A Dictionary Description Language for Macromolecular Structure

Draft DDL V 2.1.1

September 1995

John D. Westbrook
Nucleic Acid Database Project
Department of Chemistry
Rutgers, The State University of New Jersey
and

Sydney R. Hall
Crystallography Center
University of Western Australia

This Draft Introductory Document Version 1.9 prepared by
John Westbrook

Please direct comments on this document to jwest@rutchem.rutgers.edu.

Contents

1 Background

This is an evolving reference document which provides a detailed specification of a Dictionary Description Language (DDL) designed to support macromolecular structure applications. The DDL described here represents the largely relational DDL presented at the CIF [1] Workshop in Brussels by John Westbrook, which was subsequently modified by Syd Hall and John Westbrook to be more conforming with existing applications of STAR [2] [3, 2, 6, 5] and CIF [4].

The DDL presented here extends the DDL that is currently [3] used by STAR and CIF applications in several important ways. Although these extensions have been introduced primarily to address problems that have surfaced in developing a macromolecular CIF dictionary, they are sufficiently general to be useful in a wide variety of applications.

Extensions to the current DDL have been introduced in a manner that provides the greatest degree of backward compatibility with existing CIF and STAR applications. Although the extensions introduced in this DDL have a rather profound effect on the structure of the DDL, dictionaries based on the new DDL can have the same simple appearance as dictionaries based on the current DDL. And even though a different convention is used to name data items in this DDL, an alias feature has been added to ensure that any published data item names will remain valid. In fact, the alias name mechanism provides for complete backward compatibility for all existing data files. It is hoped that the transition from the DDL in current use to this new DDL should not be difficult.

[1]CIF is the acronym for the Crystallographic Information File. CIF is now actively used for archiving and electronic publication of small molecule crystallographic structures. CIF is based on a subset of the STAR grammar described in reference [4].

[2]STAR is the acronym for the Self-defining Text Archive and Retrieval File which has been proposed [3] as a universal approach to electronic data exchange and archiving.

[3]At the time of this writing, CIF dictionaries based on DDL Version 1.4 were available via anonymous ftp from ftp.iucr.ac.uk.

1.1 What is a DDL?

The DDL is a dictionary of definitions which describes a language for specifying data definitions. DDL defines the data model that provides the foundation for the description of knowledge about an application domain.

The application knowledge is collected in a dictionary of definitions which describes the domain. DDL provides the framework on which this dictionary is organized by defining the levels of abstraction that are available to hold the data description. The DDL defines both the properties that may be associated with each level of abstraction and the relationships that may exist between levels. This DDL defines a relatively simple set of abstractions which include data blocks, categories, category groups, subcategories, and items.

Dictionaries use the abstractions provided by the DDL to describe the knowledge base of the application domain. The DDL presented here is expressed in the form of a dictionary that is based on its own underlying DDL. This self-consistent description not only provides a prototype for other application dictionaries, but it also provides a mechanism by which the consistency and relational integrity of the data model can be verified independent of an application domain.

1.2 The New DDL Data Model

The lowest level of abstraction provided by this DDL is the description of an individual data item. Collections of related data items are organized in categories. Categories may be thought of as tables in which each repetition of the group of related items adds a row. Within a category, those data items which determine the uniqueness of their group are designated as key items in the category. No data item group in a category is allowed to have a set of duplicate values of its key items. Each data item is assigned membership in one or more categories. Parent-child relationships may be specified for items which belong to multiple categories. These relationships permit the expression of the very complicated data structures required to describe the science of macromolecular structure. [4]

Other levels of abstraction in addition to category are also supported. Related categories may be collected together in category groups, and parent relationships may be specified for these groups. This higher level of association provides a vehicle to organize a large complicated collection of categories into smaller more relevant and potentially interrelated groups. Within the level of a category, subcategories of data items may be defined among groups of related data items. The subcategory provides a mechanism to identify that, for example, the data items month, day, and year collectively define a date.

1.3 The New DDL Specification

Each DDL definition is enclosed in a STAR save frame cell. [5] A save frame cell is named according to a rule where a *category identifier* and an optional *item identifier* are combined to uniquely identify a cell. The term *identifier* and its abbreviation *id* are used to refer to the two components of the full STAR item name. The term *name* is used to identify a complete STAR specification of a data item. The save frame cell is closed with an unnamed *save_* token.

For data item definitions, the category and item identifiers separated by a period form the save frame identifier, and the save frame identifier preceded by an underscore character forms the full STAR data item name. For category definitions, the category identifier alone is used as the save frame identifier. The syntax of the definition within a save frame cell follows the

[4] Although the terms *category* and *data item* are used here in order to conform with previous STAR and CIF applications, these terms could be replaced by *relation* and *attribute* which are more commonly used to describe the relational model which underlies the new DDL.

[5] Previous versions of DDL have used STAR data blocks to hold dictionary definitions; however, this introduces some inconsistency in the scope rules for data blocks between data files and dictionaries. In particular, the previous practice complicates the expression of relationships between definitions.

STAR grammar defined in reference [5]. The following abbreviated example illustrates this for the definition of the hypothetical item *item_a* in category *category_1*.

```
save__category_1.item_a

    _item.name           '_category_1.item_a'

save_
```

In this example, the data item *_item.name* is assigned the value *_category_1.item_a* which is the name of the item to which this definition applies.

2 DDL Definitions Describing Categories

In this section, the DDL definitions that describe the properties of categories, category groups, and subcategories are presented. Figure 1 illustrates the schema for these categories.

2.1 CATEGORY

This category contains the data items that describe the properties of collections of related data items. A DDL category is essentially a table, and in this category the characteristics of the table as a whole are defined. This category includes the data item, *_category.id*, to identify a category; the item, *_category.description*, to describe a category; the item *_category.mandatory_code*, to indicate if the category must appear in a data block; a method identifier, *_category.method_id*, which associates a procedure with a category; and *_category.implicit_key* which can be used to merge like categories between data blocks. The category identifier, *_category.id*, is a component of the basis in most of the DDL categories in this section. The parent definition of the category identifier as well as all of its child relationships are defined in this category.

Because special rules exist in the STAR grammar for the specification of data items which belong to a common category, the organization of data items within categories has a significant influence on how these items may be expressed in a data file. For example, a data category may be specified only once within a STAR data block or save frame, and at any level of a STAR loop structure only data items of a common category may appear.

The DDL for this category is given in the following section.

```
save_CATEGORY

    _category.description
;
     Attributes defining the functionality for the entire category.
;
    _category.id                         category
    _category.mandatory_code             no
    _category_key.name                  '_category.id'
     loop_
    _category_group.id                   'ddl_group'
                                         'category_group'
```

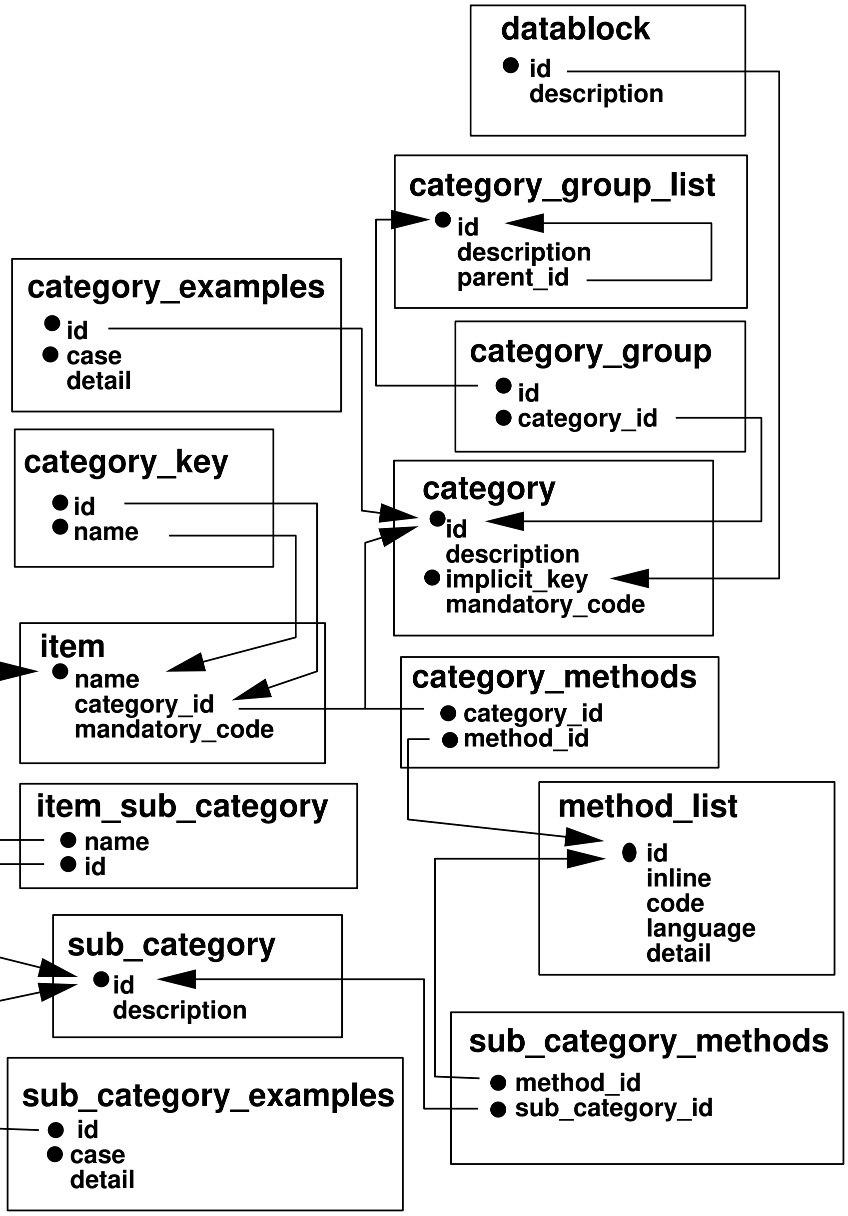

Figure 1: The schema of DDL definitions that describe categories, subcategories, and category Groups. In this figure, boxes surround data items which belong to the same category. Category identifiers are given in large font and item names are given in small font. Parent-child relationships are specified by lines connecting data items with the arrow pointing at the parent item. Key items within a category are identified by a preceding black dot.

```
     save_


save__category.id

    _item_description.description
;
     The identity of the data category. Data items may only be looped
     with items of the same category.
;
     loop_
    _item.name
    _item.category_id
    _item.mandatory_code
         '_category.id'                    category         yes
         '_category_examples.id'           category_examples implicit
         '_category_key.id'                category_key     implicit
         '_category_group.category_id'     category_group   implicit
         '_category_methods.category_id'   category_methods implicit
         '_item.category_id'               item             implicit

   _item_type.code                         idname

    loop_
    _item_linked.child_name
    _item_linked.parent_name
         '_category_examples.id'           '_category.id'
         '_category_group.category_id'     '_category.id'
         '_category_key.id'                '_category.id'
         '_category_methods.category_id'   '_category.id'
         '_item.category_id'               '_category.id'
     save_


save__category.description

    _item_description.description
;
     Text description of a category.
;
    _item.name                          '_category.description'
    _item.category_id                     category
    _item.mandatory_code                  yes
    _item_type.code                       text
     save_
```

```
save__category.implicit_key

    _item_description.description
;
     An identifier that may be used to distinguish the contents of
     like categories between data blocks.
;
     save_

save__category.mandatory_code

    _item_description.description
;
     Whether the category must be specified in a dictionary.
;
    _item.name                         '_category.mandatory_code'
    _item.category_id                    category
    _item.mandatory_code                 yes
    _item_type.code                      code
     save_
```

2.2 CATEGORY_EXAMPLES

The DDL category *category_examples* holds examples that apply to an entire category. This typically includes a complete specification of the category with annotations. An example specification consists of the text of the example, *_category_examples.case*, and an optional comment item, *_category_examples.detail*, which can be used to qualify the example. The basis for this category includes the items *_category_examples.category_id* and *_category_examples.case*. The former item is completely defined in the parent category, *category* (Section 2.1), and as a result this item is not redefined within this category. This compound basis permits multiple unique examples to be provided for each category.

The DDL for the *category_examples* category is given in the following section.

```
save_CATEGORY_EXAMPLES

    _category.description
```

```
;
     Example applications and descriptions of data items in this category.
;
    _category.id                        category_examples
    _category.mandatory_code            no
     loop_
    _category_key.name               '_category_examples.id'
                                     '_category_examples.case'
     save_


save__category_examples.case

    _item_description.description
;
     A case of examples involving items in this category.
;
    _item.name                        '_category_examples.case'
    _item.category_id                   category_examples
    _item.mandatory_code                yes
    _item_type.code                     text
     save_


save__category_examples.detail

    _item_description.description
;
     A description of an example _category_examples.case
;
    _item.name                        '_category_examples.detail'
    _item.category_id                   category_examples
    _item.mandatory_code                no
    _item_type.code                     text
     save_
```

2.3 CATEGORY_GROUP

The DDL category *category_group* names the category groups to which a category belongs. The assignment of a category to a category group is made when the category is defined. Each category group that is specified in this category must also be defined in the parent category, *category_group_list* (Section 2.4). The basis for this category also includes the category identifier *_category_group.category_id* which is completely defined in the parent category, *category* (Section 2.1).

The DDL for the *category_group* category is given in the following section.

```
save_CATEGORY_GROUP

    _category.description
;
     Provides a list of category groups to which the base category
     belongs.
;
    _category.id                     category_group
    _category.mandatory_code         no

     loop_
    _category_key.name               '_category_group.id'
                                     '_category_group.category_id'
     loop_
    _category_group.id               'ddl_group'
                                     'category_group'
    save_
```

2.4 CATEGORY_GROUP_LIST

The DDL category *category_group_list* holds data items that define category groups. Category groups are collections of related categories. Parent-child relationships may be defined for these groups. The specification of category groups and the relationships between these groups allows a complicated collection of categories to be organized into a hierarchy of more relevant groups. This higher level of structure is essential for large application dictionaries that may contain hundreds of category definitions.

The category *category_group_list*, holds the description of each category group, *_category_group_list.description*, and an optional parent group identifier, *_category_group_list.parent_id*. Category groups can be formed from collections of base categories, those categories that hold data. Category groups can also be formed from collections of base categories and category groups.

The DDL for the *category_group_list* category is given in the following section.

```
save_CATEGORY_GROUP_LIST

    _category.description
```

```
;
     This category provides the definition of each category group.
     A category group is a collection of related categories.
;
    _category.id                         category_group_list
    _category.mandatory_code             no
    _category_key.name                 '_category_group_list.id'
     loop_
    _category_group.id                   'ddl_group'
                                         'category_group'
    save_

save__category_group_list.id

    _item_description.description
;
     The name of a category group ...
;
     loop_
    _item.name
    _item.category_id
    _item.mandatory_code
         '_category_group_list.id'        category_group_list  yes
         '_category_group_list.parent_id' category_group_list  no
         '_category_group.id'             category_group       yes
    loop_
    _item_linked.child_name
    _item_linked.parent_name
         '_category_group.id'              '_category_group_list.id'
         '_category_group_list.parent_id'  '_category_group_list.id'
    _item_type.code                        idname
    save_

save__category_group_list.description

    _item_description.description
;
     Text description of a category group...
;
    _item.name                       '_category_group_list.description'
    _item.category_id                  category_group_list
    _item.mandatory_code               yes
    _item_type.code                    text

    save_
```

```
save__category_group_list.parent_id

    _item_description.description
;
     The name of the optional parent category group.
;
    save_
```

2.5 A CATEGORY_GROUP_LIST Example

The following example of *category_group_list* illustrates the category groups that are defined in this DDL. These include the group of categories that define categories, the group of categories that define data items, and the group of categories that define the dictionary. An additional compliance group is also defined for categories that are included specifically for compliance with previous versions of DDL. Each of these category groups is defined as child of the group named *ddl_group* to which all of the base DDL categories belong.

```
     loop_
    _category_group_list.id
    _category_group_list.parent_id
    _category_group_list.description

     'ddl_group'                                 .
;
     Component categories of the macromolecular DDL
;
     'datablock_group'                           'ddl_group'
;
     Categories that describe the characteristics of data blocks.
;
     'category_group'                            'ddl_group'
;
     Categories that describe the characteristics of categories.
;
     'sub_category_group'                        'ddl_group'
;
     Categories that describe the characteristics of subcategories.
;
     'item_group'                                'ddl_group'
;
     Categories that describe the characteristics of data items.
```

```
;
     'dictionary_group'                        'ddl_group'
;
     Categories that describe the dictionary.
;
     'compliance_group'                        'ddl_group'
;
     Categories that are retained specifically for compliance with
     older versions of the DDL.
;
```

2.6 CATEGORY_METHODS

The *category_methods* category is used to associate method identifiers with categories. Any number of unique method identifiers may be associated with a category. The method identifiers reference the full method definitions in the parent *method_list* category.

The DDL for the *category_methods* category is given in the following section.

```
save_CATEGORY_METHODS

    _category.description
;
    Attributes specifying the association between categories and methods.
;
    _category.id                         category_methods
    _category.mandatory_code             no
     loop_
    _category_key.name
                                      '_category_methods.method_id'
                                      '_category_methods.category_id'
     loop_
    _category_group.id
                                      'ddl_group'
                                      'category_group'
     save_
```

2.7 CATEGORY_KEY

The DDL category *category_key* identifies the data items within a category that form the basis for the category. The category basis uniquely identifies each group or tuple of items in the category. Using the analogy of the category as a table, no row in a table may have duplicate values for its key data items.

The choice of basis has important consequences in the specification of a category. It is important to ensure that the key items that form the category basis can be used to unambiguously identify any tuple of data items within the category. If this is not the case, then it may not be possible to reliably recover data items that are stored in the category. Because key items are required to address each tuple of items in a category, key items are considered mandatory items in the category.

It is interesting to note how the key data items have been selected for the categories that define the DDL, and how this choice of key items influences the structure of the DDL dictionary. In the DDL category *_category_key*, the basis includes both the the identifier for the category, *_category_key.id*, and the name of the key data item, *_category_key.name*. This choice of basis allows for any unique groups of items in a category to be defined as key items. The case of duplicate key item names within a category is forbidden by the data model. In the DDL category *_item_type*, the basis includes only the identifier for the item name, *_item_type.name*. This choice of basis has the desired effect of limiting the specification of item data type, *_item_type.code*, to a single choice for each data item.

The DDL for the *category_key* category is given in the following section.

```
save_CATEGORY_KEY

    _category.description
;
     This category holds a list of the item names that uniquely
     identify the elements of the category.
;
    _category.id                         category_key
    _category.mandatory_code             no
     loop_
    _category_key.name                 '_category_key.name'
                                       '_category_key.id'
     loop_
    _category_group.id                 'ddl_group'
```

```
                                       'category_group'
     save_

save__category_key.name

    _item_description.description
;
     The name of a data item that serves as a key identifier for the
     category (eg. a component of  the primary key).
;
     save_
```

2.8 SUB_CATEGORY

The DDL category *sub_category* provides data items to describe a subcategory and to associate a procedure with the subcategory. The procedure which is identified as *_sub_category.method_id* may be used to validate the subcategory. Subcategory validation may be required in instances where conditions are placed on the values of data items within the subcategory which are more restrictive than those associated with each component data item. A simple example of such a restriction would be a normalization restriction on the components of a subcategory. Any procedure referenced in this category must also be defined in category *method_list* (Section 4.1). The *method_list* category also contains the parent definition for the item *_sub_category.method_id*. The definition of this item specified in this category serves to qualify the special role of the item in this child category.

The DDL for the *sub_category* category is given in the following section.

```
save_SUB_CATEGORY

    _category.description
;
     The purpose of a sub-category is to define an association between
     data items within a category and optionally provide a method to
     validate the collection of items.   The sub-category named
     'cartesian' might be applied to the data items for the coordinates
      x,  y, and z.
;
    _category.id                        sub_category
    _category.mandatory_code            no
    _category_key.name                 '_sub_category.id'
     loop_
```

```
    _category_group.id                 'ddl_group'
                                       'sub_category_group'
     save_


save__sub_category.id

    _item_description.description
;
     The identity of the sub-category.
;
    _item_type.code                      idname
     loop_
    _item.name
    _item.category_id
    _item.mandatory_code
         '_sub_category.id'                      sub_category           yes
         '_sub_category_examples.id'             sub_category_examples  yes
         '_sub_category_methods.sub_category_id' sub_category_methods   yes
         '_item_sub_category.id'                 item_sub_category      yes
     loop_
    _item_linked.child_name
    _item_linked.parent_name
         '_sub_category_examples.id'                  '_sub_category.id'
         '_sub_category_methods.sub_category_id'      '_sub_category.id'
         '_item_sub_category.id'                      '_sub_category.id'
     save_


save__sub_category.description

    _item_description.description
;
     Description of the sub-category.
;
    _item.name                         '_sub_category.description'
    _item.category_id                    sub_category
    _item.mandatory_code                 yes
    _item_type.code                      text
     save_
```

2.9 SUB_CATEGORY_EXAMPLES

The DDL category *sub_category_examples* holds examples of a subcategory. A subcategory example might illustrate valid instances of the components which comprise the subcategory. An example specification contains the text of the example, *_sub_category_examples.case*, and an optional comment item, *_sub_category_examples.detail*, which can be used to qualify the example.

The basis for this category includes the items *_sub_category_examples.id* and *sub_category_examples.case*. The former item is completely defined in the parent category, *sub_category* (Section 2.8), and as a result this item is not redefined within this category. This compound basis permits multiple unique examples to be provided for each subcategory.

The DDL for the *sub_category_examples* category is given in the following section.

```
save_SUB_CATEGORY_EXAMPLES

    _category.description
;
     Example applications and descriptions of data items in this subcategory.
;
    _category.id                         sub_category_examples
    _category.mandatory_code             no
     loop_
    _category_key.name                 '_sub_category_examples.id'
                                       '_sub_category_examples.case'
     loop_
    _category_group.id                   'ddl_group'
                                         'sub_category_group'
     save_


save__sub_category_examples.case

    _item_description.description
;
     An example involving items in this subcategory.
;
    _item.name                         '_sub_category_examples.case'
    _item.category_id                    sub_category_examples
    _item.mandatory_code                 yes
    _item_type.code                      text
     save_
```

```
save__sub_category_examples.detail

    _item_description.description
;
     A description of an example _sub_category_examples.case
;
    _item.name                         '_sub_category_examples.detail'
    _item.category_id                    sub_category_examples
    _item.mandatory_code                 no
    _item_type.code                      text
     save_
```

2.10 SUB_CATEGORY_METHODS

The *sub_category_methods* category is used to associate method identifiers with subcategories. Any number of unique method identifiers may be associated with a subcategory. The method identifiers reference the full method definitions in the parent *method_list* category.

The DDL for the *sub_category_methods* category is given in the following section.

```
save_SUB_CATEGORY_METHODS

    _category.description
;
    Attributes specifying the association between subcategories and methods.
;
    _category.id                         sub_category_methods
    _category.mandatory_code             no
     loop_
    _category_key.name
                                      '_sub_category_methods.method_id'
                                      '_sub_category_methods.sub_category_id'
     loop_
    _category_group.id
                                      'ddl_group'
                                      'sub_category_group'
     save_
```

3 DDL Definitions Describing Data Items

In this section, the DDL categories that describe the properties of data items are presented. Figures 2 and 3 illustrate the schema for these categories.

3.1 ITEM

The *item* category is used to assign membership of data items to categories. This category forms the bridge between the category and data item levels of abstraction. The key data item in this category is the full STAR data item name, *_item.name*. This name contains both the category and data item identifiers, and is thus a unique identifier for the data item. The category identifier, *_item.category_id*, is included in this category as a separate mandatory data item. This has been done to provide an explicit reference to those categories that use the category identifier as a basis.

One could alternatively use the category and item identifiers as the basis for this category rather than the full STAR name, and thus eliminate the redundant specification of the category identifier. The full STAR name has been used here in order to provide compatibility with existing applications.

The *item* category also includes a code to indicate if a data item is mandatory in a category and therefore must be included in any tuple in the category. This code, *_item.mandatory_code*, may have three values: yes, no and implicit. The latter value indicates that the item is mandatory, but that the value of this item may be derived from context. In the case of an item name or a category identifier, these values can be obtained from the current save frame name. Implicit specification dramatically simplifies the appearance of each dictionary definition because it avoids the repetitive declaration of item names and category identifiers which are basis components for most categories.

The *item* category is presented in the next section of DDL. In this section, the definition of *_item.name* illustrates an important design feature of this DDL. Although the data item *_item.name* is the basis for all of the item level categories, its definition and properties need only be specified at a single point. Here, the data items that occur in multiple categories are defined only in the parent category. In certain situations, a child data item may be used in a manner which requires a description distinct from the parent data item. For instance, *_item_linked.parent_name* and *_item_linked.child_name* are both data item names as well as children of *_item.name*, but clearly the manner in

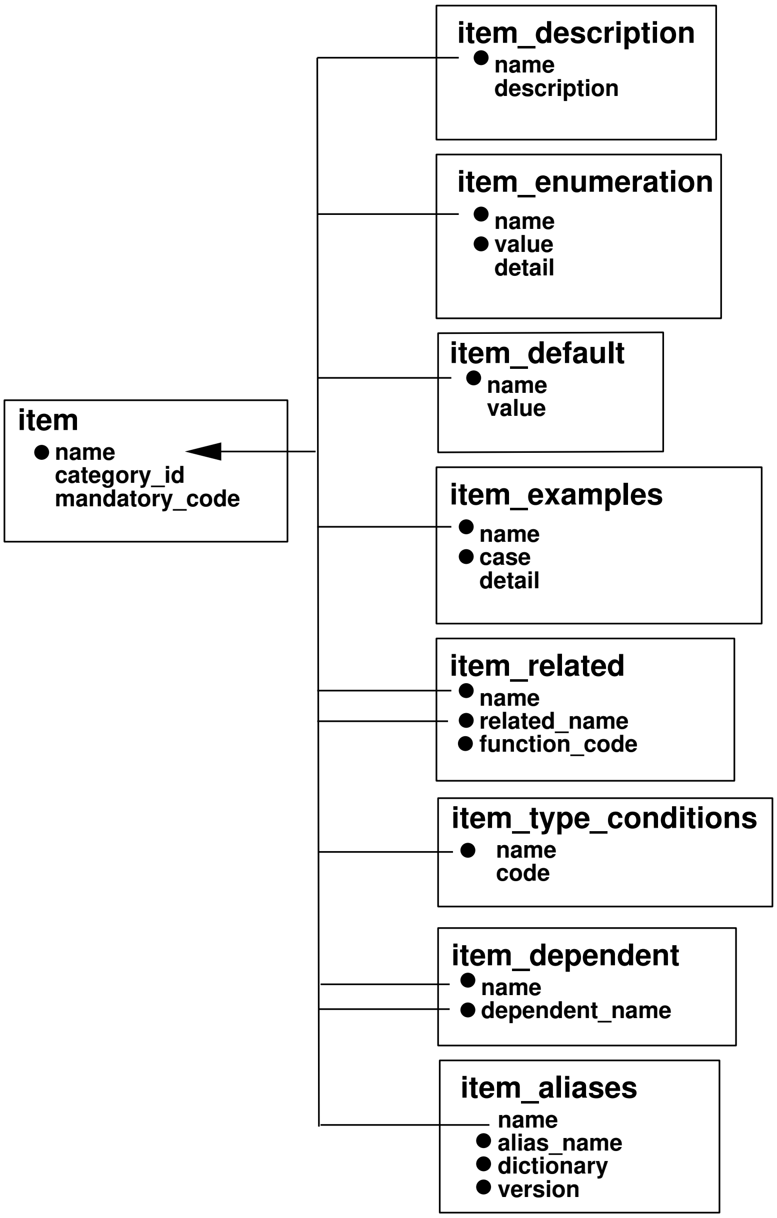

Figure 2: The schema of DDL definitions that describe data items (Part I) In this figure, boxes surround data items which belong to the same category. Category identifiers are given in large font and item names are given in small font. Parent-child relationships are specified by lines connecting data items with the arrow pointing at the parent item. Key items within a category are identified by a preceding black dot.

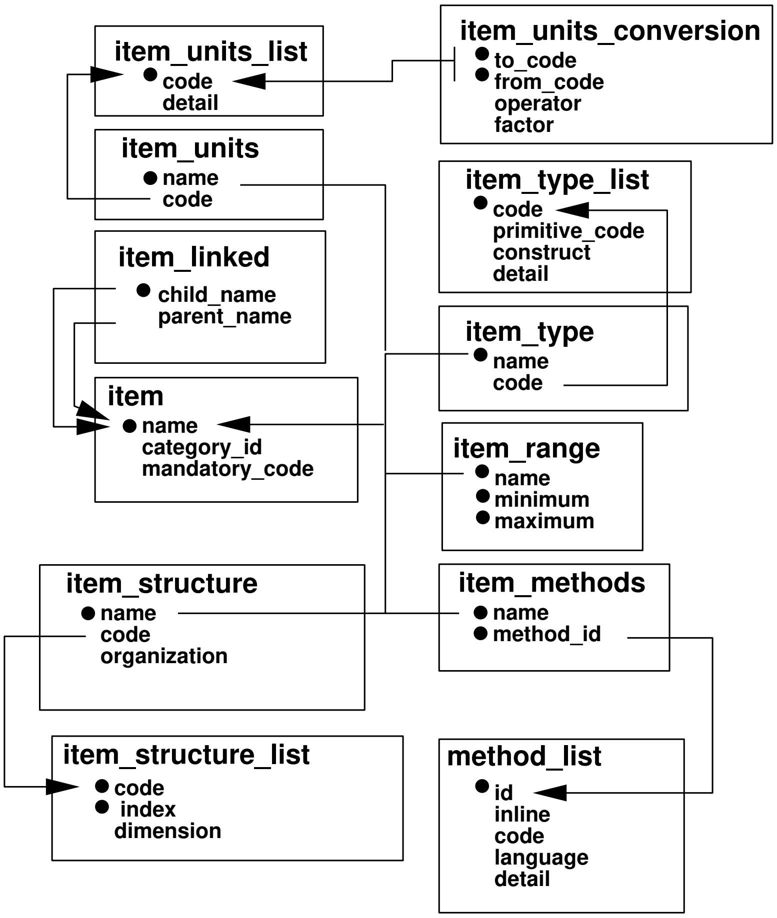

Figure 3: The schema of DDL definitions that describe data items (Part II) In this figure, boxes surround data items which belong to the same category. Category identifiers are given in large font and item names are given in small font. Parent-child relationships are specified by lines connecting data items with the arrow pointing at the parent item. Key items within a category are identified by a preceding black dot.

which these items are used in the *item_linked* category (Section 3.8) requires additional description. It is important to note that although the design of this DDL supports the definition of data items in multiple categories within the parent category, it is also possible to provide separate complete definitions within each category.

The DDL for the *item* category is given in the following section.

```
save_ITEM

    _category.description
;
     Attributes which describe the characteristics of a data item.
;
    _category.id                         item
    _category.mandatory_code             no
    _category_key.name                 '_item.name'
     loop_
    _category_group.id                  'ddl_group'
                                        'item_group'
     save_


save__item.name

    _item_description.description
;
     Data name of the defined item.
;
     _item_type.code                          name

      loop_
     _item.name
     _item.category_id
     _item.mandatory_code
          '_item.name'                      item                      implicit
          '_category_key.name'              category_key              yes
          '_item_aliases.name'              item_aliases              implicit
          '_item_default.name'              item_default              implicit
          '_item_dependent.name'            item_dependent            implicit
          '_item_dependent.dependent_name'  item_dependent            yes
          '_item_description.name'          item_description          implicit
          '_item_enumeration.name'          item_enumeration          implicit
          '_item_examples.name'             item_examples             implicit
          '_item_linked.child_name'         item_linked               yes
          '_item_linked.parent_name'        item_linked               implicit
```

```
         '_item_methods.name'              item_methods              implicit
         '_item_range.name'                item_range                implicit
         '_item_related.name'              item_related              implicit
         '_item_related.related_name'      item_related              yes
         '_item_type.name'                 item_type                 implicit
         '_item_type_conditions.name'      item_type_conditions      implicit
         '_item_structure.name'            item_structure            implicit
         '_item_sub_category.name'         item_sub_category         implicit
         '_item_units.name'                item_units                implicit

      loop_
     _item_linked.child_name
     _item_linked.parent_name
         '_category_key.name'              '_item.name'
         '_item_aliases.name'              '_item.name'
         '_item_default.name'              '_item.name'
         '_item_dependent.name'            '_item.name'
         '_item_dependent.dependent_name'  '_item.name'
         '_item_description.name'          '_item.name'
         '_item_enumeration.name'          '_item.name'
         '_item_examples.name'             '_item.name'
         '_item_linked.child_name'         '_item.name'
         '_item_linked.parent_name'        '_item.name'
         '_item_methods.name'              '_item.name'
         '_item_range.name'                '_item.name'
         '_item_related.name'              '_item.name'
         '_item_related.related_name'      '_item.name'
         '_item_type.name'                 '_item.name'
         '_item_type_conditions.name'      '_item.name'
         '_item_structure.name'            '_item.name'
         '_item_sub_category.name'         '_item.name'
         '_item_units.name'                '_item.name'

     save_


save__item.mandatory_code

    _item_description.description
;
     Signals if the defined item is mandatory for the proper description
     of its category.
;
    _item.name                         '_item.mandatory_code'
    _item.category_id                    item
```

```
    _item.mandatory_code                  yes
    _item_type.code                       code
     loop_
    _item_enumeration.value
    _item_enumeration.detail
            yes      'required item in this category'
            no       'optional item in this category'
            implicit 'required item but may be determined from context'
     save_
```

3.2 ITEM_ALIASES

The DDL category *item_aliases* defines the alias names that can be substituted for a data item name. The alias mechanism provides a means of identifying items by names other than those which follow the naming conventions used in this DDL. This feature should be used primarily to guarantee the stability of names defined in previously published dictionaries.

The items *_item_aliases.name*, *_item_aliases.dictionary* and *_item_aliases.dictionary_version* form the basis for this category. The items *_item_aliases.dictionary* and *_item_aliases.version* are provided to distinguish between dictionaries and different versions of the same dictionary. Any number of unique alias names can be defined for a data item.

The DDL for the *item_aliases* category is given in the following section.

```
save_ITEM_ALIASES

    _category.description
;
    This category holds a list of possible alias names or synonyms for
    each data item.  Each alias name is identified by the name and
    version of the dictionary to which it belongs.
;
    _category.id                          item_aliases
    _category.mandatory_code              no
     loop_
    _category_key.name                   '_item_aliases.alias_name'
                                         '_item_aliases.dictionary'
```

```
                                  '_item_aliases.dictionary_version'
     save_


save__item_aliases.alias_name

    _item_description.description
;
     Alias name for the data item.
;
    _item.name                    '_item_aliases.alias_name'
    _item.category_id               item_aliases
    _item.mandatory_code            yes
    _item_type.code                 aliasname
     save_

save__item_aliases.dictionary

    _item_description.description
;
     The dictionary in which the alias name is defined.
;
    _item.name                    '_item_aliases.dictionary'
    _item.category_id               item_aliases
    _item.mandatory_code            yes
    _item_type.code                 char
     save_


save__item_aliases.version

    _item_description.description
;
     The version of the dictionary in which the alias name is defined.
;
    _item.name                    '_item_aliases.version'
    _item.category_id               item_aliases
    _item.mandatory_code            yes
    _item_type.code                 char
     save_
```

3.3 ITEM_DEFAULT

The DDL category *item_default* holds default values assigned to data items. Default data values are specified in item *_item_default.value*. Default values are assigned to data items which are not declared within a category. The key item for this category is *_item_default.name* which is a child of item, *item.name* (Section 3.1). A single default value may be specified for a data item.

The DDL for the *item_default* category is given in the following section.

```
save_ITEM_DEFAULT

    _category.description
;
     Attributes specifying the default value for a data item.
;
    _category.id                        item_default
    _category.mandatory_code            no
    _category_key.name                '_item_default.name'
     loop_
    _category_group.id                'ddl_group'
                                      'item_group'
     save_


save__item_default.value

    _item_description.description
;
     The default value for the defined item if it is not specified
     explicitly. If a data value is not declared, the default is
     assumed to be the most likely or natural value.
;
    _item.name                      '_item_default.value'
    _item.category_id                 item_default
    _item.mandatory_code              no
    _item_type.code                   any
     save_
```

3.4 ITEM_DEPENDENT

The *item_dependent* category defines dependency relationships among data items within a category. Each data item on which a particular data item depends is specified as item *_item_dependent.dependent_name*. For a data item to be considered completely defined, each of its dependent data items must also be specified.

The DDL for the *item_dependent* category is given in the following section.

```
save_ITEM_DEPENDENT

    _category.description
;
     Attributes which identify other data items that must be specified
     for the defined data item to be valid.
;
    _category.id                         item_dependent
    _category.mandatory_code             no
     loop_
    _category_key.name                 '_item_dependent.name'
                                       '_item_dependent.dependent_name'
     save_


save__item_dependent.dependent_name

    _item_description.description
;
     Data name of a dependent item.
;
     save_
```

3.5 ITEM_DESCRIPTION

The DDL category *item_description* holds a description for each data item. The key item for this category is *item_description.name* which is defined in the parent category *item*. The text of the item description is held by item

_item_description.description. A single description may be provided for each data item.

The DDL for the *item_description* category is given in the following section.

```
save_ITEM_DESCRIPTION

    _category.description
;
     This category holds the descriptions of each data item.
;
    _category.id                        item_description
    _category.mandatory_code            yes
     loop_
    _category_key.name                '_item_description.name'
                                      '_item_description.description'
     loop_
    _category_group.id                  'ddl_group'
                                        'item_group'
     save_



save__item_description.description

    _item_description.description
;
     Text decription of the defined data item.
;
    _item.name                        '_item_description.description'
    _item.category_id                   item_description
    _item.mandatory_code                yes
    _item_type.code                     text
     save_
```

3.6 ITEM_ENUMERATION

The DDL category *item_enumeration* holds lists of permissible values for enumerated data items. Each enumerated value is specified in item *_item_enumeration.value*, which may each have an associated description item *_item_enumeration.detail*. The combination of items *_item_enumeration.name* and *_item_enumeration.value* form the basis for this category. The former item is defined in the parent category, *item*. Multiple unique enumeration values may be specified for each data item.

The DDL for the *item_enumeration* category is given in the following section.

```
save_ITEM_ENUMERATION

    _category.description
;
     Attributes which specify the permitted enumeration of the items.
;
    _category.id                        item_enumeration
    _category.mandatory_code            no
     loop_
    _category_key.name                '_item_enumeration.name'
                                      '_item_enumeration.value'
      loop_
     _category_group.id               'ddl_group'
                                      'item_group'
     save_


save__item_enumeration.value

    _item_description.description
;
     A permissible value, character or number, for the defined item.
;
    _item.name                        '_item_enumeration.value'
    _item.category_id                   item_enumeration
    _item.mandatory_code                yes
    _item_type.code                     any
     save_


save__item_enumeration.detail

    _item_description.description
;
     A description of a permissible value for the defined item.
;
    _item.name                        '_item_enumeration.detail'
    _item.category_id                   item_enumeration
    _item.mandatory_code                no
    _item_type.code                     text
```

```
    save_
```

3.7 ITEM_EXAMPLES

The DDL category *item_examples* is provided to hold examples associated with individual data items. An example specification consists of the text of the example, *_item_examples.case*, and an optional comment item, *_item_examples.detail*, which can be used to qualify the example. Multiple examples may be provided for each item.

The DDL for the *item_examples* category is given in the following section.

```
save_ITEM_EXAMPLES

    _category.description
;
    Attributes for describing application examples of the data item.
;
    _category.id                         item_examples
    _category.mandatory_code             no
     loop_
    _category_key.name                 '_item_examples.name'
                                       '_item_examples.case'
     loop_
     _category_group.id                'ddl_group'
                                       'item_group'
     save_


save__item_examples.case

    _item_description.description
;
     An example application of the defined data item.
;
    _item.name                          '_item_examples.case'
    _item.category_id                    item_examples
    _item.mandatory_code                 no
```

```
    _item_type.code                      text
     save_


save__item_examples.detail

    _item_description.description
;
     A description of an example specified in _item_example.case
;
    _item.name                         '_item_examples.detail'
    _item.category_id                    item_examples
    _item.mandatory_code                 no
    _item_type.code                      text
     save_
```

3.8 ITEM_LINKED

The *item_linked* category defines parent-child relationships between data items. This provides the mechanism to specify the relationships between data items that may exist in multiple categories. Link relationships are most commonly defined between key items, which form the bases for many different categories.

In the DDL presented here, all child relationships are expressed within the parent category. Because the item, *_item_linked.parent_name*, has been defined as an implicit item, the child relationships can be specified most economically in the parent category where the parent item name can be automatically inferred. If link relationships are specified in a child category, then both parent and child item names must be specified.

Both parent and child item names in this category are children of *_item.name* which ensures that all link relationships can be properly resolved. However, it is possible to define cyclical link relationships within this category. Any implementation of this DDL category should include a method to check for the existence of such pathological cases.

The DDL for the *item_linked* category is given in the following section.

```
save_ITEM_LINKED

    _category.description
;
     Attributes which describe how equivalent data items are linked
```

```
     within categories and across different categories.
;
    _category.id                        item_linked
    _category.mandatory_code            no
     loop_
    _category_key.name                 '_item_linked.child_name'
                                       '_item_linked.parent_name'
     loop_
    _category_group.id                 'ddl_group'
                                       'item_group'
     save_


save__item_linked.child_name

    _item_description.description
;
     Name of the child data item.
;
     save_


save__item_linked.parent_name

    _item_description.description
;
     Name of the parent data item.
;
     save_
```

3.9 ITEM_METHODS

The *item_methods* category is used to associate method identifiers with data items. Any number of unique method identifiers may be associated with a data item. The method identifiers reference the full method definitions in the parent *method_list* category.

The DDL for the *item_methods* category is given in the following section.

```
save_ITEM_METHODS

    _category.description
;
    Attributes specifying the association between data items and methods.
;
```

```
    _category.id                           item_methods
    _category.mandatory_code               no
     loop_
    _category_key.name
                                       '_item_methods.method_id'
                                       '_item_methods.name'

     loop_
    _category_group.id
                                       'ddl_group'
                                       'item_group'
     save_
```

3.10 ITEM_RANGE

The *item_range* category defines a restricted range of permissible values for a data item. The restrictions are specified as one or more sets of the items *_item_range.minimum* and *_item_range.maximum*. These items give the lower and upper bounds for a permissible range. To specify that an item value may be equal to the upper or lower bound or a range, the minimum and maximum values of the range are equated. The special STAR value indicating that a data value is not appropriate can be used to avoid expressing an upper or lower bound value. When limits are applied to character data, comparisons are made following the collating sequence of the character set. When limits are applied to abstract data types, then methods must be provided to define any comparison operations which must be performed to check the boundary conditions.

The DDL for the *item_range* category is given in the following section.

```
save_ITEM_RANGE

    _category.description
;
     The range of permissible values of a data item.  When multiple
     ranges are specified they are interpreted sequentially
     using a logical OR.  To specify that an item value may be
     equal to a boundary value,  specify an item range where the
     maximum and mimimum values equal the boundary value.
;
    _category.id                           item_range
    _category.mandatory_code               no
     loop_
```

```
    _category_key.name                  '_item_range.name'
                                        '_item_range.minimum'
                                        '_item_range.maximum'
     loop_
     _category_group.id               'ddl_group'
                                      'item_group'
     save_


save__item_range.minimum

    _item_description.description
;
     Minimum permissible value of a data item or the lower bound
     of a permissible range.  ( minimum value <  data value)
;
    _item.name                        '_item_range.minimum'
    _item.category_id                   item_range
    _item.mandatory_code                no
    _item_type.code                     any
     save_


save__item_range.maximum

    _item_description.description
;
     Maximum permissible value of a data item or the upper bound
     of a permissible range.  ( maximum value >  data value)
;
    _item.name                        '_item_range.maximum'
    _item.category_id                   item_range
    _item.mandatory_code                no
    _item_type.code                     any
     save_
```

3.11 ITEM_RELATED

The *item_related* category describes specific relationships that exist between data items. These relationships are distinct from the parent-child relationships that are expressed in the *item_linked* category (Section 3.8). The related item is identified as *_item_related.related_name* which is a child of *_item.name*. The item relationships defined by *_item_related.function_code* in this category include some of the following: an item is related to another item by conversion factor; an item is a replacement for another item, an item is replaced by another item, an item is an alternative expression of an item; items which differ only in some convention of their expression; and items which express a set of related characteristics. One can also identify that the declaration of an item is mutually exclusive with its alternative item. Multiple related items can be associated with each data item, and multiple relationship codes can be specified for each related item.

The DDL for the *item_related* category is given in the following section.

```
save_ITEM_RELATED

    _category.description
;
    Attributes which specify recognized relationships between data items.
;
    _category.id                         item_related
    _category.mandatory_code             no
     loop_
    _category_key.name                 '_item_related.name'
                                       '_item_related.related_name'
                                       '_item_related.function_code'
     loop_
    _category_group.id                 'ddl_group'
                                       'item_group'
     save_


save__item_related.related_name

    _item_description.description
;
     Identifies a data item by name which is closely related to the
     defined data item by the manner described by _item_related.function_code
;
     save_


save__item_related.function_code

    _item_description.description
;
     The code for the type of relationship of the item identified by
     _item_related.name and the defined item.
```

```
     ALTERNATE  indicates that the item identified in
     _item_related.related_name is an alternative expression in terms
     of its application and attributes to the item in this definition.

     ALTERNATE_EXCLUSIVE indicates that the item identified in
     _item_related.related_name is an alternative expression in terms
     of its application and attributes to the item in this definition.
     Only one of the alternative forms may be specified.

     CONVENTION  indicates that the item identified in
     _item_related.related_name differs from the defined item only
     in terms of a convention in its expression.

     CONVERSION_CONSTANT indicates that the item identified in
     _item_related.related_name differs from the defined item only
     by a known constant.

     CONVERSION_ARBITRARY indicates that the item identified in
     _item_related.related_name differs from the defined item only
     by a arbitrary constant.

     REPLACES  indicates that the defined item replaces the item identified
     in _item_related.related_name.

     REPLACEDBY  indicates that the defined item is replaced by the
     item identified in  _item_related.related_name.

     ASSOCIATED_VALUE indicates that the item identified in
     _item_related.related_name is meaningful when associated with the
     defined item.

     ASSOCIATED_ESD indicates that the item identified in
     _item_related.related_name is the estimated standard deviation of
     of the defined item.
;
   _item.name                        '_item_related.function_code'
   _item.category_id                   item_related
   _item.mandatory_code                yes
   _item_type.code                     code

    loop_
   _item_enumeration.value
   _item_enumeration.detail
        alternate           'alternate form of the item'
```

```
        alternate_exclusive  'mutually exclusive alternate form of the item'
        convention           'depends on defined convention'
        conversion_constant  'related by a known conversion factor'
        conversion_arbitrary 'related by a arbitrary conversion factor'
        replaces             'a replacement definition'
        replacedby           'an obsolete definition'
        associated_value     'a meaningful value when related to the item'
        associated_esd       'an estimated standard deviation of the item'
     save_
```

3.12 ITEM_STRUCTURE

The *item_structure* category holds a code which identifies a structure definition that is associated with a data item. A structure in this context is a reusable matrix or vector definition declared in category *item_structure_list* (Section 3.13). The data item *_item_structure.code* is a child of the item *_item_structure_list.code*. The item *_item_structure.code* provides an indirect reference into the list of structure type definitions in category *item_structure_list*. The *_item_structure.organization* item describes the row/column precedence of the matrix organization.

The DDL for the *item_structure* category is given in the following section.

```
save_ITEM_STRUCTURE

    _category.description
;
    This category holds the association between data items and
    named vector/matrix declarations.
;
    _category.id                          item_structure
    _category.mandatory_code               no
    _category_key.name                    '_item_structure.name'
     loop_
    _category_group.id                    'ddl_group'
                                          'item_group'
    save_


save__item_structure.organization

    _item_description.description
;
     Identifies if the struct is defined in column or row major order.
```

```
     Only the unique elements of symmetric matrices are specified.
;

    _item.mandatory_code                    yes
    _item_type.code                         code
     loop_
    _item_enumeration.value
    _item_enumeration.detail
                           'columnwise'     'column major order'
                           'rowwise'        'row major order'
    save_
```

3.13 ITEM_STRUCTURE_LIST

The *item_structure_list* category holds definitions of matrices and vectors that can be associated with data items. A component of the key for this category is *_item_type_list.code*, which is referenced by *_item_structure.code* to assign a structure type to a data item.

The definition of a structure involves the specification of a length for each dimension of the matrix structure. The combination of items *_item_structure_list.code* and *_item_structure_list.index* forms the basis for this category. The latter index item is the identifier for the dimension, hence multiple unique dimensions can be specified for each structure code. The length of each dimension is assigned to *_item_structure_list.dimension*.

The DDL for the *item_structure_list* category is given in the following section.

```
save_ITEM_STRUCTURE_LIST

    _category.description
;
     This category holds a description for each structure type.
;
    _category.id                          item_structure_list
    _category.mandatory_code              no

     loop_
    _category_key.name                    '_item_structure_list.code'
                                          '_item_structure_list.index'
     loop_
    _category_group.id                    'ddl_group'
                                          'item_group'
```

```
    save_

save__item_structure_list.code

    _item_description.description
;
     The name of the matrix/vector structure declaration.
;
     loop_
    _item.name
    _item.category_id
    _item.mandatory_code
        '_item_structure_list.code'   item_structure_list   yes
        '_item_structure.code'        item_structure        yes

    _item_linked.parent_name            '_item_structure_list.code'
    _item_linked.child_name             '_item_structure.code'
    _item_type.code                       code

    save_

save__item_structure_list.index

    _item_description.description
;
    Identifies the one based index of a row/column of the structure.
;
    _item.name                          '_item_structure_list.index'
    _item.category_id                     item_structure_list
    _item.mandatory_code                  yes
     loop_
    _item_range.minimum
    _item_range.maximum
     1   1
     1   .
    _item_type.code                       int

save_

save__item_structure_list.dimension

    _item_description.description
;
    Identifies the length of this row/column of the structure.
;
```

```
    _item.name                       '_item_structure_list.dimension'
    _item.category_id                  item_structure_list
    _item.mandatory_code               yes
     loop_
    _item_range.minimum
    _item_range.maximum
     1   1
     1   .
    _item_type.code                    int

    save_
```

3.14 ITEM_SUB_CATEGORY

The *item_sub_category* category is used to assign subcategory membership for data items. A data item may belong to any number of subcategories. Each subcategory must be defined in category *subcategory*.

The DDL for the *item_sub_category* category is given in the following section.

```
save_ITEM_SUB_CATEGORY

    _category.description
;
     This category assigns data items to subcategories.
;
    _category.id                         item_sub_category
    _category.mandatory_code             no
     loop_
    _category_key.name                  '_item_sub_category.id'
                                        '_item_sub_category.name'
     loop_
    _category_group.id                  'sub_category_group'
                                        'item_group'
     save_
```

3.15 ITEM_TYPE

The *item_type* category holds a code which identifies the data type of each data item. The data item *_item_type.code* is a child of the item, *_item_type_list.code*. Data type definitions are actually made in the *item_type_list* parent category. The item, *_item_type.code*, provides an indirect reference into the list of data type definitions in category *item_type_list*. This indirect reference is provided as a convenience to avoid the redeclaration of the full data type specification for each data item. The key item for this category is *_item_type.name* which is defined in the parent category, *item*. Only one data type may be specified for a data item.

The DDL for the *item_type* category is given in the following section.

```
save_ITEM_TYPE

    _category.description
;
     Attributes for specifying the data type code for each data item.
;
    _category.id                         item_type
    _category.mandatory_code             no
    _category_key.name                  '_item_type.name'
     loop_
    _category_group.id
                                        'ddl_group'
                                        'item_group'
     save_
```

3.16 ITEM_TYPE_CONDITIONS

The category *item_type_conditions* defines special conditions applied to a data item type. This category has been included in order to comply with previous applications of STAR and CIF. Since the constructions that are embodied in this category are antithetical to the data model that underlies this DDL, it is recommended that this category only be used for the purpose of parsing existing data files and dictionaries.

The DDL for the *item_type_conditions* category is given in the following section.

```
save_ITEM_TYPE_CONDITIONS

    _category.description
;
    Attributes for specifying additional conditions associated with
    the data type of the item.
;
    _category.id                         item_type_conditions
```

```
    _category.mandatory_code             no
    _category_key.name                 '_item_type_conditions.name'
     loop_
    _category_group.id                 'ddl_group'
                                       'item_group'
                                       'compliance_group'
     save_


save__item_type_conditions.code

    _item_description.description
;
     Codes defining conditions on the _item_type.code specification.

     'esd' permits a number string to contain an appended standard
           deviation number enclosed within parentheses. E.g. 4.37(5)

     'seq' permits data to be declared as a sequence of values
           separated by a comma <,> or a colon <:>.
           * The sequence v1,v2,v3,. signals that v1, v2, v3, etc.
             are alternative values or the data item.
           * The sequence v1:v2 signals that v1 and v2 are the boundary
             values of a continuous range of values. This mechanism
             was used to specify permitted ranges of an item in
             previous DDL versions.
      Combinations of alternate and range sequences are permitted.
;
    _item.name                         '_item_type_conditions.code'
    _item.category_id                    item_type_conditions
    _item.mandatory_code                 yes
    _item_type.code                      code
     loop_
    _item_enumeration.value
    _item_enumeration.detail

          none   'no extra conditions apply to this data item'
          esd    'numbers may have esd values appended within ()'
          seq    'data may be declared as a comma or colon separated sequence'

     save_
```

3.17 ITEM_TYPE_LIST

The *item_type_list* category holds the list of item data type definitions. The key item in this category is *_item_type_list.code*. Data types are associated with data items by references to this key from the *item_type* category. One of the data type codes defined in this category must be assigned to each data item.

The definition of a data type consists of the specification of the item's primitive type and a regular expression [1] which defines the pattern that must be matched by any occurrence of the item. The primitive type code, *_item_type_list.primitive_code*, can assume values of *char*, *numb*, and *null*. This code is provided for backward compatibility with STAR and CIF applications that employ loose data typing. The data item *_item_type_list.construct* holds the regular expression that must be matched by the data type. Simple regular expressions can be used to define character fields of restricted width, floating point and integer formats.

MIF [6] applications have extended the notion of the regular expression to include data item components. This permits the construction of complex data items from one or more component data items using regular expression algebra. These extended regular expressions are defined in category *item_type_conditions*.

The DDL for the *item_type_list* category is given in the following section.

```
save_ITEM_TYPE_LIST

    _category.description
;
     Attributes which define each type code.
;
    _category.id                         item_type_list
    _category.mandatory_code             no
    _category_key.name                 '_item_type_list.code'
     loop_
    _category_group.id                  'ddl_group'
                                        'item_group'
     save_

save__item_type_list.code

    _item_description.description
;
     The codes specifying the nature of the data value.
```

```
;
     loop_
    _item.name
    _item.category_id
    _item.mandatory_code
         '_item_type_list.code'   item_type_list  yes
         '_item_type.code'        item_type       yes

    _item_type.code                              code
    _item_linked.child_name                    '_item_type.code'
    _item_linked.parent_name                   '_item_type_list.code'
     save_


save__item_type_list.primitive_code

    _item_description.description
;
     The codes specifying the primitive type of the data value.
;
    _item.name                        '_item_type_list.primitive_code'
    _item.category_id                   item_type_list
    _item.mandatory_code                yes
    _item_type.code                     code
     loop_
    _item_enumeration.value
    _item_enumeration.detail
                       numb  'numerically-interpretable string'
                       char  'character or text string (case-sensitive)'
                       uchar 'character or text string (case-insensitive)'
                       null  'for dictionary purposes only'
     save_

save__item_type_list.construct

    _item_description.description
;
    When a data value can be defined as a pre-determined sequence of
    characters, or optional characters, or data names (for which the
    definition is also available), it is specified as a construction.
    The rules of construction conform to the the regular expression
    (REGEX) specificatiopns detailed in the IEEE document P1003.2
    Draft 11.2 Sept 1991 (ftp file '/doc/POSIX/1003.2/p121-140').
    Resolved data names for which _item_type_list.construct
    specifications exist are replaced by these constructions,
```

```
    otherwise the data name string is not replaced.
;
    _item.name                         '_item_type_list.construct'
    _item.category_id                    item_type_list
    _item.mandatory_code                 no
    _item_type.code                      char
    _item_examples.case                '{_year}-{_month}-{_day}'
    _item_examples.detail              'typical construction for _date'
     save_

save__item_type_list.detail

    _item_description.description
;
     An optional description of the data type
;
    _item.name                         '_item_type_list.detail'
    _item.category_id                    item_type_list
    _item.mandatory_code                 no
    _item_type.code                      text
     save_
```

3.18 An Example of the ITEM_TYPE_LIST Category

The following example section illustrates the data types that are defined within this DDL. The DDL uses a number of character data types which have subtly different definitions. For instance, the data type identified as *code* defines a single word character string; *char* extends the *code* type with the addition of a white space character; and *text* extends the *char* type with the addition of a new line character. Two special character data types *name* and *idname* are used to define the full STAR data name and the STAR name components, respectively. The data type *any* is used to match any potential data type. This type is used for data items that may hold a variety of data types. The data type *int* is defined as one of more decimal digits, and the yyyy-mm-dd type defines a date string.

```
#
#               DATA TYPE CONVERSION TABLE
#               --------------------------
#
     loop_
    _item_type_list.code
    _item_type_list.primitive_code
```

```
    _item_type_list.detail
    _item_type_list.construct
     code    char      'A single word'
      '[^\t\n "]*'
     char    char      'A single line of text'
      '[^\n]*'
     text    char      'Text which may span lines'
      '.*'
     int     numb      'Unsigned integer data'
      '[0-9]+'
     name    uchar     'A data item name (restrictive type)'
      '_[_A-Za-z0-9]+[.][][_A-Za-z0-9%/-]+'
aliasname    uchar     'A DDL 1.4 data item name (less restrictive type)'
      '_[^\t\n "]+'
     idname  uchar     'A data item name component or identifier'
      '[_A-Za-z0-9]+'
     any     char      'Any data type'
      '.*'
     yyyy-mm-dd char   'A date format'
      '[0-9][0-9][0-9][0-9]-[0-9]?[0-9]-[0-9][0-9]'
```

3.19 ITEM_UNITS

The *item_units* category holds a code which identifies the system of units in which a data item is expressed. The data item *_item_units.code* is a child of the item, *_item_units_list.code*. Unit definitions are actually made in the *item_units_list* parent category. The item, *_item_units.code*, provides an indirect reference into the list of data type definitions in category *item_units_list*. This indirect reference is provided as a convenience to avoid the redeclaration of the full data type specification for each data item. The key item for this category is *_item_units.name* which is defined in the parent category, *item*. Only one type of units may be specified for a data item.

The DDL for the *item_units* category is given in the following section.

```
save_ITEM_UNITS

    _category.description
;
     Specifies the physical units in which data items are expressed.
;
    _category.id                         item_units
    _category.mandatory_code             no
```

```
    _category_key.name                 '_item_units.name'
     loop_
    _category_group.id
                                       'ddl_group'
                                       'item_group'
     save_
```

3.20 ITEM_UNITS_CONVERSION

The *item_units_conversion* category holds a table of conversion factors between the systems of units described in the *item_units_list* category. The systems of units are identified by a *from_code* and a *to_code*, which are both children of the item *_item_units_list.code*. The conversion is defined in terms of an arithmetic operator and a conversion factor, *_item_units_conversion.operator* and *_item_units_conversion.factor*, respectively.

The DDL for the *item_units_conversion* category is given in the following section.

```
save_ITEM_UNITS_CONVERSION

    _category.description
;
     Conversion factors between the various units of measure defined
     in the ITEM_UNITS_LIST category.
;
    _category.id                         item_units_conversion
    _category.mandatory_code             no
     loop_
    _category_key.name                  '_item_units_conversion.from_code'
                                        '_item_units_conversion.to_code'
     loop_
    _category_group.id                  'ddl_group'
                                        'item_group'
     save_

save__item_units_conversion.from_code

    _item_description.description
;
     The unit system on which the conversion operation is applied
     to produce the unit system specified in _item_units_conversion.to_code.

         <to_code> =  <from_code> <operator> <factor>
```

```
;
     save_

save__item_units_conversion.to_code

    _item_description.description
;
     The unit system produced after an operation is applied to the unit
     system specified by  _item_units_conversion.from_code.

         <to_code> =  <from_code> <operator> <factor>
;
     save_

save__item_units_conversion.operator

    _item_description.description
;
     The arithmetic operator required to convert between the
     unit systems:
         <to_code> =  <from_code> <operator> <factor>
;
    _item.name                        '_item_units_conversion.operator'
    _item.category_id                   item_units_conversion
    _item.mandatory_code                yes
    _item_type.code                     code
     loop_
    _item_enumeration.value
    _item_enumeration.detail
                        '+'  'addition'
                        '-'  'subtraction'
                        '*'  'multiplication'
                        '/'  'division'
     save_

save__item_units_conversion.factor

    _item_description.description
;
     The arithmetic operation required to convert between the
     unit systems:
         <to_code> =  <from_code> <operator> <factor>
;
    _item.name                        '_item_units_conversion.factor'
    _item.category_id                   item_units_conversion
```

```
    _item.mandatory_code                 yes
    _item_type.code                      any
     save_
```

3.21 ITEM_UNITS_LIST

The *item_units_list* category holds the descriptions of systems of physical units. The key item in this category is is *_item_units_list.code*. Units are assigned to data items by references to this key from the *item_units* category.

The DDL for the *item_units_list* category is given in the following section.

```
save_ITEM_UNITS_LIST

    _category.description
;
     Attributes which describe the physical units of measure
     in which data items may be expressed.
;
    _category.id                         item_units_list
    _category.mandatory_code             no
    _category_key.name                 '_item_units_list.code'
     loop_
    _category_group.id                  'ddl_group'
                                        'item_group'
     save_

save__item_units_list.code

    _item_description.description
;
     The code specifying the name of the unit of measure.
;
     loop_
    _item.name
    _item.category_id
    _item.mandatory_code
          '_item_units_list.code'              item_units_list         yes
          '_item_units.code'                   item_units              yes
          '_item_units_conversion.from_code'   item_units_conversion   yes
          '_item_units_conversion.to_code'     item_units_conversion   yes

    _item_type.code                               code
     loop_
    _item_linked.child_name
```

```
    _item_linked.parent_name
          '_item_units.code'                     '_item_units_list.code'
          '_item_units_conversion.from_code'    '_item_units_list.code'
          '_item_units_conversion.to_code'      '_item_units_list.code'

     save_


save__item_units_list.detail

    _item_description.description
;
     A description of the unit of measure.
;
    _item.name                          '_item_units_list.detail'
    _item.category_id                     item_units_list
    _item.mandatory_code                  no
    _item_type.code                       text
     save_
```

4 DDL Definitions Describing Methods

In this section the DDL categories that define the methods associated with data blocks, categories, subcategories, and items are presented. Figures 1, 4, and 3 illustrate the relationships between the method categories and other DDL categories.

4.1 METHOD_LIST

The *method_list* category defines methods that can be associated with data blocks, categories, subcategories, and items. This category attempts to capture only the essential information required to define these methods, without defining any implementation details. The implementation details are appropriately left to application dictionary developers. It is assumed here, that within a domain of dictionaries, a consistent method interface will be adopted that is tailored to the requirements of that domain. This of course complicates the sharing of methods between domains; however, it would be impossible at this time to define an implementation strategy inside the DDL that would even begin to satisfy the diverse requirements of potential DDL users. Consequently, the definition of each method is limited to its unique identifier, *_method_list.id*; a textual description, *_method_list.detail*; the source text of the method, *_method_list.inline*; the name of the language in which the method is expressed, *_method_list.language*; and a code to identify the the purpose of the method,*_method_list.code*.

The DDL for the *method_list* category is given in the following section.

```
save_METHOD_LIST

    _category.description
;
    Attributes specifying the list of methods applicable to data items,
    sub-categories, and categories.
;
    _category.id                          method_list
    _category.mandatory_code              no
    _category_key.name                  '_method_list.id'
     loop_
    _category_group.id
                                         'ddl_group'
                                         'item_group'
                                         'category_group'
```

```
     save_


save__method_list.id

    _item_description.description
;
     Identity of method in the list referenced by _method.id
;
     loop_
    _item.name
    _item.category_id
    _item.mandatory_code
         '_method_list.id'                   method_list            yes
         '_item_methods.method_id'           item_methods           yes
         '_category_methods.method_id'       category_methods       yes
         '_sub_category_methods.method_id'   sub_category_methods   yes
         '_datablock_methods.method_id'      datablock_methods      yes
    _item_type.code                      idname
     loop_
    _item_linked.child_name
    _item_linked.parent_name
         '_item_methods.method_id'             '_method_list.id'
         '_category_methods.method_id'         '_method_list.id'
         '_sub_category_methods.method_id'     '_method_list.id'
         '_datablock_methods.method_id'        '_method_list.id'
     save_

save__method_list.detail

    _item_description.description
;
     Description of application method in _method_list.id
;
    _item.name                         '_method_list.detail'
    _item.category_id                    method_list
    _item.mandatory_code                 no
    _item_type.code                      text
     save_


save__method_list.inline

    _item_description.description
;
```

```
     Inline text of a method associated with the data item.
;
    _item.name                        '_method_list.inline'
    _item.category_id                   method_list
    _item.mandatory_code                yes
    _item_type.code                     text
     save_


save__method_list.code

    _item_description.description
;
     A code that describes the function of the method.
;
    _item.name                        '_method_list.code'
    _item.category_id                   method_list
    _item.mandatory_code                yes
    _item_type.code                     code
     loop_
    _item_examples.case
    _item_examples.detail
          calculation      'method to calculate the item '
          verification     'method to verify the data item '
          cast             'method to provide cast conversion '
          addition         'method to define item + item '
          division         'method to define item / item '
          multiplication   'method to define item * item '
          equivalence      'method to define item = item '
          other            'miscellaneous method '
     save_


save__method_list.language

    _item_description.description
;
     Language in which the method is expressed.
;
    _item.name                        '_method_list.language'
    _item.category_id                   method_list
    _item.mandatory_code                yes
    _item_type.code                     code
     loop_
    _item_examples.case                 BNF
```

```
                                    C
                                    C++
                                    FORTRAN
                                    LISP
                                    PASCAL
                                    PEARL
                                    TCL
                                    OTHER
save_
```

5 DDL Definitions Describing Dictionaries and Data Blocks

In this section, the DDL categories that describe the characteristics of a dictionaries and data blocks are presented. In this context, a dictionary is defined as a group of related definitions within a STAR data block. Figure 4 illustrates the schema for these categories.

5.1 DATABLOCK

The *datablock* category holds the essential identifying information for a data block. The items recorded in this category include the name of the data block, *_datablock.id*, and a description of the block *_datablock.description*. The *_datablock.id* is the parent identifier for both *_category.implicit_key* and *_dictionary.datablock_id*.

The DDL for the *datablock* category is given in the following section.

```
save_DATABLOCK

    _category.description
;
     Attributes defining the characteristics of a data block.
;
    _category.id                        datablock
    _category.mandatory_code            no
    _category_key.name                '_datablock.id'
     loop_
    _category_group.id                 'ddl_group'
                                       'datablock_group'
     save_


save__datablock.id

    _item_description.description
;
     The identity of the data block.
;
     loop_
    _item.name
    _item.category_id
    _item.mandatory_code
```

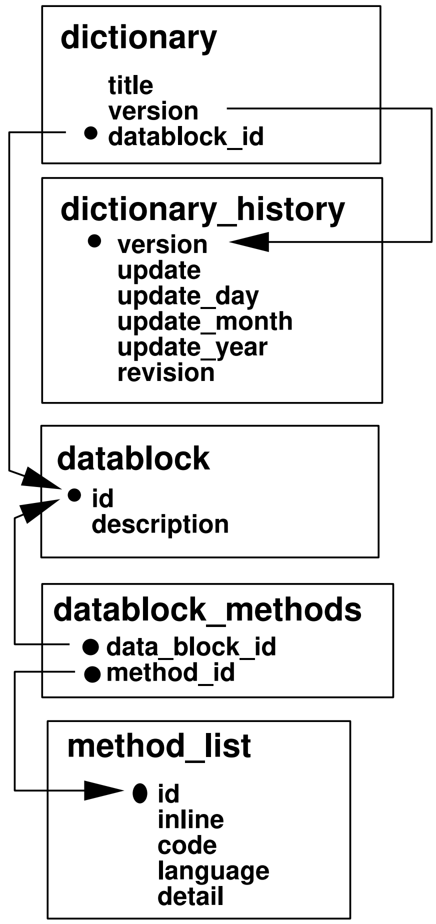

Figure 4: The schema of DDL definitions that describe the characteristics of dictionaries and data blocks. In this figure, boxes surround data items which belong to the same category. Category identifiers are given in large font and item names are given in small font. Parent-child relationships are specified by lines connecting data items with the arrow pointing at the parent item. Key items within a category are identified by a preceding black dot.

```
        '_datablock.id'                      datablock          implicit
        '_datablock_methods.datablock_id'    datablock_methods  implicit
        '_dictionary.datablock_id'           dictionary         implicit
        '_category.implicit_key'             category           implicit
   _item_type.code                           code

    loop_
    _item_linked.child_name
    _item_linked.parent_name
        '_datablock_methods.datablock_id'       '_datablock.id'
        '_dictionary.datablock_id'              '_datablock.id'
        '_category.implicit_key'                '_datablock.id'
     save_


save__datablock.description

    _item_description.description
;
     Text description of the data block.
;
    _item.name                         '_datablock.description'
    _item.category_id                    datablock
    _item.mandatory_code                 yes
    _item_type.code                      text
     save_
```

5.2 DATABLOCK_METHODS

The *datablock_methods* category is used to associate method identifiers with a data block. Any number of unique method identifiers may be associated with a data block. The method identifiers reference the full method definitions in the parent *method_list* category.

The DDL for the *datablock_methods* category is given in the following section.

```
save_DATABLOCK_METHODS

    _category.description
;
```

```
    Attributes specifying the association between data blocks and methods.
;
    _category.id                           datablock_methods
    _category.mandatory_code               no
     loop_
    _category_key.name
                                       '_datablock_methods.method_id'
                                       '_datablock_methods.datablock_id'
     loop_
    _category_group.id
                                       'ddl_group'
                                       'datablock_group'
     save_
```

5.3 DICTIONARY

The *dictionary* category holds the essential identifying information for a dictionary. The items recorded in this category include the title for the dictionary, *_dictionary.title*, the current version identifier, *_dictionary.version*, and the data block identifier in which the dictionary is defined, *_dictionary.data_block_id*. The version identifier references the parent identifier in the *dictionary_history* category in which each dictionary revision is described.

The DDL for the *dictionary* category is given in the following section.

```
save_DICTIONARY

    _category.description
;
     Attributes for specifying the dictionary title, version and
     data block identifier.
;
    _category.id                           dictionary
    _category.mandatory_code               yes
    _category_key.name                    '_dictionary.datablock_id'
     loop_
    _category_group.id
                                          'ddl_group'
                                          'datablock_group'
                                          'dictionary_group'
     save_
```

```
save__dictionary.datablock_id

    _item_description.description
;
     The identifier for the data block containing the dictionary.
;
     save_


save__dictionary.title

    _item_description.description
;
     Title identification of the dictionary.
;
    _item.name                         '_dictionary.title'
    _item.category_id                    dictionary
    _item.mandatory_code                 yes
    _item_type.code                      char
     save_



save__dictionary.version

    _item_description.description
;
     A unique version identifier for the dictionary.
;
     save_
```

5.4 DICTIONARY_HISTORY

The *dictionary_history* category holds the revision history for a dictionary. Each revision is assigned a version identifier which acts as the key item for the category. Along with the version information, a text description of the revision and date of revision must be specified.

The DDL for the *dictionary_history* category is given in the following section.

```
save_DICTIONARY_HISTORY

    _category.description
;
```

```
   Attributes for specifying the revision history of the dictionary.
;

    _category.id                         dictionary_history
    _category.mandatory_code             no
    _category_key.name                 '_dictionary_history.version'
     loop_
    _category_group.id
                                       'ddl_group'
                                       'dictionary_group'
     save_


save__dictionary_history.version

    _item_description.description
;
     A unique version identifier for the dictionary revision.
;
     loop_
    _item.name
    _item.category_id
    _item.mandatory_code
         '_dictionary_history.version'  dictionary_history     yes
         '_dictionary.version'          dictionary             yes
    _item_type.code                     char
    _item_linked.child_name           '_dictionary.version'
    _item_linked.parent_name          '_dictionary_history.version'
     save_


save__dictionary_history.update

    _item_description.description
;
     The date that the last dictionary revision took place.
;
    _item.name                         '_dictionary_history.update'
    _item.category_id                    dictionary_history
    _item.mandatory_code                 yes
    _item_type.code                      yyyy-mm-dd
     save_

save__dictionary_history.revision
```

```
    _item_description.description
;
     Text description of the dictionary revision.
;
    _item.name                          '_dictionary_history.revision'
    _item.category_id                     dictionary_history
    _item.mandatory_code                  yes
    _item_type.code                       text
     save_
```

References

[1] *Draft Standard for Information Technology: Portable Operating System Interface (POSIX) Part 2: Shells and Utilities IEEE P1003.2 Draft 11.2*, 1991.

[2] F. H. Allen, J. M. Barnard, A. F. P. Cook, and S. R. Hall. The molecular information file (mif): Initial specifications. *J. Chem. Inf. Comput. Sci.* to be submitted.

[3] S. R. Hall. The star file: A new format for electronic data transfer and archiving. *J. Chem. Inf. Comput. Sci.*, 31, 1991.

[4] S. R. Hall, F. H. Allen, and I. D. Brown. A new standard archive file for crystallography. *Acta Crystallogr.*, A47, 1991.

[5] S. R. Hall and N. Spadaccini. The star file: Detailed specifications. *J. Chem. Inf. Comput. Sci.*, 31, 1994.

[6] N. Spadacciini and S. R. Hall. Star base: Accessing star file data. *J. Chem. Inf. Comput. Sci.*, 31, 1994.