An Oracle White Paper
June 2009

Oracle SQL Developer Data Modeler Naming Standardization

Introduction

Oracle SQL Developer Data Modeler provides a full spectrum of data modeling tools and utilities, including Entity Relationship modeling, Relational (Database Design), Physical, Data Type and Multi-dimensional modeling; full forward and reverse engineering between Logical and Relational models, DDL code generation. It includes importing from and exporting to a variety of sources and targets, provides a variety of formatting options and validates the models through a predefined set of design rules.

Modeling tools are powerful communications devices, providing developers with a vehicle to communicate with business users about the business. Developers use these same models to generate the code. They can also be used to enforce standards within an application. Using a tool to enforce or control standards in an application means significant productivity gains. It also means that the names can be changed and the standards re-applied quickly and efficiently throughout a design.

SQL Developer Data Modeler allows you to set and apply naming standards to your diagrams and generated code. This white paper sets out the various naming standards options available and how to work with design rules to enforce them.

Overview

There are a few areas we'll look at in this document with regard implementing naming standards. Initially we look at defining and setting a set of templates to enforce the naming standards for keys, constraints and indexes. You can apply these to a complete design or part of a design at any stage of development. The tool does not automatically enforce these template driven standards; instead, the developer drives this aspect of the implementation naming standards, by first setting them and then applying them.

You can define naming patterns for entities, attributes, tables, columns and domains. Together with a defined glossary of permitted and classified words and abbreviations they constitute a naming standard used in name translation and name validation process. Apart from naming standards, the name abbreviation tool can be used to apply name abbreviations to elements in relational model.

SQL Developer Data Modeler provides and uses different elements when applying naming standards to models. You can define and then apply these standards to the different models available in SQL Developer Data Modeler. Used together, these elements help you to use a wide range of naming standards and restrictions in your models.

We'll group these elements into three parts:

- Definitions
- Design rule validation
- Name transformations – both for translating and formatting names

This white paper covers the following aspects:

- Name Templates
- Glossary Support
- Abbreviation Support
- Name Translation

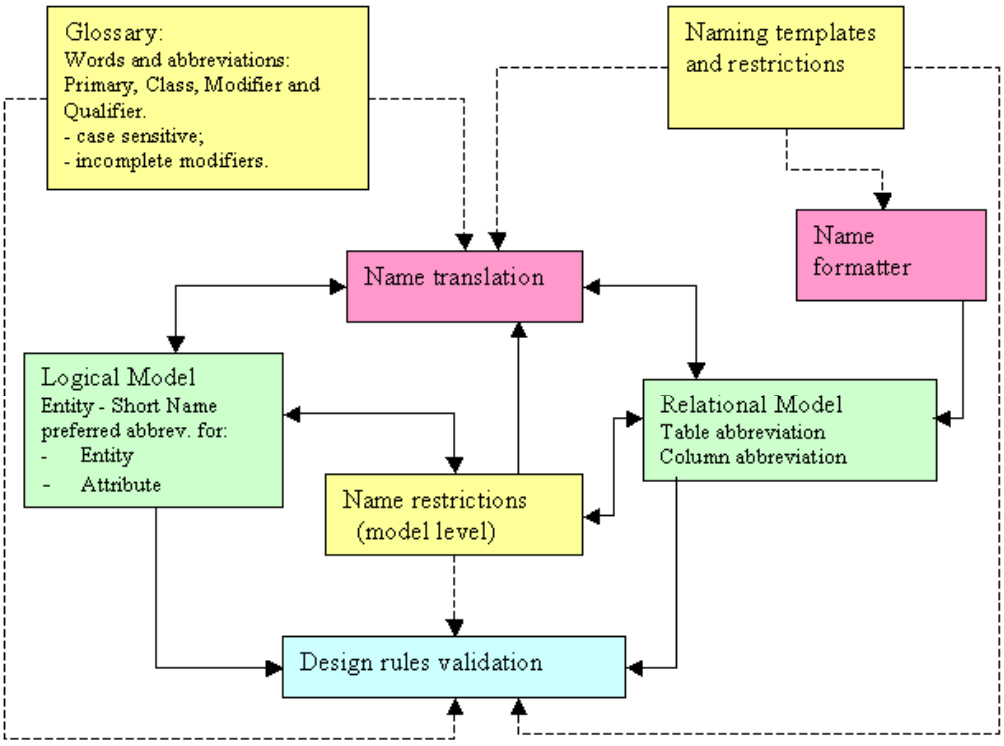

Figure 1: SQL Developer Data Modeler Naming Standard Elements

Figure 1 gives an overview of the different naming standard elements provided by SQL Developer Data Modeler and where they are applied. For example, you can create a template or name pattern and then apply this pattern to the relational model to update the table or view names in the model.

Defining Templates for Keys, Indexes and Constraints

You can define templates (name patterns) for keys, indexes and constraints, using combinations of predefined variables, using the **Tools > General Options** menu. Expand **Naming Standard** in the tree and select **Templates**. You can define the name pattern for each of the following elements:

- Primary Key
- Foreign Key
- Check Constraint
- Unique Constraint

- Index
- Column Check Constraint

For each element you can use a set or predefined variable to set the pattern. The predefined variables available include the following:

- {table}
- {table abbr}
- {child}
- {child abbr}
- {parent}
- {parent abbr}
- {column}
- {column abbr}
- {ref column}
- {ref column abbr}
- {seq nr}
- {model}
- Alphanumeric constants

Using a combination of these, and optionally the SUBSTR function with the syntax SUBSTR (index, length, direction, expression), you can build up a naming pattern for these elements.

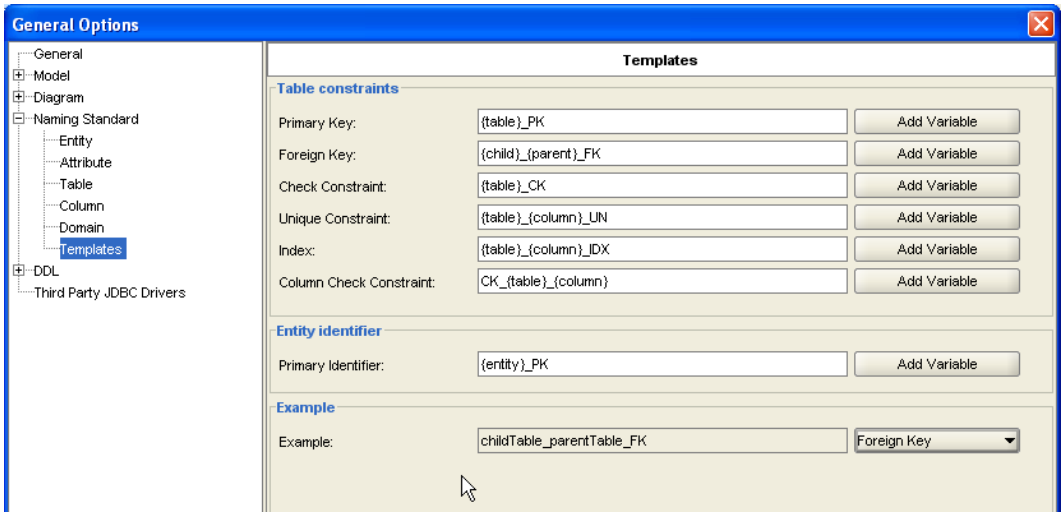

Figure 2: Setting the Templates for Keys, Constraints and Indexes

Figure 2 displays a set of possible patterns for the Keys, Constraints and Indexes in the relational model.

The examples below use SUBSTR, the table **ADMIN** and relational model **ORACLEDEMO**.

SUBSTR (7,4,FRONT, {model}) – > DEMO

SUBSTR (1,3,FRONT, {table}) – > ADM

SUBSTR (1,3,FRONT, TABLE) – > TAB (Where "TABLE" is a constant not a variable.)

The following formula,
IX_SUBSTR (7,4,FRONT, {model})_SUBSTR (1,3,FRONT, {table})_{seq nr}, produces IX_DEMO_ADM_1, for the first index of table ADMIN.

Applying the Templates to the Relational Model

Once you have defined the general pattern for these objects, you can apply the pattern to all objects in the relational model or to individual tables in the model. To apply these standards to an entire relational model, select the model in the browser and invoke the context menu. Select **Apply Naming Standards to Keys and Constraints**. This displays a dialog that allows you to specify the elements you want to apply the standards to. The context menu is displayed below.

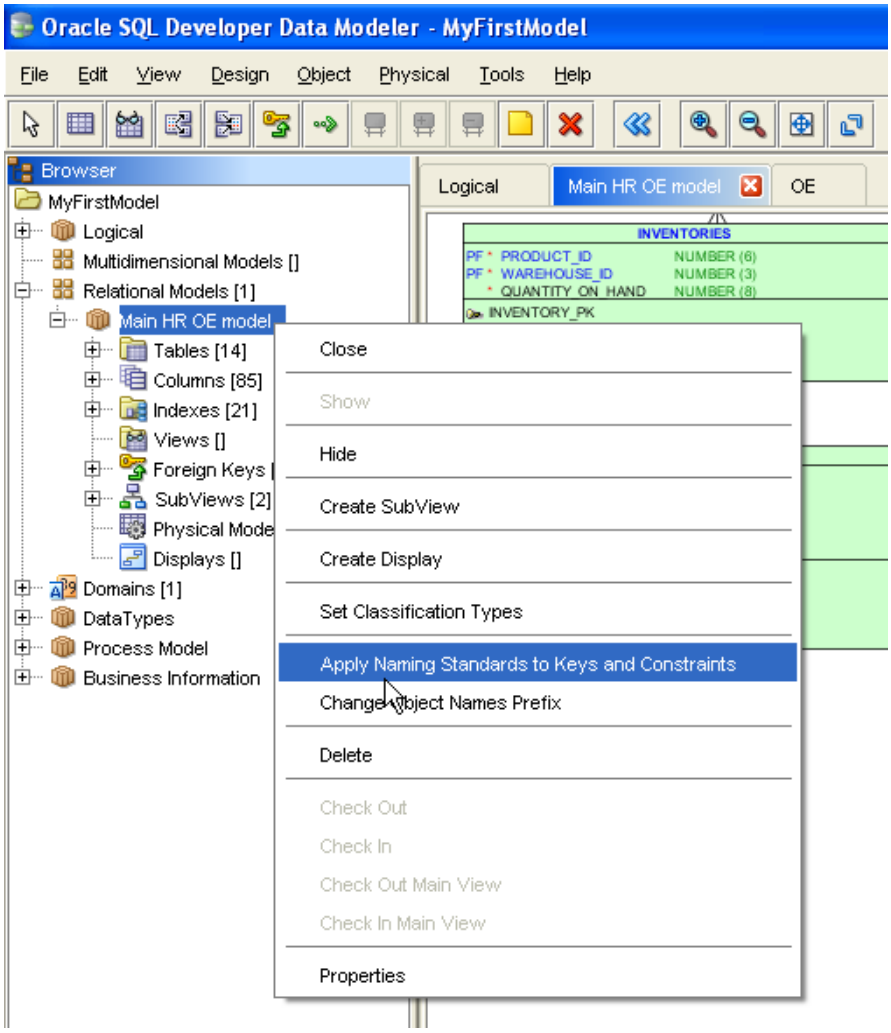

Figure 3: Apply Naming Standards to Full Model

To apply the standards to an individual table, invoke the table dialog and select the Naming Standards button. This invokes the same dialog as before, allowing you to select the elements you want to apply the standards to.

Prefix Management

Very often prefixes are introduced to the name of objects in order to represent different aspects of their life cycle, ownership or usage. There are two alternatives available:

- Permanently changing the object name - in some cases, you may want to replace or add a prefix to objects in the design. You can apply these changes to tables or views, (columns and indexes also supported) represented in a specific subview or to objects in

whole relational model – you can either add a new prefix or replace an existing prefix. You can also relate a prefix to defined classification types. Use Tools > General Options > Diagram > Classification Types

- Temporarily change the name of the object when you generate the DDL script. In this case define the old_prefix and the new_prefix and then apply the name substitution during the DDL generation. This approach has no impact on the names of the objects in the models.

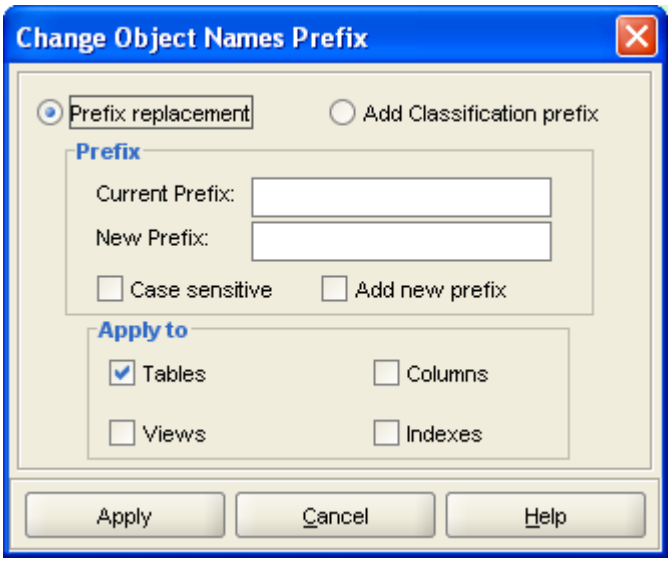

Figure 4: Replacing or adding prefixes to objects names

Setting Name Patterns for Elements

You can also control naming standards for entities, attributes, tables, columns and domains by setting design patterns. This pattern provides a structure for each element name. You define or set the pattern by specifying an unrestricted sequence of allowed word types and whether they are optional or mandatory. SQL Developer Data Modeler supports four word classification types:

- Prime word
- Class word
- Modifier
- Qualifier

Defining Word Classification Types

The following word classification types are also used by various data modeling tools in the industry and are defined below with examples.

Prime Word: The prime word identifies the object or element being defined. Typically, these objects represent a person, place, thing, or event about which an organization wishes to maintain information. Prime words may act as primary search identifiers when querying a database system and provide a basic list of keywords for developing a general-to-specific classification scheme based on business usages. CUSTOMER in Customer Address is an example of a prime word.

Class Word: A class word is the most important noun in a data element name. Class words identify the use or purpose of a data element. Class words designate the type of information maintained about the object (prime word) of the data element name. ADDRESS in Customer Address is an example of a class word

Modifier: A modifier gives additional information about the class word or prime word. Modifiers may be adjectives or nouns. DELIVERY in Customer Delivery Address is an example of a modifier. Other modifier examples: ANNUAL, QUARTERLY, MOST, and LEAST

Qualifier: A qualifier is a special kind of modifier that is used with a class word to further describes a characteristic of the class word within a domain of values, or to specify a type of information that can be attached to an object. Examples: FEET, METERS, SECONDS, and WEEKS.

Supporting Naming Patterns with SQL Developer Data Modeler

To set the name patterns for entities, attributes, tables, column and domain, invoke the **Tools > General Options** and expand **Naming Standard** in the tree. You can set the patterns for each of the individual element types.

Figure 5 illustrates an example pattern for attributes. Here we define the name structure with one optional modifier, one mandatory prime word, one optional modifier and a mandatory class word. E.g. Permanent Employee Average Salary.

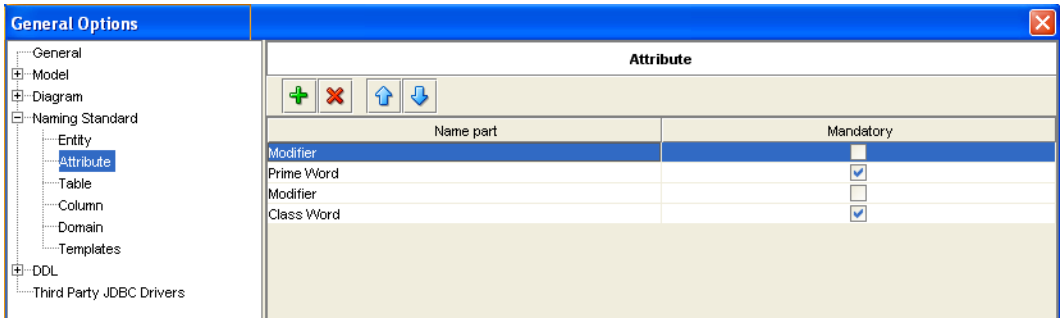

Figure 5: Naming Pattern for an Attribute

Reviewing Separator Settings for Elements in a Logical Model

In addition to defining the parts that make up an element in the Logical model, you can define the separator setting as part of the name pattern. There are three available options, as shown in Figure 6. The options are:

- **Space:** Used as a separator for the different parts of the name
- **Title Case:** Here the separator is not a break between each part of the name, such as a hyphen, space or underscore; instead, it is a capitalized first letter of each name part within a single word. (Also referred to as CamelCase.) E.g. GovernmentAccounts or PayableCodeIndicator
- **Character:** Define a single character as separator between the various parts of the name. You can also use this to fill in spaces by not providing a character at all.

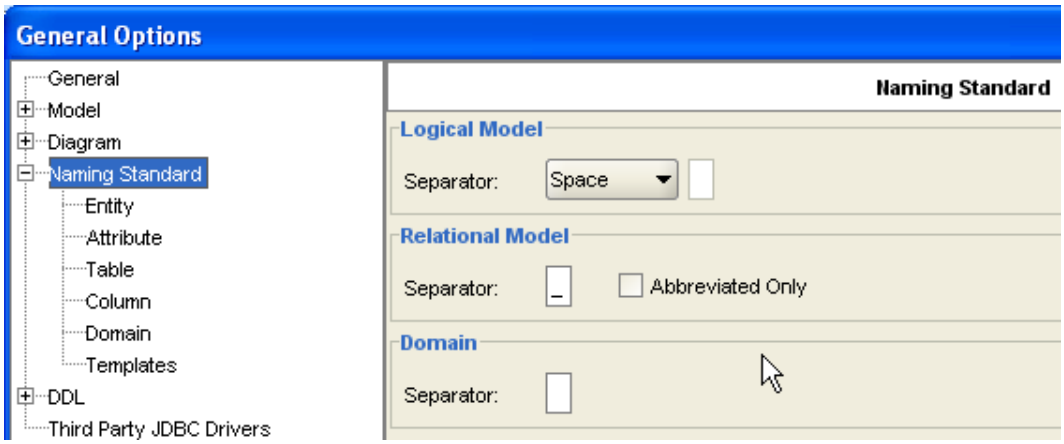

Figure 6: Additional Settings for the Logical and Relational Models

Reviewing Additional Settings for Elements in a Relational Model

There are additional settings that you can set up for the Relational model when defining the patterns, which differ from those you can set in the Logical model. The options are as follows:

- **Separator**: Use one character, such as an underscore, space or even no character between parts of the name. Note that the name validation could fail if you don't use a character. This is because non-unique abbreviations and abbreviations can be represented as the concatenation of other abbreviations.
- **Abbreviated Only**: Leave this option unchecked in order to allow non-abbreviated words to be used as parts of names.

These additional settings are illustrated in the Figure 6 above.

Using a Glossary

SQL Developer Data Modeler Naming Standards work on the underlying assumption that all terms (parts) used in names are defined in a glossary. The glossary is used during the validation and name translation process. If there is no glossary defined then no translation takes place.

Defining Glossaries

You can define one or more glossaries as validation glossaries. If there is more than one glossary, then a name is considered to be valid if it can be validated using any one of the defined glossaries. You can use different glossaries to represent separate domains (areas) of interest. However, using many glossaries together can lead to unpredictable results, especially when abbreviations are used in the validation process. E.g AP could be "Accounts-Payable", but also can match to "Actual Placement" if defined in another glossary. To add more glossaries used in validation process, select the Add button and include any additional glossaries.

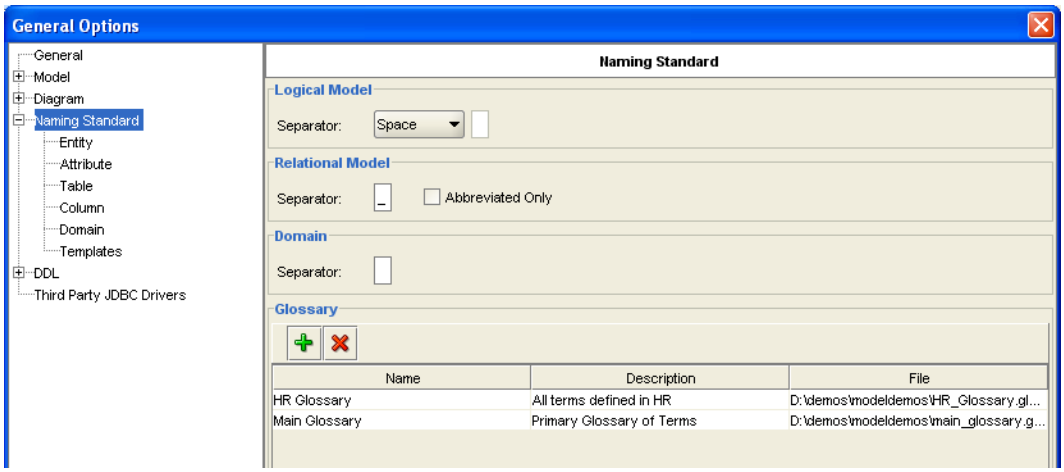

Figure 7: Glossaries used in the model for name validation

Using the Glossary Editor

You can make use of new or existing glossaries by adding them to the Naming Standard list. SQL Developer Data Modeler provides the facility to create new glossaries using the Glossary Editor. To invoke the editor, select **Tools -> Glossary Editor**. Using the editor you can import or define your own glossary with your own unique terms, using single or multiple words, and classify each term as a Primary or Class word, a Modifier or a Qualifier or any combination of these. You can also define an abbreviation, alternate abbreviation and a short description for each term.

When using the editor, you can build a glossary from scratch, or import and modify definitions from another SQL Developer Data Modeler glossary file.

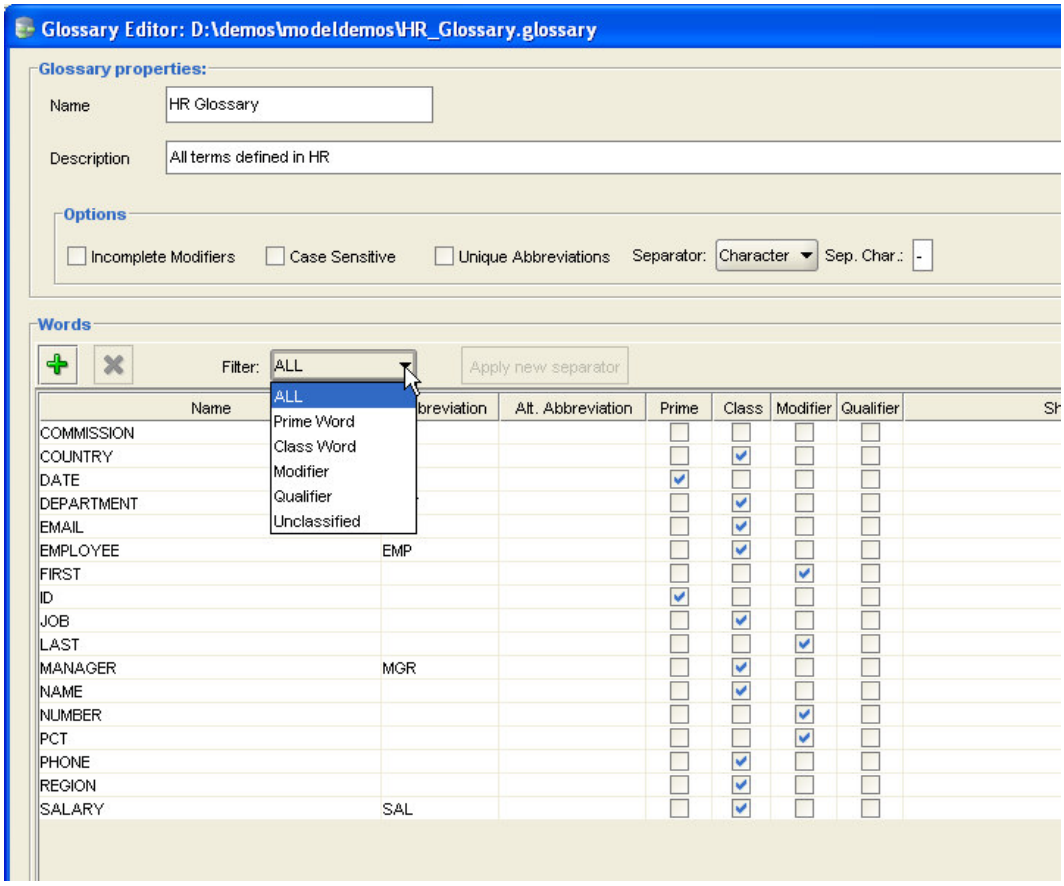

Figure 8: Glossary Editor

In addition to a name and a description, you can also set the following options:

- **Incomplete Modifiers:** The base assumption is that all terms used in names must be defined in a glossary. This option, if checked, means that it's not mandatory for modifiers and qualifiers to be defined in the glossary. This means that the name validation succeeds if the name parts, which cannot be found in the glossary, correspond to a modifier or qualifier in the name structure.
- **Case Sensitive**: This defines whether validation is case sensitive. E.g Code and CODE are different when "Case Sensitive" is checked.
- **Unique Abbreviations**: The uniqueness of abbreviations is not forced, thus one abbreviation can be used for all forms of a single word. E.g ADMIN = Administrator, Administration, Administrative or you can have three terms with same abbreviation. These definitions should be maintained carefully because name validation and name translation will return the correct result only in the case when all terms have the same classification settings. A report with the following information is displayed when the status is changed from unchecked to checked:

  - Non-unique abbreviations
  - Alternate abbreviations
  - Words without abbreviations
- **Separator**: This defines word separators for multi-word terms. Separator settings are checked when the glossary is loaded into the glossary editor. If the separator is not a space character, then a report showing all terms with a space in the name is displayed. There is also an option to replace the space with a defined glossary separator
- **Apply new separator**: This replaces existing separators in multi-word terms.

Importing Glossaries

SQL Developer Data Modeler allows centralized maintenance of glossaries and their usage. It also supports synchronization in various places, which is managed by importing glossaries. You can import an updated version of the original glossary to synchronize changes. The new definitions are added to the existing glossary and updates to the previously imported definitions can either be applied or skipped. Figure 9, below, illustrates the dialog for importing a glossary.

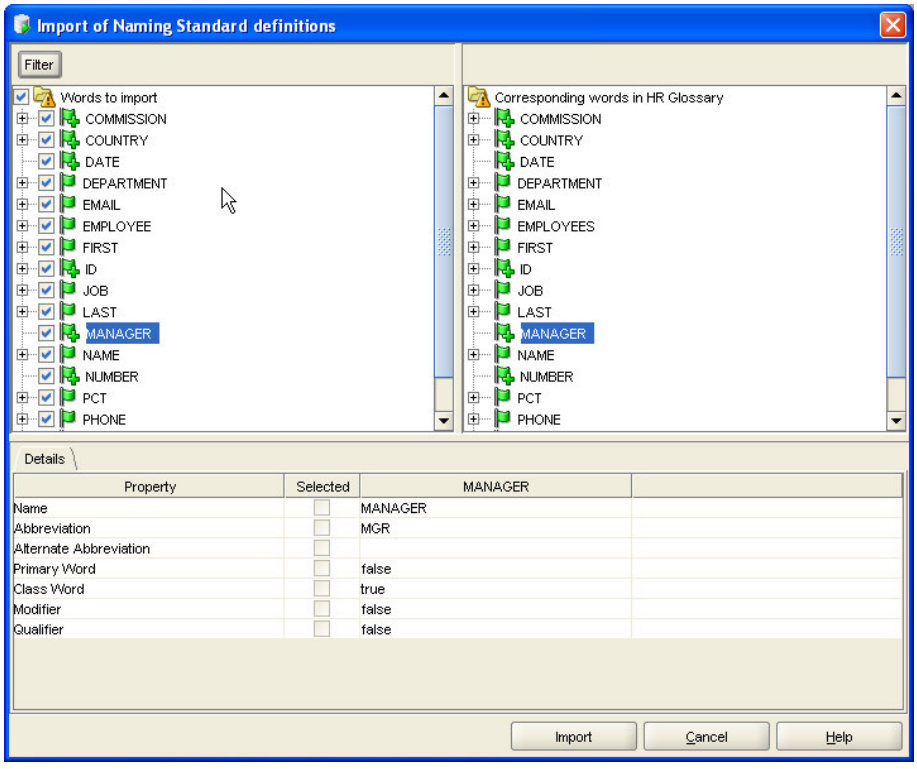

Figure 9: Importing Naming Standard Definitions

Importing a CA Erwin Data Modeler Naming Standard Glossary

You can import CA ERwin Data Modeler glossaries that were exported in .csv or .txt format. CA ERwin Data Modeler classifications "Modifier1" and "Modifier2" are transformed into SQL Developer Data Modeler modifier during import.

Using Abbreviations

You can define "Preferred abbreviation" at attribute and at entity level. These abbreviations can then be used when an attribute or entity name is transformed to a column or table name.

You can also define "table abbreviation" and "column abbreviation" , as shown in the image below, at table and column level. These are used in the templates when setting naming standards in a relational model.

A "Short Name" can be defined for an entity. This is synchronized with the "table abbreviation" of the related table during the forward engineering process.. You can also exclude this from the engineering process using the "Compare/Copy options" in the engineering dialog.

Note: During an import from the Oracle Designer repository, the entity "Short Name" in Oracle Designer is used for the entity "Short Name" and into the "Table Abbreviation" of the mapped table in Oracle SQL Developer Data Modeler. The Oracle Designer "Plural" property for the entity is used in the SQL Developer Data Modeler "Preferred Abbreviation" property of the entity.

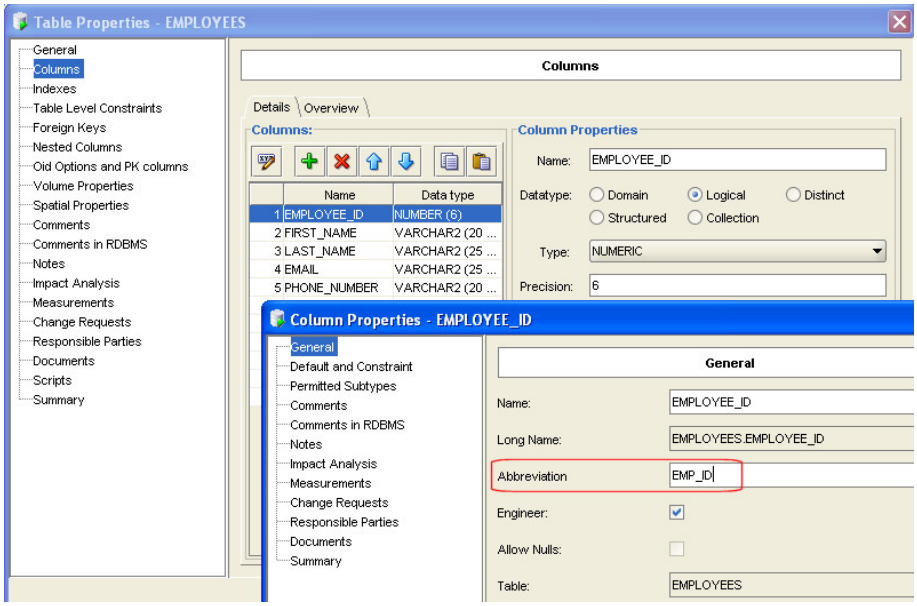

Figure 10: Setting Abbreviations

Using the Name Abbreviations Utility

SQL Developer Data Modeler provides a separate utility that allows words used in the Name field to be replaced with their defined abbreviations. Matching the word to the abbreviation is defined using a simple .csv file (see below). You can use this to replace names for tables, views, columns and indexes. This replacement can also be done in the opposite direction, where the abbreviation is replaced by a complete word. The figure below shows the dialog for importing an abbreviation file. The dialog allows you to select the direction and scope (target) of the transformation. . The transformation is applied to the name of tables, views, columns and indexes when "All Objects" is selected as the scope. The transformed name of table or column is stored in the "Table Abbreviation" or "Column Abbreviation" property of the respective object if "Abbreviations" is selected as scope.

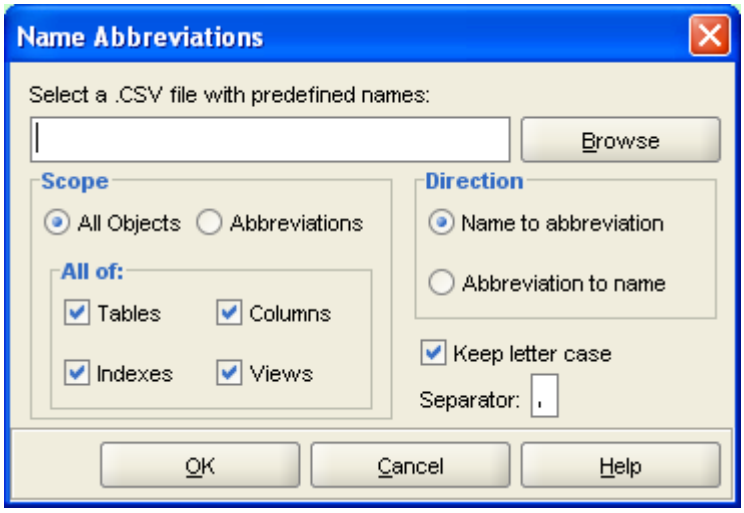

Figure 11: Name Abbreviations

The figure below displays part of an abbreviation file. In this sample the file contains abbreviation-word pairs. It can be structured as list of word (name)-abbreviation pairs. The paring list is not related to the Direction property, shown in Figure11 above. By applying "name abbreviations" transformation in both direction should return you to initial state of the names if the abbreviations (and terms/names in file) are unique.

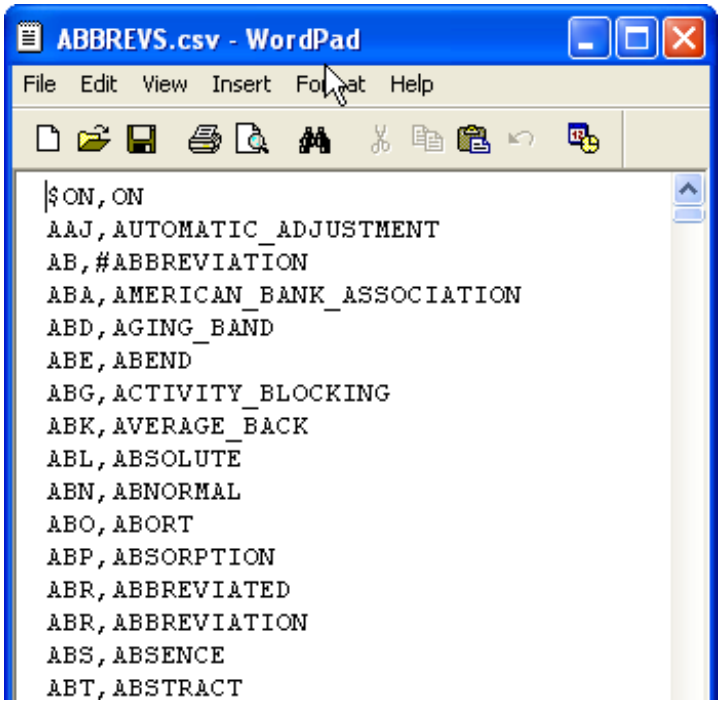

Figure 12: File With Abbreviation-Word Pairs

Name Translation

Names are translated from a normal name to an abbreviated name, or the reverse, when objects from the Logical model are transformed (forward engineered) to objects in the Relational model, or the reverse, when objects in a Relational model are transformed (reverse engineered) to a Logical model.

Glossaries defined in **Tools > General Options>Naming Standard** are used in the name translation process. The name translation, using a glossary is only applied for valid names. A name is considered to be valid:

- If there is a glossary and name patterns are defined, the following checks should succeed:
  - All primary and class words are defined in the glossary
  - All modifiers and qualifiers are defined in the glossary, if the glossary property "Incomplete Modifiers" is not checked.
  - All modifiers and qualifiers are not defined in the glossary and "Incomplete Modifiers" *is checked* in the glossary editor. Validation could succeed, but translation for name parts is only applied if there are corresponding words in the glossary, and abbreviations, or alternative abbreviations, are defined.
- If there is glossary defined, but no name patterns are defined. Missing name patterns mean that there are no restrictions on the name structure and name parts are validated on their existence in the glossary, assuming that the "Incomplete Modifiers" glossary property is not checked. If "Incomplete Modifiers" *is checked*, then the name is considered to be valid, and the name translation is applied for those name parts with corresponding entries in the glossary.

Note: Equal separators for multi-word terms that are defined in the glossary and for parts of names defined in name patterns, should be avoided because they could result in incorrect abbreviated names.

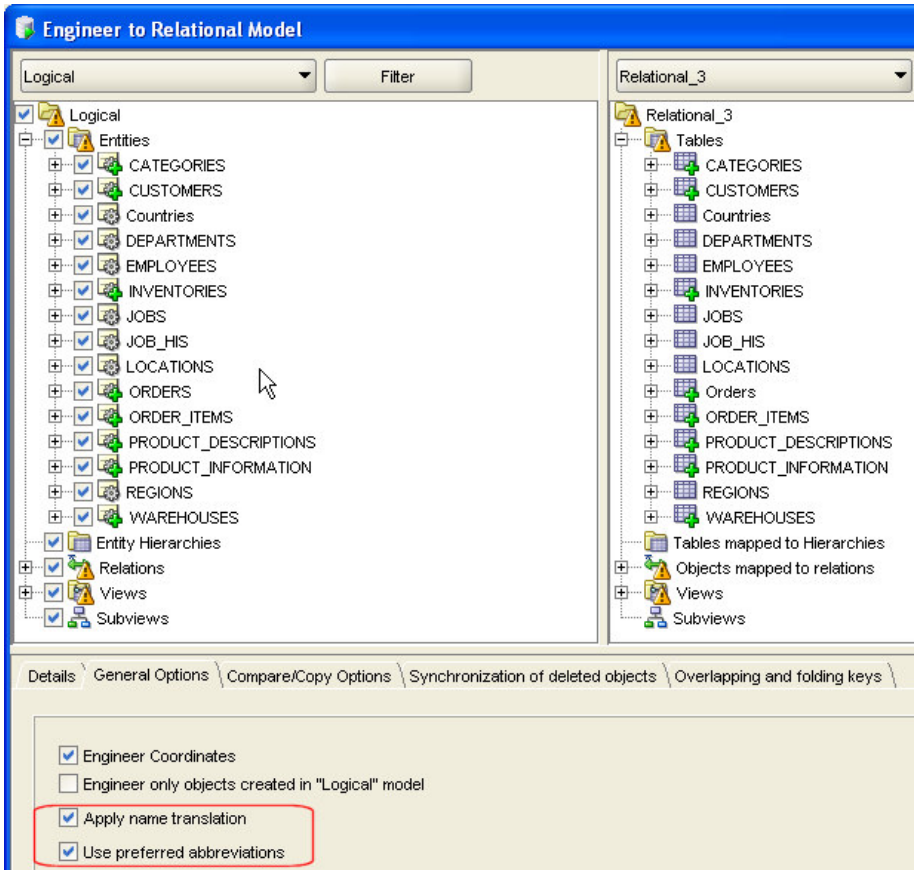

Figure 13: Engineering to Relational Model

Name translation comes into effect if you select "Apply name translation" in the engineering dialog. These are in addition to the glossary translation and differ, depending on whether you are doing forward or reverse engineering

In the figure above, there are two properties related to name translation in forward engineering from the Logical to the Relational model. These are

- Apply name translation
- Use preferred abbreviations – You can define a preferred abbreviation for entities and attributes

In contrast the property *Use preferred abbreviations* is not applicable for engineering from the Relational to the Logical model. The name of entity or attribute is not transformed using the glossary if there is preferred abbreviation defined for that entity or attribute.

Note: For users who use the singular term for entities, such as CUSTOMER, and then the plural for tables, CUSTOMERS, can use the abbreviations field in the attributes property palette and then check the two properties *Apply name translation* and *Use preferred abbreviations* when forward engineering from the logical model.

Model level restrictions are also used during name transformation:

- Max Name Length – used in conjunction with glossary – is used in the transformation from the Logical to Relational model. The name is first transformed using the abbreviations defined in glossary. If the new name doesn't comply with max length restriction then alternate abbreviations are used to construct the name.
- Character case setting – no case transformation is applied if "Mixed Case" is selected, otherwise the name is transformed to the selected case.

Setting Model Level Name Restrictions

In addition to the naming standards discussed above, you can set restrictions for various elements in the Logical and Relational models at model level. To set these restrictions, select the properties dialog for the model and expand the Naming Options node in the tree. For the Logical model you can set restrictions for Entities, Attributes and Views and for Relational Models you can set these restrictions for Tables, Columns and Views. The restrictions you can apply are

- Max name length
- Character case
- Patterns for valid characters

These restrictions are taken into account when applying the Design Rules before DDL generation.

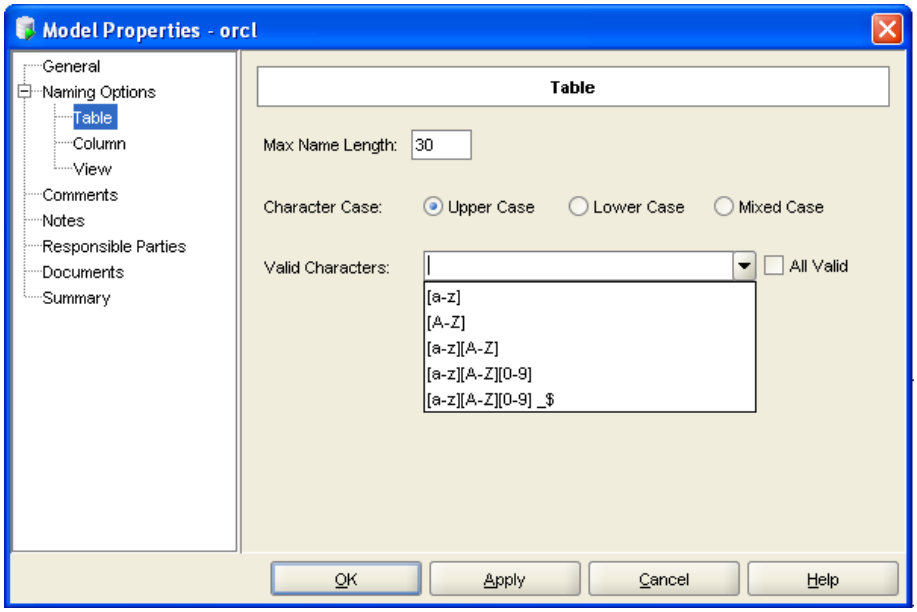

Figure 14: Model Level Name Restrictions

Using Design Rules Validation Related to Naming Standards

SQL Developer Data Modeler provides a set of predefined Design Rules. You can validate your models by applying these Design Rules to catch possible errors. Typically, you might run Design Rules before generating DDL scripts. When verifying the Design Rules, the following name standardization related checks are available:

- Matching name patterns and defined glossary terms
- Maximum name length at database level
- Model level name restrictions

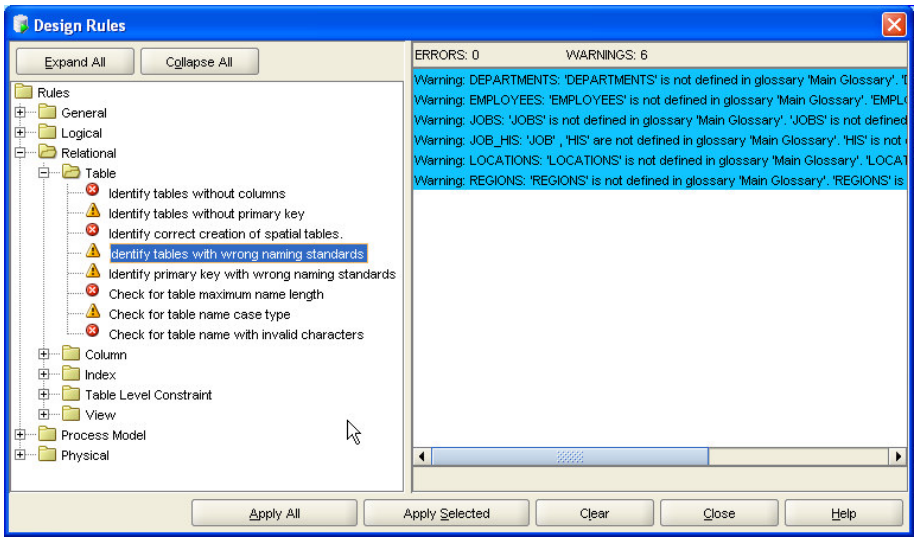

Figure 15: Applying Design Rules

In the figure above, the Design Rule "Identify tables with wrong naming standards" was selected. The warnings on the right show that some elements in the model are not defined in the Main Glossary, associated with the model.

Conclusion

Oracle SQL Developer Data Modeler provides a variety of features that allow users to define and set naming standards across a model, whether at the logical or relational level. You can define name patterns that are used during forward or reverse engineering and make use of glossaries and abbreviation files to aid the process. Even at the DDL generation process, Design Rules cross reference the naming standards set and will highlight inconsistencies. These features mean that models created can comply with company or industry standards as required.

Resources

For further information regarding Oracle SQL Developer Data Modeler, see Oracle Technology Network http://www.oracle.com/technology/products/database/datamodeler

SQL Developer Data Modeler
Naming Standardization
June 2009
Author: Philip Stoyanov, Sue Harper

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

0109