1. Why map an UML schema to a RDBM?

Numerous disciplines require information concerning phenomena implicit or explicitly associated with a location relative to the Earth. Disciplines using Geographic Information (GI) in particular are those within the earth and physical sciences, and increasingly those within social science and medical fields. Therefore geographic datasets are increasingly being shared, exchanged and frequently re-purposed for uses beyond their original intended use.

ISO Technical Committee 211 (ISO/TC 211) together with Open Geospatial Consortium (OGC) provide a series of standards and guidelines for developing application schemas which should:

1. capture relevant conceptual aspects of the data involved;
2. be sufficient to satisfy previously defined use-cases of a specific or cross-domain schema;
3. be extensible and evolve without breaking backward compatibility.

Within a full model-driven approach the UML should remain at the centre of any implementation claiming to represent the model itself. In this context, a UML -> XSD -> RDBM transformation is not always possible because of the both the XSD and RDBM, and even on OWL, implementation, have the specific to represent the same model. In the example scenario an ingested XML document is exported into a single XML-type field. This approach works well when the core data are a fraction of the whole document, and even better if the main aim of the RDBM is to store rather than enrich XML objects.

In both UML and GI an object class can inherit attributes and/or methods either from one or many parent classes. However, only a few RDBMs can partially handle inheritance rules. To address this issue the ISO/TC 211 created an abstract interface which links inheritance and composition rules. Composition may appear less "natural" than inheritance but it provides a more expression would fit better a model driven approach.

Inheritance vs Composition

In both UML and OO a child class can inherit attributes and/or methods either from one or many parent classes. However, only a few RDBMs can partially handle inheritance rules. To address this issue the ISO/TC 211 created an abstract interface which links inheritance and composition rules. Composition may appear less "natural" than inheritance but it provides a more natural interface than inheritance.

A UML model may map the same information to multiple foreign key relationships between tables. This allows the advantage over inheritance by being a more thorough translation of nested hierarchies which may be described by a hierarchy of descendants classes. A RDBM may then exploit this information through use of multiple foreign key relationships between tables.

Why map an UML schema to a RDBM?

The essence of a UML model consists of three components: object, relation, constraint.

An object is typically a common, language independent, Object Oriented (OO) class with its attributes and methods.

A relation connects two objects, whose types are forced either when defined at class level, i.e. inheritance, or at the attribute level, i.e. via multiplicity.

A constraint is a logical condition on invoking the object's features.

An RDBM is established through:

1. creating a collection of O-O classes and further enriching these classes with appropriate methods to satisfy the model constraints.
2. enforcing the UML relationships in the RDBM with an appropriate collection of foreign keys and subsequently create additional tables (many-to-many or circular relations).

However, it should be noted that:

- the UML is not a comprehensive language and does not cover all the needs of a good database design.
- the UML is not a programming language and does not cover all the needs of a good database implementation.
- the UML is not a hardware description language and does not cover all the needs of a good database deployment.
- the UML is not a user interface specification and does not cover all the needs of a good database presentation.

3. Inheritance vs Composition

In both UML and GI a child class can inherit attributes and/or methods either from one or many parent classes. However, only a few RDBMs can partially handle inheritance rules. To address this issue the ISO/TC 211 created an abstract interface which links inheritance and composition rules. Composition may appear less "natural" than inheritance but it provides a more natural interface than inheritance.

A UML model may map the same information to multiple foreign key relationships between tables. This allows the advantage over inheritance by being a more thorough translation of nested hierarchies which may be described by a hierarchy of descendants classes. A RDBM may then exploit this information through use of multiple foreign key relationships between tables.

4. Implementation details

Once a XMI format of the UML model has been exported a Java application parses the XMI retrieving the necessary models to import "on-the-fly" through an inner name: http://en.wikipedia.org/wiki/Composition_over_inheritance lookup table, creating an internal description of the model. The application then uses Apache Velocity to generate a python collection of OO classes and an SQLAlchemy mapping script. Within a complex project this means that the application can potentially generate between the collection of classes in any specific OO language and XSD or OWL schemas.

Implementation details

A typical python application based on Urilib uses the UML2RDBM parser (UML2RDBM) to generate a python object of UML design, specifically an EPB (a user written class/interface) which offers an online version of the model. The EPB is a user written class/interface which in a single object allows the database to be defined in the UML. It allows the transformation of the UML model into an SQLAlchemy mapping script. Within a complex project this means that the application can potentially generate between the collection of classes in any specific OO language and XSD or OWL schemas.