International Tables for Crystallography (2006). Vol. G, Section 5.2.2.2, pp. 488–489.

5.2. STAR File utilities

BY N. SPADACCINI, S. R. HALL AND B. MCMAHON

5.2.1. Introduction

The STAR File, described in Chapter 2.1, has a simple format intended to allow the flexible and extensible representation of data without regard to specific data models. In crystallography and related disciplines, the restricted format chosen for the Crystallographic Information File (CIF, Chapter 2.2) and Crystallographic Binary File (CBF, Chapter 2.3) lends itself to rather flat data models. In particular, the relationships between data items enforced through DDL2 dictionaries in applications such as mmCIF (Chapter 3.6) are essentially equivalent to the data structures and relationships of a relational database. Of course, properly normalized relational tables can represent a hierarchy of structure, although this may not be an efficient representation.

There are other applications, such as the molecular information file (MIF, Chapter 2.4), that make use of additional features of the STAR File, such as multiple-level loop structures, global variable scoping and data-instance encapsulation in save frames. These applications may more efficiently represent certain hierarchical or object-oriented data models.

While particular applications require software tools tailored to their specific purposes, it is helpful to have programs or libraries capable of manipulating arbitrary STAR File data, relying solely on the syntax rules and format of the STAR File and taking no account of the semantic content of the included data.

In this chapter, the stand-alone program *Star_Base* is described in detail. This program uses a local query language to demonstrate the ability to retrieve or re-order data with their associated context. There is also a brief review of *Star.vim* and *StarMarkUp*, applications for editing and browsing STAR Files. The chapter concludes by reviewing a number of object classes and libraries for a variety of STAR and generalized CIF applications: prototypical approaches *OOSTAR* and *CIF++*, *CIFOBJ* and *starlib* used by major macromolecular data repositories, and the document-object model package *StarDOM*.

5.2.2. Data instances and context

In a STAR File, a data item consists of a *value*, which is a simple ASCII character string, and an associated identifier or *data name* which precedes the value, and is invariably an ASCII character string beginning with an underscore character and not including any white-space character, such as `_date` or `_chemical_formula_sum`. (The detailed and formal syntax rules for STAR Files are given in Chapter 2.1.)

5.2.2.1. Single and multiple values

A data item may have a single value, in which case the data name may immediately precede the data value, separated only by white space, *e.g.*

```
_chapter_title  'STAR File utilities'
```

Alternatively, a data item may occur multiple times, in a vector or a list. In such a case, the data identifiers appear in a loop header and the values follow in the order of presentation in the loop header. For the simple example of a tabular array, the loop header plays the role of column header, *e.g.*

```
loop_
   _chapter_number
   _chapter_title
       5.2  'STAR File utilities'
       5.3  'Syntactic utilities for CIF'
```

Here the instances of the data item identified by the data name `_chapter_number` have two values, `5.2` and `5.3`. Likewise the instances of the data item identified by `_chapter_title` have two values.

Note an important point: the example has been chosen to suggest to the reader a tabular relationship between the two data items, and in many STAR File applications such a relationship is intended and perhaps formalized through an external dictionary defining the relationships between these data names. However, *the existence of such a relationship is not mandated by the STAR File syntax*. It is legitimate for a generic STAR application to extract a single data item from such an aggregated loop without making any supposition about its relationship with other data items in the same loop. (It should be emphasized that in practice such physical juxtaposition of data items will almost invariably represent a real relationship, and that most application-specific programming will depend on this fact; but it is not an essential component of STAR in its most abstract form.)

It is also axiomatic that the ordering of the multiple values within a list structure has no intrinsic significance in the STAR paradigm. (Again, specific applications may override this by enforcing an ordering, but this is not fundamental to STAR.)

5.2.2.2. Loop packets and context within lists

Where multiple data names are declared in a loop header, STAR does however enforce the notion of a 'loop packet'. The loop packet is the data structure including all individual data values at a particular iteration through the loop. Hence, in the simple example above, `5.2` and `STAR File utilities` comprise the tuple of values in a single loop packet. For the single level of loop considered so far, the loop packet plays the role of a table row.

For nested loops, the situation is more complex. Consider Fig. 5.2.2.1, which is an example of quantum chemistry basis sets for hydrogen and lithium. (The examples in this chapter are derived from various test applications, and do not represent specific adopted exchange protocols in the selected subject areas.) For each element, a list of basis sets is presented, each containing a set of parameters and a table of functional values. At the outermost level of looping in this example, a loop packet comprises all the data associated with an individual atom type, for example hydrogen. At the next inner level of looping, a loop packet corresponds to an individual basis set (including its embedded table of

Affiliations: N. SPADACCINI, School of Computer Science and Software Engineering, University of Western Australia, 35 Stirling Highway, Crawley, Perth, WA 6009, Australia; SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia; BRIAN MCMAHON, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England.

Copyright © 2006 International Union of Crystallography

```
data_Gaussian

loop_
     _basis_set_atomic_name
     _basis_set_atomic_symbol
     _basis_set_atomic_number
     _basis_set_atomic_mass
    loop_
     _basis_set_contraction_scheme
     _basis_set_funct_per_contraction
     _basis_set_primary_reference
     _basis_set_source_exponent
     _basis_set_source_coefficient
     _basis_set_atomic_energy
        loop_
     _basis_set_function_exponent
     _basis_set_function_coefficient

      hydrogen    H    1    1.0079
#     --------
 (2)->[2]   1:    PKC1.1.1   R44 .       -0.485813
      1.3324838E+01    1.0
      2.0152720E-01    1.0 stop_
 (2)->[2]   1:    PKC1.2.1   R33 .       -0.485813
      1.3326990E+01    1.0
      2.0154600E-01    1.0 stop_
 (2)->[1]   2     PKC1.14.1  R24 R24     -0.485813
      1.3324800E-01    2.7440850E-01
      2.0152870E-01    8.2122540E-01 stop_
 (3)->[2]   2:1   PKC1.23.1  R75 R75     -0.496979
      4.5018000E+00    1.5628500E-01
      6.8144400E-01    9.0469100E-01
      1.5139800E-01    1.0000000E+01 stop_ stop_

    lithium    Li   3     6.94
#    -------
 (4)->[4]   1:    PKC3.1.1   R44 .       -7.376895
       3.4856175E+01     1.0
       5.1764114E+00     1.0
       1.0514394E+00     1.0
       4.7192775E-02     1.0 stop_
 (9,4)->[3,2]  7:2:1,3:1
                  PKC3.9.1   R2  R98     -7.431735
       921.271    0.001367       138.730    0.010425
       31.9415    0.049859       9.35329    0.160701
       3.15789    0.344604       1.15685    0.425197
       0.44462    0.169468       0.44462   -0.222311
       0.076663   1.116477       0.028643   1.0
       1.488      0.038770       0.2667     0.236257
       0.07201    0.830448       0.02370    1.0     stop_
 (4,3)->[3,2]   4:2:1,2:1
                  PKC3.30.1  R77 R77     -7.419509
       1.09353E+02    1.90277E-02 1.64228E+01    1.30276E-01
       3.59415E+00    4.39082E-01 9.05297E-01    5.57314E-01

       5.40205E-01   -2.63127E-01 1.02255E-01    1.14339E+00
       2.85645E-02    1.00000E+00

       5.40205E-01    1.61546E-01 1.02255E-01    9.15663E-01
       2.85645E-02    1.00000E+00  stop_ stop_
```

Fig. 5.2.2.1. Example quantum chemistry basis set functions in STAR File format.

coefficients). At the innermost loop level, a loop packet is simply a row within a table of exponents and coefficients of the basis set function.

If one were to treat this example file as a database of indeterminate structure and query the values associated with one of the data names, for example, `_basis_set_function_exponent`, one would retrieve a series of strings `1.3324838E+01`, `2.0152720E-01` *etc.* However, the value strings in themselves are insufficient to allow the reconstitution of any data structure in the file. One also needs an expression of the levels within the nested loop structure at which the values were located, and an indication that they were associated with different packets of information at those various levels. This additional information about the context of each value is sufficient to determine its position within the data structure

```
data_Gaussian
loop_
    loop_
        loop_
     _basis_set_function_exponent
        stop_
    stop_

        1.3324838E+01 2.0152720E-01 stop_
        1.3326990E+01 2.0154600E-01 stop_
        1.3324800E-01 2.0152870E-01 stop_
        4.5018000E+00 6.8144400E-01 1.5139800E-01 stop_
    stop_

        3.4856175E+01 5.1764114E+00 1.0514394E+00
        4.7192775E-02 stop_

        921.271 138.730 31.9415 9.35329 3.15789 1.15685
        0.44462 0.44462 0.076663 0.028643 1.488 0.2667
        0.07201 0.02370 stop_

        1.09353E+02 1.64228E+01 3.59415E+00 9.05297E-01
        5.40205E-01 1.02255E-01 2.85645E-02 5.40205E-01
        1.02255E-01 2.85645E-02 stop_
    stop_
```

Fig. 5.2.2.2. Retrieval from the example file in Fig. 5.2.2.1 of the value of `_basis_set_function_exponent` with associated context.

without any other *a priori* information regarding the data model. The context is most easily expressed by listing the output values in STAR File format.

Fig. 5.2.2.2 is an output listing of the requested values for this example, where the context is expressed as the innermost of three nested loop levels and distinct packets at this level are indicated. It will be seen also that by tracing the disposition of `stop_` words the embedding within higher-level loop packets can also be inferred.

5.2.2.3. Context in data sets

Another indicator of context in the previous example is the data-block header, which was reproduced in the output of Fig. 5.2.2.2.

The STAR File allows data instances in three types of location: in a data block, in a save frame or in a global block.

The usual way to partition a STAR File is by data blocks; each such block represents a data set in which a data name (associated with a single or multiple values) may be declared once only.

Data blocks may include save frames. A save frame is an encapsulated subsidiary data set, effectively insulated from the contents of the surrounding data block, in which data items may occur that have the same names as items in the parent data block. Indeed, 'parent' is potentially a misleading term, since no relationship is implied between the data within a save frame and those in the data block in which the save frame occurs. A reference to a save frame may, however, occur as a data *value* within the data block where the save frame is specified. Recall from Section 2.1.3.6 that save frames within a data block are uniquely identified by the *frame-code* header.

Global blocks may also occur in a STAR File, preceding or interspersed between data blocks. For each data item defined within a global block, that definition is inherited by each succeeding data block that does not contain an internal definition of a data item with the same name. If there is a definition of a data item with the same name within a data block, that internal definition overrides the global definition within that data block. The situation is then re-evaluated in the next data block. If that data block does not contain an internal definition, the global definition holds.