NORMA Multi-page Relational View

Matthew Curland

ORM Solutions, USA
mcurland@live.com

Abstract. The NORMA (Natural Object-Role Modeling Architect) tool has long supported a single-page view of the generated relational model. This view automatically shows one shape per table. Unfortunately, as the number of tables and foreign keys grows, this single-page approach becomes impractical. Therefore, for advanced users wishing to display readable relational models, it is necessary to move beyond this single-page approach. The NORMA *multi-page relational view* was created to address these concerns by allowing multiple relational diagrams in the same model. However, the multi-page environment has inherent issues that do not occur with a single-page approach. This paper discusses how we handle shape creation for tables and foreign keys with no corresponding display shape, how we display foreign keys when the referenced table is not on the diagram, and how column associations are displayed for foreign keys. We also discuss the display options used to balance explicit display of column information and complexity of the connecting lines. The result is a flexible system for rapid population of targeted relational diagrams that scales seamlessly for large models.

1 Introduction

One of the strongest features of Object-Role Modeling (ORM) methodology [1] is the ability to automatically generate a relational model from a conceptual ORM model. The Natural ORM Architect (NORMA) tool [2], which implements a version of ORM2 [3, 4], also has a feature to graphically display this relational model as a diagram. The *relational view* extension enables visualization of the relational form of the model and facilitates communication with database practitioners who might not be familiar with the ORM methodology and notation.

The relational diagram included in the open source version of NORMA automatically creates a shape for each relational table, places these shapes in a single diagram, and then connects these table shapes with directional arrows to indicate foreign key constraints. The user can manually arrange the table shapes on the single diagram to achieve a presentation that is reasonable for a small relational model, but the usability of the diagram rapidly deteriorates for models that produce a significant number of tables and connections. As a model approaches 100 tables the single-page view is overwhelmingly complex.

This paper will discuss the design of a multi-page relational view extension that allows users to spread their complex relational models across multiple pages. Section 2

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 487–491, 2013.

discusses the design approach for helping users manually populate table shapes and section 3 discusses available display options for foreign key connectors.

2 Populating Table Shapes

As with the single-page relational view extension, the multi-page relational view (MPRV) is added to a NORMA model using the *Extension Manager* dialog. When the single-page view is added, a diagram entitled *Relational View* is automatically created and populated with one shape per table in the generated relational model. The MPRV extension also adds a diagram called *RelationalPage1*, but no table shapes are placed on this diagram. Unlike the single *Relational View* diagram, the user is free to rename the relational page diagram, and add additional relational pages using the diagram tab context menu in NORMA. This behavior parallels the mechanism for adding multiple ORM diagrams to a model.

The primary impetus for the MPRV is to distribute table shapes across multiple pages, so automatically adding all table shapes to a newly created relational page is counterproductive. This means the user must manually add a table shape to one or more relational page diagrams for each table in the relational model. The direct way to do this is to drag a table from the *Relational Schema* branch of the *ORM Model Browser* and drop the table on a diagram. While this is a supported approach, there are several areas where the tool can help the user know when the model file contains a diagrammatic representation of all of the tables and foreign keys in their model.

1. The tool can track which tables have no shape on any MPRV diagram.
2. The tool can track foreign keys which are not displayed.
3. The tool can manage a validation error if the state of #1 or #2 changes.
4. The tool can simplify adding shapes for tables that are referenced by an existing table shape, but do not have a table on the same diagram.

The MPRV accomplishes the first three items by introducing a *group type* that automatically populates a *group* with undisplayed tables and foreign keys. A group in NORMA is a collection of references to elements defined in the model. Grouping allows the user to relate elements of the same or different types for arbitrary conceptual reasons. For example, a user may create a group named *Under Discussion* and populate it with references to fact types and constraints that require additional analysis. While this freeform use of groups is valuable, the true power of group comes when a *group type* is associated with that group.

A group type associates behavior with elements in a group, or automates the population of the group. An example of a behavior associated with a group type is shape coloring, whereby user-specified colors can be associated with elements explicitly included in the group by the user, or automatically by a group type. For the MPRV, we create a group type that automatically causes the group to reference tables and foreign keys with missing shapes (see Figure 1). The behavior of the group type is as follows:

1. When the MPRV extension loads for the first time in a model, create a new group called *Schema 'SCHEMANAME' Missing Shapes* and associate the *Relational Shape Missing* group type. Note that a group with this group type could also be manually added or deleted by the user at any time.
2. The group type populates the group by adding a reference to each table with no corresponding shape. The table reference can be dragged from the model browser onto an MPRV diagram to create a shape for that table.
3. The group includes a reference to a foreign key if a shape exists for the referencing table, but the referenced table does not have a shape on any diagram that displays a referencing shape. Dragging a foreign key onto an MPRV diagram creates a shape for either the referencing or referenced table: if a shape already exists for one table, then dropping the foreign key will create a shape for the other table. If neither table is shown on the drop-target diagram, then the referencing table is created and a drag action initiated to create the opposite shape.
4. Add a validation error if a group with a *Relational Shape Missing* group type has any elements in it. To minimize clutter in the *Error List* tool window, we create one error regardless of how many tables are missing shapes. Double-clicking the error opens the *shape missing* group in the model browser tool window and selects the node for the group type.
5. Explicitly deleting a table or foreign key from this group will add an 'x' over the model browser icon for that item and change the text to gray. This is a standard NORMA group feature used to enable the user to override automatically included items. The action can be reversed at any time with the *Include In Group* command, which is available on the model browser context menu for each explicitly excluded item.

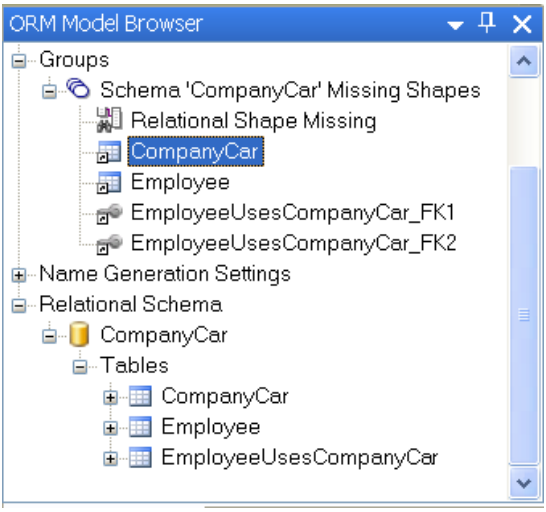

Fig. 1. The ORM Model Browser tool window displaying a *Relational Shape Missing* group for a small model. Comparing all tables and the group content indicates that the *EmployeeUsesCompanyCar* table is on an MPRV diagram with two unconnected foreign keys.

The *Relational Shape Missing* group type allows the user to easily track the remaining work required to diagrammatically represent the full relational model. However, it offers very little help in determining referenced tables that are not on the same diagram as a shape for a referencing table. To find referenced tables, the user would need to locate the table in the model browser, then expand that table and its *Foreign Keys* section, then drag those foreign keys onto the diagram. To fix this problem, the default MPRV display options show a stub foreign key connection arrow pointing away from the shape for each foreign key that references a table that has no shape on the diagram. Dragging these placeholder foreign key arrows has the same behavior as dragging a foreign key from the model browser—dropping creates the referenced table. Once all of the desired shapes are displayed, any remaining unconnected foreign key arrows can be hidden by adjusting the *UnattachedForeignKeyDisplay* option.

3 Foreign Key Connector Options

The single-page view connects tables related by foreign key constraints with lines that automatically route around intervening tables and have arrow heads on the referenced end. These lines can connect at any point on the connected shapes, so there is no graphical feedback to indicate the referencing and referenced columns. The lines also jump over each other when they cross. Jumping may be desired when lines are sparse, but jumps add a lot of graphical distractions when they are large or frequent. The MPRV display options default to no jumps. The per-model defaults for display options on all diagrams can be controlled from the expandable *DefaultDisplaySettings* property shown with each MPRV diagram. There is also a corresponding *LocalDisplaySettings* property that can override the default settings for each diagram.

The feature that has the most effect on the display of an MPRV diagram is the *ForeignKeyColumnDisplay* property, which determines whether the foreign key connection lines on a diagram should begin and/or end at the columns associated with that foreign key. See Figure 2 for connections displayed with different options. Anchoring keys to specific columns conveys significantly more information than a simple shape-to-shape connection, but also causes additional display issues.

1. Only the left and right sides of the table shape can be used for attach points because the top and bottom edges do not correspond to specific columns. The edge is dynamically determined based on relative table location. (The built-in engine provided by Microsoft determines the final routing.)
2. It is common to have multiple foreign key constraints pointing to columns from the same primary uniqueness constraint. This overlap causes multiple connector lines to point to the same location on a referenced shape.
3. It is much harder to get straight lines between shapes when the connectors have fixed attach points on both shapes.
4. It is possible to have the same unique columns be in the referencing table for one foreign key and the referenced table of another. In this case, as well as when where multiple foreign keys share columns, we use multiple vertical connection channels on the side of the shape to minimize overlap.

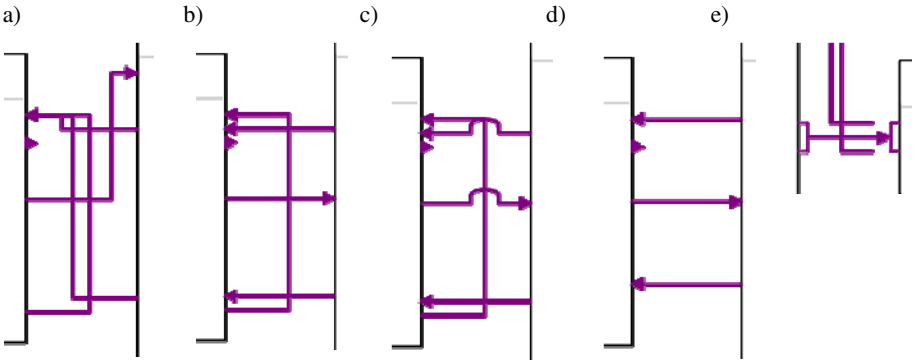

Fig. 2. Foreign key connectors between tables shapes. Items *a* through *d* display the same connections with different display options. The left table has one unconnected foreign key, one self-referential key, two incoming keys from the right table, and one outgoing key to the right table. Item *a* shows the default display (all columns anchored), *b* anchors only the referencing columns, *c* is the same as *b* with line jumps turned on, and *d* attaches with no column affinity. The self-referential constraint is not shown in *d* because it displays on the bottom edge of the left shape, far away from the other keys. Item *e* shows a two-column foreign key connector referencing a uniqueness constraint where both columns reference other uniqueness constraints, resulting in one paired incoming key and two individual outgoing keys for the right table. Note the bracketing used to graphically associate a foreign key with two or more columns. Hovering over any line highlights the entire line and displays a tooltip details of the foreign key.

4 Conclusion

This paper has provided a brief overview of the *Multi-Page Relational View* extension for NORMA. The implementation allows the user to rapidly add table shapes to multiple custom diagrams, track undisplayed tables, and configure the display options for each diagram. While this extension works side-by-side with the single-page relational view, it is expected that modelers in large systems will prefer the relational display and scalability offered by this extension.

The future work in this area includes additional display options for the interior of the table shapes, and possibly color support on the relational diagrams.

References

1. Halpin, T.: Object-Role Modeling: Principles and Benefits. International Journal of Information Systems Modeling and Design 1(1), 32–54 (2010)
2. Curland, M., Halpin, T.: The NORMA Software Tool for ORM 2. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBIP, vol. 72, pp. 190–204. Springer, Heidelberg (2011)
3. Halpin, T.: ORM 2. In: Meersman, R., Tari, Z. (eds.) OTM-WS 2005. LNCS, vol. 3762, pp. 676–687. Springer, Heidelberg (2005)
4. Halpin, T., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann, San Francisco (2008)