2. CONCEPTS AND SPECIFICATIONS

of data items in lists. In this way the DDL, and consequently the dictionaries constructed from the DDL, invoke aspects of relational and object-oriented database paradigms. It is therefore useful to summarize these briefly here.

A relational database model (Kim, 1990) presents data as tables with a hierarchy of links that define the dependencies among tables. These explicit relationships enable certain properties of data to be shared, and, for related data values, to be derived. The structure of data links in a relational database is usually defined separately from the component data. This is an important strength of this approach. However, when data types and dependencies change continually, static relationships are inappropriate, and there is a need for non-relational extensions.

The object-oriented database model (Gray *et al.*, 1992) allows data items and tables to be defined without static data dependencies. A data item may be considered as a self-contained 'object' and its relationships to other objects handled by 'methods' or 'actions' defined within the objects. A database may have a base of statically defined explicit relationships with a dynamic layer of relationships provided by presenting some (or all) items as objects. Objects have well defined attributes, some of which may involve relationships with other data items, but objects need not have pre-ordained links imposed by the static database structure.

The attributes and the functionality of CIF dictionaries incorporate aspects of both the basic relational and the object-oriented model. These provide the flexibility and extensibility associated with object-oriented data, as well as the relational links important to data validation and, ultimately, data derivation.

2.5.3. Definition attributes

Efficient data exchange depends implicitly on the prior knowledge of the data. For CIF data, this knowledge is specified in a data dictionary using definition attributes. A unique set of attribute values exists for each kind of data item, be it numerical, textual or symbolic, because these characteristics represent its identity and function. This is illustrated below with two simple examples.

2.5.3.1. Example 1: attributes of temperature

Every numerical data item has distinct properties. Consider the number 20 as a measure of temperature in degrees. To understand this number it is essential to know its measurement units. If these are degrees Celsius, one knows the item is in the *temperature* class, *degrees Celsius* sub-class, and that a lower enumeration boundary of any value can be specified at -273.15. Such a constraint can be used in data validation. More to the point, without any knowledge of both the class and subclass, a numerical value has no meaning. The number 100 is unusable unless one knows what it is a measure of (*e.g.* temperature or intensity) and, equally, unless one knows what the units are (*e.g.* degrees Celsius, Kelvin or Fahrenheit; or electrons or volts).

2.5.3.2. Example 2: attributes of Miller indices

Knowing the inter-dependency of one data item on another plays a major role in the understanding and validation of data. If a triplet of numbers $5, 3, 0$ is identified as Miller indices h, k, l, one immediately appreciates the significance of the index triplet as a vector in reciprocal crystal space. This stipulates that the three numbers form a single non-scalar data item in which the indices are non-associative (*e.g.* $3, 5, 0$ is not equivalent to $5, 3, 0$) and irreducible (*e.g.* the index 3 alone has no meaning). As a reciprocal-space vector, the triplet has other properties if is part of a list of other

Table 2.5.4.1. *Comparison of DDL1 and DDL2 variants*

DDL1	DDL2
Data names identify the category of data as `_<category>_<detail>`	Data names identify the category as `_<category>.<detail>`
Definitions are declared as data blocks with `data_<dataname>`	Definitions are declared starting with `save_<dataname>` and ending with `save_`
An irreducible set of items is declared within one definition *e.g.* Miller indices h, k, l	All items are defined in separate frames related by `_item_dependent.name`
Items that appear in lists are identified with the attribute `list_`	List and non-list data items are not distinguished
List dependencies are declared within each definition *e.g.* `_list_reference`	Dependencies are declared in a category definition *e.g.* `_category_key.name`
	Identifies subcategories of data within category groupings *e.g.* matrix
	Provides aliases to equivalent names, including those in DDL1 dictionaries

reciprocal-space data items; namely, its value represents a key or list pointer (*i.e.* a unique key to a row of items in a list table) to other data items in the list associated with this vector. This means that data forming a 'reflection' list are inaccessible if these indices are absent, or invalid if there is more than one occurrence of the same triplet in the list. Such interdependency and relational information is very important to the application of data, and needs to be specified in a dictionary to enable unambiguous access and validation. Other types of data dependencies will be described in Section 2.5.6.

2.5.4. DDL versions

The capacity of a DDL to precisely define data items depends implicitly on the scope of the available attributes. It is quite possible, therefore, that a completely new data property cannot be specified using an existing DDL. In a field where data types evolve rapidly, the currently used dictionary language may be inadequate for the precise specification of an item. It is inevitable that future data items will exceed the capacity of existing dictionary attributes and methods to describe new data properties, and the dictionaries must evolve much in the same way that spoken languages continually adapt to changing modes of expression.

The first DDL used in crystallography (in 1990) was developed to compile a dictionary of 'core structural' crystallographic data items (Hall *et al.*, 1991). These data items were intended for use when submitting a manuscript to the journal *Acta Crystallographica* (and still are). The 'core' DDL is known commonly as DDL1 (Hall & Cook, 1995). Several years later, the definition of macromolecular crystallographic data items needed hierarchical descriptors for the different levels of structural entities, and an 'mmCIF' DDL, known as DDL2 (Westbrook & Hall, 1995), was developed. DDL2 was used to build the mmCIF dictionary (Bourne *et al.*, 1997). DDL1 is described in this chapter and DDL2 is described in Chapter 2.6. The DDL1 attributes have been used to construct the crystallographic core (fundamental structural), pd (powder diffraction), ms (modulated and composite structures) and rho (electron density) dictionaries. These dictionaries are discussed in Chapters

```
data_atom_site_Cartn_
    loop_  _name                     '_atom_site_Cartn_x'
                                      '_atom_site_Cartn_y'
                                      '_atom_site_Cartn_z'
    _category                        atom_site
    _type                            numb
    _type_conditions                 esd
    _list                            yes
    _list_reference                  '_atom_site_label'
    _units                           A
    _units_detail                    'angstroms'
    _definition
;    The atom-site coordinates in angstroms specified
     according to a set of orthogonal Cartesian axes
     related to the cell axes as specified by the
     _atom_sites_Cartn_transform_axes description.
;
                            (a)
save__atom_site.Cartn_x
    _item_description.description
;    The x atom-site coordinate in angstroms
     specified according to a set of orthogonal
     Cartesian axes related to the cell axes as
     specified by the description given in
     _atom_sites.Cartn_transform_axes.
;
    _item.name                       '_atom_site.Cartn_x'
    _item.category_id                atom_site
    _item.mandatory_code             no
    _item_sub_category.id            cartesian_coordinate
    _item_aliases.alias_name         '_atom_site_Cartn_x'
    _item_aliases.dictionary         cif_core.dic
    _item_aliases.version            2.0.1
    loop_
    _item_dependent.dependent_name
                                     '_atom_site.Cartn_y'
                                     '_atom_site.Cartn_z'
    _item_related.related_name
                                     '_atom_site.Cartn_x_esd'
    _item_related.function_code associated_esd
    _item_type.code                  float
    _item_type_conditions.code  esd
    _item_units.code                 angstroms
    save_
                            (b)
```

Fig. 2.5.4.1. Comparison of DDL versions: (*a*) DDL1, (*b*) DDL2.

```
data_cell_formula_units_Z
    _name                       '_cell_formula_units_Z'
    _category                   cell
    _type                       numb
    _enumeration_range          1:
    _definition
; The number of the formula units in the unit cell
  as specified by _chemical_formula_structural,
  _chemical_formula_moiety or _chemical_formula_sum.
;
```

Fig. 2.5.5.1. DDL1 definition with a few attributes.

```
data_on_this_dictionary
    _dictionary_name          cif_example.dic
    _dictionary_version       0.0
    _dictionary_update        1999-03-15
    _dictionary_history
;  1999-03-11  Created as a dictionary example
   1999-03-15  Further simplifications
;
data_parameter_ABC
    _name                         '_parameter_ABC'
#       <<<<<<<<<< other data attributes here
data_factor_XYZ
    _name                         '_factor_XYZ'
#       <<<<<<<<<< other data attributes here
data_and_so_on
    _name                         '_and_so_on'
#       <<<<<<<<<< other data attributes here
```

Fig. 2.5.5.2. DDL1 definition showing basic dictionary organization.

3.2–3.5 and presented in full in Chapters 4.1–4.4, respectively, of this volume.

It is helpful to give a brief comparison of the definition capabilities of DDL1 and DDL2. The differences may be best illustrated by simply comparing their application to the definition of the same data item. The two separate definitions of `_atom_site_Cartn_x` are shown in Fig. 2.5.4.1.

Although both DDL versions conform to the syntax of the STAR File (DDL1 conforms to the CIF syntax but DDL2 uses STAR File save frames not permitted in CIF data files) and are composed of similar attributes, there are clear differences. These are summarized in Table 2.5.4.1.

The similarity in the two DDL versions is such that software exists for parsing and validating CIFs against the DDL1 or DDL2 dictionaries interchangeably (*e.g.* see the CIF toolbox software *CIFtbx* described in Chapter 5.4).

2.5.5. The structure of DDL1 definitions

The organization of definitions in a CIF dictionary is straightforward. We are all familiar with how words are described in a spoken-language dictionary. Each defined word is followed by a sequence of attribute descriptions, such as phonetic annotation, grammatical context, word origins, meanings, examples of use and so on. The definition of data items in a CIF dictionary is organized in the same way. Each item description is preceded by the item's unique identifying tag as a datablock code, and is composed of a sequence of attribute items specifying the item's characteristics. A full description of the DDL1 attributes is given in Chapter 4.9.

A series of example definitions will be used to introduce the basic structure of the dictionary and definitions. The definitions are of familiar crystallographic data items, extracted from the core CIF dictionary in Chapter 4.1. Note that, in some cases, the text part of the definitions has been abbreviated for conciseness.

2.5.5.1. Definition example 1: formula units per cell (*Z*)

A definition for formula units in a crystal unit cell (*Z*) is shown in Fig. 2.5.5.1. The definition includes only the attributes `_name`, `_category`, `_type`, `_enumeration_range` and `_definition`. These identify the item's unique properties and enable its validation. The `_category` attribute indicates the class of the defined data item. Since lists may contain only items of one category type, this attribute is critical to data items appearing in lists (see Section 2.5.5.5). The attribute `_enumeration_range` has the value '1:' which stipulates that the item must be 'one or a higher number'.

2.5.5.2. Definition example 2: dictionary audit information

A CIF dictionary file contains, in addition to the data definitions, information about the nature of the dictionary. Fig. 2.5.5.2 shows the typical organization of definitions in a dictionary. Each dictionary starts with the audit information giving its version and creation history. This is followed by definitions in separate data

references