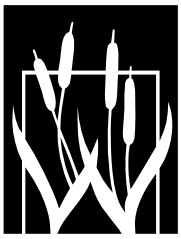

Whitemarsh
Information Systems Corporation

Whitemarsh Metabase
Data Modeler: Implemented Data Model
Users Guide

December 2007

Whitemarsh Information Systems Corporation
2008 Althea Lane
Bowie, Maryland 20716
Tele: 301-249-1142
Email: Whitemarsh@wiscorp.com
Web: www.wiscorp.com

Table of Contents

List of Figures

1 Introduction

The implemented data model (IDM) component of the data modeler module is designed to capture specifications of database data. That is, data models constrained within the boundaries of a particular database. Implemented data models are sometimes called logical data models. In contrast, specified data models are more conceptual. Specified data models are independent of database, DBMS, and any sort of physical constraints on a computing environment. The key characteristic of an implemented data model is that its collection of tables, columns, and relationship are all within a single schema.

Once the DBMS and operating system and computing environment is chosen, an implemented data model is often transformed to meet the required capacity and performance characteristics. This last type of data model is called the operational data model.

There is therefore a hierarchical relationship between the Specified, Implemented, and Operational data models.

Data elements and the semantics that assist in data element definition are defined within the data element module of the metabase. In addition to being a source for column semantics, a function within the implemented data model allows both the re-designation of the data element from which a column draws its semantics and also the ability to promote a column to either be an attribute or to be a data element.

Collectively the implemented data models and their associated semantics act as database templates for operational data models. If an organization already has operational databases (usually only about 100% of the time), then, there is a promotion capability from the operational data model to the implemented data model. Once an implemented data model is highlighted and the promotion button is pressed, the complete implemented data model is "promoted" up to be a specified data model.

The document, *Data Modeler Architecture and Concept of Operations*, which can be downloaded from the Whitemarsh website, www.wiscorp.com is an essential prerequisite reading for the correct use of this data modeler component. It presents the "business problem" being addressed. This user guide only briefly presents how to accomplish the solution.

Presumed Knowledge

This user guide, and all the other metabase user guides presume that the reader has read and is completely familiar with the following documents: Metabase Common Processes, and Metabase Bill of Materials and Single File Recursion (BOM/SFR Guide). These two documents serve as metabase teaching guides for processes that commonly occur throughout the metabase system.

Metabase Example

The metabase example, Movies, is a complete example of a business which is available from the Whitemarsh website. The Movies Rental Corporation was modeled after the largest movies rental corporation in the United States. As such, the example has national, regional, and retail

outlets. There are two data models, one for an original data capture, store based system, and another which is a data warehouse for rented movies.

2 Software Installation

Metabase installation is explained in the Metabase Administrators Guide.

3 Database Design

The meta model of the implemented data model module is depicted in Figure 1.

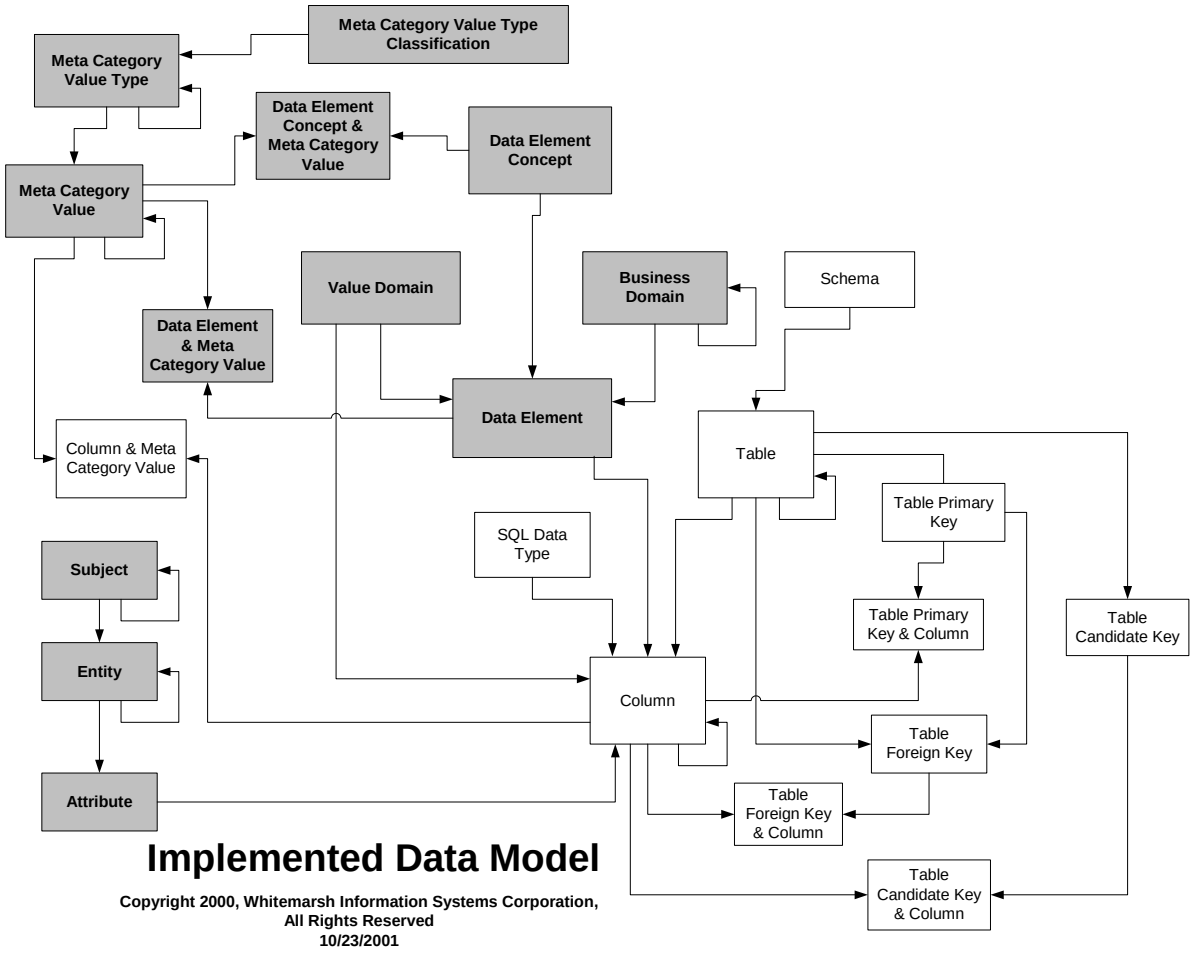

Figure 1. Implemented data model meta model

The database design contains the following tables:

- Column
- Column & Meta Category Value
- Schema
- SQL Data Type
- Table
- Table Candidate Key
- Table Candidate Key & Column
- Table Foreign Key
- Table Foreign Key and Column
- Table Primary Key
- Table Primary Key & Column

Explicit in this database design are the following:

- Columns are the manifestation of the semantics of a data element within a table of a schema. Additionally, a column is a deployment of the semantics of an attribute. Columns may have additional semantics that further refine the column within th e context of either the attribute or the data element. The order of processing these additional semantics is that the column must first be a subset of the attribute, which in turn must be a subset of the data element. Not all the columns of a table must map to attributes from a single entity.

- Column value domains are collections of a more refined set of value domains for specific columns. Column value domains are a subset of attribute value domains.

- Column & meta category values are the relationship that exists between an column and its set of assigned meta category value semantics. These assigned semantics are always a subset of those assigned to the column's "parent" data element and also a subset of those assigned to the column's "parent" attribute. The order of processing these additional semantics is that the column must first be a subset of the attribute, which in turn must be a subset of the data element.

- A table is intended to be a well defined expression of one policy within a schema. Ideally, the collection of all the tables within a schema area should define a coherent collection policy. Although unlikely, some tables and even some schemas may never be represented within operational data models. Additionally, some columns within an table may never be employed. A table may contains columns that map to attributes from multiple entities. Tables can be sub-typed.

- Table candidate keys represent a collection of columns within an table that when their values are collectively employed would result in the retrieval or update of a single row of data for that table. There may be multiple candidate keys within an

table. Columns of candidate keys are not allowed to overlap each other or the table's primary key.

- Table candidate key & columns are the relationship between an table candidate key and the columns that comprise the key. The columns exist within a specified sequence. Candidate key columns are not allowed to include any columns within the table's primary key.

- Table foreign keys represent a related table's primary key. The name of the foreign key should match closely the relationship that the key is to represent. The columns of the foreign key should be able to be deleted entirely from the table without any loss of policy. The columns of the foreign key are not allowed to overlap the columns of the table's primary key. In addition to the foreign key's columns there are additional rules governing inserts, updates, and deletes.

- Table foreign key & columns are the relationship between an table foreign key and the columns that comprise the key. The columns exist within a specified sequence. Foreign key columns are not allowed to include any columns within the table's primary key.

- Table primary keys represent a collection of columns within an table that when their values are collectively employed would result in the retrieval or update of a single row of data for that table if that table had actually been a table. There can only be one primary key within an table. Columns of primary key are not allowed to overlap each other or the table's candidate key.

- Table primary key & column are the relationship between an table candidate key and the columns that comprise the key. The columns exist within a specified sequence. Candidate key columns are not allowed to include any columns within the table's primary key.

- Schemas represent a database structure of tables and relationships within the enterprise. Implemented data models data models are cast within the domain of a schema. The set of all tables within a schema is not required to be taken from a single set of entities within a subject area.

- SQL data type is a classification of the values represented by a column of a row of data. The data types common represented are character, integer, binary, and the like. Each SQL data type imposes a set of rules regarding allowable values and allowed operations on the values. For example adding an integer value to a date value, but disallowing the adding of two dates.

4 Reference Data

The reference data in the implemented data model consists of the SQL Data Type. Readers are encouraged to thoroughly review and understand the Data Modeler Architecture and Concept of Operations book that is available from the Whitemarsh website, www.wiscorp.com.

5 Operation

Once the application is installed it is ready to use. Just invoke the software from the metabase program. The application is a traditional windows application. Metabase reports are accomplished through any ODBC class report writer such as Crystal Reports.

5.1 Log In Process

Figure 2 shows the log-in screen that appears immediately after the application is started. Entered is your user name and your password. These are created by the Metabase Administrator through the metabase administration module. Please contact your metabase administrator to set up your user name and password. Once a user name and password is established, all the user's information can be changed by the user through a restricted use version of the administrator software. Once the send button is pressed the specific metabase database instances that can be accessed by the user is presented. The metabase is such that users are allowed to use specific metabase instances and specific metabase modules.

In this particular example, the user, once they sent their user name and password are shown the metabase database that they can access, that is, Movies. Highlight the choice and press the Select button. Once that is done then the metabase name, Movies, is shown as the data set that is being accessed.

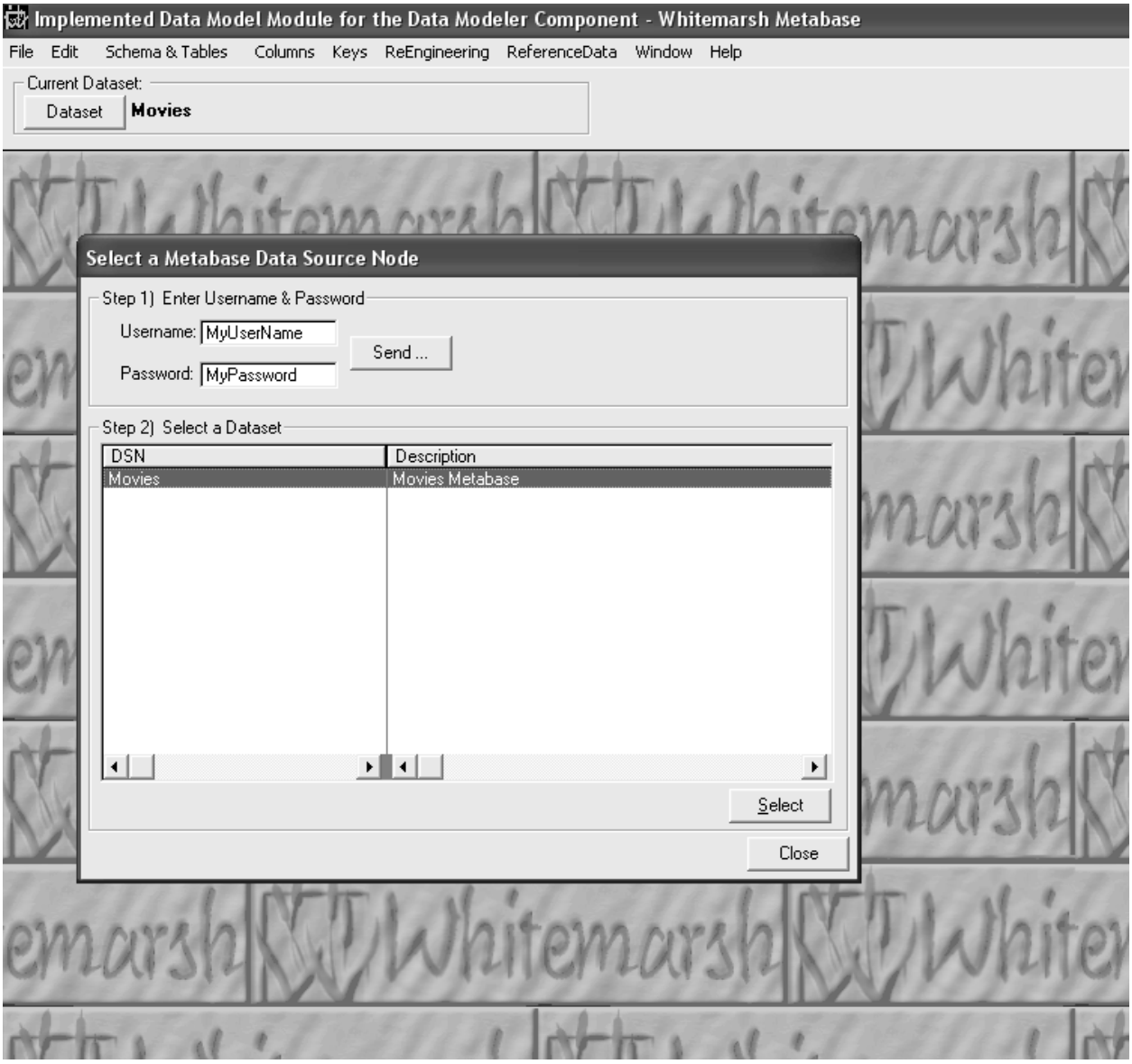

Figure 2. Login process.

6 Process Model

The implemented data modeler process consists of three classes of processes:

- Reference Data
- Fact Data Entry
- Reports

The top level menu for implemented data model contains the following top level items:

- Schemas & Tables
- Columns
- Keys
- Reference Data
- Reports

Each menu item contains as appropriate, nested subordinate menu items. The complete menu is presented in the table that follows. When a actual process is activated, its existing list is presented. To add, change or delete an item on the browse list, the Insert, Change, or Delete button is pressed. The form that is then presented supports the entry of all the data that is needed.

-- Schema Tables
    -- Schema
    -- Table
    -- Data Model Tree

-- Import And Export

-- Import From Specified Data Model
    -- Import Subject Entity Set
    -- Import Entity Tree from Subject
    -- Import Single Entity from Subject
    -- Import Attributes from Subject

-- SQL DDL
    -- Import
    -- Export

-- Columns
    -- Create One Column
    -- Create Many Columns
    -- Data Hierarchies

-- Maintenance
    -- Columns Maintenance
    -- Column Value Domains
    -- Column Meta Category Value

-- Keys
    -- Table Primary Key
    -- Table Primary Key Columns
    -- Table Foreign Keys
    -- Table Candidate Key
    -- Table Candidate Key Column

-- ReEngineering
    -- Reassign Columns to Attributes
    -- Reassign Columns to Data Elements
    -- Synchronize Columns, Attributes, and Data Elements
    -- Reassign Columns to SQL Data Types
    -- Reassign Columns to Tables
    -- Synchronize Local Definitions
    -- Reassign Tables to Schemas
    -- Reassign Tables to Tables
    -- Promote Implemented Data Model to Specified Data Model
    -- Promote Implemented Data Model Table to Specified Data Model
    -- Promote Column to Data Element
    -- Remove Column Meta Category Values
    -- Remove Column Attribute Assignments

-- ReferenceData
    -- SQL Column Data Type

Menu for Implemented Data Model

6.1 Reference Data

Figure 3 presents the list of SQL data types. To insert, change or delete a SQL data type press the appropriate button. Figure 4 presents the SQL data type update screen.

6.2 Fact Data

The fact data consists of:

- Schema Tables
- Columns
- Keys
- Reference Data

The one reference data item, SQL data types is addressed in Section 6.1.

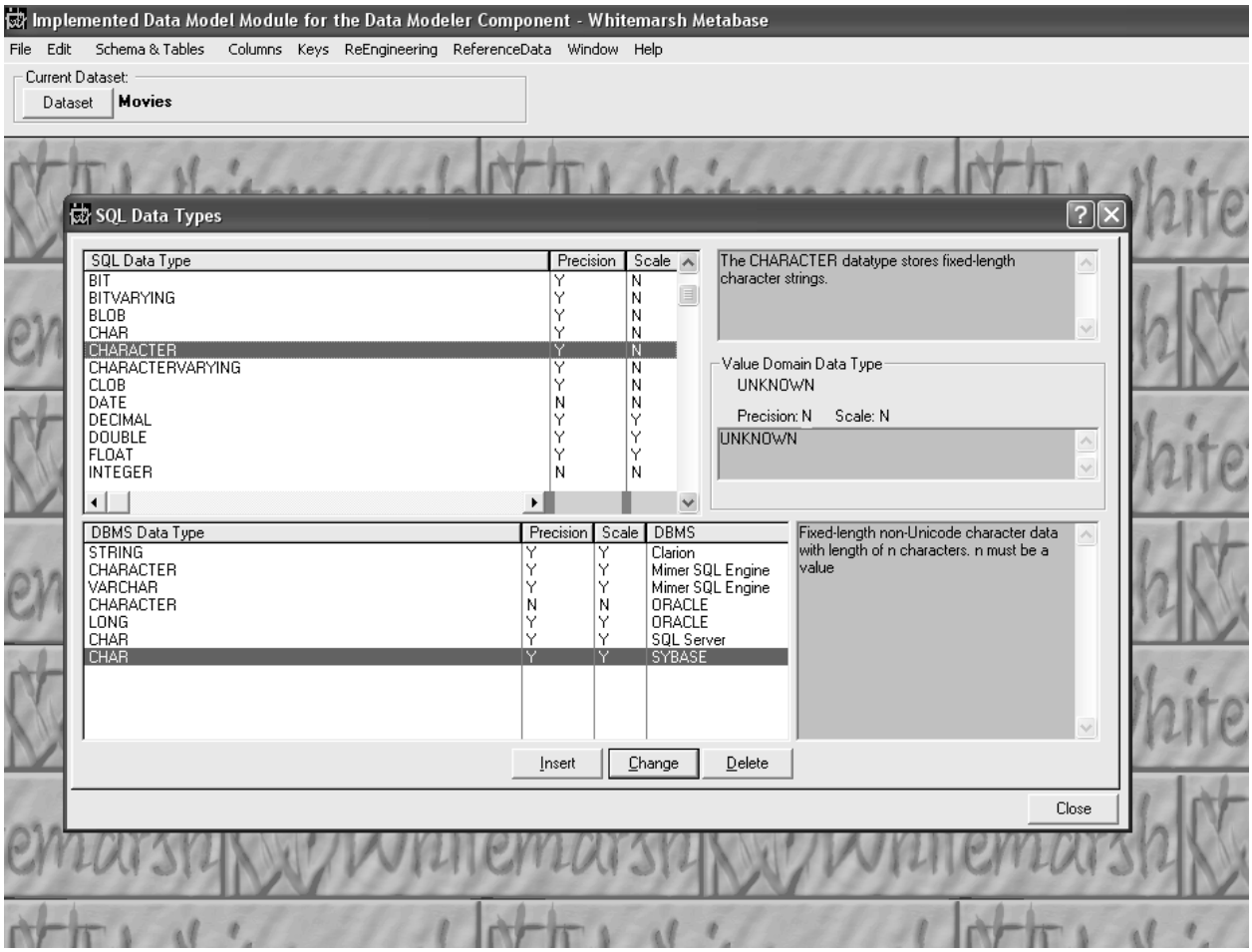

Figure 3. SQL Data Types.

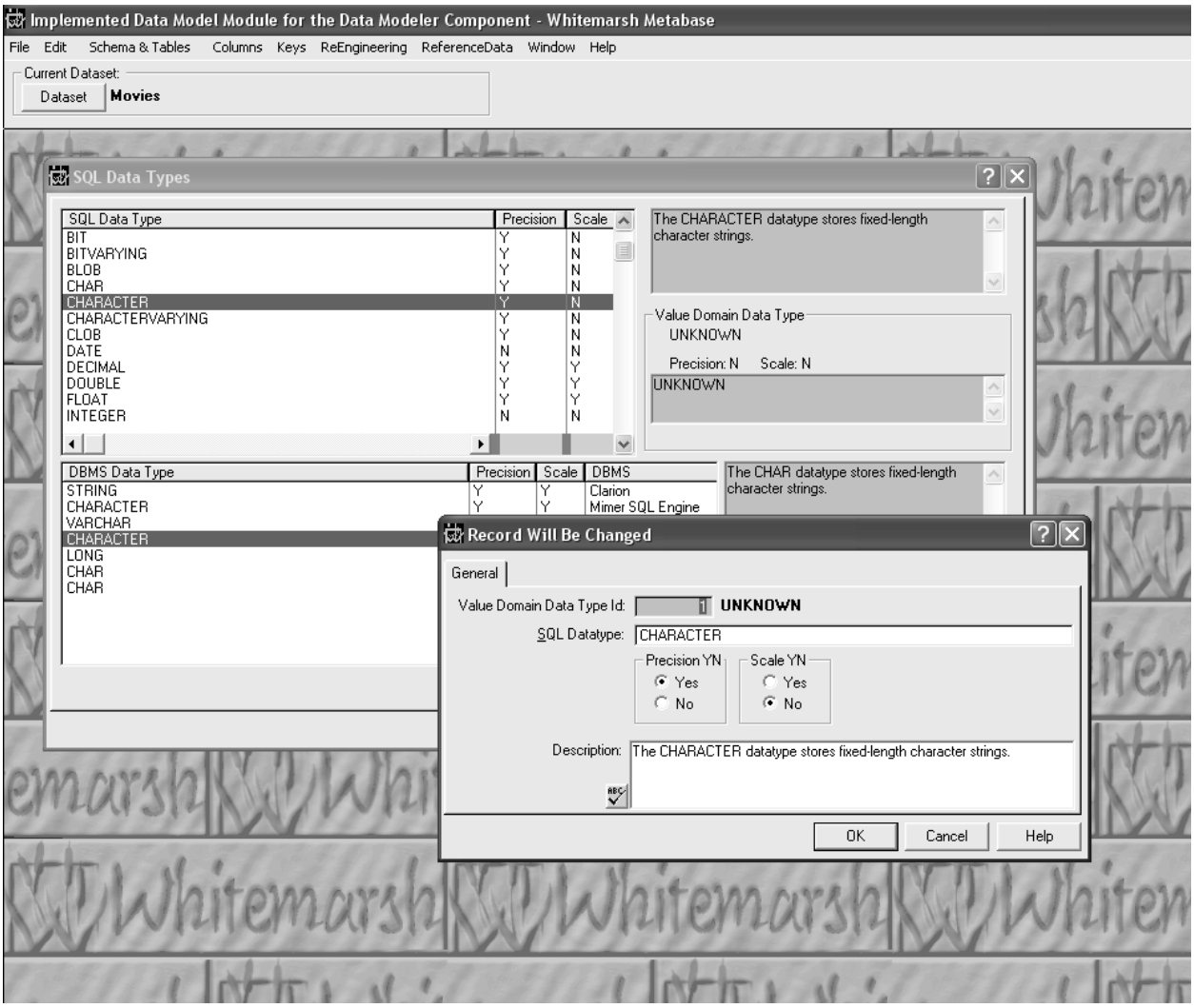

Figure 4. SQL Data Type update screen.

6.2.1 Schema Tables

The Schemas Tables processes enable the entry and update of the main components of a implemented data model. It consists of the following:

- Schema
- Table
- Import Entities

6.2.1.1 Schema

Schemas with respect to the implemented data model are expressions of enterprise policy within a defined database. Figure 5 presents a list of schema. This list shows an "unknown" schema. This is necessary so that when one or more DBMS tables, DBMS columns and the like are promoted to be an implemented model set of tables and columns, there is a valid foreign key identifier for the newly created table, albeit "unknown." Once these are created then the reverse engineering process of changing the schema for a table can be accomplished.

If a entirely new schema is to be entered, press Insert. A screen like Figure 6 is presented. The schema's name, abbreviations, and description can then be entered.

6.2.1.2 Tables

An table is a self contained aspect of policy within a schema. The columns within the table provide for the values that instantiate the policy. A collection of tables within the same schema area provide a comprehensive view of the schema's policy.

Figure 7 presents a list of tables. Within the schema, customer management, there are four tables. Within the highlighted address table are a collection of columns. The button, Delete All Tables for Schema, will accomplish what the button implies. It also deletes all columns and keys. The button, AutoMake Primary Key will create not only the Primary Key but also a column of the Form <table name> ID. Figure 7 also shows the local and contextual definitions for the table.

To add a new table, highlight the containing schema and then press the Insert button. A screen like Figure 8 is displayed. A table can be created underneath the "unknown" schema and then re-assigned later.

Also shown in Figure 8 is the local and contextual definitions. This is where you make the local definition and the contextual definition. As stated above, just make a simple phrase for the local definition. When the AutoDef button is pressed then all the contextual parts of the entities definition are gathered and employed in a more comprehensive description of the table

The table's name, abbreviations, and description can be added or changed. The columns associated with an table are addressed in Section 6.2.2.

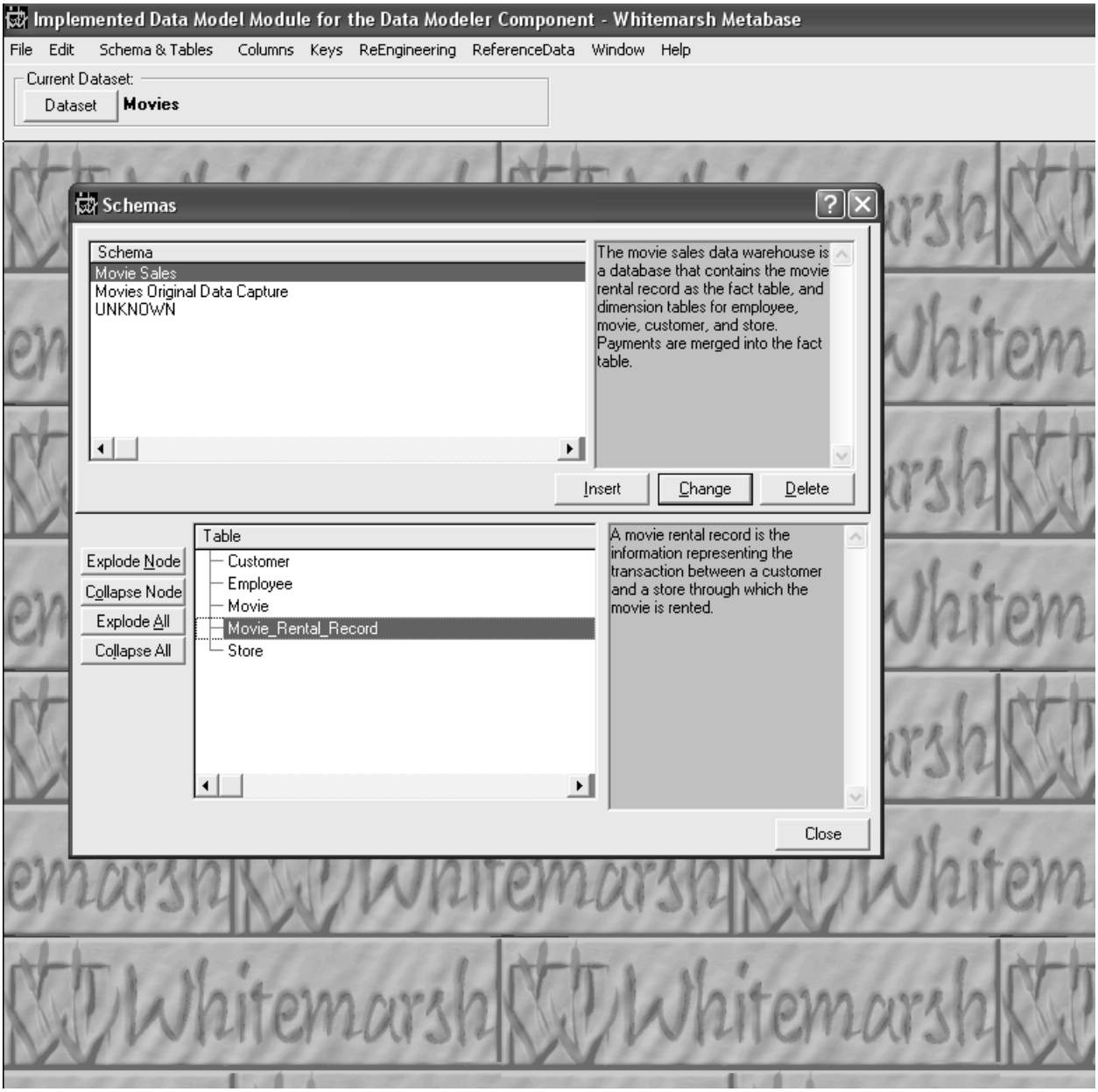

Figure 5. Schemas.

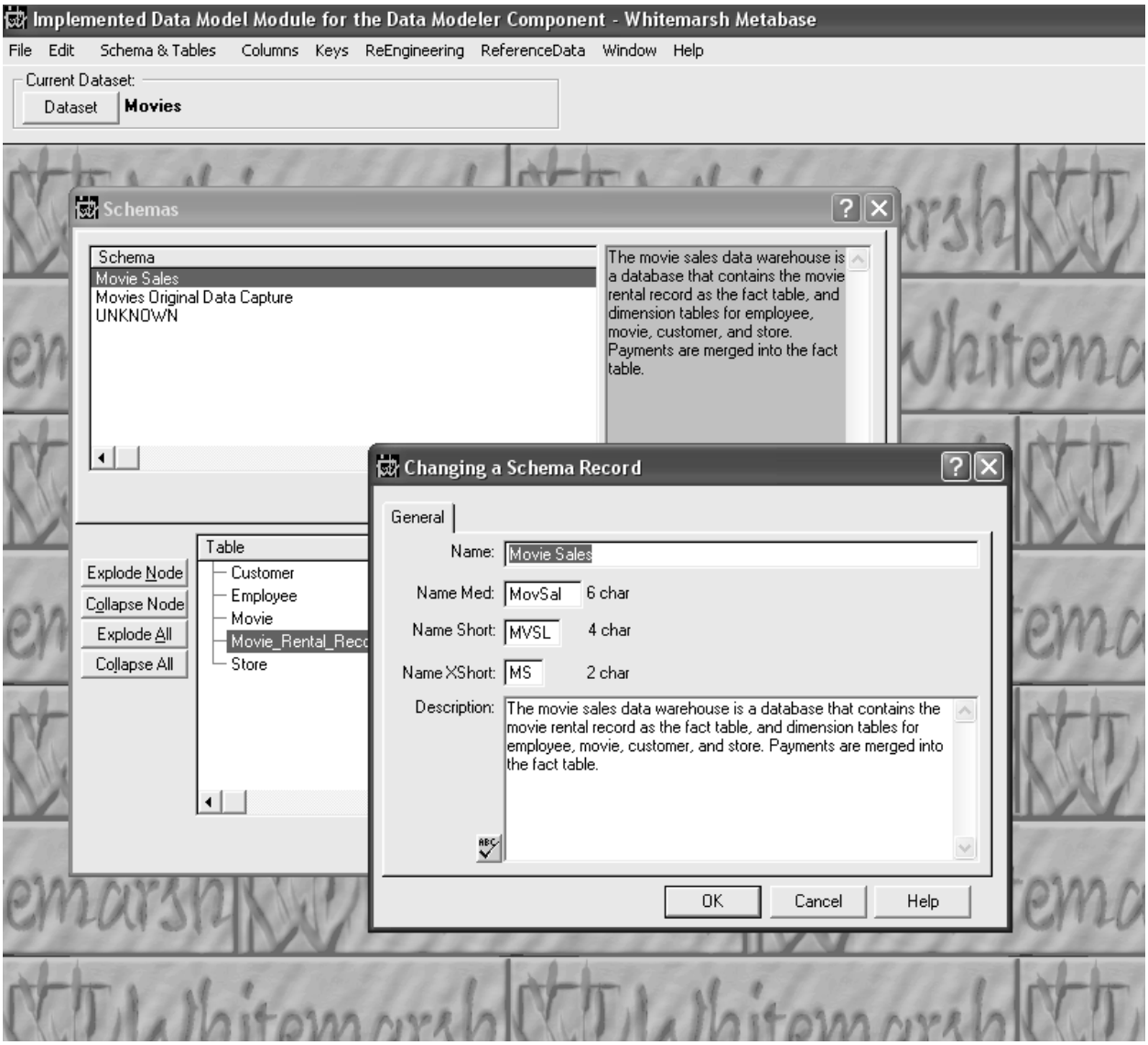

Figure 6. Schema update screen.

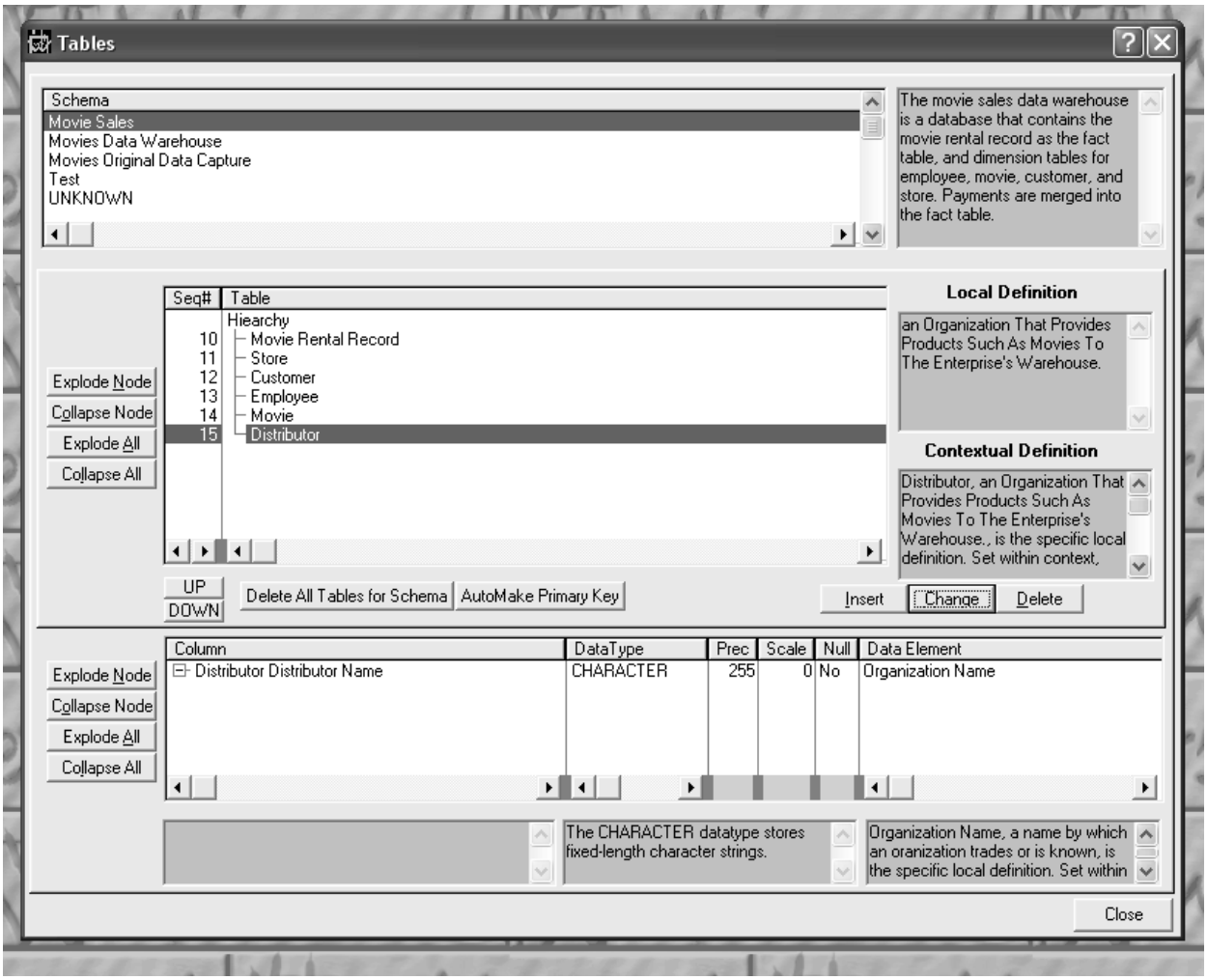

Figure 7. Tables within schema.

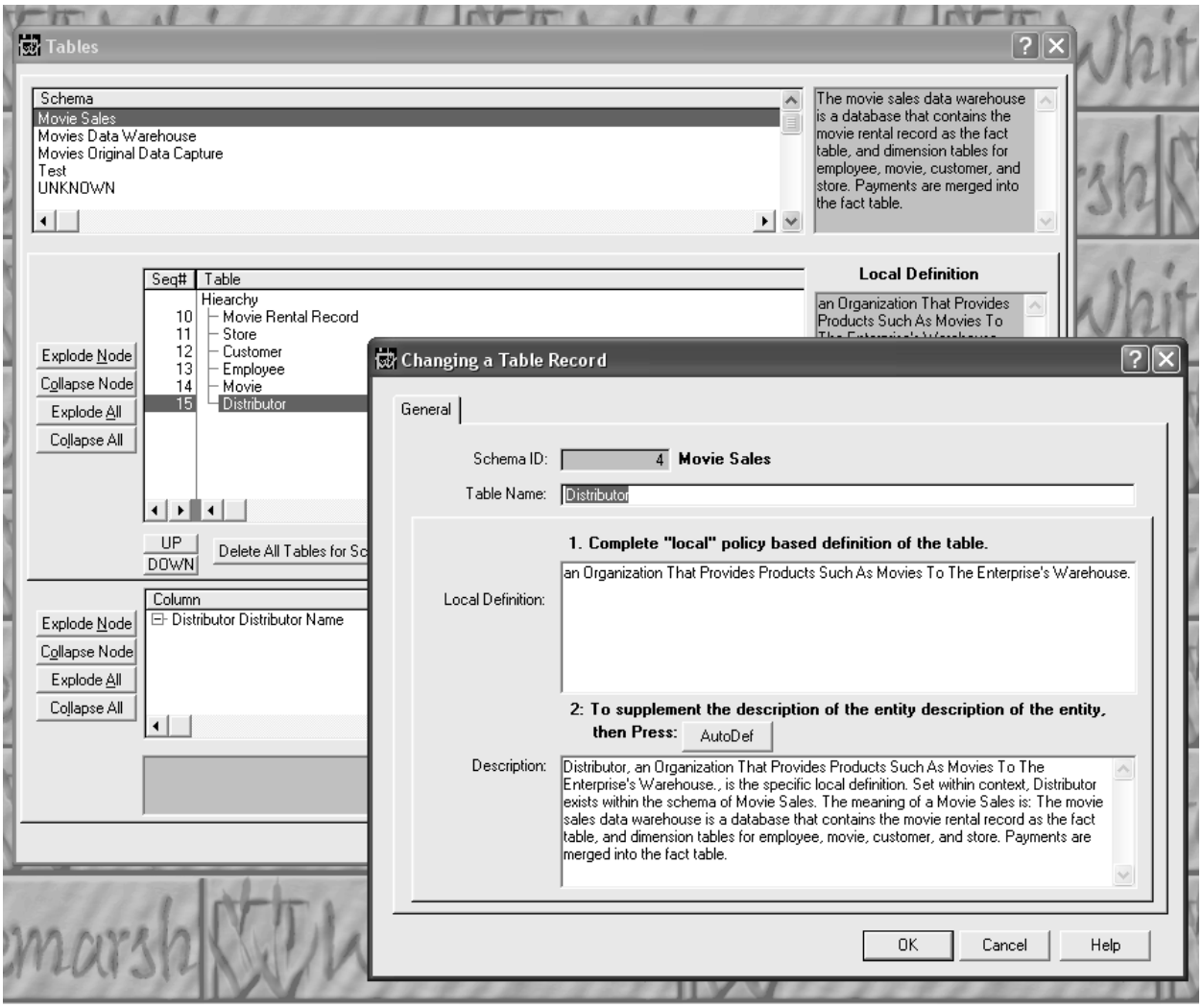

Figure 8. Table update screen.

6.2.1.3 Import Entities

The process of importing entities from the specified data model to make implemented data model tables is presented in four scenarios:

- Import Subject Entity Set
- Import Entity Tree
- Import Single Entity
- Import Attributes from an Entity

6.2.1.3.1 Import Subject Entity Set

The import subject entity set is depicted in Figure 9. First highlight the subject. The entities within that subject are then displayed. Then select a target schema. If the target schema is not present, then create it through the Insert button. Finally, press the Import button. What will be imported with be JUST the entities within that subject area. In this case, just the movie and movie copy would be imported.

All keys, primary, candidate, and foreign are imported as well. If however an entity has foreign keys that reference entities outside the domain of the subject, then those foreign keys and their columns are not imported. If the foreign key columns participate in the primary key of an imported entity then those columns are imported.

The last step that is taken is that all newly created columns are “pointed back” to the attributes.

If multiple subject areas are imported into the implemented data model through this process, they are obviously unconnected. Sorry, but there’s no magic. Additional foreign key relationships have to be built.

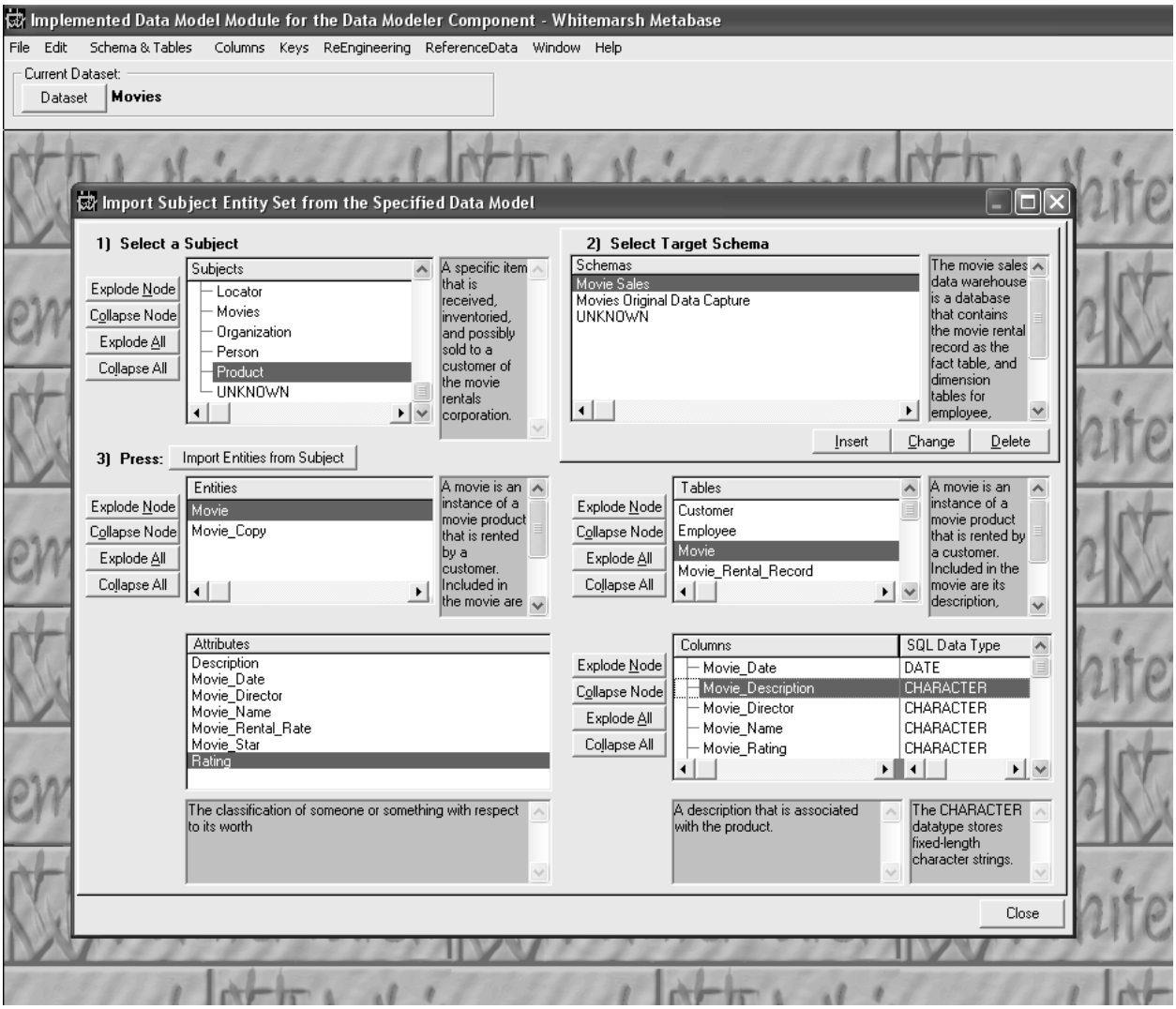

Figure 9. Import Subject Entity set.

6.2.1.3.2 Import Entity Tree

The import entity tree process is depicted in Figure 10. First highlight the subject, then the entity that is at the top of a hierarchy. To ensure that you have an apex entity, highlight what seems to be the apex entity and then press the button, Ancestor and Descendent tree. Then press the Display Tree. The bottom left window displays the built tree. The entity that is black is the entity that is the root entity of the displayed tree. Descendent entities are in blue, ancestor entities are in red, and subtyped entities are in cyan. If all the displayed tables are in blue then the one you picked is an apex table. You can however import a tree which has both ancestors and decedents.

You can also more fully display all the entities, attributes and keys across the entities of the tree by pressing the Display Data Model Tree button. The data model tree is presented in Figure 11.

When a target schema is able to be selected, highlight the target schema and the press the Import button. If the target schema is not present, then create it through the Insert button. Finally, press the Import Tree button. Built will be all the implemented data model tables, columns, and relationships. All the mapping between the newly created implemented data model and the specified data model will be created as well. The only entities and attributes that will be imported are those in a direct ancestor-descendent relationship of the displayed tree root node. Skipped are foreign keys and attributes of foreign keys that are not in a direct line with the root table.

If multiple specified data model trees are imported into the implemented data model through this process, they are obviously unconnected. Sorry, but there's no magic. Additional foreign key relationships will have to be built between as primary key of one newly created table from one specified data model tree and another of the newly created table of a different data model tree.

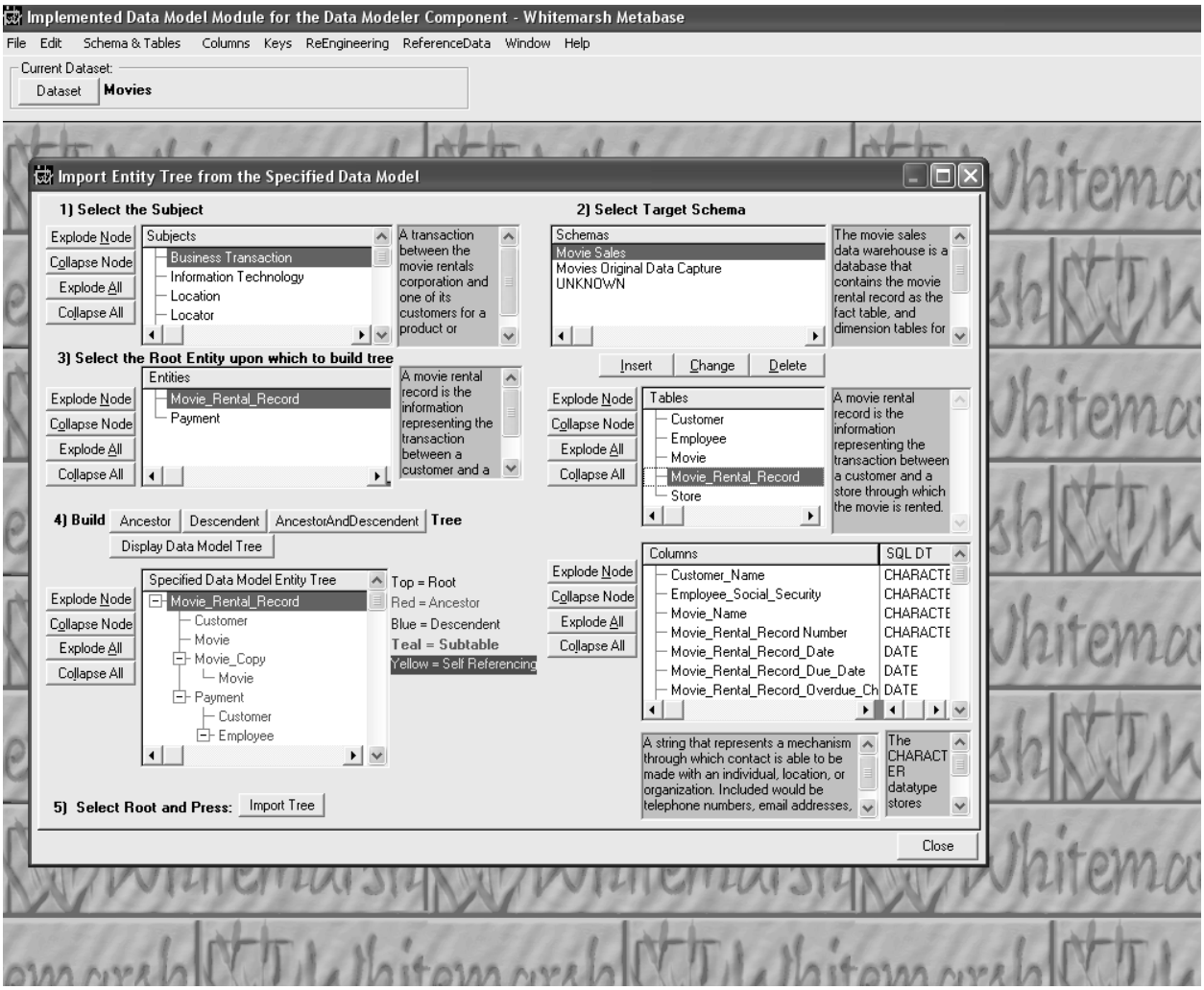

Figure 10. Importing an Entity Tree.

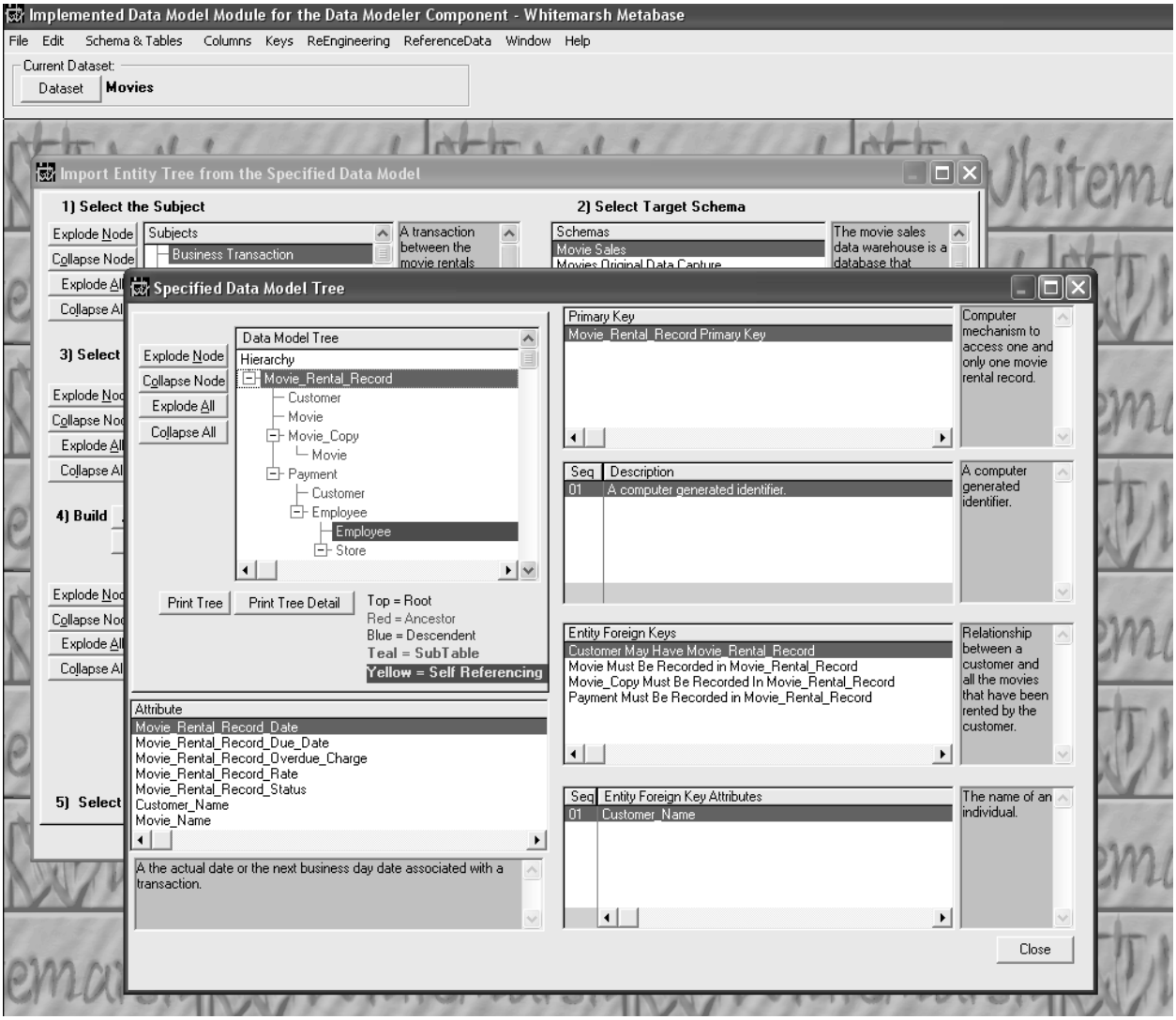

Figure 11. Display of an Entity tree.

6.2.1.3.3 Import Single Entity

The screen for importing a single entity is depicted in Figure 12. First highlight the subject, then the entity that is at the top of a hierarchy. Now, on the right side of the window, highlight the target schema. If none are shown then create one.

When a target schema is able to be selected, highlight it and the press the Import button. Built will be the implemented data model table, columns, and relationships. All the mapping between the newly created implemented data model and the specified data model will be created as well.

The entity that is imported into the implemented data model through this process is obviously unconnected. Sorry, but there's no magic. Additional foreign key relationships will have to be built between the created table and existing tables.

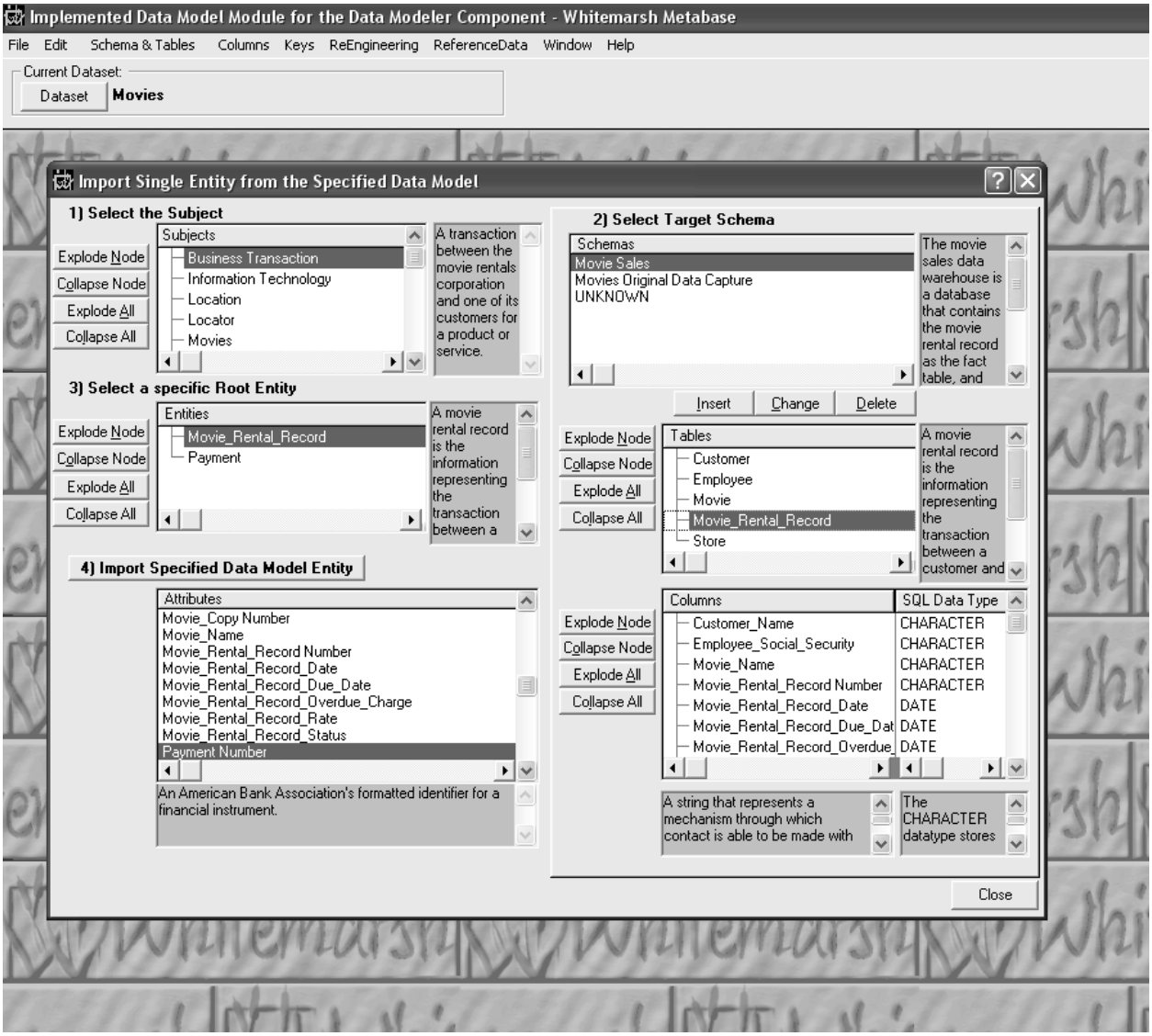

Figure 12. Importing a single Entity.

6.2.1.3.4 Import Attributes from an Entity

The screen for importing one or more attributes is depicted in Figure 13. First highlight the subject, then the entity. Then tag one or more attributes. On the right side, select the schema and then tag ONE table. It is the table into which the attributes will be imported. All the mapping between the newly created attributes and the specified data model entity attributes will be created as well.

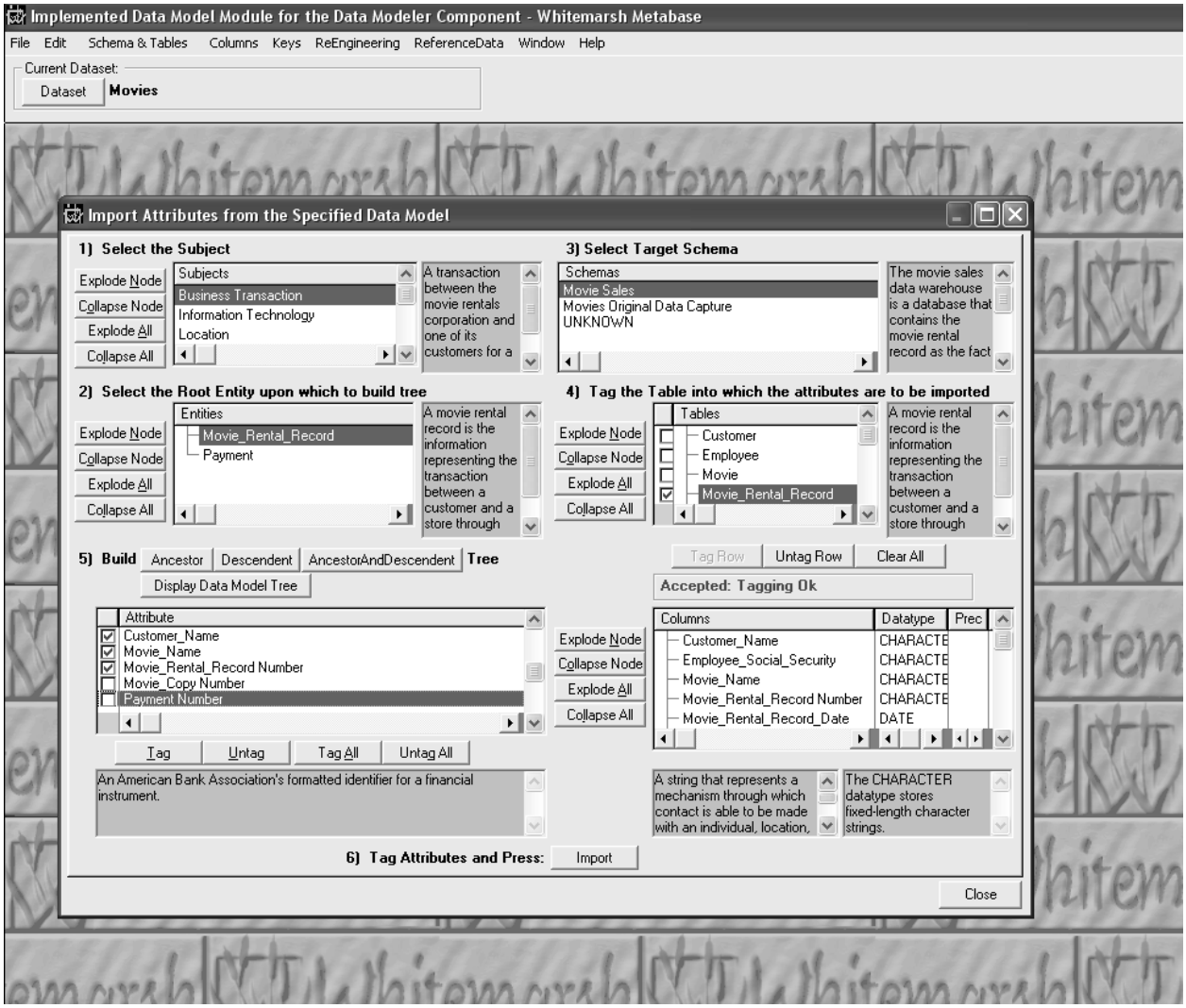

Figure 13. Import Attributes.

6.2.2 Columns

Columns are the value based characteristics of an table. Included in the definition of columns are:

- Creation
- Maintenance
- Data Hierarchies

6.2.2.1 Creation

Figure 14 presents the screen for creating columns. From the point of view of the implemented data model, an column is really an "intersection record" between data element and table. First highlight the subject then the table. Tag the appropriate table. Then highlight the business domain, and then tag one or more data elements. Once an table and one or more data elements are tagged, press the Build button. The underlying process then creates column table rows within the highlighted table. Each newly created column is mapped to the previously tagged data element. The complete set of columns for the table are then displayed in the bottom window. Once columns are created through this process they are related to the "unknown" attribute. To reassign columns to a known attribute (and by inference to an entity and subject) employ the *Assign Columns to Attribute* process.

Figure 15 presents the process for creating one column within multiple tables. The result is many columns. Each is set within the context of the containing table. To accomplish this, select the business domain, then tag one data element. Then select a schema and then tag one or more tables. When the Build button is pressed the data element, suitably renamed for the context of the table is created in each of the tagged tables. Once columns are created through this process they are related to the "unknown" attribute. To reassign columns to a known attribute (and by inference to an entity and subject) employ the *Assign Columns to Attribute* process.

6.2.2.2 Maintenance

Included in the maintenance of columns are:

- Maintaining columns
- Maintaining column value domains
- Maintaining column meta category values

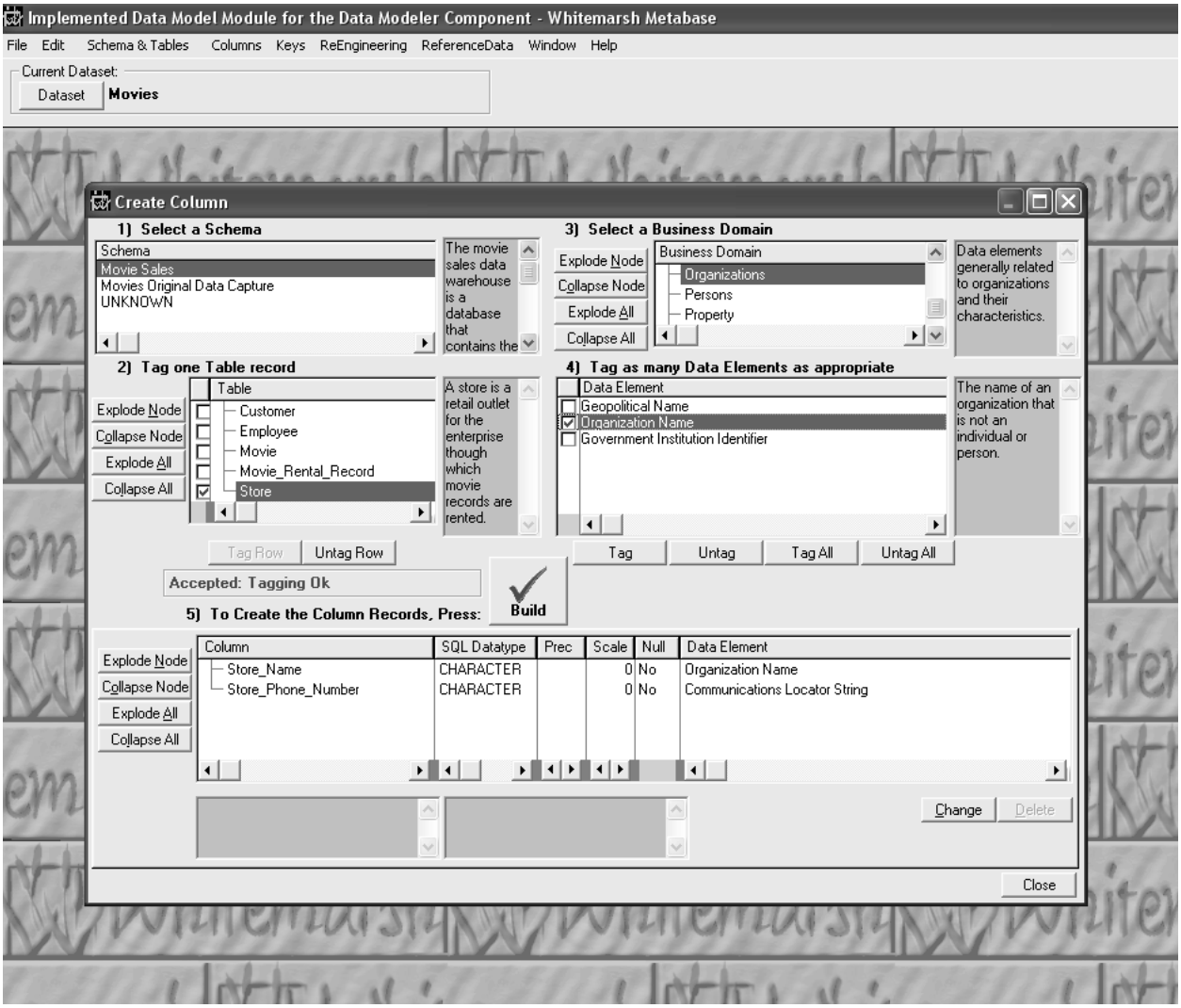

Figure 14. Column creation (one column).

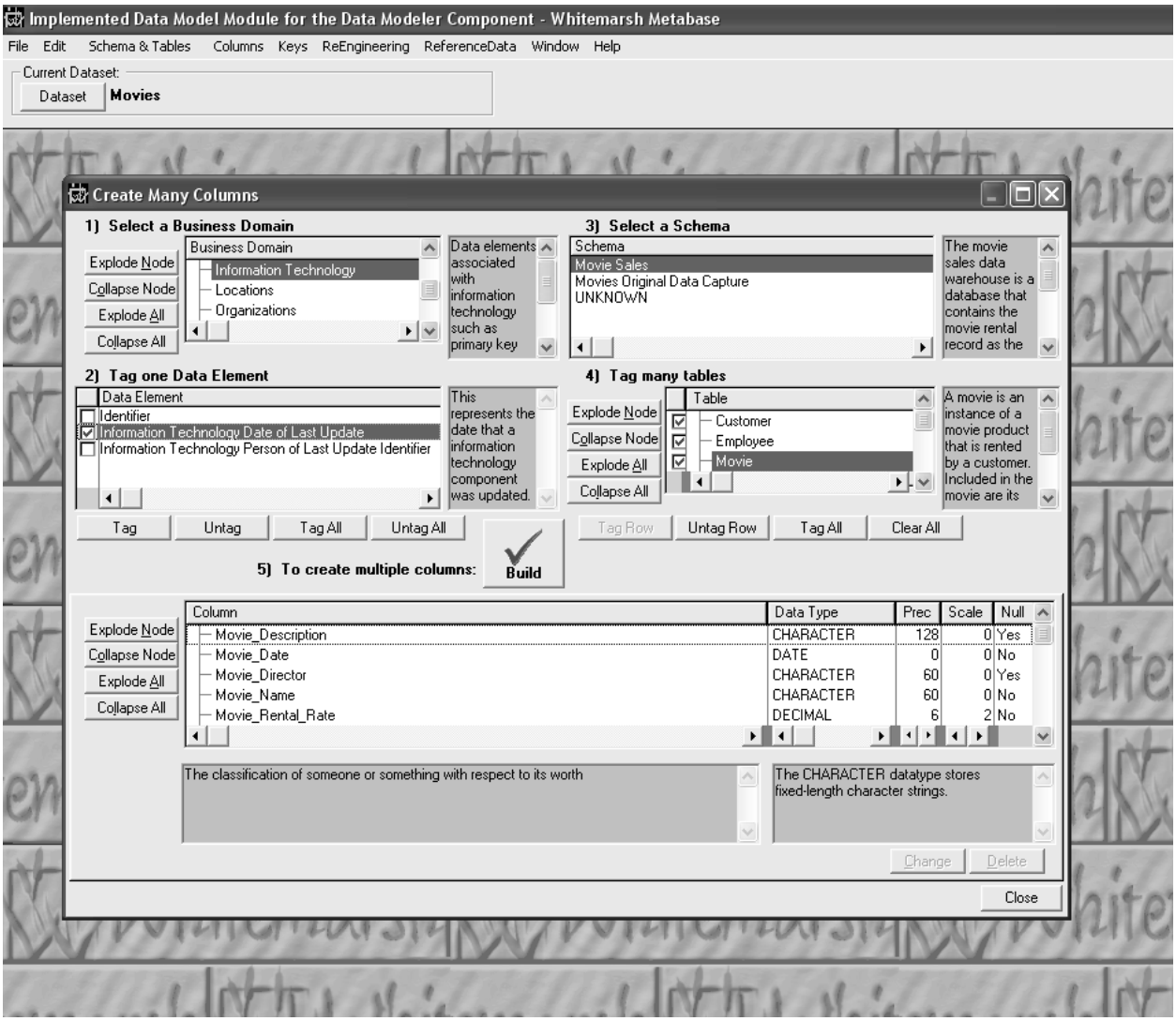

Figure 15. Creating multiple columns in multiple tables.

6.2.2.2.1 Maintain Columns

Maintenance of columns starts with a list of columns. Figure 16 shows the list of columns for highlighted subject and table. Below the highlighted column are all the meta category values that have been assigned. The Print Tree button displays the tree structure of a column if it is complex.

There four buttons on this screen of special interest:

- Replace Underscores with Blanks
- Refresh Selected Table's Fkey...Press
- Refresh the Foreign key ....Press
- Generate definitions.... AutoDef

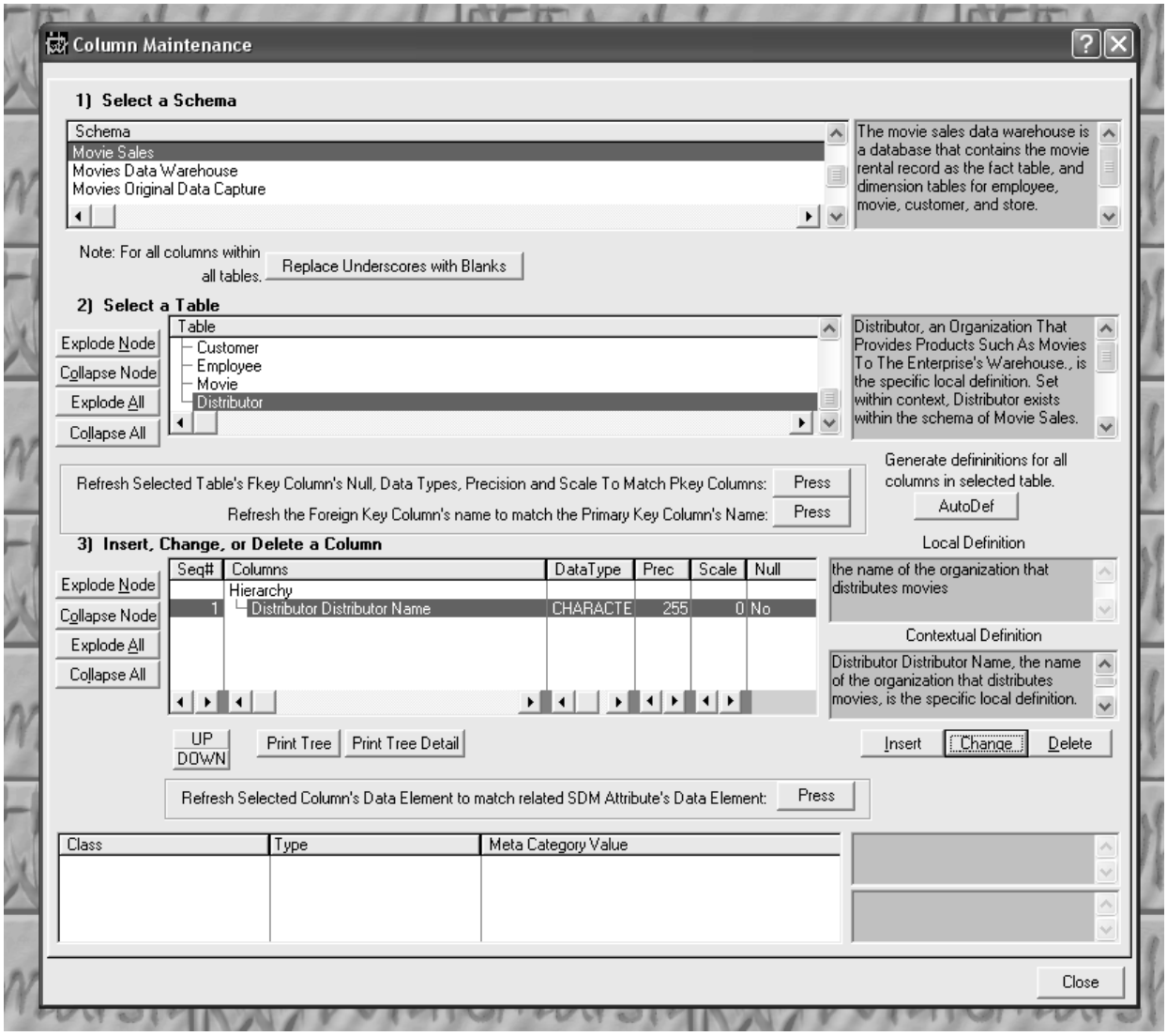

Figure 16. Column maintenance screen.

In the event that you have imported SQL DDL from some outside source into the Implemented Data Model, then underscores are inappropriate. Blanks are much better because at this level of specification, there models are just logical versus bound to a particular DBMS. Underscores are required for the DBMS-based models and the metabase accomplishes that for you automatically. Pressing Replace Underscores with Blanks does exactly what its name says for all attributes of all entities within the Selected schema.

The second button, Refresh Selected Table's Fkey...Press, this refreshes all the columns that participate in a Foreign Key to match those in the source table's Primary Key (Pkey).

The third button, Refresh the Foreign key ....Press, changes the column names in a foreign key to match those in the corresponding Primary Key.

The final button, For all attributes within entity...AutoDef, this causes all the columns in the Selected table to have the AutoDef function execute. The AutoDef process takes the local definition of a table and sets it within the context of all the meta objects that are the "parents" of the table.

If an column is to be changed then press the Change button. Figure 17 is then presented. If the column is not within a foreign key then all the meta columns of the column can be changed. If, however, the column is part of foreign key Figure 18 is displayed with the message that the only meta column that can be changed is its description.

New columns can also be created through Figure 16 by pressing the Insert button. Via this approach the data element must be chosen. Figure 19 presents the list of data elements that is displayed after tabbing off the Data Element Id column. Highlight the business domain and then the appropriate data element. Once the appropriate data element is identified then press the Select button.

Figure 18 also shows the UnAbbrev process. In the event that an SQL data definition language stream is imported, the physical names, that is abbreviated may be what is imported. For example, Empl_Hr_Dt. Under this update process, once the Business Domain has been chosen via the Select Abbreviation Business Domain button, the UnAbbrev button can be pressed. If there has been a good quantity of abbreviations established on various business domains, then the physical name can be automatically translated to a logical name. If there are multiple logical names for a given physical abbreviation, a screen presents itself and an appropriate logical name can be chosen for a given word. In this example, the full name becomes, Employee Hire Date.

Finally, enter the Local Definition phrase. Keep it just a phrase without a capital letter at the start and no period or comma at the end. For a complete contextual definition, then press the AutoDef button.

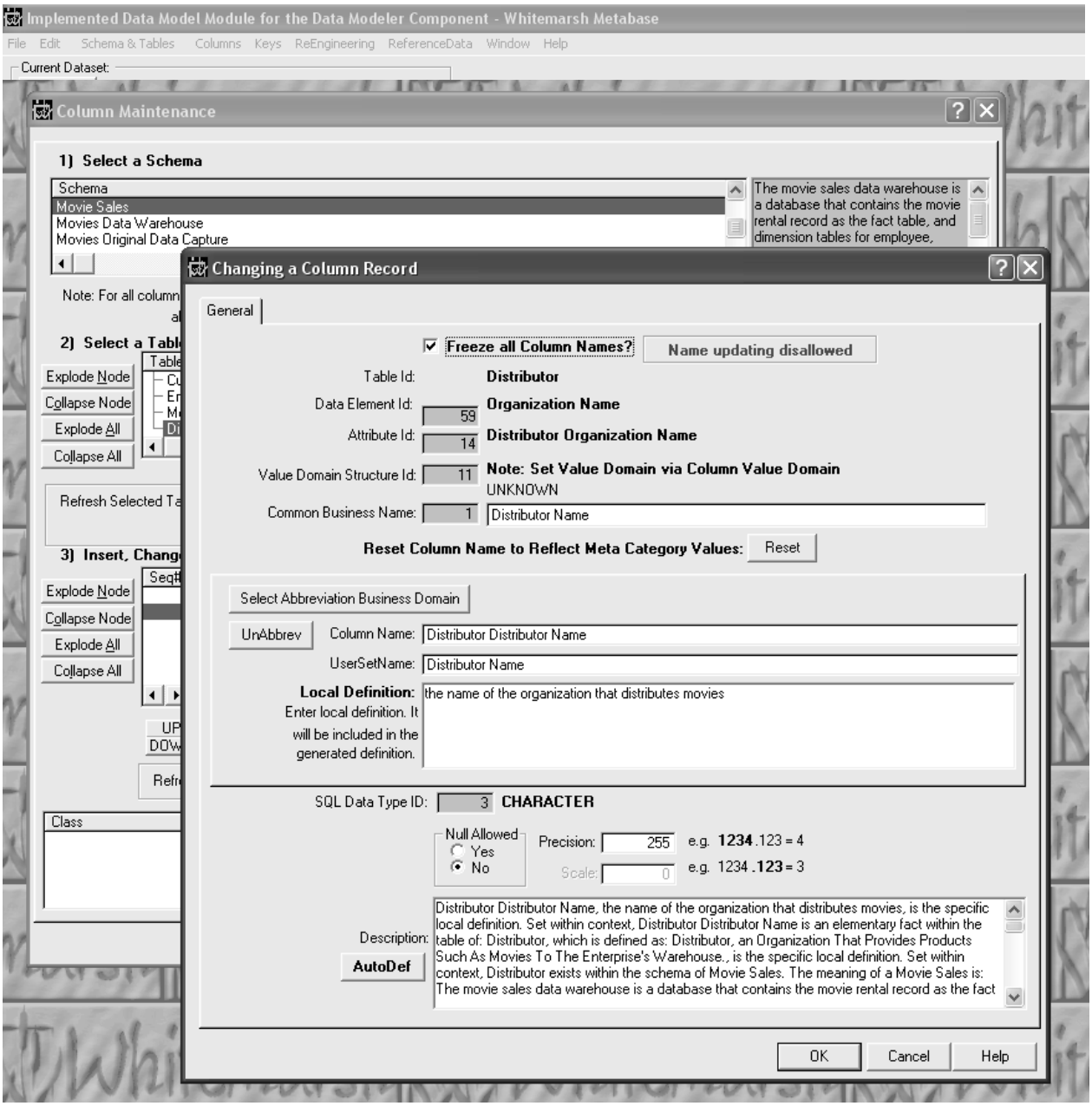

Figure 18. Column update screen.

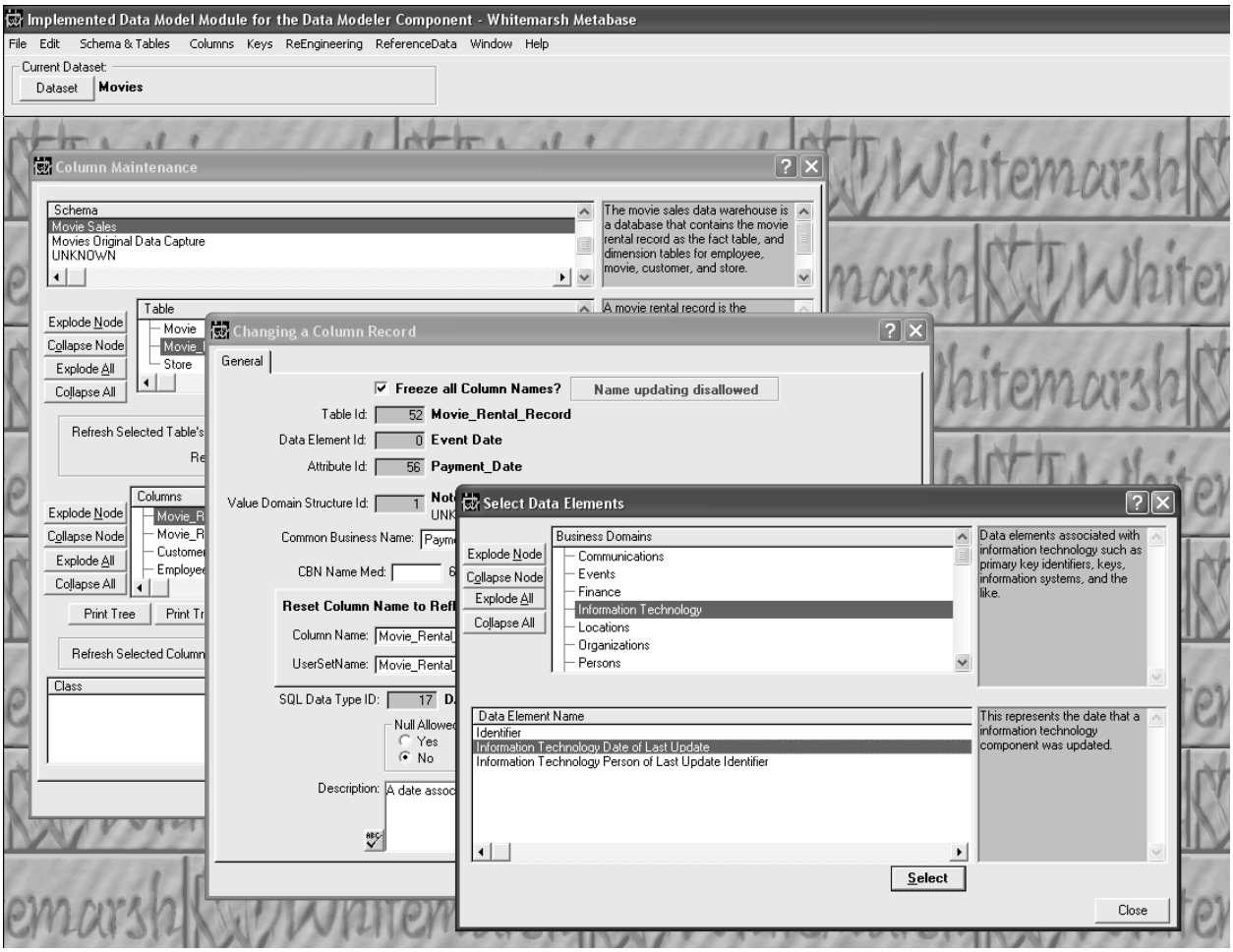

Figure 19. List of selectable Data Elements for a Column update.

6.2.2.2.2 Maintain Column Value Domains

Column value domains exist within the context of data element value domains which in turn exist within the context of data element concept domains and then value domains. Figure 20 presents the screen for viewing and then maintaining column value domains. To see the specific value domain for a column, highlight the schema, table, and then column. The related attribute and data element for the column is then displayed along with any value domains that are allowed to be assigned. If there already are column value domains associated with the column's attribute or data element they are shown.

To assign a value domain, highlight it and press the Select button. If the selected value domain is within the value domain of one already assigned to the column's attribute and data element the assignment is accepted. If the selected value domain is a superset of an already

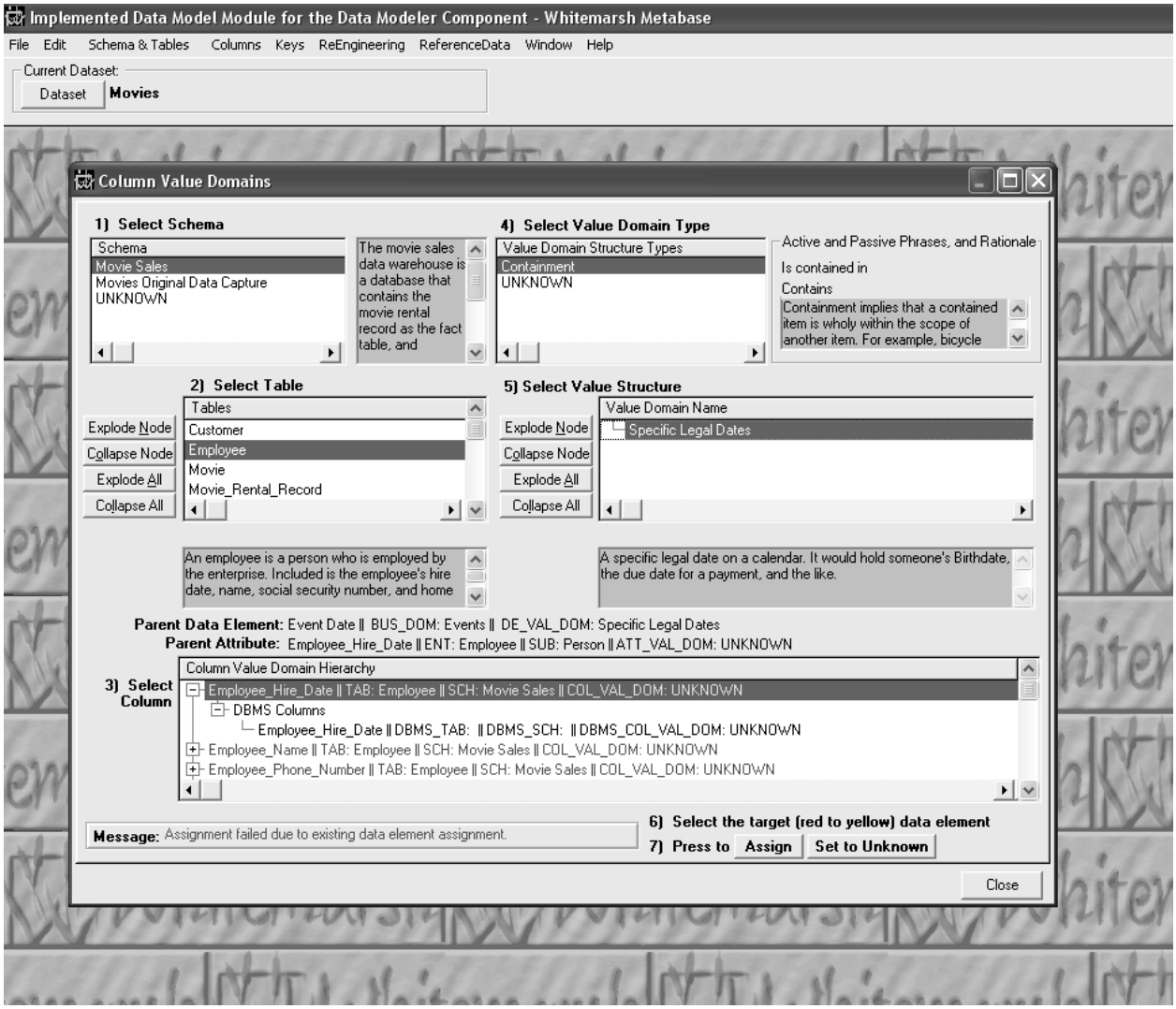

Figure 20. Assigning a Column Value Domain.

assigned value domain the assignment is rejected.

Simply put, the value domains must be hierarchical with the most generalized value at the top and the most narrow value domains at the bottom of the hierarchy. Given that value domains are assigned, then the most generalized would be at the Data Element Concept, then Data Element, then Attribute, then Column, then DBMS Column.

If the column does not have an assigned attribute but does have an assigned data element the assignment is allowed. Similarly, if there is no assigned data element but there is an assigned attribute the assignment is allowed. In the case there is neither an assigned data element nor an assigned attribute then the assignment is rejected.

6.2.2.2.3 Maintain Column Meta Category Values

The semantics of columns are both inherited and assigned. They are inherited through its selected data element. Assigning semantics is illustrated in Figure 21. In this assignment process, the appropriate subject, table, and column is highlighted. If there are any already assigned semantics then they are shown in the bottom browse window. The right two windows contain the set of meta category value types and type classes, and the meta category values for a particular highlighted meta category value. Only one meta category value from each meta category value type can be tagged. For example, it makes no sense to assign both estimated and projected to the same column.

Suffix meta category values will then appear at after the column's name. Prefix meta category values will appear prior to the data element's common business name. The order of appearance is not arbitrary or able to be change by a metabase end-user. The order is specified within the table, meta category type. So, if for example there is a geographic meta category and it's value type's sequence number is 2 and the temporal meta category value type's is 3 then regardless of the sequence of tagging, geographic meta category values will always be prefixed into the column's constructed name before those for the temporal meta category value type.

Once the meta category values are tagged and the build button is pressed, the meta category values are assigned to the column. The assignment process determines one by one that the tagged meta category values are semantic subsets of those already assigned either to a "parent" data element and data element domain. If the tagged semantic is a semantically acceptable then it is assigned. Otherwise the assignment for the offending semantic is rejected.

Once a semantic is assigned, if the column is employed in the creation of a DBMS column, then none of the semantics for the column can be deleted. New ones can be added, but none can be deleted.

When ever semantics are added, the column's name remains the same until it is reset in the column update screen.

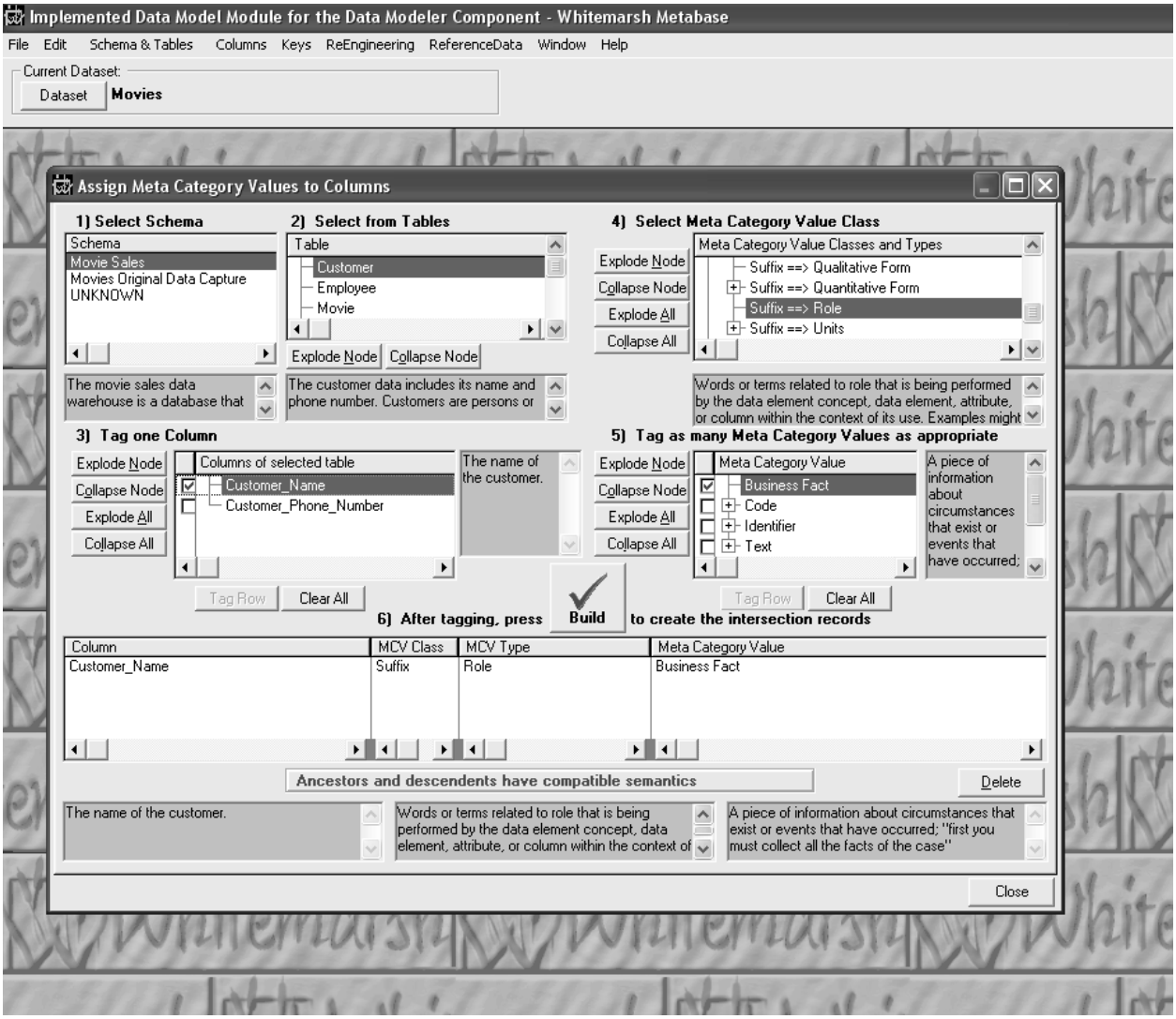

Figure 21. Assigning meta category value to a Column.

6.2.2.3 Data Hierarchies

A key value in the entire set of data modeler modules is the presentation of data hierarchies. For the implemented data model, any highlighted column has only one parent attribute, data element and data element domain. Hence, in Figure 34, they are shown above the traditional browse.

As a browse progresses from one column to the next the respective attribute, data element and data element domain changes as well as the related set of operational data model DBMS columns.

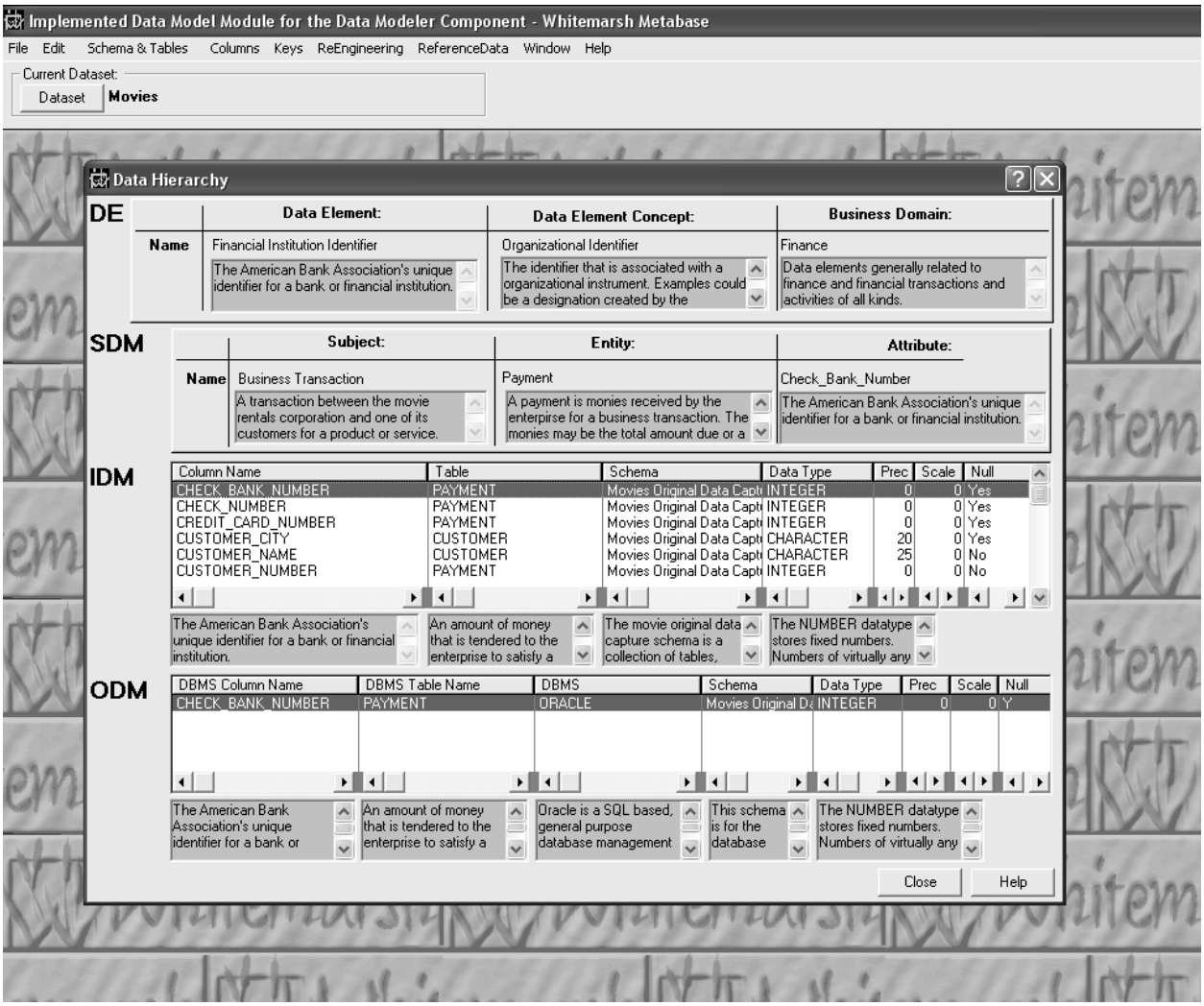

Figure 22. Implemented data model data hierarchies.

6.2.3 Keys

Entities are accessed through the use of keys. The keys included in the implemented data model include:

- Primary
- Foreign
- Candidate

6.2.3.1 Primary

Included in the definition of a complete primary key is:

- Primary key definition
- Allocation of columns to the primary key

6.2.3.1.1 Primary Key Definition

A primary key of an table is a set of one or more columns that represent values that when employed result in only one selected row. Figure 23 shows the current set of primary keys. There can, of course, only be one primary key for each table. To see the columns assigned to a particular primary key, highlight the appropriate subject, and then table. To add a primary key if there is none for an table press Insert.

If a primary key is already defined then an error message is displayed. If the primary key has already been used as the basis of relationships with other tables (that then has the primary key manifest as a foreign key) then a severe warning message is given before the delete operation is permitted to continue. If a delete operation is tried and if the primary key column is employed as a column in the implemented data model the operation fails.

If the Insert or change action succeeds, the Figure 24 is displayed. On an Insert, the name is automatically constructed as the concatenation of the table name and the string, “Primary Key.” The name can be changed. In addition to the name a description can be added or changed.

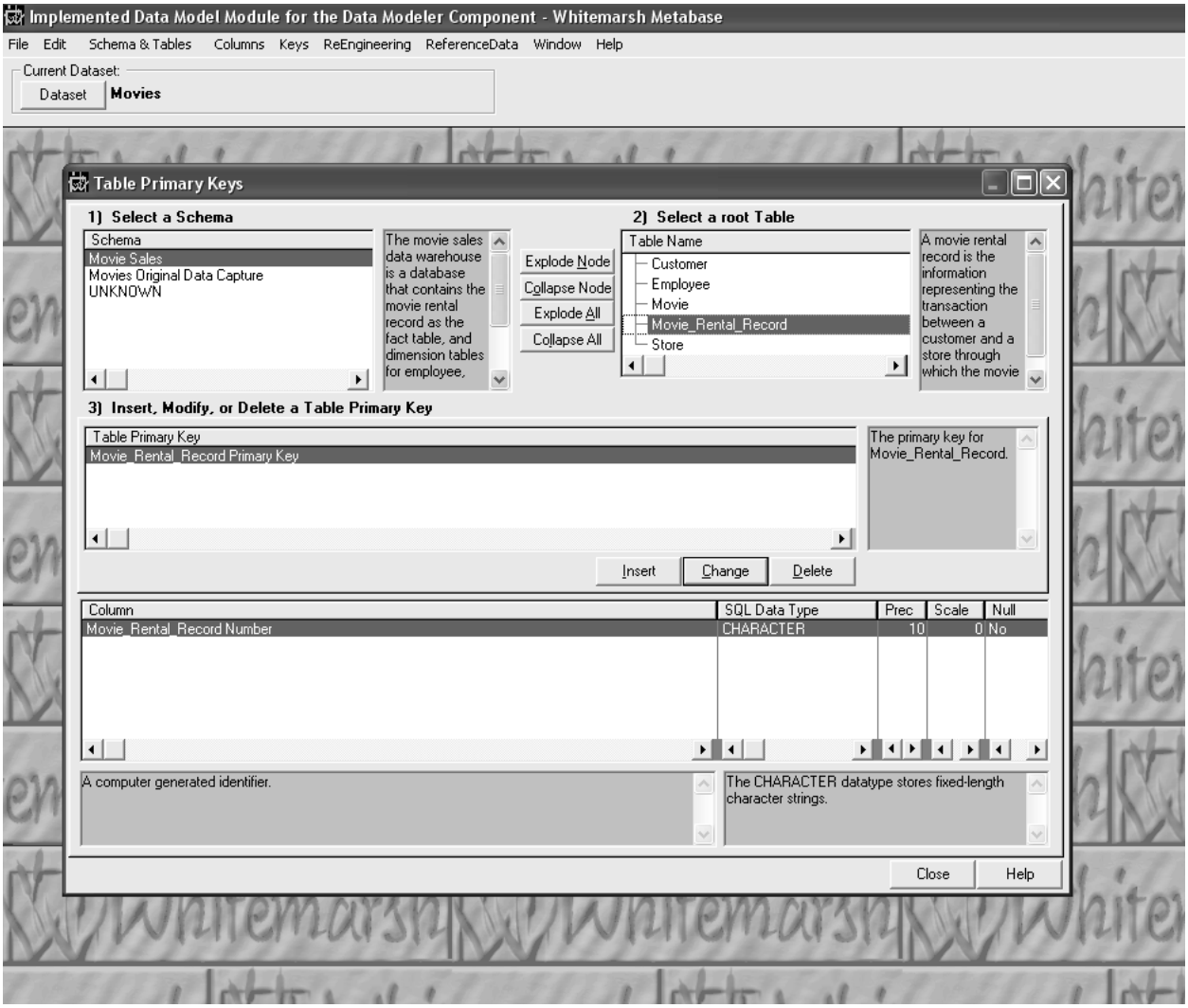

Figure 23. Primary keys.

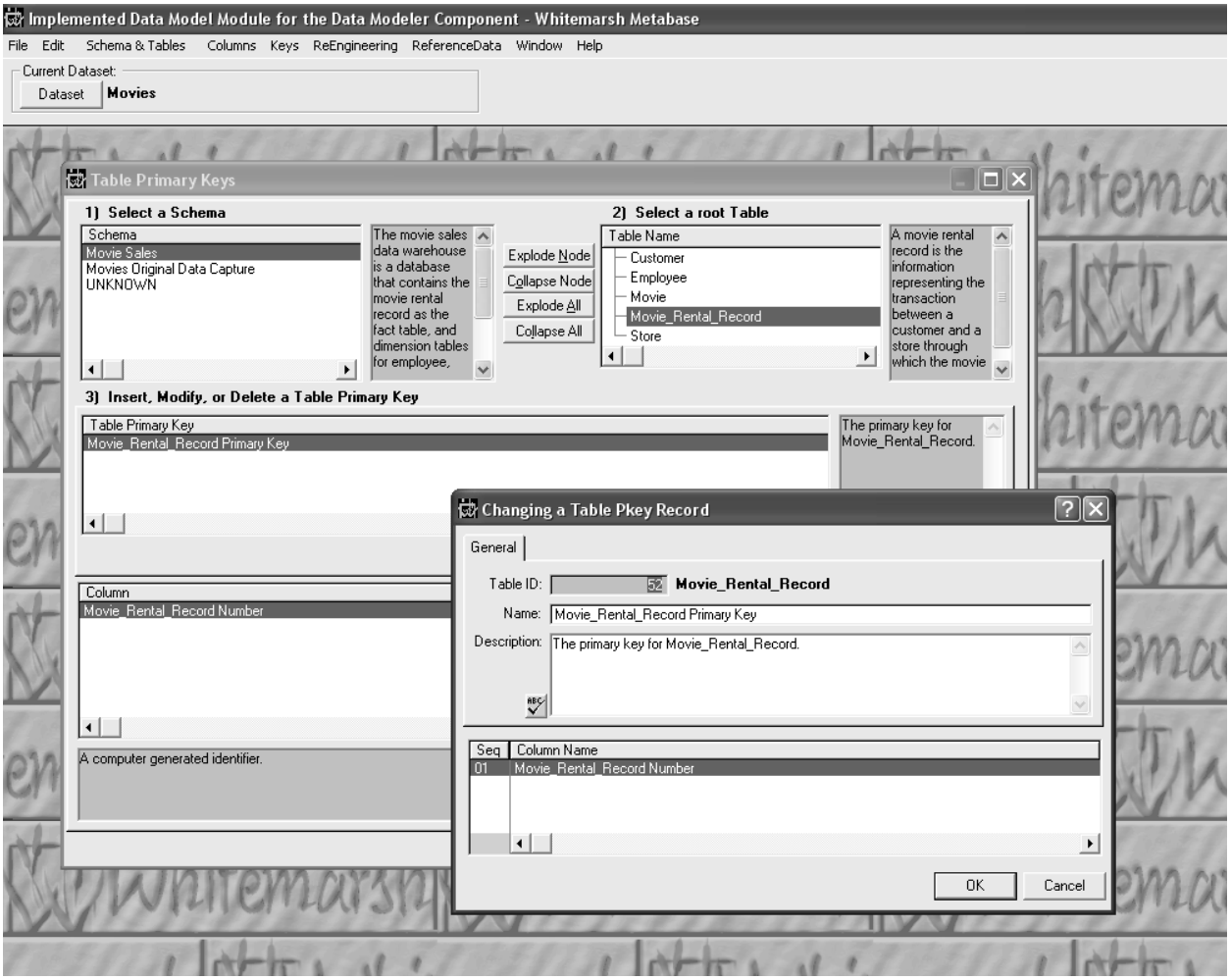

Figure 24. Primary Key update screen.

6.2.3.1.2 Allocation of Columns to the Primary Key

A primary key is a collection of columns. The order of the columns within the primary key imply the order of the values used to select rows of data given that the table is in fact a table. Figure 25 presents a list of the primary keys and the current set of columns assigned to each. To assign an column to a primary key, highlight the subject then the table, then tag the appropriate table primary key. Then, from the table's shown columns, tag the one or more columns that comprise the primary key. Finally, press the Build button. The columns that then comprise the primary key are shown in the bottom window. Once the columns are assigned, the Up|Down buttons can be used to change the sequence of the columns within the primary key.

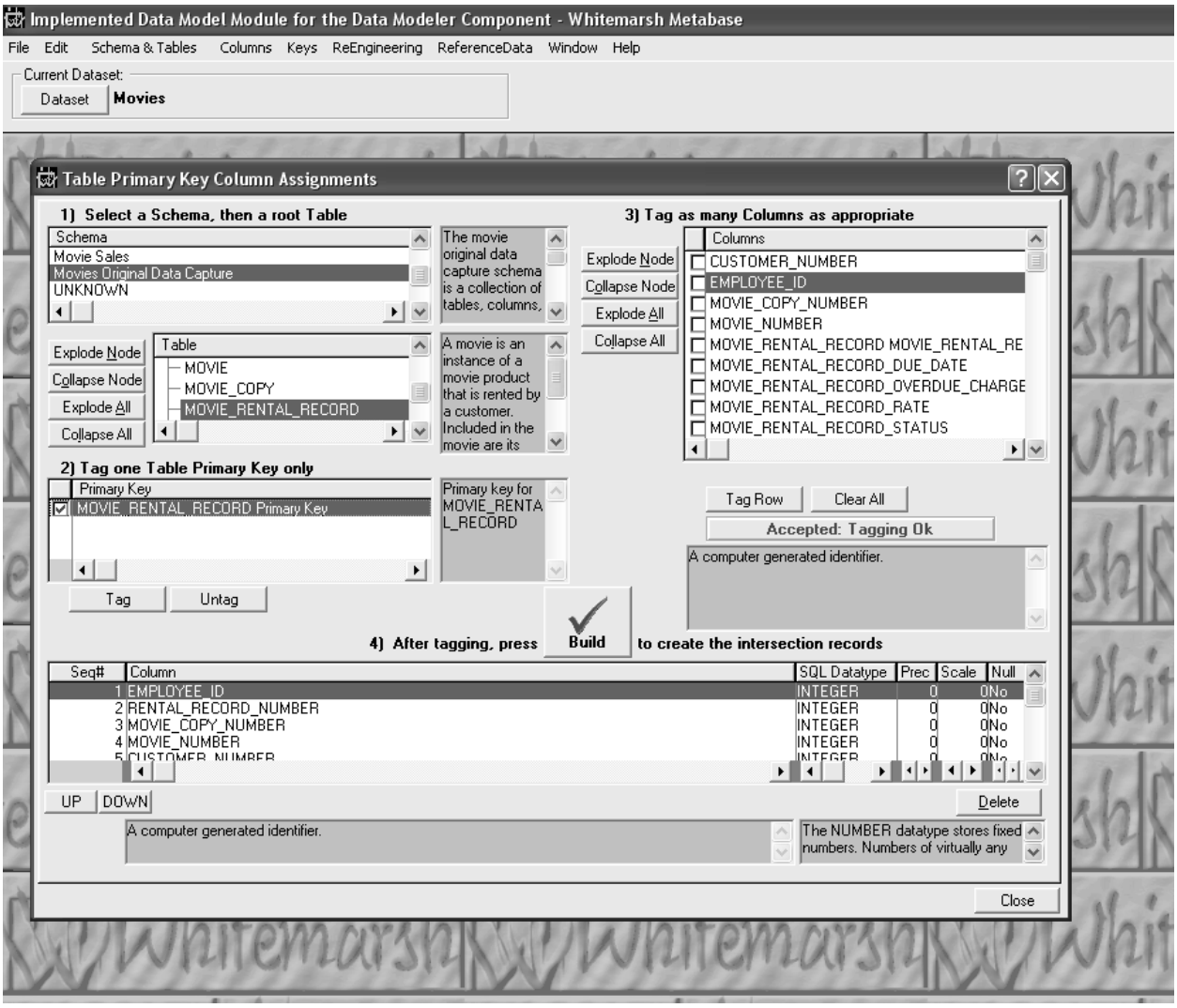

Figure 25. Adding Columns to a Primary Key.

6.2.3.2 Foreign

Foreign keys are primary keys from another table. Hence this key is a *foreign* key. There are only two critical pieces of information necessary to create a foreign key are the specific primary key of the source table, and the target table. Once these two pieces of information are provided then the rest is automatic. That is, the foreign key's name, and columns that comprise it.

Figure 26 presents the current list of foreign keys. To enter a new foreign key, highlight the subject, and then the table that is to contain the foreign key. The current list of foreign keys within that table are listed. To then create a new foreign key, press Insert. To change an existing foreign key press Change, and to delete an existing foreign key press Delete.

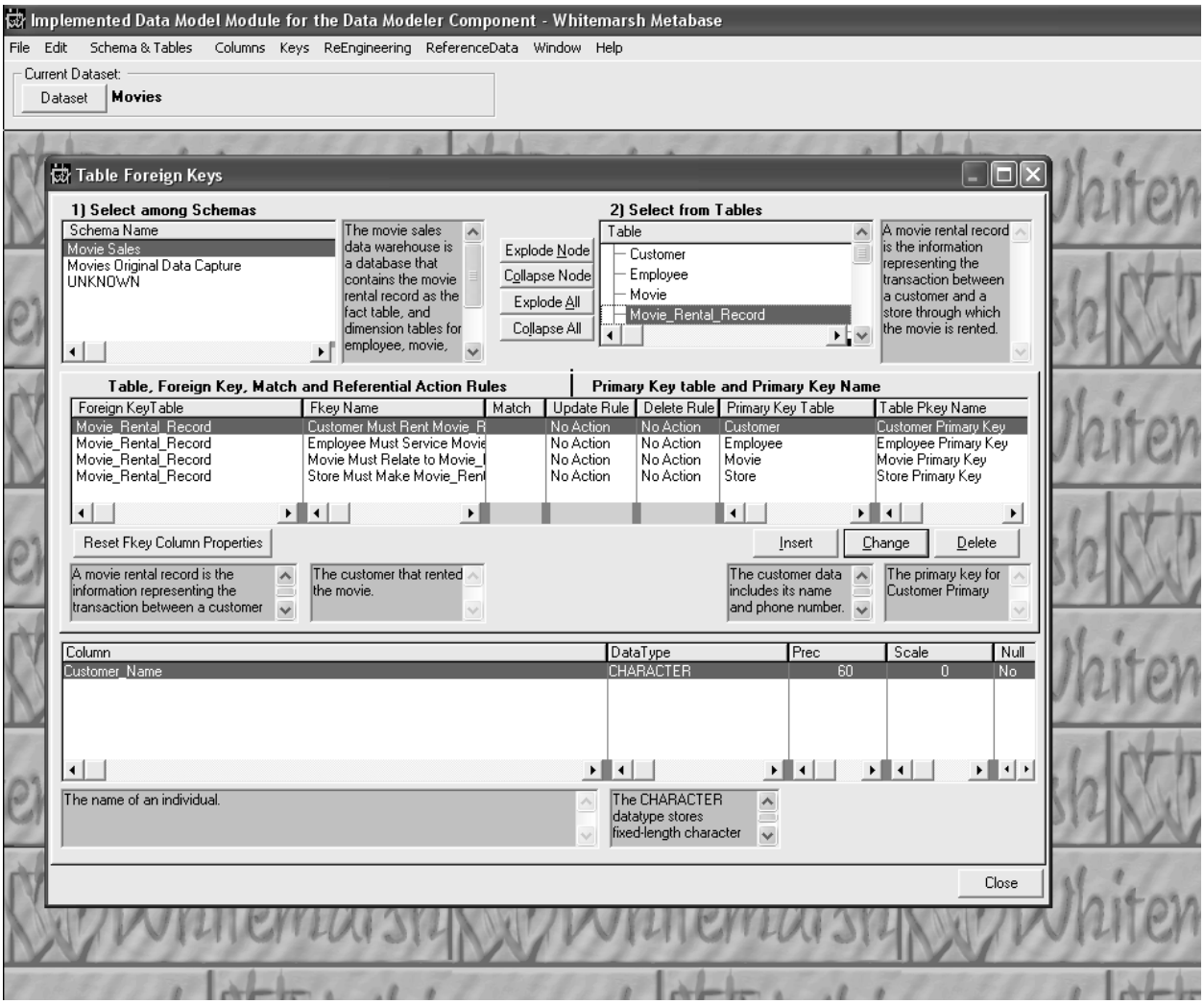

Figure 26. List of Foreign Keys.

Figure 27 presents the form for entering a new foreign key. The first step is the enter the value for Table Pkey Id. If valued with a zero and then the Tab key is pressed, a list like the one in Figure 28 is presented. Highlight the appropriate subject and then the appropriate primary key, which also shows the source table name. Then press Select.

If the table that was to contain the foreign key had been previously highlighted before the Insert button was pressed then the table's Id and name appears as the second data entry item. If a zero appears, then press Tab to cause a list of possible target entities. Figure 29 shows that list. Highlight the appropriate table and press Select.

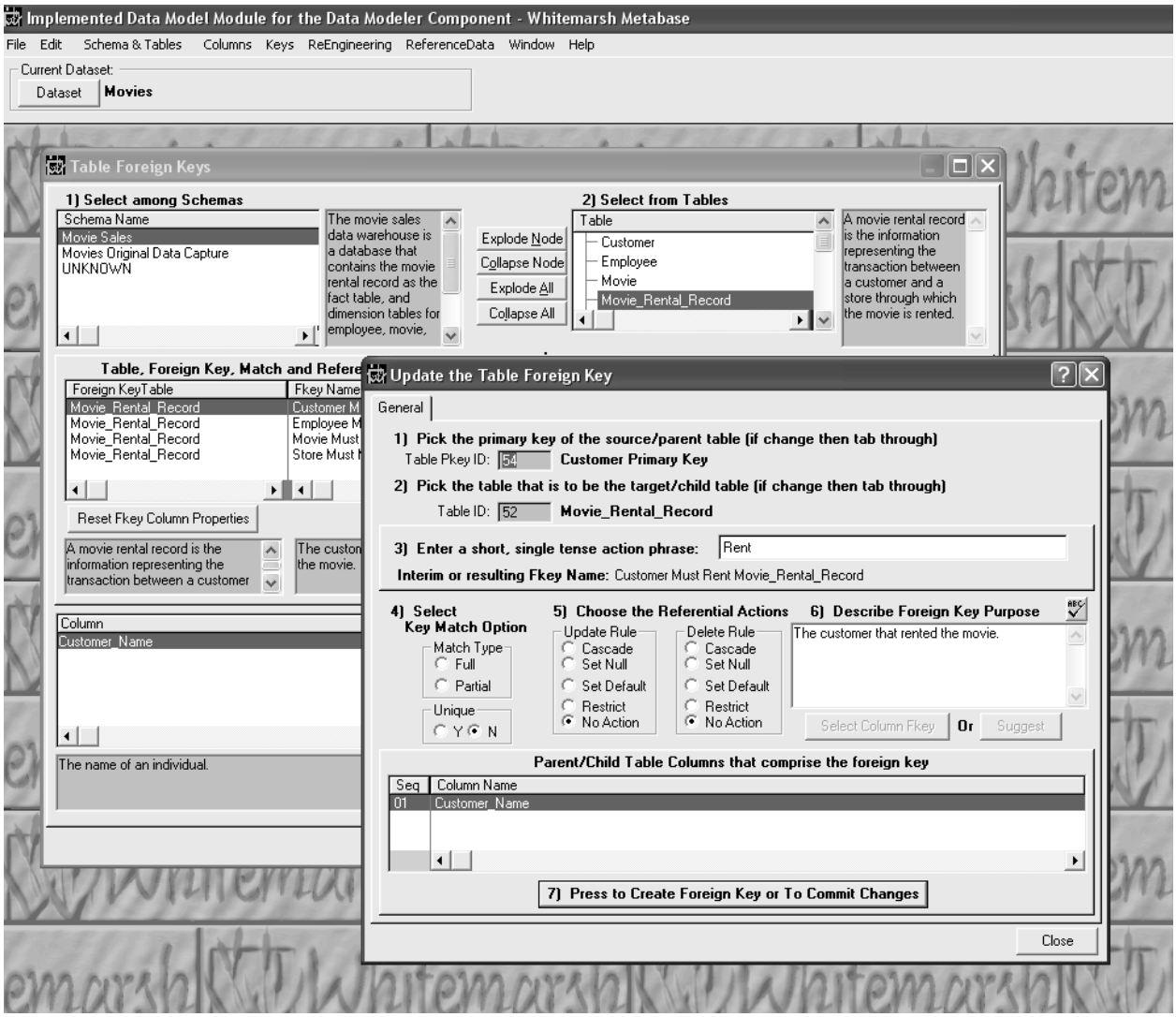

Figure 27. Foreign Key update screen.

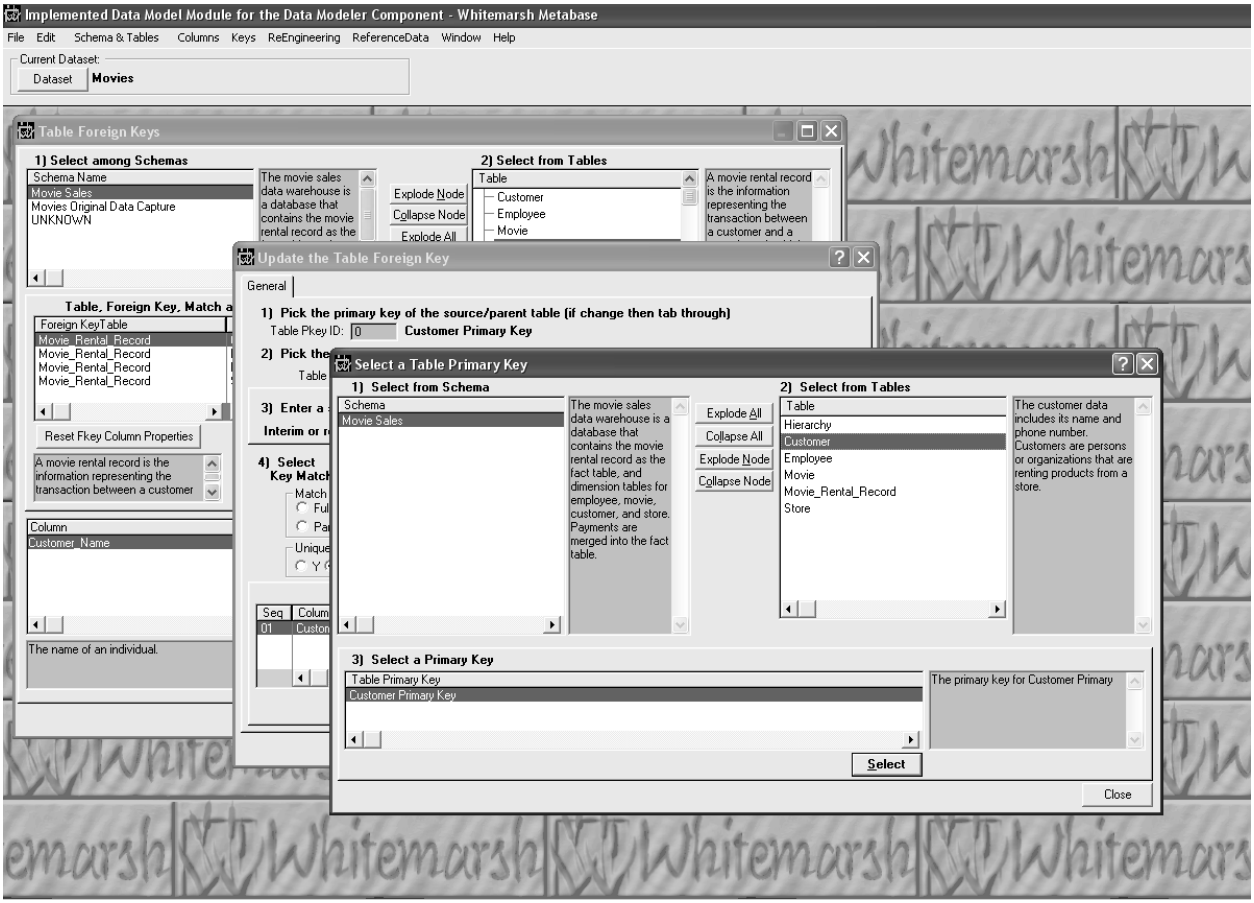

Figure 28. Selecting the source Primary Key of a Foreign Key relationship.

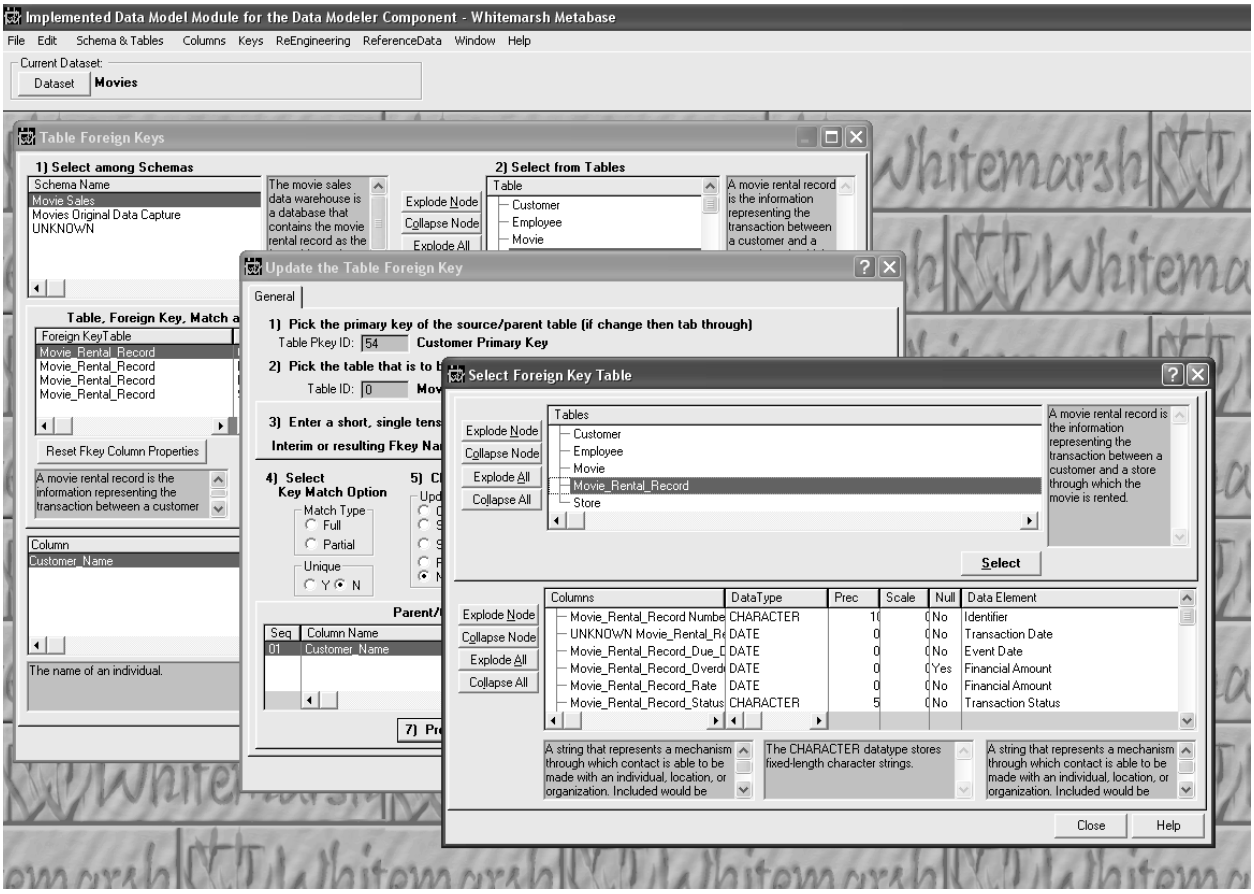

Figure 29. Selecting the target Table for a Foreign Key relationship.

Then, the third step is to enter a singular present tense action phrase. This phrase, for example, have, is employed to construct the foreign key's name. Figure 30 illustrates the entry of the singular present tense action phrase. Immediately below the phrase, Transact, is the constructed foreign key name, Customer must transact Movie Rental Record. Customer is the source table. Movie Rental Record is the target table, and "Transact" is the action phrase. The word must results from the default or selected Referential Actions, No Action. A complete set of the meanings of the Referential actions is contained in the Data Modeler Architecture book from the Whitemarsh website.

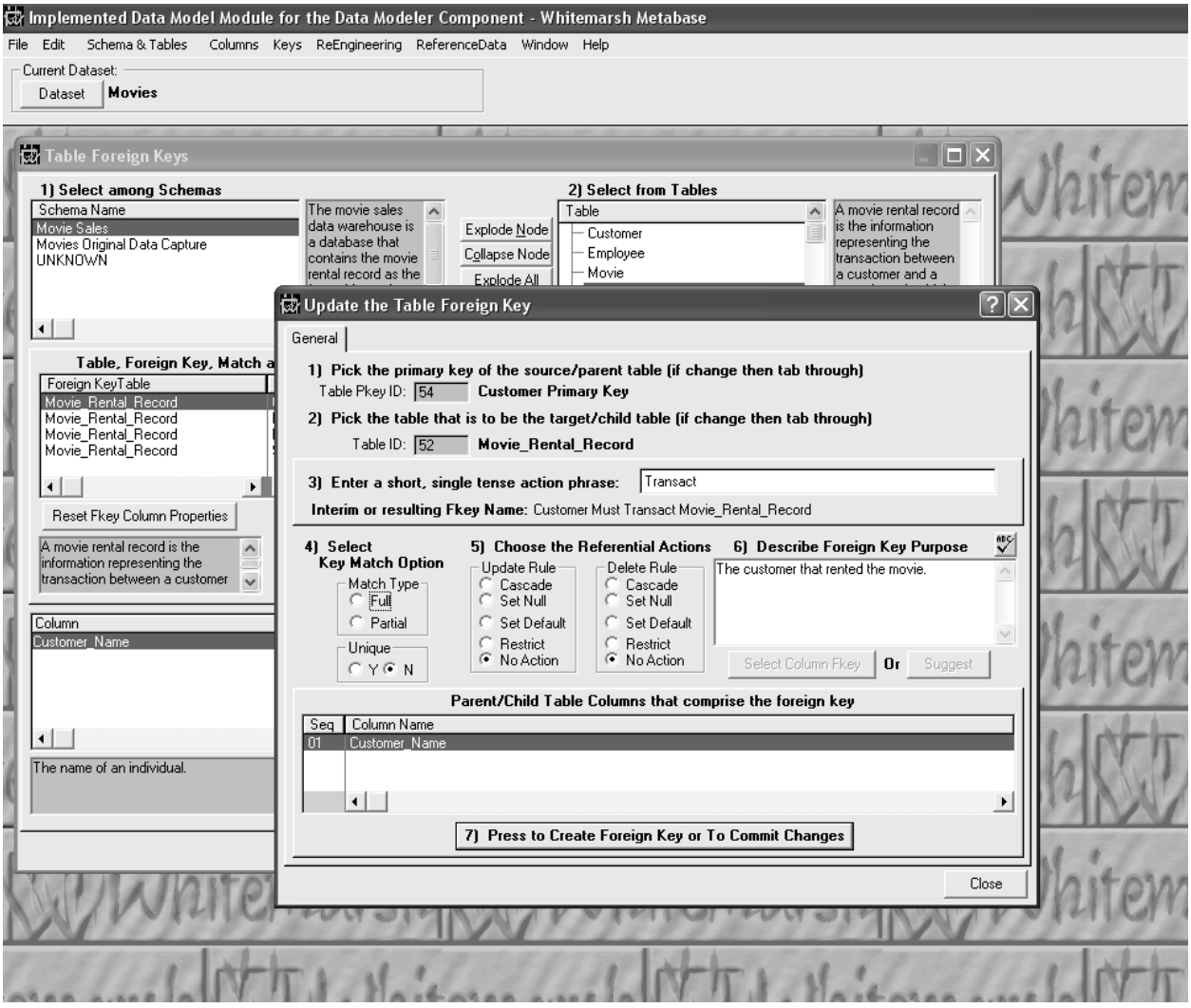

Figure 30 Entering the singular present tense action phrase for the Foreign Key relationship.

6.2.3.3 Candidate

Included in the definition of a complete candidate key is:

- Candidate key definition
- Allocation of columns to the candidate key

6.2.3.3.1 Candidate Key Definition

A candidate key of an table is a set of one or more columns that represent values that when employed result in only one selected row. Figure 31 shows the current set of candidate keys. There can be multiple candidate keys for each table. To see the columns assigned to a particular candidate key, highlight the appropriate subject, and then table.

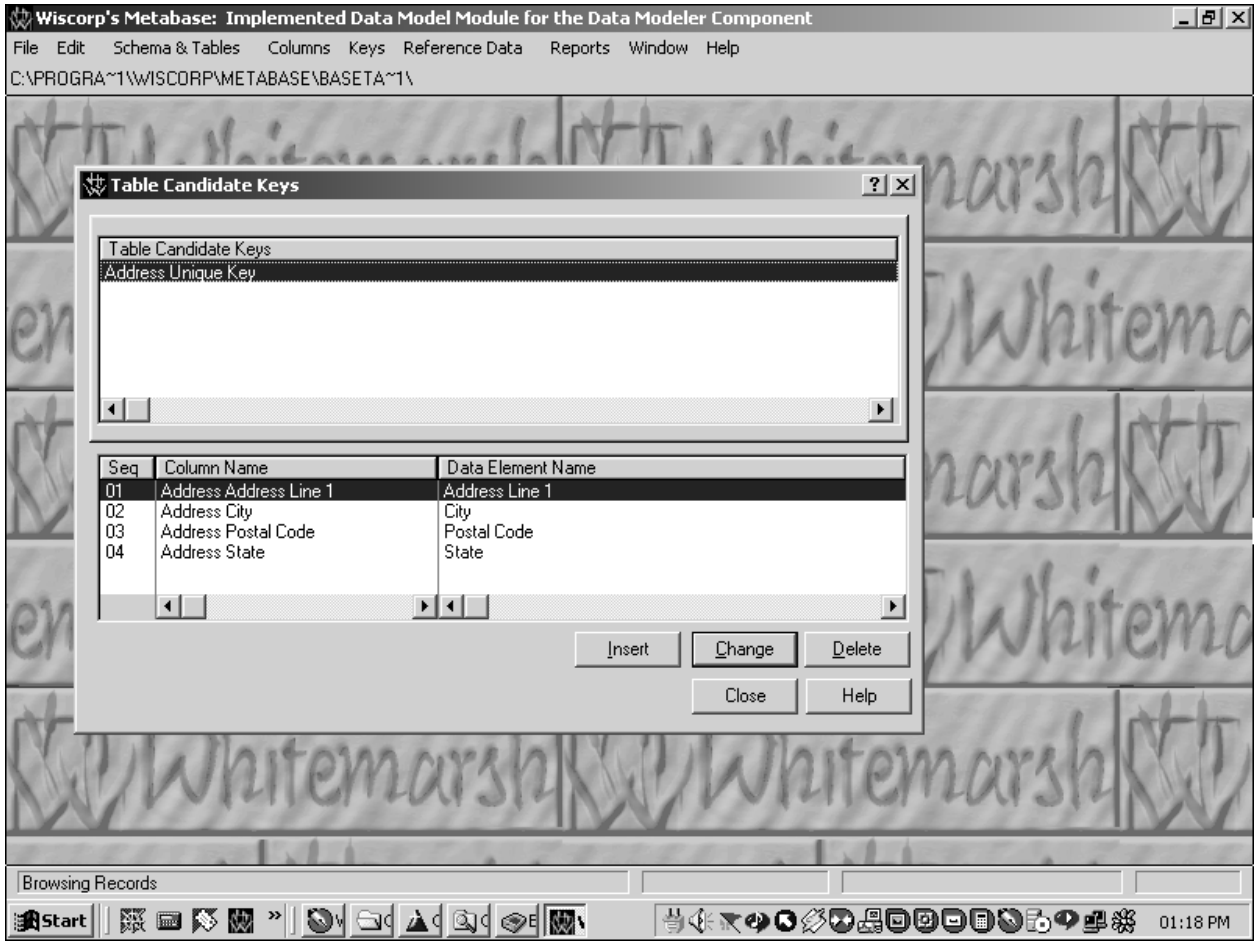

Figure 31. Listing of candidate keys.

To add a candidate for an table press Insert. If the Insert or change action succeeds, the Figure 44 is presented. On an Insert, the name is automatically constructed as the concatenation of the table name and the string, “Candidate Key.” The name can be changed. In addition to the name a description can be added or changed.

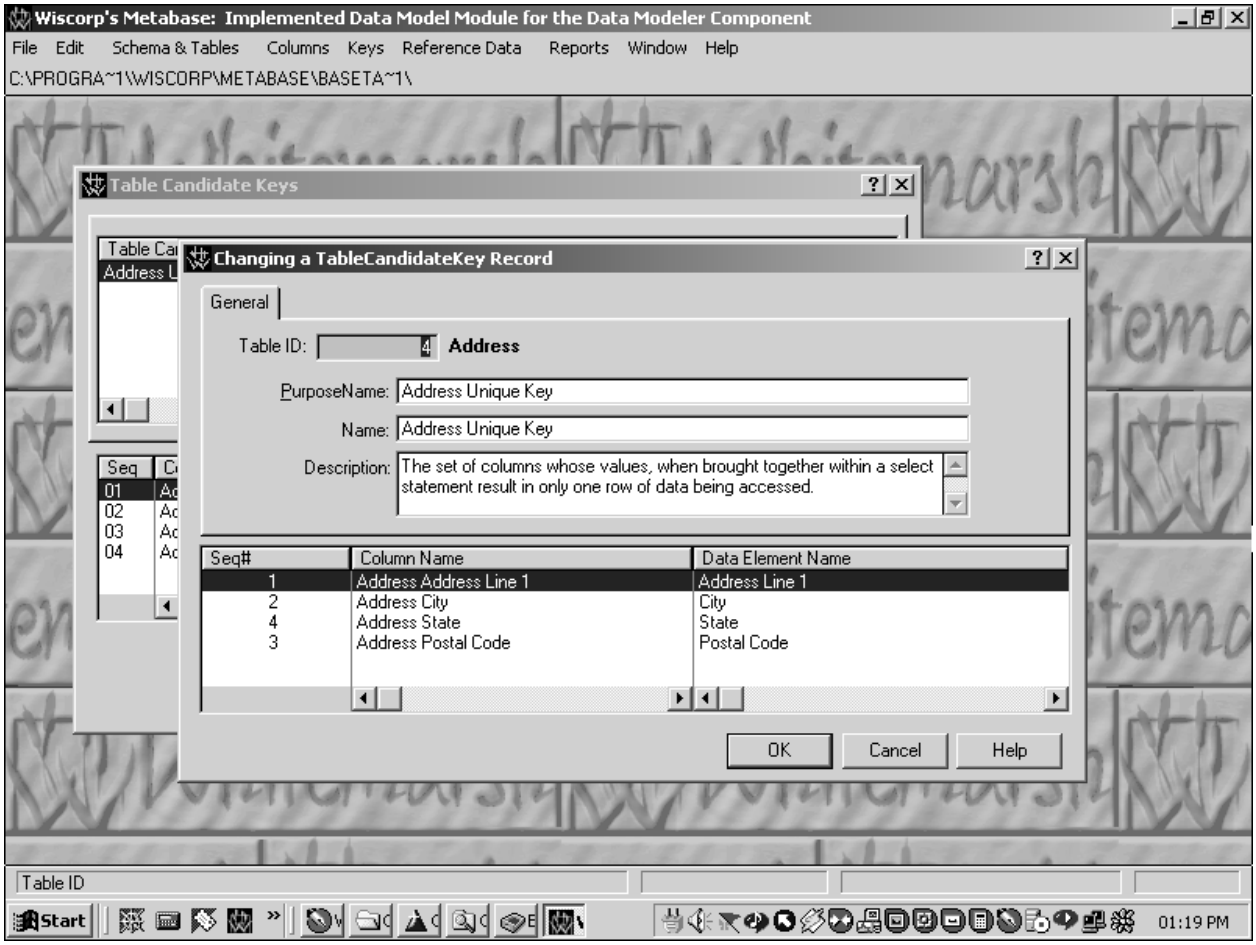

Figure 32. Updating a candidate key.

6.2.3.3.2 Allocation of Columns to the Candidate Key

A candidate key is a collection of columns. The order of the columns within the candidate key imply the order of the values used to select rows of data given that the table is in fact a table. Figure 33 presents a list of the candidate keys and the current set of columns assigned to each. To assign an column to a candidate key, highlight the subject then the table, then tag the appropriate table candidate key. Then, from the table's shown columns, tag the one or more columns that comprise the candidate key. Finally, press the Build button. The columns that then comprise the candidate key are shown in the bottom window. Once the columns are assigned, the Up|Down buttons can be used to change the sequence of the columns within the candidate key.

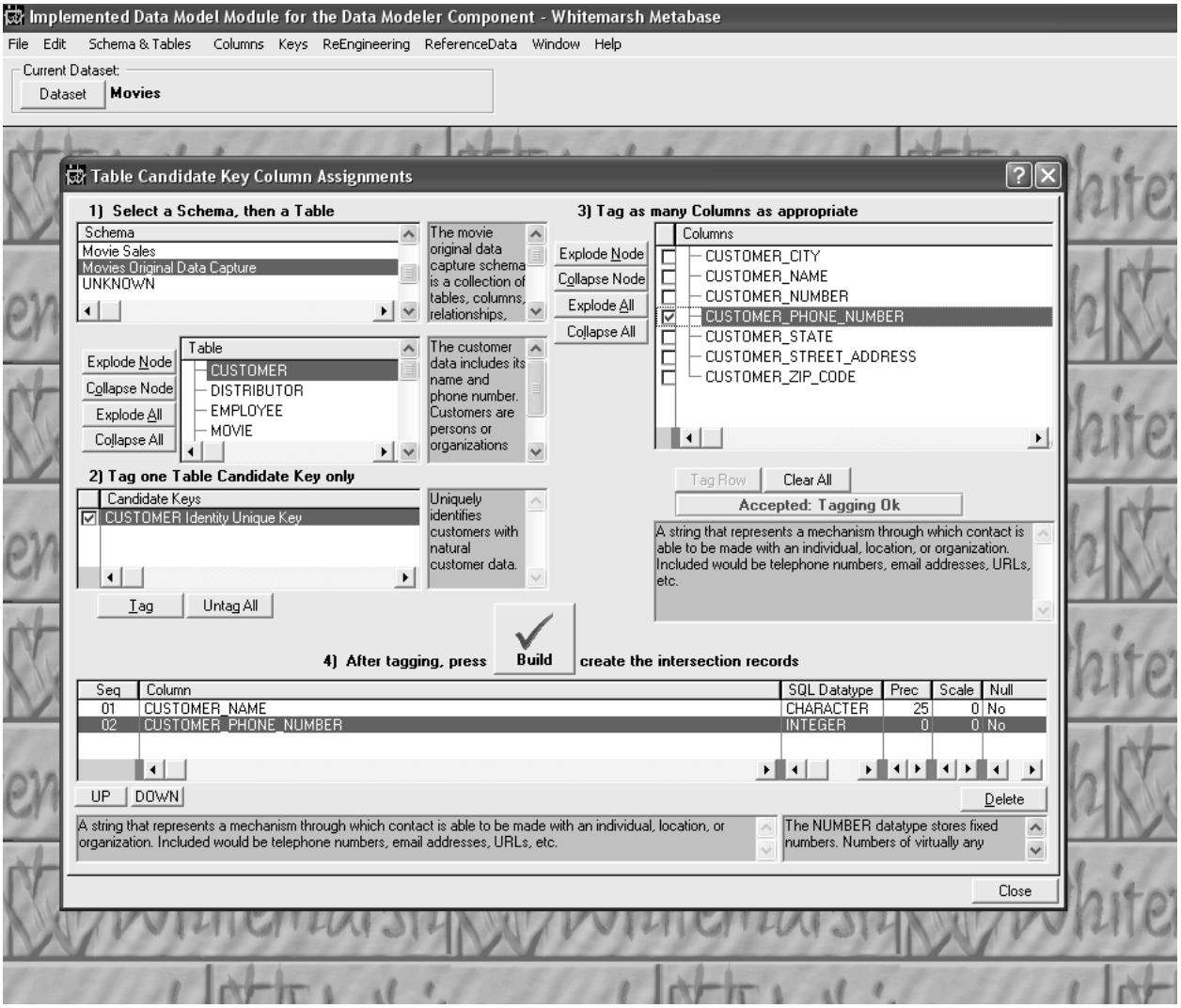

Figure 33. Adding Columns to a Candidate Key.

6.2.4 Reverse Engineering

Reverse engineering is the process of recasting an already created collection of implemented data model components. Included are:

- Reassign column to attribute
- Reassign column data elements
- Synchronize Columns, Attributes, and Data Elements
- Reassign column SQL data type
- Reassign columns to table
- Synchronize Local Definitions
- Reassigning table schema
- Reassigning tables to tables
- Promote implemented data model to specified data model
- Promote implemented data model table to specified data model
- Promote column to data element
- Remove column meta category values
- Remove column attribute assignments

6.2.4.1 Reassign Columns to Attribute

Figure 34 presents the screen through which columns are re-assigned different attribute. Tag one or more columns in the left window. Then tag one attribute in the right window and then press the Re-Assign button.

6.2.4.2 Reassign Columns to Data Elements

Figure 35 presents the screen through which columns are re-assigned to different data elements. Tag one or more columns in the left window. Then Tag one data element in the right window and then press the Re-Assign button. Once the re-assignment process is compete the reallocated columns appears in the left window.

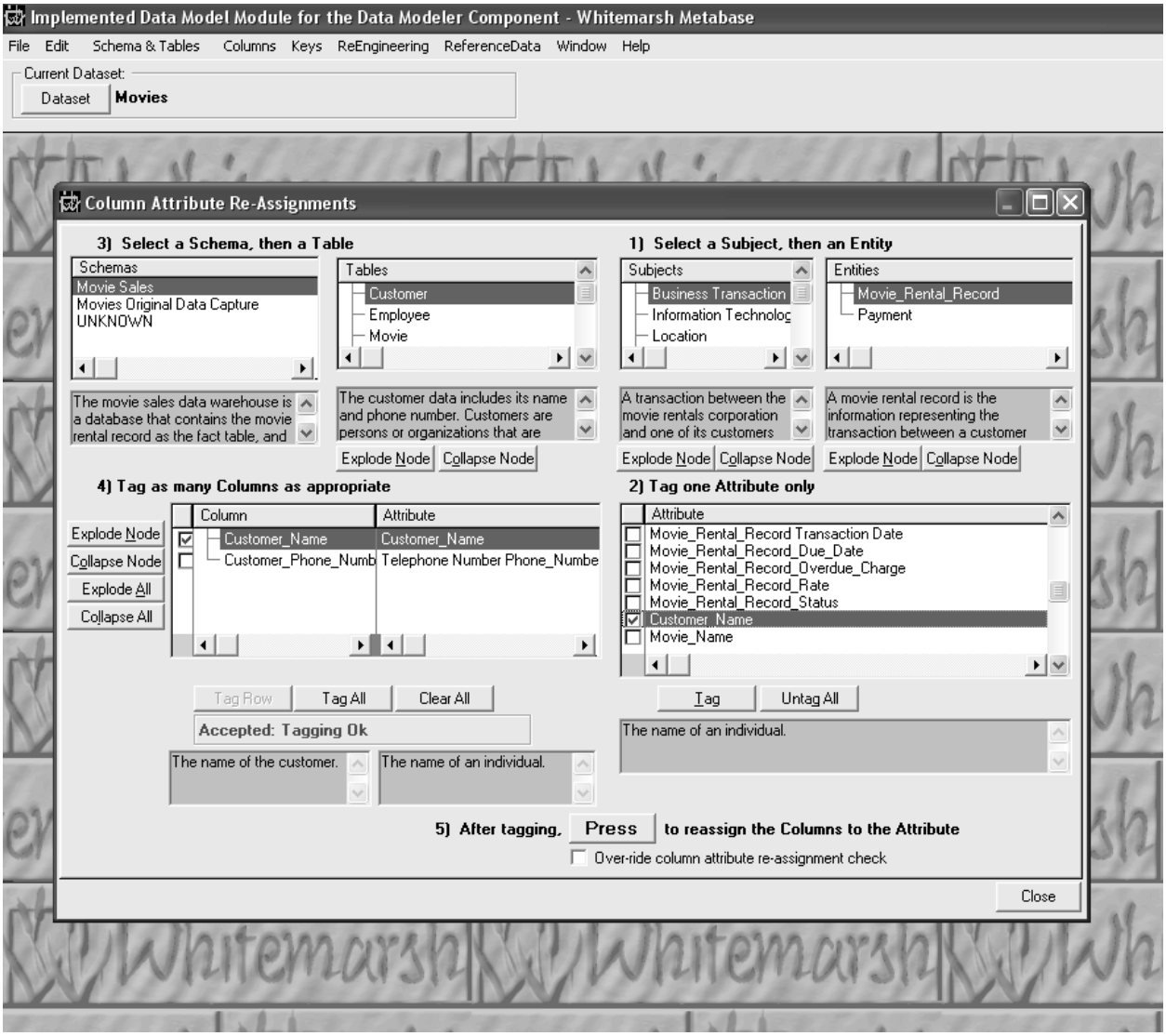

Figure 34. Reassigning Columns to Attributes.

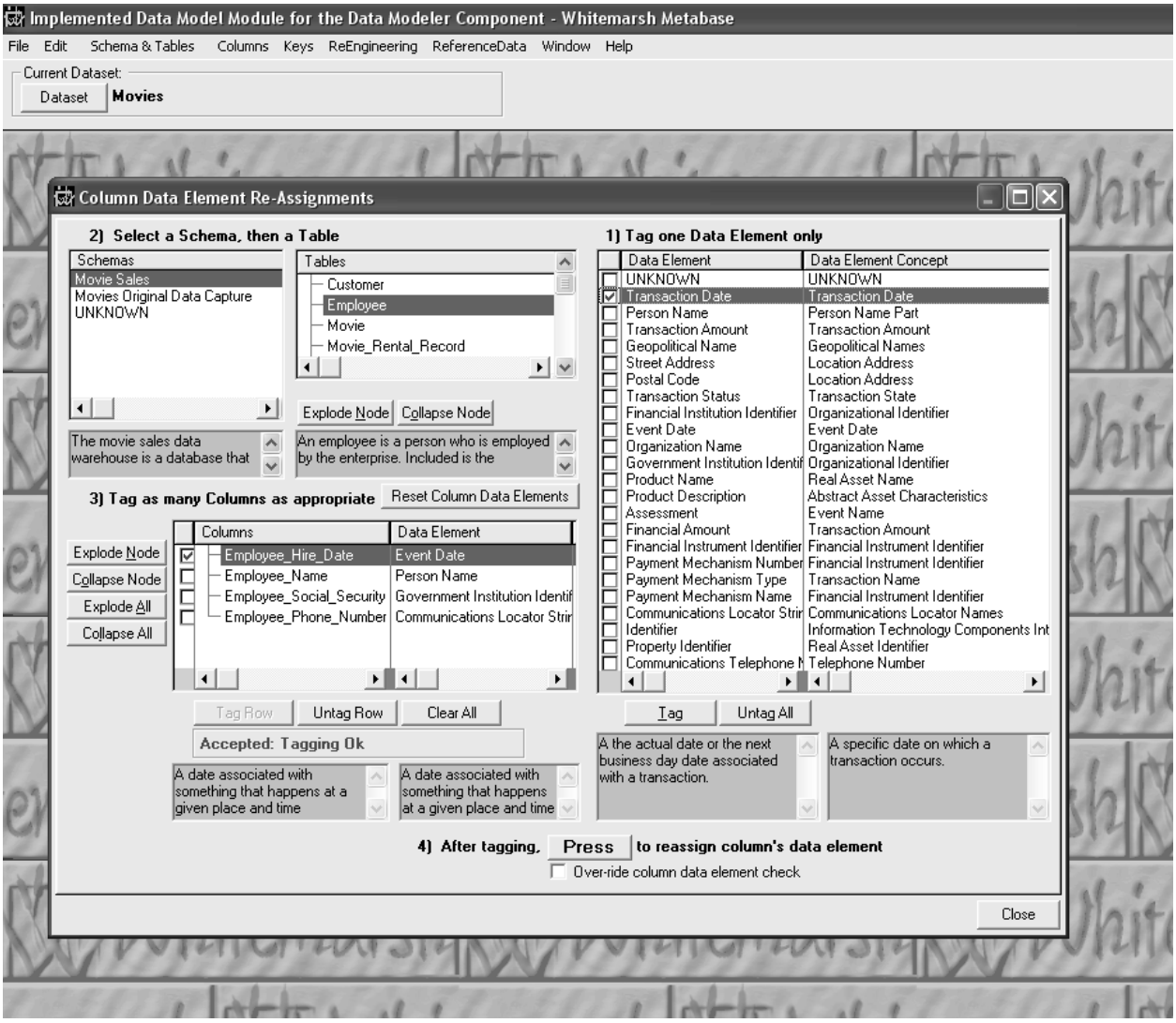

Figure 35. Reassign Columns to Data Elements.

6.2.4.3 Synchronize Columns to Attributes to Data Elements

Every columns is related to a parent data element, known, or Unknown. So too is an attribute related to a parent data element. Again, known or Unknown. Finally, a column is related to its parent attribute, and again, it can be known or Unknown. If a column is added to a table and it is related to an attribute and also to a data element, that should appear OK. However, if the selected attribute is related to a data element that is different than the column's selected data element then ambiguity occurs. What the metabase software does is reset the column's attribute to Unknown.

Figure 36 provides a fix that can be used to synchronize the relationships among columns, attributes and data elements. Shown in the left side of Figure 36 is a selected schema, table, and column. Note that the data element for the column is Unknown. That column is tagged. Now, on the right side of Figure 36, there is a selected Subject, entity, and attribute. Note that the attribute's data element is valued. That attribute is tagged. When the button Press is pressed, the metabase software changes the column's attribute to be what is tagged on the right side. Additionally, the column's data element is changed from Unknown to be that which is related to the attribute. The results in a synchronization between the column, attribute, and data element.

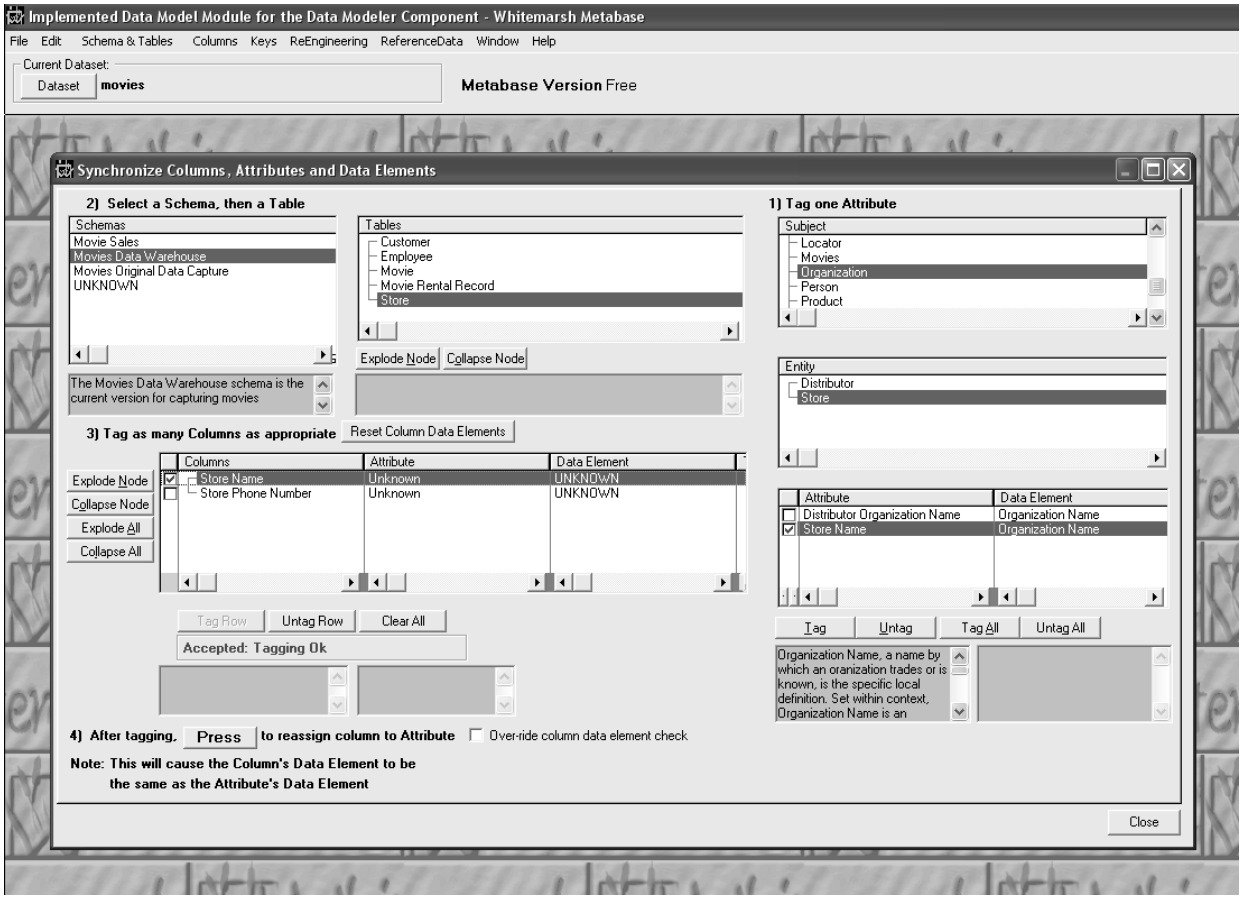

Figure 36. Synchronize Column to Attribute and Data Element.

6.2.4.4 Assign Columns to SQL Data Types

Figure 37 presents the screen through which columns are re-assigned different SQL data types. Tag one or more columns in the left window. Then Tag one SQL data type in the right window and then press the Re-Assign button. Once the re-assignment process is compete the reallocated columns appears in the left window.

When a specified data model is imported as an implemented data model or when an operational data model is promoted to be an implemented data model then in both cases the default assigned SQL data type is "unknown." Because of that, this process is required. Data types are unknown in the specified data model as that is an unnecessary detail for that level of data model. Data types are not definitively known from the operational data model because the assigned DBMS column's data type might be specific to a specific DBMS.

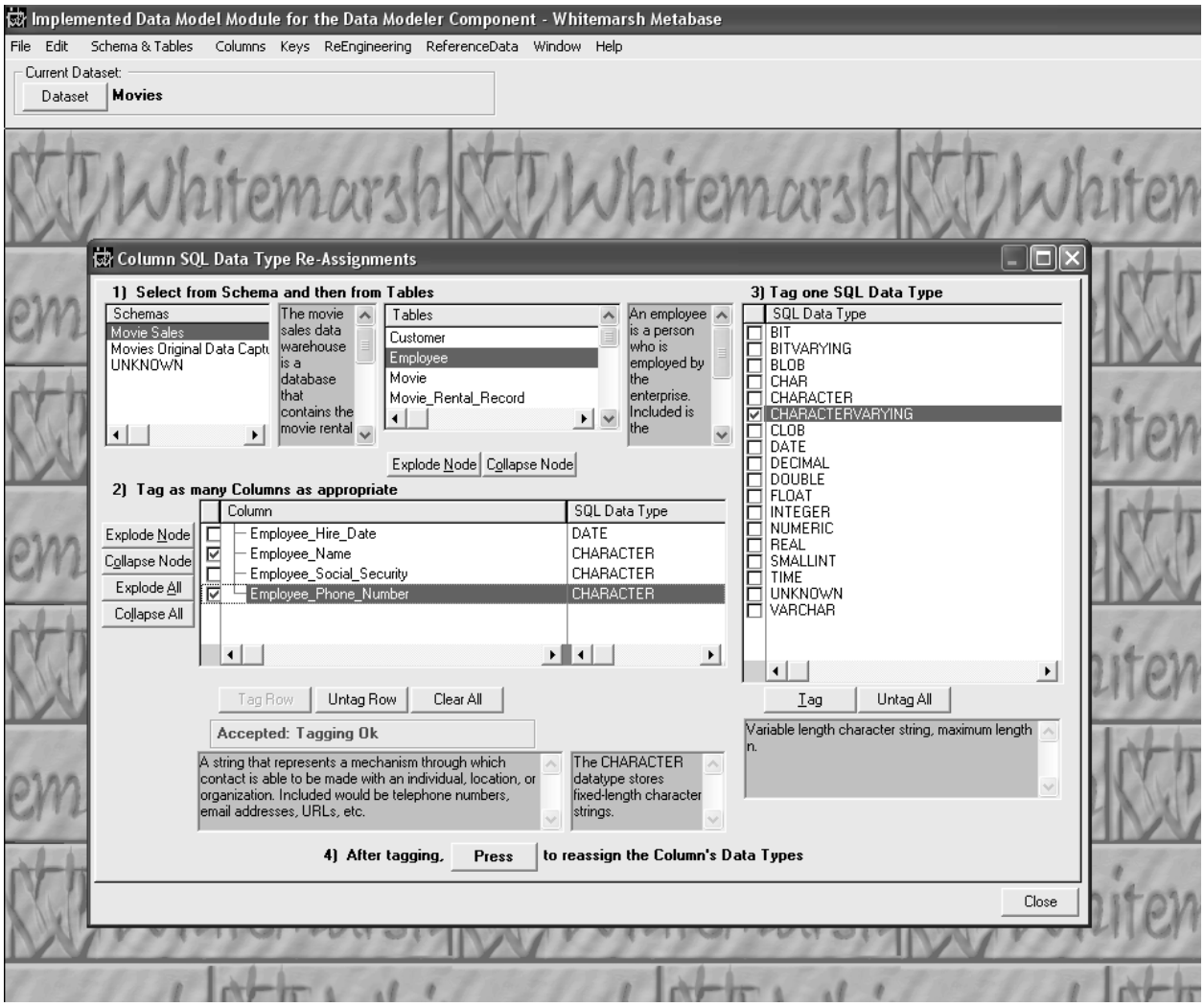

Figure 37. Reassign Columns to SQL Data Types.

6.2.4.5 Reassigning Columns to Table

Tables can be sub-typed. Essentially that means that the complete set of columns can be spread across a table family. An table family is a root table and one or more sub-tables or their sub-tables. During the process of creating columns within a table it might be discovered that the column really belongs in a different table. This type of reassignment supports moving a column from one table within a table family to another table.

Figure 38 illustrates the process of reassigning one or more columns from one table to another table within the same family. Highlight the schema and then the table. Then tag one or more columns that are to be moved. Then tag the table to which the columns are to be moved. Press the reassigning button. The columns are then moved.

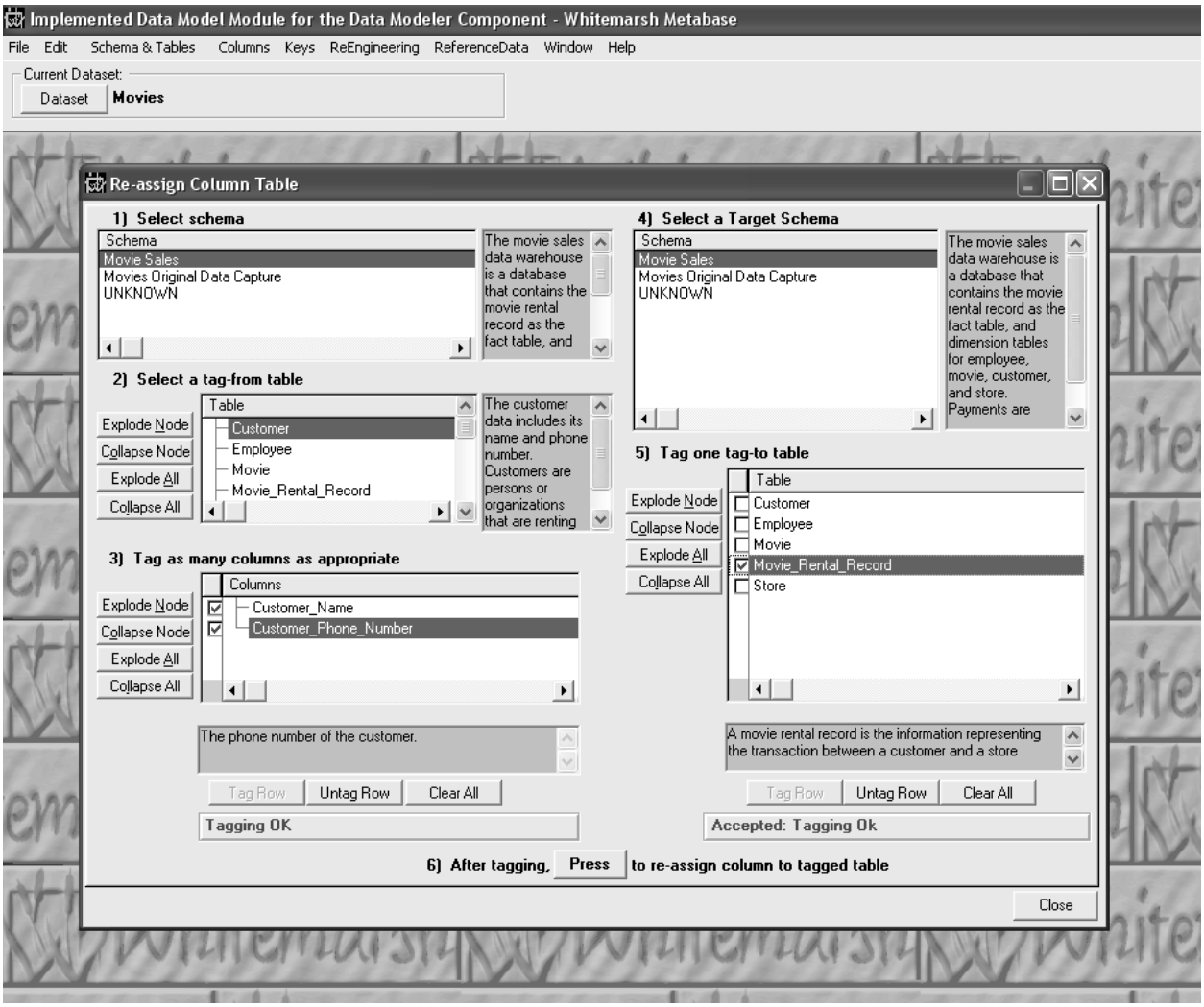

Figure 38. Reassigning Columns to Tables.

6.2.4.6 Synchronize Columns to Local Definitions

Columns have local definitions. Some columns within different tables and schemas are to have the same local definition. For example, in several of the tables associated with three different Movies schemas there is the movie's name. For sure, in all cases the local definition would be, "the name of the movie." If this local definition is different across the columns of the different tables and schemas, users will wonder why the difference exists. What's the hidden meaning because of the difference? Figure 39 provides the ability to synchronize local definitions across columns. On the left side, select a schema, table, and then tag the column that is to be source of the local definitions. On the right side of Figure 39, select as many different columns as may be appropriate. Do this by selecting the schemas, tables, and columns. Different columns from the different tables, and schemas can be selected and tagged. Once selected and tagged, press the Synchronize definition button. Note: the only definition that is synchronized is the local definition. The contextual definition will have to be re-generated.

6.2.4.7 Reassigning Tables to Schema

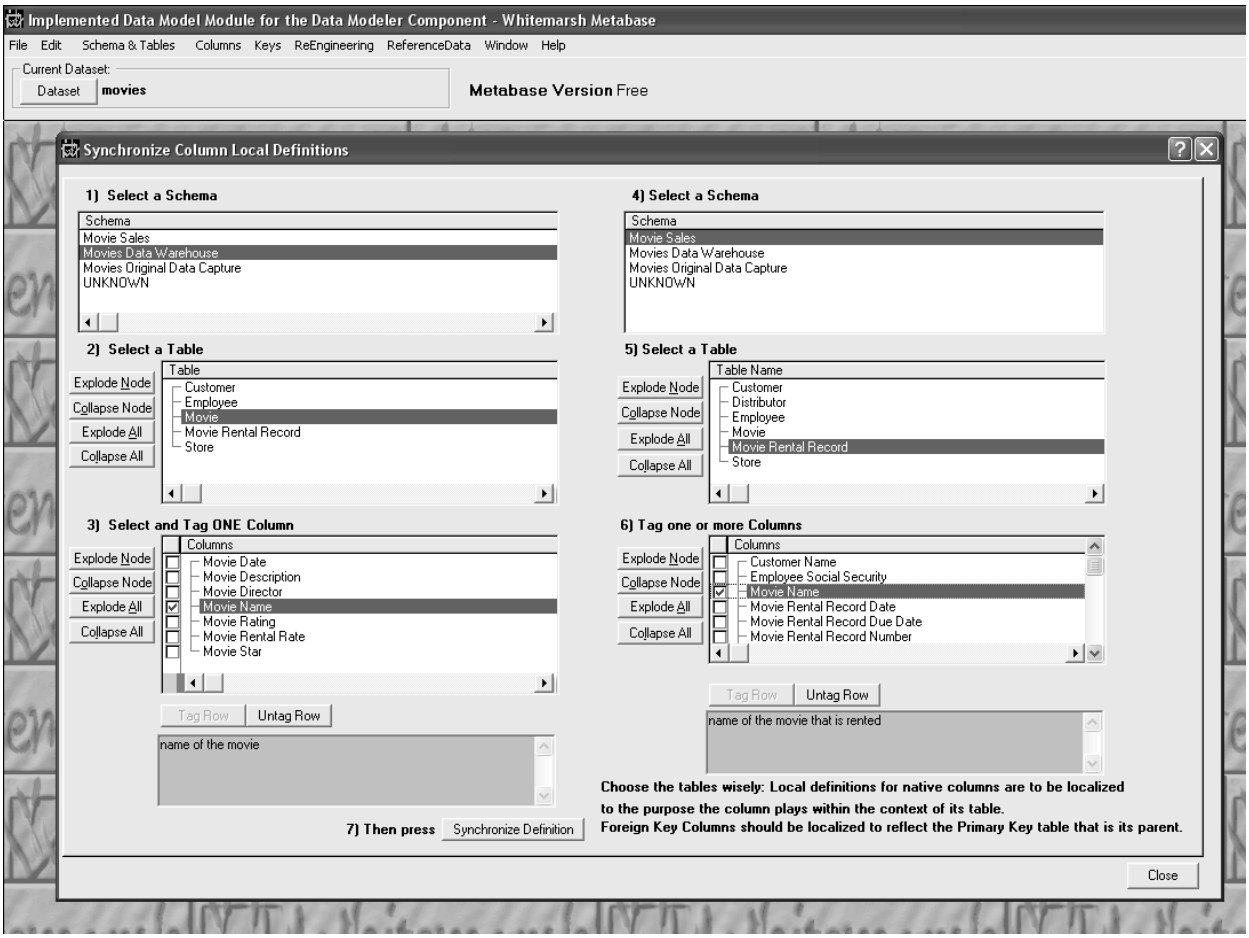

Figure 39. Synchronize local column definitions.

Figure 40 displays the process for re-assigning tables to different schemas. The process starts with highlighting the table that is to be re-assigned. Tag one or more tables that are to be re-assigned. Then highlight and tag the new schema. Once the tables and a schema is tagged, press the Re-Assign button. The underlying process then re-assigns the table from the current schema to the new schema. Once all the tagged tables are reassigned the windows are redisplayed with the newly assigned schemas. When a table, for example, customer is reassigned, so too are all the descendants of customer, for example, Order Header, Order Detail, and Address.

If a root table is tagged then all its sub-typed tables are moved as well. Also automatically moved are all tables that are related to the root table by primary-to-foreign key relationships. To move just single tables, start with "leaves" and work backwards towards the table's root.

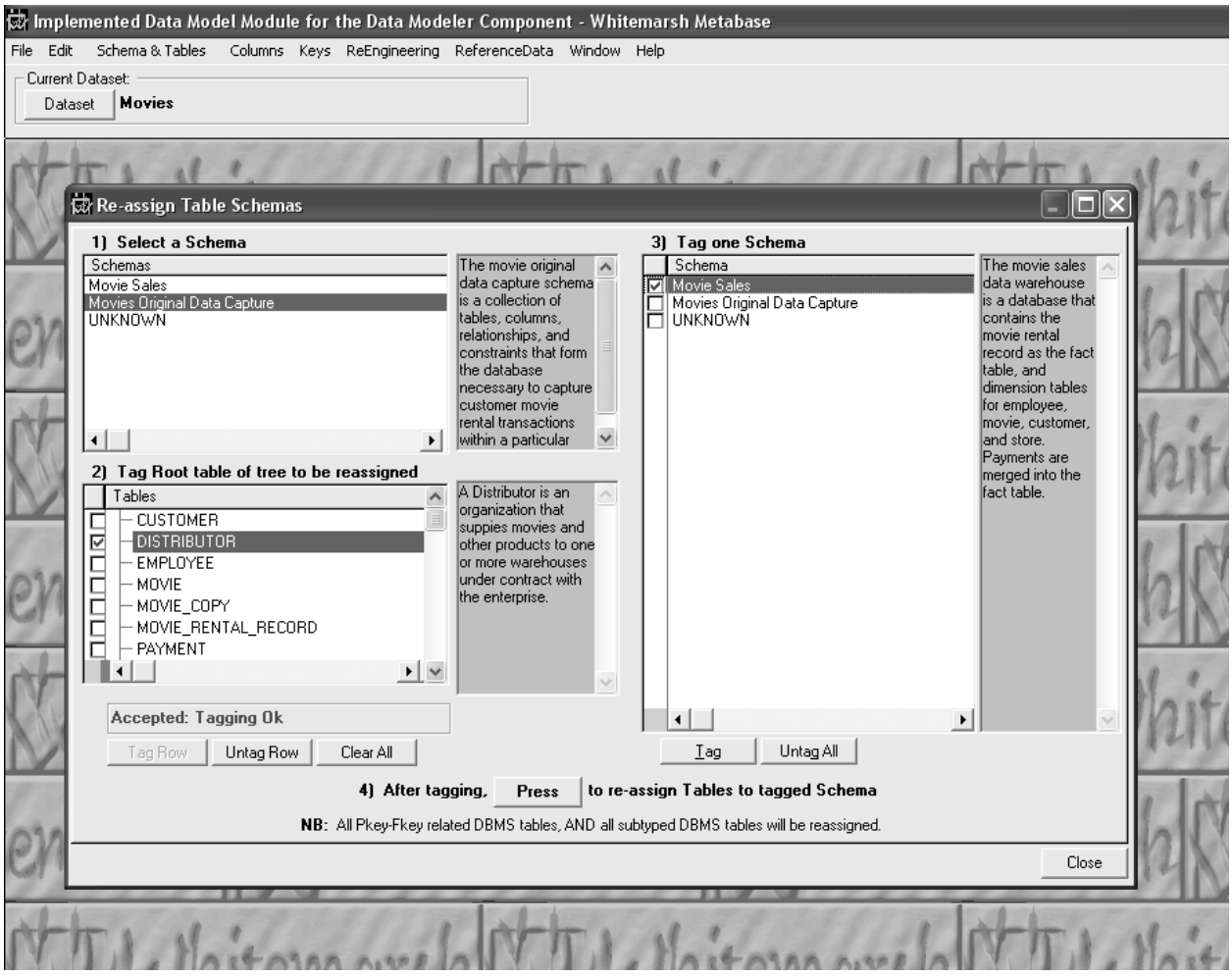

Figure 40. Reassign Tables to Schema.

6.2.4.8 Reassigning Tables to Tables

As stated above, tables can have sub-typed tables. This is shown in Figure 41. A table might be pushed too far down in the table family. In this case the reassignment allows the "parent" of an sub-typed table to be changed. To accomplish this, tag one or more tables in the left window and then the new parent in the right window. Then press the Re-assign button. If a re-assign message is appropriate on either the left or right window it will be displayed. These messages are designed to prevent inappropriate re-assignments. For example, making a different root table.

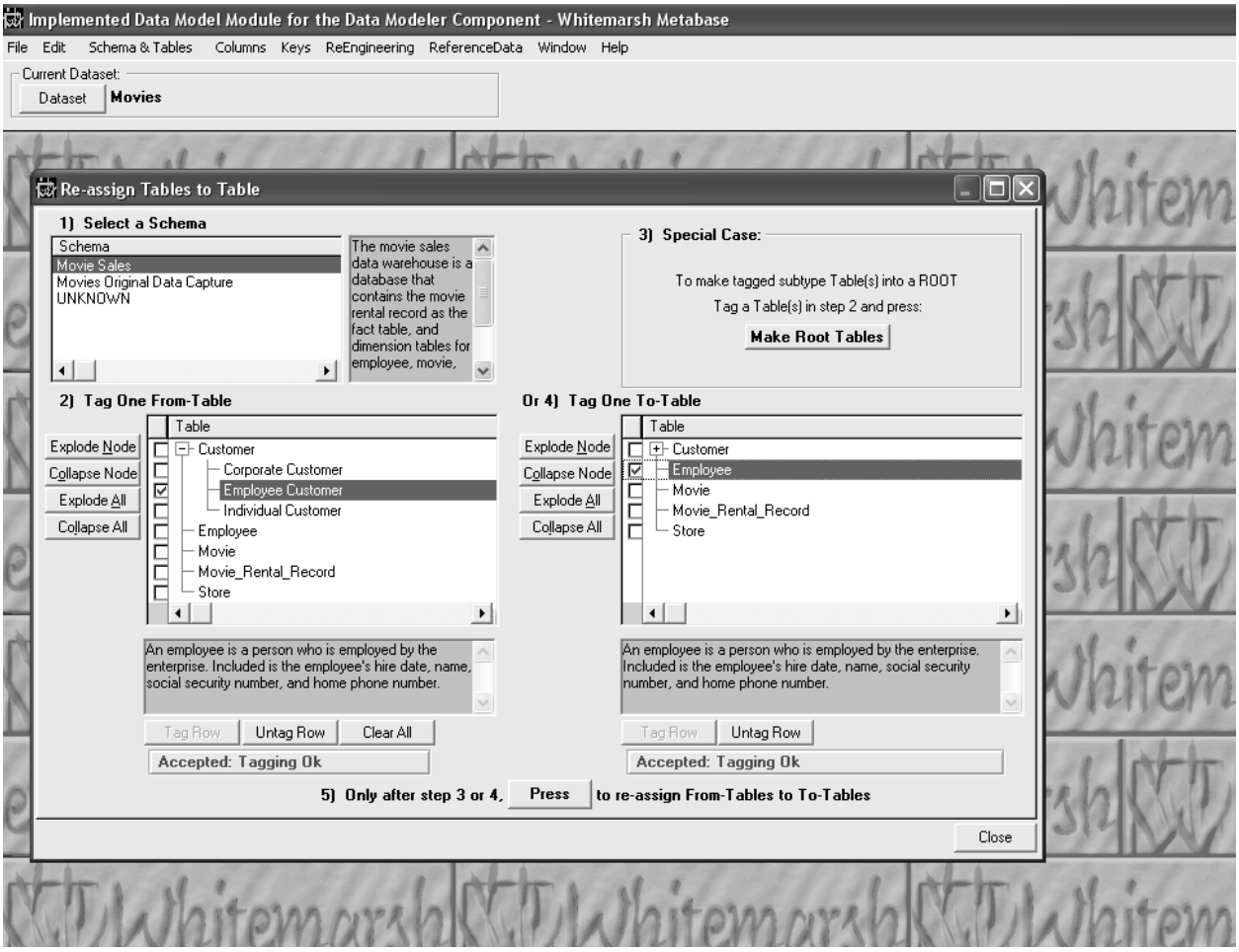

Figure 41. Reassigning subtables to tables.

6.2.4.9 Promote Implemented Data Model to Specified Data Model

Figure 42 presents the screen for promoting an implemented data model to be a specified data model. When an implemented data model is promoted, its collected set of tables, columns, data element references, meta category value assignments, column value domains, and primary, foreign and candidate keys are all created anew within the specified data model. Once the new specified data model components are created, the implemented data model meta category value assignments and the column value domains are deleted. That is because they are automatically inherited by the implemented data model and would therefore be redundant.

The promotion process is simple. Highlight the implemented data model schema and then the subject of the specified data model. Then press the Promotion button. The process accomplishes what is described above and then the newly promoted entities and attributes appear underneath the highlighted subject.

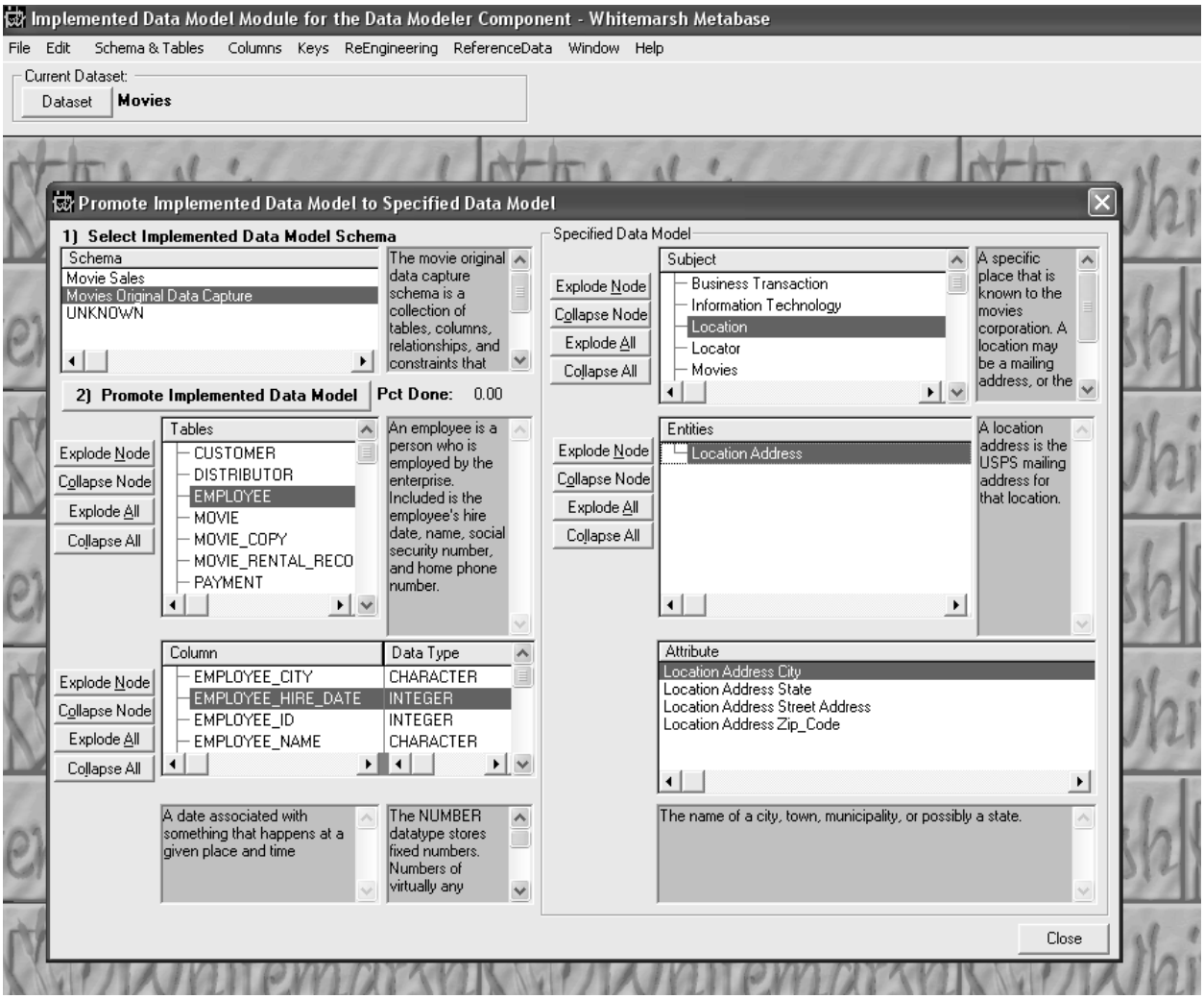

Figure 42. Promoting an Implemented Data Model to Specified Data Model.

6.2.4.10 Promote Implemented Data Model Table to Specified Data Model

Figure 43 presents the screen for promoting a table within an implemented data model to be an entity within a specified data model. When an implemented data model table is promoted, its columns, data element references, meta category value assignments, column value domains, and primary, foreign and candidate keys are all created anew within the specified data model. Once the new specified data model components are created, the implemented data model meta category value assignments and the column value domains are deleted. That is because they are automatically inherited by the implemented data model and would therefore be redundant.

The promotion process is simple. Highlight the implemented data model schema then the implemented data model table. Then select the subject of the specified data model. Then press the Promotion button. The process accomplishes what is described above and then the newly promoted entity and attributes appear underneath the highlighted subject.

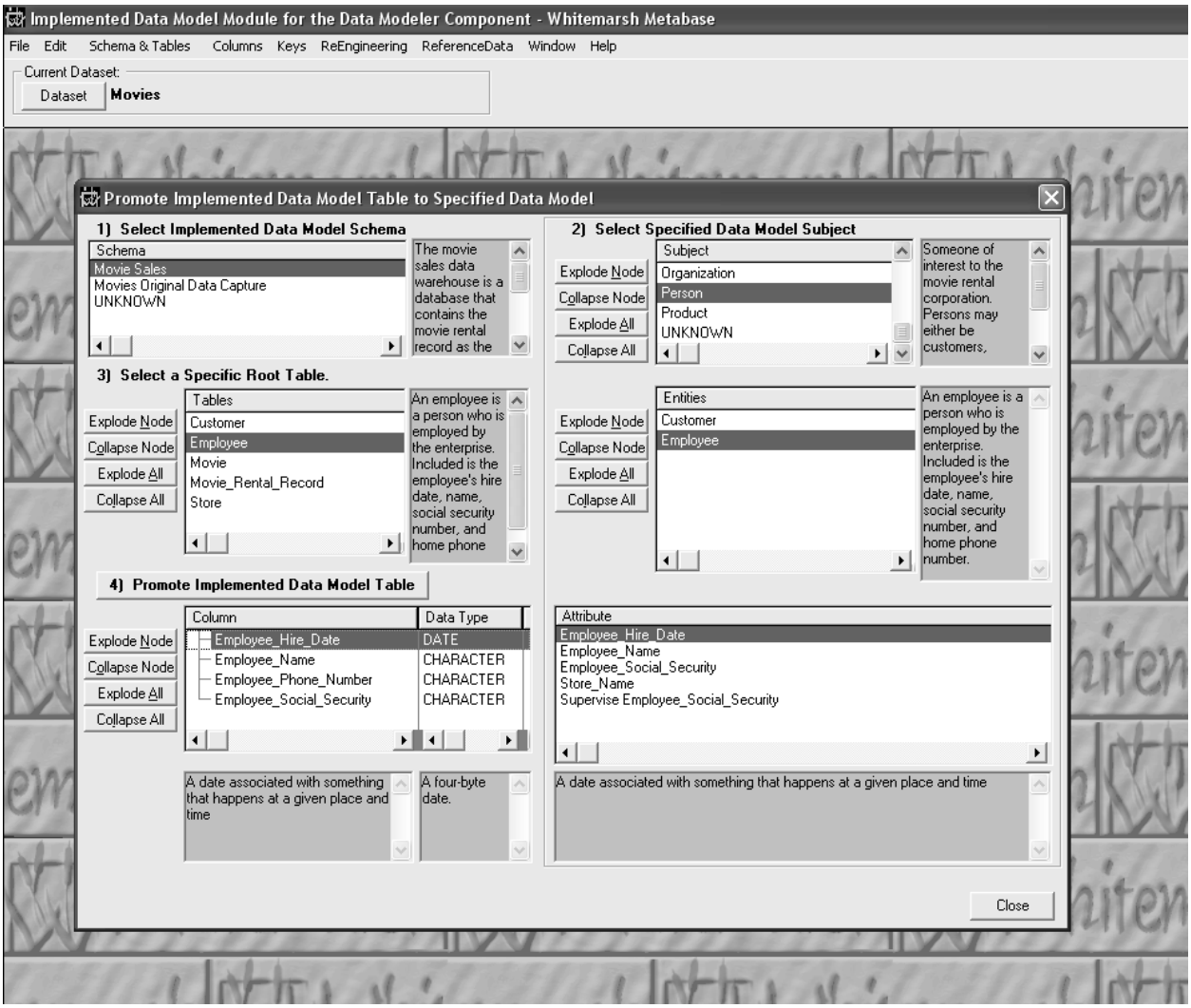

Figure 43. Promotion of a Implemented Data Model Table to a Specified Data Model Subject.

6.2.4.11 Promoting Column to Data Element

Figure 44 displays the window for promoting an column to a data element. The column is identified by highlighting the subject then the table, and finally the column. Then highlight the data element concept. Once the column and the data element concept is highlighted, press the Promote button. The underlying process a new data element is created from column and assigns to that newly created data element all the semantics assigned to the promoted column. The semantics assigned to the column are then deleted as they are then inherited from the newly created data element. The existing column is not automatically removed as it may have been the "parent" of one or more columns. If a delete operation is attempted, and the column is not a parent of a column then the delete process will succeed.

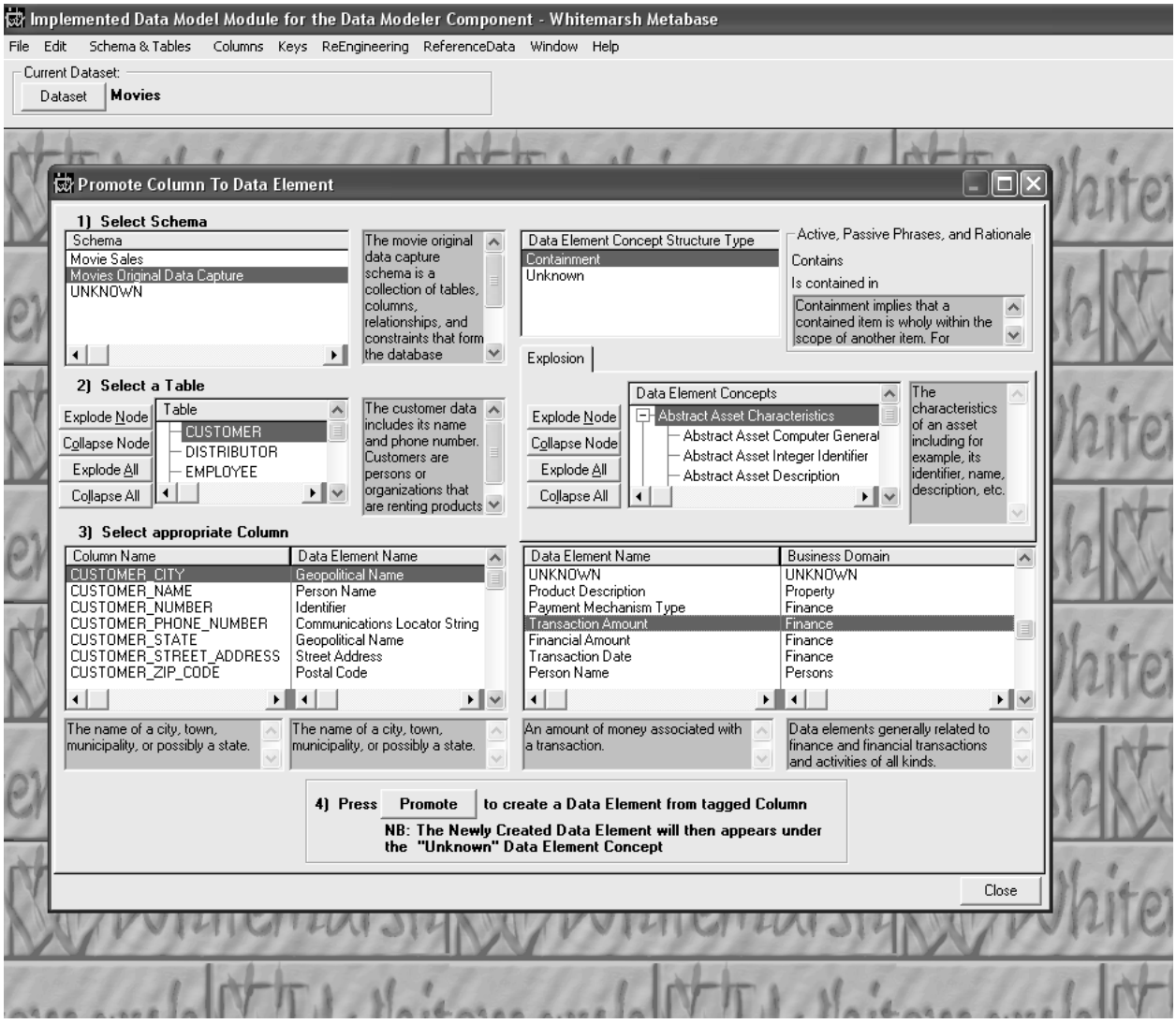

Figure 44. Promote Column to Data Element.

6.2.4.12 Remove Column Meta Category Values

Figure 45 displays the window for removing meta category values from an attribute. Select the subject, entity, and attribute. Select the specific meta category value that is to be deleted. Then press the Delete button.

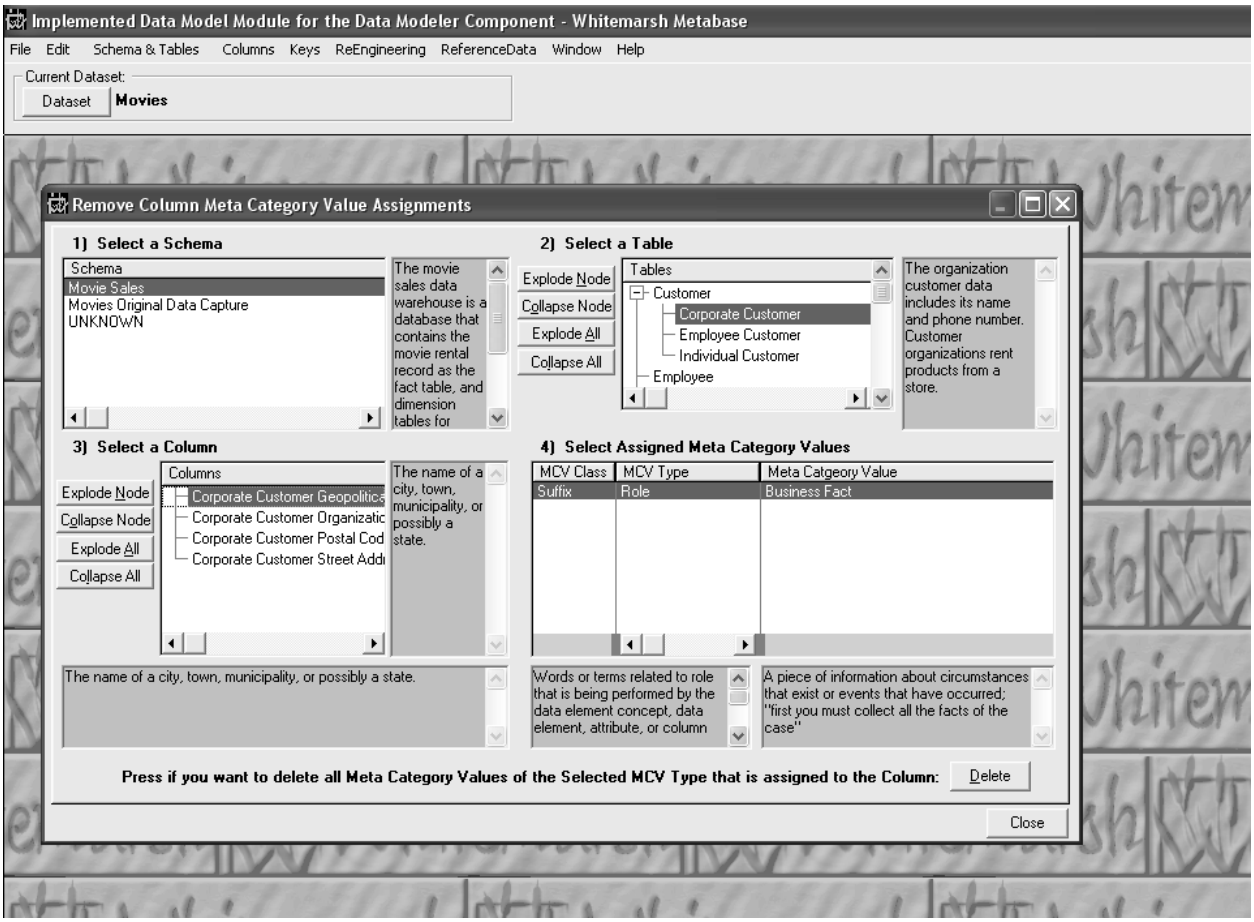

Figure 45. Remove Column Meta Category Value assignments.

6.2.4.13 Remove Column Attribute Assignments

Figure 46 displays the window for removing the attribute assignments for a column. Select the schema and table. Then press the "Press" button to remove the assignments. When complete, all the column assignments will be shown as "unknown.

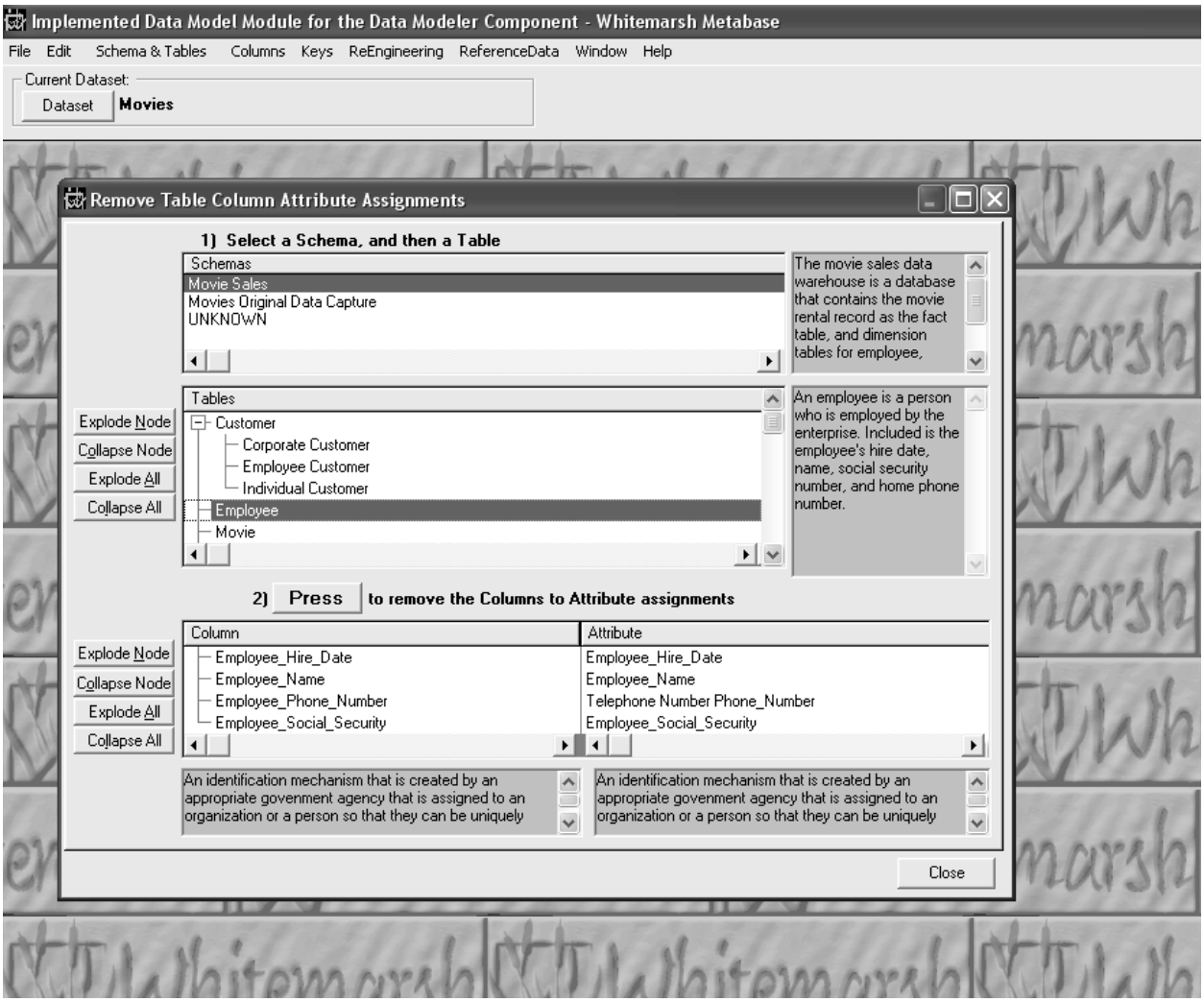

Figure 46. Removing Column Attribute assignments.

6.2.5 SQL DDL

Once an implemented data model has been completely entered, that is, its tables, columns and relationships, it can be displayed either graphically within the implemented data modeler, or through an ER modeler such as DeZine (www.datanamic.com).

To diagram within DeZiner, the data model must imported. DeZiner, like many tools has an SQL DDL import facility. Therefore, the metabase has an SQL DDL export facility. Additionally, another data modeling tool may have been used to create what is seen as an implemented data model. Thus, the metabase has an SQL DDL import facility.

6.2.5.1 Export

Figure 47 presents the first screen in this process. The top browse shows the schemas. The second browse shows an alphabetical listing of the tables within that schema. If you select the Movies Schema and then the Movie Rentals Record, and then press the Display Implemented Data Model is pressed, then a screen like that of Figure 48 is presented.

The process that is invoked starts with the highlighted table and traces through all its foreign keys to then "know" all related tables. The hierarchy that is displayed in Figure 46 is all the descendants of the table and all the direct ancestors (no uncles or aunts). When an ancestor table is displayed it's color is red. Descendent tables are blue and the originally highlighted table's color is black.

As each table in Figure 48 is highlighted, the surrounding browses then display the table's primary key and columns, foreign keys and columns, and columns. If the Print Tree button is pressed a hierarchy tree is sent to the default printer. If the Print Tree Detail button is pressed the data model tree is printed along with an additional level of detail.

Figure 47 also has a set of buttons at the bottom. The first button, Select Output File for Generated DDL, causes a Select File display as presented in Figure 49. A default file name is presented that can be accepted or changed. The default directory is the current working directory.

Once the output file is selected, two buttons are then available for selection. The first is Generate Schema Based Data Model button. If selected, all the tables within the schema area are accessed and then linguistically expressed using SQL. The value of this DDL file is that it can then be imported by another software package.

The Generate Table Based Data Model button only generates SQL DDL for all tables that are descendants and direct ancestors of the highlighted table.

There are three options for representing subtyped tablees: One, Each, and SQL:1999. The one option causes all the columns from the contained subtyped tables to appear within the root table. The Each option causes each subtyped table to be expressed as a separate SQL table with its primary key the same as the root table's and the foreign key column reference to also be the same as it's primary key. This ensures a 1:1 relationship.

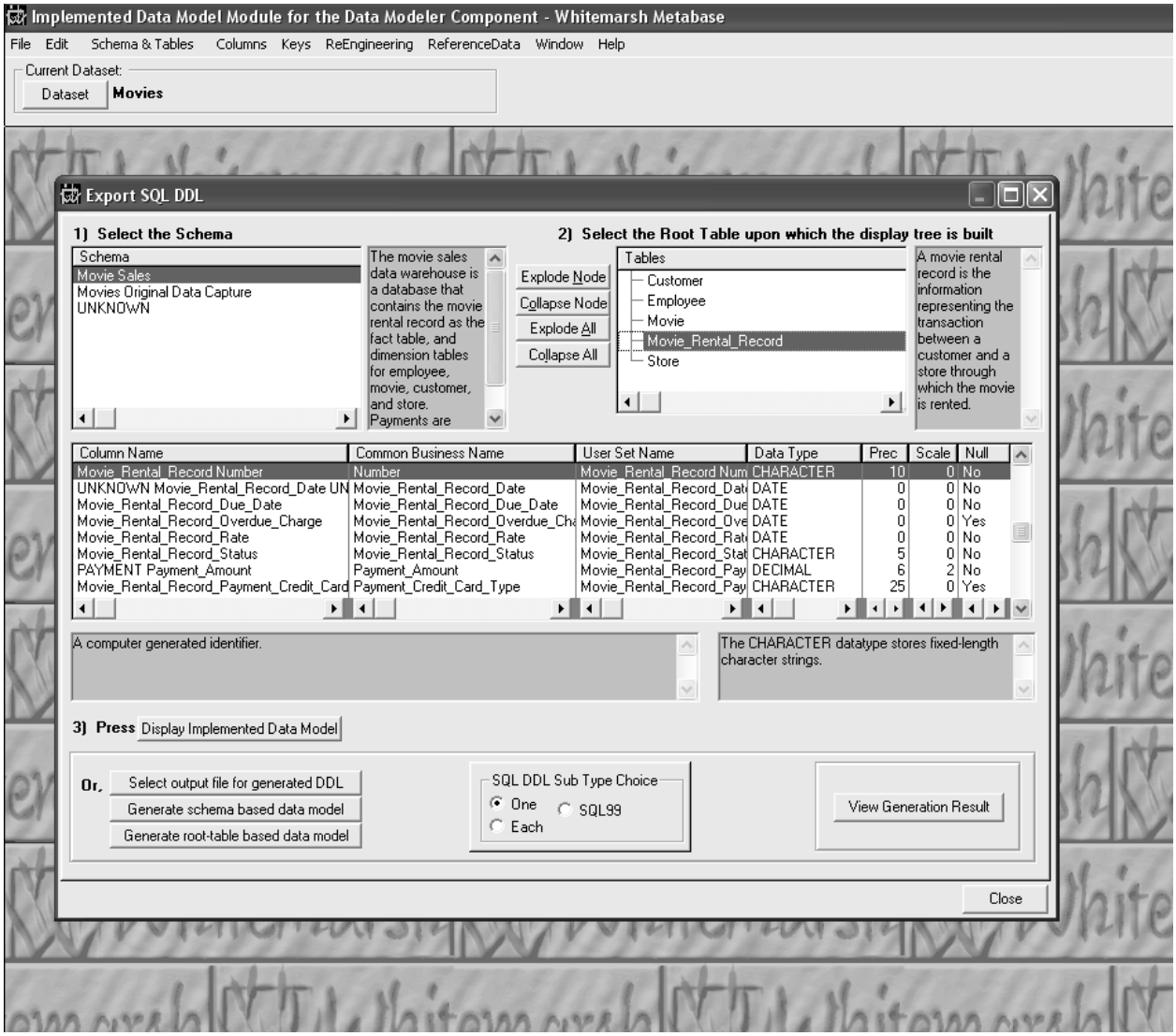

Figure 47. Export SQL DDL screen.

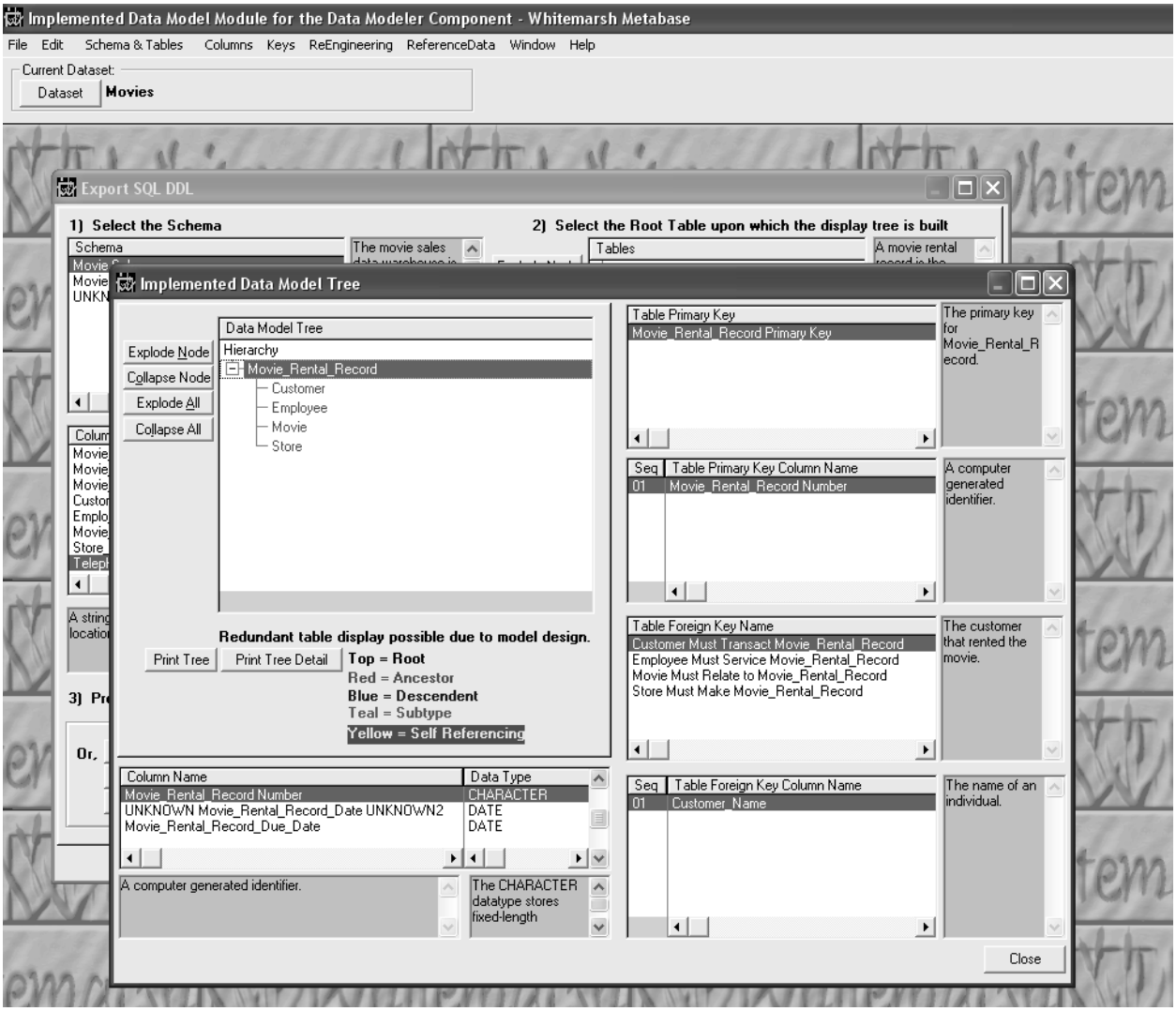

Figure 48. Data Model tree.

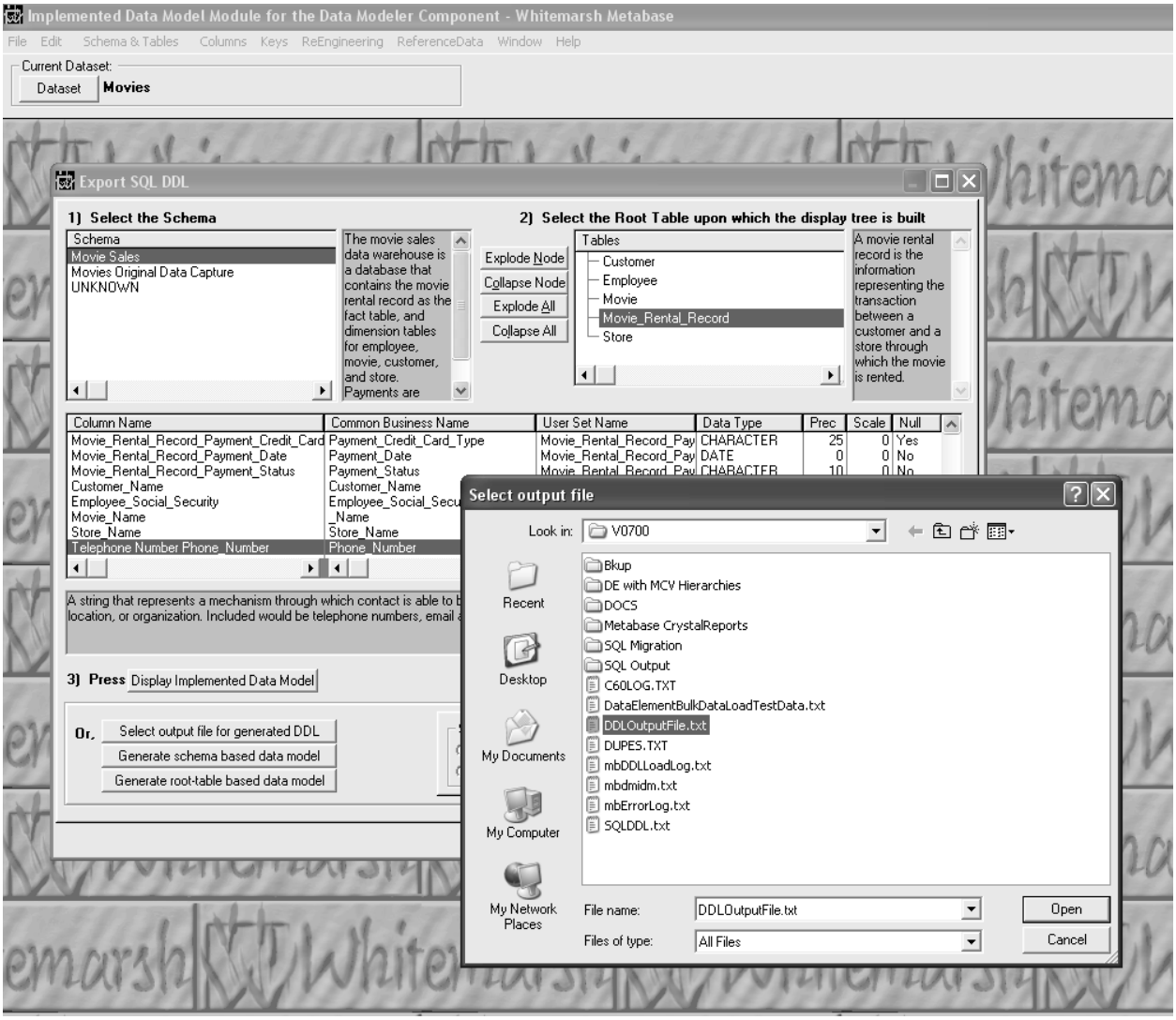

Figure 49. Selecting an output file for SQL DDL export.

To display the SQL DDL file, press the button, View Generation Result. Another Select File window is presented. Highlight the file from within the working directory and then press the Open button. SQL DDL as presented in Figure 50 is displayed.

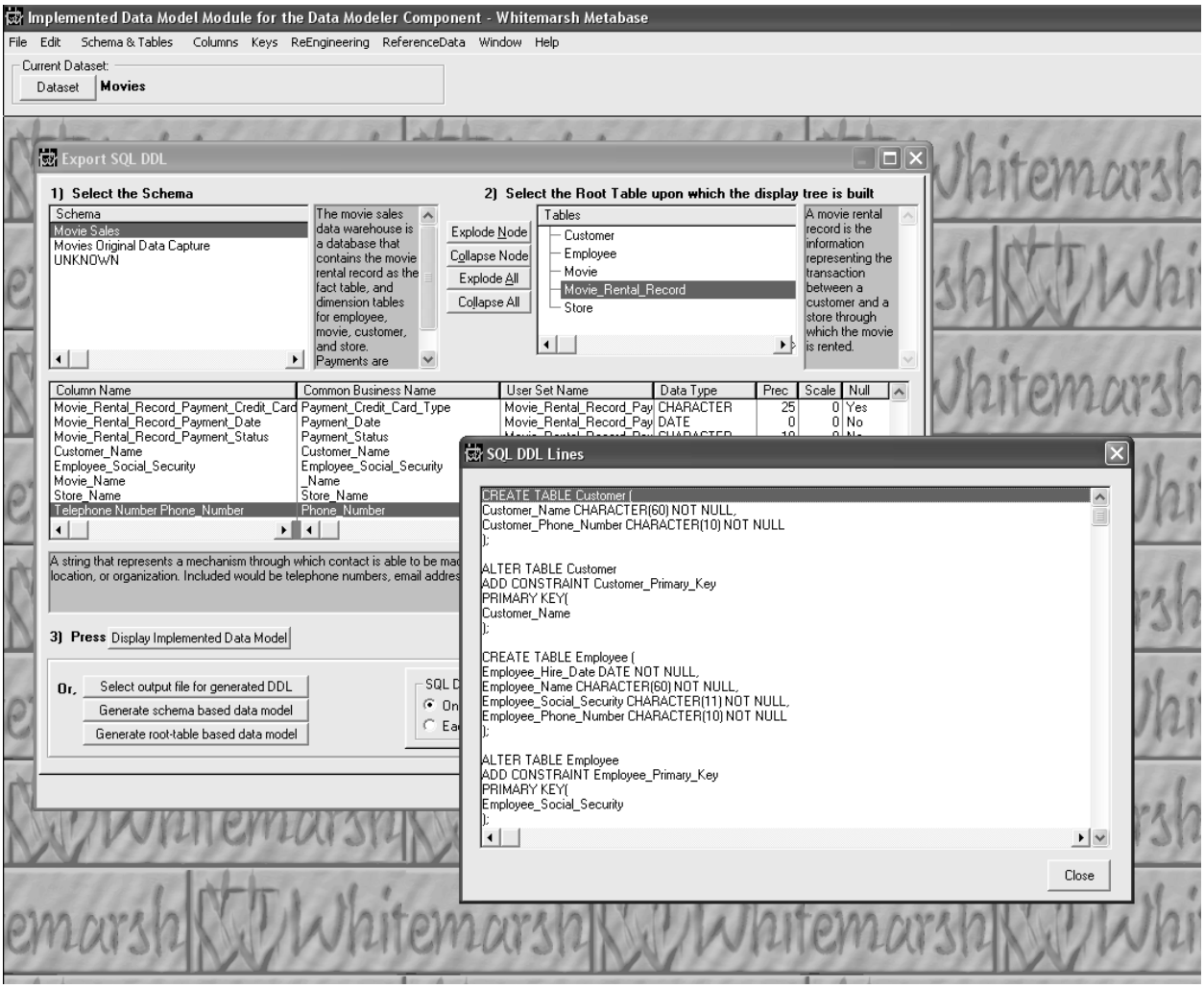

Figure 50. SQL DDL output file display.

6.2.5.2 Import SQL DDL

SQL DDL streams can be imported directly into the implemented data model. Figure 51 presents the first screen of this process. Fundamentally, the process is to first create a schema, then identify the SQL DDL text file, press the View SQL file if desired, and finally, press the Import SQL File button. The process will scan the SQL DDL file to determine if there are any errors. If there are errors then the entire loading process is aborted. If there are no errors then the SQL DDL is loaded. Created are tables, columns, data types, primary and foreign keys.

Figure 51 presents the first screen. If the schema does not already exist then press the Insert button. The update Schema screen is presented. Once the schema is created, then highlight the just created schema and proceed.

If for whatever reason the schema is to be deleted, then press the Delete button. The schema and all associated tables, columns and keys will be removed. The delete process will of course not start if any column of the loaded schema participates in a database object or is related to any DBMS table column.

Figure 52 presents the screen that appears when the Select SQL file button is pressed. Once the file is found and fills the File Name data entry box then press the Open button. At that point the file is selected. To ensure that the correct file has been selected, the View SQL file button can be pressed. The SQL DDL that has been selected is then presented in a text screen window like the one in Figure 53.

Not all forms of SQL DDL can be imported. At this time, all Primary and Foreign Keys must be in the Alter Table format.

Then the Import SQL File button can be pressed. The stream of SQL DDL is scanned for errors. If none are found the import process commences. If any errors are found they are reported and the import process stops at the point of the error. A log file can be created of the importing process. Figure 54 presents a screen of the log file lines.

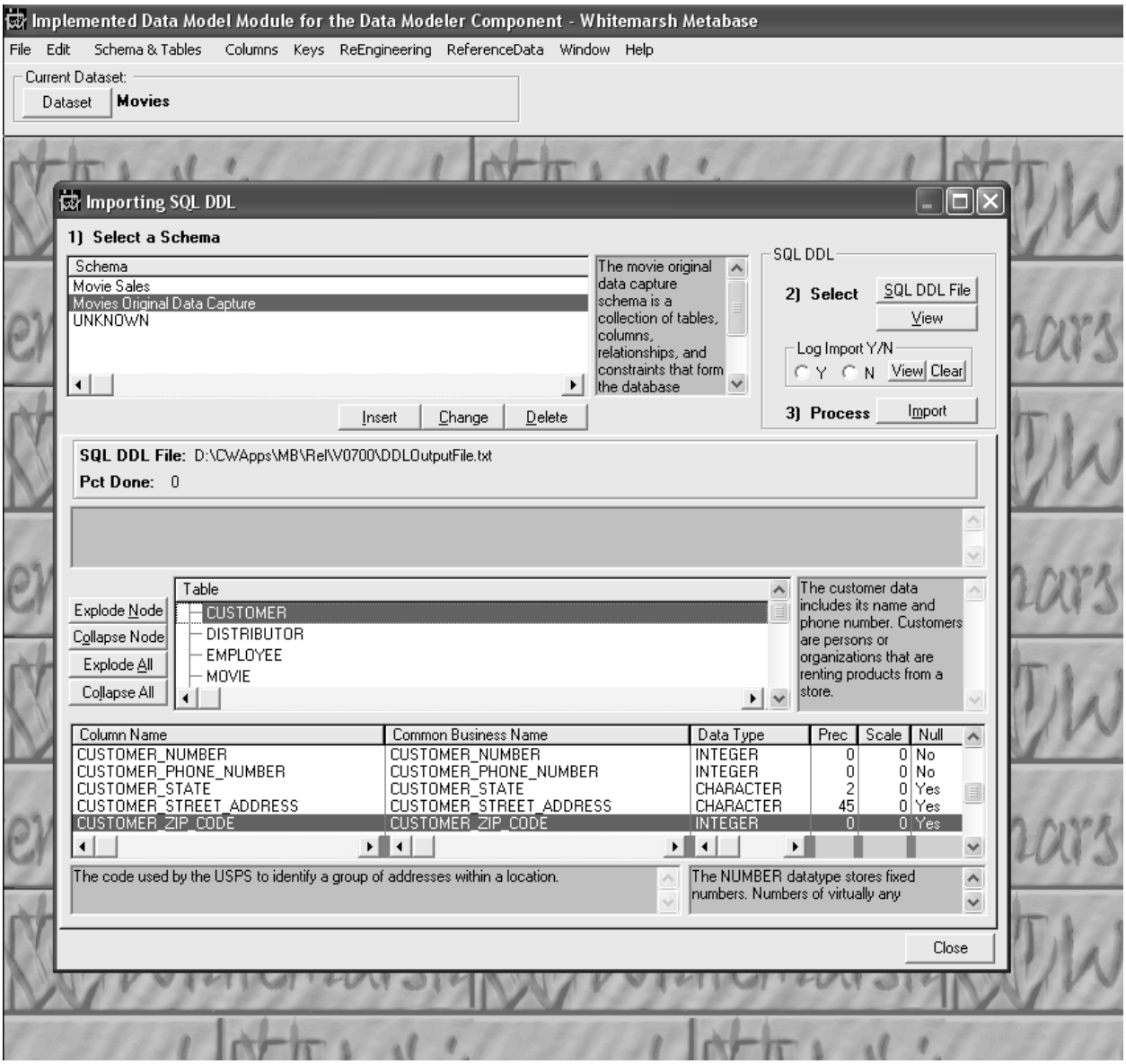

Figure 51. SQL DDL Import screen.

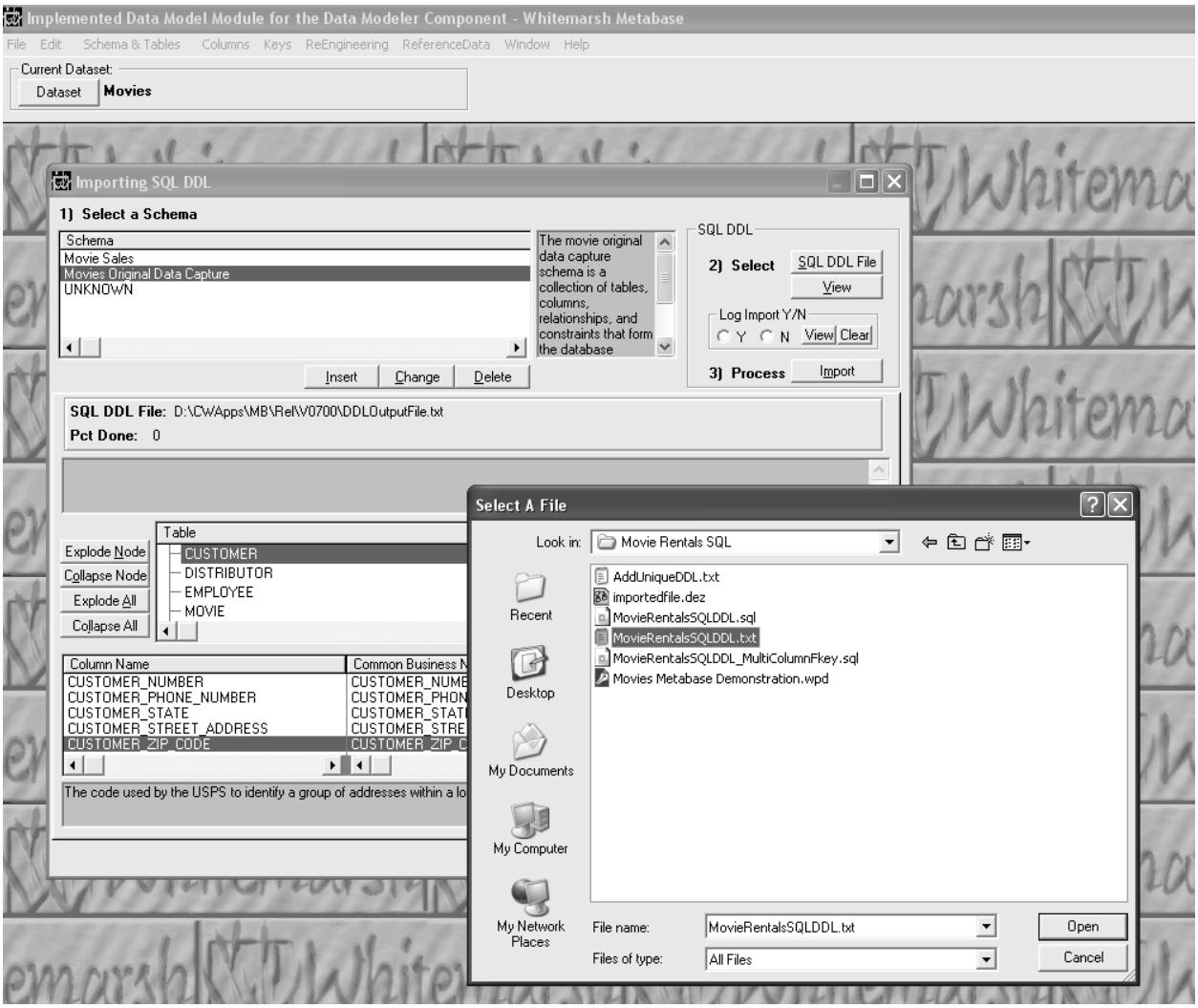

Figure 52. Selecting a SQL DDL file for importing.

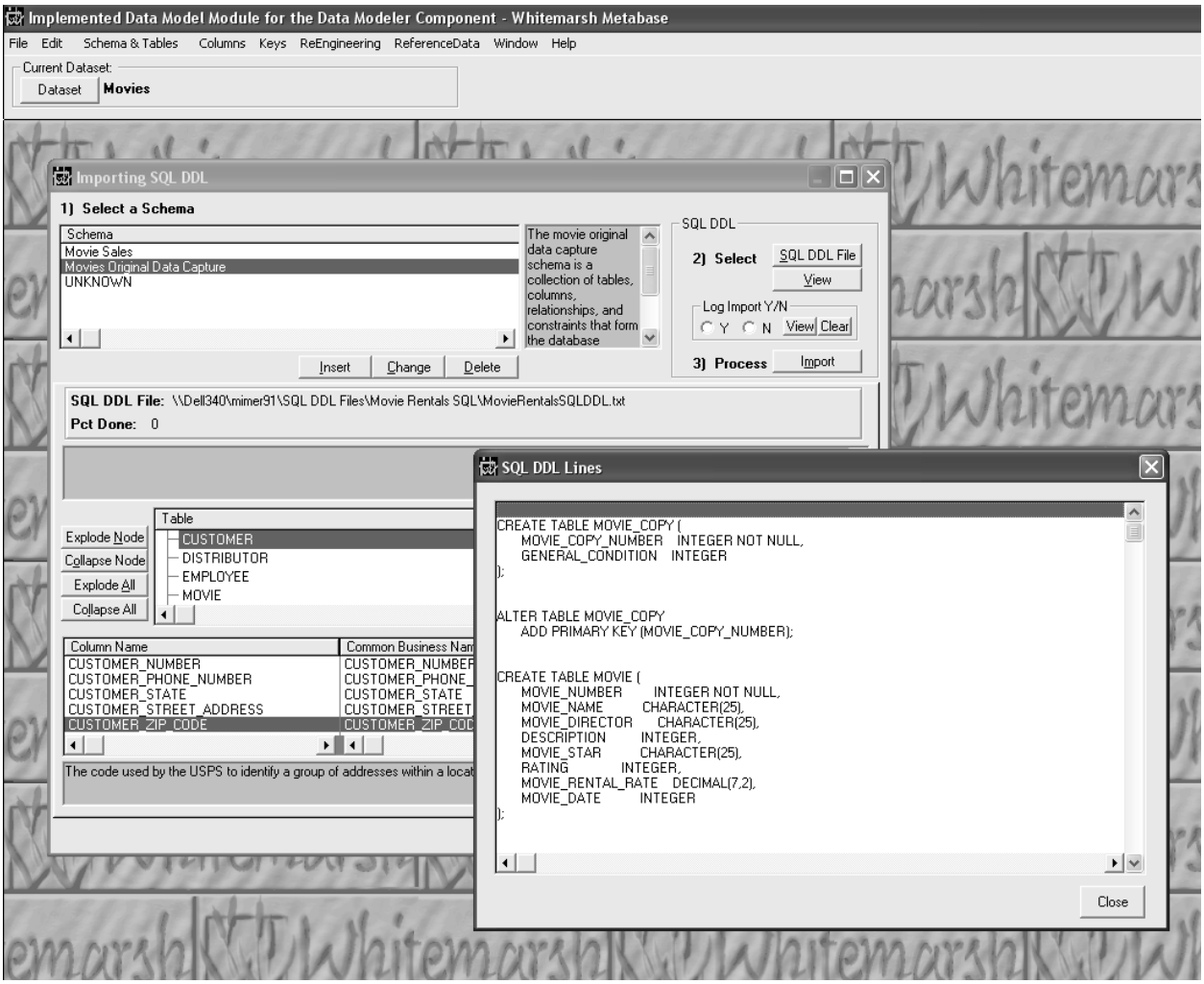

Figure 53. Example of an SQL DDL import file.

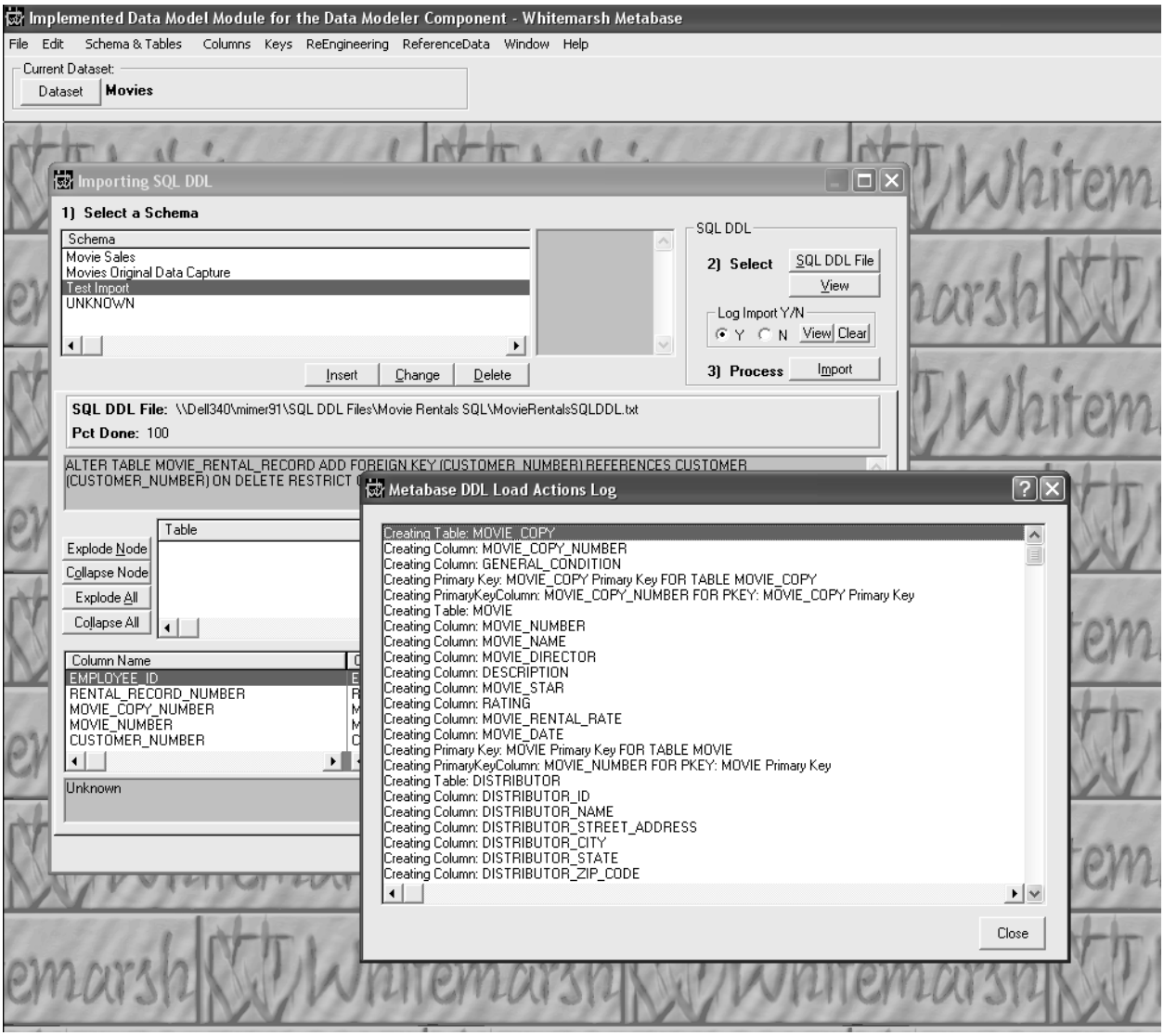

Figure 54. Log file display of the SQL load process.

6.3 Reports

Reports are accomplished through access to a particular metabase database instance through commercial report writers such as Crystal Reports. Whitemarsh provides about 100 such report templates for Crystal Report access from the Whitemarsh website.