Microsoft's new database modeling tool: Part 2

Terry Halpin
Microsoft Corporation

Abstract: This is the second in a series of articles introducing the Visio-based database modeling component of Microsoft Visual Studio Enterprise Architect. Part 1 discussed how to create a new ORM source model, add sentence types, basic internal constraints and examples in the fact editor, drag fact types onto the drawing window from the business rules editor, and save the model. It also explained how to map an ORM model to a logical database model by creating a database model project, adding the ORM source model, and then building the logical model. Finally it showed how to generate the physical database model from the logical model by selecting the target DBMS and generating a DDL script. Part 2 discusses how to use the verbalizer, mark an object type as independent, objectify an association, and add some other ORM constraints to an ORM source model.

Introduction

Microsoft Visio for Enterprise Architects (VEA) is included in the Enterprise Architect version of Visual Studio. NET, currently in beta 2. Following the release of beta 2, a number of enhancements and bug fixes to the product have been made. The final release of VEA will be a superset of Microsoft Visio Professional 2002. As such, the product will include both the simple ORM drawing solution from Visio Professional, as well as the deep ORM source model solution that provides forward and reverse engineering to/from physical database schemas. In the help files for the product, the simple drawing stencil is referred to as the ORM diagrammer and the deep modeling solution is referred to as the ORM source model. The ORM source model can be used for both diagramming and engineering, and is the only ORM solution discussed in this series of articles. The help files were mostly omitted from the beta, but are now available as a web download.

This series of articles provides a simple introduction to using the Visio-based database modeling solution within Visual Studio Enterprise Architect. In this article, the emphasis is on the verbalizer, object independence, nesting and some other constraints in the ORM solution. Familiarity with ORM and relational database modeling is assumed. Overviews of ORM are downloadable [1, 2], and a thorough treatment of ORM and database modeling is discussed in my latest book [3].

The previous article discussed how to create a simple ORM model and map it to a logical and then a physical database schema [4]. In that article, I indicated that all the Visio stencils and templates come in both US units and metric versions. In the final release, only the version (US or metric) specific to your locale will be installed by default. If you want both versions installed, you should choose the custom installation option and indicate the solutions for which you also need the other version.

Verbalizer

Both the ORM source model and the logical database model solutions provide automatic verbalization of any part of the model that you select, including any examples that you may have entered. This feature is very useful for communicating the meaning of a model to non-technical domain experts. To illustrate this feature, let's open the sample Employee ORM source model that comes with the product. To open this model, choose File > New > Browse Sample Drawings then select the Database folder and the Employee ORM source sample file, and hit the Open button (see Figure 1).

The Employee page of the Employee source model should now appear. The full model is spread over three pages, called Employee, Project and Room. The name of the currently displayed page appears in a tab below the drawing window. By default, only the Database Properties and Business Rules windows appear below the drawing window. To open the verbalizer window, choose Database > View > Verbalizer from the main menu. This should now appear below the drawing window. Now use the mouse to select the part of the model you want verbalized. In Figure 2, I've selected a primary, external uniqueness constraint. The verbalization appears in the verbalizer window.

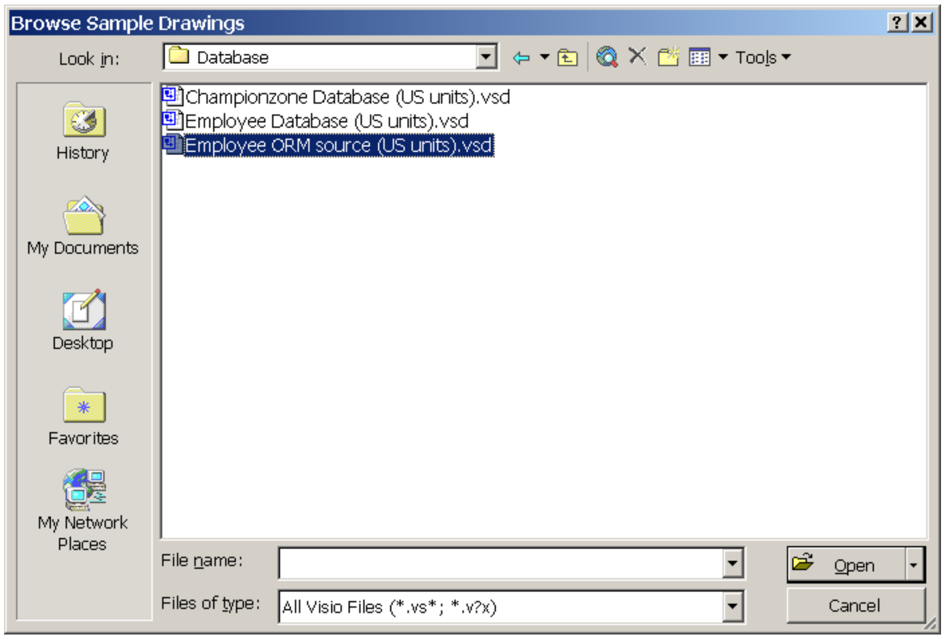
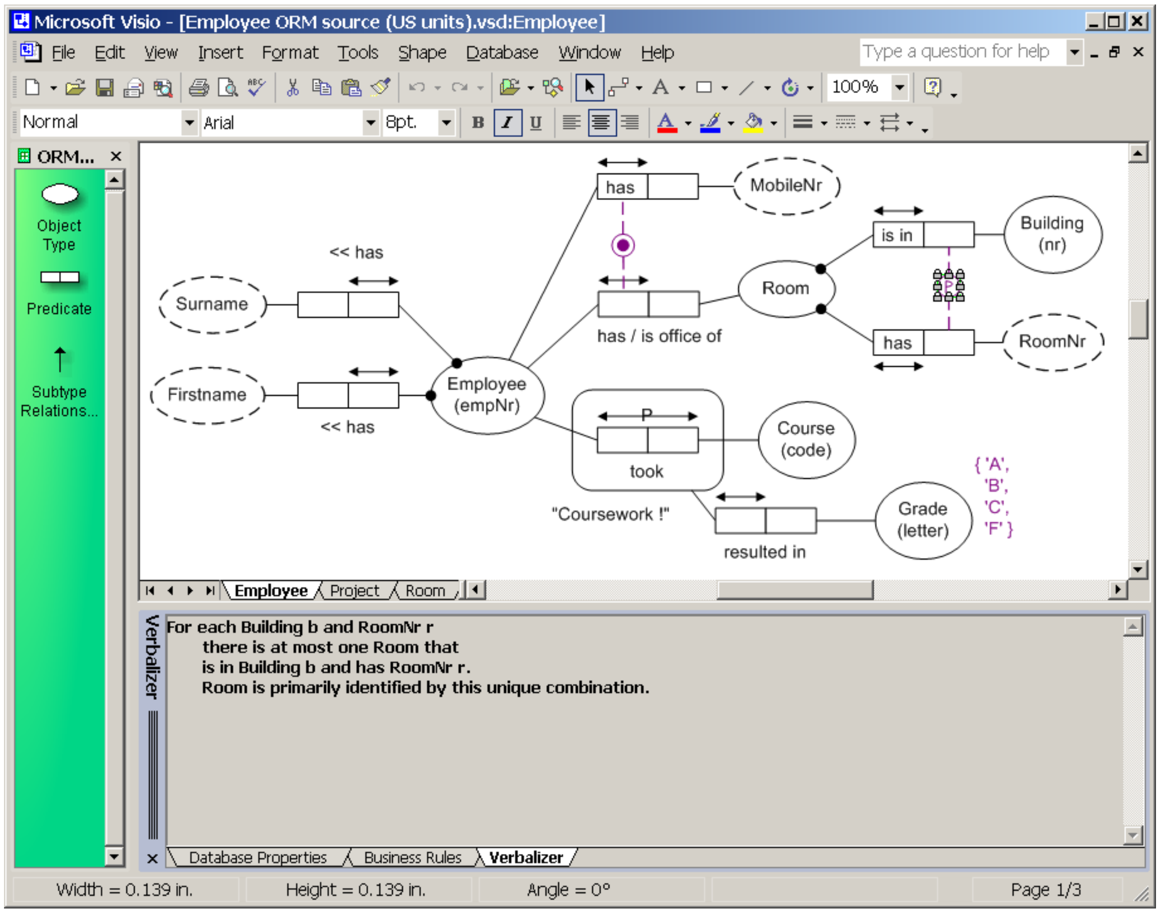

Browse Sample Drawings
Look in: Database
Tools
History
My Documents
Desktop
Favorites
My Network Places
Championzone Database (US units).vsd
Employee Database (US units).vsd
Employee ORM source (US units).vsd
File name:
Files of type: All Visio Files (*.vs*; *.v?x)
Open
Cancel

Figure 1 Opening a sample ORM Source model



For each Building b and RoomNr r
there is at most one Room that
is in Building b and has RoomNr r.
Room is primarily identified by this unique combination.

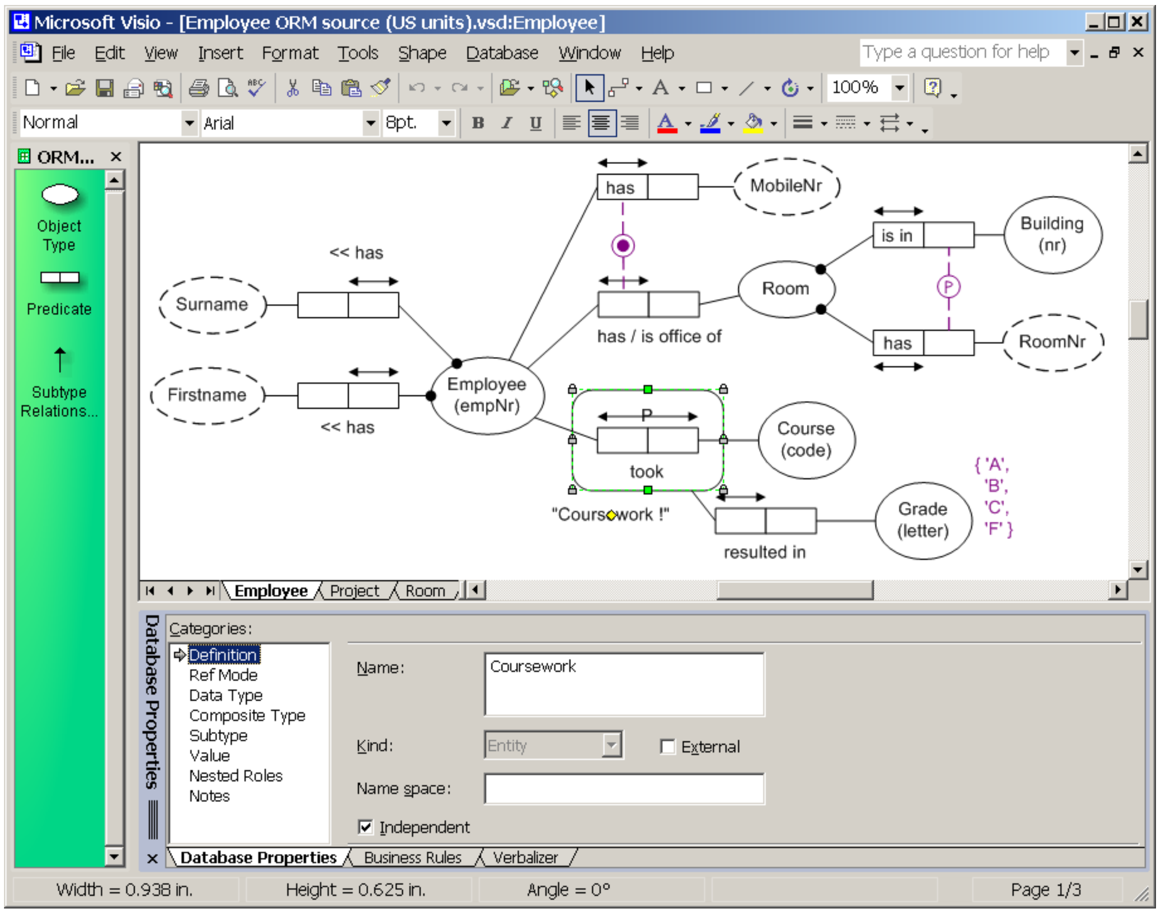
Figure 2 Verbalization of a primary, external uniqueness constraint

To select a single model element, simply click on it. To select an area of the model to be verbalized, hold the left house button down and drag the cursor diagonally over the area. All aspects of the model within that area will be verbalized (including fact examples if you have added them). The verbalizer window remains open until you close it. I usually leave it open. You can choose which of the windows below the diagrammer are displayed at any time by selecting the relevant tab at the bottom of the displayed window (e.g. Database Properties, Business Rules or Verbalizer).

Independent object types, and the database properties sheet

In Figure 2, the binary association Employee took Course has been objectified as Coursework. In this universe of discourse (UoD), we can know that an employee took a given course without knowing the rating he/she eventually gets for that course. Hence instances of Coursework can exist independently of whether they play any other role in the UoD. For this reason, Coursework has been marked *independent*, as displayed visually by the " !" appended to its name. This mark is entered automatically by the tool when you specify the object type is independent.

To specify that an object type is independent, you need to select the object type and make an entry in its *database properties sheet*. If the database properties sheet window is currently being displayed, all you need do is click the object type to view its properties. Otherwise, *double-click the object type* and this will display its properties sheet. The previous article discussed how to specify basic aspects of a single sentence type using the fact editor. The database properties sheet can be used to do anything the fact editor can do, as well as more advanced things. One of the extra things it provides is a check box for indicating whether an object type is independent (see Figure 3). Here the Definition pane of the sheet is displayed. Any other pane of the properties sheet can be displayed simply by selecting it in the Categories list.



Categories:
- Definition
- Ref Mode
- Data Type
- Composite Type
- Subtype
- Value
- Nested Roles
- Notes

Name: Coursework

Kind: Entity   ☐ External

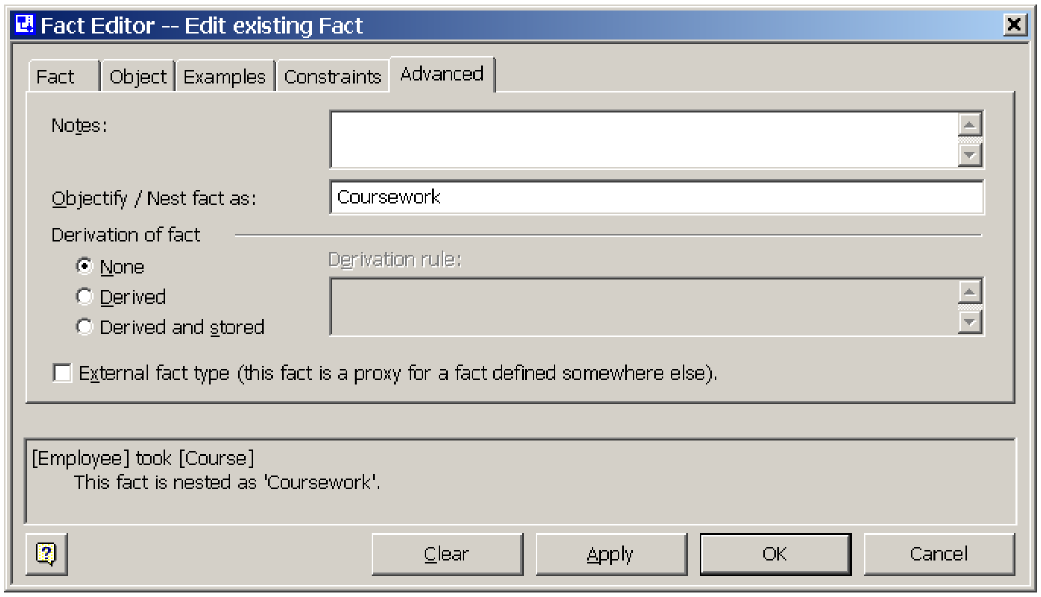
Name space:

☑ Independent

Figure 3 The independence of Coursework is specified on its database properties sheet

Objectifying an association (nesting)

Consider the association Employee took Course. To be able to record the grade (if any) that an employee gets for a given course, the association has been objectified as Coursework, and the fact type Coursework resulted in Grade has been added. The object type Coursework is said to be nested, since it nests an association inside it. Nesting is specified using the Advanced pane of the Fact Editor. If you have just entered a new fact type (e.g. Employee plays Sport) in the fact editor and the editor is still open, you can add the nesting before closing the editor. If instead you have a fact type on the diagram that you want to objectify, then first select the fact type and then open the fact editor (using Database > View > Fact Editor). Now select the Advanced tab and enter a name for the objectified association in the field labelled "Objectify / Nest fact as:". For example, you might objectify Employee plays Sport as Play. Figure 4 shows the nesting declaration for the Coursework association in the sample model.



Notes:

Objectify / Nest fact as: Coursework

Derivation of fact

[✓ None] Derived / Derived and stored

Derivation rule:

☐ External fact type (this fact is a proxy for a fact defined somewhere else).

[Employee] took [Course]
This fact is nested as 'Coursework'.

Figure 4 Nesting is specified using the Advanced pane of the Fact Editor

If the association is already on the drawing window, hitting the OK button in the fact editor causes the nesting envelope to be displayed around the association. A nested object type can also be displayed by dragging it out from the business rules editor. The name of the objectified association appears outside the nesting envelope (see Figure 3). You can reposition this name by selecting the nested object type, and then dragging its control handle (which appears as a small, yellow diamond). You can also resize the envelope vertically by dragging a shape handle (small green square).

ORM currently requires that each objectified association either has a spanning uniqueness constraint, or is a 1:1 association. This rule is enforced when a *model error check* is performed. You can perform a basic model error check at any time by selecting Database > Model Error Check from the main menu.

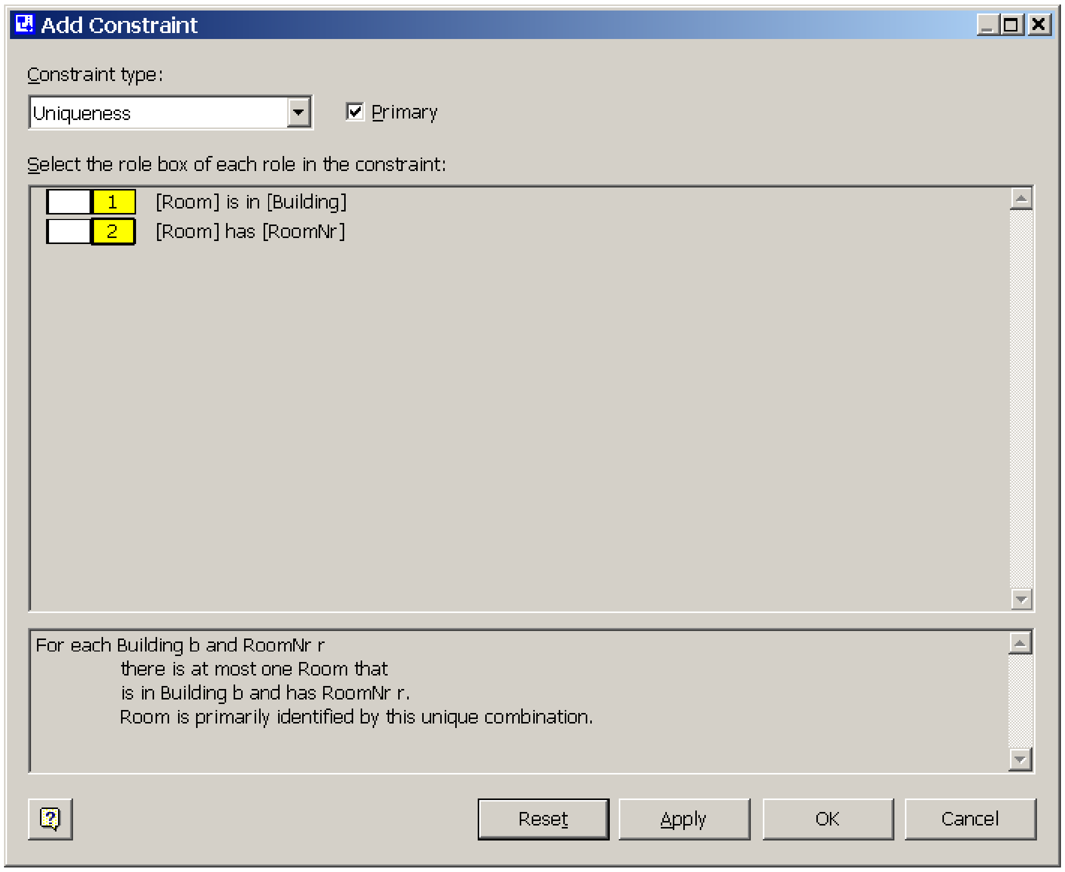
When you create a nested object type, the tool automatically creates derived predicates between the nested object type and the object types involved in its defining predicate. This provides a uniform way of navigating from any object type (nested or unnested) to the rest of the schema, and is mainly designed to facilitate conceptual queries. By default the derived predicates are named "involves" or "is involved in". If you wish, you may rename these predicates by double-clicking the nested object type on the diagram window to display its Database Properties Sheet, then selecting the Nested Roles category and renaming the nested role readings.

Adding external uniqueness constraints

Part 1 discussed how to add internal uniqueness and simple mandatory role constraints to a single fact type using the fact editor. Any other role-based constraint is best declared by selecting the relevant predicates in the drawing window, then using the Add Constraints dialog box, accessible from the right-click menu, or by choosing Database > Add Constraints from the main menu.

Let's use the sample Employee model to practise adding constraints. For each constraint, first delete it from the model (select the constraint then hit the delete key), and then add it back in again using the Add Constraints dialog.

Assuming the external primary uniqueness constraint in Figure 3 is deleted, you can add it back as follows. Holding down the Shift key, select the Room is in Building and Room has RoomNr fact types, then right-click and choose the Add Constraints option. The Add Constraint dialog box opens with the Constraint type set to Uniqueness. Check the Primary box to indicate this constraint provides the primary reference scheme for Room, and then select the Building and RoomNr roles (just click them in turn—there is no need to hold the shift key down). The constraint verbalization is shown in the lower section (see Figure 5). If the external uniqueness constraint is not used for primary reference, do not check the Primary box.



Constraint type: Uniqueness ✓ Primary

[Room] is in [Building]
[Room] has [RoomNr]

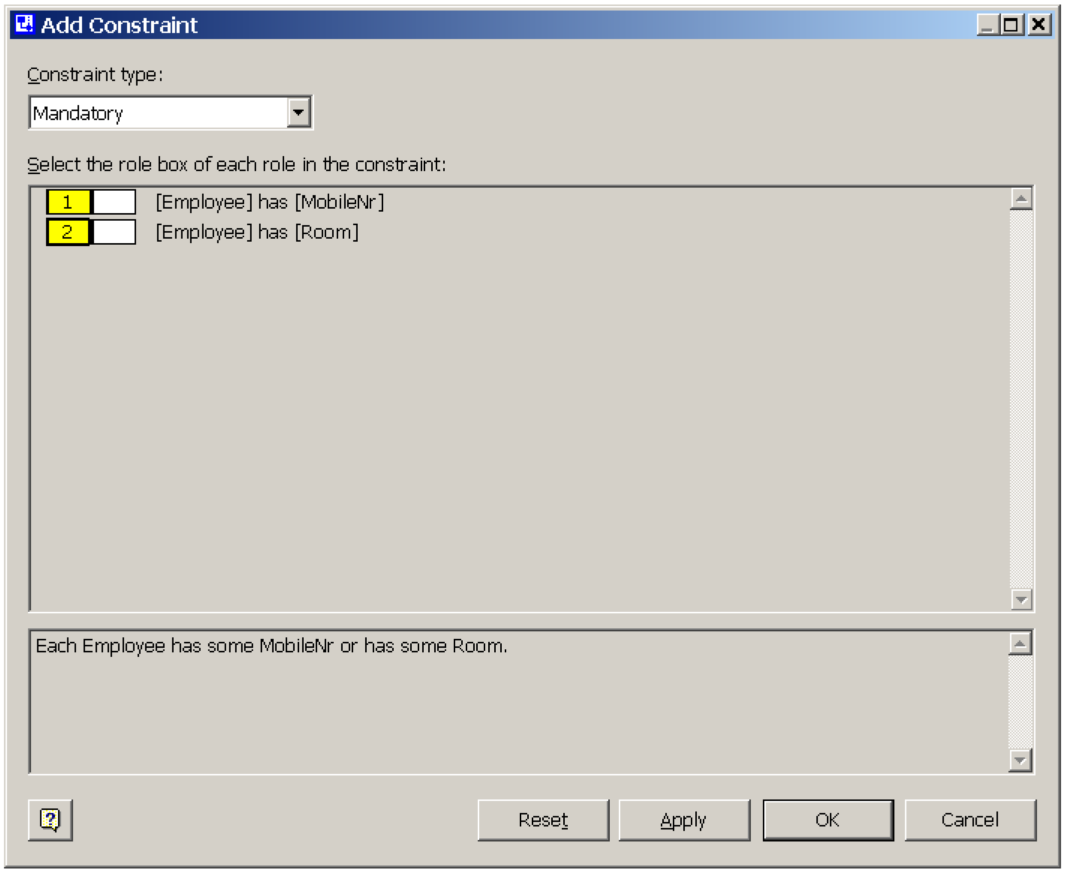
For each Building b and RoomNr r
there is at most one Room that
is in Building b and has RoomNr r.
Room is primarily identified by this unique combination.

Figure 5 Adding a primary, external uniqueness constraint in the Add Constraint dialog box.

If you hit the OK button, the dialog closes and the constraint appears on the drawing window. If you have other constraints to apply to one or more of these predicates, you can hit the Apply button to apply the constraint, leaving the dialog open for adding more constraints on those predicates.

Adding disjunctive mandatory role (inclusive-or) constraints

The circled dot in Figure 3 is a disjunctive mandatory role (inclusive-or) constraint, indicating that each employee must have a mobile number or a room (or both). For example, a contractor might have a mobile number but not a room, and a permanent employee might have a room (and perhaps a mobile number too). Assuming this constraint is deleted, we can add it back thus: select the Employee has MobileNr and Employee has Room fact types; right-click to choose the Add Constraints option; when the Add Constraint dialog appears, change the Constraint type to Mandatory; click the two employee roles (see Figure 6); hit the OK or Apply button.



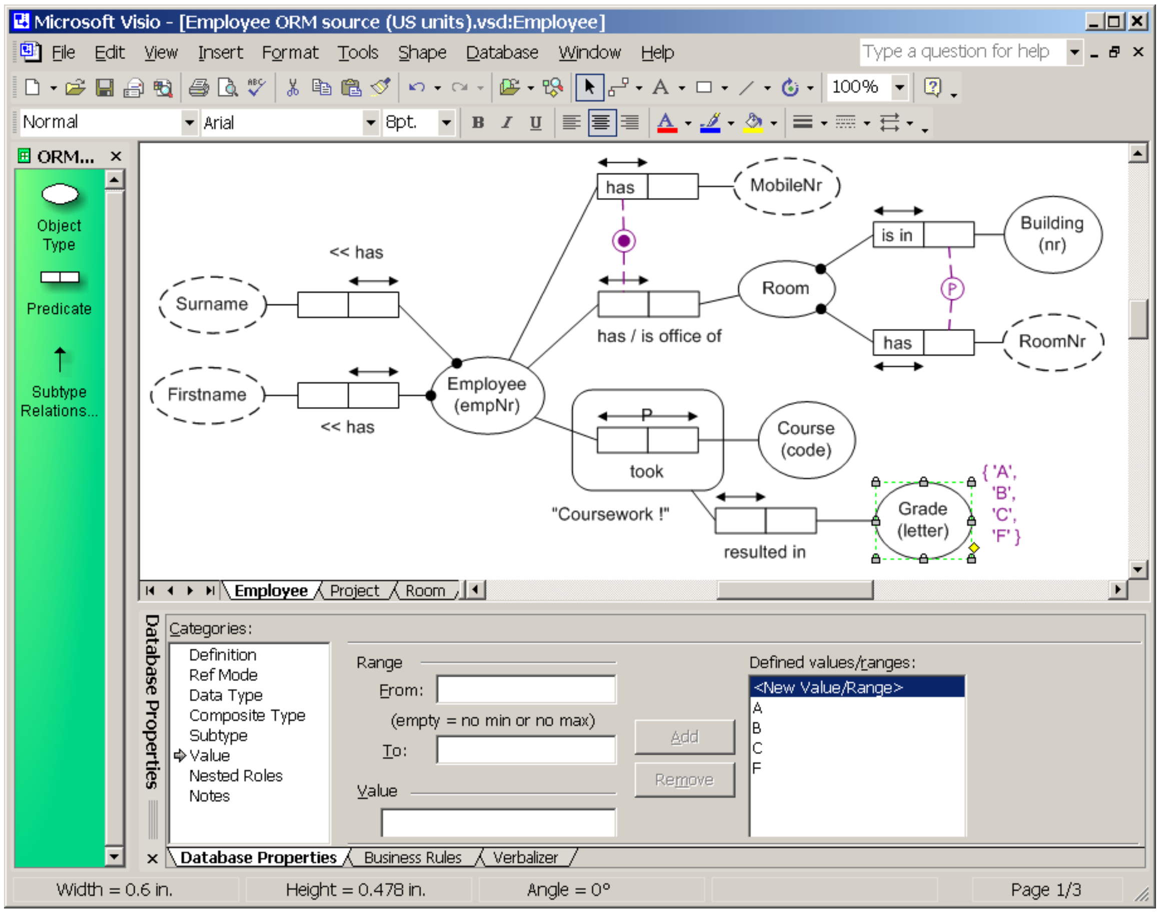
Each Employee has some MobileNr or has some Room.

Figure 6 Adding a disjunctive mandatory role (inclusive-or) constraint in the Add Constraint dialog box

The Add Constraint dialog can also be used to specify subset, exclusion, equality, frequency, ring, and index constraints (see Part 3).

Adding value constraints

Value constraints on object types are declared by clicking (or double-clicking) the object type in the drawing window to display its property sheet, selecting the Value category and then entering the relevant values and/or range(s) of values. In Figure 3, the Grade object type is restricted to the following set of possible letter values {'A', 'B', 'C', 'F'}. Assuming this constraint is deleted, you can add it back by entering the value A in the Value field (don't include quotes), then hitting the Add button, then repeating this procedure for B, C and F (see Figure 7). You can remove a value by hitting the Remove button. To reposition the value constraint on the diagram, select the object type (not the constraint) and then move the control handle (small yellow diamond) that appears.



Categories:

Definition
Ref Mode
Data Type
Composite Type
Subtype
Value
Nested Roles
Notes

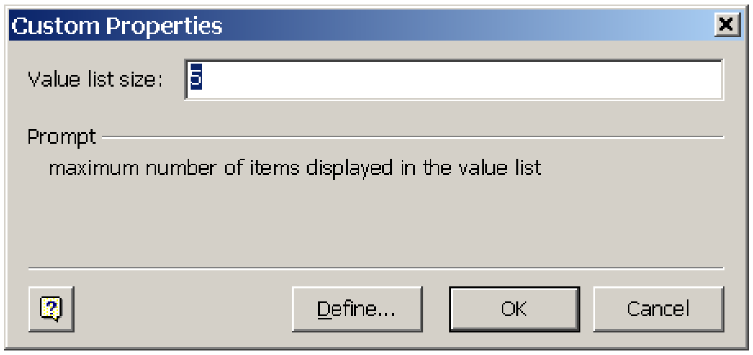
Range
From:
(empty = no min or no max)
To:
Value
Add
Remove

Defined values/ranges:
<New Value/Range>
A
B
C
F

Figure 7 Adding a simple value constraint in the database properties sheet

By default, up to 5 entries are displayed in a value constraint. You can change this setting for an individual object type by right-clicking the object type, then choosing Shape > Custom Properties and changing the Value list size number in the custom properties sheet (see Figure 8). This dialog can also be used to introduce new custom properties of your own.

If you set the number to be less than the number of entries in the constraint, an ellipsis "…" is appended to the display list to indicate that other values are possible but their display has been elided. This is useful when the list is quite large. Regardless of how many values are displayed on the diagram, all the values you enter in the constraint are included in the constraint for DDL generation purposes.


Custom Properties

Value list size: 5

Prompt
maximum number of items displayed in the value list

Figure 8 The maximum number of entries displayed in a value constraint is controlled via a custom property

Conclusion

You now have enough information to recreate the Employee page of the Employee ORM source model. As a challenge, you may wish to see if you can reproduce the rest of the Employee model shown on the Project and Room pages. I'll discuss in detail how to do this in Part 3. If you have any constructive feedback on this article, please e-mail me at: TerryHa@microsoft.com.

References

1. Halpin, T. A. 1998a, 'Object Role Modeling: an overview', white paper, (online at www.orm.net).
2. Halpin, T.A. 1998b, 'Object-Role Modeling (ORM/NIAM)', *Handbook on Architectures of Information Systems*, Bernus, P., Mertins, K. & Schmidt (eds), Springer, Heidelberg, Ch. 4. (online at www.orm.net).
3. Halpin, T.A. 2001a, *Information Modeling and relational Databases*, Morgan Kaufmann Publishers, San Francisco (www.mkp.com/books_catalog/catalog.asp?ISBN=1-55860-672-6).
4. Halpin, T.A. 2001b, 'Microsoft's new database modeling tool: Part 1', *Journal of Conceptual Modeling* (online at www.InConcept.com and www.orm.net).