Relational Designer

Written by Noah Langenwalter

This requirements document was initially created in spring, 2007. The requirements were gathered from discussions with Kevin, Terry, and Matt and from researching other relational diagramming programs. My hope is that any new people attached to this subproject will be able to get an idea of what the relational designer does and how it works, along with what we’re planning to do with it.

The relational designer shows the user what his/her ORM diagram would look like as a database. Eventually, it will also be a way of customizing the database before you make it.

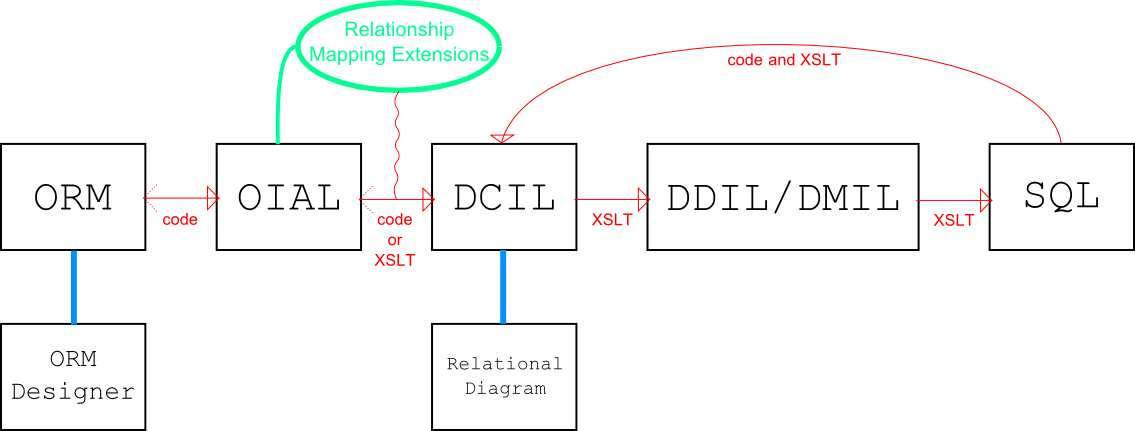
At this point, there are two options for how to build the relational designer: using the DSL Tools design surface or using WPF. The relational designer is currently drawn with DSL.

Whichever we choose, all of the data we want to keep about the relational diagram will need to be in an object model. The object model will be a representation of all of the elements of a relational diagram. DCIL, or Database Conceptual Intermediate Language, describes everything within a database we need (tables, columns, constraints, etc.) and will be used as a definition of our object model as well.

The information to create the object model is in OIAL. OIAL is a digested form of ORM. It was designed to serve as a preliminary layer for translating ORM to relational, UML, ER, etc. When an OIAL model is created, it can be translated into a DCIL object model for the relational designer.

This object model can then be used to show a diagram on a DSL design surface or in WPF.

To give an idea of where this fits, the diagram is a layout of the information exchange that underlies nORMa. This requirements document deals with only the relational diagram section.



Hopefully, all of that will give anyone assigned to this project at least an idea of the questions to ask.

The following requirements are not an exhaustive list of everything we can imagine we want the relational designer to be able to do, but they do neatly define the next big chunk of functionality.

Relational Element Attributes---

1. Model
   1. Name Prefixes for tables, stored procedures, triggers, etc.
2. Table
   1. Name
   2. Description
   3. Columns

(Name and Description are editable)

1. Column
   1. Attributes:
      1. Name
      2. Datatype
      3. Nullable
      4. Length
      5. Precision
      6. Scale
      7. Default Value
      8. Identity
      9. Identity Seed
      10. Identity Increment
      11. Row Guid
      12. Description
   2. All above attributes are editable.
2. Keys and Uniqueness Constraints
   1. Name
   2. Description
   3. Attributes specific to primary keys
      1. Columns
   4. Attributes specific to foreign key relationships
      1. Referencing Columns (tablename.columnname)
      2. Referenced Columns
      3. Referential Action
         1. Cascade
         2. Set Null
         3. Set Default
         4. Restrict
         5. No Action (default setting)
   5. Attributes specific to uniqueness constraints
      1. Columns

(Name and Description are editable)

General Interface---

1. Allow diagram to be split into multiple pages.
   1. Allow addition of tables (that already exist in the model) to pages.
      1. Allow tables to be added by dragging from the Model Browser.

LiveOIAL---

1. Tie model to LiveOIAL.
   1. Update model when LiveOIAL changes.
   2. Submit modifications made to the model to LiveOIAL so that they may be back-propagated.

Diagram Window---

1. Show a graphical relational model.
   1. Show Tables
      1. Show table name
      2. Show toggle to collapse all (except columns) compartments
         1. To avoid problems with attaching foreign keys to columns, the columns compartment will not be collapsible.
      3. Show columns
         1. Show column attributes (name, datatype, etc.)
            1. Make column attribute display optional.

Use template concept to manage display.

Allow creation of custom templates.

Allow assignment of a name for each custom template.

Allow selection of attributes to be included in collection.

Allow modification of custom templates.

Allow edit of name.

Allow modification of attribute list.

Provide pre-made templates for common configurations.

Allow application of a template to:

One table.

Multiple tables.

All tables.

Save custom templates so that users can use them across sessions.

* + - 1. Allow column order to be changed.
    1. Show table keys (and uniqueness constraints) on columns
       1. Show keys next to columns (PK, U, FK)
          1. Tooltip shows name(s) of key(s).
          2. Show number next to Us and FKs to identify them across columns. (FK1, FK2, FK1)

Show order of columns in constraint

(Table1

FK1.1 ColA

FK1.2 ColB

Table2

FK1.2 ColB

FK1.1 ColA)

* + - * 1. Show a glyph for each key on a column.

Display key glyphs in order: PK - U - FK

* + 1. Show table keys separately
       1. Show key attributes (name, associated columns)
       2. Show toggle to make keys compartment collapsible.
    2. (Possible other compartments for constraints, triggers, etc.)
  1. Show foreign keys between tables
     1. Attach foreign key end-points to specific columns
     2. Show direction of reference
     3. Tooltip shows referenced columns (Users.UserID = Posts.User)
     4. Make lines movable (Possible requirement)

1. Make graphical elements selectable:
   1. Table
   2. Collapse/Expand All Toggle (again, this is actually collapse all but columns)
   3. Column
   4. Column Attribute
   5. Compartment Collapse Toggles
   6. Key (in key compartment)
   7. Key Attribute
   8. Foreign Key Lines/Links

Populate property grid---

1. Show attributes of selected relational element(s).
2. Allow editing of attributes. (see note above for list of editable attributes)

Future/possible/tangential plans

iconOne of the things that would be cool to do, especially if we decided to use WPF, was to display an FK lollipop next to a table when the other table(s) in the relationship are not present on the diagram. Here’s what a lollipop would look like:

This table icon at the end of the lollipop could be selected/click-dragged to add that table to the diagram.