International Tables for Crystallography (2006). Vol. G, Section 3.1.6, pp. 79–83.

`_units_detail`. The latter is a character field describing the units; the `_units` attribute is a code that may be interpreted by machine. In DDL1-based dictionaries, type codes are purely conventional, and there is no mechanism for converting units or relating quantities in different units. Table 3.1.5.1 lists the units codes used in the DDL1-based dictionaries described in this volume. There can be some inconsistencies: two codes ('s' and 'sec') are already in use to indicate the time unit of seconds.

The original CIF paper (Hall *et al.*, 1991) described a convention allowing physical quantities to be listed in a CIF in units other than those specified in the dictionary. Under this convention, a data name representing a value expressed in different units could be constructed by appending one of a series of known 'units extension codes' to the standard data name. Thus `_cell_length_a_pm` would represent a cell length expressed in picometres instead of the default ångströms. This approach is now deprecated, and all quantities must be expressed in the single unit permitted in their definition block. However, to allow the formal validation of old CIFs, a 'compatibility dictionary' is available which defines all data names that could have been constructed under this convention in a properly DDL1.4-compliant form. *This dictionary should only be used for validating old CIFs, and must not be used to construct new data files.* The dictionary is called cif_compat.dic in the IUCr CIF dictionary register (see Section 3.1.8.2).

Table 3.1.5.1. *Units codes and their interpretation in DDL1-based dictionaries*

Unit code (`_units`)	Meaning (`_units_detail`)
`A`	Ångströms
`A^-1^`	Reciprocal ångströms
`A^2^`	Ångströms squared
`A^3^`	Ångströms cubed
`Da`	Daltons
`K`	Kelvins
`Kmin^-1^`	Kelvins/minute
`Mgm^-3^`	Megagrams per cubic metre
`\ms`	Microseconds
`deg`	Degrees
`deg/min`	Degrees per minute
`eV`	Electronvolts
`e_A^-3^`	Electrons per cubic ångström
`fm`	Femtometres
`kPa`	Kilopascals
`kV`	Kilovolts
`kW`	Kilowatts
`mA`	Milliamperes
`min`	Minutes
`mm`	Millimetres
`mm^-1^`	Reciprocal millimetres
`s`	Seconds
`sec`	Seconds

3.1.6. Constructing a DDL2 dictionary

The DDL2 dictionary definition language was designed to specify a relational data model and has provision for including within a dictionary tables of relationships between data entries. Like a relational database which contains tables describing the data tables in the database, DDL2-based dictionaries contain definition blocks describing CIF categories, units and relationships as well as data items.

Unlike DDL1 dictionaries, a DDL2 dictionary is presented as a single data block. Within this data block a number of looped lists describe properties of the dictionary as a whole, or properties and relationships shared across the items defined in the dictionary. Typically these are: the dictionary name, version identifiers and revision history; the category groupings that give structure to the items defined by the dictionary; the labels that identify closely related data items; and the physical units employed in the dictionary, their definitions in terms of base units and their interconversion factors.

Definitions of individual data items and categories are contained within save frames. While the save frames are not referenced by name in any dictionary application, they permit multiple occurrences of data definition tags within the scope of a single data block and are therefore suitable for structuring a data dictionary. It is a convention that the name of a save frame defining a category is given in capitals, and the name of a save frame for a definition of a data item is given as lower-case. For example, `save_ATOM_SITE` is the name of the save frame defining the category with the `atom_site` identifier, while `save__atom_site.details` is the name of the save frame holding the definition of the individual data name `_atom_site.details` (note how the initial underscore character of the data name is preserved following the initial `save_` string of the save-frame name).

As with DDL1 dictionaries, the name of the dictionary itself (given by the data name `_dictionary.title`) is usually of the form `cif_`*`identifier`*`.dic`, where the *identifier* is a short code for the topic area of the dictionary (*e.g.* 'img' for the image dictionary, 'sym' for the symmetry dictionary).

As is invariable with DDL2 data names, the names themselves are formed from the category name separated by a full stop from the specific descriptor of the item.

Fig. 3.1.6.1 shows the structure of the macromolecular CIF dictionary. The ordering of the various looped lists and save frames is of no significance for machine parsing. The sole data block has the same name as the dictionary title string and the data block is introduced by the dictionary identification data items. The dictionary revision history introduces the file, followed by information about the extended data types and physical units used within the current dictionary. These are followed by the lists of closely related items (corresponding to 'irreducible sets' in DDL1 dictionaries and called 'subcategories' in the terminology of DDL2) and lists of category groupings. The body of the dictionary contains category and item definitions. Each category definition is followed by the definitions of its component data items. The ordering is alphabetic by category and then alphabetic by item name within categories.

3.1.6.1. Dictionary identification

Dictionary files must contain information that unambiguously states their identity and version. In DDL2-based dictionaries this is done using the dictionary attributes described in Section 2.6.6.4. The name of the data block comprising the whole content of a DDL2 dictionary is by convention the same as the dictionary identification string given as `_dictionary.title`. This value is repeated as the value of `_dictionary.datablock_id` (see Example 3.1.6.1) for use in checking the consistency of the dictionary.

The dictionary history is also an important audit record of changes to the dictionary content. Unlike in DDL1-based dictionaries where the history is contained in a single field, DDL2 provides a looped list of version labels, dates and annotations. For convenience, the history records in large DDL2-based dictionaries are sometimes placed at the end of the dictionary file.

3.1.6.2. Subcategory definitions

In the DDL1 formalism, particular relationships between data items may sometimes be stated within a text description or may be implied by the organization of the dictionary (where several data

```
data_mmcif_std.dic

    _dictionary.title           mmcif_std.dic
    _dictionary.version         2.0.09
    _dictionary.datablock_id    mmcif_std.dic
                              (a)

     loop_
         _dictionary_history.version
         _dictionary_history.update
         _dictionary_history.revision   .   .   .
                              (b)

     loop_
    _sub_category.id
    _sub_category.description    .   .

     loop_
    _category_group_list.id
    _category_group_list.parent_id
    _category_group_list.description   .   .   .
                              (c)

    loop_
    _item_type_list.code
    _item_type_list.primitive_code
    _item_type_list.construct
    _item_type_list.detail

    loop_
    _item_units_list.code
    _item_units_list.detail   .   .

     loop_
    _item_units_conversion.from_code
    _item_units_conversion.to_code
    _item_units_conversion.operator
    _item_units_conversion.factor   .   .   .   .
                              (d)

  save_CATEGORY_A   .   .   .   save_
    save__category_a.item_1   .   .   .   save_
    save__category_a.item_2   .   .   .   save_
    save__category_a.item_3   .   .   .   save_

  save_CATEGORY_B   .   .   .   save_
    save__category_b.item_1   .   .   .   save_
    save__category_b.item_2   .   .   .   save_
                              (e)
```

Fig. 3.1.6.1. Schematic structure of the macromolecular CIF dictionary. (*a*) Dictionary identifiers. (*b*) Dictionary history. (*c*) Subcategory and category group listings. (*d*) Data types, units descriptions and conversion tables. (*e*) Multiple category and item definition blocks.

Example 3.1.6.1. *DDL2 dictionary identification entries.*

```
data_mmcif_std.dic

    _dictionary.title           mmcif_std.dic
    _dictionary.version         2.0.09
    _dictionary.datablock_id    mmcif_std.dic

     loop_
    _dictionary_history.version
    _dictionary_history.update
    _dictionary_history.revision
   0.1.1  1993-02-11
;  Highlighted all notes with # %%%%% surrounds.
;
   .  .  .
```

Example 3.1.6.2. *DDL2 subcategories defined in the mmCIF dictionary.*

```
loop_
    _sub_category.id
    _sub_category.description
     'fractional_coordinate'
; The collection of x, y, and z components of a
  position specified with reference to unit cell
  directions.
;
   'matrix'
; The collection of elements of a matrix.
;
   'miller_index'
; The collection of h, k, and l components of the
  Miller index of a reflection.
;
   'cell_length'
; The collection of a, b, and c axis lengths of a
  unit cell.
;
   'mm_atom_site_label'
; The collection of alt id, asym id, atom id, comp id
  and seq id components of the label for a
  macromolecular atom site.
;
```

items are defined in the same data block and are understood to share the common attributes itemized in that data block).

Within DDL2, there are mechanisms for more formal and machine-parsable statements of relationships. The `_sub_category.id` attribute is a label shared by several data items within a category that are related in a specific way described by the associated `_sub_category.description` attribute. The relationships may be rather general, such as elements of a matrix; or they may be specific physical properties or attributes, such as the collection of axis lengths of a unit cell. The dictionary should list all such labels that occur within its included data definition blocks. Example 3.1.6.2 is an extract from the macromolecular dictionary.

3.1.6.3. Category groupings

In the DDL2 data model, a *category* of data corresponds to a set of related data items that may be stored in a single relational database table. A number of such tables may collectively describe the complete properties of some physical object. This is expressed formally by assigning the same label (`_category_group.id`) to the relevant categories. While relationships between categories are implied in DDL1 dictionaries by the hierarchical structure of the names of data items, in DDL2 dictionaries the relationships are formally stated.

For subcategories, the category-group relationships present in the dictionary are listed in a separate looped list. Example 3.1.6.3 is an extract from the macromolecular dictionary. The `inclusive_group` entry shows the common parentage of all categories (and ultimately all data items) in the dictionary.

3.1.6.4. Category definitions

In the DDL2 formalism, a category of data items may be mapped to a relational table. The dictionary entry for a category includes the name of the category (an identifying label which is referenced by the `_item.category_id` attribute of each component data item) and a list of the category groups of which it may be considered a member. The category *key* is explicitly specified – that is, the data item (or group of items) that uniquely identifies an individual row in a table of data of that category.

Where a category encompasses a set of data items that are not normally specified in a looped list, the category may nevertheless be taken to represent a degenerate table with a single row, and therefore there is still a category key. For degenerate categories the key value is often set equal to the name of the parent data block.

Example 3.1.6.3. *Category groups in a DDL2 dictionary.*

```
loop_
  _category_group_list.id
  _category_group_list.parent_id
  _category_group_list.description
    'inclusive_group'   .
; Categories that belong to the macromolecular
  dictionary.
;
    'atom_group'
    'inclusive_group'
; Categories that describe the properties of atoms.
;
    'audit_group'
    'inclusive_group'
; Categories that describe dictionary maintenance and
  identification.
;
    'cell_group'
    'inclusive_group'
; Categories that describe the unit cell.
;
```

Example 3.1.6.4. *A category description in a DDL2 dictionary.*

```
save_EXPTL
    _category.description
;   Data items in the EXPTL category record details
    about the experimental work prior to the
    intensity measurements and details about the
    absorption-correction technique employed.
;
    _category.id                  exptl
    _category.mandatory_code      no
    _category_key.name          '_exptl.entry_id'
     loop_
    _category_group.id           'inclusive_group'
                                 'exptl_group'
     loop_
    _category_examples.detail
    _category_examples.case
# - - - - - - - - - - - - - - - - - - - - - - - - -
;  Example 1 - based on laboratory records for
   Yb(S-C5H4N)2 (THF)4
;
; _exptl.entry_id                      datablock1
  _exptl.absorpt_coefficient_mu        1.22
  _exptl.absorpt_correction_T_max      0.896
  _exptl.absorpt_correction_T_min      0.802
  _exptl.absorpt_correction_type       integration
  _exptl.absorpt_process_details
  ; Gaussian grid method from SHELX76
    Sheldrick, G. M., "SHELX-76: structure
    determination and refinement program",
    Cambridge University, UK, 1976
  ;
  _exptl.crystals_number               1
  _exptl.details
  ; Enraf-Nonius LT2 liquid nitrogen
    variable-temperature device used
  ;
  _exptl.method    'single-crystal x-ray diffraction'
  _exptl.method_details
  ; graphite monochromatized Cu K(alpha) fixed tube
    and Enraf-Nonius CAD4 diffractometer used
  ;
;
# - - - - - - - - - - - - - - - - - - - - - - - - -
save_
```

Example 3.1.6.4 shows a category of non-looped core data items. It may be compared with the DDL1 version in Example 3.1.5.2.

For categories of looped items (those normally presented in a table of values) it is sometimes appropriate to have as the category key a data item that has the sole function of indexing unique table rows. However, it is also often the case that a composite key is formed from existing data items, and in these cases the category definition must loop the components of the key, as in Example 3.1.6.5 from the macromolecular dictionary definition of the GEOM_BOND category.

Example 3.1.6.5. *A DDL2 category with a composite key.*

```
save_GEOM_BOND
    _category.description
;   Data items in the GEOM_BOND category record
    details about the bond lengths as calculated
    from the contents of the ATOM, CELL and
    SYMMETRY data.
;
    _category.id                   geom_bond
    _category.mandatory_code       no
     loop_
    _category_key.name    '_geom_bond.atom_site_id_1'
                          '_geom_bond.atom_site_id_2'
                          '_geom_bond.site_symmetry_1'
                          '_geom_bond.site_symmetry_2'
     loop_
    _category_group.id            'inclusive_group'
                                  'geom_group'
     loop_
    _category_examples.detail
    _category_examples.case
# - - - - - - - - - - - - - - - - - - - - - - - - - -
;  Example 1 - based on data set TOZ of Willis,
   Beckwith & Tozer [Acta Cryst. (1991), C47,
   2276-2277].
;
;    loop_
     _geom_bond.atom_site_id_1
     _geom_bond.atom_site_id_2
     _geom_bond.dist
     _geom_bond.dist_esd
     _geom_bond.site_symmetry_1
     _geom_bond.site_symmetry_2
     _geom_bond.publ_flag
      O1  C2   1.342  0.004  1_555  1_555  yes
      O1  C5   1.439  0.003  1_555  1_555  yes
      C2  C3   1.512  0.004  1_555  1_555  yes
      C2  O21  1.199  0.004  1_555  1_555  yes
      C3  N4   1.465  0.003  1_555  1_555  yes
      C3  C31  1.537  0.004  1_555  1_555  yes
      C3  H3   1.00   0.03   1_555  1_555  ?
      N4  C5   1.472  0.003  1_555  1_555  yes
    # - - - - data truncated for brevity - - - -
;
# - - - - - - - - - - - - - - - - - - - - - - - - - -
save_
```

It must be remembered that, in practice, data files may lack some of the items required to determine the category key formally. For example, in the data set given in the GEOM_BOND example here, it is possible that the `_geom_bond.site_symmetry_` items may be absent because the listing is for a single connected molecule within an asymmetric unit. Robust parsing software must construct data keys by assigning NULL or other suitable default values to the missing key components.

Careful inspection of corresponding definitions in the DDL1 and DDL2 versions of core data items will demonstrate that the explicit category key specification in DDL2 dictionaries may be deduced within DDL1 dictionaries from the appropriate `_list_reference`, `_list_mandatory` and `_list_uniqueness` attributes of data-item definitions within a category (see also Section 2.5.6.4).

3.1.6.5. Data-item definitions

The bulk of a DDL2 data dictionary comprises the save frames that include descriptions of the meaning and properties of individual data names.

Unlike DDL1 dictionaries, where the definitions of several data names may be contained in a single data block (most commonly for a set of items that form a logical irreducible set), save frames in

Example 3.1.6.6. *Illustration of parent/child relationships between identifiers in related categories.*

```
loop_
  _struct_site.id
  _struct_site.details
    'P2 site C'
; residues with a contact < 3.7 Angstrom to an atom
  in the P2 moiety of the inhibitor in the
  conformation with _struct_asym.id = C
;
    'P2 site D'
; residues with a contact < 3.7 Angstrom to an atom
  in the P1 moiety of the inhibitor in the
  conformation with _struct_asym.id = D
;

loop_
  _struct_site_gen.id
  _struct_site_gen.site_id
  _struct_site_gen.label_comp_id
  _struct_site_gen.label_asym_id
  _struct_site_gen.label_seq_id
  _struct_site_gen.symmetry
  _struct_site_gen.details
    1  'P2 site C'  VAL  A   32  1_555  .
    2  'P2 site C'  ILE  A   47  1_555  .
    3  'P2 site C'  VAL  A   82  1_555  .
    4  'P2 site C'  ILE  A   84  1_555  .
    5  'P2 site D'  VAL  B  232  1_555  .
    6  'P2 site D'  ILE  B  247  1_555  .
    7  'P2 site D'  VAL  B  282  1_555  .
    8  'P2 site D'  ILE  B  284  1_555  .
```

DDL2 dictionaries each contain the definition for a single addressable concept.

For example, the three Miller index components of a diffraction reflection (**_diffrn_refln_index_h**, **_diffrn_refln_index_k**, **_diffrn_refln_index_l** that are described in the DDL1 core CIF dictionary in the data block **data_diffrn_refln_**) are described in a DDL2 dictionary in three separate save frames, **save__diffrn_refln.index_h**, **save__diffrn_refln.index_k** and **save__diffrn_refln.index_l**. In the DDL2 formalism, the intimate relationship between these three components is expressed through the common **_item_sub_category.id** value of `miller_index` and the mutual reference of the other Miller-index components by the **_item_dependent.dependent_name** entries in each separate save frame.

An apparent exception to this general rule is the case of save frames defining an item, often a category key, that is an identifier common to several categories. In this case, the save frame defining the 'parent' identifier implicitly defines the complete property set of each child identifier. For completeness, the respective child identifiers are each declared in their own save frames, but these act only as back references to the parent definition. This is explained more completely in Section 3.1.6.5.1 below.

3.1.6.5.1. *Inheritance of identifiers*

Example 3.1.6.6 is from an mmCIF of two related categories that describe characteristics of an active site in a macromolecular complex. The sites are described in general terms with a label and textual description in the STRUCT_SITE category (the first looped list in the example). Details of how each site is generated from a list of structural features form the STRUCT_SITE_GEN category (second loop or table).

It is clear that each instance of the data item **_struct_site_gen.site_id** in the second table must have one of the values listed as **_struct_site.id** in the first loop, because it is the purpose of these identifiers to relate the two sets of data: they are the glue between the two separate tables and must have the same values to ensure the referential integrity of the data set (that is, the consistency and completeness of cross-references between tables). Within a group of related categories like this, it is normal to consider one as the 'parent' and the others as 'children'.

Example 3.1.6.7. *A definition of an identifier which is parent to identifiers in other categories.*

```
save__struct_site.id
  _item_description.description
;     The value of _struct_site.id must uniquely
      identify a record in the STRUCT_SITE list.

      Note that this item need not be a number;
      it can be any unique identifier.
;
  loop_
    _item.name
    _item.category_id
    _item.mandatory_code
'_struct_site.id'            struct_site          yes
'_struct_site_gen.site_id'   struct_site_gen      yes
'_struct_site_keywords.site_id'
                             struct_site_keywords yes
'_struct_site_view.site_id'  struct_site_view     yes
  loop_
    _item_linked.child_name
    _item_linked.parent_name
'_struct_site_gen.site_id'        '_struct_site.id'
'_struct_site_keywords.site_id'   '_struct_site.id'
'_struct_site_view.site_id'       '_struct_site.id'

  _item_type.code               line
save_
```

Because all such linking data items must have compatible attributes, it is conventional in DDL2 dictionaries to define all the attributes in a single location, namely the save frame which hosts the definition of the 'parent' data item. In early drafts of DDL2 dictionaries, the 'children' were not referenced at all in separate save frames; software validating a data file against a dictionary was required to obtain all information about a child identifier from the contents of the save frame defining the parent. However, subsequent drafts introduced a minimal save frame for the children to accommodate dictionary browsers that depended on the existence of a separate definition block for each individual data item.

Consequently, the definition blocks in current DDL2 dictionaries conform to the structure in Example 3.1.6.7, which refers to the simple STRUCT_SITE example used above.

Note that the dependent data names are listed twice: once in the loop that declares their **_item.name** values and the categories with which they are associated; and again in a loop that makes the direction of the relationship explicit. A parent data item may have several children, but each child can have only a single parent (*i.e.* related data name whose value may be checked for referential integrity). Note also that each listed item has an **_item.mandatory_code** value of `yes`: because they are identifiers which link categories, they must be present in a table to allow the relationships between data items in different tables to be traced.

Other than the specific description text field, any declared attributes (in this example only the data type) have a common value across the set of related identifiers.

As mentioned above, it is not formally necessary to have a separate save frame for the individual children; but it is conventional to have such individual save frames containing minimal definitions that serve as back references to the primary information in the parent frame. These also provide somewhere for the specific text definitions for the children to be stored. The definition frame for **_struct_site_gen.id** is shown in Example 3.1.6.8.

Example 3.1.6.8. *Definition of a child identifier.*

```
save__struct_site_gen.id
  _item_description.description
;     The value of _struct_site_gen.id must uniquely
      identify a record in the STRUCT_SITE_GEN list.

      Note that this item need not be a number;
      it can be any unique identifier.
;
  _item.name                   '_struct_site_gen.id'
  _item.category_id              struct_site_gen
  _item.mandatory_code           yes
  _item_type.code                line

save_
```

Example 3.1.6.9. *DDL2 definition of a physical quantity.*

```
save__diffrn.ambient_temp
    _item_description.description
;       The mean temperature in kelvins at which the
        intensities were measured.
;
    _item.name                    '_diffrn.ambient_temp'
    _item.category_id               diffrn
    _item.mandatory_code            no
    _item_aliases.alias_name
                              '_diffrn_ambient_temperature'
    _item_aliases.dictionary        cif_core.dic
    _item_aliases.version           2.0.1
     loop_
    _item_range.maximum
    _item_range.minimum               .    0.0
                                    0.0   0.0
    _item_related.related_name
                                '_diffrn.ambient_temp_esd'
    _item_related.function_code     associated_esd
    _item_type.code                 float
    _item_type_conditions.code      esd
    _item_units.code                kelvins
save_
```

3.1.6.5.2. *Definitions of single quantities*

While it is important to ensure the referential integrity of the data in a CIF through proper book-keeping of links between tables, the crystallographer who wishes to create or extend a CIF dictionary will be more interested in the definitions of data items that refer to real physical quantities, the properties of a crystal or the details of the experiment. The DDL2 formalism makes it easy to create a detailed machine-readable listing of the attributes of such data.

Example 3.1.6.9 parallels the example chosen for DDL1 dictionaries of the ambient temperature during the experiment.

In the definition save frame, the category is specifically listed (although it is deducible from the DDL2 convention of separating the category name from the rest of the name by a full stop in the data name). The data type is specified as a floating-point number. (In the core dictionary there are fewer data types and the fact that the value may be a real rather than integer number must be inferred from the declared range.) The range of values is also specified with separate maximum and minimum values (unlike in DDL1 dictionaries, which give a single character string that must be parsed into its component minimum and maximum values). The assignment of the same value to a maximum and a minimum means that the absolute value is permitted; without the repeated '0.0' line the range in this example would be constrained to be positive definite; the equal value of 0.0 for maximum and minimum means that it may be identically zero.

The `_item_units.code` value must be one of the entries in the units table for the dictionary and can thus be converted into other units as specified in the units conversion table.

The aliases entries identify the corresponding quantity defined in the DDL1 core dictionary.

3.1.6.6. Units

As with data files described by DDL1 dictionaries, the physical unit associated with a quantitative value in a DDL2-based file is specified in the relevant dictionary. There is no option to express the quantity in other units. However, DDL2 permits a dictionary file to store not only a table of the units referred to in the dictionary (listed under `_item_units_list.code` and the accompanying descriptive item `_item_units_list.detail`), but also a table specifying the conversion factors between individual codes in the `_item_units_list.code` list. In principle, this allows a program to combine or otherwise manipulate different physical quantities while handling the units properly.

3.1.7. Composing new data definitions

Preceding sections have described the framework within which CIF dictionaries exist and are used, and their individual formal structures. While this is important for presenting the definition of new data items, it does not address what is often the most difficult question: what quantities, concepts or relationships merit separate data items? On the one hand, the extensibility of CIF provides great freedom of choice: anything that can be characterized as a separate idea may be assigned a new data name and set of attributes. On the other hand, there are practical constraints on designing software to write and read a format that is boundless in principle, and some care must be taken to organize new definitions economically and in an ordered way.

3.1.7.1. Granularity

Perhaps the most obvious decision that needs to be made is the level of detail or granularity chosen to describe the topic of interest. CIF data items may be very specific (the deadtime in microseconds of the detector used to measure diffraction intensities in an experiment) or very general (the text of a scientific paper). In general, a data name should correspond to a single well defined quantity or concept within the area of interest of a particular application. It can be seen that the level of granularity is determined by the requirements of the end application.

A practical example of determining an appropriate level of granularity is given by the core dictionary definitions for bibliographic references cited in a CIF. The dictionary originally contained a single character field, `_publ_section_references`, which was intended to contain the complete reference list for an article as undifferentiated text. *Notes for Authors* in journals accepting articles in CIF format advised authors to separate the references within the field with blank lines, but otherwise no structure was imposed upon the field. In a subsequent revision to the core dictionary, the much richer CITATION category was introduced to allow the structured presentation of references to journal articles and chapters of books. This was intended to aid queries to bibliographic databases. However, a full structured markup of references with multiple authors or editors in CIF requires additional categories, so that the details of the reference may be spread across three tables corresponding to the CITATION, CITATION_AUTHOR and CITATION_EDITOR categories. Populating several disjoint tables greatly complicates the author's task of writing a reference list. Moreover, the CITATION category does not yet cover all the many different types of bibliographic reference that it is possible to specify, and is therefore suitable only for references to journal articles and chapters of books. However, it is pos-

references