The Data Administration Newsletter
Since 1997

Many-To-Many Relationships in XML Schema: Hints and Tips For the Busy E-R Modeler

by Bill Lewis
Published: January 1, 2006

(Article URL: http://www.tdan.com/view-articles/5046)

Published in TDAN.com January 2006

Background

Recently I've had the opportunity to interface with several government initiatives concerned with modeling data to be shared among law enforcement agencies at federal, state, local and tribal levels. These initiatives typically are labeled with exotic and arcane acronyms: FEA DRM, NEIM, TWPDES, GJXDM, ICMWG, NDex, R-Dex. It may come as a surprise to some in the data management community that the common data modeling ingredient in this alphabet soup is not ER, RDBMS, or even ORM--for better or worse, it's XML.

In these modeling domains, as in most, many-to-many relationships are a fact of life. Suspects or "entities of interest" can be sighted in many locations; locations can house multiple suspects; an identification document can be used by multiple perpetrators; a perpetrator can use many passports.

Modeling of relationships, especially many-to-many relationships, is particularly challenging in XML. It's no surprise that XML, being inherently hierarchical, accommodates basic one-to-many relationships relatively easily. If all one needs to represent is that a suspect has many addresses, this can be laid out in an XML schema with multiple Address elements as sub-elements within a Suspect element. Other techniques are also available for modeling relationships in XML, and attempts to settle on a standard from among this meta-modeling grab bag has caused significant controversy within XML-based efforts.

Many-to-Many

Many-to-many relationships add to the uncertainty. Say, for example, that multiple Suspects have been sighted at the same address-a terrorist cell, perhaps, or a safe house. What if we need to know the same passport or driver's license has been used by multiple collaborators, and each of these has used multiple other fake IDs? Let's take the case of Suspects and Addresses. In E-R terms, we need three entities. For broad applicability, we can generalize Suspect to Party, and Address to Location. Then we need an associative entity--Party Location Association. (I know, data modeling 101...but bear with me.) The three entities are shown in Figure 1 below.

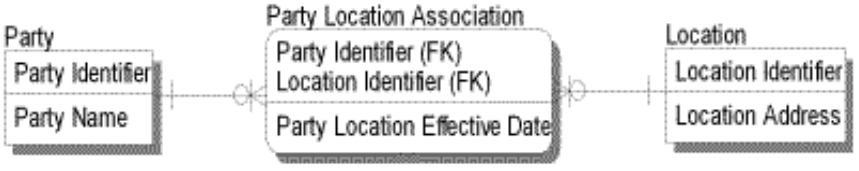

Figure 1

This is a straightforward enough convention in E-R modeling, and of course, it translates directly to a relational database schema.

So, how could we potentially use an XML schema representation to assure that this data is accurately and consistently shared among law enforcement agencies?

First and foremost, we cannot create LOCATION as a sub-element to PARTY, or vice versa. This would effectively declare that one is dependent on the other, whereas they are independent.

Just as it is represented in an E-R model as a separate but dependent associative entity, the many-to-many relationship can be represented in an XML schema as a separate but dependent Party_Location_Association element. A less-than-XML-fluent data modeler may have some clue as to the basic syntax of this element definition, but how should we represent, in XML, its relationships to the Party and Location elements?

Technology to the Rescue

Ah, the joy and wonder of software tools! XMLSpy is one tool that should be in the toolbox of any data modeler who brushes up against XML from time to time. And if there is any magic bullet for ER/relational conversion to XML, especially in the thorny case we're considering here, it's the "Create XML Schema from DB Structure" function in XMLSpy, shown in its pull-down list in Figure 2 below.

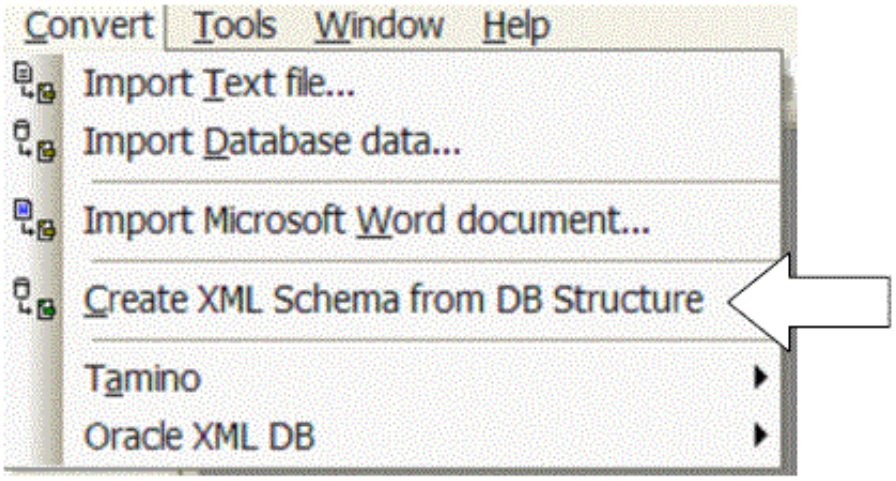

Figure 2

After the model above has been forward-engineered into three tables in a relational database, invoking this function will produce an equivalent XML schema (to the extent that equivalence is possible, which is another topic).

The collapsed view in Figure 3 below shows the "entity elements" -Location, Party, and Party_Location_Association--and their key fields, as represented in the schema generated by XMLSpy.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >
<xs:element name="Location" sql:key-fields="Location_Identifier">
<xs:element name="Party" sql:key-fields="Party_Identifier">
<xs:element name="Party_Location_Association" sql:key-
  fields="Party_Identifier,Location_Identifier">
  </xs:schema>
```

Figure 3

Expanding the schema reveals the details below. (Some additional details have been omitted in the interest of

legibility.) In Figure 4, the dashed lines delineate the three entity elements, and within each entity element, the entity names are highlighted.

In addition, the relationships between Party_Location_Association and its two parent entities are shown. These are represented by the xs:key element of Party_Location_Association and the xs:keyref elements of its parents.

- Within the Party element, the xs:keyref subelement contains an xs:selector subelement whose xpath attribute references the Party_Location_Assocation entity element. The xs:keyref element also contains an xs:field subelement, and the value of its xpath attribute references the first xpath attribute of the Party_Location_Assocation element.

- Within the Location element, the xs:keyref subelement contains an xs:selector subelement whose xpath attribute also references the Party_Location_Assocation entity element. Its xs:keyref element also contains an xs:field subelement, and the value of its xpath attribute references the second xs:field subelement in the Party_Location_Assocation element.

```
------------------------------------------------------------------
s:element name="Party">
<xs:key name="Party_PrimaryKey_0">
      <xs:selector xpath="." />
      <xs:field xpath="Party_Identifier" />
</xs:key>
<xs:keyref name="Party_To_Party_Location_Association_FK2">
      <xs:selector xpath="Party_Location_Association" />
      <xs:field xpath="Party_Identifier" />
</xs:keyref>
xs:element>
------------------------------------------------------------------
element name="Location">
<xs:key name="Location_PrimaryKey_0">
      <xs:selector xpath="." />
      <xs:field xpath="Location_Identifier" />
</xs:key>
<xs:keyref name="Location_To_Party_Location_Association_FK1">
      <xs:selector xpath="Party_Location_Association" />
      <xs:field xpath="Location_Identifier" />
</xs:keyref>
xs:element>
------------------------------------------------------------------
s:element name="Party_Location_Association">
<xs:key name="Party_Location_Association_PrimaryKey_0">
      <xs:selector xpath="." />
      <xs:field xpath="Party_Identifier" />
      <xs:field xpath="Location_Identifier" />
</xs:key>
xs:element>
------------------------------------------------------------------
```

Figure 4

Whew...now that wasn't so bad, was it? (Insert smiley face here...)

In this XML representation, the parents in effect reference the child, in contrast to a relational database implementation where the child table's foreign keys would reference its parents. This, understandably, may seem rather backwards from a relational perspective.

Late-breaking Consumer Advisory

A common data modeling practice is to represent a self-referencing many-to-many relationship as a "linkage"

entity. One of the most familiar examples of this is the Party Relationship, shown in Figure 5 below.

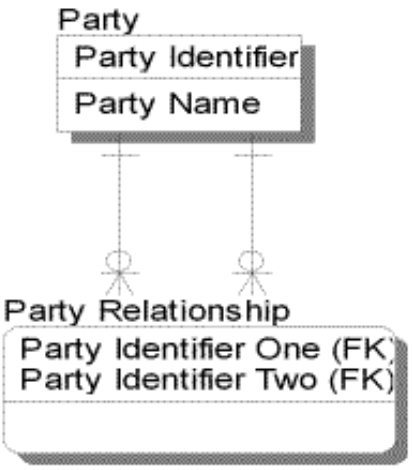

Figure 5

What appears to be bad news is that in cases such as this, the XMLSpy schema-conversion function generates an invalid schema. Evidently having multiple references (two, in this case) between a child entity element and another parent entity element is not syntactically valid.

Parting Shots

In my humble opinion, XML's incomplete, verbose and diverse attempts at representing relational constraints (a.k.a. business rules or semantics) are foremost among the many reasons that it scarcely qualifies as an eXcellent Modeling Language. But its widespread adoption as a standard for modeling data requirements, while troublesome, cannot be ignored.

Hopefully, data modelers-some of whom, like myself, may still be considerably XML-challenged--will find some of these hints and tips (and maybe even opinions) of use. Better yet, poke holes in them and let me know about it at datamodel@aol.com.

Go to Current Issue | Go to Issue Archive

Recent articles by Bill Lewis

- Data Types of the Future!
- Data-Driven Molecular Specifications, Part 2
- Data-Driven Molecular Specifications, Part 1

Bill Lewis - Bill is a Data Architect with IBM Global Business Solutions. His more than 25 years of information technology experience span the financial services, energy, health care, software and consulting industries. In addition to his current specializations in data management, metadata management and business intelligence, he has been a leading-edge practitioner and thought leader on topics ranging from software development tools to IT architecture. He has contributed to numerous online and print publications, and is the author of *Data Warehousing and E-Commerce*. He can be reached at lewisw@us.ibm.com.