisement would be entered by the advertising office. The responsible editor checks the new incoming adverts and only after his approval and maybe some corrections they will be changed to confirmed. Such a process ensures the validity and consistency of a case base during day to day work.

The process of case data management usually needs to be embedded into the workflow where the CBR-Works application is deployed. In the used car example a new advertisement would be entered by the advertising office. The responsible editor checks the new incoming adverts and only after his approval and maybe some corrections they will be changed to confirmed. Such a process ensures the validity and consistency of a case base during day to day work.

Now start to enter some new cases for the used car application. Select the *New Case* button from the tool bar. The *New Case Dialog* will open (figure 2-22).

It shows a tree view that displays the structure of the case concepts of your application. You may have a number of different case concepts in one application because cases not necessarily have the same structure. If we would have advertisements for utility trucks in our example application then we certainly would describe them different than cars.

Figure 2-22: New Case Dialog

Actually, you see only one case concept - **Used Car**. Select it and type a name for the new case in the *Naming Field* below the tree view. You may leave the automatically generated name, but often specific names are preferred.

Note: Cases have a unique identifier that CBR-Works assigns automatically, hence, case names can occur more than once.

The newly created case appears now in the tree view of the *CBR-Works Case Explorer*. If you want to create another new case simply repeat the procedure. Here, in out little example we have created two new cases called **Vectra1** and **Vectra 2**. Both have been inserted below the **Used Car** concept (figure 2-23).

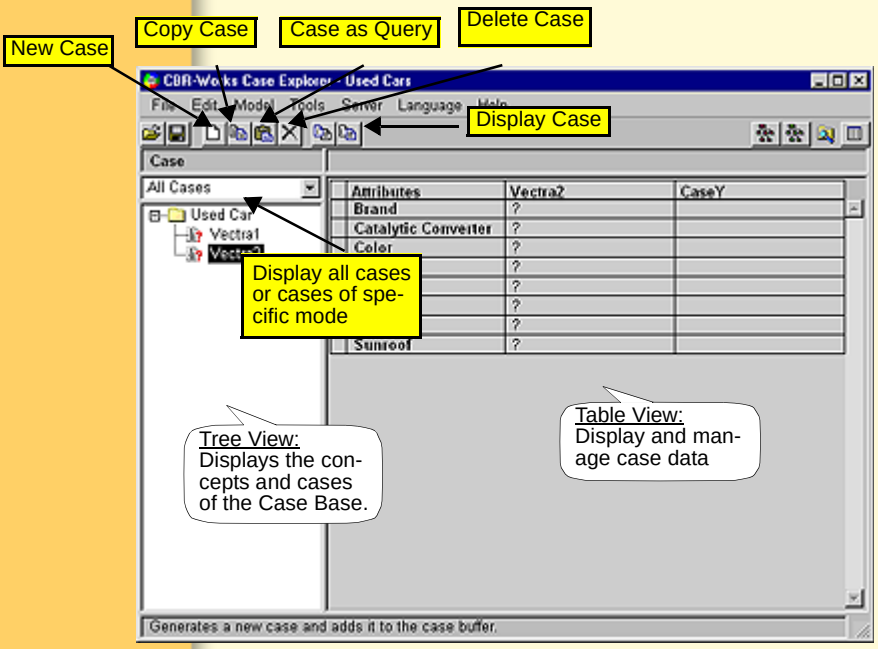


Figure 2-23: Newly created Cases in the Unconfirmed Mode

To make new cases available during the search process you have to change their mode either to *confirmed* or to *protected*. Thereby you can include or exclude cases from the retrieval process as necessary. You don't have to change the modes at this point of modeling. Step forward according to our description and then change the modes when it is mentioned in the text. This will be some steps later before you start doing case retrieval.

Cases of the different modes may be viewed separately using the drop down menu above the tree view. You can select to show either all cases or cases of a specific mode as described in table 2-6 on page 24.

When you double-click a case it will be displayed in the second column of the table view on the right hand side of the screen. The first column displays the attribute names of the case. The third one can also display a case to ease comparison during case data edition. You can view a selected case by selecting one of the two *Display Case* buttons from the toolbar.

To enter or modify case data simply double-click the respective *Attribute Value Field* in the table view. Depending on the attributes type a dialog will appear that lets you edit the attributes value. After editing always select the *OK* button.

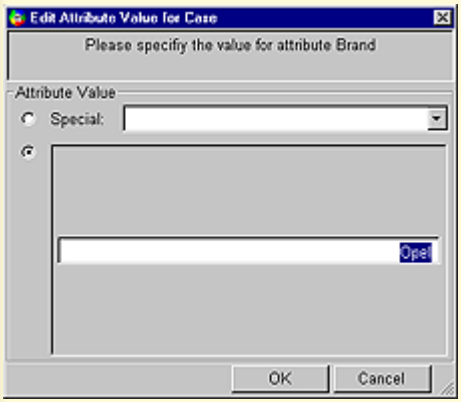


Figure 2-24: Input Dialog for an Enumerated Symbol

For any attribute's value you may specify that the value is *undefined* or *unknown*. CBR-Works will even work if you do not know exactly what brand, color, mileage etc. you prefer. For numerical value ranges the input dialog additionally displays the defined upper and lower bounds for valid entries.

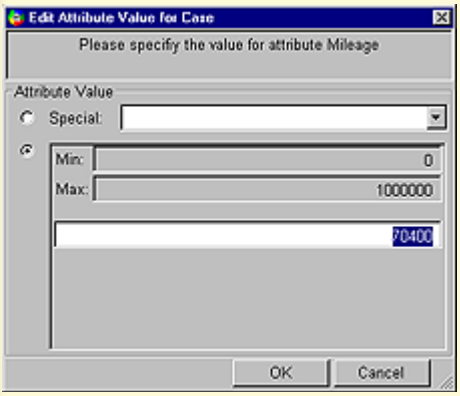


Figure 2-25: Input Dialog for a Numerical Value

In the top of an input dialog for an attribute value the standard phrase *Please specify the value for attribute ...* will be displayed. If you specify a question in the attribute definition in the CBR-Works *Concept Manager* this will be shown instead (Chapter 4.2).

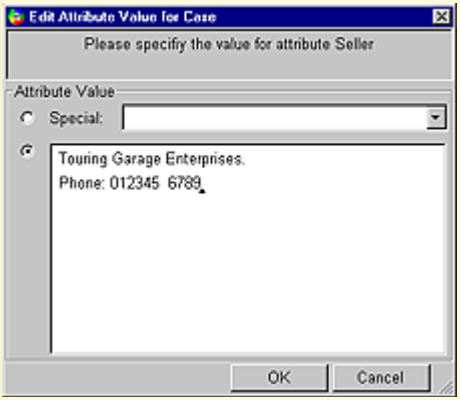


Figure 2-26: Input Dialog for a Textual Value

After you have entered your cases your screen will have the design of the following figure.

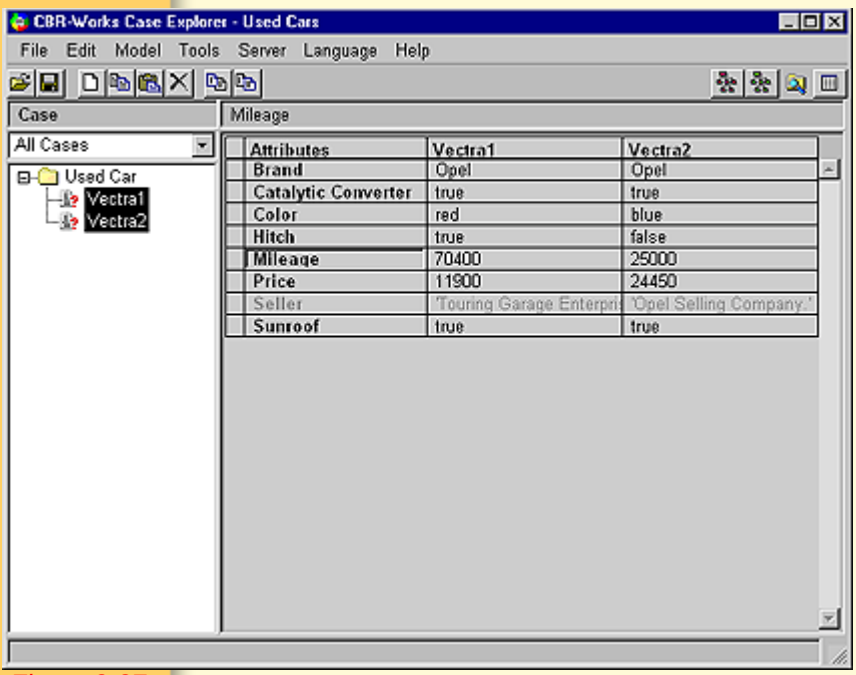


Figure 2-27: CBR-Works Case Explorer

What you should have in mind is that both new cases are still unconfirmed. Now is the time to change their modes to be able to do case retrieval. Simply select the *Select All* item from the *Edit* menu and when all cases in the tree view are selected, choose the *Confirmed* item in the *Mode* submenu (see figure 2-29 on page 31).

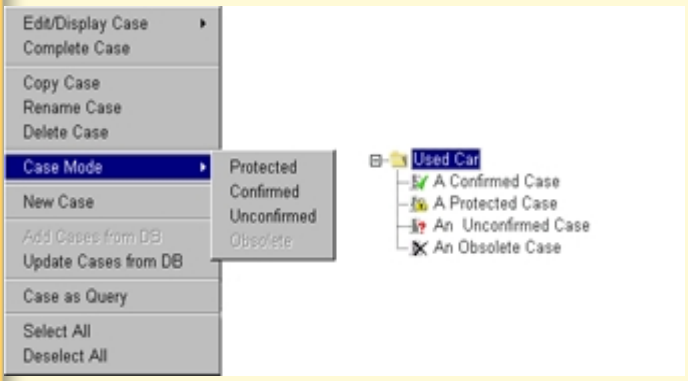


Figure 2-28: Case Mode Menu

What you will see now is demonstrated in figure 2-29. The former mode which was unconfirmed has changed to the confirmed mode.

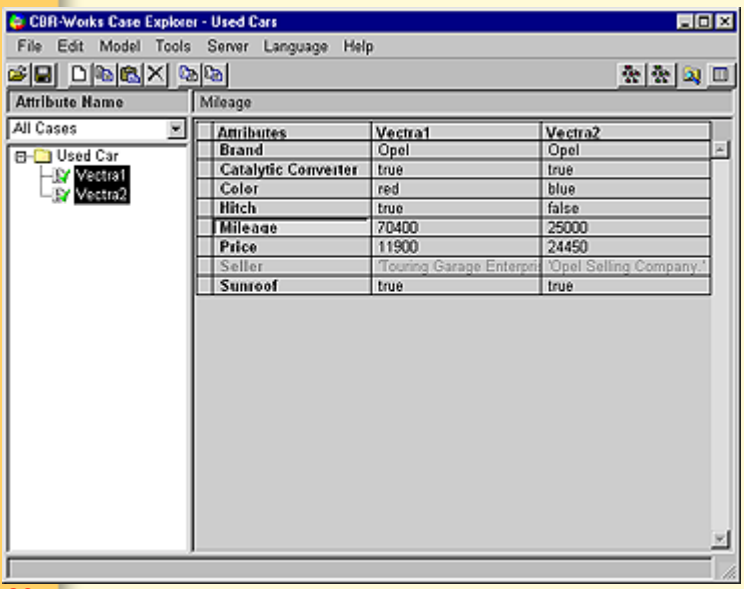


Figure 2-29: Case Explorer in Confirmed Mode

Note: If you want to protect cases from (accidentally) modifying their contents, change their mode to *protected* instead of using *confirmed*.

2.6 Consulting the Case Base

Now, that you have entered your first cases you can start doing retrievals on the case base.

To switch to the *CBR-Works Case Navigator*, select the *Case Navigator* button from the right toolbar. The structure and the handling of this generic consultation interface is similar to the table view of the *CBR-Works Case Navigator*. What you see now on your screen is illustrated by figure 2-30.

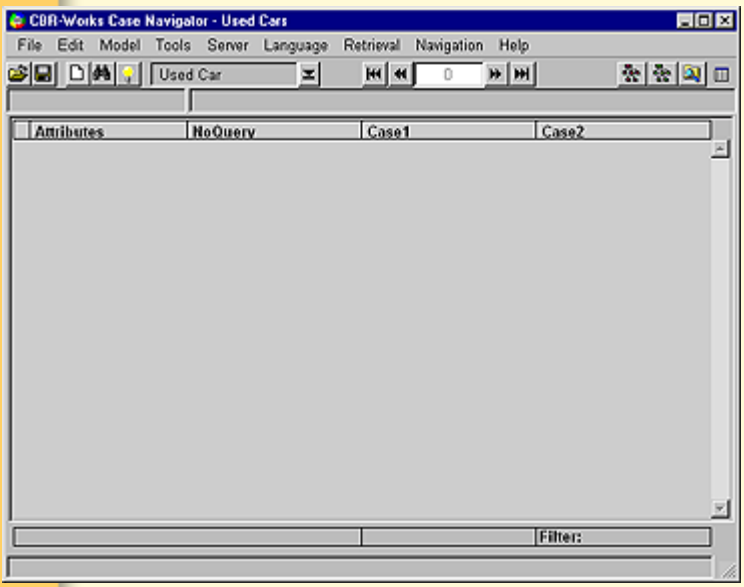


Figure 2-30: Works Case Navigator

First you have to create a new query by selecting the *New Query* button. If there are subconcepts of the case-concept then you will be asked for the specific concept as when creating a new case.

In this case, choose **Used Car**. In our example domain, **Used Car** will automatically be chosen as concept for the query.

.

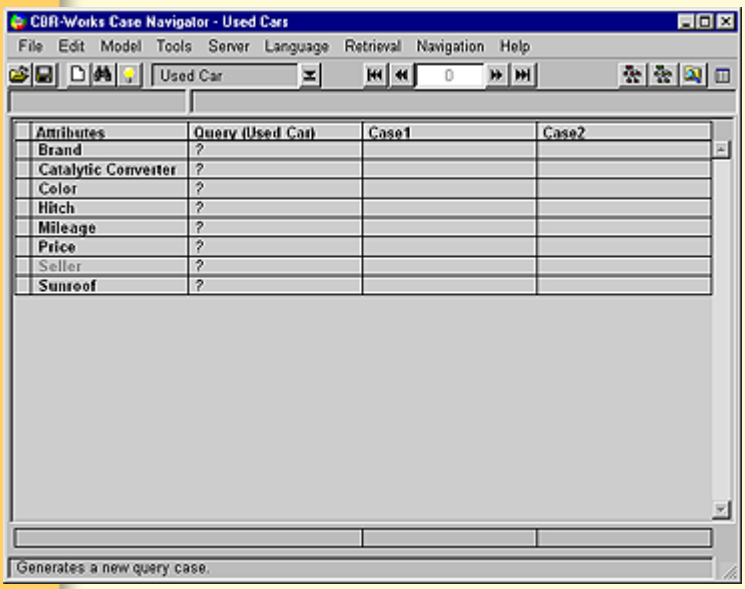


Figure 2-31: Ready to enter Values

Now your screen looks like the one in figure 2-31. To enter values for the query just double-click the respective attribute cell in the query column of the table view. The entry handling is the same as that of the *Case Explorer*.

Note:

You can also use an existing case as query. Switch to the *Case Explorer* and select a case in the tree view on the left hand side of the screen.Then select *Case as Query* from the *Edit* menu and answer *OK* to the following dialog. When you switch back to the *Case Navigator* screen the data of the selected case will appear as the query.

Now apply the note above and select the case **Vectra1** as the query.

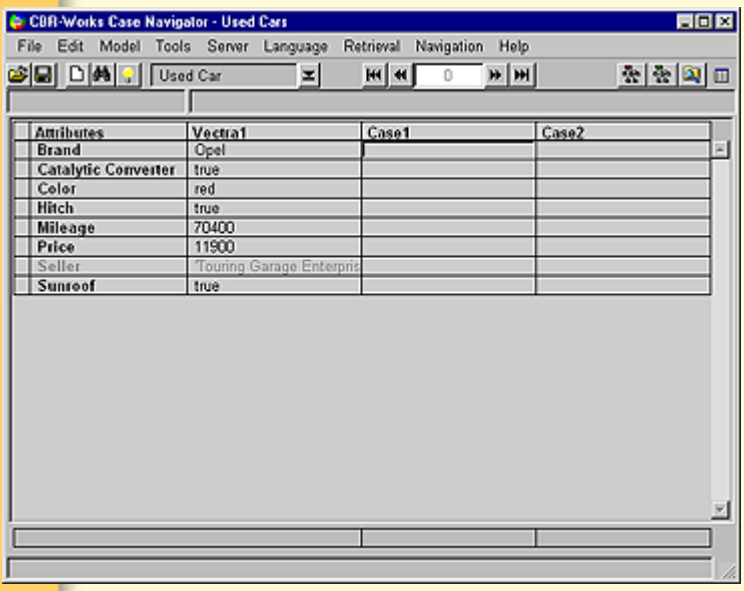


Figure 2-32: Vectra 1 as Query

To start the retrieval process select the *Retrieval* button above the attributes column. The two resulting most similar cases are displayed in the third and fourth column. The most similar case is **Vectra1** itself, as expected. It has the similarity **1.0**.

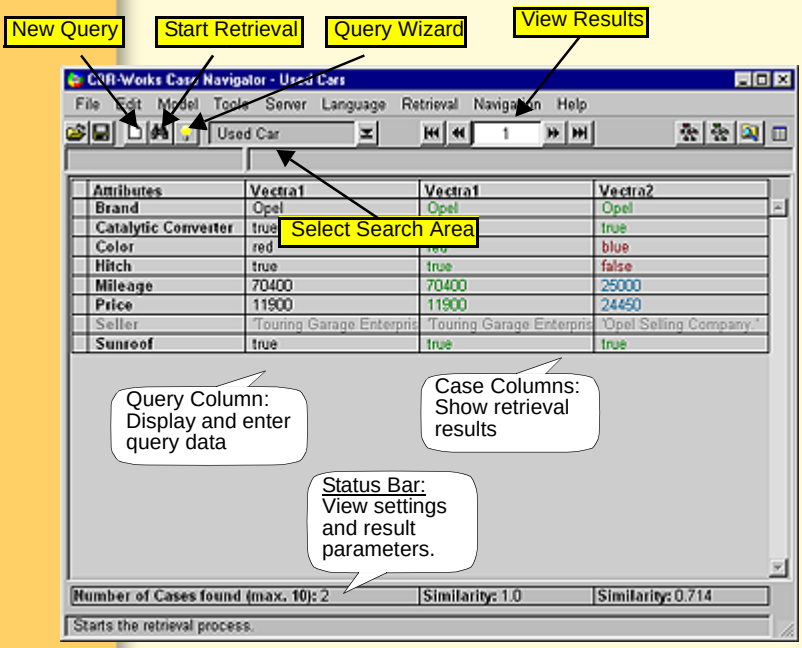


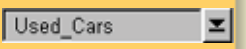
Figure 2-33: CBR-Works Case Navigator

The similarity of the retrieved cases shown in the third and fourth column is shown in the status bar right below each case. The left field of the status bar shows the maximum number of cases to be retrieved (which can be set in the preferences) and the number of cases found in the case base which are similar to the specified query.

The result of a retrieval can consist of several cases. To navigate through this list you utilize the *Navigation* buttons from the toolbar. They work similar as one is used to from a VCR: *forward/backward*

moves one case forward or back in the list. *Fast forward/backward* jumps to the begin or end of the list, respectively. The number between the buttons displays the position in the result list, i. e., a **1** indicates that the most similar case is displayed, a **2** for the second best and so on.

When your application contains a structured case base like two differently described sorts of vehicles, e. g., used cars and utility vehicles, then you can select the search area that the retrieval engine will consider. When you click on the *Select Search Area* menu field then a *Concept Selection* dialog will show up, if there are subconcepts defined for the case-concept. It is the same like the ones when creating a new case or query. The retrieval engine will search through all those cases that belong to the concept or its subconcepts that you choose. In our modeling task there are no subconcepts defined. Therefore the dialog will not show up.



2.7 Advanced Queries

Often you need to specify advanced conditions for a query. If you own a caravan then you certainly will need a car that has a hitch. If you look for a blue car with a hitch and a mileage of **20000** in our **Used Car** example then you will retrieve **Vectra2** as the most similar case, but **Vectra2** does not have a hitch. Maybe the car fits better to you than the other but it's obviously not the car you hoped to find. For better results the system needs further information that can be described as a ranking of your desires. Here, in our small modeling example you need to express that a hitch is important for you or that you are only looking for cars that have a hitch. This is possible by

applying importance and query in the *CBR-Works Case Navigator*.

For queries, the input dialog for attributes allows to change the filter (figure 2-34) and importance (figure 2-35) properties of an attribute additional to its value.

Importance and filter are properties that influence the behavior of the retrieval engine. By specifying a filter for an attribute you restrict the search to all those cases which fulfill the filter condition.

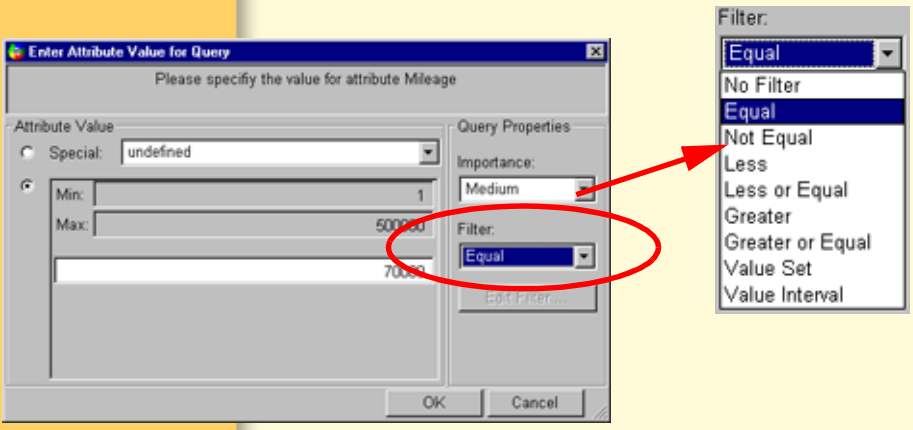


Figure 2-34: Options for Filter

In our **Used Car** domain we can enter the query attribute value true for Mileage and select the filter condition **Equal**, i. e., Mileage = **70000**. The search is now restricted to cars that have a mileage of 70.000 km. Of course, thus a hard condition would not be very helpful. But you may set a filter that limits the query to cars that have a hitch by specifying a filter condition **Hitch = true.**

The possible filter conditions are given by the attributes kind of type, e. g., in the case of free textual values only **Equal, Not Equal, No Filter and Value Set** are offered. In general you can select one of the options shown in figure 2-34.

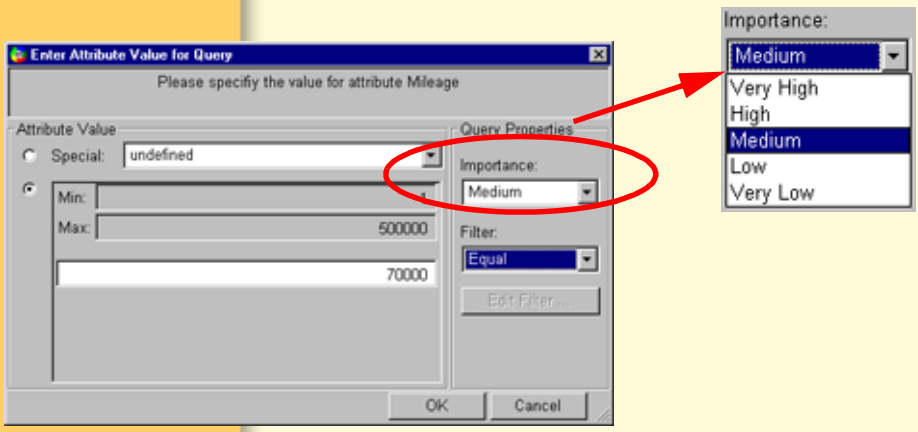


Figure 2-35: Options for Importance

Importance influences the calculation of the similarity. The higher an attribute's importance the more it will count in the calculation of the concept's similarity. One can choose of five levels of importance as shown in figure 2-35. For example, by setting the importance of the hitch attribute to **high** the influence of the other attributes becomes minor.

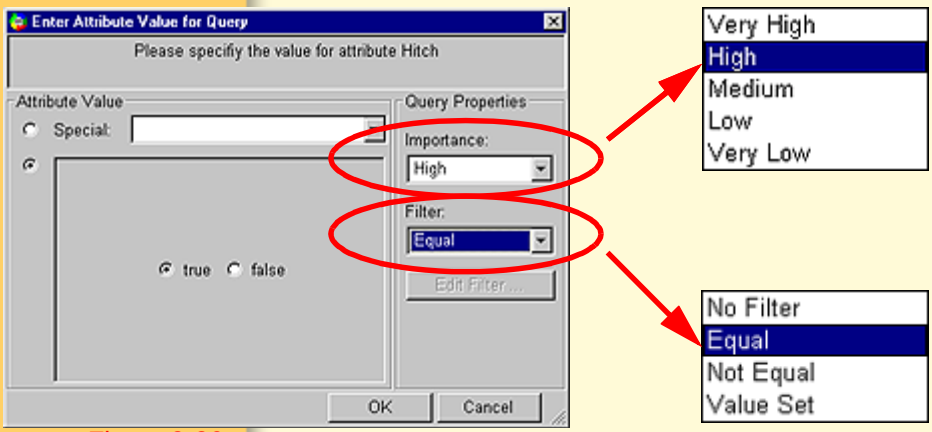


Figure 2-36: Options for Filter and Importance of Textual Values

During the similarity calculation only those attributes that have a defined value are considered.

If you did follow all the steps in our small modeling exercise you will now see a screen that looks like the next figure.

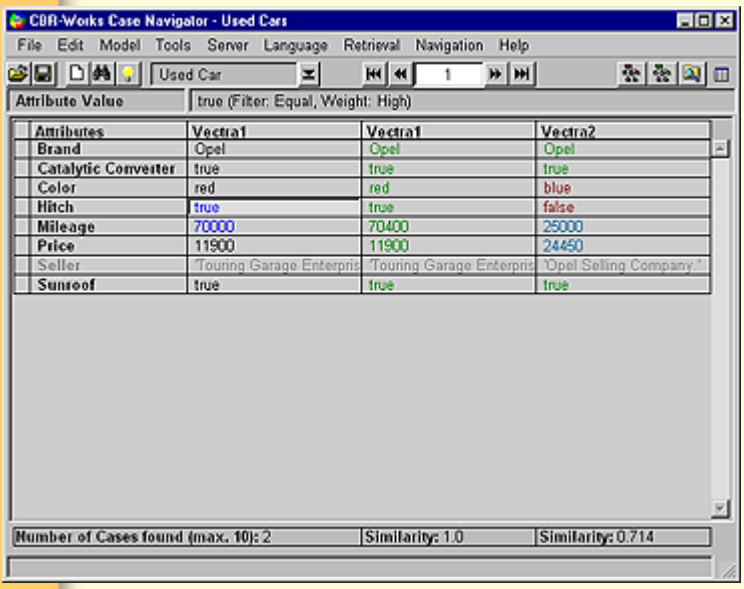


Figure 2-37: CBR-Works Case Navigator after setting the Options

2.8 Query Wizard

The *Query Wizard* offers an additional way of querying a case base. Instead of letting you specify attribute values in an arbitrary order it determines an order that lets you retrieve cases by entering the minimal amount of required values, i. e., the *Query Wizard* offers a guidance through the retrieval process.

CBR-Works offers different kinds of how the order is determined. By default, this is done by computing the so-called *Information Gain* of the discriminant attributes. It is a measure for the degree of differen-

tiation of an attribute for the case base, i.e., which attributes are sufficient to answer to find the most similar cases.

Before you start the *Query Wizard*, make sure to create a new query. (It makes no sense to use the wizard with an already filled query). CBR Works will ask you to discard the old query. Select the *OK* button.

To start the wizard, click on the *Query Wizard* button in the upper left tool bar. The wizard's window is shown in figure 2-38.

The list of questions in the top of the window displays the questions for the relevant attributes ordered by their respective information gain. When you choose a question, the according value editor shows up below the list.

When the screen appears for the first time after pushing the *Query Wizard* button the question dealing with the **Hitch** is preselected.

If you wonder why, remember that the former step you did at last was to specify the value for the attribute **Hitch** by setting the importance and filter options to high and equal.

In our little example we have selected the question confirming the attribute **Color**.

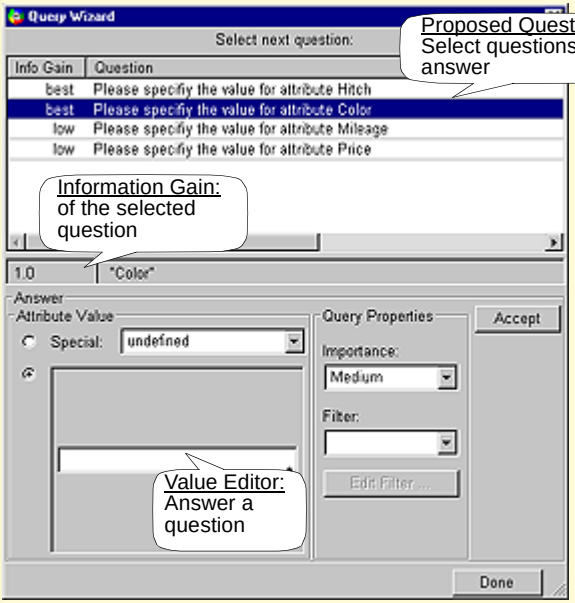


Figure 2-38: Query Wizard

In the Value Editor we entered the color *red.*

After selecting or entering a value and clicking on the *Accept* button, the most similar cases are retrieved and the list of questions will be updated based on the results.

You may return to the normal mode of the consultation at any time by selecting *Done*. Because of your decision for the color red in the Value Editor the color now appears in the query column. Look at the nextcoming figure 2-39.

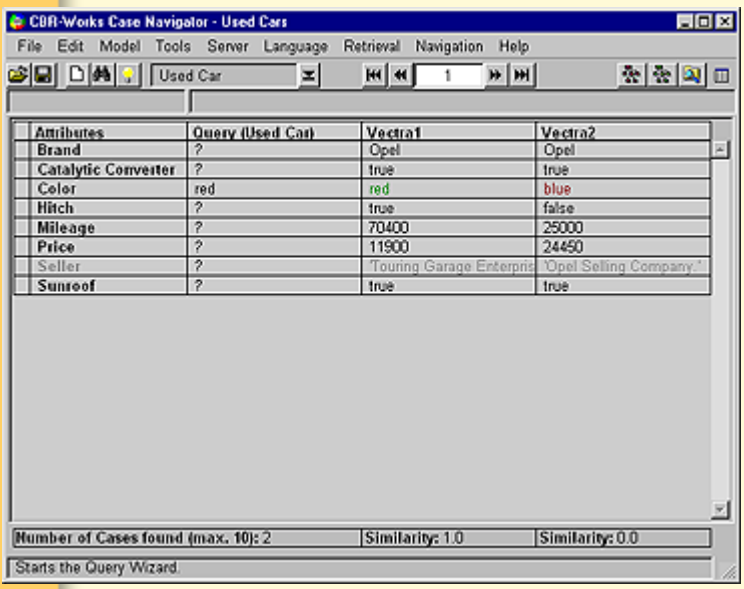


Figure 2-39: After modeling with the Query Wizard

2.9 Summary

You have now visited all main parts of the CBR-Works application development environment, which were

- the *Concept Manager* to describe the “objects” of your application domain;
- the *Type Manager* to define the valid value ranges of the objects’ attributes;
- the *Case Explorer* to enter and to manage the case data;
- the *Case Navigator* to test the application.

3 Enhanced Similarity Model

Our first **Used Car** application domain model was very simple and only meant to illustrate the main parts of CBR-Works. In the next chapters, we will have a closer look at the domain model and apply some more sophisticated modeling techniques to reflect its particularities.

The domain model describes the application domain and therefore determines the quality and the convenience of the application for its users. We want to motivate this based on a modeling task that we call the "Color Problem".

3.1 The Color Problem

The term “color” is a very ambiguous one. Every individual has got its own imagination and feeling of colors. This makes it difficult to describe and handle a color in a way that can be easily understood by someone else.

In our car example from Chapter 2 we just enumerated possible colors of a car like **blue**, **green** or **red**. In real life this could also be **dark blue** or even **dark metallic blue**. This illustrates that the final enumeration would be exhaustive.

The next problem arises when comparing these colors: Is **dark metallic red** closer to **metallic red** or to **dark metallic blue**. The answer depends on whom to ask - it often is a matter of taste, i.e., something one

cannot and should not code into a simple data model. A programmatic attempt could perhaps reduce all colors to their respective base, or primitive, color- **dark metallic red** is reduced to **red** this way -, use a syntactical approach or a physical-based model, for example. But furthermore, how to classify the color **bordeaux,** which is normally a sort of **dark metallic red**, within an enumeration like the above?

3.2 Taxonomy Modeling

A different modeling approach to overcome the problems of enumeration and to model the similarity of colors could be to structure the colors in a hierarchy. This results in defining a taxonomy of the used colors (figure 3-1). Thereby we obtain a *partial order* of the colors but we do not overcome the problem of having a kind of metallic red as well as a kind of normal red. Being metallic or normal seems not to be a common property of basic colors.

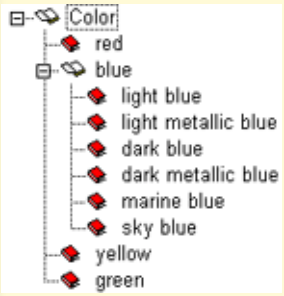


Figure 3-1: A Taxonomy of Colors

Using a taxonomy is often a suitable modeling alternative if an enumeration does not provide the information needed to calculate a similarity of an attribute. Here, in the domain of car paintings, the hierarchy defined by the taxonomy does not really

lead to satisfying results. Therefore, we may look for a different modeling approach.

3.3 Detailing and Similarity

The above considerations show that it is not possible to discuss a term like color without its context. In the case of cars we particularly are not talking about color. Instead we are talking about the concept of car paintings that has a number of aspects of which the color is only one. So, to model the painting of a car, we may detail the “color” by using a *sub-concept* that consists of different attributes of the painting. The painting of a car is basically comprised of

- the varnish type (**metallic** or **normal**)
- its primitive color (**white**, **black**, **red**, **green**, **blue**, **orange**, **yellow**, **grey,** etc.)
- and its brightness (e. g. **light**, **medium**, **dark**).

Based on this concept, it is easy to describe any usual car color in a common and unified way. The color **bordeaux**, for example, has the varnish type **metallic**, the primitive color **red** and is of **dark** brightness.

To compare two car paintings one has to compare the respective properties. Such a comparison requires a measure, i. e., something describing the meaning of similarity: a sort of function that is called a similarity measure. For the type of varnish this is the equality, **metallic** is equal to **metallic** but unequal to **normal**. In the case of the primitive color it can be expressed utilizing a table and in the case of brightness it can be derived from the order of values, i. e., **light** is more similar to **medium** than to **dark**.

Figure 3-2: Car Painting Concept

To measure the degree of similarity we use a figure between **0** (unequal) and **1** (identical) that can be seen as a percentage. A cross-table can be used to define all similarities. For the attributes of **Car Painting** these are shown in tables 3-1 to 3-3. Because the **Varnish type** only consists of two alternatives, **normal** and **metallic**, the similarity is not really spectacular. The tables for the attributes **Color Brightness** and **Car Colors** show a more interesting similarity definition - nevertheless all similarities used in this example are symmetric.

Note:

Similarities are defined on the type of an attribute in respect to the range of the possible attribute values. The resulting similarity for a complex subconcept, i. e., of two car paintings, can be expressed as the weighted sum of its attributes' similarities.

Table 3-1: . Similarity for Varnish Type

Varnish Type	normal	metallic
normal	1.0	0.0
metallic	0.0	1.0

Table 3-2: Similarity for Color Brightness

Color Brightness	light	medium	dark
light	1.0	0.5	0.0
medium	0.5	1.0	0.5
dark	0.0	0.5	1.0

The similarity measures for **Varnish Type** and **Color Brightness** is implicitly given by their type: A simple enumeration of symbols, as for **Varnish Type**, results in a similarity which is **1** only if a queried value is equal to the one in a case, else **0**. Using an *Ordered Symbol* type for **Color Brightness**, entering **light**, **medium**, and **dark** in this order, results in a similarity as shown in table 3-2.

Table 3-3: Similarity for Car Colors

Car Colors	black	blue	green	red	white	yellow
black	1.0	0.6	0.0	0.0	0.0	0.0
blue	0.6	1.0	0.6	0.0	0.0	0.0
green	0.0	0.6	1.0	0.0	0.0	0.3
red	0.0	0.0	0.0	1.0	0.0	0.3
white	0.0	0.0	0.0	0.0	1.0	0.6
yellow	0.0	0.0	0.3	0.3	0.6	1.0

For **Car Colors** type, the similarity has to be specified explicitly using the Similarity Tab (figure 3-3). Here, select the *Add* button which will open an input dialog requesting for the name of the new similarity. With the *Edit* button the *Similarity Editor* (figure 3-4) will be opened. Use the *Similarity Mode Selector* to specify the mode to be used for entering the similarity. For our example, choose *standard*.

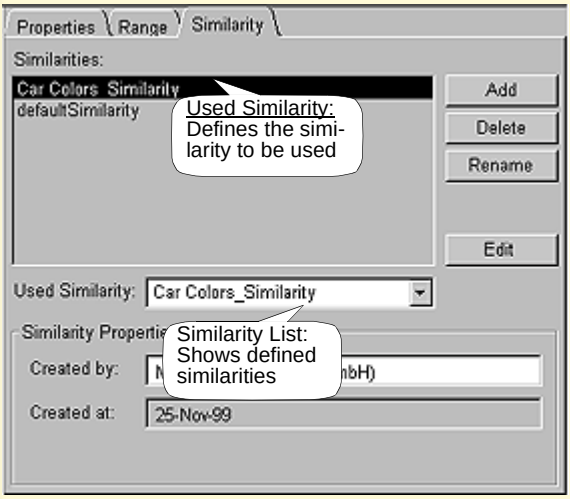


Figure 3-3: Type Manager Similarity Card

When you finished entering the similarity values, push the *OK* button to accept and close the editor.

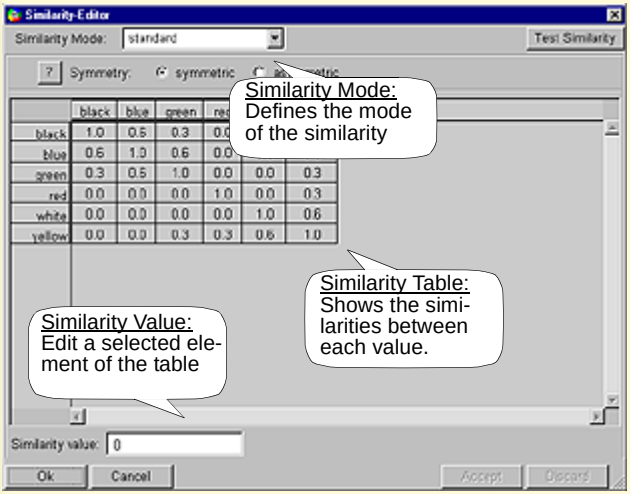


Figure 3-4: Similarity Editor using a Table

To activate the similarity you just created, select it as current similarity using the *Used Similarity* selector on the S*imilarity* Card.

3.4 Summary

Now, it is obvious that in the first and second attempt too many different aspects have been coded into the color names. A good model describes the terms and concepts of a domain on a level of detail that allows for a sufficient common understanding by most of the final system's users. This is neither a formal definition nor an operational one. Anyway, it stresses the underpinning key point: understandability.

Note: When modeling, one should always try to reveal the nature of a domain's concepts. In our used car example we recognized that the nature of a car color is not being a color but being a painting.

One could argue that there often is a well defined concept description, e.g., color can be defined as the

> "Quality or wavelength of light emitted or reflected from an object. Visible white light consists of electromagnetic radiation of various wavelengths, and if a beam is refracted through a prism, it can be spread out into a spectrum, in which the various colors correspond to different wavelengths. From long to short wavelengths (from about 700 to 400 nanometers) the colors are red, orange, yellow, green, blue, indigo, and violet."

But this is not what people have in mind when they talk about color except they are physicists.

Modeling is unveiling the thing's nature. Sometimes this nature is not as simple as it seems at the first glimpse. This is the challenge of the modeling task.

4 Enhancing Usability & Online Interface

Would you like to support different languages in your application and provide additional help for the user? If you are going to develop a commercial application with CBR-Works, the answer will be "Yes, of course!". Usability is a key feature of CBR applications and CBR-Works provides a lot of elements to support the developer as well as the users of the application.

In addition, your users may access the CBR-Works application online through a web browser. In this chapter you will learn how the basic online interfaces work that CBR-Works provides for every application

4.1 The Apartments Demo

For the following examples, please open the Apartments demo that comes with CBR-Works. The Apartments application demonstrates how to use CBR-Works for accessing its case base where you can search for apartments around Berlin by more or less detailed entered wishes for your tomorrow's home.

We will first have a look at the CBR application in CBR-Works and later switch to the standard online interfaces provided.

To illustrate the modeling concepts for usability, we will use the sample application "Apartments" that comes with CBR-Works. Please start CBR-Works Professional 4, load the application using the *File/Open*-Dialog and select *Samples/Apartments/Apartments* as shown in figure 4-1.You can specify to load the domain model, the case base and the rules independently by changing the corresponding check boxes. Please load all three parts of the application In our little example when the screen shows up the three checkboxes will be selected by default.

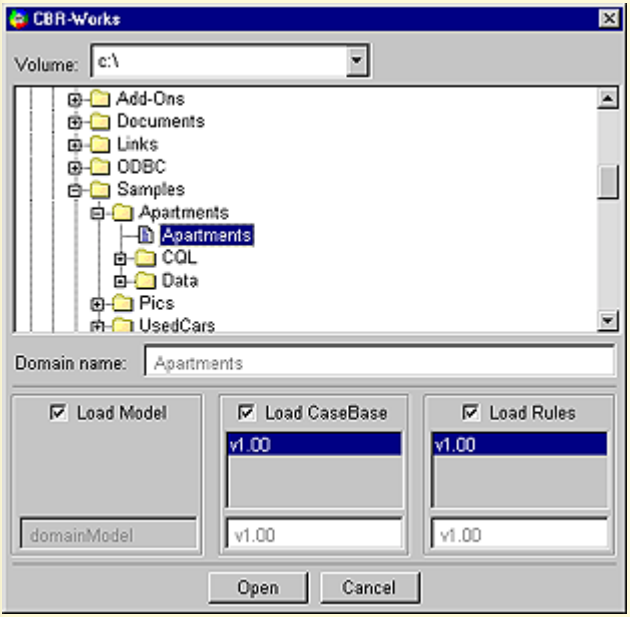


Figure 4-1: Loading the Apartments Demo in CBR-Works

To go one step further push once the *Open* button. This will lead you to a screen that looks like the following figure.

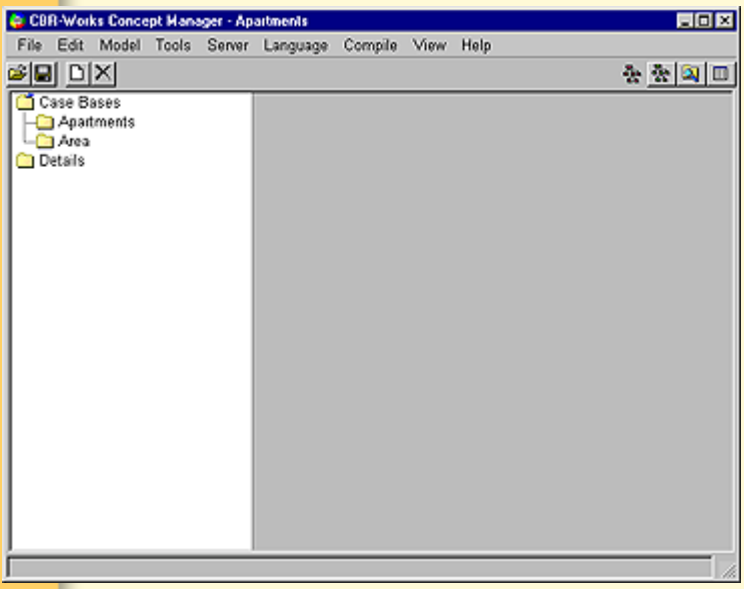


Figure 4-2: Screen after loading Model, Case Base and Rules

The application uses two different, pseudo-hierarchical case bases: the Apartments themselves and the Area in which a specific apartment is located. The background for this modeling approach is easy to understand. A real estates agent may update the case base for the available apartments frequently, but the information about the area does not change often.

Furthermore, a lot of apartments may be located in the same area. Storing the area information with the apartment would be very space consuming, requires additional work and leads to errors due to inconsistency. Therefore, the area information is referenced by the apartments, as shown in figure 4-2.

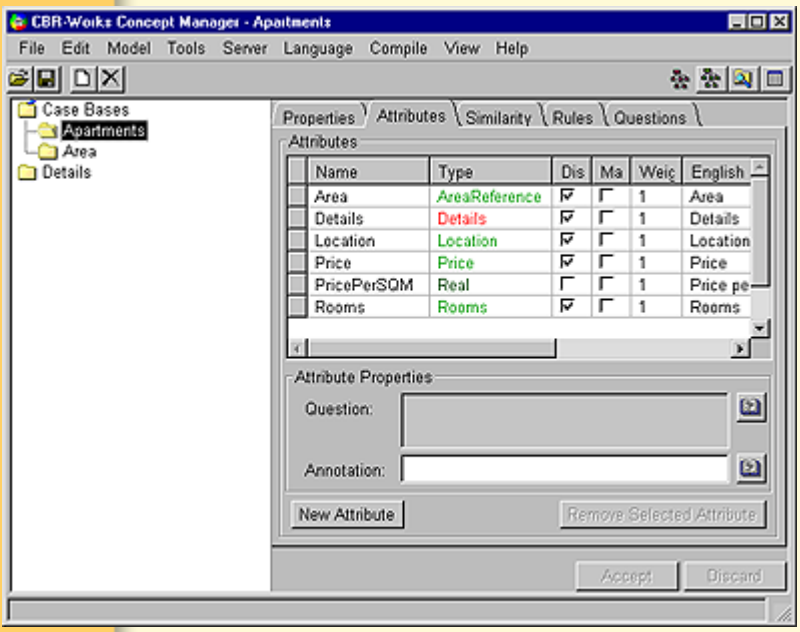


Figure 4-3: The Apartments Demo uses the Reference Type AreaReference to connect an apartment and the Area it is located in

In addition, the model uses a subconcept called Details. Whereas in Apartments the hard facts about the items are stored, Details contains additional information about parking, cable-TV, availability and so on. Please select *Details*.

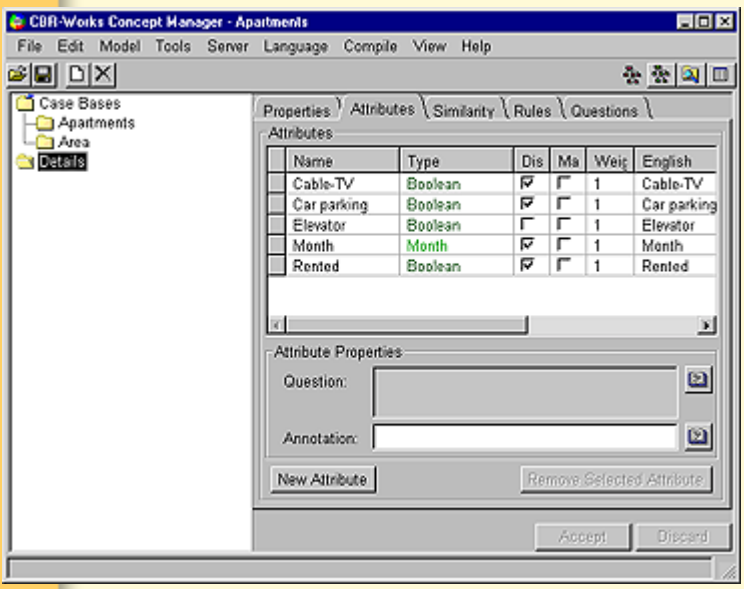


Figure 4-4: The Details Sub-Concept

4.2 Questions

By default, CBR-Works uses the standard phrase *Please specify the value for attribute...* when querying a value. Of course, this is not very intuitive and forces the developer to use self-describing names for all attributes.

CBR-Works allows to specify a verbal question for every attribute. This question is then used instead of the default phrase.

So, your application may use the question *Do you need an elevator?* instead of the default setting *Please specify the value for attribute Elevator*.

Here, in our little example the question *Do you need an elevator?* will appear automatically after selecting *Elevator* in the *Attribute Names* column.

All questions are defined in the *Question* input box of the *Concept Manager* in the *Attribute Properties* section, as shown in figure 4-5.

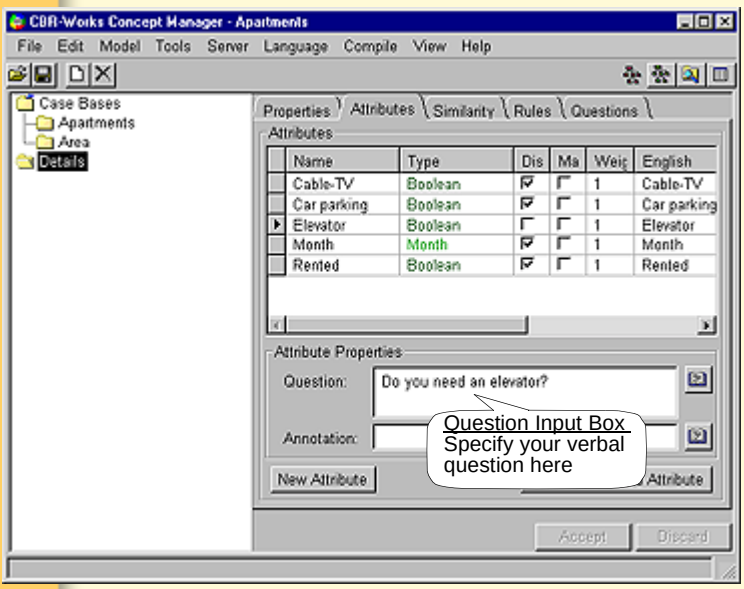


Figure 4-5: Specifying a Question for an Attribute

Note: You should always use descriptive names for attributes and concepts, even if your application makes use of questions!

Questions specified this way are not only used in the input box for a value, as shown in Chapter 2, but are also used for the *Online Query Wizard*. You will learn to use the *Wizard* later in this chapter.

4.3 Language Settings

What if you would like to deploy your CBR-Works application in different countries with different languages? Creating a bunch of independent applications would result in a lot of inconsistent case bases. CBR-Works comes with a handy feature that allows to use multiple languages that can be used at the same time.

Perhaps you have already noticed the Language menu. There, you determine the language that CBR-Works uses internally. The default System uses the language setting of your actual environment. But you may also override this parameter and force CBR-Works to use English, French, German or Spanish as the language setting.

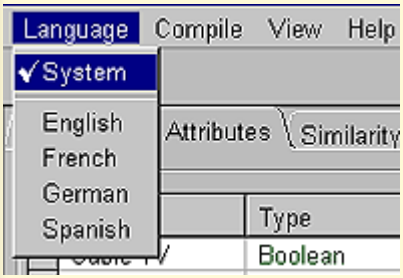


Figure 4-6: Language Settings

The settings directly correspond to information given in the Properties and Attributes sheets of the *Concept Manager*.

With the *Lexicon* icon, you can specify names and questions in the four languages independently. To use the *Lexicon* icon it is first necessary to switch from the Attributes Tab in the upper middle of your screen to the Properties Tab.

Based on the language settings, CBR-Works selects the appropriate phrase for querying and output to the user. Nevertheless, the interface of CBR-Works itself remains the same when switching the language setting.

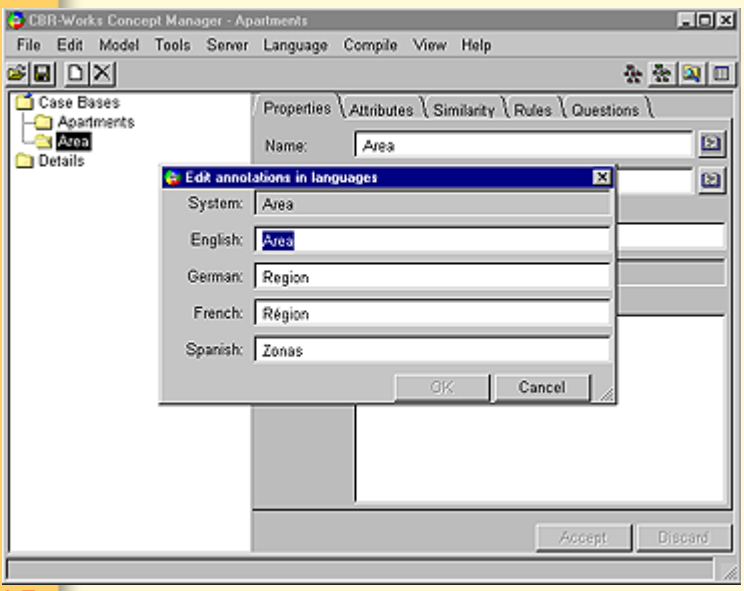


Figure 4-7: Defining the Name for a Concept in different Languages

Now have a look at how to choose different languages and questions concerning the details.

First of all select *Details* in the three view on the left side of the screen. Then change from the Properties

Tab to the Attributes Tab. To follow our little example choose the attribute *Car parking*. The following screen will appear. Maybe you wonder why you cannot see the French and Spanish column. This is because in our example we changed the size of some columns situated between the Name column and the English column from normal size to very small.

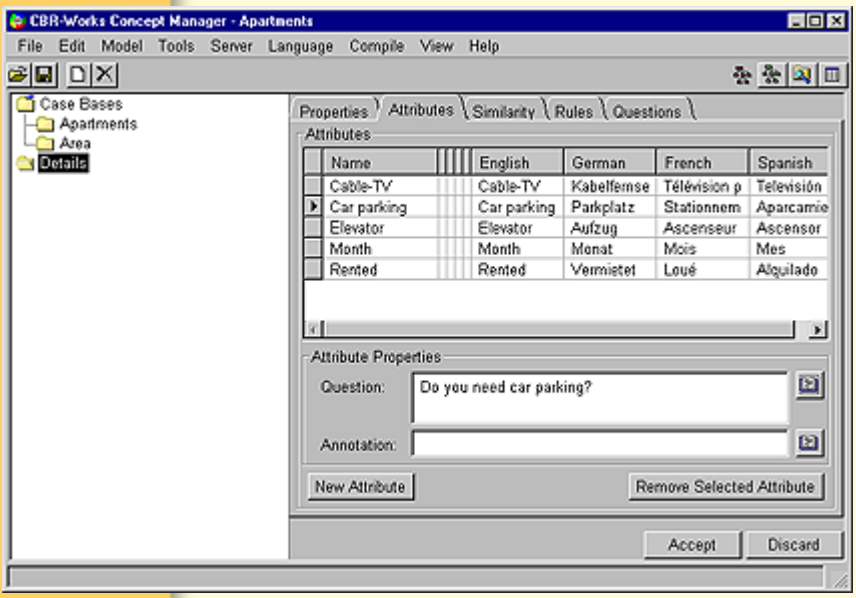


Figure 4-8: Attribute Names in different Languages

If you now push the upper *Lexicon* icon which is situated on the right side near the *Question Field* CBR-Works offers you the possibility to edit the questions in different languages.

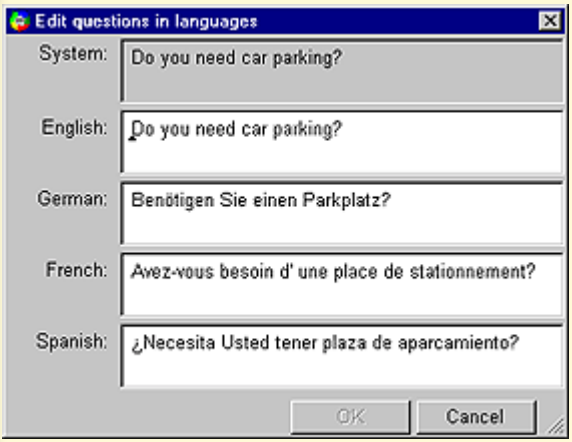


Figure 4-9: Editing Questions in different Languages

Note: In this example, English and System use the same notations. You may as well use the System language entries for a different language. The System entries are used throughout the development tool as a reference to the defined objects.

4.4 Annotations

Annotations complete the tools of CBR-Works to enhance the usability of the applications built. You may specify additional information or resources independently for every language and every attribute.

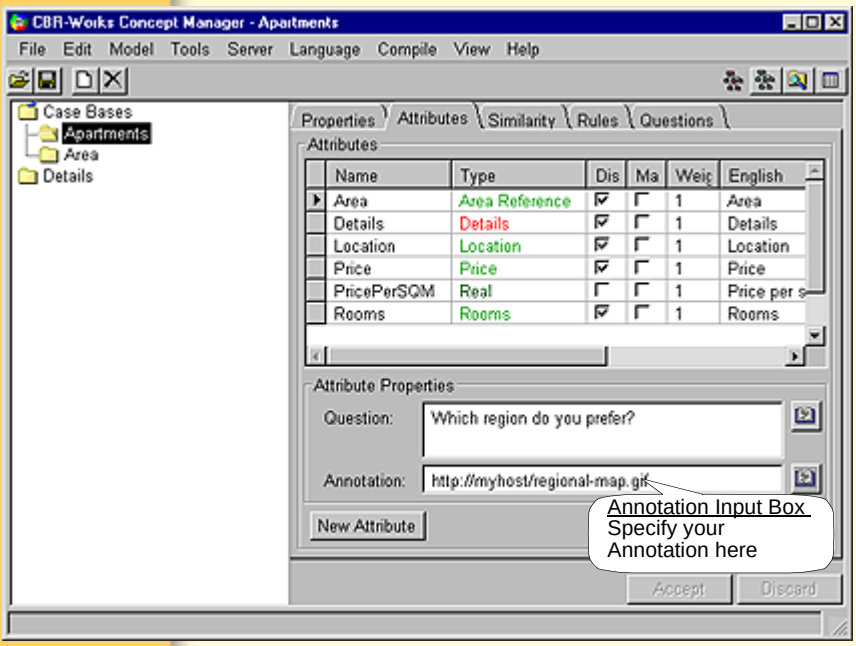


Figure 4-10: Using an URL as an Annotation Reference

By using annotations, you can build an effective help system for the user. When developing an application to be deployed on the Web, the annotations can be used to deliver special URLs to help documents, pictures explaining a certain task or any other kind of information.

Figure 4-11: Editing Annotations for different Languages

If the application is used in a new language setting, the annotations automatically switch to the corresponding entry the same way the attribute names and questions are handled.

4.5 Question Ordering

The questions specified for querying the different attribute values do not only influence the way CBR-Works displays its output for the user. Questions can be used to influence the information gathering strategy used by the application as well.

In the Questions Tab of the *Concept Manager*, you are able to specify explicitly an order for the questions defined for the discriminate attributes.

The specified order is used by the *Query Wizard* instead of other question strategies like the highest information gain. Perhaps this concept looks a bit suspicious to you at first. But it's great if, for example, your application domain uses given scripts for interviews. Often, call center staff has to follow a detailed script when dealing with customers. The concept of question ordering enables CBR-Works to

mimic thus scripts without loosing the intelligent retrieval capabilities.

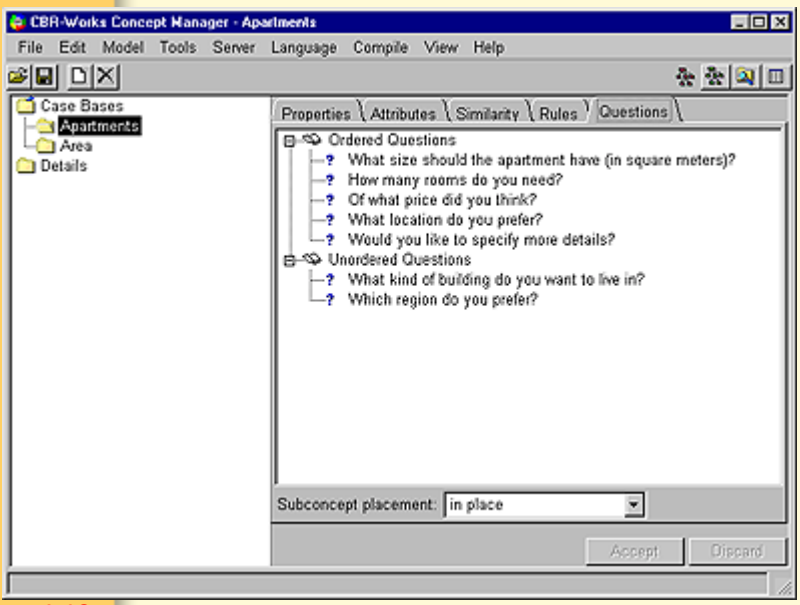


Figure 4-12: The Questions Tab allows to define an Order in which the Questions are asked

Note: Only the questions for discriminant attributes of the concept are presented. There are two kinds of questions: ordered and unordered! Only the sequence of the questions in the "ordered" section is relevant for the Query Wizard.

It’s really simple to change the order of questions or move them between the ordered an unordered section. Just use the mouse to drag and drop the ques-

tion in the correct place. The dropdown menu specifying the subconcept placement allows detailed control how the questions specified in a subconcept are inserted in the given question order.

To notice a difference in the behavior of CBR-Works, you have to ensure that the system uses Ordering as the question strategy for the *Query Wizard*. The default for CBR-Works is the information gain strategy as discussed earlier. Check the setting for the *Query Wizard* by selecting *Preferences/General Preferences* from the *File* menu (figure 4-13).

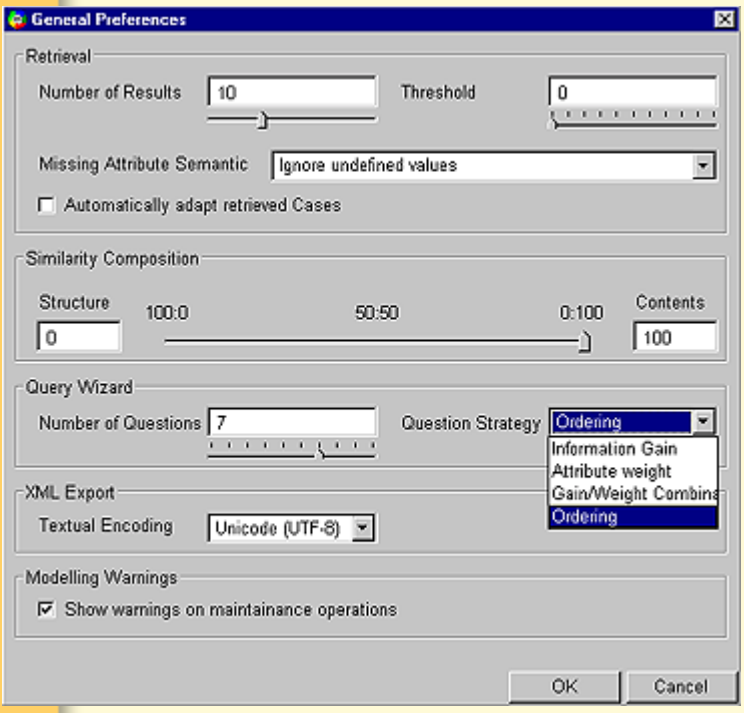


Figure 4-13: Setting the Question Strategy for the Query Wizard

4.6 Using the Application Online through the Web

Note: For testing the Online Feature of CBR-Works, TCP/IP has to be installed on your system!

The Apartments example application demonstrates how to use CBR-Works' generic Online features for a prototypical website.

You can search for apartments in several parts of Berlin online by specifying different features. It is only a simple test environment without order processing or similar things which are needed for a commercial web site.

First, we need to do some preparations: Close CBR-Works now and then start the application Apartments from the start menu of your Windows environment. It is located in the *Applications/ Apartments* submenu of the CBR-Works 4 Professional group. This automatically starts the CQL server which is used for searching for the apartments. Your screen should look like in figure 4-14.

Figure 4-14: After starting the Apartments Demo from the Start Menu

To use the Application through the web interface, please start your favorite web browser now. The Online feature can be started by using the URL

http://localhost:8051/launch/Online?qc=Apartments&rc=Apartments.

The parameters given here are necessary to determine the concept for query (qc) and for retrieval result (rc). If you enter the above link - simply select the URL and press RETURN on your keyboard - you should see a screen similar to the following figure. Please note that no additional HTML-coding was done for this example - it's the basic interface of every CBR-Works online application. Of course, you can build your own interface if you wish.

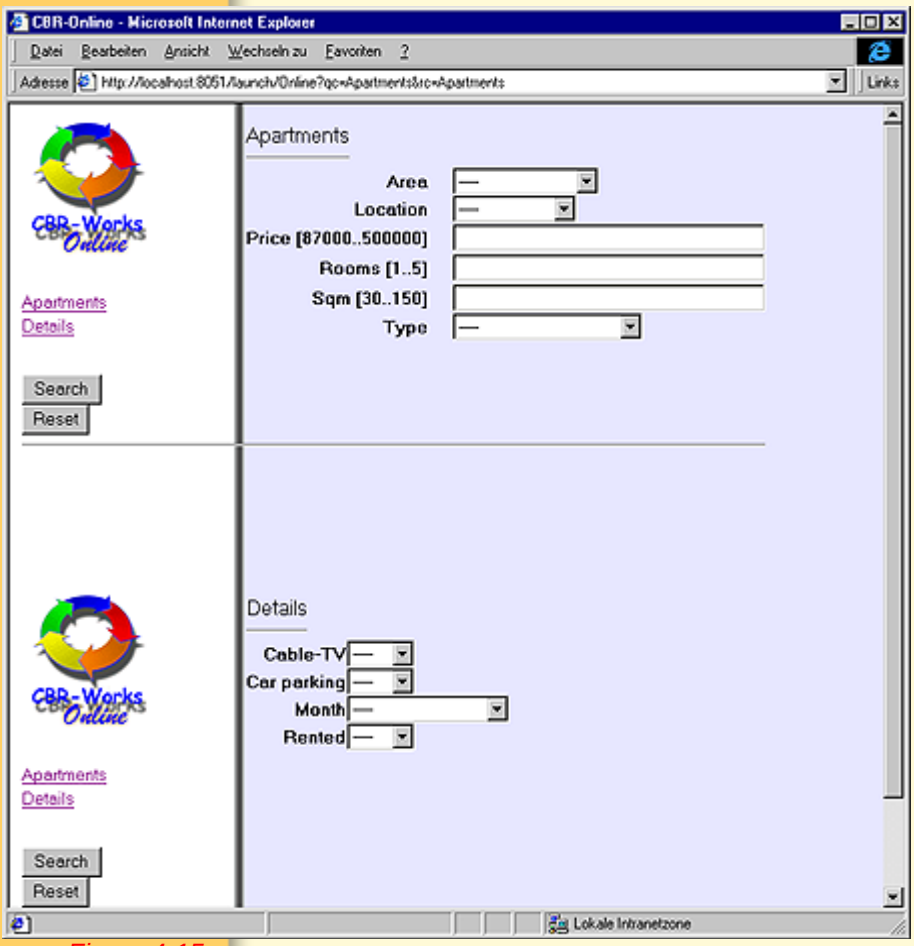


Figure 4-15: The Apartments Demo accessed through a Web Browser

In this application, the specification of the query features is done in one step. On top of this page you can define the basic features describing an apartment. You can further refine your query by using the features for more detailed information below the basic ones.

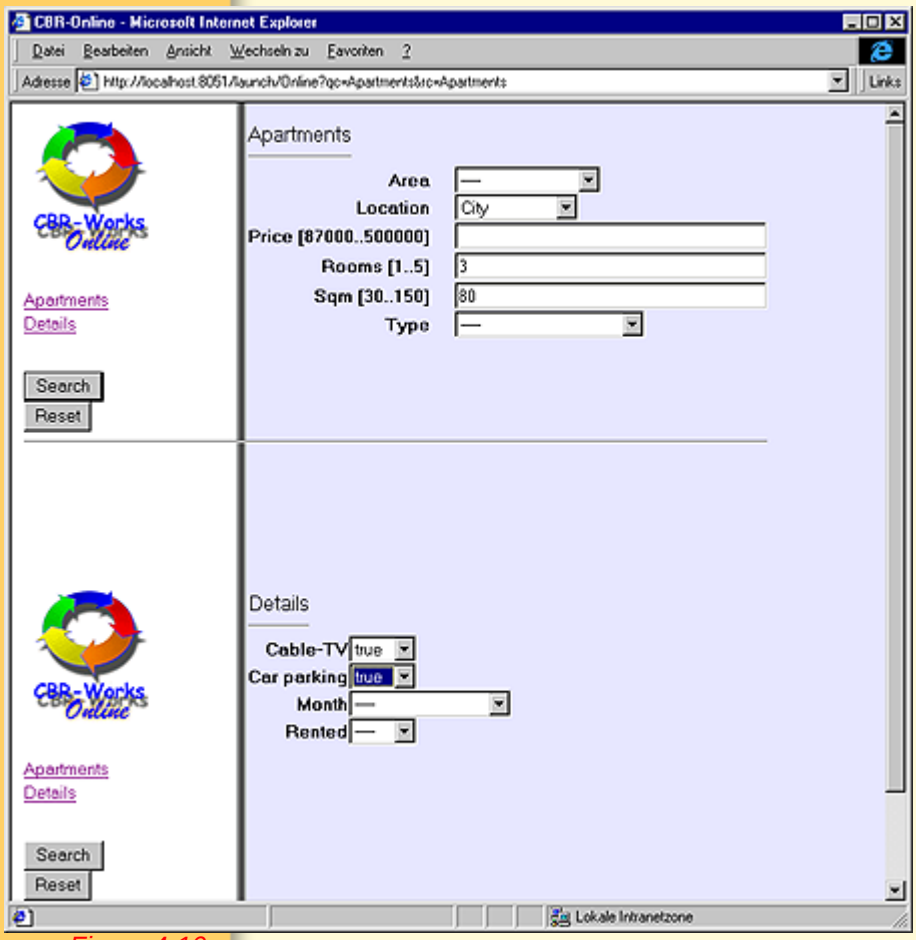


Figure 4-16: Select the Features of the Apartment

In the example shown in figure 4-16 we have set the following features: We are looking for an 3-room-apartment with a size of about 80 sqm which is located in the city. Further, our Apartment shall have Cable-TV and Car parking. Pushing the *Search* button takes us to another page.

Choose Apartments instead of Overview in the right combo box. Now the result page will appear, where we can choose of found Apartments using the left combo box in the top frame. Additionally, to have a look at the detailed information (Category) about the selected Apartments, we use the right combo box. Setting the latter to Apartments will bring up the following contents:

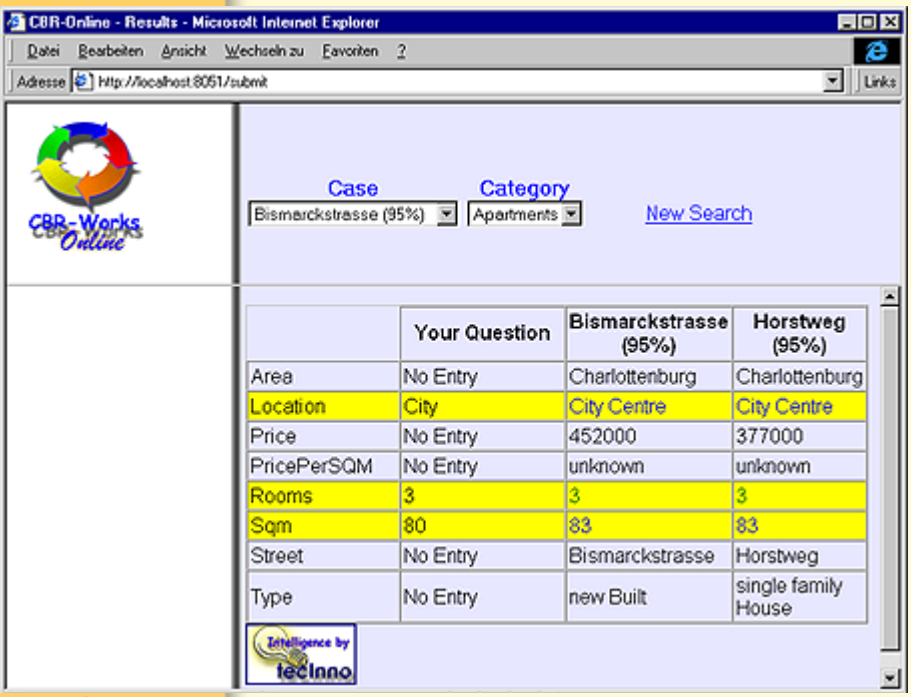


Figure 4-17: Comparing Query and the selected Case

The results page will compare the selected case and the closest alternative with the query you specified. Please note the color coding in the results page.

Of course, you may have selected another case to be compared with your query in the screen before.

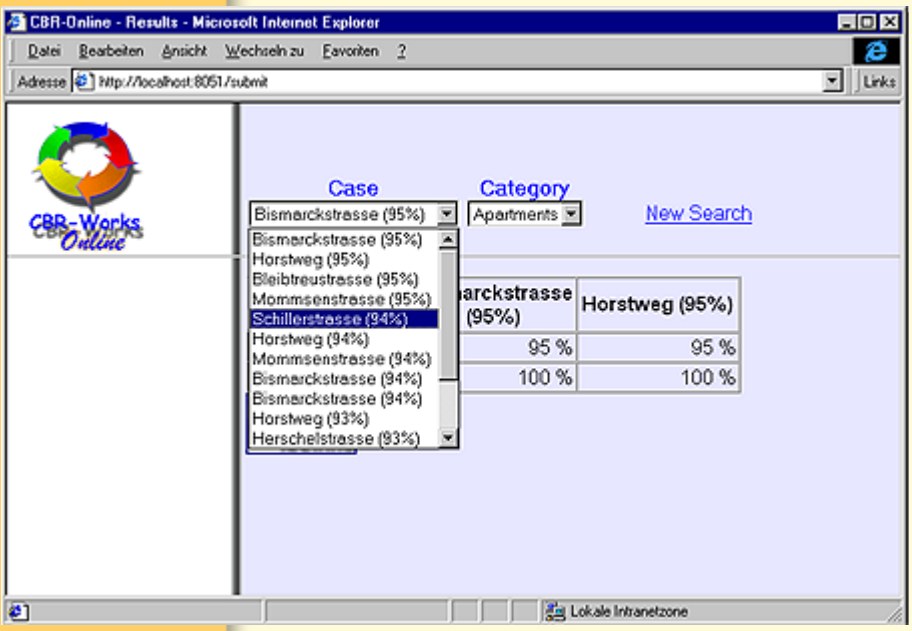


Figure 4-18: Selecting another Case

Another possibility to test a build domain is to use the Query Wizard. It can be started by using the URL:

http://localhost:8051/launch/OnlineWizard?qc=Apartments&rc=Apartments

The parameters are the same like before. If you enter the above link now - select the URL and press RETURN on your keyboard - a new browser window looking similar figure 4-19 should appear.

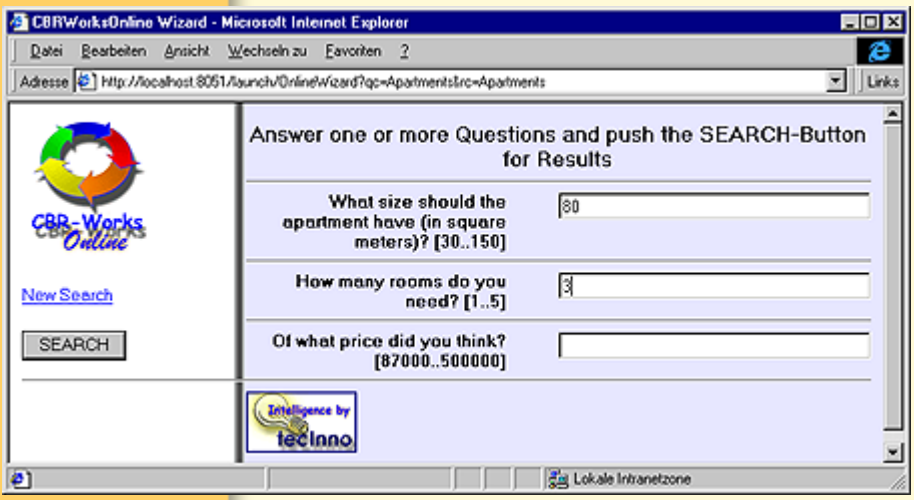


Figure 4-19: Using the Online Query Wizard

Here, the questions are presented to be answered that were specified in the application. The order is given by the strategy we defined in the application for Apartments, as seen before. Answering the first two questions and pushing the *Search* button takes us to a page which looks like the following one. To put the figure in this documentation we have shortened the result list a little bit. On your screen there will appear some more Apartment results.

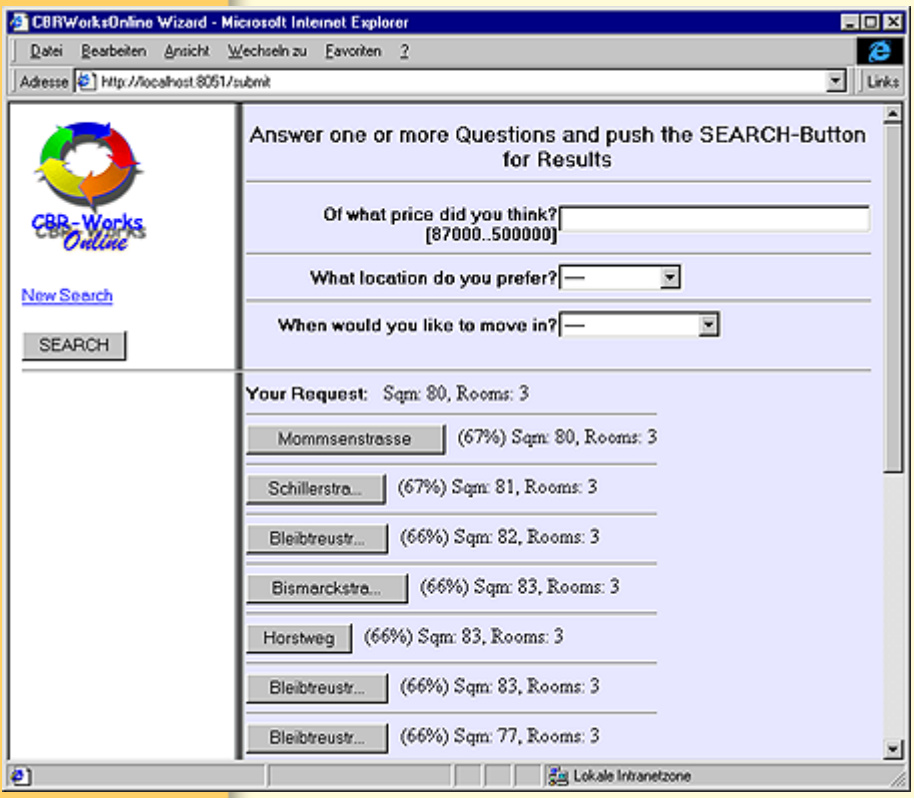


Figure 4-20: Results Page for the Online Query Wizard

Besides further questions to specify more details of the apartment, a list of matching Apartments is given. To have a look at the details of one of the Apartments, simply push the button labeled with the name of the Apartment's street. The following page will show up for the first Apartment.

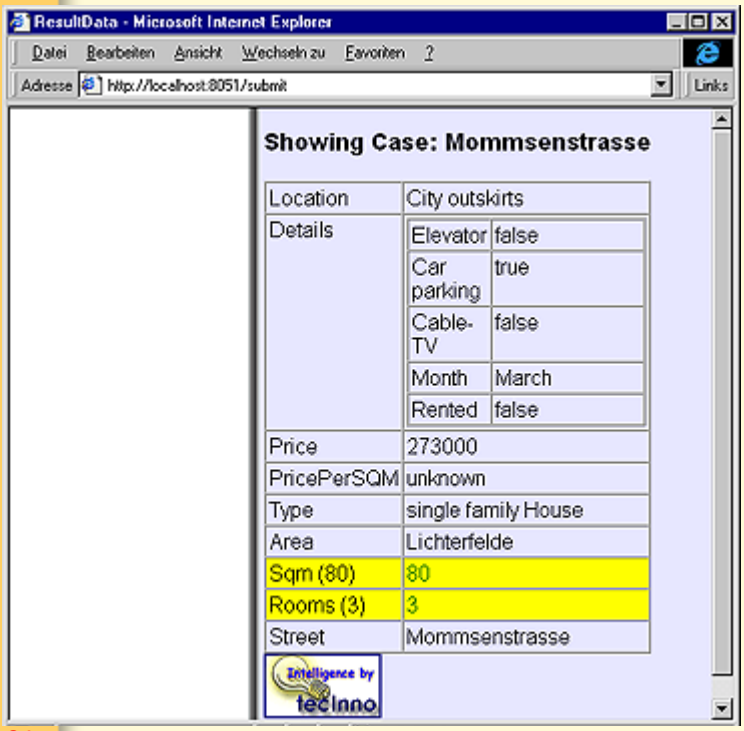


Figure 4-21: Detailed Online Presentation for a selected Case

4.7 Summary

In this chapter we have presented some elements that enhance the usability of CBR-Works applications.

Besides a multilanguage feature that enables the developer to deliver up to five localized user interfaces in the same CBR application without duplicating any cases, the software provides detailed control of the query strategy used by the system. While some

questioning strategies automatically calculate an "optimal" sequence of the questions asked by the *Query Wizard* (for example based on the information gain or the weight of the attribute), explicit question ordering restricts the application to the given sequence.

This is especially valuable if either the domain requires to mimic a specific script or if the questions should be asked in a natural, logical sequence. If the Apartments demo would ask you as a first question if you prefer to have cable-TV, but not what region you would like to live in or how many money you are able to spend, you would perhaps get somehow confused.

Another great feature of CBR-Works is its ability to make every application accessible online through a basic Web interface. You have learned how to access this functionality and how it is interconnected to the other usability features.

5 Automatic Calculation with Rules

Lets consider that the application should not only use the case data as it is, but calculate some additional information. For the Apartment demo mentioned in the previous chapter, one could calculate the price per sqm by a simple division of the attribute values for Price and Sqm when loading the case data - presuming that this attribute is not part of the original data set. CBR-Works is capable of doing exactly this by using a completion rule.

Furthermore, a second kind of rules - so called adaptation rules - are used to modify the resulting cases of a query.

5.1 The Vacation Demo

To illustrate the basic concepts, we will use the sample application *Vacations* that comes with CBR-Works. Please load the application using the *File/open* Dialog and select *Samples/Vacations/Vacations* as shown in figure 5-1.

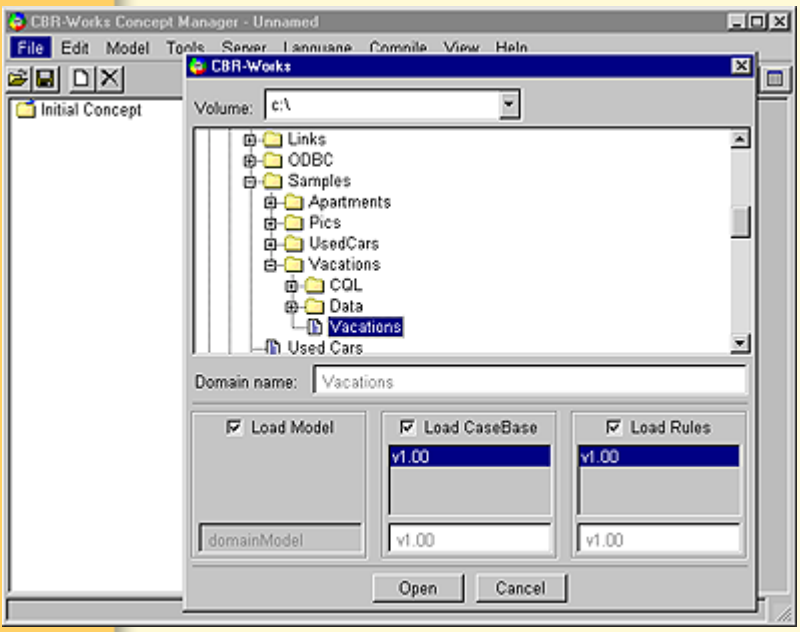


Figure 5-1: Loading the Vacation Demo

You can specify to load the domain model, the case base and the rules independently by changing the corresponding check boxes. Please load all three parts of the application. Use the *Open* button once after selecting the three check boxes (figure 5-1). This will lead you to a new screen. Select *Vacation* on the upper left side. As a consequence on the right side of the screen a table view with the vacations' attributes will appear by default. Change to the Rules Tab. You will see a screen similar to the following figure 5-2.

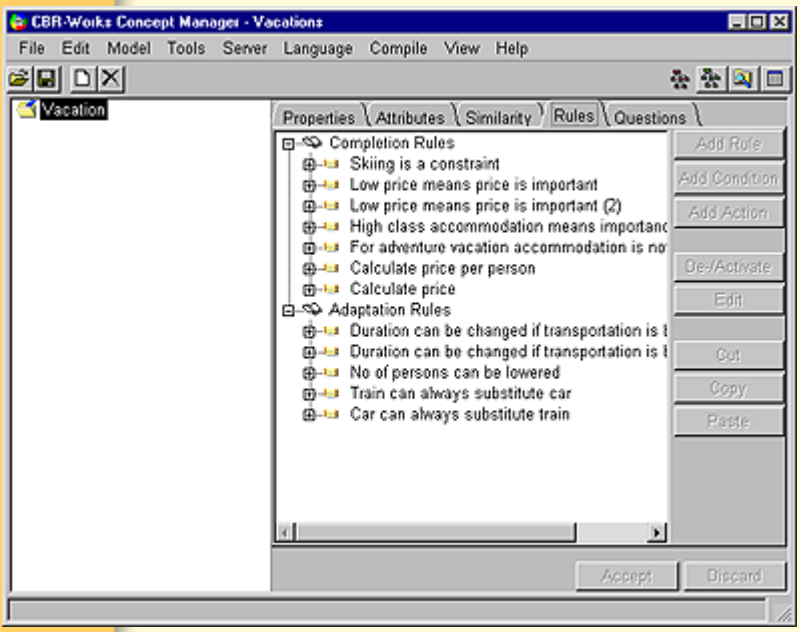


Figure 5-2: The Rules Tab of the CBR-Works Concept Manager

The *Vacations* application demonstrates how to use CBR-Works for accessing its prototypical web site where you can search for journeys all around the world by selecting your favorite options.

It operates different kinds of rules to automate some tasks either when adding new cases or during retrieval. The rules are accessed and defined through the Rules Tab of the CBR-Works *Concept Manager*.

5.2 Different Kinds of Rules

Rules can be compared to an "if-then" construction. They only execute if the condition is true. Under this circumstances the action part is performed - the rule

"fires". CBR-Works incorporates two different kinds of rules:

- Completion rules
- Adaptation rules

Completion rules are used to complete a query or a case from the case base. They are executed while entering the values of attributes of a case, and, in case of retrieval, after all possible adaptation rules have fired. Some actions (i.e., weights, filter and similarity) are executed only for the current query and not for the cases in the case base. Adaptation rules can be used to derive a new result case from the query and retrieved cases. The main differences to a completion rule are the access to attribute values and the number of actions.

Completion rules are executed after two events: The obvious event is at the retrieval of a case from the case base. After each input of the user the rule system checks its rules. The second event is after executing the adaptation rules. In this case only the rule base for the result case is executed. Adaptation rules are executed during a consultation session after the retrieval of cases. A result case is created by copying the retrieved case. All actions of the adaptation rules now refer to this result case rather than acting on the retrieved case or the query.

5.3 Defining Rules

All rules consist of two parts:

- The Condition that must be fulfilled for the rule to fire.
- The Action that is performed by the rule.

In the next figure you can see that multiple conditions and actions for a rule may be defined. The Rules Tab displays conditions with a leading question mark, whereas actions are preceded by an exclamation mark

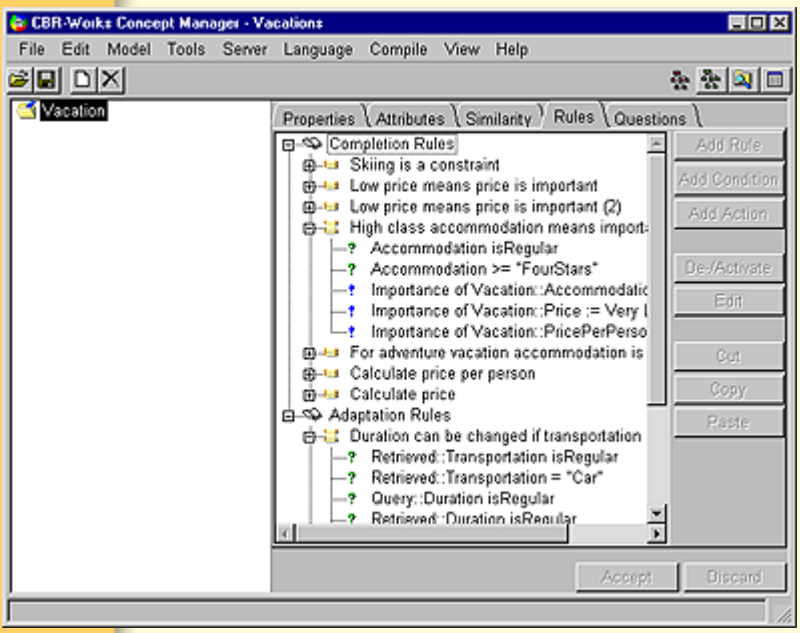


Figure 5-3: Multiple Conditions and Actions defining a Rule

To define a new rule, select the insertion point (completion or adaptation rule) and execute *Add Rule*. CBR-Works asks you for the name of the rule. Every rule is referenced by its name - so choose a descriptive one! You may rename existing rules as well, using the *Rename* command.

Figure 5-4: Adding and naming a new Rule

The rule is inserted without any condition or action parts. Now, after selecting the rule you can add conditions and actions according to your needs. The following figures show a typical rule that uses a calculation that is performed with a variable in the condition part, and the editors for defining conditions and actions in CBR-Works.

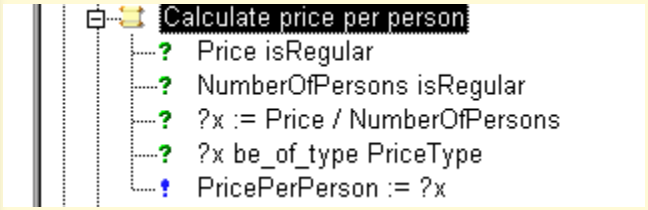


Figure 5-5: Rule incorporating the Calculation of a Variable

For some calculations it is necessary to save the results in an auxiliary variable. For this purpose, any value can be assigned to variables, which are only visible in the defined rule. It is impossible to share variables with other rules. Variables are represented by their name preceded by a question mark, e.g., **?x** represents a variable named **x**.

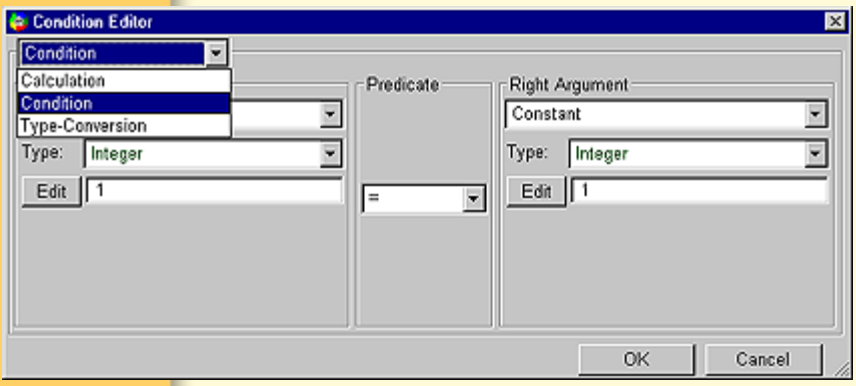


Figure 5-6: Condition Editor

The *Condition Editor* can be used to specify conditions and for performing calculations and type-conversions as well.

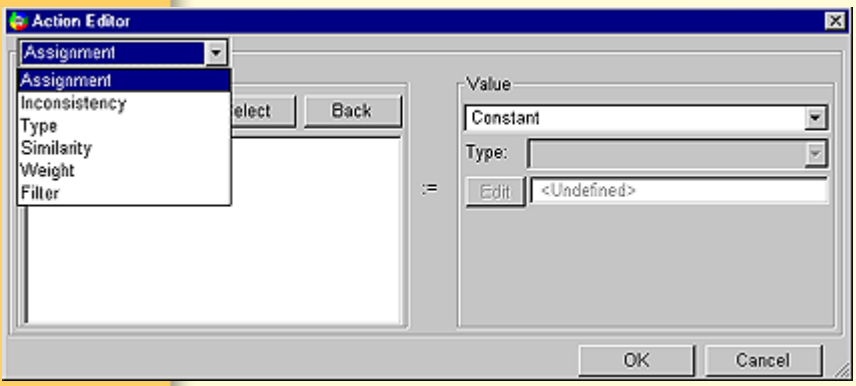


Figure 5-7: Action Editor

The *Action Editor* allows to specify different kinds of action to perform. Try modifying and extending the given rules system by yourself. The Reference Manual gives you more detailed information about the different types of conditions and actions used by CBR-Works.

5.4 Controlling the Execution of Rules

You may as well deactivate (or activate) a single rule. A rule which is deactivated will not work, even if all conditions are fulfilled.

Note: Deactivated Rules are displayed as "crossed out". In this Case the *Deactivate* button changes from Inactivate to Activate.

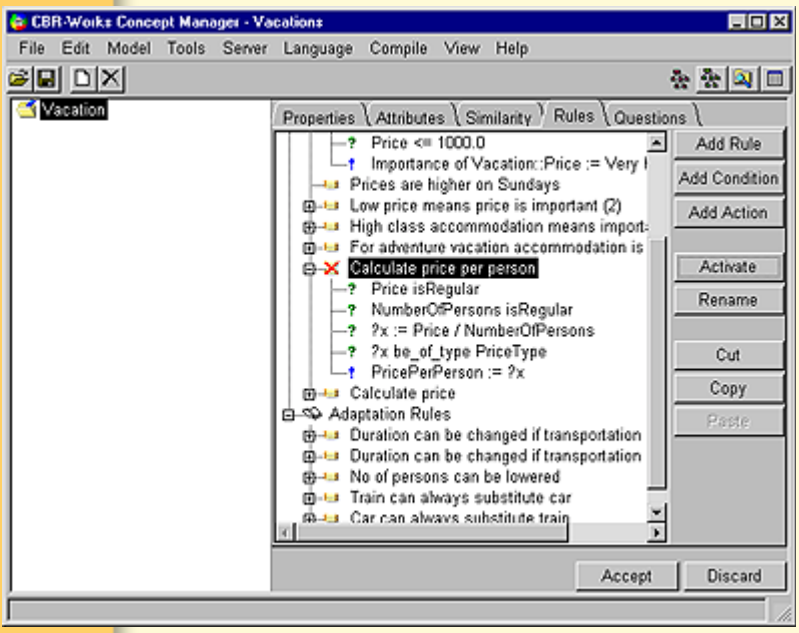


Figure 5-8: Deactivating Rules

But you may specify the overall behavior of CBR-Works regarding to the rules system. Choose *Preferences/Rules* from the *File* menu to control, how rules are executed. Using rules without caution may result in unpredictable behavior of your CBR application. Rules can overwrite attribute values and lead to calculation errors during execution.

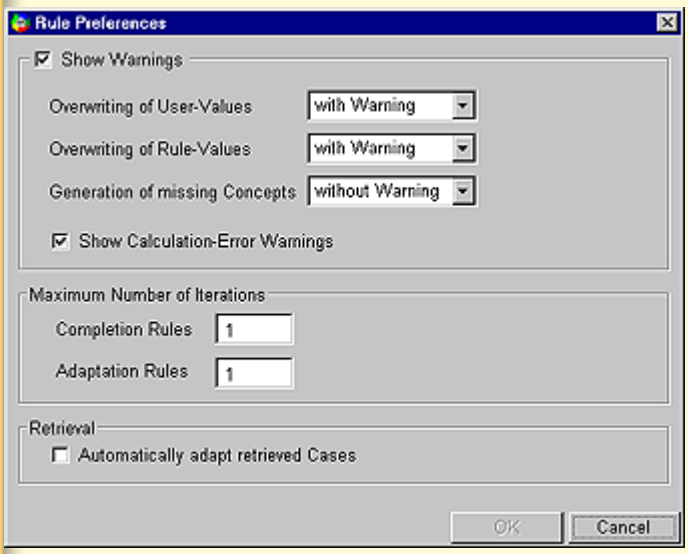


Figure 5-9: Setting the Rules Preferences in CBR-Works

In theory, an endless loop could occur in which rules interact with each other. CBR-Works helps to control this problems by limiting the number of iterations a completion or adaptation rule is allowed to fire.

5.5 Summary

The rule system is a very powerful tool to complete cases. Every rule is divided into two parts: the conditions and the actions. If all conditions of a rule are fulfilled the system executes the defined actions.

Two different kinds of rules are implemented in CBR-Works: completion and adaptation rules. Building a consistent rule system is a complex task. For more information about using rules, please consult the reference manual.

6 Some Remarks on Application Development

To develop a CBR application from scratch, we encourage you to use the **INRECA** Methodology. This approach defines a straightforward process that incorporates the following steps:

1. Define the case structure
2. Collect cases
3. Define the importance of features
4. Define relationships
5. Define the similarities
6. Build the user-interface
7. Integrate the application into existing systems
8. Refine the CBR application

Based on this approach, developing an application with CBR-Works leads to four main tasks, as shown in figure 6-1.

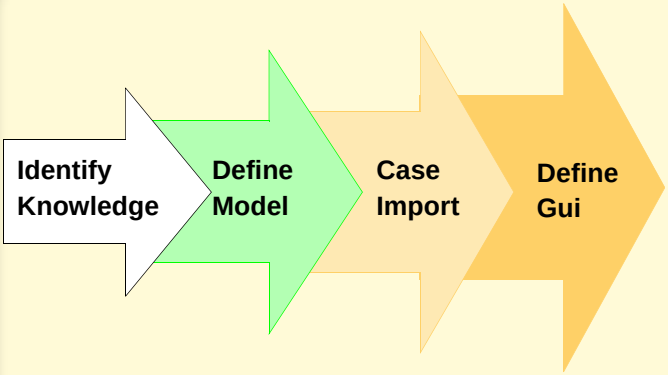


Figure 6-1: Developing a CBR Application

6.1 Identify Knowledge

First, the real world domain must be analyzed to identify the knowledge and the structure of cases. Remember the color definition problem mentioned earlier - often, the first idea is not the best one because knowledge and meaning is modeled into an attribute!

6.2 Define Domain Model

Based on this analysis, defining the domain model stands for identifying types, objects, components, specializations, similarities and rules used in the application. The Apartments demo presented a few examples for using hierarchical case bases and subconcepts. By specifying the types for your attributes, you are determining the clearness and usability of your application as well - do not limit yourself to the predefined types of CBR-Works!

During this modeling task, you may keep in mind that existing databases might be used as a resource.

6.3 Case Import

Therefore, the next step - importing the necessary cases - may incorporate a database import and/or manual and automatic case authoring. Completion rules for example can help to calculate attributes that are not stored in the database explicitly.

6.4 Build GUI

The last task when developing a CBR-Works application consists of building an easy-to-use interface. During the tutorial you have not only learned the basic concepts of a web interface for a CBR-Works application, but noticed the importance of the language settings and the possibility of defining different questions, names and annotations for the language concepts System, English, German, French and Spanish.

7 From Data Bases to intelligent Web Applications

In the previous chapter, we have already indicated that most CBR applications are not developed completely from scratch. Instead, existing databases are often used as a starting point for the new tool. A major advantage of CBR-Works is the ability to import existing data and therefore greatly enhance an existing "thumb" database with the intelligent retrieval of a CBR application.

To do so, the development process presented in the previous section is slightly adapted as shown in figure 7-1.

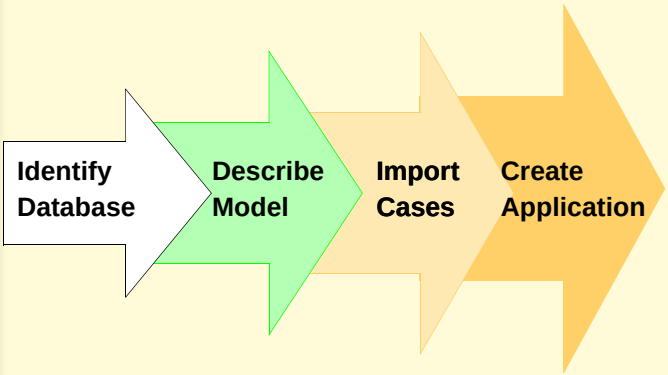


Figure 7-1: Application Development using an existing Database

7.1 Identify Database

Due to the build-in ODBC interface, CBR Works enables you to use a variety of data sources - not only existing databases. You may access databases like MS Access and dBase, spreadsheets like MS Excel, text files and SQL server. ODBC provides a flexible interface to the different kinds of data sources.

Note: To establish an ODBC connection, you have to ensure that the correct ODBC drivers are installed on your system. You'll find the necessary files in the ODBC directory of the CBR-Works installation.

For establishing a new ODBC connection, first start the ODBC data source administration tool of your windows system (double-click on the icon). Please have a look at figure 7-2.

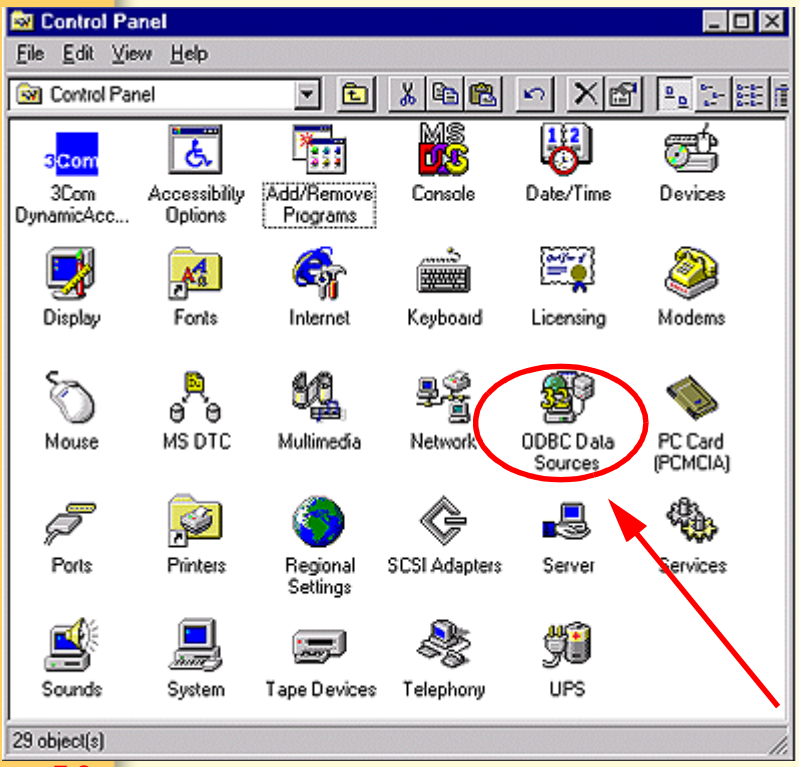


Figure 7-2: Data Source Administration Tool

This will lead you to following figure 7-3.

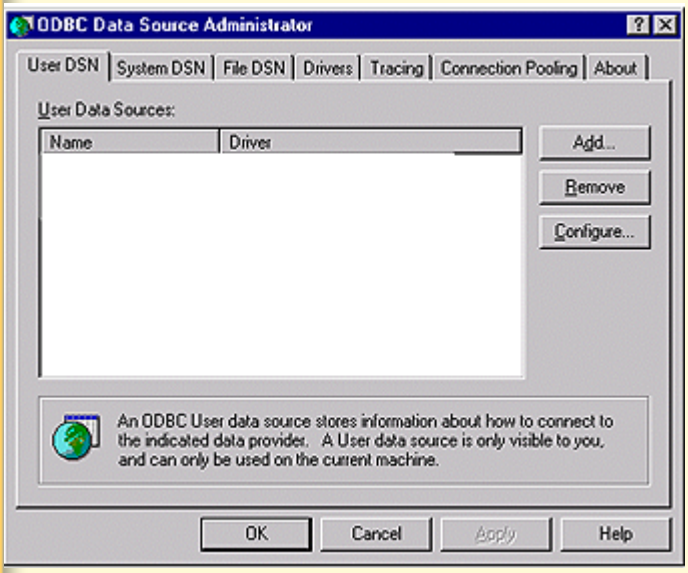


Figure 7-3: Adding a new ODBC Connection

If you now press the *Add* button a collection of available drivers will appear. Select the appropriate one as shown in figure 7-4. Use the *Finish* button.

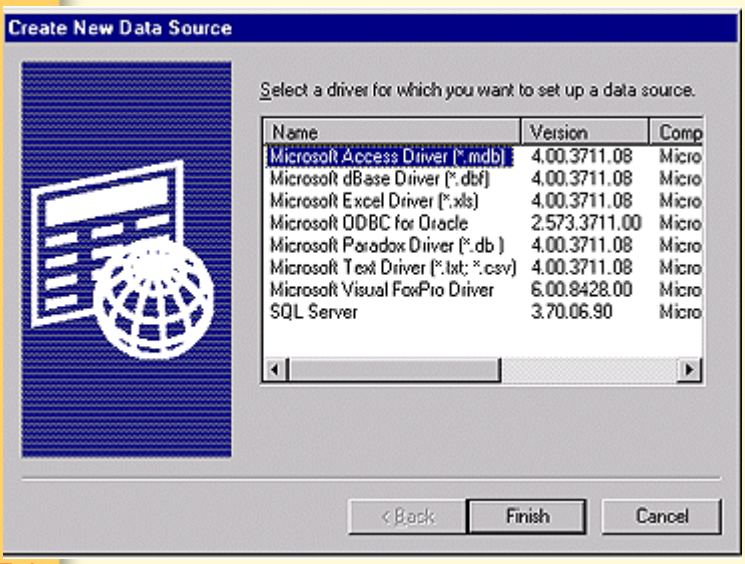


Figure 7-4: Driver Selection

In the second step, the setup of the new ODBC source must be completed. This part requires to name the connection and to select the underlying file that contains the data to be accessed (here: **C:/CBR-Works/DB/travel.mdb**). Please pay attention to the fact that your own file address is probably completely different. You may as well provide a description for the data source. After completing this step, the ODBC source is accessible from CBR-Works.

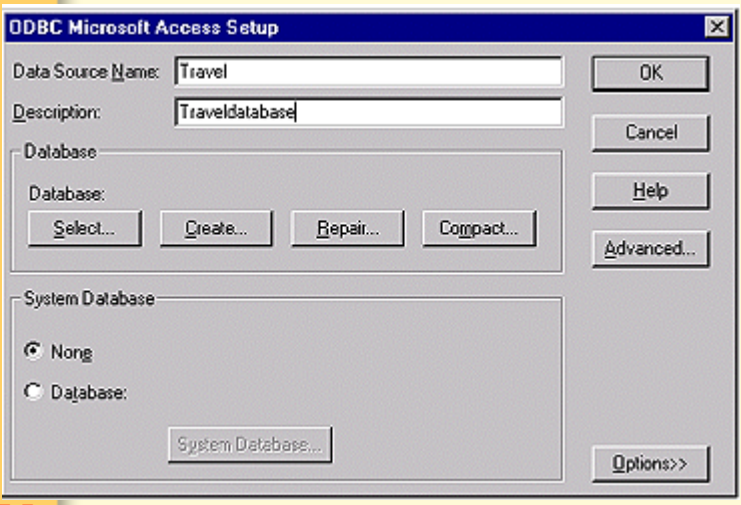


Figure 7-5: Setting up the ODBC Source

Now, switch to CBR-Works and establish the connection to the new ODBC source.

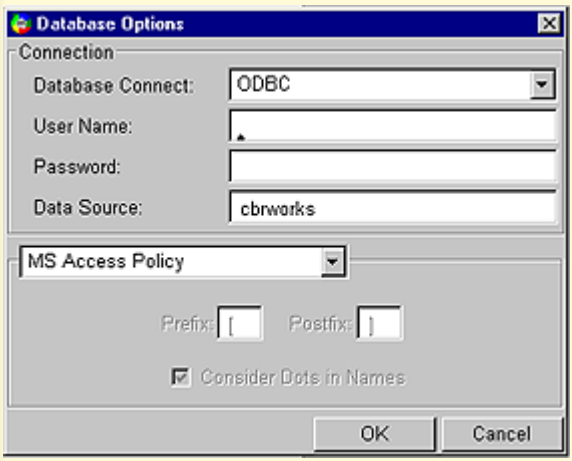


Figure 7-6: Establishing the ODBC connection in CBR-Works

You will see figure 7-6 if you select *Preferences/ Database* from the *File* menu after beginning a new

CBR-Works project. Now, CBR-Works is able to import the cases from your data source via ODBC.

7.2 Describe Model

The domain model consists of concepts and their attributes. These attributes may either be complex ones that consist of sub-concepts themselves, or are indivisible and correspond to a, often user-defined, type. To create a model from the ODBC source, you have to:

- first create the concepts
- then define and assign the corresponding types

7.2.1 Creating Concepts from an ODBC Source

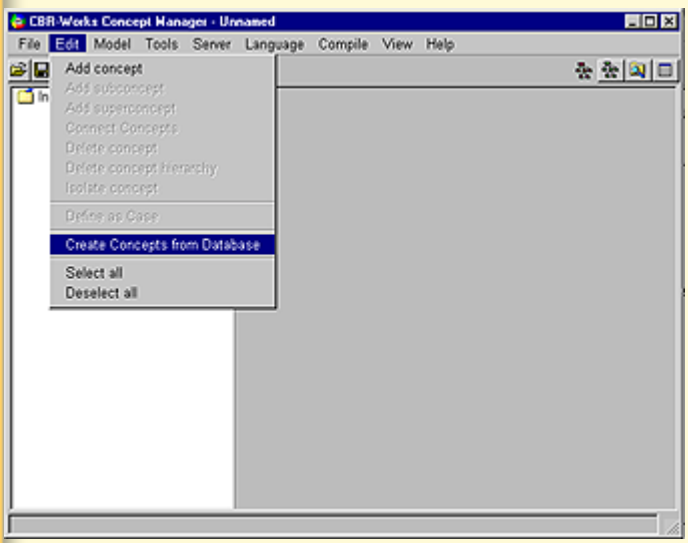


Figure 7-7: Starting the Concept Wizard

To automatically create the concepts, choose *Create Concepts from Database* in the *Edit*-Menu of the *Concept Manager* as shown in figure 7-7.

This will start the *Concept Wizard*. First the *Concept Wizard* will appear empty.

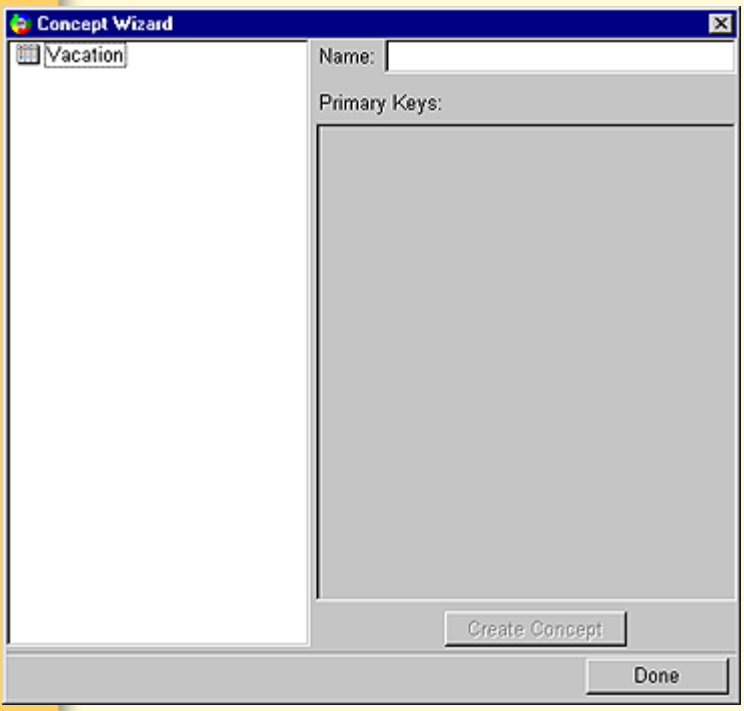


Figure 7-8: Empty Concept Wizard

Next, choose the table and *Primary Key* of the data source. The *Primary Key* consists of one or more attributes and uniquely identifies exact one record, or row, of the table. A table view on the ride hand side of the screen will show up after selecting *Vacation*.

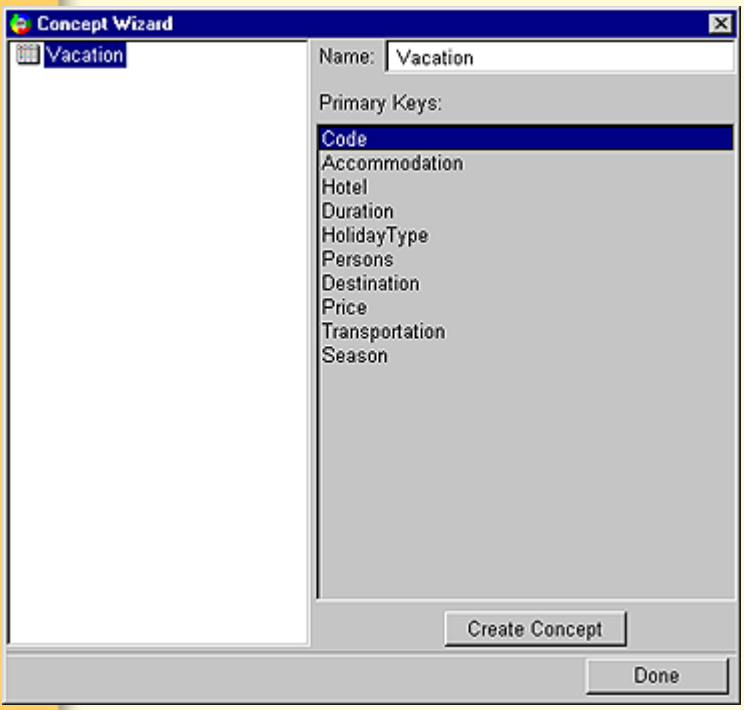


Figure 7-9: Choosing Table and Primary Key

After this selection, pushing the *Create Concept button* tells the *Concept Wizard* to calculate the appropriate concepts for the table. The next screen will appear nearly empty. You will notice that the table section on the right side of the screen is gone. In the three view on the left hand side *Vacation* is marked red. Press now the *Done* button at the bottom of the screen to step forward to figure 7-10.

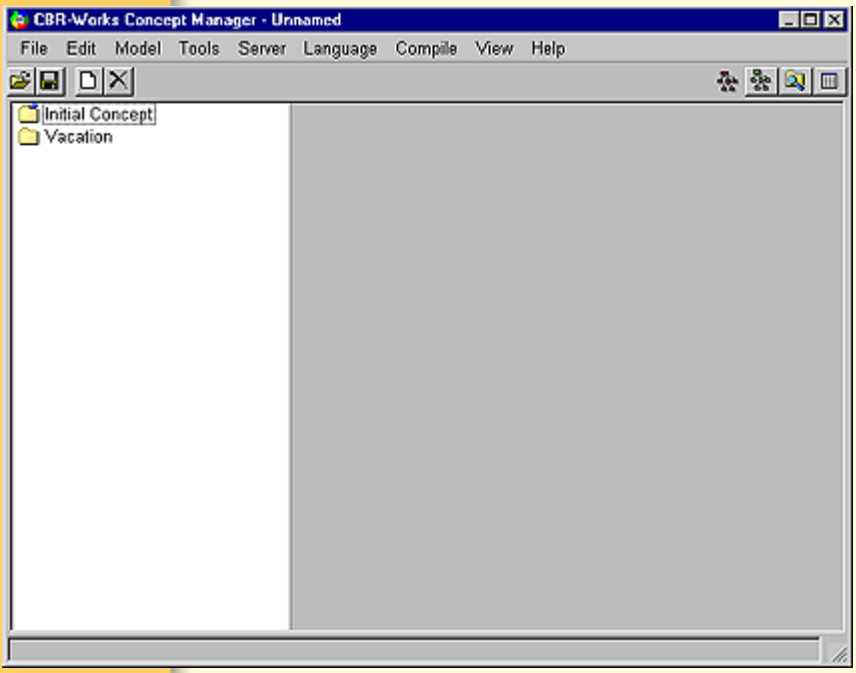


Figure 7-10: Screen before defining the Concept

Next, define the concept that should be used as the overall case concept. Select the concept first and choose *Define as Case* from the *Edit* menu (figure 7-11). Now, every case in the case base to be build complies with one record of the data source and has the structure defined by the case concept.

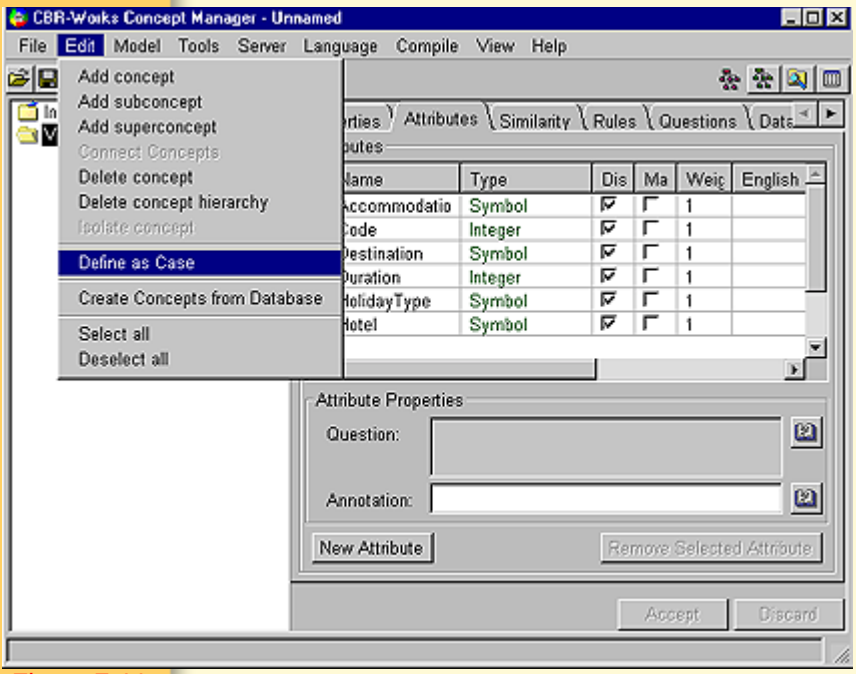


Figure 7-11: Defining the Concept as Case

7.2.2 Defining Types from the Data Source

Stepping forward, we have to create suitable types for the attributes defined from the data source. The *Type Wizard* interacts with you to define new types based on the attribute values given in the data source. To start the *Type Wizard*, select *Create Type from Database* in the *Edit* menu of the *Type Manager*.

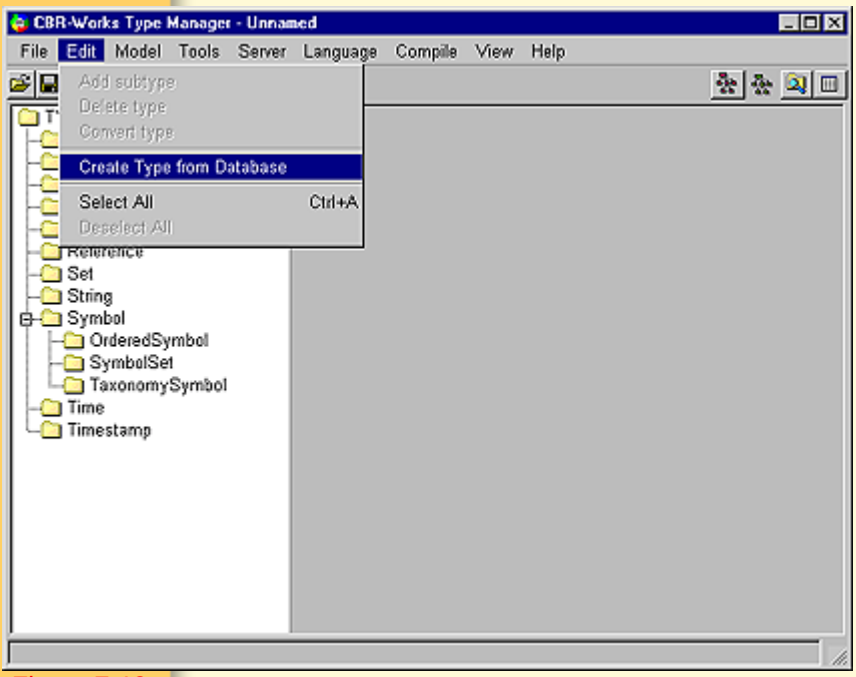


Figure 7-12: Starting the Types Wizard

The CBR-Works *Type Wizard* helps you to create a suitable CBR-type for every column in the table, i. e., every attribute in the case concept specified. To do so, the *Wizard* looks at all values given and provides help in defining the optimal type.

Note: Do not try to work with the standard types, like **symbol**, **date** or **string**, but define suitable *subtypes* wherever possible. You greatly enhance the usability of the application for the users and the developers by doing so!

First you will see an empty *Type Wizard.*

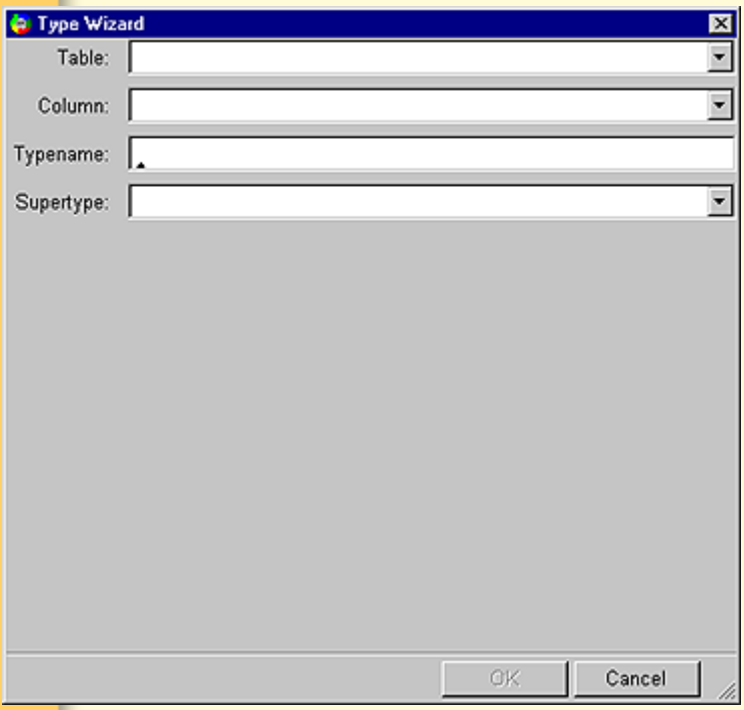


Figure 7-13: Empty Type Wizard

To follow our little example select the drop-down *Table Field* and choose the table named *Vacation*. All fields are then filled automatically. You have the possibility to change the setting according to your preferences. In our example we changed the *Column Field* to *Accommodation* (figure 7-14). Again the other fields will change automatically, always depending on what you decided to choose in the *Column Field*.

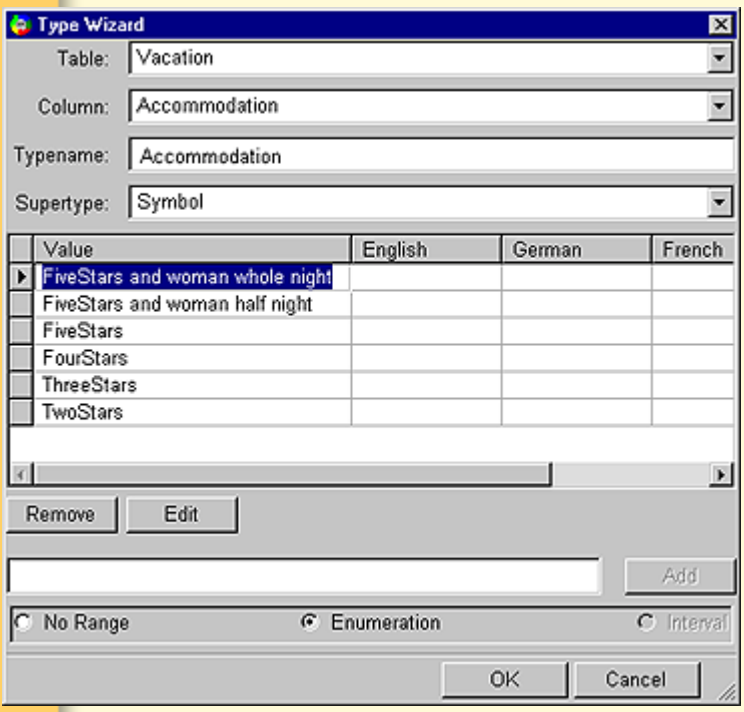


Figure 7-14: Creating a User-Defined Type with the Type Wizard

Last but not least, you have to assign the new defined types to the attributes of the concept. You have already seen how to do this: Select the concept in the *Concept Manager* and assign the correct type for an attribute with the combo-box in the Attributes Tab, as shown in the next figure 7-15.

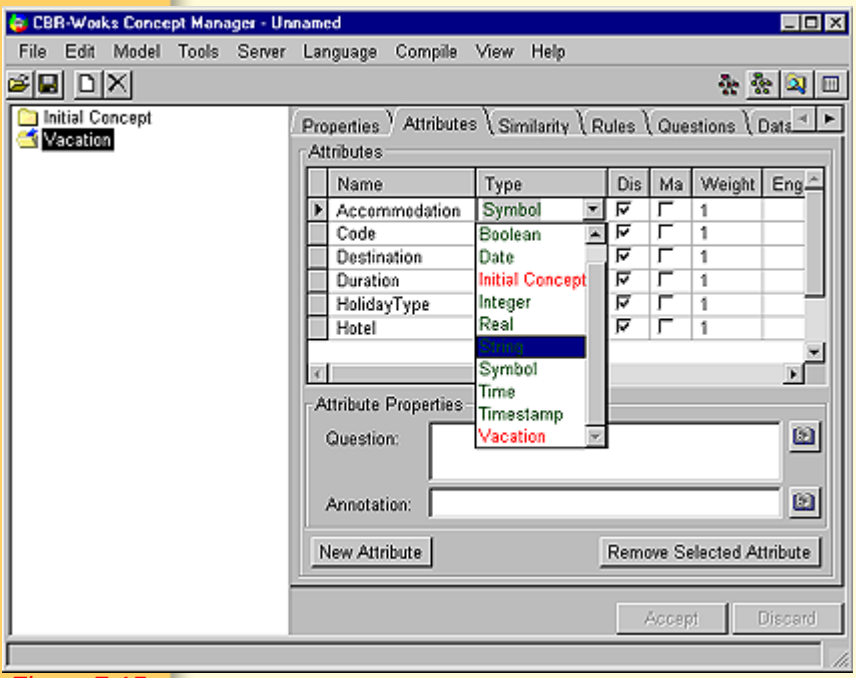


Figure 7-15: Choosing the newly defined Types for the Attributes

After assigning all types correctly, CBR-Works now knows the complete data model corresponding to the data source given. Next, we can import the case data.

7.3 Import Cases

Our new case base is empty until now. We just have defined the concepts, attributes and types that together form the model of our domain. Next, we have to import the contents of the data source into the case base. Every case consists of a single record in the source, like a row of a table.

To import the case data, please change to the *Case Explorer*. Now select the case itself. Then choose

Add case from Database from the *Edit* menu of the *Case Explorer*.

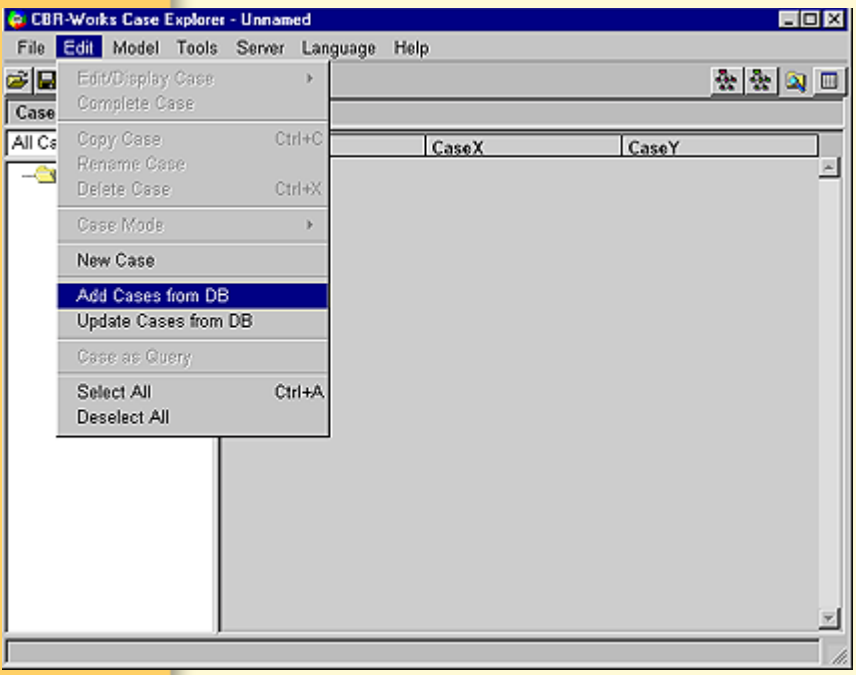


Figure 7-16: Starting the Import Process

Next, you have to choose the attribute defining the name of a case (figure 7-17). This parameter is necessary because cases do not only contain data, but are referenced by their names as well. If you are not sure which attribute is suitable here, the *Use Primary Key* function generates the case names from the primary key discussed earlier.

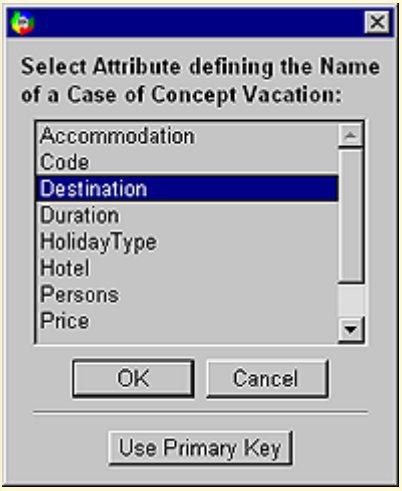


Figure 7-17: Selecting the Attribute defining the Name of a Case

Now, the records stored in the data source accessed through the ODBC connection are automatically imported and stored in the case base defined. After completion, the case base defined only from the data source replaces the old data-base and is ready for querying and retrieval.

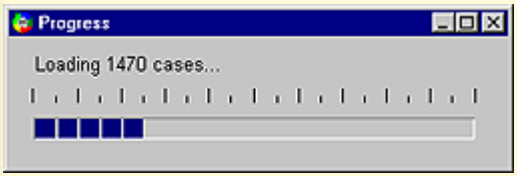


Figure 7-18: Importing the Case Data

7.4 Create Application

You may now add more information to the domain model, define questions and similarities, insert rules and so on. We have explained you the details to these steps throughout the tutorial.

If the newly created application should be available with all the case data online for query through an automatically started Web server, a corresponding Web server has to be created and the application must be deployed. To create a Web server, select *Server Console* from the *Server* menu.

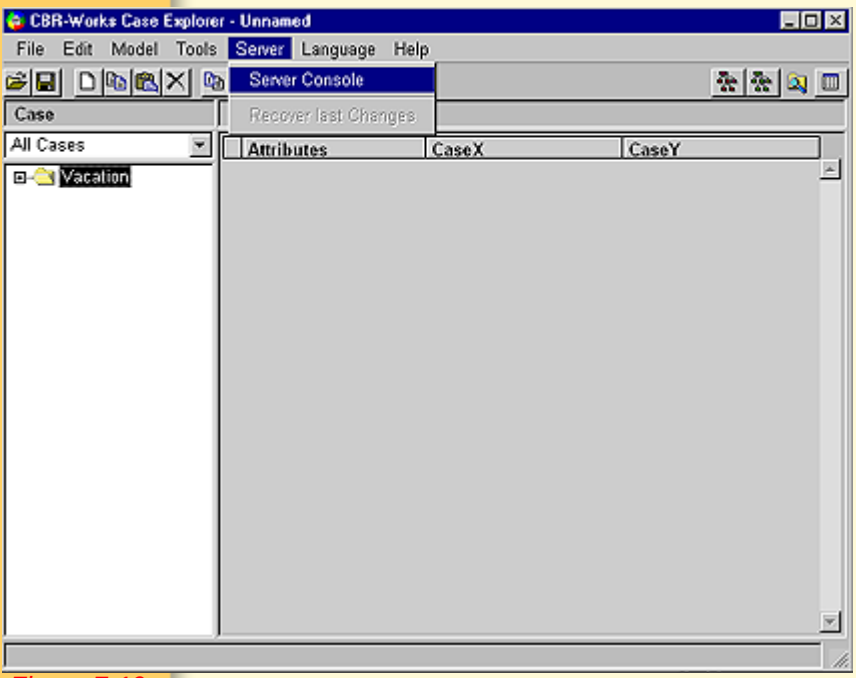


Figure 7-19: Starting the Server Console

Next, the *CBR-Works Server Console* appears and you may start creating a Web server for your application. To do so press the *Create Server* button.

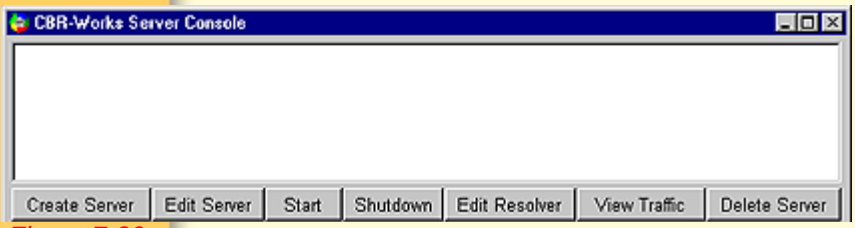


Figure 7-20: CBR-Works Server Console

To complete the server creation process, you have to define the server type (here: *TinyHttpServer*), the hostname (here: *localhost*) and the port used to access the server (here: *8008*). Be sure to enter a port address that is not used by another system or the system itself! Finish the task by selecting *Start server on system start up* and close the dialog with *Create and Start* (figure 7-21).

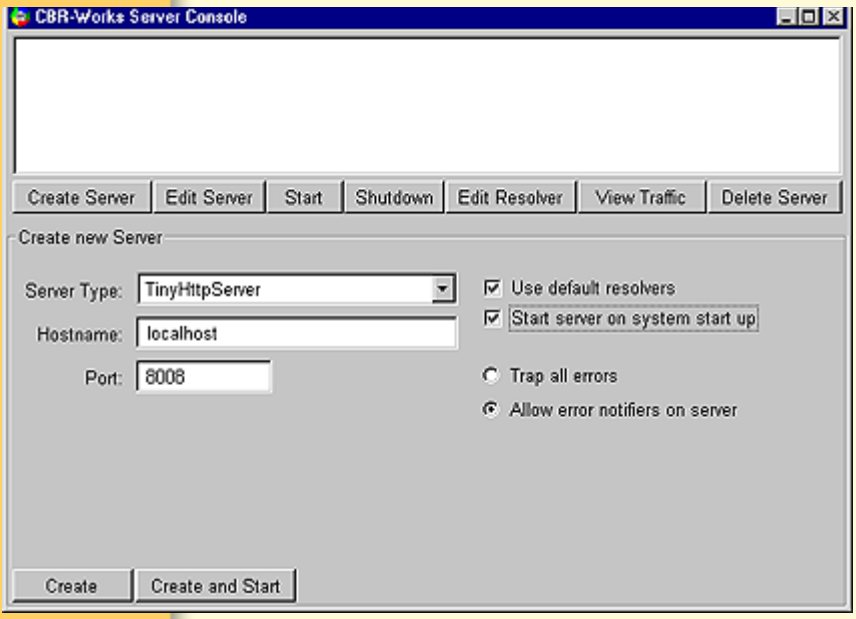


Figure 7-21: Specifying the Parameters for creating a Web Server

Now, your application is accessible through a web browser - using the standard querying interfaces provided by CBR-Works. Do not forget to save and deploy your new application! Select *Deploy/Application* from the *File* menu to do so.

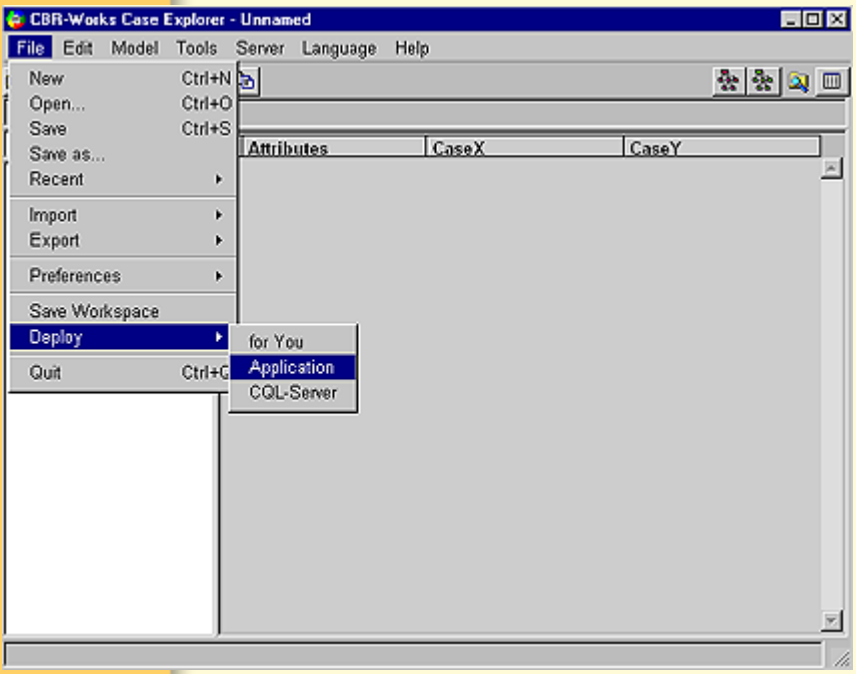


Figure 7-22: Deploying the Application

You will be asked to name the application (figure 7-23). After deploying, the application can be started by double clicking on the file stored.The running application created in this process may be accessed through the standard Web interfaces provided by CBR-Works. We have discussed this features in chapter 4.

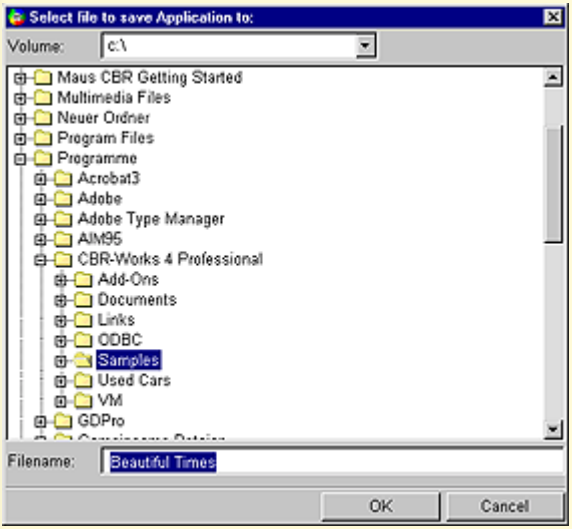


Figure 7-23: Saving the Application

REMINDER: Locally, you may access the Online Wizard through the URL **http://localhost:8051/launch/OnlineWizard**. Hostname and port may differ, according to the parameters given when creating the Web server.

7.5 Conclusion

You have just learned the basics on how to upgrade an existing thumb database to an intelligent CBR application accessible online without loosing your data.

8 Concluding Remarks

This "Getting Started" tutorial is intended to provide you with a general overview about the functionality of CBR-Works. It is not a programming manual nor a comprehensive guide on how to develop a concise domain model for a given task.

In conjunction with the reference manual, you may now adapt the demo applications according to your needs. Perhaps you would like to extend a domain model, add more cases, change similarities and rules. You may as well modify the standard online interface according to the style and corporate design of your company.

If you will need further assistance, please feel free to contact us. The tec:inno GmbH can help you in application development and offers a complete range of seminars and training for novice and experienced application developers.