5.5. THE USE OF mmCIF ARCHITECTURE FOR PDB DATA MANAGEMENT

5.5.2.3. Supporting other data formats and data delivery methods

One of the greatest benefits of a dictionary-based informatics infrastructure is the flexibility that it provides in supporting alternative data formats and delivery methods. Because the data and all of their defining attributes are electronically encoded, translation between data and dictionary formats can be achieved using lightweight software filters without loss of any information.

XML provides a particularly good example of the ease with which data can be converted to and from the mmCIF format. XML translations of mmCIF data files are currently provided on the Worldwide PDB ftp site (ftp://ftp.wwpdb.org/pub/pdb/data/structures/divided/XML/). These XML files use mmCIF dictionary data-item names as XML tags. These files were created by a translation tool (http://sw-tools.pdb.org/apps/MMCIF-XML-UTIL/) that translates mmCIF data files to XML in compliance with an XML schema. The XML schema is similarly software-translated from the PDB exchange data dictionary.
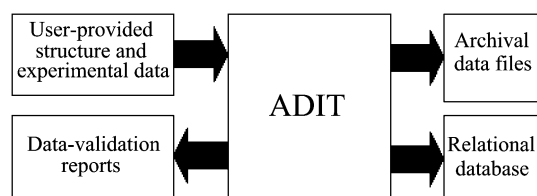
Other delivery methods such as Corba (http://www.omg.org/cgi-bin/doc?lifesci/00-02-02) do not require a data format, as data are exchanged using an application program interface (API). A Corba API for macromolecular structure (Greer *et al.*, 2002) based on the content of the mmCIF data dictionary has been approved by the Object Management Group (OMG). Software tools supporting this Corba API (*OpenMMS*, http://openmms.sdsc.edu, and *FILM*, http://sw-tools.pdb.org/apps/FILM) take full advantage of the data dictionary in building the interface definitions and supporting server on which the API is based (see also Section 5.3.8.2).

5.5.3. Integrated data-processing system: overview

The RCSB PDB data-processing system has been designed to take full advantage of the features of the mmCIF metadata framework. The AutoDep Input Tool (*ADIT*) is an integrated data-processing system developed to support deposition, data processing and annotation of three-dimensional macromolecular structure data.

This system, which is outlined in Fig. 5.5.3.1, accepts experimental and structural data from a user for deposition. Data are input in the form of data files or through a web-based form interface. The input data can be validated in a very basic sense for syntax compliance and internal consistency. Other computational validation can also be applied, including checking the input structure data against a variety of community standard geometrical criteria and comparing the input experimental data with the derived structure model. The suite of validation software used within *ADIT* is distributed separately (http://sw-tools.pdb.org/apps/VAL/). All of this validation information is returned to the user as a collection of HTML reports.

In addition to providing data-validation reports, *ADIT* also encodes data in archival data files and loads data into a relational database. The loading of data into the relational database is aided by an expert annotator. The *ADIT* system customizes its behaviour according to the user's requirements. One important distinction is between the behaviour of the interface provided for depositing data and that of the interface used for annotating the data. The depositor is focused only on data collection and provides the simplest possible presentation of the information to be input. The annotator sees the detail of all possible data items as well as the full functionality of the supporting data-processing software and database system.

Although the *ADIT* system was originally developed to support the centralized data deposition and annotation of macromolecular structure data, it is not limited to these particular applications. Because the architecture of the *ADIT* system derives the full scope of information to be processed from a data dictionary, the system can transparently provide data input and processing functionality for any content domain. This feature has been exploited in building a data-input tool for the BioSync project (Kuller *et al.*, 2002). The *ADIT* system can also be configured in workstation mode to provide single-user data collection and processing functionality. This version of the *ADIT* system as well as the supporting mmCIF parsing and data-management tools are currently distributed by the RCSB PDB under an open-source licence (http://sw-tools.pdb.org/apps/ADIT).

5.5.3.1. *ADIT*: functional description

The basic functions of the *ADIT* deposition system are shown in Fig. 5.5.3.2. Users interact with the *ADIT* system through a web server. The CGI components of the *ADIT* system (that is, functional software components interacting with web input data through the Common Gateway Interface protocol) dynamically build the HTML that provides the system user interface. These CGI components are currently implemented as compiled binaries from C++ source code.

User data can be provided in the form of data files or as keyboard input. Input files can be accepted in a variety of formats. *ADIT* uses a collection of format filters to convert input data to the data specification defined in a persistent data dictionary. Data in the form of data files are typically loaded first. Any input data that are not included in uploaded files can be keyed in by the user. *ADIT* builds a set of HTML forms for each category of data to be input. At any point during an input session, a user may choose to view or deposit the input data. Users who are depositing data may also use the data-validation services through the *ADIT* interface.

Comprehensive data ontologies like the PDB exchange dictionary contain vast numbers of data definitions. A data-input application may only need to access a small fraction of these definitions at any point. To address the problem of selecting only the relevant set of input data items from a data dictionary *ADIT* uses a view database. In addition to defining the scope of the data items to be edited by the *ADIT* application, an *ADIT* data view also stores

Fig. 5.5.3.1. Functional diagram of the *ADIT* system.
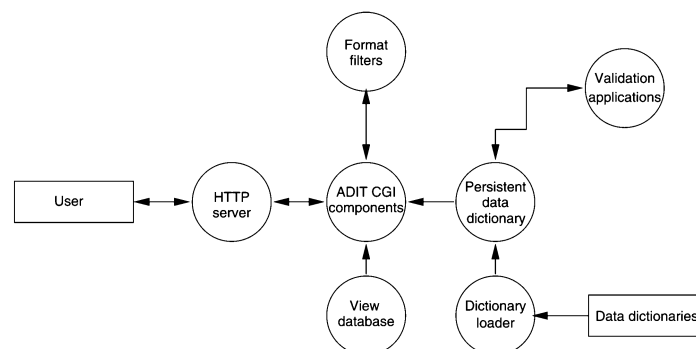
Fig. 5.5.3.2. Schematic diagram of *ADIT* editing, format translation and validation functions.
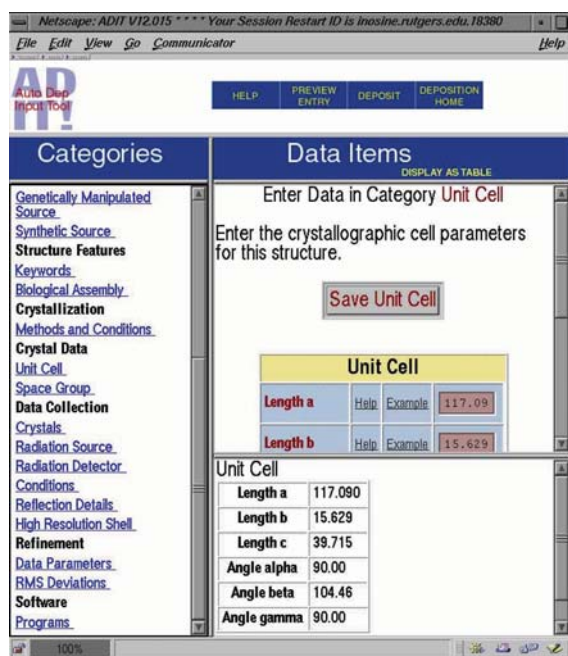
Fig. 5.5.3.3. Example *ADIT* data-input screen.

Fig. 5.5.3.4. Schematic diagram of *ADIT* database loading functions.

presentation details that are used in building the HTML input forms. An important use of the data view is to provide a simple and intuitive presentation of information for novice users which disguises the complex details of a data dictionary.

Fig. 5.5.3.3 shows an example *ADIT* editing screen for the crystallographic unit cell. The data dictionary category containing this information is named CELL, and the length of the first cell axis is defined in the dictionary as `_cell.length_a` (Fig. 5.5.2.2*b*). In this case, the data view has substituted *Unit Cell* and *Length a* for the dictionary data names. Although this example is simple, some dictionary data names are as long as 75 characters, and in these instances the ability to display a simpler name is essential.

Precise dictionary definitions and examples obtained from the data dictionary are accessible from the *ADIT* interface through buttons next to each data item. *ADIT* makes full use of the dictionary specification in data-input operations. Data items defined to assume only specific values have pulldown menus or selection boxes. Data type and range restrictions are checked when data are input and diagnostics are displayed to the user if errors are detected.

For performance reasons, the data dictionary is converted from its tabular text structure to an object representation using *CIFOBJ*. The class supporting the object representation provides efficient access functions to all of the data dictionary attributes. A dictionary loader is used to check the consistency of the data dictionary and to load the object representation from the text form of the data dictionary.

Any dictionary that complies with the dictionary description language (DDL2) can be loaded and used by *ADIT*. All *ADIT* software components gain their knowledge of the input data from the data dictionary and any associated data views. Consequently, *ADIT* can be tailored for use in virtually any data-input and data-processing application.

5.5.3.2. Generalized database support

In addition to the data editing and processing functions, *ADIT* also supports a versatile database loader (*mmCIF Loader*; http://sw-tools.pdb.org/apps/MMCIF-LOADER) that builds database schemata and extracts the processed data required to load
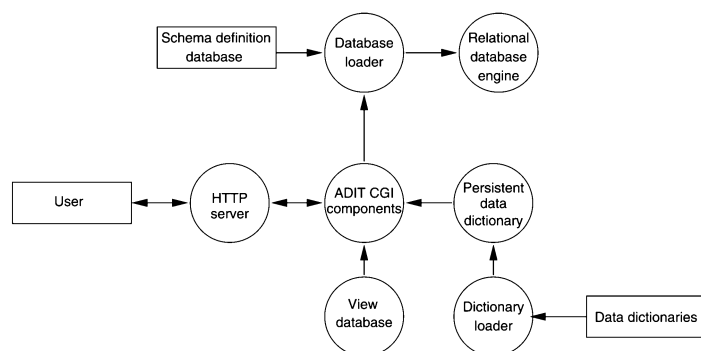
database instances. The relation of the database loader to the central components of the *ADIT* system is shown in Fig. 5.5.3.4.

Schemata are defined in a metadata repository that is accessed by the loader application. In the simplest case, a schema can be constructed that is modelled directly from the data dictionary. Since the data model underlying the dictionary description language used to build *ADIT* data dictionaries is essentially relational, mapping a data dictionary specification to a relational schema is straightforward.

In other cases, a mapping is required between the target schema and the data dictionary specification. This mapping is encoded in the schema metadata repository. The database loader uses this mapping information to extract items from data files and translate these data into a form that can be loaded into the target database schema. The definition of the mapping operation can include: selection operations with equijoin constraints (*e.g.* the value of `_entity.type` where `_entity.id` = 1), aggregation (*e.g.* count, sum, average), collapse (*e.g.* vector to string), type conversions and existence tests.

Schema definitions are converted by the database loader into SQL instructions that create the defined tables and indices. Loadable data are produced either as SQL insert/update instructions or in the more efficient table copy formats used by popular database engines (*i.e.* DB2, Sybase, Oracle and MySQL). Loadable data can also be produced in XML.

5.5.3.3. Building a structure-determination data pipeline

One goal of high-throughput structural genomics is the automatic capture of all the details of each step in the process of structure determination. Fig. 5.5.3.5 shows a simplified structure-determination data pipeline. The essential details of each pipeline step are extracted and later assembled to make a data file for PDB deposition. The RCSB PDB data-processing infrastructure has been developed in anticipation of a data pipeline in which automated deposition would be the terminal step. The dictionary technology and software tools developed by the RCSB PDB to process and manage mmCIF data can be reused to provide the data-handling operations required to build the pipeline.

Dictionary definitions have been carefully developed to describe the details of each step in the structure-determination pipeline. These data items are typically accessible in electronic form after each program step. The information is either exported directly in mmCIF format or is printed in a program output file. To deal with the latter case, a utility program, *PDB_EXTRACT* (http://sw-tools.pdb.org/apps/PDB_EXTRACT), has been developed to parse program output files and extract key data values. In either case, the results of this incremental extraction of data from each program step must be merged to build a complete mmCIF

542

references