Data Modeling with IBM Rational Rose: Things to Remember

This is my draft of class notes for 15 Sep 2005, along with some of my own UML notes. Please send any mistakes you find to smriti@cs.utexas.edu [18 September, 2005]

1. **HOWTO: Transform to Data Model**
   a. Tree View →Logical View →Right click on the *package* →Data Modeler → Transform to Data Model
   b. Check Schemas for the generated schema

2. **HOWTO: Generate SQL**
   a. Right click on the schema → Forward Engineer → Save as .ddl file
   b. Automatically saves in <Installation Folder>/Rose/

3. **Right click** works for most 'new' operations

4. **Object Model: Non-directional associations not in default Toolbox.** Add the arrow explicitly by right click on toolbox →Customize.

5. **Does not draw automatically or delete automatically**
   a. Delete from Tree View. Deleting an association from the diagram does *not* remove it from your model.
   b. Drag drop from Tree View to Diagram to draw.
   c. The diagram is not gospel truth – the Tree View is.

6. **Choose logical data types** when drawing the object model, not physical

7. **Only persistent classes make it to the schema**:
   a. Detail: check off 'persistent' to make classes appear in the schema

8. All classes must belong to a **package in the Logical View**

9. **To designate your own Primary key**
   a. Rational generates a PK for each persistent class. To designate your own: Tree View → right click on the attribute → Data Modeler →Part of Object Identity

10. **Identifying (by value) versus Non-Identifying (by reference) Relationships**
    a. Object Model: Properties→ Role A/B → check 'by value' (UML composition)
    b. 'by value' filled aggregation diamond in OM = identifying relationship in DM
    c. 'by reference' unfilled diamond in OM = non-id relationship in DM

d. A non-identifying relationship means two keys will be added – PK and FK. An identifying relationship will create one key and use it both as FK and PK.

11. Many:Many relationship

a. **1..n : 1..n**
b. Make sure Detail → 'by value' is NOT checked (must be a non-identifying relationship
c. Default: generates a default third class
    i. Specify your own third class: Attach an association class to the association. (add the association class symbol to the toolbox first Toolbox right click → Customize)

Quick UML Refresher

Smriti Ramakrishnan
18 September, 2005

Unidirectional Aggregation: filled diamond with arrow: *"Has" Relationship, by value or by reference.*
http://www.cs.rhul.ac.uk/CompSci/Computers/rational/html/rose_ada/1ood_83.html

"Has" relationships are not part of the UML notation. However, they can be created in Rose using the **View > As Booch** option. When viewed using the Booch or OMT notation, they are displayed as unidirectional aggregation relationships.
The "has" (aggregation) relationship denotes a whole/part association. There are two distinct types of "has" relationships: by-value and by-reference. A by-value "has" relationship, also known as physical containment, generally indicates that the part does not exist independently of the whole, and/or the whole is responsible for construction and destruction of the part. A by-reference relationship, also referred to as logical containment, indicates that the part is not physically contained within the whole and is potentially shared with other objects.
A "has" relationship becomes a component in the client's class record type. The type of the record component depends on the by-value or by-reference nature of the relationship. If the relationship is by-value, the type of the component is the class type of the part class (i.e., Object). If the relationship is by-reference, the component type must use the access type of the part class (i.e., Handle).

Aggregation: empty diamond: (by reference), component may exist by itself, implied multiplicity at diamond end is 0..1. The component can live on without the parent
Eg: (computer system, monitor).

Composition: filled diamond: (by value), components cannot exist by themselves. Implied multiplicity at diamond end is 1..1. (You use aggregation, and make it 'by value' for the role corresponding to component implying it is an identifying relationship in the Data Model). If parent dies, component dies too.
Eg: (employee, access card) *Question: Does composition map to representing weak entities?*

Unidirectional Association: single arrow: also got by the "Navigable" text box in Role detail. Bidirectional associations have both roles navigable. You can 'navigate' from one entity to another – arrow from dependent to employee means you can find employee-id of a dependent from the Dependent table, but cant find the dependent-id of an employee from the employee table. (*Changing bidirectional to unidirectional associations is an example of a common refactoring action*)

ER Modeling References

a) Garcia-Molina Textbook
b) Raghu Ramakrishnan, ER Modeling Concepts, Lecture Notes (www.cs.utexas.edu/~smriti/ta/cs386/er_ramakrishnan.ps)