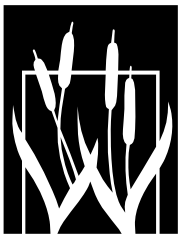

Whitemarsh
Information Systems Corporation

Whitemarsh Metabase
Data Modeler: Operational 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 operational data model (ODM) component of the data modeler module is designed to capture specifications of DBMS specific. Truly these are physical database designs. The key characteristic of an operational data model is that its collection of DBMS tables, DBMS columns, and DBMS relationships within a single database design targeted towards a specific DBMS, hardware, and operating system environment..

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

DBMS column semantics are inherited exclusively from its foreign key related column from within a table of an implemented data model.

If an organization already has operational databases (usually only about 100% of the time), then, there is a SQL DDL import facility that can cause the creation of operational data models from SQL DDL streams that are generated from DBMSs or from CASE tools, like Erwin. Operational data models can also be promoted to an the implemented data model. Thus, if there are multiple operational data models that have grown up over time within a single functional area, for example, human resources, then the "biggest" can be selected for promotion from operational to implemented. That promoted implemented data model should be transformed to a third normal form design so that it is more of a "logical" design. Then the other imported HR data models can be mapped to the single promoted implemented 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.

F7 invokes automatic spell checking on all text fields like names and descriptions.

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 operational data model module depicted in Figure 1.

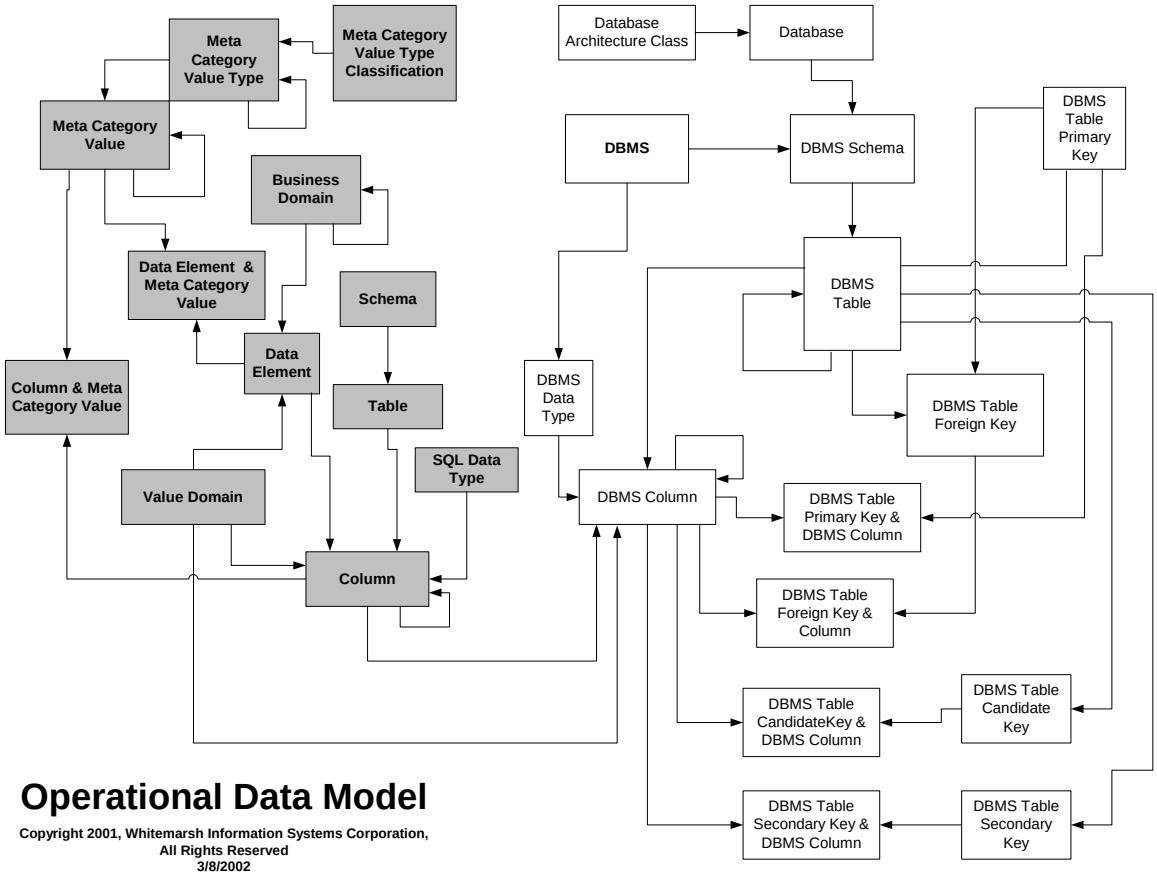

Figure 1. Operational data model meta model design.

The operational data model has the following DBMS tables:

- Database
- Database Architecture Class
- Database Nature
- Database Production Status
- DBMS
- DBMS Column
- DBMS Schema
- DBMS Table
- DBMS Table Candidate Key
- DBMS Table Candidate Key & DBMS Column
- DBMS Table Foreign Key
- DBMS Table Foreign Key and DBMS Column
- DBMS Table Primary Key
- DBMS Table Primary Key & DBMS Column
- DBMS Table Secondary Key
- DBMS Table Secondary Key & DBMS Column
- DBMS Data Type
- DBMS Data Type Picture

Explicit in this database design are the following:

- A database is named collection of DBMS tables within the structure of a DBMS schema. A well designed database presents a unified set of policy evidenced through the rows of data within a specific subject area.

- Database Architecture Class is a characterization of a database. That is, original data collection, transaction data staging area (TDSA), subject area database, data warehouse (wholesale or retail), or reference data. An explanation of these database architecture classes is presented in a paper named, Data Architectures, that is on the Whitemarsh website.

- Database Nature is a general classification of the business type and use of the database. Value examples would include operational, control, management.

- Database Production Status is a classification of the status of the database. Commonly employed values are development, test and production.

- DBMS a software system that defines, creates, accesses and maintains databases.

- DBMS columns are the manifestation of the semantics of a column within a DBMS table of a DBMS schema. Not all the DBMS columns of a DBMS table must map to attributes from a single table.

- A DBMS data type is a DBMS vendor's 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 DBMS 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.

- DBMS Data Type Picture is the picture clause associated with a data type so that a complete DDL script can be produced for the Clarion for Windows application development environment.

- A DBMS table is intended to be a well defined expression of one policy within a DBMS schema. Ideally, the collection of all the DBMS tables within a DBMS schema area should define a coherent collection policy. A DBMS table may contains DBMS columns that map to columns from multiple tables. This enables operational databases to be non-third-normal form while the implemented data model is. Additionally the DBMS table can have formally defined subtypes.

- DBMS table candidate keys represent a collection of DBMS columns within an DBMS table that when their values are collectively employed would result in the retrieval or update of a single row of data for that DBMS table. There may be multiple candidate keys within an DBMS table. DBMS columns of candidate keys are not allowed to overlap each other or the DBMS table's primary key.

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

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

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

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

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

- DBMS table Secondary Key represents a collection of DBMS columns within an DBMS table that when their values are collectively employed would result in the retrieval or update of one or more rows of data for that DBMS. There can be multiple secondary keys within an DBMS table. DBMS columns of secondary key are allowed to overlap each other or the DBMS table's secondary key.

- DBMS table Secondary Key & DBMS column are the relationship between an DBMS table secondary key and the DBMS columns that comprise the key. The DBMS columns exist within a implemented sequence. Secondary key DBMS columns are not allowed to include any DBMS columns within the DBMS table's secondary key.

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

4 Reference Data

The reference data in the operational 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.

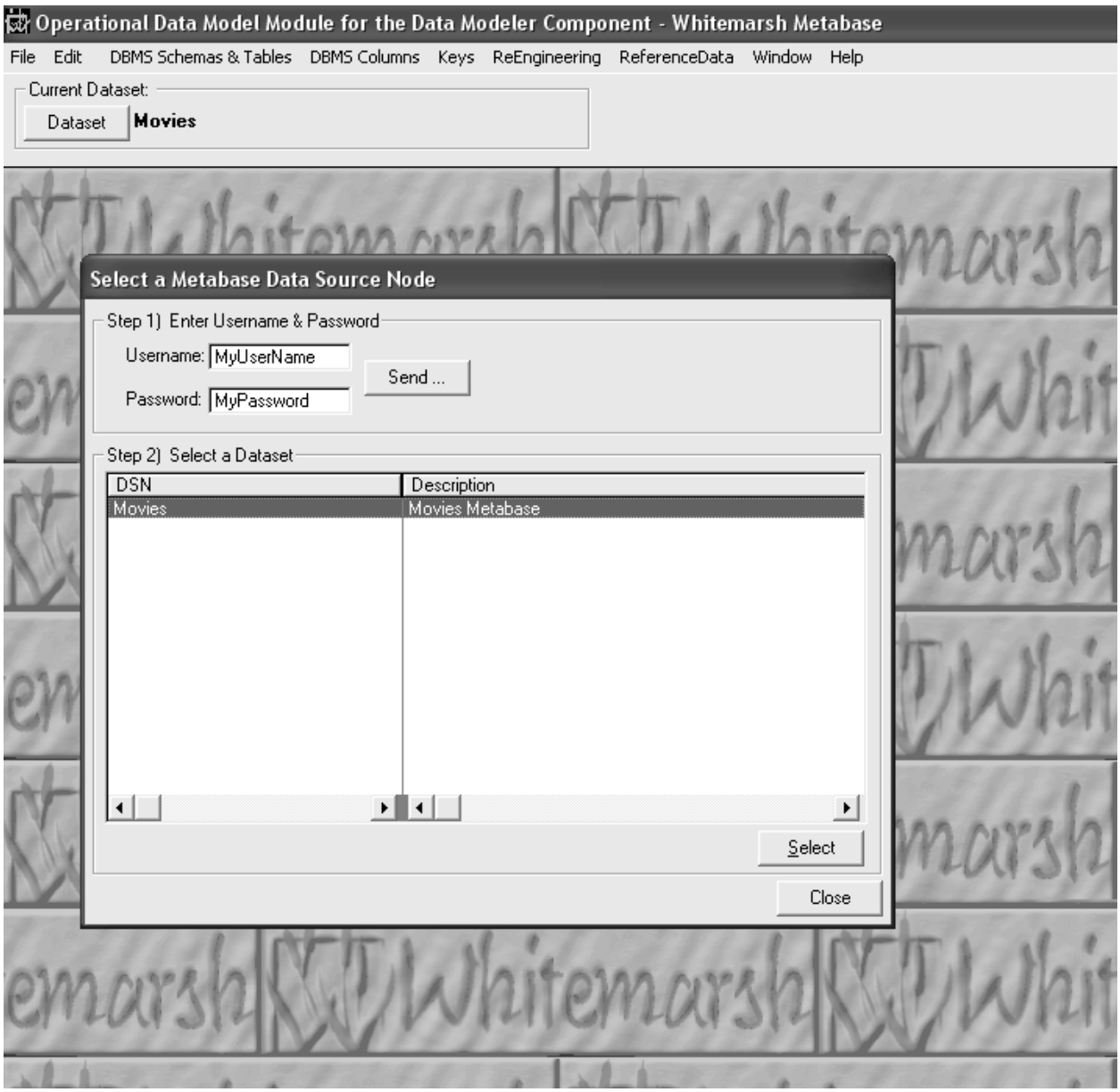

Figure 2. Login screen.

6 Process Model

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

- Reference Data
- Fact Data Entry
- Reports

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

- DBMS schemas & DBMS tables
- DBMS columns
- Keys
- Reference Data

Each menu item contains as appropriate, nested subordinate menu items. The complete menu is provided 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.

-- DBMS Schemas Tables
    -- DBMS Schema
    -- DBMS Table
    -- Data Model Tree

-- Import and Export
    -- Export To Clarion TXD

-- Import From Implemented Data Model
    -- Import Schema Table Set
    -- Import Table Tree from Schema
    -- Import Single Table from Schema
    -- Import Columns from Schema

-- SQL DDL
    -- Import
    -- Export

-- DBMS Columns
    -- DBMS Columns
    -- DBMS Column Value Domains
    -- Data Hierarchies

-- Keys
    -- DBMS Table Primary Key
    -- DBMS Table Primary Key Columns
    -- DBMS Table Foreign Key
    -- DBMS Table Candidate Key
    -- DBMS Table Candidate Key Columns
    -- DBMS Table Secondary Key
    -- DBMS Table Secondary Key Columns

-- ReEngineering
    -- Reassign DBMS Columns to Columns
    -- Reassign DBMS Column to DBMS Data Types
    -- Reassign DBMS Columns to DBMS Tables
    -- Synchronize Column Local Definitions
    -- Reassign DBMS Tables to DBMS Schemas
    -- Reassign DBMS Tables to DBMS Tables
    -- Promote Operational Data Model to Implemented Data Model
    -- Promote Operational Data Model Table To Implemented Data Model
    -- Remove DBMS Table DBMS Column Column Assignments

-- Reference Data

-- Database Information
    -- Data Architecture Class
    -- Nature
    -- Production Status

-- DBMS Information
    -- DBMSs
    -- DBMS Data Types
    -- DBMS Data Type Pictures

Menu for Operational Data Model

6.1 Reference Data

There are six types of reference data in the operational data model. These are:

- Database
- DBMS
- DBMS Data Type
- DBMS Data Type Picture
- Database Architecture Class
- Nature
- Production Status

6.1.1 Database

Figure 3 presents a list of databases. The information related to a database is its overall quantity of columns (fields), quantity of rows, size in megabytes, and it's status in terms of database architecture class, production status and nature. The complete description of a database is of course represented through its semantics, that is it's schema and then all the related tables, columns, and relationships. Databases may be implemented through one or more DBMSs. Figure 4 presents the update screen for adding or modifying a database. Entering the specific database architecture class, production status and nature involves entering a zero, then tabbing through. A selects screen then appears. Figure 5 illustrates the select screen for Database Architecture Classes. The select screens for Database Nature and Production Status are the similar.

6.1.2 DBMS (Database Management System)

Databases and DBMSs are very different things. A database is a collection of rows of data across a set of tables within a schema. In contrast, a DBMS is commercial vendor's product that manages a database. For example, Oracle, Sybase, or IBM's DB/2. Figure 6 presents a set of DBMSs. Figure 7 presents an update screen for DBMS.

6.1.3 DBMS Data Types

Figure 8 presents the list of DBMS data types. First highlight the specific DBMS then the appropriate data type within the DBMS. To then insert, change or delete a SQL data type press the appropriate button. Figure 9 presents the DBMS data type update screen.

6.1.4 DBMS Data Type Pictures

Figure 10 presents the list of DBMS Data Type pictures. First highlight the specific Data Type Picture, and then the various data types within the different DBMS data types are shown. To then insert, change or delete a DBMS Data Type Picture press the appropriate button. Figure 11 presents the DBMS Data Type Picture update screen.

6.1.5 Database Architecture Class

Figure 12 presents the list of Database Architecture Classes. To then insert, change or delete a Database Architecture Class press the appropriate button. Figure13 presents the Database Architecture Class update screen.

6.1.6 [Database] Nature

Figure 14 presents the list of Database Natures. To then insert, change or delete a Database Nature press the appropriate button. Figure 15 presents the Database Nature update screen.

6.1.6 [Database] Production Status

Figure 16 presents the list of DBMS data types. First highlight the specific DBMS then the appropriate data type within the DBMS. To then insert, change or delete a SQL data type press the appropriate button. Figure 17 presents the DBMS data type update screen.

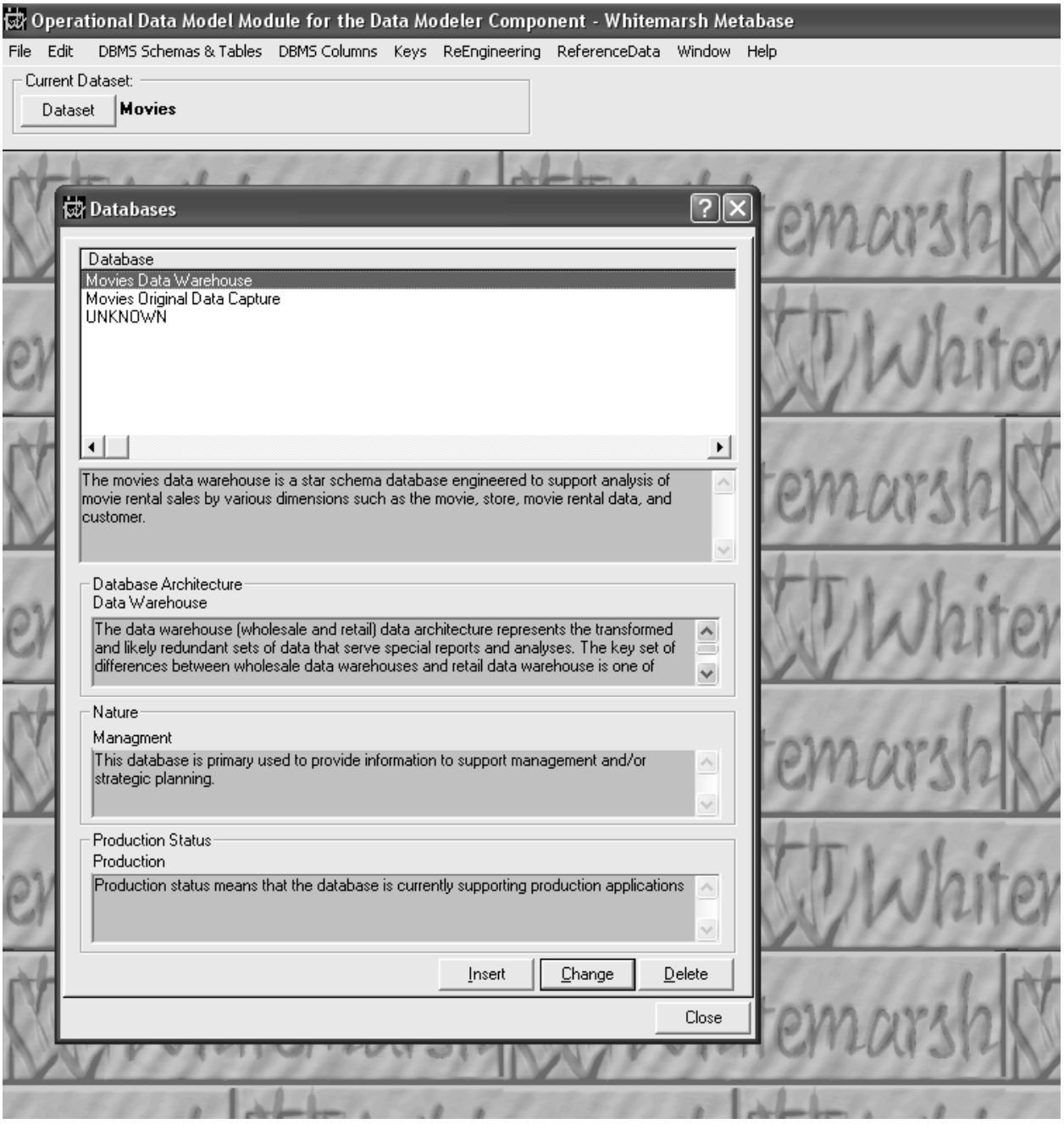

Figure 3. Database list.

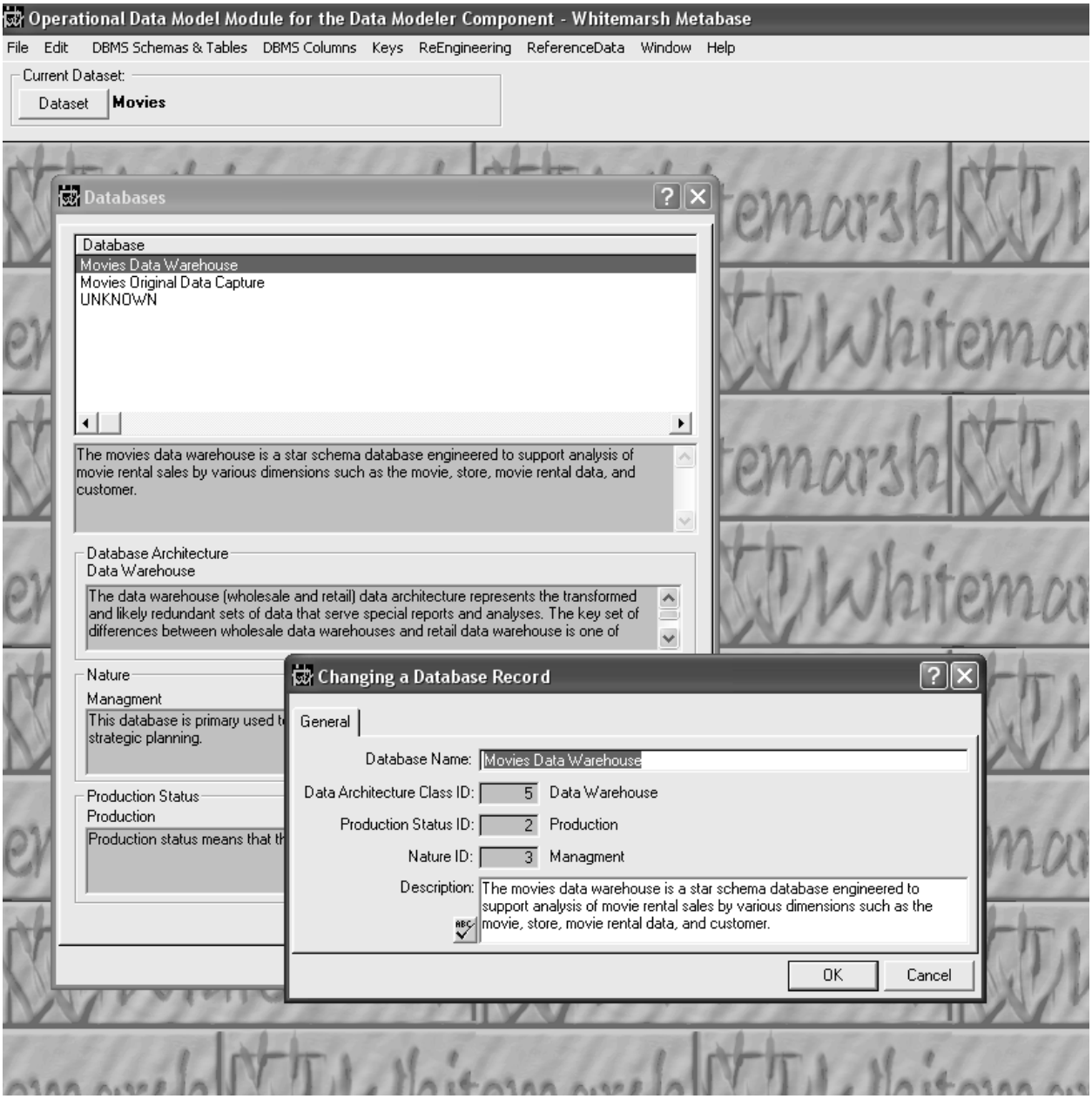

Figure 4. Database update screen.

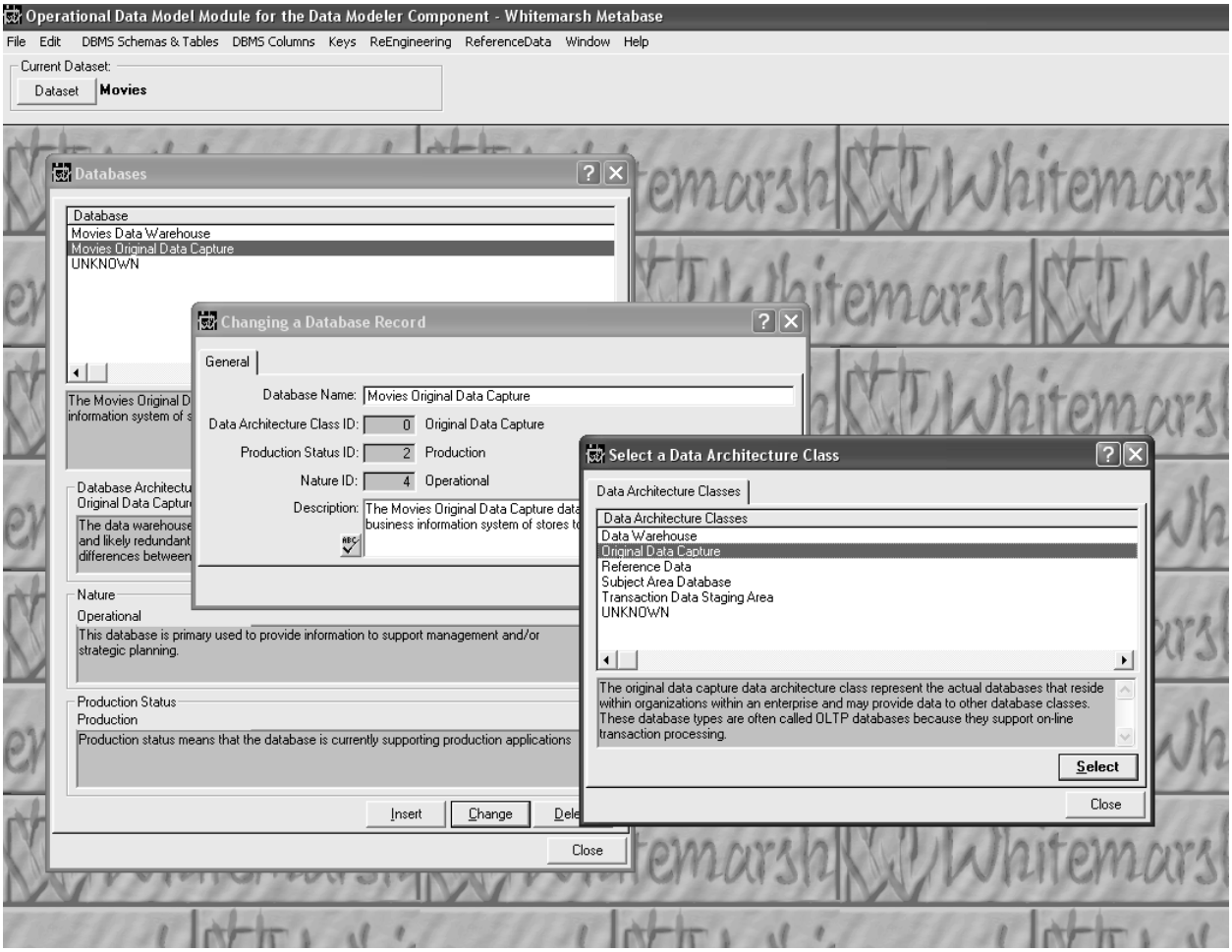

Figure 5. Data Architecture Class selection for a database.

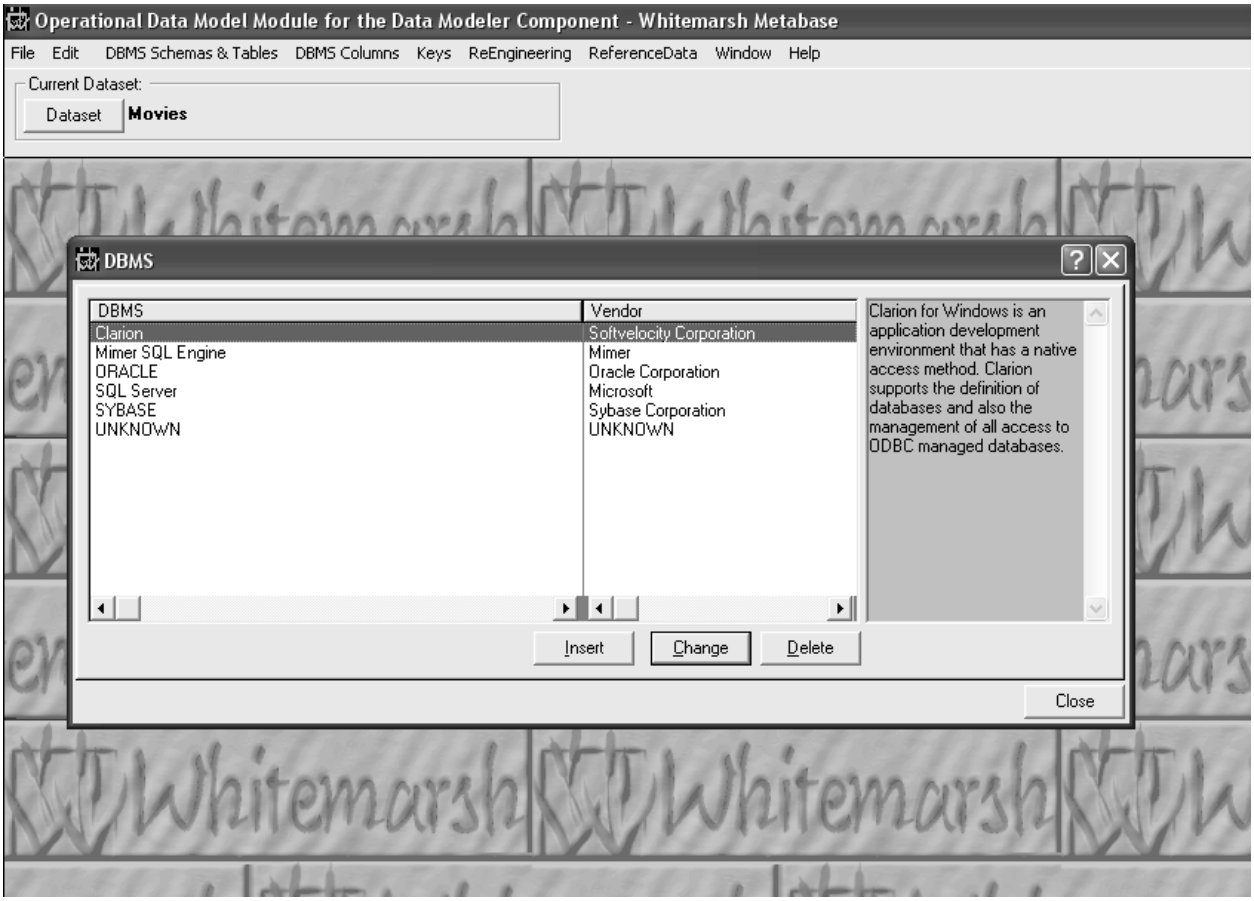

Figure 6. Database Management Systems (DBMS) list.

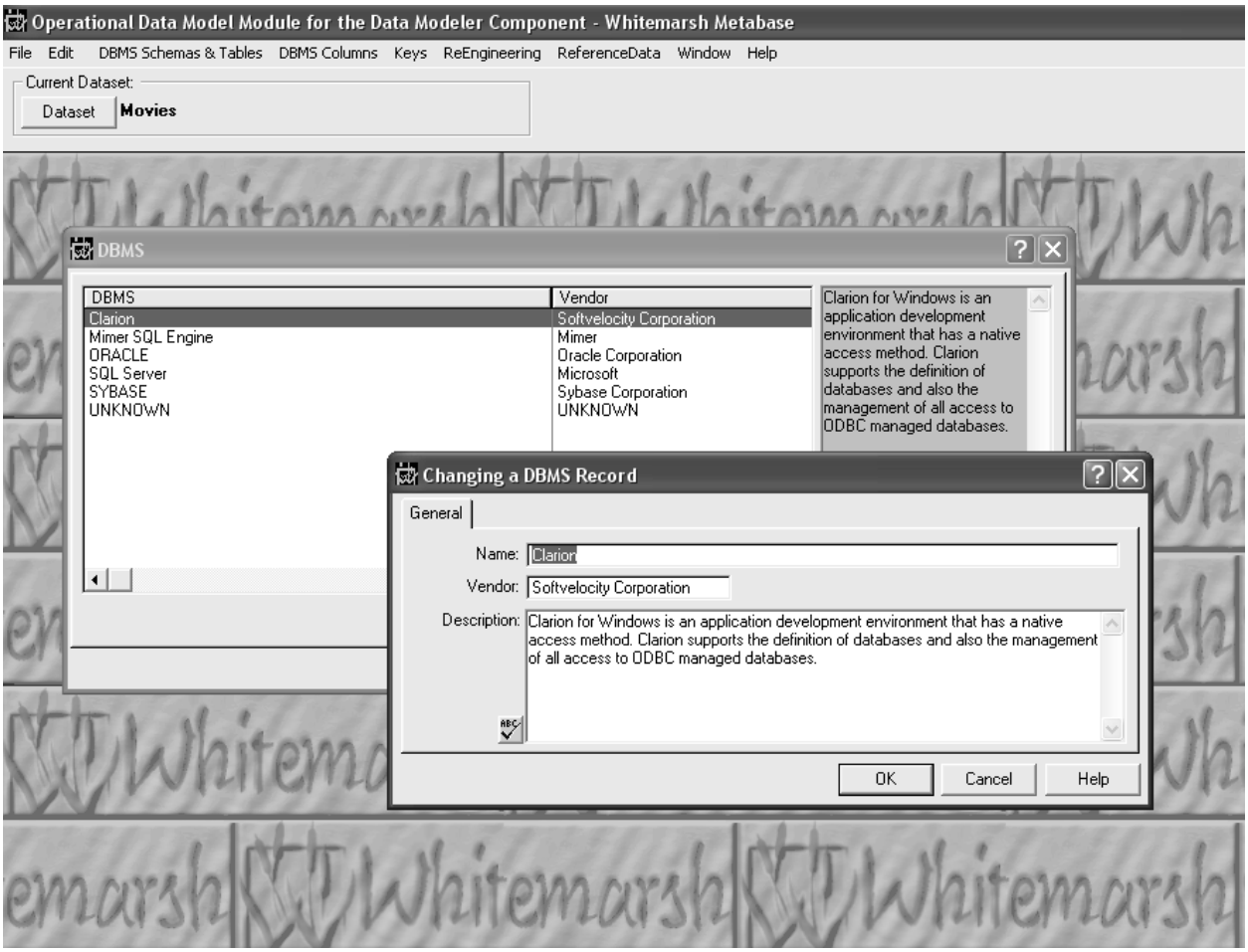

Figure 7. DBMS update screen.

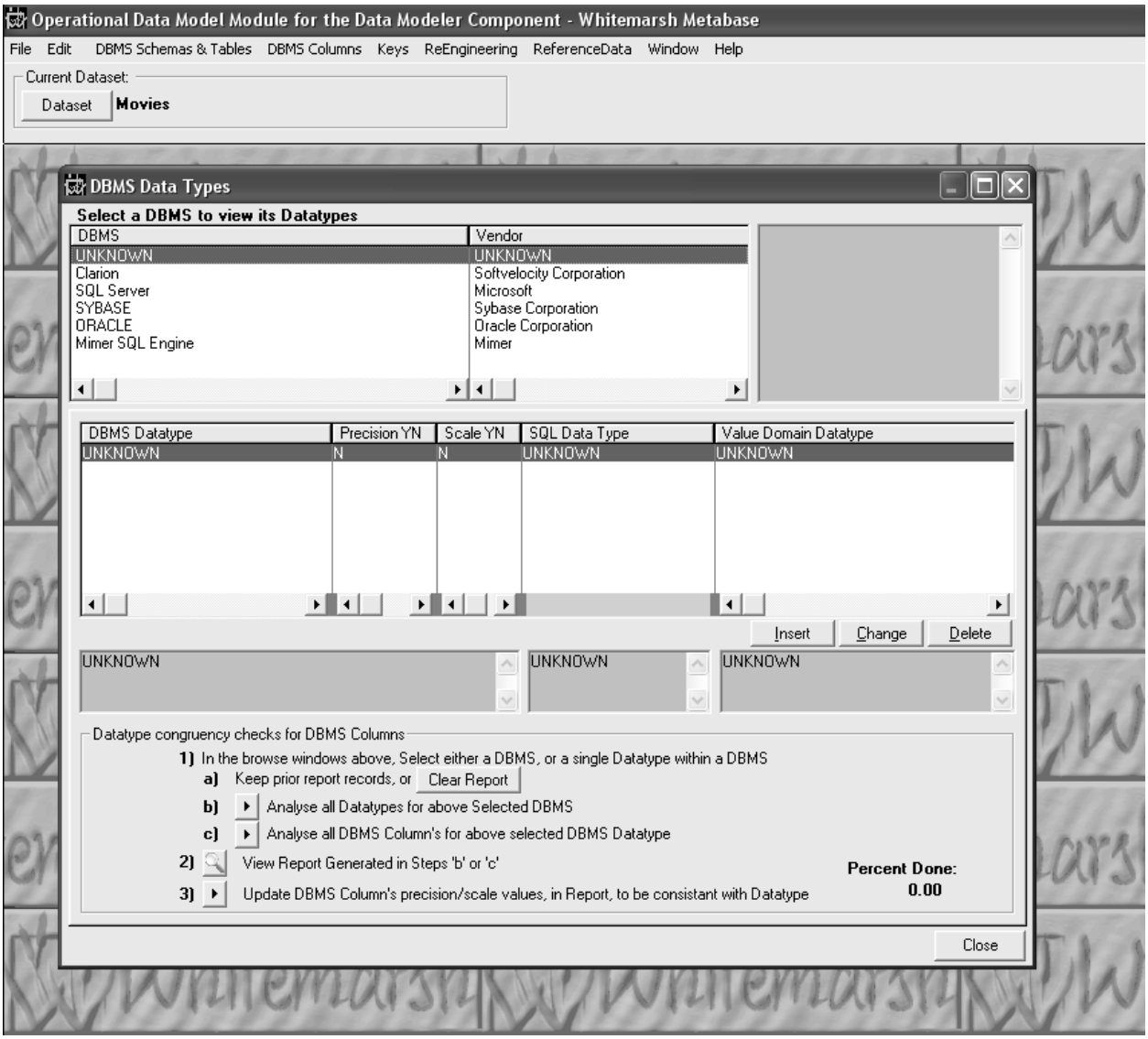

Figure 8. DBMS Data Types list.

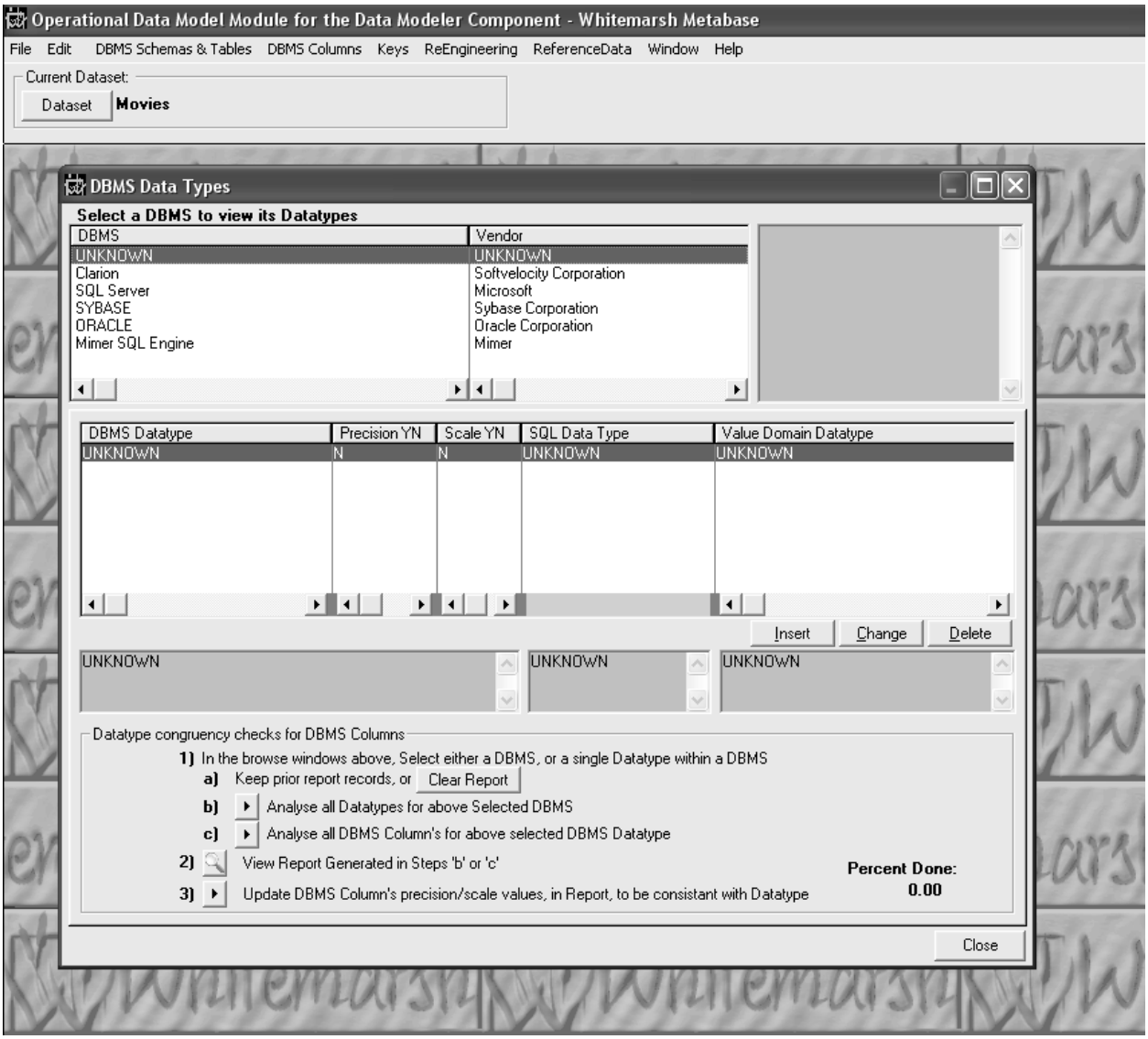

Figure 9. DBMS Data Type update screen.

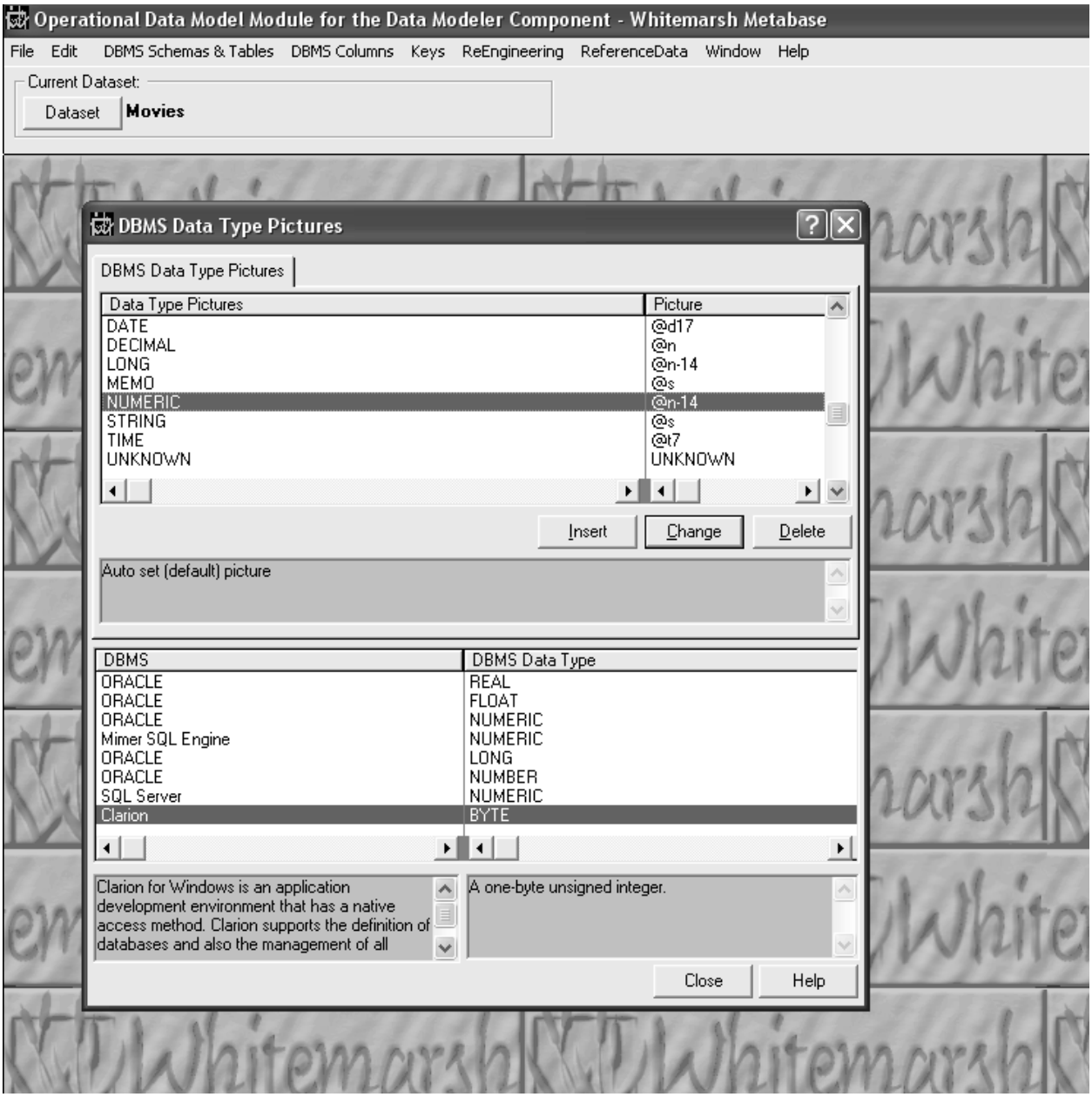

Figure 10. DBMS Data Type Pictures list.

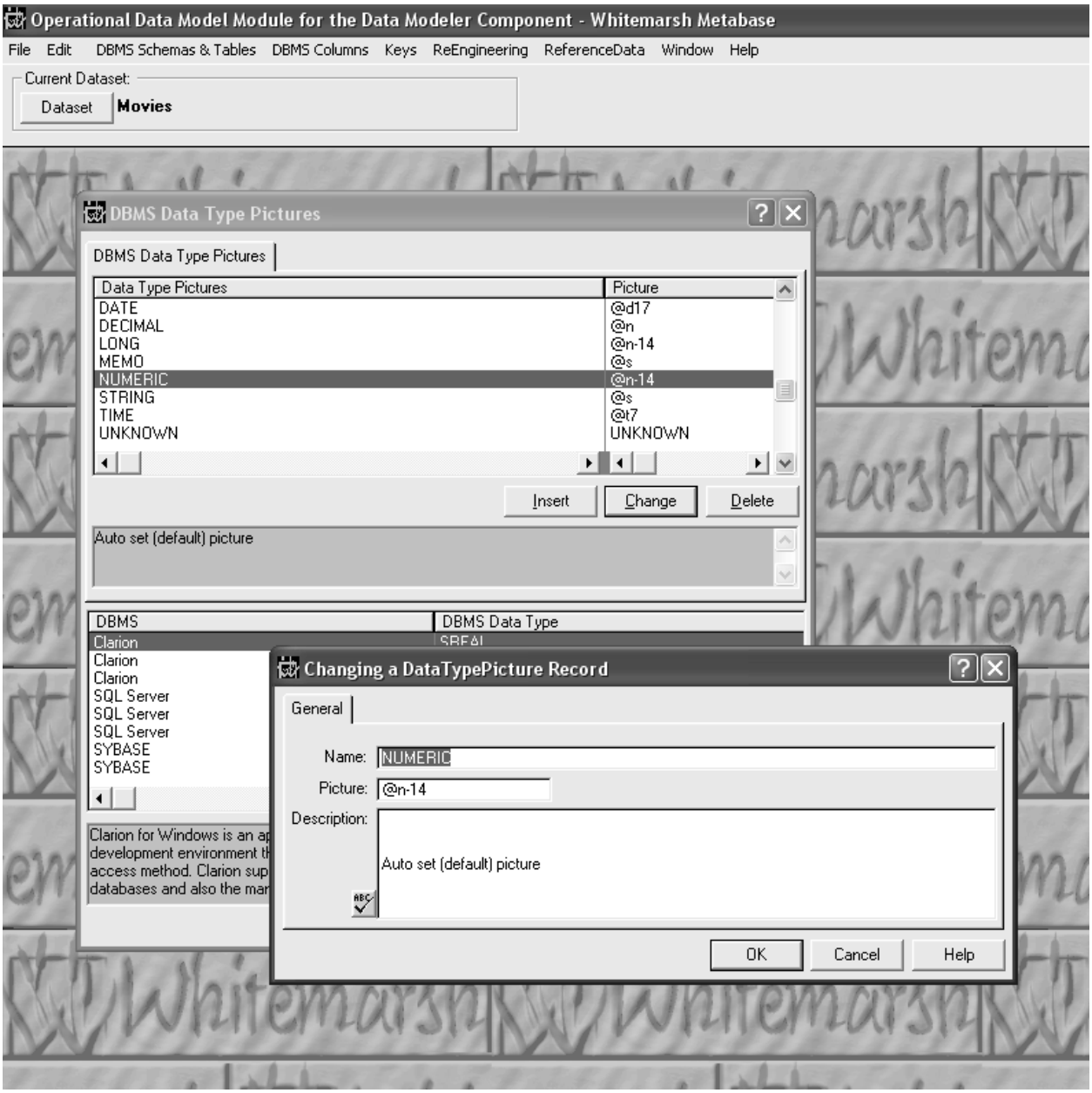

Figure 11. DBMS Data Type update screen.

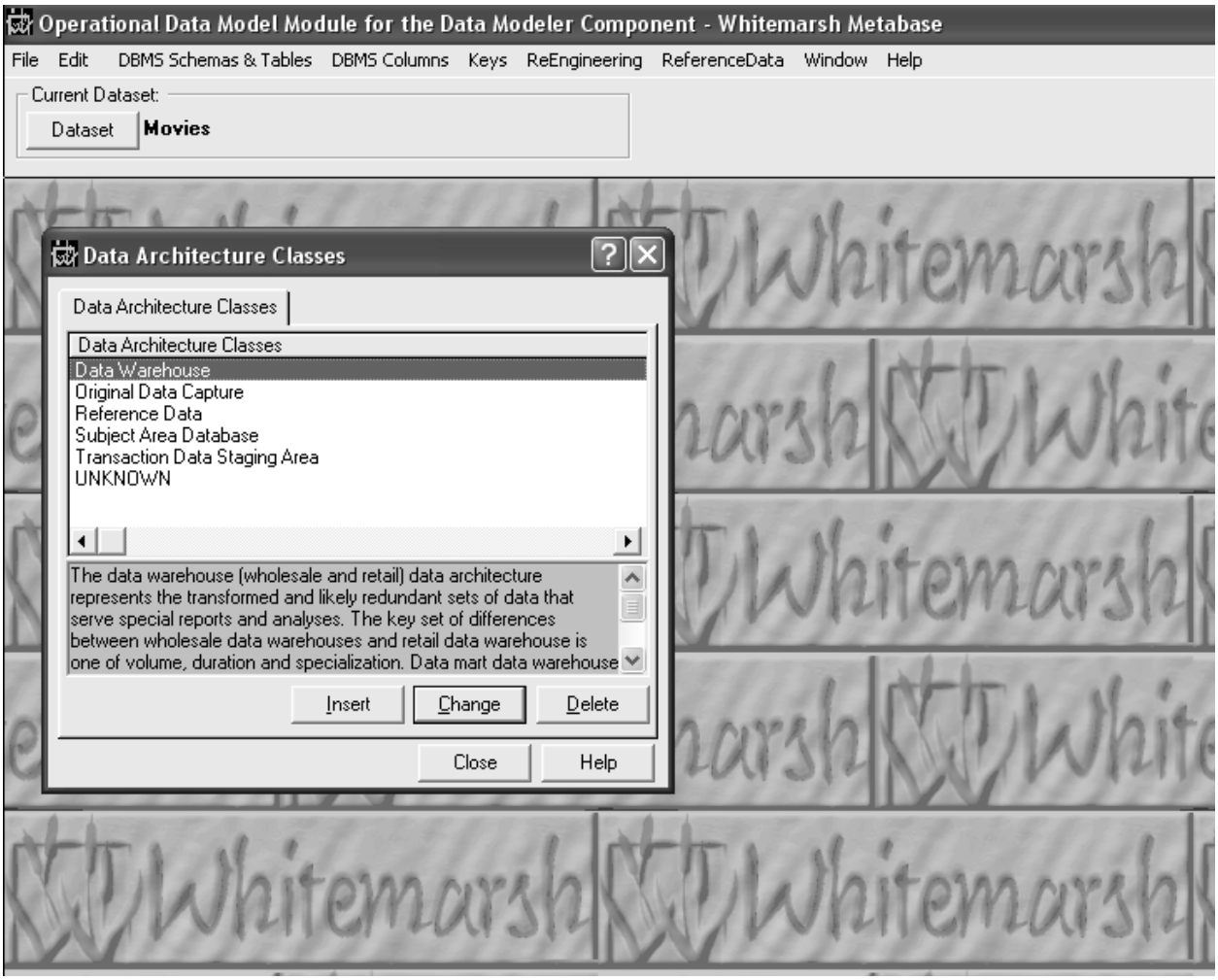

Figure 12. Data Architecture Classes list.

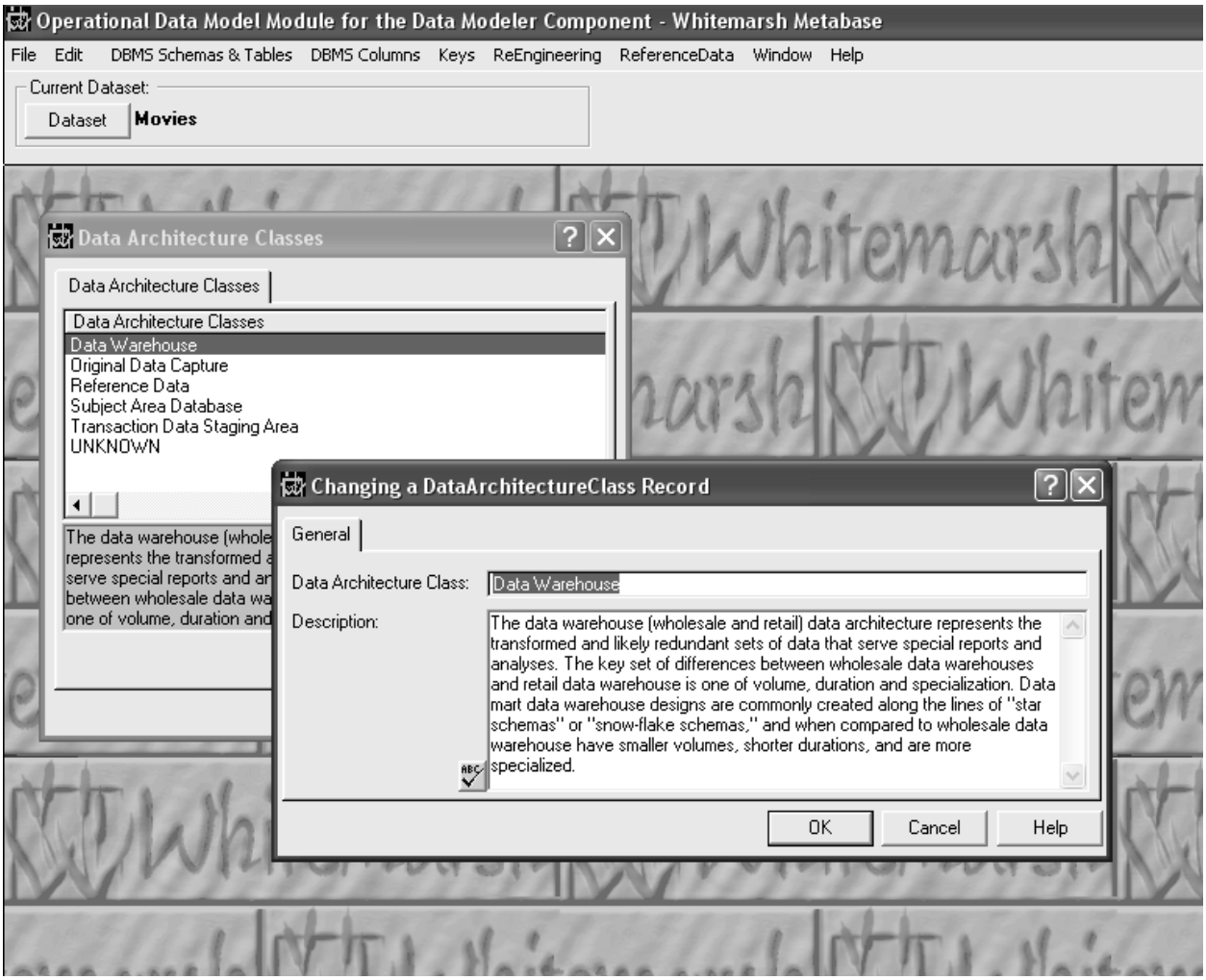

Figure 13. Data Architecture Classes update screen.

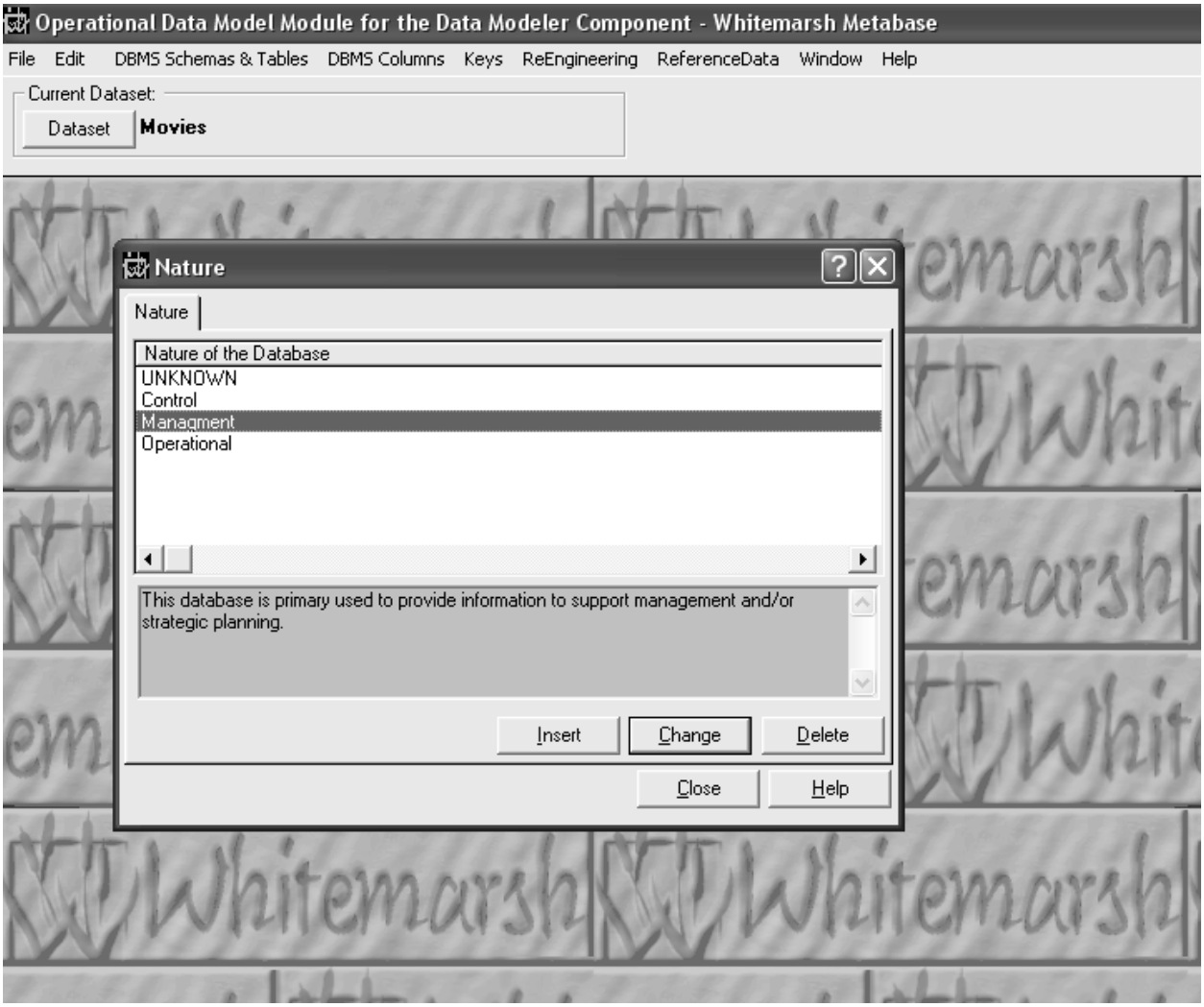

Figure 14. Database "Nature" list.

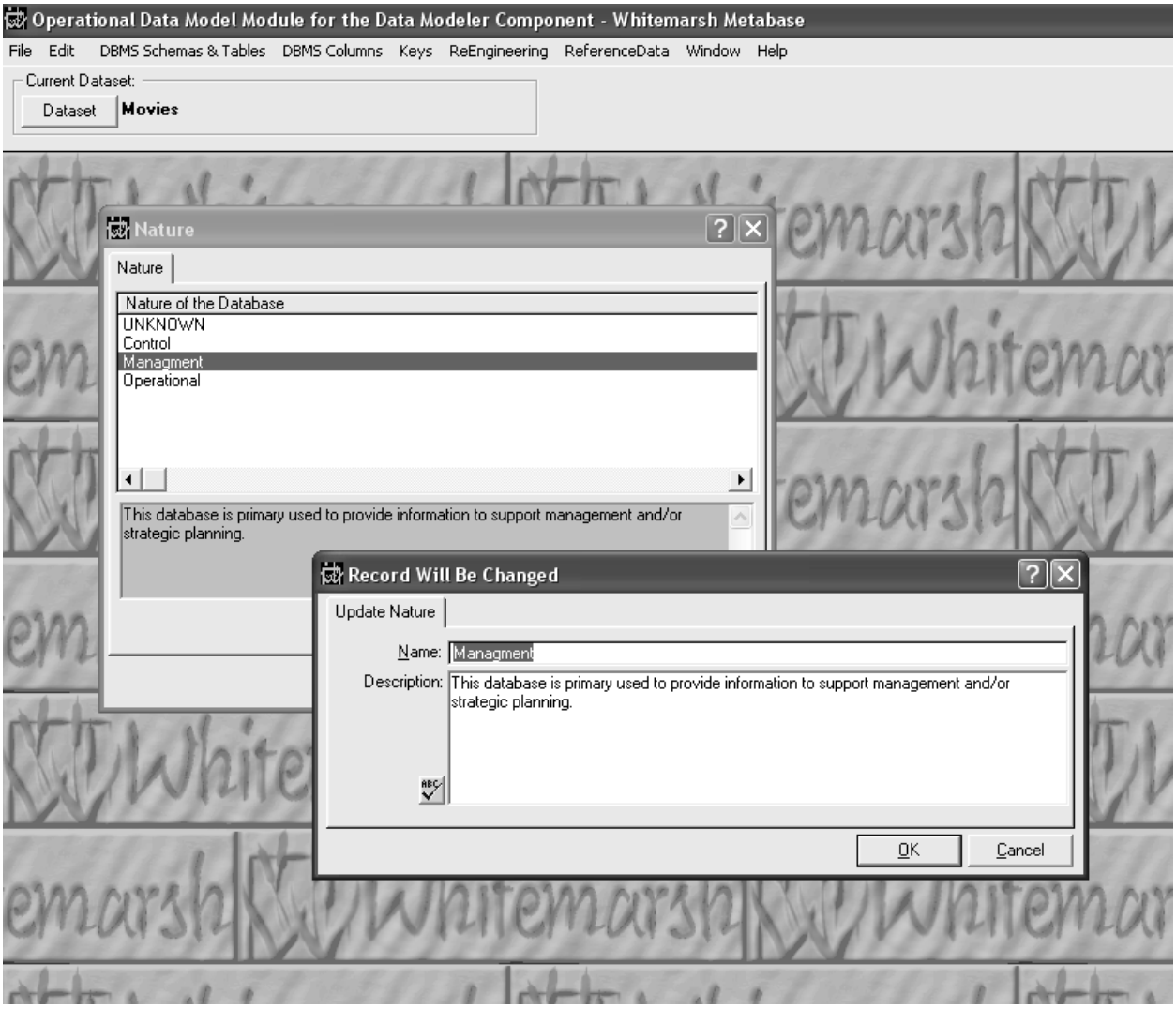

Figure 15. Database Nature update screen.

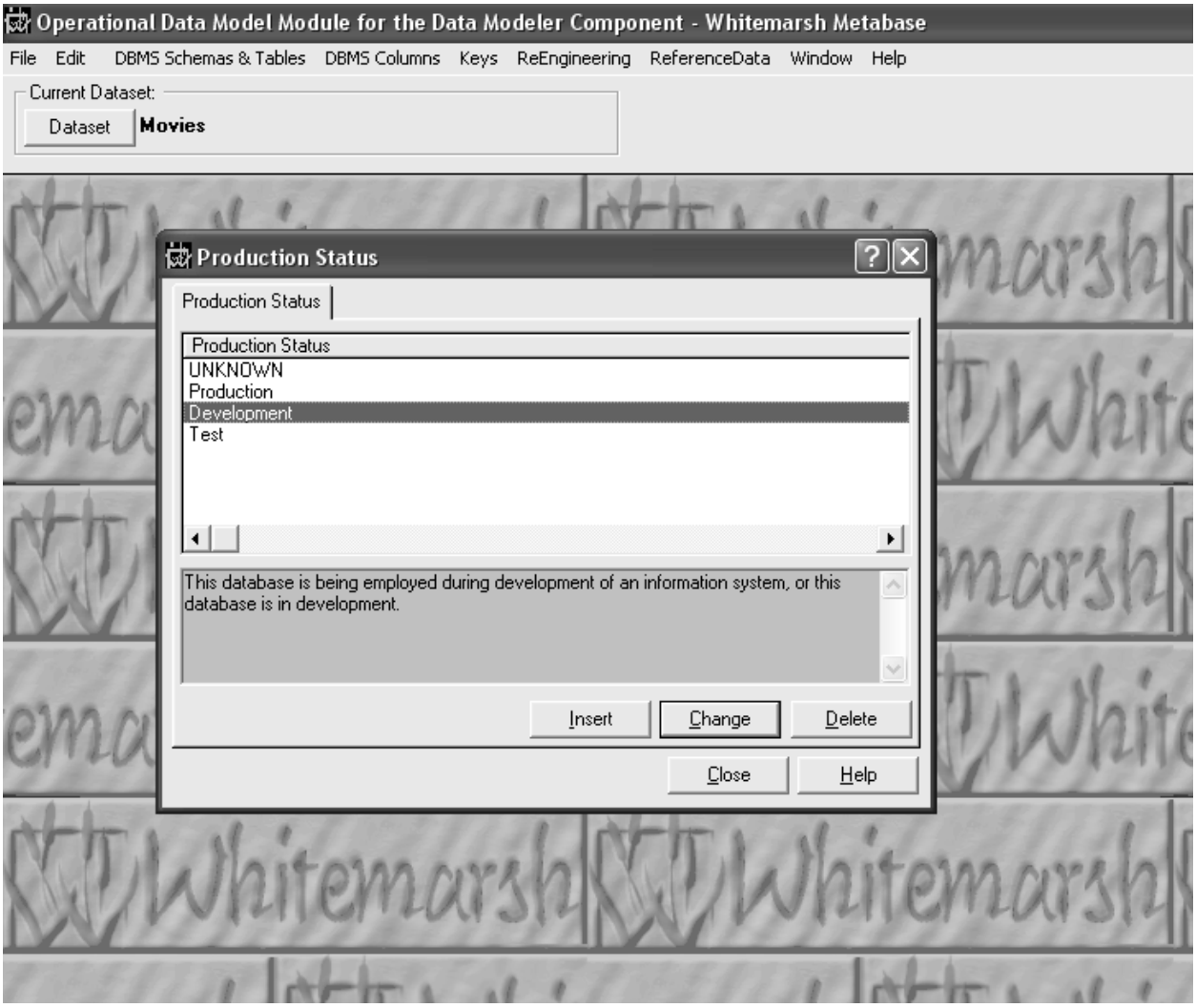

Figure 16. Database Production Status list.

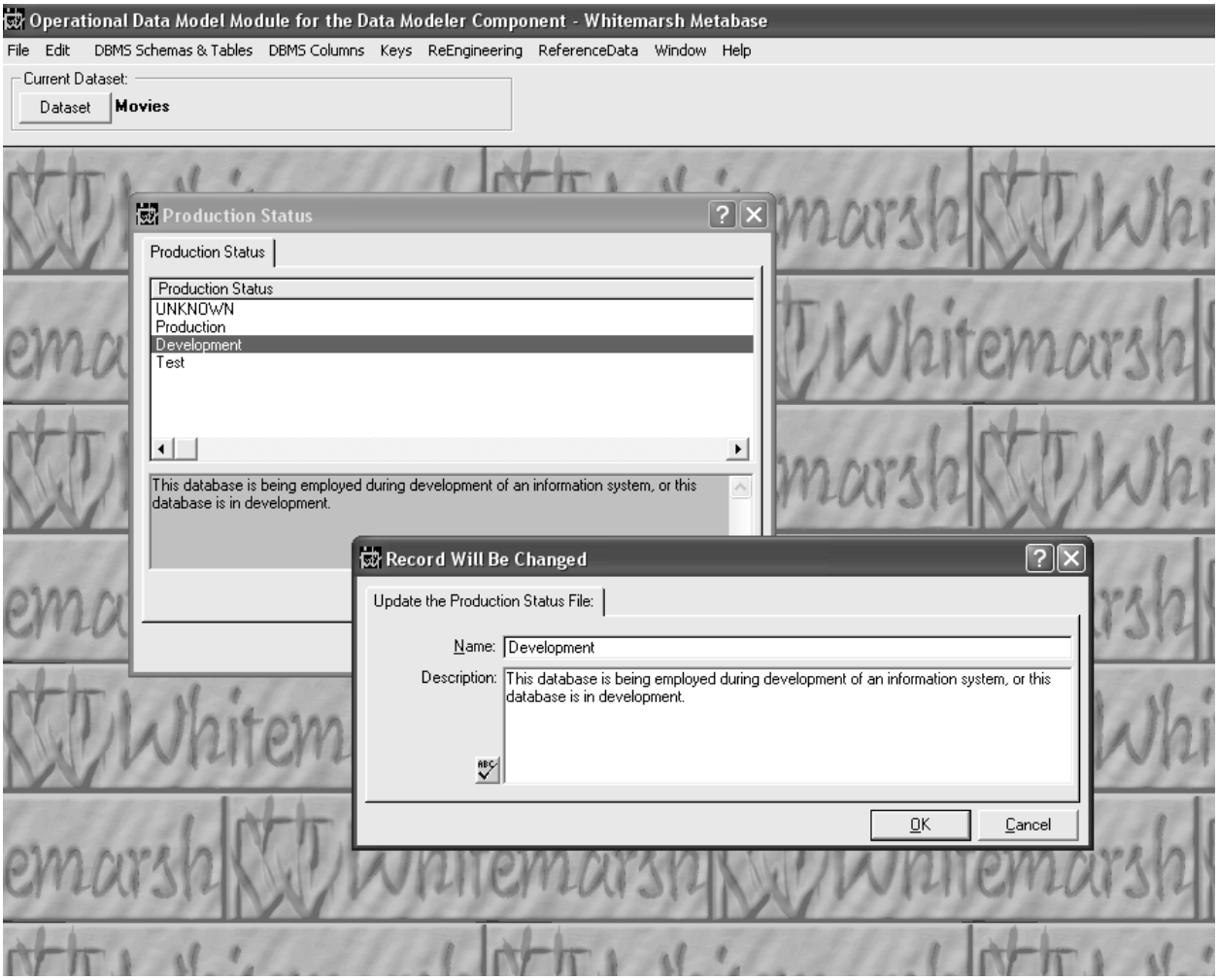

Figure 17. Database Production Status update screen.

6.2 Fact Data

The fact data consists of:

- DBMS schema Tables
- DBMS columns
- Keys
- Re-Engineering
- Reference Data

The reference data items are addressed in Section 6.1.

6.2.1 DBMS Schema DBMS tables

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

- DBMS schema
- DBMS table
- Import Tables
- Reverse Engineering
- SQL DDL

6.2.1.1 DBMS schema

DBMS schemas with respect to the operational data model are expressions of enterprise policy within a defined database. Figure 18 presents a list of DBMS schema. This list shows an "unknown" DBMS schema. This is necessary so that when one or more DBMS DBMS tables, DBMS DBMS columns and the like are promoted to be an operational model set of DBMS tables and DBMS columns, there is a valid foreign key identifier for the newly created DBMS table, albeit "unknown." Once these are created then the reverse engineering process of changing the DBMS schema for a DBMS table can be accomplished.

If a entirely new DBMS schema is to be entered, press Insert. A screen like Figure 19 is presented. The DBMS schema's name, abbreviations, and description can then be entered.

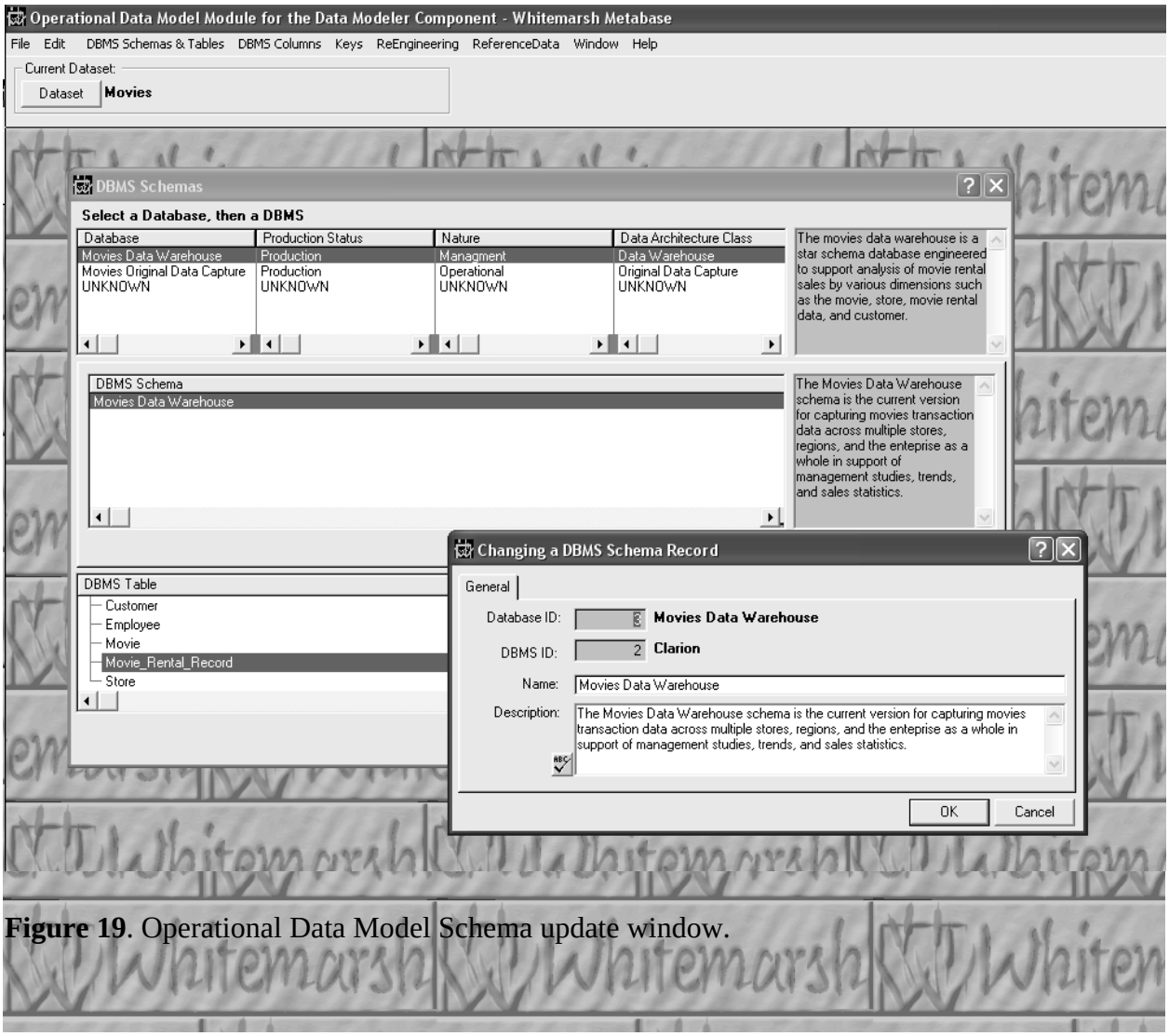

Figure 19. Operational Data Model Schema update window.

Figure 18. List of Operational Data Model Schemas.

6.2.1.2 DBMS tables

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

Figure 20 presents a list of DBMS tables. Within the DBMS, there are DBMS tables. Within the highlighted address DBMS table are a collection of DBMS columns.

This screen contains buttons for the DBMS schema, DBMS tables, and DBMS columns. For the DBMS Schema browse, the buttons are:

- Select Abbreviation Business Domain
- Auto Abbreviate all columns in all tables
- Replace blanks with underscores

The first button enables the selection of the business domain from within which the abbreviations are chosen. If no abbreviation exists, then "****" is substituted for the "not found" abbreviation. The second button then triggers the process of abbreviating all columns in all tables for that schema. The third button replaces any blanks with underscores in the event that a column's name remains that has blanks.

For the DBMS Table browse, the buttons are:

- Delete All Tables.... DBMS Schema
- Select Abbreviation Business Domain
- Auto Abbreviate all columns in all tables

The first button causes all tables, columns, and keys to be deleted for a Selected DBMS schema. If there are DBMS columns associated with a View then the association between those View Columns and the DBMS columns is deleted as well.

The second button enables the choice of the domain for creating abbreviations for the specifically selected table, and the third button triggers the process of making the abbreviations.

For the DBMS Column browse, the two buttons are:

- Select Abbreviation Business Domain
- Abbreviate selected column

To add a new DBMS table, highlight the containing DBMS schema and then press the Insert button. A screen like Figure 21 is displayed. A DBMS table can be created underneath the "unknown" DBMS schema and then re-assigned later. The DBMS table's name, abbreviations, and description can be added or changed. The DBMS columns associated with an DBMS table are addressed in Section 6.2.2.

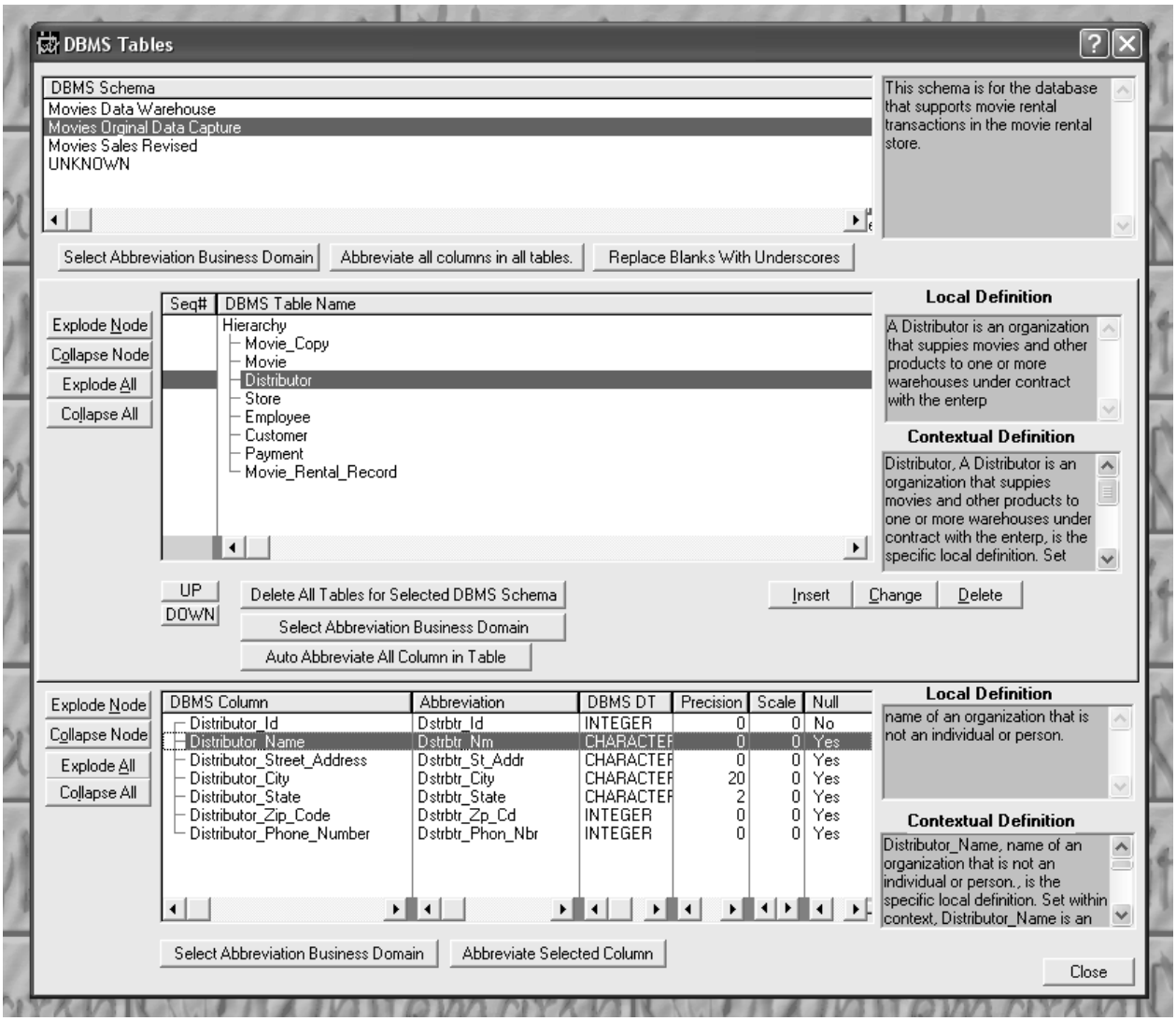

Figure 20. List of DBMS Tables.

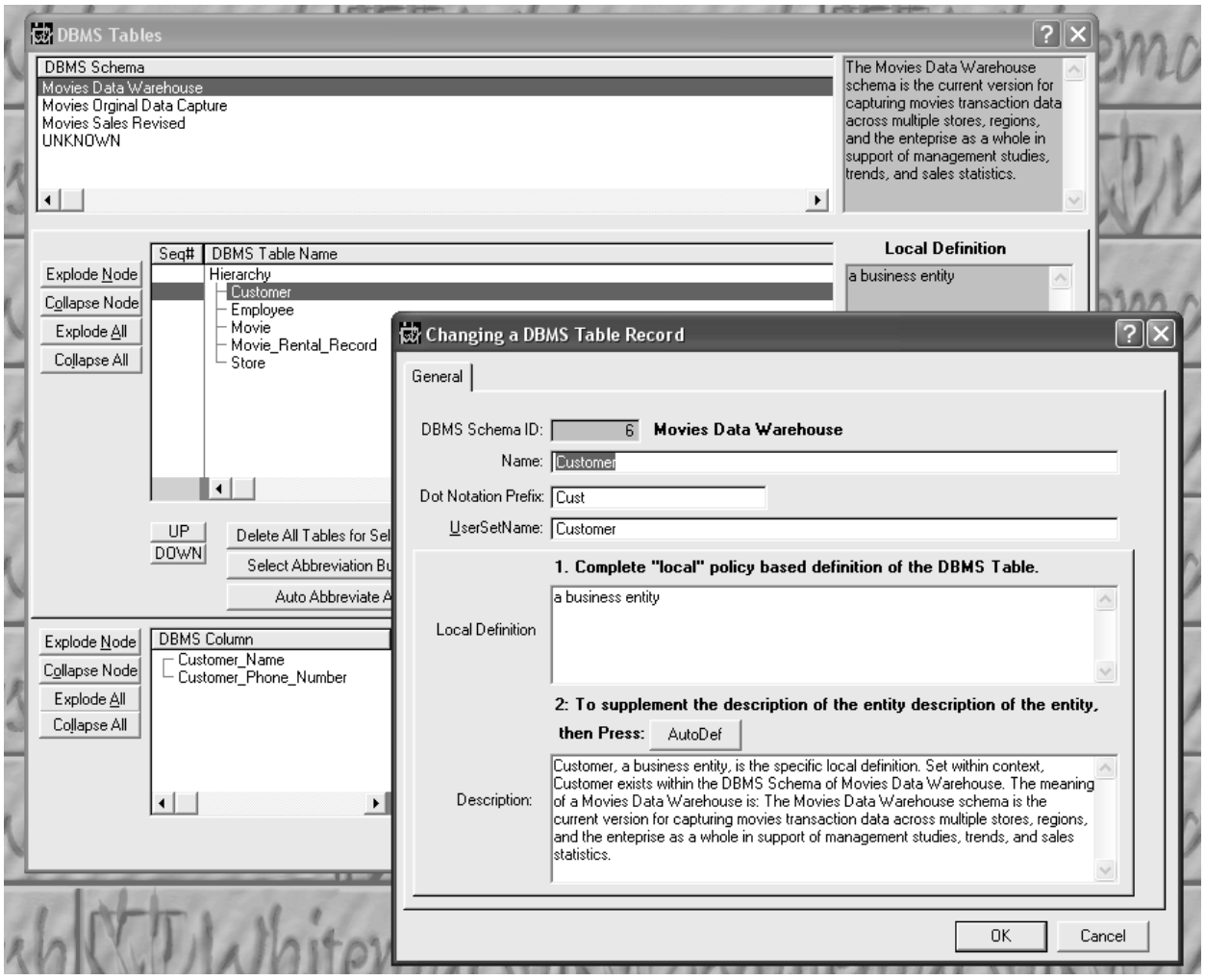

Figure 21. DBMS Table update screen.

Figure 21, the DBMS Table update screen is where you make the local definition and the contextual definition. Just make a simple phrase for the local definition. When the AutoDef button is pressed then all the contextual parts of the DBMS Table's definition are gathered and employed in a more comprehensive description of the DBMS Table.

6.2.1.3 Import Tables

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

- Import Schema Table Set
- Import Table Tree
- Import Single Table
- Import Columns from an Entity

6.2.1.3.1 Import Schema Table Set

The import subject entity set is depicted in Figure 22. First highlight the schema. The tables within that schema are then displayed. Then select a target DBMS Schema. If the target DBMS Schema is not present, then create it through the Insert button. Finally, press the Import button. What will be imported with be JUST the tables within that schema. In this case, just the movie and movie copy would be imported. All keys, primary, candidate, and foreign are imported as well.

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

6.2.1.3.2 Import Table Tree

The process of importing a table tree from an Implemented Data Model is depicted in Figure 23. First highlight the Schema, then the Table that is at the top of a hierarchy. To ensure that you have an apex table, highlight what seems to be the apex table and then press the button, Ancestor and Descendent tree. Then press the Display Tree. The bottom left window displays the built tree. The table that is black is the table that is the root table of the displayed tree. Descendent tables are in blue, ancestor tables are in red, and subtyped tables 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 tables, columns and keys across the tables of the tree by pressing the Display Data Model Tree button. The data model tree is presented in Figure 24.

When a target DBMS schema is able to be selected, highlight the target DBMS schema and the press the Import button. If the target DBMS schema is not present, then create it through the Insert button. Finally, press the Import Tree button. Built will be all the operational data model DBMS tables, DBMS columns, and DBMS relationships. All the mapping between the newly created operational data model and the implemented data model will be created as well.

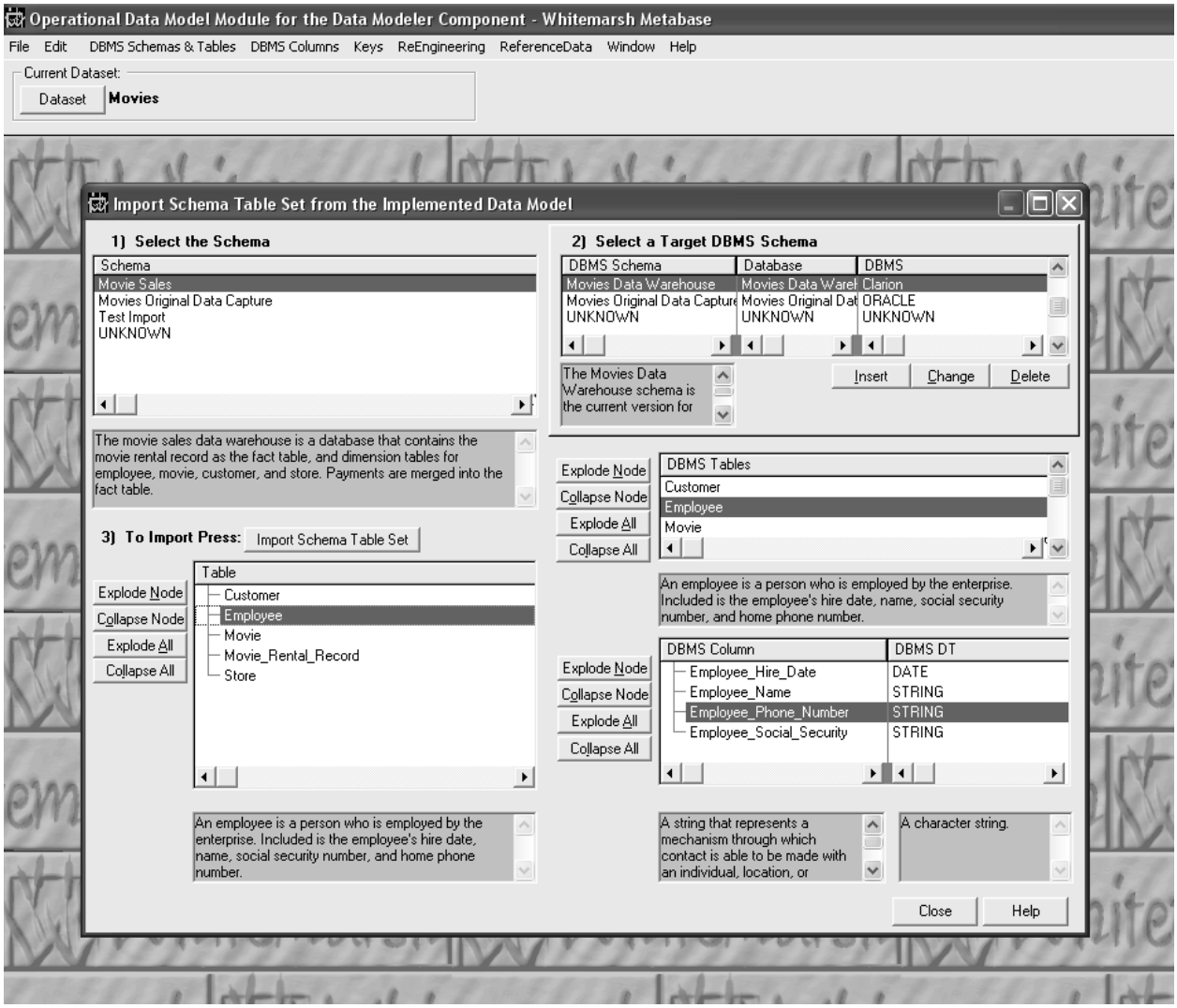

Figure 22. Importing a Schema Table set into an Operational Data Model Schema.

The only tables and columns that will be imported are those in a direct ancestor-descendent relationship of the displayed tree root node. Skipped are foreign keys and columns of foreign keys that are not in a direct line with the root table.

If multiple specified data model trees are imported into the operational 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 DBMS table from one operational data model tree and another of the newly created table of a different data model tree. With this technique you can create an operational data model set of DBMS tables from more than one Implemented Data Model Schema's set of tables.

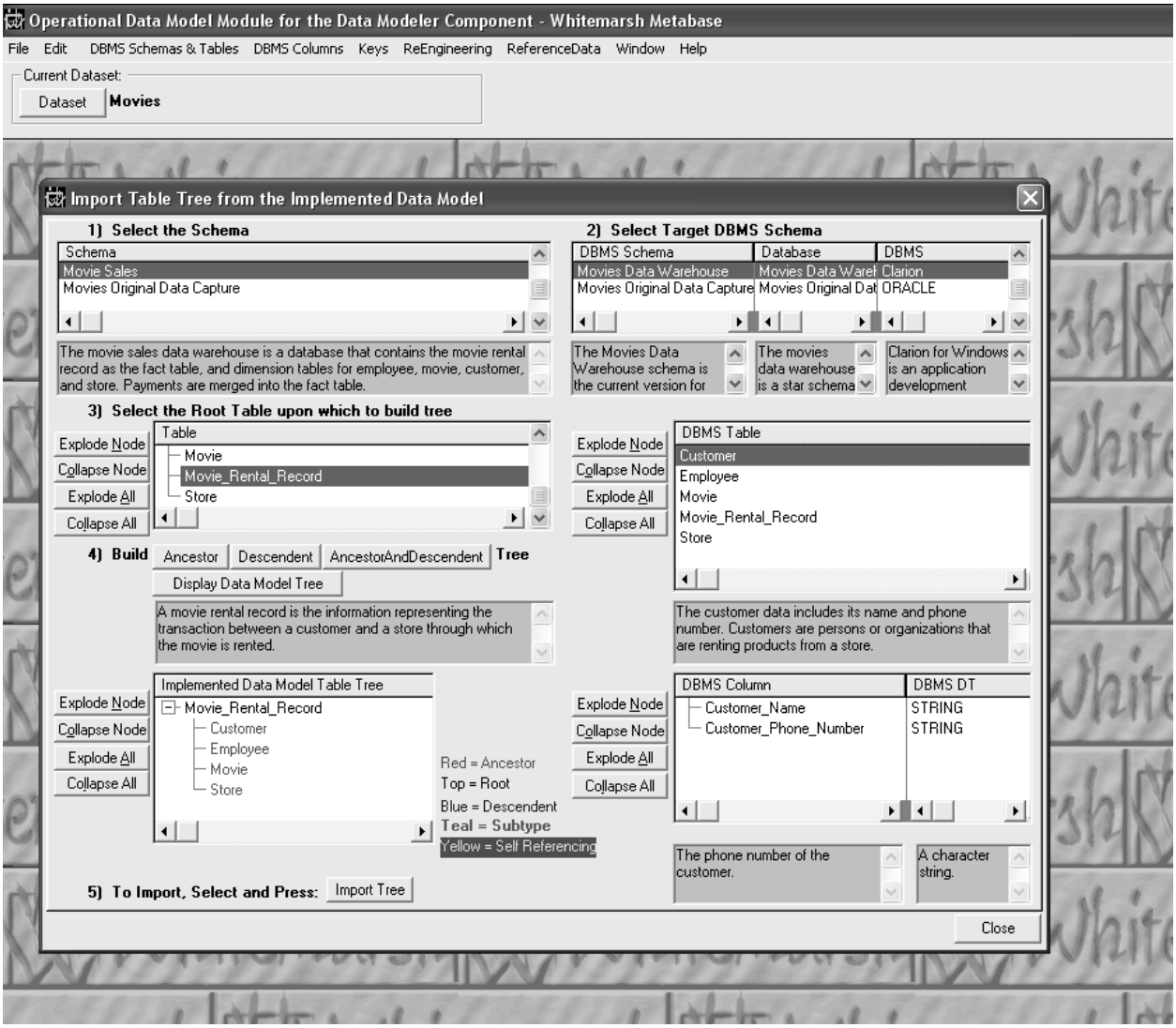

Figure 23. Importing a Data Model Tree.

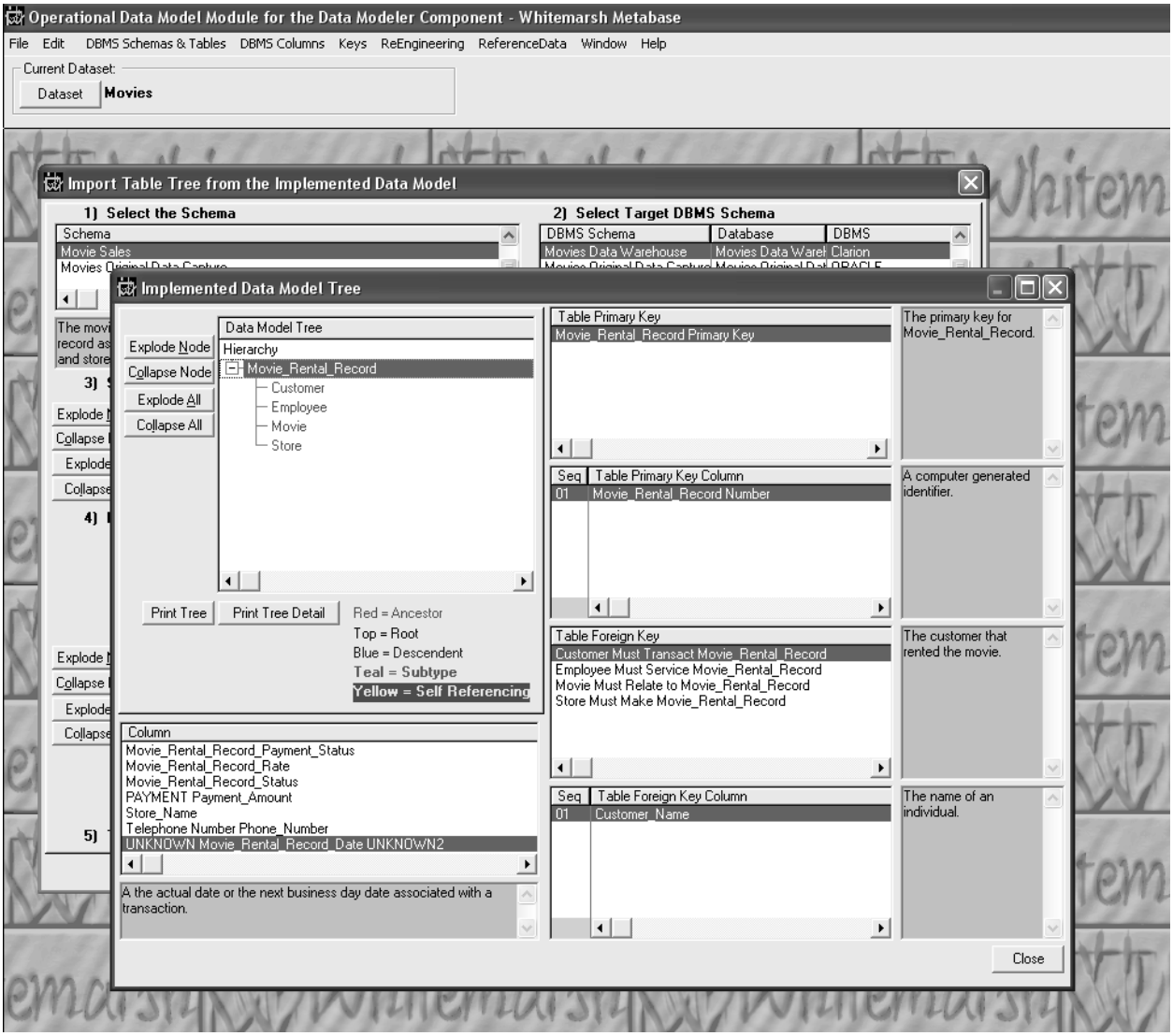

Figure 24. Data Model Tree.

6.2.1.3.3 Import Single Table

The screen for importing a single table from an Implemented Data Model is depicted in Figure 25. First highlight the schema, then the table that is at the top of a hierarchy. Now, on the right side of the window, highlight the target DBMS schema. If none are shown then create one.

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

The table that is imported into the operational 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 DBMS table and existing DBMS tables

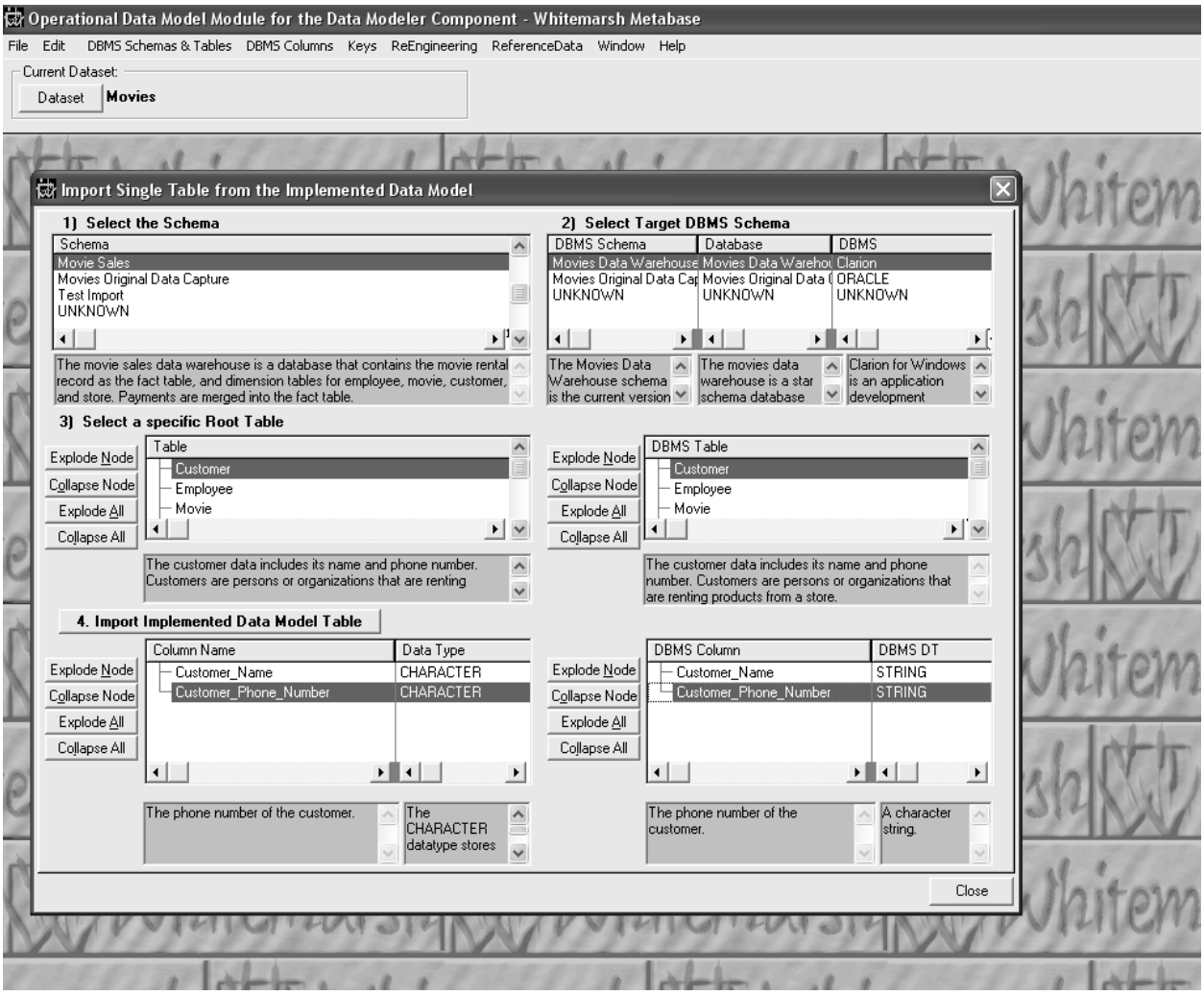

Figure 25. Importing a single Table.

6.2.1.3.4 Import Columns from a Table

The screen for importing one or more columns from an Implemented Data Model is depicted in Figure 26. First highlight the schema, then the table. Then tag one or more columns. On the right side, select the DBMS schema and then tag ONE DBMS table. It is the DBMS table into which the columns will be imported. All the mapping between the newly created DBMS columns and the implemented data model table columns will be created as well.

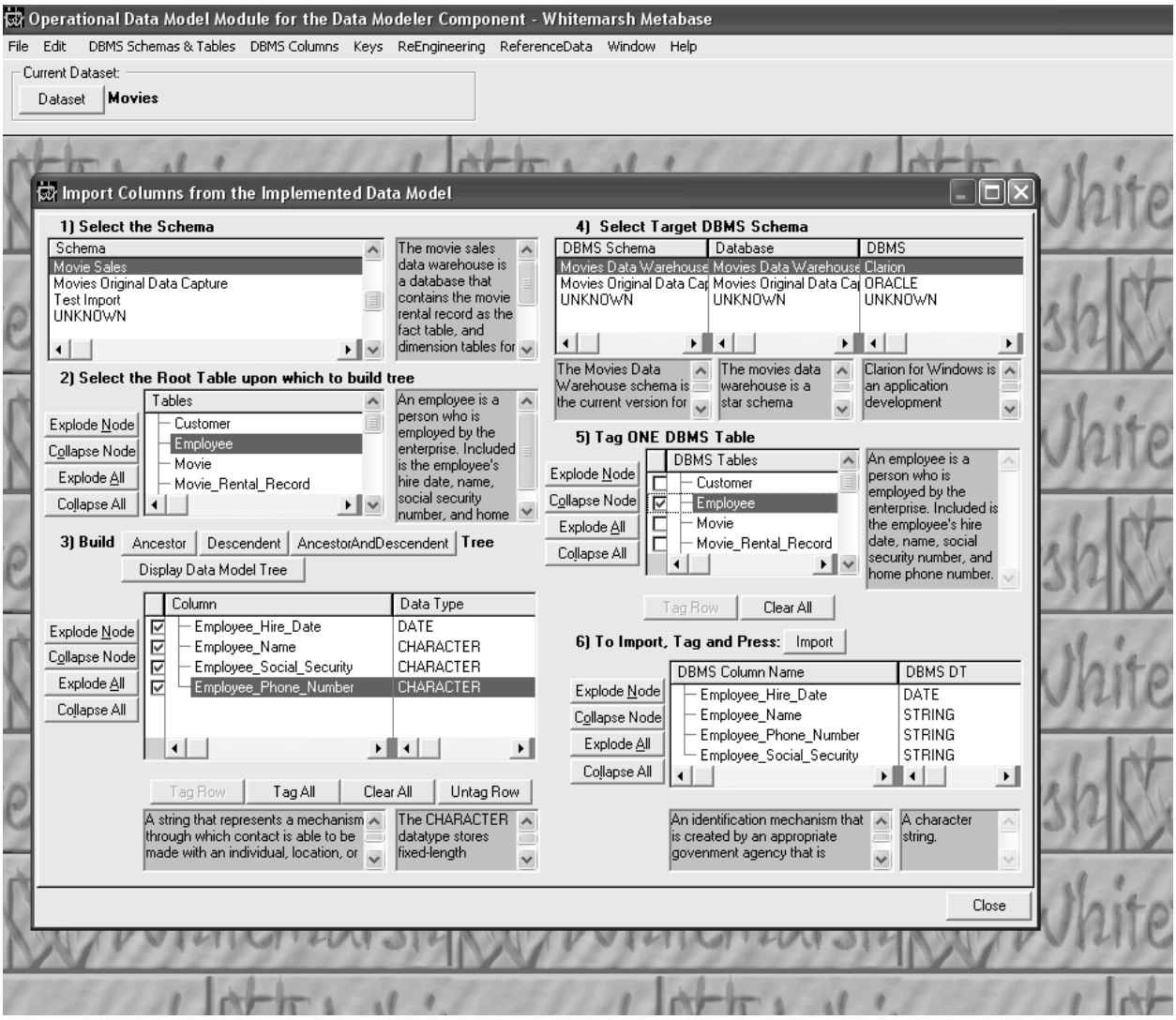

Figure 26. Importing a single column.

6.2.2 DBMS columns

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

- DBMS columns
- DBMS column value domains
- Data Hierarchies

6.2.2.1 DBMS Columns

Creation or maintenance of DBMS columns starts with a list of DBMS columns. Figure 27 shows the list of DBMS columns for highlighted subject and DBMS table. If an DBMS column is to be changed then press the Change button. Figure 28 is then presented. New DBMS columns can also be created through Figure 27 by pressing the Insert button. An operational data model DBMS column is not the use of a data element within a DBMS table as an Implemented Data Model column is within a table. Rather it's entirely new and independent. If you want an Implemented Data Model column to be the template then use the Import Column process in an earlier section. Alternatively you can create the new DBMS column, and then through the ReEngineering feature, assign the Operational Data Model DBMS Table DBMS column to an Implemented Data Model Table Column. Either way works. Finally, you can leave the newly created DBMS column independent.

If the DBMS column is not within a foreign key then all the meta DBMS columns of the DBMS column can be changed. If, however, the DBMS column is part of foreign key Figure 29 is displayed with the message that the only meta DBMS column that can be changed is its description.

Figure 28 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.

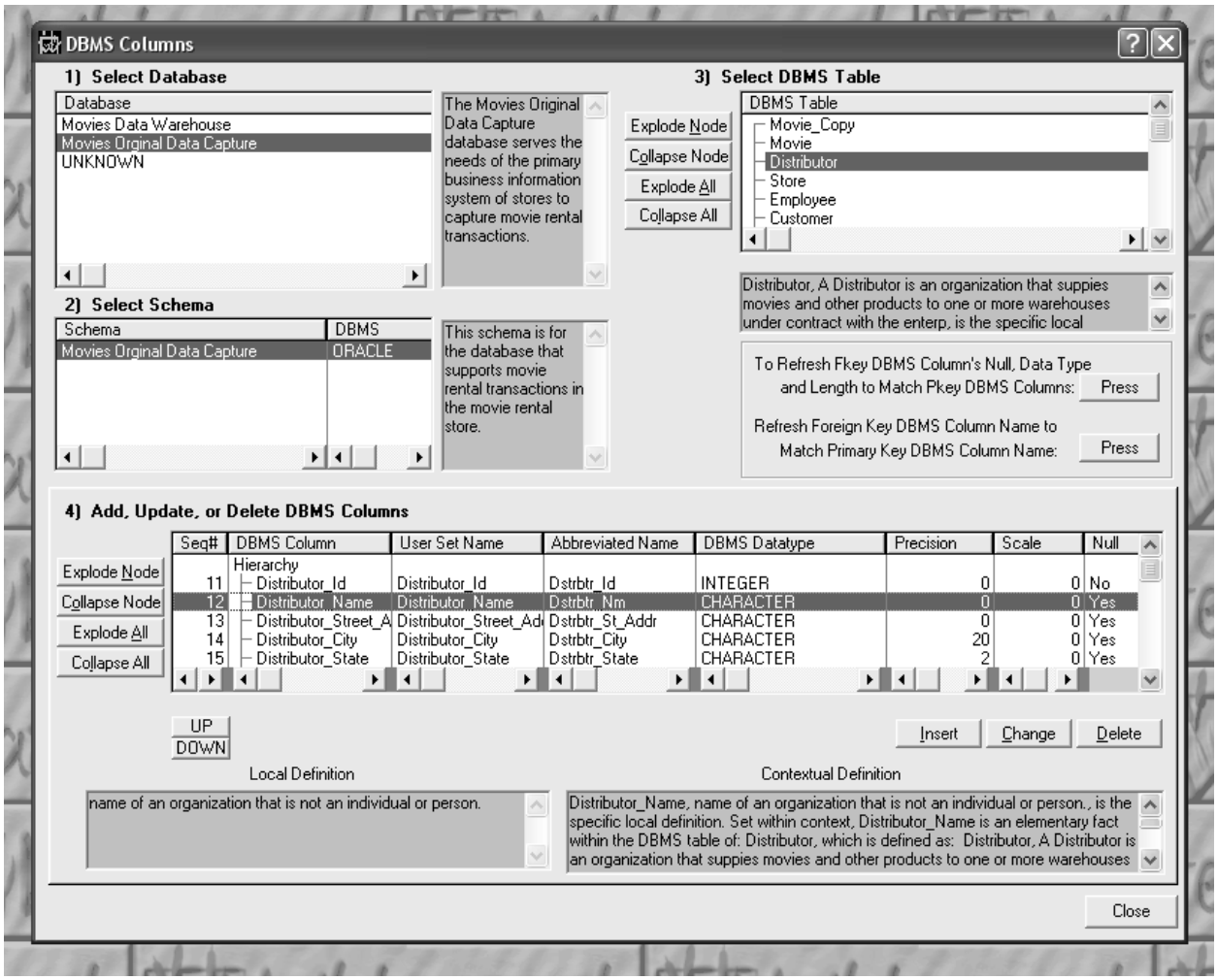

Figure 27. List of DBMS Columns.

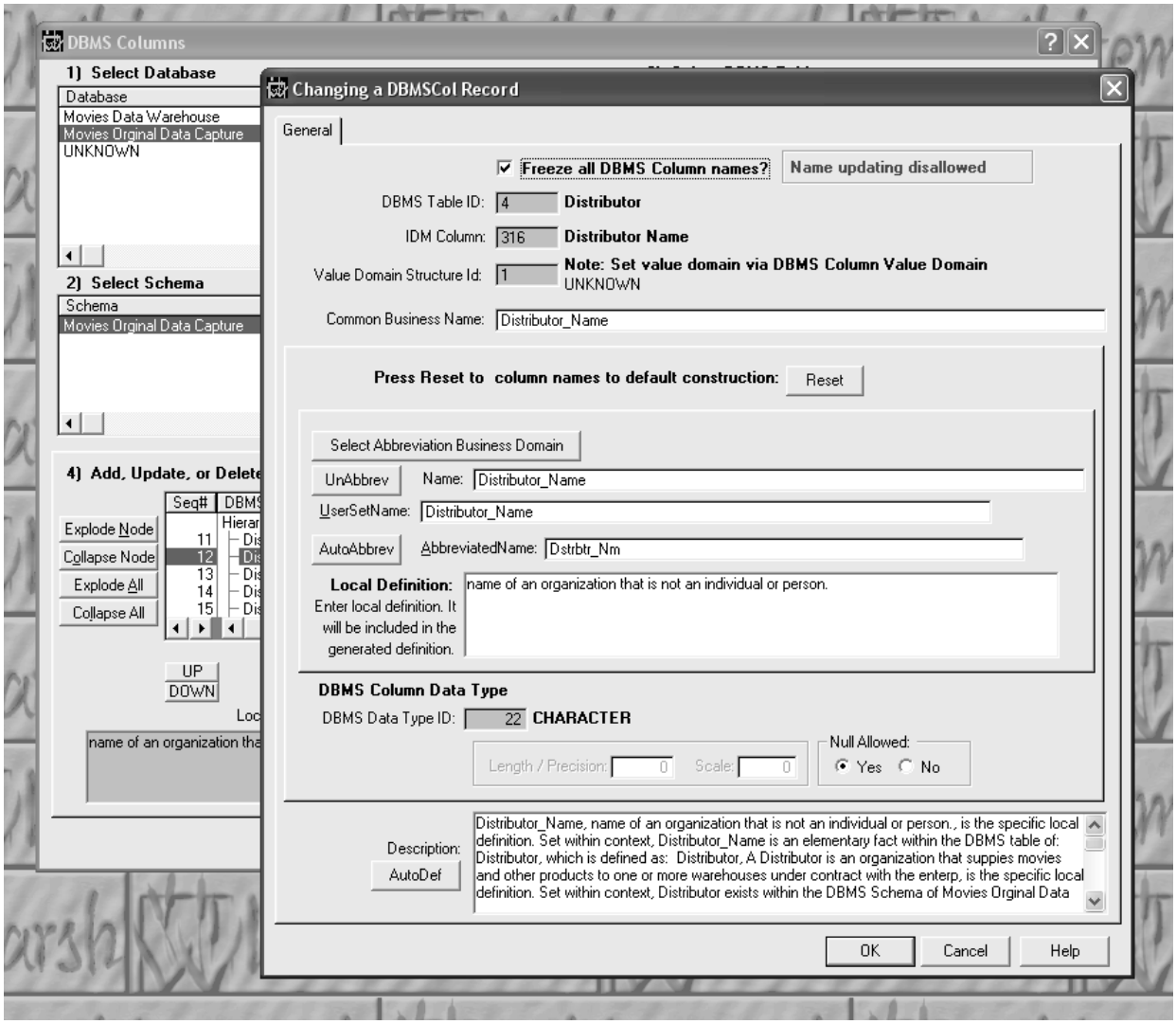

Figure 28. DBMS Column update screen.

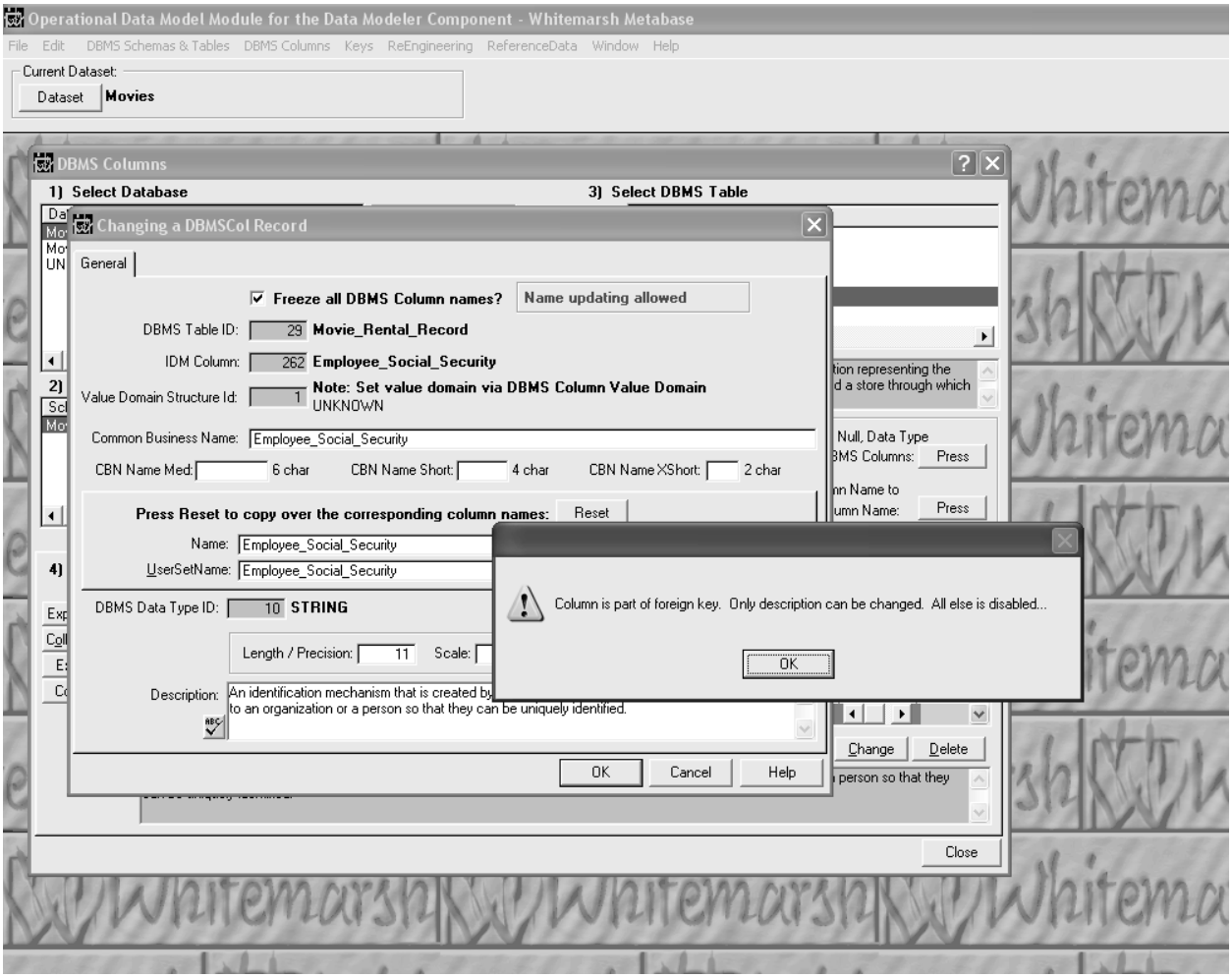

Figure 29. Error message received when updating a Foreign Key Column.

6.2.2.2 Maintain DBMS Column Value Domains

DBMS Column value domains exist within the context of column value domains which in turn exist within the context of data element value domains, data element concept domains and then value domains. Figure 30 presents the screen for viewing and then maintaining DBMS column value domains. To see the specific value domain for a DBMS column, highlight the DBMS schema, DBMS table, and then DBMS column. The related column, attribute and data element for the DBMS column is then displayed along with any value domains that are allowed to be assigned. If there already are DBMS column value domains associated with the DBMS column's column, 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 DBMS column's column and data element the assignment is accepted. If the selected value domain is a superset of an already assigned value domain the assignment is rejected. If the DBMS column does not have an assigned column the assignment is rejected. While this seems counterintuitive, it is because of the lack of context that an assignment cannot then be validated. Value domains are within the context of Conceptual Value Domains. Data Element Concepts are within the context of Conceptual Value Domains (as well as Concepts). A Data Element is the contextual deployment of a Data Element Concept and a Value Domain. Without these contexts, the assignment cannot be validated; hence the assignment is rejected.

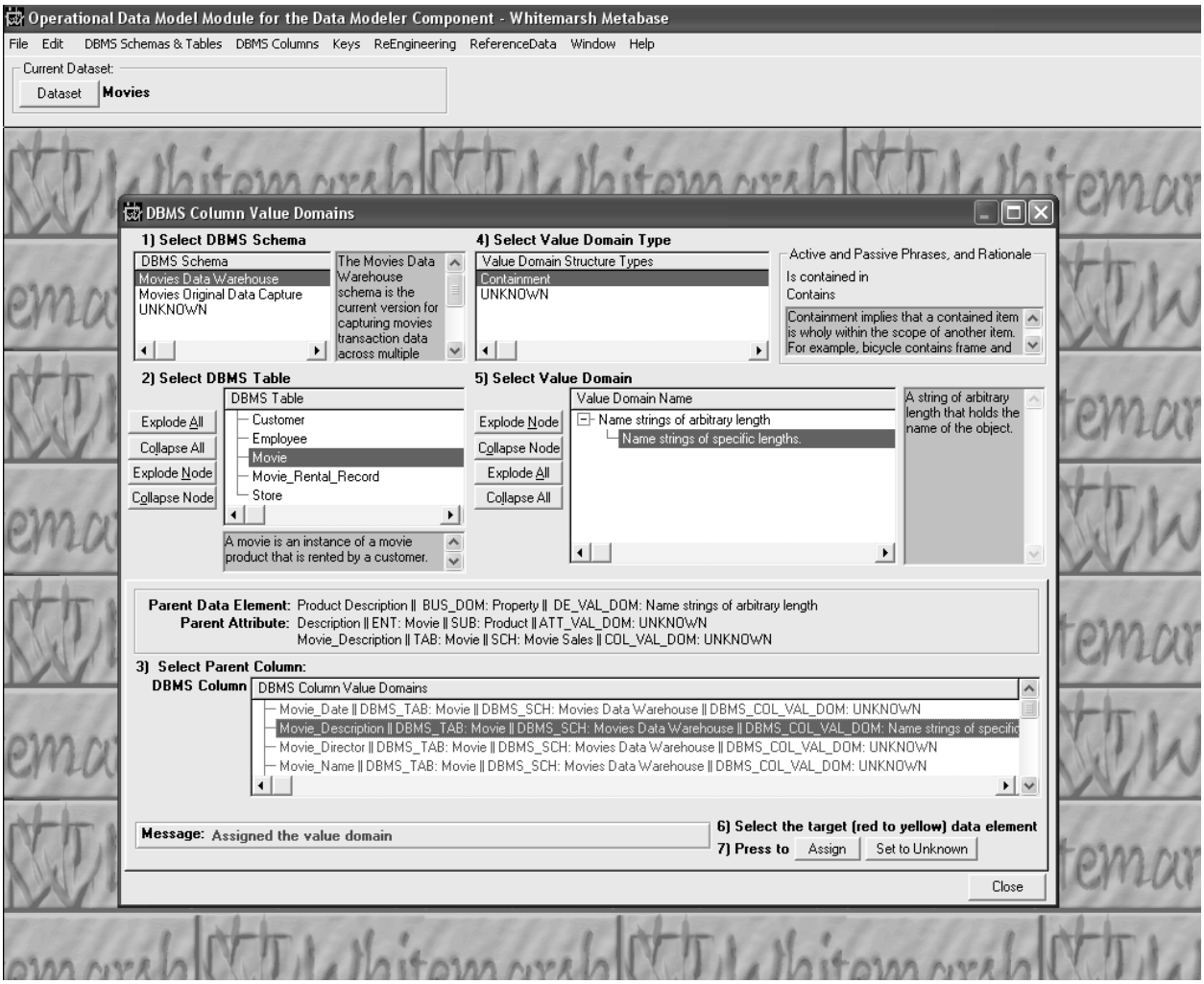

Figure 30. Assigning a value domain to a DBMS Column.

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 operational data model, any highlighted DBMS column has only one parent column, attribute, data element and data element domain. Hence, in Figure 31, they are shown above the traditional browse.

As a browse progresses from one DBMS 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 DBMS columns.

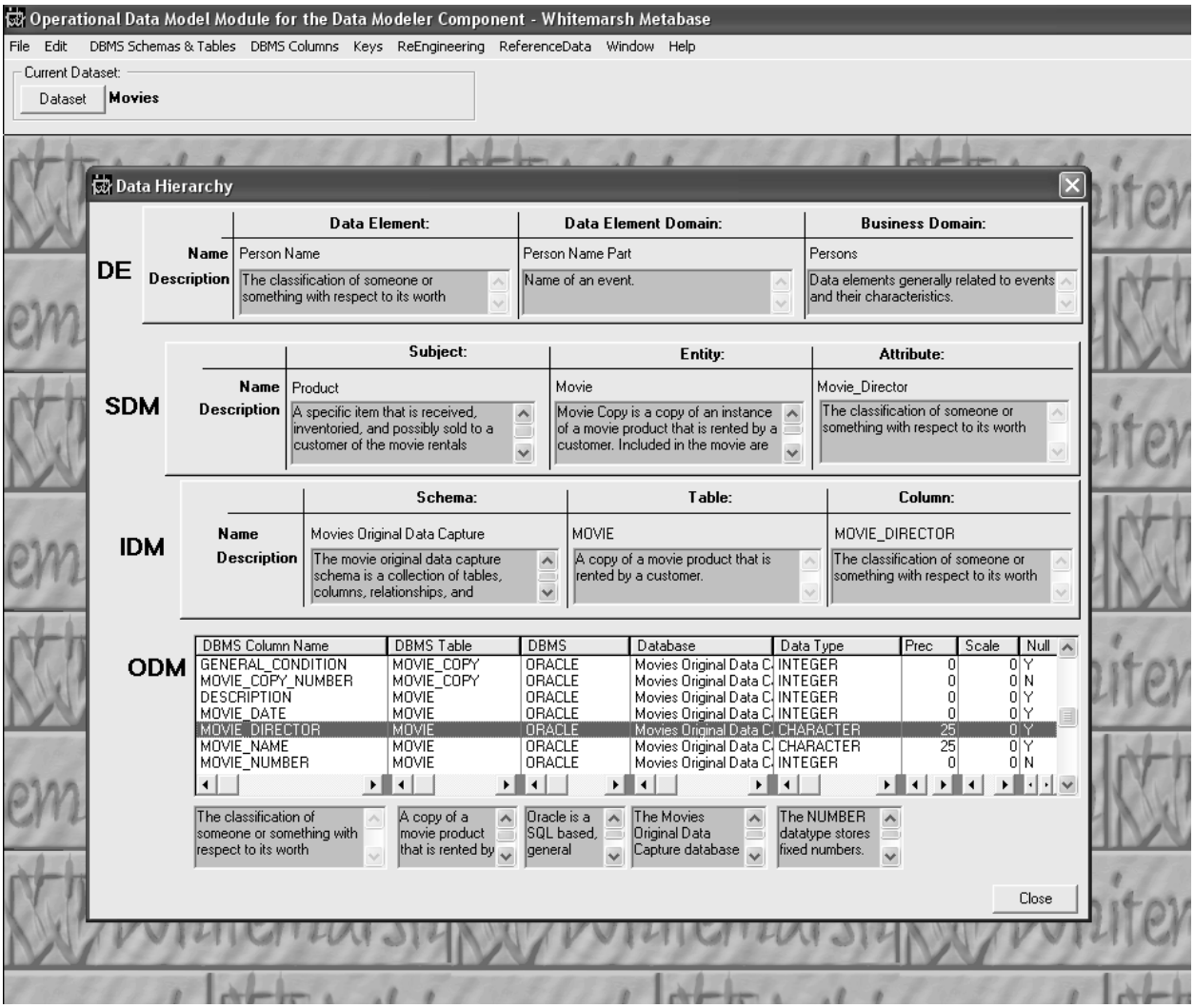

Figure 31. DBMS Column based data hierarchies.

6.2.3 Keys

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

- Primary
- Foreign
- Candidate
- Secondary

6.2.3.1 Primary

Included in the definition of a complete primary key is:

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

6.2.3.1.1 Primary Key Definition

A primary key of an DBMS table is a set of one or more DBMS columns that represent values that when employed result in only one selected row. Figure 32 shows the current set of primary keys. There can, of course, only be one primary key for each DBMS table. To see the DBMS columns assigned to a particular primary key, highlight the appropriate subject, and then DBMS table. To add a primary key if there is none for an DBMS 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 DBMS 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 DBMS column is employed as a DBMS column in the operational data model the operation fails.

If the Insert or change action succeeds, the Figure 33 is displayed. On an Insert, the name is automatically constructed as the concatenation of the DBMS table name and the string, "Primary Key." The name can be changed. In addition to the name a description can be added or changed.

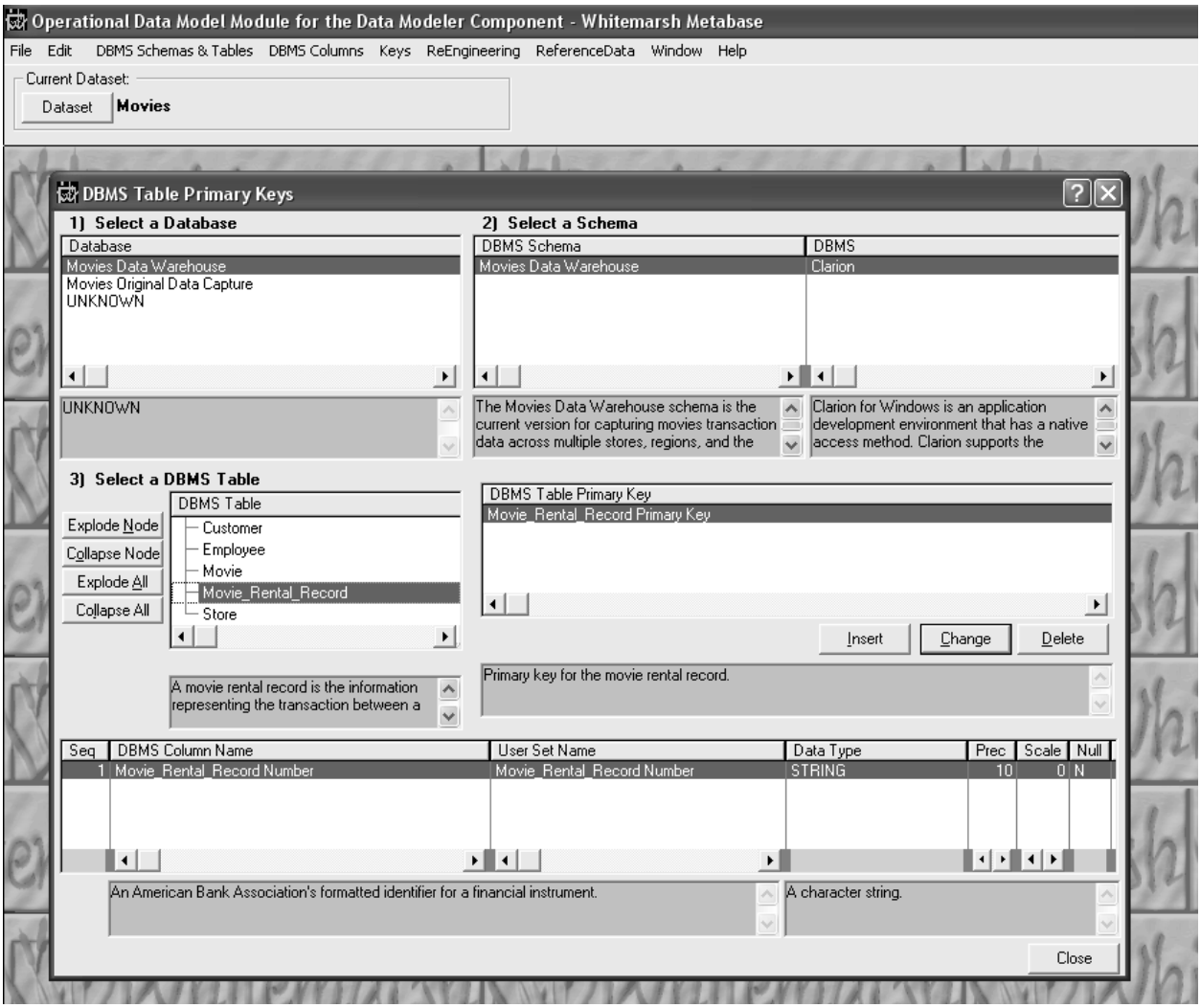

Figure 32. List of Primary Keys.

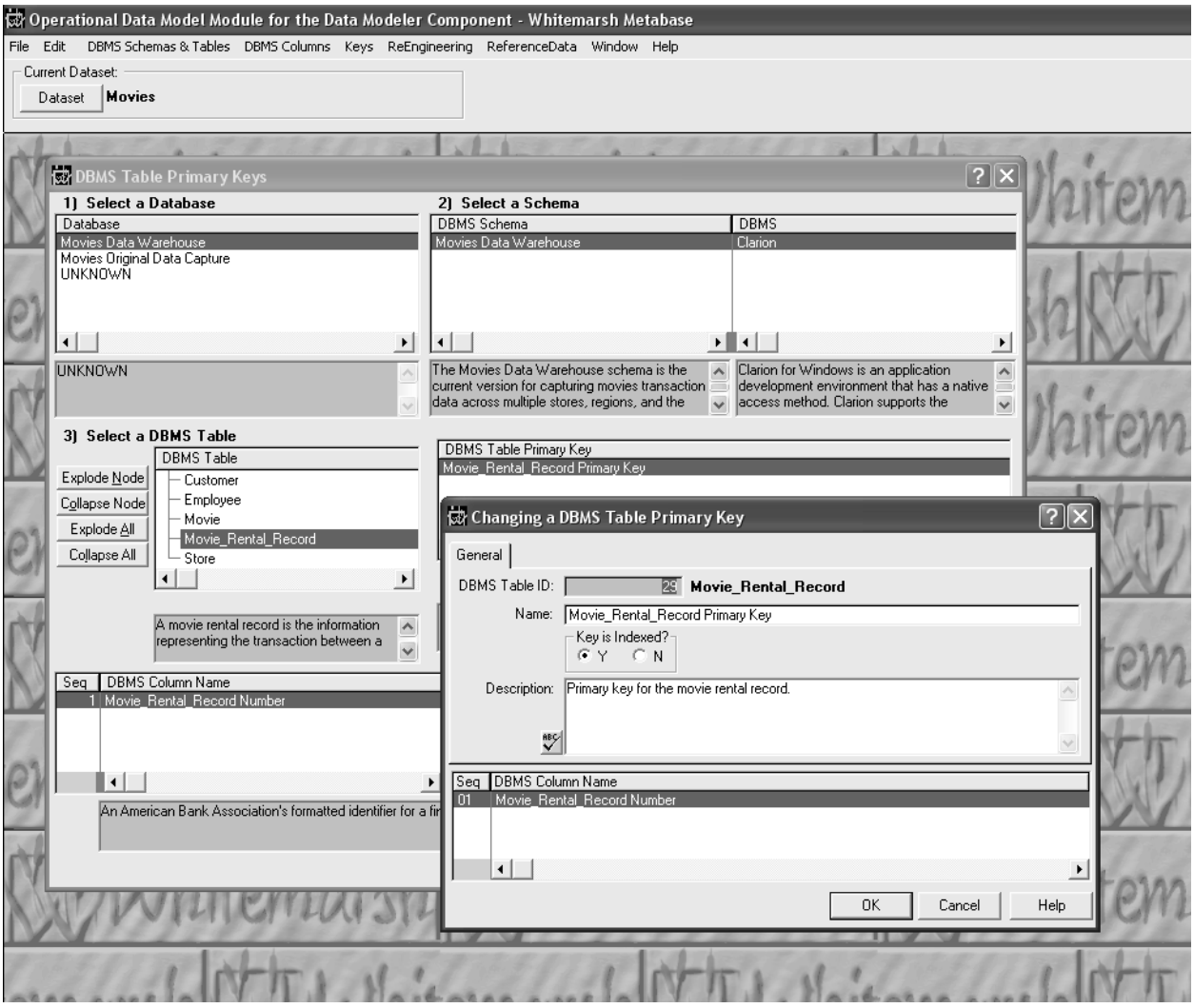

Figure 33. Primary Key update screen.

6.2.3.1.2 Allocation of DBMS columns to the Primary Key

A primary key is a collection of DBMS columns. The order of the DBMS columns within the primary key imply the order of the values used to select rows of data given that the DBMS table is in fact a DBMS table. Figure 34 presents a list of the primary keys and the current set of DBMS columns assigned to each. To assign an DBMS column to a primary key, highlight the subject then the DBMS table, then tag the appropriate DBMS table primary key. Then, from the DBMS table's shown DBMS columns, tag the one or more DBMS columns that comprise the primary key. Finally, press the Build button. The DBMS columns that then comprise the primary key are shown in the bottom window. Once the DBMS columns are assigned, the Up|Down buttons can be used to change the sequence of the DBMS columns within the primary key.

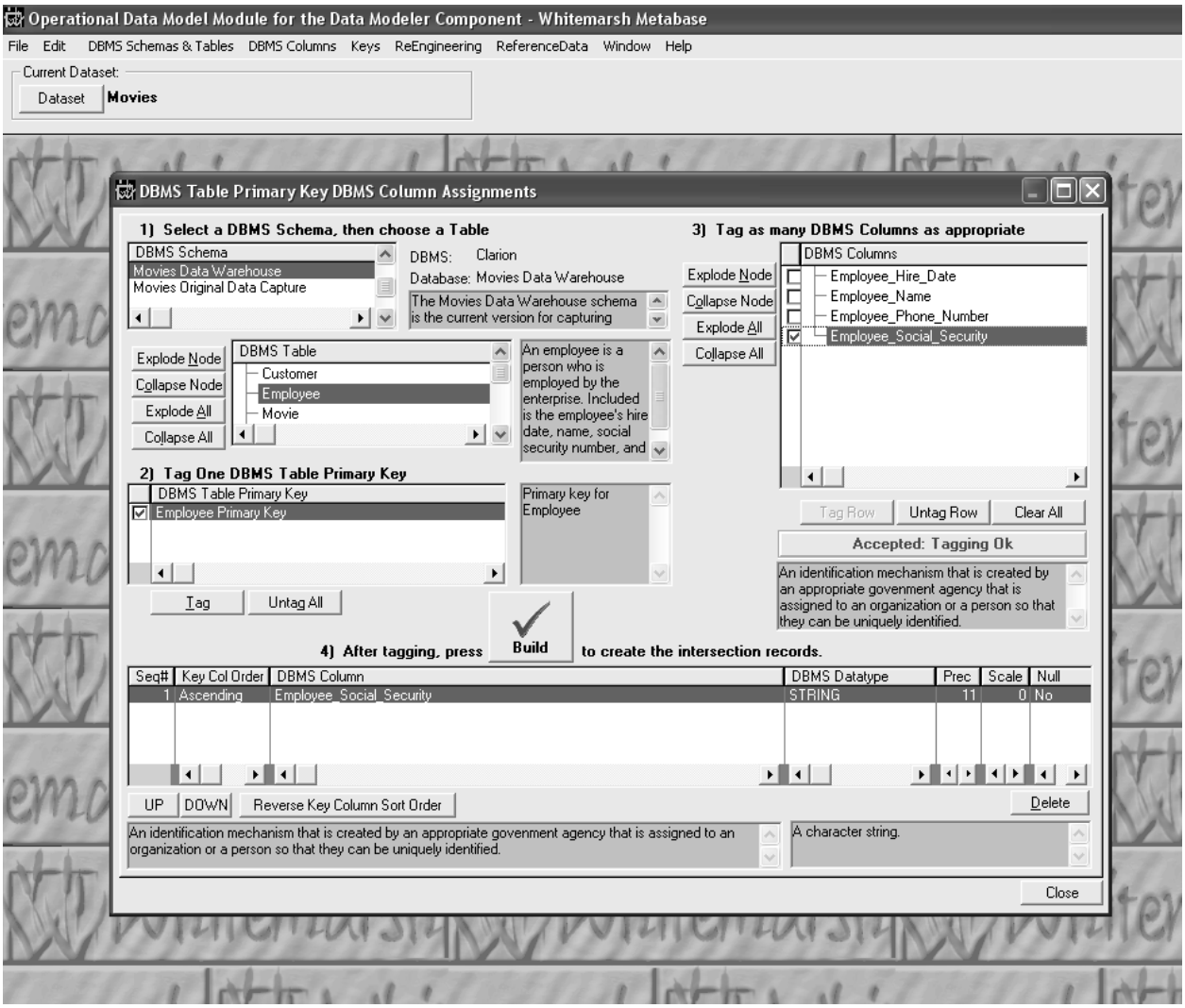

Figure 34. Assigning DBMS Columns to a Primary Key.

6.2.3.2 Foreign

Foreign keys are primary keys from another DBMS 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 DBMS table, and the target DBMS table. Once these two pieces of information are provided then the rest is automatic. That is, the foreign key's name, and DBMS columns that comprise it.

Figure 35 presents the current list of foreign keys. To enter a new foreign key, highlight the subject, and then the DBMS table that is to contain the foreign key. The current list of foreign keys within that DBMS 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. The foreign key update screen is presented in Figure 36.

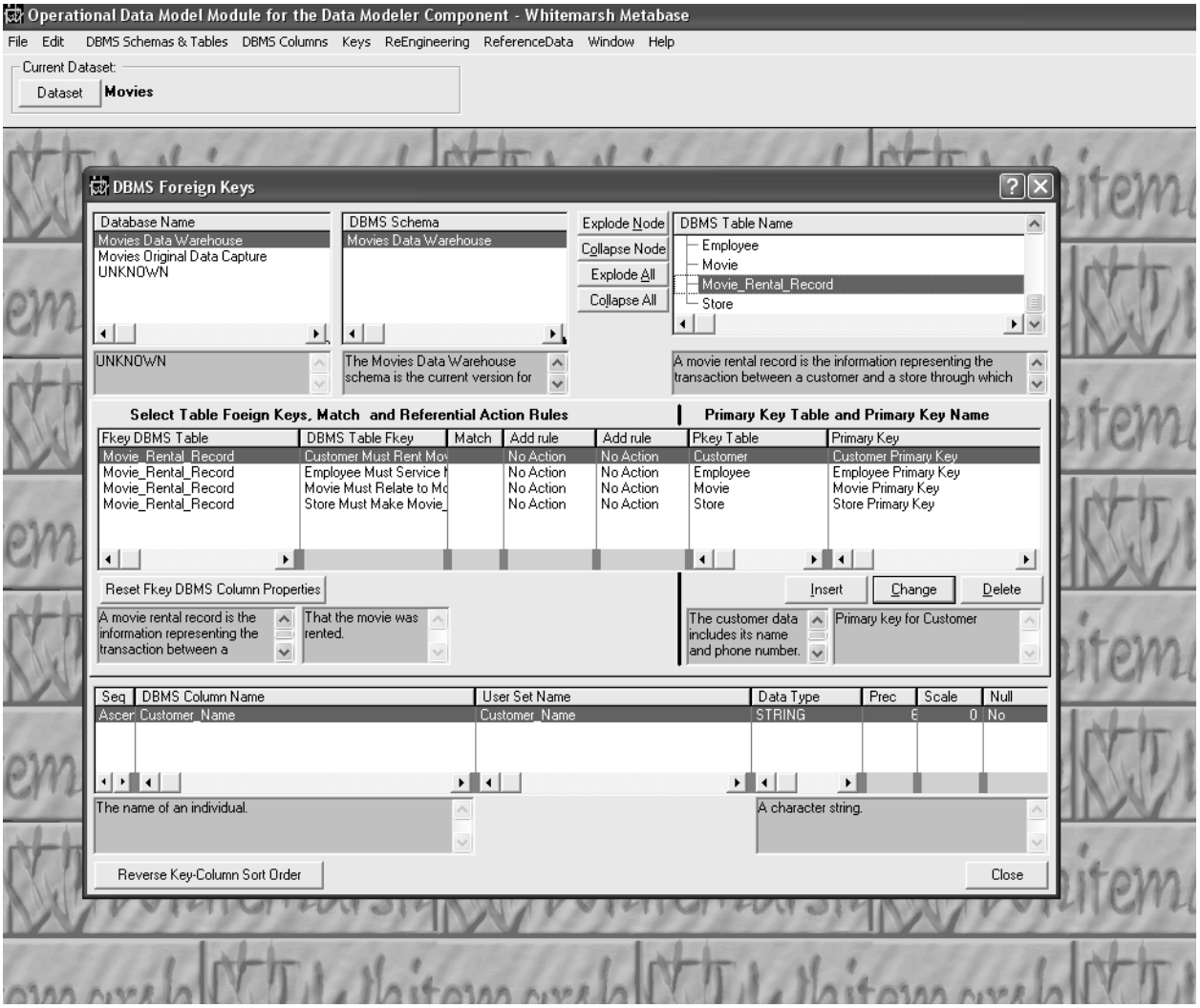

Figure 35. Adding or updating a Foreign Key.

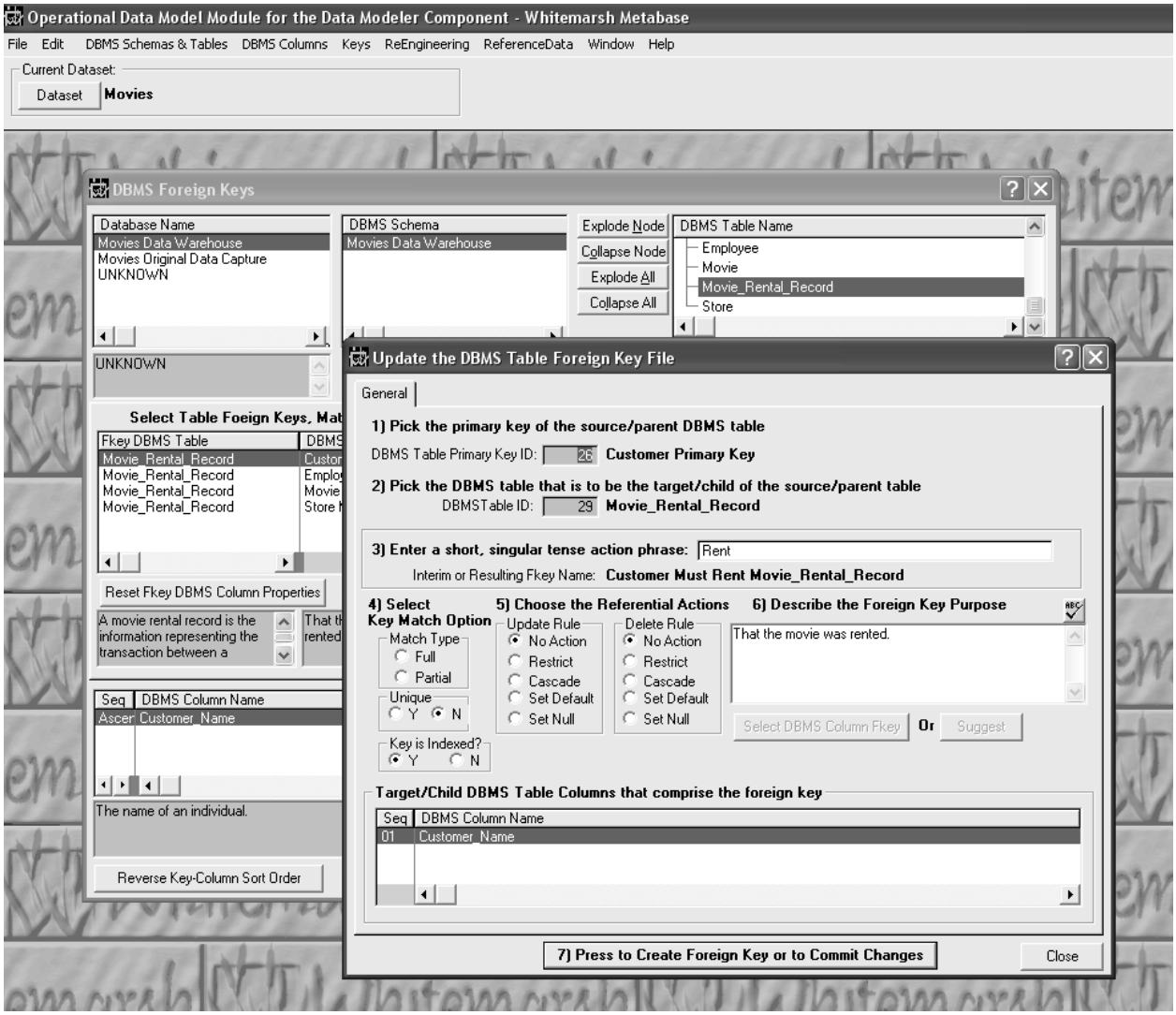

Figure 36. Foreign Key addition or update screen.

Figure 36 presents the form for entering a new foreign key. The first step is the enter the value for DBMS table Pkey Id. If valued with a zero and then the Tab key is pressed, a list like the one in Figure 37 is presented. Highlight the appropriate subject and then the appropriate primary key, which also shows the source DBMS table name. Then press Select.

If the DBMS table that was to contain the foreign key had been previously highlighted before the Insert button was pressed then the DBMS 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 38 shows that list. Highlight the appropriate DBMS table and press Select.

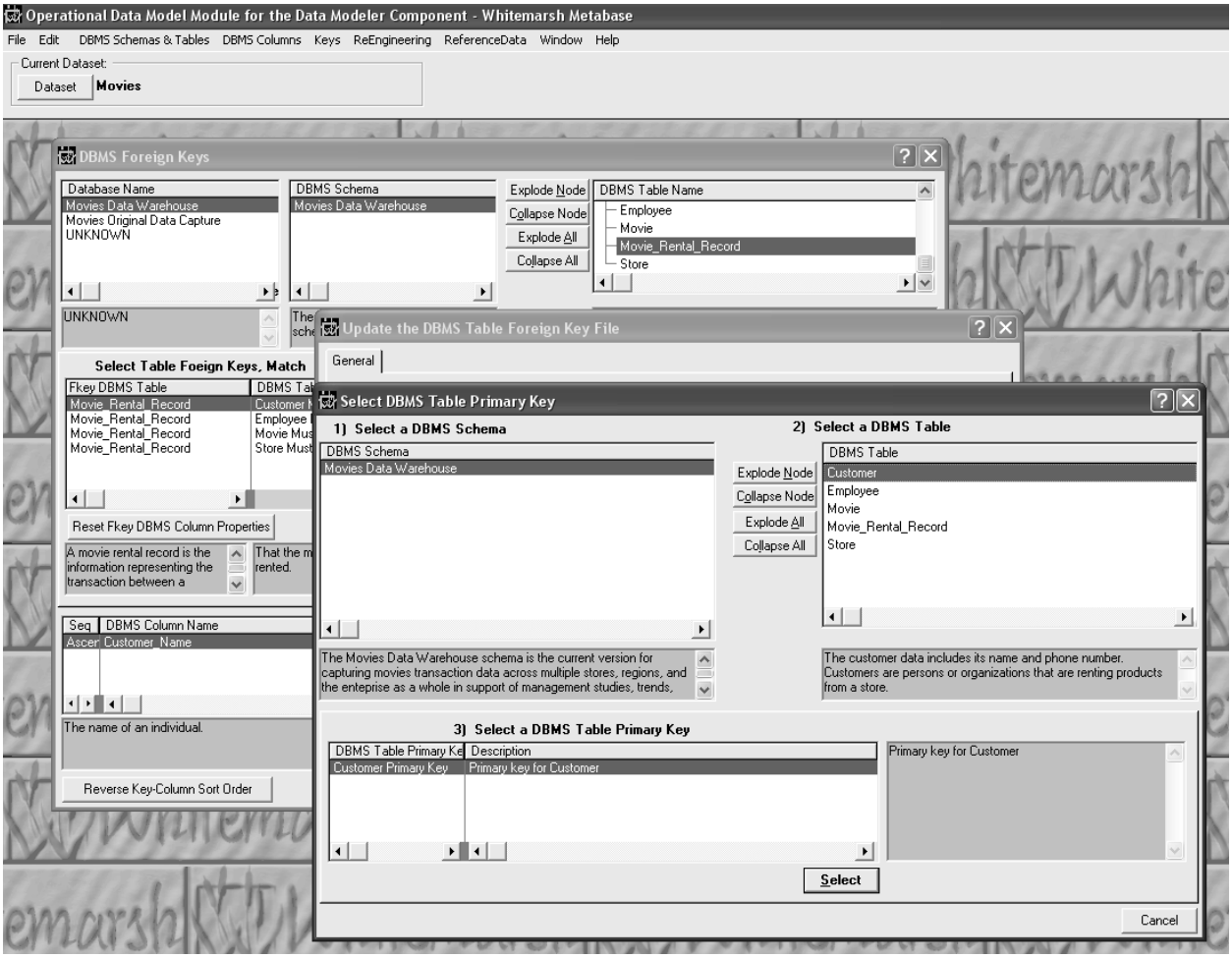

Figure 37. List of DBMS Tables for source for a Primary Key.

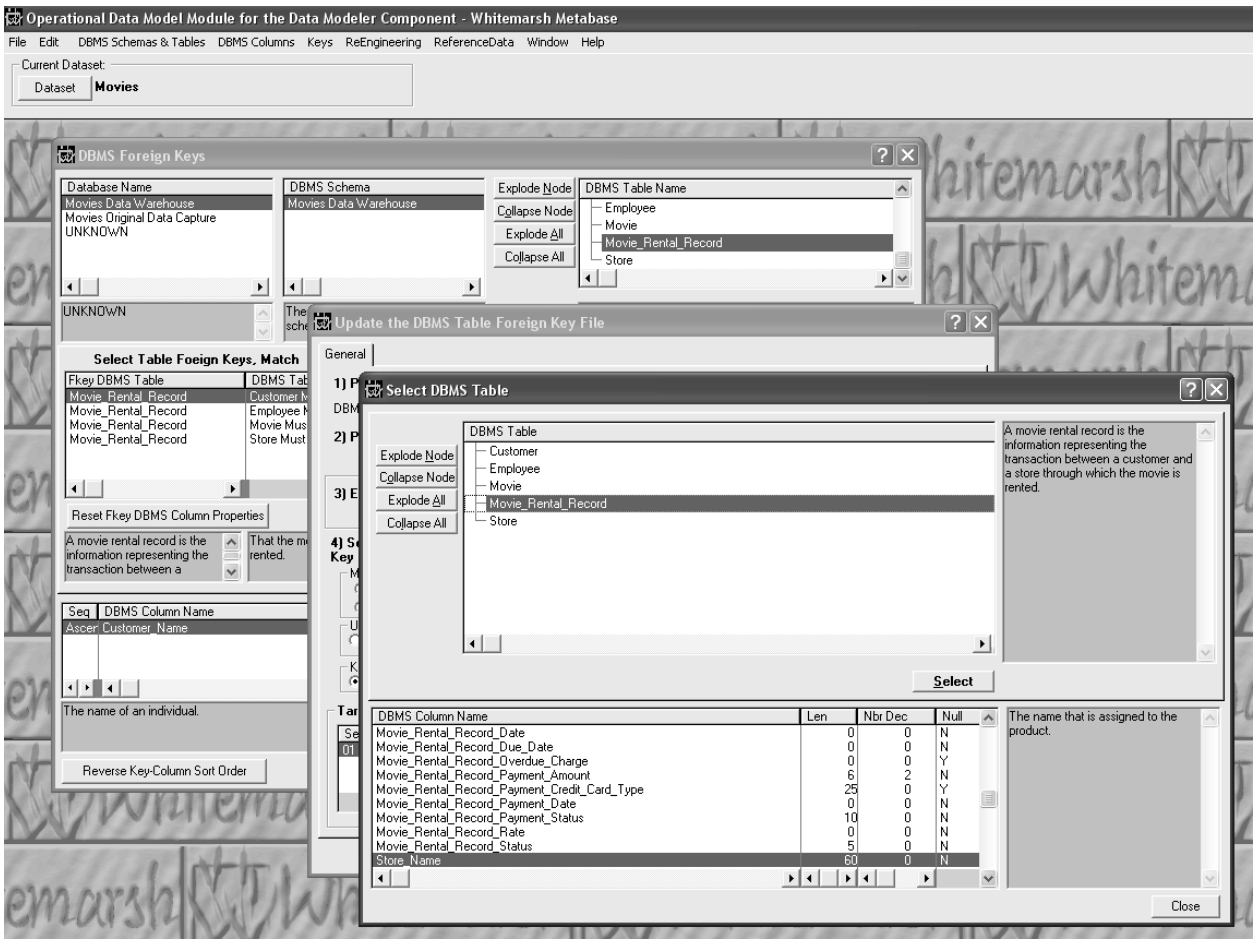

Figure 38. List of Tables for Target of Foreign Key.

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 39 illustrates the entry of the singular present tense action phrase. Immediately below the phrase, Have, is the constructed foreign key name, Customer must have order header. Customer is the source DBMS table. Order header is the target DBMS table, and "have" 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.

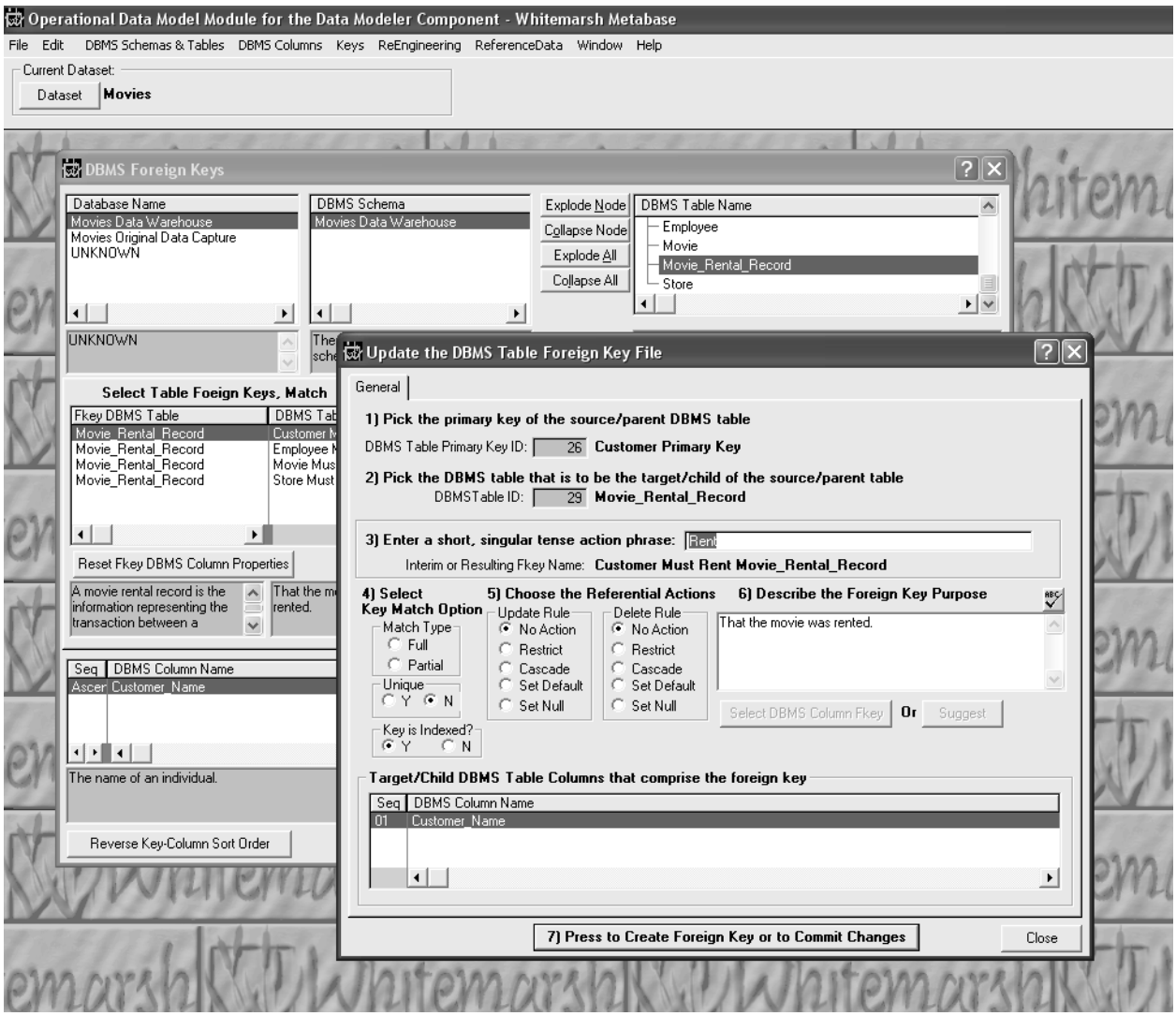

Figure 39. Adding the Foreign Key present tense action phrase.

6.2.3.3 Candidate

Included in the definition of a complete candidate key is:

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

6.2.3.3.1 Candidate Key Definition

A candidate key of an DBMS table is a set of one or more DBMS columns that represent values that when employed result in only one selected row. Figure 40 shows the current set of candidate

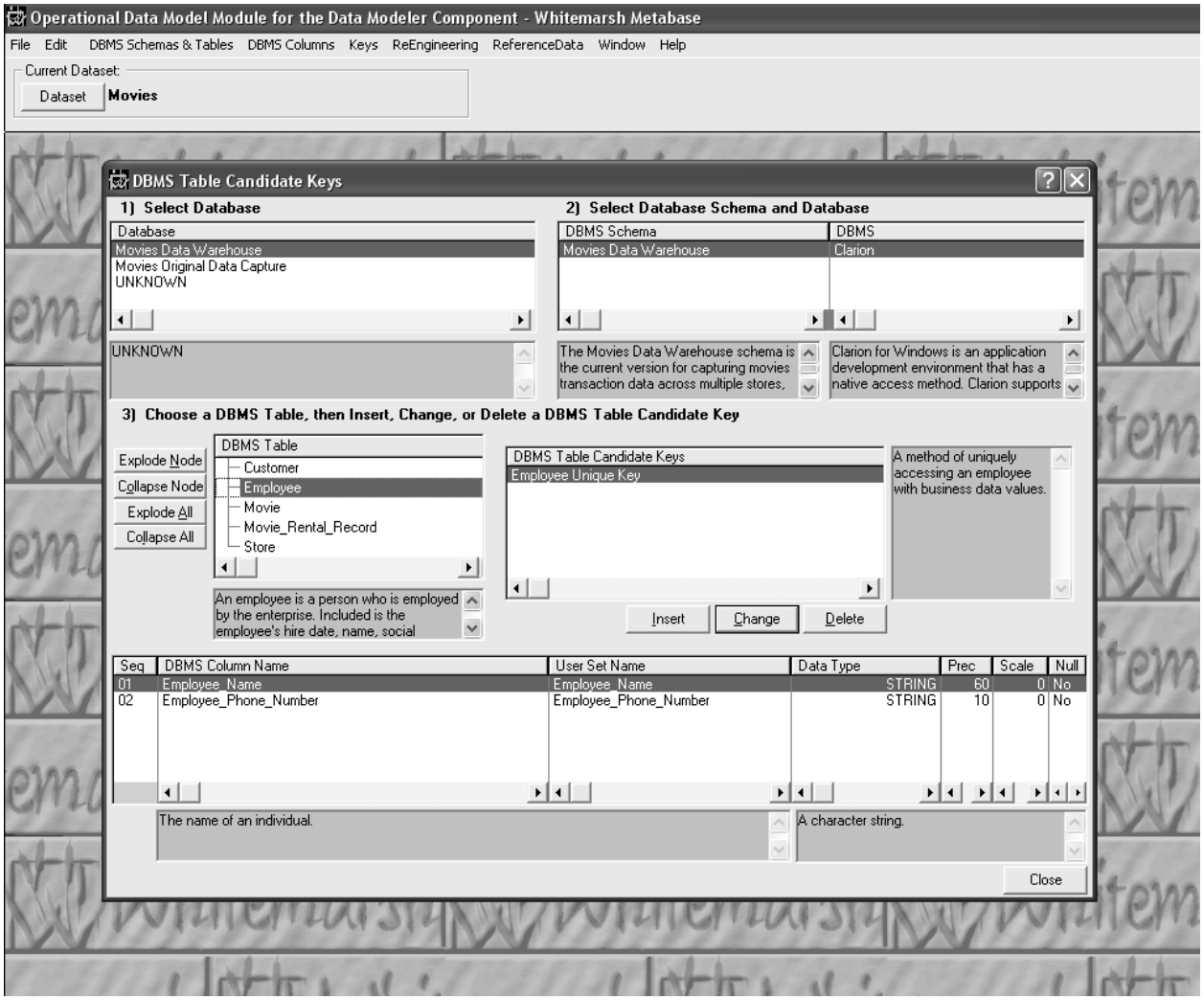

Figure 40. Candidate Key list.

keys. There can be multiple candidate keys for each DBMS table. To see the DBMS columns assigned to a particular candidate key, highlight the appropriate subject, and then DBMS table.

To add a candidate for an DBMS table press Insert. If the Insert or change action succeeds, the Figure 41 is presented. On an Insert, the name is automatically constructed as the concatenation of the DBMS table name and the string, "Candidate Key." The name can be changed. In addition to the name a description can be added or changed.

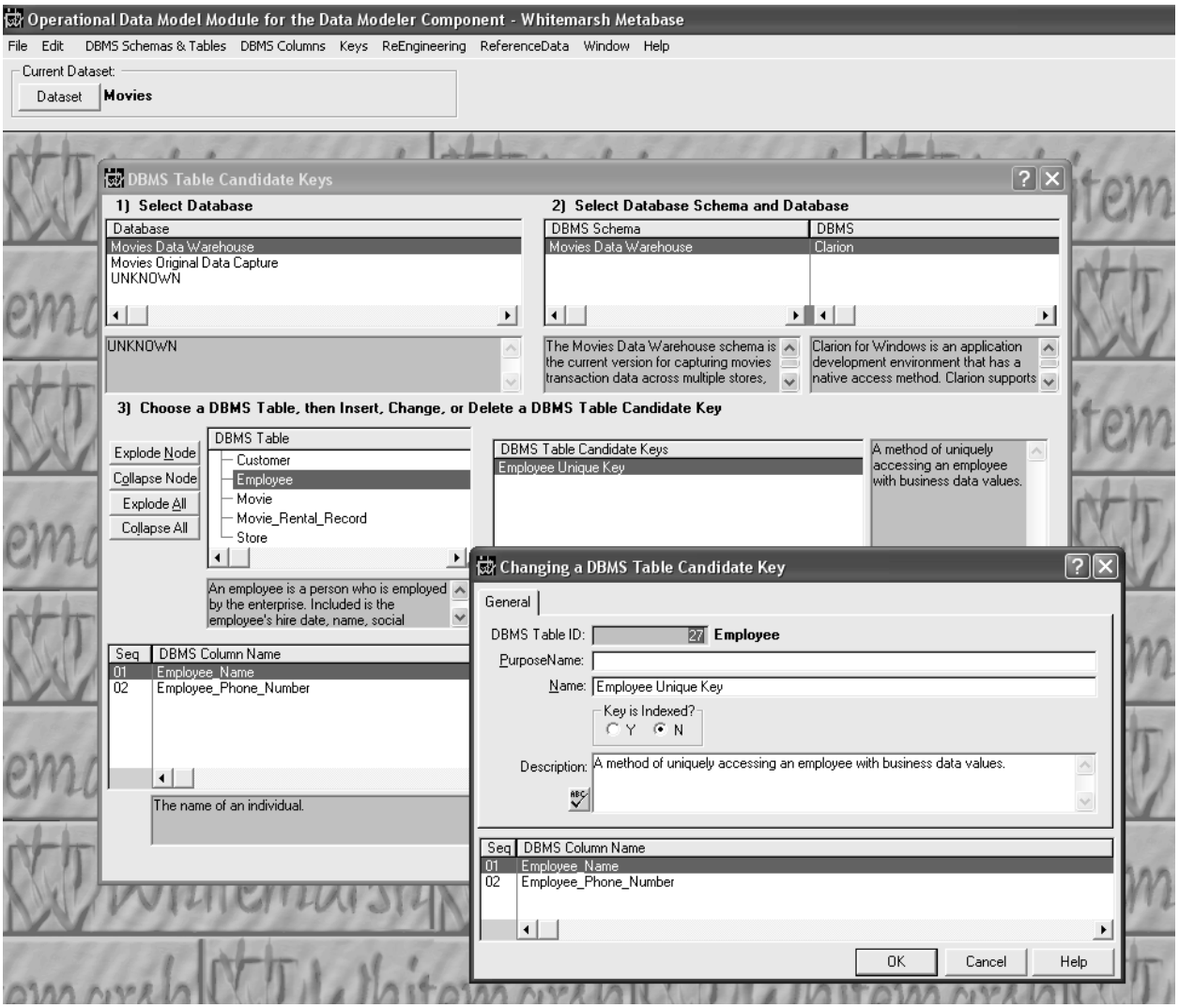

Figure 41. Candidate Key update screen.

6.2.3.3.2 Allocation of DBMS columns to the Candidate Key

A candidate key is a collection of DBMS columns. The order of the DBMS columns within the candidate key imply the order of the values used to select rows of data given that the DBMS table is in fact a DBMS table. Figure 39 presents a list of the candidate keys and the current set of DBMS columns assigned to each. To assign an DBMS column to a candidate key, highlight the subject then the DBMS table, then tag the appropriate DBMS table candidate key. Then, from the DBMS table's shown DBMS columns, tag the one or more DBMS columns that comprise the candidate key. Finally, press the Build button. The DBMS columns that then comprise the candidate key are shown in the bottom window. Once the DBMS columns are assigned, the Up|Down buttons can be used to change the sequence of the DBMS columns within the candidate key.

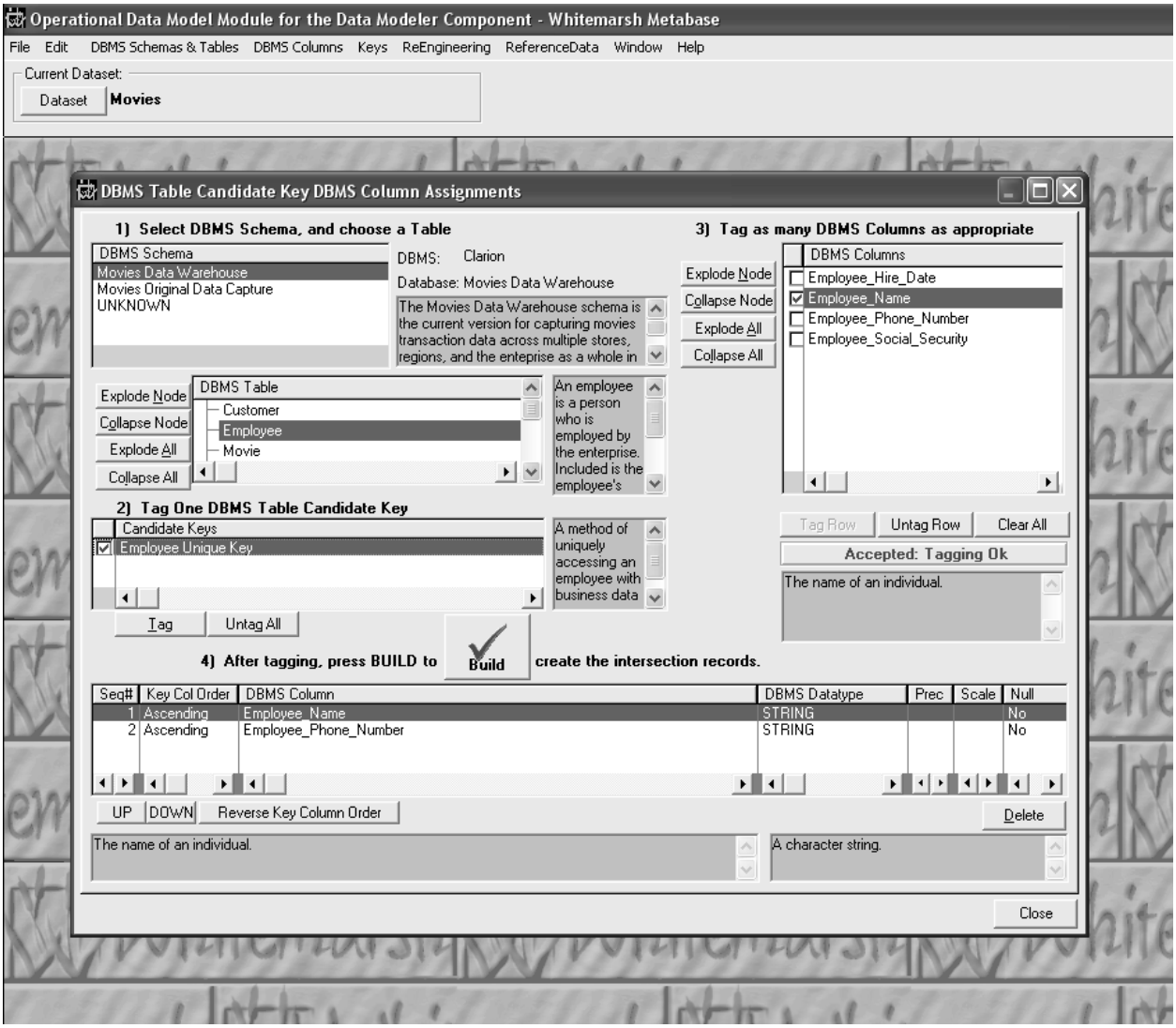

Figure 42. Candidate Key Column assignment.

6.2.3.3 Secondary

Included in the definition of a complete secondary key is:

- Secondary key definition
- Allocation of DBMS columns to the secondary key

6.2.3.3.1 Secondary Key Definition

A secondary key of an DBMS table is a set of one or more DBMS columns that represent values that when employed result in one or more selected rows. Figure 43 shows the current set of

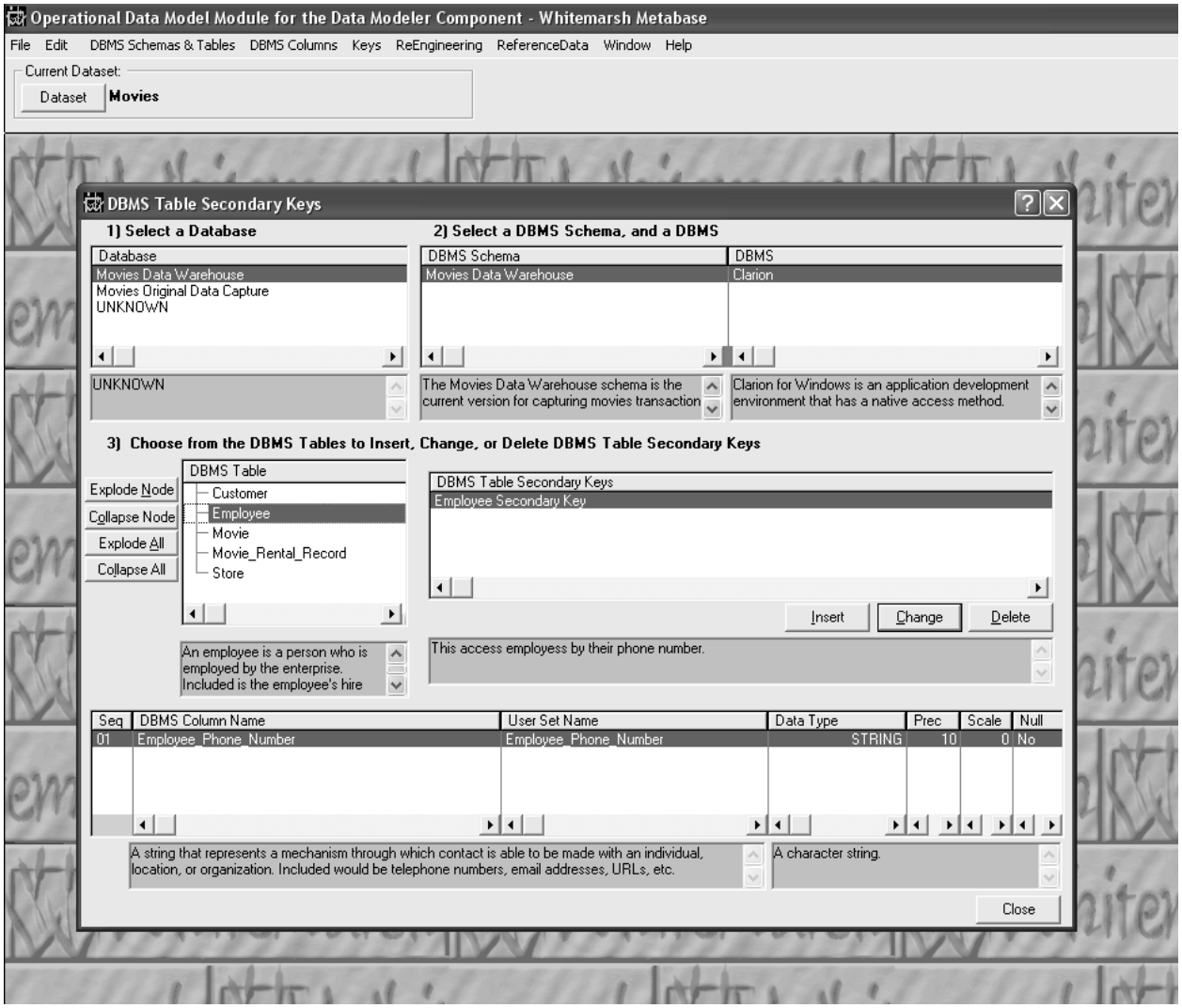

Figure 43. List of Secondary Keys.

secondary keys. There can be multiple secondary keys for each DBMS table. To see the DBMS columns assigned to a particular secondary key, highlight the appropriate subject, and then DBMS table.

To add a secondary for an DBMS 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 DBMS table name and the string, "Secondary Key." The name can be changed. In addition to the name a description can be added or changed.

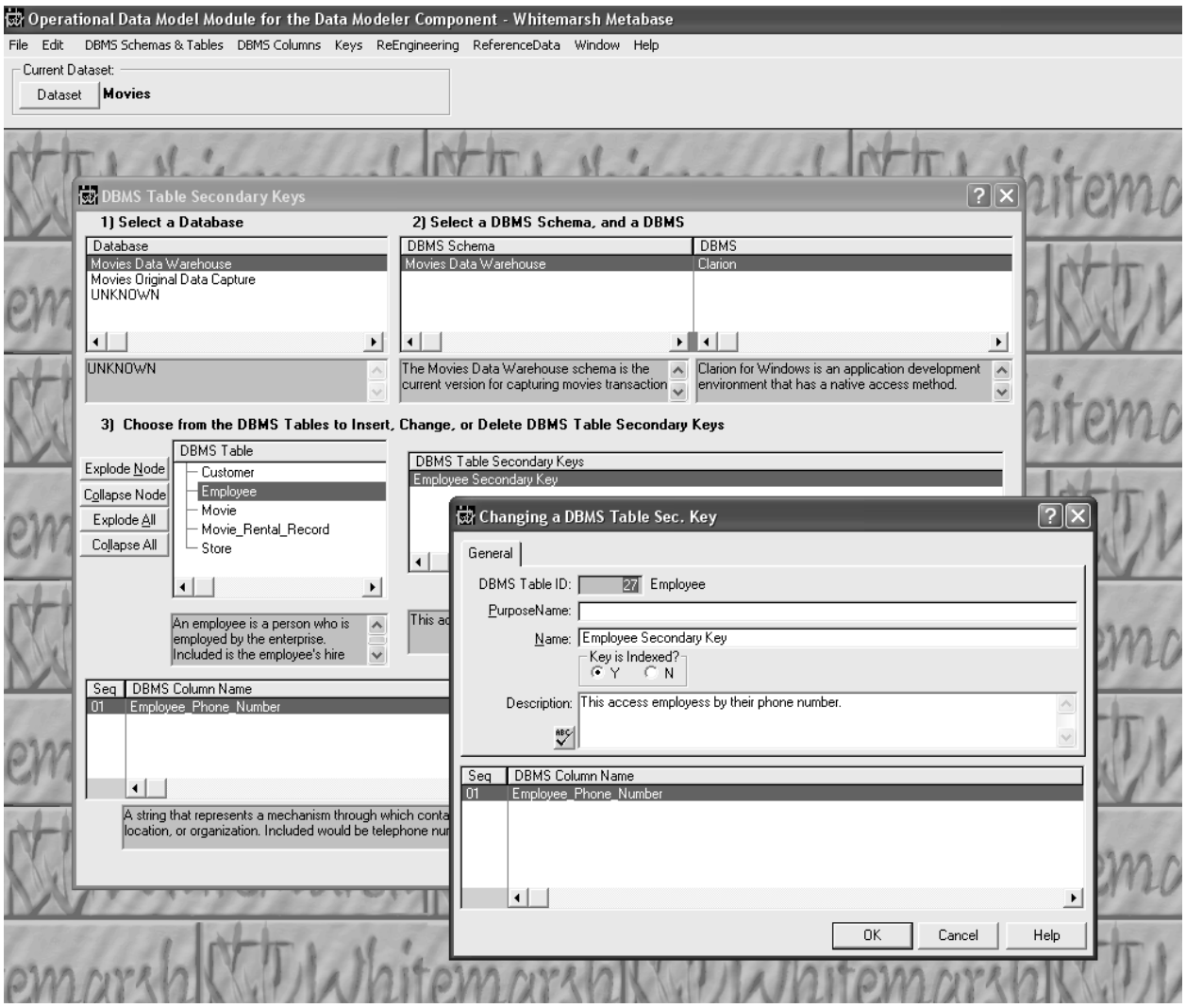

Figure 44. Secondary Key update screen.

6.2.3.3.2 Allocation of DBMS columns to the Secondary Key

A secondary key is a collection of DBMS columns. The order of the DBMS columns within the secondary key imply the order of the values used to select rows of data given that the DBMS table is in fact a DBMS table. Figure 45 presents a list of the secondary keys and the current set of DBMS columns assigned to each. To assign an DBMS column to a secondary key, highlight the subject then the DBMS table, then tag the appropriate DBMS table secondary key. Then, from the DBMS table's shown DBMS columns, tag the one or more DBMS columns that comprise the secondary key. Finally, press the Build button. The DBMS columns that then comprise the secondary key are shown in the bottom window. Once the DBMS columns are assigned, the Up|Down buttons can be used to change the sequence of the DBMS columns within the secondary key.

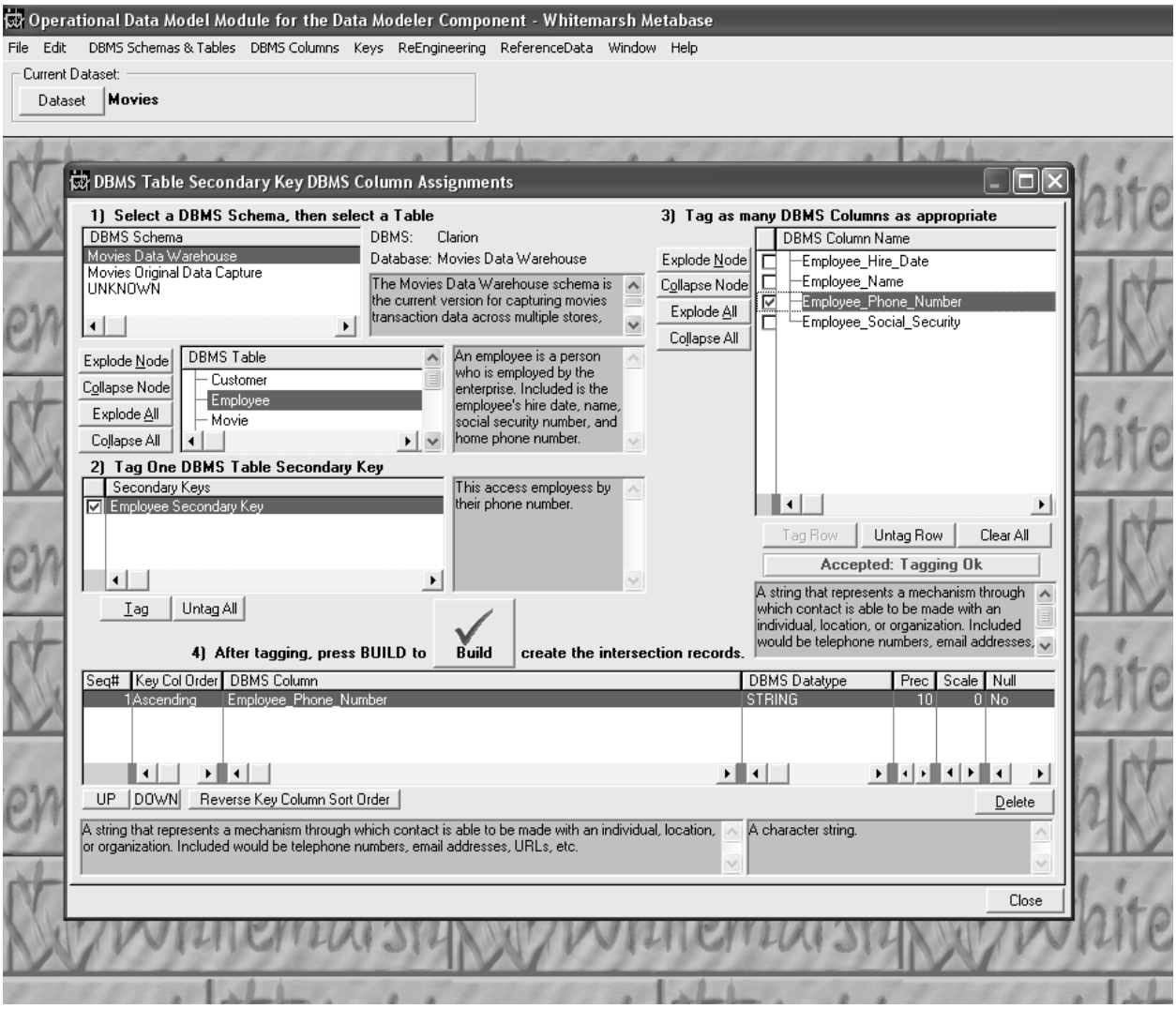

Figure 45. Assigning a DBMS Column to a Secondary Key.

6.2.4 Reverse Engineering

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

- Reassign DBMS column to column
- Reassign DBMS column to DBMS data type
- Reassign DBMS column to DBMS table
- Synchronize DBMS Column Local Definitions
- Reassign DBMS tables to DBMS schema
- Reassign DBMS table to DBMS Table
- Promote operational data model to implemented data model
- Promote operational data model table to an implemented data model
- Remove DBMS Table DBMS Column to Column assignments

6.2.4.1 Reassign DBMS Columns to Column

Figure 46 presents the screen through which DBMS columns are re-assigned different SQL data types. Tag one or more DBMS 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 DBMS columns appears in the left window.

A reason for reassigning a DBMS column to a column include, for example, importing a legacy database design and then mapping it to a canonical database design at the Implemented Data Model level. Or, to have created a canonical data model design at the Implemented Data Model level and to make several, but different Operational Data Model Designs.

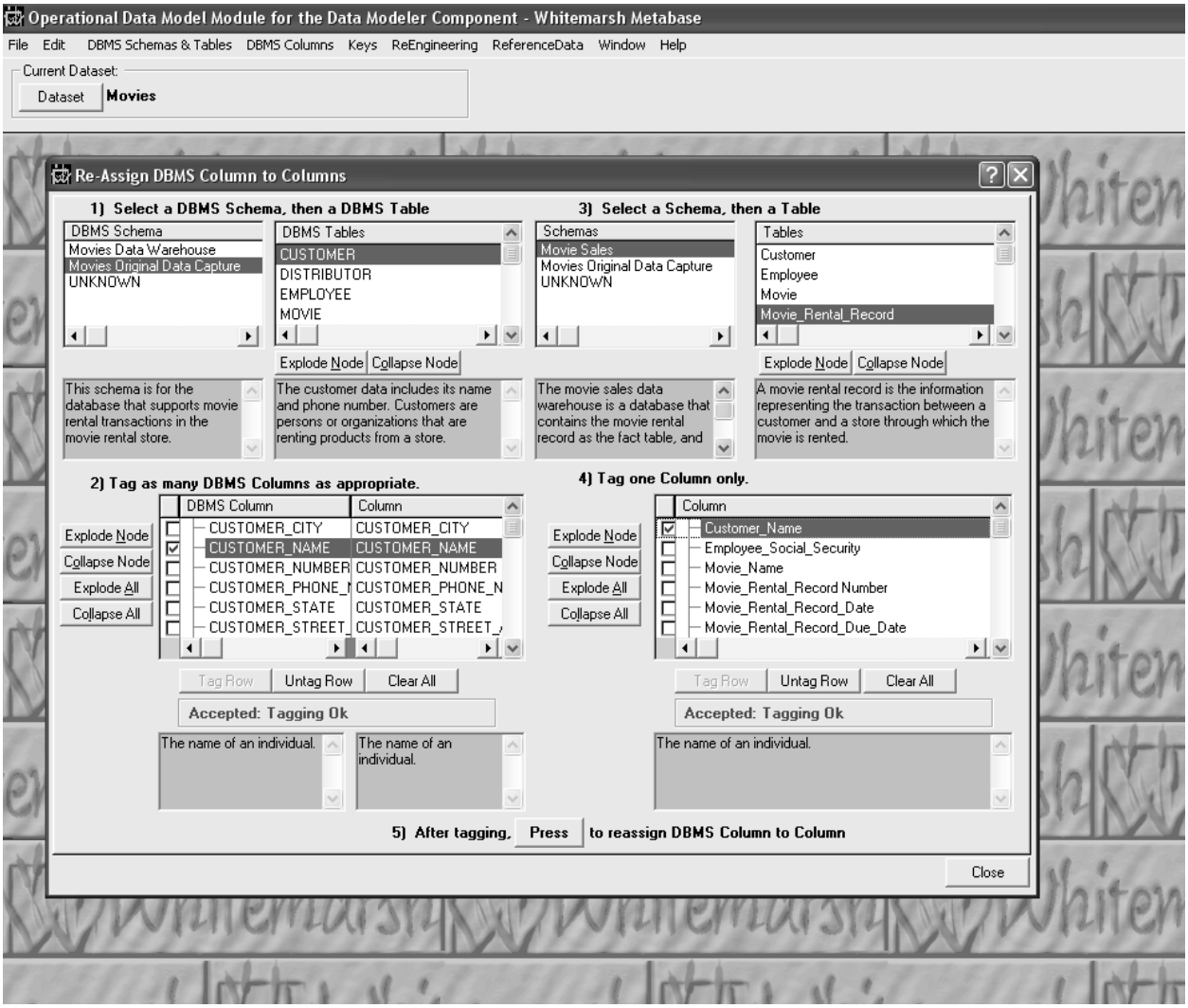

Figure 46. Reassigning a DBMS Column to a Column.

6.2.4.2 Reassign DBMS Columns to SQL Data Types

Figure 47 presents the screen through which DBMS columns are re-assigned different SQL data types. Tag one or more DBMS 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 DBMS columns appears in the left window.

When a implemented data model is imported as an operational data model or when an operational data model is promoted to be an operational 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 implemented 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.

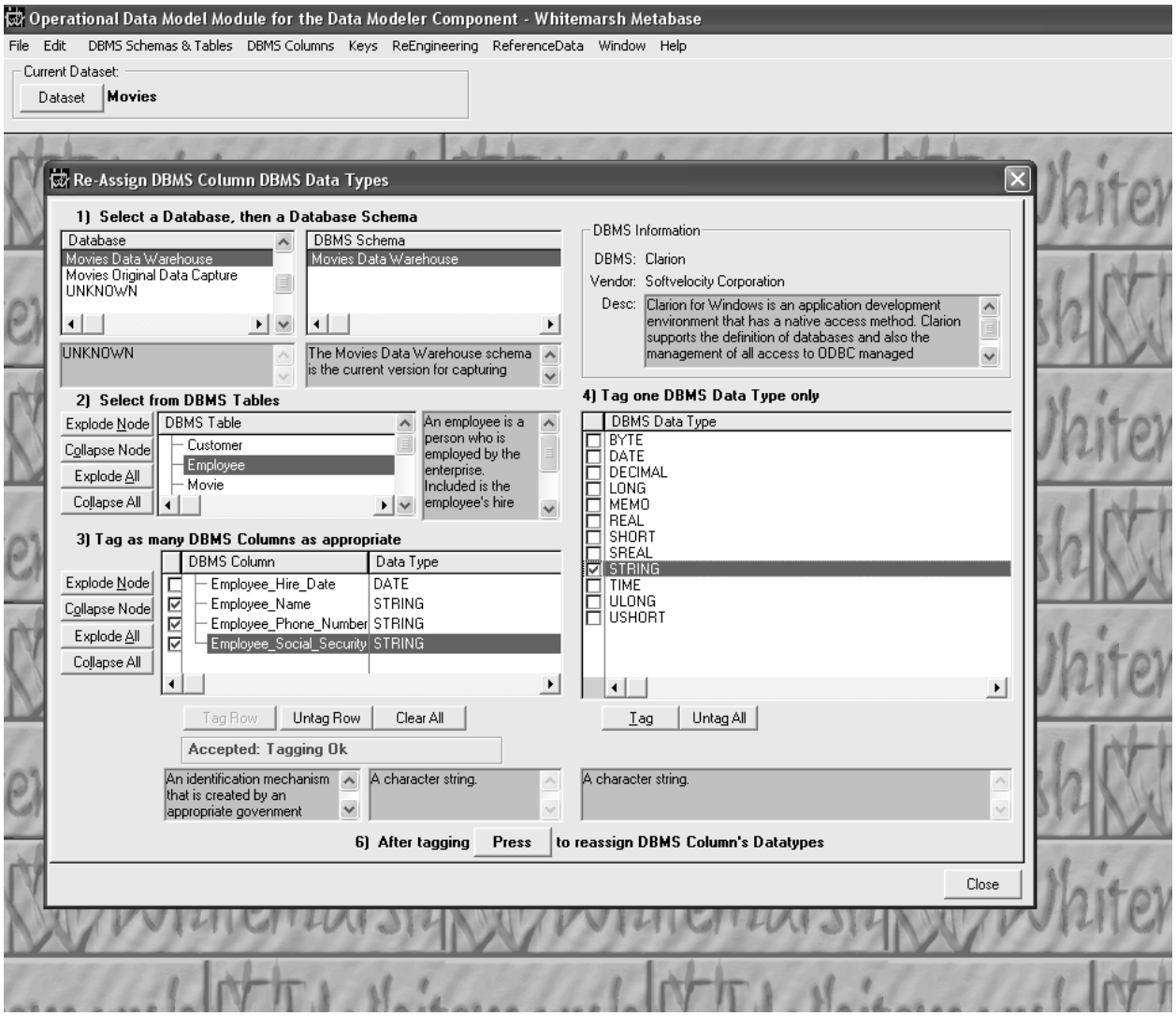

Figure 47. Re-assigning a DBMS Column Data Type to a different Data Type.

6.2.4.3 Reassign DBMS Column to DBMS Table

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

Figure 48 illustrates the process of reassigning one or more DBMS columns from one DBMS table to another DBMS table within the same family. Highlight the schema and then the DBMS table. Then tag one or more DBMS columns that are to be moved. Then tag the DBMS table to which the DBMS columns are to be moved. Press the reassigning button. The DBMS columns are then moved.

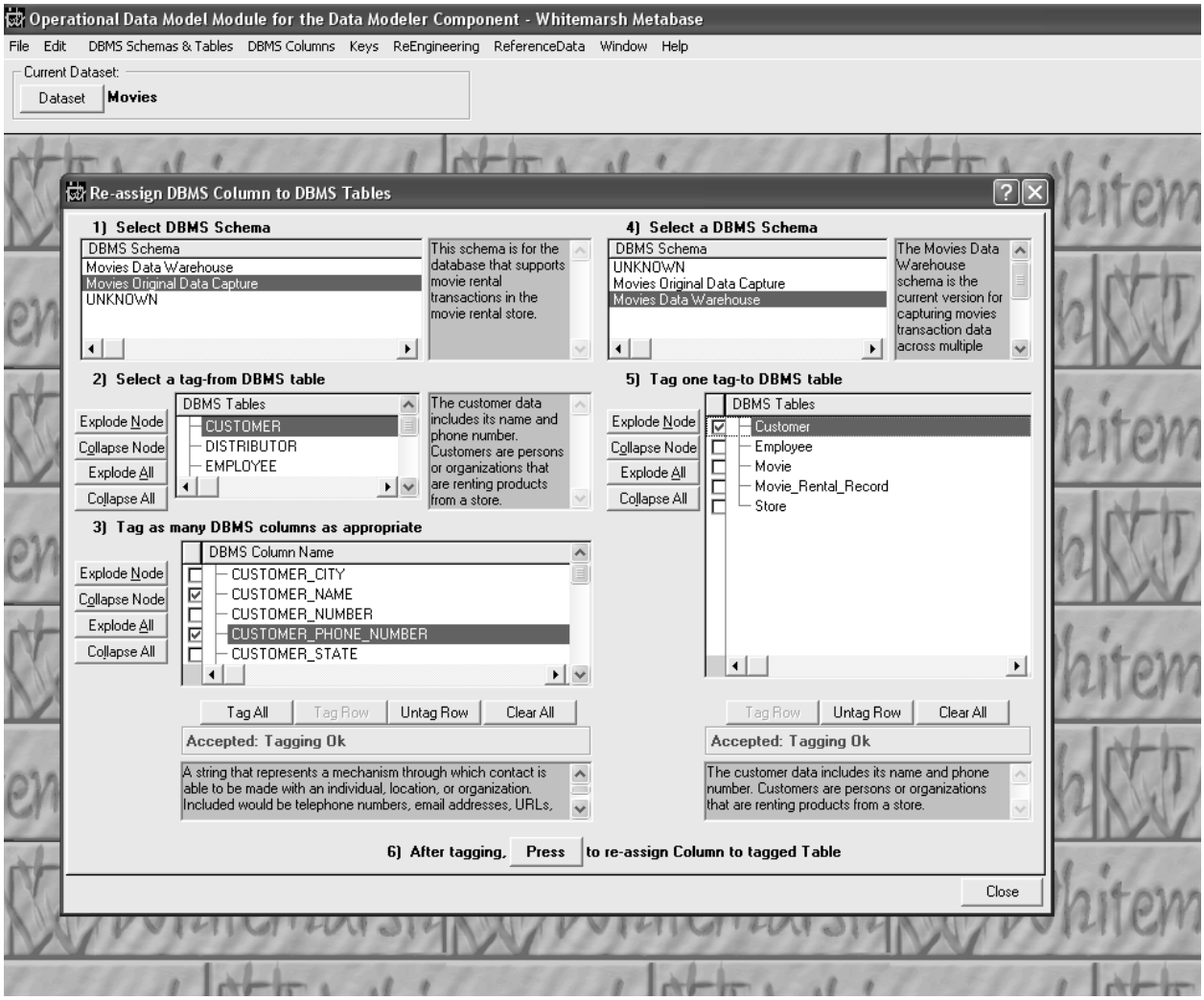

Figure 48. Re-assigning a DBMS Column to a different DBMS Table.

6.2.4.4 Synchronize DBMS Columns to Local Definitions

DBMS Columns have local definitions. Some DBMS columns within different DBMS tables and DBMS Schemas are to have essentially the same local definition. For example, in several of the DBMS tables within this movies demo database has a Movie Name attribute. For sure, in all cases the local definition would be, "the name of the movie." If this local definition is different across the DBMS columns of the different DBMS tables and DBMS Schemas, users will wonder why the difference exists. What's the hidden meaning because of the difference? Figure 49 provides the ability to synchronize local definitions across DBMS columns. On the left side, select a DBMS Schema, DBMS table, and then tag the DBMS column that is to be source of the local definitions. On the right side of Figure 49, select as many different DBMS columns as may be appropriate. Do this by selecting the DBMS Schemas, DBMS tables, and DBMS columns. Different DBMS columns from the different DBMS tables, and DBMS 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.

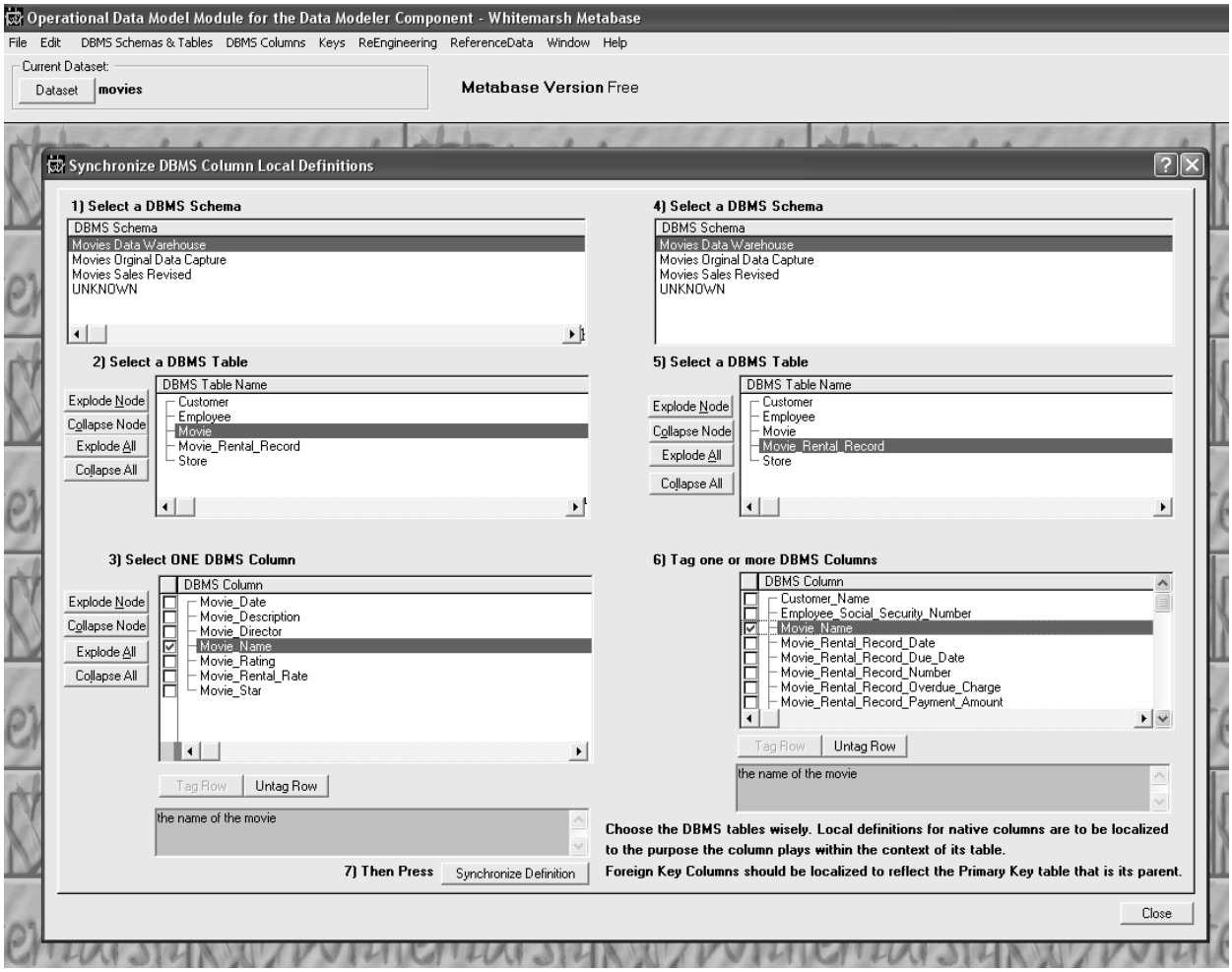

Figure 49. Synchronize DBMS Column local definitions.

6.2.4.5 Reassign DBMS Tables to DBMS Schema

Figure 50 displays the process for re-assigning DBMS tables to different DBMS schemas. The process starts with highlighting the DBMS table that is to be re-assigned. Tag one or more DBMS tables that are to be re-assigned. Then highlight and tag the new DBMS schema. Once the DBMS tables and a DBMS schema is tagged, press the Re-Assign button. The underlying process then re-assigns the DBMS table from the current DBMS schema to the new DBMS schema. Once all the tagged DBMS tables are reassigned the windows are redisplayed with the newly assigned DBMS schemas.

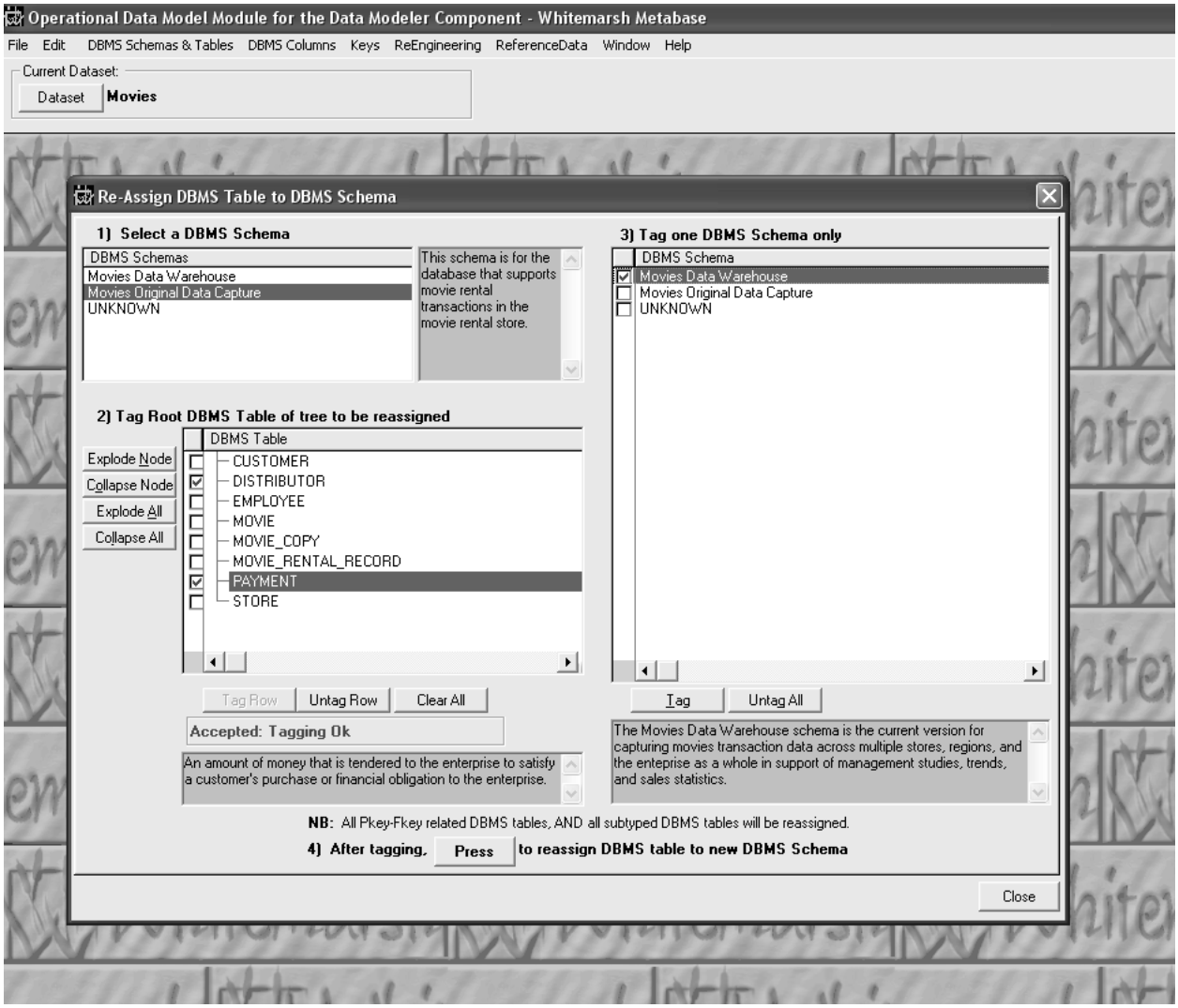

Figure 50. Re-assign DBMS Tables to a different DBMS Schema.

6.2.4.5 Reassign DBMS Tables to DBMS Table

As stated above, DBMS tables can have sub-typed DBMS tables. This is shown in Figure 51. A DBMS table might be pushed too far down in the DBMS table family. In this case the reassignment allows the "parent" of an sub-typed DBMS table to be changed. To accomplish this, tag one or more DBMS 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 DBMS table.

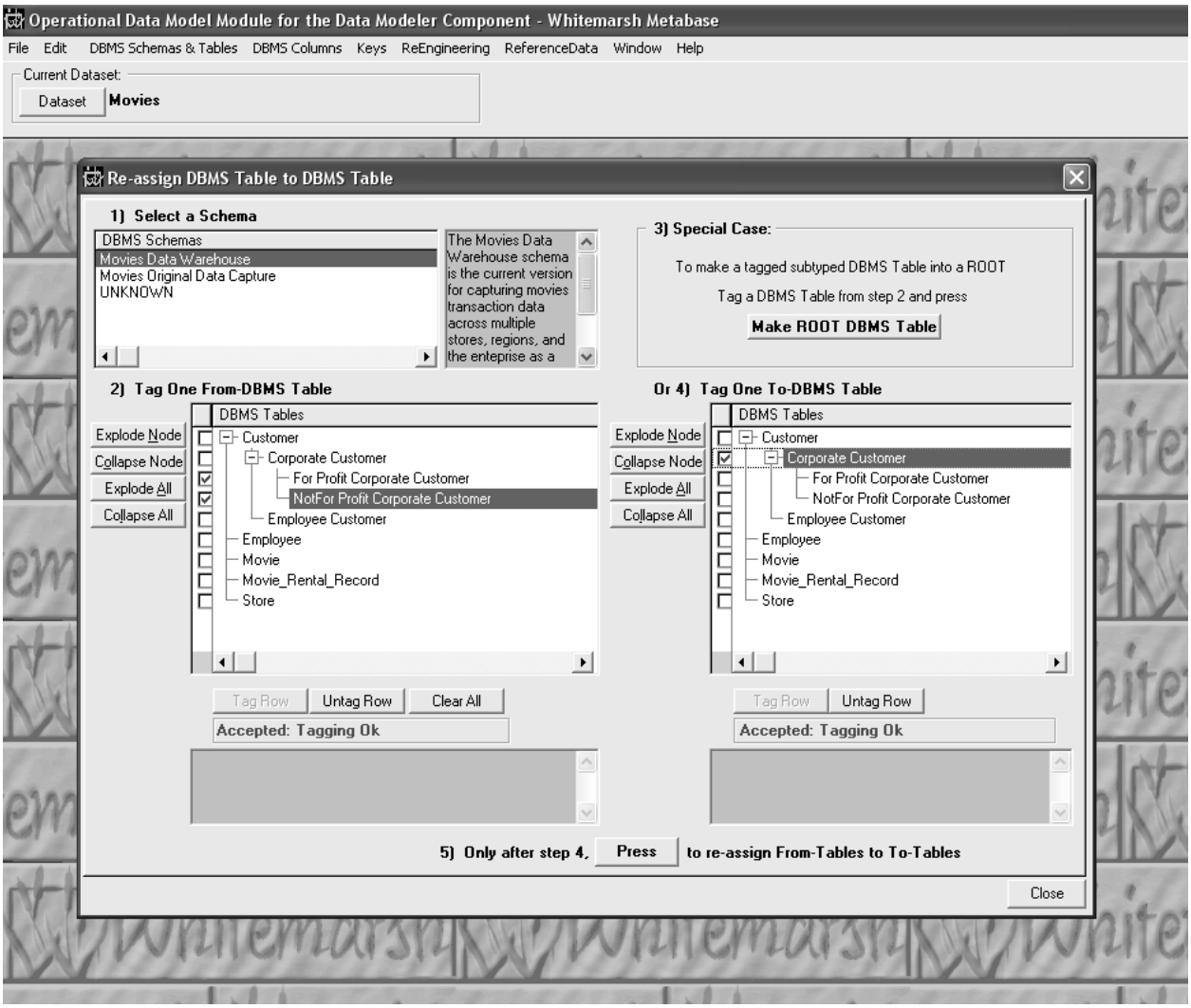

Figure 51. Re-assigning a subDBMS Table to a different DBMS Table.

6.2.4.7 Promote Operational Data Model to Implemented Data Model

Figure 52 presents the screen for promoting an operational data model to be a implemented data model. When an operational data model is promoted, its collected set of DBMS tables, DBMS columns, DBMS column value domains, and primary, foreign and candidate keys are all created anew within the implemented data model. Once the new implemented data model components are created, the operational data model DBMS column value domains are deleted. That is because they are automatically inherited by the operational data model and would therefore be redundant.

The promotion process is simple. Highlight the operational data model DBMS schema then press the Promotion button. The process accomplishes what is described above and then the newly promoted tables and columns appear underneath the highlighted schema.

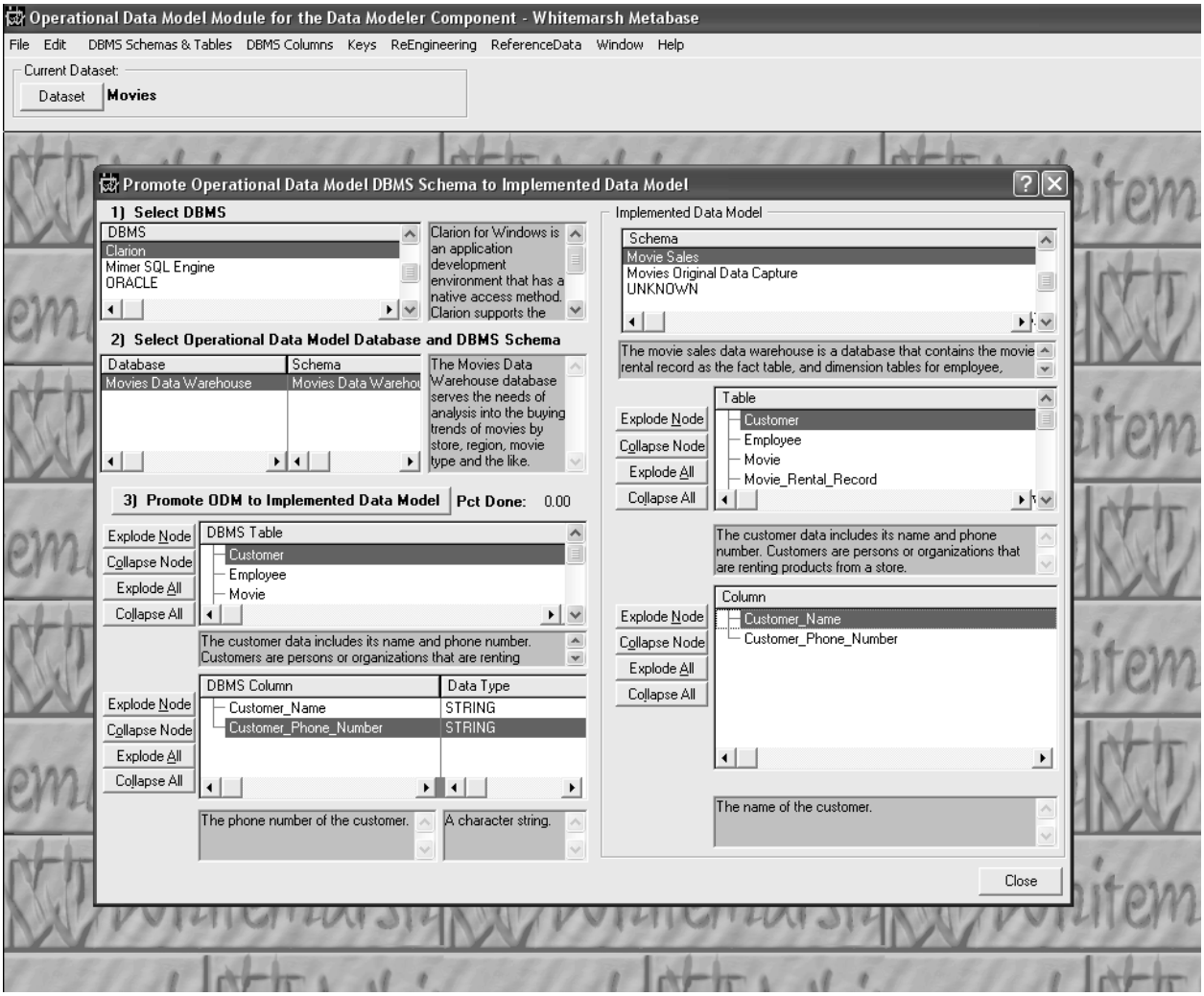

Figure 52. Promote Operational Data Model to Implemented Data Model.

6.2.4.8 Promote Operational Data Model Table to Implemented Data Model

Figure 53 presents the screen for promoting an operational data model table to be a implemented data model. When an operational data model table is promoted, the DBMS table, DBMS columns, DBMS column value domains, and primary, and candidate keys are all created anew within the implemented data model. Once the new implemented data model components are created, the operational data model DBMS column value domains are deleted. That is because they are automatically inherited by the operational data model and would therefore be redundant.

The promotion process is simple. Highlight the operational data model DBMS schema and then the DBMS Table. Then press the Promotion button. The process accomplishes what is described above and then the newly promoted table and columns appear underneath the highlighted schema.

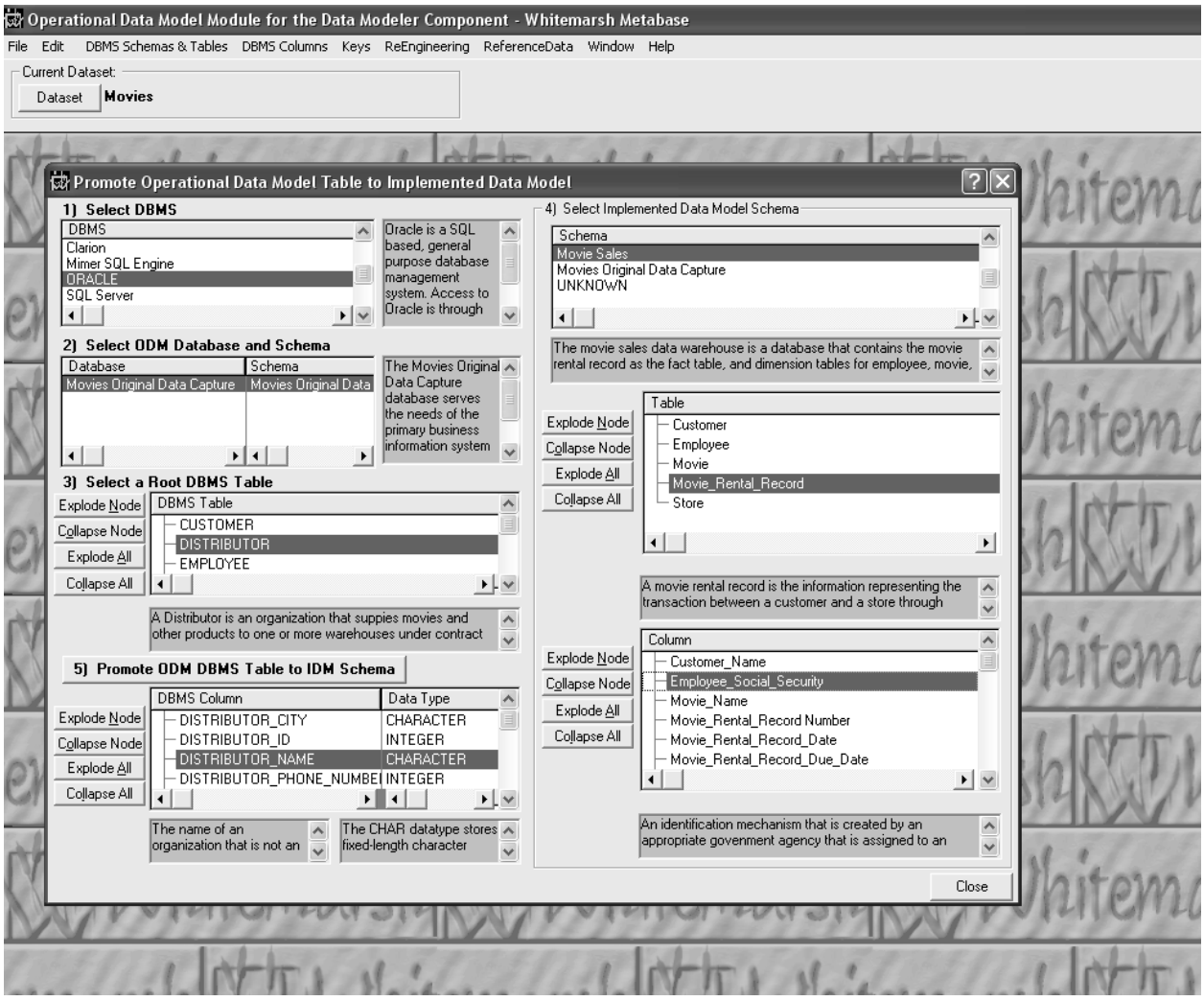

Figure 53. Promote DBMS Table to a Implemented Data Model Schema.

6.2.4.9 Remove DBMS Table DBMS Column to Column Assignments

Figure 54 displays the window for removing the column assignments for a DBMS column. Select the schema and table. Then press the "Press" button to remove the assignments. When complete, all the DBMS Column assignments will be shown as "unknown.

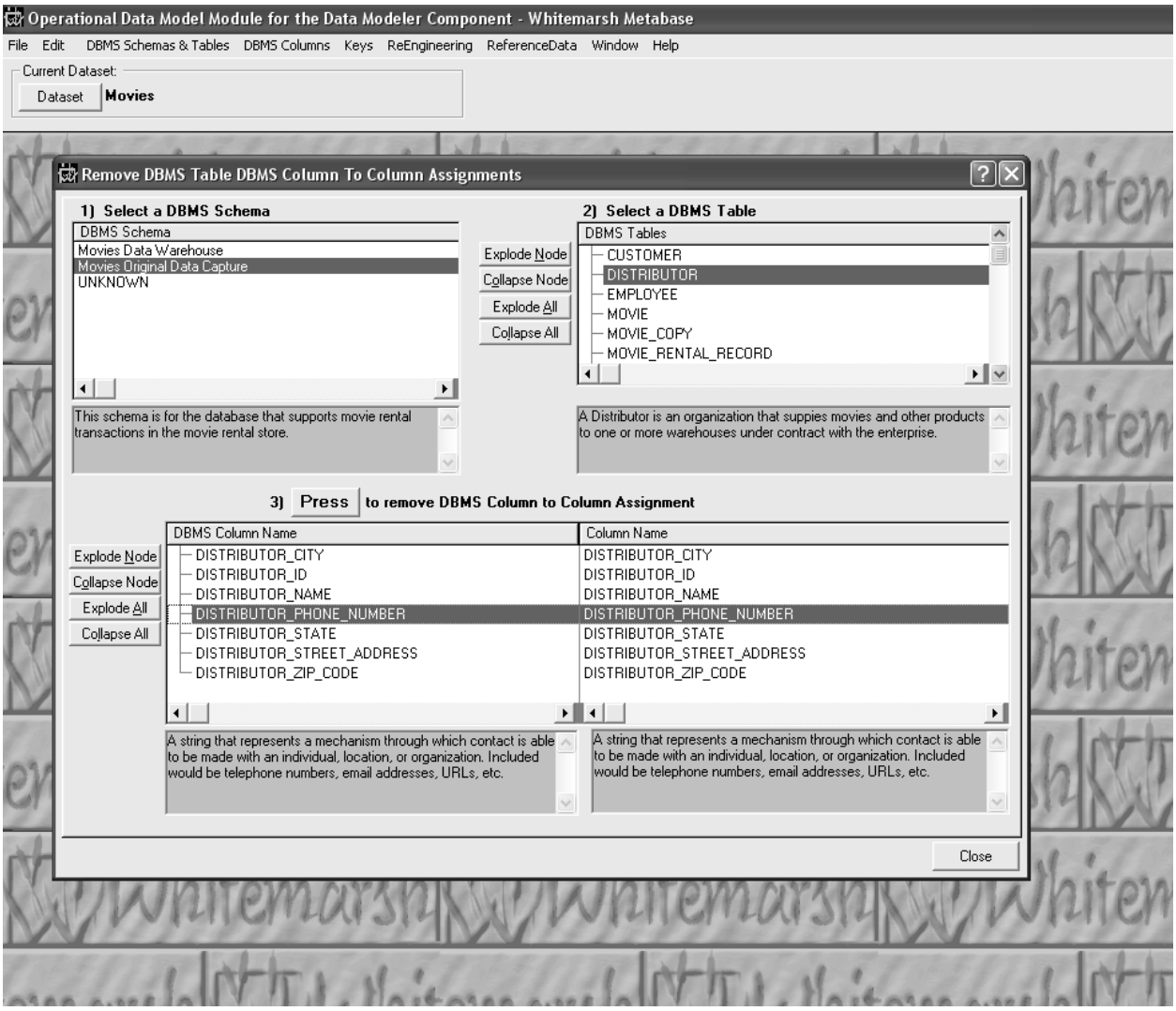

Figure 54. Remove DBMS Table DBMS Column to Column assignments.

6.2.5 SQL DDL

Once an operational data model has been completely entered, that is, its its DBMS tables, DBMS columns and relationships, it can be displayed either graphically within the specified 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 operational data model. Thus, the metabase has an SQL DDL import facility.

6.2.5.1 SQL DDL Export

Figure 55 presents the first screen in this process. The top browse shows the DBMS schemas. The second browse shows an alphabetical listing of the DBMS tables within that DBMS schema. Obviously the root DBMS table is not always the first. If the root DBMS table is highlighted and then the button Display Operational Data Model is pressed, then a screen like that of Figure 55 is presented.

The process that is invoked starts with the highlighted DBMS table and traces through all its foreign keys to then "know" all related DBMS tables. The hierarchy that is displayed in Figure 56 is all the descendants of the DBMS table and all the direct ancestors (no uncles or aunts). When an ancestor DBMS table is displayed it's color is red. Descendent DBMS tables are blue and the originally highlighted DBMS table's color is black.

As each DBMS table in Figure 56 is highlighted, the surrounding browses then display the DBMS table's primary key and DBMS columns, foreign keys and DBMS columns, and DBMS 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 55 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 57. 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 DBMS schema Based Data Model button. If selected, Figure 58 is displayed and all the DBMS tables within the DBMS 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 DBMS table Based Data Model button only generates SQL DDL for all DBMS tables that are descendants and direct ancestors of the highlighted DBMS table.

There are three sets of options for generating the SQL DDL. These are:

- Name Choice: Full, User Set, or Abbreviation
- Case Choice: All Lower Case, all upper case, or camel case
- SQL DDL Subtype choice: One, each, or SQL 1999

The first two sets of options are obvious and need not be explained. The third choice regards representing subtyped DBMS tables: One, Each, and SQL:1999. The one option causes all the DBMS columns from the contained subtyped DBMS tables to appear within the root DBMS table. The Each option causes each subtyped DBMS 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.

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 59 is displayed.

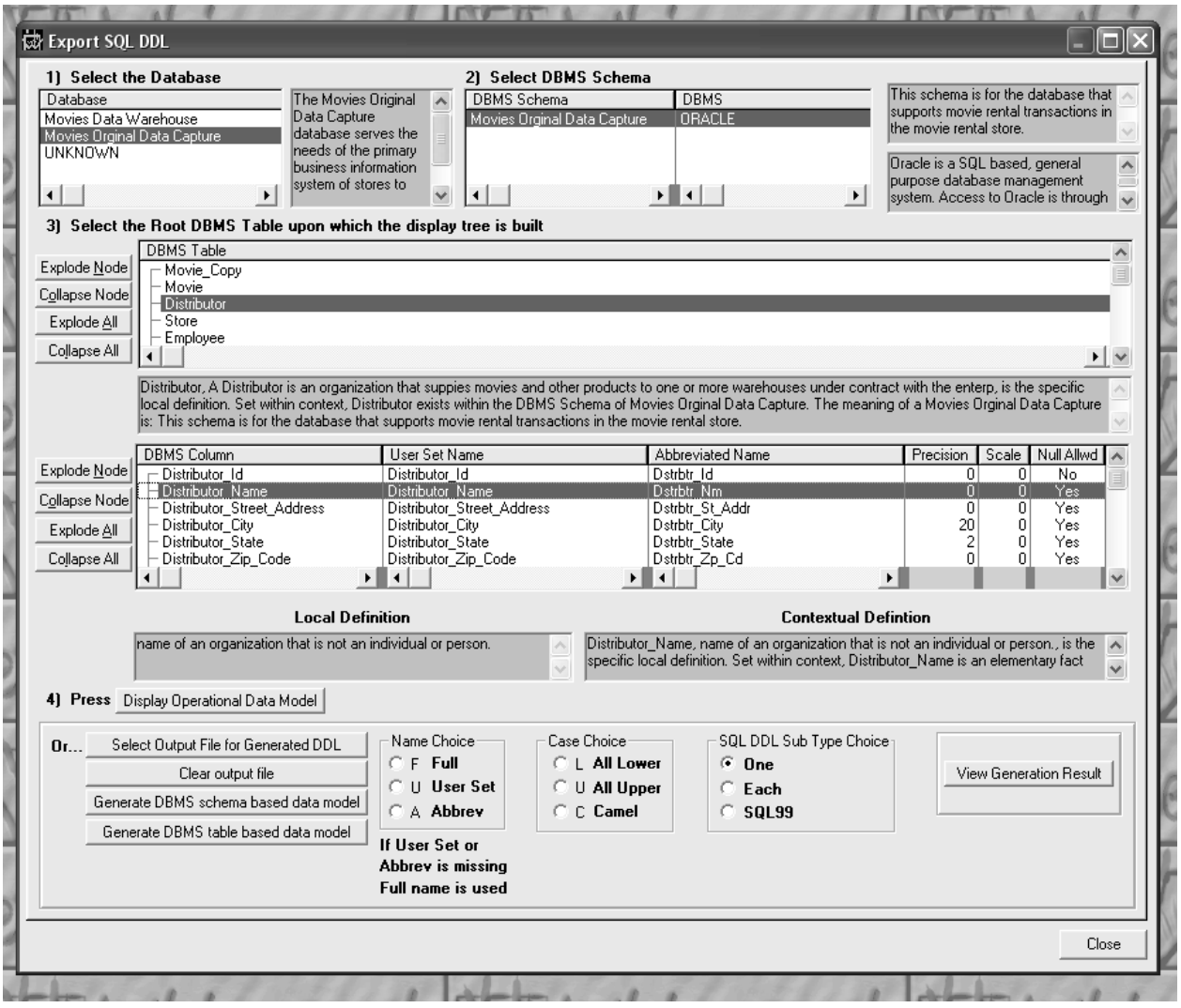

Figure 55. SQL DDL export screen.

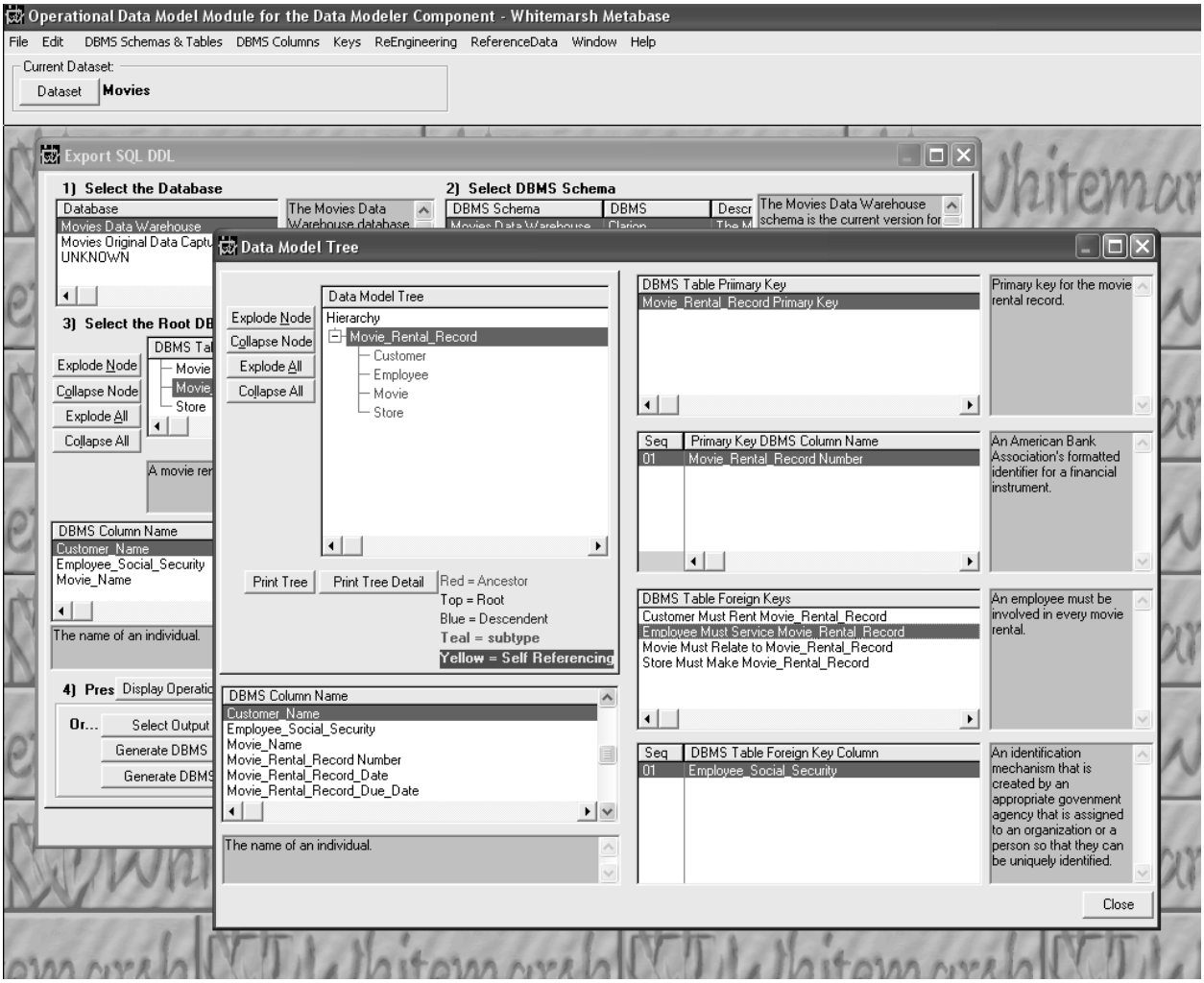

Figure 56. Message from Schema based SQL DDL generation.

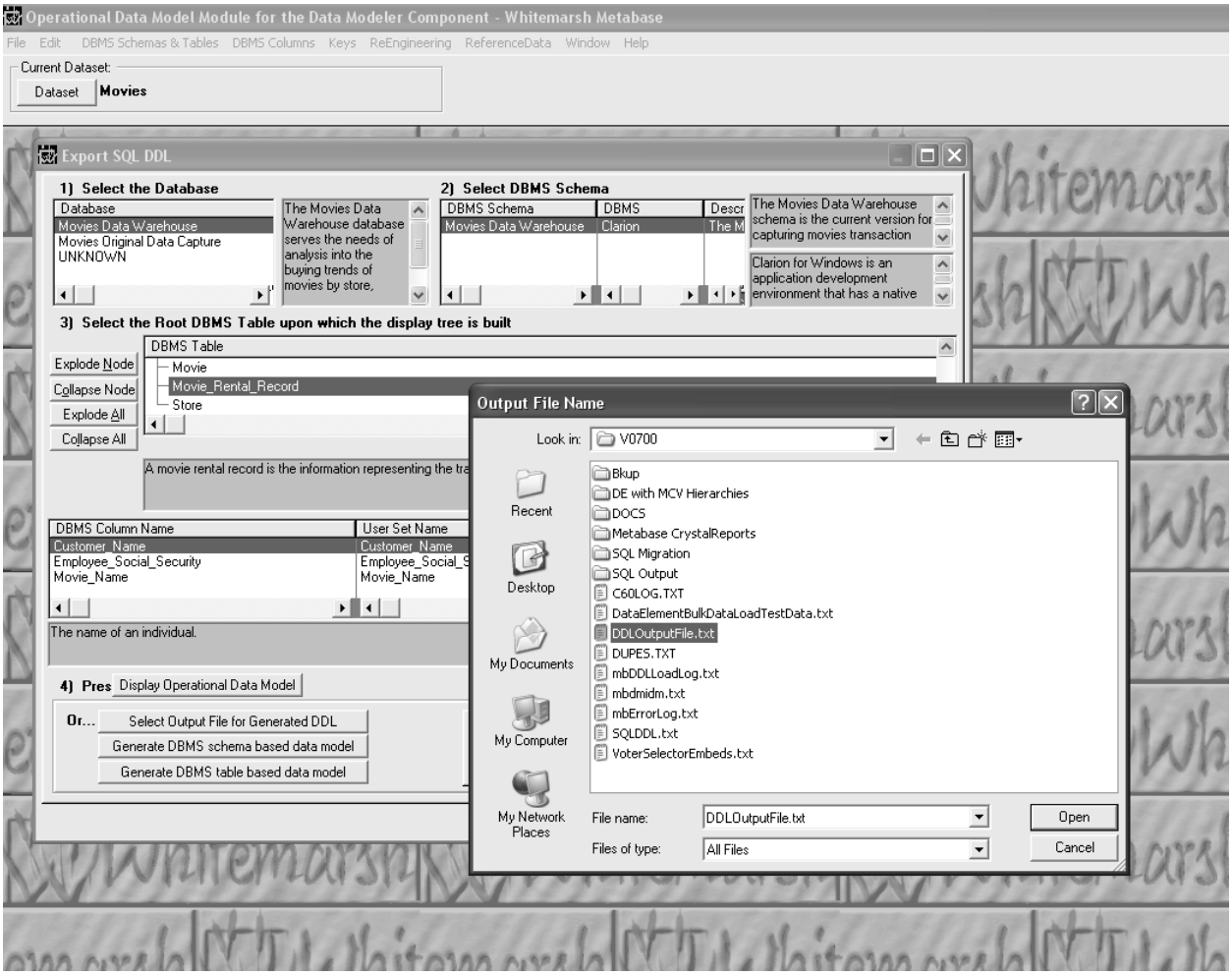

Figure 57. Selecting output file for SQL DDL generation.

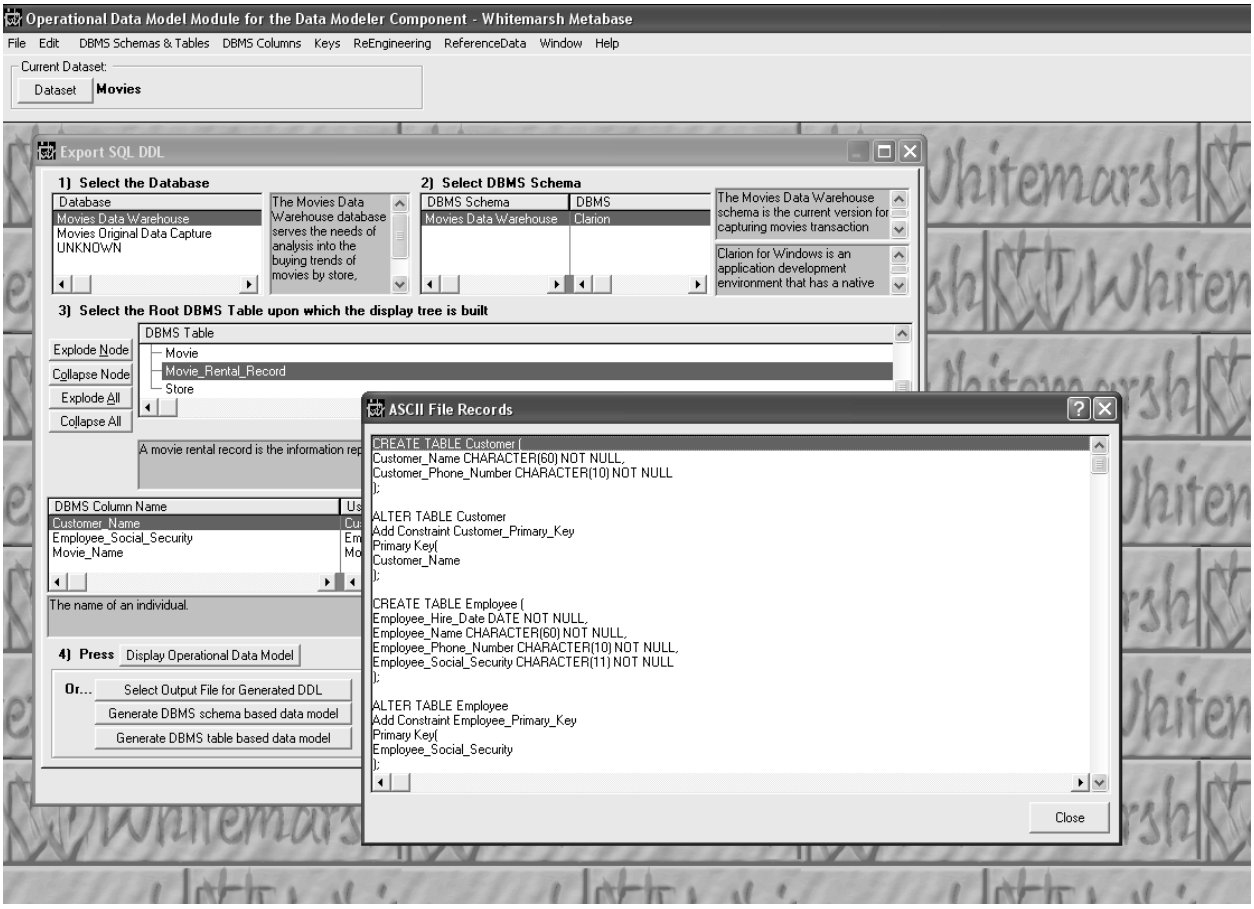

Figure 58. Generated SQL DDL output file listing.

6.2.5.2 Import

SQL DDL streams can be imported directly into the Operational Data Model. Figure 59 presents the first screen of this process. Fundamentally, the process is to first create a DBMS 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 DBMS tables, DBMS columns, data types, primary and foreign keys.

Figure 59 presents the first screen. If the DBMS schema does not already exist then press the Insert button. The update Schema screen is presented. Once the DBMS schema is created, then highlight the just created DBMS schema and proceed.

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

Figure 60 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 61.

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 62 presents a screen of the log file lines.

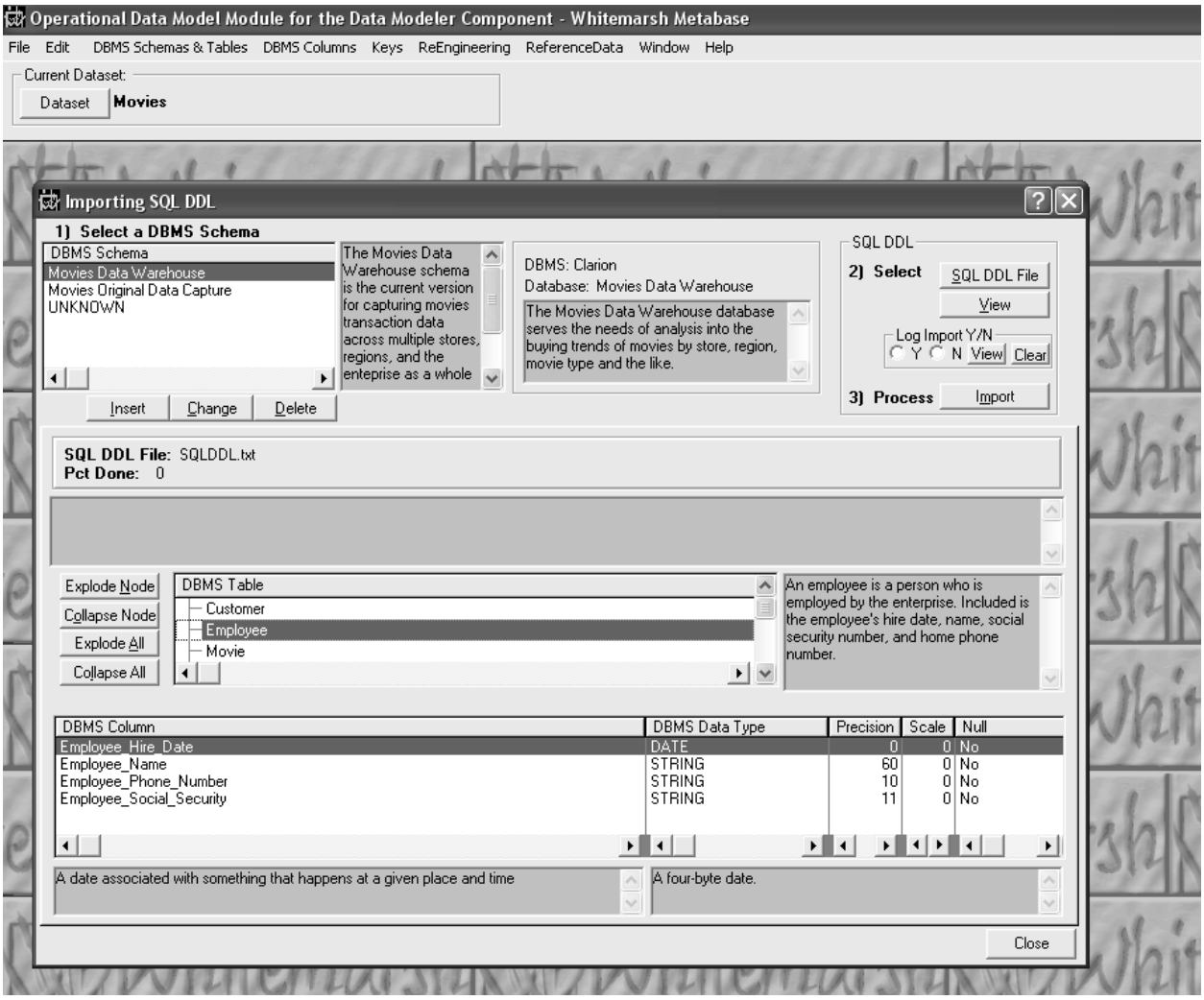

Figure 59. SQL DDL Import screen.

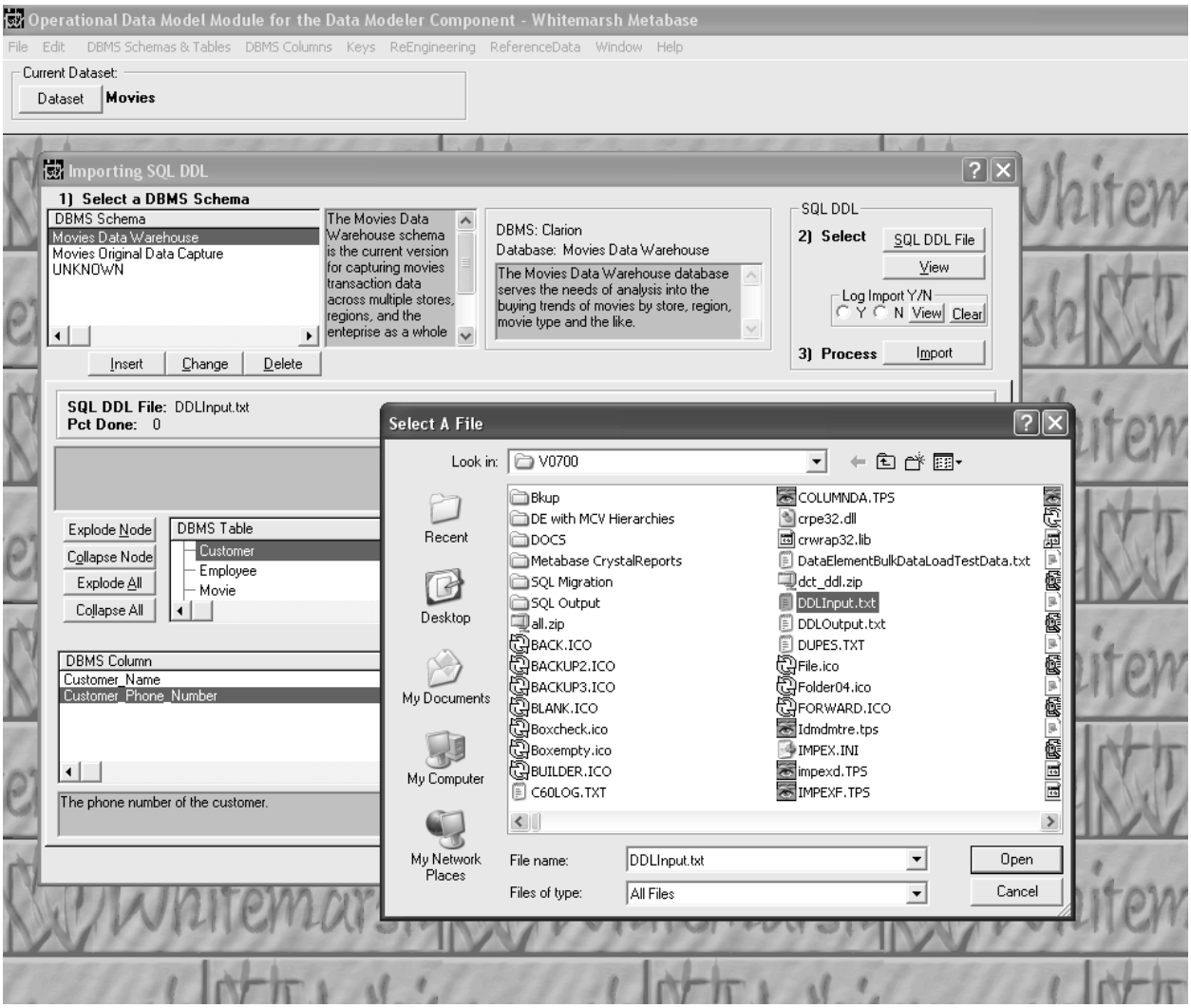

Figure 60. Selecting a SQL DDL import file.

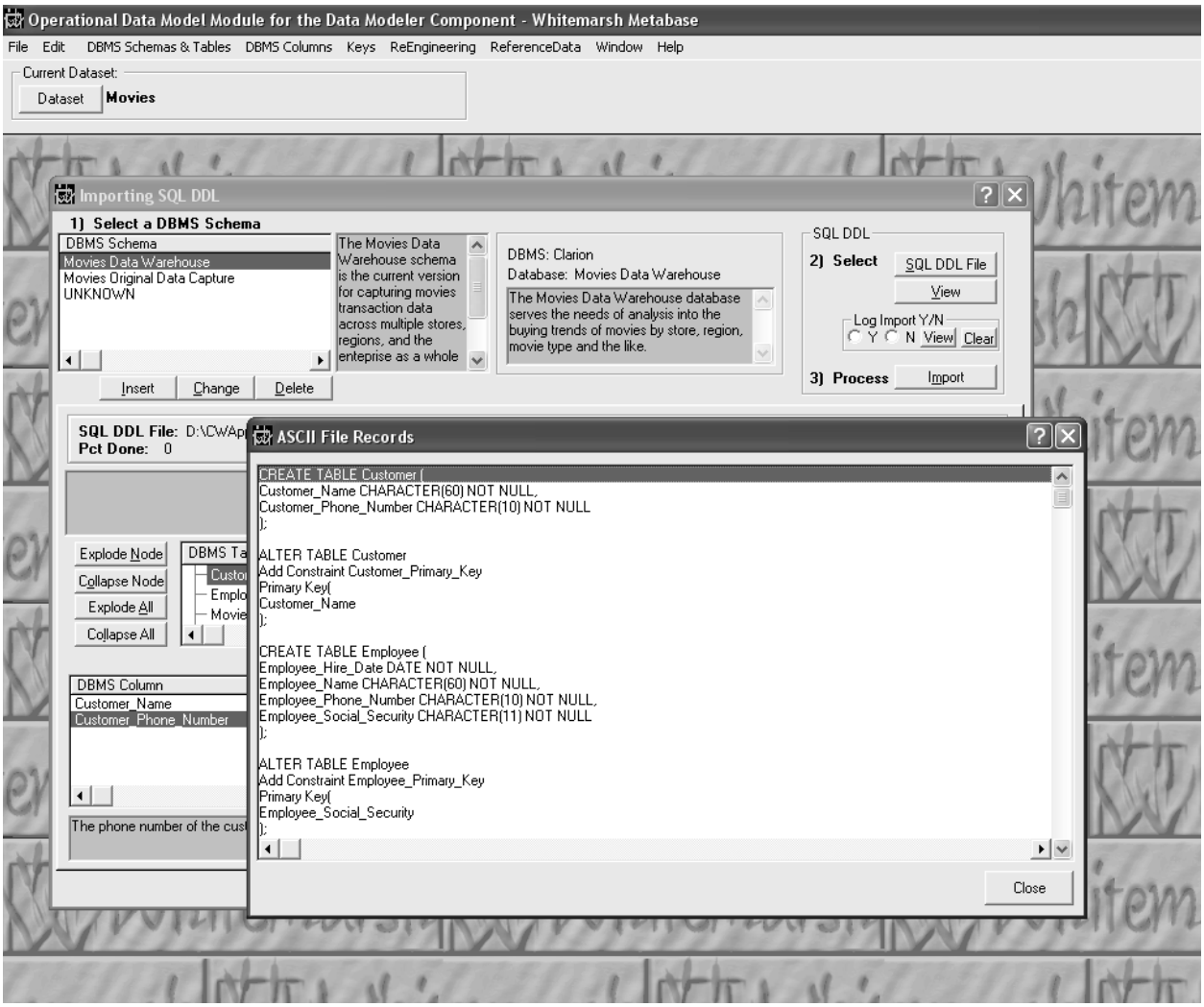

Figure 61. Example of a SQL DDL import file.

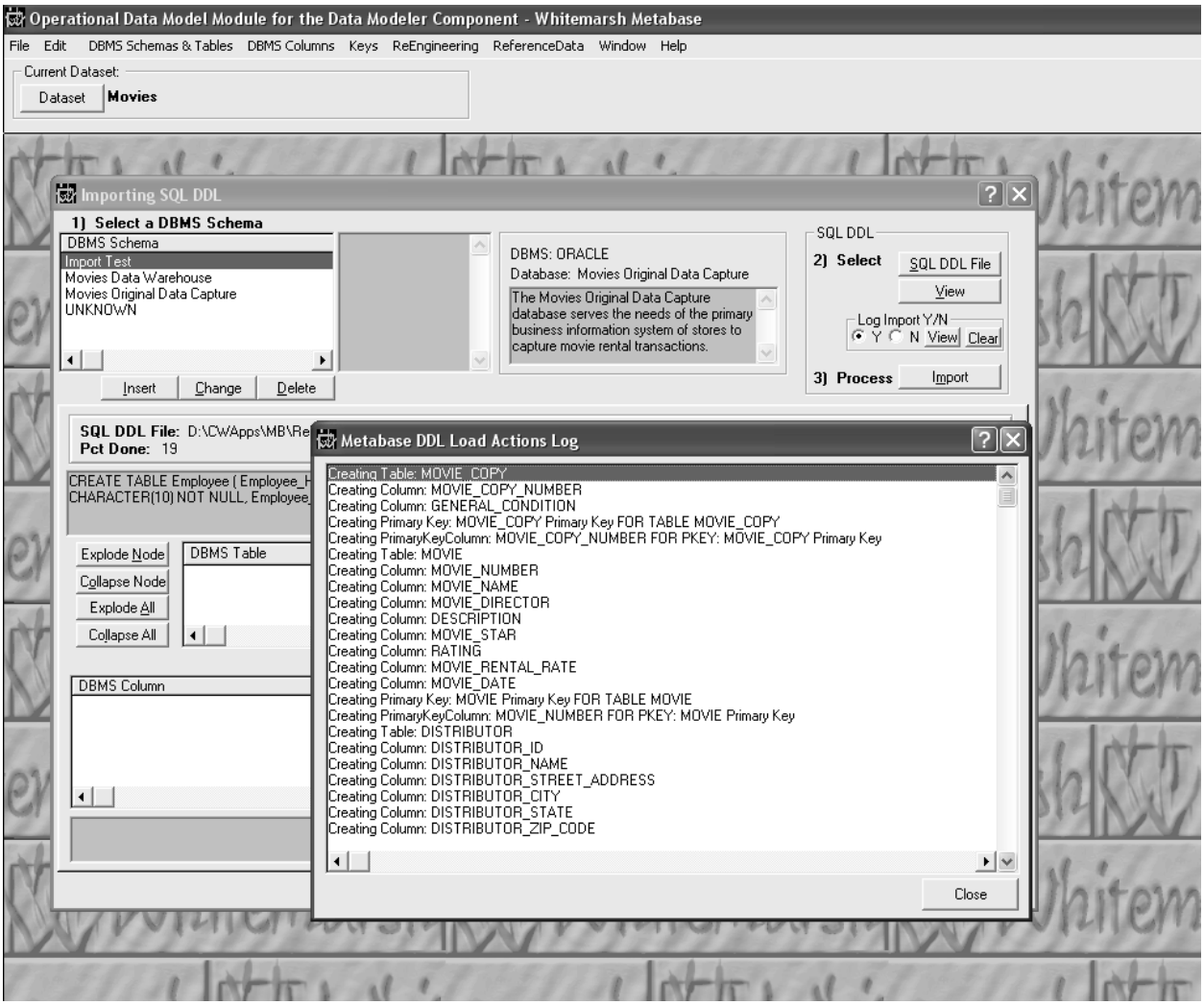

Figure 62. Log File display of the SQL load process.

6.2 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.