The Life Sciences Research domain task force of the Object Management Group (OMG, 2001) is concerned with the development of standards for data exchange in biomolecular sciences, and in 2002 approved a macromolecular structure Corba specification. Corba (the common object request broker architecture) is a middleware architecture intended to serve just this purpose of providing access to standard objects representing discrete logical entities suitable for programmatic manipulation. Corba promotes interoperability across networked applications by separating entirely the API from the implementation of the underlying data objects. For applications such as the macromolecular structures database hosted by the Protein Data Bank, the attraction of networked interoperability is that information can be accessed through distributed and federated databases, and can be delivered on demand to any compatible software.

A Corba application comprises an interface definition language (IDL) and an API that together define access to a data structure that encapsulates the abstract representation of the objects and relationships relevant to a particular area of knowledge. In general terms, this data structure may be described as an 'ontology' (Westbrook & Bourne, 2000). The ontology adopted for macromolecular structure (MMS) data was based on the mmCIF dictionary following a submission by the Research Collaboratory for Structural Bioinformatics to a Request for Proposal (Greer, 2000).

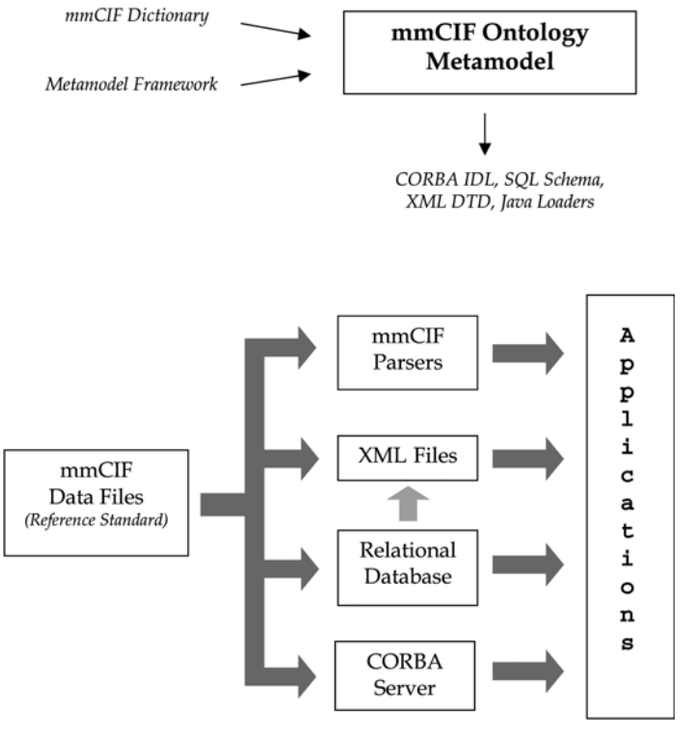

Fig. 5.3.8.1. The *OpenMMS* metamodel and data flow.

5.3.8.2.1. *The OpenMMS toolkit*

In practice, the ontology was developed in a 'metamodel' that combined the definitions and relationships between data items specified in the mmCIF dictionary with a generic metamodel framework. The metamodel extracts the information in the mmCIF dictionary but maintains it in a representation that is independent of the mmCIF STAR or any other file format. The standard building block of the metamodel is an *Entry* object, modelling a single macromolecular structure.

From a suitable metamodel, it then becomes relatively straightforward to generate alternative expressions of the information to suit different access requirements. The *OpenMMS* toolkit (Greer *et al.*, 2002) was built using Java source code to generate a Corba interface, an SQL schema for relational database loading and an XML representation of macromolecular data sets (Fig. 5.3.8.1).

The toolkit contains an mmCIF parsing module capable of direct access to the underlying data archive of mmCIF data files. This is important, because the data files represent a common reference for all the derived representations. Any errors or discrepancies between the expressed forms of the Corba, XML or SQL representations are resolved against the standard mmCIF reference form.

The relational database supporting an SQL-92 compatible interface provides an appropriate API for many applications, particularly ones that require extensive string searches. The close relationship between the mmCIF data model and relational database models has already been described earlier in this volume (Chapter 2.6).

Advantages of the SQL interface are that it provides rapid access direct to the binary data storage representation and that individual components of a data set may be efficiently retrieved without the need to search sequentially through an entire entry.

This efficiency of access and the ability to retrieve individual MMS data elements from a remote server is best realized through the Corba interface, the primary purpose of which is indeed to facilitate such high-performance access.

The bulk exchange of data is addressed through the generation of XML files. XML is a simple, powerful and widely used standard for interchanging data, and its use for transporting macromolecular data obviates the need for target applications to build their own STAR parsers. However, the use of markup tags around every individual data element does make the files much larger than their mmCIF progenitors. This is not an insurmountable problem in large-scale application environments, but it can undermine the effectiveness of XML as a representation mechanism in such applications as web browsers. A possible approach to this could be to define different, less verbose, XML representations and populate these on demand from a database store, either by SQL or XML queries. This is not an approach that the current *OpenMMS* toolkit supports directly.

Fig. 5.3.8.2 is an extract from an XML data file generated from the PDB structure 1xy2. The XML uses a reserved name space *PDBx* conforming to the schema http://deposit.pdb.org/pdbML/pdbx-v0.905.xsd. Data tags map cleanly to the corresponding data names in the mmCIF dictionary formed by concatenating the XML element name with its parent category name. For example, the entry `<PDBx:length_a>27.080</PDBx:length_a>` included in the `<PDBx:cellCategory>` container tag can be directly translated to the corresponding mmCIF data item **`_cell.length_a 27.080`**. CIF data loops are represented by repeated instances of the XML tag representing the corresponding CIF data name (for example, the multiple `<PDBx:audit_author name>` tags are equivalent to a CIF **`loop_ _audit_author.name`** construct). Nonstandard items with a *pdbx_* prefix (*e.g.* `<PDBx:pdbx_description>` in the `<PDBx:entityCategory>` group) refer to private data names in the PDB extension dictionary (Appendix 3.6.2).

5.3.8.3. *mmLib*: a Python toolkit for bioinformatics applications

While the libraries developed for use within the Protein Data Bank provide powerful functionality, their very size and complexity make them inappropriate for some applications. Indeed, considerable effort may be needed to compile the C++ code on non-standard platforms. The *mmLib* toolkit (Painter & Merritt, 2004)