A Tutorial Guide to DB-Discover, Version 3.4

Colin L. Carter, Howard J. Hamilton,
William B. Hase, and Carlos B. Rivera

Technical Report CS-99-02
April, 1999

ISSN 0828-3494
ISBN 0-7731-0389-9

1. Overview

1.1 Introduction

DB-Discover is a research software package being developed at the University of Regina which permits the discovery of useful information from large amounts of database data. DB-Discover is based on data-mining techniques recently developed in Knowledge Discovery from Databases research [CCH91, CH95a, CH95b, CH95c, Han94, HCC92].

The DB-Discover software package is useful for data access and summarization. As a data access tool, DB-Discover allows a data analyst to dynamically organize his or her data according to many different high level organizations without modifying the data itself. The analyst can then query the data according to high level concepts rather than according to specific data codes, even though these concepts are not present in the database. As a summarization tool, DB-Discover can generalize and summarize the data in large databases to many different levels, so that useful patterns in the data become apparent. Since the database itself is not used for these operations, they can be done quickly.

DB-Discover runs on a PC under Windows 95, Windows NT, OS/2 and on a UNIX-based machine (IRIX and SunOS) with a graphical, X-windows interface, or with a text-based, command line interface. DB-Discover's client-server architecture allows connection to databases running on other platforms.

1.2 Types of Tasks

Three types of tasks are implemented in DB-Discover, version 2.0: *Data Retrieval* tasks, *Derive Hierarchy* tasks, and *Characteristic Discovery* tasks.

A *Data Retrieval* task simply retrieves any mix of ungeneralized data and data converted to concepts so that the results can be viewed by the user. A *Derive Hierarchy* task relates the values of one column to the leaf nodes of the concept hierarchy of a second column. This classifies the first column according to the context of the second and provides a means to transport domain knowledge between databases in the form of derived hierarchies. A *Characteristic Discovery* task attempts to find some interesting relationship between various columns of one or more relations in the database. Performing *Characteristic Discovery* tasks is the primary use of DB-Discover. The task descriptions will cover the general operations to be carried out and refer to Section 3 for the individual systems.

1.3 Method used in Characteristic Discovery Tasks

DB-Discover is based on an column-oriented generalization algorithm which takes as input a relation retrieved from a database and generalizes the data guided by a set of concept hierarchies. A *concept hierarchy* is a tree of concepts arranged hierarchically according to generality. For discrete valued columns, leaf nodes correspond to actual data values which may be found in the database. For continuous valued columns, leaf nodes represent ranges of values. Higher level nodes represent more general concepts created by combining groups of lower level concepts under unique names. Two examples of concept hierarchies are given in Figure 1. The leaf nodes in the hierarchy for the AMOUNT hierarchy are discrete values, such as “Alberta” or “0-10K”, which may correspond to continuous ranges.

Using DB-Discover to summarize data is called a discovery task. First, relevant data if retrieved from a database. Generalization begins by converting the data values to matching leaf concepts from the appropriate concept hierarchy. The generalization process is limited by a set of user defined *column thresholds*, which specify the maximum number of distinct values that may exist for each column of the generalized relation. When each column of the input relation has been generalized to within the bounds of its threshold, many tuples of the relation are identical to other tuples. A count of each set of identical tuples is then stored in one tuple and the other tuples are eliminated. The result is called the *prime relation*.

A second limiting threshold is the *table threshold* which specifies the maximum number of tuples that may exist in the final generalized relation. After generalization to the column thresholds, there may yet be more tuples in the prime relation than the table threshold allows. An column is then chosen by some method and generalized one level. Duplicates are eliminated and the number of tuples remaining compared to the table threshold. This further reduction is iterated until the total number of tuples is less than or equal to the table threshold. The result is the *final generalized relation.*

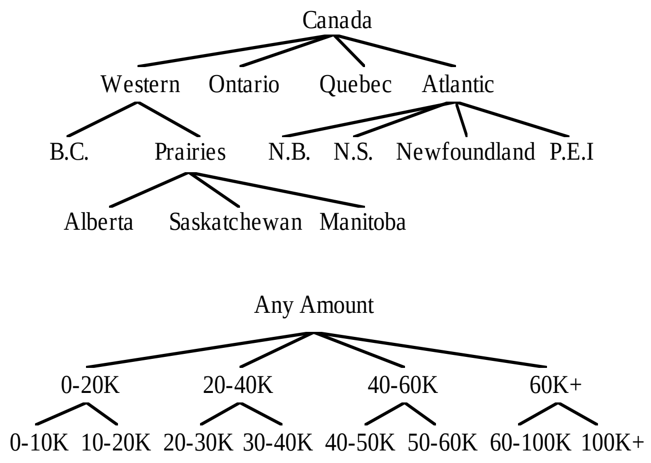


Figure 1. Concept Hierarchies for the PROVINCE and AMOUNT Columns

The final generalized relation is then presented the user who may wish to adjust generalization levels, sort the relation according to various sort orders, display numerical summary information, etc. The system provides functionality for all of these options.

This document will now describe in more detail the steps in defining a DB-Discover knowledge discovery task, covering most of the currently implemented options.

2. Performing Characteristic Knowledge Discovery Tasks

Defining a characteristic knowledge discovery task requires several steps. After the DB-Discover system has been started, a database connection must be established. Then concept hierarchy files must be loaded, after which a discovery task may be defined. The discovery task definition includes specifying the target objects and predicates, defining the parameters to limit generalization, starting the discovery task, viewing the results and adjusting the generalization to suit the needs of the discovery task.

2.1 Establishing a Database Connection

When the DB-Discover system is started, the first task is to establish a database connection. This involves choosing a machine that hosts a database management system (DBMS) and logging on.

2.1.1 Logging on

Each user must log on to the database. This requires a logon name and may require a password, depending on the database account. The DB-Discover research group has a general logon name and password for all members under which all research databases are available.

2.2 Loading Hierarchy Files

After connecting to a database, the user must load one or more concept hierarchy files. *Concept hierarchy files* contain the definition of one or more concept hierarchies. In characteristic knowledge discovery tasks, only those columns for which a concept hierarchy exists can be generalized. The concept hierarchies become generally available after being loaded from a concept hierarchy definition file. Only those definitions which have a matching column in the current database will be loaded. If some definition does not match a valid column, a warning will be issued and the hierarchy skipped.

Concept hierarchies are defined using tabbed ASCII files. Lines which have no indentation indicate the beginning of a new definition. The hierarchical structure of the file is then defined with each tab character being equivalent to a level in the concept hierarchy tree. An example of a tabbed hierarchy file for the PROVINCE column is shown in Figure 2.

2.2.1 Multiple Hierarchies

A hierarchy is given a name in the form <column_name>[:<concept_name>]. Each hierarchy is given the name of the column with which it is associated. In DB-Discover, version 3.4, more than one concept hierarchy can be defined per column. Defining multiple hierarchies per column allows more possibilities in refining and focusing your knowledge discovery. For example, the province hierarchy in Figure 2 might be called

- province
  - Canada
    - Ontario
    - Quebec
    - Western
      - British Columbia
      - Prairies
        - Alberta
        - Saskatchewan
        - Manitoba
    - Atlantic
      - New Brunswick
      - Nova Scotia
      - Newfoundland
      - PEI
    - Other in Canada
  - Outside Canada

Figure 2. A concept hierarchy for the column PROVINCE

PROVINCE:REGION. This hierarchy could be useful for studying data in the context of various regions. Another hierarchy for province based on average income could be defined as PROVINCE:AVERAGE_INCOME. This hierarchy might help you decide in what province to market a product first. And finally a hierarchy based on population in a province could be PROVINCE:POPULATION.

Multiple hierarchies becomes of interest when you select your target columns and you target predicate for your knowledge discovery task. You can see in the Selecting Columns dialog (Figure 12) that there are three different hierarchy concepts based on the column PROVINCE. One or more of these column:concept items may be selected for the generalized output columns. These column:concept items are also available for comparison when you are setting column qualifications in the Column Qualification dialog.(Figure 13)

2.3 Specifying Target Objects

When the hierarchies have been loaded, you may then begin defining a characteristic task. The first step is to define the database objects which specify the data to be extracted. These include the target tables and columns.

2.3.1 Target tables

The target tables are the database relations from which data will be extracted.

2.3.2 Target columns (or concepts)

The target columns are the columns from the target tables that will be retrieved from the database. With multiple hierarchies a target column is specified with the name of the target concept, giving the name of the concept which is associated with a column instead of the column name itself. An example of this is **Province:Region** where Region is the concept that will be used to generalize the column Province.

2.4 Specifying Predicates

When the target columns and tables have been specified, the data may need to be limited by some predicate ("where" clause). Two types of predicates may be specified, join predicates and regular predicates.

2.4.1 Join predicates

When more than one table is specified as the target table, a join predicate must be specified. The join predicate tells which columns are common between tables so that invalid data is not returned by a discovery task. An example of a join of tow tables is:

```
select <some columns> from award a, province b where a.org_code = b.org_code
```

Everything following the "where" above is the join predicate. However, the join predicate will still need to be set as part of the regular predicate.

2.4.2 Regular predicates

The regular predicate specifies the rest of the conditions that limit the data extraction and refer to any columns that are available in the current join of target tables. You are not limited to conditions on the target columns. An example of a database query with a non-join predicate is:

```
select amount, province from <some tables> where amount >= 22000 and disc_code = "Computer" and
province "Western"
```

The non-join predicate may specify data according to any high level concept in the target column's hierarchy. The database query above extracts data where the *province = "Western"*, but *Western* is not an actual data value in the province column of the data. *Western* is a concept defined in the province hierarchy (see Figure 2). The system will translate the predicate *province = "Western"* into *province in (B.C., Alberta, Saskatchewan, Manitoba).*

2.5 Other task parameters

The remaining task parameters affect other aspects of the discovery task. These include values related to data retrieval, values to limit the amount of generalization and parameters controlling the way information is displayed. A number of thresholds constrain the data retrieval and generalization process. The generalized relations may then be sorted according to different sort strategies. The amount of information shown in the generalized relation can also be adjusted, showing absolute and percentage sums of numerical columns and hiding or showing columns that have been maximally generalized. These functions will be covered in this section.

2.5.1 Data retrieval parameters

2.5.1.1 Retrieval threshold

The *retrieval threshold* specifies how many tuples will be retrieved from the database. This is not to be confused with the table threshold, which specifies the maximum number of tuples that may exist in the final generalized relation. When dealing with large relations, the user may want to try a discovery task on the first 1,000 tuples or so. Setting the retrieval threshold to 1,000 will accomplish this. If there are fewer than 1,000 records matching the discovery task specification, only those that are found will be returned. Setting the retrieval threshold to 0 tells DB-Discover to retrieve all data.

2.5.1.2 Retrieval block size

When data is retrieved from the database, it may take some time, especially if many tuples are being retrieved. Some feedback is helpful to let the user know how things are going. This feedback is provided by retrieving the data in blocks and reporting back to the user as each block is completed. The size of the block may be adjusted according to the size of the data relation being retrieved or the user's preference.

2.5.2 Thresholds

There are several thresholds for tuning the results of a discovery task. These include the column thresholds, the table threshold, the noise threshold.

2.5.2.1 Column threshold

The *column threshold* specifies the maximum number of distinct values that an column may have in the prime relation. An column may have an extremely large number of possible values in the database if it is a numerical value, and it may have a great number of values if it is a discrete (nominal) valued column. The DB-Discover system reduces the number of distinct values for numerical columns by mapping the values into a finite number of discrete ranges. These ranges form the leaf nodes of the hierarchies for those columns. For discrete valued columns, the leaves are the actual values that may be encountered in the database. The concept hierarchies then combine related lower level concepts into higher level concepts, reducing the number of nodes at each higher level. Since the concept hierarchies guide the generalization process, more general relations have fewer distinct values for each column than less-general ones. The column threshold specifies the maximum number of distinct values that may exist for each column and may be arbitrarily set by the user. For example, setting the column threshold for PROVINCE to 5 for the concept hierarchy shown in Figure 2 means that at most 5 of the values shown in Figure 2 will appear for province in the prime relation

A default column threshold is defined for any column that has not specifically set. Changes to the default column threshold are mirrored in all such columns. Each column threshold may be individually set. After being set, the column's threshold will not vary when the default threshold is changed.

2.5.2.2 Table threshold

The table threshold specifies the maximum number of tuples that may exist in the final generalized relation. After the relation has been reduced to the column thresholds in the first round of generalization, the number of tuples in the prime relation is compared to the table threshold. If there are more tuples than the table threshold allows, the relation is repeatedly generalized until the number fall within the bound specified.

2.5.2.3 Noise threshold

The noise threshold limits the amount of data displayed to the user based on the percentage of original ungeneralized tuples represented by a generalized tuple in the prime or final generalized relation. For example, the user may only want to see tuples that represent more than 5% of the tuples in the original ungeneralized relation. Setting the noise threshold to 5 will therefore cause tuples that represent less than 5% of original tuples not to be displayed in the output. This does not destroy or delete that information from the generalized relation, it simply suppresses the printing of all tuples below that threshold. Setting the noise threshold to 0 causes all tuples to be printed.

2.5.3 Sorting

In addition to the various thresholds that control the amount of output, the generalized relations may also be sorted. The sort order may be specified before or after the data is retrieved from the database. In characteristic knowledge discovery tasks, the relation is not actually sorted until right before display since it is much less expensive to sort a small generalized relation than a large ungeneralized one. It is also cheaper to sort it in memory than to ask the database to sort it for us. For data retrieval tasks, however, the sort must be defined before the data is retrieved, therefore the information is passed on to the database and sorted there.

The sort is defined by two parameters, the order of the sort columns and the direction of the sort. The relation may be sorted according to one or more columns. The first to be defined is the primary sort column and all others are secondary sort columns. The sort columns will have precedence in the order they are set. Those specified first will take precedence over those defined later. The sort direction applies to each individual column and may be either ascending (A-Z, 0-9) or descending (Z-A, 9-0). The default direction if it is not specified is ascending.

2.5.4 Numerical summaries

When information is being retrieved, the user can ask for certain numerical summary information to be displayed. This includes information about the number of original ungeneralized tuples contributing to each generalized tuple and the sums of numerical columns.

2.5.4.1 Counts

The number of original ungeneralized tuples contributing to a generalized tuple is termed the tuple’s *count*. An ungeneralized tuple’s count is one. When it is generalized and inserted into a generalized relation, the relation is searched for a match. If none is found, the tuple is inserted as a new tuple with a count of one. If a match is found, the matched tuple’s count is incremented and the tuple to be inserted is discarded. If a generalized tuple is further generalized, the same process is followed, except that the total of the inserted tuple’s counts is added to the matched tuple.

2.5.4.1.1 Display of Counts

Setting *counts* on will cause the count of the generalized tuple is to be displayed when printing the generalized relation. Turning *counts* off will cause the counts not be printed.

2.5.4.1.2 Display of Percent Counts

Setting *percent counts* on causes the proportion of the current tuple's counts in relation to the sum of all tuples' counts to be displayed when the relation is printed. For example, if 500 tuples were retrieved from the database and one generalized tuple represents 200 input tuples, the percent count of that tuple would be 40% since it represents 40% of the total tuples read. Turning *percent counts* off causes this information not to be printed.

2.5.4.2 Sums

A relation's columns may be a mixture of numeric and non-numeric values. For numeric values, some may represent things like product id's while some may record arbitrary values like cost or number of items. While the user would not want the sum of all product id's, he or she might want the total cost or total number of items. The *sum* of an column will be the total number of items for non-numeric data and the total of the values for numeric data. Sums can be automatically calculated by the DB-Discover system.

2.5.4.2.1 Display of Sums

Specifying an column as a sum column will cause the values of that column to be recorded as they are retrieved from the database and summed as they are generalized and combined with other tuples. When the data is printed for the user, the sum of all values will also be printed with the heading #<column_name> (where the # sign specifies the sum). If the user wants any sum information to be displayed, then the column should be specified as either a sum or percent sum column before the data is retrieved. If neither "sum" nor "percent sum" are specified for an column when the data is retrieved, then the actual numeric values are discarded. Setting an column to a percent sum after the data has been retrieved requires that the data be reread from the database. If the retrieved relation is large, this could take substantial extra time.

2.5.4.2.2 Display of Percent sums

Specifying an column as a percent sum column causes the percentage of an column's sum in proportion to the sum of all sums to be printed with the output relation. For example, if the total number of items represented by a discovery task is 10,000 and one generalized tuple represents 1,000 of these, its percent sum value would be printed as 10% under the heading %<column_name> (where the % specifies a percent sum).

2.6 Initiating the Learning Task

When all relevant task parameters have been set, data retrieval and generalization may be started. At the stage, the target object information and the predicates are used to build an SQL query, the database is initialized with this discovery task, and the data are retrieved and stored in memory. After the data have been retrieved, they are immediately generalized to the column threshold and then to the table threshold if necessary. If an error occurs in any of these steps, a message will be printed and appropriate action may be taken. Otherwise, the results will be displayed on the screen and the user may then adjust various parameters to tune the results to his or her liking.

2.7 Refining the Discovery Results

The user may or may not like the results of the discovery task as he or she first formulated it. However, many of the parameters can be adjusted and the results redisplayed. The column, table and noise thresholds may all be adjusted and the results will automatically be adjusted on the screen. The user may also set or clear the counts and percent counts. If an column was specified as a sum or percent sum before the data retrieval, then sums or percent sums for each appropriate column may be set or cleared without causing the data to be reread. As mentioned above, however, if the column was neither a sum or percent sum, then setting it as either will cause the data to be reread.

In addition to the above refinements, the user may choose to further generalize the results in a step by step manner. This may be accomplished by either specifying a particular column to be generalized further or by specifying a generalization strategy which will choose an column to generalize. If the user specifies a particular column, then that column will simply be generalized one more level and the results redisplayed. Once an column has reached the most general value of "ANY", this option will not longer have any result.

DB-Discover, version 3.4 provides two generalization strategies. The first is to reduce according to the column with the most distinct values. This approach chooses the column with the most distinct values at the current level of generalization and generalizes it one level. The other strategy generalizes the user selected column by one level.

2.8 Saving Discovery results

The Windows 95/NT, OS/2 and the command line version have the ability to save the results of the current session as text and the ability to save the state of the DB-Discover engine so that a discovery task can be returned to at a later time. DB-Discover's saved files have the extension *.DBD and can be loaded by any of the supported platforms running version 3.4 or higher.

3. Tutorial Guide

3.1 Introduction

The appearance and functionality of the Windows 95/NT, OS/2 and X-windows versions of DB-discover are very similar. Therefore, the following guide will give an overview of executing a discovery task with the Windows 95 interface.

3.2 Tutorial Conventions

• Words that are in all capitals and italics are columns or settings for the DB-Discover software package. For example: an column is *AMOUNT*, and a setting is *OFF*.
• Words that are in bold with the first letters in Capitals are buttons in Windows 95 version of DB-Discover. For example: **Ok** and **Set Query**.
• Normal text starting with capitals letters are titles of dialog boxes. For example: Task Definition and Column Qualification.

• The paragraphs surrounded by single line boxes contain the exact steps necessary to execute the tutorial discovery task.
• All screen shots are from the Windows 95 version of DB-Discover

3.3 Task Definition for Tutorial

For the GUI's and command line interface of DB-Discover the following discovery task will be performed (discovery task settings that are in Bold are optional settings):

Discovery Task: To examine whether there is a relationship between AMOUNT and PROVINCE for DISC_CODE = "HARDWARE" or DISC_CODE = "SOFTWARE within the NSERC database.

Task type : *Characteristic*
Hierarchy file: *nserc.chf*
Selected relations: *AWARD, ORGANIZATION*
Join tables: *AWARD A, ORGANIZATION B*
Join columns: Relation 1: *ORG_CODE*
Relation 2: *ORG_CODE*
Columns selected: *AMOUNT, AREA_CODE, DISC_CODE, PROVINCE*
Column qualification: *DISC_CODE = "HARDWARE"*
Additional qualification: *OR DISC_CODE = "SOFTWARE"*
Sort Order of columns: *AMOUNT = descending, AREA_CODE = ascending, DISC_CODE = descending, PROVINCE = ascending*
Column thresholds: *DEFAULT = 10, AMOUNT = 25, AREA_CODE = 10, DISC_CODE = 10, PROVINCE = 12*
Set Count: *ON*
Set Percent Count: *ON*
Sums: *AMOUNT*
Percent sums: *AMOUNT*
Retrieval threshold: *0*
Retrieval block size: (Not implemented)
Table threshold: (Not implemented)
Noise threshold: *0*
"ANY" columns: *OFF*

3.4 Windows 95/NT Tutorial

3.4.1 Introduction

The Windows 95/NT version of DB-Discover was written using Microsoft Visual C++ running on Windows 95 and Windows NT. The name of the executable is DB_Discover.exe. You can install this using the Install Wizard by running the setup.exe on the installation disk. You will be prompted for which version you wish to install, the ODBC version of the Oracle version. The Install Wizard will than install DB-Discover into the folder **C:\Program Files\UofR\DB-Discover\** by default. A link to DB-Discover will be installed into the 'Start Menu' of Windows 95. If you wish to uninstall DB-

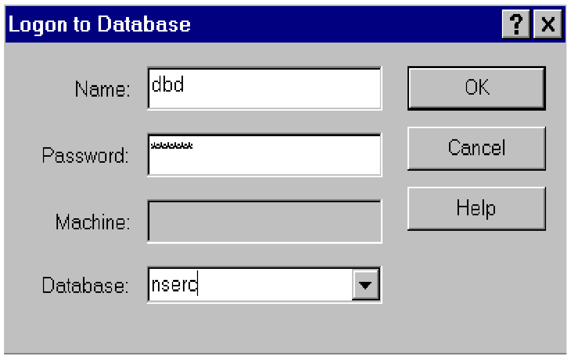
Discover use the control panel 'Add/Remove Programs' in Windows 95 or use the Uninstall.exe in Windows NT.

The ODBC version of DB-Discover requires the 32bit version of ODBC. A version of this is included in the DB-Discover folder. To install it just double click on the file ODBC32.exe to expand the archive and then run the setup.exe.

The Oracle version of DB-Discover requires SQL*Net and must be obtained through Oracle.

The tutorial user should:
• install DB-Discover using the Install Wizard (setup.exe)
• insure that the 32bit version of ODBC or SQL*Net is installed

3.4.2 Logging on



Name: dbd
Password: ******
Machine:
Database: nserc

Figure 3. Logon dialog box

After DB-Discover is started you can open the logon dialog box (Figure 3) by selecting **Logon** under the **Database** menu. The dialog box appears with text boxes for entering a logon name, password, database server and database. The **Database** text box will show the available servers and databases in the ODBC version. Click the pointer in the top boxes and enter the logon name, then repeat for the password in the next box. The pointer must remain in the box that the user is entering text in. The other two text boxes show the server being connected to and the database being used. To fill in these boxes, the database and database server name can be typed in or the user can select a server and database from the list at the bottom by clicking on it. The database will then be automatically filled in. The user should then click the **Ok** button to continue. After a short wait the logon dialog will disappear. If there is no error message you are logged on.

The tutorial user should:
• enter the logon name and password
• select the *nserc* database
• and click **Ok** to logon

3.4.3 Starting a discovery task

The main window where the final relation will be displayed and refined will now be active (Figure 4). To start a discovery task, select **Define** from the **Task** menu. The large grid underneath the tool bar will display the results from the discovery task after all applicable information is entered. The buttons at the top of the screen won't become active until a relation is produced.

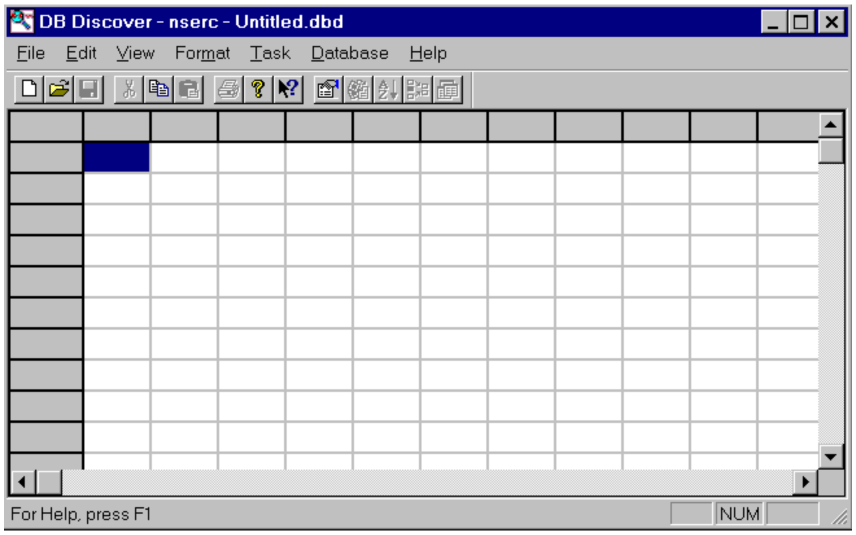
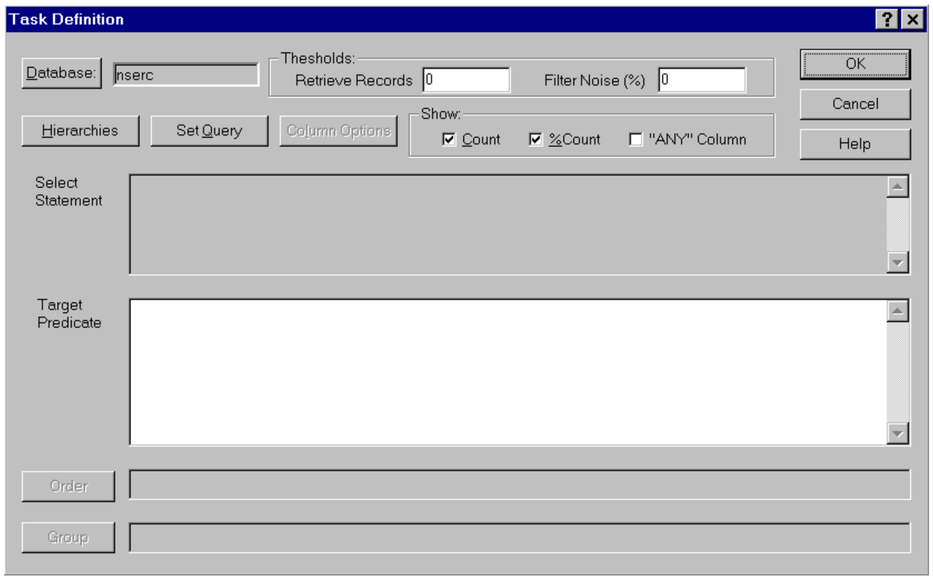

DB Discover - nserc - Untitled.dbd
File Edit View Format Task Database Help
For Help, press F1
NUM

Figure 4. Main Window

After selecting Define, a large dialog box, the Task Definition dialog, will appear (Figure 5).



Database: nserc

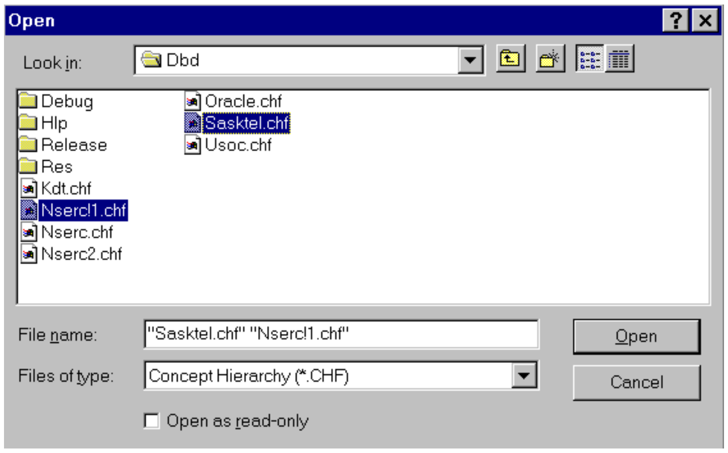
Thesholds:
Retrieve Records 0
Filter Noise (%) 0

Show:
☑ Count ☑ %Count ☐ "ANY" Column

Figure 5. Task Definition dialog

3.4.4 Load hierarchy file

The user must begin a session by loading a hierarchy file. If a no hierarchy file is loaded clicking the **Hierarchies** button will immediately open a file open dialog box (Figure 6). Once a hierarchy has been loaded the Hierarchy dialog box will open when the **Hierarchies** button is clicked. Click the **Hierarchies** button and when a dialog appears select the *.chf file from the list by double clicking it or selecting it and clicking the **Open** button. Multiple hierarchy files can be selected and opened in the file open dialog box by holding down the control key and clicking on the desired files.

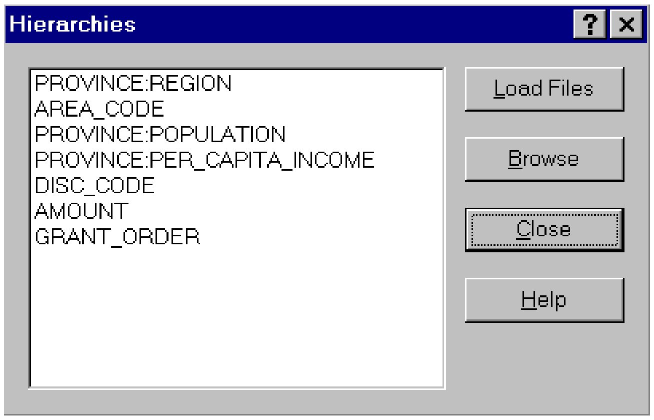


File name: "Sasktel.chf" "Nserc!1.chf"

Files of type: Concept Hierarchy (*.CHF)

Figure 6. Loading Hierarchy files

The top level of the loaded hierarchies is displayed in the Hierarchy dialog box (Figure 7). To see the whole hierarchy tree the user must click on the **Browse** button. This opens the Hierarchy Browse dialog box. When the desired hierarchy file is selected click **Close**.



PROVINCE:REGION
AREA_CODE
PROVINCE:POPULATION
PROVINCE:PER_CAPITA_INCOME
DISC_CODE
AMOUNT
GRANT_ORDER

Figure 7. Hierarchy dialog

The tutorial user should:
- click the **Hierarchies** button
- load the *nserc.chf* file
- click the **Hierarchies** button again to view the top level of the hiearchies

3.4.5 Setting the discovery task

The user should next click on the **Set Query** button in the Task Definition dialog (Figure 5). This will open the Task Query dialog box (Figure 8).

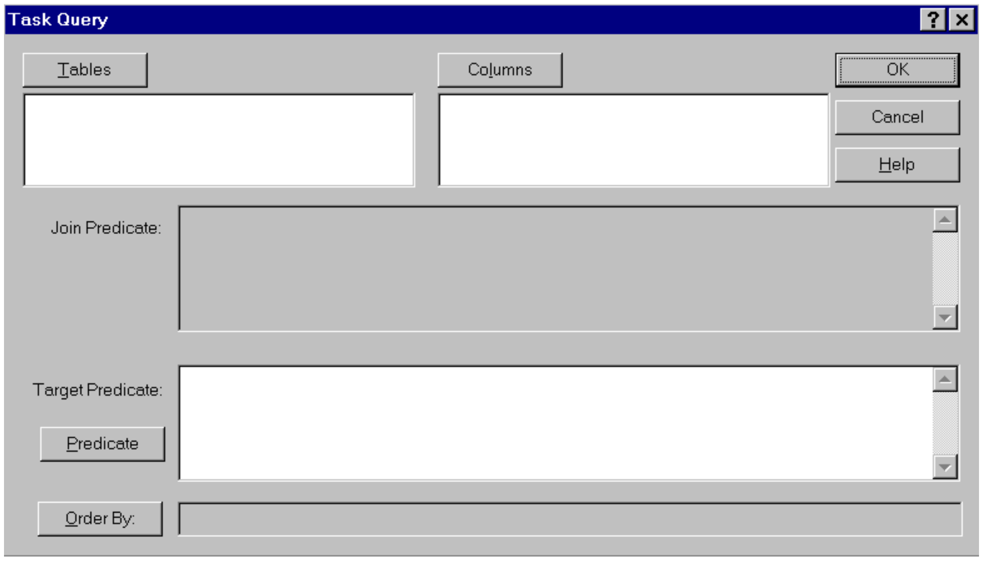


Figure 8. Task Query dialog box

To set the required tables for the query click on the **Tables** button. The Select Tables dialog (Figure 9) will appear showing the relations that are available from the database. The user selects one or more of them by double clicking on their names, then clicking **Ok**.

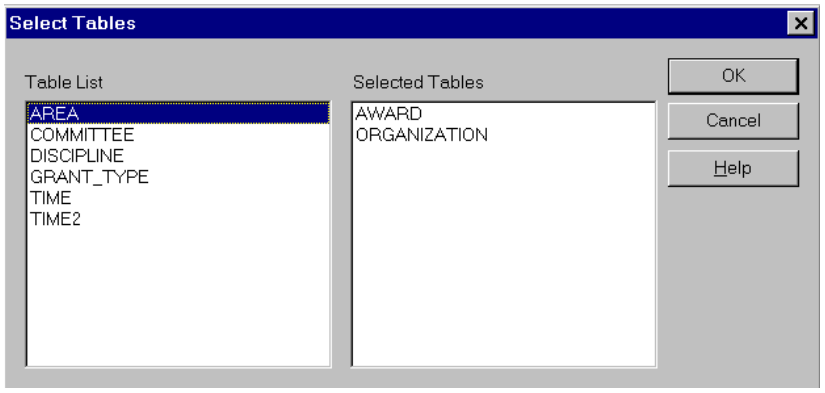
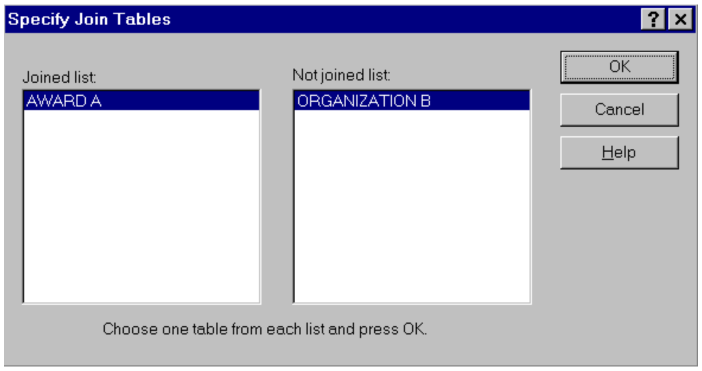


Figure 9. Select Tables dialog box

<u>The tutorial user should:</u>
• click the **Set Query** button
• click on **Tables** button
• double click on *AWARD* and *ORGANIZATION* from the list

• and click **Ok**.

When the Specify Join Tables dialog appears (Figure 10), the user should click one table from each window listing the join tables and then click **Ok**. This may continue for several dialog boxes if more then one join predicate can be set.



Joined list:
AWARD A

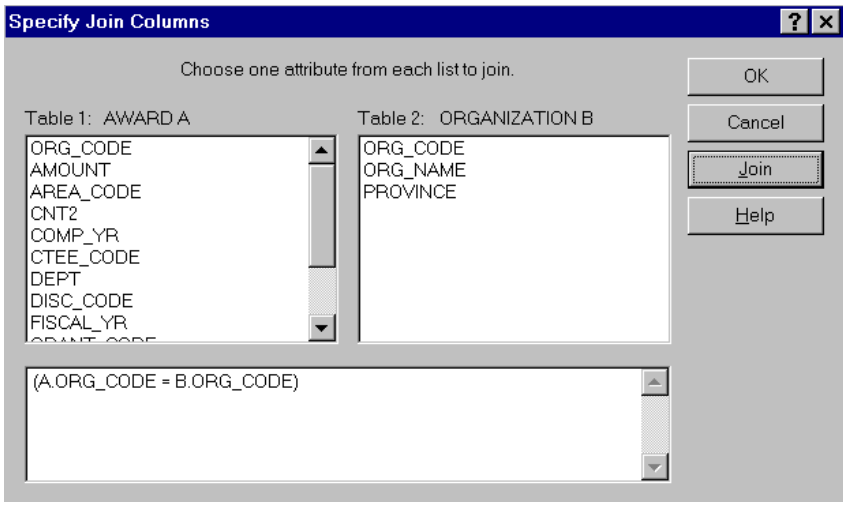
Not joined list:
ORGANIZATION B

Choose one table from each list and press OK.

Figure 10. Join Tables dialog box

<u>The tutorial user should:</u>
• click on *AWARD A* from the left window and *ORGANIZATION B* from the right window.
• click the **Ok** button.

When all the join tables have been set, the Specify Join Columns dialog (Figure 11), with two list windows, will appear. The user should again click on one column from each list window. The two columns selected should be in the same domain since these are the two columns that will be joined. Click the **Join** button to make the join predicate for the two selected columns. When all the column joins are specified click on **Ok**.



Choose one attribute from each list to join.

Table 1: AWARD A
ORG_CODE
AMOUNT
AREA_CODE
CNT2
COMP_YR
CTEE_CODE
DEPT
DISC_CODE
FISCAL_YR

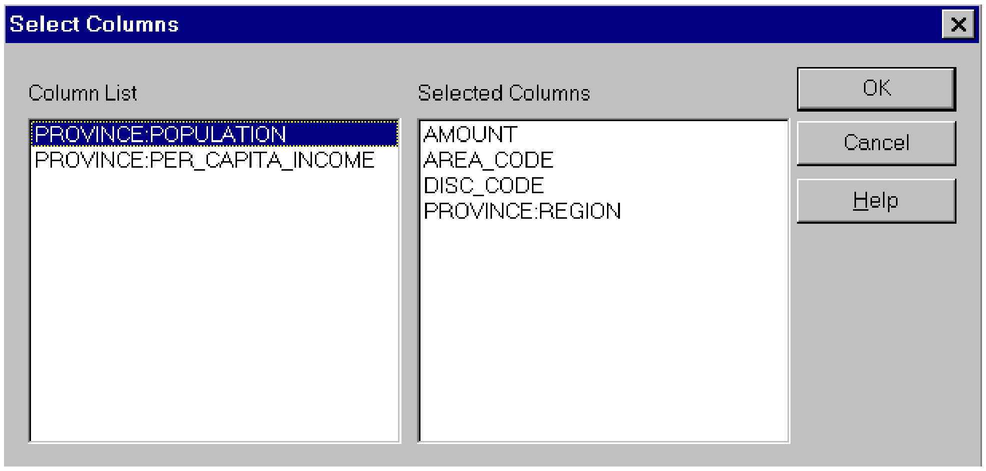
Table 2: ORGANIZATION B
ORG_CODE
ORG_NAME
PROVINCE

(A.ORG_CODE = B.ORG_CODE)

Figure 11. Join Columns dialog box

<u>The tutorial user should:</u>
• click on *ORG_CODE* from each of the lists
• click on the **Join** button
• click the **Ok** button.

Next click on the **Columns** button in the Task Query dialog box (Figure 8). This will open the Select Columns dialog box (Figure 12) with a listing of all the columns. The user should select one or more of the columns by double clicking on the column name. . The user should click the **Ok** button when finished. The selected columns will appear as the columns of the final output. This is not true if the column goes to 'ANY'.



Column List:
PROVINCE:POPULATION
PROVINCE:PER_CAPITA_INCOME

Selected Columns:
AMOUNT
AREA_CODE
DISC_CODE
PROVINCE:REGION

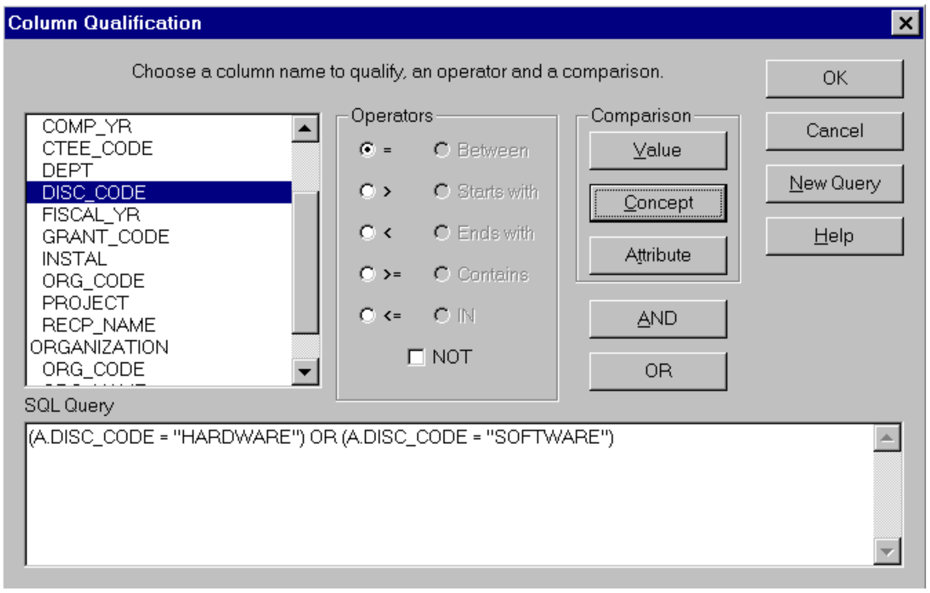
Figure 12. Select Columns dialog box

<u>The tutorial user should:</u>
• click the **Columns** button in the Task Query dialog box.
• click on the following items in the list: *AMOUNT*, *AREA_CODE*, *DISC_CODE*, and *PROVINCE:REGION*
• click the **Ok** button.

The two top list boxes in the Task Query dialog (Figure 8) and the top large edit box underneath them should all contain text now. If the user doesn't want to set the target predicate or order of columns then he or she can click **Ok** from the Task Query dialog and move to Section 3.4.8, otherwise he or she should follow the procedure in 3.4.6 and 3.4.7.

3.4.6 Setting target predicates (Optional)

Setting the target predicate is not required for the discovery task but it is strongly recommended to limit the search scope. Click the Target **Predicate** button from the Task Query dialog (Figure 8) to bring up the Column Qualification dialog (Figure 13).



(A.DISC_CODE = "HARDWARE") OR (A.DISC_CODE = "SOFTWARE")

Figure 13. Column Qualification dialog

Qualification is accomplished by selecting a column from the list on the left side, then selecting an operator from the list given in the center and finally selecting one of the three comparison buttons on the right side. The large window across the bottom of the dialog will show the qualification as it is entered. The operators shown in the middle are self-explanatory. If an column that is selected has a hierarchy attached then the middle button on the right, **Concept,** will not be shaded out. If **Concept** is selected from the buttons on the right side, the hierarchy dialog box (Figure 14) will appear showing the hierarchies attached to the column.

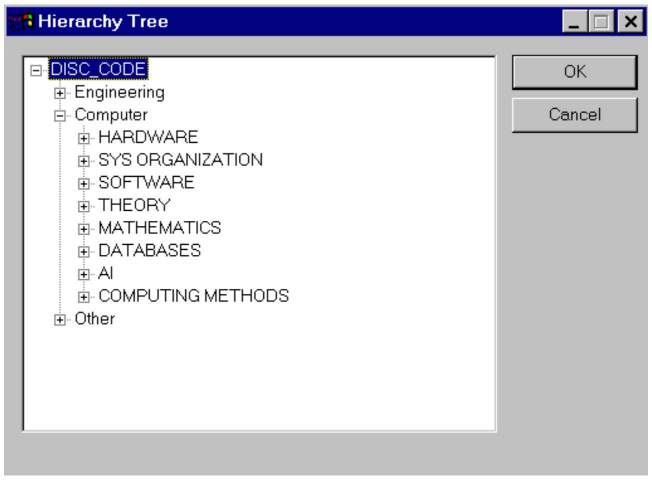
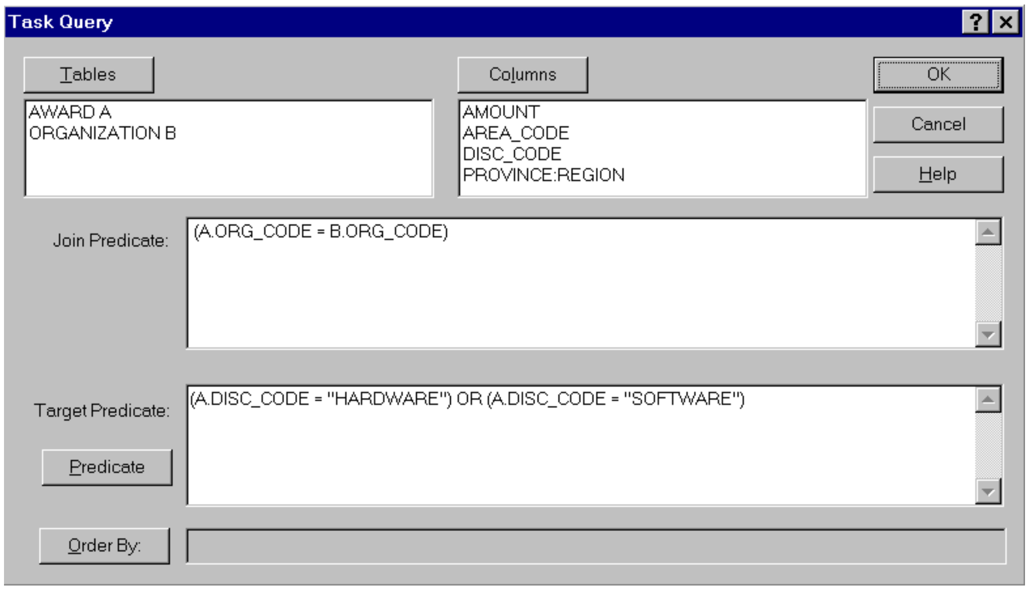


Figure 14. Column Qualification Hierarchy dialog

Clicking on the plus (+) signs causes additional levels of the hierarchy to be shown. When the desired level of qualification is found click in the box on the desired concept and then click the **Ok** button. Selecting a concept that is not at the bottom level of qualification will select all concepts underneath it. Between each column qualification the user must enter an **OR** or **AND** logical operator. This can be done with the buttons or it can be typed in manually. When the user is done qualifying columns the **Ok** button should be clicked. If the qualification generated is not what the user desired, the **New Query** button will clear this qualification. The user can also manually edit the qualification in the edit box at the bottom of the Column Qualification dialog. After clicking the **Ok** button in the Column Qualification dialog (Figure 13), the Target Predicate edit box in the Task Query dialog (Figure 15) should be filled in.



AWARD A
ORGANIZATION B

AMOUNT
AREA_CODE
DISC_CODE
PROVINCE:REGION

Join Predicate: (A.ORG_CODE = B.ORG_CODE)

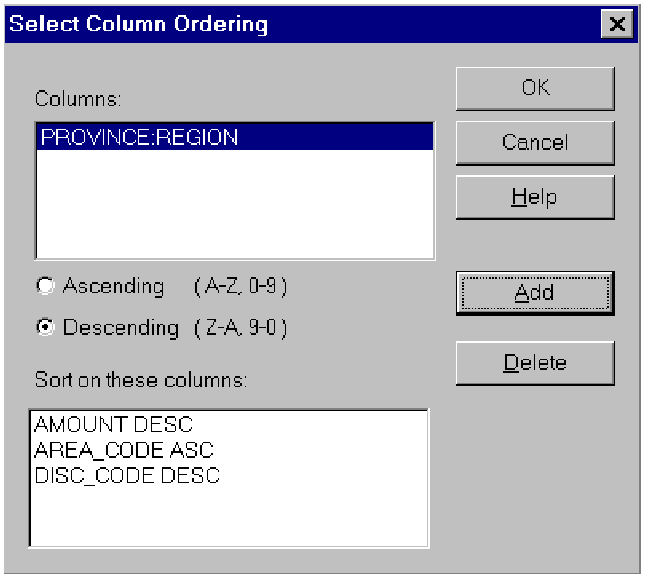
Target Predicate: (A.DISC_CODE = "HARDWARE") OR (A.DISC_CODE = "SOFTWARE")

Figure 15. Column Qualification dialog (filled in)

<u>The tutorial user should:</u>
• click on *DISC_CODE* from the *AWARD* table in the list at the left of the Column Qualification dialog
• click on the equals (=) button from the **Operators** in the middle
• click the **Concept** button
• expand the hierarchy out till *HARDWARE* is available
• select *HARDWARE* by clicking on the name
• click the **Ok** button in the Column Qualification Hierarchy dialog
• click on the **OR** button to put a logical operator between the two qualifications
• the first 6 steps should be done again but *SOFTWARE* should be selected instead of *HARDWARE*
• when the two predicates are finished being entered the user should click the Ok button at the bottom of the Column Qualification dialog. The SQL Query box should be identical to Figure 13

3.4.7 Ordering of columns (Optional)

The order of columns in the final output can be controlled by clicking on the **Order by** button in the Task Query dialog box (Figure 8). The Select Column Ordering dialog (Figure 16), will appear.



AMOUNT DESC
AREA_CODE ASC
DISC_CODE DESC

Figure 16. Select Column Order dialog

The list box at the top of the dialog will show the columns that do not yet have an order associated with them yet and the box at the bottom will show those columns and the order associated with them. The order is set for a column by clicking on an column at the top, choosing **Ascending** or **Descending** from the check boxes underneath and then clicking on the **Add** button. If you make a mistake, the **Delete** button will remove the selected column from the bottom window. You can also double click on the column names or drag and drop them to move them from one list to the other. The columns in the bottom list box can be reordered by dragging the column to the position you want it, and then dropping it. The order of the columns in the bottom list specifies the order the actual columns will be sorted in. When the desired columns have an order set, click the **Ok** button at the bottom.

One note of caution: If you set a sort on the columns before you retrieve data from the database the sort will be done on the ungeneralized data by the database itself. If you set the sort at the main window after the data has been retrieved only the generalized data will be sorted which can save processing time since there is less data to be sorted. For this example we are not concerned about processing time because the nserc database is small.

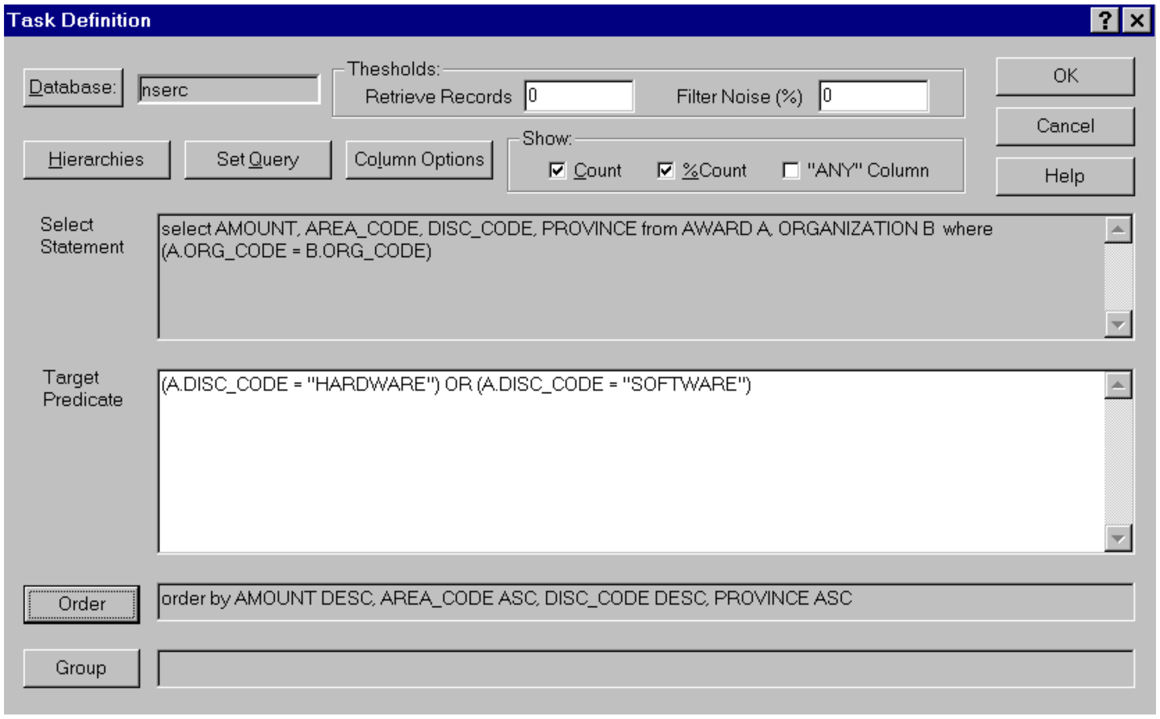
<u>The tutorial user should:</u>
• click *AMOUNT* from the list at the top, make sure the **descending** button is on and click the **Add** button
• click *AREA_CODE* from the list at the top, make sure the **ascending** button is on and click the **Add** button

• click *DISC_CODE* from the list at the top, make sure the **descending** button is on and click the **Add** button
• click *PROVINCE* from the list at the top, make sure the **ascending** button is on and click the **Add** button
• when all the columns have an ordering associated with them, click the **Ok** button.

3.4.8 Finishing setting a discovery task

Click **Ok** in the Task Query dialog to return to the Task Definition dialog. If the discovery task is set properly then the Select Statement and Target Predicate areas in the Task Definition dialog must have text in them. There may be text in the Order By area, if column orders were specified. If the desired areas are filled, click the **Ok** button to submit this discovery task. If there is an error in the discovery task, an error message will appear describing the problem. The user can, optionally, change the Thresholds, Column Options and Count by following the instructions in the next two sections.

If the tutorial user has set the discovery task correctly, the Task Definition dialog should appear as in Figure 17.



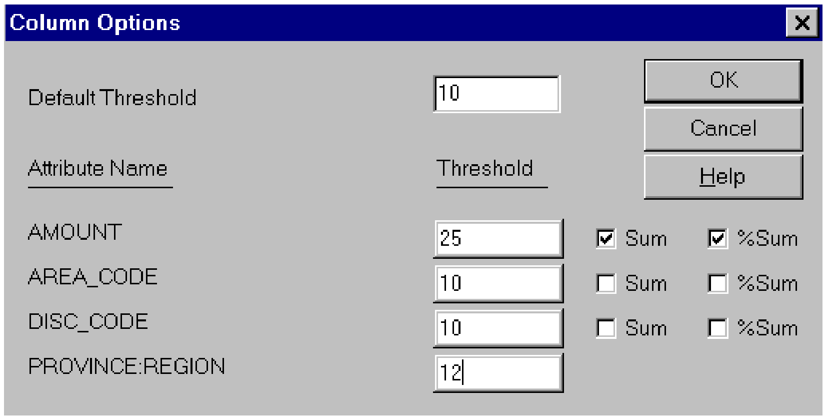
Database: nserc
Retrieve Records: 0
Filter Noise (%): 0
Show: ☑ Count ☑ %Count ☐ "ANY" Column
Select Statement: select AMOUNT, AREA_CODE, DISC_CODE, PROVINCE from AWARD A, ORGANIZATION B where (A.ORG_CODE = B.ORG_CODE)
Target Predicate: (A.DISC_CODE = "HARDWARE") OR (A.DISC_CODE = "SOFTWARE")
Order: order by AMOUNT DESC, AREA_CODE ASC, DISC_CODE DESC, PROVINCE ASC

Figure 17. Task Define dialog (Filled in)

3.4.9 Column options (Optional)

The column thresholds specifies the maximum number of distinct values that a column may have in the prime relation. Setting the thresholds is optional since they are already set to a default value. The **Column Options** button on the Task Definition dialog will open the Column Options dialog (Figure 17) which allows the user to set the default column threshold to a new value or to change the column thresholds individually. This dialog also allows the *SUMS* and *PERCENT SUMS* display fields to be activated for columns. *SUM* or *PERCENT SUM* can be set for a particular column by clicking the check boxes to the far right of the column name. Changing the thresholds is accomplished by clicking in the text box to the right of the desired column and typing in the new threshold. After all thresholds have been set to the desired levels, click the **Ok** button.

The tutorial user should:
• open the Column Options dialog by clicking on the corresponding button
• place the pointer in the text box to the right of Default Threshold and type 10
• place the pointer in the text box to the right of AMOUNT and type 25
• place the pointer in the text box to the right of PROVINCE and type 12
• since the other 2 columns are set to the original default value in the tutorial knowledge discovery task they will be updated to the new default value
• the Sum and % Sum boxes should be checked to the right of AMOUNT
• click the **Ok** button when finished.
After the tutorial user has set the thresholds, the Set Column Options dialog should appear as shown in Figure 18.


Column Options

Default Threshold: 10

Attribute Name	Threshold		
AMOUNT	25	✓ Sum	✓ %Sum
AREA_CODE	10	☐ Sum	☐ %Sum
DISC_CODE	10	☐ Sum	☐ %Sum
PROVINCE:REGION	12		

Figure 18. Set Column Options dialog (Filled in)

3.4.10 Setting other thresholds (Optional)

Setting the Retrieval threshold, Retrieval block size (not implemented in version 3.4), Table threshold (not implemented in version 3.4) and Noise threshold to new values is done by typing the new values in the text boxes in the Task Define dialog (Figure 17). Setting the *Show Count*, *Count %* and *"ANY" Columns* is done by clicking on the text.

The values given in Section 3.3 are the default values so no changes are needed.

3.4.11 Executing the discovery task

When the values in the Task Definition dialog (Figure 17) are all set to the desired values, click the **Ok** button in the top left corner to execute this discovery task. The dialog will disappear and the original window will be active again. The DB-Discover package will now start to retrieve values that match the specified discovery task. A small dialog will appear showing the number of tuples retrieved. If the user wishes to stop this process, he or she can hit the **Break** key on the keyboard. The final relation, consisting of the tuples matching the discovery task settings, will appear in the main grid area of the original window after retrieval.

The tutorial user can check whether the entered values match the ones in Section 3.3 by examining Figure 17 and Figure 18. These Figures shows a Task Definition dialog and Column Options dialog set to the values needed to execute the discovery task correctly. If the values match, the user should click the **Ok** button at the top of the Task Definition dialog.

3.4.12 Refining the discovery task

After the final relation is displayed the user can further refine the relation using the **Adjust Task** button at the top of the original window. This will open the Adjust Task dialog (Figure 19) where you can select a column and generalize or specialize the relation as described in Section 2.7 of this manual.

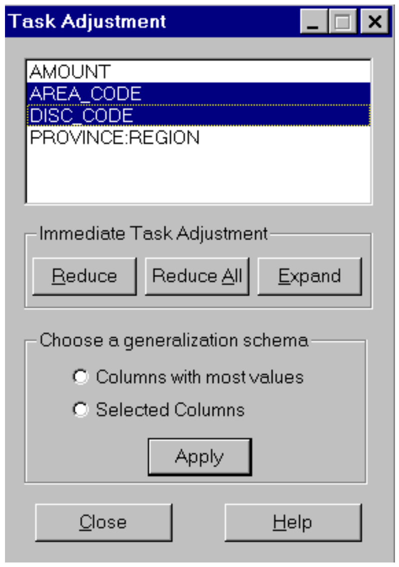


Figure 19. Task Adjustment dialog

You can also refine the task by going back into the Task Definition dialog where the settings and query can be modified. The discovery task can then be re-executed with the changed settings.

4. References

[CCH91] Y.D. Cai, N. Cercone, and J. Han, "Attribute-Oriented Induction in Relational Databases," in G. Piatetsky-Shapiro and W. J. Frawley, eds., *Knowledge Discovery in Databases,* AAAI/MIT Press, Cambridge, MA, 1991, pp. 213-228.

[CH95a] C. L. Carter and H. J. Hamilton, "A Fast, On-Line Generalization Algorithm for Knowledge Discovery," *Appl. Math. Lett.* **8** (2), 1995, pp. 5-11 .

[CH95b] C. L. Carter and H. J. Hamilton, "FIGR: A Fast, Incremental Generalization Algorithm for Knowledge Discovery," *Proceedings of the Eighth Florida Artificial Intelligence Research Symposium (FLAIRS-95),* Melbourne, FL, April, 1995, pp. 319-323.

[CH95c] C.L. Carter and H.J. Hamilton, "Performance Evaluation of Attribute-Oriented Algorithms for Knowledge Discovery from Databases," In *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence (ICTAI'95),* Washington, DC, November, 1995, pp. 486-489

[Han94] J. Han, "Towards Efficient Induction Mechanisms in Database Systems," *Theoretical Computing Science,* 133, 1994, pp. 361-385.

[HCC92] J. Han, Y. Cai, and N. Cercone, "Knowledge Discovery in Databases: An Attribute-Oriented Approach," *Proceedings of the 18th VLDB Conference,* Vancouver, British Columbia, 1992, pp. 547-559.