

Towards a UML Persistence Model (Relational)

Scott W. Ambler

President, Ronin International

scott.ambler@ronin-intl.com

www.ronin-intl.com

www.ambysoft.com



Copyright 2000 Scott W. Ambler

Scott W. Ambler



■ **Consultant:**

- Object/Component Architecture & Modeling
- Software Process Mentoring
- Object/Component Development Mentoring

■ **Author/Editor:**

- The Object Primer 2nd Edition
- Building Object Applications That Work
- Process Patterns
- More Process Patterns
- The Elements of Java Coding Style
- The Unified Process Series (R&D Books)

■ **Contributing Editor/Writer:**

- Software Development
- Computing Canada

Overview



- What is persistence modeling
- Why we need persistence modeling
- Requirements
- Are data models sufficient?
- How do we model keys? ==> Implications
- Conclusions

What Is Persistence Modeling?



- Persistent objects are stored in a persistence mechanism
- Unless your system takes a “pure” approach and uses an objectbase, you will need to model your persistent schema
- Data models are traditionally used to model relational database schemas
- File schemas for flat files
- Relational databases are the norm within the industry

Why Persistence Modeling?



- Most software requires persistent objects
- There is a wide selection of persistence mechanisms
 - Flat files
 - Relational databases
 - Object-relational databases
 - Objectbases
- First you model something, then you build it. A lot of object developers need to build relational databases to support their software yet the UML has little to offer on this issue.

Proposed Scope



- We need to at least model physical RDB schemas
- Logical persistence models?
- Non-RDB approaches to storage?

Towards Requirements For a UML Persistence Model (Relational)



- Relational technology is based on tables, columns, rows and the relationships between them -- classes and objects are completely different animals
- Relational databases support stored procedures and triggers, not member functions
- Associations are implemented via keys, not references
- Tables are joined, not traversed
- Traceability within a relational database can be complex yet important for extensibility
- Access paths are important to understand for performance tuning
- Relational databases allow you to implement indices and views on tables
- Visit <http://www.ambysoft.com/umlPersistenceProfile.html> for details

Design: Are Data Models Sufficient?



- Data models show tables/entities, the relationships between them, and the columns of the tables/entities
- Views?
- Indices?
- Stored procedures?
- Access paths?

Design: How Should Keys Be Modeled?



Stereotypes?

Constraints?

Stereotypes with tagged values?

Implied by indices?

Implication: It isn't quite as easy as you would initially think.

Conclusions



- Object persistence is clearly an important issue
- The UML currently has little to say on the issue
- Persistence modeling is complex, arguably stretching the UML meta model
- It isn't clear that data models are sufficient for physical modeling
- Let's define the scope of the effort first, then requirements, then develop a profile or entire new model