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.

2.6.6.1.16. *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 DDL2, it is recommended that this category only be used for the purpose of parsing existing data files and dictionaries.

2.6.6.1.17. *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 that 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`, `uchar`, `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.

Molecular Information File (MIF) applications (Allen *et al.*, 1995) 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 the category ITEM_TYPE_CONDITIONS.

Example 2.6.6.1 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 newline 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 or more decimal digits and the `yyyy-mm-dd` type defines a date string.

2.6.6.1.18. *ITEM_UNITS*

The ITEM_UNITS category holds a code that 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 unit may be specified for a data item.

Example 2.6.6.1. *The description of permitted data types in the DDL2 dictionary.*

```
#               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'
                '_[_A-Za-z0-9]+[.][][_A-Za-z0-9%/-]+'

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]'
```

2.6.6.1.19. *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.

2.6.6.1.20. *ITEM_UNITS_LIST*

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

2.6.6.2. DDL2 definitions describing categories

In this section, the DDL definitions that describe the properties of categories, category groups and subcategories are presented. Fig. 2.6.4.2 illustrates the organization of these categories.

2.6.6.2.1. *CATEGORY*

The category named CATEGORY contains the data items that describe the properties of collections of related data items. A DDL category is essentially a table. In this category the characteristics of the table as a whole are defined. This category includes the data items `_category.id` to identify a category name; `_category.description` to describe a category; `_category.mandatory_code` to indicate whether the category must appear in a data block; 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 key in most of the DDL categories in this section. The parent definition of the category identifier and all its child relationships are defined in this category.

Because special rules exist in the STAR grammar for the specification of data items that 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.

2.6.6.2.2. *CATEGORY_EXAMPLES*

The category named 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 key for this category includes the items **_category_examples.id** and **_category_examples.case**. The former is completely defined in the parent category named CATEGORY.

2.6.6.2.3. *CATEGORY_GROUP*

The 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. The basis for this category also includes the category identifier **_category_group.category_id**, which is completely defined in the parent category named CATEGORY.

2.6.6.2.4. *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 allow 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 identifier of the parent group, **_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.

Example 2.6.6.2 illustrates the category groups that are defined in this DDL. These include the group of categories that define categories, the group of categories defining data items and the group of categories that define properties of 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 a child of the group named **ddl_group** to which all of the base DDL categories belong.

2.6.6.2.5. *CATEGORY_KEY*

The 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. In the analogy of the category as a table, no row in a table may have duplicate values for its key data items.

Example 2.6.6.2. *Category groups defined in the DDL2 dictionary.*

```
     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.
;
```

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 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 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. Duplicate key-item values within a category are 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.

2.6.6.2.6. *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.

2.6.6.2.7. *SUB_CATEGORY*

The category SUB_CATEGORY provides data items to describe a subcategory and to associate a procedure with the subcategory (see Section 2.6.6.2.9). A subcategory is a set of data items within a category that have a particular association. A typical example would be a triad of positional coordinates *x*, *y*, *z* that are collectively assigned to a 'cartesian' subcategory.

2.6.6.2.8. *SUB_CATEGORY_EXAMPLES*

The DDL category SUB_CATEGORY_EXAMPLES holds examples of a subcategory. A subcategory example might illustrate valid instances of the items comprising the subcategory. An example specification contains the text of the example, `_sub_category_examples.case`, and an optional comment item, `_sub_category_examples.detail`, that can be used to qualify the example. The key for this category includes the items `_sub_category_examples.id` and `_sub_category_examples.case`. This compound basis permits multiple unique examples to be provided for each subcategory.

2.6.6.2.9. *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 procedure that is identified as `_sub_category_methods.method_id` may be used to validate the subcategory identified as `_sub_category_methods.sub_category_id`. Subcategory validation may be required in instances where conditions are placed on the values of data items within the subcategory that 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 the category METHOD_LIST.

2.6.6.3. DDL2 definitions describing methods

In this section, the DDL categories that define the methods associated with data blocks, categories, subcategories and items are presented. Figs. 2.6.4.1, 2.6.4.2 and 2.6.4.3 illustrate the relationships between the method categories and other DDL categories.

2.6.6.3.1. *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 purpose of the method, `_method_list.code`.

2.6.6.4. DDL2 definitions describing dictionaries and data blocks

In this section, the DDL categories that describe the characteristics of dictionaries and data blocks are presented. In this context, a dictionary is defined as a group of related definitions within a STAR data block. Fig. 2.6.4.3 illustrates the organization for these categories.

2.6.6.4.1. *DATABLOCK*

The DATABLOCK category holds the essential identifying information for a data block: 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 former guarantees that the identifier for the data block, and hence the dictionary, is added implicitly to the key of each category.

2.6.6.4.2. *DATABLOCK_METHODS*

The DATABLOCK_METHODS category may be associated with a data block. The method identifiers reference the full method definitions in the parent METHOD_LIST category.

2.6.6.4.3. *DICTIONARY*

The DICTIONARY category holds the essential identifying information for a data 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.datablock_id`. The version identifier references the parent identifier in the DICTIONARY_HISTORY category in which each dictionary revision is described.

2.6.6.4.4. *DICTIONARY_HISTORY*

The DICTIONARY_HISTORY category holds the revision history for a dictionary. Each revision is assigned a version identifier that 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.

References

Allen, F. H., Barnard, J. M., Cook, A. F. P. & Hall, S. R. (1995). *The Molecular Information File (MIF): core specifications of a new standard format for chemical data. J. Chem. Inf. Comput. Sci.* **35**, 412–427.

Hall, S. R. (1991). *The STAR File: a new format for electronic data transfer and archiving. J. Chem. Inf. Comput. Sci.* **31**, 326–333.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hall, S. R. & Cook, A. P. F. (1995). *STAR dictionary definition language: initial specification. J. Chem. Inf. Comput. Sci.* **35**, 819–825.

Hall S. R. & Spadaccini, N. (1994). *The STAR File: detailed specifications. J. Chem. Inf. Comput. Sci.* **34**, 505–508.

Westbrook, J. D. & Hall, S. R. (1995). *A dictionary description language for macromolecular structure.* http://ndbserver.rutgers.edu/mmcif/ddl/.