IUCr forums

Discussions on IUCr projects and activities

Draft API specifications

Snapshot of recent discussion thread, 14 August 2013

CIF Data Model

by jcbollinger » Fri Jan 20, 2012 9:05 pm

With James's permission, I would like to introduce an initiative of his that he and I have been working on: the CIF data model. This is a *logical* model with which the structure and data content of any CIF can be described, and I present it to the group with the intention that it should serve at least two purposes:

1) most importantly, to provide a common vocabulary with which to discuss the form and parts of CIFs, but also

2) as a reference against which whatever physical data models we later consider can be validated and compared.

The need for (1) particularly struck me as I was compiling the requirements list.

I emphasize that this logical model is not proposed to be directly translated into a physical model. On the other hand, any physical model that adequately covers the CIF API's problem space by necessity has a subset that maps to the logical model presented here:

(1) A CIF FILE is a set of zero or more (BLOCKNAME, DATABLOCK) pairs, wherein all the BLOCKNAMEs are distinct
(2) A DATABLOCK is a CONTAINER plus zero or more (FRAMENAME, SAVEFRAME) pairs, wherein all the FRAMENAMEs are distinct
(3) BLOCKNAMEs and FRAMENAMEs are sequences of one or more Unicode code points from the CIF character set, excluding CIF whitespace
(4) A SAVEFRAME is a CONTAINER
(5) A CONTAINER is a set of zero or more LOOPs satisfying the constraints described in (9) and (10)
(6) A LOOP is a collection of one or more PACKETS for a DOMAIN
(7) A DOMAIN is a set of DATANAMEs (see note ii)
(8) A PACKET for a given DOMAIN is a mapping from each DATANAME in that DOMAIN to a DATAVALUE
(9) Each CONTAINER contains at most one single-PACKET loop (see note iii)
(10) All DOMAINs of all LOOPs in the same CONTAINER must be disjoint
(11) A DATAVALUE is any one of TABLE, LIST, MAYBENUMB, CHAR, NUMB, NULL, or UNKNOWN
(12) A TABLE is a set of zero or more (KEY, DATAVALUE) pairs, wherein all the KEYs are distinct
(13) A KEY is a CHAR
(14) A LIST is an ordered sequence of zero or more DATAVALUEs
(15) MAYBENUMB is a (NUMB, CHAR) pair, constituting separate numeric and character interpretations of one value (see note v)
(16) A CHAR value is a sequence of zero or more Unicode code points
(17) A NUMB value is a (NUMBER, NUMBER) pair, constituting a numeric value and its standard uncertainty
(18) NUMBER is a numeric value
(19) NULL and UNKNOWN are primitive values

Note that:
(i) The word "set" is used to imply order-independence.
(ii) DATANAME is the same as "data name" as described in the CIF 2.0 syntax specification.

(iii) The one allowed single-packet loop per CONTAINER comprises all 'top-level' key-value pairs as well as the joined contents of all syntactic one-packet loops in that CONTAINER, and that means
(iv) distinct physical CIFs have the same representation in this data model when they differ only in whether data are presented as one-packet loops or in flattened form.
(v) MAYBENUMB is the result of parsing a data value having numeric form, prior to any validation to establish its correct data type. NUMB, on the other hand, is a value that is known to be numeric, whether through validation or some other (e.g. programmatic) means.
(vi) This formulation of the model is not intended to define how code point sequences are judged to be distinct or equal.

Re: CIF Data Model

by **jamesrhester** » Mon Feb 04, 2013 2:30 am

One significant difference between the data model posted above and the "relational datamodel" is that, for the relational datamodel, values in any given column must be drawn from the same "domain", where "domain" in this context roughly means type. The various DDL dictionaries indeed specify a particular type for each dataname, so from the point of view of current CIF practise, columns are indeed expected to draw values from the same domain.

Question 1: Do we wish to make this domain requirement part of the datamodel? Or do we wish to view it is an optional restraint that may be imposed by a dictionary language?

If a domain requirement is not part of the datamodel, then a program wishing to cleanly separate the dictionary layer from the datamodel (e.g. a generic parser) must keep track of the datatype of every datavalue, in the event that the calling application is based on a dictionary that allows a variety of datatypes within a single column.

If a domain requirement is part of the datamodel, then it would be a syntax or datamodel error to mix types in a single column, for example 'MAYBENUMB' and 'CHAR' or TABLE and LIST. To a certain extent subtypes could be subsumed, so for example a column containing single values and values in lists could promote the isolated values to single-membered lists without error. Possible rules for mixed types in a single column:
TABLE and non-TABLE: error
LIST and any non-compound value: non-compound values promoted to single-member list containing that value / error
MAYBENUMB and CHAR: CHAR
MAYBENUMB and NUMB: NUMB
NUMB and CHAR: error

I have deliberately left NULL and UNKNOWN out of the list, as these are special values that are part of every domain.

Note that when parsing a text stream, the only potential issue that arises is between MAYBENUMB and CHAR. Consider the following fragment:

Code: Select all
```
loop_
  _dataname_1
  _dataname_2
1.0 2.0
3.0 4.0
'5.0' 6.0
```

Under the above promotion rules, dataname_1 is promoted to CHAR type due to the delimiters around the final value, and so no values in that column should be interpreted as numbers. Is this what we want?

Re: CIF Data Model

by **jcbollinger** » Mon Feb 04, 2013 10:20 pm

Inasmuch as the CIF specifications do not restrict values in the same column to be drawn from a common domain, and given that there seems no viable, general method for type coercion between table values and values any other CIF2 data type, I think we cannot justify incorporating domain restrictions into the data model. Otherwise, there would be well-formed CIFs that the model cannot represent.

Moreover, in relational theory, as I understand it, the term "domain" refers not strictly to base data type, but rather to the allowed values within a base type, which may or may not comprise all the possible values of that base type. That sense of the term is relevant to CIF dictionary definitions, and at this point I'm inclined to think that it would be best to put the whole responsibility for value coercion and domain validation in one place. Although that place could, in principle, be implementations of the data model, I don't think that should be required by the data model definition.

I guess that means I prefer to leave domain restrictions as external, supplemental constraints imposed on the data by external authorities such as dictionaries.